ON Semiconductor

Is Now

To learn more about onsemi™, please visit our website at
www.onsemi.com

LC87F1M16A
USB Application Note

2012/09/05

SANYO Semiconductor Co., Ltd.
LSI Division
Microcontroller & Flash Development Dept.

Contents

Figures and Tables

Chapter 1. Initialization

Chapter 2. State Transitions and Interrupts

Chapter 3. Control Transfers

Chapter 4. Data Transfers

Chapter 1. Initialization

1.1. Control Program Overview

The LC87F1M16A notifies the CPU of USB data transmissions and receptions and special signal processing in the form of interrupts. As such, its control program is composed of an initialization program and a USB interrupt processing program. Assuming that USB sources are to be processed by interrupt processing, the schematic system flow of the LC87F1M16A looks like as shown in below.
The function Main needs to initialize the LC87F1M16A and USB functions. Figure 1-1 shows a sample control program configuration.

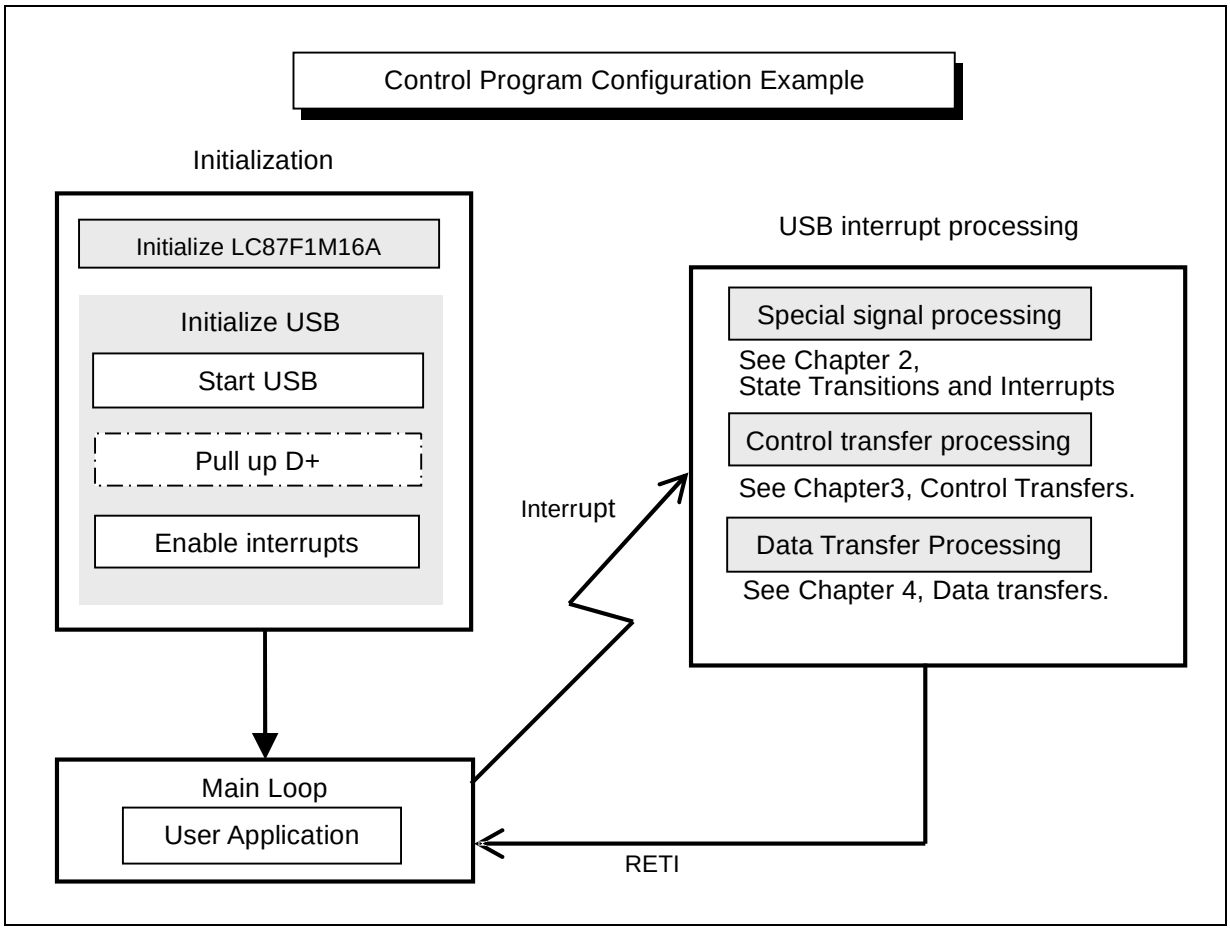


Figure 1-1 Sample Control Program Configuration

1.2. Related Registers

The table below shows a list of registers that need to be initialized before using the USB functions.

Register Name	Symbol	Address	R/W	Function
USB clock division control register	**USBDIV**	FE04h	R/W	USB clock devision
Master interrupt enable control register	**IE**	FE08h	R/W	Enables interrupts
System clock division control register	**CLKDIV**	FE0Ch	R/W	Clock devision
PLL control register	**PLLCNT**	FE0Dh	R/W	PLL
Oscillation control register	**OCR**	FE0Eh	R/W	Oscillation
USB operation control register	**USCTRL**	FE80h	R/W	USB operation control
USB interrupt control register	**USBINT**	FE82h	R/W	USB interrupt control
Endpoint buffer mode register	**EPBMOD**	FEABh	R/W	Mapping address in RAM of the endpoint buffer

Table 1-1 Initialization Related Registers

1.3. Initializing the LC87F1M16A

When using USB, it is necessary to synchronize the system clock and 48MHz clock for USB. The 48MHz clock for USB is created by PLL circuit using a main clock. The oscillation for the main clock is enabled by connecting a ceramic oscillator and a capacitor across the CF1 and CF2 pins. The system clock selects the divided clock of 48MHz for USB created by the PLL. After the PLL circuit gets stabilized, a wait time need to be provided to avoid unstable operation due to clock related problems. Since the wait time varies depending on the oscillator to be used, an adequate wait time should be provided according to the characteristics of the oscillator.

Figure 1-2 shows a sample procedure for initializing the LC87F1M16A.

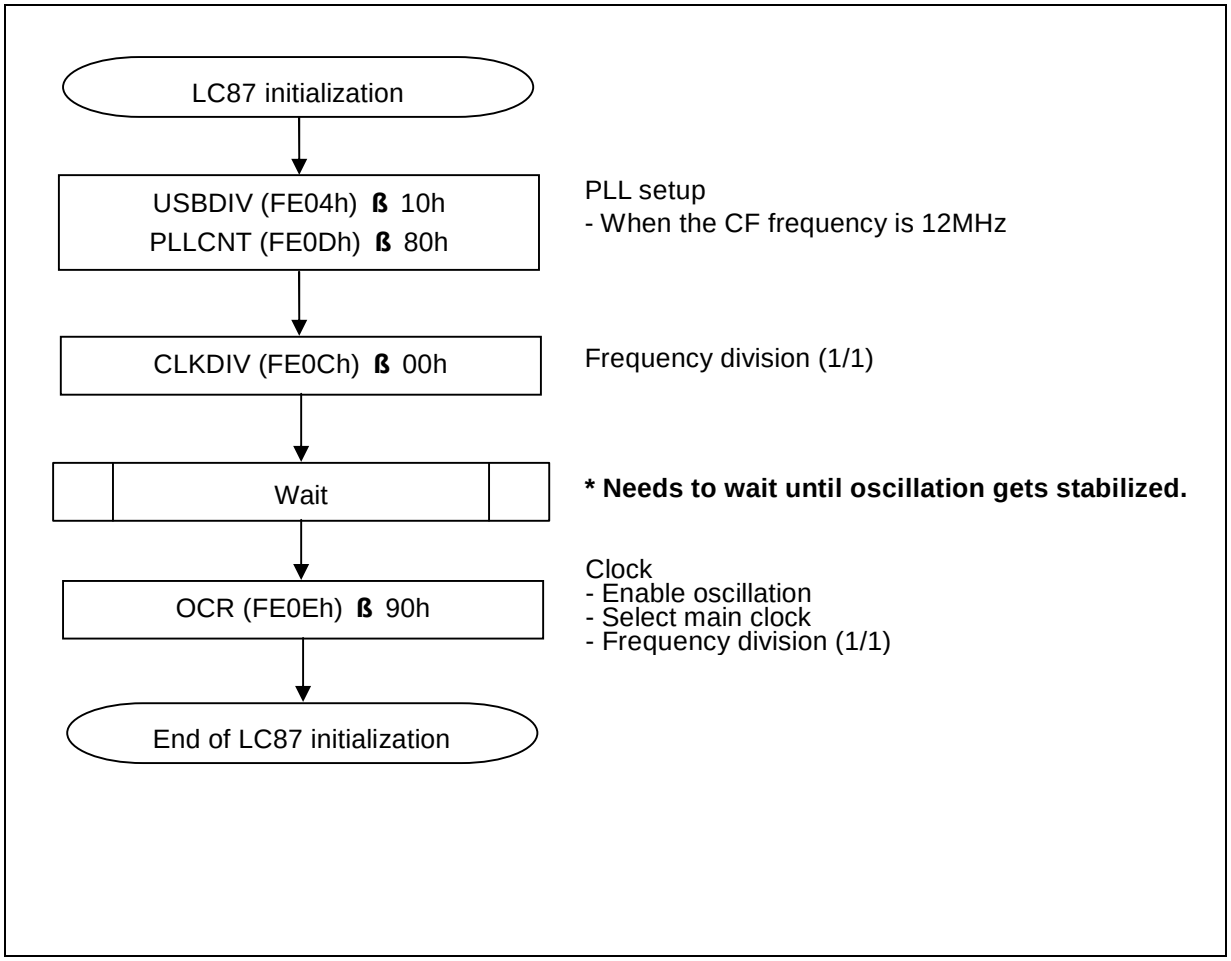


Figure 1-2 LC87F1M16A Initialization Example

1.4. Initializing the USB Functions

It is the Powered state that a USB device initially enters. Accordingly, it is necessary to set up the USB functions so that the device can run in the Powered state. USB initialization should be carried out after completing the initialization of the LC87F1M16A. For details on the device states, see Chapter 2, "State Transitions and Interrupts."
Figure 1-3 shows a sample procedure for initializing the USB.
The endpoint buffer for data transmission and reception (64 bytes maximum) is mapped into RAM. The address mapping RAM of endpoint buffer can be selected by configuring the endpoint buffer mode register (EPBMOD). Table 1-2 – Table 1-10 show the endpoint configuration. In this document EPBMOD=0 (default) is used.

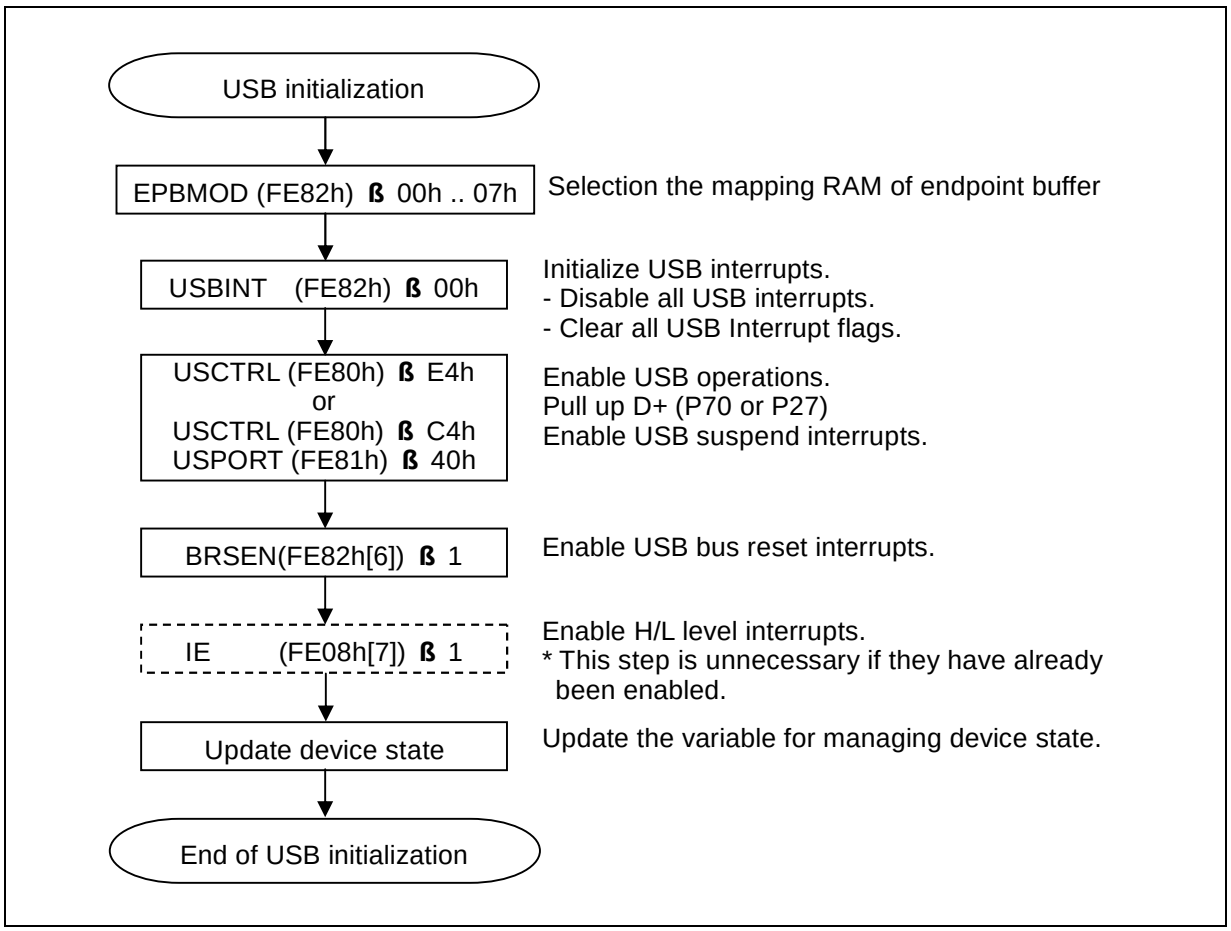


Figure 1-3 USB Initialization Example

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	64	0200H – 023FH
	Transmit		0240H – 027FH
EP1	Bank 0	64	0280H – 02BFH
	Bank 1		02C0H – 02FFH
EP2	Bank 0	64	0300H – 033FH
	Bank 1		0340H – 037FH
EP3	Bank 0	64	0380H – 03BFH
	Bank 1		03C0H – 03FFH
EP4	Bank 0	64	0380H – 03BFH
	Bank 1		03C0H – 03FFH
EP5	Bank 0	64	0300H – 033FH
	Bank 1		0340H – 037FH
EP6	Bank 0	64	0280H – 02BFH
	Bank 1		02C0H – 02FFH

Table 1-2 Endpoint Configuration (**EPBMOD**=0)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	64	0380H – 03BFH
	Transmit		03C0H – 03FFH
EP1	Bank 0	64	0300H – 033FH
	Bank 1		0340H – 037FH
EP2	Bank 0	64	0280H – 02BFH
	Bank 1		02C0H – 02FFH
EP3	Bank 0	16	0260H – 026FH
	Bank 1		0270H – 027FH
EP4	Bank 0	16	0240H – 024FH
	Bank 1		0250H – 025FH
EP5	Bank 0	16	0220H – 022FH
	Bank 1		0230H – 023FH
EP6	Bank 0	16	0200H – 020FH
	Bank 1		0210H – 021FH

Table 1-3 Endpoint Configuration (**EPBMOD**=1)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	64	0380H – 03BFH
	Transmit		03C0H – 03FFH
EP1	Bank 0	32	0340H – 035FH
	Bank 1		0360H – 037FH
EP2	Bank 0	32	0300H – 031FH
	Bank 1		0320H – 033FH
EP3	Bank 0	32	02C0H – 02DFH
	Bank 1		02E0H – 02FFH
EP4	Bank 0	32	0280H – 029FH
	Bank 1		02A0H – 02BFH
EP5	Bank 0	32	0240H – 025FH
	Bank 1		0260H – 027FH
EP6	Bank 0	32	0200H – 021FH
	Bank 1		0220H – 023FH

Table 1-4 Endpoint Configuration (**EPBMOD**=2)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	32	03C0H – 03DFH
	Transmit		03E0H – 03FFH
EP1	Bank 0	32	0380H – 039FH
	Bank 1		03A0H – 03BFH
EP2	Bank 0	32	0340H – 035FH
	Bank 1		0360H – 037FH
EP3	Bank 0	32	0300H – 031FH
	Bank 1		0320H – 033FH
EP4	Bank 0	32	02C0H – 02DFH
	Bank 1		02E0H – 02FFH
EP5	Bank 0	32	0280H – 029FH
	Bank 1		02A0H – 02BFH
EP6	Bank 0	32	0240H – 025FH
	Bank 1		0260H – 027FH

Table 1-5 Endpoint Configuration (**EPBMOD**=3)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	32	03C0H – 03DFH
	Transmit		03E0H – 03FFH
EP1	Bank 0	32	0380H – 039FH
	Bank 1		03A0H – 03BFH
EP2	Bank 0	32	0340H – 035FH
	Bank 1		0360H – 037FH
EP3	Bank 0	32	0300H – 031FH
	Bank 1		0320H – 033FH
EP4	Bank 0	16	02E0H – 02EFH
	Bank 1		02F0H – 02FFH
EP5	Bank 0	16	02C0H – 02CFH
	Bank 1		02D0H – 02DFH
EP6	Bank 0	16	02A0H – 02AFH
	Bank 1		02B0H – 02BFH

Table 1-6 Endpoint Configuration (**EPBMOD**=4)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	32	03C0H – 03DFH
	Transmit		03E0H – 03FFH
EP1	Bank 0	16	03A0H – 03AFH
	Bank 1		03B0H – 03BFH
EP2	Bank 0	16	0380H – 038FH
	Bank 1		0390H – 039FH
EP3	Bank 0	16	0360H – 036FH
	Bank 1		0370H – 037FH
EP4	Bank 0	16	0340H – 034FH
	Bank 1		0350H – 035FH
EP5	Bank 0	16	0320H – 032FH
	Bank 1		0330H – 033FH
EP6	Bank 0	16	0300H – 030FH
	Bank 1		0310H – 031FH

Table 1-7 Endpoint Configuration (**EPBMOD**=5)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	16	03E0H – 03EFH
	Transmit		03F0H – 03FFH
EP1	Bank 0	16	03C0H – 03CFH
	Bank 1		03D0H – 03DFH
EP2	Bank 0	16	03A0H – 03AFH
	Bank 1		03B0H – 03BFH
EP3	Bank 0	16	0380H – 038FH
	Bank 1		0390H – 039FH
EP4	Bank 0	8	0370H – 0377H
	Bank 1		0378H – 037FH
EP5	Bank 0	8	0360H – 0367H
	Bank 1		0368H – 036FH
EP6	Bank 0	8	0350H – 0357H
	Bank 1		0358H – 035FH

Table 1-8 Endpoint Configuration (**EPBMOD**=6)

	Setting	Max. Size (in bytes)	RAM Address
EP0	Receive	8	03F0H – 03F7H
	Transmit		03F8H – 03FFH
EP1	Bank 0	8	03E8H – 03EFH
	Bank 1		03B8H – 03BFH
EP2	Bank 0	8	03E0H – 03E7H
	Bank 1		03B0H – 03B7H
EP3	Bank 0	8	03D8H – 03DFH
	Bank 1		03A8H – 03AFH
EP4	Bank 0	8	03D0H – 03D7H
	Bank 1		03A0H – 03A7H
EP5	Bank 0	8	03C8H – 03CFH
	Bank 1		0398H – 039FH
EP6	Bank 0	8	03C0H – 03C7H
	Bank 1		0390H – 0397H

Table 1-9 Endpoint Configuration (**EPBMOD**=7)

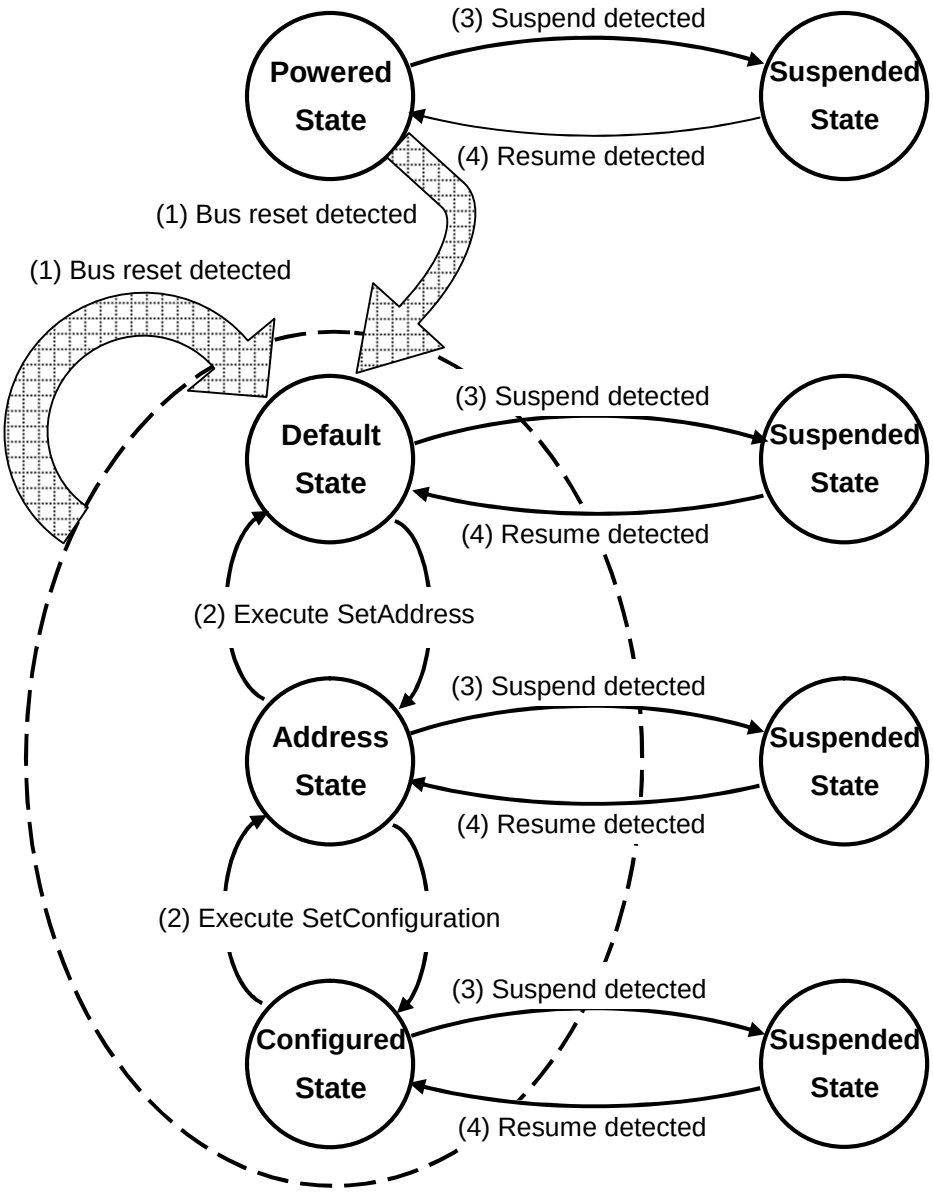
RAM Address	0	1	2	3	4	5	6	7
0x200	EP0Rx	EP6BK0	EP6BK0					
0x210		EP6BK1						
0x220		EP5BK0	EP6BK1					
0x230		EP5BK1						
0x240	EP0Tx	EP4BK0	EP5BK0	EP6BK0				
0x250		EP4BK1						
0x260		EP3BK0	EP5BK1	EP6BK1				
0x270		EP3BK1						
0x280	EP1BK0 EP6BK0	EP2BK0	EP4BK0	EP5BK0				
0x290								
0x2A0			EP4BK1	EP5BK1	EP6BK0			
0x2B0					EP6BK1			
0x2C0	EP1BK1 EP6BK1	EP2BK1	EP3BK0	EP4BK0	EP5BK0			
0x2D0					EP5BK1			
0x2E0			EP3BK1	EP4BK1	EP4BK0			
0x2F0					EP4BK1			
0x300	EP2BK0 EP5BK0	EP1BK0	EP2BK0	EP3BK0	EP3BK0	EP6BK0		
0x310						EP6BK1		
0x320			EP2BK1	EP3BK1	EP3BK1	EP5BK0		
0x330						EP5BK1		
0x340	EP2BK1 EP5BK1	EP1BK1	EP1BK0	EP2BK0	EP2BK0	EP4BK0		
0x350						EP4BK1	EP6BK0	
							EP6BK1	
0x360			EP1BK1	EP2BK1	EP2BK1	EP3BK0	EP5BK0	
							EP5BK1	
0x370						EP3BK1	EP4BK0	
							EP4BK1	
0x380	EP3BK0 EP4BK0	EP0Rx	EP0Rx	EP1BK0	EP1BK0	EP2BK0	EP3BK0	
0x390						EP2BK1	EP3BK1	EP6BK1
								EP5BK1
0x3A0				EP1BK1	EP1BK1	EP1BK0	EP2BK0	EP4BK1
								EP3BK1
0x3B0						EP1BK1	EP2BK1	EP2BK1
								EP1BK1
0x3C0	EP3BK1 EP4BK1	EP0Tx	EP0Tx	EP0Rx	EP0Rx	EP0Rx	EP1BK0	EP6BK0
								EP5BK0
0x3D0							EP1BK1	EP4BK0
								EP3BK0
0x3E0				EP0Tx	EP0Tx	EP0Tx	EP0Rx	EP2BK0
								EP1BK0
0x3F0							EP0Tx	EP0Rx
								EP0Tx

Table 1-10 Endpoint Buffer RAM Mapping

Chapter 2. State Transitions and Interrupts

2.1. Device State Transition

Devices have states and make transitions between them. The figure below shows a USB device state transition diagram for this microcontroller. (Refer to 9.1,"USB Device States", of USB 2.0 Specification).



<Transition source>	<Interrupt used>
(1) Bus reset detected	Bus reset interrupt
(2) Set Address Set Configuration	Endpoint 0 Interrupt
(3) Suspend detected	Suspend interrupt
(4) Resume detected	Bus active interrupt

Figure 2-1 Device State Transitions and Transition Source

2.1.1. Device States

The LC87F1M16A manages the device state transitions among Powered, Default, Address, Configured, and Suspended under program control. The user should set up and control variables as necessary. Table 2-1 summarizes the major actions that the LC87F1M16A takes in every state, interrupts (transition sources) that are enabled in that state, and the next state that is entered on the corresponding source).

State Major Actions	Enabled Interrupt (Transition Source)	Next State
Powered Waits for a bus reset.	Bus reset	Default
	Suspend (Suspended state)	Suspended
Default Responds to address 0 and has an address assigned.	Bus reset	Default
	Suspend (Suspended state)	Suspended
	Endpoint 0 SetAddress(0)	Default
	SetAddress(1-127)	Address
Address Responds only to the assigned address and has a configuration assigned.	Bus reset	Default
	Suspend (Suspended state)	Suspended
	Endpoint 0 SetAddress(0)	Default
	SetAddress(1-127)	Address
	SetConfiguration(Not 0)	Configured
	SetConfiguration(0)	Address
Configured Performs usual USB communication. (Specified configuration gets available.)	Bus reset	Default
	Suspend (Suspended state)	Suspended
	Endpoint 0 SetConfiguration(nonzero)	Configured
	SetConfiguration(0)	Address
Suspended Stops USB and waits for a bus active.	Bus active	Return to original state

Table 2-1 Device States and Transition Sources (1)

The table below lists the source and destination states associated with the transition sources.

Source State	Transition Source	Interrupt Used	Destination State
Powered Default Address Configured	(1) Bus reset detected	Bus reset	Default
(Default) Address	(2) SetAddress(0)	Endpoint 0	Default
Default (Address)	(2) SetAddress(1-127)		Address
(Address) Configured	(2) SetConfiguration(0)		Address
Address (Configured)	(2) SetConfiguration (Not 0)		Configured
Powered Default Address Configured	(3) Suspend detected	Suspend	Suspended
Suspended	(4) Resume detected	Bus active	State entered before suspension

Table 2-2 Device States and Transition Sources (2)

Note: The numbers (1)-(4) in the column "Transition Source" of Table 2-2 correspond to the numbers in Figure 2-2.
The transition source number (2) identifies requests from the host. Their meanings are given below (the requests other than those described here do not become the source of any device state transition).

- SetAddress (0) .....The host specified the USB address 0.
- SetAddress(1-127) .....The host assigned a USB address 1-127.
- SetConfiguration(0) .....The host specified configuration value 0.
- SetConfiguration(Not 0) .....The specified a nonzero configuration value.

(An error will be raised if a configuration value that is not specified in the configuration descriptor is encountered.)

2.1.2. Transition Sources

The details of the device state transition sources (1)-(4) shown in Figure 2-2 and Table 2-2 are given below.

(1) Bus reset detected (bus reset interrupt)
The host issues a reset command when it recognizes a new device connected or when it initializes the devices. The device recognizes a reset condition when the USB bus line stays in the SE0 state for 2.5 microseconds. The reset device enters the initialized state and transits into the Default state. Since this microcontroller generates a USB bus reset interrupt on detection of a reset, the bus reset interrupt processing routine must initialize the device and place it into the Default state (see 2.4, "USB Bus Reset Interrupt").

(2) SetAddress/SetConfiguration executed
SetAddress is a request that is issued by the host when assigning a USB address to a device. When the device is assigned an address (1 to 127) through this request, it enters the Address state. (The device transits into the Default state when address 0 is given since address 0 is the default address.)
SetConfiguration is a request that is issued by the host when specifying a configuration value. This request configures the device as specified and places it into the Configured state (the device enters the Address state if a configuration value of 0 is specified).
When this microcontroller receives this request, it generates an endpoint 0 interrupt. Consequently, the request must be processed during endpoint 0 processing, after which the device must switch into the pertinent state (see Section 2.5, "Endpoint 0 Interrupt").

(3) Suspend detected (base timer interrupt)
The device is regarded as being suspended when it stays in the idle state on the USB bus line for 3 milliseconds or longer. When this microcontroller detects a suspended state, it generates a suspend interrupt. The suspend interrupt processing routine must place it into the Suspend state (see Section 2.6, "Suspend Interrupt").

(4) Resume detected (Bus active interrupt, remote wakeup)
The device is regarded as being resumed if a bus activity occurs when the device is in the Suspended state. Restoration from the Suspended state can be accomplished either by a USB bus active interrupt (resume detected) or by remote wakeup (resume transmitted).
This microcontroller regards an activity (packet transmission or reset) as being initiated by the host and generates a bus active interrupt when the USB bus line state changes from the idle state. The device must be returned into the original state that it was in before the Suspended state by the bus active interrupt processing routine. If the remote wakeup feature is available, it is possible to send the resume signal and switch the device into the state before the Suspended state using the interrupt processing routine associated with the source of wakeup (see Sections 2.7, "USB Bus Active Interrupt" and 2.8, "Remote Wakeup").

2.2. Bus Enumeration

The host identifies and controls the state change of USB devices through a process called "bus enumeration" when connecting or disconnecting them. The enumeration process of the device is shown in Figure 2-2.

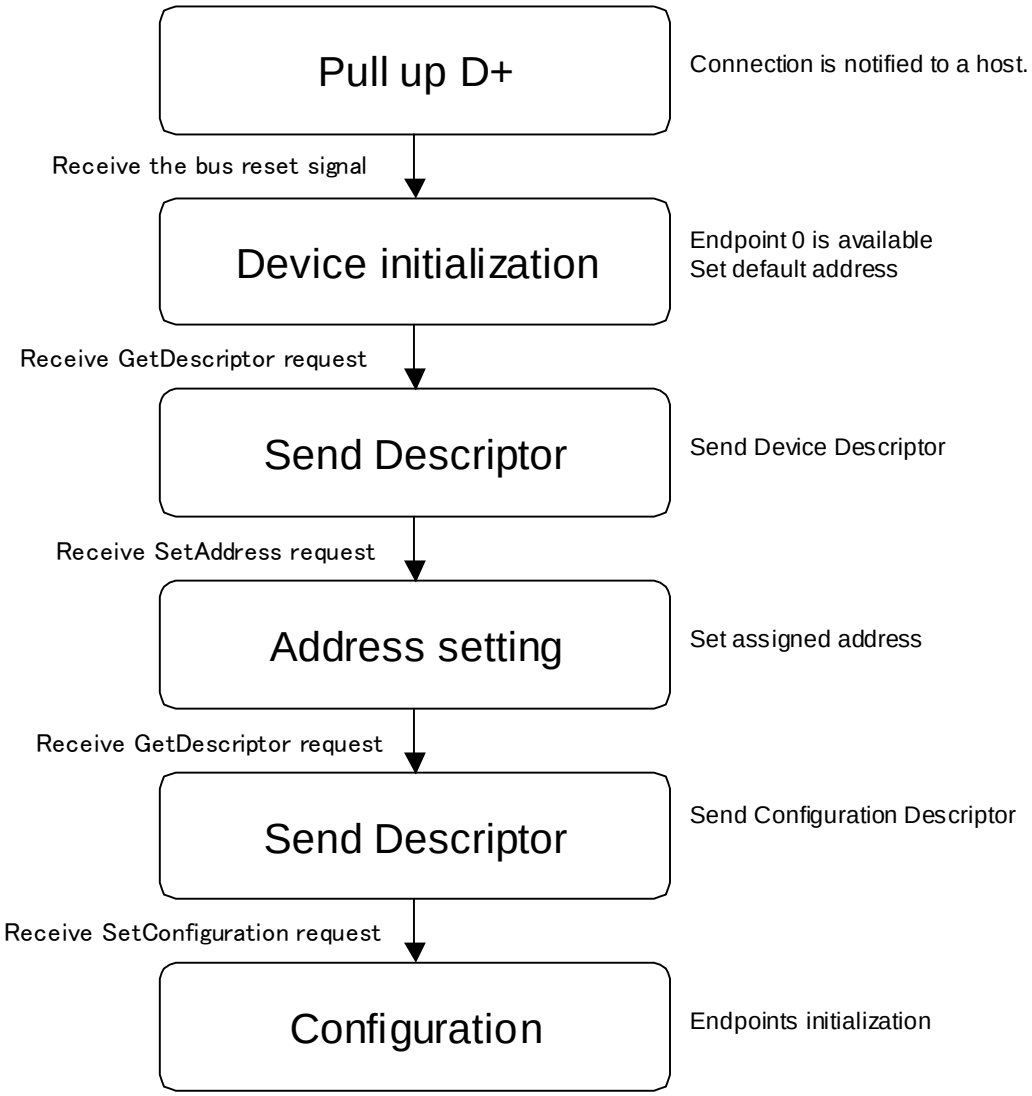


Figure 2-2 Outline of the bus enumeration process (device side)

The figure below shows the outline of firmware processing from the time a USB cable is connected till the time the device's primary functionality gets available.

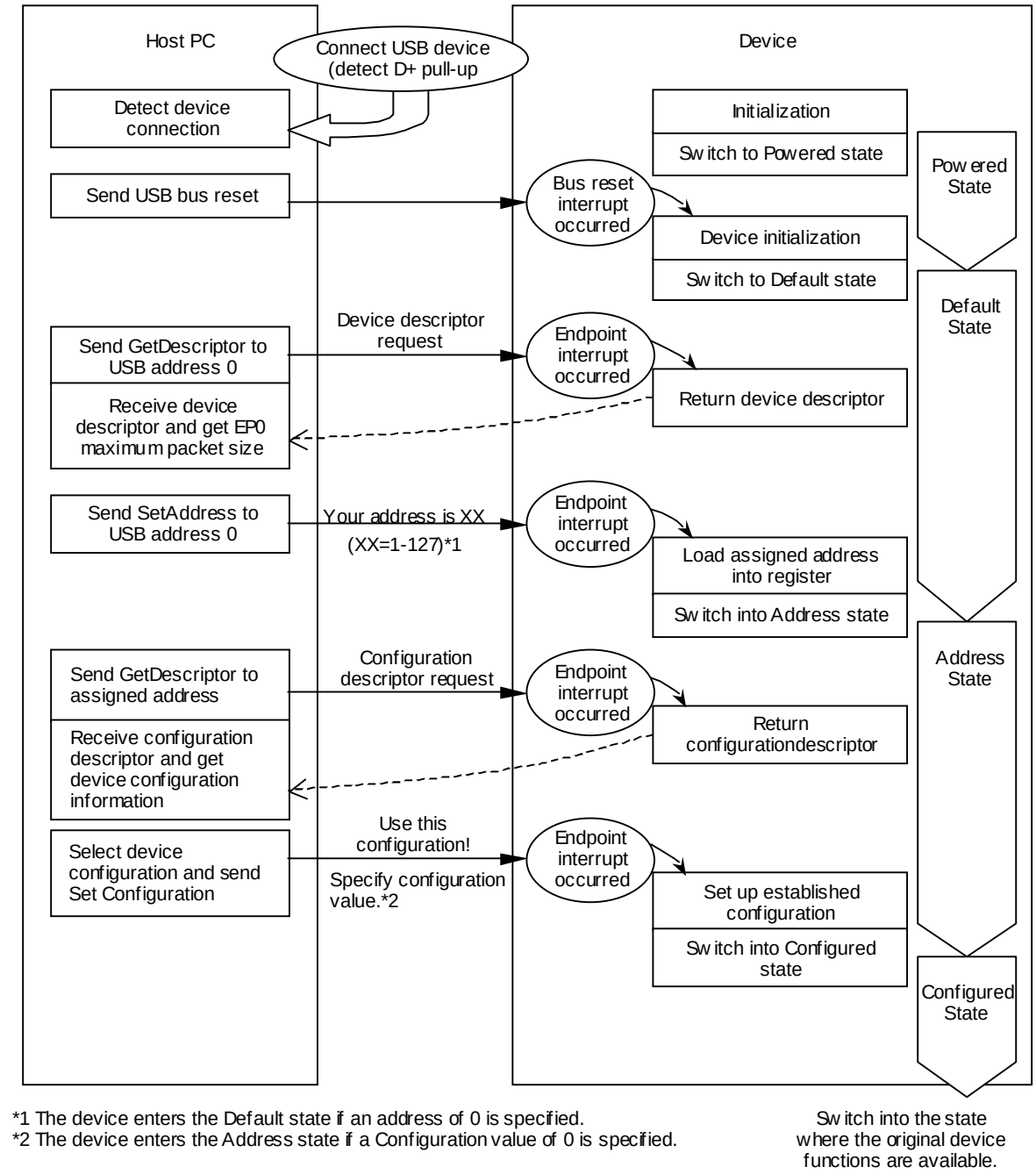


Figure 2-3 Example of Bus Enumeration Processing

2.3. USB Interrupts

2.3.1. Overview

The interrupts associated with the USB functions are used for flow control of data transfers and for device state management of the USB. The interrupts used for the USB are classified as follows:

Vector	Interrupt Name		Flag Detected	Enable Flag	Source
0013h	USB bus active		***BACFG***	***BACEN***	Bus active detected
0033h	USB bus reset		***BRSFG***	***BRSEN***	Bus reset detected
	USB suspend		***IDLFG***	***IDLEN***	Suspend detected
003Bh	SOF		***SOFFG***	***SOFEN***	SOF detected
	EP0INT	ACK	***AC0FG***	***AC0EN***	EP0 transaction ends with ACK.
		NAK	***NK0FG***	***NK0EN***	EP0 transaction ends with NAK.
		Error	***ER0FG***	***ER0EN***	Error detected in EP0 transaction.
		STALL	***ST0FG***	***ST0EN***	EP0 transaction ends with STALL.
	EP1INT	ACK	***AK1FG***	***AK1EN***	EP1 transaction ends with ACK.
		NAK	***NK1FG***	***NK1EN***	EP1 transaction ends with NAK.
		Error	***ER1FG***	***ER1EN***	Error detected in EP1 transaction.
		STALL	***ST1FG***	***ST1EN***	EP1 transaction ends with STALL.
	EP2INT	ACK	***AK2FG***	***AK2EN***	EP2 transaction ends with ACK.
		NAK	***NK2FG***	***NK2EN***	EP2 transaction ends with NAK.
		Error	***ER2FG***	***ER2EN***	Error detected in EP2 transaction.
		STALL	***ST2FG***	***ST2EN***	EP2 transaction ends with STALL.
	EP3INT	ACK	***AK3FG***	***AK3EN***	EP3 transaction ends with ACK.
		NAK	***NK3FG***	***NK3EN***	EP3 transaction ends with NAK.
		Error	***ER3FG***	***ER3EN***	Error detected in EP3 transaction.
		STALL	***ST3FG***	***ST3EN***	EP3 transaction ends with STALL.
	EP4INT	ACK	***AK4FG***	***AK4EN***	EP4 transaction ends with ACK.
		NAK	***NK4FG***	***NK4EN***	EP4 transaction ends with NAK.
		Error	***ER4FG***	***ER4EN***	Error detected in EP4 transaction.
		STALL	***ST4FG***	***ST4EN***	EP4 transaction ends with STALL.
	EP5INT	ACK	***AK5FG***	***AK5EN***	EP5 transaction ends with ACK.
		NAK	***NK5FG***	***NK5EN***	EP5 transaction ends with NAK.
		Error	***ER5FG***	***ER5EN***	Error detected in EP5 transaction.
		STALL	***ST5FG***	***ST5EN***	EP5 transaction ends with STALL.
	EP6INT	ACK	***AK6FG***	***AK6EN***	EP6 transaction ends with ACK.
		NAK	***NK6FG***	***NK6EN***	EP6 transaction ends with NAK.
		Error	***ER6FG***	***ER6EN***	Error detected in EP6 transaction.
		STALL	***ST6FG***	***ST6EN***	EP6 transaction ends with STALL.

