NUC442/472 Series Errata Sheet

Errata Sheet for 32-bit NuMicro™ Family

Document Information

Abstract	This errata sheet describes the functional problem known at the release date of this document.
Apply to	NUC442/472 Series.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 Overview

Functional Problem	Description
Arbitration problem of USBD endpoints data buffer	USB buffer arbitration issue: The written data will be unexpected if the Cortex®-M4 core and any other master concurrently write data into USBD endpoints data buffer; vice versa, the read data will be unexpected if the Cortex®-M4 core and any other master concurrently read data from USBD endpoints data buffer. ISO IN access issue: The read data will be unexpected if Host reads ISO IN endpoint data buffer and any other master (Cortex®-M4 core or USB DMA) reads data concurrently from USBD endpoints data buffer.
USBD Endpoint Number problem	All endpoints of a USB device controller cannot be configured as the same Endpoint Number.
Control SETUP Transaction Failed	Control SETUP Transaction may fail when the SPLIT transaction occurs.

2 Functional Problems

2.1 Arbitration Problem of USBD Endpoints Data Buffer

Description:

For USBD controller, there's a data buffer shared by all endpoints. The IN/OUT transfer data are temporarily stored in this data buffer. During USB device operation, the masters including Cortex®-M4 core, USBD DMA and remote USB Host may access this data concurrently.

Problem:

If the Cortex®-M4 core and any other master concurrently read or write the data buffer, the read or written data will be unexpected. If the remote USB Host reads the ISO endpoint data buffer and any one mater (Cortex®-M4 core or USB DMA) read the endpoint data buffer concurrently, the read data will be unexpected.

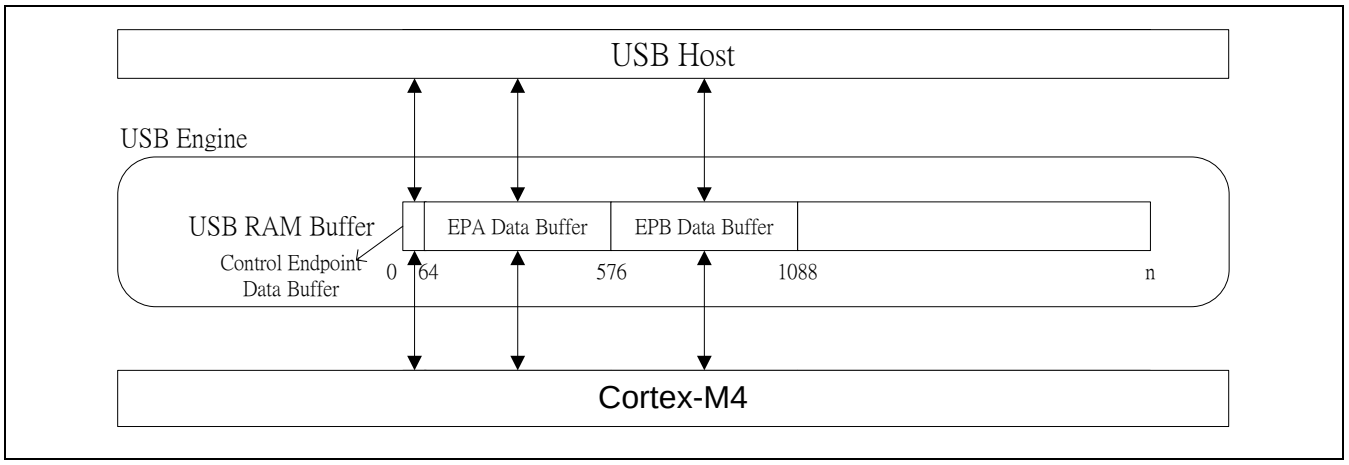

Figure 2-1 USB Endpoints Data Buffer

Cortex®-M4 core access issue:

For example, if the Cortex®-M4 core writes data into IN endpoint buffer and remote USB Host issues OUT transfer to write data buffer at the same time, the written data will be unexpected. Similarly, if the Cortex®-M4 core is reading data from OUT endpoint buffer while the remote USB host is issuing IN transfer to read data buffer, the read data will be unexpected.

ISO IN access issue:

If any master (Cortex®-M4 core or USB DMA) is reading data from endpoint buffer while the remote USB host is issuing IN transfer for ISO endpoint to read data buffer, the read data will be unexpected.

