EPSON
EXCEED YOUR VISION

USB2.0 High Speed Device Controller LSI with IDE

S1R72V03

Technical Manual

SEIKO EPSON CORPORATION

Rev.1.5

NOTICE

General Rules

Scope of Application

This specification applies to the USB2.0 Controller “S1R72V03” manufactured by the Semiconductor Operations Division of Seiko Epson Corporation.

Table of Contents

Chapter 1 Overview

The S1R72V03** is a USB device controller LSI that supports USB2.0-compliant high-speed mode. Its numerous advanced features make it suitable for the portable equipment that comes with an IDE I/F and incorporates an HDD.

Chapter 2 Features

<< USB2.0 device functions >>

- Supports HS (480 Mbps) and FS (12 Mbps) transfers
- Contains FS/HS terminations (eliminating the need for external circuits)
- VBUS 5-V input
- Supports control, bulk, and interrupt transfers
- Supports two Bulk transfer lines, one Interrupt transfer line, and Endpoint 0
- Allocates a 1,024-byte FIFO area for each Bulk transfer and a 64-byte FIFO area for each Interrupt and Control transfer.

<< CPU I/F >>

- Accommodates a 16-bit- or 8-bit-wide general-purpose CPU interface
- Incorporates two DMA channels (Multi-word procedure)
- Big endian (incorporating a bus swap function to support little-endian CPUs)
- Variable interface voltages (3.3 V or 1.8 V)

<< IDE I/F >>

- Supports ATA/ATAPI6
  PIO modes 0–4, Multi-word DMA, and UDMA modes 0–5

<< Other >>

- Accommodates a 12-MHz or 24-MHz crystal resonator for clock input (The LSI incorporates an oscillator circuit and a 1-MΩ feedback resistor.)
- Triple-power supply system: 3.3 V, 1.8 V and variable CPU interface power
- Package type: QFP15-128 or PFBGA7UX100
- Guaranteed operating temperature range: –40° C to 85° C

* Not designed to resist radiation

Chapter 3 Block Diagram

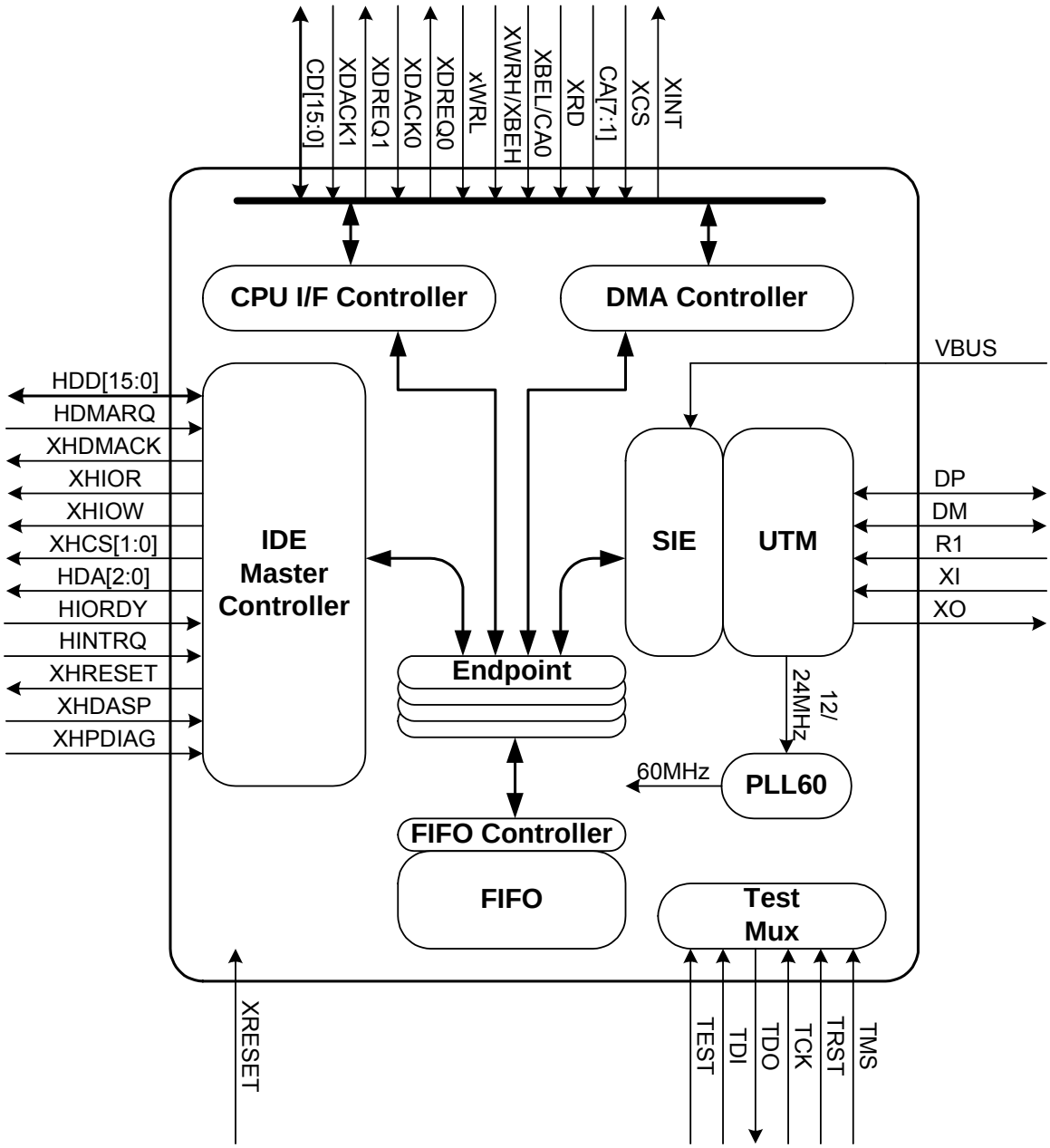


Figure 3-1 General Block Diagram

3.1 USB Transceiver Macro (UTM)

This is a UTMI1.03-compliant USB2.0 transceiver macro. Equipped with an analog circuit and a high-speed logic circuit, it supports HS mode (480 Mbps) and FS mode (12 Mbps).

It incorporates the transmitter, receiver, termination, and squelch circuit for HS/FS, as well as an elasticity buffer, serial-parallel/parallel-serial conversion circuit, bit stuff/unstuff circuit, and a SYNC/EOP-adding/eliminating circuit, which together comprise a USB interface.

Furthermore, its built-in PLL generates the 480-MHz clock needed for USB transfers. Internal oscillator is the clock source of the PLL.

3.2 PLL60

It generates the 60-MHz clock needed for operation of the internal logic. The oscillator circuit accommodates a 12-MHz or 24-MHz crystal resonator for its input clock.

3.3 Serial Interface Engine (SIE)

This manages transactions and generates packets. Furthermore, it controls bus events such as suspend, resume, and reset.

3.4 FIFO and FIFO Controller

This is a buffer (2.5 kB) for endpoints.

3.5 CPU I/F Controller

This controls the timing of the CPU interface to allow for register access.

3.6 DMA Controller

This controls the DMA timing of the CPU interface to allow for FIFO access. The LSI contains two DMA channels.

3.7 IDE Master Controller

This is the IDE interface for ATA/ATAPI6. It supports PIO modes 0–4, Multi-word DMA, and UDMA modes 0–5.

3.8 TestMUX

This is a test circuit.

Chapter 4 Pin Layout Diagram

4.1 Pin Layout Diagram (QFP Package)

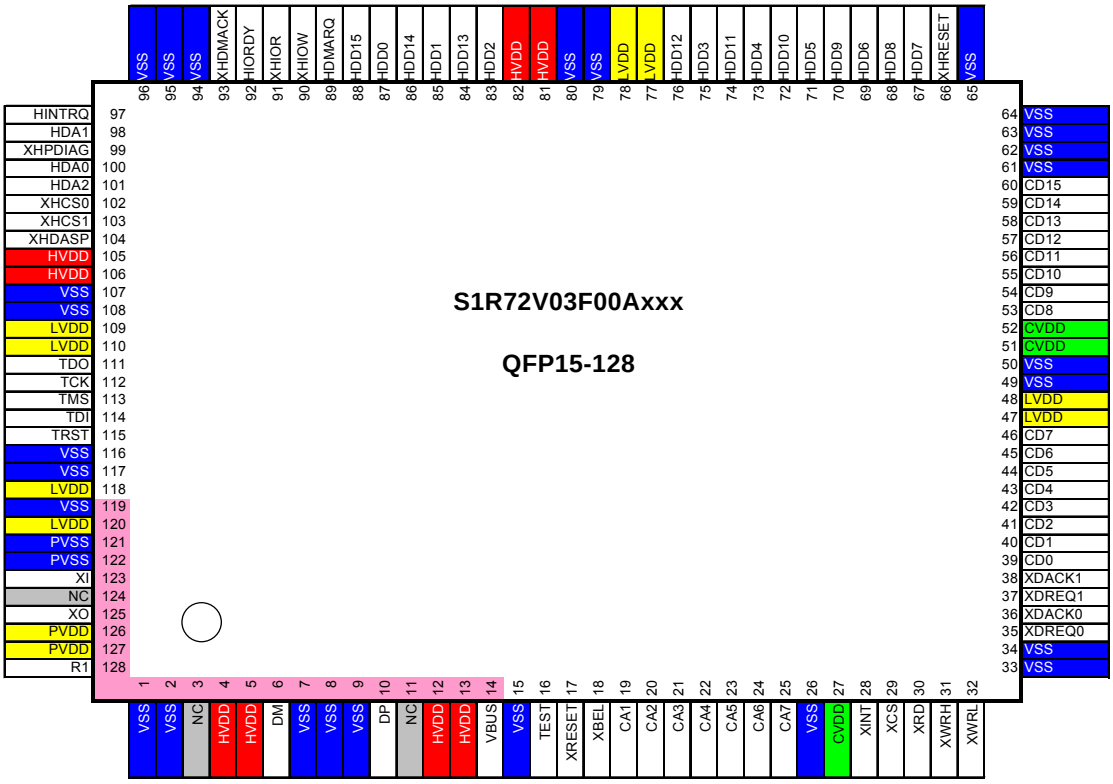


Figure 4-1 Pin Layout Diagram of the QFP Package

4.2 Pin Layout Diagram (BGA Package)

PFBGA7UX100

	1	2	3	4	5	6	7	8	9	10	
A	NC	R1	PVDD	XO	XI	LVDD	LVDD	HVDD	HDA0	NC	A
B	HVDD	VSS	PVSS	LVDD	VSS	TCK	XHDASP	HDA2	HINTRQ	XHDMACK	B
C	DM	VSS	VSS	TRST	TMS	VSS	XHCS0	HDA1	XHIOW	HIORDY	C
D	DP	VBUS	XRESET	TDI	TDO	XHCS1	XHPDIAG	XHIOR	HDD15	HDD14	D
E	HVDD	TEST	XBEL	CA3	CD1	CD3	HDMARQ	HDD1	HDD13	HVDD	E
F	CA2	CA1	CA6	CA5	CD0	VSS	HDD0	HDD2	HDD3	HDD12	F
G	CA7	CA4	XWRH	XDREQ0	CD5	CD10	CD11	HDD11	HDD4	HDD10	G
H	CVDD	XCS	XWRL	XDREQ1	CD6	CD9	CD14	HDD5	HDD9	HDD8	H
J	XINT	XRD	XDACK0	CD2	CD7	CD8	CD13	VSS	HDD6	XHRESET	J
K	NC	VSS	XDACK1	CD4	LVDD	CVDD	CD12	CD15	HDD7	NC	K
	1	2	3	4	5	6	7	8	9	10	

Figure 4-2 Pin Layout Diagram of the BGA Package (Top View)

Chapter 5 Pin Description

GENERAL

Pin	Ball	Name	I/O	RESET	Pin Type	Pin Description
17	D3	XRESET	IN	-	-	Reset signal
123	A5	XI	IN	-	Analog	Input to the internal oscillator circuit 12 MHz/24 MHz
125	A4	XO	OUT	-	Analog	Output from the internal oscillator circuit

XRESET is part of the CVDD power supply system.

XI and XO are parts of the PVDD power supply system.

TEST

Pin	Ball	Name	I/O	RESET	Pin Type	Pin Description
16	E2	TEST	IN	-		Test pin (fixed low)
111	D5	TDO	OUT	Hi-Z	2mA	JTAG TDO pin
112	B6	TCK	IN	-		JTAG TCK pin
113	C5	TMS	IN	-		JTAG TMS pin
114	D4	TDI	IN	-		JTAG TDI pin
115	C4	TRST	IN	-		JTAG TRST pin

TEST is part of the CVDD power supply system.

The other pins listed above are parts of the PVDD power supply system.

When the JTAG function is unused, fix the TEST, TCK, TMS, TDI, and TRST pins low and leave the TDO pin open.

PD : Pull Down, PU : Pull Up

USB

Pin	Ball	Name	I/O	RESET	Pin Type	Pin Description
128	A2	R1	IN	-	Analog	Internal operation setup pin. A 6.2-kΩ ± 1% resistor is connected between this pin and the VSS.
10	D1	DP	BI	Hi-Z	Analog	USB device data line, Data+
6	C1	DM	BI	Hi-Z	Analog	USB device data line, Data–
14	D2	VBUS	IN(PD)	(PD)	Analog	USB device bus detection signal

CPU I/F

Pin	Ball	Name	I/O	RESET	Pin Type	Pin Description		
				Bus	Mode ⇒	16bit Strobe mode	16bit BE mode	8bit mode
30	J2	XRD	IN	-	-	Read strobe		
32	H3	XWRL (XWR)	IN	-	-	Write strobe, lower	Write strobe	
31	G3	XWRH (XBEH)	IN	-	-	Write strobe, upper	High byte enable	Fixed high
29	H2	XCS	IN	-	-	Chip select signal		
28	J1	XINT	OUT	High	2 mA Tristate	Interrupt output signal		
35	G4	XDREQ0	OUT	High	2mA	DMA0 request		
36	J3	XDACK0	IN	-	-	DMA0 acknowledge		
37	H4	XDREQ1	OUT	High	2mA	DMA1 request		
38	K3	XDACK1	IN	-	-	DMA1 acknowledge		
18	E3	XBEL (CA0)	IN	-	-	Fixed high or low	Low byte enable	Address 0
19	F2	CA1	IN	-	-	CPU bus address		
20	F1	CA2	IN	-	-			
21	E4	CA3	IN	-	-			
22	G2	CA4	IN	-	-			
23	F4	CA5	IN	-	-			
24	F3	CA6	IN	-	-			
25	G1	CA7	IN	-	-			
39	F5	CD0	BI	Hi-Z	2mA	CPU data bus		CPU data bus
40	E5	CD1	BI	Hi-Z	2mA			
41	J4	CD2	BI	Hi-Z	2mA			
42	E6	CD3	BI	Hi-Z	2mA			
43	K4	CD4	BI	Hi-Z	2mA			
44	G5	CD5	BI	Hi-Z	2mA			
45	H5	CD6	BI	Hi-Z	2mA			
46	J5	CD7	BI	Hi-Z	2mA			
53	J6	CD8	BI	Hi-Z	2mA			Pull Up or Pull Down
54	H6	CD9	BI	Hi-Z	2mA			
55	G6	CD10	BI	Hi-Z	2mA			
56	G7	CD11	BI	Hi-Z	2mA			
57	K7	CD12	BI	Hi-Z	2mA			
58	J7	CD13	BI	Hi-Z	2mA			
59	H7	CD14	BI	Hi-Z	2mA			
60	K8	CD15	BI	Hi-Z	2mA			

The pins listed above are parts of the CVDD power supply system.

PD: Pull Down

PU: Pull Up

IDE I/F

Pin	Ball	Name	I/O	RESET	Pin Type	Pin Description
101	B8	HDA2	OUT	Hi-Z	4mA	IDE register address
98	C8	HDA1	OUT	Hi-Z	4mA	
100	A9	HDA0	OUT	Hi-Z	4mA	
103	D6	XHCS1	OUT	Hi-Z	4mA	Chip select for control register access
102	C7	XHCS0	OUT	Hi-Z	4mA	Chip select for command-block register access
91	D8	XHIOR	OUT	Hi-Z	4mA	IDE read strobe
90	C9	XHIOW	OUT	Hi-Z	4mA	IDE write strobe
89	E7	HDMARQ	IN (PD)	(PD)	-	DMA transfer request
93	B10	XHDMACK	OUT	Hi-Z	4mA	DMA transfer acknowledge
92	C10	HIORDY	IN (PU)	(PU)	-	IDE register ready signal
97	B9	HINTRQ	IN (PU)	(PU)	-	IDE interrupt request
66	J10	XHRESET	OUT	Hi-Z	4mA	IDE bus reset
104	B7	XHDASP	IN (PU)	(PU)	-	Drive enable/slave drive available
99	D7	XHPDIAG	IN (PU)	(PU)	-	Diagnostic-sequence end signal
88	D9	HDD15	BI (PU)	(PU)	4mA	IDE data bus
86	D10	HDD14	BI (PU)	(PU)	4mA	
84	E9	HDD13	BI (PU)	(PU)	4mA	
76	F10	HDD12	BI (PU)	(PU)	4mA	
74	G8	HDD11	BI (PU)	(PU)	4mA	
72	G10	HDD10	BI (PU)	(PU)	4mA	
70	H9	HDD9	BI (PU)	(PU)	4mA	
68	H10	HDD8	BI (PU)	(PU)	4mA	
67	K9	HDD7	BI (PD)	(PD)	4mA	
69	J9	HDD6	BI (PU)	(PU)	4mA	
71	H8	HDD5	BI (PU)	(PU)	4mA	
73	G9	HDD4	BI (PU)	(PU)	4mA	
75	F9	HDD3	BI (PU)	(PU)	4mA	
83	F8	HDD2	BI (PU)	(PU)	4mA	
85	E8	HDD1	BI (PU)	(PU)	4mA	
87	F7	HDD0	BI (PU)	(PU)	4mA	

The pins listed above are parts of to the HVDD power supply system.

PD: Pull Down

PU: Pull Up

POWER

Pin	Ball	Name	Voltage	Pin Description
4, 5, 12, 13, 81, 82, 105, 106	B1, E1, A8, E10	HVDD	3.3V	Power supply for the IDE and USB I/O
27, 51, 52	K6, H1	CVDD	3.3V/1.8V	Power supply for the CPU I/O
47, 48, 77, 78, 109, 110, 118, 120	A7, A6, B4, K5	LVDD	1.8V	Internal power supply
126, 127	A3	PVDD	1.8V	Power supply for the PLL
1, 2, 7, 8, 9, 15, 26, 33, 34,49, 50, 61, 62, 63, 64, 65, 79, 80, 94, 95, 96, 107, 108, 116, 117, 119	B2, C2, K2, C3, B5, C6, F6, J8	VSS	0V	GND
121, 122	B3	PVSS	0V	GND for the PLL
3, 11, 124	A1, K1, A10, K10	NC	0V	NC (Must be fixed to GND)

Chapter 6 Functional Description

This section describes the operation of the LSI. In the explanation below, the registers are described according to the following naming conventions.

- Names indicating a register comprising one address
  Register name + register
  Example: “MainIntStat register”
- Names indicating individual register bits
  Register name.bit name + bit, or bit name + bit
  Example: “MainIntStat.RcvEP0SETUP bit” or “ForceNAK bit in the EP0ControlOUT register”
- Registers provided for each endpoint
  EPx{x=0, a-c} XXX register, EPx{x=a-c} YYY register
  Example: “EPx{x=0, a-c}IntStat register” or “EPx{x=a-c}Control register”
- Registers provided for each DMA channel
  DMAx {x=0, 1} ZZZ register
  Example: “DMAx{x=0, 1}Config register” or “EPx{x=0, a-c}Join.JoinDMAx{x=0, 1}_Wr bit”

6.1 USB Device Control

The following describes the USB device functions.

6.1.1 Endpoints

The LSI stipulated herein has an endpoint for control transfer (EP0) and three general-purpose endpoints (EPa, EPb, and EPc). Each of the endpoints EPa, EPb, and EPc, can be used as the endpoint for bulk or interrupt transfer.

The hardware of the LSI provides endpoints for the purpose of transaction management. However, it does not provide management functions for the interfaces defined in the USB standard (hereinafter referred to as the “USB-defined interface”). The USB-defined interface should be implemented by the user firmware. Set up and combine endpoints as appropriate for the descriptor definitions specific to the device, in order to configure the USB-defined interface.

Each endpoint has fixed basic setup items determined by the USB-defined interface and variable control items and status to be controlled for each transfer. The basic setup items should be set up when the chip is initialized or the USB-defined interfaces are changed.

Table 6-1 lists the basic setup items for the endpoint EP0 (default control pipe).

The endpoint EP0 shares the register set and FIFO area for transfers in the IN and OUT directions. The direction of data transaction should be set by the firmware as appropriate for execution of the data and status stages at the endpoint EP0.

Table 6-1 Basic Setup Items of the Endpoint EP0

Item	Register/Bit	Description
Max packet size	EP0MaxSize	Sets the value for the max. packet size to 8, 16, 32, or 64 during FS operation, or to 64 during HS operation. A 64-byte area beginning with the FIFO address 0 is allocated for the endpoint EP0.

Table 6-2 lists the basic setup items for the general-purpose endpoints (EPa, EPb, and EPc). As the endpoints EPa, EPb, and EPc enable the transaction direction and endpoint number to be set as desired, up to three independent endpoints can be used. Set up endpoints as appropriate for the contents of definitions of the USB-defined interface, and enable the set endpoints as necessary to configure the USB-defined interface.

The endpoint EPa has 64 bytes of the FIFO area allocated for it, and the endpoints EPb and EPc respectively have 1,024 bytes of the FIFO area allocated for them.

Table 6-2 Basic Setup Items of the General-Purpose Endpoints

Item	Register/Bit	Description
Transaction direction	EPx{x=a-c}Config_0.INxOUT	Sets the direction of transfer at each endpoint.
Max packet size	EPx{x=b-c}MaxSize_H, EPx{x=a-c}MaxSize_L	Sets the max. packet size for each endpoint to 8, 16, 32, 64, or 512 bytes. For the endpoints at which bulk transfer is to be performed, however, set the max. packet size to 8, 16, 32, or 64 bytes in FS mode, or to 512 bytes in HS mode. (Note: The max. packet size that can be set for EPa is limited to 64 bytes.)
Endpoint number	EPx{x=a-c}Config_0.EndpointNumber	Sets the endpoint number for each endpoint to any value between 0x1 to 0xF.
Toggle mode	EPx{x=a-c}Config_0.IntEP_Mode	Sets the operation mode of interrupt transfer. For the endpoints at which bulk transfer is to be performed, always set to 0 irrespective of the transaction direction. For the IN-direction endpoints, set the mode of the toggle sequence. For the OUT-direction endpoints, if interrupt transfer is to be performed, always set to 0.
Endpoint enable	EPx{x=a-c}Config_0.EnEndpoint	Enables each endpoint. Make this setting for any endpoint when the USB-defined interface that will use that endpoint is enabled.

6.1.2 Transactions

The LSI provides transaction execution functions and the interface needed for the firmware to execute transactions, both in the hardware. The interface for the firmware is implemented as an interrupt signal that is asserted by a control register, status register, or status. For settings concerning the assertion of an interrupt by status, refer to the relevant section of this specification in which the registers are described.

The LSI sends the status to the firmware for each transaction performed. However, the firmware does not always need to manage each individual transaction. When responding to a transaction request, the LSI inspects the FIFO to determine the data quantity or free space in it in order to determine whether the data transfer can be performed, for the automatic processing of transactions.

For an OUT endpoint, for example, the firmware reads the data out of the FIFO through the CPU interface (DMA read or register read) or the IDE interface (IDE write) to create free space in the FIFO, thereby allowing OUT transactions to be automatically executed in succession. For an IN endpoint as well, the firmware writes data to the FIFO through the CPU interface (DMA write or register write) or the IDE interface (IDE read) in order to create valid data in the FIFO, thereby allowing IN transactions to be automatically executed in succession.

Table 6-3 lists the control items and status relating to the transaction control for the endpoint EP0.

Table 6-3 Endpoint EP0 Control Items and Status

Item	Register/Bit	Description
Transaction direction	EP0Control.INxOUT	Sets the direction of transfer in the data and status stages.
Descriptor reply enable	EP0Control.ReplyDescriptor	Invokes an automatic descriptor response.
Descriptor reply address	DescAdrs_H, DescAdrs_L	Specifies the start address in FIFO of the data to be returned by automatic descriptor response.
Descriptor size	DescSize_H, DescSize_L	Specifies the data quantity to be returned by automatic descriptor response.
Control protect	SETUP_Control.ProtectEP0	When this bit is set, the ForceNAK and ForceSTALL bits in the EP0ControlIN and EP0ControlOUT registers are disabled against access. This bit is set in the hardware by the LSI when the RcvEP0SETUP status is turned on, and can be cleared by a register access by the CPU.
Short-packet transmit enable	EP0ControlIN.EnShortPkt	Enables transmission of short packets less than the max. packet size. This bit is cleared when an IN transaction that transmitted a short packet is completed.
Toggle sequence bit	EP0ControlIN.ToggleStat, EP0ControlOUT.ToggleStat	Indicates the status of the toggle sequence bits. These bits are automatically initialized by the SETUP stage.

Toggle set	EP0ControlIN.ToggleSet, EP0ControlOUT.ToggleSet	Sets the toggle sequence bits.
Toggle clear	EP0ControlIN.ToggleClr, EP0ControlOUT.ToggleClr	Clears the toggle sequence bits.
Forced NAK response	EP0ControlIN.ForceNAK, EP0ControlOUT.ForceNAK	Always responds with NAK for IN or OUT (including PING) transactions, irrespective of the data quantity and free space in the FIFO.
STALL response	EP0ControlIN.ForceSTALL, EP0ControlOUT.ForceSTALL	Responds with STALL for IN or OUT (including PING) transactions.
Automatic ForceNAK set	EP0ControlOUT.AutoForceNAK	Sets the EP0ControlOUT.ForceNAK bit each time an OUT transaction is completed.
SETUP receive status	MainIntStat.RcvEP0SETUP	Indicates that a SETUP transaction has been executed.
Transaction status	EP0IntStat.OUT_ShortACK, EP0IntStat.IN_TranACK, EP0IntStat.OUT_TranACK, EP0IntStat.IN_TranNAK, EP0IntStat.OUT_TranNAK, EP0IntStat.IN_TranErr, EP0IntStat.OUT_TranErr	Indicates the result of a transaction.
Descriptor reply data stage end status	FIFO_IntStat.DescriptorCmp	Indicates that the data stage for automatic descriptor response has ended.

Table 6-4 lists the control items and status relating to transaction processing for the general-purpose endpoints EPa, EPb, and EPc.

Table 6-4 General-Purpose Endpoint Control Items and Status

Item	Register/Bit	Description
Automatic ForceNAK set	EPx{x=a-c}Control.AutoForceNAK	Sets the EPx{x=a-c}Control.ForceNAK bit for an endpoint each time an OUT transaction at that endpoint is completed.
Short packet transmit enable	EPx{x=a-c}Control.EnShortPkt	Enables the transmission of short packets less than the max. packet size for the IN transaction. This bit is cleared when an IN transaction that transmitted a short packet is completed.
Automatic ForceNAK set by short-packet reception disable	EPx{x=a-c}Control. DisAF_NAK_Short	Disables the function(*) for automatically setting the EPx{x=a-c}Control.ForceNAK bit for an endpoint when a short packet is received for that endpoint in an OUT transaction. *: This function remains enabled unless it is disabled by this bit.
Toggle sequence bit	EPx{x=a-c}Control.ToggleStat	Indicates the status of the toggle sequence bit.
Toggle set	EPx{x=a-c}Control.ToggleSet	Sets the toggle sequence bit.

Toggle clear	EPx{x=a-c}Control.ToggleClr	Clears the toggle sequence bit.
Forced NAK response	EPx{x=a-c}Control.ForceNAK	Always responds with NAK for transactions, irrespective of the data quantity and free space in the FIFO.
STALL response	EPx{x=a-c}Control.ForceSTALL	Responds with STALL for a transaction.
Transaction status	EPx{x=a-c}IntStat.OUT_ShortACK, EPx{x=a-c}IntStat.IN_TranACK, EPx{x=a-c}IntStat.OUT_TranACK, EPx{x=a-c}IntStat.IN_TranNAK, EPx{x=a-c}IntStat.OUT_TranNAK, EPx{x=a-c}IntStat.IN_TranErr, EPx{x=a-c}I~tStat.OUT_TranErr	Indicates the result of a transaction.

6.1.2.1 SETUP Transactions

SETUP transactions addressed to the endpoint EP0 of the local node are unconditionally executed. (USB functions must be enabled by the USB_Control.ActiveUSB bit before this can occur.)

When a SETUP transaction is issued, the LSI stores the entire contents of the data packet (8 bytes) in the EP0SETUP_0 through EP0SETUP_7 registers, and then returns an ACK response. Furthermore, except for SetAddress() requests, the LSI sends the RcvEP0SETUP status to the firmware.

If an error occurs during a SETUP transaction, the LSI does not return a response, nor does it issue status.

When a SETUP transaction is completed, the LSI sets the ForceNAK bit and clears the ForceSTALL bit in the EP0ControlIN and EP0ControlOUT registers. It also sets the ToggleStat bit. Furthermore, it sets the SETUP_Control.ProtectEP0 bit. When the firmware has finished setting up the endpoint EP0 and is ready to move to the next stage, it should clear the SETUP_Control.ProtectEP0 bit and then the ForceNAK bit in the EP0ControlIN or EP0ControlOUT register for the direction concerned.

Figure 6-1 shows how a SETUP transaction will be performed. In (a), the host issues a SETUP token addressed to the endpoint EP0 of this node. In (b), the host continues to send an 8-byte-long data packet. The LSI writes this data to the EP0SETUP_0 to EP0SETUP_7 registers. In (c), the LSI automatically returns an ACK response. Furthermore, it sets up the registers to be automatically set, and sends the status to the firmware.

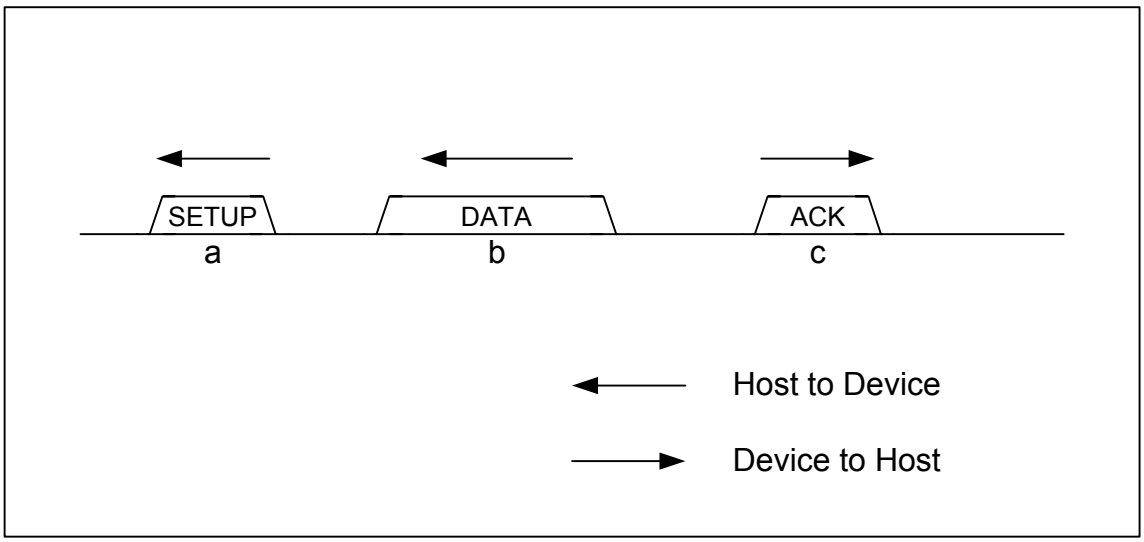


Figure 6-1 SETUP Transaction

6.1.2.2 Bulk/Interrupt OUT Transactions

In a bulk/interrupt OUT transaction, the LSI starts receiving data if the FIFO has free space greater than the max. packet size.

When all bytes of data have been received normally in a bulk/interrupt OUT transaction, the LSI completes the transaction and returns an ACK or a NYET response. Furthermore, it transmits the OUT_TranACK status for the relevant endpoint (EPx{x=0, a-c}IntStat.OUT_TranACK bit) to the firmware. It also updates the FIFO and, assuming the data to have been received, reserves a storage area.

Furthermore, when all data bytes of a short packet have been received in a bulk/interrupt OUT transaction, the LSI transmits] the OUT_ShortACK status (EPx{x=0, a-c}IntStat.OUT_ShortACK bit), in addition to the transaction-complete processing described above. Furthermore, if the EPx{x=0, a-c}Control.DisAF_NAK_Short bit is cleared, the LSI sets the EPx{x=a-c}ForceNAK bit for the endpoint.

If a toggle mismatch occurs in a bulk/interrupt OUT transaction, the LSI responds with ACK for the transaction but does not transmit the status. In such a case, the FIFO is not updated.

If an error occurs in a bulk/interrupt OUT transaction, the LSI does not return a response for the transaction. It also transmits the OUT_TranErr status (EPx{x=0, a-c}IntStat.OUT_TranErr bit). In such a case, the FIFO is not updated.

If all bytes of data could not be received in a bulk/interrupt OUT transaction, the LSI responds with NAK for the transaction. It also transmits the OUT_TranNAK status (EPx{x=0, a-c}IntStat.OUT_TranNAK bit). In such a case, the FIFO is not updated.

Figure 6-2 shows how a bulk or interrupt OUT transaction will be performed in cases in which the transaction is completed normally. In (a), the host issues an OUT token addressed to the OUT-direction endpoint present in this node. In (b), the host continues to send a data packet within the max. packet size. The LSI writes this data to the FIFO for the relevant endpoint. In (c), the LSI automatically returns an ACK response when it successfully received the data. It also sets up the registers to be automatically set, and sends the status to the firmware.

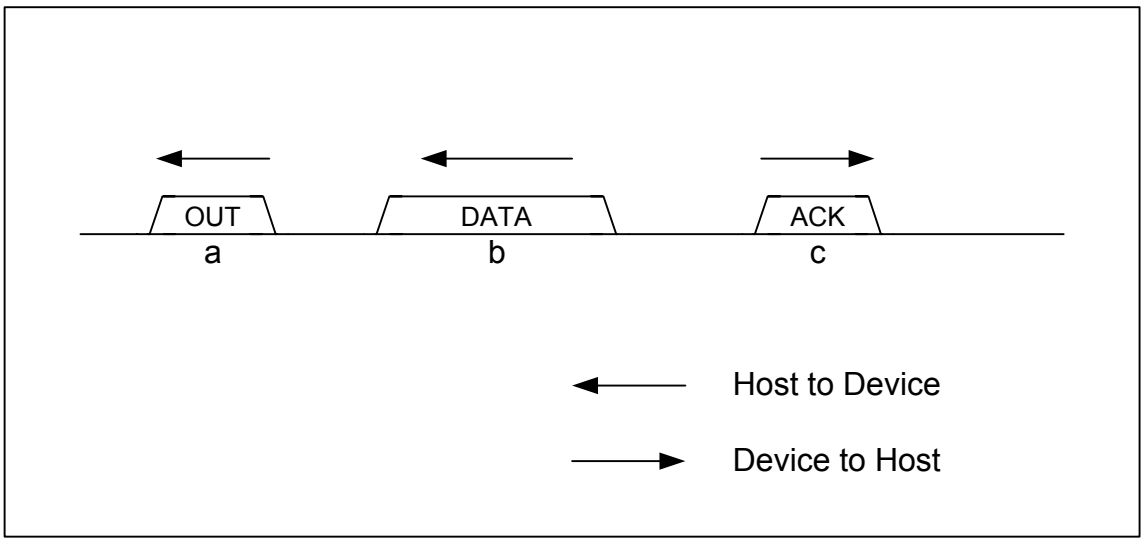


Figure 6-2 OUT Transaction

6.1.2.3 Bulk/Interrupt IN Transactions

When the FIFO has a quantity of data equivalent to the max. packet size for an IN-direction bulk/interrupt endpoint, or when short-packet transmission for that endpoint has been enabled by the firmware, the LSI sends back a data packet in response to the IN transaction.

Transmission of short packets (including those with data length = 0) is enabled by setting the EP0ControlIN.EnShortPkt bit or EPx{x=a-c}Control.EnShortPkt bit. When transmitting short packets, make sure no new data will be written to the FIFO for the endpoint concerned before the transaction is completed after packet transmission has been enabled.

For the endpoint EP0, when the IN transaction for transmitting a short packet is completed, the EP0ControlIN.ForceNAK bit is set.

When ACK is received in the IN transaction by which data was sent back to the host, the LSI completes the transaction and sends the IN_TranACK status (EPx{x=0, a-c}IntStat.IN_TranACK bit) to the firmware. It also updates the FIFO and, assuming the transmitted data to have been transmitted, frees the storage area.

If ACK is not received in the IN transaction by which data was sent back to the host, the LSI assumes that the transaction has failed and sends the IN_TranErr status (EPx{x=0, a-c}IntStat.IN_TranErr bit) to the firmware. It does not update the FIFO, nor does it free the storage area.

When the FIFO does not have a quantity of data equivalent to the max. packet size for an IN-direction bulk/interrupt endpoint, or short-packet transmission for that endpoint has not been enabled by the firmware, the LSI responds with NAK for the IN transaction and sends the IN_TranNAK status (EPx{x=0, a-c}IntStat.IN_TranNAK bit) to the firmware. It does not update the FIFO, nor does it free the storage area.

Figure 6-3 shows how a bulk or interrupt IN transaction will be performed in cases in which the transaction is completed normally. In (a), the host issues an IN token addressed to the IN-direction endpoint present in this node. In (b), if the LSI can respond to this IN transaction, it transmits a data packet that does not exceed the max. packet size. In (c), the host responds with ACK. When the LSI receives ACK, it sets up the registers to be automatically set and sends the status to the firmware.

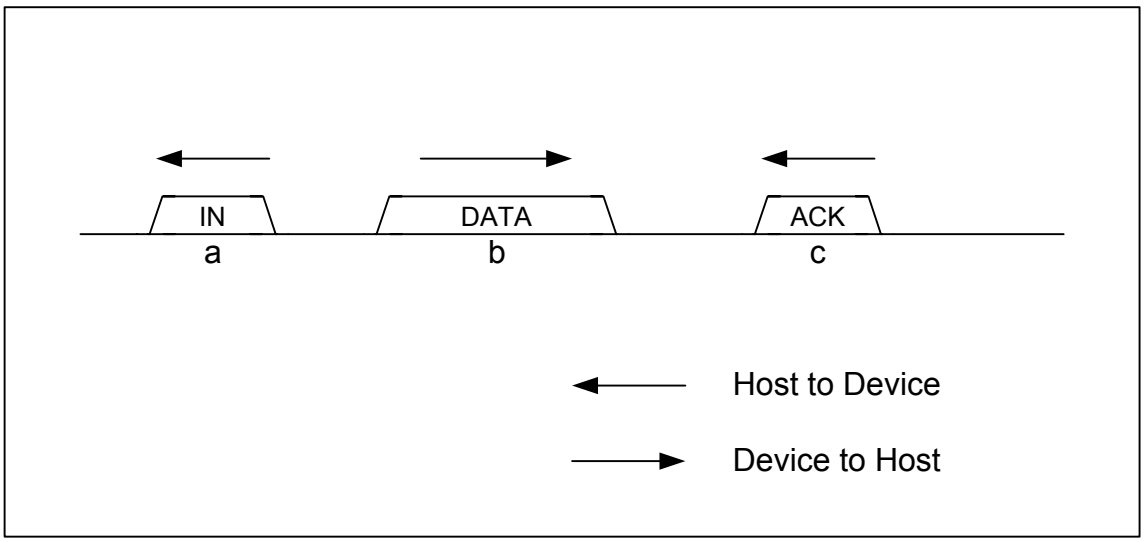


Figure 6-3 IN Transaction

6.1.2.4 PING Transactions

At bulk OUT-direction endpoints, a PING transaction is executed during operation in HS mode.

If the FIFO for the relevant endpoint has free space greater than the max. packet size, the LSI responds with ACK for the PING transaction. It does not send the status to the firmware, however.

If the FIFO for the relevant endpoint has less free space than the max. packet size, the LSI responds with NAK for the PING transaction. It also sends the OUT_TranNAK status (EPx{x=0, a-c}IntStat.OUT_TranNAK bit) to the firmware.

In no case will the FIFO be updated in PING transactions.

Figure 6-4 shows how a PING transmission will be performed when responded to with ACK. In (a), the host issues a PING token addressed to the OUT-direction endpoint present in this node. In (b), if the FIFO has free space equal to the max. packet size, the LSI responds with ACK for the PING transaction. It also sends the status to the firmware.

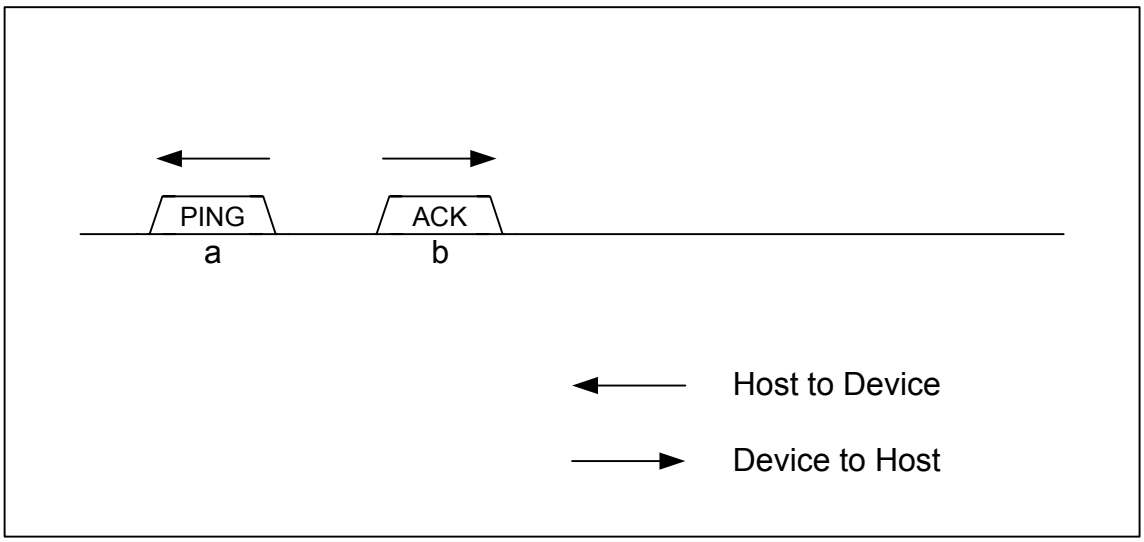


Figure 6-4 PING Transaction

6.1.3 Control Transfers

Control transfers at the endpoint EP0, except for SetAddress() requests, are controlled as a combination of individual transactions. SetAddress() requests are automatically processed using the automatic address setup function, which will be described later.

Figure 6-5 shows how a control transfer will be performed in cases in which the data stage is directed for OUT. In (a), the host starts a control transfer by means of a SETUP transaction. The firmware of the device analyzes the content of the request in order to prepare to respond to a data stage. In (b), the host issues an OUT transaction to execute a data stage, and the device receives data. In (c), the host issues an IN transaction to execute a status stage, and the device sends a packet with a data length of zero back to the host.

For control transfers without data stages, the operation is executed without the data stage described in this example.

Transition to the status stage is accomplished by the host as it issues a transaction for the direction opposite to the data stage. The firmware should monitor the IN_TranNAK status (EP0IntStat.IN_TranNAK bit) to take advantage of an opportunity for transition from the data stage to the status stage.

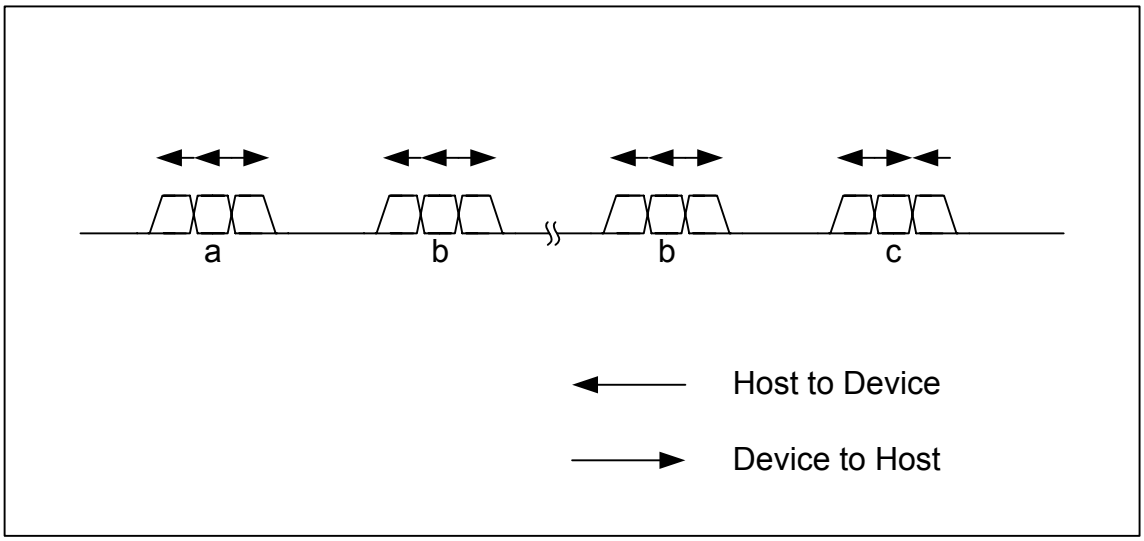


Figure 6-5 Control Transfer when the Data Stage is Directed for OUT

Figure 6-6 shows how a control transfer will be performed when the data stage is directed for IN. In (a), the host starts a control transfer by means of a SETUP transaction. The firmware of the device analyzes the content of the request in order to prepare to respond to a data stage. In (b), the host issues an IN transaction to execute a data stage, and the device transmits data. In (c), the host issues an OUT transaction to execute a status stage, and the device responds to it with ACK.

Transition to the status stage is accomplished by the host as it issues a transaction for the direction opposite to the data stage. The firmware should monitor the OUT_TranNAK status (EP0IntStat.OUT_TranNAK bit) to take advantage of an opportunity for transition from the data stage to the status stage.

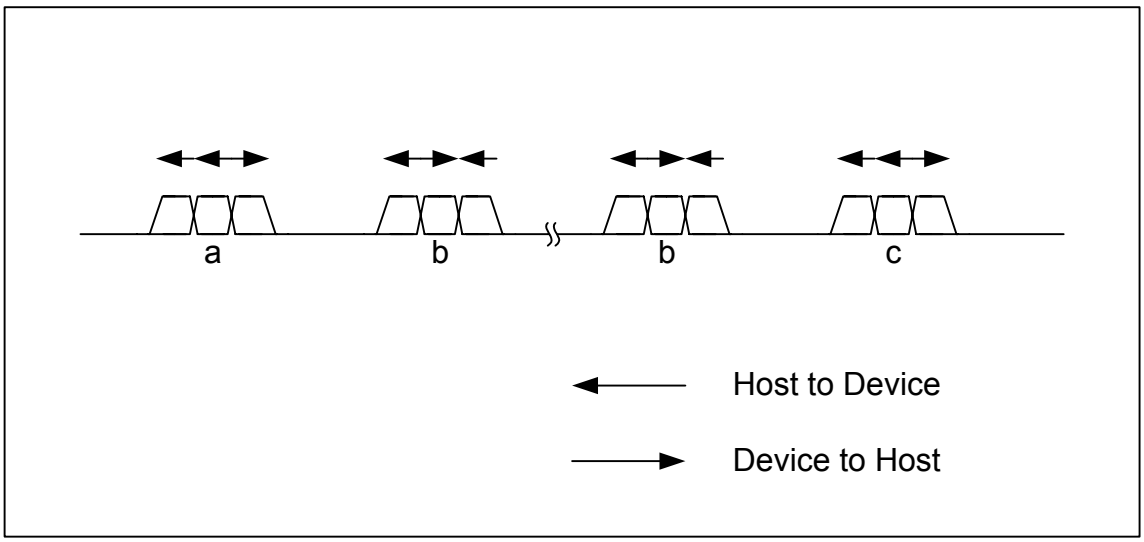


Figure 6-6 Control Transfer when the Data Stage is Directed for IN

For the data and status stages of control transfers, flow control by NAK is effective, as ordinary OUT and IN transactions are performed in those stages. The device is permitted in order to prepare to respond within a designated time.

6.1.3.1 Setup Stage

When a SETUP token addressed to the local node is received, the LSI automatically executes a setup transaction.

The firmware should monitor the RcvEP0SETUP status and analyze the request by reading the EP0SETUP_0 through EP0SETUP_7 registers in order to control the control transfer.

If the received request is for control transfers with an OUT-direction data stage, clear the INxOUT bit in the EP0Control register to direct the endpoint EP0 for OUT in order to move to the data stage.

If the received request is for control transfers with an IN-direction data stage, set the INxOUT bit in the EP0Control register to direct the endpoint EP0 for IN in order to move to the data stage.

If the received request is for control transfers without a data stage, set the INxOUT bit in the EP0Control register to direct the endpoint EP0 for IN in order to move to the status stage.

6.1.3.2 Data Stage and Status Stage

Move to the next stage according to the content of a request analyzed by reading the EP0SETUP_0 through EP0SETUP_7 registers.

If that stage is for the OUT direction, clear the INxOUT bit in the EP0Control register to direct it for OUT, and then set up the EP0ControlOUT register as appropriate, in order to control the stage. When the SETUP stage has finished, the ForceNAK bit must have been set. Similarly, the SETUP_Control.ProtectEP0 bit must have been set.

If that stage is for the IN direction, set the INxOUT bit in the EP0Control register to direct it for IN and then set up the EP0ControlIN register as appropriate, in order to control the

stage. When the SETUP stage has finished, the ForceNAK bit must have been set. Similarly, the SETUP_Control.ProtectEP0 bit must have been set.

6.1.3.3 Automatic Address Setup Function

The LSI stipulated herein has a function for automating the processing of SetAddress() requests in control transfers at the endpoint EP0.

The LSI checks the content of a request by reading the EP0SETUP_0 through EP0SETUP_7 registers in the hardware. If the request is found to be a valid SetAddress() request, the LSI movers on to processing of the status stage for that request without notifying the firmware thereof. When the status stage is completed, the LSI sets the relevant address in the USB_Address register and sends the SetAddressCmp status (SIE_IntStat.SetAddressCmp bit) to the firmware.

The firmware should monitor the SetAddressCmp status, so that when the status is issued, it can confirm the address by reading the USB_Address register.

6.1.3.4 Descriptor Reply Function

The LSI stipulated herein has a descriptor reply function that is effective for requests in control transfers at the endpoint EP0, such as GetDescriptor() requesting data that has been issued a number of times.

For requests in which the data stage is for IN transfers, the firmware can make use of this function.

Before starting a response to the data stage by clearing the EP0ControlIN.ForceNAK bit, set the start address of the internal data of the FIFO's descriptor area to be returned in the DescAdrs_H,L registers or the total number of bytes of the data to be returned in the DescSize_H,L registers to set the EPOControl.ReplyDescriptor.

The descriptor reply function executes an IN transaction by sending back data packets in response to the IN transaction of the data stage by the time all of the set bytes of data have been transmitted. If an IN transaction is issued after all of the set bytes of data have been transmitted, the function responds to it with NAK. If odd items of data smaller than the max. packet size exists, the descriptor reply function sets the EP0ControlIN.EnShortPkt bit to enable an IN transaction to be responded to before all bytes of data are sent back.

When transition to the status stage is detected by the reception of an OUT token, the function clears the EP0Control.ReplyDescriptor bit and sends the DescriptorCmp status (FIFO_IntStat.DescriptorCmp bit) to the firmware. When the DescriptorCmp status is detected, the firmware should execute the status stage.

For details on the descriptor area, refer to the explanation of the FIFO in the functional description.

6.1.4 Bulk Transfer and Interrupt Transfer

Bulk and interrupt transfers at the general-purpose endpoints EPa, EPb, and EPc can be controlled as a data flow (see 6.1.5) or as successive individual transactions (see 6.1.2).

6.1.5 Data Flow

The following describes the general data flow control for OUT and IN transfers.

6.1.5.1 OUT Transfer

The data received by an OUT transfer is written into the FIFO for each relevant endpoint. There are several methods available to read data out of the FIFO: a register read through the CPU interface, a DMA read through the CPU interface, or a write transfer to the IDE.

To read data out of the FIFO by means of a register read through the CPU interface, select only one endpoint using the EPx{x=0, a-c}Join.JoinCPU_Rd bit. The FIFO for the selected endpoint can be read in the order in which data was received by the EPnFIFO_Rd or EPnFIFO_ByteRd register. The bytes of data that can be read out of the FIFO can be determined by inspecting the EPnRdRemain_H, L registers. As empty FIFOs cannot be read out, be sure to check the EPnRdRemain_H, L registers in order to determine the number of bytes of data in the FIFO, and confirm that those bytes of data will not be exceeded when the FIFO is read.

To read data out of the FIFO by means of an DMA read through the CPU interface, select only one endpoint for each DMA channel using the EPx{x=0, a-c}Join.JoinDMAx{x=0,1}_Rd bit. The FIFO for the selected endpoint can be read out in the order in which data was received by executing a DMA procedure. The number of remaining bytes of data in the FIFO can be determined by inspecting the DMAx{x=0,1}_Remain_H, L registers. When the FIFO is emptied, the CPU interface automatically causes the DMA to pause for flow control.

To read data out of the FIFO by means of a write transfer through the IDE interface, select only one OUT endpoint using the EPx{x=a-c}Join.JoinIDE bit. The FIFO for the selected endpoint can be read in the order in which data was received by executing an IDE transfer by IDE_Control.DTGo. When the FIFO is emptied, the IDE interface automatically causes the write transfer to pause for flow control.

If the FIFO has sufficient free space to receive data packets, data can be received by automatically responding to an OUT transaction. Therefore, OUT transfers can be performed without the need for control of individual transactions by the firmware. However, if short packets (including those with a data length of zero) are received while the EPx{x=a-c}Control.DisAF_NAK_Short bit is cleared (default), the EPx{x=a-c}Control.ForceNAK bit for the relevant endpoint is set. Therefore, when the next data transfer is readied, be sure to clear the EPx{x=a-c}Control.ForceNAK bit.

6.1.5.2 IN Transfer

Write the data to be sent by an IN transfer into the FIFO for each relevant endpoint. To write data into the FIFO, several methods are available: a register write through the CPU interface, a DMA write through the CPU interface, or a read transfer from the IDE.

To write data into the FIFO by means of a register write through the CPU interface, select only one endpoint by using the EPx{x=0, a-c}Join.JoinCPU_Wr bit. The FIFO for the selected endpoint can be written to using the EPnFIFO_Wr register; the data is transmitted in packets in the order written. The amount of free space in the FIFO can be determined by inspecting the EPnWrRemain_H, L registers. Filled FIFOs cannot be written to. Always be sure to check the EPnWrRemain_H, L registers in order to determine the number of free bytes in the FIFO, and make sure those bytes will not be exceeded when the FIFO is written to.

To write data into the FIFO by means of a DMA write through the CPU interface, select only one endpoint for each DMA channel by using the EPx{x=0,

a-c}Join.JoinDMAx{x=0,1}_Wr bit. The FIFO for the selected endpoint can be written to by executing a DMA procedure in the CPU interface, and the data is transmitted in packets in the order written. When the FIFO is filled, the CPU interface automatically causes the DMA to pause for flow control.

To write data into the FIFO by means of a read transfer through the IDE interface, select only one IN endpoint by using the EPx{x=a-c}Join.JoinIDE bit. The FIFO for the selected endpoint can be written to in the order read from the IDE by executing an IDE transfer by IDE_Control.IDE_Go; the data is transmitted in packets in the order written. When the FIFO is filled, the IDE interface automatically causes the read transfer to pause for flow control.

If the FIFO contains data equal to or larger than the max. packet size, the data can be transmitted by automatically responding to an IN transaction. Therefore, IN transfers can be performed without need for the control of individual transactions by the firmware. However, if it is necessary to transmit a short packet at the end of data transfer, set the EnShortPkt bit. This bit is cleared upon completion of the IN transaction by which the short packet is to be transmitted. The bit can be set when the LSI has finished writing data to the FIFO. Furthermore, if, while the DMAx{x=0,1}_FIFO_Control.AutoEnShort bit is set, the FIFO contains odd number data smaller than the max. packet size when a DMA write through the CPU interface has been completed, the EnShortPkt bit for the relevant endpoint is automatically set.

6.1.6 Bulk-Only Support

The LSI stipulated herein has a bulk-only support function that, in bulk transfers at the endpoints EPb and EPc, provides support for Command Block Wrapper (CBW) receptions and Command Status Wrapper (CSW) transmissions specific to the USB Mass Storage Class (BulkOnly Transport Protocol).

Setting the BulkOnlyConfig.EPx{x=b,c}BulkOnly bit enables the bulk-only support function for the target endpoint.

While CBW support or CSW support of the bulk-only support function is being executed, the LSI uses the area reserved as the CBW or CSW area, rather than the FIFOs normally reserved for endpoints, when it performs packet reception (CBW) or transmission (CSW).

6.1.6.1 CBW Support

The firmware can use CBW support when it performs a command transport of the BulkOnly Transport Protocol. When the BulkOnlyConfig.EPx{x=b,c}BulkOnly bit is set, CBW support for the corresponding OUT endpoint is enabled. Control should be exercised in such a way that CBW support is enabled for only one endpoint at a time. Setting the BulkOnlyControl.GoCBW_Mode bit while CBW support is effective causes CBW support to be executed, so that the data received in an OUT transaction at the target endpoint is handled as CBW.

If a data packet is 31 bytes in length, or the anticipated data length of the CBW, the LSI saves the data in the CBW area and sends the CBW-complete status (BulkIntStat.CWB_Cmp bit) to the firmware. It also automatically clears the BulkOnlyControl.GoCBW_Mode bit to terminate execution of CBW support. Furthermore, if the BulkOnlyControl.GoCSW_Mode bit remains set at this time, the LSI clears it as well.

If the data packet has a data length of less than or greater than 31 bytes, the LSI sends the CBW data-length error status (BulkIntStat.CBW_LengthErr bit) to the firmware. It also

automatically clears the BulkOnlyControl.GoCBW_Mode bit to terminate execution of CBW support. Furthermore, if the BulkOnlyControl.GoCSW_Mode bit remains set at this time, the LSI clears it as well. If the CBW_LengthErr status is sent, a phase mismatch has occurred in the BulkOnly Transport Protocol. Therefore, the firmware should restore communication by, for example, STALL'ing the endpoint.

If EPx{x=b,c}Control.ForceSTALL is set at the target endpoint and an OUT transaction is responded to with STALL, or if a CRC error or other transaction error occurred in an OUT transaction, the LSI sends the CBW error status (BulkIntStat.CBW_Err bit) to the firmware without receiving data. In such a case, the BulkOnlyControl.GoCBW_Mode bit is not cleared and execution of CBW support is continued. Furthermore, even if the BulkOnlyControl.GoCSW_Mode bit remains set at this time, it is not cleared either.

The data received in the CBW area can be read using the RAM_Rd function.

6.1.6.2 CSW Support

The firmware can use CSW support when it performs a status transport of the BulkOnly Transport Protocol. When the BulkOnlyConfig.EPx{x=b,c}BulkOnly bit is set, CSW support for the corresponding IN endpoint is enabled. Control should be exercised in such a way that CSW support is enabled for only one endpoint at a time. Setting the BulkOnlyControl.GoCSW_Mode bit while CSW support is effective causes CSW support to be executed, so that the data to be transmitted in an IN transaction at the target endpoint is handled as CSW.

If, in an IN transaction, ACK is received from the host after 13 bytes of CSW data have been sent back to the host and the transaction is thereby completed, the LSI sends the CSW-complete status (BulkIntStat.CSW_Cmp bit) to the firmware. It also automatically clears the BulkOnlyControl.GoCSW_Mode bit to terminate execution of CSW support. At the same time, it sets the BulkOnlyControl.GoCBW_Mode bit to initiate execution of CBW support.

If, in an IN transaction, ACK cannot be received from the host after 13 bytes of data has been sent back to the host, the LSI sends the CSW error status (BulkIntStat.CSW_Err bit) to the firmware. At this time, the LSI does not clear the BulkOnlyControl.GoCSW_Mode bit, and continues execution of CSW support. At the same time, it sets the BulkOnlyControl.GoCBW_Mode bit in the hardware to initiate execution of CBW support. In such a case, therefore, execution of CSW support and execution of CBW support are performed at the same time. If the host could not receive CSW, resulting in an error, CSW will be retried; however, because CSW support is being executed, a response can be returned. Furthermore, if the device could not receive ACK, resulting in an error, the next CBW will be performed; however, because CBW support is being executed, a response can be returned, and execution of CSW support is terminated by the CBW support thus executed.

