AN3051

ISP1161A1 frequently asked questions

Application Notes

Abstract

This document contains the frequently asked questions related to the ISP1161A1.

CD00246765 Rev 02 2009-09-30 ISP1161A1

Legal Information

Contents

1 About this document

1.1 Purpose

This document provides answers to frequently asked questions related to the ISP1161A1.

1.2 Scope

These frequently asked questions are for users who use the ISP1161A1. For additional details on the product, refer to the respective data sheets, application notes, and user manuals.

1.3 Prerequisites

1.3.1 Documentation

Users are urged to read the *ISP1161A1 USB single-chip host and device controller* data sheet.

1.4 Revision information

Table 1 Revision history

Date	Rev.	Comments
2001-07-01	01	First version.
2009-09-30	02	Rebranded to the ST-Ericsson template. Removed question on package information. Changed all occurrence of ISP1161 to ISP1161A1.

1.5 Reference list

[1]	*ISP1161A1 USB single-chip host and device controller data sheet*	*CD00222683*
[2]	*Universal Serial Bus Specification Rev. 2.0*	*www.usb.org*

2 General

2.1 Maximum sink and source currents

What are the maximum source and sink currents in the data bus, GL_N, PSWn_N pins?

The source and sink currents are 4 mA.

2.2 Crystal specification

What is the specification of the crystals used in the ISP1161A1 designs?

A 50 ppm quartz crystal of 6 MHz is used. Two external capacitors of around 22 pF are required. Check the clock waveform and decide in your PCBA system to get a better clock waveform for the capacitors.

2.3 D_VBUS, D_DP, and D_DM termination

If the device controller is not used, how should D_VBUS, D_DP, or D_DM terminate?

Leave the D_DP and D_DM pins open. Pull D_VBUS to LOW.

2.4 H_DP2 and H_DM2 termination

If only one USB downstream port is required, how should H_DP2 and H_DM2 terminate?

Leave them open.

2.5 Overcurrent protection

I do not need the overcurrent protection circuitry in the ISP1161A1. How should I connect the relevant pins?

The OC1_N and OC2_N pins must be pulled to HIGH and PSW1_N and PSW2_N must be left open (when V_{CC} = 5 V). Note, however, that according to USB specifications, the host controller must implement overcurrent protection on the downstream ports. When Vcc of 3.3 V is used together with an external overcurrent protection circuitry, bit 10 of the HcHardwareConfiguration register (20h) must be cleared to 0 so that the digital input (of 5 V or 3.3 V) can be received on the OC1_N pin.

2.6 DACK and EOT pins

If we do not require the host or device DMA, how should we connect the DACK and EOT pins?

You can pull them HIGH through a 10 kΩ resistor.

Note: Ensure that DACKInputPolarity in the HcHardwareConfiguration register (20h) and DAKPOL in DcHardwareConfiguration (BAh) are 0.

2.7 EMI test

Was any EMI test performed on the ISP1161A1?

ST-Ericsson has tested for horizontal and vertical polarization for our PC evaluation kit ver 1.0. The following figures are a general plot of the emission characteristic of the ISP1161A1 USB host and device controllers with ferrite bead and open circuit at CLKOUT (48 MHz).

