OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide

Literature Number: SPRU761A
March 2004

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
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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

Preface

Read This First

About This Manual

This document describes the universal serial bus (USB) host on the OMAP5912 multimedia processor.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

OMAP5912 Multimedia Processor Device Overview and Architecture Reference Guide (literature number SPRU748) introduces the setup, components, and features of the OMAP5912 multimedia processor and provides a high-level view of the device architecture.

OMAP5912 Multimedia Processor OMAP 3.2 Subsystem Reference Guide (literature number SPRU749) introduces and briefly defines the main features of the OMAP3.2 subsystem of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor DSP Sybsystem Reference Guide (literature number SPRU750) describes the OMAP5912 multimedia processor DSP subsystem. The digital signal processor (DSP) subsystem is built around a core processor and peripherals that interface with: 1) The ARM926EJS via the microprocessor unit interface (MPUI); 2) Various standard memories via the external memory interface (EMIF); 3) Various system peripherals via the TI peripheral bus (TIPB) bridge.

OMAP5912 Multimedia Processor Clocks Reference Guide (literature number SPRU751) describes the clocking mechanisms of the OMAP5912 multimedia processor. In OMAP5912, various clocks are created from special components such as the digital phase locked loop (DPLL) and the analog phase-locked loop (APLL).

OMAP5912 Multimedia Processor Initialization Reference Guide (literature number SPRU752) describes the reset architecture, the configuration, the initialization, and the boot ROM of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Power Management Reference Guide (literature number SPRU753) describes power management in the OMAP5912 multimedia processor. The ultralow-power device (ULPD) generates and manages clocks and reset signals to OMAP3.2 and to some peripherals. It controls chip-level power-down modes and handles chip-level wake-up events. In deep sleep mode, this module is still active to monitor wake-up events.This book describes the ULPD module and outline architecture.

OMAP5912 Multimedia Processor Security Features Reference Guide (literature number SPRU754) describes the security features of teh OMAP5912 multimedia processor. The OMAP5912 security scheme relies on the OMAP3.2 secure mode. The distributed security on the OMAP3.2 platform is a Texas Instruments solution to address m-commerce and security issues within a mobile phone environment. The OMAP3.2 secure mode was developed to bring hardware robustness to the overall OMAP5912 security scheme.

OMAP5912 Multimedia Processor Direct Memory Access (DMA) Support Reference Guide (literature number SPRU755) describes the direct memory access support of the OMAP5912 multimedia processor. The OMAP5912 processor has three DMAs:

- The system DMA is embedded in OMAP3.2. It handles DMA transfers associated with MPU and shared peripherals.
- The DSP DMA is embedded in OMAP3.2. It handles DMA transfers associated with DSP peripherals.
- The generic distributed DMA (GDD) is an OMAP5912 resource attached to the SSI peripheral. It handles only DMA transfers associated with the SSI peripheral.

OMAP5912 Multimedia Processor Memory Interfaces Reference Guide (literature number SPRU756) describes the memory interfaces of the OMAP5912 multimedia processor.

- SDRAM (external memory interface fast, or EMIFF)
- Asynchronous and synchronous burst memory (external memory interface slow, or EMIFS)
- NAND flash (hardware controller or software controller)
- CompactFlash on EMIFS interface
- Internal static RAM

OMAP5912 Multimedia Processor Interrupts Reference Guide (literature number SPRU757) describes the interrupts of the OMAP5912 multimedia processor. Three level 2 interrupt controllers are used in OMAP5912:

- One MPU level 2 interrupt handler (also referred to as MPU interrupt level 2) is implemented outside of OMAP3.2 and can handle 128 interrupts.
- One DSP level 2 interrupt handler (also referred to as DSP interrupt level 2.1) is instantiated outside of OMAP3.2 and can handle 64 interrupts.
- One OMAP3.2 DSP level 2 interrupt handler (referenced as DSP interrupt level 2.0) can handle 16 interrupts.

OMAP5912 Multimedia Processor Peripheral Interconnects Reference Guide (literature number SPRU758) describes various periperal interconnects of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Timers Reference Guide (literature number SPRU759) describes various timers of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Serial Interfaces Reference Guide (literature number SPRU760) describes the serial interfaces of the OMAP5912 multimedia processor.

OMAP5912 Multimedia Processor Universal Serial Bus (USB) Reference Guide (literature number SPRU761) describes the universal serial bus (USB) host on the OMAP5912 multimedia processor. The OMAP5912 processor provides several varieties of USB functionality. Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

OMAP5912 Multimedia Processor Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU762) describes the three multi-channel buffered serial ports (McBSPs) available on the OMAP5912 device. The OMAP5912 device provides multiple high-speed multichannel buffered serial ports (McBSPs) that allow direct interface to codecs and other devices in a system.

OMAP5912 Multimedia Processor Camera Interface Reference Guide (literature number SPRU763) describes two camera inerfaces implemented in the OMAP5912 multimedia processor: compact serial camera port and camera parallel interface.

OMAP5912 Multimedia Processor Display Interface Reference Guide (literature number SPRU764) describes the display interface of the OMAP5912 multimedia processor.

- LCD module
- LCD data conversion module
- LED pulse generator
- Display interface

OMAP5912 Multimedia Processor Multimedia Card (MMC/SD/SDIO) (literature number SPRU765) describes the multimedia card (MMC) interface of the OMAP5912 multimedia processor. The multimedia card/secure data/secure digital IO (MMC/SD/SDIO) host controller provides an interface between a local host, such as a microprocessor unit (MPU) or digital signal processor (DSP), and either an MMC or SD memory card, plus up to four serial flash cards. The host controller handles MMC/SD/SDIO or serial port interface (SPI) transactions with minimal local host intervention.

OMAP5912 Multimedia Processor Keyboard Interface Reference Guide (literature number SPRU766) describes the keyboard interface of the OMAP5912 multimedia processor. The MPUIO module enables direct I/O communication between the MPU (through the public TIPB) and external devices. Two types of I/O can be used: specific I/Os dedicated for 8 x 8 keyboard connection, and general-purpose I/Os.

OMAP5912 Multimedia Processor General-Purpose Interface Reference Guide (literature number SPRU767) describes the general-purpose interface of the OMAP5912 multimedia processor. There are four GPIO modules in the OMAP5912. Each GPIO peripheral controls 16 dedicated pins configurable either as input or output for general purposes. Each pin has an independent control direction set by a programmable register. The two−edge control registers configure events (rising edge, falling edge, or both edges) on an input pin to trigger interrupts or wake−up requests (depending on the system mode). In addition, an interrupt mask register masks out specified pins. Finally, the GPIO peripherals provide the set and clear capabilities on the data output registers and the interrupt mask registers. After detection, all event sources are merged and a single synchronous interrupt (per module) is generated in active mode, whereas a unique wake−up line is issued in idle mode. Eight data output lines of the GPIO3 are ORed together to generate a global output line at the OMAP5912 boundary. This global output line can be used in conjunction with the SSI to provide a CMT−APE interface to the OMAP5912.

OMAP5912 Multimedia Processor VLYNQ Serial Communications Interface Reference Guide (literature number SPRU768) describes the VLYNQ of the OMAP5912 multimedia processor.

VLYNQ is a serial communications interface that enables the extension of an internal bus segment to one or more external physical devices. The external devices are mapped into local, physical address space and appear as if they are on the internal bus of the OMAP 5912. The external devices must also have a VLYNQ interface. The VLYNQ module serializes bus transactions in one device, transfers the serialized data between devices via a VLYNQ port, and de-serializes the transaction in the external device.

OMAP5912 includes one VLYNQ module connected on OCPT2 target port and OCPI initiator port. These connections are configured via a static switch, which selects either SSI or VLYNQ module. This switch, forbids the simultaneous use of GDD/SSI and VLYNQ. The switch is controlled by the VLYNQ_EN bit in the OMAP5912 configuration control register (CONF_5912_CTRL).

OMAP5912 Multimedia Processor Pinout Reference Guide (literature number SPRU769) provides the pinout of the OMAP5912 multimedia processor. After power-up reset, the user can change the configuration of the default interfaces. If another interface is available on top of the default, it is possible to enable a new interface for each ball by setting the corresponding 3-bit field of the associated FUNC_MUX_CTRL register. It is also possible to configure on-chip pullup/pulldown. This document also describes the various power domains so that the user can apply the different interfaces seamlessly with external components.

OMAP5912 Multimedia Processor Window Tracer (WT) Reference Guide (literature number SPRU770) describes the window tracer module used to capture the memory transactions from four interfaces: EMIFF, EMIFS, OCP-T1, and OCP-T2. This module is located in the OMAP3.2 traffic controller (TC).

OMAP5912 Multimedia Processor Real-Time Clock Reference Guide (literature number SPRUxxx) describes the real-time clock of the OMAP5912 multimedia processor. The real-time clock (RTC) block is an embedded real-time clock module directly accessible from the TIPB bus interface.

Trademarks

OMAP and the OMAP symbol are trademarks of Texas Instruments.

Contents

Figures

Tables

Universal Serial Bus (USB)

This document describes the universal serial bus (USB) host on the OMAP5912 multimedia processor.

1 Overview

The OMAP5912 processor provides several varieties of USB functionality, including:

❑ USB host: OMAP5912 provides a three-port *USB Specification Revision 1.1*-compliant host controller, which is based on the *OHCI Specification for USB Release 1.0a*.

❑ USB device: OMAP5912 provides a full-speed USB device.

❑ USB On-The-Go (OTG): OMAP5912 acts as an OTG dual-role device; the USB device functionality and one port of the USB host controller act in concert to provide an OTG port.

Flexible multiplexing of signals from the OMAP5912 USB host controller, the OMAP5912 USB function controller, and other OMAP5912 peripherals allow a wide variety of system-level USB capabilities. Many of the OMAP5912 pins can be used for USB-related signals or for signals from other OMAP5912 peripherals. The OMAP5912 top-level pin multiplexing controls each pin individually to select one of several possible internal pin signal interconnections. When these shared pins are programmed for use as USB signals, the OMAP5912 USB signal multiplexing selects how the signals associated with the three OMAP5912 USB host ports and the OMAP5912 USB function controller can be brought out to OMAP5912 pins.

2 USB Host Controller

The OMAP5912 USB host controller (HC) is a three-port controller that communicates with USB devices at the USB low-speed (1.5M bit-per-second maximum) and full-speed (12M bit-per-second maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 2.0* and the *Open HCI—Open Host Controller Interface Specification for USB,* Release 1.0a*,* available through the Compaq Computer Corporation web site, and hereafter called the *OHCI Specification for USB.* It is assumed that users of the OMAP5912 USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

The OMAP5912 OTG controller can use one of the USB host controller ports as part of a USB OTG-capable connection. When used for an OTG connection, the host controller port acts as the upstream device when OMAP5912 controls the OTG link, and the USB function controller acts as the downstream device when OMAP5912 acts as an OTG downstream device. The OMAP5912 OTG controller is described in Section 15.4, *USB OTG Controller*.

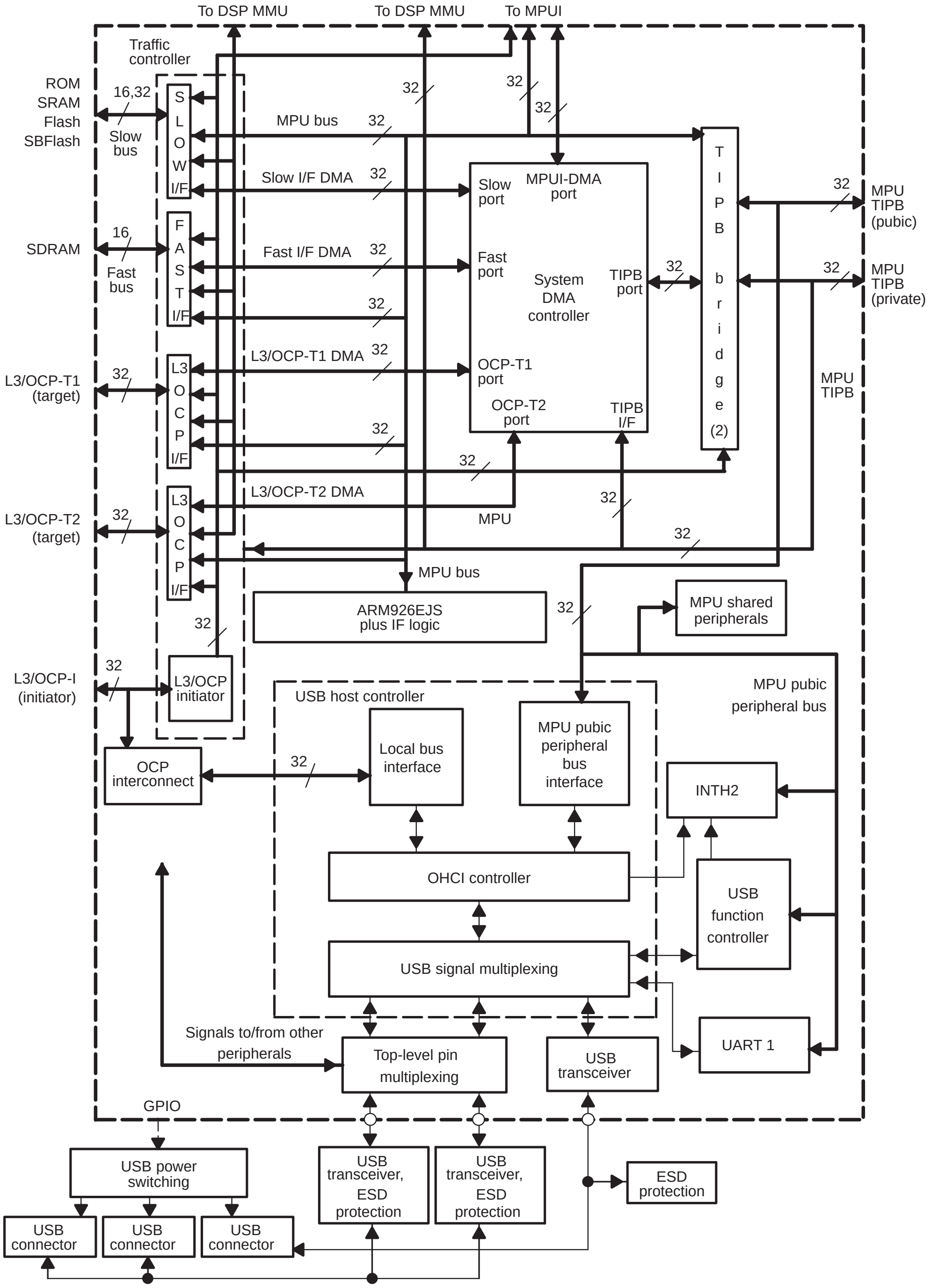
The OMAP5912 USB host controller implements the register set and makes use of the memory data structures defined in the *OHCI Specification for USB*. These registers and data structures are the mechanism by which a USB host controller driver software package can control the OMAP5912 USB host controller. The *OHCI Specification for USB* also defines how the USB host controller implementation must interact with those registers and data structures in system memory. The OMAP5912 MPU accesses these registers via the OMAP5912 MPU public peripheral bus.

To reduce processor software and interrupt overhead, the USB host controller generates USB traffic based on data structures and data buffers stored in system memory. The OMAP5912 USB host controller accesses these data structures without direct intervention by the processor using the OMAP5912 open-core protocol (OCPI) bus. These data structures and data buffers can be located in internal or external system RAM.

The USB host controller provides an interrupt to the MPU level 2 interrupt handler to signal certain hardware events to the host controller driver software.

Figure 1 shows the OMAP5912 USB host controller.

Figure 1. *USB Host Controller*



2.1 USB Open Host Controller Interface Functionality

2.1.1 OHCI Controller Overview

The *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, defines a set of registers and data structures stored in system memory that define how a USB host controller interfaces to system software. This specification, in conjunction with the *Universal Serial Bus Specification Version 2.0*, defines most of the USB functionality that the OMAP5912 USB host controller provides.

The *OHCI Specification for USB* focuses on two main aspects of the hardware implementation of a USB host controller: its register set and the memory data structures that define the activity to appear on the USB bus. Also discussed are issues such as interrupt generation, USB host controller state, USB frame management, and the methods that the hardware must use to process the lists of data structures in system memory.

This document does not duplicate the information presented in the *OHCI Specification for USB* or the *USB Specification*. OMAP5912 USB host controller users can refer to the *USB Specification* and the *OHCI Specification for USB* for detailed discussions of USB requirements and OHCI controller operation.

2.2 OMAP5912 USB Host Controller Differences From OHCI Specification for USB

The OMAP5912 USB host controller implementation does not implement every aspect of the functionality defined in the *OHCI Specification for USB*. The differences focus on power switching, overcurrent reporting, and the OHCI ownership change interrupt. Other restrictions are imposed by the effects of OMAP5912 pin multiplexing options.

2.2.1 Power Switching Output Pins Not Supported

The OMAP5912 device does not provide pins that can be controlled directly by the USB host controller OHCI port power control features. The OHCI RHPORTSTATUS register port power control bits can be programmed by the USB host controller driver software, but this does not have any direct effect on any VBUS switching implemented on the board.

Users can use software control of GPIO pins or other implementation-specific control mechanisms to control VBUS switching.

2.2.2 Overcurrent Protection Input Pins Not Supported

The OMAP5912 device does not provide any pins that allow the USB host controller OHCI RHPORTSTATUS overcurrent protection status bits to be directly controlled by external hardware.

Users can use software monitoring of GPIO pins or other implementation-specific control mechanisms to report port overcurrent information to the USB host controller driver.

2.2.3 HMC_MODE and Top-Level Pin Multiplexing and OHCI Registers

The USB signal multiplexing modes selected by HMC_MODE provide selections where 0, 1, 2, or 3 USB host controller ports can be brought to OMAP5912 pins. The OHCI RHDESCRIPTORA register always reports three available USB host ports, regardless of the HMC_MODE setting (see Table 20) or top-level pin multiplexing settings. When the HMC_MODE setting disables a USB host controller port, the USB host controller sees that port as unattached.

When OMAP5912 top-level pin multiplexing configures a pin for functionality other than the USB, the USB host controller is disconnected from that pin and that pin does not affect the USB host controller.

2.2.4 No Ownership Change Interrupt

The OMAP5912 USB host controller does not implement the OHCI ownership change interrupt.

2.3 OMAP5912 Implementation of *OHCI Specification for USB*

2.3.1 Isochronous Transmit Descriptor (TD) OFFSETX/PSWX Values

The OMAP5912 USB host controller implements the *OHCI Specification for USB* optional feature of checking isochronous OFFSETX/PSWX values. If either OFFSETX or OFFSET(X+1) does not have a condition code of *Not Accessed*, or if the OFFSET(X+1) value is not greater than or equal to OFFSETX, then an unrecoverable error is reported. Unrecoverable errors issued for these reasons do not cause an update of the HOSTUEADDR, HOSTUESTATUS, or HOSTTIMEOUTCTRL registers.

2.3.2 OMAP5912 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The OHCI *Specification for USB* provides a specific sequence of operations for the host controller driver to perform when setting up the host controller. Failure to follow that sequence can result in malfunction. As a specific example, the HCCONTROLHEADED and HCBULKHEADED pointer registers and the 32 HCCAINTERRUPTTABLE pointers must all point to valid physical addresses of valid endpoint descriptors.

The OMAP5912 USB host controller does not check HCCONTROLHEADED registers, HCBULKHEADED registers, or the values in the 32 HCCAINTERRUPTTABLE pointers before using them to access EDs. If any of these pointers are NULL when the corresponding list enable bit is set, the OMAP5912 USB host controller attempts to access using the physical address of 0, which causes an unrecoverable error to be signaled. HOSTUEADDR, HOSTUESTATUS, and HOSTTIMEOUTCTRL registers are updated in this case.

2.3.3 OHCI USB Suspend State

The OMAP5912 USB host controller ignores upstream traffic from downstream devices for about 3 ms after the host controller state (HCCONTROL.HCFS) changes from USB resume state to USB operational state. If any TDs cause generation of downstream packets during that time, the downstream packets are sent, but any response provided by the downstream device is ignored. Any such TDs are aborted with completion codes marked as *Device Not Responding*. TDs on any of the lists (periodic, control, bulk, and isochronous) can cause such an occurrence.

The USB specification requires that system software must provide a 10-ms resume recovery time (T_{RSMRCY}) after a bus segment transitions from resume signaling to normal operational mode. During that time, only start of frame packets are to be sent on the bus segment. It is recommended that system software disable all list enable bits (HCCONTROL.PLE, HCCONTROL.IE, HCCONTROL.CLE, and HCCONTROL.BLE) and then wait for at least 1 ms before setting the host controller into USB suspend state (via HCCONTROL.HCFS). When restoring from suspend, system software must set the host controller into USB resume state, and wait for the host controller to transition into USB operational state. System software must then wait 10 ms before enabling the host controller list enable bits.

When the host controller has been placed into the USB suspend state under software control, but is brought out by a remote wake-up, system software must monitor the HCRHPORTSTATUS[x].PSS and HCRHPORTSTATUS[x].PSSC bits. The HCRHPORTSTATUS[x].PSS bit changes to 0 only after completion of resume signaling on the bus segment completes and completion of the 3-ms period where packets from downstream devices are ignored.

When using port-specific suspend, it is not necessary to disable the host controller lists so long as there are no active EDs and TDs directed toward devices that are downstream of the suspended port. For port-specific suspend operations, the host controller does not issue a root hub status change interrupt with the HCRHPORTSTATUS[n].PSSC bit = 1 and

HCRHPORTSTATUS[n].PSS = 0 until after the approximately 3-ms delay after resume signaling completes.

When using port-specific suspend, system software must ensure that there are no active EDs for devices that are downstream of the suspended port before setting the port into suspend mode. While the port is in suspend or being resumed, system software must not enable any EDs for any devices downstream of the suspended port. Once the root hub status change interrupt

occurs as a result of the suspended port PSS bit changing to 0, EDs can be enabled for devices downstream of the port that is now operational.

2.4 USB Host Controller Registers

Most of the OMAP5912 host controller (HC) registers are the OHCI operational registers, which are defined by the *OHCI Specification for USB*. Four additional registers not specified by the *OHCI Specification for USB* provide additional information about the USB host controller state. USB host controller registers can be accessed in user and supervisor modes.

The OMAP5912 USB host controller registers are listed in Table 1. Table 2 through Table 29 describe specific register bits.

Table 1. USB Host Controller Registers

Name	Description	R/W	Size†	Address
HCREVISION	OHCI revision number	R	32	FFFB:A000h
HCCONTROL	HC operating mode	R/W	32	FFFB:A004h
HCCOMMANDSTATUS	HC command and status	R/W	32	FFFB:A008h
HCINTERRUPTSTATUS	HC interrupt status	R/W	32	FFFB:A00Ch
HCINTERRUPTENABLE	HC interrupt enable	R/W	32	FFFB:A010h
HCINTERRUPTDISABLE	HC interrupt disable	R/W	32	FFFB:A014h
HCHCCA	Physical address of HCCA‡	R/W	32	FFFB:A018h

† Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.

‡ Restrictions apply to the physical addresses used in these registers. See Section 2.9, *Physical Addressing*.

§ This register provides control and status for the OMAP5912 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.

¶ This register provides control and status for the OMAP5912 pins normally associated with USB port 1 for some HMC_MODE values.

# This register provides control and status for the OMAP5912 pins normally associated with USB port 2 for some HMC_MODE values.

Table 1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size[†]	Address
HCPERIODCURRENTED	Physical address of current periodic endpoint descriptor[‡]	R/W	32	FFFB:A01Ch
HCCONTROLHEADED	Physical address of head of control endpoint descriptor list[‡]	R/W	32	FFFB:A020h
HCCONTROLCURRENTE D	Physical address of current control endpoint descriptor[‡]	R/W	32	FFFB:A024h
HCBULKHEADED	Physical address of head of bulk endpoint descriptor list[‡]	R/W	32	FFFB:A028h
HCBULKCURRENTED	Physical of current bulk endpoint descriptor[‡]	R/W	32	FFFB:A02Ch
HCDONEHEAD	Physical address of head of list of retired transfer descriptors[‡]	R	32	FFFB:A030h
HCFMINTERVAL	HC frame interval	R/W	32	FFFB:A034h
HCFMREMAINING	HC frame remaining	R	32	FFFB:A038h
HCFMNUMBER	HC frame number	R	32	FFFB:A03Ch
HCPERIODICSTART	HC periodic start	R/W	32	FFFB:A040h
HCLSTHRESHOLD	HC low speed threshold	R/W	32	FFFB:A044h
HCRHDESCRIPTORA	HC root hub A	R, R/W	32	FFFB:A048h
HCRHDESCRIPTORB	HC root hub B	R/W	32	FFFB:A04Ch
HCRHSTATUS	HC root hub status	R, R/W	32	FFFB:A050h
HCRHPORTSTATUS1	HC port 1 control and status[§]	R, R/W	32	FFFB:A054h
HCRHPORTSTATUS2	HC port 2 control and status[¶]	R, R/W	32	FFFB:A058h
HCRHPORTSTATUS3	HC port 3 control and status[#]	R, R/W	32	FFFBA05Ch

[†] Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
[‡] Restrictions apply to the physical addresses used in these registers. See Section 2.9, *Physical Addressing*.
[§] This register provides control and status for the OMAP5912 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.
[¶] This register provides control and status for the OMAP5912 pins normally associated with USB port 1 for some HMC_MODE values.
[#] This register provides control and status for the OMAP5912 pins normally associated with USB port 2 for some HMC_MODE values.

Table 1. USB Host Controller Registers (Continued)

Name	Description	R/W	Size†	Address
Reserved	Reserved	None		FFFB:A060h to FFFB:A0DFh
HOSTUEADDR	Host UE address	R	32	FFFB:A0E0h
HOSTUESTATUS	Host UE status	R	32	FFFB:A0E4h
HOSTTIMEOUTCTRL	Host time-out control	R/W	32	FFFB:A0E8h
HOSTREVISION	Host revision	R	32	FFFB:A0ECh
Reserved	Reserved	None		FFFB:A0F0h to FFFB:AFFFh

[†] Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
[‡] Restrictions apply to the physical addresses used in these registers. See Section 2.9, *Physical Addressing*.
[§] This register provides control and status for the OMAP5912 pins normally associated with USB port 0 (the integrated USB transceiver) for some HMC_MODE values.
[¶] This register provides control and status for the OMAP5912 pins normally associated with USB port 1 for some HMC_MODE values.
[#] This register provides control and status for the OMAP5912 pins normally associated with USB port 2 for some HMC_MODE values.

The OCHI revision number (Table 2) register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

Table 2. OHCI Revision Number Register (HCREVISION)

Bit	Name	Description	Type	Reset
31:8	Reserved	Reserved		0x00 0000
7:0	REV	OHCI specification revision—the OHCI revision number upon which the USB host controller is based. Write has no effect.	R	0x10

The HC operating mode register (Table 3) controls the operating mode of the USB host controller.

Table 3. HC Operating Mode Register (HCCONTROL)

Bit	Name	Value	Description	Type	Reset
31:11	Reserved		Reserved		
10	RWE		Remote wake-up enable. This bit has no effect in OMAP5912. The OMAP5912 USB host controller does not provide a processor wake-up mechanism.	R/W	0

Table 3. HC Operating Mode Register (HCCONTROL)(Continued)

Bit	Name	Value	Description	Type	Reset
9	RWC		Remote wake-up connected. This bit has no effect in OMAP5912. The OMAP5912 USB host controller does not provide a processor wake-up mechanism.	R/W	0
8	IR		Interrupt routing. The OMAP5912 USB host controller does not provide an SMI interrupt. This bit must be 0 to allow the USB host controller interrupt to propagate to the MPU level 2 interrupt controller.	R/W	0
7:6	HCFS		Host controller functional state:	R/W	00
		00	USB reset		
		01	USB resume		
		10	USB operational		
		11	USB suspend		
			A transition to USB operational causes SOF generation to begin in 1 ms. The USB host controller can automatically transition from USB suspend to USB resume if a downstream resume is received. The USB host controller enters USB suspend after a software reset. The USB host controller enters USB reset after a hardware reset. The USB reset state resets the root hub and causes downstream signaling of USB reset.		
5	BLE		Bulk list enable:	R/W	0
		0	Bulk ED list not processed in the next 1-ms frame. Host controller driver can modify the list. If driver removes the ED pointed to by the HCBULKCURRENTED from the ED list, it must update HCBULKCURRENTED to point to an ED still on the list before it re-enables the bulk list.		
		1	Enables processing of bulk ED list. HCBULKHEADED must be 0 or point to a valid ED before setting this bit. HCBULKCURRENTED must point to a valid ED or be 0 before setting this bit.		
4	CLE		Control list enable:	R/W	0

Table 3. HC Operating Mode Register (HCCONTROL)(Continued)

Bit	Name	Value	Description	Type	Reset
		0	Control ED list is not processed in the next 1-ms frame. Host controller driver can modify the control ED list. If driver removes the ED pointed to by the HCCONTROLCURRENTED from the ED list, it must update HCCONTROLCURRENTED to point to an ED still on the list before it re-enables the control list.		
		1	Enables processing of the control ED list. HCCONTROLHEADED must be 0 or point to a valid ED before setting this bit. HCCONTROLCURRENTED must be 0 or point to a valid ED before setting this bit.		
3	IE		Isochronous enable	R/W	0
		0	Isochronous EDs are not processed. The USB host controller checks this bit every time it finds an isochronous ED in the periodic list. When this bit is written to 1, processing of isochronous EDs can be enabled in the next frame, if not in the current frame.		
		1	Enables processing of isochronous EDs		
2	PLE		Periodic list enable	R/W	0
		0	The periodic ED lists are not processed. When written to 0, periodic list processing is disabled beginning with the next frame.		
		1	Enables processing of the periodic ED lists. When written to 1, periodic list processing begins in the next frame.		
1:0	CBSR		Control/bulk service ratio Specifies the ratio between control and bulk EDs processed in a frame	R/W	00
		00	One control ED per bulk ED		
		01	Two control EDs per bulk ED		
		10	Three control EDs per bulk ED		
		11	Four control EDs per bulk ED		

The HC command and status register shows the current state of the host controller and accepts commands from the host controller driver.

Table 4. HC Command and Status Register (HCCOMMANDSTATUS)

Bit	Name	Description	Type	Reset
31:18	Reserved	Reserved		
17:16	SOC	Scheduling overrun count Counts the number of times a scheduling overrun occurs. This count is incremented even if the host controller driver has not acknowledged any previous pending scheduling overrun interrupt.	R	00
15:4	Reserved	Reserved		
3	OCR	Ownership change request This bit is set by the host controller driver to gain ownership of the host controller. OMAP5912 does not support SMI interrupts, so no ownership change interrupt occurs.	R/W	0
2	BLF	Bulk list filled The host controller driver must set this bit if it modifies the bulk list to include new TDs. If HCBULKCURRENTED is 0, the USB host controller does not begin processing bulk list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the bulk list, it clears this bit.	R/W	0
1	CLF	Control list filled The host controller driver must set this bit if it modifies the control list to include new TDs. If HCCONTROLHEADED is 0, the USB host controller does not begin processing control list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the control list, it clears this bit.	R/W	0
0	HCR	Host controller reset Write of 0 has no effect. 1: This bit initiates a software reset of the USB host controller. This transitions the USB host controller to the USB suspend state. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this register returns a 0. A write of 1 to this bit does not reset the root hub and does not signal USB reset to downstream USB functions.	R/W	0

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

Table 5. HC Interrupt and Status Register (HCINTERRUPTSTATUS)

Bit	Name	Description	Type	Reset
31	Reserved	Reserved		
30	OC	Ownership change The OMAP5912 USB host controller does not implement ownership change interrupts.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change When 1, indicates a root hub status change has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
5	FNO	Frame number overflow When 1, indicates a frame number overflow has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
4	UE	Unrecoverable error When 1, indicates that an unrecoverable error has occurred on the OCPI bus or that an isochronous TD PSW field condition code was not set to *Not Accessed* when the USB host controller attempted to perform a transfer using that PSW/offset pair. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
3	RD	Resume detected When 1, indicates that a downstream device has issued a resume request. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
2	SF	Start of frame When 1, indicates that a SOF has been issued. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

Table 5. HC Interrupt and Status Register (HCINTERRUPTSTATUS) (Continued)

Bit	Name	Description	Type	Reset
1	WDH	Write done head When 1, indicates that the USB host controller has updatedthe HCDONEHEAD register. Write of 0 has no effect. Write of 1 clears this bit. The host controller driver must read the value from HCDONEHEAD before writing 1 to this bit.	R/W	0
0	SO	Scheduling overrun When 1, indicates that a scheduling overrun has occurred. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0

The HC interrupt enable register (Table 6) enables various OHCI interrupt sources to generate interrupts to the OMAP5912 level 2 interrupt handler.

Table 6. HC Interrupt Enable Register (HCINTERRUPTENABLE)

Bit	Name	Description	Type	Reset
31	MIE	Master interrupt enable When 1, allows other enabled OHCI interrupt sources to propagate to the OMAP5912 level 2 interrupt controller. When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the OMAP5912 level 2 interrupt controller. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP5912.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change When 1 and MIE is 1, allows root hub status change interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, root hub status change interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

Table 6. HC Interrupt Enable Register (HCINTERRUPTENABLE) (Continued)

Bit	Name	Description	Type	Reset
5	FNO	Frame number overflow When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, frame number overflow interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W:	0
4	UE	Unrecoverable error When 1 and MIE is 1, allows unrecoverable error interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, unrecoverable error interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
3	RD	Resume detected When 1 and MIE is 1, allows resume detected interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, resume detected interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
2	SF	Start of frame When 1 and MIE is 1, allows start of frame interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, start of frame interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

Table 6. HC Interrupt Enable Register (HCINTERRUPTENABLE) (Continued)

Bit	Name	Description	Type	Reset
1	WDH	Write done head When 1 and MIE is 1, allows write done head interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, write done head interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0
0	SO	Scheduling overrun When 1 and MIE is 1, allows scheduling overrun interrupts to propagate to the OMAP5912 level 2 interrupt controller. When 0, or when MIE is 0, scheduling overrun interrupts do not propagate. A write of 0 has no effect on this bit. A write of 1 sets this bit.	R/W	0

The HC interrupt disable register is used to clear bits in the HCINTERRUPTENABLE register.

Table 7. HC Interrupt Disable Register (HCINTERRUPTDISABLE)

Bit	Name	Description	Type	Reset
31	MIE	Master interrupt enable Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE MIE bit.	R/W	0
30	OC	Ownership change This bit has no effect on OMAP5912.	R	0
29:7	Reserved	Reserved		
6	RHSC	Root hub status change Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RHSC bit.	R/W	0

Table 7. HC Interrupt Disable Register (HCINTERRUPTDISABLE) (Continued)

Bit	Name	Description	Type	Reset
5	FNO	Frame number overflow Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE FNO bit.	R/W	0
4	UE	Unrecoverable error Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE UE bit.	R/W	0
3	RD	Resume detected Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE RD bit.	R/W	0
2	SF	Start of frame Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SF bit.	R/W	0
1	WDH	Write done head Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE WDH bit.	R/W	0
0	SO	Scheduling overrun Read always returns 0. Write of 0 has no effect. Write of 1 clears the HCINTERRUPTENABLE SO bit.	R/W	0

The HCAA address register defines the physical address of the beginning of the HCCA.

Table 8. HC HCAA Address Register (HCHCCA)

Bit	Name	Description	Type	Reset
31:8	HCCA	Physical address of the beginning of the HCCA	R/W	0x000000
7:0	Reserved	Reserved	R	x00

The HC current periodic register defines the physical address of the next endpoint descriptor (ED) on the periodic ED List.

Table 9. HC Current Periodic Register (HCPERIODCURRENTED)

Bit	Name	Description	Type	Reset
31:4	PCED	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses.	R	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC head control register defines the physical address of the head ED of the control ED list.

Table 10. HC Head Control Register (HCCONTROLHEADED)

Bit	Name	Description	Type	Reset
31:4	CHED	Physical address of head ED on the control ED list This field represents bits 31:4 of the physical address of the head ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The HC current control register defines the physical address of the next ED on the control ED list.

Table 11. HC Current Control Register (HCCONTROLCURRENTED)

Bit	Name	Description	Type	Reset
31:4	CCED	Physical address of current ED on the control ED list This field represents bits 31:4 of the physical address of the next ED on the control ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the control ED list without finding any transfers to process. This register is automatically updated by the USB host controller.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The head bulk register defines the physical address of the head ED on the bulk ED list.

Table 12. HC Head Bulk Register (HCBULKHEADED)

Bit	Name	Description	Type	Reset
31:4	BHED	Physical address of head ED on the bulk ED list This field represents bits 31:4 of the physical address of the head ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The current bulk register defines the physical address of the next ED on the bulk ED list.

Table 13. HC Current Bulk Register (HCBULKCURRENTED)

Bit	Name	Description	Type	Reset
31:4	BCED	Physical address of current ED on the bulk ED list This field represents bits 31:4 of the physical address of the next ED on the bulk ED list. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 2.9, *Physical Addressing*, for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the bulk ED list without finding any transfers to process. The USB host controller automatically updates this register.	R/W	0x0000000
3:0	Reserved	Reserved	R	0x0

The head done register defines the physical address of the current head of the done TD queue.

Table 14. HC Head Done Register (HCDONEHEAD)

Bit	Name	Description	Type	Reset
31:4	DH	Physical address of the last TD that has added to the done queue. This field represents bits 31:4 of the physical address of the top TD on the done TD queue. TDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. A value of 0x00000000 indicates that there are no TDs on the done queue. The USB host controller automatically updates this register.	R	0x0000000
3:0	Reserved	Reserved	R	0x0

The frame interval register defines the number of 12-MHz clock pulses in each USB frame.

Table 15. HC Frame Interval Register (HCFMINTERVAL)

Bit	Name	Description	Type	Reset
31	FIT	Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.	R/W	0
30:16	FSMPS	Largest data packet Largest data packet size allowed for full-speed packets, in bit times.	R/W	0x0000
15:14	Reserved	Reserved		
13:0	FI	Frame interval Number of 12-MHz clocks in the USB frame. Nominally, this is set to 11,999, to give a 1-ms frame. The host controller driver can make minor changes to this field to attempt to manually synchronize with another clock source.	R/W	0x2EDF

The HC frame remaining register reports the number of full-speed bit times remaining in the current frame.

Table 16. HC Frame Remaining Register (HCFMREMAINING)

Bit	Name	Description	Type	Reset
31	FRT	Frame remaining toggle This bit is loaded with the frame interval toggle bit every time the USB host controller loads the frame interval field into the frame remaining field.	R	0
30:14	Reserved	Reserved		
13:0	FR	Frame remaining The number of full-speed bit times remaining in the current frame. This field is automatically reloaded with the frame interval field value at the beginning of every frame.	R	0x0000

The HC frame number register reports the current USB frame number.

Table 17. HC Frame Number Register (HCFMNUMBER)

Bit	Name	Description	Type	Reset
31:16	Reserved	Reserved		
15 :0	FN	Frame number This field reports the current USB frame number. It is incremented when the frame remaining field is reloaded with the frame interval field value. Frame number automatically rolls over from 0xFFFF to 0x0000. After frame number is incremented, its new value is written to the HCCA and the USB host controller sets the SOF interrupt status bit and begins processing the ED lists.	R	0x0000

The HC periodic start register defines the position within the USB frame where EDs on the periodic list have priority over EDs on the bulk and control lists.

Table 18. HC Periodic Start Register (HCPERIODICSTART)

Bit	Name	Description	Type	Reset
31:14	Reserved	Reserved		
13:0	PS	Periodic start The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.	R/W	0x0000

The HC low-speed threshold register defines the latest time in a frame that the USB host controller can begin a low-speed packet.

Table 19. HC Low-Speed Threshold Register (HCLSTHRESHOLD)

Bit	Name	Description	Type	Reset
31:14	Reserved	Reserved		
13:0	LST	Low-speed threshold This field defines the number of full-speed bit times in the frame after which the USB host controller cannot start an 8-byte low-speed packet. The USB host controller only begins a low-speed transaction if the frame remaining field is greater than the low-speed threshold. The host controller driver must set this field to a value that ensures that an 8-byte low-speed TD completes before the end of the frame. When set, the host controller driver must not change the value.	R/W	0x0628

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

Table 20. HC Root Hub A Register (HCRHDESCRIPTORA)

Bit	Name	Value	Description	Type	Reset
31:24	POTPG		Power-on to power-good time This field defines the minimum amount of time (2 ms × POTPG) between the USB host controller turning on power to a downstream port and when the USB host can access the downstream device. This field has no effect on USB host controller operation. After turning on power to a port, the USB host controller driver must delay the amount of time implied by POTPG before attempting to reset an attached downstream device. The required amount of time is implementation-specific and must be calculated based on the amount of time the VBUS supply takes to provide valid VBUS to a worst-case downstream USB function controller. The implementation-specific value must be computed and then written to this register before the USB host controller driver is initialized. Because OMAP5912 does not provide a direct control from the USB host controller to switch VBUS on and off, this value must take into account any delays caused by other methods of controlling VBUS externally. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0xA
23:13	Reserved		Reserved		
12	NOCP		No overcurrent protection When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. OMAP5912 does not provide signals to allow connection of external overcurrent indication signals to the USB host controller, so this bit defaults to 1. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	1

Table 20. HC Root Hub A Register (HCRHDESCRIPTORA) (Continued)

Bit	Name	Value	Description	Type	Reset
11	OCPM		Overcurrent protection mode	R/W	0
			OMAP5912 does not provide host controller overcurrent protection input signals, so this bit has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.		
10	DT		Device type	R	0
			This bit is always 0, which indicates that the USB host controller implemented is not a compound device.		
9	NPS		No power switching	R/W	1
		0	Indicates that VBUS power switching is supported and is either per-port or all-port switched per the power switching mode field.		
		1	Indicates that VBUS power switching is not supported and that power is available to all downstream ports when the USB host controller is powered.		
			Because OMAP5912 does not provide connections from the USB host controller to control external VBUS switching, this bit defaults to 1.		
			This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.		
8	PSM		Power switching mode	R/W	0
		0	Indicates that all ports are powered at the same time		
		1	Indicates that individual port power switching is supported		

Table 20. HC Root Hub A Register (HCRHDESCRIPTORA) (Continued)

Bit	Name	Value	Description	Type	Reset
			Because OMAP5912 does not provide signals from the USB host controller to control external VBUS switching, this bit defaults to 0.		
7:0	NDP		Number of downstream ports This register defaults to 3 to indicate three downstream ports. The USB signal multiplexing mode and OMAP5912 top-level pin multiplexing features can place the OMAP5912 device in a mode where 0, 1, 2, or 3 of the USB host controller downstream ports are usable. This register reports three ports, regardless of USB signal multiplexing mode and OMAP5912 top-level pin multiplexing mode.	R	0x03

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

Table 21. HC Root Hub B Register (HCRHDESCRIPTORB)

Bit	Name	Description	Type	Reset
31:16	PPCM	Port power control mask Each bit defines whether a corresponding downstream port has port power controlled by the global power control. If set, per-port power control is implemented for the corresponding port. If clear, global power control is implemented for the corresponding port. PPCM bit 0 is reserved. PPCM bit 1 is the port power control mask for downstream port 1. PPCM bit 2 is the port power control mask for downstream port 2. PPCM bit 3 is the port power control mask for downstream port 3. PPCM bits 4 through 15 are reserved. OMAP5912 does not provide connections from the USB host controller to pins to provide external port power switching. Systems that implement port power switching must use other mechanisms to control port power. System software can update these bits to simplify host controller driver and/or OTG driver coding.	R/W	0x0000
15 :0	DR	Device removable Each bit defines whether a corresponding downstream port has a removable or non-removable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a non-removable device attached. DR bit 0 is reserved. DR bit 1 is the device removable bit for downstream port 1. DR bit 2 is the device removable bit for downstream port 2. DR bit 3 is the device removable bit for downstream port 3. DR bits 4 through 15 are reserved.	R/W	0x0000

The HC root hub status register reports the USB host controller root hub status.

Table 22. HC Root Hub Status Register (HCRHSTATUS)

Bit	Name	Description	Type	Reset
31	CRWE	Clear remote wake-up enable Write of 0 has no effect. Write of 1 clears the device remote wake-up enable bit.	R/W	0
30:18	Reserved	Reserved		
17	OCIC	Overcurrent indication change This bit is automatically set when the overcurrent indicator bit changes. Write of 0 has no effect. Write of 1 clears this bit. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0
16	LPSC	Local power status change This bit defaults to 0 because the root hub does not support the local power status feature. Write of 0 has no effect. Write of 1 sets the port power status bits for all ports if power switching mode is 0. A write of 1 sets port power status bits for ports with their corresponding port power control mask bits cleared if power switching mode is 1. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0
15	DRWE	Device remote wake-up enable When 1, this bit enables a connect status change event to be treated as a resume event, which causes a transition from USB suspend to USB resume state and sets the resume detected interrupt status bit. When 0, connect status change events do not cause a transition from USB suspend to USB resume state and the resume detected interrupt is not changed. Write of 0 has no effect. Write of 1 sets the device remote wake-up enable bit.	R/W	0
14:2	Reserved	Reserved		

Table 22. HC Root Hub Status Register (HCRHSTATUS) (Continued)

Bit	Name	Description	Type	Reset
1	OCI	Overcurrent indicator This bit reports global overcurrent indication if global overcurrent reporting is selected. When 1, this bit indicates that an overcurrent condition has been sensed. When 0, no overcurrent condition has been sensed. Because OMAP5912 does not provide signals for external hardware to report overcurrent status to the USB host controller, this bit is always 0. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R	0
0	LPS	Local power status The root hub does not support the local power status feature. This bit always reads as 0. Write of 0 has no effect. Write of 1 when in global power mode (power switching mode = 0), turns off power to all ports. If in per-port power mode (power switching mode = 1), a write of 1 turns off power to those ports whose corresponding port power control mask bit is 0. Because OMAP5912 does not provide signals from the USB host controller to external VBUS switching circuitry, this bit has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

The HC port 1 status and control register reports and controls the state of USB host port 1. HCRHPORTSTATUS1 can provide status and control for the host controller port that is usually associated with the OMAP5912 integrated USB transceiver and the associated OMAP5912 pins:

❑ USB.DP

❑ USB.DM

Alternately, OMAP5912 top-level pin multiplexing can also configure the host controller port associated with this register to provide status and control for a USB host port connected to the following pins:

❑ MCSI2.DOUT/USB0.TXEN
❑ UART2.TX/USB0.TXD
❑ UART2.RTS/USB0.SE0
❑ UART2.CTS/USB0.RCV.
❑ MCSI2.DIN/USB0.VP
❑ UART2.RX/USB0.VM
❑ MCSI2.SYNC/USB0.SPEED
❑ MCSI2.CLK/USB0.SUSP

The host controller port associated with this register can also be held in a state where it always appears to be disconnected, depending on the HMC_MODE value and top-level pin multiplexing. See Table 22 and 4.3, *Pin Multiplexing*.

HC port 1 can act as the host portion of an OTG controller. See Section 4, *USB OTG Controller*.

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1)

Bit	Name	Description	Type	Reset
31:21	Reserved	Reserved		
20	PRSC	Port 1 reset status change This bit indicates, when 1, that the port 1 port reset status bit has changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
19	OCIC	Port 1 overcurrent indicator change This bit indicates, when 1, that the port 1 port overcurrent indicator has changed. Write of 0 has no effect. Write of 1 clears this bit. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset
18	PSSC	Port 1 suspend status change This bit indicates, when 1, that the port 1 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
17	PESC	Port 1 enable status change This bit indicates, when 1, that the port 1 port enable status changed. Write of 0 has no effect. Write of 1 clears this bit.	R/W	0
16	CSC	Port 1 connect status change This bit indicates, when 1, that the port 1 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable USB device on port 1, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 1 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 1. A 0 in this bit indicates a full-speed device. This bit is valid only when port 1 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 1 port power status. A write of 0 to this bit has no effect. OMAP5912 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset
8	PPS/SPP	Port 1 port power status/set port power This bit indicates, when read as 1, that the port 1 power is enabled. When read as 0, port 1 power is not enabled. The OMAP5912 does not provide signals from the USB host controller to control external port power, so if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding. A write of 1 to this bit sets the port 1 port power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 1 port reset status/set port reset When read as 1, indicates that port 1 is signalling the USB reset. When read as 0, USB reset is not being sent to port 1. A write of 1 to this bit sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 1 port overcurrent indicator/clear suspend status When read as 1, indicates a port 1 port overcurrent condition has occurred. When 0, no port 1 port overcurrent condition has occurred. OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 1 port suspend status is 1 causes resume signaling on port 1. A write of 1 when port 1 port suspend status is 0 has no effect. A write of 0 has no effect.	R/W	0

Table 23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) (Continued)

Bit	Name	Description	Type	Reset
2	PSS/SPS	Port 1 port suspend status/set port suspend When read as 1, indicates that port 1 is in the USB suspend state or is in the resume sequence. When 0, indicates that port 1 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 1 current connect status is 1, a write of 1 to this bit sets the port 1 port suspend status bit and places port 1 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 1 port enable status/set port enable When read as 1, indicates that port 1 is enabled. When read as 0, this bit indicates that port 1 is not enabled. This bit is automatically set at completion of port 1 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. A write of 1 to this bit when port 1 current connect status is 1 sets the port 1 port enable status bit. A write of 1 when port 1 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 1 current connection status/clear port enable When read as 1, indicates that port 1 currently has a USB device attached. When 0, indicates that no USB device is attached to port 1. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable device on port 1. A write of 1 to this bit clears the port 1 port enable bit. A write of 0 to this bit has no effect.	R/W	0

The HC port 2 status register reports and controls the state of USB host port 2. Depending on HMC_MODE (see Table 23) value, HCRHPORTSTATUS1 can provide status and control for the host controller port that is usually associated with the following OMAP5912 pins:

- ❑ MCBSP.CLKX/USB1.TXEN
- ❑ MCSI1.DOUT/USB1.TXD
- ❑ RST_HOST_OUT/USB1.SE0
- ❑ MCSI1.DIN/USB1.RCV
- ❑ MCSI1.SYNC/USB1.VP
- ❑ MCSI1.CLK/USB1.VM
- ❑ CLK32K_OUT/USB1.SPEED
- ❑ MPU_BOOT/USB1.SUSP

The host controller port associated with this register can also be held in a state where it always appears disconnected, depending on the HMC_MODE value and top-level pin multiplexing (see Section 4.3, *Pin Multiplexing*).

HC port 2 can act as the host portion of an OTG controller. See Section 4, *USB OTG Controller*.

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2)

Bit	Name	Description	Type	Reset
31:21	Reserved	Reserved		
20	PRSC	Port 2 reset status change This bit indicates, when 1, that the port 2 port reset status bit has changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
19	OCIC	Port 2 overcurrent indicator change This bit indicates, when 1, that the port 2 port overcurrent indicator has changed. A write of 1 clears this bit. A write of 0 has no effect. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0
18	PSSC	Port 2 suspend status changed This bit indicates, when 1, that the port 2 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2) (Continued)

Bit	Name	Description	Type	Reset
17	PESC	Port 2 enable status change This bit indicates, when 1, that the port 2 port enable status changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
16	CSC	Port 2 connect status change This bit indicates, when 1, that the port 2 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HCRHDESCRIPTORB.DR[2] bit is set to indicate a nonremovable USB device on port 2, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 2 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 2. A 0 in this bit indicates a full-speed device. This bit is valid only when port 2 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 2 port power status. A write of 0 to this bit has no effect. The OMAP5912 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means.	R/W	0

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2) (Continued)

Bit	Name	Description	Type	Reset
8	PPS/SPP	Port 2 port power status/set port power This bit indicates, when read as 1, that the port 2 power is enabled. When read as 0, port 2 power is not enabled. The OMAP5912 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. A write of 1 to this bit sets the port 2 port power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 2 port reset status/set port reset When read as 1, indicates that port 2 is sending a USB reset. When read as 0, USB reset is not being sent to port 2. A write of 1 to this bit sets the port 2 port reset status bit and causes the USB host controller to begin signaling USB reset to port 2. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 2 port overcurrent indicator/clear suspend status When read as 1, indicates that a port 2 port overcurrent condition has occurred. When 0, no port 2 port overcurrent condition has occurred. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 2 port suspend status is 1 causes resume signaling on port 2. A write of 1 when port 2 port suspend status is 0 has no effect. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 24. HC Port 2 Status and Control Register (HCRHPORTSTATUS2) (Continued)

Bit	Name	Description	Type	Reset
2	PSS/SPS	Port 2 port suspend status/set port suspend When read as 1, indicates that port 2 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 2 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 2 current connect status is 1, a write of 1 to this bit sets the port 2 port suspend status bit and places port 2 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 2 port enable status/set port enable When read as 1, indicates that port 2 is enabled. When read as 0, this bit indicates that port 2 is not enabled. This bit is automatically set at completion of port 2 USB reset if it was not already set before the USB reset completed and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. A write of 1 to this bit when port 2 current connect status is 1 sets the port 2 port enable status bit. A write of 1 when port 2 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 2 current connection status/clear port enable When read as 1, indicates that port 2 currently has a USB device attached. When 0, indicates that no USB device is attached to port 2. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[2] bit is set to indicate a non-removable device on port 2. A write of 1 to this bit clears the port 2 port enable bit. A write of 0 to this bit has no effect.	R/W	0

The HC port 3 status and control register reports and controls the state of USB host port 3. Depending on HMC_MODE (see Table 24) value and top-level pin multiplexing settings, HCRHPORTSTATUS2 can provide status and control for the host controller port that is usually associated with the following OMAP5912 USB pins:

❏ MCSI2.DOUT/USB2.TXEN
❏ UART2.TX/USB2.TXD
❏ UART2.RTS/USB2.SE0
❏ UART2.CTS/USB2.RCV
❏ MCSI2.DIN/USB2.VP
❏ UART2.RX/USB2.VM
❏ MCSI2.SYNC/USB2.SPEED
❏ MCSI2.CLK/USB2.SUSP

The host controller port associated with this register can also be held in a state where it always appears to be disconnected, depending on the HMC_MODE value and top-level pin multiplexing (see Section 4.3, *Pin Multiplexing*).

HC port 3 is not capable of acting as the host portion of an OTG controller. OTG functionality is only available through HC port 1 and HC port 2. See Section 4, *USB OTG Controller*.

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3)

Bit	Name	Description	Type	Reset
31:21	Reserved	Reserved		
20	PRSC	Port 3 reset status change This bit indicates, when 1, that the port 3 port reset status bit has changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
19	OCIC	Port 3 overcurrent indicator change This bit indicates, when 1, that the port 3 port overcurrent indicator has changed. A write of 1 clears this bit. A write of 0 has no effect. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3) (Continued)

Bit	Name	Description	Type	Reset
18	PSSC	Port 3 suspend status change This bit indicates, when 1, that the port 3 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
17	PESC	Port 3 enable status change This bit indicates, when 1, that the port 3 port enable status changed. A write of 1 clears this bit. A write of 0 has no effect.	R/W	0
16	CSC	Port 3 connect status change This bit indicates, when 1, that the port 3 current connect status has changed because of a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set. A write of 1 clears this bit. A write of 0 has no effect. If the HcRhDescriptorB.DR[3] bit is set to indicate a non-removable USB device on Port 3, this bit is set only after a root hub reset to inform the system that the device is attached.	R/W	0
15:10	Reserved	Reserved		
9	LSDA/CPP	Port 3 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 3. A 0 in this bit indicates a full-speed device. This bit is only valid when port 3 current connect status is 1. The host controller driver can write a 1 to this bit to clear the port 3 port power status. A write of 0 to this bit has no effect. OMAP5912 USB host controller does not control external port power using OHCI mechanisms, so, if required, USB host port power must be controlled through other means. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	0

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3) (Continued)

Bit	Name	Description	Type	Reset
8	PPS/SPP	Port 3 power status/set port power This bit indicates, when read as 1, that the port 3 power is enabled. When read as 0, port 3 power is not enabled. The OMAP5912 does not provide signals from the USB host controller to control external port power, so, if required, USB host port power control signals must be controlled through other means. Software can track the current power state using the port power status bit and other power control bits, but those bits have no direct effect on external port power control. A write of 1 to this bit sets the port 3 power status bit. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS. System software can update this register to simplify host controller driver and/or OTG driver coding.	R/W	1
7:5	Reserved	Reserved		
4	PRS/SPR	Port 3 reset status/set port reset When read as 1, indicates that port 3 is sending a USB reset. When read as 0, USB reset is not being sent to port 3. A write of 1 to this bit sets the port 3 reset status bit and causes the USB host controller to begin signaling USB reset to port 3. A write of 0 to this bit has no effect.	R/W	0
3	POCI/CSS	Port 3 overcurrent indicator/clear suspend status When read as 1, indicates that a port 3 overcurrent condition has occurred. When 0, no port 3 overcurrent condition has occurred. The OMAP5912 does not provide inputs for signaling external overcurrent indication to the USB host controller. Overcurrent monitoring, if required, must be handled through some other mechanism. A write of 1 to this bit when port 3 suspend status is 1 causes resume signaling on port 3. A write of 1 when port 3 suspend status is 0 has no effect. A write of 0 has no effect. This bit has no relationship to the OTG controller register bits that relate to VBUS.	R/W	0

Table 25. HC Port 3 Status and Control Register (HCRHPORTSTATUS3) (Continued)

Bit	Name	Description	Type	Reset
2	PSS/SPS	Port 3 port suspend status/set port suspend When read as 1, indicates that port 3 is in the USB suspend state, or is in the resume sequence. When 0, indicates that port 3 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. If port 3 current connect status is 1, a write of 1 to this bit sets the port 3 suspend status bit and places port 3 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device. A write of 0 to this bit has no effect.	R/W	0
1	PES/SPE	Port 3 enable status/set port enable When read as 1, indicates that port 3 is enabled. When read as 0, this bit indicates that port 3 is not enabled. This bit is automatically set at completion of port 3 USB reset if it was not already set before the USB reset completed and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. A write of 1 to this bit when port 3 current connect status is 1 sets the port 3 enable status bit. A write of 1 when port 3 current connect status is 0 has no effect. A write of 0 has no effect.	R/W	0
0	CCS/CPE	Port 3 current connection status/clear port enable When read as 1, indicates that port 3 currently has a USB device attached. When 0, indicates that no USB device is attached to port 3. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[3] bit is set to indicate a non-removable device on port 3. A write of 1 to this bit clears the port 3 enable bit. A write of 0 to this bit has no effect.	R/W	0

The host UE address register reports the physical address of the last OCPI bus access that caused an unrecoverable error (UE). This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 26. Host UE Address Register (HOSTUEADDR)

Bit	Name	Description	Type	Reset
31:0	UE_ADDR	Unrecoverable error address This register captures the physical address of any OCPI bus operation that is started by the USB host controller that encounters an unrecoverable error condition. This information, along with the information in HOSTUESTATUS, can help a developer determine why the USB host issued an OCPI access to a physical address that resulted in an unrecoverable error.	R	0x0000 0000

The host UE status register reports the OCPI bus cycle-type for the last unrecoverable error that occurred. This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Table 27. Host UE Status Register (HOSTUESTATUS)

Bit	Name	Description	Type	Reset
31:1	Reserved	Reserved	R	xxxxxxxx
0	UEAccess	Access type when unrecoverable error occurred When an unrecoverable error occurs because of time-out of a OCPI bus write, this bit is set. When an unrecoverable error occurs because of time-out of a OCPI bus read, this bit is cleared. This bit has no meaning before an unrecoverable error occurs. This information, along with the information in HOSTUEADDR, can help a developer determine why the USB host issued an OCPI bus access that resulted in an unrecoverable error.		0

The host time-out control register controls the USB host controller OCPI bus time-out mechanism. This register is not defined by the OHCI specification.

Table 28. Host Time-out Control Register (HOSTTIMEOUTCTRL)

Bit	Name	Description	Type	Reset
31:1	Reserved	Reserved		
0	TO_DIS	OCPI bus time-out disable When 1, the USB host controller OCPI bus time-out counter is disabled and the host controller waits indefinitely for completion of a USB host controller access to system memory. When 0, the USB host controller waits indefinitely to access system memory. When cleared (the default state), the USB host controller waits no more than 4096 OCPI bus clocks for completion of a OCPI bus access to system memory. If the OCPI bus cycle does not complete in that time, the USB host controller signals an unrecoverable error.	R/W	0

The host revision register returns the revision number for the OMAP5912 USB host controller. This register is not defined by the OHCI specification.

Table 29. Host Revision Register (HOSTREVISION)

Bit	Name	Description	Type	Reset
31:8	Reserved	Reserved	R	xxxxxx
7:4	MAJORREV	Major revision number MAJORREV indicates the major revision number of the USB host controller. The original OMAP5912 USB host controller version implements a major revision number of 0.	R	xx
3:0	MINORREV	Minor revision number MINORREv indicates the minor revision number of the USB host controller. The original OMAP5912 USB host controller version implements a minor revision number of 0.	R	xx

2.5 USB Host Controller Reserved Registers and Reserved Bit Fields

To enhance code reusability with possible future versions of the USB host controller, reads and writes to reserved USB host controller register addresses are to be avoided. Unless otherwise specified, when writing registers that have reserved bits, read-modify-write operations must be used so that the reserved bits are written with their previous values.

2.6 USB Host Controller Registers, USB Reset, and USB Clocking

When the USB host controller clock is disabled, or when the ULPD does not provide 48 MHz to the USB host controller, reads from and writes to the USB host controller registers do not occur correctly. To properly access the USB host controller registers, the USB host controller must be clocked and must be out of reset.

USB host controller clock enable is controlled as described in Table 30**.**

Table 30. USB Host Controller Clock Control

OTG_SYS-CON_ 2 OTG_ PADEN	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_ HHC_UHOST_EN_R	OTG_ SYSCON2 UHOST_EN	USB Host Clock Enabled?	USB Host in Reset?
0	Don’t care	0	No	Yes
0	Don’t care	1	Yes	No
1	0	Dont care	No	Yes
1	1	Dont care	Yes	No

The USB host controller hardware reset is controlled by the ARM_RSTCT2.PER_EN bit and the OTG_SYSCON_1.SOFT_RESET bit. The USB host controller is held in reset whenever ARM_RSTCT2.PER_EN is 0 or OTG_SYSCON_1.SOFT_RESET is 1. The USB host controller can transition out of reset whenever the clock is enabled and ARM_RSTCT2.PER_EN is 1 and OTG_SYSCON_1.SOFT_RESET is 0.

The USB host controller completes its reset within about 72 cycles of the 48-MHz clock after the host controller clock is transitioned from disabled to enabled using the mechanisms shown in Table 30 and the host controller reset is removed. After system software turns on the clock to the USB host controller and removes it from reset, it is necessary to wait until the USB host controller internal reset completes. To ensure that the USB host controller has completely reset, system software must wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

2.7 OHCI Interrupts

The OMAP5912 USB host controller provides an interrupt output to the MPU level 2 interrupt handler on its IRQ_06 interrupt input. This is a level-sensitive interrupt signal, and the MPU level 2 interrupt handler IRQ_06 must be programmed as a level-sensitive input.

2.7.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

2.7.2 OHCI HCDONEHEAD Writeback Interrupt

The OHCI HCDONEHEAD writeback interrupt is supported as described in the *OHCI Specification for USB*.

2.7.3 OHCI Start Of Frame Interrupt

The OHCI start of frame interrupt is supported as described in the *OHCI Specification for USB*.

2.7.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

2.7.5 OHCI Unrecoverable Error Interrupt

The OHCI unrecoverable error interrupt is supported as described in the *OHCI Specification for US*B. This interrupt occurs if the USB host controller is unable to complete an OCPI bus read or OCPI write within 4096 OCPI bus clocks when the USB host OCPI bus time-out feature is enabled [see Table 15-28, *Host Time-out Control Register (HOSTTIMEOUTCTRL*)]. When an OCPI bus time-out causes an unrecoverable error, HOSTUEADDR and HOSTUESTATUS are updated. When an isochronous TD is processed with an OFFSET/PSW field that is not set for *Not Accessed*, an unrecoverable error interrupt is generated, but HOSTUEADDR and HOSTUESTATUS are not updated.

2.7.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

2.7.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*. The OMAP5912 does not provide a connection between the USB host controller and USB port overcurrent detection hardware, so the root hub status change interrupt does not occur because of a port overcurrent event.

2.7.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

2.8 USB Host Controller Access to System Memory

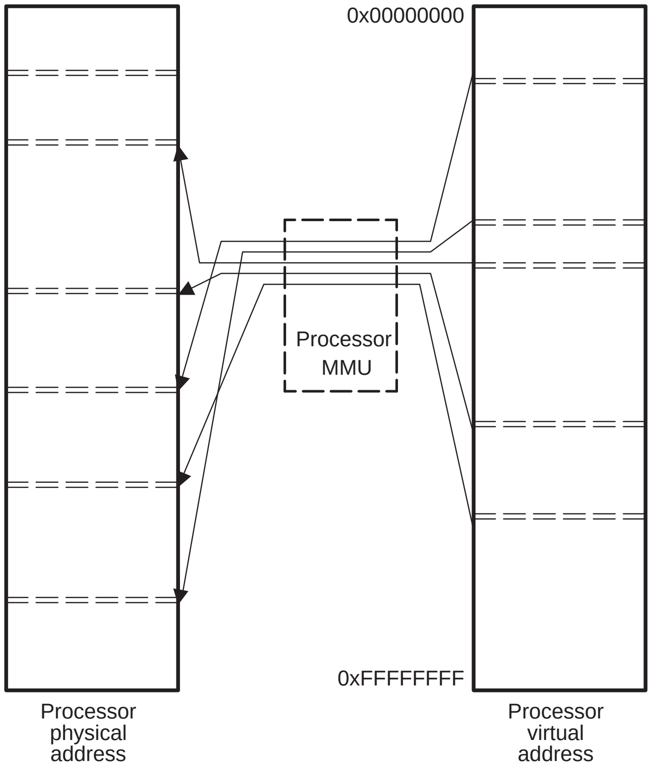
The USB host controller must have access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic. The OMAP5912 OCPI bus allows the USB host controller to access OMAP5912 system memory, as shown in Figure 1.

2.9 Physical Addressing

Transactions on the OMAP5912 OCPI bus use physical addresses, so all system memory accesses initiated by the USB host controller must use physical addresses. The OMAP5912 MPU can be configured to use virtual addressing. In this case, MPU software manipulates virtual addresses that may or may not be identical to physical addresses. When virtual addressing is used, system software must perform the appropriate virtual address to physical address and physical address to virtual address conversions when manipulating the USB host controllers data structures and pointers to those data structures.

Figure 2 shows the MPU virtual address to physical address conversion.

Figure 2. Relationships Between Virtual Address Physical Address



2.10 Cache Coherency in OHCI Data Structures and Data Buffers

The OMAP5912 traffic controller does not provide mechanisms to flush (or writeback) the MPU cache when a DMA controller or OCPI access to system memory occurs. Because there is no forced coherency mechanism, the system implementation must ensure that the OMAP5912 USB host controller can access the correct data from system memory and that the MPU accesses that same data. This requires that any system memory accessed by the USB host controller be allocated in non-cached system memory.

If the OHCI data structures and/or data buffers are allocated in cached portions of system memory, a cache coherency problem can exist because the MPU can read from, and, if in writeback mode, write to the cache; but the USB host controller accesses are always directly to the physical system memory. If the data structures are in a cached portion of system memory and writeback mode is enabled, it is possible that the USB host controller can read stale data that has not been updated by a cache writeback.

Similarly, if the data structure is in memory that is currently in the MPU cache (either writeback or writethrough mode) and the OHCI controller modifies the information in physical memory, the MPU can read stale data from the cache.

Cache coherency problems can be avoided by allocating the OHCI data structures (HCCA, EDs, and TDs) and the USB data buffers in non-cacheable system memory. In this case, every MPU access directly accesses physical memory, so there is no coherency issue. Configuration of cacheable portions of the MPU virtual address space is provided via the MPU memory management unit. See the description of the MPU MMU in the *OMAP3.2 Hardware Engine Reference Guide (SWPU019C)*.

2.11 OCPI Bus Addressing and OHCI Data Structure Pointers

The USB host controller OHCI registers that point to the HCCA and the ED lists must be programmed with values that correspond to the physical addresses of the particular data structure. System software must use physical addresses and not processor addresses when manipulating the OHCI control registers that point to the HCCA, and to the ED and TD lists. System software must also use physical addresses for the ED and TD pointers that are stored within ED and TD entries.

The USB host controller driver software must also be able to examine the list of completed transfer descriptors that the host controller creates as it retires transfer descriptors. This list is pointed to by the HCDONEHEAD register, which contains a physical address that points to the most recent transfer descriptor that has been retired. This requires that the host controller driver software must be able to convert from the physical address back to an MPU virtual address.

Several address conversion functions are helpful to enable proper addressing by the MPU software and the USB host controller. The functionality required for proper address conversion depends on the settings used in the MPU MMU, so it is system-dependent. The conversion functions are described in general terms in the following subsections.

2.11.1 MPUVAtoPA()—MPU Virtual Address to Physical Address Conversion Function

This function converts an MPU virtual address to the equivalent system physical address. This function must understand the way that the system software has configured the MPU MMU. This function must provide a conversion that has a result that is identical to the conversion done in hardware by the MPU MMU.

2.11.2 PAtoMPUVA()—Physical Address to MPU Virtual Address Conversion Function

This function is a reverse version of the MPU virtual address to physical address conversion. It accepts a physical address as an argument and returns the equivalent MPU virtual address.

It is possible to program the MPU MMU to allow two (or more) different MPU virtual addresses to map to the same physical address. This is especially common in systems that use linear addressing within a task. System software implementers must be careful to avoid that situation or to perform the conversion in a way that understands the task-specific conversion requirements.

2.12 NULL Pointers

The *OHCI Specification for USB* uses NULL pointers to indicate the end of a list. The OMAP USB host controller compares the ED and TD pointers against the value 0x00000000 to determine if the pointer is a null pointer. Address conversion routines must understand this usage.

2.13 OMAP5912 OCPI Bus and the USB Host Controller

The OMAP5912 OCPI bus provides a mechanism whereby OMAP5912 peripheral modules can access system memory. Unlike peripherals that use the DMA controller, peripherals connected to the OCPI bus can generate the address, byte enables, and read/write indication for each OCPI access. This functionality allows the USB host controller to implement a scatter-gather engine to access the OHCI data structures from system memory in an efficient manner.

2.14 OCPI Registers

The USB host controller connects to an OCPI arbiter, which then connects to the transfer controller OCP-I port. There are no OCPI arbiter register settings that affect USB host controller access to system memory, but the transfer controller OCP-I port must be properly configured to allow access to system memory.

Because the USB host must be able to access system memory, the OMAP5912 security features must be properly configured to allow USB host controller access to system memory.

2.15 USB Host Controller Clock Control

The OMAP5912 clock generation and system reset management module (ULPD) provides a 48-MHz clock to the USB OTG controller, USB device controller, and USB host controller. This clock can be stopped by software to reduce USB OTG controller, USB device controller, and USB host controller power consumption when USB functionality is not needed.

The ULPD controls the 48-MHz clock to the USB host controller via:

❑ CLOCK_CTRL_REG.USB_MCLK_EN

❑ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ

❑ SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

When these ULPD register bits are properly configured, the USB host controller receives a clock when the UHOST_EN signal is active. See discussions of OT_SYSCON_2.OTG_PADEN and OTG_SYSCON_2.HOST_EN in Section 4.

The USB host controller is held in reset and receives no clock at any time that UHOST_EN is inactive.

2.16 USB Host Controller Hardware Reset

The ULPD module provides reset of the USB host controller. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5912 peripherals, including the USB host controller. When held in reset, the USB host controller does not generate any USB activity on its USB ports. The USB host controller requires that its 48-MHz clock input (from the OMAP5912 ULPD module) be active and that UHOST_EN be set (see Table 64, *OTG System Configuration Register 2 (OTG_SYSCON_2)*) in order to complete its reset sequence.

There is a delay of approximately 72 cycles of the ULPD USB host controller 48-MHz clock before the USB host controller is successfully reset. This delay starts at the latest of the PER_EN bit set, UHOST_EN set, or 48-MHz clock start. When the USB host controller is in hardware reset, read or write accesses to its registers have no effect. It is recommended that USB host controller software read the HCREVISION and HCHCCA registers after deasserting reset to verify the proper reset values. If the read values for both HCREVISION and HCHCCA are not correct, reset the values and continue reading until the proper reset values are seen.

The UHOST_EN bit, when cleared, also holds the USB host controller in a hardware reset. While the USB host controller is in reset, reads from the USB host controller registers do not return valid data, and writes to the USB host controller registers have no effect. When UHOST_EN is cleared, all USB host controller internal state information is lost.

Software that initializes the USB host controller must ensure that the reset is turned off, that the ULPD 48-MHz clock for the USB host controller is enabled, and that UHOST_EN is set. It must then wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

2.17 USB Host Controller OHCI Reset

The *OHCI Specification for USB* provides the HCR bit in the HCCOMMANDSTATUS register, which resets the OHCI controller. This reset can be used to reset the OHCI functionality and has no effect on the USB host controller OCPI and the MPU public peripheral bus interfaces. The OHCI reset does not affect the USB host controller clock.

