Semiconductors

AN10010_2

ISP1181A/ISP1181B Frequently Asked Questions (FAQs)

Rev. 02.00 — 14 April 2004 Application note

Document information

Info	Content
Keywords	ISP1181A, ISP1181B, FAQ, USB
Abstract	This document is a compilation of frequently asked questions (FAQs) on Philips full-speed Universal Serial Bus peripheral controllers: the ISP1181B and the ISP1181A.

PHILIPS

Revision history

Rev	Date	Description
02	Apr 2004	- Updated Section 4.5; - Changed terminology of “interface device” and “device controller” to “peripheral controller”.
01	Mar 2002	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*

Contents

1. General Product Information

1.1. What are the differences between the ISP1181A/ISP1181B and the PDIUSBD12?

The ISP1181A/ISP1181B is a high-end Universal Serial Bus (USB) peripheral controller that complies with ***Universal Serial Bus Specification Rev. 2.0***, supporting data transfer at full-speed (12 Mbit/s). It offers a generic parallel interface with faster access speed than the PDIUSBD12, enabling you to connect it to any high-speed microcontroller or RISC processor.

Table 1-1 lists the major differences between the PDIUSBD12 and the ISP1181A/ISP1181B.

Table 1-1: Characteristic Differences between the PDIUSBD12 and the ISP1181A/ISP1181B

Characteristics	*PDIUSBD12*	*ISP1181A/ISP1181B*
Generic parallel interface	8-bit	8-bit or 16-bit
Interface	2 Mbyte/s	11.1 Mbyte/s
Physical FIFO memory size	312 bytes	2462 bytes
Number of endpoints	6 endpoints (including control IN and OUT)	14 configurable endpoints and 2 fixed control IN/OUT

1.2. Is the ISP1181A/ISP1181B compliant with USB 2.0, USB 1.1 or USB 1.0?

The ISP1181A/ISP1181B is used only as the physical layer and basic protocol layer interface. It is compliant with the USB 2.0 (full-speed), 1.1 and 1.0 specifications. The other handshakes are handled by the microcontroller.

2. Interfacing

2.1. What is the use of the V_{BUS} sensing input pin?

V_{BUS} is the 5 V-power supply pin from the USB connector. The ISP1181A/ISP1181B has a separate input pin to detect the presence of V_{BUS}. In a self-powered system, the ISP1181A/ISP1181B detects the removal of the USB cable through the V_{BUS} input pin. When V_{BUS} is lost, the ISP1181A/ISP1181B prepares itself to go in the 'suspend' state. The GOSUSP bit in the ***Mode*** register must first be set and then cleared to set the device to suspend. This changes the state of the SUSPEND output pin. The behavior of the SUSPEND output depends on the PWROFF bit in the ***Hardware Configuration*** register.

In a self-powered system with an internal pull-up resistor (SoftConnect™), when the host is powered down, V_{BUS} is lost. The chip automatically disables the internal pull-up resistor of 1.5 kΩ. This prevents any current flow from the device to the host. The detection of V_{BUS} is done by the V_{BUS} sensing pin.

In a self-powered system with an external pull-up resistor, when the host is powered down, the external pull-up resistor will still pull D+ HIGH. This causes current to flow from the device to the host. Designers must design a workaround circuit to prevent this current flow. There will be no such issues if you use the internal pull-up resistor.

2.2. What is the use of the $V_{reg(3.3)}$ pin?

The $V_{reg(3.3)}$ pin allows a regulated voltage supply of 3.3 V to the 1.5 kΩ pull-up resistor. You can, however, use the internal SoftConnect resistor instead. It is recommended not to load this pin, other than with the 1.5 kΩ resistor.

2.3. How does EOT work in ISP1181A/ISP1181B?

The End of Transfer (EOT) signal terminates the DMA transfer. In the ISP1181A/ISP1181B, the DMA transfer can be terminated in the following ways:

- External EOT signal
- Internal DMA Counter completes its count (when the ***DMA Counter*** register is enabled)
- DMA transfer finishes a short packet received on the OUT endpoint FIFO (when the short packet mode is enabled).

