Semiconductors

AN10037

Interfacing the ISP176x to the Intel® PXA25x Processor

Rev. 01 — 4 October 2004 Application note

Document information

Info	Content
Keywords	isp1760, usb, universal serial bus, pxa250, pxa255
Abstract	This application note explains interfacing the ISP176x to the Intel PXA250/255 processors. **Note**: The ISP176x denotes the ISP1760 and ISP1761 Hi-Speed Universal Serial Bus controllers, and any future derivative. **Note**: PXA25x denotes PXA250 and PXA255 processors.

PHILIPS

Revision history

Rev	Date	Description
01	20041004	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

Note: The ISP176x denotes the ISP1760 and ISP1761 Hi-Speed Universal Serial Bus controllers, and any future derivative.

Note: PXA25x denotes PXA250 and PXA255 processors.

1. Introduction

The ISP176x is a Hi-Speed Universal Serial Bus (USB) Host Controller that can be directly connected to a generic processor interface. The programmable CPU interface allows you to connect most of the RISC processors available without additional glue logic. The data bus can be configured as 16-bit or 32-bit, ensuring direct connection to most of the RISC processors.

The ISP176x is compatible with *Enhanced Host Controller Interface Specification Rev 1.0* and provides three USB ports that support high-speed, full-speed and low-speed modes.

The internal ISP176x architecture contains an internal hub. The three available USB ports are the downstream ports of the internal hub.

An internal Transaction Translator (TT) is implemented to support full-speed and low-speed modes. This allows a cost-effective and simple architecture, avoiding the need of OHCI companion controllers.

The ISP176x is a slave Host Controller This means that it does not need access to the local bus of the system to transfer data, unlike the case of the PCI Hi-Speed USB controllers. Therefore, the data to be transferred will be moved to or from the ISP176x internal memory by the system as described in the Section 2.

The access to the ISP176x is ‘memory-mapped’ and data transfers can be done using either programmed I/O (PIO) or direct memory access (DMA). It is highly recommended that you use DMA to avoid high CPU usage time.

The DMA of the ISP176x is ‘slave-type’ that can generate DREQ when enabled, with configurable burst size and maximum number of bytes transferred. If DREQ is not required by the system, the ISP176x can be accessed by the system’s DMA in PIO mode and the ISP176x DMA programming is not required. Programming the memory register is necessary only during memory read cycles.

Programming the ISP176x for enumeration and data transfer to an external device involves the programming of:

- The CPU interface configuration registers that mainly define the behavior of the control signals on the interface to the host system.
- EHCI operational registers for initializing EHCI.
- The Philips Transfer Descriptors (PTDs) found in the ISP176x memory are the dedicated memory areas defined at the CPU address 0400h–0FFFh.
- The payload area that contains the data to be transferred (1000h–FFFFh).

The payload is located at the predefined CPU address—1000h–FFFFh—specified by the PTD. The ISP176x implements 63 kbytes of internal memory that consists of the PTDs and payload area. This memory size is optimized to balance the maximum USB performance, with minimal system loading and cost.

The address lines A[17:1] determine a total addressing space of 64 kbytes (16-bit or 32-bit addressing mode) of which 1 kbyte is occupied by the registers addressing space.

2. Initializing sequence of the ISP176x

The following sections explain the steps involved when initializing the ISP176x sequence to enumerate an external Hi-Speed USB device. The steps are listed in sequence and simply following the order and the example values listed should ensure the functionality described. Some simple changes may be necessary. For example, in the HW Mode Control register to initialize the concrete system implementation.

2.1 Programming the CPU interface

This section provides the steps necessary to program the CPU interface to set up control signals. You may need some changes in the steps provided, depending on your application. For details on programming the ISP176x, refer to the ISP1760 and ISP1761 data sheets. The usual sequence requires the following steps:

1. SW Reset register (address: 030Ch)

   Example: 00000003h

   Action: This will reset both the CPU interface and EHCI registers. An internal reset pulse is generated and both bits are automatically cleared by the hardware.

2. HW Mode Control register (address: 0300h)

   Example: 00000125h

   Action: As a result, the following ISP176x settings are achieved:

- Global interrupt enabled: IRQ will be asserted as soon as any of the enabled events occur.
- IRQ is defined as ‘edge triggered’. A pulse of predefined width is generated when IRQ occurs.
- IRQ is active HIGH.
- DREQ is active HIGH. DREQ will become 1 when asserted.
- DACK is active LOW. A low-active DACK is expected from the host system.
- Data bus width is set for 32-bit mode.

3. Interrupt Enable register (address: 0314h), as necessary

   Example: 00000004h

   Action: An INT will be generated only when the DMA transfer is completed.

4. Edge Interrupt Count register (address: 0340h) can also be programmed with suitable values (optional).

   Example: 0000FFFFh

   Action: The INT pulse width is approximately 1 ms. This is the maximum value that can be programmed using bits [15:0]. In this example, no minimum delay between two subsequent INTs is specified. If a delay is specified, only one INT is generated after the specified time because of the specified events. The Interrupt register will contain the cumulated INT events (not an INT queue). All bits corresponding to occurred events will be set, regardless of whether the respective bit sources in the Interrupt Enable register were enabled or not. The IRQ line will be asserted only if one of the enabled events has occurred.

5. DMA Configuration register (address: 0330h), if the ISP176x DMA operation is required

   Example: 0020000Eh

   Action: As a result:

- DMA is programmed for write to the ISP176x RAM.
- DMA is enabled.
- A 16-cycle DMA burst is selected.
- The transfer length is 8 kbytes.

6. Port 1 Control register (address: 0374h), applicable for configuration of Port 1 only.

   Example: 00800018h

   Action: Port 1 ATX internal MUX is switched to the Host Controller. The V_{BUS} of Port 1 will be enabled as soon as the port power is enabled. The external power source drives V_{BUS}.

2.2 Programming the internal EHCI

Programming the internal EHCI concludes with enabling the internal Host Controller port and connecting the internal hub. For details on programming the register, refer to the ISP1760 and ISP1761 data sheets, and *Enhanced Host Controller Interface Specification Rev 1.0.*

This sequence should follow the steps mentioned in the preceding section, with the minimal steps described as follows:

1. USBCMD register (address: 0020h)

   Example: 00000002h

   Action: Host Controller reset.

2. CONFIGFLAG register (address: 0060h)

   Example: 00000001h

   Action: All ports are routed to EHCI.

3. PortSC1 register (address: 0064h)

   Example: 00001000h

   Action: Port power enable.

4. PortSC1 register (address: 0064h)

   Example: 00001100h

   Action: Port reset.

5. PortSC1 register (address: 0064h)

   Example: 00001000h

   Action: Port reset condition removed; port power still present.

6. USBCMD register (address: 0020h)

   Example: 00010021h

   Action: R/S = 1; ITC[7:0] = 01h; ASYNC schedule enable = 1.

7. USBINTR register (address: 0028h)

   Example: 00000001h

   Action: USB interrupt enabled.

