EPSON
EXCEED YOUR VISION

# S1R72V17***
# Technical Manual

SEIKO EPSON CORPORATION

Rev.1.6

## NOTICE

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

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

©SEIKO EPSON CORPORATION 2007, All rights reserved.

# General Rules

## Scope of Application

This specification applies to the USB2.0 Controller "S1R72V17B00A***/S1R72V17B00B***/S1R72V17F00C***" manufactured by the Semiconductor Operations Division of Seiko Epson Corporation.

# Table of Contents


**1. Overview ..... 1**

**2. Features ..... 2**

**3. Block Diagram ..... 3**

3.1 Multi Transceiver Macro (MTM) ..... 4

3.2 Oscillator ..... 4

3.3 Device Serial Interface Engine (Device SIE) ..... 4

3.4 Host Serial Interface Engine (Host SIE) ..... 4

3.5 FIFO and FIFO Controller ..... 4

3.6 CPU I/F Controller ..... 4

3.7 DMA Controller ..... 4

3.8 Test MUX ..... 4

**4. Pin Layout Diagram ..... 5**

**5. Pin Description ..... 7**

**6. Functional Description ..... 10**

6.1 Selection of USB Device/Host ..... 10

6.1.1 Selection of the USB Device/Host Functions ..... 10

6.1.2 USB Port State Change Detection Status ..... 11

6.1.2.1 Example Usage of USB Port State Change Detection Status ..... 11

6.1.2.1.1 Device Port Change Status ..... 11

6.1.2.1.2 Host Port Change Status ..... 12

6.2 USB Device Control ..... 13

6.2.1 Endpoints ..... 13

6.2.2 Transactions ..... 15

6.2.2.1 SETUP Transactions ..... 17

6.2.2.2 Bulk/Interrupt OUT Transactions ..... 18

6.2.2.3 Isochronous OUT Transaction ..... 19

6.2.2.4 Bulk/Interrupt IN Transactions ..... 20

6.2.2.5 Isochronous IN Transaction ..... 21

6.2.2.6 PING Transactions ..... 21

6.2.3 Control Transfers ..... 22

6.2.3.1 Setup Stage ..... 24

6.2.3.2 Data Stage and Status Stage ..... 24

6.2.3.3 Automatic Address Setup Function ..... 24

6.2.3.4 Descriptor Reply Function ..... 25
6.2.4 Bulk Transfer and Interrupt Transfer ..... 25
6.2.5 Data Flow ..... 25
6.2.5.1 OUT Transfer ..... 25
6.2.5.2 IN Transfer ..... 26
6.2.6 Bulk Only Support ..... 27
6.2.6.1 CBW Support ..... 27
6.2.6.2 CSW Support ..... 28
6.2.7 Auto Negotiation Function ..... 29
6.2.7.1 DISABLE ..... 30
6.2.7.2 IDLE ..... 30
6.2.7.3 WAIT_TIM3US ..... 30
6.2.7.4 WAIT_CHIRP ..... 30
6.2.7.5 WAIT_RSTEND ..... 31
6.2.7.6 DET_SUSPEND ..... 31
6.2.7.7 IN_SUSPEND ..... 31
6.2.7.8 CHK_EVENT ..... 31
6.2.7.9 WAIT_RESTORE ..... 31
6.2.7.10 ERR ..... 32
6.2.7.11 Individual Description of Each Negotiation Function ..... 32
6.2.7.11.1 Suspend Detection (HS Mode) ..... 32
6.2.7.11.2 Suspend Detection (FS Mode) ..... 34
6.2.7.11.3 Reset Detection (HS Mode) ..... 36
6.2.7.11.4 Reset Detection (FS Mode) ..... 37
6.2.7.11.5 HS Detection Handshaking ..... 38
6.2.7.11.5.1 When Connected to an FS Downstream Port ..... 39
6.2.7.11.5.2 When Connected to an HS Downstream Port ..... 41
6.2.7.11.5.3 When Reset during Sleep ..... 43
6.2.7.11.6 Issuance of Resume ..... 45
6.2.7.11.7 Detection of Resume ..... 47
6.2.7.11.8 Insertion of USB Cable ..... 49
6.3 USB Host Control ..... 51
6.3.1 Channels ..... 51
6.3.1.1 Channel Overview ..... 51
6.3.1.2 Control-only Channel ..... 53
6.3.1.3 General-purpose Channels ..... 54
6.3.1.4 Example for Using Channels ..... 56
6.3.1.4.1 For One Storage Device Connected ..... 56

6.3.1.4.2 Connecting a Storage Device via a Hub ..... 57
6.3.2 Scheduling ..... 58
6.3.3 Transactions ..... 58
6.3.3.1 SETUP Transactions ..... 59
6.3.3.2 Bulk OUT Transaction ..... 60
6.3.3.3 Interrupt OUT Transaction ..... 61
6.3.3.4 Bulk IN Transaction ..... 62
6.3.3.5 Interrupt IN Transaction ..... 63
6.3.3.6 PING Transaction ..... 64
6.3.3.7 Low-Speed (LS) Transaction ..... 65
6.3.3.8 Split Transactions ..... 67
6.3.4 Control Transfer ..... 68
6.3.4.1 Setup Stage ..... 69
6.3.4.2 Data Stage and Status Stage ..... 69
6.3.4.3 Control Transfer Support Function ..... 70
6.3.5 Bulk and Interrupt Transfers ..... 73
6.3.6 Data Flow ..... 73
6.3.6.1 OUT Transfer ..... 73
6.3.6.2 IN Transfer ..... 74
6.3.7 Zero-length Packet Auto Issue Function ..... 75
6.3.7.1 Zero-length Packet Auto Issue Function in Bulk/Interrupt OUT Transfers ..... 75
6.3.8 Bulk Only Support Function ..... 75
6.3.9 Host State Management Support Function ..... 80
6.3.9.1 Host States ..... 80
6.3.9.1.1 IDLE ..... 82
6.3.9.1.2 WAIT_CONNECT ..... 82
6.3.9.1.3 DISABLED ..... 83
6.3.9.1.4 RESET ..... 83
6.3.9.1.5 OPERATIONAL ..... 85
6.3.9.1.6 SUSPEND ..... 85
6.3.9.1.7 RESUME ..... 86
6.3.9.2 Detection Functions ..... 86
6.3.9.2.1 VBUS Error Detection ..... 86
6.3.9.2.2 Disconnection Detection ..... 88
6.3.9.2.2.1 When HS Device is Disconnected ..... 88
6.3.9.2.2.2 When FS or LS Device is Disconnected ..... 89
6.3.9.2.3 Remote Wakeup Detection ..... 90
6.3.9.2.3.1 When HS Device is Connected ..... 90

6.3.9.2.3.2 When FS Device is Connected ..... 92
6.3.9.2.3.3 When LS Device is Connected ..... 93
6.3.9.2.4 Device Chirp Detection Function ..... 94
6.3.9.2.4.1 When a Correct Device Chirp is Detected ..... 94
6.3.9.2.4.2 When an Erratic Device Chirp is Detected ..... 96
6.3.9.2.5 Port Error Detection ..... 97
6.3.9.3 Description of Individual Host State Management Support Function ..... 97
6.3.9.3.1 GoIDLE ..... 97
6.3.9.3.2 GoWAIT_CONNECT ..... 99
6.3.9.3.2.1 When FS Device is Connected ..... 99
6.3.9.3.2.2 When LS Device is Connected ..... 101
6.3.9.3.3 GoDISABLED ..... 103
6.3.9.3.3.1 When HS Device is Connected ..... 103
6.3.9.3.3.2 When FS Device is Connected ..... 105
6.3.9.3.3.3 When LS Device is Connected ..... 106
6.3.9.3.4 GoRESET ..... 107
6.3.9.3.4.1 Reset for an HS Device ..... 107
6.3.9.3.4.2 Erratic Device Chirp Detected ..... 109
6.3.9.3.4.2.1 When Chirp Complete Disable (H_NegoControl_1.DisChirpFinish) = 0 ..... 109
6.3.9.3.4.2.2 When Chirp Complete Disable (H_NegoControl_1.DisChirpFinish) = 1 ..... 111
6.3.9.3.4.3 Reset for an FS Device ..... 114
6.3.9.3.4.4 Reset for an LS Device ..... 116
6.3.9.3.5 GoOPERATIONAL ..... 117
6.3.9.3.6 GoSUSPEND ..... 118
6.3.9.3.6.1 When HS Device is Connected ..... 118
6.3.9.3.6.2 When FS Device is Connected ..... 120
6.3.9.3.6.3 When LS Device is Connected ..... 122
6.3.9.3.7 GoRESUME ..... 124
6.3.9.3.7.1 When HS Device is Connected ..... 124
6.3.9.3.7.2 When FS Device is Connected ..... 126
6.3.9.3.7.3 When LS Device is Connected ..... 128
6.3.9.3.8 GoWAIT_CONNECTtoDIS ..... 130
6.3.9.3.9 GoWAIT_CONNECTtoOP ..... 131
6.3.9.3.9.1 When HS Device is Connected ..... 131
6.3.9.3.9.2 When FS or LS Device is Connected ..... 133
6.3.9.3.10 GoRESETtoOP ..... 135
6.3.9.3.10.1 When HS Device is Connected ..... 135
6.3.9.3.10.2 When FS or LS Device is Connected ..... 136

6.3.9.3.11 GoSUSPENDtoOP ..... 137
6.3.9.3.12 GoRESUMEtoOP ..... 139
6.4 Power Management Function ..... 140
6.4.1 SLEEP (Sleep) ..... 141
6.4.2 SNOOZE (Snooze) ..... 142
6.4.3 ACTIVE (Active) ..... 142
6.4.4 CPU_Cut Mode ..... 142
6.5 FIFO Management ..... 143
6.5.1 FIFO Memory Map ..... 143
6.5.2 Descriptor Area ..... 144
6.5.2.1 Writing Data into the Descriptor Area ..... 145
6.5.2.2 Executing a Data Stage (IN) in the Descriptor Area ..... 145
6.5.3 CBW Area ..... 146
6.5.3.1 CBW Area (during USB Device Mode) ..... 146
6.5.3.2 CBW Area (during USB Host Mode) ..... 146
6.5.4 CSW Area ..... 147
6.5.4.1 CSW Area (during USB Device Mode) ..... 147
6.5.4.2 CSW Area (during USB Host Mode) ..... 147
6.5.5 Method for Accessing the FIFO ..... 147
6.5.5.1 Method for Accessing the FIFO (RAM_Rd) ..... 147
6.5.5.2 Method for Accessing the FIFO (RAM_WrDoor) ..... 148
6.5.5.3 Method for Accessing the FIFO (Register Access) ..... 148
6.5.5.4 Method for Accessing the FIFO (DMA) ..... 148
6.5.5.5 Limitations on FIFO Access ..... 149
6.6 CPUIF ..... 150
6.6.1 Mode Switching ..... 150
6.6.2. Notes on Mode Switchover ..... 150
6.6.2.1. When Using 16-bit BE Mode ..... 150
6.6.2.2. When Using 8-bit Mode ..... 152
6.6.3 Block Configuration ..... 153
6.6.3.1 REG (S1R72V17 Registers) ..... 153
6.6.3.1.1 Synchronous Register Access (Write) ..... 153
6.6.3.1.2 Synchronous Register Access (Read) ..... 153
6.6.3.1.3 FIFO Access (Write) ..... 154
6.6.3.1.4 FIFO Access (Read) ..... 154
6.6.2.1.5 Processing Odd Bytes in FIFO Access ..... 155
6.6.2.1.6 RAM_Rd Access ..... 158
6.6.2.1.7 Asynchronous Register Access (Write) ..... 158

6.6.2.1.8 Asynchronous Register Access (Read) ..... 158
6.6.2.2 DMA (DMA Channel) ..... 158
6.6.2.2.1 Basic Functionality ..... 158
6.6.2.2.2 Pin Settings ..... 160
6.6.2.2.3 Count Mode (Write) ..... 160
6.6.2.2.4 Count Mode (Read) ..... 162
6.6.2.2.5 Free-running Mode (Write) ..... 163
6.6.2.2.6 Free-running Mode (Read) ..... 164
6.6.2.2.7 REQ Assert Count Option (Write) ..... 164
6.6.2.2.8 REQ Assert Count Option (Read) ..... 166
6.6.2.2.9 FIFO Access Odd Bytes Processing in DMA ..... 166
**7. Registers ..... 167**
7.1 Device/Host Shared Register Map ..... 167
7.2 Device Register Map ..... 173
7.3 Host Register Map ..... 177
7.4 Detailed Description of Device/Host Shared Registers ..... 182
7.4.1 000h *MainIntStat (Main Interrupt Status)* ..... 182
7.4.2 001h *USB_DeviceIntStat (USB Device Interrupt Status)* ..... 184
7.4.3 002h *USB_HostIntStat (USB Host Interrupt Status)* ..... 186
7.4.4 003h CPU_IntStat (CPU Interrupt Status) ..... 188
7.4.5 004h FIFO_IntStat (FIFO Interrupt Status) ..... 189
7.4.6 008h *MainIntEnb (Main Interrupt Enable)* ..... 190
7.4.7 009h *USB_DeviceIntEnb (Device Interrupt Enable)* ..... 191
7.4.8 00Ah *USB_HostIntEnb (Host Interrupt Enable)* ..... 192
7.4.9 00Bh CPU_IntEnb (CPU Interrupt Enable) ..... 193
7.4.10 00Ch FIFO_IntEnb (FIFO Interrupt Enable) ..... 194
7.4.11 010h *RevisionNum (Revision Number)* ..... 195
7.4.12 011h *ChipReset (Chip Reset)* ..... 196
7.4.13 012h *PM_Control (Power Management Control)* ..... 197
7.4.14 014h *WakeupTim_H (Wakeup Time High)* ..... 199
7.4.15 015h *WakeupTim_L (Wakeup Time Low)* ..... 199
7.4.16 016h *H_USB_Control (Host USB Control )* ..... 200
7.4.17 017h *H_XcvrControl (Host Xcvr Control)* ..... 201
7.4.18 018h *D_USB_Status (Device USB Status)* ..... 203
7.4.19 019h *H_USB_Status (Host USB Status)* ..... 204
7.4.20 01Bh MTM_Config (Multi Transceiver Macro Config) ..... 205
7.4.21 01Fh *HostDeviceSel (Host Device Select)* ..... 206
7.4.22 020h FIFO_Rd_0 (FIFO Read 0) ..... 207

7.4.23 021h FIFO_Rd_1 (FIFO Read 1) ..... 207
7.4.24 022h FIFO_Wr_0(FIFO Write 0) ..... 208
7.4.25 023h FIFO_Wr_1(FIFO Write 1) ..... 208
7.4.26 024h FIFO_RdRemain_H (FIFO Read Remain High) ..... 209
7.4.27 025h FIFO_RdRemain_L (FIFO Read Remain Low) ..... 209
7.4.28 026h FIFO_WrRemain_H (FIFO Write Remain High) ..... 210
7.4.29 027h FIFO_WrRemain_L (FIFO Write Remain Low) ..... 210
7.4.30 028h FIFO_ByteRd(FIFO Byte Read) ..... 211
7.4.31 030h RAM_RdAdrs_H (RAM Read Address High) ..... 212
7.4.32 031h RAM_RdAdrs_L (RAM Read Address Low) ..... 212
7.4.33 032h RAM_RdControl (RAM Read Control) ..... 213
7.4.34 035h RAM_RdCount (RAM Read Counter) ..... 214
7.4.35 038h RAM_WrAdrs_H (RAM Write Address High) ..... 215
7.4.36 039h RAM_WrAdrs_L (RAM Write Address Low) ..... 215
7.4.37 03Ah RAM_WrDoor_0 (RAM Write Door 0) ..... 216
7.4.38 03Bh RAM_WrDoor_1 (RAM Write Door 1) ..... 216
7.4.39 040h RAM_Rd_00 (RAM Read 00) ..... 217
7.4.40 041h RAM_Rd_01 (RAM Read 01) ..... 217
7.4.41 042h RAM_Rd_02 (RAM Read 02) ..... 217
7.4.42 043h RAM_Rd_03 (RAM Read 03) ..... 217
7.4.43 044h RAM_Rd_04 (RAM Read 04) ..... 217
7.4.44 045h RAM_Rd_05 (RAM Read 05) ..... 217
7.4.45 046h RAM_Rd_06 (RAM Read 06) ..... 217
7.4.46 047h RAM_Rd_07 (RAM Read 07) ..... 217
7.4.47 048h RAM_Rd_08 (RAM Read 08) ..... 217
7.4.48 049h RAM_Rd_09 (RAM Read 09) ..... 217
7.4.49 04Ah RAM_Rd_0A (RAM Read 0A) ..... 217
7.4.50 04Bh RAM_Rd_0B (RAM Read 0B) ..... 217
7.4.51 04Ch RAM_Rd_0C (RAM Read 0C) ..... 217
7.4.52 04Dh RAM_Rd_0D (RAM Read 0D) ..... 217
7.4.53 04Eh RAM_Rd_0E (RAM Read 0E) ..... 217
7.4.54 04Fh RAM_Rd_0F (RAM Read 0F) ..... 217
7.4.55 050h RAM_Rd_10 (RAM Read 10) ..... 217
7.4.56 051h RAM_Rd_11 (RAM Read 11) ..... 217
7.4.57 052h RAM_Rd_12 (RAM Read 12) ..... 217
7.4.58 053h RAM_Rd_13 (RAM Read 13) ..... 217
7.4.59 054h RAM_Rd_14 (RAM Read 14) ..... 217
7.4.60 055h RAM_Rd_15 (RAM Read 15) ..... 217

7.4.61 056h RAM_Rd_16 (RAM Read 16) ..... 217
7.4.62 057h RAM_Rd_17 (RAM Read 17) ..... 217
7.4.63 058h RAM_Rd_18 (RAM Read 18) ..... 217
7.4.64 059h RAM_Rd_19 (RAM Read 19) ..... 217
7.4.65 05Ah RAM_Rd_1A (RAM Read 1A) ..... 217
7.4.66 05Bh RAM_Rd_1B (RAM Read 1B) ..... 217
7.4.67 05Ch RAM_Rd_1C (RAM Read 1C) ..... 217
7.4.68 05Dh RAM_Rd_1D (RAM Read 1D) ..... 217
7.4.69 05Eh RAM_Rd_1E (RAM Read 1E) ..... 217
7.4.70 05Fh RAM_Rd_1F (RAM Read 1F) ..... 218
7.4.71 061h DMA_Config (DMA Config) ..... 219
7.4.72 062h DMA_Control (DMA Control) ..... 221
7.4.73 064h DMA_Remain_H (DMA FIFO Remain High) ..... 222
7.4.74 065h DMA_Remain_L (DMA0 FIFO Remain Low) ..... 222
7.4.75 068h DMA_Count_HH (DMA Transfer Byte Counter High/High) ..... 223
7.4.76 069h DMA_Count_HL (DMA Transfer Byte Counter High/Low) ..... 223
7.4.77 06Ah DMA_Count_LH (DMA Transfer Byte Counter Low/High) ..... 223
7.4.78 06Bh DMA_Count_LL (DMA Transfer Byte Counter Low/Low) ..... 223
7.4.79 06Ch DMA_RdData_0 (DMA Read Data 0) ..... 225
7.4.80 06Dh DMA_RdData_1 (DMA Read Data 1) ..... 225
7.4.81 06Eh DMA_WrData_0 (DMA Write Data 0) ..... 226
7.4.82 06Fh DMA_WrData_1 (DMA Write Data 1) ..... 226
7.4.83 071h *ModeProtect(Mode Protection)* ..... 227
7.4.84 073h *ClkSelect (Clock Select)* ..... 228
7.4.85 075h *CPU_Config (CPU Configuration)* ..... 229
7.4.86 077h *CPU_ChgEndian(CPU Change Endian)* ..... 231
7.4.87 080h AREA0StartAdrs_H (AREA0 Start Address High) ..... 232
7.4.88 081h AREA0StartAdrs_L (AREA0 Start Address Low) ..... 232
7.4.89 082h AREA0EndAdrs_H (AREA0 End Address High) ..... 233
7.4.90 083h AREA0EndAdrs_L (AREA0 End Address Low) ..... 233
7.4.91 084h AREA1StartAdrs_H (AREA1 Start Address High) ..... 234
7.4.92 085h AREA1StartAdrs_L (AREA1 Start Address Low) ..... 234
7.4.93 086h AREA1EndAdrs_H (AREA1 End Address High) ..... 235
7.4.94 087h AREA1EndAdrs_L (AREA1 End Address Low) ..... 235
7.4.95 088h AREA2StartAdrs_H (AREA2 Start Address High) ..... 236
7.4.96 089h AREA2StartAdrs_L (AREA2 Start Address Low) ..... 236
7.4.97 08Ah AREA2EndAdrs_H (AREA2 End Address High) ..... 237
7.4.98 08Bh AREA2EndAdrs_L (AREA2 End Address Low) ..... 237