Workaround:

USBD firmware should prevent such case that may cause data unexpected and here are workaround solutions for the arbitration problem.

- Cortex®-M4 core access issue
  - User must make sure that no any other master reads or writes endpoints data buffer concurrently
- ISO IN access issue
  - Prevent to read data and USB Host reads ISO IN endpoints data buffer concurrently
    - Before reading endpoint data buffer, it needs to make sure that the ISO IN data buffer is empty (already read by USB Host).

The following example illustrates an USB device application example with ISO IN endpoint and Cortex®-M4 core Read/Write (for Control Endpoint), and introduces how to apply the above rules to avoid data unexpected issue.

Audio Device– Speaker & Microphone:

A UAC (USB Audio Class) application consists of Control Endpoint, ISO IN Endpoint, ISO OUT Endpoint, and interrupt IN Endpoint.

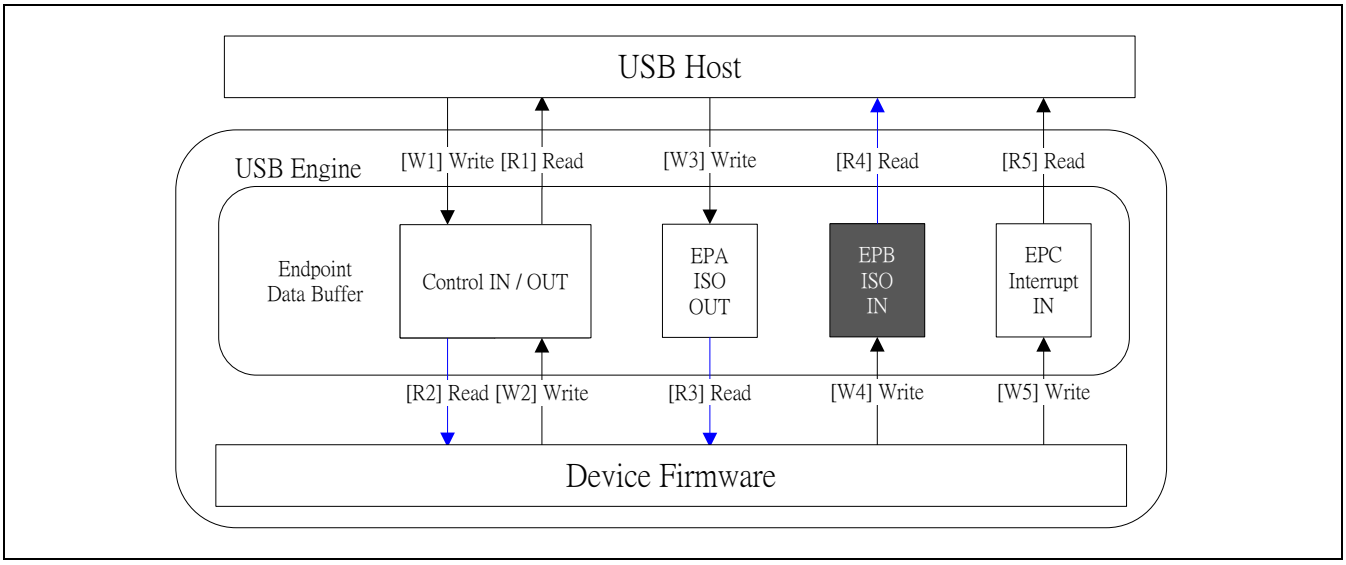

Figure 2-2 Audio Class Device

The UAC application uses:

- Control Endpoint to get (or set) volume control, mute control and sampling rate (use the Cortex®-M4 core).
- ISO IN Endpoint to transfer record data (Use USBD DMA).
- ISO OUT Endpoint to transfer audio data for playback (Use USBD DMA).
- Interrupt IN Endpoint to transfer HID data (Use USBD DMA).

Cortex®-M4 core access issue:

First, unless you can make sure that Cortex®-M4 core and any other master will not read or write endpoints data buffer concurrently, please use USBD DMA to access endpoints data buffer instead of Cortex®-M4 core.

Here is a workaround solution (for Control Endpoint only) that using the Cortex®-M4 core to access Control Endpoint data buffer and avoid the data unexpected issue.

UAC Class request belongs to the Control Read and Write Sequence (see Figure 2-3) and it contains Setup stages, Data stage, and Status stages.

