ISP1763A programming guide

PM0070

Programming manual

Abstract

The ISP1763A is a single-chip Hi-Speed Universal Serial Bus (USB) On-The-Go (OTG) controller integrated with the advanced slave host controller and the ISP1582 peripheral controller. This document provides general guidelines on the software development for the ISP1763A. It covers procedures for major operations for each functional module. This document serves as a start point for software development for the ISP1763A.

Keywords

isp1763a; host controller; peripheral controller; otg controller; usb; universal serial bus

Legal information

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

Contents

1 About this document 5
1.1 Purpose 5
1.2 Audience 5
1.3 Prerequisites 5
1.4 Revision information 5
1.5 Reference list 5

2 Introduction 6

3 Host programming guide 7
3.1 Controller registers and memory description 8
3.1.1 Host controller register description 8
3.1.2 Memory description 9
3.2 Functional description 12
3.2.1 Host controller memory management 13
3.2.2 Host controller register initialization 14
3.2.3 Interrupt handling 16
3.2.4 ATL transfer descriptor handling 20
3.2.5 INT transfer descriptor handling 26
3.2.6 ISO transfer descriptor handling 34
3.2.7 Internal hub enumeration 39
3.2.8 External device enumeration 40
3.2.9 Power management 41

4 Peripheral programming guide 43
4.1 Basic operations 43
4.1.1 Power on initialization 43
4.1.2 Configure endpoints 43
4.1.3 Interrupt function 44
4.1.4 Reading data from the endpoint buffer 45
4.1.5 Writing date to the endpoint buffer 45
4.2 USB control transfer 45

4.2.1 Set-up token with data IN stage 45
4.2.2 Set-up token with data OUT stage 47
4.2.3 Set-up token without data stage 48
4.2.4 Stall set-up token 49

5 OTG programming guide 50
5.1 OTG driver software architecture 50
5.2 OTG interrupt handling basic 51
5.3 Basic register bit explanation 51
5.4 OTG timer 52
5.5 Generating SRP 52
5.6 A_host state diagram 54
5.7 A_host pseudo code 56
5.8 B-device state diagram 60
5.9 B-device pseudo code 61
5.10 Generating HNP 62

Glossary 64

1 About this document

1.1 Purpose

This document provides guidelines on the software development for the ISP1763A OTG controller.

1.2 Audience

Developers who develop software for the ISP1763A.

1.3 Prerequisites

Fundamental knowledge of the USB technology, such as, USB host, USB peripheral, and basic USB operations, such as enumeration and basic USB protocols.

1.4 Revision information

Table 1 Revision history

Date	Rev.	Comments
2010-04-01	1	First release.
2013-04-04	2	Improved the quality of the PDF rendition. No other change in the content.
2013-10-02	3	Applied ST branding. No change in the content.

1.5 Reference list

[1] *ISP1763A Hi-Speed USB OTG controller data sheet* *CD00264885*

[2] *ISP1582/83 firmware programming guide (AN10039)* *CD00222774*

[3] *Universal Serial Bus Specification Rev. 2.0* *www.usb.org*

[4] *On-The-Go Supplement to the USB Specification Rev. 1.3* *www.usb.org*

[5] *Enhanced Host Controller Interface Specification for Universal Serial Bus Rev. 1.0*

2 Introduction

The ISP1763A is a single-chip Hi-Speed Universal Serial Bus (USB) On-The-Go (OTG) controller integrated with the advanced slave host controller and the ISP1582 peripheral controller.

The Hi-Speed USB host controller and peripheral controller comply with *Universal Serial Bus Specification Rev. 2.0* and support data transfer speeds of up to 480 Mbit/s. The Enhanced Host Controller Interface (EHCI) core implemented in the host controller is adapted from *Enhanced Host Controller Interface Specification for Universal Serial Bus Rev. 1.0*. The OTG controller complies with *On-The-Go Supplement to the USB Specification Rev. 1.3*.

The ISP1763A has two USB ports. Port 1 can be configured to function as a downstream port, an upstream port, or an OTG port. Port 2 is always configured as a downstream port. Port 2 supports Session Request Protocol (SRP) detection from the B-device. Both port 2 and port 1 can support the same peripheral at the same speed and the same transport, when simultaneously acting as hosts. When port 2 is a host, port 1 can function as a peripheral. The OTG port can switch its role from host to peripheral, and peripheral to host. The OTG port can become a host through Host Negotiation Protocol (HNP) as specified in *On-The-Go Supplement to the USB Specification Rev. 1.3*.

This document is arranged as follows:

- Section 3 covers guidelines for the ISP1763A acting as a USB host.
- Section 4 covers guidelines for the ISP1763A acting as a USB peripheral.
- Section 5 covers guidelines for the ISP1763A acting as an OTG device.

3 Host programming guide

This section provides software programming guidelines for the ISP1763A host controller.

Overview

Figure 1 shows the ISP1763A host controller in a typical environment with emphases on interfaces.

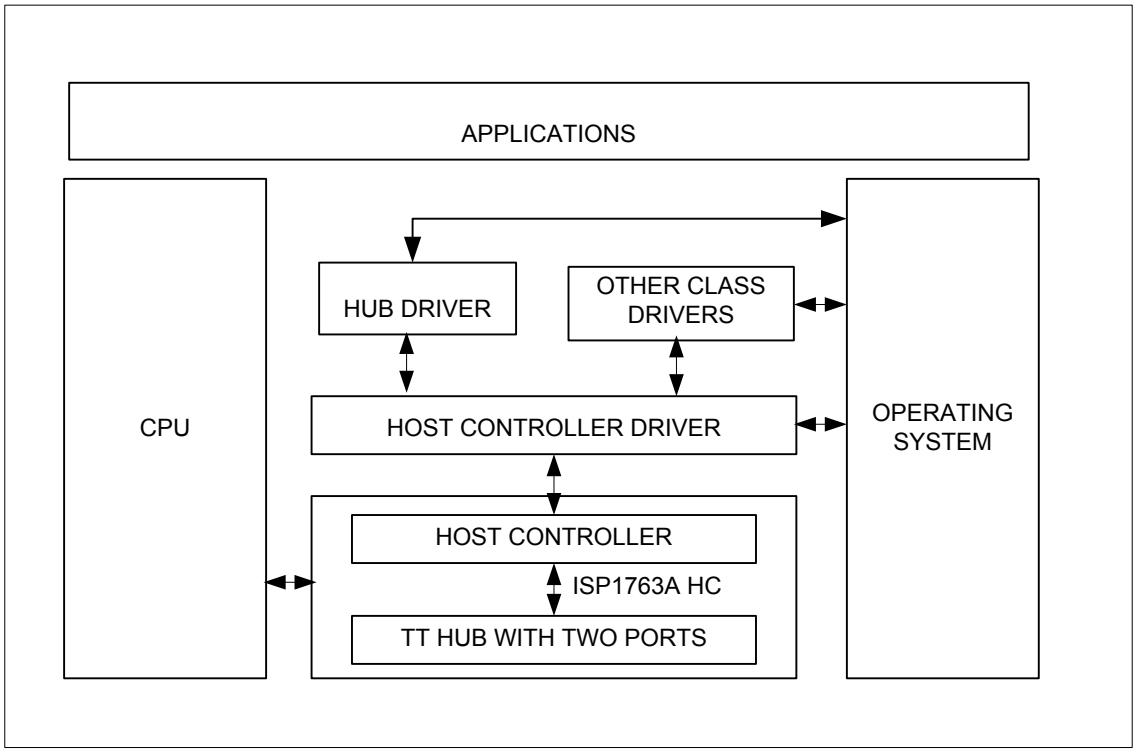


Figure 1 ISP1763A host controller system interface

The ISP1763A host controller is integrated with Transaction Translator (TT) for full-speed and low-speed devices. It does not require the OHCI controller. The ISP1763A host controller supports high-speed, full-speed, and low-speed transactions.

3.1 Controller registers and memory description

The detailed implementation of the ISP1763A host controller is described in the following sections.

- Host controller register description
- Host controller memory description

3.1.1 Host controller register description

The following types of registers are used in the ISP1763A host controller.

- EHCI operational registers
- Configuration registers
- Interrupt registers

3.1.1.1 EHCI operational registers

EHCI operational registers are used for the following operations:

- Reset, run, and stop the host controller
- Configure the root port routing logic
- Manage frame intervals
- Schedule transfer descriptors for control or bulk, interrupt, and Isochronous (ISO) transfers
- Describe the status of the transfer descriptor for control or bulk, interrupt, and ISO transfers

3.1.1.2 Configuration registers

Configurations registers are used for the following operations:

- Configure the data bus width, interrupt polarities, interrupt level, and DMA polarities
- Access memory
- Configure DMA
- Power management operations
- Reset host controller registers

3.1.1.3 Interrupt registers

These registers are used for:

- Status information of various interrupt sources
- Enable host controller interrupts
- Enable the interrupt for transfer descriptor completion

3.1.2 Memory description

The internal addressable host controller buffer memory is 24 kB.

The total amount of memory allocated to the payload determines the maximum transfer size specified by a Proprietary Transfer Descriptor (PTD). A larger internal memory size results in lesser CPU interruption for transfer programming. This means lesser time spent in context switching, resulting in better CPU usage.

A larger buffer also implies a larger amount of data to be transferred. However, this transfer can be done over a longer period of time. To maintain overall system performance, each transfer of the USB data on the USB bus can span up to a few milliseconds before requiring further CPU intervention for data movement. The internal architecture of the ISP1763A host controller allows a flexible definition of the memory buffer for optimization of data transfer on the CPU extension bus and the USB.

It is possible to implement different data transfer schemes, depending on the number and type of USB devices present. For example: push-pull; data can be written to half of the memory while data in the other half is accessed by the host controller and sent on the USB bus. This is useful especially in the case of a high-bandwidth continuous or periodic.

3.1.2.1 Structure of the host controller memory

The internal memory of the host controller is 24 kB.

- 1 kB for the register space
- 3 kB for the PTD area
- 3 kB PTD memory area is further divided into three dedicated areas for each main type of USB transfer
    - ISO PTD: 512 bytes
        - Reserved: 512 bytes
    - INT PTD: 512 bytes
        - Reserved: 512 bytes
    - ATL PTD: 512 bytes
        - Reserved: 512 bytes
- 20 kB for the data payload area

The PTD areas for ISO, INT, and ATL are grouped at the beginning of the memory, occupying address range 0400h to 0FFFh, following the registers' address space.

The payload or data area occupies the next memory address range 1000h to 5FFFh, meaning that 20 kB of memory is allocated for the payload data.

A maximum of 16 PTD areas and their allocated payload areas can be defined for each type of transfer. The structure of a PTD is similar for every transfer type and consists of eight Double Words (DWs) that must be correctly programmed for a correct USB data transfer. The reserved bits of a PTD must be set to logic 0.

The transfer size specified by the PTD determines the contiguous USB data transfer that can be performed without any CPU intervention. The respective payload memory area must be equal to the transfer size defined. The maximum transfer size is flexible and can be optimized, depending on the number and nature of USB devices or PTDs defined and, their respective MaxPacketSize defined.

The RAM is structured in blocks of PTDs and payloads so that while the USB is executing on an active transfer-based PTD, the processor can simultaneously fill up another block area in the RAM. A PTD and its payload can then be updated on-the-fly without stopping or delaying any other USB transaction or corrupting the RAM data.

Some of the design features are:

- The internal memory contains isochronous, interrupt, and asynchronous PTDs, and respective defined payloads.
- Internal memory address range calculation:

  Memory address = (CPU address − 0400h) (shift right >> 3). Base address is 0400h.

Table 2 Memory address

Memory map	CPU address	Memory address	Description
ISO PTD	0400h to 05FFh	0000h to 003Fh	Maximum of 16 PTDs allocated for ISO endpoints. One ISO PTD per endpoint.
INT PTD	0800h to 09FFh	0080h to 00BFh	Maximum of 16 PTDs allocated for INT endpoints. One INT PTD per endpoint.
ATL PTD	0C00h to 0DFFh	0100h to 013Fh	Maximum of 16 PTDs allocated for ATL endpoints. One ATL PTD per endpoint.
Payload	1000h to 5FFFh	0180h to 0B7Fh	20 kB payload memory area is shared among all ISO, INT, and ATL PTDs for data transfers.

3.1.2.2 Memory access in PIO mode

The ISP1763A host controller memory can be accessed in PIO mode using the Memory and Data Port registers. The Memory register (C4h) must be set with the base address before the first memory read or write cycle. Once written, the address will be latched and incremented for every read and write operation until a new address is written to the Memory register (C4h) to change its base address.

Once the base address is specified in the Memory register (C4h), data can be written to or read from the address using the Data Port register (C6h).

For example:

Write operation: Follow these steps to write data to the memory starting from 0400h to 0500h.

1. Configure the bus width to 16-bit or 8-bit using the DATA_BUS_WIDTH bit in the Hardware Mode Control register.
2. Specify the start address as 0400h in the Memory register (C4h).
3. Wait for 100 ns delay between the Memory register (C4h) and the Data Port register (C6h).
4. Write data to the Data Port register (C6h) continuously in a loop for a given data transfer length (256 times in 8-bit mode, 128 times in 16-bit mode). In 16-bit mode, it writes 2-byte data in a single Data Port register (C6h) access. In 8-bit mode, it writes 1-byte data in single Data Port register (C6h) access.

Read operation: Follow these steps to read data from the memory starting from 0400h to 0500h.

1. Configure the bus width to 16-bit or 8-bit using the DATA_BUS_WIDTH bit in the Hardware Mode Control register.
2. Specify the start address as 0400h in the Memory register (C4h).
3. Wait for 100 ns delay between the Memory register (C4h) and the Data Port register (C6h).
4. Read data from the Data Port register (C6h) continuously in a loop for a given data transfer length (256 times in 8-bit mode, 128 times in 16-bit mode). In 16-bit mode, it reads 2-byte data in a single Data Port register (C6h) access. In 8-bit mode, it reads 1-byte data in a single Data Port register (C6h) access.

3.1.2.3 Memory access in DMA mode

The ISP1763A host controller memory can be accessed in DMA mode using the DMA Start Address register (CCh) and the HcDMAConfiguration register (BCh). The DMA Start Address register (CCh) must be set with the base address before the first memory read or write cycle. Once written, the address will be latched and incremented for every read and write operation in DMA mode until a new address is written to the DMA Start Address register (CCh) to change its base address.

Once the base address is specified in the DMA Start Address register (CCh), data can be written to or read from the address using DMA mode through the HcDMAConfiguration register (BCh).

For example:

Write operation: Follow these steps to write data to the memory starting from 0400h to 0500h:

1. Configure the bus width to 16-bit or 8-bit using the DATA_BUS_WIDTH bit in the Hardware Mode Control register. This will set both the DMA and PIO data bus width.
2. Enable GLOBAL_INTR_EN (bit 0), set DREQ_POL (bit 5) and INTR_POL (bit 2) according to the application processor's requirement in the Hardware Mode Control register (B6h).
3. Enable DMAEOTINT_E (bit 3) in the HcInterruptEnable register (D6h).
4. Specify the start address as 0400h in the DMA Start Address register (CCh).
5. Configure processor DMA configuration registers.

6. Specify the source address as the application data buffer (contains data that must be written to the ISP1763A memory 0400h) and the destination address as the Data Port register (C6h).
7. Specify the burst length, data bus width, and transfer length in processor DMA configuration registers.
8. Enable DMA in processor DMA configuration registers.
9. Disable DMA_READ_WRITE_SEL (bit 0) for the write DMA operation, set BURST_LEN[1:0] (bits 3 to 2), enable ENABLE_DMA (bit 1), and update DMA_COUNTER[23:0] (bits 31 to 8) with data length in the HcDMAConfiguration register (BCh).
10. Wait for the DMA EOT interrupt to check if the DMA write transfer is successfully completed.

Read operation: Follow these steps to read data from the memory starting from 0400h to 0500h.

1. Configure the bus width to 16-bit or 8-bit using the DATA_BUS_WIDTH bit in the Hardware Mode Control register. This will set both the DMA and PIO data bus width.
2. Enable GLOBAL_INTR_EN (bit 0), set DREQ_POL (bit 5) and INTR_POL (bit 2) according to application processor's requirements in the Hardware Mode Control register (B6h).
3. Enable DMAEOTINT_E (bit 3) in the HcInterruptEnable register (D6h).
4. Specify the start address as 0400h in the DMA Start Address register (CCh).
5. Configure processor DMA Configuration registers.
6. Specify the source address as the ISP1763A Data Port register (C6h) and the destination address as the application data buffer (place holder for data that is read from the ISP1763A memory address).
7. Specify the burst length, data bus width, and transfer length in processor DMA configuration registers.
8. Enable DMA in processor DMA configuration registers.
9. Enable DMA_READ_WRITE_SEL (bit 0) for the read DMA operation, set BURST_LEN[1:0] (bits 3 to 2), enable ENABLE_DMA (bit 1), and update DMA_COUNTER[23:0] (bits 31 to 8) with data length in the HcDMAConfiguration register (BCh).
10. Wait for the DMA EOT interrupt to check if the DMA write transfer is successfully completed.

3.2 Functional description

The following sections describe the function of the ISP1763A host controller:

- Host controller memory management

- Host controller initialization
- Interrupt handling
- Handling ATL transfer descriptors
- Handling INTL transfer descriptors
- Handling ISO transfer descriptors
- Internal hub enumeration
- Power management

3.2.1 Host controller memory management

The ISP1763A has 24 kB of on-chip memory that must be mapped on the CPU address. This memory is shared between the CPU and the host controller to manage transactions over the USB bus. This memory is double-word aligned and used to perform transactions.

Memory is divided into two parts: one for the PTD header and another for the data payload. The header contains 16 PTDs for each transfer type. The payload contains data to be transferred to or received from the bus. The number of PTDs is limited to 16 PTDs for each type of transfer.

Figure 2 illustrates the usage of the ISP1763A host controller memory.

