To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Application Notes

8

7641 Group

USB Application Notes

Renesas Single-Chip Microcomputers

Renesas Electronics
www.renesas.com

Ver.1.3 2006.10

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.

● **About '7641 Group USB Application Notes'**

*Explicit samples for 7641 Group USB implementation

*Programs listed in these notes are examples and should be modified according to the applications

● **Related Materials**

*RENESAS Microcomputers 7641 Group Users manual and Datasheet

[**http://www.renesas.com/en/usb**]

*USB Specification Ver.2.0

[http://www.usb.org/developers/docs.html]

● **Notes on USB Communication**

In applications requiring high-reliability, we recommend providing the system with protective measures such as USB function initialization by software or USB reset by the host to prevent USB communication from being terminated unexpectedly, for example due to external causes such as noise.

Table of Contents

Chapter 1
USB Initialization

In order to use the USB block provided in 7641 Group MCUs, it must be properly enabled before transfers can begin.

1.1 Outline of USB Functions

M37641 comes with a built-in USB function control unit.
The USB function control unit enables efficient data transfer to and from a host PC. This circuit was developed in compliance with USB Specification Version 2.0.
USB Specification Version 2.0 defines the following four transfer types.

- **Control Transfer**
- **Isochronous Transfer**
- **Interrupt Transfer**
- **Bulk Transfer**

The M37641 USB block includes 5 endpoints (Endpoints 0 to 4). Each endpoint is equipped with an IN (send) FIFO and OUT (receive) FIFO. Endpoints 1 through 4 can be used for isochronous, bulk, or interrupt transfers. Endpoint0 can only be used for control transfers, which basically handle data in the same manner as bulk transfers.
Each endpoint is automatically set to bulk transfer mode at reset. Therefore, the user must set or initialize an endpoint for the type of data to be transferred.
USB interrupts include both USB function interrupts and USB SOF interrupts.
The USB block must be enabled in order to use USB functions.

1.2 Setting up USB

To use USB functions, the USB block must be enabled and the necessary endpoints must be initialized.

Table 1.2.1 USB Block Setup: Related Registers

Register Name (abbreviation)	Address (H)
USB control register (USBC)	0013
Clock control register (CCR)	001F
USB endpoint index register (USBINDEX)	0058
USB endpoint FIFO mode register (USBFIFOMR)	005F
Frequency synthesizer control register (FSC)	006C
Frequency synthesizer multiply register 1 (FSM1)	006D
Frequency synthesizer multiply register 2 (FSM2)	006E
Frequency synthesizer divide register (FSD)	006F

Note: All USB related registers (addresses 0050_{16} to 0064_{16}) other than USBC, CCR, and FSC can be accessed after the USB block and USB clock are enabled. Make sure this procedure is properly performed during USB block initialization or USB resume operation.

1.2.1 Enabling the USB block

Always enable the USB block before accessing to USB-related registers. Use the following steps to enable the USB block after hardware reset.

1. Set CCR Xin divider select bit to “1”, which selects f(Xin). Set the frequency synthesizer related registers (FSM1, FSM2, FSD) to generate the 48MHz USB clock. After setting these registers, enable the frequency synthesizer by setting the frequency synthesizer input selector bit to “0” (f(XIN)) and the frequency synthesizer enable bit to “1”. At this point, the fPIN should be set to more than 1MHz.
2. To stabilize the frequency synthesizer, wait for at least 2ms before going to the next step.
3. Using the frequency synthesizer lock status bit (FSC), confirm that the frequency synthesizer is in the locked state. If not, check it every 0.1ms. Do not go to the next step until it is locked.
4. Set the LPF current control bits (bits 5, 6) of the FSC to “10” (intermediate current).
5. When Vcc = 5V:
   The built-in DC-DC converter must be used, set the USB line driver supply selection bit of the USBC register to “1”. Set the transceiver voltage converter High/Low current mode selection bit (USBC3) to “0”, which enables the USB line driver to the high current mode.
   USBC3 should be set to”0” in the normal state and to “1” in the suspend state.
   When Vcc = 3V:
   Disable the built-in DC-DC converter. Set the USB transceiver voltage converter enable bit (USBC4) to “0”.
6. Include a delay of (C +1)ms for the voltage on the Ext. Cap pin to stabilize. “ C” indicates the size (1ms/ µ F) of the capacitor connected to the Ext. Cap pin. For example, 2µ2 F and 0µ1 F capacitors connected to the Ext. Cap pin in parallel would require a (2.3+1) ms wait.
7. Enable a 48MHz clock supply to the USB block. Set the USB clock enable bit of the USBC register to “1” and then wait 4 or more ø cycles.
8. Set the USB enable bit of the USBC to “1”. Then wait 250ns or more before accessing other USB-related registers.

1.2.2 Endpoint initialization

Select an endpoint that you are going to configure with the USB index register. Then set the maximum IN/OUT packet size and the transfer type for this endpoint. You will need to perform this operation for each endpoint to use. When using either Endpoint 1 or 2, the IN/OUT FIFO size is variable. Determine the FIFO size with the USB endpoint FIFO mode register. Enable each USB interrupt as necessary.

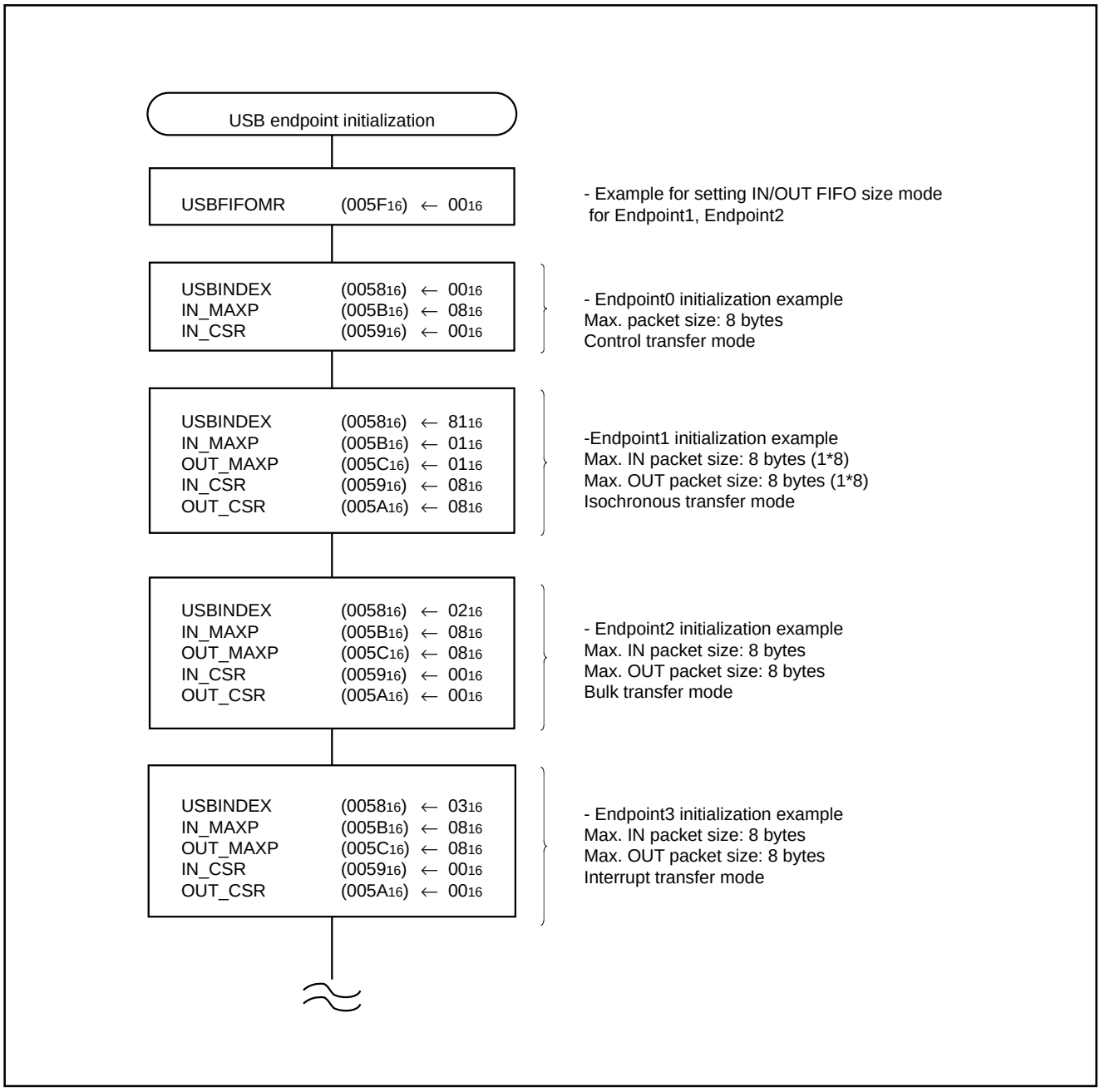


Fig.1.2.1 Endpoint Initialization Example

1.2.3 USB block disabling procedure

Use the following procedure to disable the USB block for systems that require the USB functions to be disabled after the USB function control unit has been enabled.

1: Disable the USB block by clearing the USB enable bit (USBC7) to “0”.

2: Set the transceiver voltage converter High/Low current mode selection bit (USBC3) to “1” (low current mode).

3: Disable the USB clock by clearing the USB clock enable bit (USBC5) to “0”.

4: Disable the USB line driver by clearing the USB transceiver voltage converter enable bit (USBC4) to “0”.

5: Disable the frequency synthesizer by clearing the frequency synthesizer enable bit (FSE) to “0”.

Note that if f_{SYN} is selected as the clock source (CPMA6), stopping the PLL will cause the device to stop functioning as well.

Normally, it is not necessary to disable the USB function control unit in systems that are designed for USB functions to be continuously enabled. For example, in order to put the USB block in the suspend state without receiving a suspend signal, disable the USB block. Executing the STP instruction alone will not put the USB block in the suspend state, and will not result in low current operation. Be aware that, in this case, USB-related interrupts cannot be used to recover from the stop mode.

1.2.4 Re-enumeration during USB transfers

When you need to re-enumerate during a USB transfer, you must disable the USB block once, then re-enable it again. The following steps show this procedure. Also see Chapter 2 for more information concerning enumeration.

1: Disable the USB block by clearing the USB enable bit (USBC7) to “0”.

2: Disable the USB clock by clearing the USB clock enable bit (USBC5) to “0”.

3: Disable the internal DC-DC converter by clearing the USB line driver by clearing the USB transceiver voltage converter enable bit (USBC4) to “0”.

Note: Step 3 is not necessary when Vcc = 3.3V.

4: Wait for 2 to 5 ms for the D+ discharge. The length of the wait period depends on the system. Please fully evaluate your target system.

5: Enable the internal DC-DC converter by setting USBC4 to “1”.

Note: Step 5 is not necessary when Vcc = 3.3V.

6: Wait for (C + 1)ms. C is the value (µ F) of the external capacitor on the Ext. Cap pin.

Example: When Ext. Cap pin is connected to 2.2µF and 0.1µ F, {(2.2 + 0.1) + 1} = 3.3ms.

7: Enable the USB clock by setting USBC5 to “1”.

8: Wait for 4 to 5 ø cycles. (ø: internal operating clock)

9: Enable the USB block by setting USBC7 to “1”.

Chapter 2
State Transition and USB Interrupts

M37641 has these USB related interrupts:
USB function interrupt and USB SOF interrupt.
This chapter discusses both USB state transitions and USB interrupts.

2.1 USB Enumeration Operations

When connecting/disconnecting a USB device to/from a host PC enumeration is used to, identify the device and manage device state changes.
Figures 2.1.1 to 2.1.3 flowcharts describe basic operations, firmware processing, and device states from the point of USB connector insertion to the host PC until the USB device can begin transfer operations.

