PHILIPS

Philips Semiconductors

Connectivity and Interoperability Solutions

1 October 1998

FAQ – PDIUSBD12

1. General Product Info

1.1 What is the difference between PDIUSBD11 and PDIUSBD12 ?

PDIUSBD11 offers an I^2C interface to any micro-controller. The operating speed of the I^2C interface can be up to 1Mbit/sec. It has a total of 4 endpoints, including the default endpoint 0. Each of the endpoints are bi-directional with a buffer size of 8 bytes. The 8 bytes buffer size and the simple I^2C connectivity makes it suitable (not restricted) for applications such as Multi-function Keyboards, high-end Joystick (forced feedback), Monitor Control and other HID-based systems.

The PDIUSBD12 on the other hand offers a generic parallel interface to any micro-controller. This allows faster access time of up to 2 Mbyte/sec. There are 3 bi-directional endpoints including the default endpoint 0. The buffer size is 16 bytes for endpoint 0 and 1, and a double buffered 64 bytes (Bi-directional mode) for endpoint 2. Endpoint 2 is also configuration to work as an IN only endpoint or a OUT only endpoint. In addition, Endpoint 2 can be used for Isochronous, Bulk or Interrupt operation. The D12 would be appropriate to be used in Scanners, Printers, Digital Cameras, Modems, Telephony.

1.2 How much current does the PDIUSBD12 consume ?

During normal operation, PDIUSBD12 consumes 15 mA. In suspend mode, the PDIUSBD12 internally shuts off blocks that are not essential. This allows an operating current of 15 uA during suspend. This is particularly important for bus-powered system as the USB Specifications requires the suspend current to be 500 uA or less. Also refer to FAQ 3.1.

PDIUSBD12 offers bus-powered capability as it can go into deep sleep drawing only 15 uA. This feature is not included in USB interface ICs from other Vendors.

1.3 Is PDIUSBD12 compliant to USB Specifications 1.0 or the 1.1 ?

The USB Spec 1.1 was released dated 23 September 1998. As expected, the USB 1.1 only clarifies the on the USB Spec 1.0. Much information on the timing on the protocol layer has been included. This may impact the firmware on the micro-controller, however does not affect the compliance of D12 to the 1.1 Specifications.

All the physical layer specifications have been re-checked to ensure compatibility to the USB Spec 1.1.

(The USB Specification 1.1 (Draft) has been released on 28 July 98 and the USB Specification 1.1 is expected to be out in September 1998. PDIUSBD12 is only used as the physical layer (Chapter 7) and basic protocol layer (Chapter 8.1 through Chapter 8.4.5) interface. It is compliant to both USB 1.0 and 1.1(Draft) Specifications. The handshake responses are handled through the micro-controller. This allows PDIUSBD12 to be upwards compatible to changes on Chapter 8.4.5 through Chapter 9.)

2. On Power up

2.1 What is the suspend output on power up?

The suspend pin is low right after PDIUSBD12 is powered up.

2.2 What is the default clock output on power up?

The default clock output frequency is 4 MHz.

2.3 What's the suggested time constant of the reset circuits?

The PDIUSBD12 has built-in power-on-reset circuit; so the RESET_N pin can be simply connected to Vcc.

3. Suspend

3.1 What are the suspend current ratings for PDIUSBD12?

The USB Specifications requires bus-powered devices to draw less than 500 uA during suspend mode. To meet this stringent requirement, the PDIUSBD12 shuts off non-essential internal blocks when in suspend mode. This significantly reduces the current ratings to a maximum of only 15 uA. This allows more margin for the rest of the hardware to meet the 500 uA requirement in a typical bus-powered system.

In addition, PDIUSBD12 (D12) supports remote wakeup. A peripheral using D12 can initiate a system wakeup as well.

Note : The 15 uA does not include the mandatory pull-up resistor on the D+ line which adds 200 uA on all USB devices. Hence, in total, using the PDIUSBD12 as a USB front-end would consume a maximum of 215 uA on suspend. The actual measured value is 202 uA. Also refer to FAQ 1.2.

