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

38K2 Group

USB Application Notes

Renesas Single-Chip Microcomputers

Renesas Electronics
www.renesas.com

Ver.1.1 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 '38K2 Group USB Application Notes'**

  *Explicit samples for 38K2 Group USB implementation and Hub implementation

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

- **Related Materials**

  *RENESAS Microcomputers 38K2 Group Data Sheet

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

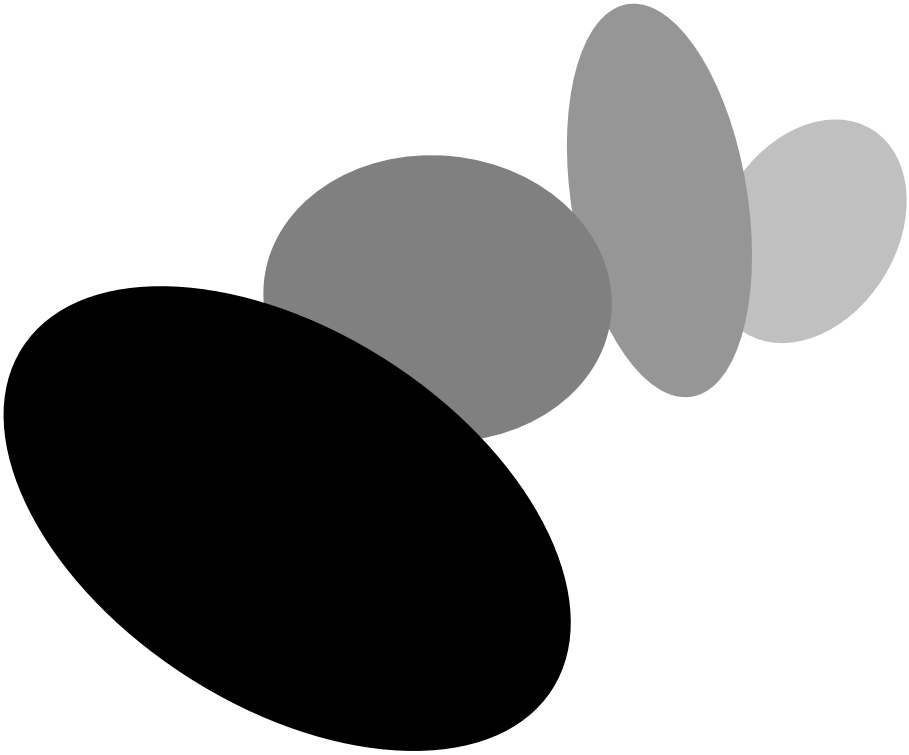
  *USB Specification Ver2.0

  [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 Initial Setting

USB initialization consists of enabling the USB device block and USB HUB device block every endpoint setting of the transfer type and the interrupt configuration.

1.1. USB Function Outline

The 38K2 Group is equipped with a USB Device Control Unit (USBDCU = USB Block). The USBDCU enables highly effective interfacing with the host computer. This circuit is in compliance with USB Specification Version 2.0 and supports the following four transfer types, as defined in the specification.

- Control Transfer
  Mainly used for data transfer during setup, when the USB device is first attached (two-way transfer).

- Isochronous Transfer
  Used for transfer of relatively large quantities of data, such as voice or movie data, which requires a constant transfer rate (one-way transfer).

- Interrupt Transfer
  Used for transfer of relatively small quantities of data to be transferred in real-time, such as keyboard or mouse input (one-way transfer).

- Bulk Transfer
  Used for transfer of data that does not require real-time operations, such as printer output, etc. (one-way transfer).

The USB Device Control Unit has 4 endpoints (Endpoint 0 to 3). Endpoints 1 to 3 can be used for Isochronous, Bulk and Interrupt transfers. Endpoint 0 can only provide Control transfer.

Any data transfer is inhibited during reset on 38K2 MCUs. Initialize the necessary endpoints for the type of data transfer to be used. The USB block provides the following USB interrupts: USB bus reset interrupt, USB SOF interrupt and USB device interrupt. Make sure the USB block is in the enabled-state before using the USB Function.

The USBDCU is equipped with 2 USB address and 6 endpoints; one address is for the USB Function, 1 address is for the Hub function.

The USB address for the Hub function has 2 endpoints (Endpoints 0 and 1). Endpoint0 handles Control transfers only. Endpoint1 handles interrupt transfers only and is used for transferring update information concerning the USB Hub/down-port status.

The USB address for peripheral functions has 4 endpoints (Endpoints 1 to 3). Endpoints 1 to 3 can be used for Isochronous, Bulk and Interrupt transfers. Endpoint0 is fixed for Control transfers only. Furthermore, all types of transfers in 38K2 Group MCUs are inhibited during a reset. All of these details should be considered when setting the endpoints for transfer.

Furthermore, 38K2 USB block supports the following USB interrupts: USB bus reset interrupt, USB SOF interrupt, and USB device interrupt. The USB block must be set to the enabled state before using the USB Function.

1.2. USB HUB Function Outline

The 38K2 Group USBHCU, as described above, is equipped with functions for down-port control as well as Host-to-38K2 Group communications. This circuit is compliant with USB Specification Version 2.0.
The USBHCU controls 2 external down-ports and 1 internal down-port. The internal down port is assigned to the 38K2 Group USB Function.

1.3.USB Related Registers

There are a total of 18 USB-related registers, including the index register and the register window. Nine registers are defined in one register window (same address area), which can be selected by the index register. In addition to these registers, the USBDCU has three interrupts: USB bus reset interrupt, USB SOF interrupt and USB device interrupt. See Chapter 2 for more details concerning USB interrupts.

Table 1.1 is a list of USB-related registers and Table 1.2 is a list of registers that require an index-switch.

Refer to the 38K2 Group data sheet for the bit names of each register.

USB Hub related registers are used for down port control and are described in detail in Chapter 4.

Table 1.1. USB-Related Registers

Address	Symbol	Register Name
0010H	USBCON	USB Control Register
0011H	USBAE	USB Function Enable Register
0012H	USBA0	USB Function Address Register
0013H	USBA1	USB HUB Address Register
0014H	FNUML	Frame Number Register LOW
0015H	FNUMH	Frame Number Register HIGH
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
0018H	USBINDEX	Endpoint Index Register
0019H	EPXXREG1	Endpoint Field Register 1
001AH	EPXXREG2	Endpoint Field Register 2
001BH	EPXXREG3	Endpoint Field Register 3
001CH	EPXXREG4	Endpoint Field Register 4
001DH	EPXXREG5	Endpoint Field Register 5
001EH	EPXXREG6	Endpoint Field Register 6
001FH	EPXXREG7	Endpoint Field Register 7

0FECH	EPXXREG8	Endpoint Field Register 8
0FEDH	EPXXREG9	Endpoint Field Register 9

0FF8H	PLLCON	PLL Control Register

003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1

Refer to Table 1.2

Table 1.2. List of Registers Switched by Endpoint Index Register (USBINDEX)

Index Register Value	Address	Symbol	Register Name
00H	0019H	EP00STG	EP00 Status Register
	001AH	EP00CON1	EP00 Control Register 1
	001BH	EP00CON2	EP00 Control Register 2
	001CH	EP00CON3	EP00 Control Register 3
	001DH	EP00REQ	EP00 Interrupt Factor Register
	001EH	EP00BYT	EP00 Byte Number Register
	001FH	-	-
	0FECH	-	-
	0FEDH	EP00BUF	EP00 Buffer Area Setup Register
01H	0019H	EP01CFG	EP01 Setup Register
	001AH	EP01CON1	EP01 Control Register 1
	001BH	EP01CON2	EP01 Control Register 2
	001CH	EP01CON3	EP01 Control Register 3
	001DH	EP01REQ	EP01 Interrupt Factor Register
	001EH	EP01BYT0	EP01 Byte Number Register 0
	001FH	EP01BYT1	EP01 Byte Number Register 1
	0FECH	EP01MAX	EP01 MAX Packet Size Register
	0FEDH	EP01BUF	EP01 Buffer Area Set Register
02H	0019H	EP02CFG	EP02 Set Register
	001AH	EP02CON1	EP02 Control Register 1
	001BH	EP02CON2	EP02 Control Register 2
	001CH	EP02CON3	EP02 Control Register 3
	001DH	EP02REQ	EP02 Interrupt Factor Register
	001EH	EP02BYT0	EP02 Byte Number Register 0
	001FH	EP02BYT1	EP02 Byte Number Register 1
	0FECH	EP02MAX	EP02 MAX Packet Size Register
	0FEDH	EP02BUF	EP02 Buffer Area Set Register
03H	0019H	EP03CFG	EP03 Setup Register
	001AH	EP03CON1	EP03 Control Register 1
	001BH	EP03CON2	EP03 Control Register 2
	001CH	EP03CON3	EP03 Control Register 3
	001DH	EP03REQ	EP03 Interrupt Factor Register
	001EH	EP03BYT0	EP03 Byte Number Register 0
	001FH	EP03BYT1	EP03 Byte Number Register 1
	0FECH	EP03MAX	EP03 MAX Packet Size Register
	0FEDH	EP03BUF	EP03 Buffer Area Set Register
04H	0019H	EP10STG	EP10 Status Register
	001AH	EP10CON1	EP10 Control Register 1
	001BH	EP10CON2	EP10 Control Register 2
	001CH	EP10CON3	EP10 Control Register 3
	001DH	EP10REQ	EP10 Interrupt Factor Register
	001EH	EP10BYT	EP10 Byte Number Register
	001FH	-	-
	0FECH	-	-
	0FEDH	EP10BUF	EP10 Buffer Area Setup Register
05H	0019H	EP11CFG	EP11 Setup Register
	001AH	EP11CON1	EP11 Control Register 1
	001BH	EP11CON2	EP11 Control Register 2
	001CH	-	-
	001DH	EP11REQ	EP11 Interrupt Factor Register
	001EH	EP11BYT0	EP11 Byte Number Register 0
	001FH	-	-
	0FECH	-	-
	0FEDH	EP11BUF	EP11 Buffer Area Setup Register

*USBINDEX: 00H to 03H is for the USB Function, 04H to 05H is for USB Hub use.

1.4. How to Use the USB Function

In order to use the USB HUB function, enable the USB device block and USB HUB device block, and set the appropriate endpoints. Please refer to the 38K2 Group Data Sheet [USB Register List] and [HUB Register List] for details on how to set each register.

Table 1.3 shows the list of registers that enable the USB device block.

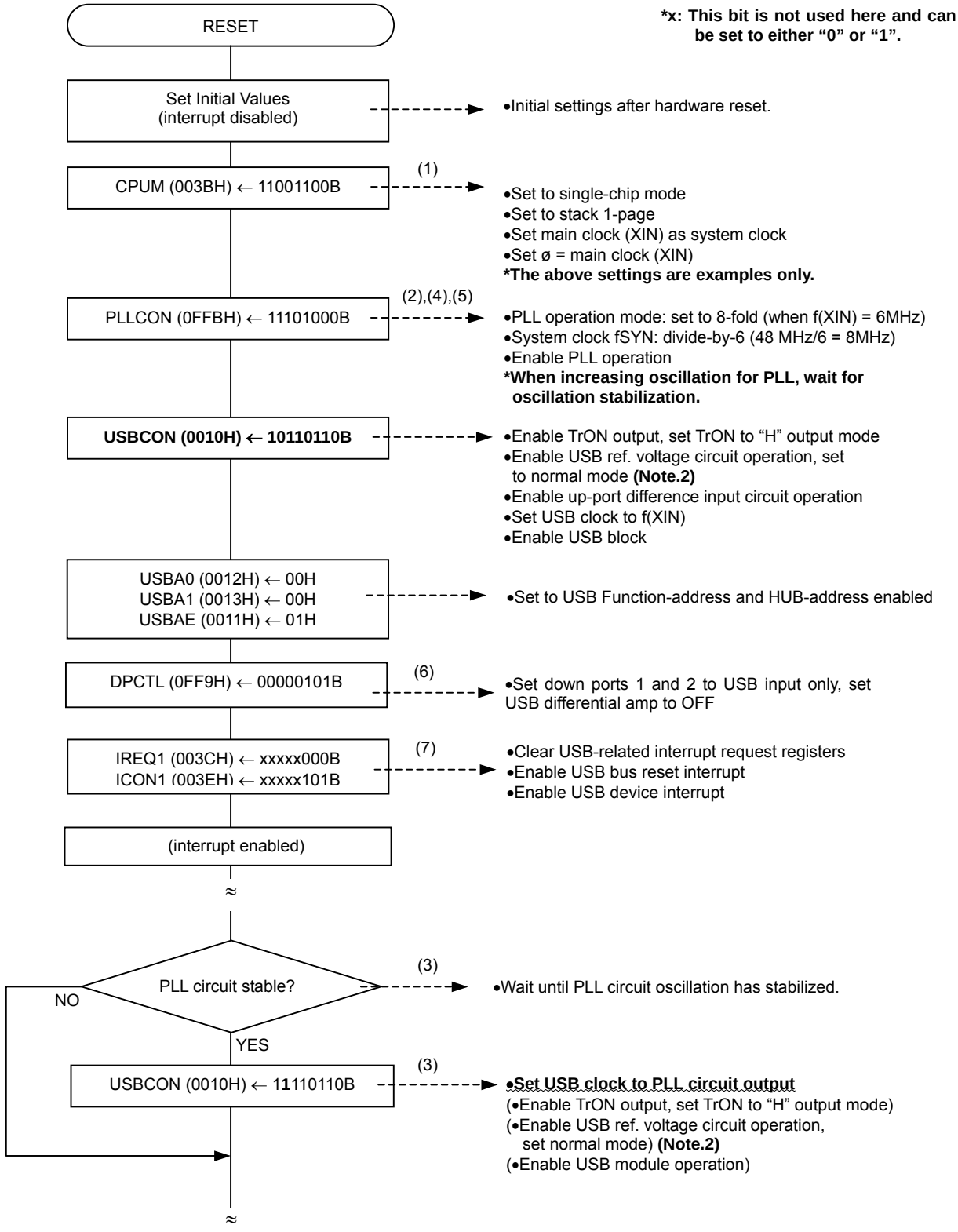
Table 1.3. USB Registers that Enable USB Block

Address	Symbol	Register Name
0010H	USBCON	USB Control Register
0011H	USBAE	USB Function Enable Register
0012H	USBA0	USB Function Address Bit
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
0018H	USBINDEX	Endpoint Index Register
0FF8H	PLLCON	PLL Control Register
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1

1.4.1. How to Enable the USB Device Block

Always enable the USB block after the oscillation of the MCU has stabilized. After enabling the PLL operation, add a wait period (about 1ms) to avoid any instability caused by the clock. Then enable the PLL circuit output after the wait period. By enabling the TrON port of the USBCON register to output “H”, the D+ signal is pulled up and the host (upstream-side) detects the attachment of a USB device.

Figure 1.1 shows how to enable the USB block after a hardware reset. Figure 1.2 shows how to setup after a USB bus reset interrupt.



*x: This bit is not used here and can be set to either "0" or "1".

Note 1: This flowchart only shows steps that are related to the USB Function/HUB.
Note 2: This setting is at Vcc=5V operation. At Vcc=3V operation, USB reference voltage enable bit (VREFE) must be set to disable ("0") (USB reference voltage control bit is ignored).

Figure 1.1 USB Device Block Enable and Initialization Process Example (at hardware reset)

The following are detailed descriptions of the steps shown in [Figure 1.1 USB Device Block Enable and Initialization Process Example (at hardware reset)].

(1) After setting the PLLCON register to generate a 48MHz USB clock, enable the PLL operation. The actual USB clock (fUSB) is supplied by setting the USB clock selection bit of the USBCON register to fVCO.
(2) The PLL circuit needs about 1ms for the output clock to stabilize after operation is enabled. Therefore, you must set the USB clock selection bit of USBCON to external oscillation f(XIN) for the first USB clock f(USB). At hardware reset, the USB clock select bit is “0” already.
(3) After PLL output clock stabilization, set the USBCON register USB clock select bit to “1” and the PLL clock output circuit (fVCO) is enabled..
(4) The USBCON USB reference voltage control bit is set to “0” in normal mode and “1” in suspend mode.*1
(5) The USBCON USB module operation enable bit is set to “1” in USB module operation enabled.

◆About USB module operation enable bit(USBE)

The USB module is initialized by setting the USB module operation enable bit of the USBCON register to “0”.

The registers that are initialized by a USB module reset are listed below.
For more details, refer to the description “at S/W reset” in the [USB Related Registers] Chapter of the 38K2 Group Data Sheet.

Note that only USBA0 (USB Function Address) and USBA1 (USB HUB Address) is automatically initialized (“00” state) when a USB bus reset interrupt is generated.

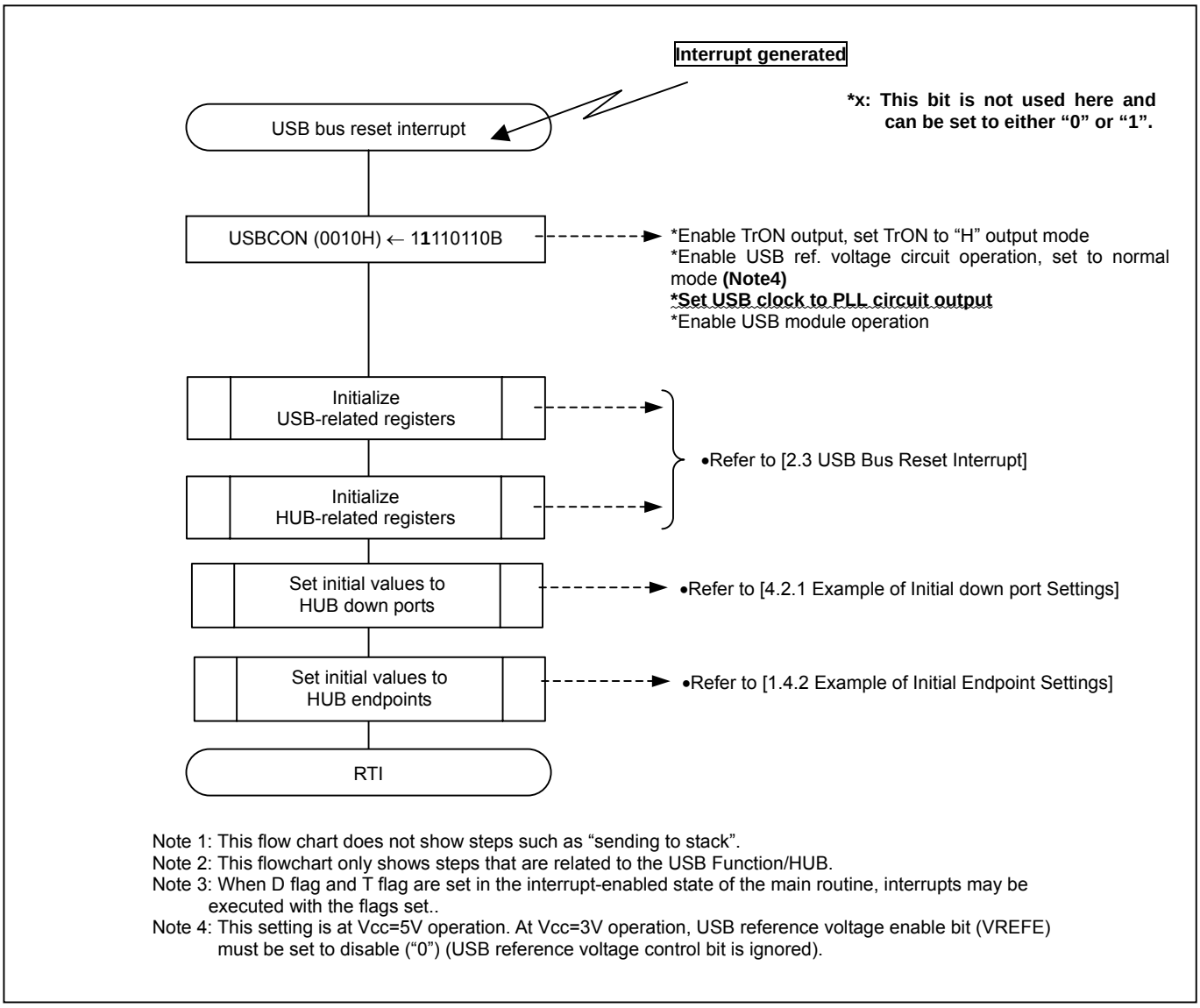
Table 1.4 USB Related Registers at S/W Reset

Address	Symbol	Register Name
0012H	USBA0	USB Function Address Register
0013H	USBA1	USB HUB Address Register
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
0019H	EPxxSTG	EP00 Status Register (when USBINDEX = “00”)
001DH	EPxxREQ	EPXX Interrupt Factor Register (XX = USBINDEX = “00” to “03”)

(6) Set the down-portx function selection bit of the down-port control register (DPCTL register) to USB input only port and the USB differential amp to OFF.
(7) The USB bus reset interrupt and USB Device interrupt are enabled in the ICON1 register setting.

Other USB-related registers are set after the USB bus reset interrupt is executed. Refer to [Chapter 2.3 USB Bus Reset Interrupt] for more details.

*1 This setting is at Vcc=5V operation. At Vcc=3V operation, USB reference voltage enable bit (VREFE) must be set to disable (“0”) (USB reference voltage control bit is ignored).



Note 1: This flow chart does not show steps such as "sending to stack".
Note 2: This flowchart only shows steps that are related to the USB Function/HUB.
Note 3: When D flag and T flag are set in the interrupt-enabled state of the main routine, interrupts may be executed with the flags set..
Note 4: This setting is at Vcc=5V operation. At Vcc=3V operation, USB reference voltage enable bit (VREFE) must be set to disable ("0") (USB reference voltage control bit is ignored).

Figure 1.2 Initialization Process Example at USB Bus Reset Interrupt

1.4.2. Initialization of USB HUB Endpoints

After specifying the endpoints to be set in the USB index register, set the MAX OUT packet size, transfer type, etc., for each endpoint to be used. Enable the USB interrupt as necessary.
When setting the start address for the buffer area of each endpoint, make sure the addresses for single buffer and double buffer do not overlap. For more details, refer to [Chapter 3: USB Transfer/Receive].
Figure 1.3 shows an example of initial endpoint settings. Make sure the USB block and USB clock are both enabled before programming the related registers.

After specifying the endpoints to be set in the USB index register, set the endpoints for each register. Enable USB interrupts as necessary

Figure 1.3 shows the initial settings of USB Hub endpoints. Always enable the USB device block, USB Hub device block, and USB block before writing to Hub related registers.

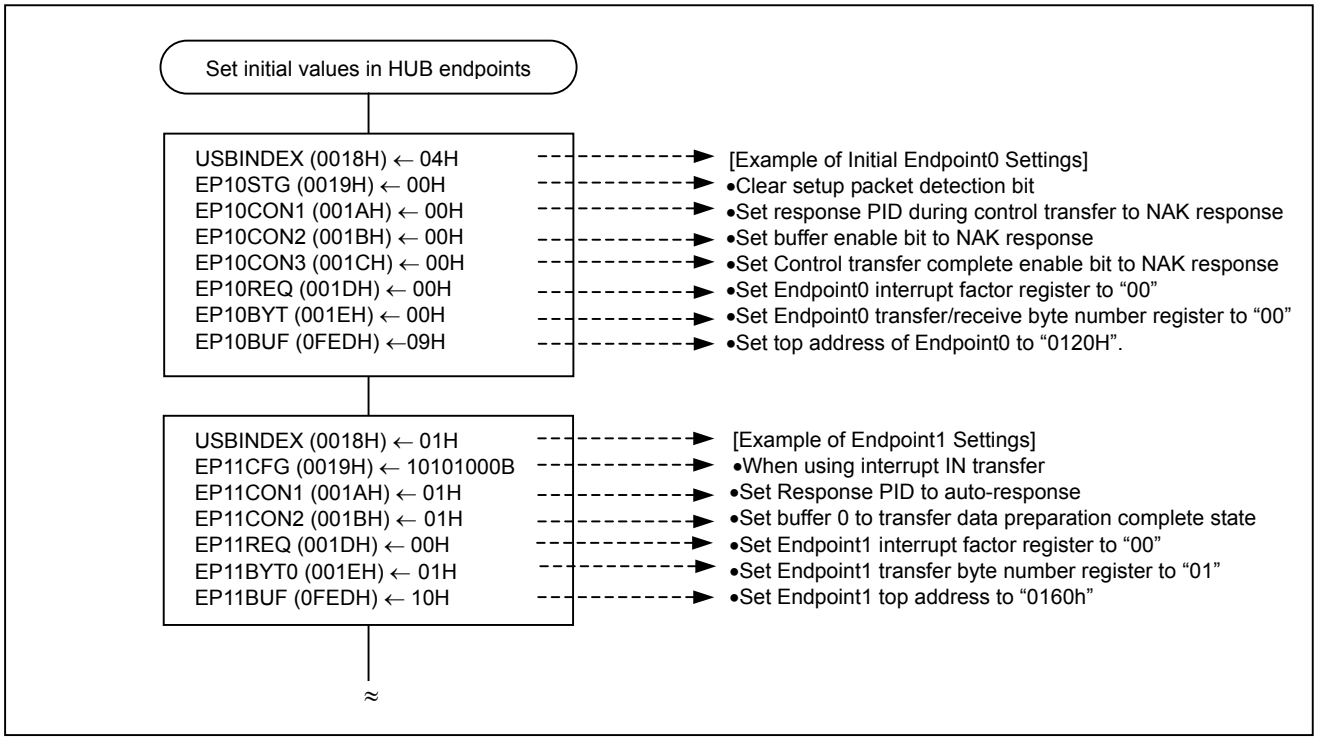
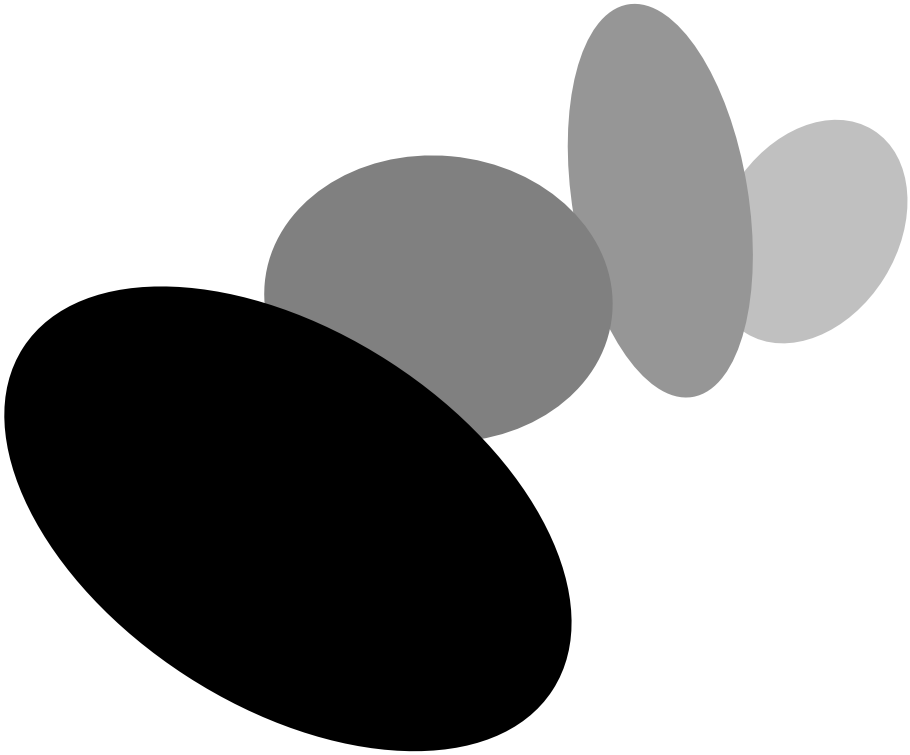


Figure 1.3 Examples of Initial Endpoint Settings



Chapter 2
State Transition and USB Interrupts

The 38K2 Group USB Interrupts consist of the device interrupt, the SOF interrupt and the Bus Reset Interrupt. This Chapter explains the USB Interrupts and State Transition.

2.1. Device State Transition

The USB device moves from one state to another according to the current operations. In the 38K2 Group, device state transition occurs with each USB interrupt (reset, suspend, resume) or Endpoint0 device standard request processing.

Figure 2.1 shows device state transitions for the 38K2 Group.

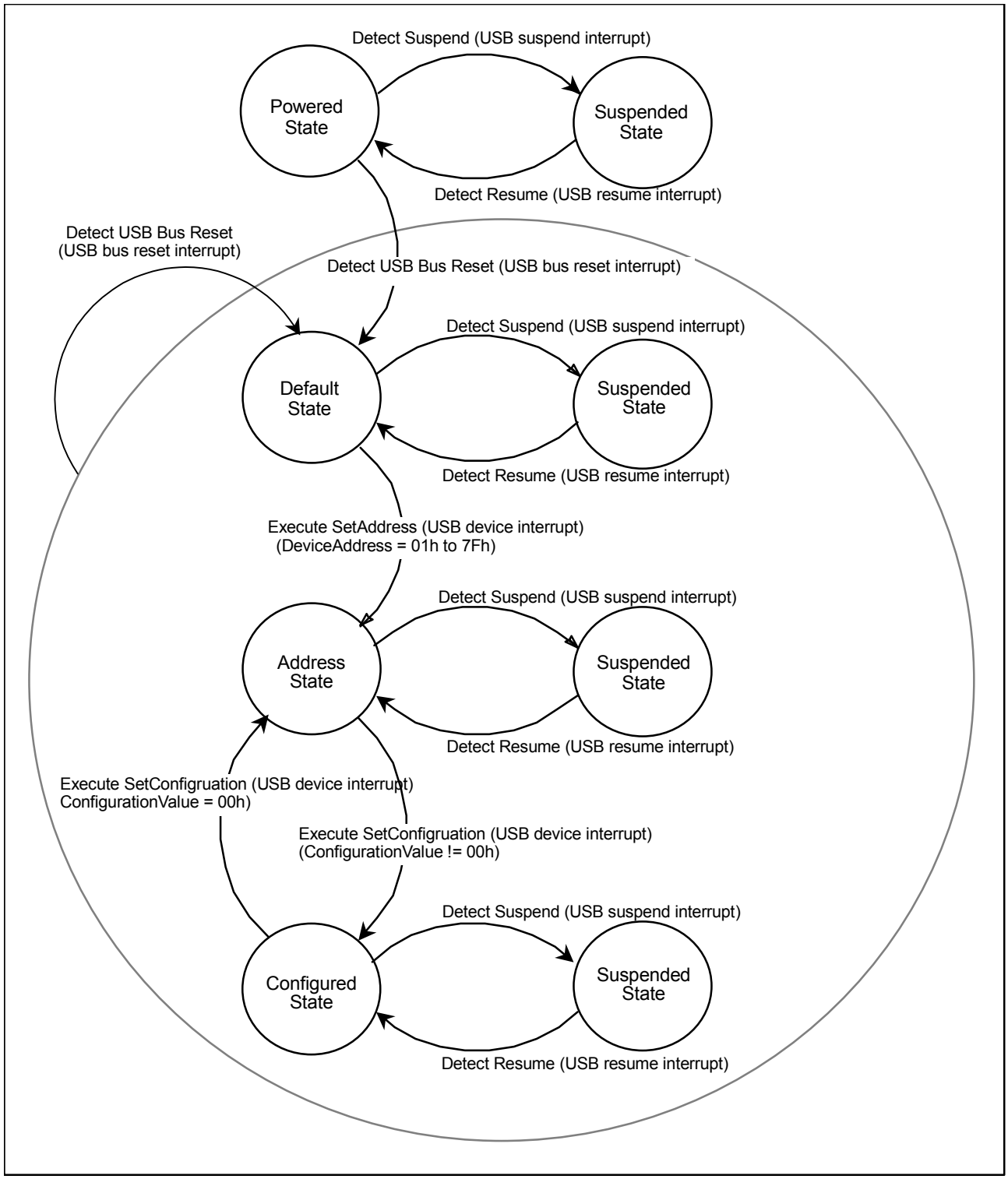


Figure 2.1 Device State Transitions

2.2. How to Use USB Interrupts

The 38K2 Group has three basic sources of USB interrupts: bus reset, device, and USB SOF. The USB device interrupt is used for data flow control and special USB signals (suspend, resume). Interrupt vector addresses are shown in Table 2.1 and registers related to USB interrupts are shown in Table 2.2

For details concerning the states and interrupts of each register, refer to Chapters [Interrupts] and [USB Register List] in the 38K2 Group Data Sheet.

Table 2.1 Interrupt Vector Addresses and Priority

Interrupt Source	Priority	Vector Address		Interrupt Request Generation Conditions
		Upper	Lower	
Reset	1	FFFDH	FFFCH	Reset
USB bus reset	**2**	**FFFBH**	**FFFAH**	**Detection of USB bus reset signal (SE0 for 2.5 µs)**
USB SOF	**3**	**FFF9H**	**FFF8H**	**Detection of USB SOF signal**
USB device	**4**	**FFF7H**	**FFF6H**	**Detection of Resume signal (K state or SE0) or Detection of Suspend signal (bus idle for 3ms) or Transaction complete**
External bus	5	FFF5H	FFF4H	

Table 2.2 USB Registers and Interrupt Registers

Address	Symbol	Register Name
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
0018H	USBINDEX	Endpoint Index Register
0019H	EPXXREG1	Endpoint Field Register 1
001AH	EPXXREG2	Endpoint Field Register 2
001BH	EPXXREG3	Endpoint Field Register 3
001CH	EPXXREG4	Endpoint Field Register 4
001DH	EPXXREG5	Endpoint Field Register 5
001EH	EPXXREG6	Endpoint Field Register 6
001FH	EPXXREG7	Endpoint Field Register 7
0FECH	EPXXREG8	Endpoint Field Register 8
0FEDH	EPXXREG9	Endpoint Field Register 9
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1

2.3. USB Bus Reset Interrupt

The USB bus reset interrupt is generated when SE0 continues for 2.5µs or more on the USB bus. Table 2.3 lists registers related to USB devices and Figure 2.2 shows an example of the USB device interrupt routine.

(1) Cause of USB bus reset detection

A reset is detected when SE0 ("L" level in the D+/D- line) continues on the USB bus for at least 2.5µs.

(2) Action when a USB bus reset is detected

When a USB bus reset is detected as described above, the USB bus reset interrupt request bit of IREQ1 (Interrupt Request Register) is set to "1".

(3) USB bus reset interrupt settings

Set the USB bus reset interrupt enable bit of ICON1 (Interrupt Control Register) to "1".

(4) Conditions for accepting a USB bus reset interrupt

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

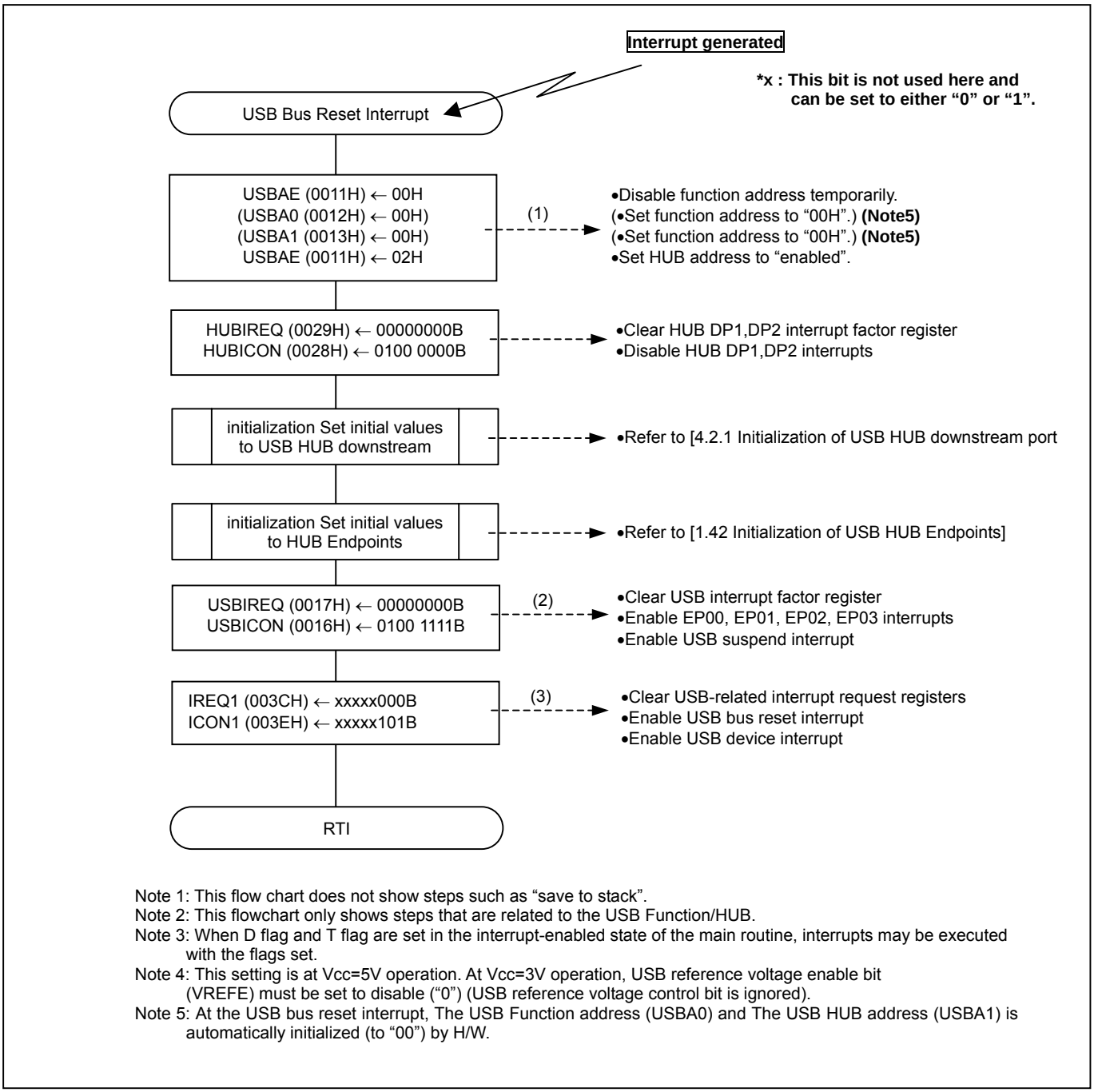
- I Flag (Interrupt Disable Flag) of the processor status register is "0" (interrupt enabled).
- The USB bus reset interrupt enable bit of ICON1 is "1" (enabled state)
- USB bus reset interrupt request bit of IREQ1 is "1" (interrupt generated)

(5) USB bus reset interrupt routine

- Figure 2.2 shows an example of the USB bus reset interrupt routine.
- When a USB bus reset interrupt is received, the interrupt request flag is cleared at the same time the interrupt is generated. Therefore, it is not necessary to clear the flag in the interrupt routine.

Table 2.3 Registers related to USB Reset Interrupt

Address	Symbol	Register Name
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1



Note 1: This flow chart does not show steps such as “save to stack”.
Note 2: This flowchart only shows steps that are related to the USB Function/HUB.
Note 3: When D flag and T flag are set in the interrupt-enabled state of the main routine, interrupts may be executed with the flags set.
Note 4: This setting is at Vcc=5V operation. At Vcc=3V operation, USB reference voltage enable bit (VREFE) must be set to disable (“0”) (USB reference voltage control bit is ignored).
Note 5: At the USB bus reset interrupt, The USB Function address (USBA0) and The USB HUB address (USBA1) is automatically initialized (to “00”) by H/W.

Figure 2.2 Example of USB Bus Reset Interrupt Routine

The following are detailed descriptions of the steps shown in [Figure 2.2 Example of USB Bus Reset Interrupt Routine].

(1) USB address initialization
- When setting an address to the USBA1 register, if the USB HUB enable bit AD1E (USBAE register) changes from “00H” to”01H”, the contents of the USBA1 register become valid.

(2) USB-related interrupt settings #1
- Clear USBIREQ.
- Enable SUSE (USB Suspend Interrupt) and EP00E (Endpoint0 interrupt) of the USBICON register.
- Enable the SUSE interrupt and Endpoints 1 to 3 interrupts (EP0xE) as necessary.

*We recommend clearing the interrupt request bits before enabling interrupts.

(3) USB-related interrupt settings #2
- Clear IREQ1 interrupts (USB bus reset, USB SOF, and USB device interrupts).
- Enable USB bus reset and USB device interrupts of ICON1.

2.4. USB Device Interrupt

The USB device interrupt is used for data flow control and USB special signals (suspend, resume). The types of interrupts used for data flow control of USB HUB are listed below.

[USB HUB]
- ♦Interrupt generated when Endpoint0 data transfer/receive is complete
- ♦Interrupt generated when Endpoint1 data transfer is complete

The types of interrupts used for data flow control of USB Function are listed below.

[USB Function]
- ♦Interrupt generated when Endpoint0 data transfer/receive is complete
- ♦Interrupt generated when Endpoint1 data transfer/receive is complete
- ♦Interrupt generated when Endpoint2 data transfer/receive is complete
- ♦Interrupt generated when Endpoint3 data transfer/receive is complete

The following two interrupts are used for USB special signals (suspend, resume).