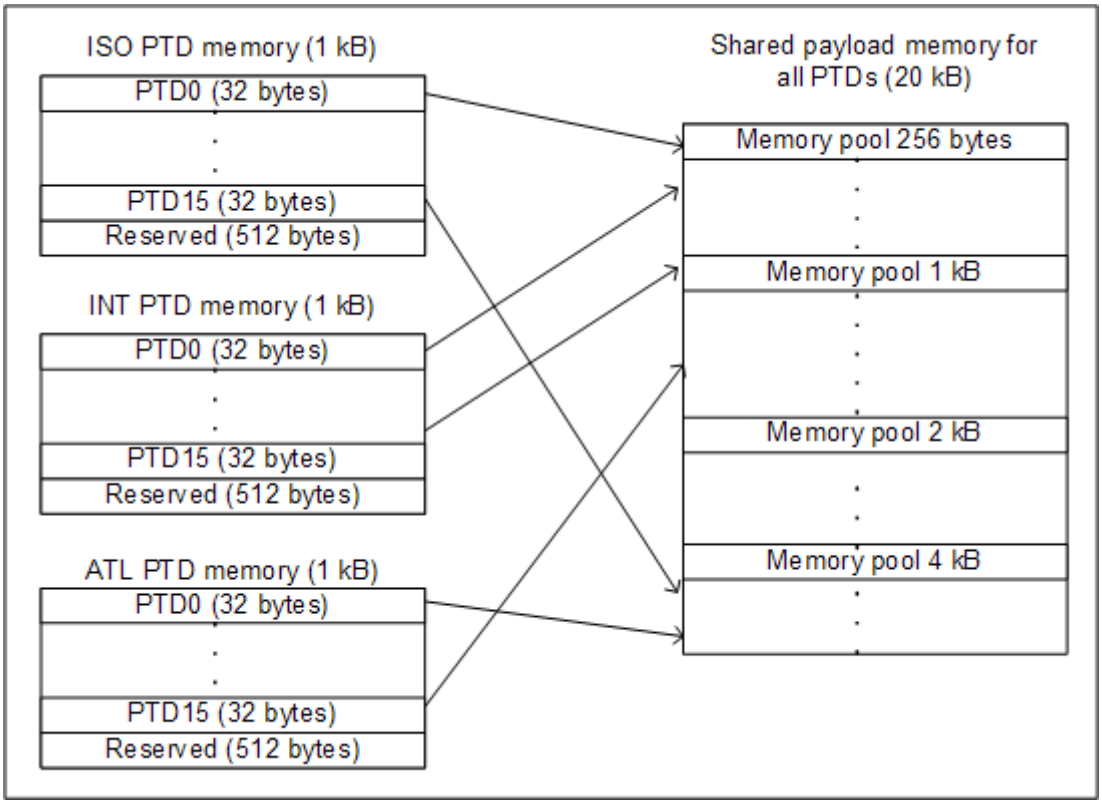


Figure 2 ISP1763A memory management

For each transfer, the payload data memory (20 kB) can be statically or dynamically divided into blocks. The static allocation assigns appropriate blocks when requested by the HCD. The HCD fills up the block by using either the PIO method or the DMA method.

Skip Map register: Each bit in the Skip Map register indicates whether the corresponding PTD is in schedule. Each Skip Map bit is mapped to one of the 16 scheduled PTDs. When a bit in the PTD Skip Map register is set to logic 1 that PTD will be skipped (not scheduled by the controller).

Done Map register: Each bit in the Done Map register indicates the completion status of the corresponding PTD. When the bit is set, it means the transfer is completed and must be removed from the host controller shared memory.

Figure 3 shows the relationship between skip map, done map, and PTD for each transfer type.

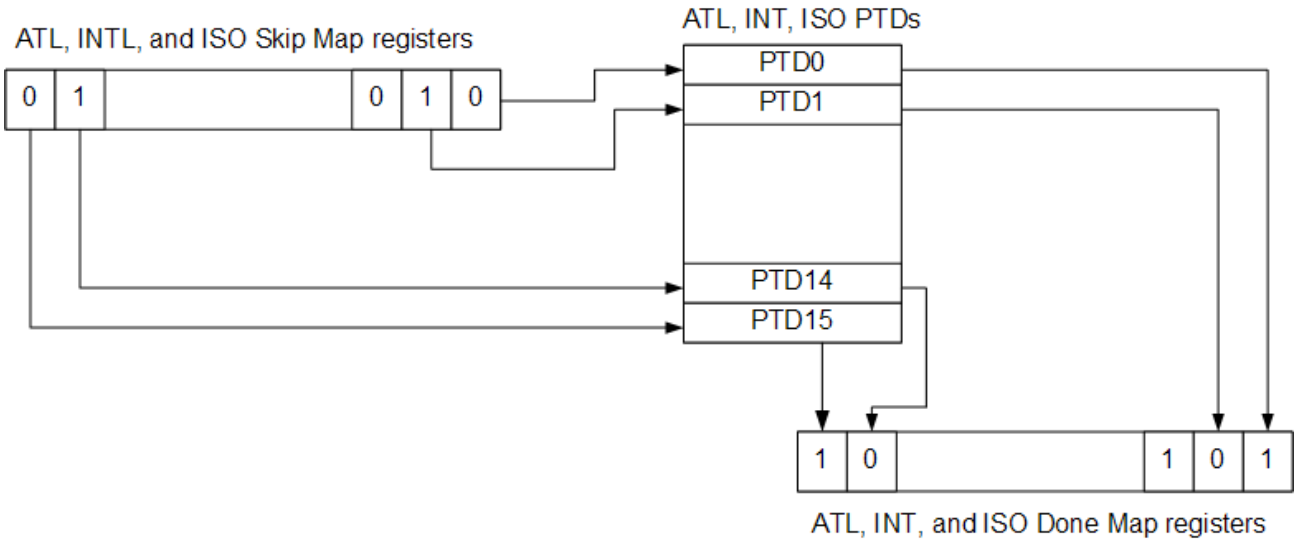


Figure 3 Relationship between skip map, done map, and PTD

3.2.2 Host controller register initialization

The ISP1763A is initialized when the software driver is loading. The following steps must be performed to initialize and put the controller in the operational state.

1. Read the Chip ID register (70h) a few times to stabilize the host controller access.
2. Enable the RESET_ALL bit and write to the SW Reset register (B8h). Read this register until the RESET_ALL bit is cleared by the hardware. Check that all host controller-specific registers and other CPU interface registers are cleared.
3. Enable the RESET_HC bit, write to the SW Reset register (B8h), and read this register until the RESET_HC bit is cleared by the hardware. Check that host controller-specific registers are cleared.
4. Enable ATX_RESET (bit 3), write to the SW Reset register (B8h), and wait for 1 ms. Disable ATX_RESET (bit 3), write to the SW Reset register (B8h), and ensure that the transceiver is reset.
5. Enable RESET_HC (bit 1), write to the USB Command register (8Ch), and read until the RESET_HC bit is cleared.

6. Reset DATA_BUS_WIDTH (bit 4) in the Hardware Mode Control register (B6h) to put the host controller in 16-bit mode or set DATA_BUS_WIDTH to put the host controller in 8-bit mode. Set INTF_LOCK (bit 3) in the Hardware Mode Control register (B6h) to lock the bus interface in the above setting or reset INT_LOCK (bit 3) to unlock the bus interface.

7. Write FFFFh to the ISO, INTL, and ATL PTD Skip Map registers to put host controller transfer descriptors in “no transfer active mode”.

8. Initialize the HcBufferStatus register (BAh) with ATL, INT, and ISO buffer fill. This can also be done during the PTD scheduling. Set ATL_BUF_FILL (bit 0), INT_BUF_FILL (bit 1), or ISO_BUF_FILL (bit 2) after filling the ATL, INT, ISO, or PTD memory. This indicates that ATL, INT, or ISO PTD is filled, and it will be processed by the host controller. Clear ATL_BUF_FILL (bit 0), INT_BUF_FILL (bit 1), or ISO_BUF_FILL (bit 2) in the HcBufferStatus register. It indicates that ATL, INT, or ISO PTD is not filled, and ATL, INT, or ISO PTD area will not be processed by the host controller.

9. Clear all the interrupt sources in the HcInterrupt register (D4h) by writing logic 1 to the corresponding interrupt sources.

10. Enable the required interrupts (ISO, ATL, INTL, SOF, DMA_ EOT, clock ready, and OPR) by writing to the HcInterruptEnable register (D6h).

    - Setting MSOFINT_E (bit 0) indicates that the SOF interrupt is enabled.
    - Setting DMAEOTINT_E (bit 3) indicates that the DMA completion interrupt is enabled.
    - Setting OPR_REG_E (bit 4) indicates that the OPR interrupt is enabled. It controls the generation of interrupt when there is a change in operational registers. It is mainly used for resume detection after recovering from the host controller suspend.
    - Setting HCSUSP_E (bit 5) indicates that the interrupt will be generated when the host controller enters suspend mode.
    - Setting CLKREADY_E (bit 6) indicates that the interrupt will be generated when host controller internal clock signals are running stable. It is useful after wake up.
    - Setting INT_IRQ_E (bit 7), ATL_IRQ_E (bit 8), or ISO_IRQ_E (bit 9) indicates that interrupt will be generated when one or more INT, ATL, or ISO PTDs matching INT, ATL, or ISO IRQ Mask AND or INT, ATL, or ISO Mask OR register bit combinations are completed.

11. Setting OTG_IRQ_E (bit 10) indicates that the interrupt will be generated when there is a change in the OTG Interrupt Latch register.

12. Set OTG_DISABLE (bit 10) by writing to the OTG Control register (set: E4h) and clear SW_SLE_HC_DC (bit 7) by writing to the OTG Control register (clear: E6h) to configure PORT1 in host mode.

13. Clear HC_2_DIS (bit 15) in the OTG Control register (clear: E6h) to configure PORT2 in host mode.

14. Set the ATL IRQ Mask OR register (DCh), the INT IRQ Mask OR register (DAh), and the ISO IRQ Mask OR register (D8h) to FFFFh to indicate that interrupts will be generated when any of the ATL, INT, or ISO PTDs are completed. You can set these registers on demand basis just before enabling the PTD.

15. Set the ATL IRQ Mask AND register (E2h), the INT IRQ Mask AND register (E0h), and the ISO IRQ Mask AND register (DEh) to 0000h to indicate that there is no AND condition set between any of the ATL, INT, or ISO PTDs.
    - If the ATL, INT, or ISO interrupts to be generated after completion of two or more ATL, INT, or ISO PTDs are completed, then set an AND condition in these ATL, INT, or ISO IRQ Mask AND registers. Set these registers on demand basis just before enabling the PTD.
16. Configure the following bits in the Hardware Mode Control register (B6h).
    - Set MSOFINT_E (bit 0) to enable the interrupt generation. Interrupts will be asserted according to the HcInterruptEnable register (D6h).
    - Set DREQ_POL (bit 5) if the host controller is to be in DMA mode.
    - Set INTR_LEVEL (bit 1) if the host controller is to be in edge interrupt mode. Reset INTR_LEVEL (bit 1) if the host controller is to be in level interrupt mode.
    - Set INTR_POL (bit 2) if the interrupt polarity is to be active HIGH. Reset this bit if the interrupt polarity is to be active LOW.
17. Initialize the following registers to control the PTD processing.
    - Initialize the ISO PTD LAST PTD register (A8h) with 1h to state that PTDs corresponding to bit 0 are the last PTDs to schedule among all ISO PTD list.
    - Initialize the INT PTD LAST PTD register (AEh) with 8000h to state that PTDs corresponding to bit 15 are the last PTDs to schedule among all INT PTD list.
    - Initialize the ATL PTD LAST PTD register (B4h) with 8000h to state that PTDs corresponding to bit 15 are the last PTDs to schedule among all ATL PTD list.
18. Run the host controller by enabling bit 1 of the USB Command register (8Ch).
19. Set Configure Flag (bit 0) of the Configure Flag register (9Ch) to put the host controller in configured mode.
20. Enable Port Power (bit 12) in the PORTSC1 register (A0h), and wait for 50 ms.
21. After 50 ms, read the PORTSC1 register (A0h) to ensure that Line Status (bits 11 to 10) is 2h, and Port Power (bit 12), Current Connect Status (bit 0), and Current Status Change (bit 1) are set.
22. PORTSC1 (A0) must reflect connection status to indicate that the internal hub is connected.

3.2.3 Interrupt handling

The ISP1763A host controller transfers can be handled in two modes.

- Millisecond SOF interrupt mode
- ATL, INT, or ISO interrupt mode

3.2.3.1 SOF interrupt mode

The following steps describe transfer descriptors handling in SOF mode.

1. Ensure that millisecond based SOF (bit 0 of the HcInterruptEnable register D6h), interrupt, and global interrupt are enabled during the host controller initialization. For details, see the host controller register initialization.
2. If the millisecond based SOF interrupt in the HcInterruptEnable register (D6h) is enabled, interrupts will start coming for millisecond.

The flow chart in Figure 4 describes the SOF interrupt handling.