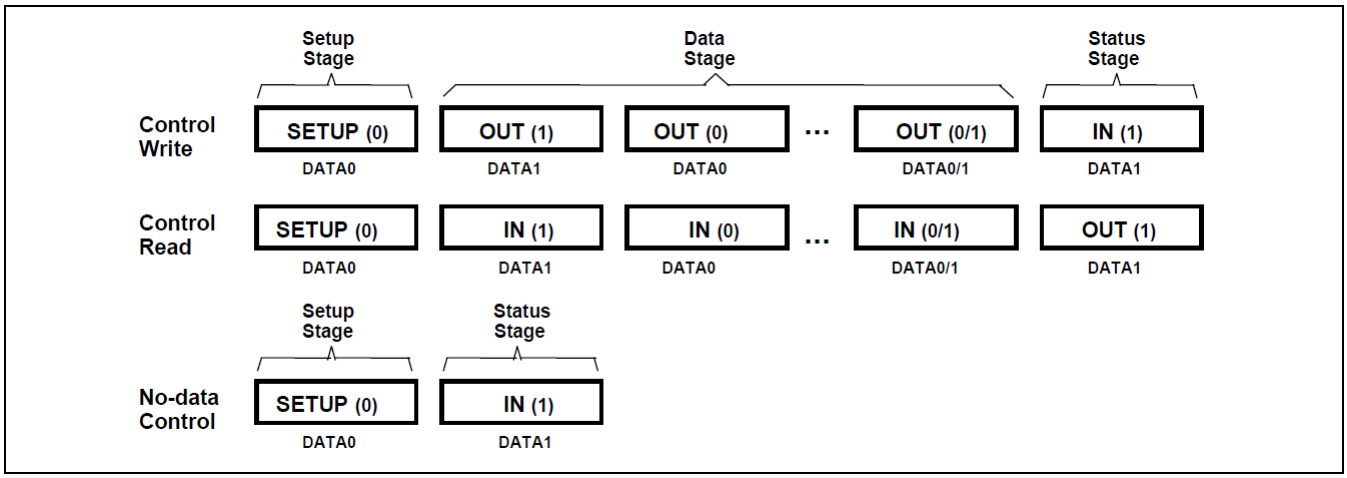

Figure 2-3 Control Read and Write Sequences

For Control Write sequence (e.g. Set Volume):

The data unexpected condition is that the Cortex®-M4 core reads the Control Endpoint data buffer (see Figure 2-2 [R2]) while Host issues ISO IN transfer (see Figure 2-2 [R4]) or Interrupt IN transfer (see Figure 2-2 [R5]).

Normally, user can read Control Endpoint data buffer when getting the data received interrupt (after Data Stage). But the data unexpected issue may occur when USB Host may issue IN Token (Host reads) for other endpoints before Status Stage (see Figure 2-4).

