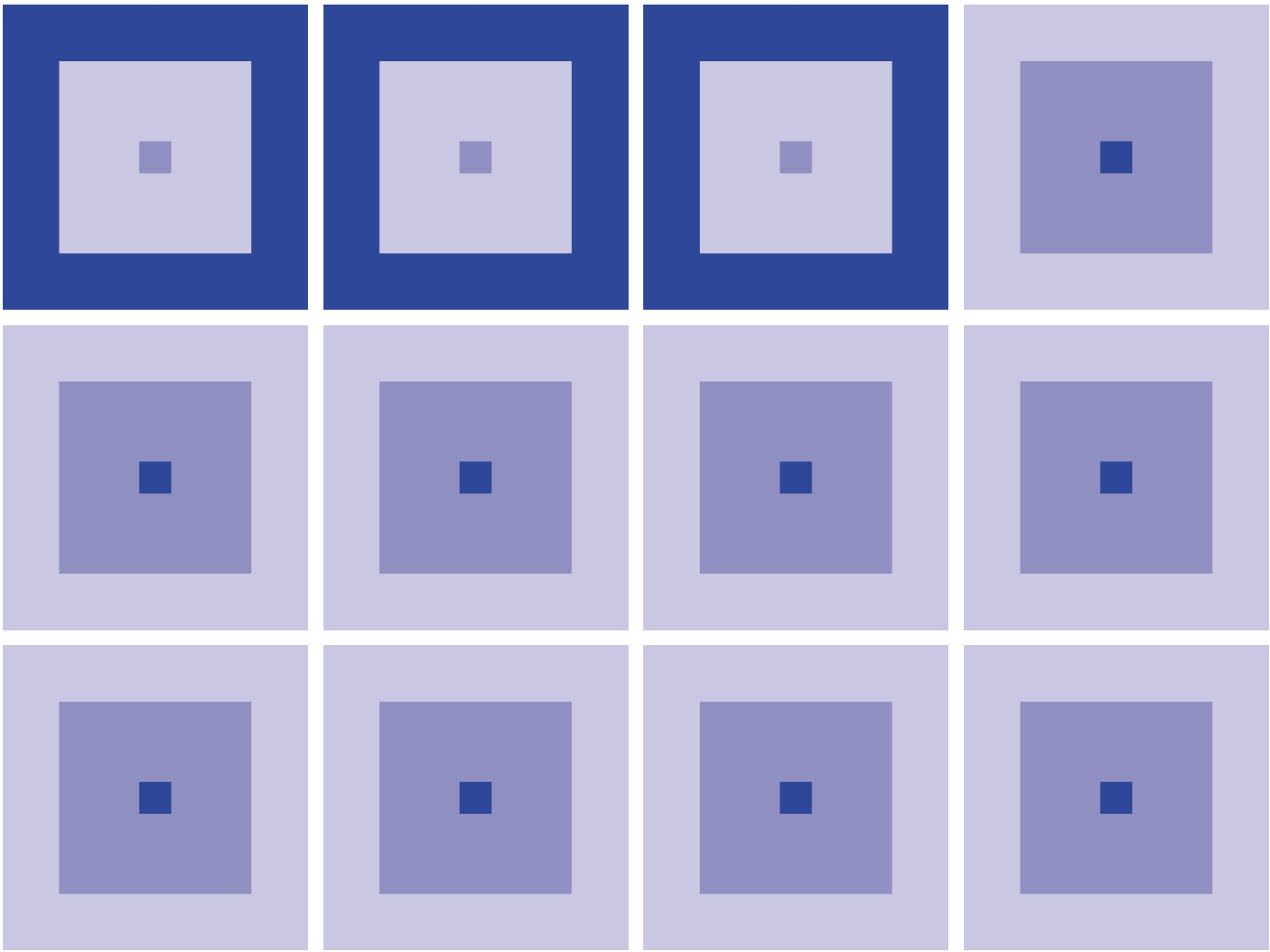
EPSON

USB2.0 Device Controller

# S1R72003

# Technical Manual





SEIKO EPSON CORPORATION

## NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. This material or portions thereof may contain technology or the subject relating to strategic products under the control of the Foreign Exchange and Foreign Trade Law of Japan and may require an export license from the Ministry of International Trade and Industry or other approval from anther government agency.



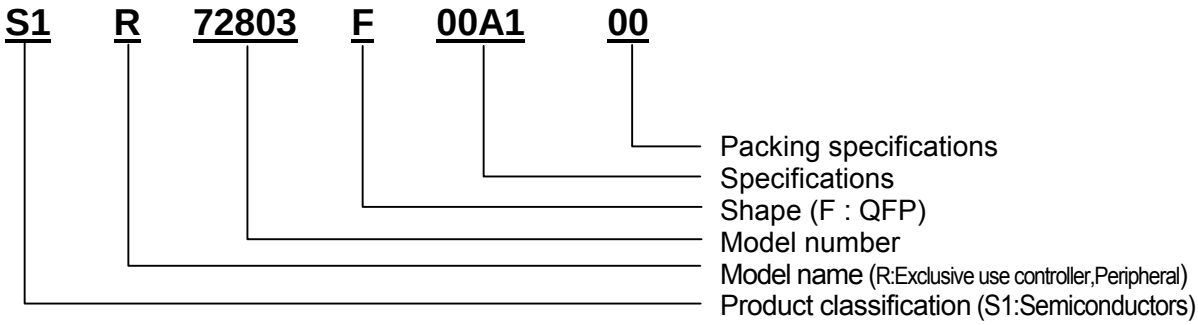
All other product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

## The information of the product number change

Starting April 1, 2001 the product number has been changed as listed below. Please use the new product number when you place an order. For further information, please contact Epson sales representative.

## Configuration of product number

●DEVICES



# Contents

www.DataSheet4U.com

# 1. DESCRIPTION

The S1R72003F00B100 is a general-purpose USB device controller LSI that supports the USB 2.0 high-speed mode. With the field-proven, UTMI Rev 1.0-compliant transceiver circuit, it assures connectivity of USB devices.

# 2. FEATURES

- Supports HS (480Mbps) and FS (12Mbps) transfers.
- Supports Control, Bulk, Interrupt, and Isochronous transfers.
- Supports three general-purpose Endpoints and Endpoint 0.
- Contains a 2.5-KB programmable FIFO for Endpoint use.
- Incorporates IDE and general-purpose DMA ports.



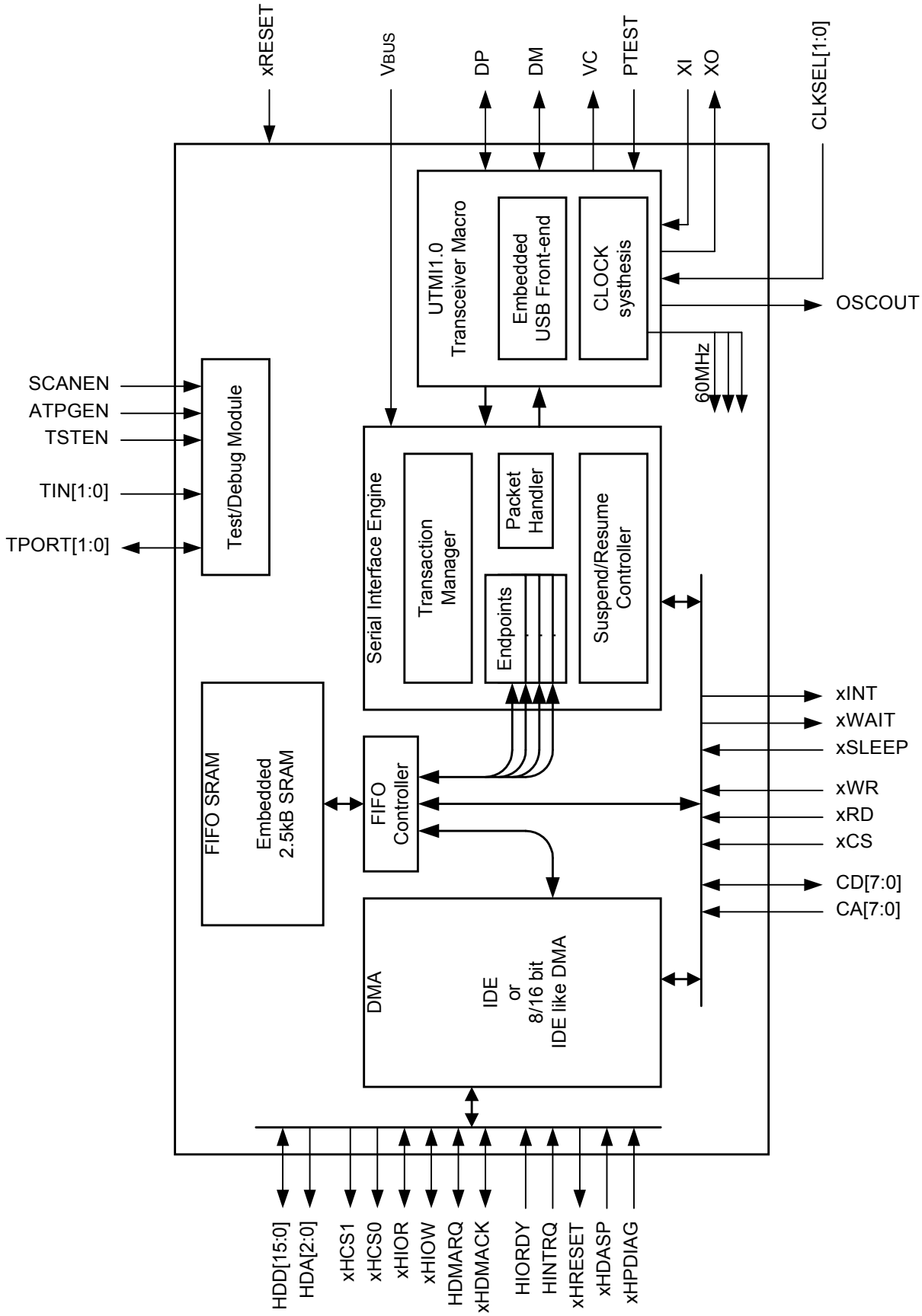
IDE:
  Supports PIO modes 1/2/3/4.
  Supports Multiword DMA modes 0/1/2.
  Supports Ultra-DMA modes 1/2/3/4.

General-purpose DMA:
  The bus width can be set to 8 or 16 bits.
  Both master and slave are supported.

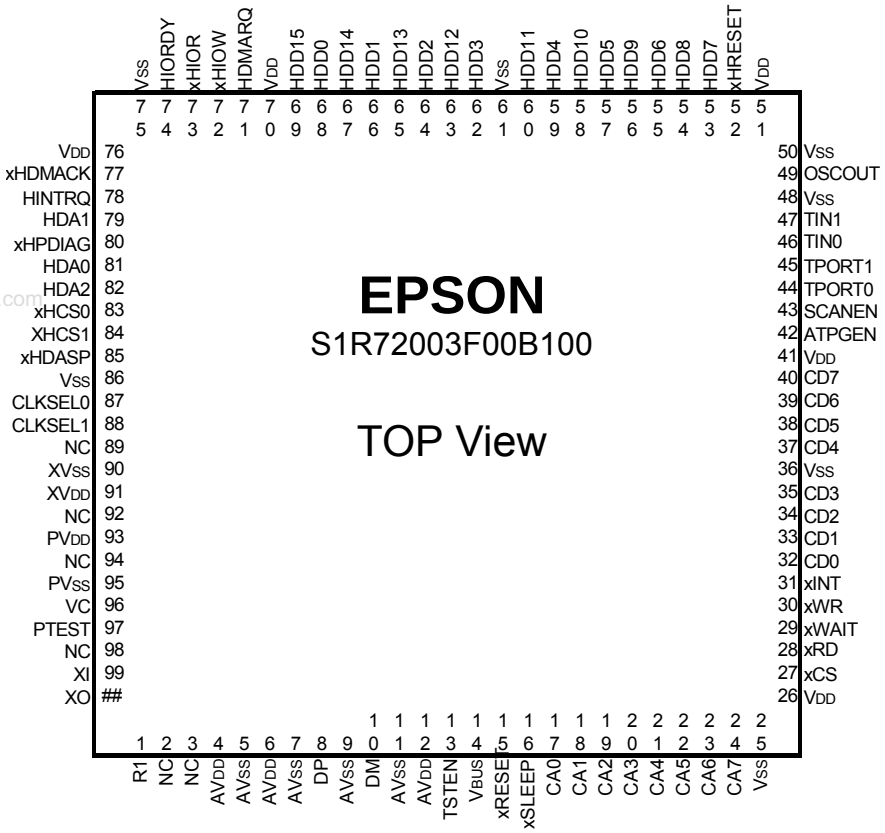
- Incorporates an 8-bit, general-purpose CPU interface.
- Accommodates 12, 16, 20, and 24-MHz crystal resonators for clock input.
- Multiple Power management mode
  Snooze: PLL halt mode triggered by detection of the suspend state of USB.
  Sleep: Oscillation circuit halt mode triggered by xSLEEP pin assertion.
- Operates on a single 3.3 V power supply
- Uses 5 V tolerant cells for VBUS, CPU interface, and DMA port input pins.
- Supplied as a 100-pin QFP package

* No radiation resistant design measure has been incorporated.

# 3. BLOCK DIAGRAM



# 4. PIN ASSIGNMENT



# 5. PIN DESCRIPTION

## 5.1 CPU Interface

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| CD7<br>CD6<br>CD5<br>CD4<br>CD3<br>CD2<br>CD1<br>CD0 | CPU Data | 40<br>39<br>38<br>37<br>35<br>34<br>33<br>32 | I/O<br>(3 state<br>pull up) | CPU data bus<br>During reads, register data is output from this bus.<br>During writes, the CPU delivers the register data to be set through this bus.<br>Uses a 5V tolerant cell. |
| CA7<br>CA6<br>CA5<br>CA4<br>CA3<br>CA2<br>CA1<br>CA0 | CPU Address | 24<br>23<br>22<br>21<br>20<br>19<br>18<br>17 | I (pull up) | CPU address bus<br>This bus specifies the register address.<br>Uses a 5V tolerant cell. |
| xRD | Read Strobe | 28 | I (pull up) | CPU read strobe. Uses a 5V tolerant cell. |
| xWR | Write Strobe | 30 | I (pull up) | CPU write strobe. Uses a 5V tolerant cell. |
| xCS | Chip Select | 27 | I (pull up) | Register select signal Uses a 5V tolerant cell. |
| xSLEEP | Sleep mode | 16 | I (pull up) | Sleep mode set signal. Uses a 5V tolerant cell.<br>When this pins is asserted during the snooze mode, the S1R72003F00B100 enters the sleep mode.<br>The oscillation circuit halts during the sleep mode. You should be careful when the CPU uses OSCOUT. The S1R72003F00B100 is roused from sleep mode in the following cases:<br>• When resume is asserted on the USB interface<br>• When the TPORT1 or TPORT0 signal changes states |
| xINT | Interrupt signal | 31 | O | Interrupt signal to the CPU. The initial value is Hi-z/0. This can be set to 1 or 0. |
| xWAIT | Wait signal | 29 | O | Wait signal to the CPU. The initial value is Hi-z/0. This can be set to 1 or 0. |
| OSCOUT | Oscillator output | 49 | O | CPU clock output. The frequency generated by the resonator connected to XI and XO pins is output from this pin. |

## 5.2 IDE/General-purpose Port Interface

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| HDD15<br>HDD14<br>HDD13<br>HDD12<br>HDD11<br>HDD10<br>HDD9<br>HDD8<br>HDD7<br>HDD6<br>HDD5<br>HDD4<br>HDD3<br>HDD2<br>HDD1<br>HDD0 | IDE Data/<br>Universal Data Bus | 69<br>67<br>65<br>63<br>60<br>58<br>56<br>54<br>53<br>55<br>57<br>59<br>62<br>64<br>66<br>68 | I/O<br>(3 state) | IDE data bus. Uses a 5V tolerant cell.<br>This bus also serves as a general-purpose port data bus, depending on the settings of the internal registers. If set to serve as a general-purpose port, the bus width can be set to 8 or 16 bits. |
| HDA2<br>HDA1<br>HDA0 | IDE<br>Register Address | 82<br>79<br>81 | O | IDE register address signal. Uses a 5V tolerant cell.<br>If the general-purpose port function is selected, this signal is not used. |
| xHCS1 | Control register select | 84 | O | Chip select for control register access<br>If the general-purpose port function is selected, this signal is not used. |
| xHCS0 | Command register select | 83 | O | Chip select for command block register access<br>If the general-purpose port function is selected, this signal is not used. |
| xHIOR | IDE Read strobe | 73 | O (IDE mode/port master mode)<br>I (port slave mode) | IDE read strobe. Uses a 5V tolerant cell. |
| xHIOW | IDE Write strobe | 72 | | IDE write strobe. Uses a 5V tolerant cell.<br>During IDE, this signal and xHIOR both serve as outputs. If the general-purpose port function is selected, this signal serves as output during master mode and serves as input during slave mode, depending on the settings of the internal registers. For detailed information on the signal timing, refer to the section, “AC Timing.” |
| HINTRQ | IDE interrupt request | 78 | I | IDE interrupt request Uses a 5V tolerant cell.<br>If the general-purpose port function is selected, this signal is not used. |
| HIORDY | I/O ready | 74 | I | IDE register ready signal Uses a 5V tolerant cell.<br>If the general-purpose port function is selected, this signal is not used. |
| HDMARQ | DMA request | 71 | I (IDE mode/port master mode)<br>O (port slave mode) | DMA transfer request Uses a 5V tolerant cell.<br>This signal serves as input during master mode and serves as output during slave mode depending on the settings of the internal registers. For detailed information on the signal timing, refer to the section, “AC Timing.” |

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| xHDMACK | DMA acknowledge | 77 | O (IDE mode/port master mode) I (port slave mode) | DMA transfer acknowledge Uses a 5V tolerant cell. This signal serves as output during master mode and serves as input during slave mode depending on the settings of the internal registers. For detailed information on the signal timing, refer to the section, "AC Timing." |
| xHPDIAG | Passed diagnostics | 80 | I | Diagnostic sequence-finished signal Uses a 5V tolerant cell. |
| xHDASP | Drive active/ Slave present | 85 | I | Drive active/slave drive present Uses a 5V tolerant cell. |
| xHRESET | IDE reset | 52 | O | IDE bus reset |



## 5.3 USB Interface

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| DP | USB positive signal | 8 | I/O | USB data line, Data+ |
| DM | USB negative signal | 10 | I/O | USB data line, Data- |
| VBUS | USB bus detect signal | 14 | I | USB bus detect signal. Uses a 5V tolerant cell. |
| R1 | Internal operation set pin | 1 | I/O | Internal operation set pin Connect a 6.2 kΩ ±1% resistor between this pin and AVSS (pin 5). This resistor must be connected as close as possible to pins 1 and 5. |
| XI | Resonator input | 99 | I | XI is an input for the internal oscillator circuit or input from an external oscillator. XO is an output for the internal oscillator circuit. Leave these pins open when using a crystal oscillator. |
| XO | Resonator output | 100 | O | |
| VC | PLLVCO | 96 | I | Internal test pin. During normal use, connect this pin to GND. |
| PTEST | PLL test | 97 | I | Internal test pin. During normal use, connect this pin to GND. |

## 5.4 System Blocks and Others

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| XRESET | Chip reset | 15 | I | Chip reset. Uses a 5V tolerant cell. |
| CLKSEL1 | Oscillator clock select | 88 | I | Oscillation frequency select |
| CLKSEL0 | | 87 | I | CLKSEL1 / CLKSEL0 / Frequency of the connected resonator<br>LOW / LOW / 12 MHz<br>LOW / HIGH / 16 MHz<br>HIGH / LOW / 20 MHz<br>HIGH / HIGH / 24 MHz |
| NC | No Connection | 2, 3, 89, 92, 94, 98 | — | These pins are not connected internally. |

## 5.5 Test Signals

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| TIN1<br>TIN0 | Test mode | 47<br>46 | I<br>(pull down) | Mode set input pin<br>00: Normal<br>Others: Internal test mode |
| TPORT1<br>TPORT0 | Test Port I/O | 45<br>44 | I/O | General-purpose input/output port for debugging |
| TSTEN | Internal test mode | 13 | I<br>(pull down) | Internal pulldown. During normal use, connect this pin low. |
| ATPGEN | Internal test mode | 42 | I<br>(pull down) | Test input pin. During normal use, connect this pin low. |
| SCANEN | Internal test mode | 43 | I<br>(pull down) | Test input pin. During normal use, connect this pin low. |



## 5.6 Power Supply and GND

| Symbol | Pin Name | Pin No. | Type | Description |
|---|---|---|---|---|
| $V_{DD}$ | Power supply for Logic part | 26, 41, 51, 70, 76 | P | 3.3V power supply pin for the logic block |
| $V_{SS}$ | Ground for Logic part | 25, 36, 48, 50, 61, 75, 86 | P | Ground pin for the logic block |
| $XV_{DD}$ | Power Supply for Logic Part in UTMI | 91 | P | 3.3V power supply pin for the transceiver macro unit logic block |
| $XV_{SS}$ | Ground for Logic part in UTMI | 90 | P | Ground pin for the transceiver macro unit logic block |
| $PV_{DD}$ | Power Supply for PLL Part in UTMI | 93 | P | 3.3V power supply pin for the transceiver macro unit PLL |
| $PV_{SS}$ | Ground for PLL Part in UTMI | 95 | P | Ground pin for the transceiver macro unit PLL |
| $AV_{DD}$ | Power Supply for Analog Part in UTMI | 4, 6, 12 | P | 3.3V power supply pin for the transceiver macro unit analog block |
| $AV_{SS}$ | Ground for Analog Part in UTMI | 5, 7, 9, 11 | P | Ground pin for the transceiver macro unit analog block. |

# 6. FUNCTIONAL DESCRIPTION

The function of each block of the S1R72003F00B100 is described below.

## 6.1 UTMI1.0 Transceiver Macro

This is a UTMI1.0-compliant USB 2.0 transceiver macro. It supports HS mode (480 Mbps) and FS mode (12 Mbps).

The transceiver macro contains an analog HS/FS driver, receiver, and terminator to provide a USB interface. It also contains an oscillator circuit which generates a 480 MHz clock required for HS transfer and a 60 MHz clock required for the operation of the internal logic. This oscillator circuit accepts as its input clock a 12, 16, 20, or 24 MHz crystal resonator. The transceiver macro uses an 8-bit parallel interface for interface with the SIE.

The transceiver macro processes the communication bit stream by NRZI encoding/decoding. It also has an internal data handler that adds SYNC, EOP, and bit stuff to the transmit data. When receiving data, it detects/removes SYNC and EOP and removes the bit stuff.



The transceiver macro incorporates the Elasticity Buffer to counter data underruns/overruns caused by frequency deviations on the data transmit/receive sides in the HS mode, and a squelch circuit to discriminate between serial data and noise.

For more information, refer to the UTMI 1.0 specifications.

## 6.2 Serial Interface Engine

### 6.2.1 Packet Handler

This unit processes the packet (by dissolving it into the various fields: PID, ADDR, DATA, CRC, Endpoint Number, and Frame Number). It also checks and generates CRC.

### 6.2.2 Transaction Manager

This unit manages transactions such as USB address verification and handshake verification/creation.

### 6.2.3 Endpoints

The serial interface incorporates Endpoint 0 (IN/OUT) and three general-purpose Endpoints (EPa, EPb, and EPc). The IN/OUT direction, maximum packet size, and transfer type (Bulk, Interrupt, or Isochronous) of the general-purpose Endpoints can be individually set using an internal register. (isochronous transfer is supported only by EPc.)

### 6.2.4 Suspend/Resume Controller

This unit controls Suspend and Resume.

## 6.3 FIFO SRAM

This buffer is used to accommodate the Endpoints (2.5 KB).

It is user programmable, but total 128 bytes comprised of 64 bytes for endpoint and 64 bytes for CBW/CSW are reserved area.

The FIFO SRAM reserves a space for MaxPacketSize (twice the size with the DoubleBuf setting) according to each endpoint setting. The amount of space the FIFO SRAM reserves must not exceed 2.5 kB.

## 6.4 FIFO Controller

This unit manages the FIFO SRAM address (user programmable), generates timing signals, and arbitrates bus contention.

## 6.5 DMA

The DMA in the S1R72003F00B100 supports general-purpose DMA ports and IDE interface.

The general-purpose DMA ports accommodate both master and slave operations. The bus width can be switched to 8 or 16 bits. The DMA can function as the IDE master, and supports PIO modes 0/1/2/3/4, Multiword DMA modes 0/1/2, and Ultra-DMA modes 0/1/2/3/4.

## 6.6 Test/Debug Module

The operation mode (test mode) of this module is switched by an input signal.

# 7. REGISTER

## 7.1 Register Map

Indicates the register or bit that can be read and/or written even if the controller is in the snooze mode.

| Address | Register Name | Reset | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | MainIntStat | 0x00 | SIEIntStat | BulkIntStat | EPrIntStat | IDEIntStat | | PortIntStat | | RcvEP0Setup |
| 0x01 | SIEIntStat | 0x00 | VBUSChanged | Non_J | DetectReset | DetectSuspend | RcvSOF | ChirpCmp | RestoreCmp | |
| 0x02 | BulkIntStat | 0x00 | CBWCmp | CBWShort | CBWLong | CBWErr | CSWCmp | CSWErr | BulkInCmp | BulkOutCmp |
| 0x03 | EPrIntStat | 0x00 | | | | | EP0IntStat | EPaIntStat | EPbIntStat | EPcIntStat |
| 0x04 | IDEIntStat | 0x00 | | | | | | DTCmp | DetectINTRQ | DetectTerm |
| 0x05 | (reserved) | 0xXX | | | | | | | | |
| 0x06 | PortIntStat | 0x00 | | | | | | | PortInt1 | PortInt0 |
| 0x07 | (reserved) | 0xXX | | | | | | | | |
| 0x08 | EP0IntStat | 0x00 | PingTranACK | | INTranACK | OUTTranACK | INTranNAK | OUTTranNAK | INTranErr | OUTTranErr |
| 0x09 | EPaIntStat | 0x00 | PingTranACK | OUTShortACK | INTranACK | OUTTranACK | INTranNAK | OUTTranNAK | INTranErr | OUTTranErr |
| 0x0A | EPbIntStat | 0x00 | PingTranACK | OUTShortACK | INTranACK | OUTTranACK | INTranNAK | OUTTranNAK | INTranErr | OUTTranErr |
| 0x0B | EPcIntStat | 0x00 | PingTranACK | OUTShortACK | INTranACK | OUTTranACK | INTranNAK | OUTTranNAK | INTranErr | OUTTranErr |
| 0x0C | (reserved) | 0xXX | | | | | | | | |
| 0x0D | (reserved) | 0xXX | | | | | | | | |
| 0x0E | (reserved) | 0xXX | | | | | | | | |
| 0x0F | (reserved) | 0xXX | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x10 | MainIntEnb | 0x00 | EnSIEIntStat | EnBulkIntStat | EnEPrIntStat | EnIDEIntStat | | EnPortIntStat | | EnRcvEP0 Setup |
| 0x11 | SIEIntEnb | 0x00 | EnVBUS Changed | EnNon_J | EnDetectReset | EnDetect Suspend | EnRcvSOF | EnChirpCmp | EnRestoreCmp | |
| 0x12 | BulkIntEnb | 0x00 | EnCBWCmp | EnCBWShort | EnCBWLong | EnCBWErr | EnCSWCmp | EnCSWErr | EnBulkInCmp | EnBulkOut Cmp |
| 0x13 | EPrIntEnb | 0x00 | | | | | EnEP0IntStat | EnEPaIntStat | EnEPbIntStat | EnEPcIntStat |
| 0x14 | IDEIntEnb | 0x00 | | | | | | EnDTCmp | EnDetectINTRQ | EnDetectTerm |
| 0x15 | (reserved) | 0xXX | | | | | | | | |
| 0x16 | PortIntEnb | 0x00 | | | | | | | EnPortInt1 | EnPortInt0 |
| 0x17 | (reserved) | 0xXX | | | | | | | | |
| 0x18 | EP0IntEnb | 0x00 | EnPingTran ACK | | EnINTranACK | EnOUTTran ACK | EnINTranNAK | EnOUTTran NAK | EnINTranErr | EnOUTTranErr |
| 0x19 | EPaIntEnb | 0x00 | EnPingTran ACK | EnOUTShort ACK | EnINTranACK | EnOUTTran ACK | EnINTranNAK | EnOUTTran NAK | EnINTranErr | EnOUTTranErr |
| 0x1A | EPbIntEnb | 0x00 | EnPingTran ACK | EnOUTShort ACK | EnINTranACK | EnOUTTran ACK | EnINTranNAK | EnOUTTran NAK | EnINTranErr | EnOUTTranErr |
| 0x1B | EPcIntEnb | 0x00 | EnPingTran ACK | EnOUTShort ACK | EnINTranACK | EnOUTTran ACK | EnINTranNAK | EnOUTTran NAK | EnINTranErr | EnOUTTranErr |
| 0x1C | (reserved) | 0xXX | | | | | | | | |
| 0x1D | (reserved) | 0xXX | | | | | | | | |
| 0x1E | (reserved) | 0xXX | | | | | | | | |
| 0x1F | (reserved) | 0xXX | | | | | | | | |

| Address | Register Name | Reset | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x20 | ChipReset | 0x00 | | | | | | | ResetSIE | ResetALL |
| 0x21 | RevisionNum | 0x31 | | | | | | | | |
| 0x22 | PMContorol | 0x00 | InSnooze | | | ResetUTM | PLLSel | SleepEnb | Analog Pwdown | Snooze |
| 0x23 | USBControl | 0x00 | DisBusDetect | | | | SendWakeup | RestoreUSB | GoChirp | ActiveUSB |
| 0x24 | USBStatus | 0xXX | VBUS | FSxHS | | | | | LineState[1:0] | |
| 0x25 | XcvrControl | 0x41 | TermSelect | XcvrSelect | | | | | OpMode[1:0] | |
| 0x26 | USBTest | 0x00 | EnHSTest | | | | SE0_NAK | TEST_J | TEST_K | TestPacket |
| 0x27 | (reserved) | 0xXX | | | | | | | | |
| 0x28 | USBAddress | 0x00 | | USBAddress[6:0] | | | | | | |
| 0x29 | EPrControl | 0x00 | DMARunning | | | ALLFIFOClr | | AutoEnShort | ALLForceNAK | EPrForce STALL |
| 0x2A | BulkOnlyControl | 0x00 | | | | | | GoCBWMode | GoCSWMode | CSWSel |
| 0x2B | BulkOnlyConfig | 0x00 | CBWEPNumber | | | | CSWEPNumber | | | |
| 0x2C | (reserved) | 0xXX | | | | | | | | |
| 0x2D | (reserved) | 0xXX | | | | | | | | |
| 0x2E | (reserved) | 0xXX | | | | | | | | |
| 0x2F | ChipConfig | 0x00 | RDYxWAIT | WaitMode | IntMode | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x30 | EP0Setup_0 | 0x00 | | | | | | | | |
| 0x31 | EP0Setup_1 | 0x00 | | | | | | | | |
| 0x32 | EP0Setup_2 | 0x00 | | | | | | | | |
| 0x33 | EP0Setup_3 | 0x00 | | | | | | | | |
| 0x34 | EP0Setup_4 | 0x00 | | | | | | | | |
| 0x35 | EP0Setup_5 | 0x00 | | | | | | | | |
| 0x36 | EP0Setup_6 | 0x00 | | | | | | | | |
| 0x37 | EP0Setup_7 | 0x00 | | | | | | | | |
| 0x38 | FrameNumber_H | 0x80 | FnInvalid | | | | | FrameNumber[10:8] | | |
| 0x39 | FrameNumber_L | 0x00 | FrameNumber[7:0] | | | | | | | |
| 0x3A | (reserved) | 0xXX | | | | | | | | |
| 0x3B | (reserved) | 0xXX | | | | | | | | |
| 0x3C | (reserved) | 0xXX | | | | | | | | |
| 0x3D | (reserved) | 0xXX | | | | | | | | |
| 0x3E | (reserved) | 0xXX | | | | | | | | |
| 0x3F | (reserved) | 0xXX | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x40 | EP0Config_0 | 0x00 | INxOUT | | | | | | | |
| 0x41 | (reserved) | 0xXX | | | | | | | | |
| 0x42 | EP0Control_0 | 0x00 | AutoForceNAK | InEnShortPkt | | | InForceNAK | InForceSTALL | OutForceNAK | OutForce STALL |
| 0x43 | EP0Control_1 | 0x00 | InToggleStat | | InToggleSet | InToggleClr | OutToggleStat | | OutToggleSet | OutToggleClr |
| 0x44 | (reserved) | 0xXX | | | | | | | | |
| 0x45 | EP0FIFORemain | 0x00 | | EP0FIFORemain Counter[6:0] | | | | | | |
| 0x46 | EP0FIFOforCPU | 0xXX | EP0FIFOData | | | | | | | |
| 0x47 | EP0FIFOControl | 0x80 | FIFOEmpty | FIFOFull | | | | FIFOClr | EnFIFOwr | EnFIFOrd |
| 0x48 | (reserved) | 0xXX | | | | | | | | |
| 0x49 | (reserved) | 0xXX | | | | | | | | |
| 0x4A | (reserved) | 0xXX | | | | | | | | |
| 0x4B | (reserved) | 0xXX | | | | | | | | |
| 0x4C | (reserved) | 0xXX | | | | | | | | |
| 0x4D | (reserved) | 0xXX | | | | | | | | |
| 0x4E | (reserved) | 0xXX | | | | | | | | |
| 0x4F | (reserved) | 0xXX | | | | | | | | |

| Address | Register Name | Reset | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x50 | EPaConfig_0 | 0x00 | INxOUT | | | | EndPointNumber[3:0] | | | |
| 0x51 | EPaConfig_1 | 0x00 | JoinIDE | ToggleMode | EnEndPoint | DoubleBuf | | MaxPacketSize[2:0] | | |
| 0x52 | EPaControl_0 | 0x00 | AutoForceNAK | EnShortPkt | AutoForceNAK Short | | | | ForceNAK | ForceSTALL |
| 0x53 | EPaControl_1 | 0x00 | | | | | ToggleStat | | ToggleSet | ToggleClr |
| 0x54 | EPaFIFORemain_H | 0x00 | EPaFIFORemain Counter[15:8] | | | | | | | |
| 0x55 | EPaFIFORemain_L | 0x00 | EPaFIFORemain Counter[7:0] | | | | | | | |
| 0x56 | EPaFIFOforCPU | 0xXX | EPaFIFOData | | | | | | | |
| 0x57 | EPaFIFOControl | 0x80 | FIFOEmpty | FIFOFull | | | | FIFOClr | EnFIFOwr | EnFIFOrd |
| 0x58 | EPbConfig_0 | 0x00 | INxOUT | | | | EndPointNumber[3:0] | | | |
| 0x59 | EPbConfig_1 | 0x00 | JoinIDE | ToggleMode | EnEndPoint | DoubleBuf | | MaxPacketSize[2:0] | | |
| 0x5A | EPbControl_0 | 0x00 | AutoForceNAK | EnShortPkt | AutoForceNAK Short | | | | ForceNAK | ForceSTALL |
| 0x5B | EPbControl_1 | 0x00 | | | | | ToggleStat | | ToggleSet | ToggleClr |
| 0x5C | EPbFIFORemain_H | 0x00 | EPbFIFORemain Counter[15:8] | | | | | | | |
| 0x5D | EPbFIFORemain_L | 0x00 | EPbFIFORemain Counter[7:0] | | | | | | | |
| 0x5E | EPbFIFOforCPU | 0xXX | EPbFIFOData | | | | | | | |
| 0x5F | EPbFIFOControl | 0x80 | FIFOEmpty | FIFOFull | | | | FIFOClr | EnFIFOwr | EnFIFOrd |



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x60 | EPcConfig_0 | 0x00 | INxOUT | ISO | | | EndPointNumber[3:0] | | | |
| 0x61 | EPcConfig_1 | 0x00 | JoinIDE | ToggleMode | EnEndPoint | DoubleBuf | | MaxPacketSize[2:0] | | |
| 0x62 | EPcControl_0 | 0x00 | AutoForceNAK | EnShortPkt | AutoForceNAK Short | | | | ForceNAK | ForceSTALL |
| 0x63 | EPcControl_1 | 0x00 | | | | | ToggleStat | | ToggleSet | ToggleClr |
| 0x64 | EPcFIFORemain_H | 0x00 | EPcFIFORemain Counter[15:8] | | | | | | | |
| 0x65 | EPcFIFORemain_L | 0x00 | EPcFIFORemain Counter[7:0] | | | | | | | |
| 0x66 | EPcFIFOforCPU | 0xXX | EPcFIFOData | | | | | | | |
| 0x67 | EPcFIFOControl | 0x80 | FIFOEmpty | FIFOFull | | | | FIFOClr | EnFIFOwr | EnFIFOrd |
| 0x68 | IsoMaxSize_H | 0x00 | | | | | | IsoMaxPacketSize[10:8] | | |
| 0x69 | IsoMaxSize_L | 0x00 | IsoMaxPacketSize[7:2] | | | | | | | |
| 0x6A | (reserved) | 0xXX | | | | | | | | |
| 0x6B | (reserved) | 0xXX | | | | | | | | |
| 0x6C | (reserved) | 0xXX | | | | | | | | |
| 0x6D | (reserved) | 0xXX | | | | | | | | |
| 0x6E | (reserved) | 0xXX | | | | | | | | |
| 0x6F | (reserved) | 0xXX | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x70 | (reserved) | 0xXX | | | | | | | | |
| 0x71 | (reserved) | 0xXX | | | | | | | | |
| 0x72 | (reserved) | 0xXX | | | | | | | | |
| 0x73 | (reserved) | 0xXX | | | | | | | | |
| 0x74 | (reserved) | 0xXX | | | | | | | | |
| 0x75 | (reserved) | 0xXX | | | | | | | | |
| 0x76 | (reserved) | 0xXX | | | | | | | | |
| 0x77 | (reserved) | 0xXX | | | | | | | | |
| 0x78 | (reserved) | 0xXX | | | | | | | | |
| 0x79 | (reserved) | 0xXX | | | | | | | | |
| 0x7A | (reserved) | 0xXX | | | | | | | | |
| 0x7B | (reserved) | 0xXX | | | | | | | | |
| 0x7C | (reserved) | 0xXX | | | | | | | | |
| 0x7D | (reserved) | 0xXX | | | | | | | | |
| 0x7E | (reserved) | 0xXX | | | | | | | | |
| 0x7F | (reserved) | 0xXX | | | | | | | | |

| Address | Register Name | Reset | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x80 | IDE_Status | 0xXX | DMARQ | DMACK | INTRQ | IORDY | | | PDIAG | DASP |
| 0x81 | IDE_Config_0 | 0x00 | IDEBusReset | | | | | NotIDE | Ultra | DMA |
| 0x82 | IDE_Config_1 | 0x00 | ActiveIDE | DelayStrobe | Slave | InterLock | PDREQLevel | Swap | | Bus8 |
| 0x83 | (reserved) | 0xXX | | | | | | | | |
| 0x84 | IDE_Rmod | 0x00 | RegisterAssertPulseWidth[3:0] | | | | RegisterNegatePulseWidth[3:0] | | | |
| 0x85 | IDE_Tmod | 0x00 | TransferAssertPulseWidth[3:0] | | | | TransferNegatePulseWidth[3:0] | | | |
| 0x86 | IDE_Umod | 0x00 | | | | | UltraDMACycle[3:0] | | | |
| 0x87 | (reserved) | 0xXX | | | | | | | | |
| 0x88 | IDE_Control_0 | 0x00 | IDEFlush | IDEFClr | | | | | | DTGO |
| 0x89 | (reserved) | 0xXX | | | | | | | | |
| 0x8A | IDE_Count_H | 0x00 | Count[23:16] | | | | | | | |
| 0x8B | IDE_Count_M | 0x00 | Count[15:8] | | | | | | | |
| 0x8C | IDE_Count_L | 0x00 | Count[7:0] | | | | | | | |
| 0x8D | IDE_CRCControl | 0x00 | | | | | | | | Clear |
| 0x8E | IDE_CRC_H | 0x4A | CRC[15:8] | | | | | | | |
| 0x8F | IDE_CRC_L | 0xBA | CRC[7:0] | | | | | | | |

| | | | |
|---|---|---|---|
| 0x90 | IDE_CS00 | 0xXX | |
| 0x91 | IDE_CS01 | 0xXX | |
| 0x92 | IDE_CS02 | 0xXX | |
| 0x93 | IDE_CS03 | 0xXX | |
| 0x94 | IDE_CS04 | 0xXX | |
| 0x95 | IDE_CS05 | 0xXX | |
| 0x96 | IDE_CS06 | 0xXX | |
| 0x97 | IDE_CS07 | 0xXX | |
| 0x98 | IDE_CS10 | 0xXX | |
| 0x99 | IDE_CS11 | 0xXX | |
| 0x9A | IDE_CS12 | 0xXX | |
| 0x9B | IDE_CS13 | 0xXX | |
| 0x9C | IDE_CS14 | 0xXX | |
| 0x9D | IDE_CS15 | 0xXX | |
| 0x9E | IDE_CS16 | 0xXX | |
| 0x9F | IDE_CS17 | 0xXX | |

| | | | |
|---|---|---|---|
| 0xA0 | CBW_00 | 0xXX | |
| 0xA1 | CBW_01 | 0xXX | |
| 0xA2 | CBW_02 | 0xXX | |
| 0xA3 | CBW_03 | 0xXX | |
| 0xA4 | CBW_04 | 0xXX | |
| 0xA5 | CBW_05 | 0xXX | |
| 0xA6 | CBW_06 | 0xXX | |
| 0xA7 | CBW_07 | 0xXX | |
| 0xA8 | CBW_08 | 0xXX | |
| 0xA9 | CBW_09 | 0xXX | |
| 0xAA | CBW_10 | 0xXX | |
| 0xAB | CBW_11 | 0xXX | |
| 0xAC | CBW_12 | 0xXX | |
| 0xAD | CBW_13 | 0xXX | |
| 0xAE | CBW_14 | 0xXX | |
| 0xAF | CBW_15 | 0xXX | |

| Address | Register Name | Reset | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xB0 | CBW_16 | 0xXX | | | | | | | | |
| 0xB1 | CBW_17 | 0xXX | | | | | | | | |
| 0xB2 | CBW_18 | 0xXX | | | | | | | | |
| 0xB3 | CBW_19 | 0xXX | | | | | | | | |
| 0xB4 | CBW_20 | 0xXX | | | | | | | | |
| 0xB5 | CBW_21 | 0xXX | | | | | | | | |
| 0xB6 | CBW_22 | 0xXX | | | | | | | | |
| 0xB7 | CBW_23 | 0xXX | | | | | | | | |
| 0xB8 | CBW_24 | 0xXX | | | | | | | | |
| 0xB9 | CBW_25 | 0xXX | | | | | | | | |
| 0xBA | CBW_26 | 0xXX | | | | | | | | |
| 0xBB | CBW_27 | 0xXX | | | | | | | | |
| 0xBC | CBW_28 | 0xXX | | | | | | | | |
| 0xBD | CBW_29 | 0xXX | | | | | | | | |
| 0xBE | CBW_30 | 0xXX | | | | | | | | |
| 0xBF | (reserved) | 0xXX | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xC0 | CSW0_00 | 0xXX | | | | | | | | |
| 0xC1 | CSW0_01 | 0xXX | | | | | | | | |
| 0xC2 | CSW0_02 | 0xXX | | | | | | | | |
| 0xC3 | CSW0_03 | 0xXX | | | | | | | | |
| 0xC4 | CSW0_04 | 0xXX | | | | | | | | |
| 0xC5 | CSW0_05 | 0xXX | | | | | | | | |
| 0xC6 | CSW0_06 | 0xXX | | | | | | | | |
| 0xC7 | CSW0_07 | 0xXX | | | | | | | | |
| 0xC8 | CSW0_08 | 0xXX | | | | | | | | |
| 0xC9 | CSW0_09 | 0xXX | | | | | | | | |
| 0xCA | CSW0_10 | 0xXX | | | | | | | | |
| 0xCB | CSW0_11 | 0xXX | | | | | | | | |
| 0xCC | CSW0_12 | 0xXX | | | | | | | | |
| 0xCD | (reserved) | 0xXX | | | | | | | | |
| 0xCE | (reserved) | 0xXX | | | | | | | | |
| 0xCF | (reserved) | 0xXX | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xD0 | CSW1_00 | 0xXX | | | | | | | | |
| 0xD1 | CSW1_01 | 0xXX | | | | | | | | |
| 0xD2 | CSW1_02 | 0xXX | | | | | | | | |
| 0xD3 | CSW1_03 | 0xXX | | | | | | | | |
| 0xD4 | CSW1_04 | 0xXX | | | | | | | | |
| 0xD5 | CSW1_05 | 0xXX | | | | | | | | |
| 0xD6 | CSW1_06 | 0xXX | | | | | | | | |
| 0xD7 | CSW1_07 | 0xXX | | | | | | | | |
| 0xD8 | CSW1_08 | 0xXX | | | | | | | | |
| 0xD9 | CSW1_09 | 0xXX | | | | | | | | |
| 0xDA | CSW1_10 | 0xXX | | | | | | | | |
| 0xDB | CSW1_11 | 0xXX | | | | | | | | |
| 0xDC | CSW1_12 | 0xXX | | | | | | | | |
| 0xDD | (reserved) | 0xXX | | | | | | | | |
| 0xDE | (reserved) | 0xXX | | | | | | | | |
| 0xDF | (reserved) | 0xXX | | | | | | | | |

| Address | Register Name | Reset | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xE0 | PortDir | 0x00 | | | | | | | PortDir1 | PortDir0 |
| 0xE1 | PortData | 0xXX | | | | | | | PortData1 | PortData0 |
| 0xE2 | (reserved) | 0xXX | | | | | | | | |
| 0xE3 | (reserved) | 0xXX | | | | | | | | |
| 0xE4 | (reserved) | 0xXX | | | | | | | | |
| 0xE5 | (reserved) | 0xXX | | | | | | | | |
| 0xE6 | (reserved) | 0xXX | | | | | | | | |
| 0xE7 | (reserved) | 0xXX | | | | | | | | |
| 0xE8 | (reserved) | 0xXX | | | | | | | | |
| 0xE9 | (reserved) | 0xXX | | | | | | | | |
| 0xEA | (reserved) | 0xXX | | | | | | | | |
| 0xEB | (reserved) | 0xXX | | | | | | | | |
| 0xEC | (reserved) | 0xXX | | | | | | | | |
| 0xED | (reserved) | 0xXX | | | | | | | | |
| 0xEE | (reserved) | 0xXX | | | | | | | | |
| 0xEF | (reserved) | 0xXX | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xF0 | (reserved) | 0xXX | | | | | | | | |
| 0xF1 | (reserved) | 0xXX | | | | | | | | |
| 0xF2 | (reserved) | 0xXX | | | | | | | | |
| 0xF3 | (reserved) | 0xXX | | | | | | | | |
| 0xF4 | (reserved) | 0xXX | | | | | | | | |
| 0xF5 | (reserved) | 0xXX | | | | | | | | |
| 0xF6 | (reserved) | 0xXX | | | | | | | | |
| 0xF7 | (reserved) | 0xXX | | | | | | | | |
| 0xF8 | (reserved) | 0xXX | | | | | | | | |
| 0xF9 | (reserved) | 0xXX | | | | | | | | |
| 0xFA | (reserved) | 0xXX | | | | | | | | |
| 0xFB | (reserved) | 0xXX | | | | | | | | |
| 0xFC | (reserved) | 0xXX | | | | | | | | |
| 0xFD | (reserved) | 0xXX | | | | | | | | |
| 0xFE | (reserved) | 0xXX | | | | | | | | |
| 0xFF | (reserved) | 0xXX | | | | | | | | |

* Access to reserved registers is prohibited.