[USB device (USB HUB + USB Function)]
- ♦USB suspend interrupt
- ♦USB resume interrupt

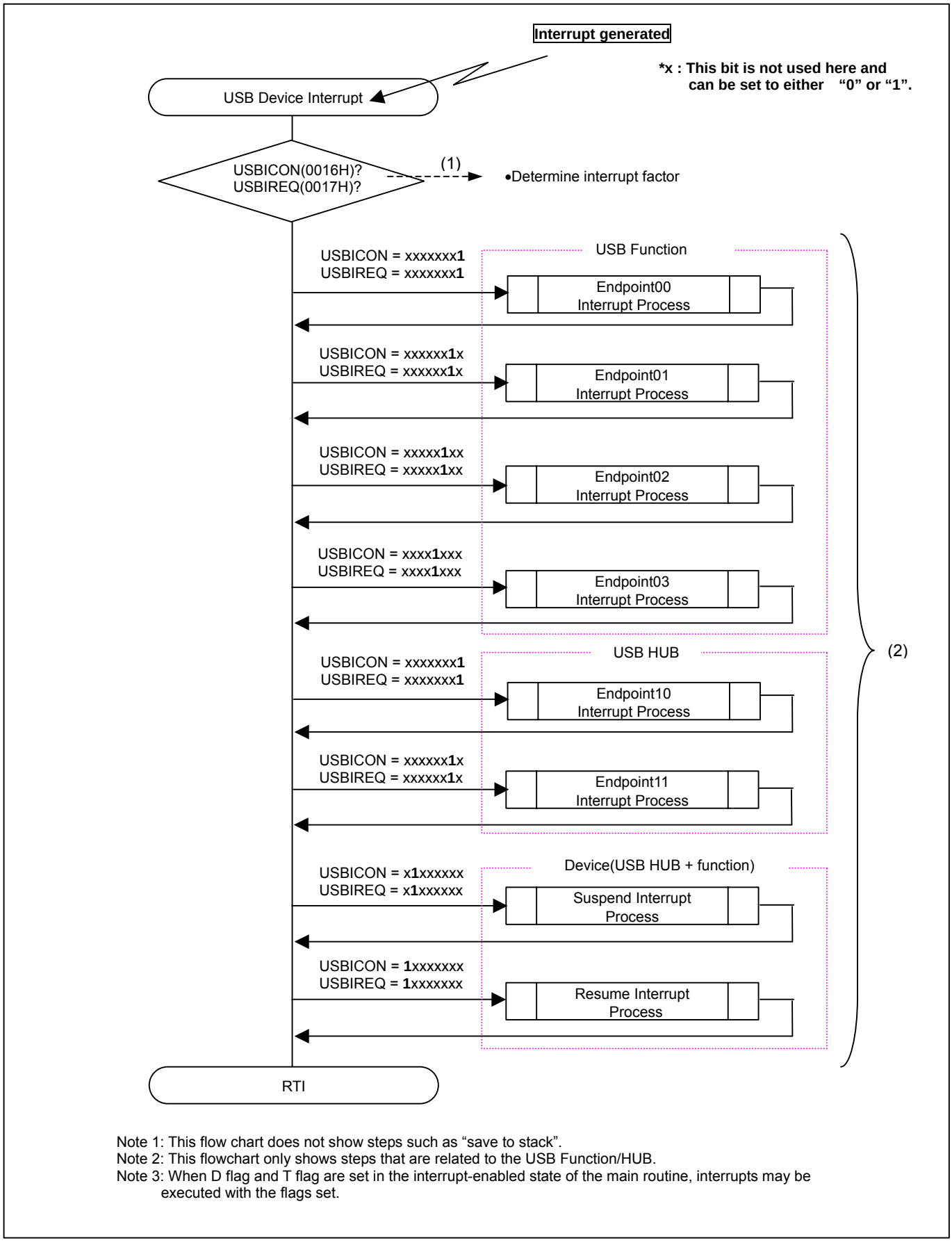
In order to enable the USB device interrupt, set the USB device interrupt enable bit of ICON1 (Interrupt Control Register) to “1”.

To enable any of the above interrupts, set the corresponding USBICON (USB Interrupt Factor Enable Register) bit to “1”. The bit corresponding to USBIREQ (USB Interrupt Request Register) or EP0xREQ (EP0x Interrupt Factor Register, x = 1 to 3) will indicate the interrupt request state.

Table 2.4 lists the USB device interrupt registers and Figure 2.3 shows an example of the USB device interrupt routine.

Table 2.4 USB Device Interrupt Registers

Address	Symbol	Register Name
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
0018H	USBINDEX	Endpoint Index Register
001DH	EP0xREQ	EP0x Interrupt Factor Register (x = 0 to 3)
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1



Note 1: This flow chart does not show steps such as "save to stack".
Note 2: This flowchart only shows steps that are related to the USB Function/HUB.
Note 3: When D flag and T flag are set in the interrupt-enabled state of the main routine, interrupts may be executed with the flags set.

Figure 2.3 Example of USB Bus Device Interrupt Routine

The following are detailed descriptions of the steps shown in [Figure 2.3 Example of Bus Reset Interrupt Routine].

(1) Interrupt Factor Determination Method #1

The bus device interrupt is determined by the AND condition of the enable bit state of USBICON (USB Interrupt Factor Enable Register) and the factor bit state of USBIREQ (USB Interrupt Factor Register). In other words, when both bits are “1”, the next step in the process (the Chapter indicated by (2) in Figure 2.3) is valid.

(2) Interrupt Factor Determination Method #2

To prevent factors from being ignored and left unprocessed when multiple interrupt factors are generated simultaneously, each factor is checked and processed according to order of receipt. Depending on the transmission timing, each time interrupts are generated, all bits of both USBICON and USBIREQ are judged and interrupts are processed accordingly to avoid continuously processing only Endpoint0.

◆ Sample procedures (1) and (2) are included in RENESAS’s “38K2 Group USB Sample Firmware.”

2.4.1. USB HUB/Endpoint Interrupts

The interrupts available for Endpointx are as follows.

♦Interrupt generated when Endpoint0 data transfer/receive is complete
1. Endpoint0 data buffer ready interrupt
2. Control transfer complete interrupt
3. Status stage transition interrupt
4. Endpoint0 SETUP buffer ready interrupt
5. Error interrupt

♦Interrupt generated when Endpoint1 data transfer is complete
1.Endpoint1 data buffer0 ready interrupt

♣ Conditions for accepting a USB HUB/endpointx interrupt

A USB HUB/endpointx interrupt will be generated when all of the following conditions are fulfilled:

•I Flag (Interrupt Disable Flag) of the processor status register is “0” (interrupt enabled).

•The USB device interrupt enable bit of ICON1 (Interrupt Control Register) is “1” (interrupt enabled).

•The USB device interrupt factor bit of IREQ1 (Interrupt Request Register) is “1” (interrupt requested)

•EP1xE (USB HUB/Endpointx Interrupt Enable Bit) which corresponds to the USBICON (Interrupt Factor Enable Register) endpoint is “1” (enabled state).

•EP1x (USB HUB/Endpointx Interrupt Bit) which corresponds to the USBIREQ (Interrupt Factor Register) endpoint is “1” (interrupt requested).

*This bit is set when at least one bit of EP1x is set to “1”. This cannot be set or cleared by software.

•The interrupt bits that correspond to the endpoint (value specified by USBINDEX (Endpoint Index Register)) and EP1xREQ (EP1x Interrupt Factor Register) are “1” (interrupt requested).

(When referencing EP1xREQ, set the endpointx that corresponds to USBINDEX.)

♣ USB Hub/endpointx interrupt processing

•Refer to [Chapter 3.2.3 Endpoint0 Interrupt] and [Chapter 3.3.3 Endpoint1 Interrupt] for examples of interrupt processing.

•When a USB HUB/endpointx interrupt is generated, the USB Device Interrupt Request Flag (the root flag) is cleared as soon as the interrupt is accepted. Therefore, there is no need to clear the flags when processing. EP1x of USBIREQ cannot be cleared by software. By clearing the EP1xREQ that corresponds to the endpoint (the value specified in USBINDEX) to “00”, the endpoint will be cleared by hardware.

2.4.2. USB Suspend Interrupt

The USB suspend interrupt occurs when an idle state is detected on the USB bus lines (upstream-port) for 3.0ms or more.

In the USB suspended state, the USB bus can only bring the total drive current to 500μA or less. Therefore, when in bus-powered operations, the MCU must be put in the low-power consumption mode after suspend is detected.

In order to reactivate the device from the suspended state, a request must be sent to the host via a bus activity on the USB up port or resume request a remote wake-up function.

In the suspended state, the USB clock will not oscillate but the USB block will remain enabled, and therefore, bus activity can be detected by a resume interrupt. (For more details concerning the resume interrupt refer to [Chapter 2.4.3 USB Resume Interrupt].)

If the device supports remote wake-up, a remote wake-up signal can be delivered according to user system specifications. For more details concerning remote wake-up, refer to [Chapter 2.4.3.1 Remote Wake-up].

Table 2.5 shows a list of registers related to USB suspend interrupt.

(1) Cause of USB suspend detection

USB suspend is detected when the bus lines (upstream-port) are idle (D+ line is “H”, the D- line is “L”) for a period of 3.0ms or more.

(2) Action when a USB suspend is detected

When a USB suspend is detected as described above, bit SUS (USB Suspend Interrupt Request Bit) of USBIREQ (USB Interrupt Request Register) is set to “1”.

(3) USB suspend interrupt settings

Set bit SUSE (Suspend Interrupt Enable Bit) of USBICON (USB Interrupt Factor Enable Register) to “1” (enabled state).

(4) Conditions for accepting a USB suspend interrupt

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

- I Flag (Interrupt Disable Flag) of the processor status register is “0” (interrupt enabled).
- The USB device interrupt enable bit of ICON1 (Interrupt Control Register) is “1” (enabled state).
- The USB device interrupt request bit of IREQ1 (Interrupt Request Register) is “1” (requested).
- Bit SUSE of USBICON is “1” (enabled state).
- Bit SUS of USBREQ (USB Interrupt Factor Enable Register) is “1” (requested).

(5) Recovery from the USB suspend status

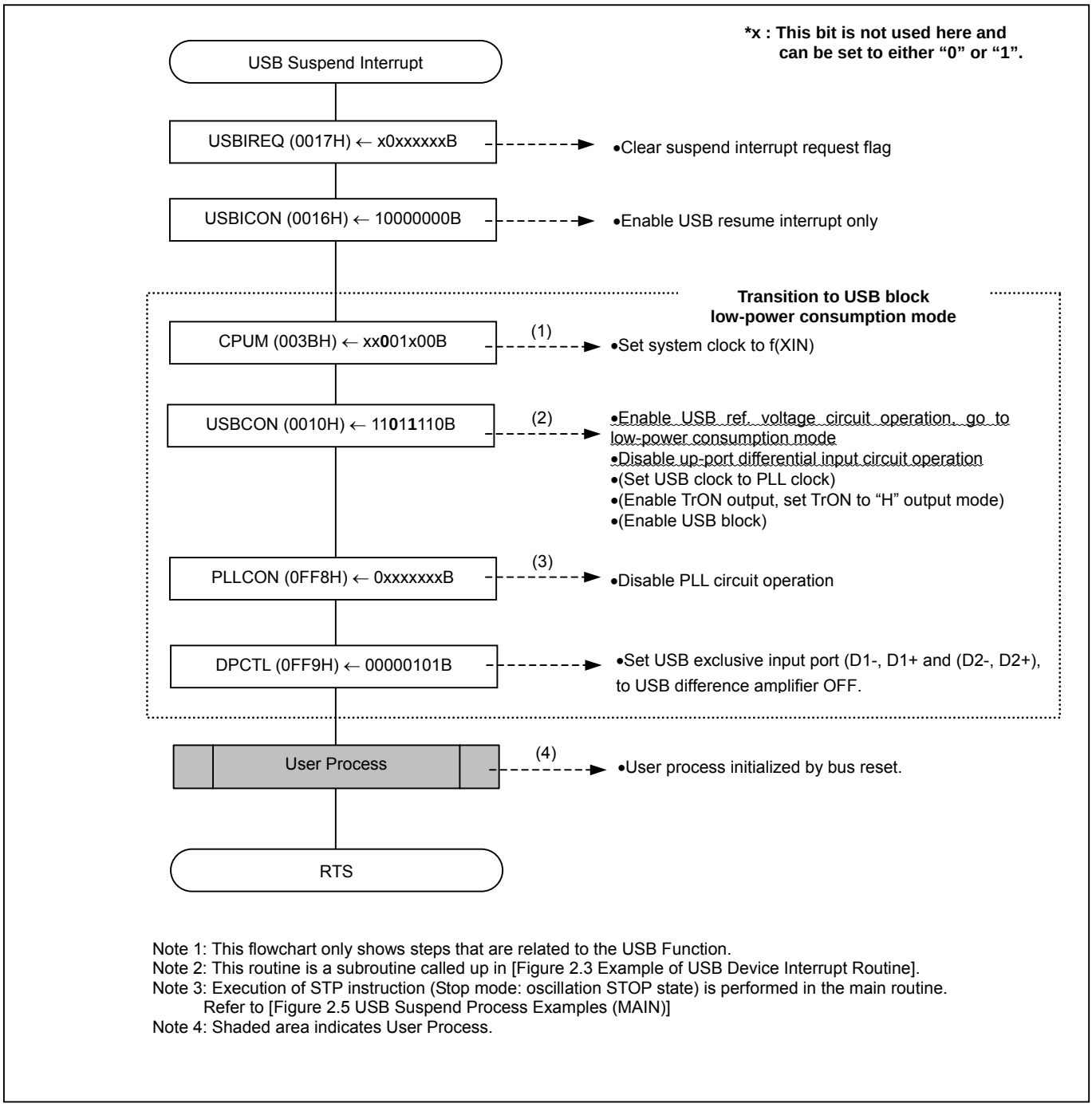
The MCU is recovered from the USB suspend status by a USB resume interrupt or remote wake-up.
For more details concerning remote wake-up signals, refer to [Chapter 2.4.3.1 Remote Wake-Up].
Set one of wake-up factors in the USB suspend interrupt routine.

(6) USB suspend interrupt processing

- An example of the processing routine executed when a USB suspend interrupt is received is shown in Figure 2.4.
- When a USB suspend interrupt is accepted, USB Device Interrupt Request Flag (the root flag) is cleared as soon as the interrupt is accepted and does not need to be cleared by software during the interrupt routine. However, bit SUS of USBIREQ must be cleared by software.

Figure 2.5 USB Suspend Interrupt Registers

Address	Symbol	Register Name
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1



Note 1: This flowchart only shows steps that are related to the USB Function.
Note 2: This routine is a subroutine called up in [Figure 2.3 Example of USB Device Interrupt Routine].
Note 3: Execution of STP instruction (Stop mode: oscillation STOP state) is performed in the main routine.
Refer to [Figure 2.5 USB Suspend Process Examples (MAIN)]
Note 4: Shaded area indicates User Process.

Figure 2.4 Example of USB Suspend Interrupt Routine

The following are detailed descriptions of the steps shown in [Figure 2.4 Example of USB Suspend Interrupt Routine].

(1) System clock setting
- Since the USB clock will stop, it is necessary to select the main clock f(XIN) with the system clock selection bit of CPU Mode Register (CPUM).

(2) USB Control Register (USBCON)
- Enable the USB ref. voltage circuit operation (VREFE = "1")*1, go to low-power consumption mode (VREFCON = "1"), and input the up-port differential.
- Change the setting to circuit operation disabled (USBDIFE = "0"). All other bits remain in the pre-suspend state.
- Although the USB reference circuit goes to the low-power consumption mode, TrON stays in the 3V state. The difference between low-power consumption mode and normal mode is the drive capacity.

(3) Disable PLL operation
- When Disable PLL circuit operation without changing the PLL operation mode (any setting other than "00") of PLLCON (PLL Control Register), the PLL circuit is stopped, the PLL circuit output clock (fVCO) is fixed to "L".

(4) User system settings when using BUS Power
- When the USB Hub detects an over-current or power ON/OFF event at a down-port device while in the suspended state, it must transmit a remote wake-up signal. Therefore, it enables an external interrupt that provides the remote wake-up condition. However, the remote wake-up signal can only be transmitted if enabled by the host.

The STP instruction (STOP mode: oscillation STOP state) is executed in the main routine after the USB suspend interrupt processing shown in Figure 2.4.

Examples of USB suspend processes, which are executed in the main routine, are shown in Figure 2.5.

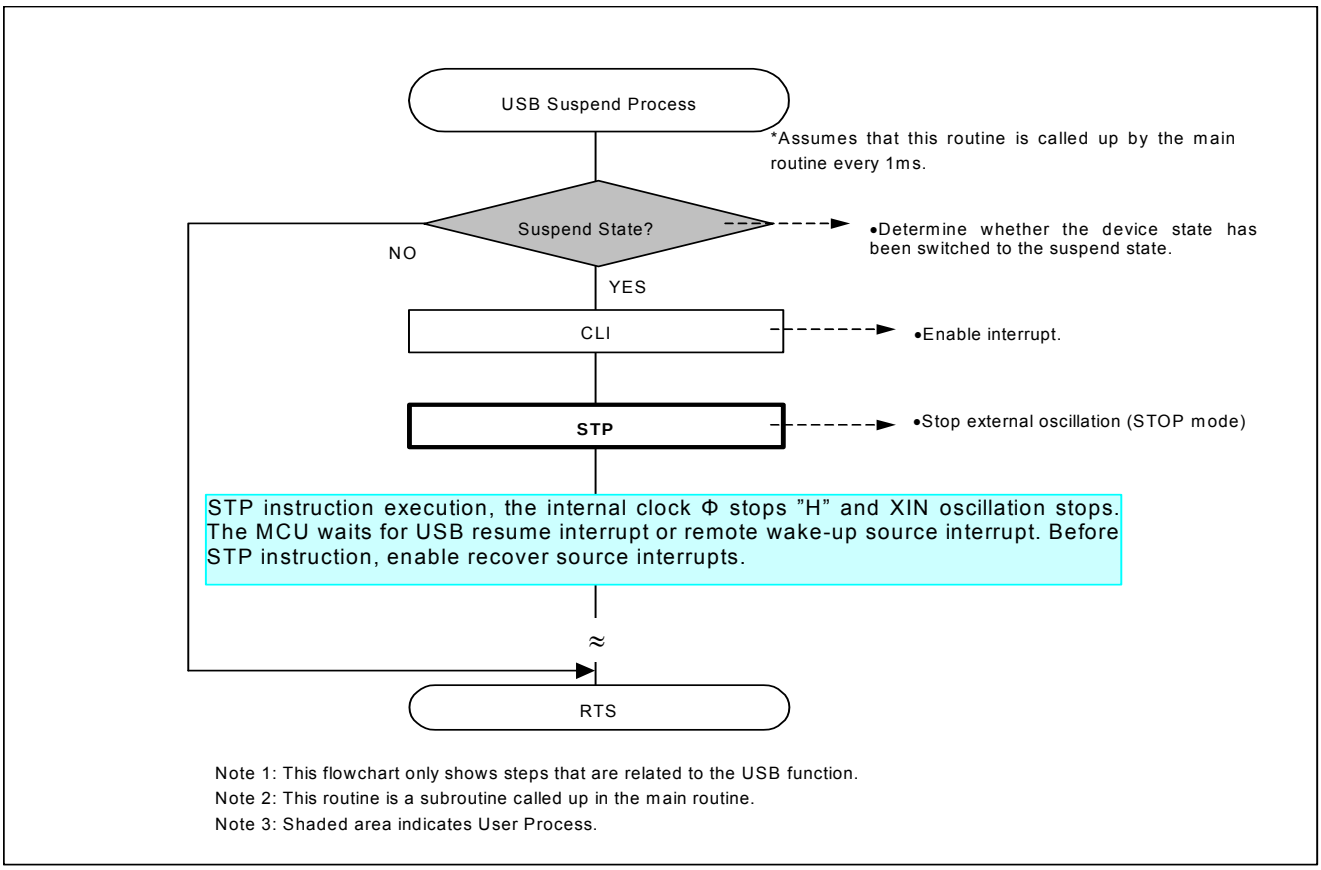


Figure 2.5 USB Suspend Processing Example (Main Routine)

*1 This setting is at Vcc=5V operation. At Vcc=3V operation, USB reference voltage enable bit (VREFE) must be set to disable ("0") (USB reference voltage control bit is ignored).

2.4.3. USB Resume Interrupt

USB resume interrupt occurs when the USB block detects a state change of the USB bus during the suspended state. Because the USB resume interrupt is generated due to a state change of the USB bus, the interrupt will occur even if no clock (USB clock) is supplied to USB block.
Registers related to the USB resume interrupt are listed in Table 2.7.

(1) Cause of USB resume detection

A USB resume is detected when the state of the USB bus changes: when detecting a change of “J” to “K” or “SE0”.

(2) Action when a USB resume is detected

When a USB resume is detected as described above, bit RSM (Resume Interrupt Bit) of USBIREQ (USB Interrupt Factor Register) is set to “1”.

(3) USB resume interrupt settings

Set bit RSME (Resume Interrupt Enable Bit) of USBICON (USB Interrupt Factor Enable Register) to “1” (enabled state).

(4) Conditions for accepting USB resume interrupt

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

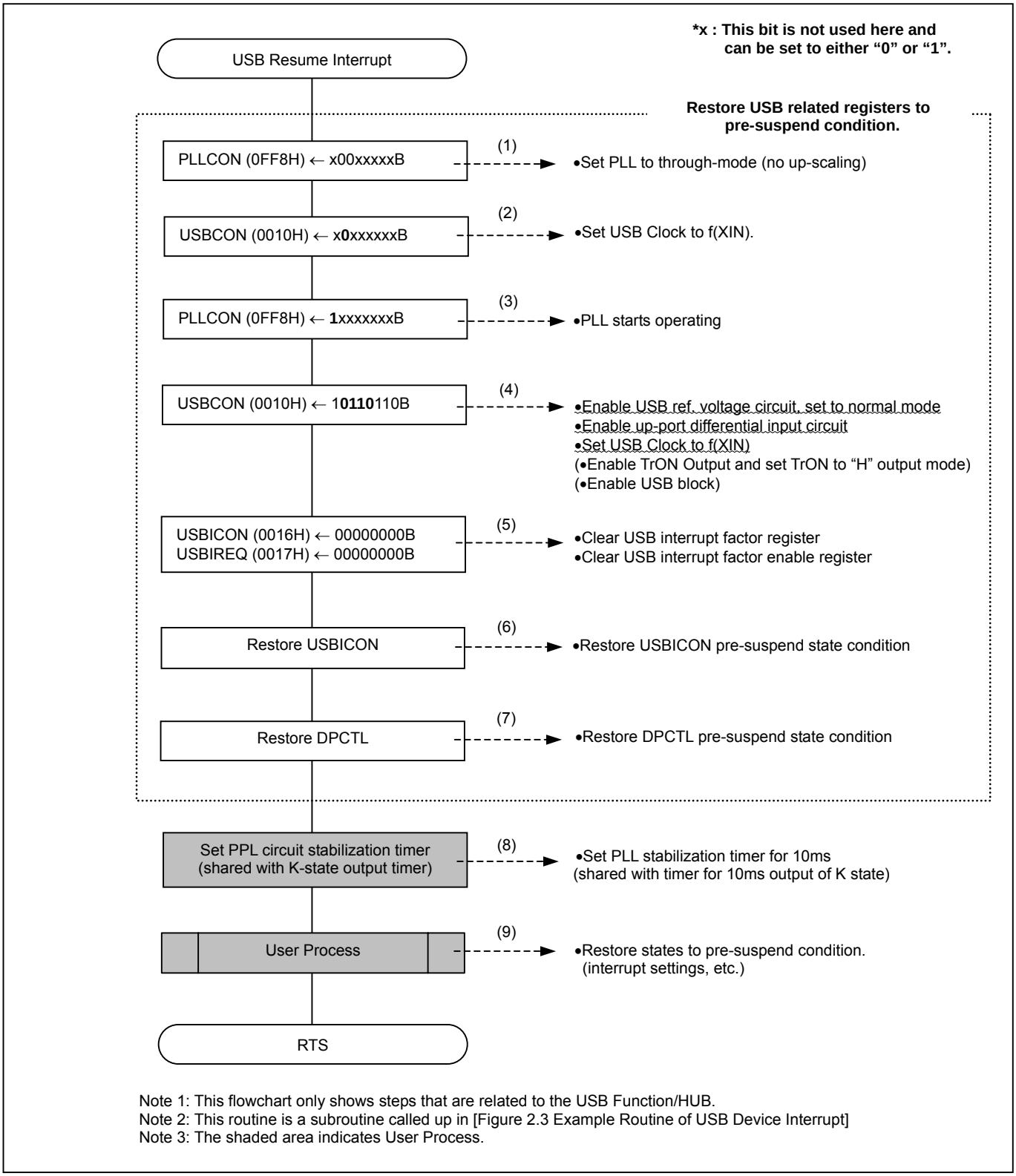
- I Flag (Interrupt Disable Flag) of the processor status register is “0” (interrupt enabled).
- The USB Function interrupt enable bit of ICON1 (Interrupt Control Register) is “1” (enabled state).
- The USB Function interrupt request bit of IREQ1 (Interrupt Request Register) is “1” (requested)
- Bit RSME of USBICON is “1” (enabled state)
- Bit RSM of USBIREQ is “1” (requested).

(5) USB resume interrupt processing

- An example of the processing routine that is executed when a USB suspend interrupt is accepted is shown in Figure 2.6.
- When a USB resume interrupt is accepted, USB Device Interrupt Request Flag (the root flag) is cleared as soon as the interrupt is accepted and does not need to be cleared by software during the interrupt routine. Please note that bit RSM of USBIREQ cannot be cleared by software. Setting 0 to bit RSME of USBICON will automatically clear bit RSM.

Table 2.6 USB Resume and Remote Wakeup Registers

Address	Symbol	Register Name
0010H	USBCON	USB Control Register
0016H	USBICON	USB Interrupt Factor Enable Register
0017H	USBIREQ	USB Interrupt Factor Register
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1



Note 1: This flowchart only shows steps that are related to the USB Function/HUB.
Note 2: This routine is a subroutine called up in [Figure 2.3 Example Routine of USB Device Interrupt]
Note 3: The shaded area indicates User Process.

Figure 2.6 Example of USB Resume Interrupt Routine

The following are detailed descriptions of the steps shown in [Figure 2.6 Example of USB Resume Interrupt Routine].

(1) Setting the PLL operation mode to through mode
- The PLL circuit output clock (fVCO) is fixed to "L" during the suspended state. The PLL circuit must be released in the following manner. (If the clock remains in the "L" level, the USB clock in Step (2) cannot be switched).
- When Set the PLL operation mode of PLLCON (PLL Control Register) to "no-up scaling" ("00" = through mode), in the through mode, external clock f(XIN) will be output through PLL even if PLL circuit operation is disabled.

(2) USB clock settings
- Set the USB clock to f(SIN) (UCLKCON = "0") in order to initiate PLL clock operation.

*After the PLL circuit is enabled, the output clock needs approximately about 1ms stabilize. Therefore, the USB clock selection bit of USBCON must be set to "0" and f(XIN) (external oscillation) must be used as the initial USB clock.

(3) Starting PLL circuit operation
- Set PLL circuit operation of PLLCON to "1". After the PLL circuit has stabilized (after 1ms) set the USB clock to PLL circuit output clock (fVCO) in the main routine.

(4) USB Control Register (USBCON) settings
- Return the MCU to the pre-suspend state settings: USB ref. voltage circuit operation enabled (VREF = "1")*[1], normal mode (VREFCON = "0"), and up-port differential input circuit operation enabled (USBDIFE = "1").

(5) Clearing USBIREQ, USBICON
- Clear USBIREQ (USB Interrupt Factor Register) and USBICON (USB Interrupt Factor Enable Register).
- Clear the interrupt request bit by setting bit RSME (Resume Interrupt Enable Bit) of USBICON to "0" (disabled), This will clear bit RSM (Resume Interrupt Factor Bit) of USBIREQ by hardware. This bit cannot be cleared by software.

(6) Recovering USBICON (USB Interrupt Factor Enable Register)
- Return the contents of USBICON to its pre-suspend state settings.

(7) Recovering DPCTL (Downstream port control register)
- Return the contents of DPCTL to its pre-suspend state settings.

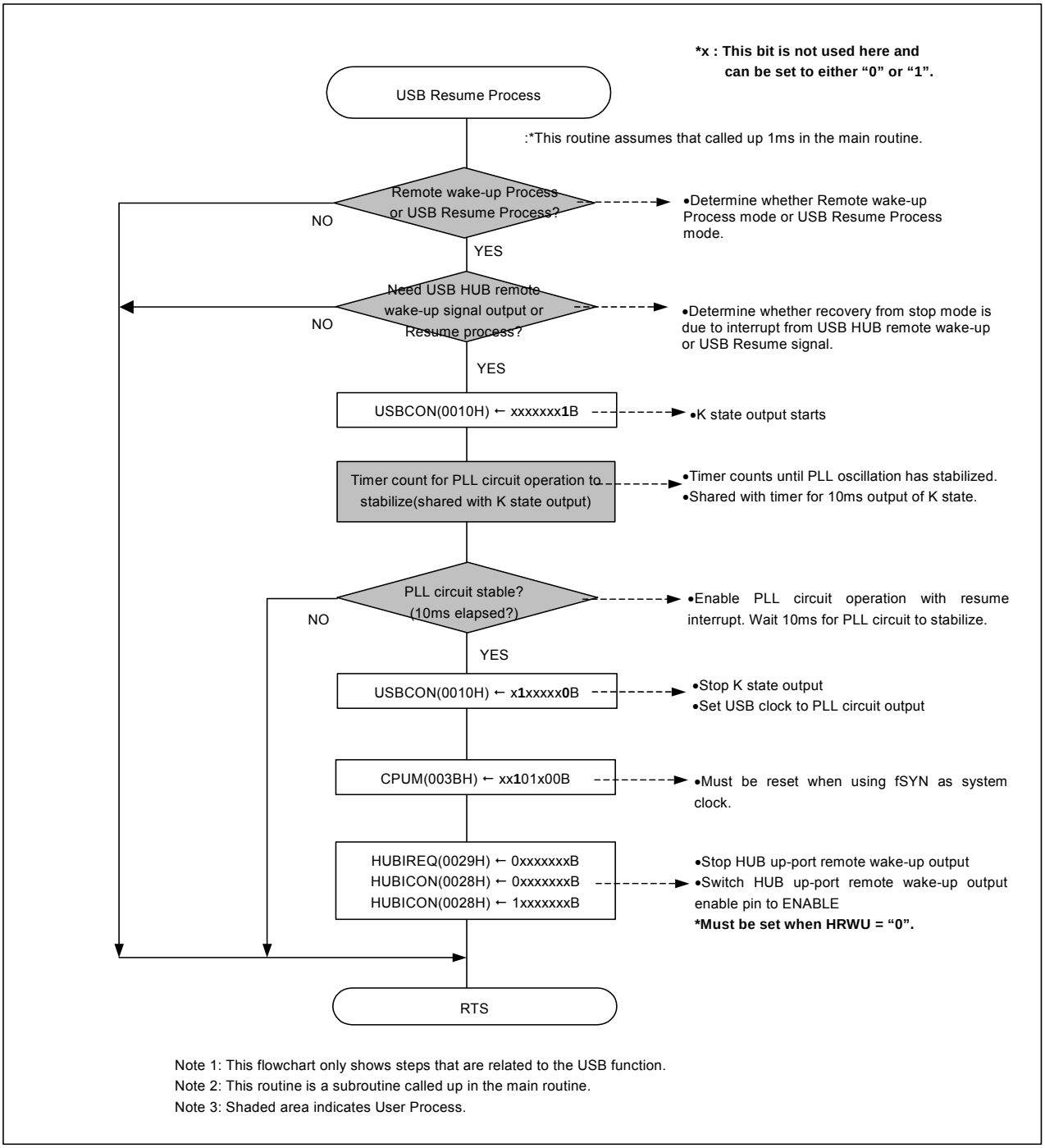
(8) Setting PLL circuit operation stabilization timer and remote wake-up signal "K" state output timer
- Set PLL circuit oscillation stabilization wait timer count value.
- For a case that a remote wake-up signal "K" state output is required, set the "K" state output time counter.

◆"38K2 group sample firmware" sets the timer count value above, which is also used as the main counter, to 12ms.

(9) Setting bus-powered user system
- Interrupts, etc., set by the user must be restored to their pre-suspend state.
- Restore peripheral circuit control and low-power consumption mode to normal mode.

Examples of USB resume processes, which are executed in the main routine, are shown in Figure 2.7. For more details on the resume interrupt and remote wake-up, refer to [Chapter 2.4.3.1 Remote Wake-up].

*[1] This setting is at Vcc=5V operation. At Vcc=3V operation, USB reference voltage enable bit (VREFE) must be set to disable ("0") (USB reference voltage control bit is ignored).



Note 1: This flowchart only shows steps that are related to the USB function.
Note 2: This routine is a subroutine called up in the main routine.
Note 3: Shaded area indicates User Process.

Figure 2.7 USB Resume Processing Example (Main Routine)

2.4.3.1. Remote Wake-up

When reactivating the MCU in the suspended state with a factor from the device side, send out a remote wake-up signal which will trigger a request for the host to reactivate operations.

The remote wake-up function in the 38K2 Group outputs K state to the up-port for 10ms, and then stops the K state output (10ms output time must be cleared by F/W.).

To reactivate the MCU in the suspended state using a device factor, after the remote wake-up interrupt is generated execute the device interrupt process at the same time the resume interrupt is generated. Then process the remote wake-up output. Figure 2.8 shows an example of the remote wake-up process routine.

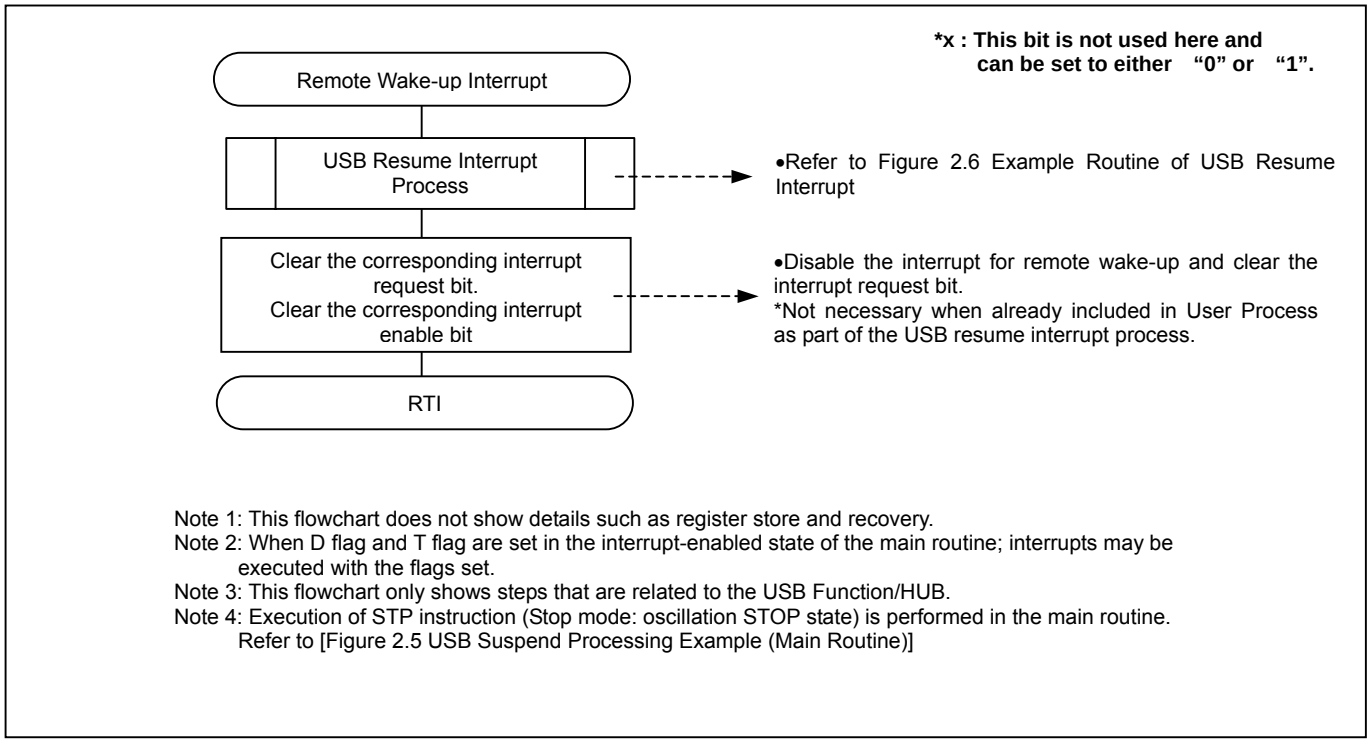


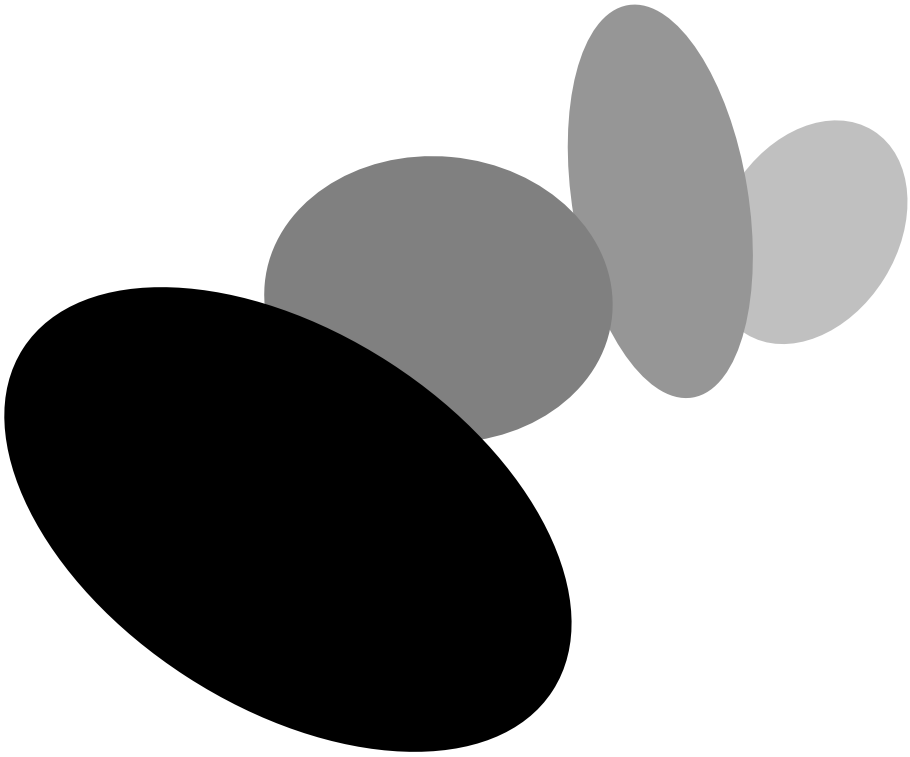
Figure 2.8 Example of Remote Wake-Up Interrupt Routine

●Main Factors for USB Hub Remote Wake-Up

(1) Detection of attach/detach of device at down-port
(2) Detection of over-current from down-port device
(3) Detection of USB Hub power ON/OFF

To execute a resume using one of the above factors when the USB Hub is in the suspended state, send a remote wake-up signal to trigger a resume request to the Host. The 38K2 will then set the USB control register (USBCON) remote wake-up bit (WKUP) to "1" and output the K state. During this output, the timer count is controlled by firmware.

When 2.5 μs or more of the K state to the down-port is detected, the resume request from the device connected to the down-port is recognized and the MCU outputs the K state signal to the up-port through the USB Hub Control Unit. During output, the timer count is controlled by firmware. When the down-port K state signal is detected, the Hub up-port remote wake-up output bit (HRWU of HUBIREQ register) is set by hardware. Note that this HRWU bit is not an interrupt factor. For more details, refer to Chapter 4.



Chapter 3
USB HUB

This Chapter explains each transfer type.

3.1. Endpoint Setup Register

The 38K2 USB Hub uses the following USB transmit/receive operations: Endpoint0 (indicated as Endpoint10 in the data sheet) is fixed for control transfers only and Endpoint1 (indicated as Endpoint11 in the data sheet) for interrupt transfers only.

Registers (endpoints) used for USB transfers by the USB Hub are shown in Figures 3.1 to 3.3. When using Endpoints 10 and 11 for USBINDEX (Endpoint Index Register), set the values indicated in Figure 3.1.

Table 3.1 Endpoint Index Register

Address	Symbol	Register Name
0018H	USBINDEX	Endpoint Index Register

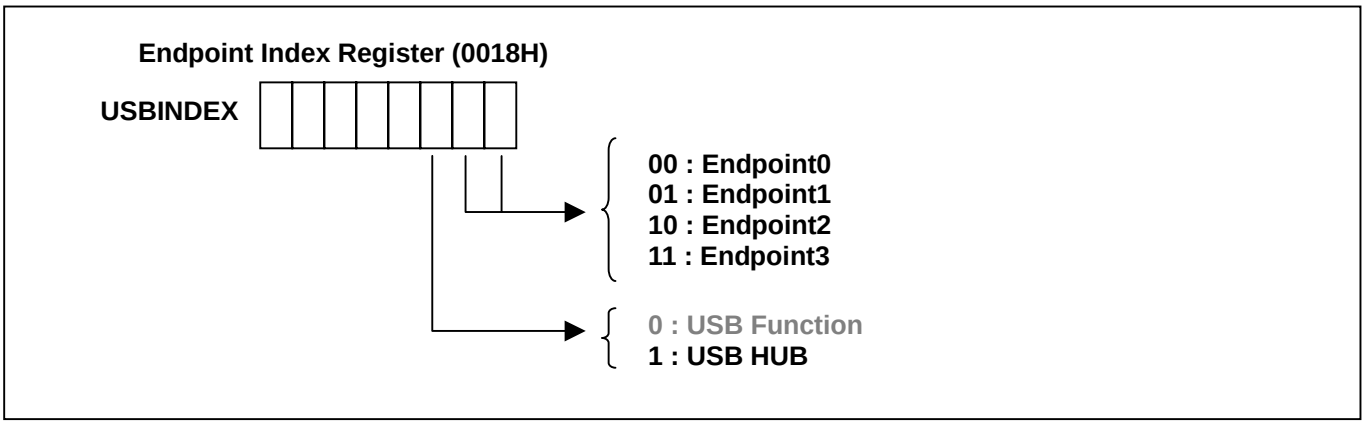


Figure 3.1 Endpoint Index Register Settings

Table 3.2 shows the registers required for setting Endpoint0 (control transfers).