8. ATL PTD Skip Map register (address: 0154h)

   Example: 10000000h

   Action: Set the ATL Skip Map.

   Remark: Do not skip the current TD, which will be programmed.

9. ATL PTD Last PTD register (address: 0158h)

   Example: 10000000h

   Action: Set the ATL Last PTD Map.

   Remark: Set the last TD at a position higher than the current.

2.3 Enumerating the internal hub

The Internal hub is always connected on the internal root hub port. The internal hub connection can be verified by reading once the PORTSC1 register, after removing the EHCI RESET condition. In this case, the IRQ generation is not used.

This section covers enumerating the internal hub, enabling the downstream ports, and enabling the port power of these ports. The enumeration of the internal hub means sending standard USB hub configuration commands. These configuration commands contain a series of standard requests. Each command has a Setup token phase and an IN token phase.

To achieve correct communication with the internal hub, the correct PTD programming is necessary. The PTD area is situated in memory at address 0400h–0FFFh, just above the registers addressing space. The PTD area consists of the ISO, INT and ATL areas. Also, for the Setup token, a certain payload will be used, specific to each request. For details, refer to the PTD description in the data sheet.

The Setup token requires the following PTD settings:

00000800h—DW1

10038000h—DW2

83800000h—DW3

00000000h—DW4

00000000h—DW5

00000000h—DW6

00000000h—DW7

21000041h—DW0

DW0 contains the number of bytes to be transferred. It should be the last value written because it sets the PTD valid bit V that enables the PTD execution. Alternatively, the Skip Map register may be set to skip the current PTD bit first and then unskip after completing the PTD initialization.

DW1 contains the device address and endpoint. It should be adjusted after setting the hub address. For example, after Set Address, DW1 should be changed to 00000810h. The same applies to the IN token. These are referred to as ‘Setup token 2’ and ‘IN token 2’ in the following example.

DW2 contains Set Address, that is, the address at which the payload should be placed for an OUT transfer and from where data will be retrieved for an IN transfer.

These eight double words of data must be written to the ATL PTD memory area. For example, at CPU addresses: C00h, C20h, C40h and so on.

The payload preparation is performed first at the memory address specified in the PTD DW2 and then the PTD is set up. The payload data is the data that will be sent to the internal hub. The payload address specified in the PTD is an internal ISP176x address calculated according to the formula given in Section 7.2.2 of the ISP1760 and ISP1761 data sheets. For example, the ISP176x internal address 380h in DW2 corresponds to the external system or CPU address 2000h. As a result, the system CPU must write the payload data at the ISP176x internal memory address 2000h. Once the PTD is correctly set up and the payload data is ready, the valid bit V in DW0 will be set, which will enable the execution of the PTD. Alternatively, the Skip Map register may be used to disable or enable PTD execution.

Similarly, the IN token requires correct setting up of the PTD area.

The IN token PTD setting:

21000200h—DW0

00002400h—DW1

10038100h—DW2

83800000h—DW3

00000000h—DW4

00000000h—DW5

00000000h—DW6

00000000h—DW7

21000201h—DW0

Similar to the Setup token, the last line sets the V bit, enabling the execution of the IN token PTD.

Get Descriptor

Payload:

Lower 32: 01000680h (At the ISP176x internal memory address 2000h.)

Upper 32: 00120000h (At the ISP176x internal memory address 2004h.)

Setup token

IN token

Set address:

Payload:

Lower 32:(00020500h) (At the ISP176x internal memory address 2000h.)

Upper 32:(00000000h) (At the ISP176x internal memory address 2004h.)

Setup token

IN token

Set Hub Configuration:

Payload:

Lower 32:(00010900h)

Upper 32:(00000000h)

Setup token

IN token

Set_Feature (Port_Power #2)

Payload:

Lower 32:(00080323h)

Upper 32:(00000002h)

Setup token 2

IN token 2

Clear_Feature (C_Port_Connection Port #2)

Payload:

Lower 32:(00100123h)

Upper 32:(00000002h)

Setup token 2

IN token 2

Set_Feature(Port_Reset #2)

Payload:

Lower 32:(00040323h)

Upper 32:(00000002h)

Setup token 2

IN token 2

The Setup token 2 is the same as the Setup token example, except that the value of DW1 will be 00000810h, as mentioned earlier. The same applies to IN token 2.

These steps listed will enable the downstream port 2 and V_{BUS} will be present on this port, that is, port power is enabled.

If a USB device is connected when port 2 is reset, chirp will be generated and the port will be automatically switched to the speed corresponding to the USB device connected to the respective downstream port. The chirp sequence is hardware-controlled and is initiated by the device as a response to the Host Controller port reset. μSOF will be generated each 125 μs on port 2 at this stage on the downstream port, if a high-speed device is connected and the chirp sequence was successful. Complete enumeration of an external USB device can be consequently performed.

3. ISP176x DMA or PIO data transfers programming

3.1 Microprocessor DMA—ISP176x DMA case

This section explains the programming of the ISP176x DMA in the microprocessor DMA—ISP176x DMA case.

1. Program the DREQ and DACK active level polarity in the HW Mode Control register.

   Example: 0x300 <= 0x125

   Action:

   GLOBAL_INT_EN—an IRQ will be generated on any enabled event.

   IRQ is level-triggered.

   IRQ is active HIGH.

   DREQ is active HIGH.

   DACK is active LOW.

   CPU interface is set to 32-bit data bus width.

   IRQ, DREQ and DACK of the Peripheral Controller are routed to the respective Peripheral Controller dedicated pins. Common mode is not selected.

   Digital overcurrent scheme is implemented.

   No ATX is reset.

2. Set the correct DMA start address in the DMA Start Address register.

   Example: 0x344 <= 0x400

   Action: System's DMA must start the transfer from address 0x400. This is the external address of the actual CPU memory.

   Remark: Usually memory addresses above 0x1000 will be specified in this register because the payload area is located above address 0x1000.

3. Program the system's DMA according to the ISP176x DMA settings.

   Programming steps are system-specific. All system DMA parameters must match the ISP176x DMA programmed settings. Signal's active polarity, DMA burst length, transfer count, start address, and so on.

   The system DMA can be enabled at this step, if configured to wait for the ISP176x DREQ. If the system DMA will not wait for the ISP176x DREQ, then this step should be performed after step 4.

4. Set the HcDMAConfiguration register.

   Example: 0x330 <= 0x0020000b

   Action:

Selected DMA READ—from the ISP176x memory.

DMA enabled—DREQ is immediately asserted.

Selected 8-cycle DMA burst.

Programmed 8-kbyte data transfer.

Fig 1 shows the timing diagram of a typical DMA data transfer, corresponding to these settings.

