ISP1362

Single-chip USB On-The-Go controller

Rev. 06 — 21 January 2009 **Product data sheet**

1. General description

The ISP1362 is a single-chip Universal Serial Bus (USB) On-The-Go (OTG) controller integrated with the advanced ST-NXP Wireless slave host controller and the ST-NXP Wireless ISP1181B peripheral controller. The USB OTG controller is compliant with Ref. 1 “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0a”. The host and peripheral controllers are compliant with Ref. 2 “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support only), supporting data transfer at full-speed (12 Mbit/s) and low-speed (1.5 Mbit/s).

The ISP1362 has two USB ports: port 1 and port 2. Port 1 can be hardware configured to function as a downstream port, an upstream port or an OTG port whereas port 2 can only be used as a downstream port. The OTG port can switch roles from host to peripheral, or from peripheral to host. The OTG port can become a host through Host Negotiation Protocol (HNP) as specified in the OTG supplement.

A USB product with OTG capability can function either as a host or as a peripheral. For instance, with this dual-role capability, a PC peripheral such as a printer may switch roles from a peripheral to a host for connecting to a digital camera so that the printer can print pictures taken by the camera without using a PC. When a USB product with OTG capability is inactive, the USB interface is turned off. This feature has made OTG a technology well-suited for use in portable devices, such as, Personal Digital Assistant (PDA), Digital Still Camera (DSC) and mobile phone, in which power consumption is a concern. The ISP1362 is an OTG controller designed to perform such functions.

2. Features

- Complies fully with:
  - Ref. 2 “Universal Serial Bus Specification Rev. 2.0”
  - Ref. 1 “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0a”
- Supports data transfer at full-speed (12 Mbit/s) and low-speed (1.5 Mbit/s)
- Adapted from Ref. 4 “Open Host Controller Interface Specification for USB Release 1.0a”
- USB OTG:
  - Supports Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) for OTG dual-role devices
  - Provides status and control signals for the software implementation of HNP and SRP
  - Provides programmable timers required for HNP and SRP
  - Supports built-in and external source of V_{BUS}
  - Output current of the built-in charge pump is adjustable by using an external capacitor

ST-NXP Wireless

- USB host:
  - Supports integrated physical 4096 bytes of multiconfiguration memory
  - Supports all four types of USB transfers: control, bulk, interrupt and isochronous
  - Supports multiframe buffering for isochronous transfer
  - Supports automatic interrupt polling rate mechanism
  - Supports paired buffering for bulk transfer
  - Directly addressable memory architecture; memory can be updated on-the-fly
- USB device:
  - Supports high performance USB interface device with integrated Serial Interface Engine (SIE), buffer memory and transceiver
  - Supports fully autonomous and multiconfiguration Direct Memory Access (DMA) operation
  - Supports up to 14 programmable USB endpoints with two fixed control IN/OUT endpoints
  - Supports integrated physical 2462 bytes of multiconfiguration memory
  - Supports endpoints with double buffering to increase throughput and ease real-time data transfer
  - Supports controllable LazyClock (110 kHz ± 50 %) output during ‘suspend’
- Supports two USB ports: port 1 and port 2
  - Port 1 can be configured to function as a downstream port, an upstream port or an OTG port
  - Port 2 can be used only as a downstream port
- Supports software-controlled connection to the USB bus (SoftConnect[1])
- Supports good USB connection indicator that blinks with traffic (GoodLink[2])
- Complies with USB power management requirements
- Supports internal power-on and low-voltage reset circuit, with possibility of a software reset
- High-speed parallel interface to most CPUs available in the market, such as Hitachi SH-3, Intel StrongARM, NXP XA, Fujitsu SPARClite, NEC and Toshiba MIPS, ARM7/9, Freescale DragonBall and PowerPC Reduced Instruction Set Computer (RISC):
  - 16-bit data bus
  - 10 MB/s data transfer rate between the microprocessor and the ISP1362
- Supports Programmed I/O (PIO) or DMA
- Supports suspend and remote wake-up
- Uses 12 MHz crystal or direct clock source with on-chip Phase-Locked Loop (PLL) for low ElectroMagnetic Interference (EMI)
- Operates at 3.3 V power supply
- Operating temperature range from −40 °C to +85 °C
- Available in 64-pin LQFP and TFBGA packages

1. SoftConnect is a trademark of ST-NXP Wireless.
2. GoodLink is a trademark of ST-NXP Wireless.

3. Applications

The ISP1362 can be used to implement a dual-role USB device in any application, USB host or USB peripheral, depending on the cable connection. If the dual-role device is connected to a typical USB peripheral, it behaves like a typical USB host. The dual-role device, however, can also be connected to a PC or any other USB host and behave like a typical USB peripheral.

3.1 Host/peripheral roles

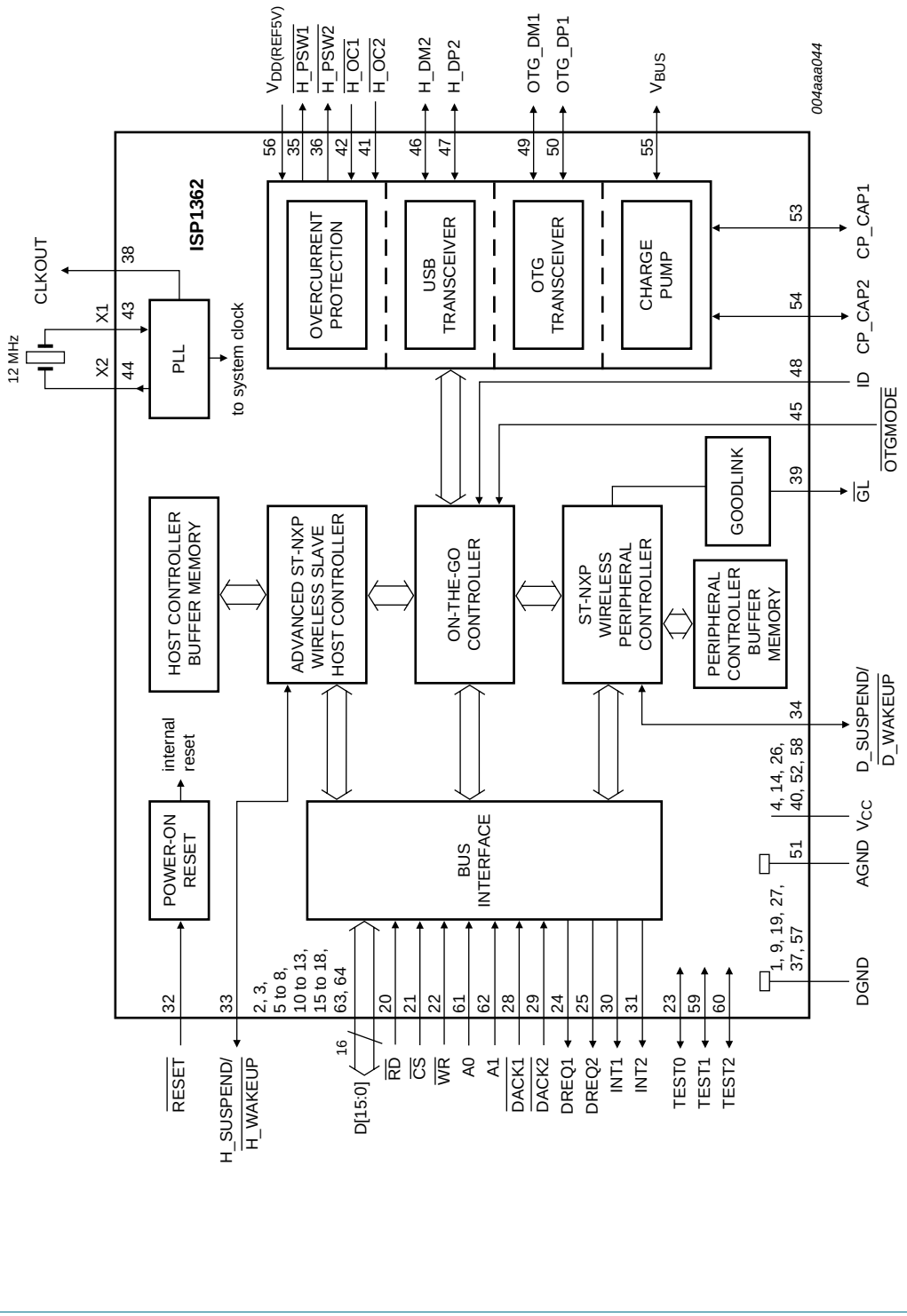
- Mobile phone to/from:
  - Mobile phone: exchange contact information
  - Digital still camera: e-mail pictures or upload pictures to the web
  - MP3 player: upload, download and broadcast music
  - Mass storage: upload and download files
  - Scanner: scan business cards
- Digital still camera to/from:
  - Digital still camera: exchange pictures
  - Mobile phone: e-mail pictures, upload pictures to the web
  - Printer: print pictures
  - Mass storage: store pictures
- Printer to/from:
  - Digital still camera: print pictures
  - Scanner: print scanned image
  - Mass storage: print files stored in a device
- MP3 player to/from:
  - MP3 player: exchange songs
  - Mass storage: upload and download songs
- Oscilloscope to/from:
  - Printer: print screen image
- Personal digital assistant to/from:
  - Personal digital assistant: exchange files
  - Printer: print files
  - Mobile phone: upload and download files
  - MP3 player: upload and download songs
  - Scanner: scan pictures
  - Mass storage: upload and download files
  - Global Positioning System (GPS): obtain directions, mapping information
  - Digital still camera: upload pictures
  - Oscilloscope: configure oscilloscope

4. Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
ISP1362BD	LQFP64	plastic low profile quad flat package; 64 leads; body 10 × 10 × 1.4 mm	SOT314-2
ISP1362EE	TFBGA64	plastic thin fine-pitch ball grid array package; 64 balls; body 6 × 6 × 0.8 mm	SOT543-1

5. Block diagram



The figure shows the LQFP pinout. For the TFBGA ballout, see Table 2.

Fig 1. Block diagram

6. Pinning information

6.1 Pinning

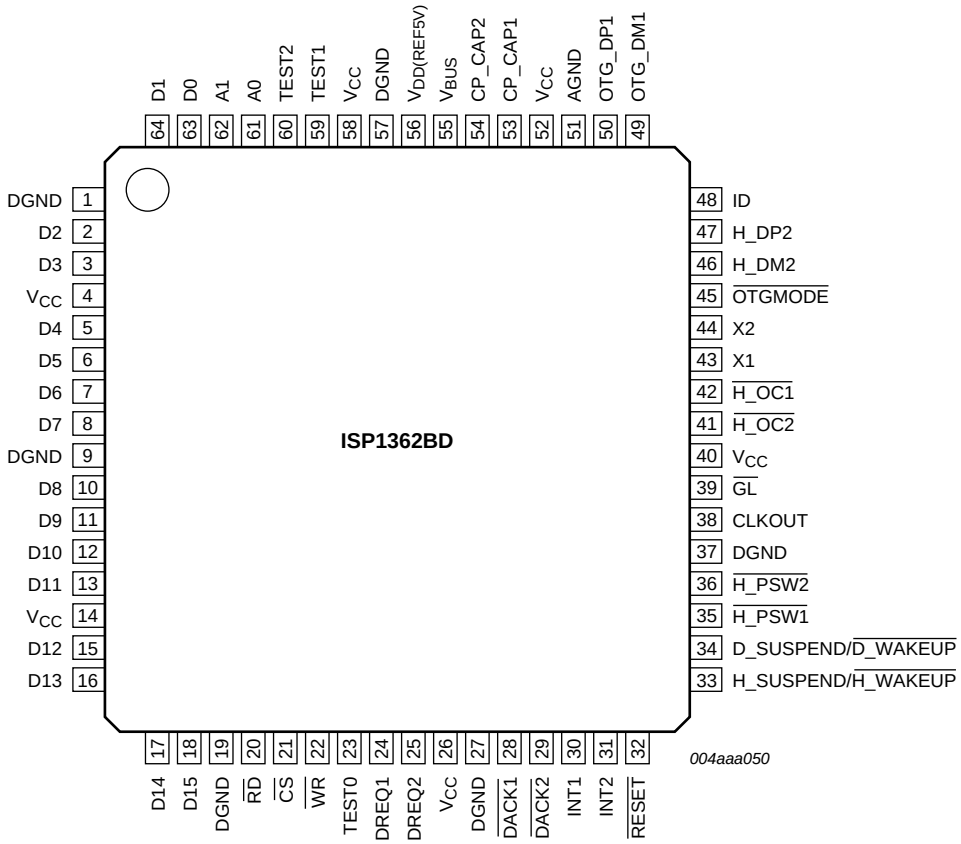


Fig 2. Pin configuration LQFP64

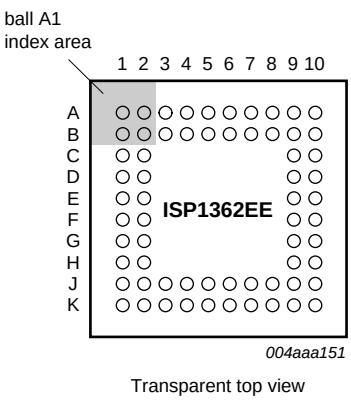


Fig 3. Pin configuration TFBGA64

6.2 Pin description

Table 2. Pin description

Symbol[1]	Pin		Type	Description
	LQFP64	TFBGA64		
DGND	1	B1	-	digital ground
D2	2	C2	I/O	bit 2 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D3	3	C1	I/O	bit 3 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
V_{CC}	4	D2	-	supply voltage (3.3 V); it is recommended that you connect a decoupling capacitor of 0.01 μF
D4	5	D1	I/O	bit 4 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D5	6	E2	I/O	bit 5 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D6	7	E1	I/O	bit 6 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D7	8	F2	I/O	bit 7 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
DGND	9	F1	-	digital ground
D8	10	G2	I/O	bit 8 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D9	11	G1	I/O	bit 9 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D10	12	H2	I/O	bit 10 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D11	13	H1	I/O	bit 11 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output

Table 2. Pin description *…continued*

Symbol[1]	Pin		Type	Description
	LQFP64	TFBGA64		
V_{CC}	14	J2	-	supply voltage (3.3 V); it is recommended that you connect a decoupling capacitor of 0.01 μF
D12	15	J1	I/O	bit 12 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D13	16	K1	I/O	bit 13 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D14	17	K2	I/O	bit 14 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D15	18	J3	I/O	bit 15 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
DGND	19	K3	-	digital ground
$\overline{RD}$	20	J4	I	read strobe input; when asserted LOW, it indicates that the host controller/peripheral controller driver is requesting a read to the buffer memory or the internal registers of the host controller/peripheral controller input with hysteresis
$\overline{CS}$	21	K4	I	chip select input (active LOW); enables the host controller/peripheral controller driver to access the buffer memory and registers of the host controller/peripheral controller input
$\overline{WR}$	22	J5	I	write strobe input; when asserted LOW, it indicates that the host controller/peripheral controller driver is requesting a write to the buffer memory or the internal registers of the host controller/peripheral controller input with hysteresis
TEST0	23	K5	I/O	for test input and output; pulled HIGH by a 100 kΩ resistor bidirectional, push-pull input, 3-state output
DREQ1	24	J6	O	DMA request output; when active, it signals the DMA controller that a data transfer is requested by the host controller; the active level (HIGH or LOW) of the request is programmed by using the HcHardwareConfiguration register (20h/A0h) If the OneDMA bit of the HcHardwareConfiguration register is set to logic 1, both the host controller and the peripheral controller DMA channel will be routed to DREQ1 and $\overline{DACK1}$. push-pull output

Table 2. Pin description *…continued*

Symbol[1]	Pin		Type	Description
	LQFP64	TFBGA64		
DREQ2	25	K6	O	DMA request output; when active, it signals the DMA controller that a data transfer is requested by the peripheral controller; the active level (HIGH or LOW) of the request is programmed by using the DcHardwareConfiguration register (BAh/BBh) push-pull output
V_{CC}	26	J7	-	supply voltage (3.3 V); it is recommended that you connect a decoupling capacitor of 0.01 μF
DGND	27	K7	-	digital ground
$\overline{DACK1}$	28	J8	I	DMA acknowledge input; indicates that a request for DMA transfer from the host controller has been granted by the DMA controller; the active level (HIGH or LOW) of the acknowledge signal is programmed by using the HcHardwareConfiguration register (20h/A0h); when not in use, this pin must be connected to V_{CC} through a 10 kΩ resistor input with hysteresis
$\overline{DACK2}$	29	K8	I	DMA acknowledge input; indicates that a request for DMA transfer from the peripheral controller has been granted by the DMA controller; the active level (HIGH or LOW) of the acknowledge signal is programmed by using the DcHardwareConfiguration register (BAh/BBh); when not in use, this pin must be connected to V_{CC} through a 10 kΩ resistor input with hysteresis
INT1	30	J9	O	interrupt request from the host controller; provides a mechanism for the host controller to interrupt the microprocessor; for details, see HcHardwareConfiguration register (20h/A0h) Section 14.4.1 If the OneINT bit of the HcHardwareConfiguration register is set to logic 1, both the host controller and the peripheral controller interrupt request will be routed to INT1. push-pull output
INT2	31	K9	O	interrupt request from the peripheral controller; provides a mechanism for the peripheral controller to interrupt the microprocessor; for details, see DcHardwareConfiguration register (BAh/BBh) Section 15.1.4 push-pull output
$\overline{RESET}$	32	K10	I	reset input input with hysteresis and internal pull-up resistor **Remark:** During reset, ensure that all the input pins to the ISP1362 are not toggling and are in their inactive states.
$\overline{H\_SUSPEND}$/ H_WAKEUP	33	J10	I/O	I/O pin (open-drain); goes HIGH when the host controller is in suspend mode; a LOW pulse must be applied to this pin to wake up the host controller; connect a 100 kΩ resistor to V_{CC} bidirectional, push-pull input, 3-state open-drain output
$\overline{D\_SUSPEND}$/ $\overline{D\_WAKEUP}$	34	H9	I/O	I/O pin (open-drain); goes HIGH when the peripheral controller is in suspend mode; a LOW pulse must be applied to this pin to wake up the peripheral controller; connect a 100 kΩ resistor to V_{CC} bidirectional, push-pull input, 3-state open-drain output

Table 2. Pin description *…continued*

Symbol[1]	Pin		Type	Description
	LQFP64	TFBGA64		
$\overline{H\_PSW1}$	35	H10	O	connects to the external PMOS switch; required when the external charge pump or external V_{BUS} is used for providing V_{BUS} to the downstream port **LOW —** switches on the PMOS providing V_{BUS} to the downstream port **HIGH —** switches off the PMOS when not in use, leave this pin open open-drain output
$\overline{H\_PSW2}$	36	G9	O	connects to the external PMOS switch **LOW —** switches on the PMOS providing V_{BUS} to the downstream port **HIGH —** switches off the PMOS when not in use, leave this pin open open-drain output
DGND	37	G10	-	digital ground
CLKOUT	38	F9	O	programmable clock output; the default clock frequency is 12 MHz and can be varied from 3 MHz to 48 MHz push-pull output
$\overline{GL}$	39	F10	O	GoodLink LED indicator output; the LED is off by default, blinks on at USB traffic open-drain output; 4 mA
V_{CC}	40	E9	-	supply voltage (3.3 V); it is recommended that you connect a decoupling capacitor of 0.01 μF
$\overline{H\_OC2}$	41	E10	I	overcurrent sense input for downstream port 2; both the digital and analog overcurrent inputs can be used for port 2, depending on the hardware mode register setting; when not in use, it is recommended that you connect this pin to the $V_{DD(REF5V)}$ pin
$\overline{H\_OC1}$	42	D9	I	overcurrent sense input for downstream port 1; both the digital and analog overcurrent inputs can be used for port 1, depending on the hardware mode register setting; when not in use, it is recommended that you connect this pin to the $V_{DD(REF5V)}$ pin
X1	43	D10	AI	crystal input; directly connected to a 12 MHz crystal; when this pin is connected to an external clock oscillator, leave pin X2 open
X2	44	C9	AO	crystal output; directly connected to a 12 MHz crystal; when pin X1 is connected to an external clock oscillator, leave this pin open
$\overline{OTGMODE}$	45	C10	I	to select whether port 1 is operating in OTG or non-OTG mode; see Table 8 input with hysteresis
H_DM2	46	B9	AI/O	downstream D− signal; host only, port 2; when not in use, leave this pin open and set bit ConnectPullDown_DS2 of the HcHardwareConfiguration register
H_DP2	47	B10	AI/O	downstream D+ signal; host only, port 2; when not in use, leave this pin open and set bit ConnectPullDown_DS2 of the HcHardwareConfiguration register
ID	48	A10	I	input pin for sensing OTG ID; the status of this input pin is reflected in the OTGStatus register (bit 0); see Table 8 input with hysteresis

Table 2. Pin description *…continued*

Symbol[1]	Pin		Type	Description
	LQFP64	TFBGA64		
OTG_DM1	49	A9	AI/O	D− signal of the OTG port, the downstream host port 1 or the upstream device port; when not in use, leave this pin open and set bit ConnectPullDown_DS1 of the HcHardwareConfiguration register[2]
OTG_DP1	50	B8	AI/O	D+ signal of the OTG port, the downstream host port 1 or the upstream device port; when not in use, leave this pin open and set bit ConnectPullDown_DS1 of the HcHardwareConfiguration register[2]
AGND	51	A8	-	analog ground; used for OTG ATX
V_{CC}	52	B7	-	supply voltage (3.3 V); it is recommended that you connect a decoupling capacitor of 0.01 μF
CP_CAP1	53	A7	AI/O	charge pump capacitor pin 1; low ESR; see Section 10.6
CP_CAP2	54	B6	AI/O	charge pump capacitor pin 2; low ESR; see Section 10.6
V_{BUS}	55	A6	I/O	analog input and output **OTG mode —** built-in charge pump output or V_{BUS} voltage comparators input; connect to pin V_{BUS} of the OTG connector **Peripheral controller mode —** input as V_{BUS} sensing; connect to pin V_{BUS} of the upstream connector **Host controller mode —** not used; leave open
$V_{DD(REF5V)}$	56	B5	I	supply reference voltage (5 V); to be used together with built-in overcurrent circuit; when built-in overcurrent circuit is not in use, this pin can be connected to V_{CC}; it is recommended that you connect a decoupling capacitor of 0.01 μF
DGND	57	A5	-	digital ground
V_{CC}	58	B4	-	supply voltage (3.3 V); it is recommended that you connect a decoupling capacitor of 0.01 μF
TEST1	59	A4	I/O	for test input and output, pulled to GND by a 10 kΩ resistor bidirectional, push-pull input, 3-state output
TEST2	60	B3	I/O	for test input and output, pulled to GND by a 10 kΩ resistor bidirectional, push-pull input, 3-state output
A0	61	A3	I	command or data phase input
A1	62	B2	I	**LOW —** PIO bus of the host controller is selected **HIGH —** PIO bus of the peripheral controller is selected input
D0	63	A2	I/O	bit 0 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output
D1	64	A1	I/O	bit 1 of the bidirectional data bus that connects to the internal registers and buffer memory of the ISP1362; the bus is in the high-impedance state when it is idle bidirectional, push-pull input, 3-state output

[1] Symbol names with an overscore (for example, $\overline{NAME}$) represent active LOW signals.

[2] In OTG mode, this pin is pulled down by an internal resistor.

7. Functional description

7.1 On-The-Go (OTG) controller

The OTG controller provides all the control, monitoring and switching functions required in OTG operations.

7.2 Advanced ST-NXP Wireless slave host controller

The advanced ST-NXP Wireless slave host controller is designed for highly optimized USB host functionality. Many advanced features are integrated to fully utilize the USB bandwidth. A number of tasks are performed at the hardware level. This reduces the requirement on the microprocessor and thus speeds up the system.

7.3 ST-NXP Wireless peripheral controller

The ST-NXP Wireless peripheral controller is a high performance USB device with up to 14 programmable endpoints. These endpoints can be configured as double-buffered endpoints to further enhance the throughput.

7.4 Phase-Locked Loop (PLL) clock multiplier

A 12 MHz-to-48 MHz clock multiplier PLL is integrated on-chip. This allows the use of a low-cost 12 MHz crystal that also minimizes ElectroMagnetic Interference (EMI) because of low frequency. No external components are required for the operation of PLL.

7.5 USB and OTG transceivers

The integrated transceivers (for typical downstream port) directly interface to the USB connectors (type A) and cables through some termination resistors. The transceiver is compliant with Ref. 2 “Universal Serial Bus Specification Rev. 2.0”.

7.6 Overcurrent protection

The ISP1362 has a built-in overcurrent protection circuitry. This feature monitors the current drawn on the downstream V_{BUS} and switches off V_{BUS} when the current exceeds the current threshold. The built-in overcurrent protection feature can be used when the port acts as a host port.

7.7 Bus interface

The bus interface connects the microprocessor to the USB host and the USB device, allowing fast and easy access to both.

7.8 Peripheral controller and host controller buffer memory

4096 bytes (host) and 2462 bytes (device) of built-in memory provide sufficient space for the buffering of USB traffic. Memory in the host controller is addressable by using the fast and versatile direct addressing method.

7.9 GoodLink

Indication of a good USB connection is provided through the GoodLink technology (open-drain, maximum current: 4 mA). During enumeration, LED indicators momentarily blink on corresponding to the enumeration traffic of the ISP1362 ports. The LED also blinks on whenever there is valid traffic to the USB ports. In suspend mode, the LED is off.

This feature of GoodLink provides a user-friendly indication on the status of the USB traffic between the host and the hub, as well as the connected devices. It is a useful diagnostics tool to isolate faulty equipment, and helps to reduce field support and hotline costs.

7.10 Charge pump

The charge pump generates a 5 V supply from 3.3 V to drive V_{BUS} when the ISP1362 is an A-device in OTG mode. For details, see Section 10.6.

8. Host and device bus interface

The interface between the external microprocessor and the ISP1362 Host Controller (HC) and peripheral controller is separately handled by the individual bus interface circuitry. The host or device automultiplex selects the path for the host access or the device access. This selection is determined by the A1 address line. For any access to the host controller or peripheral controller registers, the command phase and the data phase are needed, which is determined by the A0 address line.

All the functionality of the ISP1362 can be accessed using a group of registers and two buffer memory areas (one for the host controller and the other for the peripheral controller). Registers can be accessed using Programmed I/O (PIO) mode. The buffer memory can be accessed using both PIO and Direct Memory Access (DMA) modes.

When $\overline{CS}$ is LOW (active), address pin A1 has priority over DREQ and $\overline{DACK}$. Therefore, as long as the $\overline{CS}$ pin is held LOW, the ISP1362 bus interface does not respond to any $\overline{DACK}$ signals. When $\overline{CS}$ is HIGH (inactive), the bus interface will respond to DREQn and $\overline{DACK}$n. Address pin A1 will be ignored when $\overline{CS}$ is inactive.

An active $\overline{DACK}$n signal when DREQn is inactive will be ignored. If DREQ1, $\overline{DACK1}$, DREQ2 and $\overline{DACK2}$ are active, the bus interface will be switched off to avoid potential data corruption.

Table 3 provides the bus access priority for the ISP1362.

Table 3. Bus access priority table for the ISP1362

Priority	$\overline{CS}$	A1	$\overline{DACK1}$	$\overline{DACK2}$	DREQ1	DREQ2	Host controller and peripheral controller active
1	LOW	LOW	X	X	X	X	host controller
2	LOW	HIGH	X	X	X	X	peripheral controller
3	HIGH	X	LOW	X	HIGH	LOW	host controller[1]
4	HIGH	X	X	LOW	LOW	HIGH	peripheral controller[1]
5	HIGH	X	X	X	HIGH	HIGH	no driving

[1] Only to enable and disable the bus. Depends only on the $\overline{DACK}$ signal.

8.1 Memory organization

The buffer memory in the host controller uses a multiconfigurable direct addressing architecture. The 4096 bytes host controller buffer memory is shared by the ISTL0, ISTL1, INTL and ATL buffers. ISTL0 and ISTL1 are used for isochronous traffic (double buffer), INTL is used for interrupt traffic, and ATL is used for control and bulk traffic.

The allocation of the buffer memory follows the sequence ISTL0, ISTL1, INTL, ATL and unused memory. For example, consider that the buffer sizes of the ISTL, INTL and ATL buffers are 1024 bytes, 1024 bytes and 1024 bytes, respectively. Then, ISTL0 will start from memory location 0, ISTL1 will start from memory location 1024 (size of ISTL0), INTL will start from memory location 2048 (size of ISTL0 + size of ISTL1) and ATL will start from memory location 3072 (size of ISTL0 + size of ISTL1 + size of INTL).

The Host Controller Driver (HCD) has the responsibility to ensure that the sum of the four memory buffers does not exceed the total memory size. If this condition is violated, it will lead to data corruption. The buffer size must be a multiple of 2 bytes (one word).

The buffer memory of the peripheral controller follows a similar architecture. Details on the peripheral controller memory area allocation can be found in Section 12.3. Note that the peripheral controller buffer memory does not support direct addressing mode.

8.1.1 Memory organization for the host controller

The host controller in the ISP1362 has a total of 4096 bytes of buffer memory. This buffer area is divided into four parts (see Table 4 and Figure 4).

Table 4. Buffer memory areas and their applications

Buffer memory area	Application
ISTL0 and ISTL1	isochronous transfer (double buffering)
INTL	interrupt transfer
ATL	control and bulk transfer

The ISTL0 and ISTL1 buffers must have the same size. Memory is allocated by the host controller according to the value set by the HCD in HcISTLBufferSize, HcINTLBufferSize and HcATLBufferSize. All buffer sizes must be multiples of 2 bytes (one word).

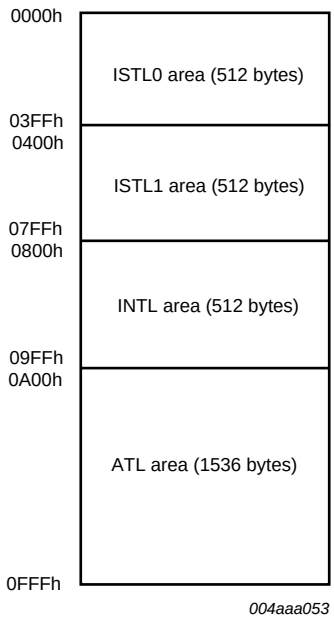


Fig 4. Recommended values of the ISP1362 buffer memory allocation

The INTL and ATL buffers use 'blocked memory management' scheme to enhance the status and control capability of each and every individual Proprietary Transfer Descriptor (PTD) structure. The INTL and ATL buffers are further divided into blocks of equal sizes, depending on the value written to the HcATLBlkSize register (ATL) and the HcINTLBlkSize register (INTL). The ISP1362 host controller supports up to 32 blocks in the ATL and INTL buffers. Each of these blocks can be used for one complete PTD data.

Note that the block size does not include the 8 bytes PTD header and is strictly the size of the payload. Both the ATL and INTL block sizes must be a multiple of double word (4 bytes).

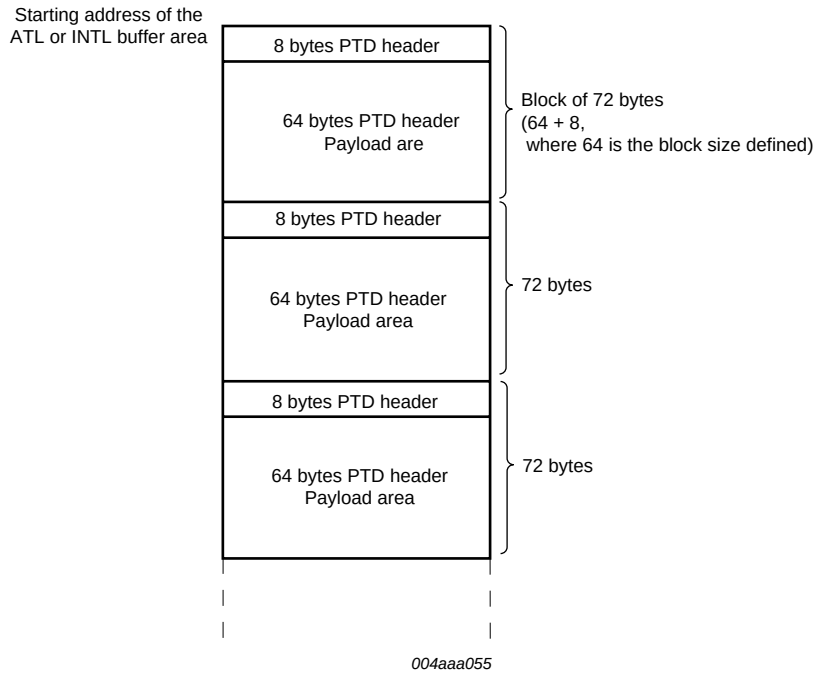
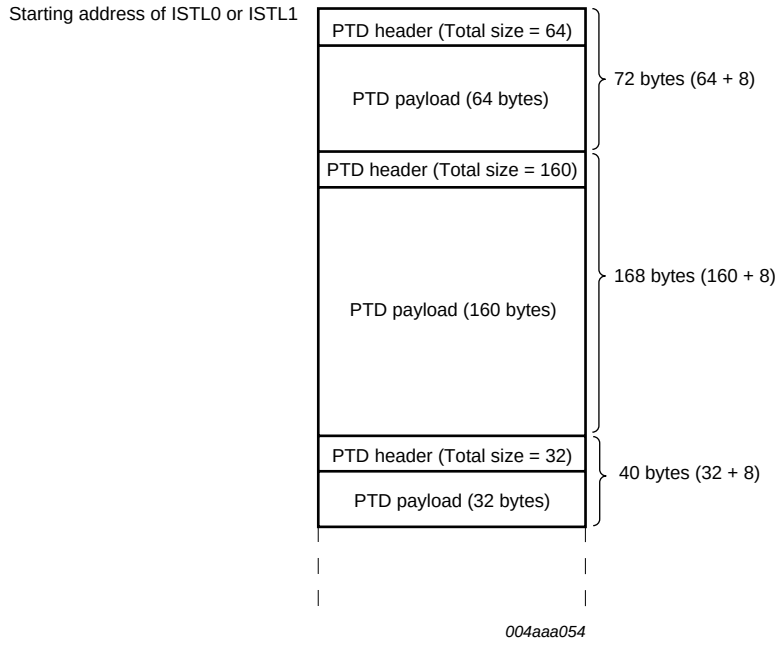


Fig 5. A sample snapshot of the ATL or INTL memory management scheme

Figure 5 provides a snapshot of a sample ATL or INTL buffer area of 256 bytes with a block size of 64 bytes. The HCD may put a PTD with payload size of up to 64 bytes but not more. Depending on the ATL or INTL buffer size, up to 32 ATL blocks and 32 INTL blocks can be allocated. Note that a portion of the ATL or INTL buffer remains unused. This is allowed but can be avoided by choosing the appropriate ATL or INTL buffer size and block size.