2.18 USB Host Controller Power Management

Power management of the OMAP5912 USB host controller is limited to disabling the clock to the USB host by clearing UHOST_EN (see (see Table 64, *OTG System Configuration Register 2 (OTG_SYSCON_2)*). When UHOST_EN is 0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by

HMC_MODE is not affected, so a HMC_MODE setting that multiplexes USB function controller and/or UART1 signals to OMAP5912 top-level multiplexing can still make use of the USB function controller and/or UART1.

When the OMAP5912 host controller 48-MHz clock is disabled or UHOST_EN is 0, all USB host controller OHCI registers and the HOSTUEADDR, HOSTUESTATUS, HOSTTIMEOUTCTRL, and HOSTREVISION registers are inaccessible.

2.19 OCPI Clocking

The OCPI clock must be active to allow USB host controller access to system memory. The OCPI clock is controlled by the ARM_IDLECT3.EN_OCPI_CK and IDLOCPI_ARM bits.

3 USB Device Controller

The USB device controller supports the implementation of a full-speed device fully compliant with the USB 1.1 standard.

It provides an interface between the MPU and the USB wire and handles USB transactions with minimal MPU software intervention.

The USB device controller module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific items of a configuration are for each endpoint, the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or ISO), and the associated number.

The USB device controller module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions.

3.1 USB Device Controller Registers

Table 31 lists the USB device controller registers. Table 32 through Table 54 describe the register bits.

Table 31. USB Device Controller Registers

Register	Description	R/W	Address
REV	Revision		0xFFFB4000
	Endpoint		
EP_NUM	Endpoint selection		0xFFFB4004

Table 31. USB Device Controller Registers(Continued)

Register	Description	R/W	Address
DATA	Data		0xFFFB4008
CTRL	Control		0xFFFB400C
STAT_FLG	Status		0xFFFB4010
RXFSTAT	Receive FIFO status		0xFFFB4014
SYSCON1	System configuration 1		0xFFFB4018
SYSCON2	System configuration 2		0xFFFB401C
DEVSTAT	Device status		0xFFFB4020
SOF	Start of frame		0xFFFB4024
IRQ_EN	Interrupt enable		0xFFFB4028
DMA_IRQ_EN	DMA interrupt enable		0xFFFB402C
IRQ_SRC	Interrupt source		0xFFFB4030
EPN_STAT	Non-ISO endpoint interrupt enable		0xFFFB4034
DMAN_STAT	Non-ISO DMA interrupt enable		0xFFFB4038
Reserved			0xFFFB403C
DMA Configuration			
RXDMA_CFG	DMA receive channels configuration		0xFFFB4040
TXDMA_CFG	DMA transmit channels configuration		0xFFFB4044
DATA_DMA	DMA FIFO data		0xFFFB4048
Reserved			0xFFFB404C
TXDMA0	Transmit DMA control 0		0xFFFB4050
TXDMA1	Transmit DMA control 1		0xFFFB4054
TXDMA2	Transmit DMA control 2		0xFFFB4058
Reserved			0xFFFB405C
RXDMA0	Receive DMA control 0		0xFFFB4060
RXDMA1	Receive DMA control 1		0xFFFB4064
RXDMA2	Receive DMA control 2		0xFFFB4068

Table 31. USB Device Controller Registers(Continued)

Register	Description	R/W	Address
Reserved			0xFFFB406C...407F
	Endpoint Configuration		
EP0	Endpoint configuration 0		0xFFFB4080
EP1_RX...EP15RX	Receive endpoint configuration 1...15		0xFFFB4084...40BC
Reserved			0xFFFB40C0
EP1_TX...EP15TX	Transmit endpoint configuration 1...15		0xFFFB40C4...40FC

- ❑ 16-bit access: all bits of the register are accessed.
- ❑ 8-LSB bit access: the 8 least significant bits are accessed.
- ❑ 8-MSB bit access: the 8 most significant bits are accessed.

Table 32. Revision Register (REV)

Bit	Name	Description
15:8	-	Reserved
7:0	REV_NB	This 8-bit field indicates revision number of current USB device controller module—value fixed by hardware: 0x01: Revision 0.1 0x02: Revision 0.2 0x21: Revision 2.1

This read-only register contains the revision number of the module. A write to this register is denied.

MPU and USB reset have no effect on this register.

Table 33. Endpoint Selection Register (EP_NUM)

Bit	Name	Description
15:7	−	Reserved
6	SETUP_SEL	The setup FIFO select bit is set by the USB device controller to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If EP_NUM.EP_DIR bit is set to 0, the MPU can read data from endpoint RX FIFO by reading DATA register. If EP_NUM.EP_DIR bit is set to 1, the MPU can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the USB device controller must absolutely clear this bit: 0: No access 1: Access permitted Note: When the USB device controller sets this bit, it must set SETUP_SEL bit to 0. After having accessed the endpoint FIFO either for read or for write access, the USB device controller must clear this bit by writing a 0 to it. The value after MPU or USB reset is low.
5	EP_SEL	The TX/RX FIFO select bit is set by the USB device controller in response to a set-up general USB interrupt, to access the EP0 read-only setup FIFO when reading DATA register. Setting this bit clears the IRQ_SRC.SETUP interrupt bit. When this bit is set, the value of other EP_NUM register bits must be 0. 0: No access 1: Access permitted Note: After having read the setup FIFO, the USB device controller must clear this bit by writing a 0 to it. Value after MPU or USB reset is low.

Table 33. Endpoint Selection Register (EP_NUM) (Continued)

Bit	Name	Description
4	EP_DIR	The endpoint direction bit gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM: 0: OUT endpoint 1: IN endpoint Value after MPU or USB reset is low.
3:0	EP_NUM	The endpoint number binary encoded in these four bits, associated with the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status and the control and data locations are for this endpoint. 0000: EP0 0001: EP1 … 1111: EP15 Value after MPU or USB reset is low.

This read/write register selects and enables the endpoint that can be accessed by the USB device controller.

Table 34. Data Register (DATA)

Bit	Name	Description
15:0	DATA[†]	Transmit/receive FIFO data: EP_NUM.EP_DIR = 1: This register contains the data written by the USB device controller to be sent to the USB host during the next IN transaction. Data can be written successfully only if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: This register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data).

[†] A write into the data register (DATA) when EP_NUM.EP_DIR = 0 and a read from the register when EP_NUM.EP_DIR = 1 are denied.

This register is the entry point to write into a selected TX endpoint or to read data from a selected RX endpoint, or to read data from setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is denied. If selected endpoint direction is IN, this register is write-only and a read to this register is denied.

Table 35. Control Register (CTRL)

Bit	Name	Description
15:8	−	Reserved
7	CLR_HALT[†]	The clear halt endpoint (non-ISO) bit only concerns non-ISO endpoints—used by the USB device controller to clear an endpoint halt condition: 0: No action 1: Clear halt condition Always read 0
6	SET_HALT[†]	The set halt endpoint (non-ISO) only concerns non-ISO endpoints—used by the USB device controller to halt the selected endpoint. The halted endpoint returns STALL handshakes to the USB host. The USB device controller can disable the endpoint interrupt if it does not want to be informed of STALL handshakes. Note: If the endpoint to halt is used by a DMA channel, the USB device controller must disable the DMA channel before setting halt condition for this endpoint. 0: No action 1: Halt endpoint Always read 0
5:3	−	Reserved

[†] It is not required to set EP_NUM.EP_SEL bit before setting this bit. If this bit is set during the handling of an interrupt to the endpoint, however, the USB device controller must not set EP_NUM.EP_SEL bit before setting CTRL.CLR_HALT bit, in order to avoid a possible effect on interrupts . The USB device controller must check that FIFO is empty before setting the halt feature for the endpoint. A STALL transaction clears the FIFO.

Table 35. Control Register (CTRL) (Continued)

Bit	Name	Description
2	SET_FIFO_EN[†]	The set FIFO enable (non-ISO) bit only concerns non-ISO endpoints. If the selected endpoint direction is IN, the USB device controller uses this bit to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, the USB device controller uses this bit to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not, setting the device returns a NAK handshake. ISO endpoints FIFO are always enabled. Note: The USB device controller must never enable endpoint 0 FIFO out of control transfers. For bulk and interrupt endpoints, FIFO must never be enabled when the halt feature is set or when RX FIFO is not empty. Furthermore, during EP interrupts handling, the USB device controller must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts). 0: No action 1: FIFO enabled Always read 0
1	CLR_EP	Clear endpoint: the USB device controller sets this bit to clear the selected endpoint FIFO pointers and flags. This bit resets the FIFO pointers, the FIFO empty status bit is set, and the FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. It also clears the previous transaction handshake status. For ISO endpoints or non-ISO double-buffered endpoints, both foreground and background FIFO are cleared. 0: No action 1: Clear endpoint Always read 0
0	RESET_EP	The endpoint reset (non-control) bit only concerns non-control endpoints.The USB device controller sets this bit to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears the halt condition, and clears the FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an ISO endpoint, it only clears the FIFO (both foreground and background). 0: No action 1: Reset endpoint Always read 0

[†] It is not required to set EP_NUM.EP_SEL bit before setting this bit. If this bit is set during the handling of an interrupt to the endpoint, however, the USB device controller must not set EP_NUM.EP_SEL bit before setting CTRL.CLR_HALT bit, in order to avoid a possible effect on interrupts . The USB device controller must check that FIFO is empty before setting the halt feature for the endpoint. A STALL transaction clears the FIFO.

This set-only register controls the FIFO and status of the selected endpoint. A read access to this register always returns 0.

Endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register CTRL is implemented for this FIFO because the USB device controller cannot control it.

Table 36. Status Register (STAT_FLG)

Bit	Name	Description
15	NO_RXPACKET	The isochronous no packet received (ISO OUT) bit only concerns ISO OUT endpoints.This bit notifies the USB device controller that the core has not received any isochronous packet on the endpoint. This bit is updated on a SOF (start of frame). 0: The endpoint received a packet on the previous frame. 1: The endpoint did not receive a packet on the previous frame. Value after MPU or USB reset is high.
14	MISS_IN	The isochronous missed IN token for the previous frame (ISO IN) bit only concerns ISO IN endpoints.This bit notifies the USB device controller that the core missed a valid ISO IN token during the previous frame and that TX data were flushed from the FIFO instead of being transmitted to the USB host. This bit is updated on a SOF (start of frame). 0: The endpoint received an IN token the previous frame. 1: The endpoint did not receive an IN token the previous frame and TX data were flushed. Value after MPU or USB reset is low.
13	DATA_FLUSH	The isochronous receive data flush (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that data were flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the USB device controller does not read all of the data from the foreground FIFO in a frame. This bit is updated in every frame. 0: Not significant 1: Data was flushed. Value after MPU or USB reset is low.

Table 36. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
12	ISO_ERR	The isochronous receive data error (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that the ISO data packet was received incorrectly. This happens when the core detects an error in the data packet (CRC, bit stuffing, PID check) or when there is an overrun condition in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty. This bit is updated in every frame. 0: Not significant 1: ISO packet received with errors Value after MPU or USB reset is low.
11:10	−	Reserved
9	ISO_FIFO_EMPTY	The ISO FIFO empty bit only concerns ISO endpoints.This bit is set when the FIFO for the selected ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. 0: ISO FIFO is not empty. 1: ISO FIFO is empty. Value after MPU or USB reset is high (FIFO empty).
8	ISO_FIFO_FULL	The ISO FIFO full bit only concerns ISO endpoints.This bit is set when the FIFO for the selected ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host). 0: ISO FIFO is not full. 1: ISO FIFO is full. Value after MPU or USB reset is low (FIFO empty).
7	-	Reserved
6	EP_HALTED	The endpoint halted flag (non-ISO) bit only concerns non-ISO endpoints.This bit when asserted 1 indicates that the selected endpoint is halted. The endpoint can be put into the halt state only by the USB device controller writing the endpoint halt control bit (in response to a SET_FEATURE request, for instance). 0: The selected endpoint is not halted. 1: The selected endpoint is halted. Value after MPU or USB reset is low.

Table 36. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
5	STALL	The transaction stall (non-ISO) bit only concerns non-ISO endpoints.This status bit is set at the end of a transaction if a STALL handshake packet was returned to the USB host, and if no non-handled interrupt is pending on the current buffer (see Section 3.11, *Important Note on USB Device Interrupts*). The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by a halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No STALL handshake was returned. 1: A STALL handshake packet was returned. Value after MPU or USB reset is low.
4	NAK	The transaction non-acknowledge (non-ISO) bit only concerns non-ISO endpoints with SYSCON1.NAK_EN bit asserted.This status bit is set at the end of a transaction if a NAK handshake is returned to the USB host, and if no non-handled interrupt is pending on the current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint, and the STAT_FLG.FIFO_EN bit is not set for the endpoint. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No NAK handshake was returned (SYSCON1.NAK_EN bit is set). 1: A NAK handshake packet was returned and SYSCON1.NAK_EN bit is set. Value after MPU or USB reset is low.
3	ACK	The transaction acknowledge (non-ISO) bit only concerns non-ISO endpoints.This bit is set at the end of a non-transparent valid IN transaction if the data packet was sent successfully to the USB host and the ACK handshake was received, or at the end of a non-transparent valid OUT transaction if the data packet was received successfully by the USB device and the ACK handshake was returned. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection). 0: No ACK handshake packet was returned. 1: An ACK handshake packet was returned. Value after MPU or USB reset is low.

Table 36. Status Register (STAT_FLG) (Continued)

Bit	Name	Description
2	FIFO_EN	The FIFO enable status (non-ISO) bit only concerns non-ISO endpoints.This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and is cleared automatically after a transaction completes with an ACK, STALL. 0: The non-ISO endpoint FIFO is disabled. 1: The non-ISO endpoint FIFO is enabled. Value after MPU or USB reset is low.
1	NON_ISO_FIFO_EMPTY	The non-ISO FIFO empty bit only concerns non-ISO endpoints.This bit is set when the FIFO for the selected non-ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO. 0: Non-ISO FIFO is not empty. 1: Non-ISO FIFO isempty. Value after MPU or USB reset is high (FIFO empty).
0	NON_ISO_FIFO_FULL	The non-ISO FIFO full bit only concerns non-ISO endpoints.This bit is set when the FIFO for the selected non-ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host). 0: Non-ISO FIFO is not full. 1: Non-ISO FIFO is full. Value after MPU or USB reset is low (FIFO empty).

This read-only register provides a status of the FIFO and the results of the transactions handshakes for the selected endpoint. The 8 MSB are reserved for ISO endpoints, whereas the 8 LSB are reserved for non-ISO endpoints. This register cannot be read if EP_NUM.EP_SEL bit is not asserted for the endpoint. No status flag exists for the read-only set-up FIFO, which is always enabled.

Note:

The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if SYSCON1.NAK_EN is asserted to 1. An ERR handshake or a NAK handshake when SYSCON1.NAK_EN is 0 is considered transparent.

A write to this register has no effect.

Table 37. Receive FIFO Status Register (RXFSTAT)

Bit	Name	Description
15:10	−	Reserved
9:0	RXF_COUNT	The receive FIFO byte count 10-bit field indicates the number of bytes currently in the receive FIFO. Values after MPU or USB reset is low (all 10 bits).

This read-only register indicates how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The USB device controller cannot read this register if the EP_NUM.EP_SEL bit is not set for the endpoint. No receive FIFO status exists for the setup FIFO, because 8 bytes are always expected.

Table 38. System Configuration Register 1 (SYSCON1)

Bit	Name	Description
15:9	-	Reserved
8	CFG_LOCK	Device configuration locked bit: after the USB device controller has entered the device configuration (registers 0x20 to 0x3F), it must set the CFG_LOCK bit so that the device can be used. When the device configuration is not locked, the device is not ready to be used: 0: Device configuration is not locked. Device is not ready. 1: Device configuration is locked. The value after the USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).

Table 38. System Configuration Register 1 (SYSCON1) (Continued)

Bit	Name	Description
7	DATA_ENDIAN	The data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write): 0: Little endian 1: Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
6	DMA_ENDIAN	The DMA data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write): 0: Little endian 1: Big endian The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
5	-	Reserved
4	NAK_EN	The NAK enable bit can be set by the USB device controller to be signaled for NAK transaction handshake response. When this bit is set, STAT_FLG.NAK is set on a NAK handshake if no non-handled interrupt is pending on the endpoint, and the endpoint interrupt is asserted. In the normal mode, when cleared, NAK handshake response to the USB host is made transparent to the USB device controller and no interrupt is asserted: 0: NAK is disabled. 1: NAK is enabled. The value after MPU or USB reset is low. Note: If the USB device controller sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data.
3	AUTODEC_DIS	The autodecode process disabled can be set to force software to handle all EP0 transactions (except the SET_ADDRESS transaction): 0: Autodecode process is activated (see Table 55, *Autodecoded Versus Non-Autodecoded Control Requests*). 1: Autodecode process is deactivated. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.

Table 38. System Configuration Register 1 (SYSCON1) (Continued)

Bit	Name	Description
2	SELF_PWR	The self-powered bit indicates to the USB host whether the device is bus-powered or self-powered. This information is needed for an autodecoded request. The USB device controller must update this bit after a SET_CONFIGURATION according to the self-powered bit D6 given in the configuration descriptor (see *USB 1.1 Specification*). 0: Bus powered 1: Self-powered The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
1	SOFF_DIS	When the shutoff disable bit is set, it disables the power shutoff circuitry: 0: Power shutoff circuitry is enabled. 1: Power shutoff circuitry is disabled. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
0	PULLUP_EN	The external pullup enable bit allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent USB traffic when the device is not ready: 0: Pull-up is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off. 1: Pull-up is enabled. The value after USB device controller hardware reset is low; after USB reset, it is high; and after detach, it is unchanged.

This read/write register provides control functions for power management and miscellaneous control for the core.

Values depend on whether the reset action comes from the USB device controller (MPU) or the USB host.

Figure 3. Little Endian and Big Endian Formats

15 8	7 0
Byte 1	Byte 0

Little Endian Format (16 bits)

15 8	7 0
Byte 0	Byte 1

Big Endian Format (16 bits)

Table 39. System Configuration Register 2 (SYSCON2)

Bit	Name	Description
15:7	−	Reserved
6	RMT_WKP	The set-only remote wake-up bit, when written with a 1, initiates the remote wake-up sequence even if DEVSTAT.R_WK_OK bit was not previously set to 1 by the USB host. Reading this bit always returns 0. Writing 0 into this bit has no effect. To generate a resume, the software must check the remote wake-up enable value before initiating any wake-up sequence: 0: No action 1: Initiates the remote wake-up sequence Always read 0
5	STALL_CMD	The set-only stall command bit only concerns non-autodecoded requests on control endpoint (EP0).This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests. 0: No action 1: Stall current USB command Always read 0
4	Reserved	Reserved
3	DEV_CFG	If the USB device controller receives a SET_CONFIGURATION with a valid configuration value, and the device is in addressed state, it must write a 1 to the device configured (DEV_CFG) bit to inform the command decode that the device moves to configured state. The core sets the DEVSTAT.CFG bit to 1. If the device is already configured when the SET_CONFIGURATION request is received, the USB device controller does not have to set this bit. If the new configuration value is 0, USB device controller must set the SYSCON2.CLR_CFG bit to move to addressed state. Reading this bit always returns 0. Writing 0 into this bit has no effect. 0: No action 1: Allows DEVSTAT.CFG to be set Always read 0

Table 39. System Configuration Register 2 (SYSCON2) (Continued)

Bit	Name	Description
2	CLR_CFG	If the USB device controller receives a SET_CONFIGURATION with a configuration value of 0, and if the device is configured, it must write a 1 to the clear configured (CLR_CFG) bit to inform the command decode that the device becomes deconfigured (moves to addressed state). The core clears the DEVSTAT.CFG bit. Reading this bit always returns 0. Writing 0 into this bit has no effect. 0: No action 1: Allows DEVSTAT.CFG to be cleared Always read 0
1:0	Reserved	Reserved

This set-only register provides miscellaneous controls for the function. A read to this register always returns 0.

Table 40. Device Status Register (DEVSTAT)

Bit	Name	Description
15:10	Reserved	Reserved
9	B_HNP_ENABLE	The HNP enable for B-device bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: HNP enable for B-device Value after MPU or USB reset is low.
8	A_HNP_SUPPORT	The B-device is directly connected to an A-device port that supports HNP. This A_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: B-device connection to HNP capable A-device Value after MPU or USB reset is low.

Table 40. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
7	A_ALT_HNP_SUPPORT	The B-device is connected to an A-device port that is not capable of HNP, but the A-device has an alternate port that is capable of HNP. This A_ALT_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*. 0: Not significant 1: B-device connection to not HNP capable A-device Value after MPU or USB reset is low.
6	R_WK_OK	The remote wake-up granted bit is automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host grants the function the ability to assert remote wake-up. 0: Not significant 1: Remote wake-up granted Value after MPU or USB reset is low.
5	USB_RESET	USB reset signaling is active. The USB_RESET bit returns 1 when the USB host resets the USB bus. A valid USB reset resets the endpoint FIFOS, the other control register bits, except for SYSCON1.CFG_LOCK and the associated configuration registers (0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_CHG, and forces the device to the default state for DEVSTAT (this register). This bit is cleared at the end of reset. This bit, as well as other DEVSTAT bits, is double-buffered. If a pending interrupt has not been handled when a USB reset occurs and is handled only when USB reset is finished, the USB device controller does not see the USB_RESET bit going high and then low. 0: Device is not being reset by USB host. 1: Device is being reset by USB host. The value after the USB device controller hardware reset is low, and during USB reset is high (low after USB reset).

Table 40. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
4	SUS	Suspended state bit: the device in suspended state is, at a minimum, attached to the USB, powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It can also have a unique address and be configured for use. Because the device is suspended, however, the host cannot use the device function. This bit returns 1 when the USB device is in a suspended state. 0: Not suspended 1: Suspended Value after MPU or USB reset is low.
3	CFG	Configured state bit: the device is attached to the USB, powered, has been reset, has a unique address, and is configured. The host can use the function provided by the device. This bit returns 1 when the USB device has been configured after a set SYSCON2.DEV_CFG = 1. This bit remains set to 1 until the device becomes deconfigured. This bit is cleared when the core receives a valid SET_CONFIGURATION request and the USB device controller sets the SYSCON2.CLR_CFG bit. During the time this bit is not set to 1, transactions that are not for control EP0 are ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled. 0: Not configured 1: Configured Value after MPU or USB reset is low.
2	ADD	Addressed state bit: the device is attached to the USB, powered, has been reset, and a unique device address has been assigned. This bit (ADD) returns 1 after an ET_ADDRESS standard request. This bit remains set to 1 until the device becomes de-addressed. 0: Not addressed 1: Addressed Value after MPU or USB reset is low.

Table 40. Device Status Register (DEVSTAT) (Continued)

Bit	Name	Description
1	DEF	The default state bit returns 1 when the USB device is attached to the USB and powered, and has been reset. This bit remains set to 1 until the device becomes de-powered. The device moves into default state as soon as the USB reset is effective. 0: Not in default 1: Default The value after MPU is low, and after USB reset is high.
0	ATT	The attached state bit returns 1 when the device is attached to the USB and is powered. This bit remains set to 1 until the device powers down. 0: Not attached 1: Attached The value after USB device controller hardware reset is low (unattached) or high (attached), and after USB reset is high.

This read-only register provides a status reflecting the visible device states as defined in *USB1.1 Specification*. A write to this register has no effect.

Note:

This register is double buffered. If IRQ_EN.DS_CHG_IE is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS_CHG interrupt. Therefore, if there is a state change and a pending DS_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

The values depend on whether the reset action comes from the USB device controller (MPU) or USB host. They also depend on whether the device is attached or not when the USB device controller hardware reset event occurs.

Table 41. Start of Frame Register (SOF)

Bit	Name	Description
15:13	Reserved	Reserved
12	FT_LOCK	Frame timer locked bit: the USB host sends out a start of frame (SOF) packet every millisecond. When the device does not receive a SOF packet because of a bus error, a local start of-frame that is used for ISO FIFO switch is generated. Once the core receives two valid SOF, separated by TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in frame interval IF allowed by the USB 1.1 specification (IF = [11964:12036] USB bit time). If TF is out of this interval, SOF.FT_LOCK value remains 0, and a local SOF is generated by the core. When SOF.FT_LOCK is set and the frame timer is locked to the timing TF, a local SOF is generated, if no valid SOF has been received in an interval of TF since the last valid SOF. The SOF.FT_LOCK bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame time. If the core does not receive a valid SOF, the frame timer remains locked to TF. When SOF.FT_LOCK is cleared, a local SOF is generated after the 12036 USB bit time, if no valid SOF has been received, and SOF.FT_LOCK remains 0. 0: Frame timer is not locked. 1: Frame timer is locked. The value after MPU or USB reset is low. Note: Each time the core receives a valid SOF out of the allowed interval IF, a local SOF is generated and the ISO FIFO switches.
11	TS_OK	The timestamp OK bit indicates that the timestamp in SOF.TS is valid for the current frame. It returns a 1 if a valid SOF packet was received from the USB host and a 0 otherwise. 0: Timestamp is invalid. 1: Timestamp is valid. Value after MPU or USB reset is low.
10:0	TS	The timestamp number field returns the timestamp from the last USB host-valid SOF packet. The frame number is valid if SOF.TS_OK is 1. In case of an SOF miss, this value is not updated and TS_OK is cleared. The value after MPU or USB reset is low (all 11 bits).

This read-only register provides a frame timer status for use in ISO communications. A write to this register is denied.

Table 42. Interrupt Enable Register (IRQ_EN)

Bit	Name	Description
15:8	Reserved	Reserved
7	SOF_IE	Start of frame interrupt enable
6	Reserved	Reserved
5	EPn_RX_IE	Receive endpoint n interrupt enable (non-ISO)
4	EPn_TX_IE	Transmit endpoint n interrupt enable (non-ISO)
3	DS_CHG_IE	Device state changed interrupt enable
2:1	Reserved	Reserved
0	EP0_IE	EP0 transactions interrupt enable

This read/write register enables all non-DMA interrupts (control, state changed, ISO, and non-ISO).

The values depend on whether the reset action comes from the USB device controller (MPU) or the USB host.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding IRQ_SRC bit is asserted to 1 by the core for any IRQ_SRC bit controlled by this bit. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after MPU or USB reset is low for all bits except IRQ_EN.DS_CHG_IE bit, which remains unchanged after a reset from the USB host.

Table 43. DMA Interrupt Enable Register (DMA_IRQ_EN)

Bit	Name	Description
15:11	Reserved	Reserved
10	TX2_DONE_IE	Transmit DMA channel 2 done interrupt enable
9	RX2_CNTt_IE	Receive DMA channel 2 transactions count interrupt enable
8	RX2_EOT_IE	Receive DMA channel 2 end of transfer interrupt enable
7	Reserved	Reserved

Table 43. DMA Interrupt Enable Register (DMA_IRQ_EN) (Continued)

Bit	Name	Description
6	TX1_DONE_IE	Transmit DMA channel 1 done interrupt enable
5	RX1_CNT_IE	Receive DMA channel 1 transactions count interrupt enable
4	RX1_EOT_IE	Receive DMA channel 1 end of transfer interrupt enable
3	Reserved	Reserved
2	TX0_DONE_IE	Transmit DMA channel 0 done interrupt enable
1	RX0_CNT_IE	Receive DMA channel 0 transactions count interrupt enable
0	RX0_EOT_IE	Receive DMA channel 0 end of transfer interrupt enable

This read/write register enables all DMA interrupts.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding bit in the IRQ_SRC register is asserted to 1 by the core. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after MPU or USB reset is low for all bits.

Table 44. Interrupt Source Register (IRQ_SRC)

Bit	Name	Description
15:11	Reserved	Reserved
10	TXn_DONE	The transmit DMA channel n done interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer. The core automatically sets this bit when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXDMAn.TXn_TSC (transfer size counter) equals 0, and the last IN transaction completes with an ACK. When this bit is asserted, the USB device controller must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. The endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA channel n. 0: No action 1: Non-ISO transmit DMA transfer for a channel has ended. The value after MPU or USB reset is low.

Table 44. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
9	RXn_CNT	The receive DMA channel n transactions count interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit during an active Receive DMA transfer each time RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after the RX DMA data have been read (end of DMA request). When this bit is asserted, the USB device controller must read DMAN_STAT register to identify the endpoint number for which the transfer completed. An RXn_CNT IT is asserted also if RXDMAn.RXn_STOP is set; in this case, both IRQ_SRC.RXn_EOT and IRQ_SRC.RXn_CNT are asserted. 0: No action 1: Non-ISO receive DMA transfer for a channel has reached transactions count level. The value after MPU or USB reset is low.
8	RXn_EOT	The receive DMA channel n end of transfer interrupt flag (non-ISO) bit is for non-ISO DMA transfer only. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit when a receive DMA channel detects an end-of-transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after the RX DMA data have been read (end of DMA request). When this happens, the DMA assigned endpoint FIFO is kept disabled (STAT_FLG.FIFO_EN = 0) to avoid receiving a new packet data from the USB host. The USB device controller can grant DMA transfer again to the same endpoint by enabling the FIFO again (STAT_FLG.FIFO_EN = 1). An end of transfer is detected when the core receives a data packet that is less than the configured endpoint FIFO size (or is empty), or when RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status and the RXDMAn.RXn _STOP bit is set. When this bit is asserted, the USB device controller must read DMAN_STAT.DMAn_RX_IT_SRC to identify the endpoint number for which the transfer completed, and DMAN_STAT.DMAn_RX_SB must be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from the DATA_DMA register). The endpoint interrupt IRQ_SRC.EPn_RX is never set to RX DMA channel for the assigned endpoint. 0: No action 1: Non-ISO receive DMA transfer for a channel has ended. The value after MPU or USB reset is low.

Table 44. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
7	SOF	Start of frame interrupt flag bit: every millisecond, the USB host outputs a start-of-frame packet to the functions. The SOF bit reflects when a new SOF is received. Writing a 1 in the SOF bit location clears the flag. Writing a 0 has no effect. In accordance with the USB1.1 specification, if SOF is received corrupted or is not received, the core still sets this flag at the same rate (if bit SOF.FT_LOCK = 1) or after 12043 USB bit time (if bit SOF.FT_LOCK = 0). 0: No action 1: Start-of-frame packet received (or internal SOF) The value after MPU or USB reset is low.
6	Reserved	Reserved
5	EPn_RX	The EPn OUT transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets the EPn_RX bit when a handshake sequence occurs for an OUT transaction to an interrupt of the bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt. 0: No action 1: OUT transaction detected on an endpoint The value after MPU or USB reset is low.
4	EPn_TX	The EPn IN transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets this bit when a handshake sequence occurs for an IN transaction to an interrupt of bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt. 0: No action 1: IN transaction detected on an endpoint value after MPU or USB reset is low.
3	DS_CHG	Device state changed interrupt flag bit: the core automatically resets the DS_CHG bit when the state of the device changes. This is when the core modifies any of the bits present in the DEVSTAT register. When this bit is cleared, the background DEVSTAT register moves into foreground position. 0: No action 1: Device state change detected Value after USB device controller hardware reset is low and after USB reset is high.

Table 44. Interrupt Source Register (IRQ_SRC) (Continued)

Bit	Name	Description
2	SETUP	Setup transaction interrupt flag bit: the core automatically sets the SETUP bit when a valid setup transaction completes on control endpoint for a non-autodecoded control request and automatically clears it when the USB device controller sets EP_NUM.SETUP_SEL bit when reading setup data. A write 1 to it has no effect. 0: No action 1: Valid setup transaction occurred on endpoint 0. Value after MPU or USB reset is low.
1	EP0_RX	EP0 OUT transactions interrupt flag bit: the core automatically sets the EP0_RX bit when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0: No action 1: OUT transaction on EP0 Value after MPU or USB reset is low.
0	EP0_TX	EP0 IN transactions interrupt flag bit: the core automatically sets this bit when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). 0: No action 1: IN transaction on EP0 Value after MPU or USB reset is low.

This read/clear only register identifies and clears the source of the interrupt signaled by a set flag.

The value depends on whether the reset action comes from the USB device controller (MPU) or the USB host.

The USB device controller can clear a set bit location only by writing a 1 into the bit location (except for the setup bit, which is automatically cleared by the core). A write 0 has no effect.

When the core sets a bit location to 1, an interrupt is signaled to the USB device controller if the interrupt was enabled.

- 0: No interrupt
- 1: Interrupt signaled

The value after the MPU or USB reset is low except for the IRQ_SRC.DS_CHG bit, which is high after a USB reset.

Table 45. Non-ISO Endpoint Interrupt Status Register (EPN_STAT)

Bit	Name	Description
15:12	Reserved	Reserved
11:8	EPn_RX_IT_SRC	The receive endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0000: No receive endpoint interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).
7:4	Reserved	Reserved
3:0	EPn_TX_IT_SRC	Transmit endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints.When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. 0000: No transmit endpoint interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

This read-only register identifies the non-ISO endpoint causing an EPn interrupt. A write into it is denied.

Note:

If a nontransparent transaction occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one and EPN_STAT is updated with the corresponding interrupt assertion.

Table 46. Non-ISO DMA Interrupt Status Register (DMAN_STAT)

Bit	Name	Description
15:11	Reserved	Reserved
12	DMAn_RX_SB	The DMA receive single byte (non-ISO) bit only concerns non-ISO endpoints only (ISO endpoints receive a constant number of bytes). This bit is set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core received an odd number of bytes during the last transaction. It is used to know the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the IRQ_SRC.RXn_EOT flag is cleared, this bit reads as 0. 0: No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction. 1: An EOT DMA interrupt is pending and an odd number of bytes was received during the last transaction. Value after MPU or USB reset is low.
11:8	DMAn_RX_IT_SRC	The DMA receive interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0. 0000: No receive DMA interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
7:4	Reserved	Reserved
3:0	DMAn_TX_IT_SRC	The DMA transmit interrupt source (non-ISO) bit only concerns non-ISO endpoints only. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0. 0000: No transmit DMA interrupt is pending. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

This read-only register identifies the endpoint that causes a DMAn interrupt. A write into it is denied.

Note:

If a DMA interrupt occurs before a previous one on another endpoint in the same direction, the second interrupt is asserted only after the USB device controller clears the first one. DMAn_STAT is updated when the corresponding interrupt is asserted.

Table 47. DMA Receive Channels Configuration Register (RXDMA_CFG)

Bit	Name	Description
15:13	Reserved	Reserved
12	RX_REQ	The RX DMA request active level or pulse bit allows the RXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active during 2 cycles. 0: RX DMA request active level 1: RX DMA request active pulse Value after MPU or USB reset is low.
11:8	RXDMA2_EP	Receive endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 2 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

Table 47. DMA Receive Channels Configuration Register (RXDMA_CFG) (Continued)

Bit	Name	Description
7:4	RXDMA1_EP	Receive endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 1 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).
3:0	RXDMA0_EP	Receive endpoint number for DMA channel 0. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint. 0000: Receive DMA channel 0 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all 4 bits).

This read/write register enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by an RX DMA channel must have been configured (through register EPn_RX). The RXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (RXDMA0_EP). In this case, the ISO endpoint is configured on channel 2,and it can never be serviced with low software and a fast USB host.

The USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with OMASP5912 controller inputs.

CAUTION

System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Table 48. DMA Transmit Channels Configuration Register (TXDMA_CFG)

Bit	Name	Description
15:13	Reserved	Reserved
12	TX_REQ	The TX DMA request active level or pulse bit allows the TXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active for two cycles. 0: TX DMA request active level 1: TX DMA request active pulse Value after MPU or USB reset is low.
11:8	TXDMA2_EP	Transmit endpoint number for DMA channel 2: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 2 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).

Table 48. DMA Transmit Channels Configuration Register (TXDMA_CFG) (Continued)

Bit	Name	Description
7:4	TXDMA1_EP	Transmit endpoint number for DMA channel 1: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 1 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).
3:0	TXDMA0_EP	Transmit endpoint number for DMA channel 0: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint. 0000: Transmit DMA channel 0 is deactivated. 0001: EP1 … 1111: EP15 Values after MPU or USB reset are low (all four bits).

This read/write register enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each. An endpoint used by a TX DMA channel must have been configured (through register EPn_TX). TXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (TXDMA0_EP). In this case, your ISO endpoint is configured on the channel 2 and it can never be serviced with low software and a fast USB host.

USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with OMASP5912 controller inputs.

Table 49. DMA FIFO Data Register (DATA_DMA)

Bit	Name	Description
15:0	DATA_DMA	DMA FIFO data. When an RX DMA request is active for a channel (only one active at a given time), this register contains the data that the core received from the USB host OUT transaction using this channel. Data can be accessed by the main DMA controller engine (read access) in response to the DMA request for the channel. When a TX DMA request is active for a channel (only one active at a given time), this register contains the data written by the main DMA controller engine (write access) in response to a DMA request for the transmit channel to be sent to the USB host during the next IN transaction. Warning: It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that caused the RX DMA request to be active, and a write access affects the endpoint that caused the TX DMA request to be active. Caution: The USB device controller must not access this register directly; however, there is no hardware mechanism to prevent such access. No access into this register must be made out of DMA request handling.

This register is the entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1 or 2.

Table 50. Transmit DMA Control Register n (TXDMAn)

Bit	Name	Description
15	TXn_EOT	Transmit DMA channel n end of transfer: the TXn_EOT bit can be either 0 or 1 for BULK DMA transfer. When the USB device controller sets it to 1, this bit signals the core that the transfer size set in TXDMAn.TXn_TSC is in bytes. A TX one interrupt (IRQ_SRC.TXn_DONE) is asserted with the last IN transaction. If the number of bytes set in TXDMAn.TXn_TSC is a multiple of the endpoint buffer size, the TX done interrupt is asserted only after an IN transaction with an empty data packet. When cleared, the transfer size set in TXn_TSC is in full buffer size for the endpoint selected (BULK only). A TX done interrupt is asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes. 0: DMA transfer size is in buffers. 1: DMA transfer size is in bytes. Value after MPU or USB reset is low.
14	TXn_START	Transmit DMA channel n start. The USB device controller sets this bit to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, unless the USB device controller clears the endpoint in the TXDMA_CFG register. A write 0 into this bit has no effect and a read to this bit always returns 0. The IRQ_SRC.TXn_DONE interrupt bit is asserted when the DMA transfer ends. 0: No action 1: DMA transfer start Always reads 0

Table 50. Transmit DMA Control Register n (TXDMAn) (Continued)

Bit	Name	Description
13:10	Reserved	Reserved
9:0	TXn_TSC	Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, which the USB device controller writes into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMAn.TXn_EOT) to be transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has still to transmit. Read mode is only provided for software debug purposes. Caution: For ISO transfer, the user must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to prevent this situation. If this situation occurs, results are unpredictable. Caution: For bulk transfer, when TXDMAn.TXn_EOT = 0, a set value of TXDMAn.TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, 0001, 000, stop. When TXDMAn.TXn_EOT = 1, a set value of TXDMAn.TXn_TSC = 0 a NULL packet is sent in response to the next IN token. Values after MPU or USB reset are low (all 10 bits).

This read/write register controls the operation of the transmit DMA channel n (n = 0, 1, 2).

Table 51. Receive DMA Control Register n (RXDMAn)

Bit	Name	Description
15	RXn_STOP	Receive DMA channel n transfer stop. When this bit is set, an RXn_EOT interrupt is asserted to the USB device controller after n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT), and the USB device controller expects a given amount of data for the transfer. Caution. At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The USB device controller must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel. Values after MPU or USB reset are low.
14:8	Reserved	Reserved
7:0	RXn_TC	Receive DMA channel n transactions count. The USB device controller can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register must be programmed to the desired transaction watermark limit prior to enabling the DMA transfer for the receive DMA channel n. Caution: A reached watermark does not disable an active DMA transfer if RXDMAn.RXn_STOP was not set. If RXDMAn.RXn_STOP was set for the transfer both RXn_CNT and RXn_EOT interrupts are asserted. A read to that register returns the number of transactions remaining before the IRQ_SRC.RXn_CNT interrupt flag is asserted. This read mode is only provided for software debug purposes. Values after MPU or USB reset are low (all 8 bits).

This read/write register monitors incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

Table 52. Endpoint 0 Configuration Register (EP0)

Bit	Name	Description
15:14	Reserved	Reserved
13:12	EP0_SIZE	Endpoint 0 FIFO size. This field contains the endpoint 0 FIFO size value and must match the value sent by the USB device controller to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY and STAT_FLG.NON_ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0. The USB device controller must fill this field before setting SYSCON1.CFG_LOCK. 00: 8 bytes 01: 16 bytes 10: 32 bytes 11: 64 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that value is unknown until first write access).
11	Reserved	Reserved
10:0	EP0_PTR	Endpoint 0 pointer. This field contains the address of the endpoint 0 pointer. Value 0x000 is not permitted (reserved for setup FIFO). 0x000: address = BASE (not permitted) 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Note: The pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

This read/write register gives the device configuration for control endpoint 0.

Values depend on whether the reset action comes from USB device controller (MPU) or the USB host.

Table 53. Receive Endpoint n Configuration Register (EPn_RX)

Bit	Name	Description
15	EPn_RX_VALID	The receive endpoint n valid bit must be set by the USB device controller to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0: Receive endpoint n does not exist for this configuration. 1: Receive endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_RX_SIZE/DB	Receive non-ISO (or ISO) endpoint n double-buffer (DB).This bit is only for non-ISO endpoints. For ISO endpoints, which are always double-buffered, this bit is endpoint size MSB. This bit must be set by the USB device controller to allow double buffering for receive non-ISO endpoint n. This is used to reduce number of transactions resulting in NAK handshake. 0: No double buffer for non-ISO receive endpoint n. 1: Double buffer used for non-ISO receive endpoint n. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

Table 53. Receive Endpoint n Configuration Register (EPn_RX) (Continued)

Bit	Name	Description
13:12	EPn_RX_SIZE	The receive endpoint n size field contains the endpoint n FIFO size value. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL, STAT_FLG. ISO_FIFO_EMPTY, and STAT_FLG.ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n. Non-ISO: 00: 8 bytes ISO: 000: 8 bytes [13:12] 01: 16 bytes [14:12] 001: 16 bytes 10: 32 bytes 010: 32 bytes 11: 64 bytes 011: 64 bytes 100: 128 bytes 101: 256 bytes 110: 512 bytes 111: 1023 bytes Warning: For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents it from being programmed with a 2K-byte USB device controller. Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

Table 53. Receive Endpoint n Configuration Register (EPn_RX) (Continued)

Bit	Name	Description
11	EPn_RX_ISO	The receive ISO endpoint n field must be set if the receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0: Receive endpoint n type is isochronous. 1: Receive endpoint n type is bulk or interrupt. Values after USB device controller hardware reset or USB reset are unchanged.
10:0	EPn_RX_PTR	The receive endpoint n pointer field contains the address of the receive endpoint n pointer. Value 0x000 is not permitted (reserved for setup FIFO). Caution: For ISO endpoints or for non-ISO endpoints that allow double-buffering, 2*RX buffer size must be reserved for ping-pong. 0x000: address = BASE (not permitted) 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Note: Pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

This read/write register gives the device configuration for non-control receive endpoint n (n: 115). The endpoints size fields must match values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

Values depend on whether the reset action comes from USB device controller (MPU) or USB host.

Table 54. Transmit Endpoint n Configuration Register (EPn_TX)

Bit	Name	Description
15	EPn_TX_VALID	The transmit endpoint n valid bit must be set by the USB device controller to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core. 0: Transmit endpoint n does not exist for this configuration. 1: Transmit endpoint n is part of the device configuration. Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_TX_SIZE/Db	Transmit non-ISO (or ISO) endpoint n double buffer. This bit is only for non-ISO endpoints used in DMA mode. For ISO endpoints, which are always double buffered, this bit is endpoint size MSB. For non-ISO endpoints that are not used in a DMA transfer, double buffering is not provided.This bit must be set by the USB device controller to allow double buffering for transmit non-ISO endpoint n, when used in a DMA transfer. This is used to reduce the number of transactions resulting in NAK handshake. 0: No double buffer for non-ISO transmit endpoint n. 1: Double buffer used for non-ISO transmit endpoint n. Values after MPU or USB reset is unchanged. Or transmit ISO endpoint n size[2]
13:12	EPn_TX_SIZE	Transmit endpoint n size

Table 54. Transmit Endpoint n Configuration Register (EPn_TX) (Continued)

Bit	Name	Description
11	EPn_TX_ISO	The transmit ISO endpoint n field must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt). 0: Transmit endpoint n type is isochronous. 1: Transmit endpoint n type is bulk or interrupt. Values after MPU or USB reset is unchanged.
10:0	EPn_TX_PTR	The transmit endpoint n pointer field contains the address of the transmit endpoint n pointer. Caution: For ISO endpoints or for non-ISO endpoints that allow double-buffering, (2*TX buffer size) must be reserved for ping-pong. 0x000: address = BASE 0x001: address = BASE + 8 bytes 0x002: address = BASE + 16 bytes 0x003: address = BASE + 24 bytes ... 0x0FF: address = BASE + 2040 bytes Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access). Pointer value must be set to a value < 0xFF, because the memory size is 2K bytes and a pointer coded value = 0xFF corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

This read/write register gives the device configuration for non-control transmit endpoint n (n: 115). The endpoints size fields must match the values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

Note:

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

3.1.1 EPn_TX[14:12].EPn_TX_SIZE: Transmit Endpoint n Size

EPn_TX.[14] bit description only applies for ISO endpoints.

This field contains the endpoint n FIFO size value. Status flags (FIFO_EMPTY, FIFO_FULL) and underrun condition are based on this value for all IN transactions to endpoint n.

Non-ISO:	00:	8 bytes	ISO:	000:	8 bytes
[13:12]	01:	16 bytes	[14:12]	001:	16 bytes
	10:	32 bytes		010:	32 bytes
	11:	64 bytes		011:	64 bytes
				100:	128 bytes
				101:	256 bytes
				110:	512 bytes
				111:	1023 bytes

WARNING

ISO Endpoints

For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents programming with a 2K-byte USB device controller.

Values after USB device controller hardware reset or USB reset are unchanged (value is unknown until first write access).

3.2 USB Device Transactions

There is an interrupt to the MPU at the end of an USB transaction if the MPU has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at start of frame interrupts. The MPU ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following subsections describe in detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one has been handled by the MPU are not taken into account in this part. The information is organized so that each section deals with one type and direction of transaction, such as:

- ❏ Non-isochronous, non-setup OUT transactions
- ❏ Non-isochronous IN transactions
- ❏ Isochronous OUT transactions
- ❏ Isochronous IN transactions

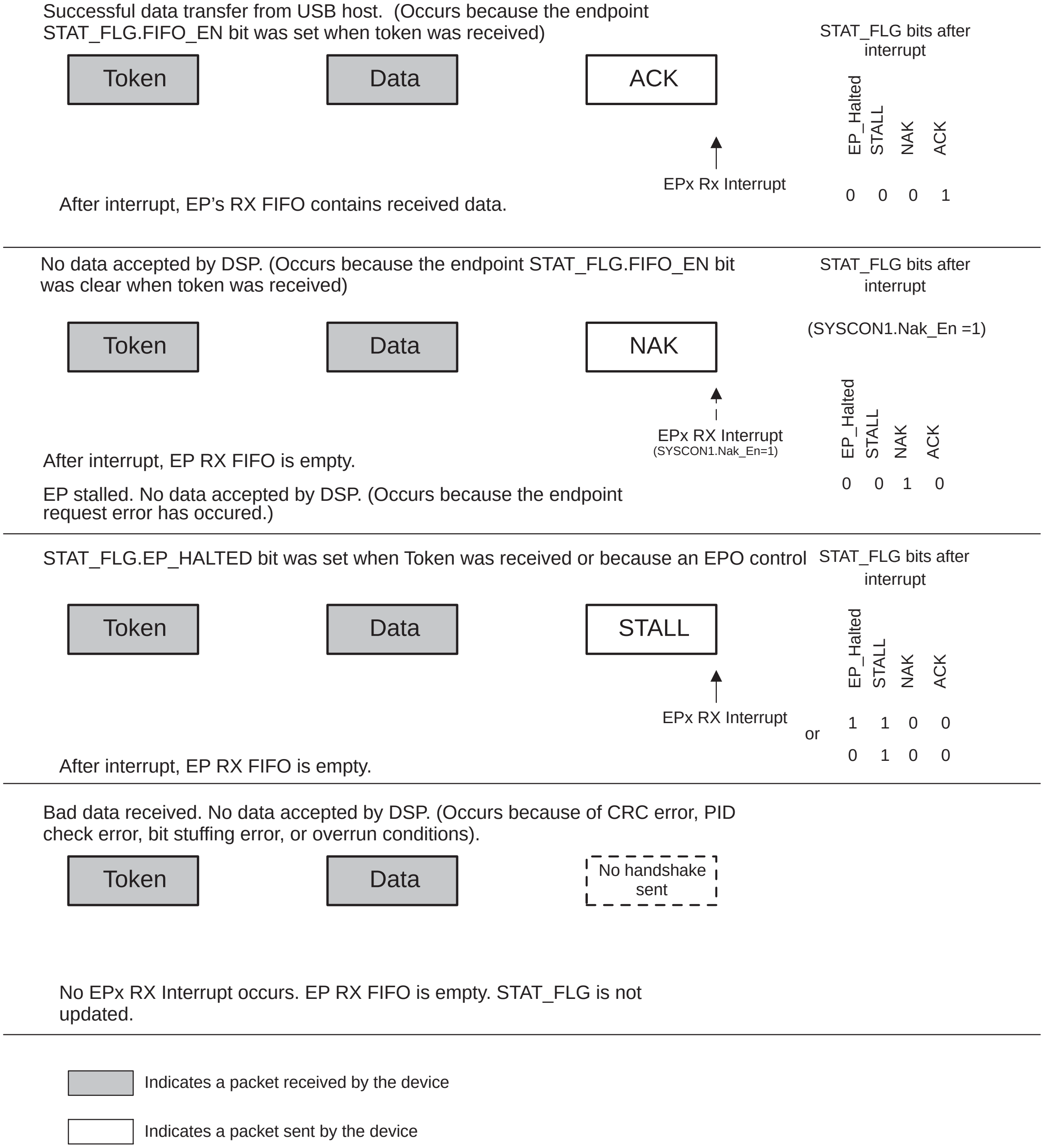
This allows each section to focus only on a specific style of transaction without adding in the confusion of special cases related to other styles.

3.3 Non-Isochronous, Non-Setup OUT (USB HOST -> MPU) Transactions

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the MPU, the USB handshaking protocols are in effect, and data transmission is ensured. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to non-setup transactions on control endpoints.

Figure 4 shows the various USB protocol conditions that can occur during non-isochronous, non-setup OUT transactions. The diagram shows the three phases that can occur in an OUT transaction, the direction of information flow for each phase, when endpoint interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking modes: acknowledge (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case is an abnormal instance where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first (acknowledge) case.

Figure 4. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts



Successful data transfer from USB host. (Occurs because the endpoint STAT_FLG.FIFO_EN bit was set when token was received)

EP_Halted	STALL	NAK	ACK
0	0	0	1

After interrupt, EP's RX FIFO contains received data.

No data accepted by DSP. (Occurs because the endpoint STAT_FLG.FIFO_EN bit was clear when token was received)

EP_Halted	STALL	NAK	ACK
0	0	1	0

After interrupt, EP RX FIFO is empty.

EP stalled. No data accepted by DSP. (Occurs because the endpoint request error has occured.)

STAT_FLG.EP_HALTED bit was set when Token was received or because an EPO control

EP_Halted	STALL	NAK	ACK
1	1	0	0
or			
0	1	0	0

After interrupt, EP RX FIFO is empty.

Bad data received. No data accepted by DSP. (Occurs because of CRC error, PID check error, bit stuffing error, or overrun conditions).

No EPx RX Interrupt occurs. EP RX FIFO is empty. STAT_FLG is not updated.

Indicates a packet received by the device
Indicates a packet sent by the device

3.3.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

An endpoint CTRL.SET_FIFO_EN bit provides the main control for the endpoint ability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint, the endpoint STAT_FLG.FIFO_EN bit is 1, the USB module is allowed to accept the OUT transaction data to the RX FIFO, and, when the transaction completes, the USB module can return ACK to the PC to indicate that the data was received correctly (this is the top case shown in Figure 4). If, however, the endpoint STAT_FLG.FIFO_EN bit was 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (the second case shown in Figure 4).

The USB host need not send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The MPU code must be careful not to read too much data. MPU code must read RXFSTAT.RXF_COUNT value before reading data from the RX FIFO.

At the end of an USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint STAT_FLG.FIFO_EN bit. Once the MPU software has dealt with the OUT transaction data in the endpoint RX FIFO, it must re-enable the endpoint OUT transaction reception by setting the endpoint CTRL.SET_FIFO_EN bit. MPU software can use the endpoint CTRL.SET_FIFO_EN bit as a receive flow control mechanism.

Acknowledged Transactions (ACK)

At completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the MPU and STAT_FLG is updated. In response to the endpoint interrupt, the MPU must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If STAT_FLG.FIFO_EMPTY is cleared, the transaction sent 1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO), and the data is in the endpoint RX FIFO. The MPU knows the number of bytes to read from RX FIFO by reading RXFSTAT.RXF_COUNT value. The MPU can then read RX data from DATA register. Once the MPU has read the data from the FIFO, it sets the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clears the EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

The device can be configured via the SYSCON1.NAK_EN either to inform the MPU of a NAKed transaction or not. If SYSCON1.NAK_EN is cleared, no interrupt is asserted to the MPU if an OUT transaction completes with a NAK handshake and STAT_FLG.NAK bit is not set. If SYSCON1.NAK_EN is set, the USB module issues an endpoint-specific interrupt to the MPU at completion of an OUT transaction to an endpoint, and STAT_FLG.NAK bit is set. In response to the endpoint interrupt, the MPU must read EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The MPU must set the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

3.3.2 Non-Isochronous, Non-Control OUT Transaction Error Conditions

STALLed Transactions

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint either if the endpoint STAT_FLG.EP_HALTED bit is set or if a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STAT_FLG.STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. STAT_FLG.FIFO_EN is of lower priority than STAT_FLG.EP_HALTED; when the STAT_FLG.EP_HALTED bit is set and transactions to the RX endpoint are stalled, regardless of the STAT_FLG.FIFO_EN value. If STAT_FLG.FIFO_EN is set, the STAT_FLG.FIFO_EN bit is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If STAT_FLG.EP_HALTED has been set by the MPU and can be removed, the MPU must set CTRL.CLR_HALT to clear the condition and set CTRL.SET_FIFO_EN to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO. If STAT_FLG.EP_HALTED has been set in response to a SET_FEATURE request sent by the USB host, or if the bit is cleared (control transaction only), the MPU has no action to perform and must clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Packet Errors

In case of a receive data error during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU (the fourth case shown in Figure 4). Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared. If the MPU clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

Sequence Bit Errors

If the core does not receive expected DATA PID during an OUT transaction, the module automatically returns ACK handshake to the USB host, regardless of the STAT_FLG.FIFO_EN bit (per the USB specification). Data is ignored, and no interrupt is asserted to the MPU.

This error occurs if ACK handshake from previous OUT transaction is received corrupted by the USB host.

3.3.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions

If the USB host attempts to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the MPU. Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared.

The MPU must not read more data from RX FIFO than the value indicated by RXFSTAT.RXF_COUNT.

3.4 Non-Isochronous IN (MPU->USB HOST) Transactions

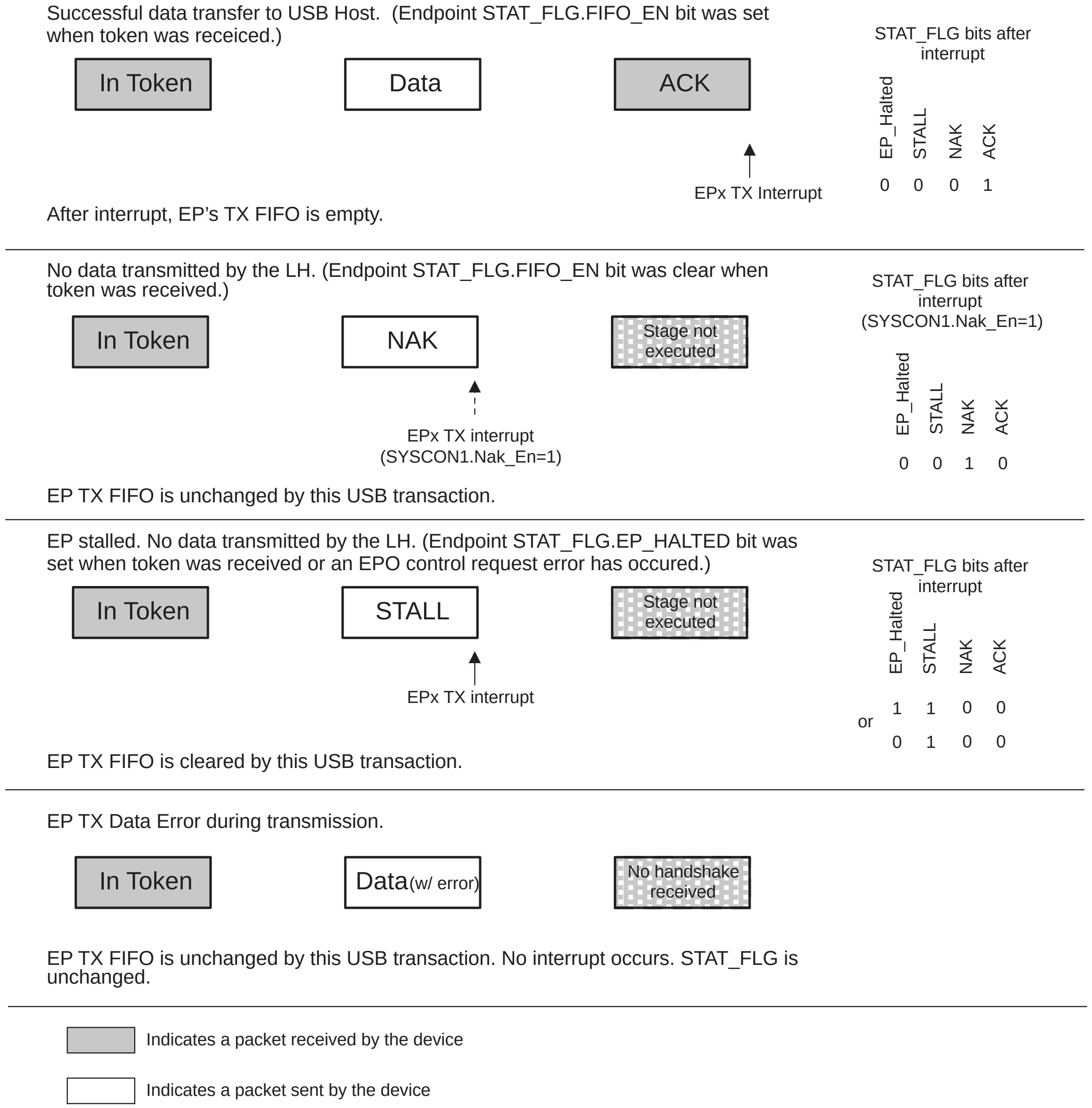
Non-isochronous IN transactions refer to USB transactions in which data is moved from the MPU to the USB host, where the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not ensure USB bandwidth.

To provide data for an endpoint IN transaction, the MPU code writes the transmit data into the endpoint transmit FIFO. MPU code must first wait until the USB is done with any previous TX data for the endpoint (if data had previously been written to the TX FIFO). This must be done by proper response to endpoint-specific transmit interrupts. When an IN transaction to the endpoint occurs, if the endpoint STAT_FLG.FIFO_EN bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the TX FIFO is empty and STAT_FLG.FIFO_EN is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint previous transmit activity is taken care of, the MPU code gains access to endpoint FIFO and status by setting the EP_NUM.EP_SEL bit. Then the MPU can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO). Once all of the transmit data has been written to the endpoint FIFO, MPU code sets the CTRL.SET_FIFO_EN bit to allow the USB to use the endpoint TX FIFO, and then clears the EP_NUM.EP_SEL bit. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

Figure 5 shows the various USB protocol conditions that can occur during non-isochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific nterrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: acknowledge (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case in which there is an error either in the token packet received by the core or in the data packet received by the USB host.

Figure 5. Non-Isochronous IN Transaction Phases and Interrupts


Successful data transfer to USB Host. (Endpoint STAT_FLG.FIFO_EN bit was set when token was receiced.)
In Token
Data
ACK
EPx TX Interrupt
STAT_FLG bits after interrupt
EP_Halted
STALL
NAK
ACK
0 0 0 1
After interrupt, EP's TX FIFO is empty.
No data transmitted by the LH. (Endpoint STAT_FLG.FIFO_EN bit was clear when token was received.)
In Token
NAK
Stage not executed
EPx TX interrupt (SYSCON1.Nak_En=1)
STAT_FLG bits after interrupt (SYSCON1.Nak_En=1)
EP_Halted
STALL
NAK
ACK
0 0 1 0
EP TX FIFO is unchanged by this USB transaction.
EP stalled. No data transmitted by the LH. (Endpoint STAT_FLG.EP_HALTED bit was set when token was received or an EPO control request error has occured.)
In Token
STALL
Stage not executed
EPx TX interrupt
STAT_FLG bits after interrupt
EP_Halted
STALL
NAK
ACK
1 1 0 0
or
0 1 0 0
EP TX FIFO is cleared by this USB transaction.
EP TX Data Error during transmission.
In Token
Data(w/ error)
No handshake received
EP TX FIFO is unchanged by this USB transaction. No interrupt occurs. STAT_FLG is unchanged.
Indicates a packet received by the device
Indicates a packet sent by the device

3.4.1 Non-Isochronous IN Endpoint Handshaking

Per USB specifications for IN transactions, the USB host can provide only one of two handshakes to the USB function during the handshake phase: ACK or no handshake at all. The first indicates successful transfer (first case shown in Figure 5), and the second indicates that the host received a garbled data packet (last case shown in Figure 5).

Acknowledged Transactions (ACK)

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the MPU (see first case in Figure 5). In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received an ACK handshake from the USB host, and the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the MPU has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Non-Acknowledged Transactions (NAK)

For the case in which the MPU is not ready to provide transmit data for transactions to an IN endpoint, the core provides a NAK handshake to the host for any USB IN transaction to that endpoint . Readiness to transmit data is signaled via the endpoint STAT_FLG.FIFO_EN bit; when 1 it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint STAT_FLG.FIFO_EN bit is 0 and an IN transaction to the endpoint occurs, NAK handshake is sent, indicating that the MPU is not ready to handle the request.

If SYSCON1.NAK_EN bit is cleared, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT_FLG is not updated and no endpoint-specific interrupt to the MPU is generated. If the SYSCON1.NAK_EN bit is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the STAT_FLG.NAK bit is set and an endpoint-specific interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint sent a NAK handshake to the USB host. If the MPU has data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. The MPU must then clear the

EP_NUM.EP_SEL bit. That clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register. Signaling NAK does not cause the endpoint TX FIFO to be cleared (because the MPU still retains control of the FIFO).

Signaling NAK handshake for several endpoint transactions in a row can cause the PC host to discard the transaction, so NAK is not necessarily a good mechanism in cases where the MPU is not able to service a request for long periods of time.

3.4.2 Non-Isochronous IN Transaction Error Conditions

STALLed Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint either if the endpoint STAT_FLG.EP_HALTED flag is set, or if a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it is not able to transfer data and instructs the USB host not to retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, usually a control transfer via endpoint 0. The MPU can set the endpoint EP_HALTED bit by selecting the endpoint by writing the appropriate value in EP_NUM register, and then setting the endpoint CTRL.Set_HALT bit, and clear it by selecting the endpoint, and then setting the endpoint CTRL.Clr_HALT bit. When the endpoint EP_HALTED bit is set, the endpoint signals STALL for its IN transactions until the HALT condition is cleared. When the STALL handshake is sent in response to a transaction to the endpoint, the STAT_FLG.STALL bit is set, and an endpoint-specific
interrupt to the MPU is generated.

In response to the endpoint interrupt, the MPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The MPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint sent a STALL handshake to the USB host. The MPU must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; so for a given endpoint number, the TX can be halted when the RX is not.

Packet Errors

If an error (CRC, bit stuffing, or PID check) occurs during the token packet of a USB IN transaction to a non-isochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the MPU occurs for transactions with corrupted packets. If the MPU clears the TX FIFO during the data packet of an IN transaction, a bit stuffing error is forced.

If the USB host returns no handshake after an IN transaction (in case of an error during transmission), the USB device controller module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO to be re-sent during next IN transaction, STAT_FLG.FIFO_EN is not cleared, and no interrupt is asserted to the MPU.

3.4.3 Non-Isochronous IN Endpoint FIFO Error Conditions

The MPU cannot write more data into the TX FIFO than the configured FIFO size.

3.5 Isochronous OUT (USB HOST-> MPU) Transactions

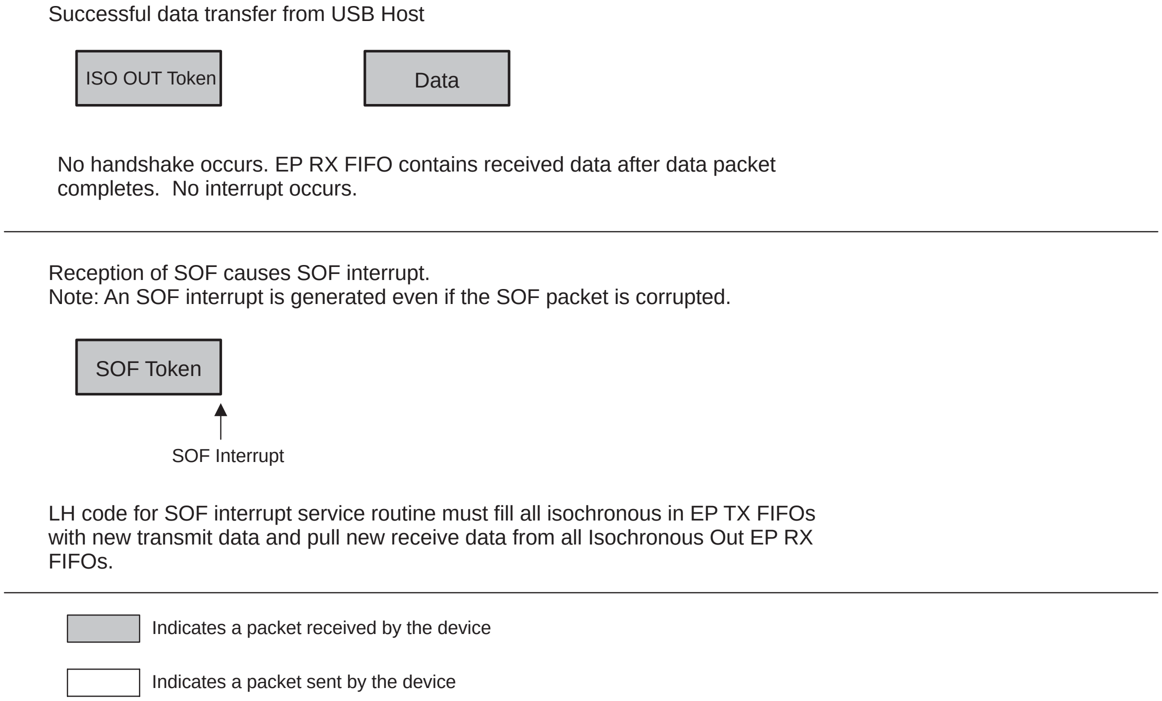
Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the MPU is generated at completion of an isochronous OUT transaction. The MPU is responsible for handling isochronous OUT data at each start of frame (SOF) interrupt.

At every SOF interrupt, for each isochronous OUT endpoint, MPU code must select the endpoint by writing the appropriate value in the EP_NUM register and check STAT_FLG.ISO_FIFO_EMPTY. If the RX FIFO contains data, code must read the RXFSTAT.RXF_COUNT value (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the data register, and then clear the EP_NUM.EP_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to write to the background RX FIFO, and the MPU is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each start of frame (SOF). Isochronous endpoint FIFOs in the background are always enabled to the USB, whereas the foreground FIFOs are enabled to the MPU.

Figure 6 shows the two phases (ISO OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the MPU is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the MPU at each start of frame (SOF) interrupt, which is shown as the second case in Figure 6.

Figure 6. Isochronous OUT Transaction Phases and Interrupts


Successful data transfer from USB Host
ISO OUT Token
Data
SOF Token
SOF Interrupt
Indicates a packet received by the device
Indicates a packet sent by the device

No handshake occurs. EP RX FIFO contains received data after data packet completes. No interrupt occurs.

Reception of SOF causes SOF interrupt.
Note: An SOF interrupt is generated even if the SOF packet is corrupted.

LH code for SOF interrupt service routine must fill all isochronous in EP TX FIFOs with new transmit data and pull new receive data from all Isochronous Out EP RX FIFOs.

3.5.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

3.5.2 Isochronous OUT Transaction Error Conditions

If the MPU fails to read all of the data in the ISO OUT endpoint foreground FIFO by the time the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed, and the STAT_FLG.DATA_FLUSH bit is asserted for the duration of the next frame.

There is no special indication for the case in which the USB host does not provide a transaction to an ISO OUT endpoint during a frame, but once the

FIFO that was background in that frame is foreground, the FIFO is empty (a 0-length data ISO OUT transaction also results in an empty FIFO and cannot be distinguished from a missed ISO OUT transaction).

If an ISO OUT transaction occurs with data error (CRC, PID check, or bit stuffing), the RX FIFO is empty at the next SOF interrupt, and the STAT_FLG.ISO_ERR bit is asserted for the duration of the next frame.

3.5.3 Isochronous OUT Endpoint FIFO Error Conditions

The MPU must never read more data than the value given by RXFSTAT.RXF_COUNT.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the STAT_FLG.ISO_ERR is set at the next SOF interrupt. A properly configured USB system does not do this.

Note:

Both foreground and background isochronous FIFOs are cleared when the CTRL.CLR_EP bit is set.

3.6 Isochronous IN (MPU->USB HOST) Transactions

Isochronous IN transactions are USB transactions in which a given amount of data is transferred from the USB device controller module device to the USB host every 1-ms USB frame. No handshaking is provided.

The USB module provides double-buffering of data for ISO IN endpoints; the background FIFO is used as the source of data for IN transactions to the ISO endpoint, and the foreground FIFO can be written to by the MPU. When an IN transaction to an ISO endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The MPU is responsible for providing new data to the isochronous IN endpoint foreground TX FIFO at each start of frame interrupt.

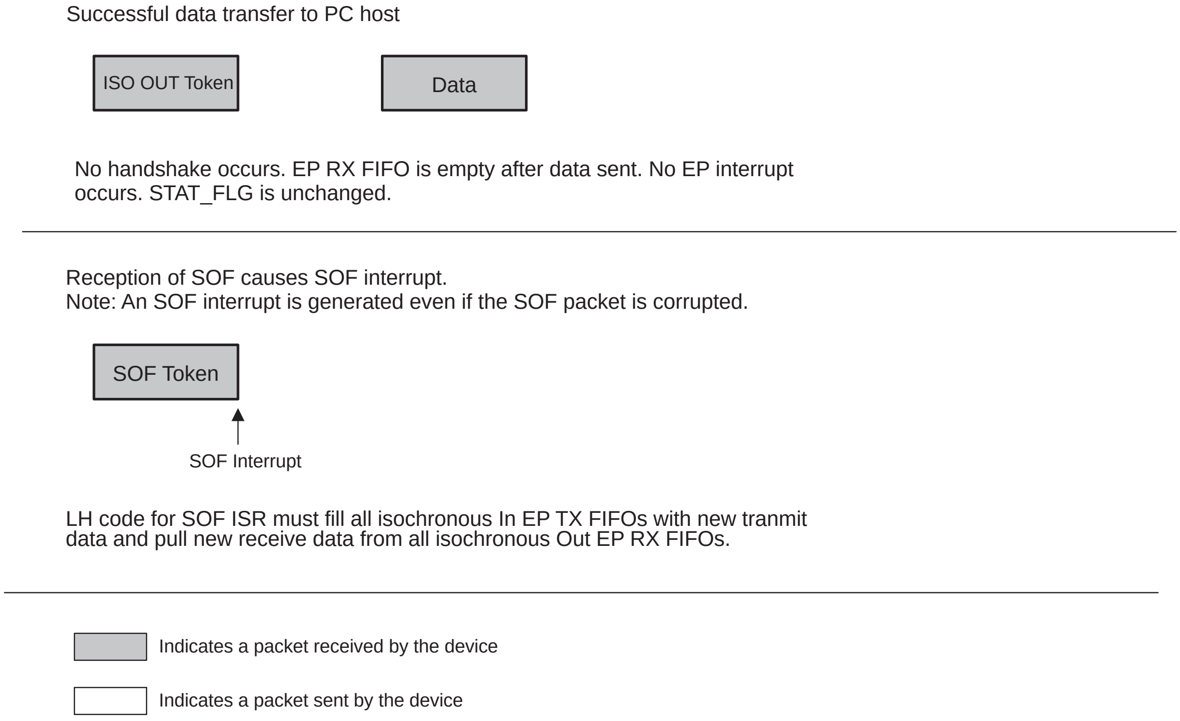
In response to the SOF interrupt, for each isochronous IN endpoint, MPU code selects the endpoint (via the EP_NUM register), and then fills the endpoint TX FIFO (via the DATA register). Once all the transmit data have been written to the FIFO, the MPU code must clear the EP_NUM.EP_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint.