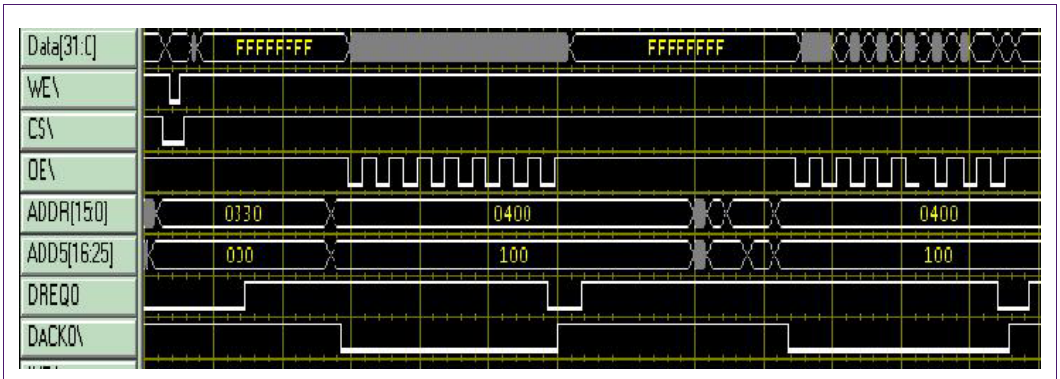


Fig 1. Typical DMA data transfer.

It is observed that DREQ is immediately asserted after the ISP176x DMA is enabled and the system will assert DACK in response, executing a burst according to the programmed parameters. DREQ will be deasserted on the last write or read cycle of each burst, and will be reasserted until the programmed transfer count is completed.

If the IRQ generation at the end of the DMA transfer is necessary, the ISP176x IRQ can be enabled at the End-Of-Transfer (EOT) in the Interrupt Enable register. For example: for 08h, only the DMA EOT IRQ is enabled.

3.2 Microprocessor DMA—ISP176x PIO case

You can obtain a better data transfer rate on the extension bus in this mode by eliminating the DREQ and DACK latencies, and also because of the ISP176x-specific timing.

In this case, the ISP176x DMA programming is not necessary. The system's DMA will access the ISP176x memory as in a typical memory-to-memory access. It is not necessary to connect the DREQ and DACK signals. A 10 kΩ pull-up resistor should be placed on the ISP176x DACK input signal, if not used.

The system must generate CS_N instead of DACK, as during a normal PIO access.

The ISP176x will not generate IRQ on completing DMA because there is no transfer count programmed. The system's DMA, however, will stop on its programmed transfer count and an internal processor INT on EOT can be usually generated.

There is no preparatory programming necessary in the case of a write to the ISP176x memory.

Special attention must be given when reading from the ISP176x memory because the 0x33C register must be programmed with the correct read start address, before starting the memory read operation. As long as the read address is contiguous, it is not necessary to program register 0x33C again.

For details, refer to the ISP176x data sheet.

4. Schematics description

4.1 ISP176x pin connection and general description

The ISP176x can be accessed on either a 16-bit or 32-bit data bus, address line A0 is never used because the 8-bit mode access is not possible.

In the case of a 16-bit data bus, address line A1—the first address line used—must be connected to the system's A1.

In the case of a 32-bit data bus, address line A2—the first address line used—must be connected to the system's A2. A1 is not used in this case and can be connected to the system's A1 or GND.

The ISP1760 and ISP1761 have the same packaging and pinout. Therefore, the ISP1761-specific pins that are not used in the ISP1760 are defined as test pins and must be connected as follows:

- Pin 3 GND
- Pin 81 GND
- Pin 117 GND
- Pin 120 pull-up to $V_{CC(I/O)}$
- Pin 124 connect a 220 nF capacitor between this pin and pin 125
- Pin 125 connect a 220 nF capacitor between this pin and pin 124
- Pin 126 connect to 3.3 V.

When the ISP176x is connected to a system with a 16-bit data bus and the upper 16 data lines are not used, these data lines must be connected to 3.3 V through a 15 kΩ pull-up resistor. A single external resistor can connect the upper 16 data lines together.

If the design requires only two USB ports, the port that is not used must have both DP and DM pulled up through individual 15 kΩ resistors. The respective OCn_N pin is also not used. Therefore, the pin must be pulled up using a 10 kΩ resistor.

The ISP176x has separate power pins for the digital CPU interface—pins 10, 40, 48, 59, 67, 75, 83, 94, 104 and 115—and separate power inputs for the digital core and analog blocks—pins 6 and 7. Internal linear voltage regulators—5 V to 3.3 V (analog) and 5 V to 1.8 V (digital core)—are powered using the power source on pins 6 and 7.

If a stable 3.3 V power source that can supply a minimum additional current of 150 mA is already available in the design, pins 6 and 7 can be directly powered by 3.3 V. This will reduce the power consumption, avoiding unnecessary power dissipation in the ISP176x internal power supply block.

Pin 9 is the output of the ISP176x internal linear regulator (5 V to 3.3 V) and requires decoupling capacitors of 100 nF and 4.7 µF to 10 µF to be placed as close as possible.

Pins 5, 50, 85 and 118 are connected to the output of the ISP176x internal linear regulator (5 V to 1.8 V). Each pin must be connected to a 100 nF decoupling capacitor, placed as close as possible to the respective pin. An additional 4.7 µF to 10 µF capacitor must be connected to any of these pins.

Similarly there are separate GND connections. In a concrete design, these can be connected together to a single GND plane.

Depending on your design, it may be a good idea to connect reference GND pins 15, 22 and 29 to GND through a properly selected inductor, if isolation of potential noise from the local GND plane is necessary.

It is recommended that you place electrostatic discharge (ESD) protection components on each USB port. For example, the Philips IP4059 that contains four ultra-low capacity ESD protection diodes will ensure protection of up to ±8 kV contact and ±15 kV air discharge. These components will be connected to the DP and DM signals, V_{BUS} and GND, and must be placed as close as possible to the USB connector. The traces to the DP and DM signals must be as short as possible, preferably below 5 mm.

If the DC_IRQ line is connected to one of the system's IRQ lines and you intend to use it, program correct polarity for both the ISP176x and the processor, even if, only the Host Controller may be used in the first phase and the peripheral does not need programming. Maintaining default polarity settings may produce signal contention. An undesirable result may be a higher current consumption in suspend mode.

4.2 Overcurrent

The ISP176x supports both the analog and digital overcurrent schemes. Both solutions are implemented in the example schematics, considering the components can be optionally soldered for one solution or the other. For example, the analog solution means soldering U15, R30, R33 and C43 but not MIC2026 and R21.

The analog overcurrent detection is achieved by sensing the voltage drop on PMOS transistor U15. This will automatically deassert the PSW2_N signal that will cut off the V_{BUS} voltage on the respective port. The voltage drop detection is in the range of 45 mV to 90 mV.

For an overcurrent limit of 500 mA, a PMOS transistor with R_{DSON} = 100 mΩ must be selected. The overcurrent limit can be adjusted by modifying the series resistor. For example, R30 for port 2 in the schematics.

The digital overcurrent scheme requires using a standard power switch with integrated overcurrent detection such as:

- LM3526, MIC2526 (2 ports) or
- LM3544 (4 ports).

The advantage of these integrated power control switches is that they implement thermal shutdown and internal filter of 1 ms to 3 ms to prevent false overcurrent reporting that may appear because of in-rush currents when plugging a USB device.