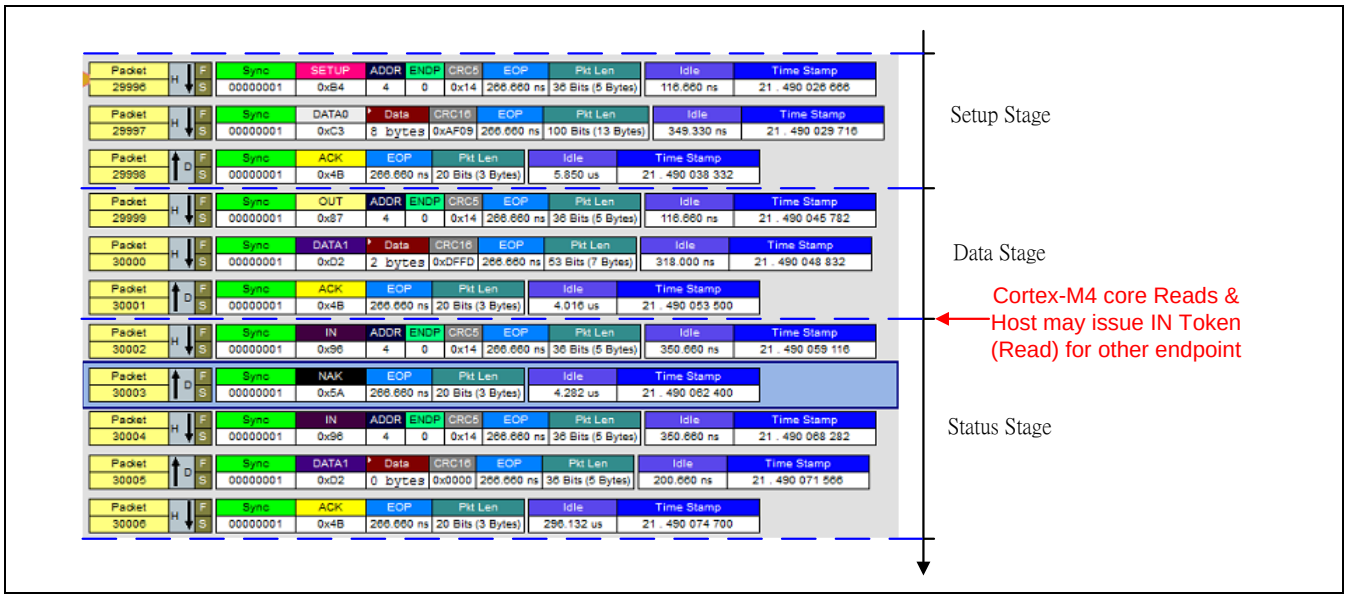

Figure 2-4 Control Write Sequence (Unsafe Reading Timing)

To avoid this situation, user can read Control Endpoint data buffer when getting the Control IN Token for Status Stage instead of the time after getting data received interrupt. In Status Stage, Host does nothing and only waits the ACK from device. Thus, it is safe to read the data from Control endpoint data buffer in Status Stage by the Cortex®-M4 core (see Figure 2-5).

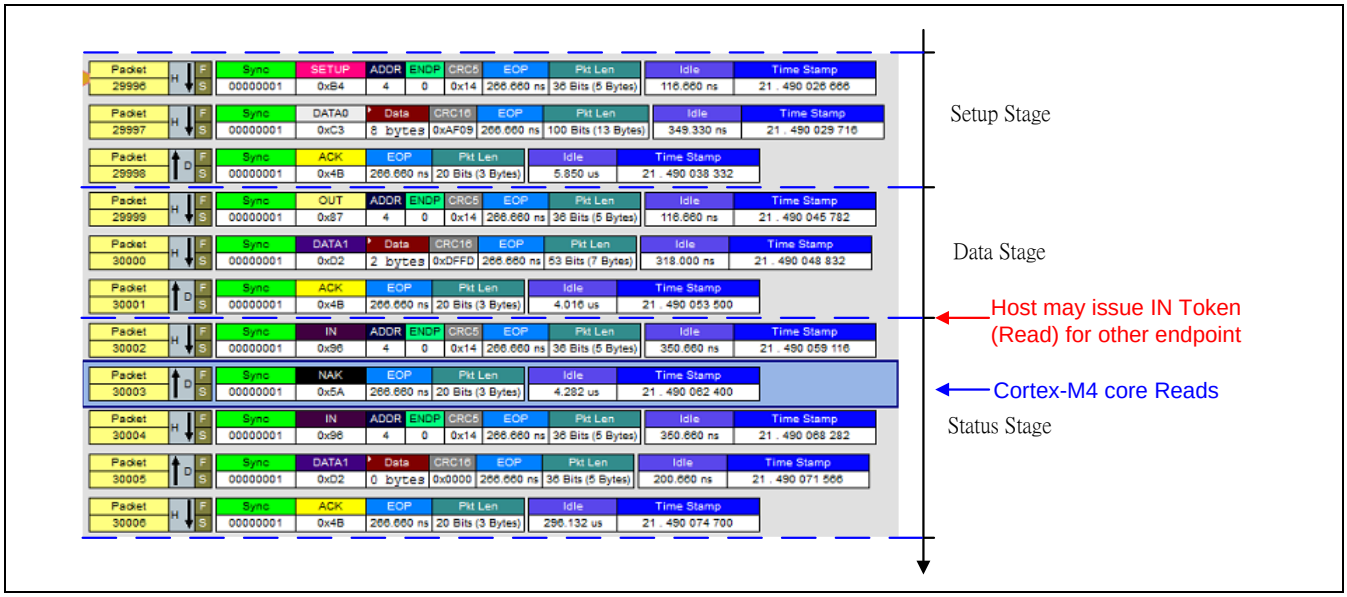

Figure 2-5 Control Write Sequence (Safe Reading Timing)

For Control Read Sequence (e.g. Get Volume):

The data unexpected condition is that the Cortex®-M4 core writes the Control Endpoint data buffer (see Figure 2-2 [W2]) while Host issues ISO OUT transfer (see Figure 2-2 [W4]).