3.2 When does the system go to suspend?

The host requests that you go to suspend, or, when the host itself is in suspend, then, the USB lines are in idle mode. This means that there's no activity on the USB bus. When this happens, the PDIUSBD12 detects the absence of 3 consecutive Start Of Frames (SOF) and pulls the Suspend pin high. The corresponding suspend bit in the Interrupt Register is also set.

3.3 How does the system get out of suspend?

There is two ways in which the USB device can get out of suspend. Host initiated or device initiated.

Host Initiated :
When the host recovers from a suspend state, the USB traffic becomes active again through the SOF every millisecond. The interrupt line from the PDIUSBD12 becomes active-low to indicate that there has been a change of condition on Suspend. This may be used to generate a wakeup to your micro.

Device Initiated :
To wake up the system using PDIUSBD12, you can use the Send Resume command. This would toggle the D+/D- lines to send a resume signal to the host.

3.4 What is $V_{out3.3}$ on suspend?

On suspend, this value drops to 2.0V, however, it is still capable of supplying 10 mA of current.

3.5 What is the CLKOUT frequency during suspend ?

The behaviour of the output clock is configured based on the Configuration Byte written using the Set Mode Command (0xF3).

Configuration Byte		CLKOUT
No Lazy Clock	Clock Running	
0	0	CLKOUT switches to Lazy Clock on suspend. The output frequency is 18KHz to 48 KHz. The PLL clock switched off to reduce current consumption.
1	0	The CLKOUT stops on suspend.
0	1	CLKOUT switches to Lazy Clock on suspend. The output frequency is 18KHz to 48 KHz. The PLL clock remains on.
1	1	The suspend state does not affect the CLKOUT frequency with this configuration

4. Clocking

4.1 What is the available clockout frequency ?

The clockout frequency can be set via the Clock Divison Factor using the Set Mode Command (0xF3).

The output frequency is based on the equation here :

CLKOUT = 48/(N+1) MHz, where N – Clock Division Factor

N (Clock Division Factor)	**CLKOUT**
0x00	48 MHz (Maximum clocking frequency of PDIUSBD12)
0x01	24 MHz
0x02	16 MHz
0x03	12 MHz
0x04	9.6 MHz
..	..
0x0A	4.36 MHz
0x0B	4 MHz (Power up value, Default CLKOUT frequency, Minimum Clocking frequency)

4.2 What is the suspend CLKOUT frequency ?

Refer to Section on Suspend.

4.3 What is the Start-up time of the CLKOUT ?

CLKOUT frequency is derived from the oscillator (XTAL1, XTAL2 pins). The start-up time for CLKOUT depends on the start-up time of the crystal oscillator. This has been measured to be typically below 2 ms. Since some microcontrollers (for example the 8051 family) use synchronous reset, the reset circuitry has to be designed to be active for 2 ms to be reset properly.

4.4 What is the Clocking $V_{peak\text{-}to\text{-}peak}$ that be fed into the Xtal1 pin ?

The Clocking voltage can only take a peak-to-peak voltage of 3.3V as the internal oscillator is built on a 3.3V core. Hence, to use a 5V external oscillator, it needs to be tied to the Xtal1 pin through a 200 kohms resistor.

5. Interfacing

5.1 What is the fastest transfer speed achievable on the parallel interface ?

There should be a minimum of 500 ns delay between two consecutive Read or Write phases. The RD_N and WR_N are to be held low for at least 20 ns for each Read or Write respectively. This is valid for both DMA and non-DMA mode.

This transfer speed across the parallel interface should be well above the transfer speed of a generic micro-controller.

For a high speed RISC or micro-controller that matches the T_{WL} (Wr_N low pulse width) and the T_{RL} (Rd_N low pulse width), care must be taken to ensure that the 500 ns T_{WC} (Write Cycle time) is observed. This can be accomplished through NOP (No Operation) delays between two consecutive Read/Write in the firmware.

