

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

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

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

April 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

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

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

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
  - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
  - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
  - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

# M16C/6C Group

## USB Sample Software

### Introduction

This application note explains how to write the low-level software for Universal Serial Bus (USB) operation on the M16C/6C microcontroller.

The USB module is based on the USB Specification revision 2.0. It supports Full Speed operation.

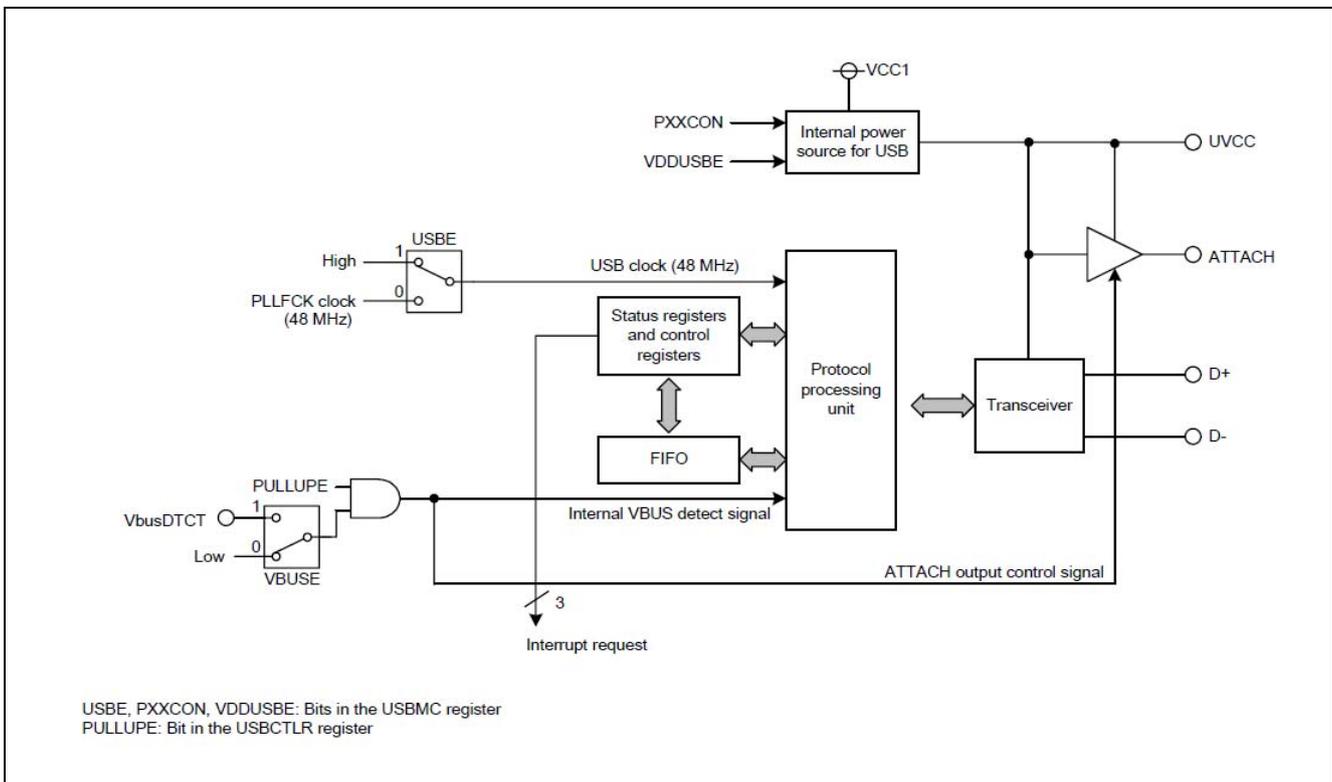
There are seven endpoints: Control IN/OUT, Bulk IN × 2, Bulk OUT × 2 and Interrupt IN × 2.

Control, Bulk IN, Bulk OUT and Interrupt IN transfer modes are supported.

Each endpoint has its own FIFO buffer as specified below.

**Table 1: Endpoint configuration**

Endpoint	Transfer Type	Maximum Packet Size (bytes)	FIFO Buffer Capacity (bytes)	DMA Transfer
Endpoint 0	Setup	8	8	—
	Control IN	16	16	—
	Control OUT	16	16	—
Endpoint 1	Bulk OUT	64	64 X 2	Available
Endpoint 2	Bulk IN	64	64 X 2	Available
Endpoint 3	Interrupt IN	16	16	—
Endpoint 4	Bulk OUT	64	64 X 2	Available
Endpoint 5	Bulk IN	64	64 X 2	Available
Endpoint 6	Interrupt IN	16	16	—



**Figure 1. USB Block Diagram**

## Contents

1. Initialisation.....	3
2. Cable Connection.....	7
3. Enumeration / Control Transfers .....	8
4. Bulk Transfer .....	14
5. Interrupt Transfer.....	17
6. Stall .....	18
7. Suspend / Resume.....	18
8. Cable Disconnection .....	18
9. Reference.....	19

## 1. Initialisation

This section shows how to initialise the M16C/6C so that it is ready to detect a USB cable connection and be enumerated by a host.

### 1.1 Pins

There are five I/O pins associated with the USB module. Configure these as specified in the following table.

**Table 2: I/O pin assignments**

Pin Name	Function	Configure as
ATTACH	Output used for D+ 1.5kΩ pull-up	Input pin. Set bit PD9_2 = 0.
VbusDTCT	Input the power supply signal from a host	Input pin. Set bit PD8_5 = 0.
D+	USB D+ input/output	Input pin. Set bit PD9_1 = 0.
D-	USB D- input/output	Input pin. Set bit PD9_0 = 0.
UVCC	Either input power supply for pins ATTACH, D+, and D- or if using internal USB power then 3.3V is output.	Dedicated pin, no configuration necessary.

The following code snippet configures the pins. Note that the Port 9 Direction Register is write protected by default.

```
prc2 = 1; /* Enable writes to the PD9 register */
pd9 = 0xf8; /* Set ATTACH(P9_2), D+(P9_1) and D-(P9_0) pins as INPUT */
prc2 = 0; /* Disable writes to the PD9 register */

pd8_5 = 0; /* Set VbusDTCT pin as an INPUT */
```

### 1.2 Module Enable

The USB module must be enabled, by clearing the USBE bit in the USBMC register to 0, before any of the other USB registers can be accessed.

```
/* Enable USB module */
usbe_usbmc = 0;
```

### 1.3 Clocks

The USB module requires that the PLLFCK clock is configured to be 48 MHz. The following code sample does this assuming a 6 MHz main crystal frequency.

```
/* Set PLLFCK generation enabled */
plc06 = 1;
/* Set *8 and /3 therefore: */
/* PLL = 6*8*2/3 = 32MHz */
/* PLLFCK = 6*8 = 48MHz */
plcf = 0x04;

/* Enable PLL */
plc07 = 1;
```

See Figure 2. USB Initial Settings on page 5.

## 1.4 Power

If the voltage supplied to VCC1 is at least 4V then the required 3.3V D+/D- reference voltage can be produced internally by setting the VDDUSB and PXXCON bits in the USBMC register. Note: This option requires the 125 KHz on-chip oscillator to be operating.

```

prc0 = 1;           /* Enable access to cml register */
cml4 = 0;           /* Start 125 KHz oscillator */
prc0 = 0;           /* Write protect cml registers */

vddusbe_usbmc = 1; /* 3.3V USB Internal power supplied */
pxxcon_usbmc = 1;  /* Enable VDDUSB bit */

```

If VCC1 is less than 4V the internal 3.3V power source cannot be used. A 3.3V supply must be connected to pin UVCC, with the VDDUSB bit set to 0 and the PXXCON bit set to 1.

If powering the device from the USB VBus line (5V) then set the PWMD bit in the USBCTLR register to 1. The value of this bit is used by the M16C/6C when it automatically responds to the standard USB command 'GetStatus'.

Note that before the USBCTLR register can be accessed the USB module must be enabled by setting the USBE bit in the USBMC register to 0.

```

/* Bus powered mode */
pwmd_usbctlr = 1;

```

See Figure 2. USB Initial Settings on page 5.