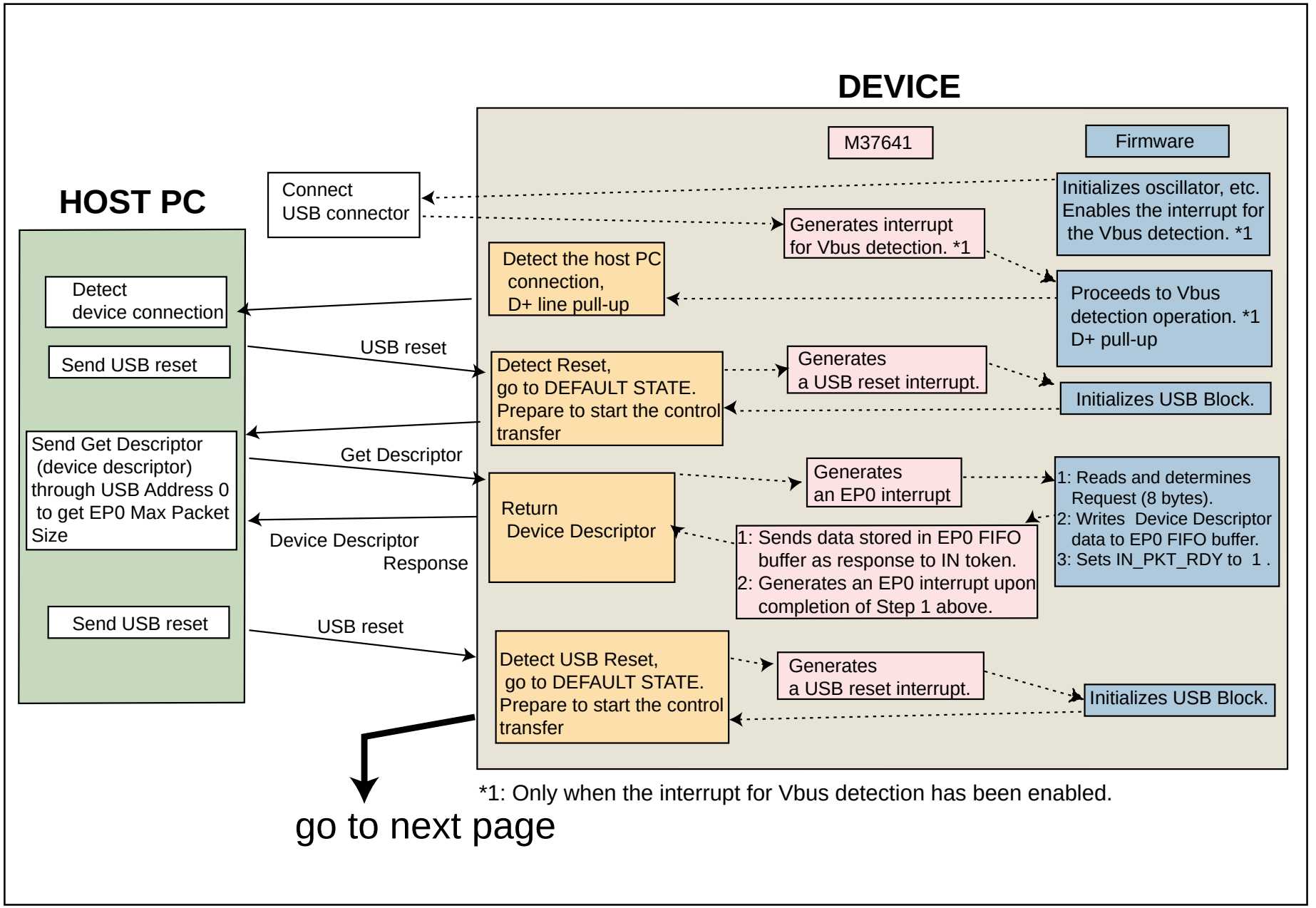


Fig.2.1.1 USB Enumeration Process

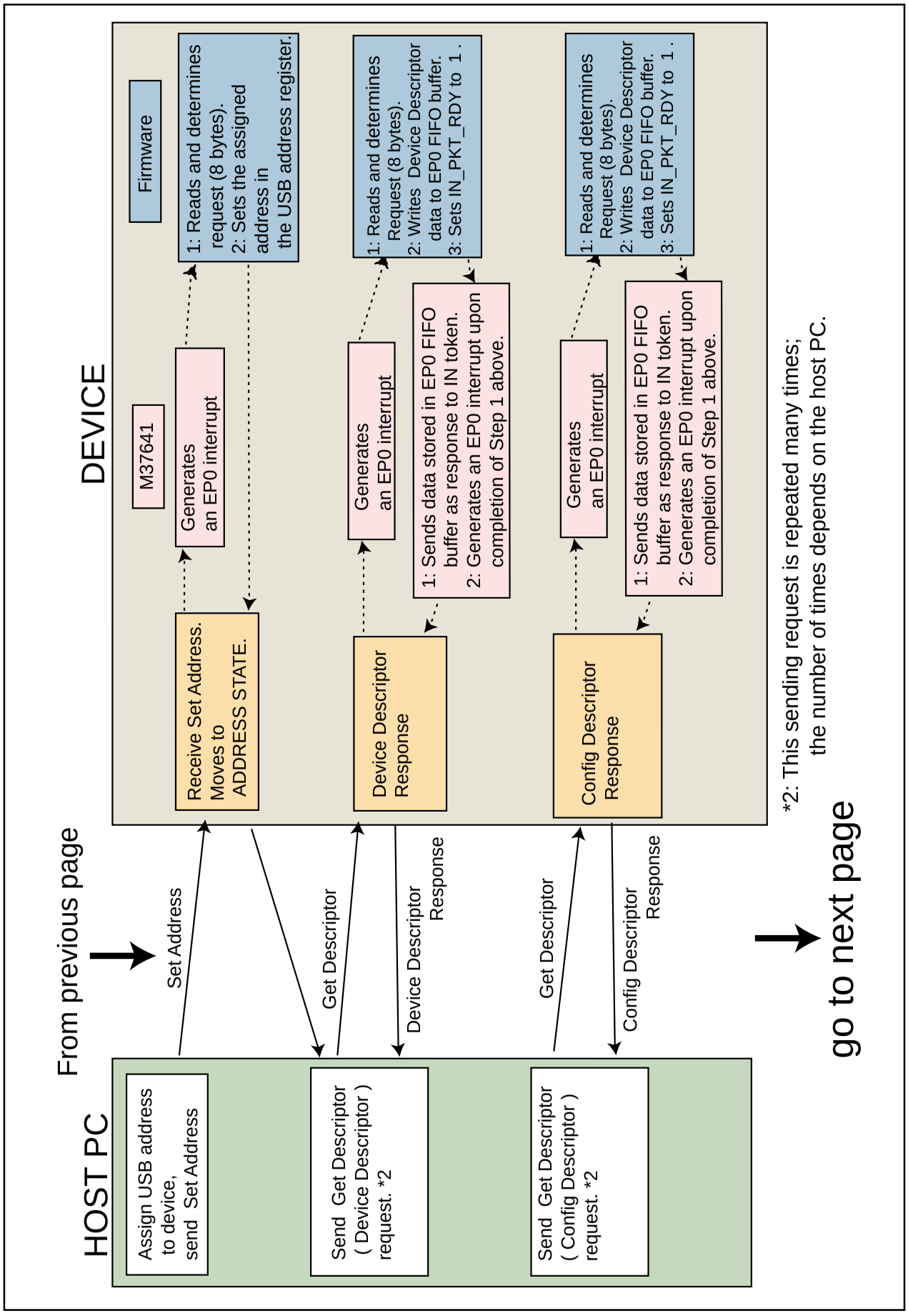


Fig.2.1.2 USB Enumeration Process continued

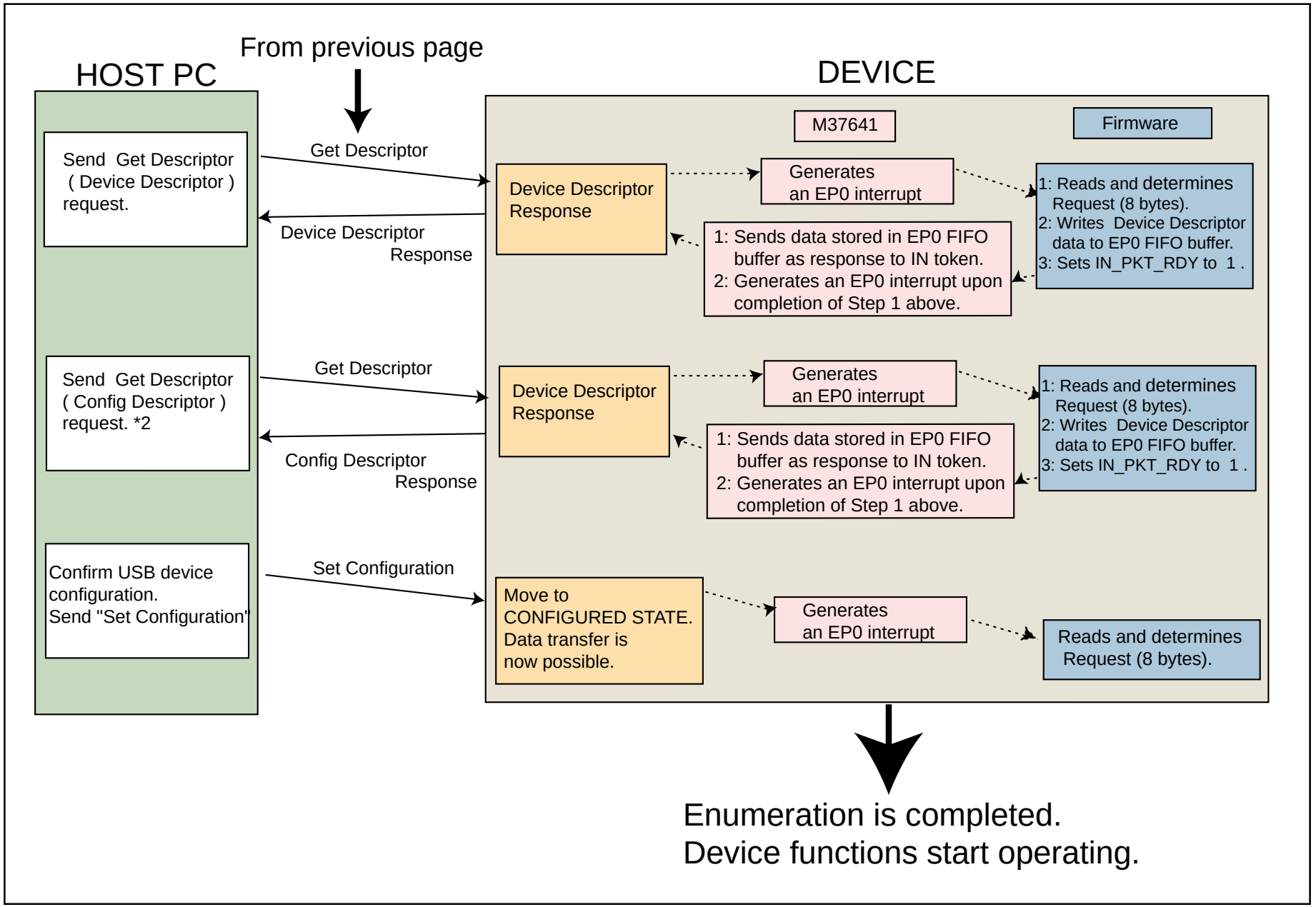


Fig.2.1.3 USB Enumeration Process continued

2.2 Device State Transition

The USB device moves from one state to another according to current operations. In M37641, device state transition occurs with a signal interrupt request (reset, suspend, resume) or an Endpoint0 device standard request.

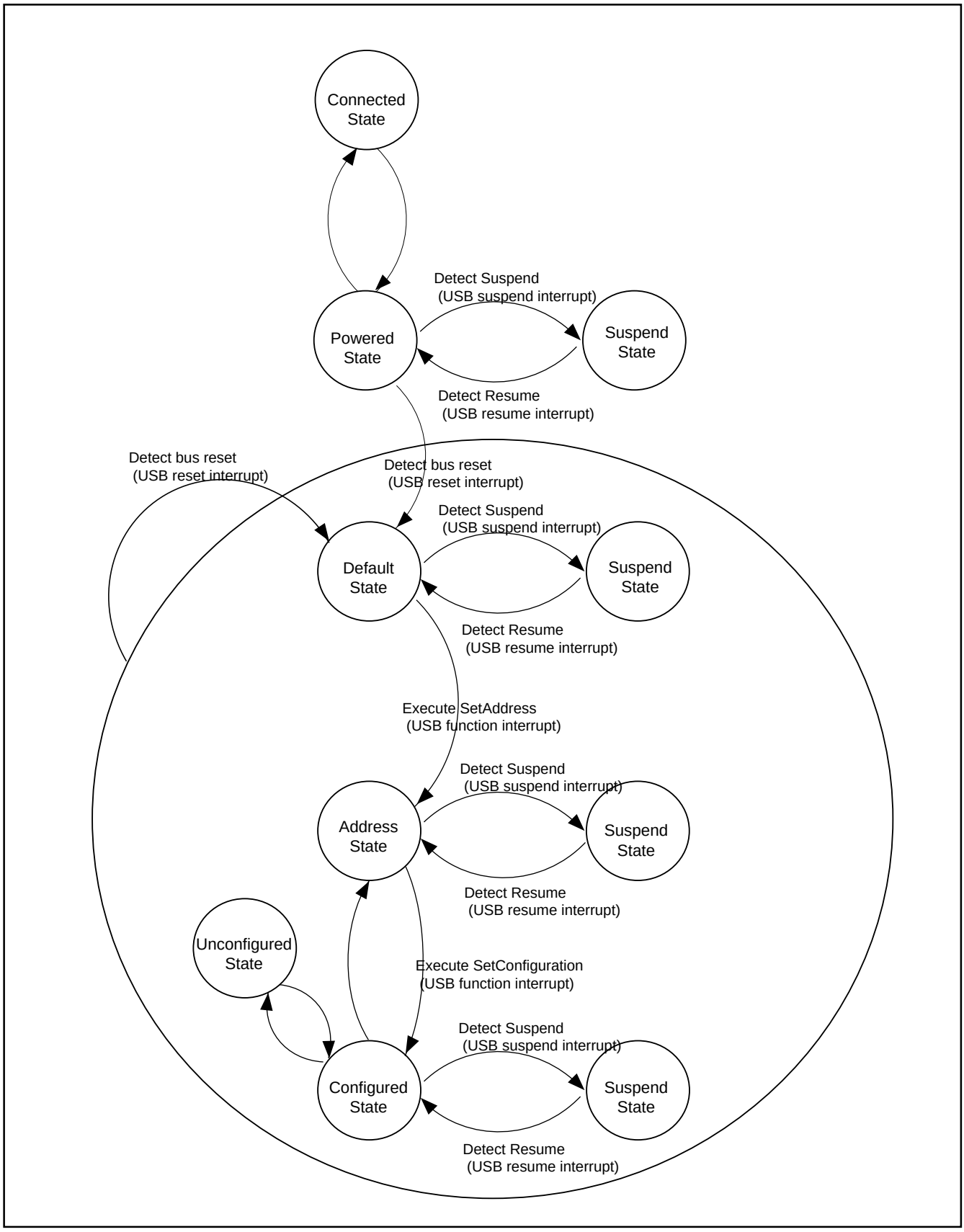


Fig.2.2.1 USB Device State Transition

2.3 USB Function Interrupts

The USB function interrupts are used for data flow control as well as USB power management. The interrupts available are as follows:

- Interrupt request generated when data send/receive is complete.
  Endpoint0 interrupt, Endpointx IN (x = 1 to 4) interrupt, Endpointx OUT (x = 1 to 4) interrupt
- Interrupt request generated when an overrun/underrun occurs due to an isochronous transfer
  Overrun/underrun interrupt
- Interrupt request generated when a USB reset signal is detected
  USB reset interrupt
- Interrupt request generated when a USB suspend signal is detected.
  USB suspend interrupt
- Interrupt request generated when a USB resume signal is detected.
  USB resume interrupt

In order to enable a USB function interrupt, set the following bits to “1”: USB function interrupt enable bit of interrupt control register A (address 0005_{16}), and the bits corresponding to USB interrupt enable register 1 (address 0054_{16}) and USB interrupt enable register 2 (address 0055_{16}). The USB reset interrupt is in the enabled status by default. The USB interrupt status flags of USB interrupt status registers 1 and 2 display the interrupt request status.
Table 2.3.1 shows registers related to USB function interrupts and Figure 2.3.1 shows an example of the USB function interrupt routine. Each USB function interrupt is called from the flow as shown in the example, shown in Fig. 2.3.1.

Table 2.3.1 USB Interrupt-Related Registers

Register name (abbreviation)	Address (H)
Interrupt request register A (IREQA)	0002
Interrupt control register A (ICONA)	0005
USB control register (USBC)	0013
Clock control register (CCR)	001F
USB power management register (USBPM)	0051
USB interrupt status register 1 (USBIS1)	0052
USB interrupt status register 2 (USBIS2)	0053
USB interrupt enable register 1 (USBIE1)	0054
USB interrupt enable register 2 (USBIE2)	0055
USB frame number register Low (USBSOFL)	0056
USB frame number register High (USBSOFH)	0057

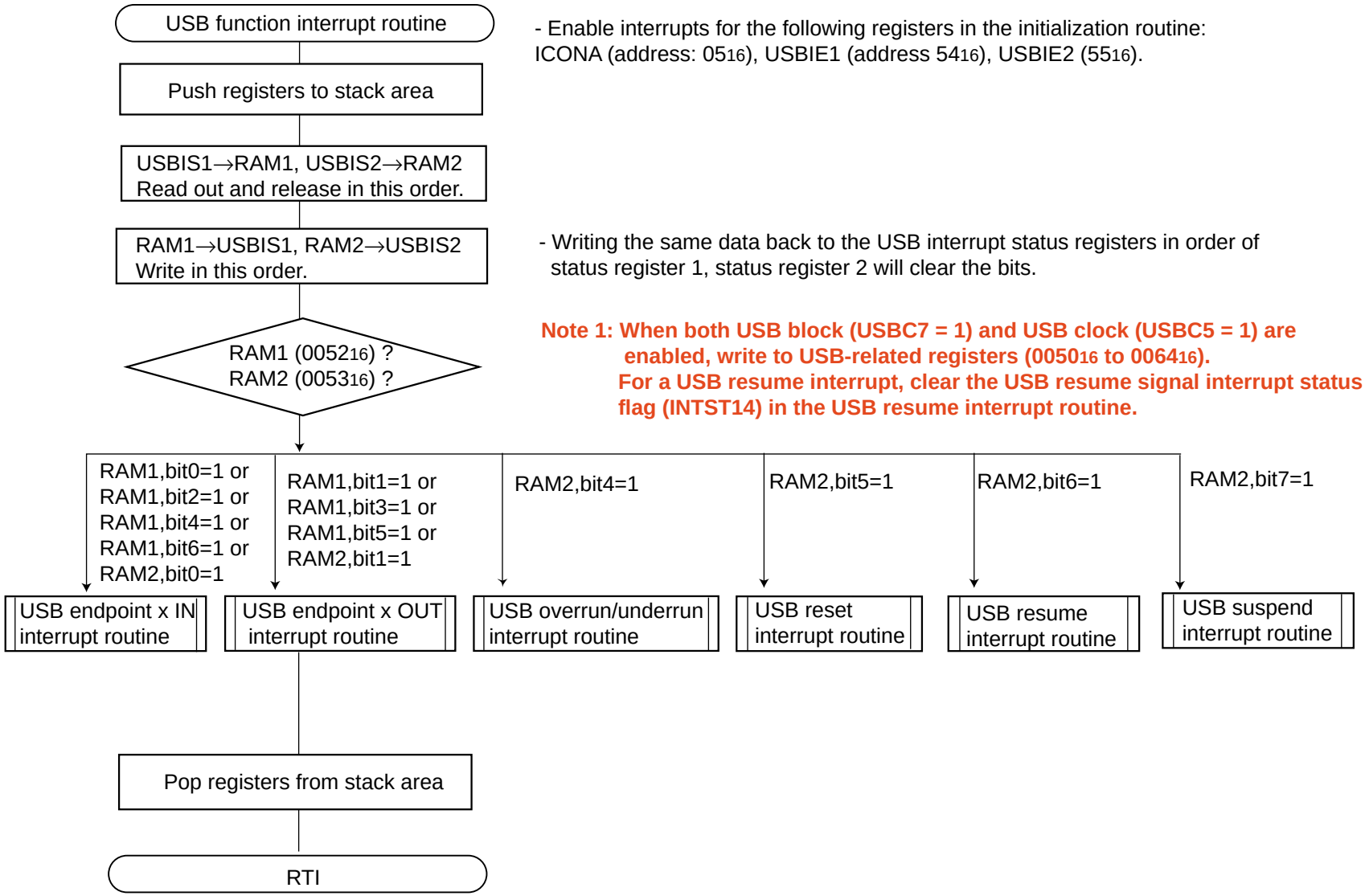


Fig.2.3.1 USB Function Interrupt Routine Example

2.3.1 Interrupts generated by transmission completion or overrun/underrun

Interrupts generated by data transmit/receive completion or overrun/underrun are described below.
The USB interrupt status flags for USB interrupt status registers 1 and 2 are set to “1” when one of the listed conditions occurs. When a USB interrupt status flag is set to “1”, the corresponding interrupt request is generated. See Chapter 4 for more details.

Interrupt requests generated by data transmit/receive completion

- Endpoint0 interrupts

Endpoint0 1 packet of data successfully transmitted
Endpoint0 1 packet of data successfully received
Endpoint0 DATA_END bit = “0”
Endpoint0 FORCE_STALL = “1”
Endpoint0 SETUP_END = “1”

- Endpointx IN (x = 1 to 4) interrupts

Endpointx (x = 1 to 4) 1 packet of data successfully transmitted
Endpointx (x = 1 to 4) UNDER_RUN = “1”

-Endpointx OUT (x = 1 to 4) interrupts

Endpointx (x = 1 to 4) 1 packet data successfully received
Endpointx (x = 1 to 4) OVER_RUN = “1”
Endpointx (x = 1 to 4) FORCE_STALL = “1”

Interrupt requests generated by overrun/underrun

- Overrun/underrun interrupt

Data contained in FIFO overruns/underruns during isochronous transfer

When all of the following conditions are met, any of the interrupts listed above will be generated.
- Interrupt disable flag (I flag) is “0” (interrupt enabled)
- ICONA USB function interrupt enable bit is “1”
- The corresponding interrupt enable bit of USBIE1, USBIE2 is “1”
- The corresponding status flag of USBIS1, USBIS2 is “1”

2.3.2 USB reset

At USB reset, all USB internal registers (addresses 0050_{16} to $005E_{16}$) other than the USB reset interrupt status flag return to their default settings.

(1) Cause of USB reset detection

A USB reset is detected in M37641 when the USB function control unit detects an “L” level in the D+/D- line for at least 2.5us.

(2) When a USB reset is detected

When a USB reset is detected as described above, the USB reset interrupt status flag of USB interrupt status register 2 becomes a “1” and a USB reset interrupt request is generated.

(3) USB reset interrupt settings

The USB reset interrupt is enabled at default.

- Set the USB function interrupt enable bit of interrupt control register A to “1”.

(4) Conditions for receiving a USB reset interrupt

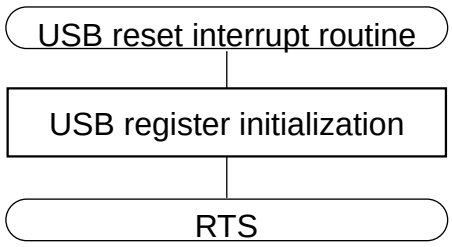
A USB reset interrupt will be generated when all of the following conditions are met:

- The interrupt disable flag (I flag) of the processor status register is “0” (interrupt enabled).
- The USB function interrupt enable bit of interrupt control register A is “1” (USB function interrupt enabled).
- The USB function interrupt request bit of interrupt request register A is “1” (USB function interrupt request issued).
- The USB reset interrupt status flag of USB interrupt status register 2 is “1”.

(5) USB reset interrupt process

Figure 2.3.2 shows an example of the USB reset interrupt process.

When a USB reset interrupt is received, the USB reset interrupt status flag of USB interrupt status register 2 is cleared (by writing a “1” to the flag, as shown in the USB function interrupt routine example in Fig. 2.3.1).



*This routine is a subroutine called by the USB function interrupt routine example shown in Fig.2.3.1.