The ISTL0 or ISTL1 buffer memory (for isochronous transfer) uses a different memory management scheme (see Figure 6). There is no fixed block size for the ISTL buffer memory. While the PTD header remains 8 bytes for all PTDs, the PTD payload can be of any size. The PTD payload, however, is padded to the next double word boundary when the host controller calculates the location of the next PTD header. The ISP1362 host controller checks the payload size from the 'Total size' field of the PTD itself and calculates the location of the next PTD header based on this information.



'Total size' is a 10-bit field in the PTD.

Fig 6. A sample snapshot of the ISTL memory management scheme

8.1.2 Memory organization for the peripheral controller

The ISP1362 peripheral controller has a total of 2462 bytes of built-in buffer memory. This buffer memory is multiconfigurable to support the requirements of different applications. The peripheral controller buffer memory is divided into 16 areas to be used by control OUT, control IN and 14 programmable endpoints.

Figure 7 provides a snapshot of the peripheral controller buffer memory.

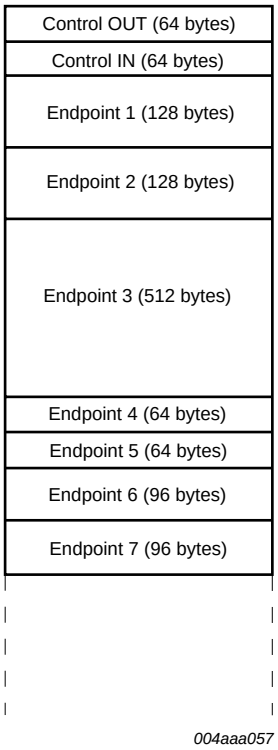


Fig 7. Peripheral controller buffer memory organization

The buffer memory is configured by DcEndpointConfiguration Registers (ECRs). Although the control endpoint has a fixed configuration, all 16 endpoints (control OUT, control IN and 14 programmable endpoints) must be configured before the peripheral controller internally allocates the buffer. The 14 programmable endpoints can be programmed into sizes ranging from 16 bytes to 1023 bytes, single or double buffering.

The peripheral controller buffer memory for each endpoint can be accessed through the DcEndpointStatusImage registers.

8.2 PIO access mode

The ISP1362 provides PIO mode for external microprocessors to access its internal control registers and buffer memory. It occupies only four I/O ports or four memory locations of a microprocessor. An external microprocessor can read or write to the internal control registers and buffer memory of the ISP1362 through PIO operating mode. Figure 8 shows the PIO interface between a microprocessor and the ISP1362.

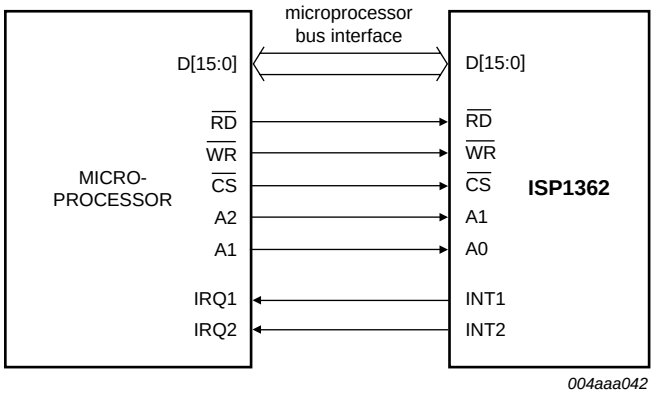


Fig 8. PIO interface between a microprocessor and the ISP1362

8.3 DMA mode

The ISP1362 also provides DMA mode for external microprocessors to access the internal buffer memory of the ISP1362. The DMA operation enables data to be transferred between the system memory of a microprocessor and the internal buffer memory of the ISP1362.

Remark: The DMA operation must be controlled by the DMA controller of the external microprocessor system (master). Figure 9 shows the DMA interface between a microprocessor system and the ISP1362.

The ISP1362 provides two DMA channels. DMA channel 1 (controlled by the DREQ1 and DACK1 signals) is for the DMA transfer between the system memory of a microprocessor and the internal buffer memory of the ISP1362 host controller. DMA channel 2 (controlled by the DREQ2 and DACK2 signals) is for the DMA transfer between the system memory of a microprocessor and the internal buffer memory of the ISP1362 peripheral controller. The ISP1362 provides an internal End-Of-Transfer (EOT) signal to terminate the DMA transfer.

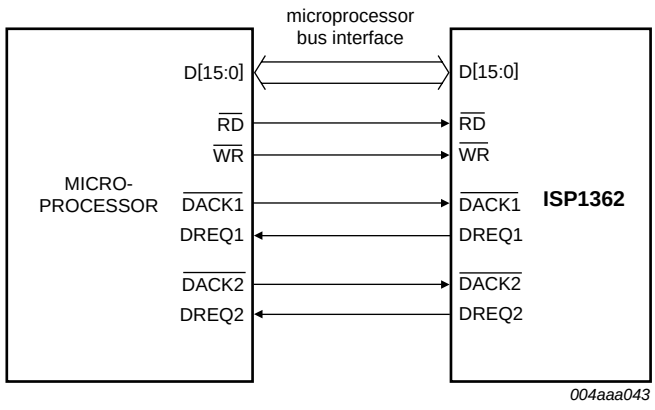


Fig 9. DMA interface between a microprocessor and the ISP1362

8.4 PIO access to internal control registers

Table 5 shows the I/O port addressing in the ISP1362. The complete I/O port address decoding must combine with the chip select signal ($\overline{CS}$) and address lines (A1 and A0). The direction of access of I/O ports, however, is controlled by the $\overline{RD}$ and $\overline{WR}$ signals.

When $\overline{RD}$ is LOW, the microprocessor reads data from the data port of the ISP1362 (see Figure 10). When $\overline{WR}$ is LOW, the microprocessor writes command to the command port or writes data to the data port (see Figure 11).

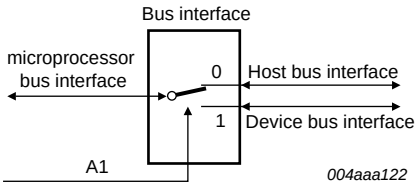
Table 5. I/O port addressing

$\overline{CS}$	A1	A0	Access	Data bus width	Description
LOW	LOW	LOW	R/W	16 bits	host controller data port
LOW	LOW	HIGH	W	16 bits	host controller command port
LOW	HIGH	LOW	R/W	16 bits	peripheral controller data port
LOW	HIGH	HIGH	W	16 bits	peripheral controller command port

The register structure in the ISP1362 is a command-data register pair structure. A complete register access needs a command phase followed by a data phase. The command (also named as the index of a register) is used to inform the ISP1362 about the register that will be accessed at the data phase.

On the 16-bit data bus of a microprocessor, a command occupies the lower byte and the upper byte is filled with zeros (see Figure 12).

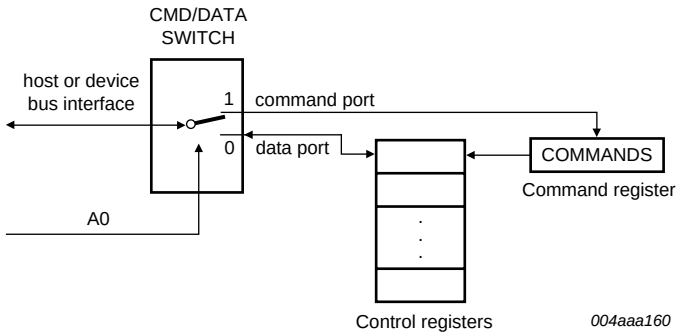
For 32-bit registers, the access cycle is shown in Figure 13. It consists of a command phase followed by two data phases.



When A1 = LOW, the microprocessor accesses the host controller.

When A1 = HIGH, the microprocessor accesses the peripheral controller.

Fig 10. Microprocessor access to the host controller or the peripheral controller



When A0 = LOW, the microprocessor accesses the data port.

When A0 = HIGH, the microprocessor accesses the command port.

Fig 11. Access to internal control registers

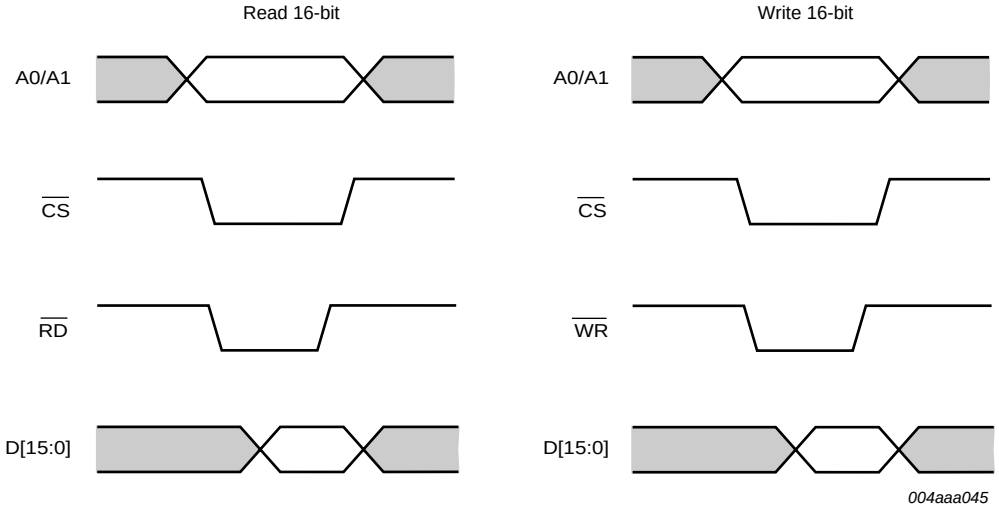


Fig 12. PIO register access

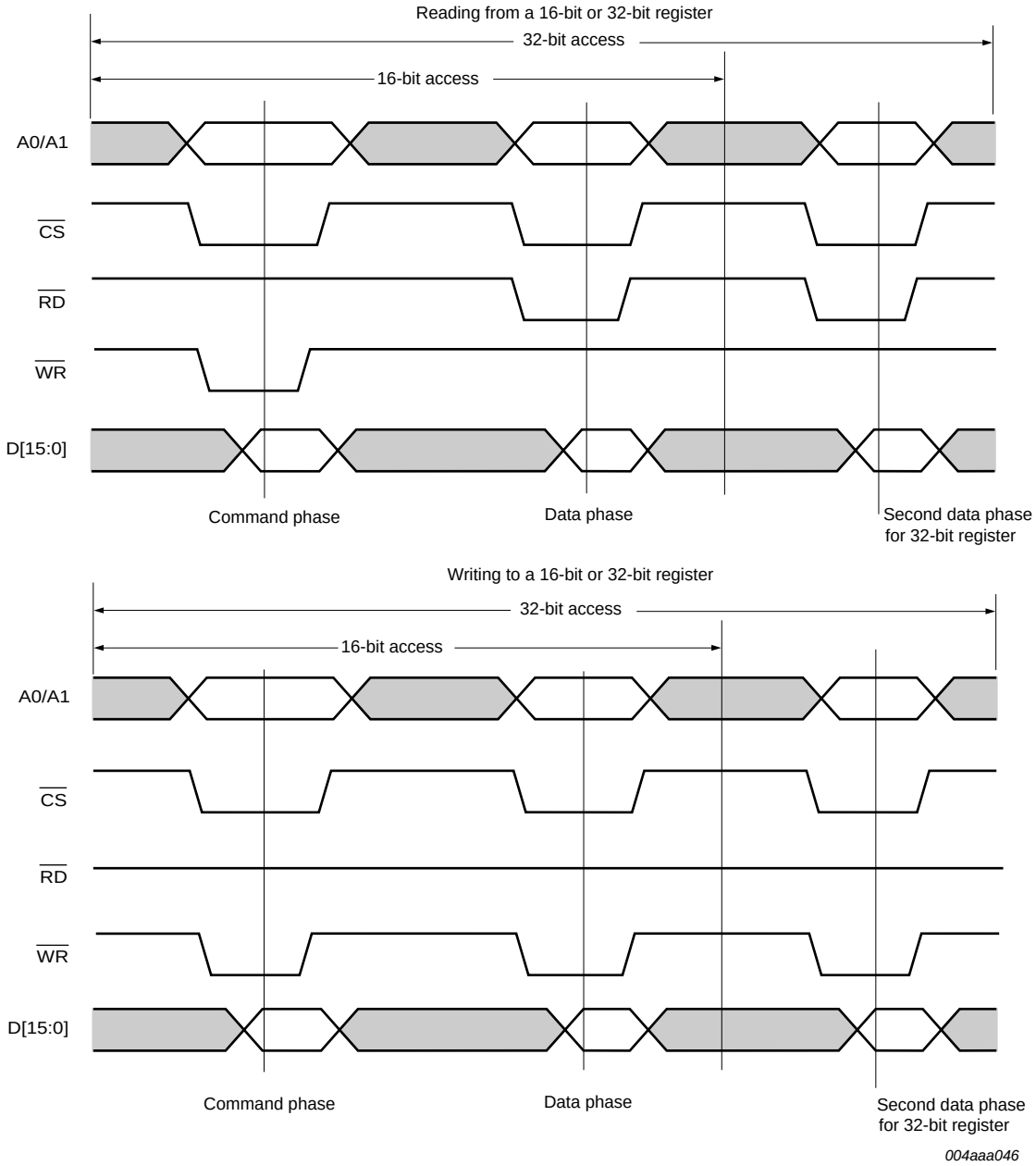


Fig 13. PIO access for a 16-bit or 32-bit register

The following is a sample code for PIO access to internal control registers:

```
unsigned long read_reg32(unsigned char reg_no)
{
 unsigned int result_l,result_h;
 unsigned long result;

 outport(hc_com, reg_no); // Command phase
 result_l = inport(hc_data); // Data phase
 result_h = inport(hc_data); // Data phase
```

```
 result = result_h;
 result = result<<16;
 result = result+result_l;

 return(result);
}

void write_reg32(unsigned char reg_no, unsigned long data2write)
{
 unsigned int low_word;
 unsigned int hi_word;

 low_word=data2write&0x0000FFFF;
 hi_word=(data2write&0xFFFF0000)>>16;

 outport(hc_com,reg_no|0x80); // Command phase
 outport(hc_data,low_word); // Data phase
 outport(hc_data,hi_word); // Data phase
}

unsigned int read_reg16(unsigned char reg_no)
{
 unsigned int result;

 outport(hc_com, reg_no); // Command phase
 result = inport(hc_data); // Data phase

 return(result);
}

void write_reg16(unsigned char reg_no, unsigned int data2write)
{
 outport(hc_com,reg_no|0x80); // Command phase
 outport(hc_data,data2write); // Data phase
}
```

8.5 PIO access to the buffer memory

The buffer memory in the ISP1362 can be addressed using either the direct addressing method or the indirect addressing method.

8.5.1 PIO access to the buffer memory by using direct addressing

This method uses the HcDirectAddressLength register to specify two parameters required to randomly access the ISP1362 buffer memory (total of 4096 bytes). These two parameters are:

Starting address — location to start writing or reading

Data length — number of bytes to write or read

The following is a sample code to set the HcDirectAddressLength register:

```
void Set_DirAddrLen(unsigned int data_length,unsigned int addr)
{
 unsigned long RegData = 0;

 RegData =(long)(addr&0x7FFF);
 RegData|=(((long)data_length)<<16);

 write_reg32(HcDirAddrLen,RegData);
}
```

After the proper value is written to the HcDirectAddressLength register, data is accessible from the HcDirectAddressData register (called as HcDirAddr_Port in the following sample code). A sample code to write word_size bytes of data from *w_ptr to the memory locations of the ISP1362 buffer starting from the address start_addr is as follows:

```
void direct_write(unsigned int *w_ptr,unsigned int start_addr,unsigned int word_size)
{
 unsigned int cnt = 0;

 Set_DirAddrLen(word_size*2,start_addr);
 outport(hc_com,HcDirAddr_Port|0x80); // hc_com is system address of
                                      // HC command port

 do
 {
      outport(hc_data,*(w_ptr+cnt)); // hc_data is system address of
                                     // HC data port
     cnt++;
 }
 while(cnt<word_size);
}
```

Direct addressing allows fast and random access to any location within the ISP1362 memory. Your program, however, needs the address location of each buffer area to access them.

8.5.2 PIO access to the buffer memory by using indirect addressing

Indirect addressing is the addressing method that is compatible with ST-NXP Wireless ISP1161 addressing mode. This method uses a unique data port for each buffer memory area (ATL, INTL, ISTL0 and ISTL1). These four data areas share the HcTransferCounter register that is used to indicate the number of bytes to be transferred.

A sample code to write an array at *a_ptr into the ATL memory area with word_size as the word size is given as follows:

```
void write_atl(unsigned int *a_ptr, unsigned int word_size)
{
 int cnt;
 write_reg16(HcTransferCnt,word_size*2);
 outport(hc_com,HcATL_Port|0x80);  // hc_com is system address of HC
                                   // command port
```

```
cnt=0;
do
{
outport(hc_data,*(a_ptr+cnt));  // hc_data is system address of HC
                                // data port
cnt++;
}
while(cnt<(word_size));
```

Remark: The HcTransferCounter register counts the number of bytes even though the transfer is in number of words. Therefore, the transfer counter must be set to word_size × 2. Incorrect setting of the HcTransferCounter register may cause the ISP1362 to go into an indeterminate state.

The buffer memory access using indirect addressing always starts from location 0 of each buffer area. Only the front portion of the memory (example: first 64 bytes of a 1024 bytes buffer) can be accessed. Therefore, to access a portion of the memory that does not start from memory location 0, all memory locations before that location must be accessed in a sequential order. The method is similar to the sequential file access method.

8.6 Setting up a DMA transfer

The ISP1362 uses two DMA channels to individually serve the host controller and the peripheral controller. The DMA transfer allows the system CPU to work on other tasks while the DMA controller transfers data to or from the ISP1362. The DMA slave controller, in the ISP1362, is compatible with the 8327 type DMA controller.

The DMA transfer can be used with direct addressing mode or indirect addressing mode. The registers used in these two modes are shown in Table 6.

Table 6. Registers used in addressing modes

Addressing mode[1]	HcDMAConfiguration bit[3:1]	Total bytes to transfer
Direct addressing	1XXB	HcDirectAddressLength
Indirect addressing	0XXB	HcTransferCounter

[1] In direct addressing mode, HcTransferCounter must be set to 0001h.

8.6.1 Configuring registers for a DMA transfer

To set up a DMA transfer, the following host controller registers must be configured, depending on the type of transfer required:

- HcHardwareConfiguration
  - DREQ1 output polarity (bit 5)
  - $\overline{\text{DACK1}}$ input polarity (bit 6)
  - $\overline{\text{DACK}}$ mode (bit 8)
- HcμPInterruptEnable
  - If you want an interrupt to be generated after the DMA transfer is complete, set EOTInterruptEnable (bit 3).
- HcμPInterrupt
  - Before initiating the DMA transfer, clear AllEOTInterrupt (bit 3). This bit is set when the DMA transfer is complete.

- HcTransferCounter
  - If DMACounterEnable of the HcDMAConfiguration register is set (that is, the DMA counter is enabled), HcTransferCounter must be set to the number of bytes to be transferred.
- HcDMAConfiguration
  - Read or write DMA (bit 0)
  - Targeted buffer: ISTL0, ISTL1, ATL and INTL (bits 1 to 3)
  - DMA enable or disable (bit 4)
  - Burst length (bits 5 to 6)
  - DMA counter enable (bit 7)

Remark: Configure the HcDMAConfiguration register only after you have configured all the other registers. The ISP1362 will assert DREQ1 once the DMA enable bit in this register is set.

8.6.2 Combining the two DMA channels

The ISP1362 allows systems with limited DMA channels to use a single DMA channel (DMA1) for both the host controller and the peripheral controller. This option can be enabled by writing logic 1 to the OneDMA bit of the HcHardwareConfiguration register. If this option is enabled, the polarity of the peripheral controller DMA and the host controller DMA must be set to DACK active LOW and DREQ active HIGH.

8.7 Interrupts

Various events in the host controller, the peripheral controller and the OTG controller can be programmed to generate a hardware interrupt. By default, the interrupt generated by the host controller and the OTG controller is routed out at the INT1 pin and the interrupt generated by the peripheral controller is routed out at the INT2 pin.

8.7.1 Interrupt in the host controller and the OTG controller

There are two levels of interrupts represented by level 1 and level 2 (see Figure 14).

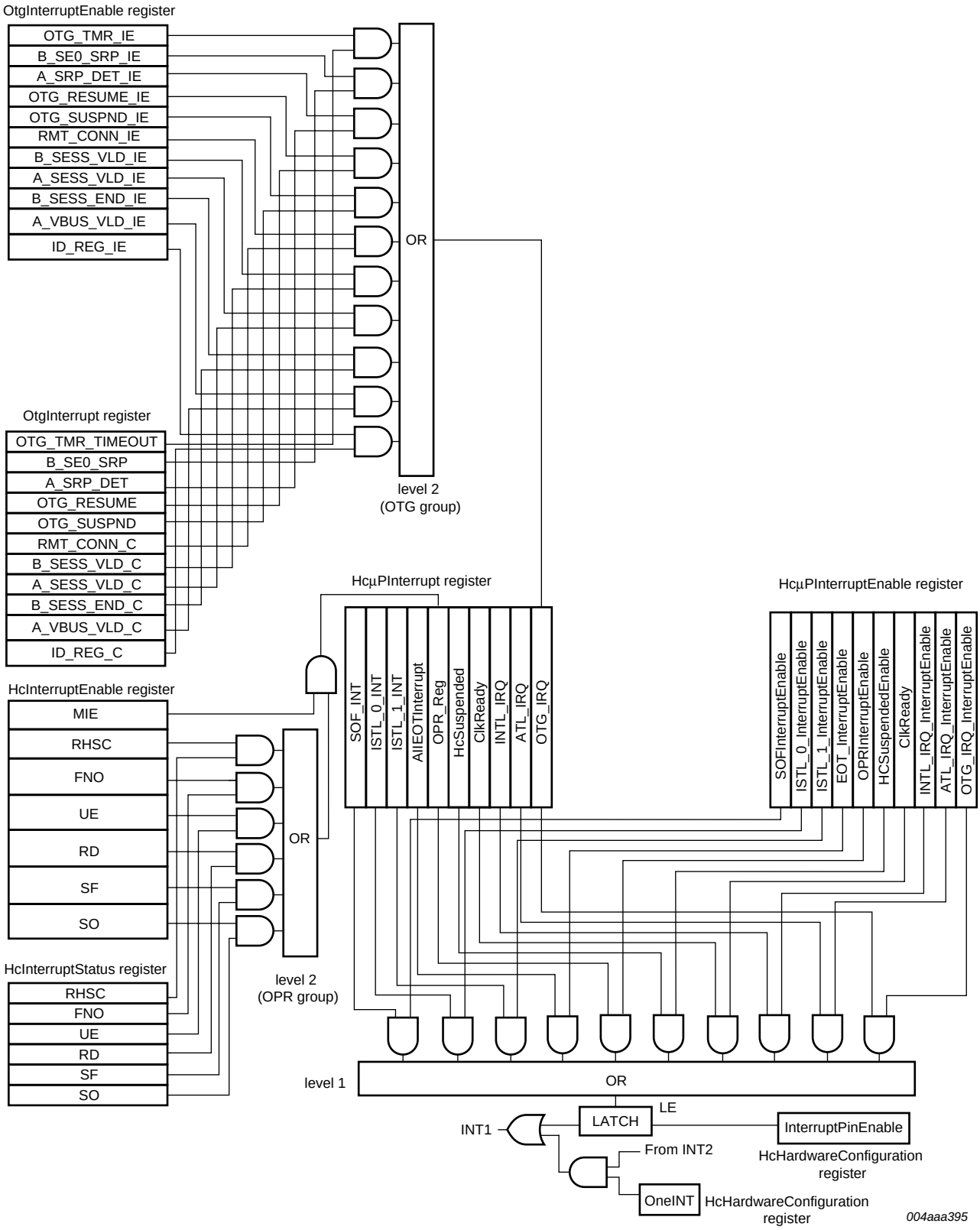


Fig 14. HC and OTG interrupt logic

Interrupt level 2 (OPR group) contains six possible interrupt events (recorded in the HcInterruptStatus register). When any of these events occurs, the corresponding bit will be set to logic 1, and if the corresponding bit in the HcInterruptEnable register is also logic 1, the 6-input OR gate will output logic 1. This output is combined with the value of MIE (bit 31 of HcInterruptEnable) using the AND operation and logic 1 output at this AND gate will cause the OPR bit in the HcμPInterrupt register to be set to logic 1.

Interrupt level 2 (OTG group) contains 11 possible interrupt events (recorded in the OtgInterrupt register). When any of these events occurs, the corresponding bit will be set to logic 1, and if the corresponding bit in the OtgInterruptEnable register is also logic 1, the 11-input OR gate will output logic 1 and cause the OTG_IRQ bit in the HcμPInterrupt register to be set to logic 1.

Level 1 interrupts contains 10 possible interrupt events. The HcμPInterrupt and HcμPInterruptEnable registers work in the same way as the HcInterruptStatus and HcInterruptEnable registers. The output from the 10-input OR gate is connected to a latch, which is controlled by InterruptPinEnable (the bit 0 of HcHardwareConfiguration register).

When the software wishes to temporarily disable the interrupt output of the ISP1362 host controller and OTG controller, follow this procedure:

1. Set the InterruptPinEnable bit in the HcHardwareConfiguration register to logic 1.
2. Clear all bits in the HcμPInterrupt register.
3. Set the InterruptPinEnable bit to logic 0.

To re-enable the interrupt generation, set the InterruptPinEnable bit to logic 1.

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

If INT1 is asserted, and the HCD wishes to temporarily mask off the INT signal without clearing the HcμPInterrupt register, follow this procedure:

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

To re-enable the interrupt generation:

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

8.7.2 Interrupt in the peripheral controller

The registers that control the interrupt generation in the ISP1362 peripheral controller are:

- DcMode (bit 3)
- DcHardwareConfiguration (bits 0 and 1)
- DcInterruptEnable
- DcInterrupt

The DcMode register (bit 3) is the overall peripheral controller interrupt enable.

DcHardwareConfiguration determines the following features:

- Level-triggered or edge-triggered (bit 1)
- Output polarity (bit 0)

For details on the interrupt logic in the peripheral controller, refer to Ref. 5 “Interrupt Control application note”.

8.7.3 Combining INT1 and INT2

In some embedded systems, interrupt inputs to the CPU are a very scarce resource. The system designer might want to use just one interrupt line to serve the host controller, the peripheral controller and the OTG controller. In such a case, make sure the OneINT feature is activated.

When OneINT (bit 9 of the HcHardwareConfiguration register) is set to logic 1, both the INT1 (HC or OTG controller) interrupt and the INT2 (peripheral controller) interrupt are routed to pin INT1, thereby reducing the hardware resource requirements.

Remark: Both the host controller (or OTG controller) and the peripheral controller interrupts must be set to the same polarity (active HIGH or active LOW) and the same trigger type (edge or level). Failure to conform to this will lead to unpredictable behavior of the ISP1362.

8.7.4 Behavior difference between level-triggered and edge-triggered interrupts

In many microprocessor systems, the operating system disables an interrupt when it is in an Interrupt Service Routine (ISR). If there is an interrupt event during this period, it will lead to level-triggered interrupt and edge-triggered interrupt.

8.7.4.1 Level-triggered interrupt

When the ISP1362 interrupt asserts, the operating system takes no action because it disables the interrupt when it is in the ISR. The interrupt line of the ISP1362 remains asserted. When the operating system exits the ISR and re-enables the interrupt processing, it sees the asserted interrupt line and immediately enters the ISR.

8.7.4.2 Edge-triggered interrupt

When the ISP1362 outputs a pulse, the operating system takes no action because it disables the interrupt when it is in the ISR. The interrupt line of the ISP1362 goes back to the inactive state. When the operating system exits the ISR and re-enables the interrupt processing, it sees no pending interrupt. As a result, the interrupt is missed.

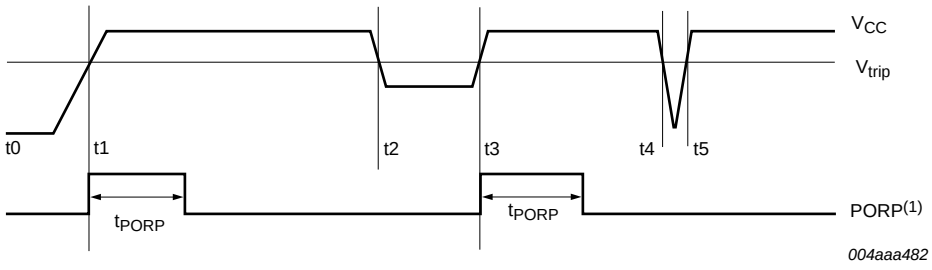
If the system needs to know whether an interrupt (approximately 160 ns pulse width) occurs during this period, it may read the HcμPInterrupt register (see Table 70).

9. Power-On Reset (POR)

When V_{CC} is directly connected to the $\overline{RESET}$ pin, the internal POR pulse width (t_{PORP}) will typically be 800 ns. The pulse is started when V_{CC} rises above V_{trip} (2.03 V).

To give a better view of the functionality, Figure 15 shows a possible curve of V_{CC} with dips at t2 to t3 and t4 to t5. If the dip at t4 to t5 is too short (that is, < 11 μs), the internal POR pulse will not react and will remain LOW. The internal POR starts with a HIGH at t0. At t1, the detector will see the passing of the trip level and a delay element will add another t_{PORP} before it drops to LOW.

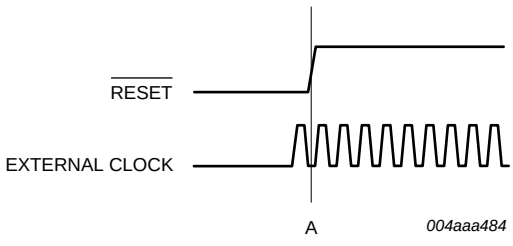
The internal POR pulse will be generated whenever V_{CC} drops below V_{trip} for more than 11 μs.



(1) PORP = Power-On Reset Pulse.

Fig 15. Internal power-on reset timing

The $\overline{RESET}$ pin can be either connected to V_{CC} (using the internal POR circuit) or externally controlled (by the micro, ASIC, and so on). Figure 16 shows the availability of the clock with respect to the external reset pulse.



Stable external clock is available at A.

Fig 16. Clock with respect to the external power-on reset

10. On-The-Go (OTG) controller

10.1 Introduction

OTG is a supplement to the Hi-Speed USB (USB 2.0) specification that augments existing USB peripherals by adding to these peripherals limited host capability to support other targeted USB peripherals. It is primarily targeted at portable devices because it addresses concerns related to such devices, such as a small connector and low power. Non-portable devices (even standard hosts), nevertheless, can also benefit from OTG features.

The ISP1362 OTG controller is designed to perform all the tasks specified in the OTG supplement. It supports Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) for dual-role devices. The ISP1362 uses the software implementation of HNP and SRP for maximum flexibility. A set of OTG registers provides the control and status monitoring capabilities to support the software HNP or SRP.

Besides the normal USB transceiver, timers and analog components required by OTG are also integrated on-chip. The analog components include:

- Built-in 3.3 V-to-5 V charge pump
- Voltage comparators
- Pull-up or pull-down resistors on data lines
- Charge or discharge resistors for V_{BUS}

10.2 Dual-role device

When port 1 of the ISP1362 is configured in OTG mode, it can be used as an OTG dual-role device. A dual-role device is a USB device that can function either as a host or as a peripheral. As a host, the ISP1362 can support all four types of transfers (control, bulk, isochronous and interrupt) at full-speed or low-speed. As a peripheral, the ISP1362 can support two control endpoints and up to 14 configurable endpoints, which can be programmed to any of the four transfer types.

The default role of the ISP1362 is controlled by the ID pin, which in turn is controlled by the type of plug connected to the mini-AB receptacle. If ID = LOW (mini-A plug connected), it becomes an A-device, which is a host by default. If ID = HIGH (mini-B plug connected), it becomes a B-device, which is a peripheral by default.

Both the A-device and the B-device work on a session base. A session is defined as the period of time in which devices exchange data. A session starts when V_{BUS} is driven and ends when V_{BUS} is turned off. Both the A-device and the B-device may start a session. During a session, the role of the host can be transferred back and forth between the A-device and the B-device any number of times by using HNP.

If the A-device wants to start a session, it turns on V_{BUS} by enabling the charge pump. The B-device detects that V_{BUS} has risen above the B_SESS_VLD level and assumes the role of a peripheral, asserting its pull-up resistor on the DP line. The A-device detects the remote pull-up resistor and assumes the role of a host. Then, the A-device can communicate with the B-device as long as it wishes. When the A-device finishes communicating with the B-device, the A-device turns-off V_{BUS} and both the devices finally go into the idle state. See Figure 18 and Figure 19.

If the B-device wants to start a session, it must initiate SRP by 'data line pulsing' and 'V_{BUS} pulsing'. When the A-device detects any of these SRP events, it turns on its V_{BUS} (note that only the A-device is allowed to drive V_{BUS}). The B-device assumes the role of a peripheral, and the A-device assumes the role of a host. The A-device detects that the B-device can support HNP by getting the OTG descriptor from the B-device. The A-device will then enable the HNP hand-off by using SetFeature (b_hnp_enable) and then go into the suspend state. The B-device signals claiming the host role by deasserting its pull-up resistor. The A-device acknowledges by going into the peripheral state. The B-device then assumes the role of a host and communicates with the A-device as long as it wishes. When the B-device finishes communicating with the A-device, both the devices finally go into the idle state. See Figure 18 and Figure 19.

10.3 Session Request Protocol (SRP)

As a dual-role device, the ISP1362 can initiate and respond to SRP. The B-device initiates SRP by data line pulsing, followed by V_{BUS} pulsing. The A-device can detect either data line pulsing or V_{BUS} pulsing.

10.3.1 B-device initiating SRP

The ISP1362 can initiate SRP by performing the following steps:

1. Detect initial conditions (read ID_REG, B_SESS_END and SE0_2MS (bits 0, 2 and 9) of the OtgStatus register).
2. Start data line pulsing (set LOC_CONN (bit 4) of the OtgControl register to logic 1).
3. Wait for 5 ms to 10 ms.
4. Stop data line pulsing (set LOC_CONN (bit 4) of the OtgControl register to logic 0).
5. Start V_{BUS} pulsing (set CHRG_VBUS (bit 1) of the OtgControl register to logic 1).
6. Wait for 10 ms to 20 ms.
7. Stop V_{BUS} pulsing (set CHRG_VBUS (bit 1) of the OtgControl register to logic 0).
8. Discharge V_{BUS} for about 30 ms (by using DISCHRG_VBUS (bit 2) of the OtgControl register), optional.

The B-device must complete both data line pulsing and V_{BUS} pulsing within 100 ms.

10.3.2 A-device responding to SRP

The A-device must be able to respond to one of the two SRP events: data line pulsing or V_{BUS} pulsing. The ISP1362 allows you to choose which SRP to support and has a mechanism to disable or enable the SRP detection. This is useful for some applications under certain cases. For example, if the A-device battery is low, it may not want to turn on its V_{BUS} by detecting SRP. In this case, it may choose to disable the SRP detection function.