7.4.99 08Ch AREA3StartAdrs_H (AREA3 Start Address High) ..... 238
7.4.100 08Dh AREA3StartAdrs_L (AREA3 Start Address Low) ..... 238
7.4.101 08Eh AREA3EndAdrs_H (AREA3 End Address High) ..... 239
7.4.102 08Fh AREA3EndAdrs_L (AREA3 End Address Low) ..... 239
7.4.103 090h AREA4StartAdrs_H (AREA4 Start Address High) ..... 240
7.4.104 091h AREA4StartAdrs_L (AREA4 Start Address Low) ..... 240
7.4.105 092h AREA4EndAdrs_H (AREA4 End Address High) ..... 241
7.4.106 093h AREA4EndAdrs_L (AREA4 End Address Low) ..... 241
7.4.107 094h AREA5StartAdrs_H (AREA5 Start Address High) ..... 242
7.4.108 095h AREA5StartAdrs_L (AREA5 Start Address Low) ..... 242
7.4.109 096h AREA5EndAdrs_H (AREA5 End Address High) ..... 243
7.4.110 097h AREA5EndAdrs_L (AREA5 End Address Low) ..... 243
7.4.111 09Fh AREAnFIFO_Clr (AREAn FIFO Clear) ..... 244
7.4.112 0A0h AREA0Join_0 (AREA0 Join 0) ..... 245
7.4.113 0A1h AREA0Join_1 (AREA0 Join 1) ..... 246
7.4.114 0A2h AREA1Join_0 (AREA1 Join 0) ..... 248
7.4.115 0A3h AREA1Join_1 (AREA1 Join 1) ..... 249
7.4.116 0A4h AREA2Join_0 (AREA2 Join 0) ..... 251
7.4.117 0A5h AREA2Join_1 (AREA2 Join 1) ..... 252
7.4.118 0A6h AREA3Join_0 (AREA3 Join 0) ..... 254
7.4.119 0A7h AREA3Join_1 (AREA3 Join 1) ..... 255
7.4.120 0A8h AREA4Join_0 (AREA4 Join 0) ..... 257
7.4.121 0A9h AREA4Join_1 (AREA4 Join 1) ..... 258
7.4.122 0AAh AREA5Join_0 (AREA5 Join 0) ..... 260
7.4.123 0ABh AREA5Join_1 (AREA5 Join 1) ..... 261
7.4.124 0AEh ClrAREAnJoin_0 (Clear AREA n Join 0) ..... 263
7.4.125 0AFh ClrAREAnJoin_1 (Clear AREA n Join 1) ..... 264
7.5 Detailed Description of Device Registers ..... 265
7.5.1 0Bh *D_SIE_IntStat (Device SIE Interrupt Status)* ..... 265
7.5.2 0Bh D_BulkIntStat (Device Bulk Interrupt Status) ..... 267
7.5.3 0Bh D_EPrIntStat (Device EPr Interrupt Status) ..... 268
7.5.4 0B5h D_EP0IntStat (Device EP0 Interrupt Status) ..... 270
7.5.5 0B6h D_EPaIntStat (Device EPa Interrupt Status) ..... 272
7.5.6 0B7h D_EPbIntStat (Device EPb Interrupt Status) ..... 274
7.5.7 0B8h D_EPcIntStat (D_EPc Interrupt Status) ..... 276
7.5.8 0B9h D_EPdIntStat (D_EPd Interrupt Status) ..... 278
7.5.9 0BAh D_EPeIntStat (D_EPe Interrupt Status) ..... 280
7.5.10 0BCh D_AlarmIN_IntStat_H (Device AlarmIN Interrupt Status High) ..... 282

7.5.11 0BDh D_AlarmIN_IntStat_L (Device AlarmIN Interrupt Status Low) ..... 282
7.5.12 0BEh D_AlarmOUT_IntStat_H (Device AlarmOUT Interrupt Status High) ..... 283
7.5.13 0BFh D_AlarmOUT_IntStat_L (Device AlarmOUT Interrupt Status Low) ..... 283
7.5.14 0C0h *D_SIE_IntEnb (Device SIE Interrupt Enable)* ..... 284
7.5.15 0C3h D_BulkIntEnb (Device Bulk Interrupt Enable) ..... 285
7.5.16 0C4h D_EPrIntEnb (Device EPr Interrupt Enable) ..... 286
7.5.17 0C5h D_EP0IntEnb (Device EP0 Interrupt Enable) ..... 287
7.5.18 0C6h D_EPaIntEnb (DeviceEPa Interrupt Enable) ..... 288
7.5.19 0C7h D_EPbIntEnb (Device EPb Interrupt Enable) ..... 289
7.5.20 C8h D_EPcIntEnb (Device EPc Interrupt Enable) ..... 290
7.5.21 0C9h D_EPdIntEnb (Device EPd Interrupt Enable) ..... 291
7.5.22 0CAh D_EPeIntEnb (Device EPe Interrupt Enable) ..... 292
7.5.23 0CCh D_AlarmIN_IntEnb_H (Device AlarmIN Interrupt Enable High) ..... 293
7.5.24 0CDh D_AlarmIN_IntEnb_L (Device AlarmIN Interrupt Enable Low) ..... 293
7.5.25 0CEh D_AlarmOUT_IntEnb_H (Device AlarmOUT Interrupt Enable High) ..... 294
7.5.26 0CFh D_AlarmOUT_IntEnb_L (Device AlarmOUT Interrupt Enable Low) ..... 294
7.5.27 0D0h D_NegoControl (Device Nego Control) ..... 295
7.5.28 0D3h D_XcvrControl (Device Xcvr Control) ..... 297
7.5.29 0D4h D_USB_Test (Device USB_Test) ..... 298
7.5.30 0D6h D_EPnControl (Device Endpoint Control) ..... 300
7.5.31 0D8h D_BulkOnlyControl (Device BulkOnly Control) ..... 301
7.5.32 0D9h D_BulkOnlyConfig (Device BulkOnly Configuration) ..... 302
7.5.33 0E0h D_EP0SETUP_0 (Device EP0 SETUP 0) ..... 304
7.5.34 0E1h D_EP0SETUP_1 (Device EP0 SETUP 1) ..... 304
7.5.35 0E2h D_EP0SETUP_2 (Device EP0 SETUP 2) ..... 304
7.5.36 0E3h D_EP0SETUP_3 (Device EP0 SETUP 3) ..... 304
7.5.37 0E4h D_EP0SETUP_4 (Device EP0 SETUP 4) ..... 304
7.5.38 0E5h D_EP0SETUP_5 (Device EP0 SETUP 5) ..... 304
7.5.39 0E6h D_EP0SETUP_6 (Device EP0 SETUP 6) ..... 304
7.5.40 0E7h D_EP0SETUP_7 (Device EP0 SETUP 7) ..... 304
7.5.41 0E8h D_USB_Address (Device USB Address) ..... 305
7.5.42 0EAh D_SETUP_Control(Device SETUP Control) ..... 306
7.5.43 0EEh D_FrameNumber_H (Device FrameNumber High) ..... 307
7.5.44 0EFh D_FrameNumber_L (Device FrameNumber Low) ..... 307
7.5.45 0F0h D_EP0MaxSize (Device EP0 Max Packet Size) ..... 308
7.5.46 0F1h D_EP0Control (Device EP0 Control) ..... 309
7.5.47 0F2h D_EP0ControlIN (Device EP0 Control IN) ..... 311
7.5.48 0F3h D_EP0ControlOUT (Device EP0 Control OUT) ..... 313

7.5.49 0F8h D_EPaMaxSize_H (Device EPa Max Packet Size High) ..... 315
7.5.50 0F9h D_EPaMaxSize_L (Device EPa Max Packet Size Low) ..... 315
7.5.51 0FAh D_EPaConfig (Device EPa Configuration) ..... 316
7.5.52 0FCh D_EPaControl (Device EPa Control) ..... 318
7.5.53 100h D_EPbMaxSize_H (Device EPb Max Packet Size High) ..... 320
7.5.54 101h D_EPbMaxSize_L (Device EPb Max Packet Size Low) ..... 320
7.5.55 102h D_EPbConfig (Devie EPb Configuration) ..... 321
7.5.56 104h D_EPbControl (Device EPb Control) ..... 323
7.5.57 108h D_EPcMaxSize_H (Device EPc Max Packet Size High) ..... 325
7.5.58 109h D_EPcMaxSize_L (Device EPc Max Packet Size Low) ..... 325
7.5.59 10Ah D_EPcConfig (Device EPc Configuration) ..... 326
7.5.60 10Ch D_EPcControl (Device EPc Control) ..... 328
7.5.61 110h D_EPdMaxSize_H (Device EPd Max Packet Size High) ..... 330
7.5.62 111h D_EPdMaxSize_L (Device EPd Max Packet Size Low) ..... 330
7.5.63 112h D_EPdConfig (Device EPd Configuration) ..... 331
7.5.64 114h D_EPdControl (Device EPd Control) ..... 333
7.5.65 118h D_EPeMaxSize_H (Device EPe Max Packet Size High) ..... 335
7.5.66 119h D_EPeMaxSize_L (Device EPe Max Packet Size Low) ..... 335
7.5.67 11Ah D_EPeConfig (Device EPe Configuration) ..... 336
7.5.68 11Ch D_EPeControl (Device EPe Control) ..... 338
7.5.69 120h D_DescAdrs_H (Device Descriptor Address High) ..... 340
7.5.70 121h D_DescAdrs_L (Device Descriptor Address Low) ..... 340
7.5.71 122h D_DescSize_H (Device Descriptor Size High) ..... 341
7.5.72 123h D_DescSize_L (Device Descriptor Size Low) ..... 341
7.5.73 126h D_EP_DMA_Ctrl (Device EP DMA Control) ..... 342
7.5.74 128h D_EnEP_IN_H (Device Enable Endpoint-IN High) ..... 343
7.5.75 129h D_EnEP_IN_L (Device Enable Endpoint-IN Low) ..... 343
7.5.76 12Ah D_EnEP_OUT_H (Device Enable Endpoint-IN High) ..... 344
7.5.77 12Bh D_EnEP_OUT_L (Device Enable Endpoint-IN Low) ..... 344
7.5.78 12Ch D_EnEP_IN_H (Device Enable Endpoint-IN High) ..... 345
7.5.79 12Dh D_EnEP_IN_L (Device Enable Endpoint-IN Low) ..... 345
7.5.80 12Eh D_EnEP_OUT_ISO_H (Device Enable Endpoint-OUT Isochronous High) ..... 346
7.5.81 12Fh D_EnEP_OUT_ISO_L (Device Enable Endpoint-OUT Isochronous Low) ..... 346
7.6 Detailed Description of Host Registers ..... 347
7.6.1 140h H_SIE_IntStat_0(Host SIE Interrupt Status 0) ..... 347
7.6.2 141h H_SIE_IntStat_1(SIE Host Interrupt Status 1) ..... 349
7.6.3 143h H_FrameIntStat(Host Frame Interrupt Status ) ..... 351
7.6.4 144h H_CHrIntStat (Host CHr Interrupt Status) ..... 352

7.6.5 145h H_CH0IntStat (Host CH0 Interrupt Status) ..... 353
7.6.6 146h H_CHaIntStat (Host CHa Interrupt Status) ..... 356
7.6.7 147h H_CHbIntStat (Host CHb Interrupt Status) ..... 359
7.6.8 148h H_CHcIntStat (Host CHc Interrupt Status) ..... 361
7.6.9 149h H_CHdIntStat (Host CHd Interrupt Status) ..... 363
7.6.10 14Ah H_CHeIntStat (Host CHe Interrupt Status) ..... 365
7.6.11 150h H_SIE_IntEnb_0 (Host SIE Interrupt Enable) ..... 367
7.6.12 151h H_SIE_IntEnb_1(SIE Host Interrupt Enable 1) ..... 368
7.6.13 152h Reserved ..... 369
7.6.14 153h H_FrameIntEnb(Host Frame Interrupt Enable ) ..... 370
7.6.15 154h H_CHrIntEnb(Host CHr Interrupt Enable) ..... 371
7.6.16 155h H_CH0IntEnb(Host CH0 Interrupt Enable) ..... 372
7.6.17 156h H_CHaIntEnb (Host CHa Interrupt Enable) ..... 373
7.6.18 157h H_CHbIntEnb (Host CHb Interrupt Enable) ..... 374
7.6.19 158h H_CHcIntEnb (Host CHc Interrupt Enable) ..... 375
7.6.20 159h H_CHdIntEnb (Host CHd Interrupt Enable) ..... 376
7.6.21 15Ah H_CHeIntEnb (Host CHe Interrupt Enable) ..... 377
7.6.22 160h H_NegoControl_0 (Host NegoControl 0) ..... 378
7.6.23 162h H_NegoControl_1 (Host NegoControl 1) ..... 380
7.6.24 164h H_USB_Test (Host USB_Test) ..... 381
7.6.25 170h H_CH0SETUP_0 (Host CH0 SETUP 0) ..... 383
7.6.26 171h H_CH0SETUP_1 (Host CH0 SETUP 1) ..... 383
7.6.27 172h H_CH0SETUP_2 (Host CH0 SETUP 2) ..... 383
7.6.28 173h H_CH0SETUP_3 (Host CH0 SETUP 3) ..... 383
7.6.29 174h H_CH0SETUP_4 (Host CH0 SETUP 4) ..... 383
7.6.30 175h H_CH0SETUP_5 (Host CH0 SETUP 5) ..... 383
7.6.31 176h H_CH0SETUP_6 (Host CH0 SETUP 6) ..... 383
7.6.32 177h H_CH0SETUP_7 (Host CH0 SETUP 7) ..... 383
7.6.33 17Eh H_FrameNumber_H (Host FrameNumber High) ..... 384
7.6.34 17Fh H_FrameNumber_L (Host FrameNumber Low) ..... 384
7.6.35 180h H_CH0Config_0(Host Channel 0 Configuration0) ..... 385
7.6.36 181h H_CH0Config_1(Host Channel 0 Configuration1) ..... 387
7.6.37 183h H_CH0MaxPktSize (Host Channel 0 Max Packet Size) ..... 388
7.6.38 186h H_CH0TotalSize_H (Host Channel 0 Total Size High) ..... 389
7.6.39 187h H_CH0TotalSize_L (Host Channel 0 Total Size Low) ..... 389
7.6.40 188h H_CH0HubAdrs (Host Channel 0 Hub Address) ..... 390
7.6.41 189h H_CH0FuncAdrs (Host Channel 0 Function Address) ..... 391
7.6.42 18Bh CTL_SupportControl (Host ControlTransfer Support Control) ..... 392

7.6.43 18Eh H_CH0ConditionCode (Host Channel 0 Condition Code) ..... 394
7.6.44 190h H_CHaConfig_0(Host Channel a Configuration0) ..... 396
7.6.45 191h H_CHaConfig_1(Host Channel a Configuration1) ..... 398
7.6.46 192h H_CHaMaxPktSize_H (Host Channel a Max Packet Size High) ..... 399
7.6.47 193h H_CHaMaxPktSize_L (Host Channel a Max Packet Size Low) ..... 399
7.6.48 194h H_CHaHubAdrs (Host Channel a Hub Address) ..... 400
7.6.49 195h H_CHaTotalSize_HL (Host Channel a Total Size High-Low) ..... 400
7.6.50 196h H_CHaTotalSize_LH (Host Channel a Total Size Low-High) ..... 400
7.6.51 197h H_CHaTotalSize_LL (Host Channel a Total Size Low-Low) ..... 400
7.6.52 198h H_CHaHubAdrs (Host Channel a Hub Address) ..... 402
7.6.53 199h H_CHaFuncAdrs (Host Channel a Function Address) ..... 403
7.6.54 19Ah H_CHaBO_SupporotCtl (Host CHa Bulk Only Transfer Supporot Control) ..... 404
7.6.55 19Bh H_CHaBO_CSW_RcvDataSize (Host CHa Bulk Only Transfer Support CSW Receive Data Size) ..... 406
7.6.56 19Ch H_ChaBQ_OUT_EP_Ctl (Host CHa Bulk Only Transfer Support OUT Endpoint Control) ..... 407
7.6.57 19Dh H_ChaBO_IN_EP_Ctl (Host CHa Bulk Only Transfer Support IN Endpoint Control) ..... 408
7.6.58 19Eh H_CHaConditionCode (Channel a Condition Code) ..... 409
7.6.59 1A0h H_CHbConfig_0(Host Channel b Configuration0) ..... 411
7.6.60 1A1h H_CHbConfig_1(Host Channel b Configuration1) ..... 413
7.6.61 1A2h H_CHbMaxPktSize_H (Host Channel b Max Packet Size High) ..... 414
7.6.62 1A3h H_CHbMaxPktSize_L (Host Channel b Max Packet Size Low) ..... 414
7.6.63 1A4h H_ChbTotalSize_HH (Host Channel b Total Size High-High) ..... 415
7.6.64 1A5h H_CHbTotalSize_HL (Host Channel b Total Size High-Low) ..... 415
7.6.65 1A6h H_CHbTotalSize_LH (Host Channel b Total Size Low-High) ..... 415
7.6.66 1A7h H_CHbTotalSize_LL (Host Channel b Total Size Low-Low) ..... 415
7.6.67 1A8h H_CHbHubAdrs (Host Channel b Hub Address) ..... 417
7.6.68 1A9h H_CHbFuncAdrs (Host Channel b Function Address) ..... 418
7.6.69 1AAh CHbInterval_H(Channel b Interval High) ..... 419
7.6.70 1ABh CHbInterval_L(Channel b Interval Low) ..... 419
7.6.71 1AEh H_CHbConditionCode (Host Channel b Condition Code) ..... 420
7.6.72 1B0h H_CHcConfig_0(Host Channel c Configuration0) ..... 422
7.6.73 1B1h H_CHcConfig_1(Host Channel c Configuration1) ..... 424
7.6.74 1B2h H_CHcMaxPktSize_H (Host Channel c Max Packet Size High) ..... 425
7.6.75 1B3h H_CHcMaxPktSize_L (Host Channel c Max Packet Size Low) ..... 425
7.6.76 1B4h H_CHcTotalSize_HH (Host Channel c Total Size High-High) ..... 426
7.6.77 1B5h H_CHcTotalSize_HL (Host Channel c Total Size High-Low) ..... 426
7.6.78 1B6h H_CHcTotalSize_LH (Host Channel c Total Size Low-High) ..... 426

7.6.79 1B7h H_CHcTotalSize_LL (Host Channel c Total Size Low-Low) ..... 426
7.6.80 1B8h H_CHcHubAdrs (Host Channel c Hub Address) ..... 428
7.6.81 1B9h H_CHcFuncAdrs (Host Channel c Function Address) ..... 429
7.6.82 1BAh H_CHcInterval_H(Host Channel c Interval High) ..... 430
7.6.83 1BBh H_CHcInterval_L(Host Channel c Interval Low) ..... 430
7.6.84 1BEh H_CHcConditionCode (Host Channel c Condition Code) ..... 431
7.6.85 1C0h H_CHdConfig_0(Host Channel d Configuration0) ..... 433
7.6.86 1C1h H_CHdConfig_1(Host Channel d Configuration1) ..... 435
7.6.87 1C2h H_CHdMaxPktSize_H (Host Channel d Max Packet Size High) ..... 436
7.6.88 1C3h H_CHdMaxPktSize_L (Host Channel d Max Packet Size Low) ..... 436
7.6.89 1C4h H_CHdTotalSize_HH (Host Channel d Total Size High-High) ..... 437
7.6.90 1C5h H_CHdTotalSize_HL (Host Channel d Total Size High-Low) ..... 437
7.6.91 1C6h H_CHdTotalSize_LH (Host Channel d Total Size Low-High) ..... 437
7.6.92 1C7h H_CHdTotalSize_LL (Host Channel d Total Size Low-Low) ..... 437
7.6.93 1C8h H_CHdHubAdrs (Host Channel d Hub Address) ..... 439
7.6.94 1C9h H_CHdFuncAdrs (Host Channel d Function Address) ..... 440
7.6.95 1CAh H_CHdInterval_H(Host Channel d Interval High) ..... 441
7.6.96 1CBh H_CHdInterval_L(Host Channel d Interval Low) ..... 441
7.6.97 1CEh H_CHdConditionCode (Host Channel d Condition Code) ..... 442
7.6.98 1D0h H_CHeConfig_0(Host Channel e Configuration0) ..... 444
7.6.99 1D1h H_CHeConfig_1(Host Channel e Configuration1) ..... 446
7.6.100 1D2h H_CHeMaxPktSize_H (Host Channel e Max Packet Size High) ..... 447
7.6.101 1D3h H_CHeMaxPktSize_L (Host Channel e Max Packet Size Low) ..... 447
7.6.102 1D4h H_CHeTotalSize_HH (Host Channel e Total Size High-High) ..... 448
7.6.103 1D5h H_CHeTotalSize_HL (Host Channel e Total Size High-Low) ..... 448
7.6.104 1D6h H_CHeTotalSize_LH (Host Channel e Total Size Low-High) ..... 448
7.6.105 1D7h H_CHeTotalSize_LL (Host Channel e Total Size Low-Low) ..... 448
7.6.106 1D8h H_CHeHubAdrs (Host Channel e Hub Address) ..... 450
7.6.107 1D9h H_CHeFuncAdrs (Host Channel e Function Address) ..... 451
7.6.108 1DAh H_CHeInterval_H(Host Channel e Interval High) ..... 452
7.6.109 1DBh H_CHeInterval_L(Host Channel e Interval Low) ..... 452
7.6.110 1DEh H_CHeConditionCode (Host Channel e Condition Code) ..... 453
**8. Electrical Characteristics ..... 455**
8.1 Absolute Maximum Ratings ..... 455
8.2 Recommended Operating Conditions ..... 455
8.3 D.C. Characteristics ..... 456
8.4 A.C. Characteristics ..... 459
8.4.1 RESET Timing ..... 459

8.4.2 Clock Timing ..... 459
8.4.3 CPU and DMA I/F Access Timing ..... 460
8.4.4 USB I/F Timing ..... 462
**9. Connection Examples ..... 463**
9.1 CPU I/F Connection Example ..... 463
9.2 USB I/F Connection Example ..... 464
9.2.1 For the PFBGA5UX60 ..... 464
9.2.2 For the PFBGA8UX81 ..... 465
9.2.3 For the QFP14-80 ..... 466
**10. Package Dimensions ..... 467**
10.1 PFBGA5UX60 ..... 467
10.2 PFBGA8UX81 ..... 468
10.3 QFP14-80 ..... 469
**Appendix A. Connecting to Little Endian CPUs ..... 470**
**Appendix B. Toggle Settings for Endpoint Changeover ..... 485**
**Appendix C. SUSPEND during HOST High-Speed Operation ..... 486**
**Appendix D. About Responses to a SetAddress Request ..... 490**
**Appendix E. Joining Endpoints/Channels to FIFO Areas ..... 493**

# 1. Overview

The S1R72V17 is a USB host/device controller LSI that supports USB2.0-compliant high-speed mode. The host ports and device ports of this LSI are shared, allowing it to operate as a USB host or as a USB device when control is switched over.