Fig.2.3.2 USB Reset Interrupt Routine Example

2.3.3 USB suspend

(1) What is a USB Suspend?

Suspend is where the device powers down due to a USB-related signal. In the USB suspend state, the total drive current should be brought to a low-power consumption of 500μA or less. In this state, the USB clock will not oscillate but the USB block will remain enabled.

(2) Cause of USB suspend detection

USB suspend is detected when the D+ line is “H”, the D- line is “L” and no activity is detected for a period of 3ms.

(3) When a USB suspend is detected

When a USB suspend is detected as described above, the USB suspend detection flag (SUSPEND) of the USB power management register (address 0051_{16}) and the USB suspend signal interrupt status flag (INTST15) of USB interrupt status register 2 are set to “1” automatically, and a USB suspend interrupt request is generated.

(4) USB suspend interrupt settings

- Set the USB function interrupt enable bit of interrupt control register A to “1”.
- Set the USB suspend/resume interrupt enable bit of USB interrupt enable register 2 to “1”.

(5) Conditions for receiving a USB suspend interrupt

A USB suspend interrupt will be generated when all of the following conditions are met:

- The interrupt disable flag (I flag) of the processor status register is “0” (interrupt enabled).
- The USB function interrupt enable bit of interrupt control register A is “1”.
- The USB function interrupt request bit of interrupt request register A is “1”.
- The USB suspend/resume interrupt enable flag of USB interrupt enable register 2 is “1”.
- The USB suspend detection flag of the USB power management register is “1”.
- The USB suspend signal interrupt status flag of USB interrupt status register 2 is “1”.

(6) How to return from a USB suspend state

The MCU is returned from a USB suspend state by a USB resume interrupt or remote wake up.
The following interrupts can be used for remote wake up.

- INT0, INT1
- CNTR0, CNTR1
- Timers that use an external clock (Timers X, Y)
- Serial I/O that uses an external clock
- Key-on wake-up

Set one of the above wake-up factors in the USB suspend interrupt routine.

● USB Suspend interrupt processing

Figure 2.3.3 shows an example of the USB Suspend interrupt routine.

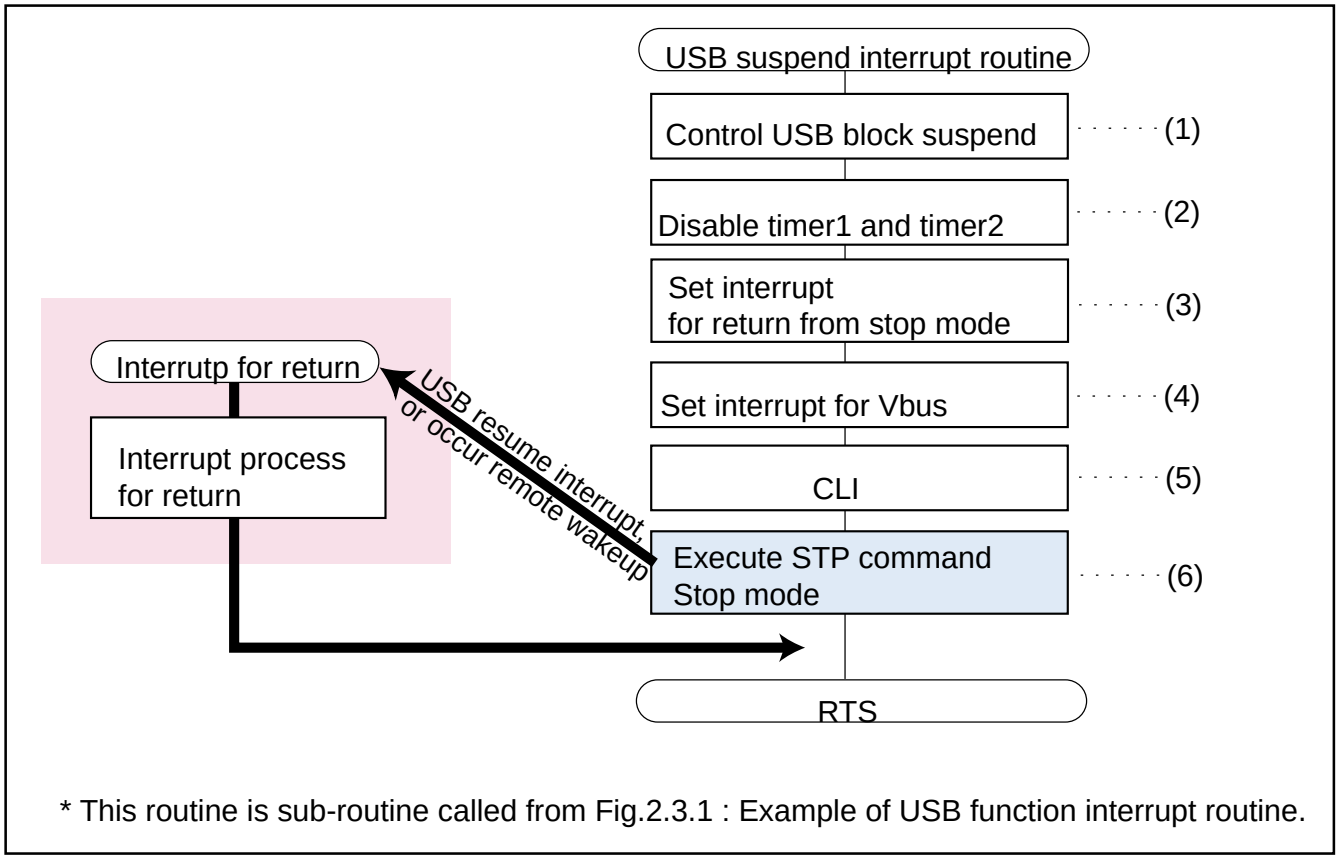


Fig. 2.3.3 USB Suspend Interrupt Routine Example

The following is an explanation of Steps 1 through 6 in Figure 2.3.3.

● The USB suspend interrupt status flag is cleared by re-writing a “1” to this flag by software. (See Fig. 2.3.1 for details.)

(1) USB block suspend control in the stop mode is performed as follows:

1. Disable the 48MHz clock supplied to the USB block by setting the USB clock enable bit of USBC to “0”.
2. Disable the frequency synthesizer by setting the frequency synthesizer enable bit to “0”.
3. Set the low current mode by setting the transceiver voltage converter High/Low current mode selection bit (USBC3) to “1”. This setting is not necessary when Vcc = 3.3V, as the internal DC-DC converter is not used.
4. Lower the total drive current to less than 500μA.

(2) Disable Timers 1 and 2, which M37641 uses for oscillation stabilization.

1. Set Timer 1 (interrupt enable bit) and Timer 2 (interrupt enable bit) of interrupt control register B to “0”.
2. Set Timer 1 (interrupt request bit) and Timer 2 (interrupt request bit) of interrupt request register B to “0”.

(3) Set the interrupt to return from the stop mode. There are several interrupts that can be used for returning to the normal process flow from within the USB suspend interrupt processing routine itself.

● Follow these steps when using the USB Resume interrupt to return from a suspend state. However, do not disable the USB suspend/resume interrupt enable bit (USBIE2, bit 7)

1. Set the USB function interrupt request bit of interrupt request register A to “0” (no request).
2. Set the USB function interrupt enable bit of interrupt enable register A to “1” (interrupt enabled).

● Enable the applicable interrupt when using a remote wake-up to return from a suspend state.

1. Set the interrupt edge (only necessary to set interrupts INT0, INT1, CNTR0, CNTR1).
2. Set the using interrupt request bits of interrupt request registers A to C to “0” (no request).
3. Set the using interrupt enable bits of interrupt control registers A to C to “1” (interrupt enabled).

(4) Set the Vbus detection interrupt. This setting is only necessary when supplying power through the self/bus powered switch system in a device that requires detection of USB cable removal. Refer to descriptions concerning Vbus detection in Section 3.2.

1. Set the using interrupt request bit of the interrupt request register to “0” (no request).
2. Set the using interrupt enable bit of the interrupt control register to “1” (interrupt enabled).

(5) Enable the interrupt by setting the interrupt enable flag (I flag) to “0”.

(6) Execute the STP instruction and return to the low-power consumption mode.

2.3.4 USB resume

Return from the suspend state can be done using the USB resume interrupt or the remote wake-up method. This section describes both methods. The remote wake-up method is basically for returning from suspend state by removing the USB cable or using a key-on wakeup.

● Returning from the suspend state with the USB resume interrupt

(1) What is a USB Resume?

Resume is when the USB resume interrupt is used to return M37641 from the suspend state.
The USB resume interrupt is generated when a change on the D+/D- line is detected, after suspend.

(2) Cause of USB resume detection

A resume is generated when activity from the host PC on the D+/D- line is detected while the M37641 is in a suspend state.

(3) When a USB resume is detected

When a USB resume is detected as described above, the following bits are set automatically and a USB resume interrupt request is generated. When the USB resume detection flag becomes a “1”, the USB suspend detection flag is automatically cleared.

- USB resume detection flag of USB power management register is “1”.
- USB resume signal interrupt status flag of USB interrupt status register 2 is “1”.
- USB suspend detection flag of USB power management register is “0”.

(4) USB resume interrupt settings

- Set USB function interrupt enable bit of interrupt control register A to “1”.
- Set USB suspend/resume interrupt enable bit of USB interrupt enable register 2 to “1”.

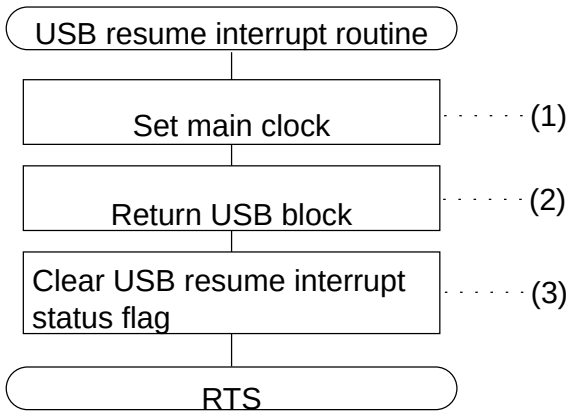
(5) Conditions for receiving a USB resume interrupt

A USB resume interrupt will be generated when all of the following conditions are met:

- The interrupt disable flag (I flag) of the processor status register is “0” (interrupt enabled).
- The USB function interrupt enable bit of interrupt control register A is “1”.
- The USB function interrupt request bit of interrupt request register A is “1”.
- The USB suspend/resume interrupt enable bit of USB interrupt enable register 2 is “1”.
- The USB resume signal interrupt status flag of USB interrupt status register 2 is “1”.

(6) USB resume interrupt processing

A processing example of when a USB resume interrupt is received is shown in Figure 2.3.4.



* This routine is sub-routine called from Fig.2.3.1 : Example of USB function interrupt routine.

Figure 2.3.4 USB Resume Interrupt Routine Example

When a return interrupt is received, oscillation restarts and the oscillation stabilization period of internal clock ø is automatically generated for both Timer 1 and Timer 2 (refer to M37641 data sheet, “Clock Generating Circuit” for more information).

The following is a description of Steps 1 through 3 of the USB resume interrupt processing routine example shown in Fig. 2.3.4.

(1) Setting the clock after the oscillation stabilization period (internal clock ø as described above)

- When Vcc = 5V

Set the XIN divider select bit of the clock control register. Executing a STP instruction sets the XIN divider select bit to “0”.

- When Vcc = 3V

Set ø to 6MHz or less. When f(XIN) = 24MHz, set “0” to the XIN divider select bit of the clock control register and select f(XIN)/2 as the system clock.

(2) Return USB block to normal operation mode. To return the USB block to normal mode, perform the following procedure.

1. Set the MCU to high current mode by setting the transceiver voltage converter High/Low current mode selection bit (USBC3) to “0”. (This setting is not necessary when Vcc = 3.3V, as the internal DC-DC converter not used.)
2. Set the frequency synthesizer enable bit to “1”. Wait 2ms or more for the frequency synthesizer to stabilize.
3. Check the frequency synthesizer lock status bit of FSC. If “0”, recheck every 0.1ms. Continue with the procedure only after the bit goes to “1”, the lock status.
4. Enable the 48MHz clock supply to the USB block. Set the USB clock enable bit to “1” and wait at least 4 ø cycles.

(3) Write a “1” to the USB resume signal interrupt status flag of USB interrupt status register 2 to clear the flag. At this time, the USB resume detection flag of the USB power management register will automatically be cleared to “0”.

● Returning from the suspend state with remote wake-up

When returning from the suspend state using remote wake-up, include the following procedure in the routine used for the remote wake-up.

(1) Set the clock after an oscillation stabilization wait period of internal clock ø (same procedure as that of USB resume interrupt processing (1)).

(2) Perform USB block return processing (same procedure as that of USB resume interrupt processing (2)).

(3) Disable the interrupt used for the remote wake-up that is now enabled. Set the interrupt enable bit that corresponds to the interrupt control register to “0” and the interrupt request bit that corresponds to the interrupt request register to “0”.

(4) Set the USB remote wake-up bit of the USB power management register to “1” and send the resume signal to the host PC. Maintain this status for 10ms to 15ms.

(5) Clear both the USB remote wake-up bit and the USB suspend detection flag of the USB power management register, which completes the transmission of the resume signal.

● Differences between USB resume interrupt and remote wake-up return processes

- USB resume interrupt:

  When the USB resume detection flag of the USB power management register is set to “1”, the USB suspend detection flag is automatically cleared to “0”.

- Remote wake-up:

  The USB suspend detection flag is not automatically cleared, and must be cleared to “0” by software. First set the remote wake-up bit of the USB power management register to “1” for 10 to 15ms. After transmitting the resume signal to the host PC, write a “0” to the bit to clear it.

2.4 USB SOF Interrupt

This interrupt is used for isochronous transfers.

(1) Cause of USB SOF interrupt request generation

A USB SOF interrupt request is generated when an SOF packet is received from the host by the USB function control unit. When the Artificial SOF enable bit of the USB control register is “1” and an SOF packet sent from the host PC is destroyed for some reason within 250ns of the original SOF packet, an SOF interrupt is generated so that the frame can be synchronized.

(2) 7641 Group operations when USB SOF interrupt is received

The frame numbers (11 bits) of the SOF packet received from the host are stored in frame number registers Low and High automatically.

(3) Setting up the USB SOF interrupt

- Set USB SOF interrupt enable bit of interrupt control register A to “1”.

(4) Conditions for receiving a USB SOF interrupt

A USB SOF interrupt will be received if all of the following conditions are met:

- Interrupt disable flag (I flag) of the processor status register is “0”.
- The USB SOF interrupt enable bit of interrupt control register A is “1”.
- The USB SOF interrupt request bit of interrupt request register A is “1”.

(5) USB SOF signal output

For isochronous transfers, pin $P7_0$ can be used as the SOF output pin by setting the USB SOF port select bit of the USB control register to “1” (set $P7_0$ to output). Each time an SOF signal is received from the host, an 83ns (ø= 12MHz) period of “L” is output from $P7_0$.

Chapter 3

Power Supply Control and USB Additional Circuitry

This chapter describes the differences in external power supply circuits using various power supply methods, how to detect Vbus and USB additional circuitry.

3.1 Power Supply Control

3.1.1 Power supply

M37641 can be run on either 5V (X_{IN} = 24MHz: ø = 12MHz) or 3.3V (X_{IN} = 24MHz: ø = 6MHz) power supply voltage. The USB function control unit (USB FCU) operates at 3.3V.
There are three methods for supplying power to a USB device.

- Self-Power:
  Used when power is supplied to the M37641 Vcc through a power connector, battery, etc.
- Bus-Power:
  Used when power is supplied from the host PC through Vbus line.
- Self/Bus Power Switching:
  Used in order to reduce battery consumption. Power is supplied through bus-power only when the device is connected to the host PC and power is supplied through Vbus.

3.1.2 External power supply circuit

The external power supply circuit for M37641 will differ according to the method of supplying power. In 3.3V operations, when either the bus-power or self/bus combination method is used to supply power to M37641, the voltage from the Vbus is dropped to 3.3V and therefore requires the connection of a regulator, etc. In addition, when using the self/bus power switching system in a system design that requires detection of the insertion/removal of the USB cable, detecting the Vbus will require additional circuitry and software processing (for more details see Section 3.2)
Figure 3.1.1 shows the block diagram of the Vcc pin area.