When the data line SRP detection is used, the ISP1362 can detect either the DP pulsing or the DM pulsing. This means a peripheral-only device can initiate data line pulsing SRP through DP (full-speed) or DM (low-speed). A dual-role device will always initiate data line pulsing SRP through DP because it is a full-speed device.

- Steps to enable the SRP detection by V_{BUS} pulsing:
  - Set A_SEL_SRP (bit 9) of the OtgControl register to logic 0.

- Set A_SRP_DET_EN (bit 10) of the OtgControl register to logic 1.

- Steps to enable the SRP detection by data line pulsing:
  - Set A_SEL_SRP (bit 9) of the OtgControl register to logic 1.
  - Set A_SRP_DET_EN (bit 10) of the OtgControl register to logic 1.
- Steps to disable the SRP detection:
  - Set A_SRP_DET_EN (bit 10) of the OtgControl register to logic 0.

10.4 Host Negotiation Protocol (HNP)

HNP is used to transfer control of the host role between the default host (A-device) and the default peripheral (B-device) during a session. When the A-device is ready to give up its role as a host, it will condition the B-device by SetFeature (b_hnp_enable) and will go into suspend. If the B-device wants to use the bus at that time, it signals a 'disconnect' to the A-device. Then, the A-device will take the role of a peripheral and the B-device will take the role of a host.

10.4.1 Sequence of HNP events

The sequence of events for HNP as observed on the USB bus is illustrated in Figure 17.

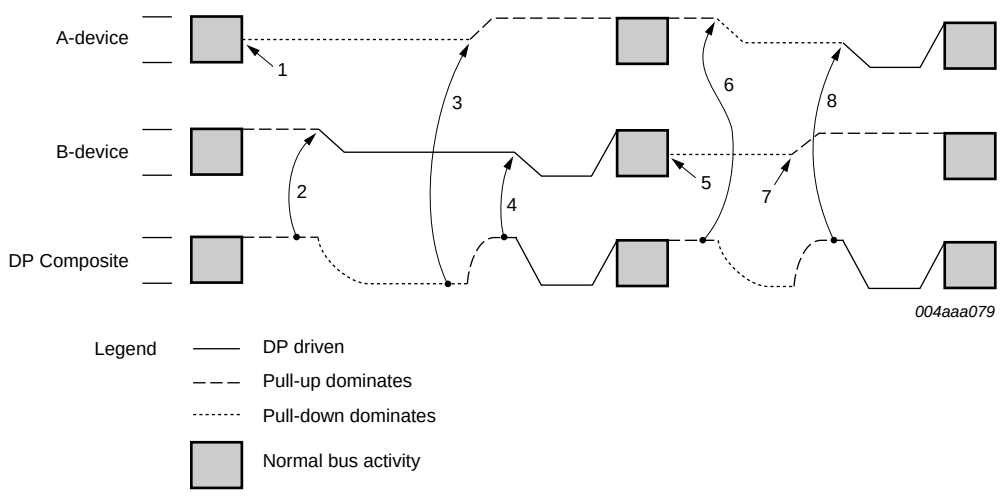


Fig 17. HNP sequence of events

As can be seen in Figure 17:

1. The A-device completes using the bus and stops all bus activities (that is, suspends the bus).
2. The B-device detects that the bus is idle for more than 5 ms and begins HNP by turning off the pull-up on DP. This allows the bus to discharge to the SE0 state.
3. The A-device detects SE0 on the bus and recognizes this as a request from the B-device to become a host. The A-device responds by turning on its DP pull-up within 3 ms of first detecting SE0 on the bus.
4. After waiting for 30 μs to ensure that the DP line is not HIGH because of the residual effect of the B-device pull-up, the B-device notices that the DP line is HIGH and the DM line is LOW (that is, J state). This indicates that the A-device has recognized the

HNP request from the B-device. At this point, the B-device becomes a host and asserts bus reset to start using the bus. The B-device must assert the bus reset (that is, SE0) within 1 ms of the time that the A-device turns on its pull-up.

5. When the B-device completes using the bus, it stops all bus activities. Optionally, the B-device may turn on its DP pull-up at this time.
6. The A-device detects lack of bus activities for more than 3 ms and turns off its DP pull-up. Alternatively, if the A-device has no further need to communicate with the B-device, the A-device may turn off V_{BUS} and end the session.
7. The B-device turns on its pull-up.
8. After waiting 30 μs to ensure that the DP line is not HIGH because of the residual effect of the A-device pull-up, the A-device notices that the DP line is HIGH (and the DM line is LOW) indicating that the B-device is signaling a connect and is ready to respond as a peripheral. At this point, the A-device becomes a host and asserts the bus reset to start using the bus.

10.4.2 OTG state diagrams

Figure 18 and Figure 19 show the state diagrams for the dual-role A-device and the dual-role B-device, respectively. For a detailed explanation, refer to Ref. 1 “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0a”.

The OTG state machine is implemented with the software. The inputs to the state machine come from four sources: hardware signals from the USB bus, software signals from the application program, internal variables with the state machines and timers:

- Hardware inputs: Include id, a_vbus_vld, a_sess_vld, b_sess_vld, b_sess_end, a_conn, b_conn, a_bus_suspend, b_bus_suspend, a_bus_resume, b_bus_resume, a_srp_det and b_se0_srp. All these inputs can be derived from the OtgInterrupt and OtgStatus registers.
- Software inputs: Include a_bus_req, a_bus_drop and b_bus_req.
- Internal variables: Include a_set_b_hnp_en, b_hnp_enable and b_srp_done.
- Timers: The HNP state machine uses four timers: a_wait_vrise_tmr, a_wait_bcon_tmr, a_aidl_bdis_tmr and b_ase0_brst_tmr. All timers are started on entry to and reset on exit from their associated states. The ISP1362 provides a programmable timer that can be used as any of these four timers.

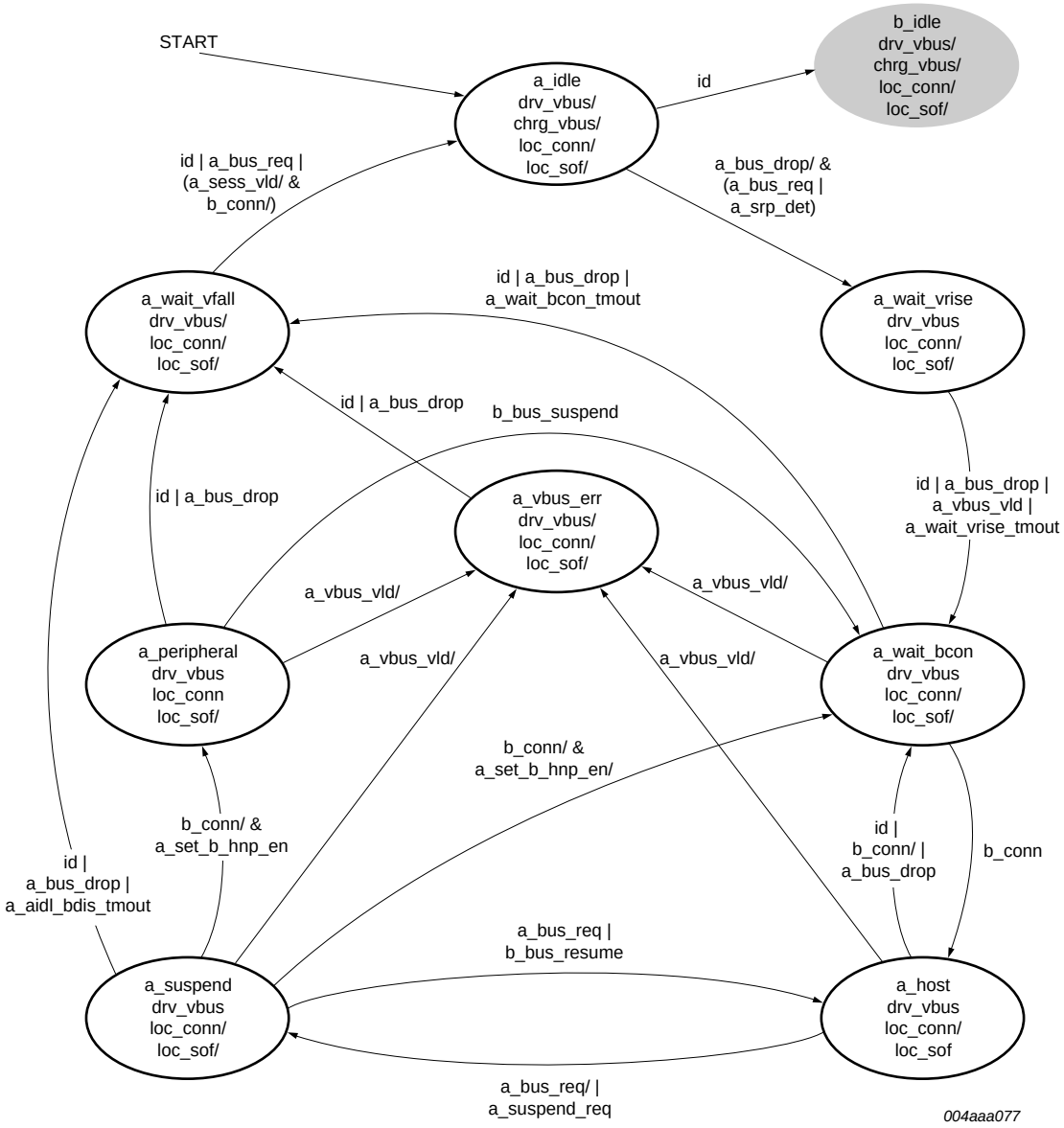


Fig 18. Dual-role A-device state diagram

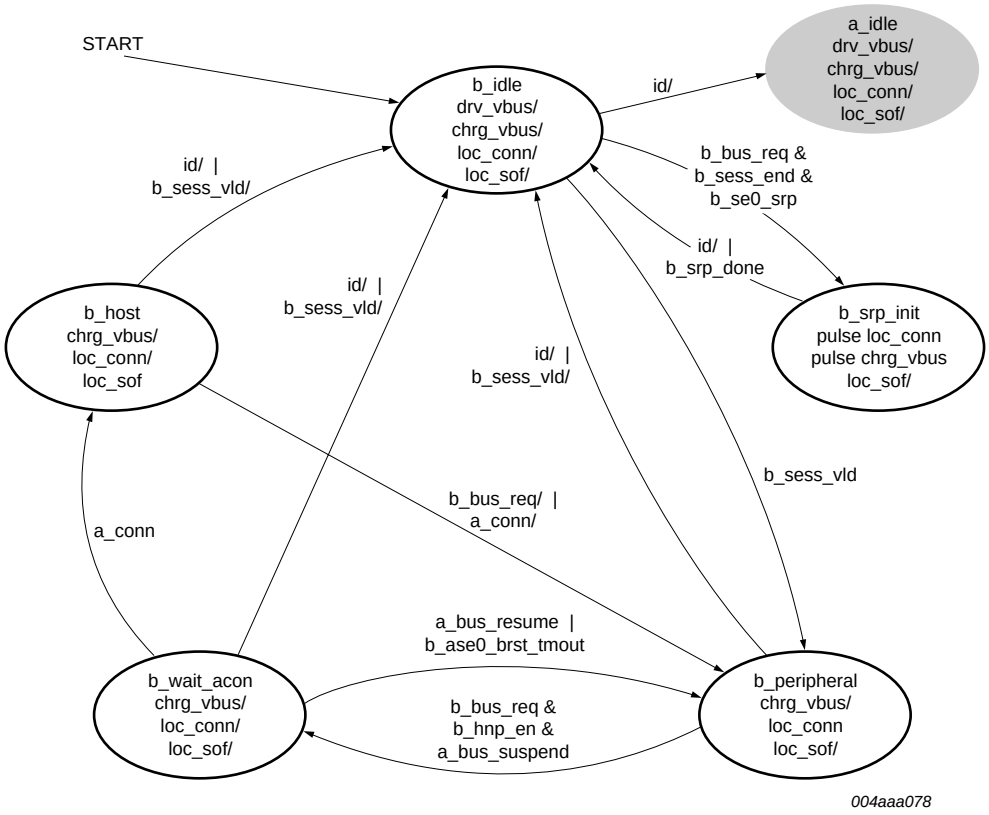


Fig 19. Dual-role B-device state diagram

10.4.3 HNP implementation and OTG state machine

The OTG state machine is the software behind all the OTG functionality. It is implemented in the microprocessor system that is connected to the ISP1362. The ISP1362 provides all input status, the output control and timers to fully support the state machine transitions in Figure 18 and Figure 19.

These registers include:

- OtgControl register: provides control to V_{BUS} driving, charging or discharging, data line pull-up or pull-down, SRP detection, and so on.
- OtgStatus register: provides status detection on V_{BUS} and data lines including ID, V_{BUS} session valid, session end, overcurrent, bus status.
- OtgInterrupt register: provides interrupts for status change in OtgStatus register bits and the OtgTimer time-out event.
- OtgInterruptEnable register: provides interrupt mask for OtgInterrupt register bits.
- OtgTimer register: provides 0.01 ms base programmable timer for use in the OTG state machine.

The OTG interrupt is generated on the INT1 pin. It is shared with the host controller interrupt. To enable the OTG interrupt, perform these steps:

1. Set the polarity and level-triggering or edge-triggering mode of the HcHardwareConfiguration register (bits 1 and 2, default is level-triggered, active LOW).
2. Set the corresponding bits of the OtgInterruptEnable register (bits 0 to 8, or some of them).
3. Set bit OTG_IRQ_InterruptEnable of the HcμPInterruptEnable register (bit 9).
4. Set bit InterruptPinEnable of the HcHardwareConfiguration register (bit 0).

When an interrupt is generated on INT1, perform these steps in the ISR to get the related OTG status:

1. Read the HcμPInterrupt register. If OTG_IRQ (bit 9) is set, then step 2.
2. Read the OtgInterrupt register. If any of the bits 0 to 4 are set, then step 3.
3. Read the OtgStatus register.

The OTG state machine routines are called when any of the inputs is changed. These inputs come from either OTG registers (hardware) or application program (software). The outputs of the state machine include control signals to the OTG register (for hardware) and states or error codes (for software). For more information, refer to ST-NXP Wireless document Ref. 3 “ISP136x Embedded Programming Guide (UM10008)”.

10.5 Power saving in the idle state and during wake-up

The ISP1362 can be put in power saving mode if the OTG device is not in a session. This significantly reduces the power consumption. In this mode, both the peripheral controller and the host controller are suspended. The PLL and the oscillator are stopped, and the charge pump is in the suspend state.

As an OTG device, however, the ISP1362 is required to respond to the SRP event. To support this, a LazyClock is kept running when the chip is in power saving mode. An SRP event will wake-up the chip (that is, enable the PLL and the oscillator). Besides this, an ID change or B_SESS_VLD detection can also wake-up the chip. These wake-up events can be enabled or disabled by programming the related bits of the OtgInterruptEnable register before putting the chip in power saving mode. If the bit is set, then the corresponding event (status change) will wake-up the ISP1362. If the bit is cleared, then the corresponding event will not wake-up the ISP1362.

You can also wake-up the ISP1362 from power saving mode by using the software. This is accomplished by accessing any of the ISP1362 registers. Accessing a register will assert $\overline{CS}$ of the ISP1362, and therefore, set it ‘awake’.

10.6 Current capacity of the OTG charge pump

The ISP1362 uses a built-in charge pump to generate a 5 V V_{BUS} supply from a 3.3 V ± 0.3 V voltage source. The only external component required is a capacitor. The value of this capacitor depends on the amount of current drive required. Table 7 provides two recommended capacitor values and the corresponding current drive.

Table 7. Recommended capacitor values

Capacitance	V_{CC}	Current
27 nF	3.0 V to 3.6 V	8 mA
82 nF	3.0 V to 3.3 V	14 mA
	3.3 V to 3.6 V	20 mA

The connection of the external capacitor (C_{ext}) is given in the partial schematics in Figure 20.

Remark: If the internal charge pump is not used, C_{ext} is not required.

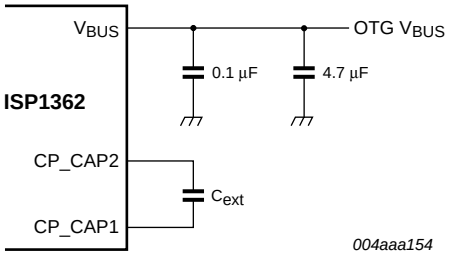


Fig 20. External capacitors connection

11. USB Host Controller (HC)

11.1 USB states of the host controller

The USB host controller in the ISP1362 has four USB states: USBOperational, USBReset, USBSuspend and USBResume. These states define the responsibilities of the host controller related to the USB signaling and bus states. These signals are visible to the Host Controller Driver (HCD), the software driver of the host controller, by using the control registers of the ISP1362 USB host controller.

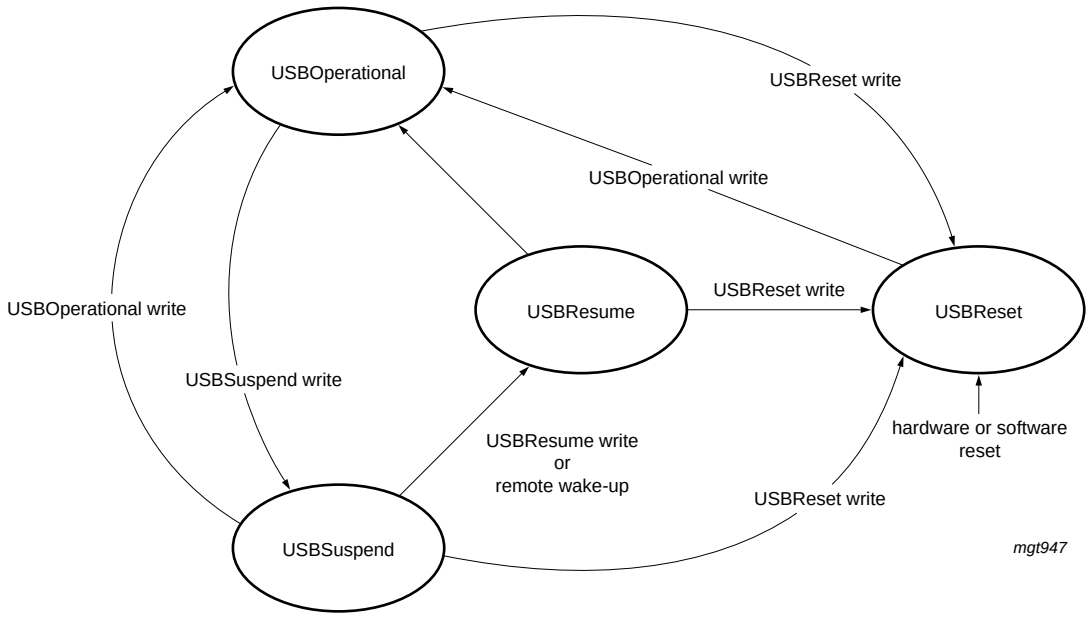


Fig 21. USB host controller states of the ISP1362

The USB states are reflected in the HostControllerFunctionalState (HCFS) field of the HcControl register. The HCD is allowed to perform only USB state transitions shown in Figure 21.

11.2 USB traffic generation

USB traffic can be generated only when the ISP1362 USB host controller is under the USBOperational state. Therefore, the HCD must set the ISP1362 USB HC in the USBOperational state. This is done by setting the HCFS field of the HcControl register before generating USB traffic.

A brief flow of the USB traffic generation is described as follows:

1. Reset the ISP1362 by using the $\overline{\text{RESET}}$ pin or the software reset.
2. Set the physical size of the ATL, interrupt, ISTL0 and ISTL1 buffers.
3. Write 32-bit hexadecimal value 8000 00FDh to the HcInterruptEnable register. This will enable all the interrupt events in the register to trigger the hardware interrupt (see Section 14.1.5).
4. Write 16-bit hexadecimal value 002Dh to the HcHardwareConfiguration register. This will set up the host controller to level triggered and active HIGH interrupt setting (see Section 14.4.1).
5. Write 0500 0B02h to HcRhDescriptorA and 0000 0000h to HcRhDescriptorB.
6. Write 16-bit hexadecimal value 0680h to the HcControl register to set the ISP1362 into operation mode (see Section 14.1.2).
7. Read the HcRhPortStatus[1] and HcRhPortStatus[2] registers. These registers contain 32-bit hexadecimal value 0001 0100h (see Section 14.3.4).
8. Connect a full-speed device to one of the downstream ports or use a 1.5 kΩ resistor to pull up the DP line (to emulate a full-speed device).

9. Read the HcRhPortStatus[1] and HcRhPortStatus[2] registers. The hexadecimal value of one of the registers must change to 0001 0101h, indicating that a device connection has been detected.
10. Write 32-bit hexadecimal value 0000 0102h into either HcRhPortStatus[1] or HcRhPortStatus[2], depending on the port that is being used.
11. Read the HcRhPortStatus[1] and HcRhPortStatus[2] registers. Depending on which port the USB device is connected to, one of the registers must contain hexadecimal value 0001 0103h.

SOF packets must be visible on DP and DM now.

The HcFmNumber register contains the current frame number, which is updated every milliseconds.

Remark: The generation of SOF is completely performed by the ISP1362 hardware. In this state of operation, if a PTD is written to the buffer memory, it will be processed and sent.

11.3 USB ports

The ISP1362 has two USB ports: port 1 and port 2. Port 1 can be configured as a downstream port (host), an upstream port (device) or a dual-role port (OTG). Port 2 is a fixed downstream port.

The function of port 1 depends on two input pins of the ISP1362, namely ID and OTGMODE.

Table 8. Port 1 function

OTGMODE	ID	Function of port 1
LOW	X	OTG
HIGH	LOW	host
HIGH	HIGH	peripheral

In OTG mode, port 2 operates as an internal host. It is not advisable to expose host port 2 to external devices because it will not respond to the SRP and HNP protocols. Besides, the current capability of V_{BUS} may be different from the OTG port's. The USB compliance checklist states that one and only one USB mini-AB receptacle is allowed on an OTG device.

11.4 Proprietary Transfer Descriptor (PTD)

PTD provides a communication channel between the HCD and the ISP1362 USB host controller. A PTD consists of a PTD header and a payload data. The size of the PTD header is 8 bytes, and it contains information required for data transfer, such as data packet size, transfer status and transfer token types. Payload data to be transferred within a particular frame must have a PTD as the header (see Figure 22).

The ISP1362 has three types of PTDs: control and bulk transfer (aperiodic transfer) PTD, interrupt transfer PTD and Isochronous (ISO) transfer PTD.

In the control and bulk transfer PTD and the interrupt transfer PTD, the buffer area is separated into equal sized blocks that are determined by HcATLBlkSize and HcINTLBlkSize. For example, if the block size is defined as 32 bytes, the first PTD

structure in the memory buffer will have an offset of 0 bytes and the second PTD structure will have an offset of 40 bytes (sum of the block size (32 bytes) and the PTD header size (8 bytes)). Because of the fixed block size of the ISP1362 host controller, however, even a PTD with 4 bytes of payload will occupy all the 40 bytes in a block.

In the isochronous PTD, the host controller uses a more flexible method to calculate the PTD offset because each PTD can have a different payload size. The actual amount of space for the payload, however, must be a multiple of double word. Therefore, a 10 bytes payload must have a reserved data size of 12 bytes. Take for example there are four PTDs in the ISTL0 buffer area with payload sizes of 200 bytes, 10 bytes, 1023 bytes and 30 bytes. Then, the offset of each of these PTDs will be as follows:

PTD1 (200 bytes) — offset = 0

PTD2 (10 bytes) — offset = (200 + 8) = 208

PTD3 (1023 bytes) — offset = (200 + 8) + (12 + 8) = 228

PTD4 (30 bytes) — offset = (200 + 8) + (12 + 8) + (1024 + 8) = 1260

The PTD data stored in the host controller buffer memory will not be processed, unless the respective control bits (ATL_Active, INTL_Active, ISTL0_BufferFull or ISTL1_BufferFull) in HcBufferStatus are set.

The PTD data in the ATL or interrupt buffer memory can be disabled by setting the respective skip bit in HcATLPTDSkipMap and HcINTLPTDSkipMap. To skip a particular PTD in the ATL or interrupt buffer, the HCD may set the corresponding bit of the SkipMap register. For example, setting the HcATLPTDSkipMap register to 0011h will cause the host controller to skip the first and the fifth PTDs in the ATL buffer memory.

Certain fields in the PTD header are used by the host controller to inform the HCD about the status of the transfer. These fields are indicated by the ‘Status Update by HC’ column. These fields are updated after every transaction to reflect the current status of the PTD.

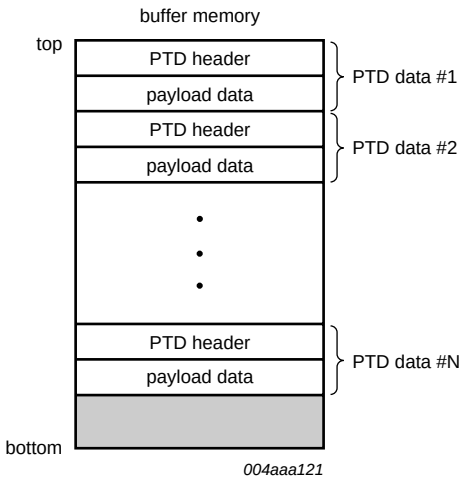


Fig 22. PTD data stored in the buffer memory

Table 9. Generic PTD structure: bit allocation

Bit[1]	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	MaxPktSize[7:0]							
Byte 3	EndpointNumber[3:0]				B3[3]	Speed	MaxPktSize[9:8]	
Byte 4	TotalBytes[7:0]							
Byte 5	B5[7]	B5[6]	reserved		DirToken[1:0]		TotalBytes[9:8]	
Byte 6	reserved	FunctionAddress[6:0]						
Byte 7	B7[7:0]							

[1] All reserved bits must be set to logic 0.

Table 10. Special fields for ATL, interrupt and ISO

Bit[1]	ATL	Interrupt	ISTL (ISO)
B3[3]	reserved	reserved	Last
B5[6]	Ping-Pong	reserved	reserved
B5[7]	Paired	reserved	reserved
B7[7:0]	reserved	PollingRate[7:5]; StartingFrame[4:0]	StartingFrame

[1] All reserved bits must be set to logic 0.

Table 11. Generic PTD structure: bit description

Name	Status update by HC	Description
ActualBytes[9:0]	Yes	This field contains the number of bytes that were transferred for this PTD.
CompletionCode[3:0]	Yes	See Table 12
Active	Yes	Set to logic 1 by the firmware to enable the execution of transactions by the host controller. When the transaction associated with this descriptor is completed, the host controller sets this bit to logic 0, indicating that a transaction for this element must not be executed when it is next encountered in the schedule.
Toggle	Yes	This bit is used to generate or compare the data PID value (DATA0 or DATA1) for IN and OUT transactions. It is updated after each successful transmission or reception of a data packet.
MaxPktSize[9:0]	No	This indicates the maximum number of bytes that can be sent to or received from the endpoint in a single data packet.
EndpointNumber[3:0]	No	This is the USB address of the endpoint within the function.
B3[3] Last (PTD)	No	This indicates that it is the last PTD of a list. Logic 1 means that this PTD is the last PTD. The last PTD is used only for ISO. This bit is not used in the interrupt and ATL transfers. The last PTD is indicated by the HcINTLLastPTD and HcATLLastPTD registers.
Speed (low)	No	This bit indicates the speed of the endpoint. **0 —** full-speed **1 —** low-speed
TotalBytes[9:0]	No	This specifies the total number of bytes to be transferred with this data structure. This can be greater than MaxPacketSize.

Table 11. Generic PTD structure: bit description *...continued*

Name	Status update by HC	Description
B5[6] Ping-Pong	No	**0 —** This is the ping buffer of the paired buffer. Paired must be logic 1. **1 —** This is the pong buffer of the paired buffer. Paired must be logic 1.
B5[7] Paired	No	If this bit is set to logic 1, two PTDs of the same endpoint and address can be made active at the same time. This bit is used with the Ping-Pong bit. The first paired PTD always starts with Ping = 0. The Pong PTD payload can be sent out only if the Ping PTD payload is sent out. You can also monitor RAM_BUFFER _STATUS to see which PTD is currently active on the USB line.
DirToken[1:0]	No	**00 —** set up **01 —** OUT **10 —** IN **11 —** reserved
FunctionAddress[6:0]	No	This field contains the USB address of the function containing the endpoint that this PTD refers to.
B7[7:5] PollingRate B7[4:0] StartingFrame (interrupt only)	No	These two fields together select a start frame number (5 bits) and polls the interrupt device at a rate specified by PollingRate (3 bits); see Section 11.6
B7[7:0] StartingFrame (ISO only)	No	The host controller compares this byte with the current frame number (can be accessed from the HcFmNumber register). The PTD will be processed and sent out only if the starting frame number equals to the current frame number.

Table 12. CompletionCode[3:0]: bit description

Value	Field name	Description
0000	NoError	General Transfer Descriptor (TD) or isochronous data packet processing completed with no detected errors.
0001	CRC	The last data packet from the endpoint contained a Cyclic Redundancy Check (CRC) error.
0010	BitStuffing	The last data packet from the endpoint contained a bit stuffing violation.
0011	DataToggleMismatch	The last packet from the endpoint had data toggle Packet ID (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	The device did not respond to the token (IN) or did not provide a handshake (OUT).
0110	PIDCheckFailure	The check bits on PID from the endpoint failed on data PID (IN) or handshake (OUT).
0111	UnexpectedPID	The received PID was not valid when encountered, or the 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 the MaxPacketSize field of ED) or the remaining buffer size.
1001	DataUnderrun	The endpoint returned is less than MaxPacketSize and that amount was not sufficient to fill the specified buffer.
1010	-	reserved
1011	-	reserved
1100	BufferOverrun	During an IN, the host controller received data from the endpoint faster than it could be written to the system memory.
1101	BufferUnderrun	During an OUT, the host controller could not retrieve data from the system memory fast enough to keep up with the USB data rate.

11.5 Features of the control and bulk transfer (aperiodic transfer)

- A paired PTD is a special feature that provides high performance single endpoint bulk transfer and handles set-up enumeration sequence within 1 ms. A paired PTD consists of two PTDs serving the same endpoint of a device that are set active and placed in the ATL RAM at the same time. A paired PTD is specially designed for high performance of a single endpoint. They are identified by the hardware by using the 'Paired' bit in the PTD.
- Possible to send up to a maximum of 18 USB bulk packets in 1 ms frame (1.152 MB/s) by using the paired PTD feature.
- Provides the status of every transfer endpoints (PTD) by monitoring the HcATLPTDDoneMap of the ISP1362. This register provides information on which PTD transfers are complete.
- Sets the IRQ after the user-specified number of transfers is done.
- Skips any PTD that is wasting bandwidth by using HcATLPTDSkipMap.

11.5.1 Sending a USB device request (Get Descriptor)

This section provides an example on how a USB transfer descriptor 'Get Descriptor' (commonly used in device enumeration) is used to illustrate the ISP1362 PTD application. To perform this example, make sure the ISP1362 is in the operational state, and then connect a USB device (for example, a USB mouse) to a port.

Remark: For details on the USB device request, refer to Chapter 9 of Ref. 2 "Universal Serial Bus Specification Rev. 2.0".

11.5.1.1 Step 1

Set the HcATLBlkSize, HcATLPTDSkipMap and HcATLLastPTD registers to 0008h, FFFEh and 0001h, respectively.

11.5.1.2 Step 2

A PTD is then constructed based on the information given in the following sample code. This sample code places information into the correct bit location within the 8 bytes of the PTD structure.

```
ActualBytes (10 bits) = 0x00
CompletionCode (4 bits) = 0x0F
Active (1 bit) = 1
Toggle (1 bit) = 0
MaxPacketSize (10 bits) = 8
EndpointNumber (4 bits) = 0
Speed (1 bit) = 0 (full-speed; use 1 for low-speed)
DirToken (2 bits) = 0 (setup token)
TotalBytes (10 bits) = 8
CompletedPTD
     0xF800, 0x0008, 0x0008, 0x0000
```

11.5.1.3 Step 3

This array is then appended with an 8 bytes payload that specifies the type of request the host controller wants to send. For example, for Get Descriptor, the payload is 0680h, 0100h, 0000h, 0012h.

11.5.1.4 Step 4

The 16 bytes of data is now a complete PTD with an accompanying payload. This array is then copied into the ATL buffer area. Table 13 shows the ATL buffer area.

Table 13. ATL buffer area

Offset	0	1	2	3	4	5	6	7
Data	F800h	0008h	0008h	0000h	0680h	0100h	0000h	0012h

11.5.1.5 Step 5

After copying data into the ATL buffer, the host controller must be notified that the ATL buffer is now full and ready to be processed. The ATL_Active bit of the HcBufferStatus register must be set to logic 1 to inform the host controller that the data in the ATL buffer is now ready for processing. Once the ATL_Active bit of the HcBufferStatus register is set, the USB packet is sent out. The active bit in the PTD is cleared once the PTD is sent. Depending on the outcome of the USB transfer, the 4-bit completion code is updated.

11.6 Features of the interrupt transfer

- An interrupt transaction is periodically sent out, according to the ‘interrupt polling rate’ as defined in the PTD.
- An interrupt transaction causes an interrupt to the CPU only if the transaction is ACK-ed or has error conditions, such as STALL or no respond. An ACK condition occurs if data is received on the IN token or data is sent out on the OUT token.
- An interrupt is activated only once every ms as long as there is ACK for different interrupt transactions in the interrupt transfer buffer.
- Each interrupt transfer (PTD) placed in the INTL buffer can automatically hold or send data for more than 1 ms. This can be done using the parameters in the PTD.

Table 14. Interrupt polling

N bits [7:5]	StartingFrame N[4:0]	Interrupt polling interval (2^N) in ms
0	frame 0 to 31	1
1	frame 0 to 31	2
2	frame 0 to 31	4
3	frame 0 to 31	8
4	frame 0 to 31	16
5	frame 0 to 31	32
6	frame 0 to 31	64
7	frame 0 to 31	128

11.7 Features of the Isochronous (ISO) transfer

- Supports multi-buffering by using the ISTL0 or ISTL1 toggling mechanism.
- The CPU can decide (in ms) how fast it can serve the ISP1362. This gives the CPU the flexibility to decide how much time it takes to read and fill in the ISO data.
- The ISTL buffer can be updated on-the-fly by using the direct addressing memory architecture.