Table 2-3 USB interrupts list

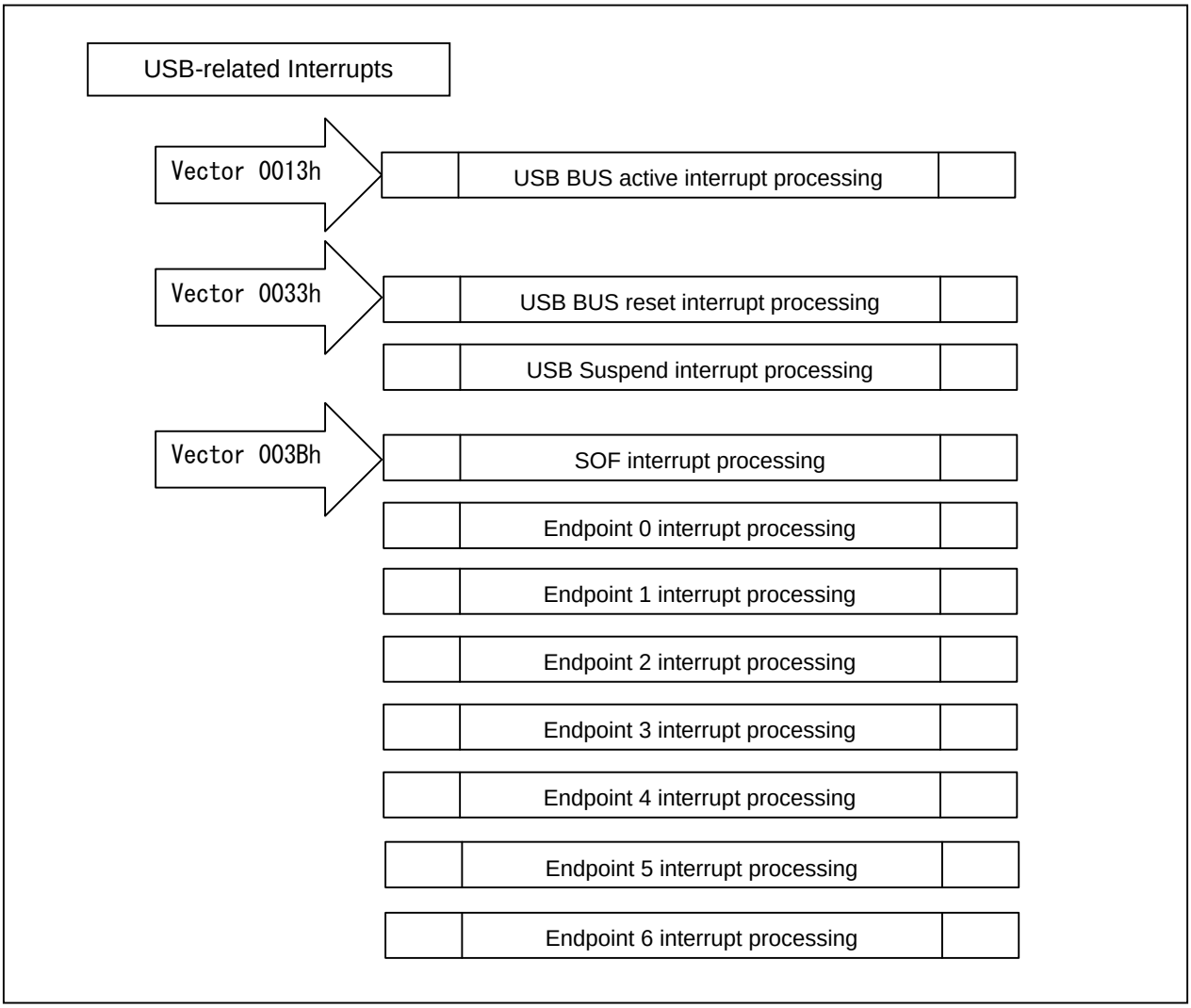


Figure 2-4 USB-related Interrupts

2.3.2. Related Registers

Shown below are the interrupt-related registers that are used by the USB functions.

Symbol	Address	R/W	Function
USCTRL	FE80h	R/W	Include Suspend interrupt control
USBINT	FE82h	R/W	USB interrupt control
EP0INT	FE83h	R/W	Endpoint 0 interrupt
EP1INT	FE84h	R/W	Endpoint 1 interrupt
EP2INT	FE85h	R/W	Endpoint 2 interrupt
EP3INT	FE86h	R/W	Endpoint 3 interrupt
EP4INT	FE87h	R/W	Endpoint 4 interrupt
EP5INT	FE88h	R/W	Endpoint 5 interrupt
EP6INT	FE89h	R/W	Endpoint 6 interrupt
IE	FE08h	R/W	Interrupt control
IP	FE09h	R/W	Interrupt level switching

Table 2-4 USB interrupts related Registers

●USB operation control register : USCTRL (FE80h)

Bit	Bit Name	R/W	Function
3	***IDLFG***	R/W	Suspend detection flag Set when a suspend condition (staying in bus idle state for 3 ms or longer) is detectedThis flag must be cleared with an instruction.
2	***IDLEN***	R/W	Suspend interrupt request enable flag 1: Enables USB suspend interrupts. 0: Disables USB suspend interrupts.

Table 2-5 USB operation control register Function list

●USB interrupt register: **USBINT** (FE82h)

Bit	Bit Name	R/W	Function
7	***BRSFG***	R/W	USB bus reset interrupt flag Set when a USB bus reset is detected. This flag must be cleared with an instruction.
6	***BRSEN***	R/W	USB bus reset interrupt enable flag 1: Enables USB bus reset interrupts. 0: Disables USB bus reset interrupts.
5	***BACFG***	R/W	USB bus active interrupt flag Set when a USB bus active condition is detected. This flag must be cleared with an instruction.
4	***BACEN***	R/W	USB bus active interrupt enable flag 1: Enables USB bus active interrupts. 0: Disables USB bus active interrupts.
3	***SOFFG***	R/W	SOF interrupt flag Set when an SOF condition is detected. This flag must be cleared with an instruction.
2	***SOFEN***	R/W	SOF interrupt enable flag 1: Enables SOF interrupts. 0: Disables SOF interrupts.
1	**-**	-	Reserved
0	***ENPEN***	R/W	Endpoint interrupt enable flag 1: Enables endpoint n (n=1-6) interrupts. 0: Disables endpoint n (n=1-6) interrupts.

Table 2-6 USB interrupt register Function list

●Endpoint n interrupt register : **EPnINT (n=0-6)**

(FE83h/FE84h/FE85h/FE86h/FE87h/FE88h/FE89h)

Bit	Bit Name	R/W	Function
7	***AKnFG***	R/W	Endpoint n ACK interrupt flag Set when the transaction terminates with an ACK. This flag must be cleared with an instruction.
6	***AKnEN***	R/W	End point n ACK interrupt enable flag 1: Enables endpoint n ACK interrupts. 0: Disables endpoint n ACK interrupts.
5	***NKnFG***	R/W	Endpoint n NAK interrupt flag Set when the transaction terminates with a NAK. This flag must be cleared with an instruction.
4	***NKnEN***	R/W	Endpoint n NAK interrupt enable flag 1: Enables endpoint n NAK interrupts. 0: Disables endpoint n NAK interrupts.
3	***ERnFG***	R/W	Endpoint n error interrupt flag Set when an error occurs in the transaction. This flag must be cleared with an instruction.
2	***ERnEN***	R/W	Endpoint n error interrupt enable flag 1: Enables endpoint n error interrupts. 0: Disables endpoint n error interrupts.
1	***STnFG***	R/W	Endpoint n STALL interrupt flag Set when the transaction terminates with a STALL. This flag must be cleared with an instruction.
0	***STnEN***	R/W	Endpoint n STALL interrupt enable flag 1: Enables endpoint n STALL interrupts. 0: Disables endpoint n STALL interrupts.

Table 2-7 Endpoint n interrupt register Function list

2.4. USB Bus Reset Interrupt

(1) USB bus reset interrupt source

The USB bus reset interrupt occurs when a reset signal from the host is detected. A reset is detected when the bus lines (D+,D-) stays in the LOW state for 2.5 microseconds or longer.

(2) When a USB bus reset is detected

The USB bus reset interrupt flag (***BRSFG***) in **USBINT** (FE82h) is set when a USB bus reset is detected,

(3) To use USB bus reset interrupts

USB bus reset interrupts must be enabled for 4 device states: Powered, Default, Address, and Configured. To enable USB bus reset interrupts, set the USB bus reset interrupt enable flag (***BRSEN***) in **USBINT**.

(4) Conditions under which USB bus reset interrupts are accepted

A USB bus reset interrupt occurs when all of the following conditions are met:

- The USB block enable flag (***USBON***) is 1.
- The USB bus reset interrupt enable flag (***BRSEN***) is 1.
- The USB bus reset interrupt flag (***BRSFG***) is 1.
- The H/L level interrupt enable flag (***IE7***) is 1.

(5) USB bus reset interrupt processing

Execution transfers to vector address 0033h when a USB bus reset interrupt is accepted. The interrupt processing routine must reset (initialize) the USB registers and place the device into the Default state. The USB bus reset interrupt flag (***BRSFG***) must be cleared before execution returns from the interrupt processing routine. Note, however, that ***BRSFG*** remains reset until the bus reset state is released. After clearing ***BRSFG***, the interrupt processing routine must confirm that ***BRSFG*** is not set before returning control.

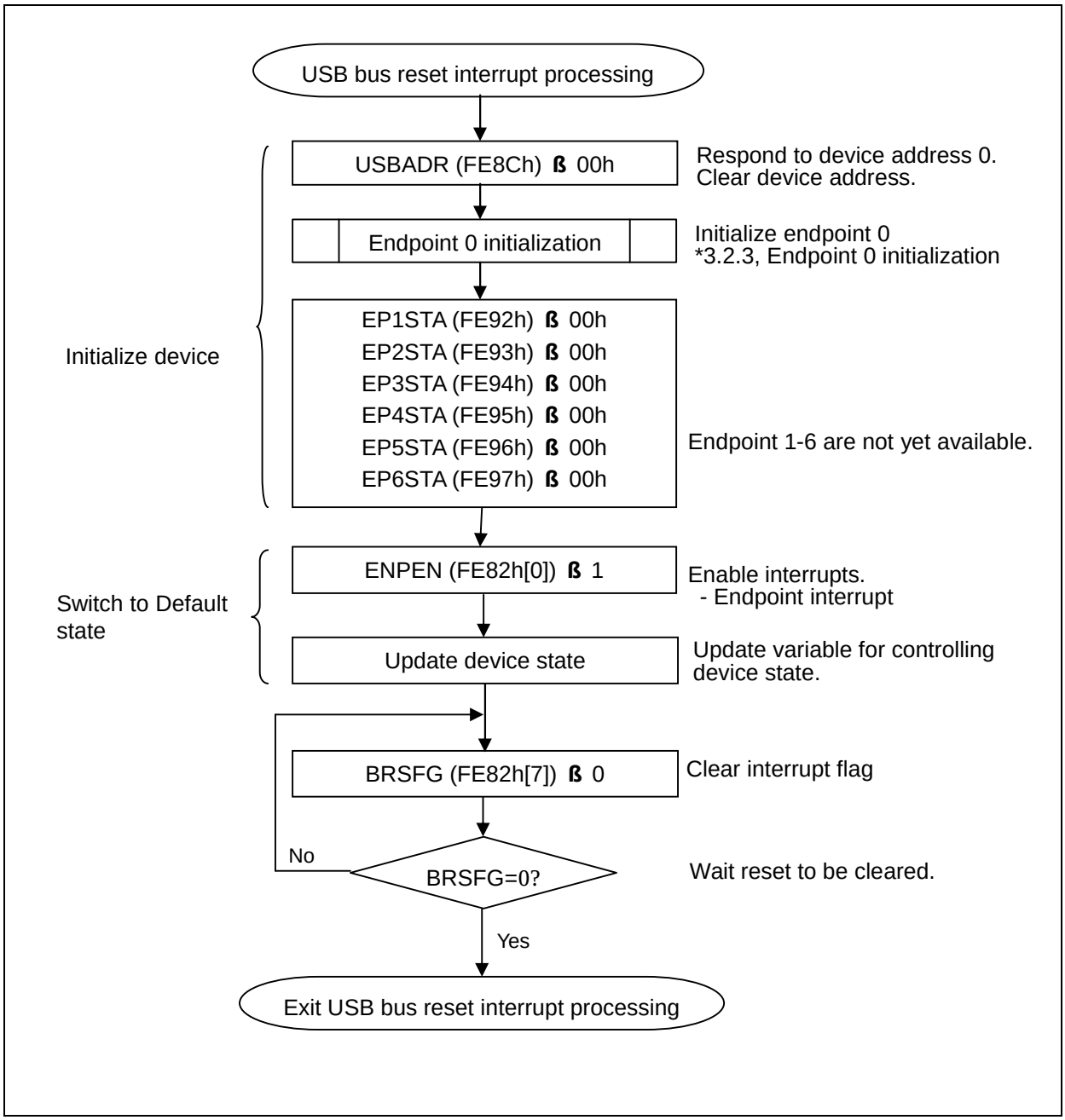


Figure 2-5 Example of USB Bus Reset Interrupt Processing

2.5. Endpoint Interrupts

There are four types of interrupts for endpoint n (n=0-6).
(1) Endpoint n ACK interrupt
(2) Endpoint n NAK interrupt
(3) Endpoint n error interrupt
(4) Endpoint n STALL interrupt

2.5.1. Endpoint n ACK interrupts

(1) Sources of endpoint n ACK interrupts

The endpoint n ACK interrupt occurs when a transaction terminates normally. Its source varies depending on the transfer type.

Transfer Type	Source
Control transfer data stage (IN) Control transfer status stage (IN) Bulk IN transfer Interrupt IN transfer	The device received an ACK handshake packet from the host after sending a data packet in response to an IN token.
Control transfer setup stage (SETUP) Control transfer data stage (OUT) Control transfer status stage (OUT) Bulk OUT transfer Interrupt OUT transfer	The device sent an ACK handshake packet after receiving a data packet following a SETUP/OUT token.
Isochronous IN transfer	The device sent a data packet in response to an IN token.
Isochronous OUT transfer	The device received an EOP packet following OUT token and data packet.

(2) When an endpoint n ACK interrupt is detected

The endpoint n ACK interrupt flag (***AKnFG***) in **EPnINT** (n=1-6) is set when a transaction terminates normally at endpoint n.

(3) To use endpoint n ACK interrupts

Endpoint 0 ACK interrupts must be enabled for device states Default, Address, and Configured. Endpoint n ACK interrupts (n=1-6) must be enabled for device states Configured. To enable endpoint n ACK interrupts, set the endpoint n ACK interrupt enable flag (***AKnFG***) in **EPnINT** and the endpoint interrupt enable flag (***ENPEN***) in **USBINT** (FE82h).

(4) Conditions under which endpoint n ACK interrupts are accepted

An endpoint n ACK interrupt occurs when all of the following conditions are met:

- The endpoint interrupt enable flag (***ENPEN***) is 1.
- The endpoint n ACK interrupt enable flag (***AKnEN***) is 1.
- The endpoint n ACK interrupt flag (***AKnFG***) is 1.
- The H/L level interrupt enable flag (***IE7***) is 1.

(5) Endpoint n ACK interrupt processing

Execution transfers to vector address 003Bh when an endpoint n ACK interrupt is accepted. The interrupt processing routine must clear the endpoint n ACK interrupt flag. It must also take necessary actions on each transaction according to the transfer flow and switch the device state as required.

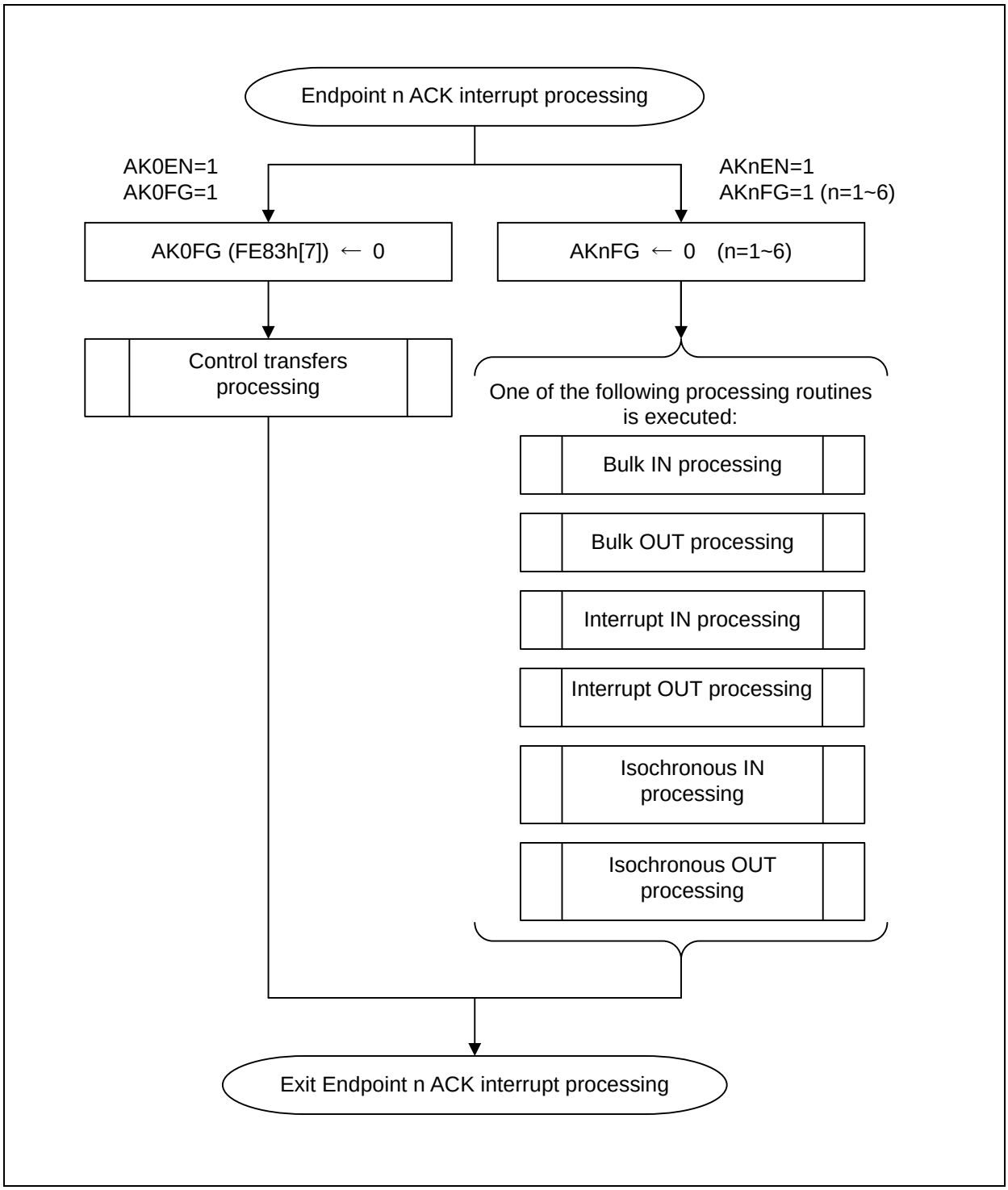


Figure 2-6 Example of Endpoint n ACK Interrupt Processing

2.6. Suspend Interrupts

(1) Source of USB Suspend detection

Suspend refers to an idle state that persists for 3 milliseconds or longer. A suspend condition is detected and the device is switched into the Suspended state when no activity is performed by the host (the state of the bus lines does not change from (D+,D-)=(1,0)) for not shorter than 3 milliseconds.

(2) When a suspend interrupt is detected

The suspend interrupt flag (***IDLFG***) in the USB operating control register (**USCTRL**) is set when the USB suspend is detected.

(3) To use the suspend interrupt

The suspend interrupt must be enabled for 4 device states: Powered, Default, Address, and Configured. The suspend interrupt can be enabled by settin the suspend interrupt request enable flag (***IDLEN***) in **USCTRL** (FE80h).

(4) Conditions under which suspend interrupt is accepted

A suspend interrupt occurs when all of the following conditions are met:

- The suspend interrupt request enable control flag (***IDLEN***) is 1.
- The suspend detectionf flag (***IDLFG***) is 1.
- The H/L level interrupt enable flag (***IE7***) is 1.

(5) Suspend Processing

When a suspend condition is detected, disable all interrupts that are enabled in the current state and enable interrupts that are used to return from the suspend condition. Subsequently, switch the device into Suspended state. In the Suspended state, the device can be run in the hold mode. When running the device in the hold mode, set the interrupt level of the interrupt for returning from the hold mode higher than that of the interrupt used to place the device into the hold mode. For example, to make the device enter the hold mode during the suspend interrupt processing routine, set the interrupt level of the interrupt to be used for returning from the suspend processing higher than that of the suspend interrupt. The interrupt level must be set using the master interrupt enable control register **IE** (FE08h) or the interrupt priority control register **IP** (FE09h).

(6) Returning from the Suspended state

The device can be returned from the Suspended state through the resume detection or remote wakeup function. For details, see Section 2.7, "USB Bus Active Interrupt (Resume Detected)" or Section 2.8 "Remote Wakeup."

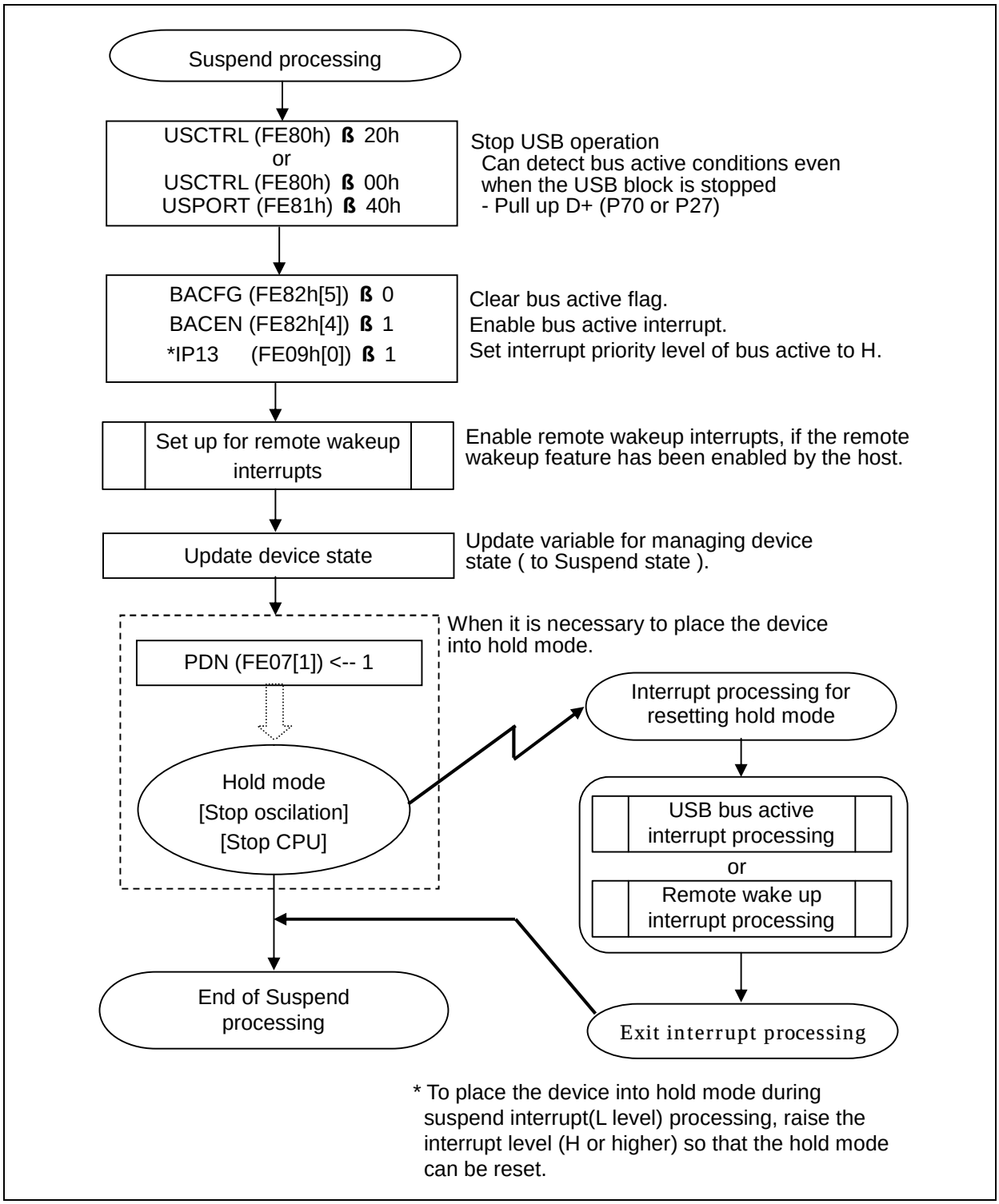


Figure 2-7 Example of Suspend Processing

2.7. USB Bus Active Interrupt (Resume Detected)

This microcontroller can detect the resume condition through the USB bus active interrupt. If the device is running in the hold mode in the Suspended state, the microcontroller can also reset the device from the hold mode through the USB bus active interrupt. To reset the hold mode, however, it is necessary to set the interrupt level of the USB bus active interrupt higher than that of the interrupt to be used when placing the device into the hold mode. This must be considered when the hold mode entry processing is to be executed during the interrupt processing routine. (No problem will arise when the hold mode entry processing is to be executed after the return from the interrupt processing).

(1) Source of USB bus active detection

A USB bus active interrupt occurs when an activity initiated by the host is detected. An activity is regarded as being detected when the bus lines change their state from the idle state (switch into the K or SE0 state).

(2) When a USB bus active interrupt is detected

The USB bus active interrupt flag (***BACFG***) in **USBINT** (FE82h) is set when a USB bus active interrupt is detected.

(3) To use USB bus active interrupts

USB bus active interrupts must be enabled in the Suspended state. USB bus active interrupts can be enabled by setting the USB bus active interrupt enable flag (***BACEN***) in **USBINT** (FE82h).

(4) Conditions under which USB bus active interrupts are accepted

A USB bus active interrupt occurs when all of the following conditions are met:

- The USB bus active interrupt enable flag (***BACEN***) is 1.
- The USB bus active interrupt flag (***BACFG***) is 1.
- The H/L level interrupt enable flag (***IE7***) is 1.

(5) USB bus active interrupt processing

Execution transfers to vector address 0013h when a USB bus active interrupt is accepted. The interrupt processing routine must clear the USB bus active interrupt flag (***BACFG***). It must also take preparatory actions to return to the original state from the Suspended state.
If the device is running in the hold mode, it is necessary to set up the clock to return to the normal mode. In this case, the routine must insert a wait before setting up the clock until the oscillation gets stabilized. Since the wait time varies depending on the oscillator to be used, an adequate wait time should be provided according to the characteristics of the oscillator.

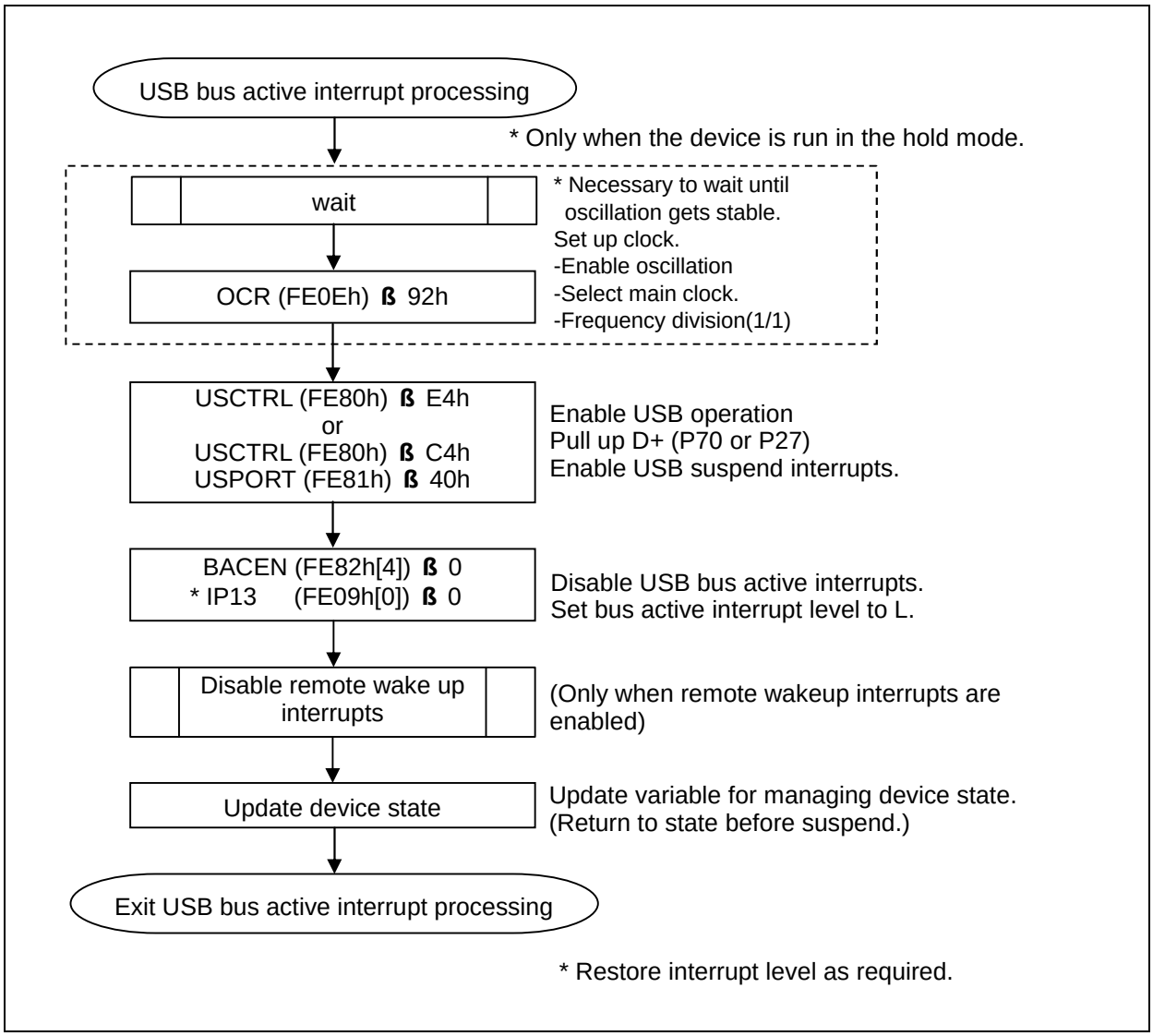


Figure 2-8 Example of USB Bus Active Interrupt Processing

2.8. Remote Wakeup

If the device wants to resume operation in the Suspended state on its own account, it must issue a resumption request to the host using the remote wakeup feature. The remote wakeup feature is not available until the host enables it. The remote wakeup feature is available only when the device is in the Configured state and it is enabled by the SetFeature request from the host.

(1) Remote wakeup interrupt

It is necessary to reset the hold mode if the device is running in the hold mode in the Suspended state. The following types of interrupts can be used to return the device from the hold to normal mode:

Interrupt Source	Pin	Interrupt Input Signal	Vector Address	Interrupt Level
INT0	P70*	L level	0003h	X or L
INT1	P71	H level	000Bh	X or L
INT2	P72	L edge	0013h	H or L
INT4	P20-P23	H edge		
INT5	P24-P27	Both edges	001Bh	H or L
Port 0	P00-P07	From "H" to "L"	004Bh	H or L

Table 2-8 Interrupts list for Returning from the Hold Mode

*When P70 is used to control “D+ pull-up”, it isn’t able to use as INT0.

(2) To use remote wakeup interrupts

Remote wakeup processing must be enabled only in the Suspended state. Remote wakeup interrupts can be enabled by setting the respective interrupt enable flags. For the hold mode to be reset successfully on a remote wakeup, it is necessary to set the interrupt level of the remote wakeup interrupt higher than that of the interrupt to be used when placing the device in the hold mode by manipulating the register (**IE** or **IP**). This must be considered when the device is to be placed into the hold mode during the interrupt processing routine.

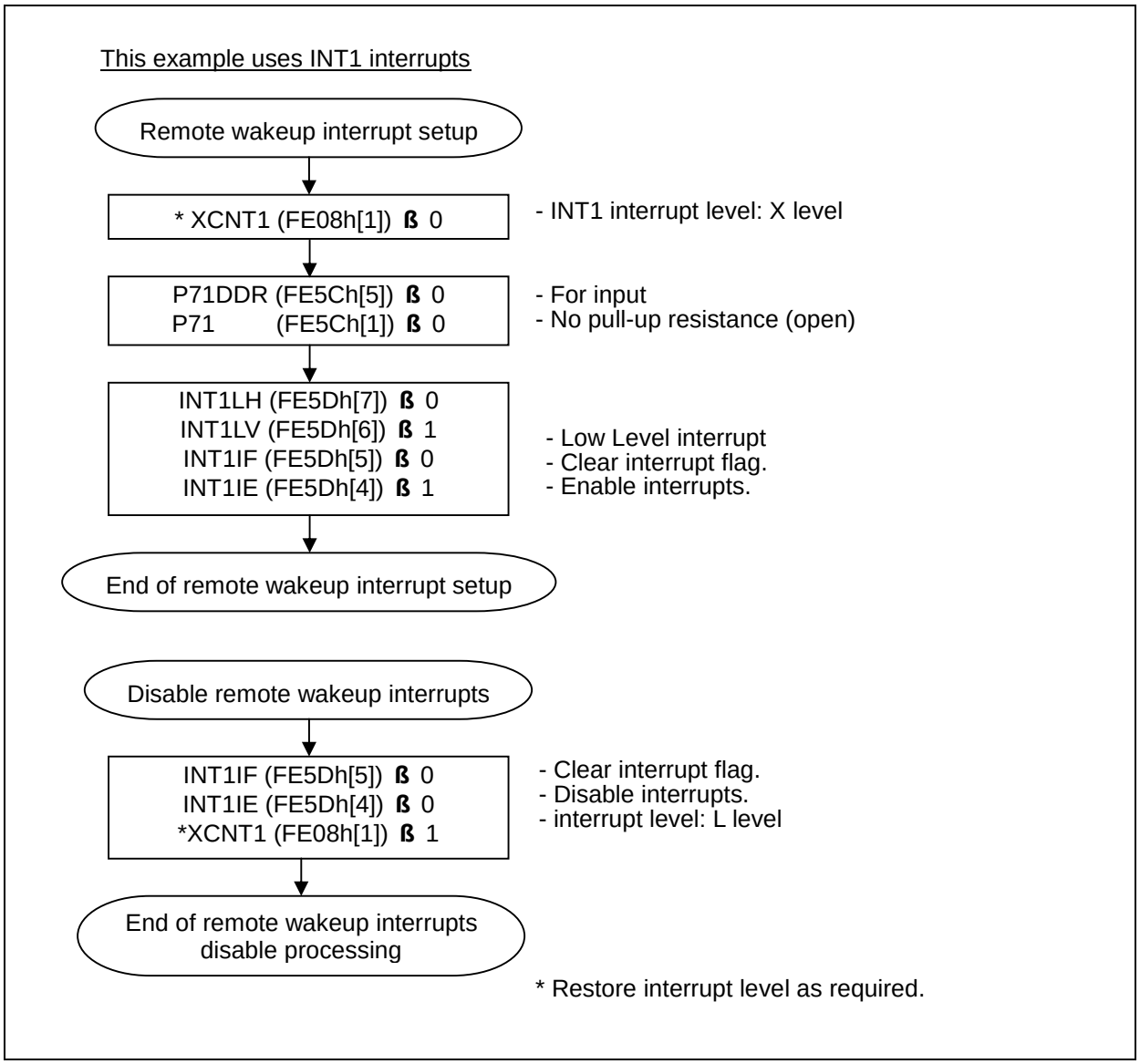


Figure 2-9 Example of Setting the Device for Remote Wakeup Interrupts (INT1 Interrupt)

(3) Remote wakeup interrupt processing

Execution transfers to the vector address to be used when a remote wakeup interrupt is accepted. The interrupt processing routine must clear the interrupt flag, disable further interrupts, and send a resume signal. In addition, it must take preparatory actions to return to the preceding state from the Suspended state. (It is only when the device switches its state from Configured to Suspended that the remote wakeup feature is available. Consequently, the destination is limited to the Configured state.) If the device is running in the hold mode, it is necessary to set up the clock to return to the normal mode at the beginning of the interrupt processing routine. In this case, the routine must insert a wait before setting up the clock until the oscillation gets stabilized. Since the wait time varies depending on the oscillator to be used, an adequate wait time should be provided according to the characteristics of the oscillator.