This LSI also has characteristics suitable for portable equipment incorporating a DMA interface.

# 2. Features

<< USB2.0 device functions >>

- Supports HS <480 Mbps) and FS (12 Mbps) transfers.
- Includes FS/HS termination (external circuit unnecessary).
- Includes VBUS 5V interface (external circuit unnecessary).
- Supports control, bulk, interrupt, and isochronous transfers.
- Supports five general-purpose (Bulk, Interrupt, and Isochronous transfer) endpoints and Endpoint 0.

<< USB2.0 host functions >>

- Supports HS (480 Mbps), FS (12 Mbps), and LS (1.5 Mbps) transfers.
- Includes pull-down resistors for downstream ports (external circuit unnecessary).
- Includes HS termination (external circuit unnecessary).
- Supports control, bulk, and interrupt transfers.
  Channel structure
  One (1) channel used exclusively for Control transfer
  Includes five general-purpose (Bulk, and Interrupt transfer) channels.
- USB power switch interface.

<< CPU interface >>

- Accepts 16-bit or 8-bit wide general-purpose CPU interfaces.
- Incorporates one DMA channel (Multiword transfer).
- Big Endian (incorporating a bus swap function for Little Endian CPUs).
- Changeable interface voltages (3.3 V to 1.8 VTyp).
- Supports CPU_Cut mode for reducing current consumption when the CPU is inactive.

<< Other >>

- Accepts a 12 MHz/24 MHz crystal resonator for clock input. (built-in Oscillator circuit and 1M Ωfeedback resistor)
- Dedicated pin for 12 MHz, 24 MHz, or 48 MHz clock input.
- Triple-power supply system: 3.3 V, 1.8 V and variable CPU interface power
- Package type: PFBGA5UX60 (S1R72V17B00A***)
  PFBGA8UX81 (S1R72V17B00B***)
  QFP14-80 (S1R72V17F00C***)
- Guaranteed operation temperature range: –40°C to 85°C

* This LSI is not designed to resist radiation.

# 3. Block Diagram

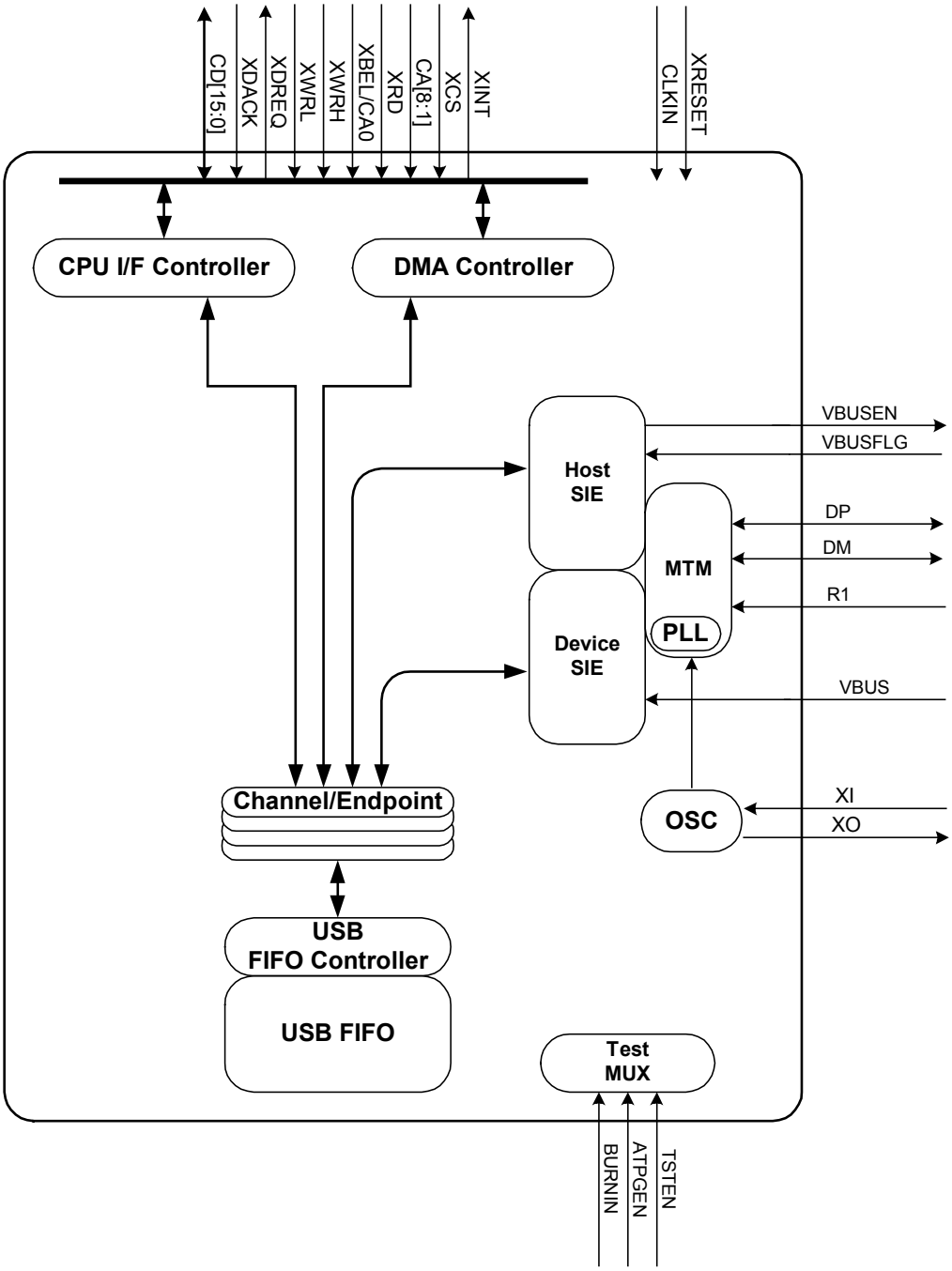



**Fig. 3.1 General block diagram**

## 3.1 Multi Transceiver Macro (MTM)

This USB 2.0 transceiver macro is shared by the USB host and USB device. Incorporating an analog circuit and high-speed logic circuit, it supports HS mode (480 Mbps) and FS mode (12 Mbps). LS mode (1.5 Mbps) is supported only for USB hosts.

Incorporates a transmitter, receiver, termination, etc. that together comprise a USB host/device interface.

Furthermore, it has a built-in PLL that generates a 480 MHz clock needed for HS transfer. Internal oscillator or incoming clock through CLKIN pin can be the clock source of the PLL.

## 3.2 Oscillator

The input clock accepts a 12 MHz or 24 MHz crystal resonator. A 1 MΩ feedback resistor is built-in.

## 3.3 Device Serial Interface Engine (Device SIE)

This block manages transactions and generates packets.

Furthermore, it controls bus events such as suspend, resume, and reset.

## 3.4 Host Serial Interface Engine (Host SIE)

This block schedules transactions, manages transactions, and generates packets.

Furthermore, it controls bus events such as suspend, resume, and reset.

It also detects connect/disconnect status and controls the VBUS (in cooperation with an external USB power switch).

## 3.5 FIFO and FIFO Controller

These blocks comprise a channel/endpoint buffer.

## 3.6 CPU I/F Controller

Controls the CPU interface timing, allowing registers to be accessed properly.

## 3.7 DMA Controller

Controls the DMA timing of the CPU interface, allowing access to FIFO. It incorporates one DMA channel.

## 3.8 Test MUX

This is a test circuit.

# 4. Pin Layout Diagram

S1R72V17B00A/PFBGA5UX60
TOP View

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | NC | LVDD | DP | DM | HVDD | R1 | LVDD | BURNIN |
| B | VBUSFLG | VSS | HVDD | VSS | VSS | VSS | VSS | XI |
| C | VBUSEN | HVDD | VBUS | CA1 | CA3 | CD15 | LVDD | XO |
| D | XRESET | XBEL | CA5 | | | CD13 | CVDD | CLKIN |
| E | CA2 | CA4 | XINT | | | CD4 | CD11 | CD14 |
| F | CA7 | CA8 | XWRH | XDACK | CD3 | CD7 | CD10 | CD12 |
| G | CA6 | LVDD | XRD | XDREQ | CD1 | CD6 | VSS | CD9 |
| H | TESTEN | XCS | XWRL | CD0 | CD2 | CD5 | CD8 | ATPGEN |

**Fig. 4.1 Pin Layout Diagram of the PFBGA5UX60 package**

S1R72V17B00B/PFBGA8UX81
TOP View

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| A | NC | LVDD | HVDD | DP | DM | HVDD | R1 | LVDD | NC |
| B | VSS | VSS | VBUS | VSS | VSS | VSS | VSS | VSS | XI |
| C | VBUSFLG | HVDD | LVDD | XBEL | CA1 | CVDD | BURNIN | LVDD | XO |
| D | XRESET | VBUSEN | CA3 | NC | NC | NC | CD12 | CD15 | CLKIN |
| E | CA2 | VSS | CA4 | NC | NC | NC | VSS | CD13 | CD14 |
| F | CVDD | CA5 | CA8 | NC | NC | NC | CD7 | CD9 | CD11 |
| G | CA7 | CA6 | TESTEN | XCS | XDACK | CD0 | CD4 | CD8 | CD10 |
| H | LVDD | XINT | XWRL | XRD | CD1 | CVDD | CD6 | ATPGEN | LVDD |
| J | NC | VSS | XWRH | XDREQ | CD2 | CD3 | CD5 | VSS | NC |

**Fig. 4.2 Pin Layout Diagram of the PFBGA8UX81 package**

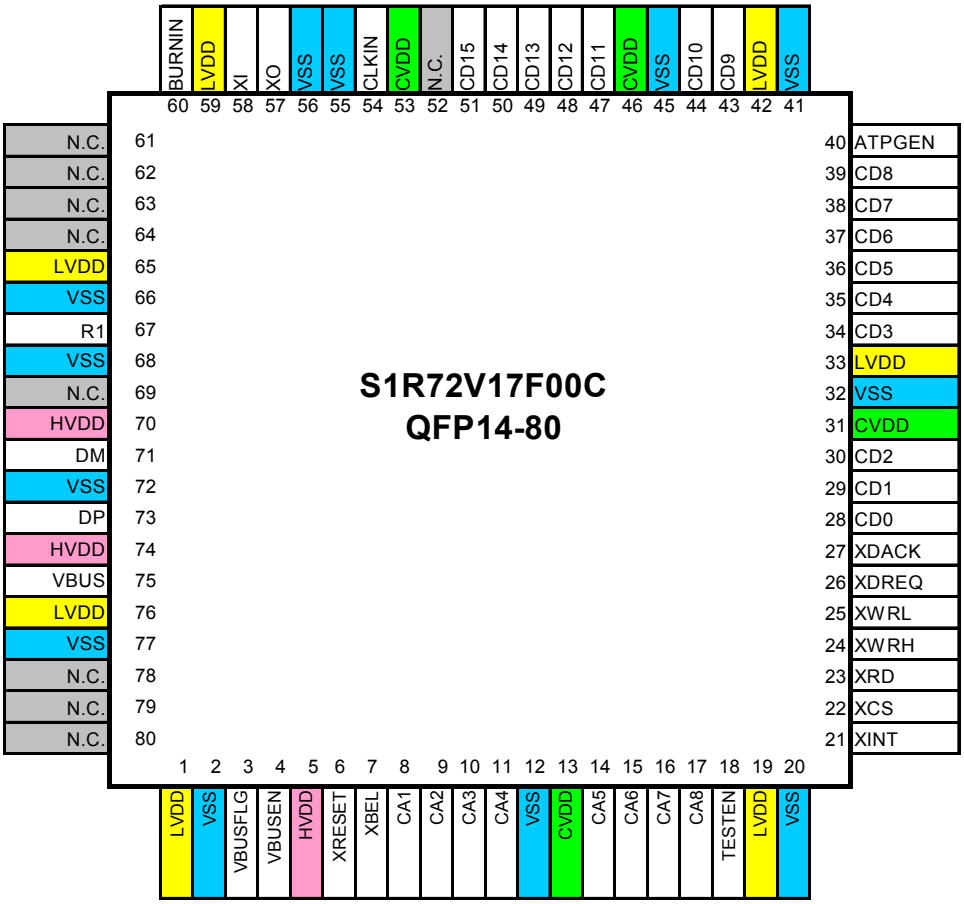



**Fig. 4.3 Pin Layout Diagram of the QFP14.80 package**

# 5. Pin Description

**OSC**

| QFP14 | PFBGA8 | PFBGA5 | Name | I/O | RESET | Pin Type | Pin Description |
|---|---|---|---|---|---|---|---|
| 58 | B9 | B8 | XI | IN | - | Analog | Input for the internal oscillator circuit<br>12 MHz/24 MHz |
| 57 | C9 | C8 | XO | OUT | - | Analog | Output for the internal oscillator circuit |

To use the internal oscillator of the LSI, connect a crystal resonator and oscillator circuit to the XI and XO pins and attach the CLKIN pin to the GND potential. To use an external clock by feeding it from the CLKIN pin, attach the XI pin to the GND potential and leave the XO pin open.

**TEST**

| QFP14 | PFBGA8 | PFBGA5 | Name | I/O | RESET | Pin Type | Pin Description |
|---|---|---|---|---|---|---|---|
| 18 | G3 | H1 | TESTEN | IN | (PD) | PD | Test pin (fixed low) |
| 40 | H8 | H8 | ATPGEN | IN | (PD) | PD | Test pin (fixed low) |
| 60 | C7 | A8 | BURNIN | IN | (PD) | PD | Test pin (fixed low) |

PD: Pull Down

PU: Pull Up

**USB**

| QFP14 | PFBGA8 | PFBGA5 | Name | I/O | RESET | Pin Type | Pin Description |
|---|---|---|---|---|---|---|---|
| 67 | A7 | A6 | R1 | IN | - | Analog | Internal operation setup pin<br>6.2 kΩ±1% resistor connected between this pin and VSS |
| 73 | A4 | A3 | DP | BI | Hi-Z | Analog | USB data line, Data+ |
| 71 | A5 | A4 | DM | BI | Hi-Z | Analog | USB data line, Data– |
| 3 | C1 | B1 | VBUSFLG | IN | (PU) | Schmitt PU | USB power switch fault detection signal<br>1: Normal; 0: Erratic |
| 4 | D2 | C1 | VBUSEN | OUT | Lo | 2mA | USB power switch control signal |
| 75 | B3 | C3 | VBUS | IN | (PD) | PD | USB device bus detection signal |

PD: Pull Down

PU: Pull Up

**CPU I/F**

| QFP14 | PFBGA8 | PFBGA5 | Name | I/O | RESET | Pin Type | Pin Description | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | **Bus Mode ⇒** | 16bit Strobe mode | 16bit BE mode | 8bit mode |
| 6 | D1 | D1 | XRESET | IN | - | Schmitt | Reset signal | | |
| 54 | D9 | D8 | CLKIN | IN | - | - | Clock input | | |
| 23 | H4 | G3 | XRD | IN | - | - | Read strobe | | |
| 25 | H3 | H3 | XWRL (XWR) | IN | - | - | Write strobe, lower | Write strobe | |
| 24 | J3 | F3 | XWRH (XBEH) | IN | - | - | Write strobe, upper | High byte enable | Fixed high |
| 22 | G4 | H2 | XCS | IN | - | Schmitt | Chip select signal | | |
| 21 | H2 | E3 | XINT | OUT | High | 2mA Tri-state | Interrupt output signal | | |
| 26 | J4 | G4 | XDREQ | OUT | High | 2mA | DMA request | | |
| 27 | G5 | F4 | XDACK | IN | - | - | DMA acknowledge | | |
| 7 | C4 | D2 | XBEL (CA0) | IN | - | - | Fixed high or low | Low byte enable | Address 0 |
| 8 | C5 | C4 | CA1 | IN | - | - | CPU bus address | | |
| 9 | E1 | E1 | CA2 | IN | - | - | | | |
| 10 | D3 | C5 | CA3 | IN | - | - | | | |
| 11 | E3 | E2 | CA4 | IN | - | - | | | |
| 14 | F2 | D3 | CA5 | IN | - | - | | | |
| 15 | G2 | G1 | CA6 | IN | - | - | | | |
| 16 | G1 | F1 | CA7 | IN | - | - | | | |
| 17 | F3 | F2 | CA8 | IN | - | - | | | |
| 28 | G6 | H4 | CD0 | BI | Hi-Z | 2mA | CPU data bus | | CPU data bus |
| 29 | H5 | G5 | CD1 | BI | Hi-Z | 2mA | | | |
| 30 | J5 | H5 | CD2 | BI | Hi-Z | 2mA | | | |
| 34 | J6 | F5 | CD3 | BI | Hi-Z | 2mA | | | |
| 35 | G7 | E6 | CD4 | BI | Hi-Z | 2mA | | | |
| 36 | J7 | H6 | CD5 | BI | Hi-Z | 2mA | | | |
| 37 | H7 | G6 | CD6 | BI | Hi-Z | 2mA | | | |
| 38 | F7 | F6 | CD7 | BI | Hi-Z | 2mA | | | |
| 39 | G8 | H7 | CD8 | BI | Hi-Z | 2mA | | | Pull Up or Pull Down |
| 43 | F8 | G8 | CD9 | BI | Hi-Z | 2mA | | | |
| 44 | G9 | F7 | CD10 | BI | Hi-Z | 2mA | | | |
| 47 | F9 | E7 | CD11 | BI | Hi-Z | 2mA | | | |
| 48 | D7 | F8 | CD12 | BI | Hi-Z | 2mA | | | |
| 49 | E8 | D6 | CD13 | BI | Hi-Z | 2mA | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 50 | E9 | E8 | CD14 | BI | Hi-Z | 2mA | | |
| 51 | D8 | C6 | CD15 | BI | Hi-Z | 2mA | | |

The internal register can be set to operate the XINT pin in 1/0 mode or in Hi-Z/0 mode.