11.8 Overcurrent protection circuit

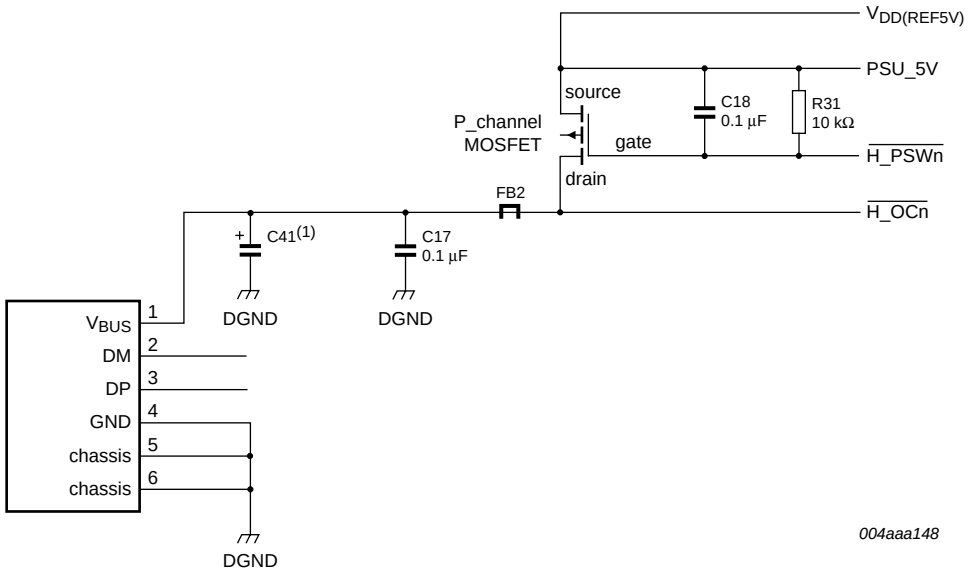
The ISP1362 has a built-in overcurrent protection circuitry. You can enable or disable this feature by setting or resetting AnalogOCEnable (bit 10) of the HcHardwareConfiguration register. If this feature is disabled, it is assumed that there is an external overcurrent protection circuitry.

11.8.1 Using internal overcurrent detection circuit

An application using the internal overcurrent detection circuit and internal 15 kΩ pull-down resistors is shown in Figure 23, where DMn denotes either OTG_DM1 or H_DM2, while DPn denotes either OTG_DP1 or H_DP2. In this example, the HCD must set both AnalogOCEnable and ConnectPullDown_DS1 (bit 10 and bit 12 of the HcHardwareConfiguration register, respectively) to logic 1.

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

In general applications, you can use a P-channel MOSFET as the power switch for V_{BUS}. Connect the 5 V power supply to the source pole of the P-channel MOSFET, V_{BUS} to the drain pole, and $\overline{\text{H\_PSWn}}$ to the gate pole. This voltage drop (ΔV) across the drain and source poles can be called 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 to turn off the P-channel MOSFET. If the P-channel MOSFET has R_{DSon} of 150 mΩ, the overcurrent threshold will be 500 mA. The selection of a P-channel MOSFET with a different R_{DSon} will result in a different overcurrent threshold.



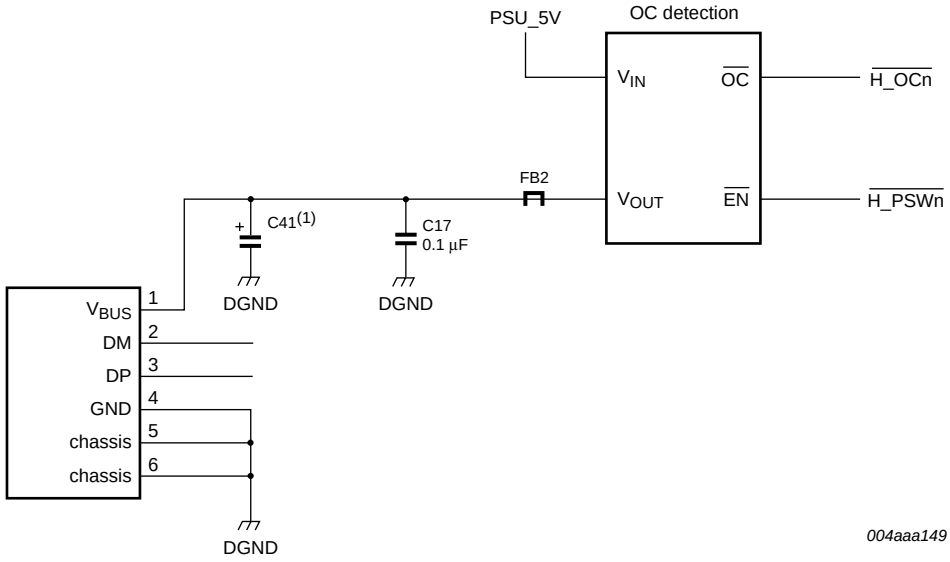
(1) 100 µF for the host port, or 4.7 µF for the OTG port.

Fig 23. Using internal overcurrent detection circuit

11.8.2 Using external overcurrent detection circuit

When $V_{DD(REF5V)}$ (pin 56) is connected to the 3.3 V power supply instead of the 5 V power supply, the internal overcurrent detection circuit cannot be used. An external overcurrent detection circuit must be used instead. Nevertheless, regardless of the V_{CC} connection, an external overcurrent detection circuit can be used from time to time. To use an external overcurrent detection circuit, set AnalogOCEnable, bit 10 of register HcHardwareConfiguration, to logic 0. By default, this bit is set to logic 0 after reset. Therefore, the HCD does not need to clear this bit.

Figure 24 shows how to use an external overcurrent detection circuit.

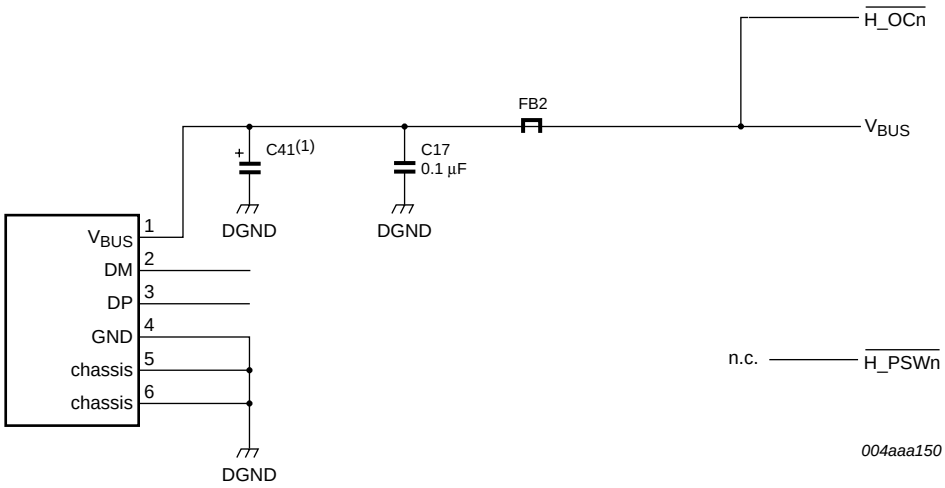


(1) 100 µF for the host port, or 4.7 µF for the OTG port.

Fig 24. Using external overcurrent detection circuit

11.8.3 Overcurrent detection circuit using internal charge pump in OTG mode

When port 1 is operating in OTG mode, you may choose to use the internal charge pump to provide 5 V V_{BUS}, or supply V_{BUS} from an external source. In this mode, the overcurrent condition is detected by a drop in V_{BUS} that will be sensed by the built-in comparator. The overcurrent condition causes a change in the A_VBUS_VLD bit of the OtgStatus register. The software must clear the DRV_VBUS bit in the OtgControl register when it detects the A_VBUS_VLD bit turning to logic 0.



(1) 100 µF for the host port, or 4.7 µF for the OTG port.

Fig 25. Using internal charge pump

11.8.4 Overcurrent detection circuit using external 5 V power source in OTG mode

In OTG mode using external 5 V power source for V_{BUS}, the circuit and the operation are the same as that for non-OTG mode (see Section 11.8.1 and Section 11.8.2).

11.9 ISP1362 host controller power management

In the ISP1362, the host controller and the peripheral controller are suspended and woken up individually. The H_SUSPEND/$\overline{\text{H\_WAKEUP}}$ and D_SUSPEND/$\overline{\text{D\_WAKEUP}}$ pins must be pulled-up by a large resistor (100 kΩ). In the suspend state, these pins are HIGH. To wake up the host controller, these pins must be pulled LOW.

The ISP1362 can partially be suspended (only the host controller or only the peripheral controller). In the partially suspended state, the clock circuit and PLL continue to work. To save power, both the host controller and the peripheral controller must be set to suspend mode.

The host controller can be suspended by writing 0680h (operational) and then 06C0h (suspend) to the HcControl register.

The host controller can be set awake by one of the following ways:

- A LOW pulse on the H_SUSPEND/$\overline{\text{H\_WAKEUP}}$ pin, minimum length of pulse is 25 ns.
- A LOW pulse on the chip select ($\overline{\text{CS}}$) pin, minimum length of pulse is 25 ns.
- A ‘resume’ signal by USB devices connected to the downstream port.

On waking up from the suspend state, the clock to the host controller will sustain for 5 ms. Within this 5 ms, the HCD must set the host controller to operational mode by writing 0680h to the HcControl register. If the HcControl register remains in the suspend state (06C0h) after waking up the host controller, the host controller will return to the suspend state after 5 ms.

12. USB peripheral controller

The design of the peripheral controller in the ISP1362 is compatible with the ST-NXP Wireless ISP1181B USB full-speed interface device IC. The functionality of the peripheral controller in the ISP1362 is similar to the ISP1181B in 16-bit bus mode. In addition, the command and register sets are also the same.

In general, the peripheral controller in the ISP1362 provides 16 endpoints for the USB device implementation. Each endpoint can be allocated RAM space in the on-chip ping pong buffer RAM.

Remark: The ping pong buffer RAM for the peripheral controller is independent of the buffer RAM for the host controller. When the buffer RAM is full, the peripheral controller transfers 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 data. The transfer of data between a microprocessor and the peripheral controller can be done in either Programmed I/O (PIO) mode or in Direct Memory Access (DMA) mode.

12.1 Peripheral controller data transfer operation

The following sessions explain how the peripheral controller in the ISP1362 handles an IN data transfer and an OUT data transfer. An IN data transfer means transfer from the ISP1362 to an external USB host (through the upstream port), and an OUT transfer means transfer from an external USB host to the ISP1362. In device mode, the ISP1362 acts as a USB device.

12.1.1 IN data transfer

1. The arrival of the IN token is detected by the Serial Interface Engine (SIE) by decoding the Packet Identifier (PID).
2. The SIE also checks the device number and the endpoint number to verify whether they are okay.
3. If the endpoint is enabled, the SIE checks the contents of the DcEndpointStatus register (ESR). If the endpoint is full, the contents of the buffer memory are sent during the data phase else an NAK handshake is sent.
4. After the data phase, the SIE expects a handshake (ACK) from the host (except for ISO endpoints).
5. On receiving the handshake (ACK), the SIE updates the contents of the DcEndpointStatus and DcInterrupt registers, 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.
6. On receiving an interrupt, the microprocessor reads the DcInterrupt register. It knows which endpoint has generated the interrupt and reads the contents of the corresponding ESR. 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.

12.1.2 OUT data transfer

1. The arrival of the OUT token is detected by the SIE by decoding the PID.
2. The SIE checks the device and endpoint numbers to verify whether they are okay.

3. If the endpoint is enabled, the SIE checks the contents of the ESR. If the endpoint is empty, the data from USB is stored in the buffer memory during the data phase else a NAK handshake is sent.
4. After the data phase, the SIE sends a handshake (ACK) to the host (except for ISO endpoints).
5. 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.
6. On receiving an interrupt, the microprocessor reads the DcInterrupt register. It knows which endpoint has generated the interrupt and reads the content of the corresponding ESR. If the buffer is full, it empties the buffer so that data can be received by the SIE at the next OUT token phase.

12.2 Device DMA transfer

12.2.1 DMA for an IN endpoint (internal peripheral controller to the 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 $\overline{\text{DACK2}}$ 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 deasserted. As a result, the DMA controller deasserts the $\overline{\text{DACK2}}$ line and releases the bus. At that moment, the whole cycle restarts for the next burst.

When the buffer is full, the DREQ2 line is deasserted and the buffer is validated (which means that it is sent to the host at the next IN token). 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 is met:

- The DMA count is complete.
- DMAEN = 0.

Remark: If the OneDMA bit in the HcHardwareConfiguration register is set to logic 1, peripheral controller DMA controller handshake signals DREQ2 and $\overline{\text{DACK2}}$ are routed to DREQ1 and $\overline{\text{DACK1}}$.

12.2.2 DMA for an OUT endpoint (external USB host to internal peripheral controller)

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. As soon as it has access, it asserts the $\overline{\text{DACK2}}$ line and starts reading data. The burst length is programmable. When the number of bytes equal to the burst length has been read, the DREQ2 line is deasserted. As a result, the DMA controller deasserts the $\overline{\text{DACK2}}$ line and releases the bus. At that moment, the whole cycle restarts for the next burst. When all the data is read, the DREQ2 line is deasserted and the buffer is cleared (this means that it can be overwritten when a new packet arrives). A DMA transfer is terminated when any of the following conditions are met:

- The DMA count is complete.
- DMAEN = 0.

Remark: If the OneDMA bit in the HcHardwareConfiguration register is set to logic 1, peripheral controller DMA controller handshake signals DREQ2 and DACK2 are routed to DREQ1 and DACK1.

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

12.3 Endpoint description

12.3.1 Endpoints with programmable buffer memory size

Each USB device is logically composed of several independent endpoints. An endpoint acts as a terminus of a communication flow between the USB host and the USB device. At design time, each endpoint is assigned a unique number (endpoint identifier, see Table 15). 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 peripheral controller has 16 endpoints: endpoint 0 (control IN and OUT) and 14 configurable endpoints, which can individually be defined as interrupt, bulk or isochronous: IN or OUT. Each enabled endpoint has an associated buffer memory, which can be accessed either by using programmed I/O interface mode or by using DMA mode.

12.3.2 Endpoint access

Table 15 lists the endpoint access modes and programmability. All endpoints support I/O mode access. Endpoints 1 to 14 also support DMA mode access. The peripheral controller buffer memory DMA access is selected and enabled using bits EPIDX[3:0] and DMAEN of the DcDMAConfiguration register. A detailed description of the peripheral controller DMA operation is given in Section 12.4.

Table 15. Endpoint access and programmability

Endpoint identifier	Buffer memory size (bytes)[1]	Double buffering	PIO mode access	DMA mode access	Endpoint type
0	64 (fixed)	no	yes	no	control OUT[2][3]
0	64 (fixed)	no	yes	no	control IN[2][3]
1 to 14	programmable	supported	supported	supported	programmable

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

[2] IN: input for the USB host (peripheral controller transmits); OUT: output from the USB host (peripheral controller receives).

[3] The data flow direction is determined by the EPDIR bit of the DcEndpointConfiguration register.

12.3.3 Endpoint buffer memory size

The size of the buffer memory determines the maximum packet size that the hardware can support for a given endpoint. Only enabled endpoints are allocated space in the shared buffer memory storage, disabled endpoints have zero bytes. Table 16 lists the programmable buffer memory sizes.

The following bits of the DcEndpointConfiguration register (ECR) affect the buffer memory allocation:

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

Remark: A register change that affects the allocation of the shared buffer memory storage among endpoints must not be made while valid data is present in any buffer memory of the enabled endpoints. Such changes render all buffer memory contents undefined.

Table 16. Programmable buffer memory 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 buffer memory can independently be configured by using its ECR, but the total physical size of all enabled endpoints (IN plus OUT) must not exceed 2462 bytes.

Table 17 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 17. Memory configuration example

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

Table 17. Memory configuration example *...continued*

Physical size (bytes)	Logical size (bytes)	Endpoint description
16	16	16 bytes interrupt IN
128	64	double-buffered 64 bytes bulk OUT
128	64	double-buffered 64 bytes bulk IN

12.3.4 Endpoint initialization

In response to standard USB request Set Interface, the firmware must program all the 16 ECRs of the peripheral controller in sequence (see Table 15), whether endpoints are enabled or not. The hardware then automatically allocates the buffer memory storage space.

If all endpoints have successfully been configured, 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 the hardware or by the USB bus occurs, the peripheral controller disables all endpoints and clears all ECRs, except the control endpoint that is fixed and always enabled.

An endpoint initialization can be done at any time. It is, however, valid only after enumeration.

12.3.5 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 (IR) are set by the SIE. The firmware then responds to the interrupt and selects the endpoint for processing.

The endpoint interrupt bit is cleared by reading the DcEndpointStatus register (ESR). The ESR 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 the peripheral controller by 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 the peripheral controller by 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.

12.3.6 Special actions on control endpoints

Control endpoints require special firmware actions. The arrival of a set-up packet flushes the IN buffer, and disables the Validate Buffer and Clear Buffer commands for the control IN and OUT endpoints. The microprocessor must re-enable these commands by sending an acknowledge set-up command to **both** the control endpoints.

This ensures that the last set-up 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 received the set-up packet.

12.4 Peripheral controller DMA transfer

Direct Memory Access (DMA) is a method to transfer data from one location to another in a computer system, without the intervention of the CPU. Many different implementations of DMA exist. The peripheral controller supports 8237 compatible mode.

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.

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: internal conditions, short or empty packet
- Programmable signal levels on pins DREQ2 and $\overline{\text{DACK2}}$

12.4.1 Selecting an endpoint for the DMA transfer

The target endpoint for DMA access is selected using bits EPDIX[3:0] of the DcDMAConfiguration register, as shown in Table 18. The transfer direction (read or write) is automatically set by the EPDIR bit in the associated ECR, to match the selected endpoint type (OUT endpoint: read; IN endpoint: write).

Automatically asserting input $\overline{\text{DACK2}}$ selects the endpoint specified in the DcDMAConfiguration register, regardless of the current endpoint used for I/O mode access.

Table 18. Endpoint selection for the 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
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

12.4.2 8237 compatible mode

This mode is selected by clearing the DAKOLY bit of the DcHardwareConfiguration register (see Table 116). The pin functions for this mode are shown in Table 19.

Table 19. 8237 compatible mode: pin functions

Symbol	Description	I/O	Function
DREQ2	DMA request of peripheral controller	O	peripheral controller requests a DMA transfer
$\overline{DACK2}$	DMA acknowledge of peripheral controller	I	DMA controller confirms the transfer
EOT	end of transfer	I	DMA controller terminates the transfer
$\overline{RD}$	read strobe	I	instructs the peripheral controller to put data on the bus
$\overline{WR}$	write strobe	I	instructs the peripheral controller 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. 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 the peripheral controller in 8237 compatible DMA mode is given in Figure 26.

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 $\overline{HLDA}$ (Hold Acknowledge). The bus operation is controlled by $\overline{MEMR}$ (Memory Read), $\overline{MEMW}$ (Memory Write), $\overline{IOR}$ (I/O Read) and $\overline{IOW}$ (I/O Write).

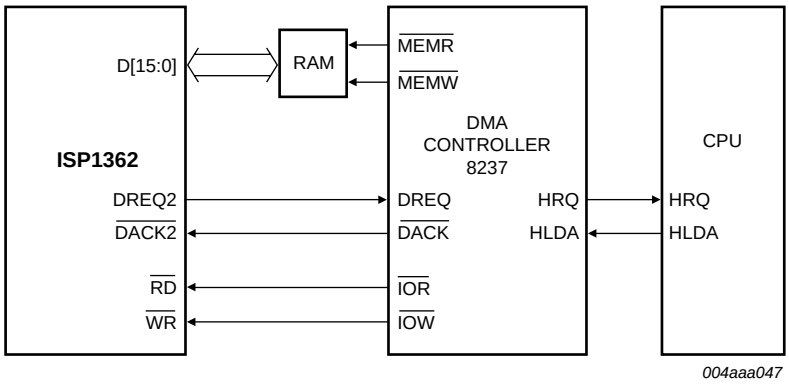


Fig 26. Peripheral controller in 8327 compatible DMA mode

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

1. The peripheral controller receives a data packet in one of its endpoint buffer memory. The packet must be transferred to memory address 1234h.
2. The peripheral controller asserts the DREQ2 signal requesting the 8237 for a DMA transfer.
3. The 8237 requests the CPU to release the bus, by asserting the HRQ signal.
4. After completing the current instruction cycle, the CPU places bus control signals ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$) and address lines in 3-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 the peripheral controller that it will start a DMA transfer.

7. The peripheral controller 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 deasserts $\overline{MEMW}$ and $\overline{IOR}$. This latches and stores the word at the desired memory location. It also informs the peripheral controller that the data on the bus lines has been transferred.
9. The peripheral controller deasserts the DREQ2 signal to indicate to the 8237 that DMA is no longer needed. In **single cycle mode**, this is done after each byte or word; in **burst mode**, following the last transferred byte or word of the DMA cycle.
10. The 8237 deasserts the $\overline{DACK}$ output, indicating that the peripheral controller must stop placing data on the bus.
11. The 8237 places bus control signals ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$) and address lines in 3-state and deasserts the HRQ signal, informing the CPU that it has released the bus.
12. The CPU acknowledges control of the bus by deasserting HLDA. After activating the bus control lines ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ and $\overline{IOW}$) and address lines, the CPU resumes the execution of instructions.

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

12.4.3 End-Of-Transfer conditions

12.4.3.1 Bulk endpoints

A DMA transfer to or from a bulk endpoint can be terminated by any of the following conditions (bit names refer to the DcDMAConfiguration register, see Table 120 and Table 121):

- 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 the DMAEN bit.

DcDMACounter register — An EOT from the DcDMACounter register is enabled by setting bit CNTREN of the DcDMAConfiguration register. The peripheral controller 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 using a control endpoint before any DMA transfer takes place. When a short packet has been enabled as an EOT indicator (SHORTP = 1), the transfer size is determined by the presence of a short packet in 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 20. Summary of EOT conditions for a bulk endpoint

EOT condition	OUT endpoint	IN endpoint
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 of the DcDMAConfiguration register	DMAEN = 0[1]	DMAEN = 0[1]

[1] The DMA transfer stops. No interrupt, however, is generated.

12.4.3.2 Isochronous endpoints

A DMA transfer to or from an isochronous endpoint can be terminated by any of the following conditions (bit names refer to the DcDMAConfiguration register, see Table 120 and Table 121):

- 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 21. Recommended EOT usage for isochronous endpoints

EOT condition	OUT endpoint	IN endpoint
DcDMACounter register zero	do not use	preferred

12.5 ISP1362 peripheral controller suspend and resume

12.5.1 Suspend conditions

The peripheral controller in the ISP1362 detects a USB suspend condition in either of the following cases:

- Constant idle state is present on the USB bus for 3 ms.
- V_{BUS} is lost.

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.

The steps leading the peripheral controller to the suspend state are as follows:

1. In the event of no SOF for 3 ms, the peripheral controller in the ISP1362 sets bit SUSPND of the DcInterrupt register. This will generate an interrupt if bit IESUSP of the DcInterruptEnable register is set.
2. When the firmware detects a suspend condition (through IESUSP), it must prepare all system components for the suspend state:
   a. All the signals connected to the peripheral controller in the ISP1362 must enter appropriate states to meet the power consumption requirements of the suspend state.

b. All the input pins of the peripheral controller in the ISP1362 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 of 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, internal clocks must be switched off by clearing bit CLKRUN of the DcHardwareConfiguration register.
5. When the firmware has set and cleared the GOSUSP bit of the DcMode register, the peripheral controller in the ISP1362 enters the suspend state. It sets the D_SUSPEND/$\overline{\text{D\_WAKEUP}}$ pin to HIGH and switches off internal clocks after 2 ms.

The peripheral controller in the ISP1362 will remain in the suspend state for at least 5 ms, before responding to external wake-up events, such as global resume, bus traffic, $\overline{\text{CS}}$ active or LOW pulse on the D_SUSPEND/$\overline{\text{D\_WAKEUP}}$ pin.

Figure 27 shows a typical timing diagram for the peripheral controller suspend and resume operations.

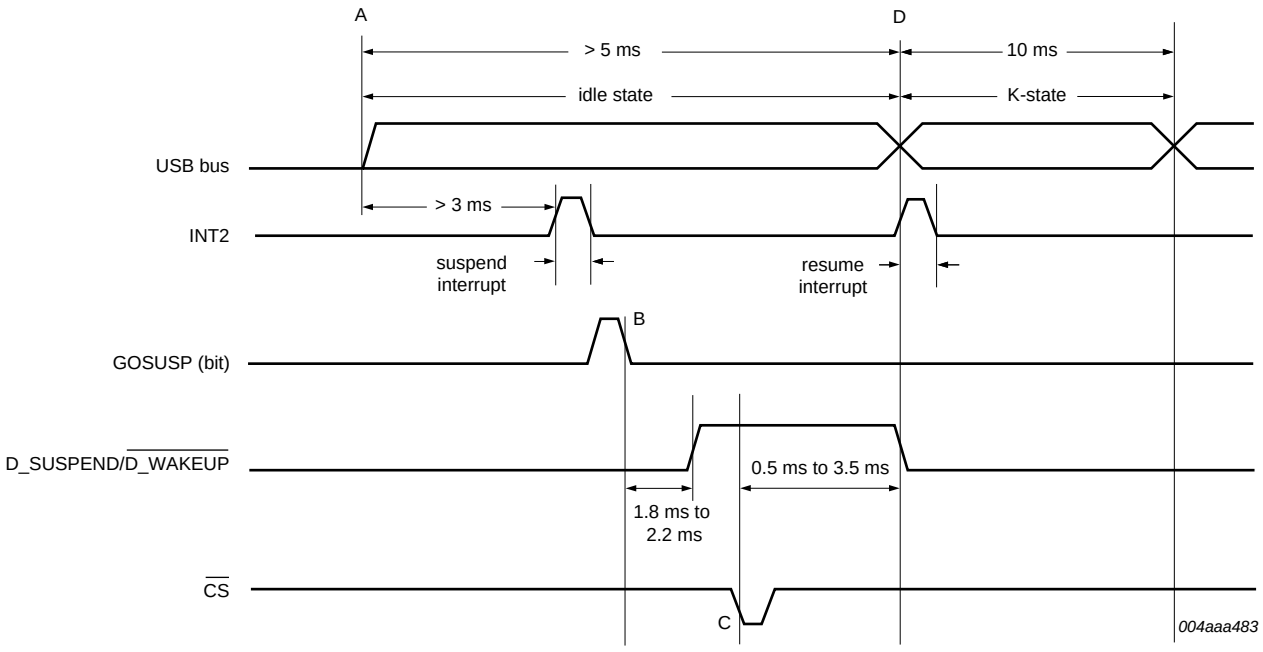


Fig 27. Suspend and resume timing

In Figure 27:

A — indicates the point at which the USB bus goes to the idle state.

B — after detecting the suspend interrupt, set and clear the GOSUSP bit in the Mode register.

C — indicates resume condition, which can be a resume signal from the host, a LOW pulse on the D_SUSPEND/$\overline{\text{D\_WAKEUP}}$ pin, or a LOW pulse on the $\overline{\text{CS}}$ pin.

D — indicates remote wake-up. The ISP1362 will drive a K-state on the USB bus for 10 ms after the D_SUSPEND/D_WAKEUP pin goes LOW or the CS pin goes LOW.

12.5.2 Resume conditions

Wake-up from the 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 using a LOW pulse on pin D_SUSPEND/D_WAKEUP or a LOW pulse on pin CS (if enabled using bit WKUPCS of the DcHardwareConfiguration register).

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

1. The internal oscillator and the PLL multiplier are re-enabled. When stabilized, clock signals are routed to all internal circuits of the peripheral controller in the ISP1362.
2. The D_SUSPEND/D_WAKEUP pin goes LOW, and the RESUME bit of the DcInterrupt register is set. This will generate an interrupt if bit IERESUME of the DcInterruptEnable register is set.
3. After 5 ms of starting the wake-up sequence, the peripheral controller in the ISP1362 resumes its normal functionality (this can be set to 100 μs by setting pin TEST0 to HIGH).
4. In a remote wake-up, the peripheral controller in the ISP1362 drives a K-state on the USB bus for 10 ms.
5. The application restores itself and other system components to normal operating mode.
6. After wake-up, internal registers of the peripheral controller in the ISP1362 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 peripheral controller in the ISP1362 to restore its full functionality.

13. OTG registers

Table 22. OTG Control registers overview

Command (Hex)		Register	Width	References	Functionality
Read	Write				
62	E2	OtgControl	16	Section 13.1 on page 59	OTG operation registers
67	not applicable	OtgStatus	16	Section 13.2 on page 61	
68	E8	OtgInterrupt	16	Section 13.3 on page 62	
69	E9	OtgInterruptEnable	16	Section 13.4 on page 64	
6A	EA	OtgTimer	32	Section 13.5 on page 65	
6C	EC	OtgAltTimer	32	Section 13.6 on page 66	

13.1 OtgControl register (R/W: 62h/E2h)

Code (Hex): 62 — read

Code (Hex): E2 — write

Table 23. OtgControl register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved				OTG_SE0_EN	A_SRP_DET_EN	A_SEL_SRP	SEL_HC_DC
Reset	-	-	-	-	0	0	0	1
Access	-	-	-	-	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	LOC_PULLDN_DM	LOC_PULLDN_DP	A_RDIS_LCON_EN	LOC_CONN	SEL_CP_EXT	DISCHRG_VBUS	CHRG_VBUS	DRV_VBUS
Reset	1	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 24. OtgControl register: bit description

Bit	Symbol	Description
15 to 12	-	reserved
11	OTG_SE0_EN	This bit is used by the host controller to send SE0 on remote connect. **0 —** no SE0 sent on remote connect detection **1 —** SE0 (bus reset) sent on remote connect detection **Remark:** This bit is normally set when the B-device goes into the b_wait_acon state (recommended sequence: LOC_CONN = 0 → DELAY → 50 µs → OTG_SE0_EN = 1 → SEL_HC_DC = 0) and is cleared when it comes out of the b_wait_acon state.
10	A_SRP_DET_EN	This bit is for the A-device only. If set, the A_SRP_DET bit in the OtgInterrupt register will be set on detecting an SRP event. **0 —** disable **1 —** enable
9	A_SEL_SRP	This bit is for the A-device to select a method to detect the SRP event (V_{BUS} pulsing or data line pulsing). **0 —** A-device responds to the V_{BUS} pulsing **1 —** A-device responds to the data line pulsing
8	SEL_HC_DC	This bit is used to select either the peripheral controller or the host controller that interfaces with the transceiver. **0 —** host controller SIE is connected to the OTG transceiver **1 —** peripheral controller SIE is connected to the OTG transceiver
7	LOC_PULLDN_DM	**0 —** disconnects the on-chip pull-down resistor on DM of the OTG port **1 —** connects the on-chip pull-down resistor on DM of the OTG port
6	LOC_PULLDN_DP	**0 —** disconnects the on-chip pull-down resistor on DP of the OTG port **1 —** connects the on-chip pull-down resistor on DP of the OTG port
5	A_RDIS_LCON_EN	This bit is for the A-device only. If set, the chip will automatically enable its pull-up resistor on DP on detecting a remote disconnect event. If cleared, the DP pull-up is controlled by the LOC_CONN bit. **0 —** disable **1 —** enable **Remark:** This bit is normally set when the A-device goes into the a_suspend state and is cleared when it comes out of the a_suspend state. The LOC_CONN bit must be set before clearing this bit.

Table 24. OtgControl register: bit description *...continued*

Bit	Symbol	Description
4	LOC_CONN	**0 —** disconnect the on-chip pull-up resistor on DP of the OTG port **1 —** connect the on-chip pull-up resistor on DP of the OTG port
3	SEL_CP_EXT	This bit is for the A-device only. This bit is used to choose the power source to drive V_{BUS}. **0 —** use on-chip charge pump to drive V_{BUS} **1 —** use external power source (5 V) to drive V_{BUS} **Remark:** When using the external power source, the $\overline{H\_PSW1}$ pin serves as the power switch that is controlled by the DRV_VBUS bit of this register.
2	DISCHRG_VBUS	This bit is for the B-device only. If set, it will enable a pull-down resistor on V_{BUS}, which will help to speed up discharging of V_{BUS} below session end threshold voltage. **0 —** disable **1 —** enable
1	CHRG_VBUS	This bit is for the B-device only. If set, it will charge V_{BUS} through a resistor. **0 —** disable charging V_{BUS} of the OTG port **1 —** enable charging V_{BUS} of the OTG port
0	DRV_VBUS	This bit is used to enable the on-chip charge pump or external power source to drive V_{BUS}. For the B-device, it shall not enable this bit at any time. **0 —** disable driving V_{BUS} of the OTG port **1 —** enable driving V_{BUS} of the OTG port

13.2 OtgStatus register (R: 67h)

Code (Hex): 67 — read only

Table 25. OtgStatus register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved						SE0_2MS	reserved
Reset	-	-	-	-	-	-	0	-
Access	-	-	-	-	-	-	R	-
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved		RMT_CONN	B_SESS_VLD	A_SESS_VLD	B_SESS_END	A_VBUS_VLD	ID_REG
Reset	-	-	0	0	0	1	0	1
Access	-	-	R	R	R	R	R	R

Table 26. OtgStatus register: bit description

Bit	Symbol	Description
15 to 10	-	reserved
9	SE0_2MS	**0 —** bus is in SE0 for less than 2 ms **1 —** bus is in SE0 for more than 2 ms

Table 26. OtgStatus register: bit description *...continued*

Bit	Symbol	Description
8 to 6	-	reserved
5	RMT_CONN	**0** — remote pull-up resistor disconnected **1** — remote pull-up resistor connected **Remark:** When the local pull-up resistor on the DP line is disabled, a 50 μs delay is applied before the RMT_CONN detection is enabled.
4	B_SESS_VLD	For the B-device (ID_REG = 1), this bit is a B-device session valid indicator (B_SESS_VLD). **0** — V_{BUS} is lower than VB_SESS_VLD **1** — V_{BUS} is higher than VB_SESS_VLD
3	A_SESS_VLD	For the A-device (ID_REG = 0), this bit is an A-device session valid indicator (A_SESS_VLD). **0** — V_{BUS} is lower than VA_SESS_VLD **1** — V_{BUS} is higher than VA_SESS_VLD
2	B_SESS_END	For the B-device (ID_REG = 1), this bit is a B-device session end indicator (B_SESS_END). **0** — V_{BUS} is higher than VB_SESS_END **1** — V_{BUS} is lower than VB_SESS_END
1	A_VBUS_VLD	For the A-device (ID_REG = 0), this bit is an A-device V_{BUS} valid indicator (A_VBUS_VLD). **0** — V_{BUS} is lower than VA_VBUS_VLD **1** — V_{BUS} is higher than VA_VBUS_VLD
0	ID_REG	This bit reflects the logic level of the ID pin. **0** — ID pin is LOW (mini-A plug is inserted in the mini-AB receptacle of the device) **1** — ID pin is HIGH (no plug or mini-B plug is inserted in the mini-AB receptacle of the device)

13.3 OtgInterrupt register (R/W: 68h/E8h)

Code (Hex): 68 — read

Code (Hex): E8 — write

Table 27. OtgInterrupt register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved					OTG_TMR_TIMEOUT	B_SE0_SRP	A_SRP_DET
Reset	-	-	-	-	-	0	0	0
Access	-	-	-	-	-	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	OTG_RESUME	OTG_SUSPND	RMT_CONN_C	B_SESS_VLD_C	A_SESS_VLD_C	B_SESS_END_C	A_VBUS_VLD_C	ID_REG_C
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 28. OtgInterrupt register: bit description

Bit	Symbol	Description
15 to 11	-	reserved
10	OTG_TMR_TIMEOUT	This bit is set whenever the OTG timer attains time-out. Writing logic 1 clears this bit. Writing logic 0 has no effect. **0** — no event **1** — OTG timer time-out
9	B_SE0_SRP	This bit is set whenever the device detects more than 2 ms of SE0. Writing logic 1 clears this bit. Writing logic 0 has no effect. **0** — no event **1** — bus has been in SE0 for more than 2 ms
8	A_SRP_DET	This bit is used to detect the Session Request Event (SRP) from the remote device. The SRP event can be either V_{BUS} pulsing or data line pulsing. Bit 9 (A_SEL_SRP) of the OtgControl register determines which SRP is selected. Writing logic 1 clears this bit. Writing logic 0 has no effect. **0** — no event **1** — SRP is detected
7	OTG_RESUME	This bit is used to detect a J to K state change when the device is in the suspend state. Writing logic 1 clears this bit. Writing logic 0 has no effect. **0** — no event **1** — a resume signal (J $\rightarrow$ K) is detected when the bus is in the suspend state
6	OTG_SUSPND	This bit is set whenever the OTG port goes into the suspend state (bus idle for > 3 ms). Write logic 1 to clear this bit. Writing logic 0 has no effect. **0** — no event **1** — suspend (bus idle for > 3 ms)
5	RMT_CONN_C	This bit is set whenever the RMT_CONN bit of the OtgStatus register changes. Write logic 1 to clear this bit. Writing logic 0 has no effect. **0** — no event **1** — RMT_CONN bit has changed
4	B_SESS_VLD_C	This bit is set whenever the B_SESS_VLD bit of the OtgStatus register changes. Write logic 1 to clear this bit. Writing logic 0 has no effect. **0** — no event **1** — bit B_SESS_VLD has changed
3	A_SESS_VLD_C	This bit is set whenever the A_SESS_VLD bit of the OtgStatus register changes. Write logic 1 to clear this bit. Writing logic 0 has no effect. **0** — no event **1** — bit A_SESS_VLD has changed

Table 28. OtgInterrupt register: bit description *...continued*

Bit	Symbol	Description
2	B_SESS_END_C	This bit is set whenever the B_SESS_END bit of the OtgStatus register changes. Write logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** no event **1 —** bit B_SESS_END has changed
1	A_VBUS_VLD_C	This bit is set whenever the A_VBUS_VLD bit of the OtgStatus register changes. Write logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** no event **1 —** bit A_VBUS_VLD has changed
0	ID_REG_C	This bit is set whenever the ID_REG bit of the OtgStatus register changes. This is an indication that the mini-A plug is inserted or removed (that is, the ID pin is shorted to ground or pulled HIGH). Write logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** no event **1 —** ID_REG bit has changed

13.4 OtgInterruptEnable register (R/W: 69h/E9h)

Code (Hex): 69 — read

Code (Hex): E9 — write

Table 29. OtgInterruptEnable register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved					OTG_TMR_IE	B_SE0_SRP_IE	A_SRP_DET_IE
Reset	-	-	-	-	-	0	0	0
Access	-	-	-	-	-	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	OTG_RESUME_IE	OTG_SUSPND_IE	RMT_CONN_IE	B_SESS_VLD_IE	A_SESS_VLD_IE	B_SESS_END_IE	A_VBUS_VLD_IE	ID_REG_IE
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 30. OtgInterruptEnable register: bit description

Bit	Symbol	Description
15 to 11	-	reserved
10	OTG_TMR_IE	Logic 1 enables interrupt when the OTG timer attains time-out. Logic 0 disables interrupt.
9	B_SE0_SRP_IE	Logic 1 enables interrupt on detecting the B_SE0_SRP status change. Logic 0 disables interrupt.
8	A_SRP_DET_IE	Logic 1 enables interrupt on detecting the SRP event. Logic 0 disables interrupt.
7	OTG_RESUME_IE	Logic 1 enables interrupt on detecting bus resume (J to K only) event. Logic 0 disables interrupt.

