Host/Hub Detection and Software Generation of USB Disconnect on the RC32355/RC32351

Notes

By Imran Hoda

Revision History

July 9, 2002: Initial publication.

Introduction

Systems which use either the RC32355 or RC32351 integrated processor as a USB slave device and generate a reset to the processor while the slave and host are connected by a USB cable will have problems with the host recognizing that it needs to reconfigure the RC32355/RC32351 after the reset is complete. This application note describes a solution for this problem. The solution can also be used to implement a software-controlled USB disconnect which can be initiated at any time under software control, independent of whether the RC32355 or RC32351 has experienced a reset or not.

Systems which do not reset the RC32355/RC32351 while the slave and host are connected by a USB cable do not require this modification.

Operation

A USB host detects the speed of a USB slave device by sensing the status of the two USB data lines. The two data lines are commonly referred to as the D+ and D- lines. The USB master has very weak (15K) pull-downs on both the D+ and D- line. The slave device will have a relatively strong (1.5 K) pull-up on one of these two lines. If the slave pulls D+ high, it indicates the slave is a high-speed USB device (12 Mb/s). If, instead, the slave pulls D- high, then it indicates the slave can only sustain low speed USB data rates (1.5 Mb/s).

These data line pull-ups also serve another function. When the USB host detects that both data lines are low, it assumes nothing is attached. This is referred to as a "disconnect" condition in the USB specification. As soon as one of the data lines is pulled up, the host assumes that a new slave has been plugged in, and it goes out and tries to configure it. This will be recognized as a "connect" condition by the USB host or hub.

The RC32355- and RC32351-based boards implement the pull-up register and the "disconnect" and "connect" conditions are correctly detected when the USB cable is disconnected from and then reconnected to the host. The problem arises when the RC32355/RC32351 are powered on or reset while the board is connected to the host through the USB cable. In the case of initial power-on, the host is notified that there is a USB device through the "connect" condition before the RC32355/RC32351 can configure the USB interface. This results in the RC32355/RC32351 not being recognized by the host. If the board is then reset, power remains applied to the pull-up, so the state of the data line never changes. As a result, when reset is completed, the host does not realize that a newly initialized USB device is now attached, and it makes no effort to re-initialize the RC32355/RC32351 USB interface.

USB Connect and Disconnect Conditions

The USB is terminated at the hub as shown in Figure 1. Full-speed and low-speed devices are differentiated by the position of the pull-up resistor on the downstream end of the cable:

- Full-speed devices are terminated with a pull-up resistor on the D+ line. The RC32355/RC32351 are full-speed devices.
- Low-speed devices are terminated with a pull-up resistor on the D- line.

Notes

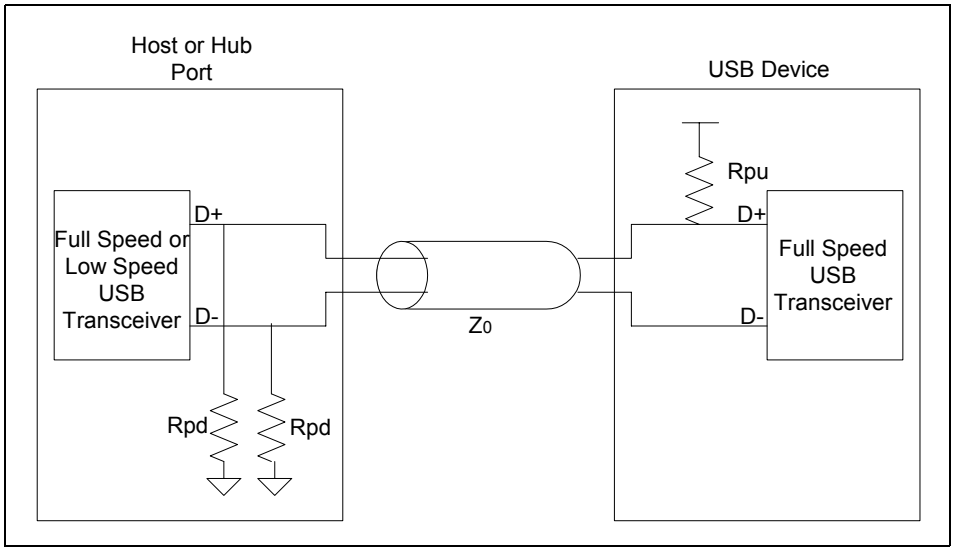

Figure 1 Full Speed Device Cable and Resistor Connections

Disconnect Condition

When no USB slave device is attached to the downstream port of the host or the hub and the host or hub is not driving that port, it will cause both D+ and D- to be pulled below the single-ended low threshold of the host or hub. The host or hub will interpret this to be a disconnect condition

Connect Condition

A connect condition will be detected when the hub detects that one of the data lines (D+ or D-) has been pulled above the host/hub's VIH threshold for more than 2.5µs. In the example shown in Figure 1, the D+ line gets pulled high through Rpu when the USB device is connected.