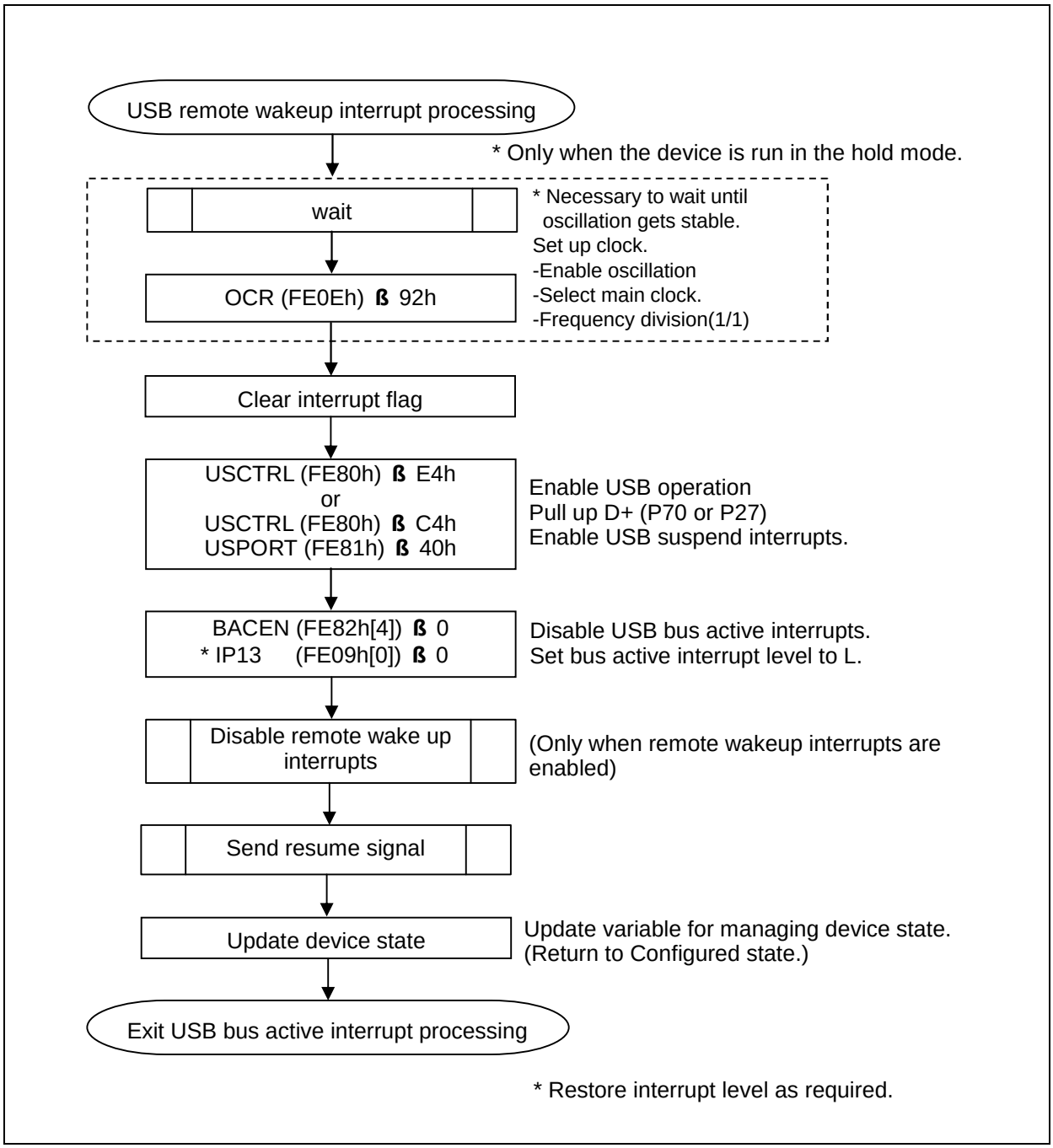


Figure 2-10 Example of Remote Wakeup Interrupt Processing

(4) Sending the resume signal

The procedure for sending the resume signal is explained below.

<1>Hold the idle state for 2 milliseconds.

The resume signal must be transmitted after interrupts are disabled by the remote wakeup interrupt processing routine. The signal, however, cannot be transmitted unless the bus stays in the idle state for at least 5 milliseconds. Since 3 milliseconds are required to identify the Suspended state, it is necessary to keep the bus in the idle state for at least 2 milliseconds before transmitting the resume signal (this ensures that the bus remains in the idle state for at least 5 milliseconds when a remote wakeup interrupt occurs immediately following the entry into the Suspended state). Use the base time or similar feature to count 2 milliseconds, and then proceed to the next step.

<2>Generating the resume signal

The resume signal places the USB bus lines in the (D+, D-)=(0,1) state (K state). To send data out of the USB port (D+, D-) of this microcontroller, clear the USB run flag (***USRUN***) in the USB block control register (**USCTRL**), then set the USB port control register (**USPORT**) to initiate generation of the K state.

<3>Hold the K state for 1 to 15 milliseconds.

When the device is to send the resume signal through the remote wakeup feature, it is necessary to maintain the USB bus in the K state for 1 to 15 milliseconds. Use the base timer or similar feature to count 1 to 15 milliseconds and proceed to the next step.

<4>End the transmission processing.

Set up the USB port control register (**USPORT**) so as to place the USB port (D+, D-) into the input mode and terminate the resume signal transmission processing. Subsequently, set the USB run flag (***USRUN***) in the block control register (**USCTRL**) to enable USB operations.

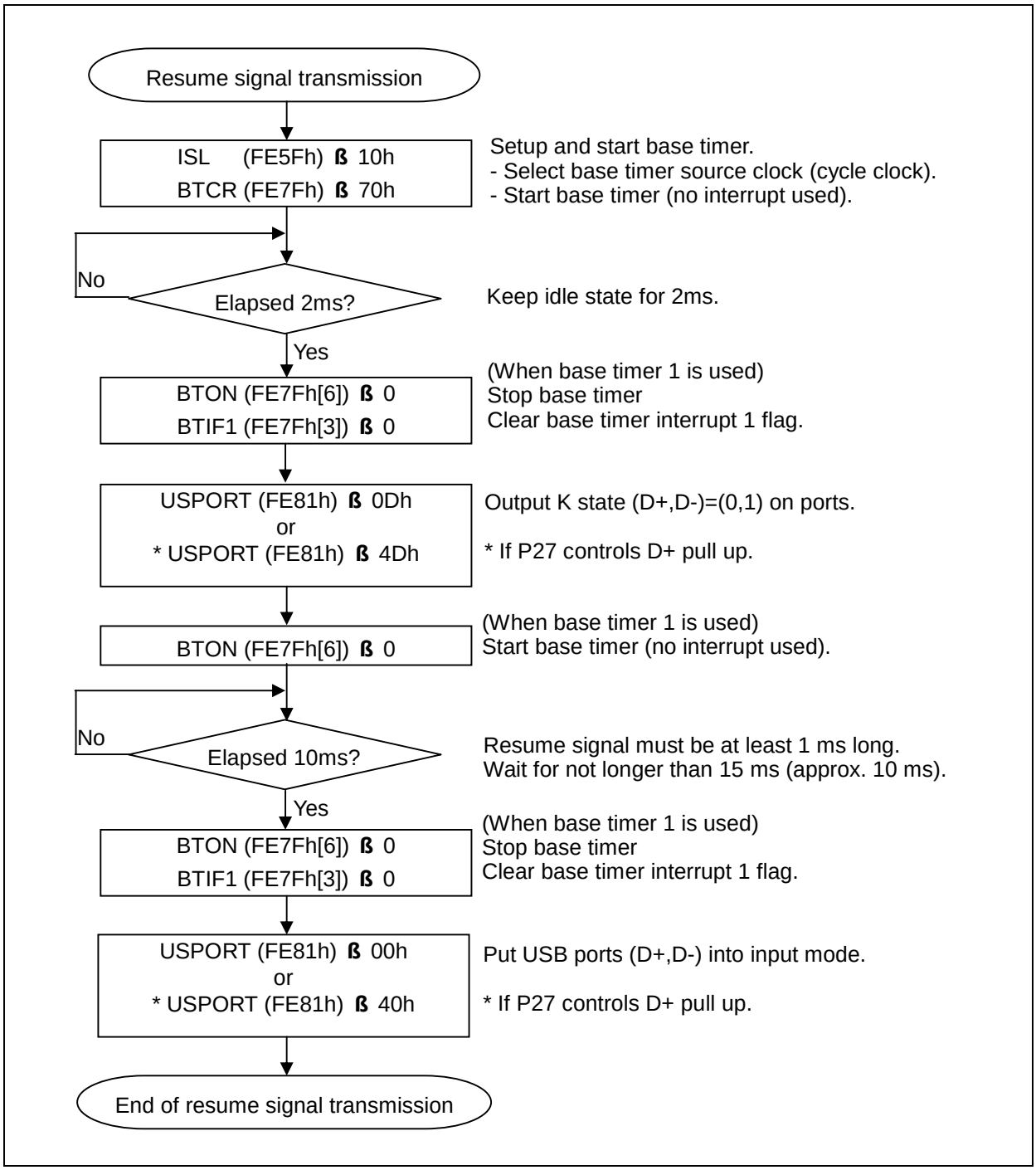


Figure 2-11 Resume Signal Transmission Processing

Chapter 3. Control Transfers

3.1. Endpoint 0 Control

Control transfers are bi-directional transfers that are used primarily to configure device. Endpoint 0 is used in control transfers. The buffers to be used for endpoint 0 transfers are mapped as transmission/receiving data areas to 8, 16, 32 or 64-bytes internal RAM areas, respectively. Control transfers are performed according to access to these transmission/receiving data areas (storing of transmitting data, and read-out of received data), and a setup of the related registers.

3.1.1. Related Registers

The registers that are related to control transfers are listed below.

Symbol	Address	R/W	Function
FRAMEL	FE8Ah	R	Frame number lower-order 8 bits
FRAMEH	FE8Bh	R	Frame number higher-order 3 bits
USBADR	FE8Ch	R/W	USB address management
EPINF0	FE8Dh	R	USB transfer information
EP0STA	FE8Eh	R/W	EP0 control
EP0MP	FE8Fh	R/W	EP0 maximum payload size
EP0RX	FE90h	R	EP0 receive data size
EP0TX	FE91h	R/W	EP0 transmit data size

Table 3-1 Control Transfer Related Registers

●Frame number low register: **FRMEL** (FE8Ah)

Bit	Bit Name	R/W	Function
7 - 0	***FRM07*** - ***FRM00***	R	Frame number lower-order 8 bits Loaded with the lower-order 8 bits of the frame number when an SOF is received.

●Frame number high register: **FRMEH** (FE8Bh)

Bit	Bit Name	R/W	Function
7 - 3	-	-	Reserved
2 - 0	***FRM08*** - ***FRM10***	R	Frame number higher-order 3 bits Loaded with the higher-order 3 bits of the frame number when an SOF is received.

●USB address register: **USBADR** (FE8Ch)

Bit	Bit Name	R/W	Function
7	***ADREN***	R/W	Address enable flag 1: Enables ADR6-ADR0. 0: Disables ADR6-ADR0 (responds to address 0).
6 - 0	***ADR6*** - ***ADR0***	R/W	Device address Stores the device address assigned by the host.

●Transfer information register: **EPINFO** (FE8Dh)

Bit	Bit Name	R/W	Function
7 - 4	***EPNO3*** - ***EPNO0***	R	Endpoint number Loaded with the endpoint number when an **EPnINT** (n=0-6) interrupt source is established.
3 2	***TKN1*** ***TKN0***	R	Token ID Loaded with the token ID when an **EPnINT** (n=0-6) interrupt source is established.

1 0	***CTKN1*** ***CTKN0***	R	Token ID for control transfer Loaded with the token ID when an **EP0INT** interrupt source is established.

●Endpoint 0 mode setup register: **EP0STA** (FE8Eh)

Bit	Bit Name	R/W	Function
7	***E0EN***	R/W	Endpoint 0 enable bit 1: Enables endpoint 0. 0: Disables endpoint 0.
6	***E0TGL***	R/W	Endpoint 0 data toggle bit This bit is inverted and the receive data is transferred to the buffer when this bit matches DATA PID (in the data receive mode). This bit is set to 1 in the SETUP receive mode. The data whose DATA PID matches this bit is transmitted (in the data transmit mode) and this bit is inverted when an ACK is received.
5	***E0OVR***	R/W	Endpoint 0 maximum payload size excess bit Set when the volume of data exceeding the maximum payload size is received. This bit must be cleared with an instruction.
4	***E0STL***	R/W	STALL bit A 1 in this bit causes STALL to be returned for tokens other than SETUP. The bit is automatically cleared after a SETUP token is received.
3	***E0ACK***	R/W	ACK bit (* See the transmission mode chart.) (When ***E0CSU***=0) 1: Performs ACK processing for IN/OUT tokens. 0: Returns NAK for IN/OUT tokens.
2	***E0CSU***	R/W	Control SETUP bit (* See the transmission mode chart.) Set when the Setup stage of a control transfer is completed. This bit is cleared when the Status stage of a control write transfer is completed.
1	***E0CST***	R/W	Control STATUS bit (* See the transmission mode chart.) A 1 in this bit identifies a status stage for a control transfer. This bit is cleared when the Setup stage is completed. The bit is set when the Status stage of a control read transfer is completed.
0	***E0CRW***	R/W	Control R/W bit (* See the transmission mode chart.) Cleared when the Setup stage is completed. 1: Identifies a control read transfer. 0: Identifies a control write transfer.

●Endpoint 0 maximum payload size register: **EP0MP** (FE8Fh)

Bit	Bit Name	R/W	Function
7	-	-	Reserved
6 - 0	***E0MP6*** - ***E0MP0***	R/W	Endpoint 0 maximum payload size Indicate the maximum payload size of endpoint 0.

●Endpoint 0 receive data size register: **EP0RX** (FE90h)

Bit	Bit Name	R/W	Function
7	-	-	Reserved
6 - 0	***E0RX6*** - ***E0RX0***	R	Endpoint 0 receive data size Indicate the size of endpoint 0 receive data.

●Endpoint 0 transmit data size register: **EP0TX** (FE91h)

Bit	Bit Name	R/W	Function
7	-	-	Reserved
6 - 0	***E0TX6*** - ***E0TX0***	R/W	Endpoint 0 transmit data size Indicate the size of endpoint 0 transmit data.

3.1.2. Endpoint 0

Endpoint 0 is mapped into 8, 16, 32 or 64-byte receiving and transmission RAM areas. Its payload size is indicated in a register for each packet. The size of received data is stored in **EP0RX** during a control write transfer. In this case, ***E0OVR*** is set if the volume of data exceeding the maximum payload size specified in **EP0MP** is received. For control read transfers, **EP0TX** must be loaded with the number of bytes to be transmitted. The bytes of data specified here are transmitted during the actual transmission processing.
The characteristics of endpoint 0 are summarized below.

Name	Payload Size	Corresponding Transfer
Endpoint 0 (reception)	(* Size of data received) Stored in **EP0RX**. (64 bytes maximum)	Control write
Endpoint 0 (transmission)	(* Size of data to be transmitted) Stored in **EP0TX**. (64 bytes maximum)	Control read

Table 3-2 Outline of Endpoint 0

Transfer Type	Maximum Packet Size	Packet Payload Size
Control	8, 16, 32, or 64 bytes	0-Maximum packet size (Up to the byte count specified by the host)

Table 3-3 Control Transfer Packet Sizes

Note: The actual payload size of the packets may be changed provided it does not exceed the endpoint 0 maximum packet size (bMaxPacketSize0) that is defined in the device descriptor. The total number of data bytes to be transmitted or received in the Data stage, however, must not exceed the size that is specified in the Setup stage (wLength in the device request). When setting up **EP0TX**, make sure that this size should not be exceeded. If data exceeding this size is received, a STALL must be returned in the next transaction. If the size of transmit data is smaller than the size specified by the host and is an integral multiple of the maximum packet size, transmit a packet of a 0 byte length as the last packet to identify the end of data.

3.2. Outline of Control Transfer

A control transfer is made up of 3 (or 2) stages called Setup stage, Data stage, and Status stage, and each stage is performed in order.

(1) Setup stage ---------- The host transmits a device request.

(2) Data stage (option) ----- The device transmits/receives data specified in the device request.

(3) Status stage ---------- The device reports the result of request processing.

3.2.1. Stage Transitions

The Setup stage always forms the first stage of a control transfer followed by the Data stage and Status stage. The presence or absence of the Data stage and its direction (IN/OUT) are specified during the Setup stage. According to the contents of the Setup stage, control transfers are divided into the following three types:

- Ø Control read transfer
  A control read transfer is made up of 3 stages that occur in the order of Setup, Data, and Status stages. The direction of the Data stage is from device to host (IN direction). In the Data stage, the data is sent in one or more IN transactions as required.
- Ø Control write transfer
  A control write transfer is made up of 3 stages that occur in the order of Setup, Data, and Status stages. The direction of the Data stage is from host to device (OUT direction). In the Data stage, the data is received in one or more OUT transactions as required.
- Ø Control no-data transfer
  A control no-data transfer is made up of 2 stages that occur in the order of Setup and Status stages. This transfer has no Data stage and its Status stage begins with an IN token.

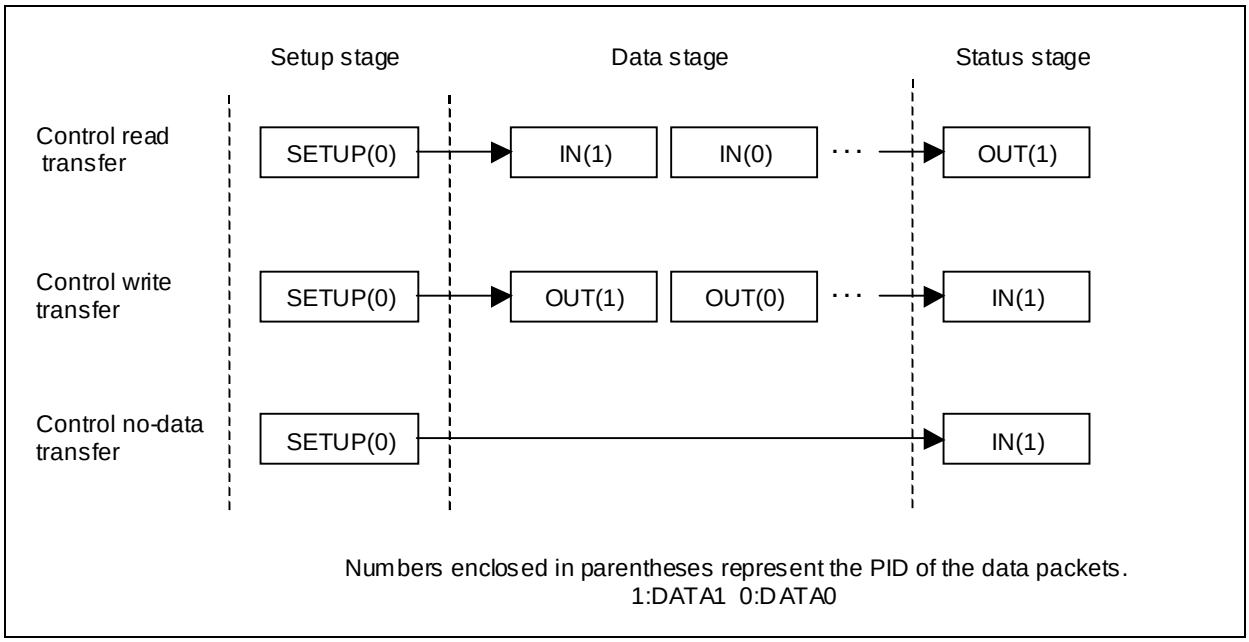


Figure 3-1 Control Transfer Stage Transitions

3.2.2. Transaction Configuration

u Setup stage

The Setup stage consists of one SETUP transaction. The SETUP transaction consists of token (SETUP), data (DATA0), and handshake packets. DATA0 PID is always used for the data packet and its content is always a device request (the size is fixed at 8 bytes). For a transaction that begins with a SETUP token packet, the device must always respond with ACK handshake packet.

Token	Data	Handshake
SETUP	**DATA0**(8byte)	**ACK**

u Data stage

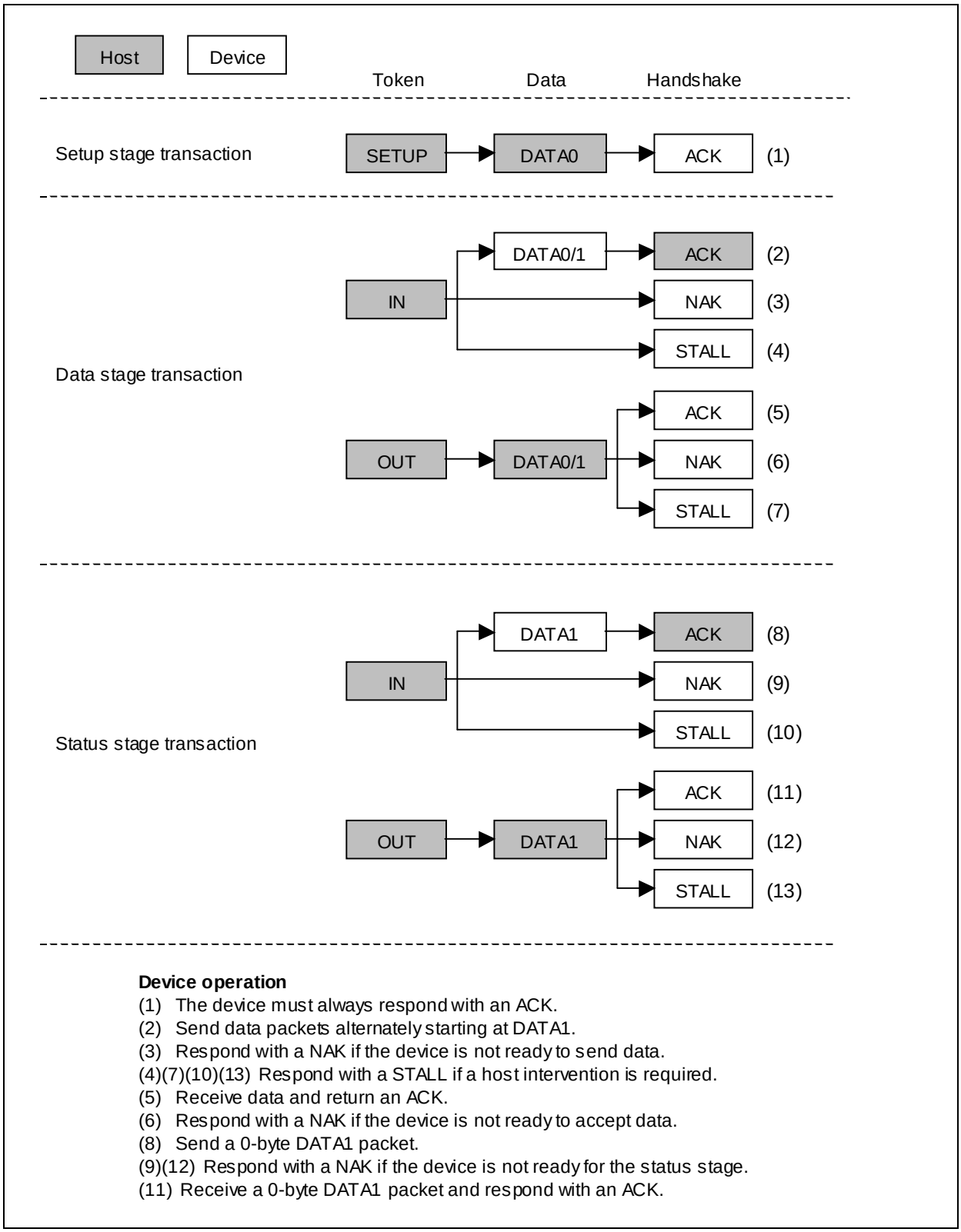
The Data stage, if present, consists of one or more IN/OUT transactions. IN/OUT transaction consists of token (IN/OUT), data (DATA1/0), and handshake packets. As many transactions as required according to the amount of data are repeated in the Data stage. DATA1 and DATA0 are used alternately as the PID of the data packets so that retransmitted and new data packets can be distinguished.

Token	Data	Handshake
IN	**DATA1/DATA0**	**ACK**
OUT	**DATA1/DATA0**	**ACK**

u Status stage

The Status stage consists of one IN/OUT transaction. IN/OUT transaction consists of token (IN/OUT), data (DATA1), and handshake packets. The token that is opposite to that which is used in the Data stage is used in the Status stage. The PID of the data packet used is DATA1 and its data size is 0 byte.

Token	Data	Handshake
IN	**DATA1**(0byte)	**ACK**
OUT	**DATA1**(0byte)	**ACK**

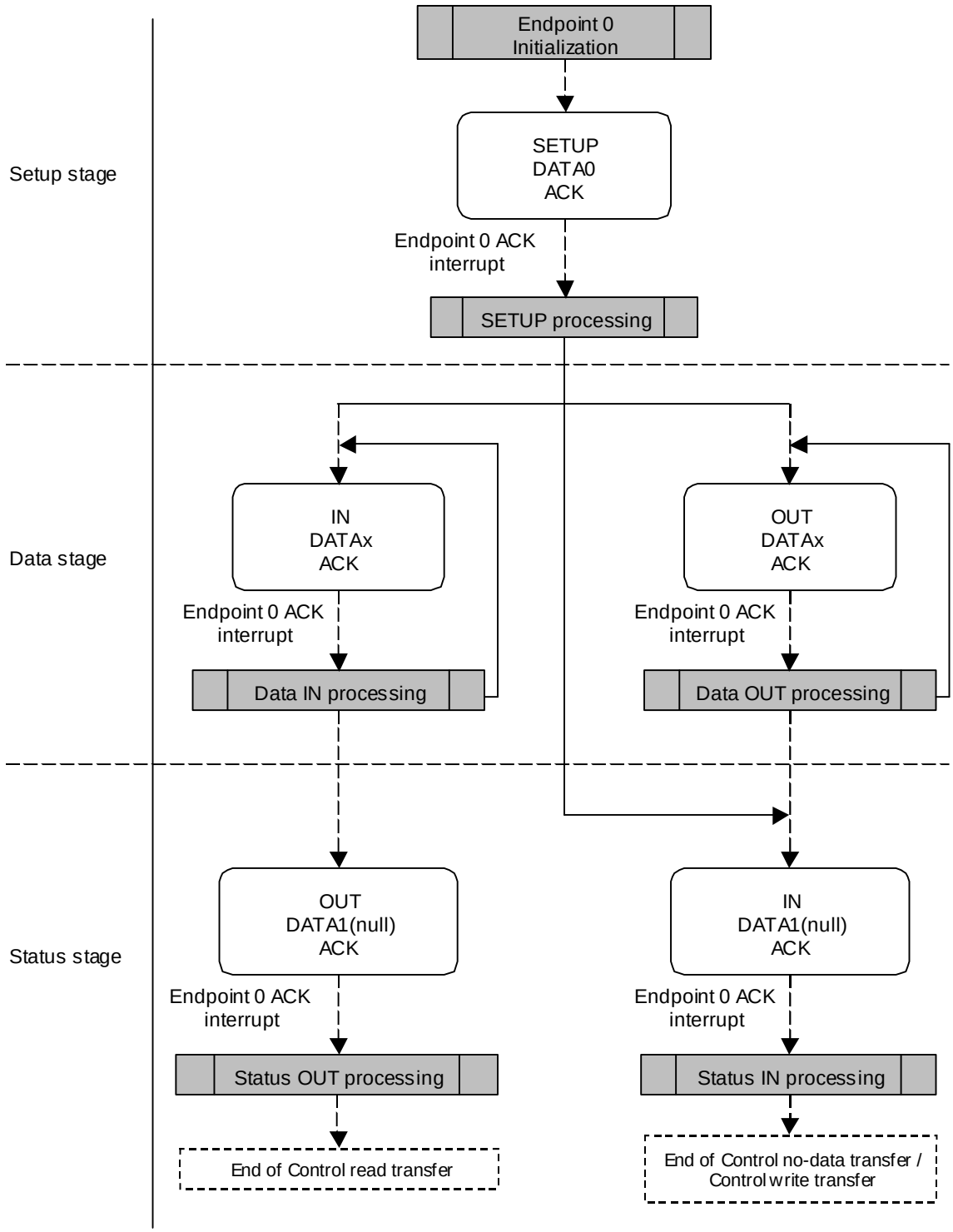


Device operation

(1) The device must always respond with an ACK.
(2) Send data packets alternately starting at DATA1.
(3) Respond with a NAK if the device is not ready to send data.
(4)(7)(10)(13) Respond with a STALL if a host intervention is required.
(5) Receive data and return an ACK.
(6) Respond with a NAK if the device is not ready to accept data.
(8) Send a 0-byte DATA1 packet.
(9)(12) Respond with a NAK if the device is not ready for the status stage.
(11) Receive a 0-byte DATA1 packet and respond with an ACK.

Figure 3-2 Stages of a Control Transfer

The type of control transfer is determined during the Setup stage, and the direction of transfer changes on the Data stage and the Status stage. The endpoint 0 operation for a token is performed according to the transmission mode set in **EP0STA**. Set up suitable transmission mode for each stage. See 3.2.5, "Setting Up the Transmission Mode". LC87F1M16A can achieve control transfer processing through the initialization for initiating control transfers and the processing of endpoint 0 ACK interrupt which occurs on each normal termination of each transaction.
The figure below shows the outline of control transfer processing.



3.2.3. Endpoint 0 Initialization

Endpoint 0 setup is required to perform a control transfer. When initializing the device as the result of a bus reset, be sure to perform endpoint 0 initialization. This application note assumes that endpoint 0 initialization is executed during bus reset interrupt processing. See Section 2.4, "USB Bus Reset Interrupts," for bus reset interrupt processing.

Since the first stage of a control transfer is the Setup stage (see Figure 3-1), the endpoint 0 initialization must be set up so that SETUP transactions for the Setup stage can be accepted. An example of endpoint 0 initialization processing is shown below.

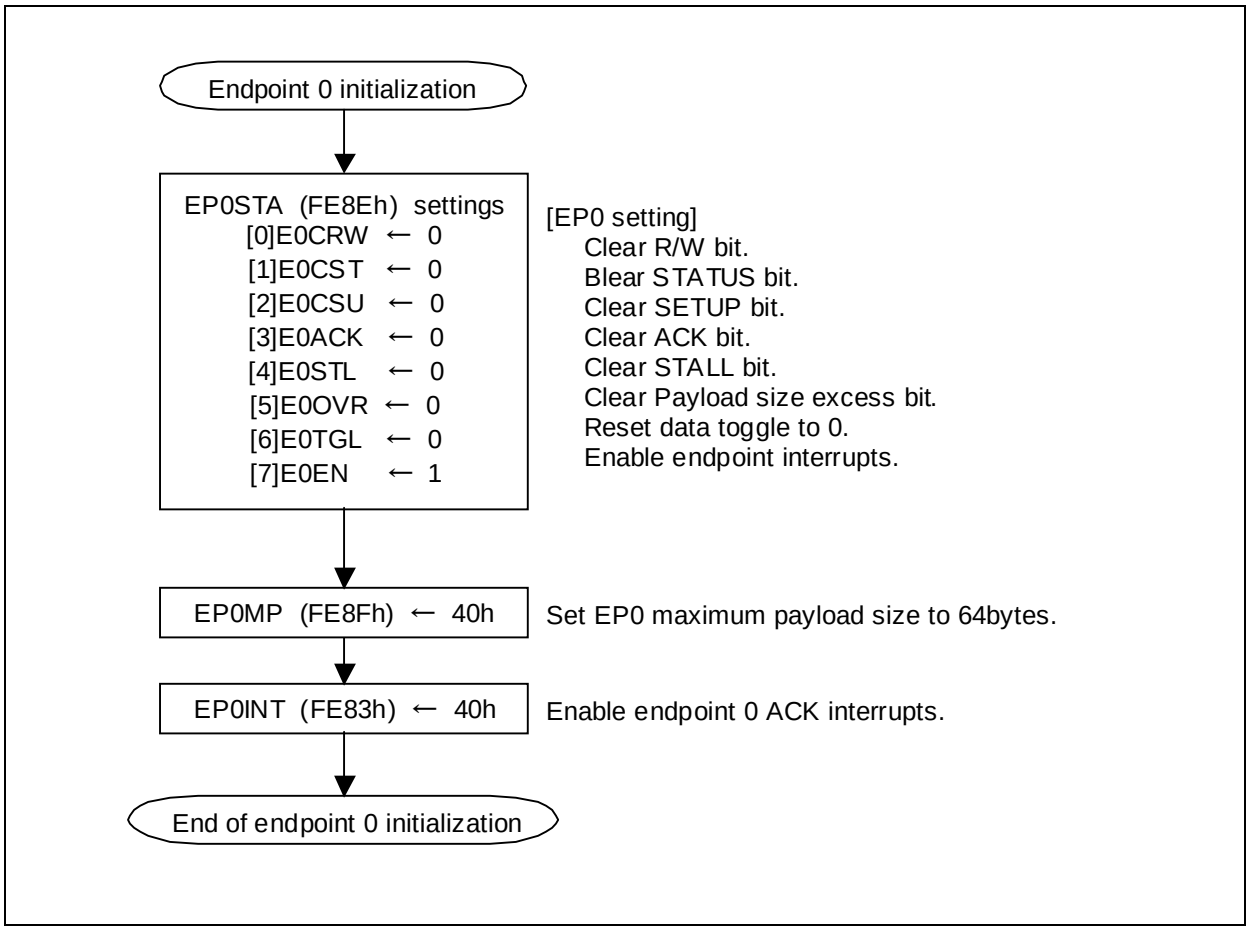


Figure 3-3 Example of Endpoint 0 Initialization Processing

3.2.4. Control Transfers Processing

An endpoint 0 ACK interrupt occurs at each normal termination of the Setup, Data, and Status stages. Since **EPINFO** (FE8Dh) is updated whenever an endpoint 0 ACK interrupt occurs, the Control transfers processing must be executed according to the token information that is stored.
The interrupt to be processed for the SETUP token is the one that occurs on completion of the SETUP transaction. The SETUP processing must be followed by the preparatory processing that is required to proceed to the next stage.
For the IN and OUT tokens, the lower-order 4 bits of **EP0STA** must be checked to identify the stage of the terminated transaction.

The Control transfers processing to be performed during the endpoint 0 ACK interrupts processing routine looks like as shown below.

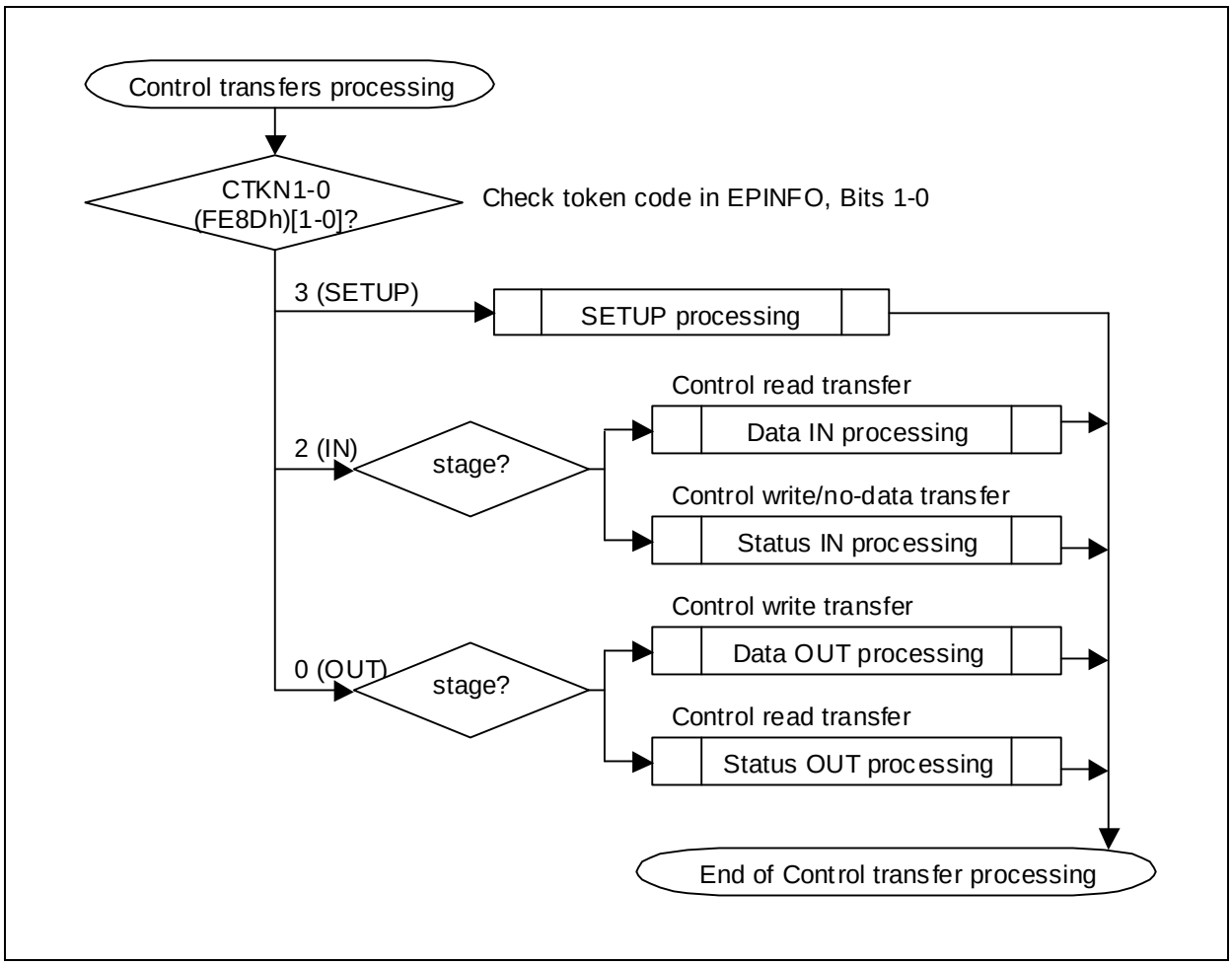
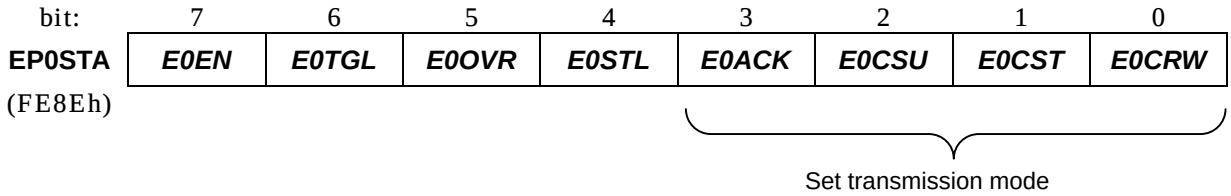


Figure 3-4 Example of Control Transfers Processing

3.2.5. Setting Up the Transmission Mode

The LC87F1M16A responds to control transfers according to the transmission mode defined in the **EP0STA**.



