ISP1161

Full-speed Universal Serial Bus single-chip host and device controller

Rev. 02 — 13 December 2002 **Product data**

1. General description

The ISP1161 is a single-chip Universal Serial Bus (USB) Host Controller (HC) and Device Controller (DC). The Host Controller portion of the ISP1161 complies with *Universal Serial Bus Specification Rev 2.0,* supporting data rates at full-speed (12 Mbit/s) and low-speed (1.5 Mbit/s). The Device Controller portion of the ISP1161 also complies with *Universal Serial Bus Specification Rev 2.0,* supporting data rates at full-speed (12 Mbit/s). These two USB controllers, the HC and the DC, share the same microprocessor bus interface. They have the same data bus, but different I/O locations. They also have separate interrupt request output pins, separate DMA channels that include separate DMA request output pins and DMA acknowledge input pins. This makes it possible for a microprocessor to control both the USB HC and the USB DC at the same time.

The ISP1161 provides two downstream ports for the USB HC and one upstream port for the USB DC. Each downstream port has its own overcurrent (OC) detection input pin and power supply switching control output pin. The upstream port has its own V_{BUS} detection input pin. ISP1161 also provides separate wake-up input pins and suspended status output pins for the USB HC and the USB DC, respectively. This makes power management flexible. The downstream ports for the HC can be connected with any USB compliant USB devices and USB hubs that have USB upstream ports. The upstream port for the DC can be connected to any USB compliant USB host and USB hubs that have USB downstream ports.

The HC is adapted from *Open Host Controller Interface Specification for USB Release 1.0a*, referred to as OHCI in the rest of this document.

The DC is compliant with most device class specifications such as Imaging Class, Mass Storage Devices, Communication Devices, Printing Devices and Human Interface Devices.

The ISP1161 is well suited for embedded systems and portable devices that require a USB host only, a USB device only, or a combined and configurable USB host and USB device capabilities. ISP1161 brings high flexibility to the systems that have it built-in. For example, a system that has ISP1161 built-in allows it not only to be connected to a PC or USB hub that has a USB downstream port, but also to be connected to a device that has a USB upstream port such as a USB printer, USB camera, USB keyboard, USB mouse, among others. ISP1161 enables point-to-point connectivity between embedded systems. An interesting application example is to connect an ISP1161 HC with an ISP1161 DC.

Consider the examples of an ISP1161 being used in a Digital Still Camera (DSC) design. Figure 1 shows the ISP1161 being used as a USB DC. Figure 2 shows the ISP1161 being used as a USB HC. Figure 3 shows the ISP1161 being used as a USB HC and a USB DC at the same time.

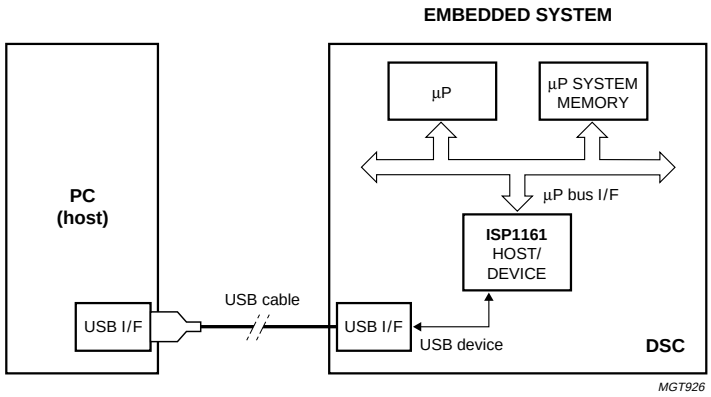


Fig 1. ISP1161 operating as a USB device.

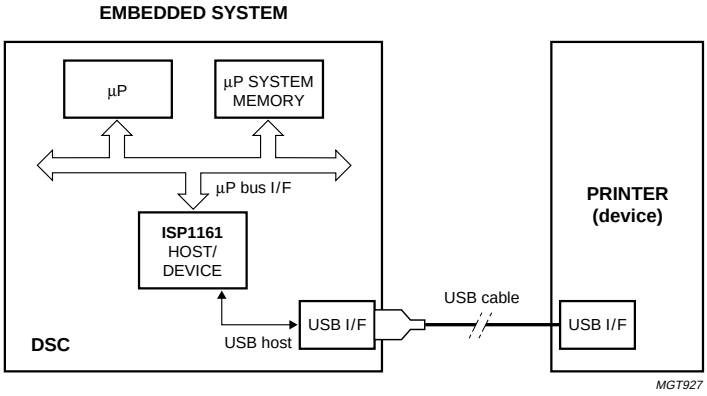


Fig 2. ISP1161 operating as a stand-alone USB host.

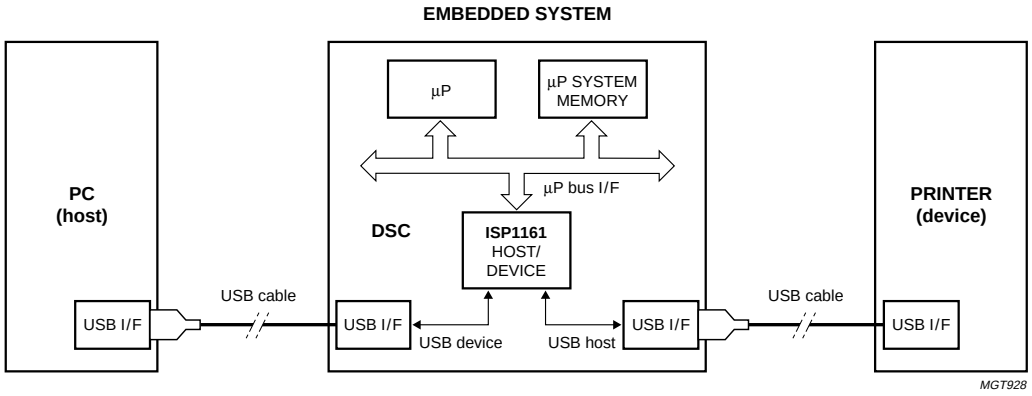


Fig 3. ISP1161 operating as both USB host and device simultaneously.

2. Features

- Complies with *Universal Serial Bus Specification Rev 2.0*
- The Host Controller portion of the ISP1161 supports data transfer at full-speed (12 Mbit/s) and low-speed (1.5 Mbit/s); the Device Controller portion of the ISP1161 supports data transfer at full-speed (12 Mbit/s)
- Combines HC and DC in a single chip
- On-chip DC complies with most Device Class specifications
- Both HC and DC can be accessed by an external microprocessor via separate I/O port addresses
- Selectable one or two downstream ports for HC and one upstream port for DC
- High-speed parallel interface to most of the generic microprocessors and Reduced Instruction Set Computer (RISC) processors such as:
  - Hitachi® SuperH™ SH-3 and SH-4
  - MIPS-based™ RISC
  - ARM7™, ARM9™, StrongARM®
- Maximum data transfer rate of 15 Mbyte/s between microprocessor and HC and 11.1 Mbyte/s between microprocessor and DC
- Supports single-cycle and burst mode DMA operations
- Up to 14 programmable USB endpoints with 2 fixed control IN/OUT endpoints for DC
- Built-in separate FIFO buffer RAM for HC (4 kbytes) and DC (2462 bytes)
- Endpoints with double buffering to increase throughput and ease real-time data transfer for both DC transfers and HC isochronous (ISO) transactions
- 6 MHz crystal oscillator with integrated PLL for low EMI
- Controllable LazyClock (100 kHz ±50%) output during ‘suspend’
- Clock output with programmable frequency (3 to 48 MHz)
- Software controlled connection to USB bus (SoftConnect) on upstream port for DC
- Good USB connection indicator that blinks with traffic (GoodLink) for DC
- Built-in software selectable internal 15 kΩ pull-down resistors for HC downstream ports
- Dedicated pins for suspend sensing output and wake-up control input for flexible applications
- Global hardware reset input pin and separate internal software reset circuits for HC and DC
- Operation at either 5 V or 3.3 V power supply input
- Operating temperature range from −40 to +85 °C
- Available in two LQFP64 packages (SOT314-2 and SOT414-1).

3. Applications

- Personal Digital Assistant (PDA)
- Digital camera
- Third-generation (3-G) phone
- Set-Top Box (STB)
- Information Appliance (IA)
- Photo printer
- MP3 jukebox
- Game console.

4. Ordering information

Table 1: Ordering information

Type number	Package		
	Name	Description	Version
ISP1161BD	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
ISP1161BM	LQFP64	plastic low profile quad flat package; 64 leads; body 7 × 7 × 1.4 mm	SOT414-1

5. Block diagram

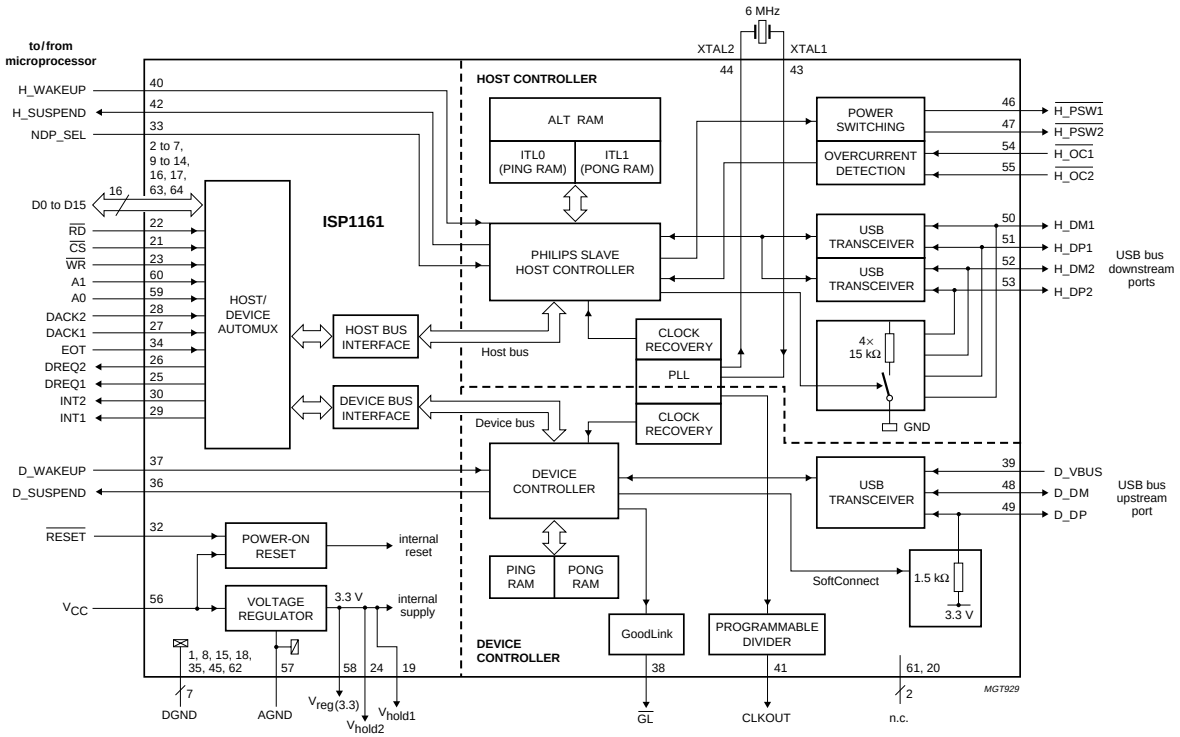


Fig 4. Block diagram.

9397 750 09567

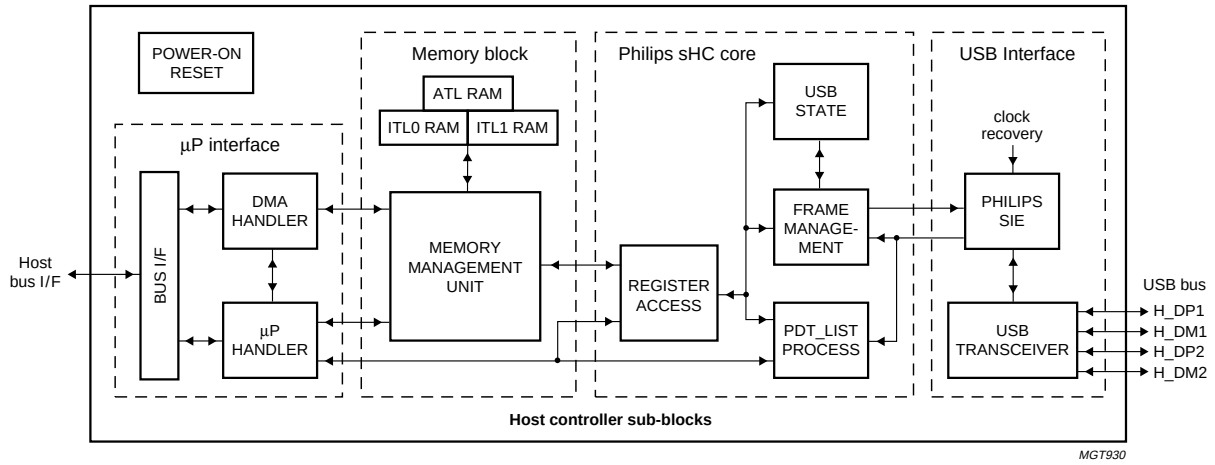


Fig 5. Host controller sub-block diagram.

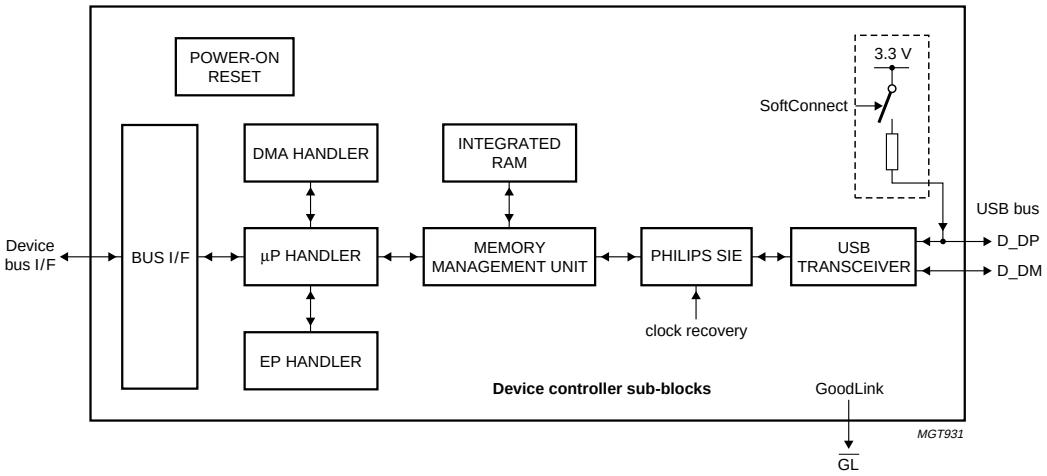


Fig 6. Device controller sub-block diagram.

6. Pinning information

6.1 Pinning

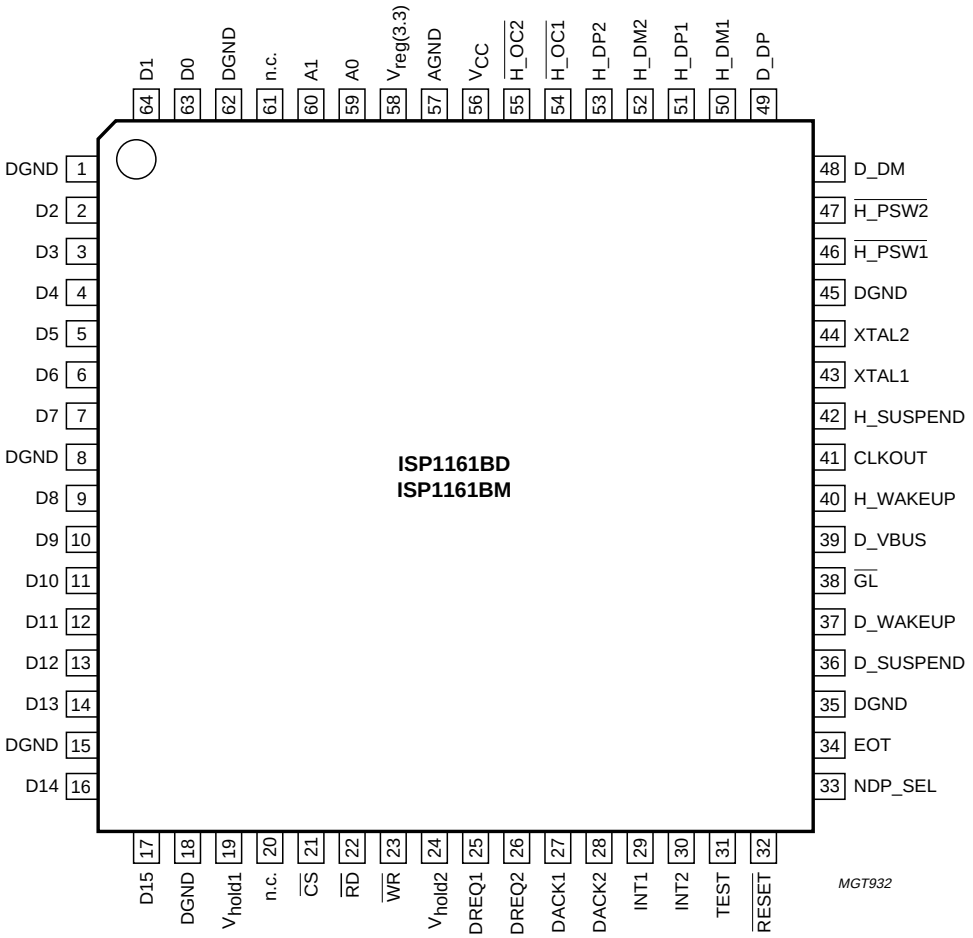


Fig 7. Pin configuration.

6.2 Pin description

Table 2: Pin description

Symbol[1]	Pin	Type	Description
DGND	1	-	digital ground
D2	2	I/O	bit 2 of bidirectional data; slew-rate controlled; TTL input; three-state output
D3	3	I/O	bit 3 of bidirectional data; slew-rate controlled; TTL input; three-state output
D4	4	I/O	bit 4 of bidirectional data; slew-rate controlled; TTL input; three-state output
D5	5	I/O	bit 5 of bidirectional data; slew-rate controlled; TTL input; three-state output

Table 2: Pin description *...continued*

Symbol[1]	Pin	Type	Description
D6	6	I/O	bit 6 of bidirectional data; slew-rate controlled; TTL input; three-state output
D7	7	I/O	bit 7 of bidirectional data; slew-rate controlled; TTL input; three-state output
DGND	8	-	digital ground
D8	9	I/O	bit 8 of bidirectional data; slew-rate controlled; TTL input; three-state output
D9	10	I/O	bit 9 of bidirectional data; slew-rate controlled; TTL input; three-state output
D10	11	I/O	bit 10 of bidirectional data; slew-rate controlled; TTL input; three-state output
D11	12	I/O	bit 11 of bidirectional data; slew-rate controlled; TTL input; three-state output
D12	13	I/O	bit 12 of bidirectional data; slew-rate controlled; TTL input; three-state output
D13	14	I/O	bit 13 of bidirectional data; slew-rate controlled; TTL input; three-state output
DGND	15	-	digital ground
D14	16	I/O	bit 14 of bidirectional data; slew-rate controlled; TTL input; three-state output
D15	17	I/O	bit 15 of bidirectional data; slew-rate controlled; TTL input; three-state output
DGND	18	-	digital ground
V_{hold1}	19	-	voltage holding pin; internally connected to the $V_{reg(3.3)}$ and V_{hold2} pins. When V_{CC} is connected to 5 V, this pin will output 3.3 V, hence do not connect it to 5 V. When V_{CC} is connected to 3.3 V, this pin can either be connected to 3.3 V or left unconnected. In **all** cases, decouple this pin to DGND.
n.c.	20	-	no connection
$\overline{CS}$	21	I	chip select input
$\overline{RD}$	22	I	read strobe input
$\overline{WR}$	23	I	write strobe input
V_{hold2}	24	-	voltage holding pin; internally connected to the $V_{reg(3.3)}$ and V_{hold1} pins. When V_{CC} is connected to 5 V, this pin will output 3.3 V, hence do not connect it to 5 V. When V_{CC} is connected to 3.3 V, this pin can either be connected to 3.3 V or left unconnected. In **all** cases, decouple this pin to DGND.
DREQ1	25	O	HC's DMA request output 1 (programmable polarity); signals to the DMA controller that the ISP1161 wants to start a DMA transfer; see Section 10.4.1
DREQ2	26	O	DC's DMA request output 2 (programmable polarity); signals to the DMA controller that the ISP1161 wants to start a DMA transfer; see Section 13.1.4
DACK1	27	I	HC's DMA acknowledge input 1; when not in use, this pin must be connected to V_{CC} via an external 10 kΩ resistor

Table 2: Pin description *…continued*

Symbol[1]	Pin	Type	Description
DACK2	28	I	DC's DMA acknowledge input 2; when not in use, this pin must be connected to V_{CC} via an external 10 kΩ resistor
INT1	29	O	HC's interrupt output; programmable level, edge triggered and polarity; see Section 10.4.1
INT2	30	O	DC's interrupt output; programmable level, edge triggered and polarity; see Section 13.1.4
TEST	31	O	test output; this pin is used for test purposes only; leave this pin open
$\overline{\text{RESET}}$	32	I	reset input (Schmitt trigger); a LOW level produces an asynchronous reset (internal pull-up resistor)
NDP_SEL	33	I	indicates to the Host Controller software the number of downstream ports present: **0 —** select 1 downstream port **1 —** select 2 downstream ports only changes the value of the NDP field in the HcRhDescriptorA register; both ports will always be enabled
EOT	34	I	DMA master device to inform ISP1161 of end of DMA transfer; active level is programmable; see Section 10.4.1
DGND	35	-	digital ground
D_SUSPEND	36	O	DC's 'suspend' state indicator output; active HIGH
D_WAKEUP	37	I	DC's wake-up input; generates a remote wake-up from 'suspend' state (active HIGH); when not in use, this pin must be connected to DGND via an external 10 kΩ resistor (internal pull-up resistor)
$\overline{\text{GL}}$	38	O	GoodLink LED indicator output (open-drain, 8 mA); the LED is ON by default, blinks OFF upon USB traffic; to connect an LED, use a series resistor of 470 Ω (V_{CC} = 5.0 V) or 330 Ω (V_{CC} = 3.3 V)
D_VBUS	39	I	DC's USB upstream port V_{BUS} sensing input; when not in use, this pin must be connected to DGND via a 1 MΩ resistor
H_WAKEUP	40	I	HC's wake-up input; generates a remote wake-up from 'suspend' state (active HIGH); when not in use, this pin must be connected to DGND via an external 10 kΩ resistor (internal pull-up resistor)
CLKOUT	41	O	programmable clock output (3 to 48 MHz); default 12 MHz
H_SUSPEND	42	O	HC's 'suspend' state indicator output; active HIGH
XTAL1	43	I	crystal input; connected directly to a 6 MHz crystal; when it is connected to an external clock oscillator, leave pin XTAL2 open
XTAL2	44	O	crystal output; connected directly to a 6 MHz crystal; when pin XTAL1 is connected to an external clock oscillator, leave this pin open
DGND	45	-	digital ground

Table 2: Pin description *...continued*

Symbol[1]	Pin	Type	Description
$\overline{\text{H\_PSW1}}$	46	O	power switching control output for downstream port 1; open-drain output
$\overline{\text{H\_PSW2}}$	47	O	power switching control output for downstream port 2; open-drain output
D_DM	48	AI/O	USB D– data line for DC's upstream port; when not in use, leave this pin open
D_DP	49	AI/O	USB D+ data line for DC's upstream port; when not in use, leave this pin open
H_DM1	50	AI/O	USB D– data line for HC's downstream port 1
H_DP1	51	AI/O	USB D+ data line for HC's downstream port 1
H_DM2	52	AI/O	USB D– data line for HC's downstream port 2; when not in use, leave this pin open
H_DP2	53	AI/O	USB D+ data line for HC's downstream port 2; when not in use, leave this pin open
$\overline{\text{H\_OC1}}$	54	I	overcurrent sensing input for HC's downstream port 1
$\overline{\text{H\_OC2}}$	55	I	overcurrent sensing input for HC's downstream port 2
V_{CC}	56	-	digital power supply voltage input (3.0 to 3.6 V or 4.75 to 5.25 V). This pin supplies the internal 3.3 V regulator input. When connected to 5 V, the internal regulator will output 3.3 V to pins $V_{reg(3.3)}$, V_{hold1} and V_{hold2}. When connected to 3.3 V, it will bypass the internal regulator.
AGND	57	-	analog ground
$V_{reg(3.3)}$	58	-	internal 3.3 V regulator output. When the V_{CC} pin is connected to 5 V, this pin outputs 3.3 V. When the V_{CC} pin is connected to 3.3 V, then this pin should also be connected to 3.3 V.
A0	59	I	address selection input 0; selects command (A0 = 1) or data (A0 = 0)
A1	60	I	address selection input 1; selects AutoMux switching to DC (A1 = 1) or AutoMux switching to HC (A1 = 0); see Table 3
n.c.	61	-	no connection
DGND	62	-	digital ground
D0	63	I/O	bit 0 of bidirectional data; slew-rate controlled; TTL input; three-state output
D1	64	I/O	bit 1 of bidirectional data; slew-rate controlled; TTL input; three-state output

[1] Symbol names with an overscore (e.g. $\overline{\text{NAME}}$) represent active LOW signals.

7. Functional description

7.1 PLL clock multiplier

A 6 to 48 MHz clock multiplier Phase-Locked Loop (PLL) is integrated on-chip. This allows for the use of a low-cost 6 MHz crystal, which also minimizes EMI. No external components are required for the operation of the PLL.

7.2 Bit clock recovery

The bit clock recovery circuit recovers the clock from the incoming USB data stream using a 4 times over-sampling principle. It is able to track jitter and frequency drift as specified in the *Universal Serial Bus Specification Rev. 2.0*.

7.3 Analog transceivers

Three sets of transceivers are embedded in the chip: two are used for downstream ports with USB connector type A; one is used for upstream port with USB connector type B.The integrated transceivers are compliant with the *Universal Serial Bus Specification Rev. 2.0*. They interface directly with the USB connectors and cables through external termination resistors.

7.4 Philips Serial Interface Engine (SIE)

The Philips SIE implements the full USB protocol layer. It is completely hardwired for speed and needs no firmware intervention. The functions of this block include: synchronization pattern recognition, parallel to serial conversion, bit (de)stuffing, CRC checking and generation, Packet IDentifier (PID) verification and generation, address recognition, handshake evaluation and generation. There are separate SIE in both the HC and the DC.

7.5 SoftConnect

The connection to the USB is accomplished by bringing D+ (for full-speed USB devices) HIGH through a 1.5 kΩ pull-up resistor. In the ISP1161's DC, the 1.5 kΩ pull-up resistor is integrated on-chip and is not connected to V_{CC} by default. The connection is established through a command sent by the external or system microprocessor. This allows the system microprocessor to complete its initialization sequence before deciding to establish connection with the USB. Re-initialization of the USB connection can also be performed without disconnecting the cable.

The ISP1161's DC will check for USB V_{BUS} availability before the connection can be established. V_{BUS} sensing is provided through pin D_VBUS.

Remark: The tolerance of the internal resistors is 25%. This is higher than the 5% tolerance specified by the USB specification. However, the overall voltage specification for the connection can still be met with a good margin. The decision to make use of this feature lies with the USB equipment designer.

7.6 GoodLink

Indication of a good USB connection is provided at pin $\overline{GL}$ through GoodLink technology. During enumeration the LED indicator will blink on momentarily. When the DC has been successfully enumerated (the device address is set), the LED indicator will remain permanently on. Upon each successful packet transfer (with ACK) to and from the ISP1161 the LED will blink off for 100 ms. During 'suspend' state the LED will remain off.

This feature provides a user-friendly indication of the status of the USB device, the connected hub and the USB traffic. It is a useful field diagnostics tool for isolating faulty equipment. It can therefore help to reduce field support and hotline overhead.

8. Microprocessor bus interface

8.1 Programmed I/O (PIO) addressing mode

A generic PIO interface is defined for speed and ease-of-use. It also allows direct interfacing to most microprocessors. To a microprocessor, the ISP1161 appears as a memory device with a 16-bit data bus and uses only two address lines: A1 and A0 to access the internal control registers and FIFO buffer RAM. Therefore, ISP1161 occupies only four I/O ports or four memory locations of a microprocessor. External microprocessors can read or write ISP1161's internal control registers and FIFO buffer RAM through the programmed I/O (PIO) operating mode. Figure 8 shows the programmed I/O interface between a microprocessor and ISP1161.

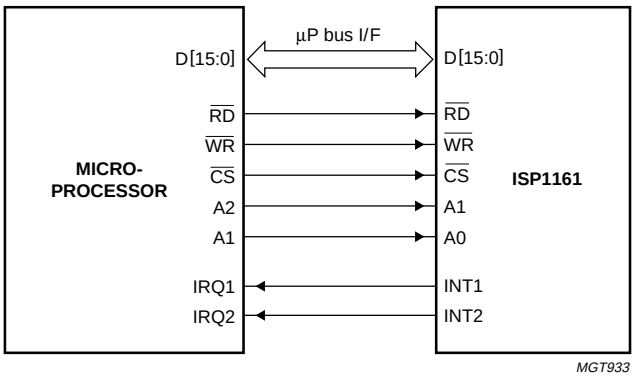


Fig 8. Programmed I/O interface between microprocessor and ISP1161.

8.2 DMA mode

The ISP1161 also provides DMA mode for external microprocessors to access its internal FIFO buffer RAM. Data can be transferred by DMA operation between a microprocessor's system memory and ISP1161's internal FIFO buffer RAM.

Remark: The DMA operation must be controlled by the external microprocessor system's DMA controller (Master).

Figure 9 shows the DMA interface between a microprocessor system and ISP1161. ISP1161 provides two DMA channels:

- DMA channel 1 (controlled by DREQ1 and DACK1 signals) is for the DMA transfer between a microprocessor's system memory and ISP1161 HC's internal FIFO buffer RAM
- DMA channel 2 (controlled by DREQ2 and DACK2 signals) is for the DMA transfer between a microprocessor's system memory and ISP1161 DC's internal FIFO buffer RAM

The EOT signal is an external end-of-transfer signal used to terminate the DMA transfer. Some microprocessors may not have this signal. In this case, ISP1161 provides an internal EOT signal to terminate the DMA transfer as well. Setting the HcDMAConfiguration register (21H - read, A1H - write) enables ISP1161's HC internal DMA counter for DMA transfer. When the DMA counter reaches the value that is set in the HcTransferCounter (22H - read, A2H - write) register to be used as the byte count of the DMA transfer, the internal EOT signal will be generated to terminate the DMA transfer.

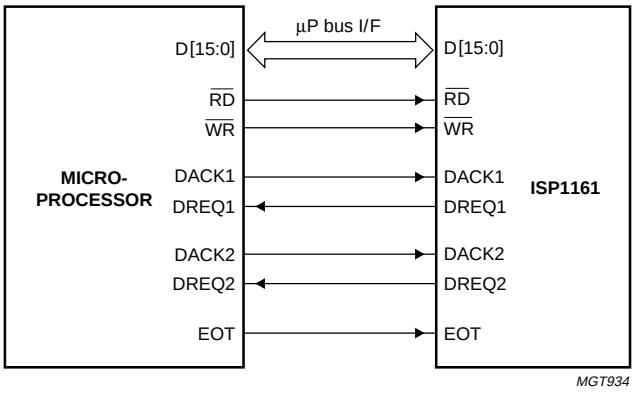


Fig 9. DMA interface between microprocessor and ISP1161.

8.3 Control registers access by PIO mode

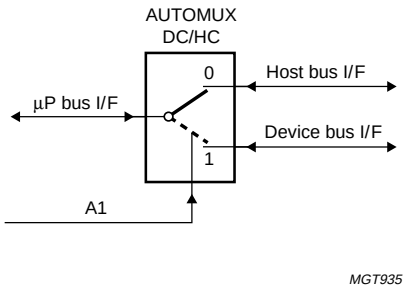
8.3.1 I/O port addressing

Table 3 shows ISP1161's I/O port addressing. Complete decoding of the I/O port address should include the chip select signal $\overline{CS}$ and the address lines A1 and A0. However, the direction of the access of the I/O ports is controlled by the $\overline{RD}$ and $\overline{WR}$ signals. When $\overline{RD}$ is LOW, the microprocessor reads data from ISP1161's data port. When $\overline{WR}$ is LOW, the microprocessor writes a command to the command port, or writes data to the data port.

Table 3: I/O port addressing

Port	Pin $\overline{CS}$	Pin A1	Pin A0	Access	Data bus width	Description
0	LOW	LOW	LOW	R/W	16 bits	HC data port
1	LOW	LOW	HIGH	W	16 bits	HC command port
2	LOW	HIGH	LOW	R/W	16 bits	DC data port
3	LOW	HIGH	HIGH	W	16 bits	DC command port

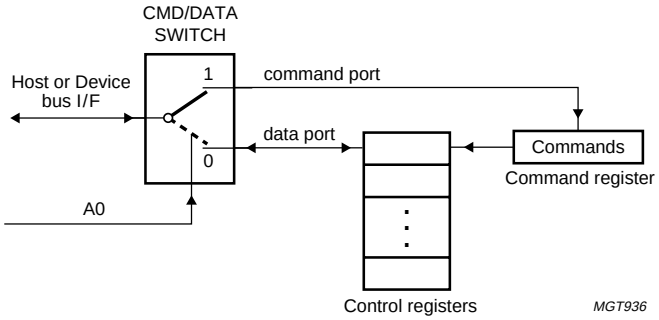
Figure 10 and Figure 11 illustrate how an external microprocessor accesses ISP1161's internal control registers.



When A1 = 0, microprocessor accesses the HC.

When A1 = 1, microprocessor accesses the DC.

Fig 10. A microprocessor accessing a HC or a DC via an automux switch.



When A0 = 0, microprocessor accesses the data port.

When A0 = 1, microprocessor accesses the command port.

Fig 11. Access to internal control registers.

8.3.2 Register access phases

ISP1161's register structure is a command-data register pair structure. A complete register access cycle comprises a command phase followed by a data phase. The command (also known as the index of a register) points the ISP1161 to the next register to be accessed. A command is 8 bits long. On a microprocessor's 16-bit data bus, a command occupies the lower byte, with the upper byte filled with zeros.

Figure 12 shows a complete 16-bit register access cycle for ISP1161. The microprocessor writes a command code to the command port, and then reads from or writes the data word to the data port. Take the example of a microprocessor attempting to read a chip's ID, which is saved in the HC's HcChipID register (27H, read only) where its command code is 27H, read only. The 16-bit register access cycle is therefore:

1. Microprocessor writes the command code of 27H (0027H in 16-bit width) to ISP1161's HC command port
2. Microprocessor reads the data word of the chip's ID from ISP1161's HC data port.

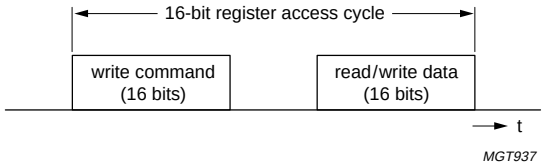


Fig 12. 16-bit register access cycle.

Most of ISP1161's internal control registers are 16 bits wide. Some of the internal control registers, however, have 32-bit width. Figure 13 shows how ISP1161's 32-bit internal control register is accessed. The complete cycle of accessing a 32-bit register consists of a command phase followed by two data phases. In the two data phases, the microprocessor should first read or write the lower 16-bit data, followed by the upper 16-bit data.

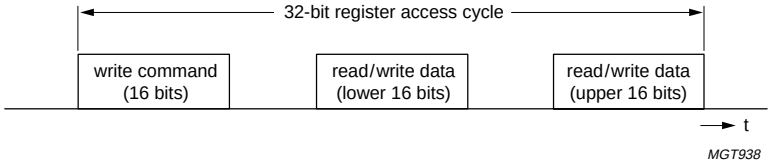


Fig 13. 32-bit register access cycle.

To further describe the complete access cycles of ISP1161's internal control registers, the status of some pin signals of the microprocessor bus interface are shown in Figure 14 and Figure 15 for the HC and the DC respectively.

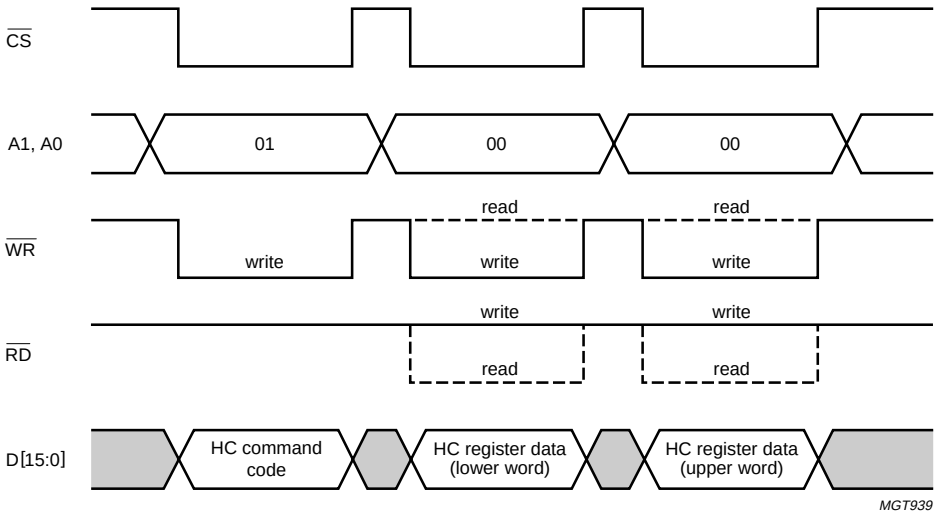


Fig 14. Accessing ISP1161 HC control registers.