## 1.5 Interrupt Vectors

There are three vectors relating to the USB module.

- Vector 54 USB Function 0
- Vector 55 USB Function 1
- Vector 56 USB Resume

Other than the Resume Interrupt, all other interrupts can be routed through either Vector USB Function 0 or 1 depending upon the setting of the USB Interrupt Select Registers (USBISR0-3). The decision as to which interrupt should use which vector is made purely on the grounds of increasing speed by reducing the number of status flags that need to be checked in a particular interrupt handler in order to ascertain what caused the interrupt.

The USB Resume interrupt only needs to be handled if the USB module is ever going to be actually disabled as part of a low-power mode suspended state. Otherwise a suspended state is exited through the SURSF interrupt which can be routed through either Vector USB Function 0 or 1.

Generally the USB interrupt handler needs to check the USB Interrupt Flag Registers (USBIFR0-3) for the interrupts that it has been set up to handle and act accordingly, including clearing the particular flag. The handling of particular interrupts shall be described in the section of this document relating to it. So for example the handling of the VBUS interrupt is described in Section 2. Cable Connection.

1.6 USB Module

The following flow diagram shows the procedure for USB initialisation.

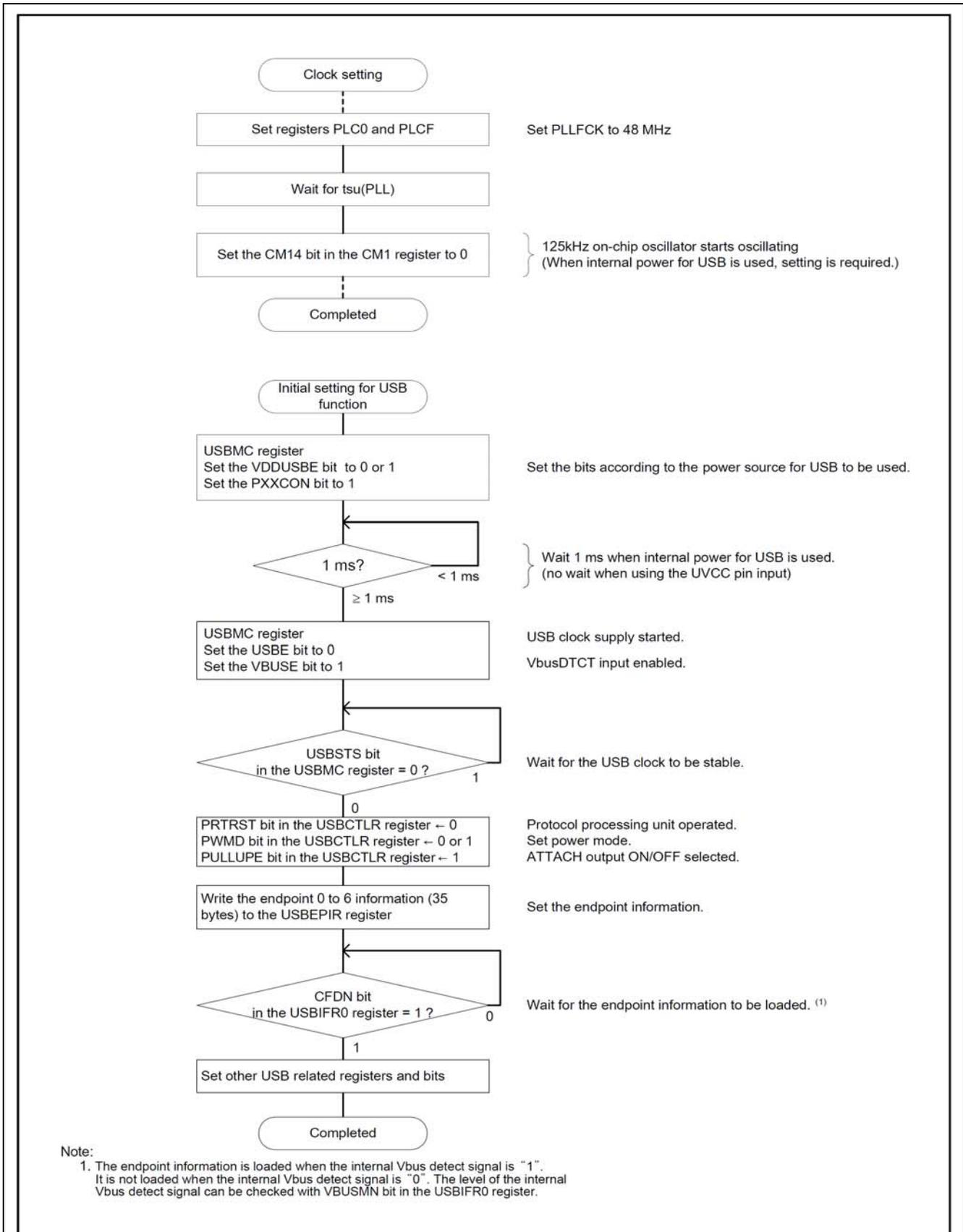


Figure 2. USB Initial Settings

A code sample of setting the Endpoint Information can be seen in Reference 9.1 USB Endpoint Information Loading on page 19.

In the above flow diagram, interrupts can be configured at the stage labelled “Set other USB related registers required”. The following code snippet shows the setting up of the interrupt priorities and the routing of all interrupts via interrupt 0 by the setting of the USB Interrupt Select Register:

```
/* Set USB interrupt priority */
usbint0ic = 0x03;
usbint1ic = 0x03;
usbrsmic = 0x03;

/* Use only USB Interrupt 0 */
usbisr0 = 0x00;
usbisr1 = 0x00;
usbisr2 = 0x00;
```

For the Cable Connection and Enumeration stages to operate as specified in the following sections the VBUSFE, BRSTE and SETUPTSE interrupts should be enabled. The handling of these interrupts will be explained in the following sections.

```
/* Enable Interrupts required for Connection and Enumeration */
brste_usbier0 = 1;
setuptse_usbier1 = 1;
vbusfe_usbier0 = 1;
```

## 2. Cable Connection

If the setup of the device has been completed as specified in the Initialisation section of this document then, when the USB cable is connected, the USB Bus Connection/Disconnection flag (VBUSF) will be set and a VBUSF interrupt generated. In the handler of this interrupt, the Internal VBUS Signal Monitor flag (VBUSMN) in the USB Interrupt Flag register 0 (USBIFR0) can be checked to see if this is a connection (VBUSMN = 1) or a disconnection (VBUSMN = 0) event. Clear the VBUSF flag and then wait for the USB Bus Reset Detect Flag (BRST) bit to cause a BRST interrupt.

The BRST interrupt should be handled by clearing the BRST flag and by clearing all Endpoint FIFOs by setting the relevant bits in the USBFCLR registers. Note that these registers are not readable so write to the register as a whole rather than doing a read-modify-write operation to access individual bits.

This code snippet clears all FIFO buffers on all endpoints:

```
usbfclr0 = 0x03; /* EP0oCLR and EP0iCLR */
usbfclr1 = 0x07; /* EP1CLR, EP2CLR & EP3CLR */
usbfclr2 = 0x07; /* EP4CLR, EP5CLR & EP6CLR */
```

If using the Endpoint Stall mechanism (See section 6. Stall) it is also a good idea to clear any possible stall conditions by setting the relevant bits in the USBEPSTL registers.