At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to read from the background TX FIFO, and the MPU is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each start of frame (SOF). Because ISO endpoints implement double-buffering, ISO endpoints do not control access to the FIFOs via a CTRL.SET_FIFO_EN bit; the CTRL.SET_FIFO_EN and the STAT_FLG.FIFO_EN bits are not implemented for ISO IN endpoints.

Figure 7 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the MPU is generated as a result of an isochronous IN transaction, and there is no handshake phase. The SOF transaction causes an SOF interrupt to the MPU; it is assumed that the MPU refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 7. Isochronous IN Transaction Phases and Interrupts


Successful data transfer to PC host

No handshake occurs. EP RX FIFO is empty after data sent. No EP interrupt occurs. STAT_FLG is unchanged.

Reception of SOF causes SOF interrupt.
Note: An SOF interrupt is generated even if the SOF packet is corrupted.

LH code for SOF ISR must fill all isochronous In EP TX FIFOs with new tranmit data and pull new receive data from all isochronous Out EP RX FIFOs.

ISO OUT Token
Data
SOF Token
SOF Interrupt
Indicates a packet received by the device
Indicates a packet sent by the device

3.6.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the MPU to report handshake results for isochronous endpoints.

3.6.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an ISO IN transaction in the previous frame, and if data were present in TX FIFO to be sent at the IN transaction, the STAT_FLG.MISS_IN bit is asserted for the duration of the following frame. If the ISO IN endpoint is cleared in the middle of a USB transaction
to the background FIFO, the macro forces a bit stuffing error for the ISO transaction.

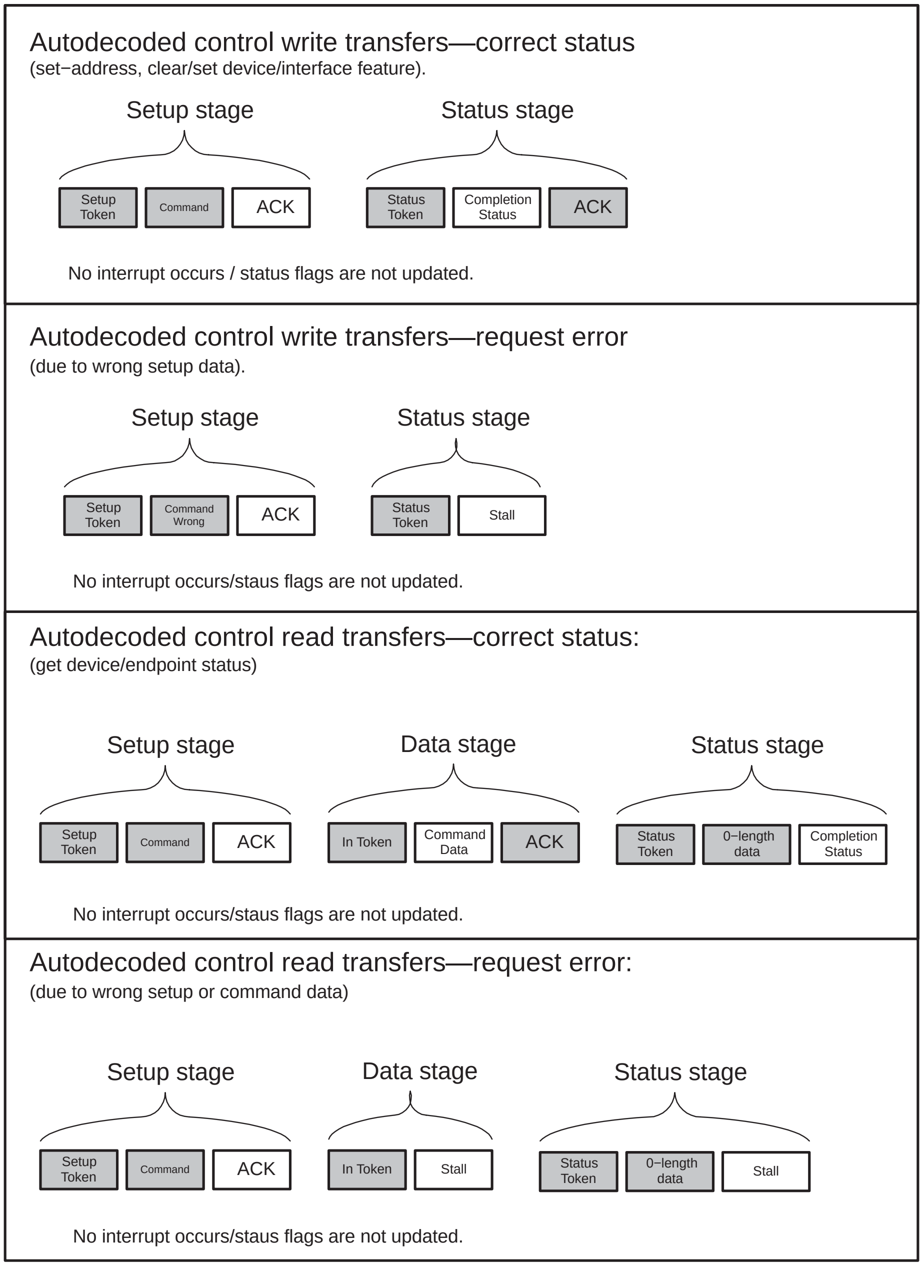
3.6.3 Isochronous IN Endpoint FIFO Error Conditions

If the MPU attempts to overfill the configured endpoint FIFO, data written to DATA register after the TX FIFO is full is lost, but any data that was successfully put into the FIFO is transmitted when that FIFO is the background FIFO and an IN transaction for that endpoint occurs. Because an ISO TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the MPU does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

3.7 Control Transfers on Endpoint 0

Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB device controller module is capable of autodecoding some control write and control read transfers. These operations are summarized in Figure 8 and Figure 9. An IN or an OUT transaction is received out of a control request. This transaction is automatically stalled by the core.

Figure 8. Stages and Transaction Phases of Autodecoded Control Transfers



Autodecoded control write transfers—correct status
(set−address, clear/set device/interface feature).

No interrupt occurs / status flags are not updated.

Autodecoded control write transfers—request error
(due to wrong setup data).

No interrupt occurs/staus flags are not updated.

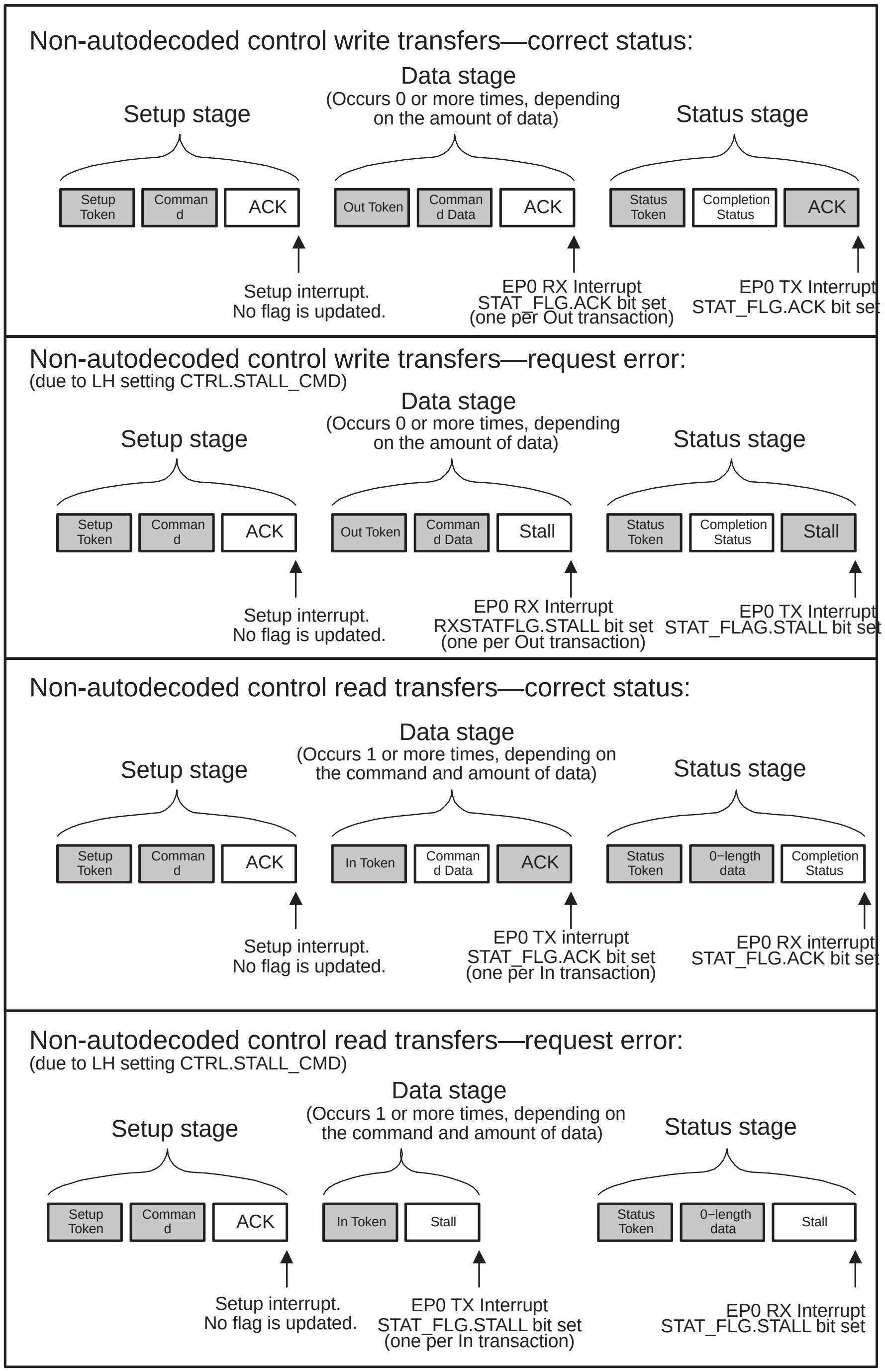
Autodecoded control read transfers—correct status:
(get device/endpoint status)

No interrupt occurs/staus flags are not updated.

Autodecoded control read transfers—request error:
(due to wrong setup or command data)

No interrupt occurs/staus flags are not updated.

Figure 9. Stages and Transaction Phases of Non-Autodecoded Control Transfers


Non-autodecoded control write transfers—correct status:
Setup stage
Data stage
(Occurs 0 or more times, depending on the amount of data)
Status stage
Setup Token
Command
ACK
Out Token
Command Data
ACK
Status Token
Completion Status
ACK
Setup interrupt. No flag is updated.
EP0 RX Interrupt STAT_FLG.ACK bit set (one per Out transaction)
EP0 TX Interrupt STAT_FLG.ACK bit set
Non-autodecoded control write transfers—request error:
(due to LH setting CTRL.STALL_CMD)
Setup stage
Data stage
(Occurs 0 or more times, depending on the amount of data)
Status stage
Setup Token
Command
ACK
Out Token
Command Data
Stall
Status Token
Completion Status
Stall
Setup interrupt. No flag is updated.
EP0 RX Interrupt RXSTATFLG.STALL bit set (one per Out transaction)
EP0 TX Interrupt STAT_FLAG.STALL bit set
Non-autodecoded control read transfers—correct status:
Setup stage
Data stage
(Occurs 1 or more times, depending on the command and amount of data)
Status stage
Setup Token
Command
ACK
In Token
Command Data
ACK
Status Token
0−length data
Completion Status
Setup interrupt. No flag is updated.
EP0 TX interrupt STAT_FLG.ACK bit set (one per In transaction)
EP0 RX interrupt STAT_FLG.ACK bit set
Non-autodecoded control read transfers—request error:
(due to LH setting CTRL.STALL_CMD)
Setup stage
Data stage
(Occurs 1 or more times, depending on the command and amount of data)
Status stage
Setup Token
Command
ACK
In Token
Stall
Status Token
0−length data
Stall
Setup interrupt. No flag is updated.
EP0 TX Interrupt STAT_FLG.STALL bit set (one per In transaction)
EP0 RX Interrupt STAT_FLG.STALL bit set

Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meaning but are not handled automatically by the core. The USB device controller block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the usual RX and TX control bits that affect transaction handshaking. A general USB interrupt to the MPU occurs at the end of each transaction of each stage of a control transfer. The MPU must perform the following actions to act on non-autodecoded control transfers:

- Process the setup phase setup USB interrupt. The MPU reads the control transfer command from the setup FIFO and decodes the command. For control reads, the MPU fetches the requested read data and places it (or as much of the read data as fits) into the endpoint 0 FIFO, and then enables the endpoint 0 FIFO. For control writes, the MPU code only enables the endpoint 0 FIFO. MPU code also sets any flags needed for processing endpoint 0 USB interrupts during the control transfer.
- Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data has been sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and when all control write data is available, interpret the write data and act on the write request. After handling the last data phase interrupt, the MPU must set the endpoint 0 control bits to signal the desired status to the host.
- Process the status stage endpoint 0 general USB interrupt. The MPU provides its completion status back to the USB host during this stage, either via status in the data phase of the transaction (for control write transfers) or via the handshake phase of the transaction (for control read transfers).

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 that have specific USB protocol meaning and are handled automatically by the USB device controller block without any intervention by the MPU. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (non-control transfer) transaction handshaking. No interrupt is asserted to the MPU during autodecoded control transfers.

If no USB1.1 specification defined request is associated with the data of the setup phase, request is stalled by the core and the MPU is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine whether it is an autodecoded or a non-autodecoded transfer and a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the MPU (if the MPU had control of the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (IRQ_SRC.setup).

In response to the setup interrupt, the MPU must select the setup FIFO by setting the EP_NUM.SETUP_SEL bit. This clears the IRQ_SRC.SETUP flag. The MPU must then read 8 bytes from the setup FIFO, clear the EP_NUM.SETUP_SEL bit, and confirm that IRQ_SRC.SETUP bit has not been reset by a new setup transaction. If the IRQ_SRC.SETUP flag is asserted, the MPU must discard the previously read data and handle the new

setup packet as explained above. Thus the MPU never misses a new occurring setup transaction (per USB 1.1 specification).

3.7.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If new address is different from 0, the device moves into addressed state (DEVSTAT.ADD set) if it was not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device remote wake-up feature, the DEVSTAT.R_WK_OK bit is set or cleared, as appropriate. If a set or clear interface feature occurs, the request is automatically stalled by the core because no feature is defined for interface (see the USB 1.1 specification).

Per the USB 1.1 specifications, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after setup stage, even if no status stage occurs.

Autodecoded Control Write Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, except if a corrupted packet is received, which the USB module ignores. The CTRL.SET_FIFO_EN and SYSCON2.STALL_CMD bits have no effect on handshaking.

Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

3.7.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET_ENDPOINT and DEVICE_STATUS. These control read transfers access information that is kept in registers inside the USB module, so MPU code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected appropriate status information (depending on the wIndex value in the setup stage data packet) during the data phase of the single IN transaction of the data stage, and provides ACK as the handshake for the status stage handshake phase. The MPU receives no interrupt.

Autodecoded Control Read Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, except if a corrupted token packet is received, which the USB module ignores. The CTRL.FIFO_EN and SYSCON2.STALL_CMD bits have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, status is STALLed and no interrupt is asserted to the MPU. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way; any data stage transaction from the host where a data error occurs is ignored.

It is possible that the USB host sends a GET_ENDPOINT/DEVICE_STATUS request with a bad parameter. If the autodecode mechanism senses a bad parameter in the setup stage data phase, the autodecode mechanism causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

3.7.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the SET_/CLEAR_ENDPOINT feature, SET_CONFIGURATION, SET_INTERFACE, SET_DESCRIPTOR and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages [setup, data (optional), and status].

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates an MPU general USB interrupt with the IRQ_SRC.SETUP flag set. The MPU must respond to this general USB interrupt by setting EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO via the DATA register, clear EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the MPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the MPU code interprets this request information and performs any application-specific activity needed because of the setup stage request. If there is one or more data stage for the transfer, the MPU must set the CTRL.SET_FIFO_EN bit for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control OUT endpoints applies. The MPU can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the MPU must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control bit STAT_FLG.FIFO_EN. The MPU can delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding STAT_FLG.FIFO_EN 0), or causing ACK handshaking by setting the CTRL.SET_FIFO_EN bit with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the MPU at completion of the status stage.

After a SET_CONFIGURATION request, the device moves in addressed or configured state as soon as the MPU sets the SYSCON2.DEV_CFG or SYSCON2.CLR_CFG bits.

Specific MPU Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate CTRL.SET_HALT control bit.

If the device receives a valid CLEAR_ENDPOINT halt feature request, it must set the appropriate CTRL.RESET_EP bit to clear the halt condition, FIFO flags, and reset data PID to DATA0 for the endpoint. If the specified endpoint number is 0, the MPU has only to set CTRL.CLR_HALT bit to clear the halt condition.

If the device receives a valid SET_CONFIGURATION request, it must reset all endpoints by setting the CTRL.RESET_EP control bits, set the SYSCON1.SELF_PWR bit to the appropriate value, and then set halt conditions for endpoints not used by the default interface set for the configuration. If the device was addressed when the SET_CONFIGURATION was received, the MPU must write 1 to the SYSCON2.DEV_CFG bit to allow the device to move into the configured state (DEVSTAT.CFG bit set). If the device was configured when the SET_CONFIGURATION was received, and the new configuration value is 0, the MPU must write 1 to the SYSCON2.CLR_CFG bit to allow the device to move back into the addressed state (DEVSTAT.CFG bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set, by setting CTRL.RESET_EP control bits, and then set halt conditions for endpoints not used by this interface.

Other MPU required actions are specific to the request and not detailed in this document.

Non-Autodecoded Control Write Transfer Handshaking

Setup stage transactions that are valid are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module, and there is no interrupt generated.

Data stage handshaking for non-autodecoded control write transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The MPU can delay completion of any transaction of the data stage by signaling NAK (via CTRL.SET_FIFO_EN bit not set). The USB specification requires that once STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the SYSCON2.STALL_CMD or the CTRL.SET_HALT (reflected in the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 STAT_FLG.FIFO_EN and SYSCON2.STALL_CMD bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a zero length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. Although NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host can choose to abort the control write after some number of NAKs.

Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs while dealing with the control write, which the MPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

3.7.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET_INTERFACE_STATUS, GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, SYNCH_FRAME and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages (setup, data, and status).

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a MPU general USB interrupt with the IRQ_SRC.SETUP flag set. The MPU must respond to this general USB interrupt by setting the EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The MPU must then read 8 bytes from the setup FIFO via the DATA register, clear the EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the MPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the MPU code interprets this request information and then prepares data for the IN transaction that follow. This includes placing the data being requested (or the first few bytes, if more than one FIFO worth of data is being returned) into the endpoint 0 FIFO, and setting the CTRL.SET_FIFO_EN bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control IN endpoints applies; the MPU can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, MPU code must move more data to the endpoint 0 FIFO, until the last bytes of the requested data have been provided. Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The MPU code must be able to operate correctly in this situation.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a zero-length data packet, and the MPU code must return its completion status for the control read standard request via standard handshaking mechanisms.

Note:

In the case of returning exactly what the host requested and that request was a multiple of the maximum packet size, no zero-length packet is required. A zero-length packet is required only when the amount of data the device has to return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size (source USB forum).

Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, the setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The SYSCON2.STALL_CMD and CTRL.SET_HALT (reflected through the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

The status stage is controlled by the CTRL.FIFO_EN and SYSCON2.STALL_CMD bits.

Successful completion of non-autodecoded control read transfers is indicated by the host sending an OUT token followed by an empty packet and the USB device controller responding with ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALLed is returned by the core and an interrupt is asserted.

Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs while dealing with the control read, which the MPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. The USB device controller module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the MPU.

3.7.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 55. Autodecoded Versus Non-Autodecoded Control Requests

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
GET_ STATUS	Device	Autodecoded (function of AUTODEC_ DIS register bit)	None	Device status is returned (SYSCON1.SELF_PWR and DEVSTAT.R_WK_OK bits).
	Interface	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the MPU.
	Endpoint	Autodecoded (function of AUTODEC_ DIS register bit)	None	The core automatically stalls the command if endpoint number is different from 0.
CLEAR/SET FEATURE	Device	Autodecoded (function of AUTODEC_ DIS register bit)	None (DS_CHG IT is asserted to the MPU after any DEVSTAT.R_ WK_OK bit modification).	The core handles the request.
	Interface	Autodecoded (function of AUTODEC_ DIS register bit)	None. (No feature is defined in USB 1.1 spec for interface. These requests are stalled).	Command is stalled anyway.

Notes:
1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.
3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 55. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
	Endpoint	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_CMD bit) if endpoint number/type/direction is not correct. The MPU must reset the EP after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, MPU must only clear or set halt condition; FIFO and data PID is always correct for next setup.	Command is passed to the MPU.
SET_ADDRESS	Device	Autodecoded	None (see Note 5) (Whether the device is addressed or not is available in DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_CHG interrupt to the MPU).	Default: device moves in the addressed state if address number is different from 0. Addressed: device takes the new address value or moves in default state if address number is 0. Configured: request is STALLed.
GET_DESCRIPTOR	All	Non-autodecoded	The MPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the MPU.

Notes: 1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).

2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.

3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.

4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.

5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 55. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
SET_ DESCRIPTOR	All	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if it does not support set descriptor requests.	Command is passed to the MPU.
GET/SET CONFIGURA-TION	Device	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if configuration number is not correct. If the request is SET_CONFIG, the MPU must reset all endpoints, halt endpoints not used by the default interface setting, set SYSCON1.SELF_PWR value if device is self-powered for the configuration set, and then set SYSCON2.DEV_CFG bit (if config nb is not 0), or set SYSCON2.CLR_CFG bit (if config nb is 0) before allowing status stage to complete. The device moves to configured state (if DEV_CFG set), or moves to addressed state (if CLR_CFG set) and a DS_CHG interrupt is asserted to the MPU.	Command is passed to the MPU.

Notes:

1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.
3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 55. Autodecoded Versus Non-Autodecoded Control Requests (Continued)

Request	Recipient	Status	MPU Required Action	Device Behavior if Device Is Not Configured
GET/SET INTERFACE	Interface	Non-autodecoded	The MPU must stall the command (via SYSCON2.STALL_ CMD bit) if interface/setting number is not correct. If the request is SET_ INTER-FACE, the MPU must reset endpoints used by the interface and halt endpoints not used by the interface setting before allowing status stage to complete.	Command is passed to the MPU.
SYNCH_ FRAME	Endpoint	Non-autodecoded	The MPU must stall the command if it does not support SYNCH_FRAME request, else write requested data in the endpoint 0 FIFO.	Command is passed to the MPU.

Notes:
1) Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).
2) If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the MPU must set halt feature for the endpoint. This does not happen if USB host works correctly.
3) If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.
4) Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.
5) During a SET_ADDRESS autodecoded command, only the 7 LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

3.7.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in the setup data packet.

During control reads, if the USB host requests more data than indicated in the setup packet, unexpected IN transaction is STALLed, causing STALL handshake for all remaining transactions of the transfer until next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLed. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because the MPU has not enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in the setup packet, the transfer is STALLed. If the USB host sends fewer bytes than were expected, the request is accepted. But if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the MPU has not enabled the TX FIFO.

3.8 USB Device Initialization

To allow communication between the device and a USB host, the MPU must configure the device by filling the configuration registers.

For each endpoint, the MPU must write on the dedicated register:

❑ Endpoint size

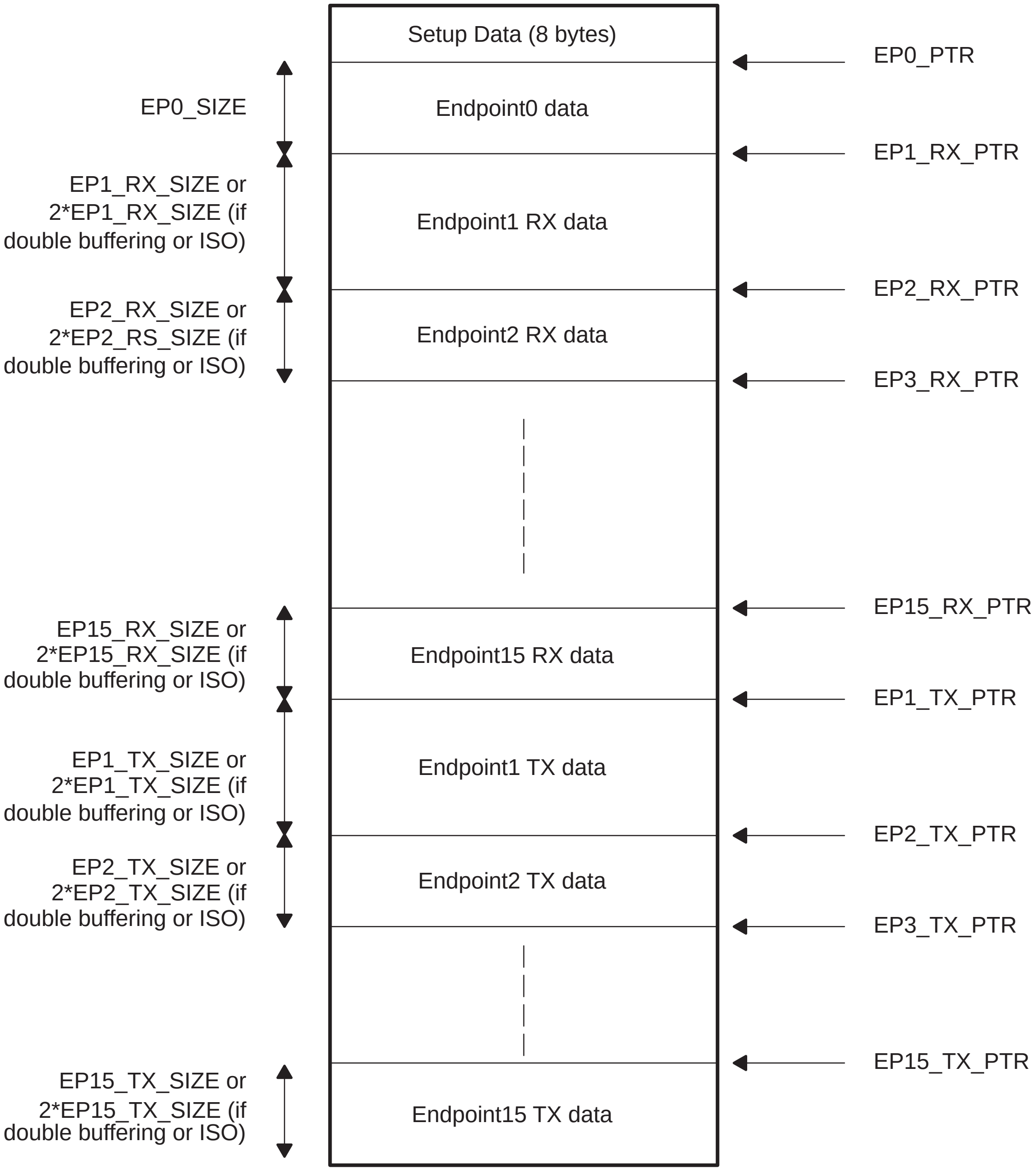
❑ Whether double-buffering is allowed for endpoint or not

❑ Endpoint type (ISO or non-ISO)

❑ Address of the pointer

System software must choose how to allocate the 2040 available bytes of USB device controller RAM to the USB endpoints. Receive endpoint size and type are configured using the EP1_RX through EP15_RX registers. Transmit endpoint size and type are configured using the EP1_TX through EP15_TX registers. Figure 10 shows an example of the RAM organization, obtained by following the flowchart shown in Figure 11.

Figure 10. Example of RAM Organization

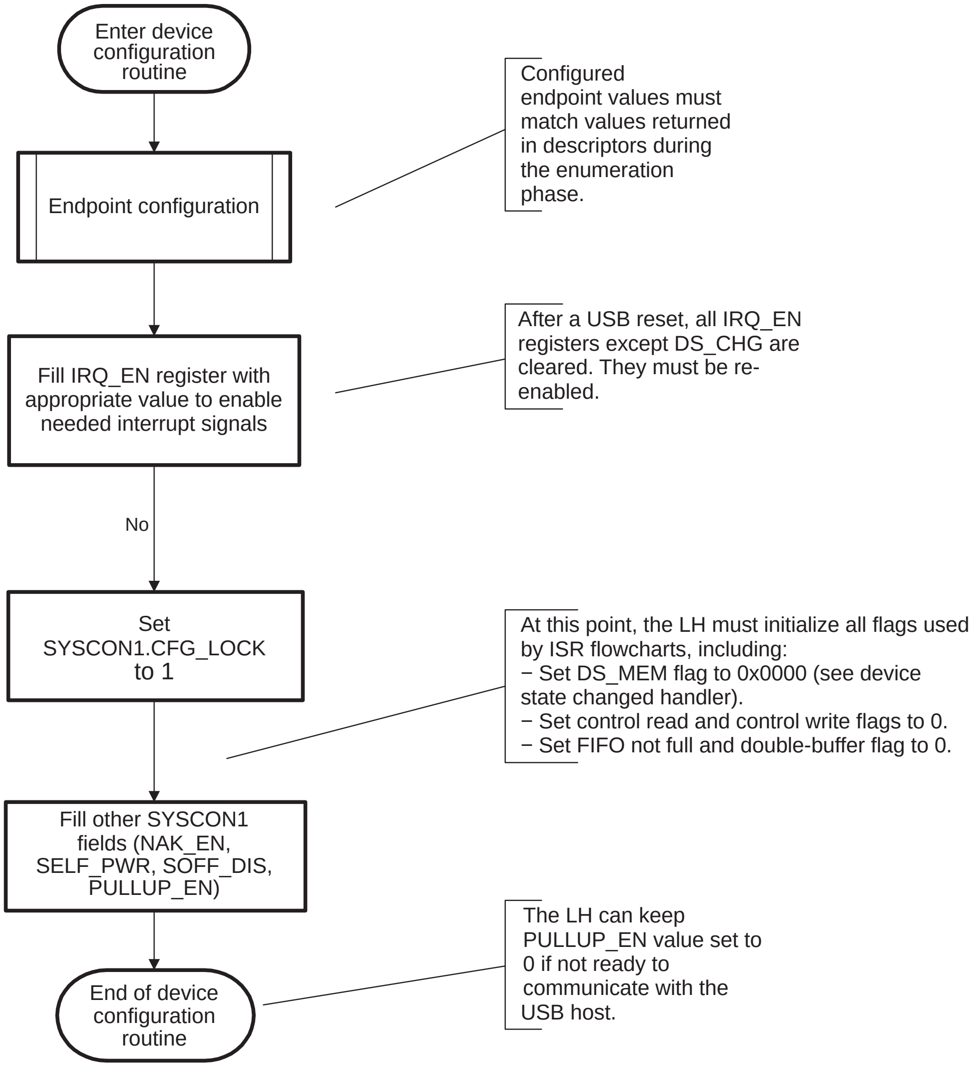

Setup Data (8 bytes)
EP0_PTR
EP0_SIZE
Endpoint0 data
EP1_RX_PTR
EP1_RX_SIZE or 2*EP1_RX_SIZE (if double buffering or ISO)
Endpoint1 RX data
EP2_RX_PTR
EP2_RX_SIZE or 2*EP2_RS_SIZE (if double buffering or ISO)
Endpoint2 RX data
EP3_RX_PTR
EP15_RX_PTR
EP15_RX_SIZE or 2*EP15_RX_SIZE (if double buffering or ISO)
Endpoint15 RX data
EP1_TX_PTR
EP1_TX_SIZE or 2*EP1_TX_SIZE (if double buffering or ISO)
Endpoint1 TX data
EP2_TX_PTR
EP2_TX_SIZE or 2*EP2_TX_SIZE (if double buffering or ISO)
Endpoint2 TX data
EP3_TX_PTR
EP15_TX_PTR
EP15_TX_SIZE or 2*EP15_TX_SIZE (if double buffering or ISO)
Endpoint15 TX data

Once the endpoints are configured, the MPU must set the SYSCON1.CFG_LOCK bit. If this bit is not set, all transactions are ignored by the core. Then, when the MPU is ready to communicate with the USB host, it must set the SYSCON1.PULLUP_EN bit. The MPU can wait until the DS_CHG attach interrupt has been detected and handled before setting the

SYSCON1.PULLUP_EN bit. The USB host cannot detect the device until this bit is set.

Figure 11 and Figure 12 show flowcharts for the configuration phase.

Figure 11. Device Configuration Routine


Enter device configuration routine
Endpoint configuration
Fill IRQ_EN register with appropriate value to enable needed interrupt signals
No
Set SYSCON1.CFG_LOCK to 1
Fill other SYSCON1 fields (NAK_EN, SELF_PWR, SOFF_DIS, PULLUP_EN)
End of device configuration routine

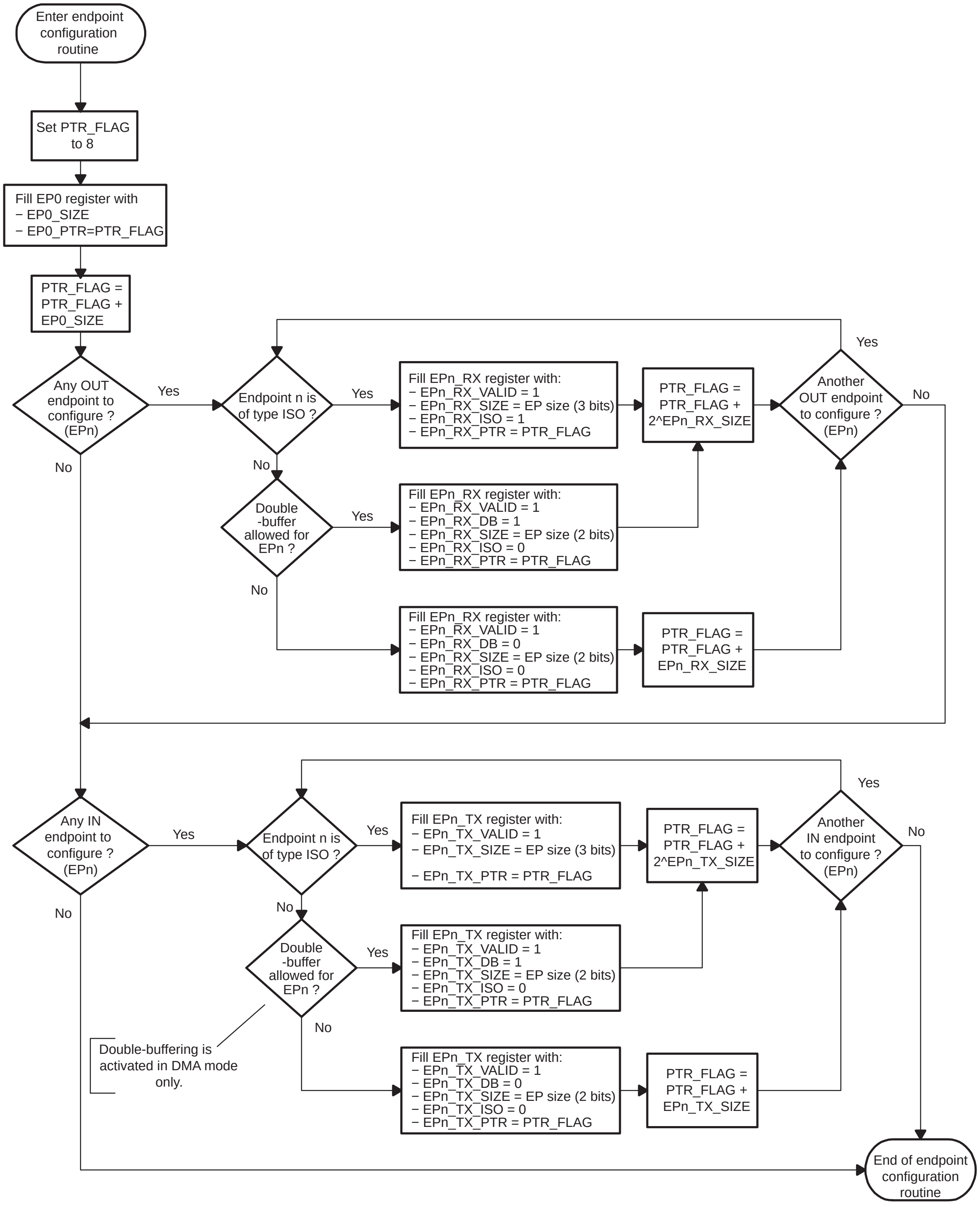
Configured endpoint values must match values returned in descriptors during the enumeration phase.

After a USB reset, all IRQ_EN registers except DS_CHG are cleared. They must be re-enabled.

At this point, the LH must initialize all flags used by ISR flowcharts, including:
− Set DS_MEM flag to 0x0000 (see device state changed handler).
− Set control read and control write flags to 0.
− Set FIFO not full and double-buffer flag to 0.

The LH can keep PULLUP_EN value set to 0 if not ready to communicate with the USB host.

Figure 12. Endpoint Configuration Routine


Enter endpoint configuration routine
Set PTR_FLAG to 8
Fill EP0 register with
− EP0_SIZE
− EP0_PTR=PTR_FLAG
PTR_FLAG = PTR_FLAG + EP0_SIZE
Any OUT endpoint to configure ? (EPn)
Yes
No
Endpoint n is of type ISO ?
Yes
No
Fill EPn_RX register with:
− EPn_RX_VALID = 1
− EPn_RX_SIZE = EP size (3 bits)
− EPn_RX_ISO = 1
− EPn_RX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^EPn_RX_SIZE
Another OUT endpoint to configure ? (EPn)
Yes
No
Double -buffer allowed for EPn ?
Yes
No
Fill EPn_RX register with:
− EPn_RX_VALID = 1
− EPn_RX_DB = 1
− EPn_RX_SIZE = EP size (2 bits)
− EPn_RX_ISO = 0
− EPn_RX_PTR = PTR_FLAG
Fill EPn_RX register with:
− EPn_RX_VALID = 1
− EPn_RX_DB = 0
− EPn_RX_SIZE = EP size (2 bits)
− EPn_RX_ISO = 0
− EPn_RX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + EPn_RX_SIZE
Any IN endpoint to configure ? (EPn)
Yes
No
Endpoint n is of type ISO ?
Yes
No
Fill EPn_TX register with:
− EPn_TX_VALID = 1
− EPn_TX_SIZE = EP size (3 bits)
− EPn_TX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^EPn_TX_SIZE
Another IN endpoint to configure ? (EPn)
Yes
No
Double -buffer allowed for EPn ?
Yes
No
Fill EPn_TX register with:
− EPn_TX_VALID = 1
− EPn_TX_DB = 1
− EPn_TX_SIZE = EP size (2 bits)
− EPn_TX_ISO = 0
− EPn_TX_PTR = PTR_FLAG
Double-buffering is activated in DMA mode only.
Fill EPn_TX register with:
− EPn_TX_VALID = 1
− EPn_TX_DB = 0
− EPn_TX_SIZE = EP size (2 bits)
− EPn_TX_ISO = 0
− EPn_TX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + EPn_TX_SIZE
End of endpoint configuration routine

3.9 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the MPU must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling (ISR flowchart).

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double buffering is allowed for the endpoint, setting the CTRL.SET_FIFO_EN bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

The MPU enters the prepare for USB RX transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether double-buffering is allowed or not is transparent to the MPU, unless both FIFOs are cleared through a CTRL.CLR_EP or CTRL.RESET_EP. In that case, and in the case where the MPU finishes to handle an interrupt without having set the CTRL.SET_FIFO_EN bit, the MPU must reenter the prepare for USB RX transfers routine.

For transmit endpoints, the MPU enters the prepare for endpoint n TX transfer routine, presented each time a new file must be transmitted from endpoint n to USB host. The MPU must not enter this routine until data written into TX FIFO from the previous transfer have all been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the CTRL.CLR_EP or CTRL.RESET_EP bits (see Figure 13 and Figure 14).

Note:

This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the MPU reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper usage of the module, you cannot prepare data on different endpoints at the same time. You can enter a routine once the routine is not being used by another endpoint (no parallelism); the EP_NUM register can be accessed via different routines.

Figure 13. Prepare for USB RX Transfers Routine

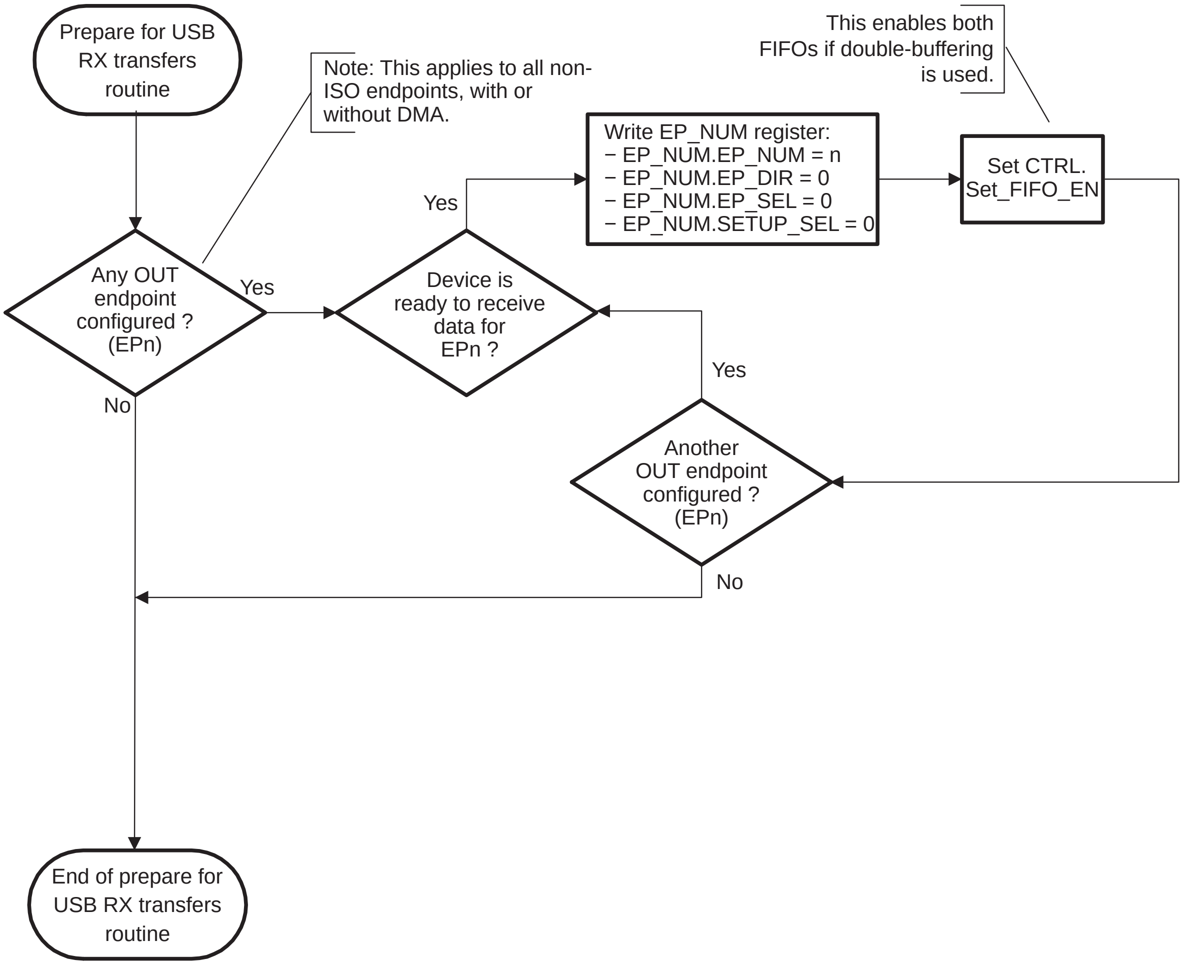
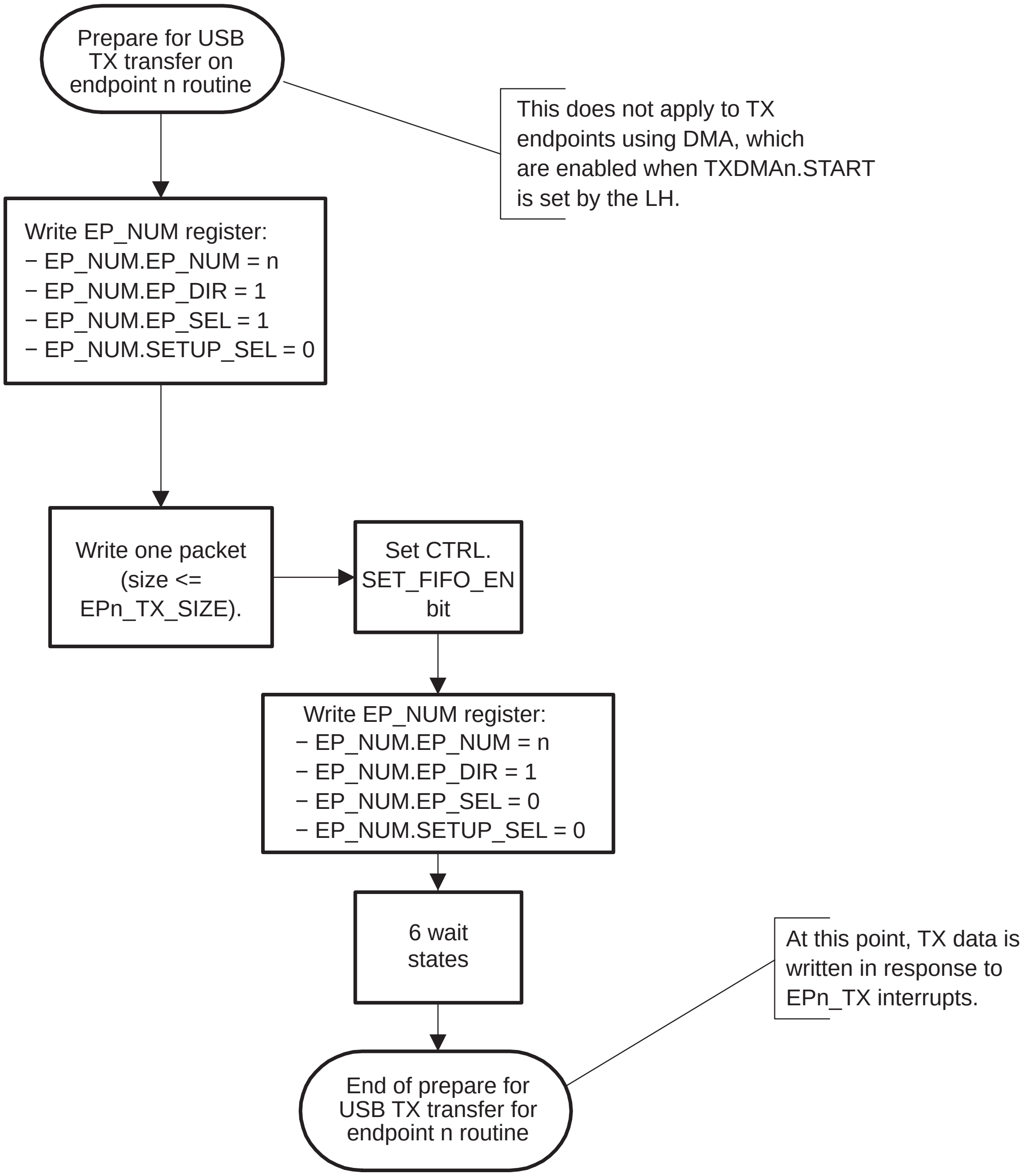

Prepare for USB RX transfers routine
Note: This applies to all non-ISO endpoints, with or without DMA.
This enables both FIFOs if double-buffering is used.
Write EP_NUM register:
− EP_NUM.EP_NUM = n
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
Set CTRL. Set_FIFO_EN
Yes
Any OUT endpoint configured ? (EPn)
Yes
Device is ready to receive data for EPn ?
Yes
No
Another OUT endpoint configured ? (EPn)
No
End of prepare for USB RX transfers routine

Figure 14. Prepare for TX Transfer on Endpoint n Routine


Prepare for USB TX transfer on endpoint n routine
This does not apply to TX endpoints using DMA, which are enabled when TXDMAn.START is set by the LH.
Write EP_NUM register:
− EP_NUM.EP_NUM = n
− EP_NUM.EP_DIR = 1
− EP_NUM.EP_SEL = 1
− EP_NUM.SETUP_SEL = 0
Write one packet (size <= EPn_TX_SIZE).
Set CTRL. SET_FIFO_EN bit
Write EP_NUM register:
− EP_NUM.EP_NUM = n
− EP_NUM.EP_DIR = 1
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
6 wait states
At this point, TX data is written in response to EPn_TX interrupts.
End of prepare for USB TX transfer for endpoint n routine

3.10 USB Device Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are left to the engineers who write the MPU and USB host code. One USB-specific interrupt register is provided (IRQ_SRC), including:

- ❏ General USB interrupts (including endpoint 0, DMA and device states interrupts) on MPU level 2 IRQ_20
- ❏ Non-ISO endpoint-specific interrupt on MPU level 2 IRQ_30
- ❏ Start of frame (SOF) interrupt for ISO transactions on MPU level 2 IRQ_29

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some specialty interrupts generated because of USB device state modifications or DMA transfers. The ISR for the endpoint-specific interrupt must handle interrupts from the USB module that are generated because of USB activity for non-isochronous endpoints. The SOF ISR is responsible for handling isochronous endpoints and, if needed by the application, tracking the USB frame number. Many flowcharts are presented in this chapter to provide guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this part suppose that the SYSCON1.NAK_EN bit is cleared.

Note:

A key assumption behind the flowcharts presented here is that the application provides separate buffers for each direction of endpoint, except for endpoint 0. The flowcharts read from these application buffers for IN transactions on TX endpoints and write to these application buffers for OUT transactions on RX endpoints.

The USB device controller does not support reentrant interrupts. Each USB device controller must be handled completely before handling another USB device controller interrupt. This restriction occurs because there is only one EP_NUM register, so endpoint control operations must be completed before working with another endpoint or endpoint direction.

3.11 Important Note on USB Device Interrupts

When an endpoint interrupt is asserted, the MPU writes the EP_NUM register with the EP_NUM.EP_SEL bit set to 1. The MPU must finish the interrupt handling before clearing the EP_NUM.EP_SEL bit, because clearing this bit clears the corresponding status bit in the STAT_FLG register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the MPU must not select and then unselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The MPU does not need to set EP_NUM.EP_SEL to 1 when setting CTRL.SET_FIFO_EN, CTRL.SET_HALT, and CTRL.CLR_HALT bits.

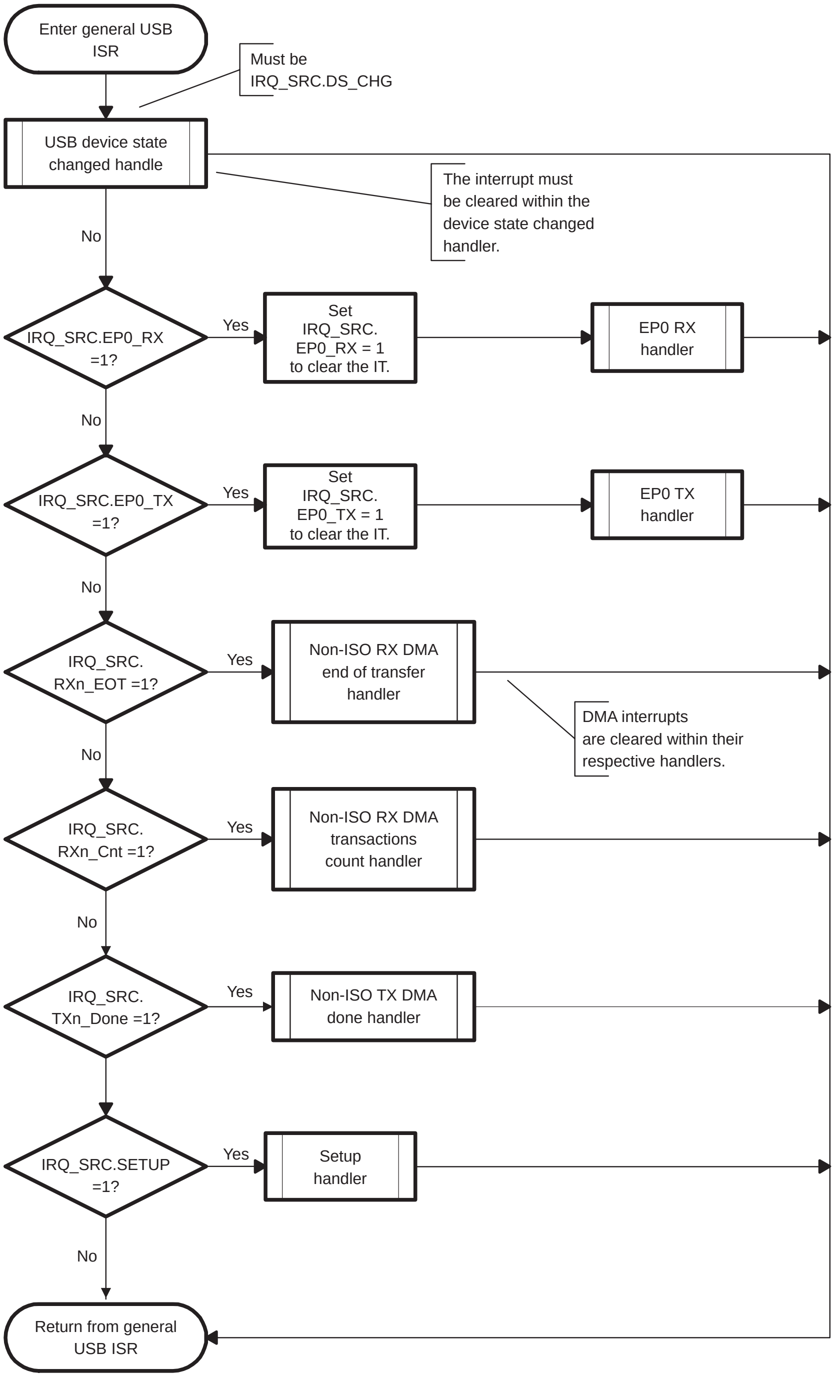
The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction has been handled. If a pending interrupt has not been handled when a new non-transparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed if the EP halt feature was set), so that the MPU never misses an ACKed transaction. If double-buffering is used for an endpoint, STAT_FLG is updated if there is zero or one interrupt pending for the endpoint, and is not updated if there are already two interrupts pending on the endpoint.

The MPU does not need to set the SYSCON1.NAK_EN bit during normal operation. However, for debugging process, this bit can be set when the MPU finishes handling an EP interrupt without having set the corresponding CTRL.SET_FIFO_EN bit. During TX transaction, if the SYSCON1.NAK_EN bit is set, the MPU must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by the NAK interrupt received while the MPU was writing the TX data.

3.12 Parsing General USB Device Interrupt

The general USB interrupt (MPU level 2 IRQ_20) ISR must parse the interrupt identifier register IRQ_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume, during enumeration phase) and control transfers on endpoint 0 or non-ISO DMA transfers in either receive or transmit mode. Multiple interrupts can be active at any time, and all must be dealt with by the ISR before returning from the ISR. Figure 15 shows an appropriate flowchart for parsing the general USB interrupts.

Figure 15. General USB Interrupt ISR Source Parsing Flowchart


Enter general USB ISR
Must be IRQ_SRC.DS_CHG
USB device state changed handle
The interrupt must be cleared within the device state changed handler.
No
IRQ_SRC.EP0_RX =1?
Yes
Set IRQ_SRC. EP0_RX = 1 to clear the IT.
EP0 RX handler
No
IRQ_SRC.EP0_TX =1?
Yes
Set IRQ_SRC. EP0_TX = 1 to clear the IT.
EP0 TX handler
No
IRQ_SRC. RXn_EOT =1?
Yes
Non-ISO RX DMA end of transfer handler
DMA interrupts are cleared within their respective handlers.
No
IRQ_SRC. RXn_Cnt =1?
Yes
Non-ISO RX DMA transactions count handler
No
IRQ_SRC. TXn_Done =1?
Yes
Non-ISO TX DMA done handler
IRQ_SRC.SETUP =1?
Yes
Setup handler
No
Return from general USB ISR

3.13 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the MPU cannot miss a setup transaction, even if it occurs during the data or status phase of another transfer (case of an aborted transfer). The setup parsing function captures the control transfer request information for use in determining which USB bus activity is needed and in controlling how the MPU must generate or respond to the control transfer. This information includes the following:

- bmRequestType
- bmRequest
- wValue
- wIndex
- wLength

The setup interrupt handler shown in Figure 16 is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 17 to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags are set by the setup interrupt handler, to be used during endpoint 0 interrupt handlers:

- Control read flag
- Control write flag

Figure 16. Setup Interrupt Handler

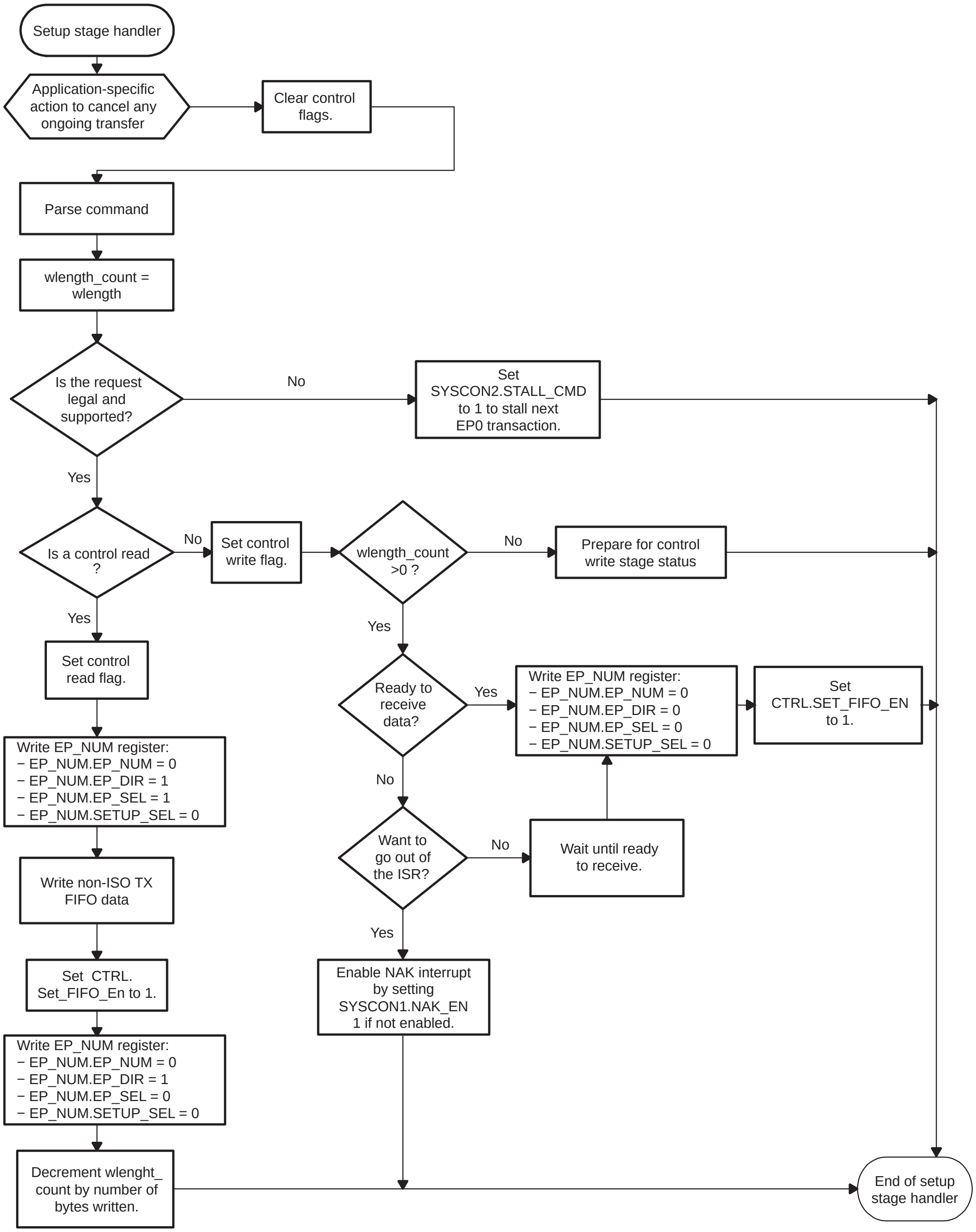
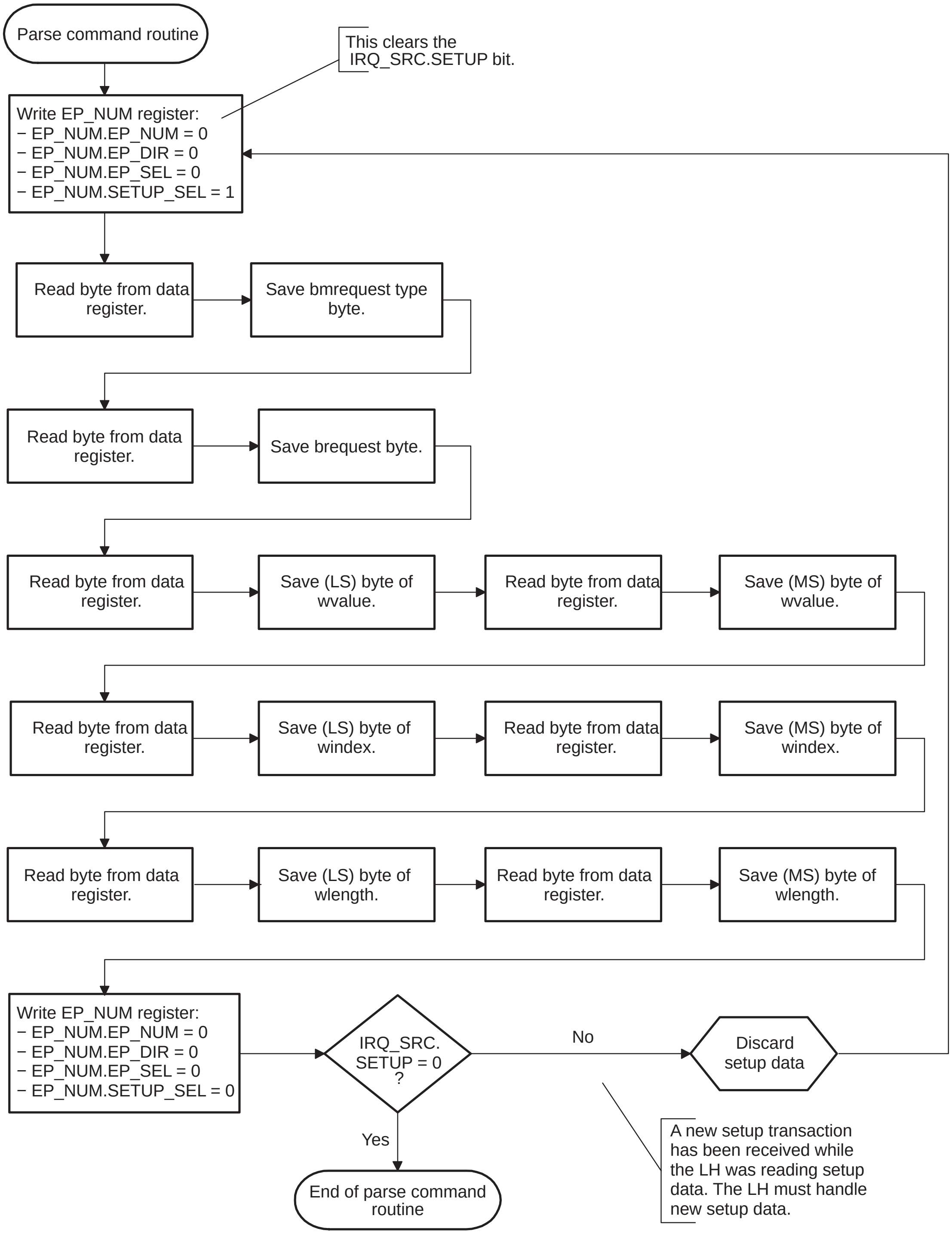


Figure 17. Parse Command Routine (Setup Stage Control Transfer Request)


Parse command routine
This clears the IRQ_SRC.SETUP bit.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 1
Read byte from data register.
Save bmrequest type byte.
Read byte from data register.
Save brequest byte.
Read byte from data register.
Save (LS) byte of wvalue.
Read byte from data register.
Save (MS) byte of wvalue.
Read byte from data register.
Save (LS) byte of windex.
Read byte from data register.
Save (MS) byte of windex.
Read byte from data register.
Save (LS) byte of wlength.
Read byte from data register.
Save (MS) byte of wlength.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
IRQ_SRC. SETUP = 0 ?
No
Discard setup data
Yes
End of parse command routine
A new setup transaction has been received while the LH was reading setup data. The LH must handle new setup data.

3.14 Endpoint 0 RX Interrupt Handler

Figure 18 shows the endpoint 0 RX portion of the general USB interrupt handler, which must handle general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 18. Endpoint 0 RX Interrupt Handler

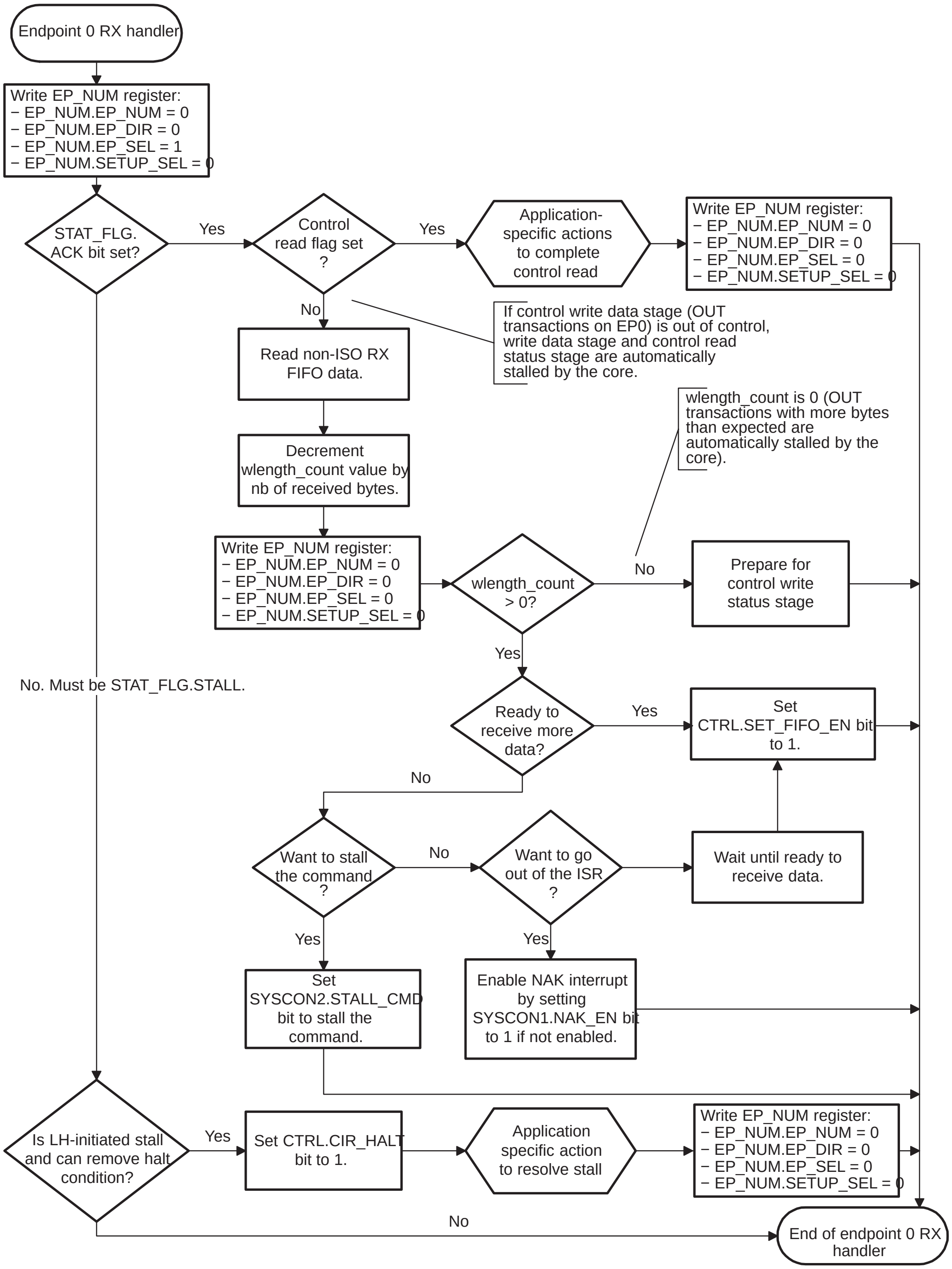
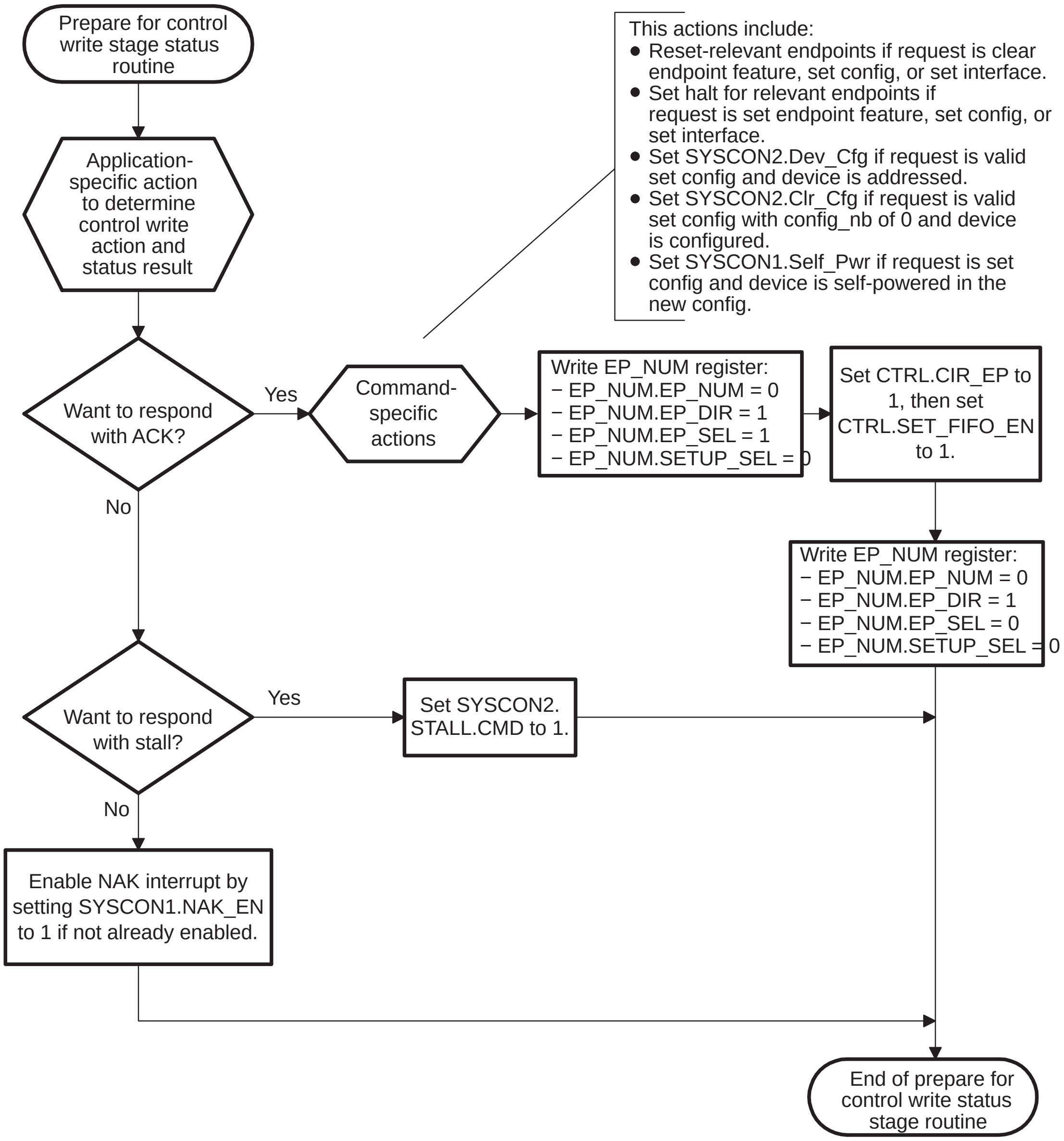

Endpoint 0 RX handler
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 1
− EP_NUM.SETUP_SEL = 0
STAT_FLG. ACK bit set?
Yes
Control read flag set ?
Yes
Application- specific actions to complete control read
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
No
If control write data stage (OUT transactions on EP0) is out of control, write data stage and control read status stage are automatically stalled by the core.
Read non-ISO RX FIFO data.
wlength_count is 0 (OUT transactions with more bytes than expected are automatically stalled by the core).
Decrement wlength_count value by nb of received bytes.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
wlength_count > 0?
No
Prepare for control write status stage
Yes
No. Must be STAT_FLG.STALL.
Ready to receive more data?
Yes
Set CTRL.SET_FIFO_EN bit to 1.
No
Want to stall the command ?
No
Want to go out of the ISR ?
Wait until ready to receive data.
Yes
Yes
Set SYSCON2.STALL_CMD bit to stall the command.
Enable NAK interrupt by setting SYSCON1.NAK_EN bit to 1 if not enabled.
Is LH-initiated stall and can remove halt condition?
Yes
Set CTRL.CIR_HALT bit to 1.
Application specific action to resolve stall
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
No
End of endpoint 0 RX handler

Figure 19. Prepare for Control Write Status Stage Routine


Prepare for control write stage status routine
Application-specific action to determine control write action and status result
Want to respond with ACK?
Yes
No
Command-specific actions
This actions include:
● Reset-relevant endpoints if request is clear endpoint feature, set config, or set interface.
● Set halt for relevant endpoints if request is set endpoint feature, set config, or set interface.
● Set SYSCON2.Dev_Cfg if request is valid set config and device is addressed.
● Set SYSCON2.Clr_Cfg if request is valid set config with config_nb of 0 and device is configured.
● Set SYSCON1.Self_Pwr if request is set config and device is self-powered in the new config.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 1
− EP_NUM.EP_SEL = 1
− EP_NUM.SETUP_SEL = 0
Set CTRL.CIR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 1
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
Want to respond with stall?
Yes
No
Set SYSCON2. STALL.CMD to 1.
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control write status stage routine

To support the SET_DESCRIPTOR command, when in commspecific actions, you must first reset all endpoints by selecting the endpoint and then clear it, unlock the configuration, charge the new one, lock the configuration, go in prepare for transfers, and enable all interrupts (see Figure 19).