Data can be written to the CSW area by using the RAM_Door function.

6.1.7 Auto Negotiation Function

Suspend Detection, Reset Detection, HS Detection Handshaking, Resume Detection, and Restore Execution are automatically performed each time the USB bus status is checked. What has actually been executed can be confirmed by checking the respective interrupts (DetectRESET, DetectSUSPEND, ChirpCmp, or RestoreCmp).

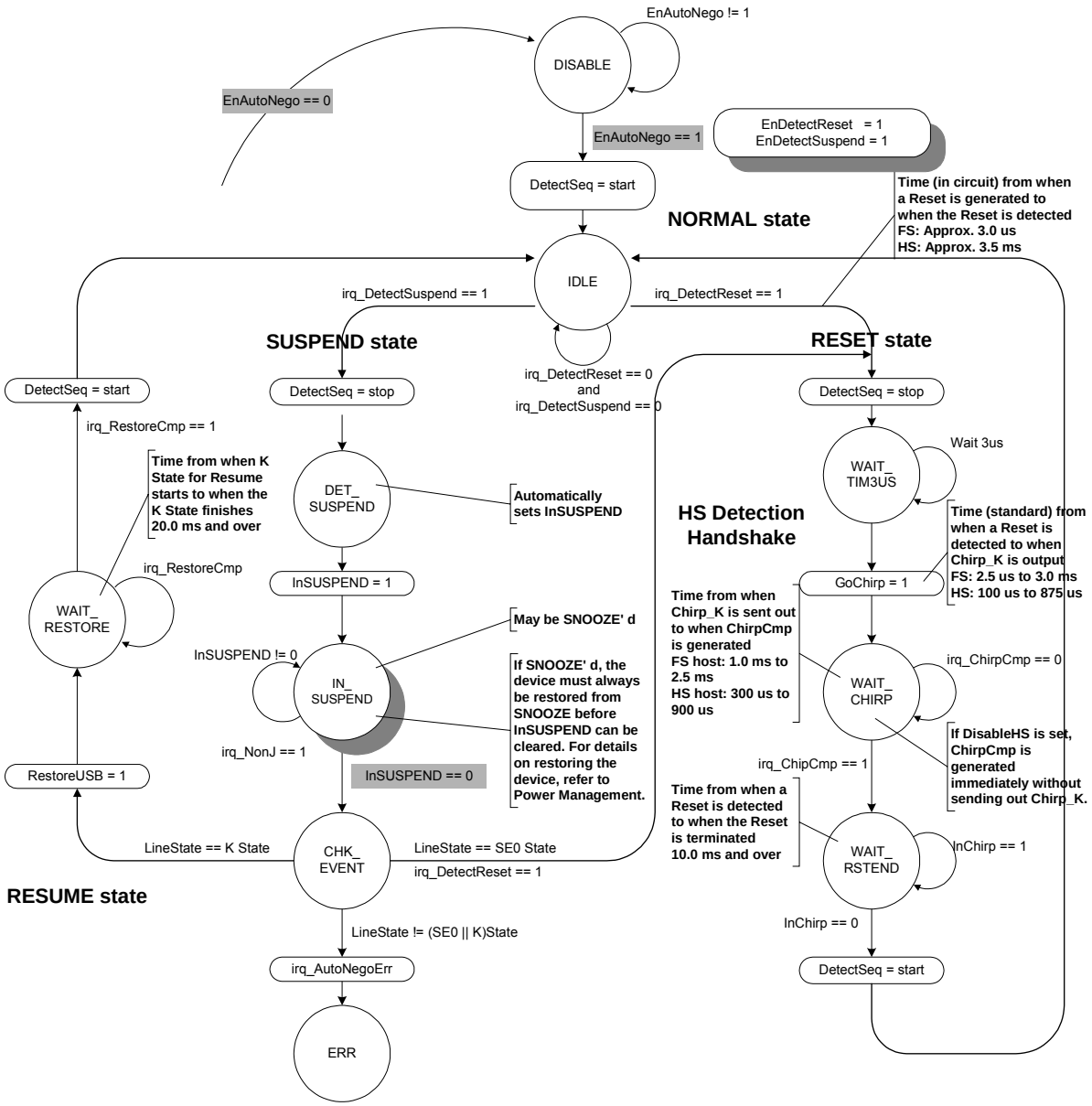


Figure 6-7 Auto Negotiator

6.1.7.1 DISABLE

This state is entered when the USB_Control.EnAutoNego bit is cleared.

If the auto-negotiation function is to be enabled, set the reset detection interrupt enable bit (SIE_IntEnb.EnDetectRESET) and the suspend detection interrupt enable bit (SIE_IntEnb.EnDetectSUSPEND) to enable both event detection interrupts before setting the USB_Control.EnAutoNego bit.

When the auto-negotiation function is enabled, the internal event detection function is enabled automatically. While the auto negotiation function remains enabled, the USB_Control.DisBusDetect bit can never be set.

6.1.7.2 IDLE

This is the state in which Reset detection or Suspend detection is waited for.

In cases in which the current USB speed is HS, if no activity on the USB bus is detected for 3 ms or more, the FS termination is temporarily enabled; then, Suspend is assumed when FS-J is detected, or Reset is assumed when SE0 is detected. In cases in which the current USB speed is FS, Reset is assumed when SE0 with a duration of 2.5 μs or more is detected, or Suspend is assumed when no bus activity is detected for 3 ms or more. A reset detection interrupt or suspend detection interrupt is generated at the same time these determinations are made, and the SIE_IntStat.DetectREST or SIE_IntStat.DetectSUSPEND bit is set.

When Suspend is assumed, the event detection function is temporarily turned off and the DET_SUSPEND state is entered.

When Reset is assumed, the event detection function is temporarily turned off and the WAIT_TIM3US state is entered.

6.1.7.3 WAIT_TIM3US

The time before HS Detection Handshaking is executed after reset detection is adjusted. The WAIT_CHIRP state is entered after the elapse of a predetermined length of time (approx. 3 μs later).

6.1.7.4 WAIT_CHIRP

HS Detection Handshaking is executed by automatically setting the USB_Control.GoChirp bit. Upon completion of HS Detection Handshaking, the Chirp-complete interrupt status (SIE_IntStat.ChirpCmp) is set and the WAIT_RSTEND state is entered. For details on HS Detection Handshaking, refer to Section 6.1.7.11.5

Furthermore, while the USB_Control.DisableHS bit remains set, the Chirp-complete interrupt status (SIE_IntStat.ChirpCmp) is set and the WAIT_RSTEND state is entered without executing HS Detection Handshaking.

Note that after this state terminates, the device operates at the transfer speed set in the USB_Status.FSxHS bit. If it is necessary to detect that the transfer speed has changed, set the SIE_IntEnb.EnChirpCmp bit to enable the Chirp-complete interrupt described above.

6.1.7.5 WAIT_RSTEND

The device waits in this state until the reset period terminates. During HS, the reset period is determined to have finished by the fact that Chirp transmission from the host (or reception for the device) has completed. During FS, the same is assumed by the fact that a transition from SE0 to J has occurred.

After the reset period is determined to have finished, the event detection function is enabled and the IDLE state is entered again.

6.1.7.6 DET_SUSPEND

When Suspend is assumed, the USB_Control.InSUSPEND bit is automatically set and the IN_SUSPEND state is entered. This USB_Control.InSUSPEND bit enables the function to detect a bus transition from FS-J to another, making it possible to detect Resume and Reset from the host.

Whether the device' s current consumption will actually be reduced during Suspend depends on the application. The LSI stipulated herein incorporates measures to reduce the current consumption in two stages (Snooze and Sleep). For details on these measures and on control procedures, refer to Section 6.2.

To ensure that Resume (FS-K), or the instruction to terminate Suspend, can be detected, the SIE_IntEnb.EnNonJ bit should be set by the firmware in order to enable the NonJ interrupt.

6.1.7.7 IN_SUSPEND

When the NonJ interrupt status (SIE_IntStat.NonJ) is set, the device assumes it to be a direction to return from Suspend, so that when the USB_Control.InSUSPEND bit is cleared by the firmware, it enters the CHK_EVENT state.

If spontaneous return from Suspend is desired in an application that has had its remote wakeup function enabled, set the USB_Control.SendWakeup bit in this state and output FS-K for a period of 1 ms or more but not exceeding 15 ms.

6.1.7.8 CHK_EVENT

Events on the USB cable are checked and, if FS-K is detected, Resume is assumed; if SE0 is detected, Reset is assumed. When Resume is assumed, the USB_Control.RestoreUSB bit is set, and the transfer speed before Suspend (which depends on the USB_Status.FSxHS value) is restored. When Reset is assumed, the event detection function is temporarily turned off, as in the case of a transition from the IDLE state, and the WAIT_TIM3US state is entered.

If a state that is neither FS-K nor SE0 is detected, the auto-negotiation error interrupt status (DBG_IntStat.AutoNegoErr) bit is set and the ERR state is entered.

6.1.7.9 WAIT_RESTORE

When the SIE_IntStat.RestoreCmp bit is set, the event detection function is enabled and the IDLE state is entered.

6.1.7.10 ERR

Once this state is entered, the device does not exit it unless the auto-negotiation function is turned off. This state is nonexistent in the USB standard.

Note that in any state, no determination is made with regard to removal of the USB cable. In the event that the USB cable is removed, therefore, the application should turn off the auto-negotiation function.

6.1.7.11 Individual Description of Each Negotiation Function

6.1.7.11.1 Suspend Detection (HS Mode)

If, while the LSI stipulated herein is operating in HS mode, no transmit/receive events are detected for 3 ms or more (T1), FS mode is entered automatically (HS termination is disabled and FS termination (Rpu) is enabled). As a result, DP is driven high, and the “J” state can be confirmed by checking the USB_Status.LineState [1:0] bits. (Be aware that if SE0 is detected, Reset is assumed, as will be described later.) Thereafter, if “J” is still detected at T2, the SIE_IntStat.DetectSUSPEND bit is set.

At this time, if both the SIE_IntEnb.EnDetectSUSPEND and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above, so that the Suspend state of USB is assumed. Shown in the diagram below is the device operation in Snooze mode.

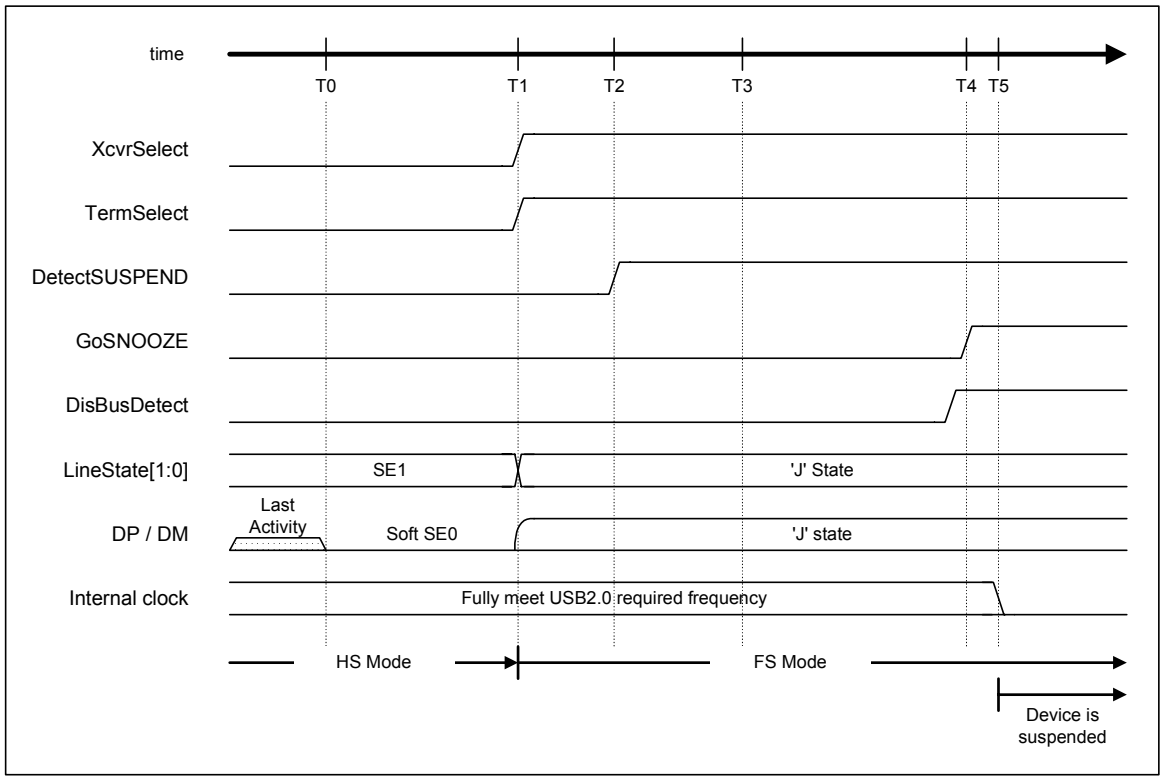


Figure 6-8 Suspend Timing (HS Mode)

Table 6-5 Suspend Timing Values (HS Mode)

Timing Parameter	Description	Value
T0	Last bus activity	0 (reference)
T1	If no bus activities are yet detected at this point in time, set XcvrSelect and TermSelect to 1 in order to change modes from HS to FS.	HS Reset T0 + 3.0ms < T1 {T_{WTREV}} < HS Reset T0 + 3.125ms
T2	Sample LineState [1:0]. If "J" is detected at this time, DetectSUSPEND goes high (= 1), so the Suspend state of USB should be assumed.	T1 + 100us < T2 {$T_{WTWRSTHS}$} < T1 + 875us
T3	RESUME cannot be issued prior to this state.	HS Reset T0 + 5ms {T_{WTRSM}}
T4	Set SnoozeUTM to 1; the device is thereby placed completely in snooze mode. Thereafter, no suspend currents greater than that stipulated for USB can be drawn from VBUS. (Set DisBusDetect to 1 before entering snooze mode.)	HS Reset T0 + 10ms {T_{2SUSP}}
T5	The internal clock is completely turned off. (Snooze current = 8 mA, typ.)	T5 < T4 + 10us

Note: Shown in { } are the names stipulated in the USB2.0 standard.

6.1.7.11.2 Suspend Detection (FS Mode)

If, while the LSI stipulated herein is operating in FS mode, no transmit/receive events are detected for 3 ms or more, or "J" in the USB_Status.LineState [1:0] bit is detected continuously (T1) and is still detected at T2, the Suspend state of USB is assumed and the SIE_IntStat.DetectSUSPEND bit is set.

At this time, if both the SIE_IntEnb.EnDetectSUSPEND and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above. Shown in the diagram below is the device operation in Snooze mode.

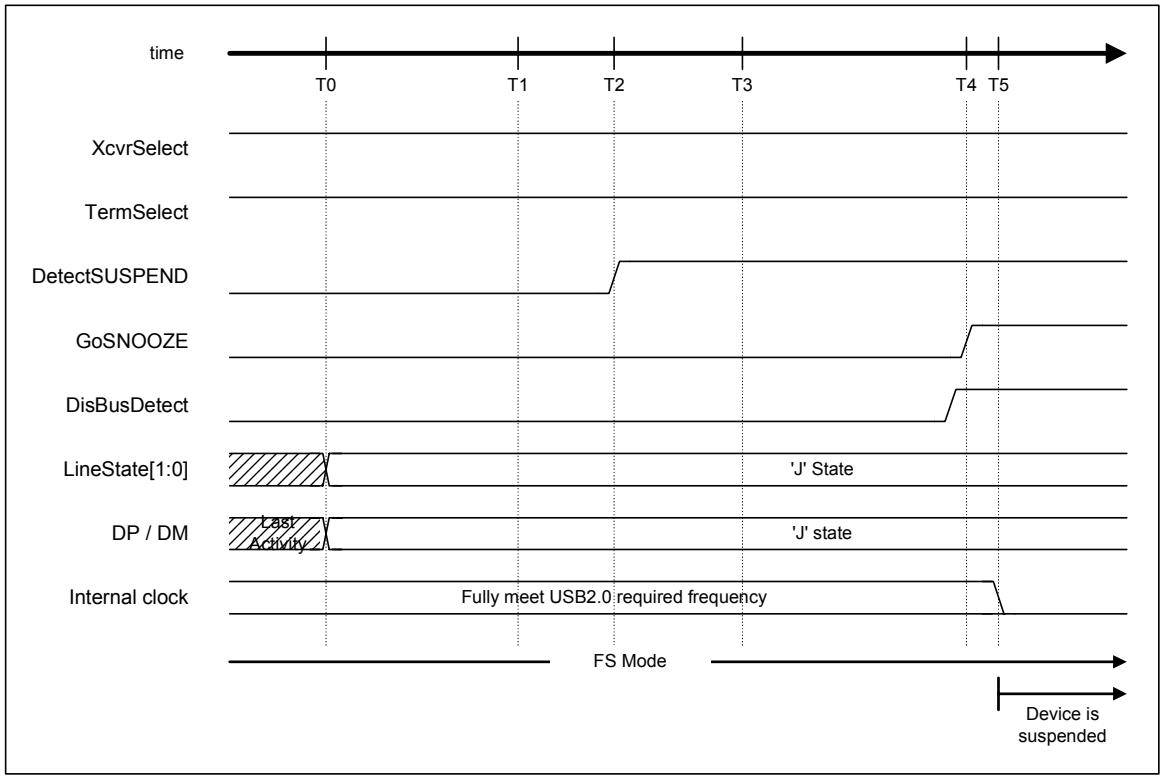


Figure 6-9 Suspend Timing (FS Mode)

Table 6-6 Suspend Timing Values (FS Mode)

Timing Parameter	Description	Value
T0	Last bus activity	0 (reference)
T1	No bus activities are yet detected at this point in time.	T0 + 3.0ms < T1 {T_{WTREV}} < T0 + 3.125ms
T2	Sample LineState [1:0]. If "J" is detected at this time, DetectSUSPEND goes high (= 1), so the Suspend state of USB should be assumed.	T1 + 100us < T2 {$T_{WTWRSTHS}$} < T1 + 875us
T3	RESUME cannot be issued prior to this state.	T0 + 5ms {T_{WTRSM}}
T4	Set SnoozeUTM to 1; the device is thereby placed completely in snooze mode. Thereafter, no suspend currents greater than that stipulated for USB can be drawn from VBUS. (Set DisBusDetect to 1 before entering snooze mode.)	T0 + 10ms {T_{2SUSP}}
T5	The internal clock is completely turned off. (Snooze current = 8 mA, typ.)	T5 < T4 + 10us

Note: Shown in { } are the names stipulated in the USB2.0 standard.

6.1.7.11.3 Reset Detection (HS Mode)

If, while the LSI stipulated herein is operating in HS mode, no transmit/receive events are detected for 3 ms or more, FS mode is entered automatically (HS termination is disabled and FS termination (Rpu) is enabled). Even after this state change, the DP line remains low, and the result of this state change can be confirmed by the fact that "SE0" in the USB_Status.LineState [1:0] bit is detected. If "SE0" is still detected at T2, the SIE_IntStat.DetectRESET bit is set.

At this time, if both the SIE_IntEnb.EnDetectRESET and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above, so an instruction for Reset should be assumed. Thereafter, execute HS Detection Handshaking (described later) after setting the USB_Control.DisBusDetect bit.

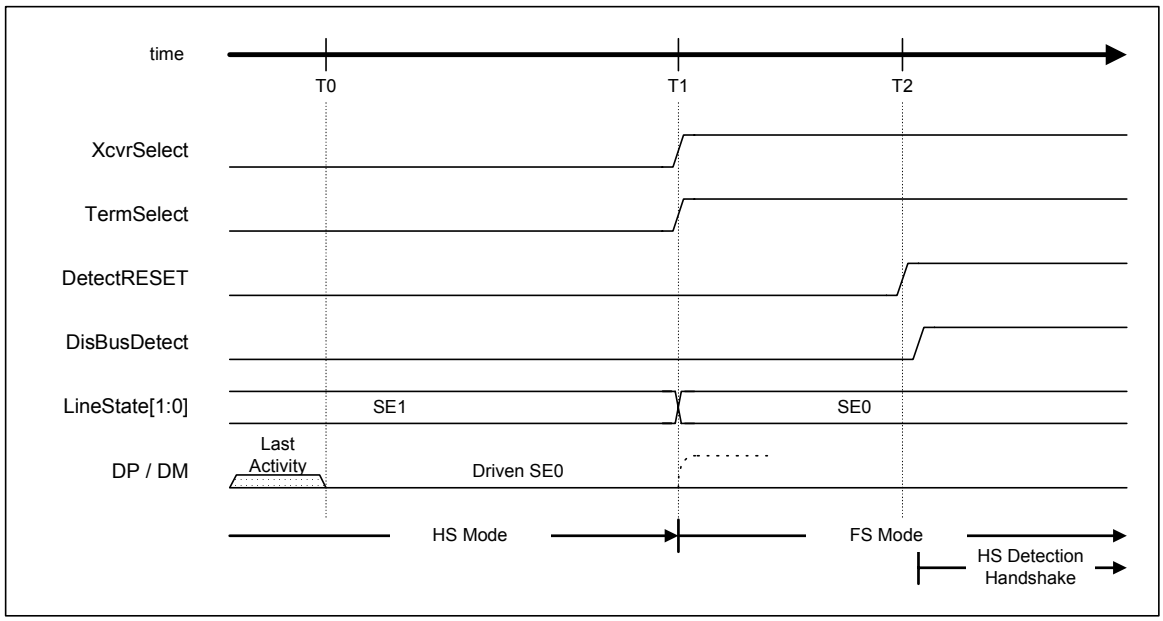


Figure 6-10 Reset Timing (HS Mode)

Table 6-7 Reset Timing Values (HS Mode)

Timing Parameter	Description	Value
T0	Last bus activity	0 (reference)
T1	If no bus activities are yet detected at this point in time, set XcvrSelect and TermSelect to 1 in order to change modes from HS to FS.	HS Reset T0 + 3.0ms < T1 {T_{WTREV}} < HS Reset T0 + 3.125ms
T2	Sample LineState [1:0]. If "SE0" is detected at this time, DetectRESET goes high (= 1), so a transition to Reset should be assumed. After detecting the Reset instruction, set DisBusDetect to 1. Thereafter, execute HS Detection Handshaking.	T1 + 100us < T2 {$T_{WTWRSTHS}$} < T1 + 875us

Note: Shown in { } are the names stipulated in the USB2.0 standard.

6.1.7.11.4 Reset Detection (FS Mode)

If, while the LSI stipulated herein is operating in FS mode, "SE0" in the USB_Status.LineState [1:0] bit is detected continuously for 2.5 μs or more (T1), the SIE_IntStat.DetectRESET bit is set.

At this time, if both the SIE_IntEnb.EnDetectRESET and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above, so an instruction for Reset should be assumed. Thereafter, execute HS Detection Handshaking (described later) after setting the USB_Control.DisBusDetect bit.

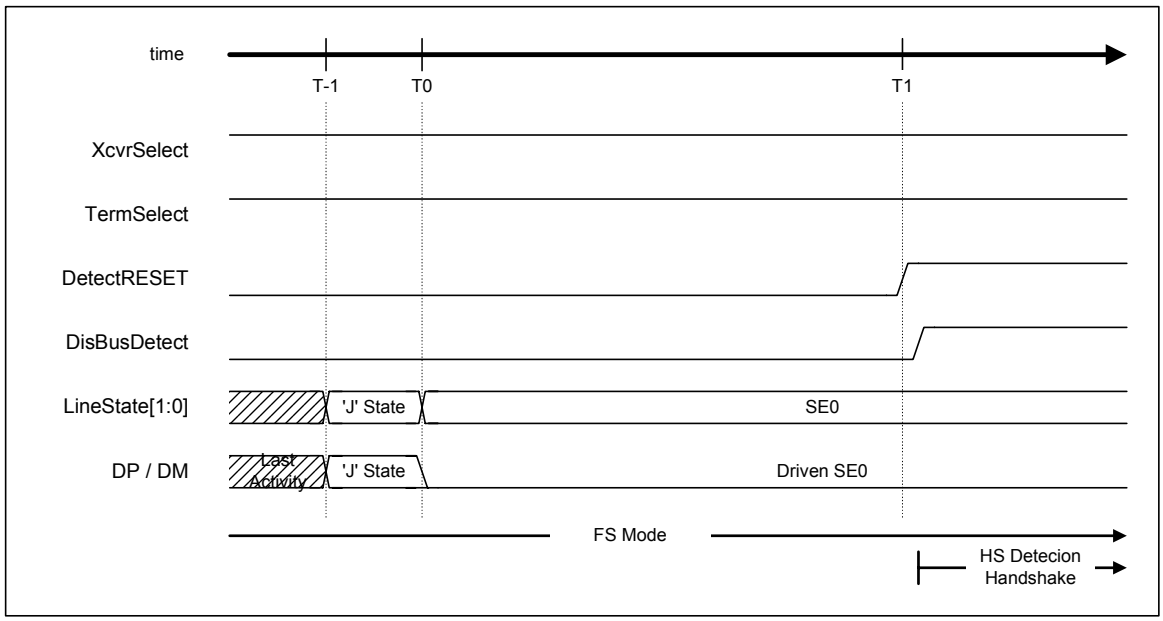


Figure 6-11 Reset Timing (FS Mode)

Table 6-8 Reset Timing Values (FS Mode)

Timing Parameter	Description	Value
T-1	Last bus activity	
T0	Reset instruction from the downstream port starts.	0 (reference)
T1	If "SE0" continues, DetectRESET is set to 1, so a transition to Reset should be assumed. After detecting the Reset instruction, set DisBusDetect to 1. Thereafter, execute HS Detection Handshaking.	HS Reset T0 + 2.5us < T1 {T_{WTREV}}

Note: Shown in { } are the names stipulated in the USB2.0 standard.

6.1.7.11.5 HS Detection Handshaking

HS Detection Handshaking is started from one of three states—Suspend, FS operation, or HS operation—by the assertion of "SE0" from the downstream port (when Reset is started from the above states). For details, refer to the USB2.0 standard.

The following describes how to switch to HS Detection Handshaking from the above three states.

While the LSI stipulated herein is operating in the Suspend state, switch to HS Detection Handshaking immediately after detecting "SE0" on the bus.

While the LSI stipulated herein is operating in FS mode, switch to HS Detection Handshaking after detecting "SE0" for a duration of 2.5 μs or more.

While the LSI stipulated herein is operating in HS mode, temporarily change modes to FS after detecting "SE0" for a duration of 3.0 ms or more in order to determine whether the Suspend or Reset state of USB is indicated, before switching to HS Detection Handshaking. To perform the changeover operation here, change the XcvrControl.XcvrSelect and XcvrControl.TermSelect bits to FS mode, disable HS termination, and enable FS termination. These mode changeover operations must be performed within 3.125 ms.

Within a period of 100 µs to 875 µs after the mode changeover, check the USB_Status.LineState [1:0] bits; if the bits indicate "J," the Suspend state of USB should be assumed; if they indicate "SE0," the Reset state of USB should be assumed. If Reset is assumed at this time, switch to HS Detection Handshaking thereafter.

In either case, a reset will be for at least 10 ms, but the exact timing differs slightly depending on the state (HS or FS) prior to the transition. Here, the time at which Reset was started is defined as "HS Reset T0"; the explanation below refers to the operation after "HS Reset T0."

Although no problems may be incurred while the LSI is operating and the internal clock has been considerably stable, if the LSI has been placed in sleep or snooze mode during Suspend, the internal clock is inactive when Reset is detected. Therefore, the PM_Control.GoActiveALL bit must be set to 1 in order to activate the internal clock before HS Detection Handshaking can be performed. For details on this operation, refer to Section 6.2, "Power Management Function."

6.1.7.11.5.1 When Connected to an FS Downstream Port

This section describes the operation of the LSI when it is connected to a downstream port that does not support HS. At the time HS Detection Handshaking starts (T0), both the XcvrControl.XcvrSelect and XcvrControl.TermSelect bits must be in FS mode (with FS termination, i.e., the DP pullup resistor (Rpu), enabled and HS termination disabled).

First, set the USB_Control.GoChirp bit. The XcvrControl.OpMode [1:0] bits will thereby be set to "Disable Bit Stuffing and NRZI encoding," and data filled with 0s will be prepared (T1). This is used to send "HS K" (chirp) to the bus. If the XcvrControl.XcvrSelect bit is set to HS mode and transmission is enabled simultaneously with the above, "HS K" (chirp) is sent to the downstream port. After the LSI has finished sending out "HS K", it waits for a "chirp" from the downstream port(T2). If the downstream port supports HS normally, "HS K" and "HS J" are sent from the downstream port successively beginning with T3 (as will be described later). If the downstream port does not support HS (as in the present case), a "chirp" is not sent from the downstream port even at T4, the XcvrControl.XcvrSelect bit is automatically changed to FS mode, and the USB_Status.FSxHS bit is then set at the same time the USB_Control.GoChirp bit is cleared. Further, the SIE_IntStat.ChirpCmp bit is set.

At this time, if both the SIE_IntEnb.EnChirpCmp and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above, so HS Detection Handshaking should be assumed to have finished.

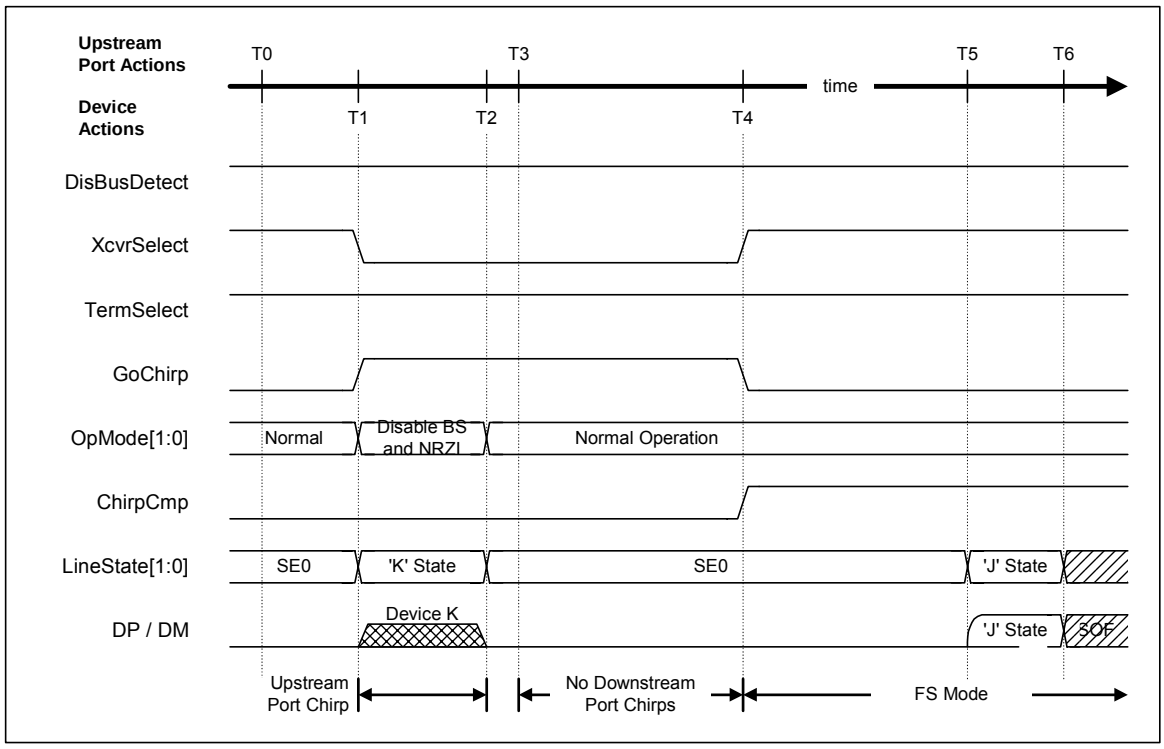


Figure 6-12 HS Detection Handshake Timing (FS Mode)

Table 6-9 HS Detection Handshake Timing Values (FS Mode)

Timing Parameter	Description	Value
T0	HS Detection Handshaking starts.	0 (reference)
T1	Enable the HS transceiver and set GoChirp to 1 in order to start sending Chirp K.	T0 < T1 < HS Reset T0 + 6.0ms
T2	Finish sending Chirp K. This signal must be sent out for at least 1 ms.	T1 + 1.0ms {T_{UCH}} < T2 < HS Reset T0 + 7.0ms {T_{UCHEND}}
T3	If the downstream port supports HS, start sending Chirp K from this point.	T2 < T3 < T2 + 100us {T_{WTDCH}}
T4	If Chirp cannot be detected, return to FS mode at this point in time; when ChirpCmp is set to 1, wait until the reset sequence finishes.	T2 + 1.0ms < T4 {T_{WTFS}} < T2 + 2.5ms
T5	The reset sequence finishes.	HS Reset T0 + 10ms {T_{DRST} (Min)}
T6	Normal operation in FS mode	T6

Note: Shown in { } are the names stipulated in the USB2.0 standard.

Note: To generate Chirp K of at least 1 ms, determination should be performed in 66,000 cycles (internal clock = 60 MHz).

6.1.7.11.5.2 When Connected to an HS Downstream Port

This section describes the operation of the LSI when it is connected to a downstream port that supports HS. At the time HS Detection Handshaking starts (T0), both the XcvrControl.XcvrSelect and XcvrControl.TermSelect bits must be in FS mode (with the FS termination, i.e., DP pullup resistor (Rpu) enabled and the HS termination disabled).

First, set the USB_Control.GoChirp bit. The XcvrControl.OpMode [1:0] bits will thereby be set to "Disable Bit Stuffing and NRZI encoding," and data containing a number of 0s will be prepared (T1). This is used to send "HS K" (chirp) to the bus. If the XcvrControl.XcvrSelect bit is set to HS mode and transmission is enabled simultaneously with the above, "HS K" (chirp) is sent to the downstream port. After the LSI has finished sending, it waits for a "chirp" from downstream (T2). As the downstream port supports HS in the present case, "HS K" (Chirp K) and "HS J" (Chirp J) are alternately sent out from the downstream port successively (T3). When this state is detected at least six times as Chirp K-J-K-J-K-J by the USB_Status.LineState [1:0] bits (T6), the XcvrControl.TermSelect bit is automatically changed to HS mode (T7), by which the device is placed completely into HS mode. At the same time, the USB_Control.GoChirp bit is cleared, as is the USB_Status.FSxHS bit, and further the SIE_IntStat.ChirpCmp bit is set.

At this time, if both the SIE_IntEnb.EnChirpCmp and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above, so HS Detection Handshaking should be assumed to have finished.

Chirp K and Chirp J from this downstream port should be recognized as bus activities, and must not be misjudged as the Suspend state of USB. To ensure this, when in HS mode, Chirp K and Chirp J are latched into the internal Suspend Timer each time they are detected.

Note that the USB_Status.LineState [1:0] bits are used to detect Chirp K-J-K-J-K-J. Unlike ordinary HS packets, Chirp K and Chirp J are very slow, and for this reason the USB_Status.LineState [1:0] bits are useful for said purpose. However, if bus signals are superimposed on the USB_Status.LineState [1:0] bits, the bus signals become extremely noisy, so that when the XcvrControl.TermSelect bit is in HS mode, the USB_Status.LineState [1:0] bits output "J" if presence of bus activity is detected, or "SE0" if the absence of bus activity is assumed.

In the diagram below, the change in the Chirp level beginning with the T6 point in time indicates that HS termination on the device side has been enabled by the XcvrControl.TermSelect bit. Normally, when XcvrControl.TermSelect is in FS mode, Chirp is approximately 800 mV, and when XcvrControl.TermSelect is in HS mode, Chirp (as for ordinary HS transmit/receive packets) is approximately 400 mV.

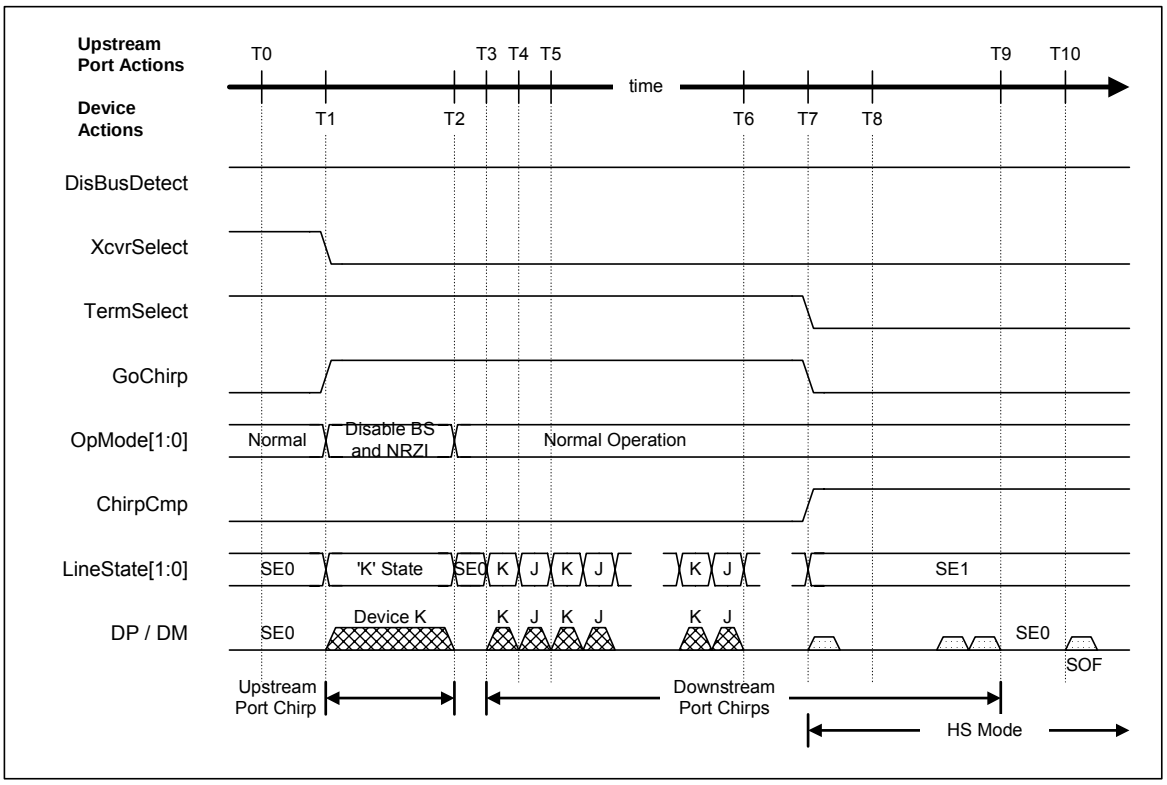


Figure 6-13 HS Detection Handshake Timing (HS Mode)

Table 6-10 HS Detection Handshake Timing Values (HS Mode)

Timing Parameter	Description	Value
T0	HS Detection Handshaking starts	0 (reference)
T1	Enable the HS transceiver and set GoChirp to 1 in order to start sending Chirp K.	T0 < T1 < HS Reset T0 + 6.0ms
T2	Finish sending Chirp K. This signal must be sent for at least 1 ms.	T1 + 1.0ms {T_{UCH}} < T2 < HS Reset T0 + 7.0ms {T_{UCHEND}}
T3	The downstream port sends the first Chirp K to the bus.	T2 < T3 < T2 + 100us {T_{WTDCH}}
T4	The downstream port stops sending Chirp K, and sends Chirp J instead.	T3 + 40us {T_{DCHBIT} (Min)} < T4 < T3 + 60us {T_{DCHBIT} (Max)}
T5	The downstream port stops sending Chirp J, and sends Chirp K instead.	T4 + 40us {T_{DCHBIT} (Min)} < T5 < T4 + 60us {T_{DCHBIT} (Max)}
T6	Chirp K-J-K-J-K-J are detected.	T6
T7	Pursuant to the detection of Chirp K-J-K-J-K-J, disable FS termination and enable HS termination. ChirpCmp is set to 1. Then, wait until Reset finishes.	T6 < T7 < T6 + 500us
T8	Recognized as bus activity due to Chirp K and Chirp J, but not recognized as packet reception underway, as SYNC cannot be detected.	T8
T9	Transmission of Chirp K and Chirp J from the downstream port finishes.	T10 - 500us {T_{DCHSE0} (Max)} < T9 < T10 - 100us{T_{DCHSE0} (Min)}
T10	The reset sequence finishes.	HS Reset T0 + 10ms {T_{DRST} (Min)}

Note: Shown in { } are the names stipulated in the USB2.0 standard.

Note: To generate Chirp K of at least 1 ms, determination should be performed in 66,000 cycles (internal clock = 60 MHz).

6.1.7.11.5.3 When Reset during Snooze

The LSI stipulated herein does not have its internal clocks output in snooze mode. The following describes a device operation assuming that the oscillator circuit is active (in snooze mode, not sleep mode).

If Reset is detected in snooze mode (T0), the SIE_IntStat.NonJ bit is set. Further, the SIE_IntEnb.EnNonJ bit is set, and if the MainIntEnb.EnSIE_IntStat bit has been set, the XINT signal will be asserted simultaneously with the above. In such a case, set the PM_Control.GoActiveALL bit to 1 in order to enable the device to immediately return from snooze mode and enter a reset sequence (T1). After the elapse of the PLL power-up time (T2), PM_Control.PM_State [2:0] becomes "ACTIVEALL," at which time the LSI starts outputting its internal clock. Thereafter, execute HS Detection Handshaking (described above).

At this time, unless the oscillator circuit is turned off (unless return from sleep mode), the internal clock is output with frequency accuracy conforming to the USB2.0 standard.

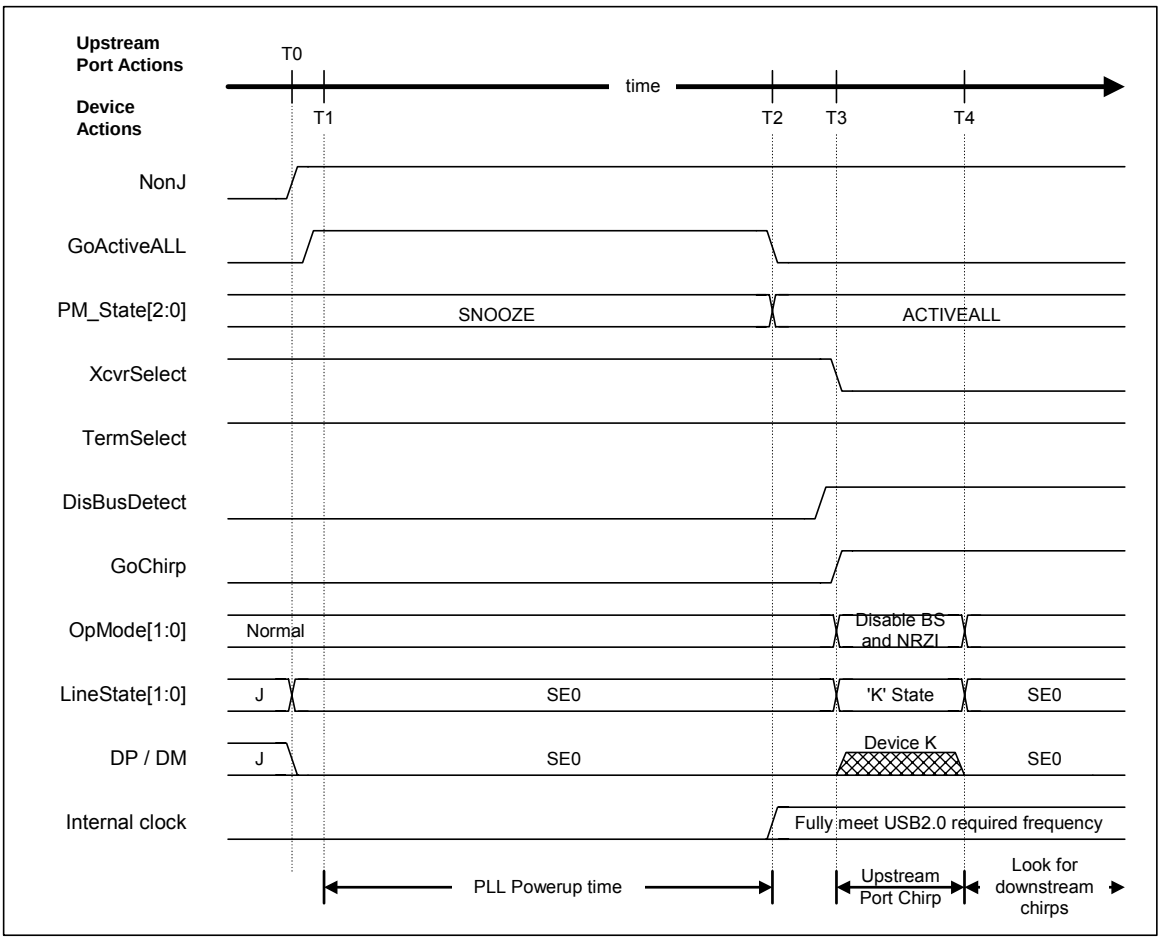


Figure 6-14 HS Detection Handshake Timing from Suspend

Table 6-11 HS Detection Handshake Timing Values from Suspend

Timing Parameter	Description	Value
T0	When NonJ is set to 1 and “SE0” is confirmed by LineState [1:0], Reset during snooze is detected.	0 (HS Reset T0)
T1	Following the detection of Reset, set GoActiveALL to 1.	T1
T2	PM_State becomes “ACTIVEALL.” The internal clock output stabilizes.	T1 + 250us < T2
T3	Set GoChirp to 1 in order to send Chirp K to the bus. (Set DisBusDetect to 1 before sending Chirp K.)	T2 < T3 < HS Reset T0 + 5.8ms
T4	Finish sending Chirp K.	T3 + 1.0ms {T_{UCH}} < T4 < HS Reset T0 + 7.0ms {T_{UCHEND}}

Note: The names stipulated in the USB2.0 standard are shown in { }.

Note: To generate Chirp K of at least 1 ms, determination should be performed in 66,000 cycles (internal clock = 60 MHz).

Note: If the oscillator circuit is also turned off (sleep mode), an OSC power-up time as well as the PLL power-up time described later are required.

6.1.7.11.6 Issuance of Resume

When remote wakeup is supported and this remote wakeup function is enabled from the host, there may be cases in which the device needs to perform Resume for some reason. This section describes how to perform Resume in such a case. Note, however, that at least 5 ms must elapse after the bus becomes idle before remote wakeup can be performed. Furthermore, no currents in the state prior to the entry into the Suspend state of USB can be drawn from VBUS before the passage of 10 ms after the Resume signal is output.

For remote wakeup to be performed, the device must first be restored from sleep/snooze. Clear the SIE_IntEnb.EnNonJ bit and set the PM_Control.GoActiveALL bit (T0); when the PM_Control.PM_State [2:0] bits are set to "ACTIVEALL" after the elapse of the PLL power-up time (T1), the internal clock starts oscillating. At this time, unless the oscillator circuit is turned off, the internal clock is output with frequency accuracy conforming to the USB2.0 standard.

Thereafter, set the USB_Control.SendWakeup bit to send out the Resume signal (T2). At this time, internally in the device, the XcvrControl.OpMode [1:0] bits are set to "Disable Bit Stuffing and NRZI encoding" and data consisting of 0s is prepared as the transmit data, and after a packet transmit state is entered, "K" (Resume signal) is sent. Upon detecting this resume signal, the downstream port returns "K" (resume signal) to the bus (T3).

By clearing the USB_Control.SendWakeup bit approximately 1 ms after the device started sending out the resume signal, it is possible to stop the resume signal from being sent out to the bus (T4). At this point in time, however, the downstream port still holds the resume signal on the bus.

Therefore, set the USB_Control.RestoreUSB bit. After the lapse of a predetermined length of time, the downstream port stops sending back the resume signal (T5) and instead sends out 2-bit LS-EOP(2*SE0) in order to change the previous speed mode prior to Suspend of USB. When this mode change is detected (or no "K" is detected), both the XcvrControl.XcvrSelect and XcvrControl.TermSelect bits are changed to the desired mode (in the present case, HS mode), and the USB_Control.RestoreUSB bit is cleared and the SIE_IntStat.RestoreCmp bit is set simultaneously. At this time, if both the SIE_IntEnb.EnRestoreCmp and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above.

When Suspend of USB starts, the speed mode (HS or FS) is saved in the USB_Status.FSxHS bit, so that when it is restored by Resume, the device returns to the mode indicated by this USB_Status.FSxHS bit. At this time, it is not necessary to execute HS Detection Handshaking for each Resume attempted. Please note that the explanation given here refers to only cases in which the speed mode prior to Suspend of USB was HS. Actually, in FS mode, the states following T5 become FS mode, and there is no significant difference in the sequence.

When the LSI stipulated herein is in snooze mode (PM_Control.PM_State [2:0] bits = SNOOZE), it does not have its internal clocks output. The device operation explained here assumes that the oscillator circuit is active (in snooze mode, not sleep mode).

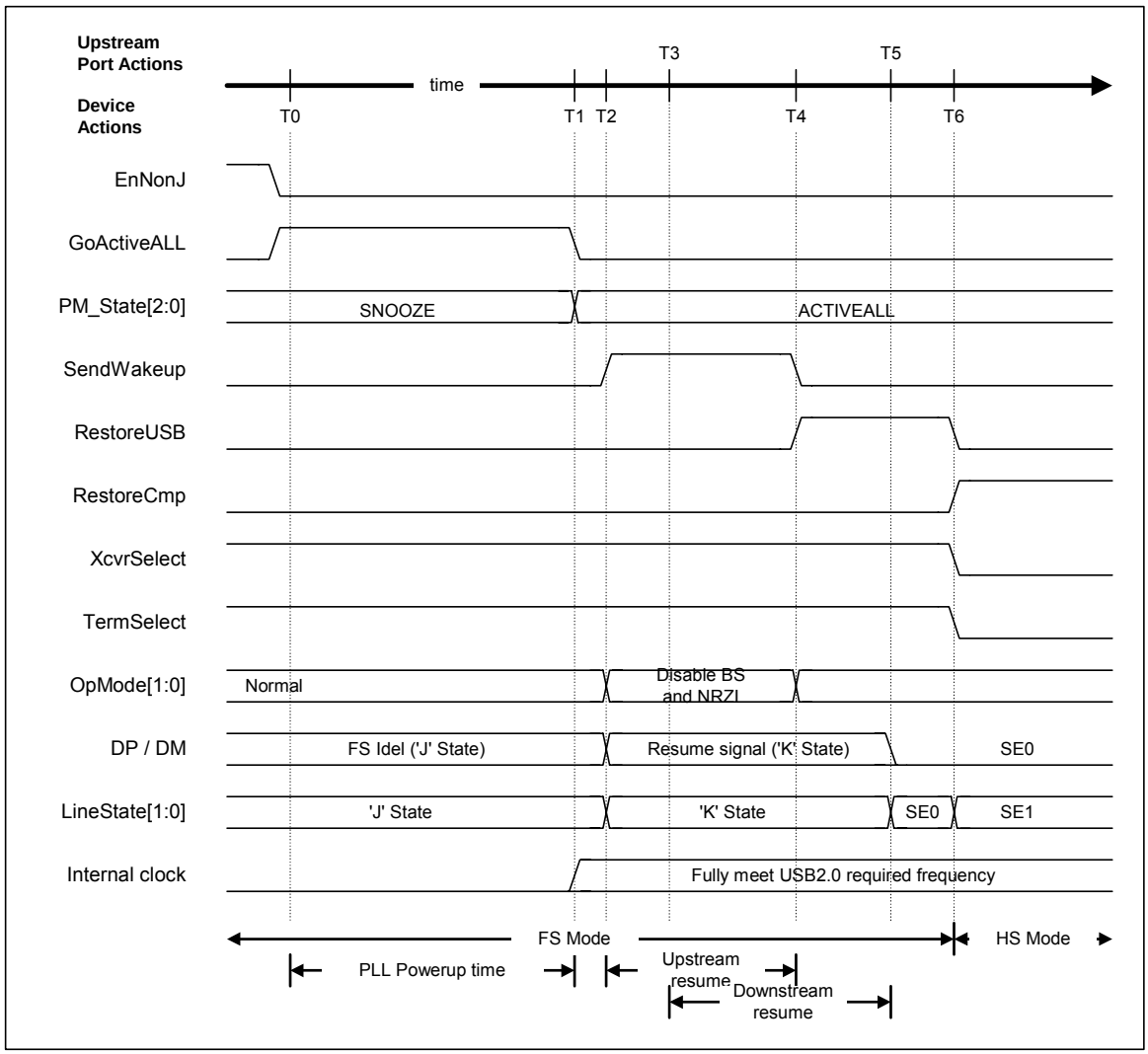


Figure 6-15 Assert Resume Timing (HS Mode)

Table 6-12 Assert Resume Timing Values (HS Mode)

Timing Parameter	Description	Value
T0	Resume starts. Set GoActiveALL to 1. (EnNonJ must be cleared to 0 before Resume starts.)	0 (reference)
T1	PM_State becomes "ACTIVEALL." Internal clock output stabilizes.	T0 + 250us < T1
T2	Set SendWakeup to 1 in order to start sending out "K" for FS. Here, no currents prior to Suspend of USB can be drawn from VBUS within 10 ms thereafter.	T0 < T2 < T0 + 10ms
T3	The downstream port returns "K" for FS.	T2 < T3 < T2 + 1.0ms
T4	Clear SendWakeup to 0 in order to finish sending out "K" for FS. After confirming "K" using LineState [1:0], set RestoreUSB to 1.	T2 + 1.0ms {T_{DRSMUP} (Min)} < T4 < T2 + 15ms {T_{DRSMUP} (Max)}
T5	The downstream port finishes sending back "K" for FS.	T2 + 20ms {T_{DRSMDN}}
T6	RestoreCmp is set to 1. If the mode prior to Suspend of USB was HS, the device automatically enters HS mode.	T5 + 1.33us {2 Low-speed bit times}

Note: The names stipulated in the USB2.0 standard are shown in { }.

6.1.7.11.7 Detection of Resume

While the LSI stipulated herein is snoozing, "J" (USB_Status.LineState [1:0] = J) is observed on the bus. If "K" is observed on the bus, an instruction for wakeup (instruction for Resume) has been received from the downstream port (T0). At this time, unless the oscillator circuit is turned off (or not in sleep mode), the SIE_IntStat.NonJ bit is set. If both the SIE_IntEnb.EnNonJ and MainIntEnb.EnSIE_IntStat bits have been set at this time, the XINT signal will be asserted simultaneously with the above.

First, set the PM_Control.GoActiveALL bit to 1 (T1); when the PM_Control.PM_State [2:0] bits are set to "ACTIVEALL" after the elapse of the PLL power-up time (T2), the internal clock starts oscillating. At this time, unless the oscillator circuit is turned off, the internal clock is output with frequency accuracy conforming to the USB2.0 standard.

Therefore, set the USB_Control.RestoreUSB bit. After the elapse of a predetermined time, the downstream port stops sending back the resume signal (T3) in order to change to the speed mode prior to Suspend of USB. When this mode change is detected (or no "K " is detected), both the XcvrControl.XcvrSelect and XcvrControl.TermSelect bits are changed to the desired mode (in the present case, HS mode), and the USB_Control.RestoreUSB bit is cleared and the SIE_IntStat.RestoreCmp bit is set simultaneously. At this time, if both the SIE_IntEnb.EnRestoreCmp and MainIntEnb.EnSIE_IntStat bits have been set, the XINT signal will be asserted simultaneously with the above.

The device operation explained here assumes that the oscillator circuit is active (in snooze mode, not sleep mode).

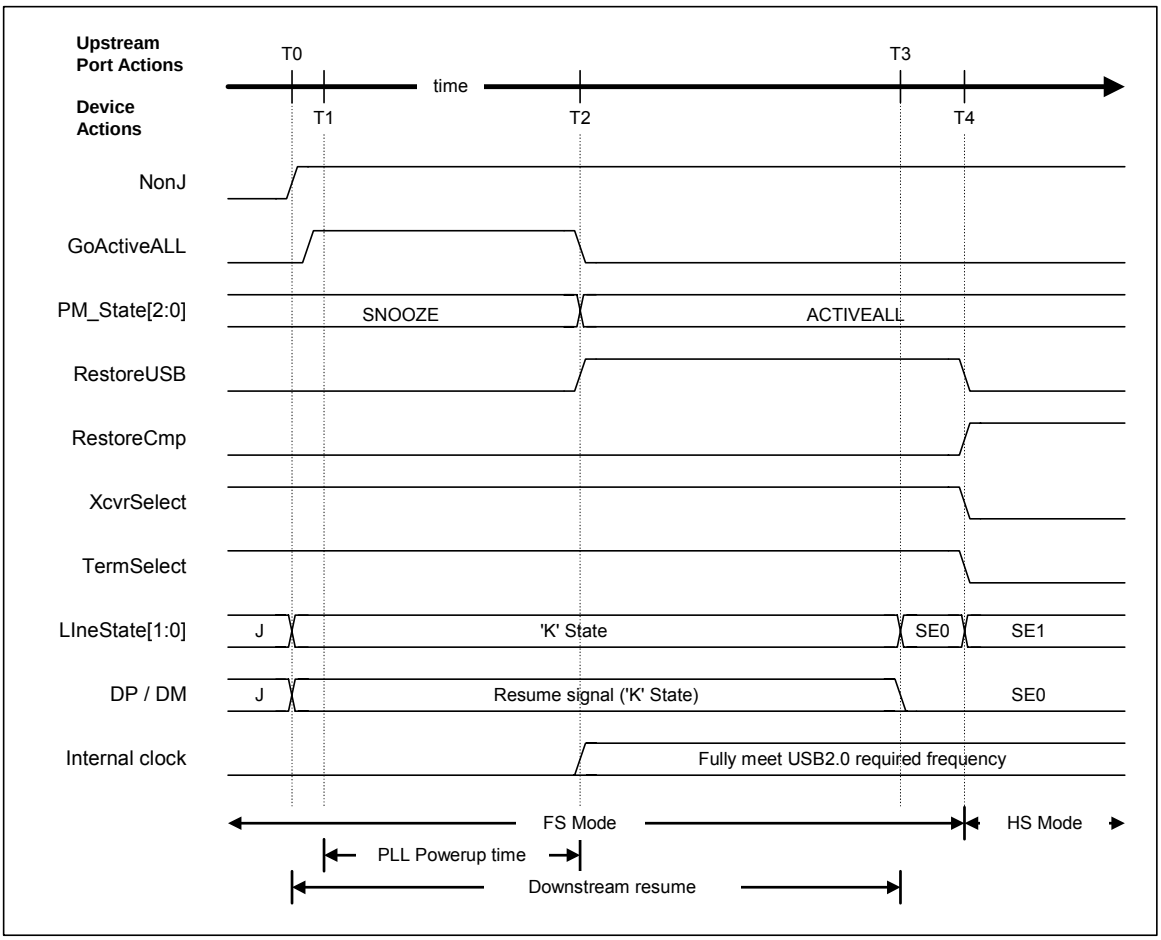


Figure 6-16 Detect Resume Timing (HS Mode)

Table 6-13 Detect Resume Timing Values (HS Mode)

Timing Parameter	Description	Value
T0	The downstream port sends “K” for FS. NonJ is set to 1.	0 (reference)
T1	Set GoActiveALL to 1.	T1
T2	PM_State becomes “ACTIVEALL.” The internal clock output stabilizes. After confirming “K” using LineState [1:0], set RestoreUSB to 1.	T1 + 250us < T2
T3	The downstream port finishes sending “K” for FS. At the same time, it enters HS mode, which it was in prior to Suspend of USB.	T2 + 20ms {T_{DRSMDN}}
T4	If the mode prior to Suspend of USB was HS, the device automatically enters HS mode.	T5 + 1.33us {2 Low-speed bit times}

Note: The names stipulated in the USB2.0 standard are shown in { }.

6.1.7.11.8 Insertion of a USB Cable

This section describes a case in which the device is connected to a hub or the host, i.e., in which it has had a USB cable inserted.

When the cable is removed or intentionally disconnected, make sure the XcvrControl.XcvrSelect and XcvrControl.TermSelect bits default to FS mode and HS mode, respectively.

When the cable is connected while no other cable is connected (T0), VBUS goes high and, at the same time, the USB_Status.VBUS bit is set (T1). If the device is placed in snooze mode at this time, set the PM_Control.GoActiveALL bit to 1 (T2), and when the PM_Control.PM_State [2:0] becomes "ACTIVEALL" after the elapse of the PLL power-up time (T3), the internal clock starts oscillating simultaneously. Thereafter, as it is necessary to temporarily enter FS mode due to the fact that an FS device has been connected, set the XcvrControl.TermSelect bit to FS mode (T4).

Thereafter, the downstream port sends out Reset (T5), from which HS Detection Handshaking starts.