5.2 How does the DMA work when data is to be transferred from the Host to the Device?

When PDIUSBD12 is set to the DMA mode; on receipt of a full packet of data from Endpoint 2, the DMReq is asserted to allow the DMA controller to retrieve the data from the PDIUSBD12’s Internal Buffer. The Read_N of the PDIUSBD12 is to be asserted by the DMA Controller. The PDIUSBD12 does not have an internal buffer to keep track how many bytes have been transferred via DMA.

5.3 How does the DMA work when data is to be transferred from the Device to the Host?

When PDIUSBD12 is set to the DMA mode, there are two conditions in which data from the buffer will be transferred to the host on an IN token.

When the internal buffer of Endpoint 2 is full (64 bytes), or,
When the EOT_N signal is asserted on the last packet that is transferred via DMA.

The DMReq is asserted once the internal buffer is empty and when the DMA enable register is set. Data is sweep to the internal buffer when the DMACK_N and DMREQ are both active at a WR_N.

The EOT_N signal is to generated by an external DMA Controller when the data count goes to zero. In the event that EOT_N signal is not present, it is recommended to have an external counter to generate EOT_N at the last packet.

5.4 What is the difference between a burst DMA Mode and a single DMA transfer mode ?

The behaviour of the DMREQ and DMACK_N are different for a single and burst mode. The specifications of the PDIUSBD12 at page 18 gives a graphic depiction between the Single and Burst mode.

In a Single DMA transfer mode, the DMREQ is asserted for every RD_N or WR_N strobe. Therefore, the number of bytes being transferred can be counted based on the falling edge of the DMREQ.

In a burst DMA mode, the DMREQ is asserted through the burst length which is defined based on the DMA Burst information given in the Set DMA Command (0xFB).

DMA Configuration Register		Remarks
Bit 1	**Bit 0**	
0	0	Single DMA Transfer mode
0	1	Burst DMA mode. 4 bytes of data is transferred on every assertion of DMREQ unless prematurely ended by EOT_N.
1	0	Burst DMA mode. 8 bytes of data is transferred on every assertion of DMREQ unless prematurely ended by EOT_N.
1	1	Burst DMA mode. 16 bytes of data is transferred on every assertion of DMREQ unless prematurely ended by EOT_N.

5.5 Does the PDIUSBD12 take 5V or 3.3V inputs?

The PDIUSBD12 can take either 5V or 3.3V inputs. To operate the IC at 5V, supply a 5V voltage to Vcc pin only and left Vout3.3 pin open (decoupled with capacitor). To operate the IC at 3.3V, supply a 3.3V voltage to both Vcc and Vout3.3 pins.

5.6 What is the output voltage swing?

There is typically two output types on the PDIUSBD12: the open drain and normal driving output. The driving strength for each of them is specified on the datasheet. The voltage swing of an open-drain output is dependent on the pull-up resistor on which it is tied to. For the other output pins, they are listed as follows:

When the V_{cc} pin of PDIUSBD12 is powered from 5V.

Pin Name	Type	Description	Voltage Swing
Data<0..7>	IO2	Input/Output with driving strength of 2 mA	5V
Suspend	OD4	Open-Drain, can sink 4 mA	Depends on pull-up
Clkout	O2	Output with 2mA drive	5V
Int_N	OD4	Open-Drain, can sink 4 mA	Depends on pull-up
GL_N	OD8	Open-Drain, can sink 8 mA	NA
DMREQ	O4	Output with 4mA drive	5V

***When the V_{cc} pin of PDIUSBD12 is powered from 3.3V.**

Pin Name	Type	Description	Voltage Swing
Data<0..7>	IO2	Input/Output with driving strength of 2 mA	3.3V
Suspend	OD4	Open-Drain, can sink 4 mA	Depends on pull-up
Clkout	O2	Output with 2mA drive	3.3V
Int_N	OD4	Open-Drain, can sink 4 mA	Depends on pull-up
GL_N	OD8	Open-Drain, can sink 8 mA	NA
DMREQ	O4	Output with 4mA drive	3.3V

Note : *When the V_{cc} of the PDIUSBD12 is powered from 3.3V, the internal regulator shuts off. All voltage regulation should be done externally. Therefore, a 3.3V source should be applied to both Vcc and $V_{out3.3}$ pins.