This code snippet clears all stall conditions on all endpoints:

```
usbepstl0 = 0x10; /* EP0 Stall Clear */
usbepstl1 = 0x70; /* EP 1, 2 & 3 Stall Clear */
usbepstl2 = 0x70; /* EP 4, 5 & 6 Stall Clear */
```

The following flowchart details this operation:-

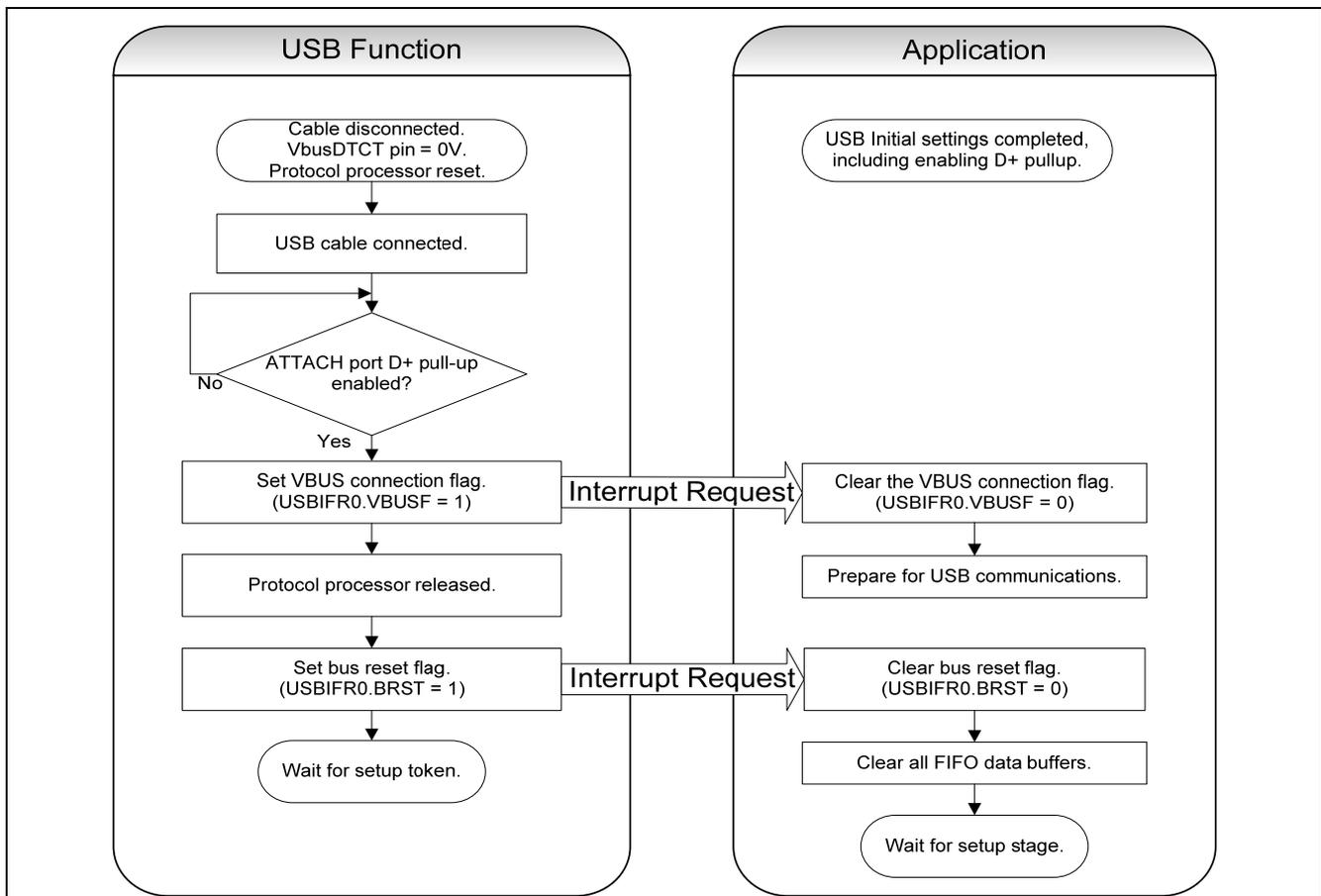


Figure 3. Cable Connection

### 3. Enumeration / Control Transfers

Following the USB Bus Connection/Disconnection Detect (VBUSF) and USB Bus Reset Detect (BRST) interrupts, the process of Enumeration will start with the reception of a Setup command.

Enumeration is the process of the host interrogating the device to discover what it is and what capabilities it has. The host does this by using Control Transfers. A Control transfer consists of the following stages:

- **Setup.** The host sends an 8-byte setup command of a fixed format.
- **Data.** Depending upon the command there may be a data stage in either direction IN or OUT.
- **Status.** This reports success or failure of the command.

#### 3.1 Setup Stage

In the setup stage, command data from the host is analysed to determine the subsequent processing required such as if a data stage is required and what direction it is in.

When a Setup command is received the Setup Command Receive Complete flag (SETUPTS) will be set causing a SETUPTS interrupt. Clear the SETUPTS flag.

Because it is possible for the host to interrupt a previous control transfer by sending a new setup command, both the IN and OUT FIFOs on the control endpoint should be cleared. The 8-byte setup command should be read, one byte at a time, from the USP Endpoint 0 Status Data register (USBEPDR0S) and then the Endpoint 0S Read Complete Trigger bit (EP0SRDFN) set to 1.

The following code snippet performs this operation:

```

/* Clear the Interrupt Flag */
/* Set setupts bit to 0. (Register must be accessed as 8bit)*/
usbifrl = (unsigned char)~BIT_3;

/*Clear EP0i and EP0o FIFOs*/
/* Register is not readable - so don't use bit definitions.*/
usbfclr0 = (unsigned char)(BIT_0 | BIT_1); /* EP0oCLR = 1 and EP0iCLR = 1 */

/* Read 8 bytes of data from EP0 into temporary buffer */
for(index = 0; index < USB_SETUP_PACKET_SIZE; index++)
{
    SetupCmdBuffer[index] = (UINT8)usbepdr0s;
}

/*Set EP0 Status Read Complete*/
/*Set EP0SRDFN bit*/
usbtrg0 = BIT_2;

```

The setup commands that need to be supported depend upon the USB class being implemented but for basic enumeration only the standard Get Descriptor request needs to be handled as the M16C/6C automatically handles the following standard requests:

- Get Configuration
- Get Interface
- Get Status
- Set Address
- Set Feature
- Clear Feature

Note: If the Setup command is not supported or has not been understood the Control Endpoint (Endpoint 0) must be stalled. See Section 6. Stall.