The maximum packet size for control transfers is always set at 8 bytes. Set the top address of the Endpoint0 buffer in EP10BUF (EP10 Buffer Area Setup Register). The buffer consists of 8 bytes each of control command buffer and data buffer. For more details refer to [Chapter 3.2.2 Endpoint0 Buffer Area Setup].

When setting/referencing Endpoint0 registers, set “04” in USBINDEX before accessing the registers.

Table 3.2 USB Endpoint0 Registers

Address	Symbol	Register Name
0019H	EP10STG	EP10 Status Register
001AH	EP10CON1	EP10 Control Register 1
001BH	EP10CON2	EP10 Control Register 2
001CH	EP10CON3	EP10 Control Register 3
001DH	EP10REQ	EP10 Interrupt Factor Register
001EH	EP10BYT	EP10 Transfer/Receive Byte No. Register
001FH	-	-
0FECH	-	-
0FEDH	EP10BUF	EP10 Buffer Area Setup Register

Registers necessary for setting Endpoints 1 (IN interrupt transfers) is listed in Table 3.3.

Perform the transfer/receive settings with the EP0xCFG (EP0x Setup Register) in accordance with the descriptor settings of Endpoints 1 to 3. The buffer has 2 settings: single buffer mode and double buffer mode (alternating transfer of Data0 and Data1). Set the top address of Buffer 0 with EP0xBUF (EP0x Buffer Area Setup Register). Set the top address of Buffer 1 bit with BSIZ01 (Double Buffer Top Address Setup Bit of EP0xCFG) as the relative address. Refer to [Chapter 3.3.2 Endpoints 1 to 3 Buffer Area Settings] for more details.

Use EP0xCON1 (EP0x Control Register 1) to set responses for transfer/receive transmissions. Use EP0xCON2 and EP0xCON3 (EP0x Control Registers 2 and 3) to enable the buffers.

"x" represents either 1, 2, or 3. When setting/referencing an endpointx register, access the endpointx register after setting the endpoint value in USBINDEX (Endpoint Index Register).

Endpoint1 can only be set to the single buffer mode. Set the start address of Buffer 0 with EP11BUF (EP11 Buffer Area Setup Register). Refer to [Chapter 3.2.2 Endpoint0 Buffer Area Settings] for more details.

Use EP11CON1 (EP11 Control Register 1) to set responses for Endpoint1 transfers. Use EP11CON2 (EP11 Control Register 2) to enable Buffer 0.

When setting Endpoint1 registers, set USBINDEX (Endpoint Index Register) to "05H" before accessing the Endpoint1 registers described in Table 3.3.

Table 3.3 USB Endpoint1 Registers

Address	Symbol	Register Name
0019H	EP11CFG	EP11 Setup Register
001AH	EP11CON1	EP11 Control Register 1
001BH	EP11CON2	EP11 Control Register 2
001CH	-	-
001DH	EP11REQ	EP11 Interrupt Factor Register
001EH	EP11BYT0	EP11 Transfer/Receive Byte No. Register 0
001FH	-	-
0FECH	-	-
0FEDH	EP11BUF	EP11 Buffer Area Setup Register

3.2. Endpoint0

Endpoint0 is used to perform control transfers. Data is transferred or received in response to a device request and USB HUB class request from the host.

Endpoint0 has a control command buffer for the SETUP stage and a data buffer for the DATA stage. Refer to [Chapter 3.2.2 Endpoint0 Buffer Area] for more details.

A USB HUB/Endpoint0 interrupt (5 factors) is generated during Endpoint0 control transfers. Refer to [Chapter 3.2.3 Endpoint0 Interrupts] for more details.

3.2.1. Control Transfer Formats

The control transfer is a bi-directional transfer mainly used with setup commands. The transfer consists of a least two transaction-stages (SETUP, STATUS). A control transfer may also include a DATA stage between the SETUP and STATUS stages.

In USB transmissions, Endpoint0 has a response mechanism for various requests that is sent from the host through control transfers.

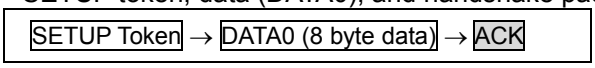
A control transfer consists of SETUP, DATA and STATUS stages. Data transfer and receive is performed in the DATA stage based on a SETUP stage request sent from the host. The control transfer ends when the STATUS stage is completed.

♦Control Transfer Transactions

The control transfer consists of the following three transaction types*1. The shaded blocks indicate data packets transmitted by a device.

(1) SETUP Stage:

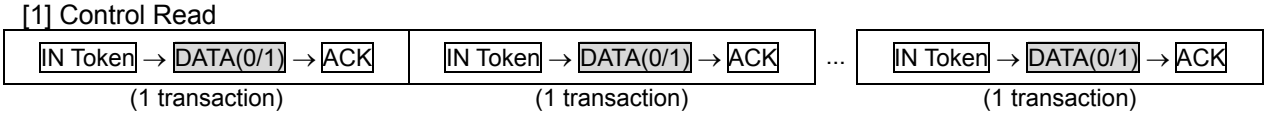
SETUP token, data (DATA0), and handshake packets.



(2) DATA Stage:

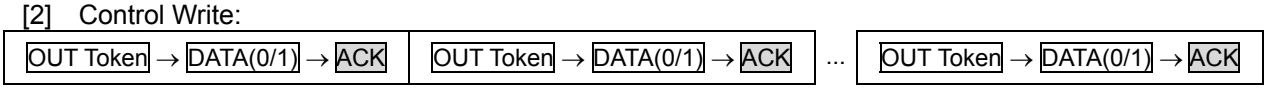
As in bulk transfers, the DATA transaction is repeated as many times as necessary.

[1] Control Read



(1 transaction) (1 transaction) (1 transaction)

[2] Control Write:

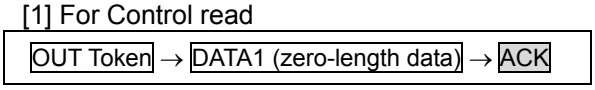


(1 transaction) (1 transaction) (1 transaction)

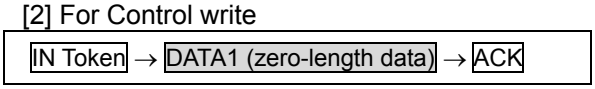
(3) STATUS Stage:

This transaction consists of an opposite-of-data-direction token packet and a zero-length DATA packet (DATA1).

[1] For Control read



[2] For Control write



*1 (1),(2) and (3) is an example in normal transfer and Handshake omits.

♦Control Transfer Sequences

The following three sequences are used for control transfers*1.

(1) Control read transfer:

SETUP(DATA0)	IN(DATA1 IN(DATA0) --- IN(DATA1) ---	OUT(DATA1)
SETUP Stage	DATA Stage	STATUS Stage

(2) Control write transfer:

SETUP(DATA0)	OUT(DATA1) OUT(DATA0) --- OUT(DATA1) ---	IN(DATA1)
SETUP Stage	DATA Stage	STATUS Stage

(3) Control write/no data transfer:

SETUP(DATA0)	IN(DATA1)
SETUP Stage	STATUS Stage

3.2.2. Endpoint0 Buffer Area Setup

The user needs to assign a buffer area for each endpoint in order to perform data transfer/receive.

Endpoint0 (control transfer) has a SETUP token control command buffer and an IN/OUT token data buffer. Each buffer (8 bytes) is stored in the RAM according to the values set by EP10BUF (EP10 Buffer Area Setup Register). Set the control command buffer top address in EP10BUF, as shown in Figure 3.2., and store the area in the RAM. An example of the setup is shown in Figure 3.3.

The top address in the control command buffer is 20H times the value set in EP10BUF. The top address in the data buffer is 20H times the value set in EP10BUF plus 8H.

See Figure 3.4 for an image of buffer access.

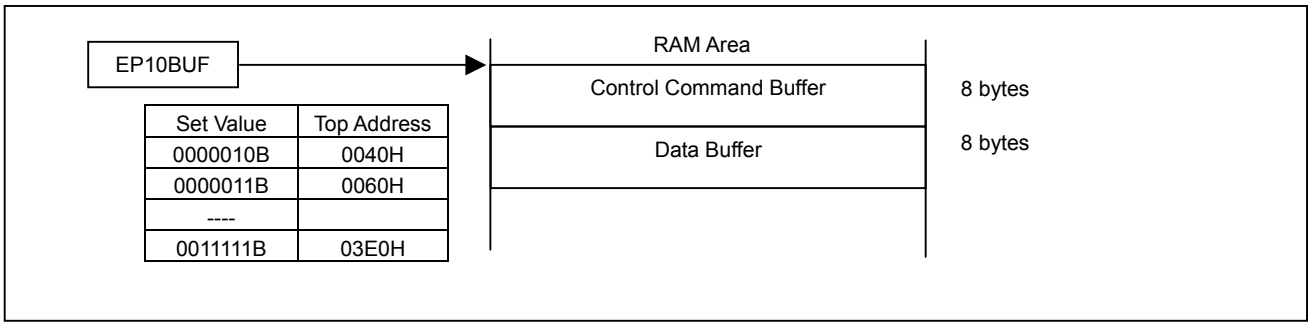


Figure 3.2 Endpoint0 Buffer

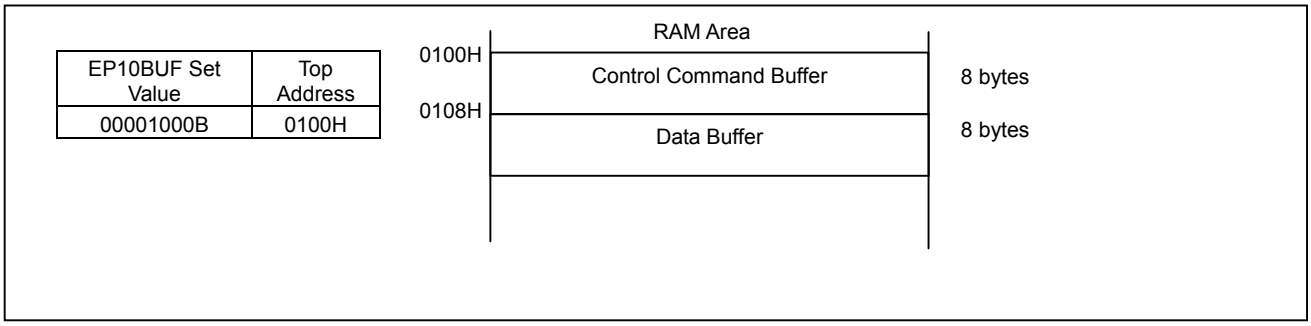


Figure 3.3 Example of Endpoint0 Buffer Setup

*1 (1),(2) and (3) is an example in normal transfer.

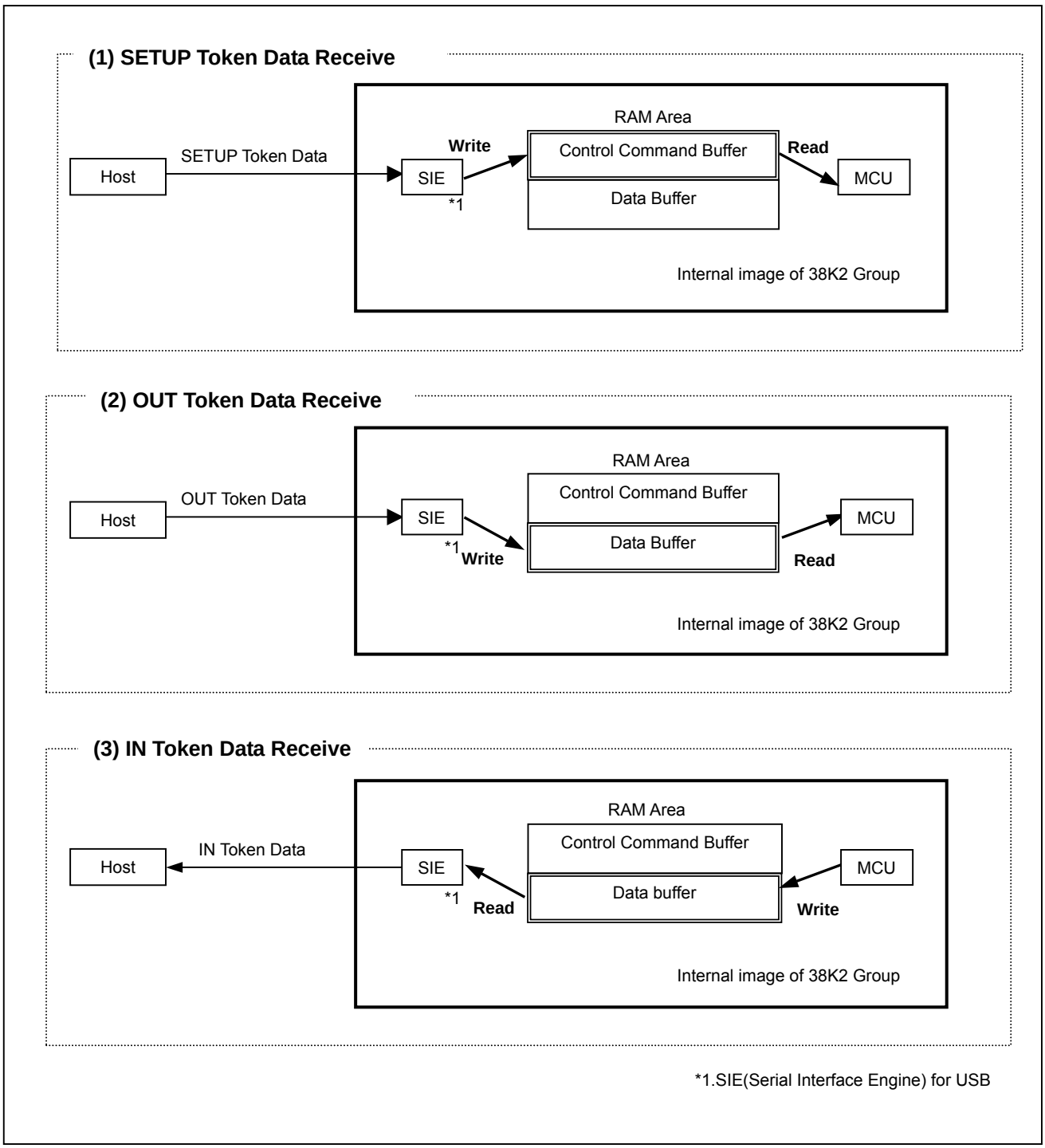


Figure 3.4 Endpoint0 Buffer Area Access

3.2.3. Endpoint0 Interrupts

Endpoint0 interrupts are used to control data flow during control transfers. Endpoint0 Interrupt Factor Register is shown in Figure 3.5.

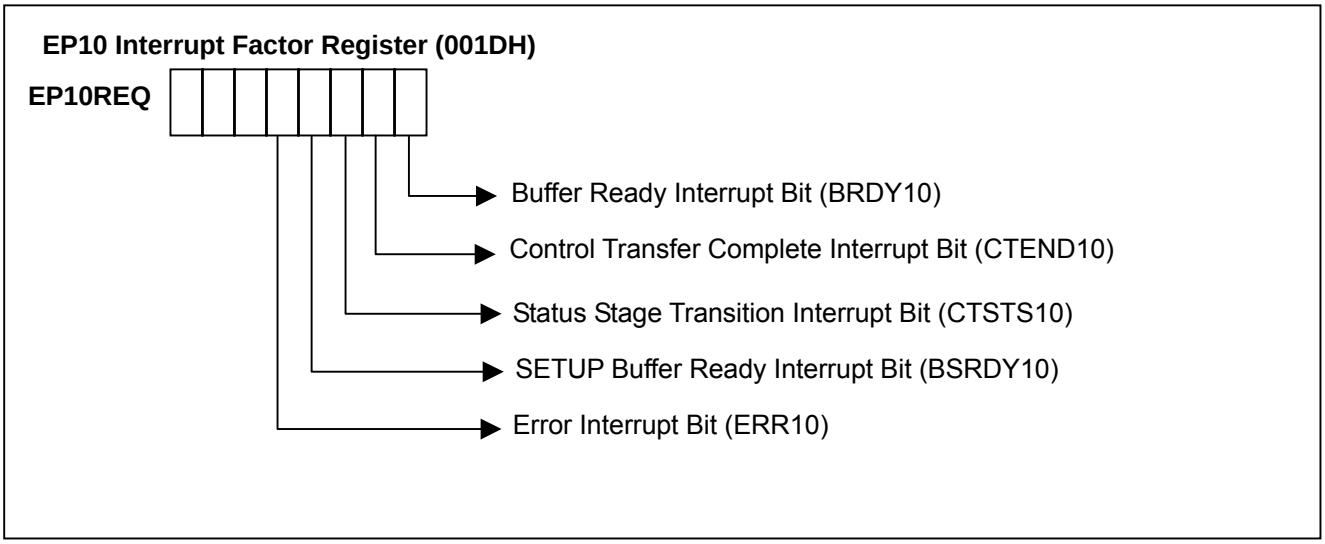


Figure 3.5 EP10 Interrupt Factor Register (EP10REQ)

(1) SETUP Buffer Ready Interrupt Bit (BRDY10)
This bit goes to “1” (H/W set) when the SETUP token buffer is in the ready state (read enabled). This bit must be cleared by F/W when an interrupt occurs. (this bit not cleared to “0” automatically by H/W.)

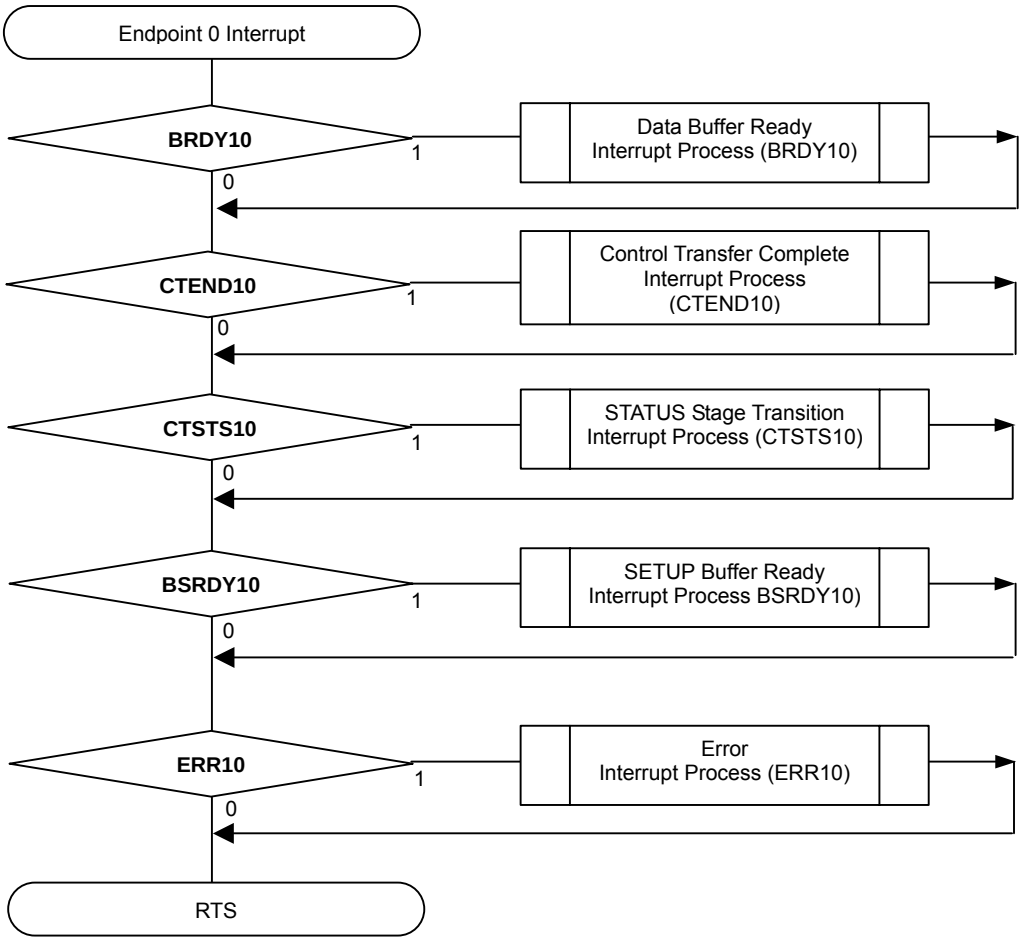
(2) Buffer Ready Interrupt Bit (BRDY10)
This bit goes to “1” (H/W set) when the data buffer is in the ready state (read/write enabled). This bit must be cleared by F/W when an interrupt occurs. (this bit not cleared to “0” automatically by H/W.)

(3) Control Transfer Complete Interrupt Bit (CTEND10)
This bit goes to “1” (H/W set) when the control transfer is completed (NULL/ACK response in STATUS stage). This bit must be cleared by F/W when an interrupt occurs. (this bit not cleared to “0” automatically by H/W.)

(4) Status Stage Transition Interrupt Bit (CTSTS10)
This bit goes to “1” (H/W set) when CTENDE10 (Control Transfer Complete Enable Bit) of EP00CON3 (EP10 Control Register 3) is “0” (disabled).
Interrupt generation conditions:
•When an IN token is received in response to DATA stage (OUT) during a control transfer.
•When an OUT token is received in response to a DATA Stage (IN) during a control transfer.
*An interrupt will not be generated during a no-data transfer.
This bit must be cleared by F/W when an interrupt occurs. (this bit not cleared to “0” automatically by H/W.)

(5) Error Interrupt Bit (ERR10)
This bit goes to “1” (H/W set) when a control transfer error occurs.
This bit must be cleared by F/W when an interrupt occurs. (this bit not cleared to “0” automatically by H/W.) However, it is automatically cleared by H/W when a SETUP token is received.

Figure 3.6 shows Endpoint0 interrupt processing. Determine the interrupt factor at each stage and then perform receive/transfer setup and buffer read/write. In the SETUP stage, each request is processed after the data is read out.



Note 1: Always judge the interrupt factor bit in the order indicated above.
Note 2: This routine is a subroutine called in [Figurer 2.3 USB Device Interrupt Routine Example].

Figure 3.6 Example of Endpoint0 Interrupt Processing

3.2.4. Control Transfer Data Receive (Control Write Transfer)

When a request is received for a control write transfer, a SETUP token is received in the SETUP stage, followed by an OUT token in the DATA stage, ending with an IN token in the STATUS stage. An example of data receive (normal transfer) in a control transfer is shown in Figure 3.7. The following is a detailed explanation of the figure.

(1) Receiving a SETUP token in the SETUP stage
When a BSRDY10 interrupt is generated, read the data from the control command buffer. After the data is read, set CTENDE10 (Control Transfer Complete Enable Bit) of EP10CON3 (EP10 Control Register 3) to "1" and BVAL10 (Buffer Enable Bit) of EP10CON2 (EP10 Control Register 2) to "1"*[1]. BVAL10 must be enabled by setting PID10 (Response PID Bit) of EP10CON1 (EP10 Control Register 1) to "01" (auto response).

(2) Receiving an OUT token in the DATA stage
When a BRDY10 interrupt is generated, read the data from the control command buffer. After the data is read, set BVAL10 of EP00CON2 to "1". (The BVAL10 bit sends a normal response (an ACK in response to the OUT token) and is automatically cleared by H/W..

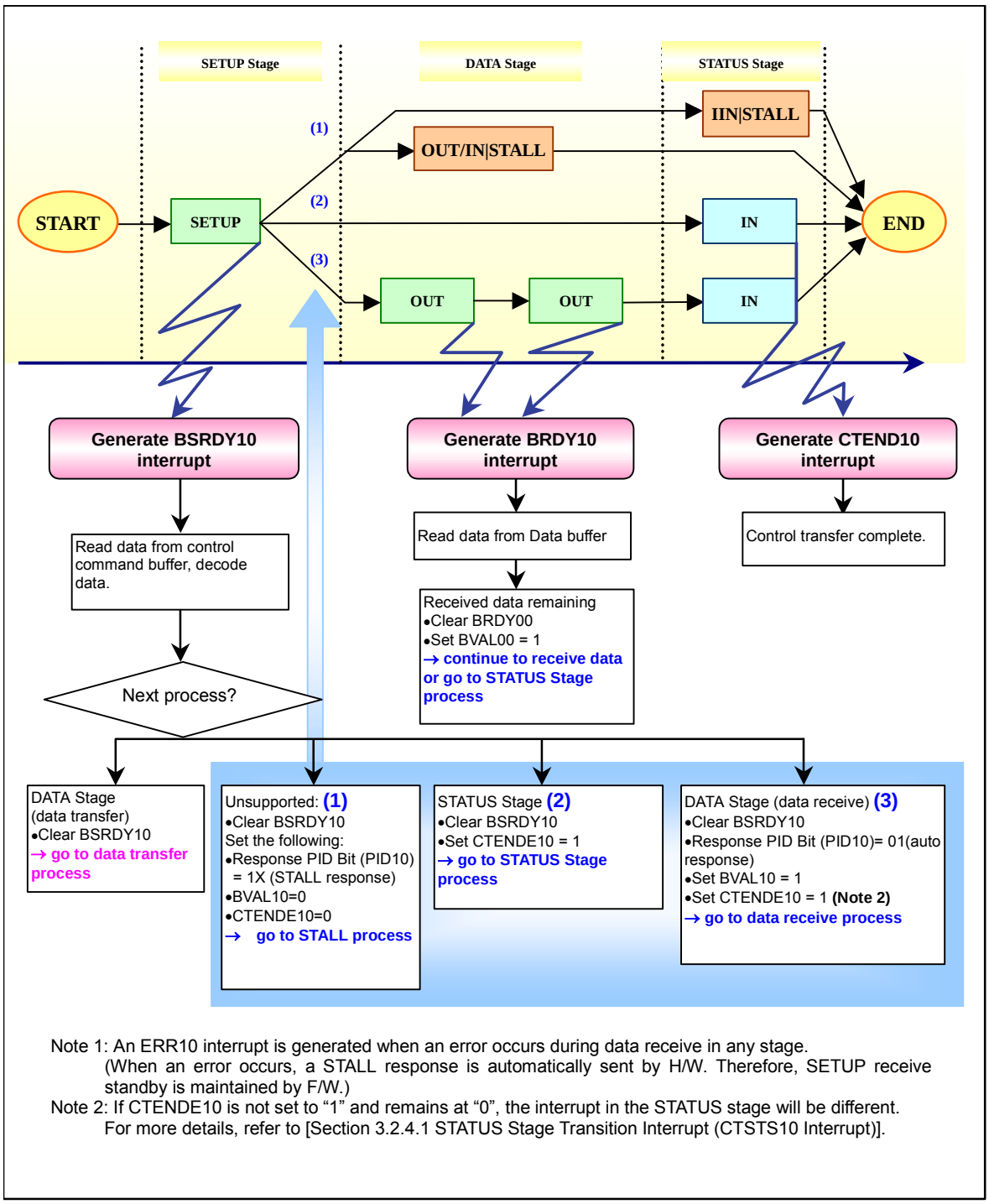
(3) Receiving an IN token in the STATUS stage
The CTEND10 interrupt is generated. At this time, the CTENDE10 (Control Transfer Enable Bit) of EP10CON3 sends a response (sends an ACK in response to the IN token) and the bit is cleared automatically by H/W.

♦ Error during data receive from Host (for control write/read transfers)
An ERR10 interrupt is generated. At this time, PID10 of EP10CON1 is automatically set to "1x" (STALL response) by H/W. The STALL response is continuously sent in response to IN/OUT tokens in the DATA and STATUS stages. However, when the next SETUP stage starts, PID10 is automatically set to "00" (NAK response) by H/W, and the STALL is cancelled. ERR10 (Error Interrupt Bit) is cleared automatically by H/W when the SETUP token is received.

♦ Receiving an unsupported request (for control write/read transfers)
SetPID10 of EP10CON1 to "1x". The STALL will continue to be sent in response to IN/OUT tokens in the DATA and STATUS stages. However, when the next SETUP stage starts, PID10 is automatically set to "00" (NAK response) by H/W, and the STALL is cancelled. ER10 is cleared automatically by H/W when the SETUP token is received.

♦ Receiving a SETUP token (for control write/read transfers)
The following bits are automatically set (cleared to "0") by H/W when a SETUP token is received.
- PID10 (Response PID Bit) of EP10CON1 = "00"
  *An ACK is automatically sent in response to the SETUP token.
- BVAL10 (Buffer Enable Bit) of EP10CON2 = "0"
- CTENDE10 (Control Transfer Complete Enable Bit) of EP10CON3 = "0"
- ERR10 (Error Interrupt Bit) of EP10REQ (EP10 Interrupt Factor Register) = "0"

*[1] Sample procedures (1) and (2) are included in RENESAS's "38K2 Group USB Sample Firmware."



Note 1: An ERR10 interrupt is generated when an error occurs during data receive in any stage.
(When an error occurs, a STALL response is automatically sent by H/W. Therefore, SETUP receive standby is maintained by F/W.)
Note 2: If CTENDE10 is not set to "1" and remains at "0", the interrupt in the STATUS stage will be different.
For more details, refer to [Section 3.2.4.1 STATUS Stage Transition Interrupt (CTSTS10 Interrupt)].

Figure 3.7 Example of Data Receive in Control Transfer (normal transfer)

3.2.4.1. STATUS Stage Transition Interrupt (CTSTS10 Interrupt)

The CTSTS10 interrupt is generated one time when the STATUS stage transition from Control stage is detected. By setting CTENDE10 (Control Transfer Complete Enable Bit) of EP10CON3 to “0”, the CTSTS00 (STATUS Stage Transition Interrupt Bit) interrupt will be generated.

To prevent the Control transfer to be completed before the device completes the processing for the setup command, this interrupt triggers the device to send a NAK response to the device.

Figure 3.8 shows an example of a control transfer receive when the CTSTS00 interrupt is generated. The following is a detailed explanation of the figure.

(1) Receiving a SETUP token in the SETUP stage

When a BSRDY10 interrupt is generated, read the data from the control command buffer. Set BVAL10 (Buffer Enable Bit) of EP10CON2 (EP10 Control Register 2) to “1”. BVAL10 must be enabled by setting PID10 (Response PID Bit) of EP10CON1 (EP10 Control Register 1) to “01” (auto response). In addition, BSRDY10 (SETUP Buffer Ready Interrupt Bit) must be cleared by F/W. By not setting CTENDE10 (Control Transfer Complete Enable Bit) of EP10CON3 to “1”, the CTSTS10 (STATUS Stage Transition Interrupt Bit) interrupt will be generated.

(2) Receiving an OUT token in the DATA stage

When a BRDY10 interrupt is generated, read the data from the control command buffer. After the data is read, set BVAL10 of EP10CON2 to “1”. (The BVAL10 bit sends a normal response (an ACK in response to the OUT token) and is automatically cleared by H/W. However, BRDY10 (Buffer Ready Interrupt Bit) must be cleared by F/W.

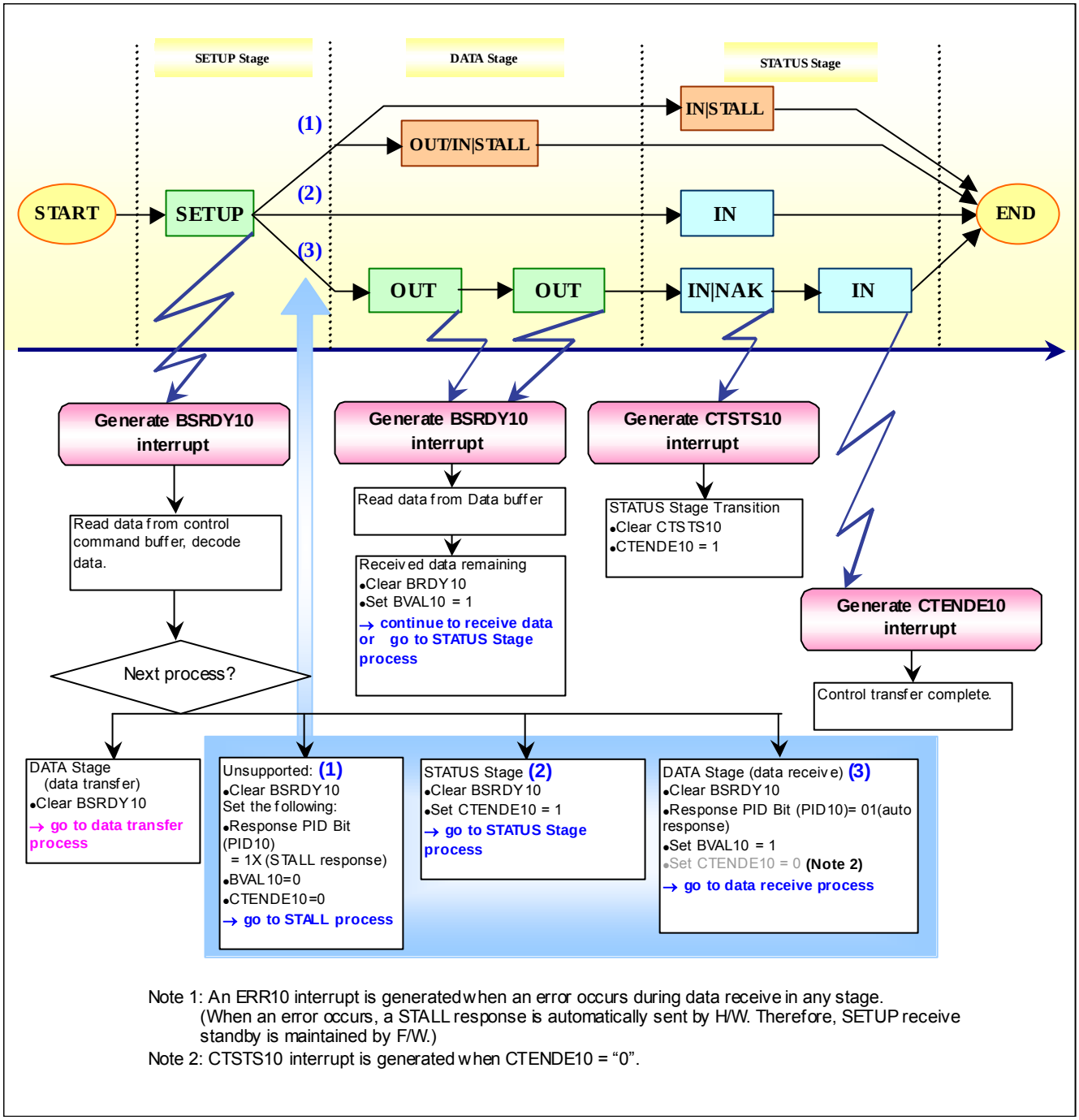
(3) Receiving an IN token in the STATUS Stage (send NAK)

The CTSTS10 interrupt is generated. Interrupt generation conditions require that the CTENDE10 is “0” and a NAK is sent in response to the IN token (sending a NAK indicates that the STATUS stage is not yet complete). However, even if these conditions continue, the CTSTS10 interrupt will only be generated in response to the first IN token. At this time, if the CTENDE10 (Control Transfer Enable Bit) of EP10CON3 is set to “1”, the CTEND10 interrupt will be generated at the next IN token. As mentioned above, the CTSTS10 interrupt is only generated once per data transfer sequence. To confirm the completion of a control transfer, a CTEND10 interrupt should be generated.

In addition, CTSTS10 (STATUS Stage Transition Interrupt Bit) must be cleared by F/W.

(4) Receiving an IN token in the STATUS stage (send ACK)

The CTEND10 interrupt is generated. At this time, the CTENDE10 (Control Transfer Enable Bit) of EP10CON3 sends a normal response (sends an ACK in response to the IN token) and the bit is cleared to “0” automatically by H/W.



Note 1: An ERR10 interrupt is generated when an error occurs during data receive in any stage.
(When an error occurs, a STALL response is automatically sent by H/W. Therefore, SETUP receive standby is maintained by F/W.)
Note 2: CTSTS10 interrupt is generated when CTENDE10 = "0".

Figure 3.8 Example of Data Receive in Control Transfer (normal transfer/CTSTS00 interrupt)

3.2.5. Control Transfer Data Send (Control Read Transfer)

When a request is received for a control read transfer, a SETUP token is received in the SETUP stage, followed by an IN token in the DATA stage, ending with an OUT token of the STATUS stage. An example of data receive in a control transfer is show in Figure 3.9. The following is a detailed explanation of the figure.

(1) Receiving a SETUP token in the SETUP stage

When a BSRDY10 interrupt is generated, read the data from the control command buffer. After the data is read, set CTENDE10 (Control Transfer Complete Enable Bit) of EP10CON3 (EP10 Control Register 3) to “1” and BVAL10 (Buffer Enable Bit) of EP10CON2 (EP10 Control Register 2) to “1”. BVAL10 must be enabled by setting PID10 (Response PID Bit) of EP10CON1 (EP10 Control Register 1) to “01” (auto response).

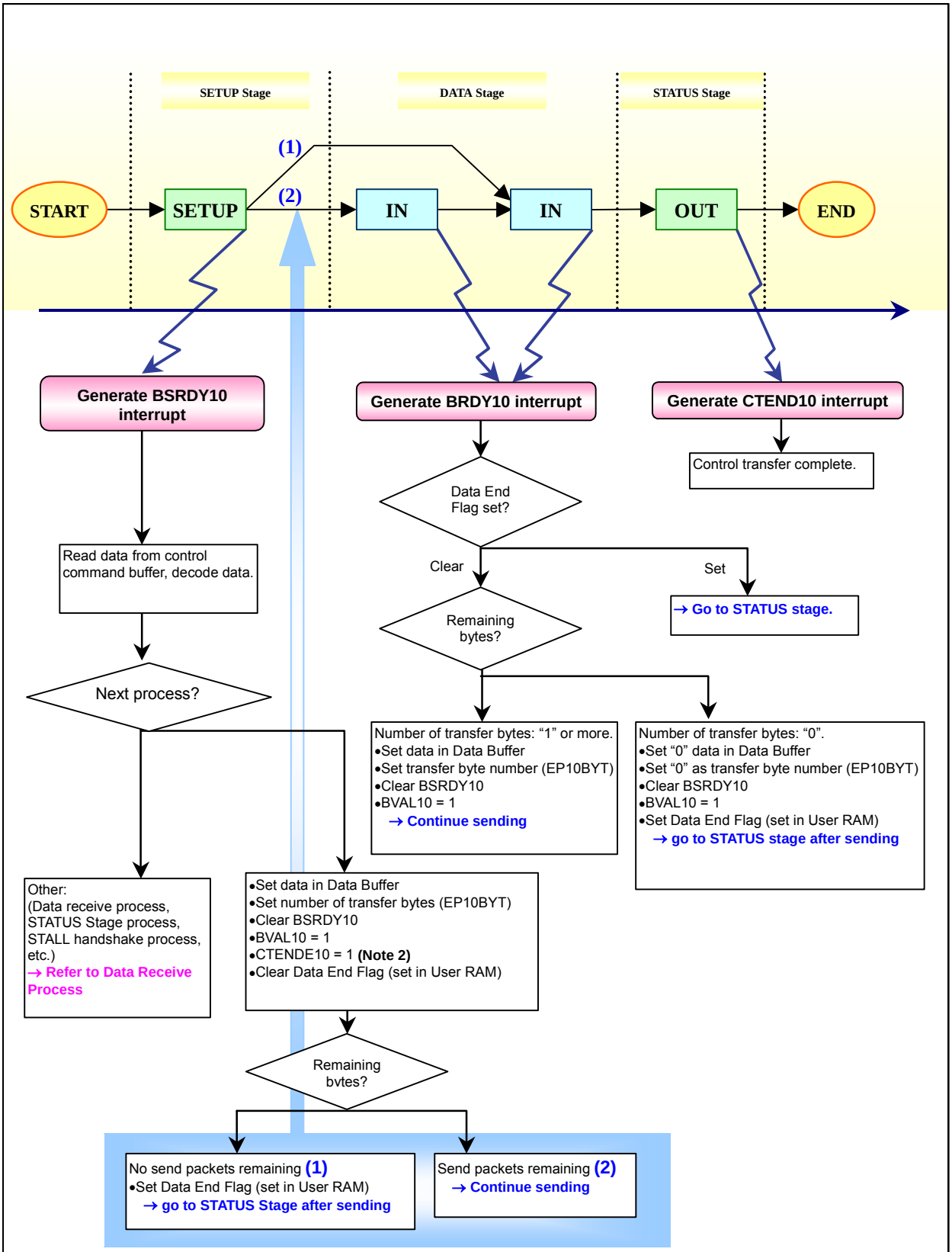
(2) Receiving an IN token in the DATA stage

When a BRDY10 interrupt is generated, write the data to the control command buffer. After the data is written, set BVAL10 of EP10CON2 to “1”. (The BVAL10 bit performs normal transmissions (receives an ACK in response to the OUT token) and is therefore automatically cleared by H/W.

Because there is no more data to be written when the BRDY10 interrupt of the last IN token is received, the BVAL10 does not need to be set to “1”.

(3) Receiving an OUT token in the STATUS stage

The CTEND10 interrupt is generated. At this time, the CTENDE10 (Control Transfer Enable Bit) of EP10CON3 performs normal transmissions (sends an ACK in response to the OUT token) and the bit is cleared to “0” automatically by H/W.



Note 1: An ERR10 interrupt is generated when an error occurs during data receive in any stage.
(When an error occurs, a STALL response is automatically sent by H/W. Therefore, SETUP receive standby is maintained by F/W.)
Note 2: If CTENDE10 is not set to "1" and remains at "0", the interrupt in the STATUS stage will be different.
For more details, refer to [Section 3.2.5.1 STATUS Stage Transition Interrupt (CTSTS10 Interrupt)].

