



# AN10058

## ISP116x PTD Programming Guide

Rev. 01 — 7 February 2006

Application note



### Document information

Info	Content
<b>Keywords</b>	isp1160; isp1160/01; isp1161a; isp1161a1; usb, universal serial bus; host controller
<b>Abstract</b>	<p>This document explains how to write and manage PTDs on the ISP116x for control or bulk and interrupt transfers.</p> <p><b>Remark:</b> ISP116x represents Philips ISP1160; ISP1160/01; ISP1161A and ISP1161A1 Host Controllers.</p>

## Revision history

Rev	Date	Description
01	20060207	First release.

## Contact information

For additional information, please visit: <http://www.semiconductors.philips.com>

For sales office addresses, please send an email to: [sales.addresses@www.semiconductors.philips.com](mailto:sales.addresses@www.semiconductors.philips.com)

## 1. Introduction

This document explains how to write and manage the Philips Transfer Descriptors (PTDs) on the ISP116x for control or bulk and interrupt transfers.

## 2. Overview

The communication between the Host Controller Driver (HCD) and the ISP116x Host Controller (HC) is in the form of PTD data. The PTD data provides Universal Serial Bus (USB) traffic information about commands, status and USB data packets.

The HCD prepares the PTD data in the microprocessor system RAM for transfer to the ISP116x HC internal FIFO buffer RAM, which is a first-in first-out serial buffer RAM. The HC determines what USB transactions are required based on the PTD data that is transferred into the internal FIFO buffer RAM. The HC performs USB transactions with the specified USB device endpoint through the USB bus interface. The USB transaction status and feedback from the specified USB device endpoint will then be placed in the ISP116x HC internal FIFO buffer RAM in the PTD data format. The HCD can read back the PTD data from the internal FIFO buffer RAM.

If you have multiple PTDs that must be written to the ISP116x, the HCD must update the HcATLBufferLength register with the total number of bytes or counts (multiple PTD length + multiple payload length). Start writing each PTD, followed by payload in the HcATLBufferPort register.

While reading back, write the count in the HcATLBufferLength register and start reading back the data.

### 2.1 PTD

[Table 1:](#) explains the PTD structure in the ISP116x. The PTD is 8 B long.

**Table 1: PTD structure in the ISP116x**

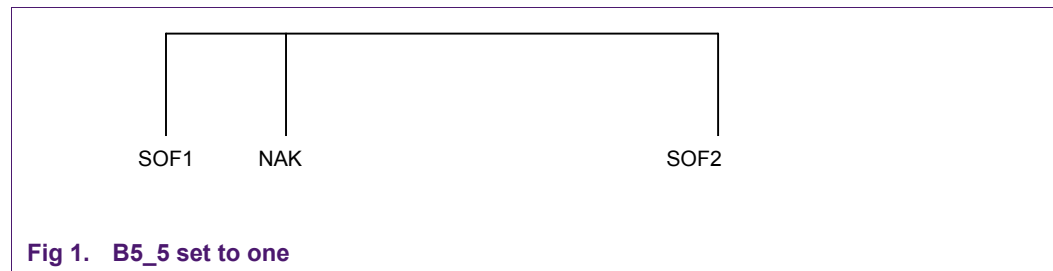
Bit	7	6	5	4	3	2	1	0
Byte 0	ActualBytes[7:0]							
Byte 1	CompletionCode[3:0]				Active	Toggle	ActualBytes[9:8]	
Byte 2	MaxPacketSize[7:0]							
Byte 3	EndpointNumber[3:0]				Last	Speed	MaxPacketSize[9:8]	
Byte 4	TotalBytes[7:0]							
Byte 5	reserved		B5_5	reserved	DirectionPID[1:0]		TotalBytes[9:8]	
Byte 6	Format				FunctionAddress[6:0]			
Byte 7	reserved							

For details on PTD fields, refer to the ISP1160, ISP1161A or ISP1161A1 data sheet.