The following flowchart details this setup operation:-

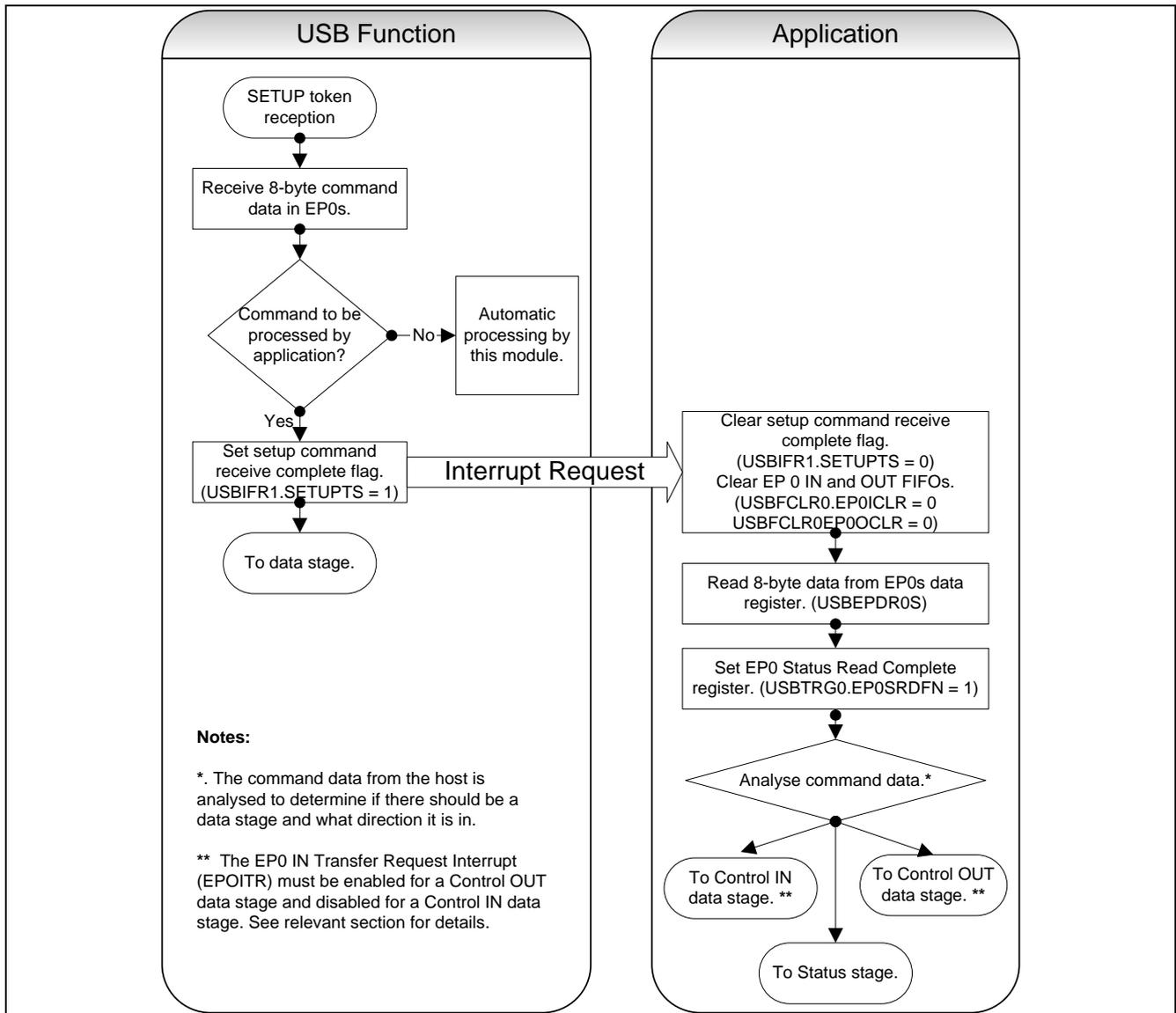


Figure 4. Setup Stage

### 3.2 Data Stage

The setup command includes a field (wLength) specifying the length of the data stage. There may be no data stage (wLength set to 0) in which case skip straight to the Status Stage. If there is a data stage then another field in the setup command (bmRequestType) says if the direction of the data flow is IN or OUT.

### 3.2.1 Control IN

A Control IN data stage is used to send data to the host on the control pipe in response to a setup command that requests it. Note that this is how descriptors will be passed to the host in response to the Get Descriptor standard request.

Enable interrupts Endpoint 0 IN Transmit Complete Flag (EP0iTS) and Endpoint 0 OUT Receive Complete Flag (EP0oTS). Ensure Endpoint 0 IN Transfer Request Flag (EP0iTR) is disabled.

Write the data a byte at a time to the USBEPDR0I register until all the data has been sent or until the maximum packet size of 16 bytes has been written. Set the Endpoint 0 IN Packet Enable Trigger bit (EP0IPKTE) to send the data and then wait for an EP0iTS or EP0oTS interrupt.

Handle an EP0iTS interrupt by sending up to another packets worth of data. If there is no more data to send then send a zero length packet by setting EP0IPKTE without first writing any data to USBEPDR0I.

An EP0oTS interrupt signals the end of the data stage as this is the host sending the status stage. See Section 3.3, 'Status Stage'.

Note that some hosts do not wait to receive all the data that they have requested before moving on to the Status Stage. In this case abandon the remainder of the data stage.

The following flowchart details this operation:-

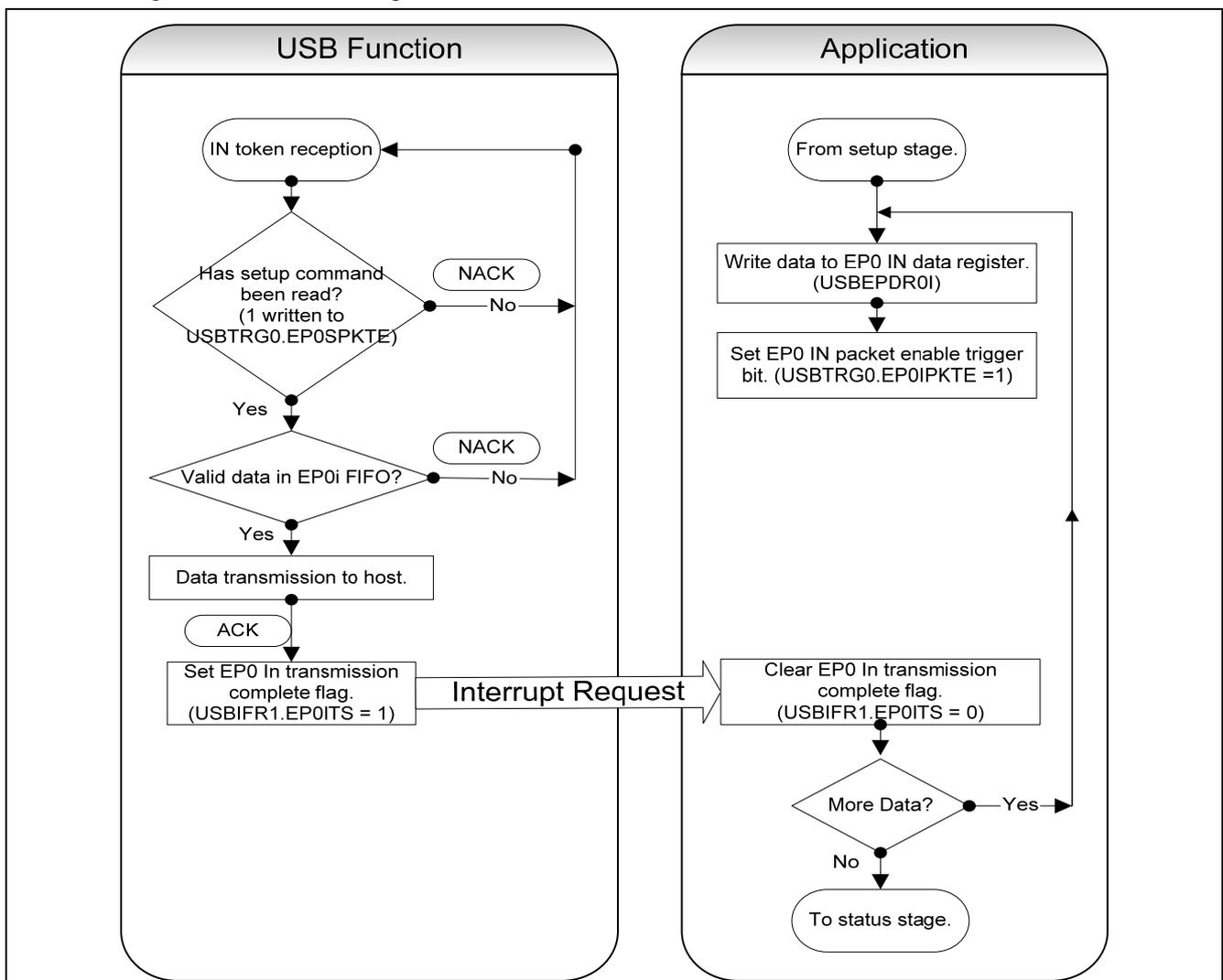


Figure 5. Data Stage – Control IN.

### 3.2.2 Control OUT

A Control OUT data stage is used to receive data from the host on the control pipe following a setup command that requires it. Note that this is not required for basic enumeration of standard requests but there are some class requests that do require it such as the Communication Device Class (CDC/ACM) which uses the ‘SET LINE CODING’ request and the Human Interface Device (HID) class that uses a ‘SET REPORT’ request.

Enable EP0iTS, EP0oTS and EP0iTR interrupts.

When data is received from the host an EP0oTS interrupt will be generated. In the handler for this read the number of bytes (as indicated in the EPSZ0o register), a byte at a time, from the Endpoint 0 OUT Data register (EPDR0o). Clear the interrupt flag and set the read complete bit EP0oRDFN. This process repeats for each packet of data the host sends. When the host has sent all the data it intends to it will try and read the status stage. This will result in an EPOiTR interrupt being generated. See Section 3.3 ‘Status Stage’.