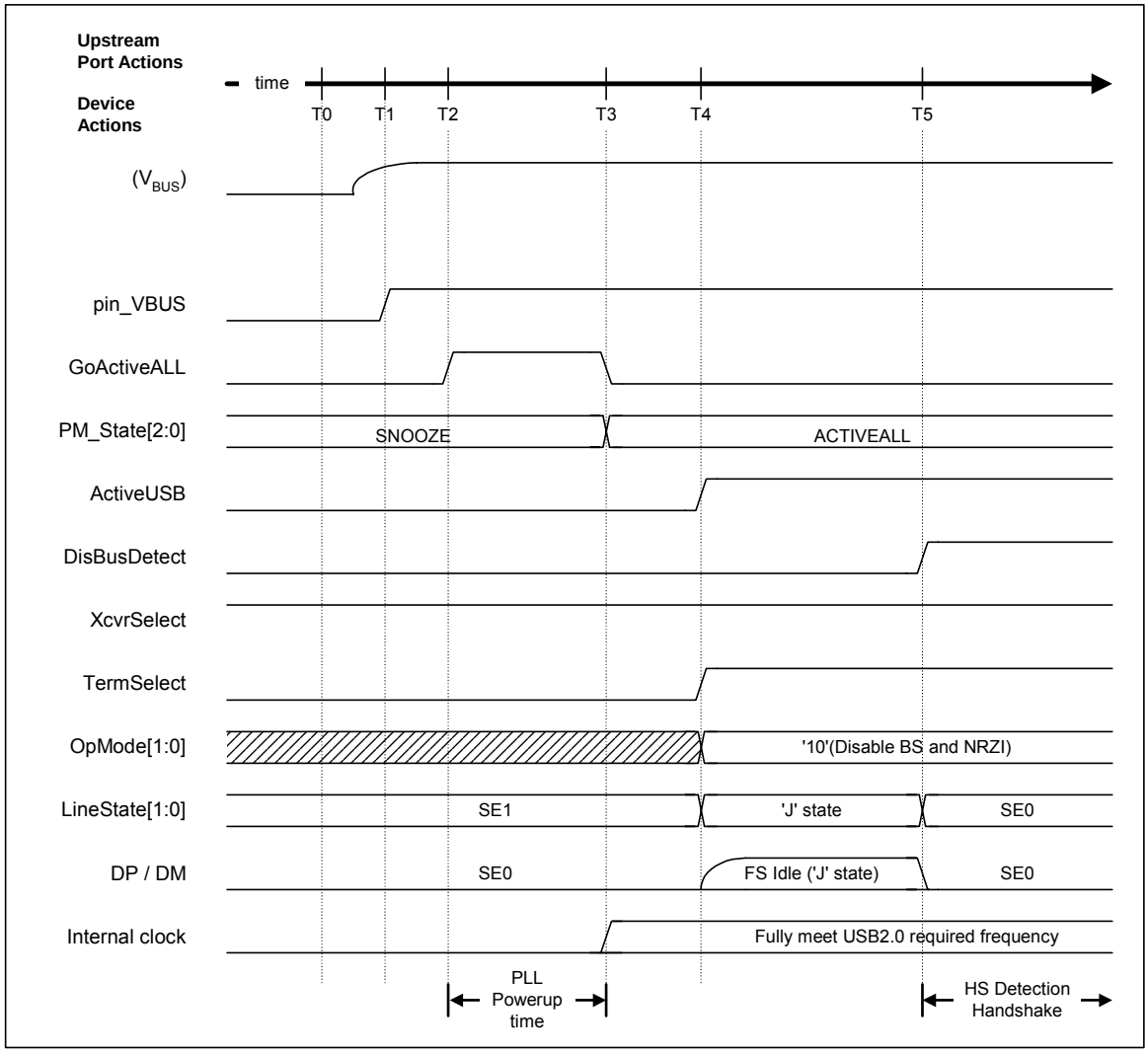


Figure 6-17 Device Attach Timing

Table 6-14 Device Attach Timing Values

Timing Parameter	Description	Value
T0	No cable is inserted.	0 (reference)
T1	A cable is inserted, and the input pin VBUS goes high.	T1
T2	Set GoActiveALL to 1.	T2
T3	PM_State becomes “ACTIVEALL.” Internal clock output stabilizes.	T2 + 250us < T3
T4	Set ActiveUSB to 1. Set TermSelect to 1. Set OpMode [1:0] to ‘00.' FS mode is entered. FS termination is enabled.	T1 + 100ms {T_{SIGATT}} < T4
T5	Reset is sent from the downstream port. Set DisBusDetect to 1.	T4 + 100ms {T_{ATTDB}} < T5

Note: The names stipulated in the USB2.0 standard are shown in { }.

6.2 Power Management Function

The power management function controls the operation of the oscillator and the PLLs (PLL480, PLL60), controlling transitions between four states: Sleep, Snooze, Active60, and ActiveAll. To enter other states, set the PM_Control.GoSLEEP, PM_Control.GoSNOOZE, PM_Control.GoActive60, or PM_Control.GoActiveALL bit; a state transition will start, and will be terminated after the assigned processing has been performed. To confirm which state the device is currently in, check the PM_Control.PM_State [2:0]. Furthermore, an SIE_IntStat.FinishedPM event is generated after a transition has finished. At this time, if both the SIE_IntEnb.EnFinishedPM and MainIntEnb.EnSIE_IntStat bits have been set, an XINT interrupt is generated.

A state can be entered from all other states, so that if the PM_Control.GoSLEEP bit is set in the ActiveAll state, the Sleep state is entered via the Active60 and Snooze states. When the state transition has fully completed, an SIE_IntStat.FinishedPM event is generated. In addition, if the PM_Control.GoActiveALL bit is set in the Sleep state, the ActiveAll state is entered via the Snooze and Active60 states. When the state transition has fully completed, an SIE_IntStat.FinishedPM event is generated. Similarly, if the PM_Control.GoSLEEP bit is set in the Active60 state, the Sleep state is entered via the Snooze state. When the state transition has fully completed, an SIE_IntStat.FinishedPM event is generated. Furthermore, if the PM_Control.GoActive60 bit is set in the Sleep state, the Active60 state is entered via the Snooze state. When the state transition has fully completed, an SIE_IntStat.FinishedPM event is generated.

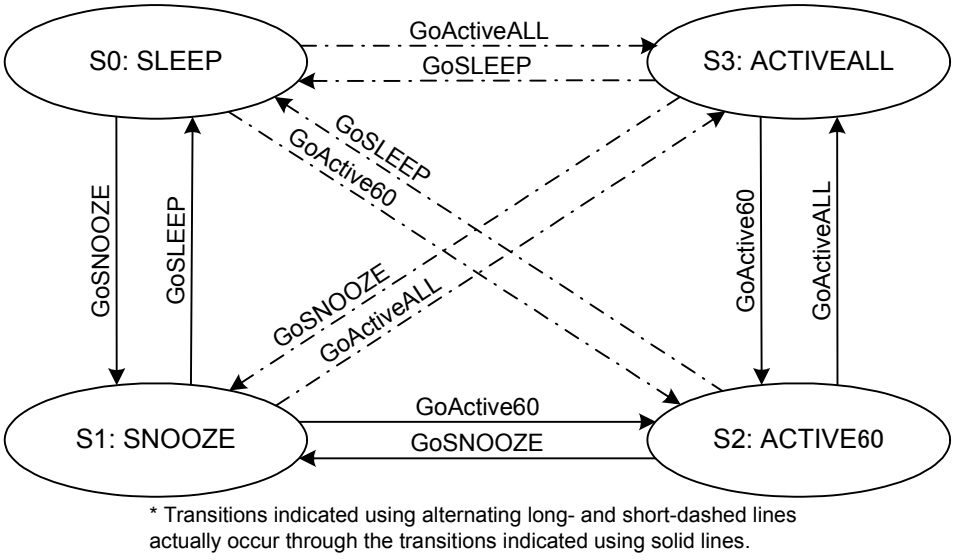


Figure 6-18 Power Management

6.2.1 SLEEP (Sleep)

This is the state in which the oscillator is not oscillating. In this state, therefore, the PLLs are not oscillating either.

When Sleep is entered into by setting the PM_Control.GoSLEEP bit in the Snooze, Active60, or ActiveAll state, the operating PLLs and OSC are turned off in the order of

PLL480 and PLL60, and finally the OSCCLK output is turned off before the clock oscillation is halted.

Conversely, if Snooze is entered into from Sleep by setting the PM_Control.GoSNOOZE, PM_Control.GoActive60, and PM_Control.GoActiveALL bits in the Sleep state, OSCCLK is gated for an oscillation stabilization time to ensure that it will not be supplied to the internal circuit until after the oscillator begins oscillating stably. As this oscillation stabilization time varies depending on the oscillator cell, resonator, peripheral circuits, and the board involved, use the WakeUpTim_H,L registers to set it.

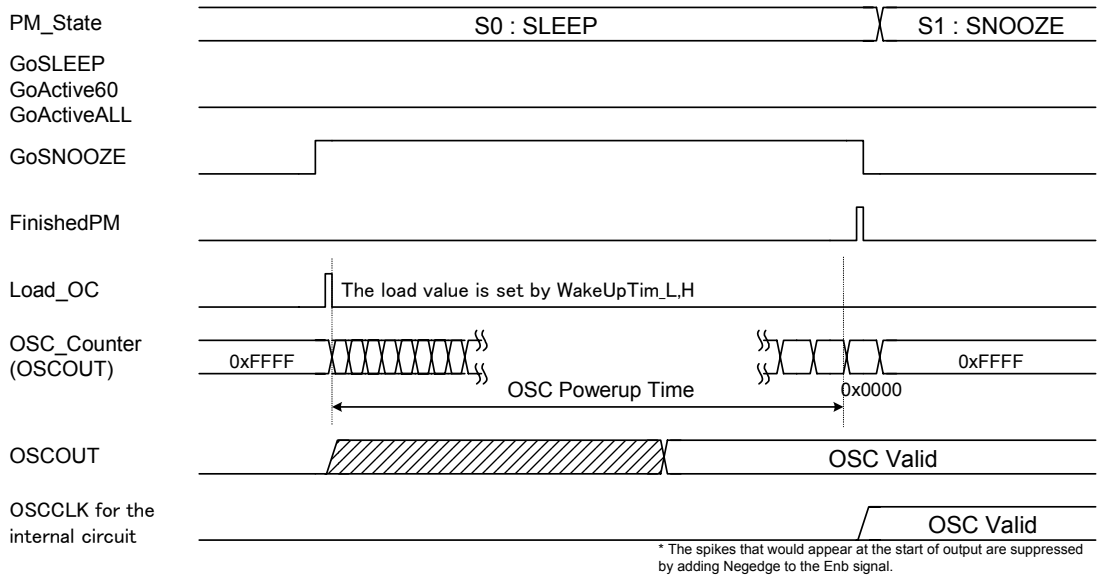


Figure 6-19 Exiting the SLEEP State (during GoSNOOZE)

6.2.2 SNOOZE (Snooze)

This is the state in which the oscillator is oscillating but the PLLs are not.

When Snooze is entered into by setting PM_Control.GoSNOOZE in the Active60 or ActiveAll state, the clock being output is halted and then the PLLs are turned off in the order of PLL480 and PLL60.

However, if Active is entered into from Snooze by setting the PM_Control.GoActiveALL and PM_Control.GoActive60 bits in the Snooze state, SCLK is gated for a PLL stabilization time (approx. 250 μs) to ensure that it will not be supplied to the internal circuit until after the PLL begins oscillating stably.

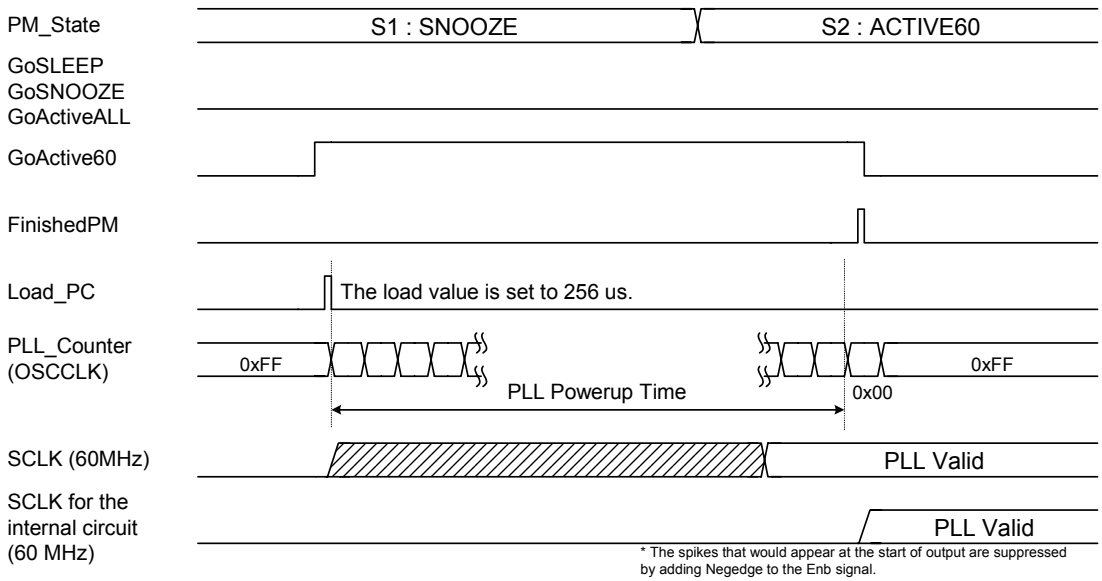


Figure 6-20 Exiting the SNOOZE State (during GoActive60)

6.2.3 ACTIVE60 (Active60)

This is the state in which the oscillator and PLL60 are operating but the PLL480 is inactive. The registers and bits shown in ***italic bold*** face in the register map can be read and written to even during Snooze and Sleep. All other registers cannot be read and written to except in the Active60 or ActiveAll state, so that when they are read, all read values are 0 s.

Note also that as the USB circuit requires SCLK480, it operates in only the ActiveAll state. The IDE and CPU circuits do not require SCLK480, so they can operate in both the Active60 and ActiveAll states. As the 480-MHz PLL consumes less current than the 60-MHz PLL, if the USB is unnecessary, it is possible to reduce the amount of power consumed in the chip by entering the Active60 state.

6.2.4 ACTIVEALL (Active480)

This is the state in which the oscillator, PLL60, and PLL480 are operating. The registers and bits shown in ***italic bold*** face in the register map can be read and written to even during Snooze and Sleep. All other registers cannot be read and written to except in the Active60 or Active480 state, so that when they are read, all read values are 0 s.

Note also that as the USB circuit requires SCLK480, it operates in only the ActiveAll state. The IDE and CPU circuits do not require SCLK480, so they can operate in both the Active60 and ActiveAll states. As the 480-MHz PLL consumes less current than the 60-MHz PLL, if the USB is unnecessary, it is possible to reduce the amount of power consumed in the chip by entering the Active60 state.

6.3 FIFO Management

6.3.1 FIFO Management

This section describes FIFO management.

6.3.1.1 FIFO Memory Map

The following shows the FIFO memory map.

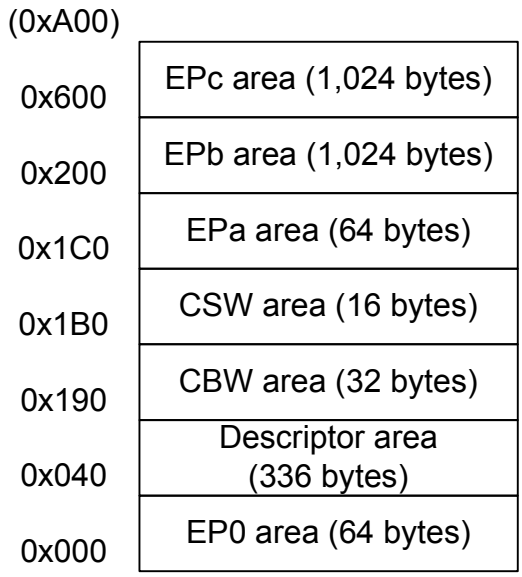


Figure 6-21 FIFO Memory Map

The FIFO memory is divided into seven areas—the EP0 area, descriptor area, CBW area, CSW area, EPa area, EPb area, and EPc area. A fixed amount of storage is allocated for each of these areas.

The EP0 area is provided for endpoint 0, which is indispensable for the USB, and is used for operation in both the IN and OUT directions. This area has 64 bytes of storage reserved for it, of which a finite amount of storage equal to the max. packet size of endpoint 0 beginning with the address 0x000 can be used. Therefore, endpoint 0 is always a single buffer.

The descriptor area is provided for use by the descriptor reply function. This area has 332 bytes of storage reserved for it, which can be used from any address in it. The actual method for using this area is described later in Section 6.3.1.2. Actually, any FIFO area can be set for use by the descriptor reply function; however, to avoid conflicts, we recommend that the area shown above be used as the descriptor area.

The CBW area is used for CBW support of the bulk-only support function. This area has 32 bytes of storage reserved for it, of which 31 bytes of storage beginning with the address 0x190 are used. The actual method for using this area is described later in Section 6.3.1.3.

The CSW area is used for CSW support of the bulk-only support function. This area has 16 bytes of storage reserved for it, of which 13 bytes of storage beginning with the address 0x1B0 are used. The actual method for using this area is described later in Section 6.3.1.4.

The EPa, EPb, and EPc areas are general-purpose endpoint areas whose endpoint numbers and directions (IN or OUT) can be set as desired. The EPa area has 64 bytes of storage reserved for it, so that it can be used for FS-mode bulk transfers or HS/FS-mode interrupt

transfers. The EPb and EPc areas have 1,024 bytes of storage reserved for each, so they can be used for HS-mode bulk transfers in addition to the above.

The EP0, EPa, EPb, and EPc areas are controlled as FIFO, so that the number of data bytes stored in each is retained. To clear this retained status, set the EPnControl.AllFIFO_Clr bit or the EPnControl.EP0FIFO_Clr and EPrFIFO_Clr.EPx{x=a-c}FIFO_Clr bits provided for each area.

This status-clearing operation only initializes the data-retention information, and does not write or clear data to or from the area. Therefore, in no case will the data in RAM be cleared by these bits; therefore, the information recorded in the descriptor area will never be lost and there is no need to write data again after clearing the status.

6.3.1.2 Method for Using the Descriptor Area

The descriptor area is provided for use by the descriptor reply function. The descriptor reply function can be used when the data stage is executed in IN transfer at endpoint 0.

To execute a data stage in the IN direction, set the start address of the data written into this area and the data size to be returned, and then execute the descriptor reply function. The data stage will be automatically executed.

This area may be used to write the content of uniquely determined equipment data, as in the case of a device descriptor. Once such data is written into this area during initialization after power-on, for example, it is possible to instruct that the data in this area be returned when a request is accepted. In this way, requests can be responded to promptly, as there is no need to write data into the EP0 area for each request.

6.3.1.2.1 Writing Data into the Descriptor Area

To write data into the descriptor area, use the RAM_WrDoor function. Set the write start address in the RAM_WrArs_H,L registers, and then write data into the RAM_WrDoor_H,L registers. The RAM_WrArs_H,L register values are updated for every data bytes written each time the registers are written to; therefore, if data needs to be written to contiguous addresses, the data can be written to the RAM_WrDoor_H,L registers successively.

Note that the RAM_WrDoor_H,L registers are write-only registers.

6.3.1.2.2 Executing a Data Stage (IN) in the Descriptor Area

To use written data in the descriptor reply function, set the start address of the data to be transmitted to the data stage in the DescAdrs_H,L registers and the data size to be returned in the DescSize_H,L registers, respectively, and then set the EP0Control.ReplyDescriptor bit to 1. Furthermore, set the EP0Control.INxOUT bit to 1 in order to permit an IN transaction to be performed. To ensure that data packets will be sent back to an IN transaction in the data stage, confirm that SETUP_Control.ProtectEP0 is cleared before the EP0Control_IN.ForceNAK bit is cleared.

After settings are made, data packets are sent back to the host in response to an IN transaction from the host while being automatically divided into the max. packet size (set by EP0MaxSize), until the number of data bytes set in the DescSize_H,L registers is reached. If the DescSize_H,L register value is less than the max. packet size or the remaining data bytes after being divided are less than the max. packet size, the data is automatically transmitted as a short packet.

When an OUT transaction is issued from the host, the EP0Control.ReplyDescriptor bit is cleared and the EPrIntStat.DescriptorCmp is set. The firmware should perform processing of the status stage.

6.3.1.3 Method for Using the CBW Area

The CBW area is used for CBW support of the bulk-only support function. When a command transport of the Bulk-Only Transport Protocol is to be performed at the Bulk Only endpoint (endpoint EPb or EPc), data can be received in this area. In this way, control of IDE or DMA transfers can be facilitated, as only data needs be received at the endpoint FIFO.

6.3.1.3.1 Receiving in the CBW Area

If, when CBW support is being executed, an OUT transaction is performed at the target endpoint and the data size is 31 bytes, the data is received in the CBW area. If the data is not 31 bytes in length, an error status is issued and the data is discarded.

6.3.1.3.2 Reading Data from the CBW Area

To read the data received in the CBW area, use the RAM_Rd function. When the RAM_RdControl.RAM_GoRdCBW bit is set, data is read from the CBW area and copied to the RAM_Rd_00 through RAM_Rd_1E registers, upon completion of which a completion status (FIFO_IntStat.RAM_RdCmp bit) is issued.

6.3.1.4 Method for Using the CSW Area

The CSW area is used for CSW support of the bulk-only support function. When a command transport of the Bulk-Only Transport Protocol is to be performed at the Bulk IN endpoint (endpoint EPb or EPc), data can be transmitted from this area. In this way, control of IDE or DMA transfers can be facilitated, as only data needs to be transmitted from the endpoint FIFO.

6.3.1.4.1 Transmitting from the CSW Area

If, when CSW support is being executed, an IN transaction is performed at the target endpoint, 13 bytes of data are transmitted from the CSW area as a data packet.

6.3.1.4.2 Writing Data into the CSW Area

To write data into the CSW area, use the RAM_WrDoor function. Write the start address of the CSW area (0x1B0) into the RAM_WrAdr_H,L registers, and write 13 bytes of valid data via the RAM_WrDoor_H,L registers. As 16 bytes of storage are reserved for the CSW area, there is no possibility of affecting other areas even when 14 bytes of data are written into the area.

6.3.2 Method for Accessing the FIFO

There are several factors accessible to the FIFO. These include the CPU (registers), CPU (DMA), IDE, and USB.

6.3.2.1 Method for Accessing the FIFO (RAM_Rd)

To access the FIFO for reading via the RAM_Rd register of the CPUIF, set the start address of the FIFO area from which to read and the data size in the RAM_RdAdr_H,L registers

and RAM_RdCount register, respectively, and then set the RAM_RdControl.RAM_GoRd bit. When the data in the specified FIFO area is ready for read from the RAM_Rd register, the FIFO_IntStat.RAM_RdCmp bit is set to 1. After confirming the RAM_RdCmp bit, read data from the RAM_Rd register. The data in the RAM_Rd register is stored in accordance with the register numbers, beginning with RAM_Rd_00. If the data size set in the RAM_RdCount register is smaller than 32 bytes, the data bytes in the RAM_Rd register after the set size is reached are ignored.

If the FIFO data is to be read via the RAM_Rd register, FIFO area settings for the respective endpoints do not need to be made.

The RAM_Adrs_H,L and RAM_Count register values are updated each time while the RAM_Rd function is in operation. Once the RAM_Rd function is activated, do not access these registers until the FIFO_IntStat.RAM_RdCmp bit is set. If these registers are accessed for reading while the RAM_Rd function is in operation, the read values cannot be guaranteed. Writing to these registers will cause the device to operate erratically.

6.3.2.2 Method for Accessing the FIFO (RAM_WrDoor)

To access the FIFO for writing via the RAM_WrDoor_H,L registers of the CPUIF, set the write start address in the RAM_WrAdr_H,L registers and write data via the RAM_WrDoor_H,L registers. The RAM_WrAdr_H,L registers are automatically incremented by an amount equal to the number of written bytes each time the FIFO is accessed for writing; therefore, if data needs to be written to contiguous addresses, the data can be written to the RAM_WrDoor_H,L registers successively.

If data is to be written to the FIFO via the RAM_WrDoor H,L registers, FIFO area settings for the respective endpoints do not need to be made.

6.3.2.3 Method for Accessing the FIFO (Register Access)

To access the FIFO for reading by means of a register access of the CPU:
Set EPx{x=0, a-c}Join.JoinCPU_Rd for any one of the endpoints to 1, and read out data via the EPnFIFO_Rd or EPnFIFO_ByteRd register.

To access the FIFO for writing by means of a register access of the CPU:
Set EPx{x=0, a-c}Join.JoinCPU_Wr for any one of the endpoints to 1, and write data in the EPnFIFO_Wr register.

The EPnRdRemain_H,L registers indicate the remaining number of data bytes that can be read from the FIFO for only one endpoint that has been set by EPx{x=0, a-c}Join.JoinCPU_Rd. Similarly, the EPnWrRemain_H,L registers indicate the remaining number of FIFO areas to which data can be written for only one endpoint that has been set by EPx{x=0, a-c}Join.JoinCPU_Wr.

Be aware that if registers are to be dumped when the firmware is debugged using an ICE or other tool, and if any EPx{x=0, a-c}Join.JoinCPU_Rd register has been set in such a case, data may inadvertently be read from the FIFO when registers are dumped.

6.3.2.4 Method for Accessing the FIFO (DMA)

To access the FIFO for reading by means of a DMA access of the CPU:
Set the EPn{n=0, a-c}Join.JoinDMAx{x=0,1}_Rd register for any one of the endpoints per DMA channel to 1, and execute a DMA procedure to read data.

To access the FIFO for writing by means of a DMA access of the CPU:
Set the EPn{n=0, a-c}Join.JoinDMAx{x=0,1}_Wr register for any one of the endpoints per DMA channel to 1, and execute a DMA procedure to write data.

The DMAx{x=0,1}_Remain_H,L registers indicate the remaining number of data bytes that can be read from the FIFO for only one endpoint per DMA channel that has been set by EPn{n=0, a-c}Join.JoinDMAx{x=0,1}_Rd. Similarly, they indicate the remaining number of FIFO areas to which data can be written for only one endpoint per DMA channel that has been set by EPn{n=0, a-c}Join.JoinDMAx{x=0,1}_Wr.

6.3.2.5 Method for Accessing the FIFO (IDE)

For the FIFO to be accessed by the IDE, set EPx{x=0, a-c}Join.JoinIDE for any one of the endpoints to 1, and execute an IDE procedure to transfer data. The direction of IDE transfer is set in accordance with the direction of the connected endpoint.

6.3.2.6 Limitations on FIFO Access

The FIFO of the LSI stipulated herein is designed in such a way that transmission/reception using the USB, register- or DMA-read/write from the CPU bus, and transmission/reception to/from the IDE are performed at the same time. Furthermore, read from the CPU bus is accomplished by look-ahead processing.

For the above reasons, the method of setup (Join) for access to the FIFO at the respective endpoints is subject basically to the following exclusion rules:

- For one endpoint, only one of writing factors can be set.
- For one endpoint, only one of reading factors can be set.
- Only one of JoinCPU_Wr, JoinCPU_Rd, JoinDMAx{x=0,1}_Wr, or JoinDMAx{x=0,1}_Rd can be set for one endpoint.
- JoinIDE, JoinCPU_Wr, JoinCPU_Rd, JoinDMAx{x=0,1}_Wr, and JoinDMAx{x=0,1}_Rd can respectively be set for only one endpoint at a time.
- JoinCPU_Rd and JoinCPU_Wr can not be set at the same time.

There is an exception for the rule that says only one writing/reading factors can be set to one endpoint. For example, although it is possible to write to an OUT-endpoint FIFO area after setting JoinCPU_Wr, no OUT transactions must be in progress when the JoinCPU_Wr is set and then also the FIFO is written to from the CPU. Similarly, although it is possible to read from an IN-endpoint FIFO area after setting JoinCPU_Rd, no IN transactions must be in progress when the JoinCPU_Rd is set and then also the FIFO is read from the CPU. It can be confirmed that no transactions are being performed by the fact that the ActiveUSB bit is cleared, EnEndpoint for the endpoint concerned is cleared, or ForceNAK is set.

6.4 CPUIF

6.4.1 Mode Switching

The CPUIF of the S1R72V03 accommodates asynchronous CPUs, and has the following three operation modes.

Table 6-15 CPUIF Operation Mode Settings

Operation Mode	BusMode	Bus8x16	Remark
16bit Strobe mode	0	0	Default
16bit BE mode	1	*	BusMode bit settings have priority.
8bit mode	0	1	

Switching between these operation modes is accomplished by setting the BusMode and Bus8x16 bits in the ChipConfig register. The value of this ChipConfig register can be protected against erroneous writing by setting the ModeProtect register.

In actual use, first set the ChipConfig register immediately after power-on to determine the operation mode. Then, set the ModeProtect register to write-protect it.

Furthermore, the CPUIF of the S1R72V03 has a bus swap function. To use this function, set the ChipConfig.CPU_BusSwap bit when initially setting up the ChipConfig register. The swap function is enabled by reading the address E9h after setting the CPU_BusSwap bit. In addition to the above, it is possible to set the XINT logic level and output mode, the logic levels of XDREQ0,1 and XDACK0,1, and the CS_Mode of DMA0,1 by setting up the ChipConfig register.

In the description below, explanations are given assuming default settings (16-bit Strobe mode, no Bus Swap) unless otherwise noted.

6.4.2 Regarding Mode Switching

The S1R72V03 can be set for the CPU bus operation mode best suited for the CPU used in the user system by setting up the ChipConfig register. Immediately after initialization, the chip is operating in 16-bit Strobe mode. If the chip needs to be switched to 16-bit BE mode or 8-bit mode, note the precautions on mode switching given below.

6.4.2.1 When Using 16-bit BE Mode

To use 16-bit BE mode as suited to the CPU used, first set up the ChipConfig register as described in Section 6.4.1. In such a case, furthermore, be sure to write to the address EFh bytewise as shown in Figure 6-22. At this time, the S1R72V03 is operating in 16-bit Strobe mode or its default mode; if there is skew like that shown below between the CPU signals Chip Select and Byte Mask High (XCS and XWRH), this skew may be misidentified as a valid write period internally in the chip. Although the S1R72V03 incorporates a filter circuit (min: 1 ns) to eliminate such skew, we recommend that the AC characteristics of the CPU used be taken into consideration in order to suppress such skew by taking preventive measures in the circuit board.

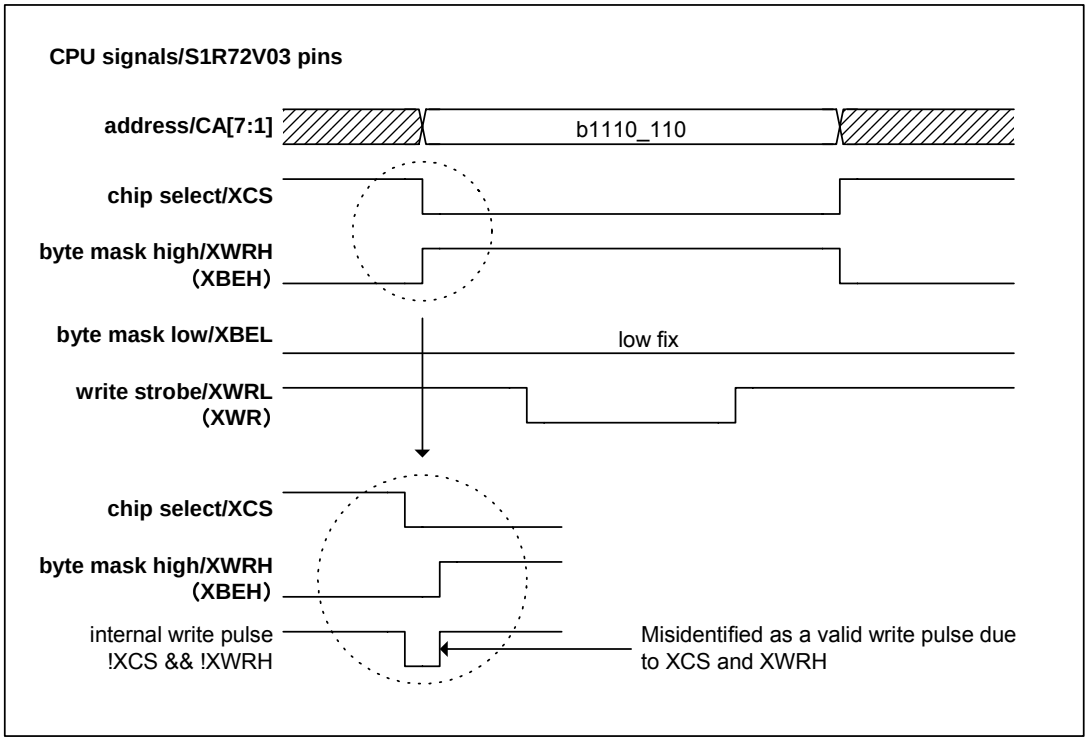


Figure 6-22 Initial Settings of the ChipConfig Register

Once operation-mode settings have been completed, this limitation does not apply, as the conditions for the internal write pulse generation are updated.

Furthermore, if the S1R72V03 is accessed for reading before the ChipConfig register is set up, internal read and write operations in the chip may inadvertently be performed at the same time, as shown in Figure 6-23. In such a case, device operation cannot be guaranteed, so be sure to set up the ChipConfig register first.

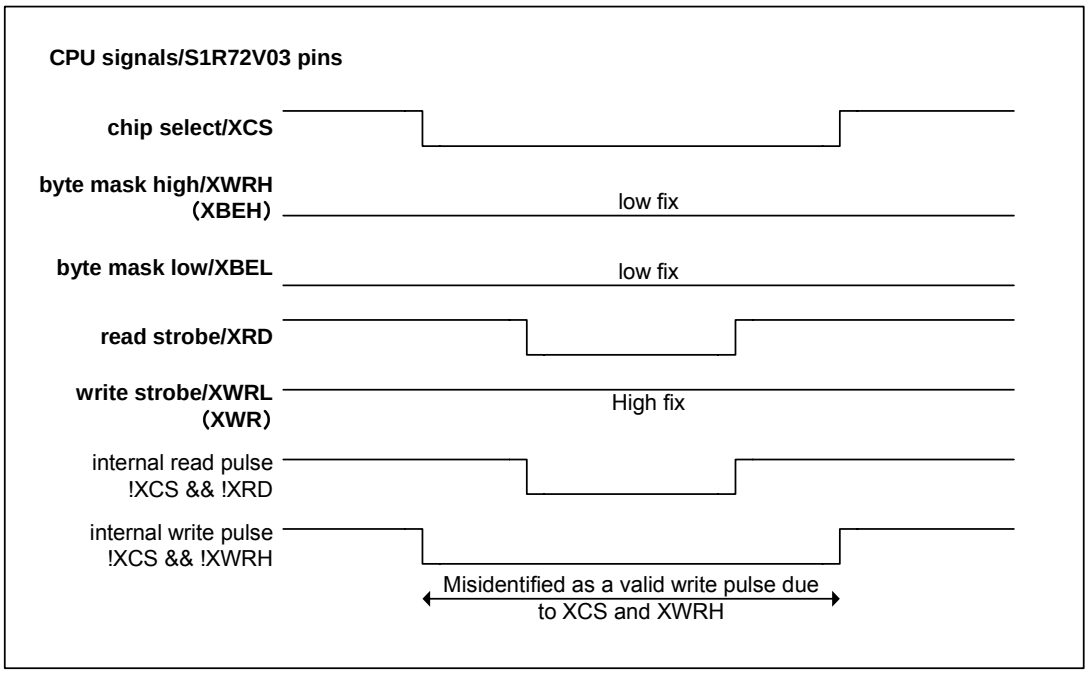


Figure 6-23 Read Access before Initial Settings of the ChipConfig Register

6.4.2.2 When Using 8-bit Mode

To use 8-bit mode as suited to the CPU used, first set up the ChipConfig register as described in Section 6.4.1. If the S1R72V03 is accessed for reading before the ChipConfig register is set up, as the S1R72V03 is operating in 16-bit Strobe mode or its default mode, all of the CD[15:0] pins are set for output. As a result, although there will be no specific problems if CD[15:8] are pulled high or low through resistors, the amount of current consumed in the chip may increase significantly if those pins are tied directly to VDD or GND. To avoid this problem, be sure to set up the ChipConfig register first.

6.4.3 Block Configuration

The block configuration of the CPUIF of the S1R72V03 (hereinafter referred to simply as the "CPUIF") is shown in Figure 6-24.

The CPUIF is comprised of three blocks—REG, DMA0, and DMA1.

- REG: Controls access to the S1R72V03 register area
- DMA0: DMA channel 0
- DMA1: DMA channel 1

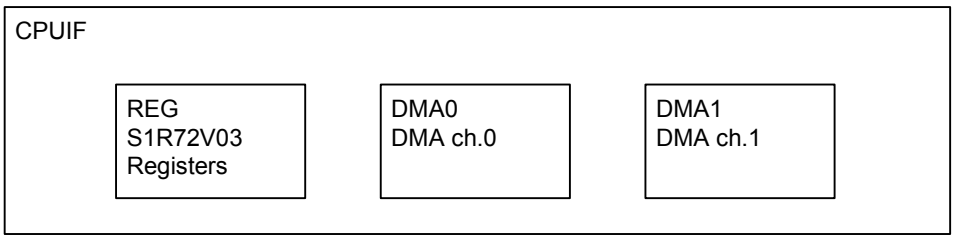


Figure 6-24 Block Configuration

6.4.3.1 REG (S1R72V03 Registers)

Controls access to the S1R72V03 register area. This includes the following access functions:

- Synchronous register access
- FIFO access
- RAM_Rd access
- Asynchronous register access

6.4.3.1.1 Synchronous Register Access (Write)

In this access, external bus data is written to the registers synchronously with the internal clock.

6.4.3.1.2 Synchronous Register Access (Read)

In this access, the register data is output to the external bus when both XCS and XRD are asserted by assuming that period as the output enable period.

In a register read operation, registers that accommodate 3 bytes or more as meaningful data, as in the case of a count value (for 8-bit mode, 2 bytes or more), require caution, as it is possible that an erroneous count value will be read that may occur during the access cycle. To avoid this problem, the lower-byte register value is latched when the most significant

byte is read, and the latched value is output on to the external bus when the lower bytes are read.

6.4.3.1.3 FIFO Access (Write)

" FIFO write access" refers to writing data to the EPnFIFO_Wr_H,L and RAM_WrDoor_H,L registers. During operation in 8-bit mode, either of the EPnFIFO_Wr_H,L registers can be accessed for writing to the FIFO without causing any problem. The same applies to the RAM_WrDoor_H,L registers.

FIFO access (write) is subject to the following limitations:

- After the EPx{x=0, a-c}Join.JoinCPU_Wr bit is set, the number of writable data bytes in the EPnWrRemain_H,L registers must be confirmed before the FIFO is accessed. This limitation does not apply to the RAM_WrDoor_H,L registers.
- When a 16-bit CPU is used, the FIFO must basically be accessed by the word (in 2-byte units). For the writing of odd bytes, the byte boundaries of the FIFO must be considered to control the strobe signal. For details, refer to Section 6.4.3.1.5, "Processing Odd Bytes in FIFO Access."
- If the EPnWrRemain_H,L registers are checked immediately after writing to the EPnFIFO_Wr_H,L registers, the read value will not show the exact amount of free space in the FIFO. Be sure to insert an interval equal to 1 CPU cycle or more before reading.
- If the RAM_WrDoorAdr_H,L registers are checked immediately after writing to the RAM_WrDoor_H,L registers, the read value will not show the exact address. Be sure to insert an interval equal to 1 CPU cycle or more before reading.

6.4.3.1.4 FIFO Access (Read)

" FIFO read access" refers to the reading out of data from the EPnFIFO_Rd_H,L and EPnFIFO_ByteRd registers. During operation in 8-bit mode, either the EPnFIFO_Rd_H or L, or EPnFIFO_ByteRd register can be accessed for reading from the FIFO without causing any problem.

FIFO read access is subject to the following limitations:

- After the EPx{x=0, a-c}Join.JoinCPU_Rd bit is set, the number of readable data bytes and the RdRemainValid bit in the EPnRdRemain_H,L registers must be checked before the FIFO is accessed.
- During operation in 16-bit mode, use the EPnFIFO_Rd_H,L registers to read data wordwise. Use the EPnFIFO_ByteRd register to read data bytewise. If byte boundaries exist, read out data bytewise. In such a case, if data is read wordwise using the EPnFIFO_Rd_H,L registers, valid data is output on only one side of the registers. For details, refer to the section "Processing Odd Bytes in FIFO Access."

6.4.3.1.5 Processing Odd Bytes in FIFO Access

This section describes how data is stored in the FIFO in relation to the FIFO access made when odd bytes of data are handled. Although the actual FIFO is 4 bytes in width, the FIFO in the explanation below is referred to as being 2 bytes in width for the sake of simplicity. There are no operational differences between 4 bytes and 2 bytes.

[Write operation]

Basically, we recommend that a write operation be performed from a byte-boundary nonexistent state.

If odd number data is found to be present after data is written wordwise from a byte-boundary nonexistent state by setting the EPrFIFO_Clr.EPx{x=a-c}FIFO_Clr bit and making other necessary settings, make sure that only the last byte of consecutive data (i.e., data Z) is written to the High side. This state is shown in (1) of Figure 6-25. The data is output from the USB in order of A, B, C, D, ... X, Y, and Z.

To write data while the FIFO has a byte boundary in it, first write data to the Low side (writing of data K) in order to eliminate the byte boundary, and then write data wordwise (data L and M). This state is shown in (2) of Figure 6-25.

The normal write operations are described above.

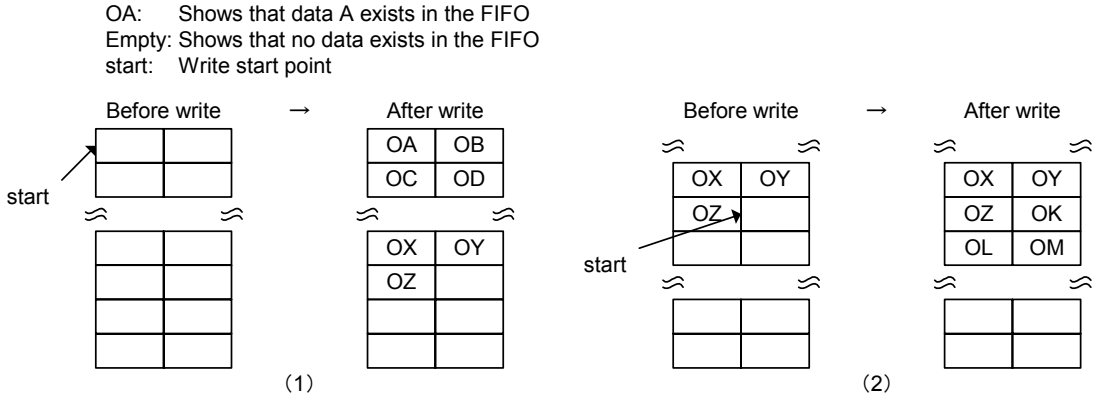


Figure 6-25 FIFO Write Processing (Normal Operations)

Described below are the write operations that require caution.

If data is written wordwise while the FIFO has a byte boundary in it, writing to the High side is ignored; data is written to only the Low side ((3) in Figure 6-26). This is the same as data being written to the Low side bytewise. Furthermore, if data is written to only the High side while the FIFO has a byte boundary in it, the writing operation performed is ignored ((4) in Figure 6-26).

If data is written to only the Low side while the FIFO has no byte boundaries in it, the write operation performed is ignored ((5) in Figure 6-26). Furthermore, if data is written wordwise while the FIFO has no byte boundaries in it and the number of writable bytes is 1, writing to the Low side is ignored; data is written to only the High side ((6) in Figure 6-26). This is the same as data being written to the High side bytewise.

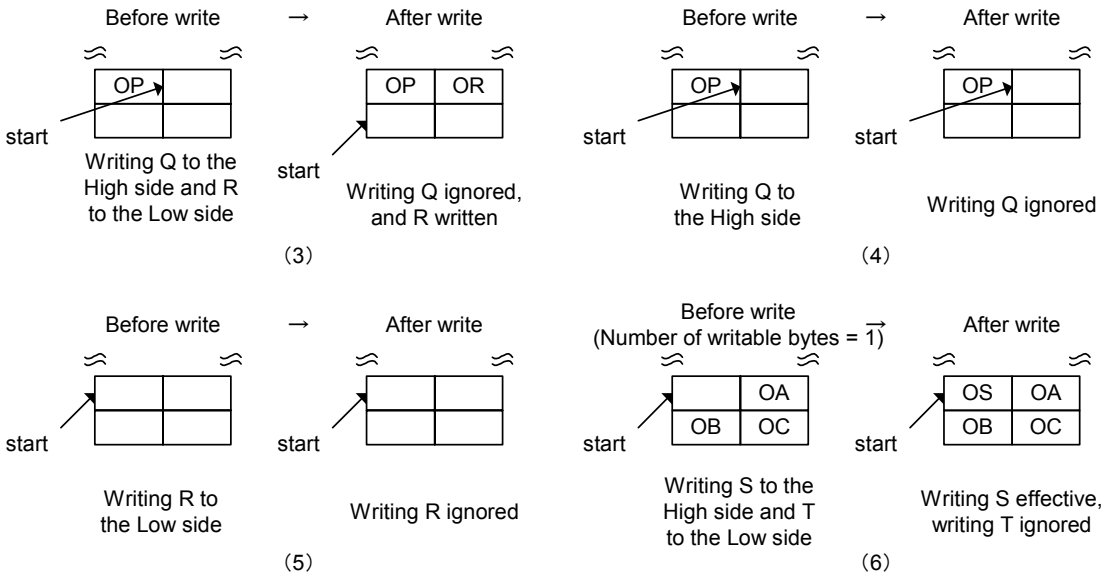


Figure 6-26 FIFO Write Processing (Operations Requiring Caution)

[Read operation]

If no byte boundaries exist, data can be read wordwise using the EPnFIFO_Rd_H,L registers or can be read bytewise using the EPnFIFO_ByteRd register without causing any problem. If any byte boundary exists, data must be read bytewise using the EPnFIFO_ByteRd register. Once the byte boundary is eliminated, data can be read either wordwise or bytewise without causing any problem.

How data is read out wordwise when no byte boundaries exist is shown in (1) of Figure 6-27. Data A,B and then data C,D are read out each time the FIFO is accessed. How data is read out bytewise is shown in (2) of Figure 6-27. Data A, data B, data C, and data D are read out each time the FIFO is accessed. Described above are the normal read operations.

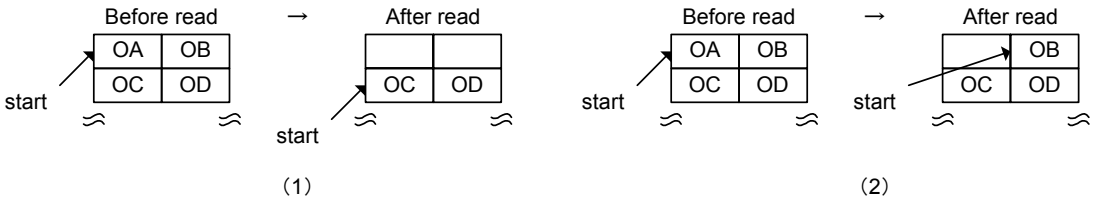


Figure 6-27 FIFO Read Processing (Normal Operations)

Described below are the read operations that require caution.

Shown in (3) of Figure 6-28 is the operation when data is read out wordwise using the EPnFIFO_Rd_H,L registers while a byte boundary exists. Indeterminate data is output to the High side, and data J is output to the Low side. The read pointer increments for only 1 byte of data. Shown in (4) of Figure 6-28 is the operation when data is read out wordwise using the EPnFIFO_Rd_H,L registers while no byte boundaries exist but the remaining bytes of data = 1. Data X is output to the High side, and indeterminate data is output to the Low side. The read pointer increments for only 1 byte of data.

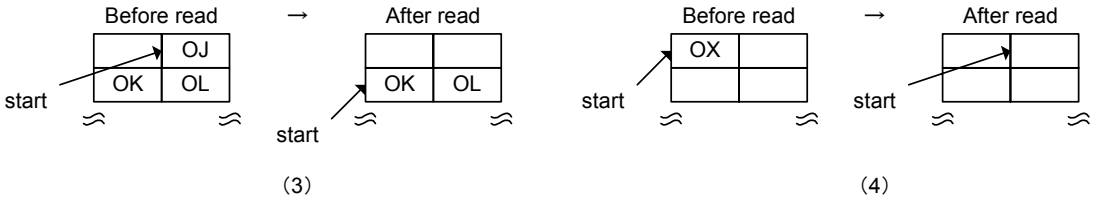


Figure 6-28 FIFO Read Processing (Operations Requiring Caution)

Based on the above, the following shows an example of read operation in odd-byte processing.

1) To read out 64 bytes of data sent from the USB, first 31 bytes and then 33 bytes
   (1) The CPUIF latches Ready for 64 bytes to start a series of read sequences.
   (2) The 30 bytes of data are read wordwise using the EPnFIFO_Rd_H,L registers or bytewise using the EPnFIFO_ByteRd register.
   (3) The 31st byte of data is read bytewise using the EPnFIFO_ByteRd register. -> A byte boundary is created.
   (4) The 32nd byte of data is read bytewise. In this case, it is recommended that the EPnFIFO_ByteRd register be used for byte reading. If the data is read wordwise using the EPnFIFO_Rd_H,L registers, the read data is output to the Low side. -> The byte boundary is eliminated.
   (5) The remaining 32 bytes of data are read wordwise using the EPnFIFO_Rd_H,L registers or bytewise using the EPnFIFO_ByteRd register.
2) To read out 64 bytes of data, sent separately in 31 bytes and then 33 bytes from the USB while JoinCPU_Rd is set, all wordwise using the EPnFIFO_Rd_H,L registers.
   (1) When 31 bytes of data have been received from the USB, the CPUIF latches Ready for 31 bytes to start a series of operation sequences.
   (2) The 30 bytes of data are read wordwise.
   (3) To eliminate the cached 31st byte of data (byte boundary), Join is temporarily disconnected.
   (4) After 33 bytes of data have been sent from the USB, Join is reconnected (1 + 33 bytes).
   (5) The CPUIF latches Ready for 34 bytes to start a series of operation sequences.
   (6) The 34 bytes of data are read wordwise.

6.4.3.1.6 RAM_Rd Access

As with synchronous register reading, data is output to the external bus when both XCS and XRD are asserted by assuming that period as the output enable period. For details, refer to the section “Method for Accessing the FIFO (RAM_Rd).”

6.4.3.1.7 Asynchronous Register Access (Write)

After a write pulse is created from the external write signals (XCS and XWRL,H), the external bus data is written to the registers.

6.4.3.1.8 Asynchronous Register Access (Read)

As with synchronous register reading, the register data is output to the external bus when both XCS and XRD are asserted by assuming that period as the output enable period.

6.4.3.2 DMA0/DMA1 (DMA Channels 0/1)

6.4.3.2.1 Basic Functionality

The basic operations of the DMA are described below.

[Write operation]

If the FIFO has writable free space, XDREQ is asserted to enable DMA transfers to be performed.

[Read operation]

When the FIFO has valid readable data and is readied for reading, XDREQ is asserted to enable DMA transfers to be performed.

The DMA has two operation modes and one operation option.

- Count mode
  DMA transfers are performed a number of times equal to the count set. When the internal FIFO contains writable free space or valid readable data and there is a remaining count in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers, XDREQ is asserted to enable DMA transfers to be performed.
- Free-running mode
  When the internal FIFO contains writable free space or valid readable data, XDREQ is asserted to enable DMA transfers to be performed.
- REQ assert count option
  This option is provided for burst read/burst writing by the CPU. This option can be used in either count mode or free-running mode. If the FIFO contains an amount of writable free space or valid readable data exceeding the assert counts set by the DMAx{x=0,1}_Config.ReqAssertCount [1:0] bits, XDREQ is asserted to enable DMA transfers to be performed. This means that once XDREQ is asserted, transfers for bytes equal to the set assert counts are guaranteed. However, even when the amount of free space or data in the FIFO is less than the set assert counts, if count mode is selected and the amount of said free space or data in the FIFO is greater than the remaining counts, XDREQ is asserted. In such a case, the guaranteed transfer bytes equal the remaining counts.

In 16-bit mode, DMA is basically data processed by the word unit. Data processing by the byte unit can be performed only when DMA is operating in count mode and the remaining counts = 1. The table below lists the relationship between XDREQ assert conditions and the number of transfers performed in each operation mode with the option used or unused.

Table 6-16 Operation Modes and Options vs. Transfer Start Conditions

Count mode with the ReqAssertCount option used (during operation in both 16-bit and 8-bit modes)

Condition	Count mode (Count > 0)			
	Count≧Req		Count＜Req	
	Ready≧Req	Ready＜Req	Ready≧Count	Ready＜Count
XDREQ	Asserted	Negated	Asserted	Negated
Number of transferable bytes	Req	-	Count	-

Free-running mode with the ReqAssertCount option used (during operation in both 16-bit and 8-bit modes)

Condition	Free-running mode	
	-	
	Ready≧Req	Ready＜Req
XDREQ	Asserted	Negated
Number of transferable bytes	Req	-

Count mode with the ReqAssertCount option unused (during operation in 16-bit mode)

Condition	Count mode (Count > 0)		
	Count≧Ready		Count＜Ready
	Ready≧2	Ready＜2	Ready≧Count
XDREQ	Asserted	Negated	Asserted
Number of transferable bytes	Ready (if Ready is an odd number, Ready - 1)	-	Count

Free-running mode with the ReqAssertCount option unused (during operation in 16-bit mode)

Condition	Free-running mode	
	-	
	Ready≧2	Ready＜2
XDREQ	Asserted	Negated
Number of transferable bytes	Ready (if Ready is an odd number, Ready - 1)	-

Count mode with the ReqAssertCount option unused (during operation in 8-bit mode)

Condition	Count mode (Count > 0)		
	Count≧Ready		Count＜Ready
	Ready≧1	Ready＜1	Ready≧Count
XDREQ	Asserted	Negated	Asserted
Number of transferable bytes	Ready	-	Count

Free-running mode with the ReqAssertCount option unused (during operation in 8-bit mode)

Condition	Free-running mode	
	-	
	Ready≧1	Ready＜1
XDREQ	Asserted	Negated
Number of transferable bytes	Ready	-

* In the above table, Req indicates the set value of DMAx{x=0,1}_Config.ReqAssertCount, Ready indicates the free space/data bytes in the FIFO, and Count indicates the value of DMAx{x=0,1}Count_HH,HL,LH,LL.

6.4.3.2.2 Pin Settings

It is possible to set each of the XDREQx{x=1,0} and XDACKx{x=1,0} logic levels by setting up the ChipConfig register. In the explanation below, both XDREQ and XDACK are described as being active-low (negative logic) unless otherwise noted.

6.4.3.2.3 Access Mode Settings

It is possible to set an operation condition that is assumed to be a DMA access by setting the ChipConfig.CS_Mode and DMAx{x=1,0}_Config.DMA_Mode bits.

Table 6-17 Access Mode Settings

CS_Mode	DMA_Mode	Description
0	0	DMA access is assumed when XDACK is asserted in response to XDREQ.
1	0	DMA access is assumed when XDACK and XCS are asserted in response to XDREQ.
*	1	DMA access is assumed when the DMAx{x=1,0}_Rd/WrData register is accessed in response to XDREQ.

The explanation below is given with respect to an access mode in which the assertion of XDACK and XCS is assumed to be an operation condition unless otherwise noted.

6.4.3.2.4 Count Mode (Write)

[Starting operation]

After setting a count value in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers, set the DMAx{x=0,1}_Control_DMA_Go bit to 1. If the internal FIFO contains 2 bytes or more of writable free space (DMA_Ready) (for 8-bit mode, 1 byte or more) and has any remaining counts, XDREQ is asserted to enable DMA transfers to be performed. If the free space available in the FIFO totals only 1 byte, count mode is selected, in which case XDREQ is asserted only when the remaining counts = 1.

If a byte boundary is created in the FIFO as a result of odd bytes written into it, clear the FIFO after data is transferred from the USB, etc. in order to eliminate the byte boundary before starting the next write operation. To transfer data from the USB 31 bytes at a time after writing data from the DMA 31 bytes at a time, for example, (1) set the DMA count value to 31 and write 31 bytes of data, (2) wait until 31 bytes of data are transferred to the USB, (3) after confirming that 31 bytes of data have been transferred from the USB, clear the FIFO. Repeat the above operations.

Until the operation stops, the DMAx{x=0,1}_Control.DMA_Running bit reads "1."

[Stopping operation]

The following two conditions cause the operation to stop:

- DMA transfers equal to the counts set in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers are completed.
- The DMAx{x=0,1}_Control.DMA_Stop bit is set by writing 1 in the software.

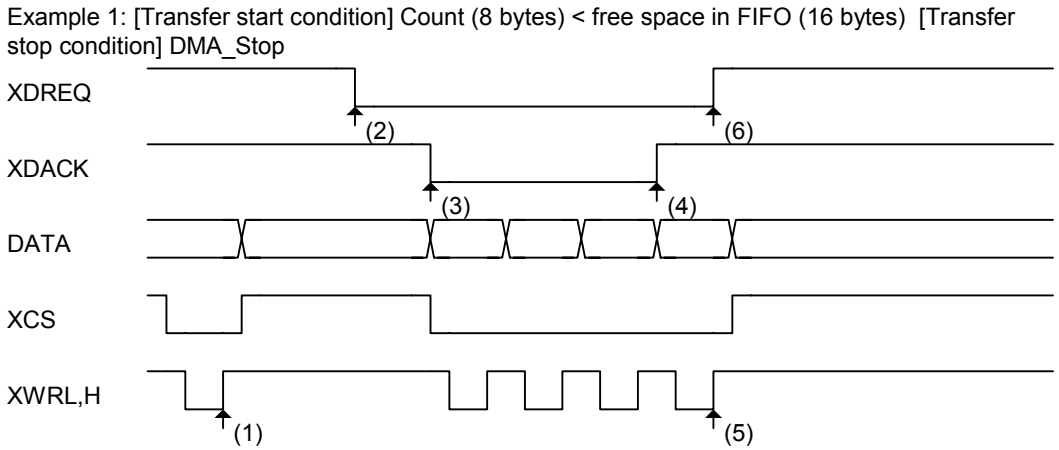
When the DMA operation stops, the CPU_IntStat.DMAx{x=0,1}_Cmp bit is set.

If the transfer has stopped due to the fact that the counts set in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers have expired, XDREQ is negated during the strobe assert period of the last access.

If the transfer has stopped due to the fact that the DMA_Stop bit is set, the chip's internal operation is halted in synchronism with the write timing of a synchronous register access,

and XDREQ is negated. To have the DMA stopped by the DMA_Stop bit, stop the DMAC (master) on the CPU side first.

Figure 6-29 shows the operation timing in cases in which a transfer is started in count mode and the transfer is stopped by setting the DMAx{x=0,1}_Control.DMA_Stop bit before transfers for the set number of counts are completed.

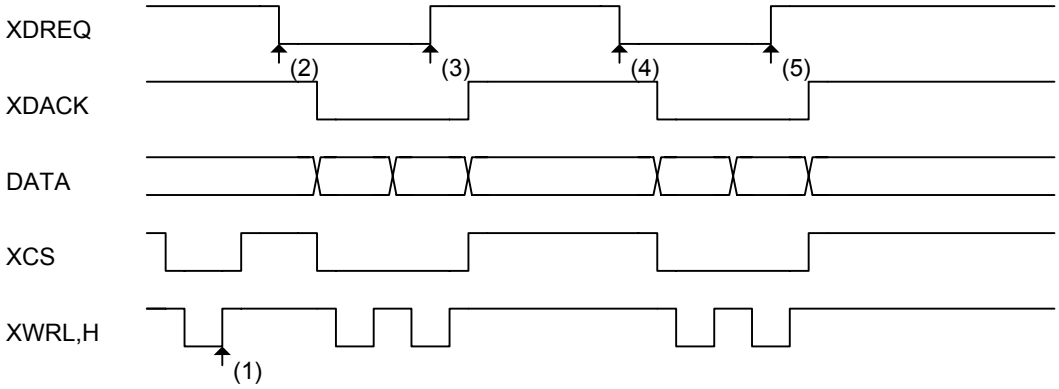


(1) The DMA circuit is activated by writing 1 to the DMA_Control.DMA_Go bit.
(2) Free space is created in the FIFO (DMA_Ready) due to the fact that data has been transferred from the USB, etc., and XDREQ is asserted in response to DMA_Ready.
(3) XDACK is asserted, causing a DMA transfer to start.
(4) The master is stopped and XDACK is negated before transfers in count mode are completed.
(5) The DMA circuit is stopped by writing 1 to the DMA_Control.DMA_Stop bit.
(6) XDREQ is negated in response to deactivation of the DMA circuit.

Figure 6-29 Count Mode Write Timing 1

Figure 6-30 shows the operation timing in cases in which a transfer is started in count mode and, due to the fact that transfers for the set number of counts has been completed, the DMA transfer has finished.