## 7.2 Register Details

### 7.2.1 00h Main Interrupt Status (MainIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 00h | MainIntStat | R | 7: SIEIntStat | 0: None | 1: SIE Interrupt Occurred | 00h |
| | | R | 6: BulkIntStat | 0: None | 1: Bulk Interrupt Occurred | |
| | | R | 5: EPrIntStat | 0: None | 1: EPr Interrupt Occurred | |
| | | R | 4: IDEIntStat | 0: None | 1: IDE Interrupt Occurred | |
| | | | 3: | 0: | 1: | |
| | | R | 2: PortIntStat | 0: None | 1: Port Interrupt Occurred | |
| | | | 1: | 0: | 1: | |
| | | R (W) | 0: RcvEP0Setup | 0: None | 1: Receive EP0 Setup Transaction | |



When an interrupt to the CPU is generated by the S1R72003F00B100, the CPU reads this register during interrupt handling to determine the cause of the interrupt. For bits that indirectly indicate the cause of the interrupt, the CPU reads the interrupt status register corresponding to one of the bits to determine the cause of the interrupt. If all interrupt causes in that interrupt status register are cleared, the corresponding bit in this register is automatically cleared. For bits that directly indicate the cause of the interrupt, it is possible to clear the cause of the interrupt by writing 1 to the corresponding bit.

If any bit in the MainIntEnb register is enabled for interrupt and the corresponding interrupt cause in this register is set to 1, the xINT pin is asserted to generate an interrupt to the CPU. When all of the corresponding interrupt causes are cleared, the xINT pin is negated.

**Bit 7** **SIEIntStat**
Indirectly indicates the cause of the interrupt. This bit is set to 1 when the cause of the interrupt exists in the SIEIntStat register and the corresponding SIEIntEnb register bit is enabled. This bit is effective even during snooze.

**Bit 6** **BulkIntStat**
Indirectly indicates the cause of the interrupt. This bit is set to 1 when the cause of the interrupt exists in the BulkIntStat register and the corresponding BulkIntEnb register bit is enabled.

**Bit 5** **EPrIntStat**
Indirectly indicates the cause of the interrupt. This bit is set to 1 when the cause of the interrupt exists in the EPrIntStat register and the corresponding EPrIntEnb register bit is enabled.

**Bit 4** **IDEIntStat**
Indirectly indicates the cause of the interrupt. This bit is set to 1 when the cause of the interrupt exists in the IDEIntStat register and the corresponding IDEIntEnb register bit is enabled.

**Bit 3** **Reserved**

**Bit 2** **PortIntStat**
Indirectly indicates the cause of the interrupt. This bit is set to 1 when the cause of the interrupt exists in the PortIntStat register and the corresponding PortIntEnb register bit is enabled.
This bit is effective even during snooze.

**Bit 1** **Reserved**

**Bit 0** **RcvEP0Setup**
Directly indicates the cause of the interrupt. This bit is set to 1 if the received data is stored in registers EP0Setup_0 through EP0Setup_7 after the setup stage at Endpoint 0 is completed. At the same time, in the EP0Control_0 register, the InForceSTALL and OutForceSTALL bits are automatically set to 0 and the InForceNAK and OutForceNAK bits are set to 1. The status of the InForceNAK, OutForceNAK, InForceSTALL and OutForceSTALL bits in the EP0Control_0 cannot be changed when the RcvEP0Setup bit is 1.

### 7.2.2 01h SIE Interrupt Status (SIEIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 01h | SIEIntStat | R (W) | 7: VBUSChanged | 0: None | 1: VBUS Changed | 00h |
| | | R (W) | 6: Non_J | 0: None | 1: Non J Interrupt Occurred | |
| | | R (W) | 5: DetectReset | 0: None | 1: USB Reset Detected | |
| | | R (W) | 4: DetectSuspend | 0: None | 1: USB Suspend Detected | |
| | | R (W) | 3: RcvSOF | 0: None | 1: Received SOF Token | |
| | | R (W) | 2: ChirpCmp | 0: None | 1: Chirp Complete | |
| | | R (W) | 1: RestoreCmp | 0: None | 1: Restore Complete | |
| | | | 0: | 0: | 1: | |

This register shows SIE-related interrupts. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.



**Bit 7** **VBUSChanged**
This bit is set to 1 when the VBUS pin status changes.
You can check the VBUS status with the VBUS bit in the USBStatus register. If this bit is set to 1 during GoChirp or RestoreUSB processing and the VBUS bit is 0, the cable is disconnected. Clear the GoChirp and RestoreUSB bits to 0 to abort the current processing. This bit is effective even during snooze.

**Bit 6** **Non_J**
This bit is set to 1 when a state other than J is detected on the USB bus during snooze.
The state (SE0 or K) can be determined by inspecting the USBStatus register LineState bit. According to the state identified, remove the S1R72003F00B100 from the snooze state, then perform GoChirp or RestoreUSB processing.
This bit is effective only during snooze.

**Bit 5** **DetectReset**
This bit is set to 1 when the USB reset state is detected.
When the USB is operating in the HS operation mode, it enters the FS operation mode for detecting a USB reset state.
When this bit is set to 1, set the USBControl register DisBusDetect bit to 1 to disable USB reset/suspend state detection.
The DisBusDetect bit should be cleared to 0 to enable USB reset/suspend state detection after the reset processing is completed.
HS Detection Handshake can be initiated using the USBControl register GoChirp bit.
The USB reset detection is effective when the USBControl register ActiveUSB bit is set to 1.

**Bit 4** **DetectSuspend**
This bit is set to 1 when the USB suspend state is detected.
When the USB is operating in the HS operation mode, it enters the FS operation mode for detecting a USB reset state.
After the USB suspend state is detected, the PLL oscillation in the S1R72003F00B100 can be halted (set to the snooze mode) by setting the PMControl register Snooze bit to 1.

**Bit 3** **RcvSOF**
This bit is set to 1 when an SOF token is received.

**Bit 2** **ChirpCmp**
This bit is set to 1 when HS Detection Handshake initiated by the USBControl register GoChirp bit finishes.
Following this interrupt, the current USB operation mode (FS or HS) can be determined by reading the USBStatus register FSxHS bit.

**Bits 1** **RestoreCmp**
This bit is set to 1 when the Resume processing initiated by the USBControl register RestoreUSB bit finishes. When this bit is set to 1, the USB returns to its operation mode (FS or HS) before being suspended.

**Bits 0** **Reserved**

### 7.2.3 02h Bulk Interrupt Status (BulkIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 02h | BulkIntStat | R (W) | 7: CBWCmp | 0: None | 1: CBW Packet Received | 00h |
| | | R (W) | 6: CBWShort | 0: None | 1:CBWShortPacketReceived | |
| | | R (W) | 5: CBWLong | 0: None | 1:CBWLongPacketReceived | |
| | | R (W) | 4: CBWErr | 0: None | 1: CBW Error | |
| | | R (W) | 3: CSWCmp | 0: None | 1: CSW Transfer Complete | |
| | | R (W) | 2: CSWErr | 0: None | 1: CSW Error | |
| | | R (W) | 1: BulkInCmp | 0: None | 1: Bulk In Transfer Complete | |
| | | R (W) | 0: BulkOutCmp | 0: None | 1:BulkOutTransfer Complete | |



This register shows the Bulk transfer related interrupts. The CBWCmp, CBWShort, CBWLong, CBWErr, CSWCmp, and CSWErr bits are used in the USB storage-class BulkOnly transport protocol. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.

**Bit 7** **CBWCmp**
This bit is set to 1 if while the BulkOnlyControl register GoCBWMode bit = 1, 31 bytes long data is received normally at the endpoint set in the BulkOnlyConfig register CBWEPNumber.

**Bit 6** **CBWShort**
This bit is set to 1 if while the BulkOnlyControl register GoCBWMode bit = 1, data of less than 31 bytes in length is received at the endpoint set in the BulkOnlyConfig register CBWEPNumber.

**Bit 5** **CBWLong**
This bit is set to 1 if while the BulkOnlyControl register GoCBWMode bit = 1, data of more than 31 bytes in length is received at the endpoint set in the BulkOnlyConfig register CBWEPNumber.

**Bit 4** **CBWErr**
When the BulkOnlyControl register GoCBWMode bit is 1, this bit is set to 1 if a transaction error occurred at an endpoint set in the BulkOnlyConfig register CBWEPNumber bits.

**Bit 3** **CSWCmp**
When the BulkOnlyControl register GoCSWMode bit is 1, this bit is set to 1 if an IN transaction is executed at an endpoint set in the BulkOnlyConfig register CSWEPNumber bits and the S1R72003F00B100 receives an ACK from the host in response to the data on registers CSW0_00 through CSW0_12 or CSW1_00 through CSW1_12 sent to the host.

**Bit 2** **CSWErr**
When the BulkOnlyControl register GoCSWMode bit is 1, this bit is set to 1 if an IN transaction is executed at an endpoint set in the BulkOnlyConfig register CSWEPNumber bits and the S1R72003F00B100 receives no ACK from the host in response to the data on registers CSW0_00 through CSW0_12 or CSW1_00 through CSW1_12 sent to the host.

**Bit 1** **BulkInCmp**
This bit is set to 1 when the DMA transfer for the number of bytes specified as the DMA transfer size completes in an IN transaction at an endpoint where the EP[a,b,c]Config_1 register JoinIDE bit is 1, completing the transfer of all data in the FIFO. The time at which this interrupt is generated depends on the EPrControl register AutoENShort bit. For more information, refer to the description of the EPrControl register AutoENShort bit.

**Bit 0** **BulkOutCmp**
This bit is set to 1 together with the DTCmp bit when the DMA transfer for the number of bytes specified as the DMA transfer size completes in an OUT transaction at an endpoint where the EP[a,b,c]Config_1 register JoinIDE bit is 1.

### 7.2.4 03h EPr Interrupt Status (EPrIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 03h | EPrIntStat | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R | 3: EP0IntStat | 0: None | 1: EP0 Interrupt Occurred | |
| | | R | 2: EPaIntStat | 0: None | 1: EPa Interrupt Occurred | |
| | | R | 1: EPbIntStat | 0: None | 1: EPb Interrupt Occurred | |
| | | R | 0: EPcIntStat | 0: None | 1: EPc Interrupt Occurred | |

This register indirectly indicates the cause of the interrupt for each endpoint. When all the enabled interrupt causes (root causes) at an endpoint indicated by a bit are cleared, that bit is cleared.



**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **EP0IntStat**
This bit is set to 1 when the cause of the interrupt exists in the EP0IntStat register, and the bit in the EP0IntEnb register corresponding to that cause of the interrupt is enabled for interrupt.

**Bit 2** **EPaIntStat**
This bit is set to 1 when the cause of the interrupt exists in the EPaIntStat register, and the bit in the EPaIntEnb register corresponding to that cause of the interrupt is enabled for interrupt.

**Bit 1** **EPbIntStat**
This bit is set to 1 when the cause of the interrupt exists in the EPbIntStat register, and the bit in the EPbIntEnb register corresponding to that cause of the interrupt is enabled for interrupt.

**Bit 0** **EPcIntStat**
This bit is set to 1 when the cause of the interrupt exists in the EPcIntStat register, and the bit in the EPcIntEnb register corresponding to that cause of the interrupt is enabled for interrupt.

### 7.2.5 04h IDE Interrupt Status (IDEIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 04h | IDEIntStat | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | R (W) | 2: DTCmp | 0: None | 1: DMA Transfer Complete | |
| | | R (W) | 1: DetectINTRQ | 0: None | 1: INTRQ Detected | |
| | | R (W) | 0: DetectTerm | 0: None | 1: Terminate Detected | |

This register shows the interrupts from the IDE interface. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.



**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **DTCmp**
This bit is set to 1 when the DMA transfer activated by the IDE_Control_0 register DTGO bit finishes by transferring as many bytes as specified by the DMA transfer size. Also, this bit is set to 1 when the DMA transfer is forcibly terminated by writing a 0 to the IDE_Control_0 register DTGO bit.

**Bit 1** **DetectINTRQ**
This bit is set to 1 when the leading edge of the HINTRQ signal on the IDE interface is detected.

**Bit 0** **DetectTerm**
This bit is set to 1 simultaneously with the DTCMP bit and the transfer is aborted if the device negates HDMARQ during ULTRA DMA transfer in IDE.

### 7.2.6 05h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 05h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.7 06h Port Interrupt Status (PortIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 06h | PortIntStat | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R (W) | 1: PortInt1 | 0: None | 1:Port1Input signal Changed | |
| | | R (W) | 0: PortInt0 | 0: None | 1:Port0Input signal Changed | |



This register shows general-purpose IO port interrupts. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.

**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **PortInt1**
This bit is set to 1 if while Port1 is set for input, the input signal on Port1 changes state.
This bit is effective even during snooze.

**Bit 0** **PortInt0**
This bit is set to 1 if while Port0 is set for input, the input signal on Port0 changes state.
This bit is effective even during snooze.

### 7.2.8 07h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 07h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.9 08h EP0 Interrupt Status (EP0IntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 08h | EP0IntStat | R (W) | 7: PingTranACK | 0: None | 1: Ping Transaction ACK | 00h |
| | | | 6: | 0: None | 1: | |
| | | R (W) | 5: INTranACK | 0: None | 1: IN Transaction ACK | |
| | | R (W) | 4: OUTTranACK | 0: None | 1: OUT Transaction ACK | |
| | | R (W) | 3: INTranNAK | 0: None | 1: IN Transaction NAK | |
| | | R (W) | 2: OUTTranNAK | 0: None | 1: OUT Transaction NAK | |
| | | R (W) | 1: INTranErr | 0: None | 1: IN Transaction Error | |
| | | R (W) | 0: OUTTranErr | 0: None | 1: OUT Transaction Error | |

This register shows the Endpoint 0 interrupt status. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.



**Bit 7** **PingTranACK**
This bit is set to 1 when an ACK is returned to the host in a Ping transaction.

**Bit 6** **Reserved**

**Bit 5** **INTranACK**
This bit is set to 1 when an ACK is received from the host in an IN transaction.

**Bit 4** **OUTTranACK**
This bit is set to 1 when an ACK is returned to the host in an OUT transaction.

**Bit 3** **INTranNAK**
This bit is set to 1 when an NAK is returned to the host in an IN transaction.

**Bit 2** **OUTTranNAK**
This bit is set to 1 when an NAK is returned to the host for an OUT or PING transaction.

**Bit 1** **INTranErr**
This bit is set to 1 when either a STALL is returned to the host, a packet error occurs or a handshake times out in an IN transaction.

**Bit 0** **OUTTranErr**
This bit is set to 1 when a STALL is returned to the host or a packet error is found in an OUT transaction.

### 7.2.10 09h EPa Interrupt Status (EPaIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 09h | EPaIntStat | R (W) | 7: PingTranACK | 0: None | 1: Ping Transaction ACK | 00h |
| | | R (W) | 6: OUTShortACK | 0: None | 1: OUT Short Packet ACK | |
| | | R (W) | 5: INTranACK | 0: None | 1: IN Transaction ACK | |
| | | R (W) | 4: OUTTranACK | 0: None | 1: OUT Transaction ACK | |
| | | R (W) | 3: INTranNAK | 0: None | 1: IN Transaction NAK | |
| | | R (W) | 2: OUTTranNAK | 0: None | 1: OUT Transaction NAK | |
| | | R (W) | 1: INTranErr | 0: None | 1: IN Transaction Error | |
| | | R (W) | 0: OUTTranErr | 0: None | 1: OUT Transaction Error | |

This register shows the Endpoint a interrupt status. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.



**Bit 7** **PingTranACK**
This bit is set to 1 when an ACK is returned to the host in a Ping transaction.

**Bit 6** **OUTShortACK**
This bit is set to 1 together with the OUTTranACK bit when a short packet is received and an ACK is returned in an OUT transaction.

**Bit 5** **INTranACK**
This bit is set to 1 when an ACK is received from the host in an IN transaction.

**Bit 4** **OUTTranACK**
This bit is set to 1 when an ACK is returned to the host in an OUT transaction.

**Bit 3** **INTranNAK**
This bit is set to 1 when an NAK is returned to the host in an IN transaction.

**Bit 2** **OUTTranNAK**
This bit is set to 1 when an NAK is returned to the host for an OUT or PING transaction.

**Bit 1** **INTranErr**
This bit is set to 1 when either a STALL is returned to the host, a packet error occurs or a handshake times out in an IN transaction.

**Bit 0** **OUTTranErr**
This bit is set to 1 when a STALL is returned to the host or a packet error is found in an OUT transaction.

### 7.2.11 0Ah EPb Interrupt Status (EPbIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 0Ah | EPbIntStat | R (W) | 7: PingTranACK | 0: None | 1: Ping Transaction ACK | 00h |
| | | R (W) | 6: OUTShortACK | 0: None | 1: OUT Short Packet ACK | |
| | | R (W) | 5: INTranACK | 0: None | 1: IN Transaction ACK | |
| | | R (W) | 4: OUTTranACK | 0: None | 1: OUT Transaction ACK | |
| | | R (W) | 3: INTranNAK | 0: None | 1: IN Transaction NAK | |
| | | R (W) | 2: OUTTranNAK | 0: None | 1: OUT Transaction NAK | |
| | | R (W) | 1: INTranErr | 0: None | 1: IN Transaction Error | |
| | | R (W) | 0: OUTTranErr | 0: None | 1: OUT Transaction Error | |

This register shows the Endpoint b interrupt status. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.



**Bit 7** **PingTranACK**
This bit is set to 1 when an ACK is returned to the host in a Ping transaction.

**Bit 6** **OUTShortACK**
This bit is set to 1 together with the OUTTranACK bit when a short packet is received and an ACK is returned in an OUT transaction.

**Bit 5** **INTranACK**
This bit is set to 1 when an ACK is received from the host in an IN transaction.

**Bit 4** **OUTTranACK**
This bit is set to 1 when an ACK is returned to the host in an OUT transaction.

**Bit 3** **INTranNAK**
This bit is set to 1 when an NAK is returned to the host in an IN transaction.

**Bit 2** **OUTTranNAK**
This bit is set to 1 when an NAK is returned to the host for an OUT or PING transaction.

**Bit 1** **INTranErr**
This bit is set to 1 when either a STALL is returned to the host, a packet error occurs or a handshake times out in an IN transaction.

**Bit 0** **OUTTranErr**
This bit is set to 1 when a STALL is returned to the host or a packet error is found in an OUT transaction.

### 7.2.12 0Bh EPc Interrupt Status (EPcIntStat)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 0Bh | EPcIntStat | R (W) | 7: PingTranACK | 0: None | 1: Ping Transaction ACK | 00h |
| | | R (W) | 6: OUTShortACK | 0: None | 1: OUT Short Packet ACK | |
| | | R (W) | 5: INTranACK | 0: None | 1: IN Transaction ACK | |
| | | R (W) | 4: OUTTranACK | 0: None | 1: OUT Transaction ACK | |
| | | R (W) | 3: INTranNAK | 0: None | 1: IN Transaction NAK | |
| | | R (W) | 2: OUTTranNAK | 0: None | 1: OUT Transaction NAK | |
| | | R (W) | 1: INTranErr | 0: None | 1: IN Transaction Error | |
| | | R (W) | 0: OUTTranErr | 0: None | 1: OUT Transaction Error | |

This register shows the Endpoint c interrupt status. The bits in this register directly indicate the cause of the interrupt. When a bit in this register is set to 1, writing a 1 to the bit can clear the cause of the interrupt.



**Bit 7** **PingTranACK**
This bit is set to 1 when an ACK is returned to the host in a Ping transaction.

**Bit 6** **OUTShortACK**
This bit is set to 1 together with the OUTTranACK bit when a short packet is received and an ACK is returned in an OUT transaction.

**Bit 5** **INTranACK**
This bit is set to 1 when an ACK is received from the host in an IN transaction.

**Bit 4** **OUTTranACK**
This bit is set to 1 when an ACK is returned to the host in an OUT transaction.

**Bit 3** **INTranNAK**
This bit is set to 1 when an NAK is returned to the host in an IN transaction.

**Bit 2** **OUTTranNAK**
This bit is set to 1 when an NAK is returned to the host for an OUT or PING transaction.

**Bit 1** **INTranErr**
This bit is set to 1 when either a STALL is returned to the host, a packet error occurs or a handshake times out in an IN transaction.

**Bit 0** **OUTTranErr**
This bit is set to 1 when a STALL is returned to the host or a packet error is found in an OUT transaction.

### 7.2.13 0Ch to 0Fh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 0Ch to 0Fh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.14 10h Main Interrupt Enable (MainIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 10h | MainIntEnb | R/W | 7: EnSIEIntStat | 0: Disable | 1: Enable | 00h |
| | | R/W | 6: EnBulkIntStat | 0: Disable | 1: Enable | |
| | | R/W | 5: EnEPrIntStat | 0: Disable | 1: Enable | |
| | | R/W | 4: EnIDEIntStat | 0: Disable | 1: Enable | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: EnPortIntStat | 0: Disable | 1: Enable | |
| | | | 1: | 0: | 1: | |
| | | R/W | 0: EnRcvEP0Setup | 0: Disable | 1: Enable | |

This register enables or disables the assertion of interrupt signals to the CPU in the MainIntStat register.
Setting any bit in this register to 1 enables the corresponding interrupt to the CPU.
The EnSIEIntStat and EnPortIntStat bits are effective even during snooze.

### 7.2.15 11h SIE Interrupt Enable (SIEIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 11h | SIEIntEnb | R/W | 7: EnVBUSChanged | 0: Disable | 1: Enable | 00h |
| | | R/W | 6: EnNon_J | 0: Disable | 1: Enable | |
| | | R/W | 5: EnDetectReset | 0: Disable | 1: Enable | |
| | | R/W | 4: EnDetectSuspend | 0: Disable | 1: Enable | |
| | | R/W | 3: EnRcvSOF | 0: Disable | 1: Enable | |
| | | R/W | 2: EnChirpCmp | 0: Disable | 1: Enable | |
| | | R/W | 1: EnRestoreCmp | 0: Disable | 1: Enable | |
| | | | 0: | 0: | 1: | |



This register enables or disables the causes of the interrupts in the SIEIntStat register. When a bit is set to 1, the MainIntStat register SIEIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.
The EnVBUSChanged and EnNon_J bits are effective even during snooze.

### 7.2.16 12h Bulk Interrupt Enable (BulkIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 12h | BulkIntEnb | R/W | 7: EnCBWCmp | 0: Disable | 1: Enable | 00h |
| | | R/W | 6: EnCBWShort | 0: Disable | 1: Enable | |
| | | R/W | 5: EnCBWLong | 0: Disable | 1: Enable | |
| | | R/W | 4: EnCBWErr | 0: Disable | 1: Enable | |
| | | R/W | 3: EnCSWCmp | 0: Disable | 1: Enable | |
| | | R/W | 2: EnCSWErr | 0: Disable | 1: Enable | |
| | | R/W | 1: EnBulkInCmp | 0: Disable | 1: Enable | |
| | | R/W | 0: EnBulkOutCmp | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the BulkIntStat register. When a bit is set to 1, the MainIntStat register BulkIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.

### 7.2.17 13h EPr Interrupt Enable (EPrIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 13h | EPrIntEnb | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: EnEP0IntStat | 0: Disable | 1: Enable | |
| | | R/W | 2: EnEPaIntStat | 0: Disable | 1: Enable | |
| | | R/W | 1: EnEPbIntStat | 0: Disable | 1: Enable | |
| | | R/W | 0: EnEPcIntStat | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the EPrIntStat register. When a bit is set to 1, the MainIntStat register EPrIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.

### 7.2.18 14h IDE Interrupt Enable (IDEIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 14h | IDEIntEnb | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: EnDTCmp | 0: Disable | 1: Enable | |
| | | R/W | 1: EnDetectINTRQ | 0: Disable | 1: Enable | |
| | | R/W | 0: EnDetectTerm | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the IDEIntStat register. When a bit is set to 1, the MainIntStat register IDEIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.

### 7.2.19 15h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 15h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.20 16h Port Interrupt Enable (PortIntEnb)



| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 16h | PortIntEnb | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: EnPortInt1 | 0: Disable | 1: Enable | |
| | | R/W | 0: EnPortInt0 | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the PortIntStat register. When a bit is set to 1, the MainIntStat register PortIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.
This register is effective even during snooze.

### 7.2.21 17h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 17h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.22 18h EP0 Interrupt Enable (EP0IntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 18h | EP0IntEnb | R/W | 7: EnPingTranACK | 0: Disable | 1: Enable | 00h |
| | | | 6: | 0: | 1: | |
| | | R/W | 5: EnINTranACK | 0: Disable | 1: Enable | |
| | | R/W | 4: EnOUTTranACK | 0: Disable | 1: Enable | |
| | | R/W | 3: EnINTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 2: EnOUTTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 1: EnINTranErr | 0: Disable | 1: Enable | |
| | | R/W | 0: EnOUTTranErr | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the EP0IntStat register. When a bit is set to 1, the EPrIntStat register EP0IntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.

### 7.2.23 19h EPa Interrupt Enable (EPaIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 19h | EPaIntEnb | R/W | 7: EnPingTranACK | 0: Disable | 1: Enable | 00h |
| | | R/W | 6: EnOUTShortACK | 0: Disable | 1: Enable | |
| | | R/W | 5: EnINTranACK | 0: Disable | 1: Enable | |
| | | R/W | 4: EnOUTTranACK | 0: Disable | 1: Enable | |
| | | R/W | 3: EnINTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 2: EnOUTTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 1: EnINTranErr | 0: Disable | 1: Enable | |
| | | R/W | 0: EnOUTTranErr | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the EPaIntStat register. When a bit is set to 1, the EPrIntStat register EPaIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.



### 7.2.24 1Ah EPb Interrupt Enable (EPbIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 1Ah | EPbIntEnb | R/W | 7: EnPingTranACK | 0: Disable | 1: Enable | 00h |
| | | R/W | 6: EnOUTShortACK | 0: Disable | 1: Enable | |
| | | R/W | 5: EnINTranACK | 0: Disable | 1: Enable | |
| | | R/W | 4: EnOUTTranACK | 0: Disable | 1: Enable | |
| | | R/W | 3: EnINTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 2: EnOUTTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 1: EnINTranErr | 0: Disable | 1: Enable | |
| | | R/W | 0: EnOUTTranErr | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the EPbIntStat register. When a bit is set to 1, the EPrIntStat register EPbIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.

### 7.2.25 1Bh EPc Interrupt Enable (EPcIntEnb)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 1Bh | EPcIntEnb | R/W | 7: EnPingTranACK | 0: Disable | 1: Enable | 00h |
| | | R/W | 6: EnOUTShortACK | 0: Disable | 1: Enable | |
| | | R/W | 5: EnINTranACK | 0: Disable | 1: Enable | |
| | | R/W | 4: EnOUTTranACK | 0: Disable | 1: Enable | |
| | | R/W | 3: EnINTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 2: EnOUTTranNAK | 0: Disable | 1: Enable | |
| | | R/W | 1: EnINTranErr | 0: Disable | 1: Enable | |
| | | R/W | 0: EnOUTTranErr | 0: Disable | 1: Enable | |

This register enables or disables the causes of the interrupts in the EPcIntStat register. When a bit is set to 1, the EPrIntStat register EPcIntStat bit will be set to 1 when the corresponding cause of the interrupt occurs.

### 7.2.26 1Ch to 1Fh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 1Ch to 1Fh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.27 20h Chip Reset (ChipReset)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 20h | ChipReset | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | W | 1: ResetSIE | 0: Normal | 1: SIE Reset | |
| | | W | 0: ResetALL | 0: Normal | 1: ALL Reset | |



This register resets the S1R72003F00B100.

**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **ResetSIE**
Resets the SIE block of the S1R72003F00B100. Setting this bit to 1 resets the USBAddress register, EP0Setup_0 through 7 registers, and the FrameNumber_H and L registers to their initial values. This bit is automatically cleared to 0 upon completion of initialization.

**Bit 0** **ResetALL**
Resets the sequencer of the S1R72003F00B100. Setting this bit to 1 resets all but a few registers to their initial values. This bit is automatically cleared to 0 upon completion of initialization. The register bits that can be accessed during snooze and the MainIntEnb register are not reset.

### 7.2.28 21h Revision Number (RevisionNum)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 21h | RevisionNum | R | 7: RevisionNum[7] | Revision Number | 31h |
| | | | 6: RevisionNum[6] | | |
| | | | 5: RevisionNum[5] | | |
| | | | 4: RevisionNum[4] | | |
| | | | 3: RevisionNum[3] | | |
| | | | 2: RevisionNum[2] | | |
| | | | 1: RevisionNum[1] | | |
| | | | 0: RevisionNum[0] | | |

This register shows the revision number of the S1R72003F00B100.
This register is effective even during snooze.

### 7.2.29 22h Power Management Control (PMControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 22h | PMControl | R | 7: InSnooze | 0: Normal | 1: In Snooze | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | R/W | 4: ResetUTM | 0: Normal | 1: UTMI Reset | |
| | | R/W | 3: PLLSel | 0: Select PLL480MHz | 1: Select PLL60MHz | |
| | | R/W | 2: SleepEnb | 0: Disable OSC Stop at Sleep | 1:Enable OSC Stop at Sleep | |
| | | R/W | 1: AnalogPwdown | 0: Disable Analog Power down | 1:Enable Analog Power down | |
| | | R/W | 0: Snooze | 0: Normal | 1: Snooze | |



This register sets the power management-related operations of the S1R72003F00B100.
This register is effective even during snooze.

**Bit 7** **InSnooze**
This bit is set to 1 when the S1R72003F00B100 is placed in the snooze state by the PMControl register Snooze bit. This bit is reset to 0 when CLK output stabilizes after the S1R72003F00B100 is freed from the snooze state by clearing the Snooze bit to 0.

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **ResetUTM**
The UTM block of the S1R72003F00B100 can be reset by setting this bit to 1. To deactivate the reset state, clear this bit to 0.

**Bit 3** **PLLSel**
Sets one of two PLL types in the S1R72003F00B100.
0: Select PLL 480 MHz.
1: Select PLL 60 MHz.

PLL480 is available in any mode. PLL60 is available in the FS mode only.
Before starting Chirp after detecting the USB reset state, select PLL 480 MHz by setting this bit to 0.

**Bit 2** **SleepEnb**
Sets whether or not to halt the oscillation circuit when the xSLEEP pin is activated during snooze.
0: Do not halt the oscillation circuit when the xSLEEP pin is activated during snooze.
1: Halt the oscillation circuit when the xSLEEP pin is activated during snooze.

The device is restored from the sleep state when an interrupt cause that is effective even during sleep arises.
In this case, the S1R72003F00B100 operates in the manner described below.
1) An interrupt cause arises (interrupt cause effective even during sleep: VBUSChanged, Non_J, Port1, and Port0).
2) Restore the oscillation circuit.
3) Wait until the oscillation circuit stabilizes. (The wait time depends on the crystal oscillation circuit and should be evaluated on the board.)
4) Assert the xINT signal.
5) The CPU negates the xSLEEP pin.

The current consumption in the S1R72003F00B100 can be reduced to several mA by turning the oscillation circuit off during sleep.

**Bit 1** **AnalogPwdown**
Controls whether or not to enable the AnalogFrontEnd unit of the internal transceiver macro.
0: Disable the AnalogFrontEnd unit of the internal transceiver macro.
1: Enable the AnalogFrontEnd unit of the internal transceiver macro.

If no cables are connected (i.e., USBStatus register VBUS bit is 0) and the S1R72003F00B100 is placed in the snooze state by setting the PMControl register Snooze bit, the device power consumption can be further reduced by setting this bit to 1.
In this case, the current consumption in the S1R72003F00B100 is reduced to several mA.

**Bit 0** **Snooze**
Setting this bit to 1 halts the PLL oscillation in the S1R72003F00B100 (snooze mode). The snooze mode is used to reduce the current consumption when the suspend state is detected on the USB. To enter the snooze mode, set the USBControl register DisBusDetect bit to 1. Set the DisBusDetect bit to 1 before setting the Snooze bit to 1. Only specific registers can be accessed during snooze. For information on which registers are effective even during snooze, refer to the description of the registers. Because the USB reset state detection by the SIEIntStat register DetectReset bit does not work during snooze, check the USBStatus register LineState bit to determine whether the detection of a request for USB reset or suspend state deactivation (resume) is possible.

Set this bit to 0 to deactivate the suspend state. After confirming that the InSnooze bit is cleared to 0, perform the necessary processing using the USBControl register GoChirp bit for reset or the RestoreUSB bit for resume. Then set the DisBusDetect bit to 0 to allow detection of the USB reset or suspend state.

0: Deactivate the snooze state.
1: Activate the snooze state.

### 7.2.30 23h USB Control (USBControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 23h | USBControl | R/W | 7: DisBusDetect | 0: Enable BusDetect | 1: Disable BusDetect | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: SendWakeup | 0: Normal | 1:Send Remotewakeup Signal | |
| | | R/W | 2: RestoreUSB | 0: Normal | 1: Restore USB | |
| | | R/W | 1: GoChirp | 0: Normal | 1: Go Chirp | |
| | | R/W | 0: ActiveUSB | 0: In Active USB | 1: Active USB | |

This register sets the USB-related operations of the S1R72003F00B100.



**Bit 7** **DisBusDetect**

Setting this bit to 1 nullifies USB reset/suspend detection.
0: Automatically detect the USB reset/suspend state.
1: Do not automatically detect the USB reset/suspend state.

When this bit is cleared to 0, the device monitors activity on the USB bus.
If no bus activity is detected for 3 ms or longer in the HS mode, the device automatically switches mode to FS, identifies the state as a reset or suspend state, and sets the relevant interrupt cause (DetectReset or DetectSuspend).
If no bus activity is detected for 3 ms or longer in the FS mode, the state is identified as a suspend state. If SE0 lasting 2.5 µs or longer is detected, the state is identified as a reset state.

Once a reset or suspend state is detected (immediately after the DetectReset or DetectSuspend interrupt cause bit is set to 1), set the DisBusDetect to 1 to disable the USB reset/suspend state detection.

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **SendWakeup**

Setting this bit to 1 causes the RemoteWakeup signal (FS-K/HS-K) to be output to the USB port.
0: Perform no operation.
1: Send the RemoteWakeup signal.

When 1 ms or longer (Max. 15 ms) has elapsed after sending the RemoteWakeup signal, clear this bit to 0 to stop sending the signal. Note that the device must be restored from the snooze state before this bit can be set to 1.
(This operation can only be performed when the PMControl register InSnooze bit is 0.)

**Bit 2** **RestoreUSB**

If this bit is set to 1 when the USB is resumed from the suspend state, it returns to the previous operation mode (FS or HS) saved before it was suspended, and the relevant interrupt cause (RestoreCmp) is set.
This bit is automatically cleared to 0 when the operation is finished. Note that the device must be restored from the snooze state before this bit can be set to 1. (This operation can only be performed when the PMControl register InSnooze bit is 0.)

0: Perform no operation.
1: Restore the USB operation mode where it was placed before the suspend state.

**Bit 1** **GoChirp**

If this bit is set to 1 while the USB bus is in the reset state, HS Detection Handshake between the host and hub is performed, setting the XcvrControl register TermSelect and XcvrSelect bits, and the USBStatus register FSxHS bit automatically. The interrupt cause (ChirpCmp) is set upon completion of the above operation.
This bit is automatically cleared to 0 after the operation is finished. The result of negotiation can be confirmed by inspecting the USBStatus register FSxHS bit after the end of operation.

0: Perform no operation.
1: Start HS Detection Handshake operation.

**Bit 0** **ActiveUSB**

When the S1R72003F00B100 is reset in hardware, this bit is cleared to 0, with all USB functions turned off. The USB can be enabled by setting this bit to 1 after setting up the S1R72003F00B100.

0: Do not enable USB functions/operation.
1: Enable USB functions/operation.

### 7.2.31 24h USB Status (USBStatus)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 24h | USBStatus | R | 7: VBUS | 0: VBUS = L | 1: VBUS = H | XXh |
| | | R/(W) | 6: FSxHS | 0: HS | 1: FS | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R | 1: LineState[1] | Line State | | |
| | | | 0: LineState[0] | | | |

This register shows the USB related status.
This register is effective even during snooze.



**Bit 7** **VBUS**
Reflects the VBUS pin status directly as is.
0: Not connected.
1: Connected.

**Bit 6** **FSxHS**
Indicates the current USB operation mode. By setting this bit, the operation mode can be changed forcibly. Usually, the user do not have to set this bit since it is automatically set after HS Detection Handshake (see Appendix A.3.).
0: HS mode
1: FS mode

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bits 1-0** **LineState [1:0]**
Indicates the USB cable signal status.
When the XcvrControl register TermSelect bit = 1 (FS termination selected), if the XcvrSelect bit is 1 (FS transceiver selected), LineState indicates the received value of the DP/DM FS single-end receiver; if XcvrSelect is 0 (HS transceiver selected), it indicates the received value of the HS differential receiver. When TermSelect = 0, LineState indicates 0b11.

| LineState | | |
|---|---|---|
| **TermSelect** | **DP / DM** | **LineState[1:0]** |
| 0 | Don't Care | 0b11 |
| 1 | SE0 | 0b00 |
| 1 | J | 0b01 |
| 1 | K | 0b10 |
| 1 | SE1 | 0b11 |

### 7.2.32 25h Xcvr Control (XcvrControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 25h | XcvrControl | R/W | 7: TermSelect | 0: HS | 1: FS | 41h |
| | | R/W | 6: XcvrSelect | 0: HS | 1: FS | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: OpMode[1] | OpMode | | |
| | | | 0: OpMode[0] | | | |

This register sets the parameters associated with the transceiver macro.