Figure 3.9 Example of Data Send in Control Transfer (normal transfer)

3.2.5.1. STATUS Stage Transition Interrupt (CTSTS10 Interrupt)

For more details of CTSTS10 interrupt, refer to [Section 3.2.4.1 STATUS Stage Transition Interrupt (CTSTS10 Interrupt)]. Figure 3.10 shows an example of a control transfer data send when the CTSTS10 interrupt is generated. The following is a detailed explanation of the figure.

(1) Receiving a SETUP token in the SETUP stage

When a BSRDY10 interrupt is generated, read the data from the control command buffer. Set BVAL10 (Buffer Enable Bit) of EP10CON2 (EP10 Control Register 2) to “1”. BVAL10 must be enabled by setting PID10 (Response PID Bit) of EP10CON1 (EP10 Control Register 1) to “01” (auto response). In addition, BSRDY10 (SETUP Buffer Ready Interrupt Bit) must be cleared. By not setting CTENDE10 (Control Transfer Complete Enable Bit) of EP10CON3 (EP10 Control Register 3) to “1”, the CTSTS10 interrupt will be generated.

(2) Receiving an IN token in the DATA stage

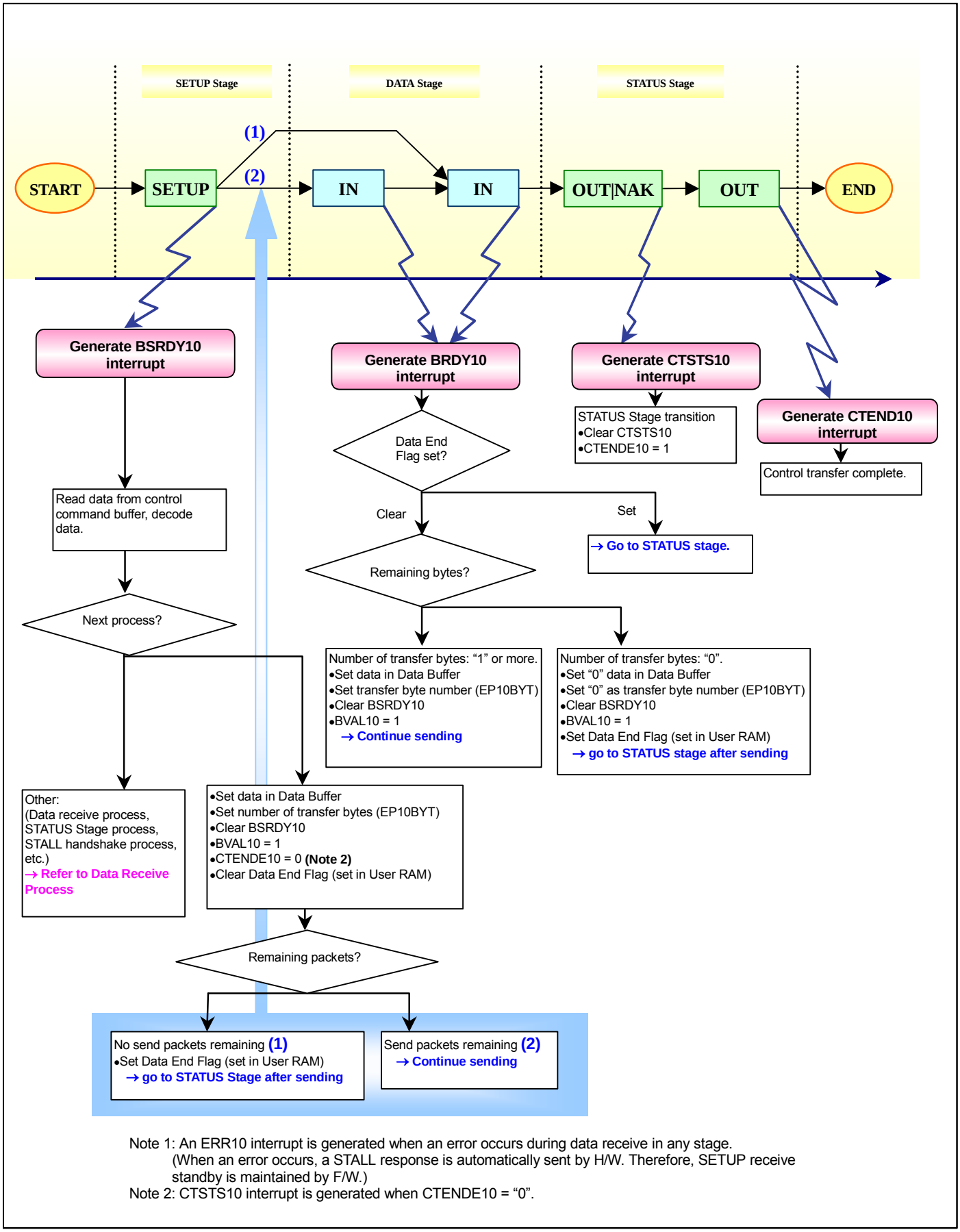
When a BRDY10 interrupt is generated, write the data to the data buffer. After the data is written, set BVAL10 of EP10CON2 to “1”. (The BVAL10 bit sends a normal response (an ACK in received in response to the IN token) and is therefore automatically cleared by H/W.

(3) Receiving an OUT token in the STATUS stage (send NAK)

The CTSTS10 interrupt is generated. Interrupt generation conditions require that the CTENDE10 is “0” and a NAK is sent in response to the OUT token (the NAK handshake indicates that the STATUS stage is not yet complete). However, even if these conditions continue to be met, the CTSTS10 interrupt will only be generated in response to the first IN token. At this time, if the CTENDE10 of EP10CON3 is set to “1”, a CTEND10 interrupt will be generated at the next IN token. As mentioned above, the CTSTS10 interrupt is only generated once per data transfer sequence. To confirm the completion of a control transfer, a CTEND10 interrupt should be generated.

(4) Receiving an OUT token in the STATUS stage (send ACK)

The CTEND10 interrupt is generated. At this time, the CTENDE10 of EP10CON3 performs normal transmissions (receives ACK in response to the OUT token) and the bit is cleared to “0” automatically by H/W.



Note 1: An ERR10 interrupt is generated when an error occurs during data receive in any stage.
(When an error occurs, a STALL response is automatically sent by H/W. Therefore, SETUP receive standby is maintained by F/W.)
Note 2: CTSTS10 interrupt is generated when CTENDE10 = "0".

Figure 3.10 Example of Data Send in Control Transfer (normal transfer/CTSTS10 interrupt)

3.3. Endpoints 1

Endpoint1 is for Interrupt transfers (IN only) and sends USB Hub status data to the Host.
The Endpoint1 data buffer must be stored in the RAM area. Set the data buffer start address with EP11BUF (EP11 Buffer Area Setup Register). For more details, refer to [3.2.2 Endpoint 1 Buffer Area Setup].
A USB Hub/Endpoint1 interrupt (1 factor) will occur during an Endpoint11 transfer. For more details, refer to [Chapter 3.3.3 Endpoint1 Interrupts]

3.3.1. Data Transfer Format

The USB Hub transfer uses a half-duplex transfer and consists of 3 packets (token packet, data packet, handshake packet). The Interrupt transfer detects transfer errors and guarantees the transfer rate.
USB Hub is only equipped with Endpoint1 and can only use Interrupt IN transfers.

♦ Data Transfer Transactions
The following types of transactions classify data transfer*1. The shaded blocks indicate a packet transmitted by a device.

(1) Interrupt IN:

This transaction consists of an IN token packet, data packets (DATA0/1), and a handshake packet.

♦ Transactions according to Data Transfer Type
The following types of transactions are used to perform transfer data*2.

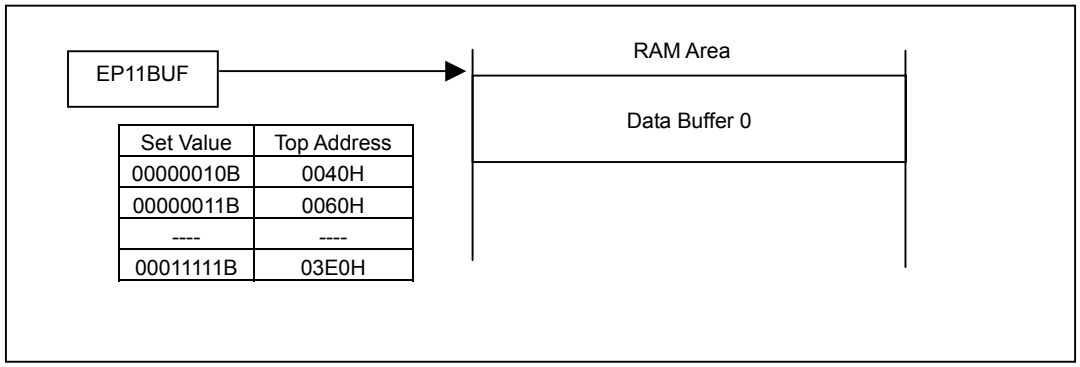
(1) Interrupt IN Transfers:

:

*1 (1) is an example in normal transfer.
*2 (1) is an example in normal transfer and Handshake omits.

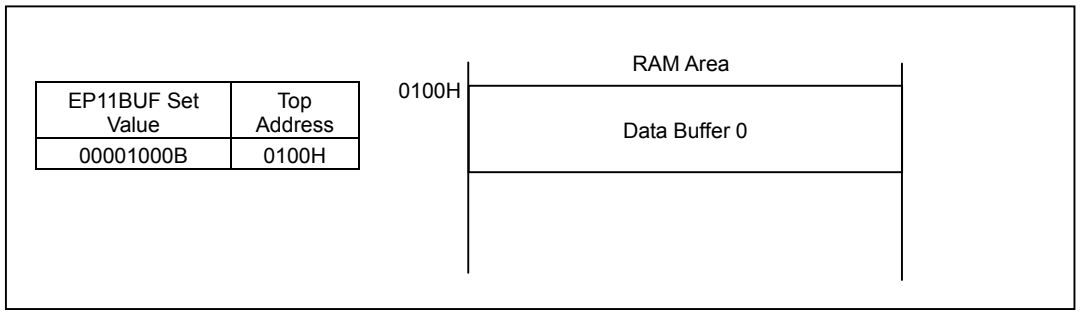
3.3.2. Buffer Area Setup for Endpoints 1

The data buffer size of Endpoint1 is MAX 1 byte.
Set the start address for the data buffer in EP11BUF (EP11 Buffer Area Setup Register), as shown in Figure 3.11. The data buffer area must be assigned in the RAM (see example in Fig. 3.12).
The start address for the data buffer is 20H times the value set in EP11 BUF.
Figure 3.13 shows an image of a data buffer access.



Set Value	Top Address
00000010B	0040H
00000011B	0060H
----	----
00011111B	03E0H

Figure 3.11 Endpoint1 Data Buffer 0



EP11BUF Set Value	Top Address
00001000B	0100H

Figure 3.12 Example of Endpoint1 Data Buffer 0 Setting

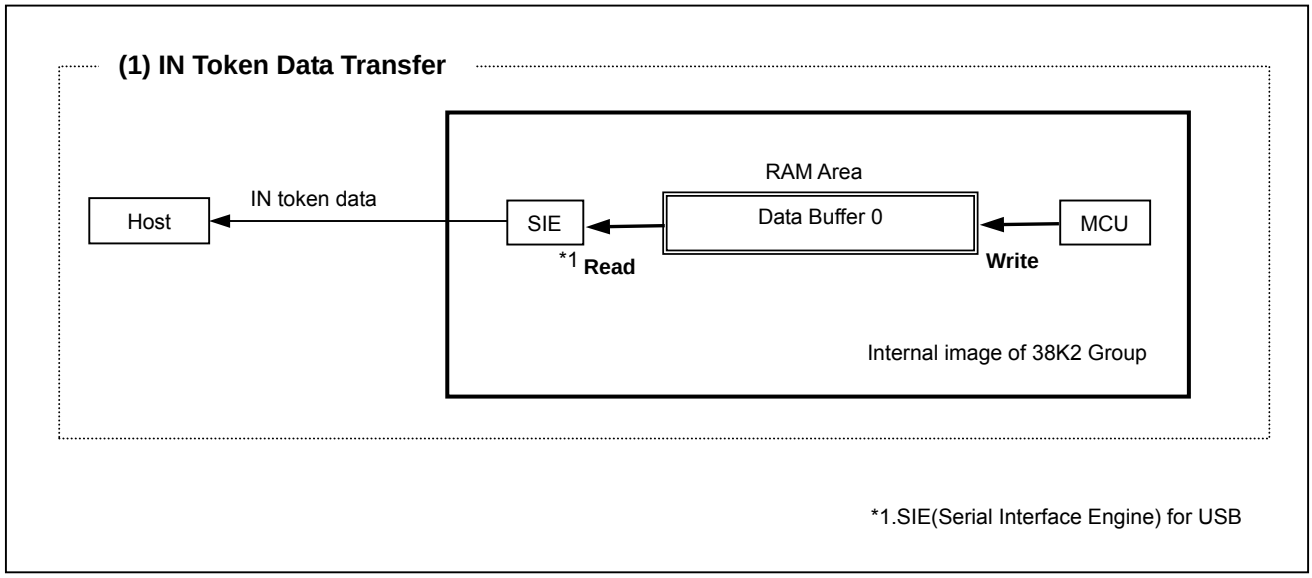


Figure 3.13 Endpoint1 Data Buffer 0 Access

3.3.3. Endpoint1 Interrupt

The Endpoint1 interrupt is used to for data flow control. Endpointx Interrupt Factor Register is shown in Figure 3.14

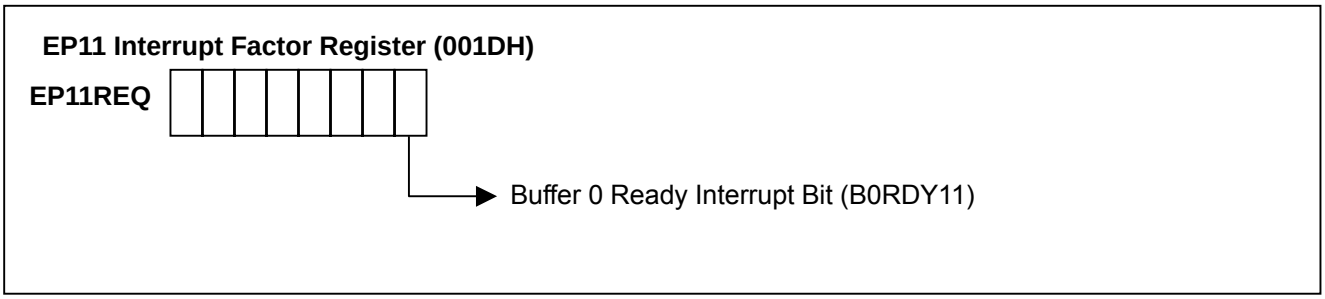
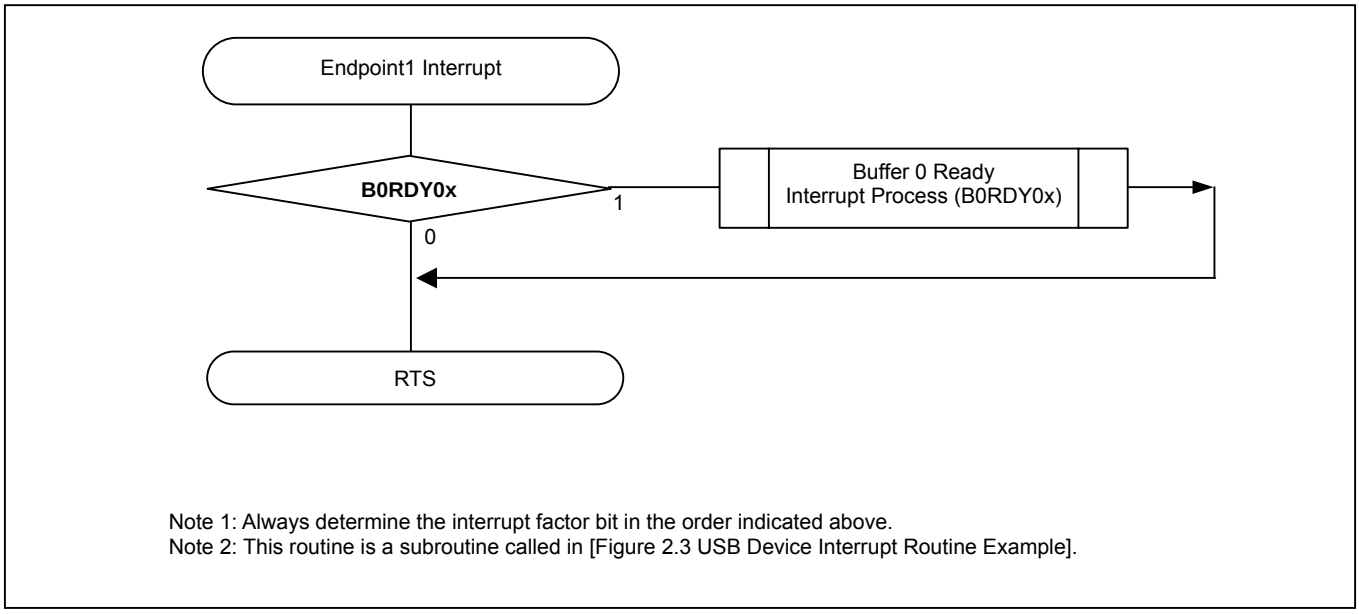


Figure 3.14 EP11 Interrupt Factor Register (EP11REQ)

(1) Buffer 0 Ready Interrupt Bit (B0RDY11)
When Buffer 0 is in the ready state (read/write enabled state), this bit goes to “1” (set by H/W).
This bit must be cleared by F/W when an interrupt occurs. (this bit not cleared to “0” automatically by H/W.)

An example of Endpoint1 interrupt processing is shown in Figure 3.15. Determine the interrupt factor ateach stage and then perform the error process, receive/transfer setup and buffer read/write



Note 1: Always determine the interrupt factor bit in the order indicated above.
Note 2: This routine is a subroutine called in [Figure 2.3 USB Device Interrupt Routine Example].

Figure 3.15 Example of Endpointx Interrupt Processing

3.3.4. Data Receive Format

(1) Receive Setup

Configure Endpointx by specifying the transfer direction (IN), transfer type (interruput).

Figure 3.16 shows an example of an Endpoint1 receive setup procedure.

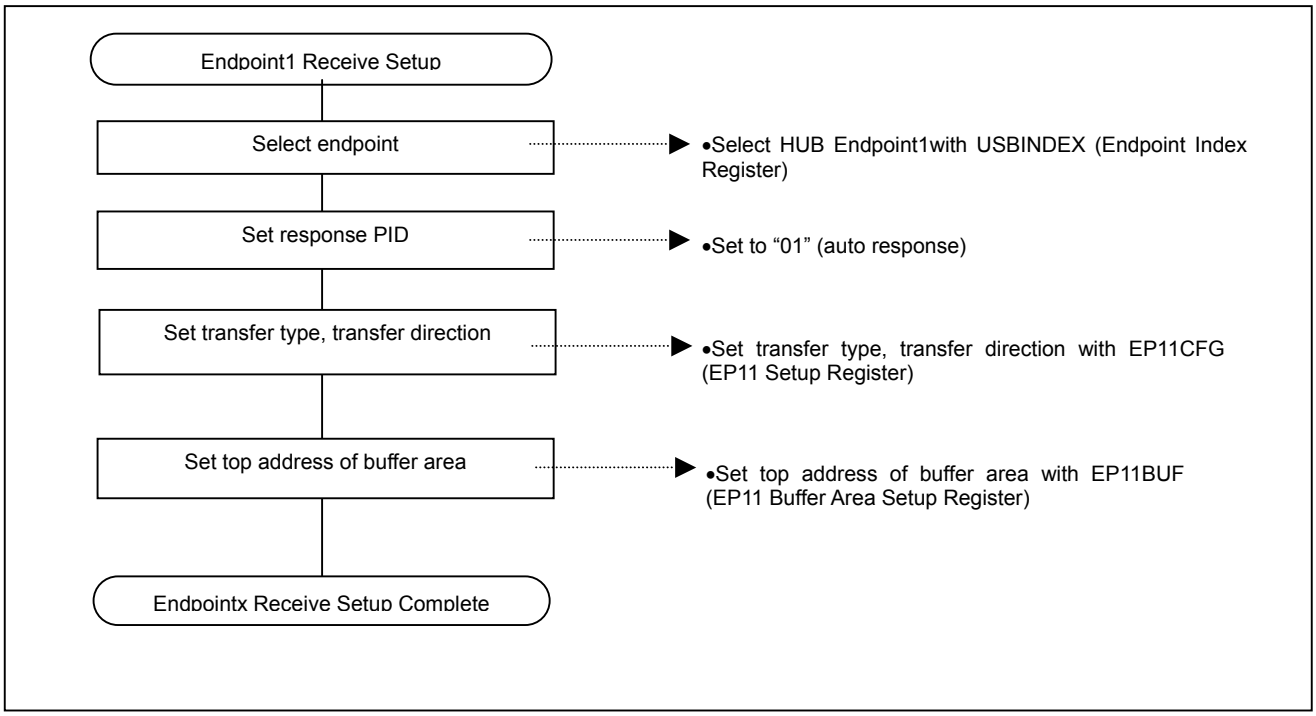


Figure 3.16 Example of Endpoint1 Receive Setup Procedure

(2) Normal Receive

When Endpoint1 sends a packet to the Host, the packet of data must be ready in the data buffer when an IN token is received from the Host.

To prepare one packet of data, confirm that Endpoint1 is not in the HALT state and the data buffer is empty, write the same number of bytes to the data buffer as in the packet data to be transferred, then set B0VAL11 (Buffer 0 Status Bit) of EP11CON2 (EP11 Control Register 2) to “11”. (The B0VAL11 bit is cleared automatically by hardware when an ACK is received normally.)

The number of bytes of packet data to be transferred must be set in EP11BYT0 (EP11 Byte Number Register 0).

When one packet is ready, the data is sent with the next IN token received from the Host.

When the data transfer is completed, an Endpoint1 Buffer 0 Ready interrupt request is generated and the data buffer has room for one packet.

Note that BORDY11 (Endpoint1 Buffer 0 Ready interrupt) must be cleared by the user.

(3) STALL Response

The STALL response sets PID11 (Response PID bit) of EP11CON1 (EP11 Control Register 1) to “1x” (STALL handshake). The STALL response is continuously output for Endpointx IN/OUT tokens in the STALL state. When a STALL is sent, communications are disabled unless the Host CPU cancels the STALL with the ClearFeature. Even when communications are restarted, the STALL is not cancelled unless the PID11 bit is set to “00” or “01”.

3.3.5. Interrupt Transfer Data Receive

(1) Transfer Type Setup

As the transfer type for Endpoint1 is fixed, the transmission is enabled in EP11CFG (EP11 Setup Register). Refer to the data transmission procedure in [Figure 3.16] for details.

(2) Data Transfer Ready Operation

Set the Sequence Toggle Bit Clear Bit (SQCL11) to “1”. Setting the bit to “1” clears the toggle bit and initializes the next data PID to DATA0.

(3) Data Transfer Operation

A NAK is automatically returned when Endpoint1 receives an IN token from the host if data is not set in the Endpoint1 data buffer.

Packet data must be set in the data buffer each time data is transferred (transmitted). Confirm that Endpoint1 is not in the HALT state and the data buffer is empty (B0VAL11 bit ”0” can be written to), then set the transfer packet data. Set the number of bytes of data to be transferred in EP11BYT0 (EP11 Byte Number Register 0), then set B0VAL11 bit to “1” (the B0VAL11 bit is cleared to “0” by H/W after a normal transfer operation).

After the data is set and the IN endpoint receives an IN token from the Host, the data is sent to the Host. When an ACK is received from the Host in response to the data packet transfer, the 1-packet transfer is complete and an Endpoint1 Buffer 0 Ready interrupt is generated.

In interrupt transfers, the data is toggled (in normal transmissions) when an ACK is received from the Host in response to an IN interrupt. But if the endpoint does not receive an ACK from the Host, the same data from the same toggle is sent in response to the next IN token.

The transfer data is written to the data buffers alternately, as follows: DATA0 → DATA1 → DATA0…
Note that the user must clear the Endpoint1 Buffer Ready interrupt (B0RDY11) when an interrupt request occurs. Figure 3.17 shows an example of an Endpoint1 Interrupt data transfer.

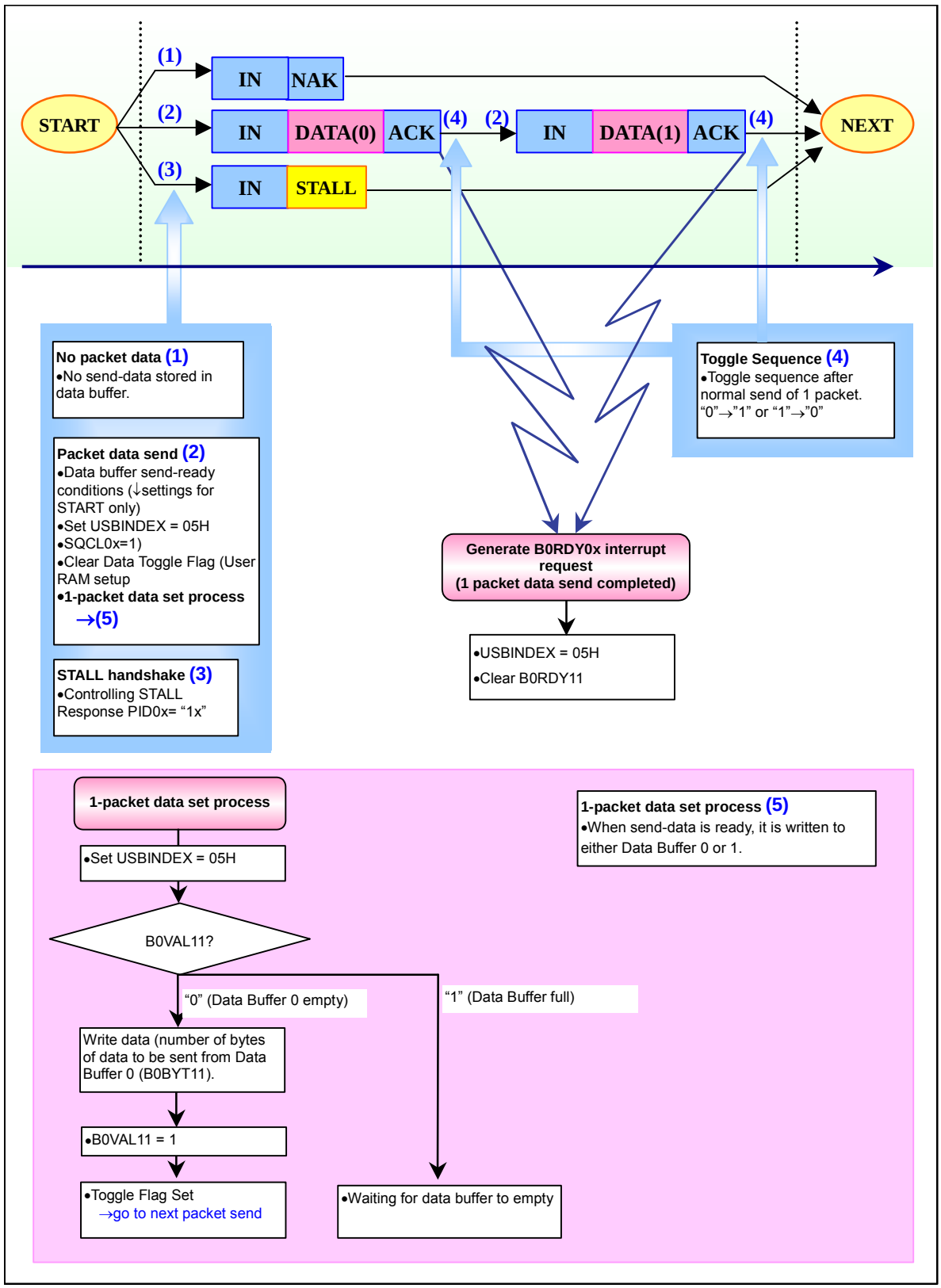
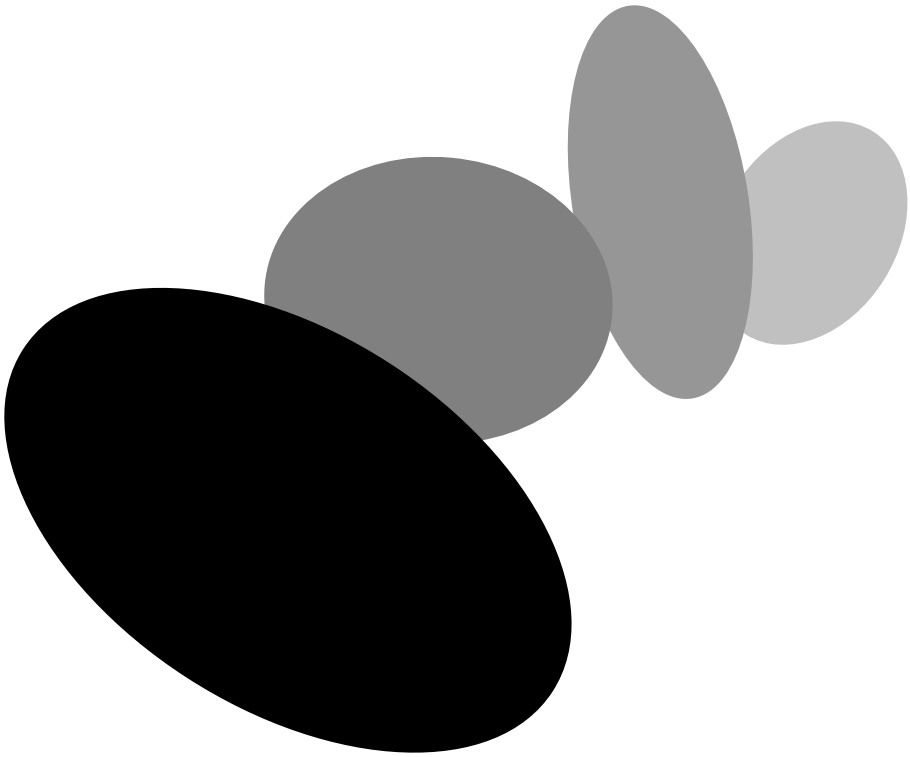


Figure 3.17 Example of Endpoint1 Bulk Data Send



Chapter 4
USB Hub Down Port Controls

The 38K2 Group comes with 2 external-down ports and one internal down-port.
This chapter explains external down-port controls.

4.1. HUB Registers

The USB Hub has 7 registers, including the index register and register window**.** Three of the registers are mapped in the same register window (same address space), and are selected in the index register. These registers are used for down-port control.

The USB Hub interrupts are used for the down-ports. Table 4.1 describes the Hub registers and Table 4.2 shows a list of register sets that can be switched by the index register.

Table 4.1 HUB Registers

Address	Symbol	Register Name
0028H	USBHUBICON	HUB interrupt factor enable register
0029H	USBHUBIREQ	HUB interrupt factor register
002AH	USBHUBINDEX	HUB port index register
002BH	DPXREG1	HUB port field register 1
002CH	DPXREG2	HUB port field register 2
002DH	DPXREG3	HUB port field register 3

0FF9H	DPCTL	Down-port control register

Table 4.2 Registers selected in Hub Down-Port Index Register (HUBINDEX)

Index Register Contents	Address	Symbol	Register Name
00H	002BH	DP1REQ	DP1 interrupt factor register
	002CH	DP1CON	DP1 control register
	002DH	DP1STS	DP1 status register
01H	002BH	DP2REQ	DP2 interrupt factor register
	002CH	DP2CON	DP2 control register
	002DH	DP2STS	DP2 status register

4.2. How to Use USB Hub Down-ports

The 38K2 Group comes with 2 external down-ports and, by using the USB internal functions, an additional internal down-port can also be operated. In 38K2 Group data sheets and in this material, the device connected to the internal down-port is referred to as the “USB Function”. The USB Function is described in detail in Chapter 5.

To use the external down-ports, the down-port functions must be set after enabling the USB Device Block and the USB Hub Device Block.

4.2.1. Down-port Initialization

The external down-ports are initialized when a power-on reset or a USB bus reset interrupt is generated to the USB Hub.

After specifying the down-port in the down-port index register, set the corresponding down-port controls. Figure 4.1 shows an example of the initialization process.

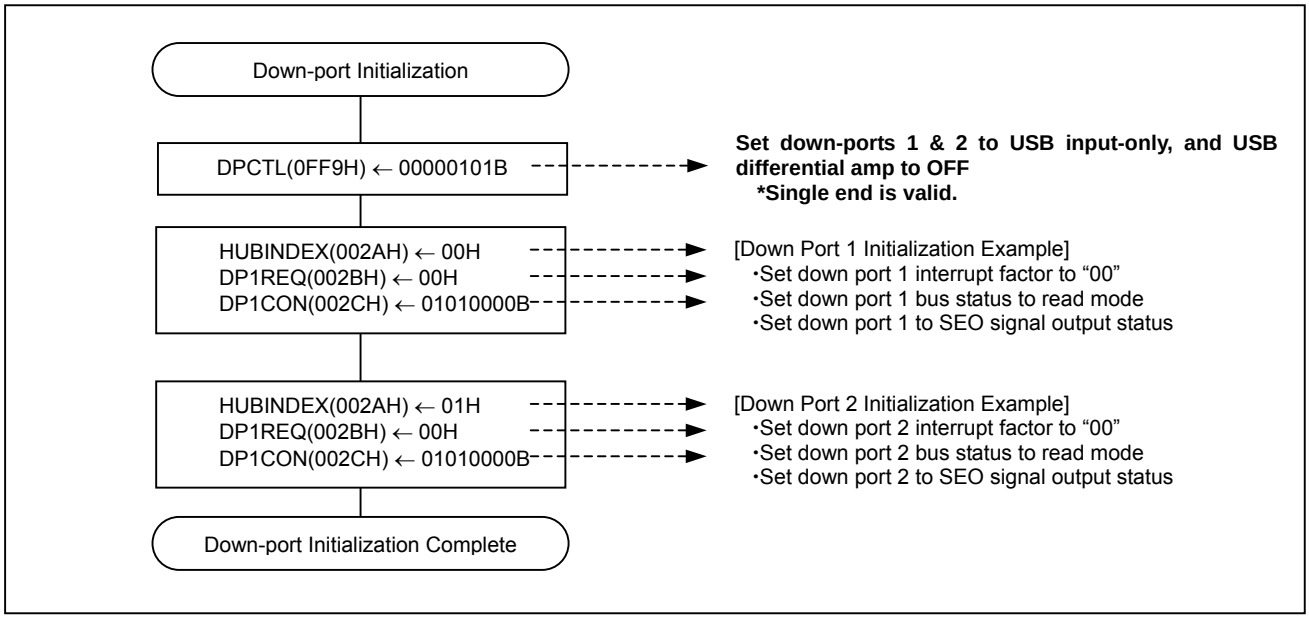


Figure 4.1 Example of Down-port Initialization

4.3. USB Hub Interrupts

The USB Hub interrupt is generated when a status change is detected at a USB Hub down-port. The factors for USB Hub interrupts are the same as that of the down-portx[*1] interrupt.

To enable the USB Hub interrupt, set the USB Hub interrupt enable bit of ICON2 (Interrupt Control Register 2) to “1” and the corresponding bit of HUBICON (Hub Interrupt Factor Register) to “1”. The corresponding bits for HUBIREQ (Hub Interrupt Factor Register) and DPxREQ (DPx Interrupt Factor Register) indicate the status of the interrupt request.

Table 4.3 lists Hub interrupt registers and Figure 4.2 shows an example of a USB Hub interrupt routine.

Table 4.3 HUB Interrupt Registers

Address	Symbol	Register Name
0028H	HUBICON	HUB interrupt factor enable register
0029H	HUBIREQ	HUB interrupt factor register
0018H	HUBINDEX	HUB down-port index register
001DH	DPxREQ	DPx interrupt factor register (x = 1, 2)
003DH	IREQ2	Interrupt request register 2
003FH	ICON2	Interrupt control register 2

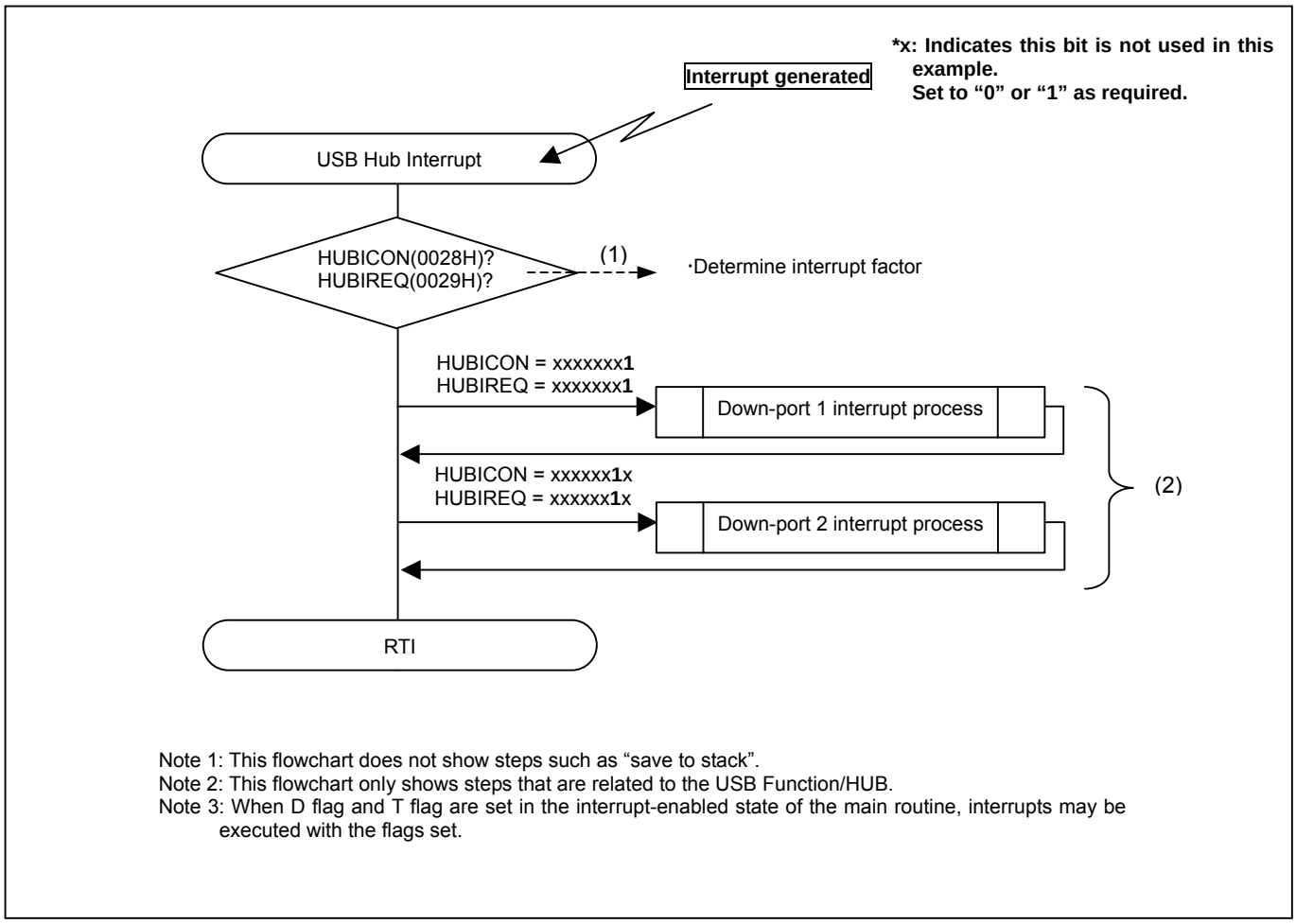


Figure 4.2 Example of USB Hub Interrupt Routine

Supplemental notes for [Figure 4.2 Example of USB Hub Interrupt Routine]:

(1) Interrupt Factor Determination Method #1
·The interrupt factor is determined with the AND condition of the enable bit states HUBICON (Hub Interrupt Factor Enable Register) and HUBIREQ (Hub Interrupt Factor Register). When both bits are "1", the factor is valid.

(2) Interrupt Factor Determination Method #2
·To prevent factors from being ignored and left unprocessed when multiple interrupt factors are generated simultaneously, each factor is checked and processed according to order of receipt, from the top address. Depending on the transmission timing, each time an interrupt is generated, all bits of both USBICON and USBIREQ are judged and interrupts are processed accordingly to avoid continuously processing one factor.

◆ Sample procedures (1) and (2) are included in RENESAS's "38K2 Group USB Sample Firmware."

4.3.1. Down-port Interrupts

This section provides supplemental information concerning the down-portx interrupts, as listed below.

○ **Down-portx interrupts**
(1) Down-portx detachment detection interrupt
(2) Down-portx attachment detection interrupt
(3) Down-portx error interrupt
(4) Down-portx resume interrupt
(5) Down-portx bus change detection interrupt

● **Conditions for generating a down-portx interrupt**

A down-portx interrupt will be generated when all of the following conditions are fulfilled:
- I Flag (Interrupt Disable Flag) of the processor status register is “0” (interrupt enabled)
- USB Hub interrupt enable bit of ICON2 is “1” (interrupt enabled)
- USB Hub interrupt request bit of IREQ2 is “1” (interrupt requested).
- Hub down-portx interrupt bit (DPxE) corresponding to down-port of HUBICON is “1” (interrupt enabled)
- Hub down-portx interrupt bit (DPx) corresponding down-port of HUBIREQ is “1” (interrupt enabled)
  *This bit is set so that at least one bit of DPxREQ (DPX Interrupt Factor Register) is set to “1”. It cannot be set or cleared by software.
- The interrupt bit for DPxREQ that corresponds to the down-port (value specified in HUBINDEX) is “1” (request generated).(To reference DPxREQ, set the down-portx that corresponds to HUBINDEX.)