Example 2: [Transfer start condition] Count (8 bytes) < free space in FIFO (4 bytes) [Transfer stop condition] Count 0



(1) The DMA circuit is activated by writing 1 to the DMA_Control.DMA_Go bit.
(2) Free space is created in the FIFO (DMA_Ready) due to the fact that data has been transferred from the USB, etc., and XDREQ is asserted in response to DMA_Ready.
(3) XDREQ is negated in synchronism with the disappearance of DMA_Ready.
(4) Free space is created in the FIFO (DMA_Ready) due to the fact that data has been transferred from the USB, etc., and XDREQ is asserted in response to DMA_Ready.
(5) XDREQ is negated in synchronism with the last data timing of DMA_Count. The DMA circuit stops due to the fact that transfers equal to DMA_Count have been completed.

Figure 6-30 Count Mode Write Timing 2

6.4.3.2.5 Count Mode (Read)

[Starting operation]

After setting a count value in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers, set the DMAx{x=0,1}_Control_DMA_Go bit to 1. If the internal FIFO contains 2 bytes or more of valid readable data (for 8-bit mode, 1 byte or more) and has any remaining counts, the FIFO is readied for reading from an external device and XDREQ is asserted to enable DMA transfers to be performed. If the remaining data available in the FIFO totals only 1 byte, count mode is selected, in which case XDREQ is asserted only when the remaining counts = 1.

In a count-mode read operation, when bytes of data exceeding the counts set in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers have accumulated in the FIFO for the endpoint to which the present DMA is connected, the ForceNAK bit is automatically set to 1 in order to return an NAK response. Furthermore, even when a short packet is received from the USB, unless the DisAF_NAK_Short has been set, the ForceNAK bit for the relevant endpoint is automatically set to 1 in order to return an NAK response.

If a byte boundary is created in the FIFO as a result of odd bytes read from it, clear the FIFO to eliminate the byte boundary before performing the next transfer. To read out data from the DMA 31 bytes at a time after data has been transferred from the USB 31 bytes at a time, for example, (1) receive 31 bytes of data from the USB (at this point in time, the ForceNAK is set and the relevant endpoint returns a NAK response), (2) read out 31 bytes of data from the DMA, (3) clear the FIFO and then the ForceNAK to allow for transfers from the USB to be received. Repeat the above operations.

Until the operation stops, the DMAx{x=0,1}_Control.DMA_Running bit reads “1.”

[Stopping operation]

The following two conditions cause the operation to stop:

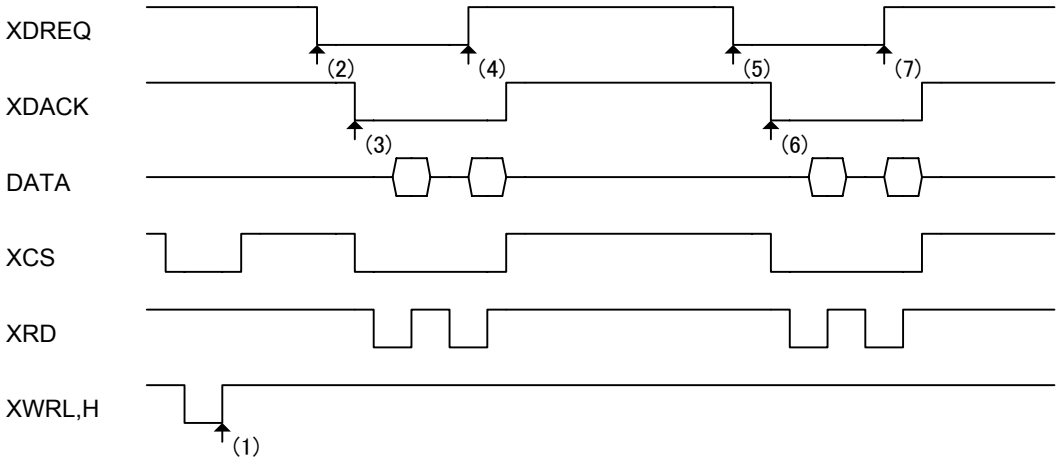
- DMA transfers equal to the counts set in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers are completed.
- The DMAx{x=0,1}_Control.DMA_Stop bit is set by writing 1 in the software.

If the transfer has stopped due to the fact that the counts set in the DMAx{x=0,1}_Count_HH,HL,LH,LL registers have expired, XDREQ is negated during the strobe-signal assert period of the last access.

If the transfer has stopped due to the fact that the DMA_Stop bit is set, the chip's internal operation is halted simultaneously with the write timing of a synchronous register access, and then XDREQ is negated. To have the DMA stopped by the DMA_Stop bit, stop the DMAC (master) on the CPU side first.

Figure 6-31 shows the operation timing in cases in which a transfer is started in count mode and, due to the fact that transfers for the set number of counts have been completed, the DMA transfer has finished.

Example: [Transfer start condition] Count (8 bytes) > data in FIFO (4 bytes) [Transfer stop condition] Count 0



(1) The DMA circuit is activated by writing 1 to the DMA_Control.DMA_Go bit.
(2) When data is written into the FIFO from the USB, etc. and the FIFO is readied for reading from an external device, XDREQ is asserted.
(3) XDACK is asserted, causing a DMA transfer to start.
(4) XDREQ is negated in synchronism with the timing at which the FIFO is emptied.
(5) When data is written into the FIFO from the USB, etc. and the FIFO is readied for reading from an external device, XDREQ is asserted.
(6) XDACK is asserted, causing a DMA transfer to start.
(7) XDREQ is negated in synchronism with the last data timing of DMA_Count.

Figure 6-31 Count Mode Read Timing

6.4.3.2.6 Free-Running Mode (Write)

[Starting operation]

After setting the DMAx{x=0,1}_Config.FreeRun bit, set the DMAx{x=0,1}_Control_DMA_Go bit by writing 1. If the internal FIFO contains 2 bytes or more of writable free space (for 8-bit mode, 1 byte or more), XDREQ is asserted to enable DMA transfers to be performed. If the free space available in the FIFO totals only 1 byte, XDREQ is not asserted in free-running mode. If transfers need to be performed, refer to the explanation of count mode in the preceding section.

Until the operation stops, the DMAx{x=0,1}_Control.DMA_Running bit reads "1."

[Stopping operation]

The following shows the condition that causes the operation to stop:

- The DMAx{x=0,1}_Control.DMA_Stop bit is set by writing 1 in the software.

When the transfer has stopped due to the fact that the DMA_Stop bit is set, the chip's internal operation is halted in synchronism with the write timing of a synchronous register access, and then XDREQ is negated. To have the DMA stopped by the DMA_Stop bit, stop the DMAC (master) on the CPU side first.

If the DMAx{x=0,1}_Count_HH,HL,LH,LL register overflows during a DMA transfer in free-running mode, the CPU_IntStat.DMAx{x=0,1}_Countup bit is set. In this case as well, the DMA transfer is continued and the count in the DMAx{x=0,1}_Count_HH,HL,LH,LL restarts incrementing.

The operation timing in free-running mode is the same as in count mode, except that there are no limitations due to DMAx{x=0,1}_Count_HH,HL,LH,LL.

6.4.3.2.7 Free-Running Mode (Read)

[Starting operation]

After setting the DMAx{x=0,1}_Config.FreeRun bit, set the DMAx{x=0,1}_Control_DMA_Go bit by writing 1. If the internal FIFO contains 2 bytes or more of valid readable data (for 8-bit mode, 1 byte or more) and is readied for reading from an external device, XDREQ is asserted. If the remaining valid data in the FIFO totals only 1 byte, DMA transfer is not started. If it is necessary to perform transfers, refer to the explanation of count mode in the preceding section.

Until the operation stops, the DMAx{x=0,1}_Control.DMA_Running bit reads "1."

[Stopping operation]

The following shows the condition that causes the operation to stop:

- The DMAx{x=0,1}_Control.DMA_Stop bit is set by writing 1 in the software.

When the transfer has stopped due to the fact that the DMA_Stop bit is set, the chip's internal operation is halted in synchronism with the write timing of a synchronous register access, and then XDREQ is negated. To have the DMA stopped by the DMA_Stop bit, stop the DMAC (master) on the CPU side first.

If the DMAx{x=0,1}_Count_HH,HL,LH,LL register overflows during a DMA transfer in free-running mode, the CPU_IntStat.DMAx{x=0,1}_Countup bit is set. In this case as well,

the DMA transfer is continued and the count in the DMAx{x=0,1}_Count_HH,HL,LH,LL restarts incrementing.

The operation timing in free-running mode is the same as in count mode, except that there are no limitations due to DMAx{x=0,1}_Count_HH,HL,LH,LL.

6.4.3.2.8 REQ Assert Count Option (Write)

[Starting operation]

After setting assert counts using the DMAx{x=0,1}_Config.ReqAssertCount [1:0] bits, set the DMAx{x=0,1}_Control.DMA_Go bit to 1. If the internal FIFO contains more bytes of writable free space than the set assert counts, XDREQ is asserted to enable DMA transfers to be performed. As a result, once XDREQ is asserted, transfers for bytes equal to the set assert counts are guaranteed. However, even when the free space in the FIFO is less than the set assert counts, if count mode is selected and said free space in the FIFO is greater than the remaining counts, XDREQ is asserted. In such a case, the guaranteed transfer bytes equal the remaining counts.

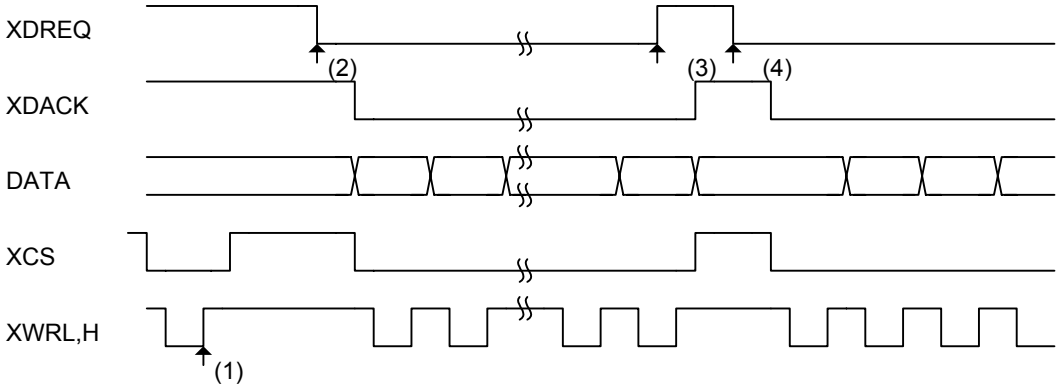
In this mode, XDREQ is temporarily negated every transfer of bytes set in the ReqAssertCount [1:0] bits.

Until the operation stops, the DMAx{x=0,1}_Control.DMA_Running bit reads "1."

[Stopping operation]

For the conditions that cause the operation to stop, refer to the explanation of count mode and free-running mode in the preceding sections.

Example: [Transfer start condition] REQ assert count (8-beat: 16 bytes)



(1) The DMA circuit is activated by writing 1 to the DMA_Control.DMA_Go bit.
As the DMA_Ready value is less than that required for successive transfers, XDREQ is not asserted.
(2) When data is transferred from the USB, etc. and valid free space greater than that required for successive transfers is created in the FIFO (DMA_Ready). XDREQ is asserted in response to DMA_Ready.
(3) XDREQ is negated in synchronism with the end timing of successive transfers (REQ assert count).
(4) Upon completion of the first round of successive transfers, free space for the next successive transfers is available (DMA_Ready). XDREQ is asserted in response to DMA_Ready.

Figure 6-32 REQ Assert Count Option Write Timing

6.4.3.2.9 REQ Assert Count Option (Read)

[Starting operation]

After setting assert counts using the DMAx{x=0,1}_Config.ReqAssertCount [1:0] bits, set the DMAx{x=0,1}_Control.DMA_Go bit to 1. If the internal FIFO has more bytes of valid readable data than the set assert counts and is readied for reading from an external device, XDREQ is asserted to enable DMA transfers to be performed. As a result, once XDREQ is asserted, transfers for bytes equal to the set assert counts are guaranteed. However, even when the data in the FIFO is less than the REQ assert counts, if count mode is selected and said data in the FIFO is greater than the remaining counts, XDREQ is asserted. In such a case, the guaranteed transfer bytes equal the remaining counts.

In this mode, XDREQ is temporarily negated every transfer of bytes set in the ReqAssertCount [1:0] bits.

Until the operation stops, the DMAx{x=0,1}_Control.DMA_Running bit reads "1."

[Stopping operation]

For the conditions that cause the operation to stop, refer to the explanation of count mode and free-running mode in the preceding sections.

For the operation timing, refer to Figure 6-31 and Figure 6-32.

6.4.3.2.10 FIFO Access Odd-Byte Processing in DMA

Refer to Section 6.4.3.1.5, "Processing Odd Bytes in FIFO Access." Note that the DMA has no entries for reading in bytes.

6.5 IDE I/F

This section describes the functions of the IDE I/F.

Before the IDE I/F functions can be used, the IDE_Config_1.ActiveIDE bit must be set by writing 1 in the software. Furthermore, to ensure that the endians in communication with IDE devices will be set correctly, write 1 to the IDE_Config_1.Swap bit. The description below is premised on an assumption that the IDE_Config_1.Swap bit is set to 1 unless otherwise noted. For details about the IDE_Config_1.Swap bit, refer to the detailed description of registers and Appendix A in this development specification.

6.5.1 Access to the IDE Task File Registers

This section describes the method for accessing the IDE task file registers.

The firmware can access the IDE task file registers via a group of registers from IDE_RegAdr to IDE_RegConfig. As read/write sequences on the IDE bus are performed in the hardware by the LSI until the operation is complete, the firmware is freed from the operation for accessing the IDE task file registers until it is notified of the termination of the sequence by polling or an interrupt after the command has been set.

When the IDE task file registers are accessed, if both XHCS0 and HDA[2:0] of the IDE bus = 0, the set value of the IDE_Tmod register is used for transfers performed; otherwise, the set value of the IDE_Rmod register is used. Therefore, an appropriate value must be set in the IDE_Tmod or IDE_Rmod register according to the IDE transfer mode before transfers start.

6.5.1.1 Reading from the IDE Task File Registers

Reading from the IDE task file registers can be performed by setting the address at which to access the IDE_RegAdrs.IDE_RegAddress[3:0] bits in advance or at the time of access, and then writing 1 to the IDE_ReqAdrs.IDE_RdReg bit.

The read operation is complete when the IDE_ReqAdrs.IDE_RdReg bit is cleared to 0 and the IDE_IntStat.IDE_RegCmp bit is set to 1.

The data read from the IDE task-file registers is stored in the IDE_RdRegValue register.

6.5.1.2 Writing to the IDE Task File Registers

Writing to the IDE task file registers can be performed by setting the data to be written in the IDE_WrRegValue register in advance, setting the address at which to access the IDE_RegAdrs.IDE_RegAddress[3:0] bits in advance or at the time of access, and then writing 1 to the IDE_ReqAdrs.IDE_WrReg bit.

The write operation is complete when the IDE_ReqAdrs.IDE_WrReg bit is cleared to 0 and the IDE_IntStat.IDE_RegCmp bit is set to 1.

6.5.1.3 Sequential Writing to the IDE Task File Registers

By using the IDE_SeqWrRegControl register, it is possible to write to the IDE task file registers sequentially for up to 16 pairs of address and data that have been set in advance.

Set the address at which to write in the IDE_SeqWrRegAdrs.IDE_SeqWrRegAddress[3:0] bits and the byte data to be written in the IDE_SeqWrRegValue register for up to 16 pairs in advance. As access is made wordwise when XHCS0 = 0 and HDA[2:0] = 0, data needs to be written twice in order of the lower byte (HDD[15:8]) and the upper byte (HDD[7:0]) (when IDE_Config_1.Swap = 1), so that two address and data pairs are used. Thereafter, a sequential write operation can be initiated by writing 1 to the IDE_SeqWrReqControl.IDE_SeqWrReg bit. The sequential write operation is complete when the IDE_SeqWrReqControl.IDE_SeqWrReg bit is cleared to 0 and the IDE_IntStat.IDE_SeqWrRegCmp bit is set to 1.

If the address and data that have been set in advance become unnecessary before a sequential write operation starts, those address and data can be discarded by writing 1 to the IDE_SeqWrReqControl.IDE_SeqWrRegClr bit.

If the firmware performs an ordinary write or read to or from the IDE task file registers during a sequential write operation, the IDE_IntStat.IDE_RegErr bit is set to 1 and the attempted write or read is ignored.

6.5.1.4 Auto Status Register Read from the IDE Task File Registers

If HINTRQ of the IDE bus is set and the IDE_ReqConfig.EnAutoStsRd bit has been set by writing 1 in the software, the LSI automatically reads the IDE status register (XHCS0 = 0, HDA[2:0] = 7) and stores the read data in the IDE_RdRegValue register. Thereafter, the IDE_IntStat.IDE_CompleteINTRQ bit is set to 1.

If the firmware performs an ordinary write or read to or from the IDE task file registers before it reads out the IDE_RdRegValue_L register after performing an auto status register read operation, the IDE_IntStat.IDE_RegErr bit is set to 1 and the attempted write or read is ignored.

6.5.2 PIO Access

This section describes the DMA functions in PIO mode.

The DMA in PIO mode uses the value set in the IDE_Tmod register for the DMA transfers performed. Therefore, an appropriate value must be set in the IDE_Tmod register according to IDE transfer mode before transfers start.

6.5.2.1 PIO Read DMA

The PIO read DMA operates in accordance with the procedure described below.

Write 0 to both the IDE_Config_0.DMA and IDE_Config_0.Ultra bits.

Set the number of transfer bytes in the IDE_CountH/M/L register.

Write 1 to the IDE_Control.IDE_Clr bit in order to initialize the IDE circuit. (This step may not always be required.)

Write 1 to the IDE_Control.IDE_Go bit in order to start a DMA operation. The content of the IDE_CountH/M/L register decrements as the transfer proceeds and, when the count reaches 0 and the DMA operation of the IDE and writing of the data read from the IDE into the internal FIFO are complete, the IDE_Control.IDE_Go bit is cleared to 0 and the IDE_IntStat.IDE_Cmp bit is set to 1, causing the DMA operation to end.

If the IDE_Control.IDE_Go bit is cleared by writing 0 while a DMA transfer is underway, the DMA transfer is aborted and the DMA is terminated. In such a case, although the data that has been transferred into the internal FIFO up to that point is valid, the data that has accumulated in the intermediate buffer of the circuit is discarded and the number of transferred bytes of the IDE therefore cannot be managed precisely.

6.5.2.2 PIO Write DMA

The PIO write DMA operates in accordance with the procedure described below.

Write 0 to both the IDE_Config_0.DMA and IDE_Config_0.Ultra bits.

Set the number of transfer bytes in the IDE_CountH/M/L register.

Write 1 to the IDE_Control.IDE_Clr bit in order to initialize the IDE circuit. (This step may not always be required.)

Write 1 to the IDE_Control.IDE_Go bit in order to start a DMA operation. The content of the IDE_CountH/M/L register decrements as the transfer proceeds and, when the count reaches 0 and the DMA write to the IDE is complete, the IDE_Control.IDE_Go bit is cleared to 0 and the IDE_IntStat.IDE_Cmp bit is set to 1, causing the DMA operation to end.

If the IDE_Control.IDE_Go bit is cleared by writing 0 while a DMA transfer is underway, the DMA transfer is aborted and the DMA is terminated. In such a case, although the data that has been written to the IDE by that time is valid, the data that has accumulated in the intermediate buffer of the circuit is discarded and the number of transferred bytes of the IDE therefore cannot be managed precisely.

6.5.3 Multi-Word DMA

This section describes the DMA functions in Multi-Word DMA mode.

DMA in Multi-Word DMA mode uses the value set in the IDE_Tmod register for the DMA transfers performed. Therefore, an appropriate value must be set in the IDE_Tmod register according to the IDE transfer mode before transfers start.

6.5.3.1 Multi-Word DMA Read

Multi-Word DMA read is conducted according to the following procedure:

Write 1 to the IDE_Config_0.DMA and 0 to the IDE_Config_0.Ultra bit.

Write appropriate values to the IDE_Config_1.DelayStrobe and IDE_Config_1.InterLock bits.

Set the number of transfer bytes in the IDE_CountH/M/L register.

Write 1 to the IDE_Control.IDE_Clr bit in order to initialize the IDE circuit. (This step may not always be required.)

Write 1 to the IDE_Control.IDE_Go bit in order to start a DMA operation. The content of the IDE_CountH/M/L register decrements as the transfer proceeds; when the count reaches 0 and the DMA operation of the IDE and writing of the data read from the IDE into the internal FIFO are complete, the IDE_Control.IDE_Go bit is cleared to 0 and the IDE_IntStat.IDE_Cmp bit is set to 1, causing the DMA operation to end.

If the IDE_Control.IDE_Go bit is cleared by writing 0 while a DMA transfer is underway, the DMA transfer is aborted and the DMA is terminated. In such a case, although the data that has been transferred to the internal FIFO by that time is valid, the data that has accumulated in the intermediate buffer of the circuit is discarded and the number of transferred bytes of the IDE therefore cannot be managed precisely.

6.5.3.2 Multi-Word DMA Write

Multi-Word DMA write is conducted according to the following procedure:

Write 1 to the IDE_Config_0.DMA and 0 to the IDE_Config_0.Ultra bit.

Write appropriate values to the IDE_Config_1.DelayStrobe and IDE_Config_1.InterLock bits.

Set the number of transfer bytes in the IDE_CountH/M/L register.

Write 1 to the IDE_Control.IDE_Clr bit in order to initialize the IDE circuit. (This step may not always be required.)

Write 1 to the IDE_Control.IDE_Go bit in order to start a DMA operation. The content of the IDE_CountH/M/L register decrements as the transfer proceeds; when the count reaches 0 and the DMA write to the IDE is complete, the IDE_Control.IDE_Go bit is cleared to 0 and the IDE_IntStat.IDE_Cmp bit is set to 1, causing the DMA operation to end.

If the IDE_Control.IDE_Go bit is cleared by writing 0 while a DMA transfer is underway, the DMA transfer is aborted and the DMA is terminated. In such a case, although the data that has been written to the IDE by that time is valid, the data that has accumulated in the intermediate buffer of the circuit is discarded and the number of transferred bytes of the IDE therefore cannot be managed precisely.

6.5.4 Ultra DMA

This section describes the DMA functions in Ultra DMA mode.

DMA in Ultra DMA mode uses the value set in the IDE_Umod register for the DMA transfers performed. Therefore, an appropriate value must be set in the IDE_Umod register according to the IDE transfer mode before transfers start.

6.5.4.1 Ultra DMA Read

Ultra DMA read is conducted according to the following procedure:

Write 1 to the IDE_Config_0.DMA and IDE_Config_0.Ultra bits.

Set the number of transfer bytes in the IDE_CountH/M/L register.

Write 1 to the IDE_Control.IDE_Clr bit in order to initialize the IDE circuit. (This step may not always be required.)

Write 1 to the IDE_Control.IDE_Go bit in order to start a DMA operation. The content of the IDE_CountH/M/L register decrements as the transfer proceeds; when the count reaches 0 and the DMA operation of the IDE and writing of the data read from the IDE into the internal FIFO are complete, the IDE_Control.IDE_Go bit is cleared to 0 and the IDE_IntStat.IDE_Cmp bit is set to 1, causing the DMA operation to end.

If the IDE_Control.IDE_Go bit is cleared by writing 0 while a DMA transfer is underway, the DMA transfer is aborted and the DMA is terminated. In such a case, although the data that has been transferred to the internal FIFO by that time is valid, the data that has accumulated in the intermediate buffer of the circuit is discarded and the number of transferred bytes of the IDE therefore cannot be managed precisely. In such a case, furthermore, the IDE device connected to the system is notified of a “host terminate” event.

If a “device terminate” event occurs during a DMA transfer, the IDE_IntStat.DetectTerm bit is set to 1; however, because the activated DMA is not terminated, despite the fact that the connected IDE device may have stopped the transfer, the DMA must be terminated by writing 0 to the IDE_Control.IDE_Go bit.

6.5.4.2 Ultra DMA Write

Ultra DMA write is conducted according to the following procedure:

Write 1 to the IDE_Config_0.DMA and IDE_Config_0.Ultra bits.

Set the number of transfer bytes in the IDE_CountH/M/L register.

Write 1 to the IDE_Control.IDE_Clr bit in order to initialize the IDE circuit. (This step may not always be required.)

Write 1 to the IDE_Control.IDE_Go bit in order to start a DMA operation. The content of the IDE_CountH/M/L register decrements as the transfer proceeds; when the count reaches 0 and the DMA write to the IDE is complete, the IDE_Control.IDE_Go bit is cleared to 0 and the IDE_IntStat.IDE_Cmp bit is set to 1, causing the DMA operation to end.

If the IDE_Control.IDE_Go bit is cleared by writing 0 while a DMA transfer is underway, the DMA transfer is aborted and the DMA is terminated. In such a case, although the data that has been written to the IDE by that time is valid, the data that has accumulated in the intermediate buffer of the circuit is discarded and the number of transferred bytes of the

IDE therefore cannot be managed precisely. In such a case, furthermore, the IDE device connected to the system is notified of a "host terminate" event.

If a "device terminate" event occurs during a DMA transfer, the IDE_IntStat.DetectTerm bit is set to 1; however, because the activated DMA is not terminated, despite the fact that the connected IDE device may have stopped the transfer, the DMA must be terminated by writing 0 to the IDE_Control.IDE_Go bit.

6.5.5 IDE Transfer Mode Settings

The tables below list the register values to be set for each mode of IDE transfer.

- Register Mode (For access to the IDE task file registers by firmware when not XHCS0 = 0 and HDA = 0)

Mode	Register set value (essential)		Register set value (recommended)
	IDE_Config_0.Ultra	IDE_Config_0.DMA	IDE_Rmod
0	Irrelevant	Irrelevant	FFh
1	Irrelevant	Irrelevant	F1h
2	Irrelevant	Irrelevant	F0h
3	Irrelevant	Irrelevant	22h
4	Irrelevant	Irrelevant	10h

- PIO Mode (For access to the IDE task file registers by firmware, when XHCS0 = 0 and HDA = 0)

Mode	Register set value (essential)		Register set value (recommended)
	IDE_Config_0.Ultra	IDE_Config_0.DMA	IDE_Tmod
0	Irrelevant	Irrelevant	FFh
1	Irrelevant	Irrelevant	88h
2	Irrelevant	Irrelevant	44h
3	Irrelevant	Irrelevant	22h
4	Irrelevant	Irrelevant	10h

- Multi-word DMA Mode (For DMA transfer)

Mode	Register set value (essential)		Register set value (recommended)
	IDE_Config_0.Ultra	IDE_Config_0.DMA	IDE_Tmod
0	0	0	FFh
1	0	0	88h
2	0	0	44h
3	0	0	22h
4	0	0	10h

- Multi-word DMA Mode (For DMA transfer)

Mode	Register set value (essential)		Register set value (recommended)
	IDE_Config_0.Ultra	IDE_Config_0.DMA	IDE_Tmod
0	0	1	BBh
1	0	1	20h
2	0	1	10h

- Ultra Mode (For DMA transfer, during DATA-OUT)

Mode	Register set value (essential)		Register set value (recommended)
	IDE_Config_0.Ultra	IDE_Config_0.DMA	IDE_Umod
0	1	1	06h
1	1	1	04h
2	1	1	03h
3	1	1	02h
4	1	1	01h
5	1	1	00h

Note: During DATA-IN in Ultra mode in which data is input from the IDE bus, data can be received in any mode regardless of how the IDE_Umode register is set.

6.6 Boundary Scan (JTAG)

Boundary Scan (JTAG) can be used when the TEST pin is held low (default). The boundary scan consists of the BSR (Boundary Scan Register) compliant with the JTAG (IEEE1149.1) specification, a scan path to connect it, and a TAP controller. Information on boundary-scan connections will be supplied to users in BSDL format.

6.6.1 Supported Instructions

The JTAG instructions supported by the LSI stipulated herein are 4 bits in width. These correspond to the JTAG instructions listed in the table below.

Table 6-18 JTAG Instruction Codes

Instruction	Description	Code
SAMPLE/PRELOAD	Latches the internal LSI status into BSR and sets data	0010
BYPASS	Bypasses the scan path configured by BSR	1111
EXTEST	Checks the physical device connection	0000
CLAMP	Bypasses the scan path while retaining the output value	0011
HIGHZ	Fixes all outputs to the Hi-Z state	0100
IDCODE	Outputs the designated DEVICE_CODE	0001

6.6.2 DEVICE_CODE

The elements that comprise DEVICE_CODE for the INCODE instruction are listed in the table below.

Table 6-19 DEVICE_CODE

Version	1
Part Number	0x000E
Manufacturer	0x0BE

Therefore, the DEVICE_CODE response for the INCODE instruction is as follows:

0001_0000000000001110_00010111110_1

6.6.3 Boundary Scan Exclusion Pins

Of the pins mounted on the LSI stipulated herein, DP, DM, XI, XO, VBUS, R1 and TEST are excluded from the scan, as they do not have boundary-scan cells inserted. Therefore, the DP, DM, and XO pins cannot be placed in the High-Z state by executing the HIGHZ instruction.

Chapter 7 Registers

7.1 Register Map

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x00	***MainIntStat***	R/(W)	0x00	***SIE_IntStat***	EPrIntStat	CPU_IntStat	FIFO_IntStat	BulkIntStat	IDE_IntStat	EP0IntStat	RcvEP0SETUP
0x01	EPrIntStat	R	0x00						EPcIntStat	EPbIntStat	EPaIntStat
0x02	***SIE_IntStat***	R/(W)	0x00	***VBUS_Changed***	***NonJ***	***FinishedPM***	DetectRESET	DetectSUSPEND	ChirpCmp	RestoreCmp	SetAddressCmp
0x03	CPU_IntStat	R/(W)	0x00		RcvSOF	DMA1_Countup	DMA1_Cmp			DMA0_Countup	DMA0_Cmp
0x04	FIFO_IntStat	R/(W)	0x00	DescriptorCmp	RAM_RdCmp	FIFO_IDE_Cmp	FIFO1_Cmp	FIFO_Full	FIFO_Empty		FIFO0_Cmp
0x05	BulkIntStat	R/(W)	0x00	CBW_Cmp	CBW_LengthErr	CBW_Err		CSW_Cmp	CSW_Err		
0x06	IDE_IntStat	R/(W)	0x00	IDE_RegCmp	IDE_RegErr	IDE_SeqWrRegCmp	CompleteINTRQ		IDE_Cmp	DetectINTRQ	DetectTerm
0x07	(DBG_IntStat)	R/(W)	0x00	CPU_CacheMiss	SIE_AutoNegoErr	FIFO_JoinErr		CPU_BufWrMiss			
0x08	EP0IntStat	R/(W)	0x00		OUT_ShortACK	IN_TranACK	OUT_TranACK	IN_TranNAK	OUT_TranNAK	IN_TranErr	OUT_TranErr
0x09	EPaIntStat	R/(W)	0x00		OUT_ShortACK	IN_TranACK	OUT_TranACK	IN_TranNAK	OUT_TranNAK	IN_TranErr	OUT_TranErr
0x0A	EPbIntStat	R/(W)	0x00		OUT_ShortACK	IN_TranACK	OUT_TranACK	IN_TranNAK	OUT_TranNAK	IN_TranErr	OUT_TranErr
0x0B	EPcIntStat	R/(W)	0x00		OUT_ShortACK	IN_TranACK	OUT_TranACK	IN_TranNAK	OUT_TranNAK	IN_TranErr	OUT_TranErr
0x0C			0xXX								
0x0D			0xXX								
0x0E			0xXX								
0x0F			0xXX								

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x10	***MainIntEnb***	R/W	0x00	***EnSIE_IntStat***	EnEPrIntStat	EnCPU_IntStat	EnFIFO_IntStat	EnBulkIntStat	EnIDE_IntStat	EnEP0IntStat	EnRcvEP0SETUP
0x11	EPrIntEnb	R/W	0x00						EnEPcIntStat	EnEPbIntStat	EnEPaIntStat
0x12	***SIE_IntEnb***	R/W	0x00	***EnVBUS_Changed***	***EnNonJ***	***EnFinishedPM***	EnDetectRESET	EnDetectSUSPEND	EnChirpCmp	EnRestoreCmp	EnSetAddressCmp
0x13	CPU_IntEnb	R/W	0x00		EnRcvSOF	EnDMA1_Countup	EnDMA1_Cmp			EnDMA0_Countup	EnDMA0_Cmp
0x14	FIFO_IntEnb	R/W	0x00	EnDescriptorCmp	EnRAM_RdCmp	EnFIFO_IDE_Cmp	EnFIFO1_Cmp	EnFIFO_Full	EnFIFO_Empty		EnFIFO0_Cmp
0x15	BulkIntEnb	R/W	0x00	EnCBW_Cmp	EnCBW_LengthErr	EnCBW_Err		EnCSW_Cmp	EnCSW_Err		
0x16	IDE_IntEnb	R/W	0x00	EnIDE_RegCmp	EnIDE_RegErr	En_SeqWrRegCmp	EnCompleteINTRQ		EnIDE_Cmp	EnDetectINTRQ	EnDetectTerm
0x17			0xXX								
0x18	EP0IntEnb	R/W	0x00		EnOUT_ShortACK	EnIN_TranACK	EnOUT_TranACK	EnIN_TranNAK	EnOUT_TranNAK	EnIN_TranErr	EnOUT_TranErr
0x19	EPaIntEnb	R/W	0x00		EnOUT_ShortACK	EnIN_TranACK	EnOUT_TranACK	EnIN_TranNAK	EnOUT_TranNAK	EnIN_TranErr	EnOUT_TranErr
0x1A	EPbIntEnb	R/W	0x00		EnOUT_ShortACK	EnIN_TranACK	EnOUT_TranACK	EnIN_TranNAK	EnOUT_TranNAK	EnIN_TranErr	EnOUT_TranErr
0x1B	EPcIntEnb	R/W	0x00		EnOUT_ShortACK	EnIN_TranACK	EnOUT_TranACK	EnIN_TranNAK	EnOUT_TranNAK	EnIN_TranErr	EnOUT_TranErr
0x1C			0xXX								
0x1D			0xXX								
0x1E			0xXX								
0x1F			0xXX								

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x20	***RevisionNum***	R	0x30	***Revision Number***							
0x21	***ChipReset***	W	0xXX	***ResetUTM***						-	***AllReset***
0x22	***PM_Control***	R/W	0x00		***PM_State [2:0]***			***GoSLEEP***	***GoSNOOZE***	***GoActive60***	***GoActiveALL***
0x23	USB_Control	R/W	0x00	DisBusDetect	EnAutoNego	InSUSPEND	DisableHS	SendWakeup	RestoreUSB	GoChirp	ActiveUSB
0x24	***USB_Status***	R	0xXX	***VBUS***	FSxHS					***LineState [1:0]***	
0x25	XcvrControl	R/W	0x41	TermSelect	XcvrSelect					OpMode [1:0]	
0x26	USB_Test	R/W	0x00	EnHS_Test				Test_SE0_NAK	Test_J	Test_K	Test_Packet
0x27			0xXX								
0x28	EPnControl	W	0xXX	AllForceNAK	EPrForceSTALL	AllFIFO_Clr					EP0FIFO_Clr
0x29	EPrFIFO_Clr	W	0xXX						EPcFIFO_Clr	EPbFIFO_Clr	EPaFIFO_Clr
0x2A	ClrAllJoin	W	0xXX	ClrJoinIDE	ClrJoin_FIFOStatus	ClrJoinDMA0_Rd	ClrJoinDMA0_Wr	ClrJoinDMA1_Rd	ClrJoinDMA1_Wr	ClrJoinCPU_Rd	ClrJoinCPU_Wr
0x2B			0xXX								
0x2C	BulkOnlyControl	R/W	0x00	AutoForceNAK_CBW					GoCBW_Mode	GoCSW_Mode	
0x2D	BulkOnlyConfig	R/W	0x00	(Reserved)	(Reserved)	(Reserved)	(Reserved)	(Reserved)	EPcBulkOnly	EPbBulkOnly	(Reserved)
0x2E	***WakeupTim_H***	R/W	0x00	***WakeupTim [15:8]***							
0x2F	***WakeupTim_L***	R/W	0x00	***WakeupTim [7:0]***							

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x30	EP0SETUP_0	R	0x00	SETUP 0							
0x31	EP0SETUP_1	R	0x00	SETUP 1							
0x32	EP0SETUP_2	R	0x00	SETUP 2							
0x33	EP0SETUP_3	R	0x00	SETUP 3							
0x34	EP0SETUP_4	R	0x00	SETUP 4							
0x35	EP0SETUP_5	R	0x00	SETUP 5							
0x36	EP0SETUP_6	R	0x00	SETUP 6							
0x37	EP0SETUP_7	R	0x00	SETUP 7							
0x38	USB_Address	R(/W)	0x00	SetAddress	USB_Address [6:0]						
0x39			0xXX								
0x3A	SETUP_Control	R/W	0x00								ProtectEP0
0x3B			0xXX								
0x3C			0xXX								
0x3D			0xXX								
0x3E	FrameNumber_H	R	0x80	FN_Invalid					FrameNumber [10:8]		
0x3F	FrameNumber_L	R	0x00	FrameNumber [7:0]							

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x40	EP0MaxSize	R/W	0x40		EP0MaxSize[6:3]						
0x41	EP0Control	R/W	0x00	INxOUT							ReplyDescriptor
0x42	EP0ControlIN	R/W	0x00		EnShortPkt		ToggleStat	ToggleSet	ToggleClr	ForceNAK	ForceSTALL
0x43	EP0ControlOUT	R/W	0x00	AutoForceNAK			ToggleStat	ToggleSet	ToggleClr	ForceNAK	ForceSTALL
0x44			0xXX								
0x45	EP0Join	R/W	0x00			JoinDMA0_Rd	JoinDMA0_Wr	JoinDMA1_Rd	JoinDMA1_Wr	JoinCPU_Rd	JoinCPU_Wr
0x46			0xXX								
0x47			0xXX								
0x48			0xXX								
0x49			0xXX								
0x4A			0xXX								
0x4B			0xXX								
0x4C			0xXX								
0x4D			0xXX								
0x4E			0xXX								
0x4F			0xXX								

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x50			0xXX								
0x51	EPaMaxSize_L	R/W	0x00		EPaMaxSize [6:3]						
0x52	EPaConfig_0	R/W	0x00	INxOUT	IntEP_Mode	EnEndpoint		EndpointNumber[3:0]			
0x53			0xXX								
0x54	EPaControl	R/W	0x00	AutoForceNAK	EnShortPkt	DisAF_NAK_Short	ToggleStat	ToggleSet	ToggleClr	ForceNAK	ForceSTALL
0x55	EPaJoin	R/W	0x00	JoinIDE	JoinFIFO_Stat	JoinDMA0_Rd	JoinDMA0_Wr	JoinDMA1_Rd	JoinDMA1_Wr	JoinCPU_Rd	JoinCPU_Wr
0x56			0xXX								
0x57			0xXX								
0x58	EPbMaxSize_H	R/W	0x00							EPbMaxSize[9:8]	
0x59	EPbMaxSize_L	R/W	0x00	EPbMaxSize[7:3]							
0x5A	EPbConfig_0	R/W	0x00	INxOUT	IntEP_Mode	EnEndpoint		EndpointNumber[3:0]			
0x5B			0xXX								
0x5C	EPbControl	R/W	0x00	AutoForceNAK	EnShortPkt	DisAF_NAK_Short	ToggleStat	ToggleSet	ToggleClr	ForceNAK	ForceSTALL
0x5D	EPbJoin	R/W	0x00	JoinIDE	JoinFIFO_Stat	JoinDMA0_Rd	JoinDMA0_Wr	JoinDMA1_Rd	JoinDMA1_Wr	JoinCPU_Rd	JoinCPU_Wr
0x5E			0xXX								
0x5F			0xXX								

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x60	EPcMaxSize_H	R/W	0x00							EPcMaxSize[9:8]	
0x61	EPcMaxSize_L	R/W	0x00	EPcMaxSize[7:3]							
0x62	EPcConfig_0	R/W	0x00	INxOUT	IntEP_Mode	EnEndpoint		EndpointNumber[3:0]			
0x63			0xXX								
0x64	EPcControl	R/W	0x00	AutoForceNAK	EnShortPkt	DisAF_NAK_Short	ToggleStat	ToggleSet	ToggleClr	ForceNAK	ForceSTALL
0x65	EPcJoin	R/W	0x00	JoinIDE	JoinFIFO_Stat	JoinDMA0_Rd	JoinDMA0_Wr	JoinDMA1_Rd	JoinDMA1_Wr	JoinCPU_Rd	JoinCPU_Wr
0x66			0xXX								
0x67			0xXX								
0x68			0xXX								
0x69			0xXX								
0x6A			0xXX								
0x6B			0xXX								
0x6C	RAM_WrAdrs_H	R/W	0x00					RAM_Adrs[11:8]			
0x6D	RAM_WrAdrs_L	R/W	0x00	RAM_Adrs[7:0]							
0x6E	RAM_WrDoor_H	W	0xXX	RAM_Door[15:8]							
0x6F	RAM_WrDoor_L	W	0xXX	RAM_Door[7:0]							

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x70	EPnFIFO_Rd_H	R	0xXX	EPnFIFO_Rd [15:8]							
0x71	EPnFIFO_Rd_L	R	0xXX	EPnFIFO_Rd [7:0]							
0x72	EPnFIFO_Wr_H	W	0xXX	EPnFIFO_Wr [15:8]							
0x73	EPnFIFO_Wr_L	W	0xXX	EPnFIFO_Wr [7:0]							
0x74	EPnRdRemain_H	R	0x00	RdRemainValid			EPnRdRemain [12:8]				
0x75	EPnRdRemain_L	R	0x00	EPnRdRemain [7:0]							
0x76	EPnWrRemain_H	R	0x00				EPnWrRemain [12:8]				
0x77	EPnWrRemain_L	R	0x00	EPnWrRemain [7:0]							
0x78	DescAdrs_H	R/W	0x00					DescAdrs[11:8]			
0x79	DescAdrs_L	R/W	0x00	DescAdrs [7:0]							
0x7A	DescSize_H	R/W	0x00							DescSize [9:8]	
0x7B	DescSize_L	R/W	0x00	DescSize [7:0]							
0x7C	EPnFIFO_ByteRd	R	0xXX	EPnFIFO_ByteRd [7:0]							
0x7D			0xXX								
0x7E			0xXX								
0x7F			0xXX								

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x80	DMA0_FIFO_Control	R/W	0x00	FIFO_Running	AutoEnShort						
0x81	DMA0_Config	R/W	0x00	FreeRun	DMA_Mode			ActiveDMA		ReqAssertCount [1:0]	
0x82	DMA0_Control	R/W	0x00	DMA_Running			CounterClr			DMA_Stop	DMA_Go
0x83			0xXX								
0x84	DMA0_Remain_H	R	0x00				DMA_Remain [12:8]				
0x85	DMA0_Remain_L	R	0x00	DMA_Remain [7:0]							
0x86			0xXX								
0x87			0xXX								
0x88	DMA0_Count_HH	R/W	0x00	DMA_Count [31:24]							
0x89	DMA0_Count_HL	R/W	0x00	DMA_Count [23:16]							
0x8A	DMA0_Count_LH	R/W	0x00	DMA_Count [15:8]							
0x8B	DMA0_Count_LL	R/W	0x00	DMA_Count [7:0]							
0x8C	DMA0_RdData_H	R	0xXX	DMA_RdData[15:8]							
0x8D	DMA0_RdData_L	R	0xXX	DMA_RdData[7:0]							
0x8E	DMA0_WrData_H	W	0xXX	DMA_WrData[15:8]							
0x8F	DMA0_WrData_L	W	0xXX	DMA_WrData[7:0]							

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x90	DMA1_FIFO_Control	R/W	0x00	FIFO_Running	AutoEnShort						
0x91	DMA1_Config	R/W	0x00	FreeRun	DMA_Mode			ActiveDMA		ReqAssertCount [1:0]	
0x92	DMA1_Control	R/W	0x00	DMA_Running			CounterClr			DMA_Stop	DMA_Go
0x93			0xXX								
0x94	DMA1_Remain_H	R	0x00				DMA_Remain [12:8]				
0x95	DMA1_Remain_L	R	0x00	DMA_Remain [7:0]							
0x96			0xXX								
0x97			0xXX								
0x98	DMA1_Count_HH	R/W	0x00	DMA_Count [31:24]							
0x99	DMA1_Count_HL	R/W	0x00	DMA_Count [23:16]							
0x9A	DMA1_Count_LH	R/W	0x00	DMA_Count [15:8]							
0x9B	DMA1_Count_LL	R/W	0x00	DMA_Count [7:0]							
0x9C	DMA1_RdData_H	R	0xXX	DMA_RdData[15:0]							
0x9D	DMA1_RdData_L	R	0xXX	DMA_RdData[7:0]							
0x9E	DMA1_WrData_H	W	0xXX	DMA_WrData[15:8]							
0x9F	DMA1_WrData_L	W	0xXX	DMA_WrData[7:0]							

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xA0	IDE_Status	R/W	0x00	DMARQ	DMACK	INTRQ	IORDY			PDIAG	DASP
0xA1	IDE_Control	R/W	0x00		IDE_Clr						IDE_Go
0xA2	IDE_Comfig_0	R/W	0x00	IDE_BusReset	IDE_LongBusReset					Ultra	DMA
0xA3	IDE_Config_1	R/W	0x00	ActiveIDE	DelayStrobe		InterLock		Swap		
0xA4	IDE_Rmod	R/W	0x00	RegisterAssertPulseWidth[3:0]				RegisterNegatePulseWidth[3:0]			
0xA5	IDE_Tmod	R/W	0x00	TransferAssertPulseWidth[3:0]				TransferNegatePulseWidth[3:0]			
0xA6	IDE_Umod	R/W	0x00					UltraDMA_Cycle[3:0]			
0xA7			0xXX								
0xA8			0xXX								
0xA9			0xXX								
0xAA	IDE_CRC_H	R/W	0x00	IDE_CRC[15:8]							
0xAB	IDE_CRC_L	R/W	0x00	IDE_CRC[7:0]							
0xAC			0xXX								
0xAD	IDE_Count_H	R/W	0x00	IDE_Count[23:16]							
0xAE	IDE_Count_M	R/W	0x00	IDE_Count[15:8]							
0xAF	IDE_Count_L	R/W	0x00	IDE_Count[7:1]							

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xB0	IDE_RegAdrs	R/W	0x00	IDE_WrReg	IDE_RdReg			IDE_RegAddress[3:0]			
0xB1			0xXX								
0xB2	IDE_RdRegValue_H	R	0x00	IDE_RdRegValue[15:8]							
0xB3	IDE_RdRegValue_L	R	0x00	IDE_RdRegValue[7:0]							
0xB4	IDE_WrRegValue_H	R/W	0x00	IDE_WrRegValue[15:8]							
0xB5	IDE_WrRegValue_L	R/W	0x00	IDE_WrRegValue[7:0]							
0xB6	IDE_SeqWrRegControl	R/W	0x00	IDE_SeqWrReg	IDE_SeqWrRegClr						
0xB7	IDE_SeqWrRegCnt	R	0x00				IDE_SeqWrRegCnt[4:0]				
0xB8	IDE_SeqWrRegAdrs	W	0xXX					IDE_SeqRegAddress[3:0]			
0xB9	IDE_SeqWrRegValue	W	0xXX	IDE_SeqWrRegValue[7:0]							
0xBA			0xXX								
0xBB			0xXX								
0xBC	IDE_RegConfig	R/W	0x00	EnAutoStatusRd							
0xBD			0xXX								
0xBE			0xXX								
0xBF			0xXX								

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xC0	RAM_Rd_00	R	0x00								
0xC1	RAM_Rd_01	R	0x00								
0xC2	RAM_Rd_02	R	0x00								
0xC3	RAM_Rd_03	R	0x00								
0xC4	RAM_Rd_04	R	0x00								
0xC5	RAM_Rd_05	R	0x00								
0xC6	RAM_Rd_06	R	0x00								
0xC7	RAM_Rd_07	R	0x00								
0xC8	RAM_Rd_08	R	0x00								
0xC9	RAM_Rd_09	R	0x00								
0xCA	RAM_Rd_0A	R	0x00								
0xCB	RAM_Rd_0B	R	0x00								
0xCC	RAM_Rd_0C	R	0x00								
0xCD	RAM_Rd_0D	R	0x00								
0xCE	RAM_Rd_0E	R	0x00								
0xCF	RAM_Rd_0F	R	0x00								

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xD0	RAM_Rd_10	R	0x00								
0xD1	RAM_Rd_11	R	0x00								
0xD2	RAM_Rd_12	R	0x00								
0xD3	RAM_Rd_13	R	0x00								
0xD4	RAM_Rd_14	R	0x00								
0xD5	RAM_Rd_15	R	0x00								
0xD6	RAM_Rd_16	R	0x00								
0xD7	RAM_Rd_17	R	0x00								
0xD8	RAM_Rd_18	R	0x00								
0xD9	RAM_Rd_19	R	0x00								
0xDA	RAM_Rd_1A	R	0x00								
0xDB	RAM_Rd_1B	R	0x00								
0xDC	RAM_Rd_1C	R	0x00								
0xDD	RAM_Rd_1D	R	0x00								
0xDE	RAM_Rd_1E	R	0x00								
0xDF	RAM_Rd_1F	R	0x00								

The registers that can be both read and written even during Sleep/Snooze are shown in ***italic bold*** face.

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xE0	RAM_RdAdrs_H	R/W	0x00					RAM_RdAdrs[11:8]			
0xE1	RAM_RdAdrs_L	R/W	0x00	RAM_RdAdrs[7:2]							
0xE2	RAM_RdControl	R/W	0x00	RAM_GoRdCBW	RAM_GoRd						
0xE3	RAM_RdCount	R/W	0x00			RAM_RdCount[5:2]					
0xE4			0xXX								
0xE5			0xXX								
0xE6			0xXX								
0xE7			0xXX								
0xE8			0xXX								
0xE9			0xXX								
0xEA			0xXX								
0xEB	***ModeProtect***	R/W	0x56	***Protected[7:0] (writing other than 56 enables protect; 0x56 disables protect)***							
0xEC			0xXX								
0xED	***ClkSelect***	R/W	0x01	***xActIDE_Term***							***ClkSelect***
0xEE			0xXX								
0xEF	***ChipConfig***	R/W	0x00	***IntLevel***	***IntMode***	***DREQ_Level***	***DACK_Level***	***CS_Mode***	***CPU_Swap***	***BusMode***	***Bus8x16***

Byte Addr.	Register Name	R/W	Reset	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
				bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0xF0	(Reserved)	R/W	0xXX								
0xF1	D_ModeControl	W	0xXX	(Reserved)	(Reserved)	(Reserved)	SetAddressMode	(Reserved)	(Reserved)	(Reserved)	(Reserved)
0xF2	(Reserved)	R/W	0xXX								
0xF3	(Reserved)	R/W	0xXX								
0xF4	(Reserved)	R/W	0xXX								
0xF5	(Reserved)	R/W	0xXX								
0xF6	(Reserved)	R/W	0xXX								
0xF7	(Reserved)	R/W	0xXX								
0xF8	(Reserved)	R/W	0xXX								
0xF9	(Reserved)	R/W	0xXX								
0xFA	(Reserved)	R/W	0xXX								
0xFB	(Reserved)	R/W	0xXX								
0xFE	(Reserved)	R/W	0xXX								
0xFD	(Reserved)	R/W	0xXX								
0xFE	(Reserved)	R/W	0xXX								
0xFF	(Reserved)	R/W	0xXX								

Refer to Appendix C for detail description for 0xF1 D_ModeControl.

7.2 Register Details

7.2.1 00h *MainIntStat (Main Interrupt Status)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
00h	***MainIntStat***	R	7: ***SIE_IntStat***	0: None	1: SIE Interrupts	00h
		R	6: EPrIntStat	0: None	1: EPr Interrupts	
		R	5: CPU_IntStat	0: None	1: CPU Interrupts	
		R	4: FIFO_IntStat	0: None	1: FIFO Interrupts	
		R	3: BulkIntStat	0: None	1: Bulk Interrupts	
		R	2: IDE_IntStat	0: None	1: IDE Interrupts	
		R	1: EP0IntStat	0: None	1: EP0 Interrupts	
		R (W)	0: RcvEP0SETUP	0: None	1: Receive EP0 SETUP	

This register shows the causes of interrupts generated in the LSI stipulated herein. Some bits in this register indicate the causes of interrupts indirectly, i.e., they indicate the 'source' from which an interrupt was generated. Other bits indicate the causes of interrupts directly, i.e., they indicate the 'reason' for which an interrupt was generated. If the cause of an interrupt is indicated indirectly by a bit in this register, read the interrupt status register corresponding to that bit. In this way, it is possible to trace back to the bit that indicates the reason the interrupt occurred. The bits indicating the causes of interrupts indirectly are read-only, and are automatically cleared by clearing the bit in another register that indicates the cause of the interrupt directly. On the other hand, the bits indicating the causes of interrupts directly are writable, so the cause of an interrupt indicated by a bit can be cleared by writing 1 to that bit. If the interrupts corresponding to each bit in an interrupt status register have been enabled by the MainIntEnb register and any bit in that register is set to 1, the XINT pin is asserted to generate an interrupt to the CPU. When all causes of interrupts in the status register are cleared, the XINT pin is negated.

Bit7 ***SIE_IntStat***

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the SIE_IntStat register and the bit corresponding to that interrupt cause in the SIE_IntEnb register has been enabled, this bit is set to 1. This bit is effective even during Sleep/Snooze.

Bit6 **EPrIntStat**

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the EPrIntStat register and the bit corresponding to that interrupt cause in the EPr_IntEnb register has been enabled, this bit is set to 1.

Bit5 **CPU_IntStat**

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the CPU_IntStat register and the bit corresponding to that interrupt cause in the CPU_IntEnb register has been enabled, this bit is set to 1.

Bit4 FIFO_IntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the FIFO_IntStat register and the bit corresponding to that interrupt cause in the FIFO_IntEnb register has been enabled, this bit is set to 1.

Bit3 BulkIntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the BulkIntStat register and the bit corresponding to that interrupt cause in the BulkIntEnb register has been enabled, this bit is set to 1.

Bit2 IDE_IntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the IDE_IntStat register and the bit corresponding to that interrupt cause in the IDE_IntEnb register has been enabled, this bit is set to 1.

Bit1 EP0IntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the EP0_IntStat register and the bit corresponding to that interrupt cause in the EP0IntEnb register has been enabled, this bit is set to 1.

Bit0 RcvEP0SETUP

This bit indicates the cause of an interrupt directly.

When the setup stage for a control transfer has finished and the received data is stored in the EP0Setup_0 through EP0Setup_7 registers, this bit is set to 1. At the same time, the ForceSTALL bit in the EP0ControlIN, EP0ControlOUT registers is cleared to 0, and the ForceNAK and ToggleStat bits in the EP0ControlIN, EP0ControlOUT registers and the ProtectEP0 bit in the SETUP Control register are set to 1, all automatically. SetAddress() requests are automatically responded to by the AutoSetAddress function, and this status is not set.

7.2.2 01h EPrIntStat (EPr interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
01h	EPrIntStat		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
		R	2: EPcIntStat	0: None	1: EPc Interrupt	
		R	1: EPbIntStat	0: None	1: EPb Interrupt	
		R	0: EPaIntStat	0: None	1: EPa Interrupt	

This register shows the interrupts for the endpoints EPr.

Bit7-3 Reserved

Bit2 EPcIntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the EPcIntStat register and the bit corresponding to that interrupt cause in the EPcIntEnb register has been enabled, this bit is set to 1.

Bit1 EPbIntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the EPbIntStat register and the bit corresponding to that interrupt cause in the EPbIntEnb register has been enabled, this bit is set to 1.

Bit0 EPaIntStat

This bit indicates the cause of an interrupt indirectly.

If the cause of the interrupt is in the EPaIntStat register and the bit corresponding to that interrupt cause in the EPaIntEnb register has been enabled, this bit is set to 1.

7.2.3 02h *SIE_IntStat (SIE Interrupt Status)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
02h	***SIE_IntStat***	R (W)	7: ***VBUS_Changed***	0: None	1: VBUS is Changed	00h
		R (W)	6: ***NonJ***	0: None	1: Detect Non J state	
		R (W)	5: ***FinishedPM***	0: None	1: Detect FinishedPM	
		R (W)	4: DetectReset	0: None	1: Detect USB Reset	
		R (W)	3: DetectSuspend	0: None	1: Detect USB Suspend	
		R (W)	2: ChirpCmp	0: None	1: Chirp Complete	
		R (W)	1: RestoreCmp	0: None	1: Restore Complete	
		R (W)	0: SetAddressCmp	0: None	1: AutoSetAddress Complete	

This register shows the interrupts associated with the SIE. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 ***VBUS_Changed***

This bit indicates the cause of an interrupt directly.

It is set to 1 when the status of the VBUS bit has changed.

Check the VBUS in the USB_Status register to confirm the VBUS status. If VBUS = 0, the cable has been removed. This bit is effective even during Sleep/Snooze.

Bit6 ***NonJ***

This bit indicates the cause of an interrupt directly.

It is set to 1 when a state other than J has been detected on the USB bus. This bit is effective when the LSI is in Snooze mode (PM_Control register's InSnooze bit = 1) and the InSUSPEND bit in the USB_Control register remains set to 1 while the AutoNegotiation function is in use.

Bit5 ***FinishedPM***

This bit indicates the cause of an interrupt directly.

It is set to 1 when the designated state is entered while GoSLEEP, GoSNOOZE, GoActive60, or GoActiveALL in the PM_Control register has been set.

Bit4 **DetectReset**

This bit indicates the cause of an interrupt directly.

It is set to 1 when a reset state of the USB has been detected. While this bit remains set, the suspend state of the USB cannot be detected (DetectSUSPEND is not set).

This reset detection is effective when the ActiveUSB bit in the USB_Control register remains set.

In HS operation mode, when no bus activity is detected for a predetermined length of time, FS termination is automatically set in order to detect a reset/suspend of the USB; when SE0 is detected, a reset is assumed and this bit is set to 1.

When the AutoNegotiation function is not in use, it is necessary to prevent the erroneous detection of subsequent resets after this bit is set to 1. Therefore, set the DisBusDetect bit in the USB_Control register to disable the detection of

reset/suspend states of the USB. Upon completion of processing for the reset, clear the DisBusDetect bit to 0 in order to reenable the detection of reset/suspend states of the USB.

When Reset is detected, HS Detection Handshaking can be started by the GoChirp bit in the USB_Control register.

For details about the AutoNe g otiation function, refer to the section in which the EnAutoNego bit in the USB_Control register is described.

Bit3 DetectSuspend

This bit indicates the cause of an interrupt directly.

It is set to 1 when a suspend state of the USB has been detected. While this bit remains set, the reset state of the USB cannot be detected (DetectRESET is not set).

In HS operation mode, when no bus activity is detected for a predetermined length of time, FS operation mode is automatically entered in order to detect a reset/suspend of the USB. After a suspend state of the USB is detected, the LSI can be placed in Snooze mode (internal PLLs turned off) by setting the Snooze bit in the PM_Control register to 1.

Bit2 ChirpCmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when HS Detection Handshaking started by the GoChirp bit in the USB_Control register has finished.

After the interrupt is generated, it is possible to determine the current operation mode (FS or HS) by reading the FSxHS bit in the USB_Status register.

Bit1 RestoreCmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when the Restore processing that was started by RestoreUSB bit in the USB_Control register has finished. When this bit is set to 1, operation mode (FS or HS) returns to the state in which the USB was prior to Suspend.

Bit0 SetAddressCmp

This bit indicates the cause of an interrupt directly.

When a SetAddress() request is received, the AutoSetAddress function (see the USB_Address register) automatically performs processing for the required control transfer. Then, when the control transfer for the SetAddress() request is completed by executing a status stage, this status bit is set to 1. At the same time, the address is set in the USB_Address register.

The synchronous bits (bits 4–0) cannot be written to (in order to eliminate the cause of the interrupt) when in the Active60 state, although they can be read normally.

When entering the Active60 state, the firmware should execute the processing described below in order to ensure that the interrupt signal XINT will not be inadvertently asserted by one of these interrupt statuses.

<When entering the Active60 state>

1) Process the interrupt status to clear it (SIE_IntStat.Bits4–0).

2) Disable the interrupt status (SIE_IntEnb.Bits4–0).

<When exiting the Active60 state>

3) Clear the interrupt status (SIE_IntStat.Bits4–0).

4) Reenable the interrupt status (SIE_IntEnb.Bits4–0).