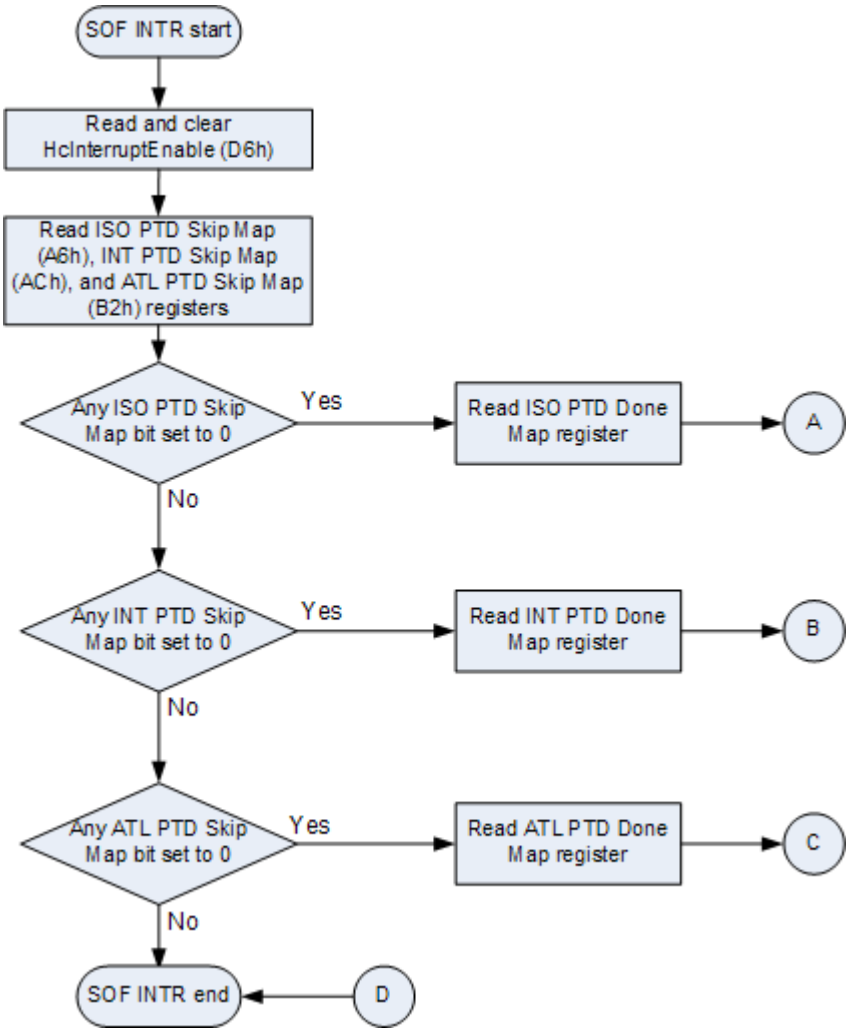


Figure 4 SOF interrupt handling

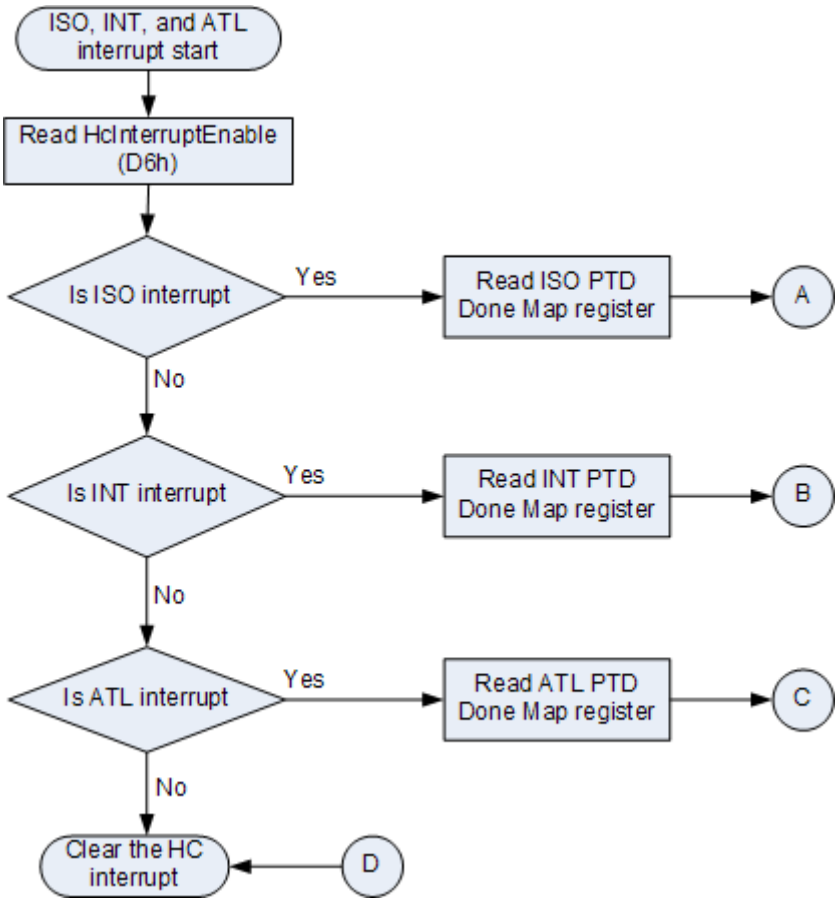


Figure 5 ATL, ISO, or INT completed based interrupt handling

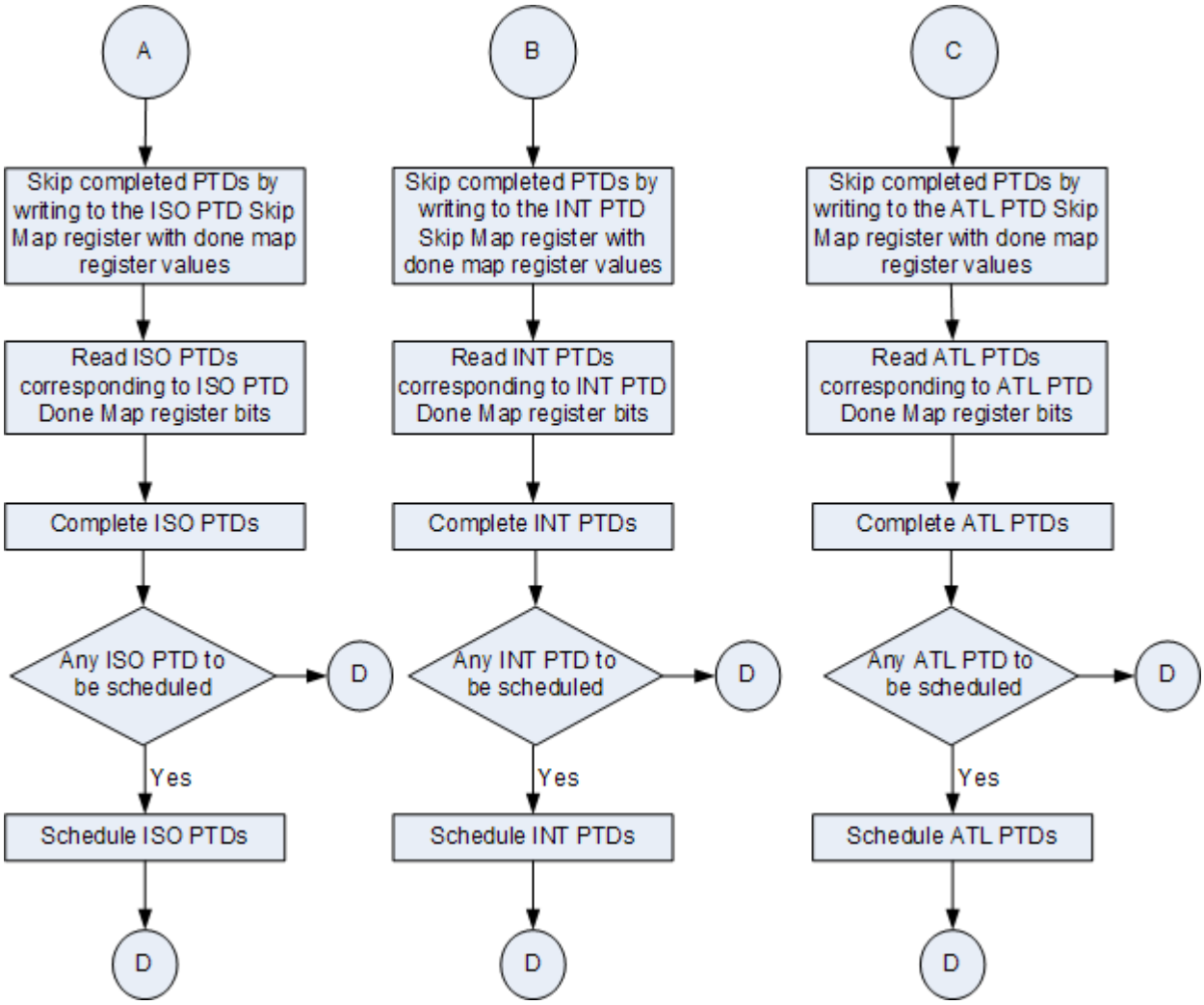


Figure 6 ATL, INT, or ISO interrupt handling

3.2.3.2 ATL, INT, and ISO complete interrupt modes

The following steps describe transfer descriptors handling in ATL, INT, and ISO complete mode.

1. Ensure that ATL, INT, and ISO (bits 7, 8, and 9 of the HcInterruptEnable register D6h) interrupt and global interrupts are enabled during the host controller initialization. For details, see host controller register initialization.
2. If the ATL, INT, and ISO interrupts in the HcInterruptEnable register (D6h) are enabled, interrupts will start coming as and when the ATL, INT, or ISO PTD retires.
3. Follow these steps to carry out the ATL, INT, or SOF interrupt handling.
   a. Read the HcInterruptEnable register (D6h).
   b. If it is not zero, check the ATL, INT, and ISO interrupt bits respectively to see which bits are set.
   c. If these bits are set, read the corresponding done map register and store the value somewhere as these registers are read to clear.

d. Write back the value that was read out from the HcInterrupt register to clear the HC interrupt.

4. Process the various types of completed PTDs based on the Done Map register.

3.2.4 ATL transfer descriptor handling

Ensure that the ISP1763A host controller is initialized as specified in Section 3.2.2.

Ensure that required interrupts are enabled and handled as specified in Section 3.2.3.

ATL transfer descriptors can handle the control and bulk transfers.

ATL transfer descriptors can be programmed to be in split transaction mode and high-speed transaction mode in both control and bulk transfers.

Before scheduling bulk transfers, ensure that the internal hub is enumerated as specified in Section 3.2.7.

3.2.4.1 Scheduling high-speed bulk IN and OUT PTDs

The following sequence describes the scheduling of high-speed bulk IN and OUT ATL PTDs.

1. For OUT transfers, check for the available or unused payload memory in the payload memory address range (1000h to 5FFFh). Fill the payload data to be transferred on the USB bus into the payload memory address in the available memory location. For details on the memory management, see Section 3.2.1.
2. If data to be transferred is more than the available payload memory, then split transfers into multiple transfers, and schedule transfer descriptors one after the other.
3. Fill following double words in any of the 16 ATL PTD memory areas ranging from C00h to DFFh. For details, see Section 3.1.2.1.

3.2.4.1.1 DW0

1. Set Valid (bit 0) of DW0 to logic 1 to indicate that the current PTD is active.
2. Specify NrBytesToTransfer[14:0] (bits 17 to 3) to indicate to the host controller scheduler that this PTD can transfer as specified in this field. If this PTD is programmed to be an OUT PTD, it tells that this PTD can transfer data from the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). If this PTD is programmed to be an IN PTD, it tells that this PTD can receive data into the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). This PTD can transfer data up to 20 kB.
3. Set the maximum packet length in bits 28 to 18 of DW0.
4. Set the endpoint number in bit 31 of DW0 and bits 2 to 0 of DW1. For control transfer, set this endpoint number to 0.

3.2.4.1.2 DW1

1. Set DeviceAddress[6:0] (bits 9 to 3) of DW1. This is the USB address of the function containing the endpoint specified in the endpoint number field.

2. Set Token[1:0] (bits 11 to 10) of DW1. This token is Packet Identifier (PID) for this transaction.
   - 00 – OUT – If the PTD is OUT PTD (control or bulk).
   - 01 – IN – If the PTD is IN PTD (control or bulk).
   - 10 – SETUP – If the PTD is set-up PTD.
   - 11 – PING – Written by the hardware, depending on bit 26 setting of DW3.
3. Specify EPType[1:0] (bits 13 to 12) of DW1.
   - Write 00 for control transfer.
   - Write 10 for bulk transfer.
4. Set the speed of the transaction in S (bit 14) of DW1.
   - 0 for high-speed transaction (control or bulk).
   - 1 for split transaction (control or bulk). Not applicable for high transfers.

3.2.4.1.3 DW2

1. Specify DataStartAddress[15:0] (bits 23 to 8) of DW2. This is the start address of the actual payload data that will be sent on or received from the USB bus.
   - This is the internal memory address and not the direct CPU address.
   - Internal RAM address = (CPU address – 0 x 400) / 8.
2. Set Reload (RL[3:0]) counter value in bits 28 to 25 of DW2.
   - If RL[3:0] is specified as a non-zero value, the software writes some value to NAK for the initial PTD launch. The software gets the PTD completion interrupt when the NAK counter decrements to 0 and Valid will be reset. The software must read the PTD, set the Valid bit and the NAK counter value, and reload the PTD.
   - If RL[3:0] is specified as 0, the hardware ignores the NAK counter value. NAK retry will be done automatically in the hardware without any software intervention.
   - RL[3:0] and NakCnt[3:0] are set to the same value before a transaction.

3.2.4.1.4 DW3

1. Specify NrBytesTransferred[14:0] (bits 14 to 0) of DW3 to 0. This field is updated by the hardware after completing the scheduled PTD. For successfully completed PTD, this field must match bits 14 to 0 of DW0.
2. Set the NAK counter value in bits 22 to 19 of DW3.
   - If RL[3:0] (bits 28 to 25 of DW2) is specified as a non-zero value, the software writes some value to the NAK for the initial PTD launch. The software gets the PTD completion interrupt when the NAK counter decrements to 0 and Valid will be reset. The software must read the PTD, set the Valid bit, set the NAK counter value, and reload the PTD.

- If RL[3:0] is specified as 0, the hardware ignores the NAK count value. The NAK retry will be automatically done in the hardware without any software intervention.
- RL[3:0] and NakCnt[3:0] are set to the same value before a transaction.

3. Set Cerr[1:0] (error counter) bits 24 to 23 of DW3 to 3h. The transaction will be retried three times before the transaction error (X, bit 28 of DW3) will be set.
4. Set Data Toggle (D, bit 25 of DW3) to start a PTD.
   - This field is set by the software before launching a PTD. The hardware updates after successful completion of the PTD.
   - This field setting depends on the previously completed PTD data toggle value for the same endpoint. If the previous successfully completed PTD data toggle value is 1 (updated by the hardware after successful completion of the PTD) for the same endpoint, set the same data toggle for the current PTD of the same endpoint.
   - If it is the first PTD to be scheduled for a particular endpoint, set this to 0 before launching.
5. For high-speed transactions, set Ping (bit 26 of DW3).
   - For the first time, the software sets the Ping bit to 0, and the hardware updates it.
   - For the successive PTD of the same endpoint, the software sets the bit based on the state of the bit for the previously completed PTD.
6. Transaction error (X, bit 28 of DW3) will be set by the hardware after the transaction retries three times when it detects any PID error.
   - If there are PID errors, this bit will be set to active, and Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive after the transaction retries three times.
7. The babble error (B, bit 29 of DW3) will be set by the hardware when it detects any babble error.
   - When the babble error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
8. The halt error (H, bit 30 of DW3) will be set by the hardware when it detects any endpoint stall or any serious error in the status of the transfer descriptor.
   - When the halt error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
9. Set Active (bit 31 of DW3) to the same value as Valid (bit 0 of DW0) before the PTD launch.
   - Active will be reset by the hardware only when there is an error, or when the PTD successfully completes the transfer.

   Note: *DW4, DW5, DW6, DW7, and all the bits not specified in DW0, DW1, DW2, and DW3 are reserved, and set to 0 before the PTD launch.*
10. Ensure that bit 0 (ATL_BUFF_FILL) of the HcBufferStatus register (BAh) is filled up if not filled so.

11. Ensure that ATL IRQ Mask OR register (DCh) bit fields are updated corresponding to the ATL PTD memory location.

12. If you want to define any AND condition between ATL PTDs, update the ATL IRQ Mask AND register (E2h).

13. Each bit in the ATL PTD Skip Map register indicates whether the corresponding ATL PTD is in the schedule or not. Each ATL PTD Skip Map bit is mapped to one of the 16 ATL PTDs. When a bit in the PTD skip map is set to logic 1 that PTD will be skipped, that is, to be scheduled by the host controller hardware. So read the ATL PTD Skip Map register (B2h) and reset the Skip bit corresponding to the ATL PTD memory location and write to the ATL PTD Skip Map register (B2h). After this, PTD will be executed by the host controller hardware.

3.2.4.2 Scheduling split bulk IN and OUT PTDs

The following sequence describes the scheduling of start and complete full-speed or low-speed bulk IN and OUT ATL PTDs.

1. For OUT transfers, check for the available or unused payload memory in the payload memory address range (1000h to 5FFFh). Fill the payload data to be transferred on the USB bus in the payload memory address in the available memory location. For details on the memory management, see Section 3.2.1.

2. If data to be transferred is more than the available payload memory, then split transfers into multiple transfers and schedule transfer descriptors one after another.

3. Fill following double words into any of the 16 ATL PTD memory areas ranging from C00h to DFFh. For details, see Section 3.1.1.

3.2.4.2.1 DW0

1. Set Valid (bit 0) of DW0 to logic 1 to indicate that the current PTD is active.

2. Specify NrBytesToTransfer[14:0] (bits 17 to 3) to indicate to the host controller scheduler that this PTD can transfer as specified in this field. If this PTD is programmed to be an OUT PTD, it states that this PTD can transfer data from the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). If this PTD is programmed to be an IN, it states that this PTD can receive data into the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). This PTD can transfer data up to 20 kB.

3. Set the maximum packet length in bits 28 to 18 of DW0.

4. Set the endpoint number in bit 31 of DW0 and bits 2 to 0 of DW1. For control transfer, set this endpoint number to 0.

3.2.4.2.2 DW1

1. Set DeviceAddress[6:0] (bits 9 to 3) of DW1. This is the USB address of the function containing the endpoint specified in the endpoint number field.

2. Set Token Type (bits 11 to 10) of DW1. This token is PID for this transaction.

   - 00 – OUT – If PTD is OUT (control or bulk) PTD.
   - 01 – IN – If PTD is IN (control or bulk) PTD.

- 10 – SETUP – If PTD is set-up PTD.
- 11 – PING – Written by the hardware; depends on bit 26 setting of DW3.

3. Specify Endpoint Type (bits 13 to 12) of DW1.
   - Write 00 for control transfer.
   - Write 10 for bulk transfer.
4. Set the speed of the transaction in bit 14 of DW1.
   - 0 for high-speed transaction (control or bulk)
   - 1 for split transaction (control or bulk).
5. Set the SE[1:0] field (bits 17 to 16 of DW1). Depends on the endpoint type and direction.
   - SE[1:0] = 10 means low-speed
   - SE[1:0] = 00 means full-speed
6. Set PortNumber[6:0] in bits 24 to 18 of DW1. This indicates the port number of the hub or embedded TT.
7. Set HubAddress[6:0] in bits 31 to 24 of DW1. This indicates the hub address.

3.2.4.2.3 DW2

1. Specify DataStartAddress[15:0] (bits 23 to 8) of DW2. This is the start address of the actual payload data that will be sent on or received from the USB bus.
   - This is the internal memory address and not the direct CPU address.
   - Internal RAM address = (CPU address – 400h) / 8.
2. Set the reload counter value (RL[3:0]) in bits 28 to 25 of DW2. For full-speed and low-speed transaction, set RL[3:0] to 0.

3.2.4.2.4 DW3

1. Specify NrBytesTransferred[14:0] of DW3 to 0. This field is updated by the hardware after the completion of the scheduled PTD. For successfully completed PTD, this field must match bits 14 to 0 of DW0.
2. Set the reload (NakCnt[3:0]) counter value in bits 22 to 19 of DW3. For full-speed and-low speed transactions, set the NAK value to 0. RL[3:0] and NakCnt[3:0] are set to the same value before a transaction.
3. Specify Cerr[1:0] (Error Counter, bits 24 to 23) of DW3. Set the error counter to 3h. The transaction will be retried three times before transaction error (X, bit 28 of DW3) will be set.
4. Set Data Toggle (DT, bit 25 of DW3) to start a PTD.
   - This field is set by the software before launching a PTD and the hardware updates after successful completion of the PTD.

- This field setting depends on the previously completed PTD data toggle value for the same endpoint. If the previous successfully completed PTD data toggle value is 1 (updated by the hardware after successful completion of the PTD) for the same endpoint, set the same data toggle for the current PTD of the same endpoint.
- If it is the first PTD to be scheduled for a particular endpoint, set this to 0 before launching.

5. Set SC (start and complete split, bit 27 of DW3) to 0 before the PTD launch. For the first time the software sets the SC bit to 0 and the hardware updates it.
6. Transaction error (bit 28 of DW3) will be set by the hardware after the transaction is retried three times when it detects any PID error.
   - If there are PID errors, this bit is set to active and Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive after the transaction is retried three times.
7. The babble error (B, bit 29 of DW3) will be set by the hardware when it detects any babble error.
   - When the babble error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
8. The halt error (H, bit 30 of DW3) will be set by the hardware when it detects any endpoint stall or any serious error in the status of the transfer descriptor.
   - When the halt error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
9. Set Active (bit 31 of DW3) to the same value as Valid (bit 0 of DW0) before the PTD launch.
   - Active will be reset by the hardware only when there is an error (H, B, and X) or when the PTD successfully completes the transfer.

   Note: DW4, DW5, DW6, DW7, and all the bits not specified in DW0, DW1, DW2, and DW3 are reserved, and set to 0 before the PTD launch.
10. Ensure that bit 0 (ATL_BUFF_FILL) of the HcBufferStatus register (BAh) is filled up if it is not filled.
11. Ensure that ATL IRQ Mask OR register (DCh) bit fields are updated corresponding to the ATL PTD memory location.
12. If you would like to define any AND condition between ATL PTDs, update the ATL IRQ Mask AND register (E2h).
13. Each bit in the ATL PTD Skip Map register indicates whether the corresponding ATL PTD is in schedule or not. Each ATL PTD Skip Map bit is mapped to one of the 16 ATL PTDs. When a bit in the PTD Skip Map is set to logic 1 that PTD will be skipped, that is, to be scheduled by the host controller hardware. So read the ATL PTD Skip Map register (B2h) and reset the skip bit corresponding to the ATL PTD memory location and write to the ATL PTD Skip Map register (B2h). After this, PTD will be executed by the host controller hardware.

3.2.4.3 Completion of high-speed and split-speed ATL PTD handling

The following sequence describes the completion handling of high-speed and split-speed bulk IN and OUT ATL PTDs.