● **Down-portx interrupt processing**
- An example of an interrupt routine is shown in Figure 4.4.
- When a down-portx interrupt is generated, the Hub Interrupt Request Flag (root flag) is cleared as soon as the interrupt is accepted, and therefore does not need to be cleared during the interrupt process. DPx (Hub Down-portx Interrupt Bit) of HUBIREQ cannot be cleared with software. The interrupt that corresponds to down-portx (value specified in HUBINDEX) is cleared by hardware when DPxREQ is cleared to “00.”

The DPx interrupt factor register is shown in Figure 4.3.

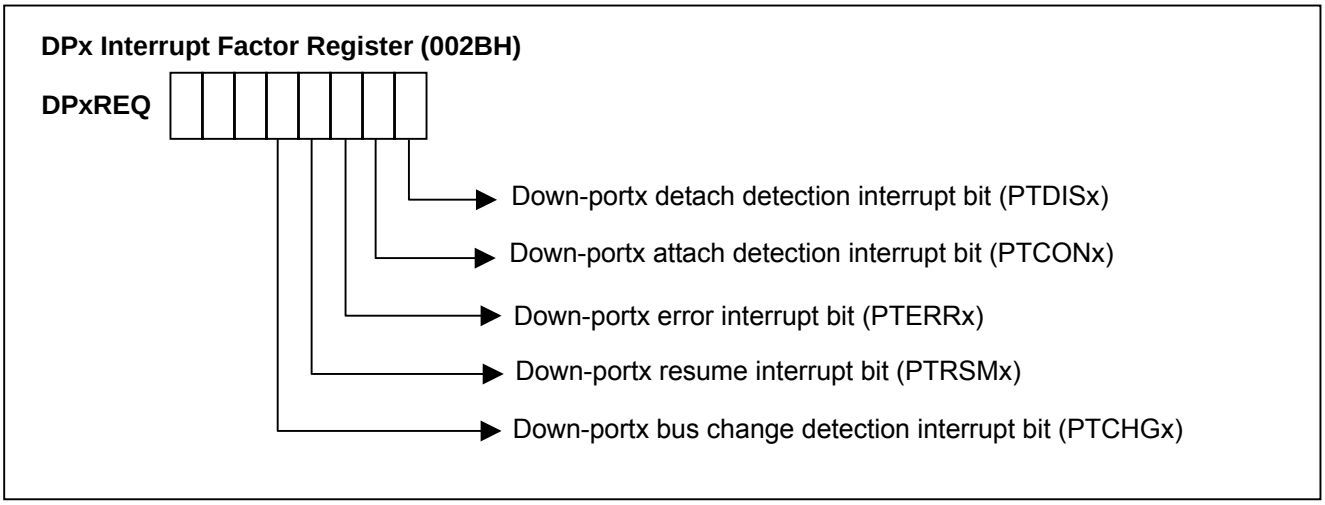


Figure 4.3 DPx Interrupt Factor Register (DPxREQ)

(1) Down-portx Detach Detection Interrupt Bit (PTDISx)
When DSCONNx = "1" and the bus detach state is detected at down-portx (2.5 μs or more SE0), PTDISx is set to "1" by hardware. This bit must be cleared when an interrupt is generated.

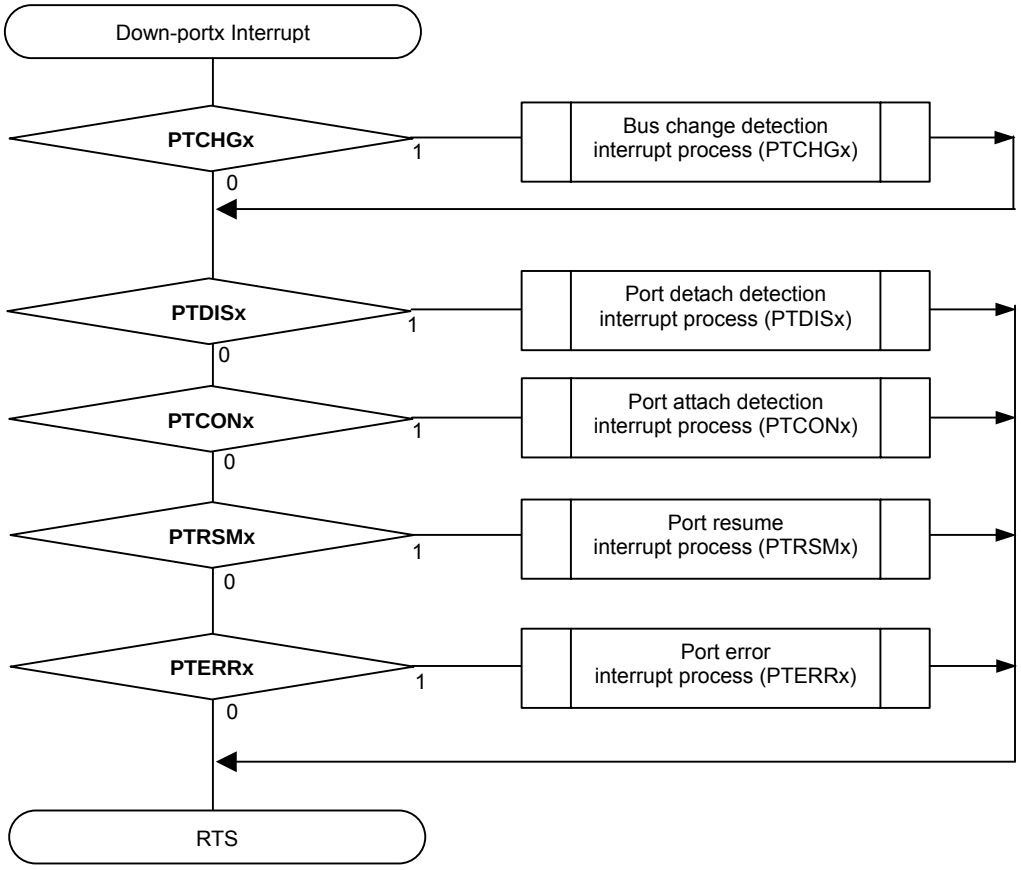
(2) Down-portx Attach Detection Interrupt Bit (PTCONx)
When DSCONNx = "0" and the bus connection state is detected at down-portx (2.5 μs or more J or K), PTCONx is set to "1" by hardware. This bit must be cleared when an interrupt is generated.

(3) Down-portx Port Error Interrupt Bit (PTERRx)
PTERRx is set to "1" by hardware when a port error is detected at down-portx, and must be cleared when an interrupt is generated.

(4) Down-portx Resume Interrupt Bit (PTRSMx)
PTRSMx is set to "1" by hardware when a resume signal is detected at down-portx in the USB Hub suspended state or port suspended state. This bit must be cleared when an interrupt is generated.

(5) Down-portx Bus Change Detection Interrupt Bit (PTCHGx)
PTCHGx is set to "1" by hardware when a bus change is detected at down-portx in the USB Hub suspended state. This bit must be cleared when an interrupt is generated.

Figure 4.4 shows an example of the down-portx interrupt routine.



Note 1: Each interrupt factor bit must be judged in the order shown above.
Note 2: This routine is a subroutine called in [Figure 4.2 Example of USB Hub Interrupt Routine].

Figure 4.4 Example of Down-portx Interrupt Routine

4.4. Down-port State Transition

The USB hub downstream port supports all states defined in USB Specification Version 2.0. The downstream port moves from one state to another according to the current transfer and interrupt request operations.

Down-port state transition in the 38K2 Group USB hub occurs with the following operations: detach detection interrupt, attach detection interrupt, port error interrupt, resume interrupt, bus change detection interrupt, and endpoint0 USB hub class request processing according to down-port interrupt requests, as shown in [4.3 USB Hub Interrupts].

Figure 4.5 shows 38K2 Group down-port state transitions. For more details, refer to the HUB Specification in USB Specification Version 2.

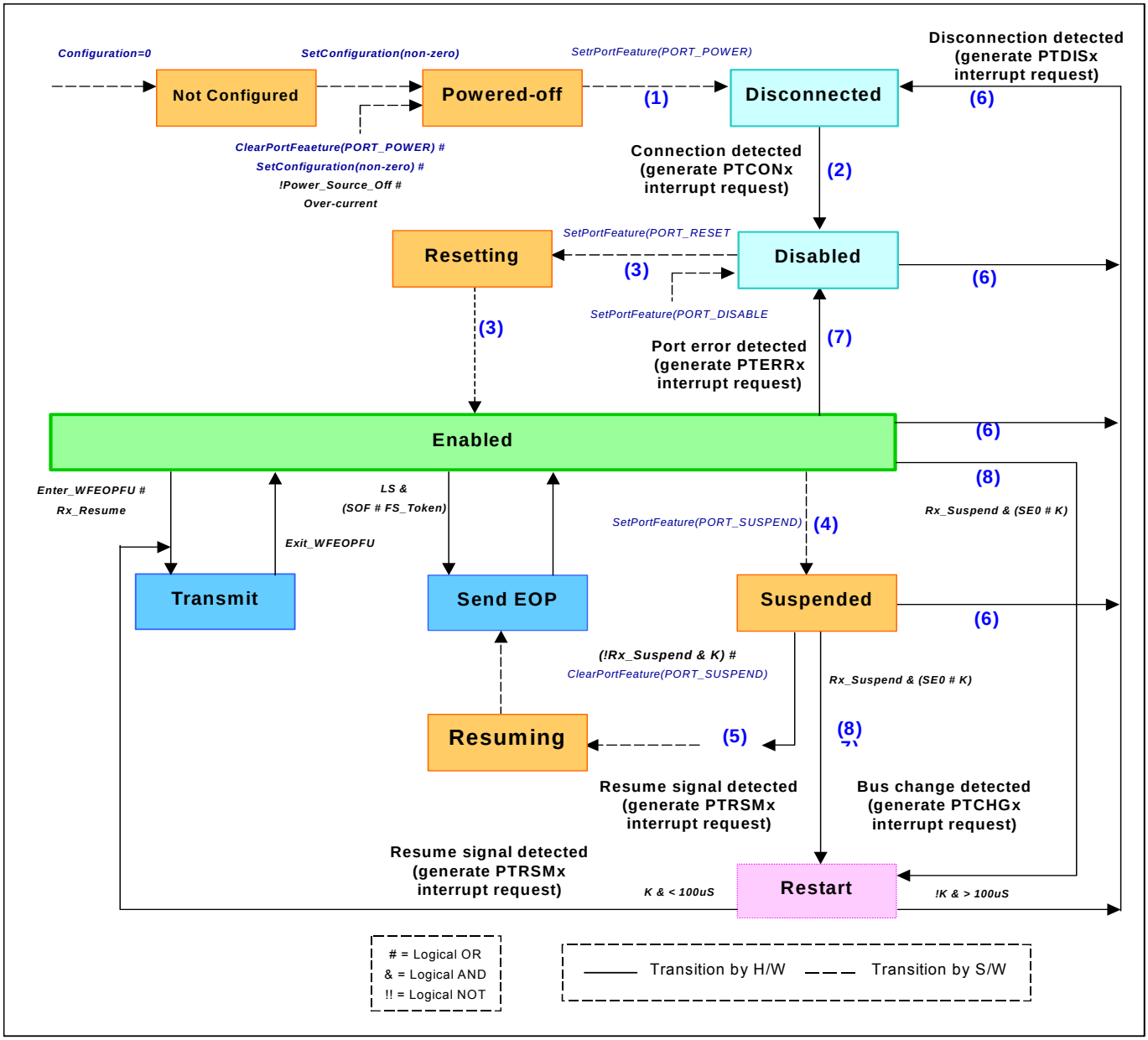


Figure 4.5 Down-port State Transitions

Supplemental information concerning [Figure 4.6 Down Port State Transition Configuration]:
This section discusses down-port state transitions, explaining in detail USB Hub down-port interrupt factors (interrupt bits of DPx interrupt factor register) and DPx control register control bits.

● USB Hub in Normal State

(1) Powered-off → Disconnected
- When the hub receives a SetPortFeature (PORT_POWER) request from the Host, it gets ready to detect a down-port attachment.
- DSCONNx (Down-portx Attachmet Bit) of DPxCON (DPx Control Register) is set to "0", DSDETEx (Down-portx Attach State Detect Enable Bit) is set to "1", and the Hub is enabled for attach detection (waiting for PTCONx interrupt request).
(However, the hub is set up with a 100ms waiting period after power is turned on, to provide power stabilization time. Attach/detach detection is disabled during this time.
- HRWUE (Hub Up-port Remote Wake-up Signal Output Enable Bit) of HUBICON is then set to "1" (enabled). By enabling this bit, when 2.5 μs or more of the K state is detected from the down-port in the Hub suspended state, the K state signal is output to the up-port and, simultaneously, HRWU is automatically set by hardware.

(2) Disconnected → Disabled
- The PTCONx interrupt request is generated when a down-port attach (D+ or D- pull-up) is detected.
- DSCONNx of DPxCON is set to "1", DSDETEx is set to "0" and then reset to "1" 4ms later, and the bus is enabled for detach detection (waiting for PTDISx interrupt request). (Attach/detach detection is disabled until after the 4ms period that follows transition to the Disabled state. The detection function is enabled after timer count is performed by firmware for 4ms.)
- When a PTCONx interrupt request occurs, the DSDETEx bit is cleared automatically by hardware.

(3) Disabled → Resetting → Enabled
- When the hub receives a SetPortFeature (PORT_RESET) request from the Host, it outputs a RESET signal (10ms or more of SE0) to the down-port.
- DSRSTOx (Down-portx Attach SE0 Signal Transmit Bit) of DPxCON is set to "1", and SE0 is output to the down-port. After timer-count is performed by firmware for 10ms (RESET signal time), DSRSTOx is set to "0" and SE0 output is ended.
- Once SE0 output is ended, the bus goes to the enabled state and DSPTENx (Down-port Enable Bit) of DPxCON is set to "1".
- DSCONNx of the DPxCON register is set to "1", DSDETEx is set to "0" and then reset to "1" 4ms later, and the bus is enabled for detach detection (waiting for PTDISx interrupt request).
- DSRMODx (Down-portx Bus State Read Mode Control Bit) of DPxCON register is set to "0", the contents of DPxSTS (Down-port Status Register) are read, and the transfer speed of the down-port device is checked. After the contents are read, DSRMODx is set to "1".
- If the down-port device is at full-speed, DSLSPDx (Down-port USB Transfer Speed Selection Bit) of DPxCON register is set to "0", if low-speed, the bit is set to "1".

(4) Enabled → Suspended
- When the hub receives a SetPortFeature (PORT_SUSPEND) from the Host, it puts the down-port in the suspended state.
- DSSUSPx (Down-portx Suspend Bit) of DPxCON is set to "1".
- DSDETEx is set to "0" and then reset to "1" 4ms later, and the bus is enabled for detach detection (waiting for PTDISx interrupt request).
- (Attach/detach detection is disabled until the 4ms period after transition to the Suspended state. The detection function is enabled after timer-count is performed by firmware for 4ms.)

(5) Suspended → Resuming

·When the hub receives a ClearPortFeature (PORT_SUSPEND) request from the Host or detects the remote wake-up signal (K state signal) from the down-port, it outputs a resume signal to the down-port.
·When the remote wake-up signal from the down-port is detected, the hub generates a PTRSMx interrupt request.
·When the down-port receives the ClearPortFeature (PORT_SUSPEND) from the Host, DSRMOx (Down-portx Resume Signal Transmit Bit) DPxCON is set to "1" and the hub outputs 20ms of the K state signal to the down-port. After timer-count is performed by firmware for approximately, DSRSMOx is set to "0", and the K state signal output is ended (at this time, Low-Speed EOP output is ended by hardware.)
·When a PTRSMx interrupt request occurs, the DSSUSPx bit is cleared automatically by hardware.

(6) Disabled or Enabled or Suspended → Disconnected

·When the hub detects a detach (cancellation of D+ or D- pull-up) at a down-port, it generates the PTDISx interrupt request.
·DSCONNx is set to "0", DSDETEx is set to "1", and the hub goes to the attach detection enable state. (waiting for PTCONx Interrupt)
·When the PTDISx interrupt request occurs, bits DSPTENx, DSSUSPx, and DSDETEx are automatically cleared by hardware.

(7) Enabled → Disabled

·When the hub receives a ClearPortFeature (PORT_ENABLE) or the down-port generates an error, a PTERRx interrupt request is generated.
·DSDETEx is set to "0", and then reset to "1" 4ms later, and the hub is enabled for detach detection (waiting for PTDISx interrupt request). (Attach/detach detection is disabled until the 4ms period after transition to the Disabled state. The detach detection is enabled after timer-count is performed by firmware for 4ms.))
·When the PTERRx interrupt request occurs, bits DSPTENx and DSDETEx are automatically cleared by hardware.
·When the hub receives a ClearPortFeature (PORT_ENABLE) for the USB Function (internal down-port) is received, the USB Function address is initialized (bit AD0E of USBAE Register is set to "0").

●USB Hub in Suspended state

(8) Suspended or Enabled to Restart → Transmit → Enabled

·When the USB hub is in the suspended state and 2.5μs or more of the K state signal is detected from the down-port, the hub generates a PTRSMx interrupt. However, when the hub is in the suspended state, supply of the USB clock to the USB device block is stopped. Therefore, when the change in the down-port bus state is first detected, a PTCHGx interrupt request is generated. This PTCHGx interrupt is generated even when the USB clock is not being supplied to the USB device block, in the same manner as a resume interrupt received when the hub is in the suspended state.
·After the PTCHGx interrupt request is generated, the hub recovers its pre-suspend settings, and enables the PLL circuit output clock. For more details on the recovery method, refer to [2.4.3 USB Resume Interrupt].
·The PTRSMx interrupt request is generated after the PTCHGx interrupt request is generated.
·The continuous 2.5μs K state signal described above is actually the remote wake-up signal from the down-port. The hub outputs the K state signal to the up-port, and simultaneously sets HRWU (Hub Up-port Remote Wake-up Output Bit) of HUBIREQ to "1" by hardware. After the timer count is performed by firmware for approximately 10ms (output period of remote wake-up signal), HRWU is set to "0", and the K state signal output is ended. At this time, the HRWUE bit is set to "0" (disabled) once, then reset to "1" (enabled).
·The DSSUSPx bit is automatically cleared to "0" by hardware when the PRTSMx interrupt request is generated.

(9) Enabled or Suspended to Restart → Disconnected

- When the USB hub is in the suspended state and 2.5μs or more of SE0 signal is detected, the hub generates a PTDISx interrupt request. However, as in (8) above, when the change in the down-port bus state is first detected, a PTCHGx interrupt request is generated.
- After the PTCHGx interrupt request is generated, the hub recovers its pre-suspend settings, and enables the PLL circuit output clock. For more details on the recovery method, refer to [2.4.3 USB Resume Interrupt].
- The PTDISx interrupt request is generated after the PTCHGx interrupt request is generated.
- The continuous 2.5μs of the SE0 signal is actually the down-port detachment signal The Hub sets WKUP (Remote Wake-Up Bit) to "1" and outputs the K state signal to the up-port. After the timer count is performed by firmware for approximately 10ms (output period of remote wake-up signal), WKUP is set to "1", and the K state signal output is ended.
- DSCONNx is set to "0", DSDETEx is set to "1", and the Hub is set to the attach detection enabled state (waiting for PTCONx interrupt request).
- The DSPTENx, DSSUSPx and DSDETEx bits are automatically cleared to "0" by hardware when the PTDISx interrupt request is generated.

(10) Disconnected → Disabled

- When the hub is in the suspended state and an attachment (either J or K state signal) is detected at the down-port, the Hub generates a PTCONx interrupt request. However, as in (8) above, when the change in the down-port bus state is first detected, a PTCHGx interrupt request is generated.
- After the PTCHGx interrupt request is generated, the hub recovers its pre-suspend settings and enables the PLL circuit output clock. For more details on the recovery method, refer to [2.4.3 USB Resume Interrupt].
- After the Hub restarts the PLL circuit output clock and the J or K state signal is detected at regular intervals, the hub generates a TDISx interrupt request.
- The WKUP bit is set to "0" and the K state signal is output to the up-port. After the timer count is performed by firmware for approximately 10ms (output period of remote wake-up signal), WKUP is set to "0", and the K state signal output is ended.
- DSCONx (Down-portx Attach Bit) of DPxCON is set to "1", DSDETEx is set to "1", and the hub is enabled for detachment detection (waiting for PTDISx interrupt request).
- The DSDETEx bit is automatically cleared to "0" by hardware when the PTCONx interrupt request is generated.

4.5. Down-port Status Information

When the USB Hub is configured, the host issues an Endpoint1 Interrupt IN within a fixed cycle and requests the down-port status information.

The USB Hub/down-port status change information is transferred according to the bit configuration shown in Figure 4.6. The 38K2 Group transfers data for 3 down-ports, and therefore, the transfer is fixed at one byte (the number of transmit bytes depends on the number of down-ports).

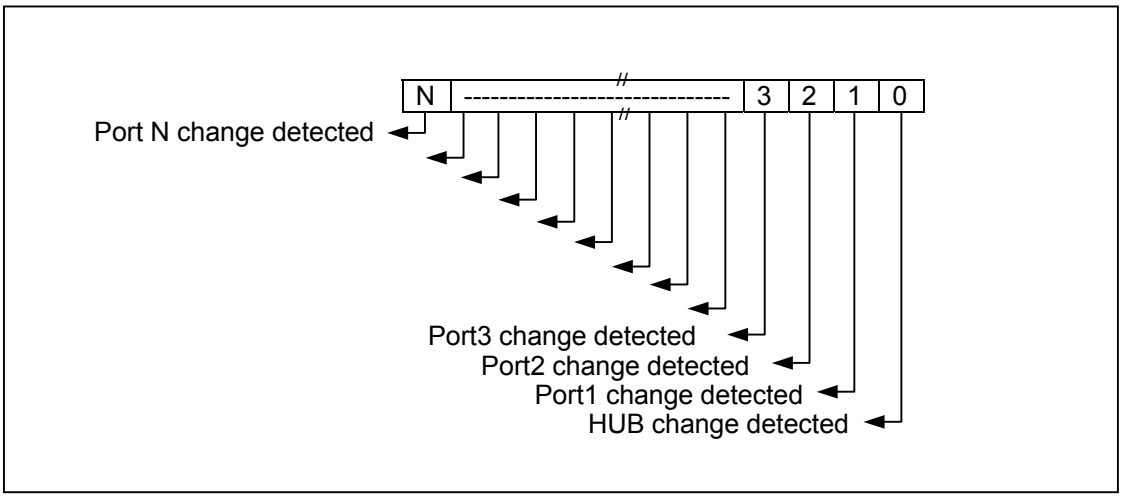


Figure 4.6 USB Hub/Port Status Endpoint1 Data Buffer 0

4.6. Down-port Over-Current Detection/Over-Current Protection

In USB Specification Version 2.0, the power supply for the bus-powered Hub down-port must have ON/OFF controls. Figure 4.7 shows a hardware example of down-port power ON/OFF, current control, and over current detection using a general-purpose current control IC. Table 4.4 lists examples of the general-purpose current control IC functions.

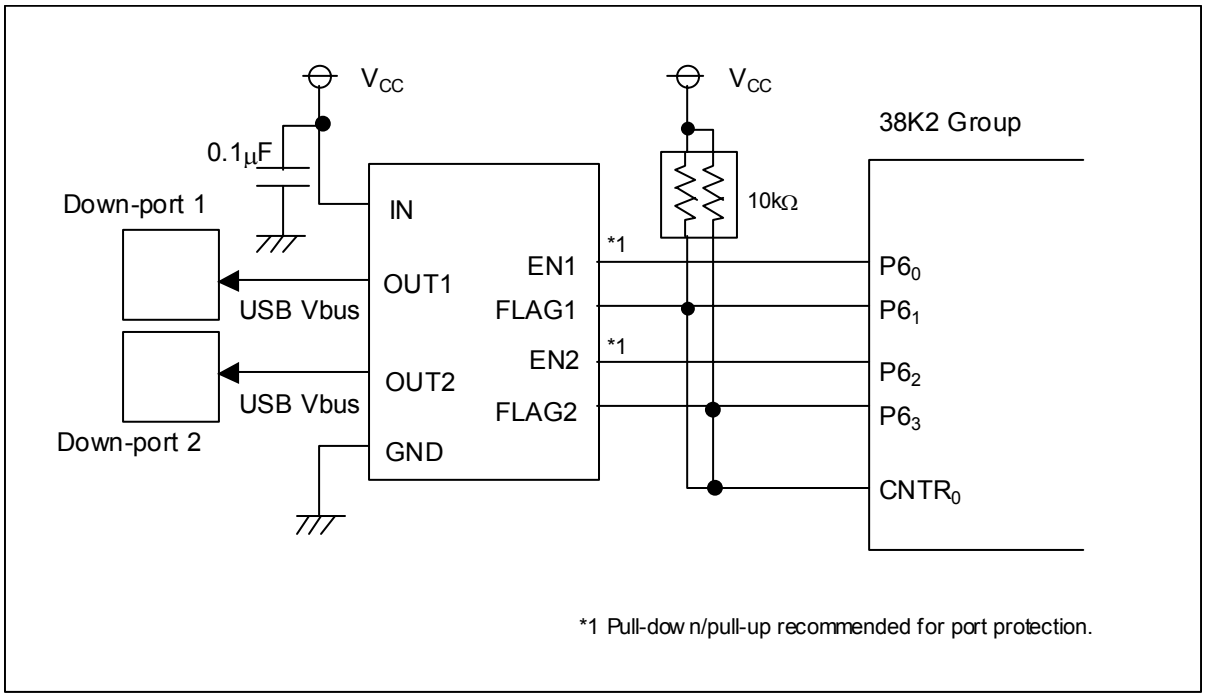


Figure 4.7 Connection Example of General-purpose Current Control IC

Table 4.4 Examples of General-Purpose Current Control IC Functions

Symbol	Function
EN1	Down-port 1 current SW
OUT1	Down-port 1 current output
FLAG1	Down-port 1 over-current detection (current limit is held in 1A)
EN2	Down-port 2 power supply SW
OUT2	Down-port 2 power supply output
FLAG2	Down-port 2 over-current detection (current limit is held in 1A)
IN	Power supply pin
GND	GND

Figure 4.7 shows an example of the flow for detecting a down-port over-current from the power supply control IC to a general-purpose port (P60, P62) or to the CNTR- pin. An example of the software process flow is shown in Figure 4.8.

(1) Over-current detection at general-purpose port

Over-current occurs when "L" level (according to power control IC spec) is detected at either P60 or P62 for 1ms or more. To ensure a stable level during detection, apply chattering prevention for a few seconds ((length of time varies according to each user's system).

When over-current is detected, power supply output to the down-port is turned OFF and the port status is set to the power-off state (power-off/over-current detection).

(2) Over-current detection at CNTR0 pin

The CNTR0 interrupt (external factor interrupt) is used to recover from an over-current detection in the suspended state.

The port recovers with a falling edge (H→L), and the over-current is detected in the same flow as described in (1) above.

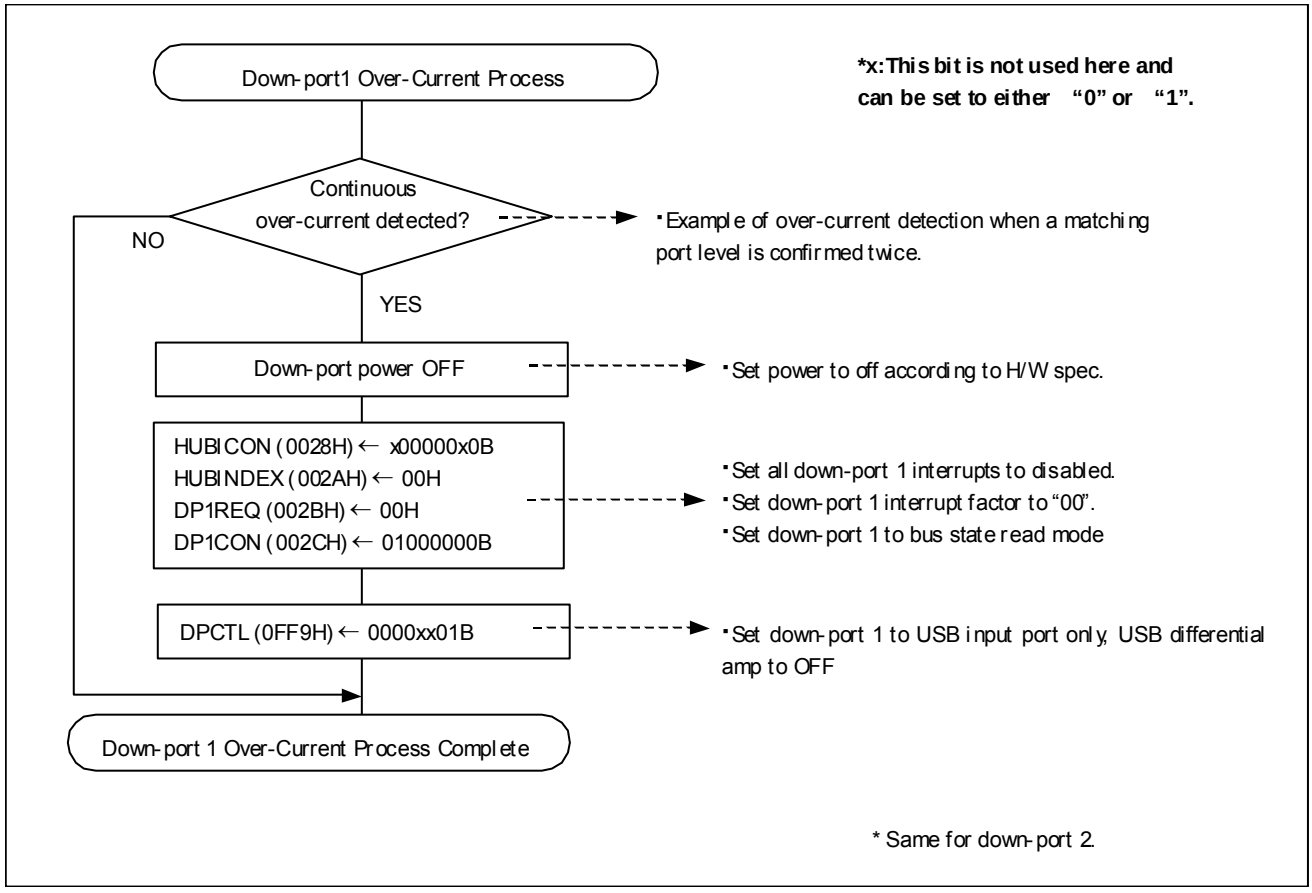
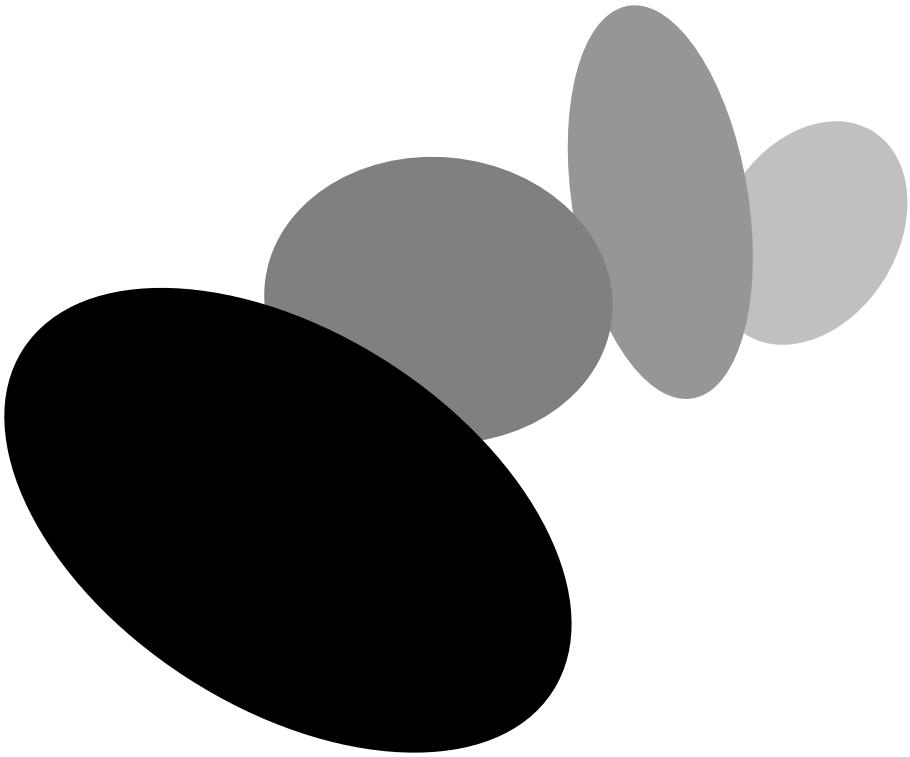


Figure 4.8 Process Flow Example at Down-port Over-Current Detection



Chapter 5
USB Function Controls

This Chapter explains each transfer type.

5.1. Using the USB Function

The 38K2 Group can operate with a pseudo-device connected to the internal down-port through use of the USB Function address.

The USB Function reset, suspend and resume processes are performed as if for a USB compound device, in the same manner as the USB Hub. However, because a pseudo-device is connected to the Hub down-port, the Host controls the USB Function as a class request (port request) to the down-port.

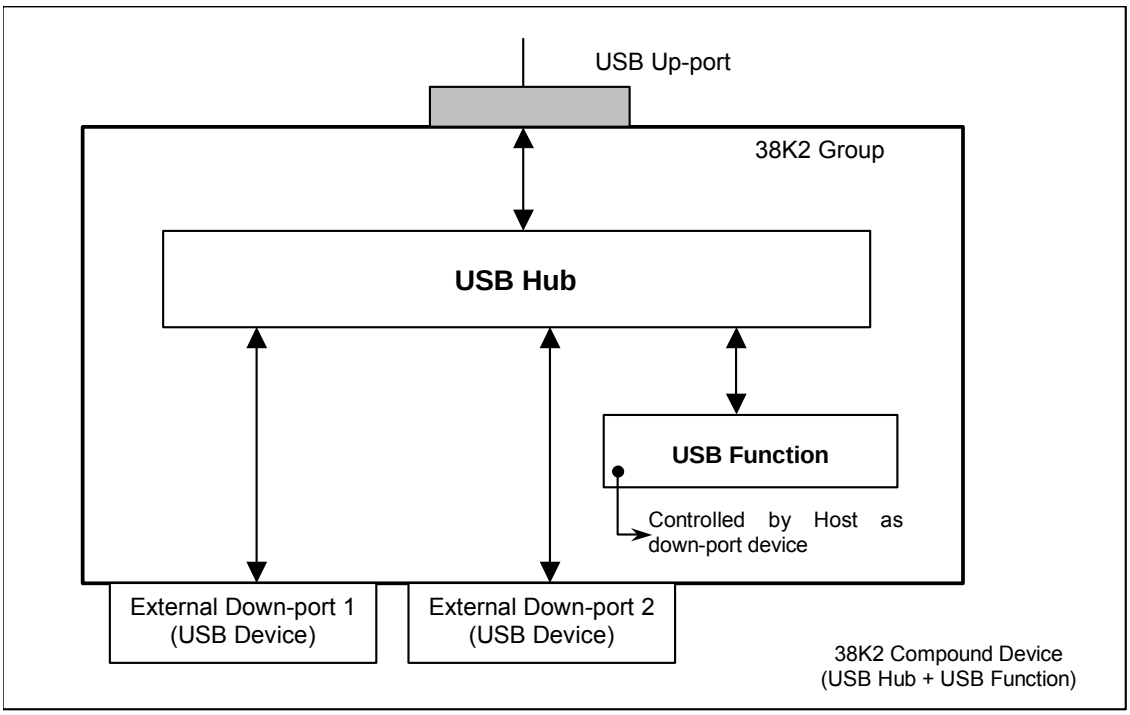


Figure 5.1 Internal Image of 38K2 Group

5.2. Device State Transition

The 38K2 Group USB Function comes with all states defined in USB Specification Version 2.0 and is able to move between states. The USB Function changes device states in the following operations: Endpoint0 Hub class request process, Endoint0 device standard request process, USB Hub USB suspend interrupt request, and USB resume interrupt request.

The device state transition configuration for the USB Function is shown in Figure 5.2

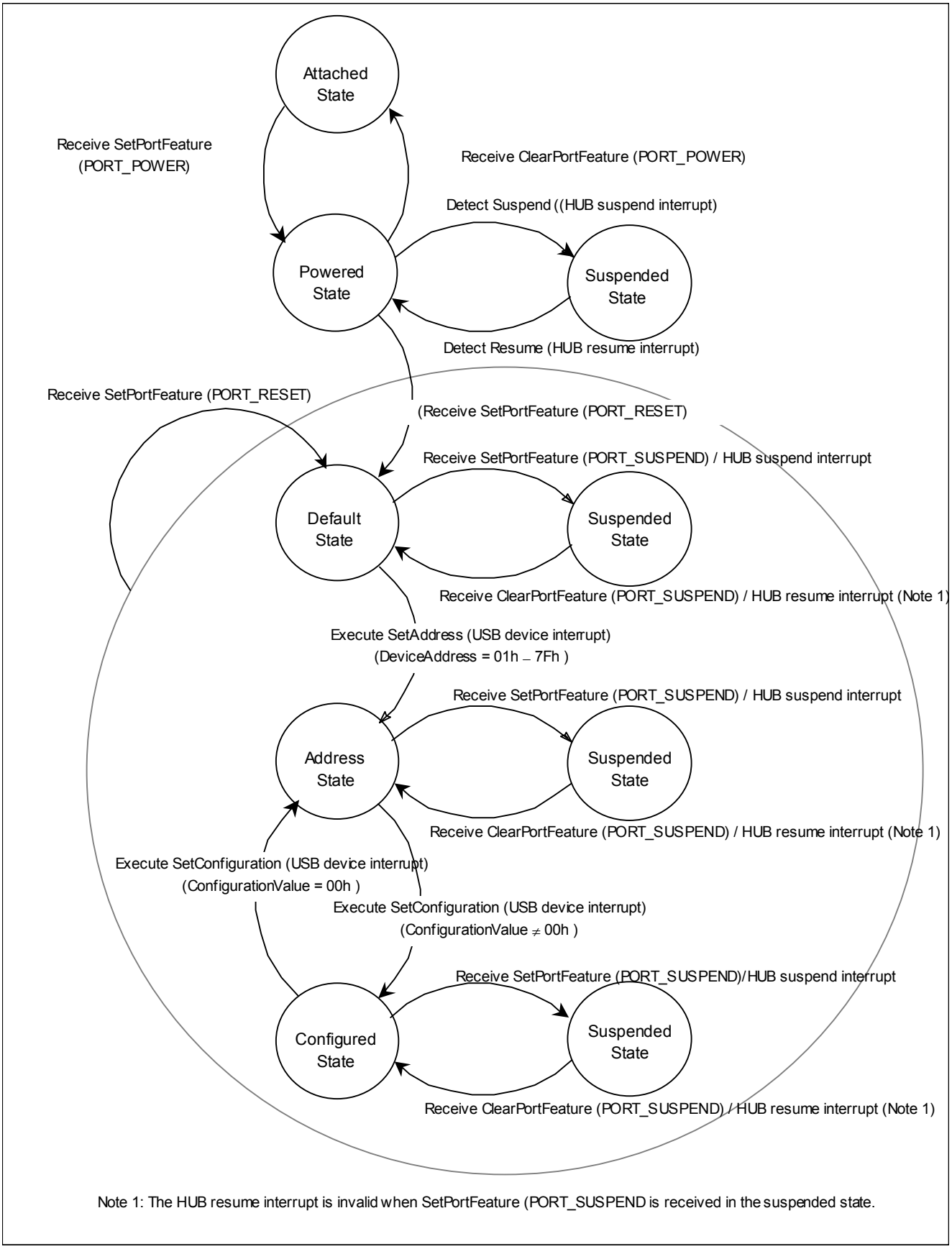


Figure 5.2 Device State Transitions for USB Function

5.3. USB Reset (initialization)

This section explains an example of the initialization procedure for the Device (USB Hub + USB Function).

●Receiving a USB bus reset interrupt

This interrupt executes the USB Function address initialization (disables the USB Function address), which is the same as the USB Hub initialization process described in [2.3 USB Bus Reset Interrupt].

●Receiving a SetPortFeature (PORT_RESET) request

When the USB Hub is configured, the Host sends a Hub request, the SetPortFeature (PORT_RESET) request, to execute a RESET of the down-port. When this request is received, the USB Function (internal down-port) is initialized.