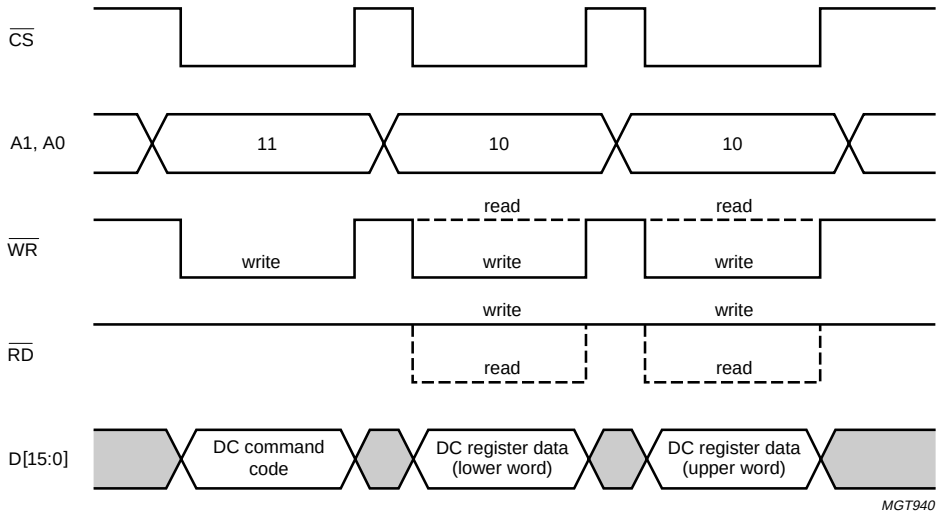


Fig 15. Accessing ISP1161 DC control registers.

8.4 FIFO buffer RAM by PIO mode

Since ISP1161's internal memory is structured as a FIFO buffer RAM, the FIFO buffer RAM is mapped to dedicated register fields. Therefore, accessing ISP1161's internal FIFO buffer RAM is just like accessing the internal control registers in multiple data phases.

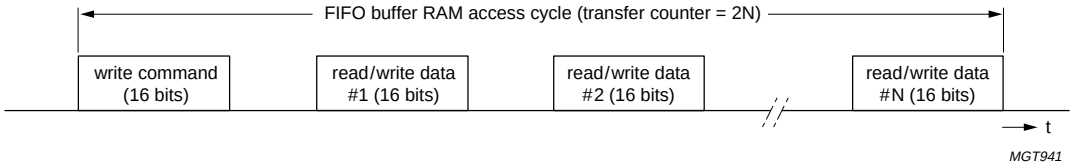


Fig 16. ISP1161's internal FIFO buffer RAM access cycle.

Figure 16 shows a complete access cycle of ISP1161's internal FIFO buffer RAM. For a write cycle, the microprocessor first writes the FIFO buffer RAM's command code to the command port, and then writes the data words one by one to the data port until half of the transfer's byte count is reached. The HcTransferCounter register (22H - read, A2H - write) is used to specify the byte count of a FIFO buffer RAM's read cycle or write cycle. Every access cycle must be in the same access direction. The read cycle procedure is similar to the write cycle.

For ISP1161 DC's FIFO buffer RAM access, see Section 11.

8.5 FIFO buffer RAM by DMA mode

The DMA interface between a microprocessor and ISP1161 is shown in Figure 9.

When doing a DMA transfer, at the beginning of every burst the ISP1161 outputs a DMA request to the microprocessor via the DREQ pin (DREQ1 for HC and DREQ2 for DC). After receiving this signal, the microprocessor will reply with a DMA

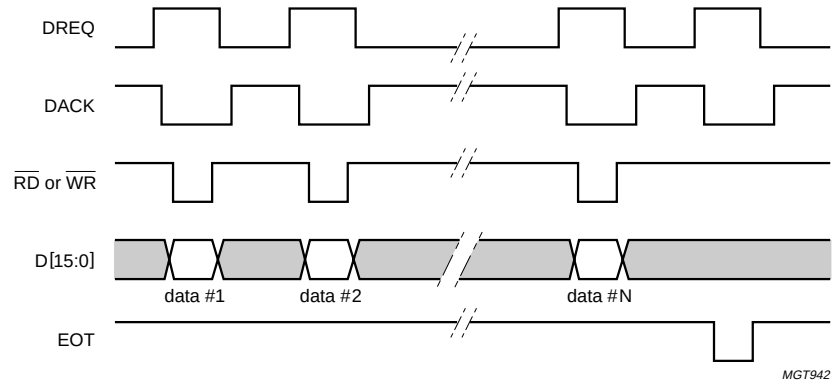
acknowledge to ISP1161 via the DACK pin (DACK1 for HC and DACK2 for DC), and at the same time, do the DMA transfer through the data bus. In the DMA mode, the microprocessor must still issue a $\overline{RD}$ or $\overline{WR}$ signal to ISP1161's $\overline{RD}$ or $\overline{WR}$ pin. ISP1161 will repeat the DMA cycles until it receives an EOT signal to terminate the DMA transfer.

ISP1161 supports both external EOT and internal EOT signals. The external EOT signal is received as input from ISP1161's EOT pin: it generally comes from the external microprocessor. The internal EOT signal is generated by ISP1161 internally.

To select either, set the DcDMAConfiguration registers. For example, for the HC, setting the DMACounterSelect bit of the HcDMAConfiguration register (21H - read, A1H - write) to logic 1 will enable the DMA counter for DMA transfer. When the DMA counter reaches the value of HcTransferCounter register, the internal EOT signal will be generated to terminate the DMA transfer.

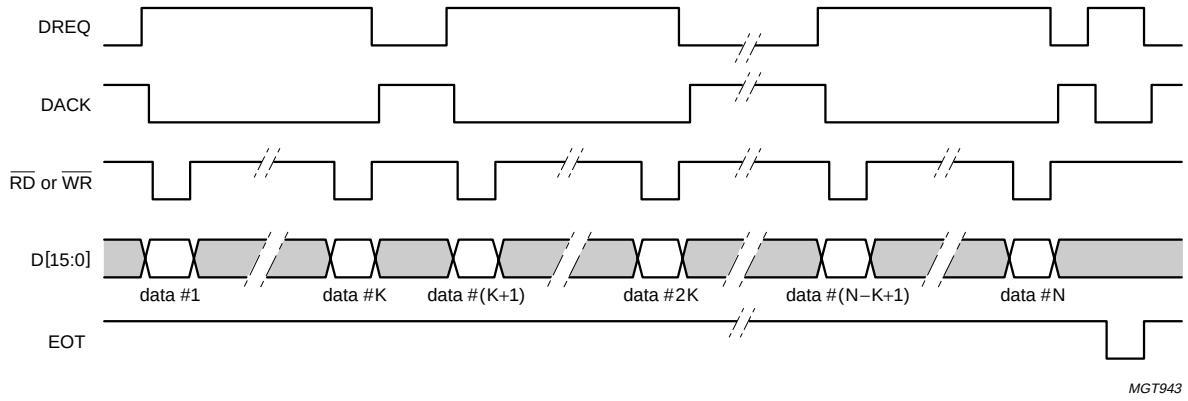
ISP1161 supports either single-cycle DMA operation or burst mode DMA operation.

In Figure 17 and Figure 18, the DMA transfer is configured such that DREQ is active HIGH and DACK is active LOW.



N = 1/2 byte count of transfer data.

Fig 17. DMA transfer in single-cycle mode.



N = 1/2 byte count of transfer data; K = number of cycles or burst.

Fig 18. DMA transfer in burst mode.

8.6 Interrupts

ISP1161 has separate interrupt request pins for USB HC (INT1) and USB DC (INT2).

8.6.1 Pin configuration

The interrupt output signals have four configuration modes:

Mode 0 Level trigger, active LOW (default at power-up)

Mode 1 Level trigger, active HIGH

Mode 2 Edge trigger, active LOW

Mode 3 Edge trigger, active HIGH.

Figure 19 shows these four interrupt configuration modes. They are programmable via the HcHardwareConfiguration register (see Section 10.4.1), which are also used to disable or enable the signals.

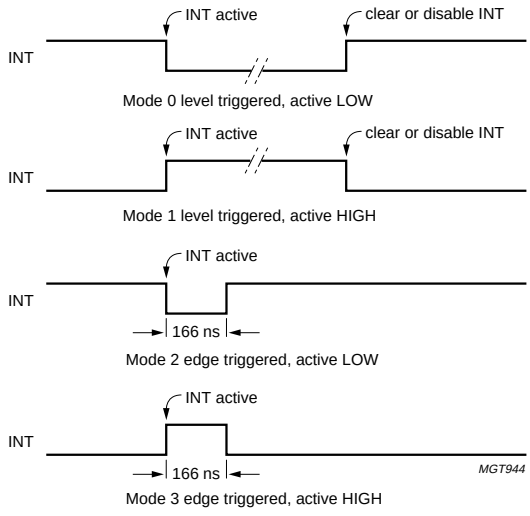


Fig 19. Interrupt pin operating modes.

8.6.2 HC's interrupt output pin (INT1)

To program the four configuration modes of the HC's interrupt output signal (INT1), set bits InterruptPinTrigger and InterruptOutputPolarity of the HcHardwareConfiguration register (20H - read, A0H - write). Bit InterruptPinEnable is used as the master enable setting for pin INT1.

INT1 has many interrupt events, as shown in Figure 20.

The interrupt events of the HcμPInterrupt register (24H - read, A4H - write) changes the status of pin INT1 when the corresponding bits of the HcμPInterruptEnable register (25H - read, A5H - write) and pin INT1's global enable bit (InterruptPinEnable of the HcHardwareConfiguration register) are all set to enable status.

However, events that come from the HcInterruptStatus register (03H - read, 83H - write) affect only the OPR_Reg bit of the HcμPInterrupt register. They cannot directly change the status of pin INT1.

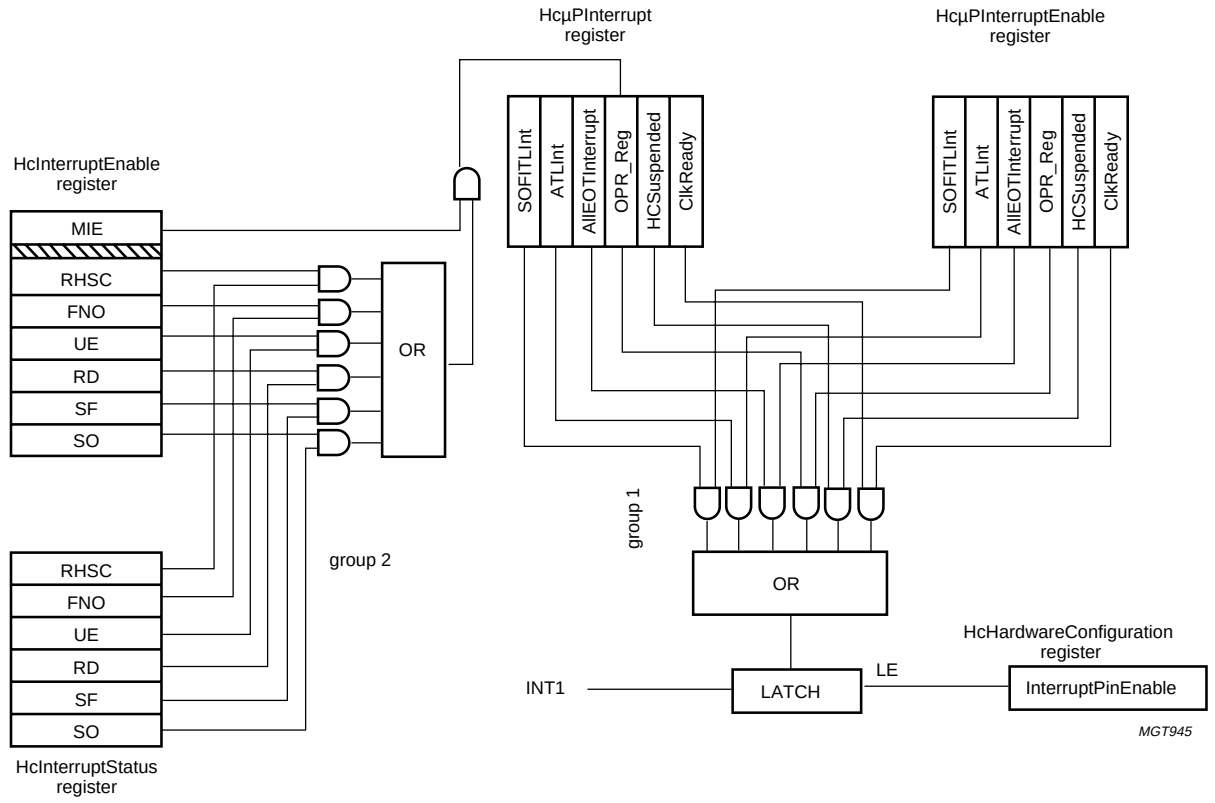


Fig 20. HC interrupt logic.

There are two groups of interrupts represented by group 1 and group 2 in Figure 20. A pair of registers control each group.

Group 2 contains six possible interrupt events (recorded in the HcInterruptStatus register). On occurrence of any of these events, the corresponding bit would be set to logic 1; and if the corresponding bit in the HcInterruptEnable register is also logic 1,

the 6-input OR gate would output logic 1. This output is AND-ed with the value of MIE (bit 31 of HcInterruptEnable). Logic 1 at the AND gate will cause bit OPR in the HcμPInterrupt register to be set to logic 1.

Group 1 contains six possible interrupt events, one of which is the output of group 2 interrupt sources. The HcμPInterrupt and HcμPInterruptEnable registers work in the same way as the HcInterruptStatus and HcInterruptEnable registers in the interrupt group 2. The output from the 6-input OR gate is connected to a latch, which is controlled by InterruptPinEnable (bit 0 of the HcHardwareConfiguration register).

In the event in which the software wishes to temporarily disable the interrupt output of the ISP1161 Host Controller, the following procedure should be followed:

1. Make sure that bit InterruptPinEnable in the HcHardwareConfiguration register is set to logic 1.
2. Clear all bits in the HcμPInterrupt register.
3. Set bit InterruptPinEnable to logic 0.

To re-enable the interrupt generation:

1. Set all bits in the HcμPInterrupt register.
2. Set bit InterruptPinEnable to logic 1.

Remark: Bit InterruptPinEnable in the HcHardwareConfiguration register latches the interrupt output. When this bit is set to logic 0, the interrupt output will remain unchanged, regardless of any operations on the interrupt control registers.

If INT1 is asserted, and the Host Controller Driver (HCD) wishes to temporarily mask off the INT signal without clearing the HcμPInterrupt register, the following procedure should be followed:

1. Make sure that bit InterruptPinEnable is set to logic 1.
2. Clear all bits in the HcμPInterruptEnable register.
3. Set bit InterruptPinEnable to logic 0.

To re-enable the interrupt generation:

1. Set all bits in the HcμPInterruptEnable register according to the HCD requirements.
2. Set bit InterruptPinEnable to logic 1.

8.6.3 DC's interrupt output pin (INT2)

The four configuration modes of DC's interrupt output pin INT2 can also be programmed by setting bits INTPOL and INTLVL of the DcHardwareConfiguration Register (BBH - read, BAH - write). Bit INTENA of the DcMode Register (B9H - read, B8H - write) is used to enable pin INT2. Figure 21 shows the relationship between the interrupt events and pin INT2.

Each of the indicated USB events is logged in a status bit of the DcInterrupt register. Corresponding bits in the DcInterruptEnable register determine whether or not an event will generate an interrupt.

Interrupts can be masked globally by means of the INTENA bit of the DcMode register (see Table 82).

The active level and signalling mode of the INT output is controlled by the INTPOL and INTLVL bits of the DcHardwareConfiguration register (see Table 84). Default settings after reset are active LOW and level mode. When pulse mode is selected, a pulse of 166 ns is generated when the OR-ed combination of all interrupt bits changes from logic 0 to logic 1.

Bits RESET, RESUME, EOT and SOF are cleared upon reading the DcInterrupt register. The endpoint bits (EP0OUT to EP14) are cleared by reading the associated DcEndpointStatus register.

Bit BUSTATUS follows the USB bus status exactly, allowing the firmware to get the current bus status when reading the DcInterrupt register.

SETUP and OUT token interrupts are generated after ISP1161's DC has acknowledged the associated data packet. In bulk transfer mode, ISP1161's DC will issue interrupts for every ACK received for an OUT token or transmitted for an IN token.

In isochronous mode, an interrupt is issued upon each packet transaction. The firmware must take care of timing synchronization with the host. This can be done via the Pseudo Start-Of-Frame (PSOF) interrupt, enabled via bit IEPSOF in the DcInterruptEnable register. If a Start-Of-Frame is lost, PSOF interrupts are generated every 1 ms. This allows the firmware to keep data transfer synchronized with the host. After 3 missed SOF events ISP1161's DC will enter 'suspend' state.

An alternative way of handling isochronous data transfer is to enable both the SOF and the PSOF interrupts and disable the interrupt for each isochronous endpoint.

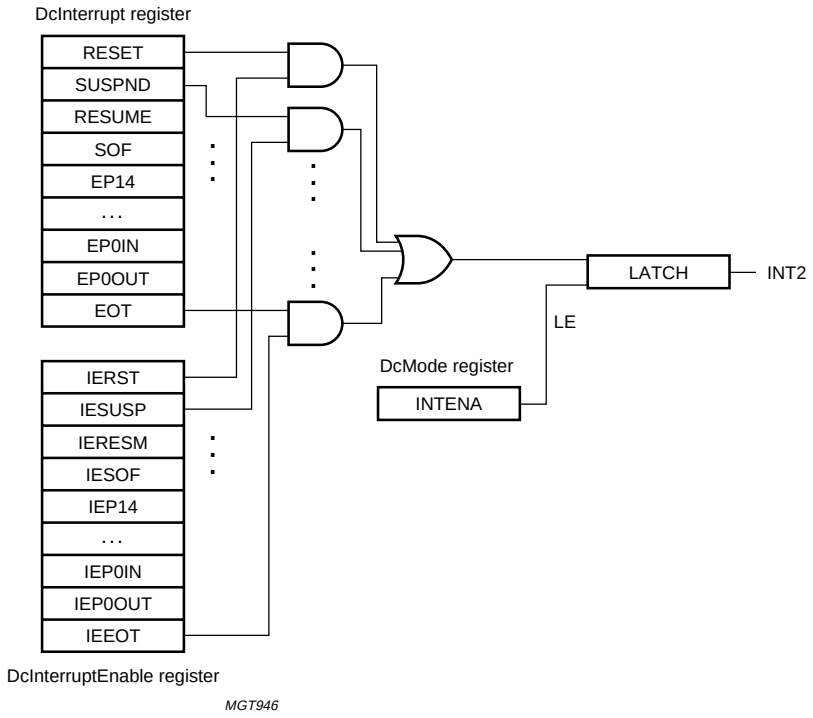
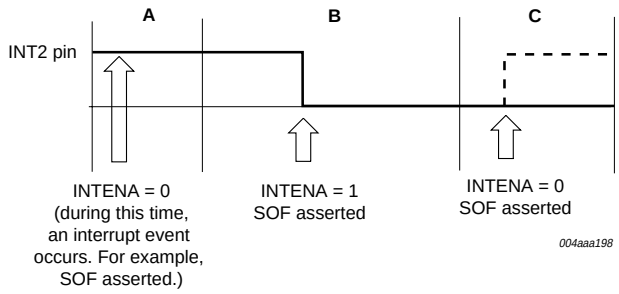


Fig 21. DC interrupt logic.

Interrupt control: Bit INTENA in the DcMode register is a global enable/disable bit. The behavior of this bit is given in Figure 22.



Pin INT2: HIGH = de-assert; LOW = assert (individual interrupts are enabled).

Fig 22. Behavior of bit INTENA bit.

Event A (see Figure 22): When an interrupt event occurs (for example, SOF interrupt) with bit INTENA set to logic 0, an interrupt will not be generated at pin INT2. However, it will be registered in the corresponding DcInterrupt register bit.

Event B (see Figure 22): When bit INTENA is set to logic 1, pin INT2 is asserted because bit SOF in the DcInterrupt register is already asserted.

Event C (see Figure 22): If the firmware sets bit INTENA to logic 0, pin INT2 will still be asserted. The bold dashed line shows the desired behavior of pin INT2.

De-assertion of pin INT2 can be achieved in the following manner. Bits[23:8] of the DcInterrupt register are endpoint interrupts. These interrupts are cleared on reading their respective DcEndpointStatus register. Bits[7:0] of the DcInterrupt register are bus status and EOT interrupts that are cleared on reading the DcInterrupt register. Make sure that bit INTENA is set to logic 1 when you perform the clear interrupt commands.

For more information on interrupt control, see Section 13.1.3, Section 13.1.5 and Section 13.3.6.

9. USB host controller (HC)

9.1 HC's four USB states

ISP1161's USB HC has four USB states–USB Operational, USB Reset, USB Suspend, and USB Resume– that define the HC's USB signaling and bus states responsibilities. The signals are visible to the HC (software) Driver via ISP1161 USB HC's control registers.

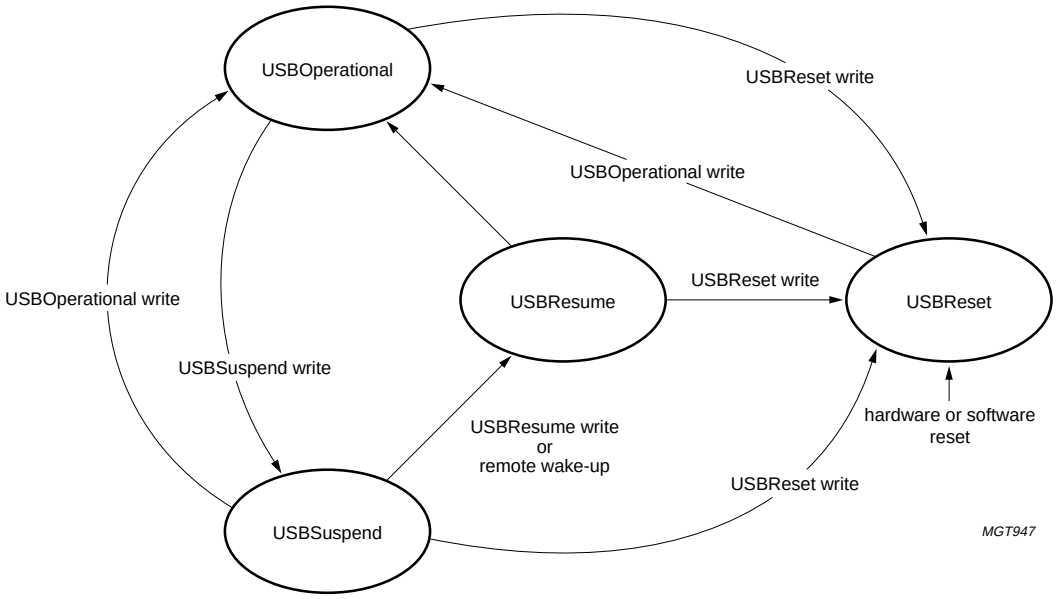


Fig 23. ISP1161 HC's USB states.

The USB states are reflected in the HostControllerFunctionalState field of the HcControl register (01H - read, 81H - write), which is located at bits 7 and 6 of the register.

The Host Controller Driver (HCD) can perform only the USB state transitions shown in Figure 23.

Remark: The Software Reset in Figure 23 is not caused by the HcSoftwareReset command. It is caused by the HostControllerReset field of the HcCommandStatus register (02H - read, 82H - write).

9.2 Generating USB traffic

USB traffic can be generated only when the ISP1161 USB HC is under the USB Operational State. Therefore, the HCD must set HostControllerFunctionalState field of the HcControl register before generating USB traffic.

A simplistic flow diagram showing when and how to generate USB traffic is shown in Figure 24. For greater accuracy, refer to both the *Universal Serial Bus Specification Revision 2.0* about the protocol and ISP1161 USB HC's register usage.

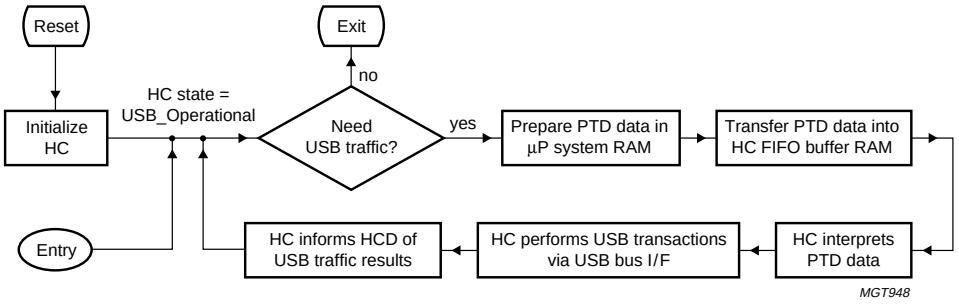


Fig 24. ISP1161 HC operating flow.

The USB traffic blocks are:

- **Reset**
  This includes hardware reset by pin $\overline{\text{RESET}}$ and software reset by the HcSoftwareReset command (A9H). The reset function will clear all the HC's internal control registers to their reset status. After reset, the HCD must initialize the ISP1161 USB HC by setting some registers.
- **Initialize HC**
  It includes:
  - Setting the physical size for the HC's internal FIFO buffer RAM by setting the HcITLBufferLength register (2AH - read, AAH - write) and the HcATLBufferLength register (2BH - read, ABH - write)
  - Setting the HcHardwareConfiguration register according to requirements
  - Clearing interrupt events, if required
  - Enabling interrupt events, if required
  - Setting the HcFmInterval register (0DH - read, 8DH - write)
  - Setting the HC's root hub registers
  - Setting the HcControl register to move the HC into USB Operational state.

  See also Section 9.5.
- **Entry**
  The normal entry point. The microprocessor returns to this point when there are HC requests.
- **Need USB traffic**
  USB devices need the HC to generate USB traffic when they have USB traffic requests such as:
  - Connecting to or disconnecting from the downstream ports
  - Issuing the Resume signal to the HC

  To generate USB traffic, the HCD must enter the USB transaction loop.
- **Prepare PTD data in µP system RAM**
  The communication channel between the HCD and ISP1161's USB HC is in the form of Philips Transfer Descriptor (PTD) data. The PTD data provides USB traffic information about the commands, status, and USB data packets.

The physical storage media of PTD data for the HCD is the microprocessor's system RAM. For ISP1161's USB HC, it is the ISP1161's internal FIFO buffer RAM.

The HCD prepares PTD data in the microprocessor's system RAM for transfer to ISP1161's HC internal FIFO buffer RAM.

- **Transfer PTD data into HC's FIFO buffer RAM**
  When PTD data is ready in the microprocessor's system RAM, the HCD must transfer the PTD data from the microprocessor's system RAM into ISP1161's internal FIFO buffer RAM.
- **HC interprets PTD data**
  The HC determines what USB transactions are required based on the PTD data that have been transferred into the internal FIFO buffer RAM.
- **HC performs USB transactions via USB interface**
  The HC performs the USB transactions with the specified USB device endpoint through the USB bus interface.
- **HC informs HCD the USB traffic results**
  The USB transaction status and the feedback from the specified USB device endpoint will be put back into the ISP1161's HC internal FIFO buffer RAM in PTD data format. The HCD can read back the PTD data from the internal FIFO buffer RAM.

9.3 PTD data structure

The Philips Transfer Descriptor (PTD) data structure provides a communication channel between the HCD and the ISP1161's USB HC. PTD data contains information required by the USB traffic. PTD data consists of a PTD followed by its payload data, as shown in Figure 25.

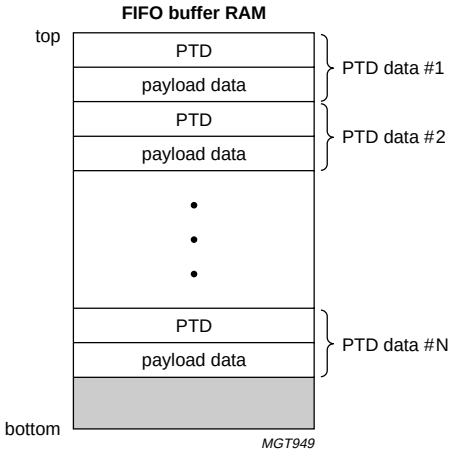


Fig 25. PTD data in FIFO buffer RAM.

The PTD data structure is used by the HC to define a buffer of data that will be moved to or from an endpoint in the USB device. This data buffer is set up for the current frame (1 ms frame) by the firmware, the HCD. The payload data for every transfer in the frame must have a PTD as a header to describe the characteristic of the transfer. PTD data is Dword aligned.

9.3.1 PTD data header definition

The PTD forms the header of the PTD data. It tells the HC the transfer type, where the payload data should go, and the payload data's actual size. A PTD is an 8-byte data structure that is very important for HCD programming.

Table 4: Philips Transfer Descriptor (PTD): bit allocation

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				DirectionPID[1:0]		TotalBytes[9:8]	
Byte 6	Format	FunctionAddress[6:0]						
Byte 7	reserved							

Table 5: Philips Transfer Descriptor (PTD): bit description

Symbol	Access	Description		
ActualBytes[9:0]	R/W	Contains the number of bytes that were transferred for this PTD		
CompletionCode[3:0]	R/W	0000	NoError	General TD or isochronous data packet processing completed with no detected errors.
		0001	CRC	Last data packet from endpoint contained a CRC error.
		0010	BitStuffing	Last data packet from endpoint contained a bit stuffing violation.
		0011	DataToggleMismatch	Last packet from endpoint had data toggle PID that did not match the expected value.
		0100	Stall	TD was moved to the Done queue because the endpoint returned a STALL PID.
		0101	DeviceNotResponding	Device did not respond to token (IN) or did not provide a handshake (OUT).
		0110	PIDCheckFailure	Check bits on PID from endpoint failed on data PID (IN) or handshake (OUT).
		0111	UnexpectedPID	Received PID was not valid when encountered or PID value is not defined.
		1000	DataOverrun	The amount of data returned by the endpoint exceeded either the size of the maximum data packet allowed from the endpoint (found in MaximumPacketSize field of ED) or the remaining buffer size.
		1001	DataUnderrun	The endpoint returned is less than MaximumPacketSize and that amount was not sufficient to fill the specified buffer.
		1010	reserved	
		1011	reserved	
		1100	BufferOverrun	During an IN, the HC received data from an endpoint faster than it could be written to system memory.
		1101	BufferUnderrun	During an OUT, the HC could not retrieve data from the system memory fast enough to keep up with the USB data rate.
Active	R/W	Set to logic 1 by firmware to enable the execution of transactions by the HC. When the transaction associated with this descriptor is completed, the HC sets this bit to logic 0, indicating that a transaction for this element should not be executed when it is next encountered in the schedule.		
Toggle	R/W	Used to generate or compare the data PID value (DATA0 or DATA1). It is updated after each successful transmission or reception of a data packet.		
MaxPacketSize[9:0]	R	The maximum number of bytes that can be sent to or received from the endpoint in a single data packet.		
EndpointNumber[3:0]	R	USB address of the endpoint within the function.		
Last	R	Last PTD of a list (ITL or ATL). Logic 1 indicates that the PTD is the last PTD.		
Speed	R	Speed of the endpoint: **0 —** full-speed **1 —** low-speed		
TotalBytes[9:0]	R	Specifies the total number of bytes to be transferred with this data structure. For Bulk and Control only, this can be greater than MaximumPacketSize.		

Table 5: Philips Transfer Descriptor (PTD): bit description...*continued*

Symbol	Access	Description
DirectionPID[1:0]	R	**00 —** SETUP **01 —** OUT **10 —** IN **11 —** reserved
Format	R	The format of this data structure. If this is a Control, Bulk or Interrupt endpoint, then bit Format = 0. If this is an Isochronous endpoint, then bit Format = 1.
FunctionAddress[6:0]	R	This is the USB address of the function containing the endpoint that this PTD refers to.

9.4 HC's internal FIFO buffer RAM structure

9.4.1 Partitions

According to the *Universal Serial Bus Specification Rev. 2.0*, there are four types of USB data transfers: Control, Bulk, Interrupt and Isochronous.

The HC's internal FIFO buffer RAM is of a physical size of 4 kbytes. This internal FIFO buffer RAM is used for transferring data between the microprocessor and USB peripheral devices. This on-chip buffer RAM can be partitioned into two areas: Acknowledged Transfer List (ATL) buffer and Isochronous (ISO) Transfer List (ITL) buffer. The ITL buffer is a Ping-Pong structured FIFO buffer RAM that is used to keep the payload data and their PTD header for Isochronous transfers. The ATL buffer is a non Ping-Pong structured FIFO buffer RAM that is used for the other three types of transfers.

For the ITL buffer, it can be further partitioned into ITL0 and ITL1 for the Ping-Pong structure. The ITL0 buffer and ITL1 buffer always have the same size. The microprocessor can put ISO data into either the ITL0 buffer or the ITL1 buffer. When the microprocessor accesses an ITL buffer, the HC can take over another ITL buffer at the same time. This architecture can improve the ISO transfer performance.

The HCD can assign the logical size for ATL buffer and ITL buffers at any time, but normally at initialization after power-on reset, by setting the HcATLBufferLength register (2BH - read, ABH - write) and HcITLBufferLength register (2AH - read, AAH - write), respectively. However, the total length (ATL buffer + ITL buffer) should not exceed 4 kbytes, the maximum RAM size. Figure 26 shows the partitions of the internal FIFO buffer RAM. When assigning buffer RAM sizes, follow this formula:

ATL buffer length + 2 × (ITL buffer size) ≤ 1000H (that is 4 kbytes)

where: ITL buffer size is ITL0 buffer length or ITL1 buffer length.

The following assignments are examples of legal uses of the internal FIFO buffer RAM:

- ATL buffer length = 800H, ITL buffer length = 400H.
  This is the maximum use of the internal FIFO buffer RAM.
- ATL buffer length = 400H, ITL buffer length = 200H.
  This is insufficient use of the internal FIFO buffer RAM.
- ATL buffer length = 1000H, ITL buffer length = 0H.
  This will use the internal FIFO buffer RAM for only ATL transfers.

- ATL buffer length = 0H, ITL buffer length = 800H.
  This will use the internal FIFO buffer RAM for only ISO transfers.

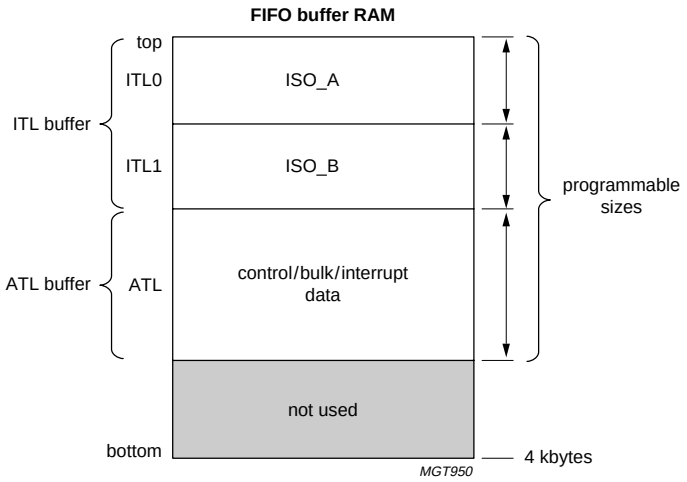


Fig 26. HC Internal FIFO buffer RAM partitions.

The actual requirement for the buffer RAM may not reach the maximum size. You can make your selection based on your application. The following are some calculations of the ISO_A or ISO_B space for a frame of data:

- Maximum number of useful data sent during one USB frame is 1280 bytes (20 ISO packets of 64 bytes). Total RAM size needed for this is:
  $20 \times 8 + 1280 = 1440$ bytes.
- Maximum number of packets for different endpoints sent during one USB frame is 150 (150 ISO packets of 1 byte). Total RAM size needed is:
  $150 \times 8 + 150 \times 1 = 1350$ bytes.
- The Ping buffer RAM (ITL0) and the Pong buffer RAM (ITL1) have a maximum size of 2 kbytes each. All data needed for one frame can be stored in the Ping or the Pong buffer RAM.

When the embedded system wants to initiate a transfer to the USB bus, the data needed for one frame is transferred to the ATL buffer or ITL buffer. The microprocessor detects the buffer status through the interrupt routines. When the HcBufferStatus register (2CH - read only) indicates that buffer is empty, then the microprocessor can write data into the buffer. When the HcBufferStatus register indicates that buffer is full, that is data is ready on the buffer, the microprocessor needs to read data from the buffer.

During every 1 ms, there might be many events to generate interrupt requests to the microprocessor for data transfer or status retrieval. However, each of the interrupt types defined in this specification can be enabled or disabled by setting the HcμPInterruptEnable register bits accordingly.