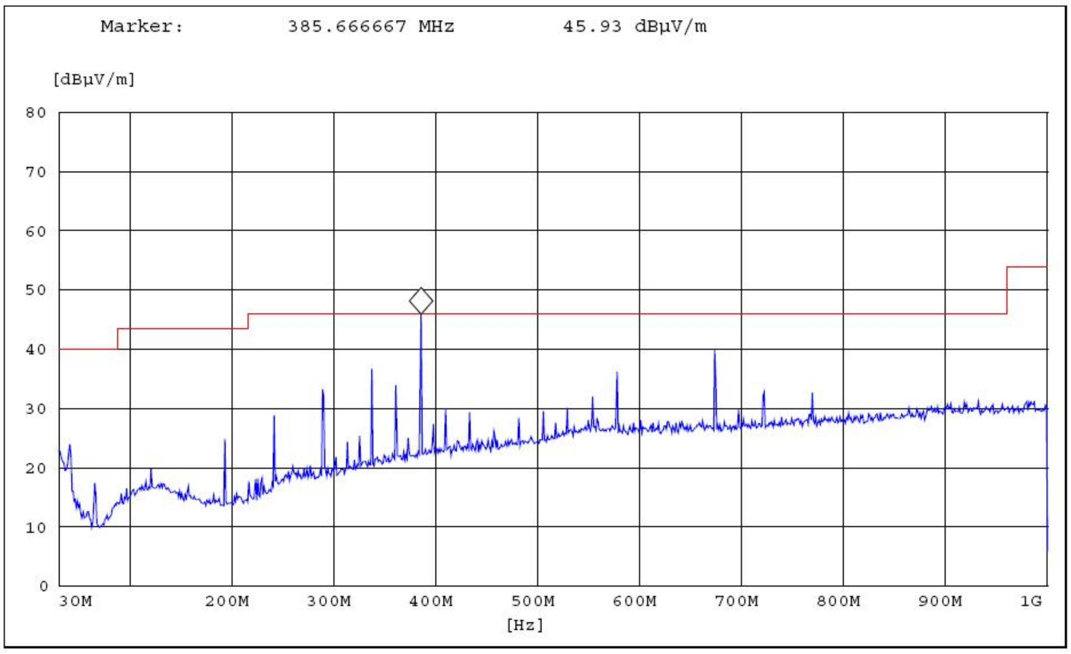


Figure 1 Horizontal polarization (peak measurement)

Table 2 Horizontal polarization

Frequency (MHz)	Level (dBuV/m)	Limit (dBuV/m)	Margin (dB)	Result
192.74	24.78	43.5	18.72	OK
241.24	28.82	46	17.18	OK
288.66	33.29	46	12.71	OK
337.16	36.71	46	9.29	OK
360.87	33.91	46	12.09	OK
385.66	45.93	46	0.07	OK
577.51	36.24	46	9.76	OK
673.43	40.05	46	5.95	OK

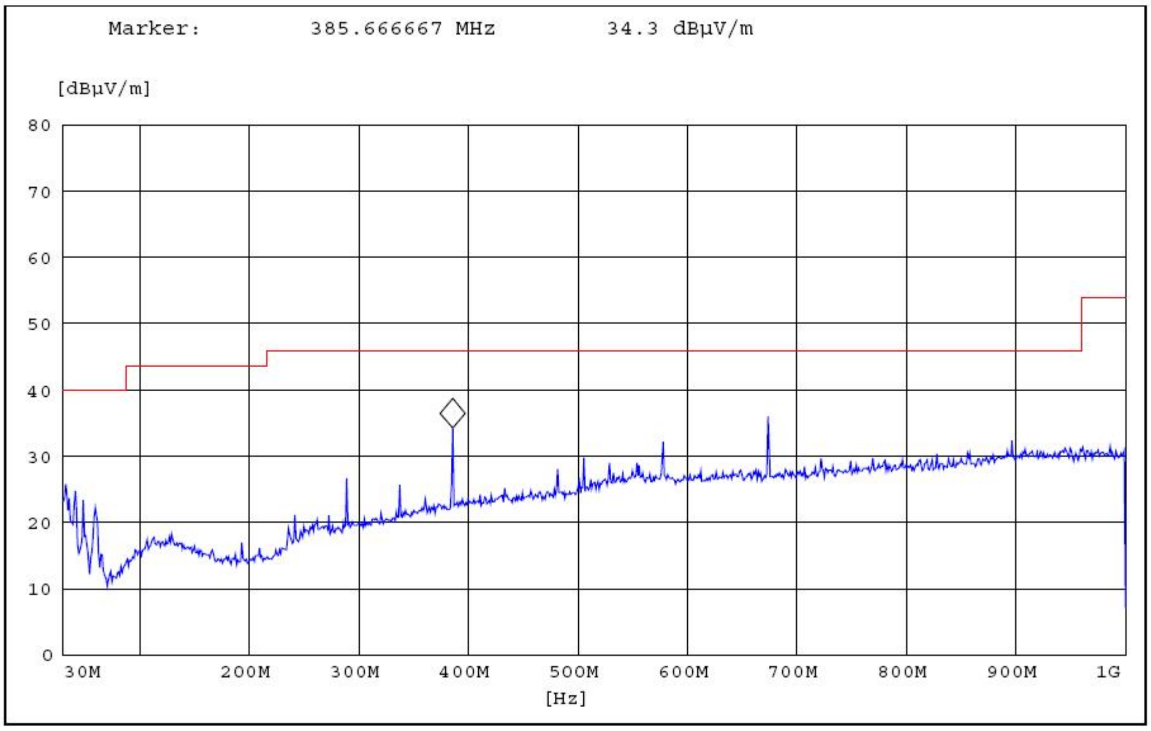


Figure 2 Vertical polarization (peak measurement)