**Bit 7** **TermSelect**
Sets either FS or HS termination as the valid termination. This bit is automatically set when HS Detection Handshake is performed by the USBControl register GoChirp bit.

0: HS
1: FS

**Bit 6** **XcvrSelect**
Sets either FS or HS transceiver as the valid transceiver. This bit is automatically set when HS Detection Handshake is performed by the USBControl register GoChirp bit.

0: HS
1: FS

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bits 1-0** **OpMode**
Sets the operation mode of the transceiver macro.

| OpMode | |
|---|---|
| 00 | “Normal Operation” |
| 01 | “Non-Driving” |
| 10 | “Disable Bitstuffing and NRZI encoding” |
| 11 | “Reserved” |

### 7.2.33 26h USB Test (USBTest)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 26h | USBTest | R/W | 7: EnHSTest | 0: Normal | 1: Enable HS Test mode | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: SE0_NAK | 0: Normal | 1: SE0 NAK | |
| | | R/W | 2: TEST_J | 0: Normal | 1: TEST J | |
| | | R/W | 1: TEST_K | 0: Normal | 1: TEST K | |
| | | R/W | 0: TestPacket | 0: Normal | 1: Test packet | |

This register sets the parameters associated with USB 2.0 test mode. When the operation mode is "HS," set the bit corresponding to the test mode specified by a SetFeature request, and then set the EnHSTest bit to 1 after the status stage is finished. The device can be operated in the test mode defined under the USB 2.0 specification.



**Bit 7** **EnHSTest**
Setting this bit to 1 while one of the 4 low-order bits of the USBTest register is set to 1 places the device in the test mode corresponding to that bit. To run a device in the test mode, set the USBControl register DisBusDetect bit to 1 to disable USB reset/suspend state detection.

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **SE0_NAK**
Setting this bit to 1, then setting the EnHSTest bit to 1 places the device in SE0_NAK test mode.

**Bit 2** **TEST_J**
Setting this bit to 1, and then setting the EnHSTest bit to 1 places the device in HS J test mode. Note that OpMode must be set to 10 (Disable Bitstuffing and NRZI encoding) before the EnHSTest bit is set to 1.

**Bit 1** **TEST_K**
Setting this bit to 1, and then setting the EnHSTest bit to 1 places the device in HS K test mode. Note that OpMode must be set to 10 (Disable Bitstuffing and NRZI encoding) before the EnHSTest bit is set to 1.

**Bit 0** **TestPacket**
Setting this bit to 1, and then setting the EnHSTest bit to 1 places the device in packet transmit test mode. Because EPc is used when operating in this test mode, several settings are required. The procedure is described below.

1) To enable EPc, set MaxPacketSize of EPc to 64 or more, the transfer direction to IN, and EndPointNumber of EPc to 15.
2) Clear the EPaConfig_1 and EPbConfig_1 register EnEndPoint bits to 0. Then set the EPcConfig_1 register EnEndPoint bit to 1.
3) Clear the FIFO of EPc and write the test packet data in this FIFO.
4) Clear the EPcIntEnb register EnINTranERR bit to 0.

Shown below are the data to be written to the FIFO during packet transmit test mode:
00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h,
00h, AAh, AAh, AAh, AAh, AAh, AAh, AAh,
AAh, EEh, EEh, EEh, EEh, EEh, EEh, EEh,
EEh, FEh, FFh, FFh, FFh, FFh, FFh, FFh,
FFh, FFh, FFh, FFh, FFh, 7Fh, BFh, DFh,
EFh, F7h, FBh, FDh, FCh, 7Eh, BFh, DFh,
EFh, F7h, FBh, FDh, 7Eh

When sending a test packet, the SIE adds PID and CRC to the transmit data. A range of test packet data specified in the USB specification must be written to the FIFO, from the data immediately following DATA0 PID to the data preceding CRC16.

### 7.2.34 27h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 27h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.35 28h USB Address (USBAddress)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 28h | USBAddress | | 7: | 0: | 1: | 00h |
| | | R/W | 6: USBAddress[6] | USB Address | | |
| | | | 5: USBAddress[5] | | | |
| | | | 4: USBAddress[4] | | | |
| | | | 3: USBAddress[3] | | | |
| | | | 2: USBAddress[2] | | | |
| | | | 1: USBAddress[1] | | | |
| | | | 0: USBAddress[0] | | | |



This register sets the USB address.

**Bit 7** **Reserved**

**Bits 6-0** **USBAddress**
Sets the USB address. The address to be set in these bits is specified by the host after the status stage of a SetAddress request is finished.

### 7.2.36 29h EPr Control (EPrControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 29h | EPrControl | R | 7: DMARunning | 0: DMA Not Running | 1: DMA Running | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | W | 4: ALLFIFOClr | 0: Normal | 1: FIFO Clear | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: AutoEnShort | 0: Disable AutoEnShort | 1: Enable AutoEnShort | |
| | | R/W | 1: ALLForceNAK | 0: Normal | 1: ALLForceNAK | |
| | | R/W | 0: EPrForceSTALL | 0: Normal | 1: EPrForceSTALL | |

This register sets and indicates the general endpoint operation.



**Bit 7** **DMARunning**
This bit is set to 1 while DMA transfer is underway. For IN transactions, this bit is cleared to 0 when the DMA transfer is finished, with all packets remaining in the FIFO thereby transmitted. For OUT transactions, this bit is cleared to 0 when the DMA transfer is finished after all packets are received and the DTGO bit is set to 0.

0: DMA not operating
1: DMA operating

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **ALLFIFOClr**
Setting this bit to 1 clears the FIFOs for all endpoints. When the MaxPacketSize or DoubleBuf bits have been set for each endpoint, always confirm that this bit is set to 1 to clear the FIFOs for all endpoints after this setting is made. This bit is automatically cleared to 0 after the FIFO is cleared.

**Bit 3** **Reserved**

**Bit 2** **AutoEnShort**
Sets the operation mode of short packet transfer during IN transactions at the endpoint for which the EP[a,b,c]Config_1 register JoinIDE bit = 1.
0: When the data remaining in the FIFO after the end of DMA is smaller than MaxPacketSize, the data in the FIFO is not transferred until the EnShortPkt bit of the relevant endpoint is set to 1. Setting the EnShortPkt bit to 1 using firmware causes data transfer in response to an IN token from the host. Upon successful completion of the IN transaction, the BulkIntStat register BulkInCmp bit is set to 1.
1: When the data remaining in the FIFO after the end of DMA is smaller than MaxPacketSize, the EnShortPkt bit of the relevant endpoint is automatically set to 1. The data in the FIFO is transferred in response to an IN token from the host. Upon completion of the IN transaction, the BulkIntStat register BulkInCmp bit is set to 1.
Check the transfer data size before starting DMA transfer. Set this bit when a short packet occurs.

**Bit 1** **ALLForceNAK**
Setting this bit to 1 allows the EP0Control_0 register InForceNAK and OutForceNAK bits, as well as the ForceNAK bits in all EP[a,b,c]_Control1_0 registers, to be set to 1.

**Bit 0** **EPrForceSTALL**
Setting this bit to 1 allows the ForceSTALL bits in all EP[a,b,c]_Control1_0 registers to be set to 1.

### 7.2.37 2Ah BulkOnly Control (BulkOnlyControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 2Ah | BulkOnlyControl | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: GoCBWMode | 0: Normal mode | 1: CBW receive mode | |
| | | R/W | 1: GoCSWMode | 0: Normal mode | 1: CSW transfer mode | |
| | | R/W | 0: CSWSel | 0: Select CSW0_00 to CSW0_12 | 1: Select CSW1_00 to CSW1_12 | |



This register sets the operations of the USB storage-class BulkOnly transport protocol.

**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **GoCBWMode**

Setting this bit to 1 places the device in the BulkOnly transport protocol's CBW receive mode. When 31 bytes of CBW are received normally from the host, the BulkIntStat register's CBWCmp interrupt is generated. The received data is stored in registers CBW_00 through CBW_30. If the received data consists of less than 31 bytes, a CBWShort interrupt is generated. Conversely, if the received data is larger than 31 bytes, a CBWLong interrupt is generated. If a STALL is returned at the corresponding end point when CBW is received, a CBWErr interrupt is generated. When a CBWCmp, CBWShort, or CBWLong interrupt is generated, the GoCBWMode bit is automatically cleared. However, when a CBWErr interrupt is generated, the GoCBWMode bit is not cleared. If data is received during CBW receive mode, the PingTranACK, OUTShortACK, INTranACK, OUTTranACK, INTranNAK, OUTTranNAK, INTranErr, and OUTTranErr bits in the EP[a,b,c]IntStat register that corresponds to the CBW endpoint remain unchanged (not set to 1) when CBW is received; only the CBWCmp, CBWShort, CBWLong, and CBWErr bits change state. MaxPacketSize that specifies the endpoint required for receiving CBW must be set to 32, 64 or 512. The CBW receive function can be used for another purpose if the Bulk OUT data size is 31 bytes.

**Bit 1** **GoCSWMode**

Setting this bit to 1 places the device in the BulkOnly transport protocol's CSW transmit mode. Either registers CSW0_00 through CSW0_12 or registers CSW1_00 through CSW1_12 can be selected for transmit data using the CSWSel bit. When an ACK from the host is received after CSW is transmitted, a CSWCmp interrupt configured in the BulkIntStat register is generated. When ACK is not received, a CSWErr interrupt is generated. After a CSWCmp or CSWErr interrupt is generated, the GoCSWMode bit is cleared to 0 and the GoCBWMode bit is set to 1 and the device is automatically placed in CBW receive mode. During CSW transmit mode, the PingTranACK, OUTShortACK, INTranACK, OUTTranACK, INTranNAK, OUTTranNAK, INTranErr, and OUTTranErr bits in the EP[a,b,c]IntStat register that corresponds to the CSW endpoint remain unchanged (not set to 1); only the CSWCmp and CSWErr bits change state. MaxPacketSize that specifies the endpoint required for transmitting CSW must be set to 16, 32, 64 or 512. The CSW transmit function can be used for another purpose if the Bulk IN data size is 13 bytes.

**Bit 0** **CSWSel**

In the BulkOnly transport protocol's CSW transmit mode, this bit sets the contents of the CSW0_00 to CSW0_12 registers or the contents of the CSW1_00 to CSW1_12 registers to be transmitted.

0: Transmit the contents of the CSW0_00 to CSW0_12 registers.

1: Transmit the contents of the CSW1_00 to CSW1_12 registers.

### 7.2.38 2Bh BulkOnly Config (BulkOnlyConfig)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 2Bh | BulkOnlyConfig | R/W | 7: CBWEPNumber[3] | CBW Endpoint Number | 00h |
| | | | 6: CBWEPNumber[2] | | |
| | | | 5: CBWEPNumber[1] | | |
| | | | 4: CBWEPNumber[0] | | |
| | | R/W | 3: CSWEPNumber[3] | CSW Endpoint Number | |
| | | | 2: CSWEPNumber[2] | | |
| | | | 1: CSWEPNumber[1] | | |
| | | | 0: CSWEPNumber[0] | | |

This register sets the endpoint numbers used in the USB storage-class BulkOnly transport protocol.



**Bits 7-4 CBWEPNumber**
These bits set the endpoint number of the Bulk OUT endpoint necessary to receive the CBW used in the BulkOnly transport protocol.
Any value in the range of 01 to 15 can be set in this register.

**Bits 3-0 CSWEPNumber**
These bits set the endpoint number of the Bulk IN endpoint necessary to transmit the CSW used in the BulkOnly transport protocol.
Any value in the range of 01 to 15 can be set in this register.

### 7.2.39 2Ch to 2Eh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 2Ch to 2Eh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.40 2Fh Chip Config (ChipConfig)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 2Fh | ChipConfig | R/W | 7: RDYxWAIT | 0: xWAIT mode | 1: READY mode | 00h |
| | | R/W | 6: WaitMode | 0: Hiz – 0 mode | 1: 1-0 mode | |
| | | R/W | 5: IntMode | 0: Hiz – 0 mode | 1: 1-0 mode | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

This register sets the operation mode of the xWAIT pin and the output of the xWait and xInt pins.
This register is effective even during snooze.



**Bit 7 RDYxWAIT**
0: Serve as a wait signal for the CPU. This signal is asserted (LOW) when requesting the CPU to wait. In the circuitry, xWAIT is the logical OR of the internal READY and xCS signals.
1: Serve as a ready signal for the CPU. When the S1R72003F00B100 is ready for read or write, this signal is asserted (HIGH). In the circuitry, this is the internal READY signal output directly from the xWAIT pin.

**Bit 6 WaitMode**
0: xWAIT output is 0 or Hi-Z.
1: xWAIT output is 0 or 1.

**Bit 5 IntMode**
0: xINT output is 0 or Hi-Z.
1: xINT output is 0 or 1.

**Bit 4 Reserved**

**Bit 3 Reserved**

**Bit 2 Reserved**

**Bit 1 Reserved**

**Bit 0 Reserved**

### 7.2.41 30h to 37h EP0 Setup0 to EP0 Setup7 (EP0Setup_0 to EP0Setup_7)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 30h<br>to<br>37h | EP0Setup_0<br>to<br>EP0Setup_7 | R | 7: EP0Setup_n[7] | Endpoint 0 Setup Data 0 to Endpoint 0 Setup Data 7 | 00h |
| | | | 6: EP0Setup_n[6] | | |
| | | | 5: EP0Setup_n[5] | | |
| | | | 4: EP0Setup_n[4] | | |
| | | | 3: EP0Setup_n[3] | | |
| | | | 2: EP0Setup_n[2] | | |
| | | | 1: EP0Setup_n[1] | | |
| | | | 0: EP0Setup_n[0] | | |

These registers are used to store data received in the Endpoint 0 setup stage.



**EP0Setup_0**
BmRequestType is set in this register.

**EP0Setup_1**
BRequest is set in this register.

**EP0Setup_2**
The 8 low-order bits of Wvalue are set in this register.

**EP0Setup_3**
The 8 high-order bits of Wvalue are set in this register.

**EP0Setup_4**
The 8 low-order bits of WIndex are set in this register.

**EP0Setup_5**
The 8 high-order bits of WIndex are set in this register.

**EP0Setup_6**
The 8 low-order bits of WLength are set in this register.

**EP0Setup_7**
The 8 high-order bits of WLength are set in this register.

### 7.2.42 38h FrameNumber High (FrameNumber_H)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 38h | FrameNumber_H | R | 7: FnInvalid | 0: Frame number valid | 1: Frame number invalid | 80h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | R | 2: FrameNumber[10] | Frame Number High | | |
| | | | 1: FrameNumber[9] | | | |
| | | | 0: FrameNumber[8] | | | |

This register shows USB frame numbers. To acquire a frame number, access the FrameNumber_H and FrameNumber_L registers in pairs. Be sure to access the FrameNumber_H register first.

**Bit 7 FnInvalid**
This bit is set to 1 when an error occurs in the SOF packet received.
0: SOF packet received normally.
1: Error occurred when receiving a SOF packet.

**Bit 6 Reserved**

**Bit 5 Reserved**

**Bit 4 Reserved**

**Bit 3 Reserved**

**Bits 2-0 FrameNumber**
These bits store the 3 high-order bits of data that represent the FrameNumber field of the received SOF packet.

### 7.2.43 39h FrameNumber Low (FrameNumber_L)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 39h | FrameNumber_L | R | 7: FrameNumber[7] | Frame Number Low | 00h |
| | | | 6: FrameNumber[6] | | |
| | | | 5: FrameNumber[5] | | |
| | | | 4: FrameNumber[4] | | |
| | | | 3: FrameNumber[3] | | |
| | | | 2: FrameNumber[2] | | |
| | | | 1: FrameNumber[1] | | |
| | | | 0: FrameNumber[0] | | |

This register acquires USB frame numbers. To acquire a frame number, access the FrameNumber_H and FrameNumber_L registers in pairs. Be sure to access the FrameNumber_H register first.



**Bits 7-0 FrameNumber**
These bits store the 8 low-order bits of data that represent the FrameNumber field of the received SOF packet.

### 7.2.44 3Ah to 3Fh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 3Ah to 3Fh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.45 40h EP0 Config_0 (EP0Control_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 40h | EP0Config_0 | R/W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

This register sets Endpoint 0.



**Bit 7** **INxOUT**
Sets the transfer direction of Endpoint 0.
0: OUT direction
1: IN direction

When setting a value in this bit, determine the direction from the request received in the setup stage. If there is a data stage, use this bit to set the transfer direction in the data stage. Then clear the EP0Control_0 register's INForceNAK or OUTForceNAK bit to 0 (whichever corresponds to the data stage transfer direction) to execute the data stage.

When the data stage transfer direction is IN after the completion of the data stage, the direction of the status stage is OUT. Then the status stage can be executed by setting this bit to 0 and clearing the EP0Control_0 register OUTForceNAK bit to 0.
When the data stage transfer direction is OUT or no data stages exist, the direction of the status stage is IN. Then the status stage can be executed by clearing the FIFO of endpoint 0 to set this bit to 1 and clearing the EP0Control_0 register INForceNAK bit to 0 to set the InEnShortPkt bit to 1.

For IN or OUT transactions in a direction different from that set in this bit, if the EP0Control_0 register corresponding to that transaction direction has its InForceSTALL or OutForceSTALL bits set, the transaction is responded by STALL. Otherwise, the transaction is responded by NAK.

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **Reserved**

**Bit 0** **Reserved**

### 7.2.46 41h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 41h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.47 42h EP0 Control_0 (EP0Control_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 42h | EP0Control_0 | R/W | 7: AutoForceNAK | 0: Normal | 1: Auto ForceNAK | 00h |
| | | R/W | 6: InEnShortPkt | 0: Normal | 1: Send Shot Packet | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: InForceNAK | 0: Normal | 1: In ForceNAK | |
| | | R/W | 2: InForceSTALL | 0: Normal | 1: In ForceSTALL | |
| | | R/W | 1: OutForceNAK | 0: Normal | 1: Out ForceNAK | |
| | | R/W | 0: OutForceSTALL | 0: Normal | 1: Out ForceSTALL | |

This register sets the operations associated with Endpoint 0 transfer.



**Bit 7** **AutoForceNAK**
This bit automatically sets the EP0Control_0 register's InForceNAK or OutForceNAK bit to 1 when the transaction completes normally.

**Bit 6** **InEnShortPkt**
Setting this bit to 1 allows the data in the current FIFO to be transmitted as a short packet for an IN transaction. This bit is automatically cleared to 0 when the packet transfer is finished. If this bit is set to 1 when no data exists in the FIFO, a packet of zero length is transmitted in response to an IN token from the host.

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **InForceNAK**
Setting this bit to 1 causes IN transaction to be responded by NAK, regardless of the data count in the FIFO. When the MainIntStat register RcvEP0Setup bit is set to 1 upon completion of the setup stage, the InForceNAK bit is automatically set to 1. This bit cannot be cleared to 0 while the MainIntStat register RcvEP0Setup bit is 1. If the data stage transfer direction is IN, the data stage can be executed by setting the EP0Config_0 register INxOUT bit for IN direction, and then clearing this bit to 0. If the data stage transfer direction is OUT, the status stage can be executed by clearing this bit to 0 after the status stage is ready to run. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect from the following transaction.

**Bit 2** **InForceSTALL**
Setting this bit to 1 causes IN transaction to be responded by STALL. This bit is given priority over the InForceNAK bit. When the MainIntStat register RcvEP0Setup bit is set to 1 upon completion of the setup stage, the InForceSTALL bit is set to 0. This bit cannot be set to 1 while the MainIntStat register RcvEP0Setup bit is 1. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect from the following transaction.

**Bit 1** **OutForceNAK**
Setting this bit to 1 makes a NAK response to the OUT transaction regardless of the free space in the FIFO. When the MainIntStat register RcvEP0Setup bit is set to 1 upon completion of the setup stage, the OUTForceNAK bit is automatically set to 1. This bit cannot be cleared to 0 while the MainIntStat register RcvEP0Setup bit is 1. If the data stage transfer direction is OUT, the data stage can be executed by setting the EP0Config_0 register INxOUT bit for OUT direction, and then clearing this bit to 0. If the data stage transfer direction is IN, the status stage can be executed by clearing this bit to 0 after the status stage is ready to run. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect from the following transaction.

**Bit 0** **OutForceSTALL**
Setting this bit to 1 causes OUT transaction to be responded by STALL. This bit is given priority over the OutForceNAK bit. When MainIntStat register RcvEP0Setup bit is set to 1 upon completion of the setup stage, the OutForceSTALL bit is set to 0. This bit cannot be set to 1 while the MainIntStat register RcvEP0Setup bit is 1. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect from the following transaction.

### 7.2.48 43h EP0 Control_1 (EP0Control_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 43h | EP0Control_1 | R | 7: InToggleStat | In Toggle Status | | 00h |
| | | | 6: | 0: | 1: | |
| | | R/W | 5: InToggleSet | 0: Normal | 1: In Transaction Toggle Set | |
| | | R/W | 4: InToggleClr | 0: Normal | 1:InTransaction Toggle Clear | |
| | | R | 3: OutToggleStat | Out Toggle Status | | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: OutToggleSet | 0: Normal | 1:OutTransaction Toggle Set | |
| | | R/W | 0: OutToggleClr | 0: Normal | 1:OutTransactionToggleClear | |

This register shows or sets the operations associated with Endpoint 0 toggle bits.



**Bit 7** **InToggleStat**
Shows the status of the IN transaction toggle sequence bit.

**Bit 6** **Reserved**

**Bit 5** **InToggleSet**
Sets the IN transaction toggle sequence bit to 1.

**Bit 4** **InToggleClr**
Clears the IN transaction toggle sequence bit to 0.

**Bit 3** **OutToggleStat**
Shows the status of the OUT transaction toggle sequence bit.

**Bit 2** **Reserved**

**Bit 1** **OutToggleSet**
Sets the OUT transaction toggle sequence bit to 1.

**Bit 0** **OutToggleClr**
Clears the OUT transaction toggle sequence bit to 0.

### 7.2.49 44h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 44h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.50 45h EP0 FIFO Remain (EP0FIFORemain)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 45h | EP0FIFORemain | | 7: | 0: | 1: | 00h |
| | | R | 6: EP0FIFORemainCounter[6] | Endpoint 0 FIFO Remain Counter | | |
| | | | 5: EP0FIFORemainCounter[5] | | | |
| | | | 4: EP0FIFORemainCounter[4] | | | |
| | | | 3: EP0FIFORemainCounter[3] | | | |
| | | | 2: EP0FIFORemainCounter[2] | | | |
| | | | 1: EP0FIFORemainCounter[1] | | | |
| | | | 0: EP0FIFORemainCounter[0] | | | |

This register shows the number of data bytes in the Endpoint 0 FIFO.

**Bit 7** **Reserved**

**Bit 6-0** **EP0FIFORemainCounter**
These bits show the number of data bytes remaining in the Endpoint 0 FIFO. When accessing the FIFO from the CPU, inspect this register to check the data counts.

### 7.2.51 46h EP0 FIFOforCPU (EP0FIFOforCPU)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 46h | EP0FIFOforCPU | R/W | 7: EP0FIFOData[7] | Endpoint 0 FIFO Access from CPU | XXh |
| | | | 6: EP0FIFOData[6] | | |
| | | | 5: EP0FIFOData[5] | | |
| | | | 4: EP0FIFOData[4] | | |
| | | | 3: EP0FIFOData[3] | | |
| | | | 2: EP0FIFOData[2] | | |
| | | | 1: EP0FIFOData[1] | | |
| | | | 0: EP0FIFOData[0] | | |

This register is used for FIFO access from the CPU.
When the EP0FIFOControl register EnFIFOwr bit is set to 1, data can be written into the FIFO by writing a value in this register.
When the EP0FIFOControl register EnFIFOrd bit is set to 1, data can be read from the FIFO by reading the value from this register.
If a value is written in this register without setting the EnFIFOwr bit, writing in the FIFO is not executed. If a value is read from the register without setting the EnFIFOrd bit, dummy data is output.



### 7.2.52 47h EP0 FIFO Control (EP0FIFOControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 47h | EP0FIFOControl | R | 7: FIFOEmpty | 0: FIFO Not Empty | 1: FIFO Empty | 80h |
| | | R | 6: FIFOFull | 0: FIFO Not Full | 1: FIFO Full | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | W | 2: FIFOClr | 0: Normal | 1: FIFO Clear | |
| | | R/W | 1: EnFIFOwr | 0: Normal | 1: Enable FIFO write | |
| | | R/W | 0: EnFIFOrd | 0: Normal | 1: Enable FIFO read | |

This register acquires or sets the Endpoint 0 FIFO status.

**Bit 7** **FIFOEmpty**
When this bit = 1, the FIFO is empty. Before reading data from the FIFO, check to see that this bit = 0.

**Bit 6** **FIFOFull**
When this bit = 1, the FIFO is full. Before writing data into the FIFO, check to see that this bit = 0.

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **FIFOClr**
Setting this bit to 1 clears the FIFO. This bit is automatically cleared to 0 after the FIFO is cleared.

**Bit 1** **EnFIFOwr**
Setting this bit to 1 allows data to be written into the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOrd bit is 1.

**Bit 0** **EnFIFOrd**
Setting this bit to 1 allows data to be read from the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOwr bit is 1.

### 7.2.53 48h to 4Fh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 48h to 4Fh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.54 50h EPa Config_0 (EPaConfig_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 50h | EPaConfig_0 | R/W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: EndPointNumber[3] | Endpoint Number | | |
| | | | 2: EndPointNumber[2] | | | |
| | | | 1: EndPointNumber[1] | | | |
| | | | 0: EndPointNumber[0] | | | |

This register sets Endpoint a.



**Bit 7** **INxOUT**
Sets the transfer direction of the endpoint.
0: OUT direction
1: IN direction

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bits 3-0** **EndPointNumber**
These bits set an endpoint number in the range from 01 to 15.

### 7.2.55 51h EPa Config_1 (EPaConfig_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 51h | EPaConfig_1 | R/W | 7: JoinIDE | 0: Not Join IDE | 1: Join IDE | 00h |
| | | R/W | 6: ToggleMode | 0: | 1: | |
| | | R/W | 5: EnEndPoint | 0: Disable Endpoint | 1: Enable Endpoint | |
| | | R/W | 4: DoubleBuf | 0: Single Buffer | 1: Double Buffer | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: MaxPacketSize[2] | Max Packet Size | | |
| | | | 1: MaxPacketSize[1] | | | |
| | | | 0: MaxPacketSize[0] | | | |

This register sets Endpoint a.



**Bit 7** **JoinIDE**
Connects the endpoint to DMA.
DMA connects to the last endpoint that has had this bit set to 1. Immediately following the reset, the JoinIDE bit is 0 for all endpoints.

0: Do not connect this endpoint to DMA.
1: Connect this endpoint to DMA.

**Bit 6** **ToggleMode**
Sets the operation mode of the toggle bit (IN transaction only).

0: Toggle only when the transaction terminates normally.
1: Always toggle for each transaction performed.

**Bit 5** **EnEndPoint**
Setting this bit to 1 enables the endpoint. Accesses to the endpoint are ignored when this bit = 0. Set the appropriate value in this bit following the SetConfiguration request from the host.

0: Disables the endpoint.
1: Enables the endpoint.

**Bit 4** **DoubleBuf**
Setting this bit to 1 configures the FIFO for the endpoint as double buffers. A memory space twice the size set by MaxPacketSize is reserved in the FIFO.

0: Configure the FIFO as a single buffer.
1: Configure the FIFO as double buffers.

**Bit 3** **Reserved**

**Bits 2-0** **MaxPacketSize**
Sets the maximum value of the packet size. The relationship between the set values and the packet sizes is shown below. (bit 2, 1, 0)

| MaxPacketSize | | |
|---|---|---|
| | FS | HS |
| 000 | Reserved | Reserved |
| 001 | 8 bytes | 8 bytes |
| 010 | 16 bytes | 16 bytes |
| 011 | 32 bytes | 32 bytes |
| 100 | 64 bytes | 64 bytes |
| 101 | | 512 bytes |
| 110 | | |
| 111 | Reserved | 1024 bytes |

The settings other than 512 bytes in the HS mode are used in interrupt transfers. Use EPc when isochronous transfer is desired.
After setting MaxPacketSize and DoubleBuf for the endpoints, be sure to set the EPrControl register ALLFIFOClr bit to 1 to clear all FIFOs.
In addition, ensure that the total FIFO area reserved by endpoints a, b, c does not exceed 2432 bytes.

### 7.2.56 52h EPa Control_0 (EPaControl_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 52h | EPaControl_0 | R/W | 7: AutoForceNAK | 0: Normal | 1: AutoForceNAK | 00h |
| | | W | 6: EnShortPkt | 0: Normal | 1: Send Short Packet | |
| | | R/W | 5: AutoForceNAKShort | 0: Normal | 1: AutoForceNAKShort | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: ForceNAK | 0: Normal | 1: ForceNAK | |
| | | R/W | 0: ForceSTALL | 0: Normal | 1: ForceSTALL | |

This register sets the operation of endpoint a.



**Bit 7** **AutoForceNAK**
This bit automatically sets the EPaControl_0 register's ForceNAK bit to 1 when the transaction completes normally.

0: Do not automatically set the ForceNAK bit.
1: Automatically set the ForceNAK bit.

**Bit 6** **EnShortPkt(IN transaction only)**
Setting this bit to 1 allows the data in the current FIFO to be transmitted as a short packet for an IN transaction. This bit is automatically cleared to 0 when the packet transfer is finished. If this bit is set to 1 when no data exists in the FIFO, a packet of zero length is transmitted in response to an IN token from the host.

**Bit 5** **AutoForceNAKShort**
When this bit = 1, if the packet received during an OUT transaction that completed normally is a short packet, the ForceNAK bit is automatically set to 1. If the AutoForceNAK bit = 1, AutoForceNAK has priority over this bit.

0: Do not automatically set the ForceNAK bit to 1.
1: Automatically set the ForceNAK bit to 1.

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **ForceNAK**
Setting this bit to 1 makes a NAK response to the transaction regardless of the data count and free space in the FIFO. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect in the subsequent transactions.

**Bit 0** **ForceSTALL**
Setting this bit to 1 causes the transaction to be responded by STALL. This bit is given priority over the ForceNAK bit. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect from the following transaction.

### 7.2.57 53h EPa Control_1 (EPaControl_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 53h | EPaControl_1 | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R | 3: ToggleStat | Toggle Status | | |
| | | | 2: | 0: | 1: | |
| | | W | 1: ToggleSet | 0: Normal | 1: Toggle Set | |
| | | W | 0: ToggleClr | 0: Normal | 1: Toggle Clear | |

This register indicates and controls the status of the Endpoint a toggle bit.



**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **ToggleStat**
Shows the status of the toggle sequence bit.

**Bit 2** **Reserved**

**Bit 1** **ToggleSet**
Setting this bit to 1 sets the toggle sequence bit to 1.

**Bit 0** **ToggleClr**
Setting this bit to 1 clears the toggle sequence bit to 0.

### 7.2.58 54h EPa FIFO Remain High (EPaFIFORemain_H)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 54h | EPaFIFORemain_H | R | 7: EPaFIFORemainCounter[15] | Endpoint a FIFO Remain Counter High | 00h |
| | | | 6: EPaFIFORemainCounter[14] | | |
| | | | 5: EPaFIFORemainCounter[13] | | |
| | | | 4: EPaFIFORemainCounter[12] | | |
| | | | 3: EPaFIFORemainCounter[11] | | |
| | | | 2: EPaFIFORemainCounter[10] | | |
| | | | 1: EPaFIFORemainCounter[9] | | |
| | | | 0: EPaFIFORemainCounter[8] | | |

This register shows the eight high-order bits that represent the remaining data counts in the Endpoint a FIFO. To acquire the FIFO's remaining data counts, access the EPaFIFORemain_H and EPaFIFORemain_L registers in pairs. Be sure to access the EPaFIFORemain_H register first.

### 7.2.59 55h EPa FIFO Remain Low (EPaFIFORemain_L)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 55h | EPaFIFORemain_L | R | 7: EPaFIFORemainCounter[7] | Endpoint a FIFO Remain Counter Low | 00h |
| | | | 6: EPaFIFORemainCounter[6] | | |
| | | | 5: EPaFIFORemainCounter[5] | | |
| | | | 4: EPaFIFORemainCounter[4] | | |
| | | | 3: EPaFIFORemainCounter[3] | | |
| | | | 2: EPaFIFORemainCounter[2] | | |
| | | | 1: EPaFIFORemainCounter[1] | | |
| | | | 0: EPaFIFORemainCounter[0] | | |

This register shows the eight low-order bits that represent the remaining data counts in the Endpoint a FIFO. To acquire the FIFO's remaining data counts, access the EPaFIFORemain_H and EPaFIFORemain_L registers in pairs. Be sure to access the EPaFIFORemain_H register first.

### 7.2.60 56h EPa FIFO for CPU (EPaFIFOforCPU)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 56h | EPaFIFOforCPU | R/W | 7: EPaFIFOData[7] | Endpoint a FIFO Access from CPU | XXh |
| | | | 6: EPaFIFOData[6] | | |
| | | | 5: EPaFIFOData[5] | | |
| | | | 4: EPaFIFOData[4] | | |
| | | | 3: EPaFIFOData[3] | | |
| | | | 2: EPaFIFOData[2] | | |
| | | | 1: EPaFIFOData[1] | | |
| | | | 0: EPaFIFOData[0] | | |

This register is used for FIFO access from the CPU.
When the EPaFIFOControl register EnFIFOwr bit is set to 1, data can be written into the FIFO by writing a value in this register.
When the EPaFIFOControl register EnFIFOrd bit is set to 1, data can be read from the FIFO by reading the value from this register.
If a value is written in this register without setting the EnFIFOwr bit, writing in the FIFO is not executed. If a value is read from the register without setting the EnFIFOrd bit, dummy data is output.



### 7.2.61 57h EPa FIFO Control (EPaFIFOControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 57h | EPaFIFOControl | R | 7: FIFOEmpty | 0: FIFO Not Empty | 1: FIFO Empty | 80h |
| | | R | 6: FIFOFull | 0: FIFO Not Full | 1: FIFO Full | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | W | 2: FIFOClr | 0: Normal | 1: FIFO Clear | |
| | | R/W | 1: EnFIFOwr | 0: Normal | 1: Enable FIFO write | |
| | | R/W | 0: EnFIFOrd | 0: Normal | 1: Enable FIFO read | |

This register shows or controls the Endpoint a FIFO status.

**Bit 7** **FIFOEmpty**
When this bit = 1, the FIFO is empty.
Before reading data from the FIFO, check to see that this bit = 0.

**Bit 6** **FIFOFull**
When this bit = 1, the FIFO is full.
Before writing data into the FIFO, check to see that this bit = 0.

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **FIFOClr**
Setting this bit to 1 clears the FIFO.
This bit is automatically cleared to 0 after the FIFO is cleared.

**Bit 1** **EnFIFOwr**
Setting this bit to 1 allows data to be written into the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOrd bit is 1.

**Bit 0** **EnFIFOrd**
Setting this bit to 1 allows data to be read from the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOwr bit is 1.

### 7.2.62 58h EPb Config_0 (EPbConfig_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 58h | EPbConfig_0 | R/W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: EndPointNumber[3] | Endpoint Number | | |
| | | | 2: EndPointNumber[2] | | | |
| | | | 1: EndPointNumber[1] | | | |
| | | | 0: EndPointNumber[0] | | | |

This register sets Endpoint b.



**Bit 7** **INxOUT**
Sets the transfer direction of the endpoint.
0: OUT direction
1: IN direction

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bits 3-0** **EndPointNumber**
These bits set an endpoint number in the range from 01 to 15.

### 7.2.63 59h EPb Config_1 (EPbConfig_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 59h | EPbConfig_1 | R/W | 7: JoinIDE | 0: Not Join IDE | 1: Join IDE | 00h |
| | | R/W | 6: ToggleMode | 0: | 1: | |
| | | R/W | 5: EnEndPoint | 0: Disable Endpoint | 1: Enable Endpoint | |
| | | R/W | 4: DoubleBuf | 0: Single Buffer | 1: Double Buffer | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: MaxPacketSize[2] | Max Packet Size | | |
| | | | 1: MaxPacketSize[1] | | | |
| | | | 0: MaxPacketSize[0] | | | |

This register sets Endpoint b.



**Bit 7** **JoinIDE**
Connects the endpoint to DMA
DMA connects to the last endpoint that had this bit set to 1.
Immediately following the reset, the JoinIDE bit is 0 for all endpoints.

0: Do not connect this endpoint to DMA.
1: Connect this endpoint to DMA.

**Bit 6** **ToggleMode**
Sets the operation mode of the toggle bit (IN transaction only).

0: Toggle only when the transaction terminates normally.
1: Always toggle for each transaction performed.

**Bit 5** **EnEndPoint**
Setting this bit to 1 enables the endpoint.
Accesses to the endpoint are ignored when this bit = 0.
Set the appropriate value for this bit following a SetConfiguration request from the host.

0: Disables the endpoint.
1: Enables the endpoint.

**Bit 4** **DoubleBuf**
Setting this bit to 1 configures the FIFO for the endpoint as double buffers. A memory space twice the size set by MaxPacketSize is reserved in the FIFO.

0: Configure the FIFO as a single buffer.
1: Configure the FIFO as double buffers.

**Bit 3** **Reserved**

**Bits 2-0** **MaxPacketSize**
Sets the maximum value of the packet size. The relationship between the set values and packet sizes is shown below. (bit 2, 1, 0)

| MaxPacketSize | | |
|---|---|---|
| | FS | HS |
| 000 | Reserved | Reserved |
| 001 | 8 bytes | 8 bytes |
| 010 | 16 bytes | 16 bytes |
| 011 | 32 bytes | 32 bytes |
| 100 | 64 bytes | 64 bytes |
| 101 | | 512 bytes |
| 110 | | |
| 111 | Reserved | 1024 bytes |

The settings other than 512 bytes in the HS mode are used in interrupt transfers. Use EPc for isochronous transfer.
After setting MaxPacketSize and DoubleBuf for the endpoints, be sure to set the EPrControl register ALLFIFOClr bit to 1 to clear all FIFOs.
In addition, ensure that the total FIFO area reserved by endpoints a, b, c does not exceed 2432 bytes.

### 7.2.64 5Ah EPb Control_0 (EPbControl_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 5Ah | EPbControl_0 | R/W | 7: AutoForceNAK | 0: Normal | 1: AutoForceNAK | 00h |
| | | W | 6: EnShortPkt | 0: Normal | 1: Send Short Packet | |
| | | R/W | 5: AutoForceNAKShort | 0: Normal | 1: AutoForceNAKShort | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: ForceNAK | 0: Normal | 1: ForceNAK | |
| | | R/W | 0: ForceSTALL | 0: Normal | 1: ForceSTALL | |

This register sets the operation of endpoint b.



**Bit 7** **AutoForceNAK**
This bit automatically sets the EPbControl_0 register's ForceNAK bit to 1 when the transaction completes normally.

0: Do not automatically set ForceNAK bit.
1: Automatically set ForceNAK bit.

**Bit 6** **EnShortPkt(IN transaction only)**
Setting this bit to 1 allows the data in the current FIFO to be transmitted as a short packet for an IN transaction. This bit is automatically cleared to 0 when the packet transfer is finished. If this bit is set to 1 when no data exists in the FIFO, a packet of zero length is transmitted in response to an IN token from the host.

**Bit 5** **AutoForceNAKShort**
When this bit = 1, if the packet received during an OUT transaction that finished normally is a short packet, the ForceNAK bit is automatically set to 1.
If the AutoForceNAK bit = 1, AutoForceNAK is given priority over this bit.

0: Do not automatically set the ForceNAK bit to 1.
1: Automatically set the ForceNAK bit to 1.

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **ForceNAK**
Setting this bit to 1 makes a NAK response to the transaction regardless of the data count and free space in the FIFO. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect in the subsequent transactions.

**Bit 0** **ForceSTALL**
Setting this bit to 1 causes the transaction to be responded by a STALL. This bit is given priority over the ForceNAK bit.
If a transaction is currently underway and this bit was set a certain time after the transaction started, the setting of this bit takes effect from the following transaction.

### 7.2.65 5Bh EPb Control_1 (EPbControl_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 5Bh | EPbControl_1 | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R | 3: ToggleStat | Toggle Status | | |
| | | | 2: | 0: | 1: | |
| | | W | 1: ToggleSet | 0: Normal | 1: Toggle Set | |
| | | W | 0: ToggleClr | 0: Normal | 1: Toggle Clear | |

This register indicates and controls the status of the Endpoint b toggle bit.



**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **ToggleStat**
Indicates the status of the toggle sequence bit.

**Bit 2** **Reserved**

**Bit 1** **ToggleSet**
Setting this bit to 1 sets the toggle sequence bit to 1.

**Bit 0** **ToggleClr**
Setting this bit to 1 clears the toggle sequence bit to 0.

### 7.2.66 5Ch EPb FIFO Remain High (EPbFIFORemain_H)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 5Ch | EPbFIFORemain_H | R | 7: EPbFIFORemainCounter[15] | Endpoint b FIFO Remain Counter High | 00h |
| | | | 6: EPbFIFORemainCounter[14] | | |
| | | | 5: EPbFIFORemainCounter[13] | | |
| | | | 4: EPbFIFORemainCounter[12] | | |
| | | | 3: EPbFIFORemainCounter[11] | | |
| | | | 2: EPbFIFORemainCounter[10] | | |
| | | | 1: EPbFIFORemainCounter[9] | | |
| | | | 0: EPbFIFORemainCounter[8] | | |

This register shows the eight high-order bits that represent the remaining data counts in the Endpoint b FIFO. To acquire the FIFO's remaining data counts, access the EPbFIFORemain_H and EPbFIFORemain_L registers in pairs. Be sure to access the EPbFIFORemain_H register first.

### 7.2.67 5Dh EPb FIFO Remain Low (EPbFIFORemain_L)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 5Dh | EPbFIFORemain_L | R | 7: EPbFIFORemainCounter[7] | Endpoint b FIFO Remain Counter Low | 00h |
| | | | 6: EPbFIFORemainCounter[6] | | |
| | | | 5: EPbFIFORemainCounter[5] | | |
| | | | 4: EPbFIFORemainCounter[4] | | |
| | | | 3: EPbFIFORemainCounter[3] | | |
| | | | 2: EPbFIFORemainCounter[2] | | |
| | | | 1: EPbFIFORemainCounter[1] | | |
| | | | 0: EPbFIFORemainCounter[0] | | |

This register shows the eight low-order bits that represent the remaining data counts in the Endpoint b FIFO. To acquire the FIFO's remaining data counts, access the EPbFIFORemain_H and EPbFIFORemain_L registers in pairs. Be sure to access the EPbFIFORemain_H register first.