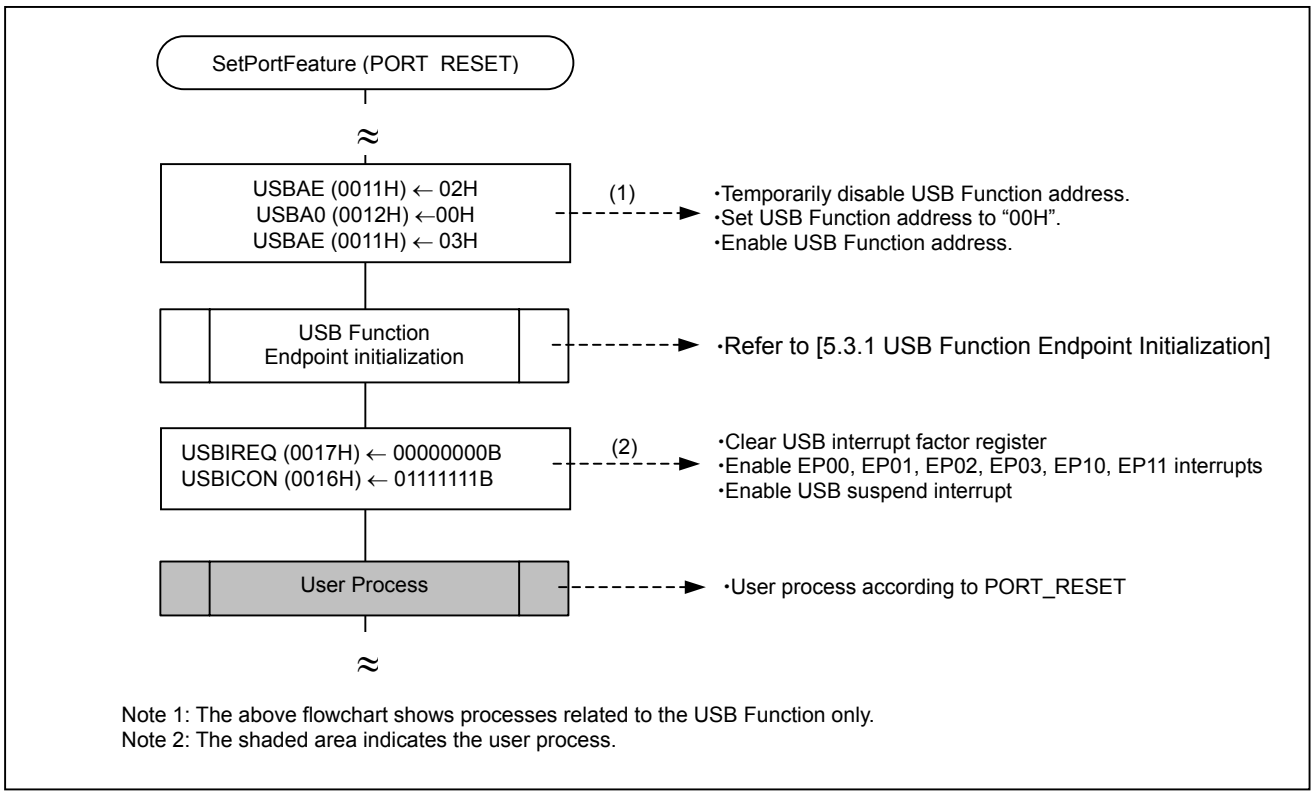


Figure 5.3 USB Example Flow of Function Initialization by SetPortFeature (PORT_RESET)

Supplemental notes for [Figure 5.3 USB Example Flow of Function Initialization by SetPortFeature (PORT_RESET)]:

(1) USB Function Address Initialization
·When setting the USBA0 register address, the contents of the USBAE register become valid by changing the USB Function enable bit (AD0E) from "0" to "1". However, if the host executes SET_ADDRESS when the AD0E bit is "1", the contents of the USBA0 register become valid automatically after the control transfer is completed (hardware control).

(2) USB Interrupt Settings
·Clear USBIREQ (USB Interrupt Factor Register). (USBIREQ is cleared automatically when USB module (USBE bit = "0") is reset).
·Enable SUSE (USB Suspend Interrupt) of USBICON, EP00E (USB Function/Endpoint0 Interrupt), EP10E (USB Hub/Endpoint0 Interrupt). Set EP0xE (USB Function/Endpoints 1 to 3 Interrupts) and EP11E (USB Hub/Endpoint1 Interrupt) as necessary.

5.3.1. USB Function Endpoint Initialization

Specify the endpoints to be setup in the USB Index Register, then set MAX packet size (for OUT transfers), transfer procedure, etc. for each endpoint. Enable USB interrupts as necessary.

When specifying the start address of the buffer area for each endpoint, make sure the addresses do not overlap in single and double buffers modes. Refer to [5.8.5 Buffer Area Setup for Endpoints 1– 3] for more details.

Figure 5.4 shows an example of USB Function endpoint initialization.

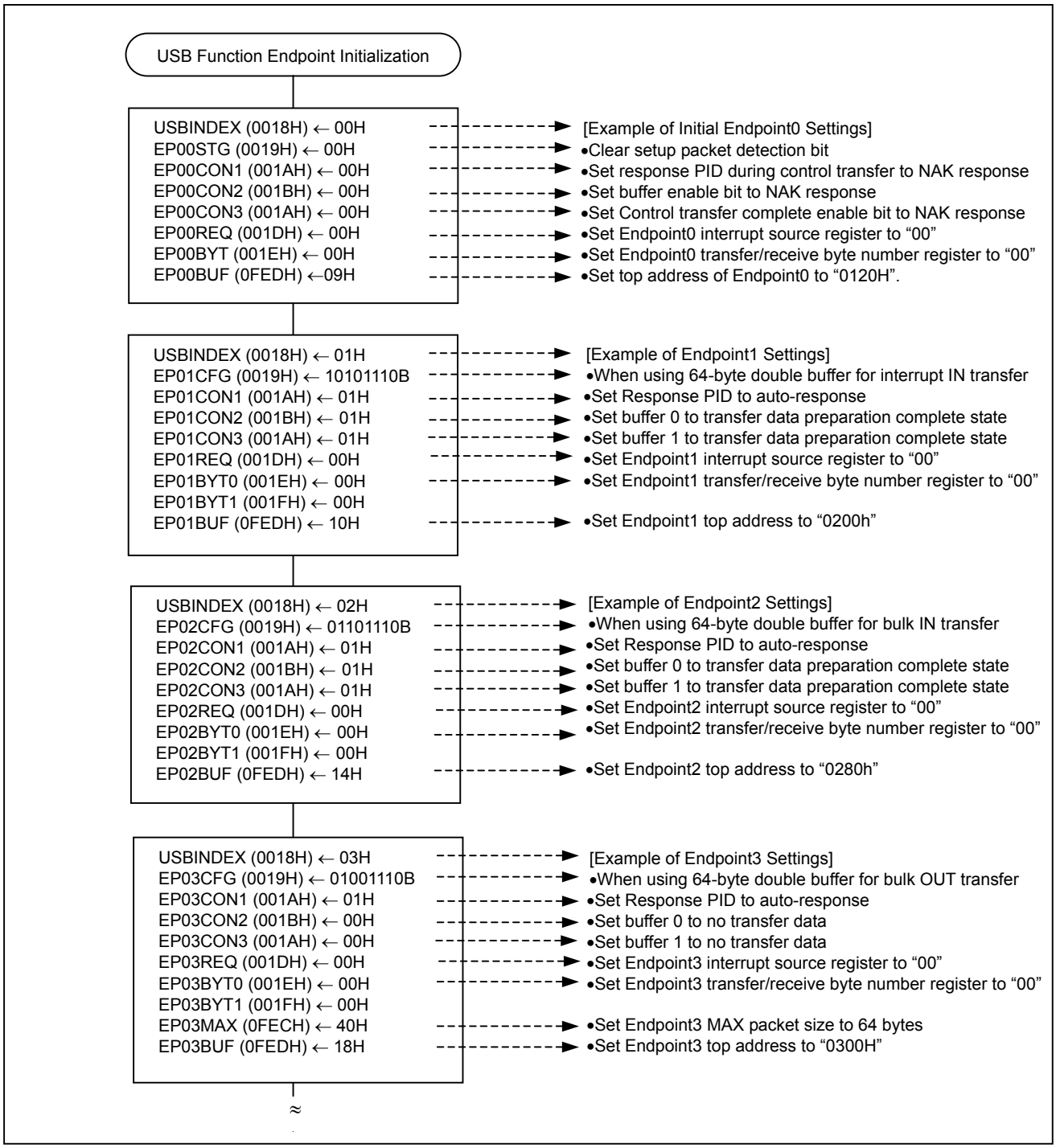


Figure 5.4 Example of USB Function Endpoint Initialization.

5.4. USB Suspend

This section explains an example of the suspend process for the Device (USB Hub + USB Function).

●**Receiving a USB suspend interrupt**

The USB Function moves to the suspended state in the same manner as the USB Hub, enabling it to operate in the low-power dissipation mode. Refer to [2.4.2 USB Suspend Interrupt] for the same process description according to the USB Hub.

To perform a resume in the suspended state using a device-side factor, send a remote wake-up signal and trigger a resume request to the Host. If the device supports the remote wake-up function and the function is enabled by the Host, the Host then enables an interrupt with a wake-up condition, in accordance to user system specifications. The same process is described in detail for the USB Hub in [2.4.3 Remote Wake-up].

●**Receiving a SetPortFeature (PORT_SUSPEND) request**

The Host issues the Hub SetPortFeature (PORT_SUSPEND) request to move the device connected to the USB Hub down-port to the suspended state.

When the 38K2 receives this request for the USB Function (internal down-port), the request is treated as the down-port status.

5.5. USB Resume

This section explains an example of the resume process for the Device (USB Hub + USB Function).

●**Receiving a USB resume interrupt**

In the suspended state, the USB Function is treated as a compound device and is released from the suspended state simultaneously with the USB Hub, then returned to the pre-suspend state. For more details, refer to the USB Hub process described in [2.4.3 USB Resume Interrupt].

●**Receiving a ClearSetPortFeature (PORT_SUSPEND) request**

The Host issues the Hub ClearPortFeature (PORT_SUSPEND) request to move a device connected to the USB hub down-port to the resume state.

When the 38K2 receives this request for the USB Function (internal down-port), the request is treated as the down-port status.

5.6. USB Function/Endpoint Interrupts

The interrupts available for Endpointx are as follows.

♦Interrupts generated when Endpoint0 data transfer/receive is complete
1. Endpoint0 data buffer ready interrupt
2. Control transfer complete interrupt
3. Status stage transition interrupt
4. Endpoint0 SETUP buffer ready interrupt
5. Error interrupt

♦Interrupts generated when Endpointx*1 data transfer/receive is complete
1. Endpointx Data Buffer 0 ready interrupt
2. Endpointx Data Buffer 1 ready interrupt
3. Error interrupt

♣ Conditions for accepting a USB Function/endpointx interrupt

A USB Function/endpointx interrupt will be generated when all of the following conditions are fulfilled:
- I Flag (Interrupt Disable Flag) of the processor status register is “0” (interrupt enabled).
- The USB device interrupt enable bit of ICON1 (Interrupt Control Register) is “1” (interrupt enabled).
- The USB device interrupt source bit of IREQ1 (Interrupt Request Register) is “1” (interrupt requested)
- EP0xE (USB Function/Endpointx Interrupt Enable Bit) which corresponds to the USBICON (Interrupt Factor Enable Register) endpoint is “1” (enabled state).
- EP0x (USB Function/Endpointx Interrupt Bit) which corresponds to the USBIREQ (Interrupt Factor Register) endpoint is “1” (interrupt requested).

  *This bit is set when at least one bit of EP0x is set to “1”. This cannot be set or cleared by software.
- The interrupt bits that correspond to the endpoint (value specified by USBINDEX (Endpoint Index Register)) and EP0xREQ (EP0x Interrupt Factor Register) are “1” (interrupt requested).

  (When referencing EP0xREQ, set the endpointx that corresponds to USBINDEX.)

♣ USB Function/endpointx interrupt processing

- Refer to [Chapter 3.2.3 Endpoint0 Interrupt] and [Chapter 5.8.6 Endpointx Interrupt] for examples of interrupt processing.
- When a USB Function/endpointx interrupt is accepted, the USB Device Interrupt Request Flag (the root flag) is cleared as soon as the interrupt is accepted. Therefore, there is no need to clear the flags when processing. EP0x of USBIREQ cannot be cleared by software. By clearing the EP0xREQ that corresponds to the endpoint (the value specified in USBINDEX) to “00”, the endpoint will be cleared by hardware.

*1 “x” indicates Endpoints 1, 2, and 3.

5.7. USB SOF Interrupt

The USB SOF interrupt is used for isochronous transfers.

(1) Cause of USB SOF interrupt request

- The USB SOF interrupt request is generated when an SOF packet is received from the host by the USBDCU.
- If an SOF packet is sent from the host PC when the USB SOF interrupt enable bit of ICON1 (Interrupt Control Register) is “1”, a USB SOF interrupt is generated. If the SOF packet is has an error for some reason and cannot be received successfully within 250ns of the frame opening, a USB SOF interrupt will not be generated.

(2) Action when a USB SOF interrupt is detected

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

(3) USB SOF interrupt settings

Set USB SOF interrupt enable bit of ICON1 to “1”.

(4) Conditions for accepting a USB SOF

A USB SOF interrupt will be accepted if all of the following conditions are fulfilled:

- I Flag (Interrupt Disable Flag) of the processor status register is “0” (interrupt enabled).
- The USB SOF interrupt enable bit of ICON1 “1” (enabled state).
- The USB SOF interrupt request bit of IREQ1 is “1” (interrupt requested).

Table 5.1 USB SOF Interrupt Registers

Address	Symbol	Register Name
003CH	IREQ1	Interrupt Request Register 1
003EH	ICON1	Interrupt Control Register 1

5.8. USB Transfers

The 38K2 Group USB Function uses the following endpoints for USB transfer/receive operations: Endpoint0 for control transfers, Endpoints 1 to 3 for interrupt, bulk and isochronous transfers.

5.8.1. Endpoint Setup Register

USB transfer/receive (endpoint) related registers are described in Tables 5.2 to 5.4. When using Endpoint0 or Endpoints 1 to 3 for USBINDEX (Endpoint Index Register), set the values indicated in Figure 3.1.

Table 5.2 Endpoint Index Register

Address	Symbol	Register Name
0018H	USBINDEX	Endpoint Index Register

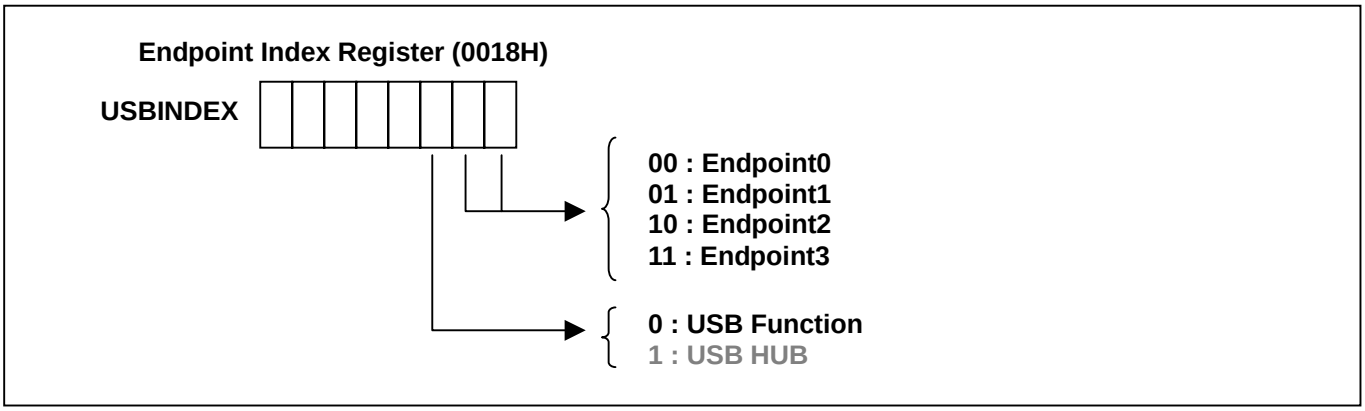


Figure 5.5 Endpoint Index Register Settings

Table 5.3 shows the registers required for setting Endpoint0 (control transfers).

The maximum packet size for control transfers is always set at 8 bytes. Set the start address of the Endpoint0 buffer in EP00BUF (EP00 Buffer Area Setup Register). The buffer consists of 8 bytes each of control command buffer and data buffer. For more details refer to [Chapter 3.2.2 Endpoint0 Buffer Area Setup].

When setting/referencing Endpoint0 registers, set “00” in USBINDEX before accessing the registers.

Table 5.3 USB Endpoint0 Registers

Address	Symbol	Register Name
0019H	EP00STG	EP00 Status Register
001AH	EP00CON1	EP00 Control Register 1
001BH	EP00CON2	EP00 Control Register 2
001CH	EP00CON3	EP00 Control Register 3
001DH	EP00REQ	EP00 Interrupt Factor Register
001EH	EP00BYT	EP00 Transfer/Receive Byte No. Register
001FH	-	-
0FECH	-	-
0FEDH	EP00BUF	EP00 Buffer Area Setup Register

Registers necessary for setting Endpoints 1 to 3 (IN/OUT interrupt, bulk and isochronous transfers) are listed in Table 5.4.

Perform the transfer/receive settings with the EP0xCFG (EP0x Setup Register) in accordance with the descriptor settings of Endpoints 1 to 3. The buffer has 2 settings: single buffer mode and double buffer mode (alternating transfer of Data0 and Data1). Set the start address of Buffer 0 with EP0xBUF (EP0x Buffer Area Setup Register). Set the start address of Buffer 1 bit with BSIZ01 (Double Buffer Start address Setup Bit of EP0xCFG) as the relative address. Refer to [Chapter 5.8.5 Endpoints 1 to 3 Buffer Area Settings] for more details.

Use EP0xCON1 (EP0x Control Register 1) to set responses for transfer/receive transmissions. Use EP0xCON2 and EP0xCON3 (EP0x Control Registers 2 and 3) to enable the buffers.

When setting/referencing an endpointx register, access the endpointx register after setting the endpoint value in USBINDEX (Endpoint Index Register).

Table 5.4 USB Endpointx Registers

Address	Symbol	Register Name
0019H	EP0xCFG	EP0x Setup Register
001AH	EP0xCON1	EP0x Control Register 1
001BH	EP0xCON2	EP0x Control Register 2
001CH	EP0xCON3	EP0x Control Register 3
001DH	EP0xREQ	EP0x Interrupt Factor Register
001EH	EP0xBYT0	EP0x Transfer/Receive Byte No. Register 0
001FH	EP0xBYT1	EP0x Transfer/Receive Byte No. Register 1
0FECH	EP0xMAX	EP0x MAX Packet Size Register
0FEDH	EP0xBUF	EP0x Buffer Area Setup Register

5.8.2. Endpoint0

The Endpoint0 process of the USB Function is the same as that of the USB Hub. Refer to [3.2 Endpoint0] for details. However, register names and bit symbols reference different fields, accordingly. For example, when reading the material for the USB Function, [EP10 Buffer Area Setup Register (EP10BUF)] for buffer setup becomes [EP00 Buffer Area Setup Register (EP00BUF)] and “10” becomes “00”.

5.8.3. Endpoints 1-3

Endpoints 1 to 3 manage three transfer types: interrupt, bulk, and isochronous. Each endpoint can handle bi-directional transfers: OUT (for receiving) and IN (for sending).

The user needs to assign data buffers in the RAM for Endpoints 1 to 3. Specify the start address of each data buffer with EP0xBUF (EP0x Buffer Area Setup Register). For more details, refer to [Chapter 5.8.5 Endpoint1-3 Buffer Area].

In order to configure the IN/OUT settings of Endpoints 1 to 3, the user must specify the transfer direction, transfer type, and MAX packet size for each endpoint descriptor. The transfer direction and type can be set with EP0xCFG (EP0x Set Register). The MAX packet size can be set with EP0xMAX (EP0x MAX Packet Size Register).

USB Endpointx interrupts (three factors) will occur when Endpointx is in any type of transfer operation. For more details, refer to [Chapter 5.8.6 Endpointx Interrupts].

5.8.4. Data Transfer Format

The USB transfer uses a half-duplex transfer and consists of at least two packets (token packet and data packet) per transfer. Transfers that support the retry function, guaranteeing valid data transfer between the Host and the device, consist of three packets: a handshake packet following the token and data packets. The three types of transfer supported by the USB Function offer distinct characteristics. Isochronous transfer, which consists of 2 packets, does not detect a transfer error but does guarantee the transfer rate. Bulk transfer, which consists of 3 packets, detects a transfer error but does not guarantee the transfer rate. Interrupt transfer, which consists of 3 packets, detects a transfer error and guarantees the transfer rate.

♦ Data Transfer Transactions

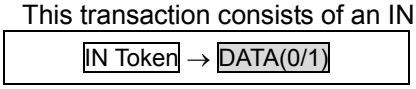
The following types of transactions classify data transfer*1. The shaded blocks indicate a packet transmitted by a device.

(1) Bulk IN and Interrupt IN:

This transaction consists of an IN token packet, data packets (DATA0/1), and a handshake packet.

(2) Isochronous IN:

This transaction consists of an IN token packet and a data packet (DATA0/1).

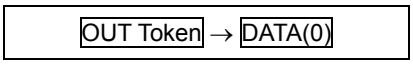


(3) Bulk OUT and Interrupt OUT:

This transaction consists of an OUT token packet, data packets (DATA0/1), and a handshake packet.

(4) Isochronous OUT

This transaction consists of an OUT token packet and a data packet (DATA0).



*1 (1), (2), (3) and (4) are examples in normal transfesr.

♦ Transactions according to Data Transfer Type
The following types of transactions are used to perform transfer data*[1].

(1) Bulk IN and Interrupt IN Transfers:

IN(DATA0) IN(DATA1) IN(DATA0) IN(DATA1) ---- IN(DATA0/1)

(2) Isochronous IN Transfer:

IN(DATA0) IN(DATA1) IN(DATA0) IN(DATA1) ---- IN(DATA0/1)

(3) Bulk OUT and Interrupt OUT Transfers:

OUT(DATA0) OUT(DATA1) OUT(DATA0) OUT(DATA1) ---- OUT(DATA0/1)

(4) Isochronous OUT Transfers:

OUT(DATA0) OUT(DATA0) OUT(DATA0) OUT(DATA0) ---- OUT(DATA0)

5.8.5. Buffer Area Setup for Endpoints 1– 3

The user needs to assign a data buffer in the RAM area for each endpoint in order to perform data transfer.

Endpoints 1 to 3 have two buffer modes: single buffer mode and double buffer mode. The buffer mode is selected with DBLB0x (Buffer Mode Setup Bit) of EP01CFG (EP0x Setup Register).

The data buffer area of each respective endpoint to be used must be assigned in the RAM area and the size (number of bytes) of the buffer must be larger than the MAX packet size of the respective endpoint descriptor.

The data buffer size of each endpoint, both IN and OUT, must be MAX 64 bytes (Single Buffer Mode).

[1] Single Buffer Mode

In the single buffer mode, Data Buffer 0 handles both DATA0 and DATA1, which are transferred alternately.

The Data Buffer 0 area of each respective endpoint to be used must be assigned in the RAM area and the size of the buffer must be larger than the MAX packet size of the respective endpoint descriptor. Figure 3.11 shows how to assign the Data Buffer 0 area in the RAM by setting the start address of the buffer in EP0xBUF (EP0x Buffer Area Setup Register). Figure 3.12 shows a setup example. The actual start address of the Data Buffer 0 area is 20H times the contents of EP0xBUF. Figure 3.13 shows an access model of the buffer.

*[1] (1), (2), (3) and (4) are examples in normal transfers with Handshake omitted.

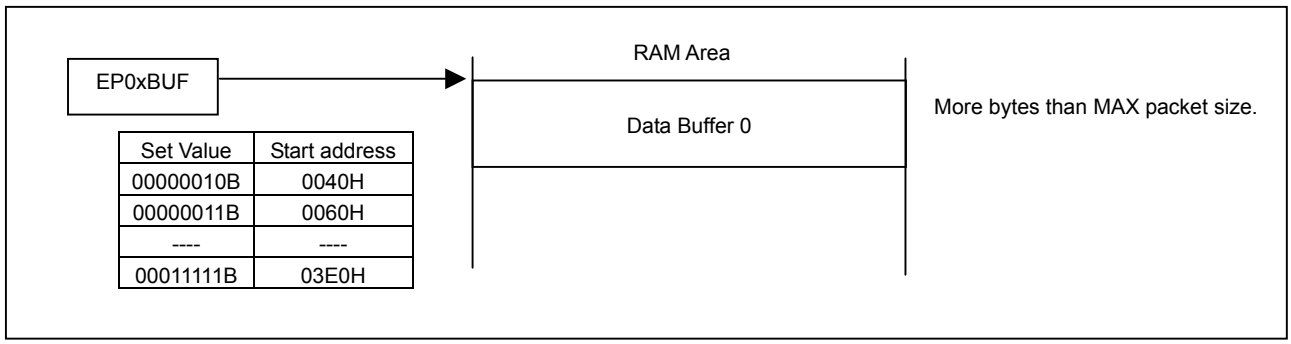


Figure 5.6 Endpointx Data Buffer 0

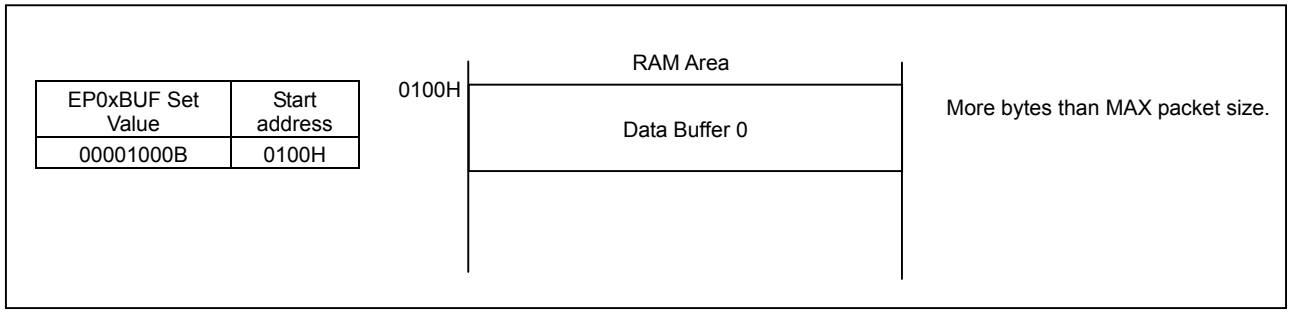


Figure 5.7 Example of Endpointx Data Buffer 0 Setting

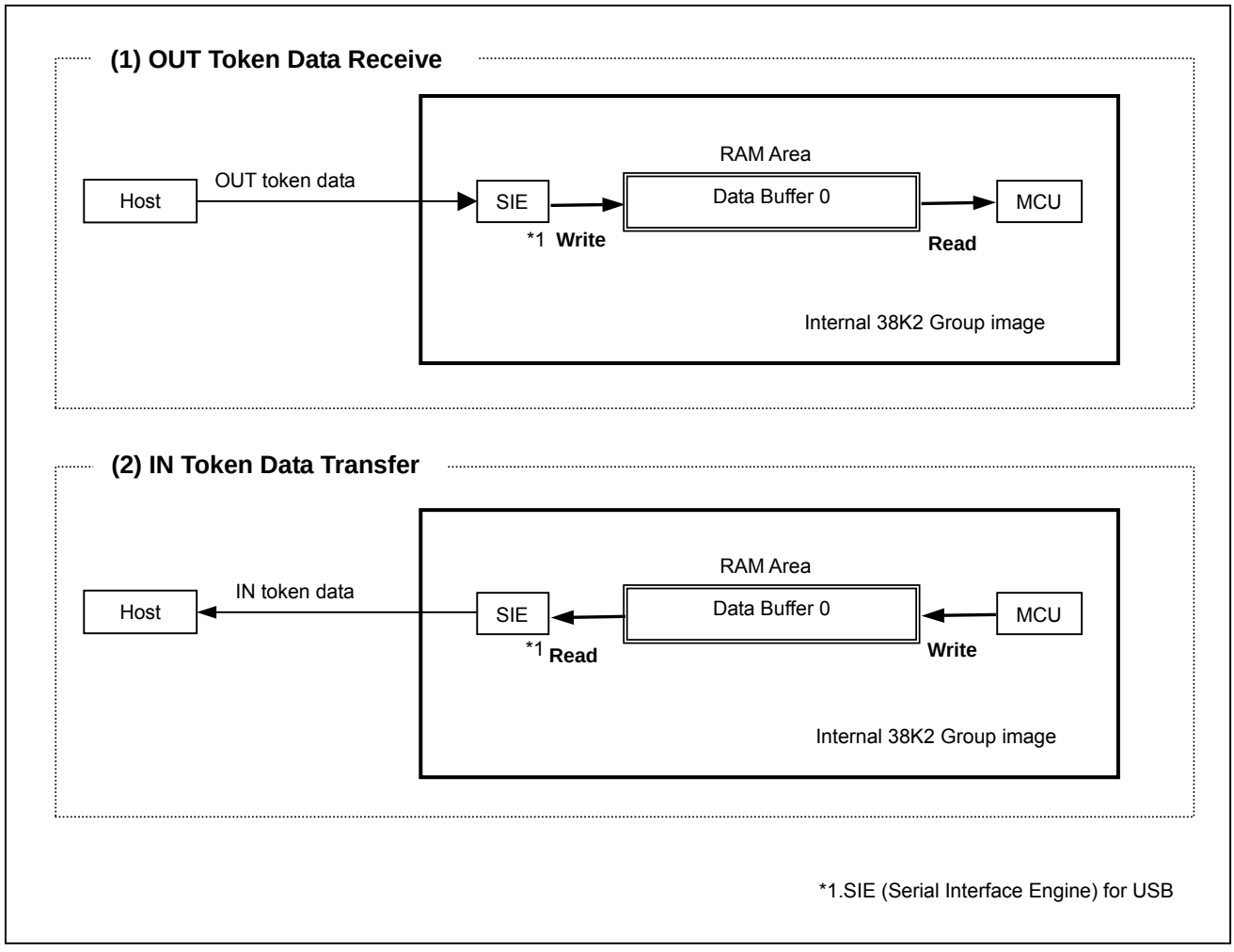


Figure 5.8 Endpointx Data Buffer 0 Access

[2] Double Buffer mode

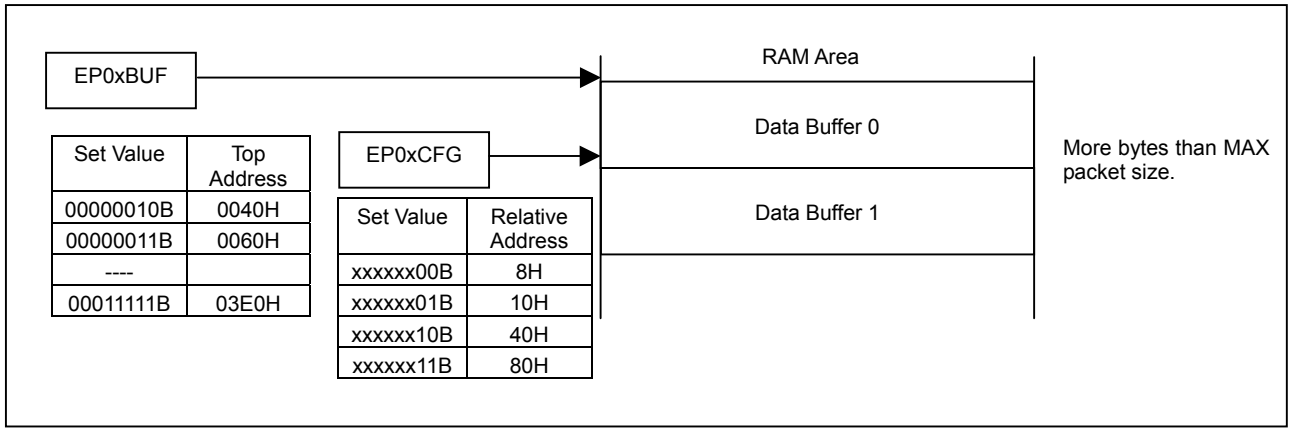
In the double buffer mode, data transfers are handled separately: DATA0 is transferred to Data Buffer 0 and DATA1 is transferred to Data Buffer 1, alternately.

Make sure the areas reserved for Data Buffer 0 and Data Buffer 1 on the RAM are larger than the MAX packet size of the descriptor for the endpoint to be used.

Set the Data Buffer 0 start address in EP0xBUF (EP0x Buffer Area Setup Register), as shown in Figure 5.9, and reserve the area on the RAM. Set the Data Buffer 1 start address in the Double Buffer Address Setup Bit of EP0xCFG (EP0x Buffer Area Setup Register), also shown in Figure 5.9, and reserve the area on the RAM. (See example in Figure 5.10.)

The start address of Data Buffer 0 is 20H times the value set in EP0xBUF. The start address of Data Buffer 1 is 20H times the value set in EP0xBUF plus the value set in EP0xCFG.

See Figure 5.11 for an image of the buffer access.

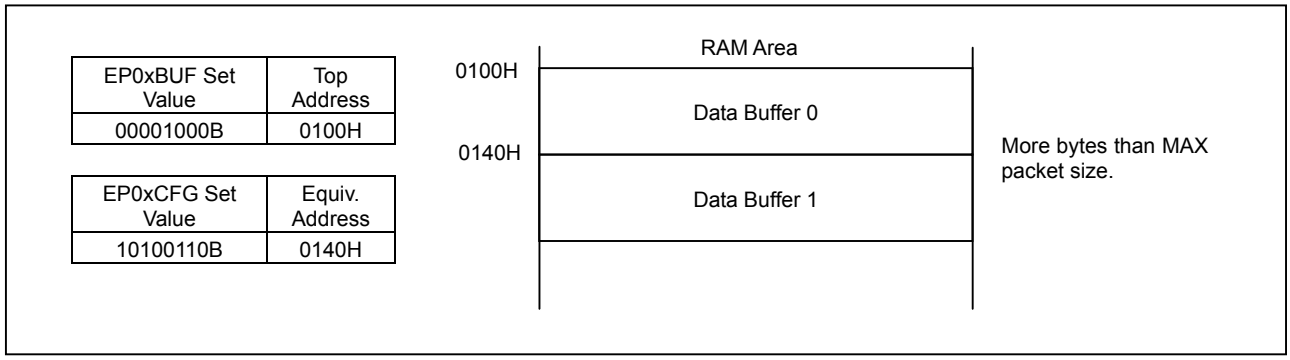


Set Value	Top Address
00000010B	0040H
00000011B	0060H

00011111B	03E0H

Set Value	Relative Address
xxxxxx00B	8H
xxxxxx01B	10H
xxxxxx10B	40H
xxxxxx11B	80H

Figure 5.9 Endpointx Data Buffer 0/1



EP0xBUF Set Value	Top Address
00001000B	0100H

EP0xCFG Set Value	Equiv. Address
10100110B	0140H

Figure 5.10 Example of Endpointx Data Buffer 0/1 Settings

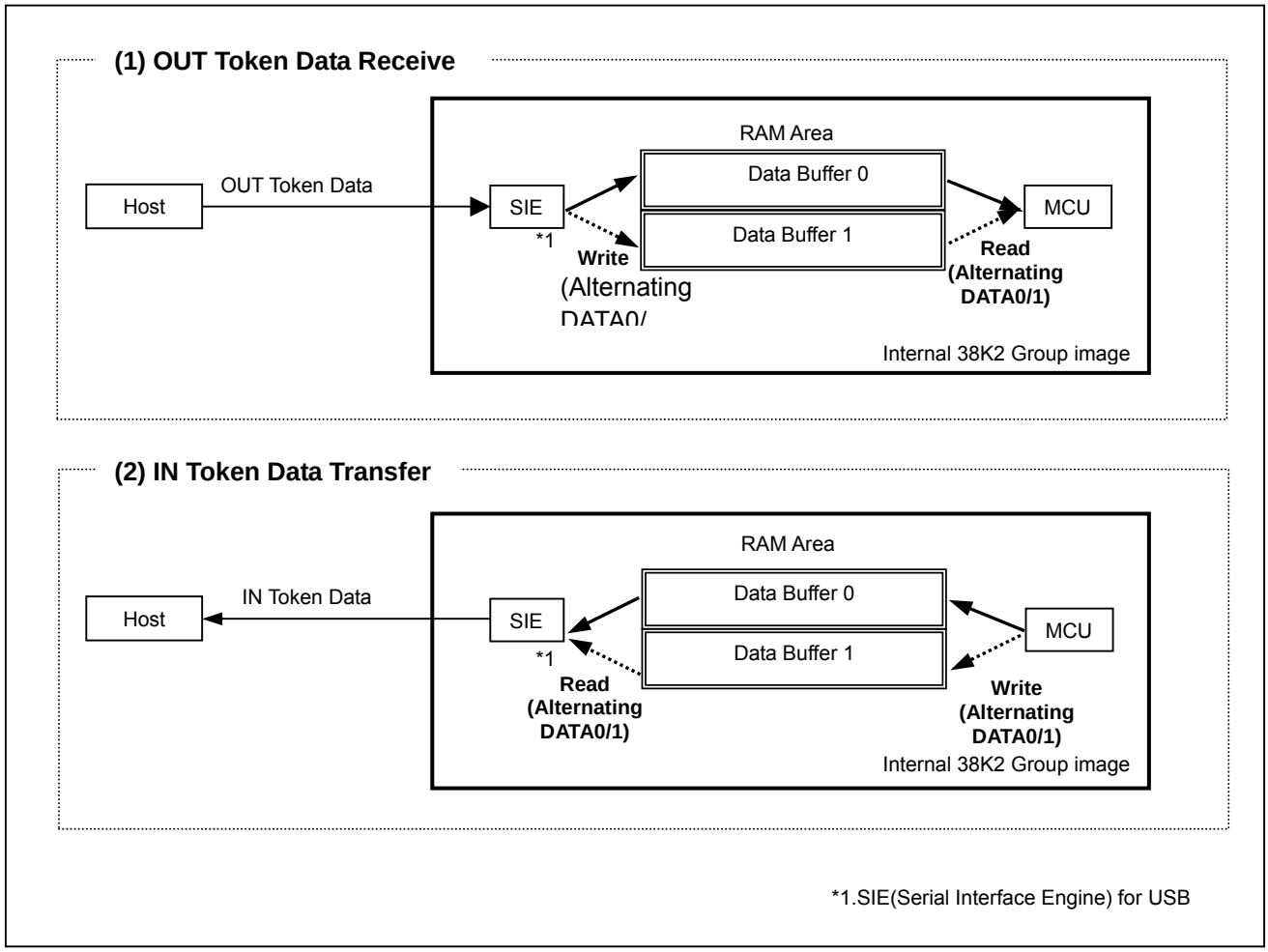


Figure 5.11 Endpointx Data Buffer 0/1 Access

5.8.6. Endpointx Interrupt

The Endpointx interrupt is used to control data flow during all three types of transfers: interrupt, bulk, and isochronous. Endpointx Interrupt Factor Register is shown in Figure 5.12.

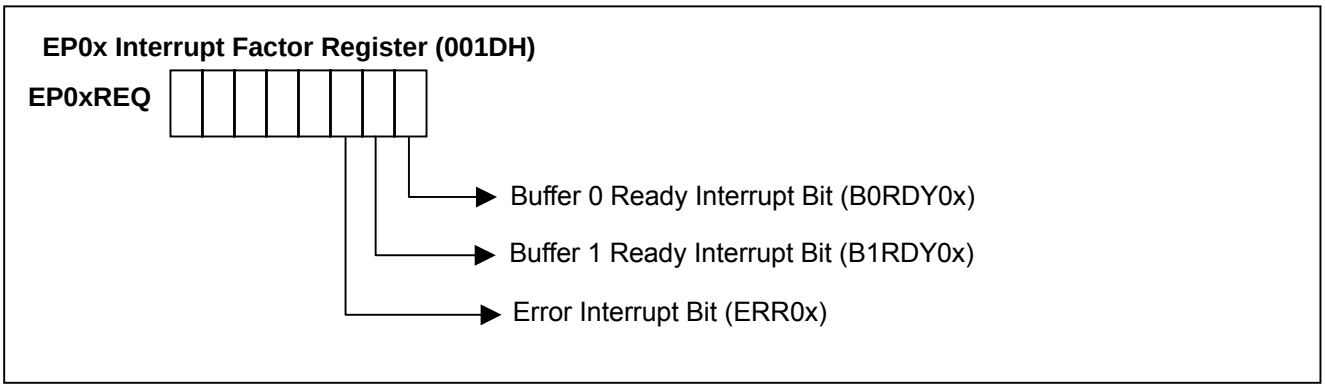


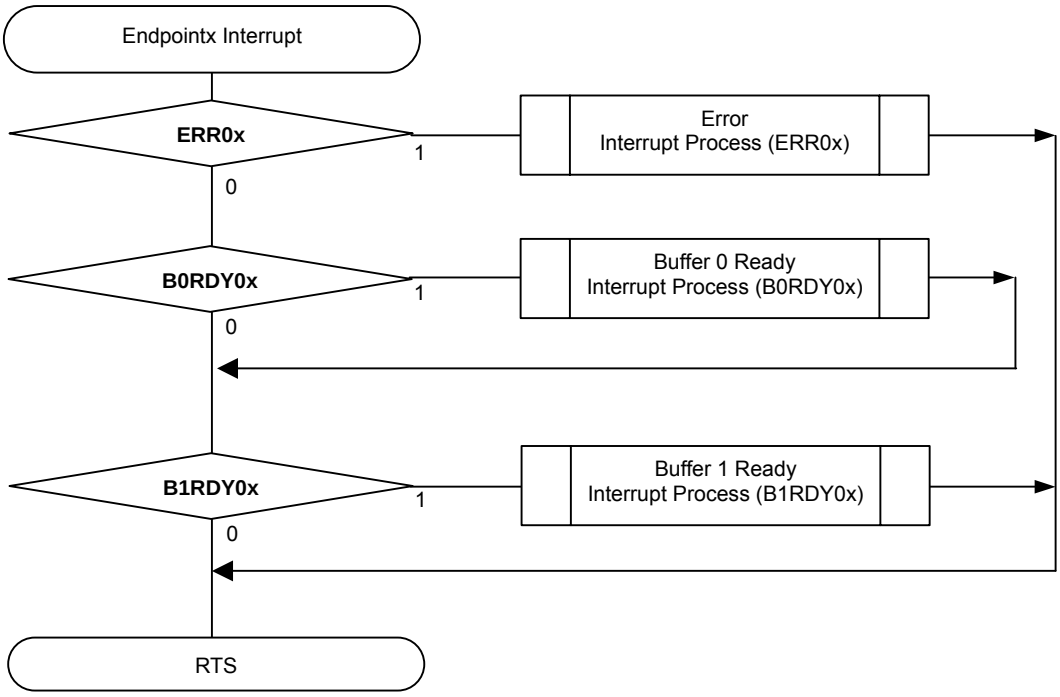
Figure 5.12 EP0x Interrupt Factor Register (EP0xREQ)

(1) Buffer 0 Ready Interrupt Bit (B0RDY0x)
When Buffer 0 is in the ready state (read/write enabled state), this bit goes to “1” (set by H/W).
This bit must be cleared by F/W when an interrupt occurs. (This bit not cleared to “0” automatically by H/W.)