To use an external clock by feeding it from the CLKIN pin, attach the XI pin to the GND potential and leave the XO pin open. To use the internal oscillator of the LSI, connect a crystal resonator and oscillator circuit to the XI and XO pins and attach the CLKIN pin to the GND potential.

PD: Pull Down

PU: Pull Up

**POWER**

| QFP14 | PFBGA8 | PFBGA5 | Name | Voltage | RESET |
|---|---|---|---|---|---|
| 5,70,74 | A3,A6,C2 | A5,B3,C2 | HVDD | 3.3V | Power supply for the USB, and I/O |
| 1,19,33,42,59, 65,76 | A2,A8,C3,C8, H1,H9 | A2,A7,C7,G2 | LVDD | 1.8V | Power supply for the internal circuit, Power supply for TEST I/O, power supply for OSC |
| 13,31,46,53 | C6,F1,H6 | D7 | CVDD | 3.3 to 1.8V | Power supply for CPU interface I/O |
| 2,12,20,32,41, 45,55,56,66, 68,72,77 | B1,B2,B4,B5, B6,B7,B8,E2, E7,J2,J8 | B2,B4,B5,B6, B7,G7 | VSS | 0V | GND |
| 52,61,62,63, 64,69,78,79, 80 | A1,A9,D4,D5, D6,E4,E5,E6, F4,F5,F6,J1,J9 | A1 | N.C. | 0V | N.C. (Connect this pin to GND) |

# 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. Registers are described with a per-byte name.

- 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.CPU_IntStat bit”
- Registers provided for each endpoint
  Described as D_EPx{x=0, a-c}…register, etc.
  Example: “D_EPx{x=0, a-c}IntStat register”
- Registers provided for each channel
  Described as H_CHx{x=0, a-e}…register, etc.
  Example: “H_CHx{x=0, a-e}IntStat register”
- Registers provided for each area
  Described as AREAn{n=0-5}…register, etc.
  Example: “AREAn{n=0-5}StartAdrs_H register”

## 6.1 Selection of USB Device/Host

To use USB with this LSI, set the HostDeviceSel.HOSTxDEVICE bit to select either the USB device or the USB host register map.

Selecting the USB device register map (hereafter referred to as device mode) enables the register bits and functions of the host/device shared registers and device registers.

Selecting the USB host register map (hereafter referred to as host mode) enables the register bits and functions of the host/device shared registers and host registers.

### 6.1.1 Selection of the USB Device/Host Functions

Table 6.1 shows the item to be set when selecting a USB device/host.

Selecting a register map

**Table 6.1 Setup Items for Selecting USB Device/Host Functions**

| Item | Register/Bit | Description |
|---|---|---|
| USB device/host selection | HostDeviceSel. HOSTxDEVICE | Selects either USB device or USB host. The registers and functions on the selected side are enabled. |

### 6.1.2 USB Port State Change Detection Status

The S1R72V17 LSI has the function to detect USB port status.

This function can be used in both SLEEP and ACTIVE states (see the section on Power Management).

#### 6.1.2.1 Example Usage of USB Port State Change Detection Status

Examples of use of device port change status and host port change status are shown.

##### 6.1.2.1.1 Device Port Change Status

This status indicates that VBUS pin of the device port has changed state.

Table 6.2 shows the register associated with the Device port pin change status.

**Table 6.2 Resisters Associated with Device Port Change Status**

| Item | Register/Bit | Description |
|---|---|---|
| VBUS pin change status | DeviceIntStat. VBUS_Changed | Indicates that the VBUS pin of the device port has changed state. |
| VBUS pin change status enable | DeviceIntEnb. EnVBUS_Changed | Enable/disables assertion of the MainIntStat.USB_DeviceIntStat bit by USB_DeviceIntStat.VBUS_Changed. |
| Device port VBUS state | D_USB_Status. VBUS | Indicates the VBUS pin state of the device port. |

If the device port status is changed, the firmware performs processing (1), (2), and (4) to (7).

(1) Clear the VBUS pin change status.

(2) Set the VBUS pin change status enable.

(3) When VBUS is supplied to a device port, a VBUS pin change status is issued.

(4) Check the VBUS pin change status.

(5) Clear the VBUS pin change status.

(6) Clear the VBUS pin change status enable.

(7) Check the device port VBUS status. If the device port VBUS status bit = 1, the firmware determines that VBUS is supplied (i.e., the device port has a host or hub connected to it).

#### 6.1.2.1.2 Host Port Change Status

This status indicates the status of the power driver that controls VBUS on host ports.

Table 6.3 shows the registers associated with the host port change status.

**Table 6.3 Registers Associated with Host Port Change Status**

| Item | Register/Bit | Description |
|---|---|---|
| VBUS error detection status | USB_HostIntStat. VBUS_Err | Indicates that an error occurred in VBUS. |
| VBUS error detection status enable | USB_HostIntEnb. EnVBUS_Err | Enable/disables assertion of the MainIntStat. USB_HostIntStat bit by USB_HostIntStat.VBUS_Err. |
| VBUS enable | H_USB_Control. VBUS_Enb | Enables an external USB power switch. |
| Host port VBUS state | H_USB_Status. VBUS_State | Indicates the VBUS state of the host port (normal or erratic). |

If a VBUS error detection status is detected, immediately turn the VBUS enable off to stop VBUS drive.

## 6.2 USB Device Control

The following describes the USB device functions.

### 6.2.1 Endpoints

The LSI stipulated herein has the endpoint for control transfer (EP0) and five general-purpose endpoints (EPa, EPb, and EPe). The endpoints EPa, EPb, EPc, EPd, and EPe can each be used simultaneously as endpoints for Bulk, Interrupt, and Isochronous transfers. The alarm endpoint function incorporated in the LSI to generate an alarm when a transaction is issued from the USB host also permits the user system to have up to 15 IN endpoints and up to 15 OUT endpoints, not including the endpoint EP0. The alarm endpoint function notifies the firmware of a state by returning a NAK response for the transaction issued to any endpoint other than the active endpoints set in EPa, EPb, EPc, EPd, and/or EPe.

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, to configure the USB-defined interface.

Each endpoint has fixed basic setup items determined by the USB-defined interface and the variable control items and status to be controlled for each transfer. The basic setup items should be set up when initializing the chip or switching USB-defined interfaces from one to another.

Table 6.4 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 IN and OUT directions. The direction of data transaction should be set by the firmware as appropriate for the execution of the data and status stages at the endpoint EP0.

Any desired transaction can be performed by joining to the FIFO area described later. First, set AREA0StartAdrs_H,L and AREA0EndAdrs_H,L to reserve memory space for the FIFO area to be used. After initializing the FIFO area with AREA0FIFO_Clr, set up AREA0Join_1.JoinEP0CH0 for the FIFO area zero. No data transfers via a FIFO area can be performed until this joining process is executed.

**Table 6.4 Basic Setup Items of the Endpoint EP0**

| Item | Register/Bit | Description |
|---|---|---|
| Max. packet size | D_EP0MaxSize | Sets the max packet size.<br>Set MaxSize to 8, 16, 32, or 64 for operation in FS mode, or 64 for operation in HS mode. |
| FIFO area | AREA0StartAdrs_H,<br>AREA0StartAdrs _L,<br>AREA0EndAdrs _H,<br>AREA0EndAdrs _L | Sets an area to be allocated for the endpoint EP0 with the specified FIFO addresses.<br>Make sure the FIFO area is allocated memory space greater than the max packet size.<br>For detailed information on how to allocate a FIFO area, refer to the section in which FIFOs are discussed. |
| FIFO area join | AREA0Join_1.JoinEP0CH0 | Joins the endpoint EP0 to the allocated area.<br>Make sure endpoint EP0 is joined to AREA0. |

Table 6.5 lists the basic setup items for the general-purpose endpoints (EPa, EPb, Epc, Epd and EPe). Since the endpoints EPa, EPb, EPc, EPd, and EPe can accept any transaction directions and endpoint numbers set as desired, data transfers can be performed for up to five independent endpoints simultaneously. 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 FIFO areas for the endpoints EPa, EPb, Epc, Epd and EPe are set by the start and the end address of each area. Any desired transaction can be executed by joining the FIFO area described later. First, set AREAn{n=1-5}StartAdrs_H,L and AREAn{n=1-5}EndAdrs_H,L to reserve memory space for the FIFO area to be used, and after initializing the FIFO area with AREAn{n=1-5}FIFO_Clr, set up AREAn{n=1-5}Join_1.JoinEPxCHx{x=a-e} for the FIFO area used. No data transfers via a FIFO area can be performed until this joining process is executed.

**Table 6.5 Basic Setup Items of the General.purpose Endpoints**

| Item | Register/Bit | Description |
|---|---|---|
| Transaction direction | D_EPx{x=a-e}Config.INxOUT | Sets the direction of transfer at each endpoint. |
| Max. packet size | D_EPx{x=a-e}MaxSize_H,<br>D_EPx{x=a-e}MaxSize_L | Sets the max packet size of each endpoint. For the endpoints at which bulk transfer is to be performed, however, set the Max. packet size to 8, 16, 32, or 64 bytes during FS mode, or to 512 bytes during HS mode. |
| Endpoint number | D_EPx{x=a-e}Config.EndpointNumber | Sets the endpoint number for each endpoint to any value between 0x1 to 0xF. |
| Toggle mode | D_EPx{x=a-e}Config.IntEP_Mode | Sets the operation mode of interrupt transfer. For the endpoints at which bulk transfer is to be performed, always set this register bit to 0 irrespective of the transaction direction.<br>For the IN direction endpoints, set the mode of toggle sequence.<br>For the OUT direction endpoints, if interrupt transfer is to be performed, always set this register bit to 0. |
| Isochronous mode | D_EPx{x=a-e}Config.ISO | To perform an Isochronous transfer, set this bit to 1. For endpoints at which Bulk or Interrupt transfers are performed, set this bit to 0. |
| FIFO area | AREAn{n=1-5}StartAdrs_H,<br>AREAn{n=1-5}StartAdrs _L,<br>AREAn{n=1-5}EndAdrs _H,<br>AREAn{n=1-5}EndAdrs _L | Sets the area to be allocated to each endpoint by FIFO address.<br>Make sure the allocated FIFO area is equal to or greater than the Max. packet size on each channel. The size of the FIFO area affects the throughput of data transfers.<br>For details on the allocation of FIFO areas, refer to the relevant section in Chapter 6 on FIFOs. |
| FIFO area join | AREAn{n=1-5}Join_1.JoinEPxCHx{x=a-e} | Joins each endpoint to its allocated area. Make sure each endpoint is joined to the corresponding areas in the following combinations. Note, however, that unused endpoints cannot be joined to the FIFO areas.<br>Endpoint EPa: AREA1<br>Endpoint EPb: AREA2<br>Endpoint EPc: AREA3<br>Endpoint EPd: AREA4<br>Endpoint EPe: AREA5 |

Table 6.6 lists the basic setup items for an alarm endpoint. Although this alarm endpoint currently is not assigned to any general-purpose endpoints, it is provided to allow implementation of an endpoint defined in the USB-defined interface. Set up these items appropriately according to the definitions specified under the USB standard interface. Also, enable the settings made to configure a USB-defined interface.

This alarm endpoint does not require a FIFO area.

**Table 6.6 Basic Setup Items for an Alarm Endpoint**

| Item | Register/Bit | Description |
|---|---|---|
| Alarm endpoint enable | D_EnEP_IN_H.EnEPn{n=8-15}IN,<br>D_EnEP_IN_L.EnEPn{n=1-7}IN,<br>D_EnEP_OUT_H.EnEPn{n=8-15}OUT,<br>D_EnEP_OUT_L.EnEPn{n=1-7}OUT | Enables an alarm endpoint |
| Isochronous mode | D_EnEP_IN_ISO_H.EnEPn{n=8-15}IN_ISO,<br>D_EnEP_IN_ISO_L.EnEPn{n=1-7}IN_ISO,<br>D_EnEP_OUT_ISO_H.EnEPn{n=8-15}OUT_ISO,<br>D_EnEP_OUT_ISO_L.EnEPn{n=1-7}OUT_ISO | To place the endpoint in Isochronous transfer mode, set this bit to 1. For endpoints set for Bulk or Interrupt transfer, set this bit to 0. |

### 6.2.2 Transactions

The LSI provides transaction execution functions in hardware and provides the firmware with the interfaces necessary to execute transactions. The interfaces for the firmware are implemented as control and status registers and the interrupt signals that are asserted by a status. For details on setting interrupt assertion by status, refer to the relevant section on registers.

The LSI issues a 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 find its data quantity or free space to determine whether a data transfer can be performed, and then performs the transaction automatically.

For an OUT endpoint, for example, the firmware can read data out of the FIFO through the CPU interface (register read) to create a free space in the FIFO, thereby allowing OUT transactions to be automatically executed in succession. For an IN endpoint also, the firmware can write data to the FIFO through the CPU interface (register write) to create valid data in the FIFO, thereby allowing IN transactions to be automatically executed in succession.

Table 6.7 lists the control items and status relating to the transaction control for the endpoint EP0.

**Table 6.7 Endpoint EP0 Control Items and Status**

| Item | Register/Bit | Description |
|---|---|---|
| Transaction direction | D_EP0Control.INxOUT | Sets the direction of transfer in the data and status stages. |
| Descriptor reply enable | D_EP0Control.ReplyDescriptor | Invokes automatic descriptor response. |
| Descriptor reply address | D_DescAdrs_H, DescAdrs_L | Specifies the start address in FIFO of the data to be returned by an automatic descriptor response. |
| Descriptor size | D_DescSize_H, DescSize_L | Specifies the data quantity to be returned by an automatic descriptor response. |
| Control protect | D_SETUP_Control.ProtectEP0 | When this bit is set, the ForceNAK and ForceSTALL bits in the EP0ControlIN and EP0ControlOUT registers are protected against access.<br>This bit is set in hardware by the LSI when a RcvEP0SETUP status is flagged, and can be cleared by a register access by the CPU. |
| Short packet transmit enable | D_EP0ControlIN.EnShortPkt | Enables transmission of short packets less than Max. packet size. This bit is cleared when the IN transaction that transmitted a short packet is completed. |
| Toggle sequence bit | D_EP0ControlIN.ToggleStat,<br>D_EP0ControlOUT.ToggleStat | Indicates the status of the toggle sequence bits. These bits are automatically initialized by a SETUP stage. |
| Toggle set | D_EP0ControlIN.ToggleSet,<br>D_EP0ControlOUT.ToggleSet | Sets the toggle sequence bits. |
| Toggle clear | D_EP0ControlIN.ToggleClr,<br>D_EP0ControlOUT.ToggleClr | Clears the toggle sequence bits. |
| Forced NAK response | D_EP0ControlIN.ForceNAK,<br>D_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 | D_EP0ControlIN.ForceSTALL,<br>D_EP0ControlOUT.ForceSTALL | Responds with STALL for IN or OUT (including PING) transactions. |
| Automatic ForceNAK set | D_EP0ControlOUT.AutoForceNAK | Sets the EP0ControlOUT.ForceNAK bit each time an OUT transaction is completed. |
| SETUP receive status | USB_DeviceIntStat.RcvEP0SETUP | Indicates that a SETUP transaction has been executed. |
| Transaction status | D_EP0IntStat.OUT_ShortACK,<br>D_EP0IntStat.IN_TranACK,<br>D_EP0IntStat.OUT_TranACK,<br>D_EP0IntStat.IN_TranNAK,<br>D_EP0IntStat.OUT_TranNAK,<br>D_EP0IntStat.IN_TranErr,<br>D_EP0IntStat.OUT_TranErr | Indicates the result of a transaction. |
| Descriptor reply data stage end status | D_FIFO_IntStat.DescriptorCmp | Indicates that the data stage of an automatic descriptor response has ended. |

Table 6.8 lists the control items and status relating to the transaction processing for the general-purpose endpoints EPa, EPb, Epc, Epd and EPe.

**Table 6.8 General.purpose Endpoint Control Items and Status**

| Item | Register/Bit | Description |
|---|---|---|
| Automatic ForceNAK set | D_EPx{x=a-e}Control.AutoForceNAK | Sets the D_EPx{x=a-e}Control.ForceNAK bit for an endpoint each time an OUT transaction at that endpoint is completed. |
| Short packet transmit enable | D_EPx{x=a-e}Control.EnShortPkt | Enables transmission of short packets less than Max. packet size for an IN transaction. This bit is cleared when the IN transaction that transmitted a short packet is completed. |
| Automatic ForceNAK set by short packet reception disable | D_EPx{x=a-e}Control. DisAF_NAK_Short | Disables the function(*) to automatically set the D_EPx{x=a-e}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 | D_EPx{x=a-e}Control.ToggleStat | Indicates the status of the toggle sequence bit. |
| Toggle set | D_EPx{x=a-e}Control.ToggleSet | Sets the toggle sequence bit. |
| Toggle clear | D_EPx{x=a-e}Control.ToggleClr | Clears the toggle sequence bit. |
| Forced NAK response | D_EPx{x=a-e}Control.ForceNAK | Always responds with NAK for transactions irrespective of the data quantity and free space in the FIFO. |
| STALL response | D_EPx{x=a-e}Control.ForceSTALL | Responds with STALL for transactions. |
| Transaction status | D_EPx{x=a-e}IntStat.OUT_ShortACK, D_EPx{x=a-e}IntStat.IN_TranACK, D_EPx{x=a-e}IntStat.OUT_TranACK, D_EPx{x=a-e}IntStat.IN_TranNAK, D_EPx{x=a-e}IntStat.OUT_TranNAK, D_EPx{x=a-e}IntStat.IN_TranErr, D_EPx{x=a-e}I~tStat.OUT_TranErr | Indicates the result of a transaction. |

#### 6.2.2.1 SETUP Transactions

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

When a SETUP transaction is issued, the LSI stores the entire content of the data packet (8 bytes) in the D_EP0SETUP_0 through D_EP0SETUP_7 registers and then returns an ACK response. Furthermore, except for SetAddress() requests, the LSI issues a RcvEP0SETUP status to the firmware.

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

When the SETUP transaction is completed, the LSI sets the ForceNAK bit and clears the ForceSTALL bit in the D_EP0ControlIN and D_EP0ControlOUT registers. It also sets the ToggleStat bit. Furthermore, it sets the D_SETUP_Control.ProtectEP0 bit. When the firmware has finished setting up the endpoint EP0 and is ready to go to the next stage, it should clear the SETUP_Control.ProtectEP0 bit and then the ForceNAK bit in the D_EP0ControlIN or D_EP0ControlOUT register for the direction concerned.

Fig. 6.1 shows how a SETUP transaction is performed in device mode. 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 bytes long data packet. The LSI writes this data to the D_EP0SETUP_0 to D_EP0SETUP_7 registers. In (c), the LSI automatically returns an ACK response. Furthermore, it sets up the registers to be automatically set and issues a status to the firmware.

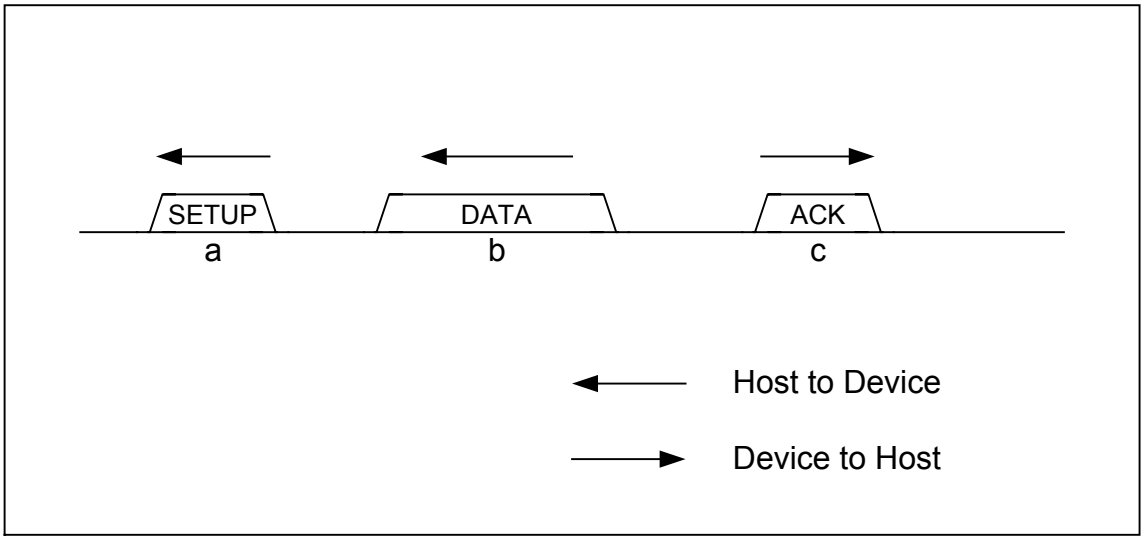



**Fig. 6.1 SETUP transaction in device mode**

#### 6.2.2.2 Bulk/Interrupt OUT Transactions

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

When all bytes of data are received correctly in a bulk/interrupt OUT transaction, the LSI completes the transaction and returns an ACK or a NYET response. It then issues an OUT_TranACK status for the corresponding endpoint (D_EPx{x=0, a-e}IntStat.OUT_TranACK bit) to the firmware. It also updates the FIFO and assuming that data has all 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 issues an OUT_ShortACK status (D_EPx{x=0, a-e}IntStat.OUT_ShortACK bit), in addition to the transaction-complete processing described above. Furthermore, if the D_EPx{x=0, a-e}Control.DisAF_NAK_Short bit is cleared, the LSI sets the D_EPx{x=a-e}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 issue a status. The FIFO is not updated.