3.15 Endpoint 0 TX Interrupt Handler

Figure 20 shows the TX portion of the general USB interrupt handler, which must handle general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt occurs signaling an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information (see Figure 21).

Figure 20. Endpoint 0 TX Interrupt Handler

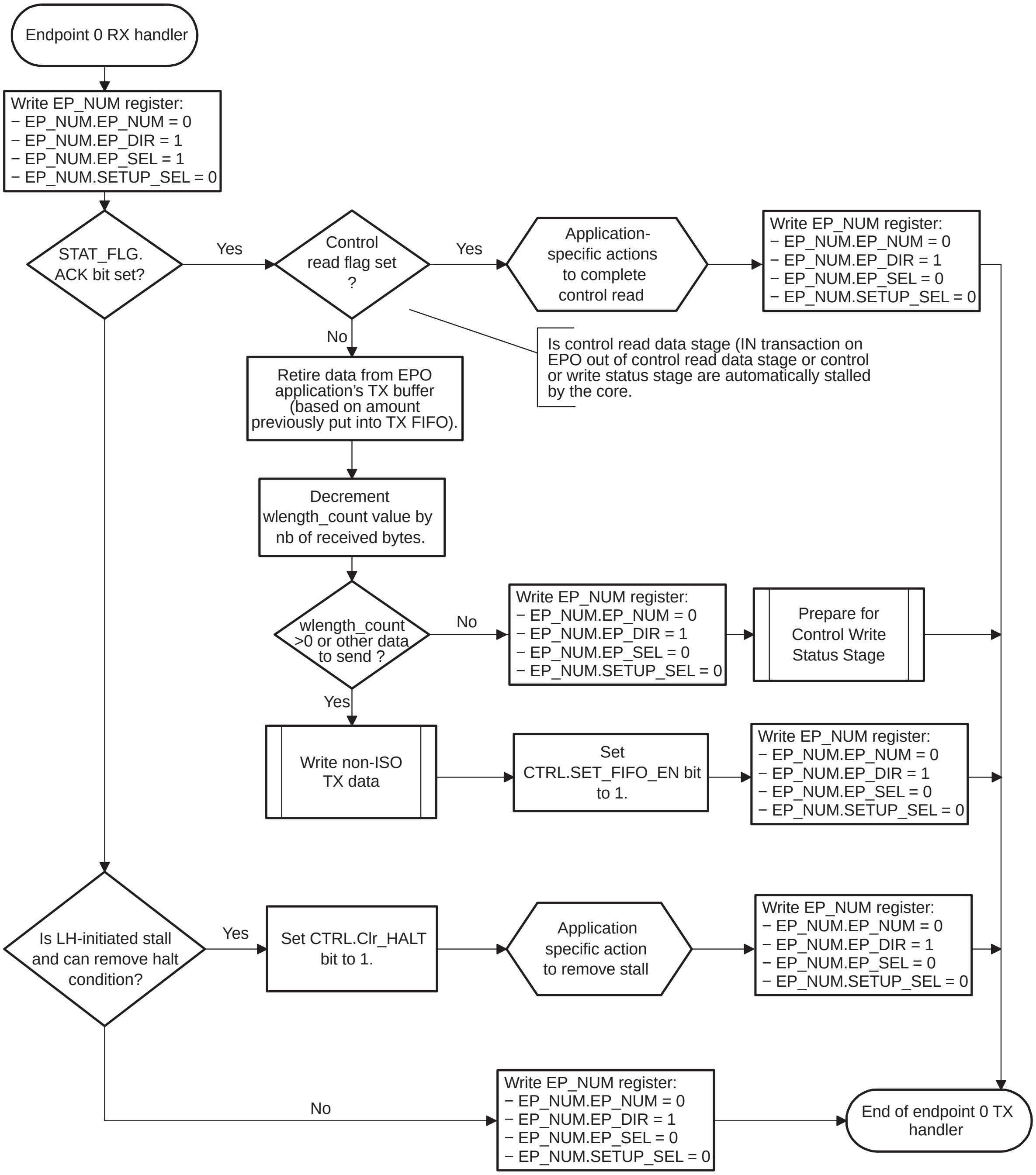
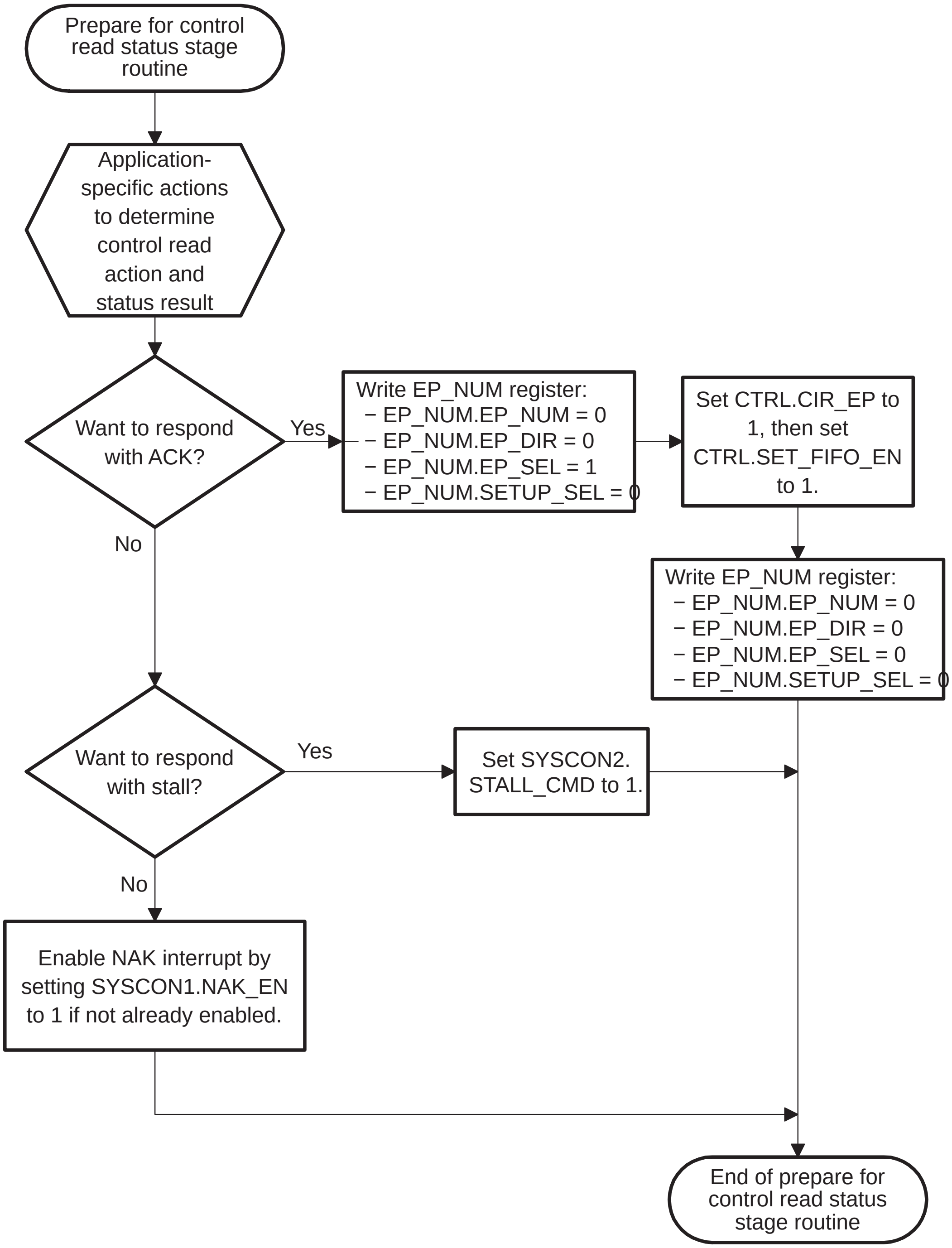


Figure 21. Prepare for Control Read Status Stage Routine


Prepare for control read status stage routine
Application-specific actions to determine control read action and status result
Want to respond with ACK?
Yes
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 1
− EP_NUM.SETUP_SEL = 0
Set CTRL.CIR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
Write EP_NUM register:
− EP_NUM.EP_NUM = 0
− EP_NUM.EP_DIR = 0
− EP_NUM.EP_SEL = 0
− EP_NUM.SETUP_SEL = 0
No
Want to respond with stall?
Yes
Set SYSCON2. STALL_CMD to 1.
No
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control read status stage routine

3.16 Device States Changed Handler

This section describes how USB device states and transitions states are decoded by the USB device controller and how they can be handled.

The state machine (see Figure 22) shows how the USB device controller device moves from one state to another state with respect to the USB1.1 specification. Attach/unattach transition is not shown in the transition flow.

Because SET_CONFIGURATION is not decoded by the core, the MPU has the responsibility to distinguish a SET_CONFIGURATION with a nonvalid configuration value from other SET_CONFIGURATION requests and to set SYSCON2.DEV_CFG only if the configuration value is valid (value 0 is nonvalid), when the device is in addressed state. When the device is in configured state, the MPU has the responsibility to set SYSCON2.CLR_CFG if the configuration number is 0 so that the device moves to addressed state.

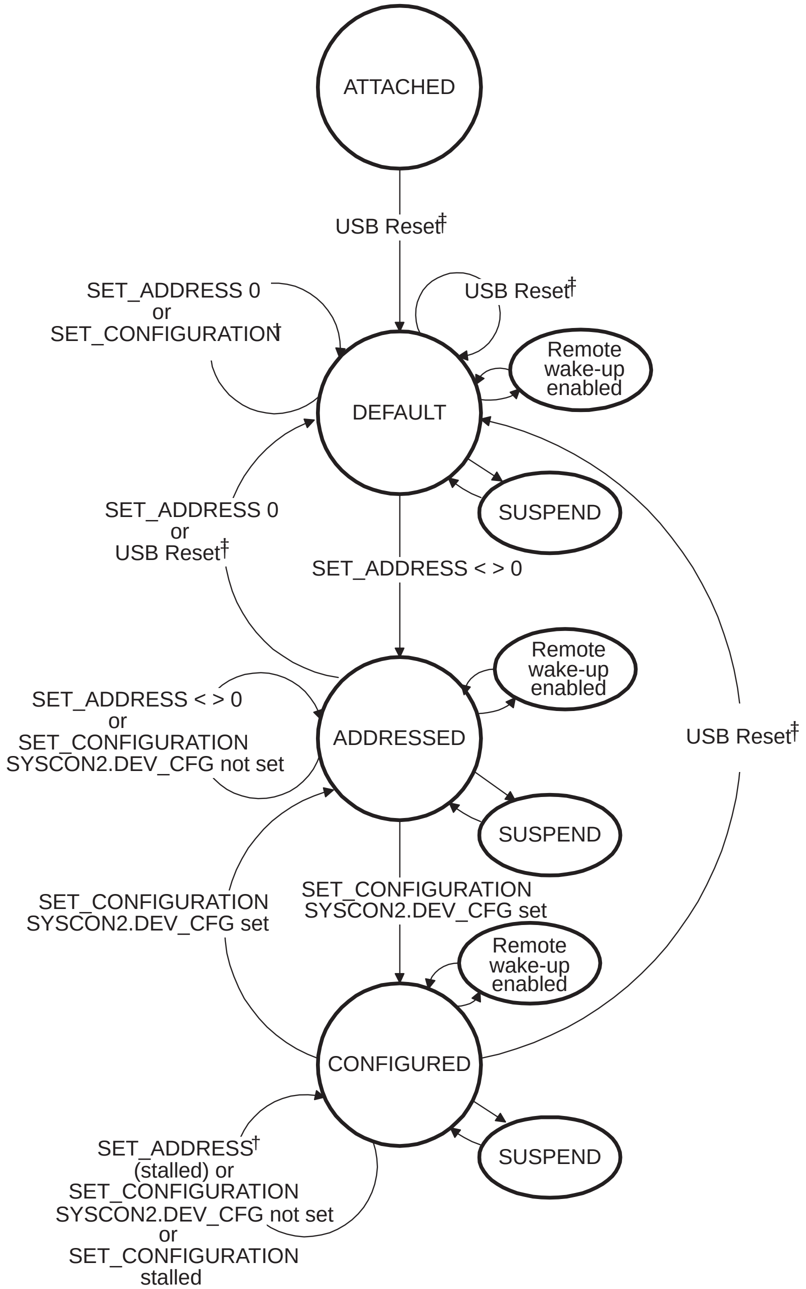
Device states are visible in the DEVSTAT register and are decoded as follows.

- ❑ Attached: The device is attached to the USB and powered.
- ❑ Default: The device is attached to the USB, is powered, and has been reset.
- ❑ Addressed: The device is attached to the USB, is powered, has been reset, and an address has been assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number other than 0.
- ❑ Configured: The device is attached to the USB, is powered, has been reset, has an address other than 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request, only if the MPU has set the SYSCON2.DEV_CFG bit (meaning the configuration is valid).
- ❑ Suspended: The device is at minimum default and has not seen bus activity for 5 ms.
- ❑ Reset: When set, the device is receiving a valid USB host reset.
- ❑ R_WK_OK: This bit is set (cleared) automatically after a valid SET_DEVICE_FEATURE (CLEAR_DEVICE_FEATURE) request from the USB host.

Any change in the DEVSTAT register bits triggers a device change interrupt (IRQ_SRC.DS_CHG), if enabled.

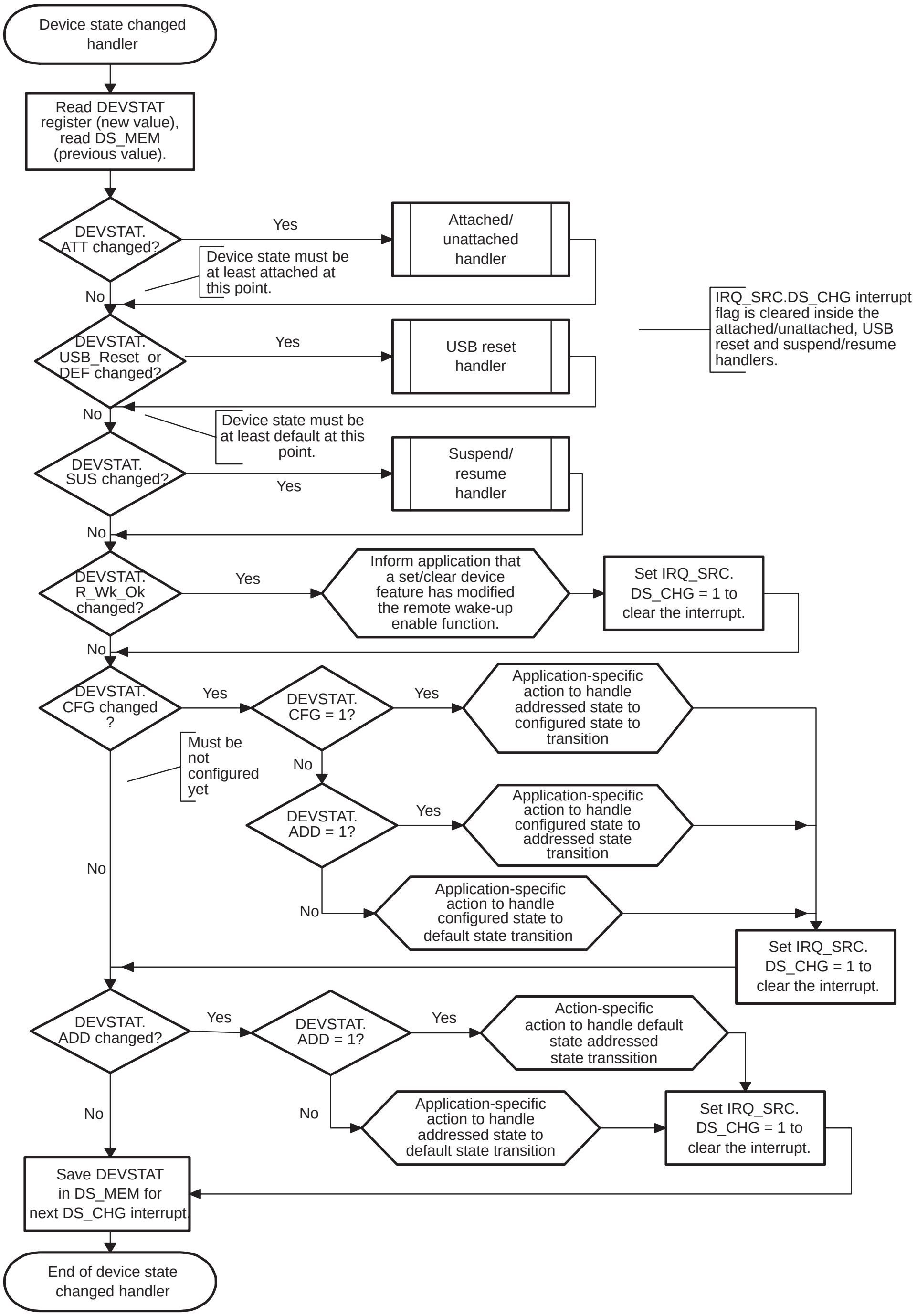
The device moves to addressed state after the status stage of a valid SET_ADDRESS, even if the status stage ACK handshake is received corrupted or not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the MPU sets the SYSCON2.CLR_CFG or SYSCON2.DEV_CFG bit (see Figure 23).

Figure 22. USB Device Controller Device State Transitions



Behavior not specified by USB 1.1 specifications (see chapter 9)
USB reset generates two interrupts (when USB reset is asserted and then when USB reset completes).
‡† No interrupt is asserted by the core for tansitions shown with dashed lines.

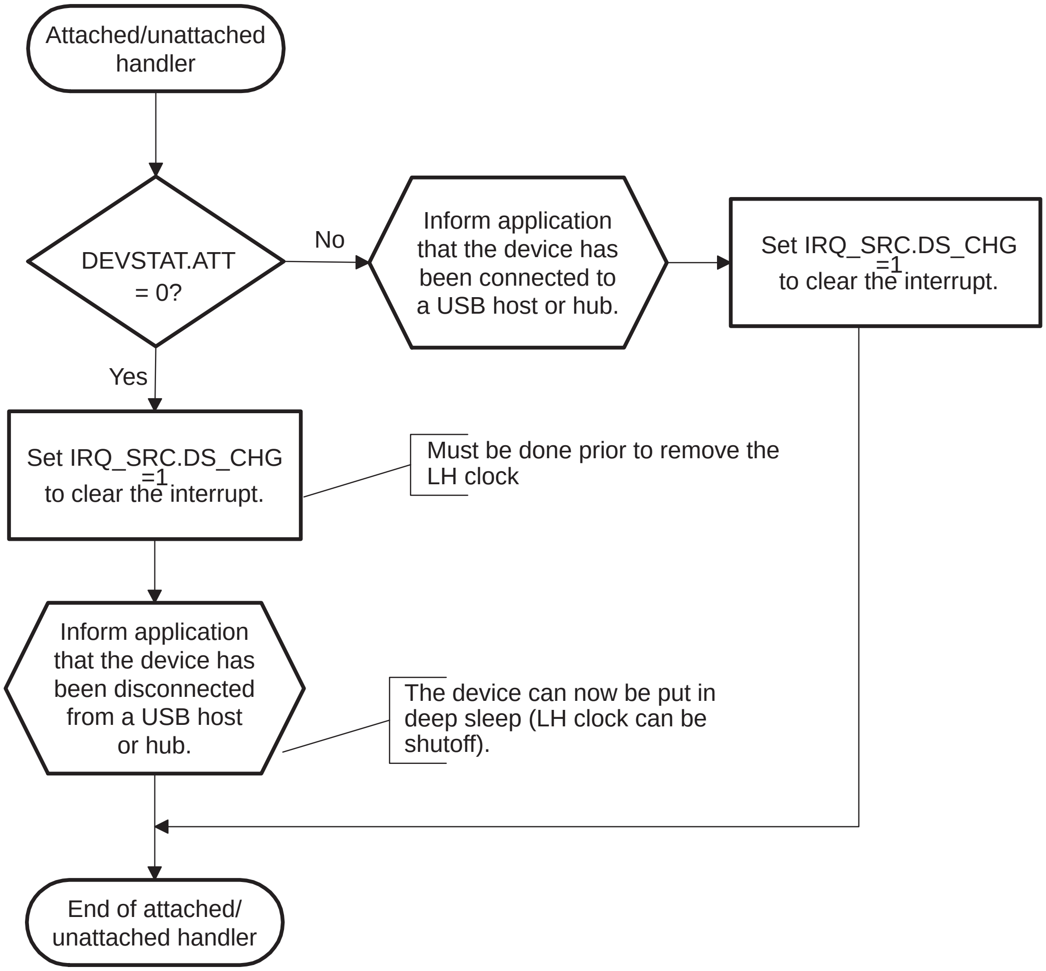
Figure 23. Typical Operation for USB Device State Changed Interrupt Handler


Device state changed handler
Read DEVSTAT register (new value), read DS_MEM (previous value).
DEVSTAT. ATT changed?
Yes
Attached/ unattached handler
Device state must be at least attached at this point.
No
IRQ_SRC.DS_CHG interrupt flag is cleared inside the attached/unattached, USB reset and suspend/resume handlers.
DEVSTAT. USB_Reset or DEF changed?
Yes
USB reset handler
No
Device state must be at least default at this point.
DEVSTAT. SUS changed?
Yes
Suspend/ resume handler
No
DEVSTAT. R_Wk_Ok changed?
Yes
Inform application that a set/clear device feature has modified the remote wake-up enable function.
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
No
DEVSTAT. CFG changed ?
Yes
DEVSTAT. CFG = 1?
Yes
Application-specific action to handle addressed state to configured state to transition
Must be not configured yet
No
DEVSTAT. ADD = 1?
Yes
Application-specific action to handle configured state to addressed state transition
No
No
Application-specific action to handle configured state to default state transition
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
DEVSTAT. ADD changed?
Yes
DEVSTAT. ADD = 1?
Yes
Action-specific action to handle default state addressed state transsition
No
No
Application-specific action to handle addressed state to default state transition
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
Save DEVSTAT in DS_MEM for next DS_CHG interrupt.
End of device state changed handler

3.17 Device States Attached/Unattached Handler

Device attached/unattached interrupts occur when the device detects that its VBUS has changed. System software can disable the USB device controller clock after IRQ_SRC.DS_CHG is cleared after servicing an unattached interrupt. Disabling the USB device controller clock before IRQ_SRC.DS_CHG is cleared can result in improper functionality for future USB device controller interrupts (see Figure 24).

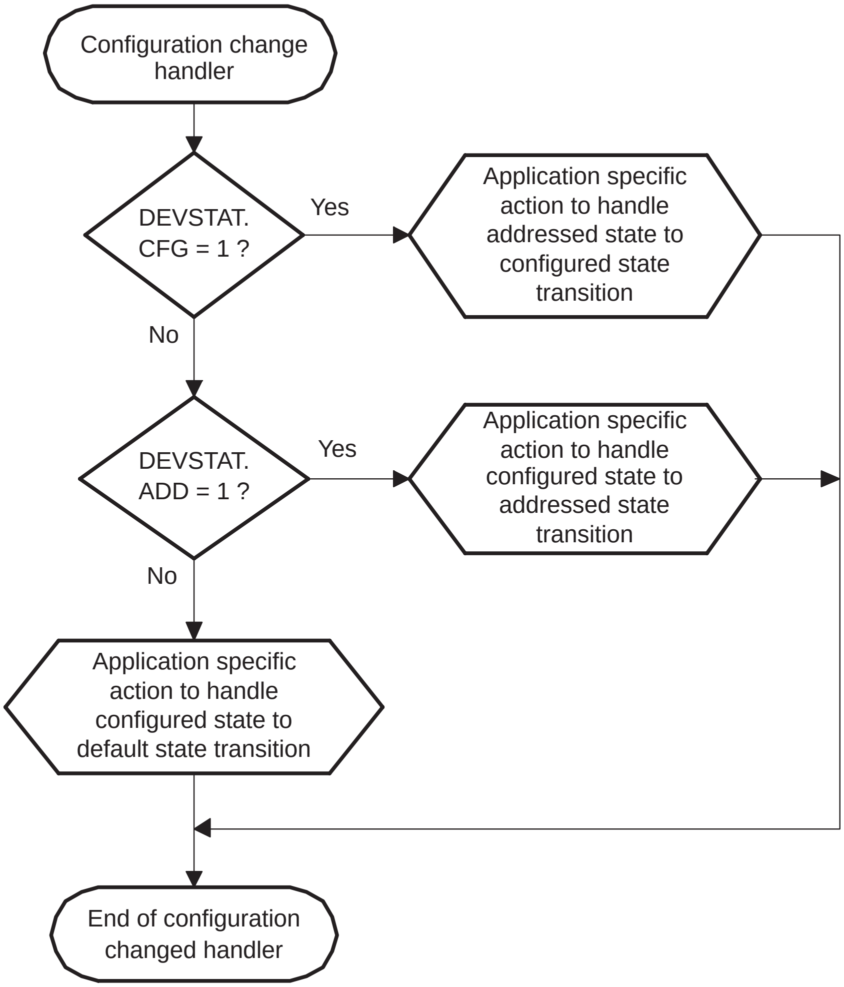
Figure 24. Attached/Unattached Handler


Attached/unattached handler
DEVSTAT.ATT = 0?
No
Inform application that the device has been connected to a USB host or hub.
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
Yes
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
Must be done prior to remove the LH clock
Inform application that the device has been disconnected from a USB host or hub.
The device can now be put in deep sleep (LH clock can be shutoff).
End of attached/ unattached handler

3.18 Device State Configuration Changed Handler

When a configuration changed interrupt occurs, the USB device has received a set configuration operation. When this occurs, the configuration-changed handler performs the operations shown in Figure 25.

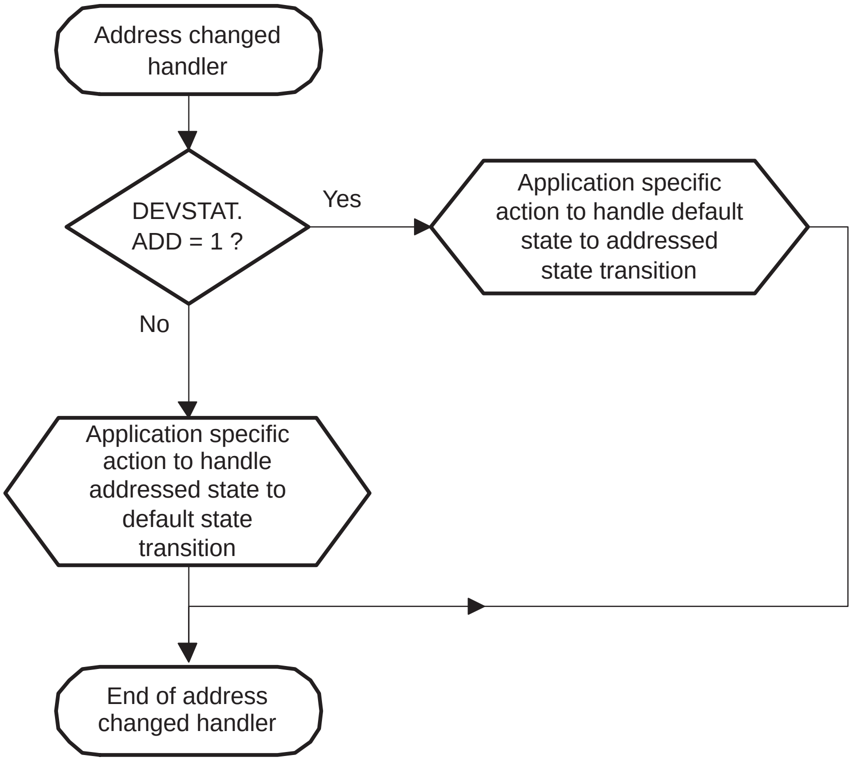
Figure 25. Configuration Changed Handler


Configuration change handler
DEVSTAT. CFG = 1 ?
Yes
Application specific action to handle addressed state to configured state transition
No
DEVSTAT. ADD = 1 ?
Yes
Application specific action to handle configured state to addressed state transition
No
Application specific action to handle configured state to default state transition
End of configuration changed handler

3.19 Device State Address Changed Handler

When a address changed interrupt occurs, the USB device has received a set address operation. When this occurs, the address-changed handler performs the operations shown in Figure 26.

Figure 26. Address Changed Handler

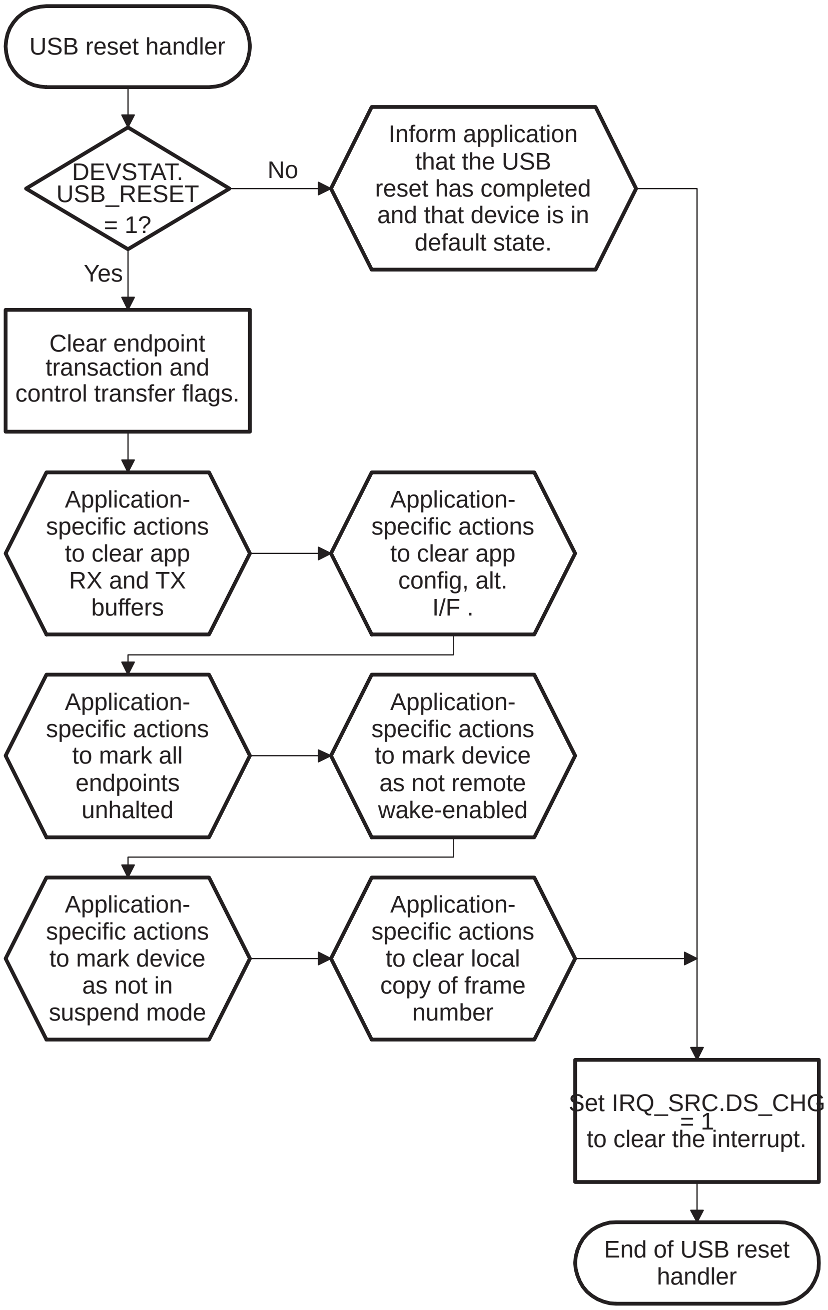

Address changed handler
DEVSTAT. ADD = 1 ?
Yes
Application specific action to handle default state to addressed state transition
No
Application specific action to handle addressed state to default state transition
End of address changed handler

3.20 USB Device Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the MPU (see Figure 27). The MPU responds to this interrupt by performing the following operations:

- ❑ Cancels any ongoing USB transaction and/or control transfer handling
- ❑ Clears any copies that the application has of configuration number or alternate interface numbers
- ❑ Clears any application-specific information relating to halted endpoints
- ❑ Clears any application-specific information relating to the remote wake enable flag
- ❑ Clears any application-specific information relating to the suspend mode flag
- ❑ Clears any application-specific copy of the frame number

Figure 27. USB Device Reset Handler Flowchart


USB reset handler
DEVSTAT. USB_RESET = 1?
No
Inform application that the USB reset has completed and that device is in default state.
Yes
Clear endpoint transaction and control transfer flags.
Application-specific actions to clear app RX and TX buffers
Application-specific actions to clear app config, alt. I/F .
Application-specific actions to mark all endpoints unhalted
Application-specific actions to mark device as not remote wake-enabled
Application-specific actions to mark device as not in suspend mode
Application-specific actions to clear local copy of frame number
Set IRQ_SRC.DS_CHG = 1 to clear the interrupt.
End of USB reset handler

3.21 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend mode. The MPU code must determine which and react appropriately (see Figure 28).

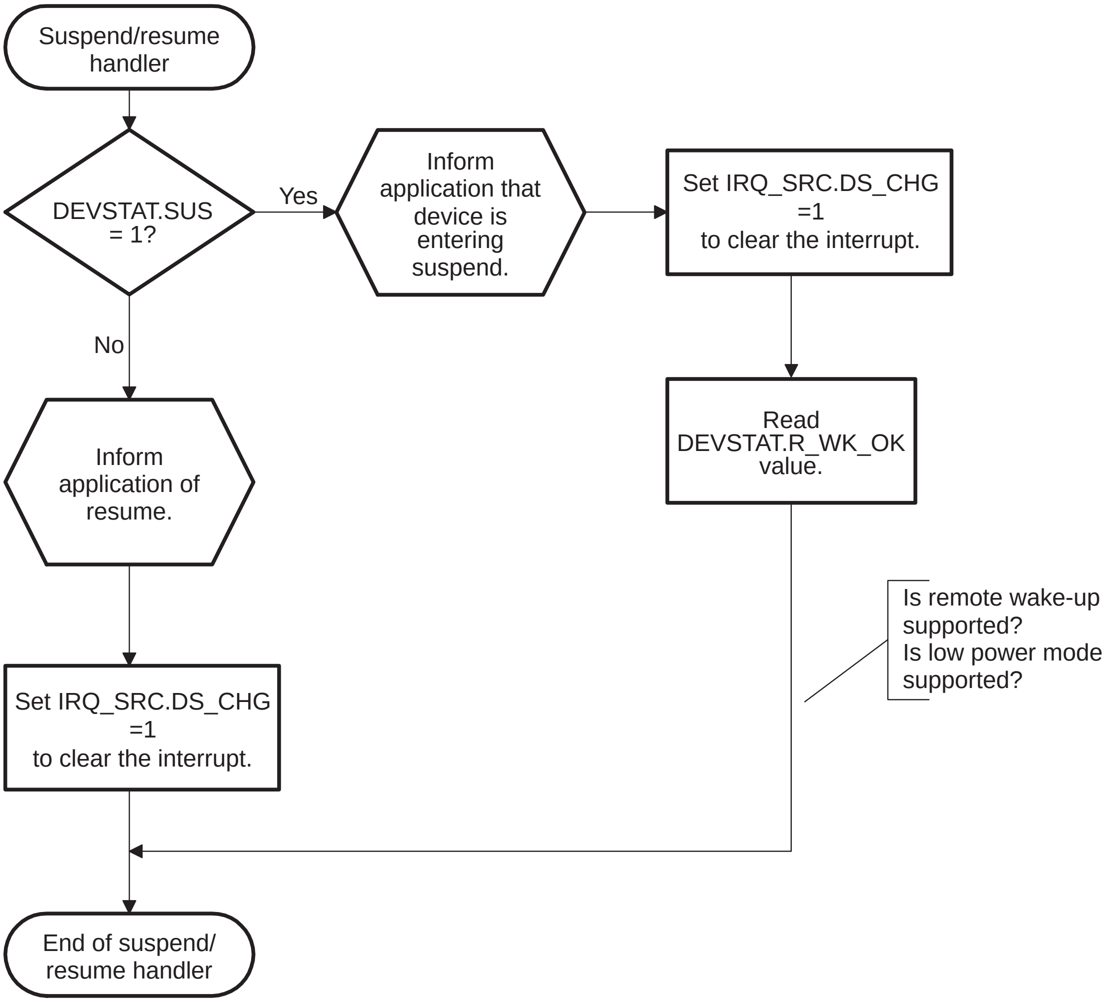
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the *USB Specification Version 1.1* T_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume).

If the MPU wants to wake the device from suspend mode and remote wake-up enable is set (DEVSTAT.R_WK_OK = 1), it must first turn its clock on (if stopped), and then set SYSCON2.RMT_WKP. The device then drives resume.

If shutoff is enabled (SYSCON1.SOFF_DIS=0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or MPU driven). Setting or not setting the SYSCON1.SOFF_DIS bit is part of the device configuration. However, the MPU can modify its value at suspend interrupt time if necessary.

A USB reset is also a valid way to exit suspend mode. But the suspend/resume handler and the USB reset handler do not have to take this into account, because three interrupts are generated in that case (1 for resume, 1 for reset, and 1 for end of reset).

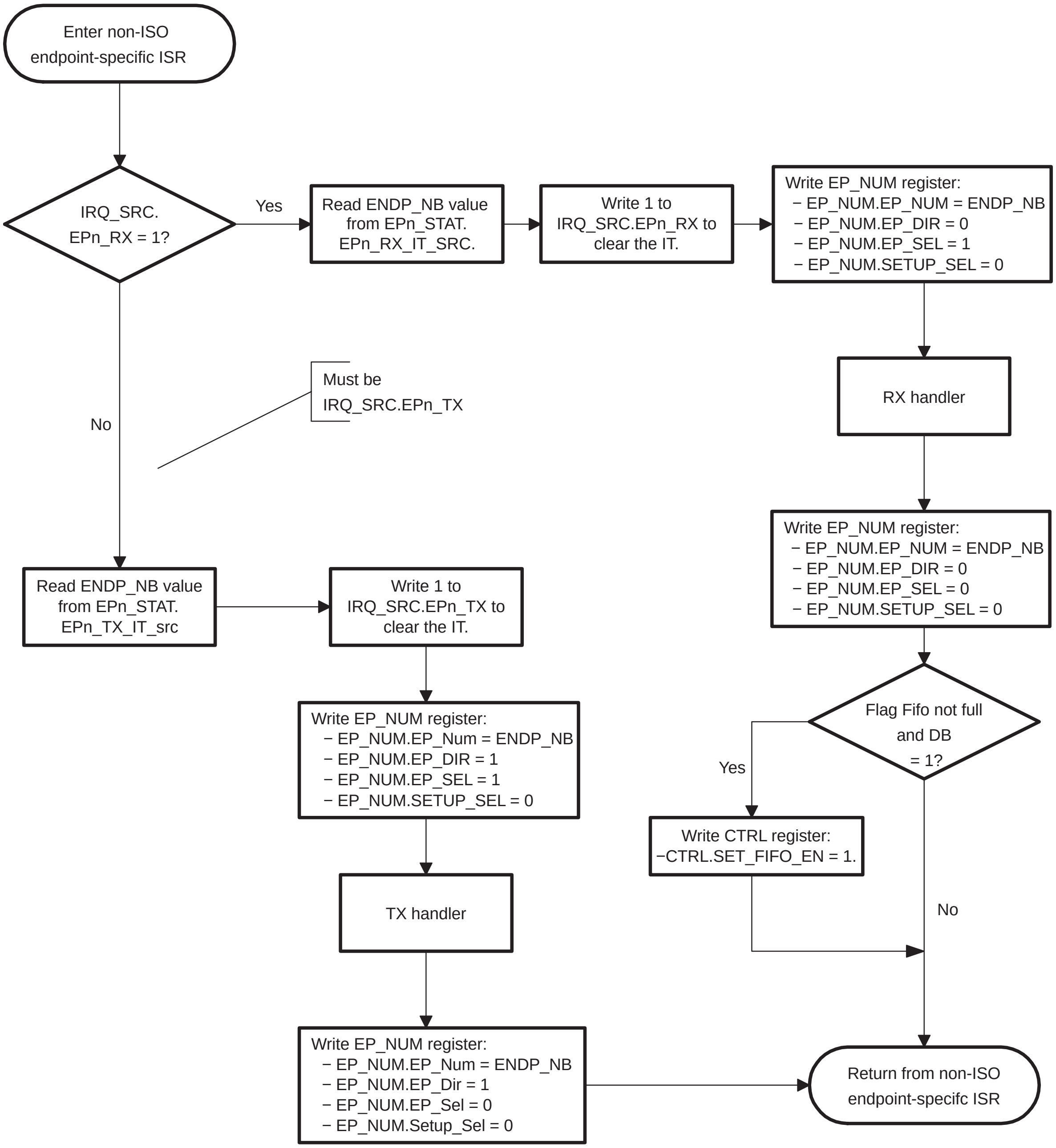
Figure 28. Typical Operation for USB Suspend/Resume General USB Interrupt Handler


Suspend/resume handler
DEVSTAT.SUS = 1?
Yes
Inform application that device is entering suspend.
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
No
Read DEVSTAT.R_WK_OK value.
Inform application of resume.
Is remote wake-up supported?
Is low power mode supported?
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
End of suspend/ resume handler

3.22 Parsing Non-ISO Endpoint-Specific Interrupt

The endpoint-specific interrupt ISR (also known as the non-isochronous interrupt, on MPU Level 2 IRQ_30) must parse the interrupt identifier register IRQ_SRC to determine the interrupts that are active (IRQ_SRC.EPN_RX, IRQ_SRC.EPN_TX, or both). The two interrupts can be active at any time. The ISR must then read the EPN_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time (see Figure 29).

Figure 29. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart



3.23 Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 30 shows the operations necessary to handle non-ISO, non-control OUT endpoint-specific receive interrupts. This flowchart shows two different RX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control receive endpoints, where the only differences are in the EP_NUM register value set and the selection of the proper application RX data buffer in the read non-ISO RX FIFO data routine (see Figure 31).

This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control writes.

Figure 30. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler

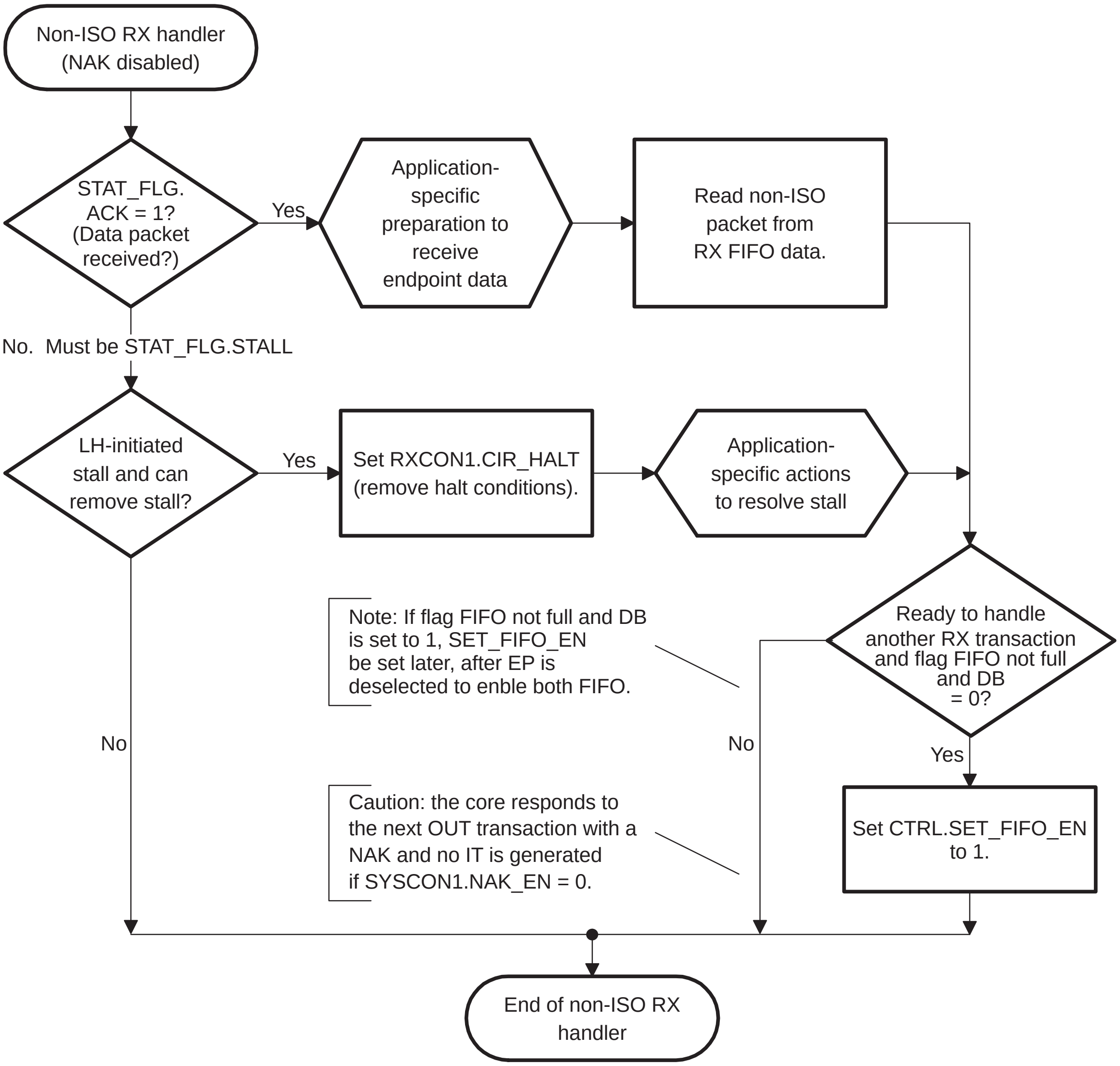
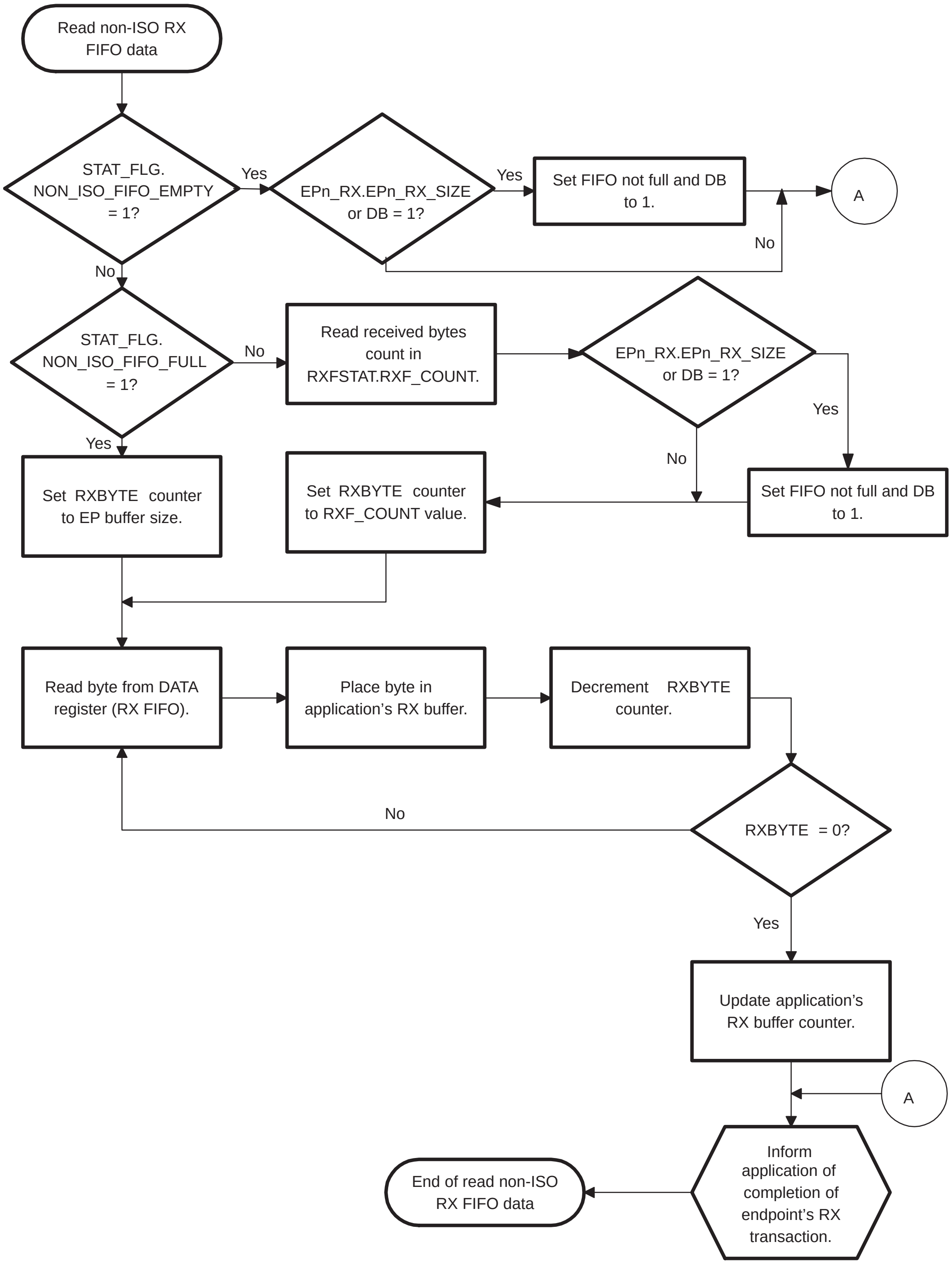

Non-ISO RX handler (NAK disabled)
STAT_FLG. ACK = 1? (Data packet received?)
Yes
Application-specific preparation to receive endpoint data
Read non-ISO packet from RX FIFO data.
No. Must be STAT_FLG.STALL
LH-initiated stall and can remove stall?
Yes
Set RXCON1.CIR_HALT (remove halt conditions).
Application-specific actions to resolve stall
Ready to handle another RX transaction and flag FIFO not full and DB = 0?
Note: If flag FIFO not full and DB is set to 1, SET_FIFO_EN be set later, after EP is deselected to enble both FIFO.
No
No
Yes
Caution: the core responds to the next OUT transaction with a NAK and no IT is generated if SYSCON1.NAK_EN = 0.
Set CTRL.SET_FIFO_EN to 1.
End of non-ISO RX handler

Figure 31. Read Non-Isochronous RX FIFO Data Flowchart


Read non-ISO RX FIFO data
STAT_FLG. NON_ISO_FIFO_EMPTY = 1?
Yes
EPn_RX.EPn_RX_SIZE or DB = 1?
Yes
Set FIFO not full and DB to 1.
A
No
No
STAT_FLG. NON_ISO_FIFO_FULL = 1?
No
Read received bytes count in RXFSTAT.RXF_COUNT.
EPn_RX.EPn_RX_SIZE or DB = 1?
Yes
No
Yes
Set RXBYTE counter to EP buffer size.
Set RXBYTE counter to RXF_COUNT value.
Set FIFO not full and DB to 1.
Read byte from DATA register (RX FIFO).
Place byte in application's RX buffer.
Decrement RXBYTE counter.
No
RXBYTE = 0?
Yes
Update application's RX buffer counter.
A
Inform application of completion of endpoint's RX transaction.
End of read non-ISO RX FIFO data

3.24 Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler

Figure 32 shows the operations necessary to handle non-ISO, non-control IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control transmit endpoints, where the only differences are in the EP_NUM register value set and in the routine selection of the application TX buffer (see Figure 33).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control reads.

Figure 32. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler

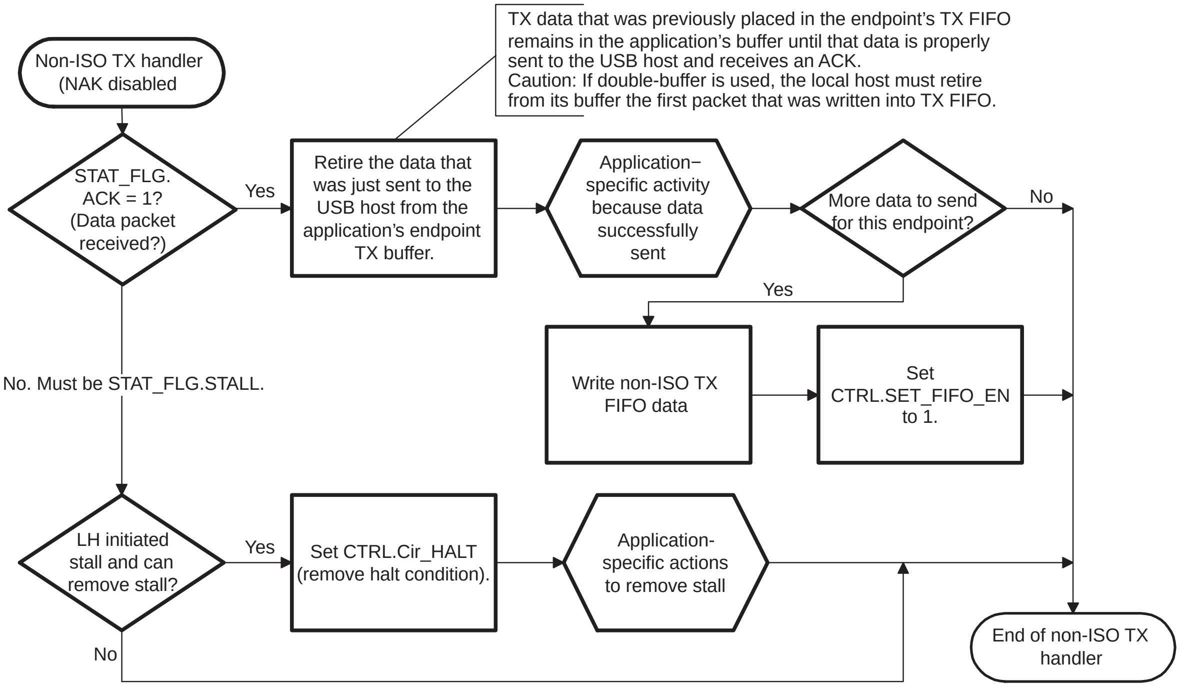
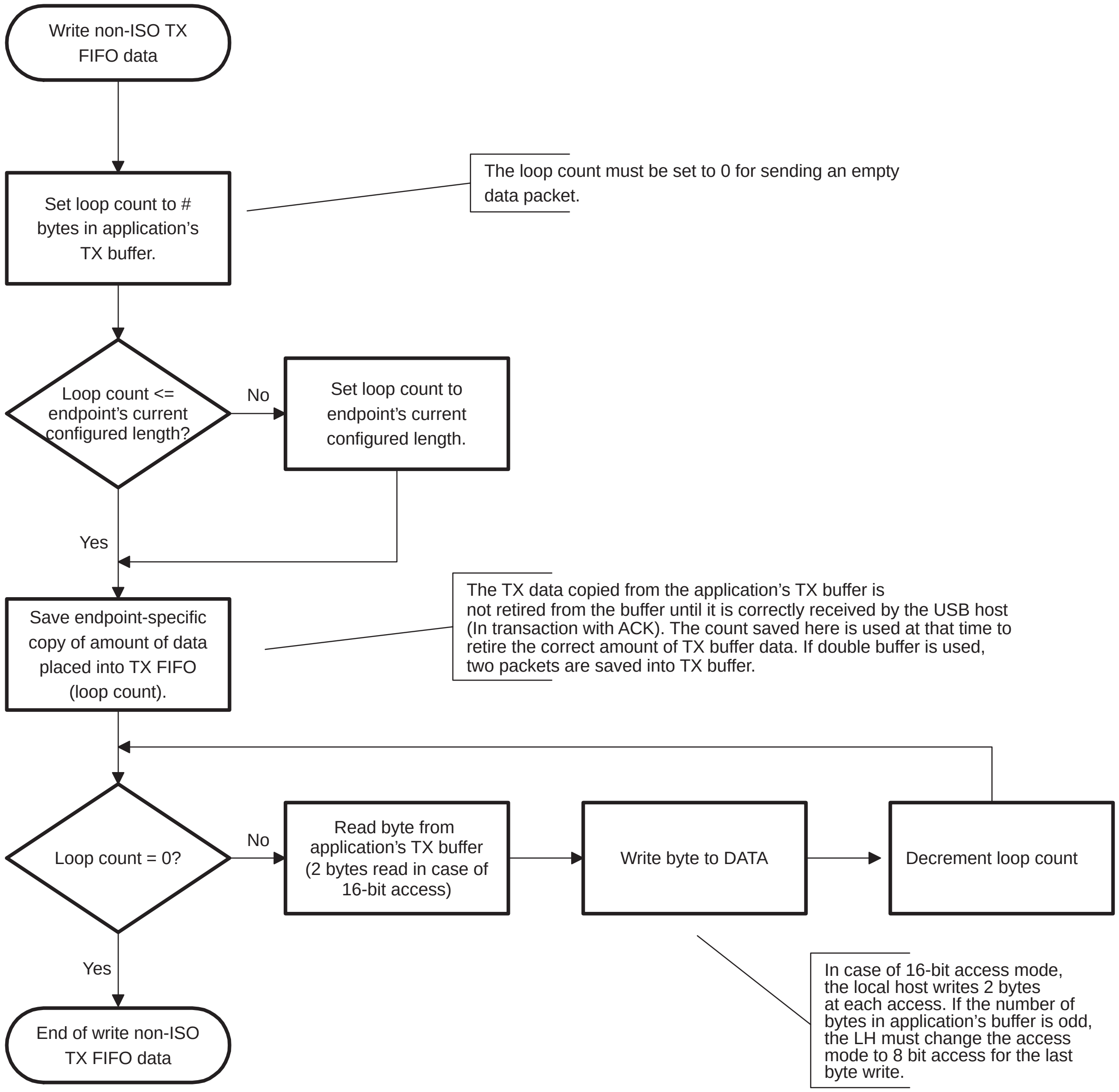

TX data that was previously placed in the endpoint's TX FIFO remains in the application's buffer until that data is properly sent to the USB host and receives an ACK.
Caution: If double-buffer is used, the local host must retire from its buffer the first packet that was written into TX FIFO.
Non-ISO TX handler (NAK disabled
STAT_FLG. ACK = 1? (Data packet received?)
Yes
Retire the data that was just sent to the USB host from the application's endpoint TX buffer.
Application− specific activity because data successfully sent
More data to send for this endpoint?
No
Yes
No. Must be STAT_FLG.STALL.
Write non-ISO TX FIFO data
Set CTRL.SET_FIFO_EN to 1.
LH initiated stall and can remove stall?
Yes
Set CTRL.Cir_HALT (remove halt condition).
Application- specific actions to remove stall
No
End of non-ISO TX handler

Figure 33. Write Non-Isochronous TX FIFO Data Flowchart


Write non-ISO TX FIFO data
Set loop count to # bytes in application's TX buffer.
Loop count <= endpoint's current configured length?
No
Set loop count to endpoint's current configured length.
Yes
Save endpoint-specific copy of amount of data placed into TX FIFO (loop count).
Loop count = 0?
No
Read byte from application's TX buffer (2 bytes read in case of 16-bit access)
Write byte to DATA
Decrement loop count
Yes
End of write non-ISO TX FIFO data

The loop count must be set to 0 for sending an empty data packet.

The TX data copied from the application's TX buffer is not retired from the buffer until it is correctly received by the USB host (In transaction with ACK). The count saved here is used at that time to retire the correct amount of TX buffer data. If double buffer is used, two packets are saved into TX buffer.

In case of 16-bit access mode, the local host writes 2 bytes at each access. If the number of bytes in application's buffer is odd, the LH must change the access mode to 8 bit access for the last byte write.

3.25 SOF Interrupt Handler

SOF interrupts to the MPU (also known as isochronous interrupts on MPU level 2 IRQ_29) occur once per USB frame. The MPU must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. Figure 34 shows the SOF ISR flowchart. The read ISO RX FIFO data and write ISO TX FIFO data procedures are shown Figure 35 and Figure 36, respectively.

Figure 34. SOF Interrupt Handler Flowchart

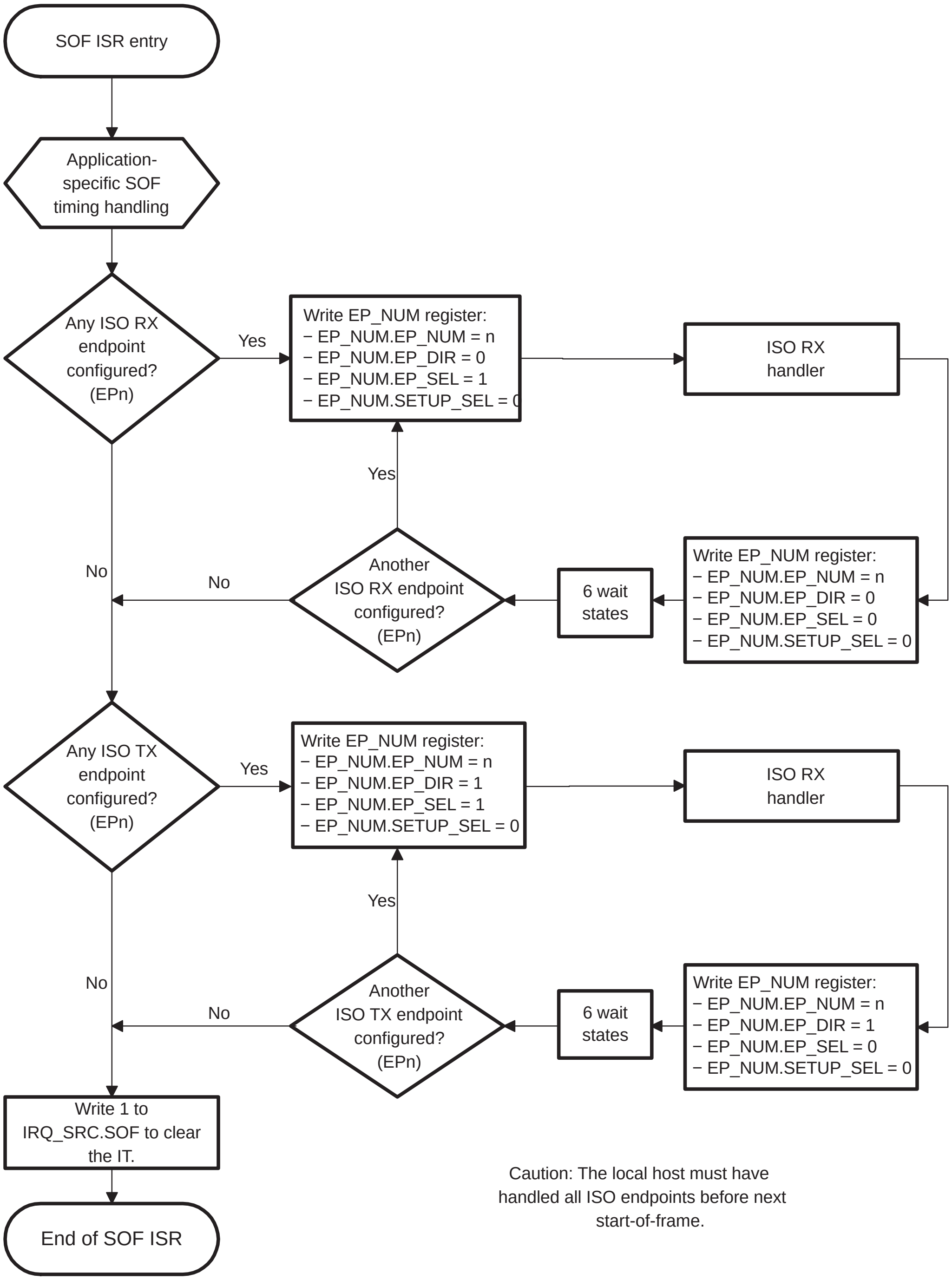


Figure 35. Read Isochronous RX FIFO Data Flowchart

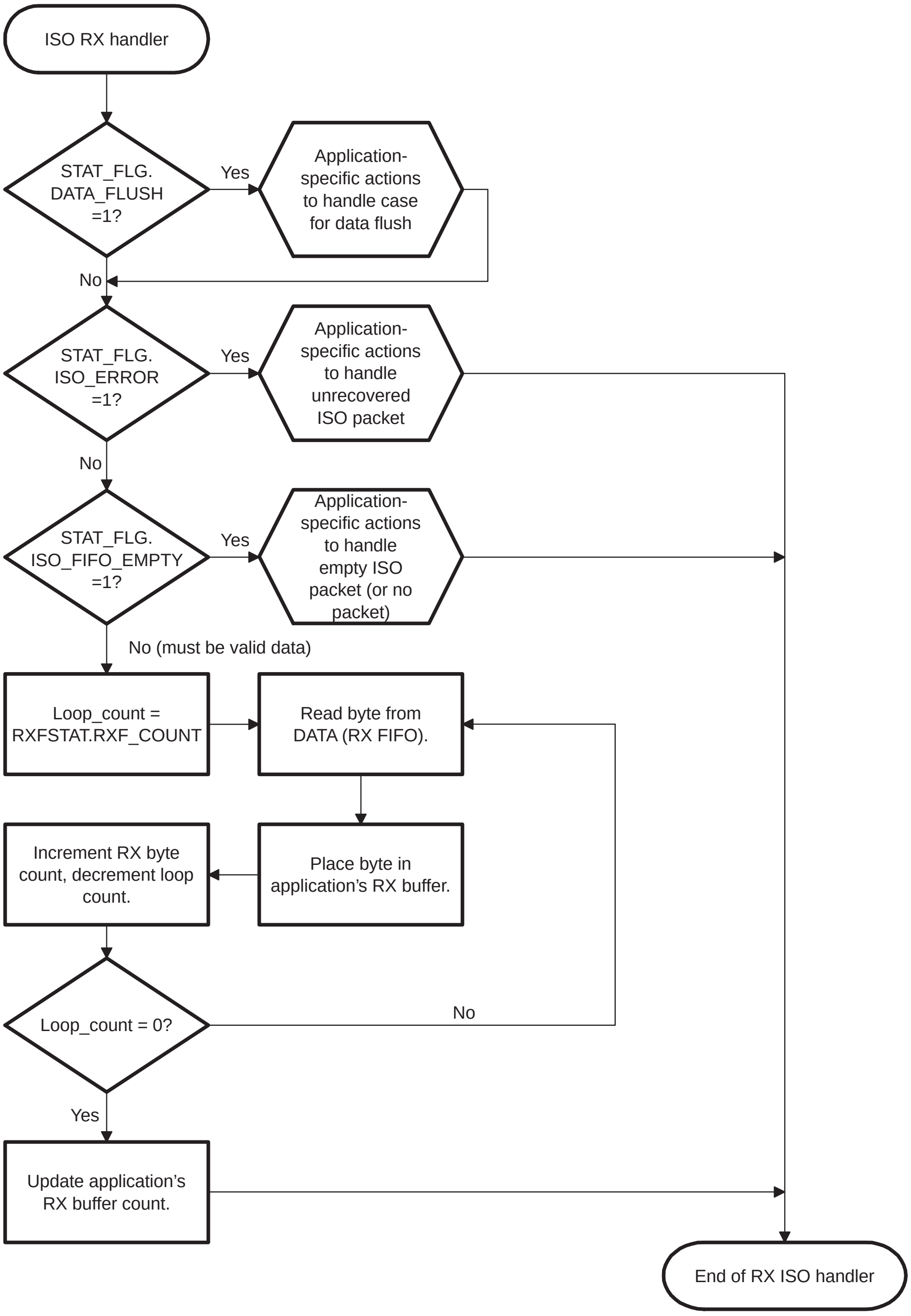
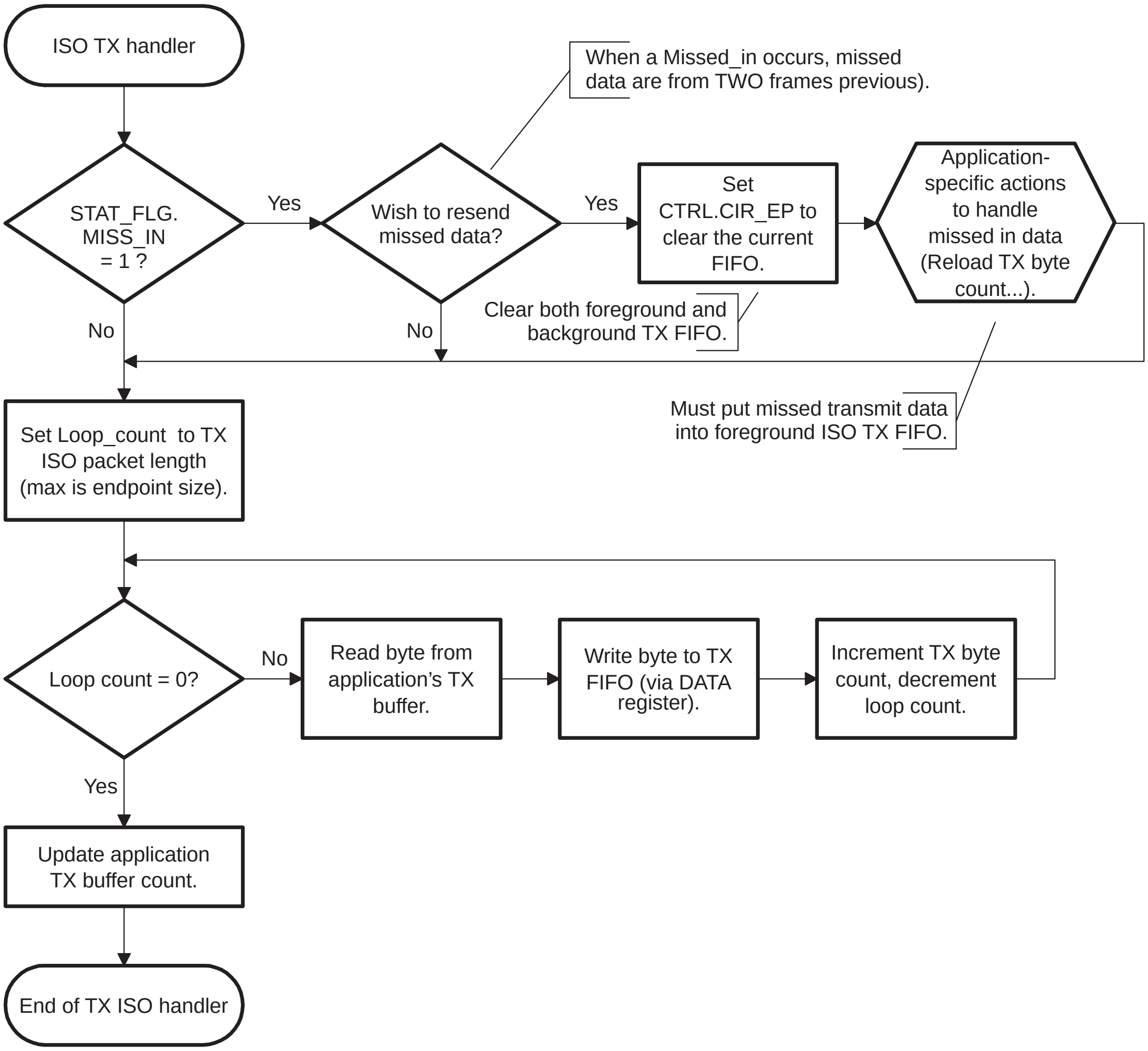

ISO RX handler
STAT_FLG. DATA_FLUSH =1?
Yes
Application-specific actions to handle case for data flush
No
STAT_FLG. ISO_ERROR =1?
Yes
Application-specific actions to handle unrecovered ISO packet
No
STAT_FLG. ISO_FIFO_EMPTY =1?
Yes
Application-specific actions to handle empty ISO packet (or no packet)
No (must be valid data)
Loop_count = RXFSTAT.RXF_COUNT
Read byte from DATA (RX FIFO).
Place byte in application's RX buffer.
Increment RX byte count, decrement loop count.
Loop_count = 0?
No
Yes
Update application's RX buffer count.
End of RX ISO handler

Figure 36. Write Isochronous TX FIFO Data Flowchart


ISO TX handler
When a Missed_in occurs, missed data are from TWO frames previous).
STAT_FLG. MISS_IN = 1 ?
Yes
Wish to resend missed data?
Yes
Set CTRL.CIR_EP to clear the current FIFO.
Application-specific actions to handle missed in data (Reload TX byte count...).
No
No
Clear both foreground and background TX FIFO.
Must put missed transmit data into foreground ISO TX FIFO.
Set Loop_count to TX ISO packet length (max is endpoint size).
Loop count = 0?
No
Read byte from application's TX buffer.
Write byte to TX FIFO (via DATA register).
Increment TX byte count, decrement loop count.
Yes
Update application TX buffer count.
End of TX ISO handler

3.26 Summary of USB Device Controller Interrupts

Table 56 lists the interrupt types by endpoint types.