Table 30. OtgInterruptEnable register: bit description *…continued*

Bit	Symbol	Description
6	OTG_SUSPND_IE	Logic 1 enables interrupt on detecting the bus suspend status change. Logic 0 disables interrupt.
5	RMT_CONN_IE	Logic 1 enables interrupt on detecting the RMT_CONN status change. Logic 0 disables interrupt.
4	B_SESS_VLD_IE	Logic 1 enables interrupt on detecting the B_SESS_VLD status change. Logic 0 disables interrupt.
3	A_SESS_VLD_IE	Logic 1 enables interrupt on detecting the A_SESS_VLD status change. Logic 0 disables interrupt.
2	B_SESS_END_IE	Logic 1 enables interrupt on detecting the B_SESS_END status change. Logic 0 disables interrupt.
1	A_VBUS_VLD_IE	Logic 1 enables interrupt on detecting the A_VBUS_VLD status change. Logic 0 disables interrupt.
0	ID_REG_IE	Logic 1 enables interrupt on detecting the ID_REG status change. Logic 0 disables interrupt.

13.5 OtgTimer register (R/W: 6Ah/EAh)

Code (Hex): 6A — read

Code (Hex): EA — write

Table 31. OtgTimer register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	START_TMR	reserved						
Reset	0	-	-	-	-	-	-	-
Access	R/W	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	TMR_INIT_VALUE[23:16]							
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	TMR_INIT_VALUE[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	TMR_INIT_VALUE[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 32. OtgTimer register: bit description

Bit	Symbol	Description
31	START_TMR	This is the start or stop bit of the OTG timer. Writing logic 1 will cause the OTG timer to load TMR_INIT_VALUE into the counter and start to count. Writing logic 0 will stop the timer. This bit is automatically cleared when the OTG timer is timed out. **0 —** stop the timer **1 —** start the timer
30 to 24	-	reserved
23 to 0	TMR_INIT_VALUE[23:0]	These bits define the initial value used by the OTG timer. The timer interval is 0.01 ms. Maximum timer allowed is 167.772 s.

13.6 OtgAltTimer register (R/W: 6Ch/ECh)

Code (Hex): 6C — read

Code (Hex): EC — write

Table 33. OtgAltTimer register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	START_TMR	reserved						
Reset	0	-	-	-	-	-	-	-
Access	R/W	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	CURRENT_TIME[23:16]							
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	CURRENT_TIME[15:8]							
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	CURRENT_TIME[7:0]							
Reset	0	0	0	0	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 34. OtgAltTimer register: bit description

Bit	Symbol	Description
31	START_TMR	This is the start or stop bit of the OTG timer 2. Writing logic 1 will cause OTG timer 2 to start counting from 0. When the counter reaches FF FFFFh, this bit is auto-cleared (the counter is stopped). Writing logic 0 will stop the counting. If any bit of the OTGInterrupt register is set and the corresponding bit of the OtgInterruptEnable register is also set, this bit will be auto-cleared and the current value of the counter will be written to the CURRENT_TIME field. **0 —** stop the timer **1 —** start the timer
30 to 24	-	reserved
23 to 0	CURRENT_TIME	When read, these bits give the current value of the timer. The actual time is CURRENT_TIME × 0.01 ms.

14. Host controller registers

The host controller contains a set of on-chip control registers. These registers can be read or written by the Host Controller Driver (HCD). The operational registers are made compatible to Open Host Controller Interface (OHCI) operational registers. This enables the OHCI HCD to be easily ported to the ISP1362.

Reserved bits may be defined in future releases of this specification. To ensure interoperability, 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 desired bits 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 host controller register. When there is a write to set or clear the register, bits written to reserved fields must be logic 0.

As shown in Table 35, the offset locations (commands to read registers) of these operational registers (32-bit registers) are similar to those defined in the OHCI specification. The addresses, however, are equal to offset divided by 4.

Table 35. Host controller registers overview

Command (Hex)		Register	Width	Reference	Functionality
Read	Write				
00	not applicable	HcRevision	32	Section 14.1.1 on page 69	HC control and status registers
01	81	HcControl	32	Section 14.1.2 on page 69	
02	82	HcCommandStatus	32	Section 14.1.3 on page 71	
03	83	HcInterruptStatus	32	Section 14.1.4 on page 72	
04	84	HcInterruptEnable	32	Section 14.1.5 on page 73	
05	85	HcInterruptDisable	32	Section 14.1.6 on page 74	
0D	8D	HcFmInterval	32	Section 14.2.1 on page 75	HC frame counter registers
0E	8E	HcFmRemaining	32	Section 14.2.2 on page 76	
0F	8F	HcFmNumber	32	Section 14.2.3 on page 77	
11	91	HcLSThreshold	32	Section 14.2.4 on page 78	

Table 35. Host controller registers overview *...continued*

Command (Hex)		Register	Width	Reference	Functionality
Read	Write				
12	92	HcRhDescriptorA	32	Section 14.3.1 on page 79	HC root hub registers
13	93	HcRhDescriptorB	32	Section 14.3.2 on page 81	
14	94	HcRhStatus	32	Section 14.3.3 on page 82	
15	95	HcRhPortStatus[1]	32	Section 14.3.4 on page 83	
16	96	HcRhPortStatus[2]	32	Section 14.3.4 on page 83	
20	A0	HcHardwareConfiguration	16	Section 14.4.1 on page 87	HC DMA and interrupt control registers
21	A1	HcDMAConfiguration	16	Section 14.4.2 on page 88	
22	A2	HcTransferCounter	16	Section 14.4.3 on page 89	
24	A4	HcμPInterrupt	16	Section 14.4.4 on page 90	
25	A5	HcμPInterruptEnable	16	Section 14.4.5 on page 91	
27	not applicable	HcChipID	16	Section 14.5.1 on page 92	HC miscellaneous registers
28	A8	HcScratch	16	Section 14.5.2 on page 93	
Not applicable	A9	HcSoftwareReset	16	Section 14.5.3 on page 93	
2C	AC	HcBufferStatus	16	Section 14.6.1 on page 93	HC buffer RAM control registers
32	B2	HcDirectAddressLength	32	Section 14.6.2 on page 94	
45	C5	HcDirectAddressData	16	Section 14.6.3 on page 95	
30	B0	HcISTLBufferSize	16	Section 14.7.1 on page 95	ISO transfer registers
40	C0	HcISTL0BufferPort	16	Section 14.7.2 on page 96	
42	C2	HcISTL1BufferPort	16	Section 14.7.3 on page 96	
47	C7	HcISTLToggleRate	16	Section 14.7.4 on page 97	
33	B3	HcINTLBufferSize	16	Section 14.8.1 on page 97	interrupt transfer registers
43	C3	HcINTLBufferPort	16	Section 14.8.2 on page 97	
53	D3	HcINTLBlkSize	16	Section 14.8.3 on page 98	
17	not applicable	HcINTLPTDDoneMap	32	Section 14.8.4 on page 98	
18	98	HcINTLPTDSkipMap	32	Section 14.8.5 on page 99	
19	99	HcINTLLastPTD	32	Section 14.8.6 on page 99	
1A	not applicable	HcINTLCurrentActivePTD	16	Section 14.8.7 on page 100	
34	B4	HcATLBufferSize	16	Section 14.9.1 on page 100	aperiodic transfer registers
44	C4	HcATLBufferPort	16	Section 14.9.2 on page 100	
54	D4	HcATLBlkSize	16	Section 14.9.3 on page 101	
1B	not applicable	HcATLPTDDoneMap	32	Section 14.9.4 on page 101	
1C	9C	HcATLPTDSkipMap	32	Section 14.9.5 on page 102	
1D	9D	HcATLLastPTD	32	Section 14.9.6 on page 102	
1E	not applicable	HcATLCurrentActivePTD	16	Section 14.9.7 on page 102	
51	D1	HcATLPTDDoneThresholdCount	16	Section 14.9.8 on page 103	
52	D2	HcATLPTDDoneThresholdTimeOut	16	Section 14.9.9 on page 104	

14.1 HC control and status registers

14.1.1 HcRevision register (R: 00h)

The bit allocation of the HcRevision register is given in Table 36.

Code (Hex): 00 — read only

Table 36. HcRevision register: bit allocation

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

Table 37. HcRevision register: bit description

Bit	Symbol	Description
31 to 8	-	Reserved
7 to 0	REV[7:0]	**Revision:** This read-only field contains the Binary-Coded Decimal (BCD) representation of the version of the HCI specification that is implemented by this host controller.

14.1.2 HcControl register (R/W: 01h/81h)

The HcControl register defines operating modes for the host controller. The RWE bit is modified only by the HCD. Table 38 shows the bit allocation of the register.

Code (Hex): 01 — read

Code (Hex): 81 — write

Table 38. HcControl register: bit allocation

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

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

Table 39. 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 on detecting upstream resume signaling. When this bit and the ResumeDetected (RD) bit in HcInterruptStatus are set, a remote wake-up is signaled to the host system. Setting this bit has no impact on the generation of the hardware interrupt.
9	RWC	**RemoteWakeupConnected:** This bit indicates whether the host controller supports remote wake-up signaling. If remote wake-up is supported and used by the system, it is the responsibility of the system firmware to set this bit during POST. The host controller clears the bit on a hardware reset but does not alter it on 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 host controller has begun sending SOFs by reading the StartofFrame (SF) field of HcInterruptStatus. This field may be changed by the host controller only when it is in the USBSuspend state. The host controller may move from the USBSuspend state to the USBResume state after detecting the resume signaling from a downstream port. The host controller enters USBReset either by a software reset or by a hardware reset. The latter also resets the root hub and asserts subsequent reset signaling to downstream ports.
5 to 0	-	reserved

14.1.3 HcCommandStatus register (R/W: 02h/82h)

The HcCommandStatus register is a 4-byte register, and the bit allocation is given in Table 40. This register is used by the host controller to receive commands issued by the HCD, and it also reflects the current status of the host controller. To the HCD, it appears to be a 'write to set' register. The host controller 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 host controller without concern for corrupting previously issued commands. The HCD has normal read access to all bits.

The SchedulingOverrunCount (SOC) field indicates the number of frames with which the host controller has detected the scheduling overrun error. This occurs when the periodic list does not complete before the End-Of-Frame (EOF). When a scheduling overrun error is detected, the host controller increments the counter and sets the SchedulingOverrun (SO) field of the HcInterruptStatus register.

Code (Hex): 02 — read

Code (Hex): 82 — write

Table 40. HcCommandStatus register: bit allocation

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

Table 41. HcCommandStatus register: bit description

Bit	Symbol	Description
31 to 18	-	reserved
17 to 16	SOC[1:0]	**SchedulingOverrunCount:** This 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 the HCD to monitor any persistent scheduling problems.
15 to 1	-	reserved
0	HCR	**HostControllerReset:** This bit is set by the HCD to initiate a software reset to the host controller. Regardless of the functional state of the host controller, it moves to the USBSuspend state in which most of the operational registers are reset, except those stated otherwise. This bit is cleared by the host controller on completing the reset operation. The reset operation must be completed within 10 ms. This bit, when set, will not cause a reset to the root hub and no subsequent reset signaling will be asserted to its downstream ports.

14.1.4 HcInterruptStatus register (R/W: 03h/83h)

This register (bit allocation: Table 42) provides the status of the events that cause hardware interrupts. When an event occurs, the host controller sets the corresponding bit in this register. When a bit is set, a hardware interrupt is generated if the corresponding interrupt is enabled in the HcInterruptEnable register (see Section 14.1.5) and the MasterInterruptEnable (MIE) bit is set. The HCD must write logic 1 to the specific bits to clear the corresponding interrupt bits. The HCD cannot set any of these bits.

Code (Hex): 03 — read

Code (Hex): 83 — write

Table 42. HcInterruptStatus register: bit allocation

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

Table 43. HcInterruptStatus register: bit description

Bit	Symbol	Description
31 to 7	-	reserved
6	RHSC	**RootHubStatusChange:** This bit is set when any of the bits of HcRhPortStatus[NumberofDownstreamPort] has changed.
5	FNO	**FrameNumberOverflow:** This bit is set when the MSB of the HcFmNumber register (bit 15) changes from logic 0 to logic 1 or from logic 1 to logic 0.
4	UE	**UnrecoverableError:** This bit is set when the host controller detects a system error not related to the USB. The host controller should not proceed with any processing nor signaling before the system error is corrected. The HCD clears this bit after the host controller is reset. ST-NXP Wireless host controller interface: always set to logic 0.
3	RD	**ResumeDetected:** This bit is set when the host controller 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 the HCD sets the USBResume state.
2	SF	**StartOfFrame:** At the start of each frame, this bit is set by the host controller and an SOF is generated.
1	-	reserved
0	SO	**SchedulingOverrun:** This bit is set when the schedule is overrun for the current frame. A scheduling overrun also causes SchedulingOverrunCount (SOC) of HcCommandStatus to be incremented.

14.1.5 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 the following three conditions occur:

- A bit is set in the HcInterruptStatus register.
- The corresponding bit in the HcInterruptEnable register is set.
- The MasterInterruptEnable (MIE) bit is set.

Then, a hardware interrupt is requested on the host bus.

Writing logic 1 to a bit in the HcInterruptEnable 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. Table 44 contains the bit allocation of the register.

Code (Hex): 04 — read

Code (Hex): 84 — write

Table 44. HcInterruptEnable register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	MIE	reserved						
Reset	0	-	-	-	-	-	-	-
Access	R/W	-	-	-	-	-	-	-

Bit	23	22	21	20	19	18	17	16
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
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
Access	-	R/W	R/W	R/W	R/W	R/W	-	R/W

Table 45. HcInterruptEnable register: bit description

Bit	Symbol	Description
31	MIE	**MasterInterruptEnable** by the HCD: Logic 0 is ignored by the host controller. 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 because of root hub status change
5	FNO	**0** — ignore **1** — enable interrupt generation because of frame number overflow
4	UE	**0** — ignore **1** — enable interrupt generation because of unrecoverable error
3	RD	**0** — ignore **1** — enable interrupt generation because of resume detect
2	SF	**0** — ignore **1** — enable interrupt generation because of start-of-frame
1	-	reserved
0	SO	**0** — ignore **1** — enable interrupt generation because of scheduling overrun

14.1.6 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. Therefore, 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. Table 46 provides the bit allocation of the HcInterruptDisable register.

Code (Hex): 05 — read

Code (Hex): 85 — write

Table 46. HcInterruptDisable register: bit allocation

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

Table 47. HcInterruptDisable register: bit description

Bit	Symbol	Description
31	MIE	Logic 0 is ignored by the host controller. Logic 1 disables interrupt generation. This field is set after a hardware or software reset.
30 to 7	-	reserved
6	RHSC	**0** — ignore **1** — disable interrupt generation because of root hub status change
5	FNO	**0** — ignore **1** — disable interrupt generation because of frame number overflow
4	UE	**0** — ignore **1** — disable interrupt generation because of unrecoverable error
3	RD	**0** — ignore **1** — disable interrupt generation because of resume detect
2	SF	**0** — ignore **1** — disable interrupt generation because of start-of-frame
1	-	reserved
0	SO	**0** — ignore **1** — disable interrupt generation because of scheduling overrun

14.2 HC frame counter registers

14.2.1 HcFmInterval register (R/W: 0Dh/8Dh)

The HcFmInterval register (bit allocation: Table 48) contains a 14-bit value that indicates the bit time interval in a frame between two consecutive SOFs. In addition, it contains a 15-bit value, indicating the full-speed maximum packet size that the host controller may transmit or receive without causing a scheduling overrun. The HCD may carry out minor

adjustments on FrameInterval by writing a new value over the present one at each SOF. This provides the programmability necessary for the host controller to synchronize with an external clocking resource and to adjust any unknown local clock offset.

Code (Hex): 0D — read

Code (Hex): 8D — write

Table 48. 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	R/W	R/W	R/W	R/W	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	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		FI[13:8]					
Reset	-	-	1	0	1	1	1	0
Access	-	-	R/W	R/W	R/W	R/W	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	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 49. HcFmInterval register: bit description

Bit	Symbol	Description
31	FIT	**FrameIntervalToggle:** The HCD toggles this bit whenever it loads a new value to FrameInterval.
30 to 16	FSMPS [14:0]	**FSLargestDataPacket:** Specifies a value that 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 that can be sent or received by the host controller 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 11999. The HCD must store the current value of this field before resetting the host controller. Setting the HostControllerReset (HCR) field of the HcCommandStatus register causes the host controller to reset this field to its nominal value. The HCD may choose to restore the stored value on completing the reset sequence.

14.2.2 HcFmRemaining register (R/W: 0Eh/8Eh)

The HcFmRemaining register is a 14-bit down counter, showing the bit time remaining in the current frame. The bit allocation is given in Table 50.

Code (Hex): 0E — read

Code (Hex): 8E — write

Table 50. HcFmRemaining register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	FRT	reserved						
Reset	0	-	-	-	-	-	-	-
Access	R/W	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved		FR[13:8]					
Reset	-	-	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	FR[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 51. HcFmRemaining register: bit description

Bit	Symbol	Description
31	FRT	**FrameRemainingToggle:** This bit is loaded from the FrameIntervalToggle (FIT) field of HcFmInterval whenever FrameRemaining (FR) reaches 0. This bit is used by the HCD for synchronization between FrameInterval (FI) and FrameRemaining (FR).
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 (FI) value specified in HcFmInterval at the next bit time boundary. When entering the USBOperational state, the host controller reloads it with the content of the FrameInterval (FI) part of the HcFmInterval register and uses the updated value from the next SOF.

14.2.3 HcFmNumber register (R/W: 0Fh/8Fh)

The HcFmNumber register is a 16-bit counter, and the bit allocation is given in Table 52. It provides a timing reference for events happening in the host controller 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

Code (Hex): 8F — write

Table 52. HcFmNumber register: bit allocation

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

Table 53. 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. The value will be rolled over to 0h after FFFFh. When the USBOperational state is entered, this is automatically incremented.

14.2.4 HcLSThreshold register (R/W: 11h/91h)

The HcLSThreshold register contains an 11-bit value. This value is used by the host controller to determine whether to start a transfer of a maximum of 8 bytes LS packet before the EOF. The HCD is not allowed to change this value. Table 54 shows the bit allocation of the register.

Code (Hex): 11 — read

Code (Hex): 91 — write

Table 54. HcLSThreshold register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8
Symbol	reserved					LST[10:8]		
Reset	-	-	-	-	-	1	1	0
Access	-	-	-	-	-	R/W	R/W	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	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 55. 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 (FR) field before a low-speed transaction is initiated. The transaction is started only if FrameRemaining (FR) ≥ this field. The value is calculated by the HCD. The HCD must consider transmission and set-up overhead, while calculating this value.

14.3 HC root hub registers

All registers included in this partition are dedicated to the USB root hub, which is an integral part of the host controller although it is functionally a separate entity. The HCD emulates USB Driver (USBD) accesses to the root hub by using a register interface. The HCD maintains many USB-defined hub features that are not required to be supported in the 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 address of the root hub device and other trivial operations that are better suited to the software than to the hardware.

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 double word. These registers are only written during initialization to correspond with the system implementation. The HcRhDescriptorA and HcRhDescriptorB registers can be read or written, regardless of the USB states of the host controller. You can write to HcRhStatus and HcRhPortStatus only when the host controller is in the USBOperational state.

14.3.1 HcRhDescriptorA register (R/W: 12h/92h)

The HcRhDescriptorA register is the first of two registers describing the characteristics of the root hub. The bit allocation is given in Table 56.

Code (Hex): 12 — read

Code (Hex): 92 — write

Table 56. HcRhDescriptorA register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	POTPGT[7:0]							
Reset	1	1	1	1	1	1	1	1
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	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved			NOCP	OCPM	DT	NPS	PSM
Reset	-	-	-	0	1	0	0	1
Access	-	-	-	R/W	R/W	R	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved						NDP[1:0]	
Reset	-	-	-	-	-	-	1	0
Access	-	-	-	-	-	-	R	R

Table 57. HcRhDescriptorA register: bit description

Bit	Symbol	Description
31 to 24	POTPGT [7:0]	**PowerOnToPowerGoodTime:** This byte specifies the duration HCD must 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 root hub ports are reported. When this bit is cleared, the OverCurrentProtectionMode (OCPM) field specifies global or per-port reporting. **0 —** overcurrent status is collectively reported for all downstream ports. **1 —** no overcurrent reporting supported.
11	OCPM	**OverCurrentProtectionMode:** This bit describes how the overcurrent status for root hub ports are reported. At reset, this field should reflect the same mode as PowerSwitchingMode. This field is valid only if the NoOverCurrentProtection (NOCP) field is cleared. **0 —** overcurrent status is collectively reported 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 as 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 PowerSwitchingMode (PSM) bit specifies global or per-port switching. **0 —** Ports are power switched. **1 —** Ports are always powered on when the host controller is powered on.

Table 57. HcRhDescriptorA register: bit description *...continued*

Bit	Symbol	Description
8	PSM	**PowerSwitchingMode:** This bit is used to specify how the power switching of Root Hub ports is controlled. It is implementation specific. This field is valid only if the NoPowerSwitching (NPS) field is cleared. **0 —** All ports are powered at the same time. **1 —** Each port is individually powered. This mode allows port power to be controlled by either the global switch or per-port switching. If the PortPowerControlMask (PPCM) bit 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]	**NumberofDownstreamPort:** These bits specify the number of downstream ports supported by the root hub. The ISP1362 supports two ports and therefore, the value is 2.

14.3.2 HcRhDescriptorB register (R/W: 13h/93h)

The HcRhDescriptorB register is the second of two registers describing the characteristics of the root hub. These fields are written during initialization to correspond to the system implementation. Table 58 shows the bit allocation of the register.

Code (Hex): 13 — read

Code (Hex): 93 — write

Table 58. HcRhDescriptorB register: bit allocation

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

Table 59. 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 power state of the port 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 2 —** ganged-power mask on port 2 **Bit 1 —** ganged-power mask on port 1 **Bit 0 —** reserved
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 2 —** device attached to port 2 **Bit 1 —** device attached to port 1 **Bit 0 —** reserved

14.3.3 HcRhStatus register (R/W: 14h/94h)

The HcRhStatus register is divided into two parts. The lower word of a double-word represents the hub status field, and the upper word represents the hub status change field. Reserved bits should always be written as logic 0. See Table 60 for bit allocation of the register.

Code (Hex): 14 — read

Code (Hex): 94 — write

Table 60. HcRhStatus register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	CRWE	reserved						
Reset	0	-	-	-	-	-	-	-
Access	W	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	reserved						CCIC	LPSC
Reset	-	-	-	-	-	-	0	0
Access	-	-	-	-	-	-	R/W	R/W
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	DRWE	reserved						
Reset	0	-	-	-	-	-	-	-
Access	R/W	-	-	-	-	-	-	-
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	reserved						OCI	LPS
Reset	-	-	-	-	-	-	0	0
Access	-	-	-	-	-	-	R	R/W

Table 61. HcRhStatus register: bit description

Bit	Symbol	Description
31	CRWE	On write **ClearRemoteWakeupEnable:** Writing logic 1 clears DeviceRemoteWakeupEnable (DRWE). Writing logic 0 has no effect.
30 to 18	-	reserved
17	CCIC	**OverCurrentIndicatorChange:** This bit is set by the hardware when a change has occurred to the OverCurrentIndicator (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), logic 1 is written to this bit to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose PortPowerControlMask bit 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 from 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 DeviceRemoteWakeupEnable. 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 cleared, 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), logic 1 is written to this bit 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.

14.3.4 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. NumberofDownstreamPort represents the number of HcRhPortStatus registers that are implemented in the hardware. The lower word is used to reflect the port status, whereas the upper word reflects status change bits. Some status bits are implemented with special write behavior. Reserved bits should always be written logic 0. The bit allocation of the HcRhPortStatus[1:2] register is given in Table 62.

Code (Hex): [1] = 15, [2] = 16 — read

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

Table 62. HcRhPortStatus[1:2] register: bit allocation

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

Bit	23	22	21	20	19	18	17	16
Symbol	reserved			PRSC	OCIC	PSSC	PESC	CSC
Reset	-	-	-	0	0	0	0	0
Access	-	-	-	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
Access	-	-	-	-	-	-	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
Access	-	-	-	R/W	R/W	R/W	R/W	R/W

Table 63. 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 the root hub changes the PortOverCurrentIndicator (POCI) bit. The HCD writes logic 1 to clear this bit. Writing logic 0 has no effect. **0 —** no change in PortOverCurrentIndicator (POCI) **1 —** PortOverCurrentIndicator (POCI) has changed
18	PSSC	**PortSuspendStatusChange:** This bit is set when the full resume sequence is complete. This sequence includes the 20 ms 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 PortResetStatusChange is set. **0 —** resume is not completed **1 —** resume is completed
17	PESC	**PortEnableStatusChange:** This bit is set when the hardware events cause the PortEnableStatus (PES) bit to be cleared. Changes from the 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 (PES) **1 —** change in PortEnableStatus (PES)

Table 63. HcRhPortStatus[1:2] register: bit description *…continued*

Bit	Symbol	Description
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 (CCS) is cleared when a SetPortReset, SetPortEnable or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status because these writes should not occur if the port is disconnected. **0 —** no change in CurrentConnectStatus (CCS) **1 —** change in CurrentConnectStatus (CCS) **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	On 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 cleared, a full-speed device is attached to this port. This field is valid only when CurrentConnectStatus (CCS) is set. **0 —** full-speed device attached **1 —** low-speed device attached On write **ClearPortPower:** The HCD clears the PortPowerStatus (PPS) bit by writing logic 1 to this bit. Writing logic 0 has no effect.
8	PPS	On 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. PowerSwitchingMode (PCM) and PortPowerControlMask[NDP] (PPCM[NDP]) determine which power control switches are enabled. In global switching mode (PowerSwitchingMode = 0), only the Set/ClearGlobalPower command controls this bit. In the per-port power switching (PowerSwitchingMode = 1), if the PortPowerControlMask[NDP] (PPCM[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 (CCS), PortEnableStatus (PES), PortSuspendStatus (PSS) and PortResetStatus (PRS) should be reset. **0 —** port power is off **1 —** port power is on On write **SetPortPower:** The HCD writes logic 1 to set the PortPowerStatus (PPS) 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	On 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 (PRSC) is set. This bit cannot be set if CurrentConnectStatus (CCS) is cleared. **0 —** port reset signal is not active **1 —** port reset signal is active On write **SetPortReset:** The HCD sets the port reset signaling by writing logic 1 to this bit. Writing logic 0 has no effect. If CurrentConnectStatus (CCS) is cleared, this write does not set PortResetStatus (PRS) but instead sets ConnectStatusChange (CSC). This informs the driver that it attempted to reset a disconnected port.

Table 63. HcRhPortStatus[1:2] register: bit description *…continued*

Bit	Symbol	Description
3	POCI	On 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 On write **ClearSuspendStatus:** The HCD writes logic 1 to initiate a resume. Writing logic 0 has no effect. A resume is initiated only if PortSuspendStatus (PSS) is set.
2	PSS	On read **PortSuspendStatus:** This bit indicates whether the port is suspended or is in the resume sequence. It is set by a PortSuspendStatus write and cleared when PortSuspendStatusChange (PSSC) is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus (CCS) is cleared. This bit is also cleared when PortResetStatusChange (PRSC) is set at the end of the port reset or when the host controller is placed in the USBResume state. If an upstream resume is in progress, it should propagate to the host controller. **0 —** port is not suspended **1 —** port is suspended On write **SetPortSuspend:** The HCD sets the PortSuspendStatus (PSS) bit by writing logic 1 to this bit. Writing logic 0 has no effect. If CurrentConnectStatus (CCS) is cleared, this write does not set PortSuspendStatus (PSS); instead it sets ConnectStatusChange (CSC). This informs the driver that it attempted to suspend a disconnected port.
1	PES	On 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 PortEnableStatusChange to be set. The HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. This bit cannot be set when CurrentConnectStatus (CCS) is cleared. This bit is also set, if it is not already, at the completion of a port reset when PortResetStatusChange is set or port suspend when PortSuspendStatusChange is set. **0 —** port is disabled **1 —** port is enabled On write **SetPortEnable:** The HCD sets PortEnableStatus (PES) by writing logic 1. Writing logic 0 has no effect. If CurrentConnectStatus (CCS) is cleared, this write does not set PortEnableStatus (PES), but instead sets ConnectStatusChange (CSC). This informs the driver that it attempted to enable a disconnected port.
0	CCS	On read **CurrentConnectStatus:** This bit reflects the current state of the downstream port. **0 —** no device connected **1 —** device connected On write **ClearPortEnable:** The HCD writes logic 1 to this bit to clear the PortEnableStatus (PES) bit. Writing logic 0 has no effect. CurrentConnectStatus (CSC) is not affected by any write. **Remark:** This bit always reads logic 1 when the attached device is nonremovable (DeviceRemovable[NDP]).

14.4 HC DMA and interrupt control registers

14.4.1 HcHardwareConfiguration register (R/W: 20h/A0h)

The bit allocation of the HcHardwareConfiguration register is given in Table 64.

Code (Hex): 20 — read

Code (Hex): A0 — write

Table 64. HcHardwareConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	Disable Suspend_ Wakeup	Global Power Down	Connect PullDown _DS2	Connect PullDown _DS1	Suspend ClkNotStop	AnalogOC Enable	OneINT	DACK Mode
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	OneDMA	DACKInput Polarity	DREQ Output Polarity	DataBusWidth[1:0]		Interrupt Output Polarity	Interrupt PinTrigger	InterruptPin Enable
Reset	0	0	1	0	1	0	0	0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 65. HcHardwareConfiguration register: bit description

Bit	Symbol	Description
15	DisableSuspend_Wakeup	This bit when set to logic 1 disables the function of the D_SUSPEND/D_WAKEUP and H_SUSPEND/H_WAKEUP pins. Therefore, these pins will always remain HIGH and pulling them LOW does not wake-up the host controller and the peripheral controller.
14	GlobalPowerDown	Set this bit to logic 1 to reduce power consumption of the OTG ATX in suspend mode.
13	ConnectPullDown_DS2	**0 —** disconnect built-in pull-down resistors on H_DM2 and H_DP2 **1 —** connect built-in pull-down resistors on H_DM2 and H_DP2 for the downstream port 2 **Remark:** Port 2 is always used as a host port.
12	ConnectPullDown_DS1	**0 —** disconnect built-in pull-down resistors on OTG_DM1 and OTG_DP1 **1 —** connect built-in pull-down resistors on OTG_DM1 and OTG_DP1 **Remark:** This bit is effective only when port 1 is configured as the host port (the OTGMODE pin is HIGH, and the ID pin is LOW). When port 1 is configured as the OTG port, (the OTGMODE pin is LOW), the pull-down resistors on OTG_DM1 and OTG_DP1 are controlled by the LOC_PULL_DN_DP and LOC_PULL_DN_DM bits of the OtgControl register.
11	SuspendClkNotStop	**0 —** clock can be stopped when suspended **1 —** clock cannot be stopped when suspended

Table 65. HcHardwareConfiguration register: bit description *…continued*

Bit	Symbol	Description
10	AnalogOCEnable	**0 —** use external overcurrent detection; digital input **1 —** use on-chip overcurrent detection; analog input
9	OneINT	**0 —** host controller interrupt routed to INT1, peripheral controller interrupt routed to INT2 **1 —** host controller and peripheral controller interrupts routed to INT1 only, INT2 is unused
8	DACKMode	**0 —** normal operation; $\overline{\text{DACK1}}$ is used with read and write signals **1 —** reserved
7	OneDMA	**0 —** host controller DMA request and acknowledge are routed to DREQ1 and $\overline{\text{DACK1}}$, peripheral controller DMA request and acknowledge are routed to DREQ2 and $\overline{\text{DACK2}}$ **1 —** host controller and peripheral controller DMA requests and acknowledges are routed to DREQ1 and $\overline{\text{DACK1}}$; DREQ2 and $\overline{\text{DACK2}}$ unused
6	DACKInputPolarity	**0 —** $\overline{\text{DACK1}}$ is active LOW **1 —** $\overline{\text{DACK1}}$ is active HIGH
5	DREQOutputPolarity	**0 —** DREQ1 is active LOW **1 —** DREQ1 is active HIGH
4 to 3	DataBusWidth[1:0]	**01 —** microprocessor interface data bus width is 16 bits **Others —** reserved
2	InterruptOutputPolarity	**0 —** INT1 interrupt is active LOW; power-up value **1 —** INT1 interrupt is active HIGH
1	InterruptPinTrigger	**0 —** INT1 interrupt is level-triggered; power-up value **1 —** INT1 interrupt is edge-triggered
0	InterruptPinEnable	**0 —** power-up value **1 —** global interrupt pin INT1 is enabled; this bit should be used with the HcμPInterruptEnable register to enable pin INT1

14.4.2 HcDMAConfiguration register (R/W: 21h/A1h)

Table 66 contains the bit allocation of the HcDMAConfiguration register.