The digital overcurrent protection logic of the ISP176x uses the following two pins for power switching and overcurrent protection of each USB port:

- PSWn_N: This pin will enable or disable the respective external power switch (MIC2526 and LM3526).
- OCn_N: This is an input on which the respective port power device will signal a fault condition.

For both the analog and digital overcurrent schemes, the port power will be disabled, that is, PSWn_N is deasserted, once an overcurrent or other fault condition is detected on the respective port.

A possible alternative is to use a ‘resettable fuse’ (or polyswitch PTC device) on each port or one for 1 to 3 ports.

The advantage of this solution is cost but it will not allow a permanent cut-off of the Port Power in the case of an overcurrent event. It will protect the port by switching on or off as long as the overcurrent condition persists.

4.3 Suspend, resume and power management

Suspend mode is achieved by appropriate programming of the ISP176x EHCI registers and sending right commands to the internal hub. This will determine when the internal ISP176x clocks will stop, leaving only an internal low-frequency dedicated LazyClock running to detect events that may require a wake-up.

The Power Down Control register of the ISP176x allows power down of the ISP176x blocks to be programmed for maximum power-saving during suspend. Programming these bits will disable certain blocks. Consider the consequences of disabling functionality when you perform bit settings. The events that may produce a wake-up are:

- CS_N assertion, usually determined by any access to register or memory.
- A plug or unplug of a USB device.
- Driving LOW the SUSPEND/WAKEUP_N pin.

The SUSPEND/WAKEUP_N pin is a dual-function pin and should be usually connected to an available GPIO pin on the processor. It initially indicates the suspend or awake state of the ISP176x but can also produce the ISP176x wake-up, if driven or forced LOW during suspend (open-drain).

A resume because of any of the defined wake-up events can generate a CLKRDY_IRQ that must be enabled in the Interrupt Enable register, if IRQ assertion is needed. The IRQ assertion will show that the internal CLK signals are restored and normally running for the default 10 ms timeout.

For details on register programming, refer to the ISP1760, ISP1761 data sheets.

4.4 Specific schematics elements

This section explains specific elements in the schematics related to connecting the ISP176x to the Intel® PXA25x processor. The schematics are a concrete example of connecting the ISP176x to the extension bus of the Accelent PXA25x development system.

The ISP176x must be defined as ‘Variable Latency I/O’ in the MSC register of PXA25x because it does not operate on a synchronous interface but without the implementation of the READY signal.

If necessary, additional wait-states defined in the PXA25x register MSC0-2 will increase the access time.

The maximum data transfer that can be obtained in a system depends not only on the ‘active time’ pulse but also on the cycle-to-cycle ‘inactive time’. This is determined by the concrete setting possibilities of each system: DMA channel priority, system loading and execution of other parallel tasks. For example, the timing diagram shows that the efficiency of the DMA transfer can be sensibly affected by the DACK latency.

Although after a burst of 16, DREQ is immediately asserted again to complete the transfer, DACK is asserted only after sometime, depending on the DMA channel priority, SDRAM speed and other tasks running. This will reduce the maximum data transfer rate and is strictly related to the capability of the processor to generate DACK.

During the ISP176x memory DMA accesses only DACK should be active, and perform WR_N or RD_N, according to the cycle in progress.

5. PCB design recommendations

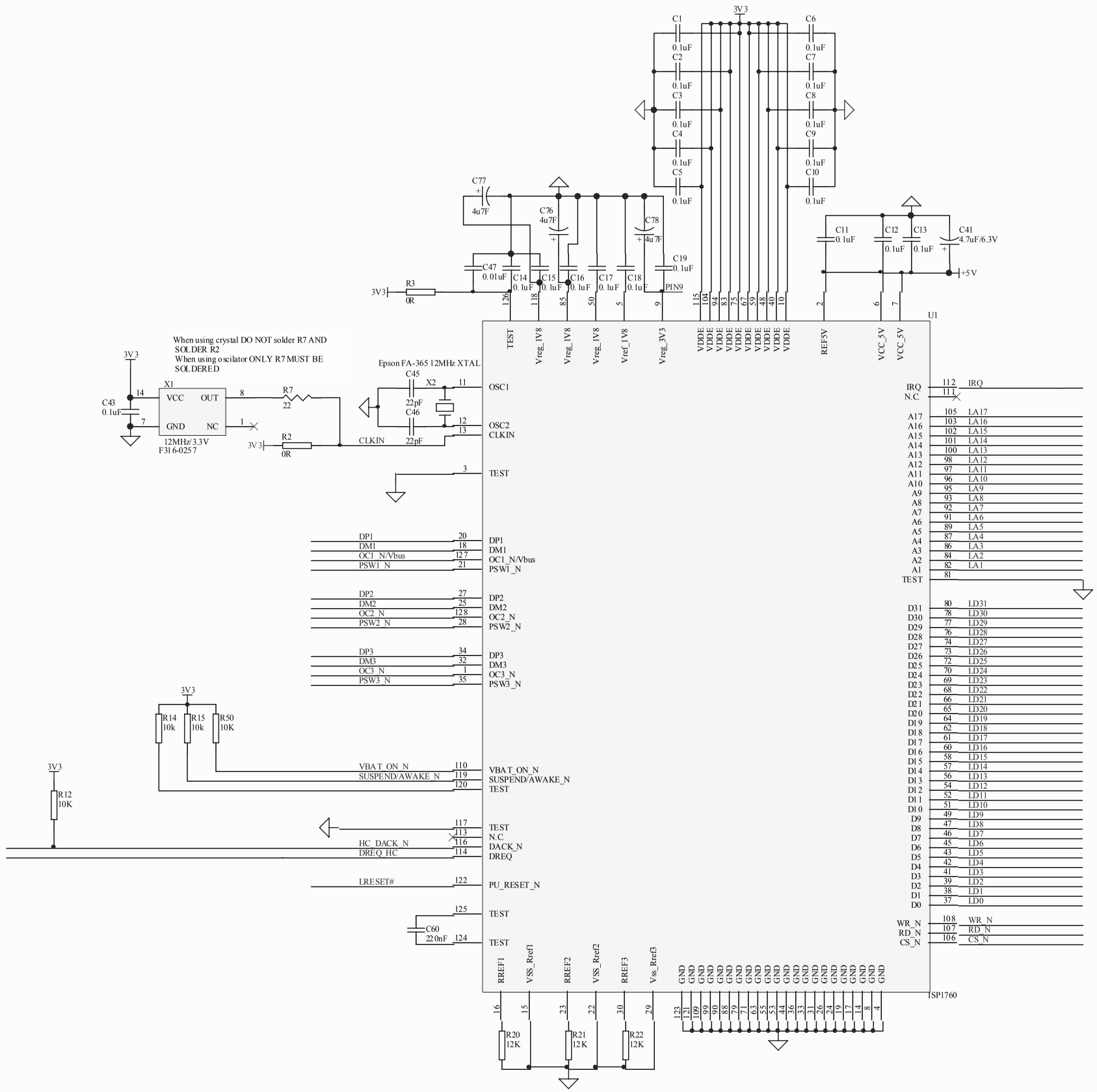
Some important checks for a successful PCB design are:

- Typically, a solution using four-layer PCB—signal 1, GND, V_{CC} and signal 2—is sufficient for proper routing, allowing you to obtain good functionality and meeting all compliance tests requirements. Start your design by placing the ISP176x chip, the major components, and routing of the high-speed DP and DM traces, and clock traces. Also, a complete 'clean' solution for routing the power and GND (plane split) must be defined before you start routing the rest of the signals.
- Route the Hi-Speed USB differential pairs over continuous GND or power planes. Avoid crossing anti-etch areas and any breaks in the internal planes (plane split). The minimum recommended distance to a plane split is 25 mils. Also avoid placing a series of VIA holes near the DP and DM lines because these will create 'break areas' in the GND plane below. This is because of the clearance imposed by the manufacturing process around any VIA holes to an internal plane.
- Try to keep the length of the DP and DM traces equal. The maximum trace length mismatch between Hi-Speed USB signal pairs must not be greater than 70 mils.
- Maintain parallelism between USB differential signals, with the trace spacing needed to achieve 90 Ω differential impedance. To achieve the required impedance of the pair traces, it is recommended that you use 8 mils traces and keep the distance between DP and DM traces at 8 mils. These values may vary depending on the actual PCB parameters.
- Avoid corners when routing the differential pair DP and DM. Any 90° direction change of traces must be accomplished with two 45° turns or by using an arc of an imaginary circle tangent to the DP and DM lines.
- Avoid routing the USB differential pairs near I/O connectors, signal headers, crystals, oscillators, magnetic devices and power connectors.
- Maintain the maximum possible distance between high-speed USB differential pairs, high-speed or low-speed clock, and non-periodic signals. The minimum recommended distances are as follows:
  - 20 mils between the DP and DM traces and low-speed non-periodic signal traces.
  - 50 mils between the DP and DM traces and clock or high-speed periodic signal traces.
  - 20 mils between two pairs of the DP and DM traces.
- Avoid creating stubs for connecting 15 kΩ pull-down resistors or for testing points. If a stub is unavoidable in the design, no stub must be greater than 80 mils.
- Route all the DP and DM lines on one layer. Do not change layers (avoid using VIAs) even to avoid crossing a plane split. It is better to place a nonsplit plane under high-speed USB signals, ground layer or power layer. It is recommended that you place ground layer beneath the DP and DM lines.
- The maximum allowed length of the DP and DM lines for onboard solutions (or [trace + cable length] for a front-panel solution) is 18 inches.
- A decoupling capacitor must be placed on V_{BUS} as close as possible to each USB connector. A value of about 150 µF/10 V is recommended on each port.
- The decoupling capacitors must be placed as close as possible to the ISP176x. A good choice is the four corners of the IC because these areas will not normally be occupied by traces or other components, according to the ISP176x pinout.
- For good EMI testing results, it is recommended that you provide a good path from the USB connector shell to the chassis ground. The USB connector shell must be connected to an isolated ground plane.

For more information, refer to the Intel document *The USB 2.0 Platform Design Guideline, Rev. 1.0* at http://developer.intel.com/technology/usb/techlit.htm.