Table 56. USB Device Controller Interrupt Type by Endpoint Type

Interrupt Type	General USB IRQs (MPU Level 2 IRQ_20)				EP-Specific IRQs (Non-ISO Interrupt on MPU Level 2 IRQ_30)		SOF ISO Interrupt on MPU Level 2 IRQ_29)
	Setup (EP0)	**Control (EP0) Out**	**Control (EP0) In**	**Other**	**Bulk or Interrupt** **Out**	**Bulk or Interrupt** **In**	**(Isochro nous)** **SOF**
Transaction ACKed		X	X		X	X	
Transaction NAKed (if enabled)		X	X		X	X	
Transaction STALLed		X	X		X	X	
Setup	X						
SOF							X
Device state changed				X			
RX DMA EOT (non_ISO)				X			
RX DMA trans count (non_ISO)				X			
TX DMA done (non_ISO)				X			

3.26.1 USB Device Controller Clock Control

The OMAP5912 clock generation and system reset management module(ULPD) provides a single 48-MHz clock to the USB OTG controller, USB device controller, and USB host controller. This clock can be stopped by software to reduce USB OTG controller, USB host, and USB device controller power consumption when USB functionality is not needed. The ULPD controls the 48-MHz clock to the USB device controller via:

- CLOCK_CTRL_REG.USB_MCLK_EN
- SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
- SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

The OTG module allows separate control of USB device controller clocking via OTG_SYSCON_1.DEV_IDLE_EN.

3.26.2 USB Device Controller Hardware Reset

Reset of the USB device controller is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5912 peripherals, including the USB device controller. When held in reset, the USB device controller does not recognize any USB activity.

When the USB device controller is in reset, its registers have no effect on USB functionality. Software must wait until OTG_SYSCON_1.RESET_DONE is 1.

3.27 DMA Operation

The USB device controller provides support for six DMA channels. Three receive DMA channels are reserved to OUT transfers (ISO or non-ISO) and three transmit DMA channels are reserved to IN transfers (ISO or non-ISO). It is not possible to operate DMA transactions on control EP0.

> **CAUTION**
>
> The MPU must not access an endpoint used in a DMA transfer through the EP_NUM, CTRL, and STAT_FLG registers (in DMA, this remark applies after the MPU has set the CTRL.SET_FIFO_EN bit to enable the RX DMA transfer). In particular, the MPU must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

> **Note:**
>
> To use the DMA channels properly, you must set the DMA configuration during the address state interrupt (DS_CHANGE).

The parameters used for DMA transactions (FIFO size, ISO endpoint, double-buffering, and pointers) are those defined for the associated endpoint.

Receive DMA Channels Overview

Receive DMA channels are programmed via the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in RXDMA_CFG.RXDMAn_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when an RX DMA request is active, through the register DATA_DMA. The RX FIFO accessed is that of the endpoint for which the DMA request is active (only one RX DMA request is active at a time).

The USB device controller transmit DMA channels 0 through 2 are connected to OMAP5912 DMA controller requests DMA_REQ_26, DMA_REQ_27, and DMA_REQ_29, respectively.

Non-Isochronous OUT (USB HOST –> MPU) DMA Transactions

During non-ISO transfers to a DMA operated OUT endpoint, a request to the MPU DMA controller is generated when data have been placed into endpoint FIFO and must be read. ACK and NAK interrupts are always disabled automatically by the core for DMA operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control non-ISO OUT transfers.

End of Transfer Interrupt (IRQ_SRC.RXn_EOT)

This interrupt signals that the core has detected an end-of-transfer (EOT). EOT occurs in the two following cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full)
- When the number of received transactions has reached the programmed value in the RXDMAn.RXn_TC field, if the RXDMAn.RXn_STOP bit has been set by the MPU

After an end of transfer interrupt, the MPU must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel.

The MPU must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

Transaction Count Interrupt (IRQ_SRC.RXn_CNT)

The intent of this interrupt is watermark control. It can be used by the MPU to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received exceeds an expected size.

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (and not bytes) has reached the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected non-ISO endpoint. RXn_COUNT interrupt is asserted even if RXDMAn.RXn_STOP has been set; in that case, both RXn_COUNT and RXn_EOT interrupts are asserted (see Figure 37 through Figure 40).

The transaction count watermark is programmed in the RXDMAn register.

Figure 37. Non-ISO RX DMA Transaction Example (RX_TC=2)

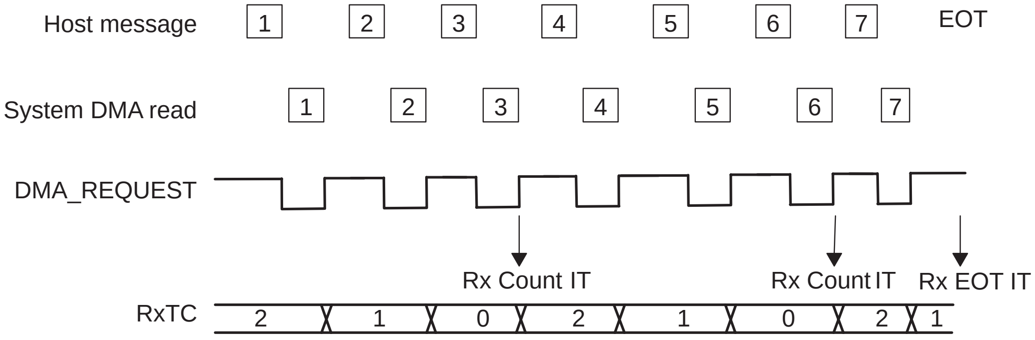


Figure 38. Non-ISO RX DMA Start Routine

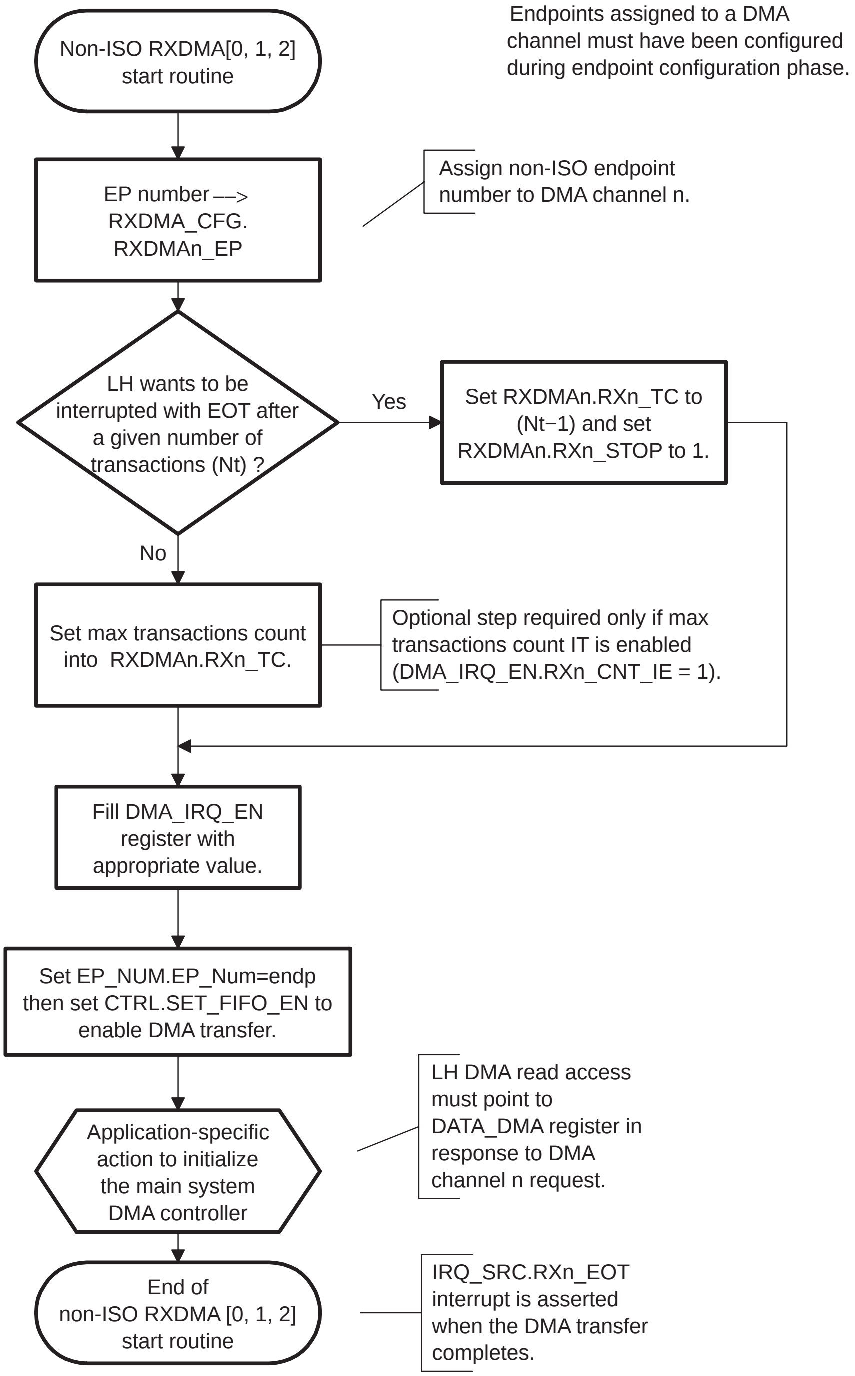

Non-ISO RXDMA[0, 1, 2] start routine
Endpoints assigned to a DMA channel must have been configured during endpoint configuration phase.
EP number —> RXDMA_CFG. RXDMAn_EP
Assign non-ISO endpoint number to DMA channel n.
LH wants to be interrupted with EOT after a given number of transactions (Nt) ?
Yes
Set RXDMAn.RXn_TC to (Nt−1) and set RXDMAn.RXn_STOP to 1.
No
Set max transactions count into RXDMAn.RXn_TC.
Optional step required only if max transactions count IT is enabled (DMA_IRQ_EN.RXn_CNT_IE = 1).
Fill DMA_IRQ_EN register with appropriate value.
Set EP_NUM.EP_Num=endp then set CTRL.SET_FIFO_EN to enable DMA transfer.
Application-specific action to initialize the main system DMA controller
LH DMA read access must point to DATA_DMA register in response to DMA channel n request.
End of non-ISO RXDMA [0, 1, 2] start routine
IRQ_SRC.RXn_EOT interrupt is asserted when the DMA transfer completes.

Figure 39. Non-ISO RX DMA EOT Interrupt Handler

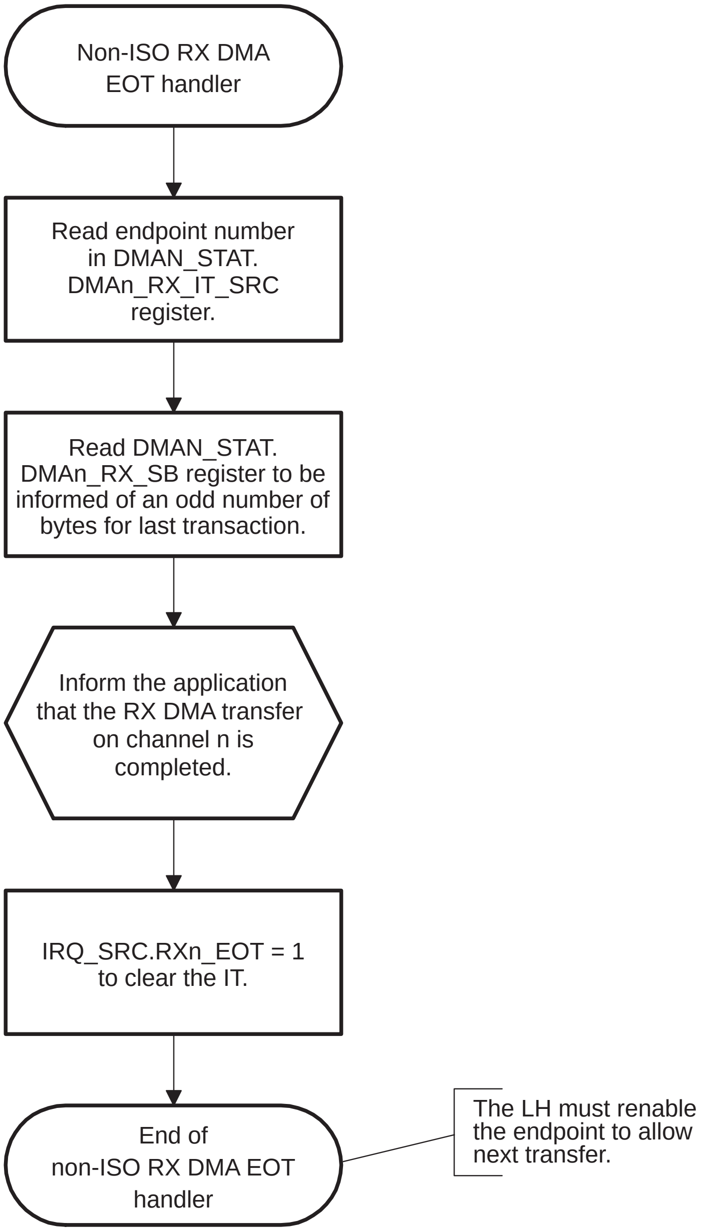
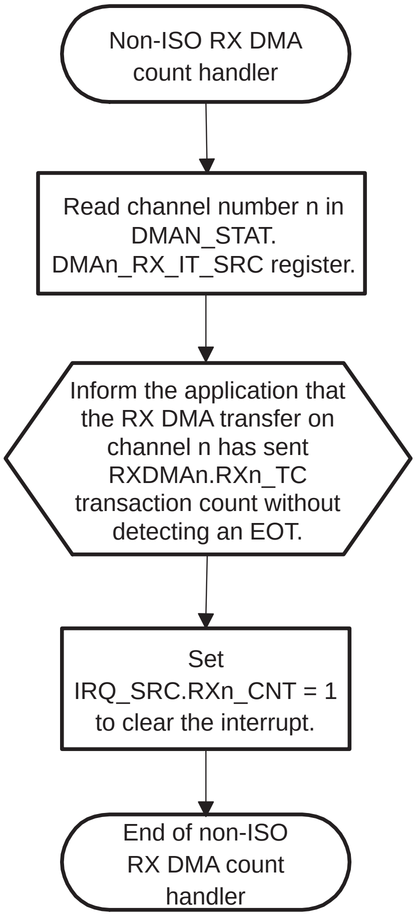

Non-ISO RX DMA EOT handler
Read endpoint number in DMAN_STAT. DMAn_RX_IT_SRC register.
Read DMAN_STAT. DMAn_RX_SB register to be informed of an odd number of bytes for last transaction.
Inform the application that the RX DMA transfer on channel n is completed.
IRQ_SRC.RXn_EOT = 1 to clear the IT.
End of non-ISO RX DMA EOT handler
The LH must renable the endpoint to allow next transfer.

Figure 40. Non-ISO RX DMA Transactions Count Interrupt Handler


Non-ISO RX DMA count handler
Read channel number n in DMAN_STAT. DMAn_RX_IT_SRC register.
Inform the application that the RX DMA transfer on channel n has sent RXDMAn.RXn_TC transaction count without detecting an EOT.
Set IRQ_SRC.RXn_CNT = 1 to clear the interrupt.
End of non-ISO RX DMA count handler

Isochronous OUT (USB HOST –> MPU) DMA Transactions

During ISO transfers to a DMA operated OUT endpoint, a request to the MPU DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no interrupt associated with DMA transfer to ISO OUT endpoints (see Figure 41 and Figure 42).

Figure 41. ISO RX DMA Transaction

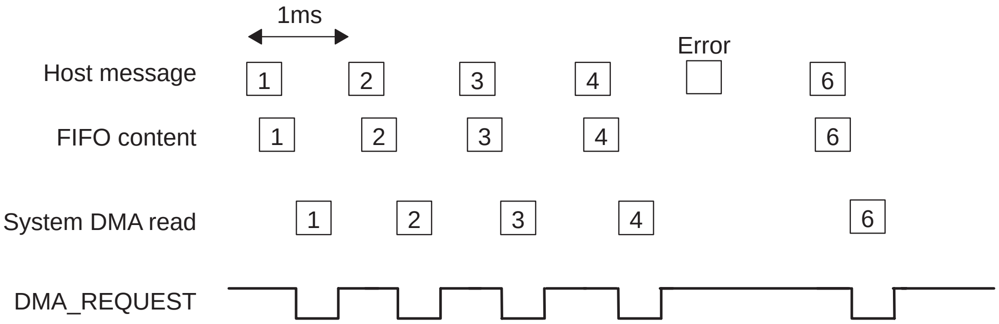
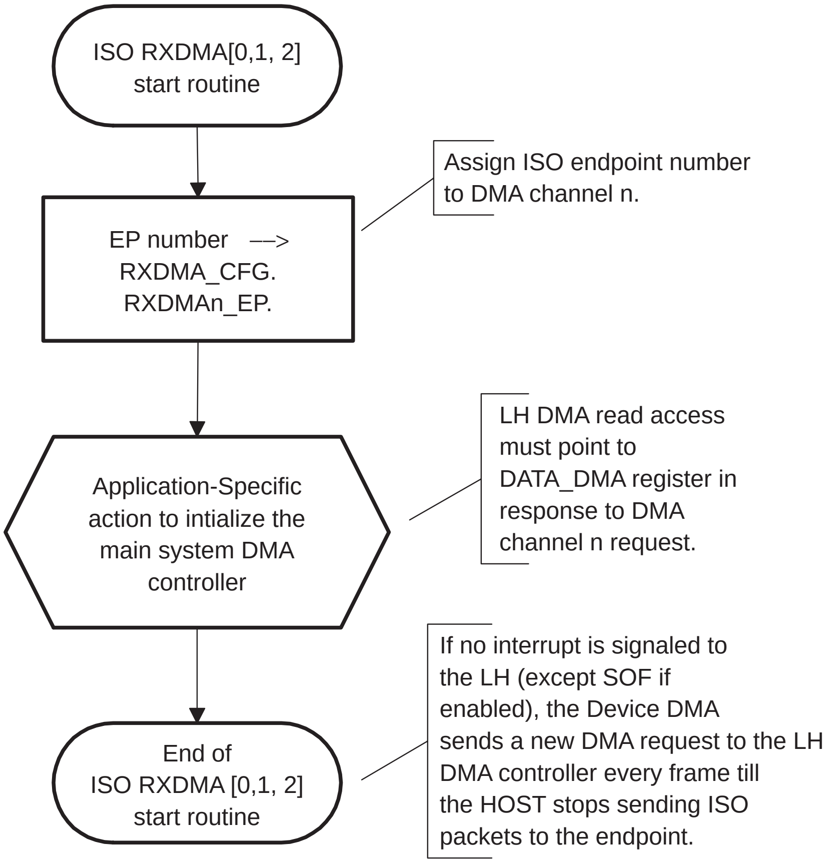


Figure 42. ISO RX DMA Start Routine



Assign ISO endpoint number to DMA channel n.

LH DMA read access must point to DATA_DMA register in response to DMA channel n request.

If no interrupt is signaled to the LH (except SOF if enabled), the Device DMA sends a new DMA request to the LH DMA controller every frame till the HOST stops sending ISO packets to the endpoint.

Transmit DMA Channels Overview

Transmit DMA channels are programmed via the three TXDMA control registers. Each channel can be assigned to a given endpoint number by assigning a non-zero value in TXDMA_CFG.TXDMAn_EP (a 0 value means the DMA channel is deselected). The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate in a different manner for ISO and non-ISO endpoints. Transmitted data must be written into the DATA_DMA when a TX DMA request is active. They are written into the TX FIFO of the endpoint associated with active request (only one TX DMA request active at a given time).

The USB device controller transmit DMA channels 0 through 2 are connected to OMAP5912 DMA controller requests DMA_REQ_29, DMA_REQ_30, and DMA_REQ_31, respectively.

Non-Isochronous IN (MPU –> USB HOST) DMA Transactions

Non-ISO (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in Figure 43 and Figure 44 show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-ISO endpoints in the following manner. The transfer size counter (TXDMAn.TXN_TSC) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request to the MPU main DMA controller is generated when the endpoint buffer is empty initially after that the START bit is set and then each time there is space free in TX FIFO for other TX packet to be written, until TXDMAn.TXn_TSC counts down to zero. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

A DMA transmit transfer done interrupt is signaled to the MPU after the last IN transaction completes successfully. This is after START bit was set and after TXDMAn.TXn_TSC equals 0 for the selected DMA channel.

The MPU must not initiate a new TX DMA transfer until it receives a TX_DONE interrupt.

A small file transfer less than 1024 bytes can be achieved in a single pass DMA signaled by a single interrupt completion. A file size equal to or greater than 1024 bytes needs two or more DMA passes, signaled by an interrupt completion after each pass.

Figure 43 shows the necessary steps to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to the MPU via a DONE interrupt, whose handler is shown in Figure 44. The start routine and the associated interrupt handler are tightly coupled.

Figure 43. Non-ISO TX DMA Start Routine

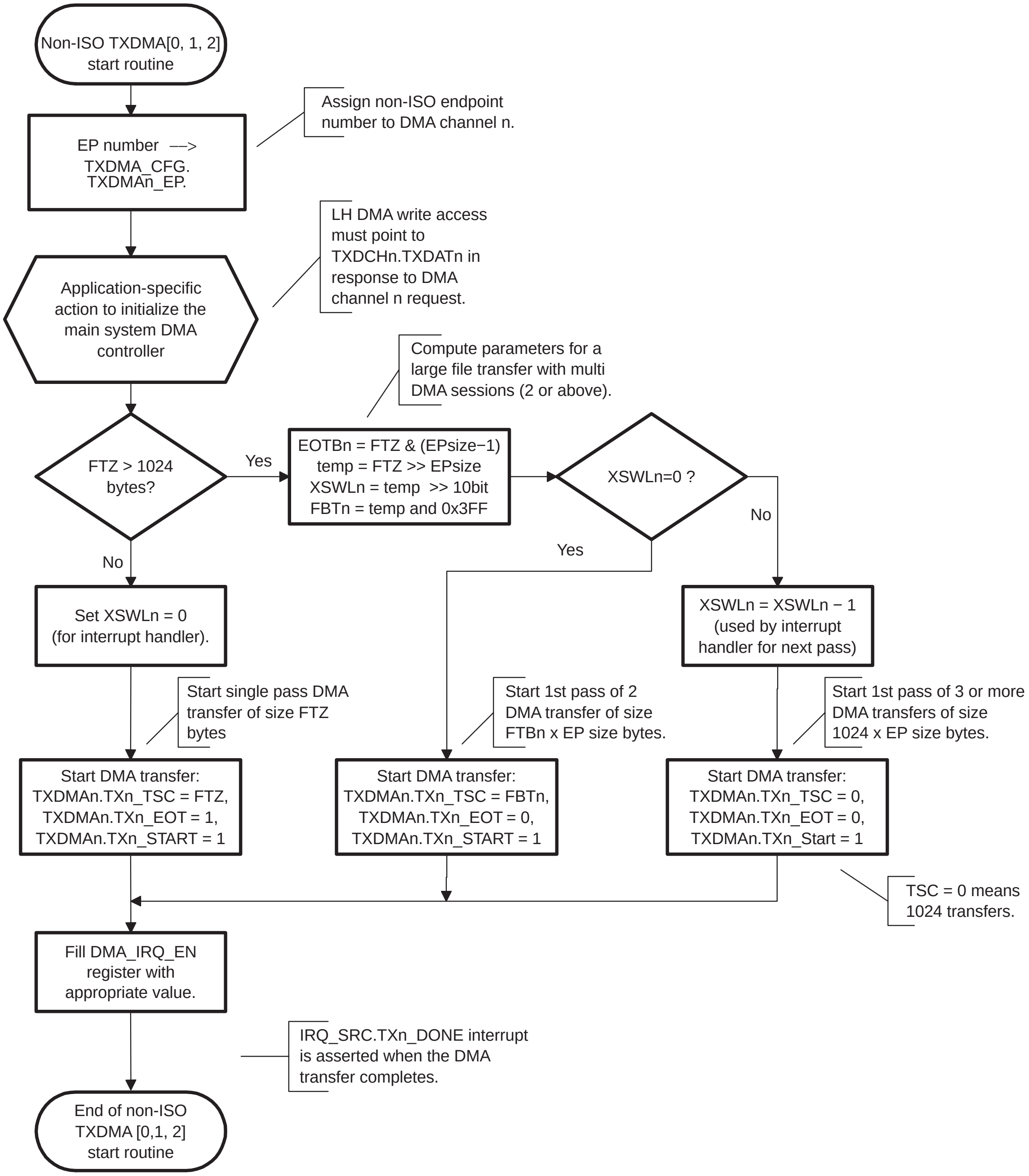
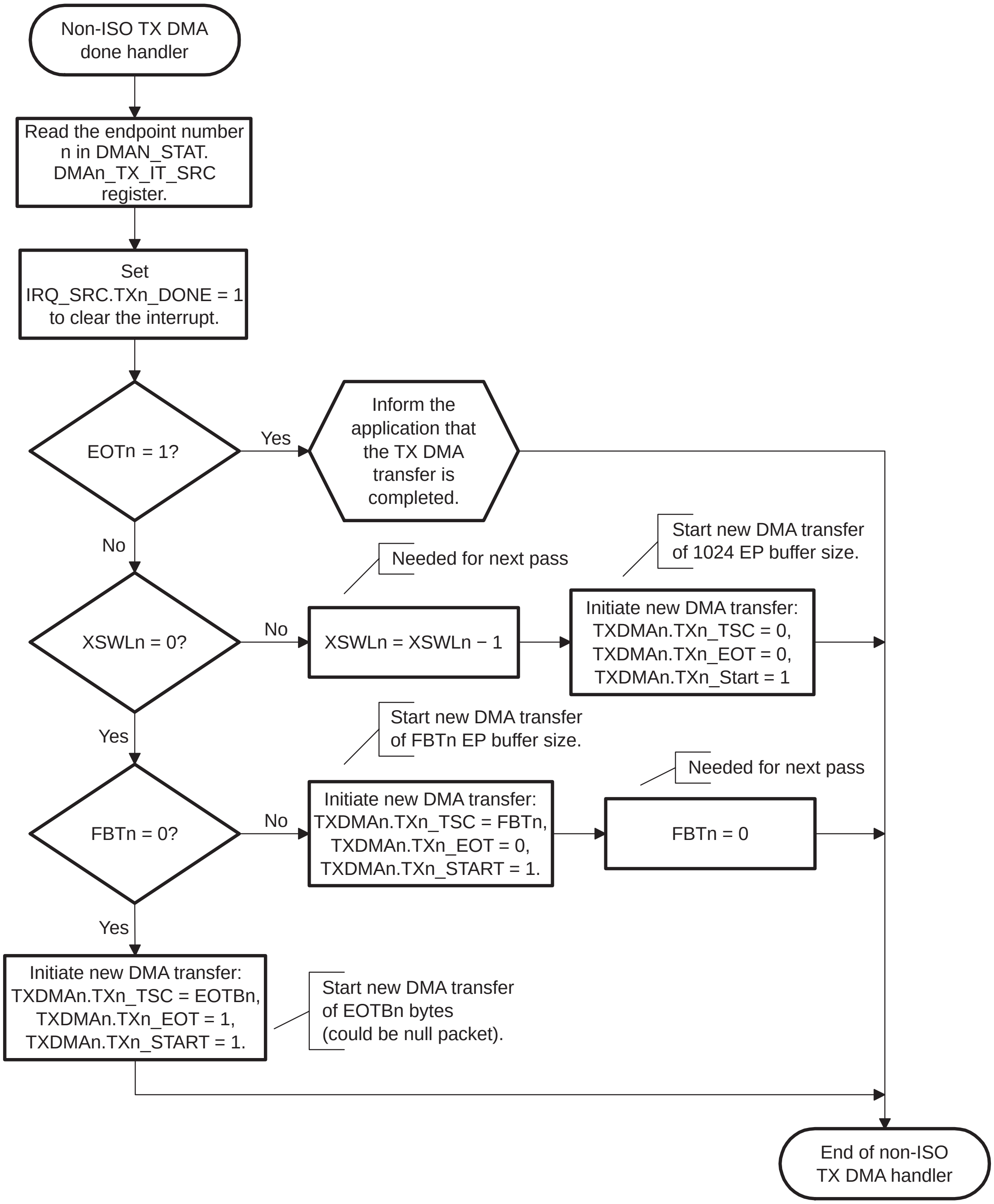

Non-ISO TXDMA[0, 1, 2] start routine
Assign non-ISO endpoint number to DMA channel n.
EP number —> TXDMA_CFG. TXDMAn_EP.
LH DMA write access must point to TXDCHn.TXDATn in response to DMA channel n request.
Application-specific action to initialize the main system DMA controller
Compute parameters for a large file transfer with multi DMA sessions (2 or above).
FTZ > 1024 bytes?
Yes
EOTBn = FTZ & (EPsize−1)
temp = FTZ >> EPsize
XSWLn = temp >> 10bit
FBTn = temp and 0x3FF
XSWLn=0 ?
No
Yes
No
Set XSWLn = 0 (for interrupt handler).
XSWLn = XSWLn − 1 (used by interrupt handler for next pass)
Start single pass DMA transfer of size FTZ bytes
Start 1st pass of 2 DMA transfer of size FTBn x EP size bytes.
Start 1st pass of 3 or more DMA transfers of size 1024 x EP size bytes.
Start DMA transfer: TXDMAn.TXn_TSC = FTZ, TXDMAn.TXn_EOT = 1, TXDMAn.TXn_START = 1
Start DMA transfer: TXDMAn.TXn_TSC = FBTn, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_START = 1
Start DMA transfer: TXDMAn.TXn_TSC = 0, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_Start = 1
TSC = 0 means 1024 transfers.
Fill DMA_IRQ_EN register with appropriate value.
IRQ_SRC.TXn_DONE interrupt is asserted when the DMA transfer completes.
End of non-ISO TXDMA [0,1, 2] start routine

Figure 44. Non-ISO TX DMA Done Interrupt Handler


Non-ISO TX DMA done handler
Read the endpoint number n in DMAN_STAT. DMAn_TX_IT_SRC register.
Set IRQ_SRC.TXn_DONE = 1 to clear the interrupt.
EOTn = 1?
Yes
Inform the application that the TX DMA transfer is completed.
No
Needed for next pass
Start new DMA transfer of 1024 EP buffer size.
XSWLn = 0?
No
XSWLn = XSWLn − 1
Initiate new DMA transfer: TXDMAn.TXn_TSC = 0, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_Start = 1
Yes
Start new DMA transfer of FBTn EP buffer size.
Needed for next pass
FBTn = 0?
No
Initiate new DMA transfer: TXDMAn.TXn_TSC = FBTn, TXDMAn.TXn_EOT = 0, TXDMAn.TXn_START = 1.
FBTn = 0
Yes
Initiate new DMA transfer: TXDMAn.TXn_TSC = EOTBn, TXDMAn.TXn_EOT = 1, TXDMAn.TXn_START = 1.
Start new DMA transfer of EOTBn bytes (could be null packet).
End of non-ISO TX DMA handler

Example 1. Example: 100603 Bytes to Transfer via 32 Bytes IN Bulk Endpoint

This gives XSWL=0x3, FBT=0x47, EOTB=0x1b,

which means five passes of DMA transfer, signaled by 5 TXn_DONE interrupts, are required:

1)	EOT=0, FBT=0, loop=3	32768 bytes transferred (1024 x 32 bytes)
2)	EOT=0, FBT=0, loop=2	32768 bytes
3)	EOT=0, FBT=0, loop=1	32768 bytes
4)	EOT=0, FBT=0x47, loop=0	2272 bytes (71 x 32 bytes)
5)	EOT=1, FBT=0x1B, loop=0	27 bytes

Isochronous IN (USB HOST –> MPU) DMA Transactions

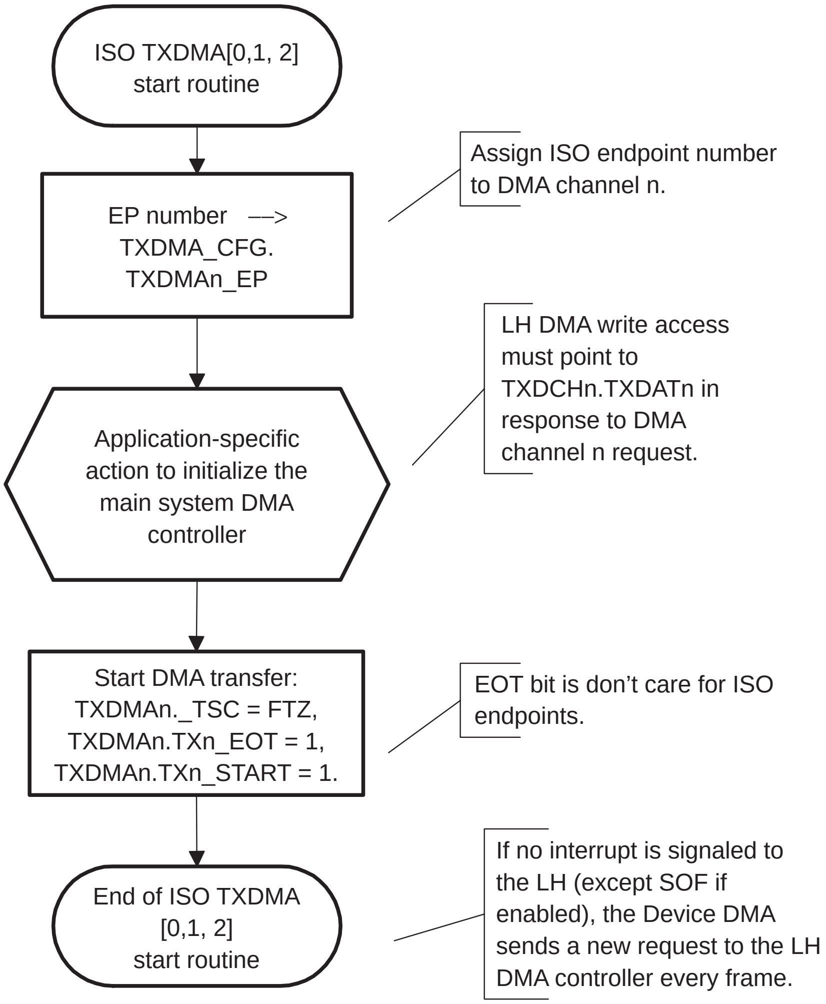
For ISO endpoints, the transfer size counter (TXDMAn.TXn_TSC) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable (see Figure 45).

A request to the MPU main DMA controller is generated when the endpoint buffer is empty initially after the START bit is set, and then after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches the TXDMAn.TXn_TSC value.

During ISO transfers to a DMA operated IN endpoint, a request to the MPU system DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with the DMA transfer.

No interrupt is signaled to the MPU during DMA operation to ISO IN endpoints.

Figure 45. ISO TX DMA Start Routine


ISO TXDMA[0,1, 2]
start routine
EP number ––>
TXDMA_CFG.
TXDMAn_EP
Assign ISO endpoint number to DMA channel n.
Application-specific action to initialize the main system DMA controller
LH DMA write access must point to TXDCHn.TXDATn in response to DMA channel n request.
Start DMA transfer:
TXDMAn._TSC = FTZ,
TXDMAn.TXn_EOT = 1,
TXDMAn.TXn_START = 1.
EOT bit is don't care for ISO endpoints.
End of ISO TXDMA
[0,1, 2]
start routine
If no interrupt is signaled to the LH (except SOF if enabled), the Device DMA sends a new request to the LH DMA controller every frame.

Important Note on DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each start-of-frame if a TX DMA channel is configured for an isochronous endpoint; this
request must be serviced imperatively.

Note on DMA Channels Deconfiguration

It is recommended that the MPU wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing a value 0 in the TX/RXDMA_CFG register. However, if needed by the application, the MPU can unselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The resulting behavior is:

❑ For RX transfer:

  - If RX DMA request is active for the endpoint when the endpoint is unselected, deconfiguration is effective only at the end of the RX DMA request (that is, after all the DMA data have been read). When double-buffering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at unselection). An EOT interrupt is asserted if an end-of-transfer is detected.

  - If the RX DMA request is not active when unselection occurs, the effect is immediate.

❑ For TX transfer:

  - If the request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request has been handled and the TX data have been sent through an IN transaction (both buffers in case of double-buffering with both buffers full). No TX_DONE interrupt is asserted even if TXDMAn.TSC bit value is 0 after the transaction.

  - If the TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in TX buffer(s) have been sent through IN transaction(s). If the TXDMAn_TSC value is 0 at this point, no TX_DONE interrupt is asserted. If TX_DONE interrupt had already been asserted when the endpoint is deselected, configuration is effective only after the TX_DONE interrupt handling.

TX/RXDMA_CFG.TX/RXDMAn_EP reflects the endpoint value until deconfiguration is effective. The MPU must read this register to know if the channel has been disabled yet or not. It must wait until the read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generate an endpoint-specific interrupt (if non-transparent).

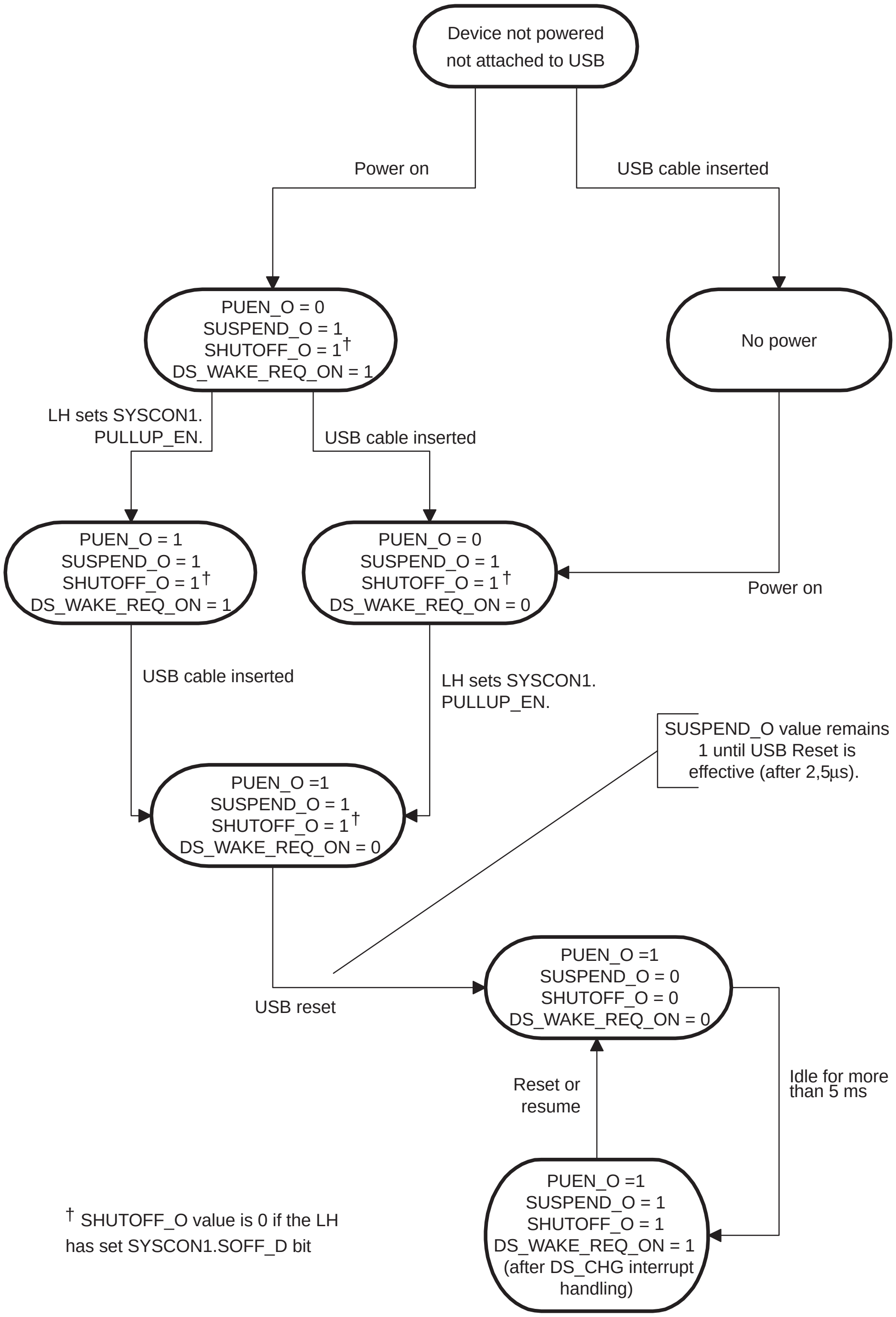
If the selected endpoint is of isochronous type, deconfiguration is effective after the TX/RX request has been serviced, and the subsequent isochronous transactions are handled at SOF interrupt through the endpoint registers (EP_NUM and STAT_FLG).

3.28 Power Management

Figure 46 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

- PUEN_O: Pullup enable signal, always reflecting the SYSCON1.PULLUP_EN register bit
- SHUTOFF_O: Power circuitry shutoff signal, controlled by the core and the SYSCON1.SOFF_DIS bit
- DS_WAKE_REQ_ON: Deep-sleep wake request, asserted low when the interface clock is needed
- SUSPEND_O: Suspend signal, asserted high when the device is in suspend mode

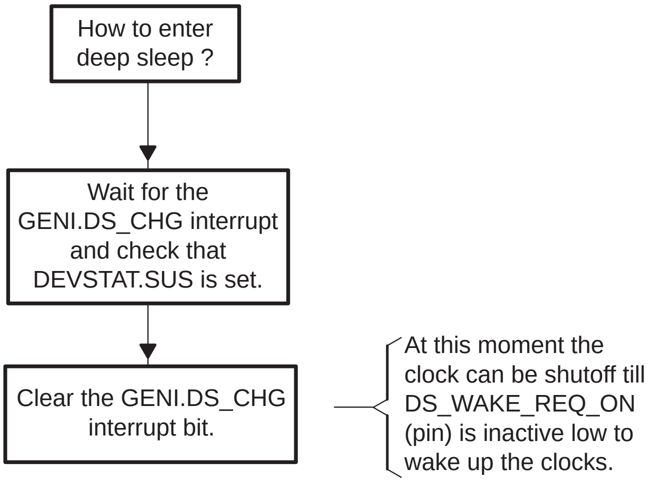
Figure 46. Power Management Signal Values



† SHUTOFF_O value is 0 if the LH has set SYSCON1.SOFF_D bit

From a software point of view, Figure 47 shows the reaction. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

Figure 47. Power Management Flowchart


How to enter deep sleep ?
Wait for the GENI.DS_CHG interrupt and check that DEVSTAT.SUS is set.
Clear the GENI.DS_CHG interrupt bit.
At this moment the clock can be shutoff till DS_WAKE_REQ_ON (pin) is inactive low to wake up the clocks.

4 USB OTG Controller

The OMAP5912 USB OTG controller works in conjunction with the OMAP5912 USB device controller and one port of the OMAP5912 host controller to provide USB On-The-Go functionality. The OMAP5912 OTG controller includes various control and status logic that switches between the two controllers as required by the USB On-The-Go protocol. The combination allows OMAP5912 to act as a USB OTG dual-role device. The OMAP5912 OTG implementation simultaneously allows one USB OTG port and up to two additional USB host ports.

The USB OTG controller provides a dual-role device capability that is compatible with the On-The-Go supplement to the *USB 2.0 Specification* (Revision 1.0). This dual-role device can provide a 12M bit-per-second connection between OMAP5912 and other USB OTG dual-role devices. The OMAP5912 OTG solution does not support 48M bit-per-second OTG connectivity.

In addition, the USB OTG controller provides control and status information for various aspects of the OMAP5912 USB host controller and the OMAP5912 USB device controller operation when On-The-Go functionality is not enabled.

4.1 OTG Controller Features

The main features of the OMAP5912 USB OTG controller include:

- ❑ USB specification version 2.0 compatibility
- ❑ When acting as an OTG a-peripheral or an OTG b-peripheral: a 12M bit-per-second communication link with configurable data transfer type, data buffer size, single- or double-buffering for each endpoint, and up to three IN endpoints using DMA and up to three OUT endpoints using DMA to support streaming USB data.
- ❑ When acting as an OTG a-host or an OTG b-host: an OHCI Rev 1.0-compatible USB host controller capable of USB full-speed (12M bits-per-second) and USB low-speed (1.5M bits-per-second) communication.

These are basic features of the OMAP5912 USB device controller and the OMAP5912 USB host controller. Additional OTG-specific features provided by the OMAP5912 OTG controller include:

- ❑ Control and status logic that allows implementation of an OTG dual-role device (DRD)
- ❑ Ability to implement an OTG peripheral-only device (that is, one that cannot act as an OTG A-device)
- ❑ Support for generation and detection of the OTG host negation protocol (HNP) and both types of OTG session request protocol
- ❑ Support for OTG transceivers compatible with the OTG working group OTG transceiver specification

4.2 OTG Controller Registers

Table 57 lists the 32-bit OTG controller registers. Table 58 through Table 72 describe the register bits.

Table 57. OTG Controller Registers

Name	Description	R/W	Address
OTG_REV	OTG controller revision number	R/O	FFFB:0400h
OTG_SYSCON_1	OTG system configuration group 1	R/W	FFFB:0404h
OTG_SYSCON_2	OTG system configuration group 2	R/W	FFFB:0408h
OTG_CTRL	OTG control	R/W	FFFB:040Ch

Table 57. OTG Controller Registers (Continued)

Name	Description	R/W	Address
OTG_IRQ_EN	OTG interrupt enable	R/W	FFFB:0410h
OTG_IRQ_SRC	OTG interrupt source identification	R/W	FFFB:0414h
OTG_OUTCTRL	OTG interrupt source identification	R/W	FFFB:0418h
OTG_TEST	OTG interrupt source identification	R/W	FFFB:0420h
Reserved	Reserved	None	FFFB:0424h to FFFB:04FBh
VC	USB vendor code	R/O	FFFB:04FCh

All bits defined as reserved must be written by software with 0s, for preserving future compatibility. When read, any reserved bit returns 0. It is good software practice to use complete mask patterns for setting or testing bit fields individually within a register.

Table 58. OTG Revision Number Register (OTG_REV)

Bit	Name	Description
31:8		
7:0	OTG_REV_NB	OTG revision number: this 8-bit field indicates the revision number of the current OTG controller, in binary-coded digital (BCD), with the major revision number in bits 7−4, and the minor revision number in bits 3−0. This value is hardware fixed. 0x10: Revision 1.0 0x11: Revision 1.1 0x12: Revision 1.2 0x13: Revision 1.3 0x14: Revision 1.4 0x15: Revision 1.5 0x16: Revision 1.6 .... 0xF2: Revision 15.2

This read-only register reflects the OTG controller revision number.

Table 59. OTG System Configuration Register 1 (OTG_SYSCON_1)

Bit	Name	Description
31:27	Reserved	Reserved
26:24	USB2_TRX_MODE†	USB port 2 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 2. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 4.5, *Transceiver Signaling Types*).
23	Reserved	Reserved
22:20	USB1_TRX_MODE‡	USB port 1 transceiver mode.These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 1. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported trans)ceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 4.5, *Transceiver Signaling Types*).
19	Reserved	Reserved
18:16	USB0_TRX_MODE§	USB port 0 transceiver mode. These bits configure the transceiver signaling type used by the USB host, USB device, and USB OTG controllers when communicating with the transceiver connected to USB pin group 0 or USB alternate pin group 2. Transceiver signaling type depends on the signaling mode used by the transceiver. The supported transceiver signaling types are 3-pin DAT/SE0 mode bidirectional transceiver signaling, 4-pin VP/VM mode bidirectional transceiver signaling, and 6-pin DAT/SE0 unidirectional transceiver signaling (see Section 4.5, *Transceiver Signaling Types*).

† See Table 60, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 61, *Pin Group 1 Transceiver Type Selection*.
§ See Table 62, *Pin Group 0 Transceiver Type Selection* and Table 63, *Alternate Pin Group 2 Transceiver Type Selection*

Table 59. OTG System Configuration Register 1 (OTG_SYSCON_1) (Continued)

Bit	Name	Description
15	OTG_IDLE_EN	OTG controller clock-gating control. This bit disables the clock to the OTG controller. 0: The OTG controller clock is not gated. Register settings in the ULPD can prevent the USB OTG controller from getting a clock even if this bit is 0. 1: The OTG controller is idled by disabling the OTG controller. Accesses to OTG controller registers are allowed, but the OTG controller clock is not gated. This register is cleared 0 by soft reset or hardware reset. For more information about the power saving circuitry, see Section 4.2.2.
14	HST_IDLE_EN	0 − Host power management circuitry disabled 1 − Host power management circuitry enabled
13	DEV_IDLE_EN	Device controller clock-gating control. This bit defines the device power-saving circuitry. 0: The USB device controller clock is not gated. Other registers in OMAP5912 can prevent the USB device controller from getting a clock even if this bit is 0. 1: The USB device controller clock is disabled if the USB device controller does not need an active clock (such as when the USB device controller is in susped). Accesses to USB device controller registers are allowed, but the USB device controller state machines does not function. If the USB device controller sees USB resume signaled when DEV_IDLE_EN is 1, a USB device controller general interrupt is generated. The interrupt service routine must clear DEV_IDLE_EN before performing the operations necessary to service the interrupt. This register is set to1 by soft reset or hardware reset. For more information about the USB device controller, see Section 3, *USB Device Controller*.
12:3	Reserved	Reserved

† See Table 60, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 61, *Pin Group 1 Transceiver Type Selection*.
§ See Table 62, *Pin Group 0 Transceiver Type Selection* and Table 63, *Alternate Pin Group 2 Transceiver Type Selection*

Table 59. OTG System Configuration Register 1 (OTG_SYSCON_1) (Continued)

Bit	Name	Description
2	RESET_DONE	Reset status information: This bit reflects the reset status of the controller (hardware and software reset, internal reset monitoring of the USB device and the USB OTG controller sections). 0: Internal reset for OTG is ongoing (includes OTG controller and USB device) 1: Reset completed This register is cleared 0 by soft reset or hardware reset. This reset done information is important for proper use of the controller. After a hardware reset or a soft reset, this bit must be checked to determine that the reset has completed.This register does not reflect the reset status of the OMAP5912 USB host controller. Software must query certain USB host controller registers to determine when the USB host controller reset is complete. See Section 2.16,*USB Host Controller Hardware Reset*.
1	SOFT_RESET	Software reset for the OTG, device, and host controllers: set this bit to 1 to trigger a reset of the OTG controller, USB host controller, and USB device controller. The bit is automatically cleared by the hardware. During reads, it always returns 0. 0: Normal mode 1: The controller is reset. Software must monitor the RESET_DONE to determine when the OTG controller and the USB device controller have completed reset. Section 2.16 *USB Host Controller Hardware Reset* describes software activities required to determine when the USB host controller completes its reset.
0	Reserved	–

† See Table 60, *Pin Group 2 Transceiver Type Selection*.
‡ See Table 61, *Pin Group 1 Transceiver Type Selection*.
§ See Table 62, *Pin Group 0 Transceiver Type Selection* and Table 63, *Alternate Pin Group 2 Transceiver Type Selection*

This read-write register controls the USB transceiver interface, the USB controller clock gating, and USB controller reset generation.

Table 60. Pin Group 2 Transceiver Type Selection

USB2_TRX_MODE	OMAP5912 Pin Group 2 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling Normal functionality is available if USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 1. If USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 0, OMAP5912 places pins USB2.TXD and USB2_TXSE0 in high-impedance mode.

0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

This register is cleared 0x0 by soft reset or hardware reset.

When both

- USB2_TRX_MODE = 0
- USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 0

OMAP5912 places pins USB2.TXD and USB2.TXSE0 in high-impedance mode to prevent contention with any signals driven by the external transceiver. Mode 0 can be used for normal 6-pin DAT/SE0 mode functionality when USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R = 1.

This register does not have an effect on the transceiver signaling type when top-level pin multiplexing selects alternate pin group 2 (see the USB0_TRX_MODE subsection). When HMC_MODE selects a transceiverless link logic connection using USB pin group 2, this register must be set to 3.

Table 61. Pin Group 1 Transceiver Type Selection

USB1_TRX_MODE	OMAP5912 Pin Group 1 Transceiver Signaling
0x0	6-pin DAT/SE0 mode unidirectional signaling. Normal functionality is available if USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 1. If USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 0, OMAP5912 places pins USB1.TXD and USB1_TXSE0 in high-impedance mode.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

This register is cleared to 0x0 by soft reset or hardware reset.

When both:

- USB1_TRX_MODE = 0
- USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 0

OMAP5912 places pins USB1.TXD and USB1_TXSE0 in high-impedance mode to prevent contention with any signals driven by the external transceiver. Mode 0 can be used for normal 6-pin DAT/SE0 mode functionality when USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R = 1.

Table 62. Pin Group 0 Transceiver Type Selection

USB0_TRX_MODE	**OMAP5912 Pin Group 0 Transceiver Signaling**
0x0	6-pin DAT/SE0 mode unidirectional signaling.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

This register is cleared to 0x0 by soft reset or hardware reset.

When using the OMAP5912 integrated USB transceiver, USB0_TRX_MODE is set to 3. Software can avoid USB0_TRX_MODE = 0 to improve software compatibility with future devices.

When OMAP5912 top-level signal multiplexing selects alternate pin group 2, the USB0_TRX_MODE signal affects the transceiver type for the transceiver connected to the following OMAP5912 pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART1.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM

Table 63. Alternate Pin Group 2 Transceiver Type Selection

USB0_TRX_MODE	**OMAP5912 Alternate Pin Group 2 Transceiver Signaling**
0x0	6-pin DAT/SE0 mode unidirectional signaling.
0x1	4-pin VP/VM mode bidirectional signaling
0x2	3-pin DAT/SE0 mode bidirectional signaling
0x3	6-pin DAT/SE0 mode unidirectional signaling
Other values	Reserved

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2)

Bit	Name	Description
31	OTG_EN	The OTG enable bit selects the OTG controller functionality: 0: The OTG controller circuitry is not activated. The USB device controller and the USB host controller can be used, but an OTG link cannot be implemented. 1: The OTG controller circuitry is activated. The OTG state machine (HNP) can be used on one of the USB ports. The USB device controller and one of the USB host controller ports can be used to implement an OTG link. This register is cleared 0 by soft reset or hardware reset. When an OTG link is not needed, this bit can be cleared and OTG_SYSCON_1.OTG_IDLE_EN can be set to reduce power consumption. A driver switch interrupt is internally generated whenever a 1 is written to OTG_EN. If OTG_IRQ_EN.DRIVER_SWITCH_EN is 1, this internal interrupt propagates to the MPU level 2 interrupt controller.
30	USBx_SYNCHRO	The USB output signals synchronized bit must be set to 1 to allow proper signal timing on OMAP5912 USB pins. This register is cleared 0 by soft reset or hardware reset.
29	OTG_MST16	This bit enables compatibility with the 16-bit OTG master controller, and it must be set to 0 to allow proper USB host controller access to OMAP5912 system memory. 0: Host controller is connected to system memory via a 32-bit bus. 1: Host controller is connected to system memory via a 16-bit bus. OMAP5912 does not support this setting. This register is cleared to 0 by soft reset or hardware reset.
28	SRP_GPDATA	Session request protocol generation pulse width on D+. This bit allows the OTG controller to extend the duration of the D+ pulse during the data line pulsing portion of a session request protocol activity. This bit is only used when the OTG controller is configured to act as an OTG default B device. 0: Data line pulse during SRP is 6 ms (nominal). 1: Data line pulse during SRP is 9 ms (nominal). This register is cleared 0 by soft reset or hardware reset. When the OTG controller performs the SRP process, it first generates a data line pulse SRP request and then a VBUS pulse SRP request.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
27	SRP_GPDVBUS	Session request protocol generation discharge VBUS enable. This bit enables the OTG controller to provide a VBUS discharge sequence as part of its SRP generation activities. This bit enables the VBUS discharge functionality after any VBUS charge process. When enabled, the duration of the VBUS discharge is defined by SRP_GPUVBUS. 0: VBUS is not discharged after the SRP VBUS pulse. 1: VBUS is discharged after the SRP VBUS pulse. This register is cleared 0 by soft reset or hardware reset.

[†] See Table 65, *OTG_PADEN Source Status*.
[‡] See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
26:24	SRP_GPUVBUS	Session request protocol generation charge VBUS duration.This bit controls the duration that the OTG controller attempts to charge VBUS when acting as a default-B OTG device and requesting SRP. 0x0: OMAP5912 does not issue a VBUS charge pulse as part of its SRP generation sequence. 0x1: VBUS is charged for 0.5 ms nominal. 0x2: During SRP generation, VBUS is charged for .5 ms nominal. 0x3: During SRP generation, VBUS is charged for 2 ms nominal. 0x4: During SRP generation, VBUS is charged for 4 ms nominal. 0x5: During SRP generation, VBUS is charged for 6 ms nominal. 0x6: During SRP generation, VBUS is charged for 10 ms nominal. 0x7: During SRP generation, VBUS is charged for 40 ms nominal. This register is cleared 000 by soft reset or hardware reset. The current sourcing capabilities of the hardware that drives the SRP VBUS pulse determines the required duration of the VBUS pulse. The value chosen must ensure that VBUS rises higher than 2 V within the selected VBUS pulse duration when attached to an OTG dual-role device. The value chosen must also ensure that VBUS does not rise higher than 2 V within the selected VBUS pulse duration when attached to a non-OTG port, such as a USB host port or a downstream port of a USB hub. SRP_GPUVBUS also controls the duration of a VBUS discharge pulse when SRP_GPDBVBUS is set to 1. If SRP_GPUVBUS is 0x0, the OTG controller does not request a VBUS discharge. The minimum time for SRP signalling completion is (the duration of the data line pulse defined by SRP_GPDATA) + (2 * VBUS pulse time defined by SRP_GPUVBUS). Disabling the VBUS discharge does not affect that minimum time. The interrupt OTG_IRQ_SRC:B_SRP_DONE is generated upon completion of the entire SRP.
23	Reserved	Reserved

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
22:20	A_WAIT_VRISE	Offset for A_WAIT_VRISE timer.These bits define the maximum duration that the OTG controller must wait for VBUS to rise above the A-device VBUS valid threshold. A_WAIT_VRISE is used only when OMAP5912 acts as a default-A device. The A_WAIT_VRISE counter begins counting upon transition from the A_IDLE state. (See the *On-The-Go Supplement to the USB 2.0 Specification Revision 1.0* description of the dual-role A-device state diagram). 0x0: Maximum delay in A_WAIT_VRISE state is 200 ms nominal. 0x1: Maximum delay in A_WAIT_VRISE state is 287.04 ms nominal. 0x2: Maximum delay in A_WAIT_VRISE state is 374.08 ms nominal. 0x3: Maximum delay in A_WAIT_VRISE state is 548.16 ms nominal. This register is cleared 0x0 by soft reset or hardware reset. If the A_WAIT_VRISE counter expires before VBUS rises above the A-device VBUS valid threshold, it indicates that there is a heavy load on VBUS. The OTG controller state machine transitions to A_VBUS_ERR and generates a VBUS error interrupt.
19	Reserved	Reserved
18:16	B_ASE0_BRST	Offset for B_ASE0_BRST timer. These bits control how the USB OTG controller manages disabling the D+ pullup when acting as a default-B OTG dual-role device and transitioning from state b_peripheral to b_wait_acon, and when acting as a default-A OTG dual-role device and transitioning from state a_peripheral to a_wait_bcon. The setting of this register controls the mechanism by which the OMAP5912 OTG controller knows that the D+ pullup is disabled. Because OMAP5912 implementations control the D+ pullup using an I^{2}C link to an OTG transceiver, the only allowed setting is 0x4. When in this mode of operation, system software must write a 1 to OTG_CTRL.OTG_PU immediately after completion of the I^{2}C operation that disables the D+ pullup. This field is cleared 0x0 by soft reset or hardware reset. This field has no effect when OTG is disabled. This field must not be changed when OTG functionality is enabled (OTG_SYSCON_2.OTG_EN=1).
15	Reserved	Reserved

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
14	SRP_DPW	Session request protocol detection—pulse width. This bit selects the width for the SRP detection when acting as a default-A OTG dual-role device. This value applies on both DATA and VBUS pulsing detections. 0: SRP pulse must be greater than 1 ms (nominal) to be sensed as a valid D+, D−, or VBUS SRP pulse. 1: SRP pulse must be greater than 167 ns (nominal) to be sensed as a valid D+, D−, or VBUS SRP pulse. This register is cleared 0 by soft reset or hardware reset. This bit has no effect when the OTG feature is disabled (OTG_SYSCON_2.OTG_EN = 0).
13	SRP_DATA	Session request protocol detection—data pulsing detection enable. This bit determines whether the OMAP5912 OTG controller considers D+ or D− pulses as SRP requests. 0: OTG controller does not consider D+ or D− pulses as SRP requests. 1: OTG controller treats a D+ or D− pulse of appropriate minimum duration as an SRP request. This register is cleared 0 by soft reset or hardware reset. This register is used only when the OMAP5912 OTG controller acts as a default-A dual-role device. SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set.
12	SRP_VBUS	Session request protocol detection—VBUS pulsing detection enable.This bit determines whether the OMAP5912 OTG controller considers VBUS pulses as SRP requests. 0: OTG controller does not consider VBUS pulses as SRP requests. 1: OTG controller treats a VBUS pulse of appropriate minimum duration as an SRP request. This register is cleared 0 by soft reset or hardware reset. This register is only used when the OMAP5912 OTG controller acts as a default-A dual-role-device. SRP detection requires that either OTG_SYSCON_2.SRP_DATA or OTG_SYSCON_2.SRP_VBUS (or both) is set.
11	Reserved	Reserved

[†] See Table 65, *OTG_PADEN Source Status*.
[‡] See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
10	OTG_PADEN†	OTG transceiver control and status information selector. This bit determines how OTG_CTRL register bits ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSEND are controlled. When using OMAP5912 OTG controller functionality to implement an On-The-Go dual-role device, this bit must be set to 0. In this case, those OTG_CTRL bits are read/write bits and are controlled by system software. When using OMAP5912 USB device controller functionality without enabling OTG functionality, this bit must be set to 1. In this case, OTG_CTRL register bits ID, VBUSVLD, BSESSEND, and ASESSEND are held at 0, and OTG_CTRL bit BSESSVLD is controlled by the OMAP5912 GPIO0/USB.VBUS pin (if top-level pin multiplexing selects the USB_VBUS mode for pin GPIO0). Software writes to those OTG_CTRL bits are ignored when OTG_PADEN = 1.
9	HMC_PADEN‡	USB pin multiplexing control selector. This bit determines whether USB signal multiplexing is controlled by registers in OTG_SYSCON_2 or by registers in the OMAP5912 configuration register module. 0 = OTG_SYSCON_2 provides the controls. 1 = OMAP5912 configuration register module registers provide the controls.
8	UHOST_EN	The host USB controller enable bit controls the clock enable and hardware reset for the OMAP5912 USB host controller when HMC_PADEN is 0. When HMC_PADEN is 1, writes to this bit have no effect on the clock enable. 0: USB host controller is not clocked and is held in hardware reset. 1: USB host controller clock is not held inactive and a USB host controller hardware reset is not forced. The USB host controller is clocked if the ULPD 48-MHz clock output to the USB controllers is active and if the OMAP5912 peripheral reset is inactive. This register is cleared 1 by soft reset or hardware reset. A read of this register gives the internal value used for UHOST_EN regardless of the setting of HMC_PADEN. See Section 15.1.20 USB host controller hardware reset for information on USB host controller access restrictions relating to UHOST_EN.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

Table 64. OTG System Configuration Register 2 (OTG_SYSCON_2) (Continued)

Bit	Name	Description
7	HMC_TLLSPEED	The HMC TLL SPEED configuration bit controls the way that the OMAP5912 transceiverless link logic models the speed of the transceiverless link when HMC_PADEN is 0. When HMC_PADEN is 1, this bit has no effect on the transceiverless link logic and writes to this bit have no effect. 0: Transceiverless link logic models a low-speed (1.5M bits-per-second) USB link. 1: Transceiverless link logic models a full-speed (12M bits-per-second) USB link. This register is cleared 0 by soft reset or hardware reset. Proper operation of the transceiverless link logic requires that the USB host controller is clocked when the transceiverless link logic is used, even if the transceiverless link logic is not connected to a USB host controller port.
6	HMC_TLLATTACH	The HMC TLL ATTACH configuration bit controls the OMAP5912 transceiverless link logic attach/detach status when HMC_PADEN is 0. When HMC_PADEN is 1, this bit has no effect on the transceiverless link logic and writes to this bit have no effect. 0: Transceiverless link logic models a detached link. 1: Transceiverless link logic models an attached link. This register is cleared 0 by soft reset or hardware reset. Proper operation of the transceiverless link logic requires that the USB host controller is clocked when the transceiverless link logic is used, even if the transceiverless link logic is not connected to a USB host controller port.
5:0	HMC_MODE	The HMC mode configuration bits control the OMAP USB signal multiplexing selection when HMC_PADEN is 0. When HMC_PADEN is 1 this field is unused and writes to this register have no effect. HMC_MODE values are discussed in Section 4.3, *Pin Multiplexing*. This register is cleared 0x0 by soft reset or hardware reset.

† See Table 65, *OTG_PADEN Source Status*.
‡ See Table 66, *HMC_PADEN: USB Signal Multiplexing Control Source*.

This read-write register provides control and status for various aspects of OTG controller, USB pin multiplexing, and USB host controller functionality.

Table 65. OTG_PADEN Source Status

OTG_PADEN	Value	Source
0	ID	OTG_CTRL.ID
	VBUSVLD	OTG_CTRL.VBUSVLD
	BSESSEND	OTG_CTRL.BSESSEND
	ASESSEND	OTG_CTRL.ASESSEND
	BSESSVLD	OTG_CTRL.BSESSVLD
	VBUS value to USB device controller	
1	ID	Tied to 0
	VBUSVLD	Tied to 0
	BSESSEND	Tied to 0
	ASESSEND	Tied to 0
	BSESSVLD	Follows GPIO0/USB.VBUS if top-level pin multiplexing selects the USB.VBUS mode for pin GPIO_0. Is tied to 0 if top-level pin multiplexing selects a mode other than USB.VBUS for pin GPIO0.
	VBUS value to USB device controller	BSESSVLD

This register is cleared 0 by soft reset or hardware reset.

Table 66. HMC_PADEN: USB Signal Multiplexing Control Source

HMC_PADEN	Value	Source
0	HMC_MODE	OTG_SYSCON_2.HMC_MODE
	HMC_TLLATTACH	OTG_SYSCON_2.HMC_TLLATTACH
	HMC_TLLSPEED	OTG_SYSCON_2.HMC_TLLSPEED
	UHOST_EN	OTG_SYSCON_2.UHOST_EN

Table 66. HMC_PADEN: USB Signal Multiplexing Control Source (Continued)

HMC_PADEN	Value	Source
1	HMC_MODE	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_MODE_R
	HMC_TLLATTACH	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_TLL_ATTACH_R
	HMC_TLLSPEED	MOD_CONF_CTRL_0. CONF_MOD_USB_HOST_HHC_HMC_TLL_SPEED_R
	UHOST_EN	MOD_CONF_CTRL_0.CONF_MOD_USB_HOST_HHC_UHOST_EN_R

This register is cleared 0 by soft reset or hardware reset.

For best software compatibility with future devices, it is recommended that system software set this register to 0.