7.2.4 03h CPU_IntStat (CPU Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
03h	CPU_IntStat		7:	0:	1:	00h
		R (W)	6: RcvSOF	0: None	1: Receive SOF Token	
		R (W)	5: DMA1_Countup	0: None	1: DMA1 Counter Overflow	
		R (W)	4: DMA1_Cmp	0: None	1: DMA1 Complete	
			3:	0:	1:	
			2:	0:	1:	
		R (W)	1: DMA0_CountUp	0: None	1: DMA0 Counter Overflow	
		R (W)	0: DMA0_Cmp	0: None	1: DMA0 Complete	

This register shows the interrupts associated with the CPU interface. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 Reserved

Bit6 RcvSOF

This bit indicates the cause of an interrupt directly.

It is set to 1 when an SOF token has been received.

Bit5 DMA1_CountUp

This bit indicates the cause of an interrupt directly.

It is set to 1 when the value of the DMA1_Count_HH,HL,LH,LL has overflowed while the LSI is operating in free-running mode of transfer. The value of the DMA1_Count_HH,HL,LH,LL recycles to 0, and the DMA operation continues.

Bit4 DMA1_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when the DMA transfer has been stopped or when termination processing has finished after completion of a specified number of transfers.

Bit3-2 Reserved

Bit1 DMA0_CountUp

This bit indicates the cause of an interrupt directly.

It is set to 1 when the value of the DMA0_Count_HH,HL,LH,LL has overflowed while the LSI is operating in free-running mode of transfer. The value of the DMA0_Count_HH,HL,LH,LL recycles to 0, with the DMA operation continued.

Bit0 DMA0_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when DMA transfer has been stopped or when termination processing has finished after the specified bytes have been transferred.

7.2.5 04h FIFO_InStat (FIFO Instat Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
04h	FIFOIntStat	R (W)	7: DescriptorCmp	0: None	1: Descriptor Complete	00h
		R (W)	6: RAM_RdCmp	0: None	1: RAM Read Complete	
		R (W)	5: FIFO_IDE_Cmp	0: None	1: FIFO-IDE Complete	
		R (W)	4: FIFO1_Cmp	0: None	1: FIFO1 Complete	
		R (W)	3: FIFO_Full	0: None	1: Selected FIFO is Full	
		R (W)	2: FIFO_Empty	0: None	1: Selected FIFO is Empty	
			1:	0:	1:	
		R (W)	0: FIFO0_Cmp	0: None	1: FIFO0 Complete	

This register shows the interrupts associated with the FIFO. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 DescriptorCmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in the Descriptor reply function, the LSI has finished sending the data bytes set in the DescriptorSize register back to the host.

If a transition to the status stage occurs (OUT token received) before all of the data bytes set in the DescriptorSize register have been sent back, the OUT_TranNAK bit in the EP0IntStat register is set to 1 along with this status bit.

Bit6 RAM_RdCmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in the RAM_Rd function, the data read out from the RAM has been readied in the RAM_Rd_XX.

Bit5 FIFO_IDE_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when, while the endpoint joined to the IDE is directed IN, the FIFO has been emptied upon completion of IDE transfer. If the endpoint joined to the IDE is directed OUT, this bit is set to 1 upon completion of the IDE transfer.

Bit4 FIFO1_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when, while the endpoint joined to DMA1 is directed IN, the FIFO has been emptied upon completion of DMA1 transfer. If the endpoint joined to DMA1 is directed OUT, this bit is set to 1 upon completion of the DMA1 transfer.

Bit3 FIFO_Full

This bit indicates the cause of an interrupt directly.

It is set to 1 when the FIFO for the endpoint selected by EPx{x=a,b,c}Join.JoinFIFO_Stat is full.

Bit2 FIFO_Empty

This bit indicates the cause of an interrupt directly.

It is set to 1 when the FIFO for the endpoint selected by EPx{x=a,b,c}Join.JoinFIFO_Stat is empty.

Bit1 Reserved

Bit0 FIFO0_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when, while the endpoint joined to DMA0 is directed IN, the FIFO has been emptied upon completion of DMA0 transfer. If the endpoint joined to DMA0 is directed OUT, this bit is set to 1 upon completion of the DMA0 transfer.

7.2.6 05h BulkIntStat (Bulk Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
05h	BulkIntStat	R (W)	7: CBW_Cmp	0: None	1: CBW Complete	00h
		R (W)	6: CBW_LengthErr	0: None	1: CBW Length Error	
		R (W)	5: CBW_Err	0: None	1: CBW Transaction Error	
		R (W)	4:	0:	1:	
		R (W)	3: CSW_Cmp	0: None	1: CSW Complete	
		R (W)	2: CSW_Err	0: None	1: CSW Error	
			1:	0:	1:	
			0:	0:	1:	

This register shows the interrupts associated with the Bulk transfer function. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 CBW_Comp

This bit indicates the cause of an interrupt directly.

It is set to 1 when 31 bytes of CBW have been received normally.

Bit6 CBW_LengthErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when the received CBW packet is not 31 bytes in length.

Bit5 CBW_Err

This bit indicates the cause of an interrupt directly.

It is set to 1 when a CRC error or other transaction error has been detected in the received CBW.

Bit4 Reserved

Bit3 CSW_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when 13 bytes of CSW have been received normally.

Bit2 CSW_Err

This bit indicates the cause of an interrupt directly.

It is set to 1 when a CSW transmit error (no ACK response) has occurred.

Bit1-0 Reserved

7.2.7 06h IDE_IntStat (IDE Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
06h	IDE_IntStat	R (W)	7: IDE_RegCmp	0: None	1: Register Access Cmplete	00h
		R (W)	6: IDE_RegErr	0: None	1: Register Access Error	
		R (W)	5: IDE_SeqWrRegCmp	0: None	1: Sequence Write Complete	
		R (W)	4: CompleteINTRQ	0: None	1: Auto Status Read Compl.	
			3:	0:	1:	
		R (W)	2: IDE_Cmp	0: None	1: DMA Complete	
		R (W)	1: DetectINTRQ	0: None	1: Detected Interrupt	
		R (W)	0: DetectTerm	0: None	1: Detected Device terminate	

This register shows the interrupts associated with the IDE. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 IDE_RegCmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when a read/write access to the IDE register via the IDE_RegAdr register has been completed.

Bit6 IDE_RegErr

This bit indicates the cause of an interrupt directly.

It is set to 1 in the following cases:

1) A read/write access to the IDE register via the IDE_RegAdr register has been attempted during the operation sequence of an auto status read via the IDE_RegConfig register.
2) A read/write access to the IDE register via the IDE_RegAdr register has been attempted during the operation sequence of a sequential write via the IDE_SeqWrRegControl register.

Bit5 IDE_SeqWrRegCmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when a sequential write operation via the IDE_SeqWrRegControl register has been completed.

Bit4 CompleteINTRQ

This bit indicates the cause of an interrupt directly.

It is set to 1 when an auto status read operation via the IDE_RegConfig register has been completed.

Bit3 Reseved

Bit2 IDE_Cmp

This bit indicates the cause of an interrupt directly.

It is set to 1 when a DMA operation via the IDE_Control register has been completed.

Bit1 DetectINTRQ

This bit indicates the cause of an interrupt directly.

It is set to 1 when, while the EnAutoStsRd bit in the IDE_RegConfig register has not been set, a rising edge of the HINTRQ signal of the IDE has been detected.

Bit0 DetectTerm

This bit indicates the cause of an interrupt directly.

It is set to 1 when, while DMA operation of the IDE via the IDE_Control register is underway in transfer mode of Ultra-DMA, a “device terminate” event has been detected.

7.2.8 07h DBG_IntStat (DEBUG Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
07h	DBG_IntStat	R	7: CPU_CacheMiss	0: None	1: Detected CPU Cashe Miss	00h
		R (W)	6: SIE_AutoNegoErr	0: None	1: Detected AutoNego Error	
		R (W)	5: FIFO_JoinErr	0: None	1: Detected FIFO Join Error	
			4:	0:	1:	
		R	3: CPU_BufWrMiss	0: None	1: Detected CPU BufWr Error	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

This register shows the interrupt statuses of debugging registers. If the cause of an interrupt is indicated directly by a bit in this register, the cause of the interrupt can be cleared by writing 1 to that bit.

Bit7 CPU_CacheMiss

This bit indicates the cause of an interrupt indirectly.

It is set to 1 when any of the CPUIF blocks--REG, DMA0, or DMA1--has been accessed for reading while a caching operation in that block has not been completed.

This bit is used for software debugging.

Bit6 SIE_AutoNegoErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when a sequence error has occurred in the AutoNegotiation function.

This bit is used for hardware debugging.

Bit5 FIFO_JoinErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when an exclusion processing violation has been committed in the setting of endpoint access rights using the EPx{x=0,a-h}Join register.

This bit is used for software debugging.

Bit4 Reserved

Bit3 CPU_BufWrMiss

This bit indicates the cause of an interrupt indirectly.

It is set to 1 when any of the CPUIF blocks--REG, DMA0, or DMA1--has been written to before the operation for writing to the FIFO has been completed.

This bit is used for hardware debugging.

Bit2-0 Reserved

7.2.9 08h EP0IntStat (EP0 Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
08h	EP0IntStat		7:	0:	1:	00h
		R (W)	6: OUT_ShortACK	0: None	1: OUT Short-Packet ACK	
		R (W)	5: IN_TranACK	0: None	1: IN Transaction ACK	
		R (W)	4: OUT_TranACK	0: None	1: OUT Transaction ACK	
		R (W)	3: IN_TranNAK	0: None	1: IN Transaction NAK	
		R (W)	2: OUT_TranNAK	0: None	1: OUT Transaction NAK	
		R (W)	1: IN_TranErr	0: None	1: IN Transaction Error	
		R (W)	0: OUT_TranErr	0: None	1: OUT Transaction Error	

This register shows the interrupt statuses of the endpoint EP0. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 Reserved

Bit6 OUT_ShortACK

This bit indicates the cause of an interrupt directly.

It is set to 1 at the time OUT_TranACK is set when ACK has been returned for the short packet received in an OUT transaction.

Bit5 IN_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been received in an IN transaction.

Bit4 OUT_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been returned to the host in an OUT transaction.

Bit3 IN_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host in an IN transaction.

Bit2 OUT_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host for the OUT or PING transaction attempted.

Bit1 IN_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an IN transaction, STALL has been returned, an error has been found in the packet, or handshaking has timed out.

Bit0 OUT_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an OUT transaction, STALL has been returned or an error has been found in the packet.

7.2.10 09h EPaIntStat (EPa Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
09h	EPaIntStat		7:	0:	1:	00h
		R (W)	6: OUT_ShortACK	0: None	1: OUT Short Packet ACK	
		R (W)	5: IN_TranACK	0: None	1: IN Transaction ACK	
		R (W)	4: OUT_TranACK	0: None	1: OUT Transaction ACK	
		R (W)	3: IN_TranNAK	0: None	1: IN Transaction NAK	
		R (W)	2: OUT_TranNAK	0: None	1: OUT Transaction NAK	
		R (W)	1: IN_TranErr	0: None	1: IN Transaction Error	
		R (W)	0: OUT_TranErr	0: None	1: OUT Transaction Error	

This register shows the interrupt statuses of the endpoint EPa. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 Reserved

Bit6 OUT_ShortACK

This bit indicates the cause of an interrupt directly.

It is set to 1 at the time OUT_TranACK is set when ACK has been returned for the short packet received in an OUT transaction.

Bit5 IN_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been received in an IN transaction.

Bit4 OUT_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been returned to the host in an OUT transaction.

Bit3 IN_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host in an IN transaction.

Bit2 OUT_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host for the OUT or PING transaction attempted.

Bit1 IN_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an IN transaction, STALL has been returned, an error has been found in the packet, or handshaking has timed out.

Bit0 OUT_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an OUT transaction, STALL has been returned or an error has been found in the packet.

7.2.11 0Ah EPbIntStat (EPb Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
0Ah	EPbIntStat		7:	0:	1:	00h
		R (W)	6: OUT_ShortACK	0: None	1: OUT Short Packet ACK	
		R (W)	5: IN_TranACK	0: None	1: IN Transaction ACK	
		R (W)	4: OUT_TranACK	0: None	1: OUT Transaction ACK	
		R (W)	3: IN_TranNAK	0: None	1: IN Transaction NAK	
		R (W)	2: OUT_TranNAK	0: None	1: OUT Transaction NAK	
		R (W)	1: IN_TranErr	0: None	1: IN Transaction Error	
		R (W)	0: OUT_TranErr	0: None	1: OUT Transaction Error	

This register shows the interrupt statuses of the endpoint EPb. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 Reserved

Bit6 OUT_ShortACK

This bit indicates the cause of an interrupt directly.

It is set to 1 at the time OUT_TranACK is set when ACK has been returned for the short packet received in an OUT transaction.

Bit5 IN_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been received in an IN transaction.

Bit4 OUT_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been returned to the host in an OUT transaction.

Bit3 IN_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host in an IN transaction.

Bit2 OUT_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host for the OUT or PING transaction attempted.

Bit1 IN_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an IN transaction, STALL has been returned, an error has been found in the packet, or handshaking has timed out.

Bit0 OUT_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an OUT transaction, STALL has been returned or an error has been found in the packet.

7.2.12 0Bh EPcIntStat (EPc Interrupt Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
0Bh	EPcIntStat		7:	0:	1:	00h
		R (W)	6: OUT_ShortACK	0: None	1: OUT Short Packet ACK	
		R (W)	5: IN_TranACK	0: None	1: IN Transaction ACK	
		R (W)	4: OUT_TranACK	0: None	1: OUT Transaction ACK	
		R (W)	3: IN_TranNAK	0: None	1: IN Transaction NAK	
		R (W)	2: OUT_TranNAK	0: None	1: OUT Transaction NAK	
		R (W)	1: IN_TranErr	0: None	1: IN Transaction Error	
		R (W)	0: OUT_TranErr	0: None	1: OUT Transaction Error	

This register shows the interrupt statuses of the endpoint EPc. The cause of an interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

Bit7 Reserved

Bit6 OUT_ShortACK

This bit indicates the cause of an interrupt directly.

It is set to 1 at the time OUT_TranACK is set when ACK has been returned for the short packet received in an OUT transaction.

Bit5 IN_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been received in an IN transaction.

Bit4 OUT_TranACK

This bit indicates the cause of an interrupt directly.

It is set to 1 when ACK has been returned to the host in an OUT transaction.

Bit3 IN_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host in an IN transaction.

Bit2 OUT_TranNAK

This bit indicates the cause of an interrupt directly.

It is set to 1 when NAK has been returned to the host for the OUT or PING transaction attempted.

Bit1 IN_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an IN transaction, STALL has been returned, an error has been found in the packet, or handshaking has timed out.

Bit0 OUT_TranErr

This bit indicates the cause of an interrupt directly.

It is set to 1 when, in an OUT transaction, STALL has been returned or an error has been found in the packet.

7.2.13 0Ch - 0Fh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
0Ch -0Fh	Reserved		7:	0:	1:	XXh
			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 *MainIntEnb (Main Interrupt Enable)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
10h	***MainIntEnb***	R / W	7: ***EnSIE_IntStat***	0: Disable	1: Enable	00h
		R / W	6: EnEPrIntStat	0: Disable	1: Enable	
		R / W	5: EnCPU_IntStat	0: Disable	1: Enable	
		R / W	4: EnFIFO_IntStat	0: Disable	1: Enable	
		R / W	3: EnBulkIntStat	0: Disable	1: Enable	
		R / W	2: EnIDE_IntStat	0: Disable	1: Enable	
		R / W	1: EnEP0IntStat	0: Disable	1: Enable	
		R / W	0: EnRcvEP0SETUP	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the interrupt signal (XINT) for the interrupt causes contained in the MainIntStat register.

These interrupt sources can be enabled for interrupt generation by setting the corresponding bits in this register to 1.

The EnSIE_IntStat bit is effective even during Sleep/Snooze.

7.2.15 11h EPrIntEnb (EPr Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
11h	EPrIntEnb		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
		R / W	2: EnEPcIntStat	0: Disable	1: Enable	
		R / W	1: EnEPbIntStat	0: Disable	1: Enable	
		R / W	0: EnEPaIntStat	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the EPrIntStat bit in the MainIntStat register for the interrupt causes contained in the EPrIntStat register.

7.2.16 12h *SIE_IntEnb (SIE Interrupt Enable)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
12h	***SIE_IntEnb***	R / W	7: ***EnVBUS_Changed***	0: Disable	1: Enable	00h
		R / W	6: ***EnNonJ***	0: Disable	1: Enable	
		R / W	5: ***EnFinishedPM***	0: Disable	1: Enable	
		R / W	5: EnDetectRESET	0: Disable	1: Enable	
		R / W	4: EnDetectSUSPEND	0: Disable	1: Enable	
		R / W	2: EnChirpCmp	0: Disable	1: Enable	
		R / W	1: EnRestoreCmp	0: Disable	1: Enable	
		R / W	0: EnSetAddressCmp	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the SEI_IntStat bit in the MainIntStat register for the interrupt causes contained in the SIE_IntStat register.

The EnVBUS_Changed, EnNonJ, and EnFinishedPM bits are effective even during Sleep/Snooze.

When in the Active60 state, the synchronous bits (bits 4–0) cannot be written to (in order to clear the cause of the interrupt), although they can be read normally. For details on the processing of these synchronous bits in the Active60 state, refer to the explanation of the SIE_IntStat register.

7.2.17 13h CPU_IntEnb (CPU Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
13h	CPU_IntEnb		7:	0:	1:	00h
		R / W	6: EnRcvSOF	0: Disable	1: Enable	
		R / W	5: EnDMA1_Countup	0: Disable	1: Enable	
		R / W	4: EnDMA1_Cmp	0: Disable	1: Enable	
			3:	0:	1:	
			2:	0:	1:	
		R / W	1: EnDMA0_Countup	0: Disable	1: Enable	
		R / W	0: EnDMA0_Cmp	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the CPU_IntStat bit in the MainIntStat register for the interrupt causes contained in the CPU_IntStat register.

7.2.18 14h FIFO_IntEnb (FIFO Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
14h	FIFO_IntEnb	R / W	7: EnDescriptorCmp	0: Disable	1: Enable	00h
		R / W	6: EnRAM_RdCmp	0: Disable	1: Enable	
		R / W	5: EnFIFO_IDE_Cmp	0: Disable	1: Enable	
		R / W	4: EnFIFO1_Cmp	0: Disable	1: Enable	
		R / W	3: EnFIFO_Full	0: Disable	1: Enable	
		R / W	2: EnFIFO_Empty	0: Disable	1: Enable	
			1:	0:	1:	
		R / W	0: EnFIFO0_Cmp	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the FIFO_IntStat bit in the MainIntStat register for the interrupt causes contained in the FIFO_IntStat register.

7.2.19 15h BulkIntEnb (Bulk Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
15h	BulkIntEnb	R / W	7: EnCBW_Cmp	0: Disable	1: Enable	00h
		R / W	6: EnCBW_LengthErr	0: Disable	1: Enable	
		R / W	5: EnCBW_Err		0: Disable	
		R / W	4:	0:	1:	
		R / W	3: EnCSW_Cmp	0: Disable	1: Enable	
		R / W	2: EnCSW_Err	0: Disable	1: Enable	
			1:	0:	1:	
			0:	0:	1:	

This register is used to enable or disable the assertion of the BulkIntStat bit in the MainIntStat register for the interrupt causes contained in the BulkIntStat register.

7.2.20 16h IDE_IntEnb (IDE Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
16h	IDE_IntEnb	R / W	7: EnIDE_RegCmp	0: Disable	1: Enable	00h
		R / W	6: EnIDE_RegErr	0: Disable	1: Enable	
		R / W	5: En_SeqWrRegCmp	0: Disable	1: Enable	
		R / W	4: EnCompleteINTRQ	0: Disable	1: Enable	
			3:	0:	1:	
		R / W	2: EnIDE_Cmp	0: Disable	1: Enable	
		R / W	1: EnDetectINTRQ	0: Disable	1: Enable	
		R / W	0: EnDetectTerm	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the IDE_IntStat bit in the MainIntStat register for the interrupt causes contained in the IDE_IntStat register.

7.2.21 17h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
17h	Reserved		7:	0:	1:	XXh
			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 EP0IntEnb (EP0 Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
18h	EP0IntEnb		7:	0:	1:	00h
		R / W	6: EnOUT_ShortACK	0: Disable	1: Enable	
		R / W	5: EnIN_TranACK	0: Disable	1: Enable	
		R / W	4: EnOUT_TranACK	0: Disable	1: Enable	
		R / W	3: EnIN_TranNAK	0: Disable	1: Enable	
		R / W	2: EnOUT_TranNAK	0: Disable	1: Enable	
		R / W	1: EnIN_TranErr	0: Disable	1: Enable	
		R / W	0: EnOUT_TranErr	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the EP0IntStat bit in the MainIntStat register for the interrupt causes contained in the EP0IntStat register.

7.2.23 19h EPaIntEnb (EPa Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
19h	EPaIntEnb		7:	0:	1:	00h
		R / W	6: EnOUT_ShortACK	0: Disable	1: Enable	
		R / W	5: EnIN_TranACK	0: Disable	1: Enable	
		R / W	4: EnOUT_TranACK	0: Disable	1: Enable	
		R / W	3: EnIN_TranNAK	0: Disable	1: Enable	
		R / W	2: EnOUT_TranNAK	0: Disable	1: Enable	
		R / W	1: EnIN_TranErr	0: Disable	1: Enable	
		R / W	0: EnOUT_TranErr	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the EPaIntStat bit in the EPrIntStat register for the interrupt causes contained in the EPaIntStat register.

7.2.24 1Ah EPbIntEnb (EPb Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
1Ah	EPbIntEnb		7:	0:	1:	00h
		R / W	6: EnOUT_ShortACK	0: Disable	1: Enable	
		R / W	5: EnIN_TranACK	0: Disable	1: Enable	
		R / W	4: EnOUT_TranACK	0: Disable	1: Enable	
		R / W	3: EnIN_TranNAK	0: Disable	1: Enable	
		R / W	2: EnOUT_TranNAK	0: Disable	1: Enable	
		R / W	1: EnIN_TranErr	0: Disable	1: Enable	
		R / W	0: EnOUT_TranErr	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the EPbIntStat bit in the EPrIntStat register for the interrupt causes contained in the EPbIntStat register.

7.2.25 1Bh EPcIntEnb (EPc Interrupt Enable)

Address	Register Name	R / W	Bit Symbol	Description		Reset
1Bh	EPcIntEnb		7:	0:	1:	00h
		R / W	6: EnOUT_ShortACK	0: Disable	1: Enable	
		R / W	5: EnIN_TranACK	0: Disable	1: Enable	
		R / W	4: EnOUT_TranACK	0: Disable	1: Enable	
		R / W	3: EnIN_TranNAK	0: Disable	1: Enable	
		R / W	2: EnOUT_TranNAK	0: Disable	1: Enable	
		R / W	1: EnIN_TranErr	0: Disable	1: Enable	
		R / W	0: EnOUT_TranErr	0: Disable	1: Enable	

This register is used to enable or disable the assertion of the EPcIntStat bit in the EPrIntStat register for the interrupt causes contained in the EPcIntStat register.

7.2.26 1Ch - 1Fh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
1Ch -1Fh	Reserved		7:	0:	1:	XXh
			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 *RevisionNum (Revision Number)*

Address	Register Name	R / W	Bit Symbol	Description	Reset
20h	***RevisionNum***	R	7: ***RevisionNum [7]***	Revision Number	30h
			6: ***RevisionNum [6]***		
			5: ***RevisionNum [5]***		
			4: ***RevisionNum [4]***		
			3: ***RevisionNum [3]***		
			2: ***RevisionNum [2]***		
			1: ***RevisionNum [1]***		
			0: ***RevisionNum [0]***		

This register indicates the revision number of the LSI stipulated herein. This register can be accessed even during Sleep/Snooze.

The revision number relating to the specifications of the LSI is 0x30.

7.2.28 21h *ChipReset (Chip Reset)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
21h	*ChipReset*	R / W	7: ***ResetUTM***	0: Do nothing	1: Reset UTM	80h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
		W	0: ***AllReset***	0: None	1: Reset	

This register is used to reset the LSI stipulated herein. This register can be accessed even during Sleep/Snooze.

Bit7 ***ResetUTM***

Setting this bit to 1 initializes the UTM block of the LSI.

To deassert the reset, clear this bit to 0.

Bit6-1 **Reserved**

Bit0 ***AllReset***

This bit resets the entire circuit of the LSI. It functions the same way as the external reset pin (XRST).

Do not write to this register unless the LSI needs to be reset.

Be aware that if writing to this register is attempted for other than resetting the LSI in violation of the A.C. characteristics (8.4.3), the LSI may operate erratically.

7.2.29 22h *PM_Control (Power Management Control)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
22h	***PM_Control***		7:	0:	1:	80h
		R	6: ***PM_State [2]***	PM_State [2:0]		
			5: ***PM_State [1]***			
			4: ***PM_State [0]***			
		R / W	3: ***GoSLEEP***	0: Do nothing	1: Go to Sleep	
		R / W	2: ***GoSNOOZE***	0: Do nothing	1: Go to Snooze	
		R / W	1: ***GoActive60***	0: Do nothing	1: Go to Active60	
		R / W	0: ***GoActiveALL***	0: Do nothing	1: Go to ActiveAll	

This register is used to set the operations relating to the power management of the LSI stipulated herein.

This register can be accessed even during Sleep/Snooze.

Bit7 Reserved

Bit6-4 *PM_State [2:0]*

Indicates the state of the power mode

000: Sleep state (OSC, PLL60, and PLL480 turned off)

001: Snooze state (OSC turned on, PLL60 and PLL480 turned off)

011: Active60 state (OSC and PLL60 turned on, PLL480 turned off)

111: ActiveALL state (OSC, PLL60, and PLL480 turned on)

Other: Unused

All registers in the register map, except those shown in italic bold face, can be accessed for both read/write in the Active60 and ActiveALL states.

The USB-related registers and bits can be accessed for read/write only in the ActiveALL state.

Note that these states are unstable for a transitory period from when PM_Control.GoXXXX is set until the SIE_IntStat.FinishedPM interrupt status is set and the PM_Control.GoXXXX bit is cleared. As such, do not refer to these states during that period.

Bit3 *GoSLEEP*

This bit causes the LSI to start entering the Sleep state from any other state.

When this bit is set to 1 in the Snooze state, the LSI turns off the oscillator, thereby entering the Sleep state.

When this bit is set to 1 in the Active60 state, the LSI first turns off the PLL60 and then the oscillator, thereby entering the Sleep state.

When this bit is set to 1 in the ActiveALL state, the LSI first turns off the PLL480, then the PLL60, and finally the oscillator, thereby entering the Sleep state.

Upon completion of the state transition, regardless of which state it has occurred from, this bit is automatically cleared and, at the same time, the SIE_IntStat.FinishedPM bit is set.

Bit2 ***GoSNOOZE***

This bit causes the LSI to start entering the Snooze state from any other state.

When this bit is set to 1 in the Sleep state, the LSI turns on the oscillator and, after the elapse of the oscillator's oscillation stabilization time (set by WakeupTim_H,L), enters the Snooze state.

When this bit is set to 1 in the Active60 state, the LSI first turns on the PLL60, thereby entering the Snooze state.

When this bit is set to 1 in the ActiveALL state, the LSI first turns on the PLL480 and then the PLL60, thereby entering the Snooze state.

Upon completion of the state transition, regardless of which state it has occurred from, this bit is automatically cleared and, at the same time, the SIE_IntStat.FinishedPM bit is set.

Bit1 ***GoActive60***

This bit causes the LSI to start entering the Active60 state from any other state.

When this bit is set to 1 in the Sleep state, the LSI turns on the oscillator and, after the elapse of the oscillator's oscillation stabilization time (set by WakeupTim_H,L), turns on the PLL60. Then, after the elapse of the PLL oscillation stabilization time (approx. 250 μs), the LSI enters the Active60 state.

When this bit is set to 1 in the Snooze state, the LSI turns on the PLL60 and, after the elapse of the PLL oscillation stabilization time (approx. 250 μs), enters the Active60 state.

When this bit is set to 1 in the ActiveALL state, the LSI turns off the PLL480, thereby entering the Active60 state.

Upon completion of the state transition, regardless of which state it has occurred from, this bit is automatically cleared and, at the same time, the SIE_IntStat.FinishedPM bit is set.

Bit0 ***GoActiveALL***

This bit causes the LSI to start entering the ActiveALL state from any other state.

When this bit is set to 1 in the Sleep state, the LSI turns on the oscillator and, after the elapse of the oscillator's oscillation stabilization time (set by WakeupTim_H,L), turns on the PLL60. Then, after the elapse of the PLL oscillation stabilization time (approx. 250 μs), the LSI turns on the PLL480 and enters the ActiveALL state after the elapse of the PLL oscillation stabilization time (approx. 250 μs).

When this bit is set to 1 in the Snooze state, the LSI turns on the PLL60 and, after the elapse of the PLL oscillation stabilization time (approx. 250 μs), turns on the PLL480 and enters the ActiveALL state after the elapse of the PLL oscillation stabilization time (approx. 250 μs).

When this bit is set to 1 in the Active60 state, the LSI turns on the PLL480 and, after the elapse of the PLL oscillation stabilization time (approx. 250 μs), enters the ActiveALL state.

Upon completion of the state transition, regardless of which state it has occurred from, this bit is automatically cleared, and at the same time, the SIE_IntStat.FinishedPM bit is set.

* The LSI stipulated herein has its XINT signal masked so as not to be asserted during Snooze by an interrupt status that cannot be accessed during Sleep/Snooze (hereinafter referred to as a "synchronous status"). However, to ensure that the XINT pin will not be asserted at the same time the LSI has exited Snooze, the firmware should execute the processing described below.

<Before entering Sleep/Snooze>

- Process the synchronous status to clear it (-IntStat).
- Disable the synchronous status (-IntEnb).

<After exiting Sleep/Snooze>

- Clear the synchronous status (-IntStat).
- Reenable the synchronous status (-IntEnb).

7.2.30 23h USB_Control (USB_Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
23h	USB_Control	R / W	7: DisBusDetect	0: Enable BusDetect	1: Disable BusDetect	00h
		R / W	6: EnAutoNego	0: Disable AutoNegotiation	1: Enable AutoNegotiation	
		R / W	5: InSUSPEND	0: Do nothing	1: Monitor NonJ	
		R / W	4: DisableHS	0: HS mode	1: Disable HS mode	
		R / W	3: SendWakeup	0: Do nothing	1: Send Remotewakeup Signal	
		R / W	2: RestoreUSB	0: Do nothing	1: Restore operation mode	
		R / W	1: GoChirp	0: Do nothing	1: Do Chirp sequence	
		R / W	0: ActiveUSB	0: Disactivate USB	1: Activate USB	

This register is used to set the USB-related operations. In the Active60 state, this register can be read out, but cannot be written to.

Bit7 DisBusDetect

Setting this bit to 1 disables the automatic detection of the reset/suspend states of the USB. If this bit remains cleared to 0, bus activities on the USB bus are monitored in order to detect the reset/suspend states of the USB.

In HS mode, if no bus activities are detected for a 3-ms period, the mode is automatically switched to FS mode, and it is then determined whether the USB is in a reset or suspend state, according to which the relevant cause of the interrupt (DetectRESET or DetectSUSPEND) is set. In FS mode, if no bus activities are detected for a 3-ms period, a suspend state of the USB is assumed. Furthermore, if SE0 with a duration of 2.5 μs or more is detected, a reset is assumed, and the relevant cause of the interrupt is set.

When the DetectRESET or DetectSUSPEND bit is set, set the DisBusDetect bit to 1 in order to disable the detection of states while a reset or suspend state of the USB continues. When the AutoNegotiation function is used, do not set this bit to 1.

Bit6 EnAutoNego

This bit enables the AutoNegotiation function, which automates a series of sequences during reset detection until the speed mode is determined after a speed negotiation has finished. For details on the AutoNegotiation function, refer to the relevant section in Chapter 6, “Functional Description.”

Bit5 InSUSPEND

When, during use of the AutoNegotiation function, a suspend state of the USB is detected, this bit is automatically set to 1 in order to enable the function to detect the NonJ state. Clear this bit to 0 when returning from a suspend state of the USB.

For a detailed explanation of the AutoNegotiation function, refer to “Auto-Negotiation Function” in Chapter 6, “Functional Description.”

Bit4 DisableHS

If this bit has been set to 1 when GoChirp is set to 1, the LSI is forcibly placed in FS mode without sending out DeviceChirp, and generates a ChirpCmp interrupt.

Bit3 SendWakeup

When this bit is set to 1, a RemoteWakeup signal (K) is output to the USB port.

After the elapse of 1 ms or more but less than 15 ms after the LSI started sending out the RemoteWakeup signal, clear this bit to 0 in order to stop the transmission.

Bit2 RestoreUSB

If this bit is set to 1 when the USB is resumed from a suspend state, the operation mode (FS or HS) is automatically switched to that of the USB prior to Suspend, and the relevant cause of the interrupt (RestoreCmp) is set.

This bit is automatically cleared to 0 upon completion of the operation.

When using the AutoNegotiation function, do not set or clear this bit, as its function is automatically controlled during that time.

Bit1 GoChirp

If this bit is set to 1 when the USB has been reset, HS Detection Handshaking is executed between the host and hub, and the TermSelect and XcvrSelect bits in the XcvrControl register and the FSxHS bit in the USB_Status register are automatically set. The cause of the interrupt (ChirpCmp) is set upon completion of the operation.

This bit is automatically cleared to 0 upon completion of the operation. The result of HS Detection Handshaking can be confirmed by inspecting the FSxHS bit in the USB_Status register.

When using the AutoNegotiation function, do not set or clear this bit, as its function is automatically controlled during that time.

Bit0 ActiveUSB

In the LSI stipulated herein, this bit is cleared to 0 and remains cleared after a hardware reset, so that the entire function of the USB is disabled. Therefore, after setting up the LSI, set this bit to 1 in order to enable the USB.

7.2.31 24h *USB_Status (USB_Status)*

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

This register shows the USB-related status.

Bit7 *VBUS*

This bit indicates the status of the VBUS pin. It is effective even during Sleep/Snooze.

Bit6 FSxHS

This bit indicates the current operation mode. It is automatically set when HS Detection Handshaking is executed by means of the USB_Control.GoChirp bit (see "Functional Description"). Although the operation mode can be forcibly changed by writing to this bit, it is recommended that this bit be manipulated when the operation mode only needs to be changed without performing HS Detection Handshaking during a simulation or the like. In the Active60 state, this bit can be read, but cannot be written to.

This bit should be set to 1 (= FS mode) when the cable is attached.

Bit5-2 Reserved

Bit1-0 *LineState [1:0]*

This bit indicates the status of signal on the USB cable. This bit is effective even during Sleep/Snooze.

If the XcvrSelect bit = 1 (FS transceiver selected) when the XcvrControl register's TermSelect bit = 1 (FS termination selected), it indicates the received value of the DP/DM FS receiver. If the XcvrSelect bit = 0 (HS transceiver selected), it indicates the received value of the HS receiver.

When TermSelect = 0, it indicates USB bus activity.

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

7.2.32 25h XcvrControl (Xcvr Control)

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

This register is used to make settings relating to the transceiver macro. In the Active60 state, this register can be read, but cannot be written to.

Bit7 TermSelect

This bit enables FS or HS termination. It is automatically set when HS Detection Handshaking is executed by means of the GoChirp bit in the USB_Control register, or when the AutoNegotiation function is executed after the EnAutoNego bit in the USB_Control register has been set.

Bit6 XcvrSelect

This bit selects the FS or HS transceiver to enable it. It is automatically set when HS Detection Handshaking is executed by means of the GoChirp bit in the USB_Control register, or when the AutoNegotiation function is executed after the EnAutoNego bit in the USB_Control register has been set.

Bit5-2 Reserved

Bit1-0 OpMode

These bits set the operation mode of the UTM.

They do not normally need to be set unless the USB cable has been removed (*), or unless the USB is in a suspend state or in test mode.

OpMode		
00	"Normal Operation"	Normal operating state
01	"Non-Driving"	Set to this state when the USB cable has been removed.
10	"Disable Bitstuffing and NRZI encoding"	Set to this state when the USB is in test mode.
11	"Power-Down"	Set to this state when the USB is suspended.

* It is recommended that, when the USB cable has been removed, this register be set to “41h.”

7.2.33 26h USB_Test (USB_Test)

Address	Register Name	R / W	Bit Symbol	Description		Reset
26h	USB_Test	R / W	7: EnHS_Test	0: Do nothing	1: EnHS_Test	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R / W	3: Test_SE0_NAK	0: Do nothing	1: Test_SE0_NAK	
		R / W	2: Test_J	0: Do nothing	1: Test_J	
		R / W	1: Test_K	0: Do nothing	1: Test_K	
		R / W	0: Test_Packet	0: Do nothing	1: Test_Packett	

This register is used to set the operations relating to USB2.0 test mode. To enter any test mode defined in the USB2.0 standard, set the bit corresponding to the test mode specified in a SetFeature request. After the status stage has finished, set the EnHS_Test bit to 1. In the Active60 state, this register can be read out, but cannot be written to.

Bit7 EnHS_Test

If, when this bit is set to 1, any of the 4 low-order bits in the USB_Test register has been set to 1, the test mode corresponding to that bit is entered. Before test mode can be entered, the USB_Control register's DisBusDetect bit must be set to 1 in order to disable the detection of suspend/reset states of the USB. Furthermore, the AutoNegotiation function must be disabled by clearing the USB_Control register's EnAutoNego bit to 0.

In addition, make sure the test mode is entered after the status stage in a SetFeature request has finished.

Bit6-4 Reserved

Bit3 Test_SE0_NAK

Test_SE0_NAK test mode can be entered by setting the EnHS_Test bit to 1 after setting this bit to 1.

Bit2 TEST_J

Test_J test mode can be entered by setting the EnHS_Test bit to 1 after setting this bit to 1. This test mode requires that TermSelect and XcvrSelect in the XcvrControl register be set according to the speed mode before the EnHS_Test bit is set to 1, and that OpMode be set to "10" (= Disable Bitstuffing and NRZI encoding).

Bit1 TEST_K

Test_K test mode can be entered by setting the EnHS_Test bit to 1 after setting this bit to 1. This test mode requires that TermSelect and XcvrSelect in the XcvrControl register be set according to the speed mode before the EnHS_Test bit is set to 1, and that OpMode be set to "10" (= Disable Bitstuffing and NRZI encoding).

Bit0 Test_Packet

Test_Packet test mode can be entered by setting the EnHS_Test bit to 1 after setting this bit to 1.

As this test mode can be used at any endpoint other than EP0, the following settings must be made:

1) Set MaxPacketSize, transfer direction, and EndpointNumber for the endpoint to be used in test mode to 64 or more, IN, and "0xF," respectively, and set the EnEndpoint bit to 1 to make the endpoint usable.
2) Make sure that other endpoint settings do not overlap the above endpoint setting. Or clear the EnEndpoint bit for other endpoints.
3) Clear the FIFO for the endpoint to be used in test mode and write the data for test packets shown below into the FIFO. Clear the EnIN_TranErr bit in the interrupt enable register to 0.
4) Each time test-packet transmission finishes, the IN_TranErr status is set to 1.

The following 53 bytes are the data to be written into the FIFO in packet-transmission 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

As the SIE adds PID and CRS to a test packet when it is transmitted, the data to be written into the FIFO should consist of only a range of the test-packet data stipulated in USB standard Rev. 2.0 from the data item next to DATA0 PID to those that follow, but not including CRC16.

7.2.34 27h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
27h	Reserved		7:	0:	1:	XXh
			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 EPnControl (Endpoint Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
28h	EPnControl	W	7: AllForceNAK	0: Do nothing	1: Set All ForceNAK	XXh
		W	6: EPrForceSTALL	0: Do nothing	1: Set EP's ForceSTALL	
		W	5: AllFIFO_Clr	0: Do nothing	1: Clear All FIFO	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
		W	0: EP0FIFO_Clr	0: Do nothing	1: Clear EP0 FIFO	

This register is used to set the endpoint operations. This is a write-only register.

Bit7 AllForceNAK

This bit sets the ForceNAK bits for all endpoints to 1.

Bit6 EPrForceSTALL

This bit sets the ForceSTALL bits for the endpoints EPa, EPb, and EPc to 1.

Bit5 AllFIFO_Clr

This bit clears the FIFOs for all endpoints. When the areas for the respective endpoints have been set, temporarily set this bit to 1 in order to clear the FIFOs for all endpoints after the settings have been completed. This bit is automatically cleared to 0 after the FIFOs have been successfully cleared.

When DMAx{x=0,1} has been joined to any endpoint and the relevant DMA is active (while DMA_Running bit = 1), do not set the bit for that endpoint to 1.

Bit4-1 Reserved

Bit0 EP0FIFO_Clr

This bit clears the FIFO for the endpoint EP0.

When set to 1, this bit only clears the FIFO and does not hold the value written to it.

When DMAx{x=0,1} has been joined to the endpoint EP0 and the relevant DMA is active (while DMA_Running bit = 1), do not set this bit to 1.

7.2.36 29h EPrFIFO_Clr (Endpoint FIFO Clear)

Address	Register Name	R / W	Bit Symbol	Description		Reset
29h	EPrFIFO_Clr		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
		W	2: EPcFIFO_Clr	0: Do nothing	1: Clear EPc FIFO	
		W	1: EPbFIFO_Clr	0: Do nothing	1: Clear EPb FIFO	
		W	0: EPaFIFO_Clr	0: Do nothing	1: Clear EPa FIFO	

This register is used to clear the FIFO for the relevant endpoint.

This is a write-only register. When set to 1, any bit in this register only clears the FIFO and does not hold the value written to it.

When DMAx{x=0,1} has been joined to any endpoint and the relevant DMA is active (while DMA_Running bit = 1), do not set the bit for that endpoint to 1.

7.2.37 2Ah ClrAllJoin (Clear All Join)

Address	Register Name	R / W	Bit Symbol	Description		Reset
2Ah	ClrAllJoin	W	7: ClrJoinIDE	0: Do nothing	1: Clear Join IDE	XXh
			6: ClrJoinFIFO_Status	0: Do nothing	1: Clear Join FIFO_Status	
		W	5: ClrJionDMA0_Rd	0: Do nothing	1: Clear Join DMA0_Rd	
		W	4: ClrJoinDMA0_Wr	0: Do nothing	1: Clear Join DMA0_Wr	
		W	3: ClrJoinDMA1_Rd	0: Do nothing	1: Clear Join DMA1_Rd	
		W	2: ClrJoinDMA1_Wr	0: Do nothing	1: Clear Join DMA1_Wr	
		W	1: ClrJoinCPU_Rd	0: Do nothing	1: Clear Join CPU_Rd	
		W	0: ClrJoinCPU_Wr	0: Do nothing	1: Clear Join CPU_Wr	

This register is used to clear the connection of each corresponding port and endpoints.

The bits in this register are automatically cleared to 0 after the connection has been cleared.

When endpoints are connected to ports (relevant bits in the EPx{x=0,a-c}Join register = 1) and each port is active, do not set the bits in this register to 1. Such a register access may cause the LSI to operate erratically.

7.2.38 2Bh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
2Bh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.39 2Ch BulkOnlyControl (BulkOnly Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
2Ch	BulkOnlyControl	R / W	7:AutoForceNAK_CBW	0: None	1: AutoForceNAK after CBW	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
		R / W	2: GoCBW_Mode	0:	1: Begin CBW Mode	
		R / W	1: GoCSW_Mode	0:	1: Begin CSW Mode	
			0:	0:	1:	

This register controls the Bulk-Only Support function.

Bit7 AutoForceNAK_CBW

If this bit is set to 1, the ForceNAK bit for the relevant endpoint is set to 1 when the OUT transaction in which a CBW is to receive for CBW support has finished.

Bit6-3 Reserved

Bit2 GoCBW_Mode

When this bit is set to 1, CBW support is executed at the relevant endpoint. For the endpoints at which CBW support will be executed, refer to the section on the BulkOnlyConfig register.

Bit1 GoCSW_Mode

When this bit is set to 1, CSW support is executed at the relevant endpoint. For the endpoints at which CSW support will be executed, refer to the section on the BulkOnlyConfig register.

Bit0 Reserved

7.2.40 2Dh BulkOnlyConfig (BulkOnly Configuration)

Address	Register Name	R / W	Bit Symbol	Description		Reset
2Dh	BulkOnlyConfig		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
		R / W	2: EPcBulkOnly	0: None	1: Enable BulkOnly on EPc	
		R / W	1: EPbBulkOnly	0: None	1: Enable BulkOnly on EPb	
			0:	0:	1:	

This register enables the Bulk-Only Support function.

Bit7-3 Reserved

Bit2 EPcBulkOnly

Setting this bit to 1 enables the Bulk-Only Support function for the endpoint EPc. If the endpoint EPc is an OUT endpoint when the Bulk-Only Support function is enabled in this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Alternatively, if the endpoint EPc is an IN endpoint, CSW support can be executed by setting the BulkOnlyControl.GoCSW_Mode bit.

Do not enable the Bulk-Only Support function for two or more OUT endpoints at the same time.

Bit1 EPbBulkOnly

Setting this bit to 1 enables the Bulk-Only Support function for the endpoint EPb. If the endpoint EPb is an OUT endpoint when the Bulk-Only Support function is enabled in this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Alternatively, if the endpoint EPb is an IN endpoint, CSW support can be executed by setting the BulkOnlyControl.GoCSW_Mode bit.

Do not enable the Bulk-Only Support function for two or more OUT endpoints at the same time.

Bit0 Reserved

7.2.41 2Eh *WakeupTim_H (Wakeup Time High)*

7.2.42 2Fh *WakeupTim_L (Wakeup Time Low)*

Address	Register Name	R / W	Bit Symbol	Description	Reset
2Eh -2Fh	***WakeupTim_H*** ***WakeupTim_L***	R / W	***WakeupTim [15:0]***	Wakeup Time [15:0]	0000h

These registers are used to set the oscillator's oscillation stabilization time to be allowed to elapse when the LSI returns from the Sleep state to the Snooze state. These registers can be accessed even during Sleep.

When the PM_Control.GoActiveALL, PM_Control.GoActive60, or PM_Control.GoSNOOZE bit is set by writing 1 in the Sleep state, the oscillator cell is enabled, causing the oscillator to start oscillating. At this time, the counter is loaded with the set value of these WakeupTim_H,L registers and starts counting down synchronously with the rising edge of the OSC. When the counter has finished counting down, the gate for the internal OSCCLK is opened, allowing CLK to be sent out to the PLL and other circuits.

This oscillation stabilization time varies depending on the resonator, oscillator cell, circuit board, and load capacitance. If it is necessary to place the LSI in the Sleep state during Suspend of the USB, the internal SCLK must be stabilized to 60 MHz ± 10% within 5.1 ms after Reset of the USB is detected.

Therefore, the sum total of the following must be 5.1 ms or less:

> Oscillator's oscillation stabilization time + PLL60 stabilization time (less than 250 μs) + PLL480 stabilization time (less than 250 μs)

7.2.43 30h EP0SETUP_0 (EP0 SETUP 0)

7.2.44 31h EP0SETUP_1 (EP0 SETUP 1)

7.2.45 32h EP0SETUP_2 (EP0 SETUP 2)

7.2.46 33h EP0SETUP_3 (EP0 SETUP 3)

7.2.47 34h EP0SETUP_4 (EP0 SETUP 4)

7.2.48 35h EP0SETUP_5 (EP0 SETUP 5)

7.2.49 36h EP0SETUP_6 (EP0 SETUP 6)

7.2.50 37h EP0SETUP_7 (EP0 SETUP 7)

Address	Register Name	R / W	Bit Symbol	Description	Reset
30h -37h	EP0SETUP_0 -EP0SETUP_7	R	7: EP0SETUP_n [7]	Endpoint 0 SETUP Data 0 -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]		

The 8 bytes of data received in the setup stage of the endpoint EP0 are stored in these registers sequentially, beginning with EP0SETUP_0.

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.

7.2.51 38h USB_Address (USB Address)

Address	Register Name	R / W	Bit Symbol	Description		Reset
38h	USB_Address	R / W	7: SetAddress	0: none	1: set USB address	00h
		R (W)	6: USB_Address [6]	USB Address		
			5: USB_Address [5]			
			4: USB_Address [4]			
			3: USB_Address [3]			
			2: USB_Address [2]			
			1: USB_Address [1]			
			0: USB_Address [0]			

This register is used by the AutoSetAddress function to set a USB address.

When a SetAddress() request is received, the AutoSetAddress function automatically performs a control transfer for it. When the status stage for the control transfer associated with the SetAddress() request is completed, the AutoSetAddress function sets USB_Address and then issues the SetAddressCmp status. During the Active60 state, this register can be read, but cannot be written to.

Bit7 SetAddress

If this bit is set when a SetAddress request is received, USB_Address is automatically set when a status stage of the request is completed. Automatic address setup mode must be disabled for this bit setting to have any effect.

Bit6-0 USB_Address

These bits are used to set a USB address.

The address is automatically written to by the AutoSetAddress function.

Any value can be written to these bits in the software, but when a SetAddress() request is received, the value will be automatically rewritten.

7.2.52 39h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
39h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.53 3Ah SETUP_Control(SETUP Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
39h	SETUP_Control		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
		R / W	0: ProtectEP0	0: None	1: Protect EP0	

This register is used for control-transfer-related settings.

Bit7-1 Reserved

Bit0 ProtectEP0

This bit is set to 1 when, after the setup stage of a control transfer has finished, the received data is stored in the registers EP0SETUP_0 through EP0SETUP_7.

At the same time, the ForceSTALL bits in the EP0ControlIN and EP0ControlOUT registers are cleared to 0 and the ForceNAK and ToggleStat bits in those registers are set to 1, all automatically.

The ProtectEP0 bit is set when a SETUP transaction is performed. Therefore, it is also set for the SetAddres() request received.

The ForceNAK and ForceSTALL bits for EP0 cannot have their settings altered while this bit remains set.

7.2.54 3Bh - 3Dh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
3Bh	Reserved		7:	0:	1:	XXh
-3Dh			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.55 3Eh FrameNumber_H (FrameNumber High)

7.2.56 3Fh FrameNumber_L (FrameNumber Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
3Eh	FrameNumber_H	R	7: FnInvalid	0: Frame number is valid	1: Frame number is not valid	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]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
3Fh	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]		

These registers show the frame number of the USB that is updated for each SOF token received. To obtain a frame number, the FrameNumber_H and FrameNumber_L registers must be accessed in pairs. In such a case, be sure to access the FrameNumber_H register first.

3Eh.Bit7 FnInvalid

This bit is set to 1 when an error occurs in a received SOF packet.

3Eh.Bit6-3 Reserved

3Eh.Bit2-0, 3Fh.Bit7-0 FrameNumber [10:0]

These bits show the FrameNumber of the received SOF packet.

7.2.57 40h EP0MaxSize (EP0 Max Packet Size)

Address	Register Name	R / W	Bit Symbol	Description		Reset
40h	EP0MaxSize		7:	0:	1:	40h
		R / W	6: EP0MaxSize [6]	Endpoint [0] Max Packet Size		
			5: EP0MaxSize [5]			
			4: EP0MaxSize [4]			
			3: EP0MaxSize [3]			
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

This register is used to set the endpoint EP0.

Bit7 Reserved

Bit6-3 EP0MaxSize [6:3]

These bits set the MaxPacketSize of the endpoint EP0.

Any size can be selected from among those listed below for use with this endpoint:

During FS 8, 16, 32, or 64 bytes

During HS 64 bytes

Bit2-0 Reserved

7.2.58 41h EP0Control (EP0 Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
41h	EP0Control	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:	
		R / W	0: ReplyDescriptor	0: Do nothing	1: Reply Descriptor	

This register sets the endpoint EP0.

Bit7 INxOUT

This bit sets the direction of transfer for the endpoint EP0.

Consider the request received in the setup stage when setting a value in this bit.

If a data stage exists, set the direction of transfer at the data stage in this bit. When the setup stage finishes, the ForceNAK bits in the EP0ControlIN and EP0ControlOUT registers are set, so be sure to clear these bits when executing the data and status stages.

When the data stage has finished, reset this bit as suited to the direction of the status stage. If the direction of the data stage is IN, the status stage is directed OUT, so set 0 in this bit. Conversely, if the direction of the data stage is OUT or a data stage is nonexistent, the status stage is directed IN, so clear the FIFO for the endpoint EP0 and set 1 in this bit.

An IN or OUT transaction in the direction opposite the set value in this bit is responded to with NAK. However, if the ForceSTALL bit in the EP0ControlIN or EP0ControlOUT register for that transaction has been set, STALL is returned.

Bit6-1 Reserved

Bit0 ReplyDescriptor

This bit executes the descriptor reply function.

When this bit is set to 1, bytes of descriptor data equal to MaxPacketSize are sent back to the host from the FIFO in response to the IN transaction for the endpoint EP0. Here, descriptor data refers to data that starts from the address set in the DescAdr_H,L registers, and whose size is set in the DescSize_H,L registers. As these values are updated during execution of the descriptor reply function, they must be set each time the ReplyDescriptor bit is set.

The DescAdr_H,L registers are incremented for each transaction performed by an amount equal to the transmitted data bytes, and the DescSize_H,L registers are decremented by an amount equal to the transmitted data bytes.

When the transmit operation has finished after the bytes of data set by DescSizeH,L have been transmitted, and when a transaction other than an IN transaction has been performed, the descriptor reply function is terminated and the ReplyDescriptor bit is cleared to 0. At the same time, both the DescriptorCmp

bit in the EPnIntStat register and the IN_TranACK bit in the EP0IntStat register are set to 1.

For more detailed explanations, refer to Chapter 6, “Functional Description”

7.2.59 42h EP0ControlIN (EP0 Control IN)

Address	Register Name	R / W	Bit Symbol	Description		Reset
42h	EP0ControlIN		7:	0:	1:	00h
		R / W	6: EnShortPkt	0: Do nothing	1: Enable short Packet	
			5:	0:	1:	
		R	4: ToggleStat	Toggle sequence bit		
		W	3: ToggleSet	0: Do nothing	1: Set Toggle sequence bit	
		W	2: ToggleClr	0: Do nothing	1: Clear Toggle sequence bit	
		R / W	1: ForceNAK	0: Do nothing	1: Force NAK	
		R / W	0: ForceSTALL	0: Do nothing	1: Force STALL	

This register sets the operations relating to IN transactions at the endpoint EP0, and shows the status of those operations.

Bit7 Reserved

Bit6 EnShortPkt

Setting this bit to 1 allows data in the FIFO that is smaller than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EP0. Upon completion of an IN transaction in which a short packet has been transmitted, this bit is automatically cleared to 0. If a packet equal to MaxPacketSize has been transmitted, this bit is not cleared.

If this bit is set to 1 while no data exists in the FIFO, a packet with a length of zero can be transmitted in response to an IN token from the host. If data is written to the FIFO while a packet in it is being transmitted after this bit was set, the written data may also be transmitted, depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission finishes and this bit is cleared.

Bit5 Reserved

Bit4 ToggleStat

This bit shows the status of the toggle sequence bit in an IN transaction at the endpoint EP0.

Bit3 ToggleSet

This bit sets the toggle sequence bit in an IN transaction at the endpoint EP0 to 1. If this bit is set at the same time the ToggleClr bit is set, the function of the ToggleClr bit has priority.

Bit2 ToggleClr

This bit clears the toggle sequence bit in an IN transaction at the endpoint EP0 to 0. If this bit is set at the same time the ToggleSet bit is set, the function of this bit has priority.

Bit1 ForceNAK

Setting this bit to 1 returns NAK for an IN transaction at the endpoint EP0, irrespective of the number of data bytes in the FIFO.

When the RcvEP0SETUP bit in the MainIntStat register is set to 1 upon completion of a setup stage, this bit is set to 1; while the RcvEP0SETUP bit

remains set at 1, the bit cannot be cleared to 0. Furthermore, this bit is set to 1 upon completion of an IN transaction in which a short packet was transmitted.

If a transaction is already being executed when the software attempts to set this bit to 1, the bit is not set until the transaction is complete, at which time it is set to 1. If no transactions are being conducted, the bit is set to 1 immediately.

Bit0 ForceSTALL

Setting this bit to 1 returns STALL for an IN transaction at the endpoint EP0. This bit has priority over the ForceNAK bit.

When the RcvEP0SETUP bit in the MainIntStat register is set to 1 upon completion of a setup stage, this bit is cleared to 0; while the RcvEP0SETUP bit remains set at 1, this bit cannot be set to 1.

If any transaction is currently underway and this bit is set a certain length of time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

7.2.60 43h EP0ControlOUT (EP0 Control OUT)

Address	Register Name	R / W	Bit Symbol	Description		Reset
43h	EP0ControlOUT	R / W	7: AutoForceNAK	0: Do nothing	1: Auto Force NAK	00h
			6:	0:	1:	
			5:	0:	1	
		R	4: ToggleStat	Toggle sequence bit		
		W	3: ToggleSet	0: Do nothing	1: Set Toggle sequence bit	
		W	2: ToggleClr	0: Do nothing	1: Clear Toggle sequence bit	
		R / W	1: ForceNAK	0: Do nothing	1: Force NAK	
		R / W	0: ForceSTALL	0: Do nothing	1: Force STALL	

This register sets the operations relating to OUT transactions at the endpoint EP0, and shows the status of those operations.

Bit7 AutoForceNAK

When an OUT transaction at the endpoint EP0 finishes normally, the ForceNAK bit in this register is set to 1.

Bit6-5 Reserved

Bit4 ToggleStat

This bit shows the status of the toggle sequence bit in an OUT transaction at the endpoint EP0.

Bit3 ToggleSet

This bit sets the toggle sequence bit in an OUT transaction at the endpoint EP0 to 1. If this bit is set at the same time the ToggleClr bit is set, the function of the ToggleClr bit has priority.

Bit2 ToggleClr

This bit clears the toggle sequence bit in an OUT transaction at the endpoint EP0 to 0. If this bit is set at the same time the ToggleSet bit is set, the function of this bit has priority.

Bit1 ForceNAK

Setting this bit to 1 returns NAK for an OUT transaction at the endpoint EP0, irrespective of the amount of free space in the FIFO.

When the RcvEP0SETUP bit in the MainIntStat register is set to 1 upon completion of a setup stage, this bit is set to 1; while the RcvEP0SETUP bit remains set at 1, this bit cannot be cleared to 0.

If a transaction is already being executed when the software attempts to set this bit to 1, the bit is not set until the transaction is complete, at which time it is set to 1. If no transactions are being executed, the bit is set to 1 immediately.

Bit0 ForceSTALL

Setting this bit to 1 returns STALL for an OUT transaction at the endpoint EP0. This bit has priority over the ForceNAK bit.

When the RcvEP0SETUP bit in the MainIntStat register is set to 1 upon completion of a setup stage, this bit is cleared to 0; while the RcvEP0SETUP bit remains set at 1, this bit cannot be set to 1.

If any transaction is currently underway and this bit is set a certain length of time after the transaction has started, the setting of this bit takes effect beginning with the next transaction.

7.2.61 44h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
44h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.62 45h EP0Join (End Point0 Join)

Address	Register Name	R / W	Bit Symbol	Description		Reset
45h	EP0Join		7:	0:	1:	00h
			6:	0:	1	
		R / W	5: JoinDMA0_Rd	0: Do nothing	1: Join EP0 to DMA0_Rd	
		R / W	4: JoinDMA0_Wr	0: Do nothing	1: Join EP0 to DMA0_Wr	
		R / W	3: JoinDMA1_Rd	0: Do nothing	1: Join EP0 to DMA1_Rd	
		R / W	2: JoinDMA1_Wr	0: Do nothing	1: Join EP0 to DMA1_Wr	
		R / W	1: JoinCPU_Rd	0: Do nothing	1: Join EP0 to CPU_Rd	
		R / W	0: JoinCPU_Wr	0: Do nothing	1: Join EP0 to CPU_Wr	

This register specifies the port with which a data transfer to or from endpoint 0 is to be performed.

Bit7-6 Reserved

Bit5 JoinDMA0_Rd

Performs a read transfer on DMA0 with the FIFO for the endpoint EP0. Specifically, when a read on DMA0 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit4 JoinDMA0_Wr

Performs a write transfer on DMA0 with the FIFO for the endpoint EP0. Specifically, when a write on DMA0 is performed, data is written into this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit3 JoinDMA1_Rd

Performs a read transfer on DMA1 with the FIFO for the endpoint EP0. Specifically, when a read on DMA1 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit2 JoinDMA1_Wr

Performs a write transfer on DMA1 with the FIFO for the endpoint EP0. Specifically, when a write on DMA1 is performed, data is written into this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit1 JoinCPU_Rd

Performs a read transfer for the CPU register access with the FIFO for the endpoint EP0. Specifically, when a read of the EPnFIFO_Rd_H,L registers or EPnFIFO_ByteRd register is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit0 JoinCPU_Wr

Performs a write transfer for the CPU register access with the FIFO for the endpoint EP0. Specifically, when a write to the EPnFIFO_Wr_H,L registers is performed, data is written into this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

When the JoinDMAx{x=0,1}_Rd or JoinDMAx{x=0,1}_Wr bit has been set, it is possible to determine the number of the remaining bytes of data in the FIFO if the endpoint is directed OUT, or the amount of free space if the endpoint is directed IN, by inspecting the DMAx{x=0,1}_Remain_H,L registers.