Disable DMA by writing DMAEN = 0 to the ***DMA Configuration*** register (*Note*: There will be no interrupt to indicate EOT).

All the preceding conditions are recognized by the ISP1181A/ISP1181B as EOT conditions.

If the external DMA controller needs to terminate the DMA transfer, it can perform an external EOT by sending the signal to the EOT pin on the ISP1181A/ISP1181B. The remaining three EOT conditions are caused internally.

If an OUT endpoint FIFO is selected for DMA transfer and there are data bytes remaining in the FIFO when the EOT condition occurs, the DMA operation is aborted and ***the remaining data in the FIFO is cleared***. (For a double-buffered endpoint, the data packet contained in another buffer is not affected by the current EOT). If an IN endpoint FIFO receives an EOT condition, the data packet that has been written to the FIFO—even shorter than the maximum packet size of the FIFO—will be sent to the host at the next IN token.

Normally, the transfer byte count must be set using a control endpoint before any DMA transfer takes place. When a short packet is enabled as EOT indicator (SHORTP = 1), the transfer size is determined by the presence of a short packet in the data. This mechanism permits the use of a fully autonomous data transfer protocol.

2.4. What are the different bus configurations and interfaces possible on the ISP1181A/ISP1181B?

To cater for various microcontrollers and microprocessors, the ISP1181A/ISP1181B allows versatile interface configurations. The bus configuration modes are selected using two pins, BUS_CONF1 and BUS_CONF0, see Table 2-1.

Table 2-1: Bus Mode Configuration

Mode	BUS_CONF[1:0]	PIO data width	DMA data width (DMAWD = 0)	DMA data width (DMAWD = 1)
0	00	DATA[15:0]	—	DATA[15:0]
1	01	reserved	reserved	reserved
2	10	DATA[7:0]	DATA[7:0]	—
3	11	reserved	reserved	reserved

Mode 0: The 16-bit I/O port shared with the 16-bit DMA port

Mode 2: The 8-bit I/O port shared with the 8-bit DMA port.

2.5. How do the input pins A0 and ALE function?

The input pin A0 functions as an address input pin. It selects a command when A0 = 1 and data when A0 = 0. In the multiplexed address and data bus configuration (that is, when the input pin AD functions as address A0 as well as bit 0 of DATA[15:0]), the input pin A0 is not used and must be connected to GND (LOW).The multiplexed address and data configuration works with reference to the input pin ALE.

In the separate address and data bus configuration, the ALE pin must be connected to GND (LOW). When the ALE pin is in use, it goes through these phases:

- *Address Phase*: A HIGH-to-LOW transition on the ALE pin latches the level on this pin as address A0 (1 = command, 0 = data).
- *Data Phase*: During reading, this pin outputs bit DATA[0]; during writing, the level on this pin is latched as DATA[0].

2.6. Does the ISP1181A/ISP1181B operate with 5.0 V or 3.3 V input?

The ISP1181A/ISP1181B can take either a 5.0 V or 3.3 V input. To operate the ISP1181A/ISP1181B at 5.0 V, connect the V_{CC} pin to the supply voltage 5.0 V and connect the other supply pins as shown in Figure 2-1 (with decoupling capacitors).

To operate the ISP1181A/ISP1181B at 3.3 V, connect the supply voltage to the V_{CC}, $V_{CC(3.3)}$, V_{ref} and $V_{reg(3.3)}$ pins along with a decoupling capacitor. This connection is as shown in Figure 2-2.