1. After scheduling the ATL PTD as described in Section 3.2.4.1 and Section 3.2.4.2, wait for the SOF or ATL interrupt as described in Section 3.2.3.
2. After the done map bit corresponding to the scheduled ATL PTD is set in the ATL PTD Done Map register, it indicates that the PTD is retired and needs to be read.
3. Read the retired ATL PTD and check the following for the completion status in sequence.

a. Check Halt (H, bit 30 of DW3), Babble (B, bit 29 of DW3), and Error (X, bit 28 of DW3). If any of the H, B, or X is set, the PTD is completed with endpoint halt or any other serious error in transfer. So abort all the successive transfer descriptors in the transfer for the same endpoint and inform the application with error status (along with the previous successfully completed transfer descriptor status in the same transfer for the same endpoint, if any).

b. If the preceding condition is not met, check if Valid (bit 0 of DW0) and Active (bit 31 of DW3) are 0, and compare NrBytesToTransfer[14:0] (bits 17 to 3 of DW0) and NrBytesTransferred[14:0] (bits 13 to 0 of DW3).

   i. If both are matching, check bits 11 to 10 of DW1.

   ii. If it is the IN transfer type, read the payload data from the payload address specified in bits 23 to 8 of DW2.

   iii. If it is the last PTD in the current transfer list, inform the application with status.

c. If the preceding condition is not met, check if Valid (bit 0 of DW0) is set to 0 and Active (bit 31 of DW3) is set 1. Do the following:

   i. Read the NakCnt[3:0] value and if it is set to 0, refill NAK with the RL[3:0] value.

   ii. Update Active and Valid to 1.

   iii. Reschedule the PTD.

d. If the preceding condition is not met, check if Valid (bit 0 of DW0) and Active (bit 31 of DW3) are 0. Compare NrBytesToTransfer[14:0] (bits 17 to 3 of DW0) and NrBytesTransferred[14:0] (bits 13 to 0 of DW3).

   i. If both are not matching, it means it can be a short transfer. Check bits 11 to 10 of DW1. If it is an IN transfer type, read the payload data from the payload address specified in bits 23 to 8 of DW2.

   ii. If it is the last PTD in the current transfer list, inform the application with status.

3.2.5 INT transfer descriptor handling

Ensure that the ISP1763A host controller is initialized as specified in Section 3.2.2.

Ensure that required interrupts are enabled and handled as specified in Section 3.2.3.

3.2.5.1 Scheduling high-speed interrupt IN and OUT PTDs

The following sequence describes the scheduling of high-speed interrupt IN and OUT PTDs.

1. For OUT transfers, check for the available or unused payload memory in the payload memory address range (1000h to 5FFFh). Fill the payload data to be transferred on the USB bus into the payload memory address in the available memory location. For details on the memory management, see Section 3.2.1.
2. If data to be transferred is more than the available payload memory, split transfers into multiple transfers and schedule transfer descriptors one after another.
3. Fill the following double words into any of the 16 INTL PTD memory areas ranging from 800h to 9FFh. For details, see Section 3.1.1.

3.2.5.1.1 DW0

1. Set Valid (bit 0) of DW0 to 1 to indicate that the current PTD is active.
2. Specify NrBytesToTransfer[14:0] (bits 17 to 3) of DW0 to indicate to the host controller scheduler that this PTD can transfer as specified in this field. If this PTD is programmed to be an OUT PTD, it specifies that this PTD can transfer data from the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). If this PTD is programmed to be an IN PTD, it specifies that this PTD can receive data into the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). This PTD can transfer data up to 20 kB. Set the maximum packet length in bits 28 to 18 of DW0.
4. Set the multiplier counter value in bits 30 to 29 of DW0.
   - The multiplier counter value informs the host controller about the number of successive packets the host controller may submit to the endpoint in the current execution (in the current microframe).
   - Setting this field to 3h means that the host controller can submit three maximum packets in a single microframe.
   - Allowed values are 1h, 2h, and 3h.
5. Set Endpoint Number (bit 31 of DW0) and bits 2 to 0 of DW1.

3.2.5.1.2 DW1

1. Set Device Address (bits 9 to 3) of DW1. This is the USB address of the function containing the endpoint specified in the endpoint number field.
2. Set Token Type (bits 11 to 10) of DW1. This token is the PID for this transaction.
   - 00 – OUT – If the PTD is an OUT PTD.
   - 01 – IN – If the PTD is an IN PTD.
3. Specify Endpoint Type (bits 13 to 12) of DW1.
   - Write 11 for interrupt transfer
4. Set the speed of the transaction in bit 14 of DW1.
   - 0 for high-speed transaction
   - 1 for split transaction.

3.2.5.1.3 DW2

1. If the polling interval is in milli-SOFs, use µFrame[4:0] (bits 7 to 3) of DW2 along with µSA[7:0] (bits 7 to 0) of DW4.

   According to *Universal Serial Bus Specification Rev. 2.0*, the INT polling rate is defined as $2^{(b-1)}$ µSOF, where b is 1 to 9.

   When b is 4, 5, 6, 7, 8, or 9, use µFrame[4:0] of DW2 to define polling. This is because the rate is in milliseconds. Also set any of µSA[7:0] of DW4 to 1.

   Table 3 defines values when the polling interval is in milliseconds.

   Note: If the polling rate is in µSOFs, see µSA[7:0] of DW4.

Table 3 Polling interval in milliseconds

b	Rate	µFrame[4:0]	µSA[7:0]
4	1 ms	0 0000	Any 1 bit set
5	2 ms	0 0001	Any 1 bit set
6	4 ms	0 0010 to 0 0011	Any 1 bit set
7	8 ms	0 0100 to 0 0111	Any 1 bit set
8	16 ms	0 1000 to 0 1111	Any 1 bit set
9	32 ms	1 0000 to 1 1111	Any 1 bit set

2. Specify Data Start Address (bits 23 to 8) of DW2. This is the start address of the actual payload data that will be sent on or received from the USB bus.

This is the internal memory address and not the direct CPU address.

Internal RAM address = (CPU address – 0 x 400) / 8.

3.2.5.1.4 DW3

1. Specify NrBytesTransferred[14:0] (bits 14 to 0) of DW3 to 0. This field is updated by the hardware after scheduled PTD completion. For successfully completed PTD, this field must match bits 14 to 0 of DW0.
2. Specify Error Counter (Cerr[1:0], bits 24 to 23) of DW3. Set Error Counter to 3h. The transaction will be retried three times before transaction error (X, bit 28 of DW3) will be set.
3. Set Data Toggle (DT, bit 25) of DW3 to start a PTD.
   - This field is set by the software before launching a PTD. The software writes the current transaction toggle value. The hardware updates after successful completion of PTD.
   - This field setting depends on the previously completed PTD data toggle value for the same endpoint. If the previous successfully completed PTD data toggle value is 1 (updated by the hardware after successful completion of the PTD) for the same endpoint, set the same data toggle for the current PTD of the same endpoint.
   - If it is the first PTD to be scheduled for a particular endpoint, set this to 0 before launching.

4. The halt error (H, bit 30 of DW3) will be set by the hardware when it detects any endpoint stall or any serious errors in the status of the transfer descriptor.
   - When the halt error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
5. Set Active (bit 31 of DW3) to the same value as Valid (bit 0 of DW0) before the PTD launch.
   - Active will be reset by the hardware only when there is an error (H, B, or X) or when the PTD successfully completes the transfer.

3.2.5.1.5 DW4

1. If the polling interval is in µSOFs, use µSA[7:0] (bits 7 to 0) of DW4 along with µFrame[4:0] (bits 7 to 3) of DW2.

   According to *Universal Serial Bus Specification Rev. 2.0*, the INT polling rate is defined as $2^{(b-1)}$ µSOF, where b is 1 to 9.

   When b is 1, 2, or 3, use µSA[7:0] of DW4 to define the polling because the rate is in µSOFs. Also set µFrame[4:0] of DW2 to 0.

   Table 4 defines values when the polling interval is in µSOFs.

   Note: If the polling rate is in milli-SOFs, see µFrame[4:0] (bits 7 to 3) of DW2.

Table 4 Polling interval in µSOF

b	Rate	µFrame[4:0]	µSA[7:0]
1	1 µSOF	0 0000	1111 1111
2	2 µSOF	0 0000	1010 1010 or 0101 0101
3	4 µSOF	0 0000	Any two bits set

2. When Frame Number (bits 7 to 3 of DW2) matches the frame number of the USB bus, these bits are checked for 1 before they are sent for µSOF.

   For example: When µSA[7:0] = 1, 1, 1, 1, 1, 1, 1, 1: send INT for every µSOF of the entire millisecond. When µSA[7:0] = 0, 1, 0, 1,0, 1, 0, 1: send INT for µSOF0, µSOF2, µSOF4, and µSOF6. When µSA[7:0] = 1, 0, 0, 0, 1, 0, 0, 0 : send INT for every fourth µSOF.

   ***Note**: DW5, DW6, DW7, and all the bits not specified in DW0, DW1, DW2, DW3, and DW4 are reserved, and set to 0 before the PTD launch.*
3. Ensure that bit 1(INTL_BUFF_FILL) of the HcBufferStatus register (BAh) is filled up if not filled so.
4. Ensure that INTL IRQ Mask OR register (DAh) bit fields are updated corresponding to the INTL PTD memory location.
5. If you would like to define any AND condition between INTL PTDs, update the INTL IRQ Mask AND register (E0h).

6. Each bit in the INTL PTD Skip Map register indicates whether the corresponding INTL PTD is in the schedule or not. Each INTL PTD Skip Map bit is mapped to one of the 16 INTL PTDs. When a bit in the PTD skip map is set to logic 1 that PTD will be skipped, that is, to be scheduled by the host controller hardware. So, read the INTL PTD Skip Map register (ACh) and reset the skip bit corresponding to the INTL PTD memory location and write to the INTL PTD Skip Map register (ACh). After this, PTD will be executed by the host controller hardware.

3.2.5.2 Scheduling split interrupt IN and OUT PTDs

The following sequence describes the scheduling of full-speed or low-speed interrupt IN and OUT PTDs.

1. For OUT transfers, check for the available or unused payload memory in the payload memory address range (1000h to 5FFFh). Fill the payload data to be transferred on the USB bus into the payload memory address in the available memory location. For details on the memory management, see Section 3.2.1.
2. If data to be transferred is more than the available payload memory, then split transfers into multiple transfers and schedule transfer descriptors one after another.
3. Fill the following double words into any of the 16 INTL PTD memory areas ranging from 800h to 9FFh. For details, see Section 3.1.1.

3.2.5.2.1 DW0

1. Set Valid (bit 0) of DW0 to 1 to indicate that the current PTD is active.
2. Specify NrBytesToTransfer[14:0] (bits 14 to 3) to indicate to the host controller scheduler that this PTD can transfer as specified in this field. If this PTD is programmed to be an OUT PTD, it specifies that this PTD can transfer data from the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). If this PTD is programmed to be an IN PTD, it specifies that this PTD can receive data into the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). This PTD can transfer data up to 4 kB.
3. Set the maximum packet length in bits 28 to 18 of DW0.
4. Set the endpoint number in bit 31 of DW0 and bits 2 to 0 of DW1.

3.2.5.2.2 DW1

1. Set Device Address (bits 9 to 3) of DW1. This is the USB address of the function containing the endpoint specified in the endpoint number field.
2. Set Token Type (bits 11 to 10) of DW1. This token is the PID for this transaction.
   - 00 – OUT – If the PTD is an OUT PTD.
   - 01 – IN – If the PTD is an IN PTD.
3. Specify Endpoint Type (bits 13 to 12) of DW1.
   - Write 11 for the interrupt transfer
4. Set the speed of the transaction in bit 14 of DW1.
   - 0 for high-speed transaction

- 1 for split transaction.

5. Set SE[1:0] (bits 17 to 16 of DW1), depending on the endpoint type and direction.
   - SE[1:0] = 10 means low-speed.
   - SE[1:0] = 00 means full-speed.
6. Set the port number in bits 24 to 18 of DW1. This indicates the port number of the hub or embedded TT.
7. Set the hub address in bits 31 to 24 of DW1. This indicates the hub address.

3.2.5.2.3 DW2

1. The polling interval is in milli-SOFs. Use μFrame[4:0] (bits 7 to 3) of DW2.

   According to *Universal Serial Bus Specification Rev. 2.0*, the INT polling rate is defined as $2^{(b-1)}$ μSOF, where b is 4 to 9.

   When b is 4, 5, 6, 7, 8, or 9, use μFrame[4:0] of DW2 to define the polling because the rate is in milliseconds.

   Table 5 defines values when the polling interval is in milliseconds.

Table 5 Polling interval in milliseconds

b	Rate	μFrame[4:0]
4	1 ms	0 0000
5	2 ms	0 0001
6	4 ms	0 0010 to 0 0011
7	8 ms	0 0100 to 0 0111
8	16 ms	0 1000 to 0 1111
9	32 ms	1 0000 to 1 1111

2. Specify Data Start Address (bits 23 to 8) of DW2. This is the start address of the actual payload data that will be sent on or received from the USB bus.
   - This is the internal memory address and not the direct CPU address.
   - Internal RAM address = (CPU address – 400h) / 8.

3.2.5.2.4 DW3

1. Specify NrBytesTransferred[14:0] (bits 11 to 0) of DW3 to 0. This field is updated by the hardware after the completion of the scheduled PTD. For successfully completed PTD, this field must match bits 14 to 2 of DW0.
2. Specify Error Counter (Cerr[1:0], bits 24 to 23) of DW3. Set the error counter to 3h. The transaction will be retried three times before transaction error (X, bit 28 of DW3) will be set.
3. Set Data Toggle (bit 25 of DW3) to start a PTD.

- This field is set by the software before launching a PTD. The software writes the current transaction toggle value and the hardware updates after the successful completion of the PTD.
- This field setting depends on the previously completed PTD data toggle value for the same endpoint. If the previous successfully completed PTD data toggle value is logic 1 (updated by the hardware after the successful completion of the PTD) for the same endpoint, set the same data toggle for the current PTD of the same endpoint.
- If it is the first PTD to be scheduled for a particular endpoint, set this to 0 before launching.

4. Set the start and complete split (SC, bit 27 of DW3) to 0 before the PTD launch. For the first time, the software sets the SC bit to 0, and the hardware updates it.
5. Transaction error (X, bit 28 of DW3) will be set by the hardware after the transaction is retried three times when it detects any error.
   - This bit status corresponds to bit 0 of Status0 (bits 10 to 8 of DW4) to Status7 (bits 31 to 29) of DW4 for every microframe transfer status.
6. Babble error (B, bit 29 of DW3) will be set by the hardware after the transaction is retried three times when it detects any error.
7. The babble error (B, bit 29 of DW3) will be set by the hardware when it detects any babble error.
   - When the babble error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
   - This bit status corresponds to bit 1 of Status0 (bits 10 to 8 of DW4) to Status7 (bits 31 to 29) of DW4 for every microframe transfer status.
8. The halt error (H, bit 30 of DW3) will be set by the hardware when it detects any endpoint stall or any serious errors in the status of the transfer descriptor.
   - When the halt error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
   - This bit is set when any microframe transfer status has a stalled or halted condition.
9. Set Active (bit 31 of DW3) to the same value as Valid (bit 0 of DW0) before the PTD launch.
   - Active will be reset by the hardware only when there is an error (H, B, or X) or when the PTD successfully completes the transfer.

3.2.5.2.5 DW4

1. µSA[7:0] (bits 7 to 0 of DW4) specifies which µSOF the start split needs to be placed.

   For OUT token, when µFrame[4:0] (bits 7 to 3 of DW2) matches the frame number of the USB bus, these bits are checked for one before they are sent for µSOF.

   For example: If the polling interval is set to be 2 ms in µFrame[4:0] (bits 7 to 3 of DW2), and bit 3 of µSA[7:0] of DW4 is also set, then the start split will be sent in the fourth microframe.

For IN token, only µSOF0, µSOF1, µSOF2, or µSOF3 can be set to 1. Nothing can be set for µSOF4 and above.

Note: *DW5, DW6, DW7, and all the bits not specified in DW0, DW1, DW2, DW3, and DW4 are reserved and set to 0 before the PTD launch.*

2. Ensure that bit 1 (INTL_BUFF_FILL) of the HcBufferStatus register (BAh) is filled up if not filled so.
3. Ensure that INTL IRQ Mask OR register (DAh) bit fields are updated corresponding to the INTL PTD memory location.
4. If you would like to define any AND condition between the INTL PTDs, update the INTL IRQ Mask AND register (E0h).
5. Each bit in the INTL PTD Skip Map register indicates whether the corresponding INTL PTD is in schedule or not. Each INTL PTD Skip Map bit is mapped to one of the 16 INTL PTDs. When a bit in the PTD skip map is set to logic 1 that PTD will be skipped, that is, to be scheduled by the host controller hardware. So read the INTL PTD Skip Map register (ACh) and reset the skip bit corresponding to the INTL PTD memory location and write into the INTL PTD Skip Map register (ACh). After this, PTD will be executed by the host controller hardware.

3.2.5.3 Completion of high-speed and split-speed INT PTD handling

The following sequence describes the completion handling of high-speed and split-speed interrupt IN and OUT INT PTDs.

1. After scheduling the INT PTD as described in Section 3.2.5.1 and Section 3.2.5.2, wait for the SOF or INTL interrupt as described in Section 3.2.3.
2. After the done map bit corresponding to the scheduled INTL PTD is set in the INTL PTD Done Map register, it indicates that the PTD is retired and needs to be read.
3. Read the retired INTL PTD and check the following for the completion status in sequence.
   a. Check Halt (H, bit 30 of DW3), Babble (B, bit 29 of DW3), and Error (X, bit 28 of DW3). If any of H, B, or X is set, then the PTD is completed with endpoint halt or any other serious error in transfer. So abort all the successive transfer descriptors in the transfer for the same endpoint and inform the application with error status (along with previous successfully completed transfer descriptor status in the same transfer for the same endpoint, if any).
   b. If the preceding conditions are not met, check if Valid (bit 0 of DW0) and Active (bit 31 of DW3) are 0, and:
      i. If it is the high-speed interrupt PTD, compare NrBytesToTransfer[14:0] (bits 17 to 3 of DW0) and NrBytesTransferred[14:0] (bits 14 to 0 of DW3).

         If it split-speed interrupt PTD, compare NrBytesToTransfer[11:0] (bits 14 to 3 of DW0) and NrBytesTransferred[11:0] (bits 11 to 0 of DW3).
      ii. If both are matching, check bits 11 to 10 of DW1.
      iii. If it is the IN transfer type, read the payload data from the payload address specified in bits 23 to 8 of DW2.

iv. If it is the last PTD in the current transfer list, inform the application with status.

c. If the preceding condition is not met, check if Valid (bit 0 of DW0) is set to 0 and if Active (bit 31 of DW3) is 1, and do the following:

i. Update the Active and Valid bits to 1.

ii. Reschedule the PTD.

d. If the preceding condition is not met, check if Valid (bit 0 of DW0) and Active (bit 31 of DW3) are 0.