Normally, user can write Control Endpoint data buffer after receiving the command (after Step Stage). But the data unexpected issue may occur when USB Host may issue OUT Token (Host writes) for other endpoints before Data Stage (see Figure 2-6).

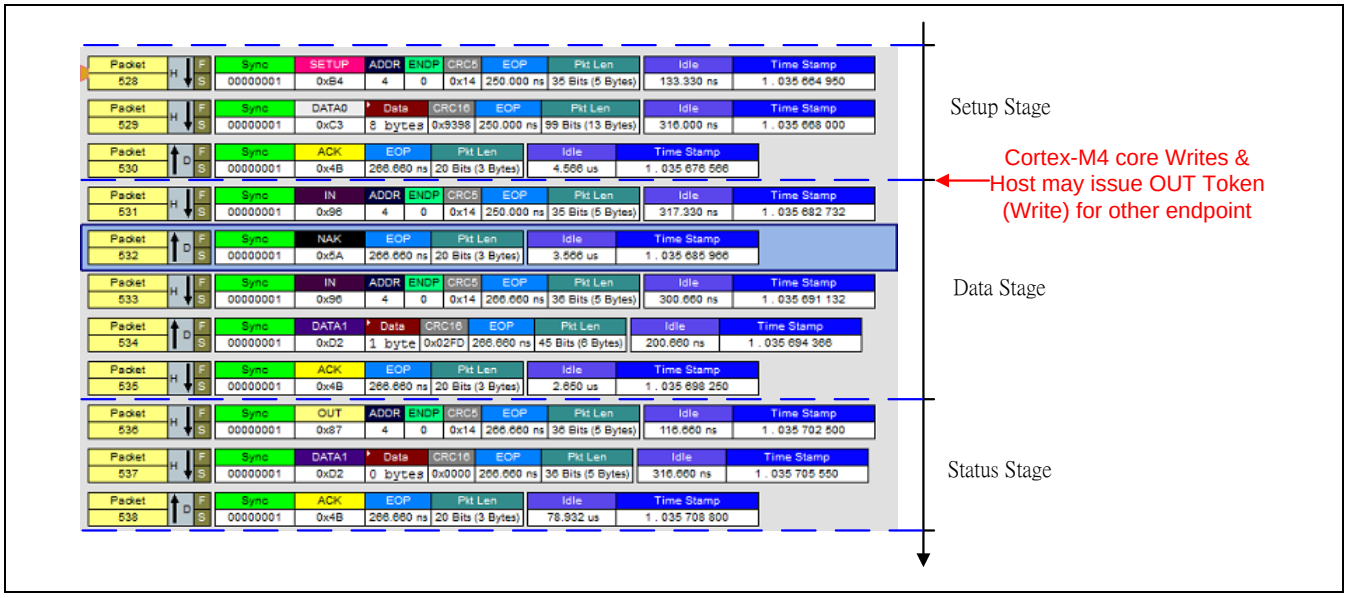

Figure 2-6 Control Read Sequence (Unsafe Reading Timing)

To avoid this situation, user can write Control Endpoint data buffer when getting the Control IN Token for Data Stage instead of the time after getting command. In Data Stage, Host does nothing and only waits the data from device. Thus, it is safe to write the data from Control endpoint data buffer in Data Stage by the Cortex®-M4 core (see Figure 2-7).