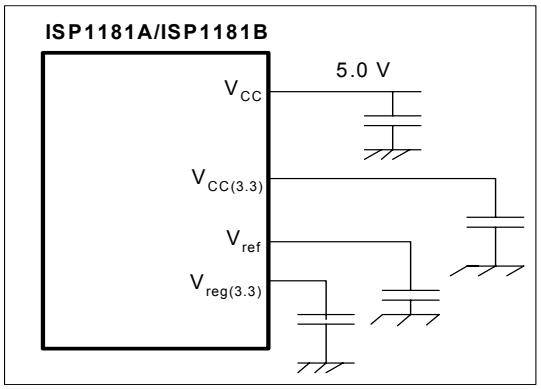


Figure 2-1: ISP1181A/ISP1181B with a 5.0 V Supply

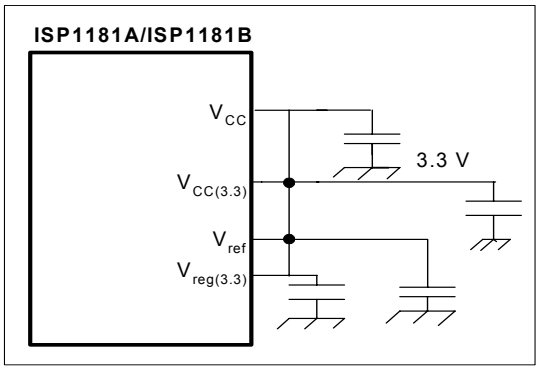


Figure 2-2: ISP1181A/ISP1181B with a 3.3 V Supply

2.7. What is the fastest speed achievable on the parallel interface?

In the 8-bit configuration, the minimum read/write cycle time is 90 ns, which amounts to a transfer speed of 11.1 Mbyte/s. For each read operation, the read pulse ($\overline{RD}$) must be held LOW for a minimum of 25 ns. For each write operation, the write pulse must be held LOW for a minimum of 22 ns.

While interfacing to any high-speed microcontrollers or RISC processors, make sure that the read and write pulse width and the read and write cycle time are taken into account. Add sufficient delays between two consecutive read and write in the firmware if the transfer speed of the microcontroller or the RISC processor is faster than the ISP1181A/ISP1181B.

3. Clocking

3.1. What is the type of crystal to be used for the ISP1181A/ISP1181B?

A 6 MHz-to-48 MHz clock multiplier Phased-Locked Loop (PLL) is integrated on-chip. This allows the use of a low cost 6 MHz (fundamental) crystal. This low value minimizes EMI. Table 3-1 shows the characteristics of the crystal that must be used for the ISP1181A/ISP1181A.

Table 3-1: Characteristics of the Crystal used for the ISP1181A/ISP1181B

Characteristics	*Value*	*Remarks*
Nominal frequency	6 MHz	fundamental
Frequency temperature stability	±50 ppm	–20 °C to +70 °C
Frequency tolerance	±50 ppm	+25 °C
Equivalent series resistance	<100 Ω	—
Load capacitance	18 pF	—
Shunt capacitance	< 7 pF	—
Operating temperature range	–10 °C to +70 °C	—
Storage temperature range	–45 °C to +80 °C	—
Maximum drive Level	0.1 mW (max.)	—

4. Suspend

4.1. When does the ISP1181A/ISP1181B enter the 'suspend' state?

The ISP1181A/ISP1181B detects the 'suspend' state in these three ways:

- A continuous J-state is present on the USB bus for more than 3 ms
- V_{BUS} is lost
- SoftConnect is disabled by clearing the SOFTCT bit in the ***Mode*** register, with the external pull-ups disabled by setting EXTPUL = 0 in the ***Hardware Configuration*** register. In this situation, the ISP1181A/ISP1181B is effectively disconnected from the USB bus.

Any of these three conditions will make the ISP1181A/ISP1181B initiate the SUSPEND process. This will set the SUSPND bit in the ***Interrupt*** register and will generate an interrupt (provided, the IESUSP bit in the ***Interrupt Enable*** register is set).

4.2. How does the ISP1181A/ISP1181B enter the SUSPEND mode after detecting the 'suspend' condition?

On detecting the 'suspend' condition, the ISP1181A/ISP1181B sets the SUSPND bit in the ***Interrupt*** register, which in turn generates the interrupt. Therefore, the firmware detects the 'suspend' condition and prepares all the system components to enter the 'suspend' state.