4. If it is the high-speed interrupt PTD, compare NrBytesToTransfer[14:0] (bits 17 to 3 of DW0) and NrBytesTransferred[14:0] (bits 14 to 0 of DW3).

   If it is the split-speed interrupt PTD, compare NrBytesToTransfer[11:0] (bits 14 to 3 of DW0) and NrBytesTransferred[11:0] (bits 11 to 0 of DW3).

   i. If both are not matching, it means it can be a short transfer. Check bits11 to 10 of DW1. If it is the IN transfer type, read the payload data from the payload address specified in bits 23 to 8 of DW2.

   ii. If it is the last PTD in the current transfer list, inform the application with status.

3.2.6 ISO transfer descriptor handling

Ensure that the ISP1763A host controller is initialized as specified in Section 3.2.2.

Ensure that required interrupts are enabled and handled as specified in Section 3.2.3.

3.2.6.1 Scheduling high-speed ISO IN and OUT PTDs

The following sequence describes the scheduling of high-speed ISO IN and OUT PTDs.

1. For OUT transfers, check for the available or unused payload memory in the payload memory address range (1000h to 5FFFh). Fill the payload data to be transferred on the USB bus into the payload memory address in the available memory location. For details on the memory management, see Section 4.1.
2. If data to be transferred is more than the available payload memory, then split the transfer into multiple transfers and schedule transfer descriptors one after another.
3. Fill the following double words into any of the 16 ISO PTD memory areas ranging from 0400h to 05FFh. For details, see Section 3.1.2.1.

3.2.6.1.1 DW0

1. Set Valid (bit 0 of DW0) to 1 to indicate that the current PTD is active.
2. Specify NrBytesToTransfer[14:0] (bits 17 to 3) of DW0 to indicate to the host controller scheduler that this PTD can transfer as specified in this field. If this PTD is programmed to be an OUT PTD, it specifies that this PTD can transfer data from the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). If this PTD is programmed to be an IN PTD, it specifies that this PTD can receive data into the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). This PTD can transfer data up to 20 kB.

3. Set the maximum packet length in bits 28 to 18 of DW0. The maximum packet size for an isochronous transfer is 1024 bytes. The maximum packet size for the isochronous transfer is also variable.
4. Set the multiplier counter value in bits 30 to 29 of DW0.
   - Multiplier counter value informs the host controller on the number of successive packets the host controller may submit to the endpoint in the current execution (in the current microframe).
   - Setting this field to 3h means that the host controller can submit three maximum packets in a single microframe.
   - Allowed values are 1h, 2h, and 3h.
5. Set Endpoint Number in bit 31 of DW0 and bits 2 to 0 of DW1.

3.2.6.1.2 DW1

1. Set Device Address (bits 9 to 3) of DW1. This is the USB address of the function containing the endpoint specified in the endpoint number field.
2. Set Token Type (bits 11 to 10) of DW1. This token is the PID for this transaction.
   - 00 – OUT – If the PTD is an OUT PTD.
   - 01 – IN – If the PTD is an IN PTD.
3. Specify Endpoint Type (bits 13 to 12) of DW1.
   - Write 01 for isochronous transfer.
4. Set the speed of the transaction in bit 14 of DW1.
   - 0 for high-speed transaction.
   - 1 for split transaction.

3.2.6.1.3 DW2

1. Set the frame number in µFrame (bits 7 to 3) of DW2 in which the PTD will be sent for ISO IN or OUT.
2. Specify Data Start Address (bits 23 to 8) of DW2. This is the start address of the actual payload data that will be sent on or received from the USB bus.
   - This is the internal memory address and not the direct CPU address.
   - Internal RAM address = (CPU address – 400h) / 8.

3.2.6.1.4 DW3

1. Specify NrBytesTransferred[14:0] (bits 14 to 0) of DW3 to 0. This field is updated by the hardware after the completion of the scheduled PTD.
2. Halt error (H, bit 30 of DW3) will be set by the hardware when it detects any endpoint stall or any serious error in the status of the transfer descriptor.
   - When halt error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.

- Set this bit to 0 before the PTD launch.

3. Set Active (bit 31 of DW3) to the same value as Valid (bit 0 of DW0) before the PTD launch.
   - Active will be reset by the hardware only when there is an error (H) or when the PTD successfully completes the transfer.

3.2.6.1.5 DW4

1. When the frame number (bits 7 to 3 of DW2) matches the frame number of the USB bus, these bits are checked for 1 before they are sent for µSOF.

   For example: When µSA[7:0] = 1, 1, 1, 1, 1, 1, 1, 1: send ISO for every µSOF of the entire millisecond. When µSA[7:0] = 0, 1, 0, 1,0, 1, 0, 1: send ISO for µSOF0, µSOF2, µSOF4, and µSOF6.

   Note: *DW5, DW6, DW7, and all the bits not specified in DW0, DW1, DW2, DW3, and DW4 are reserved, and set to 0 before the PTD launch.*

2. Ensure that bit 2 (ISO_BUFF_FILL) of the HcBufferStatus register (BAh) is filled up if not filled so.
3. Ensure that ISO IRQ Mask OR register (D8h) bit fields are updated corresponding to the ISO PTD memory location.
4. If you would like to define any AND condition between ISO PTDs, update the ISO IRQ Mask AND register (DEh).
5. Each bit in the ISO PTD Skip Map register indicates whether the corresponding ISO PTD is in schedule or not. Each INTL PTD Skip Map bit is mapped to one of the 16 ISO PTDs. When a bit in the PTD skip map is set to logic 1 that PTD will be skipped, that is, to be scheduled by the host controller hardware. So, read the ISO PTD Skip Map register (A6h) and reset the skip bit corresponding to the ISO PTD memory location, and write into the ISO PTD Skip Map register (A6h). After this, the PTD will be executed by the host controller hardware.

3.2.6.2 Scheduling split ISO IN and OUT PTDs

The following sequence describes the scheduling of split-speed ISO IN and OUT PTDs.

1. For OUT transfers, check for the available or unused payload memory in the payload memory address range (1000h to 5FFFh). Fill the payload data to be transferred on the USB bus into the payload memory address in the available memory location. For details on the memory management, see Section 3.2.1.
2. If data to be transferred is more than the available payload memory, split transfers into multiple transfers, and schedule transfer descriptors one after another.
3. Fill following double words into any of the 16 ISO PTD memory areas ranging from 0400h to 05FFh. For details, see Section 3.1.2.1.

3.2.6.2.1 DW0

1. Set Valid (bit 0) of DW0 to 1 to indicate that the current PTD is active.
2. Specify NrBytesToTransfer[14:0] (bits 17 to 3) of DW0 to indicate to the host controller scheduler that this PTD can transfer as specified in this field. If this PTD is

programmed to be an OUT PTD, it specifies that this PTD can transfer data from the payload memory (payload memory address is specified in bits 23 to 8 of DW2). If this PTD is programmed to be an IN PTD, it specifies that this PTD can receive data into the payload memory (payload memory address will be specified in bits 23 to 8 of DW2). This field is restricted to transfer data up to 1023 bytes because the maximum allowable split ISO transfers for a full-speed device is 1023 bytes. This field indirectly becomes the maximum packet size for the downstream port.

3. Specify the transaction translator maximum packet size length in bits 28 to 18 of DW0. This field indicates the maximum number of bytes that can be sent per start split, depending on the total number of bytes needed. If the total bytes to be sent for the entire millisecond are greater than 188 bytes, this field must be set to 188 bytes for an OUT token and 192 bytes for an IN token. Otherwise, this field must be equal to the total bytes sent.
4. Set Endpoint Number in bit 31 of DW0 and bits 2 to 0 of DW1.

3.2.6.2.2 DW1

1. Set Device Address (bits 9 to 3) of DW1. This is the USB address of the function containing the endpoint specified in the endpoint number field.
2. Set Token Type (bits 11 to 10) of DW1. This token is the PID for this transaction.
   - 00 – OUT – If the PTD is an OUT PTD.
   - 01 – IN – If the PTD is an IN PTD.
3. Specify Endpoint Type (bit 13 to 12) of DW1.
   - Write 01 for isochronous transfer
4. Set the speed of the transaction in bit 14 of DW1.
   - 0 for high-speed transaction
   - 1 for split transaction.
5. Set the port number field in bits 24 to 18 of DW1. This indicates the port number of the hub or embedded TT.
6. Set the hub address in bits 31 to 24 of DW1. This indicates the hub address.

3.2.6.2.3 DW2

1. Set the frame number in µFrame[4:0] (bits 7 to 3) of DW2 in which the PTD will be sent for ISO IN or OUT.
2. Specify Data Start Address (bits 23 to 8) of DW2. This is the start address of the actual payload data that will be sent on or received from the USB bus.
   - This is the internal memory address and not the direct CPU address.
   - Internal RAM address = (CPU address – 400h) / 8.

3.2.6.2.4 DW3

1. Specify NrBytesTransferred[11:0] (bits 11 to 0 of DW3) as 0. This field is updated by the hardware after completion of the scheduled PTD.

2. Set the data toggle in bit 25 of DW3 to start the PTD.
3. Set the start and complete split (SC, bit 27 of DW3) to 0 before the PTD launch. For the first time the software sets the SC bit to 0, and the hardware updates it.
4. Transaction error (X, bit 28 of DW3) will be set by the hardware after the transaction when it detects any error.
   - This bit status corresponds to bit 0 of Status0 (bits 10 to 8 of DW4) to Status7 (bits 31 to 29 of DW4) for every microframe transfer status.
5. The babble error (B, bit 29 of DW3) will be set by the hardware when it detects any babble error.
   - When the babble error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
   - This bit status corresponds to bit 1 of Status0 (bits 10 to 8 of DW4) to Status7 (bits 31 to 29 of DW4) for every microframe transfer status.
6. The halt error (H, bit 30 of DW3) will be set by the hardware when it detects any endpoint stall or any serious errors in the status of the transfer descriptor.
   - When the halt error is detected, Active (bit 31 of DW3) and Valid (bit 0 of DW0) will be set to inactive.
   - This bit is set when any microframe transfer status has a stalled or halted condition.
7. Set Active (bit 31 of DW3) to the same value as Valid (bit 0 of DW0) before the PTD launch.
   - Active will be reset by the hardware only when there is an error (H, B, or X) or when the PTD successfully completes the transfer.

3.2.6.2.5 DW4

1. When the frame number of bits 7 to 3 of DW2 matches the frame number of the USB bus, these bits are checked for 1 before they are sent for µSOF.
2. µSA[7:0] (bits 7 to 0) of DW4 specifies which µSOF the start split needs to be placed.
   - For the OUT token, when the frame number of bits 7 to 3 of DW1 matches the frame number of the USB bus, these bits are checked for one before they are sent for µSOF.
   - For IN token, only µSOF0, µSOF1, µSOF2, or µSOF3 can be set to one. Nothing can be set for µSOF4 and above.

   Note: *DW5, DW6, DW7, and all the bits not specified in DW0, DW1, DW2, DW3, and DW4 are reserved and set to 0 before the PTD launch.*
3. Ensure that bit 2 (ISO_BUFF_FILL) of the HcBufferStatus register (BAh) is filled up if not filled so.
4. Ensure that ISO IRQ Mask OR register (D8h) bit fields are updated corresponding to the ISO PTD memory location.
5. If you would like to define any AND condition between ISO PTDs, update the ISO IRQ Mask AND register (DEh).

6. Each bit in the ISO PTD Skip Map register indicates whether the corresponding ISO PTD is in schedule or not. Each INTL PTD Skip Map bit is mapped to one of the 16 ISO PTDs. When a bit in the PTD skip map is set to logic 1 that PTD will be skipped, that is, to be scheduled by the host controller hardware. So read the ISO PTD Skip Map register (A6h) and reset the skip bit corresponding to the ISO PTD memory location, and write to the ISO PTD Skip Map register (A6h). After this, PTD will be executed by the host controller hardware.

3.2.6.3 Completion of high-speed and split-speed ISO PTD handling

The following sequence describes the completion handling of high-speed and split-speed ISO IN and OUT INTL PTDs.

1. After scheduling the ISO PTD as described in Section 3.2.6.1 and Section 3.2.6.2, wait for the SOF or ISO interrupt as described in Section 3.2.3.
2. After the done map bit corresponding to the scheduled ISO PTD is set in the ISO PTD Done Map register, it indicates that the PTD is retired and needs to be read.
3. Read the retired ISO PTD and check the following for the completion status in sequence.
   a. Check Halt (H, bit 30 of DW3), Babble (B, bit 29 of DW3), and Error (X, bit 28 of DW3). If any of the H, B, or X is set, then the PTD is completed with endpoint halt or any other serious error in transfer. Inform the application of error status with bytes transferred.
   b. If the preceding condition not met, check if Valid (bit 0 of DW0) and Active (bit 31 of DW3) are 0, and:

      If it is the IN transfer type, read the payload data from the payload address specified in bits 23 to 8 of DW2 and inform the application with status.
   c. If the preceding condition is not met, check if Valid (bit 0 of DW0) is set to 0 and Active (bit 31 of DW3) is 1. Inform the application with status with bytes sent or received.

3.2.7 Internal hub enumeration

Ensure that the ISP1763A host controller is initialized as specified in Section 3.2.2.

Ensure that the required interrupts are enabled and handled as specified in Section 3.2.3.

1. Read the PORTSC register (A0h) and ensure that the port is powered, that is, the Port Power and Current Connect Status bits are set, and the Line Status field is set to 10h.
2. After detecting the Current Connect Status bit, reset and enable the port. Follow these steps to reset and enable the port.
   a. Set Port Reset (bit 8), and write it to the PORTSC register (A0h).
   b. Wait for 50 ms according to *Universal Serial Bus Specification Rev. 2.0* for the root hub port.
   c. Clear Port Reset (bit 8), and write it to the PORTSC register (A0h).
   d. Read PORTSC (A0h) until the Port Reset bit is cleared by the hardware.
   e. Ensure that Port Reset (bit 8) is 0, and that port is enabled (bit 2 is set).

f. Wait for 20 ms, and read the PORTSC register (A0h).

3. Enumerate the connected hub as follows using control transfers as specified in Section 3.2.4.
   a. Set the device address.
   b. Get the device descriptor.
   c. Get the first nine bytes of the configuration descriptor.
   d. Get the total length bytes of the configuration descriptor.
   e. Set the configuration.
   f. Get the hub descriptor.
   g. Power up hub ports using the SET_FEATURE control transfer request with PORT_POWER (8h) feature selector.
   h. Power up hub port indicators using SET_FEATURE control transfer request with PORT_INDICATOR (16h) feature selector.
4. Now the hub is in operational state and issues a one-byte interrupt PTD (with polling interval mentioned in the hub interrupt endpoint descriptor) as described in Section 3.2.5 to detect the port status changes (that is, connect, disconnect, overcurrent, suspend, resume, port enable, port disable, and so on) on hub ports.

3.2.8 External device enumeration

Ensure that the ISP1763A host controller is initialized as specified in Section 3.2.2.

Ensure that required interrupts are enabled and handled as specified in Section 3.2.3.

1. Ensure that a one-byte interrupt PTD is scheduled (with polling interval mentioned in the hub interrupt endpoint descriptor) as described in Section 3.2.5 to detect the port status changes (that is, connect, disconnect, overcurrent, suspend, resume, port enable, port disable, and so on) on the hub ports.
2. Once any device is connected, there will be a PTD completion interrupt for the interrupt PTD that is scheduled. This one-byte specifies which port has got the change notification.
3. Once you get the change notification on a particular port, find out the detailed status of the port. Submit a GetPortStatus four-byte control transfer request on the port to find out the exact status of the port.
4. Once you get the change information (Connect Status Change, Port Enable Change, Suspend Change, and Overcurrent Change), this change notification must be cleared using the Clear_Feature control transfer request.
5. Wait for 100 ms debounce time for the port to get stabilized.
6. Submit a GetPortStatus four-byte control transfer request to get the stable status information after waiting for debounce time duration.
7. After detecting Connect Status Change and getting the Port Status Information, reset and enable the port. Submit a SET_FEATURE control transfer request with feature select as PORT_RESET (4h).
8. Wait for 20 ms.

Submit one-byte interrupt PTD for any change notification.

9. After issuing port reset, you will detect port reset change notification and get an interrupt for one-byte interrupt PTD.
10. Get the actual port status by submitting four-byte control transfer request.
11. If the status is Port Reset Change (1h), Port Reset (0h), and if the port is enabled (1h), then clear Port Reset Change using the CLEAR_FEATURE control transfer request.
12. Wait for 50 ms and submit a GetPortStatus four-byte control transfer request to get the connection status.
13. If the port is connected, enumerate the device as follows using the control transfer requests as specified in Section 4.2.
    a. Set the device address.
    b. Get the device descriptor.
    c. Get the first nine bytes of the configuration descriptor.
    d. Get the total length bytes of the configuration descriptor.
    e. Set the configuration.
14. Issue a one-byte interrupt PTD (with polling interval mentioned in the hub interrupt endpoint descriptor) as described in Section 3.2.5 to detect further port status changes (that is, connect, disconnect, overcurrent, suspend, resume, port enable, port disable, and so on) on hub ports.
15. Call the respective application according to the configuration descriptor information.

3.2.9 Power management

The ISP1763A can be programmed to put the controller in suspend and low-power (deep sleep) mode.

Ensure that the ISP1763A host controller is initialized as specified in Section 3.2.2.

Ensure that required interrupts are enabled and handled as specified in Section 3.2.3.

Ensure that the USB Interrupt register (94h) is set to 4h.

3.2.9.1 Suspending the host controller

The following sequence describes the suspending of the host controller, suspending of the ATX, and putting the controller in deep sleep mode.

1. As soon you get the suspend request from the upper layer (for example, application), stop the host controller using the RS bit. Set the RS bit in the USB Command register (8Ch) to 0.
2. Set the following bits in the PORTSC register (A0h) to suspend the root port and ensure that the port is powered and enabled before making the following changes to PORTSC (A0h).
    a. Set suspend (bit 7).
3. Suspend the peripheral controller using this sequence.
    a. Read the Mode register (0Ch). Set GOSUSP (bit 5) of the Mode register.

b. Wait for 1 ms, and clear GOSUSP (bit 5) of the Mode register. This sequence sets the device in suspend mode.