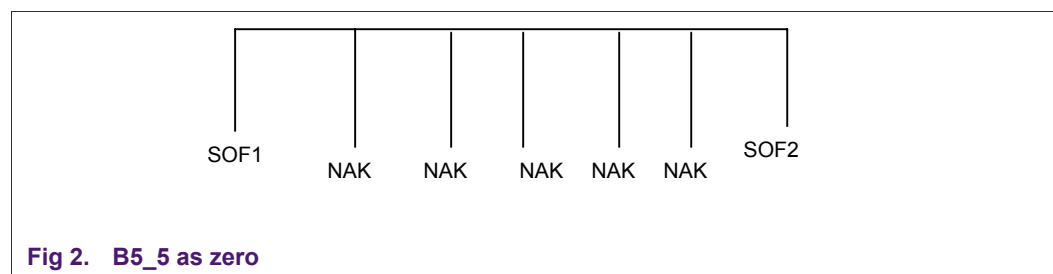
## 2.2 B5\_5

The difference between a control or bulk and interrupt PTD is only in the B5\_5 bit. For the control or bulk PTD, bit B5\_5 is kept as zero. For the interrupt PTD, this bit is written as one. If you do not set bit B5\_5 for the interrupt PTD, the HC will retry the token on the bus for the entire frame, even if the device NAKs at the first attempt. This means that the USB bus is not properly utilized because you do not need to retry the token for the interrupt PTD in the same frame.

With bit B5\_5 set, the PTD will be tried only once in a frame as shown in [Fig 1](#).



With bit B5\_5 as zero, the USB bus looks as given in [Fig 2](#).



## 2.3 PTD handling SOF driven

The important fields to consider in a PTD are:

- Active (A)
- CompletionCode (CC)
- ActualBytes (AB)
- TotalBytes (TB)

The ATL Buffer Done bit (D) in the HcBuffer Status register is also important to manage PTDs.

After the PTD is written to the ATL memory, the HC will try the PTD in the next millisecond frame. When an SOF interrupt occurs in the next millisecond, check ATL Buffer Done (D). If the Done bit is not set, do not try to read the PTD from the ATL memory. Read the PTD only when the Done bit is set.

## 2.4 Single control or bulk PTD handling

This section explains handling a single control or bulk PTD.

**Case A: D → 1, A → 0, CC → 0 and AB == TB**

Read the Buffer Status register and if the Done (D) bit is set, then read the PTD and the payload. If the Active (A) bit is zero, then check Completion Code (CC). If CC → 0, then

check ActualBytes (AB) and TotalBytes (TB). If  $AB == TB$ , it means that the PTD is done and software can process data.

**Case B:  $D \rightarrow 1$ ,  $A \rightarrow 0$ ,  $CC \rightarrow 0$  and  $AB \neq TB$**

Read the Buffer Status register and if the Done bit is set, then read the PTD and the payload. If the Active bit is zero, then check CC, AB and TB. If  $AB \neq TB$ , then update the TB field with  $TB - AB$  and rewrite the PTD in the ATL memory with the data toggle bit inverted and A set to 1. This will result in the PTD being retried.

**Case C:  $D \rightarrow 1$ ,  $A \rightarrow 0$ ,  $CC \neq 0$**

Look for the standard USB errors and handle accordingly. If there is any value in the AB field, then update the TB field with  $TB - AB$ .

**Case D:  $D \rightarrow 1$ ,  $A \rightarrow 1$ ,  $CC \rightarrow 0$  and  $AB \neq TB$**

Read the Buffer Status register first and if the Done bit is set then read the PTD and the payload. If the Active bit is still set, this means that the PTD is still active. Check CC, AB and TB. If  $AB \neq TB$ , then update the TB field with  $TB - AB$  and rewrite the PTD in the ATL memory with the data toggle bit inverted and A set to 1. This will result in the PTD being retried.

**Case E:  $D \rightarrow 0$**

If the Done bit is not set, then do not try to read the ATL memory.

**Case F: PTD is NAKed for a full frame**

Even if a PTD is NAKed for a full frame, the Done bit will be set at the next frame interrupt and it falls in one of the preceding cases.

## 2.5 Multiple control or bulk PTD handling

If there are multiple devices connected, there will be multiple PTDs present in a millisecond frame.