If an error occurs in a bulk/interrupt OUT transaction, the LSI does not respond for the transaction. In this case, it issues an OUT_TranErr status (D_EPx{x=0, a-e}IntStat.OUT_TranErr bit). 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 issues an OUT_TranNAK status (D_EPx{x=0, a-e}IntStat.OUT_TranNAK bit). The FIFO is not updated.

Fig. 6.2 shows how a Bulk or Interrupt OUT transaction is performed in device mode in cases in which the transaction is completed. 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 Max. packet size. The LSI writes this data to the FIFO for the corresponding 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 issues a status to the firmware.

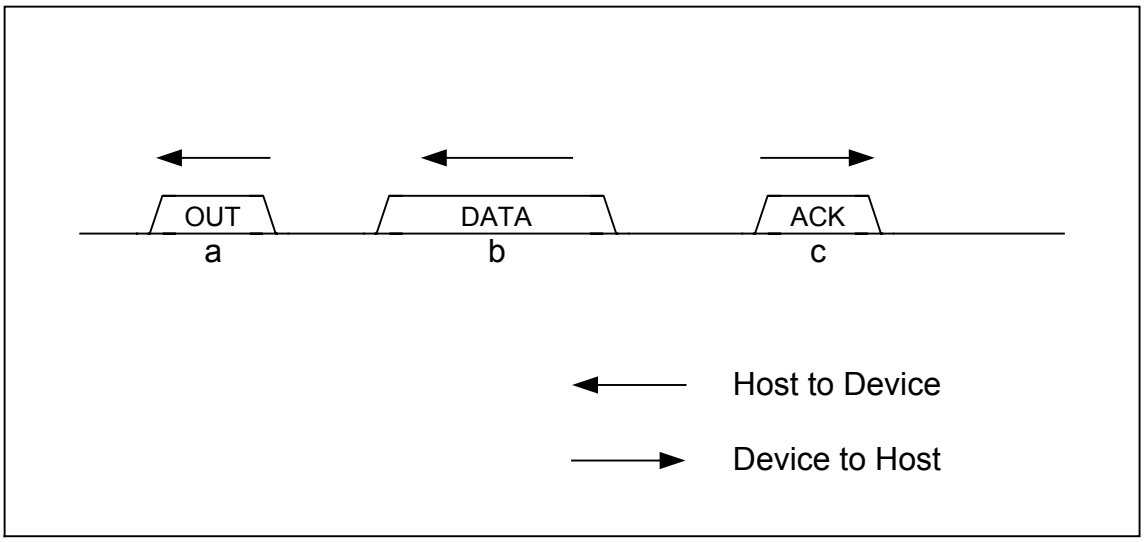



**Fig. 6.2 OUT transaction in device mode**

#### 6.2.2.3 Isochronous OUT Transaction

In an Isochronous OUT transaction, the LSI starts receiving data when the FIFO has free space equal to or greater than the max packet size. Good throughput may be obtained by allocating a FIFO area approximately twice the max packet size in order to permit the LSI to receive data while creating a free area in the FIFO by reading out data from the FIFO by means of a register read or DMA read through the CPU interface.

When all bytes of data have been received normally in an Isochronous OUT transaction, the LSI issues an OUT_TranACK status for the relevant endpoint (EPx{x=a-e}IntStat.OUT_TranACK bit) to the firmware. It also updates the FIFO and reserves an area on the assumption that the data has been received.

When all data bytes of a short packet less than the max packet size have been received in an Isochronous OUT transaction, the LSI issues an OUT_ShortACK status indication (EPx{x=a-e}IntStat.OUT_ShortACK bit), in addition to the transaction-complete processing described above. Furthermore, if the EPx{x=a-e}Control.DisAF_NAK_Short bit is cleared, the LSI sets the EPx{x=a-e}ForceNAK bit for the endpoint.

If an error occurs in an Isochronous OUT transaction, the LSI neither receives data nor updates the FIFO. Instead, it issues an OUT_TranErr status indication (EPx{x=a-e}IntStat.OUT_TranErr bit).

If all bytes of data for one packet could not be received in an Isochronous OUT transaction, the LSI issues an OUT_TranNAK status indication (EPx{x=a-e}IntStat.OUT_TranNAK bit). The FIFO is not updated.

#### 6.2.2.4 Bulk/Interrupt IN Transactions

When the FIFO for an IN-direction bulk/interrupt endpoint has a quantity of data equivalent to Max. packet size, or 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 packets with data length of 0) is enabled by setting the D_EP0ControlIN.EnShortPkt bit or D_EPx{x=a-e}Control.EnShortPkt bit. When transmitting short packets, make sure that 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 to transmit a short packet is completed, the D_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 issues an IN_TranACK status (D_EPx{x=0, a-e}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 issues an IN_TranErr status (D_EPx{x=0, a-e}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 issues IN_TranNAK status (EPx{x=0, a-e}IntStat.IN_TranNAK bit) to the firmware. It does not update the FIFO, and it does not free the storage area.

Fig. 6.3 shows how a Bulk or Interrupt IN transaction is performed in device mode in cases in which the transaction is completed. 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 within 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 issues a status to the firmware.

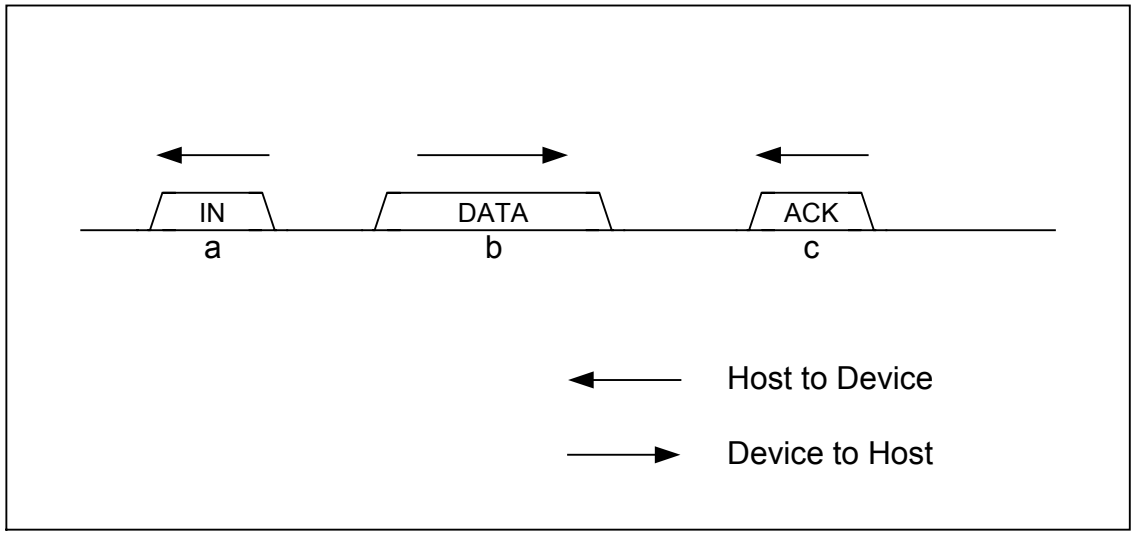



**Fig. 6.3 IN transaction in device mode**

#### 6.2.2.5 Isochronous IN Transaction

If the FIFO for an IN-direction Isochronous endpoint has a quantity of data equivalent to the max packet size or short packet transmission for that endpoint has been enabled by the firmware, the LSI returns a data packet in response to the IN transaction.

Transmission of short packets (including packets of data length zero) is enabled by setting the EPx{x=a-e}Control.EnShortPkt bit. When transmitting short packets, make sure that no new data is written to the FIFO for the endpoint in question before the transaction is completed after packet transmission has been enabled.

When a data packet is returned to the Isochronous IN transaction, the LSI completes the transaction and issues an IN_TranACK status indication (EPx{x=a-e}IntStat.IN_TranACK bit) to the firmware. It also updates the FIFO and frees the reserved area on the assumption that the transmission was completed.

If the FIFO for an Isochronous IN-direction endpoint does not have a quantity of data equivalent to the max packet size and short packet transmission for that endpoint has not been enabled by the firmware, the LSI responds with a zero-length data packet for the IN transaction and issues an IN_TranNAK status indication (EPx{x=a-e}IntStat.IN_TranNAK bit) to the firmware. It does not update the FIFO, nor does it free the reserved area.

#### 6.2.2.6 PING Transactions

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

If the FIFO for the corresponding endpoint has a free space equal to or greater than Max. packet size, the LSI responds with ACK for the PING transaction. It does not issue a status to the firmware, however.

If the FIFO for the corresponding endpoint has a free space less than Max. packet size, the LSI responds with NAK for the PING transaction. It also issues an OUT_TranNAK status (D_EPx{x=0, a-e}IntStat.OUT_TranNAK bit) to the firmware.

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

Fig. 6.4 shows how a PING transaction is acknowledged by an ACK in device mode. In (a), the host issues a PING token addressed to the OUT-direction endpoint present in this node. In (b), if the FIFO has a free space equivalent to Max. packet size, the LSI responds with ACK for the PING transaction. It also issues a status to the firmware.

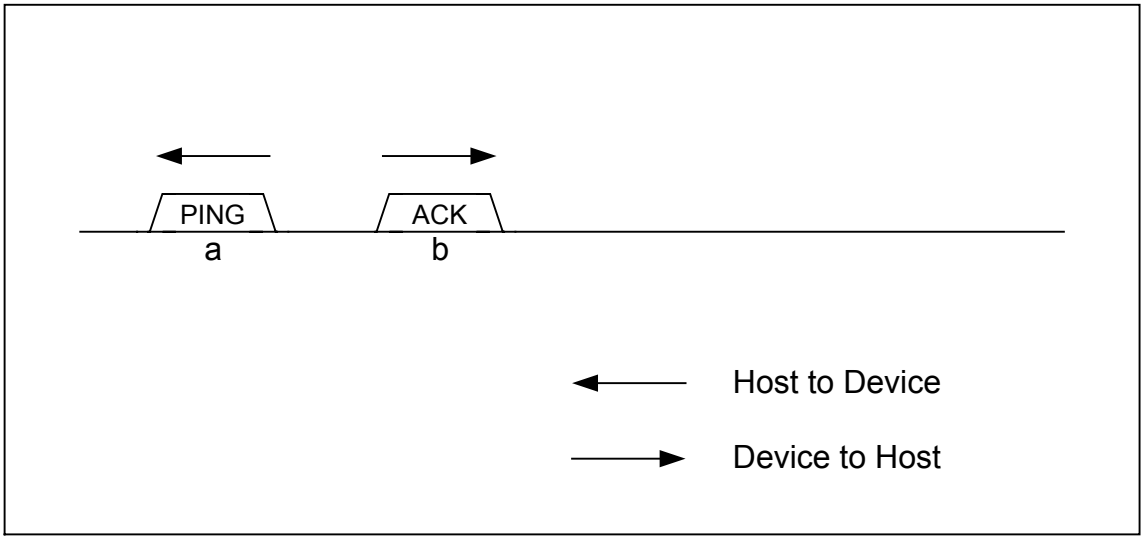



**Fig. 6.4 PING transaction in device mode**

### 6.2.3 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 that will be described later.

Fig. 6.5 shows how a Control transfer is performed in device mode in cases in which the data stage is set in the OUT direction. In (a), the host starts a control transfer via a SETUP transaction. The firmware of the device analyzes the content of the request to get prepared for responding 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 in data length of zero back to the host.

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

Transition to the status stage is accomplished by issuing a transaction for the direction opposite to the data stage from the host. The firmware monitors the IN_TranNAK status (D_EP0IntStat.IN_TranNAK bit) to seize a chance for transition from the data stage to the status stage.

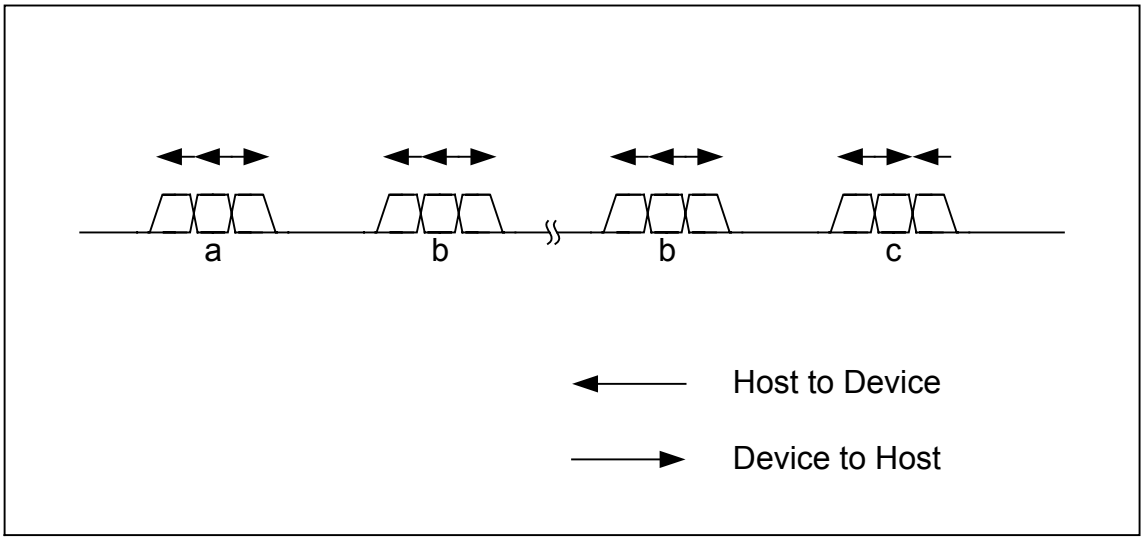



**Fig. 6.5 Control transfer in device mode when the data stage is set in the OUT direction**

Fig. 6.6 shows how a Control transfer is performed in device mode in cases in which the data stage is set in the IN direction. 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 to get ready 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 issuing a transaction for the direction opposite to the data stage from the host. The firmware monitors the OUT_TranNAK status (D_EP0IntStat.OUT_TranNAK bit) to seize a chance for transition from the data stage to the status stage.

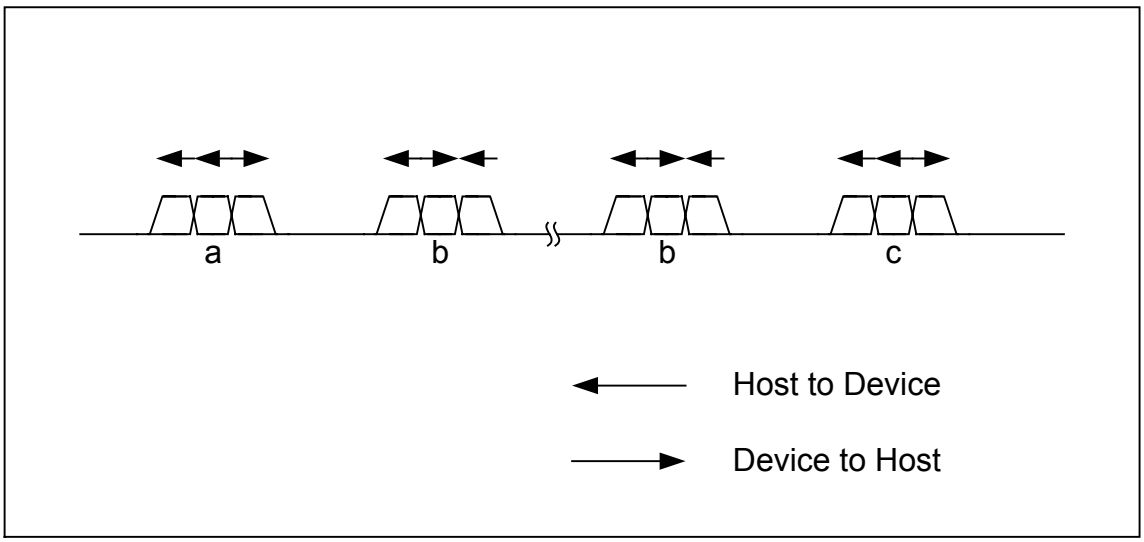



**Fig. 6.6 Control transfer in device mode when the data stage is set in the IN direction**

For the data and status stages of control transfers, a flow control by NAK is enabled, because ordinary OUT and IN transactions are performed in those stages. The device is allowed to get ready to respond within a designated time.

#### 6.2.3.1 Setup Stage

When a SETUP token addressed to the local node is received, the LSI automatically executes a setup stage. Note that this setup stage is executed unconditionally, whether or not the endpoint EP0 is joined to a FIFO area.

The firmware monitors the RcvEP0SETUP status and analyze the request by reading out the D_EP0SETUP_0 through D_EP0SETUP_7 registers to control the control transfer.

If the received request is for a control transfer with an OUT-direction data stage involved, clear the INxOUT bit in the D_EP0Control register to direct the endpoint EP0 for OUT to permit a transition to the data stage.

If the received request is for a control transfer with an IN-direction data stage involved, set the INxOUT bit in the D_EP0Control register to direct the endpoint EP0 to permit a transition to the data stage.

If the received request is for a control transfer without a data stage involved, set the INxOUT bit in the D_EP0Control register to direct the endpoint EP0 for IN to permit a transition to the status stage.

#### 6.2.3.2 Data Stage and Status Stage

Go to the next stage according to the content of a request analyzed by reading out the D_EP0SETUP_0 through D_EP0SETUP_7 registers.

If that stage is for the OUT direction, clear the INxOUT bit in the D_EP0Control register to direct it for OUT, and then set up the D_EP0ControlOUT register as appropriate to control the stage. By the time when the SETUP stage has finished, the ForceNAK bit must be set. Similarly, the D_SETUP_Control.ProtectEP0 bit must be set also.

If that stage is for the IN direction, set the INxOUT bit in the D_EP0Control register to direct it for IN, and then set up the D_EP0ControlIN register as appropriate to control the stage. By the time when the SETUP stage has finished, the ForceNAK bit must be set. Similarly, the D_SETUP_Control.ProtectEP0 bit must be set also.

#### 6.2.3.3 Automatic Address Setup Function

The LSI stipulated herein has a function to automate the processing of SetAddress() requests in control transfers at the endpoint EP0. Note that this Automatic Address Setup Function is executed unconditionally, whether or not the endpoint EP0 is joined to a FIFO area.

The LSI checks the content of a request by reading out the D_EP0SETUP_0 through D_EP0SETUP_7 registers in hardware. If the request is found to be a valid SetAddress() request, the LSI shifts to processing of the status stage for that request without notifying the firmware. When the status stage is completed, the LSI sets the relevant address in the USB_Address register and issues a SetAddressCmp status (D_SIE_IntStat.SetAddressCmp bit) to the firmware.

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

#### 6.2.3.4 Descriptor Reply Function

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

For requests where 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 D_EP0ControlIN.ForceNAK bit, set the start address of the internal data of the FIFO's descriptor area to be returned in the D_DescAdrs_H and _L registers or the total number of bytes of the data to be returned in the D_DescSize_H and _L registers, and then set the D_EP0Control.ReplyDescriptor bit.

The descriptor reply function executes an IN transaction by sending back data packets in response to the IN transaction of the data stage until the set bytes of data have all been transmitted. If an IN transaction is issued after the set bytes of data have all been transmitted, the function responds to it with NAK. If odd data less than Max. packet size exits, the descriptor reply function sets the D_EP0ControlIN.EnShortPkt bit to allow for the IN transaction to be responded until all bytes of data are sent back.