Bit	Name	Setting
7	***E0EN***	If this bit is set to 1, the endpoint 0 is enabled. The endpoint 0 must always be enabled in the stages following the Default state.
6	***E0TGL***	This bit need not be set during initialization since no toggle check is performed during the SETUP transaction. It is automatically set to 1 when the Setup stage is completed. Subsequently, the state of this bit is automatically inverted on each normal termination of the transactions in the Data stage. This bit has no meaning in the Status stage.
5	***E0OVR***	Set when the volume of data exceeding the maximum payload size is received.
4	***E0STL***	Set to 1 when a protocol violation occurs. If this bit is set when the endpoint 0 is enabled (***E0EN***=1), the endpoint 0 sends a STALL in response to an IN/OUT token regardless of the transmission mode setting. This bit is cleared when a SETUP token is received.
3	***E0ACK***	This bit has no meaning if the Setup stage is not completed (***E0CSU***=0). It is automatically reset to 0 when the Setup stage is completed. If this bit is 0, endpoint 0 responds with a NAK handshake in an IN/OUT transaction. If this bit is 1, the processing corresponding to the transmission mode is executed in an IN/OUT transaction.
2	***E0CSU***	It is automatically set to 1 when a Setup stage is complete. Since the device can respond to an IN/OUT tokens if this bit is 1, this bit must not be set by firmware. This bit is cleared when the Status stage of a control write transfer is completed.
1	***E0CST***	This bit has no meaning if the Setup stage is not completed (***E0CSU***=0). It is automatically reset to 0 when the Setup stage is complete. If this bit is 0, it identifies the Data stage of a control transfer. If 1, it identifies the Status stage. This bit must be set to 1 by firmware when a Data stage is complete. This bit is set when the Status stage of the control read transfer is completed.
0	***E0CRW***	This bit has no meaning if the Setup stage is not completed (***E0CSU***=0). It is automatically reset to 0 when the Setup stage is complete. If this bit is 0, it identifies a control write transfer. It must be set to 1 by firmware when a control read transfer is to be performed.

Transmission Mode Chart

	E0 ACK	***E0 CSU***	***E0 CST***	***E0 CRW***	Token	Receive Toggle	Receive Data	Response	ACK Interrupt
(1)	—	0	—	—	SETUP	—	Invalid	(Ignore)	
						—	Valid	Send ACK	○
					OUT	—	—	(Ignore)	
					IN			(Ignore)	
(2)	0	1	—	—	SETUP	—	Invalid	(Ignore)	
						—	Valid	Send ACK	○
					OUT	—	Invalid	(Ignore)	
						—	Valid	Send NAK	
					IN			Send NAK	
(3)	1	1	0	0	SETUP	—	Invalid	(Ignore)	
						—	Valid	Send ACK	○
					OUT	Not match	Invalid	(Ignore)	
							Valid	Send ACK	
						Match	Invalid	(Ignore)	
							Valid	Send ACK (data OUT operation)	○
					IN			Send null DATA1 packet (status IN operation)	○
			0	1	SETUP	—	Invalid	(Ignore)	
						—	Valid	Send ACK	○
					OUT	—	Invalid	(Ignore)	
						—	Valid	Send ACK (status OUT operation)	○
					IN			Send data (data IN operation)	○
			1	0	SETUP	—	Invalid	(Ignore)	
						—	Valid	Send ACK	○
					OUT	—	Invalid	(Ignore)	
						—	Valid	Send STALL	
					IN			Send null DATA1 packet (status IN operation)	○
			1	1	SETUP	—	Invalid	(Ignore)	
						—	Valid	Send ACK	○
					OUT	—	Invalid	(Ignore)	
						—	Valid	Send ACK (status OUT operation)	○
					IN			Send STALL	

Table 3-4 Transmission Mode Chart

— Don't care
Not applicable

*SETUP tokens are always accepted regardless of the transmission mode setting.
(1) Only SETUP tokens are accepted if ***E0CSU***=0. IN/OUT tokens are ignored.

(2) ***E0CSU***=1 indicates that the Setup stage has been completed. IN/OUT tokens are accepted. When ***E0ACK***=0, however, the LC87F1M16A responds with a NAK handshake packet to IN/OUT tokens.

(3) ***E0CSU***=1 indicates that the Setup stage has been completed. When ***E0ACK***=1, the LC87F1M16A can perform Data and Status stage processing for IN/OUT tokens.
[***E0CST***=0, ***E0CRW***=0] Identifies a control write transfer Data stage. In this case, the LC87F1M16A performs data OUT processing on OUT tokens and status IN processing on IN tokens.
[***E0CST***=0, ***E0CRW***=1] Identifies a control read transfer Data stage. In this case, the LC87F1M16A performs status OUT processing on OUT tokens and data IN processing on IN tokens.
[***E0CST***=1, ***E0CRW***=0] Identifies a control write transfer Status stage. In this case, the LC87F1M16A responds with a STALL handshake packet to OUT tokens and performs status IN processing on IN tokens.
[***E0CST***=1, ***E0CRW***=1] Identifies a control read transfer Status stage. In this case, the LC87F1M16A performs status OUT processing on OUT tokens and responds with a STALL handshake packet to IN tokens.

The Transmission mode setup processing to be performed during each preparatory processing is shown in Figure 3-5.

bit:	7	6	5	4	3	2	1	0
EP0STA	***E0EN***	***E0TGL***	***E0OVR***	***E0STL***	***E0ACK***	***E0CSU***	***E0CST***	***E0CRW***
(FE8Eh)	Set to 1 during initialization	Usually, a setup by the firmware is unnecessary.			Set up the transmission mode			

Transmission mode: X is automatically set by hardware X must be set by firmware

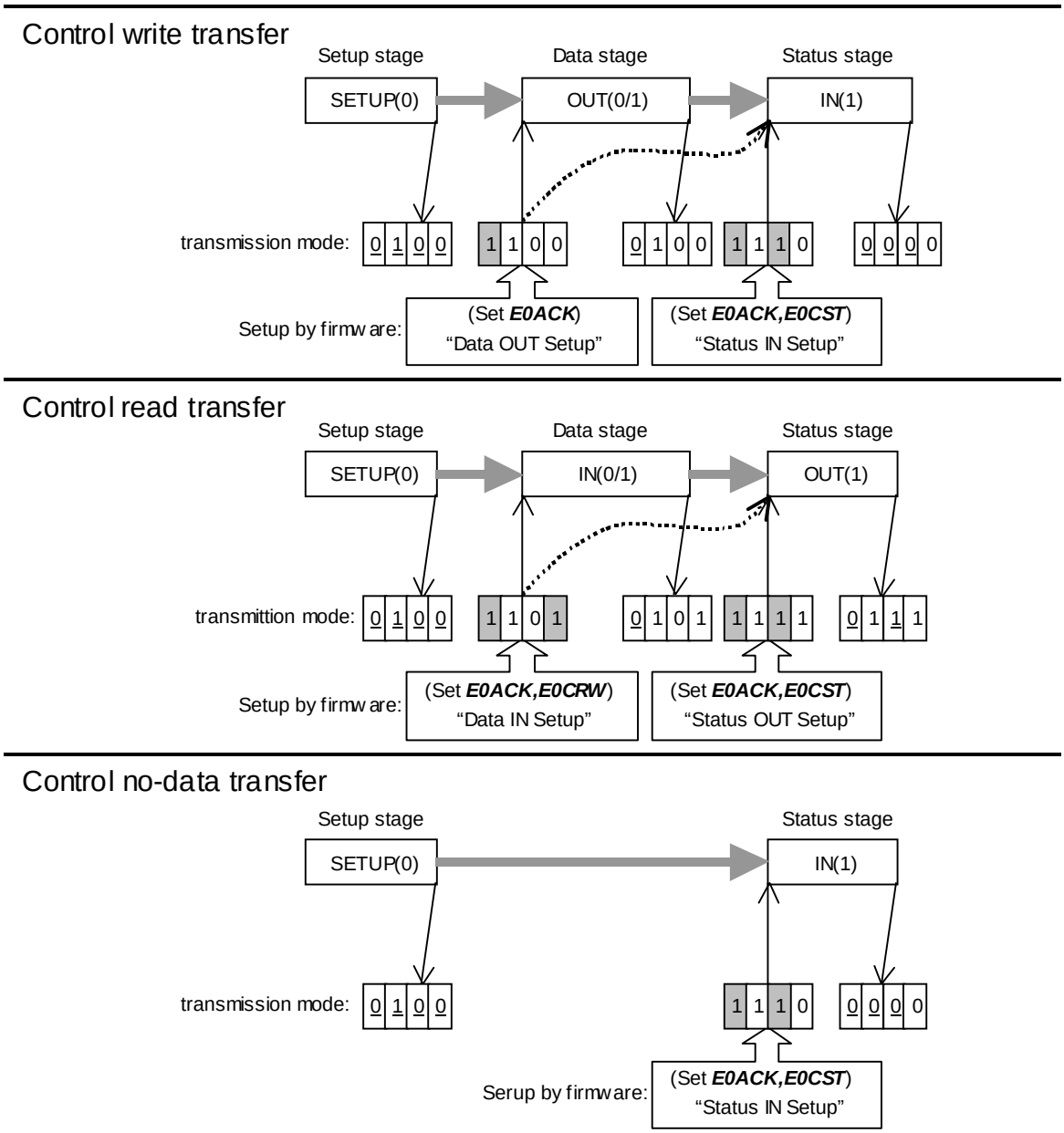


Figure 3-5 Transmission Mode Setting Flow

3.3. Setup stage

3.3.1. Outline of Setup stage SETUP transaction

The Setup stage is executed when a SETUP token is received (see Figure 3-6). It is a transaction that begins with a SETUP token packet. The actions that the device is to take vary depending on the status of the received data (presence or absence of errors).
(Refer to 8.4.5.4, "Function Response to SETUP Transaction," of the USB 2.0 Specification.)

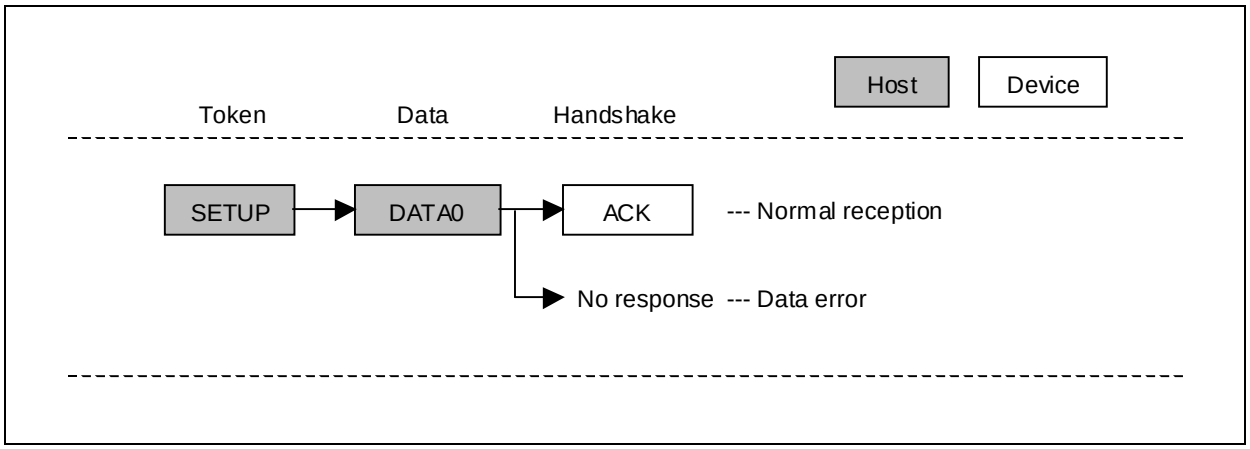


Figure 3-6 SETUP Transaction

3.3.2. SETUP Operation

The device must always be ready to receive the SETUP transaction regardless of its state. The LC87F1M16A can always accept the SETUP token the endpoint 0 is enabled (***E0EN***=1). When the LC87F1M16A receives a data packet following a SETUP token, it stores the received data in the receiving data area (200h-) of the endpoint 0. When the LC87F1M16A sends an ACK after normal reception of data, the transmission mode of the endpoint 0 is updated and the data toggle bit (***E0TGL***) is set to 1. A SETUP operation flow of the Setup stage is shown in Figure 3-7.

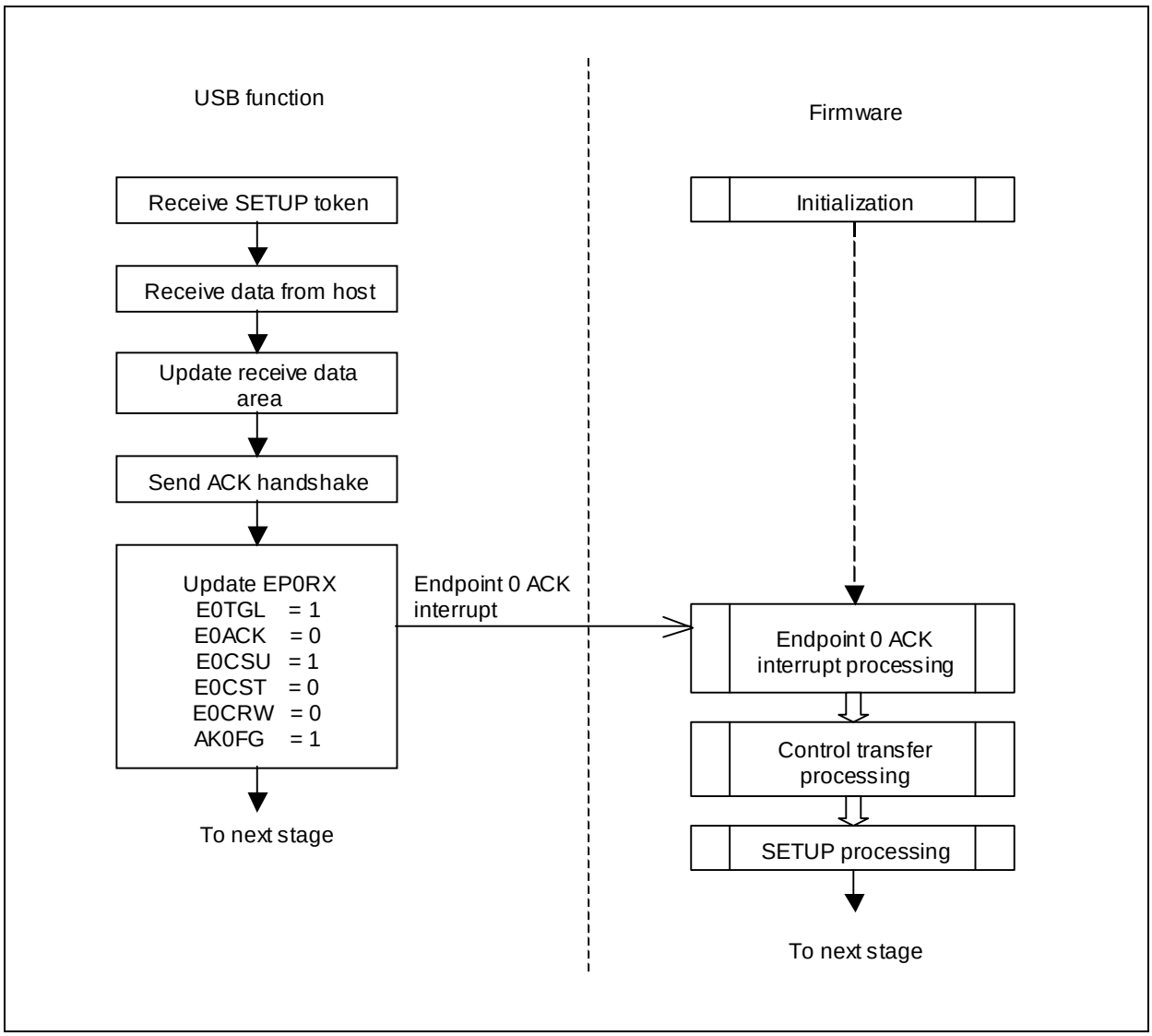


Figure 3-7 SETUP Operation Flow

3.3.3. SETUP Processing

When the SETUP transaction terminates normally, the endpoint 0 ACK interrupt flag (***AK0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1 and an endpoint 0 ACK interrupt is generated.
The LC87F1M16A takes the following actions when an endpoint 0 ACK interrupt occurs:

- Update the receive data area (200h-).
- Update the receive data size (**E0RX**).
- Set up the transmission mode (***E0ACK***, ***E0CSU***, ***E0CST***, ***E0CRW***).
- Set the toggle bit to 1 (***E0TGL***=1).

The endpoint 0 ACK interrupt processing routine must perform the following SETUP processing:

- Read the receive data.
- Analyze the request.
- Process the request.
- Take preparatory actions for the next stage. (Data IN Setup / Data OUT Setup / Status IN Setup)

The SETUP processing to be performed during the control transfers processing looks like as shown below (see 3.2.4, “Control Transfers Processing").

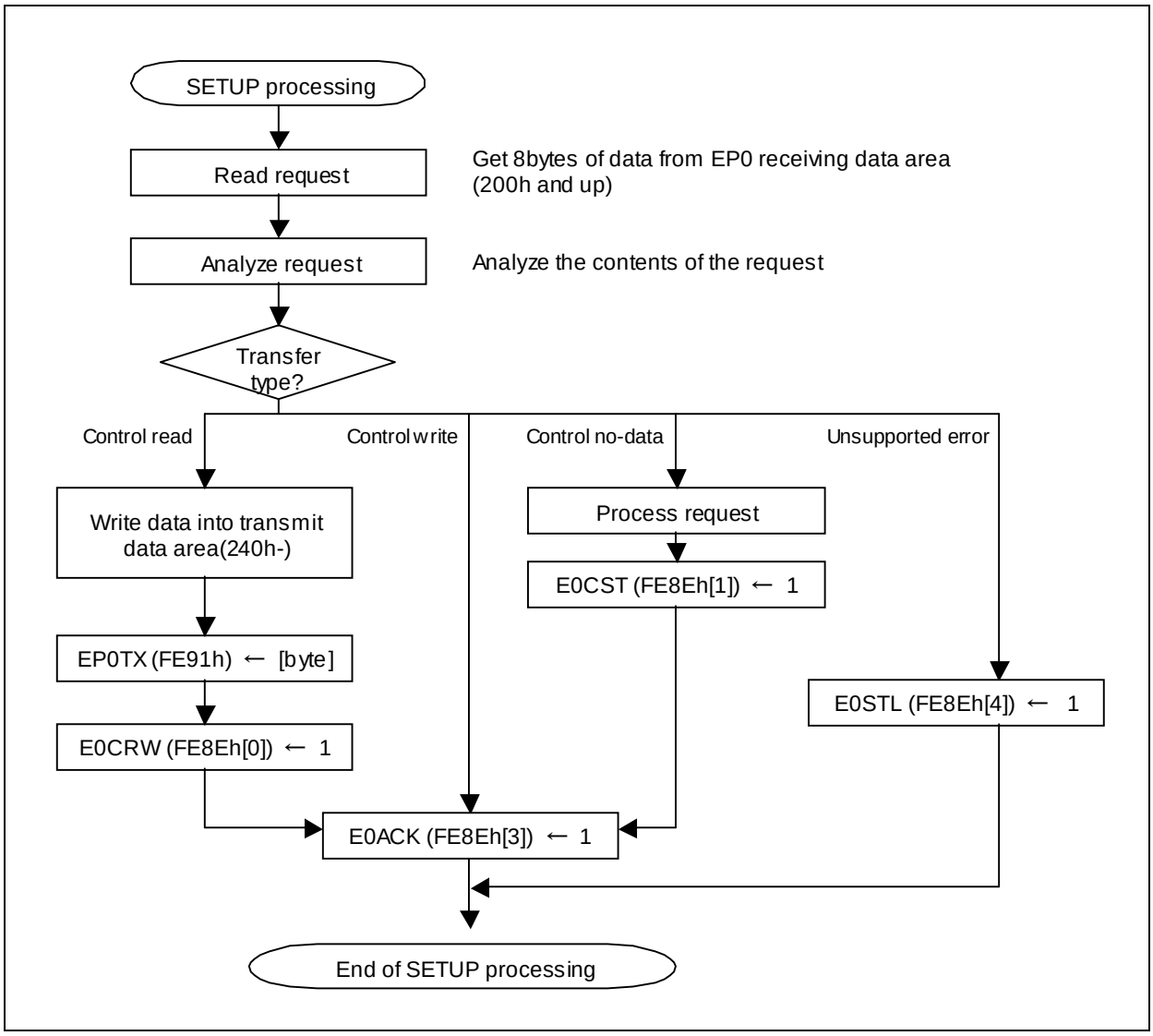


Figure 3-8 Example of SETUP Processing

3.3.4. Receive Errors

- Data error (corruption)

  The LC87F1M16A makes error checks (CRC, bit stuffing) by hardware. When the LC87F1M16A detects an error, it returns no response regardless of the transmission mode settings. In such a case, the endpoint 0 error interrupt flag (***ER0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1. Error recovery actions, then, must be taken as required. Normally, the routine can continue SETUP receive processing following the data retransmission from the host.

- Unsupported request

  The endpoint 0 ACK interrupt processing routine must send a STALL handshake packet in the next transaction even when it received data with no error and sent an ACK handshake packet if the received request is not supported. The STALL bit (***E0STL***) of the endpoint 0 status register **EP0STA** (FE8Eh) must be set to 1 by firmware.

3.4. Control Write Transfer Data Stage

3.4.1. Outline of Data stage OUT transaction

Data is transferred from the host to the device (OUT direction) in the Data stage of the control write transfer. Accordingly, the device must receive data in this stage. The outline of OUT transaction in the Data stage (OUT) is shown in Figure 3-9. The operation that the device is to perform when it receives a data packet from the host following an OUT token varies depending on the status of the received data and the state of the device.
(Refer to 8.4.5.3, "Function Response to an OUT Transaction," of USB 2.0 Specification.)

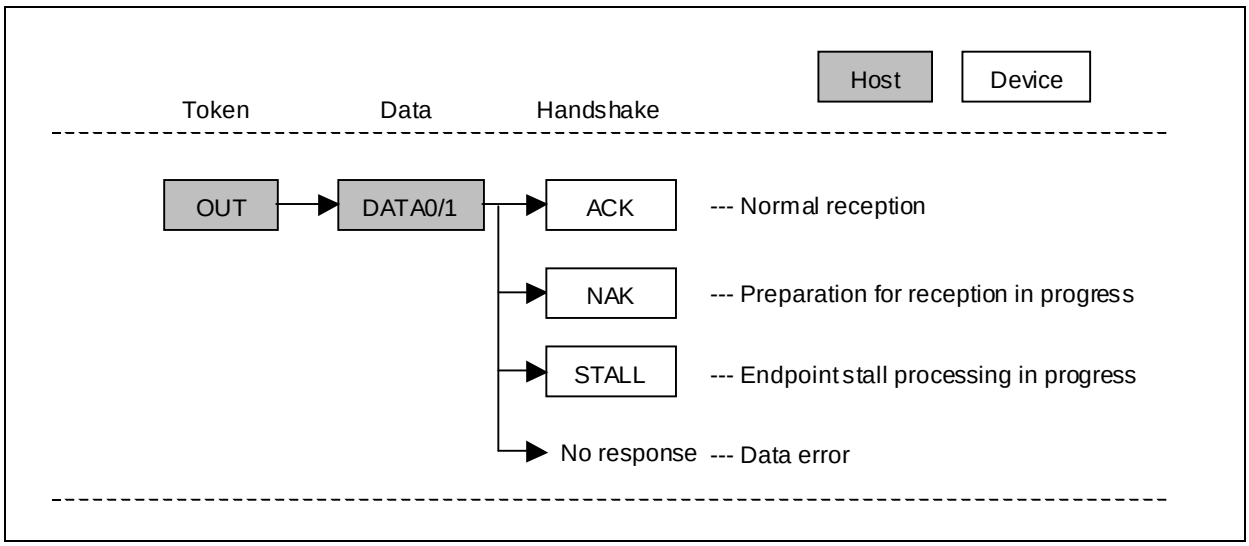


Figure 3-9 Data OUT Transaction

3.4.2. Data OUT Setup

The preparatory steps shown below are required before receiving OUT token to receive data in the OUT transactions in the Data stage.

- Setting up the transmission mode
  Set the lower-order 4 bits (***E0ACK***, ***E0CSU***, ***E0CST***, ***E0CRW***) of **EP0STA** (FE8Eh).

  [0]***E0CRW***: Must not be changed (set to 0 on completion of the Setup stage).
  [1]***E0CST***: Must not be changed (set to 0 on completion of the Setup stage).
  [2]***E0CSU***: Must not be changed (set to 1 on completion of the Setup stage).
  [3]***E0ACK***: Must be set to 1.

The control write transfer data stage is started by performing the preparation processing above after the setup stage is completed (see 3.3.3, “SETUP Processing”). If the data is received in multiple OUT transactions, the above processing is performed after the normal termination of each OUT transaction (see Section 3.4.4).

3.4.3. Data OUT Operation

The operation of the control write transfer Data stage proceeds according to the transmission mode settings stored in **EP0STA**(FE8Eh) when the endpoint 0 is enabled (***E0EN***=1). The setup procedure is explained in 3.4.2. "Data OUT setup." If data OUT setup is performed properly, the LC87F1M16A receives a data packet following an OUT token, and then returns an ACK. If the data toggle bits match, the received data is stored in the endpoint 0 receive data area (200h-), the receive size (**EP0RX**) is updated, the data toggle bit (***E0TGL***) is inverted, and the ACK bit is cleared (***E0ACK***=0). The endpoint 0 ACK interrupt occurs on completion of the OUT transaction and the endpoint 0 ACK interrupt processing routine must perform the Data OUT processing. The Data OUT operation flow of the Data stage is shown in Figure 3-10.

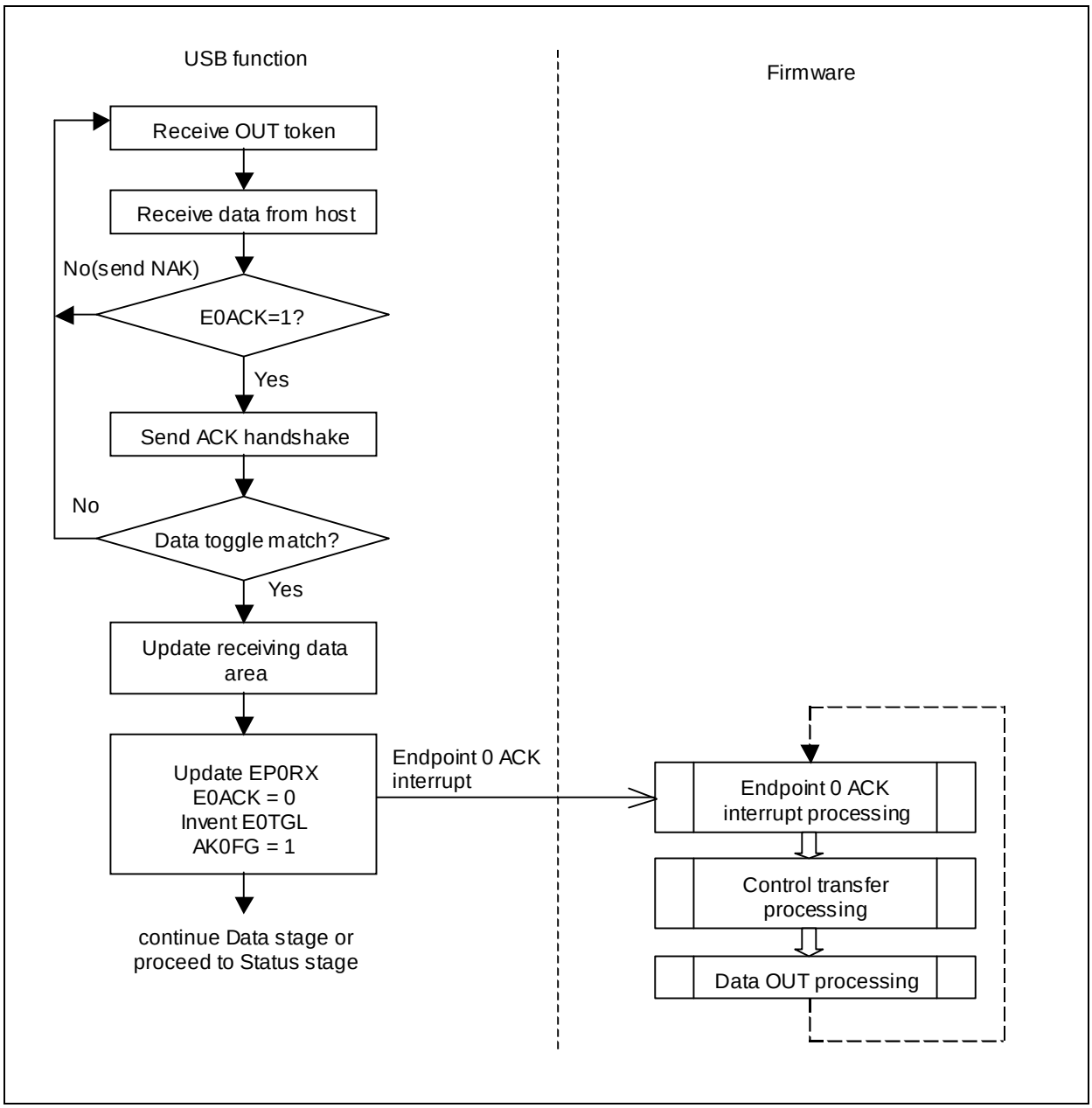


Figure 3-10 Data OUT Operation Flow

3.4.4. Data OUT Processing

When the Data stage OUT transaction terminates normally, the endpoint 0 ACK interrupt flag (***AK0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1 and an endpoint 0 ACK interrupt is generated.
The following actions are automatically taken when an endpoint 0 ACK interrupt occurs:

- Update the receive data area (200h-).
- Update the receive data size (E0RX).
- Turn on the NAK mode (E0ACK=0).
- Invert the toggle bit (E0TGL).

The endpoint 0 ACK processing routine must perform the following Data OUT processing:

- Read the receive data.
- Process the request and take preparatory actions for the next stage.

The Data OUT processing to be performed during the control transfers processing looks like as shown below (see 3.2.4, "Control Transfers processing").

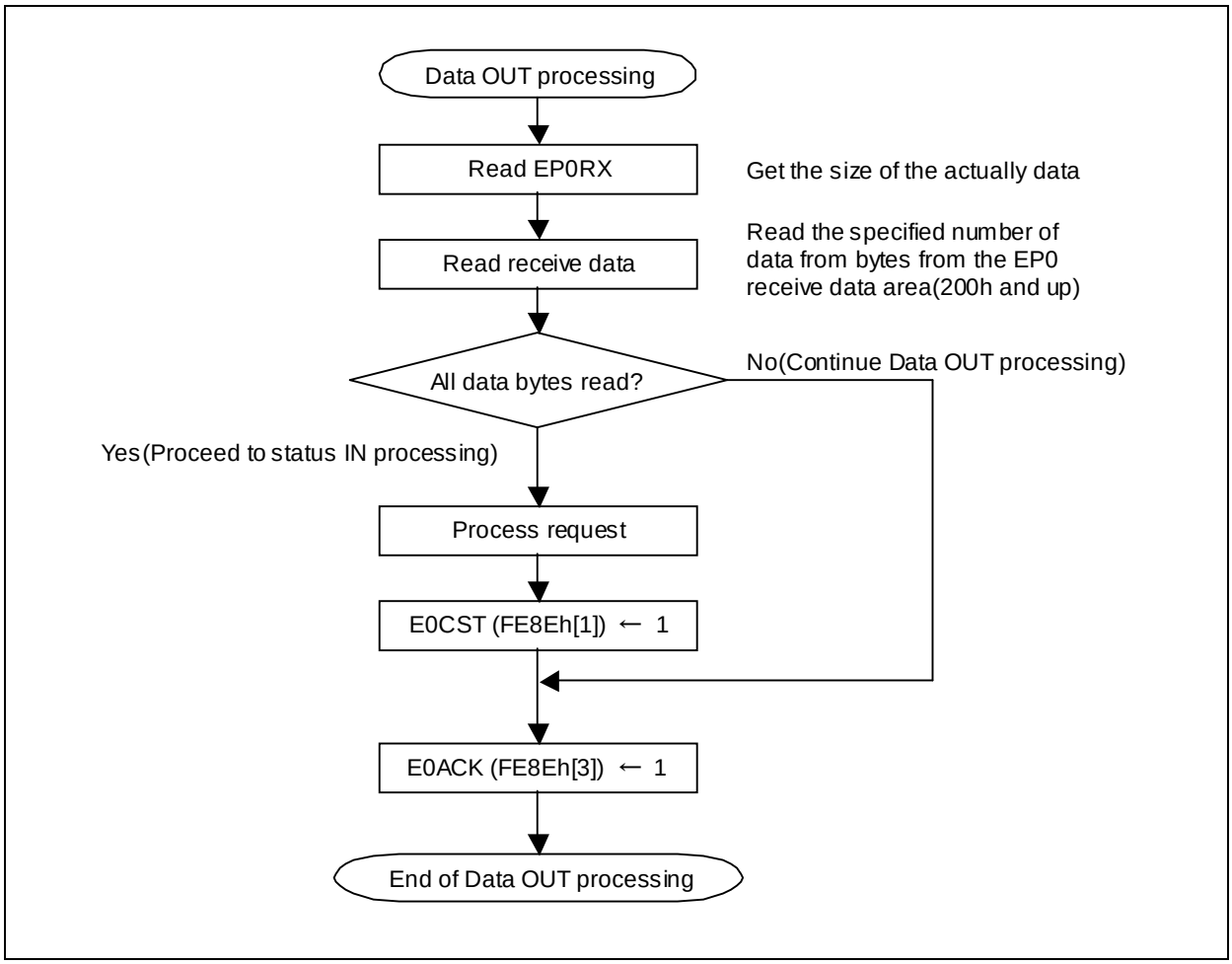


Figure 3-11 Example of Data OUT Processing

3.4.5. Receive Errors

- Toggle mismatch

  The LC87F1M16A performs hardware toggle check when it receives data. If a toggle mismatch is found, the LC87F1M16A responds with an ACK handshake packet but generates no endpoint 0 ACK interrupt. It neither updates the receive data area nor change the toggle bit (***E0TGL***) or transmission mode settings. Consequently, the routine can continue the data receive processing.

- Expected receive size exceeded (stall)

  The LC87F1M16A must respond with a STALL in the next transaction if it receives data without an error but the byte count of the received data is greater than the byte count specified in the Setup stage. In such a case, the STALL bit (***E0STL***) of the endpoint 0 status register **EP0STA** (FE8Eh) must be set to 1 by firmware. (Refer to 8.5.3.1, "Reporting Status Results," of the USB 2.0 Specification.)

- Data error (corruption)

  The LC87F1M16A makes error checks (CRC, bit stuff) by hardware. When the LC87F1M16A detects an error, it returns no response regardless of the transmission mode settings. In such a case, the endpoint 0 error interrupt flag (***ER0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1. Normally, the routine can continue data receive processing following the data retransmission from the host. Take any error correction procedure if necessary.

- Maximum payload exceeded

  The payload excess bit (***E0OVR***) of the endpoint 0 status register **EP0STA** (FE8Eh) is set to 1 even when the LC87F1M16A receives data without an error if the byte count of the received data exceeds the maximum payload size specified in **EP0MP** (FE8Fh). Normally, the routine can continue data receive processing following the data retransmission from the host. Take any error correction procedure if necessary.

3.5. Control Write and No Data Transfer Status Stage

3.5.1. Outline of Status stage IN transaction

The Status stages of the control write transfer and control no data transfer involve those transfers in which the results of the Setup and Data stages (success/failure/processing) are sent to the host (IN direction). The outline of the IN transaction in the Status stage (IN) is shown in Figure 3-12. The operation that the device is to perform when it receives an IN token from the host varies depending on the status of the received token and the state of the device.
(Refer to 8.5.3.1, "Reporting Status Results," of USB 2.0 Specification.)