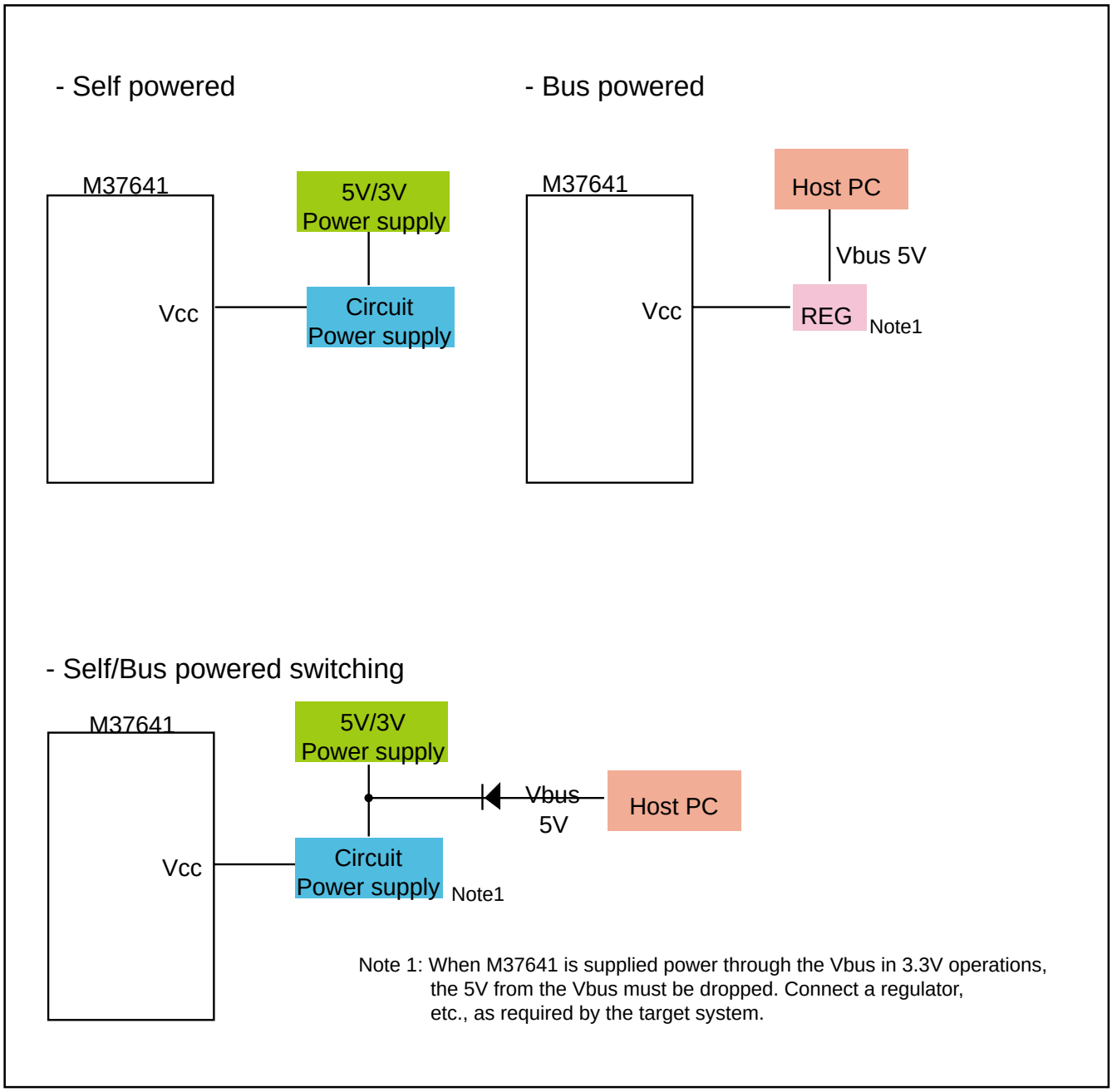


Fig.3.1.1 Block Diagram of Power Supply Options

3.1.3 Notes concerning power supply pin

- Connect ferrite beads between the AVss and Vss pins or the AVcc and Vcc pins, as necessary. The connection is shown in Figure 3.1.2.
- Connect 0.1µF and 4.7µF capacitors in parallel between the Vss and Vcc pins, and the Avss and Avcc pins, respectively. Connect the capacitors with the shortest wiring possible, and avoid placing components around the LPF pin (for noise prevention).
- Use wider wiring for Vcc and Vss lines than that of other signal lines.

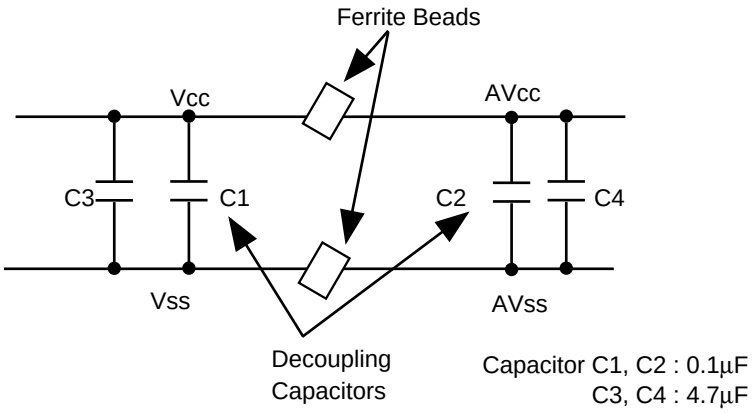


Fig.3.1.2 Example Connection of Insulated Connector

3.2 USB Cable Connect/Disconnect

3.2.1 Standby current protection: when Vbus detection is necessary

In a self/bus power switching system, bus-power is used when power can be supplied through the USB cable Vbus line from the host PC, and self-power is used when the Vbus cannot supply the power. The user must add software for detecting a change in the power supply for systems in which detection of USB cable connect/ disconnect between the host PC and the device is required. This kind of system, used for digital cameras and other applications, recognizes the change in the Vbus as an indication of connection/disconnection of the device.

3.2.2 Vbus detection

Although there are many methods for detecting changes in the Vbus status, this section describes the most widely used method: connecting the Vbus to the INT pin on M37641.

(1) Setting the interrupt

The INT pin of M37641 is connected to the Vbus as shown in Figure 3.2.1. Status changes in either pin INT_0 or pin INT_1 (used as Vbus input pins) are detected by interrupts INT0 or INT1, respectively. “L” to “H” change indicates connection of the USB cable; conversely, “H” to “L” change indicates the cable has been disconnected. Detection of a rising edge or falling edge is determined by setting the INT_0 interrupt edge select bit or the INT_1 interrupt edge selection bit of the interrupt polarity selection register (address 0011_{16}). By setting the active edge, the interrupt request bit is also automatically set. After disabling the interrupt and setting the interrupt edge, clear the interrupt request bit and enable the interrupt.

(2) Vbus detection process

Detection of Vbus activity is processed as shown in the flowcharts in Figure 3.2.2.
In order to ensure proper signaling, wait a few milliseconds for the Vbus state to stabilize.

- When USB cable is connected

Perform power supply process through the Vbus. When USB block is disabled, initialize USB block. After that, perform enumeration process.

- When USB cable has been disconnected

Follow the flowchart shown in Figure 3.2.2: disable USB block, disable USB clock supply.

(3) Vbus detection in USB suspend state

An INT interrupt will be generated even if the M37641 internal clock supply has been stopped.
Therefore, when in the suspend state while the internal clock supply is stopped, detect Vbus input only after returning the MCU from the suspend state and enabling the internal clocks. Add the Vbus detection interrupt setting in the suspend interrupt routine after the interrupt that returns the MCU from suspend state. The Vbus detection interrupt consists of setting the bit corresponding to the interrupt request register to “no request” and the bit corresponding to the interrupt control register to “enable”.

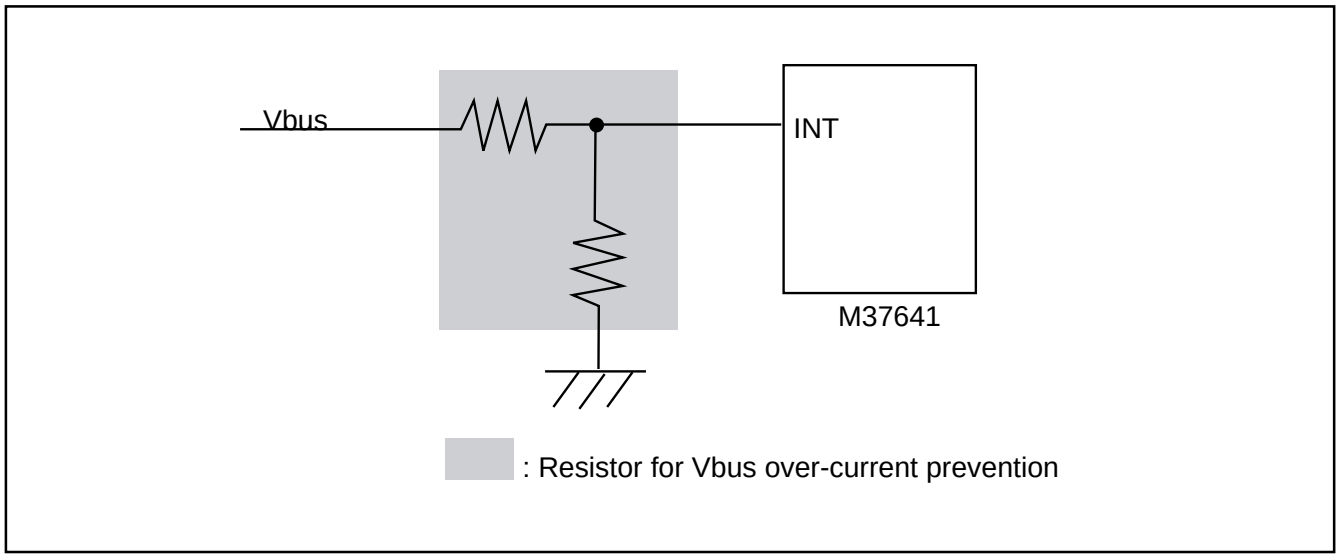


Fig.3.2.1 Vbus Detection Circuit Example

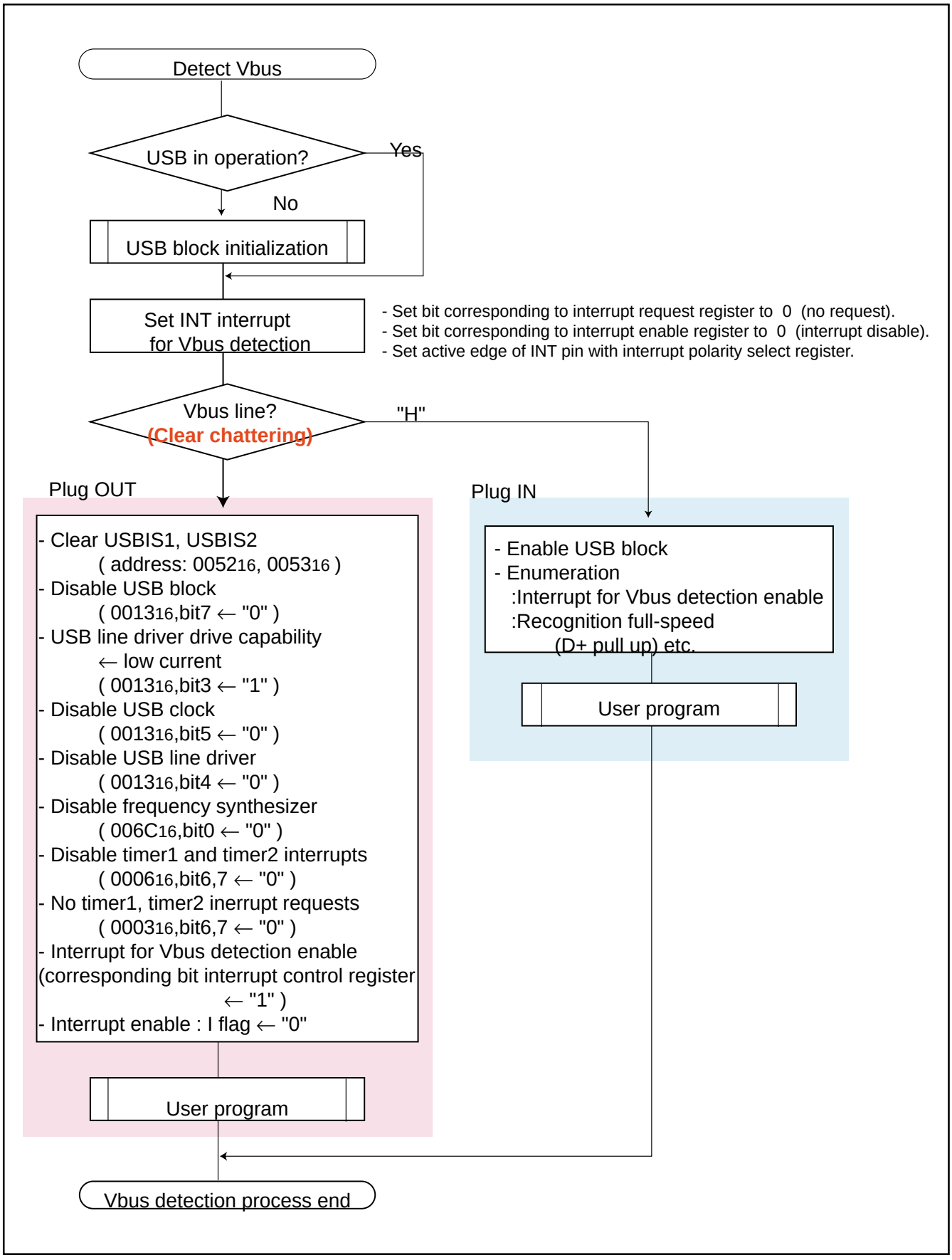


Fig.3.2.2 Example Vbus Detection Flowchart (using INT interrupt)

3.3 USB Additional Circuitry

3.3.1 Ext.Cap pin, D+/D- pin: Example of addtional circuitry

(1) Description of Ext.Cap pin and D+/D- pins

As M37641 is a full-speed USB, the D+ pin requires a 3.3V pull-up by an external circuit (1.5kΩ +/- 5%).

● **Ext.Cap pin**

This pin is used for the DC-DC converter capacitor and the D+ pull-up of the D+/D- line driver. The user must connect a capacitor.

- In Vcc =3.3 V operation, connect the Ext.Cap pin directly to the Vcc pin in order to supply power to the USB transceiver. In addition, you will need to disable the DC-DC converter in this operation (set bit 4 of the USB control register to “0 ”.)
- In Vcc =5 V operation, don't connect an external DC-DC converter to Ext.Cap pin. Please use the internal built-in DC-DC converter.

● D+ pin

This is the USB plus voltage line interface. Connect a 27 to 33 ohms (recommended value) resistor inline with the pin. (Note 1)

● D- pin

This is the USB minus voltage line interface. Connect a 27 to 33 ohms (recommended value) resistor inline with the pin. (Note 1)

Note 1: The resistor may require adjustments due to printed circuit board differences, such as impedance characteristics and layout. Please fully evaluate on your system and make adjustments accordingly.

(2) Precautionary notes for additional circuitry

Figure 3.3.1 shows an example of the circuits of the Ext.Cap. and D+/D- pins. The following are cautionary notes concerning these USB peripheral circuits.

- Connect a 2.2µF, low frequency tantalum capacitor (tolerance: within +/- 10%) and a 0.1µF ceramic capacitor (X7R type is recommended, tolerance: within +/- 10%) in parallel. Also, connect a 1.5kΩ resistor (+/- 5%) between the Ext.Cap and D+ pins. Connect additional capacitors to stabilize the D+/D- waveforms when using an external DC-DC converter (Vcc = 5V).
- USB Standard Specification Version 2.0 defines the driver impedance as 28 to 44 ohms (See 7.1.1.1 Full-speed (12Mb/s) Driver Characteristics). In order to satisfy this requirement, connect a capacitor (recommended value: 27 to 33 ohms) inline to the USB D+ pin and USB D- pin. In addition, connect a capacitor between the USB D+ pin/USB D- pins and the Vss pin, as necessary. This is to prevent ringing, as well as for adjusting the D+/D- rising or falling edge timing and cross over point. The value and structure of peripheral components may need to be adjusted to fit the characteristic impedance and layout of each printed circuit board. Always fully evaluate the MCU on your system and closely observe the waveforms. Confirm each element connection and adjust the resistor values and number of capacitors based on evaluation results.
- Make sure the USB D+/D- line wiring does not cross any other signals. Ensure plenty of GND width to protect the USB line. Use only USB connectors that satisfy the requirements of the USB specifications.
- Design-in the USB connector and USB cable with the shortest wiring possible.

3.3.2 Comparison of 5V/3.3V Operations

Settings for bits 3 and 4 [line driver current control bit (USBC3) and line driver supply enable bit (USBC4)] of the USB control register (address 0013_{16}) differ according to the operating voltage (5V/3.3V) and power supply method selected for M37641 operations.

(1) 5V operations

The USB function control unit operates on 3.3V. When M37641 is operating on 5V, the internal built-in DC-DC converter must be used to drop the voltage to 3.3V.

● **Setting for using the internal built-in DC-DC converter**

- During USB block initialization
  USBC4 = “1”: enable internal DC-DC converter
  USBC3 = “0”: set to high current
- During USB suspend interrupt routine
  USBC3 = “1”: set to low current
- During return interrupt from suspend state (USB resume interrupt or remote wake-up)
  USBC3 = “0”: set to high current

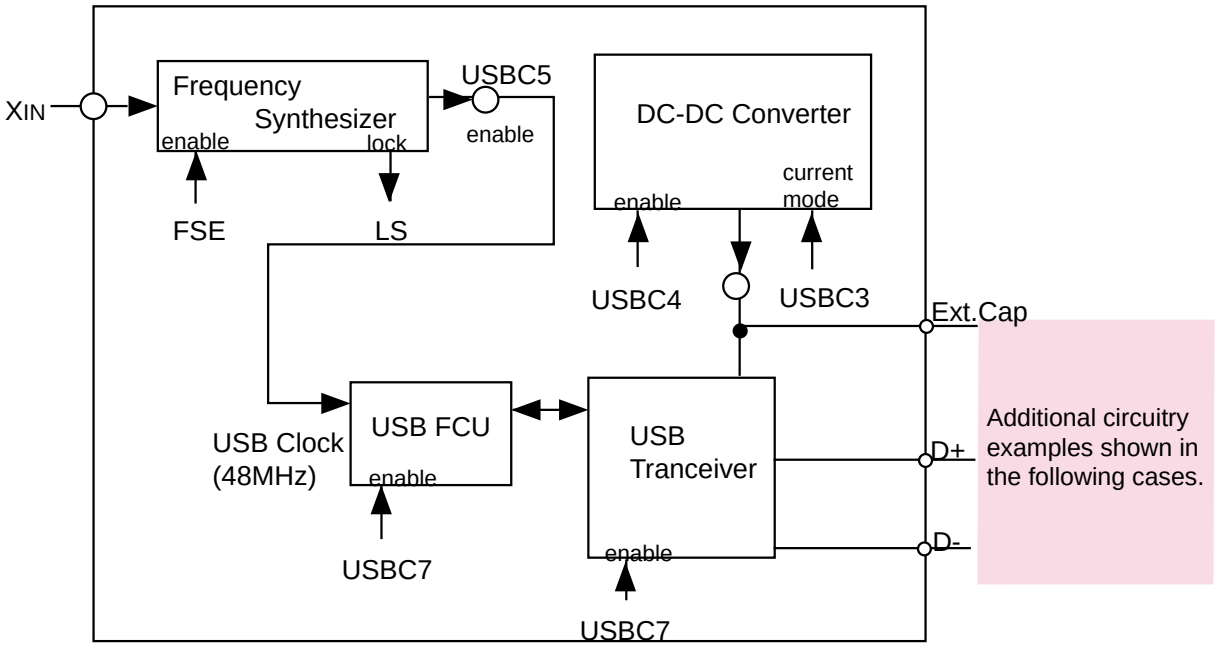
Note: In Vcc =5 V operation, don't connect an external DC-DC converter to Ext.Cap pin. Please use the internal built-in DC-DC converter.