The data transfer can be done via PIO mode or DMA mode. The data transfer rate can go up to 15 Mbyte/s. In DMA operation, single-cycle or multi-cycle burst modes are supported. For the multi-cycle burst mode, 1, 4 or 8 cycles per burst are supported for ISP1161.

9.4.2 Data organization

PTD data is used for every data transfer between a microprocessor and the USB bus, and the PTD data resides in the buffer RAM. For an OUT or SETUP transfer, the payload data is placed just after the PTD, after which the next PTD is placed. For an IN transfer, some RAM space is reserved for receiving a number of bytes that is equal to the total bytes of the transfer. After this, the next PTD and its payload data are placed (see Figure 27).

Remark: The PTD is defined for both ATL and ITL type data transfer. For ITL, the PTD data should be put into ITL buffer RAM, the ISP1161 takes care of the Ping-Pong action for the ITL buffer RAM access.

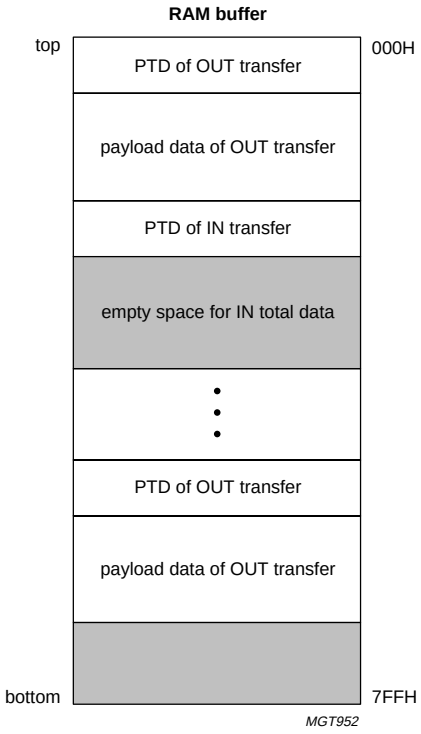


Fig 27. Buffer RAM data organization.

The PTD data (PTD header and its payload data) is a structure of Dword (double-word or 4 bytes) alignment. This means that the memory address is organized in steps of 4 bytes. Therefore, the first byte of every PTD and the first byte of every payload data are located at an address which is a multiple of 4. Figure 28 illustrates an example in which the first payload data is 14 bytes long, meaning that the last byte of the payload data is at the location 15H. The next addresses (16H and 17H) are not multiples of 4. Therefore, the first byte of the next PTD will be located at the next multiple-of-four address: 18H.

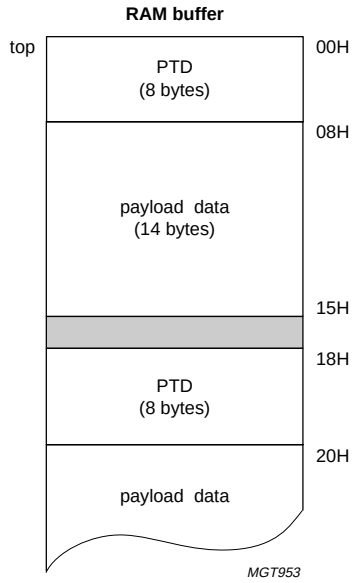


Fig 28. PTD data with Dword alignment in buffer RAM.

9.4.3 Operation and C program example

Figure 29 shows the block diagram for internal FIFO buffer RAM operations by PIO mode. ISP1161 provides one register as the access port for each buffer RAM. For the ITL buffer RAM, the access port is the ITLBufferPort register (40H - read, C0H - write). For the ATL buffer RAM, the access port is the ATLBufferPort register (41H - read, C1H - write). The buffer RAM is an array of bytes (8 bits) while the access port is a 16-bit register. Therefore, each read/write operation on the port accesses two consecutive memory locations, incrementing the pointer of the internal buffer RAM by two.

The lower byte of the access port register corresponds to the data byte at the even location of the buffer RAM, and the higher byte in the access port register corresponds to the other data byte at the odd location of the buffer RAM. Regardless of the number of data bytes to be transferred, the command code must be issued merely once, and it will be followed by a number of accesses of the data port (see Section 8.4).

When the pointer of the buffer RAM reaches the value of the HcTransferCounter register, an internal EOT signal will be generated to set bit 2 (AllEOTInterrupt) of the HcμPInterrupt register and update the HcBufferStatus register, to indicate that the whole data transfer has been completed.

For ITL buffer RAM, every start of frame (SOF) signal (1 ms) will cause toggling between ITL0 and ITL1 but this depends on the buffer status. If both ITL0BufferFull and ITL1BufferFull of the HcBufferStatus register are already logic 1, meaning that both ITL0 and ITL1 buffer RAMs are full, the toggling will not happen. In this case, the microprocessor will always have access to ITL1.

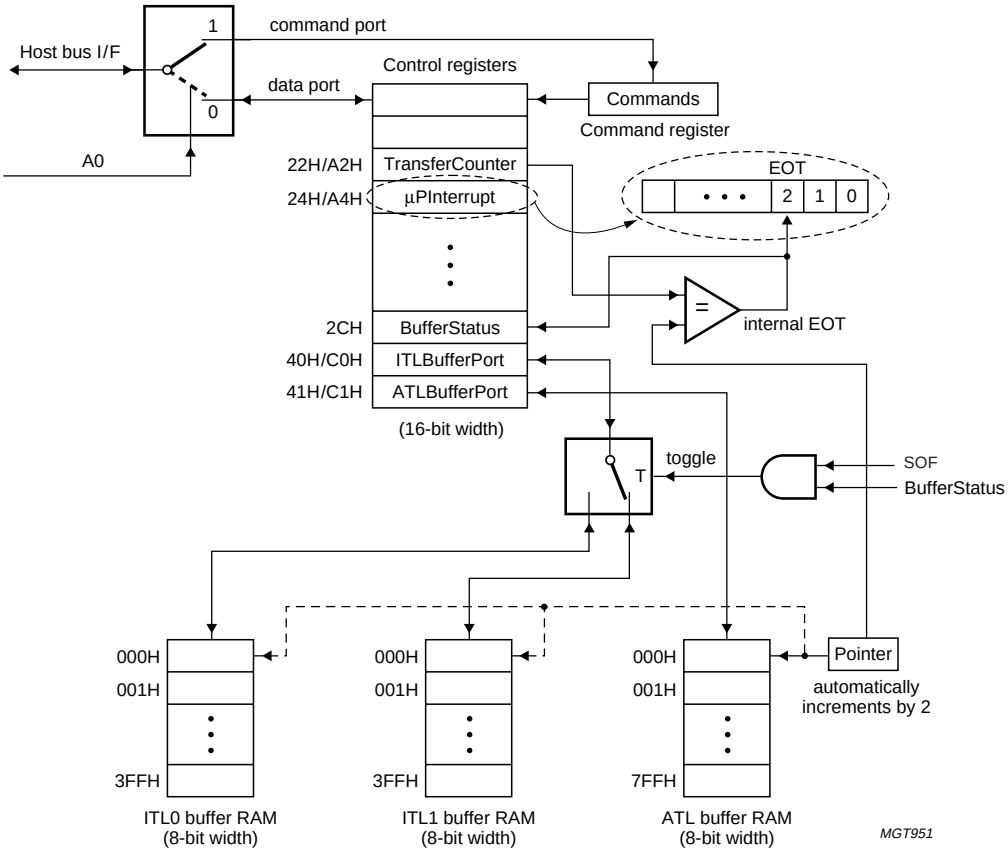


Fig 29. PIO access to Internal FIFO buffer RAM.

Following is an example of a C program that shows how to write data into the ATL buffer RAM. The total number of data bytes to be transferred is 80 (decimal) which will be set into the HcTransferCounter register as 50H. The data consists of four types of PTD data:

1. The first PTD header (IN) is 8 bytes, followed by 16 bytes of space reserved for its payload data
2. The second PTD header (IN) is also 8 bytes, followed by 8 bytes of space reserved for its payload data
3. The third PTD header (OUT) is 8 bytes, followed by 16 bytes of payload data with values beginning from 0H to FH incrementing by 1
4. The fourth PTD header (OUT) is also 8 bytes, followed by 8 bytes of payload data with values beginning from 0H to EH incrementing by 2.

In all PTD's, we assign device address 5 and endpoint 1. ActualBytes is always zero (0). TotalBytes equals the number of payload data bytes.

Table 6 shows the results after running this program.

However, if communication with a peripheral USB device is desired, the device should be connected to the downstream port and pass enumeration.

```
// The example program for writing ATL buffer RAM
#include <conio.h>
#include <stdio.h>
#include <dos.h>

// Define register commands
#define wHcTransferCounter 0x22
#define wHcuPInterrupt 0x24
#define wHcATLBufferLength 0x2b
#define wHcBufferStatus 0x2c

// Define I/O Port Address for HC
#define HcDataPort 0x290
#define HcCmdPort 0x292

// Declare external functions to be used
unsigned int HcRegRead(unsigned int wIndex);
void HcRegWrite(unsigned int wIndex,unsigned int wValue);

void main(void)
{
unsigned int i;
unsigned int wCount,wData;

// Prepare PTD data to be written into HC ATL buffer RAM:
unsigned int PTDData[0x28]=
{

0x0800,0x1010,0x0810,0x0005, // PTD header for IN token #1

// Reserved space for payload data of IN token #1
0x0000,0x0000,0x0000,0x0000, 0x0000,0x0000,0x0000,0x0000,

0x0800,0x1008,0x0808,0x0005, // PTD header for IN token #2

// Reserved space for payload data of IN token #2
0x0000,0x0000,0x0000,0x0000,

0x0800,0x1010,0x0410,0x0005, // PTD header for OUT token #1

0x0100,0x0302,0x0504,0x0706, // Payload data for OUT token #1
0x0908,0x0b0a,0x0d0c,0x0f0e,

0x0800,0x1808,0x0408,0x0005, // PTD header for OUT token #2

0x0200,0x0604,0x0a08,0x0e0c // Payload data for OUT token #2
};
HcRegWrite(wHcuPInterrupt,0x04); // Clear EOT interrupt bit
// HcRegWrite(wHcITLBufferLength,0x0);
HcRegWrite(wHcATLBufferLength,0x1000); // RAM full use for ATL
// Set the number of bytes to be transferred
HcRegWrite(wHcTransferCounter,0x50);

wCount = 0x28; // Get word count outport
(HcCmdPort,0x00c1); // Command for ATL buffer write

// write 80 (0x50) bytes of data into ATL buffer RAM
for (i=0;i<wCount;i++)
```

```
 {
 outport(HcDataPort,PTDData[i]);
 };

// Check EOT interrupt bit
 wData = HcRegRead(wHcuPInterrupt);
 printf("\n HC Interrupt Status = %xH.\n",wData);

// Check Buffer status register
 wData = HcRegRead(wHcBufferStatus);
 printf("\n HC Buffer Status = %xH.\n",wData);
 }

//
// Read HC 16-bit registers
//
unsigned int HcRegRead(unsigned int wIndex)
 { unsigned int wValue;

 outport(HcCmdPort,wIndex & 0x7f);
 wValue = inport(HcDataPort);

 return(wValue);

}
//
// Write HC 16-bit registers
//
void HcRegWrite(unsigned int wIndex,unsigned int wValue)
 {
 outport(HcCmdPort,wIndex | 0x80);
 outport(HcDataPort,wValue);
 }
```

Table 6: Run results of the C program example

Observed items	HC not initialized and not under Operational state	HC initialized and under Operational state	Comments
HcμPinterrupt register			
Bit 1 (ATLInt)	0	1	microprocessor must read ATL
Bit 2 (AllEOTInterrupt)	1	1	transfer completed
HcBufferStatus register			
Bit 2 (ATLBufferFull)	1	1	transfer completed
Bit 5 (ATLBufferDone)	0	1	PTD data processed by HC
USB			
Traffic on USB Bus	no	yes	OUT packets can be seen

9.5 HC's operational model

Upon power up, the HCD sets up all operational registers (32 bits). The FSLargestDataPacket field (bits 30 to 16) of the HcFmInterval register (0DH - read, 8DH - write) and the HcLSThreshold register (11H - read, 91H - write) determine the end of the frame for full-speed and low-speed packets. By programming these fields, the effective USB bus usage can be changed. Furthermore, the size of the ITL buffers (HcITLBufferLength, 2AH - read, AAH - write) is programmed.

In the case when a USB frame contains both ISO and AT packets, two interrupts will be generated per frame.

One interrupt is issued concurrently with the SOF. This interrupt (the ITLint is set in the HcμPInterrupt register) triggers reading and writing of the ITL by the microprocessor, after which the interrupt is cleared by the microprocessor.

Next the programmable AT Interrupt (the ATLint is set in the HcμPInterrupt register) is issued, which triggers reading and writing of the ITL by the microprocessor, after which the interrupt is cleared by the microprocessor. If the microprocessor cannot handle the ISO interrupt before the next ISO interrupt, disrupted ISO traffic can result.

To be able to send more than one packet to the same Control or Bulk endpoint in the same frame, an active bit and a 'TotalBytes of transfer' field are introduced (see Table 5). The active bit is cleared only if all data of the Philips Transfer Descriptor (PTD) is transferred or if a transaction at that endpoint contained a fatal error. If all PTD of the ATL are serviced once and the frame is not over yet, the HC starts looking for a PTD with the active bit still set. If such a PTD is found and there is still enough time in this frame, another transaction is started on the USB bus for this endpoint.

For ISO processing, the HCD has also to take care of the BufferStatus register (2CH - read only) for the ITL buffer RAM operations. After the HCD writes ISO data into ITL buffer RAM, the ITL0BufferFull or ITL1BufferFull bit (depends if it is ITL0 or ITL1) will be set to logic 1.

After the HC processes the ISO data in the ITL buffer RAM then the corresponding ITL0BufferDone or ITL1BufferDone bit will automatically be set to logic 1.

The HCD can clear the buffer status bits by a read of the ITL buffer RAM, and this must be done within the 1 ms frame from which the ITL0BufferDone or ITL1BufferDone was set. Failure to do so will cause the ISO processing to stop and a power on reset or software reset will have to be applied to the HC.

For example, in the first frame, for the HCD doing a write of ISO-A data into the ITL0 buffer. This will cause the BufferStatus register to show that the ITL0 buffer is full by setting the ITL0BufferFull bit to logic 1. At this stage the HCD cannot write ISO data into the ITL0 buffer RAM again.

In the second frame, the Host Controller will process the ISO-A data in the ITL0 buffer. At the same time, the HCD can write ISO-B data into ITL1 buffer. When the next SOF comes (the beginning of the third frame), both the ITL1BufferFull and ITL0BufferDone are automatically set to logic 1.

In the third frame the HCD has to read ITL0 buffer at least two bytes (one word) to clear **both** the ITL0BufferFull and ITL0BufferDone bits. If both are not cleared, when the next SOF comes (the beginning of the fourth frame) the ITL0BufferDone bit will be

cleared automatically, but the ITL0BufferFull bit remains at logic 1 and the ITL0BufferFull bit will be unable to be cleared. This condition will disable the HCD from writing ISO data into the ITL0 buffer again and ISO processing be unable to be carried on. This also applies to the ITL1 buffer because the ITL0 and ITL1 are Ping-Pong structured buffers. To recover from this state, the power on reset or software reset will have to be applied on the HC.

9.5.1 Time domain behavior

In example 1 (see Figure 30) the CPU is fast enough to read back and download a scenario before the next interrupt. It should be noted that on the ISO interrupt of frame N:

- The ISO packet for frame N + 1 will be written
- The AT packet for frame N + 1 will be written.

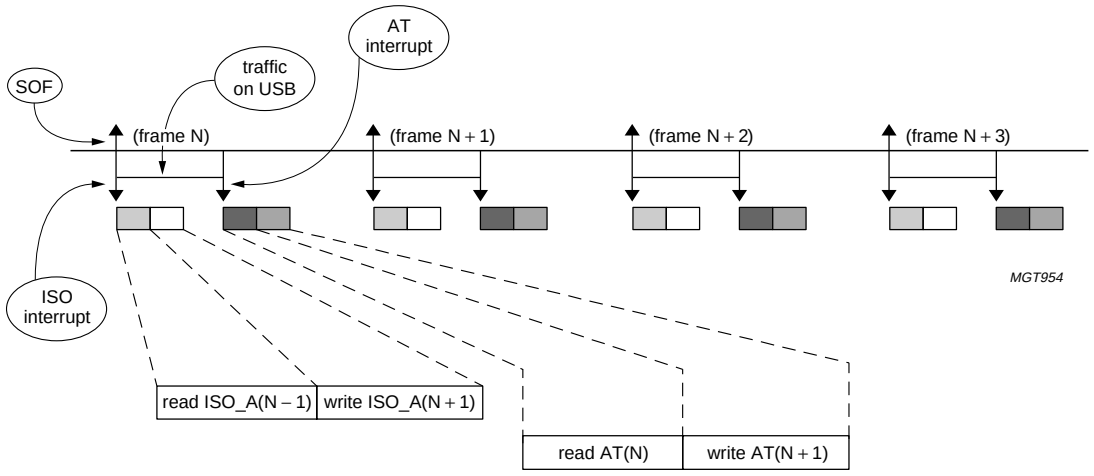


Fig 30. HC time domain behavior: example 1.

In example 2 (see Figure 31), the microprocessor is still busy transferring the AT data when the ISO interrupt of the next frame (N + 1) is raised. As a result, there will be no AT traffic in frame N + 1. The HC should not raise an AT interrupt in frame N + 1. The AT part is simply postponed until frame N + 2. On the AT N + 2 interrupt the transfer mechanism is back to normal operation. This simple mechanism ensures, among other things, that Control transfers are not dropped systematically from the USB in case of an overloaded microprocessor.

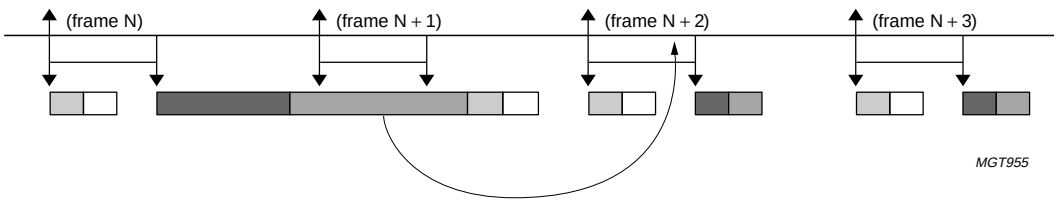


Fig 31. HC time domain behavior: example 2.

In example 3 (see Figure 32), the ISO part is still being written while the Start of Frame (SOF) of the next frame has occurred. This will result in undefined behavior for the ISO data on the USB bus in frame N + 1 (depending on the exact timing data is corrupted or not). The HC should not raise an AT interrupt in frame N + 1.

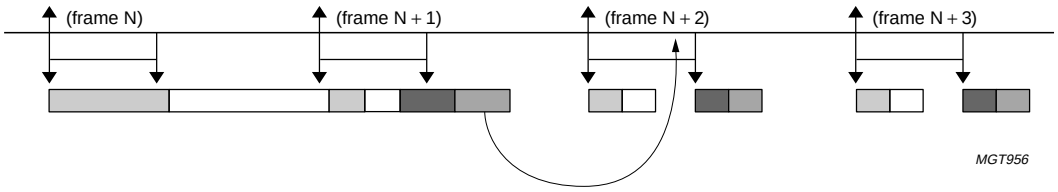


Fig 32. HC time domain behavior: example 3.

9.5.2 Control transaction limitations

The different phases of a control transfer (SETUP, Data and Status) should never be put in the same ATL.

9.6 Microprocessor loading

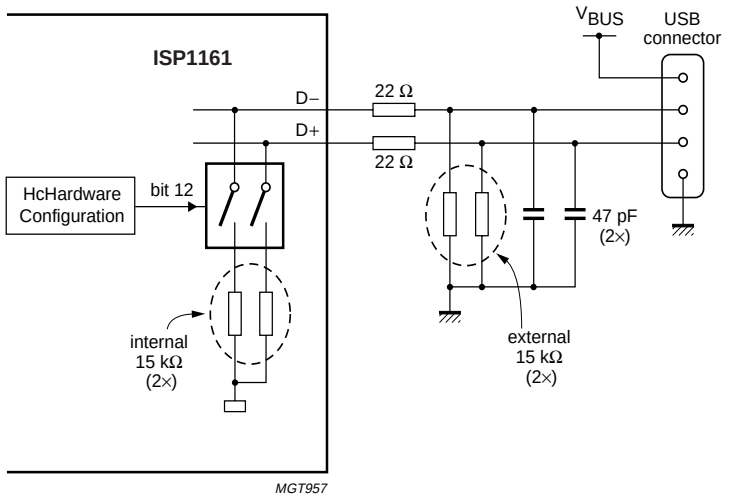
The maximum amount of data that can be transferred for an endpoint during one frame is 1023 bytes. The number of USB packets that are needed for this batch of data depends on the Maximum Packet Size that is specified.

The HCD has to schedule the transactions in a frame. On the other hand, the microprocessor must have the ability to handle the interrupts coming from HC every 1 ms. It must also be able to do the scheduling for the next frame, reading the frame information from and writing the next frame information to the buffer RAM in the time between the end of the current frame and the start of the next frame.

9.7 Internal pull-down resistors for downstream ports

There are four internal 15 kΩ pull-down resistors built in ISP1161 for the two downstream ports: two resistors for each port. These resistors are software selectable by programming bit 12 (2_DownstreamPort15Kresistorsel) of the HcHardwareConfiguration register (20H - read, A0H - write). When bit 12 is cleared to logic 0, it means that external 15 kΩ pull-down resistors should be used. Bit 12 is set to logic 1 that indicates the internal built-in 15 kΩ pull-down resistors will be used instead of external ones (see Figure 33).

This feature is a cost-saving option. However, the power-on reset default value is logic 0. If you want to use the internal resistors, the HCD must check this bit status after every reset, because a reset action will clear this bit regardless of it being a hardware reset or a software reset.



Using either internal or external 15 kΩ resistors.

Fig 33. Use of 15 kΩ pull-down resistors on downstream ports.

9.8 Overcurrent detection and power switching control

A downstream port provides 5 V power supply to the V_{BUS}. The ISP1161 has built-in hardware functions to monitor the downstream ports loading conditions and control their power switching. These hardware functions are implemented by the internal power switching control circuit and overcurrent detection circuit. Pins $\overline{H\_PSW1}$ and $\overline{H\_PSW2}$ are power switching control output pins (active LOW and open drain) for downstream port 1 and 2, respectively. Pins $\overline{H\_OC1}$ and $\overline{H\_OC2}$ are overcurrent detection input pins for downstream ports 1 and 2, respectively. Let $\overline{H\_PSWn}$ denote either $\overline{H\_PSW1}$ or $\overline{H\_PSW2}$ and $\overline{H\_OCn}$ denote either $\overline{H\_OC1}$ or $\overline{H\_OC2}$.

Figure 34 shows the ISP1161 downstream port power management scheme.

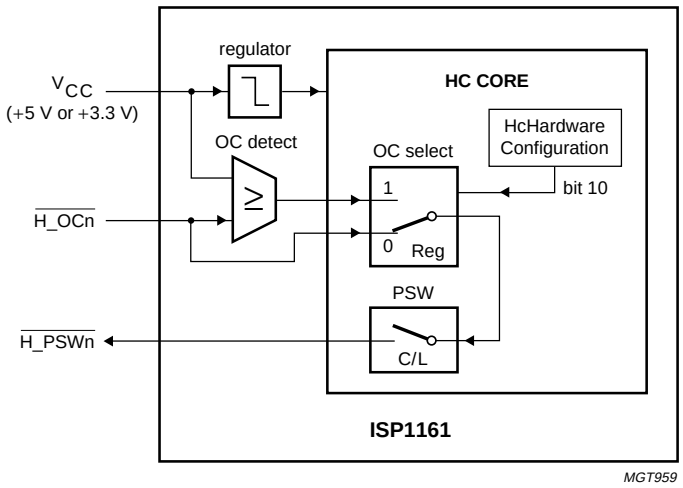


Fig 34. Downstream port power management scheme.

9.8.1 Using internal overcurrent detection circuit

The internal OC detection circuit can be used only when pin V_{CC} is connected to a 5 V power supply. The HCD must set AnalogOCEnable (bit 10 of the HcHardwareConfiguration register) to logic 1.

An application using the internal OC detection circuit and internal 15 kΩ pull-down resistors is shown in Figure 35, where H_DMn denotes either pin H_DM1 or H_DM2, while H_DPn denotes either pin H_DP1 or H_DP2. In this example, the HCD must set both AnalogOCEnable and DownstreamPort15KresistorSel to logic 1. They are bit 10 and bit 12 of the HcHardwareConfiguration register, respectively.

When $\overline{\text{H\_OCn}}$ detects an overcurrent status on a downstream port, $\overline{\text{H\_PSWn}}$ will output HIGH, logic 1 to turn off the 5 V power supply to the downstream port V_{BUS}. When there is no such detection, H_PSWn will output LOW, logic 0 to turn on the 5 V power supply to the downstream port V_{BUS}.

In general applications, we can use a P-channel MOSFET as the power switch for V_{BUS}. Connect the 5 V power supply to the drain pole of the P-channel MOSFET, V_{BUS} to the source pole, and $\overline{\text{H\_PSWn}}$ to the gate pole. We call the voltage drop (ΔV) across the drain and source poles the overcurrent trip voltage. For the internal overcurrent detection circuit, a voltage comparator has been designed-in, with a nominal voltage threshold of 75 mV. Therefore, when the overcurrent trip voltage (ΔV) exceeds the voltage threshold, $\overline{\text{H\_PSWn}}$ will output a HIGH level, logic 1 to turn off the P-channel MOSFET. If the P-channel MOSFET has a R_{DSon} of 150 mΩ, the overcurrent threshold will be 500 mA. The selection of a P-channel MOSFET with a different R_{DSon} results in a different overcurrent threshold.

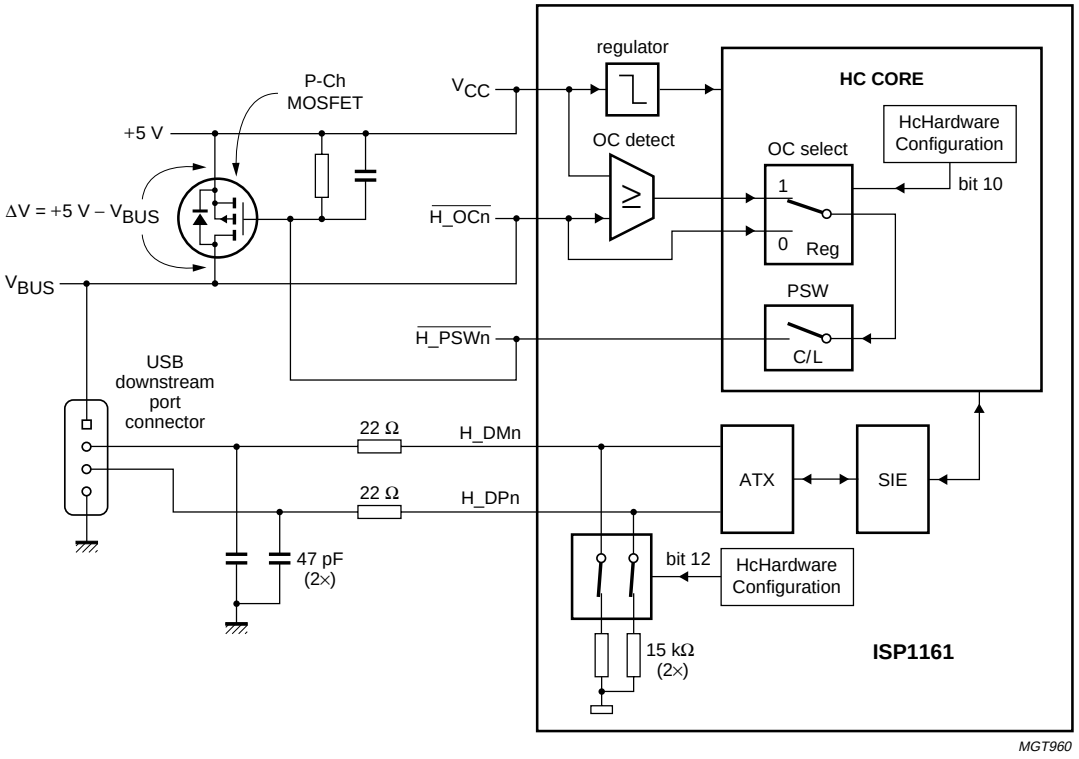


Fig 35. Using internal OC detection circuit.

9.8.2 Using external overcurrent detection circuit

When pin V_{CC} is connected to the 3.3 V power supply instead of the 5 V power supply, then the internal OC detection circuit cannot be used. An external OC detection circuit must be used instead. Nevertheless, regardless of V_{CC}'s connections, an external OC detection circuit can be used from time to time. To use an external OC detection circuit, AnalogOCEnable (bit 10 of the HcHardwareConfiguration register) should be set to logic 0. By default after reset, this bit is already set to logic 0; therefore, the HCD does not need to clear this bit.

Figure 36 shows how to use an external OC detection circuit.

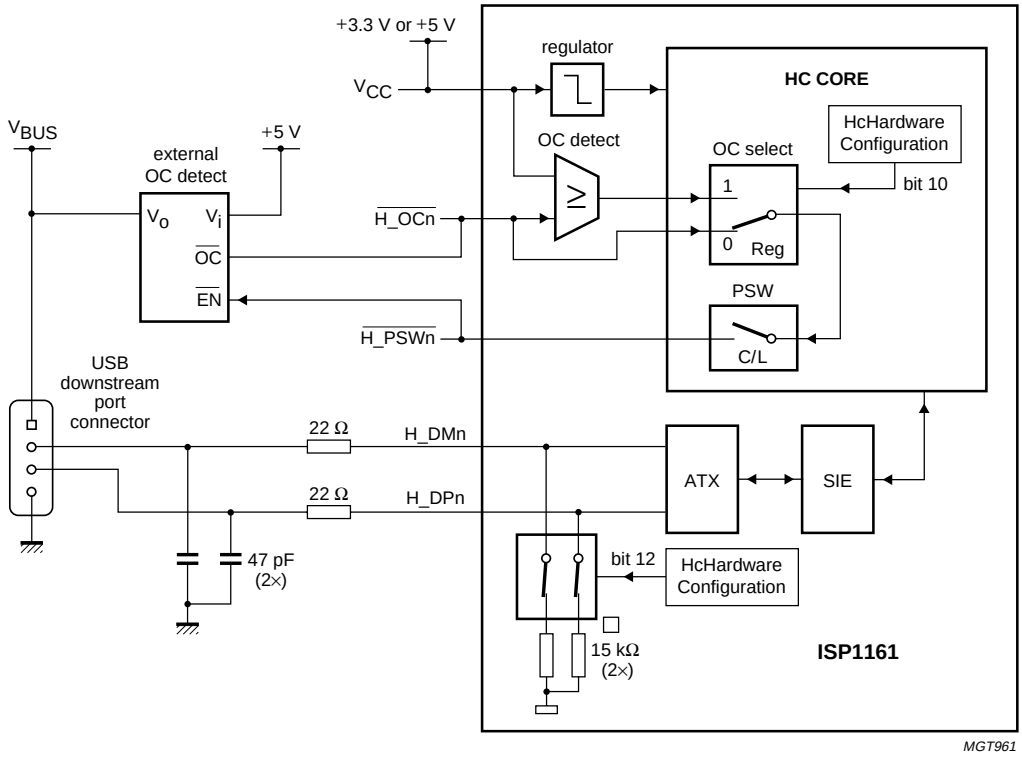


Fig 36. Using external OC detection circuit.

9.9 Suspend and wake-up

9.9.1 HC suspended state

The HC can be put into suspended state by setting the HcControl register (01H - read, 81H - write). See Figure 23 for the HC's flow of USB states changes.

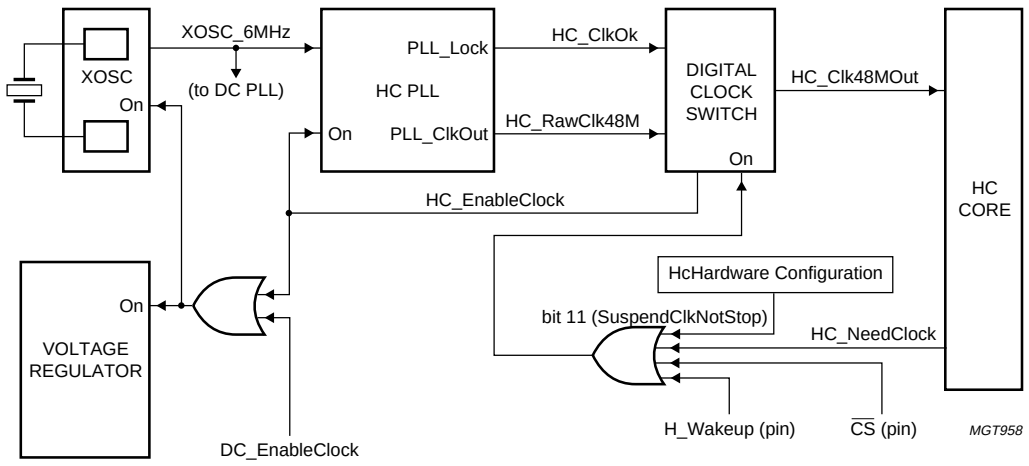


Fig 37. ISP1161 suspend and resume clock scheme.

In suspended state, the device will consume considerably less power by turning off the internal 48 MHz clock, PLL and crystal, and setting the internal regulator to power-down mode. The ISP1161 suspend and resume clock scheme is shown in Figure 37.

Remark: ISP1161 can be put into a fully suspended mode only after both the HC and the DC go into the suspended mode, when the crystal can be turned off and the internal regulator can be put into power-down mode.

Pin H_SUSPEND is the sensing output pin for HC's suspended state. When the HC goes into 'suspend' state, this pin will output a HIGH level (logic 1). This pin is cleared to LOW (logic 0) level only when the HC is put into a reset state or operational state (refer to the HcControl register bits 7 to 6; 01H - read, 81H - write). By setting bit 11 (SuspendClkNotStop) of the HcHardwareConfiguration register (20H - read, A0H - write), you can also define such that when the HC goes into 'suspend' state, its internal clock is stopped or kept running. After HC enters the 'suspend' state for 1.3 ms, the internal clock will be stopped if bit SuspendClkNotStop is logic 0.

9.9.2 HC wake-up from suspended state

There are three methods to wake up the HC from the USB 'suspend' state: hardware wake-up, software wake-up, and USB bus resume. They are described as follows:

- Wake-up by pin H_WAKEUP

  Pins H_SUSPEND and H_WAKEUP provide hardware wake-up, a way of remote wake-up control for the HC without the need to access the HC internal registers. H_WAKEUP is an external wake-up control input pin for the HC. After the HC goes into 'suspend' state, it can be woken up by sending a HIGH level pulse to pin H_WAKEUP. This will turn on the HC's internal clock, and set bit 6 (ClkReady) of the HcμPInterrupt register (24H - read, A4H - write).

  Under the 'suspend' state, once pin H_WAKEUP goes HIGH, after 160 μs, the internal clock will be up. If pin H_WAKEUP continues to be HIGH, then the internal clock will be kept running, and the microprocessor can set the HC into OPERATIONAL state during this time.

  If H_WAKEUP goes LOW for more than 1.14 ms, the internal clock stops and the HC goes back into 'suspend' state.

- Wake-up by pin $\overline{CS}$ (software wake-up)

  During the 'suspend' state, an external microprocessor issues the chip select signal through pin $\overline{CS}$ to ISP1161. This method of access to ISP1161 internal registers is a software wake-up.

- Wake-up by USB devices

  For a USB bus resume, a USB device attached to the root hub port issues a resume signal to the HC through the USB bus, switching the HC from 'suspend' state to RESUME state. This will also set the ResumeDetected bit of the HcInterruptStatus register (03H - read, 83H - write).

No matter which method is used to wake up the HC from USBSuspend state, the corresponding interrupt bits must be enabled before the HC goes into USBSuspend state so that the microprocessor can receive the correct interrupt request to wake up the HC.

10. HC registers

The HC contains a set of on-chip control registers. These registers can be read or written by the Host Controller Driver (HCD). The Control and Status register sets, Frame Counter register sets, and Root Hub register sets are grouped under the category of HC Operational registers (32 bits). These operational registers are made compatible to OpenHCI (Host Controller Interface) Operational registers. This makes a provision that the OpenHCI HCD can be ported to ISP1161 easily.

Reserved bits may be defined in future releases of this specification. To ensure inter operability, the HCD that does not use a reserved field must not assume that the reserved field contains logic 0. Furthermore, the HCD must always preserve the values of the reserved field. When a R/W register is modified, the HCD must first read the register, modify the bits desired, and then write the register with the reserved bits still containing the read value. Alternatively, the HCD can maintain an in-memory copy of previously written values that can be modified and then written to the HC register. When a write to set or clear the register is written, bits written to reserved fields must be logic 0.

As shown in Table 7, the offset locations (the commands for reading registers) of these Operational registers (the 32-bit registers) are similar to those defined in the OHCI specification, however, the addresses are equal to offset divided by 4.