5.7 Can I make use of the $V_{out3.3}$ to drive other parts of my circuit?

The Vout3.3 pin is provided to supply the pull-up voltage of the 1.5K resistor. However, user can also opt to use the internal SoftConnect resistor. Loading the Vout3.3 pin apart from this resistor is not recommended.

5.8 Can the CS_N and the DACK_N be active at the same time ?

The user should not be writing or reading by asserting CS_N during DMA when DACK_N is active.

5.9 Should I used level trigger or edge trigger on PDIUSBD12 ?

The PDIUSBD12's interrupt pin remains low as long as the Interrupt register is non-zero. Therefore, the micro-controller should be configured to be level-trigger on the interrupt.

5.10 How should the EOT_N be connected to accomplish a Vbus sensing functionality ?

In a self-powered system, when the USB connection has been removed, there may not be any indicator to the micro-controller. To detect this, the EOT_N pin has dual functionality to detect whether the Vbus is there. (The Vbus is the 5V power supply pin from the USB Connector). Thus, the EOT_N is connected to the Vbus via a 1 k resistor. A 1 M ohms bleeding resistor may be needed to leak the charges to ensure that EOT_N goes to zero when Vbus has been removed.

5.11 How do you use the ALE pin of the PDIUSBD12 ?

When the ALE of D12 is connected to the ALE pin of the micro-controller and the address/data bus is multiplexed, this pin is used by the internal logic to strobe in the information to differentiate between a command and data on the parallel bus. Thus, communicating to D12 will mean that an even address means writing/reading data to D12 and an odd address means writing a command to D12 (The CS_N has to be pulled low during data communications as per normal).

A0 will not be used and should be tied high in this instance.

5.12 Can the CS_N pin of the PDIUSBD12 be tied to ground all the time ?

The PDIUSBD12 is to be treated like any other microprocessor based device. CS_N is connected so that PDIUSBD12 can share the system bus with other devices. In some instance, the CS_N may be grounded all the time. For instance, when D12 is the only device residing on the system bus. If the system bus is shared, additional circuitry may be required to separate PDIUSBD12 from other resources via Rd_N/Wr_N if CS_N is to be grounded.

Example : To glue the MC68331 with PDIUSBD12, the Trhch/Twhch timing for the CS output need to be delayed to match the timing of D12. In this instance, CS_N of D12 can be grounded, and the Rd_N/Wr_N may all be all you need. To separate other devices that shares the same bus, a glue logic can be implemented on Rd_N/Wr_N that is going to D12.

6. Programming PDIUSBD12

6.1 What is SoftConnect ?

The SetMode Register at 0xF3 has a bit which ties directly to the pull-up resistor on the D+ USB line. When the bit is 1, this means that the pulled-up resistor is enabled. Thus, a host/hub will detect that something has been plugged onto it’s USB port even though it has been physically connected before the command was issued.

The beauty of using SoftConnect is that it allows the micro-controller to finish it’s initialisation routine first before it notifies the Host of it’s presence. This is especially valuable in a bus-powered device where the 5V power supply needs to be stable first before enumeration.

To force the Host to do a re-enumeration, the micro-controller could initial a Soft Disconnect by setting the SetMode SoftConnect bit to 0. In doing so, the Host is forced to reload the Host device driver. This allows a device initiated upgrade without user to disconnect and connect the USB cable.