When the JoinCPU_Rd or JoinCPU_Wr bit has been set, it is possible to write or read data to or from the EPnFIFO_Rd_H,L, EPnFIFO_ByteRd, or EPnFIFO_Wr_H,L registers after inspecting the EPnRdRemain_H,L or EPnWrRemain_H,L registers.

Bits 5 to 0 in this register can be set to 1 only one bit at a time. If the software writes 1 to multiple bits at the same time, the most significant bit is set.

7.2.63 46h - 4Fh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
46h -4Fh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.64 50h Reserved ()

7.2.65 51h EPaMaxSize_L (EPa Max Packet Size Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
50h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0	0:	1:	

Address	Register Name	R / W	Bit Symbol	Description		Reset
51h	EPaMaxSize_L		7:	0:	1:	00h
		R / W	6: MaxSize [6]	Endpoint [a] Max Packet Size		
			5: MaxSize [5]			
			4: MaxSize [4]			
			3: MaxSize [3]			
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

These registers set MaxPacketSize.

50h.Bit7-0 Reserved

51h.Bit7 Reserved

51h.Bit6-3 EPaMaxSize [6:3]

These bits set MaxPacketSize for the endpoint EPa. To use this endpoint for bulk transfers in FS, set MaxSize to any of the following:

During FS 8, 16, 32, or 64 bytes

To use this endpoint for interrupt transfers, set MaxSize to any desired size within the following limits:

During FS Up to 64 bytes

During HS Up to 64 bytes

51h.Bit2-0 Reserved

7.2.66 52h EPaConfig_0 (EPa Configuration 0)

Address	Register Name	R / W	Bit Symbol	Description		Reset
52h	EPaConfig_0	R / W	7: INxOUT	0: OUT	1: IN	00h
		R / W	6: IntEP_Mode	0: Normal Toggle (IN) 0: Bulk OUT (OUT)	1: Always Toggle (IN) 1: Interrupt OUT (OUT)	
		R / W	5: EnEndpoint	0: Disable Endpoint	1: Enable Endpoint	
			4:	0:	1:	
		R / W	3: EndpointNumber [3]	Endpoint Number		
			2: EndpointNumber [2]			
			1: EndpointNumber [1]			
			0: EndpointNumber [0]			

This register sets the endpoint EPa.

Make sure the combination of EndpointNumber and INxOUT set for this endpoint does not duplicate those for other endpoints.

Bit7 INxOUT

This bit sets the direction of transfer at the endpoint.

Bit6 IntEP_Mode

This bit sets interrupt transfer mode.

Do not set this bit to 1 for a Bulk endpoint.

The setting of this bit differs depending on the endpoint direction (IN or OUT). (The endpoint direction is set by INxOUT in bit 7.)

For the IN direction (INxOUT = 1), set the operation mode of the toggle sequence bit. The toggle sequence operation mode depends on the application. Select either operation mode for an Interrupt IN endpoint.

0: Normal toggle — Performs a normal toggle sequence

1: Always toggle — Always toggles for each transaction performed. For details on this mode, refer to Section 5.7.5 in the USB2.0 Standard.

For the OUT direction (INxOUT = 0), set whether to perform PING flow control at this endpoint. For an Interrupt OUT endpoint, set this bit to 1.

0: Bulk OUT — Make this setting for a Bulk OUT endpoint.

1: Interrupt OUT — Make this setting for an Interrupt OUT endpoint.

Bit5 EnEndpoint

Setting this bit to 1 enables the endpoint EPa.

When this bit is 0, accesses to the endpoint are ignored.

This bit must be set in accordance with the SetConfiguration request from the host.

Bit4 Reserved

Bit3-0 EndpointNumber

Set any endpoint number in the range of 0x1 to 0xF.

7.2.67 53h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
53h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.68 54h EPaControl (EPa Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
54h	EPaControl	R / W	7: AutoForceNAK	0: Do nothing	1: Auto Force NAK	00h
		R / W	6: EnShortPkt	0: Do nothing	1: Enable Short Packet	
		R / W	5: DisAF_NAK_Short	0: Auto Force NAK Short	1 Disable Auto Force	
		R	4: ToggleStat	Toggle sequence bit		
		W	3: ToggleSet	0: Do nothing	1: Set Toggle sequence bit	
		W	2: ToggleClr	0: Do nothing	1: Clear Toggle sequence bit	
		R / W	1: ForceNAK	0: Do nothing	1: Force NAK	
		R / W	0: ForceSTALL	0: Do nothing	1: Force STALL	

This register sets the operations of the endpoint EPa.

Bit7 AutoForceNAK

When a transaction at the endpoint EPa finishes normally, the ForceNAK bit in this register is set to 1.

Bit6 EnShortPkt

Setting this bit to 1 allows the data in the FIFO that is smaller than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPa. When an IN transaction in which a short packet has been transmitted finishes, this bit is automatically cleared to 0. If a packet equal to MaxPacketSize has been transmitted, this bit is not cleared.

If this bit is set to 1 while there is no data in the FIFO, a packet with a length of zero can be transmitted in response to an IN token from the host. If data is written to the FIFO while a packet in it is being transmitted after this bit was set, the written data may also be transmitted, depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission finishes and this bit is cleared.

Bit5 DisAF_NAK_Short

This bit enable/disables the Auto Force NAK Short function (hereinafter simply referred to as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that finished normally is a short packet.

By default, the AF_NAK_Short function is enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit has been set when this bit is set, the AutoForceNAK bit has priority.

Bit4 ToggleStat

This bit shows the status of the toggle sequence bit for the endpoint EPa.

Bit3 ToggleSet

This bit sets the toggle sequence bit for the endpoint EPa to 1. If this bit is set at the same time the ToggleClr bit is set, the function of the ToggleClr bit has priority.

Bit2 ToggleClr

This bit clears the toggle sequence bit for the endpoint EPa to 0. If this bit is set at the same time the ToggleSet bit is set, the function of this bit has priority.

Bit1 ForceNAK

Setting this bit to 1 returns NAK for a transaction at the endpoint EPa, irrespective of the number of data bytes or the amount of free space in the FIFO.

If a transaction is already being executed when the software attempts to set this bit to 1, the bit is not set until the transaction is complete, at which time it is set to 1. If no transactions are being executed, the bit is set to 1 immediately.

Bit0 ForceSTALL

Setting this bit to 1 returns STALL for a transaction at the endpoint EPa. This bit has priority over the ForceNAK bit.

If any transaction is currently being executed and this bit is set a certain length of time after the transaction has started, the setting of this bit takes effect beginning with the next transaction.

7.2.69 55h EPaJoin (End Point a Join)

Address	Register Name	R / W	Bit Symbol	Description		Reset
55h	EPaJoin	R / W	7: JoinIDE	0: Do nothing	1: Join EPa to IDE	00h
		R / W	6: JoinFIFO_Stat	0: Do nothing	1 Join EPa to show Status	
		R / W	5: JoinDMA0_Rd	0: Do nothing	1: Join EPa to DMA0_Rd	
		R / W	4: JoinDMA0_Wr	0: Do nothing	1: Join EPa to DMA0_Wr	
		R / W	3: JoinDMA1_Rd	0: Do nothing	1: Join EPa to DMA1_Rd	
		R / W	2: JoinDMA1_Wr	0: Do nothing	1: Join EPa to DMA1_Wr	
		R / W	1: JoinCPU_Rd	0: Do nothing	1: Join EPa to CPU_Rd	
		R / W	0: JoinCPU_Wr	0: Do nothing	1: Join EPa to CPU_Wr	

This register specifies the port with which a data transfer to or from the endpoint EPa is to be performed.

Bit7 JoinIDE

Performs a data transfer with the FIFO for the endpoint EPa.

When this bit is set, the transfer direction of the IDE is set for that of EPa. Specifically, if the endpoint EPa is directed OUT, the IDE performs a write transfer. If the endpoint EPa is directed IN, the IDE performs a read transfer.

Bit6 JoinFIFO_Stat

Allows the full/empty status of the FIFO for the endpoint EPa to be monitored by means of FIFO_IntStat.FIFO_Full and FIFO_IntStat.FIFO_Empty.

Bit5 JoinDMA0_Rd

Performs a read transfer on DMA0 with the FIFO for the endpoint EPa. Specifically, when a read on DMA0 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit4 JoinDMA0_Wr

Performs a write transfer on DMA0 with the FIFO for the endpoint EPa. Specifically, when a write on DMA0 is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit3 JoinDMA1_Rd

Performs a read transfer on DMA1 with the FIFO for the endpoint EPa. Specifically, when a read on DMA1 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit2 JoinDMA1_Wr

Performs a write transfer on DMA1 with the FIFO for the endpoint EPa. Specifically, when a write on DMA1 is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit1 JoinCPU_Rd

Performs a read transfer for the CPU register access with the FIFO for the endpoint EPa. Specifically, when reading of the EPnFIFO_Rd_H,L registers or EPnFIFO_ByteRd register is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit0 JoinCPU_Wr

Performs a write transfer for the CPU register access with the FIFO for the endpoint EPa. Specifically, when a write to the EPnFIFO_Wr_H,L registers is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

When the JoinDMAx{x=0,1}_Rd or JoinDMAx{x=0,1}_Wr bit has been set, it is possible to determine the number of the remaining bytes of data in the FIFO if the endpoint is directed OUT, or the amount of free space if the endpoint is directed IN, by inspecting the DMAx{x=0,1}_Remain_H,L registers.

When the JoinCPU_Rd or JoinCPU_Wr bit has been set, it is possible to write or read data to or from the EPnFIFO_Rd_H,L, EPnFIFO_ByteRd, or EPnFIFO_Wr_H,L registers after inspecting the EPnRdRemain_H,L or EPnWrRemain_H,L registers.

Bits 5 to 0 in this register can be set to 1 only one bit at a time. If the software writes 1 to multiple bits at the same time, the most significant bit is set.

7.2.70 56h - 57h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
56h -57h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.71 58h EPbMaxSize_H (EPb Max Packet Size High)

7.2.72 59h EPbMaxSize_L (EPb Max Packet Size Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
58h	EPbMaxSize_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
		R / W	1: MaxSize [9]			
			0: MaxSize [8]			

Address	Register Name	R / W	Bit Symbol	Description		Reset
59h	EPbMaxSize_L	R / W	7: MaxSize [7]	Endpoint [b] Max Packet Size		00h
			6: MaxSize [6]			
			5: MaxSize [5]			
			4: MaxSize [4]			
			3: MaxSize [3]			
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

These registers set MaxPacketSize.

58h.Bit7-2 Reserved

58h.Bit1-0, 59h.Bit7-3 EPbMaxSize [9:3]

These bits set MaxPacketSize for the endpoint EPb. To use this endpoint for bulk transfers, set MaxSize to any of the following:

During FS 8, 16, 32, or 64 bytes

During HS 512 bytes

To use this endpoint for interrupt transfers, set MaxSize to any desired size within the following limits:

During FS Up to 64 bytes

During HS Up to 512 bytes

59h.Bit2-0 Reserved

7.2.73 5Ah EPbConfig_0 (EPb Configuration 0)

Address	Register Name	R / W	Bit Symbol	Description		Reset
5Ah	EPbConfig_0	R / W	7: INxOUT	0: OUT	1: IN	00h
		R / W	6: IntEP_Mode	0: Normal Toggle (IN) 0: Bulk OUT (OUT)	1: Always Toggle (IN) 1: Interrupt OUT (OUT)	
		R / W	5: EnEndpoint	0: Disable Endpoint	1: Enable Endpoint	
			4:	0:	1:	
		R / W	3: EndpointNumber [3]	Endpoint Number		
			2: EndpointNumber [2]			
			1: EndpointNumber [1]			
			0: EndpointNumber [0]			

This register sets the endpoint EPb.

Make sure the combination of EndpointNumber and INxOUT set for this endpoint does not duplicate that for other endpoints.

Bit7 INxOUT

This bit sets the direction of transfer at the endpoint.

Bit6 IntEP_Mode

This bit sets the interrupt transfer mode.0

Do not set this bit to 1 for a Bulk endpoint.

The setting of this bit differs depending on the endpoint direction (IN or OUT). (The endpoint direction is set by INxOUT in bit 7.)

For the IN direction (INxOUT = 1), set the operation mode of the toggle sequence bit. The toggle sequence operation mode depends on the application. Select either operation mode for an Interrupt IN endpoint.

0: Normal toggle — Performs a normal toggle sequence.

1: Always toggle — Always toggles for each transaction performed. For details on this mode, refer to Section 5.7.5 in the USB2.0 Standard.

For the OUT direction (INxOUT = 0), set whether to perform PING flow control at this endpoint. For an Interrupt OUT endpoint, set this bit to 1.

0: Bulk OUT — Make this setting for a Bulk OUT endpoint.

1: Interrupt OUT — Make this setting for an Interrupt OUT endpoint.

Bit5 EnEndpoint

Setting this bit to 1 enables the endpoint EPb.

When this bit is 0, accesses to the endpoint are ignored.

This bit must be set in accordance with the SetConfiguration request from the host.

Bit4 Reserved

Bit3-0 EndpointNumber

Set any endpoint number in the range of 0x1 to 0xF.

7.2.74 5Bh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
5Bh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.75 5Ch EPbControl (EPb Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
5Ch	EPbControl	R / W	7: AutoForceNAK	0: Do nothing	1: Auto Force NAK	00h
		R / W	6: EnShortPkt	0: Do nothing	1: Enable Short Packet	
		R / W	5: DisAF_NAK_Short	0: Auto Force NAK Short	1 Disable Auto Force	
		R	4: ToggleStat	Toggle sequence bit		
		W	3: ToggleSet	0: Do nothing	1: Set Toggle sequence bit	
		W	2: ToggleClr	0: Do nothing	1: Clear Toggle sequence bit	
		R / W	1: ForceNAK	0: Do nothing	1: Force NAK	
		R / W	0: ForceSTALL	0: Do nothing	1: Force STALL	

This register sets the operations of the endpoint EPb.

Bit7 AutoForceNAK

When a transaction at the endpoint EPb finishes normally, the ForceNAK bit in this register is set to 1.

Bit6 EnShortPkt

Setting this bit to 1 allows the data in the FIFO that is smaller than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPb. When the IN transaction in which a short packet has been transmitted finishes, this bit is automatically cleared to 0. If a packet equal to MaxPacketSize has been transmitted, this bit is not cleared.

If this bit is set to 1 while no data exists in the FIFO, a packet with a length of zero can be transmitted in response to an IN token from the host. If data is written to the FIFO while a packet in it is being transmitted after this bit was set, the written data may also be transmitted, depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission finishes and this bit is cleared.

Bit5 DisAF_NAK_Short

This bit enable/disables the Auto Force NAK Short function (hereinafter referred to simply as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that finished normally is a short packet.

By default, the AF_NAK_Short function is enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit has been set when this bit is set, the AutoForceNAK bit has priority.

Bit4 ToggleStat

This bit shows the status of the toggle sequence bit for the endpoint EPb.

Bit3 ToggleSet

This bit sets the toggle sequence bit for the endpoint EPb to 1. If this bit is set at the same time as the ToggleClr bit, the function of the ToggleClr bit has priority.

Bit2 ToggleClr

This bit clears the toggle sequence bit for the endpoint EPb to 0. If this bit is set at the same time as the ToggleSet bit, the function of this bit has priority.

Bit1 ForceNAK

Setting this bit to 1 returns NAK for a transaction at the endpoint EPb, irrespective of the number of data bytes or the amount of free space in the FIFO.

If a transaction is already being executed when the software attempts to set this bit to 1, the bit is not set until the transaction finishes, at which time it is set to 1. If no transactions are being executed, the bit is set to 1 immediately.

Bit0 ForceSTALL

Setting this bit to 1 returns STALL for a transaction at the endpoint EPb. This bit has priority over the ForceNAK bit.

If any transaction is currently underway and this bit is set a certain length of time after the transaction has started, the setting of this bit takes effect beginning with the next transaction.

7.2.76 5Dh EPbJoin (End Point b Join)

Address	Register Name	R / W	Bit Symbol	Description		Reset
5Dh	EPbJoin	R / W	7: JoinIDE	0: Do nothing	1: Join EPb to IDE	00h
		R / W	6: JoinFIFO_Stat	0: Do nothing	1: Join EPb to show Status	
		R / W	5: JoinDMA0_Rd	0: Do nothing	1: Join EPb to DMA0_Rd	
		R / W	4: JoinDMA0_Wr	0: Do nothing	1: Join EPb to DMA0_Wr	
		R / W	3: JoinDMA1_Rd	0: Do nothing	1: Join EPb to DMA1_Rd	
		R / W	2: JoinDMA1_Wr	0: Do nothing	1: Join EPb to DMA1_Wr	
		R / W	1: JoinCPU_Rd	0: Do nothing	1: Join EPb to CPU_Rd	
		R / W	0: JoinCPU_Wr	0: Do nothing	1: Join EPb to CPU_Wr	

This register specifies the port with which a data transfer to or from the endpoint EPb is to be performed.

Bit7 JoinIDE

Performs a data transfer with the FIFO for the endpoint EPb.

When this bit is set, the transfer direction of the IDE is set for that of EPb. Specifically, if the endpoint EPb is directed OUT, the IDE performs a write transfer. If the endpoint EPb is directed IN, the IDE performs a read transfer.

Bit6 JoinFIFO_Stat

Allows the full/empty status of the FIFO for the endpoint EPb to be monitored by means of FIFO_IntStat.FIFO_Full and FIFO_IntStat.FIFO_Empty.

Bit5 JoinDMA0_Rd

Performs a read transfer on DMA0 with the FIFO for the endpoint EPb. Specifically, when a read on DMA0 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit4 JoinDMA0_Wr

Performs a write transfer on DMA0 with the FIFO for the endpoint EPb. Specifically, when a write on DMA0 is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit3 JoinDMA1_Rd

Performs a read transfer on DMA1 with the FIFO for the endpoint EPb. Specifically, when a read on DMA1 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit2 JoinDMA1_Wr

Performs a write transfer on DMA1 with the FIFO for the endpoint EPb. Specifically, when a write on DMA1 is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit1 JoinCPU_Rd

Performs a read transfer for the CPU register access with the FIFO for the endpoint EPb. Specifically, when a read of the EPnFIFO_Rd_H,L registers or EPnFIFO_ByteRd register is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit0 JoinCPU_Wr

Performs a write transfer for the CPU register access with the FIFO for the endpoint EPb. Specifically, when a write to the EPnFIFO_Wr_H,L registers is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

When the JoinDMAx{x=0,1}_Rd or JoinDMAx{x=0,1}_Wr bit has been set, it is possible to determine the number of the remaining bytes of data in the FIFO if the endpoint is directed OUT, or the amount of free space if the endpoint is directed IN, by inspecting the DMAx{x=0,1}_Remain_H,L registers.

When the JoinCPU_Rd or JoinCPU_Wr bit has been set, it is possible to write or read data to or from the EPnFIFO_Rd_H,L, EPnFIFO_ByteRd, or EPnFIFO_Wr_H,L registers after inspecting the EPnRdRemain_H,L or EPnWrRemain_H,L registers.

Bits 5 to 0 in this register can be set to 1 only one bit at a time. If the software writes 1 to multiple bits at the same time, the most significant bit is set.

7.2.77 5Eh - 5Fh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
5Eh -5Fh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.78 60h EPcMaxSize_H (EPc Max Packet Size High)

7.2.79 61h EPcMaxSize_L (EPc Max Packet Size Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
60h	EPcMaxSize_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
		R / W	1: MaxSize [9]			
			0: MaxSize [8]			

Address	Register Name	R / W	Bit Symbol	Description		Reset
61h	EPcMaxSize_L	R / W	7: MaxSize [7]	Endpoint [c] Max Packet Size		00h
			6: MaxSize [6]			
			5: MaxSize [5]			
			4: MaxSize [4]			
			3: MaxSize [3]			
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

These registers set MaxPacketSize.

60h.Bit7-2 Reserved

60h.Bit1-0, 61h.Bit7-3 EPcMaxSize [9:3]

These bits set MaxPacketSize for the endpoint EPc.

To use this endpoint for bulk transfers, set MaxSize to any of the following:

During FS 8, 16, 32, or 64 bytes

During HS 512 bytes

To use this endpoint for interrupt transfers, set MaxSize to any desired size within the following limits:

During FS Up to 64 bytes

During HS Up to 512 bytes

61h.Bit2-0 Reserved

7.2.80 62h EPcConfig_0 (EPc Configuration 0)

Address	Register Name	R / W	Bit Symbol	Description		Reset
62h	EPcConfig_0	R / W	7: INxOUT	0: OUT	1: IN	00h
		R / W	6: IntEP_Mode	0: Normal Toggle (IN) 0: Bulk OUT (OUT)	1: Always Toggle (IN) 1: Interrupt OUT (OUT)	
		R / W	5: EnEndpoint	0: Disable Endpoint	1: Enable Endpoint	
			4:	0:	1:	
		R / W	3: EndpointNumber [3]	Endpoint Number		
			2: EndpointNumber [2]			
			1: EndpointNumber [1]			
			0: EndpointNumber [0]			

This register sets the endpoint EPc.

Make sure the combination of EndpointNumber and INxOUT set for this endpoint does not duplicate that for other endpoints.

Bit7 INxOUT

This bit sets the direction of transfer at the endpoint.

Bit6 IntEP_Mode

This bit sets the interrupt transfer mode.

Do not set this bit to 1 for a Bulk endpoint.

The setting of this bit differs depending on the endpoint direction (IN or OUT). (The endpoint direction is set by INxOUT in bit 7.)

For the IN direction (INxOUT = 1), set the operation mode of the toggle sequence bit. The toggle sequence operation mode depends on the application. Select either operation mode for an Interrupt IN endpoint.

0: Normal toggle — Performs a normal toggle sequence

1: Always toggle — Always toggles for each transaction performed. For details on this mode, refer to Section 5.7.5 in the USB2.0 Standard.

For the OUT direction (INxOUT = 0), set whether to perform PING flow control at this endpoint. For an Interrupt OUT endpoint, set this bit to 1.

0: Bulk OUT — Make this setting for a Bulk OUT endpoint.

1: Interrupt OUT — Make this setting for an Interrupt OUT endpoint.

Bit5 EnEndpoint

Setting this bit to 1 enables the endpoint EPc.

When this bit is 0, accesses to the endpoint are ignored.

This bit must be set in accordance with the SetConfiguration request from the host.

Bit4 Reserved

Bit3-0 EndpointNumber

Set any endpoint number in the range of 0x1 to 0xF.

7.2.81 63h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
63h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.82 64h EPcControl (EPc Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
64h	EPcControl	R / W	7: AutoForceNAK	0: Do nothing	1: Auto Force NAK	00h
		R / W	6: EnShortPkt	0: Do nothing	1: Enable Short Packet	
		R / W	5: DisAF_NAK_Short	0: Auto Force NAK Short	1 Disable Auto Force	
		R	4: ToggleStat	Toggle sequence bit		
		W	3: ToggleSet	0: Do nothing	1: Set Toggle sequence bit	
		W	2: ToggleClr	0: Do nothing	1: Clear Toggle sequence bit	
		R / W	1: ForceNAK	0: Do nothing	1: Force NAK	
		R / W	0: ForceSTALL	0: Do nothing	1: Force STALL	

This register sets the operations of the endpoint EPc.

Bit7 AutoForceNAK

When a transaction at the endpoint EPc finishes normally, the ForceNAK bit in this register is set to 1.

Bit6 EnShortPkt

Setting this bit to 1 allows the data in the FIFO that is smaller than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPc. When an IN transaction in which a short packet has been transmitted finishes, this bit is automatically cleared to 0. If a packet equal to MaxPacketSize has been transmitted, this bit is not cleared.

If this bit is set to 1 while there is no data in the FIFO, a packet with a length of zero can be transmitted in response to an IN token from the host. If data is written to the FIFO while a packet in it is being transmitted after this bit was set, the written data may also be transmitted, depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission finishes and this bit is cleared.

Bit5 DisAF_NAK_Short

This bit enable/disables the Auto Force NAK Short function (hereinafter referred to simply as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that finished normally is a short packet.

By default, the AF_NAK_Short function is enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit has been set when this bit is set, the AutoForceNAK bit has priority.

Bit4 ToggleStat

This bit shows the status of the toggle sequence bit for the endpoint EPc.

Bit3 ToggleSet

This bit sets the toggle sequence bit for the endpoint EPc to 1. If this bit is set at the same time as the ToggleClr bit, the function of the ToggleClr bit has priority.

Bit2 ToggleClr

This bit clears the toggle sequence bit for the endpoint EPc to 0. If this bit is set at the same time the ToggleSet bit is set, the function of this bit has priority.

Bit1 ForceNAK

Setting this bit to 1 returns NAK for a transaction at the endpoint EPc, irrespective of the number of data bytes or the amount of free space in the FIFO.

If a transaction is already being executed when the software attempts to set this bit to 1, the bit is not set until the transaction finishes, at which time it is set to 1. If no transactions are being executed, the bit is set to 1 immediately.

Bit0 ForceSTALL

Setting this bit to 1 returns STALL for a transaction at the endpoint EPc. This bit has priority over the ForceNAK bit.

If any transaction is currently underway and this bit is set a certain length of time after the transaction has started, the setting of this bit takes effect beginning with the next transaction.

7.2.83 65h EPcJoin (End Point c Join)

Address	Register Name	R / W	Bit Symbol	Description		Reset
65h	EPcJoin	R / W	7: JoinIDE	0: Do nothing	1: Join EPc to IDE	00h
		R / W	6: JoinFIFO_Status	0: Do nothing	1 Join EPc to show Status	
		R / W	5: JoinDMA0_Rd	0: Do nothing	1: Join EPc to DMA0_Rd	
		R / W	4: JoinDMA0_Wr	0: Do nothing	1: Join EPc to DMA0_Wr	
		R / W	3: JoinDMA1_Rd	0: Do nothing	1: Join EPc to DMA1_Rd	
		R / W	2: JoinDMA1_Wr	0: Do nothing	1: Join EPc to DMA1_Wr	
		R / W	1: JoinCPU_Rd	0: Do nothing	1: Join EPc to CPU_Rd	
		R / W	0: JoinCPU_Wr	0: Do nothing	1: Join EPc to CPU_Wr	

This register specifies the port with which a data transfer to or from the endpoint EPc is to be performed.

Bit7 JoinIDE

Performs a data transfer with the FIFO for the endpoint EPc.

When this bit is set, the transfer direction of the IDE is set for that of EPc. Specifically, if the endpoint EPc is directed OUT, the IDE performs a write transfer. If the endpoint EPc is directed IN, the IDE performs a read transfer.

Bit6 JoinFIFO_Stat

Allows the full/empty status of the FIFO for the endpoint EPc to be monitored by means of FIFO_IntStat.FIFO_Full and FIFO_IntStat.FIFO_Empty

Bit5 JoinDMA0_Rd

Performs a read transfer on DMA0 with the FIFO for the endpoint EPc. Specifically, when a read on DMA0 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit4 JoinDMA0_Wr

Performs a write transfer on DMA0 with the FIFO for the endpoint EPc. Specifically, when a write on DMA0 is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit3 JoinDMA1_Rd

Performs a read transfer on DMA1 with the FIFO for the endpoint EPc. Specifically, when a read on DMA1 is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit2 JoinDMA1_Wr

Performs a write transfer on DMA1 with the FIFO for the endpoint EPc. Specifically, when a write on DMA1 is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

Bit1 JoinCPU_Rd

Performs a read transfer for the CPU register access with the FIFO for the endpoint EPc. Specifically, when reading of the EPnFIFO_Rd_H,L registers or EPnFIFO_ByteRd register is performed, data is read from this endpoint FIFO. This read transfer can be performed without concern for the direction in which the endpoint is set.

Bit0 JoinCPU_Wr

Performs a write transfer for the CPU register access with the FIFO for the endpoint EPc. Specifically, when writing to the EPnFIFO_Wr_H,L registers is performed, data is written to this endpoint FIFO. This write transfer can be performed without concern for the direction in which the endpoint is set.

When the JoinDMAx{x=0,1}_Rd or JoinDMAx{x=0,1}_Wr bit has been set, it is possible to determine the number of the remaining bytes of data in the FIFO if the endpoint is directed OUT, or the amount of free space if the endpoint is directed IN, by inspecting the DMAx{x=0,1}_Remain_H,L registers.

When the JoinCPU_Rd or JoinCPU_Wr bit has been set, it is possible to write or read data to or from the EPnFIFO_Rd_H,L, EPnFIFO_ByteRd, or EPnFIFO_Wr_H,L registers after inspecting the EPnRdRemain_H,L or EPnWrRemain_H,L registers.

Bits 5 to 0 in this register can be set to 1 only one bit at a time. If the software writes 1 to multiple bits at the same time, the most significant bit is set.

7.2.84 66h - 6Bh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
66h -6Bh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.85 6Ch RAM_WrAdrs_H (RAM Write Address High)

7.2.86 6Dh RAM_WrAdrs_L (RAM Write Address Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
6Ch	RAM_WrAdrs_H		7:		00h
			6:		
			5:		
			4:		
		R / W	3: RAM_WrAdrs [11]	RAM Write Address	
			2: RAM_WrAdrs [10]		
			1: RAM_WrAdrs [9]		
			0: RAM_WrAdrs [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
6Dh	RAM_WrAdrs_L	R / W	7: RAM_WrAdrs [7]	RAM Write Address	00h
			6: RAM_WrAdrs [6]		
			5: RAM_WrAdrs [5]		
			4: RAM_WrAdrs [4]		
			3: RAM_WrAdrs [3]		
			2: RAM_WrAdrs [2]		
			1: RAM_WrAdrs [1]		
			0: RAM_WrAdrs [0]		

These registers specify a RAM address when data is written to the RAM via the RAM_WrDoorH,L registers.

6Ch.Bit7-4 Reserved

6Ch.Bit3-0, 6Dh.Bit7-0 RAM_WrAdrs[11:0]

These bits specify a RAM address when data is written to the RAM. The address is incremented according to the number of bytes written to the RAM_WrDoorH,L registers. As exact RAM_WrAdrs cannot be known immediately after a write to the RAM_WrDoorH,L registers, insert an interval of at least 1 CPU cycle before confirming RAM_WrAdrs. For details on writing data, refer to the section on the RAM_WrDoor_H,L registers.

To read RAM_WrAdrs for confirmation, access the registers in order of RAM_WrAdrs_H and RAM_WrAdrs_L.

7.2.87 6Eh RAM_WrDoor_H (RAM Write Door High)

7.2.88 6Fh RAM_WrDoor_L (RAM Write Door Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
06Eh	RAM_WrDoor_H	W	7: RAM_WrDoor [15]	RAM Write Door	XXh
			6: RAM_WrDoor [14]		
			5: RAM_WrDoor [13]		
			4: RAM_WrDoor [12]		
			3: RAM_WrDoor [11]		
			2: RAM_WrDoor [10]		
			1: RAM_WrDoor [9]		
			0: RAM_WrDoor [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
06Fh	RAM_WrDoor_L	W	7: RAM_WrDoor [7]	RAM Write Door	XXh
			6: RAM_WrDoor [6]		
			5: RAM_WrDoor [5]		
			4: RAM_WrDoor [4]		
			3: RAM_WrDoor [3]		
			2: RAM_WrDoor [2]		
			1: RAM_WrDoor [1]		
			0: RAM_WrDoor [0]		

6Eh.Bit7-0, 6Fh.Bit7-0 RAM_WrDoor [15:0]

These registers are write-only registers, which are used to write the data to be written to the RAM.

Before writing data to these registers, set the start address of the RAM to be written to in the RAM_WrAdrs_H,L registers. Then, when data is written to the RAM_WrDoor_H,L registers, RAM_WrAdrs_H,L is automatically incremented according to the number of written bytes while the data is written to the RAM.

The RAM_WrDoor_H,L registers may be used to write data for the descriptor area. The data written from these registers can be used any number of times by the ReplyDescriptor function. That is to say, this data is neither erased nor overwritten by the descriptor reply function. However, if the area to which descriptor data has been written overlaps any area reserved for another endpoint, the data in it may be overwritten.

7.2.89 70h EPnFIFO_Rd_H (EPn FIFO Read High)

7.2.90 71h EPnFIFO_Rd_L (EPn FIFO Read Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
70h	EPnFIFO_Rd_H	R	7: EPnFIFO_Rd [15]	Endpoint n FIFO Read	XXh
			6: EPnFIFO_Rd [14]		
			5: EPnFIFO_Rd [13]		
			4: EPnFIFO_Rd [12]		
			3: EPnFIFO_Rd [11]		
			2: EPnFIFO_Rd [10]		
			1: EPnFIFO_Rd [9]		
			0: EPnFIFO_Rd [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
71h	EPnFIFO_Rd_L	R	7: EPnFIFO_Rd [7]	Endpoint n FIFO Read	XXh
			6: EPnFIFO_Rd [6]		
			5: EPnFIFO_Rd [5]		
			4: EPnFIFO_Rd [4]		
			3: EPnFIFO_Rd [3]		
			2: EPnFIFO_Rd [2]		
			1: EPnFIFO_Rd [1]		
			0: EPnFIFO_Rd [0]		

70h.Bit7-0, 71h.Bit7-0 EPnFIFO_Rd [15:0]

These registers allow the data for the endpoint FIFO that has had the EPx{x=0,a-c}Join.JoinCPU_Rd bit set to be read out.

In 8-bit mode, the FIFO data can be read out by accessing either register, EPnFIFO_Rd_H or L.

In 16-bit mode, if these registers are accessed for reading while a byte boundary exists in the FIFO, valid data will be output to only one side of them. For details, refer to Section 6.4.3.1.5, “Processing Odd Bytes in FIFO Access.”

To read FIFO data using these registers, always be sure to read the EPnRdRemain_H,L registers first in order to confirm the number of bytes that can be read.

7.2.91 72h EPnFIFO_Wr_H(EPn FIFO Write High)

7.2.92 73h EPnFIFO_Wr_L(EPn FIFO Write Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
72h	EPnFIFO_Wr_H	W	7: EPnFIFO_Wr [15]	Endpoint n FIFO Write	XXh
			6: EPnFIFO_Wr [14]		
			5: EPnFIFO_Wr [13]		
			4: EPnFIFO_Wr [12]		
			3: EPnFIFO_Wr [11]		
			2: EPnFIFO_Wr [10]		
			1: EPnFIFO_Wr [9]		
			0: EPnFIFO_Wr [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
73h	EPnFIFO_Wr_L	W	7: EPnFIFO_Wr [7]	Endpoint n FIFO Write	XXh
			6: EPnFIFO_Wr [6]		
			5: EPnFIFO_Wr [5]		
			4: EPnFIFO_Wr [4]		
			3: EPnFIFO_Wr [3]		
			2: EPnFIFO_Wr [2]		
			1: EPnFIFO_Wr [1]		
			0: EPnFIFO_Wr [0]		

72h.Bit7-0, 73h.Bit7-0 EPnFIFO_Wr [15:0]

These registers allow data to be written to the endpoint FIFO that has had the EPx{x=0,a-c}Join.JoinCPU_Wr bit set.

In 8-bit mode, the data can be written to the FIFO by accessing either register, EPnFIFO_Wr_H or L.

In 16-bit mode, if these registers are accessed for writing while a byte boundary exists in the FIFO, the data will be written to only one side of them. For details, refer to Section 6.4.3.1.5, “Processing Odd Bytes in FIFO Access.”

To write data into the FIFO using these registers, always be sure to read the EPnWrRemain_H,L registers first to confirm the number of bytes that can be written to.

7.2.93 74h EPnRdRemain_H (EPn FIFO Read Remain High)

7.2.94 75h EPnRdRemain_L (EPn FIFO Read Remain Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
74h	EPnRdRemain_H	R	7: RdRemainValid	0:None	1: Read Remain Valid	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R	3: EPnRdRemain [11]	Endpoint n FIFO Read Remain High		
			2: EPnRdRemain [10]			
			1: EPnRdRemain [9]			
			0: EPnRdRemain [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
75h	EPnRdRemain_L	R	7: EPnRdRemain [7]	Endpoint n FIFO Read Remain Low	00h
			6: EPnRdRemain [6]		
			5: EPnRdRemain [5]		
			4: EPnRdRemain [4]		
			3: EPnRdRemain [3]		
			2: EPnRdRemain [2]		
			1: EPnRdRemain [1]		
			0: EPnRdRemain [0]		

74h.Bit7 RdRemainValid

This bit is set to 1 when an endpoint has been joined to the CPU I/F by the EPx{x=0,a-c}Join. If this bit = 0, the value of EPnRdRemain is invalid.

74h.Bit6-4 Reserved

74h.Bit3-0, 75h.Bit7-0 EPnRdRemain [11:0]

These bits indicate the number of readable data bytes in the FIFO for an endpoint that has been connected to the CPU I/F by the EPx{x=0,a-c}Join.JoinCPU_Rd bit. To determine the number of readable data bytes in the FIFO, it is necessary to access the EPnRdRemain_H and EPnRdRemain_L registers in pairs. In such a case, access the EPnRdRemain_H register first.

7.2.95 76h EPnWrRemain_H (EPn FIFO Write Remain High)

7.2.96 77h EPnWrRemain_L (EPn FIFO Write Remain Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
76h	EPnWrRemain_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R	3: EPnWrRemain [11]	Endpoint n FIFO Write Remain High		
			2: EPnWrRemain [10]			
			1: EPnWrRemain [9]			
			0: EPnWrRemain [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
77h	EPnWrRemain_L	R	7: EPnWrRemain [7]	Endpoint n FIFO Write Remain Low	00h
			6: EPnWrRemain [6]		
			5: EPnWrRemain [5]		
			4: EPnWrRemain [4]		
			3: EPnWrRemain [3]		
			2: EPnWrRemain [2]		
			1: EPnWrRemain [1]		
			0: EPnWrRemain [0]		

77h.Bit7-4 Reserved

77h.Bit3-0, 78h.Bit7-0 EPnWrRemain [11:0]

These bits indicate the amount of free space in the FIFO for an endpoint that has been connected to the CPU I/F by the EPx{x=0,a-c}Join.JoinCPU_Wr bit. The exact amount of free space in the FIFO cannot be known immediately after a write to the FIFO. Insert an interval of at least 1 CPU cycle before confirming the amount of free space in the FIFO. To determine the amount of free space in the FIFO, it is necessary to access the EPnWrRemain_H and EPnWrRemain_L registers in pairs. In such a case, access the EPnWrRemain_H register first.

7.2.97 78h DescAdrs_H (Descriptor Address High)

7.2.98 79h DescAdrs_L (Descriptor Address Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
78h	DescAdrs_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R / W	3: DescAdrs [11]	Descriptor Address		
			2: DescAdrs [10]			
			1: DescAdrs [9]			
			0: DescAdrs [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
79h	DescAdrs_L	R / W	7: DescAdrs [7]	Descriptor Address	00h
			6: DescAdrs [6]		
			5: DescAdrs [5]		
			4: DescAdrs [4]		
			3: DescAdrs [3]		
			2: DescAdrs [2]		
			1: DescAdrs [1]		
			0: DescAdrs [0]		

These registers specify the Descriptor Address.

78h.Bit7-4 Reserved

78h.Bit4-0, 79h.Bit7-0 DescAdrs [11:0]

These bits specify the address of the FIFO from which a descriptor reply operation of the Descriptor Reply function is to start.

The Descriptor Address is not intended to allocate a FIFO area for the Descriptor Reply function. Regardless of how the FIFO areas have been set, any address in the entire FIFO area from 0x000 to 0x9FF (2.5 Kbytes) can be specified for the Descriptor Address.

During descriptor reply, DescAdrs is updated by an amount equal to the transmitted data bytes each time an IN transaction at the endpoint EP0 finishes. For details on the Descriptor Reply function, refer to the section on ReplyDescriptor of the EP0Control register.

As the FIFO area used for the Descriptor Reply function is not explicitly allocated, make sure this FIFO does not overlap those for other endpoints by setting up the DescAdrs_H,L and the DescSize_H,L registers properly. Any location between the end address of the reserved area for the endpoint EP0 (0x040) and the start address of the CBW area (0x190) will be appropriate.

To inspect the Descriptor Address, read DescAdrs_H and DescAdrs_L, in that order.

7.2.99 7Ah DescSize_H (Descriptor Size High)

7.2.100 7Bh DescSize_L (Descriptor Size Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
7Ah	DescSize_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
		R / W	1: DescSize [9]	Descriptor Size		
			0: DescSize [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
7Bh	DescSize_L	R / W	7: DescSize [7]	Descriptor Size	00h
			6: DescSize [6]		
			5: DescSize [5]		
			4: DescSize [4]		
			3: DescSize [3]		
			2: DescSize [2]		
			1: DescSize [1]		
			0: DescSize [0]		

These registers specify the Descriptor Size.

7Ah.Bit7-2 Reserved

7Ah.Bit1-0, 7Bh.Bit7-0 DescSize [9:0]

For the Descriptor Size, specify the total number of data bytes to be sent back to the host in the Descriptor Reply function. For details on the Descriptor Reply function, refer to the section on ReplyDescriptor bits of the EP0Control register.

Regardless of how the FIFO areas have been set, any size from 0x000 to 0x3FF can be specified for the Descriptor Size. During descriptor reply, DescSize is updated by an amount equal to the transmitted data bytes each time an IN transaction at the endpoint EP0 finishes.

As the FIFO area used for the Descriptor Reply function is not explicitly allocated, make sure this FIFO does not overlap those for other endpoints by setting up the DescAdrs_H,L and the DescSize_H,L registers properly. Any location between the end address of the reserved area for the endpoint EP0 (0x040) and the start address of the CBW area (0x190) will be appropriate.

To inspect the Descriptor Size, read DescSize_H and DescSize_L, in that order.

7.2.101 7Ch EPnFIFO_ByteRd(EPn FIFO Byte Read)

Address	Register Name	R / W	Bit Symbol	Description	Reset
7Ch	EPnFIFO_ByteRd	R	7: EPnFIFO_ByteRd [7]	Endpoint n FIFO Byte Read	XXh
			6: EPnFIFO_ByteRd [6]		
			5: EPnFIFO_ByteRd [5]		
			4: EPnFIFO_ByteRd [4]		
			3: EPnFIFO_ByteRd [3]		
			2: EPnFIFO_ByteRd [2]		
			1: EPnFIFO_ByteRd [1]		
			0: EPnFIFO_ByteRd [0]		

Bit7-0 EPnFIFO_ByteRd [7:0]

These bits allow the data for the endpoint FIFO, which has had the EPx{x=0,a-c}Join.JoinCPU_Rd bit set, to be read in byte units.

When reading FIFO data using this register, always be sure to read the EPnRdRemain_H,L registers first to confirm the number of bytes that can be read .

7.2.102 7Dh - 7Fh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
7Dh -7Fh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.103 80h DMA0_FIFO_Control (DMA0 FIFO Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
80h	DMA0_FIFO_Control	R	7: FIFO_Running	0: FIFO is not running	1: FIFO is running	00h
		R / W	6: AutoEnShort	0: Do nothing	1: Auto Enable Short Packet	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

This register shows and sets the FIFO status during DMA0 transfer.

Bit7 FIFO_Running

This bit indicates that the FIFO for the endpoint connected to DMA0 is currently operating. This bit is set to 1 when DMA0 is activated, and is cleared to 0 when DMA0 has finished and the FIFO is emptied.

Bit6 AutoEnShort

If, while this bit is set, some bytes of data less than the max. packet size remain in the FIFO for any endpoint when DMA0 has finished, the EnShortPkt bit for that endpoint is set to 1.

This bit is effective when the endpoint connected to DMA0 is directed IN.

Bit5-0 Reserved

7.2.104 81h DMA0_Config (DMA0 Config)

Address	Register Name	R / W	Bit Symbol	Description		Reset
81h	DMA0_Config	R / W	7: FreeRun	0: Count mode	1: FreeRun mode	00h
		R / W	6: DMA_Mode	0: Normal mode	1: Address Decode mode	
			5:	0:	1:	
			4:	0:	1:	
		R / W	3: ActiveDMA	0: DMA0 Inactive	1: DMA0 Active	
			2:	0:	1:	
		R / W	1: ReqAssertCount [1]	Request Assert Count		
			0: ReqAssertCount [0]			

This register sets the operation mode of DMA0.

Bit7 FreeRun

This bit sets the operation mode of DMA0.

0: Count mode

1: Free-running mode

Bit6 DMA_Mode

This bit sets DMA0 mode.

0: The DMA operates in response to XDACK from the host, regarding it as an acknowledge.

1: The DMA operates in response to an access to the DMA0_RdData/DMA0_WrData register, regarding it as an acknowledge.

Bit5-4 Reserved

Bit3 ActiveDMA

This bit activates DMA0.

0: DMA0 disabled

1: DMA0 enabled

Bit2 Reserved

Bit1-0 ReqAssertCount [1:0]

These bits set the REQ Assert Count option provided to support burst read/burst writes by the CPU.

Set an assert count for XDREQ0 (number of transfer bytes). If the FIFO has writable free space or readable valid data greater than the set assert count, XDREQ0 will be asserted.

When DMA transfers for bytes equal to the set assert count have finished, XDREQ0 is temporarily negated; when the FIFO is confirmed to have writable free space or readable valid data greater than the set assert count again, XDREQ0 is reasserted.

As a result, when XDREQ0 is asserted once, transfers of bytes equal to the set assert count are guaranteed.

However, if DMA0 is set to count mode and the count of remaining bytes in DMA0_Count_HH,HL,LH,LL is smaller than the set assert count, the remaining count in DMA0_Count_HH,HL,LH,LL has priority; therefore, XDREQ0 is asserted when the FIFO has writable free space or readable valid data greater than the remaining count in DMA0_Count_HH,HL,LH,LL.

The table below shows the relationships among DMA0_Count_HH,HL,LH,LL (represented by Count in the table), ReqAssertCount (represented by Req in the table), and the free space/readable data in the FIFO (represented by Ready in the table), and the XDREQ0 signal and the number of transferable bytes.

The remaining count in DMA0_Count_HH,HL,LH,LL must be greater than or equal to 1 if the REQ Assert Count option is to function.

	Count>=Req		**Count<Req**	
	Ready>=Req	**Ready<Req**	**Ready>=Count**	**Ready<Count**
XDREQ0	Asserted	Negated	Asserted	Negated
Number of transferable bytes	Req	-	Req	-

ReqAssertCount [1:0]	**Mode**	
	16bit mode	**8bit mode**
0b00	Normal	Normal
0b01	16Byte(8Count)	16Byte(16Count)
0b10	32Byte(16Count)	32Byte(32Count)
0b11	64Byte(32Count)	64Byte(64Count)

When set to 00 (= Normal), the REQ Assert Count option is unused.

7.2.105 82h DMA0_Control (DMA0 Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
82h	DMA0_Control	R	7: DMA_Running	0: DMA is not running	1: DMA is running	00h
			6:	0:	1:	
			5:	0:	1:	
		W	4: CounterClr	0: Do nothing	1: Clear DMA counter	
			3:	0:	1:	
			2:	0:	1:	
		W	1: DMA_Stop	0: Do nothing	1: Finish DMA	
		W	0: DMA_Go	0: Do nothing	1: Start DMA	

This register controls and shows the status of DMA0.

Bit7 DMA_Running

This bit is set to 1 and remains set while a transfer on DMA0 is underway. While this bit remains set at 1, the EPx{x=0,a-c}Join.JoinDMA0_Rd,Wr bits cannot be rewritten.

Bit6-5 Reserved

Bit4 CounterClr

Setting this bit to 1 clears the DMA0_Count_HH,HL,LH,LL registers to 0. Writing to this bit is ignored while the DMA_Running bit = 1.

Bit3-2 Reserved

Bit1 DMA_Stop

Setting this bit to 1 stops the transfer on DMA0. When the transfer on DMA0 stops, the DMA_Running bit is cleared to 0. Furthermore, the DMA0_Cmp bit in the CPU_IntStat register is set to 1. To restart a transfer on DMA0, check the DMA_Running or the DMA0_Cmp bit in order to confirm the status, and wait until the DMA terminates.

Bit0 DMA_Go

Setting this bit to 1 starts a transfer on DMA0.

7.2.106 83h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
83h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.107 84h DMA0_Remain_H (DMA0 FIFO Remain High)

7.2.108 85h DMA0_Remain_L (DMA0 FIFO Remain Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
84h	DMA0_Remain_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R	3: DMA_Remain [11]	DMA FIFO Remain High		
			2: DMA_Remain [10]			
			1: DMA_Remain [9]			
			0: DMA_Remain [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
85h	DMA0_Remain_L	R	7: DMA_Remain [7]	DMA FIFO Remain Low	00h
			6: DMA_Remain [6]		
			5: DMA_Remain [5]		
			4: DMA_Remain [4]		
			3: DMA_Remain [3]		
			2: DMA_Remain [2]		
			1: DMA_Remain [1]		
			0: DMA_Remain [0]		

84h.Bit7-4 Reserved

84h.Bit3-0, 85h.Bit7-0 DMA_Remain [11:0]

These bits indicate the number of data bytes remaining in the FIFO for the endpoint connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA0_Rd bit, or the amount of free space remaining in the FIFO for the endpoint connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA0_Wr bit.

As the exact amount of free space in the FIFO cannot be determined via these registers immediately after a write to the DMA, insert an interval of at least 1 CPU cycle before checking the amount of free space in the FIFO.

To read these registers, access them in order of DMA0_Remain_H followed by L.

7.2.109 86h - 87h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
86h -87h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.110 88h DMA0_Count_HH (DMA0 Transfer Byte Counter High/High)

7.2.111 89h DMA0_Count_HL (DMA0 Transfer Byte Counter High/Low)

7.2.112 8Ah DMA0_Count_LH (DMA0 Transfer Byte Counter Low/High)

7.2.113 8Bh DMA0_Count_LL (DMA0 Transfer Byte Counter Low/Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
88h	DMA0_Count_HH	R / W	7: DMA_Count [31]	DMA Transfer Byte Counter	00h
			6: DMA_Count [30]		
			5: DMA_Count [29]		
			4: DMA_Count [28]		
			3: DMA_Count [27]		
			2: DMA_Count [26]		
			1: DMA_Count [25]		
			0: DMA_Count [24]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
89h	DMA0_Count_HL	R / W	7: DMA_Count [23]	DMA Transfer Byte Counter	00h
			6: DMA_Count [22]		
			5: DMA_Count [21]		
			4: DMA_Count [20]		
			3: DMA_Count [19]		
			2: DMA_Count [18]		
			1: DMA_Count [17]		
			0: DMA_Count [16]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
8Ah	DMA0_Count_LH	R / W	7: DMA_Count [15]	DMA Transfer Byte Counter	00h
			6: DMA_Count [14]		
			5: DMA_Count [13]		
			4: DMA_Count [12]		
			3: DMA_Count [11]		
			2: DMA_Count [10]		
			1: DMA_Count [9]		
			0: DMA_Count [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
8Bh	DMA0_Count_LL	R / W	7: DMA_Count [7]	DMA Transfer Byte Counter	00h
			6: DMA_Count [6]		
			5: DMA_Count [5]		
			4: DMA_Count [4]		
			3: DMA_Count [3]		
			2: DMA_Count [2]		
			1: DMA_Count [1]		
			0: DMA_Count [0]		

These counter registers are used to set the data length for a transfer on DMA0 in count mode, in byte units. The data length can be set for up to 0xFFFF_FFFF bytes. The counter

starts counting down from the set value. After setting transfer bytes in these registers, set the DMA0_Control.DMA_Go bit to 1 in order to start a DMA transfer. When transfers for the transfer bytes set in these registers have finished, the DMA transfer is terminated.

In free-running mode, the counter counts up from the set value. When the value in the DMA0_Count_HH,HL,LH,LL registers overflows, the DMA0_CountUp bit in the CPU_IntStat register is set to 1. The counter is recycled to 0 and restarts counting. In this mode, the number of bytes that has been transferred on DMA can be determined.

As an exact byte count cannot be obtained via these registers immediately after a write to the DMA, insert an interval of at least 1 CPU cycle before checking the byte count. To read these registers, access them in order of DMA0_Count_HH, HL, LH, followed by LL.

7.2.114 8Ch DMA0_RdData_H (DMA0 Read Data High)

7.2.115 8Dh DMA0_RdData_L (DMA0 Read Data Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
8Ch	DMA0_RdData_H	R	7: DMA_RdData [15]	DMA Read Data High	XXh
			6: DMA_RdData [14]		
			5: DMA_RdData [13]		
			4: DMA_RdData [12]		
			3: DMA_RdData [11]		
			2: DMA_RdData [10]		
			1: DMA_RdData [9]		
			0: DMA_RdData [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
8Dh	DMA0_RdData_L	R	7: DMA_RdData [7]	DMA Read Data Low	XXh
			6: DMA_RdData [6]		
			5: DMA_RdData [5]		
			4: DMA_RdData [4]		
			3: DMA_RdData [3]		
			2: DMA_RdData [2]		
			1: DMA_RdData [1]		
			0: DMA_RdData [0]		

8Ch.Bit7-0, 8Dh.Bit7-0 DMA_RdData [15:0]

When The DMA0_Config.DMA_Mode bit is set to 1, it is possible to read data from the FIFO connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA0_Rd bit by accessing these registers.

During operation in 8-bit mode, this DMA access can be accomplished by accessing either DMA0_RdData_H or DMA0_RdData_L.

7.2.116 8Eh DMA0_WrData_H (DMA0 Write Data High)

7.2.117 8Fh DMA0_WrData_L (DMA0 Write Data Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
8Eh	DMA0_WrData_H	W	7: DMA_WrData [15]	DMA Write Data High	XXh
			6: DMA_WrData [14]		
			5: DMA_WrData [13]		
			4: DMA_WrData [12]		
			3: DMA_WrData [11]		
			2: DMA_WrData [10]		
			1: DMA_WrData [9]		
			0: DMA_WrData [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
8Fh	DMA0_WrData_L	W	7: DMA_WrData [7]	DMA Write Data Low	XXh
			6: DMA_WrData [6]		
			5: DMA_WrData [5]		
			4: DMA_WrData [4]		
			3: DMA_WrData [3]		
			2: DMA_WrData [2]		
			1: DMA_WrData [1]		
			0: DMA_WrData [0]		

8Eh.Bit7-0, 8Fh.Bit7-0 DMA_WrData [15:0]

When The DMA0_Config.DMA_Mode bit is set to 1, it is possible to write data into the FIFO connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA0_Wr bit by accessing these registers.

During operation in 8-bit mode, this DMA access can be accomplished by accessing either DMA0_WrData_H or DMA0_WrData_L.

7.2.118 90h DMA1_FIFO_Control (DMA1 FIFO Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
90h	DMA1_FIFO_Control	R	7: FIFO_Running	0: FIFO is not running	1: FIFO is running	00h
		R / W	6: AutoEnShort	0: Do nothing	1: Auto Enable Short Packet	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

This register shows and sets the FIFO status during DMA1 transfer.

Bit7 FIFO_Running

This bit indicates that the FIFO for the endpoint connected to DMA1 is currently operating. This bit is set to 1 when DMA1 is activated, and is cleared to 0 when DMA1 has finished and the FIFO is emptied.

Bit6 AutoEnShort

If, while this bit is set, some bytes of data less than the max. packet size remain in the FIFO for any endpoint when DMA1 has finished, the EnShortPkt bit for that endpoint is set to 1.

This bit is effective when the endpoint connected to DMA1 is directed IN.

Bit5-0 Reserved

7.2.119 91h DMA1_Config (DMA1 Config)

Address	Register Name	R / W	Bit Symbol	Description		Reset
91h	DMA1_Config	R / W	7: FreeRun	0: Count mode	1: FreeRun mode	00h
		R / W	6: DMA_Mode	0: Normal mode	1: Address Decode mode	
			5:	0:	1:	
			4:	0:	1:	
		R / W	3: ActiveDMA	0: DMA1 Inactive	1: DMA1 Active	
			2:	0:	1:	
		R / W	1: ReqAssertCount [1]	Request Assert Count		
			0: ReqAssertCount [0]			

This register sets the operation mode of DMA1.

Bit7 FreeRun

This bit sets the operation mode of DMA1.

0: Count mode

1: Free-running mode

Bit6 DMA_Mode

This bit sets DMA1 mode.

0: The DMA operates in response to XDACK from the host, regarding it as an acknowledge.

1: The DMA operates in response to an access to the DMA1_RdData/DMA1_WrData register, regarding it as an acknowledge.

Bit5-4 Reserved

Bit3 ActiveDMA

This bit activates DMA1.

0: DMA1 disabled

1: DMA1 enabled

Bit2 Reserved

Bit1-0 ReqAssertCount [1:0]

These bits set the REQ Assert Count option provided to support burst read/burst writes by the CPU.

Set an assert count for XDREQ1 (number of transfer bytes). If the FIFO has writable free space or readable valid data greater than the set assert count, XDREQ1 will be asserted.

When DMA transfers for bytes equal to the set assert count have finished, XDREQ1 is temporarily negated; when the FIFO is confirmed to have writable free space or readable valid data greater than the set assert count again, XDREQ1 is reasserted.

As a result, when XDREQ1 is asserted once, transfers for bytes equal to the set assert count are guaranteed.

However, if DMA1 is set to count mode and the count of remaining bytes in DMA1_Count_HH,HL,LH,LL is smaller than the set assert count, the remaining count in DMA1_Count_HH,HL,LH,LL has priority; therefore, XDREQ1 is asserted when the FIFO has writable free space or readable valid data greater than the remaining count in DMA1_Count_HH,HL,LH,LL.

The table below shows the relationships among DMA1_Count_HH,HL,LH,LL (represented by Count in the table), ReqAssertCount (represented by Req in the table), and the free space/readable data in the FIFO (represented by Ready in the table), and the XDREQ1 signal and the number of transferable bytes.

The remaining count in DMA1_Count_HH,HL,LH,LL must be greater than or equal to 1 if the REQ Assert Count option is to function.

	Count>=Req		**Count<Req**	
	Ready>=Req	**Ready<Req**	**Ready>=Count**	**Ready<Count**
XDREQ0	Asserted	Negated	Asserted	Negated
Number of transferable bytes	Req	-	Req	-

ReqAssertCount [1:0]	**Mode**	
	16bit mode	**8bit mode**
0b00	Normal	Normal
0b01	16Byte(8Count)	16Byte(16Count)
0b10	32Byte(16Count)	32Byte(32Count)
0b11	64Byte(32Count)	64Byte(64Count)

When set to 00 (= Normal), the REQ Assert Count option is unused.

7.2.120 92h DMA1_Control (DMA1 Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
92h	DMA1_Control	R	7: DMA_Running	0: DMA is not running	1: DMA is running	00h
			6:	0:	1:	
			5:	0:	1:	
		W	4: CounterClr	0: Do nothing	1: Clear DMA counter	
			3:	0:	1:	
			2:	0:	1:	
		W	1: DMA_Stop	0: Do nothing	1: Finish DMA	
		W	0: DMA_Go	0: Do nothing	1: Start DMA	

This register controls and shows the status of DMA1.

Bit7 DMA_Running

This bit is set to 1 and remains set while a transfer on DMA1 is underway. While this bit remains set, the EPx{x=0,a-c}Join.JoinDMA1_Rd,Wr bits cannot be rewritten.

Bit6-5 Reserved

Bit4 CounterClr

Setting this bit to 1 clears the DMA1_Count_HH,HL,LH,LL registers to 0. Writing to this bit is ignored while the DMA_Running bit = 1.

Bit3-2 Reserved

Bit1 DMA_Stop

Setting this bit to 1 stops the transfer on DMA1. When the transfer on DMA1 stops, the DMA_Running bit is cleared to 0. Furthermore, the DMA1_Cmp bit in the CPU_IntStat register is set to 1. To restart a transfer on DMA1, check the DMA_Running or the DMA1_Cmp bit in order to confirm the status, and wait until the DMA terminates.

Bit0 DMA_Go

Setting this bit to 1 starts a transfer on DMA1.

7.2.121 93h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
93h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.122 94h DMA1_Remain_H (DMA1 FIFO Remain High)

7.2.123 95h DMA1_Remain_L (DMA1 FIFO Remain Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
94h	DMA1_Remain_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R	3: DMA_Remain [11]	DMA FIFO Remain High		
			2: DMA_Remain [10]			
			1: DMA_Remain [9]			
			0: DMA_Remain [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
95h	DMA1_Remain_L	R	7: DMA_Remain [7]	DMA FIFO Remain Low	00h
			6: DMA_Remain [6]		
			5: DMA_Remain [5]		
			4: DMA_Remain [4]		
			3: DMA_Remain [3]		
			2: DMA_Remain [2]		
			1: DMA_Remain [1]		
			0: DMA_Remain [0]		

94h.Bit7-4 Reserved

94h.Bit3-0, 95h.Bit7-0 DMA_Remain [11:0]

These bits indicate the number of data bytes remaining in the FIFO for the endpoint connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA1_Rd bit, or the amount of free space remaining in the FIFO for the endpoint connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA1_Wr bit.

As the exact amount of free space in the FIFO cannot be determined via these registers immediately after a write to the DMA, insert an interval of at least 1 CPU cycle before checking the amount of free space in the FIFO.