Table 7: HC registers summary

Address (Hex)		Register	Width	Reference	Functionality
Read	Write				
00	N/A	HcRevision	32	Section 10.1.1 on page 46	HC Control and Status registers
01	81	HcControl	32	Section 10.1.2 on page 46	
02	82	HcCommandStatus	32	Section 10.1.3 on page 48	
03	83	HcInterruptStatus	32	Section 10.1.4 on page 49	
04	84	HcInterruptEnable	32	Section 10.1.5 on page 50	
05	85	HcInterruptDisable	32	Section 10.1.6 on page 51	
0D	8D	HcFmInterval	32	Section 10.2.1 on page 53	HC Frame Counter registers
0E	N/A	HcFmRemaining	32	Section 10.2.2 on page 54	
0F	N/A	HcFmNumber	32	Section 10.2.3 on page 55	
11	91	HcLSThreshold	32	Section 10.2.4 on page 56	
12	92	HcRhDescriptorA	32	Section 10.3.1 on page 57	HC Root Hub registers
13	93	HcRhDescriptorB	32	Section 10.3.2 on page 59	
14	94	HcRhStatus	32	Section 10.3.3 on page 60	
15	95	HcRhPortStatus[1]	32	Section 10.3.4 on page 61	
16	96	HcRhPortStatus[2]	32	Section 10.3.4 on page 61	
20	A0	HcHardwareConfiguration	16	Section 10.4.1 on page 66	HC DMA and Interrupt Control registers
21	A1	HcDMAConfiguration	16	Section 10.4.2 on page 67	
22	A2	HcTransferCounter	16	Section 10.4.3 on page 68	
24	A4	HcμPInterrupt	16	Section 10.4.4 on page 69	
25	A5	HcμPInterruptEnable	16	Section 10.4.5 on page 70	
27	N/A	HcChipID	16	Section 10.5.1 on page 71	HC Miscellaneous registers
28	A8	HcScratch	16	Section 10.5.2 on page 72	
N/A	A9	HcSoftwareReset	16	Section 10.5.3 on page 72	
2A	AA	HcITLBufferLength	16	Section 10.6.1 on page 73	HC Buffer RAM Control registers
2B	AB	HcATLBufferLength	16	Section 10.6.2 on page 73	
2C	N/A	HcBufferStatus	16	Section 10.6.3 on page 74	
2D	N/A	HcReadBackITL0Length	16	Section 10.6.4 on page 75	
2E	N/A	HcReadBackITL1Length	16	Section 10.6.5 on page 75	
40	C0	HcITLBufferPort	16	Section 10.6.6 on page 76	
41	C1	HcATLBufferPort	16	Section 10.6.7 on page 77	

10.1 HC control and status registers

10.1.1 HcRevision register (R: 00H)

Code (Hex): 00 — read only

Table 8: HcRevision register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R	R	R	R	R	R	R	R
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R	R	R	R	R	R	R	R
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	00H							
Access	R	R	R	R	R	R	R	R
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	REV[7:0]							
Reset	1	0	1	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 9: HcRevision register: bit description

Bit	Symbol	Description
31 to 8	–	Reserved
7 to 0	REV[7:0]	**Revision:** This read-only field contains the BCD representation of the version of the HCI specification that is implemented by this HC. For example, a value of 11H corresponds to version 1.1. All HC implementations that are compliant with this specification will have a value of 10H.

10.1.2 HcControl register (R/W: 01H/81H)

The HcControl register defines the operating modes for the HC. RemoteWakeupEnable (RWE) is modified only by the HCD.

Code (Hex): 01 — read

Code (Hex): 81 — write

Table 10: HcControl register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Symbol	reserved							
Reset	00H							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved					RWE	RWC	reserved
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	HCFS[1:0]		reserved					
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11: HcControl register: bit description

Bit	Symbol	Description
31 to 11	-	reserved
10	RWE	**RemoteWakeupEnable:** This bit is used by the HCD to enable or disable the remote wake-up feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus is set, a remote wake-up is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt.
9	RWC	**RemoteWakeupConnected:** This bit indicates whether the HC supports remote wake-up signaling. If remote wake-up is supported and used by the system, it is the responsibility of system firmware to set this bit during POST. The HC clears the bit upon a hardware reset but does not alter it upon a software reset. Remote wake-up signaling of the host system is host-bus-specific and is not described in this specification.
8	-	reserved
7 to 6	HCFS[1:0]	**HostControllerFunctionalState** for USB: **00 —** USBRESET **01 —** USBRESUME **10 —** USBOPERATIONAL **11 —** USBSUSPEND A transition to USBOPERATIONAL from another state causes start-of-frame (SOF) generation to begin 1 ms later. The HCD may determine whether the HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by the HC only when in the USBSUSPEND state. The HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. The HC enters USBRESET after a software reset and a hardware reset. The latter also resets the Root Hub and asserts subsequent reset signaling to downstream ports.
5 to 0	-	reserved

10.1.3 HcCommandStatus register (R/W: 02H/82H)

The HcCommandStatus register is used by the HC to receive commands issued by the HCD, and it also reflects the HC's current status. To the HCD, it appears to be a 'write to set' register. The HC must ensure that bits written as logic 1 become set in the register while bits written as logic 0 remain unchanged in the register. The HCD may issue multiple distinct commands to the HC without concern for corrupting previously issued commands. The HCD has normal read access to all bits.

The SchedulingOverrunCount field indicates the number of frames with which the HC has detected the scheduling overrun error. This occurs when the Periodic list does not complete before EOF. When a scheduling overrun error is detected, the HC increments the counter and sets the SchedulingOverrun field in the HcInterruptStatus register.

Code (Hex): 02 — read

Code (Hex): 82 — write

Table 12: HcCommandStatus register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved						SOC[1:0]	
Reset	0	0	0	0	0	0	0	0
Access	R						R	
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved							HCR
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 13: HcCommandStatus register: bit description

Bit	Symbol	Description
31 to 18	-	reserved
17 to 16	SOC[1:0]	**SchedulingOverrunCount:** The field is incremented on each scheduling overrun error. It is initialized to 00B and wraps around at 11B. It will be incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus has already been set. This is used by HCD to monitor any persistent scheduling problems.
15 to 1	-	reserved
0	HCR	**HostControllerReset:** This bit is set by HCD to initiate a software reset of HC. Regardless of the functional state of HC, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise; e.g., the InterruptRouting field of HcControl, and no Host bus accesses are allowed. This bit is cleared by HC upon the completion of the reset operation. The reset operation must be completed within 10 s. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.

10.1.4 HcInterruptStatus register (R/W: 03H/83H)

This register provides the status of the events that cause hardware interrupts. When an event occurs, the HC sets the corresponding bit in this register. When a bit becomes set, a hardware interrupt is generated if the interrupt is enabled in the HcInterruptEnable register (see Section 10.1.5) and the MasterInterruptEnable bit is set. The HCD may clear specific bits in this register by writing logic 1 to the bit positions to be cleared. The HCD may not set any of these bits. The HC will not clear the bit.

Code (Hex): 03 — read

Code (Hex): 83 — write

Table 14: HcInteruptStatus register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	00H							
Access	R/W							

Bit	7	6	5	4	3	2	1	0
Symbol	reserved	RHSC	FNO	UE	RD	SF	reserved	SO
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 15: HcInterruptStatus register: bit description

Bit	Symbol	Description
31 to 7	-	reserved
6	RHSC	**RootHubStatusChange:** This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed.
5	FNO	**FrameNumberOverflow:** This bit is set when the MSB of HcFmNumber (bit 15) changes value, from logic 0 to 1 or from logic 1 to 0.
4	UE	**UnrecoverableError:** This bit is set when the HC detects a system error not related to USB. The HC should not proceed with any processing nor signaling before the system error has been corrected. The HCD clears this bit after HC has been reset. PHCI: Always set to logic 0.
3	RD	**ResumeDetected:** This bit is set when the HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling causing this bit to be set. This bit is not set when HCD sets the USBRESUME state.
2	SF	**StartOfFrame:** At the start of each frame, this bit is set by the HC and an SOF generated.
1	-	reserved
0	SO	**SchedulingOverrun:** This bit is set when USB schedules for current frame overruns. A scheduling overrun will also cause the SchedulingOverrunCount of HcCommandStatus to be incremented.

10.1.5 HcInterruptEnable register (R/W: 04H/84H)

Each enable bit in the HcInterruptEnable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptEnable register is used to control which events generate a hardware interrupt. When these three conditions occur:

- A bit is set in the HcInterruptStatus register
- The corresponding bit in the HcInterruptEnable register is set
- The MasterInterruptEnable bit is set
- Then a hardware interrupt is requested on the host bus.

Writing logic 1 to a bit in this register sets the corresponding bit, whereas writing logic 0 to a bit in this register leaves the corresponding bit unchanged. On a read, the current value of this register is returned.

Code (Hex): 04 — read

Code (Hex): 84 — write

Table 16: HcInterruptEnable register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	MIE	reserved						
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved	RHSC	FNO	UE	RD	SF	reserved	SO
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 17: HcInterruptEnable register: bit description

Bit	Symbol	Description
31	MIE	**MasterInterruptEnable** by the HCD: Logic 0 is ignored by the HC. Logic 1 enables interrupt generation by events specified in other bits of this register.
30 to 7	-	reserved
6	RHSC	**0** — ignore **1** — enable interrupt generation due to Root Hub Status Change
5	FNO	**0** — ignore **1** — enable interrupt generation due to Frame Number Overflow
4	UE	**0** — ignore **1** — enable interrupt generation due to Unrecoverable Error
3	RD	**0** — ignore **1** — enable interrupt generation due to Resume Detect
2	SF	**0** — ignore **1** — enable interrupt generation due to Start of frame
1	-	reserved
0	SO	**0** — ignore **1** — enable interrupt generation due to Scheduling Overrun

10.1.6 HcInterruptDisable register (R/W: 05H/85H)

Each disable bit in the HcInterruptDisable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptDisable register is coupled with the HcInterruptEnable register. Thus, writing logic 1 to a bit in this register clears the corresponding bit in the HcInterruptEnable register, whereas

writing logic 0 to a bit in this register leaves the corresponding bit in the HcInterruptEnable register unchanged. On a read, the current value of the HcInterruptEnable register is returned.

Code (Hex): 05 — read

Code (Hex): 85 — write

Table 18: HcInterruptDisable register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	MIE	reserved						
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved	RHSC	FNO	UE	RD	SF	reserved	SO
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 19: HcInterruptDisable register: bit description

Bit	Symbol	Description
31	MIE	Logic 0 is ignored by the HC. Logic 1 disables interrupt generation due to events specified in other bits of this register. This field is set after a hardware or software reset.
30 to 7	-	reserved
6	RHSC	**0 —** ignore **1 —** disable interrupt generation due to Root Hub Status Change
5	FNO	**0 —** ignore **1 —** disable interrupt generation due to Frame Number Overflow
4	UE	**0 —** ignore **1 —** disable interrupt generation due to Unrecoverable Error
3	RD	**0 —** ignore **1 —** disable interrupt generation due to Resume Detect
2	SF	**0 —** ignore **1 —** disable interrupt generation due to Start of frame
1	-	reserved
0	SO	**0 —** ignore **1 —** disable interrupt generation due to Scheduling Overrun

10.2 HC frame counter registers

10.2.1 HcFmInterval register (R/W: 0DH/8DH)

The HcFmInterval register contains a 14-bit value which indicates the bit time interval in a frame (that is, between two consecutive SOFs), and a 15-bit value indicating the full-speed maximum packet size that the HC may transmit or receive without causing a scheduling overrun. The HCD may carry out minor adjustments on the FrameInterval by writing a new value over the present one at each SOF. This provides the programmability necessary for the HC to synchronize with an external clocking resource and to adjust any unknown local clock offset.

Code (Hex): 0D — read

Code (Hex): 8D — write

Table 20: HcFmInterval register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	FIT	FSMPS[14:8]						
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W						
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	FSMPS[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved		FI[13:8]					
Reset	0	0	1	0	1	1	1	0
Access	R/W		R/W					
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	FI[7:0]							
Reset	1	1	0	1	1	1	1	1
Access	R/W							

Table 21: HcFmInterval register: bit description

Bit	Symbol	Description
31	FIT	**FrameIntervalToggle:** The HCD toggles this bit whenever it loads a new value to FrameInterval.

Table 21: HcFmInterval register: bit description…*continued*

Bit	Symbol	Description
30 to 16	FSMPS [14:0]	**FSLargestDataPacket (FSMaximumPacketSize):** Specifies a value which is loaded into the Largest Data Packet Counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing a scheduling overrun. The field value is calculated by the HCD.
15 to 14	-	reserved
13 to 0	FI[13:0]	**FrameInterval:** Specifies the interval between two consecutive SOFs in bit times. The nominal value is set to be 11999. The HCD must store the current value of this field before resetting the HC. By setting the HostControllerReset bit 0 field of HcCommandStatus register will cause the HC to reset this field to its nominal value. HCD may choose to restore the stored value upon completing the Reset sequence.

10.2.2 HcFmRemaining register (R: 0EH)

The HcFmRemaining register is a 14-bit down counter showing the bit time remaining in the current frame.

Code (Hex): 0E — read

Table 22: HcFmRemaining register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	FRT	reserved						
Reset	0	0	0	0	0	0	0	0
Access	R				R			
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved		FR[13:8]					
Reset	0	0	0	0	0	0	0	0
Access	R				R			
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	FR[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R							

Table 23: HcFmRemaining register: bit description

Bit	Symbol	Description
31	FRT	**FrameRemainingToggle:** This bit is loaded from the FrameIntervalToggle field of HcFmInterval whenever FrameRemaining reaches 0. This bit is used by the HCD for synchronization between FrameInterval and FrameRemaining.
30 to 14	-	reserved
13 to 0	FR[13:0]	**FrameRemaining:** This counter is decremented at each bit time. When it reaches zero, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit time boundary. When entering the USBOPERATIONAL state, the HC reloads it with the content of the FrameInterval part of the HcFmInterval register and uses the updated value from the next SOF.

10.2.3 HcFmNumber register (R: 0FH)

The HcFmNumber register is a 16-bit counter. It provides a timing reference for events happening in the HC and the HCD. The HCD may use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

Code (Hex): 0F — read

Table 24: HCFmNumber register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	FN[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	FN[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R							

Table 25: HcFmNumber register: bit description

Bit	Symbol	Description
31 to 16	–	reserved
15 to 0	FN[15:0]	**FrameNumber:** This is incremented when HcFmRemaining is reloaded. It will be rolled over to 0H after FFFFH. When the USBOPERATIONAL state is entered, this will be incremented automatically. HC will set the StartofFrame in HcInterruptStatus.

10.2.4 HcLSThreshold register (R/W: 11H/91H)

The HcLSThreshold register contains an 11-bit value used by the HC to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF. Neither the HC nor the HCD is allowed to change this value.

Code (Hex): 11 — read

Code (Hex): 91 — write

Table 26: HcLSThreshold register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved					LST[10:8]		
Reset	0	0	0	0	0	1	1	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	LST[7:0]							
Reset	0	0	1	0	1	0	0	0
Access	R/W							

Table 27: HcLSThreshold register: bit description

Bit	Symbol	Description
31 to 11	–	reserved
10 to 0	LST[10:0]	**LSThreshold:** Contains a value that is compared to the FrameRemaining field before a low-speed transaction is initiated. The transaction is started only if FrameRemaining is equal or greater than this field. The value is calculated by the HCD, which considers transmission and set-up overhead.

10.3 HC Root Hub registers

All registers included in this partition are dedicated to the USB Root Hub, which is an integral part of the HC although it is a functionally separate entity. The HCD emulates USBD accesses to the Root Hub via a register interface. The HCD maintains many USB-defined hub features that are not required to be supported in hardware. For example, the Hub's Device, Configuration, Interface, and Endpoint Descriptors are maintained only in the HCD as well as some static fields of the Class Descriptor. The HCD also maintains and decodes the Root Hub's device address as well as other trivial operations that they are better suited to software than to hardware.

The Root Hub registers are developed to maintain the similarity of bit organization and operation to typical hubs found in the system.

Four registers are defined as follows:

- HcRhDescriptorA
- HcRhDescriptorB
- HcRhStatus
- HcRhPortStatus[1:NDP].

Each register is read and written as a Dword. These registers are only written during initialization to correspond with the system implementation. The HcRhDescriptorA and HcRhDescriptorB registers should be implemented such that they are writeable regardless of the HC's USB states. HcRhStatus and HcRhPortStatus must be writeable during the USBOPERATIONAL state.

10.3.1 HcRhDescriptorA register (R/W: 12H/92H)

The HcRhDescriptorA register is the first register of two describing the characteristics of the Root Hub. Reset values are implementation specific. The descriptor length (11), descriptor type and hub controller current (0) fields of the hub Class Descriptor are emulated by the HCD. All other fields are located in the registers HcRhDescriptorA and HcRhDescriptorB.

Remark: IS denotes an implementation specific reset value for that field.

Code (Hex): 12 — read

Code (Hex): 92 — write

Table 28: HcRhDescriptorA register: bit description

Bit	31	30	29	28	27	26	25	24
Symbol	POTPGT[7:0]							
Reset	IS							
Access	R/W							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved			NOCP	OCPM	DT	NPS	PSM
Reset	0	0	0	IS	IS	0	IS	IS
Access	R			R/W	R/W	R	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved						NDP[1:0]	
Reset	0	0	0	0	0	0	IS	
Access	R						R	

Table 29: HcRhDescriptorA register: bit description

Bit	Symbol	Description
31 to 24	POTPGT [7:0]	**PowerOnToPowerGoodTime:** This byte specifies the duration HCD has to wait before accessing a powered-on port of the Root Hub. It is implementation specific. The unit of time is 2 ms. The duration is calculated as POTPGT × 2 ms.
23 to 13	-	reserved
12	NOCP	**NoOverCurrentProtection:** This bit describes how the overcurrent status for the Root Hub ports are reported. When this bit is cleared, the OverCurrentProtectionMode field specifies global or per-port reporting. **0 —** overcurrent status is reported collectively for all downstream ports **1 —** no overcurrent reporting supported
11	OCPM	**OverCurrentProtectionMode:** This bit describes how the overcurrent status for the Root Hub ports is reported. At reset, this field should reflect the same mode as PowerSwitchingMode. This field is valid only if the NoOverCurrentProtection field is cleared. **0 —** overcurrent status is reported collectively for all downstream ports **1 —** overcurrent status is reported on a per-port basis. On power-up, clear this bit and then set it to logic 1.
10	DT	**DeviceType:** This bit specifies that the Root Hub is not a compound device—it is not permitted. This field should always read/write 0.
9	NPS	**NoPowerSwitching:** This bit is used to specify whether power switching is supported or ports are always powered. It is implementation-specific. When this bit is cleared, the bit PowerSwitchingMode specifies global or per-port switching. **0 —** ports are power switched **1 —** ports are always powered on when the HC is powered on
8	PSM	**PowerSwitchingMode:** This bit is used to specify how the power switching of the Root Hub ports is controlled. It is implementation-specific. This field is valid only if the NoPowerSwitching field is cleared. **0 —** all ports are powered at the same time **1 —** each port is powered individually. This mode allows port power to be controlled by either the global switch or per-port switching. If the bit PortPowerControlMask is set, the port responds to only port power commands (Set/ClearPortPower). If the port mask is cleared, then the port is controlled only by the global power switch (Set/ClearGlobalPower).
7 to 2	-	reserved
1 to 0	NDP[1:0]	**NumberDownstreamPorts:** These bits specify the number of downstream ports supported by the Root Hub. It is implementation-specific. The maximum number of ports supported is 2.

10.3.2 HcRhDescriptorB register (R/W: 13H/93H)

The HcRhDescriptorB register is the second register of two describing the characteristics of the Root Hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation specific.

Code (Hex): 13 — read

Code (Hex): 93 — write

Table 30: HcRhDescriptorB register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	N/A							
Access	R							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved					PPCM[2:0]		
Reset	N/A					IS		
Access	R					R/W		
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	N/A							
Access	R							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved					DR[2:0]		
Reset	N/A					IS		
Access	R					R/W		

Table 31: HcRhDescriptorB register: bit description

Bit	Symbol	Description
31 to 19	-	reserved
18 to 16	PPCM[2:0]	**PortPowerControlMask:** Each bit indicates whether a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode = 0), this field is not valid. **bit 0 —** reserved **bit 1 —** Ganged-power mask on Port 1 **bit 2 —** Ganged-power mask on Port 2
15 to 3	-	reserved
2 to 0	DR[2:0]	**DeviceRemovable:** Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable. **bit 0 —** reserved **bit 1 —** Device attached to Port 1 **bit 2 —** Device attached to Port 2

10.3.3 HcRhStatus register (R/W: 14H/94H)

The HcRhStatus register is divided into two parts. The lower word of a Dword represents the Hub Status field and the upper word represents the Hub Status Change field. Reserved bits should always be written as logic 0.

Code (Hex): 14 — read

Code (Hex): 94 — write

Table 32: HcRhStatus register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	CRWE	reserved						
Reset	0	0						
Access	W	R						
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved						OCIC	LPSC
Reset	0						0	0
Access	R						R/W	R/W
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	DRWE	reserved						
Reset	0	0	0	0	0	0	0	0
Access	R/W	R						

Bit	7	6	5	4	3	2	1	0
Symbol	reserved						OCI	LPS
Reset	0	0	0	0	0	0	0	0
Access	R						R	R/W

Table 33: HcRhStatus register: bit description

Bit	Symbol	Description
31	CRWE	On write—**ClearRemoteWakeupEnable:** Writing logic 1 clears DeviceRemoveWakeupEnable. Writing logic 0 has no effect.
30 to 18	-	reserved
17	OCIC	**OverCurrentIndicatorChange:** This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing logic 1. Writing logic 0 has no effect.
16	LPSC	On read—**LocalPowerStatusChange:** The Root Hub does not support the local power status feature. Therefore, this bit is always read as logic 0. On write—**SetGlobalPower:** In global power mode (PowerSwitchingMode = 0), this bit is written to logic 1 to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose bit PortPowerControlMask is not set. Writing logic 0 has no effect.
15	DRWE	On read—**DeviceRemoteWakeupEnable:** This bit enables bit ConnectStatusChange as a resume event, causing a state transition USBSUSPEND to USBRESUME and setting the ResumeDetected interrupt. **0 —** ConnectStatusChange is not a remote wake-up event **1 —** ConnectStatusChange is a remote wake-up event On write—**SetRemoteWakeupEnable:** Writing logic 1 sets DeviceRemoveWakeupEnable. Writing logic 0 has no effect.
14 to 2	-	reserved
1	OCI	**OverCurrentIndicator:** This bit reports overcurrent conditions when global reporting is implemented. When set, an overcurrent condition exists. When clear, all power operations are normal. If per-port overcurrent protection is implemented this bit is always logic 0.
0	LPS	On read—**LocalPowerStatus:** The Root Hub does not support the local power status feature. Therefore, this bit is always read as logic 0. On write—**ClearGlobalPower:** In global power mode (PowerSwitchingMode = 0), this bit is written to logic 1 to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing logic 0 has no effect.

10.3.4 HcRhPortStatus[1:2] register (R/W [1]:15H/95H, [2]: 16H/96H)

The HcRhPortStatus[1:2] register is used to control and report port events on a per-port basis. NumberDownstreamPorts represents the number of HcRhPortStatus registers that are implemented in hardware. The lower word is used to reflect the port status, whereas the upper word reflects the status change bits. Some status bits are

implemented with special write behavior. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes. Reserved bits should always be written logic 0.

Code (Hex): [1] = 15, [2] = 16 — read

Code (Hex): [1] = 95, [2] = 96 — write

Table 34: HcRhPortStatus[1:2] register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved			PRSC	OCIC	PSSC	PESC	CSC
Reset	0	0	0	0	0	0	0	0
Access	R/W			R/W	R/W	R/W	R/W	R/W
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved						LSDA	PPS
Reset	0						0	0
Access	R/W						R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved			PRS	POCI	PSS	PES	CCS
Reset	0	0	0	0	0	0	0	0
Access	R/W			R/W	R/W	R/W	R/W	R/W

Table 35: HcRhPortStatus[1:2] register: bit description

Bit	Symbol	Description
31 to 21	-	reserved
20	PRSC	**PortResetStatusChange:** This bit is set at the end of the 10 ms port reset signal. The HCD writes logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** port reset is not complete **1 —** port reset is complete
19	OCIC	**PortOverCurrentIndicatorChange:** This bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when Root Hub changes the PortOverCurrentIndicator bit. The HCD writes logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** no change in PortOverCurrentIndicator **1 —** PortOverCurrentIndicator has changed

Table 35: HcRhPortStatus[1:2] register: bit description...*continued*

Bit	Symbol	Description
18	PSSC	**PortSuspendStatusChange:** This bit is set when the full resume sequence has been completed. This sequence includes the 20 s resume pulse, LS EOP, and 3 ms re-synchronization delay. The HCD writes logic 1 to clear this bit. Writing logic 0 has no effect. This bit is also cleared when ResetStatusChange is set. **0 —** resume is not completed **1 —** resume is completed
17	PESC	**PortEnableStatusChange:** This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** no change in PortEnableStatus **1 —** change in PortEnableStatus
16	CSC	**ConnectStatusChange:** This bit is set whenever a connect or disconnect event occurs. The HCD writes logic 1 to clear this bit. Writing logic 0 has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected. **0 —** no change in CurrentConnectStatus **1 —** change in CurrentConnectStatus **Remark:** If the DeviceRemovable[NDP] bit is set, this bit is set only after a Root Hub reset to inform the system that the device is attached.
15 to 10	-	reserved
9	LSDA	(read) **LowSpeedDeviceAttached:** This bit indicates the speed of the device attached to this port. When set, a low-speed device is attached to this port. When clear, a full-speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. **0 —** full-speed device attached **1 —** low-speed device attached (write) **ClearPortPower:** The HCD clears the PortPowerStatus bit by writing logic 1 to this bit. Writing logic 0 has no effect.

Table 35: HcRhPortStatus[1:2] register: bit description...*continued*

Bit	Symbol	Description
8	PPS	(read) **PortPowerStatus:** This bit reflects the port power status, regardless of the type of power switching implemented. This bit is cleared if an overcurrent condition is detected. The HCD sets this bit by writing SetPortPower or SetGlobalPower. The HCD clears this bit by writing ClearPortPower or ClearGlobalPower. Which power control switches are enabled is determined by PowerSwitchingMode. In the global switching mode (PowerSwitchingMode = 0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PowerSwitchingMode = 1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. **0 —** port power is off **1 —** port power is on (write) **SetPortPower:** The HCD writes logic 1 to set the PortPowerStatus bit. Writing logic 0 has no effect. **Remark:** This bit always reads logic 1 if power switching is not supported.
7 to 5	-	reserved
4	PRS	(read) **PortResetStatus:** When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. **0 —** port reset signal is not active **1 —** port reset signal is active (write) **SetPortReset:** The HCD sets the port reset signaling by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus but instead sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port.
3	POCI	(read) **PortOverCurrentIndicator:** This bit is valid only when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to logic 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal. **0 —** no overcurrent condition **1 —** overcurrent condition detected (write) **ClearSuspendStatus:** The HCD writes logic 1 to initiate a resume. Writing logic 0 has no effect. A resume is initiated only if PortSuspendStatus is set.

Table 35: HcRhPortStatus[1:2] register: bit description…*continued*

Bit	Symbol	Description
2	PSS	(read) **PortSuspendStatus:** This bit indicates whether the port is suspended or in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. **0 —** port is not suspended **1 —** port is suspended (write) **SetPortSuspend:** The HCD sets the PortSuspendStatus bit by writing logic 1 to this bit. Writing logic 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus; instead it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port.
1	PES	(read) **PortEnableStatus:** This bit indicates whether the port is enabled or disabled. The Root Hub may clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error such as babble is detected. This change also causes PortEnabledStatusChange to be set. The HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if it is not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. **0 —** port is disabled **1 —** port is enabled (write) **SetPortEnable:** The HCD sets PortEnableStatus by writing logic 1. Writing logic 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port.
0	CCS	(read) **CurrentConnectStatus:** This bit reflects the current state of the downstream port. **0 —** no device connected **1 —** device connected (write) **ClearPortEnable:** The HCD writes logic 1 to this bit to clear the PortEnableStatus bit. Writing logic 0 has no effect. CurrentConnectStatus is not affected by any write. **Remark:** This bit always reads logic 1 when the attached device is non removable (DeviceRemoveable[NDP]).

10.4 HC DMA and interrupt control registers

10.4.1 HcHardwareConfiguration register (R/W: 20H/A0H)

- Bit 0 (InterruptPinEnable) is used as pin INT1's master interrupt enable. This bit should be used together with the register HcμPInterruptEnable to enable pin INT1.
- Bits 4 and 3 (DataBusWidth[1:0]) are fixed at logic 0 and logic 1, respectively, for ISP1161.

Code (Hex): 20 — read

Code (Hex): A0 — write

Table 36: HcHardwareConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved			2_DownstreamPort15KresistorSel	SuspendClkNotStop	AnalogOCEnable	reserved	DACKMode
Reset	0	0	0	0	0	0	0	0
Access	R/W			R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	EOTInputPolarity	DACKInputPolarity	DREQOutputPolarity	DataBusWidth[1:0]		InterruptOutputPolarity	InterruptPinTrigger	InterruptPinEnable
Reset	0	0	1	0	1	0	0	0
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W

Table 37: HcHardwareConfiguration register: bit description

Bit	Symbol	Description
15 to 13	-	reserved
12	2_DownstreamPort15KresistorSel	**0 —** use external 15 kΩ resistors for downstream ports
		1 — use built-in resistors for downstream ports
11	SuspendClkNotStop	**0 —** clock can be stopped
		1 — clock can not be stopped
10	AnalogOCEnable	**0 —** use external OC detection. Digital input
		1 — use on-chip OC detection. Analog input
9	-	reserved
8	DACKMode	**0 —** normal operation. DACK1 is used with read and write signals. Power-up value.
		1 — reserved
7	EOTInputPolarity	**0 —** active LOW. Power-up value
		1 — active HIGH
6	DACKInputPolarity	**0 —** active LOW. Power-up value
		1 — reserved

Table 37: HcHardwareConfiguration register: bit description…*continued*

Bit	Symbol	Description
5	DREQOutputPolarity	**0 —** active LOW
		1 — active HIGH. Power-up value
4 to 3	DataBusWidth[1:0]	**01 —** 16 bits
		Others — reserved
2	InterruptOutputPolarity	**0 —** active LOW. Power-up value
		1 — active HIGH
1	InterruptPinTrigger	**0 —** interrupt is level-triggered. Power-up value
		1 — interrupt is edge-triggered.
0	InterruptPinEnable	**0 —** power-up value
		1 — pin Global Interrupt INT1 is enabled

10.4.2 HcDMAConfiguration register (R/W: 21H/A1H)

Code (Hex): 21 — read

Code (Hex): A1 — write

Table 38: HcDMAConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved	BurstLen[1:0]		DMA Enable	reserved	DMACounterSelect	ITL_ATL_DataSelect	DMAReadWriteSelect
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W

Table 39: HcDMAConfiguration register: bit description

Bit	Symbol	Description
15 to 7	-	reserved

Table 39: HcDMAConfiguration register: bit description…*continued*

Bit	Symbol	Description
6 to 5	BurstLen[1:0]	**00 —** single-cycle burst DMA **01 —** 4-cycle burst DMA **10 —** 8-cycle burst DMA **11 —** reserved
4	DMAEnable	**0 —** DMA is terminated **1 —** DMA is enabled; this bit will be reset to zero when DMA transfer is completed
3	-	reserved
2	DMACounter Select	**0 —** DMA counter not used. External EOT must be used **1 —** Enables the DMA counter for DMA transfer. The HcTransferCounter register must be filled with non-zero values for DREQ1 to be raised after bit DMA Enable is set
1	ITL_ATL_ DataSelect	**0 —** ITL buffer RAM selected for ITL data **1 —** ATL buffer RAM selected for ATL data
0	DMARead WriteSelect	**0 —** read from ISP1161 HC's FIFO buffer RAM **1 —** write to ISP1161 HC's FIFO buffer RAM

10.4.3 HcTransferCounter register (R/W: 22H/A2H)

This register holds the number of bytes of a PIO or DMA transfer. For a PIO transfer, the number of bytes being read or written to the Isochronous Transfer List (ITL) or Acknowledged Transfer List (ATL) buffer RAM must be written into this register. For a DMA transfer, the number of bytes must be written into this register as well. However, for this counter to be read into the DMA counter, the HCD must set bit 2 (DMACounterSelect) of the HcDMAConfiguration register. The counter value for ATL must not be greater than 1000H, and for ITL it must not be greater than 800H. When the byte count of the data transfer reaches this value, the HC will generate an internal EOT signal to set bit 2 (AllEOTInterrupt) of the HcμPInterrupt register, and also update the HcBufferStatus register.

Code (Hex): 22 — read

Code (Hex): A2 — write

Table 40: HcTransferCounter register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	Counter value							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	Counter value							
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 41: HcTransferCounter register: bit description

Bit	Symbol	Description
15 to 0	Counter value	The number of data bytes to be read to or written from RAM.

10.4.4 HcμPInterrupt register (R/W: 24H/A4H)

All the bits in this register will be active on power-on reset. However, none of the active bits will cause an interrupt on the interrupt pin (INT1) unless they are set by the respective bits in the HcμPInterruptEnable register, and together with bit 0 (InterruptPinEnable) of the HcHardwareConfiguration register.

After this register (24H - read) is read, the bits that are active will not be reset, until logic 1 is written to the bits in this register (A4H - write) to clear it.

The bits in this register are cleared only when you write to this register indicating the bits to be cleared. To clear all the enabled bits in this register, the HCD must write FFH to this register.

Code (Hex): 24 — read

Code (Hex): A4 — write

Table 42: HcμPInterrupt register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved	ClkReady	HC Suspended	OPR_Reg	reserved	AllEOT Interrupt	ATLInt	SOFITLInt
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 43: HcμPInterrupt register: bit description

Bit	Symbol	Description
15 to 7	-	reserved

Table 43: HcμPInterrupt register: bit description *...continued*

Bit	Symbol	Description
6	ClkReady	**0 —** no event **1 —** clock is ready. After a wake-up is sent, there is a wait for clock ready. Maximum is 1 ms, and typical is 160 μs.
5	HC Suspended	**0 —** no event **1 —** the HC has been suspended and no USB activity is sent from the microprocessor for each ms. When the microprocessor wants to suspend the HC, the microprocessor must write to the HcControl register. And when all downstream devices are suspended, then the HC stops sending SOF; the HC is suspended by having the HcControl register written into.
4	OPR_Reg	**0 —** no event **1 —** There are interrupts from HC side. Need to read HcControl and HcInterrupt registers to detect type of interrupt on the HC (if the HC requires the Operational register to be updated).
3	-	reserved
2	AllEOT Interrupt	**0 —** no event **1 —** implies that data transfer has been completed via PIO transfer or DMA transfer. Occurrence of internal or external EOT will set this bit.
1	ATLInt	**0 —** no event **1 —** implies that the microprocessor must read ATL data from the HC. This requires that the HcBufferStatus register must first be read. The time for this interrupt depends on the number of clocks bit set for USB activities in each ms.
0	SOFITLInt	**0 —** no event **1 —** implies that SOF indicates the 1 ms mark. The ITL buffer that the HC has handled must be read. To know the ITL buffer status, the HcBufferStatus register must first be read. This is for the microprocessor to get ISO data to or from the HC. For more information, see the 6th paragraph of Section 9.5.

10.4.5 HcμPInterruptEnable register (R/W: 25H/A5H)

The bits 6 to 0 in this register are the same as those in the HcμPInterrupt register. They are used together with bit 0 (InterruptPinEnable) of the HcHardwareConfiguration register to enable or disable the bits in the HcμPInterrupt register.

On power-on, all bits in this register are masked with logic 0. This means no interrupt request output on the interrupt pin INT1 can be generated.

When the bit is set to logic 1, the interrupt for the bit is not masked but enabled.

Code (Hex): 25 — read

Code (Hex): A5 — write

Table 44: HcμPInterruptEnable register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved							
Reset	00H							
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved	ClkReady	HC Suspended Enable	OPR Interrupt Enable	reserved	EOT Interrupt Enable	ATL Interrupt Enable	SOF Interrupt Enable
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 45: HcμPInterruptEnable register: bit description

Bit	Symbol	Description
15 to 7	-	reserved
6	ClkReady	**0 —** power-up value **1 —** enables Clkready interrupt
5	HC Suspended Enable	**0 —** power-up value **1 —** enables HC suspended interrupt. When the microprocessor wants to suspend the HC, the microprocessor must write to the HcControl register. And when all downstream devices are suspended, then the HC stops sending SOF; the HC is suspended by having the HcControl register written into.
4	OPR Interrupt Enable	**0 —** power-up value **1 —** enables the 32-bit Operational register's interrupt (if the HC requires the Operational register to be updated)
3	-	reserved
2	EOT Interrupt Enable	**0 —** power-up value **1 —** enables the EOT interrupt which indicates an end of a read/write transfer
1	ATL Interrupt Enable	**0 —** power-up value **1 —** enables ATL interrupt. The time for this interrupt depends on the number of clock bits set for USB activities in each ms.
0	SOF Interrupt Enable	**0 —** power-up value **1 —** enables the interrupt bit due to SOF (for the microprocessor DMA to get ISO data from the HC by first accessing the HcDMAConfiguration register)

10.5 HC miscellaneous registers

10.5.1 HcChipID register (R: 27H)

Read this register to get the ID of the ISP1161 silicon chip. The high byte stands for the product name (here 61H stands for ISP1161). The low byte indicates the revision number of the product including engineering samples.