### 7.2.68 5Eh EPb FIFO for CPU (EPbFIFOforCPU)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 5Eh | EPbFIFOforCPU | R/W | 7: EPbFIFOData[7] | Endpoint b FIFO Access from CPU | XXh |
| | | | 6: EPbFIFOData[6] | | |
| | | | 5: EPbFIFOData[5] | | |
| | | | 4: EPbFIFOData[4] | | |
| | | | 3: EPbFIFOData[3] | | |
| | | | 2: EPbFIFOData[2] | | |
| | | | 1: EPbFIFOData[1] | | |
| | | | 0: EPbFIFOData[0] | | |

This register is used for FIFO access from the CPU.
When the EPbFIFOControl register EnFIFOwr bit is set to 1, data can be written into the FIFO by writing a value in this register.

When the EPbFIFOControl register EnFIFOrd bit is set to 1, data can be read from the FIFO by reading the value from this register.
If a value is written in this register without setting the EnFIFOwr bit, writing in the FIFO is not executed. If a value is read from the register without setting the EnFIFOrd bit, dummy data is output.

### 7.2.69 5Fh EPb FIFO Control (EPbFIFOControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 5Fh | EPbFIFOControl | R | 7: FIFOEmpty | 0: FIFO Not Empty | 1: FIFO Empty | 80h |
| | | R | 6: FIFOFull | 0: FIFO Not Full | 1: FIFO Full | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | W | 2: FIFOClr | 0: Normal | 1: FIFO Clear | |
| | | R/W | 1: EnFIFOwr | 0: Normal | 1: Enable FIFO write | |
| | | R/W | 0: EnFIFOrd | 0: Normal | 1: Enable FIFO read | |

This register shows or controls the Endpoint b FIFO status.

**Bit 7** **FIFOEmpty**
When this bit = 1, the FIFO is empty. Before reading data from the FIFO, check to see that this bit = 0.

**Bit 6** **FIFOFull**
When this bit = 1, the FIFO is full. Before writing data into the FIFO, check to see that this bit = 0.

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **FIFOClr**
Setting this bit to 1 clears the FIFO. This bit is automatically cleared to 0 after the FIFO is cleared.

**Bit 1** **EnFIFOwr**
Setting this bit to 1 allows data to be written to the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOrd bit is 1.

**Bit 0** **EnFIFOrd**
Setting this bit to 1 allows data to be read from the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOwr bit is 1.

### 7.2.70 60h EPc Config_0 (EPcConfig_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 60h | EPcConfig_0 | R/W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | 6: ISO | 0: Normal | 1: ISO | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: EndPointNumber[3] | Endpoint Number | | |
| | | | 2: EndPointNumber[2] | | | |
| | | | 1: EndPointNumber[1] | | | |
| | | | 0: EndPointNumber[0] | | | |

This register sets Endpoint c.



**Bit 7** **INxOUT**
Sets the transfer direction of the endpoint.
0: OUT direction
1: IN direction

**Bit 6** **ISO**
Setting this bit to 1 places the EPc endpoint in isochronous mode.
0: Use the endpoint for bulk or interrupt transfer
1: Use the endpoint for isochronous transfer

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bits 3-0** **EndPointNumber**
These bits set an endpoint number in the range of 01 to 15.

### 7.2.71 61h EPc Config_1 (EPcConfig_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 61h | EPcConfig_1 | R/W | 7: JoinIDE | 0: Not Join IDE | 1: Join IDE | 00h |
| | | R/W | 6: ToggleMode | 0: | 1: | |
| | | R/W | 5: EnEndPoint | 0: Disable Endpoint | 1: Enable Endpoint | |
| | | R/W | 4: DoubleBuf | 0: Single Buffer | 1: Double Buffer | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: MaxPacketSize[2] | Max Packet Size | | |
| | | | 1: MaxPacketSize[1] | | | |
| | | | 0: MaxPacketSize[0] | | | |

This register sets Endpoint c.



**Bit 7** **JoinIDE**
Connects the endpoint to DMA.
DMA connects to the last endpoint that had this bit set to 1. Immediately following the reset, the JoinIDE bit is 0 for all endpoints.

0: Do not connect this endpoint to DMA.
1: Connect this endpoint to DMA.

**Bit 6** **ToggleMode**
Sets the operation mode of the toggle bit (IN transaction only).

0: Toggle only when the transaction terminates normally.
1: Always toggle for each transaction performed.

**Bit 5** **EnEndPoint**
Setting this bit to 1 enables the endpoint. Accesses to the endpoint are ignored when this bit = 0. Set the appropriate value for this bit following a SetConfiguration request from the host.

0: Disables the endpoint.
1: Enables the endpoint.

**Bit 4** **DoubleBuf**
Setting this bit to 1 configures the FIFO for the endpoint as double buffers. A memory space twice the size set by MaxPacketSize is reserved in the FIFO.

0: Configure the FIFO as a single buffer.
1: Configure the FIFO as double buffers.

**Bit 3** **Reserved**

**Bits 2-0** **MaxPacketSize**
Sets the maximum value of the packet size. The relationship between set values and packet sizes is shown below. (bit 2, 1, 0)

| MaxPacketSize | | |
|---|---|---|
| | FS | HS |
| 000 | Reserved | Reserved |
| 001 | 8 bytes | 8 bytes |
| 010 | 16 bytes | 16 bytes |
| 011 | 32 bytes | 32 bytes |
| 100 | 64 bytes | 64 bytes |
| 101 | | 512 bytes |
| 110 | Reserved | Reserved |
| 111 | Reserved | 1024 bytes |

The settings other than 512 bytes in the HS mode are used in interrupt transfers. For isochronous transfer, this type of transfer can be accomplished by setting the EPcConfig_0 register ISO bit to 1. In this case, the values set in the IsoMaxSize_H and IsoMaxSize_L registers are used for MaxPacketSize, and the value set in MaxPacketSize here is ignored.

After setting MaxPacketSize and DoubleBuf for the endpoints, be sure to set the EPrControl register ALLFIFOClr bit to 1 to clear all FIFOs.
In addition, ensure that the total FIFO area reserved by endpoints a, b, c does not exceed 2,432 bytes.

### 7.2.72 62h EPc Control_0 (EPcControl_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 62h | EPcControl_0 | R/W | 7: AutoForceNAK | 0: Normal | 1: AutoForceNAK | 00h |
| | | W | 6: EnShortPkt | 0: Normal | 1: Send Short Packet | |
| | | R/W | 5: AutoForceNAKShort | 0: Normal | 1: AutoForceNAKShort | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: ForceNAK | 0: Normal | 1: ForceNAK | |
| | | R/W | 0: ForceSTALL | 0: Normal | 1: ForceSTALL | |

This register sets the operation of endpoint c.



**Bit 7** **AutoForceNAK**
This bit automatically sets the EPcControl_0 register's ForceNAK bit to 1 when the transaction completes normally.

0: Do not automatically set ForceNAK bit.
1: Automatically set ForceNAK bit.

**Bit 6** **EnShortPkt(IN transaction only)**
Setting this bit to 1 allows the data in the current FIFO to be transmitted as a short packet for an IN transaction. This bit is automatically cleared to 0 when the packet transfer is finished. If this bit is set to 1 when no data exists in the FIFO, a packet of zero length is transmitted in response to an IN token from the host.

**Bit 5** **AutoForceNAKShort**
When this bit = 1, if the packet received during an OUT transaction that completes normally is a short packet, the ForceNAK bit is automatically set to 1. If the AutoForceNAK bit = 1, AutoForceNAK is given priority over this bit.

0: Do not automatically set the ForceNAK bit to 1.
1: Automatically set the ForceNAK bit to 1.

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **ForceNAK**
Setting this bit to 1 makes a NAK response to the transaction regardless of the data count and free space in the FIFO. If any transaction is currently underway and this bit was set a certain time after the transaction started, the bit setting takes effect in the subsequent transactions.

**Bit 0** **ForceSTALL**
Setting this bit to 1 causes the transaction to be responded by a STALL. This bit is given priority over the ForceNAK bit. If a transaction is currently underway and this bit was set a certain time after the transaction started, the setting of this bit takes effect from the following transaction.

### 7.2.73 63h EPc Control_1 (EPcControl_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 63h | EPcControl_1 | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R | 3: ToggleStat | Toggle Status | | |
| | | | 2: | 0: | 1: | |
| | | W | 1: ToggleSet | 0: Normal | 1: Toggle Set | |
| | | W | 0: ToggleClr | 0: Normal | 1: Toggle Clear | |

This register indicates and controls the status of the Endpoint c toggle bit.



**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **ToggleStat**
Indicates the status of the toggle sequence bit.

**Bit 2** **Reserved**

**Bit 1** **ToggleSet**
Setting this bit to 1 sets the toggle sequence bit to 1.

**Bit 0** **ToggleClr**
Setting this bit to 1 clears the toggle sequence bit to 0.

### 7.2.74 64h EPc FIFO Remain High (EPcFIFORemain_H)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 64h | EPcFIFORemain_H | R | 7: EPcFIFORemainCounter[15] | Endpoint c FIFO Remain Counter High | 00h |
| | | | 6: EPcFIFORemainCounter[14] | | |
| | | | 5: EPcFIFORemainCounter[13] | | |
| | | | 4: EPcFIFORemainCounter[12] | | |
| | | | 3: EPcFIFORemainCounter[11] | | |
| | | | 2: EPcFIFORemainCounter[10] | | |
| | | | 1: EPcFIFORemainCounter[9] | | |
| | | | 0: EPcFIFORemainCounter[8] | | |

This register shows the eight high-order bits that represent the remaining data counts in the Endpoint c FIFO. To acquire the FIFO's remaining data counts, access the EPcFIFORemain_H and EPcFIFORemain_L registers in pairs. Be sure to access the EPcFIFORemain_H register first.

### 7.2.75 65h EPc FIFO Remain Low (EPcFIFORemain_L)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 65h | EPcFIFORemain_L | R | 7: EPcFIFORemainCounter[7] | Endpoint c FIFO Remain Counter Low | 00h |
| | | | 6: EPcFIFORemainCounter[6] | | |
| | | | 5: EPcFIFORemainCounter[5] | | |
| | | | 4: EPcFIFORemainCounter[4] | | |
| | | | 3: EPcFIFORemainCounter[3] | | |
| | | | 2: EPcFIFORemainCounter[2] | | |
| | | | 1: EPcFIFORemainCounter[1] | | |
| | | | 0: EPcFIFORemainCounter[0] | | |

This register shows the eight low-order bits that represent the remaining data counts in the Endpoint c FIFO. To acquire the FIFO's remaining data counts, access the EPcFIFORemain_H and EPcFIFORemain_L registers in pairs. Be sure to access the EPcFIFORemain_H register first.

### 7.2.76 66h EPc FIFO for CPU (EPcFIFOforCPU)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 66h | EPcFIFOforCPU | R/W | 7: EPcFIFOData[7] | Endpoint c FIFO Access from CPU | XXh |
| | | | 6: EPcFIFOData[6] | | |
| | | | 5: EPcFIFOData[5] | | |
| | | | 4: EPcFIFOData[4] | | |
| | | | 3: EPcFIFOData[3] | | |
| | | | 2: EPcFIFOData[2] | | |
| | | | 1: EPcFIFOData[1] | | |
| | | | 0: EPcFIFOData[0] | | |

This register is used for FIFO access from the CPU.
When the EPcFIFOControl register EnFIFOwr bit is set to 1, data can be written into the FIFO by writing a value in this register.
When the EPcFIFOControl register EnFIFOrd bit is set to 1, data can be read from the FIFO by reading the value from this register.
If a value is written in this register without setting the EnFIFOwr bit, writing in the FIFO is not executed. If a value is read from the register without setting the EnFIFOrd bit, dummy data is output.



### 7.2.77 67h EPc FIFO Control (EPcFIFOControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 67h | EPcFIFOControl | R | 7: FIFOEmpty | 0: FIFO Not Empty | 1: FIFO Empty | 80h |
| | | R | 6: FIFOFull | 0: FIFO Not Full | 1: FIFO Full | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | W | 2: FIFOClr | 0: Normal | 1: FIFO Clear | |
| | | R/W | 1: EnFIFOwr | 0: Normal | 1: Enable FIFO write | |
| | | R/W | 0: EnFIFOrd | 0: Normal | 1: Enable FIFO read | |

This register shows or controls the Endpoint c FIFO status.

**Bit 7** **FIFOEmpty**
When this bit = 1, the FIFO is empty.
Before reading data from the FIFO, check to see that this bit = 0.

**Bit 6** **FIFOFull**
When this bit = 1, the FIFO is full.
Before writing data into the FIFO, check to see that this bit = 0.

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **FIFOClr**
Setting this bit to 1 clears the FIFO.
This bit is automatically cleared to 0 after the FIFO is cleared.

**Bit 1** **EnFIFOwr**
Setting this bit to 1 allows data to be written to the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOrd bit is 1.

**Bit 0** **EnFIFOrd**
Setting this bit to 1 allows data to be read from the FIFO by the CPU.
This bit cannot be set to 1 when the EnFIFOwr bit is 1.

### 7.2.78 68h Iso Max Packet Size High (IsoMaxSize_H)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 68h | IsoMaxSize_H | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: IsoMaxPacketSize[10] | IsoMaxPacketSize[10:8] | | |
| | | | 1: IsoMaxPacketSize[9] | | | |
| | | | 0: IsoMaxPacketSize[8] | | | |

### 7.2.79 69h Iso Max Packet Size Low (IsoMaxSize_L)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 69h | IsoMaxSize_L | R/W | 7: IsoMaxPacketSize[7] | IsoMaxPacketSize[7:2] | | 00h |
| | | | 6: IsoMaxPacketSize[6] | | | |
| | | | 5: IsoMaxPacketSize[5] | | | |
| | | | 4: IsoMaxPacketSize[4] | | | |
| | | | 3: IsoMaxPacketSize[3] | | | |
| | | | 2: IsoMaxPacketSize[2] | | | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

When the transaction type of endpoint c is set to ISO (the EPcConfig_0.ISO bit is set to 1), the EPcConfig_1 register MaxPacketSize bit setting is ignored and the value set in this register becomes effective. Set the MaxPacketSize in units of 4 bytes.



FS: 4 to 1,020 bytes
HS: 4 to 1,024 bytes

Ensure that the total FIFO area reserved by endpoints a, b, c does not exceed 2,432 bytes.

### 7.2.80 6Ah to 7Fh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 6Ah to 7Fh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.81 80h IDE Status (IDEStatus)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 80h | IDEStatus | R | 7: DMARQ | DMARQ signal | | XXh |
| | | R | 6: DMACK | DMACK signal | | |
| | | R | 5: INTRQ | INTRQ signal | | |
| | | R | 4: IORDY | IORDY signal | | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R | 1: PDIAG | PDIAG signal | | |
| | | R | 0: DASP | DASP signal | | |

This register indicates the status of the IDE interface signals.

**Bit 7** **DMARQ**
Indicates the HDMARQ signal status with positive logic. (Reflects the value of the IDE_CONFIG_1 register PDREQLevel bit.)

**Bit 6** **DMACK**
Indicates the XHDMACK signal status with positive logic.

**Bit 5** **INTRQ**
Indicates the HINTRQ signal status with positive logic.

**Bit 4** **IORDY**
Indicates the HIORDY signal status with positive logic.

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **PDIAG**
Indicates the XHPDIAG signal status with positive logic.

**Bit 0** **DASP**
Indicates the XHDASP signal status with positive logic.

### 7.2.82 81h IDE Config_0 (IDEConfig_0)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 81h | IDEConfig_0 | W | 7: IDEBusReset | 0: Normal | 1: IDE Bus Reset | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | R/W | 2: NotIDE | 0: IDE mode | 1: Port mode | |
| | | R/W | 1: ULTRA | 0: Normal | 1: ULTRA mode | |
| | | R/W | 0: DMA | 0: Normal | 1: DMA mode | |

This register sets the operation of the IDE interface.



**Bit 7** **IDEBusReset**
Setting this bit to 1 asserts the Reset signal to the IDE interface for a 50-µs period. When this bit is read during assertion of XHRESET, it indicates the value 1. If this bit is set again during assertion, XHRESET is output for a 50-µs period from that point in time.

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **NotIDE**
This bit changes the minimum width of assert and negate pulses of strobe signals set by the IDE_Rmod and IDE_Tmod registers. When using the IDE interface in the general-purpose DMA mode, setting this bit to 1 reduces the minimum values of strobe signals, which can improve the transfer rate. Note that when using the IDE interface in the IDE bus-compatible mode, setting this bit to 1 accepts set values out of the IDE specifications.
0: AssertPulseWidth in the IDE_Rmod and IDE_Tmod registers +5 x 16.6 ns (internal clock period 60 MHz) is set as an assert pulse width.
NegatePulseWidth in the IDE_Rmod and IDE_Tmod registers +3 x 16.6 ns (internal clock period 60 MHz) is set as a negate pulse width.
1: AssertPulseWidth in the IDE_Rmod and IDE_Tmod registers +2 x 16.6 ns (internal clock period 60 MHz) is set as an assert pulse width.
NegatePulseWidth in the IDE_Rmod and IDE_Tmod registers +2 x 16.6 ns (internal clock period 60 MHz) is set as a negate pulse width.

**Bit 1** **ULTRA**
When the DMA bit is set, this bit selects ULTRA-DMA for the DMA transfer mode.
Always use this bit along with the DMA bit. Setting this bit to 1 ignores the setting of the BUS8 bit.

0: Do not perform data transfer in ULTRA-DMA mode.
1: Perform data transfer in ULTRA-DMA mode.

**Bit 0** **DMA**
Selects DMA for transfer mode. Unless this bit is set, transfers are performed in PIO mode.

0: Perform data transfer in PIO mode.
1: Perform data transfer in DMA mode.

### 7.2.83 82h IDE Config_1 (IDEConfig_1)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 82h | IDEConfig_1 | R/W | 7: ActiveIDE | 0: Non-Active IDE | 1: Active IDE | 00h |
| | | R/W | 6: DelayStrobe | 0: Normal | 1: Delay Strobe | |
| | | R/W | 5: Slave | 0: Master | 1: Slave | |
| | | R/W | 4: InterLock | 0: Normal | 1: Interlock | |
| | | R/W | 3: PDREQLevel | 0: Active High | 1: Active Low | |
| | | R/W | 2: Swap | 0: Normal | 1: Swap | |
| | | | 1: | 0: | 1: | |
| | | R/W | 0: Bus8 | 0: Bus16 | 1: Bus8 | |

This register sets the IDE interface bus operations.



**Bit 7** **ActiveIDE**
Following the reset, the IDE interface by default has all of its pins set for the Hi-Z mode. Setting this bit to 1 enables the IDE (general-purpose DMA). When this bit is set to 1, each pin switches to the input or output mode depending on how the Slave bit is set.

0: Disables the IDE and PortDMA.
1: Enables the IDE and PortDMA.

**Bit 6** **DelayStrobe**
Setting this bit to 1 delays the strobe output by 2 τ from XHDMACK during multiword DMA.

0: Do not delay the strobe output by 2 τ from XHDMACK.
1: Delay the strobe output by 2 τ from XHDMACK.

**Bit 5** **Slave**
Determines the operation mode of the IDE (general-purpose DMA) interface unit. Set this bit to 0 when using the IDE interface in the IDE bus-compatible mode.
In the slave mode, the register functions from IDE_Rmod to IDE_CS17 cannot be used.
Note that there is a delay time of 25 ns (typ) from XHIOR or XHIOW assertion to HDMARQ negation.

0: Master mode (HDMARQ functions as input; XHDMACK, XHIOR, and XHIOW function as output)
1: Slave mode (HDMARQ functions as output; XHDMACK, XHIOR, and XHIOW function as input)

**Bit 4** **InterLock**
Effective only in master DMA mode. When this bit = 0, XHDMACK is negated if the FIFO becomes incapable of transfer.
When this bit = 1, XHDMACK is not negated even if the FIFO becomes incapable of transfer.
However, XHDMACK is negated when HDMARQ is dropped.

0: Negate XHDMACK.
1: Do not negate XHDMACK.

**Bit 3** **PDREQLevel**
Determines the active level of the HDMARQ signal. Set this bit to 0 when using the IDE interface as IDE-like DMA.

0: Positive logic
1: Negative logic

**Bit 2** **Swap**
Swaps the high-order and low-order 8 bits when using the IDE interface at 16-bit width.
The order in which the IDE_CS00 register is accessed is also reversed.

0: Transfer the low-order 8-bit data to the USB side first.
1: Transfer the high-order 8-bit data to the USB side first.

**Bit 1** **Reserved**

**Bit 0** **Bus8**
Set this bit to 1 when using the IDE (general-purpose DMA) interface with an 8-bit width.
When this bit is set to 1, only the 8 low-order bits of the bus are valid. The 8 high-order bits are invalid. When using the IDE interface at 8-bit width, 8 high-order bits must be fixed at high or low using pull-up/down resistors.
Clear this bit to 0 when using the IDE interface in the IDE bus-compatible mode.
This bit is ignored if the IDE_Config_0 register ULTRA bit is set.

0: Use the IDE interface at 16-bit width.
1: Use the IDE interface at 8-bit width.

**Setting the port interface operations**

The following tables show the relationship between the operation modes set and the bit settings of this register (IDE_Config_1). When the port data bus is used as the IDE interface, bits 5, 3, and 0 do not need to be set. The maximum delay time before HDMARQ is negated in the slave mode is ≤ 37 ns, from the time at which XHIOW or XHIOR is asserted to the time at which HDMARQ is negated.

Slave/master switching of ports by the Slave bit

| | HDMARQ | XHDMACK | XHIOR / XHIOW | Remarks |
|---|---|---|---|---|
| Slave = 0 (master) | Input | Output | Output | Data input during XHIOR (Read)<br>Data output during XHIOW (Write)<br>Tmod settings effective<br>XHIOR/ XHIOW minimum pulse width: assertion >70 ns; negation >40 ns |
| Slave = 1 (slave) | Output | Input | Input | Data output during XHIOR (Read)<br>Data input during XHIOW (Write)<br>Tmod settings have no effect.<br>XHIOR/ XHIOW minimum pulse<br>Assertion ≥ 25 ns<br>Negation ≥ 25 ns<br>XHIOW period ≥ 50 ns |



Switching of operation modes by the Bus8 and Swap bits

| | | |
|---|---|---|
| Bus8 = 0 | Swap = 0 | HDD7–0 is transferred first.<br>The first data obtained by accessing IDE_CS00 is HDD7–0. |
| | Swap = 1 | HDD15–8 is transferred first.<br>The first data obtained by accessing IDE_CS00 is HDD15–8. |
| Bus8 = 1 | | Only HDD7–0 is used for transfer.<br>HDD15–8 is in input mode.<br>(These unused bits must be pulled high or low.)<br>IDE_CS00 is accessed for only HDD7–0. |

* The input/output delays depend on the magnitude of load of the connected device. The values shown above are derived by simulation assuming a load capacitance of 20 pF, with 10 ns margins added to each estimated value.

### 7.2.84 83h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 83h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.85 84h IDE Register Mode (IDE_Rmod)



| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 84h | IDE_Rmod | R/W | 7: RegisterAssertPulseWidth[3] | Register Assert Pulse Width | 00h |
| | | | 6: RegisterAssertPulseWidth[2] | | |
| | | | 5: RegisterAssertPulseWidth[1] | | |
| | | | 4: RegisterAssertPulseWidth[0] | | |
| | | R/W | 3:RegisterNegatePulseWidth[3] | Register Negate Pulse Width | |
| | | | 2:RegisterNegatePulseWidth[2] | | |
| | | | 1:RegisterNegatePulseWidth[1] | | |
| | | | 0:RegisterNegatePulseWidth[0] | | |

This register sets the access to the IDE register area on the IDE interface.

**Bits 7-4 RegisterAssertPulseWidth**
Determine the minimum value of the period for which the strobe signal is asserted when accessing the register area of the IDE interface. When the IDE_Config_0 register NotIDE bit is 0, the setting is as follows:
Internal clock (60 MHz) period multiplied by (RegisterAssertPulseWidth + 5)
When the IDE_Config_0 register NotIDE bit is 1, the setting is as follows:
Internal clock (60 MHz) period multiplied by (RegisterAssertPulseWidth + 2)
When the IDE_Config_0 register NotIDE bit is 0, the setting is as follows:

**Bits 3-0 RegisterNegatePulseWidth**
Determine the minimum value of the period for which the strobe signal is negated when accessing the register area of the IDE interface. When the IDE_Config_0 register NotIDE bit is 0, the setting is as follows:
Internal clock (60 MHz) period multiplied by (RegisterNegatePulseWidth + 3)
When the IDE_Config_0 register NotIDE bit is 1, the setting is as follows:
Internal clock (60 MHz) period multiplied by (RegisterNegatePulseWidth + 2)

Example: When the IDE_Config_0 register NotIDE bit is 0:
00h: AssertPulseWidth 16.6 ns × (0000 + 5) = 83 ns and NegatePulseWidth 16.6 ns × (0000 + 3) = 49.8 ns
11h: AssertPulseWidth 16.6 ns × (0001 + 5) = 99.6 ns and NegatePulseWidth 16.6 ns × (0001 + 3) = 66.4 ns
When the IDE_Config_0 register NotIDE bit is 1:
00h: AssertPulseWidth 16.6 ns × (0000 + 2) = 33.2 ns and NegatePulseWidth 16.6 ns × (0000 + 2) = 33.2 ns
11h: AssertPulseWidth 16.6 ns × (0001 + 2) = 49.8 ns and NegatePulseWidth 16.6 ns × (0001 + 2) = 49.8 ns

### 7.2.86 85h IDE Transfer Mode (IDE_Tmod)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 85h | IDE_Tmod | R/W | 7: TransferAssertPulseWidth[3] | Transfer Assert Pulse Width | 00h |
| | | | 6: TransferAssertPulseWidth[2] | | |
| | | | 5: TransferAssertPulseWidth[1] | | |
| | | | 4: TransferAssertPulseWidth[0] | | |
| | | R/W | 3:TransferNegatePulseWidth[3] | Transfer Negate Pulse Width | |
| | | | 2:TransferNegatePulseWidth[2] | | |
| | | | 1:TransferNegatePulseWidth[1] | | |
| | | | 0:TransferNegatePulseWidth[0] | | |

This register sets the manner in which data transfers are performed via the IDE interface.



**Bits 7-4 TransferAssertPulseWidth**
Determine the minimum value of the period for which the strobe signal is asserted when performing data transfers via the IDE interface. When the IDE_Config_0 register NotIDE bit is 0, the setting is as follows:
Internal clock (60 MHz) period multiplied by (TransferAssertPulseWidth + 5)
When the IDE_Config_0 register NotIDE bit is 1, the setting is as follows:
Internal clock (60 MHz) period multiplied by (TransferAssertPulseWidth + 2)

**Bits 3-0 TransferNegatePulseWidth**
Determine the minimum value of the period for which the strobe signal is negated when performing data transfers via the IDE interface. When the IDE_Config_0 register NotIDE bit is 0, the setting is as follows:
Internal clock (60 MHz) period multiplied by (TransferNegatePulseWidth + 3)
When the IDE_Config_0 register NotIDE bit is 1, the setting is as follows:
Internal clock (60 MHz) period multiplied by (TransferNegatePulseWidth + 2)

Example: When the IDE_Config_0 register NotIDE bit is 0:
00h: AssertPulseWidth 16.6 ns × (0000 + 5) = 83 ns and NegatePulseWidth 16.6 ns × (0000 + 3) = 49.8 ns
11h: AssertPulseWidth 16.6 ns × (0001 + 5) = 99.6 ns and NegatePulseWidth 16.6 ns × (0001 + 3) = 66.4 ns
When the IDE_Config_0 register NotIDE bit is 1:
00h: AssertPulseWidth 16.6 ns × (0000 + 2) = 33.2 ns and NegatePulseWidth 16.6 ns × (0000 + 2) = 33.2 ns
11h: AssertPulseWidth 16.6 ns × (0001 + 2) = 49.8 ns and NegatePulseWidth 16.6 ns × (0001 + 2) = 49.8 ns

### 7.2.87 86h IDE Ultra-DMA Transfer Mode (IDE_Umod)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 86h | IDE_Umod | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | R/W | 3: UltraDMACycle[3] | Ultra DMA Cycle | | |
| | | | 2: UltraDMACycle[2] | | | |
| | | | 1: UltraDMACycle[1] | | | |
| | | | 0: UltraDMACycle[0] | | | |

This register sets the minimum cycle time of the strobe signal when performing data transfers in Ultra-DMA mode via the IDE interface.

**Bit 7 Reserved**

**Bit 6 Reserved**

**Bit 5 Reserved**

**Bit 4 Reserved**

**Bits 3-0 UltraDMACycle**
Determine the minimum cycle time of the strobe signal when performing data transfers in Ultra-DMA mode via the IDE interface. This is the internal clock (60 MHz) period multiplied by (UltraDMACycle + 2).

Example: If these bits are set to '00h,' then UltraDMACycle = 16.6 ns × (0000 + 2) = 33.2 ns
If these bits are set to '01h,' then UltraDMACycle = 16.6 ns × (0001 + 2) = 49.8 ns

### 7.2.88 87h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 87h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.89 88h IDE Control_0 (IDEControl_0)



| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 88h | IDEControl_0 | W | 7: IDEFlush | 0: Normal | 1: IDE Flush | 00h |
| | | W | 6: IDEFClr | 0: Normal | 1: IDE FIFO Clear | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | R/W | 0: DTGO | 0: DMA Transfer Stop | 1: DMA Transfer Go | |

This register controls the IDE during transfer operations.

**Bit 7 IDEFlush**
Setting this bit to 1 terminates (or temporarily stops) the DMA transfer. When the DMA transfer is stopped, the DTGO bit is cleared to 0 and the IDEIntStat register DTCmp bit is set to 1.
By using this bit, DMA transfer can be stopped and restarted without causing loss of data. To restart the transfer, check that the DTGO bit is cleared to 0 (or wait until the DMA transfer stops), set a transfer count in the IDE_Count register, and set the DTGO bit to 1 again.
The use of this bit to stop the transfer during DMA reception bit may cause loss of data.
The period from the moment at which this bit is set to 1 to the moment at which the DMA transfer stops (the DTGO bit is cleared to 0) is the period necessary for 32-word (Max.) data transfer in Ultra DMA transfer or 8-byte data (Max.) transfer in the other DMA transfer modes.

0: Perform no operation.
1: Stop DMA transfer.

**Bit 6 IDEFClr**
Even if the other party to the DMA transfer aborts the operation by clearing DTGO to 0 while DMARQ it asserted remains active, DMACK is not negated. However, in such cases, setting this bit to 1 can forcibly negate DMACK.
Before setting this bit to 1, wait until the clock counts set by IDETmod expire after clearing DTGO to 0.

**Bit 5 Reserved**

**Bit 4 Reserved**

**Bit 3 Reserved**

**Bit 2 Reserved**

**Bit 1 Reserved**

**Bit 0 DTGO**
Setting this bit to 1 starts DMA transfer. This bit is automatically cleared to 0 when the DMA transfer is finished.
Setting this bit to 0 during DMA transfer forcibly terminates the transfer (the bit is cleared to 0 when the DMA transfer stops). Note that, however, forcible termination of DMA transfer may cause loss of data. When the DMA transfer is stopped, the IDEIntStat register DTCmp bit is set to 1.
The period from the moment at which this bit is set to 0 to the moment at which the DMA transfer stops is the period necessary for 32-word (Max.) data transfer in Ultra DMA transfer or 8-byte data (Max.) transfer in the other DMA transfer modes.

0: Stop DMA transfer.
1: Start DMA transfer.

### 7.2.90 89h Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 89h | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.91 8Ah IDE Transfer Byte Count High (IDE_Count_H)



| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 8Ah | IDE_Count_H | R/W | 7: IDE_Count[23] | IDE Count High | 00h |
| | | | 6: IDE_Count[22] | | |
| | | | 5: IDE_Count[21] | | |
| | | | 4: IDE_Count[20] | | |
| | | | 3: IDE_Count[19] | | |
| | | | 2: IDE_Count[18] | | |
| | | | 1: IDE_Count[17] | | |
| | | | 0: IDE_Count[16] | | |

This register sets the 23rd to 16th bits of the transfer byte count in DMA transfer. After the transfer is started by setting the IDE_Control_0 register DTGO bit, read this register to determine the value in the 23rd to 16th bits (high count value) of the remaining transfer byte count. To determine the full byte count that remains to be transferred, access registers IDE_Count_H, IDE_Count_M, and IDE_Count_L, in that order.

### 7.2.92 8Bh IDE Transfer Byte Count Middle (IDE_Count_M)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 8Bh | IDE_Count_M | R/W | 7: IDE_Count[15] | IDE Count Middle | 00h |
| | | | 6: IDE_Count[14] | | |
| | | | 5: IDE_Count[13] | | |
| | | | 4: IDE_Count[12] | | |
| | | | 3: IDE_Count[11] | | |
| | | | 2: IDE_Count[10] | | |
| | | | 1: IDE_Count[9] | | |
| | | | 0: IDE_Count[8] | | |

This register sets the 15th to 8th bits of the transfer byte count in DMA transfer. After the transfer is started by setting the IDE_Control_0 register DTGO bit, read this register to determine the value in the 15th to 8th (middle count value) of the remaining transfer byte count. To determine the full byte count that remains to be transferred, access registers IDE_Count_H, IDE_Count_M, and IDE_Count_L, in that order.

### 7.2.93 8Ch IDE Transfer Byte Count Low (IDE_Count_L)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 8Ch | IDE_Count_L | R/W | 7: IDE_Count[7] | IDE Count Low | 00h |
| | | | 6: IDE_Count[6] | | |
| | | | 5: IDE_Count[5] | | |
| | | | 4: IDE_Count[4] | | |
| | | | 3: IDE_Count[3] | | |
| | | | 2: IDE_Count[2] | | |
| | | | 1: IDE_Count[1] | | |
| | | | 0: IDE_Count[0] | | |

This register sets the 7th to 0th bits of the transfer byte count in DMA transfer. After the transfer is started by setting the IDE_Control_0 register DTGO bit, read this register to determine the value in the 7th to 0th (low count value) of the remaining transfer byte count. To determine the full byte count that remains to be transferred, access registers IDE_Count_H, IDE_Count_M, and IDE_Count_L, in that order.

### 7.2.94 8Dh IDE CRC Control (IDE_CRCControl)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| 8Dh | IDE_CRCControl | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | W | 0: Clear | 0: Normal | 1: IDE CRC Clear | |

This register controls CRC during Ultra-DMA transfer by IDE.



**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **Reserved**

**Bit 0** **Clear**
Setting this bit to 1 initializes the internal CRC calculation circuit during Ultra-DMA transfer. This bit is write-only, and is automatically cleared to 0 by completion of initialization. At DMA startup, this circuit is automatically initialized by the internal logic.

### 7.2.95 8Eh IDE CRC High (IDE_CRC_H)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 8Eh | IDE_CRC_H | R | 7: CRC[15] | CRC High | 4Ah |
| | | | 6: CRC[14] | | |
| | | | 5: CRC[13] | | |
| | | | 4: CRC[12] | | |
| | | | 3: CRC[11] | | |
| | | | 2: CRC[10] | | |
| | | | 1: CRC[9] | | |
| | | | 0: CRC[8] | | |

This register shows the 8 high-order bits of the result of the CRC calculation made when performing data transfers in Ultra-DMA mode via the IDE interface.
To acquire the CRC value of the IDE, access the IDE_CRC_H and IDE_CRC_L registers in pairs. Be sure to access the IDE_CRC_H register first.

### 7.2.96 8Fh IDE CRC Low (IDE_CRC_L)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 8Fh | IDE_CRC_L | R | 7: CRC[7] | CRC Low | BAh |
| | | | 6: CRC[6] | | |
| | | | 5: CRC[5] | | |
| | | | 4: CRC[4] | | |
| | | | 3: CRC[3] | | |
| | | | 2: CRC[2] | | |
| | | | 1: CRC[1] | | |
| | | | 0: CRC[0] | | |

This register shows the 8 low-order bits of the result of the CRC calculation made when performing data transfers in Ultra-DMA mode via the IDE interface.
To acquire the CRC value of the IDE, access the IDE_CRC_H and IDE_CRC_L registers in pairs. Be sure to access the IDE_CRC_H register first.

### 7.2.97 90h IDE_CS00 (IDE_CS00)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 90h | IDE_CS00 | R/W | 7: IDE_CS00[7] | IDE CS00 | XXh |
| | | | 6: IDE_CS00[6] | | |
| | | | 5: IDE_CS00[5] | | |
| | | | 4: IDE_CS00[4] | | |
| | | | 3: IDE_CS00[3] | | |
| | | | 2: IDE_CS00[2] | | |
| | | | 1: IDE_CS00[1] | | |
| | | | 0: IDE_CS00[0] | | |



This register shows the area accessed by the CPU as it accesses the IDE interface data ports.
The transfer mode is set to PIO mode, and the access is made under the conditions set in the IDE_Tmod register.
The settings for the IDE_Config_1 register's BUS8 and SWAP bits are reflected in the accesses performed here. Thus, for 16-bit wide operations, the IDE bus can be accessed in 16 bits by always accessing this register twice. For 8-bit wide operations, the IDE bus can be accessed in a single operation.
Accessing the IDE_CS00 register during DMA transfer is inhibited.

### 7.2.98 91h to 9Fh IDE_CS01 to IDE_CS17 (IDE_CS01 to IDE_CS17)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| 91h<br>to<br>9Fh | IDE_CS01<br>to<br>IDE_CS17 | R/W | 7: IDE_CSxx[7]<br>6: IDE_CSxx[6]<br>5: IDE_CSxx[5]<br>4: IDE_CSxx[4]<br>3: IDE_CSxx[3]<br>2: IDE_CSxx[2]<br>1: IDE_CSxx[1]<br>0: IDE_CSxx[0] | IDE CSxx | XXh |



These registers show the areas that are accessed by the CPU as it accesses the IDE interface register area.
The transfer mode is set to PIO mode, and the access is made under the conditions set in the IDE_Rmod register. Transfers are performed at a fixed length of 8 bits, using the bus signals DD7–0.

For accesses of registers IDE_CS01 through IDE_CS17 during DMA transfer, if the IDE_Config_1 register InterLock bit in DMA mode is 0, XHDMACK is temporarily negated before the CPU makes the access. However, during Ultra-DMA transfers, or if the InterLock bit = 1, the CPU cannot perform the access until XHDMACK is negated (when HDMARQ is dropped) or the transfer is complete.

The contents of each register in IDE operations are shown below:

**IDE_CS01**
Read: Shows the ATA Error register.
Write: Shows the ATA Features register.

**IDE_CS02**
Shows the ATA Sector Count register.

**IDE_CS03**
Shows the ATA Sector Number register.

**IDE_CS04**
Shows the ATA Cylinder Low register.

**IDE_CS05**
Shows the ATA Cylinder High register.

**IDE_CS06**
Shows the ATA Device/Head register.

**IDE_CS07**
Read: Shows the ATA Status register.
Write: Shows the ATA Command register.

**IDE_CS10**

**IDE_CS11**

**IDE_CS12**

**IDE_CS13**

**IDE_CS14**

**IDE_CS15**

**IDE_CS16**
Read: Shows the ATA Alternate Status register.
Write: Shows the ATA Device Control register.

**IDE_CS17**

### 7.2.99 A0h to BEh CBW_00 to CSW_30 (CBW_00 to CBW_30)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| A0h to BEh | CBW_00 to CBW_30 | R/W | 7: CBW_xx[7] | Bulk Out Received CBW Data | XXh |
| | | | 6: CBW_xx[6] | | |
| | | | 5: CBW_xx[5] | | |
| | | | 4: CBW_xx[4] | | |
| | | | 3: CBW_xx[3] | | |
| | | | 2: CBW_xx[2] | | |
| | | | 1: CBW_xx[1] | | |
| | | | 0: CBW_xx[0] | | |



These registers are used in the USB storage-class BulkOnly transport protocol.
When the BulkOnlyControl register GoCBWMode bit = 1 and valid CBW data is received at the set Bulk OUT endpoint, the BulkIntStat register's CBWCmp interrupt is generated, and the received CBW data is stored in these registers. If a CBWShort, CBWLong, or CBWErr interrupt occurs, the contents of these registers are invalid. If a CBWCmp, CBWShort, or CBWLong interrupt occurs, the BulkOnlyControl register GoCBWMode bit is automatically cleared to 0. However, when a CBWErr interrupt occurs, the GoCBWMode bit is not automatically cleared.
These registers can be accessed only when the GoCBWMode bit = 0.

### 7.2.100 BFh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| BFh | (Reserved) | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.101 C0h to CCh CSW0_00 to CSW0_12 (CSW0_00 to CSW0_12)

| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| C0h to CCh | CSW0_00 to CSW0_12 | R/W | 7: CSW0_xx[7] | Bulk In Transfer CSW0 Data | XXh |
| | | | 6: CSW0_xx[6] | | |
| | | | 5: CSW0_xx[5] | | |
| | | | 4: CSW0_xx[4] | | |
| | | | 3: CSW0_xx[3] | | |
| | | | 2: CSW0_xx[2] | | |
| | | | 1: CSW0_xx[1] | | |
| | | | 0: CSW0_xx[0] | | |

These registers are used in the USB storage-class BulkOnly transport protocol.
When the BulkOnlyControl register GoCSWMode bit = 1, if the set Bulk IN endpoint has a CSW transmit request and the BulkOnlyControl register CSWSel bit = 0, the contents of the CSW0_00 to CSW0_12 registers are transmitted. The BulkIntStat register's CSWCmp or CSWErr interrupt is generated after transmitting the CSW. When a CSWCmp interrupt occurs, the GoCSWMode bit is cleared to 0 and the GoCBWMode bit is set to 1. When a CSWErr interrupt occurs, however, neither the GoCSWMode bit is cleared, nor is the GoCBWMode bit set.
These registers can be accessed only when the GoCSWMode bit = 0.