To read these registers, access them in order of DMA1_Remain_H followed by L.

7.2.124 96h - 97h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
96h -97h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.125 98h DMA1_Count_HH (DMA1 Transfer Byte Counter High/High)

7.2.126 99h DMA1_Count_HL (DMA1 Transfer Byte Counter High/Low)

7.2.127 9Ah DMA1_Count_LH (DMA1 Transfer Byte Counter Low/High)

7.2.128 59Bh DMA1_Count_LL (DMA1 Transfer Byte Counter Low/Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
98h	DMA1_Count_HH	R / W	7: DMA_Count [31]	DMA Transfer Byte Counter	00h
			6: DMA_Count [30]		
			5: DMA_Count [29]		
			4: DMA_Count [28]		
			3: DMA_Count [27]		
			2: DMA_Count [26]		
			1: DMA_Count [25]		
			0: DMA_Count [24]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
99h	DMA1_Count_HL	R / W	7: DMA_Count [23]	DMA Transfer Byte Counter	00h
			6: DMA_Count [22]		
			5: DMA_Count [21]		
			4: DMA_Count [20]		
			3: DMA_Count [19]		
			2: DMA_Count [18]		
			1: DMA_Count [17]		
			0: DMA_Count [16]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
9Ah	DMA1_Count_LH	R / W	7: DMA_Count [15]	DMA Transfer Byte Counter	00h
			6: DMA_Count [14]		
			5: DMA_Count [13]		
			4: DMA_Count [12]		
			3: DMA_Count [11]		
			2: DMA_Count [10]		
			1: DMA_Count [9]		
			0: DMA_Count [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
9Bh	DMA1_Count_LL	R / W	7: DMA_Count [7]	DMA Transfer Byte Counter	00h
			6: DMA_Count [6]		
			5: DMA_Count [5]		
			4: DMA_Count [4]		
			3: DMA_Count [3]		
			2: DMA_Count [2]		
			1: DMA_Count [1]		
			0: DMA_Count [0]		

98h-9Bh.Bit7-0 DMA_Count [31:0]

These counter registers are used to set the data length for a transfer on DMA1 in count mode, in byte units. The data length can be set for up to 0xFFFF_FFFF bytes. The counter starts counting down from the set value. After setting transfer bytes in these registers, set the DMA1_Control.DMA_Go bit to 1 in order to start a DMA transfer. When transfers for the transfer bytes set in these registers have finished, the DMA transfer is terminated.

In free-running mode, the counter counts up from the set value. When the value in the DMA1_Count_HH,HL,LH,LL registers overflows, the DMA1_CountUp bit in the CPU_IntStat register is set to 1. The counter is recycled to 0 and restarts counting. In this mode, the number of bytes that has been DMA transferred can be inspected.

As an exact byte count cannot be obtained via these registers immediately after a write to the DMA, insert an interval of at least 1 CPU cycle before checking the byte count. To read these registers, access them in order of DMA1_Count_HH, HL, LH, and then LL.

7.2.129 9Ch DMA1_RdData_H (DMA1 Read Data High)

7.2.130 9Dh DMA1_RdData_L (DMA1 Read Data Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
9Ch	DMA1_RdData_H	R	7: DMA_RdData [15]	DMA Read Data High	XXh
			6: DMA_RdData [14]		
			5: DMA_RdData [13]		
			4: DMA_RdData [12]		
			3: DMA_RdData [11]		
			2: DMA_RdData [10]		
			1: DMA_RdData [9]		
			0: DMA_RdData [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
9Dh	DMA1_RdData_L	R	7: DMA_RdData [7]	DMA Read Data Low	XXh
			6: DMA_RdData [6]		
			5: DMA_RdData [5]		
			4: DMA_RdData [4]		
			3: DMA_RdData [3]		
			2: DMA_RdData [2]		
			1: DMA_RdData [1]		
			0: DMA_RdData [0]		

9Ch.Bit7-0, 9Dh.Bit7-0 DMA_RdData [15:0]

When The DMA1_Config.DMA_Mode bit is set to 1, it is possible to read data from the FIFO connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA1_Rd bit by accessing these registers.

During operation in 8-bit mode, this DMA access can be accomplished by accessing either DMA1_RdData_H or DMA1_RdData_L.

7.2.131 9Eh DMA1_WrData_H (DMA1 Write Data High)

7.2.132 9Fh DMA1_WrData_L (DMA1 Write Data Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
9Eh	DMA1_WrData_H	W	7: DMA_WrData [15]	DMA Write Data High	XXh
			6: DMA_WrData [14]		
			5: DMA_WrData [13]		
			4: DMA_WrData [12]		
			3: DMA_WrData [11]		
			2: DMA_WrData [10]		
			1: DMA_WrData [9]		
			0: DMA_WrData [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
9Fh	DMA1_WrData_L	W	7: DMA_WrData [7]	DMA Write Data Low	XXh
			6: DMA_WrData [6]		
			5: DMA_WrData [5]		
			4: DMA_WrData [4]		
			3: DMA_WrData [3]		
			2: DMA_WrData [2]		
			1: DMA_WrData [1]		
			0: DMA_WrData [0]		

9Eh.Bit7-0, 9Fh.Bit7-0 DMA_WrData [15:0]

When The DMA1_Config.DMA_Mode bit is set to 1, it is possible to write data into the FIFO connected to the DMA by the EPx{x=0,a-c}Join.JoinDMA1_Wr bit by accessing these registers.

During operation in 8-bit mode, this DMA access can be accomplished by accessing either DMA1_WrData_H or DMA1_WrData_L.

7.2.133 A0h IDE_Status (IDE Status)

Address	Register Name	R / W	Bit Symbol	Description		Reset
A0h	IDE_Status	R	7: DMARQ	0:HDMARQ Not Asserted	1:HDMARQ Asserted	00h
		R	6: DMACK	0:XHDMACK Not Asserted	1:XHDMACK Asserted	
		R	5: INTRQ	0:HINTRQ Not Asserted	1:HINTRQ Asserted	
		R	4: IORDY	0:HIORDY Not Asserted	1:HIORDY Asserted	
			3:	0:	1:	
			2:	0:	1:	
		R	1: PDIAG	0:xHPDIAG Not Asserted	1:xHPDIAG Asserted	
		R	0: DASP	0:xHDASP Not Asserted	1:xHDASP Asserted	

This register shows the signal status of the IDE bus. When any signal is asserted, the corresponding bit in this register reads “1.”

Be aware that for the active-low signals XHDMACK, XHPDIAG, and XHDASP, each corresponding bit reads “1” when the voltage level is 0.

Bits3-2 are reserved and read always “0”.

7.2.134 A1h IDE_Control (IDE Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
A1h	IDE_Control		7:	0:	1:	00h
		W	6: IDE_Clr	0: None	1: Clear IDE Circuit	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
		R/W	0: IDE_Go	0: None	1: IDE DMA Go	

This register controls the DMA for the IDE.

Bit7 Reserved

Bit6 IDE_Clr

Setting this bit initializes the IDE circuit. The contents set in this register do not change. Do not set this bit while a DMA operation or a register access sequence when the IDE is underway.

Bit5-1 Reserved

Bit0 IDE_Go

Setting this bit to 1 starts a DMA operation for the IDE. Upon completion of the DMA operation, the IDE_Cmp bit in the IDE_IntStat register is set to 1.

This bit is set to 1, remains set during a DMA transfer, and is reset to 0 upon completion of the transfer.

If this bit is cleared by writing 0 while it is set at 1, the DMA transfer is aborted and thereby terminated, in which case the IDE_Cmp bit in the IDE_IntStat register is not set.

7.2.135 A2h IDE_Config_0 (IDE Configuration 0)

Address	Register Name	R / W	Bit Symbol	Description		Reset
A2h	IDE_Config_0	R/W	7: IDE_BusReset	0: None	1: XHRESET Asserted	00h
		R/W	6: IDE_LongBusReset	0: None	1: XHRESET Asserted	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
		R/W	1: Ultra	0: Non Ultra mode	1: Ultra mode	
		R/W	0: DMA	0: Non DMA mode	1: DMA mode	

This register controls the DMA for the IDE.

Bit7 IDE_BusReset

Setting this bit to 1 causes the XHRESTE signal of the IDE to be asserted for 50 µs. If this bit is set again by writing 1 while it remains set at 1, the XHRESTE signal of the IDE is asserted for a further 50 µs from that point in time. When the XHRESTE signal is asserted by either bit 7 or bit 6, this bit reads "1."

Bit6 IDE_LongBusReset

Setting this bit to 1 causes the XHRESTE signal of the IDE to be asserted for 400 µs. If this bit is set again by writing 1 while it remains set at 1, the XHRESTE signal of the IDE is asserted for a further 400 µs from that point in time. When the XHRESTE signal is asserted by either bit 7 or bit 6, this bit reads "1."

Bit5-2 Reserved

Bit1 Ultra

When this bit is set to 1 simultaneously with bit 0 (= DMA), the DMA for the IDE activated by the IDE_Control register is placed in Ultra mode.

This bit cannot be rewritten while a DMA transfer is underway. The table below lists the DMA transfer modes of the IDE that are set by a combination of bits 1–0.

Bit0 DMA

When this bit is set to 1, the DMA for the IDE activated by the IDE_Control register is placed in Multiword DMA mode.

This bit cannot be rewritten while a DMA transfer is underway. The table below lists the DMA transfer modes of the IDE that are set by a combination of bits 1-0.

Bit1-0	"00"	"01"	"10"	"11"
	PIO	Multiword DMA	Settings prohibited	Ultra

7.2.136 A3h IDE_Config_1 (IDE Configuration 1)

Address	Register Name	R / W	Bit Symbol	Description		Reset
A3h	IDE_Config_1	R/W	7: ActiveIDE	0: InActivated IDE Bus	1: Activate IDE Bus	00h
		R/W	6: DelayStrobe	0: Not Delay Strobe Signal	1: Delay Strobe Signal	
			5:	0:	1:	
		R/W	4: InterLock	0: None	1: DMA InterLock	
			3:	0:	1:	
		R/W	2: Swap	0: Data Swap	1: None	
			1:	0:	1:	
			0:	0:	1:	

This register controls the status of the IDE bus.

Bit7 ActiveIDE

Setting this bit to 1 enables the output signals of the IDE. Before a register read/write command for the IDE bus can be issued or an IDE-DMA can be executed, this bit must first be set to 1. When this bit = 0, all IDE signals are set for input.

Bit6 DelayStrobe

If during a Multiword DMA transfer for IDE-DMA this bit is set to 1, the setup time is held for 2 system clock periods (approx. 33 ns) before the XHIOR/XHIOW strobe signal is asserted after the assertion of XHDMACK. If during a Multiword DMA transfer for IDE-DMA this bit = 0, the XHIOR/XHIOW strobe signal is asserted simultaneously with the assertion of XHDMACK (approx. 0 ns).

Bit5 Reserved

Bit4 InterLock

If during a Multiword DMA transfer for IDE-DMA this bit is set to 1, XHDMACK is not negated due to the fact that the LSI internally became unable to transfer data; instead, the IDE bus is kept active until data is prepared internally in the LSI. When this bit = 0, XHDMACK is temporarily released unless data cannot be prepared internally.

Bit3 Reserved

Bit2 Swap

Clearing this bit to 0 enables the swap function, so that the 8 high-order bits and the 8 low-order bits of data on the IDE bus are reversed when they are input or output. Set this bit to 1 unless this function is necessary. Be aware that this bit is initially set to 0. For details about the Swap bit, refer to Appendix A.

Bit1-0 Reserved

7.2.137 A4h IDE_Rmod (IDE Register Mode)

Address	Register Name	R / W	Bit Symbol	Description	Reset
A4h	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 strobe width for assertion or negation of XHIOR/XHIOW when the IDE bus is accessed in register mode via the CPU.

An appropriate value suitable for the transfer modes of the IDE must be selected.

Bit7-4 RegisterAssertPulseWidth [3:0]

The system clock (60 MHz) period multiplied by "RegisterAssertPulseWidth + 4" comprises the strobe width.

ex. 0000: 4×16.67nS=67nS

0001: 5×16.67nS=83nS

Bit3-0 RegisterNegatePulseWidth [3:0]

The system clock (60 MHz) period multiplied by "RegisterNegatePulseWidth + 4" comprises the strobe width.

ex. 0000: 4×16.67nS=67nS

0001: 5×16.67nS=83nS

7.2.138 A5h IDE_Tmod (IDE Transfer Mode)

Address	Register Name	R / W	Bit Symbol	Description	Reset
A5h	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 strobe width for assertion or negation of XHIOR/XHIOW when the IDE bus is accessed in PIO mode during CPU and DMA transfers.

An appropriate value suitable for the transfer modes of the IDE must be selected.

Bit7-4 TransferAssertPulseWidth [3:0]

The system clock (60 MHz) period multiplied by "TransferAssertPulseWidth + 4" comprises the strobe width.

ex. 0000: 4×16.67nS=67nS

0001: 5×16.67nS=83nS

Bit3-0 TransferNegatePulseWidth [3:0]

The system clock (60 MHz) period multiplied by "TransferNegatePulseWidth + 4" comprises the strobe width.

ex. 0000: 4×16.67nS=67nS

0001: 5×16.67nS=83nS

7.2.139 A6h IDE_Umod (IDE Ultra-DMA Transfer Mode)

Address	Register Name	R / W	Bit Symbol	Description		Reset
A6h	IDE_Umod		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R/W	3: UltraDMA_Cycle [3]	UltraDMA_Cycle		
			2: UltraDMA_Cycle [2]			
			1: UltraDMA_Cycle [1]			
			0: UltraDMA_Cycle [0]			

This register sets the access cycle width when access to the IDE bus in Ultra mode is performed during a DMA transfer.

An appropriate value suitable for the transfer modes of the IDE must be selected.

Bit7-4 Reserved

Bit3-0 UltraDMA_Cycle [3:0]

The system clock (60 MHz) period multiplied by "UltraDMA_Cycle + 2" comprises the access cycle width.

ex. 0000: 2×16.67nS=33nS

0001: 3×16.67nS=50nS

7.2.140 A7h - A9h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
A7h -A9h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.141 AAh IDE_CRC_H (IDE CRC High)

7.2.142 ABh IDE_CRC_L (IDE CRC Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
AAh	IDE_CRC_H	R	7: IDE_CRC [15]	IDE_CRC [15:8]	00h
			6: IDE_CRC [14]		
			5: IDE_CRC [13]		
			4: IDE_CRC [12]		
			3: IDE_CRC [11]		
			2: IDE_CRC [10]		
			1: IDE_CRC [9]		
			0: IDE_CRC [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
ABh	IDE_CRC_L	R	7: IDE_CRC [7]	IDE_CRC [7:0]	00h
			6: IDE_CRC [6]		
			5: IDE_CRC [5]		
			4: IDE_CRC [4]		
			3: IDE_CRC [3]		
			2: IDE_CRC [2]		
			1: IDE_CRC [1]		
			0: IDE_CRC [0]		

These registers show the CRC calculation result during a DMA transfer in Ultra mode of the IDE each time CRC is calculated. The IDE_CRC_H and IDE_CRC_L registers must be accessed in pairs. In such a case, be sure to access the IDE_CRC_H register first.

7.2.143 ACh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
ACh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.144 ADh IDE_Count_H (IDE Transfer Byte Counter High)

7.2.145 AEh IDE_Count_M (IDE Transfer Byte Counter Middle)

7.2.146 AFh IDE_Count_L (IDE Transfer Byte Counter Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
ADh	IDE_Count_H	R/W	7: IDE_Count [23]	IDE_Count [23:16]	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]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
AEh	IDE_Count_M	R/W	7: IDE_Count [15]	IDE_Count [15:8]	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]		

Address	Register Name	R / W	Bit Symbol	Description		Reset
AFh	IDE_Count_L	R/W	7: IDE_Count [7]	IDE_Count [7:1]		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:	0:	1:	

These registers are used to set the number of transfer bytes for a DMA transfer of the IDE. If an attempt is made to start the DMA while 0 bytes are set in these registers, the attempted startup is ignored. To read these registers, be sure to access IDE_Count_H, IDE_Count_M, and IDE_Count_L as one group. In such a case, access IDE_Count_H first. Note that the least significant bit in the IDE_Count_L register always reads "0."

7.2.147 B0h IDE_RegAdrs (IDE Register Address)

Address	Register Name	R / W	Bit Symbol	Description		Reset
B0h	IDE_RegAdrs	R/W	7: IDE_WrReg	0: None	1: IDE Register Write Go	00h
		R/W	6: IDE_RdReg	0: None	1: IDE Register Read Go	
			5:	0:	1:	
			4:	0:	1:	
		R/W	3: IDE_RegAddress [3]	IDE_RegAddress [3:0]		
			2: IDE_RegAddress [2]			
			1: IDE_RegAddress [1]			
			0: IDE_RegAddress [0]			

This register controls register accesses to the IDE bus by the CPU.

Bit7 IDE_WrReg

When this bit is set to 1, the LSI writes to the IDE task file registers in PIO or register mode for writing to the IDE bus using the contents preset in the IDE_WrRegValue_H/L registers. During operation, this bit reads "1"; when the operation finishes, the IDE_RegCmp bit in the IDE_IntStat register is set and this bit is reset to 0. The address for the IDE bus must be set in advance or simultaneously with the writing in IDE_RegAddress. Furthermore, the IDE_Rmod and IDE_Tmod registers must be set to the appropriate mode in advance.

Bit6 IDE_RdReg

When this bit is set to 1, the LSI reads the IDE task file registers in PIO or register mode for reading from the IDE bus, and sets the read value in the IDE_RdRegValue_H/L registers. During operation, this bit reads "1"; when the operation finishes, the IDE_RegCmp bit in the IDE_IntStat register is set and this bit is reset to 0. The address for the IDE bus must be set in advance or simultaneously with the read in IDE_RegAddress. Furthermore, the IDE_Rmod and IDE_Tmod registers must be set to the appropriate mode in advance.

Bit5-4 Reserved

Bit3-0 IDE_RegAddress [3:0]

These bits are used to set the address for a register access to the IDE bus by the CPU that is initiated by setting the IDE_WrReg or IDE_RdReg bit. The table below shows the relationship between these bits and the addresses output to the IDE bus.

IDE_RegAddress [3]	0:XHCS0=0	1:XHCS1=0
IDE_RegAddress [2]	0:HDA2=0	1:HDA2=1
IDE_RegAddress [1]	0:HDA1=0	1:HDA1=1
IDE_RegAddress [0]	0:HDA0=0	1:HDA0=1

7.2.148 B1h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
B1h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.149 B2h IDE_RdRegValue_H (IDE Register Read Value High)

7.2.150 B3h IDE_RdRegValue_L (IDE Register Read Value Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
B2h	IDE_RdRegValue_H	R	7: IDE_RdRegValue [15]	IDE_RdRegValue [15:8]	00h
			6: IDE_RdRegValue [14]		
			5: IDE_RdRegValue [13]		
			4: IDE_RdRegValue [12]		
			3: IDE_RdRegValue [11]		
			2: IDE_RdRegValue [10]		
			1: IDE_RdRegValue [9]		
			0: IDE_RdRegValue [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
B3h	IDE_RdRegValue_L	R	7: IDE_RdRegValue [7]	IDE_RdRegValue [7:0]	00h
			6: IDE_RdRegValue [6]		
			5: IDE_RdRegValue [5]		
			4: IDE_RdRegValue [4]		
			3: IDE_RdRegValue [3]		
			2: IDE_RdRegValue [2]		
			1: IDE_RdRegValue [1]		
			0: IDE_RdRegValue [0]		

These registers are used to store the value that has been read from the IDE bus via an IDE register read by the CPU that was initiated by setting the IDE_RdReg bit in the IDE_RegAdrs register. Furthermore, the value read by an auto status register read that is initiated by setting the IDE_RegConfig register is stored in these registers. To read these registers, be sure to access IDE_RdRegValue_H and IDE_RdRegValue_L in pairs. In such a case, access IDE_RdRegValue_H first.

7.2.151 B4h IDE_WrRegValue_H (IDE Register Write Value High)

7.2.152 B5h IDE_WrRegValue_L (IDE Register Write Value Low)

Address	Register Name	R / W	Bit Symbol	Description	Reset
B4h	IDE_WrRegValue_H	R/W	7: IDE_WrRegValue [15]	IDE_WrRegValue [15:8]	00h
			6: IDE_WrRegValue [14]		
			5: IDE_WrRegValue [13]		
			4: IDE_WrRegValue [12]		
			3: IDE_WrRegValue [11]		
			2: IDE_WrRegValue [10]		
			1: IDE_WrRegValue [9]		
			0: IDE_WrRegValue [8]		

Address	Register Name	R / W	Bit Symbol	Description	Reset
B5h	IDE_WrRegValue_L	R/W	7: IDE_WrRegValue [7]	IDE_WrRegValue [7:0]	00h
			6: IDE_WrRegValue [6]		
			5: IDE_WrRegValue [5]		
			4: IDE_WrRegValue [4]		
			3: IDE_WrRegValue [3]		
			2: IDE_WrRegValue [2]		
			1: IDE_WrRegValue [1]		
			0: IDE_WrRegValue [0]		

These registers are used to set in advance the data to be written to the IDE bus via an IDE register write by the CPU that is initiated by enabling the IDE_WrReg bit in the IDE_RegAdrs register.

7.2.153 B6h IDE_SeqWrRegControl (IDE Sequential Register Write Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
B6h	IDE_SeqWrRegControl	R/W	7: IDE_SeqWrReg	0:	1: IDE Sequence Write Go	00h
		W	6: IDE_SeqWrRegClr	0:	1: Clear IDE Sequence Write	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

This register controls sequential register writing to the IDE bus by the CPU.

Bit7 IDE_SeqWrReg

When this bit is set to 1, up to 16 pairs of the addresses and data preset in IDE_SeqWrRegAdrs and IDE_SeqWrRegValue are written to the IDE bus via IDE task file registers, in the order in which they have been set. When the operation finishes, the IDE_SeqWrRegCmp bit in the IDE_IntStat register is set to 1.

This bit remains set at 1 during a sequential write operation, and is reset to 0 when the operation finishes.

Bit6 IDE_SeqWrRegClr

When this bit is set to 1, up to 16 pairs of the addresses and data preset in IDE_SeqWrRegAdrs and IDE_SeqWrRegValue are discarded, and the registers can thereby be initialized. This bit cannot be set during a sequential write operation.

Bit5-0 Reserved

7.2.154 B7h IDE_SeqWrRegCnt (IDE Sequential Register Write Counter)

Address	Register Name	R / W	Bit Symbol	Description		Reset
B7h	IDE_SeqWrRegCnt		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
		R	4: IDE_SeqWrRegCnt [4]	IDE_SeqWrRegCnt [4:0]		
			3: IDE_SeqWrRegCnt [3]			
			2: IDE_SeqWrRegCnt [2]			
			1: IDE_SeqWrRegCnt [1]			
			0: IDE_SeqWrRegCnt [0]			

This register shows the number of data bytes written in the IDE_SeqWrRegValue register. A value of up to 10h is indicated. The value in this register is decremented as a sequential write to the IDE bus proceeds; when all bytes of data written in the IDE_SeqWrRegValue register have been written to the IDE bus, the value is reset to 0. The value is also reset to 0 by writing 1 to the IDE_SeqWrRegClr bit in the IDE_SeqWrRegControl register.

7.2.155 B8h IDE_SeqWrRegAdrs (IDE Sequential Register Write Address FIFO)

Address	Register Name	R / W	Bit Symbol	Description		Reset
B8h	IDE_SeqWrRegAdrs		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		W	3: IDE_SeqRegAddress [3]	IDE_SeqRegAddress [3:0]		
			2: IDE_SeqRegAddress [2]			
			1: IDE_SeqRegAddress [1]			
			0: IDE_SeqRegAddress [0]			

This register is used to set the address to be output to the IDE bus during a sequential write to the IDE bus that is initiated by setting the IDE_SeqWrRegControl register. This address must be set in pairs with the data in the IDE_SeqWrRegValue register. If the consecutive addresses to be set in this register are the same, there is no need to set the same address again once it has been set. The relationship between the addresses output to the IDE bus and the address bits in this register is the same as for the IDE_RegAddress bits in the IDE_RegAdrs register.

7.2.156 B9h IDE_SeqWrRegValue (IDE Sequential Register Write Value FIFO)

Address	Register Name	R / W	Bit Symbol	Description	Reset
B9h	IDE_SeqWrRegValue	W	7: IDE_SeqWrRegValue [7]	IDE_SeqWrRegValue [7:0]	XXh
			6: IDE_SeqWrRegValue [6]		
			5: IDE_SeqWrRegValue [5]		
			4: IDE_SeqWrRegValue [4]		
			3: IDE_SeqWrRegValue [3]		
			2: IDE_SeqWrRegValue [2]		
			1: IDE_SeqWrRegValue [1]		
			0: IDE_SeqWrRegValue [0]		

This register is used to set the data to be output to the IDE bus during a sequential write to the IDE bus that is initiated by setting the IDE_SeqWrRegControl register. This data must be set in pairs with the address in the IDE_SeqWrRegAdrs register. Up to 16 pairs of addresses and data can be set; write operations exceeding that maximum are ignored. When IDE_SeqWrRegAdrs = 0 (write to the data port for which XHCS = 0 and HDA = 0), the IDE is accessed in 16 bits; therefore, this register must be set twice (in writing order of the lower byte = HDD[15:8] and the upper byte = HDD[7:0] when the IDE_Config_1.Swap bit = 1). In such a case, two pairs of addresses and data out of a maximum of 16 pairs are used. For other addresses, the IDE is accessed in 8 bits, so this register must be set once for each write to the IDE.

7.2.157 Bah - BBh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
BAh -BBh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.158 BCh IDE_RegConfig (IDE Register Configuration)

Address	Register Name	R / W	Bit Symbol	Description		Reset
BCh	IDE_RegConfig	R/W	7: EnAutoStsRd	0: None	1: Auto Status Read Enable	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

This register controls the auto status read operation in a HINTRQ interrupt of the IDE bus.

Bit7 EnAutoStsRd

When this bit is set to 1, the LSI automatically reads the status register of the IDE bus (XHCS0 = 0, HDA = 7) when an HINTRQ interrupt of the IDE bus occurs. Upon completion of the operation, the LSI sets the read value in the IDE_RdRegValue register, and the CompleteINTRQ bit in the IDE_IntStat register is set to 1. This bit remains set even after the operation has finished.

Bit6-0 Reserved

7.2.159 BDh - BFh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
BDh	Reserved		7:	0:	1:	XXh
-BFh			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.160 C0h RAM_Rd_00 (RAM Read 00)
7.2.161 C1h RAM_Rd_01 (RAM Read 01)
7.2.162 C2h RAM_Rd_02 (RAM Read 02)
7.2.163 C3h RAM_Rd_03 (RAM Read 03)
7.2.164 C4h RAM_Rd_04 (RAM Read 04)
7.2.165 C5h RAM_Rd_05 (RAM Read 05)
7.2.166 C6h RAM_Rd_06 (RAM Read 06)
7.2.167 C7h RAM_Rd_07 (RAM Read 07)
7.2.168 C8h RAM_Rd_08 (RAM Read 08)
7.2.169 C9h RAM_Rd_09 (RAM Read 09)
7.2.170 CAh RAM_Rd_0A (RAM Read 0A)
7.2.171 CBh RAM_Rd_0B (RAM Read 0B)
7.2.172 CCh RAM_Rd_0C (RAM Read 0C)
7.2.173 CDh RAM_Rd_0D (RAM Read 0D)
7.2.174 CEh RAM_Rd_0E (RAM Read 0E)
7.2.175 CFh RAM_Rd_0F (RAM Read 0F)
7.2.176 D0h RAM_Rd_10 (RAM Read 10)
7.2.177 D1h RAM_Rd_11 (RAM Read 11)
7.2.178 D2h RAM_Rd_12 (RAM Read 12)
7.2.179 D3h RAM_Rd_13 (RAM Read 13)
7.2.180 D4h RAM_Rd_14 (RAM Read 14)
7.2.181 D5h RAM_Rd_15 (RAM Read 15)
7.2.182 D6h RAM_Rd_16 (RAM Read 16)
7.2.183 D7h RAM_Rd_17 (RAM Read 17)
7.2.184 D8h RAM_Rd_18 (RAM Read 18)
7.2.185 D9h RAM_Rd_19 (RAM Read 19)
7.2.186 DAh RAM_Rd_1A (RAM Read 1A)
7.2.187 DBh RAM_Rd_1B (RAM Read 1B)
7.2.188 DCh RAM_Rd_1C (RAM Read 1C)
7.2.189 DDh RAM_Rd_1D (RAM Read 1D)
7.2.190 DEh RAM_Rd_1E (RAM Read 1E)
7.2.191 DFh RAM_Rd_1F (RAM Read 1F)

Address	Register Name	R / W	Bit Symbol	Description	Reset
C0h -DFh	RAM_Rd_00 ～ RAM_Rd_1F	R	7: RAM Read_xx [7]	RAM Read	00h
			6: RAM Read_xx [6]		
			5: RAM Read_xx [5]		
			4: RAM Read_xx [4]		
			3: RAM Read_xx [3]		
			2: RAM Read_xx [2]		
			1: RAM Read_xx [1]		
			0: RAM Read_xx [0]		

C0h-DFh.Bit7-0 RAM_Rd_xx [7:0]

These registers are used to store the data that has been read from the RAM using the RAM_Rd function. Set the RAM_RdAdrs_H,L registers and the RAM_RdCount register, and then activate the RAM_Rd function using the relevant bit in the RAM_RdControl register. When the data in these registers becomes valid, the FIFO_IntStat.RAM_RdCmp bit is set to 1. If the value set in the RAM_RdCount register is less than 32 bytes, the data read from the RAM is stored in these registers sequentially beginning with RAM_Rd_00. The data stored in the registers exceeding the count for data bytes set in the RAM_RdCount register (e.g., if the count = 16, those stored in RAM_Rd_10 through RAM_Rd_1F) is ignored.

7.2.192 E0h RAM_RdAdrs_H (RAM Read Address High)

7.2.193 E1h RAM_RdAdrs_L (RAM Read Address Low)

Address	Register Name	R / W	Bit Symbol	Description		Reset
E0h	RAM_RdAdrs_H		7:	0:	1:	00h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
		R / W	3: RAM Read Address [11]	RAM Read Address		
			2: RAM Read Address [10]			
			1: RAM Read Address [9]			
			0: RAM Read Address [8]			

Address	Register Name	R / W	Bit Symbol	Description	Reset
E1h	RAM_RdAdrs_L	R / W	7: RAM Read Address [7]	RAM Read Address	00h
			6: RAM Read Address [6]		
			5: RAM Read Address [5]		
			4: RAM Read Address [4]		
			3: RAM Read Address [3]		
			2: RAM Read Address [2]		
			1:		
			0:		

E0h.Bit7-4 Reserved

E0h.Bit3-0, 75h.Bit7-2 RAM Read Address [11:2]

These registers are used to set the start address for RAM_Rd to be performed. After setting these registers, set the RAM_RdCount register and then the relevant bit in the RAM_RdControl register. The RAM_Rd function will be activated. While the RAM_Rd function is active, the value set in these registers changes according to the internal operation in the chip. Therefore, once the RAM_Rd function is activated by setting the relevant bit in the RAM_RdControl register, these registers should not be accessed for reading until the FIFO_IntStat.RAM_RdCmp bit is set. If these registers are accessed for reading while the RAM_Rd function is active, the value in the registers cannot be guaranteed. Note also that if data is written to these registers while the RAM_Rd function is active, the device will operate erratically.

E1h.Bit1-0 Reserved

7.2.194 E2h RAM_RdControl (RAM Read Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
E2h	RAM_RdControl	R / W	7: RAM_GoRdCBW	0: Do nothing	1: RAM Read CBW start	00h
		R / W	6: RAM_GoRd	0: Do nothing	1: RAM Read start	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

Bit7 RAM_GoRdCBW

This bit activates the RAM_Rd function to read the data that has been received in the CBW area.

When this bit is set by writing 1, the RAM_Rd function is activated to read data from the CBW area. There is no need to set the RAM_RdAdrs_H,L registers and the RAM_RdCount register. When the data stored in the RAM_Rd_00 through RAM_Rd_1E registers become valid, the FIFO_IntStat.RAM_RdCmp bit is set to 1, at which time this bit is automatically cleared.

If this bit is set simultaneously with the RAM_GoRd bit, the function of this bit is given priority.

Bit6 RAM_GoRd

This bit activates the RAM_Rd function.

After setting the start address for RAM_Rd to be performed in the RAM_RdAdrs_H,L registers, set the RAM_RdCount register and write 1 to this bit in order to activate the RAM_Rd function. Bytes of data equal to the specified count are read beginning with the specified start address; when the data stored in the RAM_Rd_xx{xx=00-1F} registers become valid, the FIFO_IntStat.RAM_RdCmp bit is set to 1, and this bit is automatically cleared.

If this bit is set simultaneously with the RAM_GoRdCBW bit, the function of the RAM_GoRdCBW bit is given priority.

Bit5-0 Reserved

7.2.195 E3h RAM_RdCount (RAM Read Counter)

Address	Register Name	R / W	Bit Symbol	Description	Reset
E3h	RAM_RdCount		7:		00h
			6:		
		R / W	5: RAM_RdCount [5]	RAM Read Counter	
			4: RAM_RdCount [4]		
			3: RAM_RdCount [3]		
			2: RAM_RdCount [2]		
			1:		
			0:		

Bit7-0 RAM_RdCount [7:0]

These bits are used to set the number of data bytes to be read into the RAM_Rd_xx{xx=00-1F} registers using the RAM_Rd function. After setting the RAM_RdAdrs_H,L registers, set this register and then the relevant bit in the RAM_RdControl register to activate the RAM_Rd function. While the RAM_Rd function is active, the value of this register changes according to the internal operation in the chip. Therefore, once the RAM_Rd function is activated by setting the relevant bit in the RAM_RdControl register, this register should not be accessed for reading until the FIFO_IntStat.RAM_RdCmp bit is set. If this register is accessed for reading while the RAM_Rd function is active, the value in it cannot be guaranteed. Note also that if data is written to this register while the RAM_Rd function is active, the device will operate erratically.

The maximum value that can be set in this register is 32 bytes. Be aware that any byte count set in this register exceeding 32 bytes will cause the device to operate erratically.

7.2.196 E4h - EAh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
E4h -EAh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.197 EBh *ModeProtect(Mode Protection)*

Address	Register Name	R / W	Bit Symbol	Description	Reset
EBh	***ModeProtect***	R / W	7: ***ModeProtect [7]***	Mode Protection	56h
			6: ***ModeProtect [6]***		
			5: ***ModeProtect [5]***		
			4: ***ModeProtect [4]***		
			3: ***ModeProtect [3]***		
			2: ***ModeProtect [2]***		
			1: ***ModeProtect [1]***		
			0: ***ModeProtect [0]***		

Bit7-0 ***ModeProtect [7:0]***

This register protects the values of the ChipConfig register and the ClkSelect.ClkSelect bit. Writing "56h" to this register removes the protection, allowing the ChipConfig register and the ClkSelect.ClkSelect bit to be accessed for writing.

During normal use, after setting values in the ChipConfig register and the ClkSelect.ClkSelect bit, write other than "56h" (e.g., 00h) to this register in order to protect the ChipConfig register and the ClkSelect.ClkSelect bit against writing.

7.2.198 ECh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
ECh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.199 EDh *ClkSelect (Clock Select)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
EDh	***ClkSelect***	R / W	7: ***xActIDE_Term***	0: IDE Termination ON	1: IDE Termination OFF	01h
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
		R / W	0: ***ClkSelect***	0: 12MHz	1: 24MHz	

Bit7 ***xActIDE_Term***

This bit turns pullup/pulldown of the IDE ports on or off.

0：IDE Termination ON

1：IDE Termination OFF

Bit6-1 **Reserved**

Bit0 ***ClkSelect***

This bit selects the clock to be used for the LSI stipulated herein.

0：12MHz

1：24MHz

7.2.200 EEh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
EEh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.201 EFh *ChipConfig (Chip Configuration)*

Address	Register Name	R / W	Bit Symbol	Description		Reset
EFh	***ChipConfig***	R / W	7: ***IntLevel***	0: Low Active	1: High Active	00h
		R / W	6: ***IntMode***	0: 1／0 mode	1: Hi-z／0 mode	
		R / W	5: ***DREQ_Level***	0: Low Active	1: High Active	
		R / W	4: ***DACK_Level***	0: Low Active	1: High Active	
		R / W	3: ***CS_Mode***	0: DACK mode	1: CS mode	
		R / W	2: ***CPU_Swap***	0: Do nothing	1: Bus Swap	
		R / W	1: ***BusMode***	0: XWRH/L mode	1: XBEH/L mode	
		R / W	0: ***Bus8x16***	0: 16bit mode	1: 8bit mode	

This register sets the operation modes of the LSI stipulated herein.

Bit7 ***IntLevel***

This bit sets the logic level of XINT.

0: Active low

1: Active high

Bit6 ***IntMode***

This bit sets the output mode of XINT.

0: 1/0 mode

1: Hi-z/0 mode

Bit5 ***DREQ_Level***

This bit sets the logic levels of XDREQ0 and 1.

0: Active low

1: Active high

Bit4 ***DACK_Level***

This bit sets the logic levels of XDACK0 and 1. XDACK0(1) must be fixed to inactive level when DMA0(1) is not enabled.

0: Active low

1: Active high

Bit3 ***CS_Mode***

This bit sets the operation mode of DMA0 and 1. Setting of this bit is applicable when DMAx{x=0,1}_Config.DMA_Mode is set to "0".

0: DMA access that is enabled by the assertion of XDACK0,1

1: DMA access that is enabled by the assertion of XCS and XDACK0,1

Bit2 ***CPU_Swap***

This bit sets the CPU bus during operation in 16-bit mode. Do not set this bit during operation in 8-bit mode.

0: The even and odd addresses of the bus are used for the upper and lower bytes, respectively.

1: The even and odd addresses of the bus are used for the lower and upper bytes, respectively.

The value set in this bit is made effective by reading the address "E9h" after writing to this register. Similarly, when the internal circuits are reset by enabling the ChipReset.ResetAll bit, this register is initialized, as are other registers; however, the value set in it does not take effect until after the address "E9h" is read.

Bit1-0 ***BusMode, Bus8x16***

These bits set the operation mode of the CPU.

Operation mode	bit1.BusMode	bit0.Bus8x16
16bit Strobe mode	0	0
16bit BE mode	1	*
8bit mode	0	1

7.2.202 F0h Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
F0h	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

7.2.203 F1h D_ModeControl (Device Mode Control)

Address	Register Name	R / W	Bit Symbol	Description		Reset
F1h	D_ModeControl	W	7: (Reserved)	Don't set "1"		XXh
		W	6: (Reserved)	Don't set "1"		
		W	5: (Reserved)	Don't set "1"		
		W	4: SetAddressMode	0: Auto Mode	1: Manual Mode	
		W	3: (Reserved)	Don't set "1"		
		W	2: (Reserved)	Don't set "1"		
		W	1: (Reserved)	Don't set "1"		
		W	0: (Reserved)	Don't set "1"		

Furthermore, refer to Appendix C in detail.

7.2.204 F0h - FFh Reserved ()

Address	Register Name	R / W	Bit Symbol	Description		Reset
F0h -FFh	Reserved		7:	0:	1:	XXh
			6:	0:	1:	
			5:	0:	1:	
			4:	0:	1:	
			3:	0:	1:	
			2:	0:	1:	
			1:	0:	1:	
			0:	0:	1:	

Chapter 8 Electrical Characteristics

8.1 Absolute Maximum Ratings

(V_{SS}=0V)

Parameter	Symbol	Rated Value	Unit
Supply Voltage	HVDD	VSS-0.3 to 4.0	V
	CVDD	VSS-0.3 to 4.0	V
	LVDD	VSS-0.3 to 2.5	V
	PVDD	VSS-0.3 to 2.5	V
Input Voltage	HVI	VSS-0.3 to HVDD+0.5	V
	CVI	VSS-0.3 to CVDD+0.5	V
	PVI	VSS-0.3 to PVDD+0.5	
	VVI *2	VSS-0.3 to 6.0	V
Output Voltage	HVO	VSS-0.3 to HVDD+0.5	V
	UVO	VSS-0.3 to UVDD+0.5	V
Output Current per Pin	IOUT	±10	mA
Storage Temperature	Tstg	-65 to 150	°C

*1 DP, DM

*2 VBUS

8.2 Recommended Operating Conditions

Parameter	Symbol	Min.	Typ.	Max.	Unit
Supply Voltage	HVDD	3.00	3.30	3.60	V
	CVDD *1	3.00	3.30	3.60	V
	CVDD *2	1.65	1.80	1.95	V
	LVDD	1.65	1.80	1.95	V
	PVDD	1.65	1.80	1.95	V
Input Voltage	HVI	-0.3	-	HVDD+0.3	V
	CVI	-0.3		CVDD+0.3	V
	PVI	-0.3	-	PVDD+0.5	V
	VVI	-0.3	-	6.0	V
Ambient Temperature	T_a	-40	25	85	°C*

*1: When using the CPU IF with 3.3 V

*2: When using the CPU IF with 1.8 V

*3: For DP and DM

Make sure the power supplies for the IC are turned on in the order shown below.

LVDD, PVDD (internal) → HVDD, CVDD (IO section)

Power to the IC must be turned off in the following sequence:

HVDD, CVDD (I/O) -> LVDD (int.)

Note: Be aware that if while LVDD and PVDD are shut down, only HVDD and CVDD are applied continuously (for 1 second or more), the reliability of the chip may be adversely affected.

8.3 D.C. Characteristics

Input Characteristics in the D.C. State (under the Recommended Operating Conditions)

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Power-Supply Current	*1					
Supply Current	IDDL	LVDD=1.8V(typ), LVDD=1.95V(max)	-	28	45	mA
	IDDH	HVDD=3.3V(typ), HVDD=3.6V(max)	-	14	23	mA
	IDDCH	CVDD=3.3V(typ), CVDD=3.6V(max)	-	1	4	mA
	IDDCL	CVDD=1.8V(typ), CVDD=1.95V(max)	-	0.7	2	mA
	IDDP	PVDD=1.8V(typ), PVDD=1.95V(max)	-	16	26	mA
Quiescent Current	*2					
Supply Current	IDDS	VIN = HVDD or VSS LVDD = 1.95V HVDD=3.6V CVDD=3.6V PVDD=1.95V	-	-	80	µA
Input Leakage						
Input Leakage Current	IL	HVDD=3.3V CVDD=3.3V/1.8V LVDD=1.8V HVIH=HVDD CVIH=CVDD VIL=VSS	-5	-	5	µA

*1: Typical values are measured current values when operating 17MB/s IDE-USB transfer at Ta=25°C.
Maximum values are approximate current values based on typical values.

*2: Quiescent current for the case in which Ta = 25°C and the bidirectional pins are set for input.

Input Characteristics in the D.C. State (under the Recommended Operating Conditions) (Continued from the preceding page)

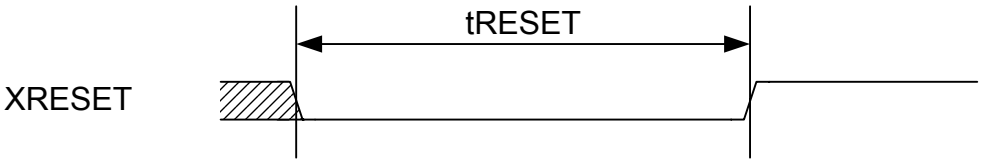
Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Input Characteristics (LVCMOS)	Pin Name:	TDI, TCK, TRST, TMS				
High-Level Input Voltage	VIH1	HVDD = 3.6V	2.1	-	-	V
Low-Level Input Voltage	VIL1	HVDD = 3.0V	-	-	0.9	V
Input Characteristics (LVCMOS)	Pin Name:	CA [7:1], CD [15:0], XCS, XRD, XWRL, XWRH, XBEL, XDACK0, XDACK1, TEST, XRESET				
High-Level Input Voltage	VIH2	CVDD=3.6V	2.1	-	-	V
Low-Level Input Voltage	VIL2	CVDD=3.0	-	-	0.9	V
High-Level Input Voltage	VIH3	CVDD=1.95V	1.22	-	-	V
Low-Level Input Voltage	VIL3	CVDD=1.65V	-	-	0.62	V
Input Characteristics (LVTTL)	Pin Name:	HDD [15:0], HDMARQ, HIORDY, HINTRQ, XHDASP, XHPDIAG				
High-Level Input Voltage	VIH4	HVDD=3.6V	1.9	-	-	V
Low-Level Input Voltage	VIL4	HVDD=3.0V	-	-	0.9	V
Schmitt Input Characteristics (USB:FS)	Pin Name:	DP, DM				
High-Level Trigger Voltage	VT+ (USB)	HVDD = 3.6V	1.1	-	1.8	V
Low-Level Trigger Voltage	VT- (USB)	HVDD = 3.0V	1.0	-	1.5	V
Hysteresis Voltage	ΔV (USB)	HVDD= 3.0V	0.1	-	-	V
Input Characteristics (USB:FS Differential Input)	Pin Name:	DP, DM				
Sensitivity of Differential Input	VDS (USB)	HVDD = 3.0V Differential input voltage 0.8V〜2.5V	-	-	0.2	V
Input Characteristics (VBUS)	Pin Name:	VBUS				
High-Level Trigger Voltage	VT+ (VBUS)	HVDD = 3.6V	1.86	-	2.85	V
Low-Level Trigger Voltage	VT- (VBUS)	HVDD = 3.0V	1.48	-	2.23	V
Hysteresis Voltage	ΔV (VBUS)	HVDD= 3.0V	0.31	-	0.64	V
Pulldown Resistance	RPLDV	VI=5.0V	110	125	150	kΩ
Output Characteristics	Pin Name:	CD [15:0], XDREQ0, XDREQ1, XINT				
High-Level Output Voltage	VOH1	CVDD = 3.0V IOH = -2.6mA	CVDD-0.4	-	-	V V
Low-Level Output Voltage	VOL1	CVDD = 3.0V IOL = 2.7mA	-	-	VSS+0.4	V
High-Level Output Voltage	VOH2	CVDD = 1.65V IOH = -1.3mA	CVDD-0.4	-	-	V
Low-Level Output Voltage	VOL2	CVDD = 1.65V IOL = 1.4mA	-	-	VSS+0.4	V
Output Characteristics	Pin Name:	HDD [15:0], HDA [2:0], XHCS1, XHCS0, XHIOR, XHIOW, XHDMACK, XHRESET,				
High-Level Output Voltage	VOH3	HVDD = 3.0V IOH = -5.2mA	HVDD-0.4	-	-	V
Low-Level Output Voltage	VOL3	HVDD = 3.0V IOL = 5.4mA	-	-	VSS+0.4	V
Output Characteristics	Pin Name:	TDO				
High-Level Output Voltage	VOH2	HVDD = 3.0V IOH = -2.6mA	HVDD-0.4	-	-	V
Low-Level Output Voltage	VOL2	HVDD = 3.0V IOL = 2.7mA	-	-	VSS+0.4	V
Output Characteristics (USB:FS)	Pin Name:	DP, DM				
High-Level Output Voltage	VOH (USB)	HVDD = 3.0V	2.8	-	-	V
Low-Level Output Voltage	VOL (USB)	HVDD = 3.6V	-	-	0.3	V

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Output Characteristics (USB:HS)	Pin Name:	DP, DM				
High-Level Output Voltage	VHSOH (USB)	HVDD = 3.0V	360	-	-	mV
Low-Level Output Voltage	VHSOL (USB)	HVDD = 3.6V	-	-	10.0	mV
Output Characteristics	Pin Name:	CA [15:0], XINT, HDD [15:0], HDA [2:0], XHCS1, XHCS0, XHDMACK, XHRESET				
OFF-State Leakage Current	IOZ	HVDD =3.3V CVDD=3.3V CVDD=1.8 VVOH = HVDD/CVDD VOL = VSS	-2	-	2	µA
Input Characteristics	Pin Name:	HDD[15:8], HDD[6:0], HIORDY, HINTRQ, XHDASP, XHPDIAG				
Pullup Resistance	RPLU	HVDD=3.0V VIL=VSS	-16.4	-	-38.5	µA
Input Characteristics	Pin Name:	HDD7, HDMARQ,				
Pulldown Resistance	RPLD	HVDD=3.0V VIH=VSS	17.2	-	38.7	µA

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Pin Capacitance	Pin Name:	All input pins				
Input Pin Capacitance	CI	f = 1MHz HVDD=CVDD= LVDD=VSS PVDD=PVSS	-	-	8	pF
Pin Capacitance	Pin Name:	All output pins				
Output Pin Capacitance	CO	f = 1MHz HVDD=LVDD= VSS PVDD=PVSS	-	-	8	pF
Pin Capacitance	Pin Name:	All input/output pins (not including DP and DM)				
Input/Output Pin Capacitance 1	CIO1	f = 1MHz HVDD=UVDD= LVDD=VSS PVDD=PVSS	-	-	8	pF
Pin Capacitance	Pin Name:	DP, DM				
Input/Output Pin Capacitance 2	CIO2	f = 1MHz HVDD=CVDD= LVDD=VSS PVDD=PVSS	-	-	8	pF

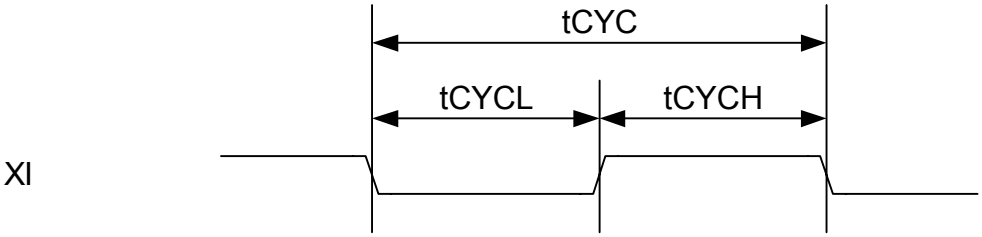
8.4 A.C. Characteristics

8.4.1 RESET Timing



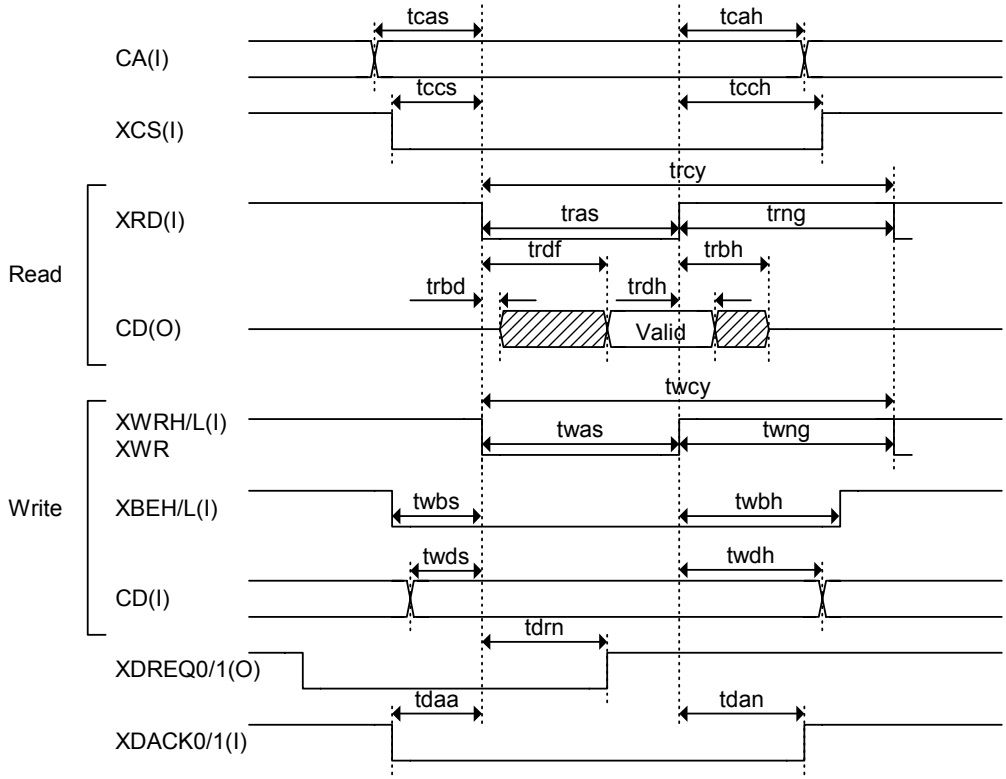
Symbol	Description	Min.	Typ.	Max.	Unit
tRESET	Reset pulse width	40	-	-	ns

8.4.2 Clock Timing



Symbol	Description	Min.	Typ.	Max.	Unit
tCYC~~(*)~~	Clock cycle (ClkSelect = 0)	11.999	12	12.001	MHz
tCYC~~(*)~~	Clock cycle (ClkSelect = 1)	23.998	24	24.002	MHz
tCYCH tCYCL	Clock duty	45	-	55	%

8.4.3 CPU and DAM I/F Access Timing



CLVDD = 3.3V (C_L=30pF)

Symbol	Parameter	Min.	Typ.	Max.	Unit
tcas	Address setup time	6	-	-	ns
tcah	Address hold time	6	-	-	ns
tccs	XCS setup time	6	-	-	ns
tcch	XCS hold time	6	-	-	ns
trcy	Read cycle	75	-	-	ns
tras	Read strobe assert time	37	-	-	ns
trng	Read strobe negate time	25	-	-	ns
trbd	Read data output start time	1	-	-	ns
trdf	Read data valid time	-	-	30	ns
trdh	Read data hold time	1	-	-	ns
trbh	Read data output delay time	-	-	7	ns
twcy	Write cycle	75	-	-	ns
twas	Write strobe assert time	37	-	-	ns
twng	Write strobe negate time	25	-	-	ns
twbs	Write byte enable setup time	6	-	-	ns
twbh	Write byte enable hold time	6	-	-	ns
twds	Write data setup time	0	-	-	ns
twdh	Write data hold time	0	-	-	ns
tdrn	XDREQ0/1 negate delay time	-	-	50	ns
tdaa	XDACK0/1 setup time	6	-	-	ns
tdan	XDACK0/1 hold time	6	-	-	ns

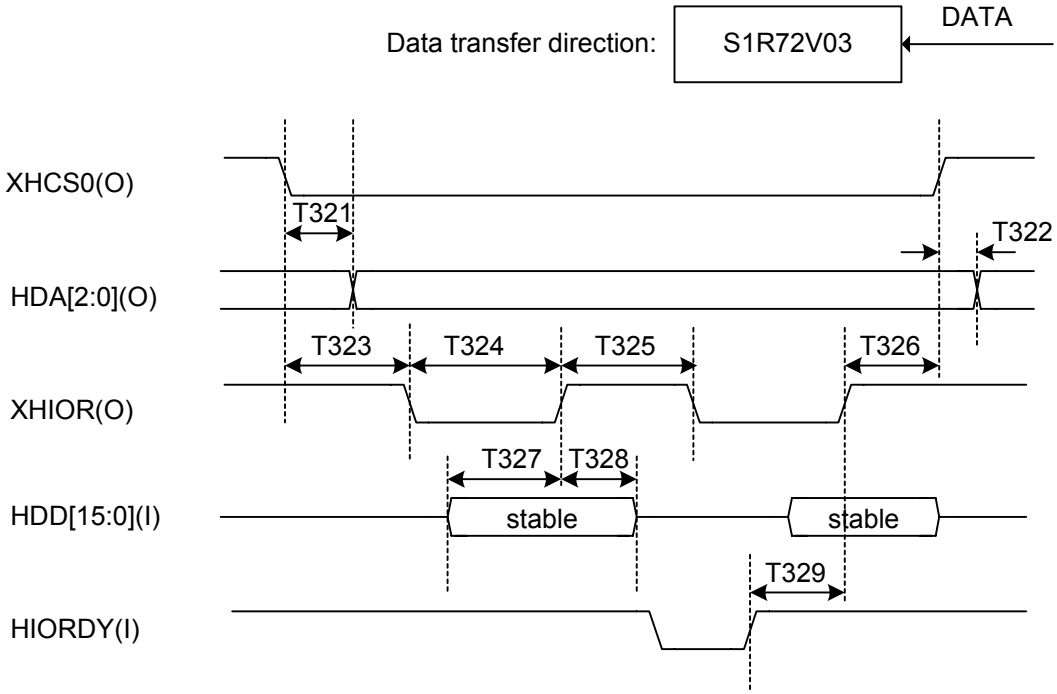
CLVDD = 1.8 V

(C_L=30pF)

Symbol	Description	Min.	Typ.	Max.	Unit
tcas	Address setup time	6	-	-	ns
tcah	Address hold time	6	-	-	ns
tccs	XCS setup time	6	-	-	ns
tcch	XCS hold time	6	-	-	ns
trcy	Read cycle	80	-	-	ns
tras	Read strobe assert time	40	-	-	ns
trng	Read strobe negate time	25	-	-	ns
trbd	Read data output start time	1	-	-	ns
trdf	Read data valid time	-	-	35	ns
trdh	Read data hold time	1	-	-	ns
trbh	Read data output delay time	-	-	8	ns
twcy	Write cycle	75	-	-	ns
twas	Write strobe assert time	40	-	-	ns
twng	Write strobe negate time	25	-	-	ns
twbs	Write byte enable setup time	6	-	-	ns
twbh	Write byte enable hold time	6	-	-	ns
twds	Write data setup time	0	-	-	ns
twdh	Write data hold time	0	-	-	ns
tdrn	XDREQ0/1 negate delay time	-	-	60	ns
tdaa	XDACK0/1 setup time	6	-	-	ns
tdan	XDACK0/1 hold time	6	-	-	ns

8.4.4 IDE I/F Timing

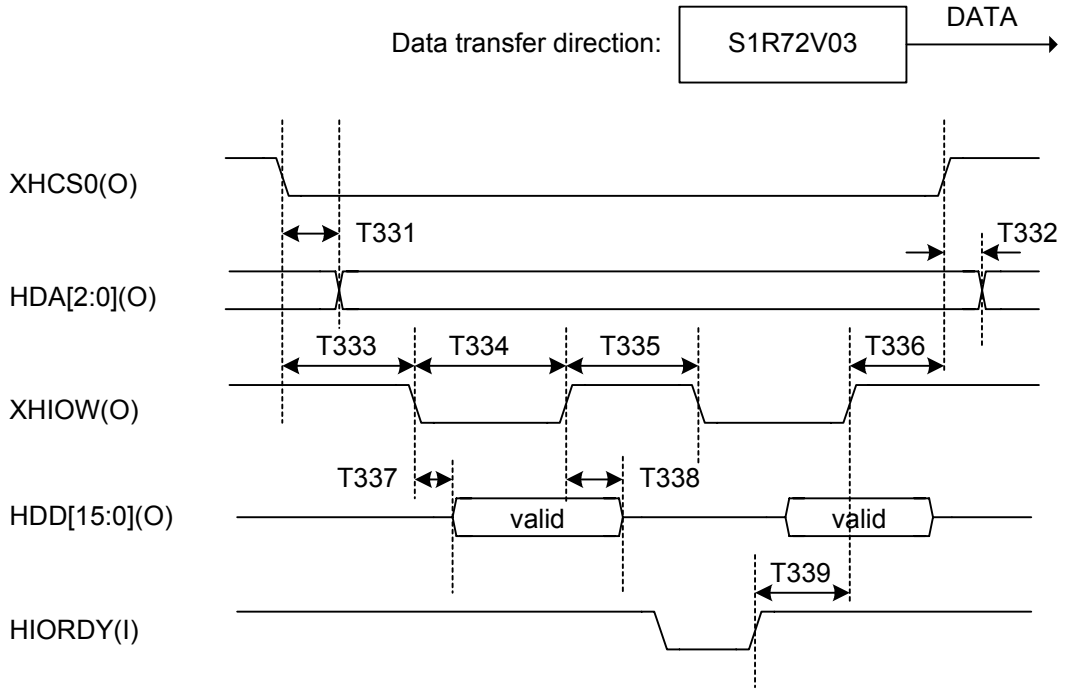
8.4.4.1 PIO Read Timing



Symbol	Description	Min.	Typ.	Max.	Unit
T321	XHCS0 ↓ → HDA HDA output delay time	-	0	-	ns
T322	XHCS0 ↑ → HDA HDA hold time	-	0	-	ns
T323	XHCS0 ↓ → XHIOR ↓ XHCS0 setup time	25	-	-	ns
T324	XHIOR ↓ → XHIOR ↑ XHIOR assert pulse width	-	(AP+4) * 16.7 - 3	-	ns
T325	XHIOR ↑ → XHIOR ↓ XHIOR negate pulse width	-	(NP+4) * 16.7 + 3	-	ns
T326	XHIOR ↑ → XHCS0 ↑ XHCS0 hold time	10	-	-	ns
T327	HDD → XHIOR ↑ Data setup time	10	-	-	ns
T328	XHIOR ↑ → HDD Data hold time	0	-	-	ns
T329	HIORDY assert → XHIOR ↑ XHIOR output delay time	-	-	25	ns

*1: For details on AP=IDE_Tmod.AssertPulseWidth and NP=IDE_Tmod.NegatePulseWidth, refer to "IDE Transfer Mode" in the register explanation.