When transition to the status stage is detected by receiving an OUT token, the function clears the D_EP0Control.ReplyDescriptor bit and issues a DescriptorCmp status (D_EP0IntStat.DescriptorCmp bit) to the firmware. When a DescriptorCmp status is detected, the firmware should execute the status stage.

For details about the descriptor area, refer to the relevant section in Chapter 6 on FIFOs.

### 6.2.4 Bulk Transfer and Interrupt Transfer

Bulk and interrupt transfers at the general-purpose endpoints EPa, EPb, Epc, Epd and EPe can be controlled as a data flow (see 6.2.5) or as successive individual transactions (see 6.2.2).

### 6.2.5 Data Flow

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

#### 6.2.5.1 OUT Transfer

The data received by an OUT transfer is written to the FIFO area joined to each corresponding endpoint. There are two methods for reading out data from the FIFO: a register read through the CPU interface or a DMA read through the CPU interface.

To read data from the FIFO by means of a register read through the CPU interface, select a single endpoint using the AREAn{n=0-5}Join_0.JoinCPU_Rd bit in the same area as the FIFO area joined to each corresponding endpoint. The FIFO for the selected endpoint can be read out in the order the data was received by using the FIFO_Rd or FIFO_ByteRd register. The bytes of data that can be read out of the FIFO can be determined from the FIFO_RdRemain_H and FIFO_RdRemain_L registers. Since empty FIFOs cannot be read out, be sure to check the FIFO_RdRemain_H and

FIFO_RdRemain_L registers to determine the bytes of data in the FIFO, and make sure those bytes of data will not be exceeded when data is read from the FIFO.

To read data from the FIFO by means of a DMA read through the CPU interface, select a single endpoint using the AREAn{n=0-5}Join_0.JoinDMA bit in the same area as the FIFO area joined to each corresponding endpoint. Set the DMA_Control.Dir bit to 1. The FIFO for the selected endpoint can be read out in the order the data was received by executing a DMA procedure in the CPU interface. The remaining bytes of data in the FIFO can be determined from the DMA_Remain_H and DMA_Remain_L registers. When the FIFO is emptied, the CPU interface automatically causes the DMA to pause for flow control.

If the FIFO has a 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 packets in data length of zero) are received while the D_EPx{x=a-e}Control.DisAF_NAK_Short bit is cleared (default), the D_EPx{x=a-e}Control.ForceNAK bit for the corresponding endpoint is set. Therefore, when you have prepared for the next data transfer, be sure to clear the D_EPx{x=a-e}Control.ForceNAK bit.

#### 6.2.5.2 IN Transfer

Write the data to be sent by an IN transfer into the FIFO joined to each corresponding endpoint. There are two methods for writing into the FIFO: a register write through the CPU interface or a DMA write through the CPU interface.

To write data into the FIFO by means of a register write through the CPU interface, select a single endpoint using the AREAn{n=0-5}Join_0.JoinCPU_Rd bit in the same area as the FIFO area joined to each corresponding endpoint. The FIFO for the selected endpoint can be written to using the FIFO_Wr. The data is transmitted in packets in the order written. The amount of free space in the FIFO can be determined by inspecting the FIFO_WrRemain_H and _L registers. Full FIFOs cannot be written to. Always be sure to check the FIFO_WrRemain_H and _L registers to know the free bytes in the FIFO, and make sure those bytes will not be exceeded when data is written to the FIFO.

To write data into the FIFO by means of a DMA write through the CPU interface, select a single endpoint using the AREAn{n=0-5}Join_0.JoinDMA bit in the same area as the FIFO area joined to each corresponding endpoint. Set the DMA_Control.Dir bit to 0. 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 full, the CPU interface automatically causes the DMA to pause for flow control.

If the FIFO contains data equal to or larger than Max. packet size, the data can be transmitted by automatically responding to an IN transaction. Therefore, IN transfers can be performed without the need for control of individual transactions by the firmware. However, if a short packet needs to be transmitted at the end of data transfer, set the EnShortPkt bit. This bit is cleared when the IN transaction that transmitted the short packet is completed. The bit can be set at the time the LSI has finished writing data to the FIFO.

### 6.2.6 Bulk Only Support

The LSI stipulated herein has a bulk-only support function which in bulk transfers at the endpoints EPa, EPb, EPc, Epd and EPe, 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=a-e}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, and not the FIFOs normally reserved for endpoints, as it performs packet reception (CBW) or transmission (CSW).

#### 6.2.6.1 CBW Support

The firmware can use CBW support when it performs a command transport of the BulkOnly Transport Protocol. When the BulkOnlyConfig.EPx{x=a-e}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 active causes CBW support to be executed, so that the data received in an OUT transaction at the target endpoint is handled as CBW.

If the data packet is 31 bytes long, or the data length expected as CBW, the LSI saves the data in the CBW area and issues a CBW-complete status (D_BulkIntStat.CWB_Cmp bit) to the firmware. It also automatically clears the C_BulkOnlyControl.GoCBW_Mode bit to terminate execution of CBW support. Furthermore, if the BulkOnlyControl.GoCSW_Mode bit remains set at this point, the LSI clears it too.

If the data packet is less than or greater than 31 bytes in data length, the LSI issues a CBW data length error status (D_BulkIntStat.CBW_LengthErr bit) to the firmware. It also automatically clears the D_BulkOnlyControl.GoCBW_Mode bit to terminate execution of CBW support. Furthermore, if the D_BulkOnlyControl.GoCSW_Mode bit remains set at this point, the LSI clears it too. If a CBW_Err status is issued, it means that a phase mismatch has occurred in the BulkOnly Transport Protocol. Therefore, the firmware should restore communication by, for example, STALL'ing the endpoint.

If D_EPx{x=a-e}Control.ForceSTALL is set at the target endpoint and an OUT transaction is responded with STALL, the LSI issues a CBW error status (D_BulkIntStat.CBW_Err bit) to the firmware and clears the D_BulkOnlyControl.GoCBW_Mode bit to terminate execution of CBW support. If the D_BulkOnlyControl.GoCSW_Mode bit remains set at this point, the LSI clears it too.

If a CRC error or other transaction error occurs in an OUT transaction, the LSI issues a CBW transaction error status (D_BulkIntStat.CBW_TranErr bit) to the firmware without receiving data. In this case, the D_BulkOnlyControl.GoCBW_Mode bit is not cleared and execution of CBW

support is continued. Even if the D_BulkOnlyControl.GoCSW_Mode bit remains set at this point, it is not cleared either.

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

#### 6.2.6.2 CSW Support

The firmware can use CSW support when it performs a status transport of the BulkOnly Transport Protocol. When the D_BulkOnlyConfig.EPx{x=a-e}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 D_BulkOnlyControl.GoCSW_Mode bit while CSW support is active 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 was sent back to the host and the transaction is thereby completed, the LSI issues a CSW-complete status (D_BulkIntStat.CSW_Cmp bit) to the firmware. It also automatically clears the D_BulkOnlyControl.GoCSW_Mode bit to terminate execution of CSW support. At the same time, it sets the D_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 was sent back to the host, the LSI issues a CSW error status (D_BulkIntStat.CSW_Err bit) to the firmware. At this point, the LSI does not clear the D_BulkOnlyControl.GoCSW_Mode bit and continues execution of CSW support. At the same time, it sets the D_BulkOnlyControl.GoCBW_Mode bit in hardware to initiate execution of CBW support. In this case, therefore, execution of CSW support and execution of CBW support are exercised at the same time. If the host could not receive CSW and the transaction resulted in an error, CSW will be retried, but because CSW support is being executed, a response can be returned. Furthermore, if the device could not receive ACK and the transaction resulted in an error, the next CBW will be performed, but because CBW support is being executed, a response can be returned. Execution of CSW support is terminated by the CBW support thus executed.

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

### 6.2.7 Auto Negotiation Function

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

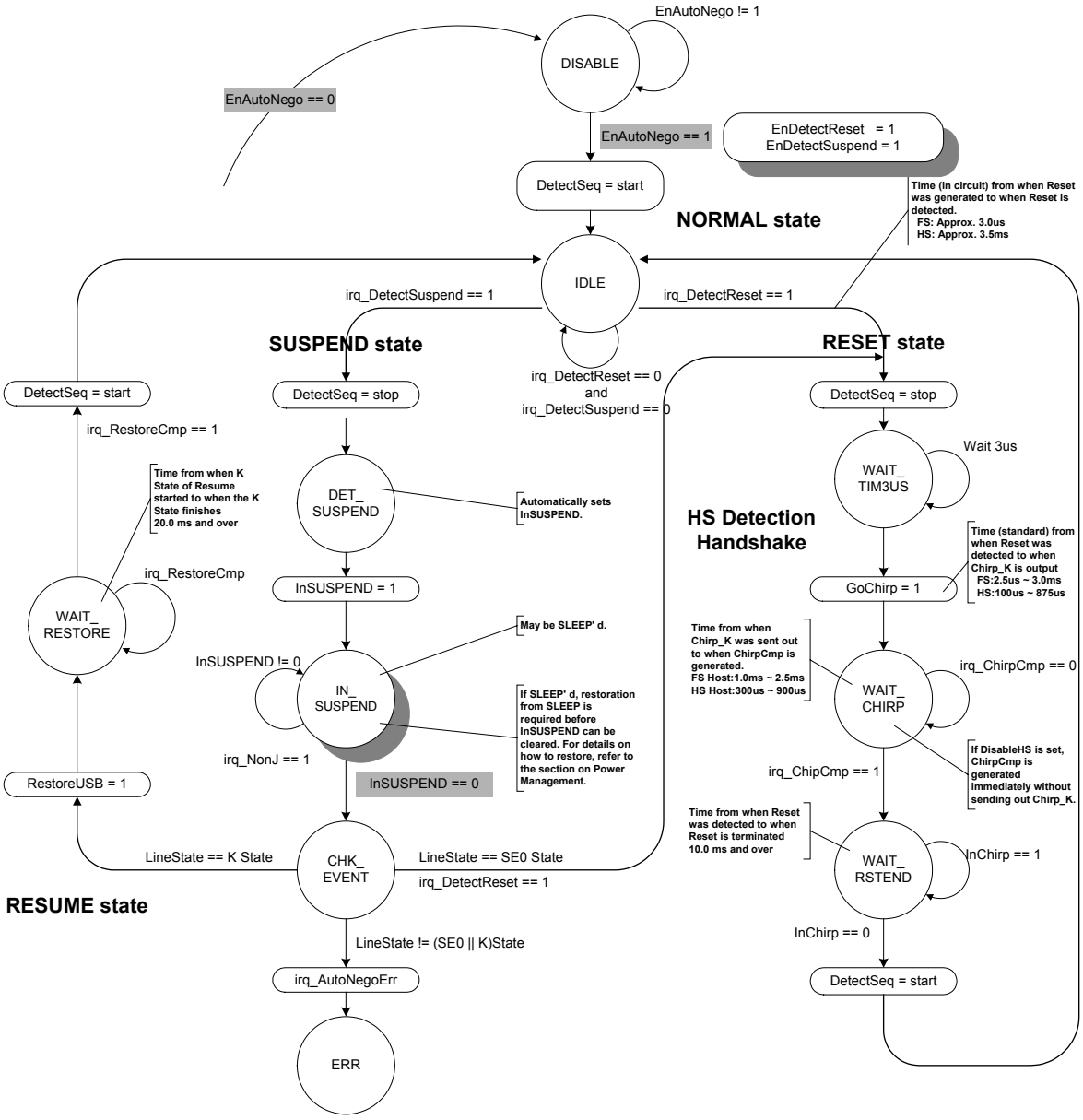



**Fig. 6.7 Auto Negotiator**

#### 6.2.7.1 DISABLE

The state shifts to this state when the D_NegoControl.EnAutoNego bit is cleared.

When enabling the auto negotiation function, set the reset detection interrupt enable bit (D_SIE_IntEnb.EnDetectRESET) and the suspend detection interrupt enable bit (D_SIE_IntEnb.EnDetectSUSPEND) to enable both event detection interrupts before setting the D_NegoControl.EnAutoNego bit.

When the auto negotiation function is enabled, the internal event detection function is enabled automatically. Do not set the D_NegoControl.DisBusDetect bit while the auto negotiation function is enabled.

#### 6.2.7.2 IDLE

This is the waiting state for Reset Detection or Suspend Detection.

When 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 and then Suspend is assumed if FS-J is detected, or Reset is assumed if SE0 is detected. When the current USB speed is FS, Reset is assumed if SE0 in duration of 2.5 μs or more is detected, or Suspend is assumed if no bus activity is detected for 3 ms or more. A reset detection or suspend detection interrupt is generated at the same time the above judgment is made, and the D_SIE_IntStat.DetectRESET or D_SIE_IntStat.DetectSUSPEND bit is set.

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

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

#### 6.2.7.3 WAIT_TIM3US

This state is provided for adjusting the time before HS Detection Handshaking is executed after reset has been detected. The WAIT_CHIRP enters the state after certain predetermined time has elapsed (approx. 3 μs later).

#### 6.2.7.4 WAIT_CHIRP

HS Detection Handshaking is executed by automatically setting the D_NegoControl.GoChirp bit. When HS Detection Handshaking finishes, the Chirp-complete interrupt status (D_SIE_IntStat.ChirpCmp) is set and the LSI is shifted to the WAIT_RSTEND state. For details about HS Detection Handshaking, refer to Section 6.2.7.11.5.

Furthermore, while the D_NegoControl.DisableHS bit remains set, the Chirp-complete interrupt status (D_SIE_IntStat.ChirpCmp) is set and the state shifts to the WAIT_RSTEND state without executing HS Detection Handshaking.

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

#### 6.2.7.5 WAIT_RSTEND

The device waits in this state until the reset period terminates. During HS, the reset period is determined as finished when the Chirp transmission from the host (or reception for the LSI) is complete. During FS, the same is assumed when a transition from SE0 to J occurred.

After the reset period is determined as finished, the event detection function is enabled and the LSI re-enters the IDLE state.

#### 6.2.7.6 DET_SUSPEND

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

Whether the reduction in the current consumption amount in the chip actually takes place during Suspend would depend on the application. The LSI stipulated herein incorporates measures to reduce the current consumption (Sleep). For details about these measures and on how to control, refer to Section 6.4, “Power Management Function.”

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

#### 6.2.7.7 IN_SUSPEND

When the NonJ interrupt status (D_SIE_IntStat.NonJ) is set, an instruction to return from Suspend is assumed, so that when the D_NegoControl.InSUSPEND bit is cleared by the firmware, the LSI is shifted to the CHK_EVENT state.

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

#### 6.2.7.8 CHK_EVENT

Events on the USB cable are checked, and if FS-K is detected, Resume is assumed, or if SE0 is detected, Reset is assumed. When Resume is assumed, the D_NegoControl.RestoreUSB bit is set, and the transfer speed before Suspend (which depends on the D_USB_Status.FSxHS value) is restored. When Reset is assumed, the event detection function is temporarily turned off as for a transition from the IDLE state, and the LSI is shifted to the WAIT_TIM3US state.

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

#### 6.2.7.9 WAIT_RESTORE

When the D_SIE_IntStat.RestoreCmp bit is set, the event detection function is enabled and the LSI is shifted to the IDLE state.

#### 6.2.7.10 ERR

Once the LSI is shifted to this state, it will not exit this state unless the Auto Negotiation function is turned off. This state is nonexistent in the USB standard.

Note that in whichever state, no determination is made with regard to the removal of the USB cable. In the event the USB cable is removed, therefore, the application should turn off the Auto Negotiation function.

#### 6.2.7.11 Individual Description of Each Negotiation Function

##### 6.2.7.11.1 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), the function is shifted to the FS mode automatically. (The HS termination is disabled and the FS termination (Rpu) is enabled.) As a result, DP shifts to ‘high' , and the “J” state can be confirmed by checking the D_USB_Status.LineState [1:0] bits. (Be aware that when SE0 is detected, Reset is assumed as noted later.) Beyond this point, if “J” is still detected at T2, the D_SIE_IntStat.DetectSUSPEND bit is set.

At this point, if the D_SIE_IntEnb.EnDetectSUSPEND and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also 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 a device operation when Sleep' d.

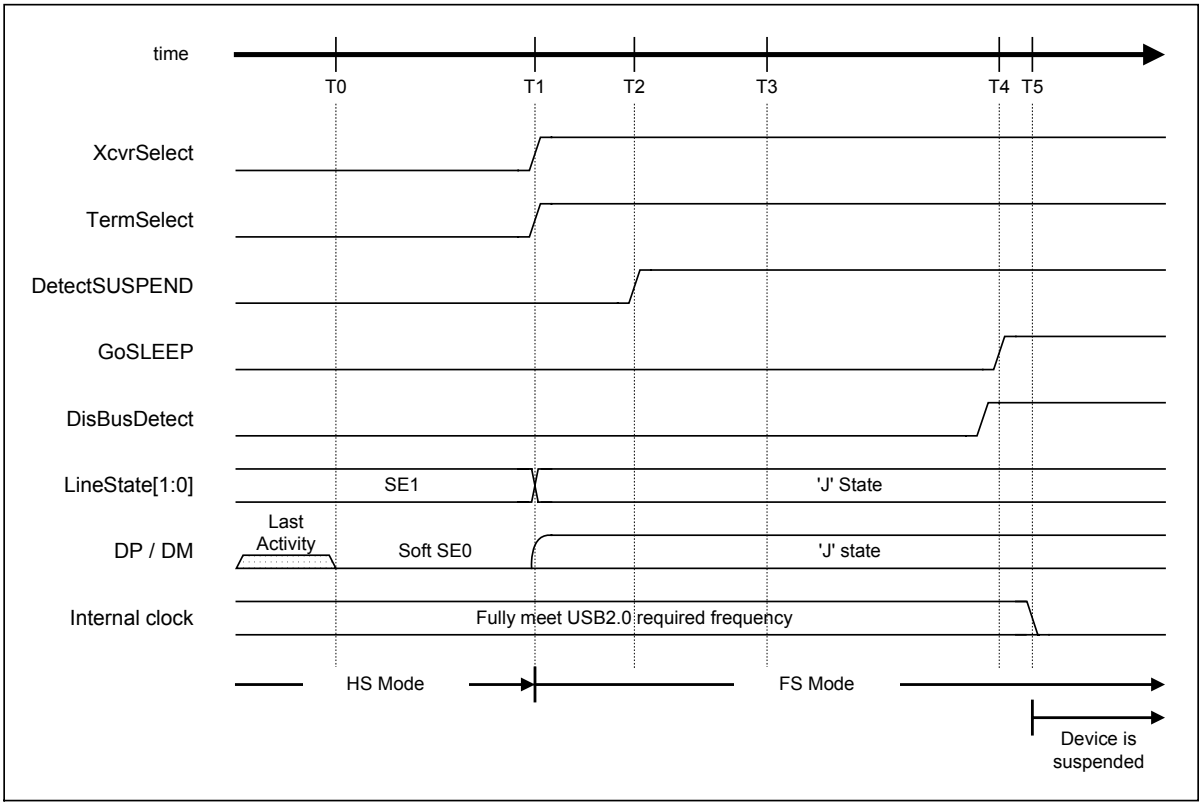



**Fig. 6.8 Suspend timing (HS mode)**

**Table 6.9 Suspend Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | Most recent bus activity. | 0 (reference) |
| T1 | If no bus activities are still detected at this point, set XcvrSelect and TermSelect to 1 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 point, DetectSUSPEND is set to 1, so that 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 | Placed into full suspend state. Beyond this point, no suspend currents greater than stipulated in USB standard can be drawn from VBUS.<br>(Set DisBusDetect to 1 before entering Sleep state.) | HS Reset T0 + 10ms {$T_{2SUSP}$} |
| T5 | The internal clock is completely turned off. | T5 < T4 + 10us |

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

#### 6.2.7.11.2 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 D_USB_Status.LineState [1:0] bits 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 point, if the D_SIE_IntEnb.EnDetectSUSPEND and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above. Shown in the diagram below is a device operation when it is Sleep' d.