### 7.2.102 CDh to CFh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| CDh | (Reserved) | | 7: | 0: | 1: | 00h |
| to | | | 6: | 0: | 1: | |
| CFh | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.103 D0h to DCh CSW1_00 to CSW1_12 (CSW1_00 to CSW1_12)



| Address | Register Name | R/W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|
| D0h | CSW1_00 | R/W | 7: CSW1_xx[7] | Bulk In Transfer CSW1 Data | XXh |
| to | to | | 6: CSW1_xx[6] | | |
| DCh | CSW1_12 | | 5: CSW1_xx[5] | | |
| | | | 4: CSW1_xx[4] | | |
| | | | 3: CSW1_xx[3] | | |
| | | | 2: CSW1_xx[2] | | |
| | | | 1: CSW1_xx[1] | | |
| | | | 0: CSW1_xx[0] | | |

These registers are used in the USB storage-class BulkOnly transport protocol.
When the BulkOnlyControl register GoCSWMode bit = 1, if the set Bulk IN endpoint has a CSW transmit request and the BulkOnlyControl register CSWSel bit = 1, the contents of the CSW1_00 through CSW1_12 registers are transmitted. The BulkIntStat register's CSWCmp or CSWErr interrupt is generated after transmitting the CSW. When a CSWCmp interrupt occurs, the GoCSWMode bit is cleared to 0 and the GoCBWMode bit is set to 1. However, when a CSWErr interrupt occurs, however, neither the GoCSWMode bit is cleared, nor is the GoCBWMode bit set.
These registers can be accessed only when the GoCSWMode bit = 0.

### 7.2.104 DDh to DFh Reserved

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| DDh | (Reserved) | | 7: | 0: | 1: | XXh |
| to | | | 6: | 0: | 1: | |
| DFh | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | | 1: | 0: | 1: | |
| | | | 0: | 0: | 1: | |

### 7.2.105 E0h Port Direction (PortDir)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| E0h | PortDir | | 7: | 0: | 1: | 00h |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: PortDir1 | 0: Input | 1: Output | |
| | | R/W | 0: PortDir0 | 0: Input | 1: Output | |

This register sets the input/output direction of each bit for general-purpose IO ports. The ports can be set for input or output modes bitwise. When reset, the direction of each bit is set for input by default.
This register is effective even during snooze.



0: Input mode
1: Output mode

**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

**Bit 2** **Reserved**

**Bit 1** **PortDir1**
Sets the input/output direction of Port1.

**Bit 0** **PortDir0**
Sets the input/output direction of Port0.

### 7.2.106 E1h Port Data (PortData)

| Address | Register Name | R/W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|
| E1h | PortData | | 7: | 0: | 1: | XXh |
| | | | 6: | 0: | 1: | |
| | | | 5: | 0: | 1: | |
| | | | 4: | 0: | 1: | |
| | | | 3: | 0: | 1: | |
| | | | 2: | 0: | 1: | |
| | | R/W | 1: PortData1 | Port1 Data | | |
| | | R/W | 0: PortData0 | Port0 Data | | |

This register sets or acquires the status of general-purpose I/O ports. Writes to bits set for input mode are ignored.
This register is effective even during snooze.

**Bit 7** **Reserved**

**Bit 6** **Reserved**

**Bit 5** **Reserved**

**Bit 4** **Reserved**

**Bit 3** **Reserved**

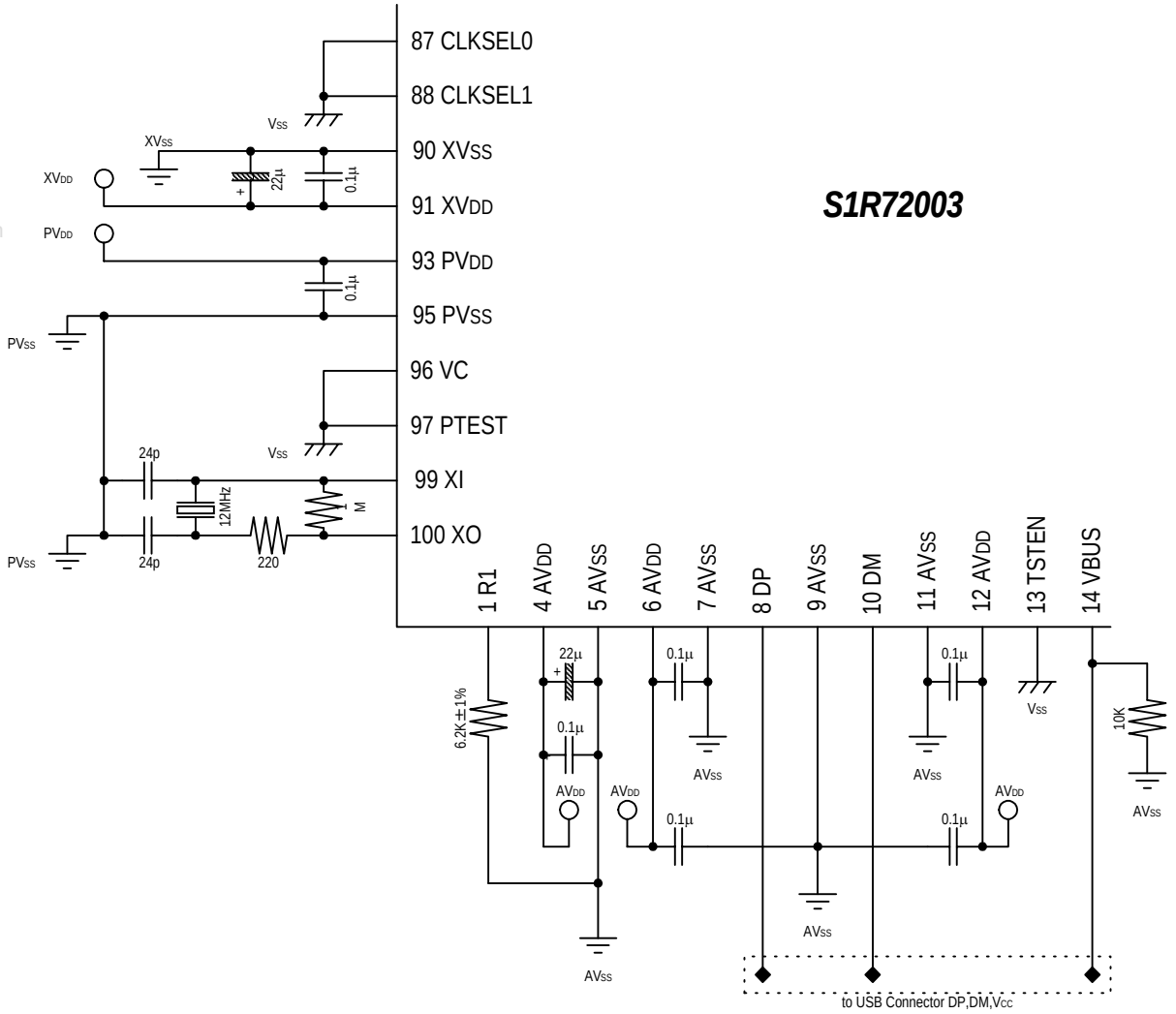
**Bit 2** **Reserved**

**Bit 1** **PortData1**
Sets or acquires the status of Port1.

**Bit 0** **PortData0**
Sets or acquires the status of Port0.

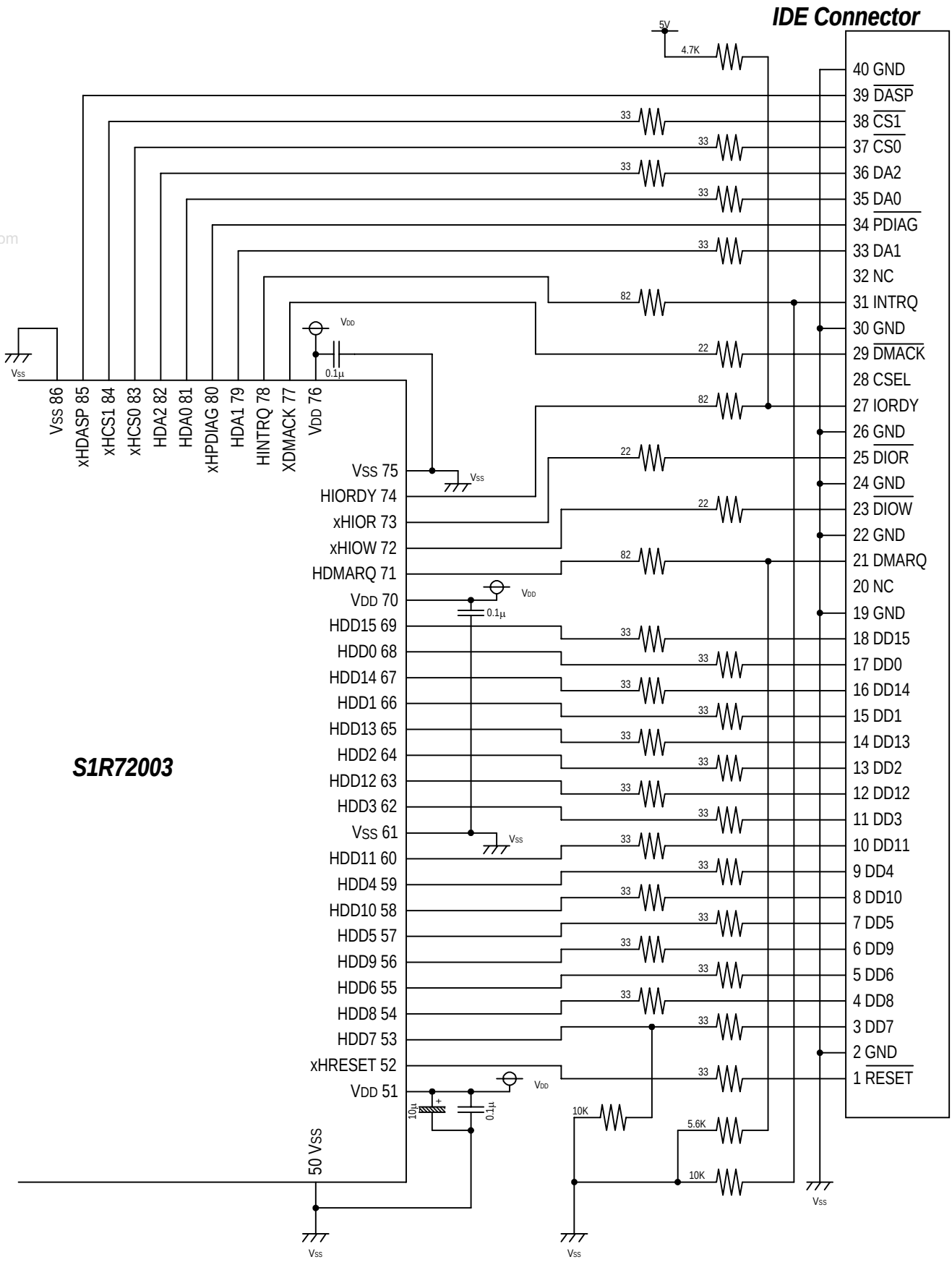
# 8. TYPICAL CONNECTIONS

## 8.1 Example of Connecting USB Interface and Other Pins

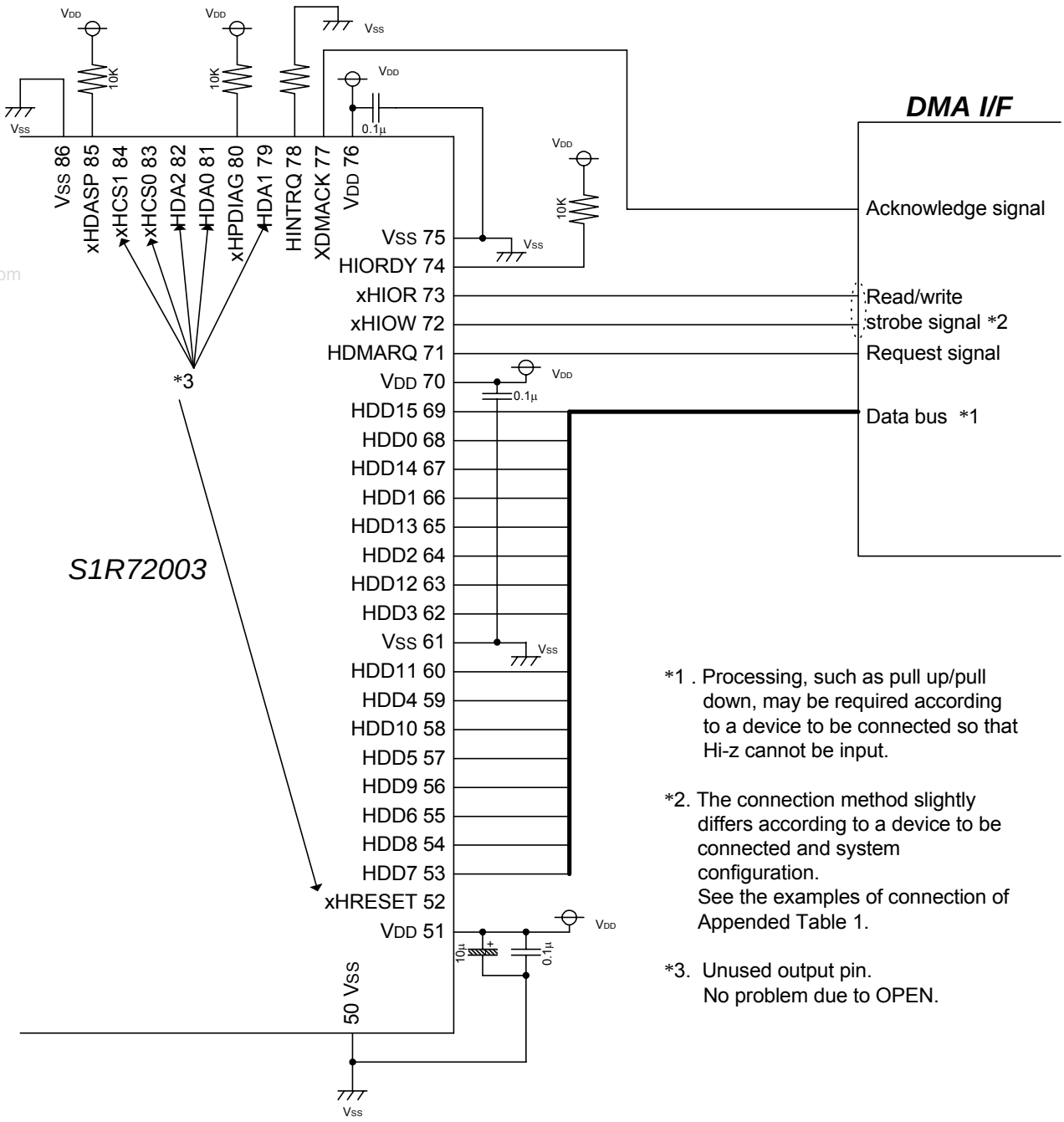



- Use 45 Ω wiring impedance for the DP/DM lines. (45 Ω between DP and GND, 45 Ω between DM and GND, and 90 Ω between DP and DM)
- Make sure the DP/DM lines have the same in lengths and are wired over the shortest possible distance.
- Connect R1 (pin 1) to AVSS (pin 5) via a 6 kΩ ± 5% resistor. Make sure the R1 pin and the resistor are wired over the shortest possible distance.
- For XI (pin 99) and XO (pin 100), refer to the example connection for the crystal resonator. (Shown above is an example for 12 MHz oscillation; CLKSEL0,1 = "00")
- Connect XVDD (pin 91) to XVSS (pin 90) by inserting a 0.1 µF (ceramic capacitor) and a 10 to 22 µF (tantalum or aluminum electrolytic capacitor) on the XVDD side.
- Connect PVDD (pin 93) to PVSS (pin 95) by inserting a 0.1 µF (ceramic capacitor) on the PVDD side. Also, connect GND for XI and XO in the vicinity of PVSS to stabilize crystal oscillator circuit operation.
- Connect AVDD on pin 4 to AVSS on pin 5 by inserting a 0.1 µF (ceramic capacitor) and a 10 to 22 µF (tantalum or aluminum electrolytic capacitor) on the AVDD side. Connect AVDD on pins 6 and 12 to AVSS on pins 7 and 11 by inserting a 0.1 µF (ceramic capacitor) on the AVDD side, respectively. Connect AVDD on pins 6 and 12 to AVSS on pin 9 by inserting a 0.1 µF (ceramic capacitor) on the AVDD side. Make sure the capacitors inserted for pins 6, 7, 9, 11, and 12 are positioned symmetrically, centering on pin 9.

## 8.2 Example of Connecting IDE Interface and Other Pins



## 8.3 Examples of Connection of IDE I/F Pins (When General-Purpose DMA is Used)

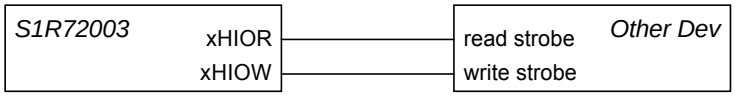
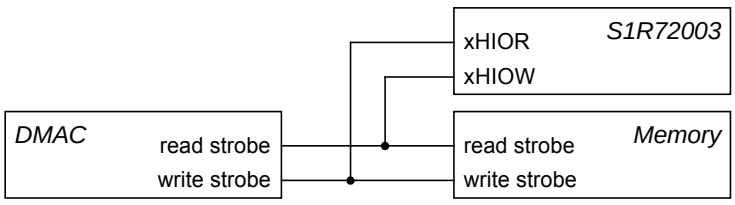



*1 . Processing, such as pull up/pull down, may be required according to a device to be connected so that Hi-z cannot be input.

*2. The connection method slightly differs according to a device to be connected and system configuration.
See the examples of connection of Appended Table 1.

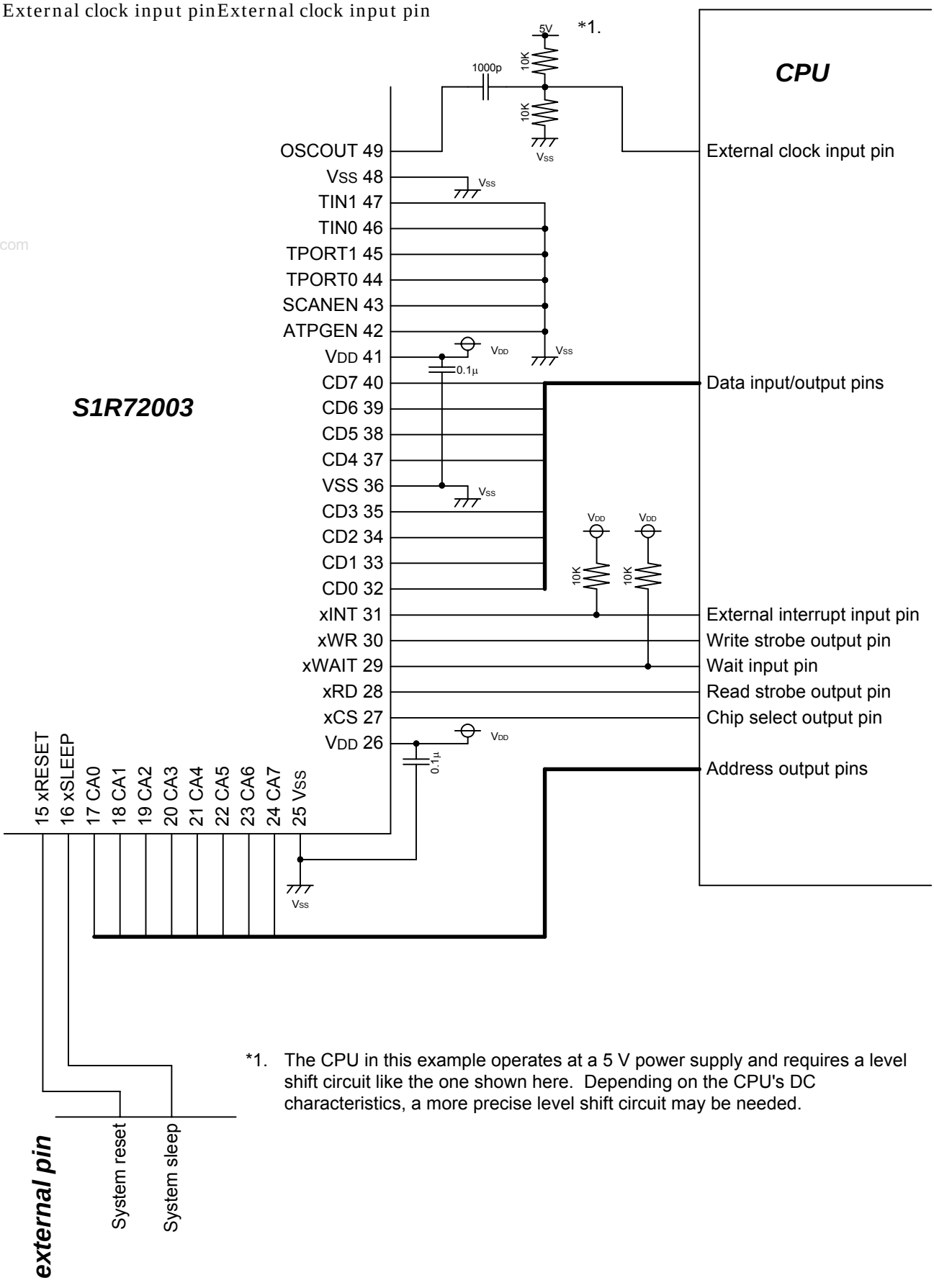
*3. Unused output pin.
No problem due to OPEN.

Appended Table 1

| | Examples of connection |
|---|---|
| When DMA transfer is performed between S1R72003 and the other device |  |
| When DMA transfer is performed between S1R72003 and the memory using an external DMAC (master) |  |

## 8.4 Example of Connecting CPU Interface and Other Pins

External clock input pinExternal clock input pin




*1. The CPU in this example operates at a 5 V power supply and requires a level shift circuit like the one shown here. Depending on the CPU's DC characteristics, a more precise level shift circuit may be needed.

# 9. ELECTRICAL CHARACTERISTICS

## 9.1 Absolute Maximum Ratings

($V_{SS}$ = 0 V)

| Parameter | Symbol | Rated Value | Unit |
|---|---|---|---|
| Power Supply Voltage | $V_{DD}$ | -0.3 to +4.0 | V |
| Input Voltage | $V_I$ | -0.3 to $V_{DD}$ + 0.5*1 | V |
| Output Voltage | $V_O$ | -0.3 to $V_{DD}$ + 0.5*1 | V |
| Output Current per Pin | $I_{OUT}$ | ±30 | mA |
| Power Supply Current | $I_{DDL}$ | 300 | mA |
| Storage Temperature | $T_{stg}$ | -65 to +150 | °C |



*1: For the pins shown in Appended Table 1, the specification value can be used from -0.3 to 7.0 V.

Appended Table 1

| Pin Name |
|---|
| xRESET, xSLEEP, CA[7:0], CD[7:0], xCS, xRD, xWR, HDD[15:0], xHIOR xHIOW, HDMARQ, xHDMACK, HIORDY, HINTRQ, xHDASP, xHDIAG, VBUS |

## 9.2 Recommended Operating Conditions

| Parameter | Symbol | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| Power Supply Voltage | $V_{DD}$ | 3.0 | 3.3 | 3.6 | V |
| Input Voltage | $V_I$ | $V_{SS}$ | — | $V_{DD}$ *1 | V |
| Ambient Temperature | Ta | 0 | 25 | 70 | °C |

*1: For the pins shown in Appended Table 1, the specification value can be used up to 5.25 V.
However, for the pins shown in Appended Table 2, the pull up exceeding $V_{DD}$ can be used.

Appended Table 2

| Pin Name |
|---|
| CD[7:0], HDD[15:0], xHIOR, xHIOW, HDMARQ, xHDMACK |

## 9.3 DC Characteristics

Input characteristics in DC state (under recommended operating conditions)

| Parameter | Symbol | Test Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Power Supply Source Current | | | | | | |
| Power Supply Current *1 | $I_{DDH}$ | $V_{DD}$ = 3.3 V | — | — | 150 | mA |
| Quiescent Current | | | | | | |
| Power Supply Current | $I_{DDS}$ | $V_{IN}$ = $V_{DD}$ or $V_{SS}$<br>$V_{DD}$ = 3.6 V | — | — | 90 | μA |
| Input Leakage | | | | | | |
| Input Leakage Current | $I_L$ | $V_{IN}$ = $V_{DD}$ or $V_{SS}$<br>$V_{DD}$ = 3.6 V | -1 | — | 1 | μA |
| Input Characteristics (CMOS) | Pin names: | TSTEN, xSLEEP, CA7..CA0, CD7..CD0, xCS, xRD<br>xWR, HDD15..HDD0, xHIOR, xHIOW, HDMARQ, xHDMACK | | | | |
| High Level Input Voltage | $V_{IH1}$ | $V_{DD}$ = 3.6 V | 2.0 | — | — | V |
| Low Level Input Voltage | $V_{IL1}$ | $V_{DD}$ = 3.0 V | — | — | 0.8 | V |
| Input Characteristics (TTL) | Pin names: | SCANEN, ATPGEN, TIN1, TIN0, CLKSEL1, CLKSEL0 | | | | |
| High Level Input Voltage | $V_{IH2}$ | $V_{DD}$ = 3.6 V | 2.0 | — | — | V |
| Low Level Input Voltage | $V_{IL2}$ | $V_{DD}$ = 3.0 V | — | — | 0.8 | V |
| Schmitt Input Characteristics(TTL) | Pin names: | xRESET, HIORDY, HINTRQ, xHDASP, xHPDIAG, TPORT1, TPORT0, VBUS | | | | |
| High Level Trigger Voltage | $V_{T1+}$ | $V_{DD}$ = 3.6 V | 1.1 | — | 2.4 | V |
| Low Level Trigger Voltage | $V_{T1-}$ | $V_{DD}$ = 3.0 V | 0.6 | — | 1.8 | V |
| Hysteresis Voltage | $\Delta V$ | $V_{DD}$ = 3.0 V | 0.1 | — | — | V |
| Schmitt Input Characteristics (USB: FS) | Pin names: | DP, DM | | | | |
| High Level Trigger Voltage | $V_{T+(USB)}$ | $V_{DD}$ = 3.6 V | 1.1 | — | 1.8 | V |
| Low Level Trigger Voltage | $V_{T-(USB)}$ | $V_{DD}$ = 3.0 V | 1.0 | — | 1.5 | V |
| Hysteresis Voltage | $\Delta V_{(USB)}$ | $V_{DD}$ = 3.0 V | 0.1 | — | — | V |
| Input Characteristics (USB: FS differential input) | Pin names: | DP and DM in pairs | | | | |
| Differential Input Sensitivity | $V_{DS(USB)}$ | $V_{DD}$ = 3.0 V<br>Differential input voltage<br>0.8 V to 2.5 V | — | — | 0.2 | V |
| Input Pulldown Characteristics | Pin names: | TSTEN | | | | |
| Pulldown Resistance Value | $R_{PLD1}$ | $V_{DD}$ = 3.3 V<br>$V_{IH}$ = $V_{DD}$ | 20 | 50 | 100 | kΩ |
| Input Pulldown Characteristics | Pin names: | SCANEN, ATPGEN, TIN1, TIN0 | | | | |
| Pulldown Resistance Value | $R_{PLD2}$ | $V_{DD}$ = 3.3 V<br>$V_{IH}$ = $V_{DD}$ | 40 | 100 | 200 | kΩ |
| Input Pullup Characteristics | Pin names: | xSLEEP, CA7..CA0, CD7..CD0, xCS, xRD, xWR, | | | | |
| Pullup Resistance Value | $R_{PLU2}$ | $V_{DD}$ = 3.0 V<br>$V_{IH}$ = $V_{SS}$ | 40 | 100 | 200 | kΩ |

Output characteristics in DC state (under recommended operating conditions)

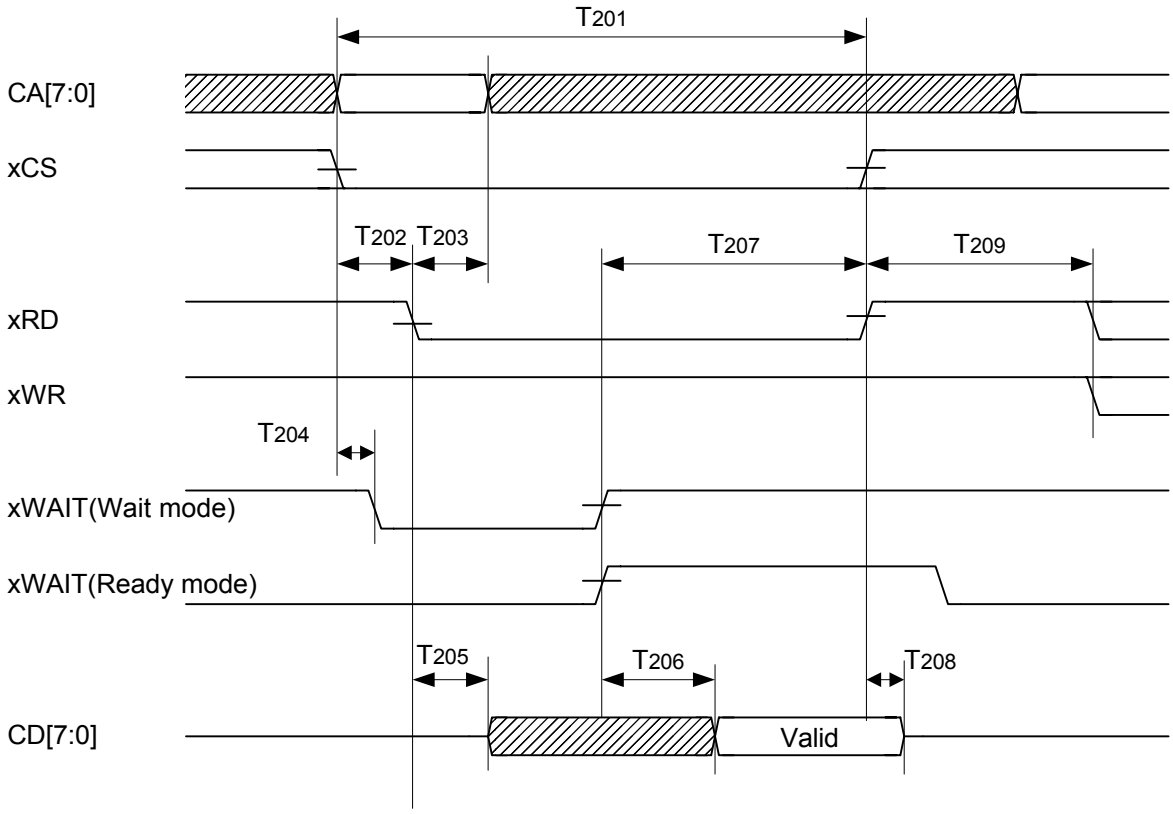
| Parameter | Symbol | Test Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Output Characteristics | Pin names: | TPORT1, TPORT0 | | | | |
| High Level Output Voltage | $V_{OH1}$ | $V_{DD}$ = 3.0 V<br>$I_{OH}$ = -2 mA | $V_{DD}$ -0.4 | — | — | V |
| Low Level Output Voltage | $V_{OL1}$ | $V_{DD}$ = 3.0 V<br>$I_{OL}$ = 2 mA | — | — | 0.4 | V |
| Output Characteristics | Pin names: | CD7..CD0, xWAIT, xINT, HDD15..HDD0, HDA2..HDA0, xHCS1, xHCS0, xHIOR xHIOW, HDMARQ, xHDMACK, xHRESET | | | | |
| High Level Output Voltage | $V_{OH2}$ | $V_{DD}$ = 3.0 V<br>$I_{OH}$ = -6 mA | $V_{DD}$ –0.4 | — | — | V |
| Low Level Output Voltage | $V_{OL2}$ | $V_{DD}$ = 3.0 V<br>$I_{OL}$ = 6 mA | — | — | 0.4 | V |
| Output Characteristics (USB: FS) | Pin names: | DP, DM | | | | |
| High Level Output Voltage | $V_{OH(USB)}$ | $V_{DD}$ = 3.0 V | 2.8 | — | — | V |
| Low Level Output Voltage | $V_{OL(USB)}$ | $V_{DD}$ = 3.6 V | — | — | 0.3 | V |
| Output Characteristics (USB: HS) | Pin names: | DP, DM | | | | |
| High Level Output Voltage | $V_{HSOH(USB)}$ | $V_{DD}$ = 3.0 V | 360 | — | — | mV |
| Low Level Output Voltage | $V_{HSOL(USB)}$ | $V_{DD}$ = 3.6 V | — | — | 10.0 | mV |
| Output Characteristics | Pin names: | All output pins | | | | |
| Off State Leakage Current | $I_{OZ}$ | $V_{DD}$ = 3.6 V<br>$V_{OH}$ = $V_{DD}$<br>$V_{OL}$ = $V_{SS}$ | -1 | — | 1 | μA |
| Pin Capacitance | Pin names: | All input pins | | | | |
| Input Pin Capacitance | $C_I$ | f = 1 MHz<br>$V_{DD}$ = $V_{SS}$ | — | — | 10 | pF |
| Pin Capacitance | Pin names: | All output pins | | | | |
| Output Pin Capacitance | $C_O$ | f = 1 MHz<br>$V_{DD}$ = $V_{SS}$ | — | — | 10 | pF |
| Pin Capacitance | Pin names: | All input/output pins | | | | |
| Input/Output Pin Capacitance | $C_{IO}$ | f = 1 MHz<br>$V_{DD}$ = $V_{SS}$ | — | — | 10 | pF |

*1 : It is current at the time of operation by recommendation operation conditions (TYP.:$V_{DD}$=3.3V and Ta = 25 °C).

## 9.4 AC Characteristics

### 9.4.1 CPU I/F Access Timing

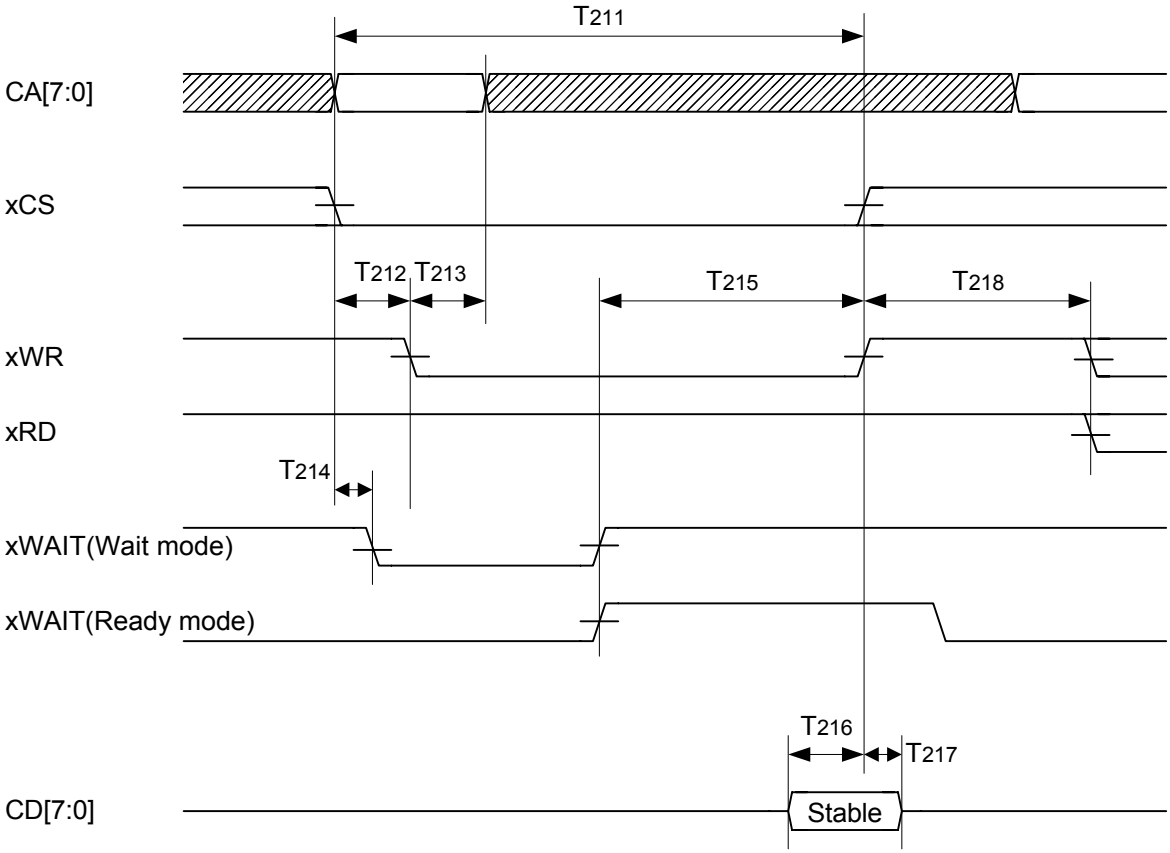
#### 9.4.1.1 Read Timing




| Symbol | Description | Min. | Max. | Unit |
|---|---|---|---|---|
| T201 | One read cycle time | 100 | — | ns |
| T202 | CA, xCS setup time relative to xRD ↓ | 2 | — | ns |
| T203 | CA hold time relative to xRD ↓ | 2 | — | ns |
| T204 | xWAIT assert time relative to xCS ↓ | — | 10 | ns |
| T205 | CD output delay time relative to xRD ↓ | — | 20 | ns |
| T206 | CD output delay time relative to xWAIT ↑ | — | 5 | ns |
| T207 | xRD negate time relative to xWAIT ↑ | 10 | — | ns |
| * T208 | Read data hold time relative to xRD ↑ or xCS ↑ | 2 | — | ns |
| T209 | Hold time from xRD ↑ to the next xRD ↓ or xWR ↓ | 20 | — | ns |

* T208 is the data hold time relative to the rising edge of xRD or xCS, whichever goes high first.

### 9.4.1.2 Write Timing

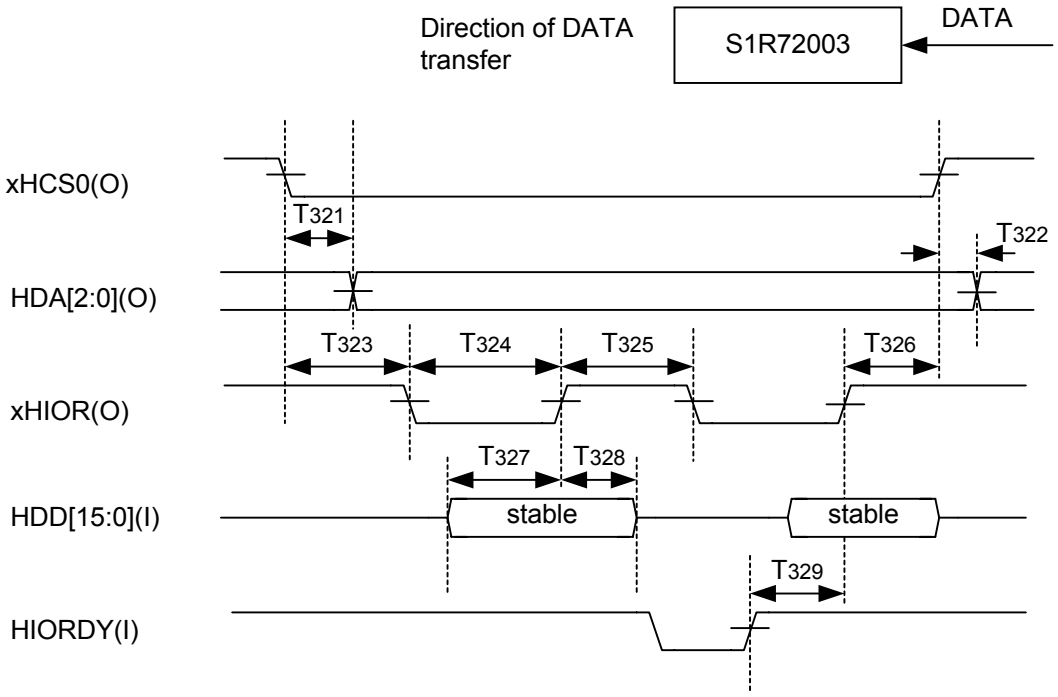



| Symbol | Description | Min. | Max. | Unit |
|---|---|---|---|---|
| T211 | One write cycle time | 100 | — | ns |
| T212 | CA, xCS setup time relative to xWR ↓ | 2 | — | ns |
| T213 | CA hold time relative to xWR ↓ | 2 | — | ns |
| T214 | xWAIT assert time relative to xCS ↓ | — | 10 | ns |
| T215 | xWAIT assert time relative to xCS ↓ | 10 | — | ns |
| * T216 | Write data setup time relative to xWR ↑or xCS ↑ | 10 | — | ns |
| * T217 | Write data hold time relative to xWR ↑ or xCS ↑ | 5 | — | ns |
| T218 | Hold time from xWR ↑ to the next xRD ↓ or xWR ↓ | 20 | — | ns |

* T216 and T217 are the data setup and hold times relative to the rising edge of xWR or xCS, whichever goes high first.

### 9.4.2 IDE I/F Timing

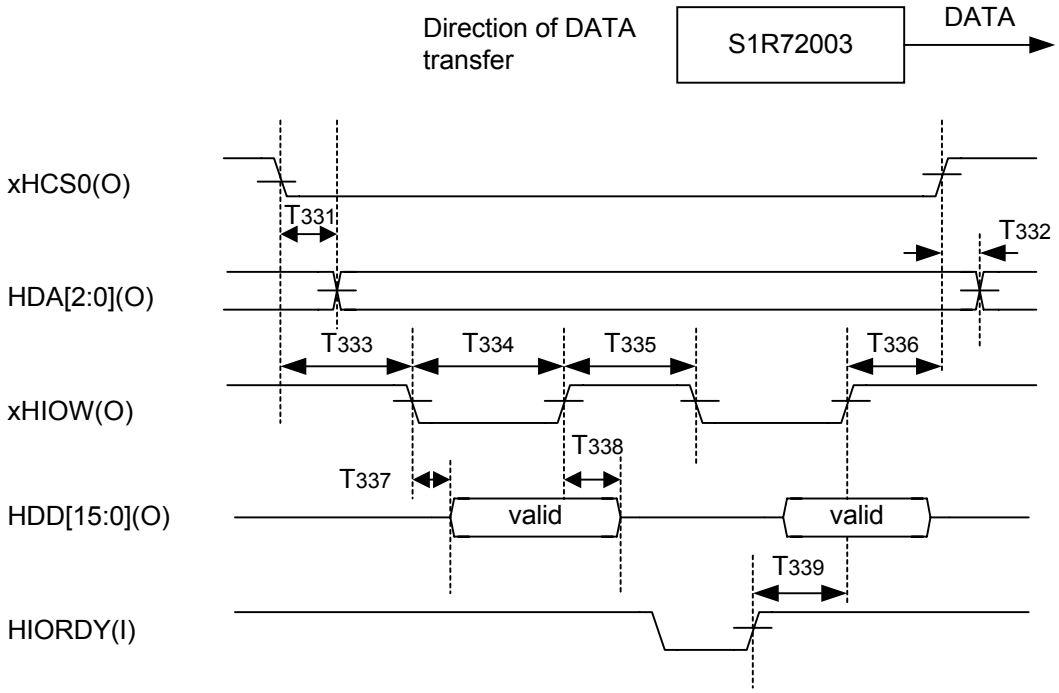
#### 9.4.2.1 PIO READ Timing




| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{321}$ | xHCS0 ↓ → HDA<br>HDA output delay time | — | 0 | — | ns |
| $T_{322}$ | xHCS0 ↑ → HDA<br>HDA hold time | — | 0 | — | ns |
| $T_{323}$ | xHCS0 ↓ → xHIOR ↓<br>xHCS0 setup time | 60 | — | — | ns |
| $T_{324}$ | xHIOR ↓ → xHIOR ↑<br>xHIOR assert pulse width | — | (AP + 5) × 16.6 *1 | — | ns |
| $T_{325}$ | xHIOR ↑ → xHIOR ↓<br>xHIOR negate pulse width | — | (NP + 3) × 16.6 *1 | — | ns |
| $T_{326}$ | xHIOR ↑ → xHCS0 ↑<br>xHCS0 hold time | 20 | — | — | ns |
| $T_{327}$ | HDD → xHIOR ↑<br>Data setup time | 10 | — | — | ns |
| $T_{328}$ | xHIOR ↑ → HDD<br>Data hold time | 0 | — | — | ns |
| $T_{329}$ | HIORDY assert → xHIOR ↑<br>xHIOR output delay time | — | — | 40 | ns |

*1: For more information, refer to the register description, “IDE Transfer Mode.”