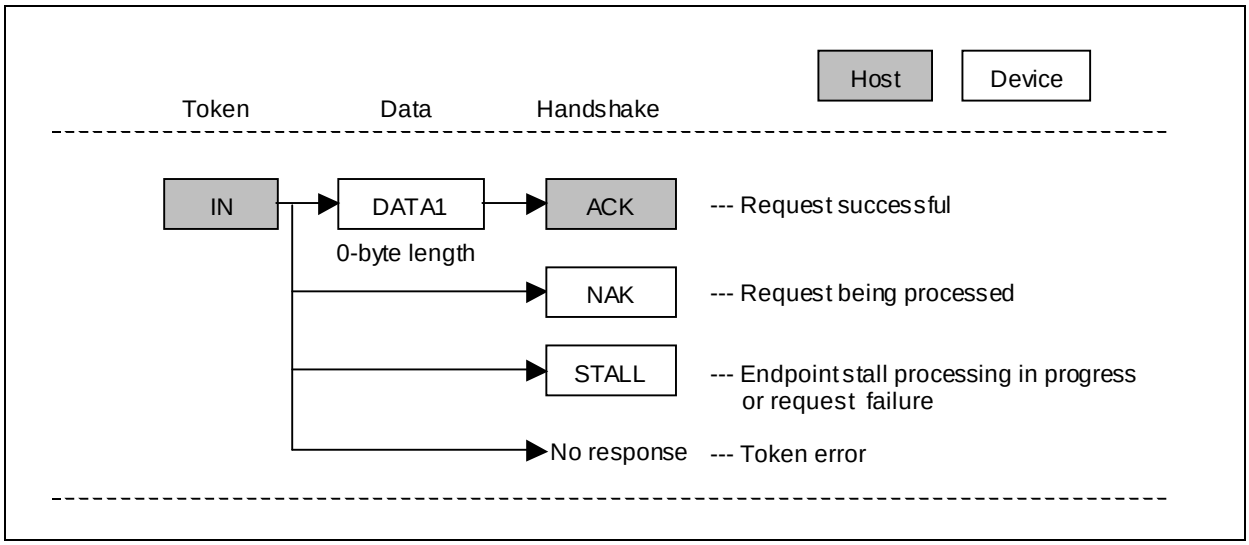


Figure 3-12 Status IN Transaction

3.5.2. Status IN Setup

The preparatory steps shown below are required before receiving IN token to send data in the IN transaction of the Status stage

- Setting up the transmission mode

  Set the lower-order 4 bits (***E0ACK***, ***E0CSU***, ***E0CST***, ***E0CRW***) of **EP0STA** (FE8Eh).

  [0]***E0CRW***: Must not be changed (set to 0 on completion of the Setup stage).
  [1]***E0CST***: Must be set to 1.
  [2]***E0CSU***: Must not be changed (set to 1 on completion of the Setup stage).
  [3]***E0ACK***: Must be set to 1.

For a control no-data transfer, the preparation processing above is performed after the setup stage is completed (see 3.3.3, “SETUP Processing”). For a control write transfer, the above processing is performed after the data stage is completed (see 3.4.4). When an IN token is received in the data stage of control write transfer, even if this preparation processing is omitted, the status stage operation is performed normally.

3.5.3. Status IN Operation

The operation of the control write no data transfer Status stage proceeds according to the transmission mode settings specified in **EP0STA** (FE8Eh) when the endpoint 0 is enabled (***E0EN***=1). The setup procedure is explained in 3.5.2, "Status IN setup." If status IN setup is performed properly, the LC87F1M16A sends a 0-byte DATA1 data packet in response to the IN token. When an ACK handshake packet is received from the host, the data toggle bit is cleared (***E0TGL***=0) as well as the ACK, SETUP, and STATUS bits are cleared (***E0ACK***=***E0CSU***=***E0CST***=0). The endpoint 0 ACK interrupt occurs on completion of the IN transaction and the endpoint 0 ACK interrupt processing routine must perform the Status IN processing. Status IN operation flow of the Status stage is shown in Figure 3-13.

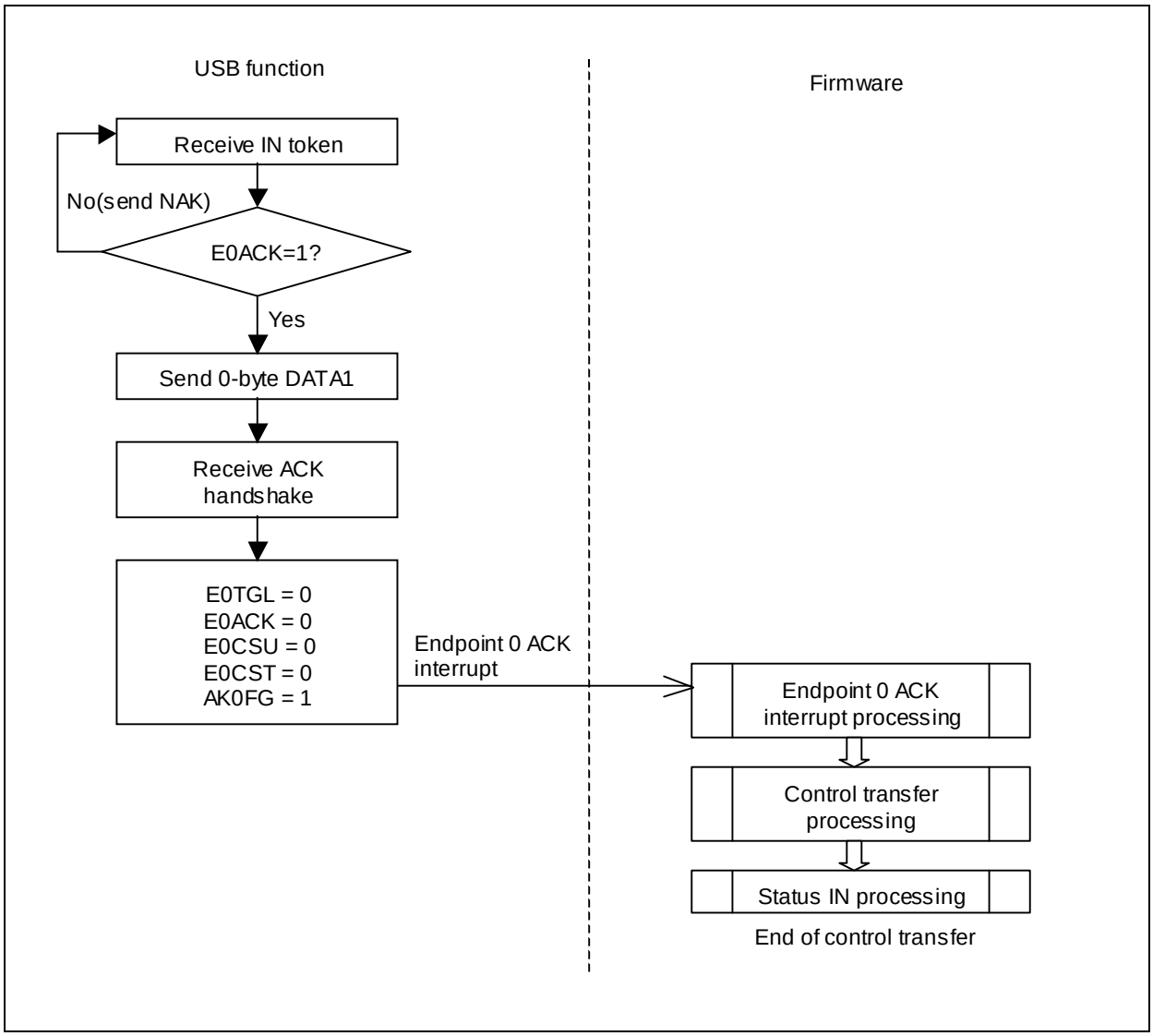


Figure 3-13 Status IN Operation Flow

3.5.4. Status IN Processing

When the IN transaction of the Status stage terminates normally, the endpoint 0 ACK interrupt flag (***AK0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1 and an endpoint 0 ACK interrupt is generated.
The following actions are automatically taken when an endpoint 0 ACK interrupt occurs:

- Turn on the NAK mode (***E0ACK***=0).
- Clear the SETUP and STATUS bits (***E0CSU***=***E0CST***=0).
- Clear the toggle bit (***E0TGL***=0).

The endpoint 0 ACK interrupt processing routine must take the actions corresponding to the standard device request SetAddress. There is no need to set up the transmission mode as the LC87F1M16A is set up to accept only SETUP tokens.

The Status IN processing to be performed during the control transfers processing looks like as shown below (see 3.2.4, “Control Transfers Processing”).

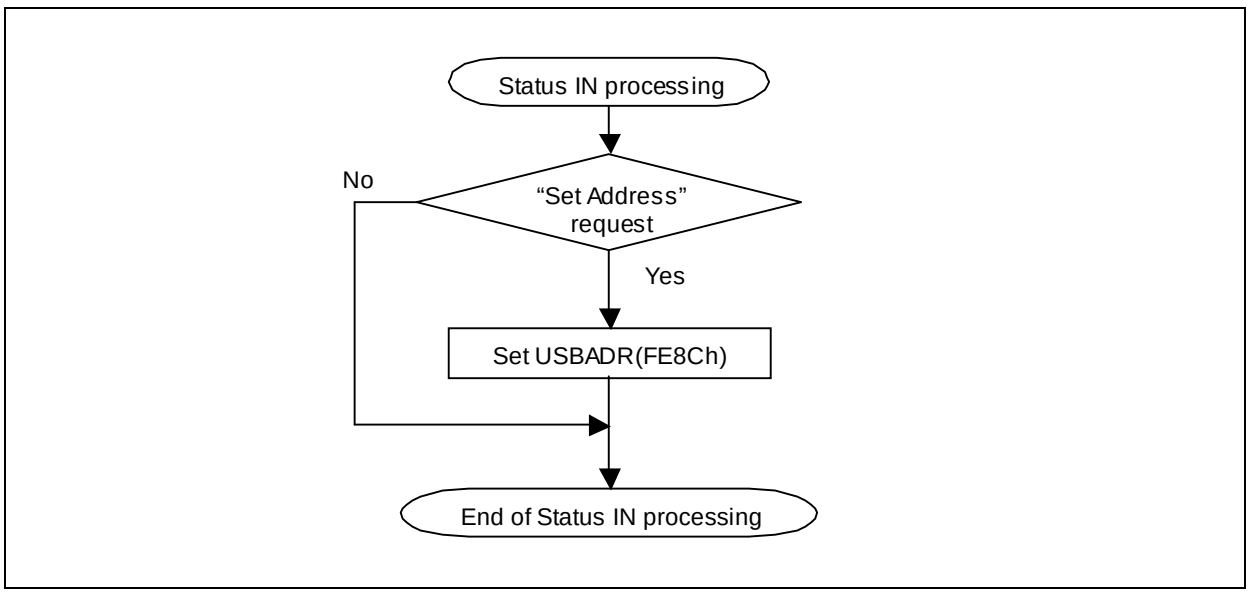


Figure 3-14 Example of Status IN Processing

3.5.5. Transmission Errors

- Stall

  The STALL bit (***E0STL***) of the endpoint 0 status register **EP0STA** (FE8Eh) must be set to 1 by firmware if the endpoint 0 is stopped or it can transmit no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN/OUT tokens received from the host.

  If an OUT token is received when the data reception in the Data stage is completed and ***E0CST*** is set to 1 (status IN), the LC87F1M16A sends a STALL handshake packet and sets the STALL bit (***E0STL***) to 1. In this case, the STALL interrupt flag (***ST0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1. Error recovery processing must be performed as required.

- Data retransmission

  If the device fails to receive an ACK handshake packet from the host after sending a 0-byte DATA1 data packet normally, it is necessary to retransmit the data packet. Since the LC87F1M16A performs retransmission automatically by hardware, there is no need for the firmware to control the retransmission processing.

3.6. Control Read Transfer Data Stage

3.6.1. Outline of Data stage IN transaction

Since data transfers from device to host (IN direction) occur in the Data stage of the control read transfer, the device must perform data transmissions. The outline of the IN transaction in the Data stage is shown in Figure 3-15. The operation that the device is to perform when it receives an IN token from the host varies depending on the status of the received token and the state of the device.
(Refer to 8.4.6.1, "Function Response to IN Transactions," of the USB 2.0 Specification.)

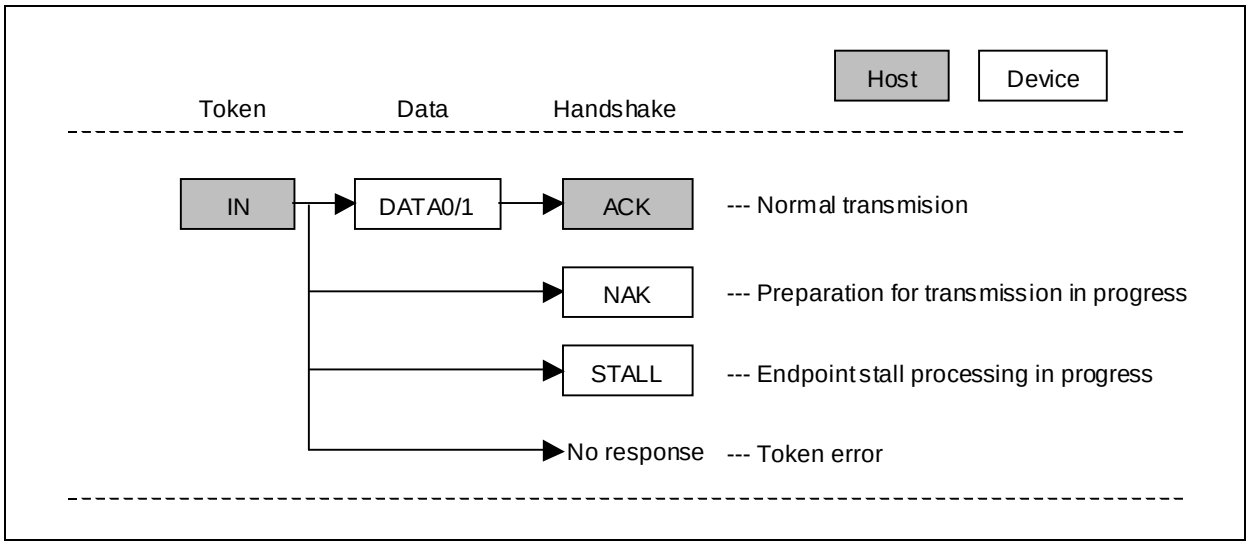


Figure 3-15 Data IN Transaction

3.6.2. Data IN Setup

The preparatory steps shown below are required before receiving IN token to send data in the IN transactions in the Data stage.

- Writing data
  Write the data into the endpoint 0 transmit data area (240h-).

- Setting the transmit data size
  Store the number of bytes to be transmitted in **EP0TX** (FE91h).

- Setting up the transmission mode
  Set the lower-order 4 bits (***E0ACK***, ***E0CSU***, ***E0CST***, ***E0CRW***) of **EP0STA** (FE8Eh).

  [0]***E0CRW***: Must not be changed (must have been set to 1 during SETUP processing).
  [1]***E0CST***: Must not be changed (set to 0 on completion of the Setup stage).
  [2]***E0CSU***: Must not be changed (set to 1 on completion of the Setup stage).
  [3]***E0ACK***: Must be set to 1.

The control read transfer data stage is started by performing the preparation processing above after the setup stage (see 3.3.3, “SETUP Processing”). If the data is sent in multiple IN transactions, the above processing is performed after the normal termination of each IN transaction (see Section 3.6.4).

3.6.3. Data IN Operation

The operation of the control read transfer Data stage proceeds according to the transmission mode settings specified in **EP0STA** (FE8Eh) when the endpoint 0 is enabled (***E0EN***=1). The setup procedure is explained in 3.6.2, "Data IN setup." If data IN setup is performed properly, the data stored in the endpoint 0 transmit data area (240h-) is transmitted over a data packet following the IN token. In this case, the PID corresponding to the data toggle bit (***E0TGL***) is used. The size of the data is equal to the byte count defined in **EP0TX**. When an ACK is received following the transmission processing, the data toggle bit (***E0TGL***) is inverted and the ACK bit is cleared (***E0ACK***=0). The endpoint 0 ACK interrupt occurs on completion of the IN transaction and the endpoint 0 ACK interrupt processing routine must perform the Data IN processing. The Data IN operation flow of the Status stage is shown in Figure 3-16.

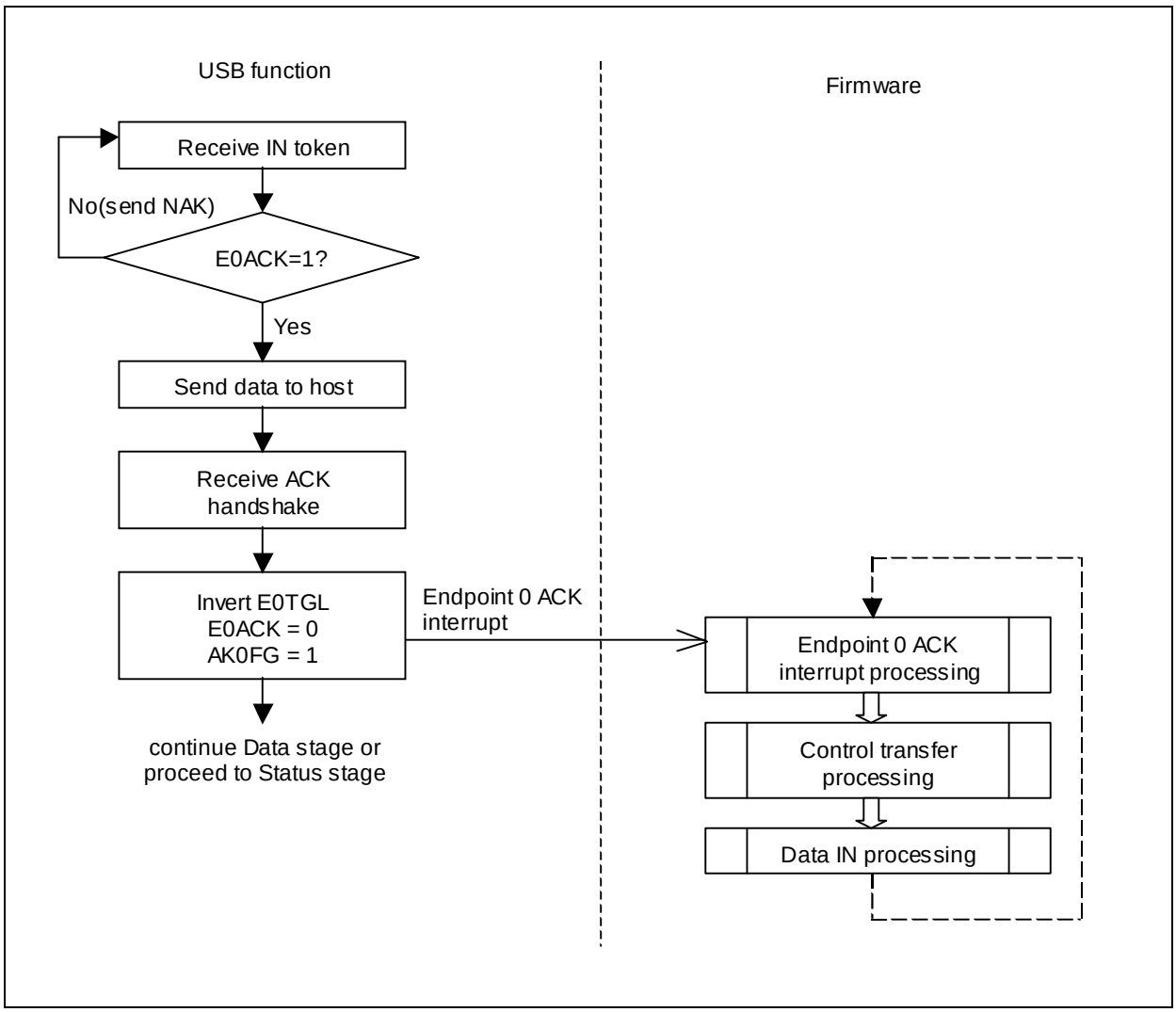


Figure 3-16 Data IN Operation Flow

3.6.4. Data IN Processing

When the IN transaction in the Data stage terminates normally, the endpoint 0 ACK interrupt flag (***AK0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1 and an endpoint 0 ACK interrupt is generated.
The following actions are automatically taken when an endpoint 0 ACK interrupt occurs:

- Turn on the NAK mode (***E0ACK***=0).
- Invert the toggle bit (***E0TGL***).

The endpoint 0 ACK interrupt processing routine must perform the following Data IN processing:

- Take preparatory actions for the next stage.

The Data IN processing to be performed during the control transfers processing looks like as shown below (see 3.2.4, "Control Transfers Processing").

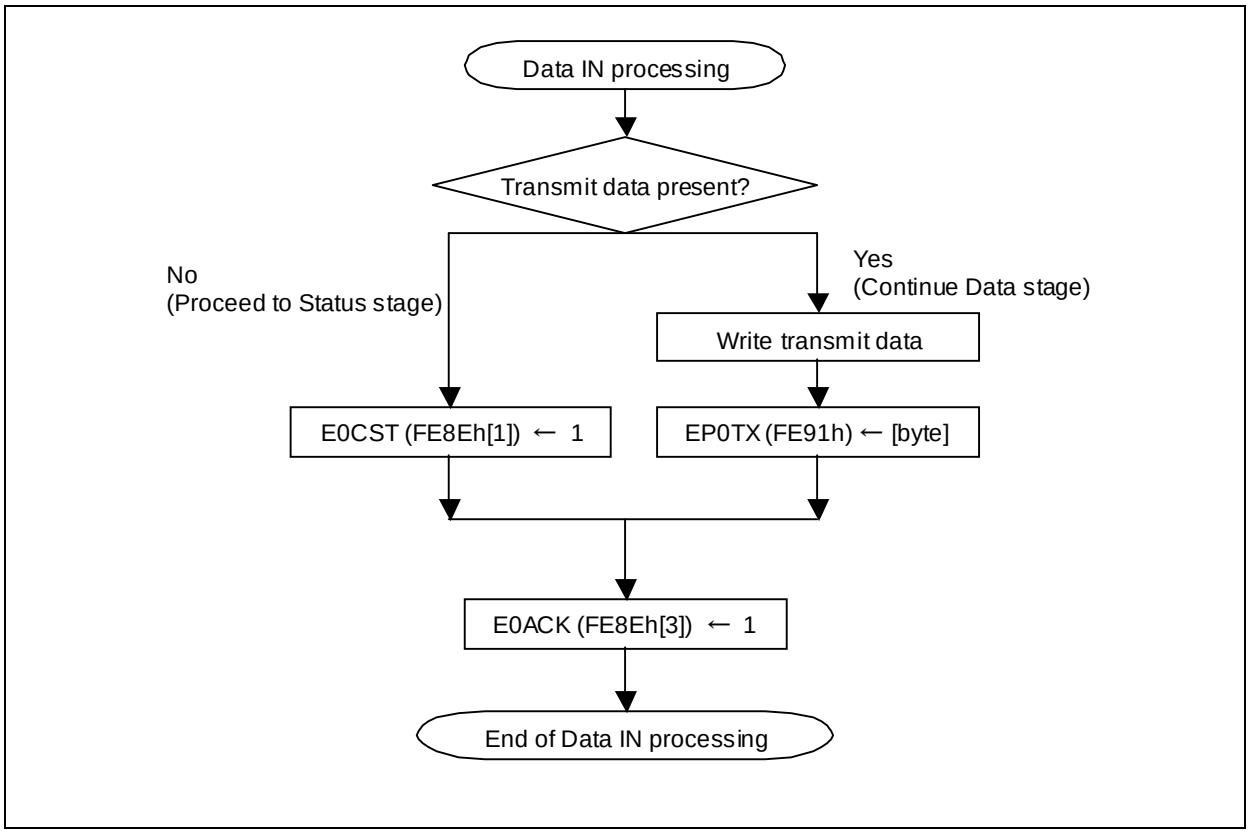


Figure 3-17 Example of Data IN Processing

3.6.5. Transmission Errors

- Stall

  The STALL bit (***E0STL***) of the endpoint 0 status register **EP0STA** (FE8Eh) must be set to 1 by firmware if the endpoint 0 is stopped or it can transmit no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN,OUT tokens received from the host.

- Data retransmission

  If the device fails to receive an ACK handshake packet from the host after sending a data packet normally, it is necessary to retransmit the data packet. Since the LC87F1M16A performs retransmission automatically by hardware, there is no need for the firmware to control the retransmission processing.

3.7. Control Read Transfer Status Stage

3.7.1. Outline of Status stage OUT transaction

The Status stage of the control read transfer involves those transfers in which the results of the setup and Data stages (success/failure/processing) `re sent to the host (OUT direction). The outline of the OUT transaction in the Status stage (OUT) is shown in Figure 3-18. The operation that the device is to perform when it receives a 0-byte DATA1 data packet from the host following an OUT token varies depending on the status of the received data and the state of the device. (Refer to 8.5.3.1, "Reporting Status Results," of the USB 2.0 Specification.)

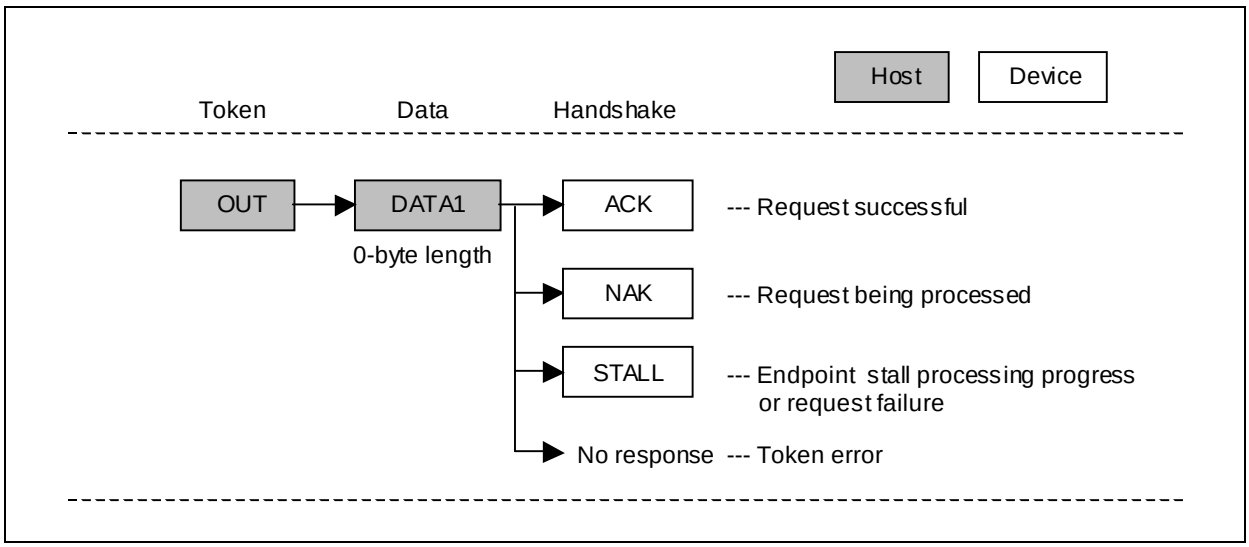


Figure 3-18 Status OUT Transaction

3.7.2. Status OUT Setup

The preparatory steps shown below are required before receiving OUT token to receive data in the OUT transactions in the Status stage.

- Setting up the transmission mode
  Set the lower-order 4 bits (***E0ACK***, ***E0CSU***, ***E0CST***, ***E0CRW***) of **EP0STA** (FE8Eh).

  [0]***E0CRW***: Must not be changed (must have been set to 1 during SETUP processing).
  [1]***E0CST***: Must be set to 1.
  [2]***E0CSU***: Must not be changed (set to 1 on completion of the Setup stage).
  [3]***E0ACK***: Must be set to 1.

The preparation processing above is performed after the data stage. When OUT token is received during the data stage of control read transfer, even if this preparation processing is omitted, the status stage operation is performed normally.

3.7.3. Status OUT Operation

The operation of the control read transfer Status stage proceeds according to the transmission mode settings specified in **EP0STA** (FE8Eh) when the endpoint 0 is enabled (***E0EN***=1). The setup procedure is explained in 3.7.2, "Status OUT setup." If status OUT setup is performed properly, the LC87F1M16A receives a data packet following an OUT token and returns an ACK. In this case, the LC87F1M16A makes no check on the PID or data size of the data packet. It also sends an ACK when the DATA0 packet or size is not 0 byte. When the LC87F1M16A sends an ACK, the data toggle bit is cleared (***E0TGL***=0), the ACK bit is cleared (***E0ACK***=0), and the STATUS bit is set (***E0CST***=1). The endpoint 0 ACK interrupt occurs on completion of the OUT transaction and the endpoint 0 ACK interrupt processing routine must perform the Status OUT processing. Status OUT operation flow of the Status stage is shown in Figure 3-19.

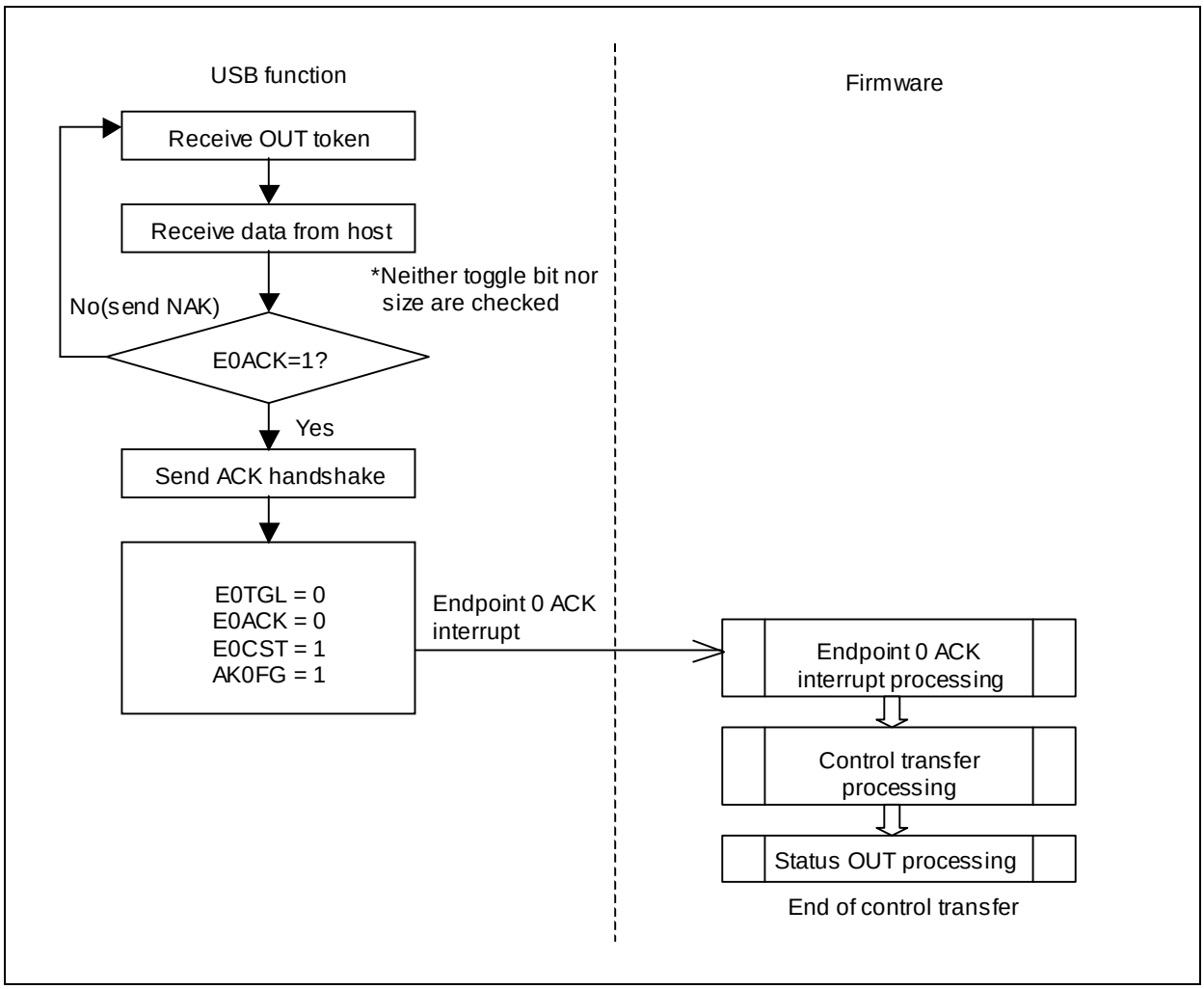


Figure 3-19 Status OUT Operation Flow

3.7.4. Status OUT Processing

When the OUT transaction of the Status stage terminates normally, the endpoint 0 ACK interrupt flag (***AK0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1 and an endpoint 0 ACK interrupt is generated.
The following actions are automatically taken when an endpoint 0 ACK interrupt occurs:

- Turn on the NAK mode (***E0ACK***=0).
- Set the STATUS bit (***E0CST***=1).
- Clear the toggle bit (***E0TGL***=0).

The transmission mode in this case is set up so as to respond with a NAK for IN/OUT tokens. Since it is considered that a control transfer is normally followed by a SETUP token, it is possible to start a Setup stage normally even when the NAK mode is on.
To inhibit the LC87F1M16A from responding IN/OUT tokens after completing a control transfer, the following status OUT processing must be performed during the endpoint 0 ACK interrupt processing:

- Clear the SETUP bit (***E0CSU***=0).

(to inhibit the LC87F1M16A from accepting tokens other than SETUP.)

The Status OUT processing to be performed during the control transfers processing looks like as shown below (see 3.2.4, "Control Transfers Processing").

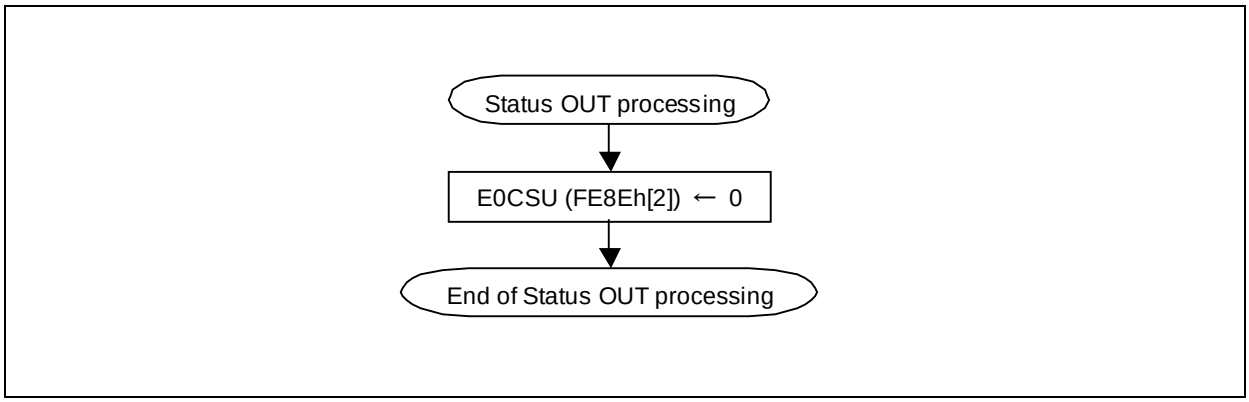


Figure 3-20 Example of Status OUT Processing

3.7.5. Receive Errors

- Stall

  The LC87F1M16A sends a STALL handshake packet and set the STALL bit (***E0STL***) to 1 if it receives an IN token even when the data transmission in the Data stage is completed and the transmission mode (**EP0STA**[0-3]) is set to [1111] (status OUT). In this case, the STALL interrupt flag (***ST0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1. Error recovery actions must be taken as required.

  The STALL bit (***E0STL***) of the endpoint 0 status register **EP0STA** (FE8Eh) must be set to 1 by firmware if the endpoint 0 is stopped or it can transmit no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN/OUT tokens received from the host.

- Data error (corruption)

  The LC87F1M16A makes error checks (CRC, bit stuffing) by hardware. When the LC87F1M16A detects an error, it returns no response regardless of the transmission mode settings. In such a case, the endpoint 0 error interrupt flag (***ER0FG***) of the endpoint 0 interrupt register **EP0INT** (FE83h) is set to 1. Error recovery actions, then, must be taken as required. Normally, the routine can continue data receive processing following the data retransmission from the host.

Chapter 4. Data Transfers

4.1. Endpoint Control

When the device switches into the Configured state as the result of a bus enumeration, the endpoints described in the host-designated configuration become available, enabling data transfers via the endpoints. The LC87F1M16A supports bulk, interrupt, and isochronous transfers.

Endpoints n (n=1-6) are used for data transfers. Since the transfer buffers are mapped into the internal RAM areas, data transfers are carried out by means of accesses to these areas (transmit data writes or receive data reads) according to the settings of the related registers.

4.1.1. Related registers

The registers that are related to data transfers (bulk/interrupt/isochronous) are listed below.

Symbol	Address	R/W	Function
EP1INT	FE84h	R/W	EP1 interrupt
EP2INT	FE85h	R/W	EP2 interrupt
EP3INT	FE86h	R/W	EP3 interrupt
EP4INT	FE87h	R/W	EP4 interrupt
EP5INT	FE88h	R/W	EP5 interrupt
EP6INT	FE89h	R/W	EP6 interrupt
EP1STA	FE92h	R/W	EP1 control
EP2STA	FE93h	R/W	EP2 control
EP3STA	FE94h	R/W	EP3 control
EP4STA	FE95h	R/W	EP4 control
EP5STA	FE96h	R/W	EP5 control
EP6STA	FE97h	R/W	EP6 control
EP1CNT	FE98h	R/W	EP1 size
EP1RX	FE99h	R	EP1 receive data size
EP2CNT	FE9Ah	R/W	EP2 size
EP2RX	FE9Bh	R	EP2 receive data size
EP3CNT	FE9Ch	R/W	EP3 size
EP3RX	FE9Dh	R	EP3 receive data size
EP4CNT	FE9Eh	R/W	EP4 size
EP4RX	FE9Fh	R	EP4 receive data size
EP5CNT	FEA0h	R/W	EP5 size
EP5RX	FEA2h	R	EP5 receive data size
EP6CNT	FEA6h	R/W	EP6 size
EP6RX	FEA8h	R	EP6 receive data size

Table 4-1 Data Transfer Related Registers

●Endpoint n interrupt register : **EPnINT (n=1-6)**(FE84h-FE89h)

Bit	Bit Name	R/W	Function
7	***AKnFG***	R/W	Endpoint n ACK interrupt flag Set when the transaction terminates with an ACK. This flag must be cleared with an instruction.
6	***AKnEN***	R/W	Endpoint n ACK interrupt enable flag 1: Enable endpoint n ACK interrupts. 0: Disables endpoint n ACK interrupts.
5	***NKnFG***	R/W	Endpoint n NAK interrupt flag Set when the transaction terminates with a NAK. This flag must be cleared with an instruction.
4	***NKnEN***	R/W	Endpoint n NAK interrupt enable flag 1: Enable endpoint n NAK interrupts. 0: Disables endpoint n NAK interrupts.
3	***ERnFG***	R/W	Endpoint n error interrupt flag Set when an error occurs in the transaction. This flag must be cleared with an instruction.
2	***ERnEN***	R/W	Endpoint n error interrupt enable flag 1: Enables endpoint n error interrupts. 0: Disables endpoint n error interrupts.
1	***STnFG***	R/W	Endpoint n STALL interrupt flag Set when the transaction terminates with a STALL. This flag must be cleared with an instruction.
0	***STnEN***	R/W	Endpoint n STALL interrupt enable flag 1: Enables endpoint n STALL interrupts. 0: Disables endpoint n STALL interrupts.