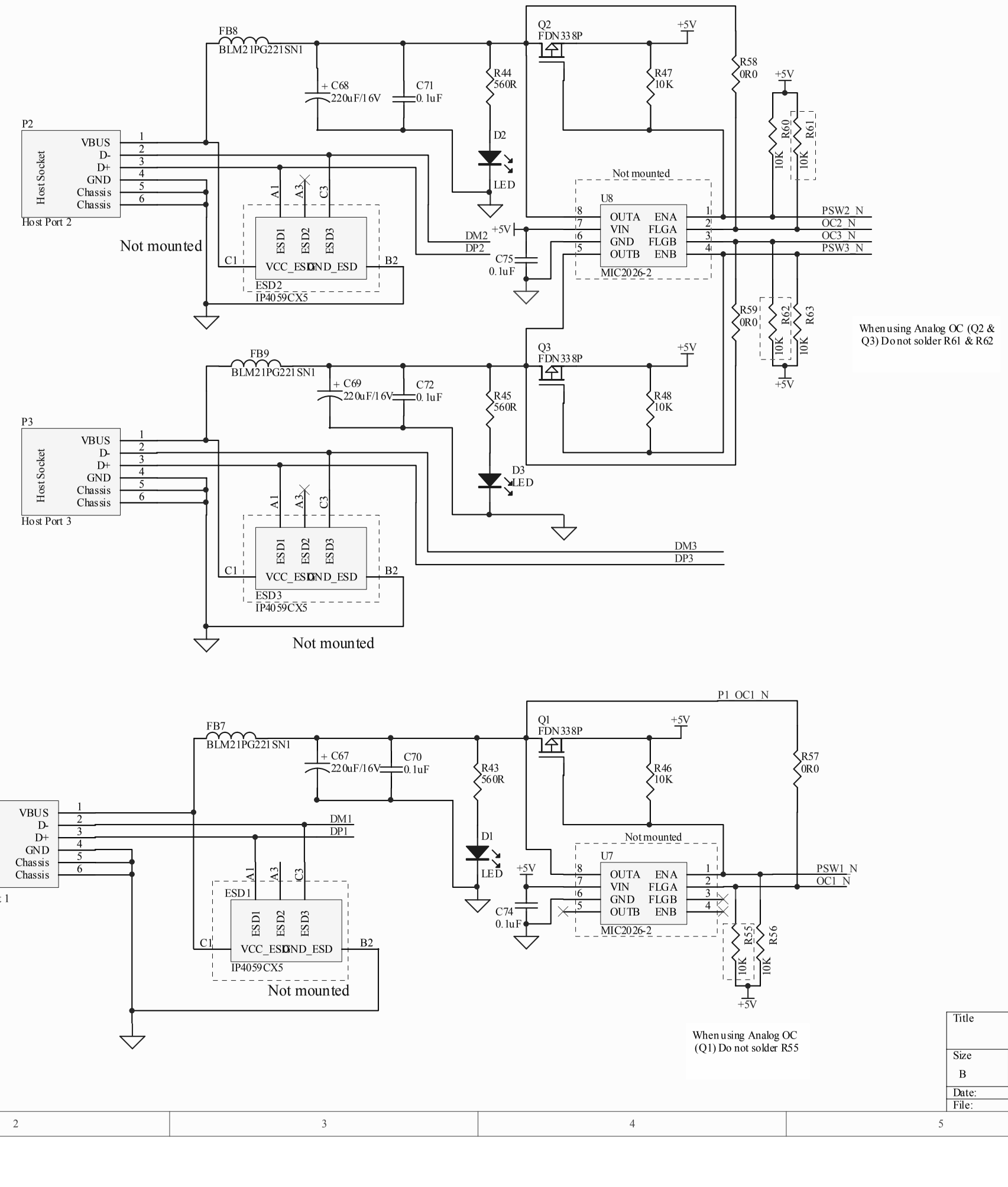
6. Schematics

Philips Semiconductors



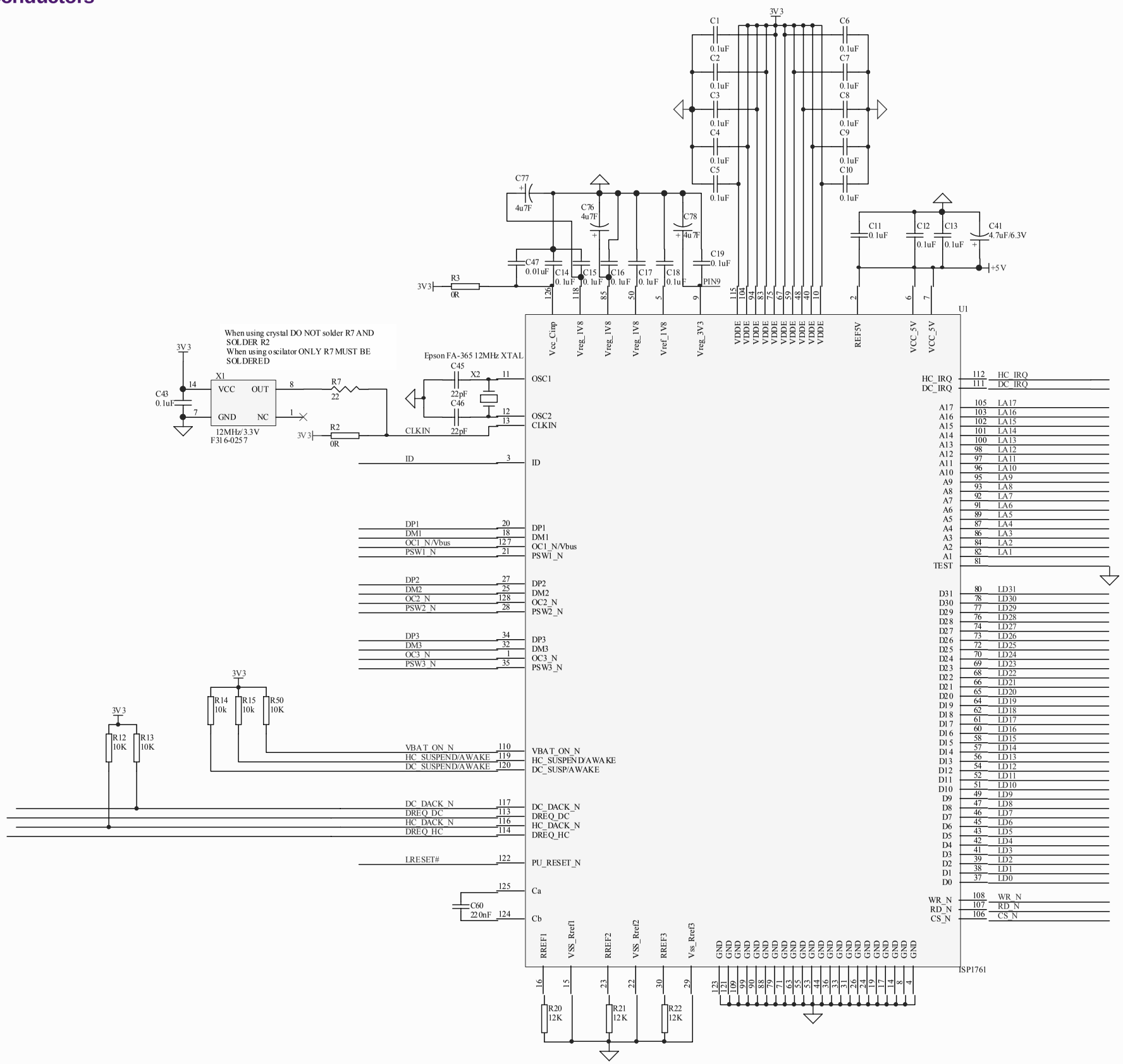
Title	ISP1760 Ver 1.0
Size: C	Number:
Revision:	
Date: 9/23/2004	Sheet of
File: E:\1761project\..\1760.Sch	Drawn By:

Philips Semiconductors



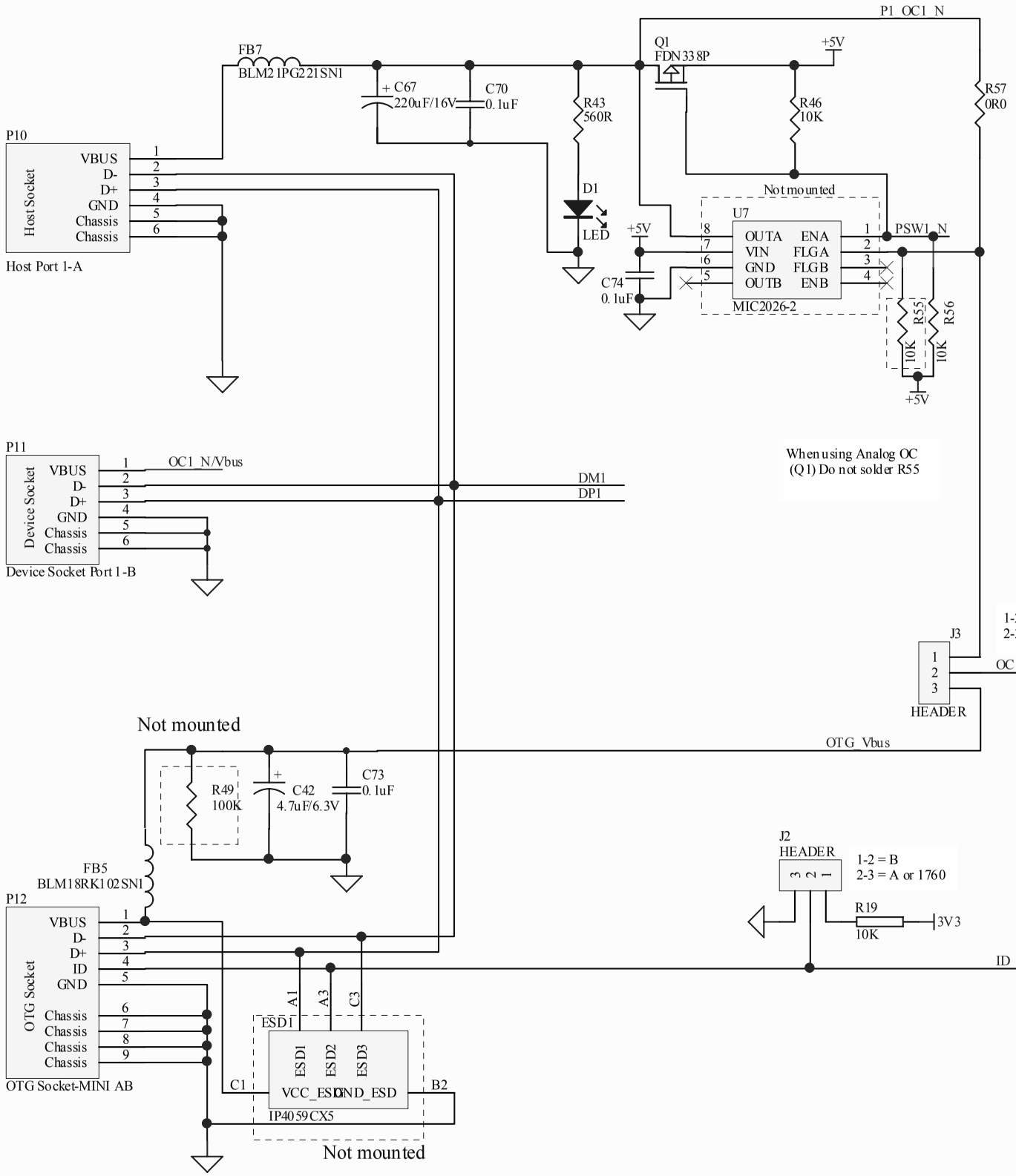
Title	ISP1760 Ver 1.0	
Size: B	Number:	Revision:
Date: 9/23/2004	Sheet of	
File: E:\1761project\..\1760USBPorts Interface\sch	Drawn By:	