4. To put the host controller in deep sleep mode, set the following bits in the Power Down Control register (D0h) as the final step.

Set CLK_OFF_COUNTER (bits 31 to 16) of the Power Down Control register. Determines the wake-up status duration after any wake-up event before the ISP1763A goes back into suspend mode. The time-out is applicable only if, during the given interval, the host controller is not programmed back to the normal functionality. The maximum value FFFFh determines a maximum wake-up time of 500 ms.

a. Set bit 5 (REG_PWR) and bit 4 (REG_SUSP_PWR) of the Power Down Control register to force the regulator to go into suspend mode.

3.2.9.2 Resuming the host controller

The following sequence describes the resuming of the host controller from normal suspend mode and deep-sleep suspend mode.

3.2.9.2.1 Resuming from deep sleep mode

1. To resume from deep-sleep suspend mode requires dummy read from the PIO interface. Dummy read from the PIO interface causes the clock to oscillate and the chip generates the clock ready interrupt. The application request is required to resume the host controller. When the application request is received, do a dummy register read access.
2. When you get the clock ready interrupt, deinitialize the host controller (detach any device if connected), and reinitialize host controller registers and enumerate the internal hub.

3.2.9.2.2 Resuming from normal suspend mode

1. Resuming from normal suspend mode requires either dummy read from the PIO interface or wake up by any remote wake up device connection.
   a. Dummy read from the PIO interface causes the clock to oscillate and the chip generates the clock ready interrupt.
   b. If you connect any remote wake-up device, it causes the OPR interrupt to be generated. Once you get the OPR interrupt, check Port Change Detect (bit 2) of the USB Status register (90h), and clear it by writing the same value to the USB Status register.
2. When you get the OPR interrupt or clock ready interrupt, deinitialize the host controller (detach any device if connected), reinitialize host controller registers, and enumerate the internal hub.

4 Peripheral programming guide

This section provides the programming guide for the peripheral module.

4.1 Basic operations

The following operations are required for the programming of the peripheral.

- Power-on initialization
- Configure endpoints
- Interrupt function
- Reading data from the endpoint buffer
- Writing data to the endpoint buffer

4.1.1 Power on initialization

Following is the procedure to initialize the ISP1763A as a peripheral:

1. Select the bus width and the interrupt polarity in the Hardware Mode Control register.
2. Read the Hardware Mode Control register, and lock interface mode.
3. Enable register write access by writing AA37h to the Unlock Device register.
4. Issue soft reset by using the Mode register.
5. Enable device mode (SW_SEL_HC_DC = 1) and bit DP_PULLUP, and disable bits DP_PULLDOWN and DM_PULLDOWN in the OTG Control register.
6. Configure device endpoints using the configure endpoints function.
7. Write 80h to the Address register.
8. Write the desired value to the Mode register. Default value is 020Ch.
9. Write the correct value INTLVL = 0 to the Interrupt Configuration register. Set the CDBGMOD, DDBGMODIN, and DDBGMODOUT fields as interrupt on all ACK (01h) for the control, data IN, and data OUT transfers. Default value is 54h.
10. Enable the required interrupt in the Interrupt Enable register. Default value is 03FF FDF9h.

4.1.2 Configure endpoints

1. Disable all the endpoints by selecting each endpoint using the Endpoint Index register and clearing all bits in the Endpoint Type register.

2. Initialize the endpoint maximum packet size for all the endpoints by selecting each endpoint using the Endpoint Index register, and writing the maximum packet size in the endpoint MaxPacketSize register.
3. Initialize the endpoint type and enable each endpoint by using the Endpoint Type register.

4.1.3 Interrupt function

1. Enable register write access by writing AA37h to the Unlock Device register.
2. Disable the global interrupt by resetting the GLINTENA bit in the Mode register.
3. Read the DcInterrupt register, and update the interrupt status variable.
4. Clear interrupt by writing back the interrupt status in the DcInterrupt register.
5. Read the Interrupt Enable register to mask the interrupt received in the interrupt status variable.
6. Handle various interrupts based on the value in the interrupt status variable.

4.1.3.1 Handling V_{BUS} interrupt

1. Read the Mode register.
2. If the VBUSSTAT bit is set in the Mode register, enable the DP_PULLUP bit in the OTG Control register.
3. If the VBUSSTAT bit is reset in the Mode register, disable the DP_PULLUP bit in the OTG Control register.

4.1.3.2 Handling reset interrupt

1. Select the bus width (8-bit or 16-bit) and the interrupt polarity in the Hardware Mode Control register.
2. Read the Hardware Mode Control register and lock interface mode.
3. Enable device mode (SW_SEL_HC_DC = 1) and bit DP_PULLUP, and disable bits DP_PULLDOWN and DM_PULLDOWN in the OTG Control register.
4. Configure device endpoints using the configure endpoints function.
5. Write 80h to the Address register.
6. Write the desired value to the Mode register. Default value is 020Ch.
7. Write the correct value INTLVL = 0 to the Interrupt Configuration register. Set the CDBGMOD, DDBGMODIN, and DDBGMODOUT fields as interrupt on all ACK (01h) for the control, data IN, and data OUT transfers. Default value is 54h.
8. Enable the required interrupt in the Interrupt Enable register. Default value is 03FF FDF9h.

4.1.4 Reading data from the endpoint buffer

1. Write the Endpoint Index register to point to the respective endpoint.
2. Read the Buffer Length register to find out the number of bytes available in the endpoint buffer.
3. Read the Data Port register for the number of bytes available in the endpoint buffer.
4. In the case of 16-bit interface, the Data Port register must be read half the number of times of the bytes available.

4.1.5 Writing date to the endpoint buffer

1. Write the Endpoint Index register to point to the respective endpoint.
2. Write the Buffer Length register with the number of bytes to be transferred.
3. Write the Data Port register for the number of bytes to be transferred.
4. In the case of 16-bit interface, the Data Port register must be written with half the number of times of the bytes to be transferred.

4.2 USB control transfer

The USB control transfer process can be divided into three categories:

- Set-up token with data IN stage
- Set-up token with data OUT stage
- Set-up token without data stage

The Endpoint Index register is used to reference the control endpoint 0 and the set-up endpoint 0. The set-up endpoint is 8 bytes in length. It is used to store the USB device request from the host. The firmware must program the EP0SETUP bit of the ISP1763A to reference the set-up endpoint.

4.2.1 Set-up token with data IN stage

Write the Endpoint Index register to point to the control IN endpoint.

Set the DSEN bit in the Control Function register for the peripheral to enter in the data stage of the control transfer.

The peripheral will send the validated data on the USB bus.

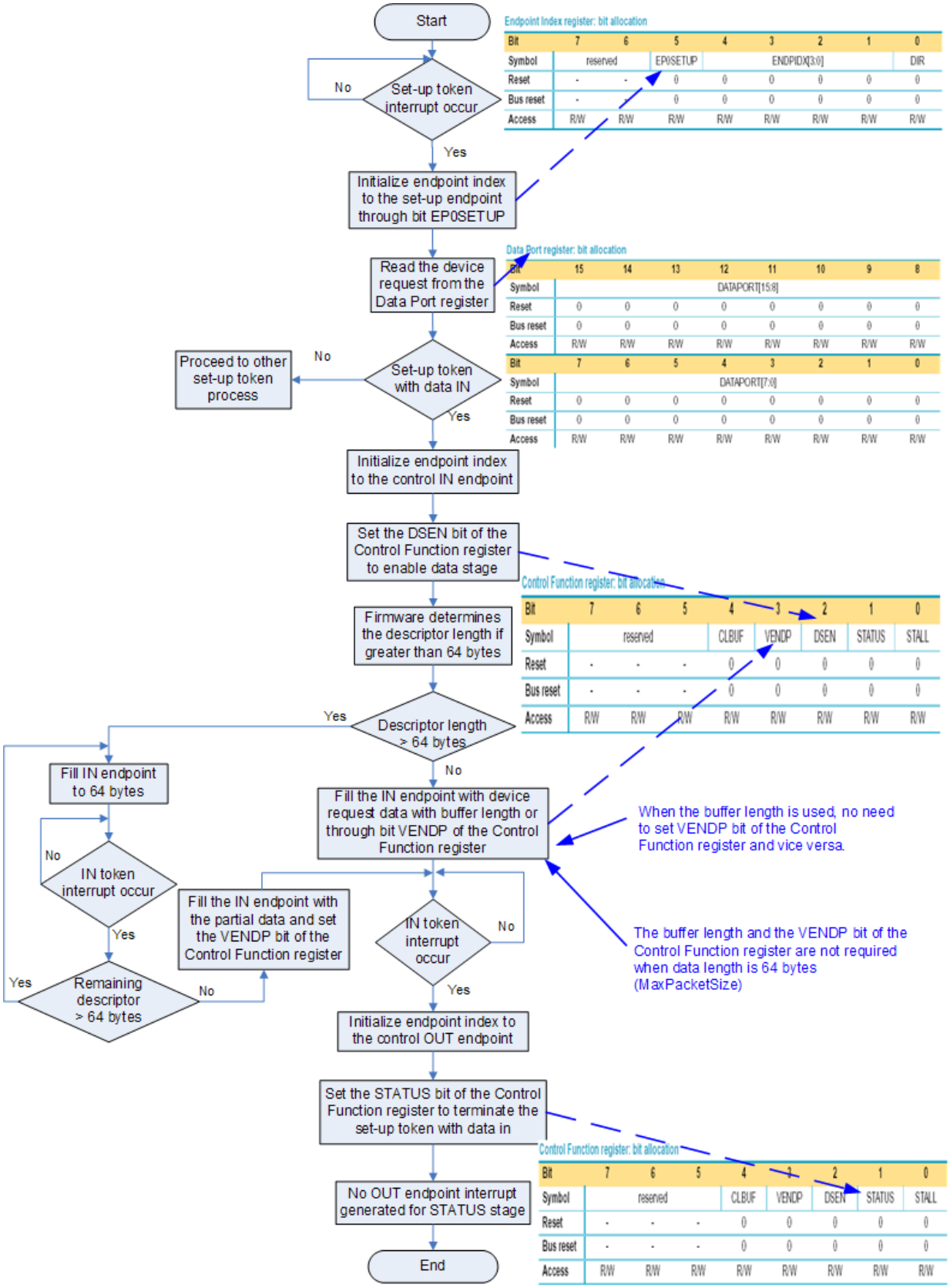


Figure 7 Set-up token with data IN stage

4.2.2 Set-up token with data OUT stage

Write the Endpoint Index register to point to the control OUT endpoint. Set the DSEN bit in the Control Function register for the peripheral to enter in the data stage of the control transfer. The peripheral will generate an ACK handshake to the OUT endpoint

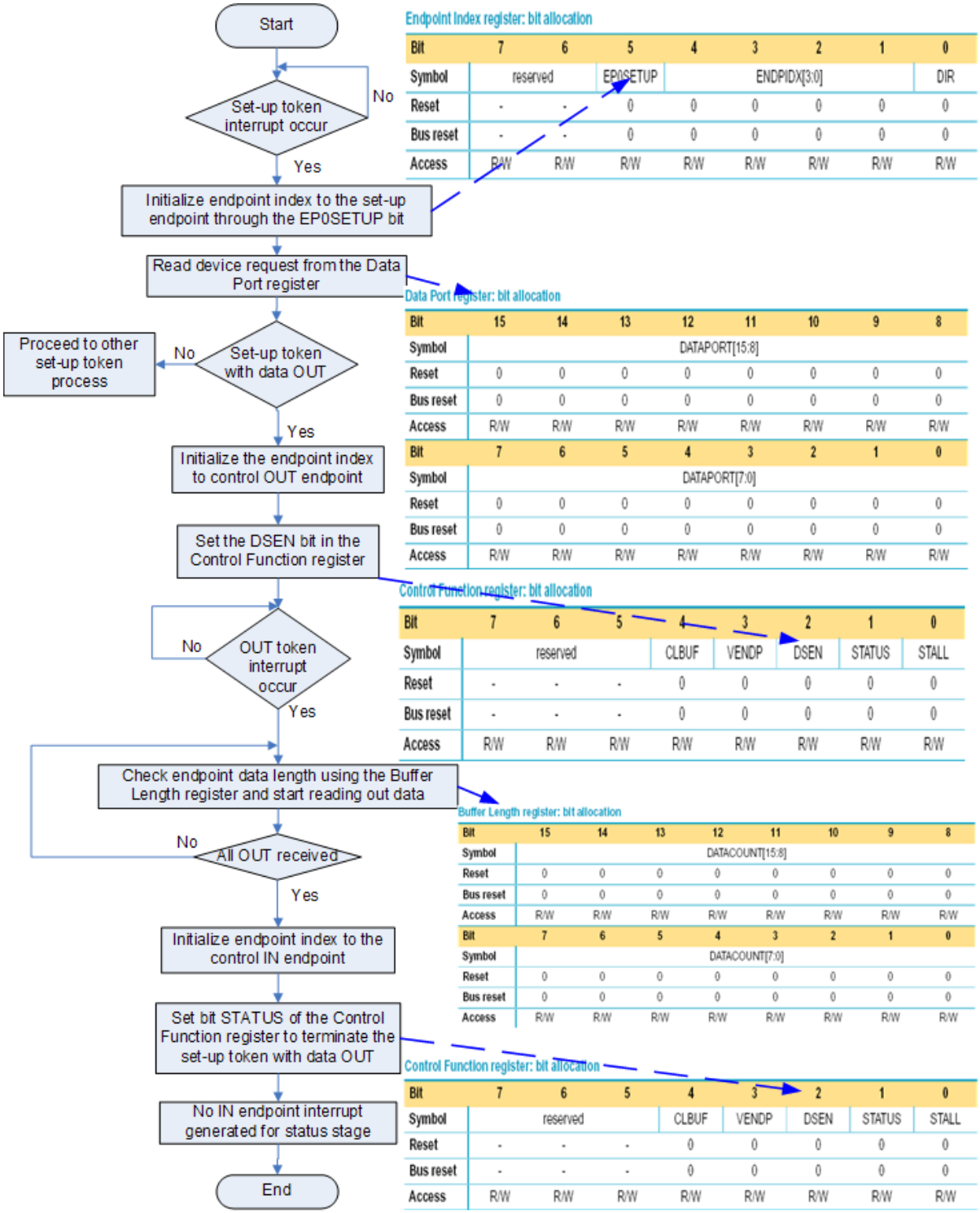


Figure 8 Set-up token with data OUT stage

4.2.3 Set-up token without data stage

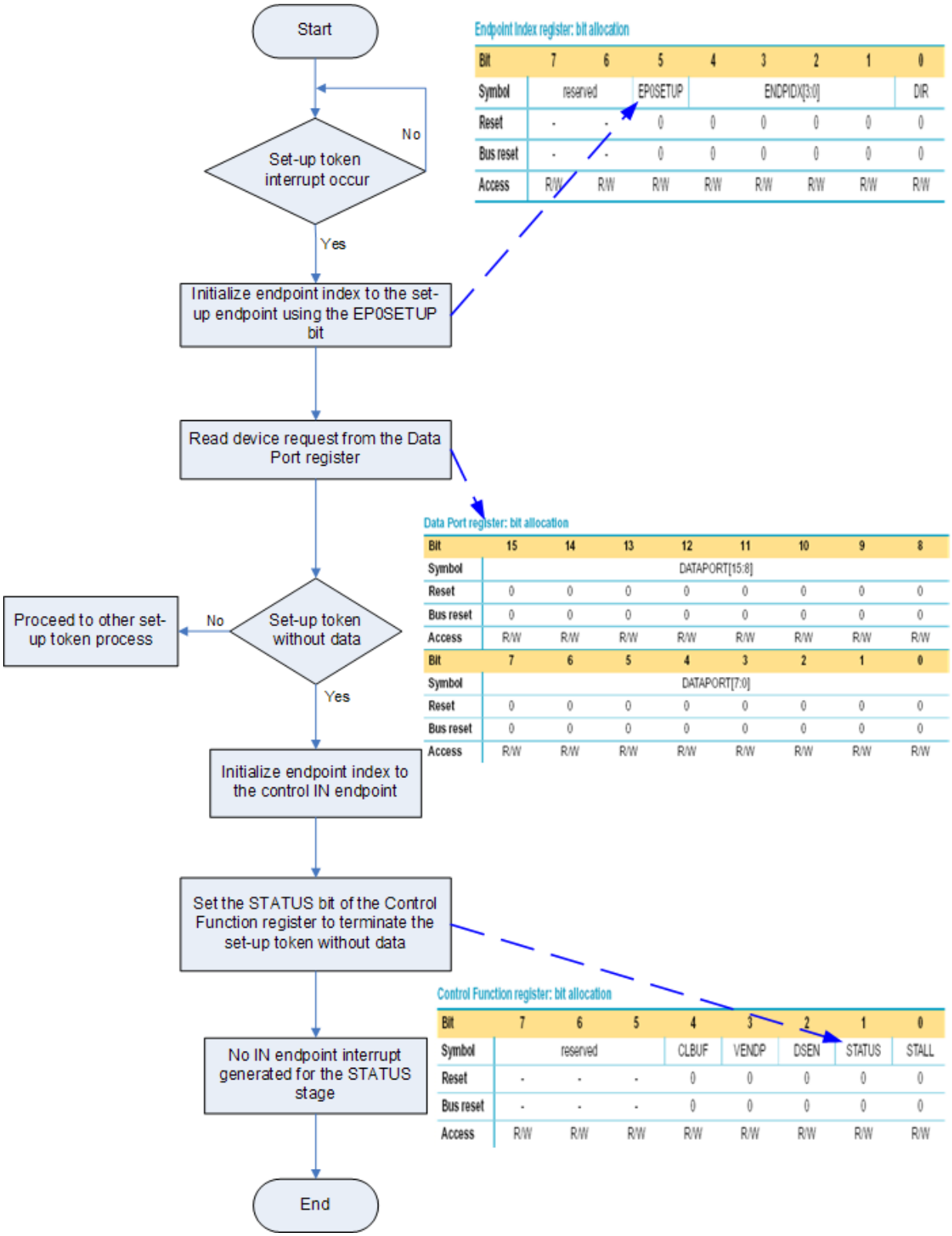


Figure 9 Set-up token without data stage

4.2.4 Stall set-up token

To stall the set-up token, see the sequence in Figure 10.