The following flowchart details this operation:-

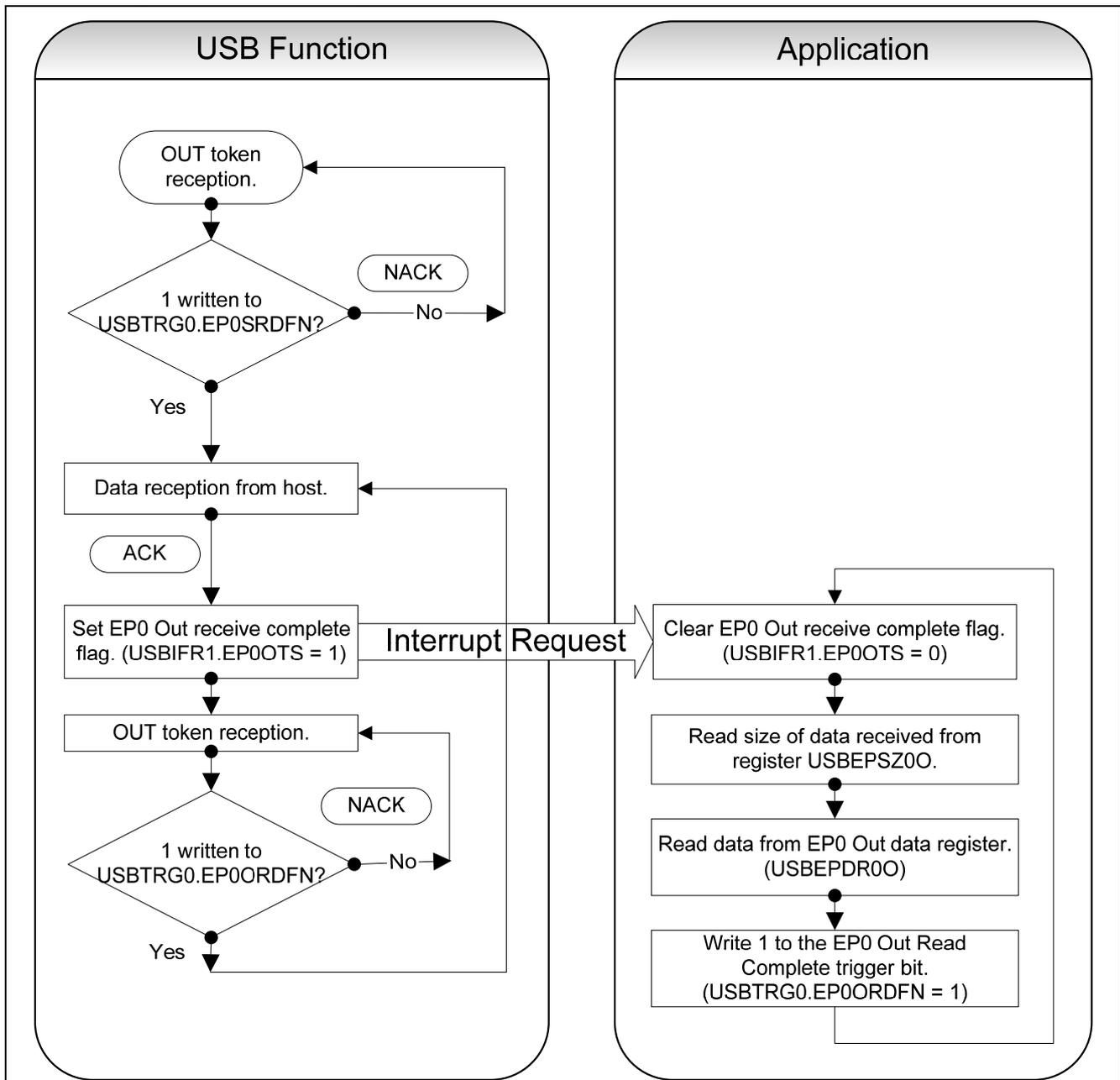


Figure 6. Data Stage – Control OUT.

### 3.3 Status Stage

#### 3.3.1 Sending

If there is not a data stage for this Control Transfer or a Data OUT stage has completed (EP0iTR interrupt received) then the device is the source of the status stage. Cause an ACK to be sent to the host by sending a zero length IN packet. Do this by setting EP0IPKTE bit without first writing any data to USBEPDR0I.

```
/* Transmit 0-byte data to host (ACK):- Packet Enable - Set EP0IPKTE bit*/
usbtrg0 = BIT_0;
```

Then handle the resulting EP0 IN Transmission Complete interrupt (EPOiTS) by clearing the interrupt flag.

The following flowchart details this operation:-

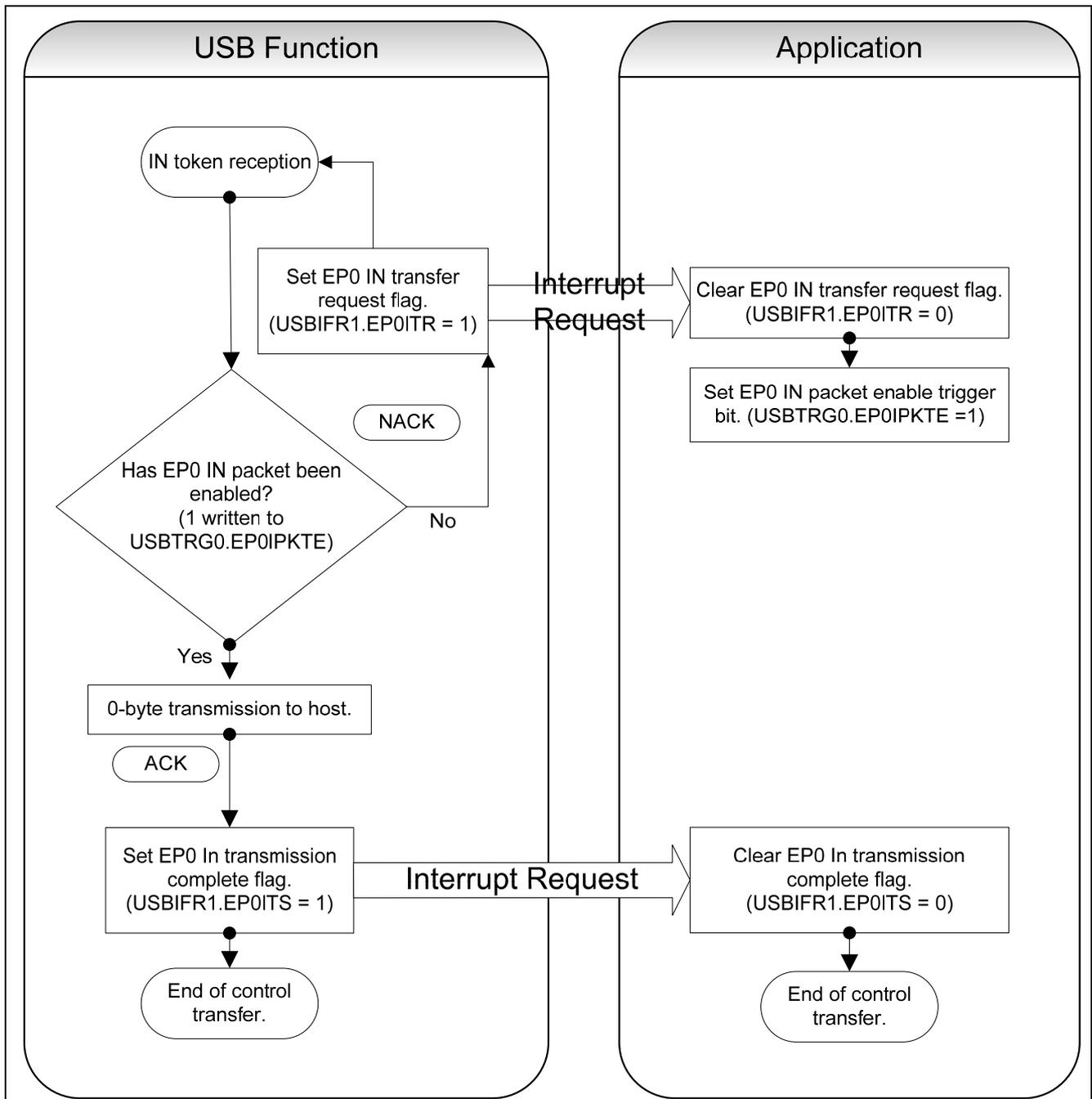


Figure 7. Status Stage (Control OUT)