In the interrupt service routine, before initiating the 'suspend' state, the firmware must check the current status of the USB bus. If the BUSTATUS bit in the ***Interrupt*** register is logic 0, it indicates that the USB bus has left the 'suspend' state and the 'suspend' process must be aborted. Otherwise, the firmware will continue the process by first setting the GOSUSP bit in the ***Mode*** register and then clearing the bit. Once this is done, the ISP1181A/ISP1181B asserts the SUSPEND output and switches off the internal clocks after 2 ms (depends on the setting of the CLKRUN bit in the ***Hardware Configuration*** register). The behavior of the SUSPEND output depends on the PWROFF bit set in the ***Hardware Configuration*** register.

For bus-powered devices, to meet the stringent current requirements for the 'suspend' state, the internal clocks must be switched off by clearing the CLKRUN bit in the ***Hardware Configuration*** register.

4.3. How does the ISP1181A/ISP1181B resume from the 'suspend' state?

The ISP1181A/ISP1181B wakes up from the 'suspend' state either by the initiation of the USB host or by the application.

- *USB host*: When the host drives the K-state on the USB bus (global resume), or when there is bus activity on the USB bus
- *Application*: Remote wake-up through a HIGH level on the input pin WAKEUP or a LOW level on the input pin $\overline{CS}$ (wake-up on chip select is possible if the WKUPCS bit in the ***Hardware Configuration*** register is enabled).

4.4. What is the purpose of the 'Unlock Device' command?

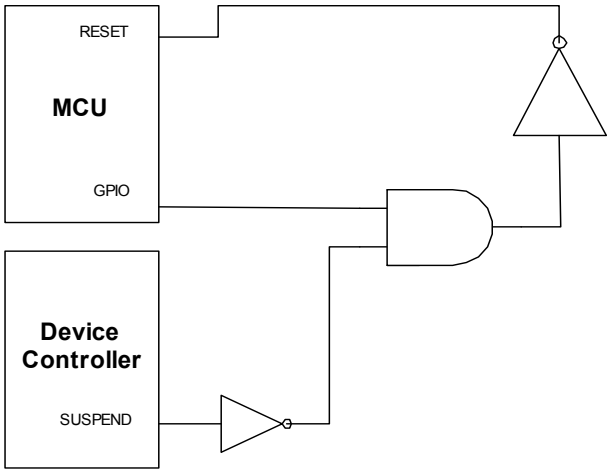
During the 'suspend' state, in a powered-off application, the power supply to the CPU and the other parts are cut off. Input pins are pulled to ground through pin buffers. Outputs are in three-state to prevent current flowing in the application. Bidirectional pins are made three-state (must be externally pulled to ground by the application). It is possible that the interface signals $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ have unknown values immediately after leaving the 'suspend' state because external components are also powered off. The uncertain value of these signals may corrupt the internal registers immediately after enabling the 'suspend' state. Therefore, to prevent corruption, the ISP1181A/ISP1181B enables a locking mechanism once the 'suspend' state is enabled.

After waking up from the 'suspend' state, all internal registers, except the Unlock register, are write-protected. A special unlock operation (using the Unlock Device command) is required to re-enable the write access to prevent data corruption during power-up of external components.

4.5. How can the SUSPEND pin be used for powered-off application in relation with the PWROFF bit?

The power ON state of the PWROFF bit is logic 0. This corresponds to a HIGH level on the SUSPEND pin when the Peripheral controller is not in the suspend mode. The PWROFF bit must be programmed to logic 1 that corresponds to a LOW level on the SUSPEND pin when Peripheral controller is not in suspend. See Figure 4-1.

- This work around assumes that the default state of the MCU GPIO is LOW.
- The MCU GPIO is LOW also after a reset.
- After initialization of the Peripheral controller, the SUSPEND pin of the Peripheral controller will normally become LOW.
- On receiving the suspend change interrupt from the Peripheral controller, the MCU suspends the Peripheral controller through the ***Mode*** register. This results in the SUSPEND pin of the Peripheral controller going HIGH.
- Before the MCU enters the low-power mode, the MCU sets the wake-up detection circuit by setting the GPIO pin HIGH and suspending itself.
- When wake up occurs, suspend becomes LOW and is inverted using an inverter. The GPIO and the inverted suspend signals will go through an AND operation, followed by an inverter to reset the MCU.