(2) 3.3V operations

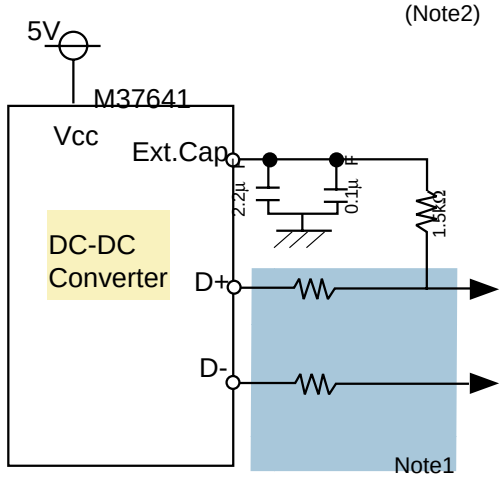
In 3.3V operations, the internal DC-DC converter is not needed and should be “disabled”.

-During USB block initializations
  USBC4 = “0”: disable internal DC-DC converter
  When USBC4 is “0”, the value of USBC3 loses any affect on USB operations.
-When supplying power through the Vbus (bus-power, self/bus switching system)
  Add an external DC-DC converter and drop the Vbus to 3.3V (see Section 3.1.2)

Note: If the D+ line is not pulled up within 100ms of connecting the USB cable, it may cause the host PC to generate a time-out. Refer to USB Specification Version 2.0, Section 7.1.7.1, “Power-0n and Connection Events Timing” for more details.

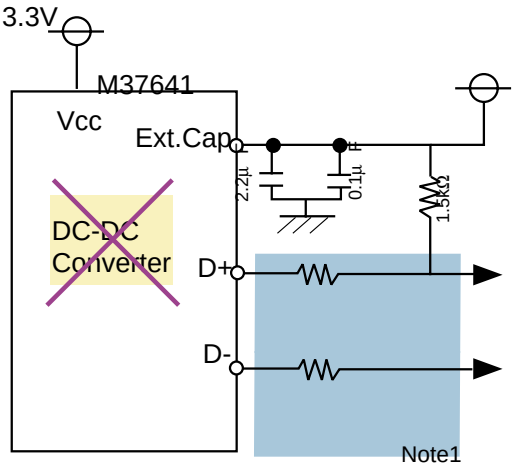


Case1: Vcc = 5V (using built-in DC-DC converter)



(Note2)

Case2: Vcc = 3.3V (Disable DC-DC converter)



Note 1: Resistor value varies according to printed circuit board layout.
Note 2: In Vcc =5 V operation, don' t connect an external DC-DC converter to Ext.Cap pin. Please use the internal built-in DC-DC converter.

Fig.3.3.1 Example of Ext.Cap Pin, D+/D- Pin with Additional Circuitry (Full speed operations)

3.3.3 Other Related Pins

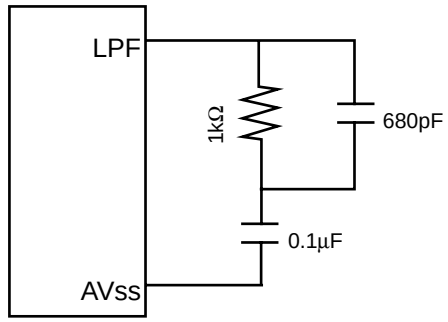
(1) LPF pin

When using the frequency synthesizer (to generate a 48MHz for the USB block), connect a low-pass filter to the LPF pin.

(2) Precautionary notes for the area surrounding the LPF pin

Using a high-tolerance capacitor may affect the USB 48MHz clock, and therefore interrupt USB transmissions.

Also, connect all passive LPF components as close as possible to the LPF pin. Figure 3.3.2 shows the recommended values.



Recommended values
- 680pF ceramic capacitor: no specified NPO precision
- 0.1µF ceramic capacitor: X7R within +/- 10% tolerance
- 1kΩ resistor: within +/- 10% tolerance

Figure 3.3.2 Recommended Values for Passive Components

Chapter 4
USB Transfers

This chapter provides detailed description of all 4 types of USB transfers.

4.1 Endpoint Control

Endpoints 0 through 4 are each equipped with independent IN (transmit) FIFO and OUT (receive) FIFO. USB transfer-related registers are listed below.

Table 4.1.1 USB Transfer Related Registers

Register Name (abbreviation)	Address (H)
USB control register (USBC)	0013
USB address register (USBA)	0050
USB interrupt status register 1 (USBIS1)	0052
USB interrupt status register 2 (USBIS2)	0053
USB interrupt enable register 1 (USBIE1)	0054
USB interrupt enable register 2 (USBIE2)	0055
USB endpoint index register (USBINDEX)	0058
USB Endpoint0 IN control register (IN_CSR)	0059
USB Endpoint x IN control register (IN_CSR)	0059
USB Endpoint x OUT control register (OUT_CSR)	005A
USB Endpoint x IN max. packet size register (IN_MAXP)	005B
USB Endpoint x OUT max. packet size register (OUT_MAXP)	005C
USB Endpoint x OUT write count register Low (WRT_CNTL)	005D
USB Endpoint x OUT write count register High (WRT_CNTH)	005E
USB endpoint FIFO mode register (USBFIFOMR)	005F
USB endpoint x FIFO register (USBFIFOx)	0060, 0061, 0062, 0063, 0064

Note: The size of Endpoints 1 and 2 can be modified by mode selection.

4.1.1 How to control endpoints

The special function registers common for each endpoint are as follows.

- USB Endpointx IN control register
- USB Endpointx OUT control register
- USB Endpointx IN max. packet size register
- USB Endpointx OUT max. packet size register
- USB Endpointx OUT write count register
- USB endpoint FIFO mode register (endpoint 1,2 only)

Specify the endpoints that will be accessed by using the USB endpoint index register, and then execute a read/write for each of the above registers.

4.1.2 FIFO buffer

The number of packets to be stored in each IN/OUT FIFO of Endpoints 1 to 4 is determined by the size of the FIFO set in the USB Endpointx IN max. packet size register and the USB endpointx OUT max. packet size register.

- Single buffer
  If the set value of each endpoint is larger than 1/2 the maximum FIFO size, the FIFO becomes a single buffer and can store 1 packet data.
- Double buffer
  If the set value of the endpoint is less than 1/2 the maximum FIFO size, the FIFO becomes a double buffer and can store up to 2 packets data.

4.2 Endpoint0 (control transfers)

Control transfers are executed through Endpoint0. Data is transmitted and received according to the device request sent from the host PC.
A USB Endpoint0 interrupt is generated when data is received from the host CPU in an Endpoint0 control transfer. However, determining whether the data was received in the SETUP stage or the DATA stage cannot be done by hardware. The user needs to analyze the received data by software, and process it within the interrupt routine. Refer to USB Specification Version 2.0 concerning the data configuration of the standard device request.
When data is received from the host PC, it is written to Endpoint0 OUT FIFO. After one packet is received successfully, confirm the number of received data by reading out USB Endpoint0 OUT write count register Low, then High. Then read out the same number of bytes of data from Endpoint0 OUT FIFO.
When transmitting data to the host, write data to Endpoint0 IN FIFO.
The Endpoint0 packet size is set in the USB Endpoint0 IN max. packet size register. Make sure you set the value to equal to or less than the maximum Endpoint0 FIFO size (16 bytes).

4.2.1 Data Received by control transfer

When Endpoint0 receives a valid packet from the host, the OUT_PKT_RDY flag is set to “1”, the lower 8 bits of the number of received bytes are set to USB Endpoint0 OUT write count register Low, and the upper two bits of the number of received bytes are set to USB Endpoint0 OUT write count register High. The same number of received data is read out from the Endpoint0 OUT FIFO and determine by software whether it was received in the DATA stage or in the STATUS stage.
After the SETUP token is received, M37641 is in the setup phase until the OUT_PKT_RDY flag is cleared. If the OUT_PKT_RDY flag has not been cleared (indicating the request from the host PC has not been decoded), a NAK is returned to all IN/OUT tokens.
When the number of data set in the SETUP stage has been completely received, set the DATA_END bit to “1”. While this bit is “1”, any requests from the host for ‘data receive’ or ‘data transmit’ are ignored, and a STALL is returned. This bit is automatically cleared when the STATUS stage is completed.

● Receiving an unsupported request
If an unsupported request is received, set the SEND_STALL bit to “1”. USB FCU returns a STALL signal to the host PC when OUT_PKT_RDY flag for receiving the request becomes “0”. However, writing “0” to the SEND_STALL bit when a CLEAR_FEATURE (Endpoint STALL) request has not been received will inhibit the next STALL from being generated.
You do the following setup at the same time.
- Set SEND_STALL bit to “1”.
- Set DATA_END bit to “1”.
- Make the OUT_PKT_RDY flag go to “0” by setting the SERVICED_OUT_PKT_RDY bit to “1”.

● When an error occurs during a receive from the host
When an error is received, the FORCE_STALL flag will be set to “1”. While this flag is “1”, a STALL is sent in response to any IN/OUT tokens in the DATA stage or STATUS stage. Clear the flag by writing “0”.

● Receiving data in DATA Stage (when receive data still remains)
Read out the receive data from the FIFO, decode the request from the host, then clear the OUT_PKT_RDY flag.

● Receiving data in Status Stage (after all data has been received)
Clear the OUT_PKT_RDY flag, then set the DATA_END bit to “1”.

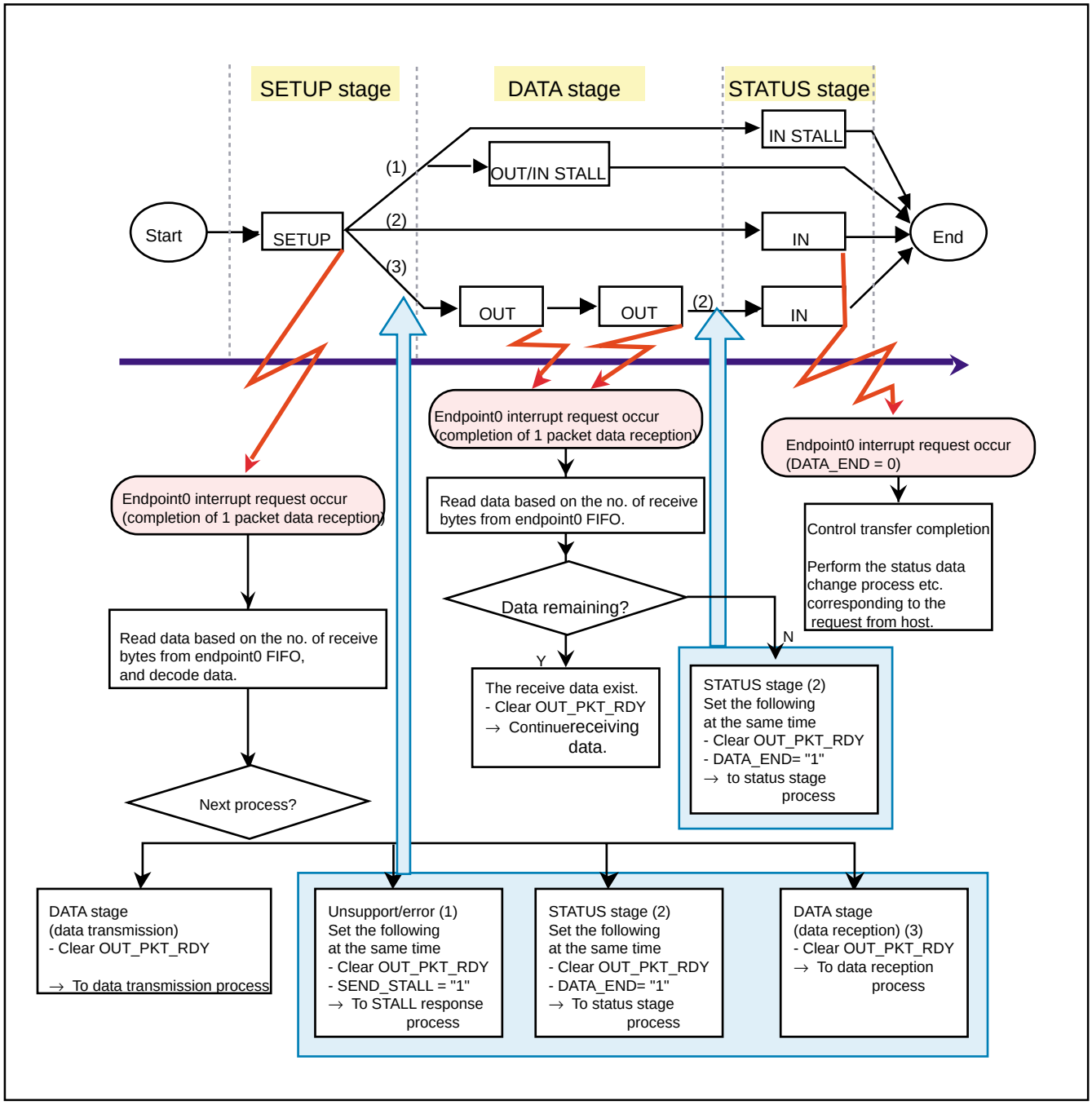


Figure 4.2.1 Endpoint0: Data Receive Example

4.2.2 Data transmitted by control transfer

When transmitting data to the host, write data less than or equal to one packet size to the Endpoint0 IN FIFO and set the IN_PKT_RDY bit to “1”. If the IN_PKT_RDY bit is set to “1” without writing data to the IN FIFO, a NULL packet of “0” length will be sent. You set the size of one packet in the USB Endpoint0 IN max. packet size register.

The IN_PKT_RDY bit is automatically cleared either when one packet data is sent to the host or when the SETUP_END bit goes to “1” during the data phase processing.

In a control transfer, set the DATA_END bit to “1” after writing the last data packet to the IN FIFO in the IN DATA stage. By setting this bit to “1”, the process will proceed to the status stage after the data is transmitted. This bit is cleared when the STATUS stage is completed.

During a control transfer, if the process is terminated before the number of data bytes, as set in the DATA stage process, is completed, the SETUP_END flag will go to “1”. In this case, you need to disable the access to the FIFO and execute the setup process before it. Clear the SETUP_END flag by setting the SERVICED_SETUP_END bit to “1”.

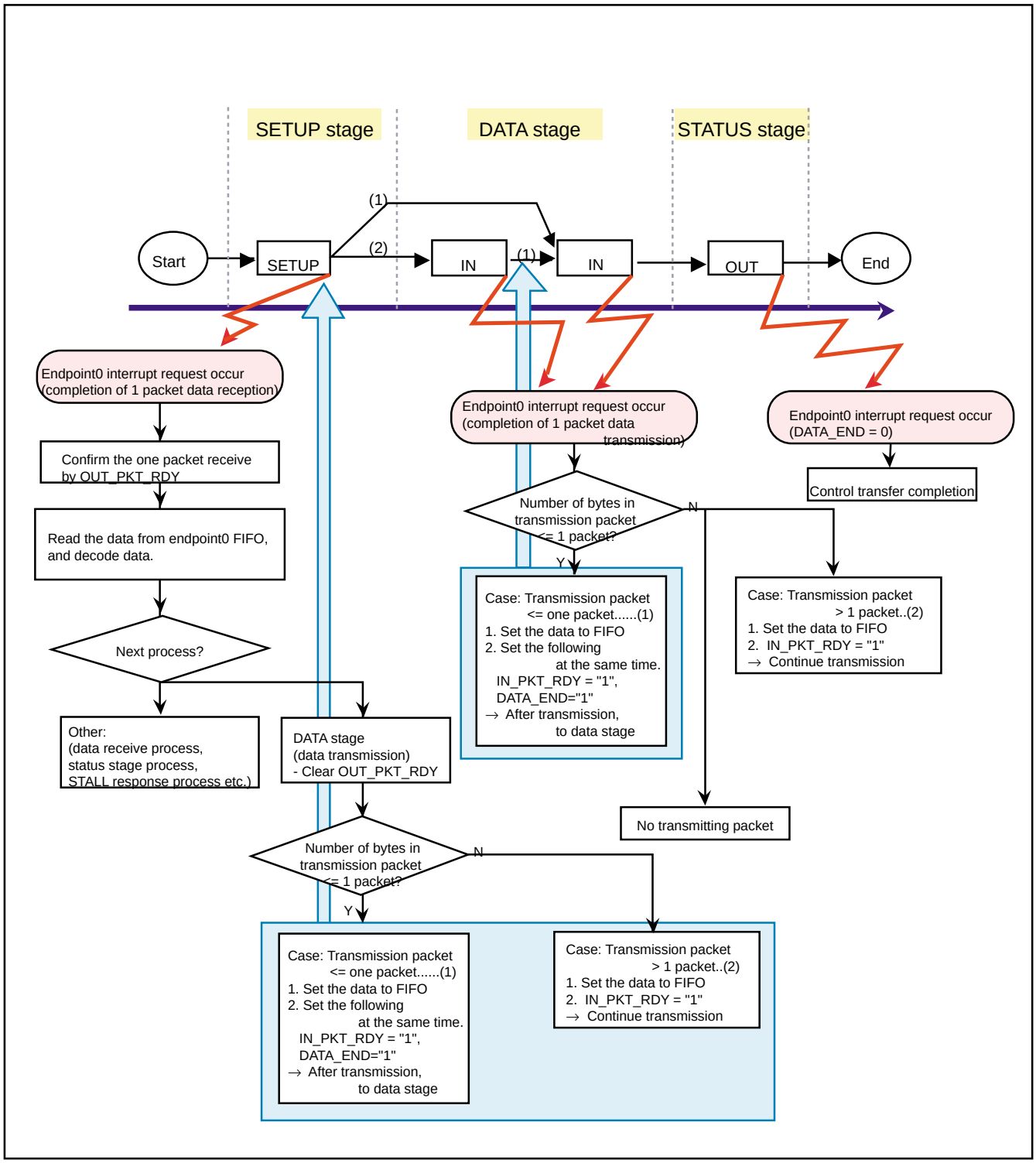


Fig. 4.2.2 Endpoint0: Data Transmission Example

4.3 Endpoints 1 to 4 Receive (OUT)

Each Endpointx (x = 0 to 4) has an independent OUT (receive) FIFO. All endpoints that can be accessed must be set up in the USB endpoint index register. The size of each OUT FIFO is listed below.