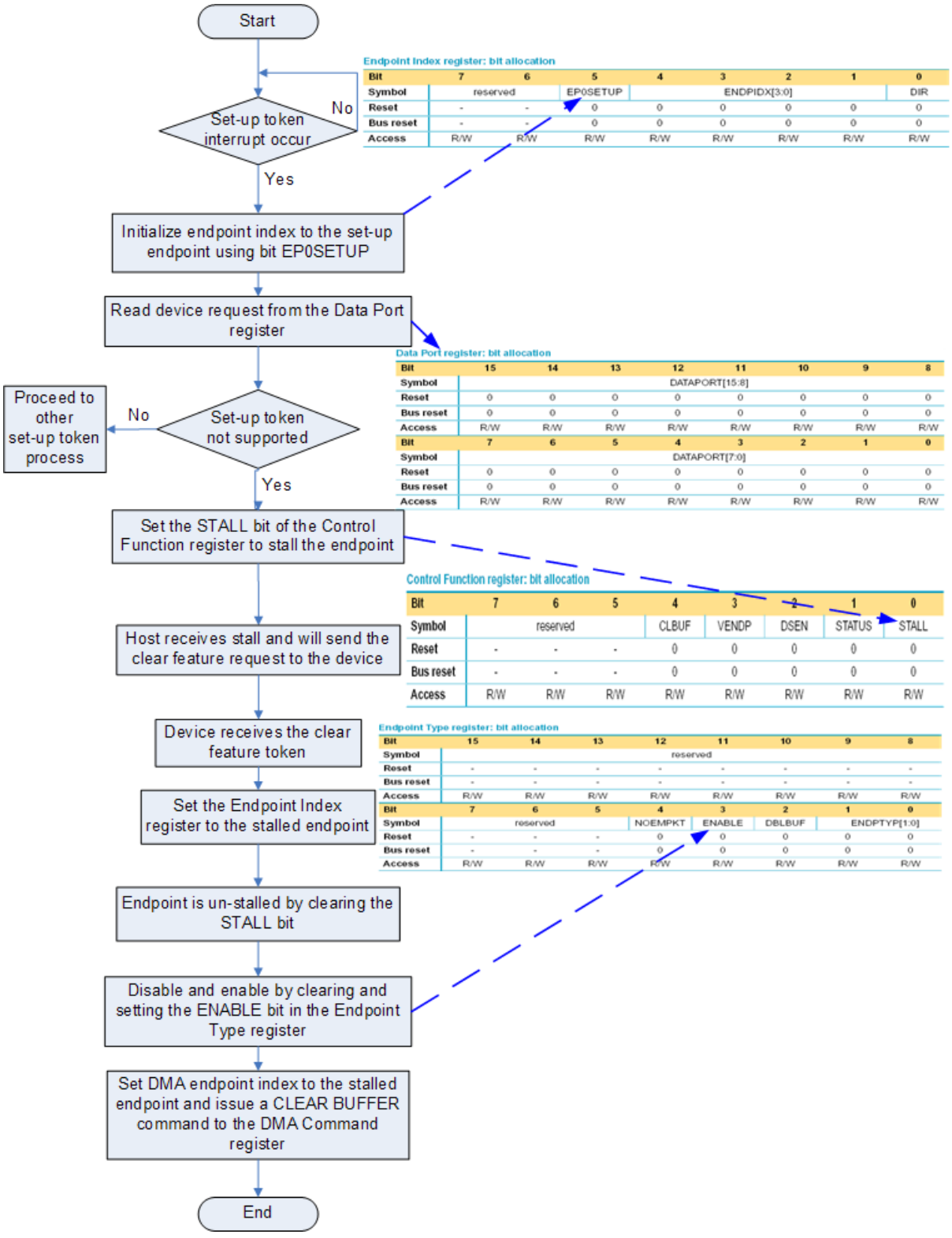


Figure 10 Stall set-up token

5 OTG programming guide

This chapter provides the programming guidelines for the ISP1763A OTG.

5.1 OTG driver software architecture

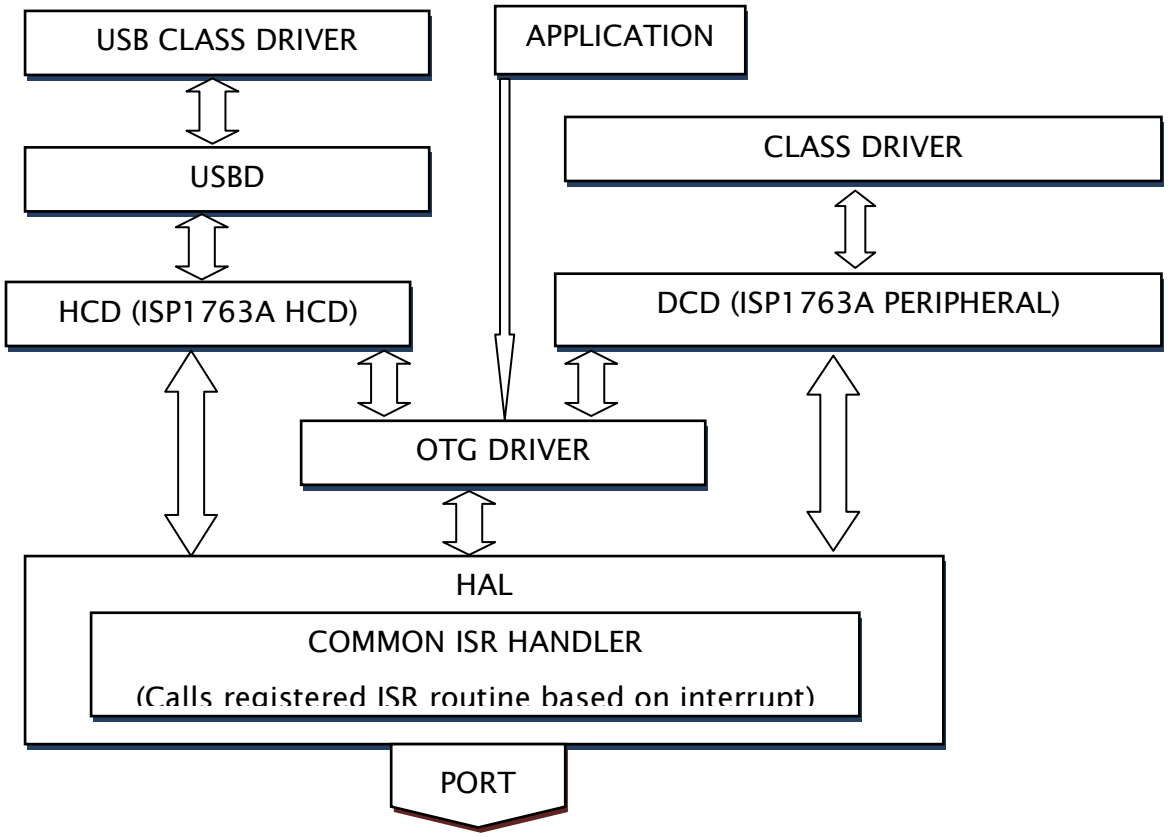


Figure 11 OTG driver software architecture

1. Application: The application accesses the interface of the OTG driver to generate SRP, driver V_{BUS}.
2. USB class driver: Common drivers that is available in the operating system. For example, MSC and serial. The USB class driver is loaded by the USB core based on the USB peripheral inserted in the port.
3. USBD: Provides basic functionality for the HCD interface and the core USB bus function.
4. HCD: Provides interface to the USBD and hardware. It also handles the URB passed by the core. It has knowledge of the host hardware. So all transfers in the USB bus are handled by this driver. This driver informs suspend, set feature, and connection event to the OTG driver. This driver is used in a_host and b_host modes.

5. USB class driver: Works in device mode. It is used when the ISP1763A works as a device and is inserted in the other host.
6. DCD: Provides endpoint handling and other bus events handling. It has knowledge of the hardware. Based on function drivers registered to them; it enables endpoints and handles transfers. This driver also handles all OTG related control transfer, sets feature B_HNP and A_HNP. It informs suspend, reset, resume, and set feature events to the OTG driver. This driver is used in b_peripheral and a_peripheral modes.
7. OTG driver: This driver handles the OTG state diagram based on the cable inserted in the port. This driver provides interface to the application to trigger SRP (in B-device) and power on V_{BUS} (in A-device). It switches the connected ATX (host or peripheral) to the port based on the OTG state diagram.
8. HAL layer: It is the layer that provides hardware access.
9. Common ISR handler: HCD driver, OTG driver, and DCD register its interrupt handling routine to this handler. This handler has the knowledge of the hardware. So based on interrupt event, it calls the corresponding driver interrupt handler.

5.2 OTG interrupt handling basic

The OTG state machine is mainly interrupt driven (both hardware events and timer). The basic OTG interrupt identification is based on OTG_IRQ (bit 10) in the HcInterrupt register (D4h). When this bit is set, it indicates the occurrence of the OTG event. Therefore, the OTG software must further check OTG-specific interrupt registers (E8h, ECh, F0h, and F4h) to know the detailed information of the OTG interrupt and process accordingly.

5.3 Basic register bit explanation

For detailed implementation of the OTG protocol, refer to *On-The-Go Supplement to the USB Specification Rev. 1.3*. This section explains certain bits in OTG registers.

Table 6 OTG Control register

Bit	Symbol	Description
9	OTG_SE0_EN	This bit is set when B-device moves from b_peripheral mode to b_host mode so that b_host sends reset once a_peripheral is connected.
8	BDIS_ACON_EN	Set this bit when a_host moves from a_host mode to a_peripheral mode. So that a_host automatically connects to b_host when a_host detects b_peripheral disconnect event.
6	VBUS_CHRG	Use this bit to generate V_{BUS} pulsing. Give some delay or time out between set and clear. It makes V_{BUS} pulsing. This bit is used in B-device mode during SRP.
5	VBUS_DISCHRG	Fast discharging of V_{BUS}. Set when a_host is moving from a_host to a-vfall.
4	VBUS_DRV	Driving V_{BUS}: Set it in a_host mode when either SRP or A_BUS_REQ is detected.

Table 7 OTG Interrupt Source register (bits are controlled by the hardware)

Bit	Symbol	Description
7	B_SESS_END	V_{BUS} has gone below its session end threshold.
4	RMT_CONN	This bit is set when a_host detects that b_peripheral is connected. This bit is set when b_host detects that a_peripheral is connected.
3	ID	This bit is set when B cable is inserted. Cleared when A cable is inserted. By default, it is set when cable is not present.
2	DP_SRP	This bit is set when a_host detects SRP.
1	A_B_SESS_VLD	B-device monitors this bit to detect V_{BUS} (B_SESS_VLD) after SRP. When this bit is cleared in B-device mode, it means that a_host stops driving V_{BUS}. B-device moves from b_idle mode to b_peripheral mode when this bit is set. When this bit is cleared, B-device must move from any state to the b_idle state. The b_idle state device can initiate SRP when this bit is not set. A-device uses this bit to evaluate A_SESS_VLD.
0	VBUS_VLD	This bit is set when a_host drives V_{BUS} above the valid voltage. The bit is cleared when a_host stops driving V_{BUS}.

5.4 OTG timer

The OTG timer is used to set time-out period for one state. The timer triggers interrupt when the timer reaches the value that is set in the OTG Timer register (FEh). The timer increments its values every 0.01 ms and expires when it reach the value that is set in the register. Timer has started running by setting 1 in bit 31. The timer can be stopped any time by clearing bit 31.

5.5 Generating SRP

The session request protocol is used to initiate a new session through the application. The B-device initiates this protocol. Two methods will be used by the B-device to request the A-device to begin a session: data-line pulsing and V_{BUS} pulsing. An A-device is only required to respond to one of the two SRP signaling methods. B-device will use both these methods when initiating SRP to ensure that an A-device responds.

B-device ensures that previous session is properly ended by ensuring V_{BUS} has dropped below its session valid threshold and 2 ms SE0 state in the bus using bits B_SESS_END and B_SE0_SRP in the OTG Interrupt Source register (E8h). This attempt must be done until either B_SESS_VLD is detected or 6 seconds expires.

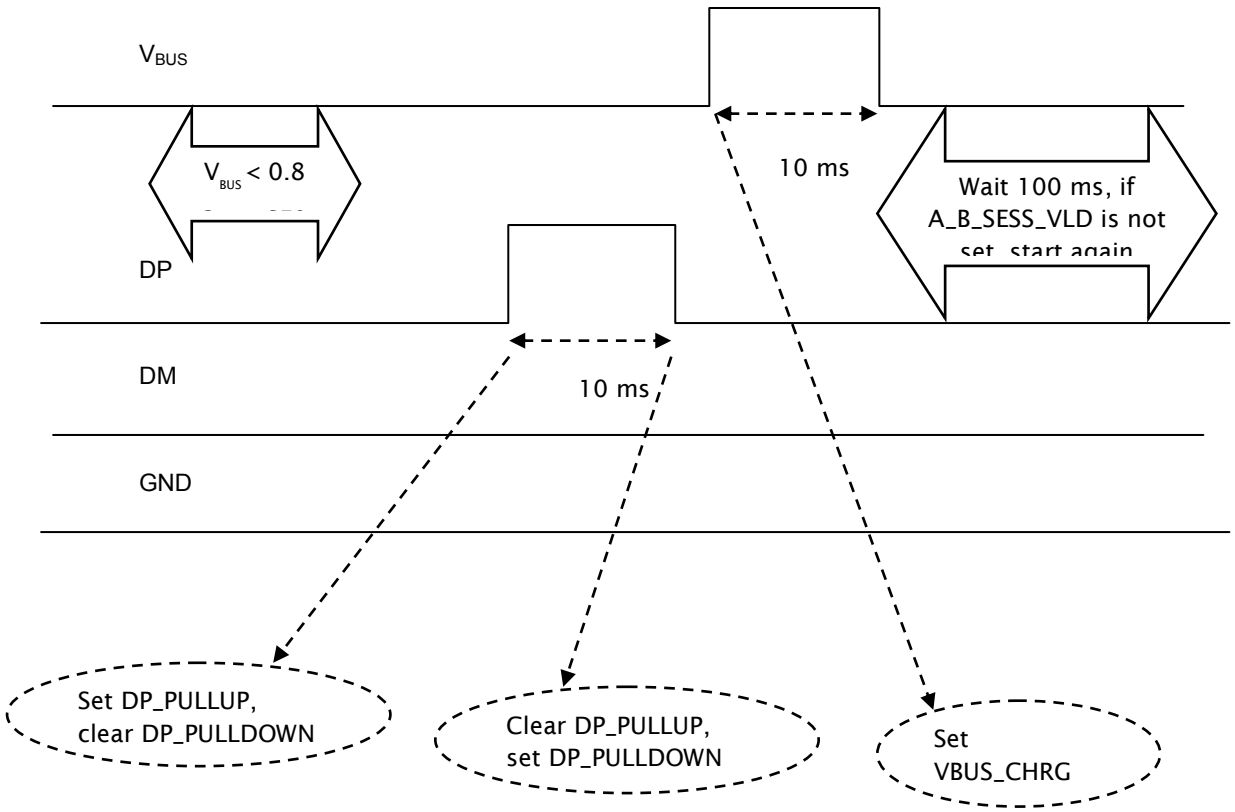


Figure 12 Generating SRP request

5.6 A_host state diagram

Follow Figure 13 to initialize as an a_host.