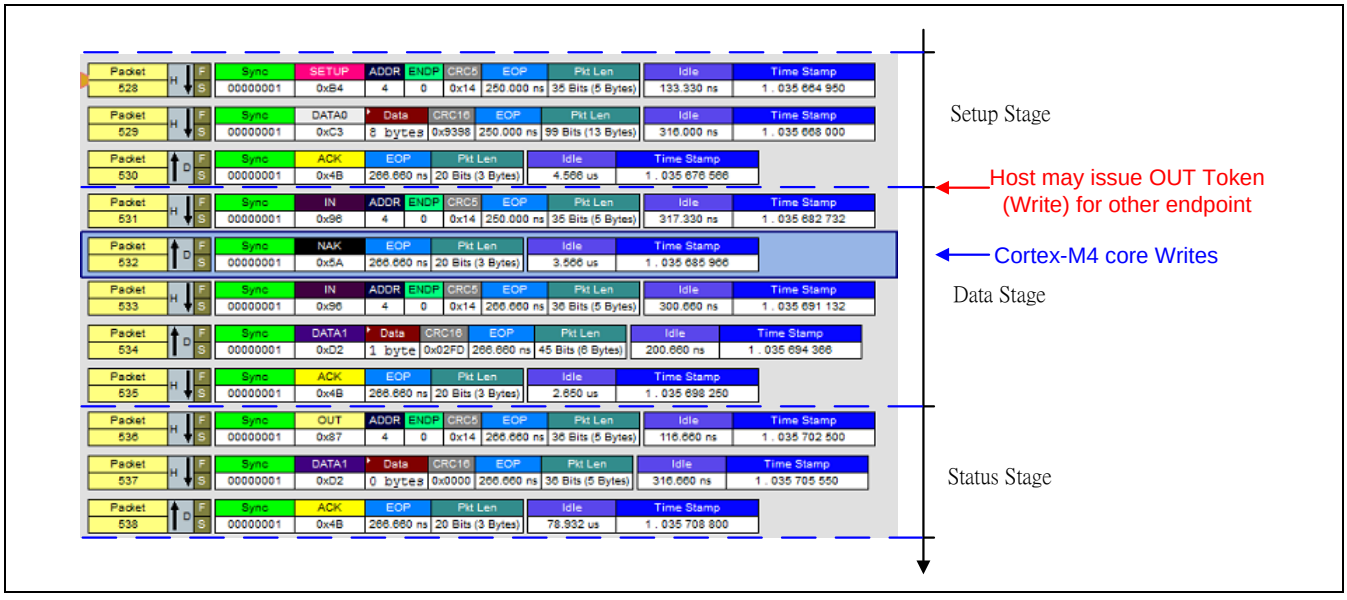

Figure 2-7 Control Read Sequence (Safe Reading Timing)

ISO IN Issue:

In addition to the Cortex®-M4 core access issue, the data unexpected issue may still occur when USBD DMA reads the ISO OUT Endpoint data buffer (see Figure 2-2 [R3]) while Host issues ISO IN transfer (see Figure 2-2 [R4]).

Here are some examples for the timing that USBD DMA reads the ISO OUT Endpoint data buffer (see Figure 2-2 [R3]) and Host issues ISO IN transfer (see Figure 2-2 [R4]). If IN token and OUT token come too closely, the data unexpected case may occur. Figure 2-8 shows the safe timing cases (Timing 1 to Timing 3) and the unsafe timing case (Timing 4).

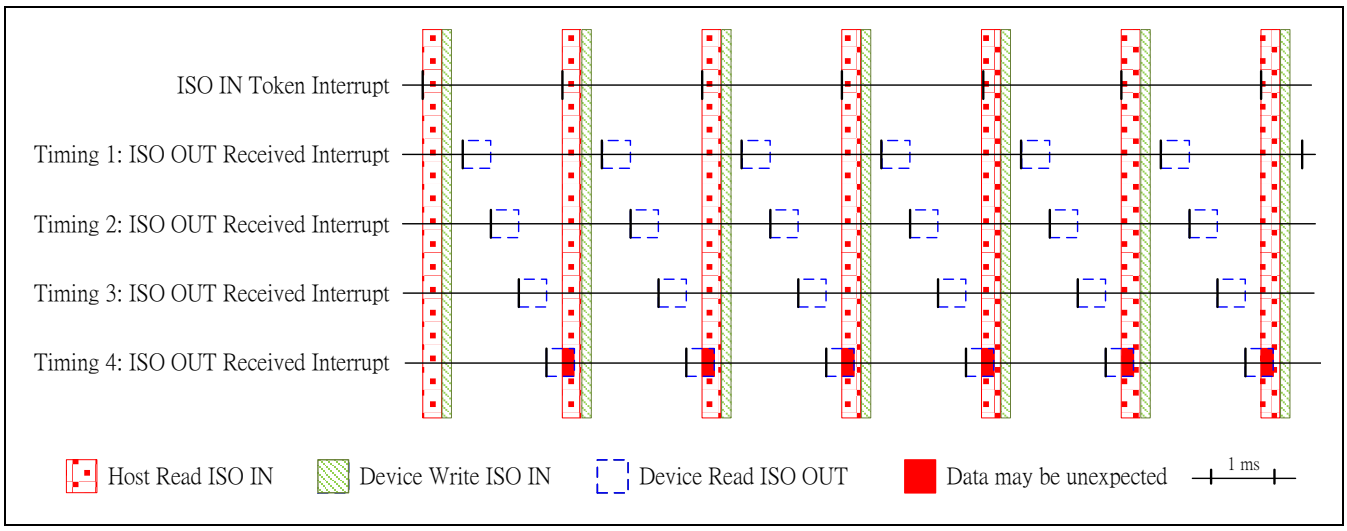

Figure 2-8 Example for Time to Access the Endpoint Buffer

Here is a solution to avoid the unsafe case (see Figure 2-8 Timing 4).