### 3.3.2 Receiving

The host is the source of the status stage if there has been a Data IN stage. An EP0oTS interrupt will occur when the ACK is received from the host. Clear the EP0oTS interrupt flag and set the EP0 OUT Read complete trigger bit (EP0oRDFN).

```

/* Clear the Interrupt Flag */
/* Set EP0oTS bit to 0. (Register must be accessed as 8bit)*/
usbifr1 = (unsigned char)~BIT_2;

/* EP0o Read complete */
/*Set EP0ORDFN bit*/
usbtrg0 = BIT_1;

```

The following flowchart details this operation:-

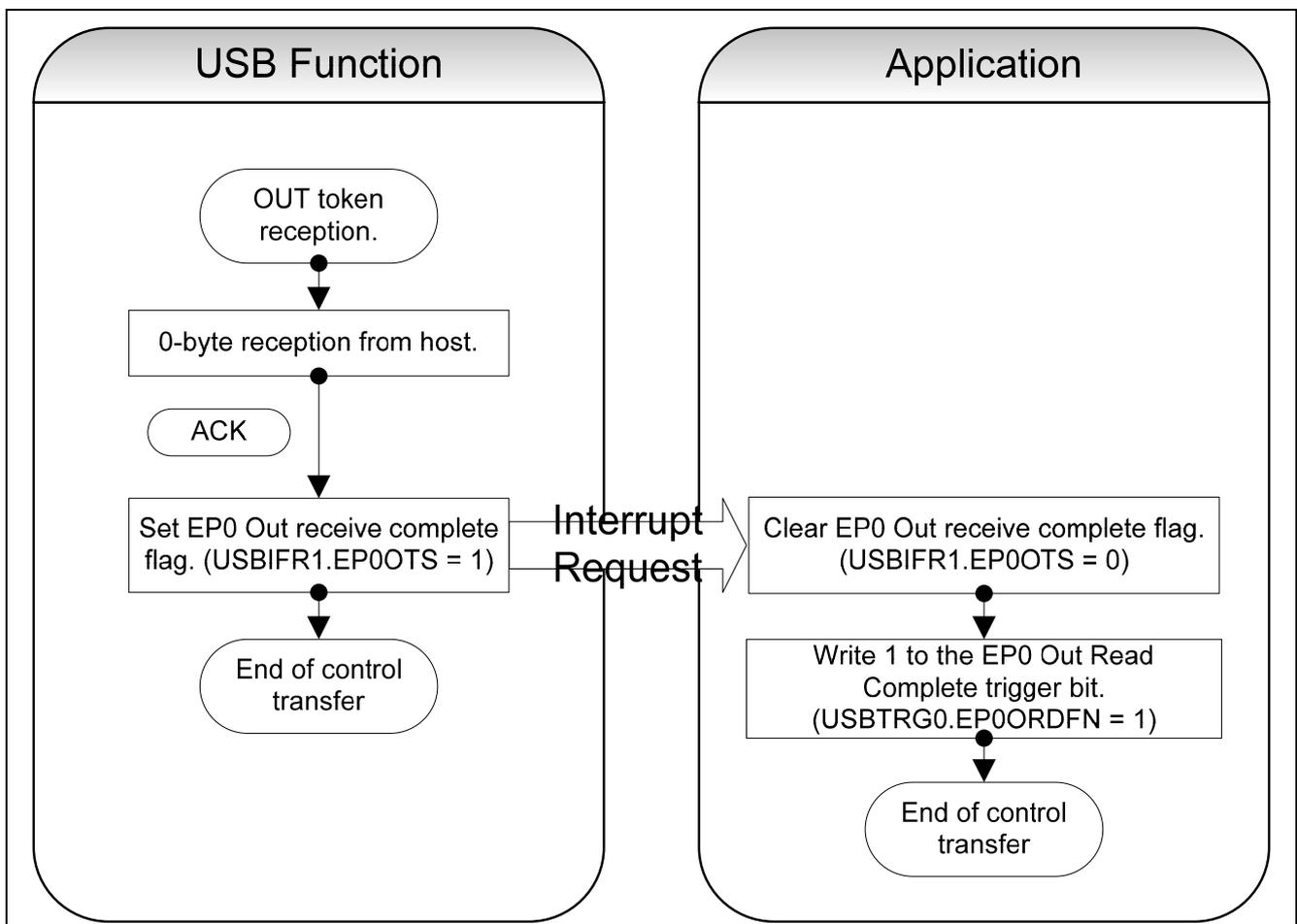


Figure 8. Status Stage (Control IN).

## 4. Bulk Transfer

The M16C/6C supports both Bulk IN and Bulk OUT.

### 4.1 Bulk IN

The M16C/6C has two identical Bulk IN endpoints, Endpoint 2 and Endpoint 5. The following description assumes that Endpoint 2 is being used but to change this to use Endpoint 5 it is simply a matter of using EP5 rather than EP2 related registers.

If you need to know when the host is requesting a Bulk IN transfer, then enable the Endpoint 2 Transfer Request (EP2TR) interrupt:

```
ep2tre_usbier2 = 1;
```

In the interrupt handler for this, or if not using the EP2TR interrupt then just at the point where you want to start a Bulk IN transfer, enable the Endpoint 2 FIFO Empty (EP2EMPTY) interrupt:

```
ep2emptye_usbier2 = 1;
```

In the EP2EMPTY interrupt handler write the data, one byte at a time, to the Endpoint 2 Data Register (USBEPDR2) until all the data has been sent or until the maximum packet size of 64 bytes has been written.

Set the Endpoint 2 Packet Enable Trigger bit (EP2PKTE) to send the data.

Repeat this handling of the EP2EMPTY interrupt until there is no more data to send. In some cases the host knows how much data the device will be sending but usually a transfer will need to be completed by ensuring that a non-complete packet of data has been sent. If the amount of data is equal to an integer number of full packets then in order to end the transfer on a non-complete packet it is necessary to send an empty packet by setting the EP2PKTE bit without first having written any data to USBEPDR2. The EP2EMPTY interrupt can then be disabled.