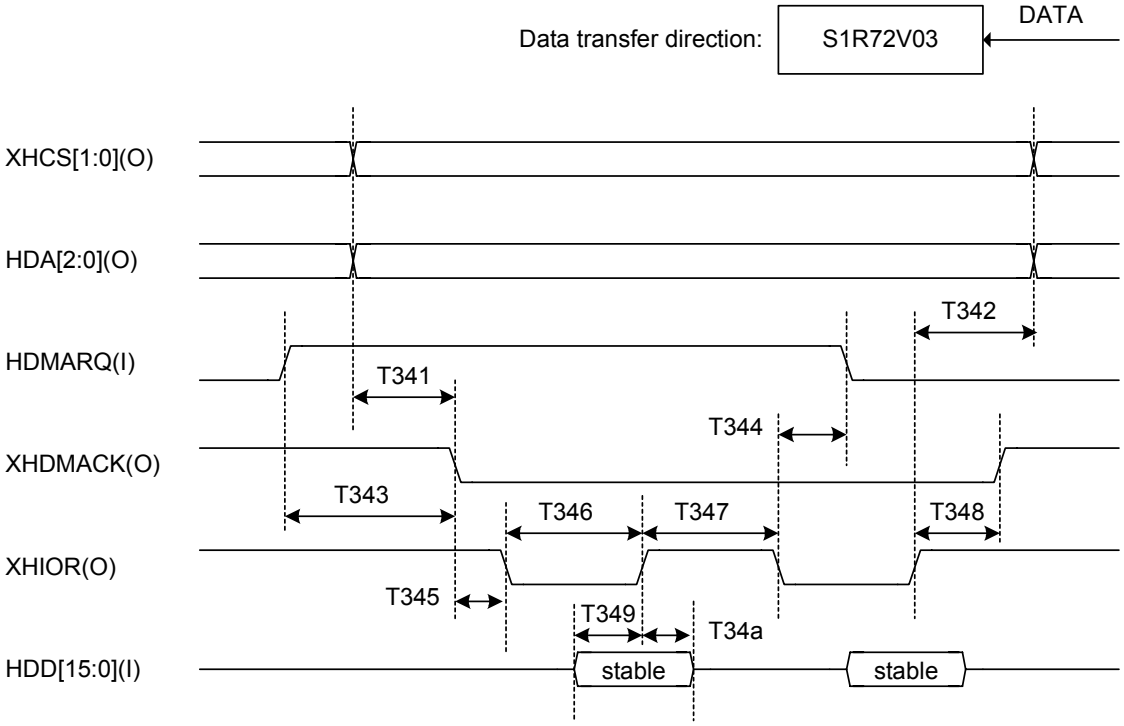
8.4.4.2 PIO Write Timing



Symbol	Description	Min.	Typ.	Max.	Unit
T331	XHCS0 ↓ → HDA HDA output delay time	-	0	-	ns
T332	XHCS0 ↑ → HDA HDA hold time	-	0	-	ns
T333	XHCS0 ↓ → XHIOW ↓ XHCS0 setup time	25	-	-	ns
T334	XHIOW ↓ → XHIOW ↑ XHIOW assert pulse width	-	(AP+4) * 16.7 - 3	-	ns
T335	XHIOW ↑ → XHIOW ↓ XHIOW negate pulse width	-	(NP+4) * 16.7 + 3	-	ns
T336	XHIOW ↑ → XHCS0 ↑ XHCS0 hold time	10	-	-	ns
T337	XHIOW ↓ → HDD Data output delay time	0	-	10	ns
T338	XHIOW ↑ → HDD Data bus negate time	10	-	45	ns
T339	HIORDY assert → XHIOW ↑ XHIOW output delay time	-	-	25	ns

*1: For details on AP=IDE_Tmod.AssertPulseWidth and NP=IDE_Tmod.NegatePulseWidth, refer to "IDE Transfer Mode" in the register explanation.

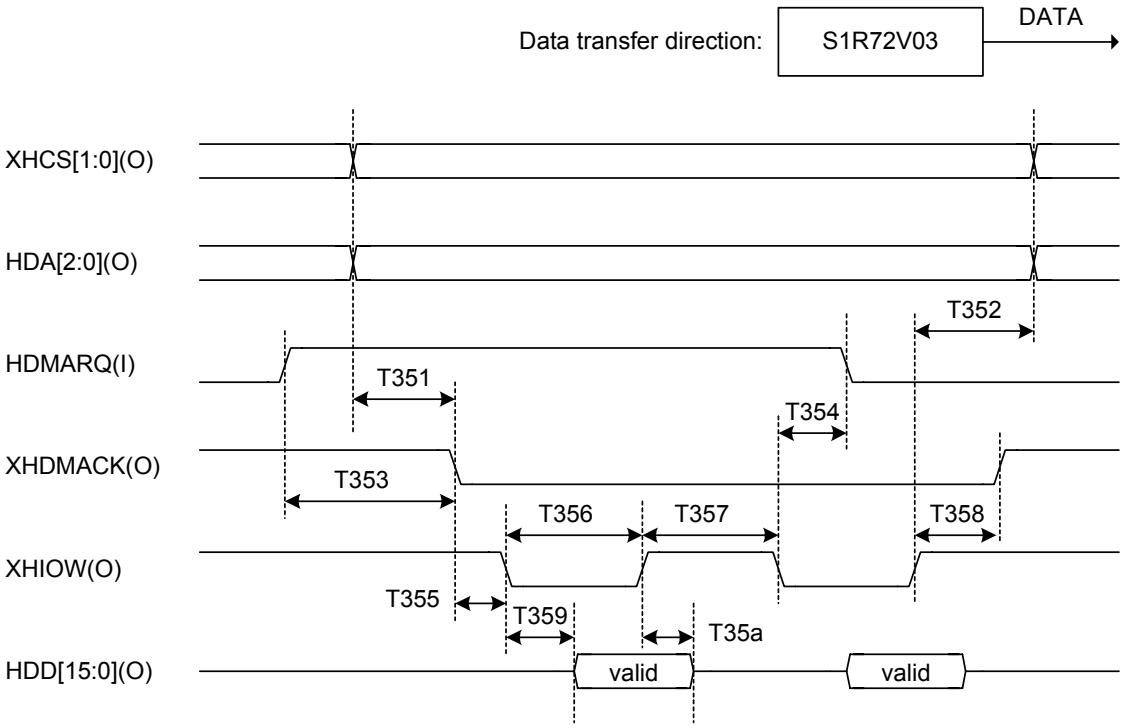
8.4.4.3 DMA Read Timing



Symbol	Description	Min.	Typ.	Max.	Unit
T341	XHCS ↑ 、HDA → XHDMACK ↓ Address setup time	70	-	-	ns
T342	XHIOR ↑ → XHCS ↑ , HDA Address hold time	10	-	-	ns
T343	HDMARQ ↑ → XHDMACK ↓ XHDMACK response time	17	-	-	ns
T344	XHIOR ↓ → HDMARQ negate HDMARQ hold time	0	-	-	ns
T345	XHDMACK ↓ → XHIOR ↓ XHDMACK setup time	0	-	-	ns
T346	XHIOR ↓ → XHIOR ↑ XHIOR assert pulse width	-	(AP+4) * 16.7 - 3	-	ns
T347	XHIOR ↑ → XHIOR ↓ XHIOR negate pulse width	-	(NP+4) * 16.7 + 3	-	ns
T348	XHIOR ↑ → XHDMACK ↑ XHDMACK hold time	5	-	90	ns
T349	HDD → XHIOR ↑ Data setup time	10	-	-	ns
T34a	XHIOR ↑ → HDD Data bus hold time	0	-	-	ns

*1: For details on AP=IDE_Tmod.AssertPulseWidth and NP=IDE_Tmod.NegatePulseWidth, refer to "IDE Transfer Mode" in the register explanation.

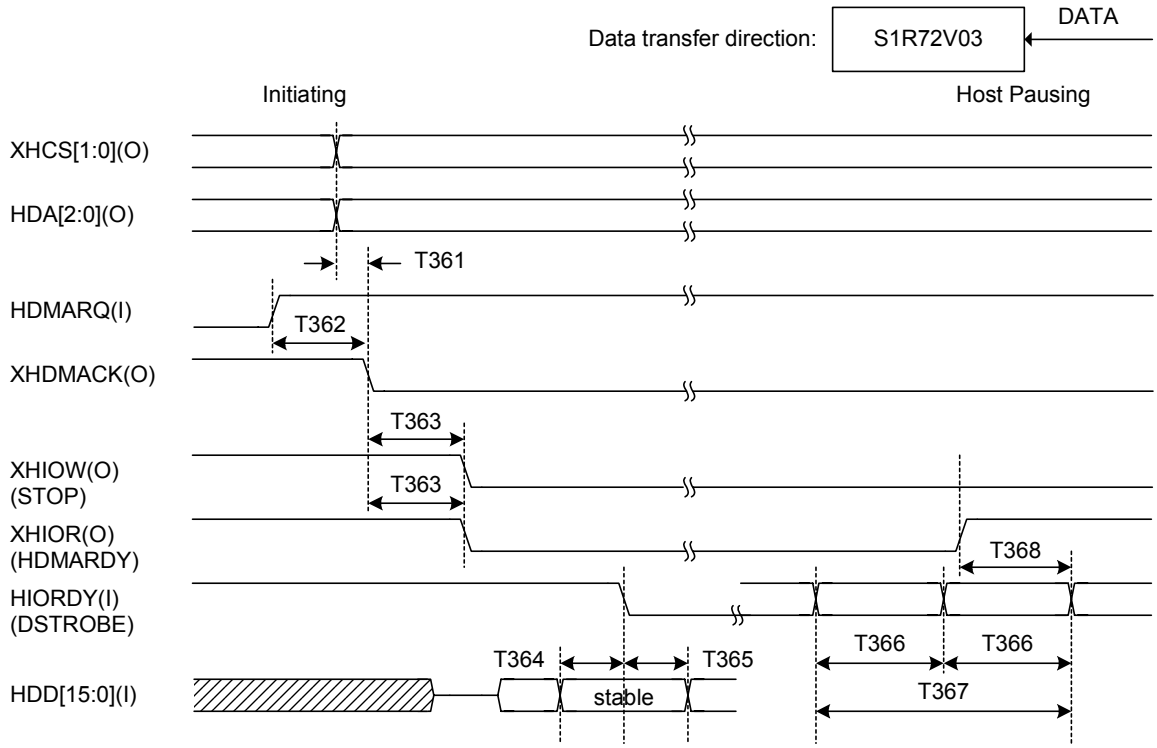
8.4.4.4 DMA Write Timing



Symbol	Description	Min.	Typ.	Max.	Unit
T351	XHCS ↑ , HDA → XHDMACK ↓ Address setup time	70	-	-	ns
T352	XHIOW ↑ → XHCS ↑ , HDA Address hold time	10	-	-	ns
T353	HDMARQ ↑ → XHDMACK ↓ XHDMACK response time	17	-	-	ns
T354	XHIOW ↓ → HDMARQ negate HDMARQ hold time	0	-	-	ns
T355	XHDMACK ↓ → XHIOW ↓ XHDMACK setup time	0	-	-	ns
T356	XHIOW ↓ → XHIOW ↑ XHIOW assert pulse width	-	(AP+4) * 16.7 - 3	-	ns
T357	XHIOW ↑ → XHIOW ↓ XHIOW negate pulse width	-	(NP+4) * 16.7 + 3	-	ns
T358	XHIOW ↑ → XHDMACK ↑ XHDMACK hold time	5	-	90	ns
T359	XHIOW ↓ → HDD Data output delay time	0	-	10	ns
T35a	XHIOW ↑ → HDD Data bus negate time	10	-	45	ns

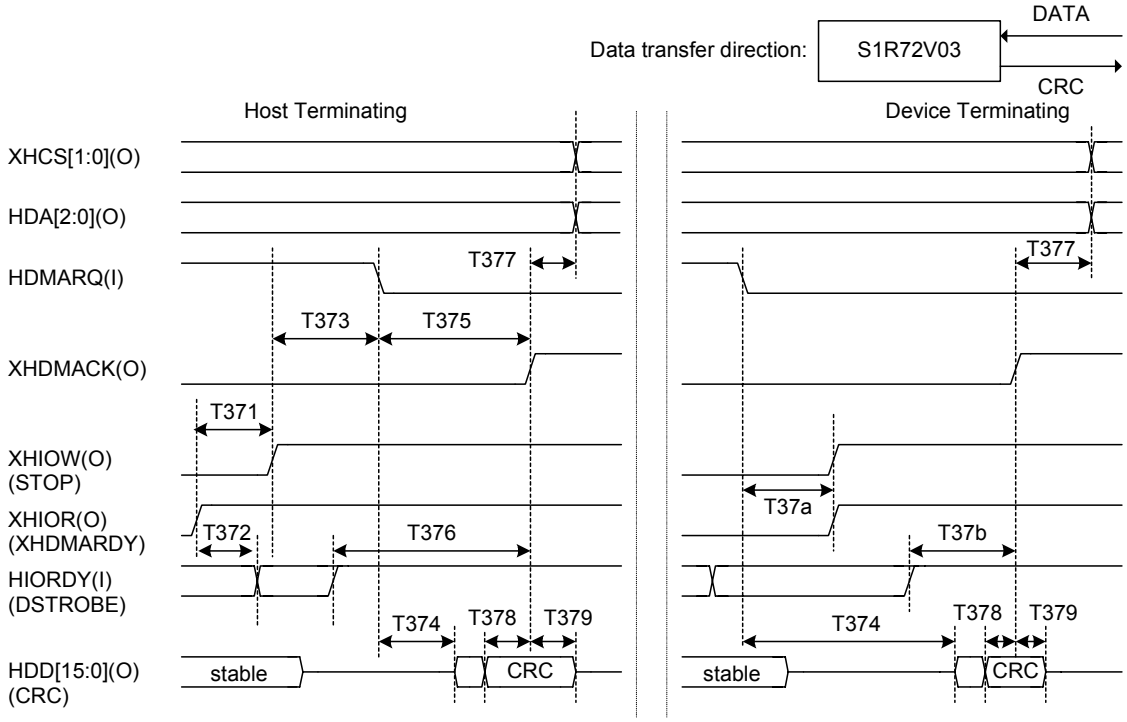
*1: For details on AP=IDE_Tmod.AssertPulseWidth and NP=IDE_Tmod.NegatePulseWidth, refer to "IDE Transfer Mode" in the register explanation.

8.4.4.5 Ultra-DMA Read Timing



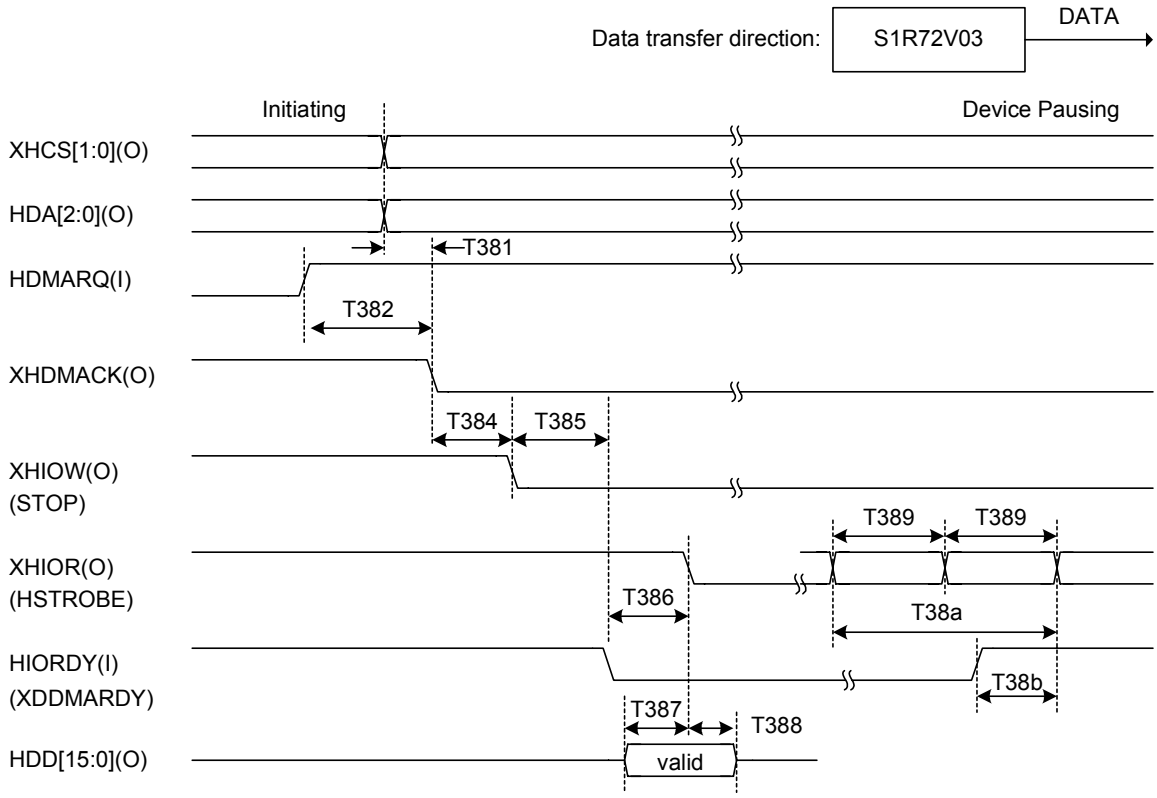
Symbol	Description	Min.	Typ.	Max.	Unit
T361	XHCS ↑ , HDA → XHDMACK ↓ Address setup time	20	-	-	ns
T362	HDMARQ ↑ → XHDMACK ↓ XHDMACK response time	0	-	-	ns
T363	XHDMACK ↓ → XHIOR(W) ↓ Envelope time	20	-	70	ns
T364	HDD → HIORDY Data setup time	4	-	-	ns
T365	HIORDY → HDD Data hold time	4	-	-	ns
T366	HIORDY → HIORDY HIORDY cycle time	15	-	-	ns
T367	HIORDY → HIORDY HIORDY cycle time x 2	30	-	-	ns
T368	XHIOR ↑ → HIORDY Last STROBE time	-	-	IDE standard t_{RFS}	ns

8.4.4.5 Ultra-DMA Read Timing (Continued from the preceding page)



Symbol	Description	Min.	Typ.	Max.	Unit
T371	XHIOR ↑ → XHIOW ↑ Time until STOP asserted	85	-	-	ns
T372	XHIOR ↑ → HIORDY Last STROBE time	-	-	IDE standard t_{RFS}	ns
T373	XHIOW ↑ → HDMARQ ↓ Limited interlock time	-	-	IDE standard t_{LI}	ns
T374	HDMARQ ↓ → HDD Output delay time	20	-	-	ns
T375	HDMARQ ↓ → XHDMACK ↑ Minimum interlock time	160	-	-	ns
T376	HIORDY → XHDMACK ↑ Minimum interlock time	20	-	-	ns
T377	XHDMACK ↑ → XHCS0,1 XHCS0,1 hold time	20	-	-	ns
T378	HDD(CRC) → XHDMACK ↑ CRCdata setup time	6.7	-	-	ns
T379	XHDMACK ↑ → HDD(CRC) CRC data hold time	6.2	-	-	ns
T37a	HDMARQ ↓ → XHIOR ↑ Limited interlock time	0	-	150	ns
T37b	HIORDY → XHDMACK ↑ Minimum interlock time	110	-	-	ns

8.4.4.6 Ultra-DMA Write Timing



Symbol	Description	Min.	Typ.	Max.	Unit
T381	XHCS ↑, HDA → XHDMACK ↓ Address setup time	20	-	-	ns
T382	HDMARQ ↑ → XHDMACK ↓ XHDMACK response time	0	-	-	ns
T384	XHDMACK ↓ → XHIOW ↓ Envelope time	20	-	70	ns
T385	XHIOW ↓ → HIORDY ↓ Limited interlock time	IDE standard t_{LI}	-	IDE standard t_{LI}	ns
T386	HIORDY ↓ → XHIOR ↓ Unlimited interlock time	20	-	-	ns
T387	HDD → XHIOR ↓ Data setup time	-	(cyc+1) * 16.7	-	ns
T388	XHIOR ↓ → HDD Data hold time	-	(cyc+1) * 16.7	-	ns
T389	XHIOR → XHIOR XHIOR cycle time	-	(cyc+2) * 16.7	-	ns
T38a	XHIOR → XHIOR XHIOR cycle time x 2	-	T389 * 2	-	ns
T38b	HIORDY ↑ → XHIOR Last STROBE time	20	-	38	ns

*1:cyc=UltraDMAcycle
For details, refer to "IDE Ultra-DMA Transfer Mode" in the register explanation.

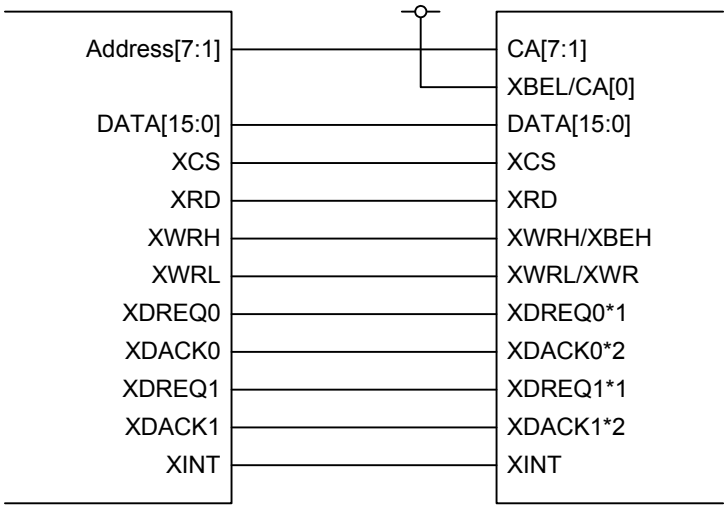
8.4.5 USB I/F Timing

The USB I/F timing conforms to the USB2.0 standard.

<Universal Serial Bus Specification Revision 2.0 released on April 27, 2000>

Chapter 9 Example Connections

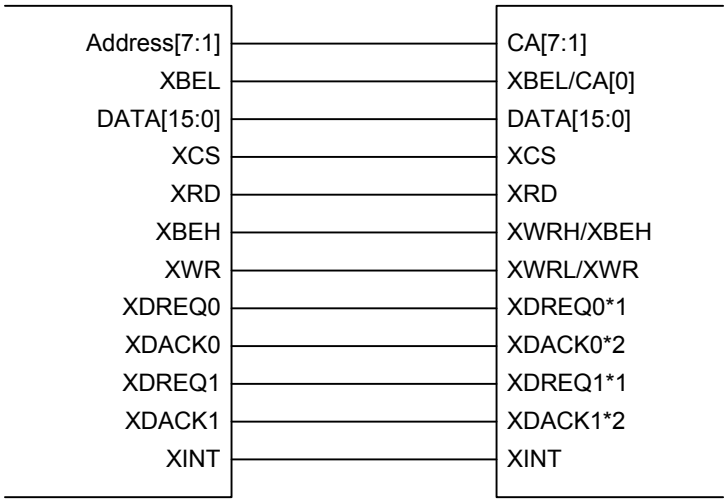
9.1 Example CPU I/F Connection



Example connection for a 16-bit CPU (XWRH/XWRL)

*1: Leave these pins open when not using DMA.

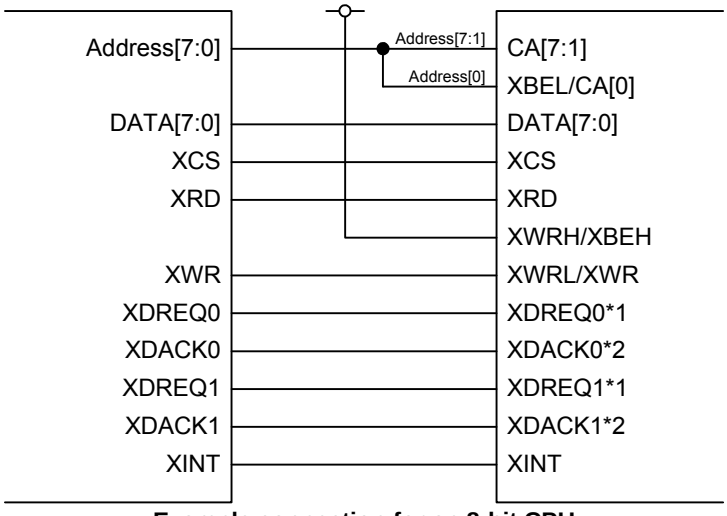
*2: Fix these pins either high or low when not using DMA.



Example connection for a 16-bit CPU (XBEH/XBEL)

*1: Leave these pins open when not using DMA.

*2: Fix these pins either high or low when not using DMA.



Example connection for an 8-bit CPU

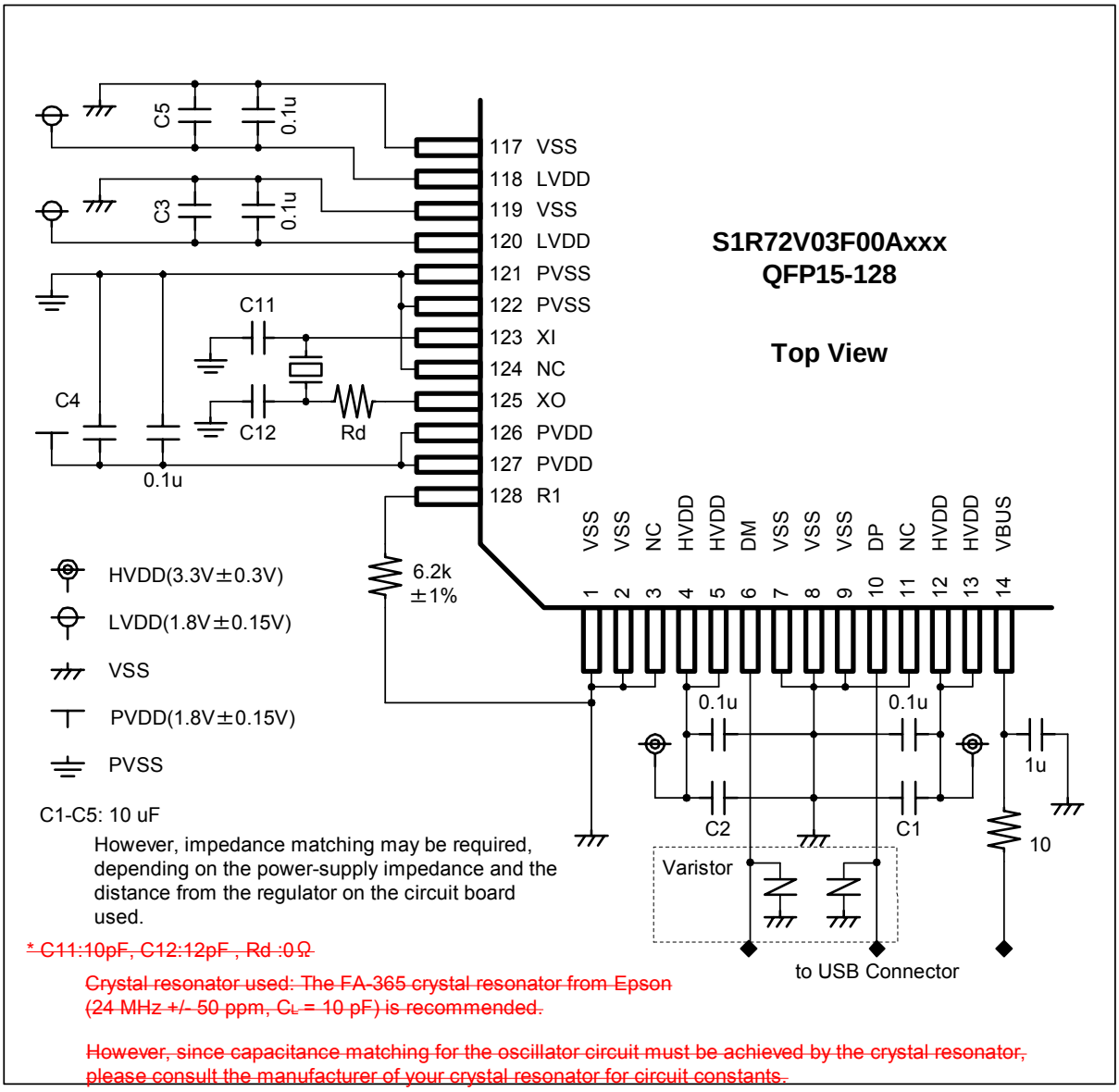
*1: Leave these pins open when not using DMA.

*2: Fix these pins either high or low when not using DMA.

9.2 Example USB I/F Connection

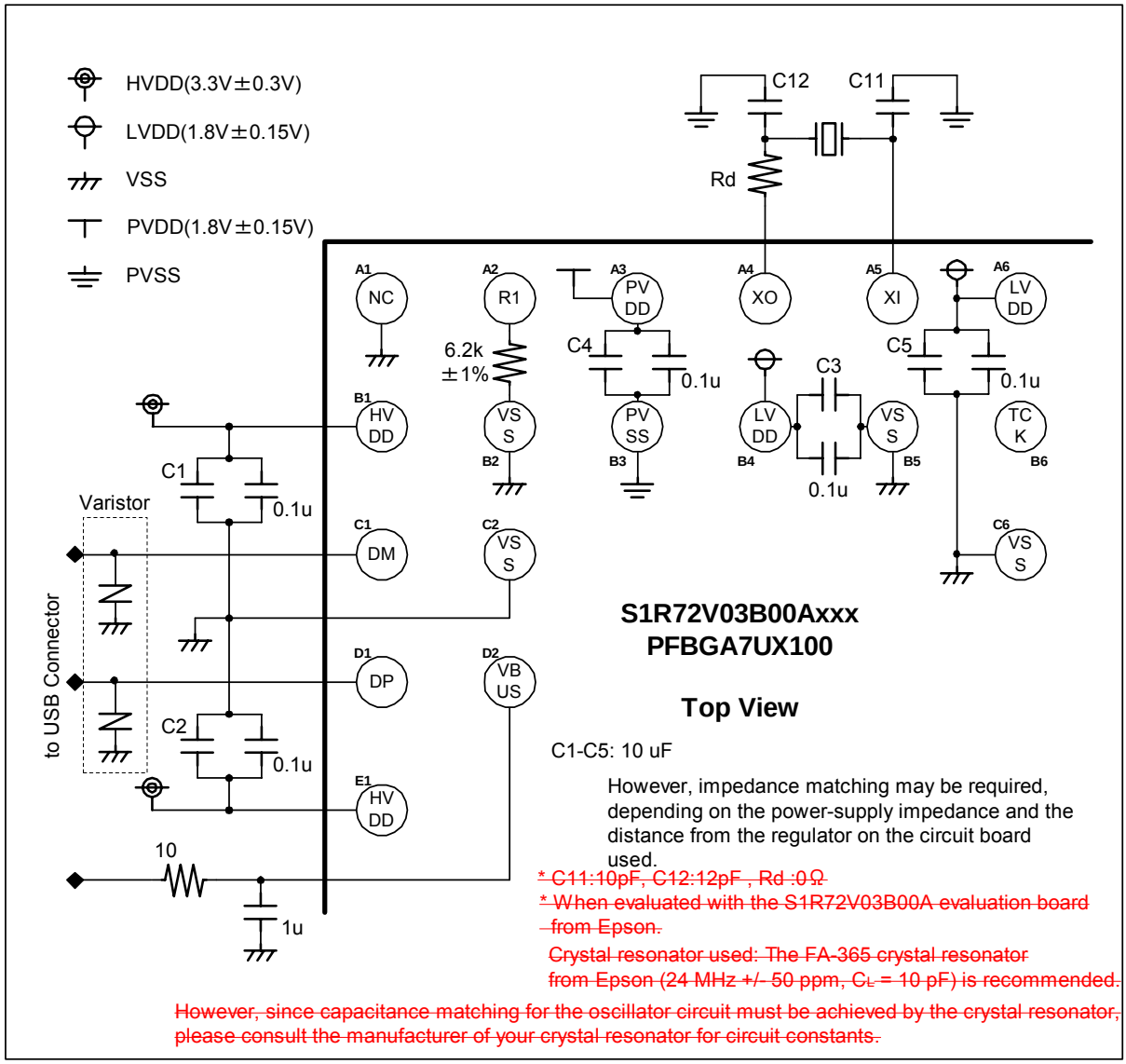
9.2.1 Example for the QFP15-128

Be especially careful when selecting power supply devices because their performance affects the quality of USB signal waveforms.



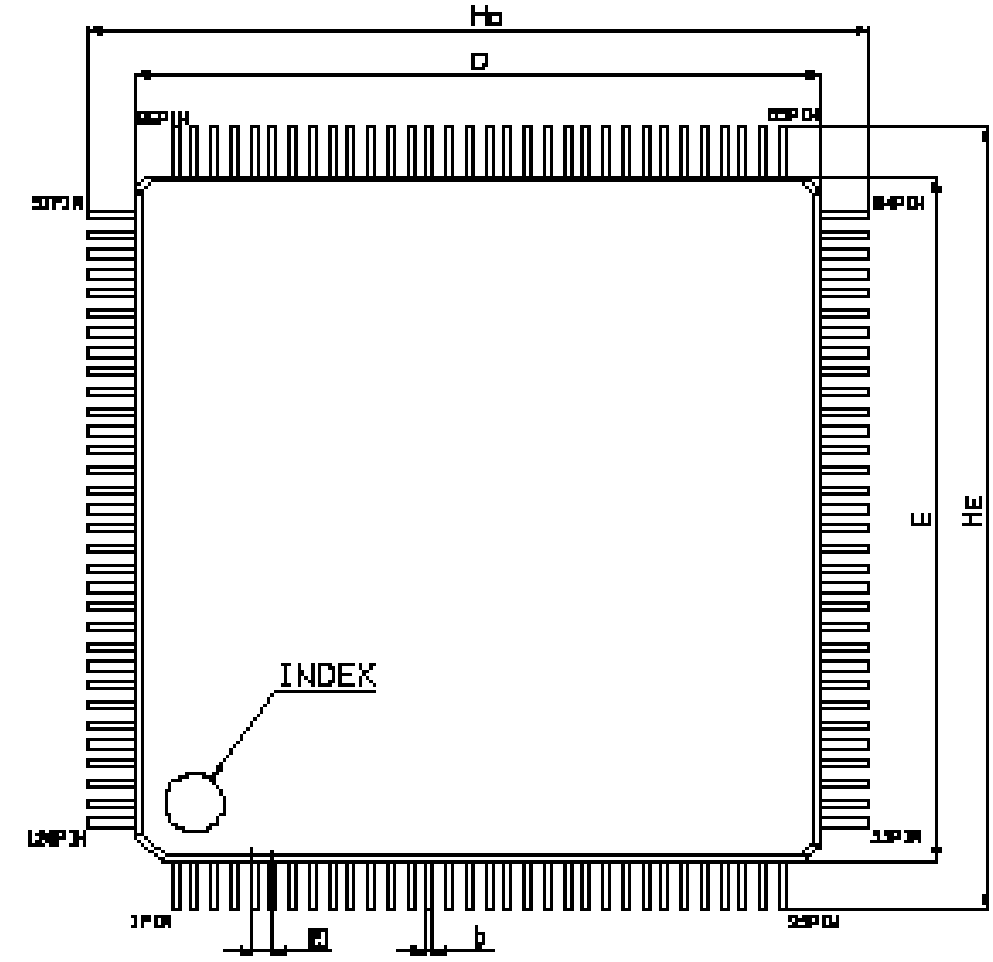
9.2.2 Example for the PFBGA7UX100

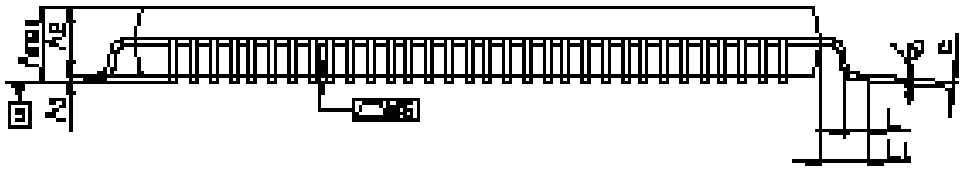
Be especially careful when selecting power supply devices because their performance affects the quality of USB signal waveforms.



Chapter 10 Mechanical Data

10.1 QFP Package



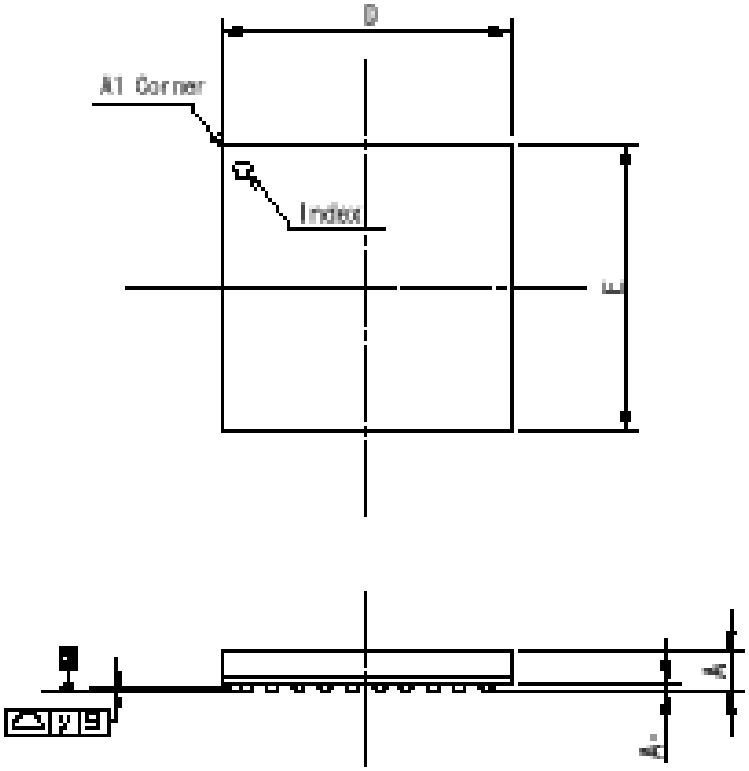


Symbol	Dimension in Millimeters		
	Min	Nom	Max
E	-	14	-
D	-	14	-
A max.	-	-	1.7
A2	-	1.4	-
A1	-	0.1	-
A3	-	0.5	-
b	0.13	-	0.23
c	0.09	-	0.2
θ	0°	-	10°
L	0.3	-	0.75
L1	-	1	-
HD	-	16	-
HE	-	16	-
y	-	-	0.08

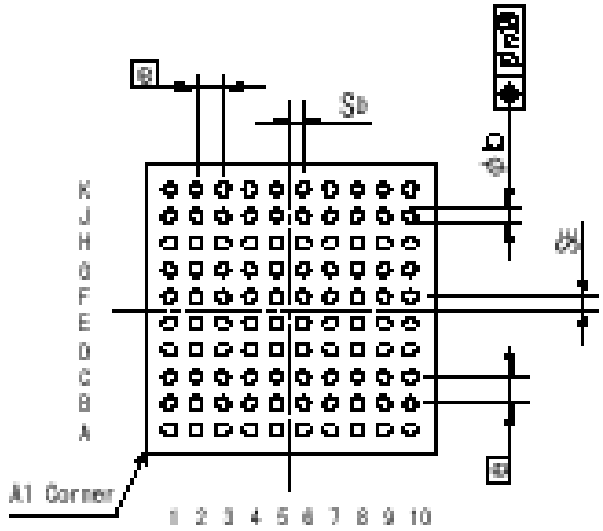
1 = 1mm

10.2 BGA Package

Top View



Bottom View



Symbol	Dimension in Millimeters		
	Min	Nom	Max
D	–	7	–
E	–	7	–
A	–	–	1.2
A_1	–	0.22	–
e	–	0.65	–
b	0.27	–	0.37
x	–	–	0.08
y	–	–	0.1
SD	–	0.325	–
SE	–	0.325	–

1 = 1mm

Appendix A. IDE_Config_1.Swap Bit Settings

The internal buses of the S1R72V03 are configured with the big endian, with the [15:8] side of data comprising the first byte. Conversely, the IDE interfaces are configured with the little endian, with the [7:0] side of data comprising the first byte. The IDE_Config_1.Swap bit allows data bus connections between the internal buses of the S1R72V03 and the IDE interface to be switched over.

The following describes the hardware operation of the S1R72V03, which differs depending on how the IDE_Config_1.Swap bit is set.

In the table below, A ⇒ B denotes that the value of A is mirrored in B.

- DMA transfer of data by the IDE_Control.IDE_Go bit

Swap	HDD[15:0]	
	IDE read	IDE write
0	HDD[15:0] ⇒ Internal bus [15:0]	Internal bus [15:0] ⇒ HDD[15:0]
1	HDD[15:0] ⇒ { Internal bus [7:0], Internal bus [15:8]}	Internal bus [15:0] ⇒ {HDD[7:0], HDD [15:8]}

- IDE data register access

Swap	IDE data register HDD[15:0]			
	IDE read		IDE write	
	IDE_RegAdrs. IDE_RdReg	IDE_RegConfig. EnAutoStsRd	IDE_RegAdrs. IDE_WrReg	IDE_SeqWrRegControl. IDE_SeqWrReg
0	HDD[15:8] ⇒ IDE_RdRegValue_H HDD[7:0] ⇒ IDE_RdRegValue_L	None	IDE_WrRegValue_H ⇒ HDD[15:8] IDE_WrRegValue_L ⇒ HDD[7:0]	IDE_SeqWrRegValue(1^{st}) ⇒ HDD[7:0] IDE_SeqWrRegValue(2^{nd}) ⇒ HDD[15:8]
1	HDD[15:8] ⇒ IDE_RdRegValue_L HDD[7:0] ⇒ IDE_RdRegValue_H	None	IDE_WrRegValue_H ⇒ HDD[7:0] IDE_WrRegValue_L ⇒ HDD[15:8]	IDE_SeqWrRegValue(1^{st}) ⇒ HDD[15:8] IDE_SeqWrRegValue(2^{nd}) ⇒ HDD[7:0]

- IDE task file register access

Swap	IDE task file register HDD[7:0]			
	IDE read		IDE write	
	IDE_RegAdrs. IDE_RdReg	IDE_RegConfig. EnAutoStsRd	IDE_RegAdrs. IDE_WrReg	IDE_SeqWrRegControl. IDE_SeqWrReg
0	HDD[7:0] ⇒ IDE_RdRegValue_H HDD[7:0] ⇒ IDE_RdRegValue_L	Same as shown at left	IDE_WrRegValue_L ⇒ HDD[7:0]	IDE_SeqWrRegValue ⇒ HDD[7:0]
1	Same as shown above	Same as shown above	IDE_WrRegValue_H ⇒ HDD[7:0]	Same as shown above

Appendix B. Connecting to Little Endian CPUs

The internal buses of the S1R72V03 are configured using the big endian, with the even and the odd addresses comprising the upper and lower bytes, respectively. When the S1R72V03 is to be used with a little endian CPU, refer to the following instructions on connecting these devices.

<Circuit board>

The little endian CPU pins and the S1R72V03 pins for data bus and control signals can be connected directly, one-to-one, under the name of each pin. Specifically, connect CD15–CD8 of the S1R72V03 to data bus bits 15–8 (i.e., to the upper byte of the CPU), and connect CD7–CD0 of the S1R72V03 to data bus bits 7–0 (i.e., to the lower byte of the CPU). Additionally, although write signal specifications may differ between CPUs, these write signals can be connected directly, high to high and low to low.

<Firmware>

To operate the S1R72V03 connected to a little endian CPU, follow the procedure described below.

(1) Set the ChipConfig.CPU_Swap bit to 1.

Although this register in the S1R72V03 is mapped to the address 0xEF, when operating in a little endian CPU the CPU behaves as if this register were mapped to the address 0xEE until the operation in (2) below is performed. This is because the S1R72V03 in its initial state is a big endian device; thus the upper and lower bytes of write signals are reversed.

(2) Read the address 0xE9.

This read operation causes the S1R72V03 to reverse the upper and lower bytes of the CPU buses. Please be aware that the byte order of the CPU buses is not changed by the execution of step (1) alone. Once this read operation is performed, all registers are mapped exactly to the addresses shown in Section 7.1, "Register Map."

Please make sure that after the above settings are made, all internal registers except EPnFIFO_Rd_H/L and EPnFIFO_Wr_H/L are accessed in Char (8 bits). The registers defined in Short (16 bits) should also be used in the same way, by accessing them in Char and then casting types in CPU memory.

The registers EPnFIFO_Rd_H/L and EPnFIFO_Wr_H/L are accessible in Short. Even when accessed using the DMAC, these registers can be accessed without resulting in any problems. (See the table below.)

- When data is received from USB sequentially in the order 01, 02, 03, 04, 05, and 06

Accessed in Short	CPU access method			
	Big endian		Little endian	
	CD[15:8]	CD[7:0]	CD[15:8]	CD[7:0]
1st	01	02	02	01
2nd	03	04	04	03
3rd	05	06	06	05

Appendix C About Responses to a SetAddress Request

If a request in which bmRequestType = not 0 (standard request) and bRequest = 0x05 is received, no RcvEP0SETUP interrupt status is issued.

This problem is attributable to the fact that since the automatic address setup function automatically processes a SetAddress request (bmRequestType==0, bRequest==0x05), the RcvEP0SETUP interrupt status is masked with the bRequest value.

To resolve this problem, do one of the following.

1. Limit vendor and class requests.
   Unless a vendor or a class request in which bRequest==0x05 is used, no particular measures need to be taken.

2. Disable the automatic address setup function.
   This problem can be solved by disabling the automatic address setup function. In this case, the function for automatically executing a status stage after receiving a SetAddress request is disabled, so that the status stage of a received SetAddress request must be executed in firmware as for other requests. However, part of the automatic address setup function may be used to automate USB_Address register setup.
   The following shows how to disable the automatic address setup function and describes a control sequence for cases when the automatic address setup function is disabled. For comparison, the control sequence is described for cases in which the automatic address setup function is enabled.

<Process for disabling the automatic address setup function>

Event/process	Automatic address setup function = enabled	Automatic address setup function = disabled
(1) Disabling the automatic address setup function	-	The firmware sets D_ModeControl.SetAddressMode = 1.

(1) Disabling the automatic address setup function
Set D_ModeControl.SetAddressMode = 1.
Once this bit is set after the chip reset, it does not need to be set again thereafter.

<Processing a SetAddress request>

Event/process	Automatic address setup function = enabled	Automatic address setup function = disabled
(1) SetAddress request received	-	The hardware issues an RcvEP0SETUP interrupt status.
(2) Checking the request	-	The firmware checks EP0SETUP0 and EP0SETUP1 for confirmation.
(3) Instructing address setup	-	The firmware makes the setting USB_Address.SetAddress = 1.
(4) Preparing a status stage response	-	The firmware sets the following: SETUP_Control.ProtectEP0 = 0 EP0Control.INxOUT = 1 EP0ControlIN = 0x40* * ForceNAK = 0, EnShortPkt = 1
(5) Status stage executed	The hardware issues a SetAddressCmp interrupt status.	The hardware issues a SetAddressCmp interrupt status.

(1) SetAddress request received
Upon receiving a SetAddress request, the hardware issues an RcvEP0SETUP interrupt status.
Since the automatic address setup function is disabled, this status indicates the receipt of a SETUP transaction for even a SetAddress request, just as for other requests.

(2) Checking the request
The firmware checks the contents of EP0SETUP0 and 1 registers* to determine the values of bmRequestType and bRequest. If bmRequestType==0 and bRequest==0x05, a SetAddress request is assumed.

(3) Instructing address setup
The firmware sets USB_Address.SetAddress = 1.
When a completed status stage is executed after this register setting, the hardware writes the address indicated in the SetAddress request to the USB_Address register over the existing address. It also indicates the completion of this operation by means of a SetAddressCmp interrupt status.

(4) Preparing a status stage response
The firmware executes a process for returning a zero-length packet as in an IN-direction status stage for other requests.
- SETUP_Control.ProtectEP0="0"
- EP0Control.INxOUT="1"
- EP0ControlIN="0x40"(ForceNAK="0", EnShortPkt="1")

(5) Status stage executed
When a status stage (IN transaction) is executed, the hardware issues a SetAddressCmp interrupt status.

Of the registers included in the above procedure, the “D_ModeControl” register shown below has been additionally defined in Development Specifications Rev. 1.60. Bit 7 of the USB_Address register is also defined in Development Specifications Rev. 1.60.

Address	Register Name	R / W	Bit Symbol	Description		Reset
F1h	D_ModeControl	W	7: (Reserved)	Don't set "1"		XXh
		W	6: (Reserved)	Don't set "1"		
		W	5: (Reserved)	Don't set "1"		
		W	4: SetAddressMode	0: Auto mode	1: Manual mode	
		W	3: (Reserved)	Don't set "1"		
		W	2: (Reserved)	Don't set "1"		
		W	1: (Reserved)	Don't set "1"		
		W	0: (Reserved)	Don't set "1"		

Bit7-5 Reserved

Bit4 SetAddressMode

Disables the automatic address setup function.

Bit3-0 Reserved

Revision History

Date	Content of Revision			
	Rev.	Page	Category	Content
04/11/15	0.79	All pages	New	Newly created
04/12/3	0.80	P5, P7, P75	Correction	Test-pin descriptions unified under “TEST”
		P3	Correction	PLL60 block added to the block diagram
		P4	Correction	Oscillator in Section 3.2 changed to PLL60
		P7	Correction	Pin functions PU/PD in Section 5 corrected
		P2	Correction	Endian description in CPU I/F outline corrected
		P23	Correction	Explanations relating to CBW_LengthErr in Section 6.1.6.1 corrected
		P24	Correction	Explanations relating to CBW_Err in Section 6.1.6.1 corrected
		P27	Correction	Section 6.1.8 corrected to Section 6.1.7.11 (Section 6.1.8.? corrected accordingly)
		P50	Correction	Errors in Section 6.3.1.3.1 corrected
		P78	Deletion	Bits deleted: 81h bits 6-4, 91h bits 6-4
		P78	Addition	Bits added: 81h bit 6, 91h bit 6 Registers added: 8Ch-8Fh, 9Ch-9Fh
		P186, 197, 194, 195, 205, 206	Addition	Detailed register descriptions added pursuant to the addition of the above bits and registers
		P55	Addition	Section 6.4.2.2.3 added
		P6	Addition	BGA package pin layout diagram in 4.2 added
		P7, P8, P9, P10	Addition	Ball name added
		P76	Deletion Correction	BulkIntStat.CBW_TranErr deleted BulkIntStat.CBW_Err moved to bit 5, as on other pages on which it is described
		P76	Addition	FIFO_IntStat.FIFO_Full, FIFO_IntStat.FIFO_Empty, and the respective enable bits added, as on other pages on which they are detailed
		P78	Addition	EPx{x=a,b,c}Join.JoinFIFO_Stat bit added, as on other pages on which it is detailed
		P77	Addition	ClrAllJoin.ClrJoinFIFO_Status bit added, as on other pages on which it is detailed
		P243, P244, P245	Correction	DC characteristics corrected
		P258, P259	Correction	Example connections corrected

		P9, P245	Correction	PU/PD additions corrected
		P48	Correction	Storage for the descriptor area corrected
		P80, P81 P83	Addition Correction	Register map DMA0_Config.ActiveDMA and DMA1_Config.ActiveDMA bits added IDE_Count_L register bit 0 deleted ClkSel.xActIDE_Term bit added ChipConfig.Bus8x16 bit name corrected
		P233	Correction Addition	Register description Table for the RAM_RdCount register corrected. Limitations on set bytes added.
		P217	Correction	Register description Explanation corrected to the effect that the IDE_Count_H register should be read first
		P237	Addition	Explanation of the xActIDE_Term bit added
		P239	Correction	Explanation of the CPU_Swap bit corrected
		P27	Correction	Section 6.1.7.8 Auto-negotiation error status bits corrected
		P89, P104	Addition	Explanation of the processing of interrupt status in Active60 added
		P89, P104, P120, P121, P122, P123, P135	Addition	Explanation made to the effect that registers cannot be accessed in Active60 added
		P235	Addition	Registers protected by ModeProtect added
		P53, P239	Addition	Explanation relating to the CPU_BusSwap bit added
		P186, P197	Correction	Errors in the explanation of ActiveDMA corrected
		P247	Addition	CPUIF AC specifications stipulated
		P53, 54	Addition	Explanation of mode-switching added
05/03/02	0.90	P24, P106	Correction	RAM_Monitor corrected to RAM_Rd
		P23	Correction	DTGo corrected to IDE_Go
		P10	Correction	Pins 118 and 120 added to the LVDD column for correction
		P125	Correction	Errors in the description of bit 0 corrected
		P149, P157, P165, P173	Correction	Erroneously written bit names corrected
		P206	Correction	Errors in the description of bits corrected
		P76	Correction	0x14h bits 3 and 2 added for correction

		P106	Correction	Bit 2 R/W attribute added for correction
		P10	Correction	VSS J3 corrected to J8
		P186, P197	Correction	Description of DMAx_Config(x=0,1).ActiveDMA bit corrected
		P149, P157, P165, P173	Correction	Exclusive bit manipulation of the EPxJoin(x=0,a-c) corrected. Only one of bits 5-0 can be set at a time
		P7	Correction	XI A1 -> A5
05/4/1	0.91	P81, P210	Correction	IDE_Config1.bit2 Swap corrected to reserved bit
		P243	Correction	Input leakage added
		P72	Correction	Instructions to write the initial value to the IDE_CRC_L/H register deleted
		P215	Correction	IDC_CRC_L/H corrected to read-only. A description of initial value settings deleted
		P122	Correction	A description added to USB_Status.bit6: FSxHS to the effect that the bit must be set to 1 when cable is attached
		P117	Correction	Description of PM_Control.bit6-4PM_State corrected
		P242	Correction	Power-on sequence added for correction
05/4/27	1.0	P246	Insertion	RESET timing and clock timing added. Page 212 inserted and the subsequent page numbers corrected
		P7	Correction	Description is made of how to process pins when JTAG functions are unused
		P243	Correction	Supply current and quiescent current written in
		P247	Correction	In CPU access timing, 3.3 V corrected and 1.8 V added
		P249~ P255	Correction	IDE standard values corrected
		P258, 259	Correction	Precautions on power supply devices added
05/5/30	1.1	P68	Correction	Description about IDE_Config_1.Swap bit added
		P69	Correction	Data write sequence during sequential write in 6.5.1.3 corrected
		P81	Addition	IDE_Config_1.Swap bit added
		P210	Addition	IDE_Config_1.Swap bit added
		P225	Addition	Description of IDE_SeqWrRegValue register write sequence supplemented
		P262	Addition	Appendix A added
		P263	Addition	Appendix B added
		P258, 259	Correction	Reference values shown for C11, C12, and Rd values. Recommended crystal resonator mentioned
05/6/2	1.2	P263	Correction	Appendix B corrected (Full page correction)
		P9	Correction	The value of the IDE I/F pin corrected when reset

05/10/3	1.3	P8	Correction	Pin type of XINT pin corrected
		P242, 245	Deletion	Description of UVDD deleted
		P242	Addition	CVDD, PVDD input voltages added to the absolute maximum ratings CVDD, PVDD, and VBUS input voltages added to the recommended operating conditions
		P7,P8, P9	Addition	Detailed descriptions of the power supply system of which each pin is a part
		P244	Correction	IVDD changed to HVDD
		P263	Correction	B9h corrected to E9h
		P257	Correction	Data width in the example connection for 8-bit CPU corrected
06/1/21	1.4	P53	Correction	The ChipConfig register address EDh in Section 6.4.2.1 corrected to EFh
06/04/10	1.5	P179	Deletion	Following error description deleted; “JoinCPU_Rd bit and the value of EPnRdRemain is valid. If, when the EPx{x=0,a-c}Join. Join CPU_Rd bit has been set, no data exists in the relevant endpoint, this bit is not set until data is written to the endpoint from the USB or the like (not RdRemainValid=0, however).”
		P258, 259	Correction	Electrostatic varistors are loaded on the DP/DM lines.
06/07/31	1.6	P2	Correction	Incorrect) Dual-power supply system Correc) Triple-power supply system
		P4	Addition	Description for the clock source of PLL added
		P7	Correction	Incorrect) Reset value and Pin type of VBUS are not specified Correct) Reset value and Pin type of VBUS are specified as PD.
		P16	Correction	Incorrect) return an ACK response Correct) return an ACK or NYET response
		P52	Corerction	Corrects description about limitation on FIFO access
		P75	Correction	Incorrect) Manufacturer code is specified as 17D Correct) Manufacturer code is specified as 0x0BE
		P75	Correction	Incorrect) R1 is not specified as Boundary Scan Exclusion Pin Correct) R1 is specified as Boundary Scan Exclusion Pin
		P77, P135	Correction	SetAddress bit is added on 0x38 USB_Address register
		P83, P241, P264	Correction	Incorrect) D_ModeControl is not specified Correct) D_ModeControl is specified on 0xF1 Adds detail information for SetAddress issue on Appendix C
		P186, P197	Correction	Description of ActiveDMA bit is corrected
		P197	Correction	Incorrect) "DMA0" on the title Correct) "DMA1"

		P239	Addition	Further description for DACK_Level bit is added
		P239	Addition	Further description for CS_Mode bit is added
		P242	Addition	Description of power-off procedure added
		P243	Correction	Power Supply Current value and condition corrected Quiescent Current value and condition corrected
		P244	Addition	Input Characteristics for Pulldown Resistance on VBUS pin is added
		P246	Deletion	Unspecified marker "*" on tCYC is deleted
		P256	Addition	Reference document name added
		P258, P259	Deletion	Description which specify a specific parts are deleted

EPSON

International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.
HEADQUARTERS
150 River Oaks Parkway
San Jose, CA 95134, U.S.A.
Phone: +1-800-228-3964 FAX: +1-408-922-0238

SALES OFFICES

West
1960 E.Grand Avenue Flr 2
El Segundo, CA 90245, U.S.A.
Phone: +1-800-249-7730 FAX: +1-310-955-5400

Central
101 Virginia Street, Suite 290
Crystal Lake, IL 60014, U.S.A.
Phone: +1-800-853-3588 FAX: +1-815-455-7633

Northeast
301 Edgewater Place, Suite 210
Wakefield, MA 01880, U.S.A.
Phone: +1-800-922-7667 FAX: +1-781-246-5443

Southeast
3010 Royal Blvd. South, Suite 170
Alpharetta, GA 30005, U.S.A.
Phone: +1-877-332-0020 FAX: +1-770-777-2637

EUROPE

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

DÜSSELDORF BRANCH OFFICE
Altstadtstrasse 176
51379 Leverkusen, GERMANY
Phone: +49-2171-5045-0 FAX: +49-2171-5045-10

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

BARCELONA BRANCH OFFICE
Barcelona Design Center
Edificio Testa, C/Alcalde Barnils 64-68, Modulo C 2a planta
E-08190 Sant Cugat del Vallès, SPAIN
Phone: +34-93-544-2490 FAX: +34-93-544-2491

UK & IRELAND BRANCH OFFICE
8 The Square, Stockley Park, Uxbridge
Middx UB11 1FW, UNITED KINGDOM
Phone: +44-1295-750-216/+44-1342-824451
FAX: +44-89-14005 446/447

Scotland Design Center
Integration House, The Alba Campus
Livingston West Lothian, EH54 7EG, SCOTLAND
Phone: +44-1506-605040 FAX: +44-1506-605041

ASIA

EPSON (CHINA) CO., LTD.
23F, Beijing Silver Tower 2# North RD DongSanHuan
ChaoYang District, Beijing, CHINA
Phone: +86-10-6410-6655 FAX: +86-10-6410-7320

SHANGHAI BRANCH
7F, High-Tech Bldg., 900, Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5522 FAX: +86-21-5423-5512

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 Electronic Technology Development (Shenzhen) LTD.
12/F, Dawning Mansion, Keji South 12th Road,
Hi- Tech Park, Shenzhen
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

EPSON TAIWAN TECHNOLOGY & TRADING LTD.
14F, No. 7, Song Ren Road,
Taipei 110
Phone: +886-2-8786-6688 FAX: +886-2-8786-6677

HSINCHU OFFICE
No. 99, Jiangong Road,
Hsinchu City 300
Phone: +886-3-573-9900 FAX: +886-3-573-9169

EPSON SINGAPORE PTE., LTD.
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

SEIKO EPSON CORPORATION
KOREA OFFICE
50F, KLI 63 Bldg., 60 Yoido-dong
Youngdeungpo-Ku, Seoul, 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

GUMI OFFICE
2F, Grand B/D, 457-4 Songjeong-dong,
Gumi-City, KOREA
Phone: +82-54-454-6027 FAX: +82-54-454-6093

SEIKO EPSON CORPORATION
SEMICONDUCTOR OPERATIONS DIVISION

IC Sales Dept.
IC Marketing Group
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 410103704
First Issue May 2005
Printed January 2006 in JAPAN