Code (Hex): 21 — read

Code (Hex): A1 — write

Table 66. HcDMAConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	DMACounter Enable	BurstLen[1:0]		DMA Enable	Buffer_Type_Select[2:0]			DMARead WriteSelect
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 67. HcDMAConfiguration register: bit description

Bit	Symbol	Description
15 to 8	-	reserved
7	DMACounterEnable	**0 —** reserved **1 —** DMA counter is enabled. Once the counter is enabled, the HCD must initialize the HcTransferCounter register to a non-zero value for DREQ to be raised after the DMAEnable bit is set to HIGH.
6 to 5	BurstLen[1:0]	**00 —** single-cycle burst DMA **01 —** 4-cycle burst DMA **10 —** 8-cycle burst DMA **11 —** reserved I/O bus with 32-bit data path width supports only single and four cycle DMA burst.
4	DMAEnable	**0 —** DMA is disabled **1 —** DMA is enabled This bit must be reset when the DMA transfer is completed.
3 to 1	Buffer_Type_Select[2:0]	See Table 68.
0	DMAReadWriteSelect	**0 —** read from the buffer memory of the host controller **1 —** write to the buffer memory of the host controller

Table 68. Buffer_Type_Select[2:0]: bit description

Bit 3	Bit 2	Bit 1	Buffer Type
0	0	0	ISTL0 (default)
0	0	1	ISTL1
0	1	0	INTL
0	1	1	ATL
1	X	X	direct addressing

14.4.3 HcTransferCounter register (R/W: 22h/A2h)

Regardless of PIO or DMA data transfer modes, this register is used to initialize the number of bytes to be transferred to or from the ISTL, INTL or ATL buffer RAM. For the count value loaded in the register to take effect, the HCD is required to set bit 7 of the HcDMAConfiguration register to logic 1. When the count value has reached, the host controller must generate an internal EOT signal to set bit 2 of the HcμPInterrupt register, AllEOTInterrupt, and update the HcBufferStatus register. The bit allocation of the HcTransferCounter register is given in Table 69.

Code (Hex): 22 — read

Code (Hex): A2 — write

Table 69. HcTransferCounter register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	CounterValue[15:0]	R/W	0000h	Number of data bytes to be read from or written to the buffer RAM.

14.4.4 HcμPInterrupt register (R/W: 24h/A4h)

All the bits in this register are active at power-on reset. None of the active bits, however, will cause an interrupt on the interrupt pin (INT1), unless they are set by the respective bits in the HcμPInterruptEnable register and bit 0 of the HcHardwareConfiguration register is also set.

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.

The bit allocation of the HcμPInterrupt register is given in Table 70.

Code (Hex): 24 — read

Code (Hex): A4 — write

Table 70. HcμPInterrupt register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved						OTG_IRQ	ATL_IRQ
Reset	-	-	-	-	-	-	0	0
Access	-	-	-	-	-	-	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	INTL_IRQ	ClkReady	HC Suspended	OPR_Reg	AllEOT Interrupt	ISTL1_ INT	ISTL0_ INT	SOF_INT
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 71. HcμPInterrupt register: bit description

Bit	Symbol	Description
15 to 10	-	reserved
9	OTG_IRQ	**0 —** no event **1 —** The OTG interrupt event must read the OtgInterrupt register to get the cause of the interrupt.
8	ATL_IRQ	**0 —** no event **1 —** Count value of the HcATLPTDDoneThresholdCount register or the time-out value of the HcATLPTDDoneThresholdTimeOut register has reached. The microprocessor is required to read HcATLPTDDoneMap to check the PTDs that have completed their transactions.
7	INTL_IRQ	**0 —** no event **1 —** The host controller has detected the last PTD, and there is at least one interrupt transaction that has received ACK from the device. The microprocessor is required to read HcINTLPTDDoneMap to check the PTDs that have received ACK from the device.
6	ClkReady	**0 —** no event **1 —** The host controller has awakened from the suspend state, and its internal clock has turned on again.

Table 71. HcμPInterrupt register: bit description *...continued*

Bit	Symbol	Description
5	HC Suspended	**0 —** no event **1 —** The host controller has been suspended and no USB activities are sent from the microprocessor for each ms. The microprocessor can suspend the host controller by setting bits 6 and 7 of the HcControl register to logic 1. Once the host controller is suspended, no SOF needs to be sent to the devices connected to downstream ports.
4	OPR_Reg	**0 —** no event **1 —** A host controller operation has caused a hardware interrupt. It is necessary for the HCD to read the HcInterruptStatus register to determine the cause of the interrupt.
3	AllEOT Interrupt	**0 —** no event **1 —** Data transfer has been completed by using the PIO transfer or the DMA transfer. This bit is set either when the value of the HcTransferCounter register has reached zero, or the EOT pin of the host controller is triggered by an external signal.
2	ISTL1_ INT	**0 —** no event **1 —** The transaction of the last PTD stored in the ISTL1 buffer has been completed. The microprocessor is required to read data from the ISTL1 buffer. The HCD must first read the HcBufferStatus register to check the status of the ISTL1 buffer, before reading data to the microprocessor.
1	ISTL0_ INT	**0 —** no event **1 —** The transaction of the last PTD stored in the ISTL0 buffer has been completed. The microprocessor is required to read data from the ISTL0 buffer. The HCD must first read the HcBufferStatus register to check the status of the ISTL0 buffer, before reading data to the microprocessor.
0	SOF_INT	**0 —** no event **1 —** The host controller is in the SOF state and it indicates the start of a new frame. The HCD must first read the HcBufferStatus register to check the status of the ISTL buffer, before reading data to the microprocessor. For the microprocessor to perform the DMA transfer of ISO data from or to the ISTL buffer, the host controller must first initialize the HcDMAConfiguration register.

14.4.5 HcμPInterruptEnable register (R/W: 25h/A5h)

Bits 9 to 0 in this register are the same as those in the HcμPInterrupt register. The bits in this register are used together with bit 0 of the HcHardwareConfiguration register to enable or disable the bits in the HcμPInterrupt register.

At power-on, all the bits in this register are masked with logic 0. This means no interrupt request output on interrupt pin INT1 can be generated. When a bit is set to logic 1, the interrupt for that bit is enabled.

The bit allocation of the register is given in Table 72.

Code (Hex): 25 — read

Code (Hex): A5 — write

Table 72. HcμPInterruptEnable register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved						OTG_IRQ_ Interrupt Enable	ATL_IRQ_ Interrupt Enable
Reset	-	-	-	-	-	-	0	0
Access	-	-	-	-	-	-	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	INTL_IRQ_ Interrupt Enable	ClkReady	HC Suspended Enable	OPR Interrupt Enable	EOT Interrupt Enable	ISTL1 Interrupt Enable	ISTL0 Interrupt Enable	SOF Interrupt Enable
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 73. HcμPInterruptEnable register: bit description

Bit	Symbol	Description
15 to 10	-	reserved
9	OTG_IRQ_InterruptEnable	**0** — power-up value **1** — enables the OTG_IRQ interrupt
8	ATL_IRQ_InterruptEnable	**0** — power-up value **1** — enables the ATL_IRQ interrupt
7	INTL_IRQ_InterruptEnable	**0** — power-up value **1** — enables the INT_IRQ interrupt
6	ClkReady	**0** — power-up value **1** — enables the ClkReady interrupt
5	HCSuspendedEnable	**0** — power-up value **1** — enables the host controller suspended interrupt
4	OPRInterruptEnable	**0** — power-up value **1** — enables the 32-bit operational register's interrupt
3	EOTInterruptEnable	**0** — power-up value **1** — enables the EOT interrupt
2	ISTL1InterruptEnable	**0** — power-up value **1** — enables the ISTL1 interrupt
1	ISTL0InterruptEnable	**0** — power-up value **1** — enables the ISTL0 interrupt
0	SOFInterruptEnable	**0** — power-up value **1** — enables the SOF interrupt

14.5 HC miscellaneous registers

14.5.1 HcChipID register (R: 27h)

This register contains the ID of the ISP1362. The upper byte identifies the product name (here 36h stands for the ISP1362). The lower byte indicates the revision number of the product, including engineering samples. Table 74 contains the bit description of the register.

Code (Hex): 27 — read only

Table 74. HcChipID register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	CHIPID[15:0]	R	3630h	chip ID of the ISP1362

14.5.2 HcScratch register (R/W: 28h/A8h)

This register is for the HCD to save and restore values when required. The bit description is given in Table 75.

Code (Hex): 28 — read

Code (Hex): A8 — write

Table 75. HcScratch register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	SCRATCH[15:0]	R/W	0000h	scratch register value

14.5.3 HcSoftwareReset register (W: A9h)

This register provides a means for the software reset of the host controller. To reset the host controller, the HCD must write a reset value of F6h to this register. On receiving this reset value, the host controller resets all the host controller and OTG registers, except its buffer memory.

Table 76 contains the bit description of the register.

Code (Hex): A9 — write only

Table 76. HcSoftwareReset register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	ResetValue[15:0]	W	0000h	Writing a reset value of F6h causes the host controller to reset all the host controller and OTG registers, except its buffer memory

14.6 HC buffer RAM control registers

14.6.1 HcBufferStatus register (R/W: 2Ch/ACh)

The bit allocation of the HcBufferStatus register is given in Table 77.

Code (Hex): 2C — read

Code (Hex): AC — write

Table 77. HcBufferStatus register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved					PairedPTD PingPong	ISTL1 BufferDone	ISTL0 BufferDone
Reset	-	-	-	-	-	0	0	0
Access	-	-	-	-	-	R	R	R

Bit	7	6	5	4	3	2	1	0
Symbol	reserved	ISTL1_ Active Status	ISTL0_ Active Status	Reset_HW PingPong Reg	ATL_Active	INTL_ Active	ISTL1 BufferFull	ISTL0 BufferFull
Reset	-	0	0	0	0	0	0	0
Access	-	R	R	R/W	R/W	R/W	R/W	R/W

Table 78. HcBufferStatus register: bit description

Bit	Symbol	Description
15 to 11	-	reserved
10	PairedPTDPingPong	**0 —** Ping of the paired PTD in ATL is active. **1 —** Pong of the paired PTD in ATL is active.
9	ISTL1BufferDone	**0 —** The ISTL1 buffer has not yet been read by the host controller. **1 —** The ISTL1 buffer has been read by the host controller.
8	ISTL0BufferDone	**0 —** The ISTL0 buffer has not yet been read by the host controller. **1 —** The ISTL0 buffer has been read by the host controller.
7	-	reserved
6	ISTL1_ActiveStatus	**0 —** The ISTL1 buffer is not accessed by the slave host. **1 —** The ISTL1 buffer is accessed by the slave host.
5	ISTL0_ActiveStatus	**0 —** The ISTL0 buffer is not accessed by the slave host. **1 —** The ISTL0 buffer is accessed by the slave host.
4	Reset_HWPingPong Reg	**0 to 1 —** Resets the internal hardware ping pong register to 0 when ATL_Active is 0. The hardware ping pong register can be read from bit 10 of this register. **1 to 0 —** Has no effect.
3	ATL_Active	**0 —** The host controller does not process the ATL buffer. **1 —** The host controller processes the ATL buffer.
2	INTL_Active	**0 —** The host controller does not process the INTL buffer. **1 —** The host controller processes the INTL buffer.
1	ISTL1BufferFull	**0 —** The host controller does not process the ISTL1 buffer. **1 —** The host controller processes the ISTL1 buffer.
0	ISTL0BufferFull	**0 —** The host controller does not process the ISTL0 buffer. **1 —** The host controller processes the ISTL0 buffer.

14.6.2 HcDirectAddressLength register (R/W: 32h/B2h)

The HcDirectAddressLength register is used for direct addressing of the ISTL, INTL or ATL buffers. This register specifies the starting address of the buffer and byte count of data to be addressed. Therefore, it allows the programmer to randomly access the buffer. The bit allocation of the register is given in Table 79.

Code (Hex): 32 — read

Code (Hex): B2 — write

Table 79. HcDirectAddressLength register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	DataByteCount[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	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	DataByteCount[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
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	reserved	BufferStartAddress[14:8]						
Reset	0	0	0	0	0	0	0	0
Access	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	BufferStartAddress[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 80. HcDirectAddressLength register: bit description

Bit	Symbol	Description
31 to 16	DataByteCount[15:0]	Total number of bytes to be accessed.
15	-	reserved
14 to 0	BufferStartAddress[14:0]	The starting address of the buffer to access data.

14.6.3 HcDirectAddressData register (R/W: 45h/C5h)

This is a data port for the HCD to access the ISTL, INTL or ATL buffers under direct addressing mode. Table 81 contains the bit description of the register.

Code (Hex): 45 — read

Code (Hex): C5 — write

Table 81. HcDirectAddressData register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	DataWord [15:0]	R/W	0000h	The data port to access the ISTL, INTL or ATL buffers. The address of the buffer and byte count of the data must be specified in the HcDirectAddressLength register.

14.7 Isochronous (ISO) transfer registers

14.7.1 HcISTLBufferSize register (R/W: 30h/B0h)

This register requires you to allocate the size of the buffer to be used for ISO transactions. The buffer size specified in the register is applied to the ISTL0 and ISTL1 buffers. Therefore, ISTL0 and ISTL1 always have the same buffer size.

Table 82 shows the bit description of the register.

Code (Hex): 30 — read

Code (Hex): B0 — write

Table 82. HcISTLBufferSize register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	ISTLBufferSize[15:0]	R/W	0000h	The size of the buffer to be used for ISO transactions and must be specified in bytes.

14.7.2 HcISTL0BufferPort register (R/W: 40h/C0h)

In addition to the HcDirectAddressData register, the ISP1362 provides this register to act as another data port to access the ISTL0 buffer. The starting address to access the buffer is always fixed at 0000h. Therefore, random access of the ISTL0 buffer is not allowed. The bit description of the register is given in Table 83.

Code (Hex): 40 — read

Code (Hex): C0 — write

Table 83. HcISTL0BufferPort register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	DataWord[15:0]	R/W	0000h	Data in the ISTL0 buffer to be accessed through this data port.

The HCD is first required to initialize the HcTransferCounter register with the byte count to be transferred and check the HcBufferStatus register. The HCD then sends the command (40h to read from the ISTL0 buffer, and C0h to write to the ISTL0 buffer) to the host controller through the I/O port of the microprocessor. After the command is sent, the HCD starts reading data from the ISTL0 buffer or writing data to the ISTL0 buffer. While the HCD is accessing the buffer, the buffer pointer of ISTL0 also automatically increases. When the pointer has reached the initialized byte count of the HcTransferCounter register, the host controller sets the AllEOTInterrupt bit of the HcμPInterrupt register to logic 1 and updates the HcBufferStatus register.

14.7.3 HcISTL1BufferPort register (R/W: 42h/C2h)

In addition to the HcDirectAddressData register, the ISP1362 provides this register to act as another data port to access the ISTL1 buffer. The starting address to access the buffer is always fixed at 0000h. Therefore, random access of the ISTL1 buffer is not allowed. The bit description of the register is given in Table 84.

Code (Hex): 42 — read

Code (Hex): C2 — write

Table 84. HcISTL1BufferPort register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	DataWord[15:0]	R/W	0000h	Data in the ISTL1 buffer to be accessed through this data port.

The HCD is first required to initialize the HcTransferCounter register with the byte count to be transferred and check the HcBufferStatus register. The HCD then sends the command (42h to read from the ISTL1 buffer, and C2h to write to the ISTL1 buffer) to the host controller through the I/O port of the microprocessor. After the command is sent, the HCD starts reading data from the ISTL1 buffer or writing data to the ISTL1 buffer. While the HCD is accessing the buffer, the buffer pointer of ISTL1 also automatically increases.

When the pointer has reached the initialized byte count of the HcTransferCounter register, the host controller sets the AllEOTInterrupt bit in the HcμPInterrupt register to logic 1 and updates the HcBufferStatus register.

14.7.4 HcISTLToggleRate register (R/W: 47h/C7h)

The rate of toggling between ISTL0 and ISTL1 is programmable. The HcISTLToggleRate register is provided to program the required toggle rate in the range of 0 ms to 15 ms at intervals of 1 ms. The bit allocation of the register is shown in Table 85.

Code (Hex): 47 — read

Code (Hex): C7 — write

Table 85. HcISTLToggleRate register: bit allocation

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

Table 86. HcISTLToggleRate register: bit description

Bit	Symbol	Description
15 to 4	-	reserved
3 to 0	ISTLToggleRate[3:0]	The required toggle rate in ms.

14.8 Interrupt transfer registers

14.8.1 HcINTLBufferSize register (R/W: 33h/B3h)

This register allows you to allocate the size of the INTL buffer to be used for interrupt transactions. The default value of the buffer size is set to 128 bytes, and the maximum allowable allocated size is 4096 bytes. Table 87 shows the bit description of the register.

Code (Hex): 33 — read

Code (Hex): B3 — write

Table 87. HcINTLBufferSize register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	INTLBufferSize[15:0]	R/W	0080h	The size of the buffer to be used for interrupt transactions and must be specified in bytes.

14.8.2 HcINTLBufferPort register (R/W: 43h/C3h)

In addition to the HcDirectAddressData register, the ISP1362 provides this register to act as another data port to access the INTL buffer. The starting address to access the buffer is always fixed at 0000h. Therefore, random access of the INTL buffer is not allowed. The bit description of the HcINTLBufferPort register is given in Table 88.

Code (Hex): 43 — read

Code (Hex): C3 — write

Table 88. HcINTLBufferPort register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	DataWord[15:0]	R/W	0000h	Data in the INTL buffer to be accessed through this data port.

The HCD is first required to initialize the HcTransferCounter register with the byte count to be transferred and check the HcBufferStatus register. The HCD then sends the command (43h to read the INTL buffer, and C3h to write to the INTL buffer) to the host controller through the I/O port of the microprocessor. After the command is sent, the HCD starts reading data from the INTL buffer or writing data to the INTL buffer. While the HCD is accessing the buffer, the buffer pointer of INTL also automatically increases. When the pointer has reached the initialized byte count of the HcTransferCounter register, the host controller sets the AllEOTInterrupt bit of the HcμPInterrupt register to logic 1 and updates the HcBufferStatus register.

14.8.3 HcINTLBlkSize register (R/W: 53h/D3h)

The ISP1362 requires the INTL buffer to be partitioned into several equal sized blocks so that the host controller can skip the current PTD and proceed to process the next PTD easily. The block size of the INTL buffer is required to be specified in this register and must be a multiple of 8 bytes. The default value of the block size is 64 bytes, and the maximum allowable block size is 1024 bytes. Table 89 shows the bit allocation of the register.

Code (Hex): 53 — read

Code (Hex): D3 — write

Table 89. HcINTLBlkSize register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved						BlockSize[9:8]	
Reset	-	-	-	-	-	-	0	0
Access	-	-	-	-	-	-	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	BlockSize[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. HcINTLBlkSize register: bit description

Bit	Symbol	Description
15 to 10	-	reserved
9 to 0	BlockSize[9:0]	The block size of the INTL buffer.

14.8.4 HcINTLPTDDoneMap register (R: 17h)

This is a 32-bit register, and the bit description is given in Table 91. Every bit of the register represents the processing status of a PTD. Bit 0 of the register represents the first PTD stored in the INTL buffer, bit 1 represents the second PTD stored in the buffer, and so

on. The register is updated once every ms by the host controller and is cleared on read by the HCD. Bits that are set represent its corresponding PTDs are processed by the host controller and the ACK token is received from the device.

Code (Hex): 17 — read only

Table 91. HcINTLPTDDoneMap register: bit description

Bit	Symbol	Access	Value	Description
31 to 0	PTDDoneBits[31:0]	R	0000h	**0 —** The PTD stored in the INTL buffer has not successfully been processed by the host controller. **1 —** The PTD stored in the INTL buffer has successfully been processed by the host controller.

14.8.5 HcINTLPTDSkipMap register (R/W: 18h/98h)

This is a 32-bit register, and the bit description is given in Table 92. Bit 0 of the register represents the first PTD stored in the INTL buffer, bit 1 represents the second PTD stored in the buffer, and so on. When a bit is set by the HCD, the corresponding PTD is skipped and is not processed by the host controller. The host controller processes the skipped PTD if the HCD has reset its corresponding skipped bit to logic 0. Clearing the corresponding bit in the HcINTLPTDSkipMap register when there is no valid data in the block will cause unpredictable behavior of the host controller.

Code (Hex): 18 — read

Code (Hex): 98 — write

Table 92. HcINTLPTDSkipMap register: bit description

Bit	Symbol	Access	Value	Description
31 to 0	SkipBits[31:0]	R/W	0000h	**0 —** The host controller processes the PTD. **1 —** The host controller skips processing the PTD.

14.8.6 HcINTLLastPTD register (R/W: 19h/99h)

This is a 32-bit register, and Table 93 shows its bit description. Bit 0 of the register represents the first PTD stored in the INTL buffer, bit 1 represents the second PTD stored in the buffer, and so on. The bit that is set to logic 1 by the HCD is used as an indication to the host controller that its corresponding PTD is the last PTD stored in the INTL buffer. When the processing of the last PTD is complete, the host controller proceeds to process ATL transactions.

Code (Hex): 19 — read

Code (Hex): 99 — write

Table 93. HcINTLLastPTD register: bit description

Bit	Symbol	Access	Value	Description
31 to 0	LastPTDBits[31:0]	R/W	0000h	**0 —** The PTD is not the last PTD stored in the buffer. **1 —** The PTD is the last PTD stored in the buffer.

14.8.7 HcINTLCurrentActivePTD register (R: 1Ah)

This register indicates which PTD stored in the INTL buffer is currently active and is updated by the host controller. The HCD can use it as a buffer pointer to decide which PTD locations are currently free to fill in new PTDs to the buffer. This indication is to prevent the HCD from accidentally writing into the currently active PTD buffer location. Table 94 shows the bit allocation of the register.

Code (Hex): 1A — read only

Table 94. HcINTLCurrentActivePTD register: bit allocation

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

Table 95. HcINTLCurrentActivePTD register: bit description

Bit	Symbol	Description
15 to 5	-	reserved
4 to 0	ActivePTD[4:0]	This 5-bit number represents the PTD that is currently active.

14.9 Control and bulk transfer (aperiodic transfer) registers

14.9.1 HcATLBufferSize register (R/W: 34h/B4h)

This register allows you to allocate the size of the ATL buffer to be used for aperiodic transactions. The default value of the buffer size is set to 512 bytes, and the maximum allowable allocated size is 4096 bytes. The bit description of the register is given in Table 96.

Code (Hex): 34 — read

Code (Hex): B4 — write

Table 96. HcATLBufferSize register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	ATLBufferSize[15:0]	R/W	0200h	The size of the buffer to be used for aperiodic transactions and must be specified in bytes.

14.9.2 HcATLBufferPort register (R/W: 44h/C4h)

In addition to the HcDirectAddressData register, the ISP1362 provides this register to act as another data port to access the ATL buffer. The starting address to access the buffer is always fixed at 0000h. Therefore, random access of the ATL buffer is not allowed. The bit description of the HcATLBufferPort register is given in Table 97.

Code (Hex): 44 — read

Code (Hex): C4 — write

Table 97. HcATLBufferPort register: bit description

Bit	Symbol	Access	Value	Description
15 to 0	DataWord[15:0]	R/W	0000h	Data of the ATL buffer to be accessed through this data port.

The HCD is first required to initialize the HcTransferCounter register with the byte count to be transferred and check the HcBufferStatus register. The HCD then sends the command (44h to read from the ATL buffer, and C4h to write to the ATL buffer) to the host controller through the I/O port of the microprocessor. After the command is sent, the HCD starts reading data from the ATL buffer or writing data to the ATL buffer. While the HCD is accessing the buffer, the buffer pointer of ATL also automatically increases. When the pointer has reached the initialized byte count of the HcTransferCounter register, the host controller sets the AllEOTInterrupt bit of the HcμPInterrupt register to logic 1 and updates the HcBufferStatus register.

14.9.3 HcATLBlkSize register (R/W: 54h/D4h)

The ISP1362 partitions the ATL buffer into several equal sized blocks so that the host controller can skip the current PTD and proceed to process the next PTD easily. The block size of the ATL buffer must be specified in this register and must be a multiple of 8 bytes. The bit allocation of the HcATLBlkSize register is given in Table 98.

Code (Hex): 54 — read

Code (Hex): D4 — write

Table 98. HcATLBlkSize register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved						BlockSize[9:8]	
Reset	-	-	-	-	-	-	0	0
Access	-	-	-	-	-	-	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	BlockSize[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 99. HcATLBlkSize register: bit description

Bit	Symbol	Description
15 to 10	-	reserved
9 to 0	BlockSize[9:0]	The block size of the ATL buffer.

14.9.4 HcATLPTDDoneMap register (R: 1Bh)

This is a 32-bit register. The bit description of the register is given in Table 100. Every bit of the register represents the processing status of a PTD. Bit 0 of the register represents the first PTD stored in the ATL buffer, bit 1 represents the second PTD stored in the buffer, and so on. The register is immediately updated after the completion of each ATL PTD processing. It is cleared when read by the HCD. Bits that are set represent its corresponding PTDs have been processed by the host controller and an ACK token has been received from the device.

Code (Hex): 1B — read only

Table 100. HcATLPTDDoneMap register: bit description

Bit	Symbol	Access	Value	Description
31 to 0	PTDDoneBits [31:0]	R	0000h	**0 —** The PTD stored in the ATL buffer was not successfully processed by the host controller. **1 —** The PTD stored in the ATL buffer was successfully processed by the host controller.

14.9.5 HcATLPTDSkipMap register (R/W: 1Ch/9Ch)

This is a 32-bit register, and the bit description is given in Table 101. Bit 0 of the register represents the first PTD stored in the ATL buffer, bit 1 represents the second PTD stored in the buffer, and so on. When the bit is set by the HCD, the corresponding PTD is skipped and is not processed by the host controller. The host controller processes the skipped PTD only if the HCD has reset its corresponding skipped bit to logic 0. Clearing the corresponding bit in the HcATLPTDSkipMap register when there is no valid data in the block will cause unpredictable behavior of the host controller.

Code (Hex): 1C — read

Code (Hex): 9C — write

Table 101. HcATLPTDSkipMap register: bit description

Bit	Symbol	Access	Value	Description
31 to 0	SkipBits [31:0]	R/W	0000h	**0 —** The host controller processes the PTD. **1 —** The host controller skips processing the PTD.

14.9.6 HcATLLastPTD register (R/W: 1Dh/9Dh)

This is a 32-bit register. Table 102 gives the bit description of the register. Bit 0 of the register represents the first PTD stored in the ATL buffer, bit 1 represents the second PTD stored in the buffer, and so on. The bit that is set to logic 1 by the HCD is used as an indication to the host controller that its corresponding PTD is the last PTD stored in the ATL buffer. When the processing of the last PTD is complete, the host controller loops back to process the first PTD stored in the buffer.

Code (Hex): 1D — read

Code (Hex): 9D — write

Table 102. HcATLLastPTD register: bit description

Bit	Symbol	Access	Value	Description
31 to 0	LastPTD Bits[31:0]	R/W	0000h	**0 —** The PTD is not the last PTD stored in the buffer. **1 —** The PTD is the last PTD stored in the buffer.

14.9.7 HcATLCurrentActivePTD register (R: 1Eh)

This register indicates which PTD stored in the ATL buffer is currently active and is updated by the host controller. The HCD can use it as a buffer pointer to decide which PTD locations are currently free to fill in new PTDs to the buffer. This indication helps to prevent the HCD from accidentally writing into the currently active PTD buffer location. Table 103 shows the bit allocation of the register.

Code (Hex): 1E — read only

Table 103. HcATLCurrentActivePTD register: bit allocation

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

Table 104. HcATLCurrentActivePTD register: bit description

Bit	Symbol	Description
15 to 5	-	reserved
4 to 0	ActivePTD[4:0]	This 5-bit number represents the PTD that is currently active.

14.9.8 HcATLPTDDoneThresholdCount register (R/W: 51h/D1h)

This register specifies the number of ATL PTDs to be done to trigger an ATL interrupt. If set to 08h, the host controller will trigger the ATL interrupt (in the HcμPInterrupt register) once every eight ATL PTDs are done. Table 105 shows the bit allocation of the register.

Remark: Do not write 0000h to this register.

Code (Hex): 51 — read

Code (Hex): D1 — write

Table 105. HcATLPTDDoneThresholdCount register: bit allocation

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

Table 106. HcATLPTDDoneThresholdCount register: bit description

Bit	Symbol	Description
15 to 5	-	reserved
4 to 0	PTDDoneCount [4:0]	Number of PTDs to be processed by the host controller to generate an ATL interrupt.

14.9.9 HcATLPTDDoneThresholdTimeOut register (R/W: 52h/D2h)

This is a time-out register used to generate an ATL interrupt. The value in this register indicates the maximum allowable time in milliseconds for the host controller to retry a NAK transaction. This register can be used in combination with HcATLPTDDoneThresholdCount. Table 107 shows the bit allocation of the HcATLPTDDoneThresholdCount register.

Remark: If the time-out indication is not required by the software, or there is no active PTD in the ATL buffer, write 0000h to this register.

Code (Hex): 52 — read

Code (Hex): D2 — write

Table 107. HcATLPTDDoneThresholdTimeOut register: bit allocation

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

Table 108. HcATLPTDDoneThresholdTimeOut register: bit description

Bit	Symbol	Description
15 to 8	-	reserved
7 to 0	PTDDoneTimeOut[7:0]	Maximum allowable time in ms for the host controller to retry a transaction with NAK returned.

15. Peripheral controller registers

The functions and registers of the peripheral controller are accessed using 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 109.

A complete access consists of two phases:

1. **Command phase:** when address pin A0 = HIGH, the peripheral controller interprets the data on the lower byte of the bus (bits D7 to D0) as command code. Commands without a data phase are immediately executed.
2. **Data phase (optional):** when address pin A0 = LOW, the peripheral controller transfers the data on the bus to or from a register or endpoint buffer memory. In case of multi-byte registers, the least significant byte or word is accessed first.

The following applies to a register or buffer memory access in 16-bit bus mode:

- The upper byte (bits D15 to D8) in the command phase or the undefined byte in the data phase are 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 two bytes of the endpoint buffer.

Table 109. Peripheral controller command and register overview

Name	Destination	Code (Hex)	Transaction[1]
Initialization commands			
Write control OUT configuration	DcEndpointConfiguration register endpoint 0 OUT	20	write 1 byte[2]
Write control IN configuration	DcEndpointConfiguration register endpoint 0 IN	21	write 1 byte[2]
Write endpoint n configuration (n = 1 to 14)	DcEndpointConfiguration register endpoint 1 to 14	22 to 2F	write 1 byte[2]
Read control OUT configuration	DcEndpointConfiguration register endpoint 0 OUT	30	read 1 byte[2]
Read control IN configuration	DcEndpointConfiguration register endpoint 0 IN	31	read 1 byte[2]
Read endpoint n configuration (n = 1 to 14)	DcEndpointConfiguration register endpoint 1 to 14	32 to 3F	read 1 byte[2]
Write or read device address	DcAddress register	B6/B7	write or read 1 byte[2]
Write or read Mode register	DcMode register	B8/B9	write or read 1 byte[2]
Write or read hardware configuration	DcHardwareConfiguration register	BA/BB	write or read 2 bytes
Write or read DcInterruptEnable register	DcInterruptEnable register	C2/C3	write or read 4 bytes
Write or read DMA configuration	DcDMAConfiguration register	F0/F1	write or read 2 bytes
Write or read DMA counter	DcDMACounter register	F2/F3	write or read 2 bytes
Reset device	resets all registers	F6	-
Data flow commands			
Write control OUT buffer	illegal: endpoint is read-only	(00)	-
Write control IN buffer	buffer memory endpoint 0 IN	01	N ≤ 64 bytes
Write endpoint n buffer (n = 1 to 14)	buffer memory endpoint 1 to 14 (IN endpoints only)	02 to 0F	isochronous: N ≤ 1023 bytes interrupt/bulk: N ≤ 64 bytes
Read control OUT buffer	buffer memory 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)	buffer memory endpoint 1 to 14 (OUT endpoints only)	12 to 1F	isochronous: N ≤ 1023 bytes[3] 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 byte[2]
Read control IN status	DcEndpointStatus register endpoint 0 IN	51	read 1 byte[2]

Table 109. Peripheral controller command and register overview *...continued*

Name	Destination	Code (Hex)	Transaction[1]
Read endpoint n status (n = 1 to 14)	DcEndpointStatus register n endpoint 1 to 14	52 to 5F	read 1 byte[2]
Validate control OUT buffer	illegal: IN endpoints only[4]	(60)	-
Validate control IN buffer	buffer memory endpoint 0 IN[4]	61	-
Validate endpoint n buffer (n = 1 to 14)	buffer memory endpoint 1 to 14 (IN endpoints only)[4]	62 to 6F	-
Clear control OUT buffer	buffer memory endpoint 0 OUT	70	-
Clear control IN buffer	illegal[5]	(71)	-
Clear endpoint n buffer (n = 1 to 14)	buffer memory endpoint 1 to 14 (OUT endpoints only)[5]	72 to 7F	-
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[6]	DcEndpointStatusImage register endpoint 0 OUT	D0	read 1 byte[2]
Check control IN status[6]	DcEndpointStatusImage register endpoint 0 IN	D1	read 1 byte[2]
Check endpoint n status (n = 1 to 14)[6]	DcEndpointStatusImage register n endpoint 1 to 14	D2 to DF	read 1 byte[2]
Acknowledge set up	endpoint 0 IN and OUT	F4	-
General commands			
Read control OUT error code	DcErrorCode register endpoint 0 OUT	A0	read 1 byte[2]
Read control IN error code	DcErrorCode register endpoint 0 IN	A1	read 1 byte[2]
Read endpoint n error code (n = 1 to 14)	DcErrorCode register endpoint 1 to 14	A2 to AF	read 1 byte[2]
Unlock device	all registers with write access	B0	write 2 bytes
Write or read DcScratch register	DcScratch register	B2/B3	write or read 2 bytes
Read frame number	DcFrameNumber register	B4	read 1 byte or 2 bytes
Read chip ID	DcChipID register	B5	read 2 bytes
Read DcInterrupt register	DcInterrupt register	C0	read 4 bytes

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

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

[3] During the isochronous transfer in 16-bit mode, because N ≤ 1023, the firmware must manage the upper byte.