The following flowchart details this operation:-

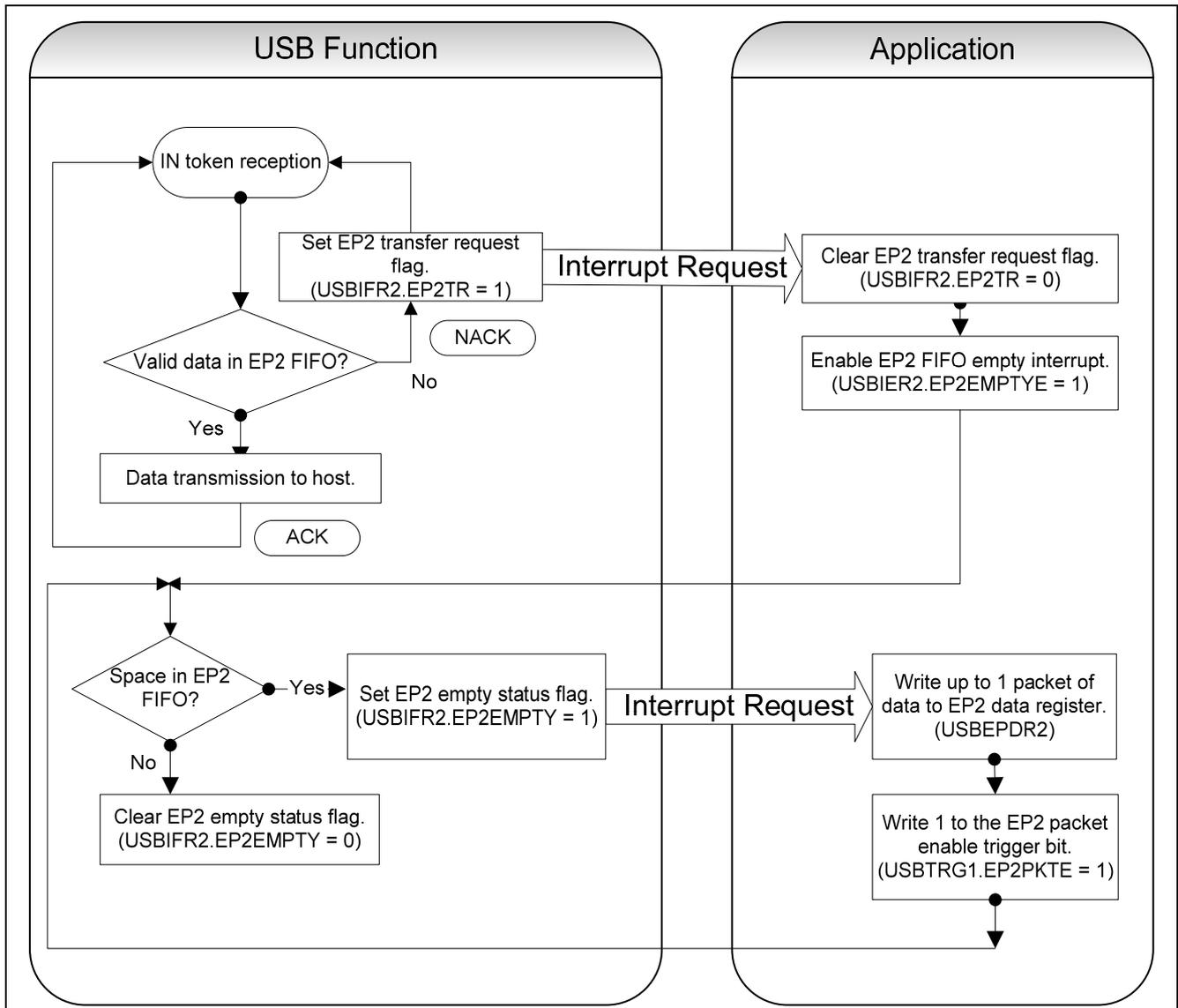


Figure 9. EP2 Bulk IN Transfer.

## 4.2 Bulk OUT

The M16C/6C has two identical Bulk OUT endpoints, EP1 and EP4. The following assumes EP1 is being used.

Enable the Endpoint 1 FIFO Full (EP1FULL) interrupt:

```
ep2tre_usbier2 = 1;
```

When an EP1FULL interrupt occurs it means that the host has sent some data. The number of bytes received can be read from the USB Endpoint 1 Receive Data Size register (USBEPSZ1). Read each byte, one byte at a time, from the USB Endpoint 1 Data register (USBEPDR1) and then set the read complete bit.

```
/*Read Complete*/
eplrdfn_usbtrg1 = 1;
```

The following flowchart details this operation:-

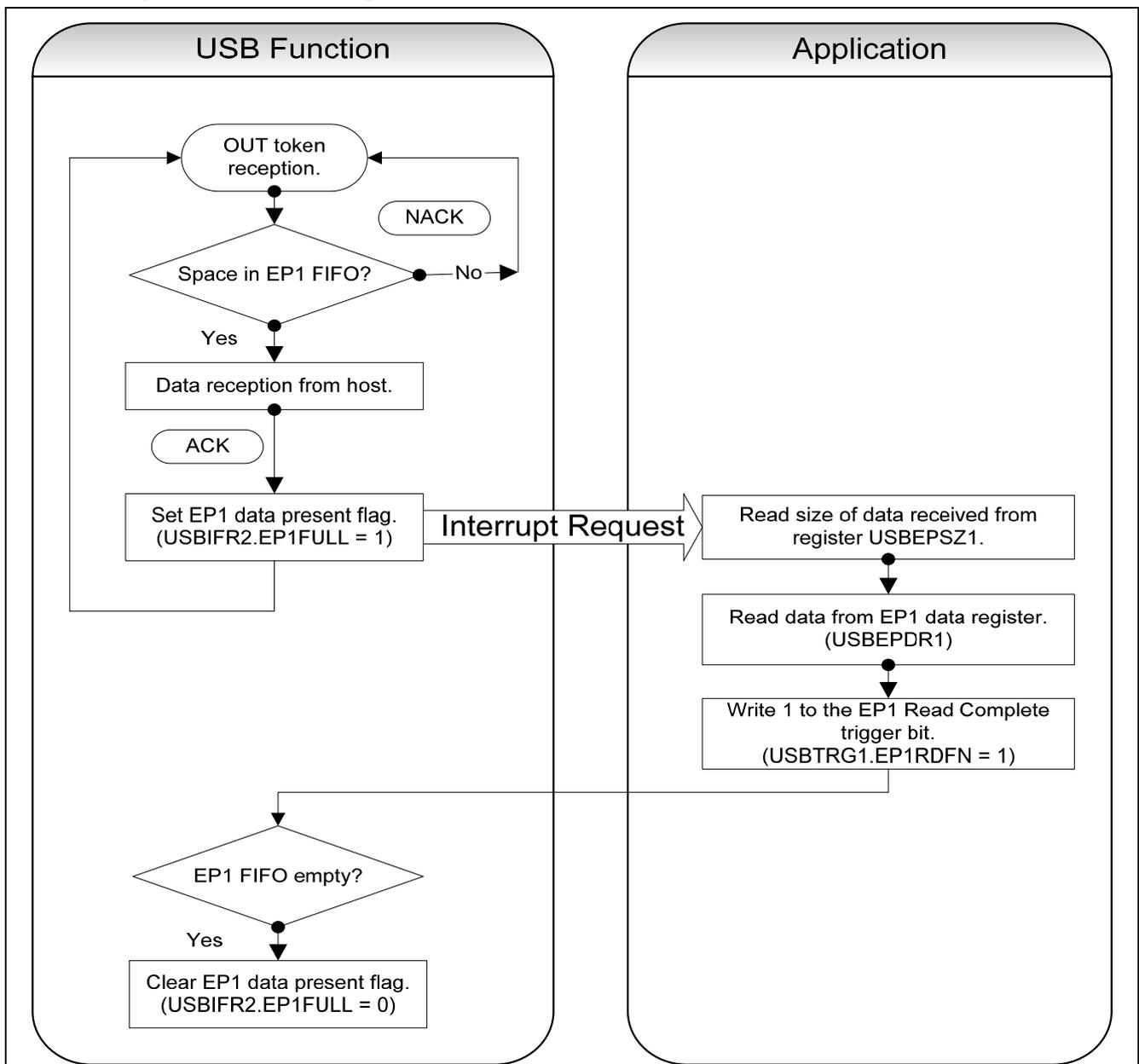


Figure 10. EP1 Bulk OUT Transfer.

### 5. Interrupt Transfer

The M16C/6C supports two identical Interrupt IN endpoints, Endpoint 3 and Endpoint 6. The following assumes EP3 is being used.

To start an Interrupt IN transfer, send up to a packets worth of data to be sent to the host by writing the data, one byte at a time, to the Endpoint 3 Data register (USBEPDR3) until it has all been sent or until the maximum packet size of 16 bytes has been written. Set the Endpoint 3 Packet Enable Trigger bit (EP3PKTE) to send the data. If a full packet has been written then enable the Endpoint 3 Transmit Complete (EP3TS) interrupt. In the EP3TS interrupt handler clear the EP3TS interrupt flag and repeat this process to send up to another packets worth of data.

```
/*Clear the interrupt flag*/
/* Set EP3TS bit to 0. (Register must be accessed as 8bit)*/
usbifr2 = (unsigned char)~BIT_4;
```

Repeat this handling of the EP3TS interrupt until there is no more data to send. In some cases the host knows how much data the device will be sending but usually a transfer will need to be ended by ensuring that a non-complete packet of data has been sent. If the amount of data is equal to an integer number of full packets then in order to end the transfer on a non-complete packet it is necessary to send an empty packet by setting the EP3PKTE bit without first having written any data to USBEPDR3. The EP3TS interrupt can then be disabled.