### 9.4.2.2 PIO Write Timing

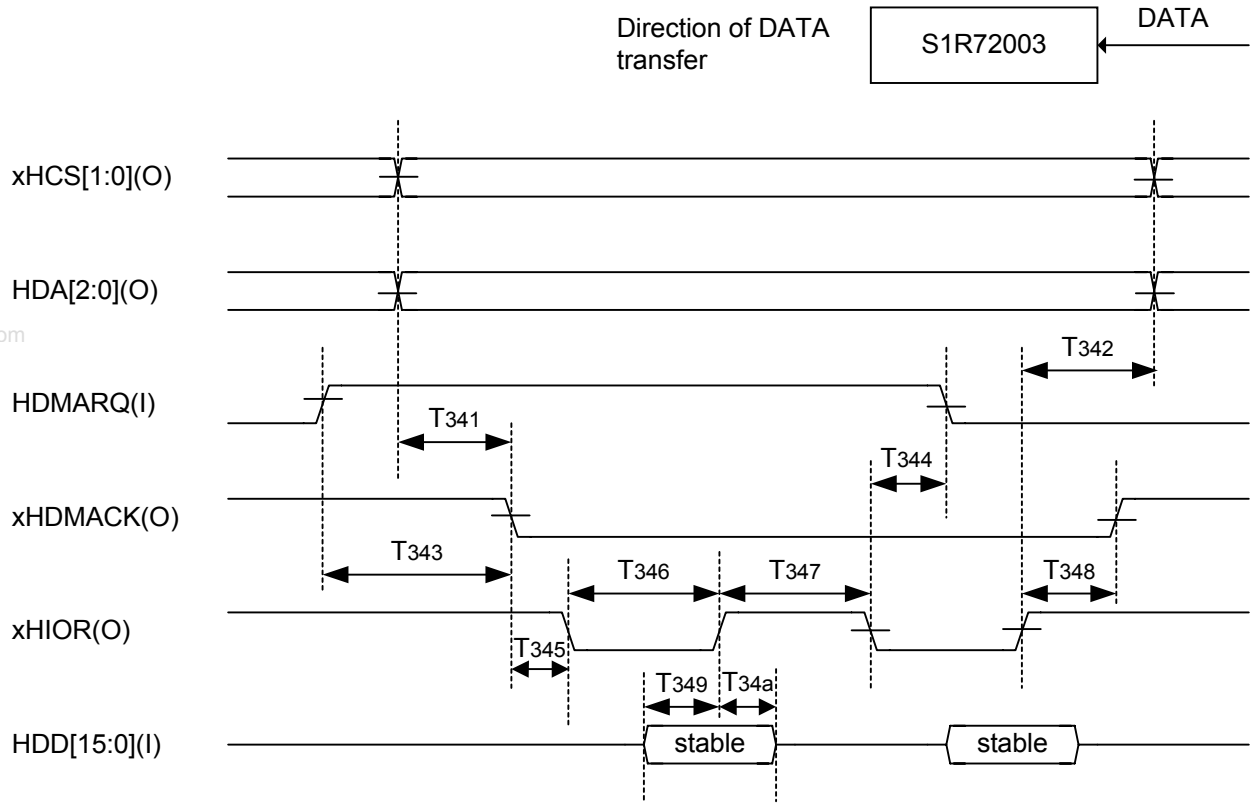





| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{331}$ | xHCS0 ↓ → HDA<br>HDA output delay time | — | 0 | — | ns |
| $T_{332}$ | xHCS0 ↑ → HDA<br>HDA hold time | — | 0 | — | ns |
| $T_{333}$ | xHCS0 ↓ → xHIOW ↓<br>xHCS0 setup time | 60 | — | — | ns |
| $T_{334}$ | xHIOW ↓ → xHIOW ↑<br>xHIOW assert pulse width | — | (AP + 5) × 16.6 *1 | — | ns |
| $T_{335}$ | xHIOW ↑ → xHIOW ↓<br>xHIOW negate pulse width | — | (NP + 3) × 16.6 *1 | — | ns |
| $T_{336}$ | xHIOW ↑ → xHCS0 ↑<br>xHCS0 hold time | 20 | — | — | ns |
| $T_{337}$ | xHIOW ↓ → HDD<br>Data output delay time | 0 | — | 20 | ns |
| $T_{338}$ | xHIOW ↑ → HDD<br>Data bus negate time | 40 | — | 60 | ns |
| $T_{339}$ | HIORDY assert → xHIOW ↑<br>xHIOW output delay time | — | — | 40 | ns |

*1: For more information, refer to the register description, “IDE Transfer Mode.”

### 9.4.2.3 DMA Read Timing

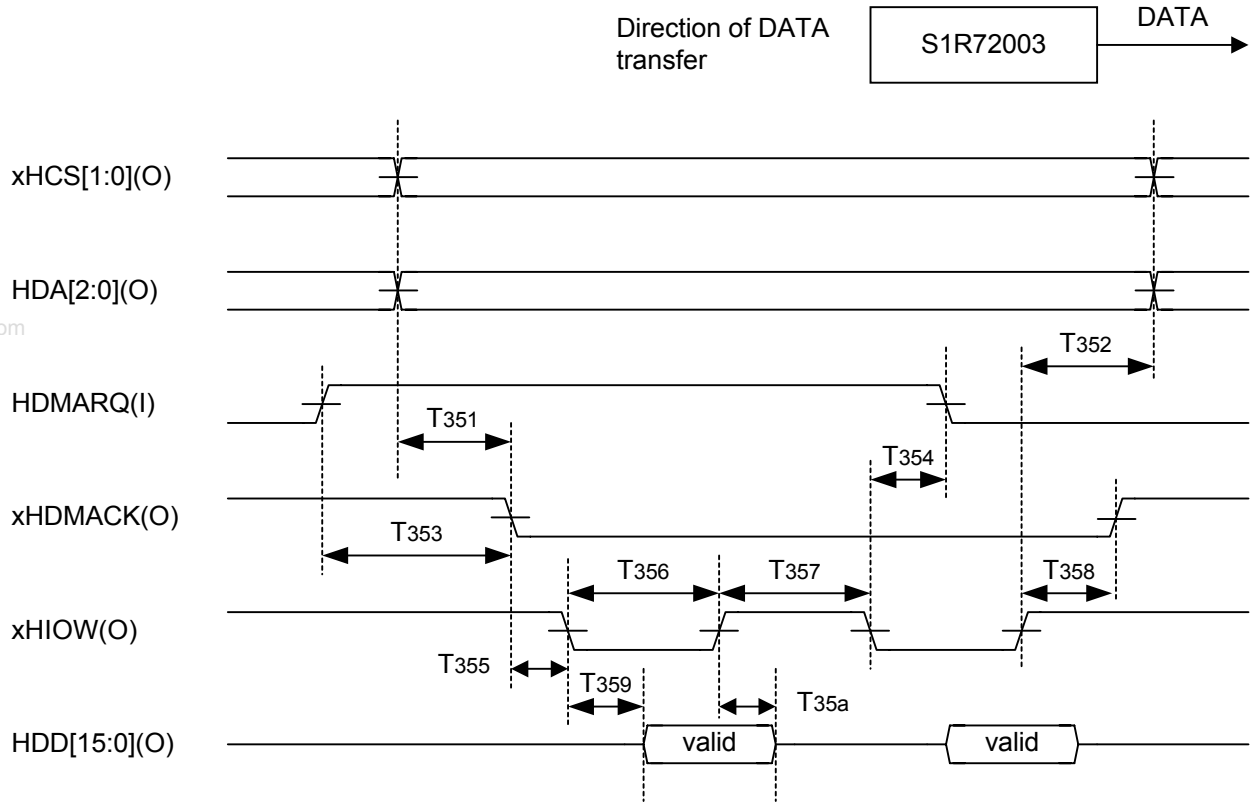





| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{341}$ | xHCS ↓, HDA → xHDMACK ↓<br>Address setup time | 60 | — | — | ns |
| $T_{342}$ | xHIOR ↑ → xHCS ↑, HDA<br>Address hold time | 25 | — | — | ns |
| $T_{343}$ | HDMARQ ↑ → xHDMACK ↓<br>xHDMACK response time | 0 | — | — | ns |
| $T_{344}$ | xHIOR ↓ → HDMARQ negate<br>HDMARQ hold time | 0 | — | — | ns |
| $T_{345}$ | xHDMACK ↓ → xHIOR ↓<br>xHDMACK setup time | 0 | — | — | ns |
| $T_{346}$ | xHIOR ↓ → xHIOR ↑<br>xHIOR assert pulse width | — | (AP + 5) × 16.6 *1 | — | ns |
| $T_{347}$ | xHIOR ↑ → xHIOR ↓<br>xHIOR negate pulse width | — | (NP + 3) × 16.6 *1 | — | ns |
| $T_{348}$ | xHIOR ↑ → xHDMACK ↑<br>xHDMACK hold time | 20 | — | — | ns |
| $T_{349}$ | HDD → xHIOR ↑<br>Data setup time | 10 | — | — | ns |
| $T_{34a}$ | xHIOR ↑ → HDD<br>Data bus hold time | 0 | — | — | ns |

*1: For more information, refer to the register description, “IDE Transfer Mode.”

### 9.4.2.4 DMA Write Timing

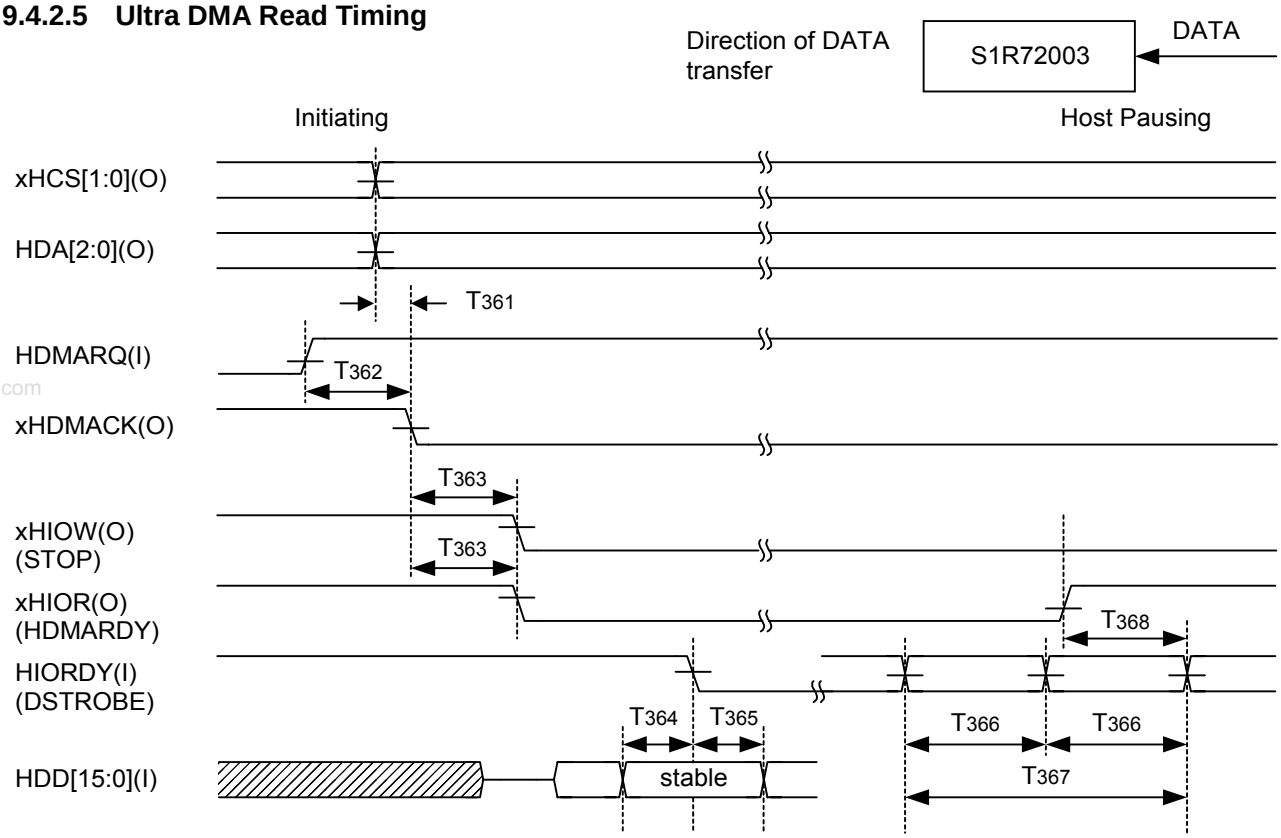





| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{351}$ | xHCS ↓, HDA → xHDMACK ↓<br>Address setup time | 60 | — | — | ns |
| $T_{352}$ | xHIOW ↑ → xHCS ↑, HDA<br>Address hold time | 20 | — | — | ns |
| $T_{353}$ | HDMARQ ↑→ xHDMACK ↓<br>xHDMACK response time | 0 | — | — | ns |
| $T_{354}$ | xHIOW ↓ → HDMARQ negate<br>HDMARQ hold time | 0 | — | — | ns |
| $T_{355}$ | xHDMACK ↓ → xHIOW ↓<br>xHDMACK setup time | 0 | — | — | ns |
| $T_{356}$ | xHIOW ↓ → xHIOW ↑<br>xHIOW assert pulse width | — | (AP + 5) × 16.6 *1 | — | ns |
| $T_{357}$ | xHIOW ↑ → xHIOW ↓<br>xHIOW negate pulse width | — | (NP + 3) × 16.6 *1 | — | ns |
| $T_{358}$ | xHIOW ↑ → xHDMACK ↑<br>xHDMACK hold time | 20 | — | — | ns |
| $T_{359}$ | xHIOW ↓ → HDD<br>Data output delay time | 0 | — | 20 | ns |
| $T_{35a}$ | xHIOW ↑ → HDD<br>Data bus negate time | 20 | — | 40 | ns |

*1: For more information, refer to the register description, “IDE Transfer Mode.”

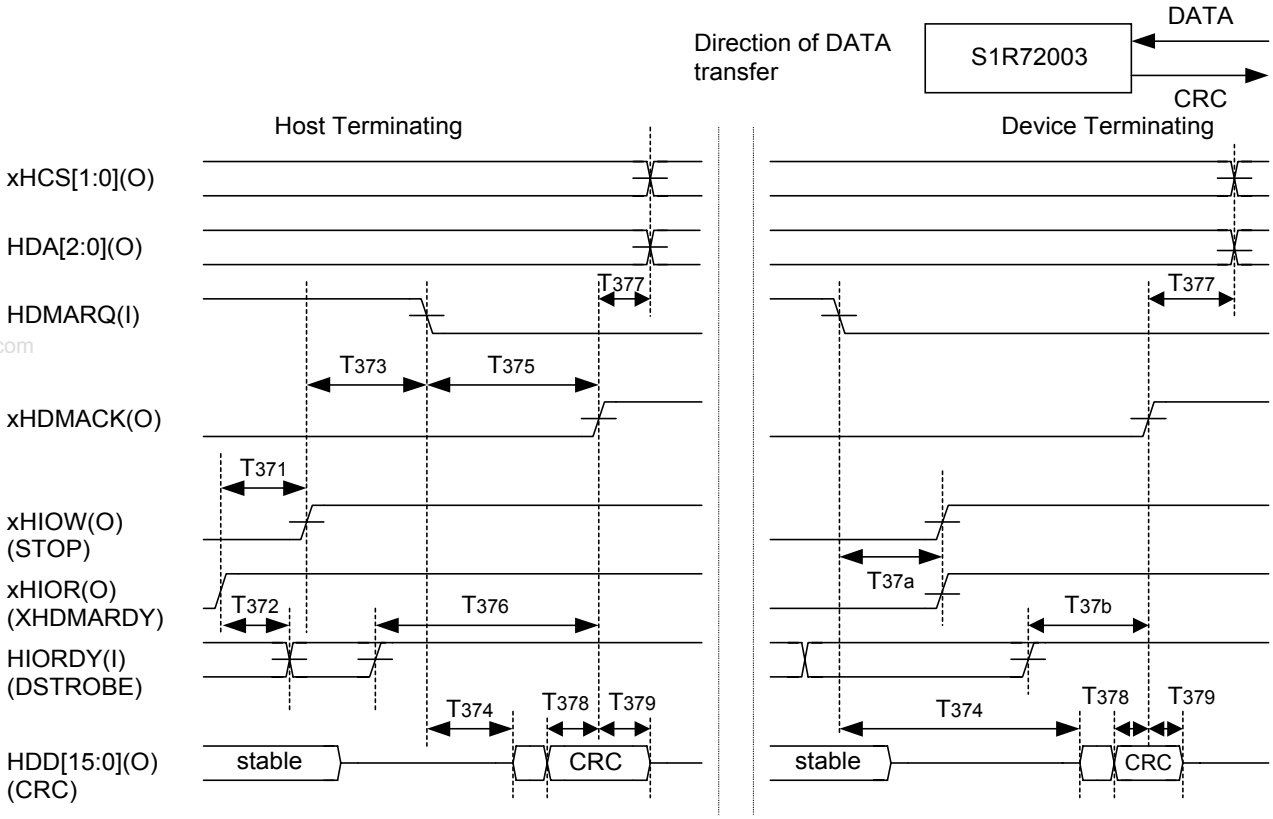
### 9.4.2.5 Ultra DMA Read Timing




| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{361}$ | xHcs ↓, HAD → xHDMACK ↓<br>Address setup time | 20 | — | — | ns |
| $T_{362}$ | HDMARQ ↑ → xHDMACK ↓<br>xHDMACK response time | 0 | — | — | ns |
| $T_{363}$ | xHDMACK ↓ → xHIOR(W) ↓<br>Envelope time | 20 | — | 55 | ns |
| $T_{364}$ | HDD → HIORDY<br>Data setup time | 6 | — | — | ns |
| $T_{365}$ | HIORDY → HDD<br>Data hold time | 6 | — | — | ns |
| $T_{366}$ | HIORDY → HIORDY<br>HIORDY cycle time | — | (cyc + 2) × 16.6 *1 | — | ns |
| $T_{367}$ | HIORDY → HIORDY<br>HIORDY cycle time × 2 | — | $T_{366} \times 2$ | — | ns |
| $T_{368}$ | XHIOR ↑ → HIORDY<br>Last STROBE time | — | — | 60 | ns |

*1: For more information, refer to the register description, “IDE Ultra-DMA Transfer Mode.”

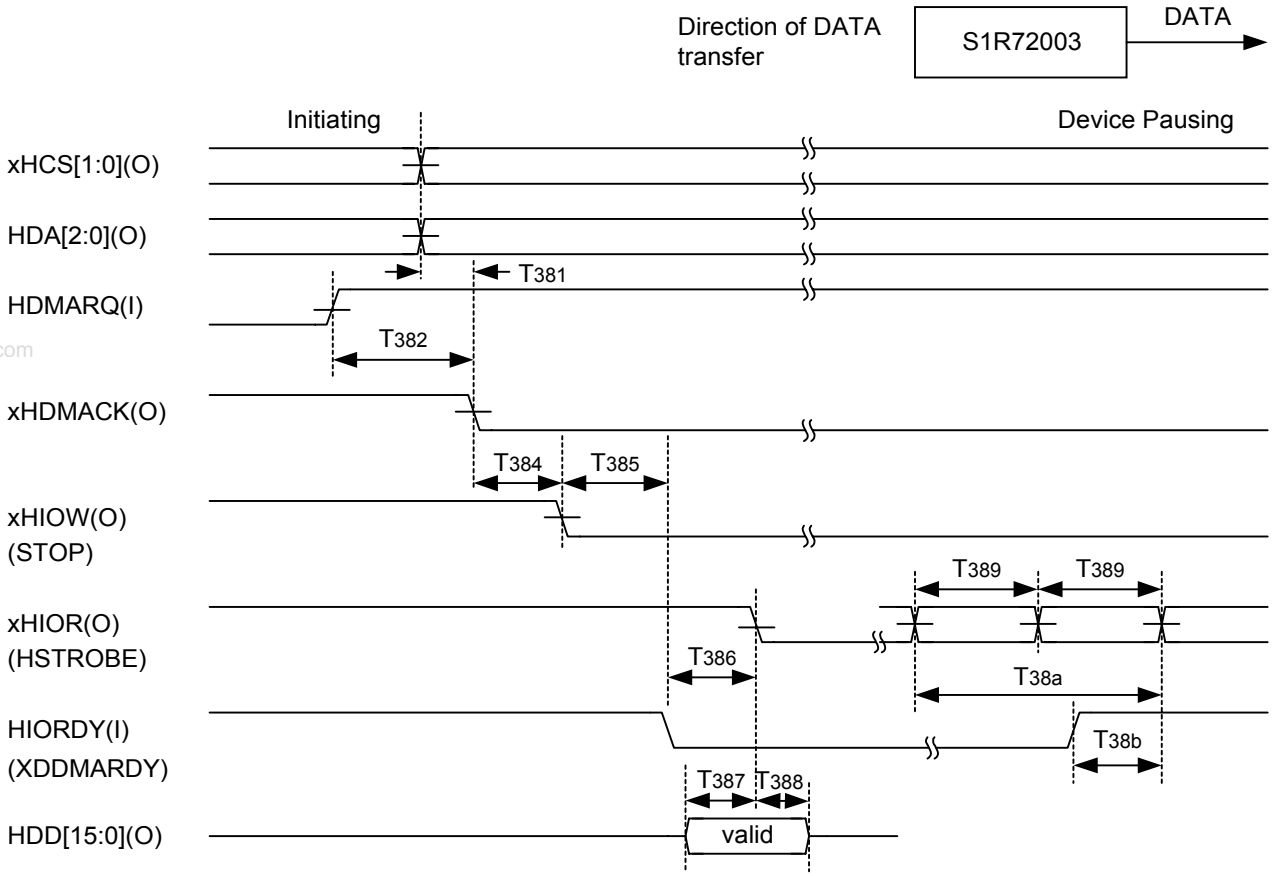
**Ultra DMA Read Timing (continued from the preceding page)**






| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{371}$ | xHIOR ↑ → xHIOW ↑<br>Time till STOP assert | 100 | — | — | ns |
| $T_{372}$ | xHIOR ↑ → HIORDY<br>Last STROBE time | — | — | 60 | ns |
| $T_{373}$ | xHIOW ↑ → HDMARQ ↓<br>Interlock time with limit | — | — | 100 | ns |
| $T_{374}$ | HDMARQ ↓ → HDD<br>Output delay time | 20 | — | — | ns |
| $T_{375}$ | HDMARQ ↓ → xHDMACK ↑<br>Minimum interlock time | 20 | — | — | ns |
| $T_{376}$ | HIORDY → xHDMACK ↑<br>Minimum interlock time | 20 | — | — | ns |
| $T_{377}$ | xHDMACK ↑ → xHCS0,1<br>xHCS0,1 hold time | 20 | — | — | ns |
| $T_{378}$ | HDD(CRC) → xHDMACK ↑<br>CRC data setup time | 6 | — | — | ns |
| $T_{379}$ | xHDMACK ↑ → HDD(CRC)<br>CRC data hold time | 6 | — | — | ns |
| $T_{37a}$ | HDMARQ ↓ → xHIOR ↑<br>Interlock time with limit | 0 | — | 100 | ns |
| $T_{37b}$ | HIORDY → xHDMACK ↑<br>Minimum interlock time | 20 | — | — | ns |

### 9.4.2.6 Ultra DMA Write Timing



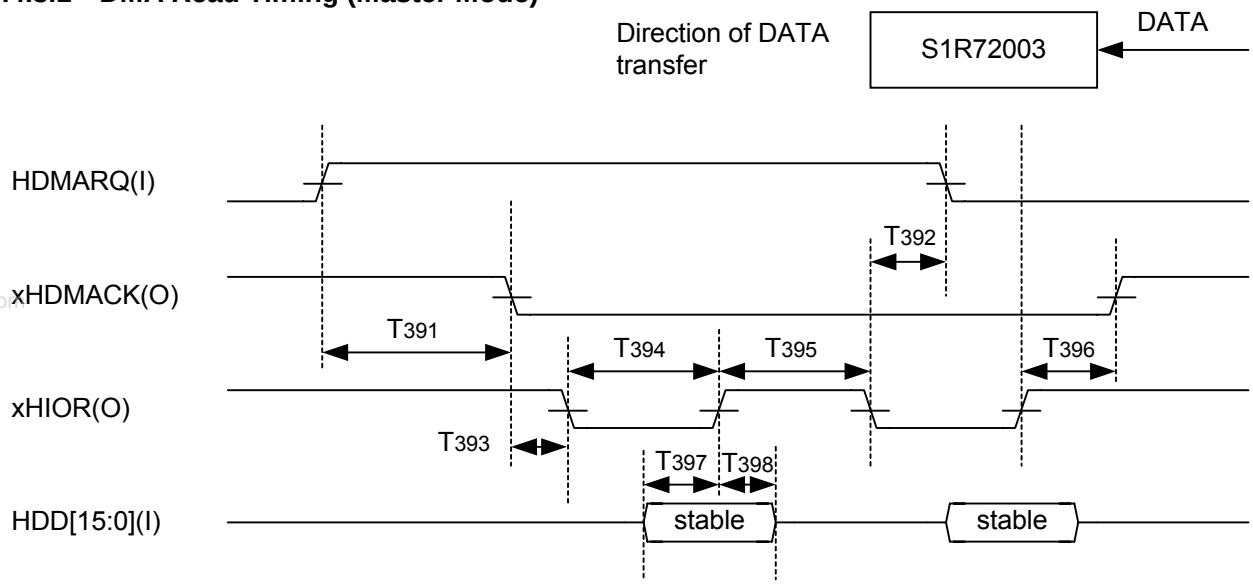



| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{381}$ | xHCS ↓, HDA → xHDMACK ↓<br>Address setup time | 20 | — | — | ns |
| $T_{382}$ | HDMARQ ↑ → xHDMACK ↓<br>xHDMACK response time | 0 | — | — | ns |
| $T_{384}$ | xHDMACK ↓ → xHIOW ↓<br>Envelope time | 20 | — | 40 | ns |
| $T_{385}$ | xHIOW ↓ → HIORDY ↓<br>Interlock time with limit | 0 | — | 100 | ns |
| $T_{386}$ | HIORDY ↓ → xHIOR ↓<br>Interlock time without limit | 0 | — | — | ns |
| $T_{387}$ | HDD → xHIOR ↓<br>Data setup time | 6 | — | — | ns |
| $T_{388}$ | xHIOR ↓ → HDD<br>Data hold time | 6 | — | — | ns |
| $T_{389}$ | xHIOR → xHIOR<br>xHIOR cycle time | — | (cyc + 2) × 16.6 *1 | — | ns |
| $T_{38a}$ | xHIOR → xHIOR<br>xHIOR cycle time × 2 | — | $T_{389}$ × 2 | — | ns |
| $T_{38b}$ | HIORDY ↑ → xHIOR<br>Last STROBE time | — | — | 60 | ns |

*1: For more information, refer to the register description, “IDE Ultra-DMA Transfer Mode.”

### 9.4.3 General-purpose Port I/F Timing

#### 9.4.3.1 DMA Read Timing (Master Mode)



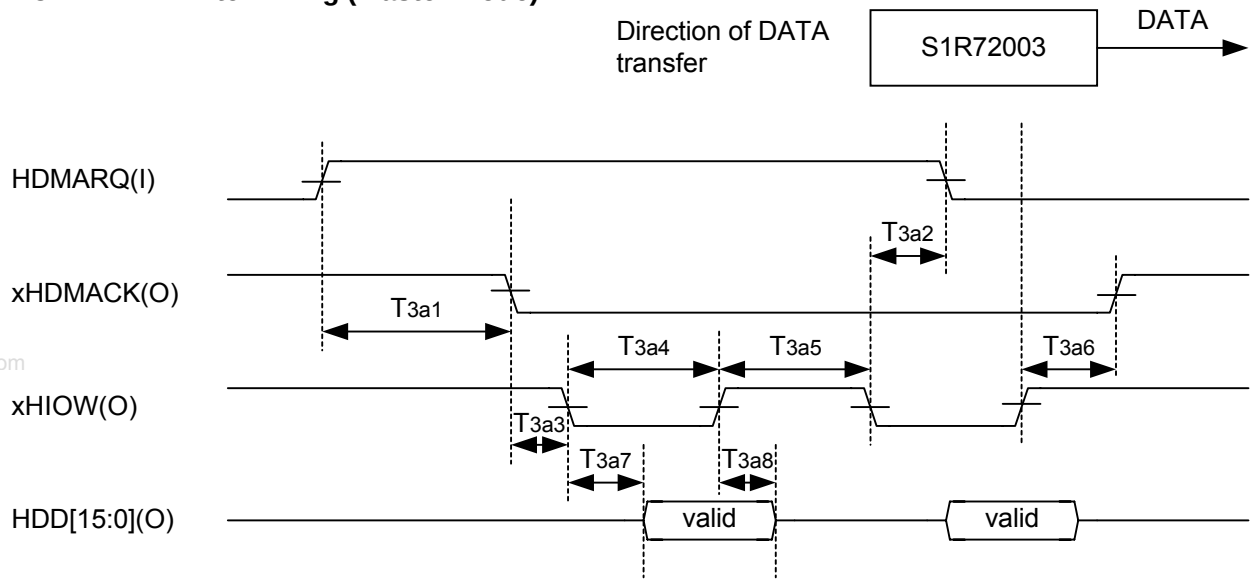



| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{391}$ | HDMARQ ↑ → xHDMACK ↓<br>xHDMACK response time | 0 | — | — | ns |
| $T_{392}$ | xHIOR ↓ → HDMARQ negate<br>HDMARQ hold time | 0 | — | — | ns |
| $T_{393}$ | xHDMACK ↓ → xHIOR ↓<br>xHDMACK setup time | 0 | — | — | ns |
| $T_{394}$ | xHIOR ↓ → xHIOR ↑<br>xHIOR assert pulse width | — | *1 | — | ns |
| $T_{395}$ | xHIOR ↑ → xHIOR ↓<br>xHIOR negate pulse width | — | *1 | — | ns |
| $T_{396}$ | xHIOR↑ → xHDMACK ↑<br>xHDMACK hold time | 20 | — | — | ns |
| $T_{397}$ | HDD → xHIOR ↑<br>Data setup time | 10 | — | — | ns |
| $T_{398}$ | xHIOR↑ → HDD<br>Data bus hold time | 0 | — | — | ns |

*1: According to register settings. For details, see the following table and register description “IDE Transfer Mode” or “IDE Config_0”.

| Symbol | IDE_Tmod | IDEConfig_0.NotIDE | *1 |
|---|---|---|---|
| $T_{394}$ | AP | 0 | (AP+5)*16.6 |
| | | 1 | (AP+2)*16.6 |
| $T_{395}$ | NP | 0 | (NP+3)*16.6 |
| | | 1 | (NP+2)*16.6 |

### 9.4.3.2 DMA Write Timing (Master Mode)



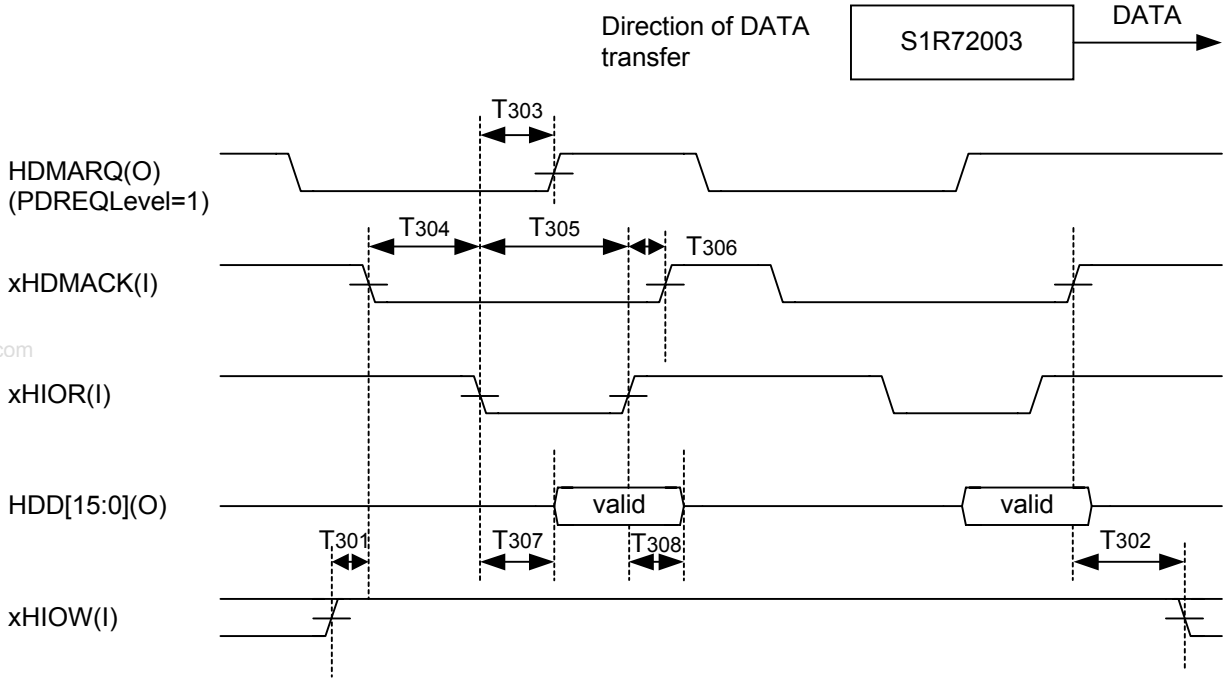



| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| T3a1 | HDMARQ ↑ → xHDMACK ↓<br>xHDMACK response time | 0 | — | — | ns |
| T3a2 | xHIOW ↓ → HDMARQ negate<br>HDMARQ hold time | 0 | — | — | ns |
| T3a3 | xHDMACK ↓ → xHIOW ↓<br>xHDMACK setup time | 0 | — | — | ns |
| T3a4 | xHIOW ↓ → xHIOW ↑<br>xHIOR assert pulse width | — | *1 | — | ns |
| T3a5 | xHIOW ↑ → xHIOW ↓<br>xHIOR negate pulse width | — | *1 | — | ns |
| T3a6 | xHIOW ↑ → xHDMACK ↑<br>xHDMACK hold time | 20 | — | — | ns |
| T3a7 | xHIOW ↓ → HDD<br>Data output delay time | 0 | — | 25 | ns |
| T3a8 | xHIOW ↑ → HDD<br>Data bus negate time | 6 | — | 40 | ns |

*1: According to register settings. For details, see the following table and register description “IDE Transfer Mode” or “IDE Config_0”.

| Symbol | IDE_Tmod | IDEConfig_0.NotIDE | *1 |
|---|---|---|---|
| T3a4 | AP | 0 | (AP+5)*16.6 |
| | | 1 | (AP+2)*16.6 |
| T3a5 | NP | 0 | (NP+3)*16.6 |
| | | 1 | (NP+2)*16.6 |

### 9.4.3.3 DMA Read Timing (Slave Mode)

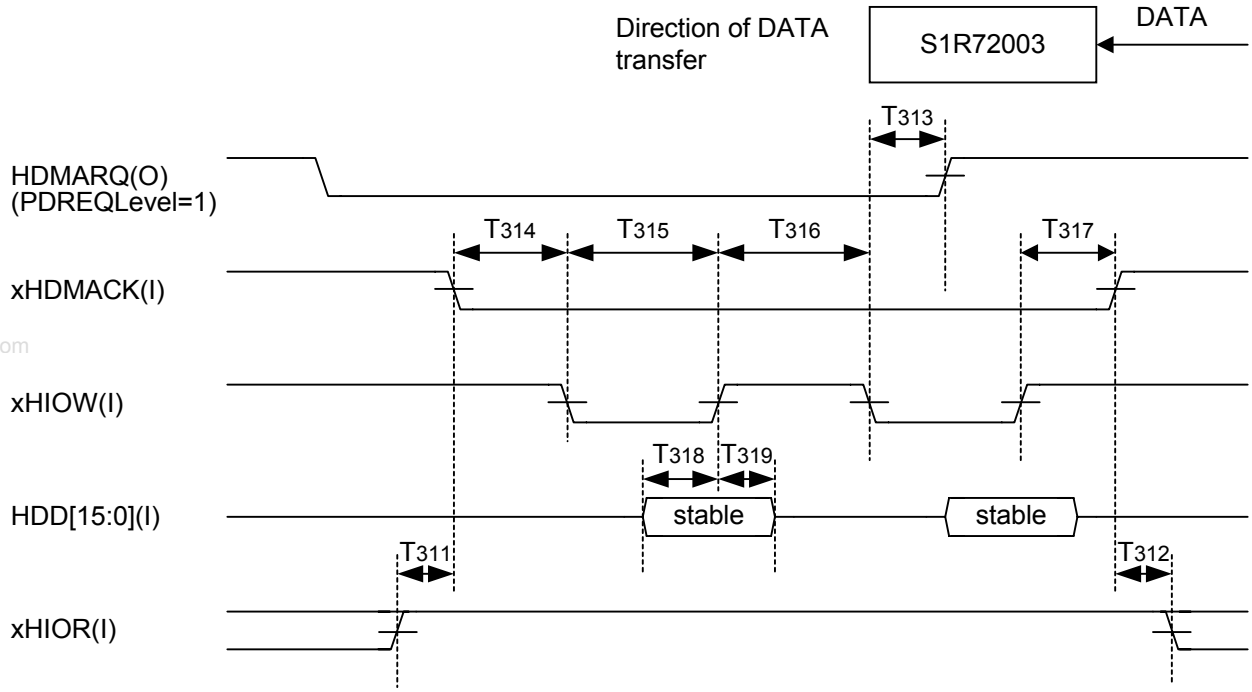





| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{301}$ | xHIOW → xHDMACK ↓<br>xHIOW setup time | 5 | — | — | ns |
| $T_{302}$ | xHDMACK ↑ → xHIOW<br>xHIOW hold time | 5 | — | — | ns |
| $T_{303}$ | xHIOR ↓ → HDMARQ negate<br>HDMARQ negate delay time | 0 | 25 | 37 | ns |
| $T_{304}$ | xHDMACK ↓ → xHIOR ↓<br>xHDMACK setup time | 0 | — | — | ns |
| $T_{305}$ | xHIOR ↓ → xHIOR ↑<br>xHIOR assert pulse width | 25 | — | — | ns |
| $T_{306}$ | xHIOR ↑ → xHDMACK ↑<br>xHDMACK hold time | 0 | — | — | ns |
| $T_{307}$ | xHIOR ↓ → HDD<br>Data output delay time *1 | 0 | — | 25 | ns |
| $T_{308}$ | xHIOR ↑ → HDD(Hi-Z)<br>Data bus negate time *1 | 6 | — | 40 | ns |

*1: Data is output to HDD only when both xHDMACK and xHIOR are asserted.
Except the above period, HDD enters the input mode.