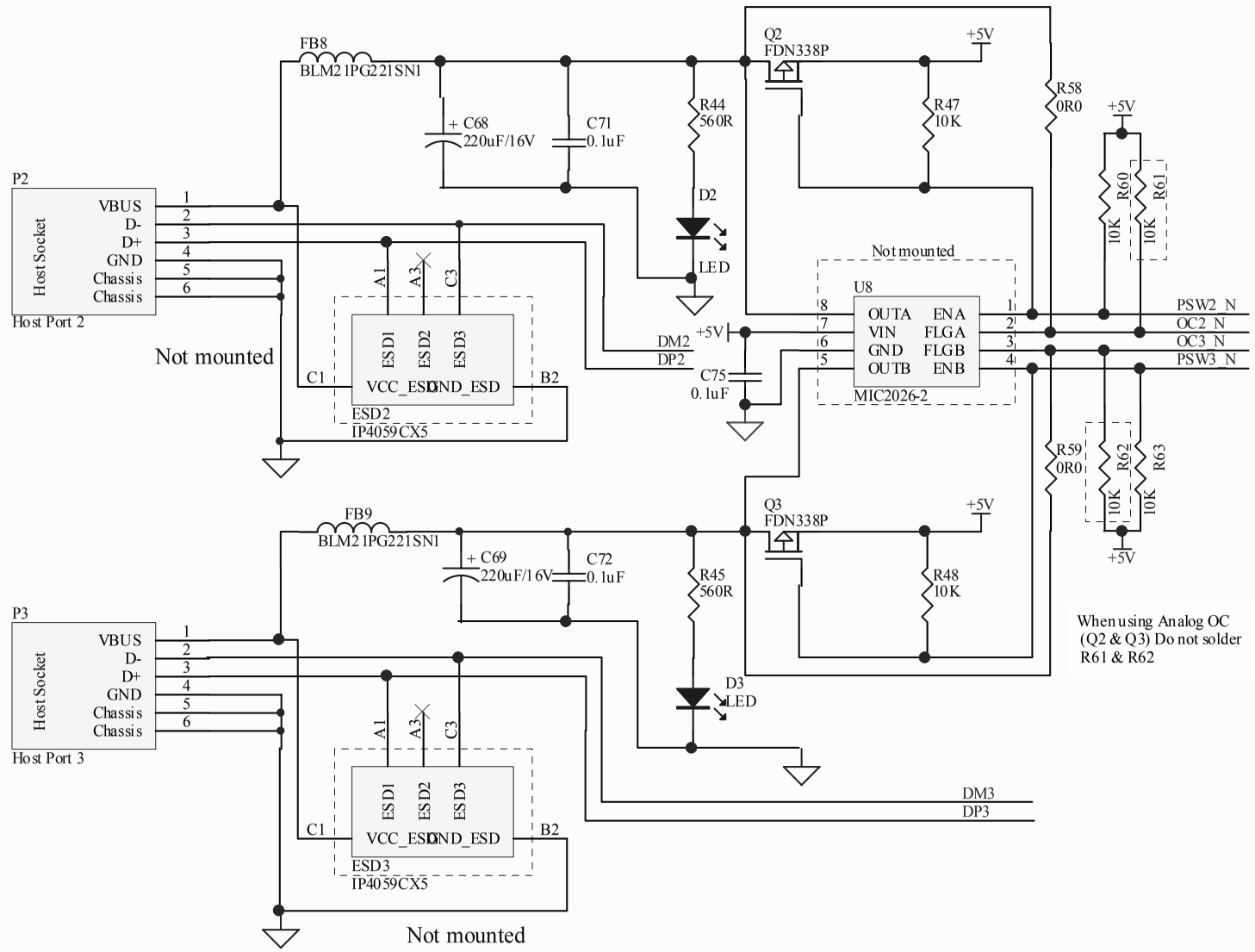
Philips Semiconductors



Title	ISP1761 Ver 1.0
Size: C	Number:
Revision:	
Date: 9/23/2004	Sheet of
File: E:\1761project\..\176x.Sch	Drawn By:

Philips Semiconductors





Title	ISP1761 Ver 1.0	
Size: B	Number:	Revision:
Date: 9/23/2004		Sheet of
File: E:\I761project\..\USBPorts Interface.sch		Drawn By:

7. Pinning configuration

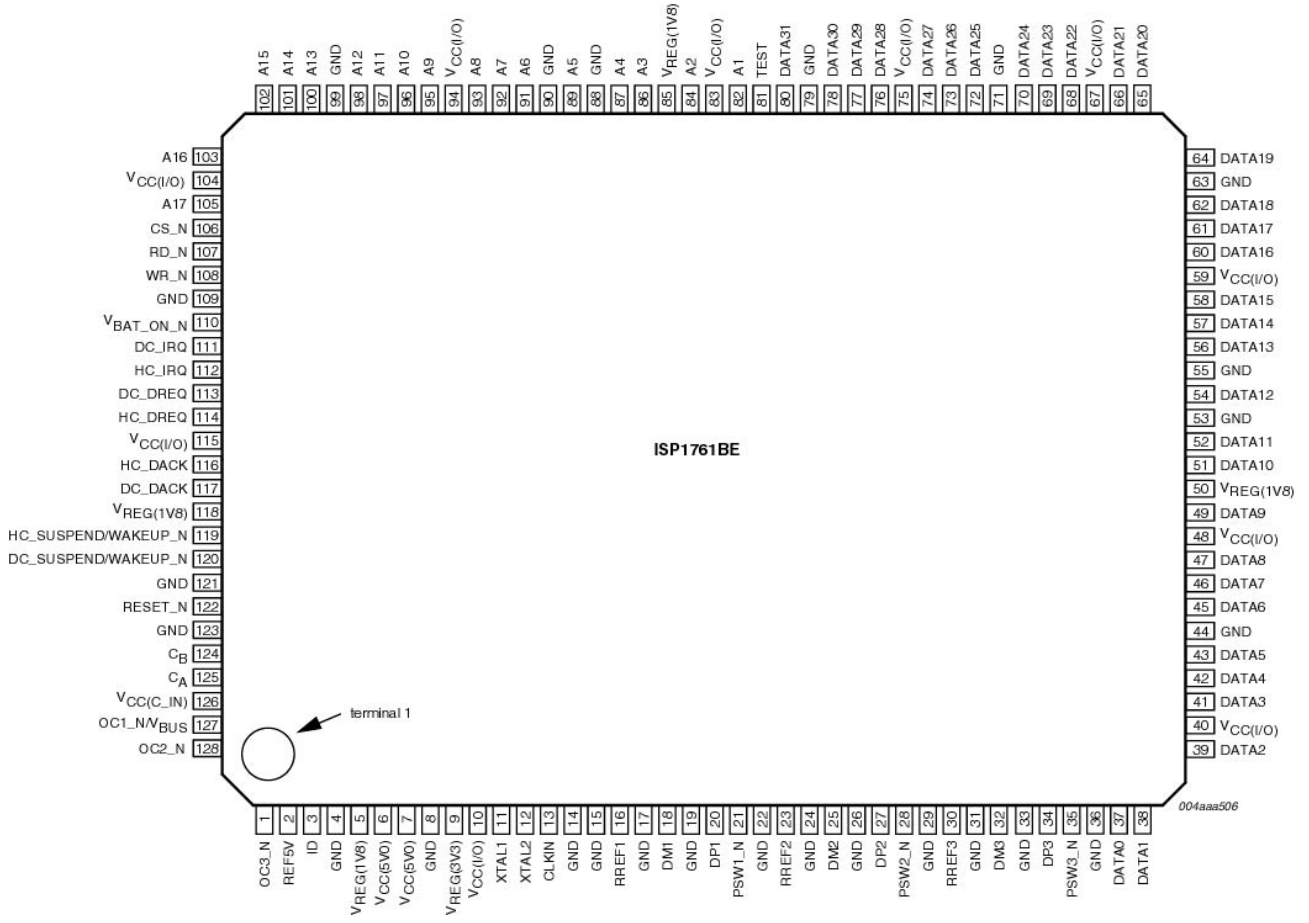


Fig 2. ISP1761 pinning diagram.

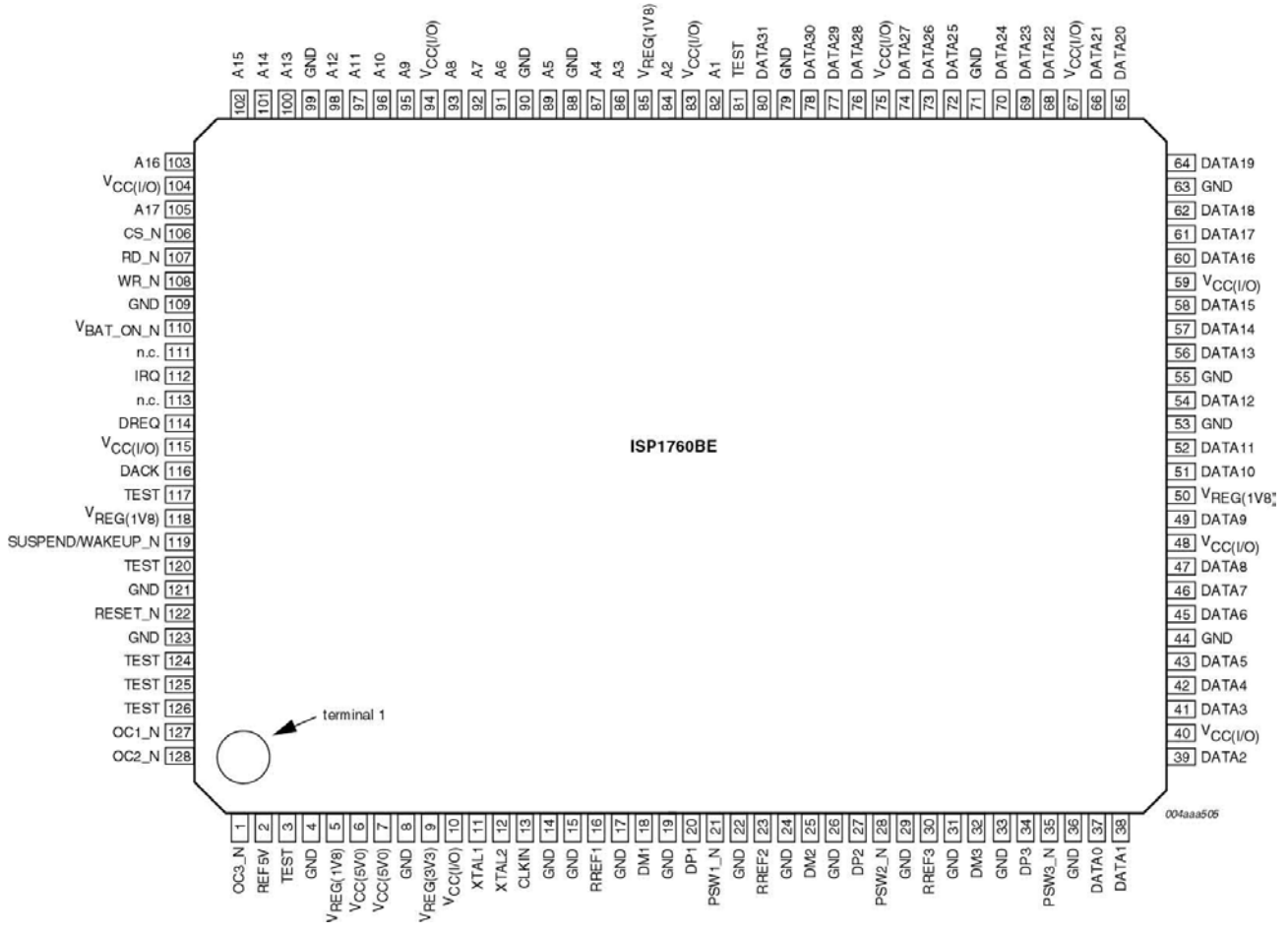


Fig 3. ISP1760 pinning diagram.

8. References

- *Universal Serial Bus Specification Rev. 2.0*
- *ISP1761 Hi-Speed Universal Serial Bus On-The-Go controller* data sheet
- *ISP1760 Hi-Speed Universal Serial Bus host controller for embedded applications* data sheet
- *The USB 2.0 Platform Design Guideline, Rev. 1.0*
- *Enhanced Host Controller Interface Specification Rev 1.0*.

9. 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.

10. Trademarks

Intel — is a registered trademark of Intel Corp.

11. Contents

Date of release: 4 October 2004

Published in The Netherlands