Problem During Power-on or RESET of the RC32355/RC32351 with USB Cable Attached

When the RC32355 or RC32351 processor is powered on, the D+ line goes high causing the host to detect the device. Although the device fulfils the "connect" condition, the detection of the device is premature. Both the RC32355 and RC32351 have to initialize all their registers and interfaces before the USB host detects them. During RC32355/RC32351 RESET, the host does not detect the status change of the RC32355/RC32351 slave port. However, the RESET nullifies the RC32355/RC32351 setup that was performed by the host after the initial power-up sequence. As a result, the RC32355and RC32351 are left in an uninitialized state and will not respond to the USB host device's transactions.

Solution

Whenever the RC32355 or RC32351 is powered on or reset, the D+ line must be driven low to indicate to the host that it needs to come in and reconfigure the RC32355/RC32351 USB port. This can be accomplished using a hardware/software solution as follows:

The hardware portion of the solution is implemented by disconnecting the 1.5K pullup from Vcc and connecting the side of the resistor that was formerly attached to Vcc to a general purpose I/O (GPIO) pin instead. Since the state of the GPIO pins is software controllable, software can then be used to manage the state of the USB port.

The D+ line needs to be held low during the entire power-on or reset operation. This applies to both hardware and software generated resets. The purpose of keeping the D+ line low is to keep the host from prematurely detecting the RC32355/RC32351 USB slave device. The D+ line should go high right before the RC32355/RC32351 software configures their USB interface. It takes about 20 ms for the host to detect

Notes

and respond to the presence of the slave device once the GPIO pin is driven high. This 20 ms window is more than adequate for the RC32355/RC32351 software to complete the initialization of the RC32355/ RC32351 USB slave interface.

Example

In this example, GPIO_26 is used. This is the GPIO that IDT used for this purpose on the 79RP351 and 79RP355 reference designs. The following software sequence needs to be placed at the beginning of the boot code:

1. Set GPIO_26 as a GPIO pin. (This pin has an alternate function. Therefore, it must be verified that it is properly configured to be a GPIO.)
2. Set the GPIO_26 as an output.
3. Set the GPIO to drive a value of zero (i.e., set it low).

This is the code in Assembly:

```
/* The PIO_BASE_ADDR is 0xb8040000*/
li t0,PIO_BASE_ADDR
li t1,0xf9ffffff
sw t1, 0x0(t0)                  /* Setting pin 26 as GPIO*/
lw t2, 0x4(t0)
or t2, 0x04000000
sw t2, 0x4(t0)                  /*Setting pin 26 as an output*/
lw t1, 0x8(t0)
and t1, ~0x04000000
sw t1, 0x8(t0)                  /*Setting pin 26 to zero*/
```

Next, perform the following steps for USB configuration.

1. Set the GPIO_26 to output 1 (i.e., set it high). This should be a part of the configuration code as the first step.

   This is the code for setting the GPIO pin 26 to 1:

   ```
   /* GPIOD register is at 0xb8040008 - set bit 26 to 1*/
   T_UINT32 * gpiod = (T_UINT32*)(0xb8040008);
   *gpiod |= 0x04000000;
   ```

2. Run the USB configuration code.

Notes

Figure 2 is a timing diagram showing each step of the USB Configuration.

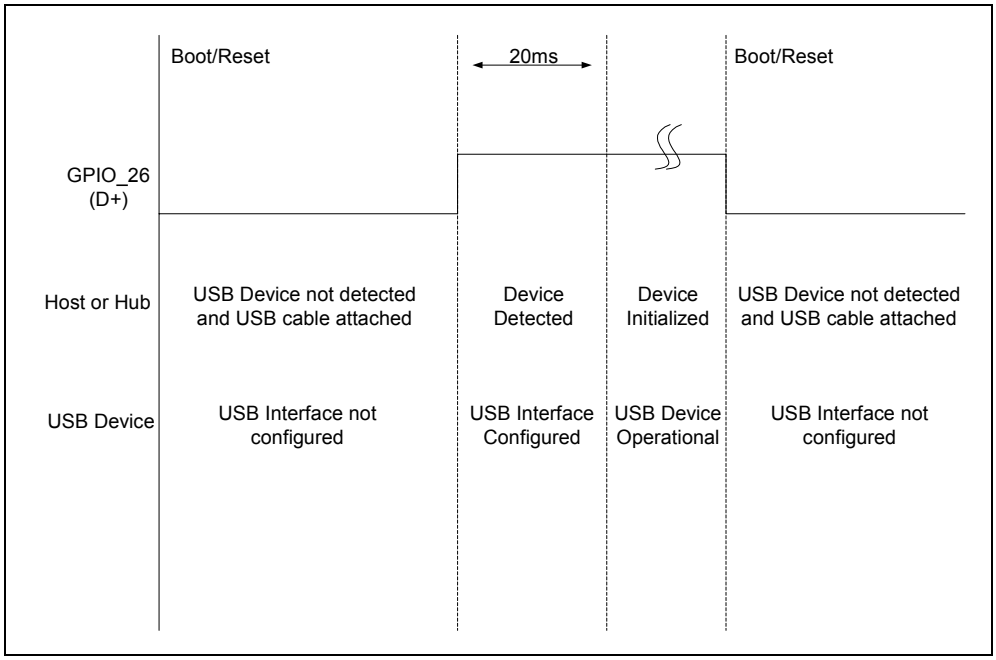

Figure 2 Timing Diagram of USB Configuration

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.