Code (Hex): 27 — read

Table 46: HcChipID register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	ChipID[15:8]							
Reset	0	1	1	0	0	0	0	1
Access	R							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	ChipID[7:0]							
Reset	0	0	1	0	0	0	0	0
Access	R							

Table 47: HcChipID register: bit description

Bit	Symbol	Description
15 to 0	ChipID[15:0]	ISP1161's chip ID

10.5.2 HcScratch register (R/W: 28H/A8H)

This register is for the HCD to save and restore values when required.

Code (Hex): 28 — read

Code (Hex): A8 — write

Table 48: HcScratch register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	Scratch[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	Scratch[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 49: HcScratch register: bit description

Bit	Symbol	Description
15 to 0	Scratch[15:0]	Scratch register value

10.5.3 HcSoftwareReset register (W: A9H)

This register provides a means for software reset of the HC. To reset the HC, the HCD must write a reset value of F6H to this register. Upon receiving the reset value, the HC resets all the registers except its buffer memory.

Code (Hex): A9 — write

Table 50: HcSoftwareReset register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	Reset[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	W							

Bit	7	6	5	4	3	2	1	0
Symbol	Reset[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	W							

Table 51: HcSoftwareReset register: bit description

Bit	Symbol	Description
15 to 0	Reset[15:0]	Writing a reset value of F6H will cause the HC to reset all the registers except its buffer memory.

10.6 HC buffer RAM control registers

10.6.1 HcITLBufferLength register (R/W: 2AH/AAH)

Write to this register to assign the ITL buffer size in bytes: ITL0 and ITL1 are assigned the same value. For example, if HcITLBufferLength register is set to 2 kbytes, then ITL0 and ITL1 would be allocated 2 kbytes each.

Must follow the formula:

ATL buffer length + 2 × (ITL buffer size) ≤ 1000H (that is 4 kbytes)

where: ITL buffer size = ITL0 buffer length = ITL1 buffer length.

Code (Hex): 2A — read

Code (Hex): AA — write

Table 52: HcITLBufferLength register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	ITLBufferLength[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	ITLBufferLength[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 53: HcITLBufferLength register: bit description

Bit	Symbol	Description
15 to 0	ITLBufferLength[15:0]	Assign ITL buffer length

10.6.2 HcATLBufferLength register (R/W: 2BH/ABH)

Write to this register to assign ATL buffer size.

Code (Hex): 2B — read

Code (Hex): AB — write

Remark: The maximum total RAM size is 1000H (4096 in decimal) bytes. That means: ITL0 (length) + ITL1 (length) + ATL (length) ≤ 1000H bytes. For example: If ATL buffer length has been set to be 800H, then the maximum ITL buffer length can only be set as 400H.

Table 54: HcATLBufferLength register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	ATLBufferLength[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	ATLBufferLength[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 55: HcATLBufferLength register: bit description

Bit	Symbol	Description
15 to 0	ATLBufferLength[15:0]	Assign ATL buffer length

10.6.3 HcBufferStatus register (R: 2CH)

Code (Hex): 2C — read

Table 56: HcBufferStatus register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved							
Reset	00H							
Access	R							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved		ATLBuffer Done	ITL1Buffer Done	ITL0Buffer Done	ATLBuffer Full	ITL1Buffer Full	ITL0Buffer Full
Reset	0	0	0	0	0	0	0	0
Access	R		R	R	R	R	R	R

Table 57: HcBufferStatus register: bit description

Bit	Symbol	Description
15 to 6	-	reserved

Table 57: HcBufferStatus register: bit description…*continued*

Bit	Symbol	Description
5	ATLBuffer Done	**0** — ATL Buffer not read by HC yet **1** — ATL Buffer read by HC
4	ITL1Buffer Done	**0** — ITL1 Buffer not read by HC yet **1** — ITL1 Buffer read by HC
3	ITL0Buffer Done	**0** — 1TL0 Buffer not read by HC yet **1** — 1TL0 Buffer read by HC
2	ATLBuffer Full	**0** — ATL Buffer is empty **1** — ATL Buffer is full
1	ITL1Buffer Full	**0** — 1TL1 Buffer is empty **1** — 1TL1 Buffer is full
0	ITL0Buffer Full	**0** — ITL0 Buffer is empty **1** — ITL0 Buffer is full

10.6.4 HcReadBackITL0Length register (R: 2DH)

This register's value stands for the current number of data bytes inside an ITL0 buffer to be read back by the microprocessor. The HCD must set the HcTransferCounter equivalent to this value before reading back the ITL0 buffer RAM.

Code (Hex): 2D — read

Table 58: HcReadBackITL0Length register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	RdITL0BufferLength[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	RdITL0BufferLength[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R							

Table 59: HcReadBackITL0Length register: bit description

Bit	Symbol	Description
15 to 0	RdITL0BufferLength[15:0]	The number of bytes for ITL0 data to be read back by the microprocessor

10.6.5 HcReadBackITL1Length register (R: 2EH)

This register's value stands for the current number of data bytes inside the ITL1 buffer to be read back by the microprocessor. The HCD must set the HcTransferCounter equivalent to this value before reading back the ITL1 buffer RAM.

Code (Hex): 2E — read

Table 60: HcReadBackITL1Length register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	RdITL1BufferLength[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	RdITL1BufferLength[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R							

Table 61: HcReadBackITL1Length register: bit description

Bit	Symbol	Description
15 to 0	RdITL1BufferLength[15:0]	The number of bytes for ITL1 data to be read back by the microprocessor

10.6.6 HcITLBufferPort register (R/W: 40H/C0H)

This is the ITL buffer RAM read/write port. The bits 15 to 8 contain the data byte that comes from the ITL buffer RAM's even address. The bits 7 to 0 contain the data byte that comes from the ITL buffer RAM's odd address.

Code (Hex): 40 — read

Code (Hex): C0 — write

Table 62: HcITLBufferPort register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	DataWord[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	DataWord[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 63: HcITLBufferPort register: bit description

Bit	Symbol	Description
15 to 0	DataWord[15:0]	read/write ITL buffer RAM's two data bytes.

The HCD must set the byte count into the HcTransferCounter register and check the HcBufferStatus register before reading from or writing to the buffer. The HCD must write the command (40H for read, C0H for write) once only, and then read or write all the data bytes in word. After every read/write, the pointer of ITL buffer RAM will be automatically increased by two to point to the next data word until it reaches the value of HcTransferCounter register; otherwise, an internal EOT signal is not generated to set the bit 2 (AllEOTInterrupt) of the HcμPInterrupt register and update the HcBufferStatus register.

The HCD must take care of the difference that the internal buffer RAM is organized in bytes. The HCD must write the byte count into the HcTransferCounter register, but the HCD reads or writes the buffer RAM by 16 bits (by 1 word).

10.6.7 HcATLBufferPort register (R/W: 41H/C1H)

This is the ATL buffer RAM read/write port. The bits 15 to 8 contain the data byte that comes from the Acknowledged Transfer List (ATL) buffer RAM's odd address. The bits 7 to 0 contain the data byte that comes from the ATL buffer RAM's even address.

Code (Hex): 41 — read

Code (Hex): C1 — write

Table 64: HcATLBufferPort register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	DataWord[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	DataWord[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W							

Table 65: HcATLBufferPort register: bit description

Bit	Symbol	Description
15 to 0	DataWord[15:0]	read/write ATL buffer RAM's two data bytes.

The HCD must set the byte count into the HcTransferCounter register and check the HcBufferStatus register before reading from or writing to the buffer. The HCD must write the command (41H for read, C1H for write) once only, and then read or write all the data bytes in word. After every read/write, the pointer of ATL buffer RAM will be automatically increased by two to point to the next data word until it reaches the value of HcTransferCounter register; otherwise, an internal EOT signal is not generated to set the bit 2 (AllEOTInterrupt) of the HcμPInterrupt register and update the HcBufferStatus register.

The HCD must take care of the difference: the internal buffer RAM is organized in bytes, so the HCD must write the byte count into the HcTransferCounter register, but the HCD reads or writes the buffer RAM by 16 bits (by 1 word).

11. USB device controller (DC)

The Device Controller (DC) in ISP1161 is based on the Philips ISP1181x Full-Speed USB Interface Device product family. The functionality and command and register sets are the same as ISP1181x in the 16-bit bus mode. If there is any difference found in ISP1181x and ISP1161 datasheet in terms of DC functionality, the ISP1161 datasheet supersedes the content in the ISP1181x datasheet.

In general the DC in ISP1161 provides 16 endpoints for USB device implementation. Each endpoint can be allocated an amount of RAM space in the on-chip Ping-Pong buffer RAM.

Remark: the Ping-Pong buffer RAM for the DC is independent of the buffer RAM in the HC. When the buffer RAM is full, the DC will transfer the data in the buffer RAM to the USB bus. When the buffer RAM is empty, an interrupt is generated to notify the microprocessor to feed in the data. The transfer of data between the microprocessor and the DC can be done in Programmed I/O (PIO) mode or in DMA mode.

11.1 DC data transfer operation

The following session explains how the DC of ISP1161 handles an IN data transfer and an OUT data transfer. In Device mode, ISP1161 acts as a USB device: an IN data transfer means transfer from ISP1161 to an external USB Host (through the upstream port) and an OUT transfer means transfer from external USB Host to ISP1161.

11.1.1 IN data transfer

Data transfer procedure:

- The arrival of the IN token is detected by the SIE by decoding the PID.
- The SIE also checks for the device number and endpoint number and verifies whether they are acceptable.
- If the endpoint is enabled, the SIE checks the contents of the DcEndpointStatus register. If the endpoint is full, the contents of the FIFO are sent during the data phase, otherwise a Not Acknowledge (NAK) handshake is sent.
- After the data phase, the SIE expects a handshake (ACK) from the host (except for ISO endpoints).
- On receiving the handshake (ACK), the SIE updates the contents of the DcEndpointStatus register and the DcInterrupt register, which in turn generates an interrupt to the microprocessor. For ISO endpoints, the DcInterrupt register is updated as soon as data is sent because there is no handshake phase.
- On receiving an interrupt, the microprocessor reads the DcInterrupt register. It will know which endpoint has generated the interrupt and reads the contents of the corresponding DcEndpointStatus register. If the buffer is empty, it fills up the buffer, so that data can be sent by the SIE at the next IN token phase.

11.1.2 OUT data transfer

Data transfer procedure:

- The arrival of the OUT token is detected by the SIE by decoding the PID.

- The SIE also checks for the device number and endpoint number and verifies whether they are acceptable.
- If the endpoint is enabled, the SIE checks the contents of the DcEndpointStatus register. If the endpoint is empty, the data from USB is stored to FIFO during the data phase, otherwise a NAK handshake is sent.
- After the data phase, the SIE sends a handshake (ACK) to the host (except for ISO endpoints).
- The SIE updates the contents of the DcEndpointStatus register and the DcInterrupt register, which in turn generates an interrupt to the microprocessor. For ISO endpoints, the DcInterrupt register is updated as soon as data is received because there is no handshake phase.
- On receiving interrupt, the microprocessor reads the DcInterrupt register. It will know which endpoint has generated the interrupt and reads the content of the corresponding DcEndpointStatus register. If the buffer is full, it empties the buffer, so that data can be received by the SIE at the next OUT token phase.

11.2 Device DMA transfer

11.2.1 DMA for IN endpoint (internal DC to external USB host)

When the internal DMA handler is enabled and at least one buffer (Ping or Pong) is free, the DREQ2 line is asserted. The external DMA controller then starts negotiating for control of the bus. As soon as it has access, it asserts the DACK2 line and starts writing data. The burst length is programmable. When the number of bytes equal to the burst length has been written, the DREQ2 line is de-asserted. As a result, the DMA controller de-asserts the DACK2 line and releases the bus. At that moment the whole cycle restarts for the next burst.

When the buffer is full, the DREQ2 line will be de-asserted and the buffer is validated (which means that it will be sent to the host when the next IN token comes in). When the DMA transfer is terminated, the buffer is also validated (even if it is not full). A DMA transfer is terminated when any of the following conditions are met:

- DMA count is complete
- Bit DMAEN = 0
- DMA controller asserts EOT.

11.2.2 DMA for OUT endpoint (external USB host to internal DC)

When the internal DMA handler is enabled and at least one buffer is full, the DREQ2 line is asserted. The external DMA controller then starts negotiating for control of the bus, and as soon as it has access, it asserts the $\overline{\text{DACK2}}$ line and starts reading the data. The burst length is programmable. When the number of bytes equal to the burst length has been read, the DREQ2 line is de-asserted. As a result, the DMA controller de-asserts the $\overline{\text{DACK2}}$ line and releases the bus. At that moment the whole cycle restarts for the next burst. When all data are read, the DREQ2 line will be de-asserted and the buffer is cleared (which means that it can be overwritten when a new packet comes in).

A DMA transfer is terminated when any of the following conditions are met:

- DMA count is complete
- Bit DMAEN = 0
- DMA controller asserts EOT.

When the DMA transfer is terminated, the buffer is also cleared (even if the data is not completely read) and the DMA handler is disabled automatically. For the next DMA transfer, the DMA controller as well as the DMA handler must be re-enabled.

11.3 Endpoint descriptions

11.3.1 Endpoints with programmable FIFO size

Each USB device is logically composed of several independent endpoints. An endpoint acts as a terminus of a communication flow between the host and the device. At design time each endpoint is assigned a unique number (endpoint identifier, see Table 66). The combination of the device address (given by the host during enumeration), the endpoint number and the transfer direction allows each endpoint to be uniquely referenced.

The ISP1161's DC has 16 endpoints: endpoint 0 (control IN and OUT) plus 14 configurable endpoints, which can be individually defined as interrupt, bulk, isochronous and IN or OUT. Each enabled endpoint has an associated FIFO, which can be accessed either via the programmed I/O interface or via DMA.

11.3.2 Endpoint access

Table 66 lists the endpoint access modes and programmability. All endpoints support I/O mode access. Endpoints 1 to 14 also support DMA access. DC FIFO DMA access is selected and enabled via bits EPIDX[3:0] and DMAEN of the DcDMAConfiguration register. A detailed description of the DC DMA operation is given in Section 12.

Table 66: Endpoint access and programmability

Endpoint identifier	FIFO size[1] (bytes)	Double buffering	I/O mode access	DMA mode access	Endpoint type
0	64 (fixed)	no	yes	no	control OUT[2]
0	64 (fixed)	no	yes	no	control IN[2]
1 to 14	programmable	supported	supported	supported	programmable

[1] The total amount of FIFO storage allocated to enabled endpoints must not exceed 2462 bytes.

[2] IN: input for the USB host (ISP1161 transmits); OUT: output from the USB host (ISP1161 receives). The data flow direction is determined by bit EPDIR in the DcEndpointConfiguration register.

11.3.3 Endpoint FIFO size

The size of the FIFO determines the maximum packet size that the hardware can support for a given endpoint. Only enabled endpoints are allocated space in the shared FIFO storage, disabled endpoints have zero bytes. Table 67 lists the programmable FIFO sizes.

The following bits in the DcEndpointConfiguration register affect FIFO allocation:

- Endpoint enable bit (FIFOEN)

- Size bits of an enabled endpoint (FFOSZ[3:0])
- Isochronous bit of an enabled endpoint (FFOISO).

Remark: Register changes that affect the allocation of the shared FIFO storage among endpoints must **not** be made while valid data is present in any FIFO of the enabled endpoints. Such changes will render **all** FIFO contents **undefined**.

Table 67: Programmable FIFO size

FFOSZ[3:0]	Non-isochronous	Isochronous
0000	8 bytes	16 bytes
0001	16 bytes	32 bytes
0010	32 bytes	48 bytes
0011	64 bytes	64 bytes
0100	reserved	96 bytes
0101	reserved	128 bytes
0110	reserved	160 bytes
0111	reserved	192 bytes
1000	reserved	256 bytes
1001	reserved	320 bytes
1010	reserved	384 bytes
1011	reserved	512 bytes
1100	reserved	640 bytes
1101	reserved	768 bytes
1110	reserved	896 bytes
1111	reserved	1023 bytes

Each programmable FIFO can be configured independently via its DcEndpointConfiguration register, but the total physical size of all enabled endpoints (IN plus OUT) must not exceed 2462 bytes (512 bytes for non-isochronous FIFOs).

Table 68 shows an example of a configuration fitting in the maximum available space of 2462 bytes. The total number of logical bytes in the example is 1311. The physical storage capacity used for double buffering is managed by the device hardware and is transparent to the user.

Table 68: Memory configuration example

Physical size (bytes)	Logical size (bytes)	Endpoint description
64	64	control IN (64-byte fixed)
64	64	control OUT (64-byte fixed)
2046	1023	double-buffered 1023-byte isochronous endpoint
16	16	16-byte interrupt OUT
16	16	16-byte interrupt IN
128	64	double-buffered 64-byte bulk OUT
128	64	double-buffered 64-byte bulk IN

11.3.4 Endpoint initialization

In response to the standard USB request Set Interface, the firmware must program all 16 DcEndpointConfiguration registers of the ISP1161's DC in sequence (see Table 66), whether the endpoints are enabled or not. The hardware will then automatically allocate FIFO storage space.

If all endpoints have been configured successfully, the firmware must return an empty packet to the control IN endpoint to acknowledge success to the host. If there are errors in the endpoint configuration, the firmware must stall the control IN endpoint.

When reset by hardware or via the USB bus, the ISP1161's DC disables all endpoints and clears all DcEndpointConfiguration registers, except for the control endpoint which is fixed and always enabled.

Endpoint initialization can be done at any time; however, it is valid only after enumeration.

11.3.5 Endpoint I/O mode access

When an endpoint event occurs (a packet is transmitted or received), the associated endpoint interrupt bits (EPn) of the DcInterrupt register will be set by the SIE. The firmware then responds to the interrupt and selects the endpoint for processing.

The endpoint interrupt bit will be cleared by reading the DcEndpointStatus register. The DcEndpointStatus register also contains information on the status of the endpoint buffer.

For an OUT (receive) endpoint, the packet length and packet data can be read from ISP1161's DC using the Read Buffer command. When the whole packet has been read, the firmware sends a Clear Buffer command to enable the reception of new packets.

For an IN (transmit) endpoint, the packet length and data to be sent can be written to ISP1161's DC using the Write Buffer command. When the whole packet has been written to the buffer, the firmware sends a Validate Buffer command to enable data transmission to the host.

11.3.6 Special actions on control endpoints

Control endpoints require special firmware actions. The arrival of a SETUP packet flushes the IN buffer and disables the Validate Buffer and Clear Buffer commands for the control IN and OUT endpoints. The microprocessor needs to re-enable these commands by sending an Acknowledge SETUP command.

This ensures that the last SETUP packet stays in the buffer and that no packets can be sent back to the host until the microprocessor has explicitly acknowledged that it has seen the SETUP packet.

11.4 Suspend and resume

11.4.1 Suspend conditions

The ISP1161's DC detects a USB 'suspend' status in the following cases:

- A J-state is present on the USB bus for 3 ms
- V_{BUS} is lost (weak pull-up or pull-down on D+ and D-).

With SoftConnect disabled, ISP1161 does not go into the 'suspend' state as long as V_{BUS} is present.

ISP1161's DC will remain in 'suspend' state for at least 5 ms, before responding to external wake-up events such as global resume, bus traffic, wake-up on $\overline{CS}$ or WAKEUP. The typical timing is shown in Figure 38.

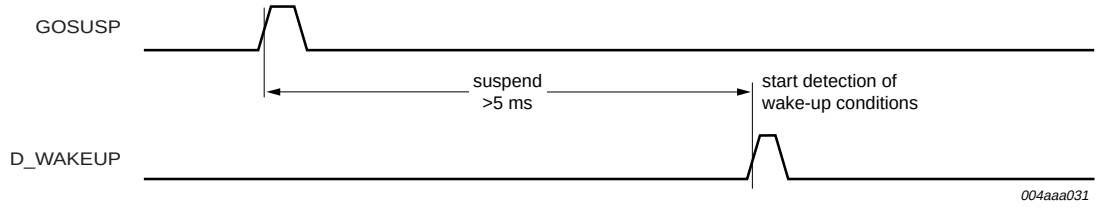


Fig 38. DC typical suspend timing.

Bus-powered devices that are suspended must not consume more than 500 µA of current. This is achieved by shutting down the power to system components or supplying them with a reduced voltage.

ISP1161's DC is always in powered-off mode during 'suspend' state. Default, bit PWROFF in the DcHardwareConfiguration register is logic 1 and this value should not be changed under any condition. This powered-off mode is explained in detail in Section 11.4.2.

The steps leading up to 'suspend' status are as follows:

1. Upon detection of a 'wake-up' to 'suspend' transition ISP1161's DC sets bit SUSPND in the DcInterrupt Register. This will generate an interrupt if bit IESUSP in the DcInterruptEnable register is set.
2. When the firmware detects a 'suspend' condition it must prepare all system components for 'suspend' state:
   a. All signals connected to ISP1161's DC must enter appropriate states to meet the power consumption requirements of 'suspend' state.
   b. All input pins of ISP1161's DC must have a CMOS logic 0 or logic 1 level.
3. In the interrupt service routine the firmware must check the current status of the USB bus. When bit BUSTATUS in the DcInterrupt Register is logic 0, the USB bus has left 'suspend' mode and the process must be aborted. Otherwise, the next step can be executed.
4. To meet the 'suspend' current requirements for a bus-powered device, the internal clocks must be switched off by clearing bit CLKRUN in the DcHardwareConfiguration register.

5. When the firmware has set and cleared the GOSUSP bit in the DcMode register, the ISP1161's DC enters 'suspend' state. In powered-off application, the ISP1161's DC asserts output D_SUSPEND and switches off the internal clocks (except LazyClock) after 2 ms.

11.4.2 Powered-off application

In powered-off application (bit PWROFF = 1 in the DcHardwareConfiguration register) the supply of the CPU and other parts of the circuit is removed during 'suspend' state. The D_SUSPEND output is active HIGH during 'suspend' state, making it suitable as a power switch control signal, e.g. for an external oscillator.

Input pins of ISP1161's DC are pulled to ground via the pin buffers. Outputs are made three-state to prevent current flowing in the application. Bi-directional pins are made three-state and must be pulled to ground externally by the application. The power supply of external pull-ups must also be removed to reduce power consumption.

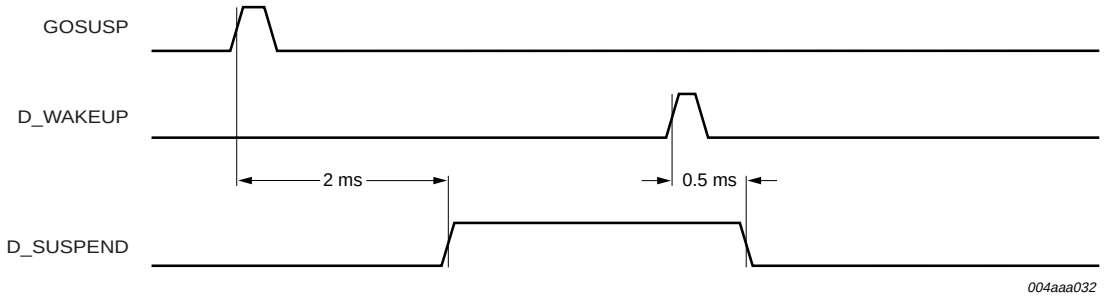


Fig 39. Suspend and resume timing for powered-off application (DC side).

Table 69: Pin states in powered-off application

Pin	Type	Appropriate state
A0	I	inactive
D[15:0]	I/O (three-state)	
D_SUSPEND	O	ISP1161's DC drives logic 1
D_WAKEUP	I	inactive
INT2	O	powered off; internally connected to ground (logic 0)
$\overline{\text{RESET}}$	I	externally driven [1] to logic 1
$\overline{\text{CS}}$	I	powered off; internally connected to ground (logic 0)
$\overline{\text{RD}}$	I	powered off; internally connected to ground (logic 0)
$\overline{\text{WR}}$	I	powered off; internally connected to ground (logic 0)
XTAL1	I	powered off; internally connected to ground (logic 0)
CLKOUT	O	ISP1161's DC drives logic 0, if the NOLAZY bit is set to logic 1 in the DcHardwareConfiguration register

[1] Externally driven refers to logic outside the ISP1161's DC.

When external components are powered-off, it is possible that interface signals $\overline{\text{RD}}$, $\overline{\text{WR}}$ and $\overline{\text{CS}}$ have unknown values immediately after leaving 'suspend' state. To prevent corruption of its internal registers, ISP1161's DC enables a locking mechanism once suspend is enabled.

After wake-up from ‘suspend’ state, all internal registers except the Unlock register are read and write protected. A special unlock operation is needed to re-enable write access. This prevents data corruption during power-up of external components.

Figure 40 shows a typical bus-powered modem application using ISP1161’s DC in powered-off mode. The SUSPEND output is used to switch off power to the microprocessor and other external circuits during ‘suspend’ state. The ISP1161’s DC is woken up via the USB bus (global resume) or by the ring detection circuit on the telephone line.

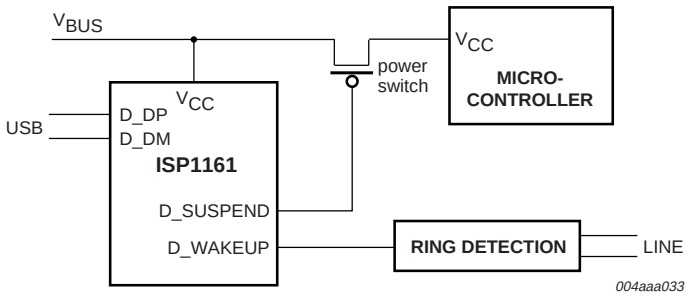


Fig 40. ISP1161’s DC SUSPEND and WAKEUP signals in a powered-off modem application.

11.4.3 Resume conditions

Wake-up from ‘suspend’ state is initiated either by the USB host or by the application:

- **USB host**: drives a K-state on the USB bus (global resume)
- **Application**: remote wake-up via a HIGH level on input WAKEUP or a LOW level on input $\overline{CS}$ (if enabled via bit WKUPCS in the DcHardwareConfiguration register). Wake-up on $\overline{CS}$ will work only if V_{BUS} is present.

The steps of a wake-up sequence are as follows:

1. The internal oscillator and the PLL multiplier are re-enabled. When stabilized, the clock signals are routed to all internal circuits of the ISP1161’s DC.
2. The D_SUSPEND output is de-asserted and the RESUME bit in the DcInterrupt Register is set. This will generate an interrupt if bit IERESUME in the DcInterruptEnable Register is set.
3. Maximum 15 ms after starting the wake-up sequence the ISP1161’s DC resumes its normal functionality.
4. In case of a remote wake-up ISP1161’s DC drives a K-state on the USB bus for 10 ms.
5. Following the de-assertion of output D_SUSPEND, the application restores itself and other system components to normal operating mode.
6. After wake-up the internal registers of ISP1161’s DC are read and write protected to prevent corruption by inadvertent writing during power-up of external components. The firmware must send an Unlock Device command to the ISP1161’s DC to restore its full functionality. See Section 13.3.2 for more details.

11.4.4 Control bits in suspend and resume

Table 70: Summary of control bits

Register	Bit	Function
DcInterrupt	SUSPND	a transition from ‘awake’ to ‘suspend’ state was detected
	BUSTATUS	monitors USB bus status (logic 1 is suspend); used when interrupt is serviced
DcInterruptEnable	IESUSP	enables output INT to signal ‘suspend’ state
DcMode	SOFTCT	enables SoftConnect pull-up resistor to USB bus
	GOSUSP	a HIGH-to-LOW transition enables ‘suspend’ state
DcHardwareConfiguration	EXTPUL	selects internal (SoftConnect) or external pull-up resistor
	WKUPCS	enables wake-up on LOW level of input $\overline{CS}$
	PWROFF	selects powered-off mode during ‘suspend’ state
Unlock	all	sending data AA37H unlocks the internal registers for writing after a ‘resume’

12. DC DMA transfer

Direct Memory Access (DMA) is a method to transfer data from one location to another in a computer system, without intervention of the Central Processor Unit (CPU). Many different implementations of DMA exist. The ISP1161 DC supports two methods:

- **8237 compatible mode**: based on the DMA subsystem of the IBM personal computers (PC, AT and all its successors and clones); this architecture uses the Intel 8237 DMA controller and has separate address spaces for memory and I/O
- **DACK-only mode**: based on the DMA implementation in some embedded RISC processors, which has a single address space for both memory and I/O.

ISP1161's DC supports DMA transfer for all 14 configurable endpoints (see Table 66). Only one endpoint at a time can be selected for DMA transfer. The DMA operation of ISP1161's DC can be interleaved with normal I/O mode access to other endpoints.

The following features are supported:

- Single-cycle or burst transfers (up to 16 bytes per cycle)
- Programmable transfer direction (read or write)
- Multiple End-Of-Transfer (EOT) sources: external pin, internal conditions, short/empty packet
- Programmable signal levels on pins DREQ2 and EOT.

12.1 Selecting an endpoint for DMA transfer

The target endpoint for DMA access is selected via bits EPDIX[3:0] in the DcDMAConfiguration register, as shown in Table 71. The transfer direction (read or write) is automatically set by bit EPDIR in the associated DcEndpointConfiguration register, to match the selected endpoint type (OUT endpoint: read; IN endpoint: write).

Asserting input DACK2 automatically selects the endpoint specified in the DcDMAConfiguration register, regardless of the current endpoint used for I/O mode access.

Table 71: Endpoint selection for DMA transfer

Endpoint identifier	EPIDX[3:0]	Transfer direction	
		EPDIR = 0	EPDIR = 1
1	0010	OUT: read	IN: write
2	0011	OUT: read	IN: write
3	0100	OUT: read	IN: write
4	0101	OUT: read	IN: write
5	0110	OUT: read	IN: write
6	0111	OUT: read	IN: write
7	1000	OUT: read	IN: write
8	1001	OUT: read	IN: write
9	1010	OUT: read	IN: write

Table 71: Endpoint selection for DMA transfer *...continued*

Endpoint identifier	EPIDX[3:0]	Transfer direction	
		EPDIR = 0	EPDIR = 1
10	1011	OUT: read	IN: write
11	1100	OUT: read	IN: write
12	1101	OUT: read	IN: write
13	1110	OUT: read	IN: write
14	1111	OUT: read	IN: write

12.2 8237 compatible mode

The 8237 compatible DMA mode is selected by clearing bit DAKOLY in the DcHardwareConfiguration register (see Table 83). The pin functions for this mode are shown in Table 72.

Table 72: 8237 compatible mode: pin functions

Symbol	Description	I/O	Function
DREQ2	DC's DMA request	O	ISP1161's DC requests a DMA transfer
$\overline{DACK2}$	DC's DMA acknowledge	I	DMA controller confirms the transfer
EOT	end of transfer	I	DMA controller terminates the transfer
$\overline{RD}$	read strobe	I	instructs ISP1161's DC to put data on the bus
$\overline{WR}$	write strobe	I	instructs ISP1161's DC to get data from the bus

The DMA subsystem of an IBM compatible PC is based on the Intel 8237 DMA controller. It operates as a 'fly-by' DMA controller: the data is not stored in the DMA controller, but it is transferred between an I/O port and a memory address. A typical example of ISP1161's DC in 8237 compatible DMA mode is given in Figure 41.

The 8237 has two control signals for each DMA channel: DREQ (DMA Request) and $\overline{DACK}$ (DMA Acknowledge). General control signals are HRQ (Hold Request) and HLDA (Hold Acknowledge). The bus operation is controlled via $\overline{MEMR}$ (Memory read), $\overline{MEMW}$ (Memory write), $\overline{IOR}$ (I/O read) and $\overline{IOW}$ (I/O write).

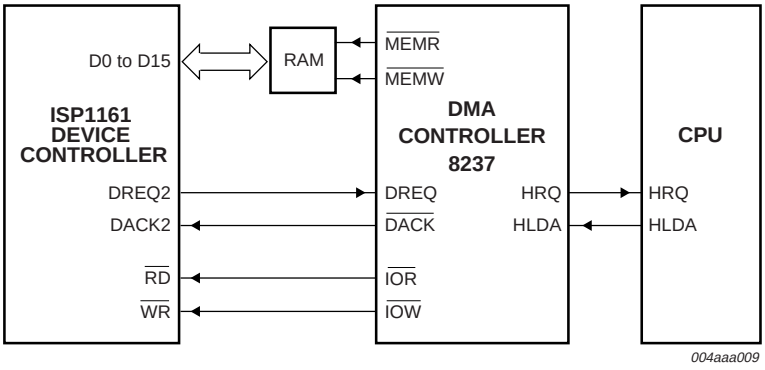


Fig 41. ISP1161's device controller in 8237 compatible DMA mode.

The following example shows the steps which occur in a typical DMA transfer:

1. ISP1161's DC receives a data packet in one of its endpoint FIFOs; the packet must be transferred to memory address 1234H.
2. ISP1161's DC asserts the DREQ2 signal requesting the 8237 for a DMA transfer.
3. The 8237 asks the CPU to release the bus by asserting the HRQ signal.
4. After completing the current instruction cycle, the CPU places the bus control signals ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$) and the address lines in three-state and asserts HLDA to inform the 8237 that it has control of the bus.
5. The 8237 now sets its address lines to 1234H and activates the $\overline{MEMW}$ and $\overline{IOR}$ control signals.
6. The 8237 asserts $\overline{DACK}$ to inform ISP1161's DC that it will start a DMA transfer.
7. ISP1161's DC now places the word to be transferred on the data bus lines, because its $\overline{RD}$ signal was asserted by the 8237.
8. The 8237 waits one DMA clock period and then de-asserts $\overline{MEMW}$ and $\overline{IOR}$. This latches and stores the word at the desired memory location. It also informs ISP1161's DC that the data on the bus lines has been transferred.
9. ISP1161's DC de-asserts the DREQ2 signal to indicate to the 8237 that DMA is no longer needed. In **Single cycle mode** this is done after each word, in **Burst mode** following the last transferred word of the DMA cycle.
10. The 8237 de-asserts the $\overline{DACK}$ output indicating that ISP1161's DC must stop placing data on the bus.
11. The 8237 places the bus control signals ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$) and the address lines in three-state and de-asserts the HRQ signal, informing the CPU that it has released the bus.
12. The CPU acknowledges control of the bus by de-asserting HLDA. After activating the bus control lines ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$) and the address lines, the CPU resumes the execution of instructions.

For a typical bulk transfer the above process is repeated, once for each byte. After each byte the address register in the DMA controller is incremented and the byte counter is decremented. When using 16-bit DMA the number of transfers is 32 and address incrementing and byte counter decrementing is done by 2 for each word.

12.3 DACK-only mode

The DACK-only DMA mode is selected by setting bit DAKOLY in the DcHardwareConfiguration register (see Table 83). The pin functions for this mode are shown in Table 73. A typical example of ISP1161's DC in DACK-only DMA mode is given in Figure 42.

Table 73: DACK-only mode: pin functions

Symbol	Description	I/O	Function
DREQ2	DC's DMA request	O	ISP1161 DC requests a DMA transfer
$\overline{DACK2}$	DC's DMA acknowledge	I	DMA controller confirms the transfer; also functions as data strobe

Table 73: DACK-only mode: pin functions...*continued*

Symbol	Description	I/O	Function
EOT	End-Of-Transfer	I	DMA controller terminates the transfer
$\overline{RD}$	read strobe	I	not used
$\overline{WR}$	write strobe	I	not used

In DACK-only mode ISP1161's DC uses the DACK2 signal as a data strobe. Input signals $\overline{RD}$ and $\overline{WR}$ are ignored. This mode is used in CPU systems that have a single address space for memory and I/O access. Such systems have no separate $\overline{MEMW}$ and $\overline{MEMR}$ signals: the $\overline{RD}$ and $\overline{WR}$ signals are also used as memory data strobes.

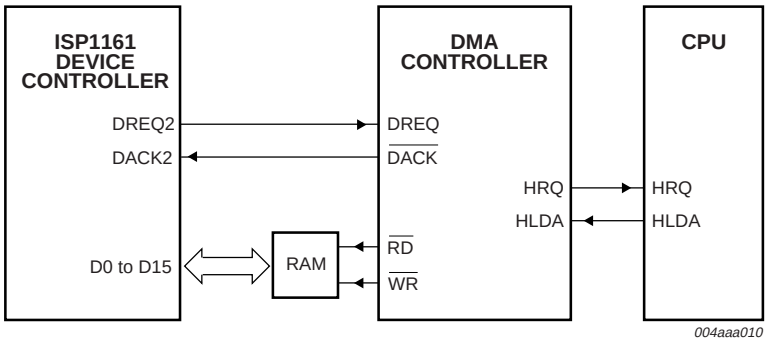


Fig 42. ISP1161's device controller in DACK-only DMA mode.