Table 3 Vertical polarization

Frequency (MHz)	Level (dBuV/m)	Limit (dBuV/m)	Margin (dB)	Result
288.66	26.71	46	19.29	OK
337.16	25.69	46	20.31	OK
385.66	34.3	46	11.7	OK
481.58	28.11	46	17.89	OK
505.3	29.85	46	16.15	OK
577.51	32.17	46	13.83	OK
673.43	36.01	46	9.99	OK

2.8 PMOS switch

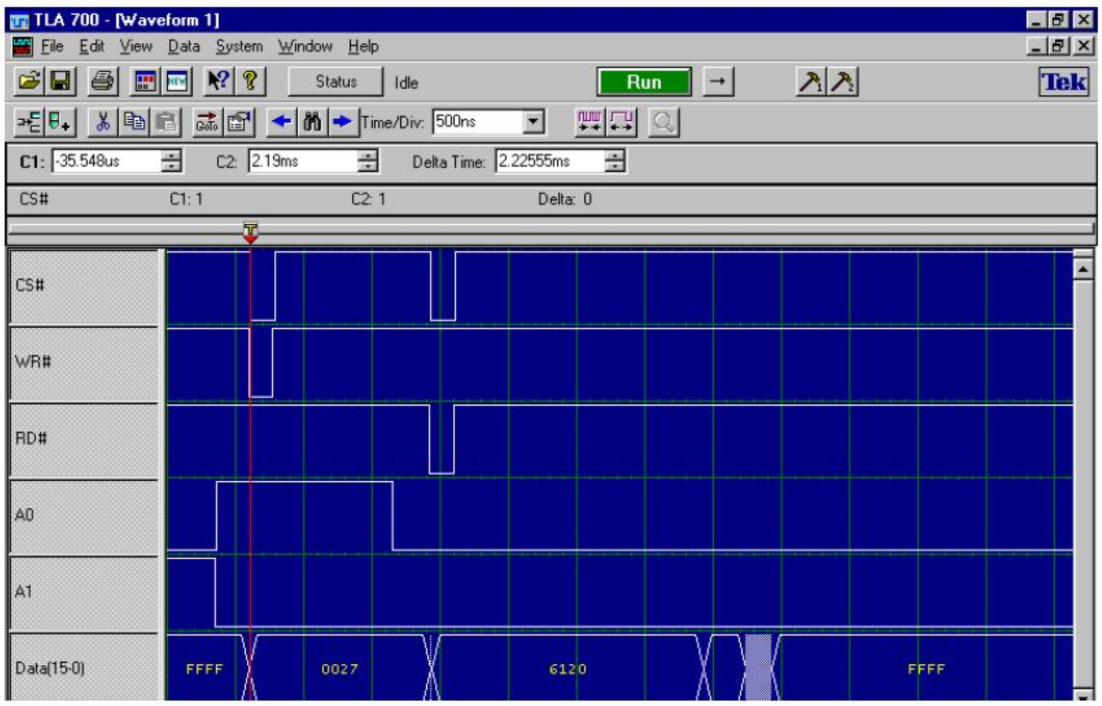
PHP109 is a Philips PMOS switch. Can you recommend switches by other vendors?

Mitsubishi FY3ABJ-03. The specification of the comparator within the ISP1161A1 is a voltage drop of about 75 mV. If you choose a Ron = 0.15 Ω of current >= 500 mA, this would be sufficient.

2.9 Reading HC

We cannot read anything from the HC. What could be wrong?

To check whether you can access the HC, try to read the ChipID or Scatch register. The following are the timing waveforms for your reference:



TLA 700 - [Waveform 1]