Endpoint0:		16 bytes
Endpoint1:	Mode 0:	800 bytes
	Mode 1:	1024 bytes
	Mode 2:	2048 bytes
	Mode 3:	0 bytes
	Mode 4:	1280 bytes
	Mode 5:	1168 bytes
Endpoint2:	Mode 0:	32 bytes
	Mode 1:	128 bytes
Endpoint3:		16 bytes
Endpoint4:		16 bytes

The IN and OUT FIFO sizes of Endpoint1 and Endpoint2 are variable. Endpoint1 offers six selectable modes and Endpoint2 offers two modes. Each mode can be selected in the USB endpoint FIFO mode select register (address $005F_{16}$).
When data is received from the host, the data is written to an Endpointx OUT FIFO (addresses 0061_{16} to 0064_{16}). After one packet has been successfully received, read out the data based on the number of bytes received as indicated in the USB Endpointx OUT write count register (Low, High) from each OUT FIFO. The max. OUT packet size for each endpoint is set in the USB Endpointx OUT max. packet size register.
After specifying the endpoints in the USB endpoint index register, set a value lower than the OUT FIFO size of each endpoint.

4.3.1 Data receive outline (For bulk/isochronous/interrupt transfer)

When Endpointx receives a valid packet from the host, the OUT_PKT_RDY flag is set to “1”, the lower 8 bits of the no. of received data are set in USB Endpointx OUT write count register Low, and the upper 2 bits are set in USB Endpointx out write count register High. At this point, read the exact number of received data from the OUT FIFO, and then clear the OUT_PKT_RDY flag to “0”. The method for clearing the OUT_PKT_RDY flag will differ according to value of the AUTO_CLR bit, as follows.

(1) Single buffer

- When AUTO_CLR bit is “1”
  When data (the size equals received OUT packet) is read from the OUT FIFO, the OUT_PKT_RDY flag is automatically cleared to “0”.
- When AUTO_CLR bit is “0”
  The user needs to clear the OUT_PKT_RDY flag through software.

(2) Double buffer

- When AUTO_CLR is “1”

  When data (the size equals received OUT packet) is read from the OUT FIFO but there is no data in the OUT FIFO, the OUT_PKT_RDY flag automatically goes to “0”. If there is one packet of data remaining, OUT_PKT_RDY will be cleared to “0” but then set to “1” after one ø cycle (when f(XIN) = 12MHz, 83ns).

- When AUTO_CLR is “0”

  Clear the OUT_PKT_RDY flag to “0” after reading data from the OUT FIFO (it will not be cleared automatically). When there is no data in the OUT FIFO, the OUT_PKT_RDY flag will go to “0”. If there is one packet of data in the OUT FIFO, OUT_PKT_RDY is cleared to “0” once, then set to “1” after one ø cycle (when f(XIN) = 12MHz, 83ns).

  For double buffer transfers, the number of bytes of data in the previously received packet is stored in the USB Endpointx OUT write count register (Low, High). Each time one packet data is read from the OUT FIFO and the OUT_PKT_RDY flag is set to “0”, the value in the USB Endpointx OUT write count register (Low, High) is updated.

4.3.2 Receive error and FIFO Flush (For bulk/isochronous/interrupt transfer)

(1) Receive error

The FORCE_STALL flag goes to “1” when Endpointx receives a data from the host which is larger than the packet size set in the USB Endpointx OUT max. packet size register. While this bit is “1”, a STALL handshake is sent to the host. Restart the transfer by clearing this bit to “0”.

(2) FIFO Flush

Writing a “1” to the FLUSH bit will erase the data in USB Endpointx OUT FIFO. If there is one packet data in the OUT FIFO, the OUT FIFO will become empty. If double buffer is used and there are 2 packets in the OUT FIFO, the oldest data will be erased.

Avoid setting the FLUSH bit while receiving data, as the data packet being received at that point may be lost. Before setting the FLUSH bit, always confirm that there is data in the FIFO (OUT_PKT_RDY flag= “1”).

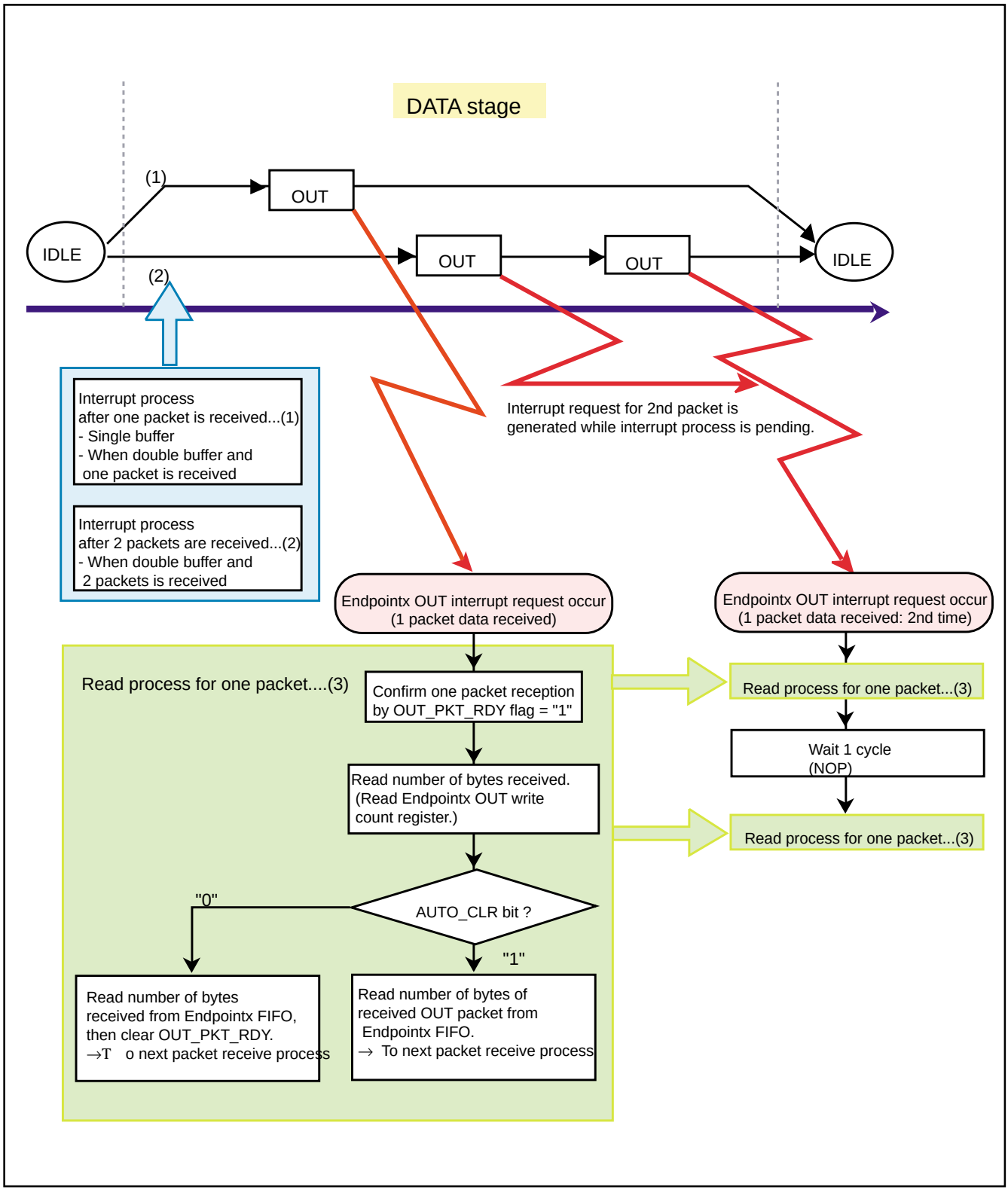


Figure 4.3.1 Endpointx: Data Receive Example

4.3.3 Data received in bulk transfer

(1) Setting the transfer type

When using Endpointx OUT for bulk transfers, the endpoint needs to be properly set up in the USB Endpointx OUT control register. Set the ISO/TOGGLE_INIT bit to “0” to enable bulk transfers.
In addition, this is a double function bit and is used for initialization of the toggle sequence.
When necessary, to re-initialize the toggle sequence to DATA0 for the bulk transfer endpoint, please execute the following procedure:

<Initialization of Toggle Sequence>

1. Set ISO/TOGGLE_INIT bit to “1”.
2. Set ISO/TOGGLE_INIT bit to “0”.
3. Fix ISO/TOGGLE_INIT bit to “0” for the remainder of the bulk transfers.

(2) Data ‘receive’ operations

The bulk OUT transfer, which transmits data from the host CPU to the device, continually repeats OUT transactions.
When Endpointx OUT receives a valid packet from the host, it transmits an ACK response to the host and generates an Endpointx OUT interrupt request. DATA0 and DATA1 of data packet of next data phase are toggled during the handshake phase of each transaction if the ACK packet sent by the receive side is received successfully by the transmit side.
M37641 transmits the following responses when data is not successfully received.

- No response if the received data is broken.
- STALL handshake is returned if M37641 is stalled. (It occurs if the packet size is greater than max packet size or SEND_STALL bit is “1”.)
- ACK packet is returned if inconsistency of sequence bit in received data.
- NAK packet is returned if the M37641 OUT FIFO is full.

During the Endpointx OUT interrupt process, after confirming that the Endpointx OUT_PKT_RDY flag is “1”, extract the number of data bytes in the received packet and read out the data from the OUT FIFO.
After reading the data from the FIFO, clear the OUT_PKT_RDY flag. (If AUTO_CLR bit is “1”, the flag will automatically go to “0”. If AUTO_CLR bit is “0”, set the flag to “0” with software.) By clearing the flag, the FIFO is now ready to receive the next packet of data.
When the host transmits 2 packets having the same data toggle, M37641 transmits an ACK response back for each packet, but the data in the 2nd packet will not be stored in the FIFO. In this case, since ACK transmitted to the host in response to 1st packet was lost, M37641 assumes that 2 packets of the same data were sent.
M37641 performs an error check in accordance with USB Specification Version 2.0 (CRC check, stuffed bit check, etc.). If an error is detected during a bulk OUT transfer, neither ACK nor NAK are returned. All error checks are performed through hardware, so the user does not need to handle errors through software.

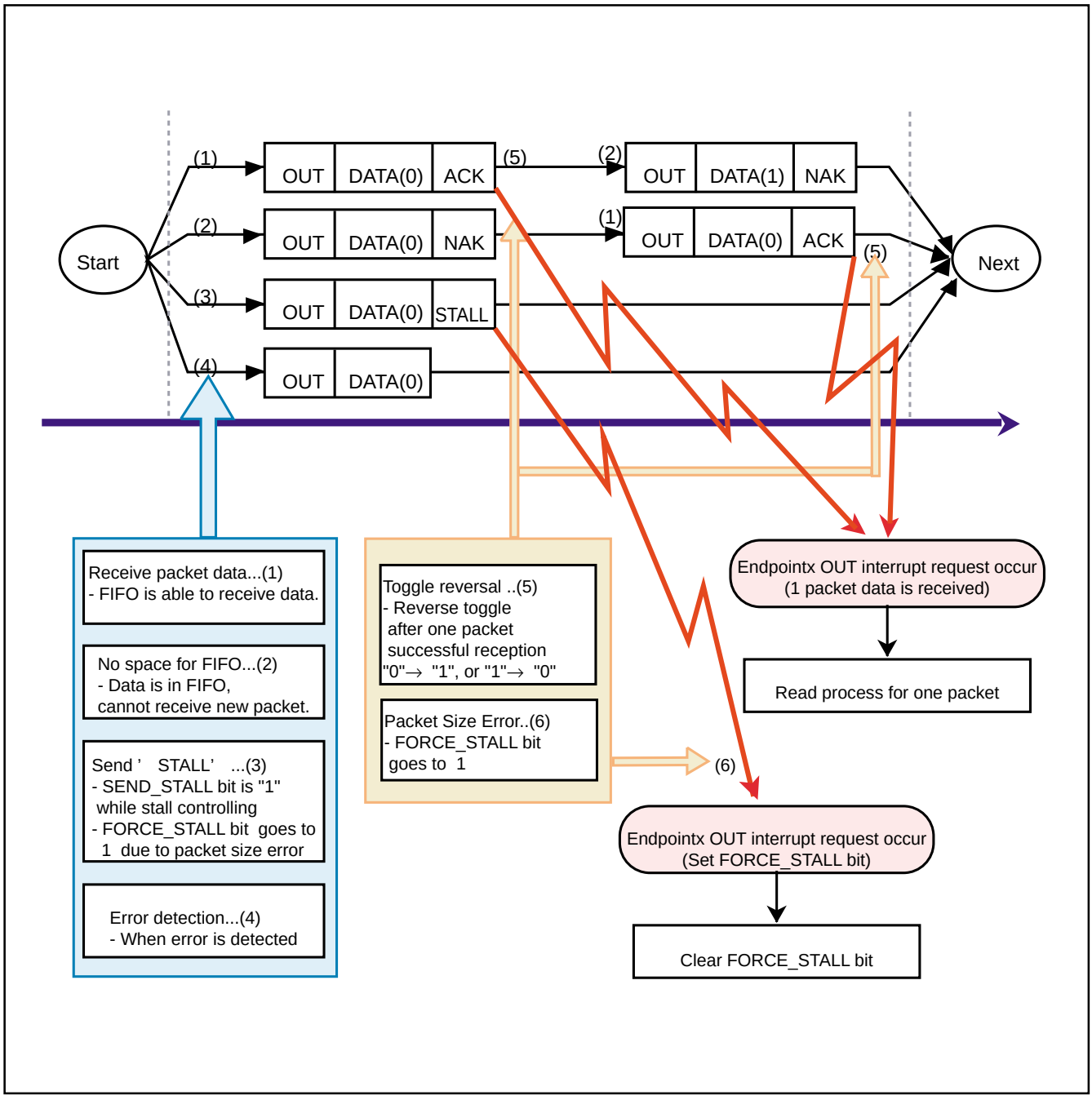


Figure 4.3.2 Endpointx: Bulk Receive Example

4.3.4 Data received in isochronous transfer

Repeat isochronous transaction (OUT) which transfers data from the host CPU to the device, or, isochronous transaction (IN) which transfers from the device to the host CPU. Isochronous transaction do not include a handshake phase. The only data packet available is DATA0, which is not toggled with DATA1.

(1) Setting the transfer type

When using Endpointx OUT for isochronous transfers, set in the USB Endpointx OUT control register. Set the ISO/TOGGLE_INIT bit to “1” to enable isochronous transfers.

When this bit is set to “1” and data is sent from the host, M37641 will receive the packet with the DATA0 PID. The ISO/TOGGLE_INIT bit should be fixed at “1” during the isochronous transfer.

(2) Data receive operation

In isochronous transfers, the OUT endpoint transmits neither ACK nor NAK back to the host. When Endpointx OUT receives a valid packet from the host, an Endpointx OUT interrupt request is generated. In an endpointx OUT interrupt process, first confirm that the Endpointx OUT_PKT_RDY flag is “1”, then determine the number of bytes of data received and read out the data from the OUT FIFO.

M37641 is equipped with an error check function (hardware), which complies with USB Specification Version 2.0. If an error is detected in a received packet during an isochronous transfer, the DATA_ERR flag will be set to “1”. This data, including the error, will be stored in the FIFO. The user should clear the DATA_ERR flag and erase the data from the FIFO.

If you are using the FLUSH bit to erase the data in the FIFO, always make sure that the OUT_PKT_RDY flag is “1” before erasing.

(3) Overrun error

An overrun error is caused when a data packet cannot be received in the OUT FIFO due to remaining data in the FIFO, and therefore, an OUT data packet from the host will not be received. When an overrun error occurs, the OVER_RUN bit is set to “1” and an Endpointx OUT interrupt request is generated. If the USB overrun/underrun interrupt is enabled at this time, this interrupt request will be generated as well.

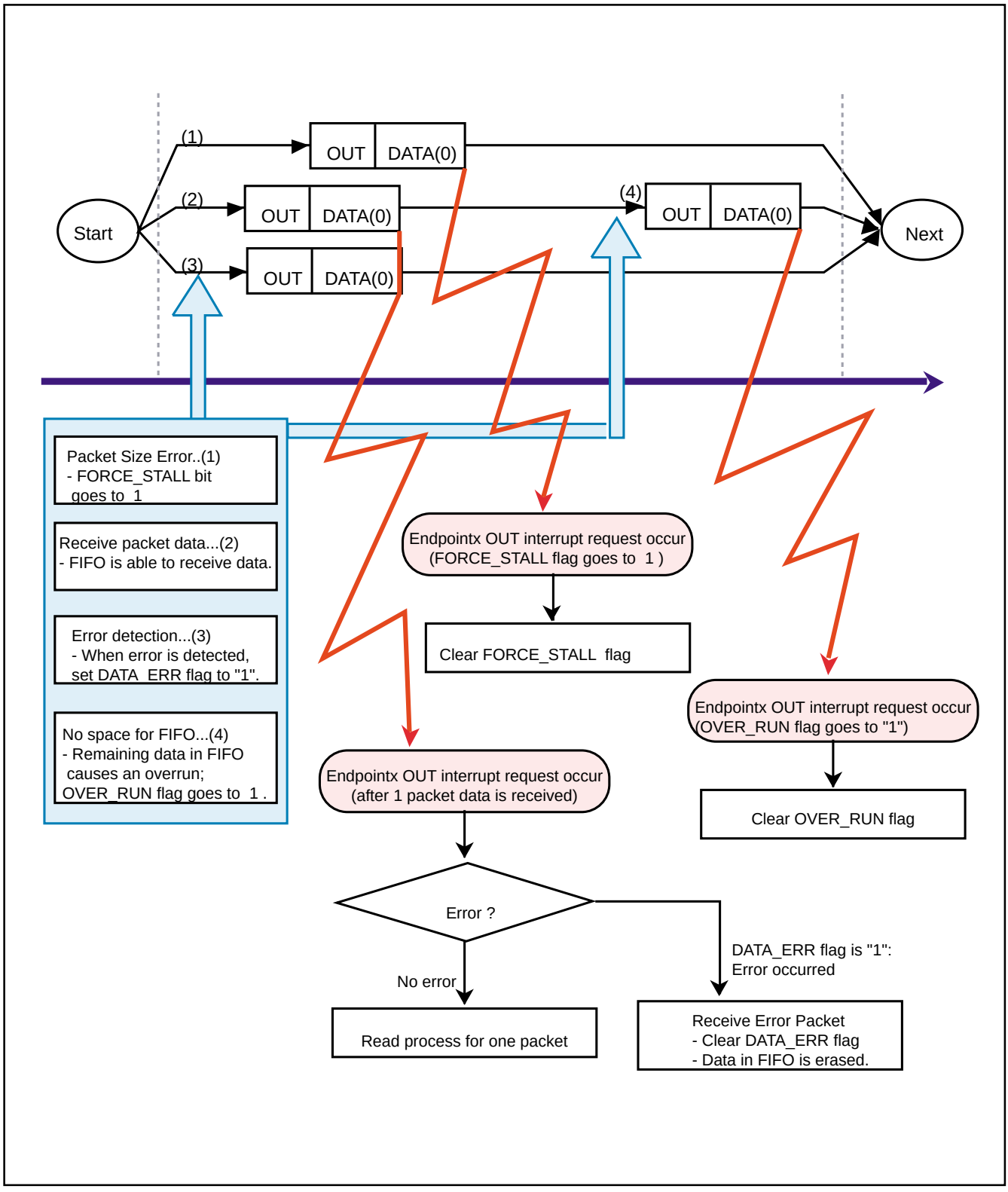


Figure 4.3.3 Endpointx: Isochronous Data Receive Example

4.3.5 Data received in interrupt transfer

(1) Setting the transfer type

When using Endpointx OUT for interrupt transfers, set in the USB Endpointx OUT Control Register. Set the ISO/TOGGLE_INIT bit to “0” to enable interrupt transfers.