	Suspend pin	!	GPIO	Reset	!
Default state	1	0	0	0	1
Init DC PWROFF = 1	0	1	0	0	1
Suspend	1	0	0	0	1
ARM (GPIO)	1	0	1	0	1
Wake up	0	1	1	1	0
After MCU reset	0	1	0	0	1

Figure 4-1: Block diagram and truth table

Note: The circuit depends on deassertion of the GPIO to bring the MCU out of the suspend state. If the MCU resets or deasserts the GPIO during the LOW-level portion of the reset pulse (refer to dashed line A in Figure 4-2), the MCU would come out of the reset state. If the MCU were looking for the rising edge of the reset pulse to reset or deassert the GPIO (refer to dashed line B in Figure 4-2), the MCU will stay in a constant reset state. Therefore, the expected behavior of the MCU should be as shown at dashed line A. The application will not function if the behavior of the MCU is as shown at dashed line B.

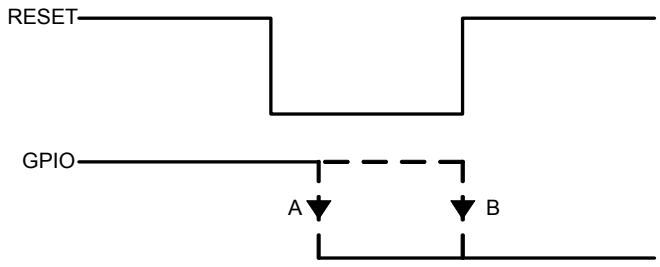


Figure 4-2: Reset of GPIO with respect to MCU

5. Others

5.1. What is the internal buffer size of the ISP1181A/ISP1181B?

The integrated physical buffer size is 2462 bytes that is shared among all the enabled USB endpoints. The size of the 14 endpoints can be independently configured through the ***Endpoint Configuration*** register, but the total physical size of all the enabled endpoints must not exceed 2462 bytes.

5.2. What is double buffering?

Double buffering allows data to be OUT or IN on the USB bus while the internal buffer is still being read or written by the microcontroller or the DMA controller. This feature increases the overall throughput because the host does not have to wait for the internal buffer to be cleared or filled before feeding or extracting the next packet.

In the ISP1181A/ISP1181B, double buffering is possible on all the 14 endpoints.

Double buffering can be enabled or disabled by setting the DBLBUF bit in the ***Endpoint Configuration*** register (in the device PDIUSBD12 double buffering is always enabled). Remember the physical size (2462 bytes), when allocating FIFO sizes for the double buffering feature for each endpoint.

5.3. What is GoodLink™?

Indication of a good USB connection is provided at pin $\overline{GL}$ through the GoodLink technology. During enumeration, the LED indicator will momentarily blink. When the ISP1181A/ISP1181B has been successfully enumerated (the device address is set), the LED indicator will remain permanently ON. During each successful packet transfer, the LED will blink OFF for 100 ms. During the 'suspend' state, the LED will remain in the OFF condition.

This feature provides a user-friendly indication on the status of the USB device, the connected hub and the USB traffic.

The LED can be connected to the $\overline{GL}$ pin (open drain, 8 mA) as shown in Figure 5-1. When the supply voltage V_{CC} is 5.0 V, the recommended series resistor 'R' must be 560 Ω. If the supply voltage V_{CC} is 3.3 V, then the series resistor 'R' can be 330 Ω.