12.4 End-Of-Transfer conditions

12.4.1 Bulk endpoints

A DMA transfer to/from a bulk endpoint can be terminated by any of the following conditions (bit names refer to the DcDMAConfiguration register, see Table 87):

- An external End-Of-Transfer signal occurs on input EOT
- The DMA transfer completes as programmed in the DcDMACounter register (CNTREN = 1)
- A short packet is received on an enabled OUT endpoint (SHORTP = 1)
- DMA operation is disabled by clearing bit DMAEN.

External EOT: When reading from an OUT endpoint, an external EOT will stop the DMA operation and **clear any remaining data** in the current FIFO. For a double-buffered endpoint the other (inactive) buffer is not affected.

When writing to an IN endpoint, an EOT will stop the DMA operation and the data packet in the FIFO (even if it is smaller than the maximum packet size) will be sent to the USB host at the next IN token.

DcDMACounter register: An EOT from the DcDMACounter register is enabled by setting bit CNTREN in the DcDMAConfiguration register. The ISP1161 has a 16-bit DcDMACounter register, which specifies the number of bytes to be transferred. When DMA is enabled (DMAEN = 1), the internal DMA counter is loaded with the value from

the DcDMACounter register. When the internal counter completes the transfer as programmed in the DMA counter, an EOT condition is generated and the DMA operation stops.

Short packet: Normally, the transfer byte count must be set via a control endpoint before any DMA transfer takes place. When a short packet has been 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.

When reading from an OUT endpoint, reception of a short packet at an OUT token will stop the DMA operation after transferring the data bytes of this packet.

Table 74: Summary of EOT conditions for a bulk endpoint

EOT condition	OUT endpoint	IN endpoint
EOT input	EOT is active	EOT is active
DcDMACounter register	transfer completes as programmed in the DcDMACounter register	transfer completes as programmed in the DcDMACounter register
Short packet	short packet is received and transferred	counter reaches zero in the middle of the buffer
DMAEN bit in DcDMAConfiguration register	DMAEN = 0[1]	DMAEN = 0[1]

[1] The DMA transfer stops. However, no interrupt is generated.

12.4.2 Isochronous endpoints

A DMA transfer to/from an isochronous endpoint can be terminated by any of the following conditions (bit names refer to the DcDMAConfiguration register, see Table 87):

- An external End-Of-Transfer signal occurs on input EOT
- The DMA transfer completes as programmed in the DcDMACounter register (CNTREN = 1)
- An End-Of-Packet (EOP) signal is detected
- DMA operation is disabled by clearing bit DMAEN.

Table 75: Recommended EOT usage for isochronous endpoints

EOT condition	OUT endpoint	IN endpoint
EOT input active	do not use	preferred
DcDMACounter register zero	do not use	preferred
End-Of-Packet	preferred	do not use

13. DC commands and registers

The functions and registers of ISP1161's DC are accessed via commands, which consist of a command code followed by optional data bytes (read or write action). An overview of the available commands and registers is given in Table 76.

A complete access consists of two phases:

1. **Command phase**: when address bit A0 = 1, the DC interprets the data on the lower byte of the bus (bits D7 to D0) as a command code. Commands without a data phase are executed immediately.
2. **Data phase (optional)**: when address bit A0 = 0, the DC transfers the data on the bus to or from a register or endpoint FIFO. Multi-byte registers are accessed least significant byte/word first.

As the ISP1161 DC's data bus is 16 bits wide:

- The upper byte (bits D15 to D8) in command phase or the undefined byte in data phase is ignored.
- The access of registers is word-aligned: byte access is not allowed.
- If the packet length is odd, the upper byte of the last word in an IN endpoint buffer is **not** transmitted to the host. When reading from an OUT endpoint buffer, the upper byte of the last word must be ignored by the firmware. The packet length is stored in the first 2 bytes of the endpoint buffer.

Table 76: Command and register summary

Name	Destination	Code (Hex)	Transaction[1]
Initialization commands			
Write Control OUT Configuration	DcEndpointConfiguration register endpoint 0 OUT	20	write 1 word
Write Control IN Configuration	DcEndpointConfiguration register endpoint 0 IN	21	write 1 word
Write Endpoint n Configuration (n = 1 to 14)	DcEndpointConfiguration register endpoint 1 to 14	22 to 2F	write 1 word
Read Control OUT Configuration	DcEndpointConfiguration register endpoint 0 OUT	30	read 1 word
Read Control IN Configuration	DcEndpointConfiguration register endpoint 0 IN	31	read 1 word
Read Endpoint n Configuration (n = 1 to 14)	DcEndpointConfiguration register endpoint 1 to 14	32 to 3F	read 1 word
Write/Read Device Address	DcAddress register	B6/B7	write/read 1 word
Write/Read DcMode register	DcMode register	B8/B9	write/read 1 word
Write/Read Hardware Configuration	DcHardwareConfiguration register	BA/BB	write/read 1 word
Write/Read DcInterruptEnable register	DcInterruptEnable register	C2/C3	write/read 2 words
Write/Read DMA Configuration	DcDMAConfiguration register	F0/F1	write/read 1 word
Write/Read DMA Counter	DcDMACounter register	F2/F3	write/read 1 word
Reset Device	resets all registers	F6	-

Table 76: Command and register summary…*continued*

Name	Destination	Code (Hex)	Transaction[1]
Data flow commands			
Write Control OUT Buffer	illegal: endpoint is read-only	(00)	-
Write Control IN Buffer	FIFO endpoint 0 IN	01	N ≤ 64 bytes
Write Endpoint n Buffer (n = 1 to 14)	FIFO endpoint 1 to 14 (IN endpoints only)	02 to 0F	isochronous: N ≤ 1023 bytes; interrupt/bulk: N ≤ 64 bytes
Read Control OUT Buffer	FIFO endpoint 0 OUT	10	N ≤ 64 bytes
Read Control IN Buffer	illegal: endpoint is write-only	(11)	-
Read Endpoint n Buffer (n = 1 to 14)	FIFO endpoint 1 to 14 (OUT endpoints only)	12 to 1F	isochronous: N ≤ 1023 bytes[2]; interrupt/bulk: N ≤ 64 bytes
Stall Control OUT Endpoint	Endpoint 0 OUT	40	-
Stall Control IN Endpoint	Endpoint 0 IN	41	-
Stall Endpoint n (n = 1 to 14)	Endpoint 1 to 14	42 to 4F	-
Read Control OUT Status	DcEndpointStatus register endpoint 0 OUT	50	read 1 word
Read Control IN Status	DcEndpointStatus register endpoint 0 IN	51	read 1 word
Read Endpoint n Status (n = 1 to 14)	DcEndpointStatus register n endpoint 1 to 14	52 to 5F	read 1 word
Validate Control OUT Buffer	illegal: IN endpoints only[3]	(60)	-
Validate Control IN Buffer	FIFO endpoint 0 IN[3]	61	none
Validate Endpoint n Buffer (n = 1 to 14)	FIFO endpoint 1 to 14 (IN endpoints only)[3]	62 to 6F	none
Clear Control OUT Buffer	FIFO endpoint 0 OUT	70	none
Clear Control IN Buffer	illegal[4]	(71)	-
Clear Endpoint n Buffer (n = 1 to 14)	FIFO endpoint 1 to 14 (OUT endpoints only)[4]	72 to 7F	none
Unstall Control OUT Endpoint	Endpoint 0 OUT	80	-
Unstall Control IN Endpoint	Endpoint 0 IN	81	-
Unstall Endpoint n (n = 1 to 14)	Endpoint 1 to 14	82 to 8F	-
Check Control OUT Status[5]	DcEndpointStatusImage register endpoint 0 OUT	D0	read 1 word
Check Control IN Status[5]	DcEndpointStatusImage register endpoint 0 IN	D1	read 1 word
Check Endpoint n Status (n = 1 to 14)[5]	DcEndpointStatusImage register n endpoint 1 to 14	D2 to DF	read 1 word
Acknowledge SETUP	Endpoint 0 IN and OUT	F4	none
General commands			
Read Control OUT Error Code	DcErrorCode register endpoint 0 OUT	A0	read 1 word [6]
Read Control IN Error Code	DcErrorCode register endpoint 0 IN	A1	read 1 word [6]

Table 76: Command and register summary…*continued*

Name	Destination	Code (Hex)	Transaction[1]
Read Endpoint n Error Code (n = 1 to 14)	DcErrorCode register endpoint 1 to 14	A2 to AF	read 1 word [6]
Unlock Device	all registers with write access	B0	write 1 word
Write/Read DcScratch register	DcScratch register	B2/B3	write/read 1 word
Read Frame Number	DcFrameNumber register	B4	read 1 word
Read Chip ID	DcChipID register	B5	read 1 word
Read DcInterrupt register	DcInterrupt register	C0	read 2 words

[1] With N representing the number of bytes; the number of words for 16-bit bus width is: (N + 1)/2.

[2] During isochronous transfer in 16-bit mode, because N ≤ 1023, the firmware must take care of the upper byte.

[3] Validating an OUT endpoint buffer causes unpredictable behavior of ISP1161's DC.

[4] Clearing an IN endpoint buffer causes unpredictable behavior of ISP1161's DC.

[5] Reads a copy of the Status register: executing this command does not clear any status bits or interrupt bits.

[6] When accessing an 8-bit register in 16-bit mode, the upper byte is invalid.

13.1 Initialization commands

Initialization commands are used during the enumeration process of the USB network. These commands are used to configure and enable the embedded endpoints. They also serve to set the USB assigned address of ISP1161's DC and to perform a device reset.

13.1.1 DcEndpointConfiguration (R/W: 30H–3FH/20H–2FH)

This command is used to access the DcEndpointConfiguration register of the target endpoint. It defines the endpoint type (isochronous or bulk/interrupt), direction (OUT/IN), FIFO size and buffering scheme. It also enables the endpoint FIFO. The register bit allocation is shown in Table 77. A bus reset will disable all endpoints.

The allocation of FIFO memory only takes place after **all** 16 endpoints have been configured in sequence (from endpoint 0 OUT to endpoint 14). Although the control endpoints have fixed configurations, they must be included in the initialization sequence and be configured with their default values (see Table 66). Automatic FIFO allocation starts when endpoint 14 has been configured.

Remark: If any change is made to an endpoint configuration which affects the allocated memory (size, enable/disable), the FIFO memory contents of **all** endpoints becomes invalid. Therefore, all valid data must be removed from enabled endpoints before changing the configuration.

Code (Hex): 20 to 2F — write (control OUT, control IN, endpoint 1 to 14)

Code (Hex): 30 to 3F — read (control OUT, control IN, endpoint 1 to 14)

Transaction — write/read 1 word

Table 77: DcEndpointConfiguration register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	FIFOEN	EPDIR	DBLBUF	FFOISO	FFOSZ[3:0]			
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 78: DcEndpointConfiguration register: bit description

Bit	Symbol	Description
7	FIFOEN	Logic 1 indicates an enabled FIFO with allocated memory. Logic 0 indicates a disabled FIFO (no bytes allocated).
6	EPDIR	This bit defines the endpoint direction (0 = OUT, 1 = IN); it also determines the DMA transfer direction (0 = read, 1 = write).
5	DBLBUF	Logic 1 indicates that this endpoint has double buffering.
4	FFOISO	Logic 1 indicates an isochronous endpoint. Logic 0 indicates a bulk or interrupt endpoint.
3 to 0	FFOSZ[3:0]	Selects the FIFO size according to Table 67.

13.1.2 DcAddress Register (R/W: B7H/B6H)

This command is used to set the USB assigned address in the DcAddress register and enable the USB device. The DcAddress register bit allocation is shown in Table 79.

A USB bus reset sets the device address to 00H (internally) and enables the device. The value of the DcAddress register (accessible by the microprocessor) is not altered by the bus reset. In response to the standard USB request Set Address the firmware must issue a Write Device Address command, followed by sending an empty packet to the host. The **new** device address is activated when the host acknowledges the empty packet.

Code (Hex): B6/B7 — write/read DcAddress register

Transaction — write/read 1 word

Table 79: DcAddress register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	DEVEN	DEVADR[6:0]						
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 80: DcAddress register: bit description

Bit	Symbol	Description
7	DEVEN	Logic 1 enables the device.
6 to 0	DEVADR[6:0]	This field specifies the USB device address.

13.1.3 DcMode Register (R/W: B9H/B8H)

This command is used to access the ISP1161's DcMode register, which consists of 1 byte (bit allocation: see Table 80). In 16-bit bus mode the upper byte is ignored.

The DcMode register controls the DMA bus width, resume and suspend modes, interrupt activity and SoftConnect operation. It can be used to enable debug mode, where all errors and Not Acknowledge (NAK) conditions will generate an interrupt.

Code (Hex): B8/B9 — write/read DcMode register

Transaction — write/read 1 word

Table 81: DcMode register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	DMAWD	reserved	GOSUSP	reserved	INTENA	DBGMOD	reserved	SOFTCT
Reset	0[1]	0	0	0	0[1]	0[1]	0[1]	0[1]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

[1] Unchanged by a bus reset.

Table 82: DcMode register: bit description

Bit	Symbol	Description
7	DMAWD	Logic 1 selects 16-bit DMA bus width (bus configuration modes 0 and 2). Logic 0 selects 8-bit DMA bus width. Bus reset value: unchanged.
6	-	reserved
5	GOSUSP	Writing logic 1 followed by logic 0 will activate ‘suspend’ mode.
4	-	reserved
3	INTENA	Logic 1 enables all interrupts. Bus reset value: unchanged; or details, see Section 8.6.3.
2	DBGMOD	Logic 1 enables debug mode, where all NAKs and errors will generate an interrupt. Logic 0 selects normal operation, where interrupts are generated on every ACK (bulk endpoints) or after every data transfer (isochronous endpoints). Bus reset value: unchanged.
1	-	reserved
0	SOFTCT	Logic 1 enables SoftConnect (see Section 7.5). This bit is ignored if EXTPUL = 1 in the DcHardwareConfiguration register (see Table 83). Bus reset value: unchanged.

13.1.4 DcHardwareConfiguration Register (R/W: BBH/BAH)

This command is used to access the DcHardwareConfiguration register, which consists of 2 bytes. The first (lower) byte contains the device configuration and control values, the second (upper) byte holds the clock control bits and the clock division factor. The bit allocation is given in Table 83. A bus reset will not change any of the programmed bit values.

The DcHardwareConfiguration register controls the connection to the USB bus, clock activity and power supply during ‘suspend’ state, output clock frequency, DMA operating mode and pin configurations (polarity, signalling mode).

Code (Hex): BA/BB — write/read DcHardwareConfiguration register

Transaction — write/read 1 word

Table 83: DcHardwareConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved	EXTPUL	NOLAZY	CLKRUN	CKDIV[3:0]			
Reset	0	0	1	0	0	0	1	1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Symbol	DAKOLY	DRQPOL	DAKPOL	EOTPOL	WKUPCS	PWROFF	INTLVL	INTPOL
Reset	0	1	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

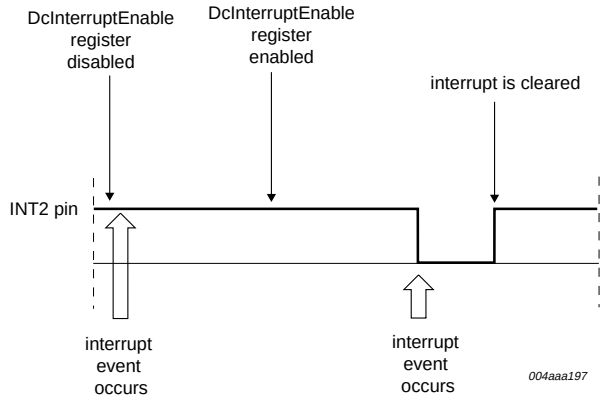
Table 84: DcHardwareConfiguration register: bit description

Bit	Symbol	Description
15	-	reserved
14	EXTPUL	Logic 1 indicates that an external 1.5 kΩ pull-up resistor is used on line D+ and that SoftConnect is not used. Bus reset value: unchanged.
13	NOLAZY	Logic 1 disables output on pin CLKOUT of the LazyClock frequency (100 kHz ±50%) during 'suspend' state. Logic 0 causes pin CLKOUT to switch to LazyClock output after approximately 2 ms delay, following the setting of bit GOSUSP in the DcMode register. Bus reset value: unchanged.
12	CLKRUN	Logic 1 indicates that the internal clocks are always running, even during 'suspend' state. Logic 0 switches off the internal oscillator and PLL, when they are not needed. During 'suspend' state this bit must be made logic 0 to meet the suspend current requirements. The clock is stopped after a delay of approximately 2 ms, following the setting of bit GOSUSP in the DcMode register. Bus reset value: unchanged.
11 to 8	CKDIV[3:0]	This field specifies the clock division factor N, which controls the clock frequency on output CLKOUT. The output frequency in MHz is given by 48/(N + 1). The clock frequency range is 3 to 48 MHz (N = 0 to 15) with a reset value of 12 MHz (N = 3). The hardware design guarantees no glitches during frequency change. Bus reset value: unchanged.
7	DAKOLY	Logic 1 selects DACK-only DMA mode. Logic 0 selects 8237 compatible DMA mode. Bus reset value: unchanged.
6	DRQPOL	Selects DREQ2 pin signal polarity (0 = active LOW, 1 = active HIGH). Bus reset value: unchanged.
5	DAKPOL	Selects DACK2 pin signal polarity (0 = active LOW). Bus reset value: unchanged.
4	EOTPOL	Selects EOT pin signal polarity (0 = active LOW, 1 = active HIGH). Bus reset value: unchanged.
3	WKUPCS	Logic 1 enables remote wake-up via a LOW level on input pin $\overline{CS}$ (For wake-up on $\overline{CS}$ to work, V_{BUS} must be present). Bus reset value: unchanged.
2	PWROFF	Logic 1 enables powering-off during 'suspend' state. Output D_SUSPEND pin is configured as a power switch control signal for external devices (HIGH during 'suspend'). This value should always be initialized to logic 1. Bus reset value: unchanged.
1	INTLVL	Selects the interrupt signalling mode on output pin INT2 (0 = level, 1 = pulsed). In pulsed mode an interrupt produces an 166 ns pulse. See Section 8.6.3 for details. Bus reset value: unchanged.
0	INTPOL	Selects INT2 pin signal polarity (0 = active LOW, 1 = active HIGH). Bus reset value: unchanged.

13.1.5 DcInterruptEnable Register (R/W: C3H/C2H)

This command is used to individually enable or disable interrupts from all endpoints, as well as interrupts caused by events on the USB bus (SOF, SOF lost, EOT, suspend, resume, reset). That is, if an interrupt event occurs while the interrupt is not enabled, nothing will be seen on the interrupt pin. Even if you then enable the interrupt during the interrupt event, there will still be no interrupt seen on the interrupt pin, see Figure 43.

The DcInterrupt register will not register any interrupt, if it is not already enabled using the DcInterruptEnable register. The DcInterruptEnable register is not an Interrupt Mask register.



Pin INT2: HIGH = de-assert; LOW = assert; INTENA = 1.

Fig 43. Interrupt pin waveform.

A bus reset will not change any of the programmed bit values.

The command accesses the DcInterruptEnable register, which consists of 4 bytes. The bit allocation is given in Table 85.

Remark: For details on interrupt control, see Section 8.6.3.

Code (Hex): C2/C3 — write/read DcInterruptEnable register

Transaction — write/read 2 words

Table 85: DcInterruptEnable register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	IEP14	IEP13	IEP12	IEP11	IEP10	IEP9	IEP8	IEP7
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Symbol	IEP6	IEP5	IEP4	IEP3	IEP2	IEP1	IEP0IN	IEP0OUT
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved		IEPSOF	IESOF	IEEOT	IESUSP	IERESM	IERST
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 86: DcInterruptEnable register: bit description

Bit	Symbol	Description
31 to 24	-	reserved; must write logic 0
23 to 10	IEP14 to IEP1	Logic 1 enables interrupts from the indicated endpoint.
9	IEP0IN	Logic 1 enables interrupts from the control IN endpoint.
8	IEP0OUT	Logic 1 enables interrupts from the control OUT endpoint.
7 to 6	-	reserved
5	IEPSOF	Logic 1 enables 1 ms interrupts upon detection of Pseudo SOF.
4	IESOF	Logic 1 enables interrupt upon SOF detection.
3	IEEOT	Logic 1 enables interrupt upon EOT detection.
2	IESUSP	Logic 1 enables interrupt upon detection of ‘suspend’ state.
1	IERESM	Logic 1 enables interrupt upon detection of a ‘resume’ state.
0	IERST	Logic 1 enables interrupt upon detection of a bus reset.

13.1.6 DcDMAConfiguration Register (R/W: F1H/F0H)

This command defines the DMA configuration of ISP1161’s DC and enables/disables DMA transfers. The command accesses the DcDMAConfiguration register, which consists of 2 bytes. The bit allocation is given in Table 87. A bus reset will clear bit DMAEN (DMA disabled), all other bits remain unchanged.

Code (Hex): F0/F1 — write/read DMA Configuration

Transaction — write/read 1 word

Table 87: DcDMAConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	CNTREN	SHORTP	reserved	reserved	reserved	reserved	reserved	reserved
Reset	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	EPDIX[3:0]				DMAEN	reserved	BURSTL[1:0]	
Reset	0[1]	0[1]	0[1]	0[1]	0	0	0[1]	0[1]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

[1] Unchanged by a bus reset.

Table 88: DcDMAConfiguration register: bit description

Bit	Symbol	Description
15	CNTREN	Logic 1 enables the generation of an EOT condition, when the DcDMACounter register reaches zero. Bus reset value: unchanged.
14	SHORTP	Logic 1 enables short/empty packet mode. When receiving (OUT endpoint) a short/empty packet an EOT condition is generated. When transmitting (IN endpoint) this bit should be cleared. Bus reset value: unchanged.
13 to 8	-	reserved
7 to 4	EPDIX[3:0]	Indicates the destination endpoint for DMA, see Table 71.
3	DMAEN	Writing logic 1 enables DMA transfer, logic 0 forces the end of an ongoing DMA transfer. Reading this bit indicates whether DMA is enabled (0 = DMA stopped, 1 = DMA enabled). This bit is cleared by a bus reset.
2	-	reserved
1 to 0	BURSTL[1:0]	Selects the DMA burst length: **00 —** single-cycle mode (1 byte) **01 —** burst mode (4 bytes) **10 —** burst mode (8 bytes) **11 —** burst mode (16 bytes). Bus reset value: unchanged.

For selecting an endpoint for device DMA transfer, see Section 11.2.

13.1.7 DcDMACounter Register (R/W: F3H/F2H)

This command accesses the DcDMACounter register. The bit allocation is given in Table 89. Writing to the register sets the number of bytes for a DMA transfer. Reading the register returns the number of remaining bytes in the current transfer. A bus reset will not change the programmed bit values.

The internal DMA counter is automatically reloaded from the DcDMACounter register when DMA is re-enabled (DMAEN = 1). See Section 13.1.6 for more details.

Code (Hex): F2/F3 — write/read DcDMACounter register

Transaction — write/read 1 word

Table 89: DcDMACounter register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	DMACR[15:8]							
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	DMACR[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 90: DcDMACounter register: bit description

Bit	Symbol	Description
15 to 0	DMACR[15:0]	DMA Counter register

13.1.8 Reset device (F6H)

This command resets the ISP1161 DC in the same way as an external hardware reset via input $\overline{\text{RESET}}$. All registers are initialized to their 'reset' values.

Code (Hex): F6 — reset the device

Transaction — none

13.2 Data flow commands

Data flow commands are used to manage the data transmission between the USB endpoints and the system microprocessor. Much of the data flow is initiated via an interrupt to the microprocessor. The data flow commands are used to access the endpoints and determine whether the endpoint FIFOs contain valid data.

Remark: The IN buffer of an endpoint contains input data **for** the host, the OUT buffer receives output data **from** the host.

13.2.1 Write/Read Endpoint Buffer (R/W: 10H,12H-1FH/01H–0FH)

This command is used to access endpoint FIFO buffers for reading or writing. First, the buffer pointer is reset to the beginning of the buffer. Following the command, a maximum of (M + 1) words can be written or read, with M given by (N + 1)/2 with N representing the size of the endpoint buffer. After each read/write action the buffer pointer is automatically incremented by 2.

In DMA access the first word (the packet length) is skipped: transfers start at the second word of the endpoint buffer. When reading, the ISP1161 DC can detect the last word via the End of Packet (EOP) condition. When writing to a bulk/interrupt endpoint, the endpoint buffer must be completely filled before sending the data to the host. Exception: when a DMA transfer is stopped by an external EOT condition, the current buffer content (full or not) is sent to the host.

Remark: Reading data after a Write Endpoint Buffer command or writing data after a Read Endpoint Buffer command will cause unpredictable behavior of the ISP1161 DC.

Code (Hex): 01 to 0F — write (control IN, endpoint 1 to 14)

Code (Hex): 10, 12 to 1F — read (control OUT, endpoint 1 to 14)

Transaction — write/read maximum (M + 1) words (isochronous endpoint: $N \leq 1023$, bulk/interrupt endpoint: $N \leq 32$)

The data in the endpoint FIFO must be organized as shown in Table 91. An example of endpoint FIFO access is given Table 92.

Table 91: Endpoint FIFO organization

Word number	Description
0 (lower byte)	packet length (lower byte)
0 (upper byte)	packet length (upper byte)
1 (lower byte)	data byte 1
1 (upper byte)	data byte 2
:	:
M = (N + 1)/2	data byte N

Table 92: Example of endpoint FIFO access

A0	Phase	Bus line	Word number	Description
1	command	D[7:0]	-	command code (00H to 1FH)
		D[15:8]	-	ignored
0	data	D[15:0]	0	packet length
0	data	D[15:0]	1	data word 1 (data byte 2, data byte 1)
0	data	D[15:0]	2	data word 2 (data byte 4, data byte 3)
:	:	:	:	:

Remark: There is no protection against writing or reading past a buffer's boundary or against writing into an OUT buffer or reading from an IN buffer. Any of these actions could cause an incorrect operation. Data residing in an OUT buffer are only meaningful after a successful transaction. Exception: during DMA access of a double-buffered endpoint, the buffer pointer automatically points to the secondary buffer after reaching the end of the primary buffer.

13.2.2 DcEndpointStatus Register (R: 50H–5FH)

This command is used to read the status of an endpoint FIFO. The command accesses the DcEndpointStatus register, the bit allocation of which is shown in Table 93. Reading the DcEndpointStatus register will clear the interrupt bit set for the corresponding endpoint in the DcInterrupt register (see Table 109).

All bits of the DcEndpointStatus register are read-only. Bit EPSTAL is controlled by the Stall/Unstall commands and by the reception of a SETUP token (see Section 13.2.3).

Code (Hex): 50 to 5F — read (control OUT, control IN, endpoint 1 to 14)

Transaction — read 1 word

Table 93: DcEndpointStatus register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	EPSTAL	EPFULL1	EPFULL0	DATA_PID	OVER WRITE	SETUPT	CPUBUF	reserved
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 94: DcEndpointStatus register: bit description

Bit	Symbol	Description
7	EPSTAL	This bit indicates whether the endpoint is stalled or not (1 = stalled, 0 = not stalled). Set to logic 1 by a Stall Endpoint command and cleared to logic 0 by an Unstall Endpoint command. The endpoint is automatically unstalled upon reception of a SETUP token.
6	EPFULL1	Logic 1 indicates that the secondary endpoint buffer is full.
5	EPFULL0	Logic 1 indicates that the primary endpoint buffer is full.
4	DATA_PID	This bit indicates the data PID of the next packet (0 = DATA PID, 1 = DATA1 PID).
3	OVERWRITE	This bit is set by hardware, logic 1 indicating that a new SETUP packet has overwritten the previous set-up information, before it was acknowledged or before the endpoint was stalled. This bit is cleared by reading, if writing the set-up data has finished. Firmware must check this bit before sending an Acknowledge SETUP command or stalling the endpoint. Upon reading logic 1 the firmware must stop ongoing set-up actions and wait for a new SETUP packet.
2	SETUPT	Logic 1 indicates that the buffer contains a SETUP packet.
1	CPUBUF	This bit indicates which buffer is currently selected for CPU access (0 = primary buffer, 1 = secondary buffer).
0	-	reserved

13.2.3 Stall Endpoint/Unstall Endpoint (40H–4FH/80H—8FH)

These commands are used to stall or unstall an endpoint. The commands modify the content of the DcEndpointStatus register (see Table 93).

A stalled control endpoint is automatically unstalled when it receives a SETUP token, regardless of the packet content. If the endpoint should stay in its stalled state, the microprocessor can re-stall it with the Stall Endpoint command.

When a stalled endpoint is unstalled (either by the Unstall Endpoint command or by receiving a SETUP token), it is also re-initialized. This flushes the buffer: if it is an OUT buffer it waits for a DATA 0 PID, if it is an IN buffer it writes a DATA 0 PID.

Code (Hex): 40 to 4F — stall (control OUT, control IN, endpoint 1 to 14)

Code (Hex): 80 to 8F — unstall (control OUT, control IN, endpoint 1 to 14)

Transaction — none

13.2.4 Validate Endpoint Buffer (R/W: 6FH/61H)

This command signals the presence of valid data for transmission to the USB host, by setting the Buffer Full flag of the selected IN endpoint. This indicates that the data in the buffer is valid and can be sent to the host, when the next IN token is received. For a double-buffered endpoint this command switches the current FIFO for CPU access.

Remark: For special aspects of the control IN endpoint see Section 11.3.6.

Code (Hex): 61 to 6F — validate endpoint buffer (control IN, endpoint 1 to 14)

Transaction — none

13.2.5 Clear Endpoint Buffer (70H, 72H–7FH)

This command unlocks and clears the buffer of the selected OUT endpoint, allowing the reception of new packets. Reception of a complete packet causes the Buffer Full flag of an OUT endpoint to be set. Any subsequent packets are refused by returning a NAK condition, until the buffer is unlocked using this command. For a double-buffered endpoint this command switches the current FIFO for CPU access.

Remark: For special aspects of the control OUT endpoint see Section 11.3.6.

Code (Hex): 70, 72 to 7F — clear endpoint buffer (control OUT, endpoint 1 to 14)

Transaction — none

13.2.6 DcEndpointStatusImage Register (D0H–DFH)

This command is used to check the status of the selected endpoint FIFO without clearing any status or interrupt bits. The command accesses the DcEndpointStatusImage register, which contains a copy of the DcEndpointStatus register. The bit allocation of the DcEndpointStatusImage register is shown in Table 95.

Code (Hex): D0 to DF — check status (control OUT, control IN, endpoint 1 to 14)

Transaction — write/read 1 word

Table 95: DcEndpointStatusImage register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	EPSTAL	EPFULL1	EPFULL0	DATA_PID	OVER WRITE	SETUPT	CPUBUF	reserved
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 96: DcEndpointStatusImage register: bit description

Bit	Symbol	Description
7	EPSTAL	This bit indicates whether the endpoint is stalled or not (1 = stalled, 0 = not stalled).
6	EPFULL1	Logic 1 indicates that the secondary endpoint buffer is full.
5	EPFULL0	Logic 1 indicates that the primary endpoint buffer is full.
4	DATA_PID	This bit indicates the data PID of the next packet (0 = DATA PID, 1 = DATA1 PID).
3	OVERWRITE	This bit is set by hardware, logic 1 indicating that a new SETUP packet has overwritten the previous set-up information, before it was acknowledged or before the endpoint was stalled. This bit is cleared by reading, if writing the set-up data has finished. Firmware must check this bit before sending an Acknowledge SETUP command or stalling the endpoint. Upon reading logic 1 the firmware must stop ongoing set-up actions and wait for a new SETUP packet.
2	SETUPT	Logic 1 indicates that the buffer contains a SETUP packet.
1	CPUBUF	This bit indicates which buffer is currently selected for CPU access (0 = primary buffer, 1 = secondary buffer).
0	-	reserved

13.2.7 Acknowledge SETUP (F4H)

This command acknowledges to the host that a SETUP packet was received. The arrival of a SETUP packet disables the Validate Buffer and Clear Buffer commands for the control IN and OUT endpoints. The microprocessor needs to re-enable these commands by sending an Acknowledge SETUP command, see Section 11.3.6.

Code (Hex): F4 — acknowledge SETUP

Transaction — none

13.3 General commands

13.3.1 Read Endpoint Error Code (R: A0H–AFH)

This command returns the status of the last transaction of the selected endpoint, as stored in the DcErrorCode register. Each new transaction overwrites the previous status information. The bit allocation of the DcErrorCode register is shown in Table 97.

Code (Hex): A0 to AF — read error code (control OUT, control IN, endpoint 1 to 14)

Transaction — read 1 word

Table 97: DcErrorCode register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	UNREAD	DATA01	reserved	ERROR[3:0]				RTOK
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 98: DcErrorCode register: bit description

Bit	Symbol	Description
7	UNREAD	Logic 1 indicates that a new event occurred before the previous status was read.
6	DATA01	This bit indicates the PID type of the last successfully received or transmitted packet (0 = DATA0 PID, 1 = DATA1 PID).
5	-	reserved
4 to 1	ERROR[3:0]	Error code. For error description, see Table 99.
0	RTOK	Logic 1 indicates that data was received or transmitted successfully.

Table 99: Transaction error codes

Error code (Binary)	Description
0000	no error
0001	PID encoding error; bits 7 to 4 are not the inverse of bits 3 to 0
0010	PID unknown; encoding is valid, but PID does not exist
0011	unexpected packet; packet is not of the expected type (token, data, or acknowledge), or is a SETUP token to a non-control endpoint
0100	token CRC error
0101	data CRC error

Table 99: Transaction error codes…*continued*

Error code (Binary)	Description
0110	time-out error
0111	babble error
1000	unexpected end-of-packet
1001	sent or received NAK (Not AcKnowledge)
1010	sent Stall; a token was received, but the endpoint was stalled
1011	overflow; the received packet was larger than the available buffer space
1100	sent empty packet (ISO only)
1101	bit stuffing error
1110	sync error
1111	wrong (unexpected) toggle bit in DATA PID; data was ignored

13.3.2 Unlock device (B0H)

This command unlocks ISP1161's DC from write-protection mode after a 'resume'. In 'suspend' state all registers and FIFOs are write-protected to prevent data corruption by external devices during a 'resume'. Also, the register access for reading is possible only after the 'Unlock Device' command is executed.

After waking up from 'suspend' state, the firmware must unlock the registers and FIFOs via this command, by writing the unlock code (AA37H) into the DcLock register. The bit allocation of the DcLock register is given in Table 100.

Code (Hex): B0 — unlock the device

Transaction — write 1 word (unlock code)

Table 100: DcLock register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	UNLOCKH[7:0] = AAH							
Reset	1	0	1	0	1	0	1	0
Access	W	W	W	W	W	W	W	W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	UNLOCKL[7:0] = 37H							
Reset	0	0	1	1	0	1	1	1
Access	W	W	W	W	W	W	W	W

Table 101: DcLock register: bit description

Bit	Symbol	Description
15 to 0	UNLOCK[15:0]	Sending data AA37H unlocks the internal registers and FIFOs for writing, following a 'resume'.

13.3.3 DcScratch Register (R/W: B3H/B2H)

This command accesses the 16-bit DcScratch register, which can be used by the firmware to save and restore information, e.g., the device status before powering down in 'suspend' state.

The register bit allocation is given in Table 102.

Code (Hex): B2/B3 — write/read DcScratch register

Transaction — write/read 1 word

Table 102: DcScratch Information register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved			SFIR[12:8]				
Reset	0	0	0	0	0	0	0	0
Access		R/W		R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	SFIR[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 103: DcScratch Information register: bit description

Bit	Symbol	Description
15 to 13	-	reserved; must be logic 0
12 to 0	SFIR[12:0]	Scratch Information register

13.3.4 Read Frame Number (R: B4H)

This command returns the frame number of the last successfully received SOF. It is followed by reading one word from the DcFrameNumber register, containing the frame number. The DcFrameNumber register is shown in Table 104.

Remark: After a bus reset, the value of the DcFrameNumber register is undefined.

Code (Hex): B4 — read frame number

Transaction — read 1 word

Table 104: DcFrameNumber register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved					SOFRH[2:0]		
Reset[1]	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	SOFRL[7:0]							
Reset[1]	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

[1] Reset value undefined after a bus reset.

Table 105: DcFrameNumber register: bits description

Bit	Symbol	Description
15 to 11	-	reserved
10 to 8	SOFRH[2:0]	SOF frame number (upper byte)
7 to 0	SOFRL[7:0]	SOF frame number (lower byte)

Table 106: Example of DcFrameNumber register access

A0	Phase	Bus line	Word number	Description
1	command	D[7:0]	-	command code (B4H)
		D[15:8]	-	ignored
0	data	D[15:0]	0	frame number

13.3.5 Read Chip ID (R: B5H)

This command reads the chip identification code and hardware version number. The firmware must check this information to determine the supported functions and features. This command accesses the DcChipID register, which is shown in Table 107.

Code (Hex): B5 — read chip ID

Transaction — read 1 word

Table 107: DcChipID register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	CHIPIDH[7:0]							
Reset	0	1	1	0	0	0	0	1
Access	R	R	R	R	R	R	R	R
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	CHIPIDL[7:0]							
Reset	0	0	1	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 108: DcChipID register: bit description

Bit	Symbol	Description
15 to 8	CHIPIDH[7:0]	chip ID code
7 to 0	CHIPIDL[7:0]	silicon version

13.3.6 Read DcInterrupt Register (R: C0H)

This command indicates the sources of interrupts as stored in the 4-byte DcInterrupt register. Each individual endpoint has its own interrupt bit. The bit allocation of the DcInterrupt register is shown in Table 109. Bit BUSTATUS is used to verify the current bus status in the interrupt service routine. Interrupts are enabled via the DcInterruptEnable register, see Section 13.1.5.