Table 67. OTG Control Register (OTG_CTRL)

Bit	Name	Description
31:21	Reserved	Reserved
20	ASESSVLD	Current A-device session valid value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{A\_SESS\_VLD}$. When VBUS is above $V_{A\_SESS\_VLD}$, ASESSVLD must be set to 1. When VBUS is below $V_{A\_SESS\_VLD}$, ASESSVLD should be set to 0. This information is only relevant when connected as an A-device (OTG_CTRL.ID = 0) and OTG is enabled (OTG_EN = 1). 0: VBUS voltage is below $V_{A\_SESS\_VLD}$. 1: VBUS voltage is above $V_{A\_SESS\_VLD}$. This register is cleared 0x0 by soft reset or hardware reset.
19	BSESSEND	Current B-device session end value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{B\_SESS\_END}$. When VBUS is below $V_{B\_SESS\_END}$, BSESSEND must be set to 1. When VBUS is above $V_{B\_SESS\_END}$, BSESSEND must be set to 0. This information is only relevant when connected as B-device (OTG_CTRL.ID = 1) and OTG is enabled (OTG_EN = 1). 0: VBUS voltage is above $V_{B\_SESS\_END}$. 1: VBUS voltage is below $V_{B\_SESS\_END}$. This register is cleared 0x0 by soft reset or hardware reset.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
18	BSESSVLD	Current B-device session valid value. When OTG is enabled (OTG_EN = 1) and OMAP5912 is connected as a dual-role B-device (OTG_CTRL.ID = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{B\_SESS\_VLD}$. When VBUS is above $V_{B\_SESS\_VLD}$, BSESSVLD must be set to 1. When VBUS is below $V_{B\_SESS\_VLD}$, BSESSVLD must be set to 0. 0: VBUS voltage is below $V_{B\_SESS\_VLD}$. 1: VBUS voltage is above $V_{B\_SESS\_VLD}$. This register is cleared 0x0 by soft reset or hardware reset. When OTG is disabled, OTG_PADEN is 0, and the USB device controller is being used, this bit must be programmed to reflect whether VBUS is high enough to allow the OMAP5912 USB device controller to function properly. 0: VBUS voltage is below $V_{A\_VBUS\_VLD}$ (if OTG is enabled), or VBUS is insufficient to allow OMAP5912 USB device controller functionality (if OTG is disabled and OTG_PADEN is 0). 1: VBUS voltage is above $V_{A\_VBUS\_VLD}$ (if OTG is enabled), or VBUS is sufficient to allow OMAP5912 USB device controller functionality (if OTG is disabled and OTG_PADEN is 0).
17	VBUSVLD	Current VBUS valid value. When OTG is enabled (OTG_EN = 1), this bit must be programmed to reflect the OTG transceiver VBUS voltage comparator that compares against $V_{A\_VBUS\_VLD}$. When VBUS is above $V_{A\_VBUS\_VLD}$, VBUSVLD must be set to 1. When VBUS is below $V_{A\_VBUS\_VLD}$, VBUSVLD must be set to 0. This bit has no meaning when acting as an OTG default-B device (OTG ID = 1 and OTG_EN = 1). This register is cleared 0x0 by soft reset or hardware reset.
16	ID	Current ID pin value. When OTG is enabled (OTG_EN = 1), system software must program this bit to reflect the OTG transceiver ID pin sensor status any time ID changes. When OTG is disabled (OTG_EN = 0) this bit has no meaning. 0: ID pin is grounded. 1: ID pin is high-impedance. This register is cleared 0x0 by soft reset or hardware reset. The OTG controller does not understand resistive ID connectivity as can be found in a car kit environment. When an OTG transceiver sees ID pin resistively tied, OTG is not possible and OTG_EN must not be set.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
15	DRIVER_SEL	Active controller/driver software. When OTG is enabled (OTG_EN = 1), this read-only bit determines which OMAP5912 USB controller (and therefore software driver) has ownership of the OTG link. When HNP occurs, the OTG controller updates this bit and issues a driver change interrupt. When OTG is disabled (OTG_EN = 0), this bit has no meaning. When a write changes OTG_EN from 0 to 1, this bit is updated to reflect the value of OTG_CTRL.ID. 0: Host driver has control of the OTG link. 1: Device driver has control of the OTG link. This bit is set to 1 by soft reset or hardware reset.
14:13	Reserved	Reserved
12	A_SETB_HNPEN	B-device HNP indication (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP5912 acts as a default-A device (OTG_CTRL.ID=0), this bit must be programmed to reflect whether or not the B-device has been enabled to issue HNP. This bit has no effect when OMAP5912 acts as a default-B device (OTG_CTRL.ID=1) or when OTG is disabled (OTG_EN = 0). 0: The B-device has not been enabled to issue HNP. The OMAP5912 OTG controller does not respond to HNP issued by the B-device. 1: The B-device has been enabled to issue HNP. The OMAP5912 OTG controller responds to HNP issued by the B-device. This bit is cleared 0 by soft reset or hardware reset.
11	A_BUSREQ	Bus request (when acting as default-A device). When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a default-A device (OTG_CTRL.ID = 0), system software must set this bit when it wishes to begin an OTG session. When acting as the OTG A-device, system software can suspend the appropriate USB host controller port and then clear this bit to allow HNP requests from the B-device. If no HNP occurs, system software can set this bit to resume ownership of the OTG link by the OMAP5912 USB host controller. This bit has no effect if OTG_EN = 0 or if OTG_CTRL.ID = 1. 0: A-device does not request host role on the bus. Depending on OTG state machine state, either the session can end or the default-B device can issue an HNP. 1: A-device requests host role on the bus. OTG state machine begins a session (if the OTG session is ended) or attempts to return to ownership of the OTG link to the OMAP5912 USB host controller as soon as possible. This bit is cleared 0 by soft reset, hardware reset, or when the A_REQ_TMROUT interrupt is set.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
10	Reserved	Reserved
9	B_HNPEN	B-device HNP enable (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a default-B device (OTG_CTRL.ID = 1), system software must set this bit when the USB device receives a set feature operation with feature selector B_HNP_ENABLE. System software clears this bit whenever it sends USB reset to the default−B device. This bit has no meaning when OTG_EN = 0 or when OTG_CTRL.ID = 0. 0: B-device is not HNP enabled. The OTG controller does not issue HNP. 1: B-device is HNP enabled. The OTG controller can issue HNP when B_BUSREQ is set. This bit is cleared 0 by soft reset or hardware reset.
8	B_BUSREQ	Bus request (when acting as default-B device). When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a default-B device (OTG_CTRL.ID = 1), system software sets this bit when the application wishes to take ownership of the OTG link and begin acting as a host. When set, this allows the OTG controller to issue SRP if a session is not currently valid, and to begin HNP at the next available opportunity if HNP is enabled (OTG_CTRL.B_HNPEN = 1). This bit is ignored by the OTG controller when OTG_CTRL.ID = 0. 0: System software does not currently wish to begin acting as host. 1: System software (acting as a default-B device) wishes to begin acting as host. HNP and, if necessary, SRP are issued. This bit is cleared 0 by soft reset or hardware reset. The OTG controller clears this bit if there is not a valid session and the SRP fails. The OTG controller clears this bit if there is a valid session but the HNP fails. If SRP or HNP fails, system software repeats the request several times. If the SRP or HNP request fails several times, system software must inform the user of the failure.
7	OTG_BUSDROP	OTG bus drop request. When OTG is enabled (OTG_EN = 1) and OMAP5912 is acting as a default-A device (OTG_CTRL.ID = 0), system software requests the end of a session by setting this bit. This bit has no effect when OTG_CTRL.ID = 1. 0: System software does not require the end of the OTG session. 1: System software requires that the OTG session be ended. This bit is cleared 0 by soft reset or hardware reset.
6	Reserved	Reserved

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
5	OTG_PD	D+ pulldown enable. When OTG is enabled (OTG_EN = 1), this read-only register indicates whether the OTG transceiver applies a pulldown to D+. When OTG is disabled, this bit has no meaning. 0: OTG transceiver does not activate the D+ pulldown. 1: OTG transceiver activates the D+ pulldown. This bit is cleared 0 by soft reset or hardware reset. This register can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG = 1, OTG_PD retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
4	OTG_PU	D+ pullup enable. When OTG is enabled (OTG_EN = 1), this register indicates whether the OTG transceiver applies a pullup to D+. When OTG is disabled, this bit has no meaning. 0: OTG transceiver disables the D+ pullup. 1: OTG transceiver activates the D+ pullup. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_PU retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared. When OTG_SYSCON_2.B_ASE0_RST = 4 and acting as default-B device (OTG_CTRL.ID = 1), and system software is performing HNP, the software must write a 1 to this register when it knows that the OTG controller D+ pullup is active (typically upon completion of I^2C activity).
3	OTG_DRV_VBUS	VBUS drive enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver drives VBUS. When OTG is disabled (OTG_EN = 0), this bit has no meaning. Driving VBUS is requested only when OMAP5912 acts as a default-A device and an OTG session is needed. 0: OTG transceiver does not drive VBUS. 1: OTG transceiver drives VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG = 1, OTG_DRV_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.

Table 67. OTG Control Register (OTG_CTRL) (Continued)

Bit	Name	Description
2	OTG_PD_VBUS	VBUS pulldown enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver is to discharge VBUS when generating SRP. Driving VBUS is requested only when OMAP5912 acts as a default-A device and an OTG session is needed or is in progress. When OTG is disabled (OTG_EN = 0), this bit has no meaning. Discharge of VBUS is requested only when OMAP5912 is acting as a default-B device, an OTG session needs to be requested, and OTG_SYSCON_2.SRP_GPDVBUS = 1. 0: OTG transceiver does not discharge VBUS. 1: OTG transceiver discharges VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_PD_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
1	OTG_PU_VBUS	VBUS pullup enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver charges VBUS. Charging VBUS is only used when OMAP5912 acts as a default-B device and is performing SRP. When OTG is disabled (OTG_EN = 0), this bit has no meaning. 0: OTG transceiver does not charge VBUS. 1: OTG transceiver charges VBUS. This bit is cleared 0 by soft reset or hardware reset. This bit can be polled when the OTG_IRQ_EN:OPRT_CHG_EN = 0. When OTG_IRQ_EN.OPRT_CHG_EN = 1 and OTG_IRQ_SRC:OPRT_CHG =1, OTG_DRV_VBUS retains its value until after OTG_IRQ_SRC.OPRT_CHG is cleared.
0	OTG_PU_ID	ID signal pullup enable. When OTG is enabled (OTG_EN = 1), this read-only bit indicates whether the OTG transceiver applies a pullup to the ID pin to assist in detection of the ID pin level. System software for implementations using typical OTG transceivers with I^{2}C interfaces ignore this bit. 0: OTG transceiver does not apply a pullup to ID. 1: OTG transceiver applies a pullup to ID. This bit is cleared 0 by soft reset or hardware reset. This register can be used for polling when the OTG_IRQ_EN:OPRT_CHG_EN is cleared 0. Otherwise, this information is frozen when the interrupt status OTG_IRQ_SRC:OPRT_CHG is active 1.

This read/write register reflects the status of some USB OTG control bits. System software uses these register bits to convey OTG transceiver control and status between the transceiver and the OTG controller.

The bits OTG_PU_ID, OTG_PU_VBUS, OTG_PD_VBUS, OTG_DRV_VBUS, OTG_PU, and OTG_PD are controlled by the OTG controller and determine how software configures the OTG transceiver. Whenever one of these bits is changed by the OTG controller, an OPRT_CHG interrupt is generated (if enabled). It is best for system software to implement an interrupt handler that updates the OTG transceiver control registers when the OPRT_CHG interrupt occurs.

The bits ID, VBUSVLD, BSESSVLD, BSESSEND, and ASESSVLD must be updated to reflect the OTG transceiver status. Typically, the OTG transceiver provides an interrupt that can be configured to assert when the OTG transceiver status changes. This interrupt output is generally connected to an OMAP5912 GPIO input that is configured as an MPU interrupt source. The interrupt service routine for the transceiver interrupt must query the OTG transceiver interrupt and appropriate status bits via I^{2}C operations and update the status bits in OTG_CTRL[31:27].

The bits OTG_BUSDROP, B_BUSREQ, B_HNPEN, A_BUSREQ, and A_SETB_HNPEN control the OTG controller state machines and provide functionality defined in the OTG supplement. System software must manage these bits as described below.

The DRIVER_SEL bit shows whether the USB host controller or the USB device controller has control of the OTG link.

Table 68. OTG Interrupt Enable Register (OTG_IRQ_EN)

Bit	Name	Description
15	DRIVER_SWITCH_EN	Driver switch interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC: DRIVER_SWITCH bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: DRIVER_SWITCH bit is set. This bit is cleared to 0 by soft reset or hardware reset.
14	Reserved	Reserved

Table 68. OTG Interrupt Enable Register (OTG_IRQ_EN) (Continued)

Bit	Name	Description
13	A_VBUS_ERR_EN	A-device Vbus error interrupt enable: This bit enables interrupt generation when the OTG_IRQ_SRC: A_VBUS_ERr bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_VBUS_ERR bit is set. This bit is cleared to 0 by soft reset or hardware reset.
12	A_REQ_TMROUT_EN	A-device request time-out interrupt enable. This bit enables generation of an interrupt when the OTG_IRQ_SRC: A_REQ_TMROUT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_REQ_TMROUT bit is set. This bit is cleared 0 by soft reset or hardware reset.
11	A_SRP_DETECT_EN	A-device SRP detection interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC: A_SRP_DETECT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC: A_SRP_DETECT bit is set. This bit is cleared to 0 by soft reset or hardware reset.
10	B_HNP_FAIL_EN	B-device HNP failed interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_HNP_FAIL bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_HNP_FAIL bit is set. This bit is cleared to 0 by soft reset or hardware reset.
9	B_SRP_TMROUT_EN	B-device SRP time-out interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_TMROUT bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_TMROUT bit is set. This bit is cleared to 0 by soft reset or hardware reset.

Table 68. OTG Interrupt Enable Register (OTG_IRQ_EN) (Continued)

Bit	Name	Description
8	B_SRP_DONE_EN	B-device SRP done interrupt enable.This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_DONE bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_DONE bit is set. This bit is cleared 0 by soft reset or hardware reset.
7	B_SRP_STARTED_EN	B-device SRP started interrupt enable. This bit enables interrupt generation when the OTG_IRQ_SRC:B_SRP_STARTED bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:B_SRP_STARTED bit is set. This bit is cleared 0 by soft reset or hardware reset.
6:1	Reserved	Reserved
0	OPRT_CHG_EN	OTG output port status change interrupt enable: this bit enables interrupt generation when the OTG_IRQ_SRC:OPRT_CHG bit is active. 0: No interrupt is generated. 1: An interrupt is generated if the OTG_IRQ_SRC:OPRT_CHG bit is set. This bit is cleared 0 by soft reset or hardware reset.

This read/write register controls which OTG controller interrupts are passed to the MPU level 2 interrupt controller IRQ_8.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC)

Bit	Name	Description
15	DRIVER_SWITCH	Driver switch interrupt status. This bit reflects the status of the driver switch interrupt source. Driver switch interrupts occur when HNP completes and control of the OTG link must switch between the OMAP5912 USB host controller driver and the OMAP5912 USB device controller driver (or vice versa).This interrupt also occurs upon enabling the OTG functionality by writing OTG_SYSCON_2.OTG_EN = 1. In this case, OGT_CTRL.DRIVER_SEL selects the controller driver that is appropriate, given the value of OTG_CTRL.ID. 0: Driver switch interrupt is inactive. 1: Driver switch interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
14	Reserved	Reserved
13	A_VBUS_ERR	A-device Vbus error interrupt status. This bit reflects the status of the A_VBUS_ERR interrupt. VBUS errors occur when the OMAP5912 OTG controller state machine acts as a default-A dual-role device and transitions into state A_VBUS_ERR. Usually, the state machine transitions into state A_VBUS_ERR if VBUS voltage drops below the A_VBUS_VALID threshold because of low battery conditions or heavy loading by the attached device. Transitions into this state can unintentionally occur if system software turns off VBUS power and writes 0 to OTG_CTRL.VBUSVLD before writing 1 to OTG_CTRL.OTG _BUS_DROP. Writing 1 to OTG_CTRL.OTG_BUS_DROP clears the source of this interrupt. 0: VBUS error interrupt is inactive. 1:VBUS error interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
12	A_REQ_TMROUT	A-device request time-out interrupt status. This bit reflects the status of the A_REQ_TMROUT interrupt. This interrupt occurs when the OMAP5912 OTG controller state machine acts as a default-A dual-role device and transitions from state a_wait_bcon to a_wait_vfall because the state machine did not see an attach. This interrupt can occur if system software attempts to power up an OTG link when the cable is not attached at both ends, or if the device at the far end of the cable does not provide a pullup resistor. 0: A-device request time-out interrupt is inactive. 1: A-device request time-out interrupt is active. This bit is cleared either by writing a 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
11	A_SRP_DETECT	A-device SRP detection interrupt status.This bit reflects the status of SRP detection when the OMAP5912 OTG controller acts as an OTG default-A device. When the OTG controller sees a valid and enabled SRP method, this interrupt is generated. SRP is not detected if OTG_CTRL.OTG_BUS_DROP is 1 or if OTG_CTRL.A_BUSREQ is active. 0: SRP detection interrupt is inactive. 1: An SRP has been detected, using the VBUS pulsing and/or data pulsing detection mechanism. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
10	B_HNP_FAIL	B-device HNP failed interrupt status. This bit reflects interrupts that are generated when the OMAP5912 OTG controller acts as an OTG default-B device and an HNP it attempts to issue fails. This interrupt is generated when the default-A device does not enable its D+ pullup in response to the HNP request within the allotted time. This interrupt is also generated if the OMAP5912 OTG controller acting as a default-B device issues HNP, but the default-A device issues a USB resume. A third instance where this interrupt can be generated is if OMAP5912 acting as a default-B device issues an HNP request, but VBUS fails before completion of the HNP. 0: B-device HNP failure interrupt is inactive. 1: B-device HNP failure interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
9	B_SRP_TMROUT	B-device SRP time-out interrupt status.This bit reflects interrupts that are generated when the OMAP5912 OTG controller acts as an OTG default-B device and issues an SRP request, but the default-A device does not power VBUS within 5.5 seconds of the beginning of the SRP request. When this interrupt occurs, system software must inform the user that something is wrong (such as bad or unconnected cabling). 0: SRP time-out interrupt is inactive. 1: SRP time-out interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.The SRP timer starts when the B-device bus request is activated.
8	B_SRP_DONE	B-device SRP done interrupt status. This bit reflects interrupts that are generated upon completion of an SRP when the OMAP5912 OTG controller acts as an OTG default-B device. The actual duration of SRP depends on the values programmed into OTG_SYSCON_2.SRP_GPDATA and OTG_SYSCON_2.SRP_GPUVBUS. 0: SRP done interrupt is inactive. 1: SRP done interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.
7	B_SRP_STARTED	B-device SRP started interrupt status. This bit reflects interrupts that are generated when the OMAP5912 OTG controller acts as a default-B device and begins to pulse D+ at the beginning of an SRP request. This interrupt does not occur when OTG_CTRL.BSESSVLD is 1. 0: SRP begin interrupt is inactive. 1: SRP begin interrupt is active. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

Table 69. OTG Interrupt StatusRegister (OTG_IRQ_SRC) (Continued)

Bit	Name	Description
6:1	Reserved	Reserved
0	OPRT_CHG	OTG output port status change interrupt status. This bit reflects interrupts that are generated when the OMAP5912 OTG controller requires a change in the OTG transceiver control signals. Any change in OTG_CTRL.[5:0] causes this interrupt. These interrupts include changes to D+ pullup or pulldown, VBUS drive, pullup, and pulldown, or ID pin pullup. Generally, system software responds to this interrupt by issuing I^2C operations to change the control registers in the OTG transceiver. When this bit is 1, OTG_CTRL.[5:0] reflect their values at the time that the interrupt is generated. Other changes on OTG_CTRL.[5:0] can occur before the interrupt is processed but these changes are not shown in OTG_CTRL.[5:0] until after the output port status change interrupt status is cleared. 0: Output port status change interrupt is inactive. 1: An interrupt is generated for a change on OTG output ports status. This bit is cleared either by writing 1 or by soft reset or hardware reset. This bit has no effect when OTG_SYSCON_2.OTG_EN is 0.

This read/write register reflects all OTG interrupt status.

Table 70. OTG Output pins control register (OTG_OUTCTRL)

Bit	Name	Description
15	RESERVED	Reserved
14	OTGVPD	OTG Pull down VBUS
13	OTGVPU	OTG Pull up VBUS
12	OTGPUID	OTG Pull up ID
11	RESERVED	Reserved
10	USB2VDR	USB2 Port Vbus drive
9	USB2PDEN	USB2 Port Pull Down enabled
8	USB2PUEN	USB2 Port Pull Up enabled
7	RESERVED	Reserved
6	USB1VDR	USB1 Port Vbus drive
5	USB1PDEN	USB1 Port Pull Down enabled

Table 70. OTG Output pins control register (OTG_OUTCTRL) (Continued)

Bit	Name	Description
4	USB1PUEN	USB1 Port Pull Up enabled
3	RESERVED	Reserved
2	USB0VDR	USB0 Port Vbus drive
1	USB0PDEN	USB0 Port Pull Down enabled
0	USB0PUEN	USB0 Port Pull Up enabled

Table 71. OTG test register (OTG_TEST)

Bit	Name	Description
15	TEST_UNLOCK	Test mode unlock
14:9	RESERVED	Reserved
8	IRQ_OTG	IRQOTGON signal control
7:0	OTG_FSM_STATE	OTG FSM state

Table 72. OTG Vendor Code Register (OTG_ VC)

Bit	Name	Description
31:16	Reserved	Reserved
15:0	VC	Vendor code identifier: this read-only register reflects the Texas Instruments vendor code identifier: 0x5449 for TI in ASCII.

This register reflects the binary coded decimal equivalent of the USB vendor code identifier assigned to Texas Instruments.

4.2.1 OTG Clock and Reset Requirements

The OTG module is clocked mainly by the 48-MHz input clock from the OMAP5912 ULPD module. The ULPD module registers control the 48 MHz.

❏ CLOCK_CTRL_REG.USB_MCLK_EN
❏ SOFT_REQ_REG.SOFT_USB_OTG_DPLL_REQ
❏ SOFT_DISABLE_REQ_REG.DIS_USB_HOST_DPLL_REQ

When the OTG module receives a 48-MHz clock from the ULPD module, the OTG controller logic can have its clocks disabled locally via the OTG_SYSCON_1.OTG_IDLE_EN bit. The OTG module register interface is clocked separately, so OTG module registers other than OTG_SYSCON_2 and OTG_CTRL can be accessed when the 48-MHz clock input is inactive. Accesses to OTG_SYSCON_2 and OTG_CTRL require that the OTG controller 48-MHz clock be active.

The OTG controller 48-MHz clock must be enabled at both the ULPD level and the OTG module level whenever USB On-The-Go functionality is needed. It is not appropriate to disable the OTG controller 48-MHz clock when a USB On-The-Go link is established with another USB On-The-Go dual-role device.

The OTG controller uses the same 48-MHz clock input from the ULPD module to provide clocks to the USB device controller and the USB host controller. When the ULPD 48-MHz clock to the OTG module is disabled, the USB host controller registers cannot be accessed and the USB host controller cannot respond to any downstream USB bus activity.

The OTG module can disable the USB device controller clock using OTG_SYSCON_1.DEV_IDLE_EN. When the ULPD 48-MHz clock to the OTG module is disabled, or when OTG_SYSCON_1.DEV_IDLE_EN is 1, the USB device controller registers cannot be accessed, but the USB device controller can respond to USB bus resume signaling by issuing a USB device controller general interrupt. System software can read USB device controller registers when OTG_SYSCON_1.DEV_IDLE_EN is 1, but must enable the 48-MHz clock to the USB device controller before writing to the USB device controller registers.

Hardware reset of the USB OTG module is provided by the ULPD module. The PER_EN bit in the MPU reset control 2 register controls the reset to many OMAP5912 peripherals, including the USB OTG module. When held in hardware reset, the USB OTG controller cannot perform USB On-The-Go SRP or HNP protocols, and its registers cannot be accessed.

The OTG controller provides a soft reset mechanism. When OTG_SYSCON_1.SOFT_RESET is written with a 1, the OTG controller, the USB device controller, and the USB host controller are reset. Soft reset is complete when OTG_SYSCON_1.RESET_DONE is 1. OTG_SYSCON_1.SOFT_RESET automatically clears itself.

4.2.2 OTG Controller Power Management

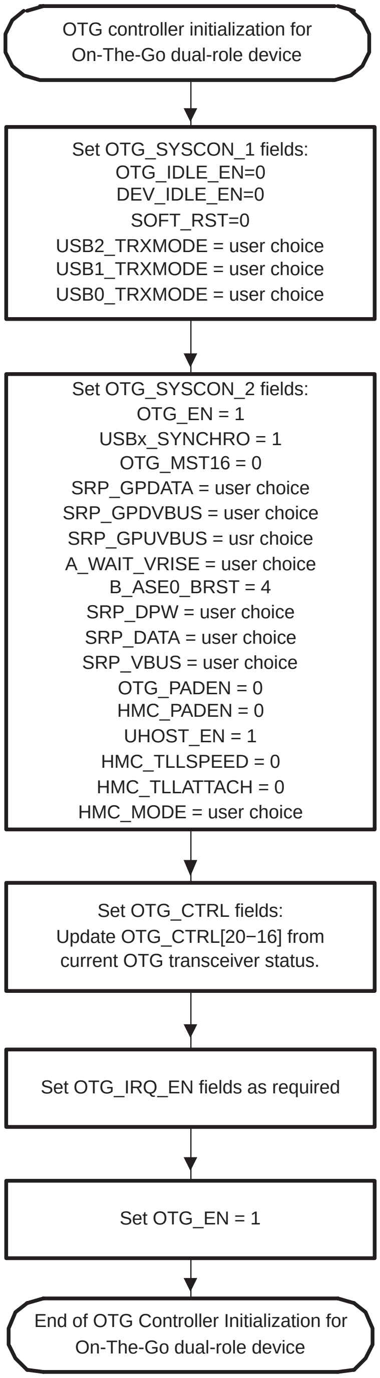
OTG controller power management is provided using the OTG_SYSCON_2.OTG_EN bit. When OTG_SYSCON_2.OTG_EN is 0, OTG controller power consumption is reduced, because the OTG controller logic is not clocked. The OTG controller cannot perform On-The-Go HNP or SRP when it is not clocked.

4.2.3 OTG Controller Initialization

OTG controller initialization operations depend on whether or not the system implements an On-The-Go dual-role device.

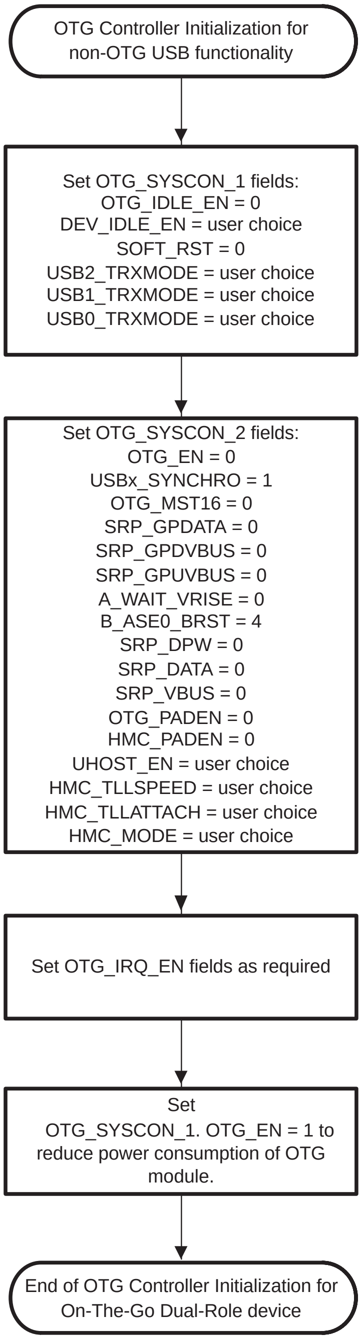
If the application implements an OTG dual-role device, follow the initialization shown in Figure 48.

Figure 48. OTG Controller Initialization When Implementing an OTG Dual-Role Device



Systems that implement USB functionality without implementing an OTG dual-role device are recommended to follow the initialization shown in Figure 49.

Figure 49. OTG Controller Initialization When Not Implementing an OTG Dual-Role Device


OTG Controller Initialization for non-OTG USB functionality
Set OTG_SYSCON_1 fields:
OTG_IDLE_EN = 0
DEV_IDLE_EN = user choice
SOFT_RST = 0
USB2_TRXMODE = user choice
USB1_TRXMODE = user choice
USB0_TRXMODE = user choice
Set OTG_SYSCON_2 fields:
OTG_EN = 0
USBx_SYNCHRO = 1
OTG_MST16 = 0
SRP_GPDATA = 0
SRP_GPDVBUS = 0
SRP_GPUVBUS = 0
A_WAIT_VRISE = 0
B_ASE0_BRST = 4
SRP_DPW = 0
SRP_DATA = 0
SRP_VBUS = 0
OTG_PADEN = 0
HMC_PADEN = 0
UHOST_EN = user choice
HMC_TLLSPEED = user choice
HMC_TLLATTACH = user choice
HMC_MODE = user choice
Set OTG_IRQ_EN fields as required
Set OTG_SYSCON_1. OTG_EN = 1 to reduce power consumption of OTG module.
End of OTG Controller Initialization for On-The-Go Dual-Role device

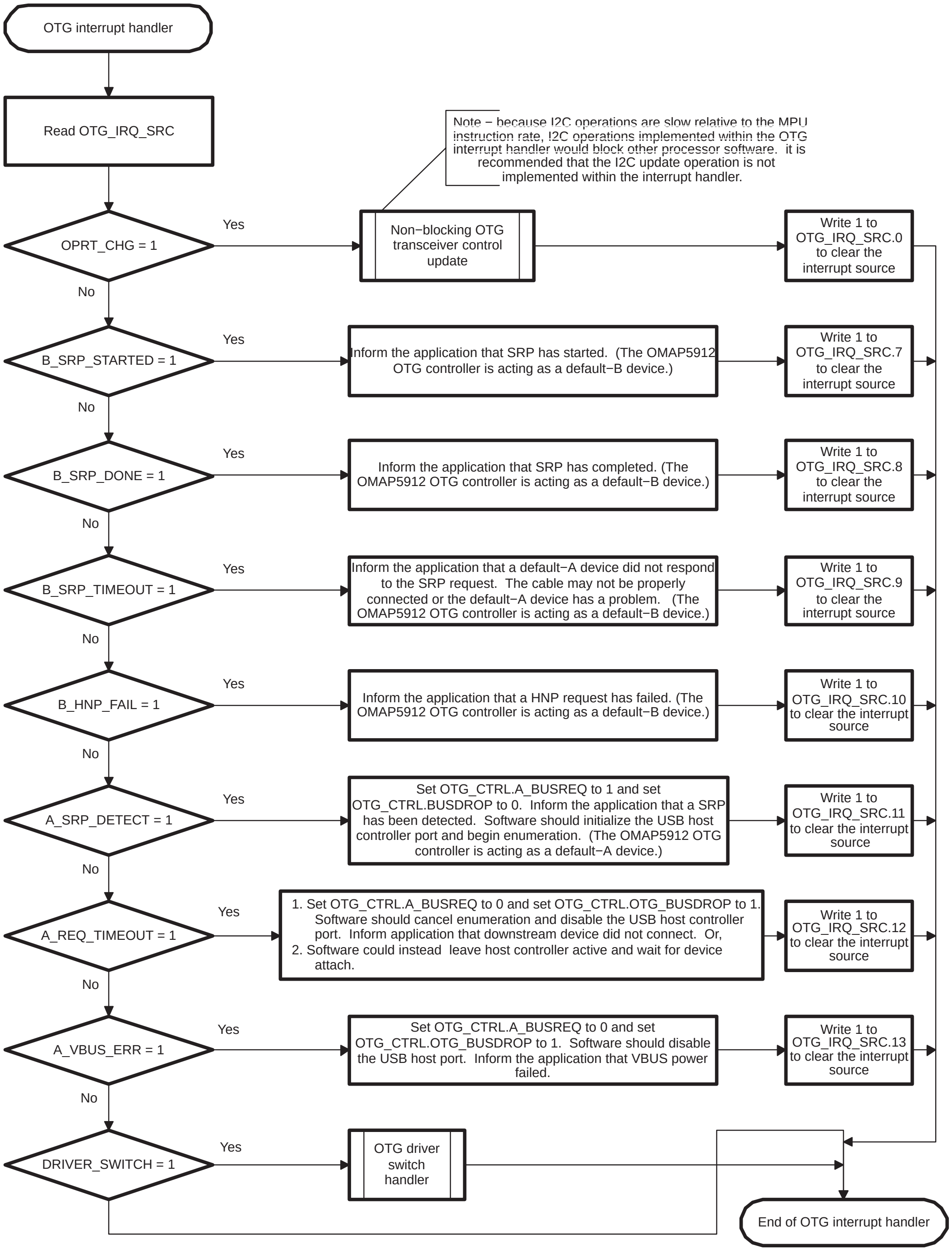
User choice values shown in Figure 48 and Figure 49 depend on the system implementation, including transceiver types being used, top-level pin multiplexing choice, and connector type. Pin multiplexing options and transceiver types are discussed in Section 4.3, *Pin Multiplexing*.

4.2.4 OTG Controller Interrupt Handler and Related Software

The OTG controller implements several interrupt sources that are separately enabled via bits in OTG_IRQ_EN. The OTG controller provides a single interrupt output that provides OTG controller interrupts to the MPU level 2 interrupt controller IRQ_8 input.

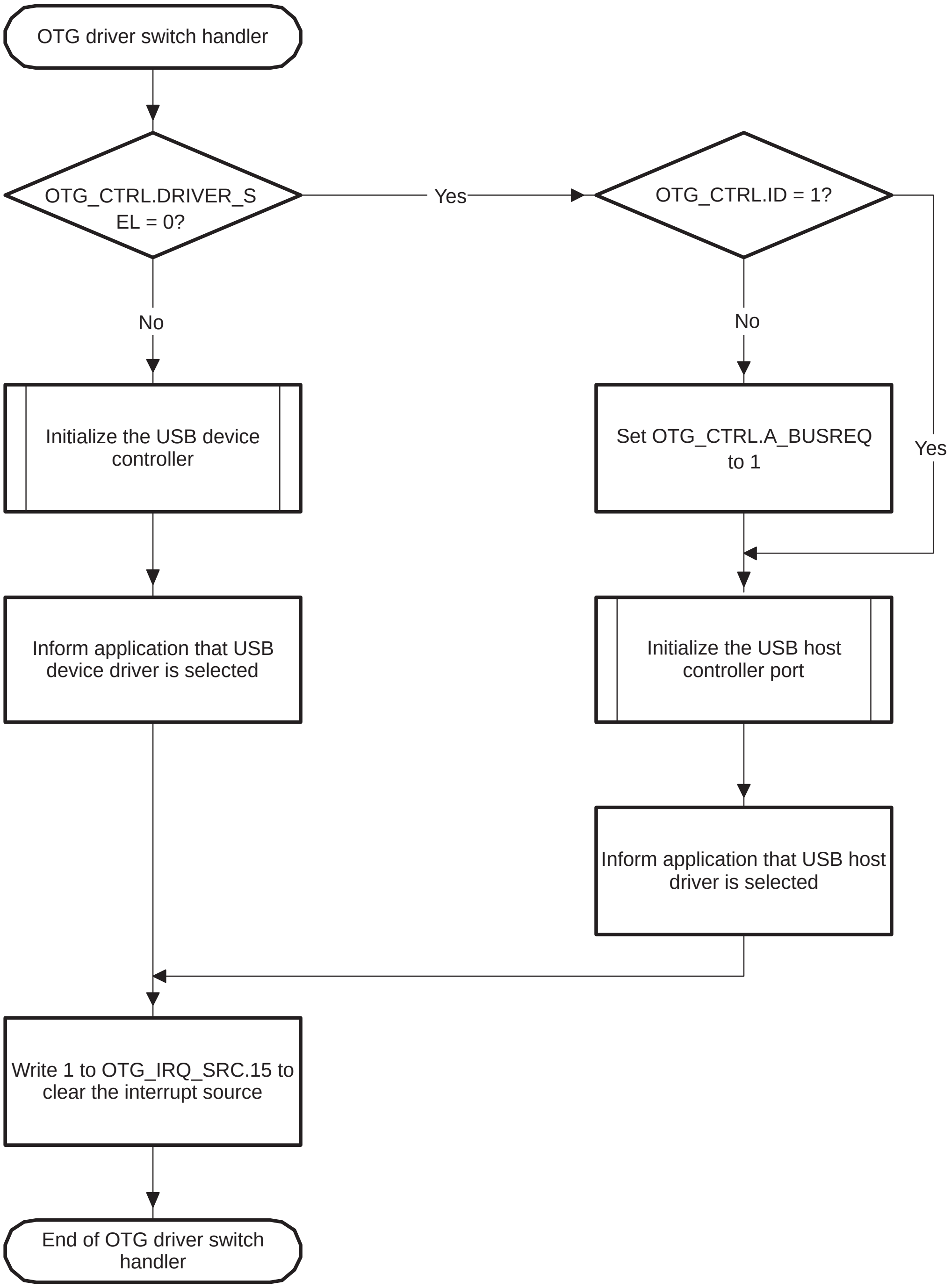
Figure 50 shows the operations required in the USB OTG controller interrupt handler. This flowchart references several other operations, which are described in flowcharts later in this section.

Figure 50. OTG Interrupt Handler


OTG interrupt handler
Read OTG_IRQ_SRC
Note − because I2C operations are slow relative to the MPU instruction rate, I2C operations implemented within the OTG interrupt handler would block other processor software. it is recommended that the I2C update operation is not implemented within the interrupt handler.
OPRT_CHG = 1
Yes
Non−blocking OTG transceiver control update
Write 1 to OTG_IRQ_SRC.0 to clear the interrupt source
No
B_SRP_STARTED = 1
Yes
Inform the application that SRP has started. (The OMAP5912 OTG controller is acting as a default−B device.)
Write 1 to OTG_IRQ_SRC.7 to clear the interrupt source
No
B_SRP_DONE = 1
Yes
Inform the application that SRP has completed. (The OMAP5912 OTG controller is acting as a default−B device.)
Write 1 to OTG_IRQ_SRC.8 to clear the interrupt source
No
B_SRP_TIMEOUT = 1
Yes
Inform the application that a default−A device did not respond to the SRP request. The cable may not be properly connected or the default−A device has a problem. (The OMAP5912 OTG controller is acting as a default−B device.)
Write 1 to OTG_IRQ_SRC.9 to clear the interrupt source
No
B_HNP_FAIL = 1
Yes
Inform the application that a HNP request has failed. (The OMAP5912 OTG controller is acting as a default−B device.)
Write 1 to OTG_IRQ_SRC.10 to clear the interrupt source
No
A_SRP_DETECT = 1
Yes
Set OTG_CTRL.A_BUSREQ to 1 and set OTG_CTRL.BUSDROP to 0. Inform the application that a SRP has been detected. Software should initialize the USB host controller port and begin enumeration. (The OMAP5912 OTG controller is acting as a default−A device.)
Write 1 to OTG_IRQ_SRC.11 to clear the interrupt source
No
A_REQ_TIMEOUT = 1
Yes
1. Set OTG_CTRL.A_BUSREQ to 0 and set OTG_CTRL.OTG_BUSDROP to 1. Software should cancel enumeration and disable the USB host controller port. Inform application that downstream device did not connect. Or,
2. Software could instead leave host controller active and wait for device attach.
Write 1 to OTG_IRQ_SRC.12 to clear the interrupt source
No
A_VBUS_ERR = 1
Yes
Set OTG_CTRL.A_BUSREQ to 0 and set OTG_CTRL.OTG_BUSDROP to 1. Software should disable the USB host port. Inform the application that VBUS power failed.
Write 1 to OTG_IRQ_SRC.13 to clear the interrupt source
No
DRIVER_SWITCH = 1
Yes
OTG driver switch handler
End of OTG interrupt handler

The OTG driver switch handler is shown in Figure 51. It is responsible for switching control of the OTG link between the USB host controller and the USB device controller. This switch occurs mainly because of HNP operations, but can also occur when OTG_SYSCON_2.OTG_EN is set to 1.

Figure 51. OTG Driver Switch Handler


OTG driver switch handler
OTG_CTRL.DRIVER_SEL = 0?
Yes
OTG_CTRL.ID = 1?
No
No
Yes
Initialize the USB device controller
Set OTG_CTRL.A_BUSREQ to 1
Inform application that USB device driver is selected
Initialize the USB host controller port
Inform application that USB host driver is selected
Write 1 to OTG_IRQ_SRC.15 to clear the interrupt source
End of OTG driver switch handler

The effects of On-The-Go functionality on OMAP5912 USB device controller operation are documented in the USB device controller flowcharts. Because USB host controller operations are mainly defined in the open host controller for USB specification, some On-The-Go functionality issues are discussed here.

Most OMAP5912 USB host controller software is not affected by USB On-The-Go functionality. A host controller driver written for non-OTG functionality needs few modifications to support On-The-Go functionality. In general, On-The-Go events precede USB host controller driver operations, so the On-The-Go event can cause the OTG driver software to pass the appropriate message to the USB host controller driver. It is never necessary for the USB host controller driver to access OTG controller registers.

For an On-The-Go connection, the SRP and HNP operations can be considered to replace the attach, detach, suspend, resume, and remote wake functions. This means that it can be beneficial for system software to control the USB host controller driver differently for an OTG link than for a typical USB host port.

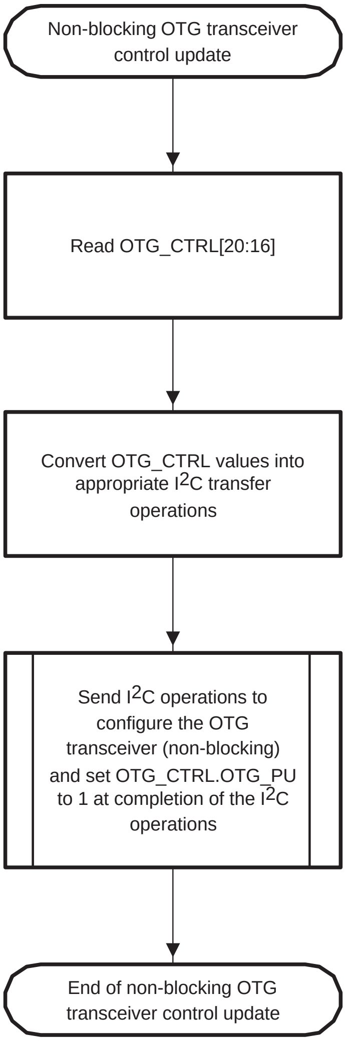
The USB host controller driver must implement a mechanism where the application can specify that the OTG port is no longer needed. The host controller driver software must respond to this message by suspending the port associated with the OTG link and cancelling all EDs and TDs associated with the OTG connection. These ED and TD removal activities are ones usually associated with a disconnect event, so the host controller driver already has these functions available. The application typically sends this message and then informs the OTG controller driver that its use of the host is complete and that the OTG controller can now release the bus for a possible HNP.

When the USB device controller owns the USB OTG link, the USB host controller port need not remain active. If no other USB host controller ports are being used, it is possible to disable the USB host controller clock for the duration while the USB device controller owns the OTG link. System software must re-enable the USB host controller clock and re-initialize the USB host on an OTG driver switch interrupt to USB host controller control of the OTG link if the host clock is dynamically disabled.

Software to support communication between the USB OTG controller and the external OTG transceiver via I^{2}C is described in Figure 52. This software must handle control information from the OTG controller to the external OTG. These software operations apply directly to the OTG controller OPRT_CHG interrupt. Because I^{2}C operations are slow (relative to the MPU instruction speed), it is best if these functions are not implemented as part of an interrupt handler;

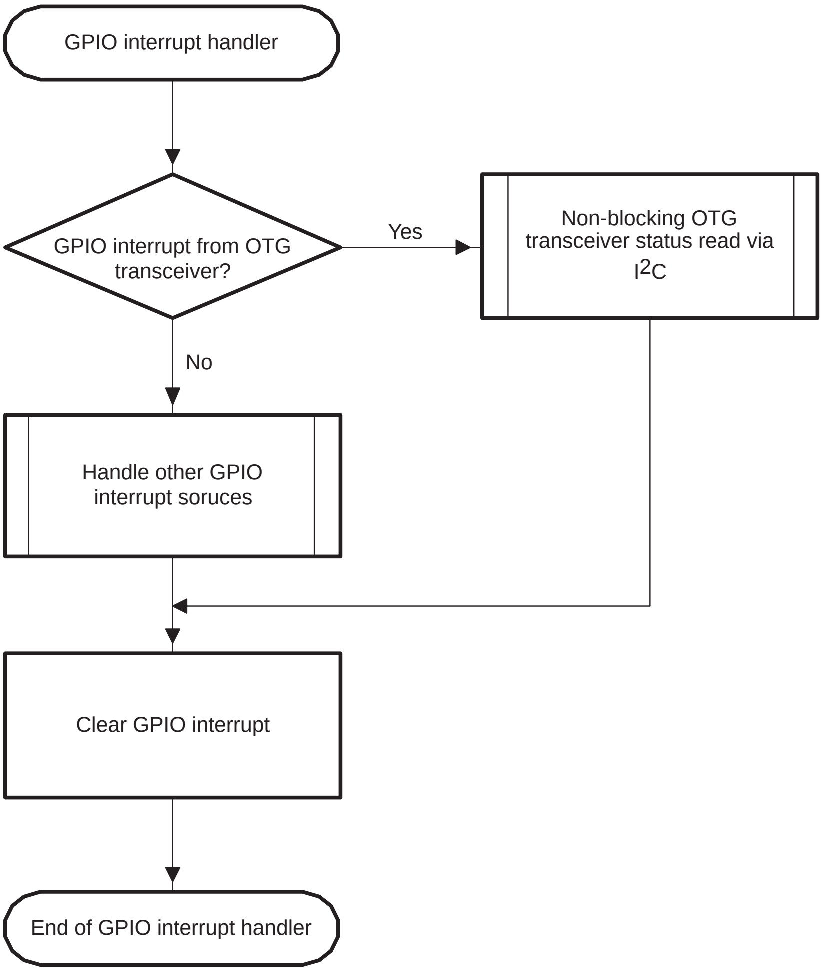
interrupt handlers block other processor software operations until the handler completes. System software must implement a non-blocking mechanism, such as a message-based system, to allow the I^{2}C operations to be completed outside of the OTG interrupt handler, but then allow updating of the OTG controller OTG_CTRL.OTG_PU bit on completion of the I^{2}C operations.

Figure 52. OTG Transceiver I^{2}C Control Handler


Non-blocking OTG transceiver control update
Read OTG_CTRL[20:16]
Convert OTG_CTRL values into appropriate I2C transfer operations
Send I2C operations to configure the OTG transceiver (non-blocking) and set OTG_CTRL.OTG_PU to 1 at completion of the I2C operations
End of non-blocking OTG transceiver control update

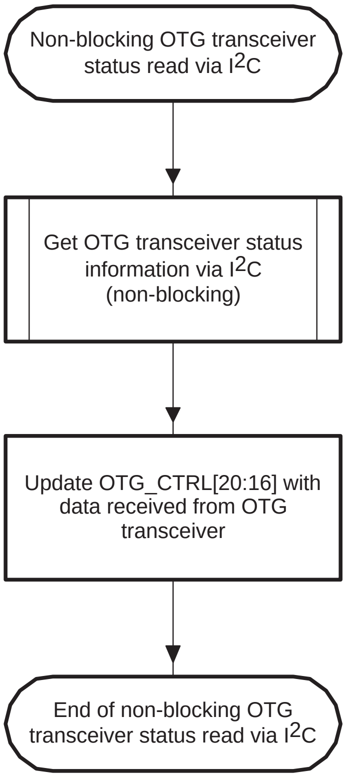
The OTG transceiver signals the need for a status transfer from the OTG transceiver to the OTG controller, using an interrupt output signal. This interrupt is generally connected to an OMAP5912 GPIO input pin, and that pin is configured to provide an interrupt to the MPU Level 2 interrupt controller. Figure 53 shows the basic GPIO interrupt handler functionality required.

Figure 53. GPIO Interrupt Handler Support for OTG Transceiver Interrupt Input


GPIO interrupt handler
GPIO interrupt from OTG transceiver?
Yes
Non-blocking OTG transceiver status read via I2C
No
Handle other GPIO interrupt soruces
Clear GPIO interrupt
End of GPIO interrupt handler

The operations in Figure 54 show the non-blocking OTG transceiver status read and update to the OTG controller registers. To reduce the effect on system performance, it is important that the I^{2}C read operations are non-blocking.

Figure 54. OTG Transceiver Status Read


Non-blocking OTG transceiver status read via I2C
Get OTG transceiver status information via I2C (non-blocking)
Update OTG_CTRL[20:16] with data received from OTG transceiver
End of non-blocking OTG transceiver status read via I2C

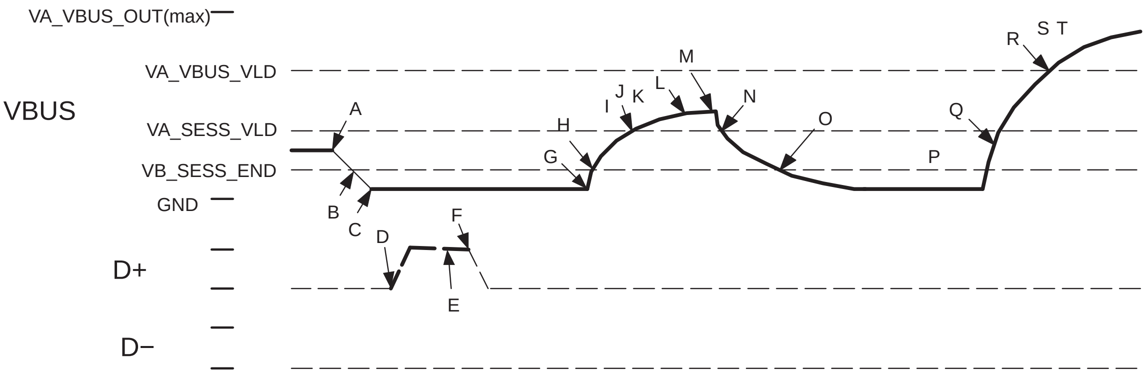
4.2.5 Typical SRP and HNP Events

This section describes the typical sequence of events for SRP and HNP events with the OMAP5912 OTG controller.

Careful system software design can optimize system performance by managing interrupt source enables at both the OTG controller and at the OTG transceiver. System software developers must carefully weigh the advantages brought by disabling unneeded OTG transceiver interrupt sources versus the performance cost associated with the I^{2}C activity to enable and disable the transceiver interrupt sources.

Figure 55 shows the typical events that occur when OMAP5912 acts as an OTG default-A dual-role device and the default-B device issues an SRP. Figure 55 shows that the default-B device pulses its D+ pullup resistor. The OMAP5912 OTG controller accepts D− pullup pulses as well as the D+ pullup pulse shown as SRP data line.

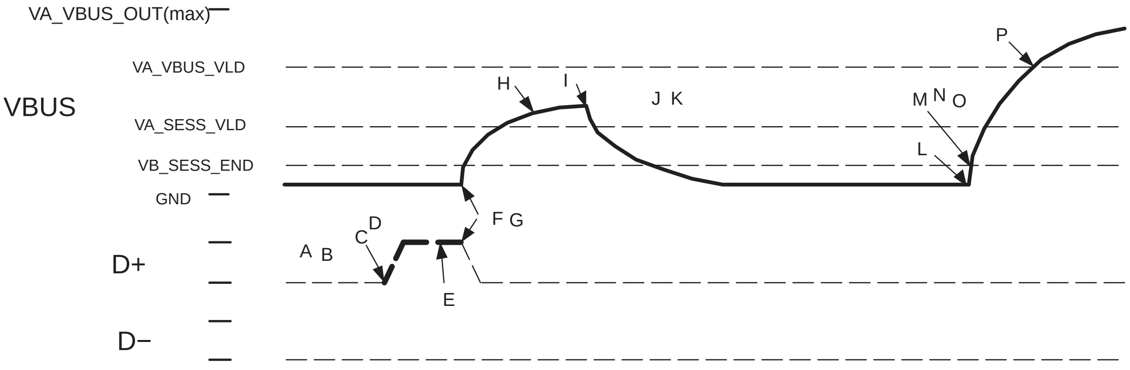
Figure 55. OMAP5912 OTG Controller Response To SRP When Acting as a Default-A Dual-Role OTG Device



A. OTG dual-role default-B device begins discharging VBUS (optional).
B. VBUS at default-B device falls below VB_SESS_END.
C. OTG dual-role default-B device stops discharging VBUS (optional).
D. Default-B device enables SRP D+ pullup.
E. OMAP5912 OTG controller recognizes D+ pulse SRP request if D+ pulse is longer than the duration programmed in OTG_SYSCON_2.SRP_DPW and if OTG_SYSCON_2.SRP_DATA =1. If recognized, OTG_CTRL.OTG_DRV_VBUS is set to 1 and OPRT_CHG and SRP_DETECT interrupts are generated.
F. Default-B device disables D+ pullup (to end SRP Data Line Pulse).
G. Initial conditions for SRP are satisfied. Default-B device enables VBUS SRP pulse.
H. VBUS voltage crosses VB_SESS_END at default-B.
I. VBUS voltage crosses VA_SESS_VLD at OMAP5912 OTG transceiver. OTG transceiver issues an interrupt.
J. OMAP5912 GPIO interrupt handler identifies OTG transceiver interrupt. Handler initiates I^{2}C operation to get transceiver interrupt source information.
K. OMAP5912 I^{2}C operations to get transceiver interrupt source information completes. System software sees VBUS > VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1.
L. OMAP5912 OTG controller recognizes VBUS pulse SRP request if OTG_CTRL.ASESSVLD is 1 for longer than the duration programmed in OTG_SYSCON_2.SRP_DPW and if OTG_SYSCON_2.SRP_VBUS=1. If recognized, OTG_CTRL.OTG_DRV_VBUS is set to 1 and OPRT_CHG and SRP_DETECT interrupts are generated.
M. Default-B device stops driving VBUS pulse.
N. VBUS voltage discharges through various leakage sources.
O. VBUS voltage drops below VB_SESS_END at default-B device.
P. OMAP5912 OTG Interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_DRV_VBUS set to 1 and initiates I^{2}C operations to configure OTG transceiver to drive VBUS. OTG interrupt handler also sees SRP_DETECT interrupt and begins initialization of OTG session as a default-A dual-role device.
Q. OMAP5912 I^{2}C operations to configure OTG transceiver to drive VBUS complete. System software writes a 1 to OTG_CTRL.OTG_PU to indicate that the I^{2}C operation has completed.
R. VBUS rises above VA_VBUS_VLD. OTG transceiver issues interrupt.
S. OMAP5912 GPIO interrupt handler sees OTG transceiver interrupt, initiates I^{2}C operations to get OTG interrupt source information.
T. OMAP5912 I2C operations complete. System software sees OTG status showing that VBUS voltage is above VA_SESS_VLD and sets OTG_CTRL.ASESSVLD to 1. OMAP5912 OTG dual-role device is now ready for default-B device to enable its pullup.

Figure 56 shows the typical events that occur when OMAP5912 acts as an OTG default-B dual-role device and issues an SRP to the default-A device. If OMAP5912 completes its SRP request but the default-A device does not drive VBUS within 5.5 seconds, the OMAP5912 OTG controller issues a B_SRP_TIMEOUT interrupt. System software must provide a message to the user that says the SRP request failed and that the user needs to check the cable and the A-device. The OMAP5912 OTG controller only implements D+ data line SRP pulses and does not request a D− pulse as a data line SRP pulse.

Figure 56. OMAP5912 OTG Controller SRP Generation When Acting as a Default-A Dual-Role OTG Device



A. OMAP5912 is acting as a default-B device. OTG_CTRL.BSESSEND=1. Application decides that it wants to communicate with the default-B dual-role device. System software sets OTG_CTRL.B_BUSREQ.

B. OMAP5912 OTG controller sees write of 1 to OTG_CTRL.B_BUSREQ and sees OTG_CTRL.BSESSEND=1. OTG controller waits until USB bus has been SE0 for the minimum time and then begins SRP process by setting OTG_CTRL.OTG_PU and issues the OPRT_CHG interrupt.

C. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 1. Handler initiates I^2C operation to enable OTG transceiver D+ pullup.

D. OMAP5912 I^2C operations to enable the D+ pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup enable operation has completed.

E. OMAP5912 OTG controller changes OTG_CTRL.OTG_PU to 0 approximately 9 ms after it had set it to 1. OTG controller also sets OTG_CTRL.OTG_VBUS_PU at the same time. OTG controller issues OPRT_CHG interrupt.

F. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0 and sees OTG_PU_VBUS = 1. Handler initiates I^2C operation to disable OTG transceiver D+ pullup and to enable OTG transceiver VBUS pullup.

G. OMAP5912 I^2C operations to disable the D+ pullup and enable the VBUS pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

H. OMAP5912 OTG controller changes OTG_CTRL.OTG_PU_VBUS to 0 after a time specified by OTG_SYSCON_2.SRP_GPUVBUS from the time the controller had set OTG_PU_VBUS to 1. If OTG_SYSCON_2.SRP_GPDVBUS = 1, OTG controller sets OTG_CTRL.OTG_PD_VBUS to 1. OTG controller issues OPRT_CHG interrupt.

I. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_CTRL.OTG_PU_VBUS = 0 and sees current value in OTG_CTRL.OTG_PD_VBUS. Handler initiates I^2C operations to disable the OTG transceiver VBUS pullup and to enable or disable the OTG transceiver VBUS pulldown depending on the value of OTG_PD_VBUS.

J. OMAP5912 OTG controller waits for the time specified by OTG_SYSCON_2.SRP_GPUVBUS from the time the controller had set OTG_PU_VBUS to 0. The OTG controller issues a B_SRP_DONE interrupt. If OTG_CTRL.OTG_PD_VBUS =1, the controller sets OTG_PD_VBUS to 0 and issues an OPRT_CHG interrupt.

K. OMAP5912 OTG interrupt handler sees OTG interrupt. If OPRT_CHG interrupt is active, interrupt handler sees that OTG_CTRL.OTG_PD_VBUS is 0 and initiates I^2C operations to disable the OTG transceiver VBUS pulldown.

L. Some time later, the default−A device responds to the SRP request by driving VBUS active.

M. The OTG transceiver sees VBUS rise above VB_SESS_VLD and issues an interrupt.

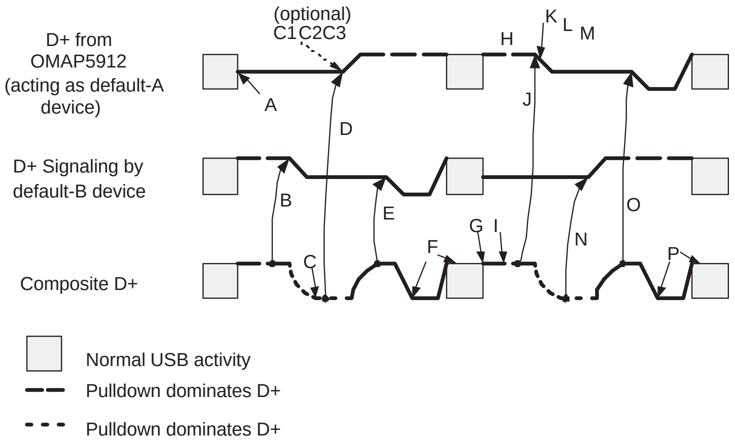
N. The OMAP5912 GPIO interrupt handler sees the OTG transceiver interrupt and initiates the I^2C operations which query the OTG transceiver interrupt status.

O. The I^2C operation completes, and OMAP5912 system software updates OTG_CTRL.ASESSVLD, OTG_CTRL.BSESSEND, OTG_CTRL.BSESSVLD, OTG_CTRL.VBUSVLD, and OTG_CTRL.ID. If BSESSVLD is 1, then the OMAP5912 USB device controller will see VBUS go active and can begin preparations for enumeration.

P. The default-A device sees VBUS rise above VA_VBUS_VLD and can begin USB reset and enumerating the OMAP5912 default-B device.

Figure 57 shows the typical events that occur when OMAP5912 acts as an OTG default-A dual-role device and transitions from acting as an A-host to acting as an A-peripheral via HNP, and then transitions back to acting as an A-host via HNP. The figure shows the optional activities associated with the OTG transceiver autoconnect feature.

Figure 57. OMAP5912 OTG Controller HNP Events When Acting as a Default-A Dual-Role OTG Device



A: OMAP5912 application has no more traffic for default-B device. System software suspends the USB host port which is being used for the OTG link, sets OTG_CTRL.A_BUSREQ to 0. If the OTG transceiver supports the autoconnect feature, system software should initiate the appropriate I^2C operations to enable it before suspending the OTG link.

B: Default-B device sees OTG link suspended by default-A device. Because the default-B device is enabled for HNP, and desires to issue HNP, it disables its D+ pullup, allowing D+ to float. Default-B controller waits for OMAP5912 to enable its D+ pullup.

C: OMAP5912 OTG controller sees OTG_CTRL.A_SETB_HNPEN=1 and sees SE0 on the bus. OTG controller sets OTG_CTRL.OTG_PU to 0 and issues both the OPRT_CHG interrupt and the driver switch interrupt.

C1. If the OTG transceiver autoconnect feature was enabled in step A, the OTG transceiver will automatically enable its D+ pullup upon seeing SE0 on the bus, and issue an interrupt.

C2.OMAP5912 GPIO interrupt handler sees an OTG transceiver interrupt and issues I^2C operations to query the transceiver interrupt source.

C3. OMAP5912 I^2C operation completes and the interrupt status shows autoconnect occurred. It is not necessary to update any OTG controller registers when auto-HNP occurs.

D. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and OTG_PU set to 1, and (if auto-HNP is not used) causes I_2C operation to enable OTG transceiver D+ pullup. OMAP5912 OTG interrupt handler sees driver switch interrupt and begins initialization of USB device controller.

E. Default-B device sees D+ pullup as HNP operation and begins acting as a USB host. It signals USB reset and enumerates the OMAP5912 dual-role device as a peripheral, and begins normal USB bus activity.

F. OMAP5912 USB device controller sees and responds appropriately to USB reset, enumeration and normal USB bus activity.

G. Default-B device finishes its bus activity and suspends the USB link.

H. OMAP5912 application decides that it wants to communicate with the default-B device. System software sets OTG_CTRL.A_BUSREQ.

I. OMAP5912 OTG controller sees suspend signaled on the OTG link, sets OTG_CTRL.OTG_PU to 0, issues OPRT_CHG interrupt.

J. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 0, issues I^2C operations to disable the OTG transceiver D+ pullup.

K. OMAP5912 I^2C operations to disable the D+ pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

L. OMAP5912 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interupt.

M. OMAP5912 OTG interrupt handler sees driver switch interrupt, begins intialization of host port.

N. Default-B device sees SE0 on the OTG link, turns on its D+ pullup.

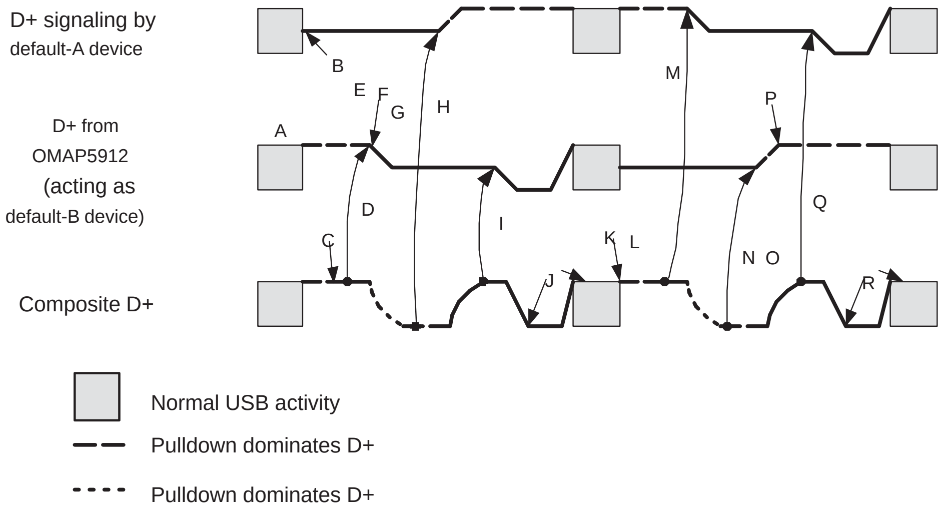
O. OMAP5912 OTG controller and USB host controller see D+ pullup as HNP operation. OMAP5912 USB host controller signals USB reset, sends the set feature signal with B_HNPEN feature enabled, and enumerates the default-B dual-role device as a peripheral, and begins normal USB bus activity.

P. Default-B device sees and responds appropriately to USB reset, the set feature signal, enumeration and normal USB bus activity.

Figure 58 shows the typical events that occur when OMAP5912 acts as an OTG default-B dual-role device and transitions from acting as a B-peripheral to acting as a B-host via HNP, and then transitions back to acting as a B-peripheral via HNP.

If the default-A device does not enable its pullup resistor within the required time, if the default-A device issues a USB resume during the HNP process, or if the VBUS voltage falls below VB_SESS_VLD threshold during the HNP process, the OMAP5912 OTG controller issues a B_HNP_FAIL interrupt.

Figure 58. OMAP5912 OTG Controller HNP Events When Acting as a Default-B Dual-Role OTG Device



A: OMAP5912 application decides that it wishes to communicate with the default-A device. System software sets OTG_CTRL.B_BUSREQ.

B: Default-A device has no more traffic for the default-B device (OMAP5912). Default-A device suspends the OTG link.

C: OMAP5912 OTG controller (acting as the OTG dual-role default-B device) sees OTG link suspended, sees OTG_CTRL.B_HNPEN set to 1, sees OTG_CTRL.B_BUSREQ set to 1. OTG controller sets OTG_CTRL.OTG_PU to 0 and issues OPRT_CHG interrupt.

D. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt and sees OTG_PU = 0. Handler initiates I^{2}C operation to disable OTG transceiver D+ pullup.

E. OMAP5912 I^{2}C operations to disable the D+ pullup complete. OMAP5912 software sets OTG_CTRL.OTG_PU to 1 to indicate that the D+ pullup disable operation has completed.

F. OMAP5912 OTG controller sees write of 1 to OTG_CTRL.OTG_PU and issues driver switch interupt.

G. OMAP5912 OTG interrupt handler sees driver switch interrupt, begins intialization of host port.

H. Default-A device sees SE0 on the OTG link, turns on its D+ pullup.

I. OMAP5912 OTG controller sees D+ pullup as HNP operation and OMAP5912 USB host contoller sees D+ pullup as an attach. OMAP5912 USB host controller signals USB reset, and enumerates the default-A dual-role device as a peripheral, and begins normal USB bus activity.

J. Default-A device sees and responds appropriately to USB reset, enumeration and normal USB bus activity.

K. OMAP5912 application decides it has no more information to communicate to the default-A device. System software suspends the USB host port which is being used for the OTG link, sets OTG_CTRL.A_BUSREQ to 0.

L. OMAP5912 OTG controller sees OTG_CTRL.A_BUSREQ set to 0. OTG controller sets OTG_CTRL.OTG_PU to 1 and issues both OPRT_CHG and driver switch interrupts.

M. Default-A device sees OTG link signal USB Suspend. Default-A device disables its D+ pullup.

N. OMAP5912 OTG interrupt handler sees OPRT_CHG interrupt, sees OTG_CTRL.OTG_PU set to 1, initiates I^{2}C operations to enable OTG transceiver D+ pullup.

O. OMAP5912 OTG interrupt handler sees Driver Switch and begins initialization of OMAP5912 USB device controller.

P. OMAP5912 I^{2}C operations to enable OTG transceiver D+ pullup complete.

Q. Default-A devcie sees D+ pullup, signals USB reset, issues Set Feature of the B_HNP_ENABLE feature, and enumerates the OMAP5912 default-B dual-role device as a peripheral, and begins normal USB bus activity.

R. OMAP5912 USB device controller sees and responds appropriately to USB reset, set feature of the B_HNP_ENABLE feature, enumeration, and normal USB bus activity.

4.2.6 System-Level OTG Considerations

The OMAP5912 USB OTG implementation allows up to two separate USB host controller ports to function in parallel with an OTG link. The USB OTG specification states that an OTG dual-role device has only one USB connector. If a system implements an OTG connector, the system can only use the other two USB host controller ports for on-board connectivity.

An OTG implementation using an OTG transceiver that is based on the *OTG Transceiver Interface Specification* can use the transceiver autoconnect feature. Enabling this OTG transceiver feature when OMAP5912 is acting as a default-A dual-role On-The-Go device eases the timing requirements on system software during an HNP transition from OMAP5912 acting as an A-host to OMAP5912 acting as an A-peripheral.

A system that implements an OTG controller has a USB mini-AB receptacle. This receptacle is used with an OTG cable to connect the system to another OTG system. It is possible to connect a non-OTG USB device to the OMAP5912-based OTG system mini-AB receptacle using an adapter cable as defined in the On-The-Go supplement to the USB specification. If the system supports this option, it may be necessary to provide a VBUS source with larger current source capability than is typically available through an OTG transceiver.

4.3 Pin Multiplexing

OMAP5912 USB signal multiplexing determines which USB functionality is available at which OMAP5912 pins. OMAP5912 provides three pin groups that can be configured to provide up to three simultaneous USB ports. These ports can be configured to provide a USB OTG port, a USB device port, and/or up to three USB host ports. When OMAP5912 is configured to provide USB OTG functionality at one pin group, it is not possible to configure another pin group to act as a USB device, because there is only one USB device controller and it is used by the OTG functionality.

A USB transceiver is needed for each USB port used in the system. It converts between signaling appropriate for the OMAP5912 USB controllers and signaling appropriate for the USB wire. OMAP5912 USB functionality includes support for several types of USB transceivers. OMAP5912 provides one integrated USB transceiver suitable for a single USB host or USB device port, but it is not capable of acting as a USB OTG transceiver. OMAP5912 provides signaling to up to two external USB transceivers and/or two external USB-OTG-capable transceivers.

Several different types of external transceiver signaling are supported. Signaling between the OMAP5912 USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be via a 6-wire, 4-wire, or 3-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I^{2}C link. OTG transceivers can support the 6-wire, 4-wire, and/or 3-wire basic signaling but generally provide an interrupt output and an I^{2}C link for the additional control and status reporting required by OTG functionality.

4.4 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring OMAP5912 for that connectivity can be done using the steps listed below.

4.4.1 Select Desired USB Functionality

Choose the desired USB functionality. OMAP5912 can bring up to three USB ports to device pins. Options include:

❑ One USB device port

❑ One USB OTG port

❑ Up to three USB host ports

Choose a maximum of three of the above. It is not possible to configure both a USB device port and a USB OTG port, because the USB OTG port must use the single USB device controller. Similarly, a USB OTG port also makes use of one of the three USB host controller ports. Some possible configurations are:

❑ One USB device port

❑ One OTG port and one host port

❑ Two host ports and one USB device port

Some illegal combinations are:

❑ One USB device port and one OTG port (only one USB device controller is available; it can be used as part of the OTG functionality or it can be used as a standard USB device port, but not both simultaneously)

❑ One OTG port and three host ports (violates the maximum of three USB ports brought to OMAP5912 pins; also violates the limitation of three available host ports where one is used by OTG)

❑ Three host ports and one USB device port (violates the maximum of three USB ports brought to OMAP5912 pins)

4.4.2 Select How USB Functionality Is Multiplexed to OMAP5912 Pins

OMAP5912 provides pin interfaces for up to three USB ports. The USB functional ports can be mapped to the different OMAP5912 USB pin groups.

❑ Pin group 0 is associated with the OMAP5912 integrated USB transceiver, which is connected to OMAP5912 pins USB.DP and USB.DM. The transceiver can be used as a USB host or USB device port, but is not for use as a USB OTG port without additional external hardware. Pin group 0 is related to USB signal multiplexing port 0. When USB alternate pin group 2 is used, OMAP5912 pins USB.DP and USB.DM cannot be used for USB transceiver functionality.

The other two USB-related pin groups are associated with standard CMOS input and output pins. USB connectivity that uses these pin groups must use USB external transceivers. OMAP5912 top-level pin multiplexing allows a wide variety of functionality to be provided via these pins, so selection of these pins for use as USB functionality limits designer ability to use those pins for other (non-USB) functionality.

❑ Pin group 1 is associated with the following OMAP5912 pins:
  - MCBSP3.CLKX/USB1.TXEN
  - MCSI1.DOUT/USB1.TXD
  - RTC_WAKE_INT/USB1.SE0
  - MCSI1.DIN/USB1.RCV
  - MCSI1.SYNC/USB1.VP
  - MCSI1.BCLK/USB1.VM
  - CLK32K_OUT/USB1.SPEED
  - MPU_BOOT/USB1.SUSP

The last pin group has two different USB-related modes:

❑ Pin group 2 maps the signals associated with USB signal multiplexing port 2 to OMAP5912 pins, and is associated with these OMAP5912 pins:
  - MCSI2.DOUT/USB2.TXEN
  - UART2.TX/USB2.TXD
  - UART2.RTS/USB2.SE0
  - UART2.CTS/USB2.RCV
  - MCSI2.DIN/USB2.VP
  - UART2.RX/USB2.VM
  - MCSI2.SYNC/USB2.SPEED
  - MCSI2.CLK/USB2.SUSP

❑ Alternate pin group 2 maps signals associated with USB signal multiplexing port 0 to the same OMAP5912 pins used in pin group 2. The pin names for alternate pin group 2 are as follows:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

The selection procedure is as follows:

1) Given the required USB functionality, the USB pin group descriptions, and any non-USB usage of the pins in the pin group descriptions, choose a USB signal multiplexing mode from Table 73.

2) Find a value of HMC_MODE where all of the required USB functionality is available from the three USB multiplexing port columns.

3) Examine the OMAP5912 pin usage shown under the pin group and alternate pin group columns and find one HMC_MODE value that meets your needs for both USB and non-USB functionality.

4) Make careful note of the information in the row that determines how the top-level pin multiplexing and HMC_MODE value must be configured (see Table 73).