●Endpoint n status register: **EPnSTA (n=1-6)** (FE92h-FE97h)

Bit	Bit Name	R/W	Function
7	***EnEN***	R/W	Endpoint n enable bit 1: Enables the endpoint n. 0: Disables the endpoint n.
6	***EnTGL***	R/W	Endpoint n data toggle bit This bit is inverted and the receive data is transferred to the buffer when this bit matches DATA PID (in the data receive mode). The data whose DATA PID matches this bit is transmitted (in the data transmit mode) and this bit is inverted when an ACK is received. No toggle inversion occurs in isochronous transfers.
5	***EnOVR***	R/W	Endpoint n maximum payload size excess bit Set when the volume of data exceeding the maximum payload size is received. This bit must be cleared with an instruction.
4	***EnSTL***	R/W	STALL bit A 1 in this bit causes STALL to be returned for the token.
3	***EnACK***	R/W	ACK bit 1: Causes ACK processing to be performed on the token. 0: Causes a NAK to be returned in response to the token.
2	***EnDIR***	R/W	Transfer direction bit 1: IN tokens are processed. 0: OUT tokens are processed.
1	***EnISO***	R/W	Isochronous bit 1: Identifies an isochronous endpoint. 0: Identifies an endpoint other than isochronous endpoints.
0	***EnBNK***	R/W	Bank select bit 1: Bank 1 0: Bank 0

●Endpoint n size register: **EPnCNT (n=1-6)** (FE98h/FE9Ah/FE9Ch/FE9Eh/FEA0h/FEA6h)

Bit	Bit Name	R/W	Function
7	-	-	Reserved
6 - 0	***EnCN6*** **-** ***EnCN0***	R/W	Endpoint n data size Specify the maximum payload size (for OUT endpoints). Specify the transmit data size (for IN endpoints).

●Endpoint n receive size register: **EPnRX (n=1-6)** (FE99h/FE9Bh/FE9Dh/FE9Fh/FEA2h/FEA8h)

Bit	Bit Name	R/W	Function
7	-	-	Reserved
6 - 0	***EnRX6*** **-** ***EnRX0***	R	Endpoint n receive data size Specify the receive data size.

4.1.2. Endpoints 1-6

Endpoints 1-6 transfer data in only one direction. Whether endpoints 1-6 are to be used for transmission or reception is determined by the direction (bEndpointAddress[bit7]) specified in their endpoint descriptor. Endpoints 1-6 are mapped into 8, 16, 32 or 64-byte (× 2 banks) RAM areas, respectively. Care must be taken when specifying the mapping address of an endpoint that it will not overlap the mapping address of another endpoint area. Refer Table 1-2 - Table 1-10 for the mapping address of endpoints 1-6.

The payload size which is actually received for each packet is indicated in the registers (**EPnRX (n=1-6)**). The registers (**EPnCNT** (**n=1-6**)) must be loaded with the size of data to be transmitted in the transmission mode (IN transfers) and with the maximum payload size of the endpoint in the receive mode (OUT transfers).

Table 4-2, 4-3 shows the outline of endpoint n (n=1-6) and the sizes of data packets.

Name	Receive Mode (OUT)		Transmit Mode (IN)
	Maximum Payload Size	Receive Size	Transmit Size
Endpoint 1 (transmit/receive)	**EP1CNT** (64 bytes max.)	**[EP1RX]**	**EP1CNT** (64 bytes max.)
Endpoint 2 (transmit/receive)	**EP2CNT** (64 bytes max.)	**[EP2RX]**	**EP2CNT** (64 bytes max.)
Endpoint 3 (transmit/receive)	**EP3CNT** (64 bytes max.)	**[EP3RX]**	**EP3CNT** (64 bytes max.)
Endpoint 4 (transmit/receive)	**EP4CNT** (64 bytes max.)	**[EP4RX]**	**EP4CNT** (64 bytes max.)
Endpoint 5 (transmit/receive)	**EP5CNT** (64 bytes max.)	**[EP5RX]**	**EP5CNT** (64 bytes max.)
Endpoint 6 (transmit/receive)	**EP6CNT** (64 bytes max.)	**[EP6RX]**	**EP6CNT** (64 bytes max.)

Table 4-2 Endpoints for Data Transfers

Transfer Type	Maximum Packet Size	Packet Transmit/Receive Size
Bulk	Either of 8, 16 ,32, and 64 bytes	0-Max. packet size
Interrupt	Any integer value between 0-64 bytes	
Isochronous	Any integer value between 0-64 bytes	

Table 4-3 Packet Sizes for Data Transfers

Note: The actual payload size of each packet type is variable within the range not exceeding the endpoint's maximum packet size (bMaxPacketSize) that is defined in the respective endpoint descriptor.

4.1.3. Setting Up the Transmission Mode

The LC87F1M16A responds to control transfers according to the setting in the **EPnSTA**.

bit:	7	6	5	4	3	2	1	0
EPnSTA	***EnEN***	***EnTGL***	***EnOVR***	***EnSTL***	***EnACK***	***EnDIR***	***EnISO***	***EnBNK***

Bit	Name	Setting
7	***EnEN***	If this bit is set to 1, the endpoint n is enabled. This bit must be set to 1 when endpoint n is enabled by requests of SetConfiguration/SetInterface.
6	***EnTGL***	This bit must be reset to 0 during the endpoint n initialization for bulk or interrupt transfers. The state of this bit is automatically inverted on each normal termination of the transactions. This bit has no meaning in the isochronous transfers.
5	***EnOVR***	Set when the volume of data exceeding the maximum payload size is received.
4	***EnSTL***	If this bit is set when the endpoint n is enabled (***EnEN***=1), the endpoint n sends a STALL in response to an IN/OUT token.
3	***EnACK***	If this bit is 0, endpoint n responds with a NAK handshake to a token defined in ***EnDIR*** bit. If this bit is 1, the ACK processing is executed to a token defined in ***EnDIR*** bit.
2	***EnDIR***	Set the direction of endpoint n during the endpoint n initialization. If this bit is 0, endpoint n can accept an OUT token only. If 1, endpoint n can accept an IN token only.
1	***EnISO***	Set the transfer type of endpoint n during the endpoint n initialization. If this bit is 0, it identifies the bulk or interrupt endpoint. If 1, it identifies the isochronous endpoint.
0	***EnBNK***	Switch the bank of endpoint 1-6. Set to 0 when endpoint n bank 0 is used. Set to 1 when endpoint n bank 1 is used.

4.2. Outline of Bulk Transfer

Bulk transfers are most often used to transfer a large volume of data asynchronously. Bulk transfers include bulk IN transfers from devices to the host and bulk OUT transfers from the host to devices. A transaction consists of token (IN or OUT), data (DATA0/1), and handshake packets. As many transactions as required according to the volume of data are repeated in a bulk transfer. In this case, DATA1 and DATA0 are used alternately as the PID of the data packets so that retransmitted and new data packets can be distinguished (the first PID is DATA0).

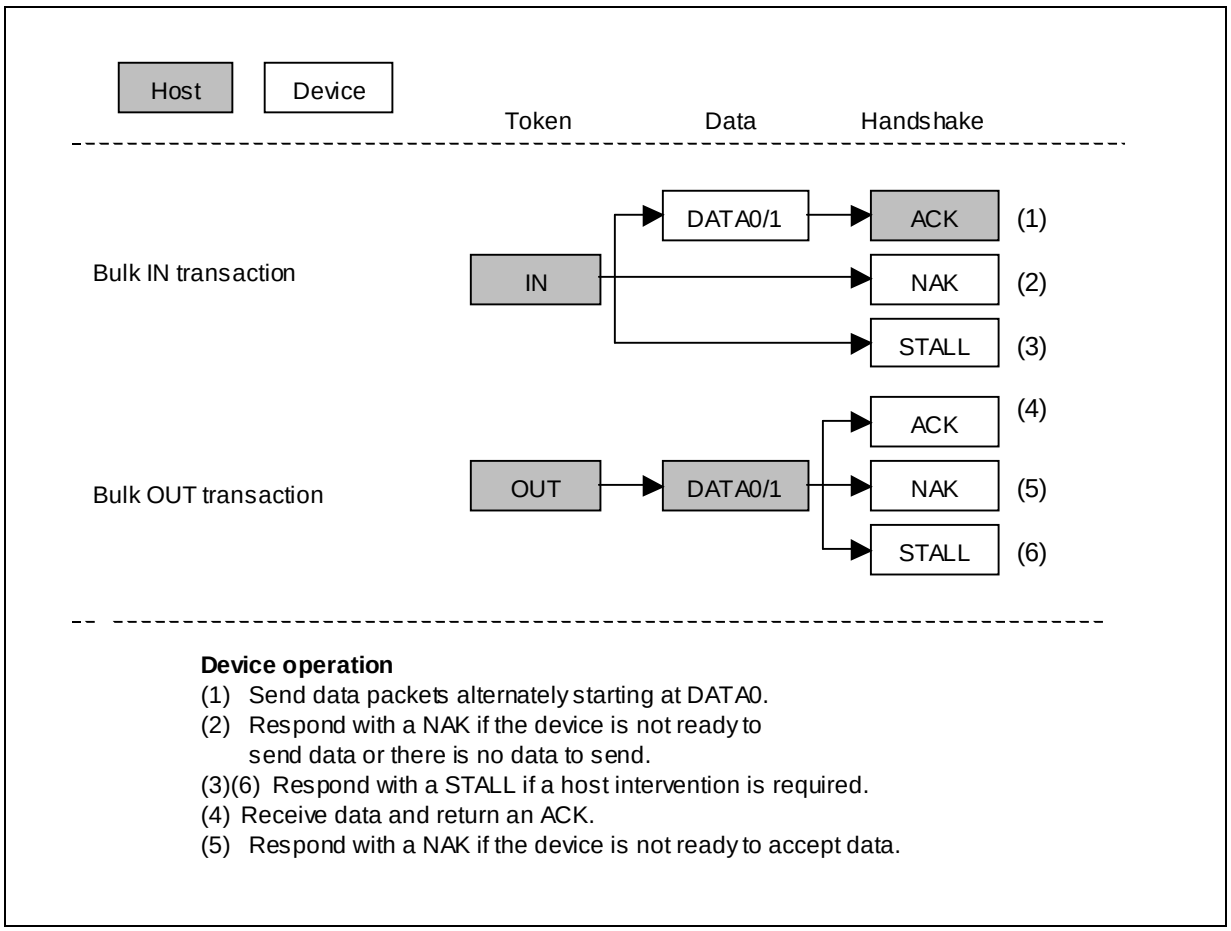
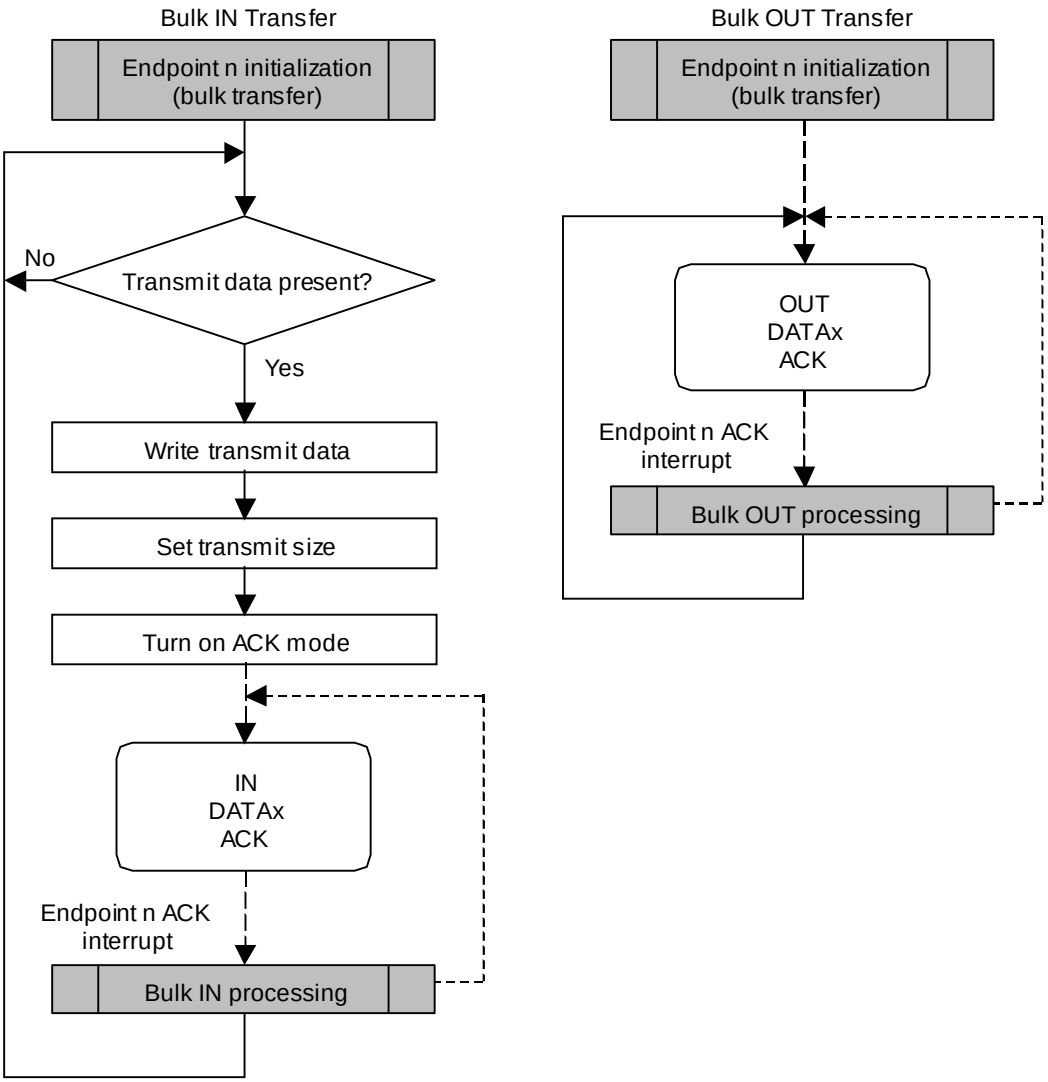


Figure 4-1 Bulk Transfer Transactions

4.2.1. Outline of Process

LC87F1M16A can achieve bulk transfer processing through the initialization for initiating bulk transfers and the processing of endpoint n ACK interrupt which occurs on each normal termination of each transaction. The figure below shows the outline of bulk transfer processing.



4.2.2. Endpoint n Initialization (bulk transfer)

A device cannot perform data communications until it is configured and enters the Configured state. This application note assumes that the devices are initialized during USB bus reset interrupt processing and those endpoints 1-6 are disabled (see 2.4, “USB Bus Reset Interrupts”). It is necessary to set up endpoints n (n=1-6) to perform bulk transfers. Make sure that endpoint n (n=1-6) is initialized when a request that will affect that endpoint (SetConfiguration / SetInterface / ClearFeature[ENDPOINT_HALT]) is accepted.

An example of endpoint n initialization processing is shown in Figure 4-2. When IN transfers, set up **EPnSTA** and enable the endpoint n ACK interrupts. When OUT transfers, set up **EPnSTA** and **EPnCNT** and enable the endpoint n ACK interrupts.

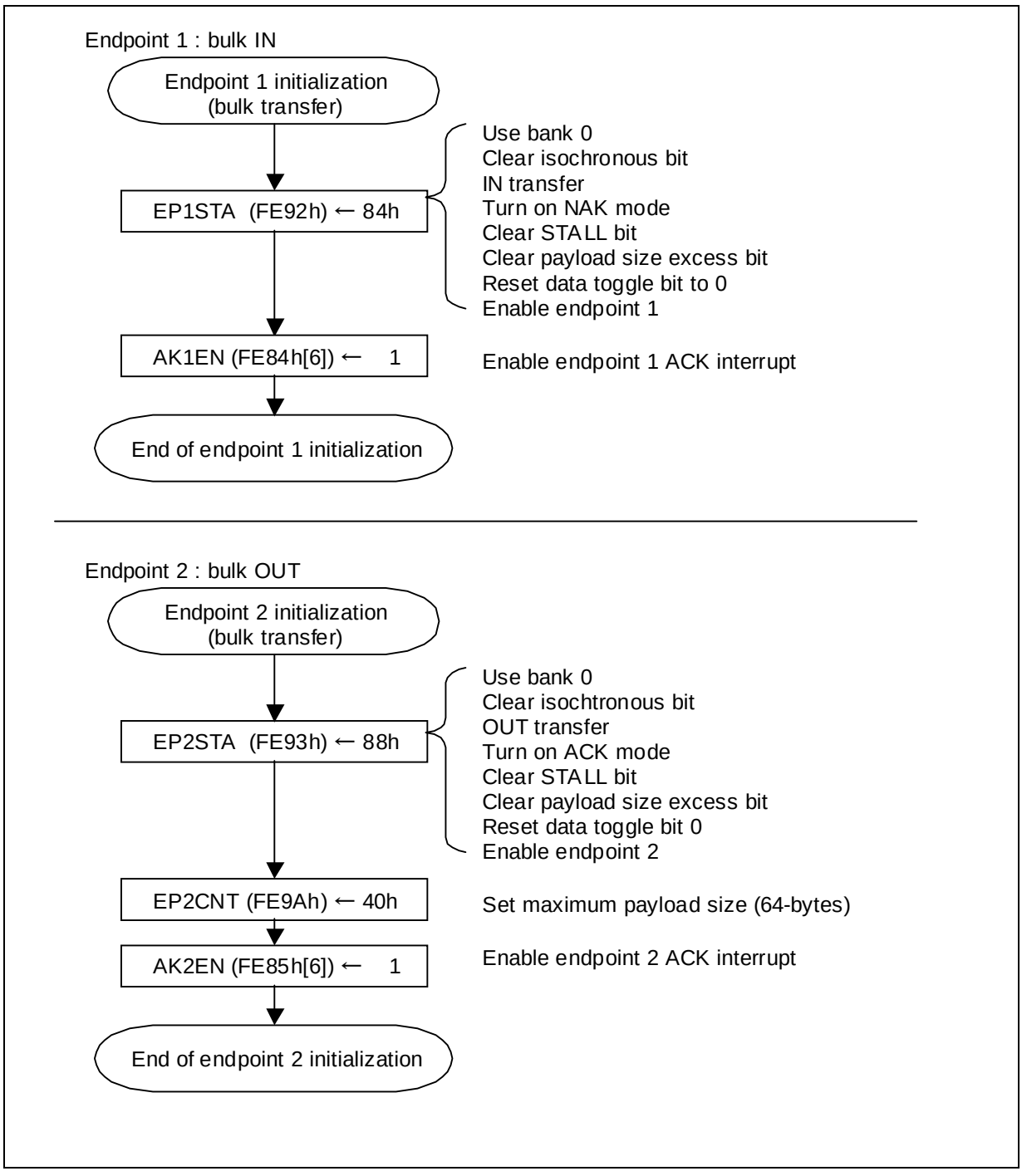


Figure 4-2 Example of Endpoint n Initialization for Bulk Transfers

4.2.3. Endpoint Descriptor Example

Endpoints 1-6 can be used in bulk transfers. Since data is transferred in one direction in bulk transfers, it is necessary to specify the data direction (transmit or receive) for each endpoint to be used. Specify the endpoint number to be used and the data direction in the endpoint address field (bEndpointAddress) in the endpoint descriptor and one of the size values 8, 16, 32, and 64 in the maximum packet size field (bMaxPacketSize).

Field	Size (in Bytes)	Value	Description
bLength	1	07h	Descriptor size
bDescriptorType	1	05h	ENDPOINT descriptor
bEndpointAddress	1	(One of the values listed below) 01h 02h 03h 04h 05h 06h 81h 82h 83h 84h 85h 86h	01h: Endpoint 1 OUT 02h: Endpoint 2 OUT 03h: Endpoint 3 OUT 04h: Endpoint 4 OUT 05h: Endpoint 5 OUT 06h: Endpoint 6 OUT 81h: Endpoint 1 IN 82h: Endpoint 2 IN 83h: Endpoint 3 IN 84h: Endpoint 4 IN 85h: Endpoint 5 IN 86h: Endpoint 6 IN
bmAttributes	1	02h	Bulk transfer
wMaxPacketSize	2	(One of the values listed below) 0008h 0010h 0020h 0040h	The maximum packet size (in bytes) must be set to one of the following values: 8, 16, 32, 64
bInterval	1	00h	Polling interval (in ms)

Table 4-4 Endpoint Descriptor (Bulk Transfer)

4.3. Bulk IN Transfers

4.3.1. Bulk IN Setup

Initialization must be performed before bulk IN transfer starts.
(see 4.2.2, “Endpoint n Initialization (bulk transfer)”).

The preparatory steps shown below are required to send data in the IN transactions in bulk transfer.

- Writing data
  Write data into the endpoint n transmit data area.

  * This step is unnecessary when sending a null packet with a 0-byte data size.

- Setting the transmit size
  Load **EPnCNT** with the byte count of data to be transmitted.

  * Make sure that the byte count of the transmit data does not exceed the maximum payload size of the endpoint n. Specify a 0 for a null packet.

- Turning on the ACK mode
  Set the ACK bit in **EPnSTA** (***EnACK***=1).

When the newly data to be transmitted is present after initialization of endpoint or completion of transfer, the bulk IN transfer is started by performing the preparation processing listed above. If the data is sent in multiple IN transactions, the above processing is performed after each normal termination of each transaction. This application note assumes that this processing is accomplished by Bulk IN processing (see Section 4.3.3.).

4.3.2. Bulk IN Operation

The bulk IN transfer operation proceeds according to the settings of the status register **EPnSTA** and size register **EPnCNT** of the endpoint n used. If the endpoint n is configured in the ACK mode (***EnACK***=1), the reception of an IN token is followed by the transmission of the data stored in the endpoint n data area in the form of a data packet. After the transmission processing, when an ACK is received, the endpoint n is automatically placed into the NAK mode (***EnACK***=0) and the data toggle bit (***EnTGL***) is inverted. The endpoint n ACK interrupt occurs on completion of the IN transaction and the endpoint n ACK interrupt processing routine must perform the Bulk IN processing. The Bulk IN operation flow of the Bulk IN transfer is shown in Figure 4-3.

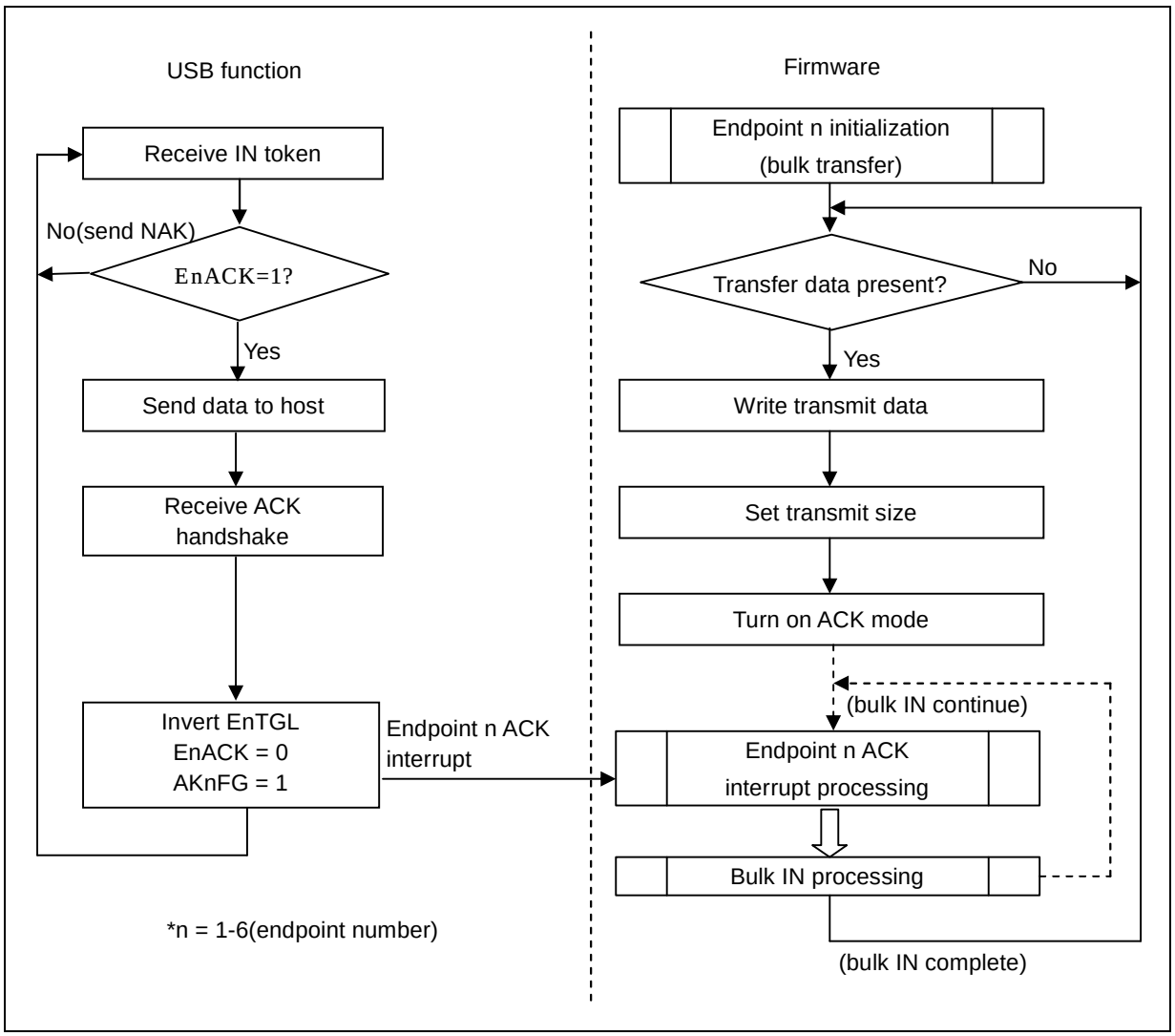


Figure 4-3 Bulk IN Operation Flow

4.3.3. Bulk IN Processing

The endpoint n ACK interrupt flag (***AKnFG***) of the endpoint n interrupt register **EPnINT** is set and an endpoint n ACK interrupt is generated when a bulk transfer IN transaction terminates normally,.
The following actions are automatically taken when an endpoint n ACK interrupt occurs:

- Turn on the NAK mode (*EnACK*=0).
- Invert the toggle bit (EnTGL).

The endpoint n ACK interrupt processing routine must perform the following bulk IN processing:

- If data to be sent remains, take preparatory actions for the next transmission.

The Bulk IN processing to be performed during the endpoint n ACK interrupt processing routine looks like as shown below (see 2.5.1, “Endpoint n ACK Interrupts”).

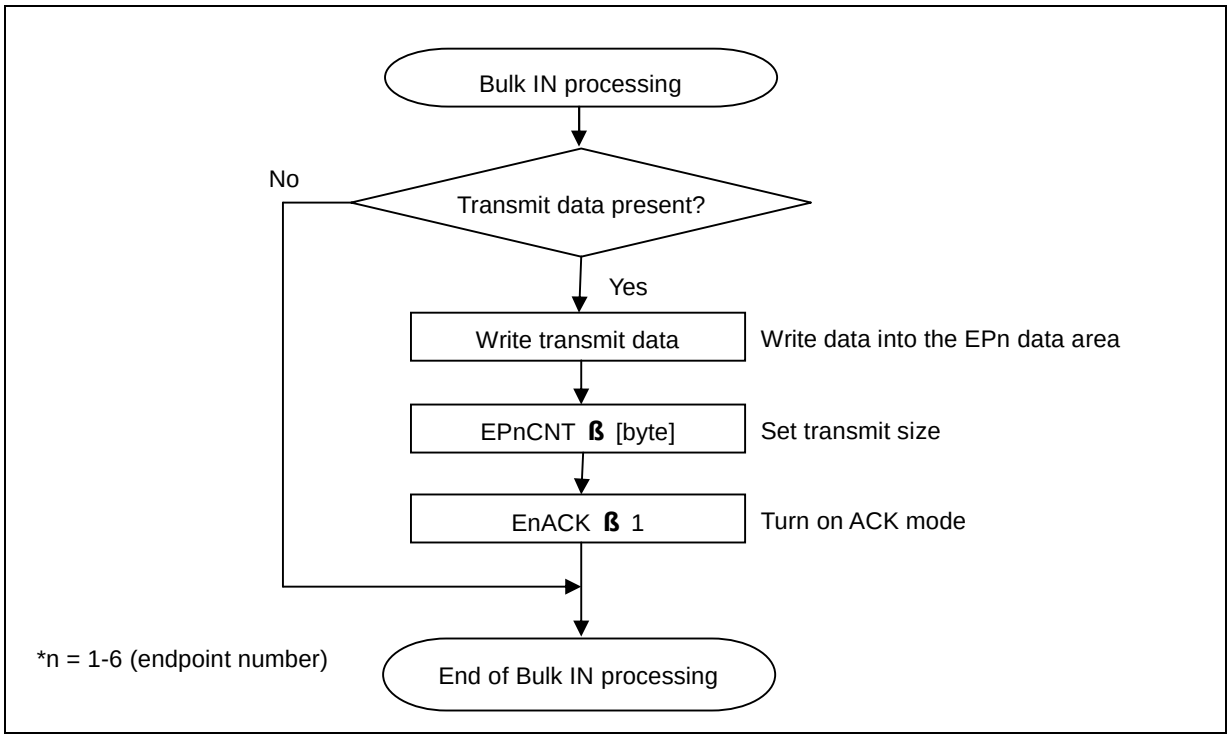


Figure 4-4 Example of Bulk IN Processing

4.3.4. Transmission Errors

- Stall
  The STALL bit (***EnSTL***) of the endpoint n status register **EPnSTA** must be set to 1 by firmware if the endpoint HALT is set or the endpoint n can transmit no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN/OUT tokens received from the host.

- Data retransmission
  If the device fails to receive an ACK handshake packet from the host after sending a 0-byte DATA1 data packet normally, it is necessary to retransmit the data packet. Since the LC87F1M16A performs retransmission automatically by hardware, there is no need for the firmware to control the retransmission processing.

4.4. Bulk OUT Transfers

4.4.1. Bulk OUT Setup

Initialization must be performed before bulk OUT transfer starts.
(see 4.2.2, “Endpoint n Initialization (bulk transfer)”).

The preparatory steps shown below are required to receive data during the OUT transactions in a bulk transfer.

- Turning on the ACK mode
  Set the ACK bit in **EPnSTA** to turn on the ACK mode (***EnACK***=1).

The bulk OUT transfer is started by performing the preparation processing above. This application note assumes that this processing is accomplished by endpoint n initialization(see Section 4.2.2.) and bulk OUT processing (see Section 4.4.3.).

4.4.2. Bulk OUT Operation

The operation of the bulk OUT transfer proceeds according to the settings of the endpoint n status register **EPnSTA**. If the endpoint n is configured in the ACK (***EnACK***=1), the reception of an OUT token is followed by the reception of a data packet and the received data is stored in the endpoint n data area. When an ACK is transmitted after the receive processing, the endpoint n is automatically configured into the NAK mode (***EnACK***=0) and the data toggle bit (***EnTGL***) is inverted. The endpoint n ACK interrupt occurs on completion of the OUT transaction and the endpoint n ACK interrupt processing routine must perform the Bulk OUT processing. The Bulk OUT operation flow of the Bulk OUT transfer is shown in Figure 4-5.

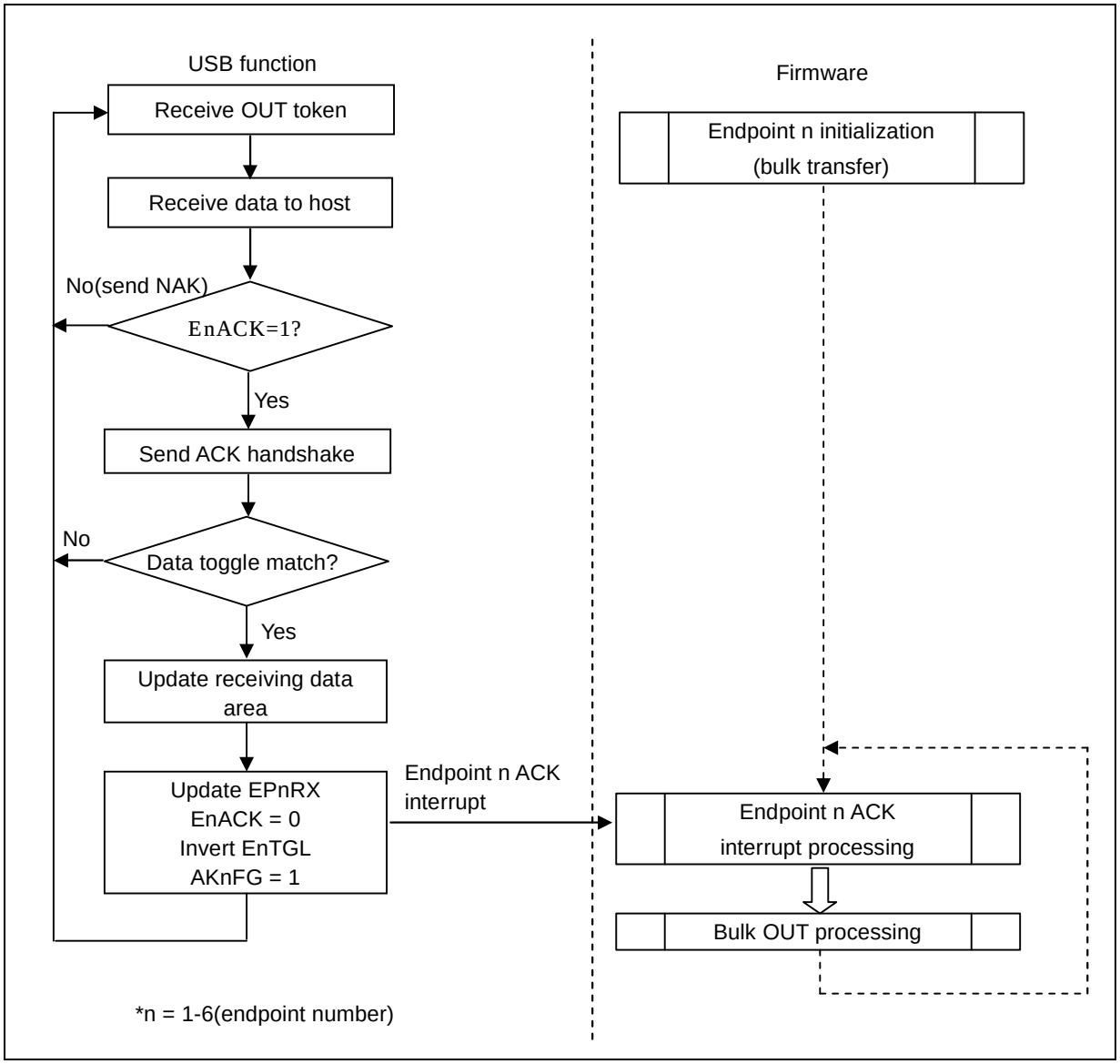


Figure 4-5 Bulk OUT Operation Flow

4.4.3. Bulk OUT Processing

When the LC87F1M16A transmits an ACK handshake packet after receiving data in an OUT transaction of a bulk transfer, the endpoint n ACK interrupt flag (***AKnFG***) of the endpoint n interrupt register **EPnINT** is set and an endpoint n interrupt is generated.
The following actions are automatically taken when an endpoint n ACK interrupt occurs:

- Update the receive data area.
- Update the receive data size (EPnRX).
- Turn on the NAK mode (***EnACK***=0).
- Invert the toggle bit (***EnTGL***).

The endpoint n ACK interrupt processing routine must perform the following bulk OUT processing:

- Read the receive data
- Take preparatory actions for the next reception.

The Bulk OUT processing to be performed during the endpoint n ACK interrupt processing routine looks like as shown below (see 2.5.1, “Endpoint n ACK Interrupts”).

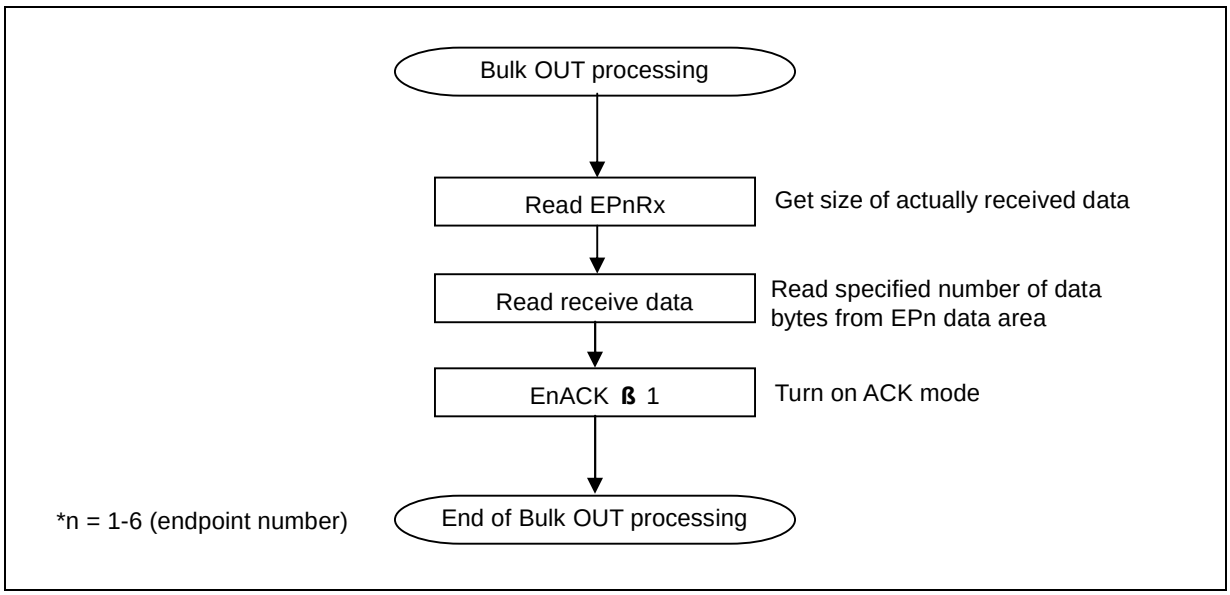


Figure 4-6 Example of Bulk OUT Processing

4.4.4. Receive Errors

- Data corruption
  The LC87F1M16A makes error checks by hardware. When the LC87F1M16A detects an error, it returns no response regardless of the transmission mode settings. In such a case, the endpoint 0 error interrupt flag (***ERnFG***) of the endpoint 0 interrupt register **EPnINT** is set to 1. Error recovery actions, then, must be taken as required. Normally, the routine can continue data receive processing following the data retransmission from the host.