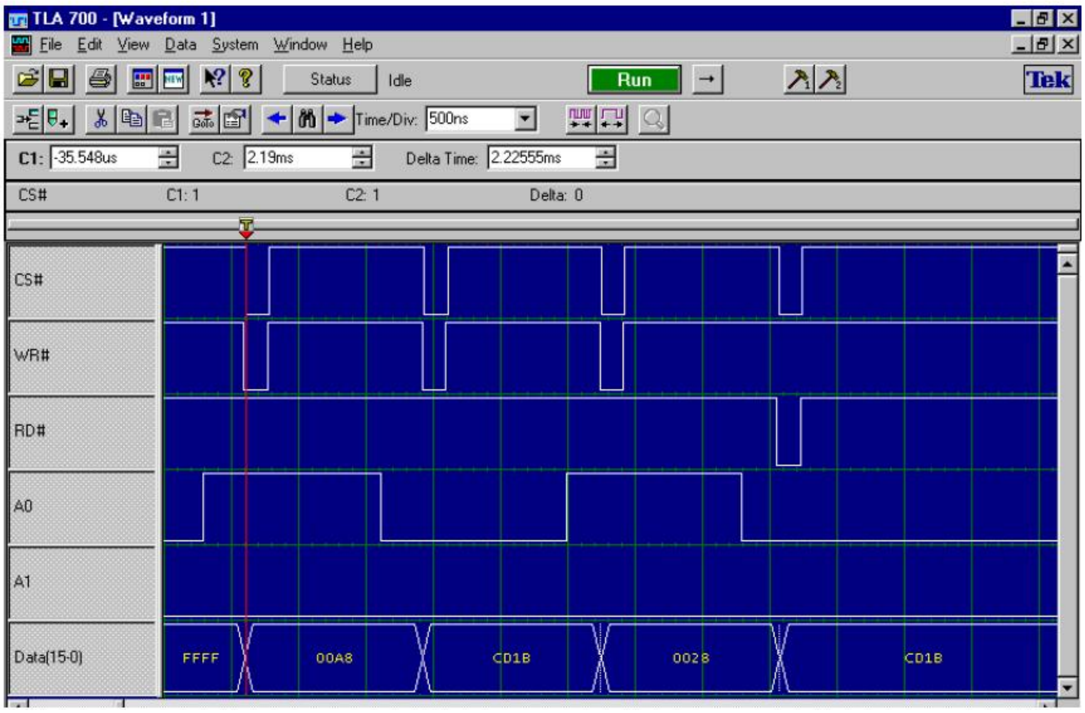
Status: Idle

Time/Div: 500ns

C1: -35.548us C2: 2.19ms Delta Time: 2.22555ms

CS# C1: 1 C2: 1 Delta: 0

Figure 3 Reading HC ChipID



Time/Div: 500ns
C1: -35.548us C2: 2.19ms Delta Time: 2.22555ms
CS# C1: 1 C2: 1 Delta: 0

Figure 4 Writing and reading of the HC Scratch register

2.10 V_{hold} pin

What is the V_{hold} pin?

This is the voltage holding pin. Connect a capacitor between this pin and ground.

2.11 NDP_SEL pin

What is the NDP_SEL pin used for?

It is used as an input pin to select the number of USB downstream ports. When pulled HIGH, it selects two downstream ports. When pulled LOW, it selects one downstream port.

2.12 V_{reg} pin

What is the V_{reg} pin for?

This pin is the internal regulator output pin. When Vcc is connected to +5 V, it outputs +3.3 V. When Vcc is connected to +3.3 V, ensure that this pin is connected to the external +3.3 V (Vcc). Do not use this pin to drive other circuitries that require 3.3 V supply.

2.13 RESET_N pin

What is the RESET_N pin used for?

This is for the hardware reset of the ISP1161A1. The pin configuration is active LOW and is used to reset all the internal logic in the ISP1161A1. The minimum timing requirement is 200 ns.

2.14 Power supply requirement

What is the power supply requirement for the ISP1161A1?

The ISP1161A1 can accept +5 V or +3.3 V as its power supply input. However, the ISP1161A1core works under +3.3 V.

2.15 Peer-to-peer USB connection

What is known as the "peer-to-peer" USB connection in the ISP1161A1?

The ISP1161A1chip integrates a USB host controller and a USB device controller. Therefore, it can emulate a PC (typically where a host controller resides), taking on the role of a slave host controller where necessary to control other USB devices. This is what is meant by a "peer-to-peer" USB connection. The protocol of the USB connectivity is still that of a master-slave relationship, and this relationship is not broken.

2.16 SoftConnect

What is SoftConnect?

SoftConnect is a trademark of ST-Ericsson, where a 1.5 kΩ resistor is integrated on the ISP1161A1, and not connected to Vcc by default. The connection to USB for full-speed devices is detected by pulling D+ to HIGH through the 1.5 kΩ resistor. This connection in the ISP1161A1is established through software. This enables the microprocessor to complete its initialization sequence before deciding to establish connection to the USB.

2.17 Maximum suspend current

What should be the maximum suspend current in the ISP1161A1?

According to the USB specification, the maximum current consumed during suspend mode must not exceed 500 µA. Because the ISP1161A1is specially targeted at PDAs where power consumption is critical, changes have been made to the ISP1161A1core so that it consumes only 40 µA (typical) of current in suspend mode while operating at +5 V or +3.3 V power supply.