Table 73. USB Signal Multiplexing Modes

HMC_ MOD E	OTG Re- quired?	USB Multi- plex- ing Port 0	USB Multi- plex- ing Port 1	USB Multi- plex- ing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alter- nate Pin Group 2
0	No	USB device	Disable d	Disable d	USBdevice: See Note 1.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB device: See Note 2.
0 (cont)	Yes	USB OTG	Disable d	Disable d	USB OTG: See Note 3.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
1	No	USB host	USB host	USB host	USB host: See Note 5.	USB host: See Note 6.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB OTG	USB host	USB host	USB OTG: See Note 3.	USB host: See note 6.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
2	No	USB host	UART1	USB host	USB host: See Note 5.	UART1: See note 9.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
3	No	USB host	USB device	USB host	USB host: See Note 5.	USB device: See Note 10.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USBhost: See Note 8.
	Yes	USB host	USB OTG	USB host	USB OTG: See Note 3.	USB OTG: See Note 11.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 12.
4	No	USB device	USB host	USB host	USB device: See Note 1.	USB host: See Note 6.	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB device: See Note 2.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
	Yes	USB OTG	USB host	USB host	USB OTG: See Note 3.	USB host: See Note 6.	USB host: See Note 7.	N/A
4 (cont)	Yes	USB OTG	USB host	Disabled	Available for non-USB usage	USB host: See Note 6.	N/A	USB OTG: See Note 4.
5	No	USB host	Disabled	USB host	USB host: See Note 5.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB OTG	Disabled	USB host	USB OTG: See Note 3.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
6	No	USB device	Disabled	USB host	USB device: See Note 1.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB device: See Note 2.
	Yes	USB OTG	Disabled	USB host	USB OTG: See Note 3.	Available for non-USB usage	USB host: See Note 7.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
7	No	USB host	Disabled	Disabled	USB host: See Note 5.	See Note 13.		N/A
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
9	No	USB host	USB host	USB host with TLL	USB host: See Note 5.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
9 (cont)	Yes	USB OTG	USB host	USB host with TLL	USB OTG: See Note 3.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB OTG: See Note 4.
10	No	USB host	UART1	USB host with TLL	USB host: See Note 5.	UART1: See Note 9.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
11	No	USB host	USB device	USB host with TLL	USB host: See Note 5.	USB device: See Note 10.	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB host	USB OTG	USB host with TLL	USB host: See Note 5.	USB OTG: See Note 11.	USB host + TLL: See Note 14.	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
12	No	USB device	USB host	USB host with TLL	USB device: See Note 1.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
					Available for non-USB usage		N/A	USB device: See Note 2.
	Yes	USB OTG	USB host	USB host with TLL	USB OTG: See Note 3.	USB host: See Note 6.	USB host + TLL: See Note 14.	N/A
12 (cont)	Yes	USB OTG	USB host	USB host with TLL	Available for non-USB usage	USB host: See Note 6.	N/A	USB OTG: See Note 4.
13	No	USB host	Disabled	USB device with TLL	USB host: See Note 5.	Available for non-USB usage	USB device + TLL: See Note 15.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
14	No	USB device	Disabled	USB host with TLL	USB device: See Note 1.	Available for non-USB usage	USB host + TLL: See Note 14.	N/A
				Disabled	Available for non-USB usage		N/A	USB device: See Note 2.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
15	No	USB host	USB host	USB device with TLL	USB host: See Note 5.	USB host: See Note 6.	USB device + TLL: See Note 15.	N/A
				Disabled	Available for non-USB usage		N/A	USB host: See Note 8.
16	No	USB host	Disabled	Disabled	USB host: See Note 5.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB OTG	Disabled	Disabled	USB OTG: See Note 3.	Available for non-USB usage	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.
17	No	USB host	USB host	Disabled	USB host: See Note 5.	USB host: See Note 6.	Available for non-USB usage	N/A
17 (cont)	No	USB host	USB host	Disabled	Available for non-USB usage	USB host: See Note 6.	N/A	USB host: See Note 8.
	Yes	USB OTG	USB host	Disabled	USB OTG: See Note 3.	USB host: See Note 6.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.
18	No	USB host	UART1	Disabled	USB host: See Note 5.	UART1: See Note 9.	Available for non-USB usage	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
					Available for non-USB usage		N/A	USB host: See Note 8.
19	No	USB host	USB device	Disabled	USB host: See Note 5.	USB device: See Note 10.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB host	USB OTG	Disabled	USB host: See Note 5.	USB OTG: See Note 11.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB host: See Note 8.
20	No	USB device	USB host	Disabled	USB device: See Note 1.	USB host: See Note 6.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB device: See Note 2.
	Yes	USB OTG	USB host	Disabled	USB OTG: See Note 3.	USB host: See Note 6.	Available for non-USB usage	N/A
					Available for non-USB usage		N/A	USB OTG: See Note 4.
21	No	USB host	USB host port 2 internally connected to USB device via TLL. USB host port 1 disabled.		USB host: See Note 5.	Available for non-USB usage	Available for non-USB usage	N/A

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
22	No	Disabled	Disabled	Disabled	Available for non-USB usage	Available for non-USB usage	Available for non-USB usage	N/A
23	No	USB host	USB host	USB host + TLL (TXDP/ TXDM)	USB host: See Note 5.	USB host: See Note 6.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
	Yes	USB host	USB OTG	USB host + TLL (TXDP/ TXDM)	USB host: See Note 5.	USB OTG: See Note 11.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
24	No	USB host	UART1	USB host + TLL (TXDP/ TXDM)	USB host: See Note 5.	UART1: See Note 9.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
25	No	USB host	USB device	USB host + TXDP/ TXDM	USB host: See Note 5.	USB device: See Note 10.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.

Table 73. USB Signal Multiplexing Modes (Continued)

HMC_MODE	OTG Required?	USB Multiplexing Port 0	USB Multiplexing Port 1	USB Multiplexing Port 2	USB Pin Group 0	USB Pin Group 1	USB Pin Group 2	USB Alternate Pin Group 2
25 (cont)	Yes	USB host	USB OTG	USB host + TXDP/ TXDM	USB host: See Note 5.	USB OTG: See Note 11.	USB host + TLL: See Note 16.	N/A
				N/A	Available for non-USB usage		N/A	USB host: See Note 8.
Other HMC_MODE values	No	Disabled	Disabled	Disabled	Available for non-USB usage	Available for non-USB usage	Available for non-USB usage	N/A

Notes:
1) Figure 76 shows typical connectivity for USB pin group 0 acting as USB device.
2) Figure 80, Figure 81, or Figure 82 show connectivity options for USB alternate pin group 2 acting as USB device.
3) Figure Figure 59 shows typical connectivity for USB pin group 0 acting as USB OTG.
4) Figure 63, Figure 64, or Figure 65 show connectivity options for USB alternate pin group 2 acting as USB OTG.
5) Figure 66 shows typical connectivity for USB pin group 0 acting as USB host.
6) Figure 67, Figure 68, or Figure 69 show connectivity options for USB pin group 1 acting as USB host.
7) Figure 70, Figure 71, or Figure 72 show connectivity options for USB pin group 2 acting as USB host.
8) Figure 73, Figure 74, or Figure 75show connectivity options for USB alternate pin group 2 acting as USB host.
9) Figure 89 shows typical connectivity for USB pin group 1 acting as UART1.
10) Figure 77, Figure 78, or Figure 79 show connectivity options for USB pin group 1 acting as USB device.
11) Figure 60, Figure 61, or Figure 62 show connectivity options for USB pin group 1 acting as USB OTG.
12) Figure 63, Figure 64, or Figure 65 show connectivity options for USB alternate pin group 2 acting as USB OTG.
13) Figure 88 shows typical connectivity for USB pin group 1 to USB pin group 2 wrap mode.
14) See connectivity when OMAP5912 acts as the USB host on a transceiver-less link connection to an on-board USB device (Figure 85) using USB TXD/USB TXSE0 signaling.
15) See connectivity when OMAP5912 acts as the USB host on a transceiver-less link connection to an on-board USB device using USB TXD/USB TXSE0 signaling (Figure 86).
16) See connectivity when OMAP5912 acts as the USB host on a transceiver-less link connection to an on-board USB device using USB TXD/USB TXSE0 signaling (Figure 87).

USB controller ports that are not specifically mentioned for a row of Table 73 are held in a disabled state. When a USB host controller port is disabled, the port appears to the controller as a disconnected port. When the USB device controller is not available to a pin group and OTG is not enabled, the USB device controller sees pin signaling equivalent to USB_RESET.

When an HMC_MODE is selected that can connect a USB multiplexing port to OMAP5912 device pins, but all of the pins associated with that pin group are set for non-USB functionality, the associated USB multiplexing port sees 6-/8-wire transceiver signaling, which implies that both D+ and D- are low. If that pin group is associated with a host controller port, that host controller port appears as a disconnected USB link. If that pin group is associated with the device controller, the device controller sees USB reset signaling.

Functionality is undefined when top-level pin multiplexing selects non-USB signalling of one or more of the pins that are required by the selected HMC_MODE and transceiver signalling type. For example, if an HMC_MODE value selects USB pin group 2 as a USB host connection and a 4-wire transceiver interface is selected, but pin UART2.CTS/USB2.RCV top-level pin multiplexing selects signal UART2.CTS, the associated USB host controller port behavior is undefined.

Select USB and/or USB OTG Transceiver Type

USB connectivity on OMAP5912 USB pin group 1, USB pin group 2, and USB alternate pin group 2 require external components, except for pin group 2 when it is configured for a TLL mode. Several different types of external transceiver signaling are supported. Signaling between the OMAP5912 USB controller and the external USB transceiver for monitoring and controlling the differential USB signal can be via a 6-wire, 4-wire, or 3-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I^{2}C link. OTG transceivers can support the 6-wire, 4-wire, and/or 3-wire basic signaling, but generally provide an interrupt output and an I^{2}C link for the additional control and status reporting required by OTG functionality.

The figures referenced in the USB Pin Group 0, USB Pin Group 1, USB Pin Group 2, and Alternate Pin Group 2 columns of Table 73 show external connectivity options for each of the possible selections. Carefully examine the options for the HMC_MODE you have identified.

Consider which OMAP5912 pins are required in each diagram and determine if any particular choice (3-wire, 4-wire, or 6-wire) is advantageous in terms of other non-USB functionality that is available for each pin group. Choose a transceiver type.

A USB OTG transceiver can generally be used in place of a USB transceiver; using an OTG transceiver with an I^{2}C interface can reduce the number of OMAP5912 pins that are required. This can assist in making non-USB functionality available on the USB pin groups.

Determine Proper Top-level Multiplexing Settings

Top-level pin multiplexing settings are determined primarily by which pins perform USB functions and the type of transceiver connected to the pins.

Pin multiplexing is controlled on a pin-by-pin basis via the top-level pin multiplexing. Each pin, referenced by default pin name, must be configured to provide the correct functional signal. To configure a pin for its USB functionality, determine the default pin name to be configured. For example, the default pin name for MCSI2.SYNC/USB0.SPEED is MCSI2.SYNC, and the default pin name for MCBSP3.CLKX/USB1.TXEN is MCBSP3.CLKX. See Table 74 to determine the name of the register and the name of the bit field in that register that controls multiplexing for the pin. Choose the desired USB signal name to be used on that pin, and use the associated value under pin multiplexing configuration value as the value to be programmed into the specified bit field in the specified pin multiplexing control register.

For example, to configure top-level pin multiplexing for UART2.CTS/USB0.RCV, locate the UART2.CTS in the default pin name column of Table 74. The register bit field that configures the top-level pin multiplexing for this pin is FUNC_MUX_CTRL_C.CONF_CTS2_R. Find USB0.RCV in the USB Signal Name column of the UART2.CTS row, and find that the Pin Multiplexing Configuration Value for USB2.RCV is 5. This means that writing a 5 to FUNC_MUX_CTRL_C.CONF_CTS2_R configures the UART2.CTS pin to act as USB0.RCV.

Table 74. Top-Level Pin Multiplexing Configuration for OMAP5912 USB-Related Pins

Default Pin Name	Pin Multiplexing Control Register	Bit Field	USB Signal Name	Pin Multiplexing Configuration Value
USB.DP	USB_TRANSCEIVER_CTRL	CONF_USB_PORT0_R	USB.DP	0
			USB.PUEN	7
USB.DM	USB_TRANSCEIVER_CTRL	CONF_USB_PORT0_R	USB.DM	0
			HIGH IMPEDANCE	7
USB.PUEN	FUNC_MUX_CTRL_D	CONF_USB_PUEN_R	USB.PUEN	0
			USB.CLK0	1
			USB.PUDIS	3
GPIO0	FUNC_MUX_CTRL_7	CONF_GPIO_0_R	USB.VBUS	2

Table 74. Top-Level Pin Multiplexing Configuration for OMAP5912 USB-Related Pins (Continued)

Default Pin Name	Pin Multiplexing Control Register	Bit Field	USB Signal Name	Pin Multiplexing Configuration Value
MCBSP3. CLKX	FUNC_MUX_CTRL_9	CONF_MCBSP3_ CLK_R	USB1.TXEN	2
MCSI1.DOUT	FUNC_MUX_CTRL_9	CONF_MCSI1_ DOUT_R	USB1.TXD	1
RTC_WAKE_ INT	FUNC_MUX_CTRL_9	CONF_WAKEUP_INT_ R	USB1_TXSE0	4
MCSI1.DIN	FUNC_MUX_CTRL_A	CONF_MCSI1_DIN_R	USB1.RCV	1
MCSI1.SYNC	FUNC_MUX_CTRL_A	CONF_MCSI1_SYNC_R	USB1.VP	2
MCSI1.BCLK	FUNC_MUX_CTRL_A	CONF_MCSI1_BCLK_R	USB1.VM	2
CLK32K_OUT	FUNC_MUX_CTRL_A	CONF_CLK32K_OUT_R	USB1.SPEED	5
MPU_BOOT	FUNC_MUX_CTRL_8	CONF_ARM_BOOT_R	USB1.SUSP	2
MCSI2.DOUT	FUNC_MUX_CTRL_B	CONF_MCSI2_ DOUT_R	USB2.TXEN	1
			USB0.TXEN	5
UART2.TX	FUNC_MUX_CTRL_C	CONF_TX2_R	USB2.TXD	2
			USB0.TXD	5
UART2.RTS	FUNC_MUX_CTRL_C	CONF_RTS2_R	USB2_TXSE0	2
			USB0_TXSE0	5
UART2.CTS	FUNC_MUX_CTRL_C	CONF_CTS2_R	USB2.RCV	1
			USB0.RCV	5
MCSI2.DIN	FUNC_MUX_CTRL_B	CONF_MCSI2_DIN_R	USB2.VP	1
			USB0.VP	5
UART2.RX	FUNC_MUX_CTRL_C	CONF_RX2_R	USB2.VM	1
			USB0.VM	5
MCSI2.SYNC	FUNC_MUX_CTRL_B	CONF_MCSI2_SYNC_R	USB2.SPEED	2
			USB0.SPEED	5

Table 74. Top-Level Pin Multiplexing Configuration for OMAP5912 USB-Related Pins (Continued)

Default Pin Name	Pin Multiplexing Control Register	Bit Field	USB Signal Name	Pin Multiplexing Configuration Value
MCSI2.CLK	FUNC_MUX_CTRL_B	CONF_MCSI2_CLK_R	USB2.SUSP	1
			USB0.SUSP	5

When a 3-wire or 4-wire (bidirectional) transceiver is connected to USB pin group 1, USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R must be 0. When a 6-wire or 8-wire (unidirectional) transceiver is connected to USB pin group 1, USB_TRANSCEIVER_CTRL.CONF_USB1_UNI_R must be 1.

When a 3-wire or 4-wire (bidirectional) transceiver is connected to USB pin group 2 or USB alternate pin group 2, USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R must be 0. When a 6-wire or 8-wire (unidirectional) transceiver is connected to USB pin group 2 or USB alternate pin group 2, USB_TRANSCEIVER_CTRL.CONF_USB2_UNI_R must be 1.

Determine OTG Module Control Register Settings

The OTG module provides registers that control several aspects of the USB pin signaling. These registers must be properly configured to allow proper USB operation, even when the OTG feature is not being used.

OTG_SYSCON_1.USB0_TRX_MODE must be 0 or 3 to allow proper operation of the integrated USB transceiver on USB pin group 0. For cases where USB alternate pin group 2 is used, OTG_SYSCON_1.USB0_TRX_MODE must be set to match the type of transceiver used on USB alternate pin group 3 (3 for a 6-wire transceiver, 1 for a 4-wire transceiver, or 2 for 3-wire transceiver).

OTG_SYSCON_1.USB1_TRX_MODE must be set to match the type of transceiver connected to USB pin group 1. When USB pin group 1 is not connected to a transceiver, those pins must be configured for non-USB operation and OTG_SYSCON_1.USB1_TRX_MODE must be set to 0.

OTG_SYSCON_1.USB2_TRX_MODE must be set to match the type of transceiver connected to USB pin group 2. When alternate USB pin group 2 is used, OTG_SYSCON_1.USB2_TRX_MODE has no effect and can be set to 0.

OTG_SYSCON_2.USBx_SYNCHRO must be set to 1 for proper transceiver operation.

OTG_SYSCON_2.OTG_PADEN must be set to 0 whenever OTG functionality is required. Writing a 1 to this bit prevents proper operation of the OTG_CTRL register. When OTG_PADEN is 0 and OTG_EN = 0, the GPIO0/USB.VBUS input is not provided to the USB device controller. In this case, software monitors GPIO0 and updates OTG_CTRL.BSESSVLD with the value from GPIO0.

When OTG_PADEN is 1 and GPIO_0/USB.VBUS is configured for USB.VBUS functionality, GPIO0/USB.VBUS propagates to the USB device controller without software intervention, but the OTG_CTRL register bits ASESSVLD, BSESSEND, BSESSVLD, VBUSVLD, and ID are read-only and are always 0. Because of this, it is impossible to implement USB-OTG functionality when OTG_PADEN is 1.

OTG_SYSCON_2.HMC_PADEN determines whether values for UHOST_EN, HMC_MODE, TLL_ATTACH, and TLL_SPEED are provided by bits in OTG_SYSCON_2 or by bits in MOD_CONF_CTRL_0, as shown in Table 75. For software compatibility with future devices, HMC_PADEN must be set to 0.

Table 75. UHOST_EN, HMC_MODE, TLL_ATTACH, TLL_SPEED Source Selection

Register	**Field**	**Value**	
OTG_SYSCON2	HMC_PADEN	0	1
MOD_CONF_CTRL_0	UHOST_EN	Don’t care	Valid
	HMC_MODE		
	HMC_TLLATTACH		
	HMC_TLLSPEED		
OTG_SYSCON_2	CONF_MOD_USB_HOST_HHC_UHOST_EN_R	Valid	Don’t care
	CONF_MOD_USB_HOST_HMC_MODE_R		
	CONF_MOD_USB_HOST_HMC_TLL_ATTACH_R		
	CONF_MOD_USB_HOST_HMC_TLL_SPEED_R		

Table 75. UHOST_EN, HMC_MODE, TLL_ATTACH, TLL_SPEED Source Selection (Continued)

Register	Field	Value	
Values used	UHOST_EN	Values from OTG_ SYSCON_2	Values from MOD_CONF_ CTRL_0
	HMC_MODE		
	TLL_ATTACH		
	TLL_SPEED		

If OTG_SYSCON_2.HMC_PADEN is 0, OTG_SYSCON_2.HMC_MODE must be programmed with the HMC_MODE value chosen from Table 73. If OTG_SYSCON_2.HMC_PADEN is 1, the chosen HMC_MODE value must be written to MOD_CONF_CTRL_0.CONF_MOD_USB_HOST_HMC_ MODE_R.

4.5 Transceiver Signaling Types

USB Transceiver Unidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP5912 and is used in unidirectional DAT/SE0 signaling mode, the signaling shown in Table 76 is used.

Table 76. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling

Logical Signal Name(s)	OMAP5912 Pin Direction	Transceiver Pin Direction	Description				
$\overline{\text{OE}}$	Output	Input	When low, USB transceiver drives D+ and D-.				
DAT and SE0	Output	Input	Controls the values output by the USB transceiver on D+ and D- when $\overline{\text{OE}}$ is low. Ignored when $\overline{\text{OE}}$ is high.				
			OEn	DAT	SE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
			1	X	X	Undriven	Undriven

Table 76. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling (Continued)

<table>
<tr><th>Logical Signal Name(s)</th><th>OMAP5912 Pin Direction</th><th>Transceiver Pin Direction</th><th colspan="3">Description</th></tr>
<tr><td rowspan="6">RCV</td><td rowspan="6">Input</td><td rowspan="6">Output</td><td colspan="3">Output from transceiver differential receiver.</td></tr>
<tr><td>D+</td><td>D-</td><td>RCV</td></tr>
<tr><td>0</td><td>0</td><td>X</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>X</td></tr>
<tr><td rowspan="4">RCVVP</td><td rowspan="4">Input</td><td rowspan="4">Output</td><td colspan="3">Output from transceiver single-ended D+ signal receiver .</td></tr>
<tr><td>D+</td><td colspan="2">RCVVP</td></tr>
<tr><td>0</td><td colspan="2">0</td></tr>
<tr><td>1</td><td colspan="2">1</td></tr>
<tr><td rowspan="4">RCVVM</td><td rowspan="4">Input</td><td rowspan="4">Output</td><td colspan="3">Output from transceiver single-ended D- signal receiver.</td></tr>
<tr><td>D-</td><td colspan="2">RCVVM</td></tr>
<tr><td>0</td><td colspan="2">0</td></tr>
<tr><td>1</td><td colspan="2">1</td></tr>
<tr><td>SPEED</td><td>Output</td><td>Input</td><td colspan="3">Determines transceiver D+, D- output characteristics.
0 = Low speed. 1 = Full speed. Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.</td></tr>
<tr><td>SUSPEND</td><td>Output</td><td>Input</td><td colspan="3">Controls transceiver powerdown modes.
0 = Active mode.
1 = Low-power mode
Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.</td></tr>
</table>

OMAP5912 does not support unidirectional transceivers that implement VPO/VMO signaling.

USB Transceiver 3-Wire Bidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP5912 and is used in 3-wire bidirectional TXDAT/TXSE0 signaling mode, the signaling shown in Table 77 is used.

Table 77. Signaling Between USB Controller and 3-Wire Bidirectional USB Transceiver Using TXDAT/TXSE0 Signaling

Logical Signal Name	**OMAP5912 Pin Direction**	**Transceiver Pin Direction**	**Description**				
$\overline{\text{OE}}$	Output	Input	When low, USB transceiver drives D+ and D-.				
TXDAT/ RCVDAT and TXSE0/ RCVSE0			When $\overline{\text{OE}}$ is low, OMAP5912 drives TXDAT and TXSE0 and the transceiver drives D+ and D- based on the values of TXDAT and TXSE0.				
	Output	Input	OEn	TXDAT	TXSE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
	Input	Output	OEn	D+	D-	RCVDAT	RCVSE0
			1	0	0	0	1
				0	1	0	0
				1	0	1	0
				1	1	Undefined	Undefined
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed 1 = Full speed Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.				
SUSPEND	Output	Input	Controls transceiver powerdown modes. 0 = Active mode 1 = Low-power mode Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.				

OMAP5912 does not support 3-wire bidirectional signaling using VP/VM signals.

USB Transceiver 4-Wire Bidirectional Signaling

When a USB or USB OTG transceiver is connected to OMAP5912 and is used in 4-wire bidirectional DAT/SE0 signaling mode, the signaling shown in Table 78 is used.

Table 78. Signaling Between USB Controller and 4-Wire Unidirectional USB Transceiver Using VP/VM Signaling

Logical Signal Name	OMAP5912 Pin Direction	Transceiver Pin Direction	Description		
$\overline{\text{OE}}$	Output	Input	When low, USB transceiver drives D+ and D-.		
TXVM/ RCVVM			Value driven to or received from D-		
	Output	Input	OEn	TXVM	D-
			0	0	0
			0	1	1
	Input	Output	OEn	D-	RCVVM
			1	0	0
			1	1	1
TXVP/ RCVVP			Value driven to or received from D+		
	Output	Input	OEn	TXVP	D+
			0	0	0
			0	1	1
	Input	Output	OEn	D+	RCVVP
			1	0	0
			1	1	1
RCV	Input	Output	Output from transceiver single-ended D- signal receiver.		
			D+	D-	RCV
			0	0	X
			0	1	0
			1	0	1
			1	1	X

Table 78. Signaling Between USB Controller and 4-Wire Unidirectional USB Transceiver Using VP/VM Signaling (Continued)

Logical Signal Name	OMAP5912 Pin Direction	Transceiver Pin Direction	Description
SPEED	Output	Input	Determines transceiver D+, D- output characteristics. 0 = Low speed 1 = Full speed Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.
SUSPEND	Output	Input	Controls transceiver power-down modes. 0 = Active mode 1 = Low-power mode Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.

OMAP5912 does not support 4-wire bidirectional signaling using DAT/SE0 signals.

4.6 USB OTG External Connectivity

To provide USB OTG connectivity, a dual-role device system that provides a USB OTG controller must implement certain features. These features include a USB mini-AB receptacle, VBUS monitoring and control, transient suppression, ID monitoring, controllable D+ and D- series, pullup and pulldown resistors, and a D+ and D- driver/receiver. USB OTG transceivers that implement many of these features are commercially available.

Some USB OTG transceivers implement a 6-wire connection to the OTG controller, whereas others require only 4 wires or 3 wires. The OMAP5912 device must be properly configured to support the signaling required by the attached transceiver.

Many OTG transceiver control and status functions are provided using I^{2}C registers. This reduces the number of connections between the OTG transceiver and the OTG controller. OTG transceiver I^{2}C registers control a variety of functions, including D+ and D- pullups, D+ and D- pulldowns, and the VBUS output. OTG transceiver I^{2}C registers provide transceiver status including status of the VBUS and ID pins and interrupt conditions.

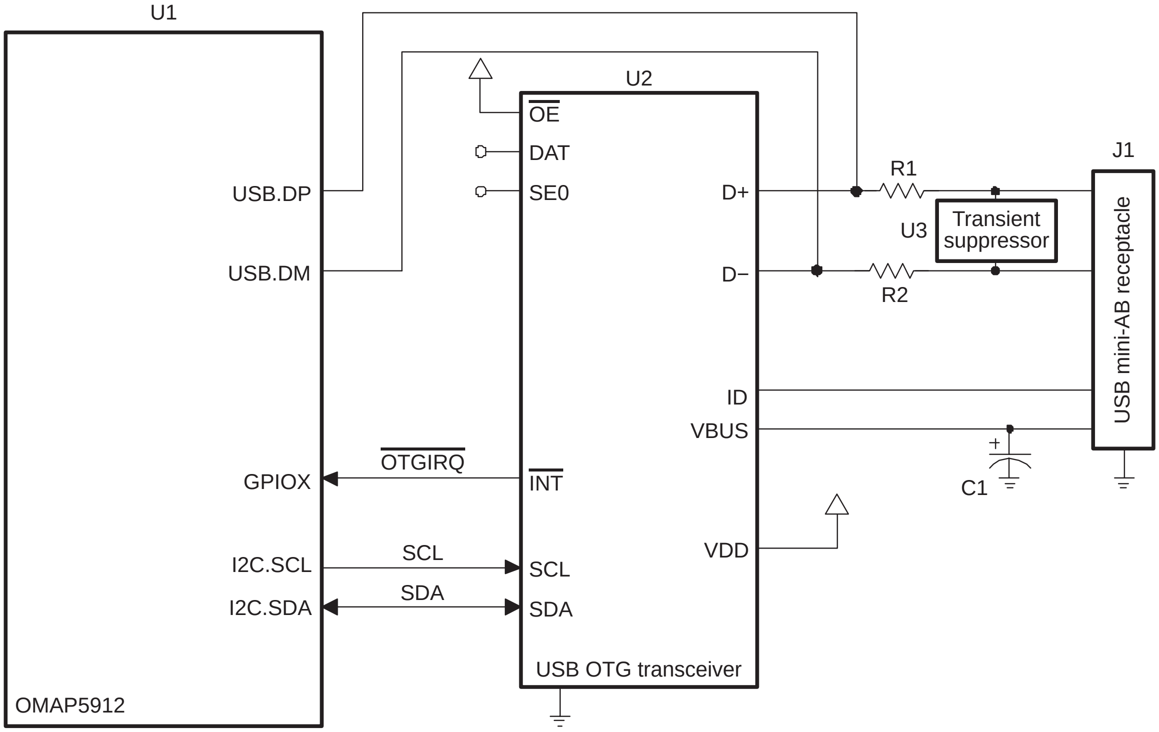
The typical OTG transceiver provides an interrupt output to the processor. This interrupt informs the processor when VBUS state changes, when ID state changes, or when any number of other transceiver-dependent conditions occur. When OTG transceiver interrupts occur, system software must query the OTG transceiver status using I^2C and update the OTG controller registers as appropriate.

Some system applications require that the USB Mini_AB receptacle can be connected to downstream non-OTG USB devices, such as mice, keyboards, or printers. In such cases, the USB OTG specification allows use of a converter cable with a mini-A plug on one end and a standard type-A receptacle on the other end. The USB OTG transceiver, which is capable of supplying the amount of VBUS current required by the USB OTG specification, might not be able to meet the VBUS current supply requirement of the USB specification. In such cases, it is necessary to provide some alternate VBUS source that meets the USB specification requirements.

Pin Group 0 OTG

The integrated USB transceiver on OMAP5912 USB pin group 0 does not provide the functionality specific to OTG transceivers. To implement an OTG dual-role device using OMAP5912 USB pin group 0, an external OTG transceiver is necessary to provide that additional functionality. In such a system, the external OTG transceiver provides the VBUS detect and drive, D+ and D- pullup and pulldown, and ID detection features, while the OMAP5912 integrated USB transceiver provides the D+ and D- signal drive and receive functions (see Figure 59).

Figure 59. OTG on USB Pin Group 0



R1, R2	27 Ohm 5%
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP5912
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins USB.DP and USB.DM and selection of an HMC_MODE value that routes the OTG controller port to pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

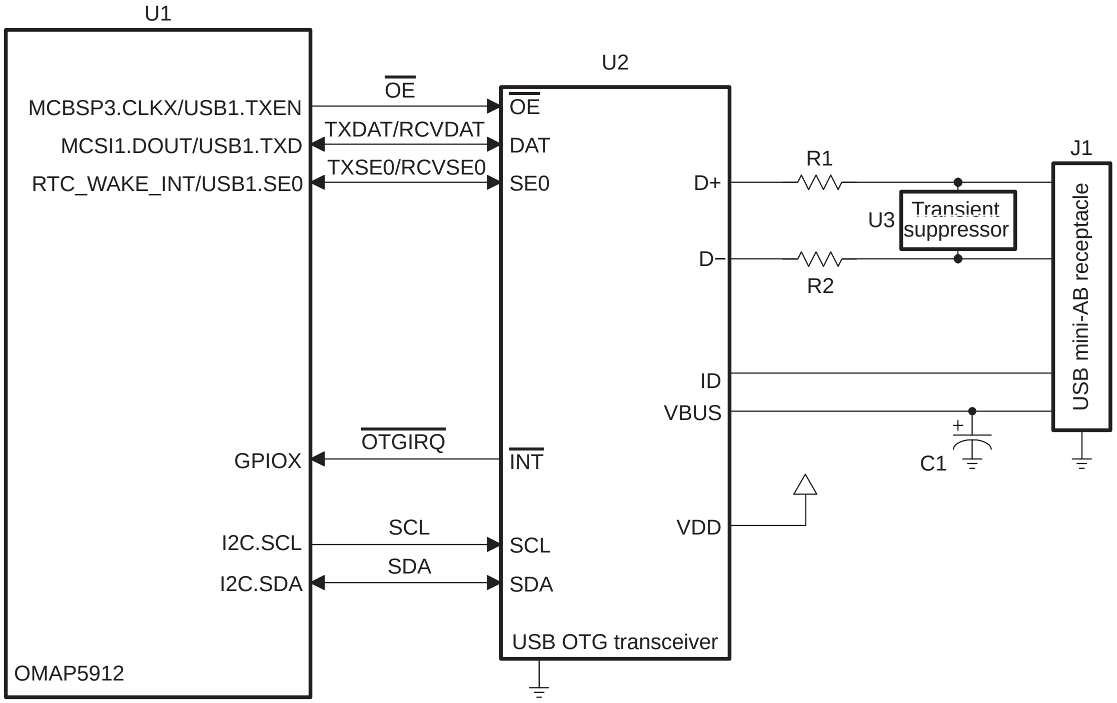
System software is responsible for transferring control signals from the OMAP5912 OTG controller to the OTG transceiver and status information from the OTG transceiver to the OMAP5912 OTG controller registers.

When acting as an OTG default-A dual-role device, system software must make use of the OTG transceiver for autoconnect mode. System software must also configure the OTG transceiver to suspend mode whenever OMAP5912 acts as a default-A device and is placing the OTG link into suspend so that the remote-B device can issue an HNP.

If you want to use the external OTG transceiver ability to provide I^2C signaling to the transceiver D+ and D- pins, it is necessary to control the OTG transceiver OEn input using an OMAP5912 GPIO pin. It is also necessary to set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the I^2C signals have no effect on the OMAP5912 USB controllers.

OTG on Pin Group 1 Using 3-Wire OTG Transceiver

Figure 60. OTG on USB Pin Group 1 Using 3-Wire OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP5912
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

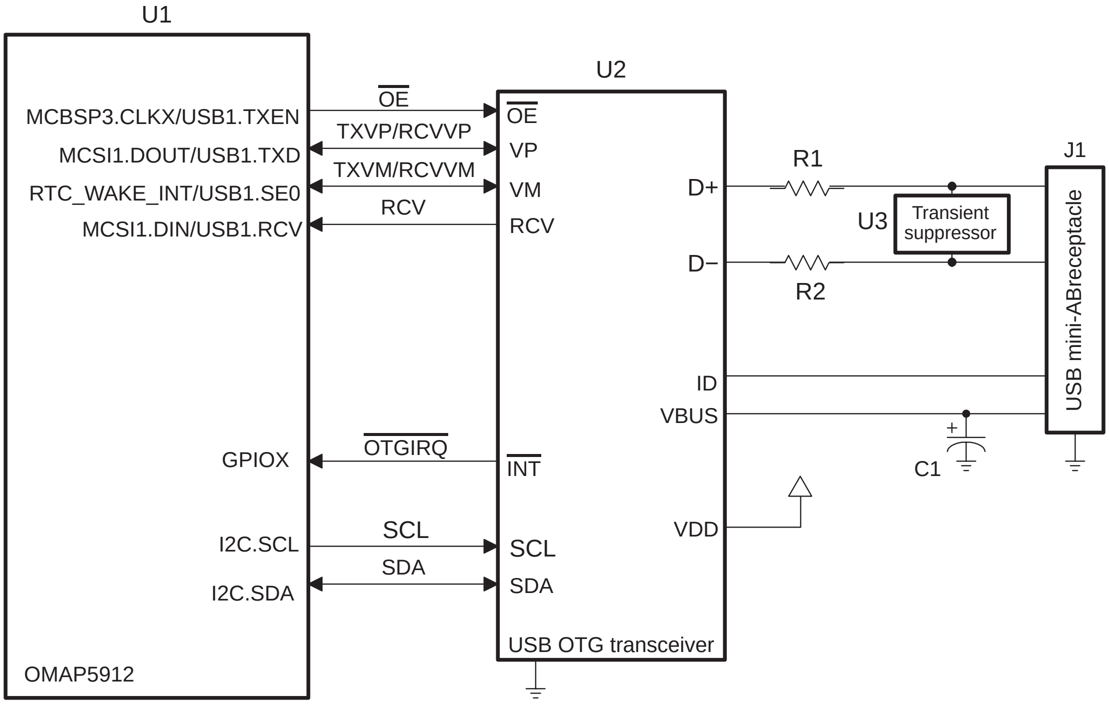
- ❑ MCBSP3.CLKX/USB1.TXEN
- ❑ MCSI1.DOUT/USB1.TXD
- ❑ RTC_WAKE_INT/USB1.SE0

See Table 74.

HMC_MODE must be set to a value that routes OTG functionality to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

OTG on Pin Group 1 Using 4-Wire OTG Transceiver

Figure 61. OTG on USB Pin Group 1 Using 4-Wire OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP5912
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

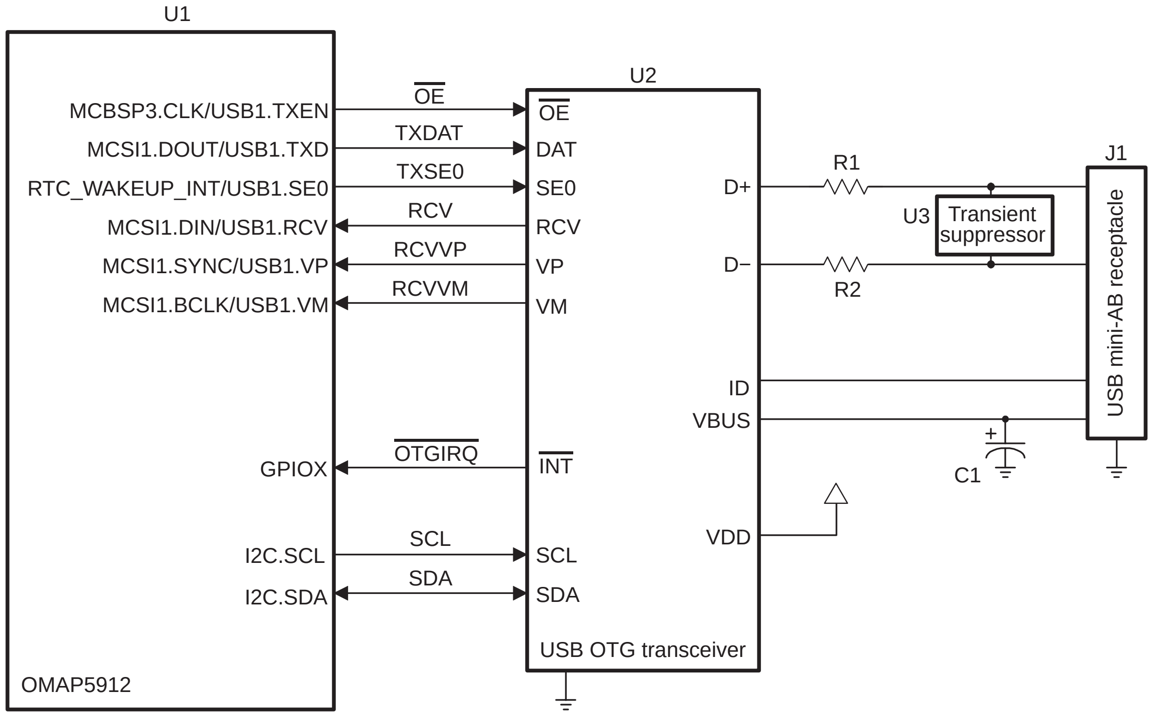
- ❑ MCBSP3.CLKX/USB1.TXEN
- ❑ MCSI1.DOUT/USB1.TXD
- ❑ RTC_WAKE_INT/USB1.SE0
- ❑ MCSI1.DIN/USB1.RCV

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

OTG on Pin Group 1 OTG Using 6-Wire OTG Transceiver

Figure 62. OTG on USB Pin Group 1 Using 6-Wire OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP5912
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

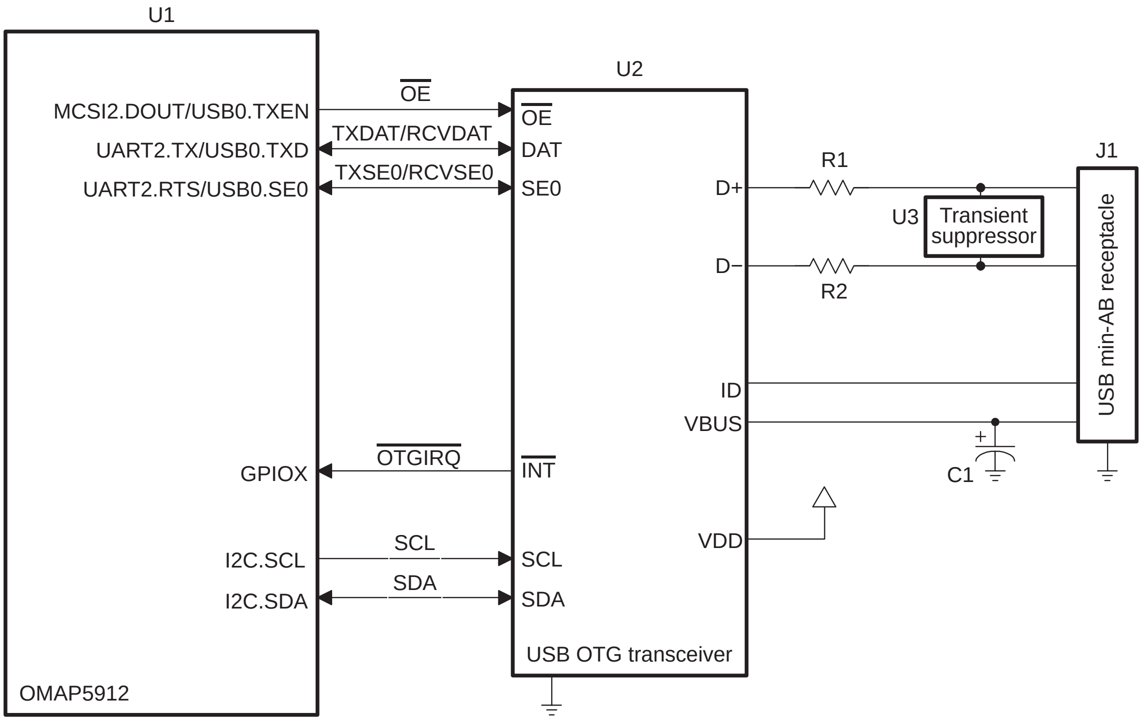
❏ MCBSP3.CLKX/USB1.TXEN
❏ MCSI1.DOUT/USB1.TXD
❏ RTC_WAKE_INT/USB1.SE0
❏ MCSI1.SYNC/USB1.VP
❏ MCSI1.BCLK/USB1.VM
❏ MCSI1.DIN/USB1.RCV

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB Pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 6-wire USB transceiver.

OTG on Alternate Pin Group 2 Using 3-Wire OTG Transceiver

Figure 63. OTG on USB Alternate Pin Group 2 Using 3-Wire OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP5912
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

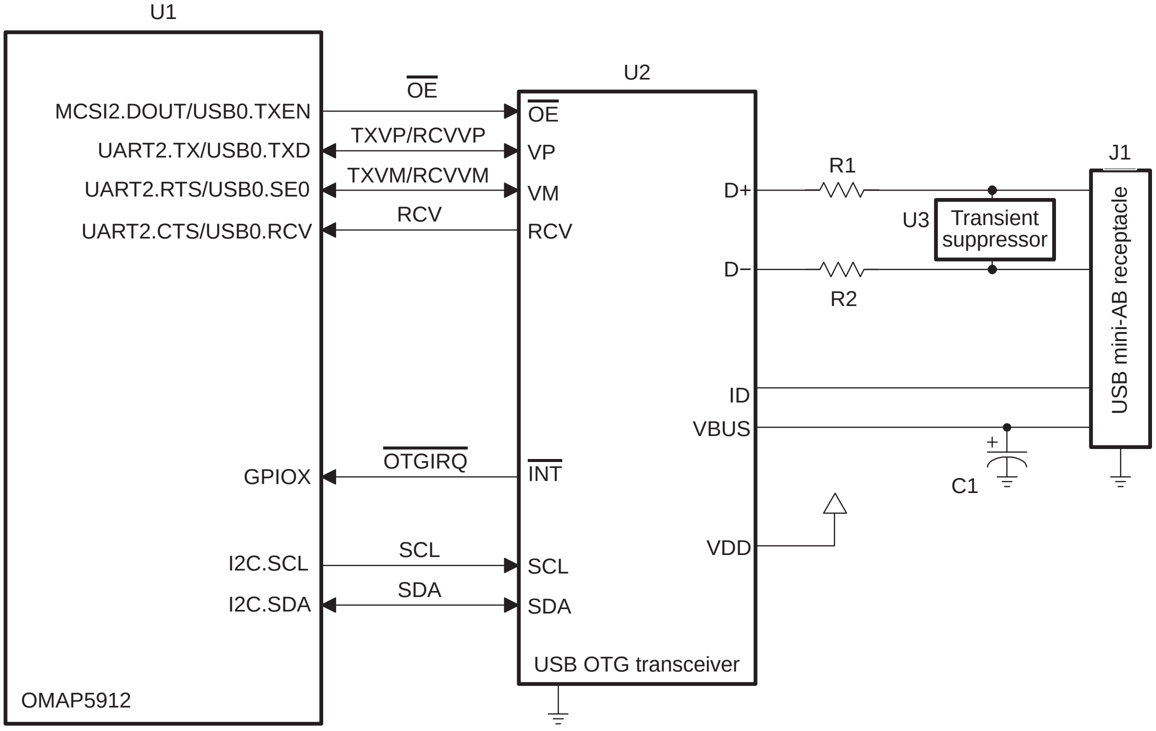
❑ MCSI2.DOUT/USB0.TXEN
❑ UART2.TX/USB0.TXD
❑ UART2.RTS/USB0.SE0

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB alternate pin group 2 (see Table 73). OTG_SYSCON_1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

OTG on Alternate Pin Group 2 Using 4-Wire OTG Transceiver

Figure 64. OTG on USB Alternate Pin Group 2 Using 4-Wire OTG Transceiver



R1, R2 Value depends on transceiver
C1 VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1 OMAP5912
U2 USB OTG transceiver
U3 Transient suppressor, like SN65220, SN65240, or SN75240
J1 USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

❑ MCSI2.DOUT/USB0.TXEN
❑ UART2.TX/USB0.TXD
❑ UART2.RTS/USB0.SE0
❑ UART2.CTS/USB0.RCV

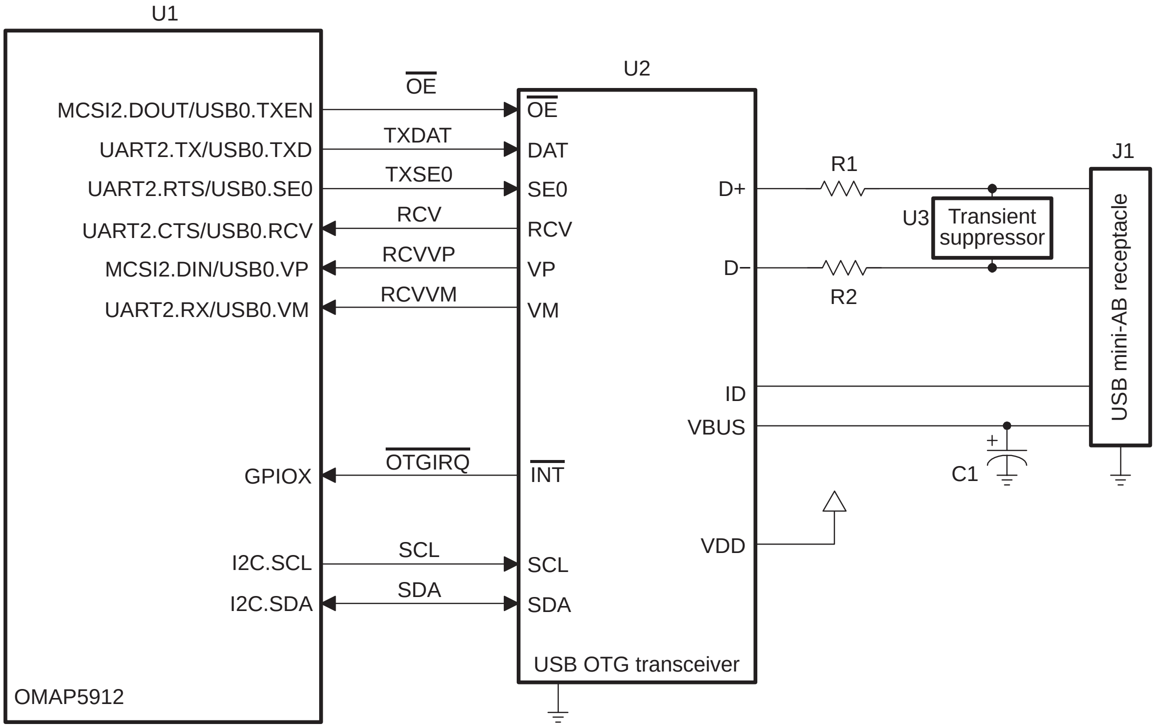
See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB alternate pin group 2 (see Table 73).

OTG_SYSCON_1.USB2_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

OTG on Alternate Pin Group 2 Using 6-Wire OTG Transceiver

Figure 65. OTG on USB Alternate Pin Group 2 Using 6-Wire OTG Transceiver



R1, R2	Value depends on transceiver
C1	VBUS capacitance must meet OTG spec C_{DRD_VBUS}
U1	OMAP5912
U2	USB OTG transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
J1	USB mini-AB receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.DIN/USB0.VP
- UART2.RX/USB0.VM

See Table 74.

HMC_MODE must be set to a value that provides OTG functionality on USB Alternate Pin group 2 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 6-wire USB transceiver.

4.7 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features. These features include a type-A receptacle, power on the VBUS signal (can be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver.

Because OMAP5912 does not provide a pin that connects to the USB host controller port power control registers, some other mechanism must be used if VBUS switching is required. Similarly, OMAP5912 does not provide any pins that connect to the USB host controller overcurrent status bits, so some other mechanism must be used if overcurrent sensing is required.

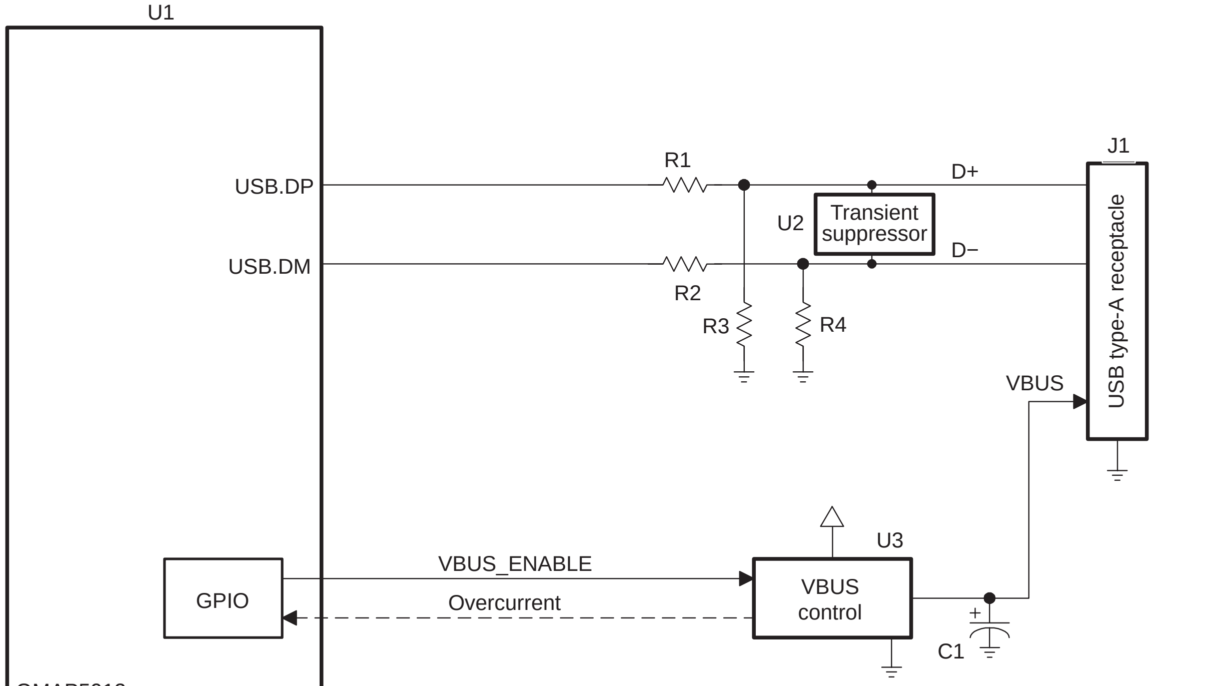
Some USB transceivers implement a 6-wire connection to the host controller, whereas others require only 4 wires or 3 wires. The OMAP5912 device must be properly configured to support the signaling required by the attached transceiver.

It is possible to use OTG transceiver connectivity to provide a USB host-only port. The OTG transceiver meets all of the requirements of a USB upstream transceiver with the exception of VBUS current sourcing. OTG transceiver VBUS current is typically limited to a maximum value that is lower than the minimum required by the USB specification for a Type-A receptacle. Systems that use an OTG transceiver to control a host-only Type-A receptacle must provide some VBUS source that is capable of meeting USB specification current requirements. Systems that implement a standard Type-A USB receptacle and use an OTG transceiver must ground the OTG transceiver ID input. If the OTG transceiver is used to monitor VBUS for a standard Type-A USB receptacle, a series resistor is recommended between the OTG transceiver VBUS pin and the connector so that the OTG transceiver cannot drive any significant current onto VBUS.

USB Host Connections Using the OMAP5912 Integrated USB Transceiver

The OMAP5912 integrated USB transceiver can provide connectivity for a host-controller port. Figure 66 shows a way to connect the integrated USB transceiver as a host-only port.

Figure 66. USB Host Connections Using the OMAP5912 Integrated USB Transceiver



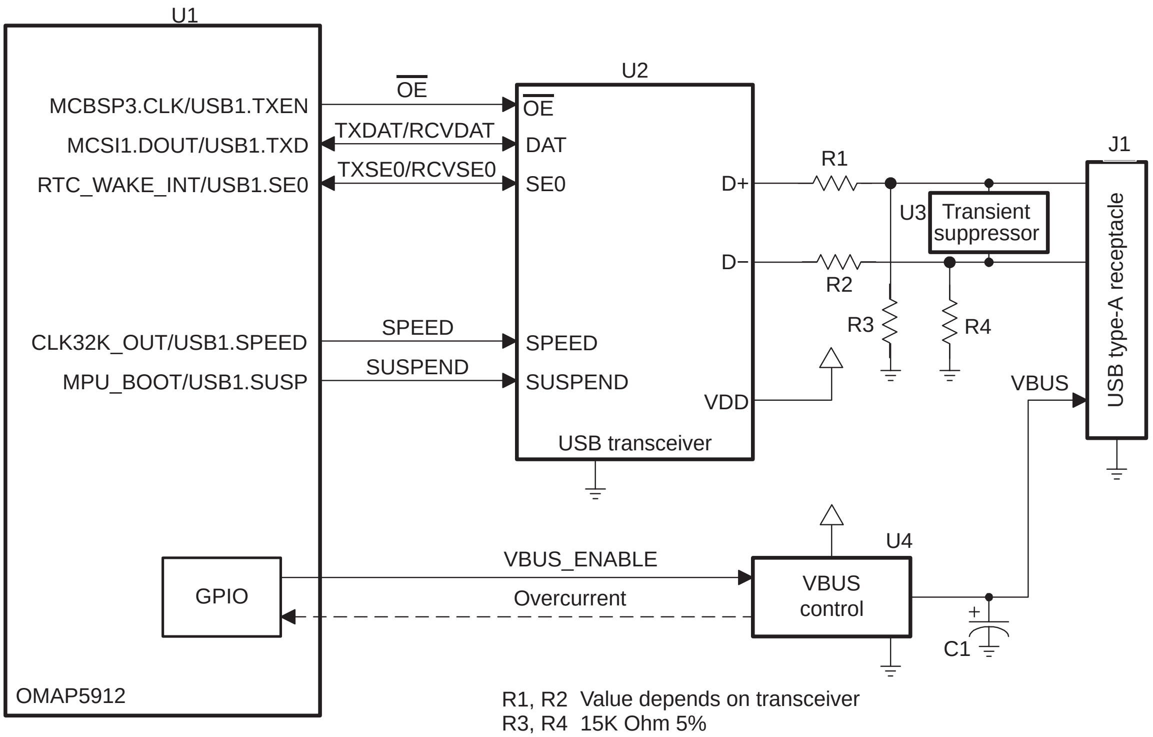
R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR Cap, minimum 120 μF
U1	OMAP5912
U2	Transient suppressor, like SN65220, SN65240, or SN75240
U3	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins USB.DP and USB.DM (see Table 74) and selection of an HMC_MODE value, which routes a host controller port to pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

OMAP5912 integrated pulldowns for pins USB.DP and USB.DM do not meet the USB specifications for D+ and D− pulldowns. The external resistors shown in Figure 66 must be implemented when using USB.DP and USB.DM for a host connection.

Pin Group 1 USB Host Using 3-Wire USB Transceiver

Figure 67. USB Host Connections On USB Pin Group 1 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR Cap, minimum 120 µF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

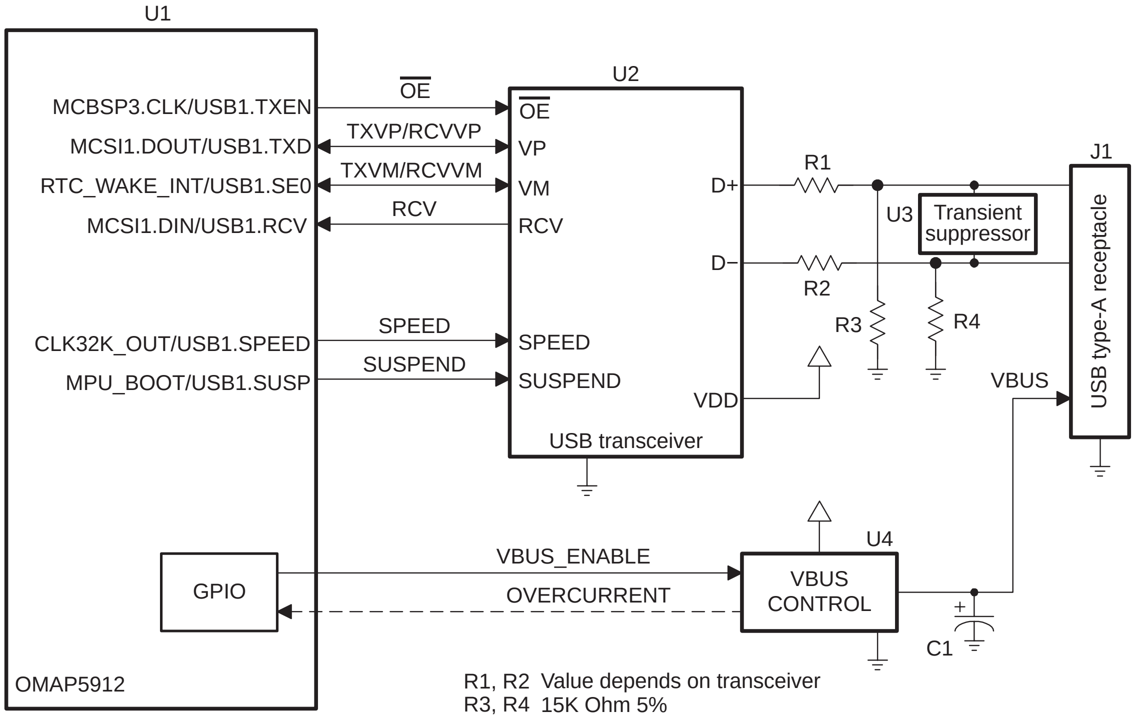
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver.

OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

Pin Group 1 USB Host Using 4-Wire USB Transceiver

Figure 68. USB Host Connections on USB Pin Group 1 Using 4-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

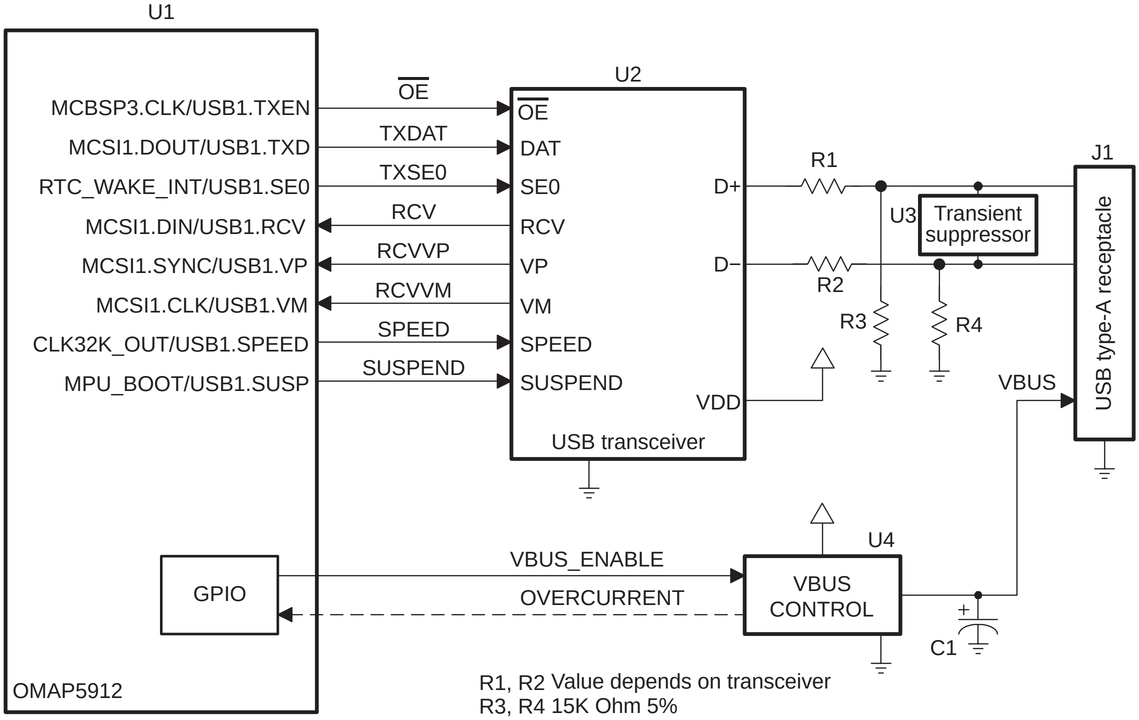
- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- MCSI1.DIN/USB1.RCV
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

Pin Group 1 USB Host Using 6-Wire USB Transceiver

Figure 69. USB Host Connections on USB Pin Group 1 Using 6-Wire OTG Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

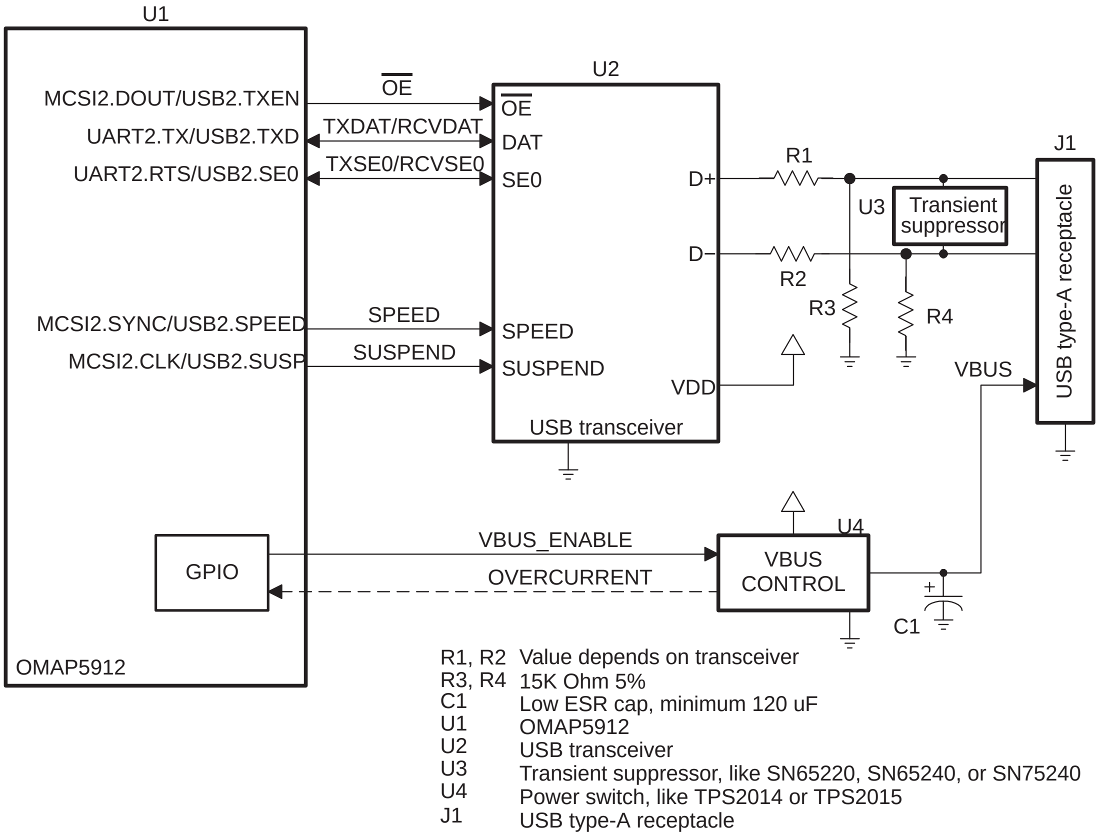
- ❑ MCBSP3.CLKX/USB1.TXEN
- ❑ MCSI1.DOUT/USB1.TXD
- ❑ RTC_WAKE_INT/USB1.SE0
- ❑ MCSI1.SYNC/USB1.VP
- ❑ MCSI1.BCLK/USB1.VM
- ❑ MCSI1.DIN/USB1.RCV
- ❑ CLK32K_OUT/USB1.SPEED
- ❑ MPU_BOOT/USB1.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

Pin Group 2 USB Host Using 3-Wire USB Transceiver

Figure 70. USB Host Connections on USB Pin Group 2 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

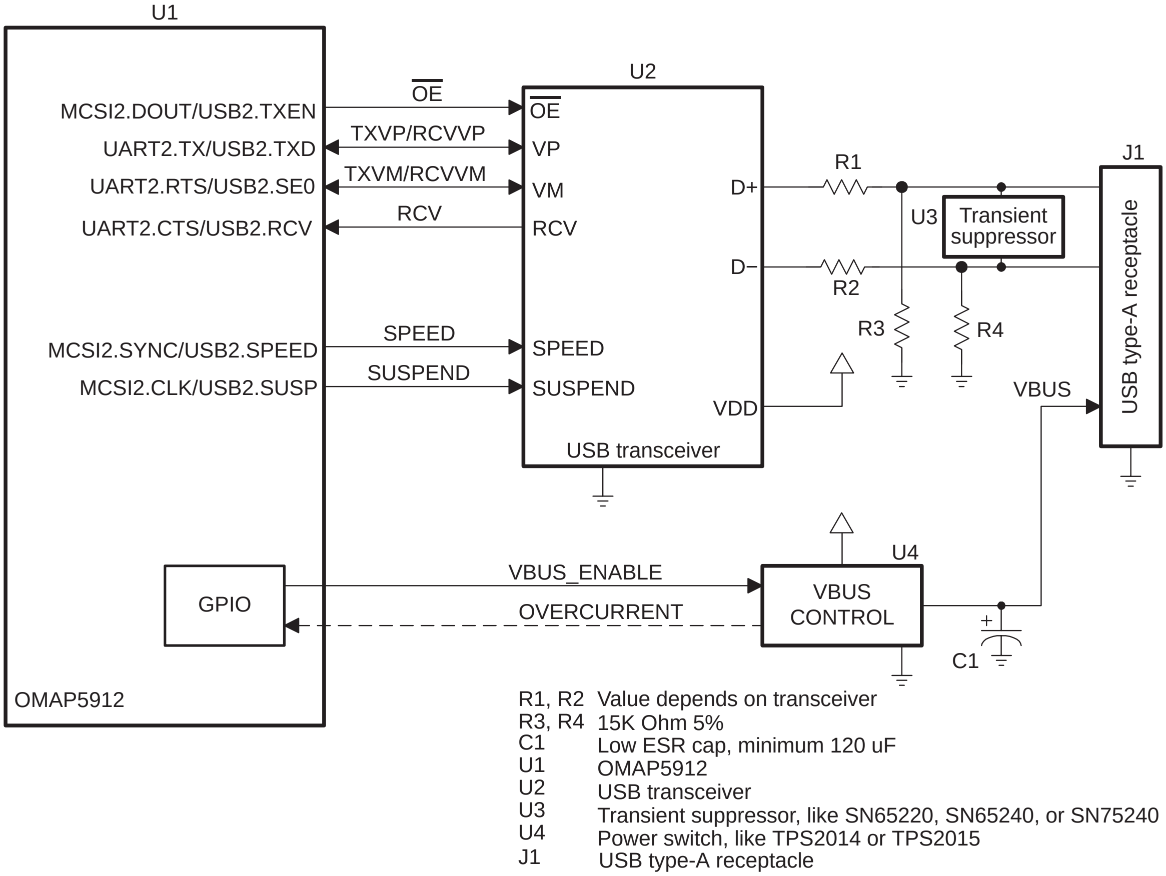
- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- MCSI2.SYNC/USB2.SPEED
- MCSI2.CLK/USB2.SUSPEND

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB2.TXD and UART2.RTS/USB2.SE0.

Pin Group 2 USB Host Using 4-Wire USB Transceiver

Figure 71. USB Host Connections on USB Pin Group 2 Using 4-Wire Transceiver



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

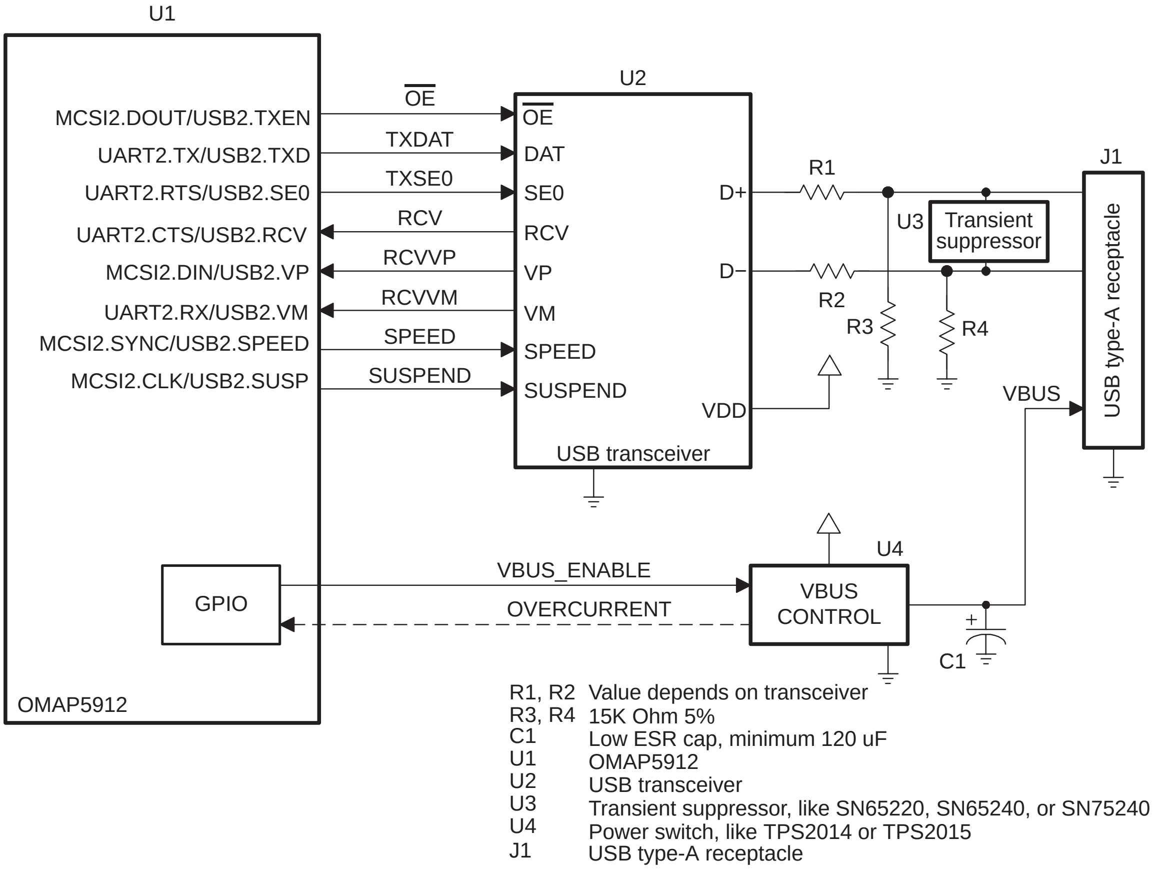
- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.SYNC/USB2.SPEED
- MCSI2.CLK/USB2.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB2.TXD and UART2.RTS/USB2.SE0.

Pin Group 2 USB Host Using 6-Wire USB Transceiver

Figure 72. USB Host Connections on USB Pin Group 2 Using 6-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM
- MCSI2.SYNC/USB2.SPEED
- MCSI2.CLK/USB2SUSP

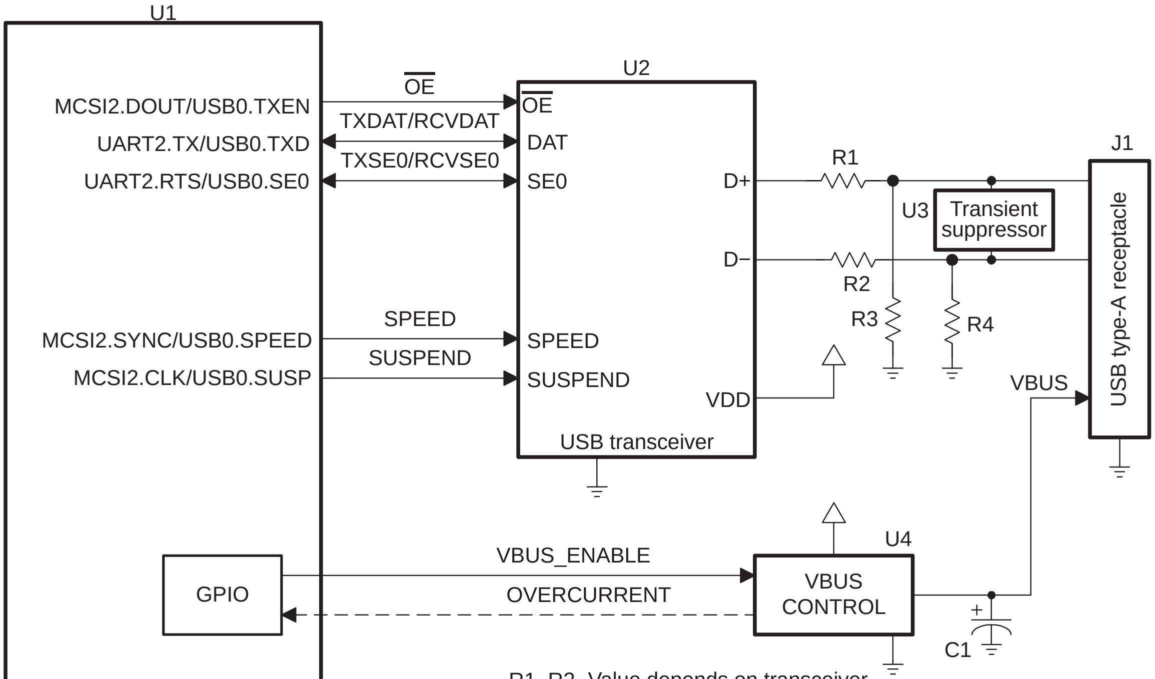
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

USB Host on USB Alternate Pin Group 2, 3-Wire USB Transceiver

Table 73 shows that pin group 2 can receive either of two sets of USB-related signaling. In normal mode, pin group 2 receives the signaling associated with the controller shown under the Port 2 column. In alternate mode, pin group 2 receives signaling associated with the controller shown under port 0. Figure 73 shows USB host port connectivity when pin group 2 is configured for alternate mode and connected using a 3-wire USB transceiver. When pin group 2 is configured in alternate mode, pins USB.DP and USB.DM are not usable for USB functionality.