### 9.4.3.4 DMA Write Timing (Slave Mode)





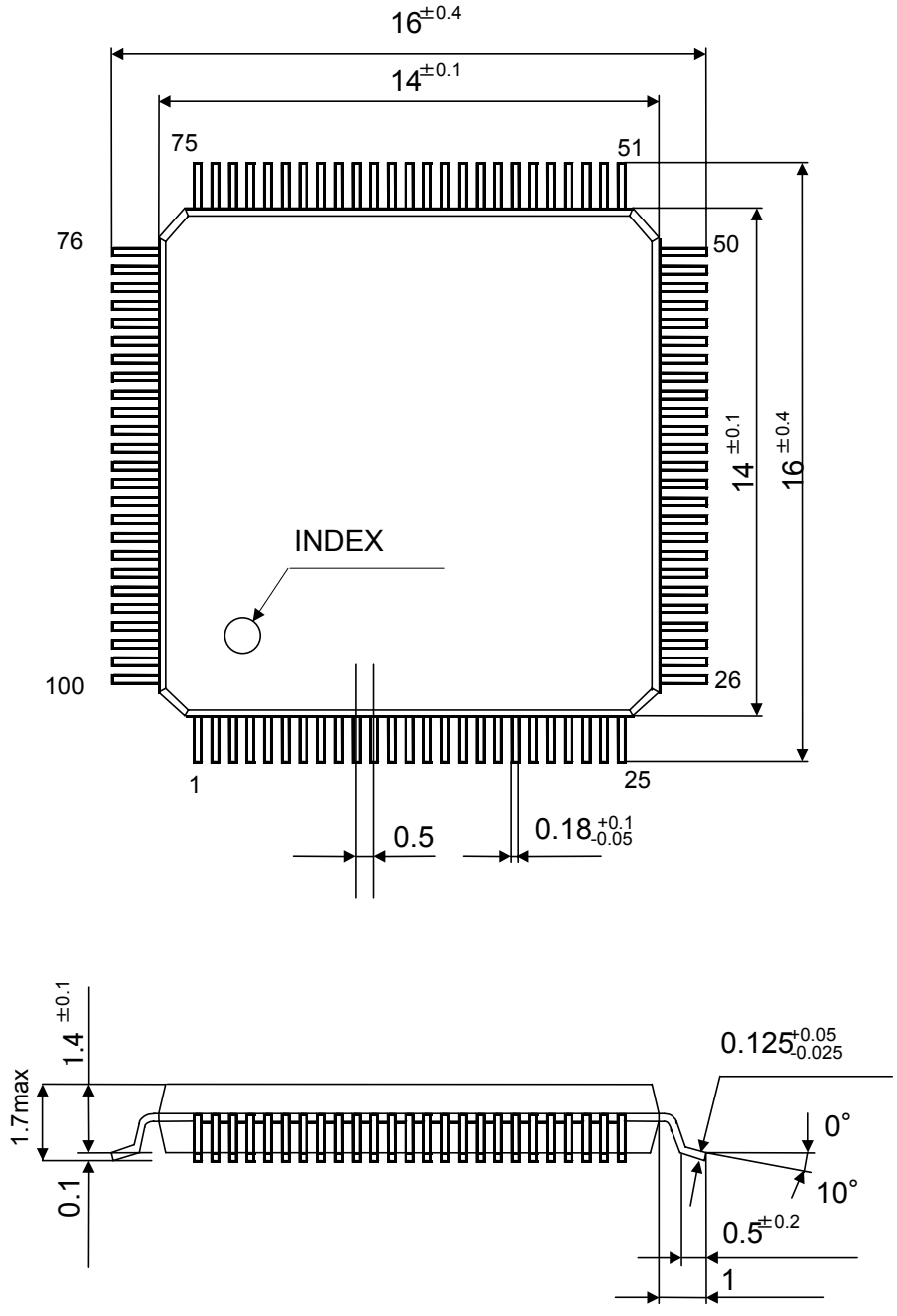

| Symbol | Description | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| $T_{311}$ | xHIOR → xHDMACK ↓<br>xHIOR setup time | 5 | — | — | ns |
| $T_{312}$ | xHDMACK ↑ → xHIOR<br>xHIOR hold time | 5 | — | — | ns |
| $T_{313}$ | xHIOW ↓ → HDMARQ negate<br>HDMARQ negate delay time | 0 | 25 | 37 | ns |
| $T_{314}$ | xHDMACK ↓ → xHIOW ↓<br>xHDMACK setup time | 0 | — | — | ns |
| $T_{315}$ | xHIOW ↓ → xHIOW ↑<br>xHIOW assert pulse width | 25 | — | — | ns |
| $T_{316}$ | xHIOW ↑ → xHIOW ↓<br>xHIOW negate pulse width | 25 | — | — | ns |
| $T_{317}$ | xHIOW ↑ → xHDMACK ↑<br>xHDMACK hold time | 0 | — | — | ns |
| $T_{318}$ | HDD → xHIOW ↑<br>Data setup time | 10 | — | — | ns |
| $T_{319}$ | xHIOW ↑ → HDD<br>Data hold time | 0 | — | — | ns |

### 9.4.4 USB I/F Timing

Conforms to USB 2.0 specification.

# 10. EXTERNAL PACKAGE

Plastic QFP15-100




Unit : mm

# APPENDIX-A. USB OPERATION OTHER THAN TRANSFER

## A.1 Suspend Detection

### A.1.1 Suspend Detection (HS Mode)

If no sending and receiving are detected for 3 ms or more ($T_1$) when this IC operates in HS mode, the mode automatically moves to the FS mode (the HS termination is disabled and the FS termination (Rpu) is enabled). This operation sets the DP line to "H", and "J" can be checked in USBStatus.LineState[1.0] (If "SE0" is detected, note that reset (described later) occurs). Subsequently, if "J" is still detected at $T_2$, the SIEIntStat. DectectSuspend bit is set.



On this occasion, because the xINT signal is asserted at the same time, judge that the state is a USB suspend state and, subsequently, enter snooze (PLL halt mode) processing until $T_4$. However, self-powered products may not be snoozed (Fig. A.1 shows the operation when snooze was performed).

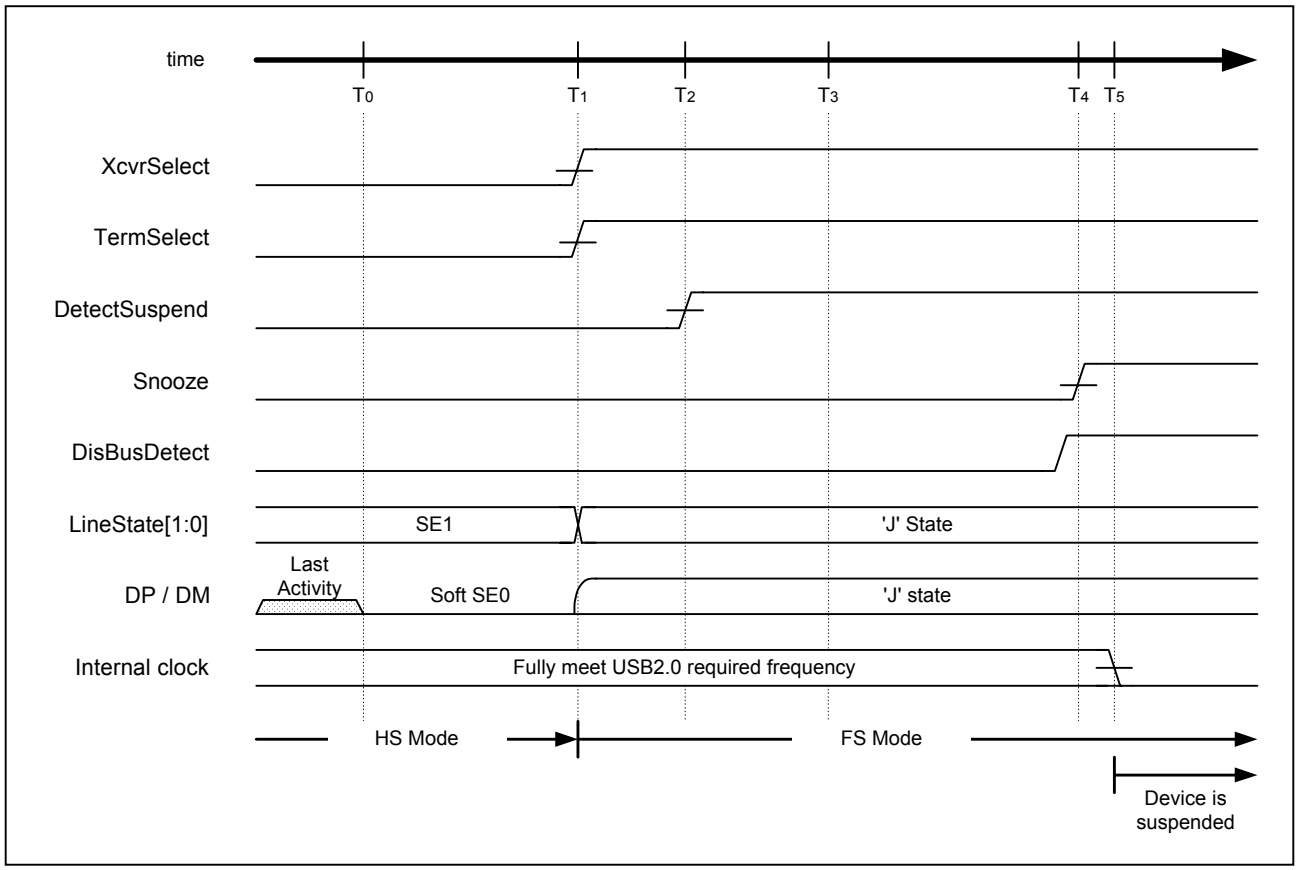



Fig. A.1 Suspend Timing (HS mode)

Table A.1 Suspend Timing Values (HS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | Last bus activity | 0 (reference) |
| $T_1$ | If there is no bus activity yet on this occasion, XcvrSelect and TermSelect are set to '1' and the HS mode is switched to the FS mode. | HS Reset $T_0$ + 3.0ms < $T_1${$T_{WTREV}$} < HS Reset $T_0$ + 3.125ms |
| $T_2$ | LineState[1:0] is sampled. At this time, DetectSuspend is set to '1' and this state is judged as a USB suspend state. | $T_1$ + 100μs < $T_2$ {$T_{WTWRSTHS}$} < $T_1$ + 875μs |
| $T_3$ | Resume must not be issued before this. | HS Reset $T_0$ + 5ms {$T_{WTRSM}$} |
| $T_4$ | Snooze is set to '1', thereby completely moving to snooze. Subsequently, the current exceeding the suspend current specified in USB must not be pulled from VBUS. (Before moving to snooze, DisBusDetect is set to '1'.) | HS Reset $T_0$ + 10ms {$T_{2SUSP}$} |
| $T_5$ | An internal clock is completely halted. (Snooze current of 8 mA (Typ.)) | $T_5$ < $T_4$ + 10μs |

Note: { } is a name standardized in the USB2.0 specifications.

### A.1.2 Suspend Detection (FS Mode)

If no sending and receiving are detected for 3 ms or more ($T_1$) when this IC operates in HS mode or if "J" continues being detected in USBStatus.LineState[1:0] ($T_1$) and "J" is still detected at T2, it is detected that the state is a USB suspend state and the SIEIntStat.DectectSuspend bit is set.

On this occasion, because the xINT signal is asserted at the same time, judge that this state is a USB suspend state and, subsequently, enter snooze (PLL halt mode) processing until $T_4$. However, self-powered products may not be snoozed (Fig. A.2 shows the operation when snooze was performed).



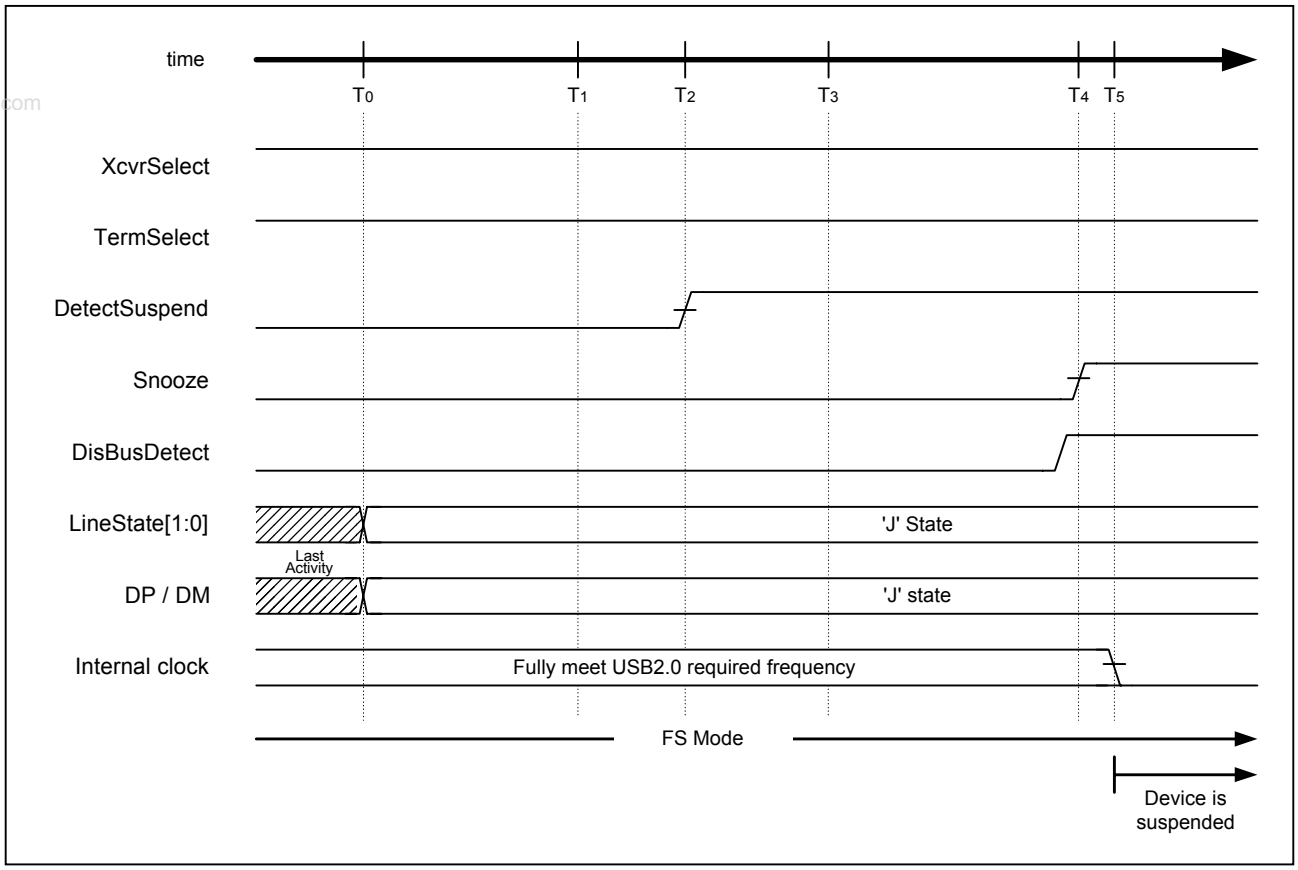



Fig. A.2 Suspend Timing (FS mode)

Table A.2 Suspend Timing Values (FS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | Last bus activity | 0 (reference) |
| $T_1$ | There is no bus activity yet at this point. | $T_0$ + 3.0ms < $T_1${$T_{WTREV}$} < $T_0$ + 3.125ms |
| $T_2$ | LineState[1:0] is sampled. At this time, if 'J', DetectSuspend is set to '1' and this state is judged as a USB suspend state. | $T_1$ + 100µs < $T_2$ {$T_{WTWRSTHS}$} < $T_1$ + 875µs |
| $T_3$ | Resume must not be issued before this. | $T_0$ + 5ms {$T_{WTRSM}$} |
| $T_4$ | Snooze is set to '1' , thereby completely moving to snooze. Subsequently, the current exceeding the suspend current specified in USB must not be pulled from VBUS. (Before moving to snooze, DisBusDetect is set to '1'.) | $T_0$ + 10ms {$T_{2SUSP}$} |
| $T_5$ | An internal clock is completely halted. (Snooze current of 8 mA (Typ.)) | $T_5$ < $T_4$ + 10µs |

Note: { } is a name standardized in the USB2.0 specifications.

## A.2 Reset Detection

### A.2.1 Reset Detection (HS Mode)

If no sending and receiving are detected for 3 ms or more ($T_1$) when this IC operates in HS mode, the mode automatically moves to the FS mode (the HS termination is disabled and the FS termination (Rpu) is enabled). Even if this operation is performed, the DP line is kept being set to "L". As a result, "SE0" can be detected even in USBStatus.LineState[1:0]. When "SE0" is still detected at $T_2$, the SIWIntStat.DetectReset bit is set.

On this occasion, because the xINT signal is asserted at the same time, judge that this is a reset indication. Subsequently, after setting the USBControl.DisBusDetect bit, perform HS Detection Handshake (described later).



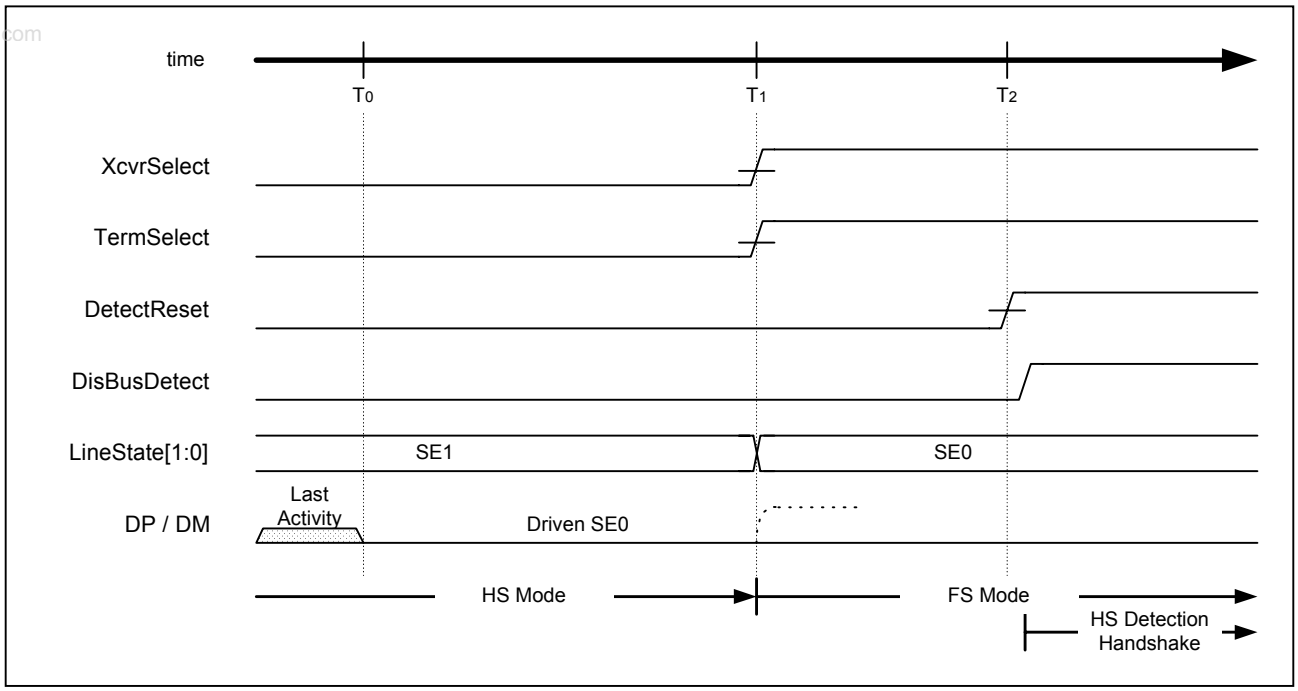



Fig. A.3 Reset Timing (HS mode)

Table A.3 Reset Timing Values (HS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | Last bus activity | 0 (reference) |
| $T_1$ | If there is no bus activity yet at this point, XcvrSelect and TermSelect are set to '1' and the HS mode is switched to the FS mode. | HS Reset $T_0$ + 3.0ms < $T_1${$T_{WTREV}$} < HS Reset $T_0$ + 3.125ms |
| $T_2$ | LineState[1:0] is sampled. At this time, if "SE0', DetectSuspend is set to '1' and this state is judged as a move to reset. After a reset indication is detected, DisBusDetect is set to '1' and, subsequently, HS Detection Handshake is performed. | $T_1$ + 100us < $T_2$ {$T_{WTWRSTHS}$} < $T_1$ + 875μs |

Note: { } is a name standardized in the USB2.0 specifications.

### A.2.2 Reset Detection (FS Mode)

If "SE0" continues being detected in USBStatus.LineState[1:0] for 2.5 µs ($T_1$) when this IC operates in FS mode, the SIEIntStat.DetectReset bit is set.

On this occasion, because the xINT signal is asserted at the same time, judge that this is a reset indication. Subsequently, after setting the USBControl.DisBusDetect bit, perform HS Detection Handshake (described later).



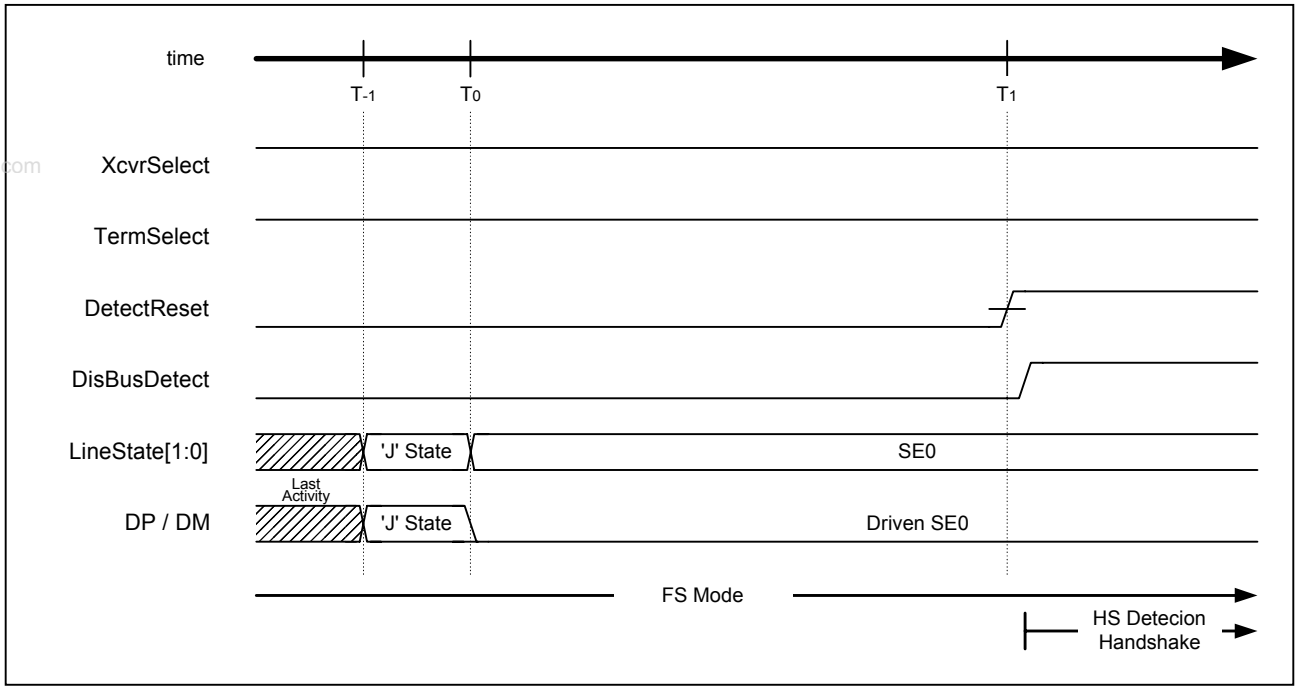



Fig. A.4 Reset Timing (FS mode)

Table A.4 Reset Timing Values (FS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_{-1}$ | Last bus activity | |
| $T_0$ | A reset indication starts from a downstream port. | 0 (reference) |
| $T_1$ | When "SE0" is continuing, DetectReset is set to '1' and this state is judged as a move to reset.<br>After a reset indication is detected, DisBusDetect is set to '1' and, subsequently, HS Detection Handshake is performed. | HS Reset $T_0$ + 2.5µs < $T_2${$T_{WTREV}$} < |

Note: { } is a name standardized in the USB2.0 specifications.

## A.3 HS Detection Handshake

High-Speed Detection Handshake is started from any one of the three states in snooze/sleep, FS operation, or HS operation by asserting "SE0" from a downstream port (when reset is started from the above state). Do not move to the HS Detection Handshake for 4 ms from the reset start. For details, see the USB2.0 specifications.

This section describes the method of moving from the above three states to the HS Detection Handshake.

1) If "SE0" is detected on a bus when this IC is in the snooze/sleep state: Subsequently, move to HS Detection Handshake.
2) If "SE0" exceeding 2.5 µs is detected when this IC is operating in FS mode: Subsequently, move to HS Detection Handshake.
3) If "SE0" exceeding 3.0 ms is detected when this IC is operating in HS mode: In the HS mode, first, because it must be judged that this state is a USB suspend state or reset, the mode must be switched to the FS mode once. Accordingly, by switching both bits XcvrControl.XcvrSelect and XcvrControl.TermSelect to the FS mode, the HS termination is disabled and the FS termination is enabled. These modes must be switched within 3.125 ms. Check USBStatus.LineState[1:0] between 100 µs or more and less than 875 µs after this mode switching. If "J", judge that this state is as a USB suspend state and if "SE0", judge it as reset. At this time, when the state is judged as the reset, subsequently, move to HS Detection Handshake.



In either case, the reset exists for 10 ms at a minimum, but the timing slightly differs according to the state (HS or FS) before move. Here, the time the reset was started is defined as "HS Reset $T_0$" and, subsequently, the operation from this "HS Reset $T_0$" is described.

In the above case 3), the IC is in operation, an internal clock is also stable sufficiently, thereby causing no problem, but attention needs to be paid to the cases 1) and 2). In the case 1), the IC enters the snooze/sleep state and an internal clock may not be output when the reset is detected. To output the internal clock so that HS Detection Handshake can be performed, PM Control.AnalogPwdown, PM Control. Snooze, and PMControl.PLLSel are all set to "0", that is, PLL480 must operate. When PMControl. Snooze is set from "1" to "0", a fixed PLL stable time is necessary. Further, if the state moves to the sleep state, an oscillation stable time is necessary. (This oscillation stable time varies according to the status of an oscillator and an oscillator circuit). In the case 2), the IC operates in FS mode and the internal clock is also stable fully. On this occasion, when PLLSel selects PLL60, the PLL stable time is necessary in the same manner as 1).

Because the PLL stable time is very short, it needs not to be conscious of much. However, because the oscillation stable time is in a unit of several ms, attention must be paid extremely.

### A.3.1 When This IC is Connected to FS Downstream Port

The operation when this IC is connected to a downstream port that does not support HS is shown. When the HS Detection Handshake starts (T0), both bits XcrvControl.XcrvSelect and XcrvControl.TermSelect must be set in FS mode (the FS termination, that is, the DP pull up resistor (Rpu) is enabled and the HS termination is disabled).

First, the USBControl.GoChirp bit is set. Subsequently, XcvrControl.OpMode[1:0] is set for "Disable Bit Stuffing and NRZI encoding" and the data padded with "0" is prepared ($T_1$). This is used to issue "HS K" (chirp) onto a bus. Further, at the same time, when XcrvControl.XcrvSelect bit is set in HS mode and set in the ready for sending state, "HS K" (chirp) is issued to the downstream port. After it is issued, a chirp is awaited from the downstream port ($T_2$). Usually, the downstream port that supports HS continuously issues "HS K" and "HS J" from $T_3$ (described later). However, when the downstream port does not support HS (this time), no chirp is issued even at $T_4$, the XcrvControl.XcrvSelect bit is automatically switched to the FS mode and the USBControl.GoChirp bit is cleared. At the same time, the USBStatus.FSxHS bit is set and the SIEIntStat.ChirpCmp bit is set.



On this occasion, because the xINT signal is asserted at the same time, judge that HS Detection Handshake terminated.

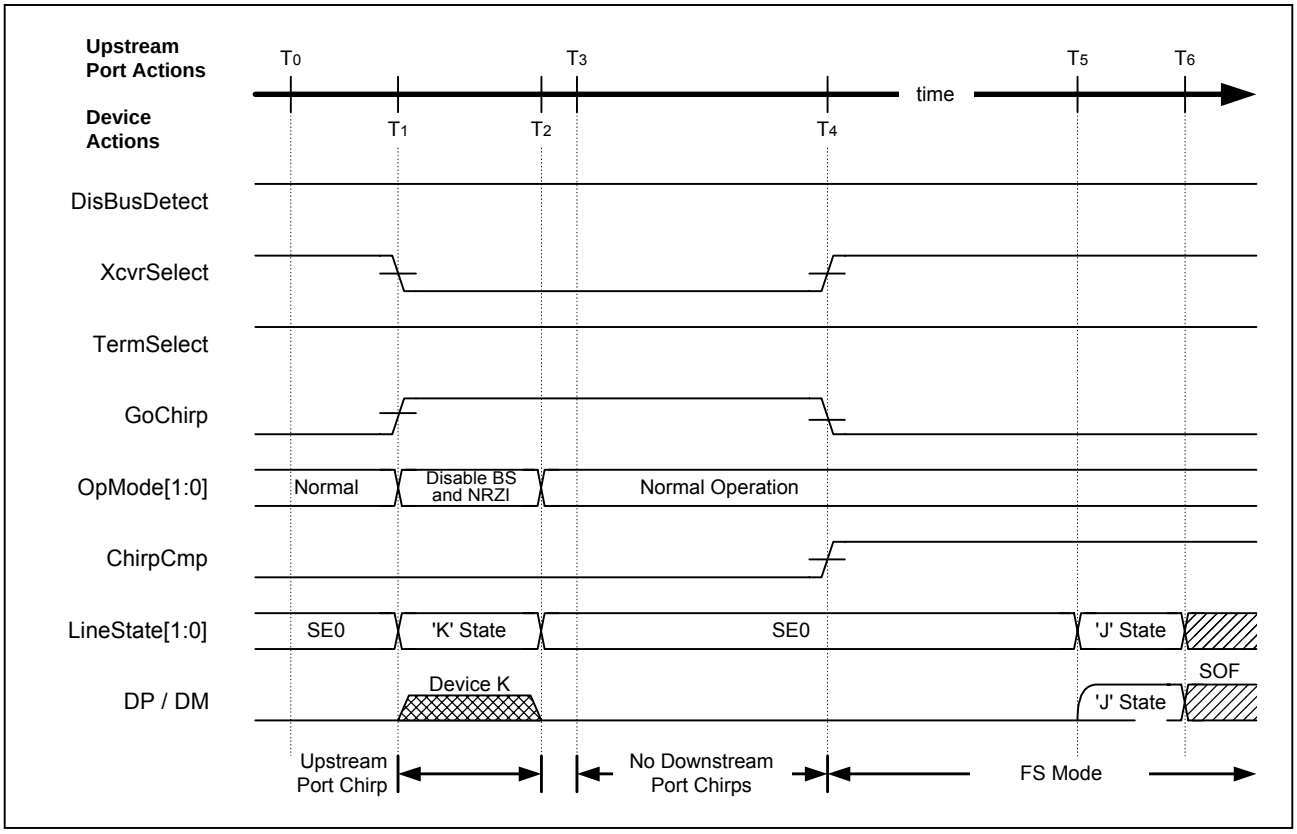



Fig. A.5 HS Detection Handshake Timing (FS mode)

Table A.5 HS Detection Handshake Timing Values (FS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | HS Detection Handshake starts. | 0 (reference) |
| $T_1$ | The HS transceiver is enabled and GoChirp is set to '1', then Chirp K starts being issued. | $T_0 < T_1 <$ HS Reset $T_0$ + 6.0ms |
| $T_2$ | The issue of Chirp K terminates. It must be issued for a minimum of 1 ms. | $T_1$ + 1.0ms {$T_{UCH}$} < $T_2$ < HS Reset $T_0$ + 7.0ms {$T_{UCHEND}$} |
| $T_3$ | When the downstream port supports HS, Chirp K starts being issued from here. | $T_2 < T_3 < T_2$ + 100μs {$T_{WTDCH}$} |
| $T_4$ | If no Chirp can be detected, at this point, the mode returns to the FS mode and ChirpCmp is set to '1', then it is awaited that the reset sequence terminates. | $T_2$ + 1.0ms < $T_4$ {$T_{WTFS}$} < $T_2$ + 2.5ms |
| $T_5$ | End of the reset sequence | HS Reset $T_0$ + 10ms {$T_{DRST}$(Min.)} |
| $T_6$ | Normal operation in FS mode | $T_6$ |

Note: { } is a name standardized in the USB2.0 specifications.

Note: To generate Chirp K for a minimum of 1 ms, judge at 66,000 cycles (internal clock: 60 MHz).

### A.3.2 When This IC is Connected to HS Downstream Port

The operation when this IC is connected to a downstream port that does not support HS is shown. When HS Detection Handshake starts (T0), both bits XcrvControl.XcrvSelect and XcrvControl.TermSelect must be set in FS mode (the FS termination, that is, the DP pull up resistor (Rpu) is enabled and the HS termination is disabled).

First, the USBControl.GoChirp bit is set. Subsequently, XcvrControl.OpMode[1:0] is set for "Disable Bit Stuffing and NRZI encoding" and the data padded with "0" is prepared ($T_1$). This is used to issue "HS K" (chirp) onto a bus. At the same time, when XcrvControl.XcrvSelect bit is set in HS mode and set in the ready for sending state, "HS K" (chirp) is issued to the downstream port. After it is issued, a chirp is awaited from the downstream port ($T_2$). Thereupon, because the downstream port supports HS, "HS K" (Chirp K) and "HS J" (Chirp J) are alternately issued continuously ($T_3$). When this state is detected six times as Chirp K-J-K-J-K-J in USBStatus.LineState[1:0] ($T_6$), the XcvrControl.TermSelect bit is automatically switched to the HS mode ($T_7$) and moves to the perfect HS mode. Simultaneously with this, the USBControl.GoChirp bit is cleared. At the same time, the USBStatus.FSxHS bit is set and the SIEIntStat.ChirpCmp bit is set.



On this occasion, because the xINT signal is asserted at the same time, judge that HS Detection Handshake terminated.

This Chirp K or Chirp J from the downstream port is recognized as a bus activity and must not be judged as a USB suspend state. Hence, in the HS mode, this Chirp K or Chirp J is detected sequentially and fetched in an internal Suspend Timer. Besides, to detect Chirp K-J-K-J-K-J, USBStatus.LineState[1:0] is used. Unlike a usual HS packet, because chirp K and Chirp J are very slow, they can be detected in USBStatus.LineState[1:0]. However, if a bus signal is loaded on USBStatus.LineState[1:0] when an original packet is received, the signal is very noisy. Accordingly, when TermSelect is in HS mode, USBStatus.LineState[1:0] outputs "SE1".

Fig. A.6 shows that the HS termination at the device side is enabled because the height of Chirp varies at $T_6$. Usually, the chirp when TermSelect is in FS mode is about 800 mV and the Chirp (equally in the HS normal sending and receiving packet) when TermSelect is in HS mode is about 400 mV.

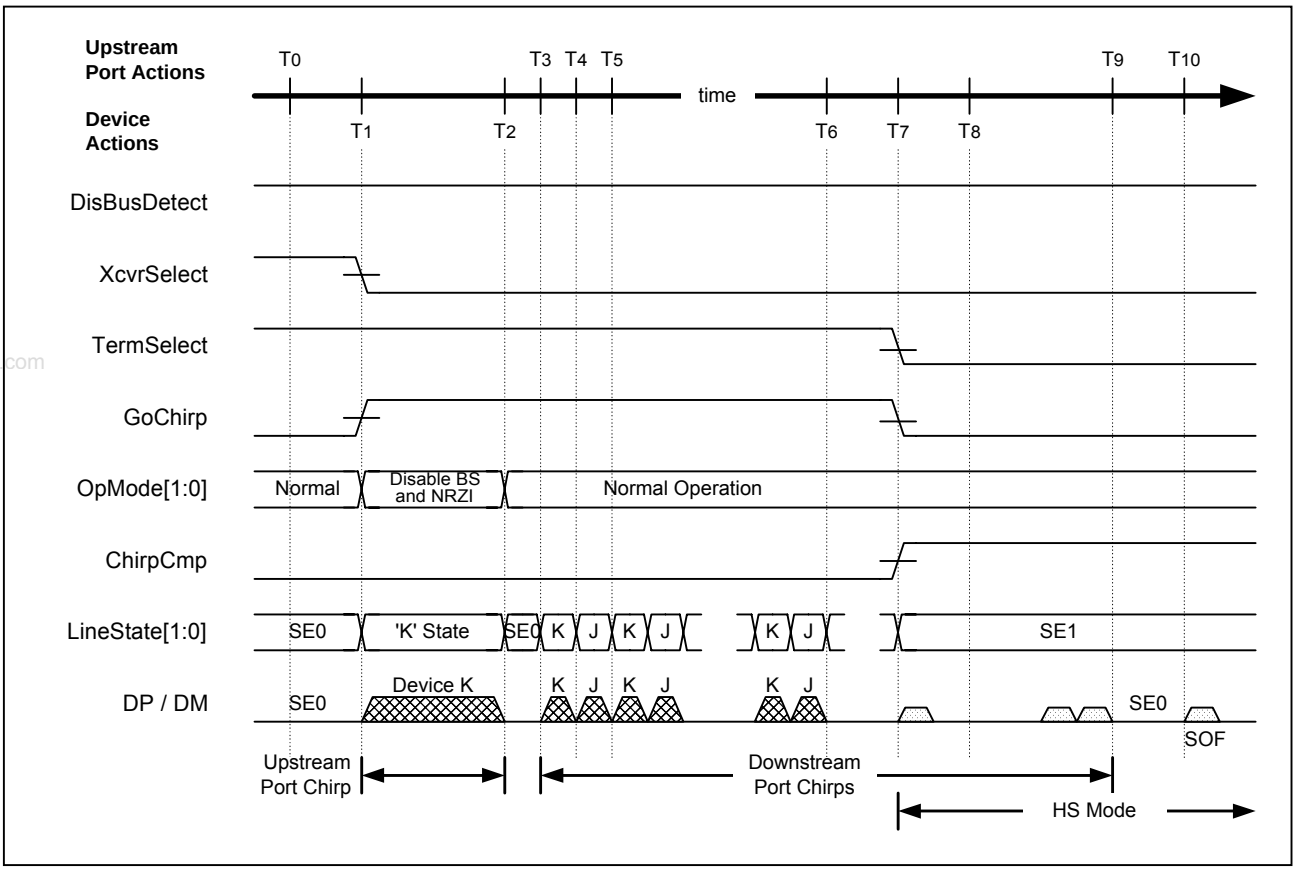



Fig. A.6 HS Detection Handshake Timing (HS mode)

Table A.6 HS Detection Handshake Timing Values (HS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | HS Detection Handshake starts. | 0 (reference) |
| $T_1$ | The HS transceiver is enabled and GoChirp is set to '1', then Chirp K starts being issued. | $T_0 < T_1$ < HS Reset $T_0$ + 6.0ms |
| $T_2$ | The issue of Chirp K terminates. It must be issued for a minimum of 1 ms. | $T_1$ + 1.0ms {$T_{UCH}$} < $T_2$ < HS Reset $T_0$ + 7.0ms {$T_{UCHEND}$} |
| $T_3$ | The downstream port issues the first Chirp K to a bus. | $T_2 < T_3 < T_2$ + 100µs {$T_{WTDCH}$} |
| $T_4$ | The downstream port switches Chirp K to Chirp J and issues Chirp J. | $T_3$ + 40µs {$T_{DCHBIT}$(Min.)} < $T_4$ < $T_3$ + 60µs { $T_{DCHBIT}$(Max.)} |
| $T_5$ | The downstream port switches Chirp J to Chirp K and issues Chirp K. | $T_4$ + 40µs {$T_{DCHBIT}$(Min.)} < $T_5$ < $T_4$ + 60µs { $T_{DCHBIT}$(Max.)} |
| $T_6$ | Chirp K-J-K-J-K-J is detected. | $T_6$ |
| $T_7$ | When Chirp K-J-K-J-K-J is detected, the FS termination is disabled and the HS termination is enabled. ChirpCmp is set to '1'. Further, reset termination is awaited. | $T_6 < T_7 < T_6$ + 500µs |
| $T_8$ | Recognized as a bus activity using Chirp K or Chirp J. However, because SYNC cannot be detected, it is not recognized that the packet is being received. | $T_8$ |
| $T_9$ | The issue of chirp K or Chirp J terminates from the downstream port. | $T_{10}$ - 500µs {$T_{DCHSE0}$(Max.)} < $T_9$ < $T_{10}$ - 100µs{$T_{DCHSE0}$(Min.)} |
| $T_{10}$ | End of the reset sequence | HS Reset $T_0$ + 10ms {$T_{DRST}$(Min.)} |

Note: { } is a name standardized in the USB2.0 specifications.

Note: To generate Chirp K for a minimum of 1 ms, judge at 66,000 cycles (internal clock: 60 MHz).

### A.3.3 When This IC is Reset in Snooze

When this IC is in the snooze state, an internal clock is also output. Here, an oscillator circuit is assumed to operate (not in the sleep state but in the snooze state) and the operation is described. The PMControl.Snooze bit affects only the PLL operation and will not affect the oscillator circuit. Accordingly, when the PMControl.Snooze bit is set from "1" to "0", the PLL Powerup time is necessary.

When the IC is in the snooze state and reset is detected ($T_0$), the SIEIntStat.NonJ bit is set and, at the same time, the xINT signal is asserted. Clear the PMControl. Snooze to "0" so that the IC can be recovered from snooze and immediately enter the reset sequence ($T_1$). After the Powerup time elapses ($T_2$), the PMControl. InSnooze bit is cleared, and, at the same time, an internal starts being output. Subsequently, perform HS Detection Handshake (described later).



At this time, if the oscillator circuit is not halted (not recovered from the snooze state), the internal clock is output at the frequency accuracy that conforms to the USB2.0 specifications.

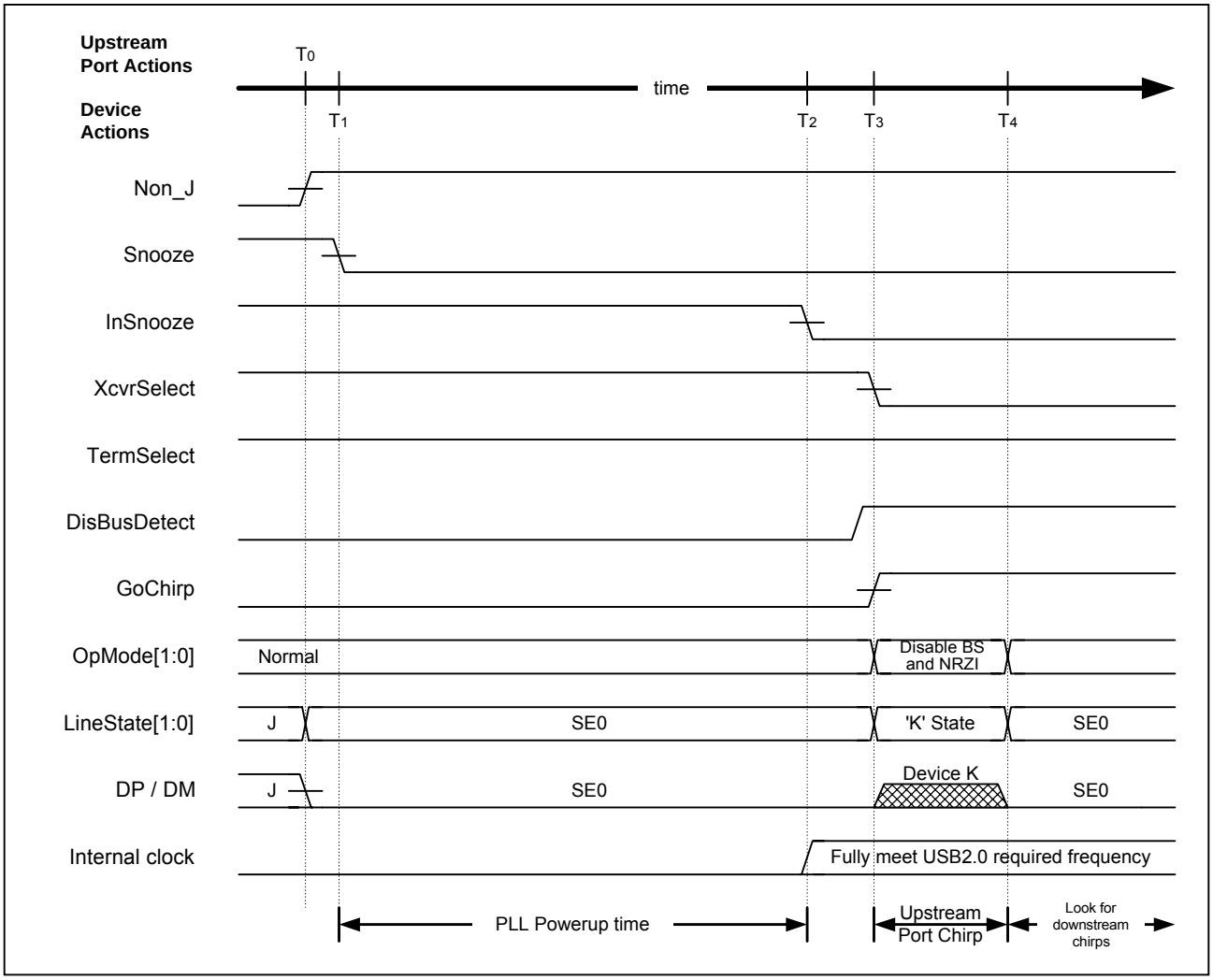



Fig. A.7 HS Detection Handshake Timing from Suspend

Table A.7 HS Detection Handshake Timing Values from Suspend

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | When Non_J is set to '1' and 'SE0' is checked in LineState[1:0], the reset in snooze is detected. | 0 (HS Reset $T_0$) |
| $T_1$ | After the reset is detected, Snooze is cleared to '0'. | $T_1$ |
| $T_2$ | InSnooze is set to '0'. The internal clock output is stable. | $T_1 + 250\mu s < T_2$ |
| $T_3$ | GoChirp is set to '1' and Chirp K is issued to a bus. (Before Chirp K is issued, DisBusDetect is set to '1'.) | $T_2 < T_3 <$ HS Reset $T_0 + 5.8ms$ |
| $T_4$ | The issue of Chirp K terminates. | $T_3 + 1.0ms$ {$T_{UCH}$} $< T_4 <$ HS Reset $T_0 + 7.0ms$ {$T_{UCHEND}$} |



Note: { } is a name standardized in the USB2.0 specifications.