(2) Buffer 1 Ready Interrupt Bit (B1RDY0x)
In the double buffer mode, when Buffer 1 is in the ready state (read/write enabled state) this bit goes to “1” (set by H/W). (This bit is not valid in the single buffer mode.)
This bit must be cleared by F/W when an interrupt occurs. (This bit not cleared to “0” automatically by H/W.)

(3) Error Interrupt Bit (ERR0x)
This bit is set under the following conditions:
•When a MAX packet error occurs during a receive (OUT) operation.
→The response PID is automatically set to STALL handshake (by H/W).
•When an error occurs during an isochronous transfer.

An example of Endpointx interrupt processing is shown in Figure 5.13. Determine the interrupt factor at each stage and then perform the error process, receive/transfer setup and buffer read/write



Note 1: Always determine the interrupt factor bit in the order indicated above.
Note 2: This routine is a subroutine called in [Figurer 2.3 USB Device Interrupt Routine Example].

Figure 5.13 Example of Endpointx Interrupt Processing

5.8.7. Data Receive Format

(1) Receive Setup

Configure Endpointx by specifying the transfer direction, transfer type, and MAX packet size for the corresponding endpoint descriptor and matching the endpoint number with the descriptor.

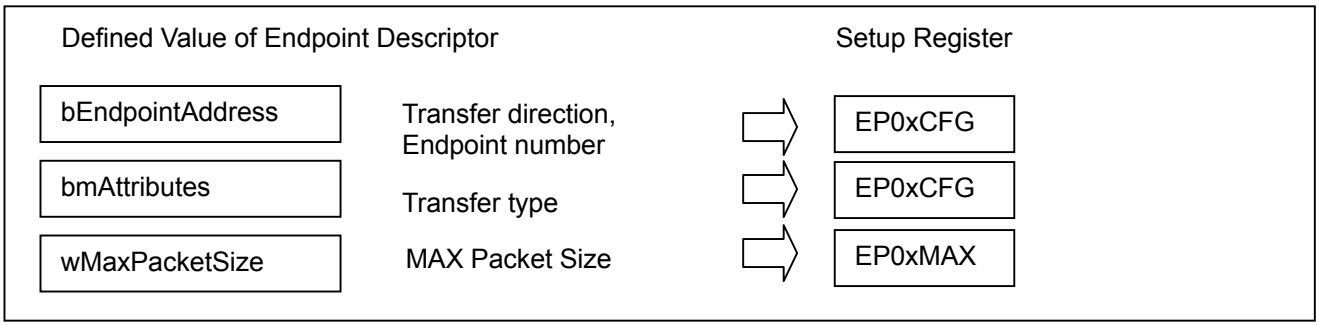


Figure 5.14 Endpoint Descriptor

Figure 5.15 shows an example of an Endpointx receive setup procedure.

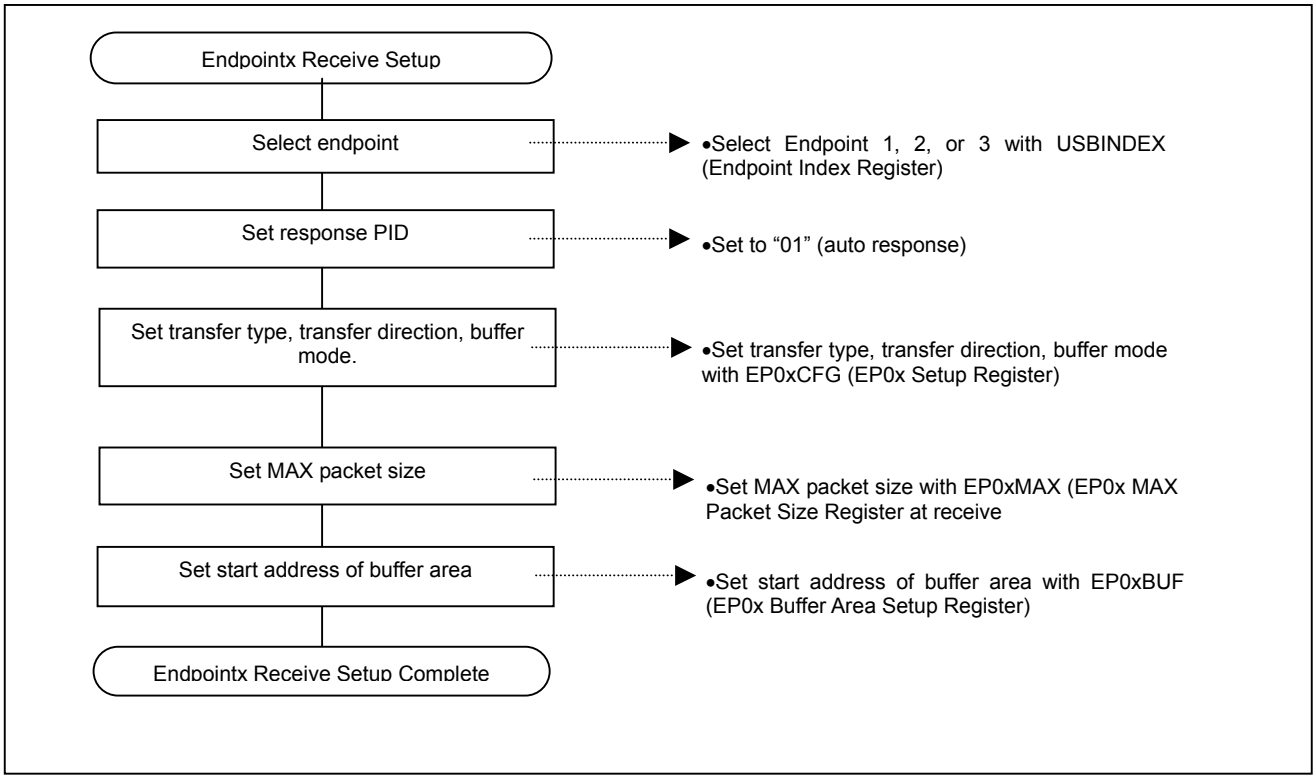


Figure 5.15 Example of Endpointx Receive Setup Procedure

(2) Normal Receive

[1] Single Buffer Mode

In the single buffer mode, the Endpointx Buffer Ready interrupt is generated when Endpointx receives a valid packet from the Host. The number of bytes of data received is automatically set in EP0xBYT0 (EP0x Byte Number Register 0) by hardware. Read the same number of bytes of data from Data Buffer 0 as received by the endpoint, then set BOVAL0x to “1”. (The B0VAL0x bit is automatically cleared by H/W in normal receive transmissions.)

[2] Double Buffer Mode

When Endpointx receives a valid packet from the Host in the double buffer mode, if the packet is DATA0, the Endpointx Buffer 0 Ready interrupt is generated. The number of bytes of data received is automatically set in EP0xBYT0 by hardware. Read the same number of bytes of data from Data Buffer 0 as received by the endpoint, then set B0VAL0x (Buffer 0 Enable Bit) of EP0xCON2 to “1”. If the packet is DATA1, the Endpointx Buffer 1 Ready Interrupt is generated. The number of bytes of data received is automatically set in EP0xBYT0 by hardware. Read the same number of bytes of data from Data Buffer 1, then set B1VAL0x (Buffer 1 Enable Bit) of EP0xCON3 to “1”. (B0VAL0x/B1VAL0x bits are automatically cleared by H/W in normal receive transmissions.)

(3) Receive Error (when ERR0x interrupt occurs)

When Endpointx receives a packet of data from the Host which is larger than the MAX OUT packet size set in EP0xMAX (EP0x MAX Packet Size Register), a MAX packet error occurs, no handshake is returned, and an ERR0x interrupt is generated. At this time, PID0x (Response PID Bit) of EP0xCON1 is automatically set to “1x” (STALL handshake) by H/W. If a MAX packet error occurs at the next OUT token as well, the same sequence is repeated (no handshake is returned and an ERR0x interrupt is generated). For any other OUT (non-MAX packet error) or IN tokens, a STALL handshake is returned. When a STALL is sent, communications are disabled unless the Host CPU cancels the STALL with the ClearFeature. Even when communications are restarted, the STALL is not cancelled unless the PID0x bit is set to “00” or “01”.

When an error occurs in an isochronous transfer, an ERR0x interrupt is generated regardless of any IN/OUT packets received.

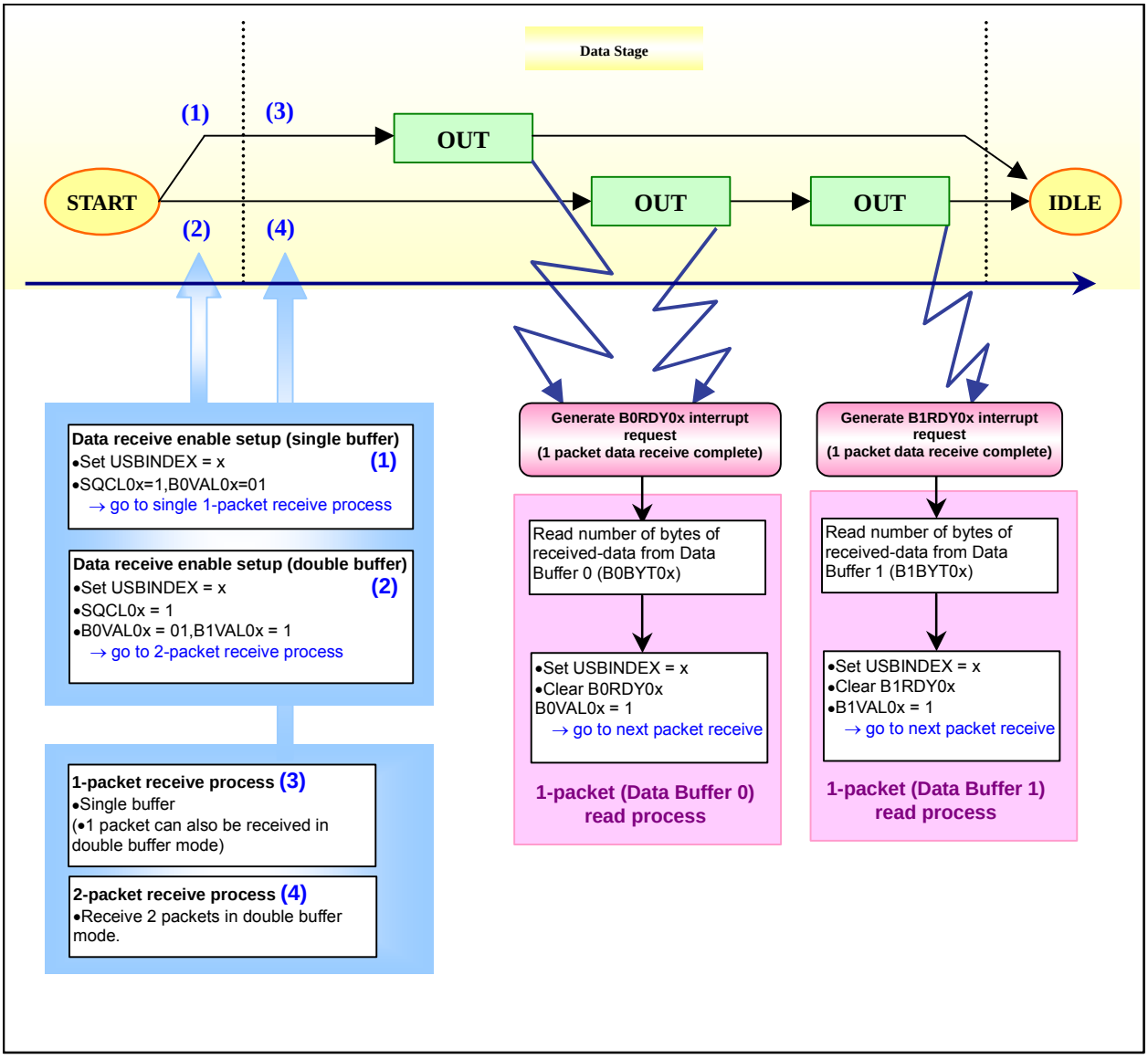


Figure 5.16 Example of Endpointx Data Receive

5.8.8. Bulk Transfer Data Receive

(1) Transfer Type Setup

To use Endpointx OUT for bulk transfers, set the endpoint to bulk transfer with EP0xCFG (EP0x Setup Register). Refer to the data receive setup procedure in [Figure 5.15. Example of Endpointx Receive Setup Procedure].

(2) Data Receive Ready

Set SQCL0x (Sequence Toggle Bit Clear Bit) to "1". By setting this bit to "1", the toggle bit is cleared and the next data PID is initialized to DATA0.

Next, set B0VAL0x (Buffer 0 Enable Bit) to "1" to put the bit in the data receive-enabled state. In the double buffer mode, B1VAL0x (Buffer 1 Enable Bit) also needs to be set to "1" to enable DATA1 receive.

(3) Data Receive Operation

An example of Endpointx bulk data receive is shown in Figure 5.17.

The bulk OUT transfer, which sends data from the Host CPU to the device, repeats the OUT transaction as many times as necessary. When Endpointx receives a valid packet from the Host, the Endpointx Buffer 0 Ready interrupt or Endpointx Buffer 1 Ready interrupt (in double buffer mode only) is generated. In the handshake phase of each transaction, if the transmit-side successfully receives the ACK packet issued by the receive-side, DATA0 and DATA1 of the data packet in the next data phase are toggled.

However, if the ACK is not received successfully, the following response is generated.

- If the received data includes an error or the MAX packet error is generated, no response is issued.
- If the endpoint is stalled, a STALL handshake is returned.
  (When PID0x (Response PID Bit) is "1x", the endpoint is stalled.)
- If there is a sequence bit mismatch in the received data, an ACK packet is returned.
- If Data Buffer 0 or 1 is full, a NAK packet is returned.

If the Host sends two packets of data from the same toggle, an ACK is returned for both packets, but the second data packet will not be stored in the data buffer. In this case, the ACK returned to the Host for the first packet is dropped and the Host assumes that the same data was sent in both packets.
Error processing covers error checks in accordance with USB Specification Ver.2.0, such as CRC check and bit stuffing. When an error is detected in the bulk OUT transfer, it is ignored. All error processing is performed by hardware and does not need to be controlled by software.

The following are details for operations in single and double buffer modes.

[1] Single Buffer Mode

When Endpointx receives a valid packet from the Host, the Endpointx Buffer 0 Ready interrupt is generated. The number of bytes of received data is automatically set in EP0xBYT0 (EP0x Byte Number Register 0) by hardware. Read the same number of bytes from Data Buffer 0, then set B0VAL0x (Buffer 0 Enable Bit) of EP0xCON2 (EP0x Control Register 2) to "1". (B0VAL0x bit is automatically cleared by hardware in normal receive transmissions.)

When the data is received in the following order, data is read out from Data Buffer 0 alternately. DATA0 → DATA1 → DATA0…

[2] Double Buffer Mode

When Endpointx receives a valid DATA0 packet from the Host in the double buffer mode, the Endpointx Buffer 0 Ready interrupt is generated. The number of bytes of received data is automatically set in EP0xBYT0 by hardware. Read the same number of bytes from Data Buffer 0, then set B0VAL0x of EP0xCON2 to "1". If the received packet is DATA1, the Endpointx Buffer 1 Ready Interrupt is generated. The number of bytes of received data is automatically set in EP0xBYT1 (EP0x Byte Number Register 1) by hardware. Read the same number of bytes from Data Buffer 1, then set B1VAL0x (Buffer 1 Enable Bit) of EP0xCON3 (EP0x Control Register 3) to "1". (B0VAL0x and B1VAL0x bits are automatically cleared by hardware in normal receive operations.)

When the data is received in the following order, DATA0 is read out from Data Buffer 0 and DATA1 is read out from Data Buffer 1 alternately. DATA0 → DATA1 → DATA0…

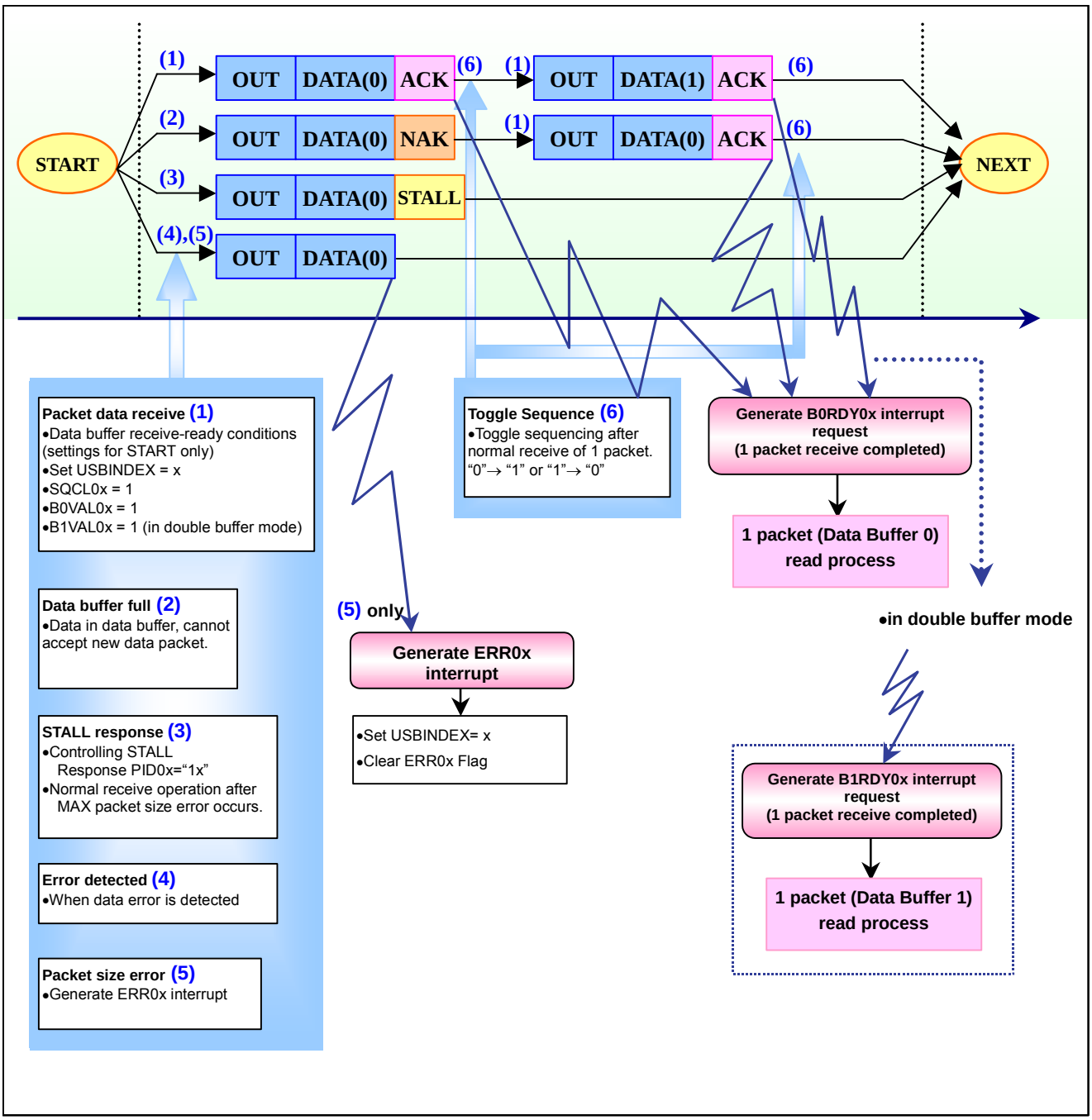


Figure 5.17 Example of Endpointx Bulk Receive

• Example of S/W Process in Double Buffer Mode

As shown in Figure 5.18, when using the double buffer mode, if an interrupt request is generated for the first packet but the interrupt is not processed (the buffer is not read) due to current execution of another process, both interrupt requests, B0RDY0x and B1RDY0x, are generated when an interrupt request is generated for the second packet.

At this time, when processing both interrupt requests, the user needs to toggle DATA0 and DATA1 with software accordingly (DATA0→DATA1 or DATA1→DATA0) so that the data buffers will be read in the appropriate order. See Figure 5.18 for an example of the software processing.

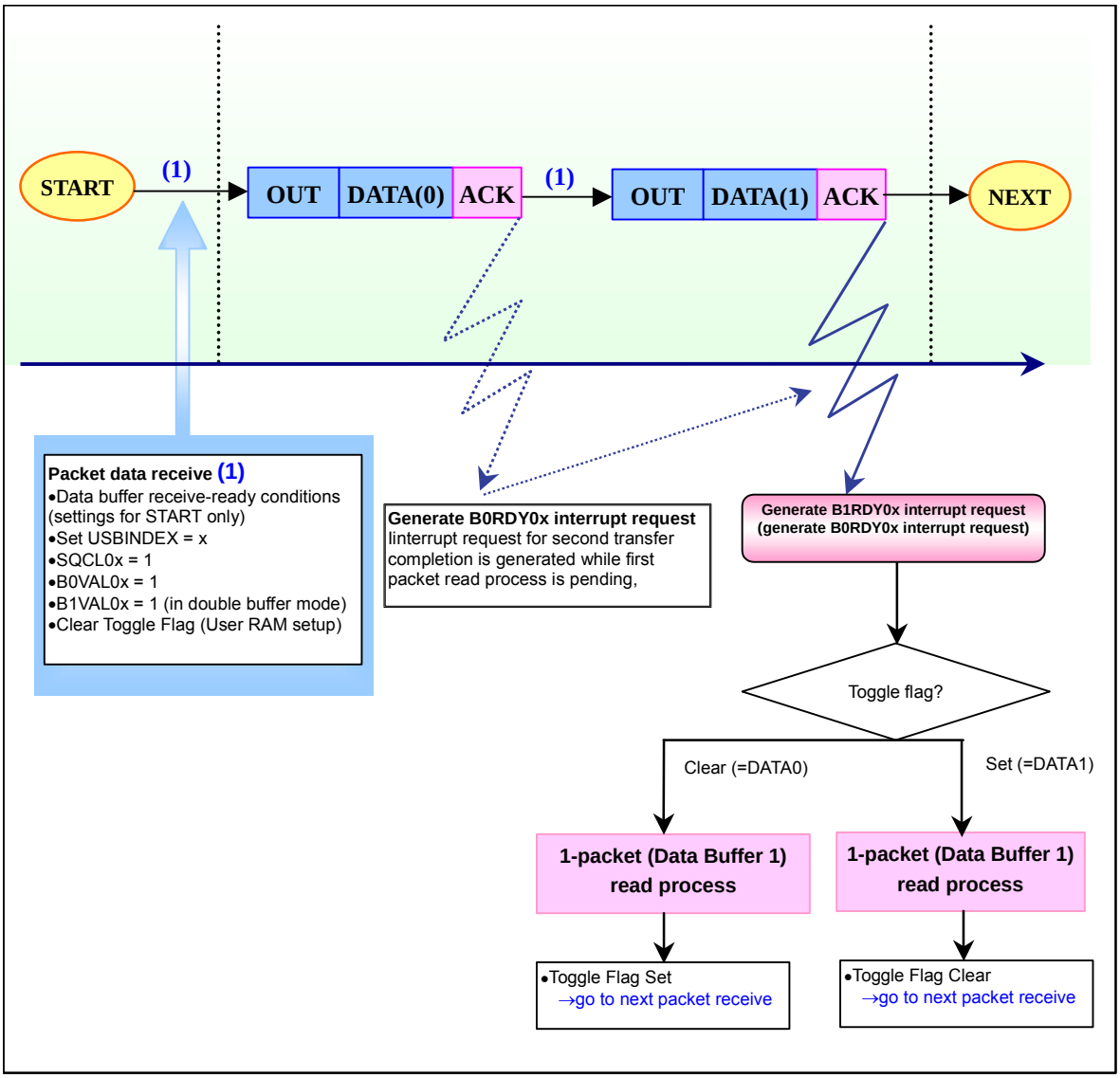


Figure 5.18 Example of Endpointx Bulk Receive (double buffer mode)

5.8.9. Interrupt Transfer Data Receive

(1) Transfer Type Setup

To use Endpointx OUT for interrupt transfers, set the endpoint to interrupt transfer with EP0xCFG (EP0x Setup Register). Refer to the procedure in [Figure 5.15 Example of Endpointx Receive Setup Procedure].

(2) Data Receive Prep

Preparation for interrupt transfer data receive is the same as that for bulk transfers. Refer to [Chapter 5.8.8 Bulk Transfer Data Receive].

(2) Data Receive Operation

Endpointx OUT operation in interrupt transfers is the same as that for bulk transfers. Refer to [Chapter 5.8.8 Bulk Transfer Data Receive].

5.8.10. Isochronous Transfer Data Receive

(1) Transfer Type Setup

To use Endpointx OUT for isochronous transfers, set the endpoint to isochronous transfer with EP0xCFG (EP0x Setup Register). Refer to the procedure in [Figure 5.15 Example of Endpointx Receive Setup Procedure].

(2) Data Receive Prep

Set SQCL0x (Sequence Toggle Bit Clear Bit) to "1". By setting this bit to "1", the toggle bit is cleared and the next data PID is initialized to DATA0.

Next, set B0VAL0x (Buffer 0 Enable Bit) to "1" to put the bit in the data receive-enabled state. In the double buffer mode, B1VAL0x (Buffer 1 Enable Bit) also needs to be set to "1" to enable DATA1 receive.

(3) Data Receive Operation

An example of Endpointx isochronous data receive is shown in Figure 5.19.

The isochronous OUT transfer, which transfers data from the Host CPU to the device, repeats the OUT transaction as many times as necessary. The isochronous transaction does not include a handshake phase (the device does not return an ACK or NAK to the Host). The isochronous transfer only involves data packet DATA0; there is no toggle sequence with DATA1.

When Endpointx receives a valid packet from the Host, an Endpointx Buffer 0 Ready interrupt or Buffer 1 Ready interrupt (in double buffer mode only) is generated.

When there is data in Data Buffer 0/1 and a new data packet cannot be accepted, the ERR0x interrupt (as an overrun error) is generated.

Detailed explanations of single buffer and double buffer mode operations are provided below.

[1] Single Buffer Mode

When Endpointx receives a valid packet from the Host, the Endpointx Buffer 0 Ready interrupt is generated. The number of bytes of received data is automatically set in EP0xBYT0 (EP0x Byte Number Register 0) by hardware. Read the same number of bytes from Data Buffer 0, then set B0VAL0x (Buffer 0 Enable Bit) of EP0xCON2 (EP0x Control Register 2) to "1". (B0VAL0x bit is automatically cleared by hardware in normal receive operations.)

[2] Double Buffer Mode

When Endpointx receives a valid packet from the Host, if the packet begins with an even number (starting from packet number 0), the Endpointx Buffer 0 Ready interrupt is generated. The number of bytes of received data is automatically set in EP0xBYT0 (EP0x Byte Number Register 0) by hardware. Read the same number of bytes from Data Buffer 0, then set B0VAL0x (Buffer 0 Enable Bit) of EP0xCON2 (EP0x Control Register 2) to "1". If the next packet is odd-numbered, the Endpointx Buffer 1 Ready interrupt is generated. The number of bytes of received data is automatically set in EP0xBYT1 (EP0x Byte Number Register 1) by hardware. Read the same number of bytes from Data Buffer 1, then set B1VAL0x (Buffer 1 Enable Bit) of EP0xCON3 (EP0x Control Register 3) to "1". (B0VAL0x and B1VAL0x bits are automatically cleared by hardware in normal receive operations.)

• Example of S/W Process in Double Buffer Mode

As shown in Figure 5.18, when using the double buffer mode, if an interrupt request is generated for the first packet but the interrupt is not processed (the buffer is not read) due to current execution of another process, both interrupt requests, B0RDY0x and B1RDY0x, are generated when an interrupt request is generated for the second packet.

At this time, when processing both interrupt requests, the user needs to manage the order in which Data Buffer 0/1 will be read. Although isochronous transfer OUT operations do not support the toggle sequence, use Figure 5.18 as an example of the software processing.

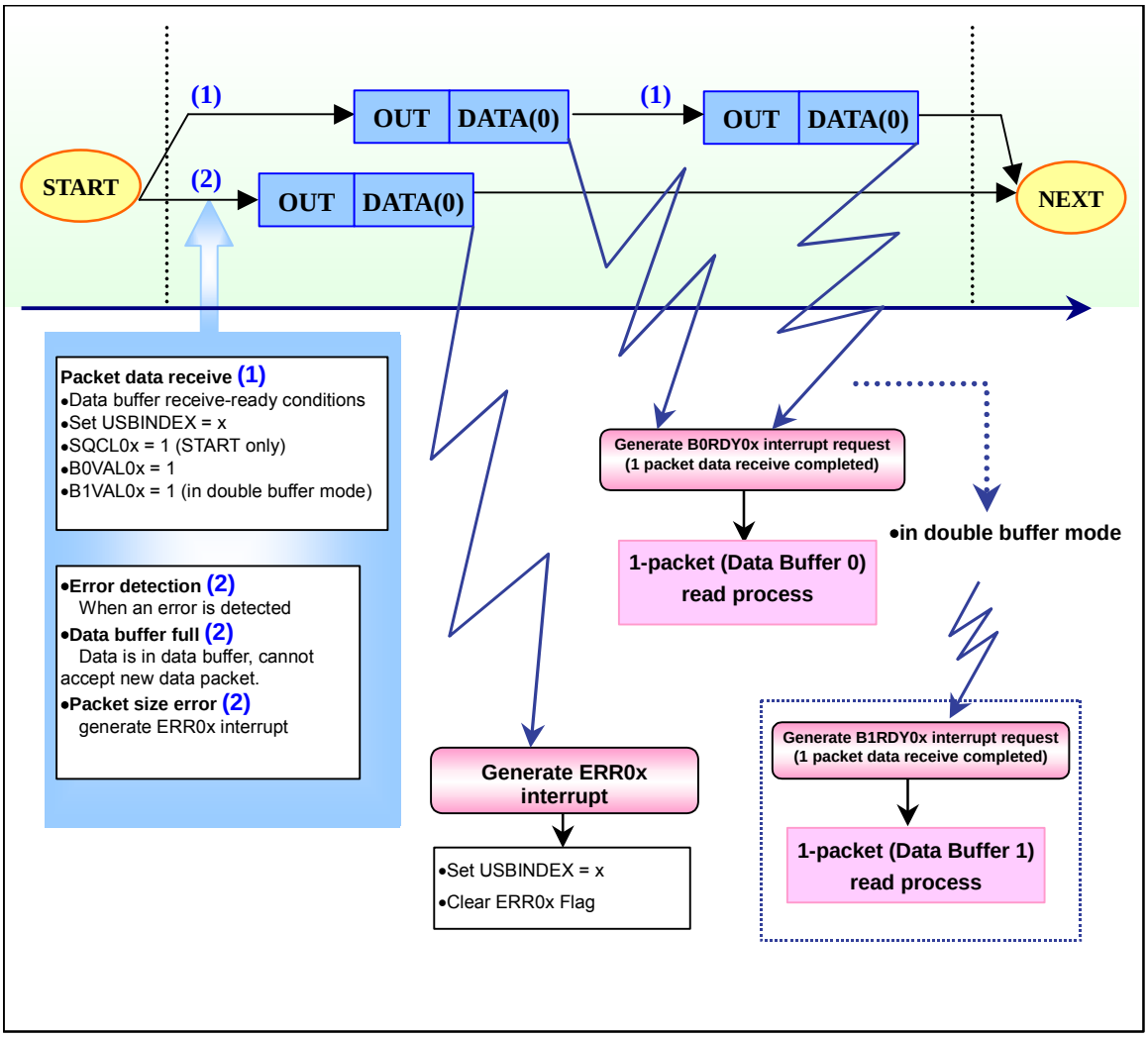


Figure 5.19 Example of Endpointx Isochronous Receive

5.8.11. Data Transmit Format

(1) Transmit Setup

Configure Endpointx by specifying the transfer direction, transfer type for the corresponding endpoint descriptor and matching the endpoint number with the descriptor.

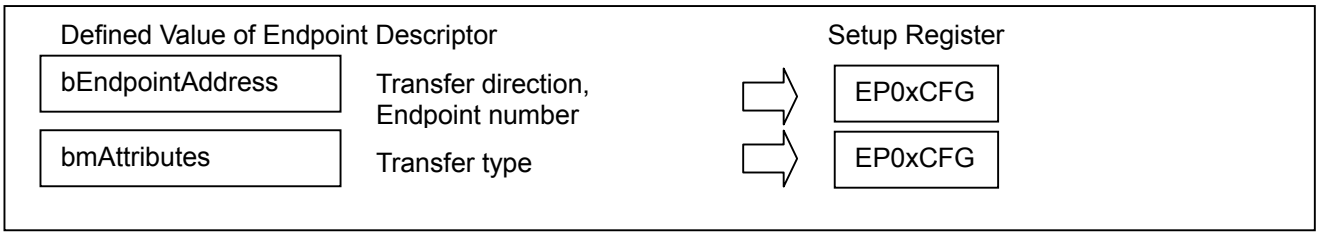


Figure 5.20 Endpoint Descriptor

Figure 5.21 shows an example of an Endpointx transmit setup procedure.

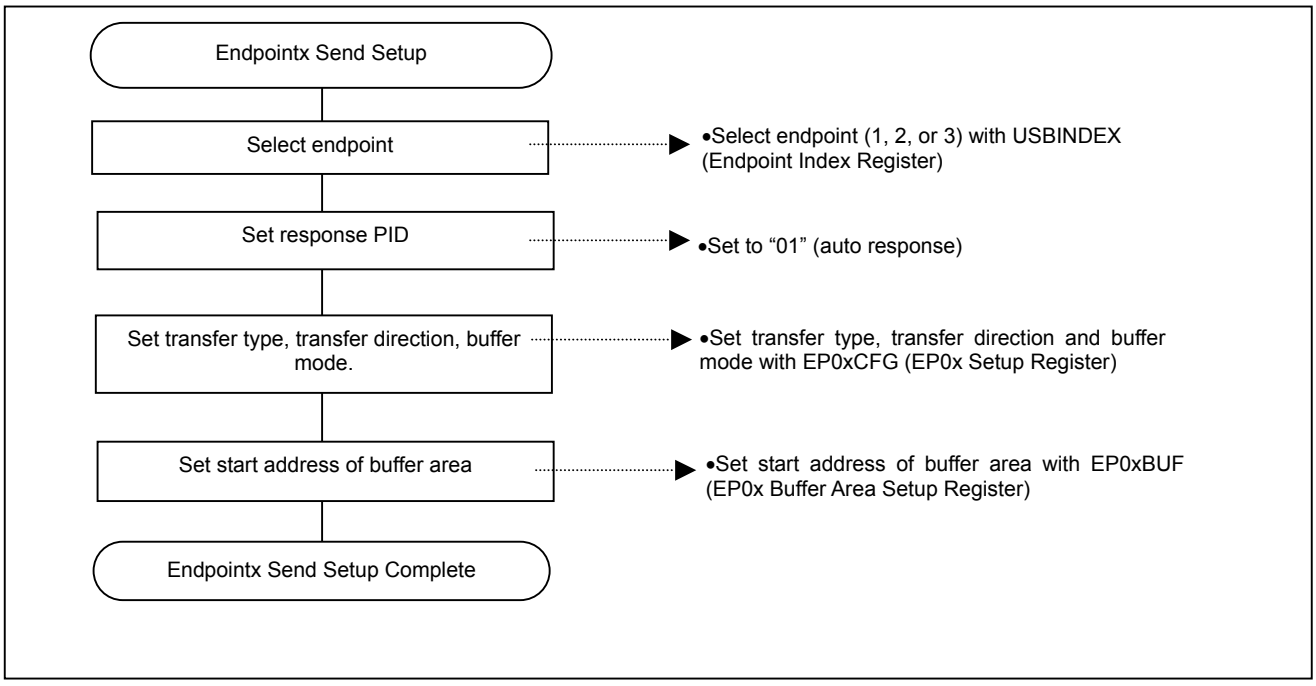


Figure 5.21 Example of Endpointx Transmit Setup Process

(2) Normal Transmit

[1] Single Buffer Mode

In the single buffer mode, for Endpointx to transmit a packet to the Host, prepare the packet data in Data Buffer 0 to be returned with the IN token from the Host.

To prepare one packet of data, confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty. After writing the data to Data Buffer 0 (the same number of bytes as the packet data to be sent), set B0VAL0x (Buffer 0 Enable Bit) of EP0xCON2 (EP0x Control Register 2) to “1”. (B0VAL0x bit is automatically cleared by H/W in normal receive transmissions.) The number of bytes of packet data to be sent must be set in EP0xBYT0 (EP0x Byte Number Register).

When one packet of data has been prepared, the data is sent with the next IN token from the Host. When this data transmit is complete, the Endpointx Buffer 0 Ready Interrupt Request is generated and Data Buffer 0 empties by one packet space.

[2] Double Buffer Mode

For Endpointx to transmit packets to the Host, prepare a packet of data (DATA0 in Data Buffer 0 and DATA1 in Data Buffer 1) to be returned with the IN token from the Host

To prepare the packets of data, confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty. For DATA0, after writing the data to Data Buffer 0 (the same number of bytes as the packet data to be sent), set B0VAL0x of EP0xCON2 to “1”. (B0VAL0x bit is automatically cleared by H/W in normal receive transmissions.) The number of bytes of packet data to be sent must be set in EP0xBYT0 (EP0x Byte Number Register).

For DATA1, after writing the data to Data Buffer1 (the same number of bytes as the packet data to be sent), set B1VAL0x of EP0xCON3 (EP0x Control Register 3) to “1”. (B1VAL0x bit is automatically cleared by H/W in normal receive transmissions.) The number of bytes of packet data to be sent must be set in EP0xBYT1 (Endpointx Buffer 1 Ready Interrupt).

When one packet of data has been prepared, the data is sent from the Host with the next IN token. When this data transmit is complete, Endpointx Buffer 0 (or 1) Ready Interrupt Request is generated and Data Buffer 0 (or 1) empties by one packet space.

(4) STALL Response

Set PID0x (Response PID Bit) of EP0xCON1 (EP0x Control Register 1) to “1x”. A STALL handshake is continually returned in response to IN/OUT tokens of the Endpointx in the STALL condition. When a STALL is sent, communications are disabled unless the Host CPU cancels the STALL with the ClearFeature. Even when communications are restarted, the STALL is not cancelled unless the PID0x bit is set to “00” or “01”.

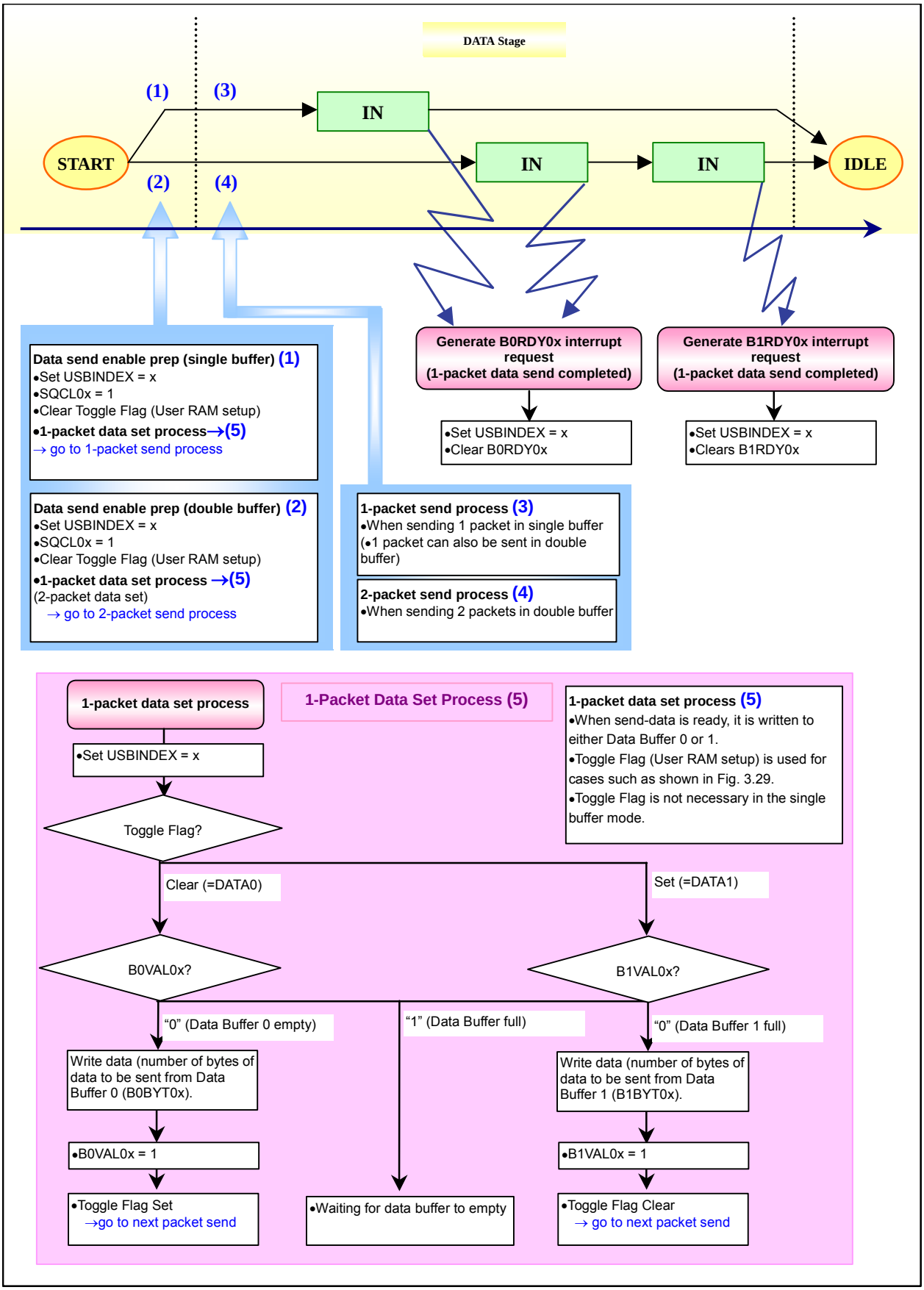


Figure 5.22 Example of Endpointx Data Transmit

5.8.12. Bulk Transfer Data Transmit

(1) Transfer Type Setup

To use Endpointx IN for bulk transfers, set the endpoint to bulk transfer with EP0xCFG (EP0x Setup Register). Refer to [Figure 5.21 Example of Endpointx Transmit Setup Procedure] for details.