3 Software

3.1 Handling errors in software

How should errors be handled in the software for the ISP1161A1?

Do not be alarmed when there are error codes reported by the ISP1161A1. There are many incidents on the USB bus that generate errors, such as noise on a bad cable and bad quality USB devices. The host controller has a built-in error reporting mechanism so that the driver software can read the error code and handle it properly. Errors are reported for the software to handle with care the payload and prepare for the next PTD.

There are two types of error reported by the ISP1161A1:

a) Fatal errors

b) Non-fatal errors

Table 4 Errors

Fatal errors	Error code	IN token	OUT token response
ERROR_CRC	01	No ACK sent	Not applicable
ERROR_Bitstuffing	02	No ACK sent	Not applicable
ERROR_DatatTogglingMismatch	03	ACK sent	Not applicable
ERROR_Stall	04	No ACK sent	Receive stall
ERROR_DeviceNotResponding	05	No ACK sent	Did not receive a HS reply within 18 bit time. Or a bad SYNC pulse.
ERROR_PIDCheckFailure	06	No ACK sent	Not applicable
ERROR_UnExpectedPID	07	No ACK sent	Corrupted ACK, STALL, or NAK
ERROR_DataOverRun	08	–	Not applicable
Non-Fatal Error (warning)			
ERROR_DataUnderRun	09	ACK sent	Not applicable

For all errors, the toggle bit is toggled by the ISP1161A1, and updated to the header, after the payload has been written to the PTD.

3.2 PTD during error

What happens to the PTD when errors occur?

- The Active bit will be set to 0, and the PTD will be stopped for that millisecond.
- The Toggle bit will be toggled all the time, for every error reported.
- The Actual Packet Size Byte is updated only to the Data payload if the Completion Code (CC) is 9.
- For all fatal errors, the actual byte is not updated even if there is an ACK sent by the ISP1161A1.
- The actual byte is not updated because the payload received is meant to be discarded.

3.3 Payload after error

What should software do with the Payload after receiving an error?

- IN/OUT-TOKEN transfer

1. For example, if the PTD started for the transfer must receive or send bytes of data to the device and...
2. If an error occurs, first check whether the active bit is 0 (TRUE). If TRUE, check whether it is a fatal error (go to step 4) or a non-fatal error (go to step 3).
3. If a non-fatal error occurs, CC09, it means that the transfer is DONE. Every transaction is ACK acknowledged. The toggle bit in the present PTD is correct. This toggle can be copied to the start of the next PTD. Read back all the data that is shown by the Actual Byte in the PTD to the application layer.
4. If fatal error occurs and the active bit is also 0, then the toggle bit in this PTD is not correct for the next PTD. The actual byte points to the correct payload to be safely delivered back to the application software.
5. To start the next PTD, the toggle bit in the last erroneous PTD must be toggled before use. The last 64 bytes sent by the device will be sent again with the same toggle bit.

Table 5 Changes in the PTD header of Active, Toggle and Actual bytes

Handshake	Active bit	Is toggle bit toggled	Actual byte bits 9 to 0 updated	Comments
IN-token				
NAK	1	No	No	If NAKS for the whole ms
ACK, no error	0	Yes	Yes	–
Non-fatal error, CC09	0	Yes	Yes	Data received in the last payload is less than the defined MaxPacketSize.
Fatal error	0	Yes	No	Address of the last payload; MaxPacket size is not updated.
OUT-token				
NAK	1	No	No	–
ACK, no error	0	Yes	Yes	–
Non-fatal error, CC09	0	Yes	Yes	Not possible. Because software has control over the payload to send out.
Fatal error	0	Yes	No	Address of the last payload. MaxPacketSize is not updated.

Error handling software protocol

Transfer errors

Transfer errors are of four types for the isochronous and asynchronous data types:

- Transmission: Error when transmitting information on the USB
- Sequence: Error when the number of bytes received does not match the number of bytes expected
- System: Error when the host controller fails

The host controller ISP1161A1 driver must handle this error as the list processor is modified from the OHCI core to handle the embedded processors.

Error handling

- CRC

- BIT_STUFF
- DEV_NOT_RESPONDING
- PID_CHECKFAILURE
- DATA_TOGGGLE

For errors under these categories, USB defines a policy that allows the transaction to be retried for up to three times before the transaction is failed. This function of the modified list processor must be handled and corrected by the driver. The following dummy code explains the code flow. When the buffer is read from the ISP1161A1, if the ACTIVE bit is true; then you can trust the completion code of the TD.