[4] Validating an OUT endpoint buffer causes unpredictable behavior of the peripheral controller.

[5] Clearing an IN endpoint buffer causes unpredictable behavior of the peripheral controller.

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

15.1 Initialization commands

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

15.1.1 DcEndpointConfiguration register (R/W: 30h to 3Fh/20h to 2Fh)

This command is used to access the DcEndpointConfiguration register (ECR) of the target endpoint. It defines the endpoint type (isochronous or bulk/interrupt), direction (OUT/IN), buffer memory size and buffering scheme. It also enables the endpoint buffer memory. The register bit allocation is shown in Table 110. A bus reset will disable all endpoints.

The allocation of the buffer memory takes place only after **all** 16 endpoints have been configured in sequence (from endpoint 0 OUT to endpoint 14). Although control endpoints have fixed configurations, they must be included in the initialization sequence and must be configured with their default values (see Table 15). Automatic buffer memory allocation starts when endpoint 14 has been configured.

Remark: If any change is made to an endpoint configuration that affects the allocated memory (size, enable/disable), the buffer memory contents of **all** endpoints become 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, endpoints 1 to 14)

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

Transaction — write or read 1 byte (code or data)

Table 110. 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 111. DcEndpointConfiguration register: bit description

Bit	Symbol	Description
7	FIFOEN	Logic 1 enables the FIFO buffer. Logic 0 disables the FIFO buffer.
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 enables the 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 buffer memory size according to Table 16.

15.1.2 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 112.

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 USB bus reset. In response to 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 or read DcAddress register

Transaction — write or read 1 byte (code or data)

Table 112. 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 113. 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.

15.1.3 DcMode register (R/W: B9h/B8h)

This command is used to access the DcMode register, which consists of 1 byte (bit allocation: see Table 114). 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, in which all errors and Not Acknowledge (NAK) conditions will generate an interrupt.

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

Transaction — write or read 1 byte (code or data)

Table 114. DcMode register: bit allocation

Bit	7	6	5	4	3	2	1	0
Symbol	reserved		GOSUSP	reserved	INTENA	DBGMOD	reserved	SOFTCT
Reset	1[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 115. DcMode register: bit description

Bit	Symbol	Description
7 to 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.
2	DBGMOD	Logic 1 enables debug mode, in which all NAKs and errors will generate an interrupt. Logic 0 selects normal operation, in which interrupts are generated on every ACK (bulk or interrupt endpoints) or after every data transfer (isochronous endpoints). Bus reset value: unchanged.
1	-	reserved
0	SOFTCT	Logic 1 enables SoftConnect. This bit is ignored if EXTPUL = 1 in the DcHardwareConfiguration register (see Table 116). Bus reset value: unchanged. **Remark:** In OTG mode, this bit is ignored. The LOC_CONN bit of the OtgControl register controls the pull-up resistor on the OTG_DP1 pin.

15.1.4 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 clock control bits and the clock division factor. The bit allocation is given in Table 116. A bus reset will not change any of programmed bit values.

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

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

Transaction — write or read 2 bytes (code or data)

Table 116. DcHardwareConfiguration register: bit allocation

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

Table 117. 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 pin OTG_DP1 (in device mode) and that SoftConnect is not used. Bus reset value: unchanged.
13	NOLAZY	Logic 1 disables output on pin CLKOUT of the LazyClock frequency (115 kHz ± 50 %) during the suspend state. Logic 0 causes pin CLKOUT to switch to LazyClock output after approximately 2 ms delay, following the setting of bit GOSUSP of the DcMode register. Bus reset value: unchanged.
12	CLKRUN	Logic 1 indicates that internal clocks are always running, even during the suspend state. Logic 0 switches off the internal oscillator and PLL, when they are not needed. During the suspend state, this bit must be made logic 0 to meet suspend current requirements. The clock is stopped after a delay of approximately 2 ms, following the setting of bit GOSUSP of the DcMode register. Bus reset value: unchanged.
11 to 8	CKDIV[3:0]	This field specifies clock division factor N, which controls the clock frequency on output CLKOUT pin. The output frequency in MHz is given by $48/(N+1)$. The clock frequency range is 3 MHz 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 the DREQ2 pin signal polarity (0 = active LOW; 1 = active HIGH). Bus reset value: unchanged.

Table 117. DcHardwareConfiguration register: bit description *...continued*

Bit	Symbol	Description
5	DAKPOL	Selects the $\overline{\text{DACK2}}$ pin signal polarity (0 = active LOW; 1 = active HIGH). Bus reset value: unchanged.
4	-	reserved
3	WKUPCS	Logic 1 enables remote wake-up using a LOW level on input $\overline{\text{CS}}$. Bus reset value: unchanged.
2	-	reserved
1	INTLVL	Selects interrupt signaling mode on output (0 = level; 1 = pulsed). In pulsed mode, an interrupt produces 166 ns pulse. Bus reset value: unchanged.
0	INTPOL	Selects the INT2 signal polarity (0 = active LOW; 1 = active HIGH). Bus reset value: unchanged.

15.1.5 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, EOT, suspend, resume, reset). 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 118.

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

Transaction — write or read 4 bytes (code or data)

Table 118. DcInterruptEnable register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
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	SP_IEEOT	IEPSOF	IESOF	IEEOT	IESUSP	IERESM	IERST
Reset	-	0	0	0	0	0	0	0
Access	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 119. 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. Logic 0 disables interrupts from the indicated endpoint.
9	IEP0IN	Logic 1 enables interrupts from the control IN endpoint. Logic 0 disables interrupts from the control IN endpoint.
8	IEP0OUT	Logic 1 enables interrupts from the control OUT endpoint. Logic 0 disables interrupts from the control OUT endpoint.
7	-	reserved
6	SP_IEEOT	Logic 1 enables interrupt on detecting a short packet. Logic 0 disables interrupt.
5	IEPSOF	Logic 1 enables 1 ms interrupts on detecting pseudo SOF. Logic 0 disables interrupts.
4	IESOF	Logic 1 enables interrupt on the SOF detection. Logic 0 disables interrupt.
3	IEEOT	Logic 1 enables interrupt on the EOT detection. Logic 0 disables interrupt.
2	IESUSP	Logic 1 enables interrupt on detecting a suspend state. Logic 0 disables interrupt.
1	IERESM	Logic 1 enables interrupt on detecting a resume state. Logic 0 disables interrupt.
0	IERST	Logic 1 enables interrupt on detecting a bus reset. Logic 0 disables interrupt.

15.1.6 DcDMAConfiguration (R/W: F1h/F0h)

This command defines the DMA configuration of the peripheral controller, and enables or disables DMA transfers. The command accesses the DcDMAConfiguration register, which consists of two bytes. The bit allocation is given in Table 120. A bus reset will clear bit DMAEN (DMA disabled), all other bits remain unchanged.

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

Transaction — write or read 2 bytes (code or data)

Table 120. DcDMAConfiguration register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	CNTREN	SHORTP	reserved					
Reset	0[1]	0[1]	-	-	-	-	-	-
Access	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[1]	0[1]
Access	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W

[1] Unchanged by a bus reset.

Table 121. 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 or empty packet mode. When receiving (OUT endpoint) a short or empty packet, an EOT condition is generated. When transmitting (IN endpoint), this bit must be cleared. Bus reset value: unchanged.
13 to 8	-	reserved
7 to 4	EPDIX[3:0]	Indicates the destination endpoint for DMA, see Table 18.
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 or not (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.

15.1.7 DcDMACounter register (R/W: F3h/F2h)

This command accesses the DcDMACounter register, which consists of two bytes. The bit allocation is given in Table 122. 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 programmed bit values.

The internal DMA counter is automatically reloaded from the DcDMACounter register. For details, see Section 15.1.6.

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

Transaction — write or read 2 bytes (code or data)

Table 122. 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 123. DcDMACounter register: bit description

Bit	Symbol	Description
15 to 0	DMACR[15:0]	This field indicates the number of bytes for a DMA transfer.

15.1.8 Reset device (F6h)

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

Code (Hex): F6 — reset the device

Transaction — none (code only)

15.2 Data flow commands

Data flow commands are used to manage data transmission between USB endpoints and the system microprocessor. Much of the data flow is initiated using an interrupt to the microprocessor. Data flow commands are used to access endpoints and determine whether the endpoint buffer memory contains valid data.

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

15.2.1 Write or read endpoint buffer (R/W: 10h,12h to 1Fh/01h to 0Fh)

This command is used to access endpoint buffer memory to read/write. First, the buffer pointer is reset to the start of the buffer. Following the command, a maximum of (N + 2) bytes can be written or read, N represents the size of the endpoint buffer. For 16-bit access, the maximum number of words is (M + 1), with M given as (N + 1) divided by 2. After each read or write action, the buffer pointer is automatically incremented by two.

In DMA, the first two bytes or the first word (the packet length) is skipped: transfers start at the third byte or the second word of the endpoint buffer. When reading, the peripheral controller can detect the last byte or word by using the EOP condition. When writing to a bulk or interrupt endpoint, the endpoint buffer must be completely filled before sending 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 data will cause unpredictable behavior of the peripheral controller.

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

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

Transaction — write or read maximum N + 2 bytes (isochronous endpoint: $N \leq 1023$, bulk/interrupt endpoint: $N \leq 32$) (code or data)

The data in the endpoint buffer memory must be organized as shown in Table 124. An example of endpoint buffer memory access is given in Table 125.

Table 124. Endpoint buffer memory organization

Word #	Description
0 (lower byte)	packet length (lower byte)
0 (upper byte)	packet length (upper byte)
1 (lower byte)	data byte 1

Table 124. Endpoint buffer memory organization *...continued*

Word #	Description
1 (upper byte)	data byte 2
…	…
M = (N + 1) / 2	data byte N

Table 125. Example of endpoint buffer memory access

A0	Phase	Bus lines	Word #	Description
HIGH	command	D[7:0]	-	command code (00h to 1Fh)
		D[15:8]	-	ignored
LOW	data	D[15:0]	0	packet length
LOW	data	D[15:0]	1	data word 1 (data byte 2, data byte 1)
LOW	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, against writing into an OUT buffer or reading from an IN buffer. Any of these actions can cause an incorrect operation. Data residing in an OUT buffer is only meaningful after a successful transaction. Exception: during the DMA access of a double-buffered endpoint, the buffer pointer automatically points to the secondary buffer after reaching the end of the primary buffer.

15.2.2 Read endpoint status (R: 50h to 5Fh)

This command is used to read the status of an endpoint buffer memory. The command accesses the DcEndpointStatus register, the bit allocation of which is shown in Table 126. Reading the DcEndpointStatus register will clear the interrupt bit set for the corresponding endpoint in the DcInterrupt register (see Table 142).

All bits of the DcEndpointStatus register are read-only. Bit EPSTAL is controlled by the Stall or Unstall commands and by the reception of a set-up token (see Section 15.2.3).

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

Transaction — read 1 byte (code only)

Table 126. 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	-
Access	R	R	R	R	R	R	R	-

Table 127. 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, cleared to logic 0 by an Unstall Endpoint command. The endpoint is automatically unstalled on receiving a set-up 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 data PID of the next packet (0 = DATA PID; 1 = DATA1 PID).
3	OVERWRITE	This bit is set by the hardware. Logic 1 indicates that a new set-up packet has overwritten the previous set-up information, before it was acknowledged or before the endpoint was stalled. Once writing of the set-up data is completed, a read back of this register clears this bit. The firmware must check this bit before sending an acknowledge set-up command or stalling the endpoint. On reading logic 1, the firmware must stop ongoing set-up actions and wait for a new set-up packet.
2	SETUPT	Logic 1 indicates that the buffer contains a set-up packet.
1	CPUBUF	This bit indicates which buffer is currently selected for the CPU access (0 = primary buffer; 1 = secondary buffer).
0	-	reserved

15.2.3 Stall endpoint or unstall endpoint (40h to 4Fh/80h to 8Fh)

These commands are used to stall or unstall an endpoint. The commands modify the content of the DcEndpointStatus register (see Table 126).

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

When a stalled endpoint is unstalled (either by using the Unstall Endpoint command or by receiving a set-up 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, endpoints 1 to 14)

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

Transaction — none (code only)

15.2.4 Validate endpoint buffer (61h to 6Fh)

This command signals the presence of valid data for transmission to the USB host. The validation occurs 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 buffer memory for CPU access.

Remark: For special aspects of the control IN endpoint, see Section 12.3.6.

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

Transaction — none (code only)

15.2.5 Clear endpoint buffer (70h, 72h to 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 buffer memory for CPU access.

Remark: For special aspects of the control OUT endpoint, see Section 12.3.6.

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

Transaction — none (code only)

15.2.6 DcEndpointStatusImage register (D0h to DFh)

This command is used to check the status of the selected endpoint buffer memory, 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 128.

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

Transaction — write or read 1 byte (code or data)

Table 128. 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	-
Access	R	R	R	R	R	R	R	-

Table 129. 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 data PID of the next packet (0 = DATA0 PID; 1 = DATA1 PID).
3	OVERWRITE	This bit is set by the hardware. Logic 1 indicates that a new set-up packet has overwritten the previous set-up information, before it was acknowledged or before the endpoint was stalled. Once writing of the set-up data is completed, a read back of this register clears this bit. The firmware must check this bit before sending an acknowledge set-up command or stalling the endpoint. On reading logic 1, the firmware must stop ongoing set-up actions and wait for a new set-up packet.
2	SETUPT	Logic 1 indicates that the buffer contains a set-up packet.
1	CPUBUF	This bit indicates which buffer is currently selected for CPU access (0 = primary buffer; 1 = secondary buffer).
0	-	reserved

15.2.7 Acknowledge set up (F4h)

This command acknowledges to the host that a set-up packet is received. The arrival of a set-up packet disables the Validate Buffer and Clear Buffer commands for the control IN and OUT endpoints. The microprocessor must re-enable these commands by sending an acknowledge set-up command, see Section 12.3.6.

Code (Hex): F4 — acknowledge set up

Transaction — none (code only)

15.3 General commands

15.3.1 Read endpoint error code (R: A0h to 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 130.

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

Transaction — read 1 byte (code or data)

Table 130. 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
Access	R	R	-	R	R	R	R	R

Table 131. DcErrorCode register: bit description

Bit	Symbol	Description
7	UNREAD	Logic 1 indicates that a new event occurred before the previous status is 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 132.
0	RTOK	Logic 1 indicates that data was successfully received or transmitted.

Table 132. 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 set-up token to a non-control endpoint
0100	token CRC error
0101	data CRC error
0110	time-out error
0111	babble error

Table 132. Transaction error codes *...continued*

Error code (Binary)	Description
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

15.3.2 Unlock Device (B0h)

This command unlocks the peripheral controller from write-protection mode after a resume. In the suspend state, all registers and buffer memory are write-protected to prevent data corruption by external devices during a resume. Also, the register access to read is possible only after the 'unlock device' command is executed.

After waking up from the suspend state, the firmware must unlock registers and buffer memory by using this command, by writing the unlock code (AA37h) into the DcLock register (8-bit bus: lower byte first). The bit allocation of the DcLock register is given in Table 133.

Code (Hex): B0 — unlock the device

Transaction — write 2 bytes (unlock code) (code or data)

Table 133. DcLock register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	UNLOCK[15:8] = 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	UNLOCK[7:0] = 37h							
Reset	0	0	1	1	0	1	1	1
Access	W	W	W	W	W	W	W	W

Table 134. DcLock register: bit description

Bit	Symbol	Description
15 to 0	UNLOCK[15:0]	Sending data AA37h unlocks internal registers and buffer memory to write, following a resume.

15.3.3 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. For example, the device status before powering down in the suspend state.

The register bit allocation is given in Table 135.

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

Transaction — write or read 2 bytes (code or data)

Table 135. 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
Access	-	-	-	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 136. 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

15.3.4 DcFrameNumber register (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 137.

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

Code (Hex): B4 — read frame number

Transaction — read 1 byte or 2 bytes (code or data)

Table 137. DcFrameNumber register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	reserved					SOFR[9:8]		
Reset [1]	-	-	-	-	-	0	0	0
Access	-	-	-	-	-	R	R	R
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	SOFR[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 138. DcFrameNumber register: bit description

Bit	Symbol	Description
15 to 11	-	reserved
10 to 0	SOFR[9:0]	frame number

Table 139. Example of the DcFrameNumber register access

A0	Phase	Bus lines	Word#	Description
HIGH	command	D[15:8]	-	ignored
		D[7:0]	-	command code (B4h)
LOW	data	D[15:0]	0	frame number

15.3.5 DcChipID (R: B5h)

This command reads the chip identification code and the hardware version number. The firmware must check this information to determine supported functions and features. This command accesses the DcChipID register, which is shown in Table 140.

Code (Hex): B5 — read chip ID

Transaction — read 2 bytes (code or data)

Table 140. DcChipID register: bit allocation

Bit	15	14	13	12	11	10	9	8
Symbol	CHIPIDH[7:0]							
Reset	0	0	1	1	0	1	1	0
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	1	0	0	0	0
Access	R	R	R	R	R	R	R	R

Table 141. DcChipID register: bit description

Bit	Symbol	Description
15 to 8	CHIPIDH[7:0]	chip ID code (36h)
7 to 0	CHIPIDL[7:0]	silicon version (30h, with 30 representing the BCD encoded version number)

15.3.6 DcInterrupt register (R: C0h)

This command indicates the sources of interrupts as stored in the 4 bytes DcInterrupt register. Each individual endpoint has its own interrupt bit. The bit allocation of the DcInterrupt register is shown in Table 142. Bit BUSTATUS is used to verify the current bus status in the interrupt service routine. Interrupts are enabled using the DcInterruptEnable register, see Section 15.1.5.

While reading the DcInterrupt register, it is recommended that both 2-byte words are read completely.

Code (Hex): C0 — read DcInterrupt register

Transaction — read 4 bytes (code or data)

Table 142. DcInterrupt register: bit allocation

Bit	31	30	29	28	27	26	25	24
Symbol	reserved							
Reset	-	-	-	-	-	-	-	-
Access	-	-	-	-	-	-	-	-
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	SP_EOT	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 143. DcInterrupt register: bit description

Bit	Symbol	Description
31 to 24	-	reserved
23 to 10	EP14 to EP1	Logic 1 indicates the interrupt source(s): endpoints 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	SP_EOT	Logic 1 indicates that an EOT interrupt has occurred for a short period.
5	PSOF	Logic 1 indicates that an interrupt is issued every 1 ms because of the pseudo SOF; after three missed SOFs, the suspend state is entered.
4	SOF	Logic 1 indicates that an 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.

16. Limiting values

Table 144. Limiting values

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

Symbol	Parameter	Conditions		Min	Max	Unit
V_{CC}	supply voltage			−0.5	+4.6	V
V_I	input voltage			−0.5	+6.0	V
I_{lu}	latch-up current	$V_I < 0$ V 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 through a 1.5 kΩ resistor (Human Body Model).

17. Recommended operating conditions

Table 145. Recommended operating conditions

DP represents OTG_DP1 and H_DP2, and DM represents OTG_DM1 and H_DM2.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{CC}	supply voltage			3.0	3.3	3.6	V
V_I	input voltage	3.3 V tolerant pins	[1]	0	3.3	3.6	V
		5 V tolerant pins	[1]	0	5.0	5.5	V
		on pin X1 when external clock is used		3.0	3.3	3.6	V
$V_{IA(I/O)}$	input voltage on analog I/O pins	pins DP and DM		0	-	3.6	V
$V_{O(od)}$	open-drain output pull-up voltage	5 V tolerant pins		0	-	5.5	V
		non 5 V tolerant pins		0	-	3.6	V
T_{amb}	ambient temperature			−40	-	+85	°C

[1] Input voltage on digital I/O lines.

18. Static characteristics

Table 146. Static characteristics: supply pins

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

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
$I_{CC(HC)}$	operating supply current for the Host Controller	peripheral controller suspended		-	33	-	mA
$I_{CC(DC)}$	operating supply current for the Peripheral Controller	host controller suspended		-	20	-	mA
$I_{CC(HC+DC)}$	operating supply current for the host and the device			-	50	-	mA
$I_{CC(susp)}$	suspend supply current	host controller and peripheral controller are suspended	[1]	-	60	-	µA

[1] The power consumption on the charge pump is not included.

Table 147. Static characteristics: digital pins

V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C 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_{CC} − 0.1	-	-	V
Leakage current							
I_{LI}	input leakage current		[2]	−5	-	+5	µA
C_{IN}	pin capacitance	pin to GND		-	-	5	pF
Open-drain outputs							
I_{OZ}	off-state output current			−5	-	+5	µA

[1] Not applicable for open-drain outputs.

[2] These values are applicable to transistor inputs. The value will be different if internal pull-up or pull-down resistors are used.

Table 148. Static characteristics: analog I/O pins (DP, DM)

V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; unless otherwise specified.[1]

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input levels						
V_{DI}	differential input sensitivity voltage	$\|V_{I(D+)} - V_{I(D-)}\|$	0.2	-	-	V
V_{CM}	differential common mode voltage range	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	-	+10	μA
Capacitance						
C_{IN}	transceiver capacitance	pin to GND	-	-	10	pF
Resistance						
$R_{pd(OTG)}$	pull-down resistance on pins OTG_DP1 and OTG_DM1	enable internal resistors	14.25	-	24.8	kΩ
$R_{pd(H)}$	pull-down resistance on pins H_DP2 and H_DM2	enable internal resistors	10	-	20	kΩ
$R_{pu(OTG)}$	pull-up resistance on OTG_DP1	bus idle	900	-	1575	Ω
		bus driven	1425	-	3090	Ω
Z_{DRV}	driver output impedance for driver which is not high-speed capable	steady-state drive [2]	29	-	44	Ω
Z_{INP}	input impedance exclusive of pull-up/pull-down (for low-/full-speed)		10	-	-	MΩ
Termination						
V_{TERM}	termination voltage for upstream facing port pull-up	for upstream port pull up (R_{PU}) [3]	3.0	-	3.6	V

[1] DP represents OTG_DP1 and H_DP2, and DM represents OTG_DM1 and H_DM2. D+ is the USB positive data line and D− is the USB negative data line.

[2] Includes external resistors of 18 Ω ± 10 % on H_DP2 and H_DM2, and 27 Ω ± 10 % on OTG_DP1 and OTG_DM1.

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

Table 149. Static characteristics: charge pump

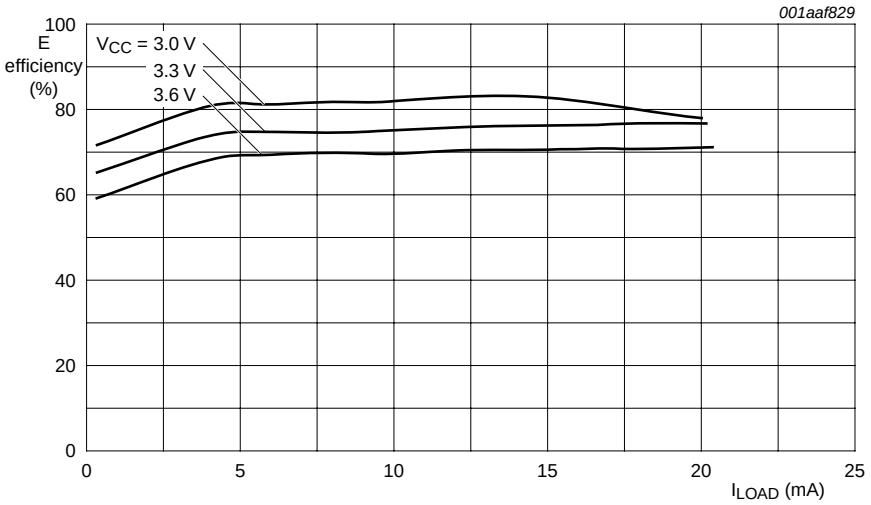
V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; C_{LOAD} = 2 μF; unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{BUS}	bus supply voltage	I_{LOAD} = 8 mA from $V_{BUS(OTG)}$; see Figure 29	-	5	5.25	V

Table 149. Static characteristics: charge pump *...continued*

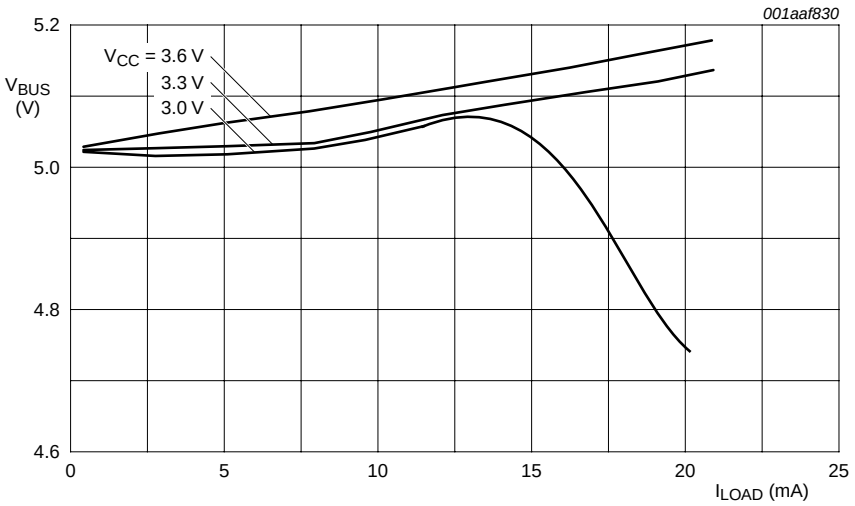
V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; C_{LOAD} = 2 µF; unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{LOAD}	load current	external capacitor of 27 nF; V_{CC} = 3.0 V to 3.6 V	-	-	8	mA
		external capacitor of 82 nF; V_{CC} = 3.0 V to 3.3 V	-	-	14	mA
		external capacitor of 82 nF; V_{CC} = 3.3 V to 3.6 V	-	-	20	mA
C_{LOAD}	output capacitance		1	-	6.5	µF
$V_{BUS(LEAK)}$	$V_{BUS(OTG)}$ leakage voltage	$V_{BUS(OTG)}$ not driven	-	-	0.2	V
$I_{CC(cp)(susp)}$	suspend supply current for charge pump	GlobalPowerDown bit of the HcHardwareConfiguration register is logic 0	-	-	45	µA
		GlobalPowerDown bit of the HcHardwareConfiguration register is logic 1	-	-	15	µA
$I_{CC(cp)}$	charge pump supply current	ATX is idle				
		I_{LOAD} = 8 mA	-	-	20	mA
		I_{LOAD} = 0 mA	-	-	300	µA
$V_{th(VBUS\_VLD)}$	V_{BUS} valid threshold		4.4	-	-	V
$V_{th(SESS\_END)}$	V_{BUS} session end threshold		0.2	-	0.8	V
$V_{hys(SESS\_END)}$	V_{BUS} session end hysteresis		-	150	-	mV
$V_{th(ASESS\_VLD)}$	V_{BUS} A valid threshold		0.8	-	2	V
$V_{hys(ASESS\_VLD)}$	V_{BUS} A valid hysteresis		-	200	-	mV
$V_{th(BSESS\_VLD)}$	V_{BUS} B valid threshold		2	-	4	V
$V_{hys(BSESS\_VLD)}$	V_{BUS} B valid hysteresis		-	200	-	mV
E	efficiency when loaded	I_{LOAD} = 8 mA; V_{IN} = 3 V; see Figure 28	-	75	-	%
$I_{VBUS(leak)}$	leakage current from V_{BUS}		-	15	-	µA
$R_{VBUS(PU)}$	V_{BUS} pull-up resistance	pull to V_{CC} when enabled	281	-	-	Ω
$R_{VBUS(PD)}$	V_{BUS} pull-down resistance	pull to GND when enabled	656	-	-	Ω
$R_{VBUS(IDLE)}$	V_{BUS} idle impedance for the A-device	when ID = LOW and DRV_VBUS = 0	40	-	100	kΩ
$R_{VBUS(ACTIVE)}$	V_{BUS} active pull-down impedance	when ID = HIGH and DRV_VBUS =1	-	350	-	kΩ



82 nF charge pump capacitor.

Fig 28. Efficiency as a function of load current



82 nF charge pump capacitor.

Fig 29. Output voltage as a function of load current

19. Dynamic characteristics

Table 150. Dynamic characteristics

V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C 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		10	-	-	ms
		crystal oscillator stopped	[1]	-	-	-	ms
Crystal oscillator							
f_{xtal}	crystal frequency		[2]	-	12	-	MHz
R_S	series resistance			-	-	100	Ω
C_{LOAD}	load capacitance	C_{X1}, C_{X2} = 22 pF		-	12	-	pF
External clock input							
J	external clock jitter			-	-	500	ps
t_{DUTY}	clock duty cycle			45	50	55	%
t_{CR}	rise time			-	-	3	ns
t_{CF}	fall time			-	-	3	ns

[1] Dependent on the crystal oscillator startup time.

[2] Tolerance of the clock frequency is ±50 ppm.

Table 151. Dynamic characteristics: analog I/O lines (DP, DM)

V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; C_L = 50 pF; R_{PU} = 1.5 kΩ ± 5 % on DP to V_{TERM}; unless otherwise specified.[1]

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
Driver characteristics							
t_{FR}	rise time	C_L = 50 pF; 10 % to 90 % of $\lvert V_{OH} - V_{OL}\rvert$		4	-	20	ns
t_{FF}	fall time	C_L = 50 pF; 90 % to 10 % of $\lvert V_{OH} - V_{OL}\rvert$		4	-	20	ns
FRFM	differential rise time/fall time matching	(t_{FR}/t_{FF})	[2]	90	-	111.11	%
V_{CRS}	output signal crossover voltage		[2][3]	1.3	-	2.0	V

[1] DP represents OTG_DP1 and H_DP2, and DM represents OTG_DM1 and H_DM2. Test circuit.

[2] Excluding the first transition from the idle state.

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

Table 152. Dynamic characteristics: charge pump

V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; C_{LOAD} = 2 µF; unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Driver characteristics						
$t_{START-UP}$	rise time to V_{BUS} = 4.4 V	I_{LOAD} = 8 mA; C_{LOAD} = 10 µF	-	-	100	ms
t_{COMP_CLK}	clock period		1.5	-	3	µs
$t_{VBUS(VALID\_dly)}$	minimum time $V_{BUS(VALID)}$ error		100	-	200	µs

Table 152. Dynamic characteristics: charge pump *…continued*

V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; C_{LOAD} = 2 µF; unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{VBUS(PULSE)}$	V_{BUS} pulsing time		10	-	30	ms
$t_{VBUS(VALID\_dly)}$	V_{BUS} pull-down time		50	-	-	ms
V_{RIPPLE}	output ripple with constant load	I_{LOAD} = 8 mA	-	-	50	mV

19.1 Programmed I/O timing

- If you are accessing only the host controller, then the host controller programmed I/O timing applies.
- If you are accessing only the peripheral controller, then the peripheral controller programmed I/O timing applies.
- If you are accessing both the host controller and the peripheral controller, then the programmed I/O timing is either the host controller or peripheral controller programmed I/O timing, depending on whichever is higher.