- Toggle mismatch
  The LC87F1M16A performs hardware toggle check when it receives data. The LC87F1M16A responds with an ACK handshake if no error is found in the data. No endpoint n ACK interrupt is generated, however, if a toggle mismatch is found. In this case, neither the receive data area is updated nor the toggle bit (***EnTGL***) or ACK bit (***EnACK***) is changed. Consequently, the routine can continue data receive processing.

- Maximum payload exceeded
  The payload excess bit (***EnOVR***) of the endpoint n status register **EPnSTA** is set to 1 even when the LC87F1M16A receives data without an error if the byte count of the received data exceeds the maximum payload size specified in **EPnMP**. In such a case, the routine must take error recovery actions as required. Normally, the routine can continue data receive processing.

- Stall
  The STALL bit (***EnSTL***) of the endpoint n status register **EPnSTA** must be set to 1 by firmware if the endpoint HALT is set or the endpoint n can receive no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN/OUT tokens received from the host.

4.5. Outline of Interrupt Transfer

Interrupt transfers are used to transfer small volumes of data at predetermined intervals. Interrupt transfers include interrupt IN transfers from devices to the host and interrupt OUT transfers from the host to devices. Like a bulk transfer, an interrupt transfer transaction consists of token (IN or OUT), data (DATA0/1), and handshake packets. As many transactions as required according to the volume of data are repeated in an interrupt transfer. In this case, DATA1 and DATA0 are used alternately as the PID of the data packets so that retransmitted and new data packets can be distinguished (the first PID is DATA0).

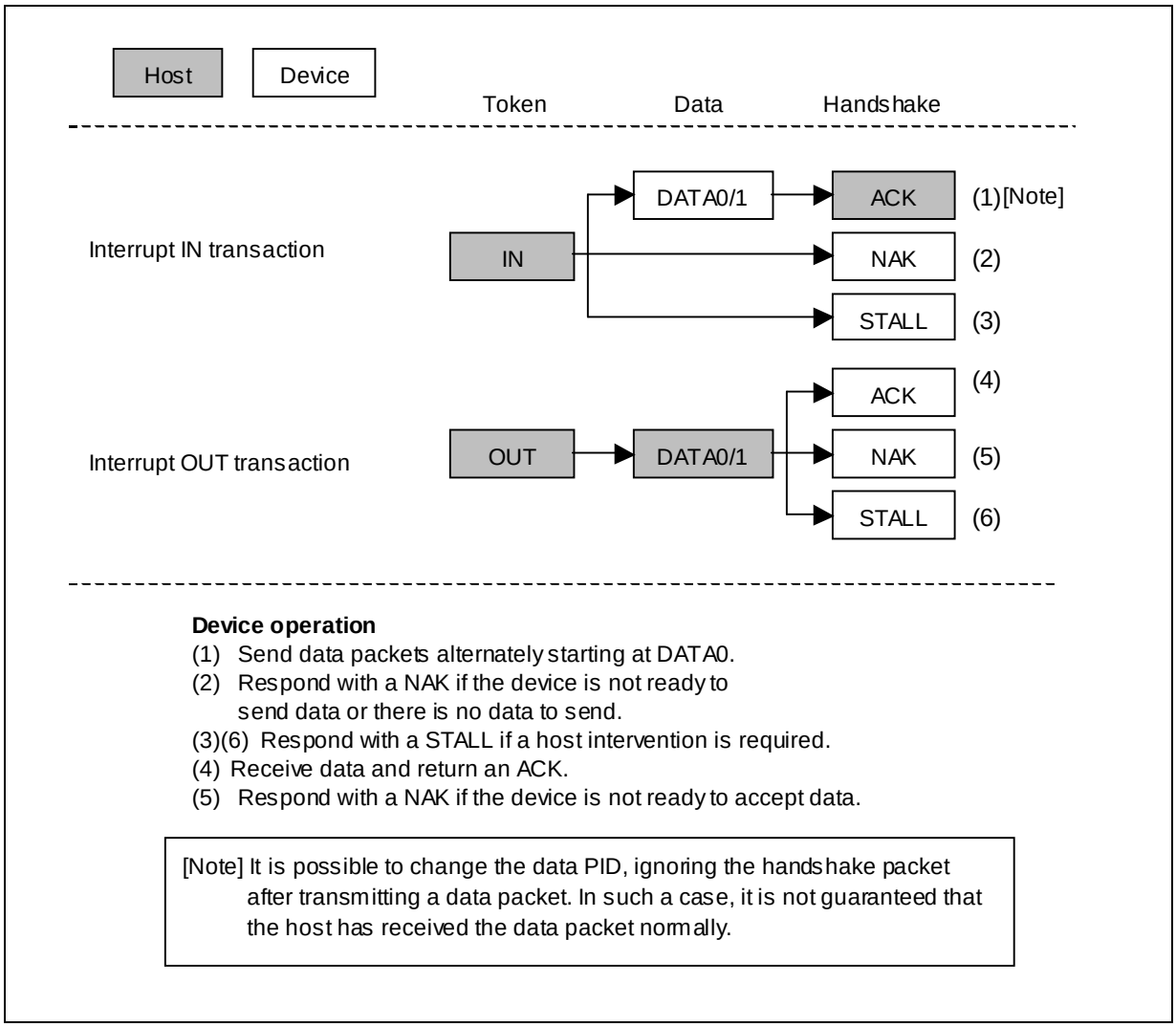
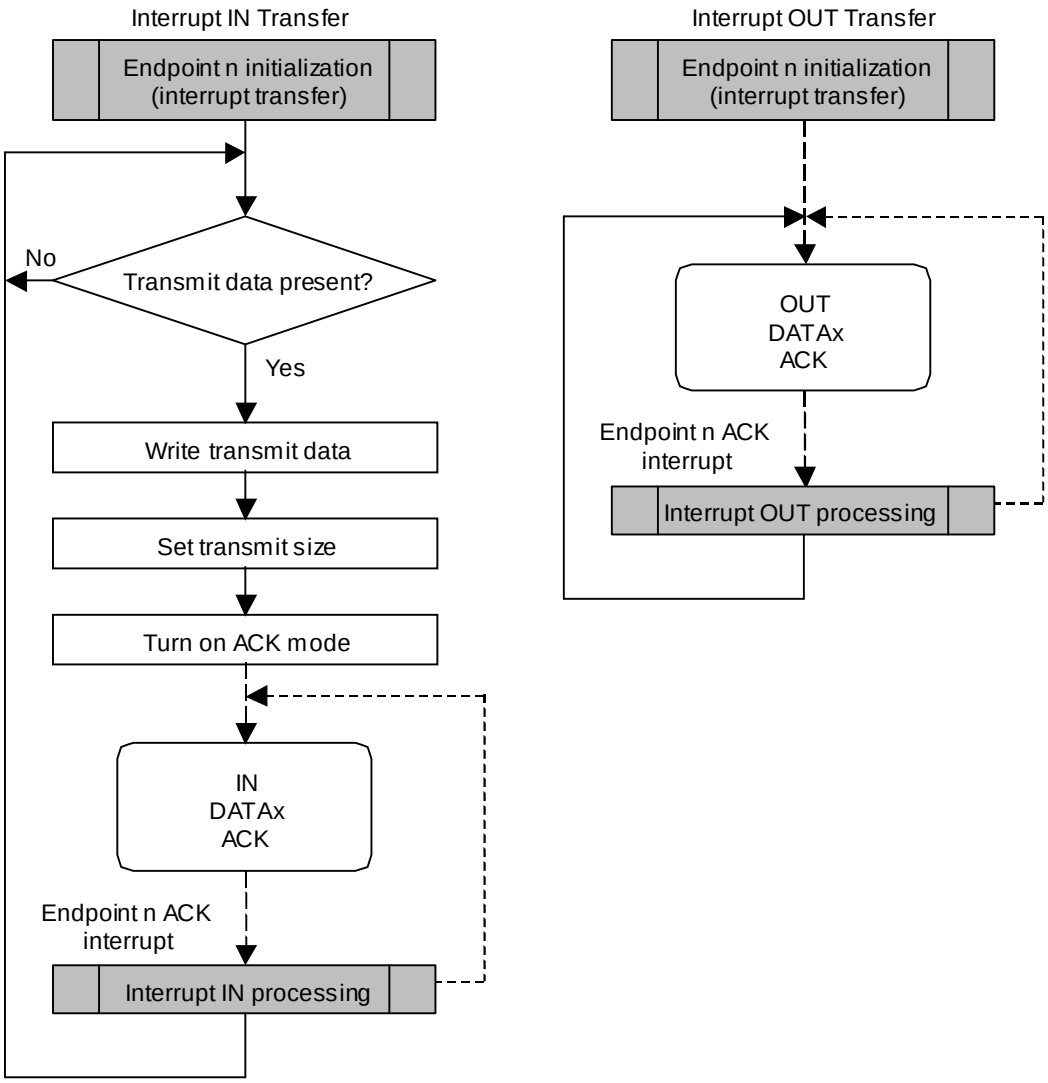


Figure 4-7 Interrupt Transfer Transactions

4.5.1. Outline of Process

LC87F1M16A can achieve interrupt transfer processing through the initialization for initiating interrupt transfers and the processing of endpoint n ACK interrupt which occurs on each normal termination of each transaction. The figure below shows the outline of interrupt transfer processing.



4.5.2. Endpoint n Initialization (interrupt transfer)

A device cannot perform data communications until it is configured and enters the Configured state. This application note assumes that the devices are initialized during USB bus reset interrupt processing and those endpoints 1-6 are disabled (see 2.4, “USB Bus Reset Interrupts”). It is necessary to set up endpoints n (n=1-6) to perform interrupt transfers. Make sure that endpoint n (n=1-6) is initialized when a request that will affect that endpoint (SetConfiguration / SetInterface / ClearFeature[ENDPOINT_HALT]) is accepted.

An example of endpoint n initialization processing is shown in Figure 4-8. When IN transfers, set up **EPnSTA** and enable the endpoint n ACK interrupts. When OUT transfers, set up **EPnSTA** and **EPnCNT** and enable the endpoint n ACK interrupts.

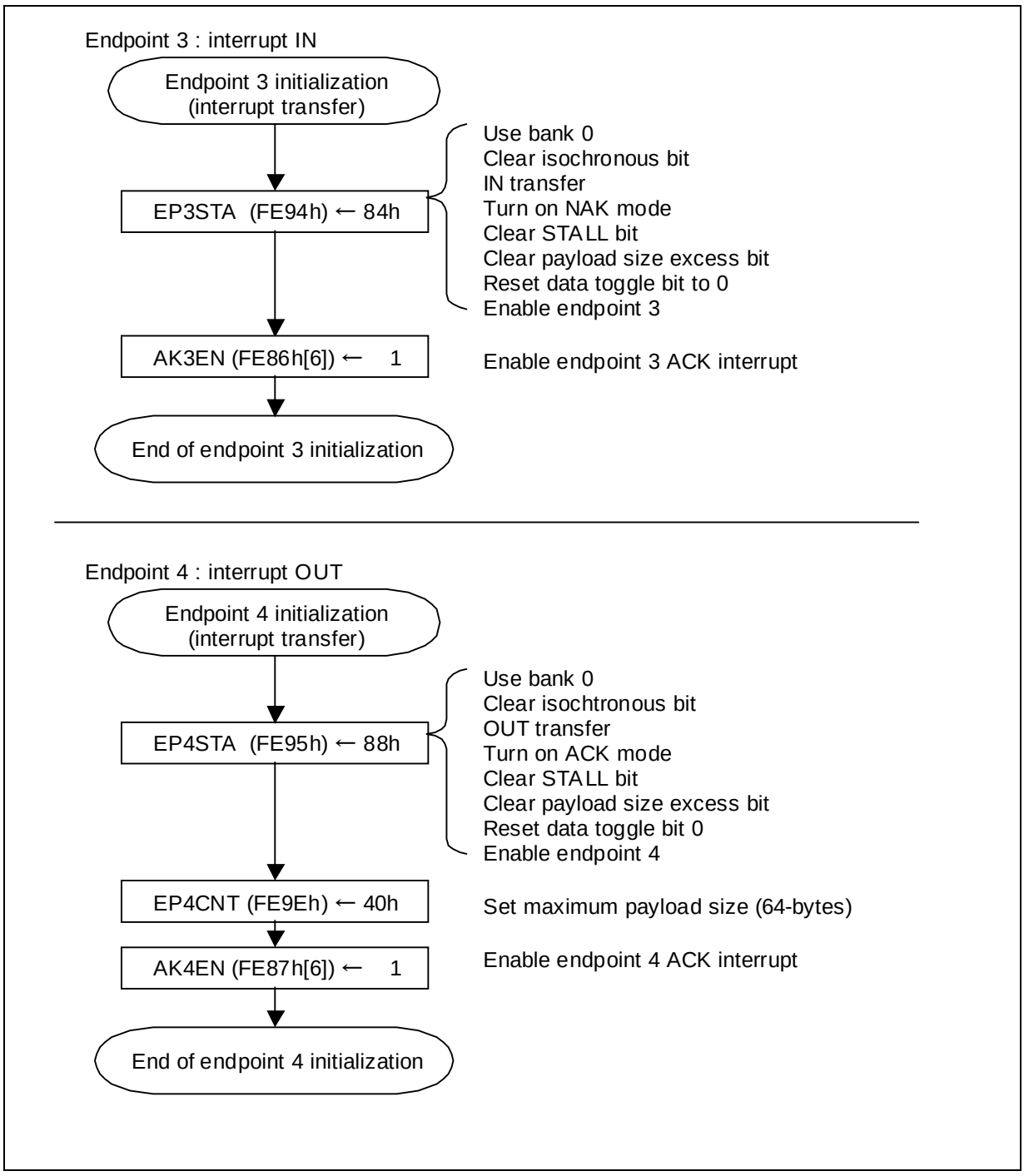


Figure 4-8 Example of Endpoint Initialization for Interrupt Transfers

4.5.3. Endpoint Descriptor Example

Endpoints 1-6 can be used in interrupt transfers. Since data is transferred in one direction in interrupt transfers, it is necessary to specify the data direction (transmit or receive) for each endpoint to be used. Specify the endpoint number to be used and the data direction in the endpoint address field (bEndpointAddress) in the endpoint descriptor and an integer value 0-64 in the maximum packet size field (bMaxPacketSize). Also specify an integer value 1 to 255 in the required maximum period field (bInterval). When an integer value N is specified in bInterval, it is guaranteed that polling occurs at least once per N milliseconds.

Field	Size (in Bytes)	Value	Description
bLength	1	07h	Descriptor size
bDescriptorType	1	05h	ENDPOINT descriptor
bEndpointAddress	1	(One of the values listed below) 01h 02h 03h 04h 05h 06h 81h 82h 83h 84h 85h 86h	01h: Endpoint 1 OUT 02h: Endpoint 2 OUT 03h: Endpoint 3 OUT 04h: Endpoint 4 OUT 05h: Endpoint 5 OUT 06h: Endpoint 6 OUT 81h: Endpoint 1 IN 82h: Endpoint 2 IN 83h: Endpoint 3 IN 84h: Endpoint 4 IN 85h: Endpoint 5 IN 86h: Endpoint 6 IN
bmAttributes	1	03h	Interrupt transfer
wMaxPacketSize	2	(One of the values listed below) 0000h-0040h	The maximum packet size (in bytes) must be set to an integer value between 0 and 64.
bInterval	1	(One of the values listed below) 01h-FFh	Polling interval (in ms)

Table 4-5 Endpoint Descriptor (Interrupt Transfer)

4.6. Interrupt IN Transfers

4.6.1. Interrupt IN Setup

Initialization must be performed before interrupt IN transfer starts.
(see 4.5.2, “Endpoint n Initialization (interrupt transfer)”).

The preparatory steps shown below are required to send data during the IN transactions in an interrupt transfer.

- Writing data
  Write data into the endpoint n transmit data area.

  * This step is unnecessary when sending a null packet with a 0-byte data size.

- Setting the transmit size
  Load EPnCNT with the byte count of data to be transmitted.

  * Make sure that the byte count of the transmit data does not exceed the maximum payload size of the endpoint n. Specify a 0 for a null packet. (Refer to 5.7.3, “Interrupt Transfer Packet Size Constraints," of the USB 2.0 Specification.)

- Turning on the ACK mode
  Set the ACK bit in **EPnSTA** (***EnACK***=1).

When the newly data to be transmitted is present after initialization of endpoint or completion of transfer, the interrupt IN transfer is started by performing the preparation processing listed above. If the data is sent in multiple IN transactions, the above processing is performed after each normal termination of the IN transaction. This application note assumes that this processing is accomplished by Interrupt IN processing (see Section 4.6.3.).

4.6.2. Interrupt IN Operation

The interrupt IN transfer operation proceeds according to the settings of the status register **EPnSTA** and size register **EPnCNT** of the endpoint n used. If the endpoint n is configured in the ACK mode (***EnACK***=1), the reception of an IN token is followed by the transmission of the data stored in the endpoint n data area in the form of a data packet. After the transmission processing, when an ACK is received, the endpoint n is automatically placed into the NAK mode (***EnACK***=0) and the data toggle bit (***EnTGL***) is inverted. The endpoint n ACK interrupt occurs on completion of the IN transaction and the endpoint n ACK interrupt processing routine must perform the Interrupt IN processing. The Interrupt IN operation flow of the Interrupt IN transfer is shown in Figure 4-9.

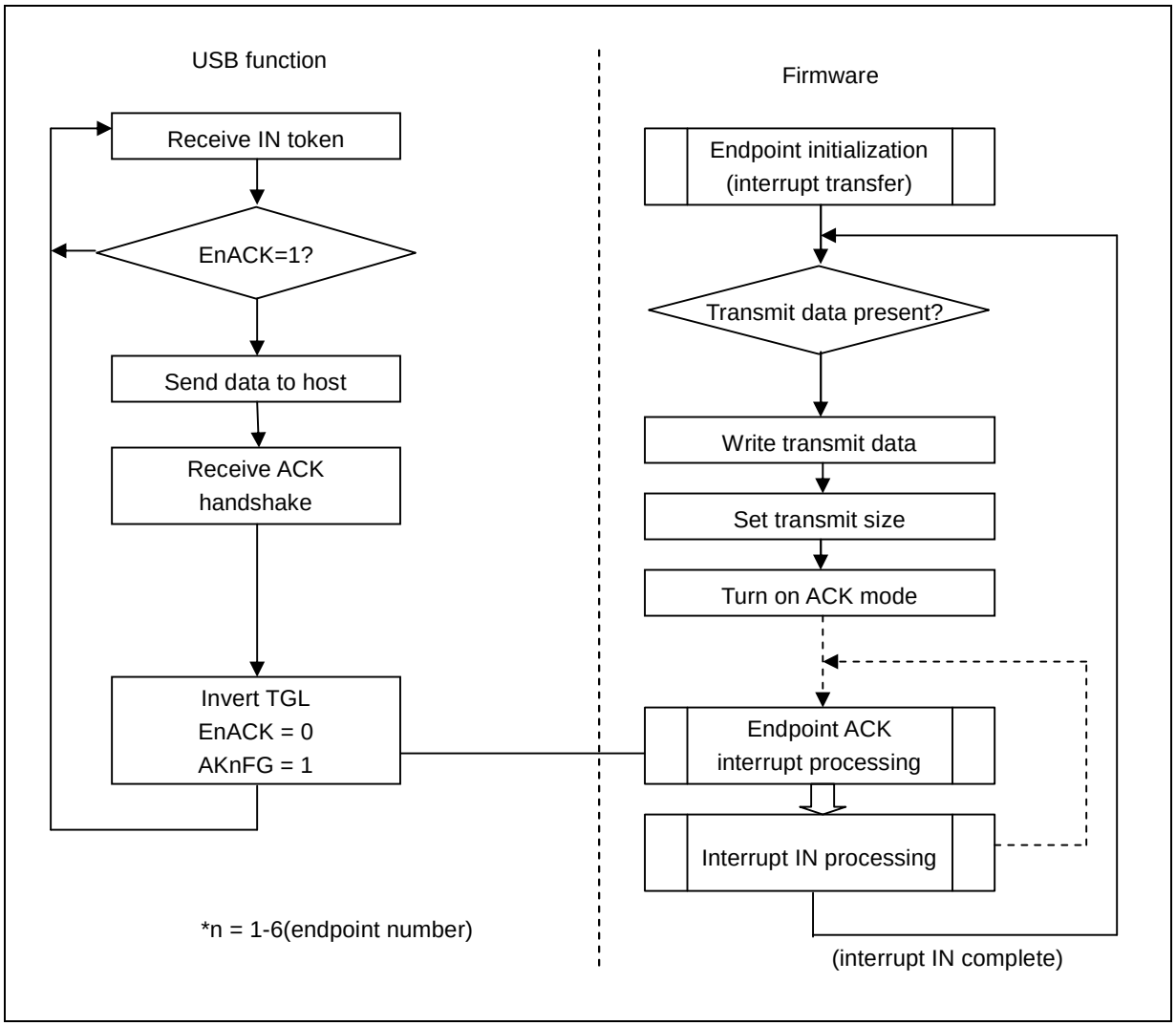


Figure 4-9 Interrupt IN Operation Flow

4.6.3. Interrupt IN Processing

The endpoint n ACK interrupt flag (***AKnFG***) of the endpoint n interrupt register **EPnINT** is set and an endpoint n ACK interrupt is generated when an interrupt transfer IN transaction terminates normally.
The following actions are automatically taken when an endpoint n ACK interrupt occurs:

- Turn on the NAK mode (***EnACK***=0).
- Invert the toggle bit (***EnTGL***).

The endpoint n ACK interrupt processing routine must perform the following interrupt IN processing:

- Take preparatory actions for the next transmission.

The Interrupt IN processing to be performed during the endpoint n ACK interrupt processing routine looks like as shown below (see 2.5.1, “Endpoint n ACK Interrupts”).

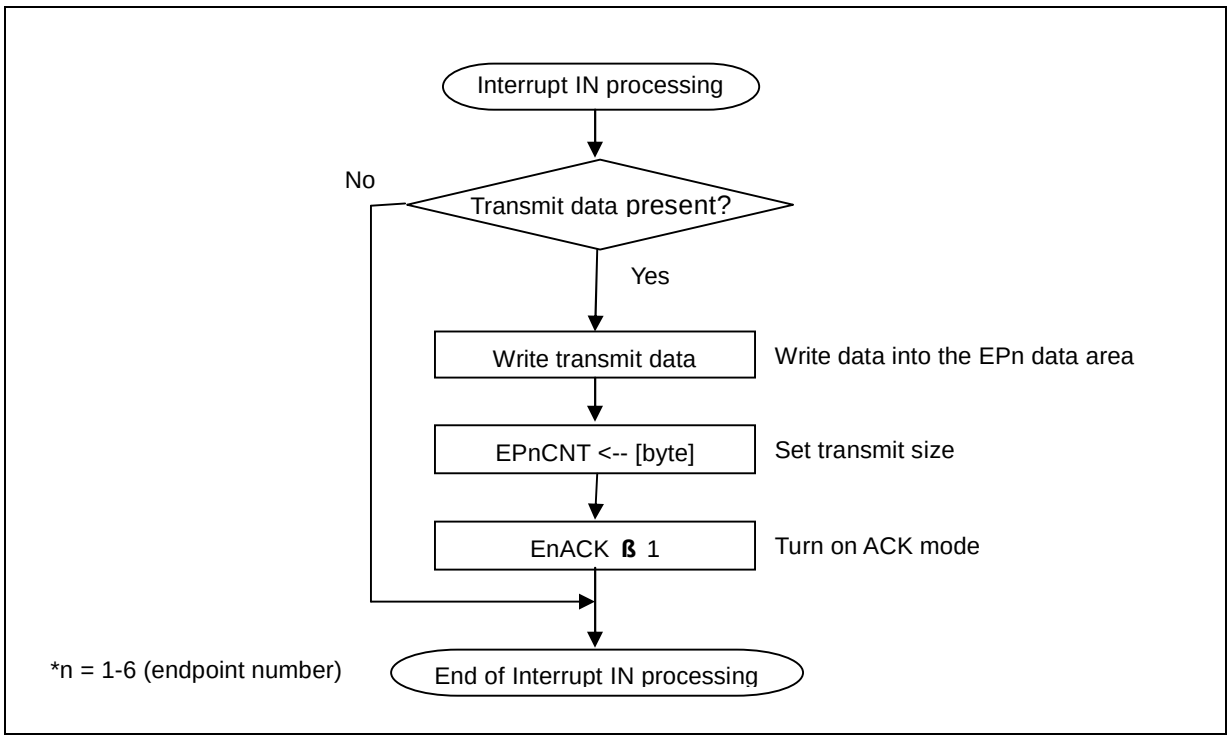


Figure 4-10 Example of Interrupt IN Processing

4.6.4. Transmission Errors

- Stall
  The STALL bit (***EnSTL***) of the endpoint n status register **EPnSTA** must be set to 1 by firmware if the endpoint HALT is set or the endpoint n can transmit no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN/OUT tokens received from the host.

- Data retransmission
  If the device fails to receive an ACK handshake packet from the host after sending a 0-byte DATA1 data packet normally, it is necessary to retransmit the data packet. Since the LC87F1M16A performs retransmission automatically by hardware, there is no need for the firmware to control the retransmission processing.

4.7. Interrupt OUT Transfers

4.7.1. Interrupt OUT Setup

Initialization must be performed before interrupt OUT transfer starts.
(see 4.5.2, “Endpoint n Initialization (interrupt transfer)”).

The preparatory steps shown below are required to receive data during the OUT transactions in an interrupt transfer.

- Turning on the ACK mode
  Set the ACK bit in **EPnSTA** to turn on the ACK mode (***EnACK***=1).

The interrupt OUT transfer is started by performing the preparation processing above. This application note assumes that this processing is accomplished by endpoint n initialization (see Section 4.5.2.) and interrupt OUT processing (see Section 4.7.3.).

4.7.2. Interrupt OUT Operation

The operation of the interrupt OUT transfer proceeds according to the settings of the endpoint n status register **EPnSTA**. If the endpoint n is configured in the ACK (***EnACK***=1), the reception of an OUT token is followed by the reception of a data packet and the received data is stored in the endpoint n data area. When an ACK is transmitted after the receive processing, the endpoint n is automatically configured into the NAK mode (***EnACK***=0) and the data toggle bit (***EnTGL***) is inverted. The endpoint n ACK interrupt occurs on completion of the OUT transaction and the endpoint n ACK interrupt processing routine must perform the Interrupt OUT processing. The Interrupt OUT operation flow of the Interrupt OUT transfer is shown in Figure 4-11.

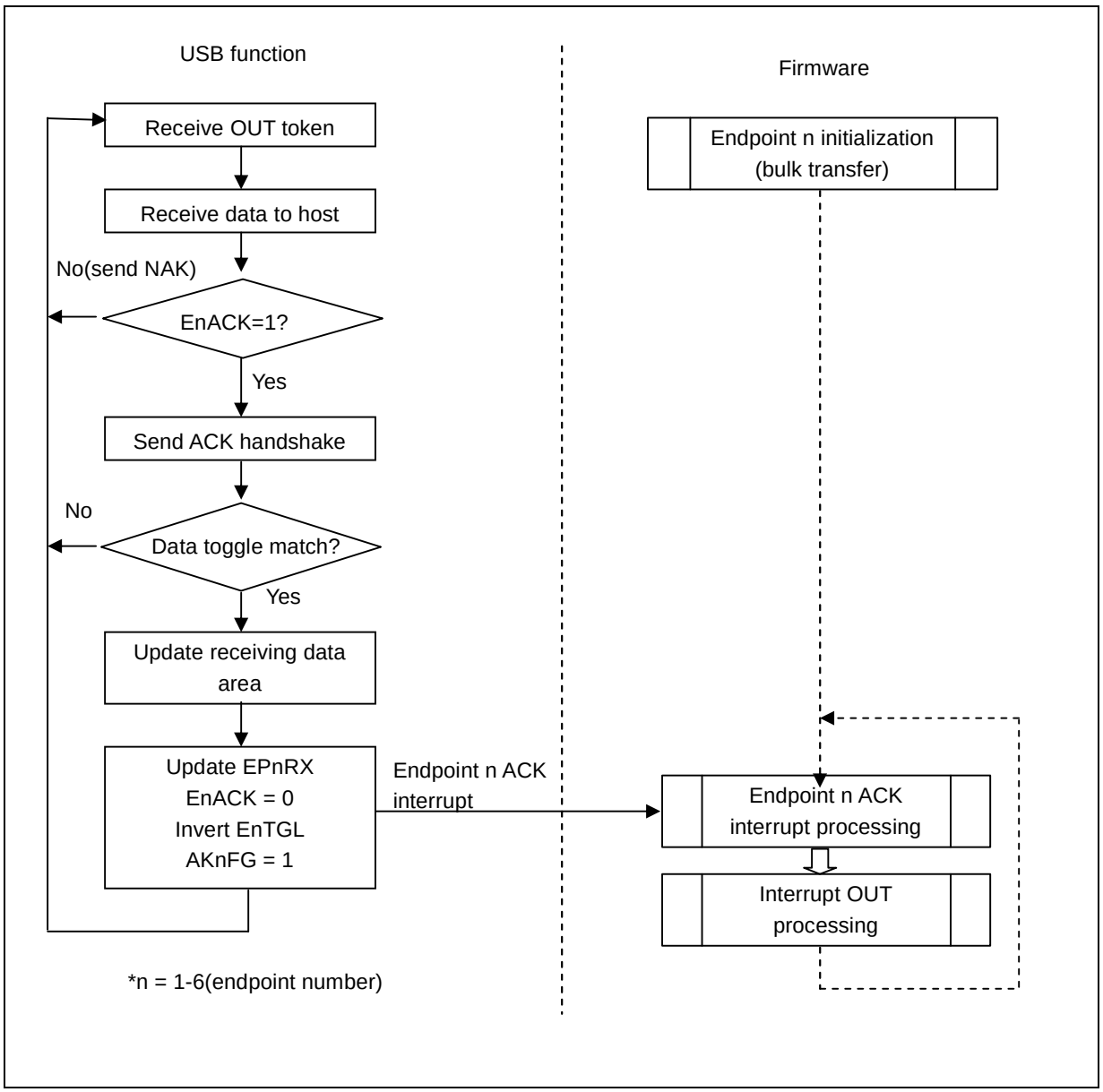


Figure 4-11 Interrupt OUT Operation Flow

4.7.3. Interrupt OUT Processing

When the LC87F1M16A transmits an ACK handshake packet after receiving data in an OUT transaction of an interrupt transfer, the endpoint n ACK interrupt flag (***AKnFG***) of the endpoint n interrupt register **EPnINT** is set and an endpoint n interrupt is generated.
The following actions are automatically taken when an endpoint n ACK interrupt occurs:

- Update the receive data area.
- Update the receive data size (**EPnRX**).
- Turn on the NAK mode (***EnACK***=0).
- Invert the toggle bit (***EnTGL***).

The endpoint n ACK interrupt processing routine must perform the following interrupt OUT processing:

- Read the receive data.
- Take preparatory actions for the next reception.

The interrupt OUT processing to be performed during the endpoint n ACK interrupt processing routine looks like as shown below (see Section 2.5.1, “Endpoint n ACK Interrupts”).

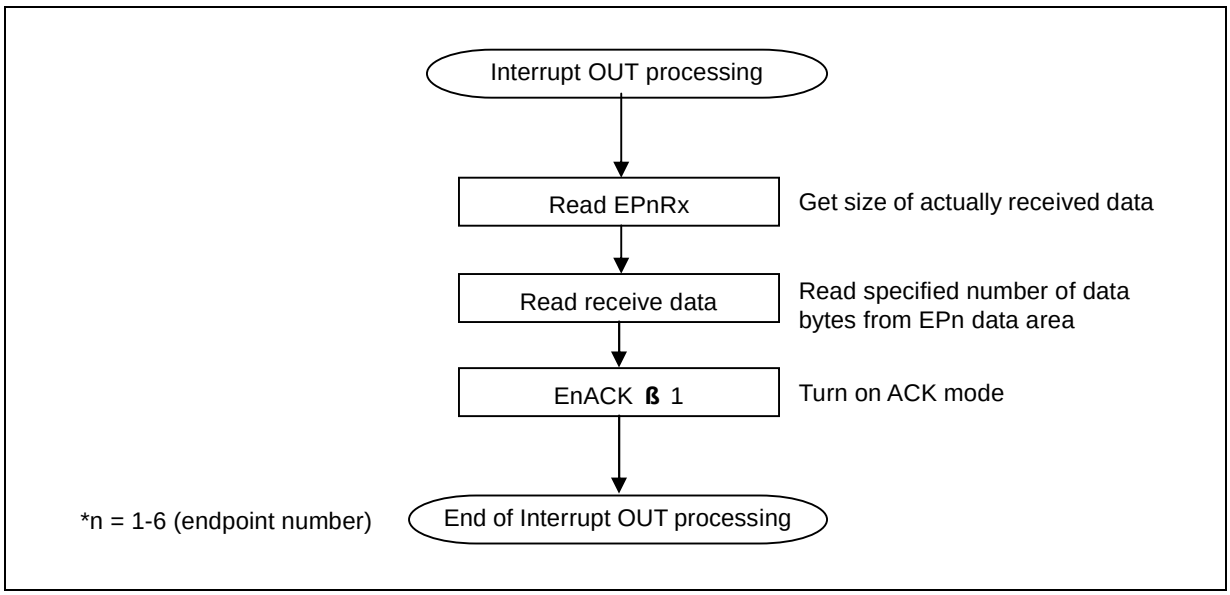


Figure 4-12 Example of Interrupt OUT Processing

4.7.4. Receive Errors

- Data corruption
  The LC87F1M16A makes error checks by hardware. When the LC87F1M16A detects an error, it returns no response regardless of the transmission mode settings. In such a case, the endpoint 0 error interrupt flag (***ERnFG***) of the endpoint 0 interrupt register **EPnINT** is set to 1. Error recovery actions, then, must be taken as required. Normally, the routine can continue data receive processing following the data retransmission from the host.

- Toggle mismatch
  The LC87F1M16A performs hardware toggle check when it receives data. The LC87F1M16A responds with an ACK handshake if no error is found in the data. No endpoint n ACK interrupt is generated, however, if a toggle mismatch is found. In this case, neither the receive data area is updated nor the toggle bit (***EnTGL***) or ACK bit (***EnACK***) is changed. Consequently, the routine can continue data receive processing.

- Maximum payload exceeded
  The payload excess bit (***EnOVR***) of the endpoint n status register **EPnSTA** is set to 1 even when the LC87F1M16A receives data without an error if the byte count of the received data exceeds the maximum payload size specified in **EPnMP**. In such a case, the routine must take error recovery actions as required. Normally, the routine can continue data receive processing.

- Stall
  The STALL bit (***EnSTL***) of the endpoint n status register **EPnSTA** must be set to 1 by firmware if the endpoint HALT is set or the endpoint n can receive no data packet for some reason. This causes the LC87F1M16A to send STALL handshake packets for any IN/OUT tokens received from the host.

4.8. Outline of Isochronous Transfer

Isochronous transfers are used to transfer audio, image, and other multimedia data. It is guaranteed that isochronous transfers distribute a certain amount (1023 bytes or less) of data within a predetermined time (Maximum bytes of all endpoints in LC87F1M16A is 64 bytes). Isochronous transfers include isochronous IN transfers from devices to the host and isochronous OUT transfers from the host to devices. A transaction consists of token (IN or OUT) and data (DATA0) packets; it has no handshake packet. Consequently, neither retransmission is allowed nor error-free communication is guaranteed. Only DATA0 is used as the data packet PID and DATA1 is not used.