(2) Data Transmit Prep

Set SQCL0x (Sequence Toggle Bit Clear Bit) to “1”. By setting this bit to “1”, the toggle bit is cleared and the next data PID is initialized to DATA0.

(3) Data Transmit Operation

An example of Endpointx bulk data transmit is shown in Figure 5.23.

Detailed explanations of single buffer and double buffer mode operations are provided below.

[1] Single Buffer Mode

A NAK handshake is automatically returned when an IN token is received from the Host but data is not yet set in Endpointx Data Buffer 0.

In order to transmit data, a packet data must be set in Data Buffer 0. Confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty (B0VAL0x Bit is “0”; write-enabled). Set the transmit packet data and set B0VAL0x bit to “1”. (B0VAL0x bit is automatically cleared by H/W in normal transmit transmissions.)

Set the number of bytes of packet data to be sent in EP0xBYT0 (EP0x Byte Number Register 0).

After the data is set, Endpointx IN transmits the data to the Host upon receiving an IN token. The Host returns an ACK handshake in response to the data packet, completing the transmission of one packet, and the Endpointx Buffer 0 Ready Interrupt is generated.

In a bulk transfer, when an IN endpoint successfully receives an ACK handshake from the Host, DATA0 and DATA1 are toggled. But if the endpoint does not receive an ACK from the Host, the same data from the same toggle is sent in response to the next IN token.

The transfer data is written to the data buffers alternately, as follows: DATA0 → DATA1 → DATA0…

[2] Double Buffer Mode

A NAK handshake is automatically returned when an IN token is received from the Host but data is not yet set in Endpointx Data Buffer 0 or 1.

In order to transmit DATA0, confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty (B0VAL0x Bit is “0”; write-enabled). Set the transmit packet data and set B0VAL0x bit to “1”. (B0VAL0x bit is automatically cleared by H/W in normal transmit transmissions.)

Set the number of bytes of packet data to be sent in EP0xBYT0 (EP0x Byte Number Register 0).

For DATA1, confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty (B0VAL0x Bit is “0”; write-enabled), in the same manner as for DATA0. Set the transmit packet data and set B1VAL0x bit to “1”. (B1VAL0x bit is automatically cleared by H/W in normal transmit operations.)

Set the number of bytes of packet data to be sent in EP0xBYT1 (EP0x Byte Number Register 1).

After the data is set, Endpointx IN transmits the data to the Host upon receiving an IN token. The Host returns an ACK handshake in response to the data packet, completing the transmission of one packet. At this time the Endpointx Buffer 0 (or 1) Ready Interrupt Request is generated and Data Buffer 0 (or 1) empties one packet space.

In a bulk transfer, when an IN endpoint successfully receives an ACK handshake from the Host, the data is toggled. But if the endpoint does not receive an ACK from the Host, the same data from the same toggle is sent in response to the next IN token.

When the data is to be sent in the following order, DATA0 is sent to Data Buffer0 and DATA1 is sent to Data Buffer 1 alternately. DATA0 → DATA1 → DATA0…

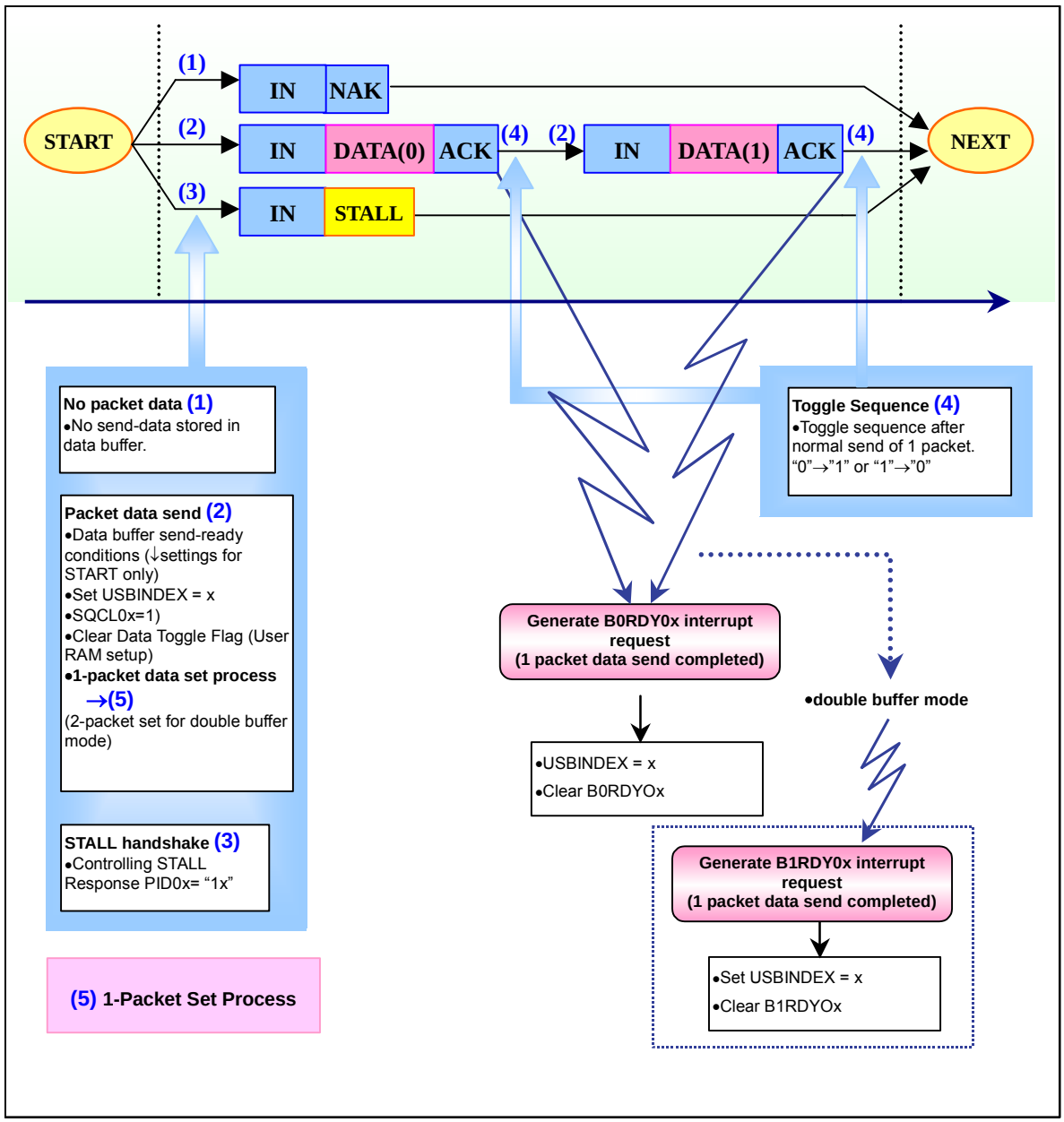


Figure 5.23 Example of Endpointx Bulk Data Transmit

• Example of S/W Process in Double Buffer Mode

As shown in Figure 5.23, when using the double buffer mode, if an interrupt request is generated for the first packet but the interrupt is not processed (the buffer is not written to) due to current execution of another process, both interrupt requests, B0RDY0x and B1RDY0x, are generated when an interrupt request is generated for the second packet.

At this time, when processing both interrupt requests, because data is sent to data buffers alternately (either DATA0 → DATA1 or DATA1 → DATA0), the user must manage the DATA0/DATA1 toggle sequence with software in order to handle data written to the two buffers. Refer to the example of software processing in Figure 5.24.

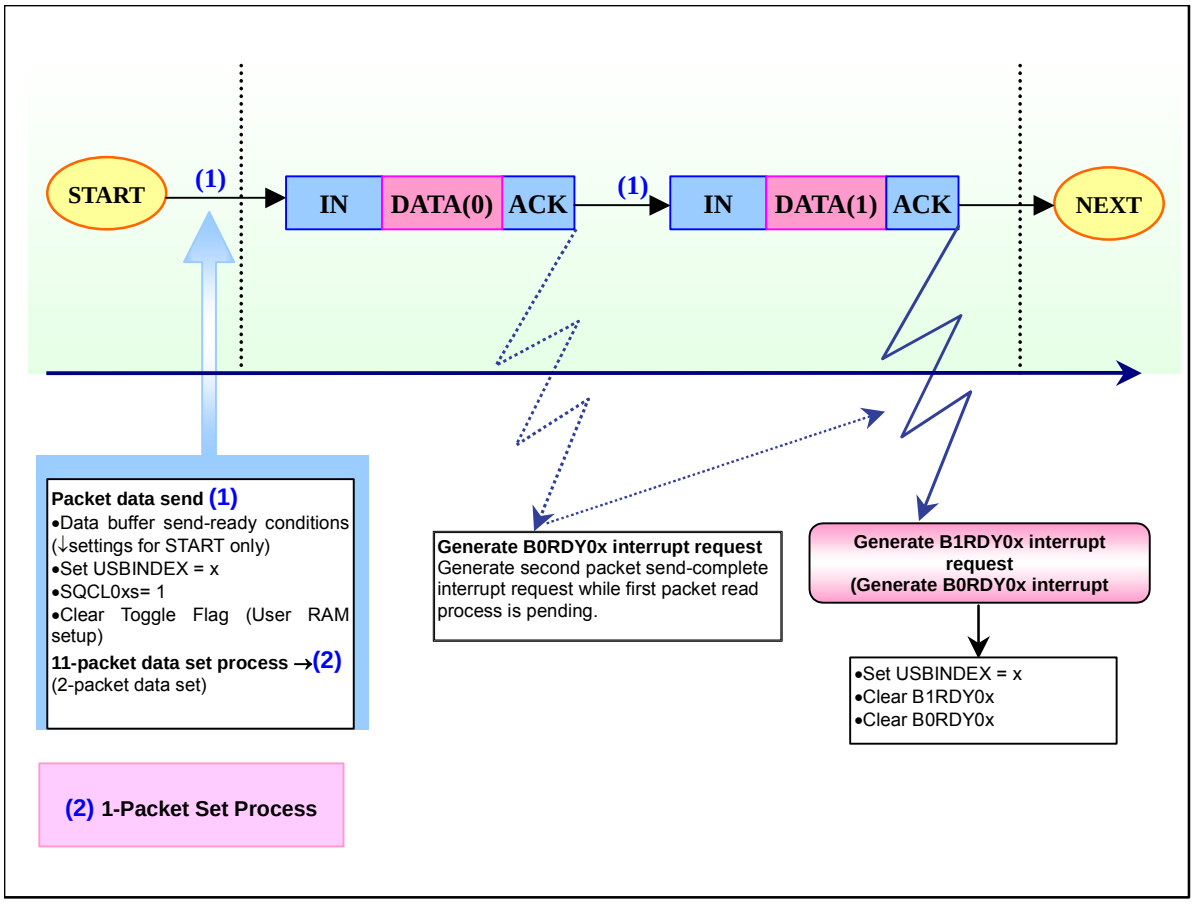


Figure 5.24 Example of Endpointx Bulk Data Transmit (double buffer mode)

5.8.13. Interrupt Transfer Data Transmit

(1) Transfer Type Setup

To use Endpointx IN for interrupt transfers, set the endpoint to interrupt transfer with EP0xCFG (EP0x Setup Register). Refer to the procedure in [Figure 5.21 Example of Endpointx Transmit Setup Procedure] for details.

(2) Data Transmit Prep

Preparation for interrupt transfers is the same as that of bulk transfers. Refer to [Chapter 5.8.12 Bulk Transfer Data Transmit] for details.

(3) Data Transmit Operation

The Endpointx IN operation for interrupt transfers is the same as that of bulk transfers. Refer to [Chapter 5.8.12 Bulk Transfer Data Transmit] for details.

5.8.14. Isochronous Transfer Data Transmit

(1) Transfer Type Setup

To use Endpointx IN for isochronous transfers, set the endpoint to isochronous transfer with EP0xCFG (EP0x Setup Register). Refer to the procedure in [Figure 5.21 Example of Endpointx Transmit Setup Procedure] for details.

(2) Data Transmit Prep

Set SQCL0x (Sequence Toggle Bit Clear Bit) to “1”. By setting this bit to “1”, the toggle bit is cleared and the next data PID is initialized to DATA0.

(3) Data Transmit Operation

An example of Endpointx isochronous data transmit is shown in Figure 5.25.
The isochronous transaction (IN) is repeated until all data is sent to the Host. The isochronous transaction does not include a handshake phase. The data packet is toggled and sent as DATA0 and DATA1.

Note: 38K2 Group MCUs toggle DATA0 and DATA1 in normal transfers.
(Refer to USB Specification Ver. 2.0 [Chapter 8.5.4 Isochronous Transactions].

If an IN token is received from the Host when there is no data in Data Buffer 0/1, the ERR0x interrupt (as an underrun error) is generated. When preparing the packet data, by not setting the data (in other words, set EP0xBYT0/EP0xBYT1 Register to “0”), the ERR0x interrupt will not be generated and an empty data packet (NULL data) can be sent.

Detailed explanations of single buffer and double buffer mode operations are provided below.

[1] Single Buffer Mode

In order to transmit data in the single buffer mode, the packet data must be set in Data Buffer 0. Confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty (B0VAL0x bit is “0”; write-enabled). Set the transmit packet data, then set B0VAL0x bit to “1”. (B0VAL0x bit is automatically cleared by H/W in normal receive transmissions.)
The number of bytes of packet data to be sent must be set in EP0xBYT0 (EP0x Byte Number Register).

After the data is set, Endpointx IN transmits the data to the Host upon receiving an IN token. When one packet of data is sent successfully, the Endpointx Buffer 0 Ready interrupt is generated.

[2] Double Buffer Mode

In order to transmit data in the double buffer mode, if the packet is an even-numbered packet (starting with packet number 0), confirm that Endpointx is not in the HALT state and that Data Buffer 0 is empty (B0VAL0x bit is “0”; write-enabled). Set the transmit packet data and set B0VAL0x bit to “1”. (B0VAL0x bit is automatically cleared by hardware in normal transmit transmissions.)

Set the number of bytes of packet data to be sent in EP0xBYT0) (EP0x Byte Number Register 0).

In the same manner, when the packet is odd-numbered, confirm that Endpointx is not in the HALT state and that Data Buffer 1 is empty (B1VAL0x bit is “0”; write-enabled). Set the transmit packet data and set B1VAL0x bit to “1”. (B1VAL0x bit is automatically cleared by hardware in normal transmit operations.)

Set the number of bytes of packet data to be sent in EP0xBYT1 (EP0x Byte Number Register 1).

After the data is set, Endpointx transmits the data to the Host upon receiving an IN token. When one packet of either DATA0 or DATA1 is sent successfully, Data Buffer 0 or 1 Ready Interrupt Request is generated accordingly.

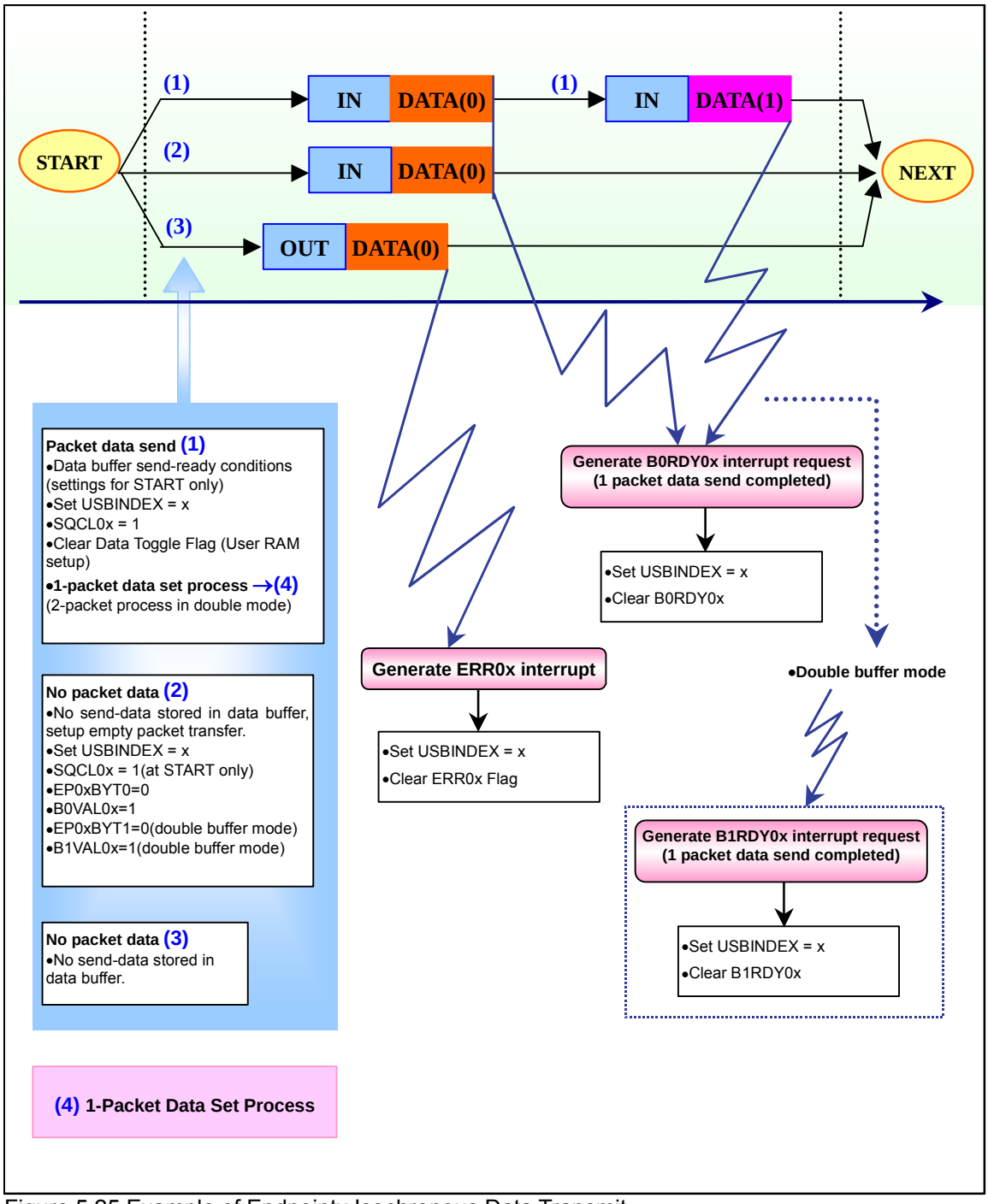
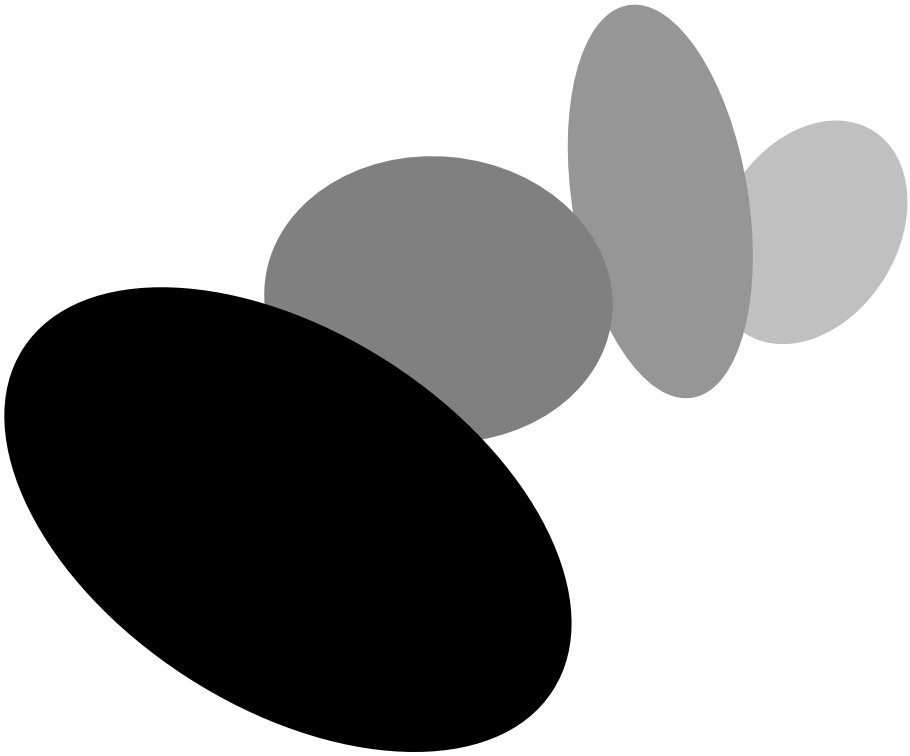


Figure 5.25 Example of Endpointx Isochronous Data Transmit

• Example of S/W Process in Double Buffer Mode

As shown in Figure 5.24, when using the double buffer mode, if an interrupt request is generated for the first packet but the interrupt is not processed (the buffer is not read) due to the current execution of another process, both interrupt requests, B0RDY0x and B1RDY0x, are generated when an interrupt request is generated for the second packet.

At this time, when processing both interrupt requests, the user needs to toggle DATA0 and DATA1 with software accordingly (DATA0→DATA1 or DATA1→DATA0) so that the data buffers will be written to in the appropriate order. See Figure 5.24 for an example of the software processing.



Chapter 6
Power Management

This chapter explains differences in external power circuits according to power supply methods, the USB Vbus detection method, and the USB external circuit.

6.1. Power Supply

The 38K2 Group provides an operating voltage of (Vcc) 3.00-5.25V (“L” version).
38K2 Group operating voltage (Vcc) differs for the system clock and internal clock (Ø), as shown in Table4.1.

Table 6.1 Correspondence of Clock and Power Voltage Vcc

f(Xin)	System Clock	Internal Clock Ø	Power Voltage V_{CC}
6MHz or 12MHz	12MHz	6MHz (2 division)	4.00~5.25
	8MHz	8MHz	4.00~5.25
	6MHz	6MHz	4.00~5.25
			3.00~4.00

The following three device power supply methods are available:

- Self-power Method:
  MCU Vcc is supplied by a DC connector or battery.
- Bus-power Method:
  MCU Vcc is supplied by the Host PC through the USB Vbus line.
- Selectable Self/Bus-power Method:
  Either Self-power and Bus-power method can be selected.

6.2. External Power Circuit

The external power circuit should be selected according to the power supply method. Bus-power and selectable Self/Bus-power methods require regulation of the operating voltage. For 3.3V operations when power is supplied from the host PC, a regulator should be inserted to drop the USB Vbus voltage (the Bus power voltage supplied by USB up-port) from 5V to 3.3V.The selectable self/bus-power method requires a USB Vbus detection circuit and software procedure for the system to detect USB cable attachment or detachment.

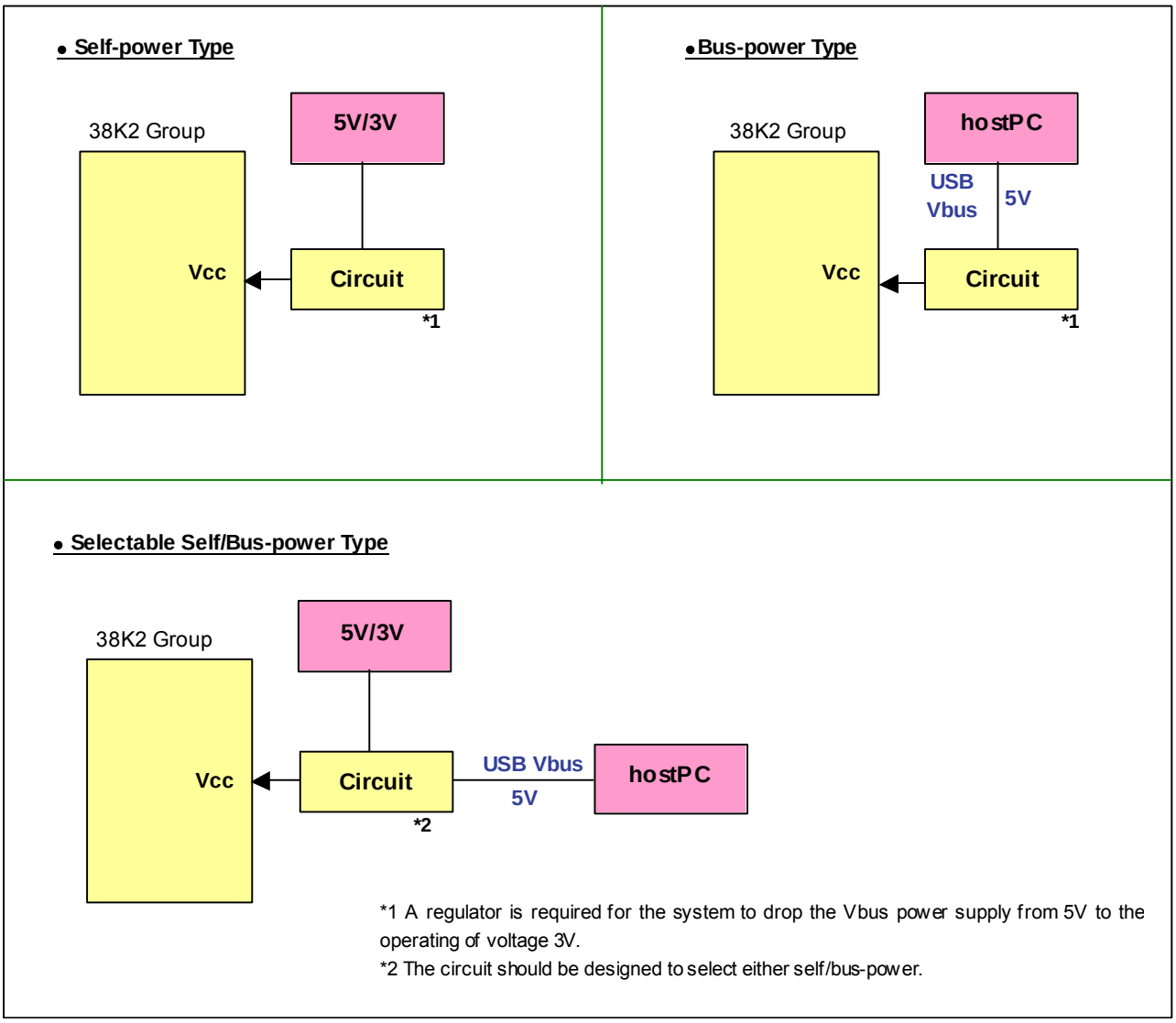


Figure 6.1 Vcc Pin External Circuit Diagram

6.3. USB External Circuit (5V/3V operation differences)

The USB external circuit and USB control register (USBCON) settings differ according to the operating voltages and power supply types.

38K2 Group operating voltage (Vcc) differs depending on the system clock and internal clock (Ø). The correspondence is shown in Table 4.2. The USB reference voltage circuit should be disabled for the USBVref pin to supply 3.00~3.60V when Vcc=3.00~4.00V and the USBVref voltage should not exceed Vcc.

Table 6.2 Clock, Power Voltage Vcc and USB Reference Voltage USBVref Correspondence

f(Xin)	System Clock	Internal Clock Ø	Power Voltage V_{CC}	USB reference voltage $USBV_{REF}$
6MHz or 12MHz	12MHz	6MHz (2division)	4.00~5.25	USB reference voltage circuit enabled
	8MHz	8MHz	4.00~5.25	
	6MHz	6MHz	4.00~5.25	
			3.00~4.00	USB reference voltage circuit disabled 3.00~3.60V directly to $USBV_{REF}$

(1) In 4.00~5.25V operation

The USB reference voltage circuit can be used and is therefore enabled.

♦At USB block initialization

USB reference voltage enable bit (VREFE) = “1” (USB reference voltage circuit enabled)

USB reference voltage control bit (VRECON) = “0” (Normal mode)

♦In USB suspend interrupt procedure routine

USB reference voltage enable bit (VREFE) = “1” (USB reference voltage circuit enabled)

USB reference voltage control bit (VRECON) = “1” (Low power consumption mode)

♦At resume interrupt procedure from suspend (USB Resume interrupt/Remote Wake-up)

USB reference voltage enable bit (VREFE) = “1” (USB reference voltage circuit enabled)

USB reference voltage control bit (VRECON) = “0” (Normal mode)

(2) In 3.00~4.00V operation

The USB reference voltage circuit cannot be used and is therefore disabled.

♦At USB block initialization

USB reference voltage enable bit (VREFE) = “0” (USB reference voltage circuit disabled)

USBVref should be connected to Vcc if Vcc=3.00~3.60V.

Figure. 6.2 and Figure 6.3 show USB external circuit connections.

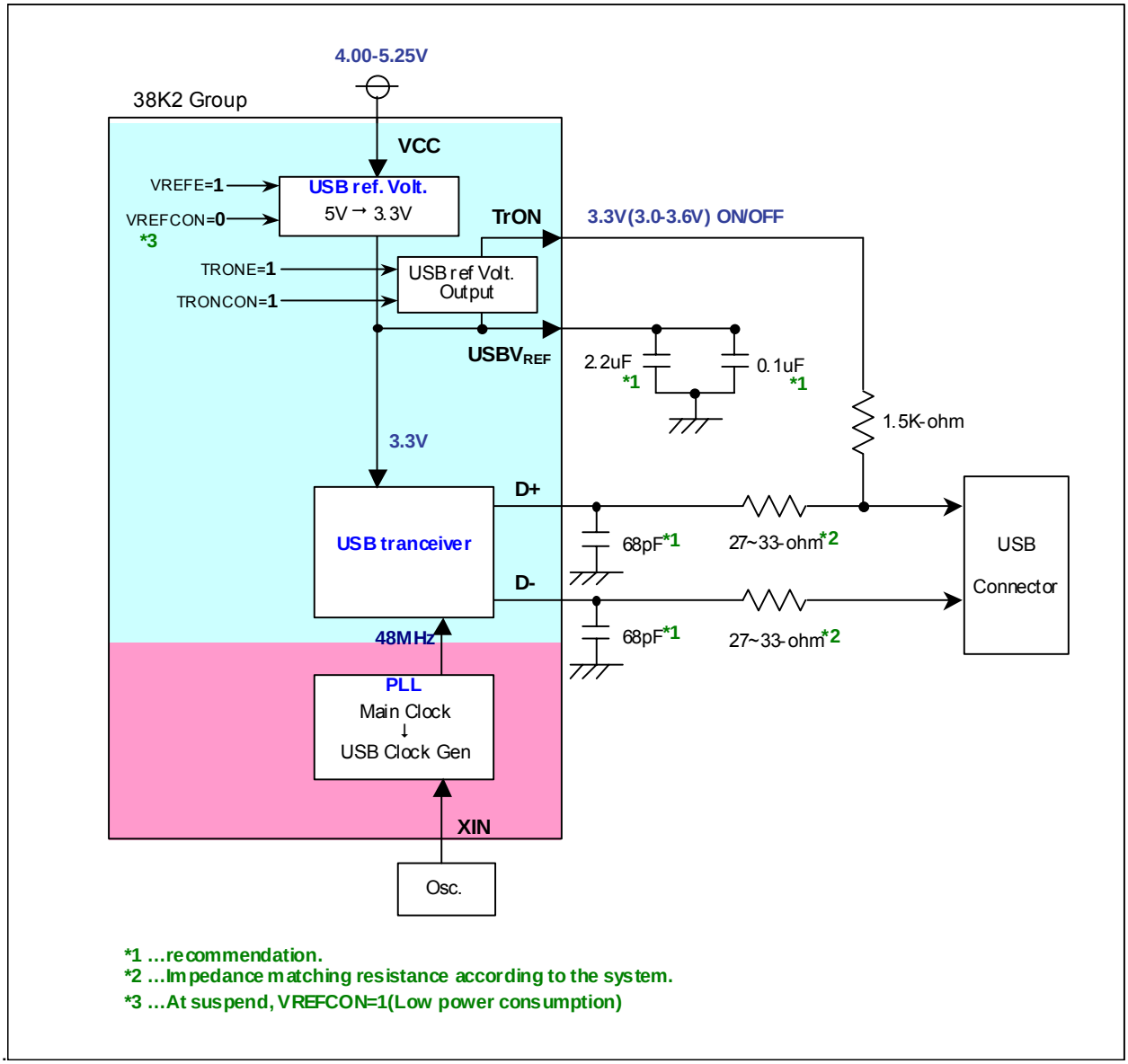


Figure 6.2 Example of USB External Circuit (USB reference voltage circuit enabled)

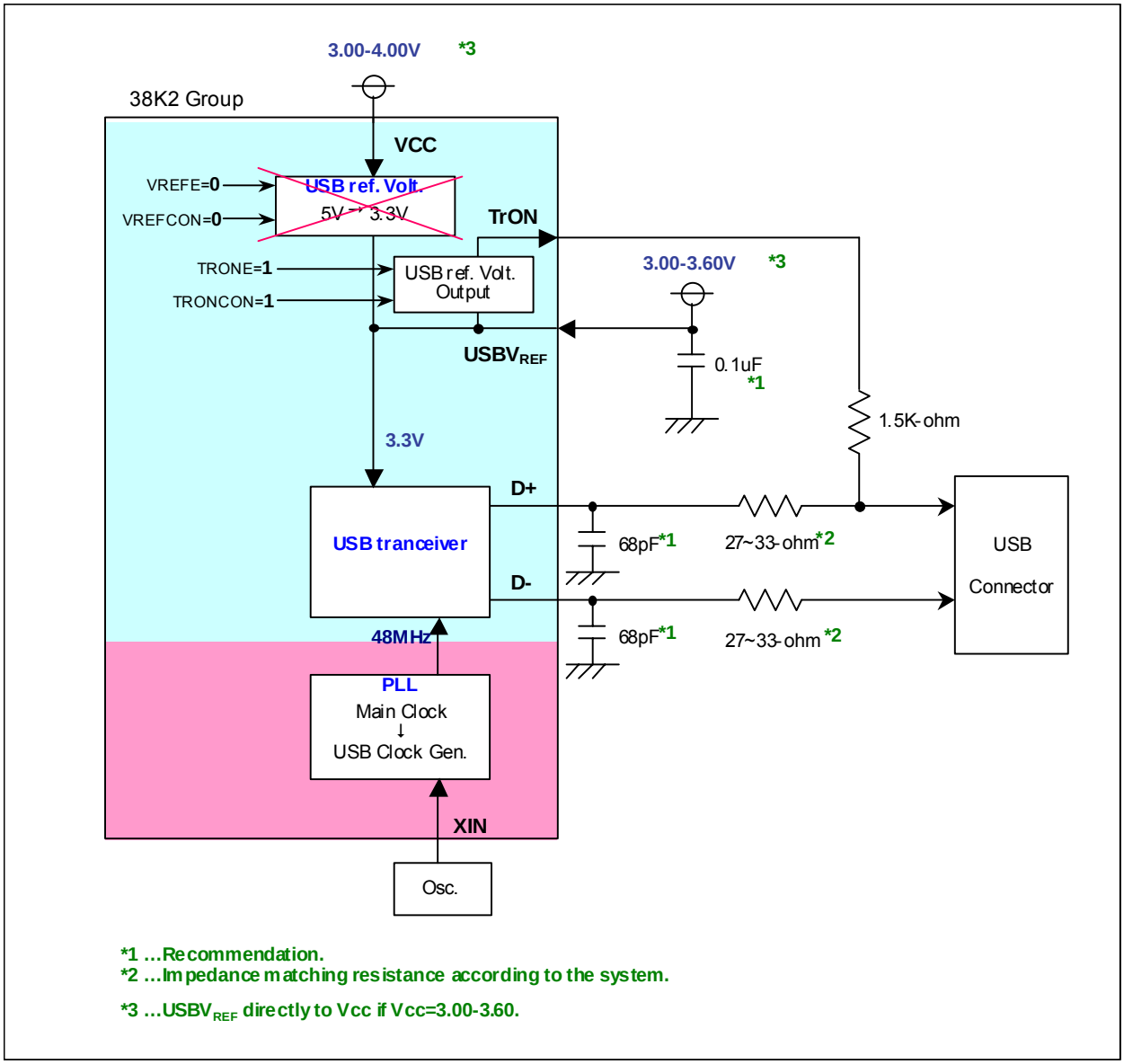


Figure 6.3. Example of USB External Circuit (USB reference voltage circuit disabled)

6.4. USB Cable Attachment/Detachment

6.4.1. For USB Vbus Detection

(1) USB reference voltage output (TrON) ON/OFF control

USB reference voltage output (TrON) is an up-port pull-up voltage output pin. The pin outputs $USBV_{REF}$ voltage, which may be supplied internally/externally. It is connected to D0+ line with a 1.5k-ohm resistor in series.

The TrON voltage output may be turned ON/OFF by the TrOn output control bit (TRONCON), bit 1 of USB Control register (USBCON). TrON is normally turned ON after USB Vbus voltage detection.

(2) Self-power USB Vbus line detection

When microcomputers are in self-power operation, USB Vbus cannot be sustained after the up-port is detached from the host PC. TrON voltage output must be turned OFF and the USB Function halted when the USB Vbus is not sustained. The USB block may be enabled after the following USB Vbus detection.

The 38K2 Group provides no H/W USB Vbus detection. It requires H/W USB Vbus detection circuits or F/W control by assigning the USB Vbus to an interrupt.

6.4.2. USB Vbus Detection Method

There are various methods of detecting the USB Vbus status, including using a comparator or monitoring changes on the Vbus line. This section explains the most common method, connecting the USB Vbus to the INT interrupt, as shown in the example in Figure 6.4.

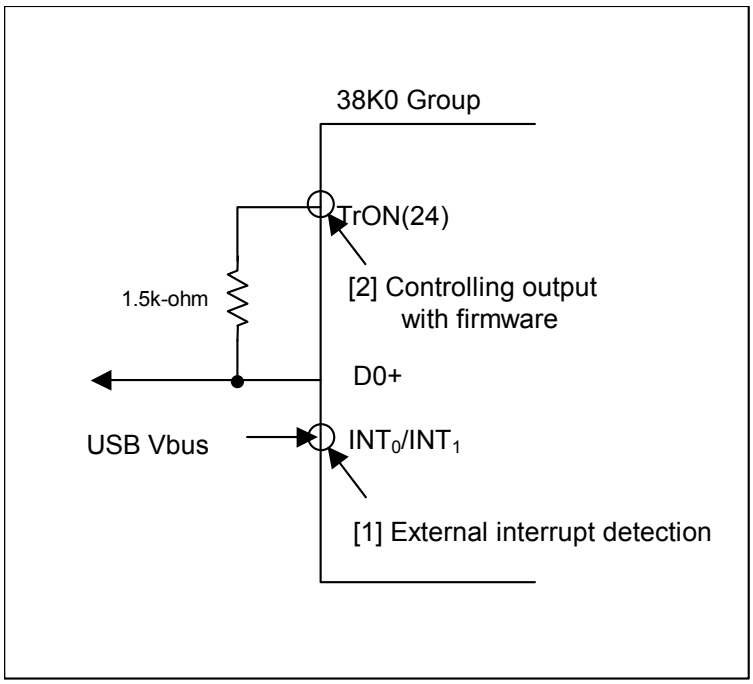


Figure 6.4 Example of USB Vbus Detection by External Interrupt

(1) INT interrupt settings

USB Vbus and INT interrupt should be connected as shown in Figure 4.4. The INT_0 interrupt or INT_1 interrupt (functioning as USB Vbus detector) detects USB cable attachment if the edge is "L" to "H" and USB cable detachment if the edge is "H" to "L". The detection edge may be programmed by INT_0 /INT_1 Interrupt Edge Select Bit of Interrupt Edge Select Register (Address $0FF3_{16}$). Interrupt Request Bit should be cleared before enabling the interrupts.

(2) USB Vbus Detection

USB Vbus detection must be performed a few times in 2~3 ms intervals to be consistent.

♦USB Cable Attachment:

USB Vbus power supply should be enabled. The disabled USB block should be enabled and initialized for enumeration.

♦USB Cable Detachment:

The USB block and USB clock supply (PLL) should be disabled.

(3) USB Vbus Detection at USB Suspend

INT interrupt occurs while the 38K2 Group internal clock supply is stopped. At USB suspend when the internal clock supply is stopped, USB Vbus input may be detected after the resume from USB suspend and restart of the internal clock supply.

INT interrupt settings should be inserted in the USB suspend interrupt routine at the resume from USB suspend (Clear Interrupt Request Bit and Set Interrupt Control Bit).

Revision History	38K2 Group Application Notes

Ver.No.	Date	Contents
1.0	Jan./29/03	First edition issued
1.1	Oct./04/06	* Notes on USB Communication is added. * Table 6.1 and Table 6.2 are revised.

RENESAS

RENESAS Single-Chip Microcomputer
38K2 Group
USB Application Notes VER.1.1
RENESAS TECHNOLOGY CORPORATION.
RENESAS SOLUTIONS CORPORATION.