19.1.1 Host controller programmed I/O timing

Table 153. Dynamic characteristics: host controller programmed interface timing

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{AS}	address set-up time before $\overline{CS}$		5	-	-	ns
t_{AH}	address hold time after $\overline{WR}$		2	-	-	ns
Read timing						
t_{SHSL_R}	first $\overline{RD}$/$\overline{WR}$ after command (A0 = HIGH)	register access	300	-	-	ns
t_{SHSL_B}	first $\overline{RD}$/$\overline{WR}$ after command (A0 = HIGH)	buffer access	462	-	-	ns
t_{SLRL}	$\overline{CS}$ LOW to $\overline{RD}$ LOW		0	-	-	ns
t_{RHSH}	$\overline{RD}$ HIGH to $\overline{CS}$ HIGH		0	-	-	ns
t_{RL}	$\overline{RD}$ 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	ns
t_{RLDV}	$\overline{RD}$ LOW to data valid		-	-	22	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		3	-	-	ns
t_{WDH}	$\overline{WR}$ data hold time		4	-	-	ns

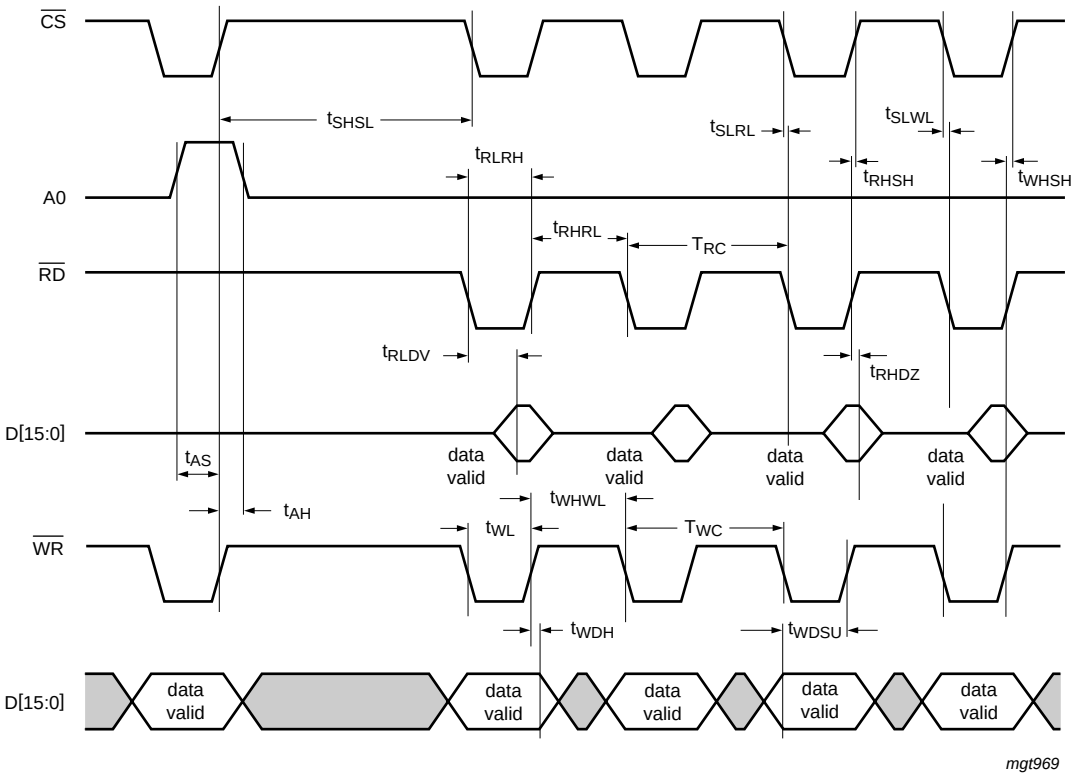


Fig 30. Host controller programmed interface timing

19.1.2 Peripheral controller programmed I/O timing

Table 154. Dynamic characteristics: peripheral controller programmed interface timing

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

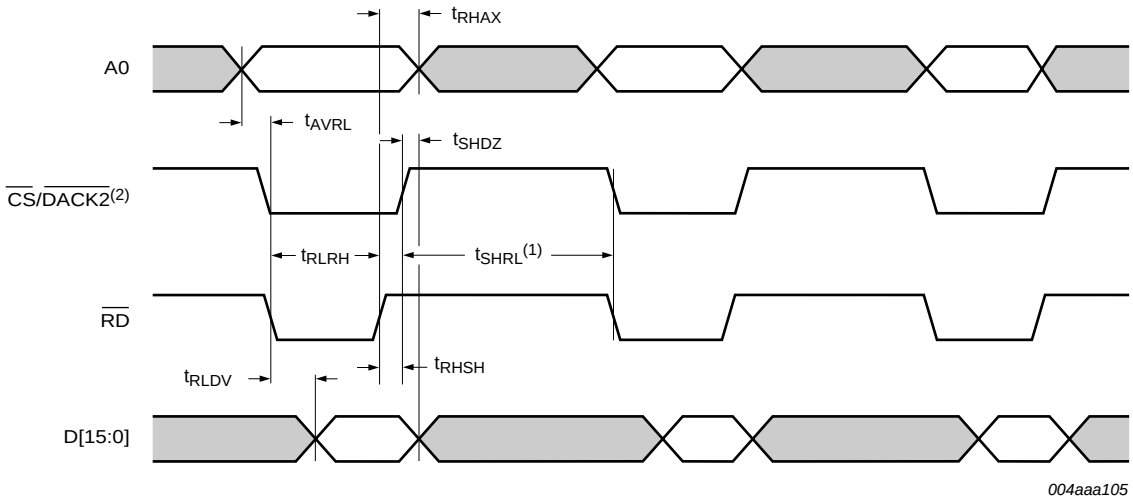
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Read timing (see Figure 31)						
t_{RHAX}	address hold time after $\overline{RD}$ HIGH		0	-	-	ns
t_{AVRL}	address set-up time before $\overline{RD}$ LOW		0	-	-	ns
t_{SHDZ}	data outputs high-impedance time after $\overline{CS}$ HIGH		-	-	3	ns
t_{RHSH}	chip deselect time 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}	$\overline{CS}$ HIGH until next ISP1362 $\overline{RD}$		120	-	-	ns
$t_{SHRL} + t_{RLRH} + t_{RHSH}$	read cycle time		180	-	-	ns
Write timing (see Figure 32)						
t_{WHAX}	address hold time after $\overline{WR}$ HIGH		1	-	-	ns
t_{AVWL}	address set-up time before $\overline{WR}$ LOW		0	-	-	ns
t_{SHWL}	$\overline{CS}$ HIGH until next ISP1362 $\overline{WR}$		120	-	-	ns
$t_{SHWL} + t_{WLWH} + t_{WHSH}$	write cycle time	[1]	180	-	-	ns
t_{WLWH}	$\overline{WR}$ pulse width		22	-	-	ns

Table 154. Dynamic characteristics: peripheral controller programmed interface timing *...continued*

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
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

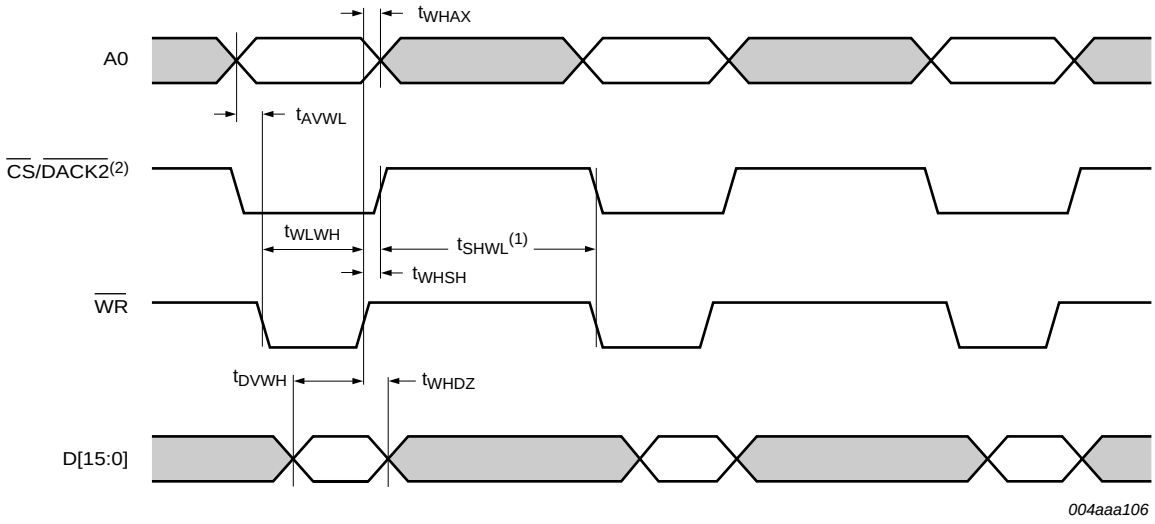
[1] In the command to data phase, the minimum value of the write command to the read data or write data cycle time must be 205 ns.



(1) For t_{SHRL}, both $\overline{CS}$ and $\overline{RD}$ must be deasserted.

(2) Programmable polarity: shown as active LOW.

Fig 31. Peripheral controller programmed interface read timing (I/O and 8237 compatible DMA)



(1) For t_{SHWL}, both $\overline{CS}$ and $\overline{WR}$ must be deasserted.

(2) Programmable polarity: shown as active LOW.

Fig 32. Peripheral controller programmed interface write timing (I/O and 8237 compatible DMA)

19.2 DMA timing

19.2.1 Host controller single-cycle DMA timing

Table 155. Dynamic characteristics: host controller single-cycle DMA timing

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Read/write timing						
t_{RL}	$\overline{RD}$ pulse width		33	-	-	ns
t_{RLDV}	read process data set-up time		30	-	-	ns
t_{RHDZ}	read process data hold time		0	-	-	ns
t_{WSU}	write process data set-up time		5	-	-	ns
t_{WHD}	write process data hold time		0	-	-	ns
t_{AHRH}	$\overline{DACK1}$ HIGH to DREQ1 HIGH		72	-	-	ns
t_{ALRL}	$\overline{DACK1}$ LOW to DREQ1 LOW		-	-	21	ns
T_{DC}	DREQ1 cycle		-[1]	-	-	ns
t_{SHAH}	$\overline{RD}$/$\overline{WR}$ HIGH to $\overline{DACK1}$ HIGH		0	-	-	ns
t_{RHAL}	DREQ1 HIGH to $\overline{DACK1}$ LOW		0	-	-	ns
t_{DS}	DREQ1 pulse spacing		146	-	-	ns

[1] $t_{RHAL} + t_{DS} + t_{ALRL}$

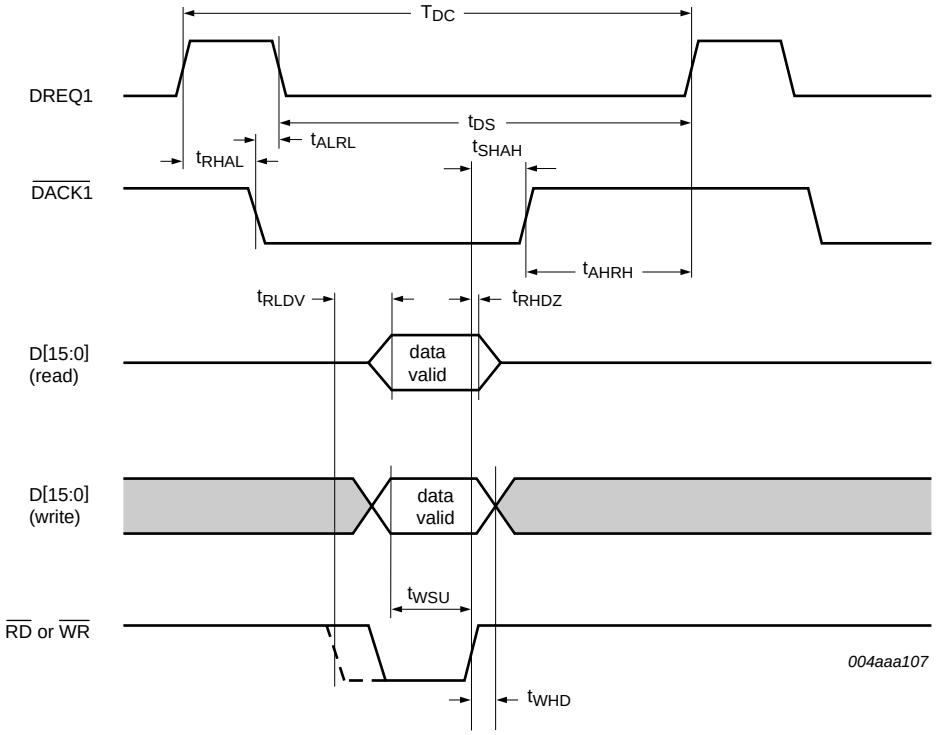


Fig 33. Host controller single-cycle DMA timing

19.2.2 Host controller burst mode DMA timing

Table 156. Dynamic characteristics: host controller burst mode DMA timing

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

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

[1] t_{SLAL} + (4 or 8)t_{RC} + t_{DS}

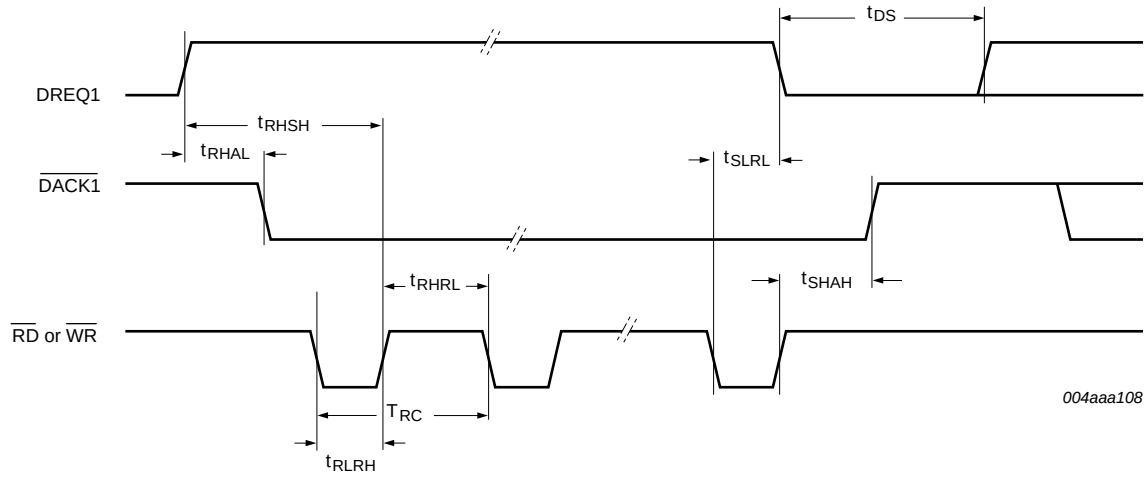
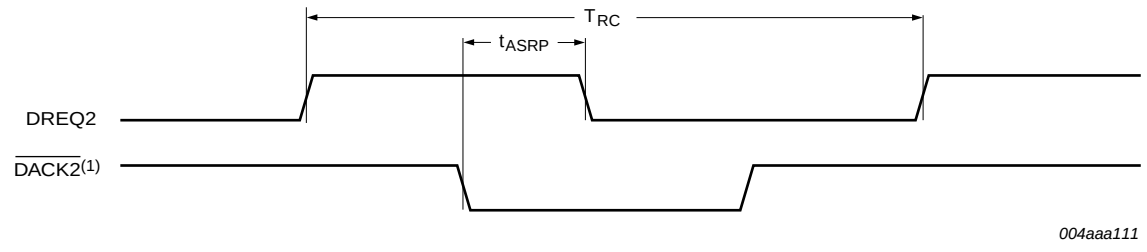


Fig 34. Host controller burst mode DMA timing

19.2.3 Peripheral controller single-cycle DMA timing (8237 mode)

Table 157. Dynamic characteristics: peripheral controller single-cycle DMA timing (8237 mode)
V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; unless otherwise specified.

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



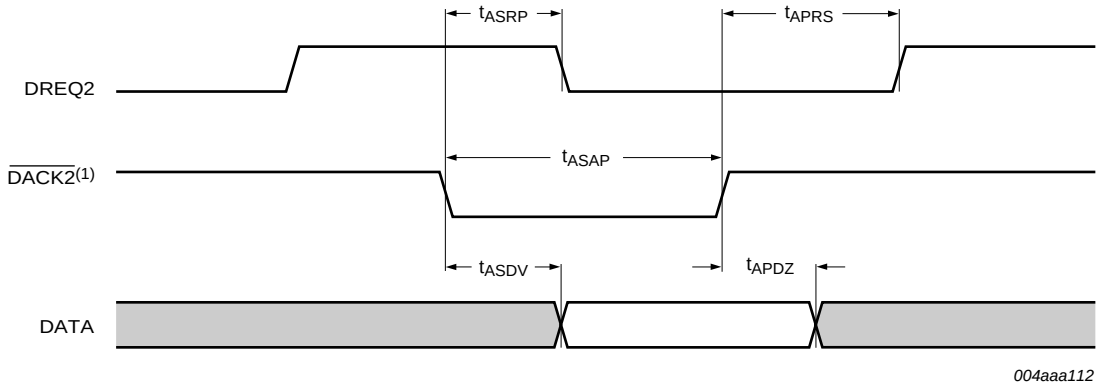
(1) Programmable polarity: shown as active LOW.

Fig 35. Peripheral controller single-cycle DMA timing (8237 mode)

19.2.4 Peripheral controller single-cycle DMA read timing in DACK-only mode

Table 158. Dynamic characteristics: peripheral controller single-cycle DMA read timing in DACK-only mode
V_{CC} = 3.0 V to 3.6 V; GND = 0 V; T_{amb} = −40 °C to +85 °C; unless otherwise specified.

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



(1) Programmable polarity: shown as active LOW.

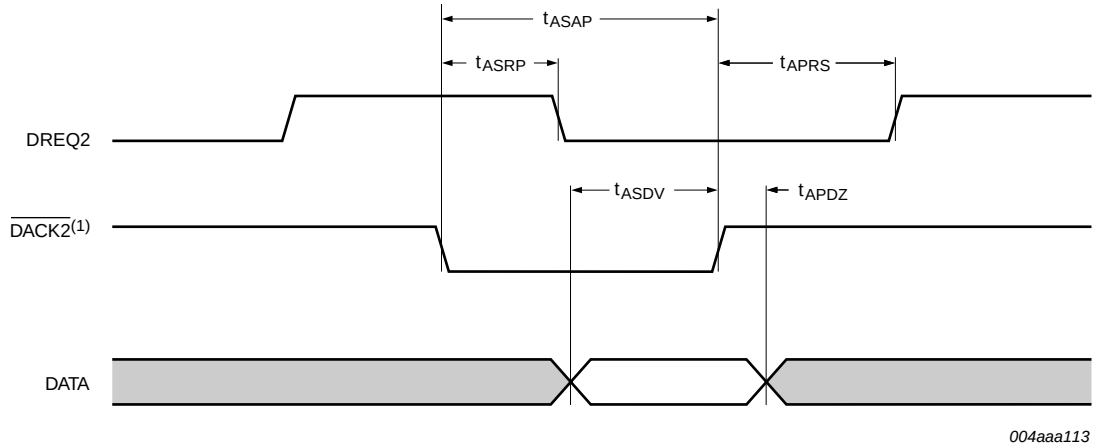
Fig 36. Peripheral controller single-cycle DMA read timing in DACK-only mode

19.2.5 Peripheral controller single-cycle DMA write timing in DACK-only mode

Table 159. Dynamic characteristics: peripheral controller single-cycle DMA write timing in DACK-only mode

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

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



(1) Programmable polarity: shown as active LOW.

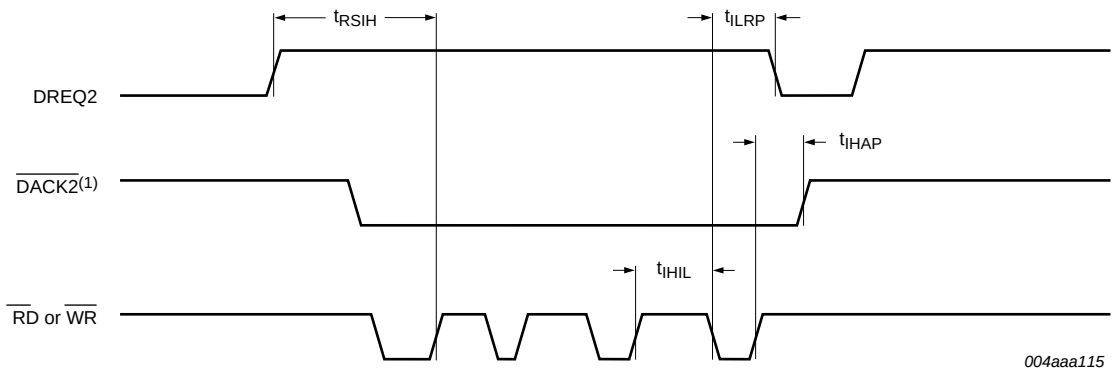
Fig 37. Peripheral controller single-cycle DMA write timing in DACK-only mode

19.2.6 Peripheral controller burst mode DMA timing

Table 160. Dynamic characteristics: peripheral controller burst mode DMA timing

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

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}	$\overline{DACK}$ 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)	$t_{\overline{RL}}$ or $t_{\overline{WL}}$ is 30 ns (min)	160	-	-	ns

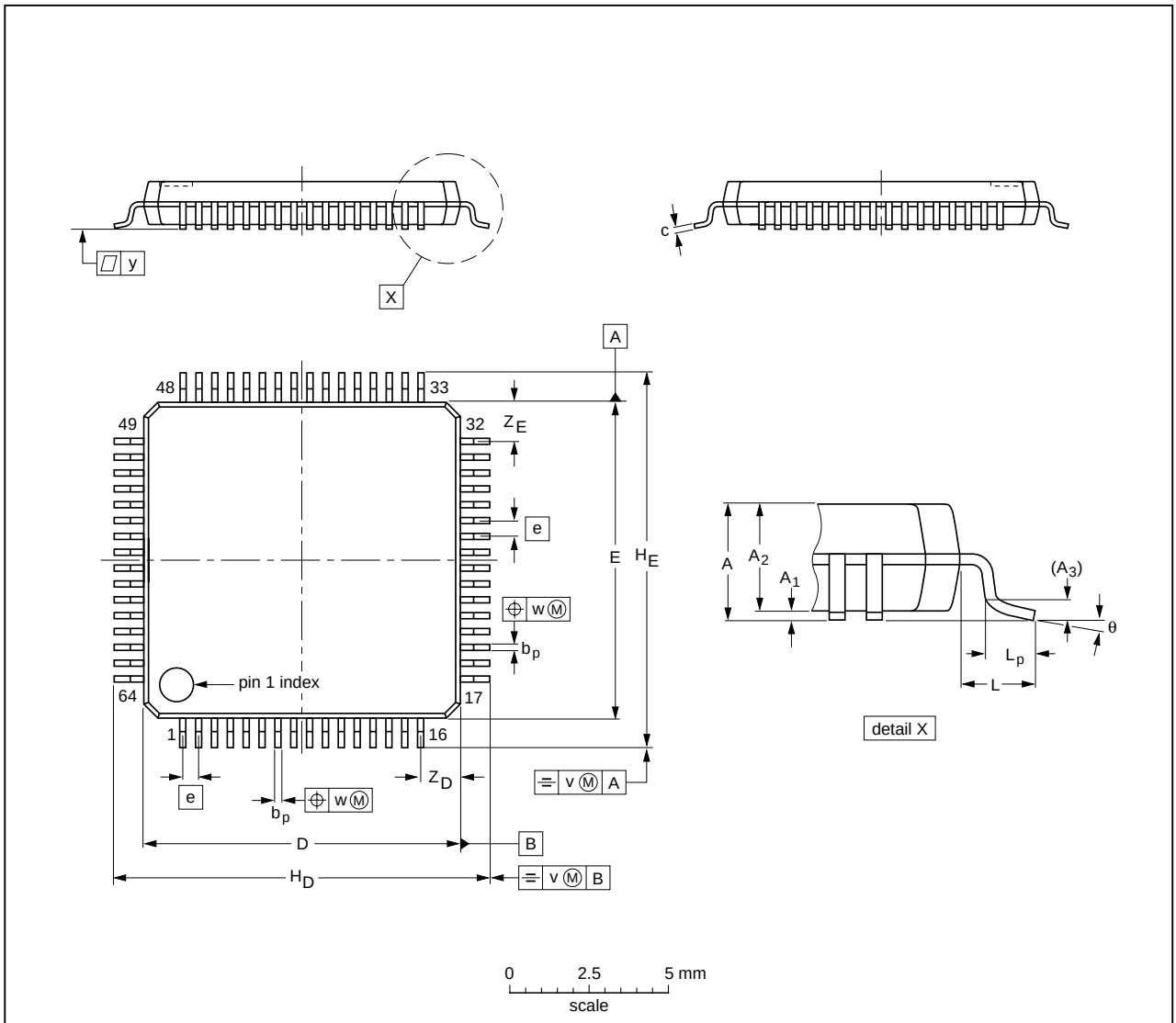


(1) Programmable polarity: shown as active LOW.

Fig 38. Peripheral controller burst mode DMA timing

20. 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	1.6	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	0.75 0.45	0.2	0.12	0.1	1.45 1.05	1.45 1.05	7^o 0^o

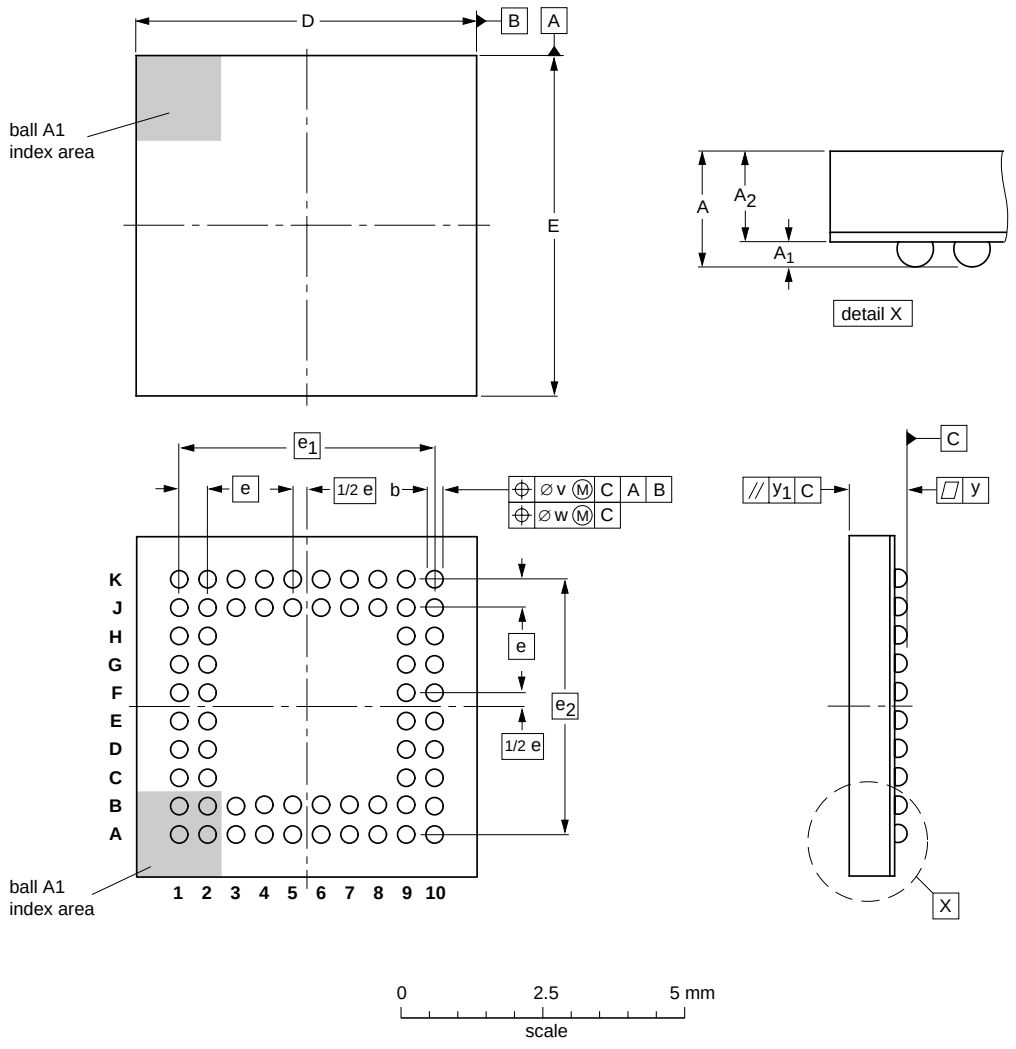
Note

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

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT314-2	136E10	MS-026				~~00-01-19~~ 03-02-25

Fig 39. Package outline SOT314-2 (LQFP64)

TFBGA64: plastic thin fine-pitch ball grid array package; 64 balls; body 6 x 6 x 0.8 mm **SOT543-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	b	D	E	e	e_1	e_2	v	w	y	y_1
mm	1.1	0.25 0.15	0.85 0.75	0.35 0.25	6.1 5.9	6.1 5.9	0.5	4.5	4.5	0.15	0.05	0.08	0.1

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT543-1	- - -	MO-195	- - -			~~00-11-22~~ 02-04-09

Fig 40. Package outline SOT543-1 (TFBGA64)

21. Soldering of SMD packages

This text provides a very brief insight into a complex technology. A more in-depth account of soldering ICs can be found in Application Note *AN10365 "Surface mount reflow soldering description"*.

21.1 Introduction to soldering

Soldering is one of the most common methods through which packages are attached to Printed Circuit Boards (PCBs), to form electrical circuits. The soldered joint provides both the mechanical and the electrical connection. There is no single soldering method that is ideal for all IC packages. Wave soldering is often preferred when through-hole and Surface Mount Devices (SMDs) are mixed on one printed wiring board; however, it is not suitable for fine pitch SMDs. Reflow soldering is ideal for the small pitches and high densities that come with increased miniaturization.

21.2 Wave and reflow soldering

Wave soldering is a joining technology in which the joints are made by solder coming from a standing wave of liquid solder. The wave soldering process is suitable for the following:

- Through-hole components
- Leaded or leadless SMDs, which are glued to the surface of the printed circuit board

Not all SMDs can be wave soldered. Packages with solder balls, and some leadless packages which have solder lands underneath the body, cannot be wave soldered. Also, leaded SMDs with leads having a pitch smaller than ~0.6 mm cannot be wave soldered, due to an increased probability of bridging.

The reflow soldering process involves applying solder paste to a board, followed by component placement and exposure to a temperature profile. Leaded packages, packages with solder balls, and leadless packages are all reflow solderable.

Key characteristics in both wave and reflow soldering are:

- Board specifications, including the board finish, solder masks and vias
- Package footprints, including solder thieves and orientation
- The moisture sensitivity level of the packages
- Package placement
- Inspection and repair
- Lead-free soldering versus SnPb soldering

21.3 Wave soldering

Key characteristics in wave soldering are:

- Process issues, such as application of adhesive and flux, clinching of leads, board transport, the solder wave parameters, and the time during which components are exposed to the wave
- Solder bath specifications, including temperature and impurities

21.4 Reflow soldering

Key characteristics in reflow soldering are:

- Lead-free versus SnPb soldering; note that a lead-free reflow process usually leads to higher minimum peak temperatures (see Figure 41) than a SnPb process, thus reducing the process window
- Solder paste printing issues including smearing, release, and adjusting the process window for a mix of large and small components on one board
- Reflow temperature profile; this profile includes preheat, reflow (in which the board is heated to the peak temperature) and cooling down. It is imperative that the peak temperature is high enough for the solder to make reliable solder joints (a solder paste characteristic). In addition, the peak temperature must be low enough that the packages and/or boards are not damaged. The peak temperature of the package depends on package thickness and volume and is classified in accordance with Table 161 and 162

Table 161. SnPb eutectic process (from J-STD-020C)

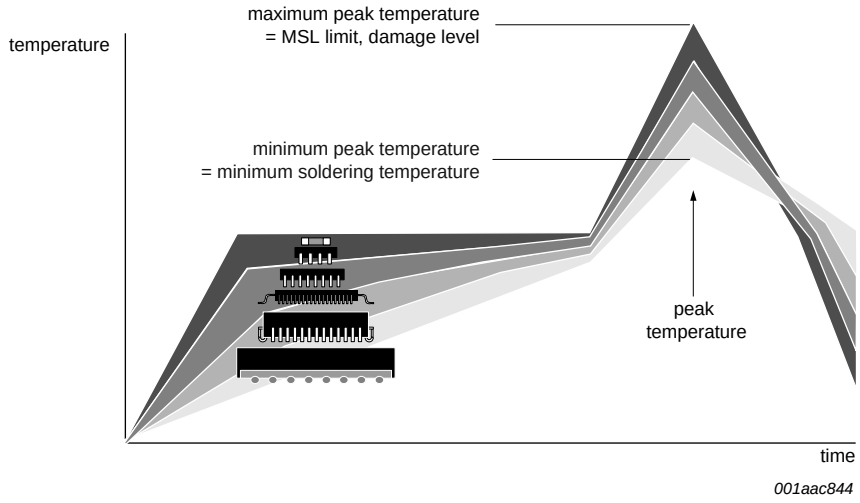
Package thickness (mm)	Package reflow temperature (°C)	
	Volume (mm^3)	
	< 350	≥ 350
< 2.5	235	220
≥ 2.5	220	220

Table 162. Lead-free process (from J-STD-020C)

Package thickness (mm)	Package reflow temperature (°C)		
	Volume (mm^3)		
	< 350	350 to 2000	> 2000
< 1.6	260	260	260
1.6 to 2.5	260	250	245
> 2.5	250	245	245

Moisture sensitivity precautions, as indicated on the packing, must be respected at all times.

Studies have shown that small packages reach higher temperatures during reflow soldering, see Figure 41.



MSL: Moisture Sensitivity Level

Fig 41. Temperature profiles for large and small components

For further information on temperature profiles, refer to Application Note *AN10365 "Surface mount reflow soldering description"*.

22. Abbreviations

Table 163. Abbreviations

Acronym	Description
ACK	Acknowledge
ASIC	Application-Specific Integrated Circuit
AT	Advanced Technology
ATL	Asynchronous Transfer List
ATX	Analog USB Transceiver
CMOS	Complementary Metal-Oxide Semiconductor
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
DSC	Digital Still Camera
ED	Endpoint Descriptor
EHCI	Enhanced Host Controller Interface
EMI	ElectroMagnetic Interference
EOF	End-Of-Frame
EOP	End-Of-Packet
EOT	End-Of-Transfer
ESR	Equivalent Series Resistance
GPS	Global Positioning System
HC	Host Controller
HCCA	Host Controller Communication Area

Table 163. Abbreviations *...continued*

Acronym	Description
HCD	Host Controller Driver
HCI	Host Controller Interface
HNP	Host Negotiation Protocol
INT	Interrupt
INTL	Interrupt Transfer List
IS	Implementation-Specific
ISO	Isochronous
ISR	Interrupt Service Routine
ISTL	Isochronous Transfer List
LS	Low-Speed
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
MSB	Most Significant Bit
NAK	Not Acknowledged
OHCI	Open Host Controller Interface
OPR	Operational
OTG	On-The-Go
PDA	Personal Digital Assistant
PID	Packet IDentifier
PIO	Programmed Input/Output
PLL	Phase-Locked Loop
PMOS	Positive Metal-Oxide Semiconductor
POR	Power-On Reset
PORP	Power-On Reset Pulse
POST	Power-On Self Test
PTD	Proprietary Transfer Descriptor
RISC	Reduced Instruction Set Computing
SIE	Serial Interface Engine
SOF	Start-Of-Frame
SRP	Session Request Protocol
TD	Transfer Descriptor
USB	Universal Serial Bus
USBD	Universal Serial Bus Device

23. References

[1] On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0a

[2] Universal Serial Bus Specification Rev. 2.0

[3] ISP136x Embedded Programming Guide (UM10008)

[4] Open Host Controller Interface Specification for USB Release 1.0a

[5] Interrupt Control application note

24. Revision history

Table 164. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
ISP1362_6	20090121	Product data sheet	-	ISP1362_5
Modifications:	• Globally changed NXP Semiconductors and NXP to ST-NXP Wireless. Also updated the legal text. • Changed PTD to Proprietary Transfer Descriptor. • Section 6 “Pinning information”: updated description of pin $\overline{\text{RESET}}$. • Section 11.9 “ISP1362 host controller power management”: updated the third paragraph. • Updated symbol description based on latest standards. • Table 146 “Static characteristics: supply pins”: updated the description. • Section 19.1 “Programmed I/O timing”: updated the last bulleted list.			
ISP1362_5	20070508	Product data sheet	-	ISP1362-04
ISP1362-04 (9397 750 13957)	20041224	Product data	-	ISP1362-03
ISP1362-03 (9397 750 12337)	20040106	Product data	-	ISP1362-02
ISP1362-02 (9397 750 10767)	20030219	Product data	-	ISP1362-01
ISP1362-01 (9397 750 10087)	20021120	Preliminary data	-	-

25. Tables

continued >>

26. Figures

27. Contents

continued >>

continued >>

For more information, please visit: http://www.stnwireless.com

Date of release: 21 January 2009
Document identifier: ISP1362_6

Please Read Carefully:

Information in this document is provided solely in connection with ST-NXP products. ST-NXP Wireless NV and its subsidiaries (“ST-NXP”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST-NXP products are sold pursuant to ST-NXP’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST-NXP products and services described herein, and ST-NXP assumes no liability whatsoever relating to the choice, selection or use of the ST-NXP products and services described herein. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST-NXP for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST-NXP’S TERMS AND CONDITIONS OF SALE ST-NXP DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST-NXP PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST-NXP REPRESENTATIVE, ST-NXP PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST-NXP PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST-NXP products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST-NXP for the ST-NXP product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST-NXP.

ST-NXP and the ST-NXP logo are trademarks or registered trademarks of ST-NXP in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST-NXP logo is a registered trademark of ST-NXP Wireless. All other names are the property of their respective owners.

ST-NXP Wireless group of companies

Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - India - Italy - Japan - Korea - Malaysia - Mexico - Netherlands - Singapore - Sweden - Switzerland - Taiwan - United Kingdom - United States of America

www.stnwireless.com