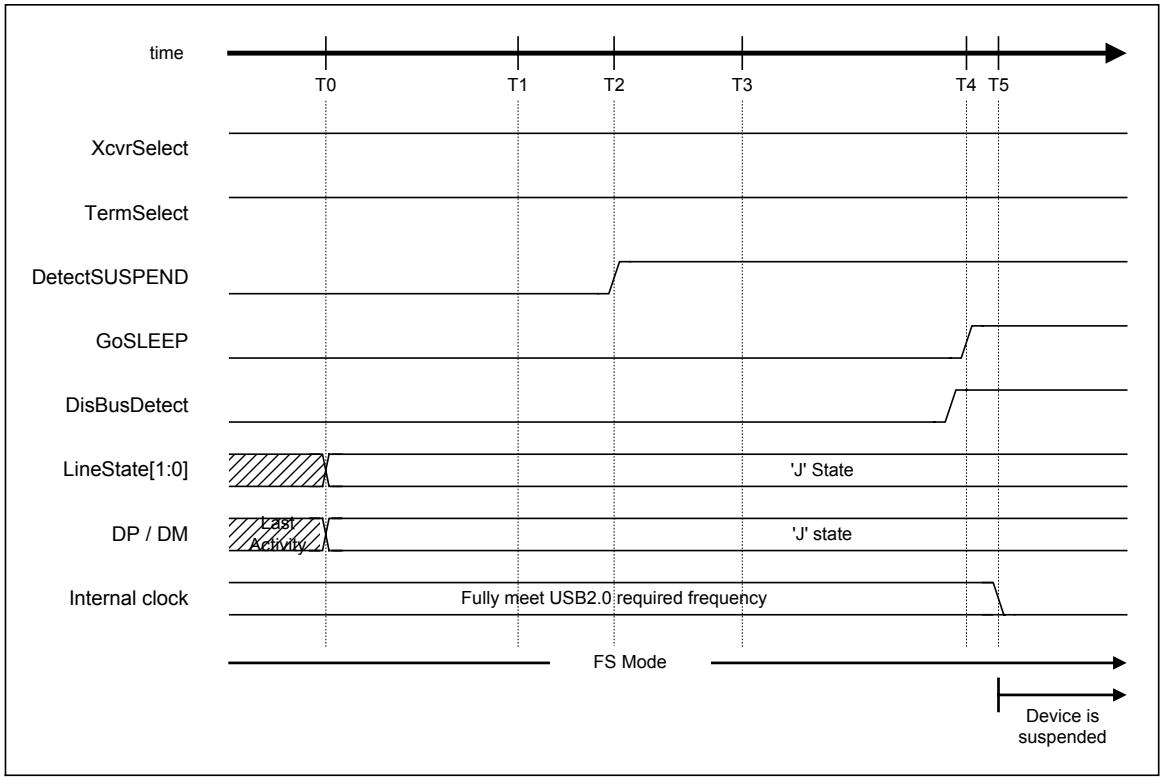



**Fig. 6.9 Suspend timing (FS mode)**

**Table 6.10 Suspend Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | Most recent bus activity. | 0 (reference) |
| T1 | No bus activities are still detected at this point. | T0 + 3.0ms < T1 {$T_{WTREV}$} < T0 + 3.125ms |
| T2 | Sample LineState [1:0]. If "J" is detected at this point, DetectSUSPEND is set to 1, so that 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 | Placed into full suspend state. Beyond this point, no suspend currents greater than stipulated in USB standard can be drawn from VBUS.<br>(Set DisBusDetect to 1 before entering Sleep state.) | T0 + 10ms {$T_{2SUSP}$} |
| T5 | The internal clock is completely turned off. | T5 < T4 + 10us |

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

#### 6.2.7.11.3 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, the function is shifted to the FS mode automatically. (The HS termination is disabled and the FS termination (Rpu) is enabled.) Even when this operation is performed, the DP line remains low, and consequently "SE0" can be detected in the D_USB_Status.LineState [1:0] bits. If "SE0" is still detected at T2, the D_SIE_IntStat.DetectRESET bit is set.

At this point, if the D_SIE_IntEnb.EnDetectRESET and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above. Therefore, assuming that this is an instruction for Reset, execute HS Detection Handshaking (described later) after setting the D_NegoControl.DisBusDetect bit.

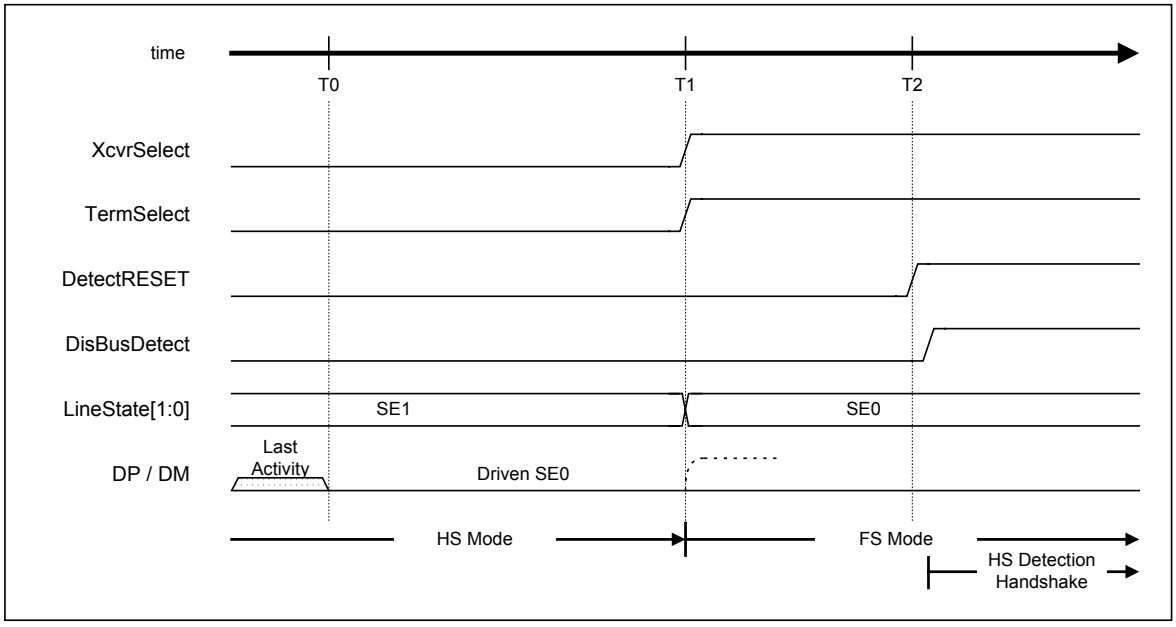



**Fig. 6.10 Reset timing (HS mode)**

**Table 6.11 Reset Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | Most recent bus activity. | 0 (reference) |
| T1 | If no bus activities are still detected at this point, set XcvrSelect and TermSelect to 1 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 point, DetectRESET is set to 1, so that a transition to Reset should be assumed. After detecting the Reset instruction, set DisBusDetect to 1 and then execute HS Detection Handshaking. | T1 + 100us < T2 {$T_{WTWRSTHS}$} < T1 + 875us |

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

#### 6.2.7.11.4 Reset Detection (FS Mode)

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

At this point, if the D_SIE_IntEnb.EnDetectRESET and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above. Therefore, assuming that this is an instruction for Reset, execute HS Detection Handshaking (described later) after setting the D_NegoControl.DisBusDetect bit.

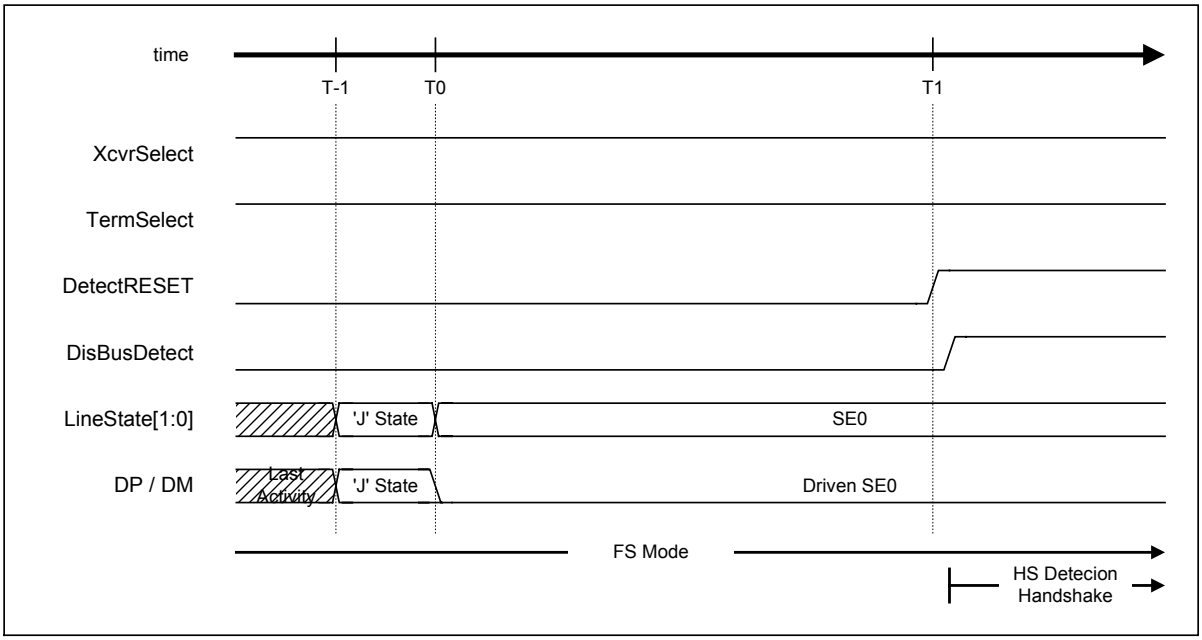



**Fig. 6.11 Reset timing (FS mode)**

**Table 6.12 Reset Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T-1 | Most recent bus activity. | |
| T0 | A reset instruction from the downstream port is initiated. | 0 (reference) |
| T1 | If “SE0” continues, DetectRESET is set to 1, so that a transition to Reset should be assumed.<br>After detecting the Reset instruction, set DisBusDetect 1 and then execute HS Detection Handshaking. | HS Reset T0 + 2.5us < T1 {$T_{WTREV}$} |

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

#### 6.2.7.11.5 HS Detection Handshaking

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

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

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

While the LSI stipulated herein is operating in FS mode, go to HS Detection Handshaking after detecting "SE0" in 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 duration of 3.0 ms or more because it is necessary to determine whether the USB state is Suspend or Reset before going to HS Detection Handshaking. To do this, change the D_XcvrControl.XcvrSelect and D_XcvrControl.TermSelect bits both to FS mode, disable the HS termination, and enable the FS termination. This mode change operation must be performed within 3.125 ms. Within a period of 100 µs to 875 µs after the mode change, check the D_USB_Status.LineState [1:0] bits, and if the bits indicate "J," the Suspend state of USB should be assumed; if "SE0," the Reset of USB should be assumed. If Reset is assumed at this point, go to HS Detection Handshaking after that.

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

Although there will be no problem during the LSI' s operation because the internal clock has been sufficiently tuned in, the internal clock will become inactive when Reset is detected if the LSI is placed in a Sleep state during Suspend. Therefore, the PM_Control.GoActiveALL bit must be set to 1 to activate the internal clock before HS Detection Handshaking can be performed. For details about this operation, refer to Section 6.4, "Power Management Function."

#### 6.2.7.11.5.1 When Connected to an FS Downstream Port

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

First, set the D_NegoControl.GoChirp bit. The XcvrControl.OpMode [1:0] bits will thereby be set to "Disable Bit Stuffing and NRZI encoding," and data fully populated with 0s will be prepared (T1). This is used to send "HS K" (chirp) on to the bus. If the D_XcvrControl.XcvrSelect bit is set to HS mode and transmission is enabled simultaneously with the above, "HS K" (chirp) is sent out to the downstream port. After the send activity, the LSI waits for "chirp" from the downstream (T2). Usually, if the downstream port supports HS, "HS K" and "HS J" will be sent out successively from the downstream port beginning with T3 (as will be described later). If the downstream port does not support HS (as in the present case), however, "chirp" will not be sent out from the downstream port even at T4. Therefore, the D_XcvrControl.XcvrSelect bit is automatically changed to FS mode, and while the D_USB_Status.FSxHS bit is set simultaneously as the D_NegoControl.GoChirp bit is cleared, along with which the D_SIE_IntStat.ChirpCmp bit is also set.

At this point, if the D_SIE_IntEnb.EnChirpCmp and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above, so that HS Detection Handshaking should be assumed to have finished.

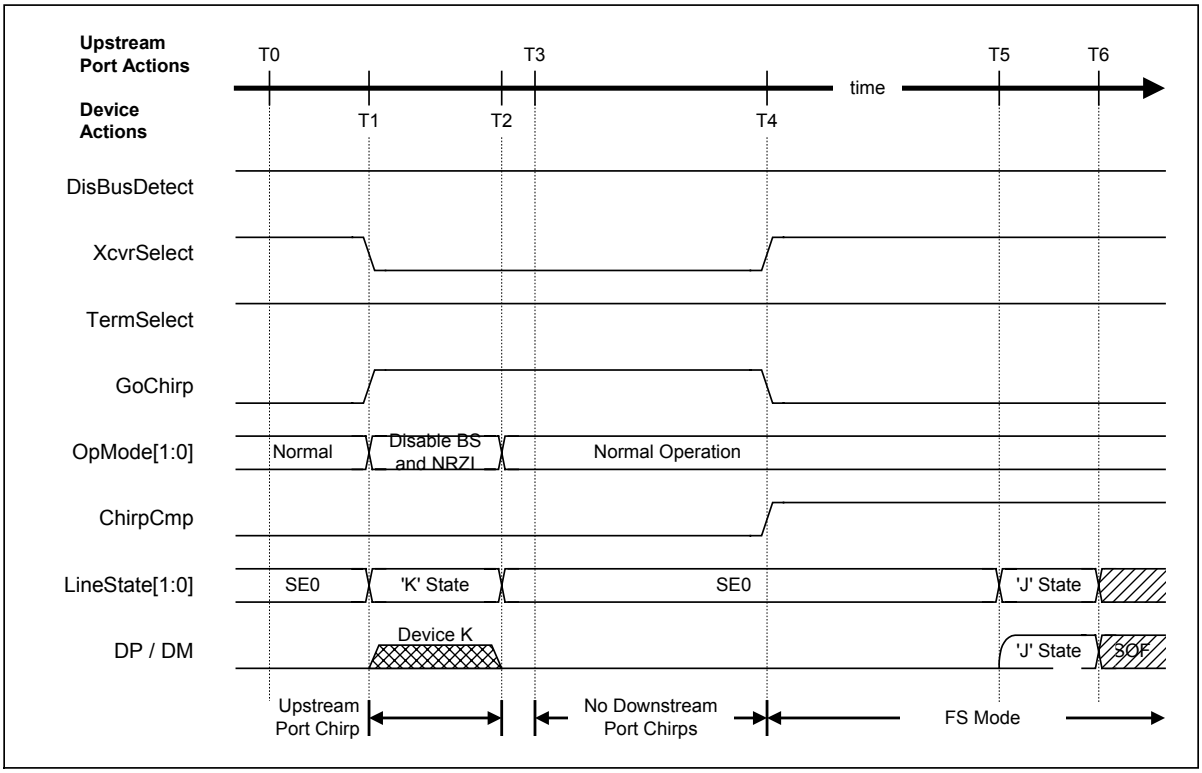



**Fig. 6.12 HS Detection Handshake timing (FS mode)**

**Table 6.13 HS Detection Handshake Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | HS Detection Handshaking is initiated. | 0 (reference) |
| T1 | HSEnable the HS transceiver and set GoChirp to 1 to start sending out Chirp K. | T0 < T1 < HS Reset T0 + 6.0ms |
| T2 | Finish sending out Chirp K. This signal must be sent out for at least 1 ms. | T1 + 1.0ms {TUCH} < T2 < HS Reset T0 + 7.0ms {TUCHEND} |
| T3 | If the downstream port supports HS, start sending out Chirp K from here. | T2 < T3 < T2 + 100us {TWTDCH} |
| T4 | If Chirp cannot be detected, return to FS mode at this point and wait until ChirpCmp is set to 1 and the reset sequence finishes. | T2 + 1.0ms < T4 {TWTFS} < T2 + 2.5ms |
| T5 | The reset sequence finishes. | HS Reset T0 + 10ms {TDRST (Min)} |
| T6 | Normal operation in FS mode. | T6 |

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

Note: To generate Chirp K of at least 1 ms, determine the duration by 66,000 cycles (internal clock = 60 MHz).

#### 6.2.7.11.5.2 When Connected to an HS Downstream Port

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

First, set the D_NegoControl.GoChirp bit. The D_XcvrControl.OpMode [1:0] bits will thereby be set to "Disable Bit Stuffing and NRZI encoding," and data fully populated with 0s will be prepared (T1). This is used to send "HS K" (chirp) on to the bus. If the D_XcvrControl.XcvrSelect bit is set to HS mode and transmission is enabled simultaneously with the above, "HS K" (chirp) is sent out to the downstream port. After the LSI has finished sending out, it waits for "chirp" from the downstream (T2). Since 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 D_USB_Status.LineState [1:0] bits (T6), the D_XcvrControl.TermSelect bit is automatically changed to HS mode (T7), by which the LSI is placed completely into HS mode. At the same time, the D_NegoControl.GoChirp bit is cleared as is the D_NegoStatus.FSxHS bit, and the D_SIE_IntStat.ChirpCmp bit is set.

At this point, if the D_SIE_IntEnb.EnChirpCmp and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above, so that 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 misinterpreted 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 as they are detected successively.

Note that the D_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 this is the reason that the D_USB_Status.LineState [1:0] bits can be used for said purpose. However, if bus signals are superimposed on the D_USB_Status.LineState [1:0] bits, the bus signals become extremely noisy, so that when the D_XcvrControl.TermSelect bit is in HS mode, the D_USB_Status.LineState [1:0] bits will output "J" if presence of bus activity is assumed, or "SE0" if absence of bus activity is assumed.

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

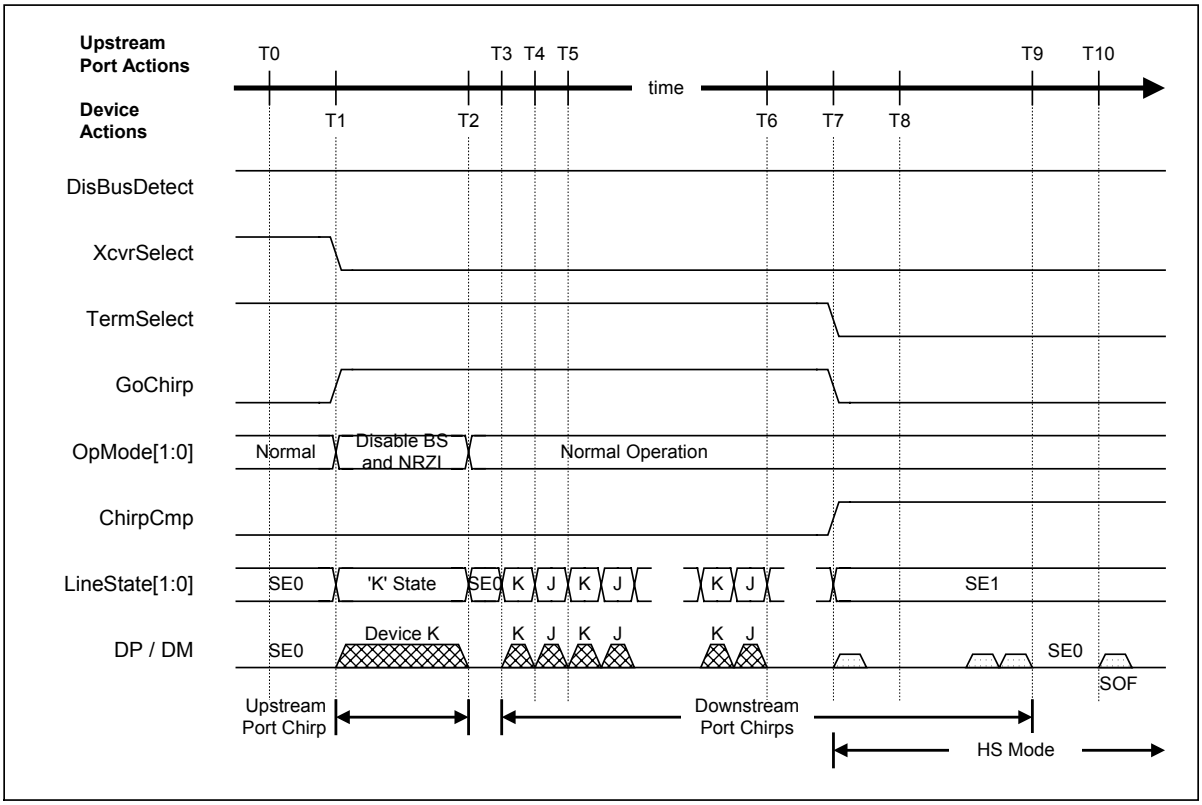



**Fig. 6.13 HS Detection Handshake timing (HS mode)**

**Table 6.14 HS Detection Handshake Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | HS Detection Handshaking is initiated. | 0 (reference) |
| T1 | Enable the HS transceiver and set GoChirp to 1 to start sending out Chirp K. | T0 < T1 < HS Reset T0 + 6.0ms |
| T2 | Finish sending out Chirp K. This signal must be sent out for at least 1 ms. | T1 + 1.0ms {TUCH} < T2 < HS Reset T0 + 7.0ms {TUCHEND} |
| T3 | The downstream port sends the first Chirp K on to the bus. | T2 < T3 < T2 + 100us {TWTDCH} |
| T4 | The downstream port stops sending Chirp K and sends out Chirp J instead. | T3 + 40us {TDCHBIT (Min)} < T4 < T3 + 60us {TDCHBIT (Max)} |
| T5 | The downstream port stops sending Chirp J and sends out Chirp K instead. | T4 + 40us {TDCHBIT (Min)} < T5 < T4 + 60us {TDCHBIT (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 the FS termination and enable the 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. However, since SYNC cannot be detected, this is not recognized as a packet reception in progress. | T8 |
| T9 | Transmission of Chirp K and Chirp J from the downstream port finishes. | T10 - 500us {TDCHSE0 (Max)} < T9 < T10 - 100us{TDCHSE0 (Min)} |
| T10 | The reset sequence finishes. | HS Reset T0 + 10ms {TDRST (Min)} |

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

Note: To generate Chirp K of at least 1 ms, determine the duration by 66,000 cycles (internal clock = 60 MHz).