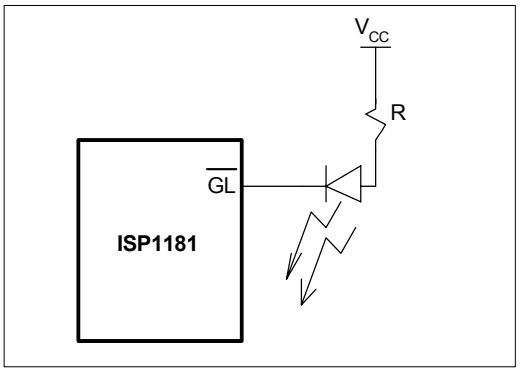


Figure 5-1: Connection of LED to the $\overline{GL}$ pin

5.4. What is SoftConnect?

You can enable the SoftConnect feature by using the SOFTCT bit in the ***Mode*** register. The SOFTCT bit is directly connected to the pull-up resistor on the D+ USB line. (The SOFTCT bit is ignored if the EXTPUL bit is set as logic 1 in the ***Hardware Configuration*** register). Setting the SOFTCT bit in the ***Mode*** register will enable the pull-up resistor of 1.5 kΩ that is required for the device detection. Therefore, the host or the hub will detect that something is plugged onto its USB port even though it was physically connected before setting up the SOFTCT bit.

SoftConnect allows the microcontroller or the processor to finish its initialization process before it notifies the host of its presence. This is especially valuable in a bus-powered device where the 5 V-power supply must be stable before the enumeration. Besides, the USB connect or disconnect scenario can be created by setting or clearing the bit SOFTCT. This forces the host to do a re-enumeration and reload the host device driver. This allows a device initiated upgrade without requiring you to physically disconnect and connect the USB cable.

6. Power up

6.1. How does the power-on reset circuitry work in the ISP1181A/ISP1181B?

The ISP1181A/ISP1181B has an internal power-on reset (POR) circuitry. In view of this, the $\overline{\text{RESET}}$ pin can be directly connected to V_{CC}. A POR is automatically generated when V_{CC} goes below the trigger voltage of 2.0 V for duration longer than 50 µs.

6.2. When is the device ISP1181A/ISP1181B accessible after power-on?

Upon power-on, the clock signal starts approximately after 0.5 ms and it normally requires 3 ms to 4 ms to stabilize. Therefore, it is better to start accessing the ISP1181A/ISP1181B 5 ms after power-on.

7. Design considerations

7.1. What are the basic design considerations that must be taken into account?

It is a very detailed subject to be covered. Nevertheless, some basic PCB design guidelines are:

- Try to maintain the (D+, D–) lines equal in length and width. It is better to keep the length of the (D+, D–) traces to the USB connector as short as possible.
- Keep the crystal oscillator very close to the device ISP1181A/ISP1181B. Keep the traces connecting to the XTAL1 and XTAL2 of the ISP1181A/ISP1181B as short as possible. Also, the two traces must be guarded with the ground lines.
- Keep the V_{CC} and GND signals as thick as possible; they must never be less than 20 mils (0.5 mm) in thickness.

7.2. What are the EMI issues that must be taken into account?

EMI and EFT issues are too broad to cover. Here are some guidelines:

- In general, ferrite beads can be added on V_{BUS} and GND at the input side of the USB connector. The recommended ferrite bead part number is *BLM32A07*.
- It is better to have capacitive coupling from the USB shield to the electrical ground.

7.3. How to suppress noise on the USB bus?

The USB bus performs differential transmission of signals. The USB 2.0 specification supports three transmission modes: 1.5 Mbps, 12 Mbps and 480Mbps, of which the ISP1181A/ISP1181B support the lower two.

It is very important to suppress noise without distorting signals, and it is necessary to match the noise suppression to the transmission mode.

Here are some basic guidelines:

- Add common-mode choke coil *PLP3216S221SL2* (220 Ω at 100 MHz, 100 mA) on the (D+, D–) lines.
- Add ferrite bead *BLM11B102S* on the (D+, D–) lines.

These values must be used only as a guideline. The actual values will vary depending on the PCB design, components, and so on. In addition, when you are choosing ferrite beads and choke coils, make sure that proper impedance values have been chosen for the design. For example, a larger impedance value has better noise suppression. Unfortunately, it has a severe effect on the signal quality.