The following flowchart details this operation:-

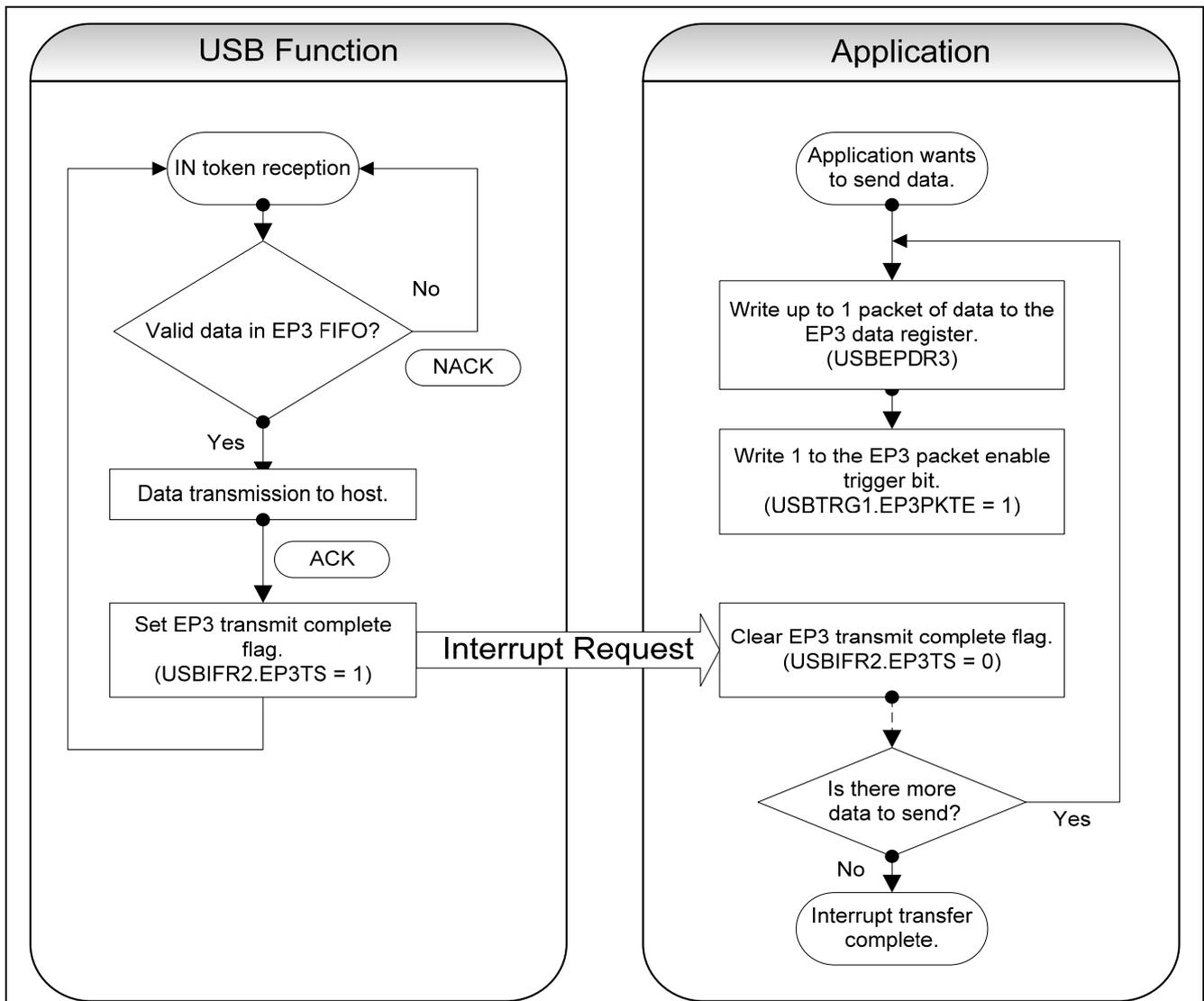


Figure 11. EP3 Interrupt Transfer.

## 6. Stall

A stall is a type of handshake packet that can be sent from a function to a host. This is used to indicate that a particular endpoint can not currently receive or transmit data or that a control pipe request is not supported. The USB Mass Storage class also uses it to end a transfer prematurely. A host can also command a function to enter a stalled state.

The M16C/6C automatically handles the ‘Set Feature (ENDPOINT\_HALT)’ and ‘Clear Feature (ENDPOINT\_HALT)’ commands for stall processing. However, if it is the function that needs to instigate the stall then this can be done by setting the endpoint’s Stall Setting Bit in the USB Endpoint Stall registers.

```
/* Stall Endpoint 0, set EPOSTLS bit*/
usbepstl0 = BIT_0;
```

Each endpoint also has a corresponding ‘Stall Clear Bit’ which can be set to clear the ‘Stall Setting Bit’. Endpoint 0’s stall is automatically cleared when the next Setup Control command is received. All endpoints can also independently be set up so that the endpoint’s Stall Setting Bit is automatically cleared after a stall condition has been returned to the host. This is achieved by setting the endpoint’s corresponding ‘Automatic Stall Clear’ bit.

```
/* Enable Automatic stall clear on EP1 */
eplasce_usbstlsr1 = 1;
```

## 7. Suspend / Resume

NOTE: This document does not cover disabling the USB module in the suspended state or the operation of the remote wakeup feature. Please refer to the M16C/6C Hardware Manual for this.

The Suspend and Resume detect flag (SURSF) interrupt can be handled to know when the USB bus has been suspended and subsequently when it has been resumed. If not actually disabling the USB Module during the suspended state there is no need to handle the Standby Clear Resume Detect (SSRSME) interrupt. The device can instigate a wake up of the USB bus from the suspended state by executing a remote wakeup operation. If not using this feature then ensure that the Resume Request bit (RSME) is always set to 0.

## 8. Cable Disconnection

When the USB cable is disconnected the USB Bus Connection/Disconnection Flag (VBUSF) will be set and a VBUSF interrupt generated. In the handler of this interrupt the VBUSMN flag in register USBIFR0 can be checked to confirm that this is a disconnection (VBUSMN = 0) event. Clear the VBUSF flag.

## 9. Reference

### 9.1 USB Endpoint Information Loading

The USB Endpoint Information Register (USBEPiR) allows each endpoint to be configured for a specific USB Configuration, Interface and Alternate setting. The endpoint number as the host sees it can also be changed. This may be useful if the host application requires specific endpoints to support specific transfer types. The following code shows writing to the USBEPiR register in order to set up all endpoints for a device with a single USB Configuration and Interface with no Alternate setting. Each endpoint requires 5 byte writes to the USBEPiR register. If not using a certain endpoint then write all zeros to it. Once the configuration has been completed it can not be changed.

```

/* EP 0 */
usbepir = 0x00;
usbepir = 0x00;
usbepir = 0x20;
usbepir = 0x00;
usbepir = 0x00;

/* EP 1 */           /* EP 4 */
usbepir = 0x14;       usbepir = 0x44;
usbepir = 0x20;       usbepir = 0x20;
usbepir = 0x80;       usbepir = 0x80;
usbepir = 0x00;       usbepir = 0x00;
usbepir = 0x01;       usbepir = 0x04;

/* EP 2 */           /* EP 5 */
usbepir = 0x24;       usbepir = 0x54;
usbepir = 0x28;       usbepir = 0x28;
usbepir = 0x80;       usbepir = 0x80;
usbepir = 0x00;       usbepir = 0x00;
usbepir = 0x02;       usbepir = 0x05;

/* EP 3 */           /* EP 6 */
usbepir = 0x34;       usbepir = 0x64;
usbepir = 0x38;       usbepir = 0x38;
usbepir = 0x20;       usbepir = 0x20;
usbepir = 0x00;       usbepir = 0x00;
usbepir = 0x03;       usbepir = 0x06;

```

## Website and Support

Renesas Technology Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

[csc@renesas.com](mailto:csc@renesas.com)

All trademarks and registered trademarks are the property of their respective owners.

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (<http://www.renesas.com>)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
  - (1) artificial life support devices or systems
  - (2) surgical implantations
  - (3) healthcare intervention (e.g., excision, administration of medication, etc.)
  - (4) any other purposes that pose a direct threat to human life
 Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.