6.2 What is the difference between Set Address/Enable and SoftConnect ?

The SetAddress/Enable is required to enable the SIE to respond to the USB request that is directed towards the address preset via SetAdress/Enable. Without enabling, PDIUSBD12 will not respond with the “ACK” or “NAK” token, even though the request is directed to it’s preset address.

7. Others

7.1 What is double buffering ?

Double buffering on Endpoint 2 allows data to be source/sink on the USB bus while the internal buffer is being read/written by the micro-controller or the DMA controller. This increases the overall throughput as the Host does not need to wait for the Internal buffer to be cleared or filled before feeding or extracting the next packet.

When data is to be extracted from the USB Device to the host, the switching from one buffer that was filled up by the micro-controller to the sending buffer at the USB end is done transparent to the micro-controller. The micro-controller do not have to keep track of which buffer to use, as it always uses the same register to acess the IN buffer.

When data is to be received from the Host to the USB Device, the switching from one buffer that was read by the micro-controller to the receiving buffer at the USB end is done transparent to the micro-controller. The micro-controller do not have to keep track of which buffer to use, as it always uses the same register to acess the OUT buffer.

7.2 What is the Internal Buffer size of PDIUSBD12 ?

The total bytes of Internal Buffer dedicated for USB transfer is 320 bytes.

Total Bytes		Endpoint 0		**Endpoint 1**		**Endpoint 2**
320	=	16 (IN) + 16 (OUT)	+	16 (IN) + 16 (OUT)	+	[64 (IN) +64 (OUT)] x 2 (Double Buffered)

7.3 Is there any EMI issues I should take care of?

EMI issues are too broad a subject to cover. In general, ferrite beads are to be added on the Vbus and the Ground at the input side of the USB connector. One such part is: BLM32A07.

It is recommended to have capacitive coupling from the USB Shield to the electrical ground.

7.4 What resistor value is recommended for the GoodLink LED (GL_N)?

This depends on how bright the user wants the LED to be and also the brightness/current rating of the LEDs. Any value from 100 to 1K ohm is normally OK. The evaluation kit recommends a value of 470 ohm.

7.5 What is the Vendor ID and Product ID ?

The Vendor ID identifies the manufacturer of the USB product. It is used to load the INF file which contains the text string of the Manufacturer and information of which device-driver to load on Windows 98.

The Vendor ID can be obtained by registering with the USB-IF. More information can be obtained on the website at WWW.USB.ORG. The Vendor ID (VID) for Philips Semiconductors is : 0471. The Product ID (PID) is unique to each USB product.

The VID and PID can be set by simply changing the Device Descriptors on the firmware. This is good for product differentiation as customers maintain their own product identity on Windows 98.

7.6 Why is the PDIUSBD12 implemented based on a 6 MHz crystal ?

The 6 MHz crystal lowers the risk of having EMI problems later in the production process.

7.7 Why does the Philips Test Application, D12Test.exe, reports a varying data transfer rate ?

The calculation of the real time transfer rate is made on every blocks of 16 Kbytes. The actual derivation formula used is :

Transfer speed (Bytes/s) = 16*1024/Time Spent (seconds).

Where the Time Spent is the completion time of transfer minus the initiated time. The Windows 98 operating system provides a 1 millisec resolution on the Time Spent. In additional, there is random overhead incurred due to system calls from user-mode to kernel mode. All in all, a 2 ms ambiguity would put the variance of the transfer rate to be about 20%.

Whatever the reported transfer rate, it is often the situation that the Host is the bottleneck. It can be verified using a CATC Analyzer that the PDIUSBD12 is fully utilizing the allocation provide by the Host.

7.8 What is the Allocation difference between Isochronous and Bulk Pipe ?

The Isochronous pipe guarantees bandwidth regardless of Data Integrity. A Bulk pipe guarantees data integrity, but, delivery is ruled based on “first require first allocate”. Hence, on a heavy USB traffic, the Bulk pipe may be starved.

For more enquiries, please contact : cis@sgp.sc.philips.com