While reading the DcInterrupt register, read both 2 bytes completely.

Code (Hex): C0 — read DcInterrupt register

Transaction — read 2 words

Table 109: DcInterrupt register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Bit	23	22	21	20	19	18	17	16
Symbol	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	EP6	EP5	EP4	EP3	EP2	EP1	EP0IN	EP0OUT
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	BUSTATUS	reserved	PSOF	SOF	EOT	SUSPND	RESUME	RESET
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 110: DcInterrupt register: bit description

Bit	Symbol	Description
31 to 24	-	reserved
23 to 10	EP14 to EP1	Logic 1 indicates the interrupt source(s): endpoint 14 to 1.
9	EP0IN	Logic 1 indicates the interrupt source: control IN endpoint.
8	EP0OUT	Logic 1 indicates the interrupt source: control OUT endpoint.
7	BUSTATUS	Monitors the current USB bus status (0 = awake, 1 = suspend).
6	-	reserved
5	PSOF	Logic 1 indicates that an interrupt is issued every 1 ms because of the Pseudo SOF; after 3 missed SOFs ‘suspend’ state is entered.
4	SOF	Logic 1 indicates that a SOF condition was detected.
3	EOT	Logic 1 indicates that an internal EOT condition was generated by the DMA Counter reaching zero.
2	SUSPND	Logic 1 indicates that an ‘awake’ to ‘suspend’ change of state was detected on the USB bus.
1	RESUME	Logic 1 indicates that a ‘resume’ state was detected.
0	RESET	Logic 1 indicates that a bus reset condition was detected.

14. Power supply

ISP1161 can operate at either 5 V or 3.3 V.

When using 5 V as ISP1161's power supply input: only V_{CC} (pin 56) can be connected to the 5 V power supply. An application with a 5 V power supply input is shown in Figure 44. ISP1161 has an internal DC/DC regulator to provide 3.3 V for its internal core. This internal 3.3 V can also be obtained from pin $V_{reg(3.3)}$ to supply the 1.5 kΩ pull-up resistor of the DC side upstream port signal D_DP. The signal D_DP is connected to the standard USB upstream data line D+.

When using 3.3 V as the power supply input, the internal DC/DC regulator will be bypassed. All four power supply pins (V_{CC}, $V_{reg(3.3)}$, V_{hold1} and V_{hold2}) can be used as power supply input.

It is recommended that you connect all four power supply pins to the 3.3 V power supply, as shown in Figure 45. If, however, you have board space (routing area) constraints, you must connect at least the V_{CC} and the $V_{reg(3.3)}$ to the 3.3 V power supply.

For both 3.3 V and 5 V operation, all four power supply pins should be connected to a decoupling capacitor.

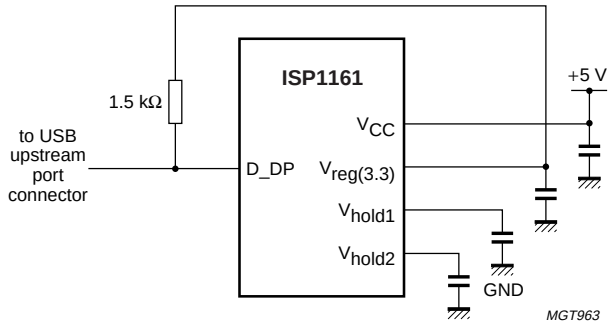


Fig 44. Using a 5 V supply.

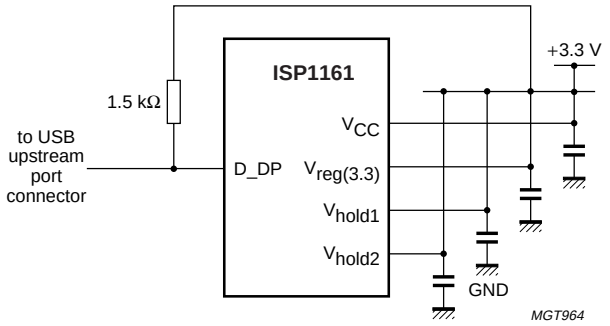


Fig 45. Using a 3.3 V supply.

15. Crystal oscillator and LazyClock

The ISP1161 has a crystal oscillator designed for a 6 MHz parallel-resonant crystal (fundamental). A typical circuit is shown in Figure 46. Alternatively, an external clock signal of 6 MHz can be applied to input XTAL1, while leaving output XTAL2 open. See Figure 47.

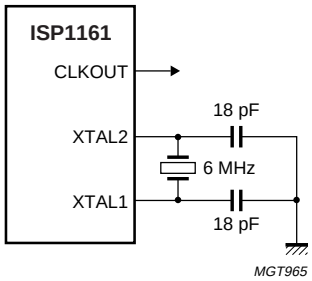


Fig 46. Oscillator circuit with external crystal.

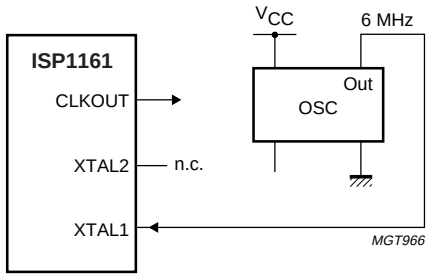


Fig 47. Oscillator circuit using external oscillator.

The 6 MHz oscillator frequency is multiplied to 48 MHz by an internal PLL. This frequency is used to generate a programmable clock output signal at pin CLKOUT, ranging from 3 to 48 MHz.

In ‘suspend’ state the normal CLKOUT signal is not available, because the crystal oscillator and the PLL are switched off to save power. Instead, the CLKOUT signal can be switched to the LazyClock frequency of 100 kHz ± 50%.

The oscillator operation and the CLKOUT frequency are controlled via the DcHardwareConfiguration register, as shown in Figure 48. The following bits are involved:

- CLKRUN switches the oscillator on and off
- CLKDIV[3:0] is the division factor determining the normal CLKOUT frequency
- NOLAZY controls the LazyClock signal output during ‘suspend’ state.

For details about the DC’s interrupt logic, see Section 8.6.3.

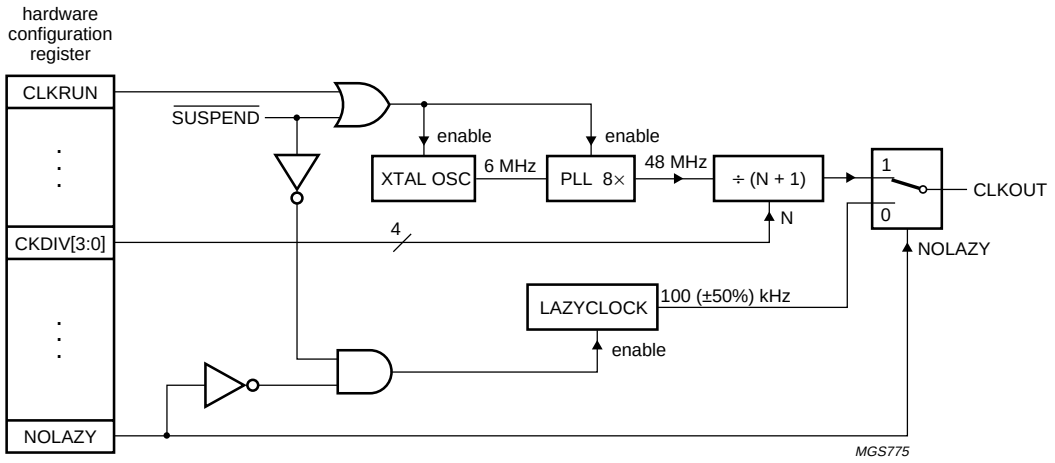
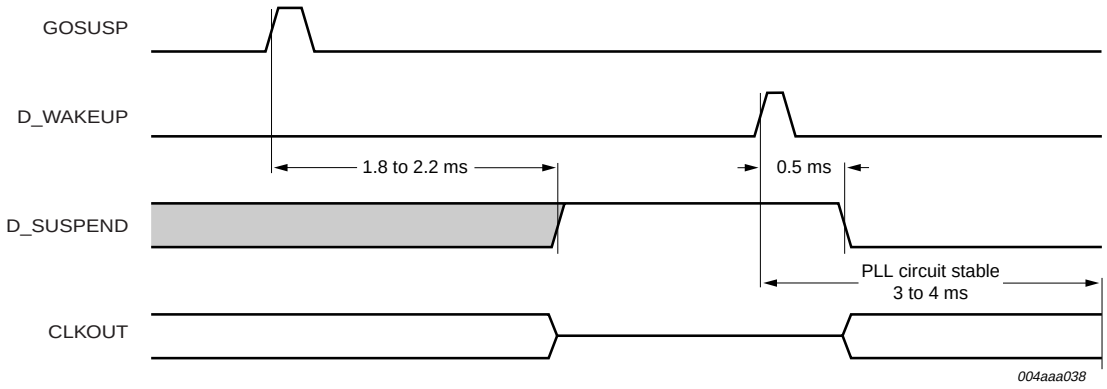


Fig 48. Oscillator and LazyClock logic.

When ISP1161's DC enters 'suspend' state (by setting and clearing bit GOSUSP in the DcMode register), outputs D_SUSPEND and CLKOUT change state after approximately 2 ms delay. When NOLAZY = 0 the clock signal on output CLKOUT does not stop, but changes to the 100 kHz ± 50% LazyClock frequency.

When resuming from 'suspend' state by a positive pulse on input D_WAKEUP, output SUSPEND is cleared and the clock signal on CLKOUT restarted after a 0.5 ms delay. The timing of the CLKOUT signal at 'suspend' and 'resume' is given in Figure 49.



If enabled, the 100 kHz± 50% LazyClock frequency will be output on pin CLKOUT during 'suspend' state.

Fig 49. CLKOUT signal timing at 'suspend' and 'resume' for device controller.

16. Limiting values

Table 111: Absolute maximum ratings

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions		Min	Max	Unit
$V_{CC(5V)}$	supply voltage on pin V_{CC}			–0.5	+6.0	V
$V_{CC(3.3V)}$	supply voltage on pin $V_{reg(3.3)}$			–0.5	+4.6	V
V_I	input voltage			–0.5	+6.0	V
I_{lu}	latch-up current	$V_I < 0$ or $V_I > V_{CC}$		-	100	mA
V_{esd}	electrostatic discharge voltage	$I_{LI} < 1$ µA	[1]	–2000	+2000	V
T_{stg}	storage temperature			–60	+150	°C

[1] Equivalent to discharging a 100 pF capacitor via a 1.5 kΩ resistor (Human Body Model).

17. Recommended operating conditions

Table 112: Recommended operating conditions

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{CC}	supply voltage	with internal regulator		4.0	5.0	5.5	V
		internal regulator bypass		3.0	3.3	3.6	V
V_I	input voltage		[1]	0	V_{CC}	5.5	V
$V_{I(AI/O)}$	input voltage on AI/O pin type (D+ and D– lines			0	-	3.6	V
$V_{O(od)}$	open-drain output pull-up voltage			0	-	V_{CC}	V
T_{amb}	ambient temperature			–40	-	+85	°C

[1] 5 V tolerant inputs.

18. Static characteristics

Table 113: Static characteristics; supply pins

V_{CC} = 3.0 to 3.6 V or 4.0 to 5.5 V; V_{GND} = 0 V; T_{amb} = −40 to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{CC} = 5 V							
$V_{reg(3.3)}$	internal regulator output		[1]	3.0	3.3	3.6	V
I_{CC}	operating supply current			-	47	-	mA
$I_{CC(susp)}$	suspend supply current			-	40	500	µA
$I_{CC(HC)}$	operating supply current for HC	DC is suspended		-	22	-	mA
$I_{CC(DC)}$	operating supply current for DC	HC is suspended		-	18	-	mA
V_{CC} = 3.3 V							
I_{CC}	operating supply current			-	50	-	mA
$I_{CC(susp)}$	suspend supply current			-	150	500	µA
$I_{CC(HC)}$	operating supply current for HC	DC is suspended		-	22	-	mA
$I_{CC(DC)}$	operating supply current for DC	HC is suspended		-	18	-	mA

[1] In suspend state, the minimum voltage is 2.7 V.

Table 114: Static characteristics: digital pins

V_{CC} = 3.0 to 3.6 V or 4.0 to 5.5 V; V_{GND} = 0 V; T_{amb} = −40 to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
Input levels							
V_{IL}	LOW-level input voltage			-	-	0.8	V
V_{IH}	HIGH-level input voltage			2.0	-	-	V
Schmitt-trigger inputs							
$V_{th(LH)}$	positive-going threshold voltage			1.4	-	1.9	V
$V_{th(HL)}$	negative-going threshold voltage			0.9	-	1.5	V
V_{hys}	hysteresis voltage			0.4	-	0.7	V
Output levels							
V_{OL}	LOW-level output voltage	I_{OL} = 4 mA		-	-	0.4	V
		I_{OL} = 20 µA		-	-	0.1	V
V_{OH}	HIGH-level output voltage	I_{OH} = 4 mA	[1]	2.4	-	-	V
		I_{OH} = 20 µA		$V_{reg(3.3)}$ − 0.1	-	-	V
Leakage current							
I_{LI}	input leakage current		[2]	-	-	±5	µA
C_{IN}	pin capacitance	pin to GND		-	-	5	pF
Open-drain outputs							
I_{OZ}	OFF-state output current			-	-	±5	µA

[1] Not applicable for open-drain outputs.

[2] This value is applicable to transistor input only. The value will be different if internal pull-up or pull-down resistors are used.

Table 115: Static characteristics: AI/O pins (D+ and D– lines)

V_{CC} = 3.0 to 3.6 V or 4.0 to 5.5 V; V_{GND} = 0 V; T_{amb} = –40 to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
Input levels							
V_{DI}	differential input sensitivity	$\|V_{I(D+)} - V_{I(D-)}\|$	[1]	0.2	-	-	V
V_{CM}	differential common mode voltage	includes V_{DI} range		0.8	-	2.5	V
V_{IL}	LOW-level input voltage			-	-	0.8	V
V_{IH}	HIGH-level input voltage			2.0	-	-	V
Output levels							
V_{OL}	LOW-level output voltage	R_L = 1.5 kΩ to +3.6 V		-	-	0.3	V
V_{OH}	HIGH-level output voltage	R_L = 15 kΩ to GND		2.8	-	3.6	V
Leakage current							
I_{LZ}	OFF-state leakage current			-	-	±10	µA
Capacitance							
C_{IN}	transceiver capacitance	pin to GND		-	-	10	pF
Resistance							
R_{PD}	pull-down resistance on pins DP and DM	enable internal resistors		10	-	20	kΩ
R_{PU}	pull-up resistance on pin D_DP	SoftConnect on		1	-	2	kΩ
Z_{DRV}	driver output impedance	steady-state drive	[2]	29	-	44	Ω
Z_{INP}	input impedance			10	-	-	MΩ
Termination							
V_{TERM}	termination voltage for upstream port pull-up (R_{PU})		[3]	3.0	-	3.6	V

[1] D+ is the USB positive data line and D– is the USB negative data line.

[2] Includes external resistors of 18 Ω ±1% on both H_D+ and H_D– lines.

[3] In suspend state, the minimum voltage is 2.7 V.

19. Dynamic characteristics

Table 116: Dynamic characteristics

V_{CC} = 3.0 to 3.6 V or 4.0 to 5.5 V; V_{GND} = 0 V; T_{amb} = −40 to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
Reset							
$t_{W(\overline{RESET})}$	pulse width on input $\overline{RESET}$	crystal oscillator running		160	-	-	µs
		crystal oscillator stopped	[1]	-		-	ms
Crystal oscillator							
f_{XTAL}	crystal frequency			-	6	-	MHz
R_S	series resistance			-	-	100	Ω
C_{LOAD}	load capacitance			-	18	-	pF
External clock input							
t_J	external clock jitter			-	-	500	ps
t_{DUTY}	clock duty cycle			45	50	55	%
t_{CR}, t_{CF}	rise time and fall time			-	-	3	ns

[1] Dependent on the crystal oscillator start-up time.

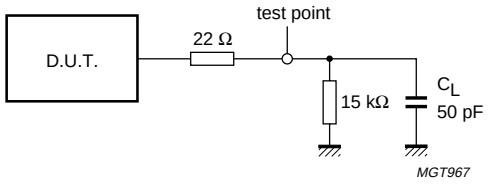
Table 117: Dynamic characteristics: AI/O pins (D+, D− lines

V_{CC} = 3.0 to 3.6 V or 4.0 to 5.5 V; V_{GND} = 0 V; T_{amb} = −40 to +85 °C; C_L = 50 pF; R_{PU} = 1.5 kΩ ±5% on D+ to V_{TERM}; measured with test circuit of Figure 50; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
Driver characteristics							
t_{FR}	rise time	C_L = 50 pF; 10% to 90% of $\|V_{OH} - V_{OL}\|$		4	-	20	ns
t_{FF}	fall time	C_L = 50 pF; 90% to 10% of $\|V_{OH} - V_{OL}\|$		4	-	20	ns
FRFM	differential rise and fall time matching (t_{FR}/t_{FF})		[1]	90	-	111.11	%
V_{CRS}	output signal crossover voltage		[1][2]	1.3	-	2.0	V

[1] Excluding the first transition from idle state.

[2] Characterized only, not tested. Limits guaranteed by design.



Load capacitance: C_L = 50 pF (full-speed mode).

Speed: full-speed mode only: internal 1.5 kΩ pull-up resistor on pin D_DP.

Fig 50. Load impedance for D_DP and D_DM pins.

19.1 Programmed I/O timing

- If you are accessing only the HC, then the HC Programmed I/O timing applies.
- If you are accessing only the DC, then the DC Programmed I/O timing applies.
- If you are accessing both the HC and the DC, then the DC Programmed I/O timing applies.

19.1.1 HC programmed I/O timing

Table 118: Dynamic characteristics: HC Programmed interface timing; see Figure 51

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{AS}	address set-up time before $\overline{WR}$ HIGH		5	-	-	ns
t_{AH}	address hold time after $\overline{WR}$ HIGH		8	-	-	ns
Read timing						
t_{SHSL}	first $\overline{RD}/\overline{WR}$ after CMD		300	-	-	ns
t_{SLRL}	$\overline{CS}$ LOW to $\overline{RD}$ LOW		0	-	-	ns
t_{RHSH}	$\overline{RD}$ HIGH to $\overline{CS}$ HIGH		0	-	-	ns
t_{RLRH}	$\overline{RD}$ LOW pulse width		33	-	-	ns
t_{RHRL}	$\overline{RD}$ HIGH to next $\overline{RD}$ LOW		110	-	-	ns
T_{RC}	$\overline{RD}$ cycle		143	-	-	ns
t_{RHDZ}	$\overline{RD}$ data hold time		3	-	22	ns
t_{RLDV}	$\overline{RD}$ LOW to data valid		-	-	32	ns
Write timing						
t_{WL}	$\overline{WR}$ LOW pulse width		26	-	-	ns
t_{WHWL}	$\overline{WR}$ HIGH to next $\overline{WR}$ LOW		110	-	-	ns
T_{WC}	$\overline{WR}$ cycle		136	-	-	ns
t_{SLWL}	$\overline{CS}$ LOW to $\overline{WR}$ LOW		0	-	-	ns
t_{WHSH}	$\overline{WR}$ HIGH to $\overline{CS}$ HIGH		0	-	-	ns
t_{WDSU}	$\overline{WR}$ data set-up time		5	-	-	ns
t_{WDH}	$\overline{WR}$ data hold time		8	-	-	ns

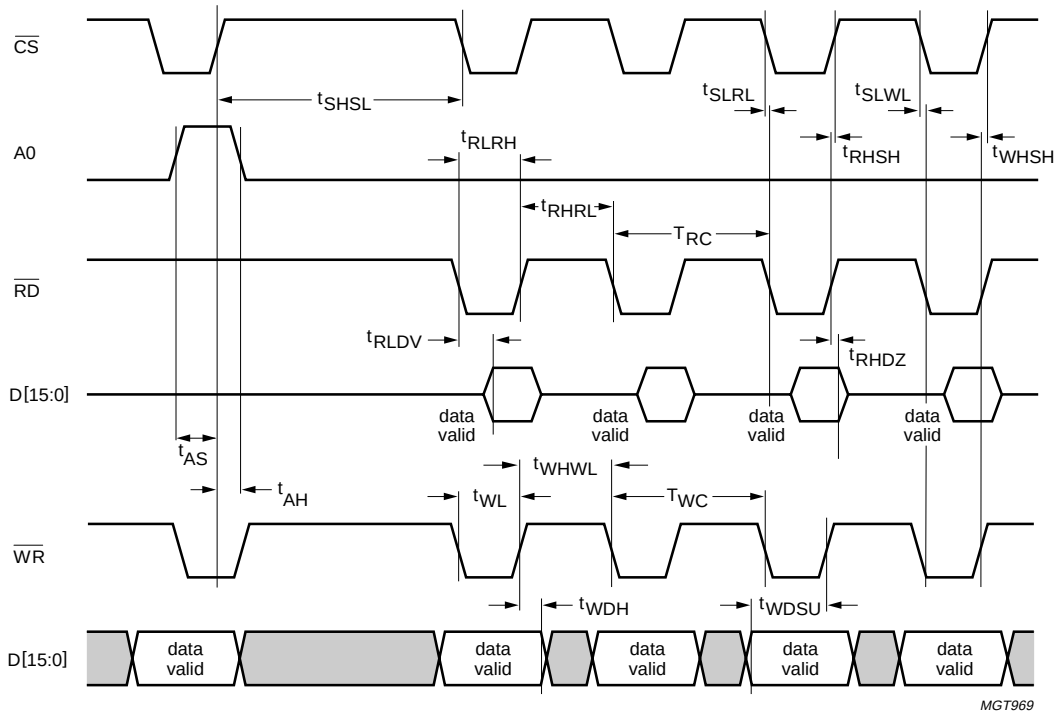
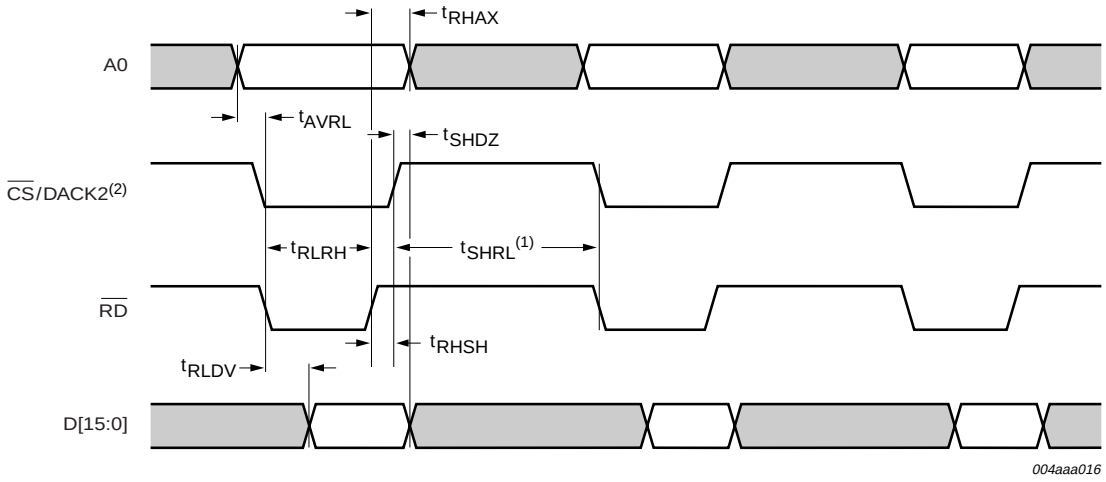


Fig 51. HC Programmed interface timing.

19.1.2 DC programmed I/O timing

Table 119: Dynamic characteristics: DC Programmed interface timing

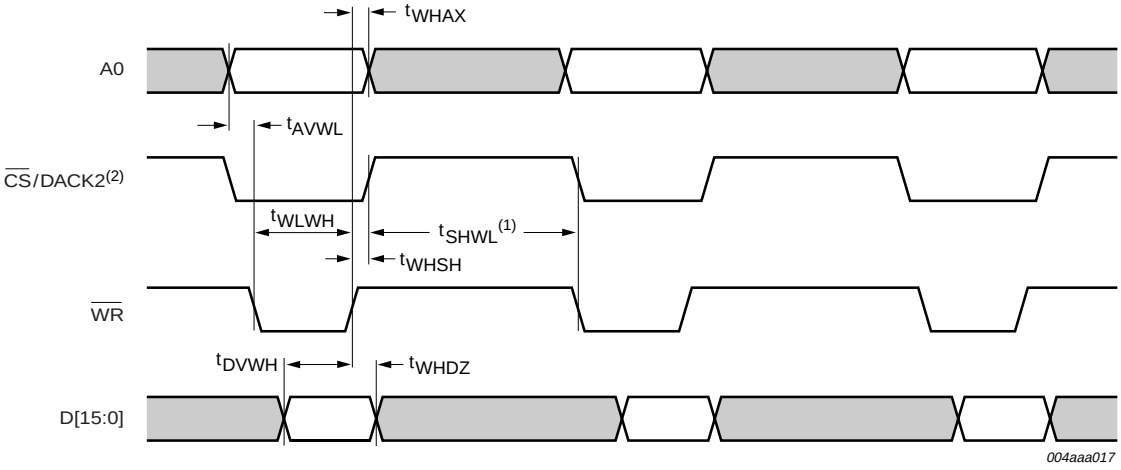
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Read timing (see Figure 52)						
t_{RHAX}	address hold time after $\overline{RD}$ HIGH		3	-	-	ns
t_{AVRL}	address set-up time before $\overline{RD}$ LOW		0	-	-	ns
t_{SHDZ}	data outputs high-impedance time after $\overline{CS}$ HIGH		-	-	3	ns
t_{RHSH}	chip deselect after $\overline{RD}$ HIGH		0	-	-	ns
t_{RLRH}	$\overline{RD}$ pulse width		25	-	-	ns
t_{RLDV}	data valid time after $\overline{RD}$ LOW		-	-	22	ns
$t_{SHRL} + t_{RLRH}$	read cycle time		180	-	-	ns
Write timing (see Figure 53)						
t_{WHAX}	address hold time after $\overline{WR}$ HIGH		3	-	-	ns
t_{AVWL}	address set-up time before $\overline{WR}$ LOW		0	-	-	ns
$t_{SHWL} + t_{WLWH}$	write cycle time		180	-	-	ns
t_{WLWH}	$\overline{WR}$ pulse width		22	-	-	ns
t_{WHSH}	chip deselect time after $\overline{WR}$ HIGH		0	-	-	ns
t_{DVWH}	data set-up time before $\overline{WR}$ HIGH		5	-	-	ns
t_{WHDZ}	data hold time after $\overline{WR}$ HIGH		3	-	-	ns



(1) For t_{SHRL} both $\overline{CS}$ and $\overline{RD}$ must be de-asserted.

(2) Programmable polarity: shown as active LOW.

Fig 52. DC Programmed interface read timing (I/O and 8237 compatible DMA).



(1) For t_{SHWL} both $\overline{CS}$ and $\overline{WR}$ must be de-asserted.

(2) Programmable polarity: shown as active LOW.

Fig 53. DC Programmed interface write timing (I/O and 8237 compatible DMA).

19.2 DMA timing

19.2.1 HC single-cycle DMA timing

Table 120: Dynamic characteristics: HC single-cycle DMA timing

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Read/write timing (see Figure 54)						
t_{RLRH}	$\overline{RD}$ pulse width		33	-	-	ns
t_{RLDV}	read process data set-up time		26	-	-	ns
t_{RHDZ}	read process data hold time		0	-	20	ns
t_{WSU}	write process data set-up time		5	-	-	ns
t_{WHD}	write process data hold time		0	-	-	ns
t_{AHRH}	DACK1 HIGH to DREQ1 HIGH		72	-	-	ns
t_{ALRL}	DACK1 LOW to DREQ1 LOW		-	-	21	ns
T_{DC}	DREQ1 cycle	[1]		-	-	ns
t_{SHAH}	$\overline{RD}/\overline{WR}$ HIGH to DACK1 HIGH		0	-	-	ns
t_{RHAL}	DREQ1 HIGH to DACK1 LOW		0	-	-	ns
t_{DS}	DREQ1 pulse spacing		146	-	-	ns

[1] $T_{DC} = t_{RHAL} + t_{DS} + t_{ALRL}$.

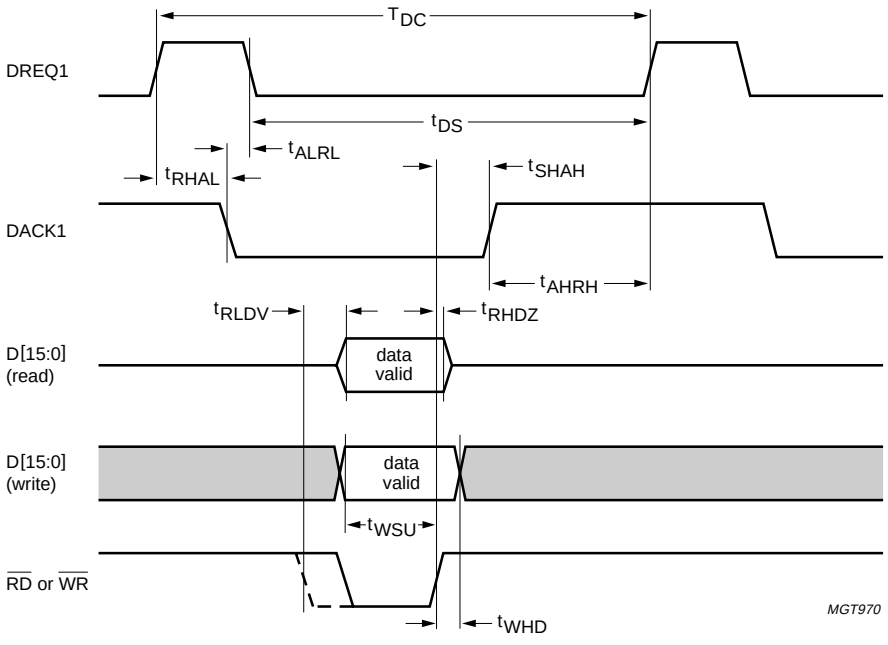


Fig 54. HC single-cycle DMA timing.

19.2.2 HC burst mode DMA timing

Table 121: Dynamic characteristics: HC burst mode DMA timing

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Read/write timing (for 4-cycle and 8-cycle burst mode); see Figure 55						
t_{RLRH}	$\overline{WR/RD}$ LOW pulse width		42	-	-	ns
t_{RHRL}	$\overline{WR/RD}$ HIGH to next $\overline{WR/RD}$ LOW		60	-	-	ns
T_{RC}	$\overline{WR/RD}$ cycle		102	-	-	ns
t_{SLRL}	$\overline{RD/WR}$ LOW to DREQ1 LOW		22	-	64	ns
t_{SHAH}	$\overline{RD/WR}$ HIGH to DACK1 HIGH		0	-	-	ns
t_{RHAL}	DREQ1 HIGH to DACK1 LOW		0	-	-	ns
T_{DC}	DREQ1 cycle		[1]	-	-	ns
$t_{DS(read)}$	DREQ1 pulse spacing (read)	4-cycle burst mode	105	-	-	ns
$t_{DS(read)}$	DREQ1 pulse spacing (read)	8-cycle burst mode	150	-	-	ns
$t_{DS(write)}$	DREQ1 pulse spacing (write)	4-cycle burst mode	72	-	-	ns
$t_{DS(write)}$	DREQ1 pulse spacing (write)	8-cycle burst mode	167	-	-	ns
t_{RLIS}	$\overline{RD/WR}$ LOW to EOT LOW		0	-	-	ns

[1] $T_{DC} = t_{SLAL} + (4 \text{ or } 8)t_{RC} + t_{DS}$.

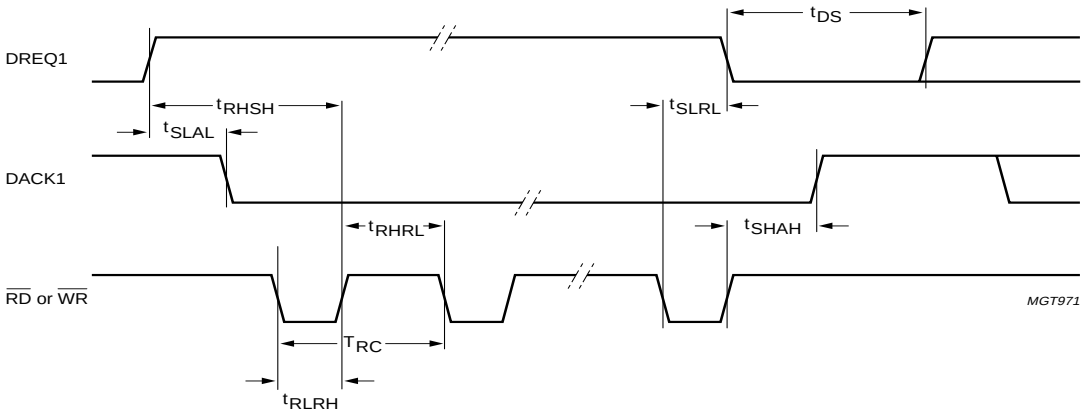


Fig 55. HC burst mode DMA timing.

19.2.3 External EOT timing for HC single-cycle DMA

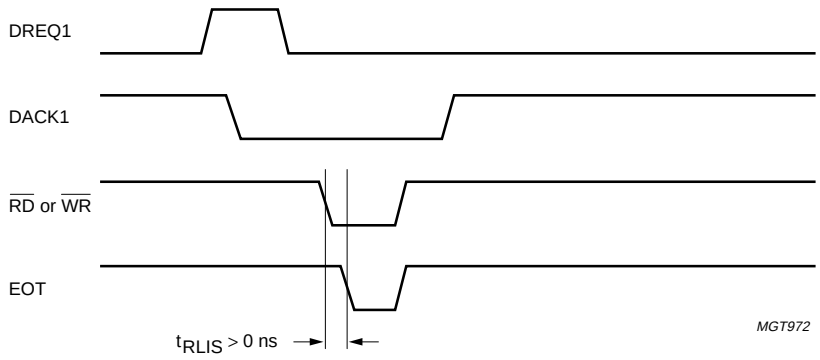


Fig 56. External EOT timing for HC single-cycle DMA.

19.2.4 External EOT timing for HC burst mode DMA

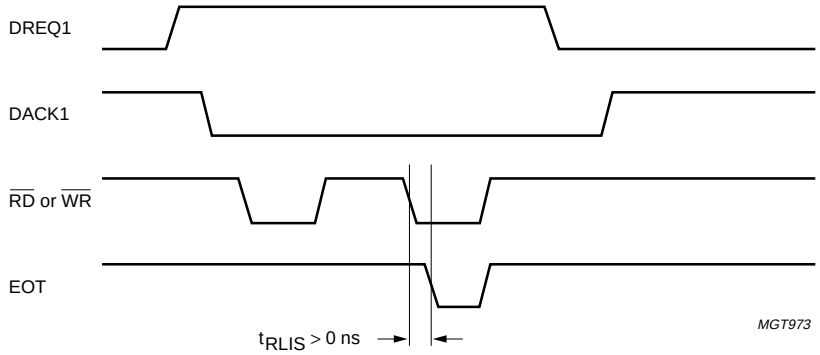
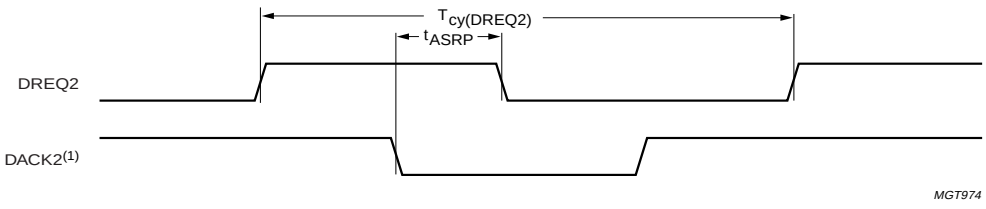


Fig 57. External EOT timing for HC burst mode DMA.

19.2.5 DC single-cycle DMA timing (8237 mode)

Table 122: Dynamic characteristics: DC single-cycle DMA timing (8237 mode)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{ASRP}	DREQ2 off after DACK2 on		-	-	40	ns
$T_{cy(DREQ2)}$	cycle time signal DREQ2		180	-	-	ns



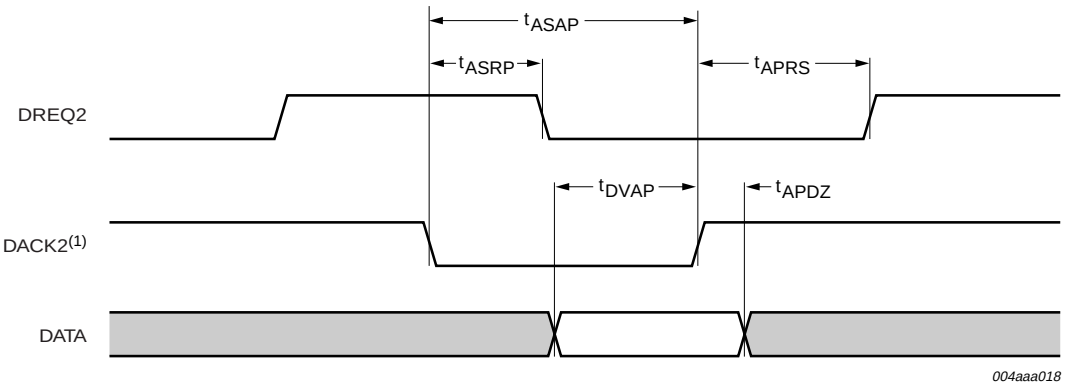
(1) Programmable polarity: shown as active LOW.