Note: To generate Chirp K for a minimum of 1 ms, judge at 66,000 cycles (internal clock: 60 MHz).

Note: The case (sleep state) where the oscillator circuit is also halted is described later (In addition to the PLL Powerup time, the OSC Powerup time is necessary).

## A.4 Issue of Resume

This section describes a method of resuming itself for some reason when a remote wakeup is supported and this remote wake-up is enabled from a host. However, the remote wakeup can be performed at least after 5 ms elapse when a bus becomes idle. Further, before a lapse of 10 ms after a resume signal is issued, the current before a device enters the USB suspend state cannot be pulled from VBUS.

The device is first recovered from snooze/sleep to wake up. The SIEIntEnb.EnNonJ bit is cleared and the PMControl.Snooze bit is cleared ($T_0$). After the PLL Powerup time elapses ($T_1$), the PMControl. InSnooze bit is cleared and, at the same time, an internal clock starts being output. At this time, if an oscillator circuit is halted, this internal clock is output at the frequency accuracy that conforms to the USB2.0 specifications.



Subsequently, the USBControl.SendWakeup bit is set and a resume signal is issued ($T_2$). At this time, internally, XcrvControl.OpMode[1:0] is set for "Disable Bit Stuffing and NRZI encoding" and "0" is prepared as transmission data. A packet sending state is set and "K" (Resume signal) is issued. A downstream port detects this resume signal and returns "K" (resume signal) onto a bus ($T_3$).

After about 1 ms when the resume signal starts being issued, the resume signal that was issued to the bus by clearing the USBControl.SendWakeup bit is halted $T_4$). However, at this point, the downstream port still holds the bus in the resume signal.

Then the USBConstrol.RestoreUSB bit is set. After a fixed time elapses, the downstream port stops the issue of the resume signal ($T_5$) and is switched to the speed mode before USB suspend. When this is detected (not "K"), both bits XcrvControl.XcrvSelect and XcrvControl.TermSelect are switched to a desired mode (HS mode at this time) and the USBControl.RestoreUSB bit is cleared. Simultaneously, the SIEIntStat.RestoreCmp bit is set and, at the same time, the xINT signal is asserted.

Here, when the USB suspend starts, the speed mode (HS or FS) is stored as the USBStatus.FSxHS bit. When a device is recovered using resume, the mode that this USBStatus.FSxHS bit indicates is set. HS Detection Handshake is not necessary every resume. Note that only the case where the mode before the USB suspend was in the HS mode is described here. Actually, in the FS mode, the normal FS mode occurs at T5 or later and there is no great sequence change in particular.

When this IC is in the snooze state (the PMControl. Snooze bit is "1"), an internal clock is not output. The operation is described here, assuming that an oscillator circuit operates (not in the sleep state but the snooze state). The PMControl. Snooze bit affects only the PLL operation and will not affect the operation of the oscillator circuit. Accordingly, when the PMControl. Snooze bit is cleared, the PLL Powerup time is necessary. If the oscillator circuit is also halted in snooze (in the sleep state and the oscillator circuit stops), the OSC Powerup time is also necessary in addition to the PLL Powerup time. This OSC Powerup time is described later.

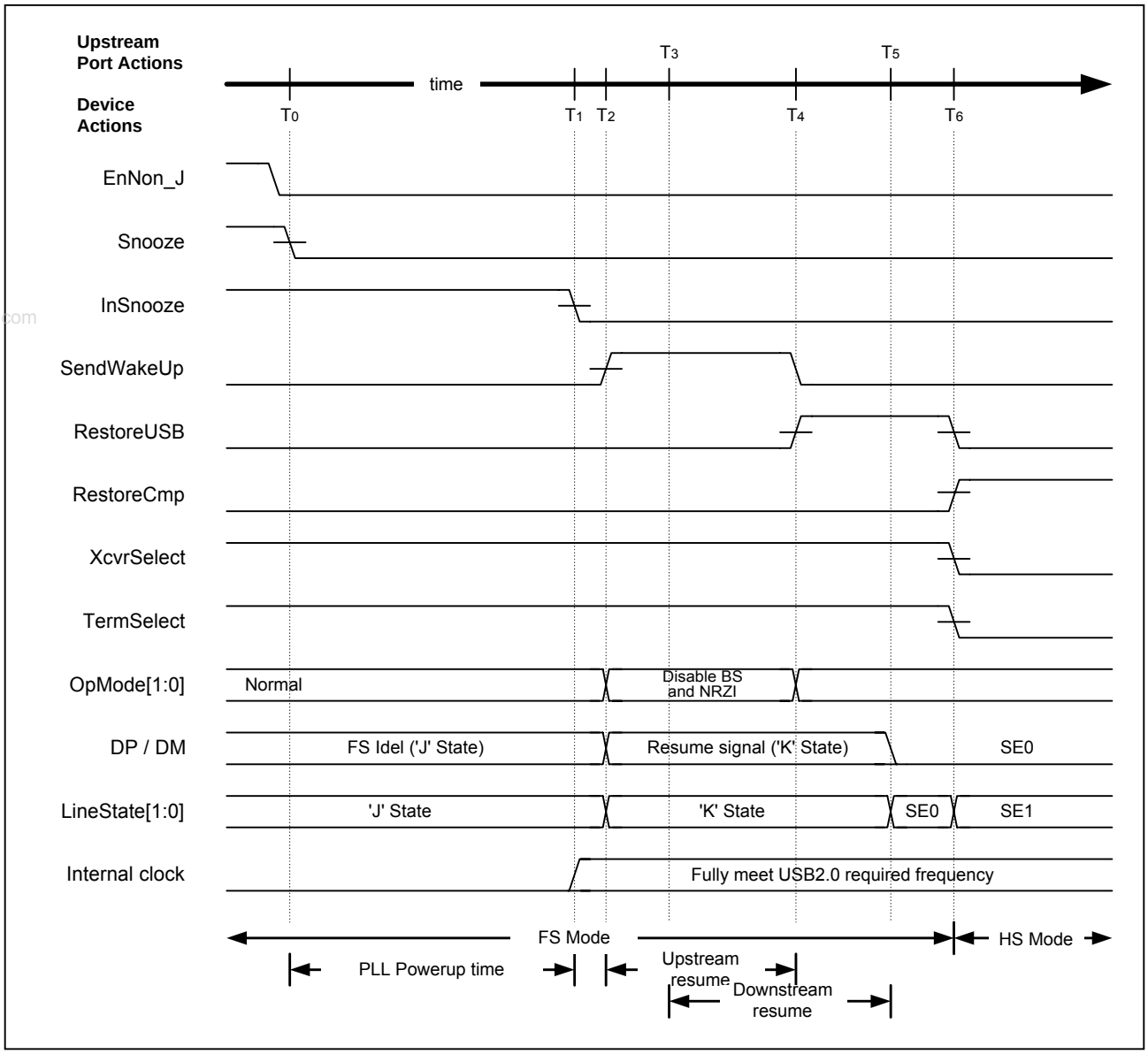



Fig. A.8 Assert Resume Timing (HS mode)

Table A.8 Assert Resume Timing Values (HS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | Resume starts. Snooze is cleared to ‘o’.<br>(Before resume starts, EnNon_J is cleared to ‘0’.) | 0 (reference) |
| $T_1$ | InSnooze is set to ‘0’. The internal clock output becomes stable. | $T_0 + 250\mu s < T_1$ |
| $T_2$ | SendWakeUp is set to ‘1’ and “K” of FS starts being issued. Here, the current before USB suspend must not be pulled within 10 ms. | $T_0 < T_2 < T_0 + 10ms$ |
| $T_3$ | The downstream port returns “K” of FS. | $T_2 < T_3 < T_2 + 1.0ms$ |
| $T_4$ | SendWakeUp is cleared to ‘0’ and the issue of “K” of FS terminates. After “K” is checked using LineState[1:0], RestoreUSB is set to ‘1’. | $T_2 + 1.0ms$ {$T_{DRSMUP}$(Min.)} < $T_4$ < $T_2 + 15ms$ {$T_{DRSMUP}$(Max.)} |
| $T_5$ | The downstream port terminates the issue of “K” of FS. | $T_2 + 20ms$ {$T_{DRSMDN}$} |
| $T_6$ | RestoreCmp is set to ‘1’. When the mode before USB suspend is the HS mode, it automatically moves to the HS mode. | $T_5 + 1.33\mu s$ {2 Low-speed bit times} |

Note: { } is a name standardized in the USB2.0 specifications.

## A.5 Detection of Resume

When this IC is snoozed, "J" (USBStatus.LineState[1.0] is "J") is observed on a bus. When "K" is observed on the bus, "K" is issued from the downstream port, and, at this time, a wakeup indication (resume indication) might have been received ($T_0$). At this time, if an oscillator circuit does not stop the operation (not in the sleep state), the SIEIntStat.NonJ bit is set and, at the same time, the xINT signal is asserted.

First, the PMControl. Snooze bit is cleared to "0" ($T_1$). After the PLL Powerup time elapses, the PMControl. InSnooze bit is cleared, and, at the same time, an internal clock starts being output. At this time, if an oscillator circuit is halted, this internal clock is output at the frequency accuracy that conforms to the USB2.0 specifications.

Then the USBControl.RestoreUSB bit is set. After a fixed time elapses, the downstream port stops the issue of the resume signal ($T_5$) and is switched to the speed mode before USB suspend. When this is detected (not "K"), both bits XcrvControl.XcrvSelect and XcrvControl.TermSelect are switched to a desired mode (HS mode at this time) and the USBControl.RestoreUSB bit is cleared. Simultaneously, the SIEIntStat.RestoreCmp bit is set and, at the same time, the xINT signal is asserted.



The operation is described here, assuming that an oscillator circuit operates (not in the sleep state but the snooze state). The PMControl. Snooze bit affects only the PLL operation and will not affect the operation of the oscillator circuit. Accordingly, when the PMControl. Snooze bit is cleared, the PLL Powerup time is necessary. If the oscillator circuit is also halted in snooze (in the sleep state and the oscillator circuit is halted), the OSC Powerup time is also necessary in addition to the PLL Powerup time. This OSC Powerup time is described later.

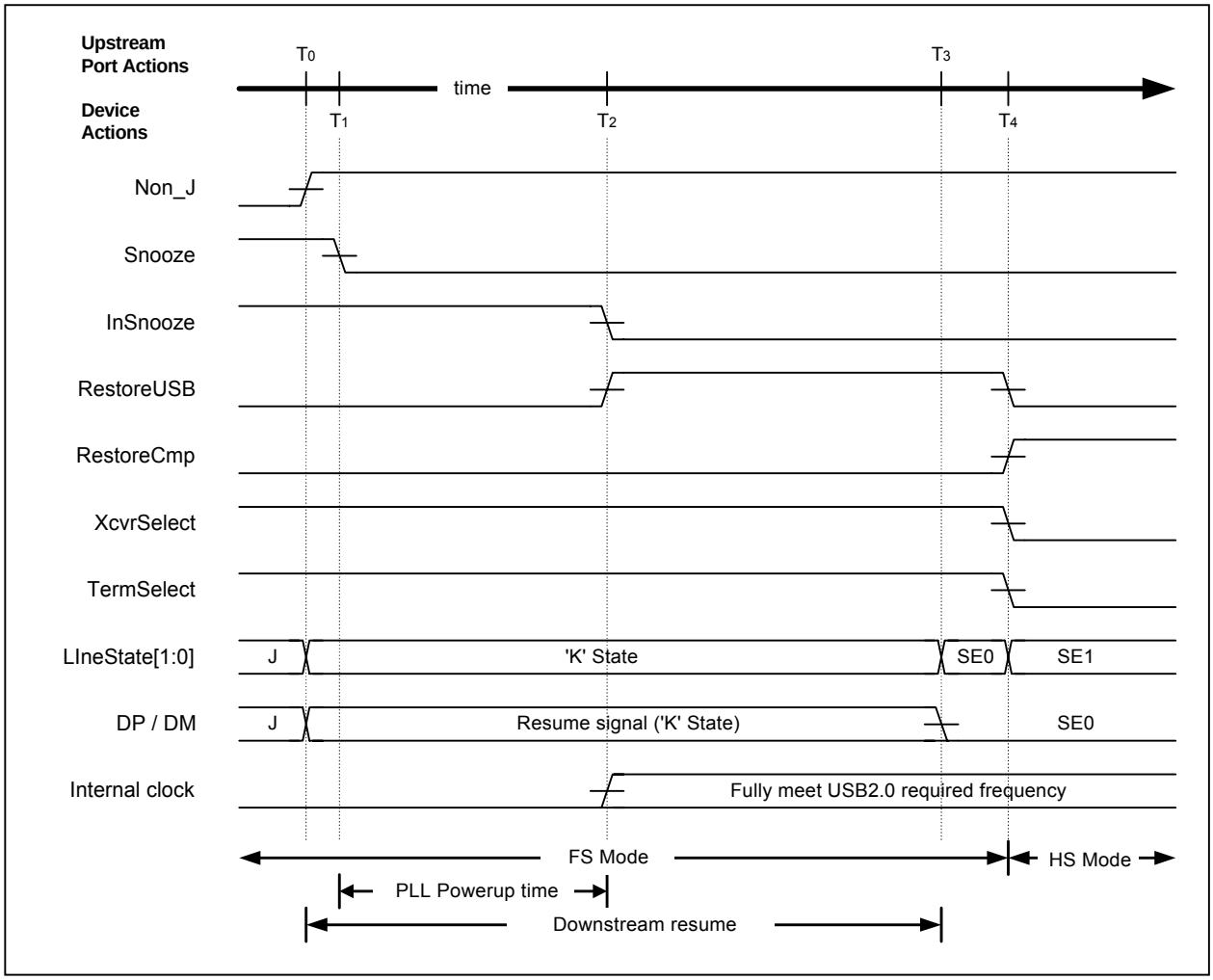



Fig. A.9 Detect Resume Timing (HS mode)

Table A.9 Detect Resume Timing Values (HS mode)

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | The downstream port issues "K" of FS. Non_J is set to '1'. | 0 (reference) |
| $T_1$ | Snooze is cleared to '0'. | $T_1$ |
| $T_2$ | InSnooze is set to '0'. The internal clock output becomes stable.<br>After "K" is checked in LineState [1:0], RestoreUSB is set to '1'. | $T_1 + 250\mu s < T_2$ |
| $T_3$ | The downstream port terminates the issue of "K" of FS. At the same time, the downstream port moves to the HS mode before USB suspend. | $T_2$ + 20ms {$T_{DRSMDN}$} |
| $T_4$ | When the mode before USB suspend is the HS mode, it automatically moves to the HS mode. | $T_5$ + 1.33μs {2 Low-speed bit times} |

Note: { } is a name standardized in the USB2.0 specifications.

## A.6 Cable Insertion

This section describes the case where a device is connected to a hub or a host, that is, the cable is inserted.

When the cable is removed or not connected intentionally, set the HS mode as an initial value for both bits XcvrControl.XcvrSelect and XcrvControl.TermSelect. When the cable is connected, VBUS is set to "H" and, at the same time, the USBStatus.VBUS bit is set ($T_0$). Subsequently, when the snooze state is set, clear the PMControl. Snooze bit ($T_1$). Hereupon, if reset applies to a built-in UTM by setting the PMControl.ResetUTM bit and clearing it after a fixed time ($T_2$), an internal clock is output before it becomes stable ($T_1$ to $T_3$). If this operation is not performed, the internal clock is not output. Accordingly, wait until the PMControl. InSnooze bit is cleared. Subsequently, first, because the FS device must be connected, set both bits XcrvControl.XcvrSelect and XcvrControl.TermSelect in the FS mode so that the FS mode can be set once.



Subsequently, the downstream port issues reset ($T_5$) and, at the same time, HS Detection Handshake starts.

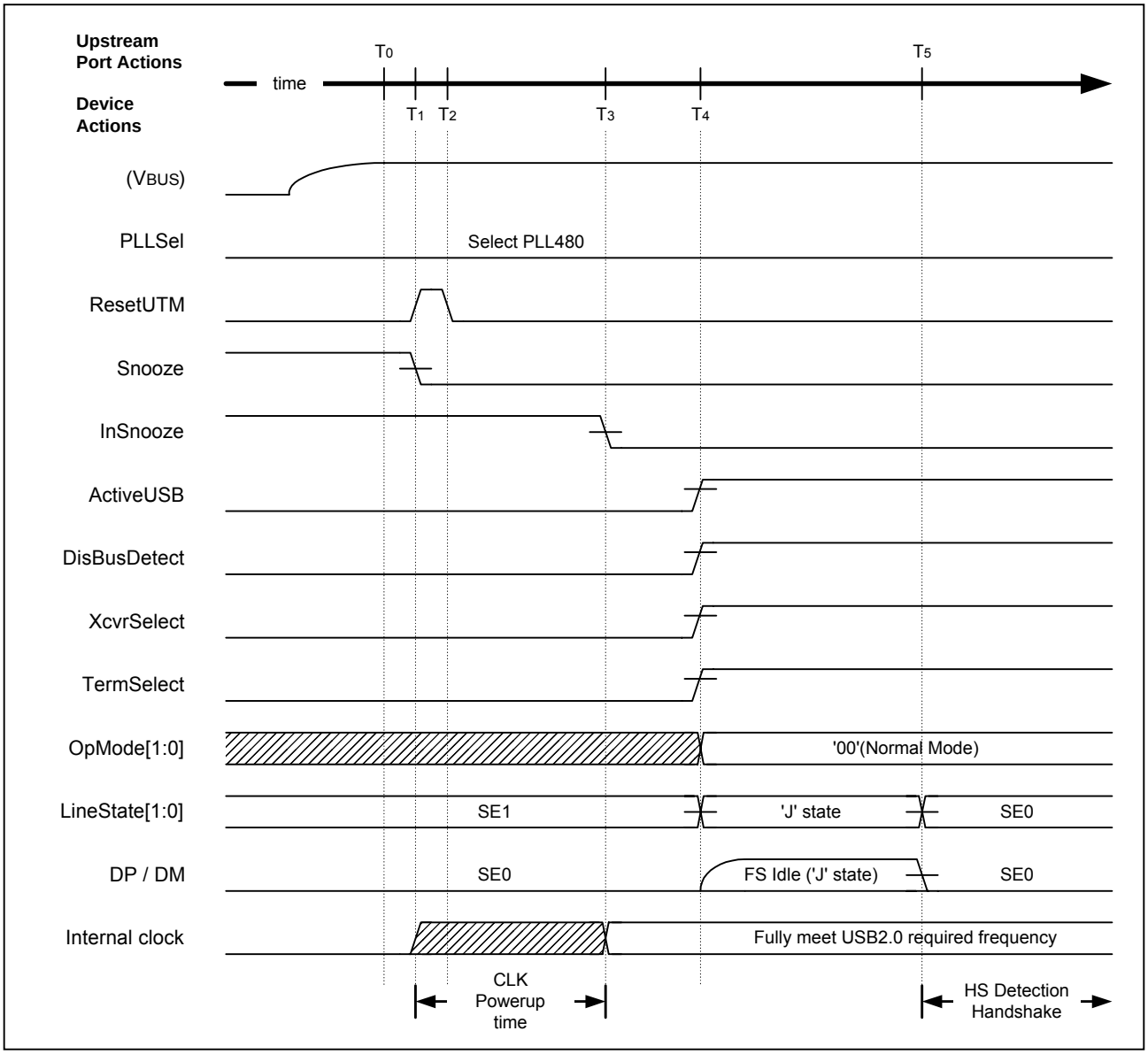



Fig. A.10 Device Attach Timing

Table A.10 Device Attach Timing Values

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | VBUS is valid. | 0 (reference) |
| $T_1$ | ResetUTM is set to '1'. Snooze is cleared to '0'. The internal clock output starts together with ResetUTM. | $T_1$ |
| $T_2$ | ResetUTM is cleared to '0'. | $T_1 + 10ns < T_2$ |
| $T_3$ | InSnooze is set to '0'. The internal clock output becomes stable. | $T_2 + 250\mu s < T_3$ |
| $T_4$ | ActiveUSB is set to '1'. DisBusDetect is set to '1'. TermSelect and XcvrSelect are set to '1'. OpMode[1:0] is set to '00'. The mode moves to the FS mode. The FS termination is valid. | $T_0 + 100ms \{T_{SIGATT}\} < T_4$ |
| $T_5$ | Reset is issued from the downstream port. | $T_4 + 100ms \{T_{ATTDB}\} < T_5$ |

Note: { } is a name standardized in the USB2.0 specifications.

## A.7 Clock

This section describes

1) a recovery method from the case where an oscillator circuit was halted,
2) a method for moving to sleep, and
3) a method for switching a PLL (480 MHz system or 60 MHz system) that operates.

Here, 1) is the processing that is performed in a steady state when the power is on and sleep is released. 2) needs to be aware of extremely because it violates the USB 2.0 specifications depending on an oscillator and an oscillator circuit (built-in) (described later). If this specification time cannot be satisfied, never sleep. 3) aims at further reducing current consumption by switching from a 480 MHz system to a 60 MHz system that is a PLL for HS when a device is connected to the downstream port that does not support HS. The device that builds in sufficient power may not be switched.



### A.7.1 Start of Oscillator Circuit

An oscillator circuit is controlled by the xSLEEP pin and the oscillator circuit is halted when it is asserted. To recover from the sleep state, when the xSLEEP pin is negated ($T_0$), the oscillator circuit starts oscillating. Then the PMControl. Snooze bit is cleared. Once the PMControl.ResetUTM bit is cleared ($T_1$, $T_2$), an internal clock starts being output. Note that the internal clock is not stable. Subsequently, after the OSC Powerup time ($T_3$) and when the PLL Powerup time elapses ($T_4$), the internal clock at the frequency 60 MHz ±500 ppm required in the USB2.0 specifications is output. Here, the PMControl.ResetUTM bit is set once and cleared. However, if this operation is not performed, the internal clock is output ($T_4$) after the PMControl. Snooze bit is cleared and after the PLL Powerup time elapses. However, it cannot be guaranteed that this internal clock conforms to the USB2.0 specifications.

Here, to obtain the internal clock that conforms to the USB2.0 specifications, the CLK Powerup time (OSC Powerup time + PLL Powerup time) are necessary. Among them, the OSC Powerup time requires a unit of several ms according to the conditions of an oscillator and an oscillator circuit. Further, the PLL Powerup time is a very short time as much as about 250 µs in comparison with the oscillator circuit. Accordingly, to obtain a UTM usable clock as quick as possible, this OSC Powerup time needs to be shortened to the utmost. Besides, the UTM usable clock is defined as 60MHz ±10% and the CLK Powerup time is defined as 5.6 ms. Further, until the frequency 60MHz ± 500ppm required in the above USB2.0 specifications is obtained, the time is defined as less than 1.4 ms from this CLK Powerup time.

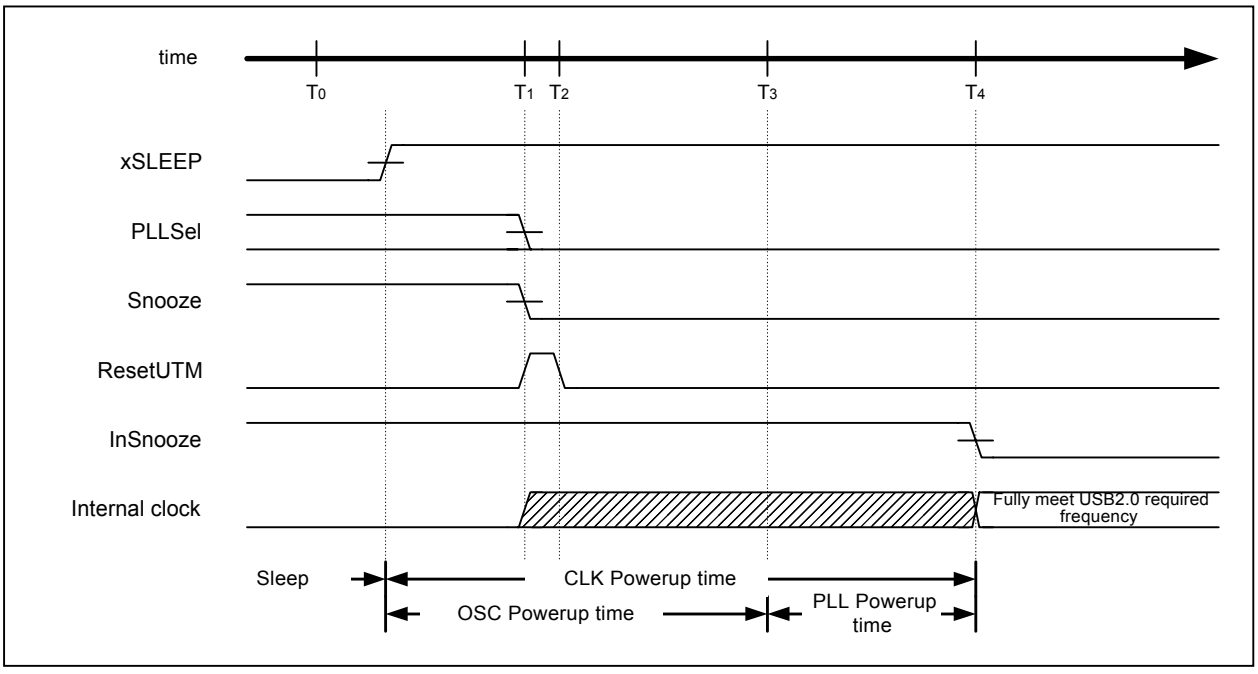



Fig. A.11 OSC-on Timing

Table A.11 OSC-on Timing Values

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | As soon as XSLEEP is negated, an oscillator circuit starts operation. | 0 (reference) |
| $T_1$ | A PLL starts operation.<br>ResetUTM is set to '1' and Snooze is cleared to '0'. | $T_1$ |
| $T_2$ | ResetUTM is cleared to '0'. | $T_1 + 10ns < T_2$ |
| $T_3$ | An oscillator circuit is stable. | user defined |
| $T_4$ | InSnooze is set to '0' and a PLL (internal clock) is stable. | $T_3 + 250\mu s < T_4$ |

### A.7.2 Sleep (Stop of Oscillator Circuit)

The move to the above snooze was controlled only with the PMControl. Snooze bit, but to move to sleep, the xSLEEP pin is controlled further. However, as described in the previous section "Start of Oscillator Circuit", when the oscillator circuit is halted, note that the OSC Powerup time is necessary.

An internal clock is halted by setting the PMControl. Snooze bit ($T_0$). Subsequently, if the xSLEEPpin is asserted when the PMControl.SleepWnb bit is set, the oscillator circuit is halted and enters the sleep state. Subsequently, to recover from this sleep state, first, the xSLEEP pin is negated ($T_2$). After the OSC Powerup time elapses ($T_3$), clear the PMControl. Snooze bit ($T_4$). After the PLL Powerup time elapses, an internal clock is output ($T_5$). This internal clock is set to the frequency 60 MHz ± 500 ppm required in the USB2.0 specifications.



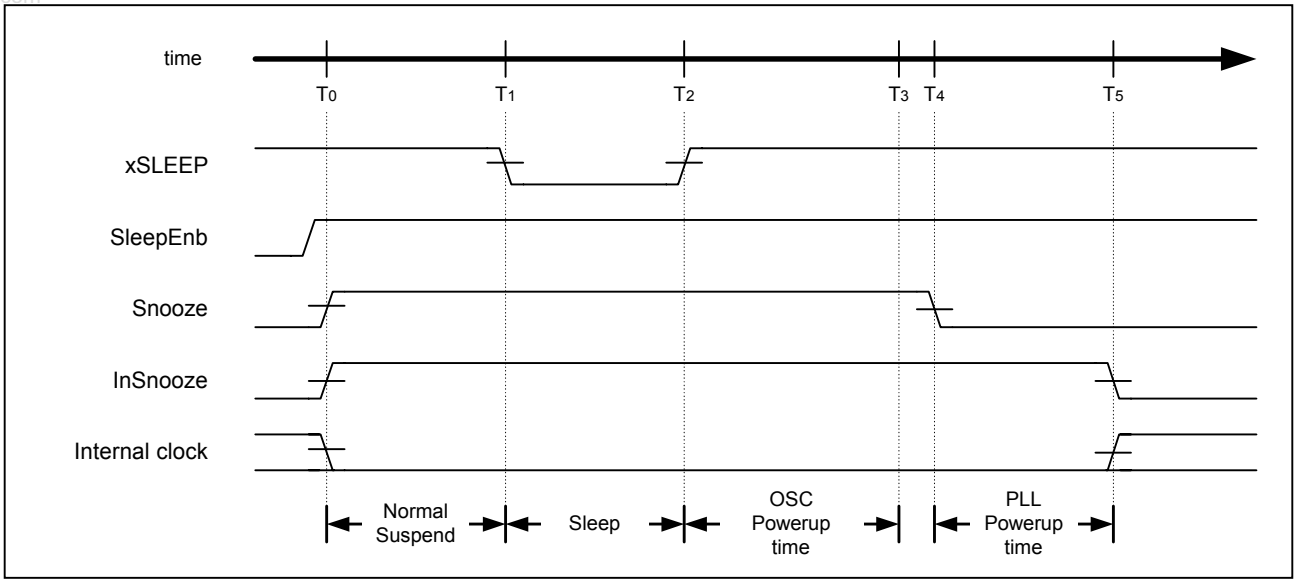



Fig. A.12 Sleep Timing

Table A.12 Sleep Timing Values

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | Snooze is set to '1'. A PLL is halted and the internal clock output is halted. (Before snooze, SleepEnb bit is set to '1'.) | 0 (reference) |
| $T_1$ | The xSLEEP pin is asserted. The oscillator circuit is halted. (Sleep current of 1 mA (Typ.)) | $T_1$ |
| $T_2$ | The xSLEEP pin is negated. The oscillator circuit starts operation. | $T_2$ |
| $T_3$ | An oscillator circuit is stable. | user defined |
| $T_4$ | Snooze is set to '0'. A PLL starts operation. | $T_4$ |
| $T_5$ | A PLL is stable. | $T_4 + 250\mu s < T_5$ |

### A.7.3 PLL Switching

This IC builds in two PLLs of 480 MHz and 60 MHz systems. The 480 MHz system can be used in both the HS and FS modes, but the 60 MHz system can be used in only the FS mode. Because the PLL of this 480 MHz has high current consumption in comparison with the PLL of the 60 MHz system. The current consumption can be suppressed greatly by selecting the 60 MHz system in FS mode.

This section describes the case where the host and hub connected to an upstream port do not support the HS mode and operate in FS mode after HS Detection Handshake terminates. In actual, after a device operates in HS mode using the PLL of the 480 MHz system and then enters the FS mode, the method of switching the PLL of the 480 MHz system to the PLL of the 60 MHz system for the purpose of reducing the current consumption is described.



When the PMControl.PLLSel bit is switched from the 480 MHz system to the 60 MHz system, the PLL of the 60 MHz system is going to start. On this occasion, because the PLL of the 60 MHz system is not operated stably, it cannot be used immediately. Accordingly, after the PLL Powerup time of this 60 MHz system elapses, the internal clock generated from the PLL of the 480 MHz system is halted ($T_1$) and is switched to a clock generated from the 60 MHz system and output ($T_2$). At this time, because the phase of both clocks is checked so that a glitch cannot be loaded on the clocks, a circuit that uses these clocks will not be affected greatly.

Equally, to switch the PMControl.PLLSel bit from the PLL of the 60 MHz system to the PLL of the 480 MHz system, when the PMControl.PLLSel bit is switched from the 60 MHz system to the 480 MHz system, the PLL of the 480 MHz system is going to start ($T_3$). On this occasion, because the PLL of the 480 MHz system is not operated stably, it cannot be used immediately. Accordingly, after the PLL Powerup time of this 480 MHz system elapses, the internal clock generated from the 60 MHz system is halted ($T_4$) and is switched to the clock generated from the 480 MHz system and output ($T_5$). At this time, because the phase of both clocks is checked so that a glitch cannot be loaded on the clocks, a circuit that uses these clocks will not be affected greatly.

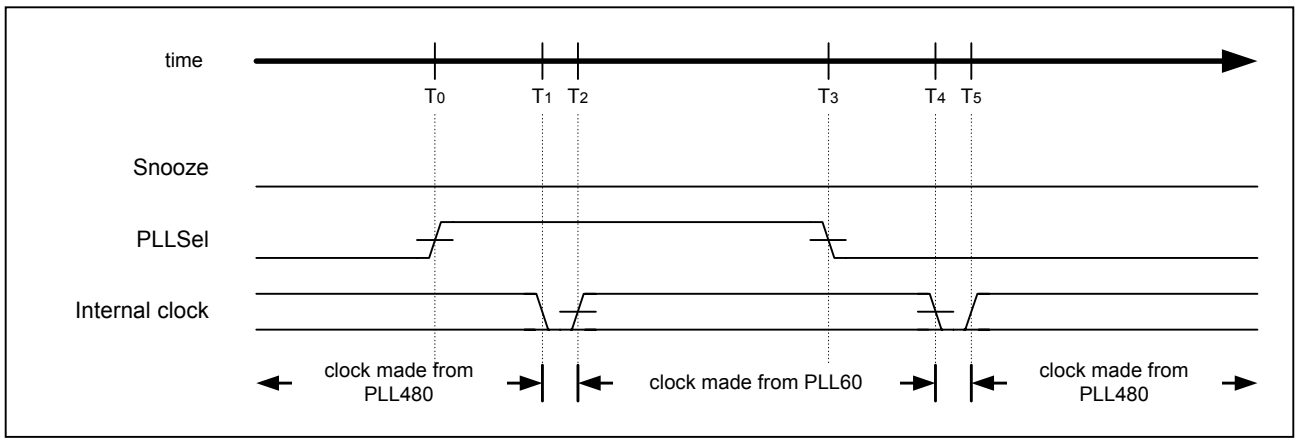



Fig. A.13 Switching PLL Timing

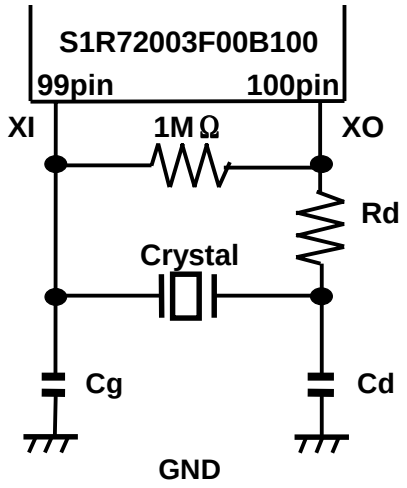
Table A.13 Switching PLL Timing Values

| Timing Parameter | Description | Value |
|---|---|---|
| $T_0$ | PLLSel is switched from PLL480 to PLL60. | 0 (reference) |
| $T_1$ | As soon as PLL60 becomes stable, the internal clock output from PLL480 is halted. | $T_0 + 250\mu s < T_1$ |
| $T_2$ | The internal clock from PLL60 starts being output. | $T_2 < T1 + 50ns$ |
| $T_3$ | PLLSel is switched from PLL60 to PLL480. | $T_3$ |
| $T_4$ | As son as PLL480 becomes stable, the internal clock output from PLL60 is halted. | $T_3 + 250\mu s < T_4$ |
| $T_5$ | The internal clock from PLL480 starts being output. | $T_5 < T_4 + 50ns$ |

# APPENDIX-B. RECOMMENDED OSCILLATOR CIRCUIT

Oscillation characteristics vary according to various conditions (parts used and substrate patterns). Because the following recommended circuit constants satisfy the optimum conditions in the NEC 72003EVA board, use them as reference values.

Determine the oscillation circuit constants after sufficient evaluation.






| Crystal oscillator made by Seiko Epson Corp. ($C_L$=16pF ±50ppm) | Oscillation frequency (MHz) | Recommended circuit constant | | | Voltage range | |
|---|---|---|---|---|---|---|
| | | Cg (pF) | Cd (pF) | Rd (Ω) | Min. (V) | Max. (V) |
| FA-365 | 12,16,20,24 | 22 | 22 | 0 | 3.0 | 3.6 |

# EPSON International Sales Operations

## AMERICA

**EPSON ELECTRONICS AMERICA, INC.**
**HEADQUARTERS**
150 River Oaks Parkway
San Jose, CA 95134, U.S.A.
Phone: +1-408-922-0200 FAX: +1-408-922-0238

**SALES OFFICES**
**West**
1960 E.Grand Avenue
El Segundo, CA 90245, U.S.A.
Phone: +1-310-955-5300 FAX: +1-310-955-5400

**Central**
101 Virginia Street, Suite 290
Crystal Lake, IL 60014, U.S.A.
Phone: +1-815-455-7630 FAX: +1-815-455-7633

**Northeast**
301 Edgewater Place, Suite 120
Wakefield, MA 01880, U.S.A.
Phone: +1-781-246-3600 FAX: +1-781-246-5443

**Southeast**
3010 Royal Blvd. South, Suite 170
Alpharetta, GA 30005, U.S.A.
Phone: +1-877-EEA-0020 FAX: +1-770-777-2637

## EUROPE

**EPSON EUROPE ELECTRONICS GmbH**
**HEADQUARTERS**
Riesstrasse 15
80992 Munich, GERMANY
Phone: +49-(0)89-14005-0 FAX: +49-(0)89-14005-110

**SALES OFFICE**
Altstadtstrasse 176
51379 Leverkusen, GERMANY
Phone: +49-(0)2171-5045-0 FAX: +49-(0)2171-5045-10

**UK BRANCH OFFICE**
Unit 2.4, Doncastle House, Doncastle Road
Bracknell, Berkshire RG12 8PE, ENGLAND
Phone: +44-(0)1344-381700 FAX: +44-(0)1344-381701

**FRENCH BRANCH OFFICE**
1 Avenue de l' Atlantique, LP 915 Les Conquerants
Z.A. de Courtaboeuf 2, F-91976 Les Ulis Cedex, FRANCE
Phone: +33-(0)1-64862350 FAX: +33-(0)1-64862355

**BARCELONA BRANCH OFFICE**
**Barcelona Design Center**
Edificio Testa
Avda. Alcalde Barrils num. 64-68
E-08190 Sant Cugat del Vallès, SPAIN
Phone: +34-93-544-2490 FAX: +34-93-544-2491

## ASIA

**EPSON (CHINA) CO., LTD.**
23F, Beijing Silver Tower 2# North RD DongSanHuan
ChaoYang District, Beijing, CHINA
Phone: 64106655 FAX: 64107319

**SHANGHAI BRANCH**
4F, Bldg., 27, No. 69, Gui Qing Road
Caohejing, Shanghai, CHINA
Phone: 21-6485-5552 FAX: 21-6485-0775

**EPSON HONG KONG LTD.**
20/F., Harbour Centre, 25 Harbour Road
Wanchai, Hong Kong
Phone: +852-2585-4600 FAX: +852-2827-4346
Telex: 65542 EPSCO HX

**EPSON TAIWAN TECHNOLOGY & TRADING LTD.**
10F, No. 287, Nanking East Road, Sec. 3
Taipei
Phone: 02-2717-7360 FAX: 02-2712-9164
Telex: 24444 EPSONTB

**HSINCHU OFFICE**
13F-3, No. 295, Kuang-Fu Road, Sec. 2
HsinChu 300
Phone: 03-573-9900 FAX: 03-573-9169

**EPSON SINGAPORE PTE., LTD.**
No. 1 Temasek Avenue, #36-00
Millenia Tower, SINGAPORE 039192
Phone: +65-337-7911 FAX: +65-334-2716

**SEIKO EPSON CORPORATION**
**KOREA OFFICE**
50F, KLI 63 Bldg., 60 Yoido-dong
Youngdeungpo-Ku, Seoul, 150-763, KOREA
Phone: 02-784-6027 FAX: 02-767-3677

**SEIKO EPSON CORPORATION**
**ELECTRONIC DEVICES MARKETING DIVISION**

**Electronic Device Marketing Department**
**IC Marketing & Engineering Group**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5816 FAX: +81-(0)42-587-5624

**ED International Marketing Department**
**Europe & U.S.A.**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5812 FAX: +81-(0)42-587-5564

**ED International Marketing Department**
**Asia**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-(0)42-587-5814 FAX: +81-(0)42-587-5110


In pursuit of **“Saving” Technology**, Epson electronic devices.
Our lineup of semiconductors, liquid crystal displays and quartz devices
assists in creating the products of our customers’ dreams.
**Epson IS energy savings.**

**S1R72003**
Technical Manual



**SEIKO EPSON CORPORATION**
ELECTRONIC DEVICES MARKETING DIVISION

■ EPSON Electronic Devices Website

http://www.epsondevice.com/



First issue December, 2001
Printed March, 2002 in Japan Ⓗ A