Error processing mechanism for the ATL buffer

```
ErrorCount is set to ‘3’.

/* Read from ISP1161 internal ATL buffer in to the local buffer */

/* Process PTDs one by one */

for (lIndex = 1; lIndex <= LastPtd; lIndex++) {

/* Get the address of the ED associated with the current PTD */

if ((struct PTD ->ActiveBit & PTD_ACTIVE) == 0) {

/* Test if the PTD is still active */

if(!(--(Currenttd->ErrorCount )) || (CompletionCode !=
TD_PIDCHECKFAIL

&& CompletionCode != TD_CC_CRC && CompletionCode !=

TD_CC_BITSTUFFING && CompletionCode != TD_DEVNOTRESP

&& CompletionCode != TD_DATATOGGLE ))

{

        /* Processes the TD */

        /* Data Toggle */

        /* Get the data toggle bit from PTD in ATL */

        /* Set the toggle bit same as the PTD in ATL */

        /*******************************/

        /* Move TDs to done-queue list */

        /*******************************/

        /* Dequeue the current TD (the first TD in the TD queue
        from ED */
        /* Move the TD to the head of done queue */
/* Move data returned from device to the buffer for IN transfer */
}
elseif (CompletionCode = = TD_DATAUNDERRUN && CurrentHcd->shortpkt
= = TD_ACCEPT_SHORTPKT)
&& Direction = OHCI_IN {
update the buffer pointer
CompletionCode =TD_CC_NOERROR
}
Set the error for the upper driver to handle.
} else if (CompletionCode == CC5 && CompletionCode == CC6 ) {
        Get the toggle bit from PTD;
        Invert the toggle bit; (1->0 / 0->1)
        Copy the toggle bit to td and ed data structures;
        Get the total number of bytes trasnfered/received from
        ptd;
        if(PTD is IN ptd) Copy the received bytes to the TD
        buffer;
        update the td->hwCBP to td->hwCBP + bytes
        received/transfered;
} // Note that the same processing will be required in the case of
the BIT_STUFF and CRC
DATAOVERRUN Implement depending on the requirement */
```

***Note**: Data toggle must also be toggled in the case of errors.*

Error processing mechanism for the ISO buffer:

```
/* Read buffer, from the ITL buffer to the CPU local memory */
/* Process PTDs one by one */
if((struct PTD ->ActiveBit & PTD_ACTIVE) == 0) {
/* Test if the PTD is still active */
for (iIndex = 1; iIndex <= LastPtd; iIndex++) {
If ( ( Direction == OHCI _IN & CompletionCode == TD_CC_NOERROR) |
(Direction == OHCI _IN & CompletionCode == TD_DATAUNDERRUN) |
(Direction == OHCI _IN & CompletionCode == TD_DATAOVERRUN))
{
```

```
/* do not care the data toggle */

/* get the actual number of bytes transferred */

/* Write the actual amount of data transferred for ISO IN
transfers,

for out it is zero */

// Copy the data from the buffer to the application buffer ( URB
buffer )

}

}

}
```

Important: *The host controller driver must handle the missed data frame in case the CPU is busy and cannot attend to the interrupt to remove the data from an ITL, or put the data to the ITL buffer and then put the TD to the done queue by making the completion code TD_NOT_ACCESSED.*

System error

System errors are related to the overrun and underrun of the internal buffers of the ISP1161A1. They are very similar to the OHCI's. An IN token is not issued unless sufficient memory is available for data. Similarly, the data stage of the setup is not issued unless it fetches all the information required to complete the transaction.

3.4 ISP1161A1 reports error

How should the software driver respond if the ISP1161A1 reports an error?

In the OHCI architecture, when a transmission error occurs, the TD must be tried up to three times. There is an error counter in TD. If the error counter = 3, the TD will be returned back to the upper layer drivers.

Note: *These three errors must occur consecutively. For example, if the ISP1161A1 reports an error for an IN token, the error counter is set to 1. In the next frame, if there is no error reported for this IN token, the error counter will be cleared to zero.*

Glossary

ACK	Acknowledgement
HC	Host Controller
ITL	Isochronous (ISO) Transfer List
OHCI	Open Host Controller Interface
PCBA	Printed Circuit Board Assembly
PMOS	Positive-Channel Metal Oxide Semiconductor
PTD	Proprietary Transfer Descriptor
TD	Transfer Descriptor
USB	Universal Serial Bus