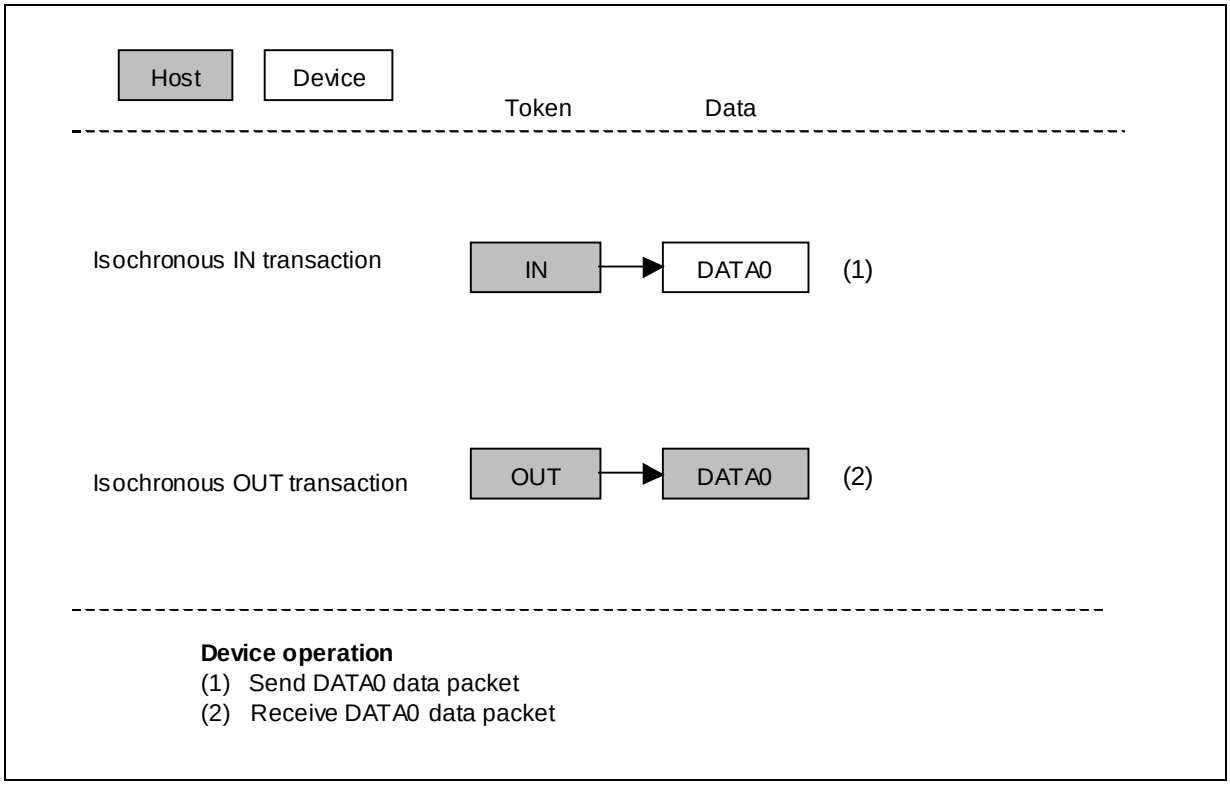
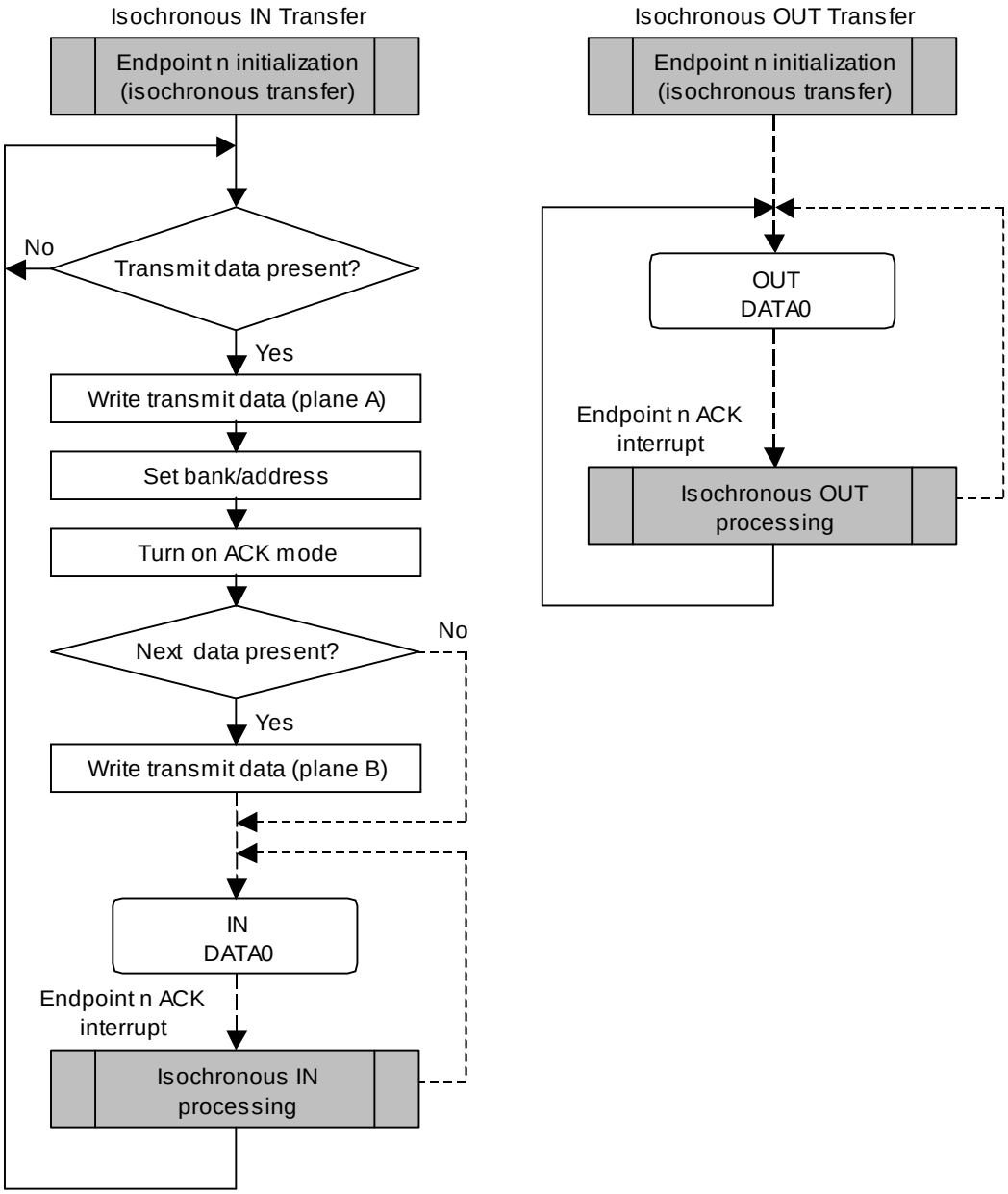


Figure 4-13 Isochronous Transfer Transactions

The LC87F1M16A performs no data toggle check on the data that it received through isochronous OUT transfers. The data packet is received normally whether its PID is DATA0 or DATA1 and the endpoint data area is updated..

4.8.1. Outline of Process

LC87F1M16A can achieve isochronous transfer processing through the initialization for initiating isochronous transfers and the processing of endpoint n ACK interrupt which occurs on each normal termination of each transaction. The figure below shows the outline of isochronous transfer processing.



4.8.2. Endpoint n Initialization (isochronous transfer)

A device cannot perform data communications until it is configured and enters the Configured state. This application note assumes that the devices are initialized during USB bus reset interrupt processing and those endpoints 1-6 are disabled (see 2.4, “USB Bus Reset Interrupts”). It is necessary to set up endpoints n (n=1-6) to perform isochronous transfers. Make sure that endpoint n (n=1-6) is initialized when a request that will affect that endpoint (SetConfiguration / SetInterface / ClearFeature[ENDPOINT_HALT]) is accepted.

An example of endpoint n initialization processing is shown in Figure 4-14. When IN transfers, set up **EPnSTA** and enable the endpoint n ACK interrupts. When OUT transfers, set up **EPnSTA** and **EPnCNT** and enable the endpoint n ACK interrupts.

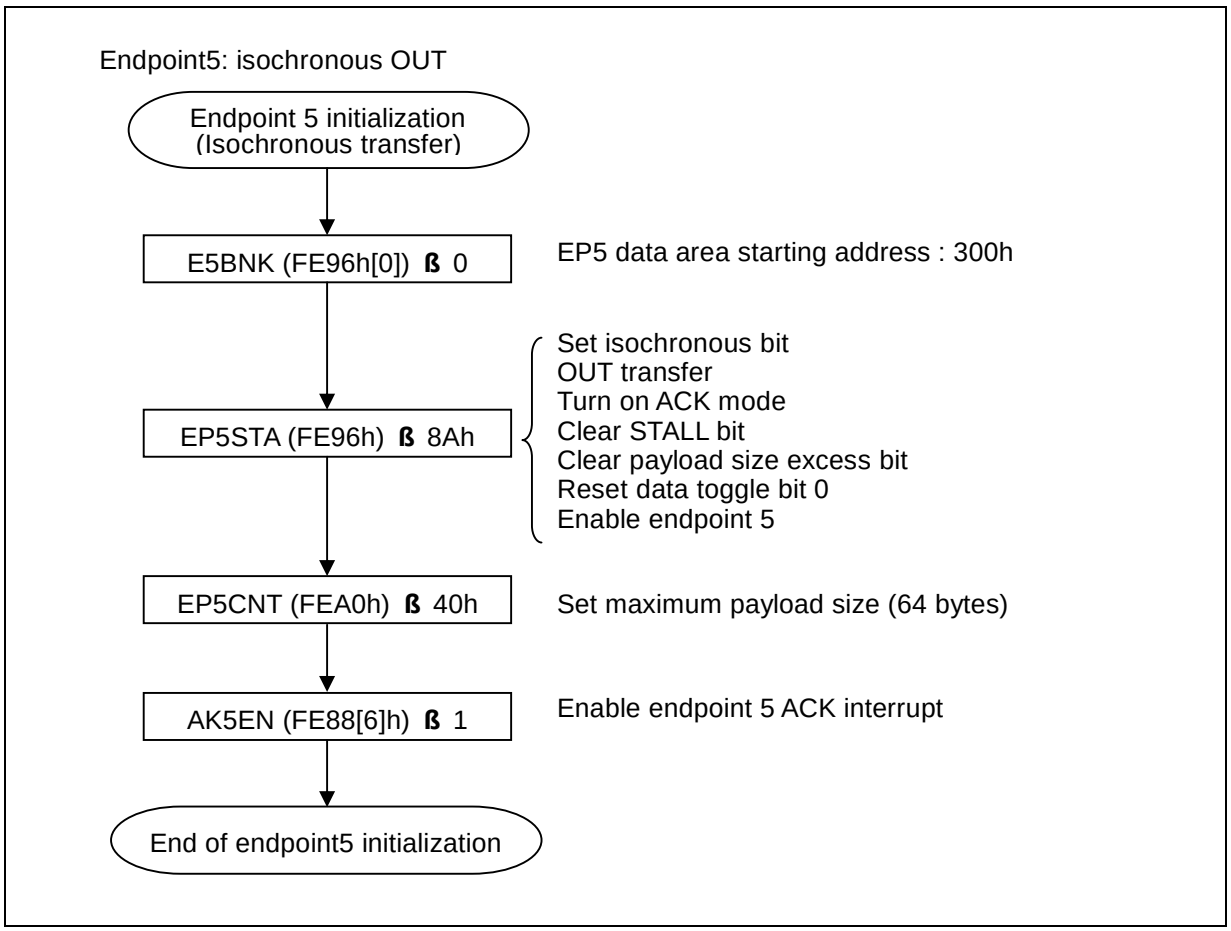


Figure 4-14 Example of Endpoint n Initialization for Isochronous Transfers

The toggle bit (***EnTGL***) in the endpoint n status register has no meaning during isochronous transfers. The PID of the data packet to be transferred during an isochronous IN transfer is always DATA0 regardless of the value of ***EnTGL***. Since the LC87F1M16A makes no toggle check during isochronous OUT transfers, data packets with a PID of DATA0 or DATA1 are received normally and the data areas are updated regardless of the value of ***EnTGL***.

4.8.3. Endpoint Descriptor Example

Endpoints 1-6 can be used in isochronous transfers. Since data is transferred in one direction in isochronous transfers, it is necessary to specify the data direction (transmit or receive) for each endpoint to be used. Specify the endpoint number to be used and the data direction in the endpoint address field (bEndpointAddress) in the endpoint descriptor and an integer value 0-1023 in the maximum packet size field (bMaxPacketSize). Also specify an integer value 1 in the required maximum period field (bInterval). Specifying 1 guarantees that periodic polls occur every 1 millisecond. (Refer to 9.6.4, "Endpoint", of USB 2.0 Specification.)

Field	Size (in Bytes)	Value	Description
bLength	1	07h	Descriptor size
bDescriptorType	1	05h	ENDPOINT descriptor
bEndpointAddress	1	(One of the values listed below) 01h 02h 03h 04h 05h 06h 81h 82h 83h 84h 85h 86h	01h: Endpoint 1 OUT 02h: Endpoint 2 OUT 03h: Endpoint 3 OUT 04h: Endpoint 4 OUT 05h: Endpoint 5 OUT 06h: Endpoint 6 OUT 81h: Endpoint 1 IN 82h: Endpoint 2 IN 83h: Endpoint 3 IN 84h: Endpoint 4 IN 85h: Endpoint 5 IN 86h: Endpoint 6 IN
bmAttributes	1	01h	Isochronous transfer
wMaxPacketSize	2	(One of the values listed below) 0000h-03FFh	The maximum packet size (in bytes) must be set to an integer value between 0 and 1023.
bInterval	1	01h	Polling interval (in milli-seconds)

Table 4-6 Endpoint Descriptor (Isochronous Transfer)

4.8.4. Endpoint Data Areas

In an isochronous transfer, a predetermined amount (specified in the wMaxPacketSize field of the endpoint descriptor) is transferred without fail, once per frame (1ms). Since transfers occur regardless of the device state (no NAK response can be made), the device must always be ready for the next transfer. Accordingly, it is necessary to switch the active bank (bank0, bank1) on each transaction.

See Table 1-2 - Table 1-10 about sizes of each endpoint and address mappings in RAM.

4.9. Isochronous IN Transfers

4.9.1. Isochronous IN Setup

Initialization must be performed before isochronous IN transfer starts.
(see 4.8.2, “Endpoint n Initialization (isochronous transfer)”).

The preparatory steps shown below are required to send data during the IN transactions in an isochronous transfer.

- Writing data
  Write the data to be transmitted first into the endpoint n transmit data area (plane A).

- Specifying the bank/address
  Set up ***EnBNK*** to specify the bank and address of the area to be used for transmission. The area specified here needs to contain the data to be transmitted in the next IN transaction. (plane A is specified here).

- Turning on the ACK mode
  Set the ACK bit in **EPnSTA** (***EnACK***=1).

- Writing data
  Write the subsequent transmit data into the other area (plane B). (Switch between plan A and plane B on each execution of transmission preparation processing.)

When the newly data to be transmitted is present, the isochronous IN transfer is started by performing the preparation processing above. The above processing is performed on each normal termination of the transaction so that the IN transaction is performed continuously. This application note assumes that this processing is accomplished by Isochronous IN processing (see Section 4.9.3.).

4.9.2. Isochronous IN Operation

The isochronous IN transfer operation proceeds according to the settings of the status register **EPnSTA** and size register **EPnCNT** of the endpoint n used. If the endpoint n is configured in the ACK mode (***EnACK***=1), the reception of an IN token is followed by the transmission of the data stored in the endpoint n data area in the form of a data packet. After the transmission processing, the endpoint n is automatically placed into the NAK mode (***EnACK***=0). Since no handshake packet is involved in isochronous transfers, the LC87F1M16A will transmit a data packet with a length of 0 byte when it receives an IN token in the NAK mode. The Isochronous IN operation flow is shown in Figure 4-15.

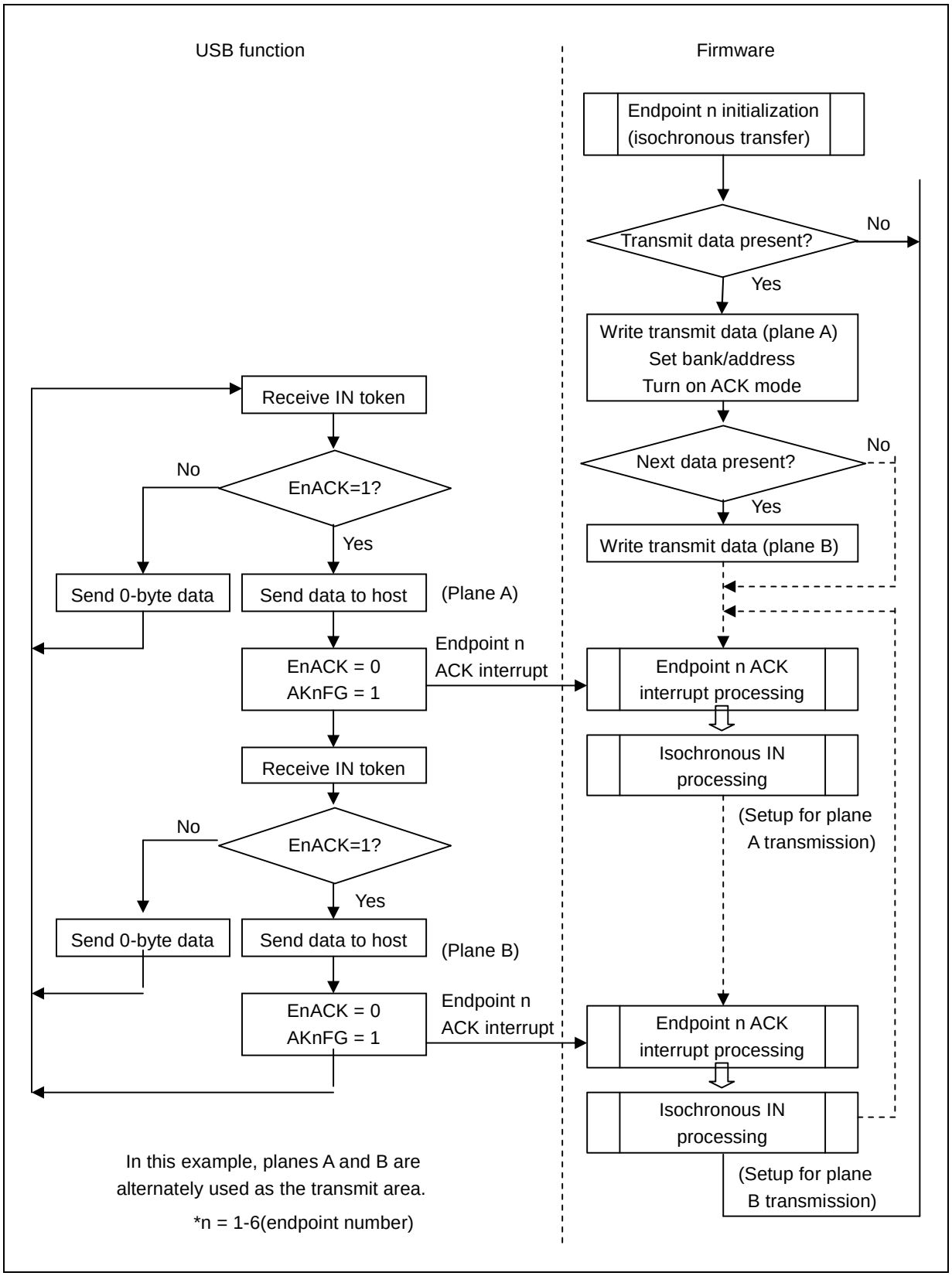


Figure 4-15 Isochronous IN Operation Flow

EnACK is cleared to 0 when an endpoint n ACK interrupt occurs. The LC87F1M16A transmits a data packet with a length of 0 byte when it receives an IN token in this state. This action, however, will not set **AKnFG** (no ACK interrupt is generated).

4.9.3. Isochronous IN Processing

The endpoint n ACK interrupt flag (***AKnFG***) of the endpoint n interrupt register **EPnINT** is set and an endpoint n ACK interrupt is generated when an isochronous transfer IN transaction terminates normally. The following actions are automatically taken when an endpoint n ACK interrupt occurs:

- Turn on the NAK mode (***EnACK***=0).

The endpoint n ACK interrupt processing routine must perform the following isochronous IN processing:

- Take preparatory actions for the next transmission.

The isochronous IN processing to be performed during the endpoint n ACK interrupt processing routine looks like as shown below (see 2.5.1, “Endpoint n ACK Interrupts”).

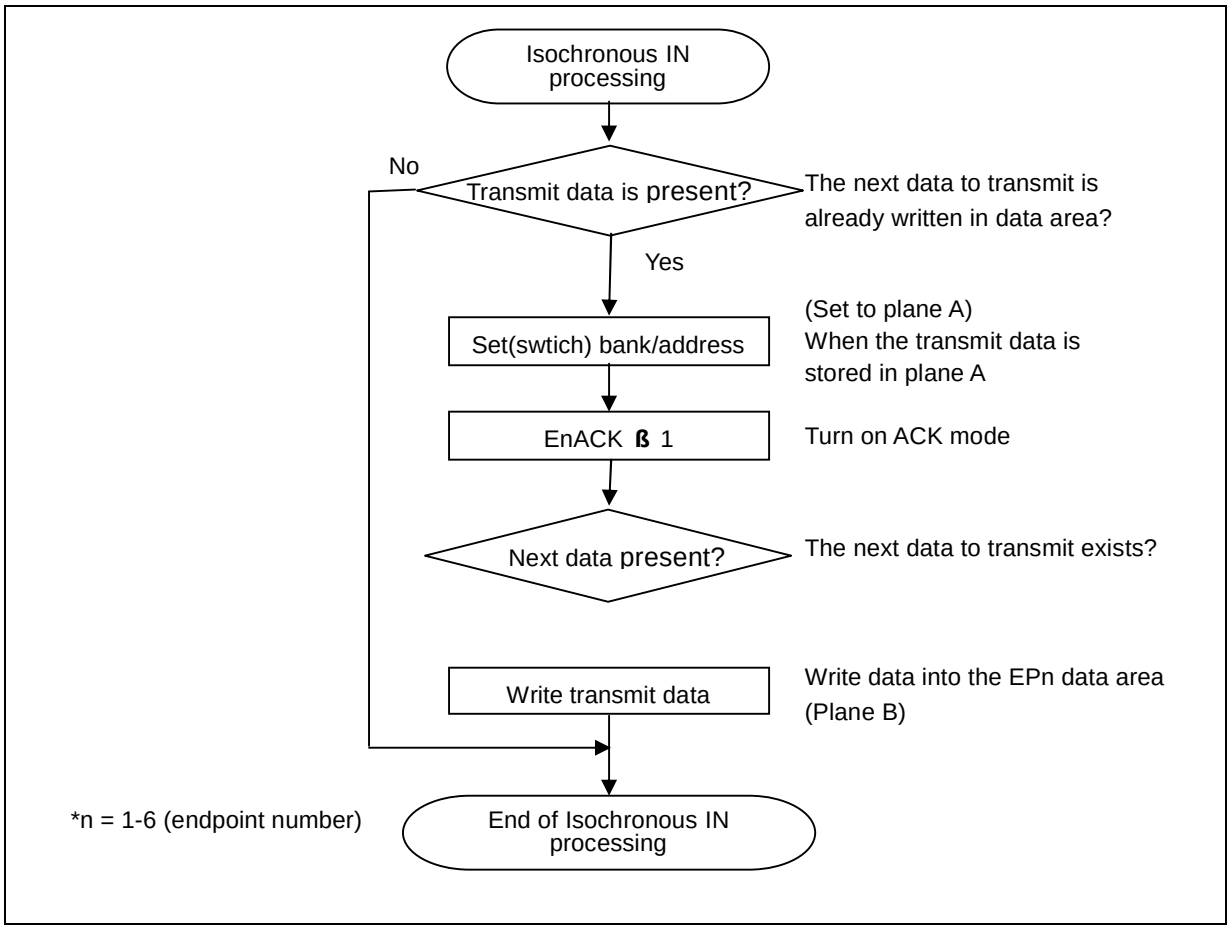


Figure 4-16 Example of Isochronous IN Processing

4.10. Isochronous OUT Transfers

4.10.1. Isochronous OUT Setup

Initialization must be performed before isochronous OUT transfer starts.
(see 4.8.2, “Endpoint n Initialization (isochronous transfer)”).

The preparatory steps shown below are required to receive data during the OUT transactions in an isochronous transfer.

- Specifying the bank/address
  Set up ***EnBNK*** to specify the bank of the area to be used for receiving data. The area specified here is loaded with the data that will be received in the next OUT transaction.

- Turning on the ACK mode
  Set the ACK bit in **EPnSTA** (***EnACK***=1).

The isochronous OUT transfer is executed by performing the preparation processing above. This application note assumes that this processing is accomplished by endpoint n initialization (see Section 4.8.2.) and isochronous OUT processing (see Section 4.10.3.).

4.10.2. Isochronous OUT Operation

The operation of the isochronous OUT transfer proceeds according to the settings of the endpoint n status register **EPnSTA**. If the endpoint n is configured in the ACK (***EnACK***=1), the reception of an OUT token is followed by the reception of a data packet and the received data is stored in the endpoint n data area. When an ACK is transmitted after the receive processing, the endpoint n is automatically configured into the NAK mode (***EnACK***=0). Since no handshake packet is involved in isochronous transfers, the LC87F1M16A will never respond with a NAK when it receives an OUT token in the NAK mode. The Isochronous OUT operation flow is shown in Figure 4-17.

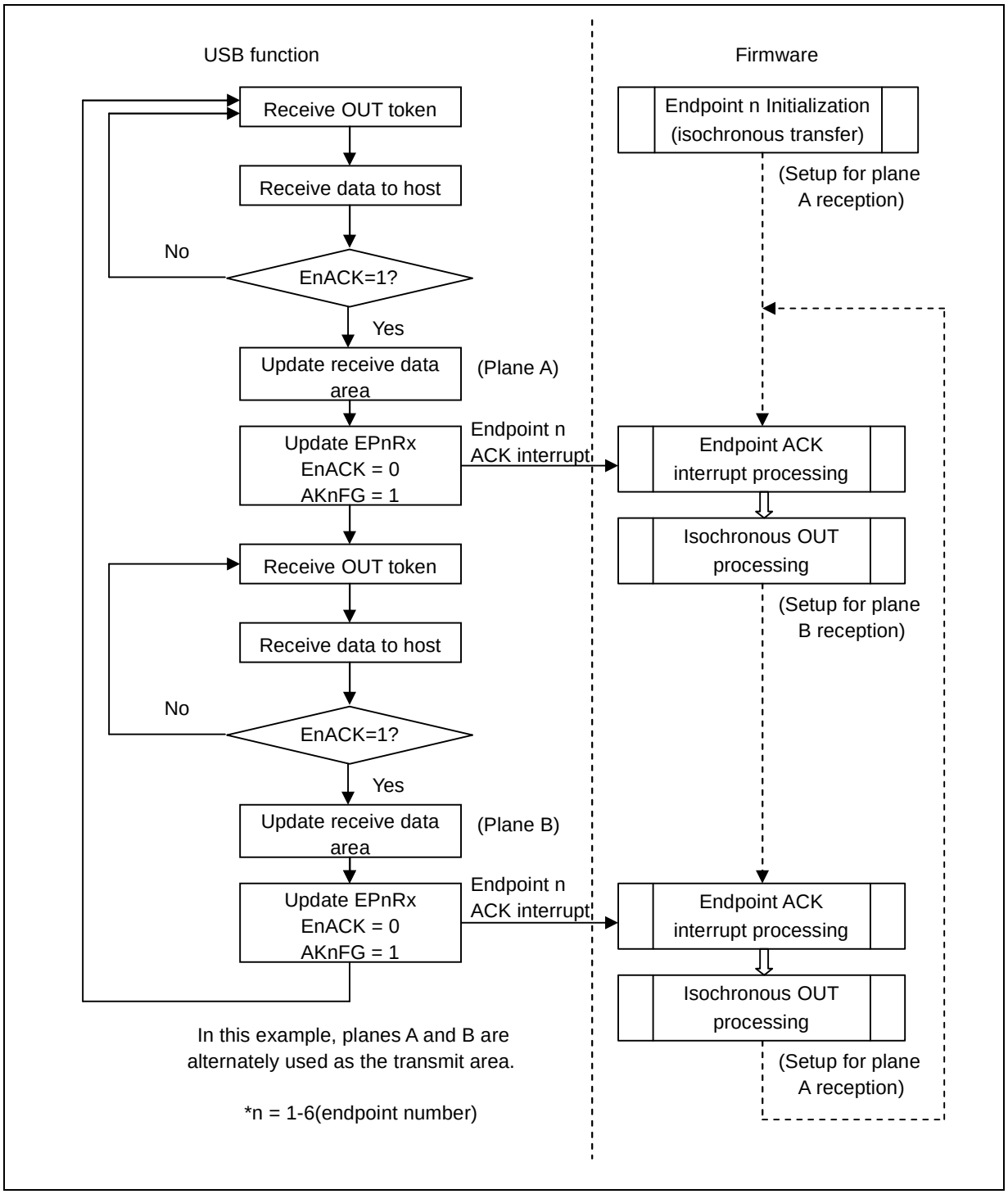


Figure 4-17 Isochronous OUT Operation Flow

4.10.3. Isochronous OUT Processing

When the LC87F1M16A transmits an ACK handshake packet after receiving data in an OUT transaction of an isochronous transfer, the endpoint n ACK interrupt flag (***AKnFG***) of the endpoint n interrupt register **EPnINT** is set and an endpoint n interrupt is generated.
The following actions are automatically taken when an endpoint n ACK interrupt occurs:

- Update the receive data area.
- Update the receive data size (**EPnRX**).
- Turn on the NAK mode (***EnACK***=0).

The endpoint n ACK interrupt processing routine must perform the following isochronous OUT processing:

- Read the receive data.
- Take preparatory actions for the next reception.

The Isochronous OUT processing to be performed during the endpoint n ACK interrupt processing routine looks like as shown below (see 2.5.1, “Endpoint n ACK Interrupts”).

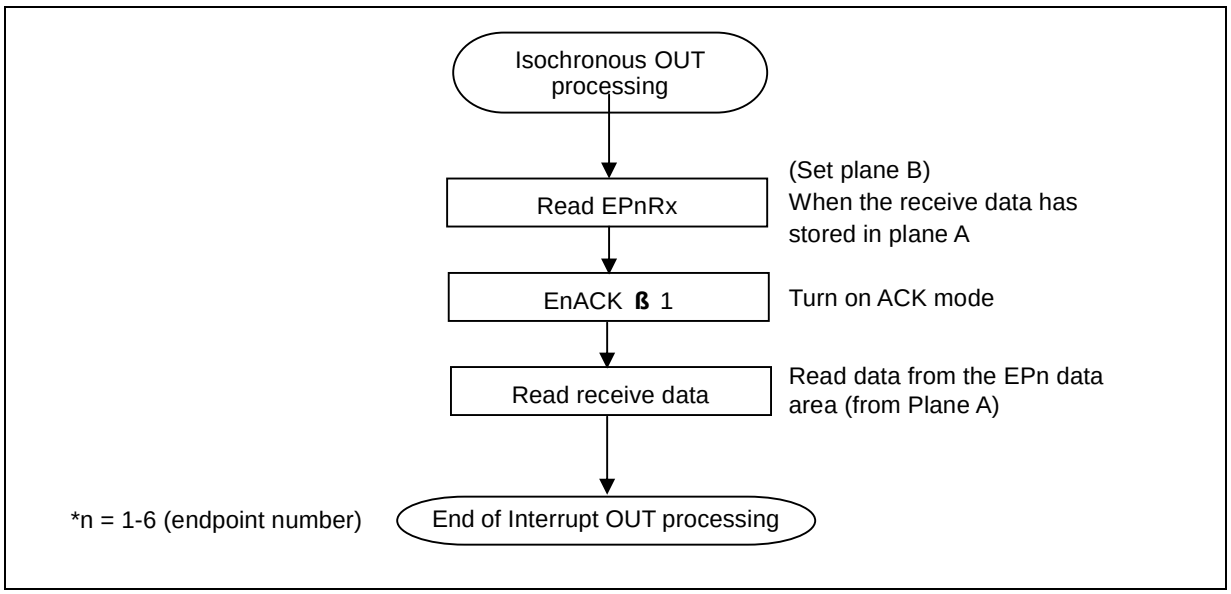


Figure 4-18 Example of Isochronous OUT Processing

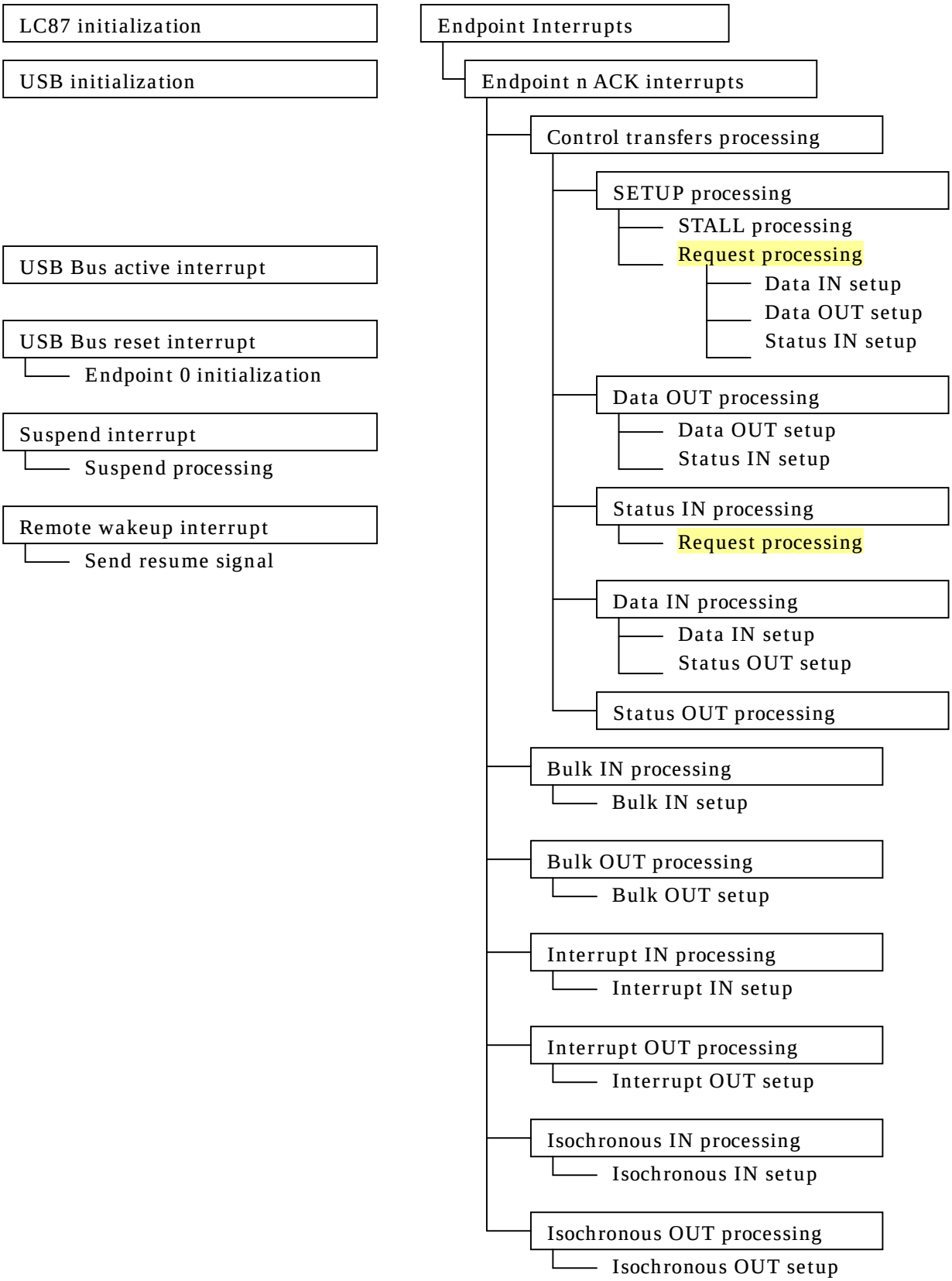
Chapter 5. Appendix

5.1. LC87F1M16A RAM Map

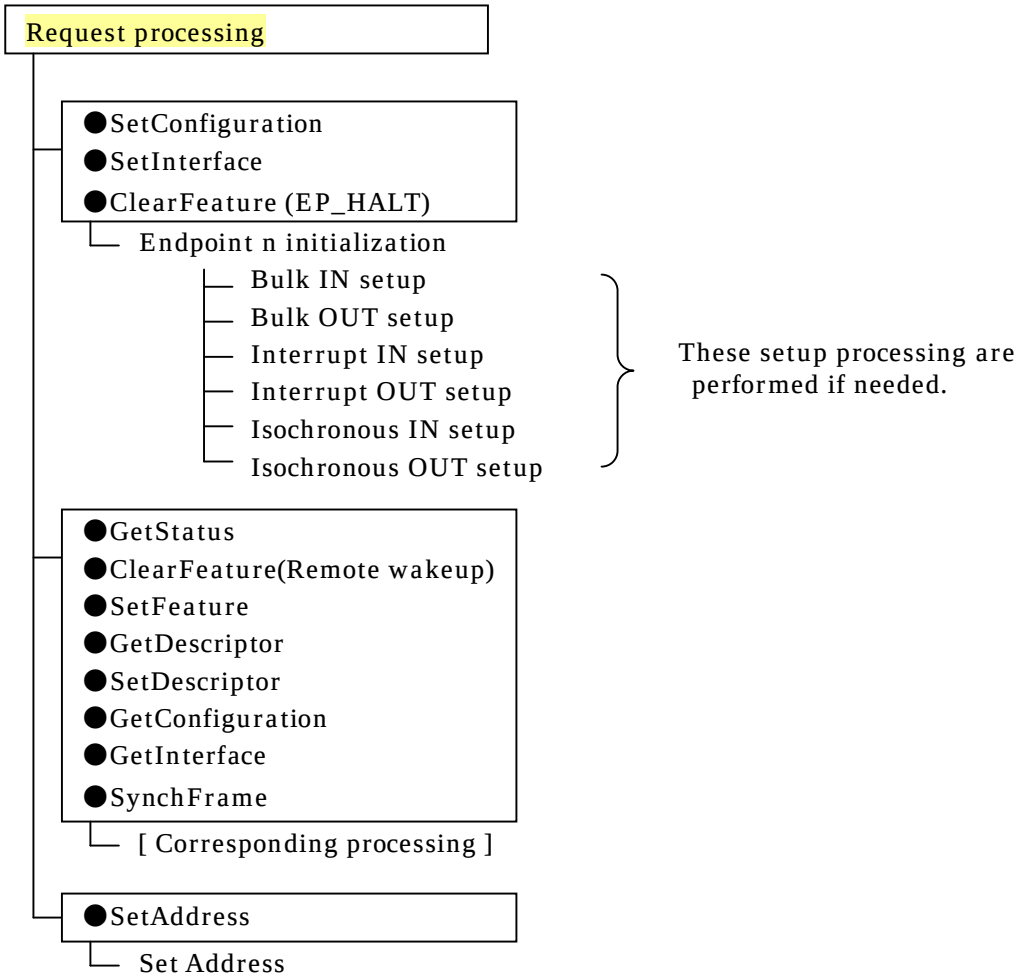
■Internal data memory space is 64K-bytes (Address : 0000h-FFFFh)
■Actually LC87F1M16A has 1024 bytes RAM.

Address		Size	
0000 ～007F	【R0～R63】	128byte	1024byte
0080 ～01FF			
0200 ～033F	EP0 – EP6 Transmit / Receive (Configure EPBMOD register and Bank)	Max 512byte	
0C00 ～FDFF	not exists		
FE00 ～FEFF	SFR	256byte	
FF00 ～FFFF	System	256byte	

5.2. Example of Control Program Configuration



In request processing, proper processing is performed according to the request which received on the SETUP stage of Control transfer. A device starts request processing after a SETUP stage. However, a SetAddress request must be performed after Status stage completion. The outline of request processing is shown below.



When SetConfiguration, SetInterface, or ClearFeature(ENDPOINT_HALT) request is received, it is necessary to initialize the related endpoint n(n=1-6). In initialization, reset the data toggle bit to 0 and enable the endpoint. Furthermore, if the endpoint can start transfers, proper setup processing can be performed.