Figure 73. USB Host Connections on USB Alternate Pin Group 2 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ MCSI2.DOUT/USB0.TXEN
- ❑ UART2.TX/USB0.TXD
- ❑ UART2.RTS/USB0.SE0
- ❑ MCSI2.SYNC/USB0.SPEED
- ❑ MCSI2.CLK/USB0.SUSP

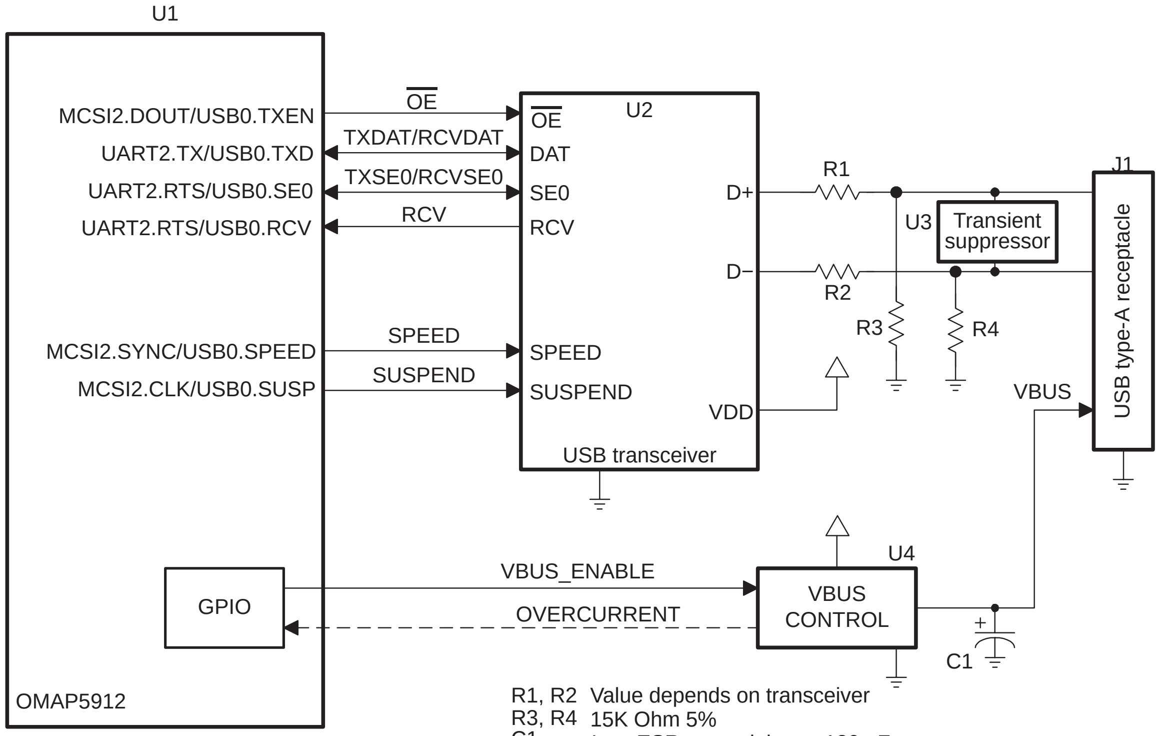
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

USB Host on USB Alternate Pin Group 2, 4-Wire USB Transceiver

Table 73 shows that pin group 2 can receive either of two sets of USB-related signaling. In normal mode, pin group 2 receives the signaling associated with the controller shown under the Port 2 column. In alternate mode, pin group 2 receives signaling associated with the controller shown under port 0. Figure 74 shows USB host port connectivity when pin group 2 is configured for alternate mode and connected using a 4-wire USB transceiver. When pin group 2 is configured in alternate mode, pins USB.DP and USB.DM are not usable for USB functionality.

Figure 74. USB Host Connections on USB Alternate Pin Group 2 Using 4-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

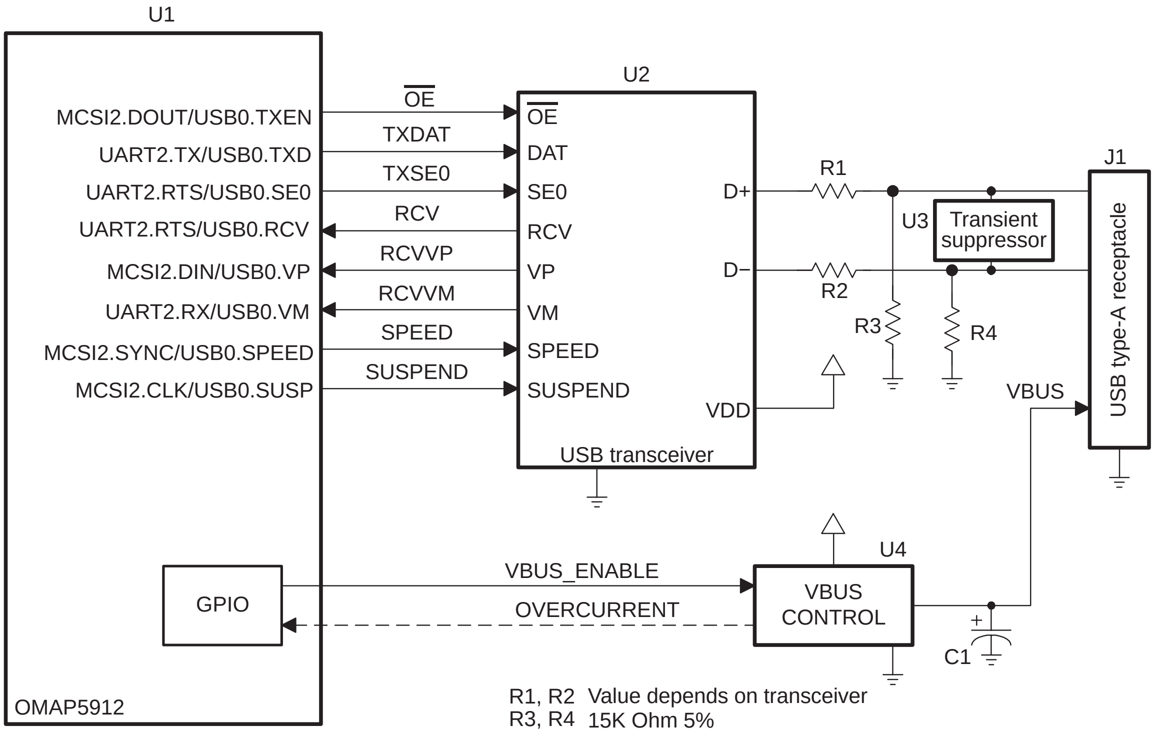
See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_0.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

USB Host on USB Alternate Pin Group 2, 6-Wire USB Transceiver

Table 73 shows that pin group 2 can receive either of two sets of USB-related signaling. In normal mode, pin group 2 receives the signaling associated with the controller shown under the Port 2 column. In alternate mode, pin group 2 receives signaling associated with the controller shown under port 0. Figure 75 shows USB host port connectivity when pin group 2 is configured for alternate mode and connected using a 6-wire USB transceiver. When pin group 2 is configured in alternate mode, pins USB.DP and USB.DM are not usable for USB functionality.

Figure 75. USB Host Connections on USB Alternate Pin Group 2 Using 6-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	15K Ohm 5%
C1	Low ESR cap, minimum 120 uF
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Power switch, like TPS2014 or TPS2015
J1	USB type-A receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins:

- ❑ MCSI2.DOUT/USB0.TXEN
- ❑ UART2.TX/USB0.TXD
- ❑ UART2.RTS/USB0.SE0
- ❑ UART2.CTS/USB0.RCV
- ❑ MCSI2.DIN/USB0.VP
- ❑ UART2.RX/USB0.VM
- ❑ MCSI2.SYNC/USB0.SPEED
- ❑ MCSI2.CLK/USB0.SUSP

See Table 74.

HMC_MODE must be set to a value that routes a host controller port to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

4.8 USB Function Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a non-OTG USB device controller port must implement certain features. These features include a USB type-B or mini-B receptacle, VBUS power detection, transient suppression, a controllable pullup resistor to the D+ or D- line, and USB–compatible upstream port transceiver. These elements are shown in Figure 76.

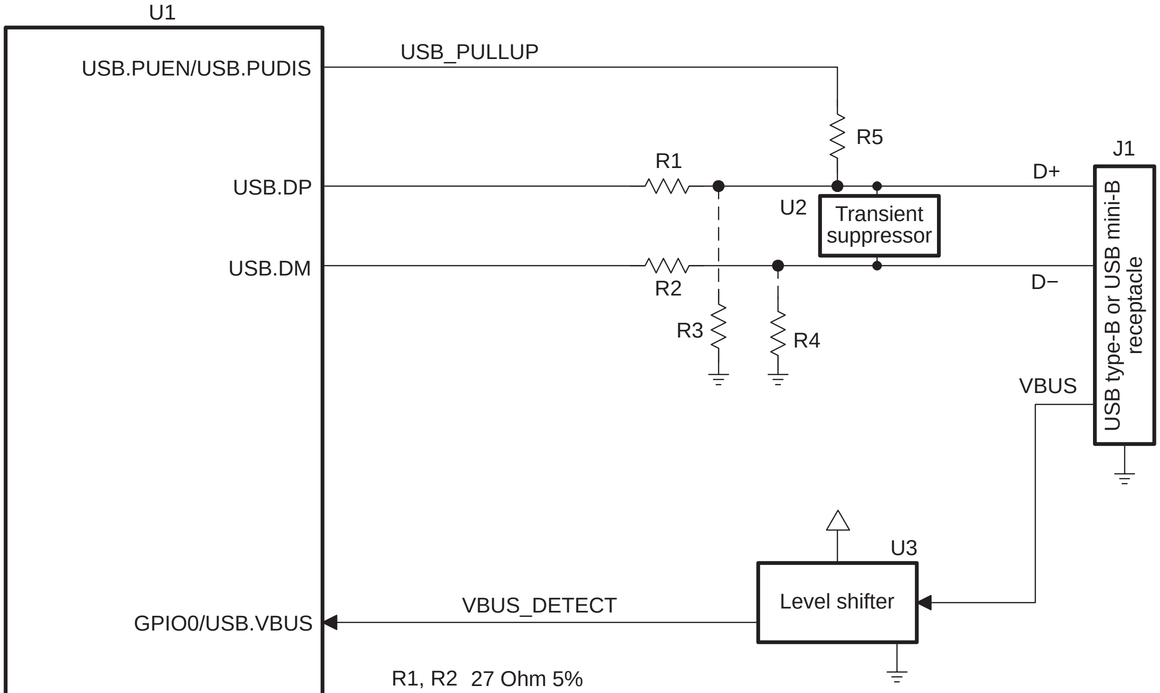
Optional weak pulldown resistors can be used to hold the D+ and D– signals at voltages below the USB transceiver VIL level when there is no USB host connected to the USB Type B connector. By keeping D+ and D– voltages below VIL, the USB transceiver IDDQ can be reduced. Choice of value for the pulldown resistors must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The OMAP5912 USB function controller only supports implementation as a full-speed USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of a full-speed USB device.

Some USB transceivers implement a 6-wire connection to the host controller, whereas others require only 4 wires or 3 wires. Figure 76 through Figure 82 show the signal connectivity options for the integrated transceiver and external 3-, 4-, and 6-wire USB transceivers on each of the available OMAP5912 USB pin groups.

Pin Group 0 USB Device

Figure 76. USB Device Connections Using OMAP5912 Integrated USB Transceiver



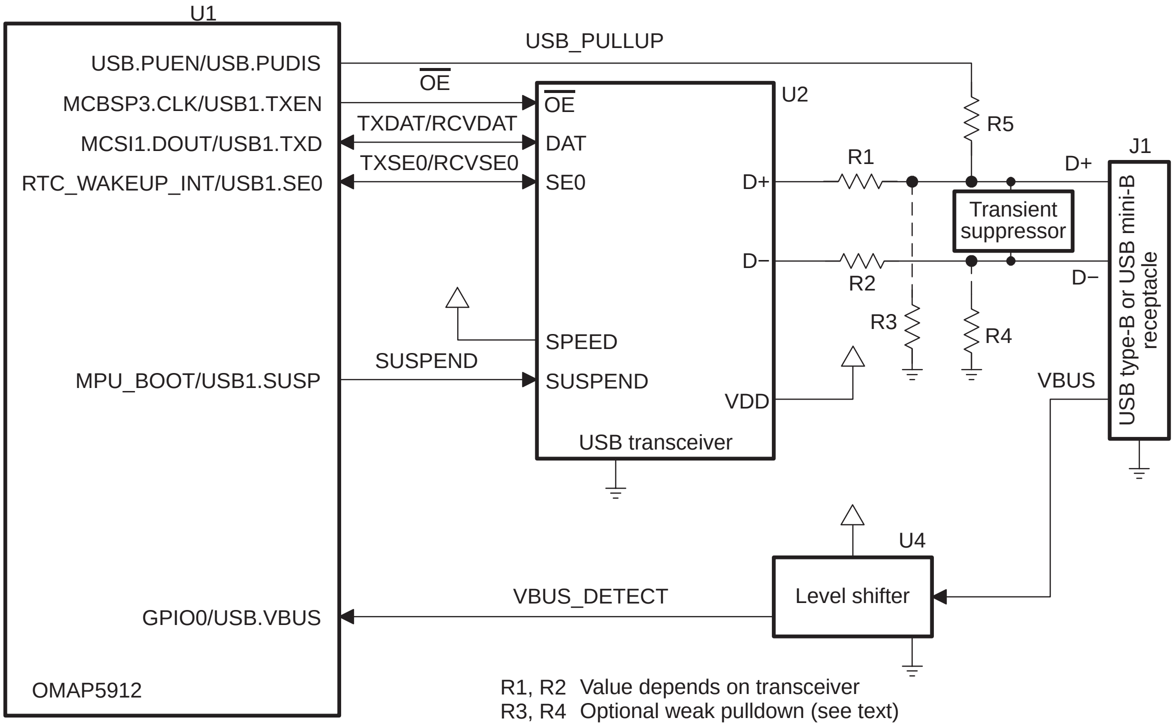
R1, R2	27 Ohm 5%
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	Transient suppressor, like SN65220, SN65240, or SN75240
U3	Level shifter
J1	USB type-B receptacle or USB mini-B receptacle

Proper initialization of the OMAP5912 to support this mode of operation requires proper setting of the top-level multiplexing for USB.DP and USB.DM (see Table 74) and selection of an HMC_MODE value that routes the USB device controller to pin group 0 (see Table 73). OTG_SYSCON_1. USB0.TRX_MODE must be set to 3 to allow proper operation of the integrated USB transceiver.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms. When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 3-Wire USB Transceiver

Figure 77. USB Device Connections on USB Pin Group 1 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

See Table 74.

HMC_MODE must be set to a value that routes the OMAP5912 USB device controller to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver.

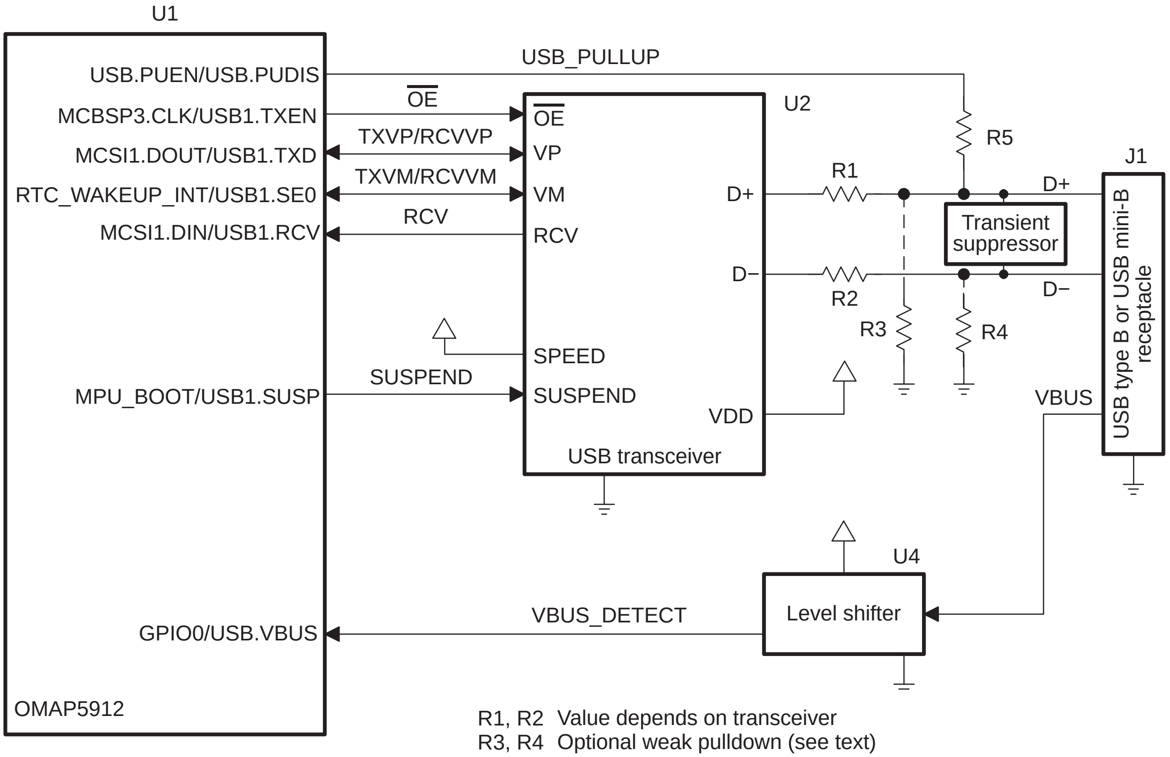
OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^{2}C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^{2}C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. The OTG transceiver ID pin must be left unconnected, and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware connections shown can be removed when an OTG transceiver is used. System software must then control those features via the OTG transceiver I^{2}C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only ses the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 4-Wire USB Transceiver

Figure 78. USB Device Connections on USB Pin Group 1 Using 4-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ MCBSP3.CLKX/USB1.TXEN
- ❑ MCSI1.DOUT/USB1.TXD
- ❑ RTC_WAKE_INT/USB1.SE0
- ❑ CLK32K_OUT/USB1.SPEED
- ❑ MPU_BOOT/USB1.SUSP

See Table 74.

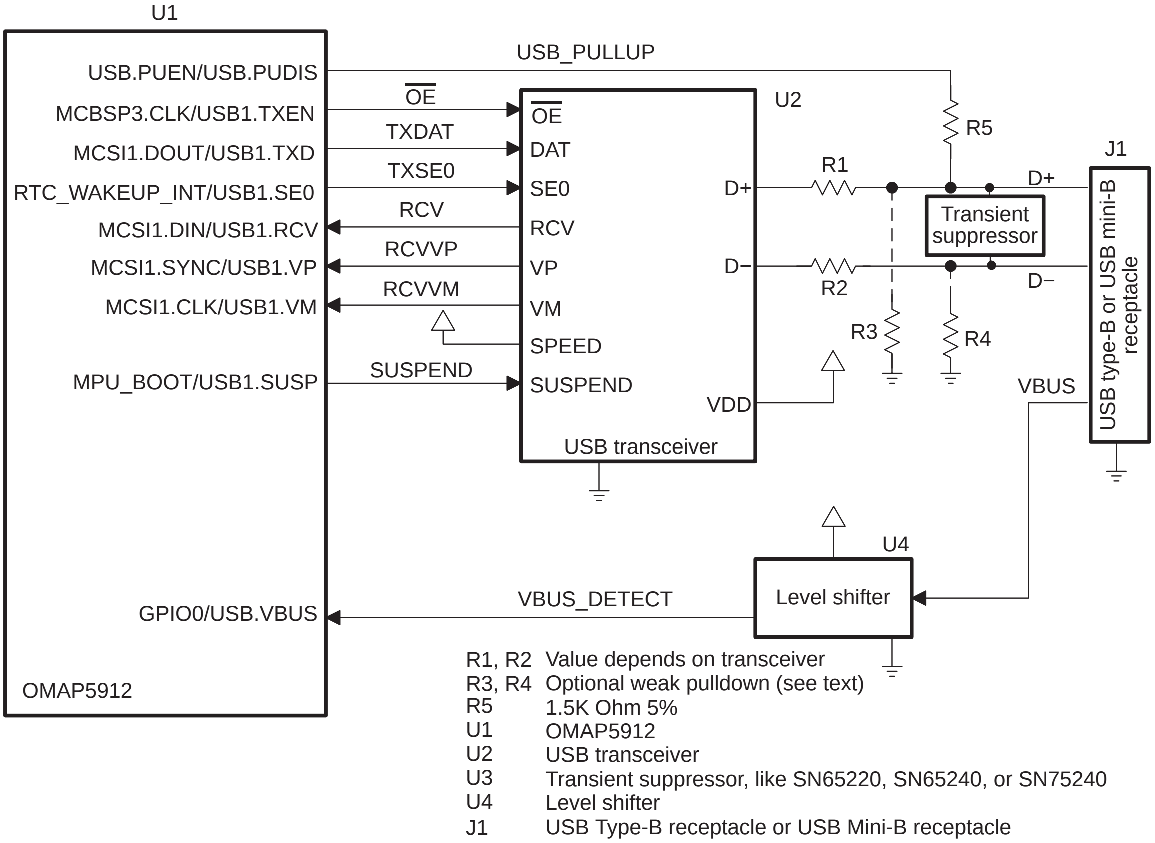
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins MCSI1.DOUT/USB1.TXD and RTC_WAKE_INT/USB1.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^{2}C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^{2}C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, the OTG transceiver ID pin is left unconnected and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware features shown can be removed when an OTG transceiver is used. System software needs to control those features via the OTG transceiver I^{2}C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

Pin Group 1 USB Device Using 6-Wire USB Transceiver

Figure 79. USB Device Connections on USB Pin Group 1 Using 6-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCBSP3.CLKX/USB1.TXEN
- MCSI1.DOUT/USB1.TXD
- RTC_WAKE_INT/USB1.SE0
- MCSI1.SYNC/USB1.VP
- MCSI1.BCLK/USB1.VM
- MCSI1.DIN/USB1.RCV
- CLK32K_OUT/USB1.SPEED
- MPU_BOOT/USB1.SUSP

See Table 74.

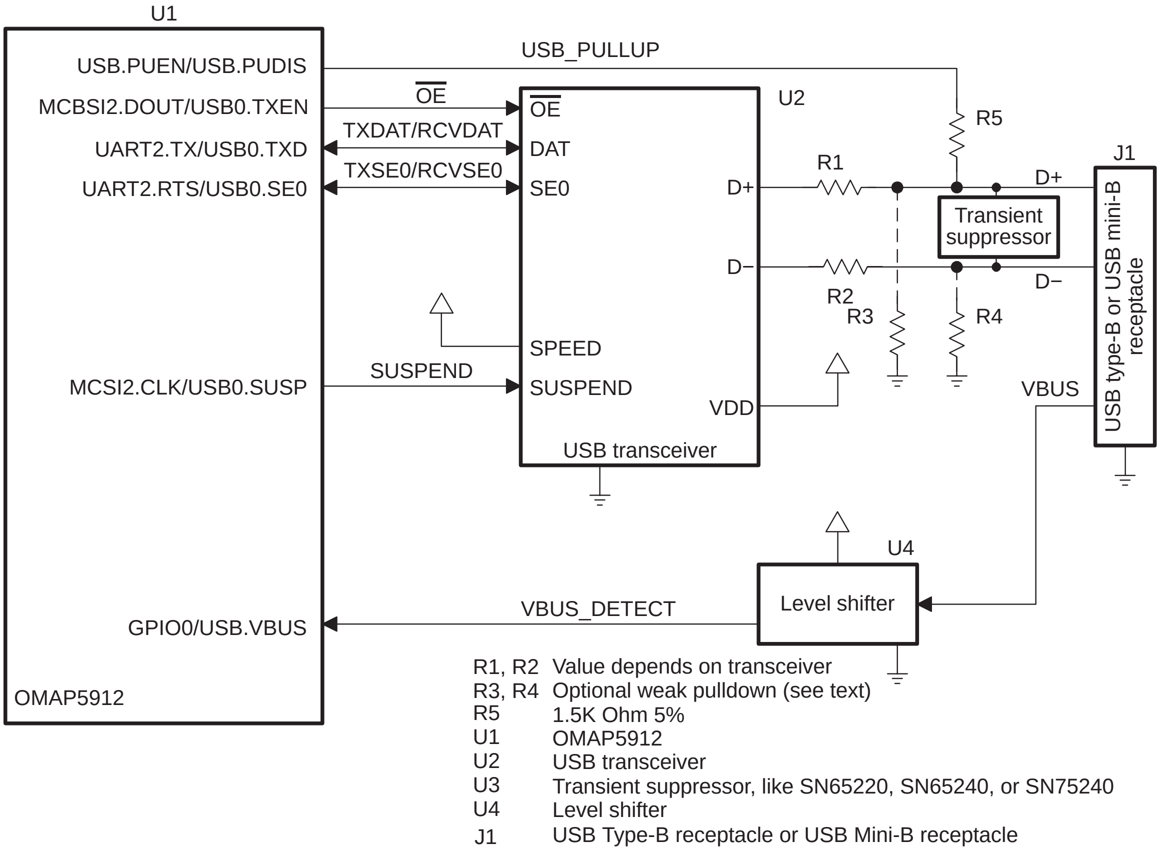
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller (USB_OTG) by reading VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, the OTG transceiver ID pin is left unconnected, and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, the hardware features shown can be removed when an OTG transceiver is used. System software must control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

USB Device on USB Alternate Pin Group 2, 3-Wire USB Transceiver

Figure 80. USB Device Connections on USB Alternate Pin Group 2 Using 3-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ MCSI2.DOUT/USB0.TXEN
- ❑ UART2.TX/USB0.TXD
- ❑ UART2.RTS/USB0.SE0
- ❑ MCSI2.SYNC/USB0.SPEED
- ❑ MCSI2.CLK/USB0.SUSP

See Table 74.

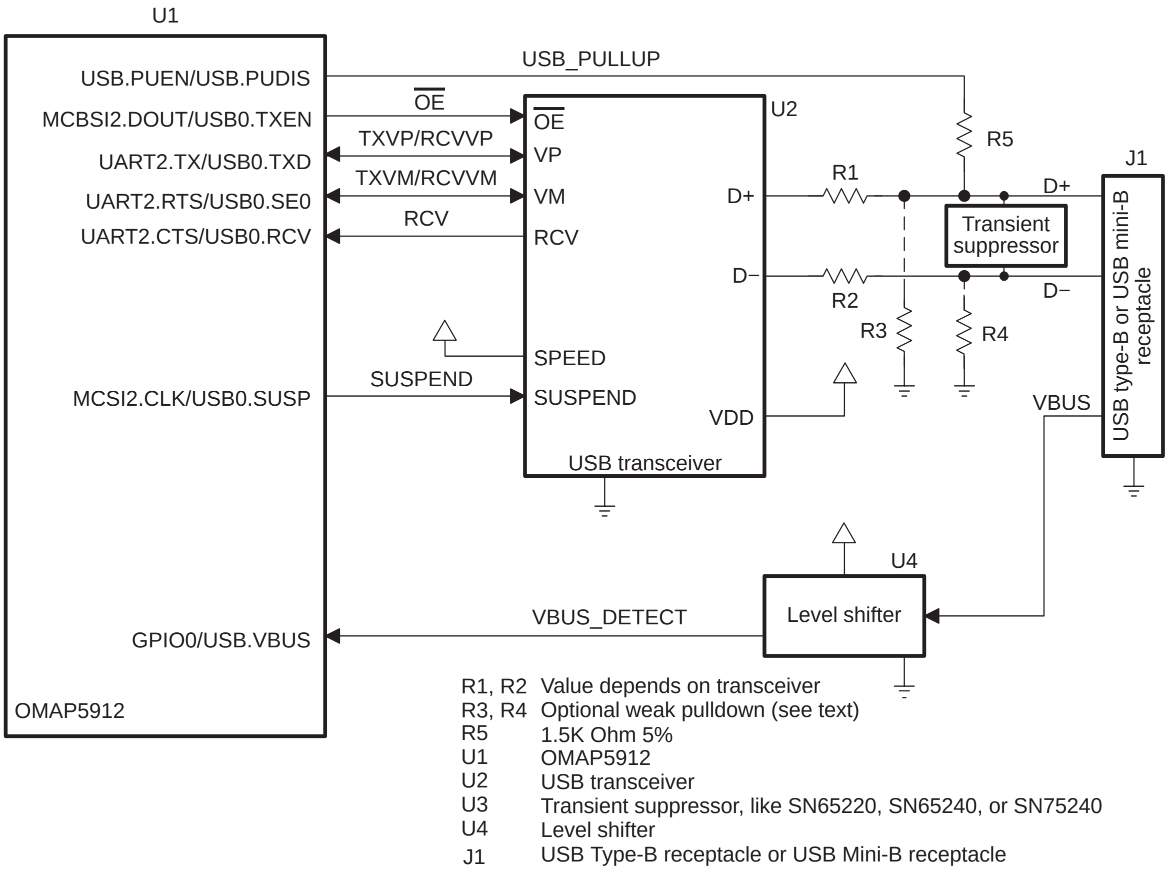
HMC_MODE must be set to a value that routes the USB device controller to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 2 to allow proper bidirectional operation of the 3-wire USB transceiver. OTG_SYSCON_2.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0, and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading the VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. The OTG transceiver ID pin is left unconnected and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware features shown can be removed when an OTG transceiver is used. System software must control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

USB Device on USB Alternate Pin Group 2, 4-Wire USB Transceiver

Figure 81. USB Device Connections on USB Alternate Pin Group 2 Using 4-Wire OTG Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB0.TXEN
- UART2.TX/USB0.TXD
- UART2.RTS/USB0.SE0
- UART2.CTS/USB0.RCV
- MCSI2.SYNC/USB0.SPEED
- MCSI2.CLK/USB0.SUSP

See Table 74.

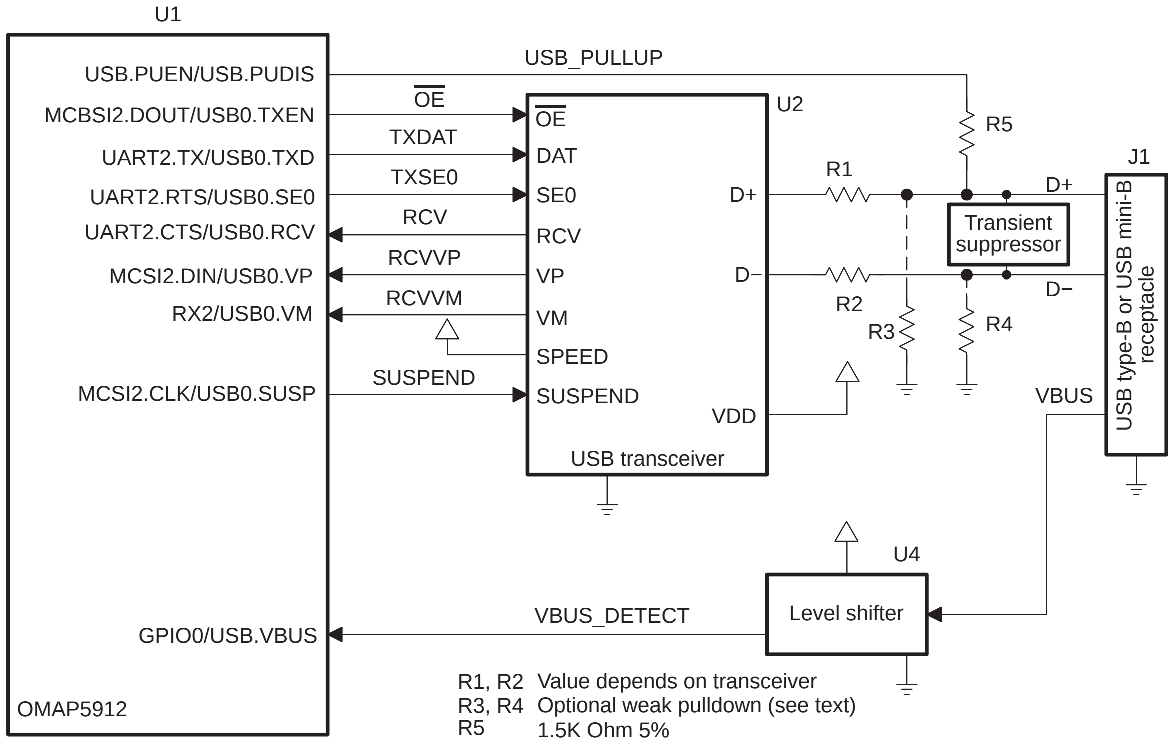
HMC_MODE must be set to a value that routes the USB device controller to USB pin group 0 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 1 to allow proper bidirectional operation of the 4-wire USB transceiver. OTG_SYSCON_0.USBx_SYNCHRO must be set to allow proper bidirectional signaling on pins UART2.TX/USB0.TXD and UART2.RTS/USB0.SE0.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity.In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. The OTG transceiver ID pin is left unconnected, and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware features shown can be removed when an OTG transceiver is used. System software must control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown), and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

USB Device on USB Alternate Pin Group 2, 6-Wire USB Transceiver

Figure 82. USB Device Connections on USB Alternate Pin Group 2 Using 6-Wire Transceiver



R1, R2	Value depends on transceiver
R3, R4	Optional weak pulldown (see text)
R5	1.5K Ohm 5%
U1	OMAP5912
U2	USB transceiver
U3	Transient suppressor, like SN65220, SN65240, or SN75240
U4	Level shifter
J1	USB Type-B receptacle or USB Mini-B receptacle

Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for pins:

❑ MCSI2.DOUT/USB0.TXEN
❑ UART2.TX/USB0.TXD
❑ UART2.RTS/USB0.SE0
❑ UART2.CTS/USB0.RCV
❑ MCSI2.DIN/USB0.VP
❑ UART2.RX/USB0.VM
❑ MCSI2.SYNC/USB0.SPEED
❑ MCSI2.CLK/USB0.SUSP

See Table 74.

HMC_MODE must be set to a value that routes the USB device controller to USB alternate pin group 2 (see Table 73). OTG_SYSCON_1.USB0_TRX_MODE must be set to 3 to allow proper unidirectional operation of the 6-wire USB transceiver.

A USB OTG transceiver can be used to provide USB device-only functionality instead of a USB transceiver, with appropriate I^2C and interrupt connectivity. In this case, OTG_SYSCON_2.OTG_PADEN must be 0 and software passes VBUS validity information from the USB OTG transceiver to the device controller by reading VBUS status via the I^2C link and updating the value sent to the device controller via OTG_CTRL.BSESSVLD. In this case, the OTG transceiver ID pin is left unconnected, and OTG functionality is not available. Because the OTG transceiver provides integrated D+ pullup capability, suspend mode control, and D+/D- pulldown controls, those hardware features shown can be removed when an OTG transceiver is used. System software must
control those features via the OTG transceiver I^2C register set.

When OTG_SYSCON_2.OTG_PADEN is 1, the status of GPIO0/USB.VBUS is automatically provided to the USB device controller via hardware mechanisms (when HMC_MODE provides USB device functionality to a USB pin group). When OTG_SYSCON_2.OTG_PADEN is 0, the USB device controller only sees the VBUS status from the software-controlled register OTG_CTRL.BSESSVLD bit. In this case, software must monitor the VBUS_DETECT signal (using GPIO0 if wired as shown) and update OTG_CTRL.BSESSVLD whenever the VBUS_DETECT signal changes.

4.9 Onboard Transceiverless Connection Using OMAP5912 Transceiverless Link

The transceiverless link logic feature of the OMAP5912 USB signal multiplexing enables connection of the OMAP5912 USB device controller to an external, onboard USB host controller or connection of the OMAP5912 USB host controller, without the use of USB transceivers or associated circuitry. When the transceiverless link logic is used, both of the USB transceivers, the series resistors, pullup and pulldown resistors, VBUS switching components, and USB connectors and cables that typically are used between a USB host controller and the downstream USB device controller are removed. The OMAP5912 transceiverless link logic feature does not support OTG session request protocol or OTG host negotiation protocol, so it cannot be used for a transceiverless OTG link.

The transceiverless link logic signaling system is not suitable for use across a cable. It is intended only for use when the OMAP5912 device is used with an external USB integrated circuit that is on the same board.

When using the transceiverless link logic, six of the external USB integrated circuit pins that typically connect to a USB transceiver connect instead directly to OMAP5912 device pins. Signaling on these pins use CMOS levels. Transceiverless link logic can be used only with external devices that support 6-wire transceiver connectivity.

Figure 83 and Figure 84 show how transceiverless link logic can be compared to a typical USB implementation. Figure 83 shows OMAP5912 being used as a USB host controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP5912 transceiverless link logic. Figure 84 shows OMAP5912 used as a USB device controller, with the top portion of the diagram showing a transceiver-based solution and the bottom portion showing a transceiverless solution using the OMAP5912 transceiverless link logic.

Figure 83. OMAP5912 USB Host Controller Connection—With and Without the OMAP5912 Transceiverless Link Logic

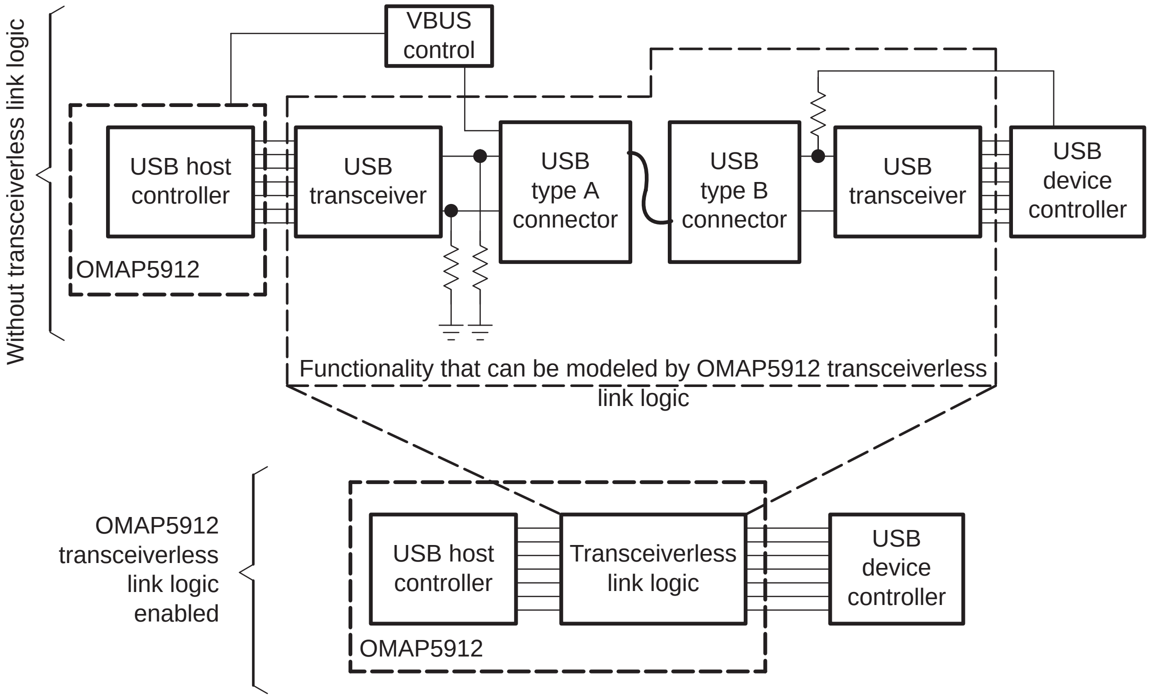
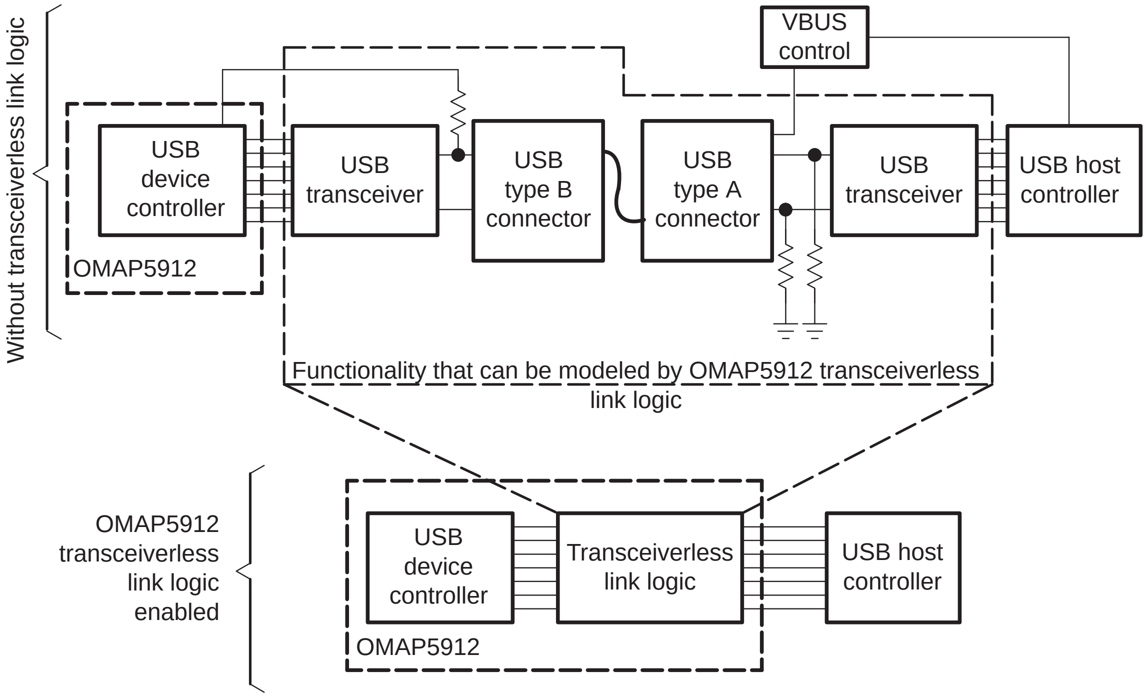


Figure 84. OMAP5912 USB Device Connection—With and Without the OMAP5912 Transceiverless Link Logic



The transceiverless link logic function in the OMAP5912 device interprets the transmit control signals from the external USB integrated circuit and similar signals from the OMAP5912 USB host controller or OMAP5912 USB device controller and computes the equivalent USB differential pair state. The computed differential pair state is interpreted and the appropriate transceiver output signals are provided to the external USB integrated circuit and to the OMAP5912 USB host controller or OMAP5912 USB device controller.

Two control bits help determine the proper differential pair state. TLL_ATTACH and TLL_SPEED provide these inputs and are controlled by OMAP5912 register bits (see Table 75). TLL_ATTACH is used to control when the differential pair models a pullup resistor as is implemented in a transceiver-based USB device. TLL_SPEED is used solely to determine whether the modeled pullup resistor is modeled on the internal version of D+ (for a full-speed transceiverless link) or D- (for a low-speed transceiverless link).

Caution

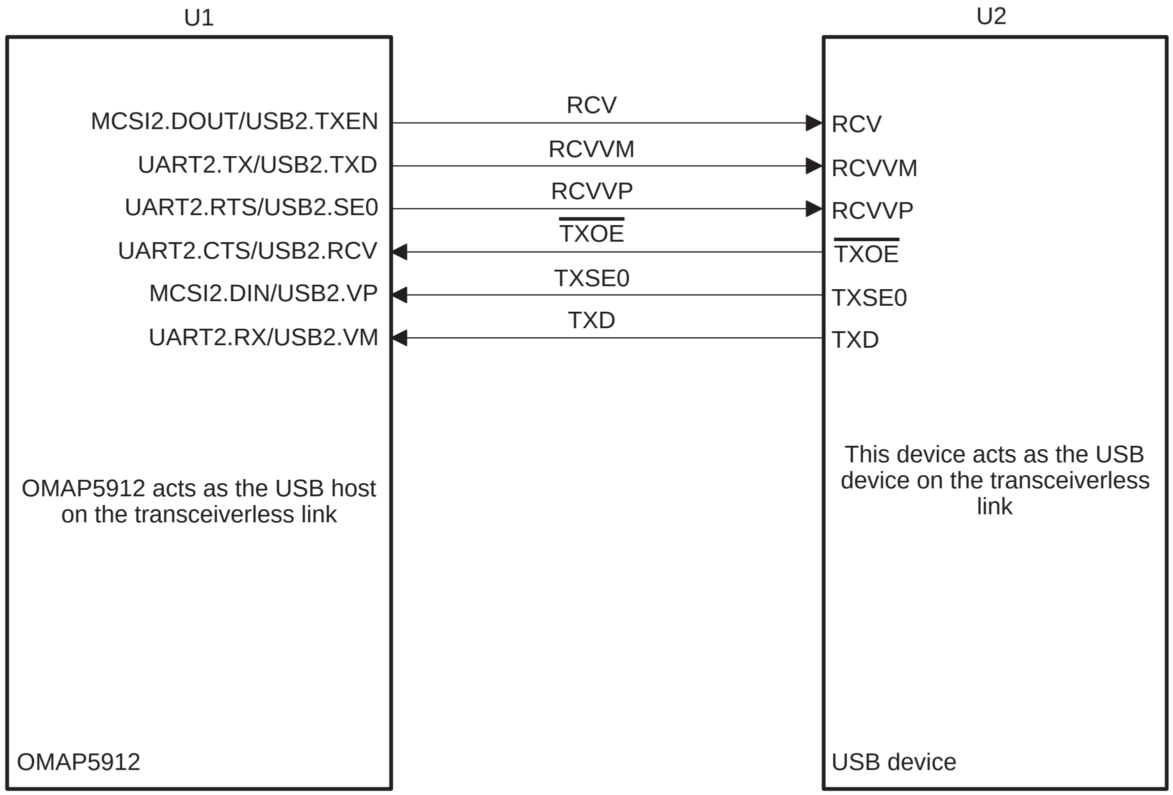
The OMAP5912 USB device cannot operate as a low-speed USB device. Do not set TLL_SPEED to 0 when HMC_MODE connects the OMAP5912 USB device controller to the transceiverless link logic.

The transceiverless link logic cannot be used as part of an OTG connection to an on-board OTG device. An OTG connection always requires an OTG transceiver. Some modes of operation simultaneously support a transceiver-based OTG connection and an additional transceiverless link logic connection to an on-board downstream USB device. OMAP5912 does not provide any modes where both a transceiverless link logic connection to an on-board USB host and a transceiver-based OTG connection are simultaneously available.

USB Host With Transceiverless Link Logic (TXD/TXSE0)

Figure 85 shows connectivity when OMAP5912 acts as the USB host controller on a transceiverless link connection to an on-board USB device controller.

Figure 85. OMAP5912 as USB Host on Transceiverless Link Using TXD/TXSE0 Signaling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

❑ MCSI2.DOUT/USB2.TXEN
❑ UART2.TX/USB2.TXD
❑ UART2.RTS/USB2.SE0
❑ UART2.CTS/USB2.RCV
❑ MCSI2.DIN/USB2.VP
❑ UART2.RX/USB2.VM

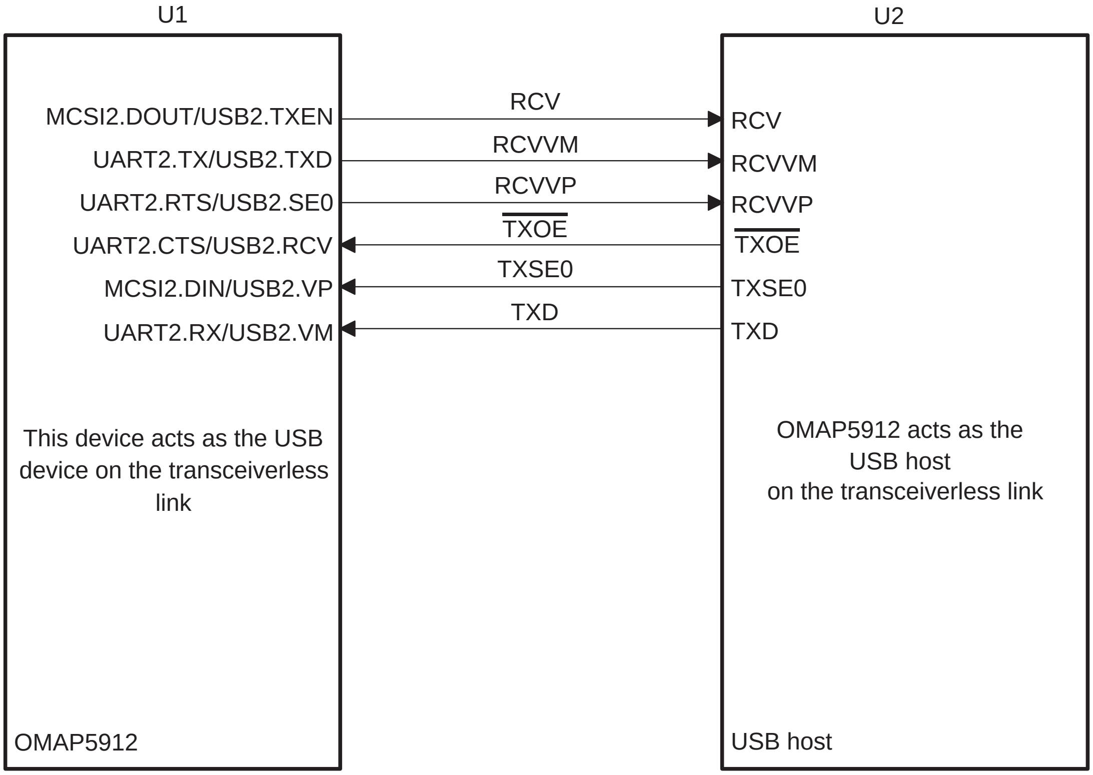
See Table 74.

HMC_MODE must be set to a value that routes a host controller port through the TLL (in TXD/TXSE0 mode) to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection.

USB Device With Transceiverless Link Logic (TXD/TXSE0)

Figure 86 shows connectivity when OMAP5912 acts as the USB device on a transceiverless link connection to an on-board USB host controller.

Figure 86. OMAP5912 as USB Device Controller on Transceiverless Link Using TXD/TXSE0 Signaling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- ❑ MCSI2.DOUT/USB2.TXEN
- ❑ UART2.TX/USB2.TXD
- ❑ UART2.RTS/USB2.SE0
- ❑ UART2.CTS/USB2.RCV
- ❑ MCSI2.DIN/USB2.VP
- ❑ UART2.RX/USB2.VM

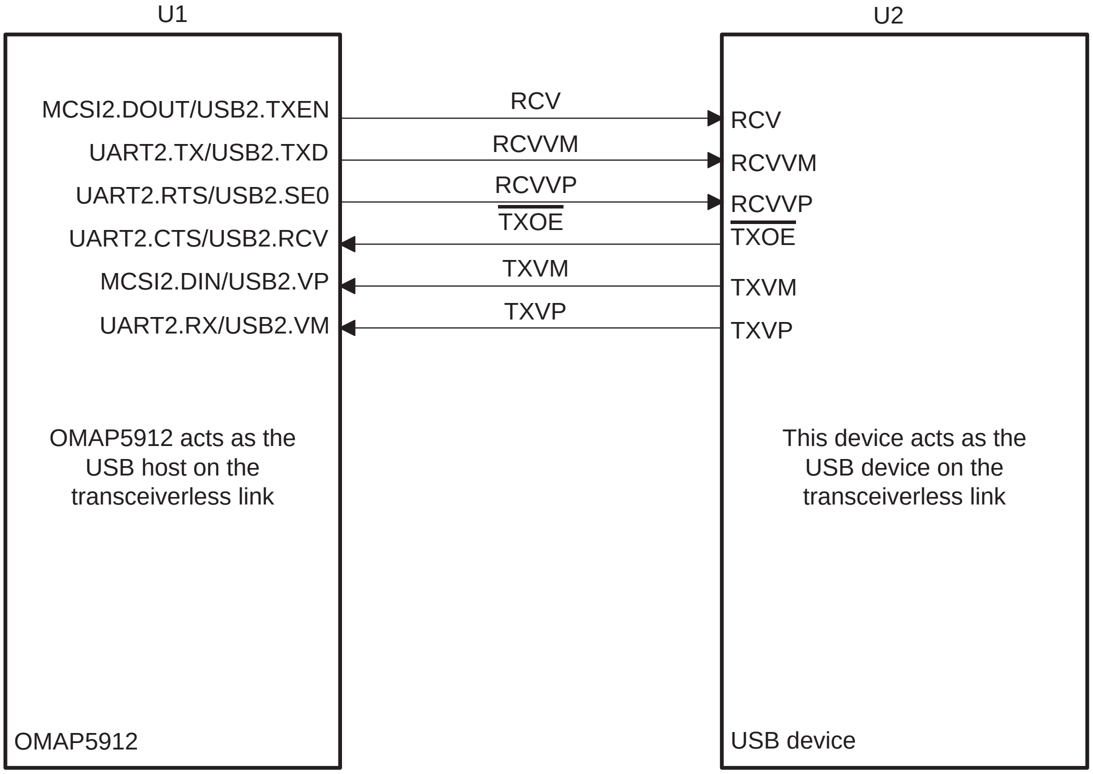
See Table 74.

HMC_MODE must be set to a value that routes the USB device controller through the TLL (in TXD/TXSE0 mode) to USB pin group 2 (see Table 73).

OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection. TLL_SPEED must be 1 when using the OMAP5912 USB device controller with transceiverless link logic.

USB Host With Transceiverless Link Logic (TXVP/TXVM)

Figure 87. OMAP5912 as USB Host Controller on Transceiverless Link Using TXVP/TXVM Signaling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

- MCSI2.DOUT/USB2.TXEN
- UART2.TX/USB2.TXD
- UART2.RTS/USB2.SE0
- UART2.CTS/USB2.RCV
- MCSI2.DIN/USB2.VP
- UART2.RX/USB2.VM

See Table 74.

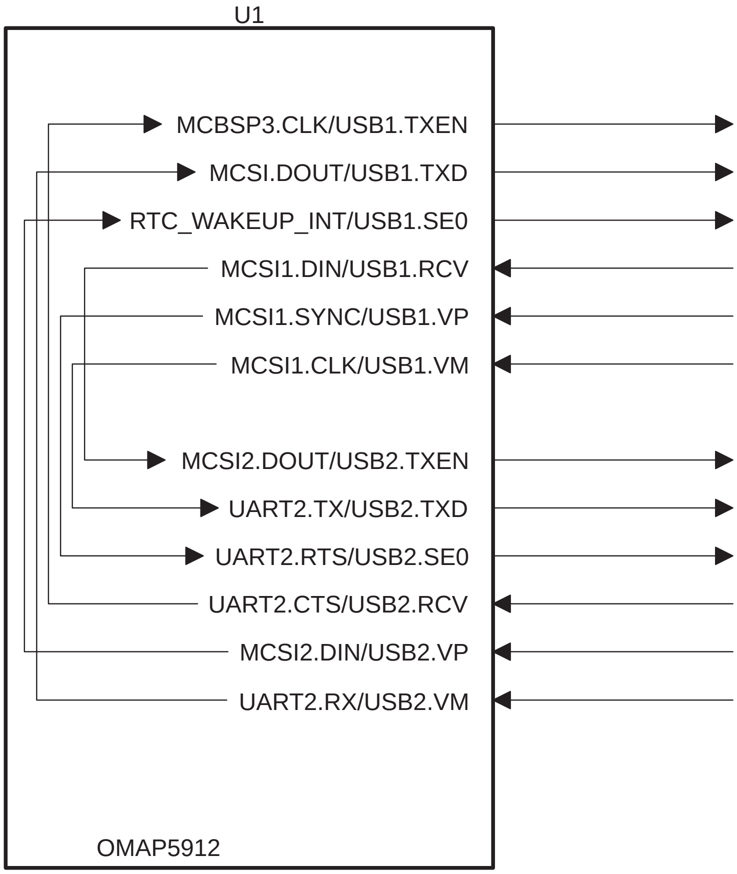
HMC_MODE must be set to a value that routes a host controller port through the TLL (in TXVP/TXVM mode) to USB pin group 2 (see Table 73). OTG_SYSCON_1.USB2_TRX_MODE must be set to 3 to allow proper operation of the transceiverless link logic connection.

4.10 Other Pin Connectivity Controlled by USB Signal Multiplexing

Pin Group 1 to Pin Group 2 Internal Loopback

Figure 88 shows how the OMAP5912 internally connects the pins when configured for HMC_MODE 7.

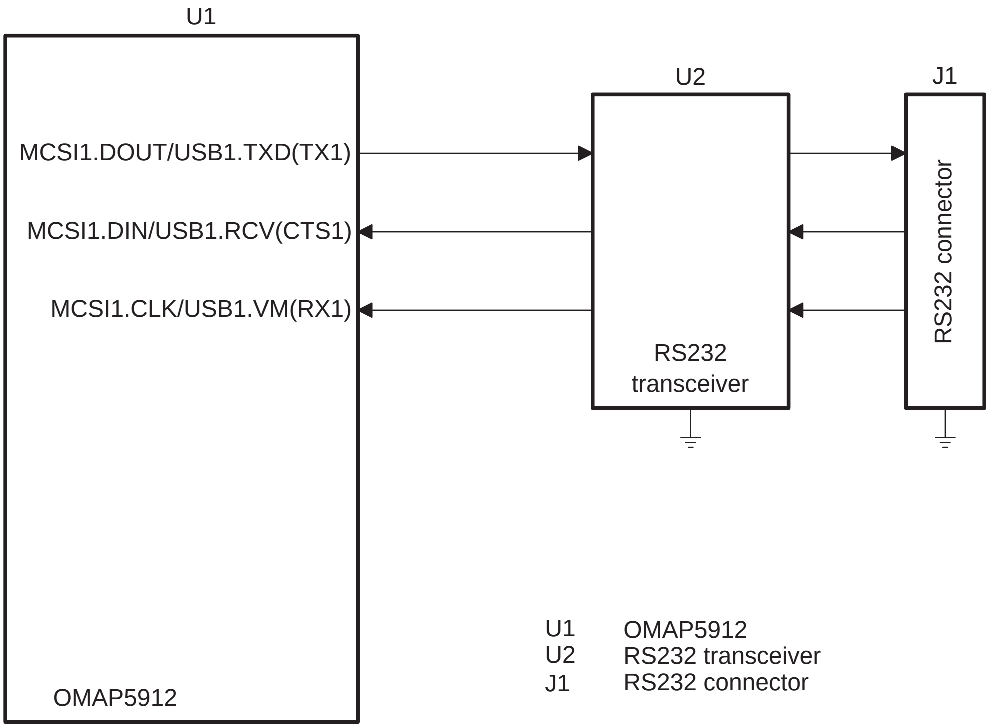
Figure 88. OMAP5912 Pin Group 1 to Group 2 Internal Loopback (HMC_MODE = 7)



UART 1 on USB Pin Group 2 Pins

When configured for some HMC_MODE values, the USB signal multiplexing mechanism routes UART1 signals to some OMAP5912 USB pin group 1 pins (see Figure 89).

Figure 89. External Connectivity When USB Pin Group 1 Configured for UART1 Signalling



Proper initialization of the OMAP5912 device to support this mode of operation requires proper setting of the top-level pin multiplexing for these pins:

❑ MCSI1.DOUT/USB1.TXD
❑ MCSI1.BCLK/USB1.VM
❑ MCSI1.DIN/USB1.RCV

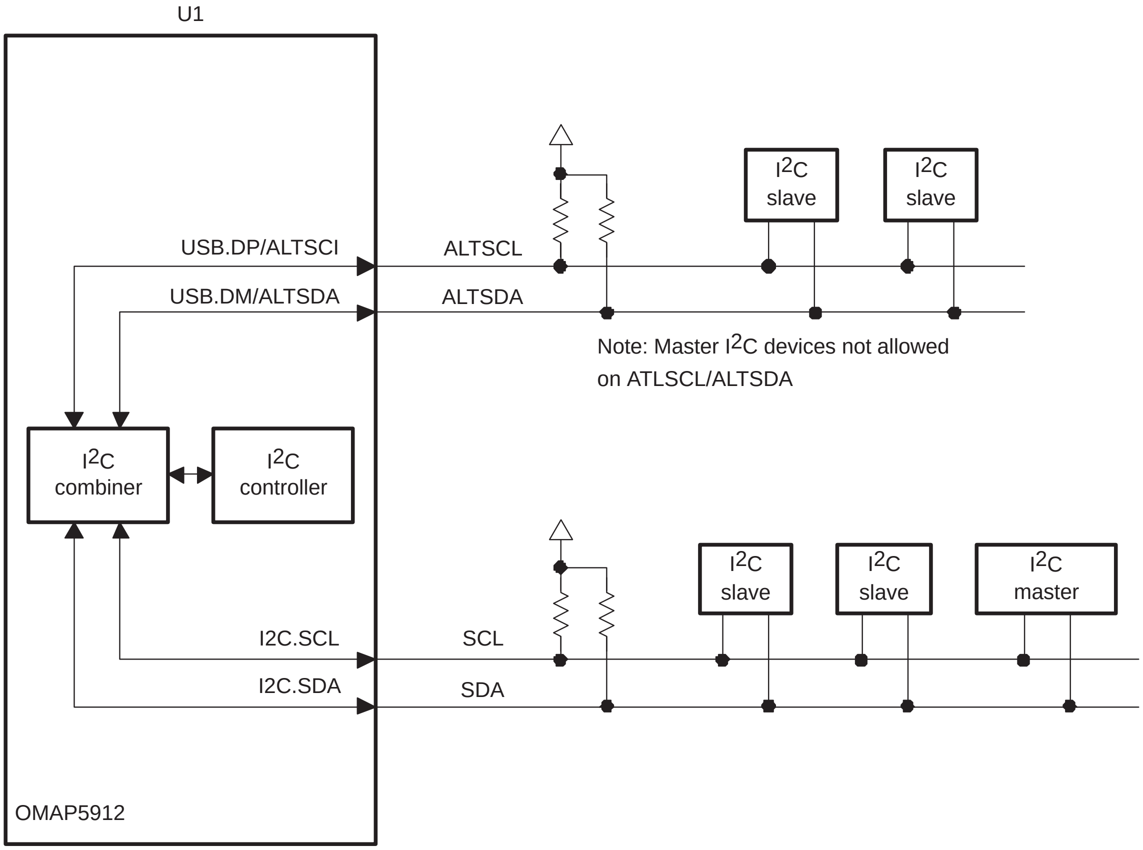
See Table 74.

HMC_MODE must be set to a value that routes the UART1 to USB pin group 1 (see Table 73). OTG_SYSCON_1.USB1_TRX_MODE must be set to 3 to allow proper operation of the 3-wire connection to the external transceiver.

I^2C on Pin Group 0

OMAP5912 supports a mode in which the OMAP5912 I^2C signals are brought out via USB pin group 0 pins. In this mode of operation, top-level pin multiplexing is configured to provide the SCL signal on the USB.DP pin and the SDA signal on the USB.DM pin. This is configured using the USB_TRANSCEIVER_CTRL.CONF_USB0 register, (see Figure 90). You must also set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the I^2C signals have no effect on the OMAP5912 USB controllers.

Figure 90. I^2C on USB Pin Group 0



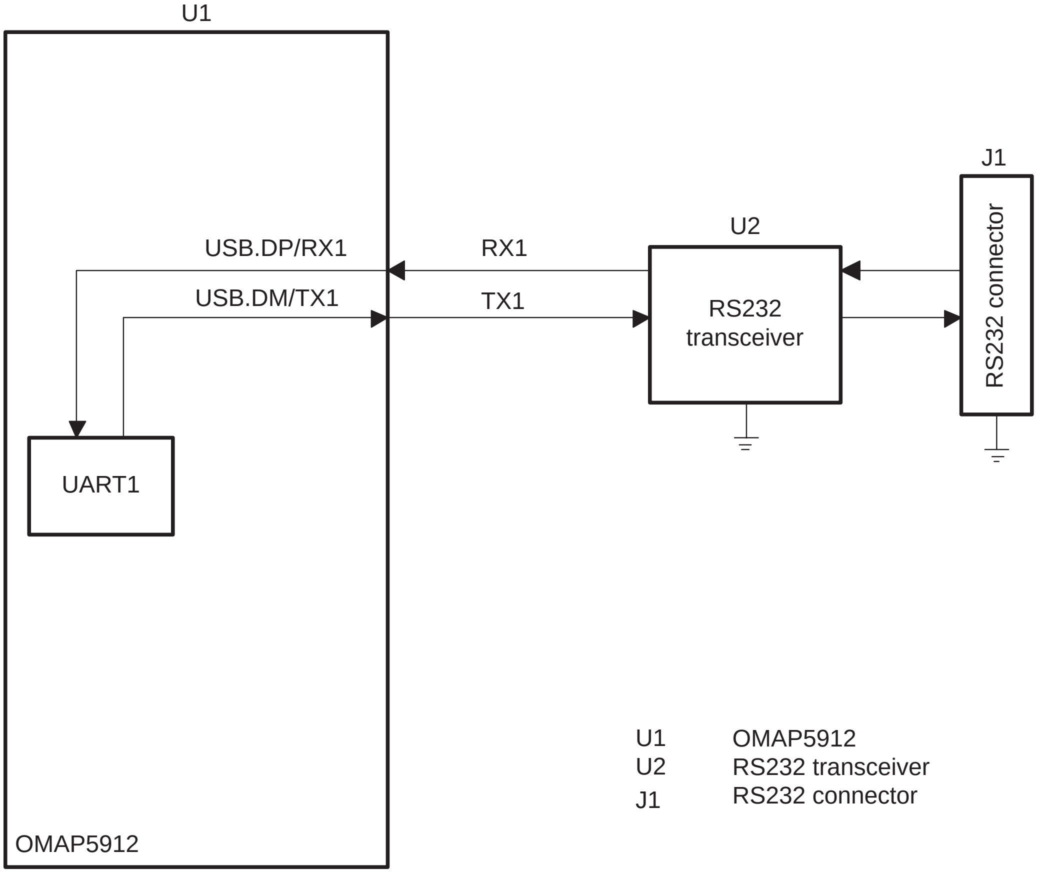
This mode of operation does not allow mastering I^2C devices on the I^2C link on pins USB.DP and USB.DM. Mastering I^2C is allowed on SCL and SDA. A master I^2C device connected externally to SCL and SDA is only able to access the I^2C slaves on SCL and SDA or the OMAP5912 I^2C controller slave function. An external master I^2C device on SCL and SDA cannot access I^2C slaves on ALTSCL and ALTSDA.

This mode of operation implements only one I^2C controller. OMAP5912 internally combines ALTSCL and SCL, and ALTSDA and SDA. This means that it is not possible to implement I^2C slaves with identical I^2C addresses on both links.

UART1 on Pin Group 0

OMAP5912 supports a mode in which the OMAP5912 UART1.TX and UART1.RX signals are brought out via USB pin group 0 pins. In this mode of operation, top-level pin multiplexing is configured to provide the RX input on USB.DP/UART1.RX and provide the TX output on USB.DM/UART1.TX. This is configured using the USB_TRANSCEIVER_CTRL.CONF_USB0 register (see Figure 91). When using this mode, set USB_TRANSCEIVER_CTRL.CONF_USB0_ISOLATE to 1 to ensure that the UART1 signals have no effect on the OMAP5912 USB controllers.

Figure 91. UART1 on USB Pin Group 0



U1	OMAP5912
U2	RS232 transceiver
J1	RS232 connector

4.11 Conflicts Between USB Signal Multiplexing and Top-Level Multiplexing

When OMAP5912 top-level signal multiplexing selects non-USB functionality for a pin but USB signal multiplexing is set to use that pin as an output, the signal from the USB signal multiplexing is ignored and the source selected by the OMAP5912 top-level signal multiplexing is used.

When OMAP5912 top-level signal multiplexing selects non-USB functionality for a pin but the USB signal multiplexing is set to use that pin as an input, the OMAP5912 top-level signal multiplexing presents a low level to the USB signal multiplexer.

It may be useful to select a HMC_MODE value that brings some USB signals to the OMAP5912 top-level signal multiplexing, but then set the top-level signal multiplexing to ignore those USB signals.

4.12 OMAP5912 USB Hardware Considerations

4.12.1 VBUS Power Switching for USB Type A Host Receptacles

The USB specification places several VBUS requirements on USB hosts, including current capability, droop, and other important characteristics. Circuits that meet the USB specification requirements can be implemented using
Texas Instruments devices such as the TPS2014 and TSP2015 power distribution switch devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

Although an OTG transceiver has the ability to power VBUS within the OTG specification limits for an OTG dual role device, it may not have sufficient current sourcing capability to meet the *USB 1.1 Specification*, which requires far greater output current on VBUS. If an OTG downstream port must also support a downstream non-OTG USB device (via a standard-A receptacle to mini-A plug as defined in the OTG specification), a VBUS switch as mentioned above can be used to meet those requirements.

4.12.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when a user connects or disconnects a USB cable can have a dramatic effect on a system if not suppressed. Texas Instruments offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 universal serial bus port transient suppressor devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

4.12.3 VBUS Monitoring for USB Function Controller

A USB function controller must be capable of monitoring the VBUS voltage provided by the upstream USB host controller. The OMAP5912 device provides the input pin USB.VBUS, which is provided to the OMAP5912 USB function controller (when HMC_MODE provides USB device functionality to a USB pin group). This input is a CMOS input that is not rated for the full VBUS range defined by the USB specification. An external signal level shifter is required to convert the VBUS signal range to a range suitable for the OMAP5912 USB.VBUS pin.

4.12.4 USB D+ Pullup Enable for USB Function Controller

When using a USB signal multiplexing mode that provides USB function controller signals to OMAP5912 pins, the OMAP5912 top-level pin multiplexing options lead to several possible USB pullup implementations.

When the USB.PUEN pin is set for top-level multiplexing configuration 0, the USB.PUEN pin is driven low when the pullup is active and is driven high when the pullup is inactive. In this mode of operation, an external inverter or an external 3-statable device can be used to provide a nominal 3.3-V signal to the supply end of the USB D+ pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 1, the USB.PUEN pin provides a clock output and cannot be used to control the USB pullup.

When the USB.PUEN pin is set for top-level multiplexing configuration 3, the USB.PUEN pin is driven high when the pullup is active and is driven low when the pullup is inactive. In this mode of operation, the pullup resistor can be connected directly between the OMAP5912 USB.PUEN and the D+ signal.

4.12.5 MPU_BOOT Signal Sharing

The OMAP5912 device implements shared functionality on the MPU_BOOT/USB1.SUSP pin. The MPU_BOOT pin is sampled at hardware reset. If low, the MPU processor boots from memory connected to CS0 on the EMIFS; if high, the MPU processor enters the boot overlay mode, causing it to boot from memory connected to CS3 on the EMIFS. After reset, the pin can be configured for other functionality, such as the USB1.SUSP output. The MPU_BOOT signal has an internal pulldown resistor that is enabled by default. The boot overlay mode requires an external pullup resister. The internal pulldown can be disabled in the OMAP configuration registers.

4.12.6 USB D+, D- Pulldown for USB Function Controller

System implementations that use the OMAP5912 USB function controller and are sensitive to supply current requirements can implement weak pulldown resistors on the USB D+ and D- signals associated with the USB Type-B receptacle. When there is no host controller attached upstream of the USB Type-B receptacle, the undriven D+ and D- wires can float to voltages that cause excessive current consumption by the USB transceiver. Weak pulldowns can help prevent this problem. Selection of pulldown resistors depends on transceiver characteristics, D+ pullup resistor implementation, and board layout, and must be designed to meet USB D+ and D- signal requirements. The OMAP5912 integrated USB transceiver includes programmable weak pulldowns:

- When USB_TRANSCEIVER_CTRL.CONF_USB0_PWRDN_DN is 0, the internal weak pulldown is enabled for OMAP5912 pin USB.DM.
- When USB_TRANSCEIVER_CTRL.CONF_USB0_PWRDN_DP is 0, the internal weak pulldown is enabled for OMAP160 pin USB.DP.

Index