Fig 58. DC single-cycle DMA timing (8237 mode).

19.2.6 DC single-cycle DMA read timing in DACK-only mode

Table 123: Dynamic characteristics: DC single-cycle DMA read timing in DACK-only mode

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{ASRP}	DREQ off after DACK on		-	-	40	ns
t_{ASAP}	DACK pulse width		25	-	-	ns
$t_{ASAP} + t_{APRS}$	DREQ on after DACK off		180	-	-	ns
t_{ASDV}	data valid after DACK on		-	-	22	ns
t_{APDZ}	data hold after DACK off		-	-	3	ns



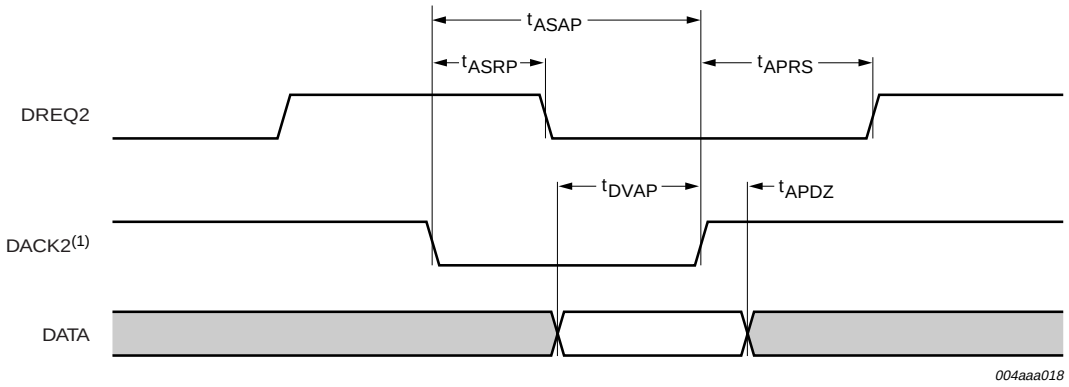
(1) Programmable polarity: shown as active LOW.

Fig 59. DC single-cycle DMA read timing in DACK-only mode.

19.2.7 DC single-cycle DMA write timing in DACK-only mode

Table 124: Dynamic characteristics: DC single-cycle DMA write timing in DACK-only mode

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{ASRP}	DREQ2 off after DACK2 on		-	-	40	ns
t_{ASAP}	DACK2 pulse width		25	-	-	ns
$t_{ASAP} + t_{APRS}$	DREQ2 on after DACK2 off		180	-	-	ns
t_{ASDV}	data valid after DACK2 on		-	-	22	ns
t_{APDZ}	data hold after DACK2 off		-	-	3	ns



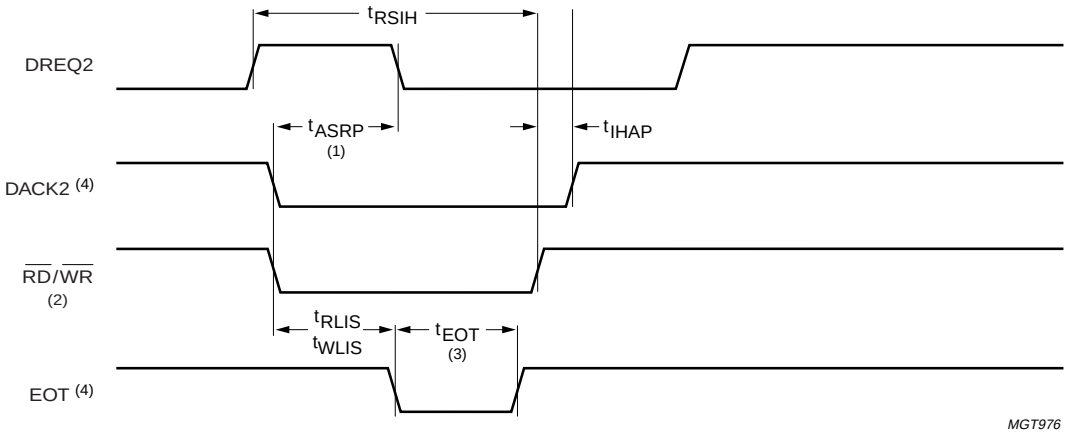
(1) Programmable polarity: shown as active LOW.

Fig 60. DC single-cycle DMA write timing in DACK-only mode.

19.2.8 EOT timing in DC single-cycle DMA

Table 125: Dynamic characteristics: EOT timing in DC single-cycle DMA

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{RSIH}	input $\overline{RD}/\overline{WR}$ HIGH after DREQ on		22	-	-	ns
t_{IHAP}	DACK off after input $\overline{RD}/\overline{WR}$ HIGH		0	-	-	ns
t_{EOT}	EOT pulse width	EOT on; DACK on; $\overline{RD}/\overline{WR}$ LOW	22	-	-	ns
t_{RLIS}	input EOT on after $\overline{RD}$ LOW		-	-	89	ns
t_{WLIS}	input EOT on after $\overline{WR}$ LOW		-		89	ns



(1) t_{ASRP} starts from DACK or $\overline{RD}/\overline{WR}$ going LOW, whichever occurs later.

(2) The $\overline{RD}/\overline{WR}$ signals are not used in DACK-only DMA mode.

(3) The EOT condition is considered valid if DACK, $\overline{RD}/\overline{WR}$ and $\overline{EOT}$ are all active (= LOW).

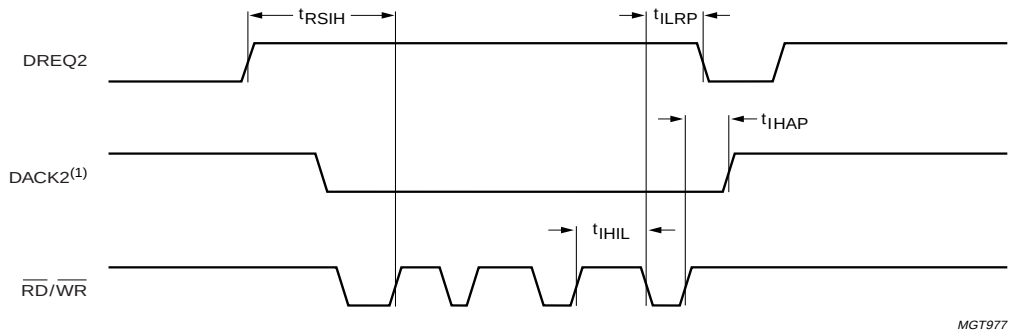
(4) Programmable polarity: shown as active LOW.

Fig 61. EOT timing in DC single-cycle DMA.

19.2.9 DC burst mode DMA timing

Table 126: Dynamic characteristics: DC burst mode DMA timing

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{RSIH}	input $\overline{RD}/\overline{WR}$ HIGH after DREQ on		22	-	-	ns
t_{ILRP}	DREQ off after input $\overline{RD}/\overline{WR}$ LOW		-	-	-	ns
t_{IHAP}	DACK off after input $\overline{RD}/\overline{WR}$ HIGH		0	-	60	ns
t_{IHIL}	DMA burst repeat interval (input $\overline{RD}/\overline{WR}$ HIGH to LOW)		180	-	-	ns



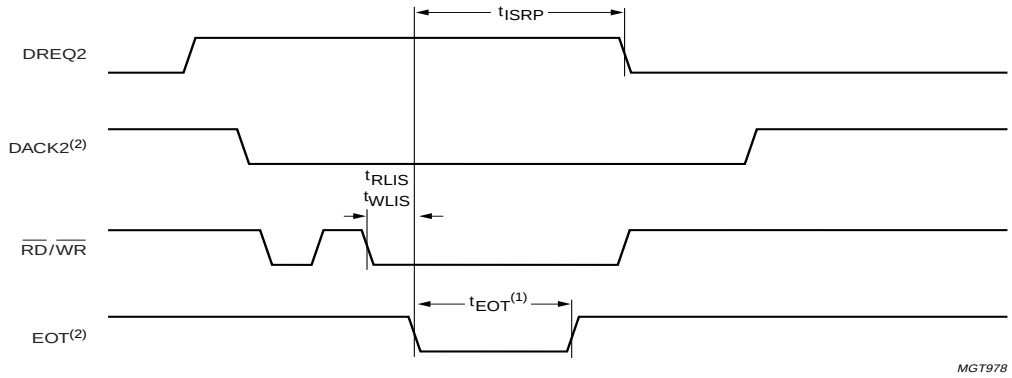
(1) Programmable polarity: shown as active LOW.

Fig 62. DC burst mode DMA timing.

19.2.10 EOT timing in DC burst mode DMA

Table 127: Dynamic characteristics: EOT timing in DC burst mode DMA

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{EOT}	EOT pulse width	EOT on; DACK on; $\overline{RD}/\overline{WR}$ LOW	22	-	-	ns
t_{ISRP}	DREQ off after input EOT on		-	-	40	ns
t_{RLIS}	input EOT on after $\overline{RD}$ LOW		-	-	89	ns
t_{WLIS}	input EOT on after $\overline{WR}$ LOW		-	-	89	ns



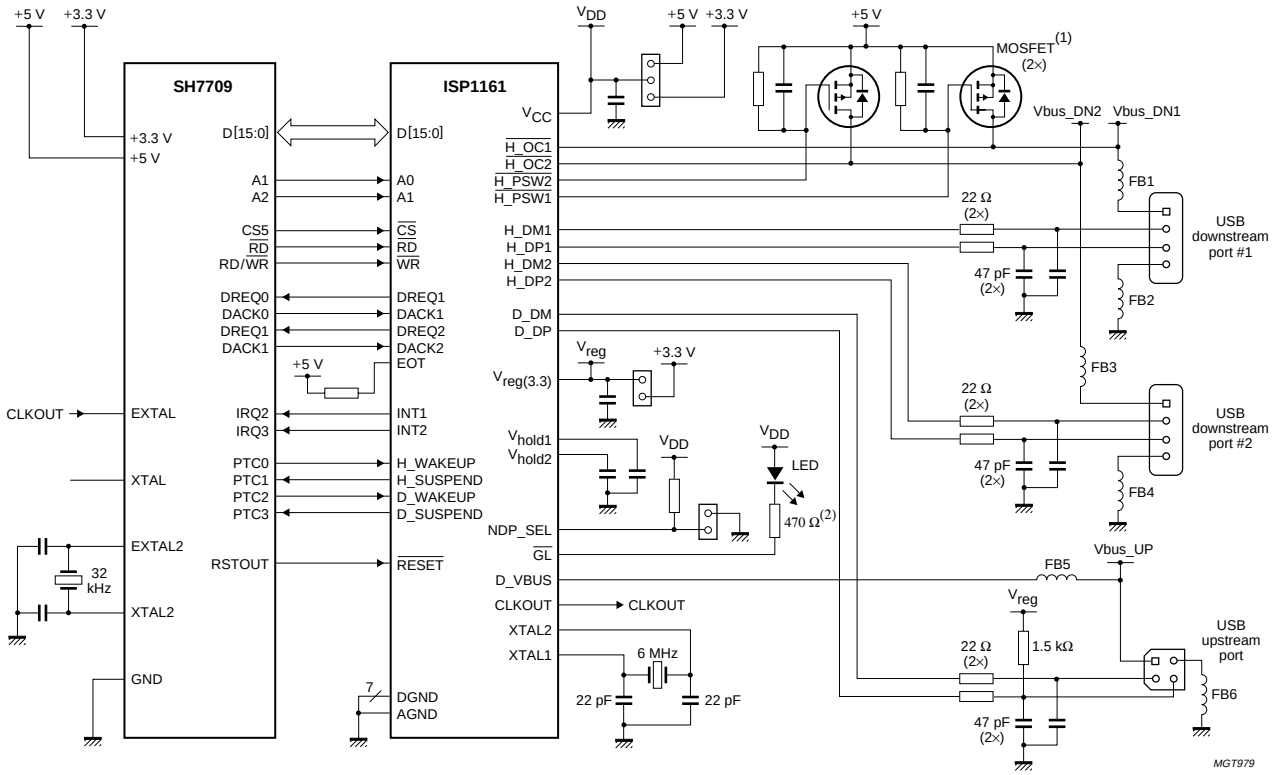
(1) The EOT condition is considered valid if DACK2, $\overline{RD}/\overline{WR}$ and $\overline{EOT}$ are all active (= LOW).

(2) Programmable polarity: shown as active LOW.

Fig 63. EOT timing in DC burst mode DMA.

20. Application information

20.1 Typical interface circuit



(1) For MOSFET, R_{DSon} = 150 mΩ.

(2) 470 Ω assuming that V_{CC} = 5 V.

Fig 64. Typical interface circuit to Hitachi SH-3 (SH7709) RISC processor.

20.2 Interfacing a ISP1161 with a SH7709 RISC processor

This section shows a typical interface circuit between ISP1161 and a RISC processor. The Hitachi SH-3 series RISC processor SH7709 is used as the example. The main ISP1161 signals to be taken into consideration for connecting to a SH7709 RISC processor are:

- A 16-bit data bus: D15 to D0 for ISP1161. ISP1161 is ‘little endian’ compatible.
- Two address lines A1 and A0 are needed for a complete addressing of the ISP1161 internal registers:
  - A1 = 0 and A0 = 0 will select the Data Port of the Host Controller
  - A1 = 0 and A0 = 1 will select the Command Port of the Host Controller
  - A1 = 1 and A0 = 0 will select the Data Port of the Device Controller
  - A1 = 1 and A0 = 1 will select the Command Port of the Device Controller
- The $\overline{CS}$ line is used for chip selection of ISP1161 in a certain address range of the RISC system. This signal is active LOW.
- $\overline{RD}$ and $\overline{WR}$ are common read and write signals. These signals are active LOW.
- There are two DMA channel standard control lines:
  - DREQ1 and DACK1
  - DREQ2 and DACK2

  In each case one channel is used by the host controller and the other channel is used by the device controller. These signals have programmable active levels.
- Two interrupt lines: INT1 (used by the host controller) and INT2 (used by the device controller). Both have programmable level/edge and polarity (active HIGH or LOW).
- The internal 15 kΩ pull-down resistors are used for the HCs two USB downstream ports.
- The $\overline{RESET}$ signal is active LOW.

Remark: SH7709’s system clock input is for reference only. Refer to SH7709’s specification for its actual use.

ISP1161 can work under either 3.3 V or 5 V power supply; however, its internal core actually works at 3.3 V. When using 3.3 V as the power supply input, the internal DC/DC regulator will be bypassed. It is best to connect all four power supply pins (V_{CC}, $V_{reg(3.3)}$, V_{hold1} and V_{hold2}) to the 3.3 V power supply (for more information see Section 14).

All of the ISP1161’s I/O pins are 5 V tolerant. This feature allows the ISP1161 the flexibility to be used in an embedded system under either a 3.3 V or a 5 V power supply.

A typical SH7709 interface circuit is shown in Figure 64.

20.3 Typical software model

This section shows a typical software requirement for an embedded system that incorporates ISP1161. The software model for a Digital Still Camera (DSC) is used as the example for illustration (as shown in Figure 65). Two components of system software are required to make full use of the features in ISP1161: the host stack and the device stack. The device stack provides API directly to the application task for device function; the host stack provides API for Class driver and device driver, both of which provide API for application tasks for host function.

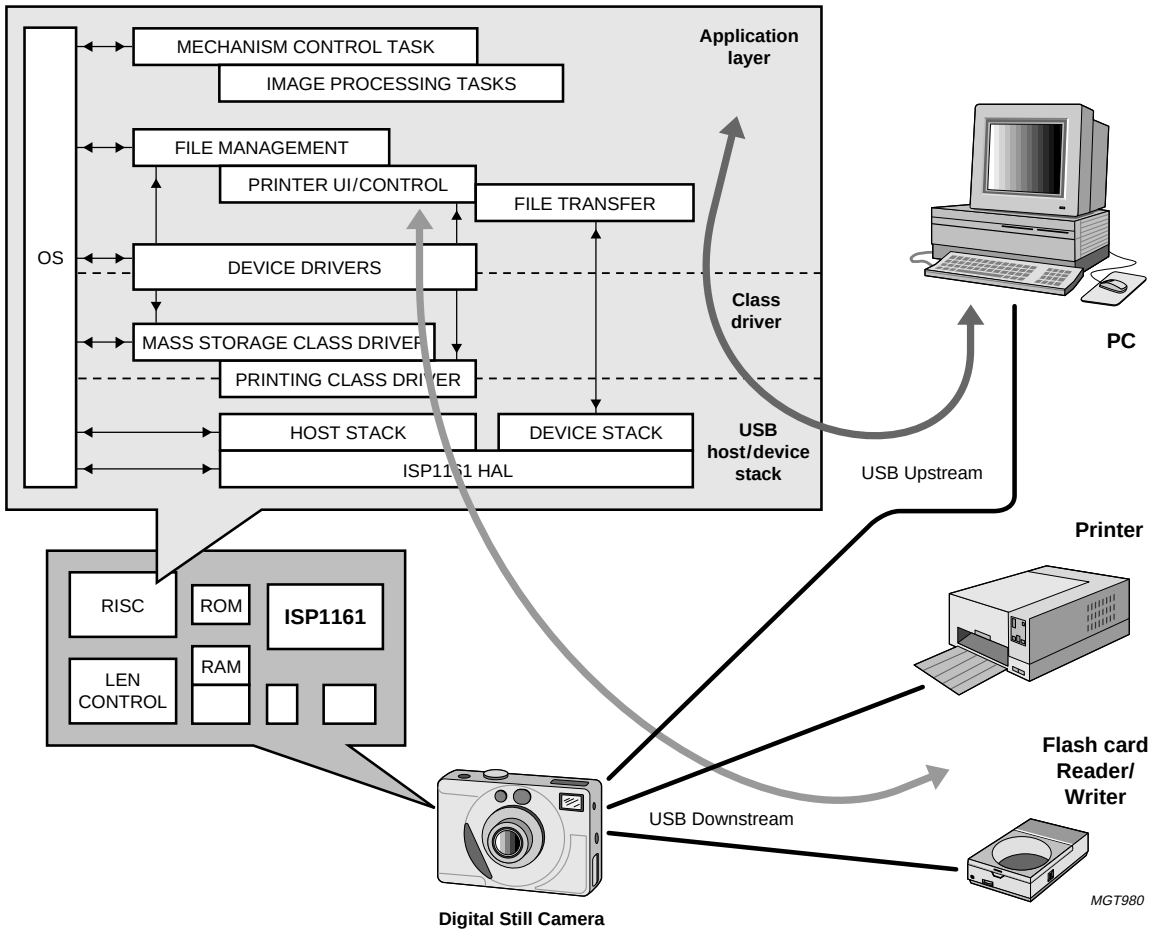
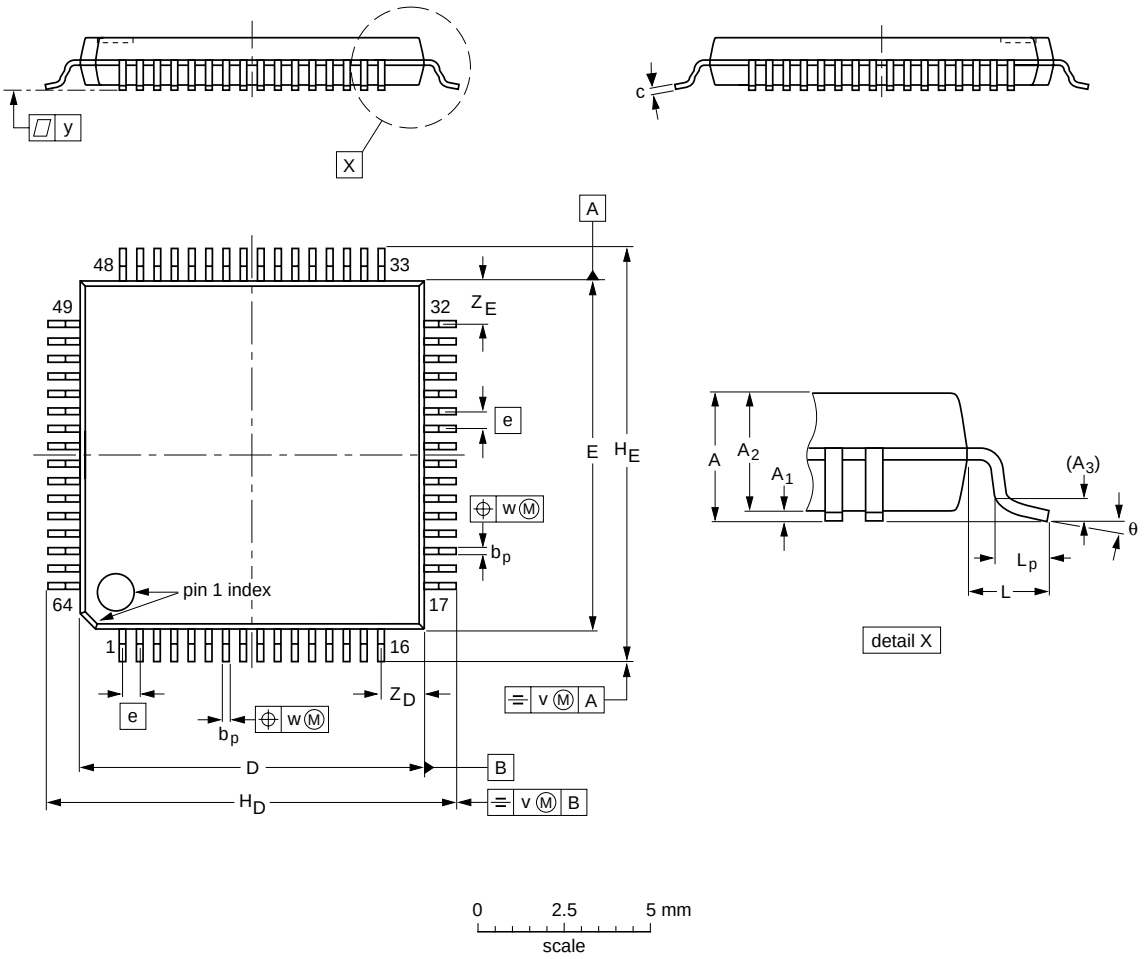


Fig 65. ISP1161 software model for DSC application.

21. Package outline

LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm **SOT314-2**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	1.60	0.20 0.05	1.45 1.35	0.25	0.27 0.17	0.18 0.12	10.1 9.9	10.1 9.9	0.5	12.15 11.85	12.15 11.85	1.0	0.75 0.45	0.2	0.12	0.1	1.45 1.05	1.45 1.05	7^o 0^o

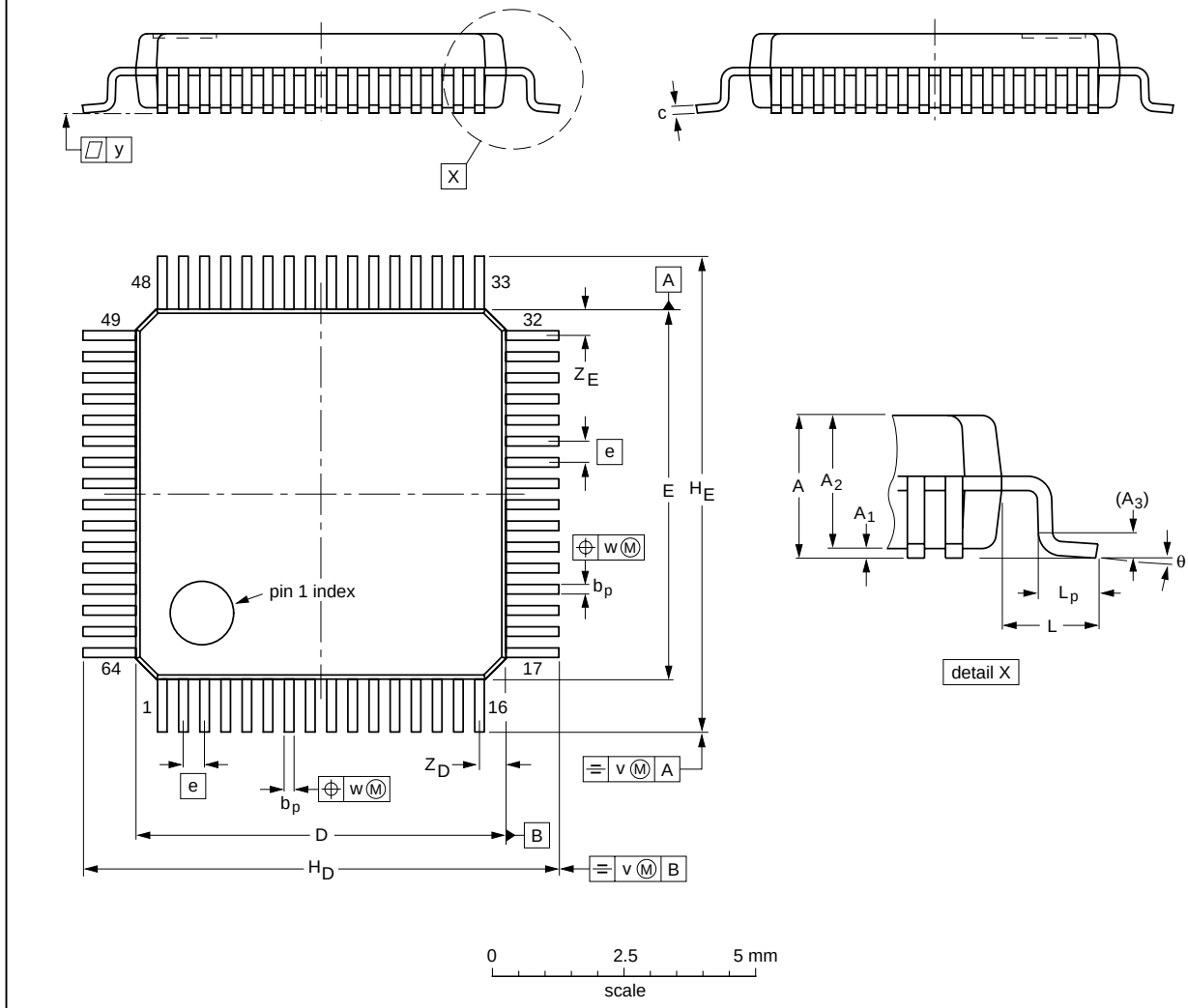
Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT314-2	136E10	MS-026				~~99-12-27~~ 00-01-19

Fig 66. LQFP64 package outline.

LQFP64: plastic low profile quad flat package; 64 leads; body 7 x 7 x 1.4 mm **SOT414-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	1.6	0.15 0.05	1.45 1.35	0.25	0.23 0.13	0.20 0.09	7.1 6.9	7.1 6.9	0.4	9.15 8.85	9.15 8.85	1.0	0.75 0.45	0.2	0.08	0.08	0.64 0.36	0.64 0.36	7^o 0^o

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT414-1	136E06	MS-026				~~99-12-27~~ 00-01-19

Fig 67. LQFP64 package outline.

22. Soldering

22.1 Introduction to soldering surface mount packages

This text gives a very brief insight to a complex technology. A more in-depth account of soldering ICs can be found in our *Data Handbook IC26; Integrated Circuit Packages* (document order number 9398 652 90011).

There is no soldering method that is ideal for all surface mount IC packages. Wave soldering can still be used for certain surface mount ICs, but it is not suitable for fine pitch SMDs. In these situations reflow soldering is recommended.

22.2 Reflow soldering

Reflow soldering requires solder paste (a suspension of fine solder particles, flux and binding agent) to be applied to the printed-circuit board by screen printing, stencilling or pressure-syringe dispensing before package placement.

Several methods exist for reflowing; for example, convection or convection/infrared heating in a conveyor type oven. Throughput times (preheating, soldering and cooling) vary between 100 and 200 seconds depending on heating method.

Typical reflow peak temperatures range from 215 to 250 °C. The top-surface temperature of the packages should preferable be kept below 220 °C for thick/large packages, and below 235 °C small/thin packages.

22.3 Wave soldering

Conventional single wave soldering is not recommended for surface mount devices (SMDs) or printed-circuit boards with a high component density, as solder bridging and non-wetting can present major problems.

To overcome these problems the double-wave soldering method was specifically developed.

If wave soldering is used the following conditions must be observed for optimal results:

- Use a double-wave soldering method comprising a turbulent wave with high upward pressure followed by a smooth laminar wave.
- For packages with leads on two sides and a pitch (e):
  - larger than or equal to 1.27 mm, the footprint longitudinal axis is **preferred** to be parallel to the transport direction of the printed-circuit board;
  - smaller than 1.27 mm, the footprint longitudinal axis **must** be parallel to the transport direction of the printed-circuit board.

  The footprint must incorporate solder thieves at the downstream end.
- For packages with leads on four sides, the footprint must be placed at a 45° angle to the transport direction of the printed-circuit board. The footprint must incorporate solder thieves downstream and at the side corners.

During placement and before soldering, the package must be fixed with a droplet of adhesive. The adhesive can be applied by screen printing, pin transfer or syringe dispensing. The package can be soldered after the adhesive is cured.

Typical dwell time is 4 seconds at 250 °C. A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

22.4 Manual soldering

Fix the component by first soldering two diagonally-opposite end leads. Use a low voltage (24 V or less) soldering iron applied to the flat part of the lead. Contact time must be limited to 10 seconds at up to 300 °C.

When using a dedicated tool, all other leads can be soldered in one operation within 2 to 5 seconds between 270 and 320 °C.

22.5 Package related soldering information

Table 128: Suitability of surface mount IC packages for wave and reflow soldering methods

Package[1]	Soldering method	
	Wave	Reflow[2]
BGA, LBGA, LFBGA, SQFP, TFBGA, VFBGA	not suitable	suitable
DHVQFN, HBCC, HBGA, HLQFP, HSQFP, HSOP, HTQFP, HTSSOP, HVQFN, HVSON, SMS	not suitable[3]	suitable
PLCC[4], SO, SOJ	suitable	suitable
LQFP, QFP, TQFP	not recommended[4][5]	suitable
SSOP, TSSOP, VSO	not recommended[6]	suitable

[1] For more detailed information on the BGA packages refer to the *(LF)BGA Application Note* (AN01026); order a copy from your Philips Semiconductors sales office.

[2] All surface mount (SMD) packages are moisture sensitive. Depending upon the moisture content, the maximum temperature (with respect to time) and body size of the package, there is a risk that internal or external package cracks may occur due to vaporization of the moisture in them (the so called popcorn effect). For details, refer to the Drypack information in the *Data Handbook IC26; Integrated Circuit Packages; Section: Packing Methods*.

[3] These packages are not suitable for wave soldering. On versions with the heatsink on the bottom side, the solder cannot penetrate between the printed-circuit board and the heatsink. On versions with the heatsink on the top side, the solder might be deposited on the heatsink surface.

[4] If wave soldering is considered, then the package must be placed at a 45° angle to the solder wave direction. The package footprint must incorporate solder thieves downstream and at the side corners.

[5] Wave soldering is suitable for LQFP, QFP and TQFP packages with a pitch (e) larger than 0.8 mm; it is definitely not suitable for packages with a pitch (e) equal to or smaller than 0.65 mm.

[6] Wave soldering is suitable for SSOP and TSSOP packages with a pitch (e) equal to or larger than 0.65 mm; it is definitely not suitable for packages with a pitch (e) equal to or smaller than 0.5 mm.

23. Revision history

Table 129: Revision history

Rev	Date	CPCN	Description
02	20021213	-	**Product data (9397 750 09567)** Modifications: • Added USB-IF certified logo. • Changed USB compliance from USB 1.1 to USB 2.0. Also, added the data rates. • Host Controller (HC) is adapted from *Open Host Controller Interface Specification for USB, release 1.0a*. • Changed peer-to-peer to point-to-point. • Globally changed ISP1181 to ISP1181B. • Globally changed the DC register names to make them consistent with the HC register names. • Globally removed references to PHP109. • Globally changed LazyClock value to 100 kHz ±50%. • Section 2: removed the feature on ESD. • Table 2: updated description for pins 19, 24, 32, 33, 37, 38 and 40. • Section 7.5: removed V_{SE} from the remark. • Updated Section 8.6. • Updated Figure 33, Figure 35 and Figure 36. • Changed register values in: – Table 21: added FSMaximumPacketSize to the FSMPS description. – Table 28: NDP is now over bits 0 and 1 only and bits 13 to 15 are read only. – Table 29: Changed description for OCPM and NOCP. – Table 30: PPCM and DR now occupy only three bits, respectively. – Table 32: added reset and access values for reserved bits; bit 17 symbol is OCIC. – Table 34: LSDA reset is 0. Also, added reset and access values for the reserved bits. – Table 36: changed reset value from 0 to1 for DREQOutputPolarity. • Section 10.5.1: modified the chip ID. • Table 50: changed reset values. • Section 10.6.1: added an example for clearer illustration. • Section 11.1.1: updated the fifth list item. • Section 11.1.2: updated the fifth list item. • Section 11.2.1 and Section 11.2.2: switched titles. • Section 11.2.1: Updated the content. • Section 11.2.2: Updated the content. • Section 11.3.6: removed “to both control endpoints” from the fourth sentence. • Section 11.4.1: changed the content in ordered list item 5. • Section 11.4.2 fourth paragraph, first sentence: changed write-protected to read and write protected. • Section 11.4.3 sixth item: added read-protected. • Section 12: removed DACK2 from the list item “Programmable.....pins DREQ2 and EOT”. • Section 12.2: removed EOP from the second sentence of the third paragraph.

Table 129: Revision history *...continued*

Rev	Date	CPCN	Description
02	**20021213**	**-**	**Product data (9397 750 09567)** Modifications (continued): • Updated Section 13.1.5. • Table 105: added the table. • Removed old Section “Reset”. • Table 111: changed the value of I_{LI}. • Table 113: added 500 µA as the maximum value for $I_{CC(susp)}$. • Table 114: changed I_{OL} and I_{OH} to 4 mA. Also, added table note 2. • Table 115: changed the values of R_{PU} and R_{PD}. • Table 116: updated. • Section 19.1: added figures 004aaa016, 004aaa017. • Section 19.2: entirely revamped. • Table 118: – changed maximum value of t_{RHDZ} to 22 ns. – changed t_{RL} to t_{RLRH} and updated Figure 51. – changed the description for t_{AH} and t_{AS}. – changed the value for t_{RLDV}. • Table 119: – Added t_{RHSH}. – Change t_{SHRL} to $t_{SHRL} + t_{RLRH}$. Also, changed the description. – Change t_{SHWL} to $t_{SHWL} + t_{WLWH}$. Also, changed the description. – Removed table notes. • Updated Figure 52. • Table 120: – Changed some maximum and minimum values. Also changed t_{RL} to t_{RLRH}. • Table 121: changed some minimum values. Also, changed t_{RL} to t_{RLRH} and added the last row. • Updated Figure 55, Figure 56 and Figure 57. • Figure 64: changed the resistor value on $\overline{GL}$.
01	**20010703**	**-**	**Product data (9397 750 08313)**

24. Data sheet status

Level	Data sheet status[1]	Product status[2][3]	Definition
I	Objective data	Development	This data sheet contains data from the objective specification for product development. Philips Semiconductors reserves the right to change the specification in any manner without notice.
II	Preliminary data	Qualification	This data sheet contains data from the preliminary specification. Supplementary data will be published at a later date. Philips Semiconductors reserves the right to change the specification without notice, in order to improve the design and supply the best possible product.
III	Product data	Production	This data sheet contains data from the product specification. Philips Semiconductors reserves the right to make changes at any time in order to improve the design, manufacturing and supply. Relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN).

[1] Please consult the most recently issued data sheet before initiating or completing a design.

[2] The product status of the device(s) described in this data sheet may have changed since this data sheet was published. The latest information is available on the Internet at URL http://www.semiconductors.philips.com.

[3] For data sheets describing multiple type numbers, the highest-level product status determines the data sheet status.

25. Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 60134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

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.

26. 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.

27. Trademarks

ARM7 and ARM9 — are trademarks of ARM Ltd.
GoodLink — is a trademark of Koninklijke Philips Electronics N.V.
Hitachi — is a registered trademark of Hitachi Ltd.
MIPS-based — is a trademark of MIPS Technologies, Inc.
SoftConnect — is a trademark of Koninklijke Philips Electronics N.V.
StrongARM — is a registered trademark of ARM Ltd.
SuperH — is a trademark of Hitachi Ltd.

Contact information

For additional information, please visit **http://www.semiconductors.philips.com**.
For sales office addresses, send e-mail to: **sales.addresses@www.semiconductors.philips.com**. **Fax: +31 40 27 24825**

9397 750 09567

Contents

Date of release: 13 December 2002 **Document order number: 9397 750 09567**

PHILIPS

Let's make things better.