In addition, this bit is a double function bit and is used for initialization of the toggle sequence. When necessary, to re-initialize the toggle sequence for an interrupt transfer endpoint, please perform the following procedure:

<Initialization of Toggle Sequence>

1. Set ISO/TOGGLE_INIT bit to “1”.
2. Set ISO/TOGGLE_INIT bit to “0”.
3. Fix ISO/TOGGLE_INIT bit to “0” for the remainder of the interrupt transfers.

(2) Data receive operation

Endpointx OUT operations during interrupt transfers are the same as that of bulk transfers. Please refer to Section 4.3.3 “Data receive in bulk transfer” for details.

4.4 Endpoints 1 to 4 Transmit (IN)

Each Endpointx (x = 0 to 4) has an independent IN (transmit) FIFO. All endpoints that can be accessed must be set up in the USB endpoint index register. The size of each IN FIFO is listed below.

Endpoint0:		16 bytes
Endpoint1:	Mode 0:	512 bytes
	Mode 1:	1024 bytes
	Mode 2:	0 bytes
	Mode 3:	2048 bytes
	Mode 4:	768 bytes
	Mode 5:	880 bytes
Endpoint2:	Mode 0:	32 bytes
	Mode 1:	128 bytes
Endpoint3:		16 bytes
Endpoint4:		16 bytes

The IN and OUT FIFO sizes of Endpoint1 and Endpoint2 are variable. Endpoint1 offers six selectable modes and Endpoint2 offers two modes. Each mode can be selected in the USB endpoint FIFO mode select register.

When transmitting data to the host, write the data to the Endpointx IN FIFO. The internal counter increments by one each time one byte is written in the IN FIFO (the internal counter cannot be read). Set up a transmit packet for the host by writing one packet of data to the IN FIFO, then setting the IN_PKT_RDY bit to “1”. The packet will be sent in response to the next IN token from the host.
In addition, the max IN packet size for each endpoint is set in the USB Endpointx IN max. packet size register. After specifying the endpoints in the USB endpoint index register, set a value lower than the IN FIFO size for each endpoint.

4.4.1 General description of data 'transmit' (For bulk/isochronous/interrupt transfer)

When Endpointx transmits a packet to the host, a packet of data must be prepared in the IN FIFO in anticipation of the IN token coming from the host.
In order to prepare one packet of data, always make sure the Endpointx IN FIFO is empty, before writing data to the IN FIFO equal to the number of bytes of packet data that will be sent to the host. Finally, set the IN_PKT_RDY bit to "1". If an empty packet will be sent, do not write data to the Endpointx IN FIFO before setting the IN_PKT_RDY bit. The IN_PKT_RDY bit setting will differ according to the value of the AUTO_SET bit.
If a packet data is ready, the data can then be transmitted in response to the next IN token received from the host. After the data is transferred to the host, an Endpointx IN interrupt request is generated, and the IN FIFO is ready for the next packet.

(1) Single buffer

- When AUTO_SET bit = "1"
  The IN_PKT_RDY bit is automatically set to "1" when the number of bytes of data written to the IN FIFO equals the value set in the Endpointx IN max. packet size register.
- When AUTO_SET bit = "0" or when a short packet (smaller than max. packet size) is transferred
  Set the IN_PKT_RDY bit to "1" with software after writing data to the FIFO. (The IN_PKT_RDY bit is not automatically set to "1")

In single buffer operations, regardless of the value of the AUTO_SET bit, the data packet will be sent to the host and the IN_PKT_RDY bit and the TX_NOT_EPT flag will be automatically cleared after the host returns an ACK.

(2) Double buffer

- When AUTO_SET is "1"
  When the number of bytes of data written to the IN FIFO equals the value set in the Endpointx IN max. packet size register and there is only 1 packet in the IN FIFO, the IN_PKT_RDY bit is automatically set to "1". However, it is cleared after one ø cycle (when f(X_{IN}) = 12MHz, 83ns) wait, and the TX_NOT_EPT flag then goes to "1" (so that one more data packet can be stored in the IN FIFO). If there are already 2 packets in the FIFO (FIFO full), both the IN_PKT_RDY bit and the TX_NOT_EPT flag go to "1".
- When AUTO_SET is "0" or when a short packet (smaller than max. packet size) is transferred
  Set the IN_PKT_RDY bit to "1" with software after one packet has been written to the IN FIFO (the IN_PKT_RDY bit is not automatically set to "1"). When one packet data is set in the IN FIFO and there is only one packet stored in the IN FIFO, the IN_PKT_RDY bit goes to "1". However, it is cleared after one ø cycle (when f(X_{IN}) = 12MHz, 83ns) wait, and the TX_NOT_EPT flag then goes to "1" (so that one more data packet can be stored in the IN FIFO). If there are already 2 packets in the IN FIFO (IN FIFO full), both the IN_PKT_RDY bit and the TX_NOT_EPT flag go to "1".

Regardless of the value of the AUTO_SET bit, all data packets will be sent to the host, and the IN_PKT_RDY bit and the TX_NOT_EPT flag will be automatically cleared after M37641 received an ACK from the host. If there is one packet data already in the IN FIFO, the IN_PKT_RDY bit will go to "0" after completion of the transmission, but the TX_NOT_EPT flag will remain at "1".

4.4.2 Transmit error and FIFO Flush (For bulk/isochronous/interrupt transfer)

(1) Transmit error

If the system goes to the STALL state (operations disabled), set the SEND_STALL flag to “1”. M37641 transmits a STALL handshake in response to IN tokens from the host.

(2) FIFO Flush

Erase the contents of the USB Endpointx IN FIFO by setting the FLUSH bit to “1”. If there is one packet data in the IN FIFO, the IN FIFO will then become empty. If there are two packets in the FIFO, the older packet will be erased. At this time the bits that indicate the state of the IN FIFO will be updated (IN_PKT_RDY and TX_NOT_EPT).

The IN_PKT_RDY and TX_NOT_EPT bits will indicate IN FIFO buffer empty information, see Table 4.4.1 for details. Do not set the FLUSH bit during a data transfer as this may damage the data packet in transit.

M37641 performs an error check in accordance with USB Specification Version 2.0 (CRC check, stuffed bit check, etc.). All error checks are performed through hardware during Endpointx IN transfers, so the user does not need to handle errors through software.

Table 4.4.1 IN FIFO States

IN_PKT_RDY	TX_NOT_EMP	IN FIFO States
0	0	No data packet
0	1	1 data packet (double buffer) Invalid (single buffer)
1	0	Invalid
1	1	1 data packet (single buffer) 2 data packets (double buffer)

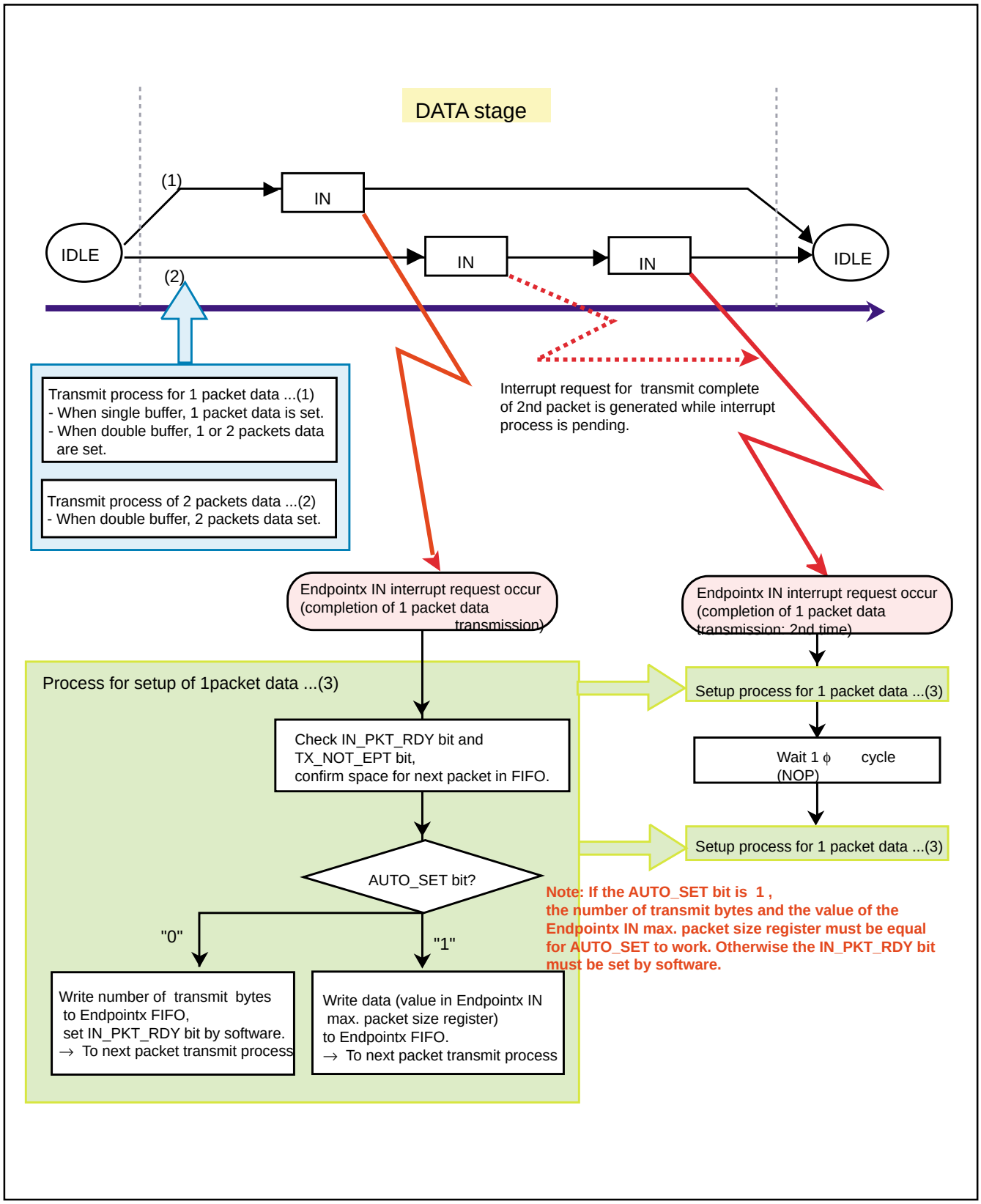


Figure 4.4.1 Endpointx: Data Transmit Example

4.4.3 Data transmit in bulk transfer

(1) Setting the transfer type

When using Endpointx IN for bulk transfers, set up in the USB Endpointx IN control register. To enable bulk transfers, set the ISO/TOGGLE_INIT bit to “0” and the INTPT bit to “0”. The ISO/TOGGLE_INIT bit is a double function bit and is used for initialization of the toggle sequence bit (resets the next data packet to DATA0).

Where necessary, to re-initialize the toggle sequence for the bulk transfer endpoint, please execute the following procedure:

<Initialization of Toggle Sequence>

1. Set ISO/TOGGLE_INIT bit to “1”.
2. Set ISO/TOGGLE_INIT bit to “0”.
3. Fix ISO/TOGGLE_INIT bit to “0” for the remainder of the bulk transfers.

(2) Data transmit operation

A NAK is automatically returned if an IN token is received from the host when there is no data set in Endpointx IN FIFO. In order to transmit data, the packet data must be set in the IN FIFO. First, confirm that there is empty space in Endpointx IN FIFO (by examining the states of IN_PKT_RDY and TX_NOT_EPT bits), then set the packet data to be transmitted. Once the data is set, it will be sent to the host when Endpointx IN receives an IN token from the host. When an ACK is received from the host in response to the data packet, the 1 packet transmission is complete and an Endpointx IN interrupt request is generated. Reconfirm that there is space in the Endpointx IN FIFO, then set the next packet data to be transmitted.

In bulk transfers, DATA0 and DATA1 are toggled after the IN Endpoint receives an ACK response from the host (DATA0 $\rightarrow$DATA 1, DATA1 $\rightarrow$DATA0). If an ACK is not received from the host, the same data in the same toggle is sent at the next IN token.

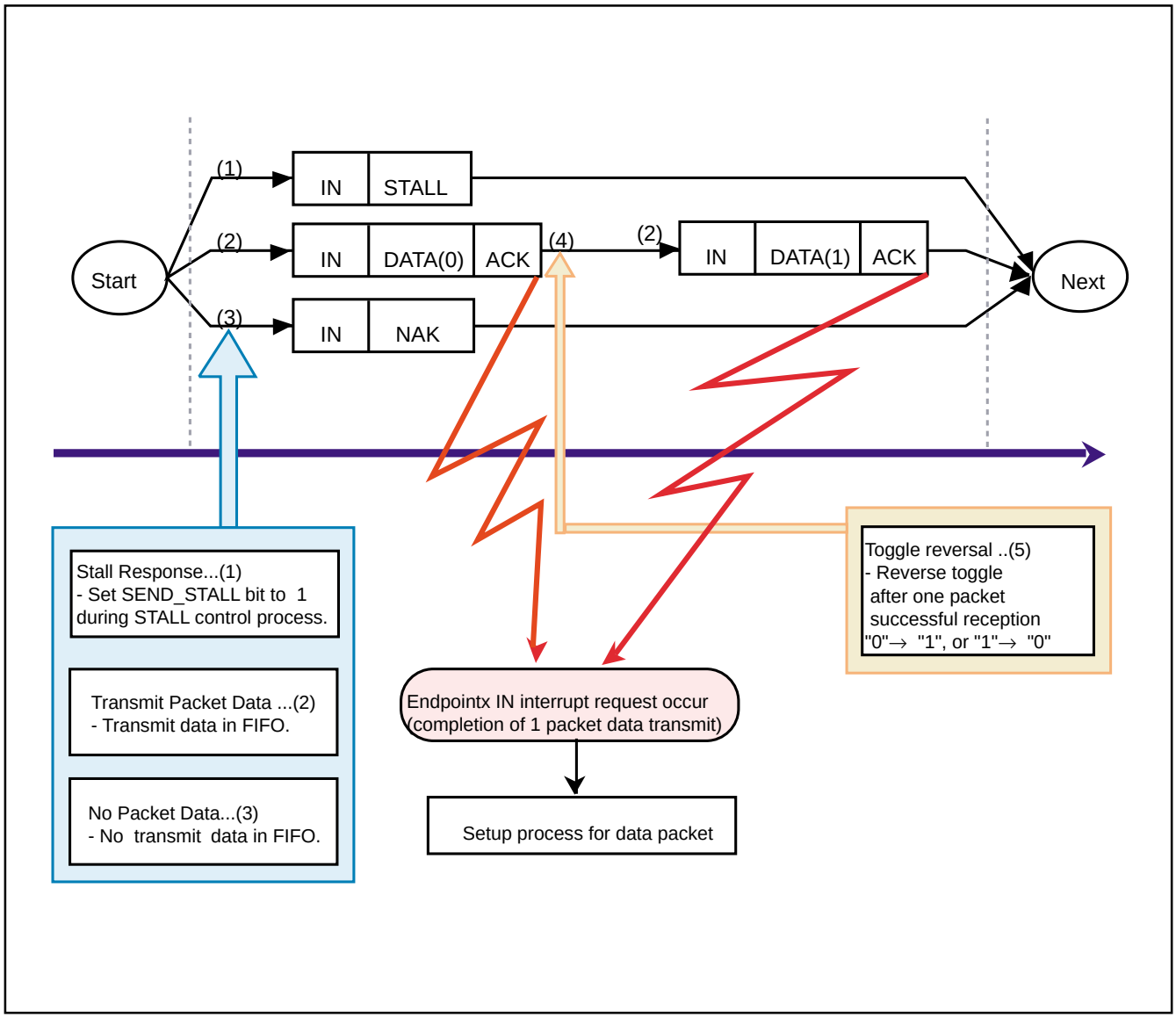


Figure 4.4.2 Endpointx: Bulk Data Transmit Example

4.4.4 Data transmit in isochronous transfer

Repeat isochronous transaction (OUT), that transfers data from the host CPU to the device or isochronous transaction (IN), that transfers from the device to the host CPU. Isochronous transaction do not include a handshake phase. The only data packet available is DATA0, which is not toggled with DATA1.

(1) Setting the transfer type

When using Endpointx IN for isochronous transfers, set in the USB Endpointx IN control register. To enable isochronous transfers, set the ISO/TOGGLE_INIT bit to “1” and INTPT bit to”0”. The ISO/ TOGGLE_INIT bit should be fixed to “1” during the isochronous transfer.