The USB Device firmware needs to service ISO IN and ISO OUT interrupt only when the other one endpoint data buffer is empty. If USB Device firmware reads ISO OUT Endpoint after ISO IN endpoint is empty, Host will not read data from ISO IN Endpoint even when Host issue IN Token. The USB Device firmware writes ISO IN Endpoint after ISO OUT Endpoint is empty to make sure that it reads ISO OUT data first if there is data in ISO OUT Endpoint data buffer. It balances the time to service ISO IN and OUT Endpoints (see Figure 2-9).

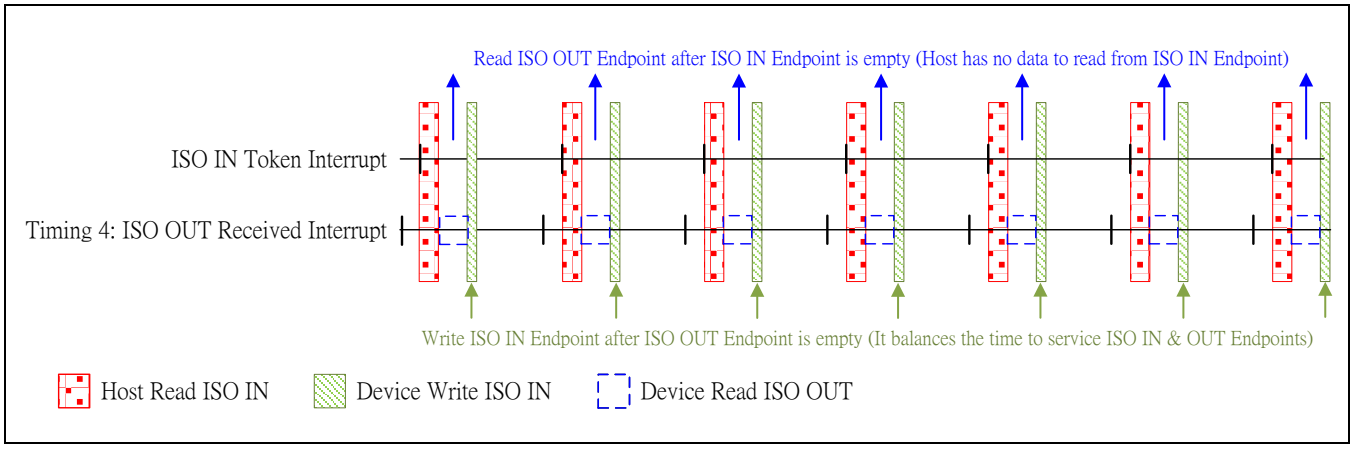

Figure 2-9 Solution Timing for UAC Example

2.2 USBD Endpoint Number

Description:

All endpoints of USB device controller cannot be configured as the same Endpoint Number.

Problem:

The USB device controller didn't check the direction when getting data from host. If configuring EPA and EPB with the same endpoint number, one for IN, and the other for OUT, the OUT data will be received into EPA buffer. The data in EPA buffer will be over-written.

Workaround:

Provide the Endpoints with different endpoint numbers.

2.3 Control SETUP Transaction Failed

Description:

The USB Device controller receives all the Setup Tokens (even though the token is not for it) on USB Bus and only responds (ACK/NAK/NYET/STALL) to the token for it.

Problem:

Host controllers and hubs support one additional transaction type called split transactions. This transaction type allows full- and low-speed devices to be attached to hubs operating at high speed. The USB Device controller may not respond (ACK/NAK/NYET/STALL) to the Setup token for it when split transactions occur and other full- or low-speed devices also do the Control SETUP Transaction.

Workaround:

No workaround solution.

Revision History

Date	Revision	Description
2016.11.25	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.