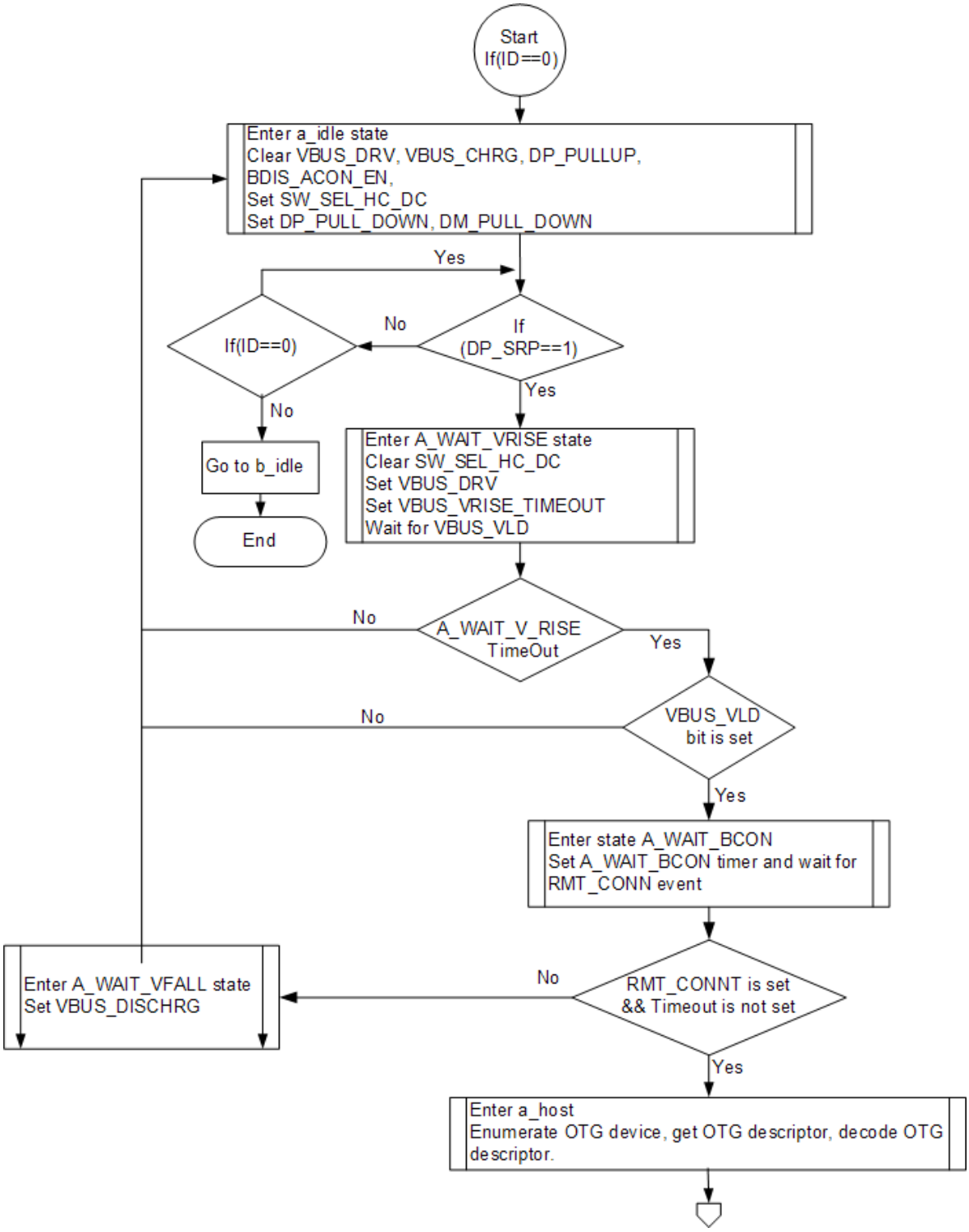


Figure 13 A_host state diagram – part 1

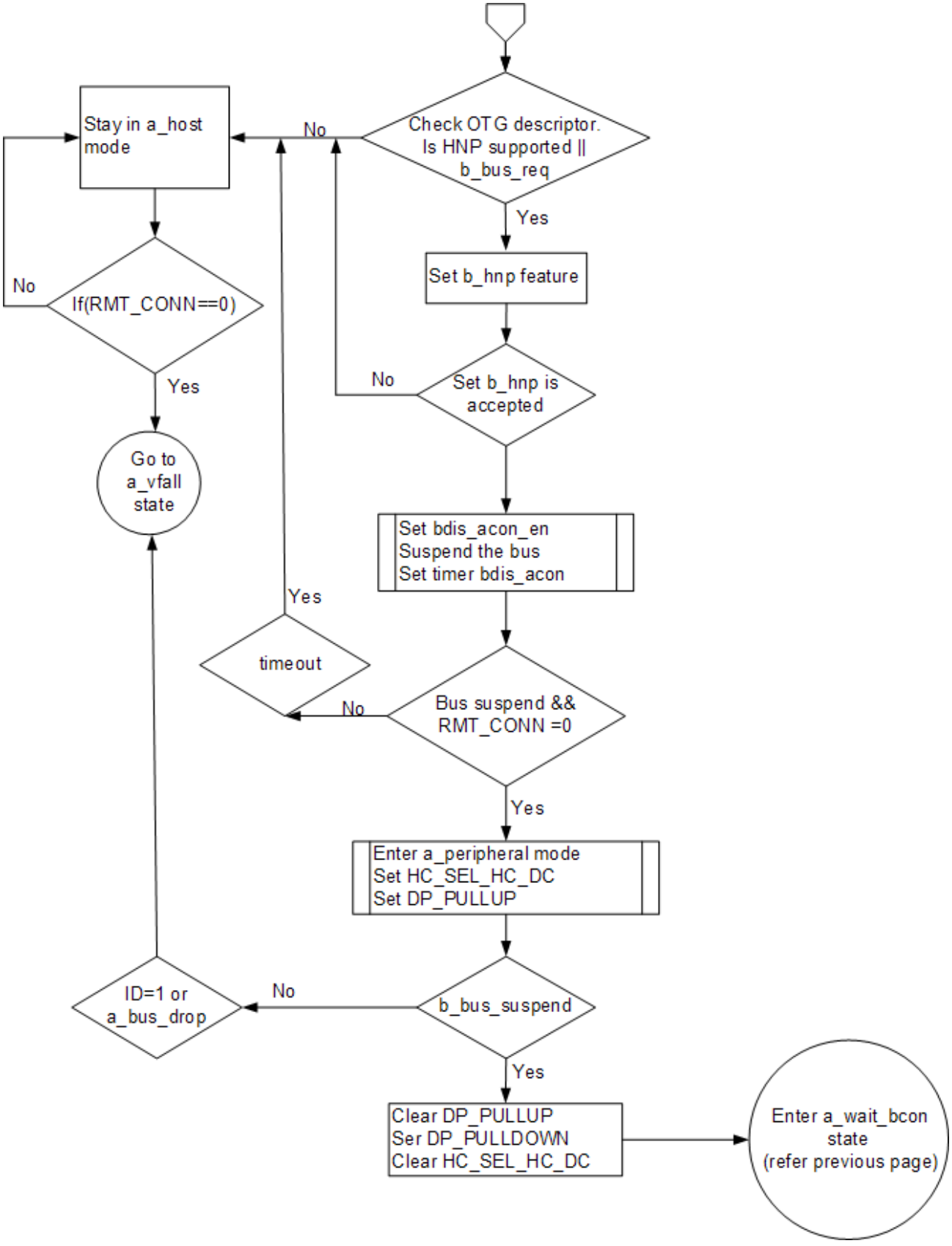
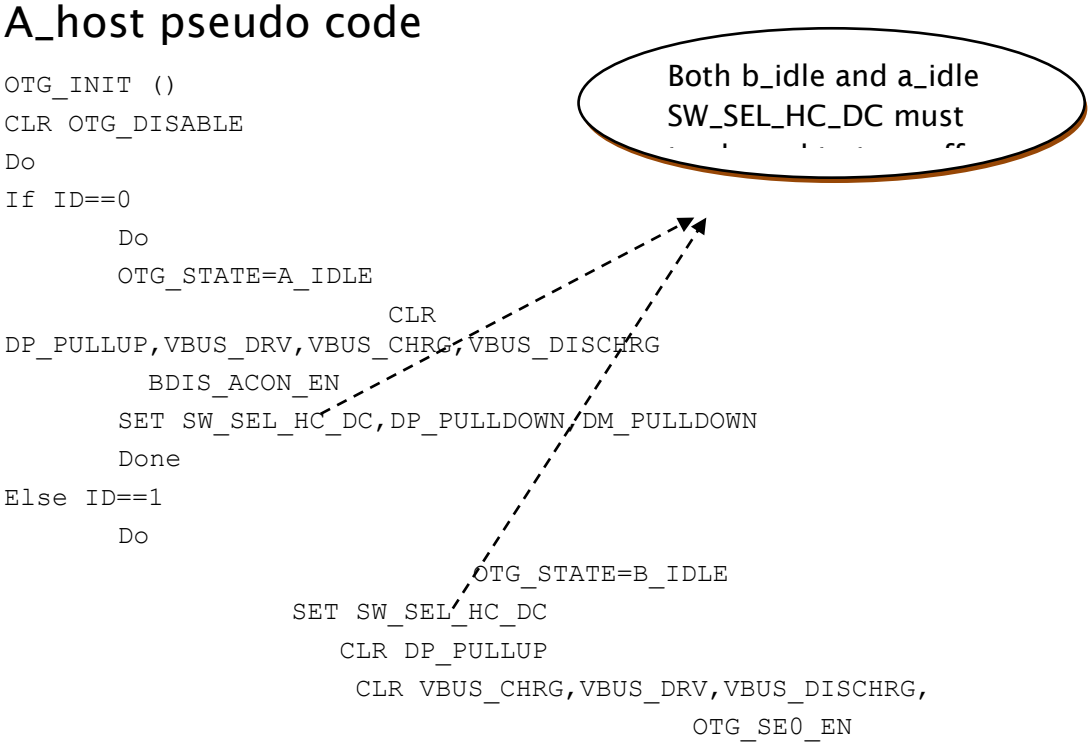


Figure 14 A_host state diagram – part 2

5.7 A_host pseudo code




```
OTGA_EVENT(){
       While(prevState!=OTG_state)
       {
       prevState=OTG_state

Switch (OTG_state){
A_IDLE:

SET   SW_SEL_HC_DC
CLR VBUS_DRV, DP_PULLUP,VBUS_DISCHRG
CLR VBUS_CHRG, BDIS_ACON_EN
If ID==1 {
 OTG_state=B_IDLE
break
}
If DP_SRP ==1 {
  Otg_state= A_WAIT_VRISE;
  Break
```

```
}
Break;

A_WAIT_VRISE:
       If ID==SET {
       Otg_State= A_WAIT_VFALL
       Break;
}

       If VBUS_VLD==1 {
        Otg_state=A_WAIT_BCON;
       Break
}
If TIMEOUT==1 {
                     Otg_state=A_VBUS_ERR
 Break
}
       CLR SW_SEL_HC_DC
       SET VBUS_DRV
                SET A_WAIT_VRISE_TMOUT
       break
A_WAIT_BCON:
       If ID==SET{
       Otg_State= A_WAIT_VFALL
       Break;
}

       If RMT_CONN ==1 {
           Otg_State=A_HOST
     break
}
If TIMEOUT ==1 {
 Otg_Sate= A_WAIT_VFALL
   break
}
SET A_WAIT_BCON_TIMER
Break

A_HOST:
       If ID==SET{
       Otg_State= A_WAIT_VFALL
       Break;
}

       If B_HNP SET {
```

```
            SET BDIS_ACON_EN
            Otg_State= A_SUSPEND
            break
}

A_SUSPEND:
        If ID==SET{
        Otg_State= A_WAIT_VFALL
        Break;
}
        If RMT_CONN !=SET {
        SET SW_SEL_HC_DC
         SET DP_PULLUP
        Otg_state=A_PERIPHERAL
        break
}
If TIMEOUT SET { //device accepted BHNP but not disconnecting
Otg_state= A_WAIT_VFALL
break

}
SET DP_PULLDOWN,DM_PULLDOWN
CLR DP_PULLUP, SW_SEL_HC_DC
SET BDIS_ACON_EN
Set A_WAIT_BDISCON_TIMER

A_PERIPHERAL:
        If ID==SET || a_bus_drop!=SET{
         CLR DP_PULLUP
        Otg_state=A_WAIT_VFALL
        Break;
}
        If BUS== SUSPEND{
        CLR DP_PULLUP
        Otg_State= A_WAIT_VFALL
        Break;
}
SET SW_SEL_HC_DC,DP_PULLUP
A_WAIT_VFALL:
        CLR SW_SEL_HC_DC
        CLR VBUS_DRV VBUS_DISCHRG VBUS_CHRG
        SET DP_PULLDOWN
        Otg_state =A_IDLE
Break
```

```
A_VBUS_ERR:
        CLR SW_SEL_HC_DC
        CLR VBUS_DRV VBUS_DISCHRG VBUS_CHRG
        SET DP_PULLDOWN
        Otg_state = A_WAIT_VFALL
        Break

}
}
}
```

5.8 B-device state diagram

To initialize as a B-device, see Figure 15.

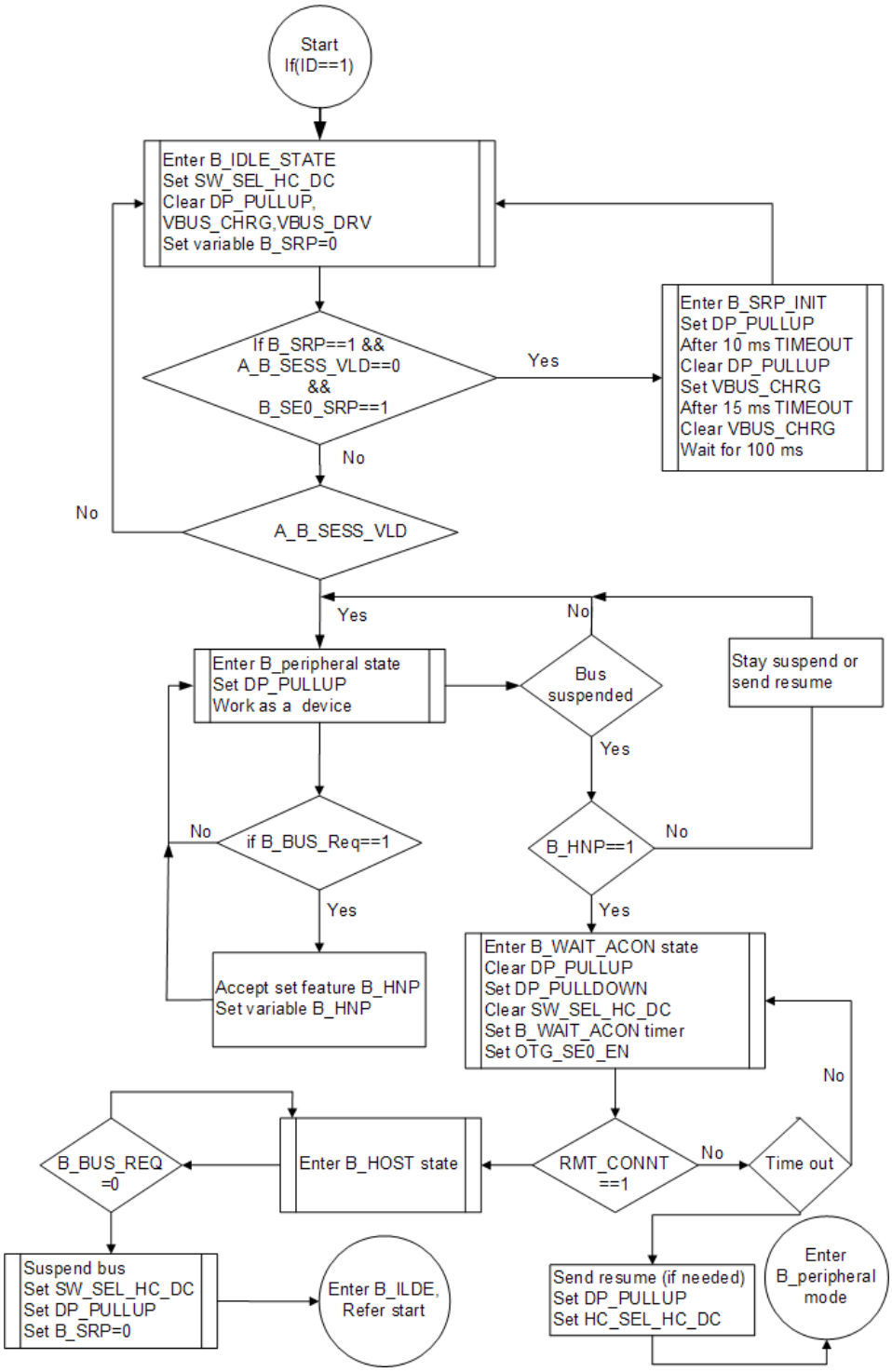


Figure 15 B-device state diagram

5.9 B-device pseudo code

```
OTGB_EVENT(){
        While(prevState!=OTG_state)
        {
        prevState=OTG_state
Switch (OTG_state){
B_IDLE:
        If ID !=SET {
        Otg_state=A_IDLE
        break
}
        If A_B_SESS_VLD ==SET {    // vbus is there
        Otg_state=B_PERIPHERAL
        Break
}
If b_bus_req && B_SE0_SRP==SET {
Otg_state=B_SRP_INIT
Break
}
SET SW_SEL_HC_DC
CLR DP_PULLUP
CLR OTG_SE0_EN VBUS_CHRG
        Break
B_SRP_INIT:
        SET DP_PULLUP
        DELAY 10ms
                    CLR DP_PULLUP
                    SET VBUS_CHRG
        DELAY 10ms
                    CLR VBUS_CHRG
        DELAY 100ms
        Otg_state=B_IDLE

        Break
B_PERIPHERAL:
        If B_SESS_VLD!=SET{
        Otg_state=B_IDLE
        break
}
        If a_bus_suspend==SET && b_hnp_en==SET {
        CLR DP_PULLUP
        SET OTG_SE0_EN
        CLR SW_SEL_HC_DC
        Otg_state= B_WAIT_ACON
```

```
        Break;
}
SET SW_SEL_HC_DC
        SET DP_PULLUP
        Break
B_WAIT_ACON:
        If B_SESS_VLD!=SET{
        Otg_state=B_IDLE
        break
}
If TIMEOUT {
Otg_state= B_PERIPHERAL
break
}
If RMT_CONN== SET {
CLR TIMER
Otg_state= B_HOST
break
}
SET SW_SEL_HC_DC
SET B_WAIT_ACON_TIMER
        Break
B_HOST:
        If B_SESS_VLD!=SET{
          Otg_state=B_IDLE
        break
}
        If b_bus_req!=SET || RMT_CONT!=SET{
          Otg_state=B_PHERIPHERAL
}
CLR SW_SEL_HC_DC
        Break
}
                        }
}
```

5.10 Generating HNP

HNP is used to transfer control of a connection from the default a_host to the default peripheral B-device. This is accomplished by having the A-device condition the B-device to take control of the bus.

The a_host sends SET_FEATURE (b_hnp_enable) command through the control transfer to the B-device. If the B-device accepts it, a_host suspends the bus. Then a_host waits for the B-device to disconnect. If b_peripheral disconnects it, a_host consider as acknowledgment and moves to the a_peripheral state.

The B-device accepts the SET_FEATURE command, if the B-device needs control of the bus. The SET_FEATURE command followed by suspend is an indication to the B-device that a_host wants to give control to the B-device. The B-device disconnects it from the bus when the bus is in the suspend state. It will be an indication to a-host that the B-device accepted the bus control change.

The B-device can stall the SET_FEATURE command when it does not want the control of the bus.

The SET_FEATURE command can be initiated by using either separate thread to poll the SET_FEATURE command with periodic interval or application.

The set-up token of this control transfer is shown in Table 8.

Table 8 Set-up token

bmRequestType	bRequest	wValue	wIndex	wLength	Data
0000 0000b	SET_FEATURE	3 b_hnp_enable	Zero	Zero	None

Glossary

ATL	Asynchronous Transfer List
ATX	Analog USB Transceiver
CPU	Central Processing Unit
DCD	Device (Peripheral) Controller Device
DW	Double Word
EHCI	Enhanced Host Controller Interface
HAL	Hardware Abstraction Layer
HCD	Host Controller Driver
HNP	Host Negotiation Protocol
INT	Interrupt
ISO	Isochronous
MSC	Mass Storage Class
OHCI	Open Host Controller Interface
OTG	On-The-Go
PID	Packet Identifier
PTD	Proprietary Transfer Descriptor
RAM	Random Access Memory
SRP	Session Request Protocol
TT	Transaction Translator
URB	USB Request Block
USB	Universal Serial Bus
USBD	Universal Serial Bus Driver