(2) Data transmit operation

In order to transmit data, a packet data must be set in the IN FIFO. First confirm that the endpointx IN FIFO has an empty space (see Table 4.4.1), and then set the packet data to be sent. Once the data is set, it will be transmitted to the host when Endpointx IN receives an IN token from the host. After one packet has been successfully transmitted, an Endpointx IN interrupt request is generated.

The timing of the data transmit from the device, in response to the IN token from the host, is determined according to the ISO_UPDATE bit in the USB Endpoint Index Register. This bit is shared by IN Endpoints 1 to 4, and is valid for all isochronous IN transfers. Even when setting the index for other endpoints that don’t use isochronous transfers, always maintain the setting of this bit. Please note that, changing the setting of this bit will cause the settings of Endpoints 1 to 4 to be changed as well.

- When ISO_UPDATE bit is “0”

  Endpointx IN transmits the IN FIFO data to the host when it receives an IN token from the host.
- When ISO_UPDATE bit is “1”

  Endpointx IN does not transmit the IN FIFO data to the host even when it receives an IN token from the host. After receiving the next SOF, Endpointx IN transmits the IN FIFO data in response to the IN token from the host. By delaying the data transfer to the host until the next SOF is received, the transmit data is synchronized with the SOF.

After the transfer is complete, confirm that there is space in the Endpointx IN FIFO, and set the next packet data to be transmitted in endpointx IN interrupt process.

(3) AUTO_FLUSH function

The AUTO_FLUSH function automatically erases the oldest packet of data in the IN FIFO when the isochronous endpoint (the ISO_UPDATE bit of USBINDEX is “1” and the corresponding AUTO_FLUSH bit is “1”) receives an SOF while the IN_PKT_RDY bit is set to “1”. The AUTO_FLUSH function works only when the isochronous IN Endpoint is functioning as a double buffer and there are 2 packets in the IN FIFO.

In single buffer operations, if the IN_PKT_RDY bit is “1”, the AUTO_FLUSH function is worked, but the data in the IN FIFO will not be transmitted out. In addition, the AUTO_FLUSH function will not worked when there is only one packet in the IN FIFO in double buffer (TX_NOT_EPT flag is "1" and IN_PKT_RDY bit is "0").

As the AUTO_FLUSH bit is shared by all endpoints (Endpointx, x = 1 to 4)) in isochronous IN transfers, it is valid for all isochronous IN transfers. Even when setting the index for other endpoints that don’t use isochronous transfers, always maintain the setting of this bit. Please note that, changing the setting of this bit will cause the settings of Endpoints 1 to 4 to be changed as well.

(4) Underrun error

If an IN token is received from the host when there is no data in the IN FIFO, an empty data packet is automatically returned and an underrun error is generated. The UNDER_RUN bit is set to “1”, and an Endpointx IN interrupt request is generated. In addition, if the USB overrun/underrun interrupt is enabled, a USB overrun/underrun interrupt request is also generated.

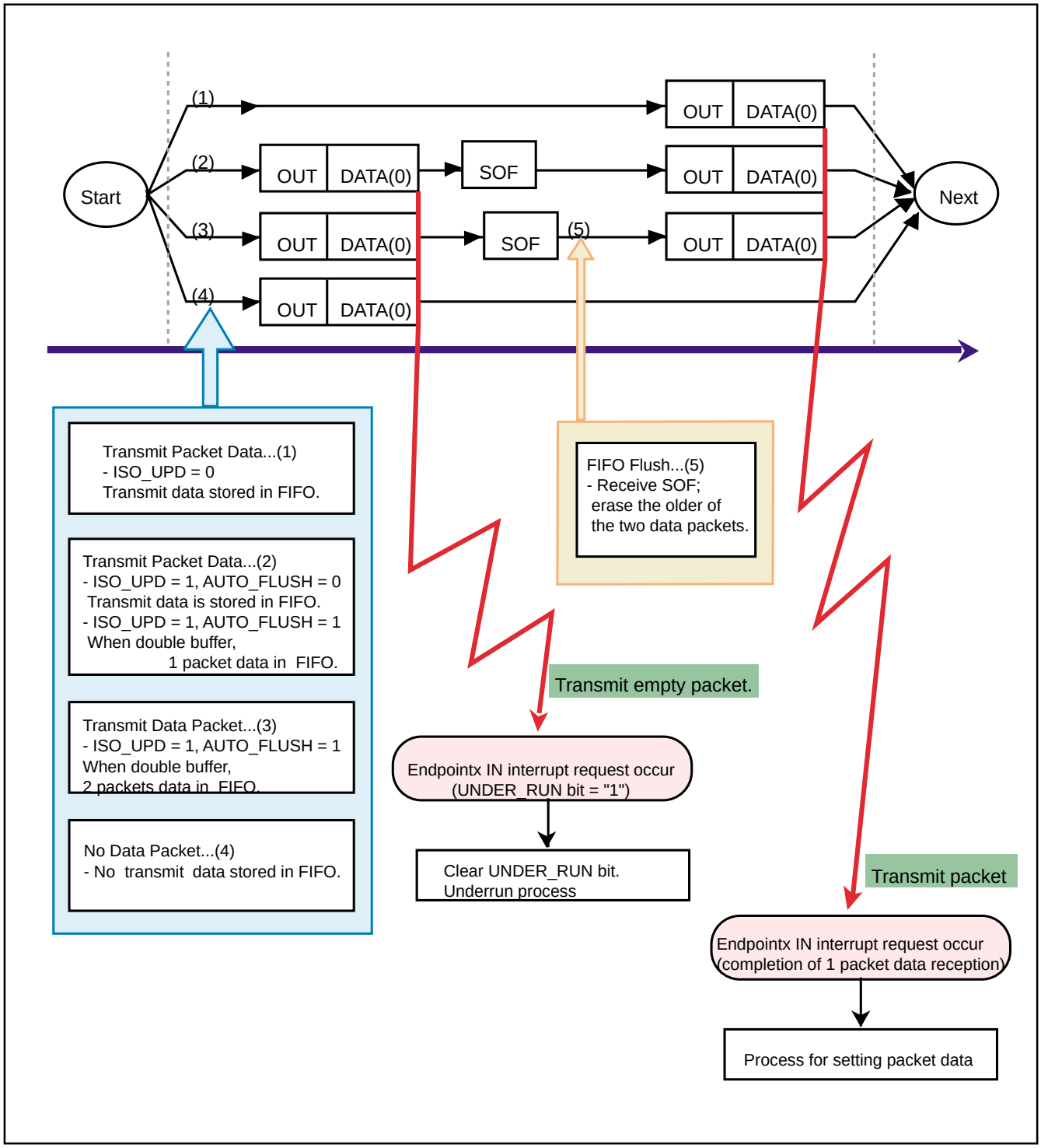


Fig4.4.3 Endpointx: Isochronous Data Transmit Example

4.4.5 Data transmit in interrupt transfer

Conditions for interrupt transfers are the same as those for bulk transfers.

(1) Setting the transfer type

In order to use Endpointx IN for interrupt transfers, set the USB Endpointx IN control status register to the interrupt transfer setting.

- Normal interrupt transfers
  Set INTPT bit to “0”.
- Rate feedback interrupt transfers
  The rate feedback function is an ISO function that feeds back information to the host concerning the transfer rate. If the isochronous device being used is equipped with the rate feedback function, use the rate feedback interrupt transfer type by setting the INTPT bit to “1”.

When setting up for interrupt transfers, set the ISO/TOGGLE_INTI bit to “0”. This is a double function bit and is used for initialization of the toggle sequence bit. In order to initialize the toggle sequence for the interrupt transfer endpoint, please execute the following procedure.

<Initialization of Toggle Sequence>

1. Set ISO/TOGGLE_INITI bit to “1”.
2. Set ISO/TOGGLE_INITI bit to “0”.
3. Fix ISO/TOGGLE_INITI bit to “0” for the remainder of the interrupt transfers.

(2) Data transmit operation

- Normal interrupt transfers
  Endpointx IN operations for normal interrupt transfers are the same as those for bulk transfers.

-Rate feedback interrupt transfers

In rate feedback interrupt transfers, the endpoint always has data ready to transmit back to the host. By setting the INTPT bit to “1”, regardless of the setting of the IN_PKT_RDY bit, the IN FIFO data will be transmitted in response to an IN token from the host. The device does not transmit a NAK in response to an IN token from the host. Other than this point, the operations of Endpointx IN during the rate feedback interrupt transfer are identical to those of the bulk IN transfer. (In normal bulk transfers, when the IN_PKT_RDY bit is not set to “1”, a NAK, not a packet data, is returned to the host.)

But, this function, it is a premise that ACK is always responded from host PC after 7641 group transmitted data for IN token.

When updating the IN FIFO data in a rate feedback interrupt transfer, Flush endpointx IN FIFO and set up the next packet data to be transmitted.

Use the following procedure to set up for rate feedback interrupt transfers.

<Rate feedback interrupt transfer setup procedure>

1. Set the USB Endpointx IN max. packet size register to single buffer. (max. packet size > 1/2 FIFO size)
2. Set INTPT bit to “1”.
3. Flush-erase the FIFO. Do not set the FLUSH bit while a data packet is in transmission.
4. Write data to FIFO, set IN_PKT_RDY bit.
5. To update information at a later point, repeat steps 3 and 4.

Chapter 5
Supplement

5.1 USB Timing

Figure 5.1.1 shows the USB connection/disconnection timing for self-powered applications. Figure 5.1.2 shows the USB connection/disconnection timing for operations using bus-powered supply.

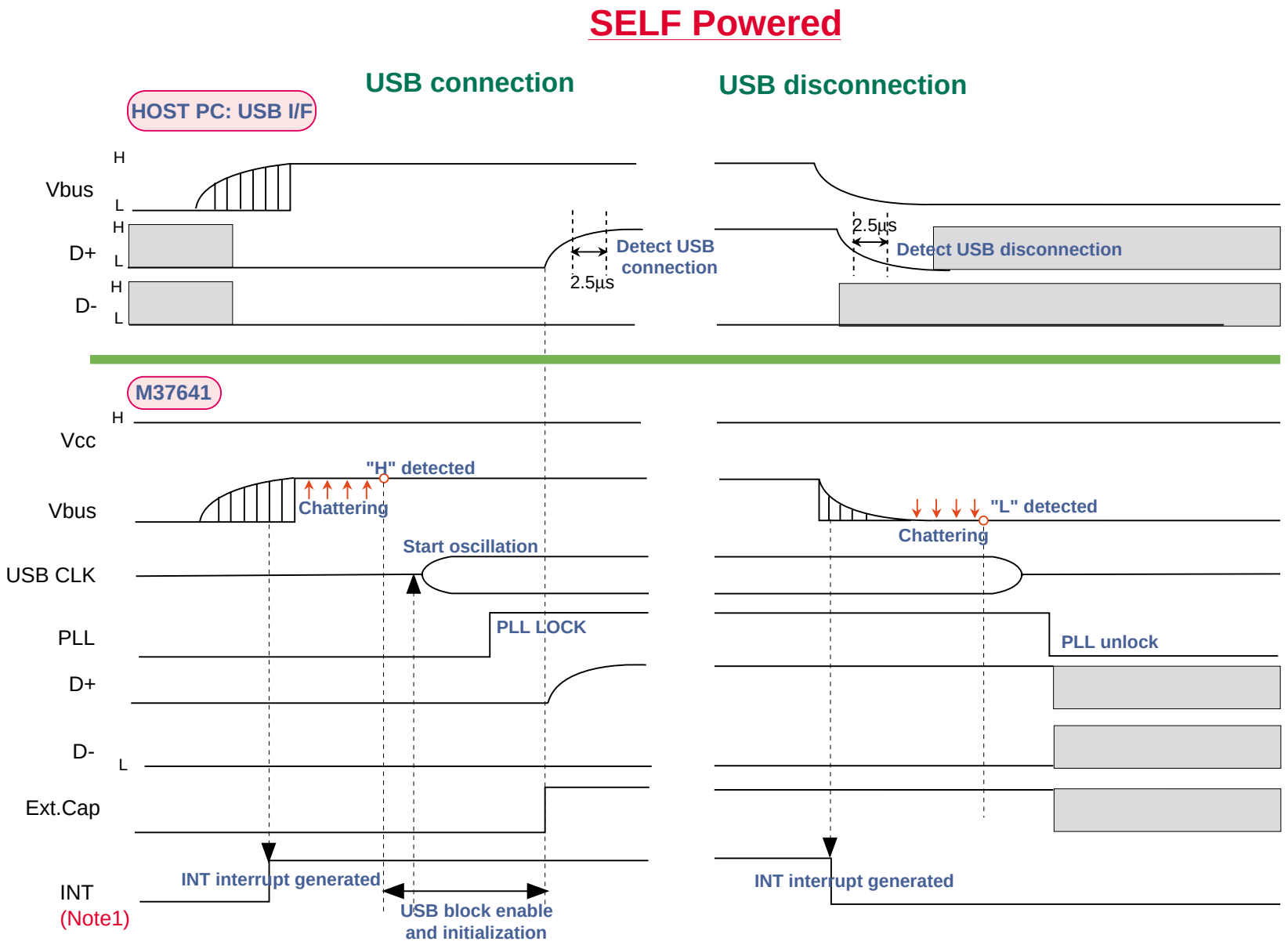


Figure 5.1.1 USB Connection/Disconnection Timing Chart (Self-Powered Applications)

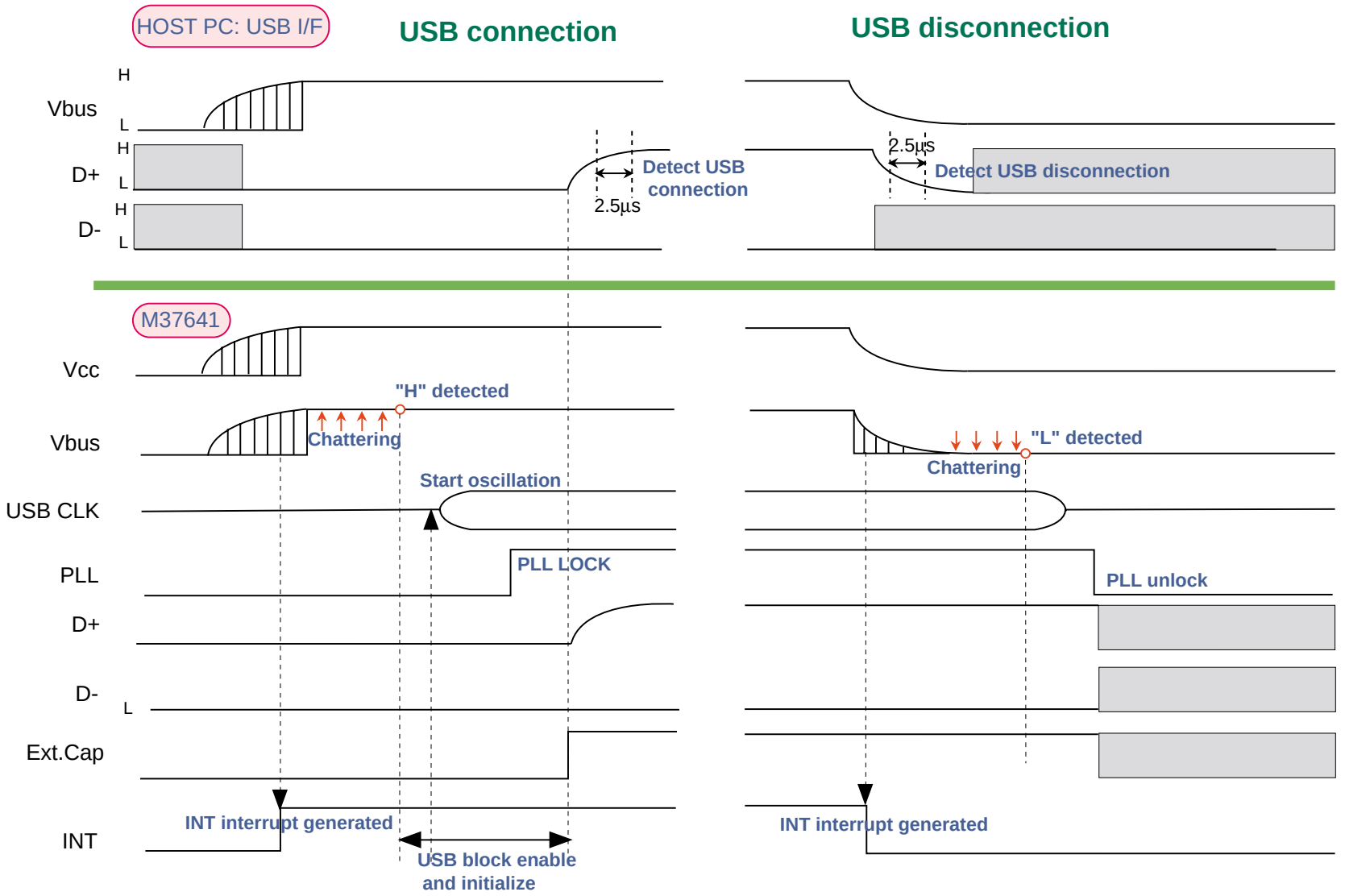


Figure 5.1.2 USB Connection/Disconnection Timing Chart (Bus-Powered Applications)

Revision History	7641 Group Application Notes

Ver.No.	Date	Page	Contents
1.0	Sep.13'01		First edition issued
1.1	Sep.18'01	12 33 53 63	Change 'Unconnected state' to 'Unconfigured state'. Add 'Full speed operations' to figure title. Note of Fig.4.4.1: Add 'Otherwise ------'. Change 'Vbus power supply' to 'BUS powered'.
1.2	Jan.11'02	30	Add 'Disable USB line driver' to figure 3.2.2.
1.3	Oct.04'06	- -	'Notes on USB Communication' is added. Change company name 'Mitsubishi' to 'Renesas'.

RENESAS

RENESAS Single-Chip Microcomputer
7641 Group
USB Application Notes Ver.1.3
RENESAS TECHNOLOGY CORPORATION.
RENESAS SOLUTIONS CORPORATION.