**Case A:** Consider that there are two PTDs in the buffer and two are done. Then in the next millisecond the Done bit will be set and it will fall into one of the cases mentioned in [Section 2.4](#).

**Case B:** Consider that there are two PTDs in the buffer. Also consider that device A's PTD is done and device B's PTD is NAK, then the A bit for the device A will be zero and the A bit for the device B will still be set. In the next millisecond interrupt, the Done bit will be set and you can read all the PTDs. Software should process A PTD because it has been successfully completed. As for B PTD, the software has to retry the PTD.

**Case C:** Consider that two devices continuously NAK. In the next millisecond interrupt, the Done bit will be set and you can read all the PTDs with payload. It will fall into one of the cases mentioned in [Section 2.3](#).

## 2.6 Adding a new PTD

Consider that device A is running on HC port 1 and you connect device B on port 2. In this case, irrespective of the status of the device A's PTD, the Done bit will be set in the following millisecond and you can add the new PTD for the device B.

## 2.7 Interrupt PTD handling

As mentioned earlier, the only difference between the control or bulk and interrupt PTD is in bit B5\_5.

**Case A:** Consider that there is only an interrupt PTD in the ATL buffer and the device is NAKing this PTD. In the following millisecond, you will still see the Done bit set and the A bit will be zero, follow the cases mentioned in [Section 2.4](#).

**Case B:** If the device responds with data, you read the PTD and proceed accordingly.

## 2.8 Interrupt and control or bulk PTDs

The PTD handling in this case is similar to that in [Section 2.5](#).

## 2.9 PTD handling ATL driven

The ATL driven software avoids a lot of CPU intervention. If there is no device connected or if there is no data transfer, the ATL driven software does not need any CPU resources because there will not be any millisecond SOF interrupt.

### 2.9.1 Dummy PTD

The ATL driven software needs a dummy PTD to be appended at the end of original PTDs + payload, for every millisecond. **A dummy PTD is a PTD header with the Active bit as zero.**

The ISP116x will generate an ATL interrupt only when there is a PTD with the A bit as zero. Therefore, the dummy PTD helps to acquire the control of the ATL buffer once the original PTD is placed.

Consider that multiple devices are connected, for example, X devices, and that you need to write X PTDs + payload. Then in an ATL driven software, you will write X + 1 PTDs, that is, the total count that you write to the HcATLBuffer Length register must include the dummy PTD length (8 B) as well.

## 3. Abbreviations

Table 2: Acronym

Acronym	Description
ATL	Asynchronous Transfer List
CPU	Central Processing Unit
FIFO	First-In First-Out
HC	Host Controller
HCD	Host Controller Driver
PTD	Philips Transfer Descriptor
RAM	Random Access Memory
SOF	Start-Of-Frame
USB	Universal Serial Bus

## 4. Disclaimers

---

**Life support** — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

**Right to make changes** — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no

licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

**Application information** — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

## 5. Trademarks

---

**Notice** — All referenced brands, product names, service names and trademarks are the property of their respective owners.

## 6. Contents

---

<b>1.</b>	<b>Introduction .....</b>	<b>3</b>
<b>2.</b>	<b>Overview .....</b>	<b>3</b>
2.1	PTD .....	3
2.2	B5_5 .....	4
2.3	PTD handling SOF driven .....	4
2.4	Single control or bulk PTD handling .....	4
2.5	Multiple control or bulk PTD handling .....	5
2.6	Adding a new PTD .....	5
2.7	Interrupt PTD handling .....	5
2.8	Interrupt and control or bulk PTDs .....	6
2.9	PTD handling ATL driven .....	6
2.9.1	Dummy PTD .....	6
<b>3.</b>	<b>Abbreviations .....</b>	<b>6</b>
<b>4.</b>	<b>Disclaimers .....</b>	<b>7</b>
<b>5.</b>	<b>Trademarks .....</b>	<b>7</b>
<b>6.</b>	<b>Contents.....</b>	<b>8</b>



© Koninklijke Philips Electronics N.V. 2006

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or other industrial or intellectual property rights.

Date of release: 7 February 2006

Published in The Netherlands