#### 6.2.7.11.5.3 When Reset during Sleep

The LSI stipulated herein does not have its internal clocks output during a Sleep state.

If Reset is detected during a Sleep state (T0), the D_SIE_IntStat.NonJ bit is set. Furthermore, if the D_SIE_IntEnb.EnNonJ and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above. In this case, to allow the LSI to immediately return from the Sleep state and go to a reset sequence, set the PM_Control.GoActiveALL bit to 1 (T1). PM_Control.PM_State [1:0] is asserted after the OSC oscillation start time and the PLL power-up time have elapsed (T2), at which time the LSI begins to output the internal clock. After that, execute HS Detection Handshaking (described above).

At this point, unless the oscillator circuit has been turned off (unless returning from a Sleep state), the internal clock is output with the frequency accuracy conforming to the USB2.0 standard.

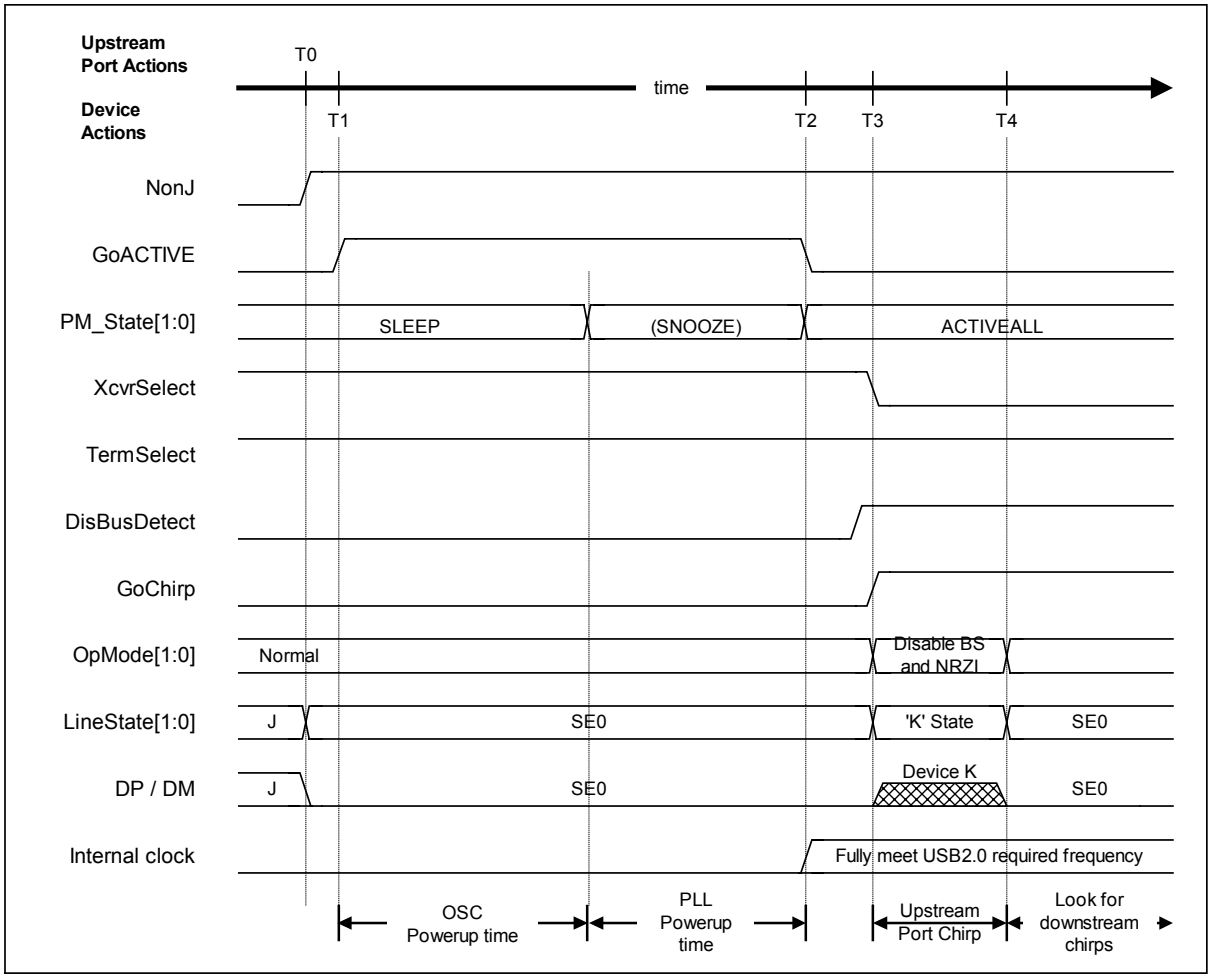



**Fig. 6.14 HS Detection Handshake Timing from Suspend**

**Table 6.15 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], a reset during sleep is detected. | 0 (HS Reset T0) |
| T1 | After detection of Reset, set GoActive to 1. | T1 |
| T2 | PM_State becomes active, completing GoACTIVE. Internal clock output stabilizes. | T1 + OSC Powerup + PLL Powerup < T2 |
| T3 | Set GoChirp to 1 to send Chirp K on to the bus.<br>(Set DisBusDetect to 1 before sending out Chirp K.) | T2 < T3 < HS Reset T0 + 5.8ms |
| T4 | Finish sending out Chirp K. | T3 + 1.0ms {TUCH} < T4 <<br>HS Reset T0 + 7.0ms {TUCHEND} |

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

Note: To generate Chirp K of at least 1 ms, determine the duration by 66,000 cycles (internal clock = 60 MHz).

#### 6.2.7.11.6 Issuance of Resume

When remote wakeup is supported and this remote wakeup function is enabled from the host, there may be a case when the device needs to resume by itself for certain reason. This section describes how to resume in such a case. Note, however, that at least 5 ms must elapse after the bus became idle before remote wakeup can be executed. Furthermore, no currents in a state prior to shift to the Suspend state of USB can be drawn from VBUS before the passage of 10 ms after the Resume signal was output.

For remote wakeup to be executed, the device must first be restored from sleep. Clear the D_SIE_IntEnb.EnNonJ bit and set the PM_Control.GoActive bit (T0), and when the PM_Control.PM_State [2:0] bits are set to "ACTIVE" after the PLL power-up time (T1) has elapsed, the internal clock starts oscillating.

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

The resume signal being sent out to the bus is stopped by clearing the D_NegoControl.SendWakeup bit about 1 ms after the device started sending out the resume signal (T4). At this point, however, the downstream port still holds the resume signal on the bus.

Therefore, set the D_NegoControl.RestoreUSB bit. After a certain predetermined time has elapsed, the downstream port stops sending back the resume signal (T5) and instead sends out a 2-bit LS-EOP (2*SE0) to switch the speed mode to that of prior to Suspend of USB. When this mode change (no longer "K") is detected, the D_XcvrControl.XcvrSelect and D_XcvrControl.TermSelect bits are both changed to the desired mode (in the present case, HS mode), and the D_NegoControl.RestoreUSB bit is cleared and the D_SIE_IntStat.RestoreCmp bit is set simultaneously with it. At this point, if the D_SIE_IntEnb.EnRestoreCmp and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above.

When Suspend of USB is initiated, the speed mode (HS or FS) is saved in the USB_Status.FSxHS bit, so that when restored by Resume, the device returns to the mode indicated by this USB_Status.FSxHS bit. At this point, HS Detection Handshaking does not need to be executed for each Resume attempted. Please note that the explanation made here refers to only the case where the speed mode prior to Suspend of USB was HS. Actually, when in FS mode, the states following T5 become FS mode and there is no significant difference in the sequence.

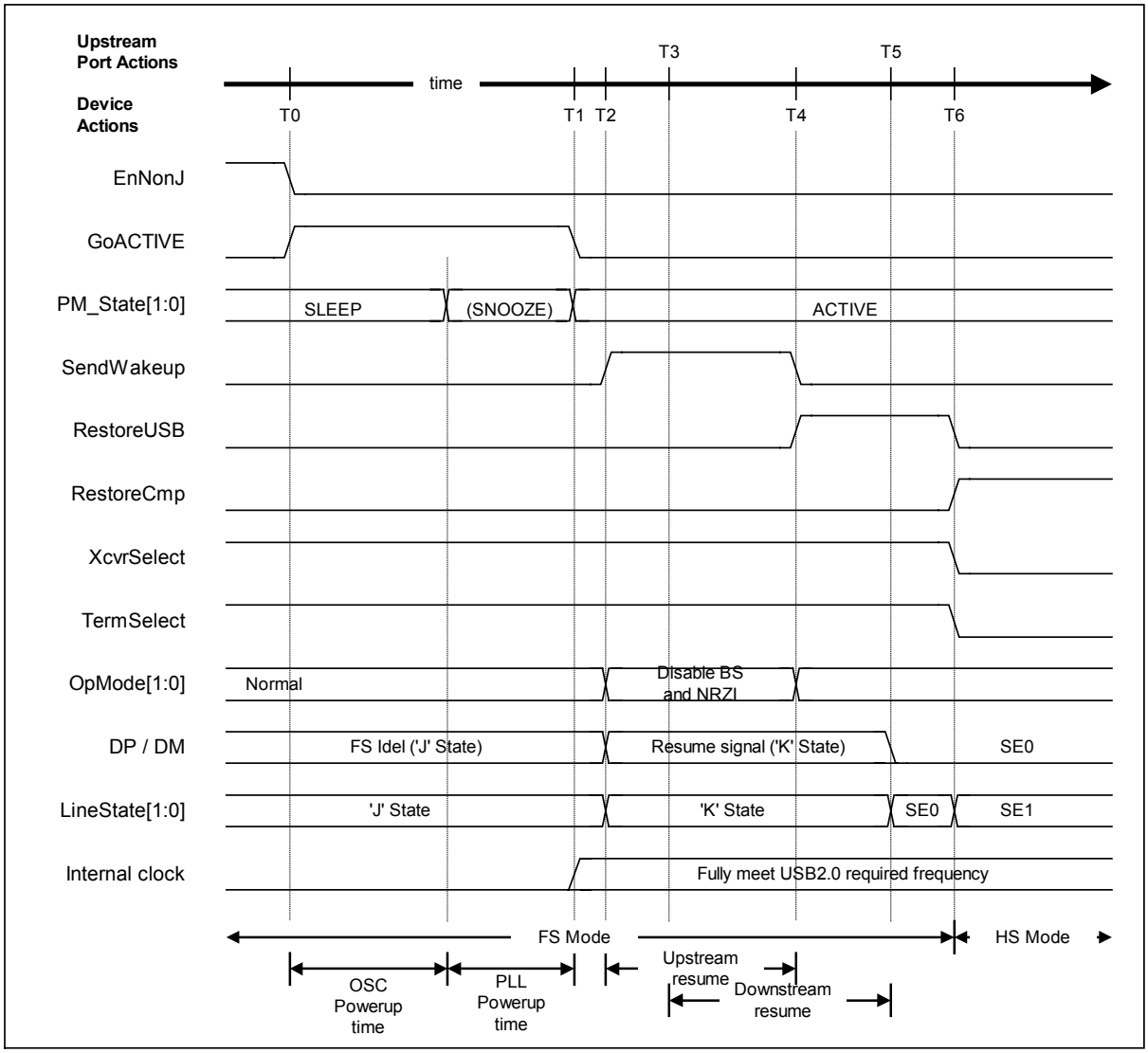



**Fig. 6.15 Assert Resume timing (HS mode)**

**Table 6.16 Assert Resume Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | Resume is initiated. Set GoActive to 1.<br>(EnNonJ must be cleared to 0 before Resume is initiated.) | 0 (reference) |
| T1 | PM_State becomes active, completing GoACTIVE. Internal clock output stabilizes. | T0 + OSC Powerup + PLL Powerup < T1 |
| T2 | Set SendWakeup to 1 to start sending out “K” for FS. Here, no currents in a state prior to Suspend of USB can be drawn from VBUS within 10 ms after that. | T0 < T2 < T0 + 10ms |
| T3 | The downstream port returns “K” for FS. | T2 < T3 < T2 + 1.0ms |
| T4 | Clear SendWakeup to 0 to finish sending out “K” for FS. After confirming “K” by 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 “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 shifts to HS mode. | T5 + 1.33us {2 Low-speed bit times} |

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

#### 6.2.7.11.7 Detection of Resume

While the LSI stipulated herein is sleeping, “J” (D_USB_Status.LineState [1:0] = J) will be observed on the bus. If “K” is observed on the bus, it means that an instruction for wakeup (instruction for Resume) from the downstream port has been received (T0). At this time, the D_SIE_IntStat.NonJ bit is set. If the D_SIE_IntEnb.EnNonJ and DeviceIntEnv.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set at this point, the XINT signal will be asserted simultaneously with the above.

The LSI sets the PM_Control.GoACTIVE bit to 1 (T1). When PM_Control.PM_State [1:0] is asserted after the OSC oscillation start time and the PLL power-up time have elapsed (T2), the internal clock is output simultaneously.

Therefore, set the D_NegoControl.RestoreUSB bit. After certain predetermined time has elapsed, the downstream port will stop sending the resume signal (T3) to switch the speed mode to that of prior to Suspend of USB. When this mode change (no longer “K”) is detected, the D_XcvrControl.XcvrSelect and D_XcvrControl.TermSelect bits both are changed to the desired mode (in the present case, HS mode), and the D_NegoControl.RestoreUSB bit is cleared and the D_SIE_IntStat.RestoreCmp bit is set simultaneously with it. At this point, if the D_SIE_IntEnb.EnRestoreCmp and DeviceIntEnb.EnD_SIE_IntStat bits have both been set and the MainIntEnb.EnDeviceIntStat bit has also been set, the XINT signal will be asserted simultaneously with the above.

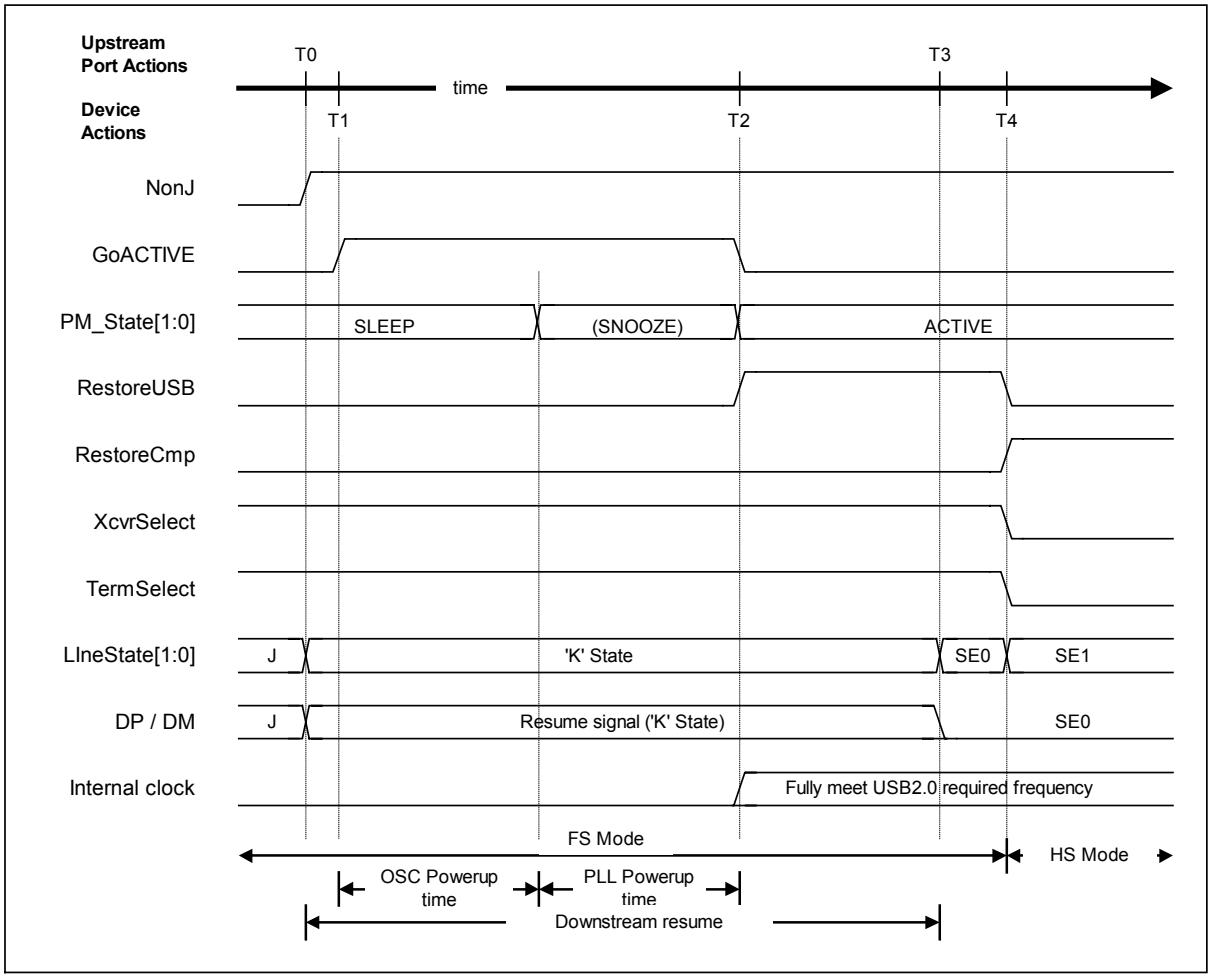



**Fig. 6.16 Detect Resume timing (HS mode)**

**Table 6.17 Detect Resume Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | The downstream port sends out “K” for FS. NonJ is set to 1. | 0 (reference) |
| T1 | Set GoActive to 1. | T1 |
| T2 | PM_State becomes active, completing GoACTIVE. Internal clock output stabilizes.<br>After confirming “K” with LineState [1:0], set RestoreUSB to 1. | T1 + OSC Powerup + PLL Powerup < T2 |
| T3 | The downstream port finishes sending out “K” for FS. At the same time, it shifts to HS mode in which it was prior to Suspend of USB. | T2 + 20ms {TDRSMDN} |
| T4 | If the mode prior to Suspend of USB was HS, the device automatically shifts to HS mode. | T5 + 1.33us {2 Low-speed bit times} |

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

#### 6.2.7.11.8 Insertion of USB Cable

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

When cable is removed or intentionally maintained in a disconnected state, make sure the D_XcvrControl.XcvrSelect and D_XcvrControl.TermSelect bits default to FS mode and HS mode, respectively.

When cable is connected while no cable has been connected (T0), VBUS shifts to ‘high' and at the same time, the D_USB_Status.VBUS bit is set (T1). If placed in a sleep state at this time, the LSI sets the PM_Control.GoACTIVE bit to 1 (T2). When PM_Control.PM_State [1:0] is asserted after the OSC oscillation start time and the PLL power-up time have elapsed (T3), the internal clock is output simultaneously. After this, since the function must shift to the FS mode temporarily in order to pretend that the FS device has been connected, set the D_XcvrControl.TermSelect bit to FS mode (T4).

Beyond this point, the downstream port sends out Reset (T5), from which HS Detection Handshaking is initiated.

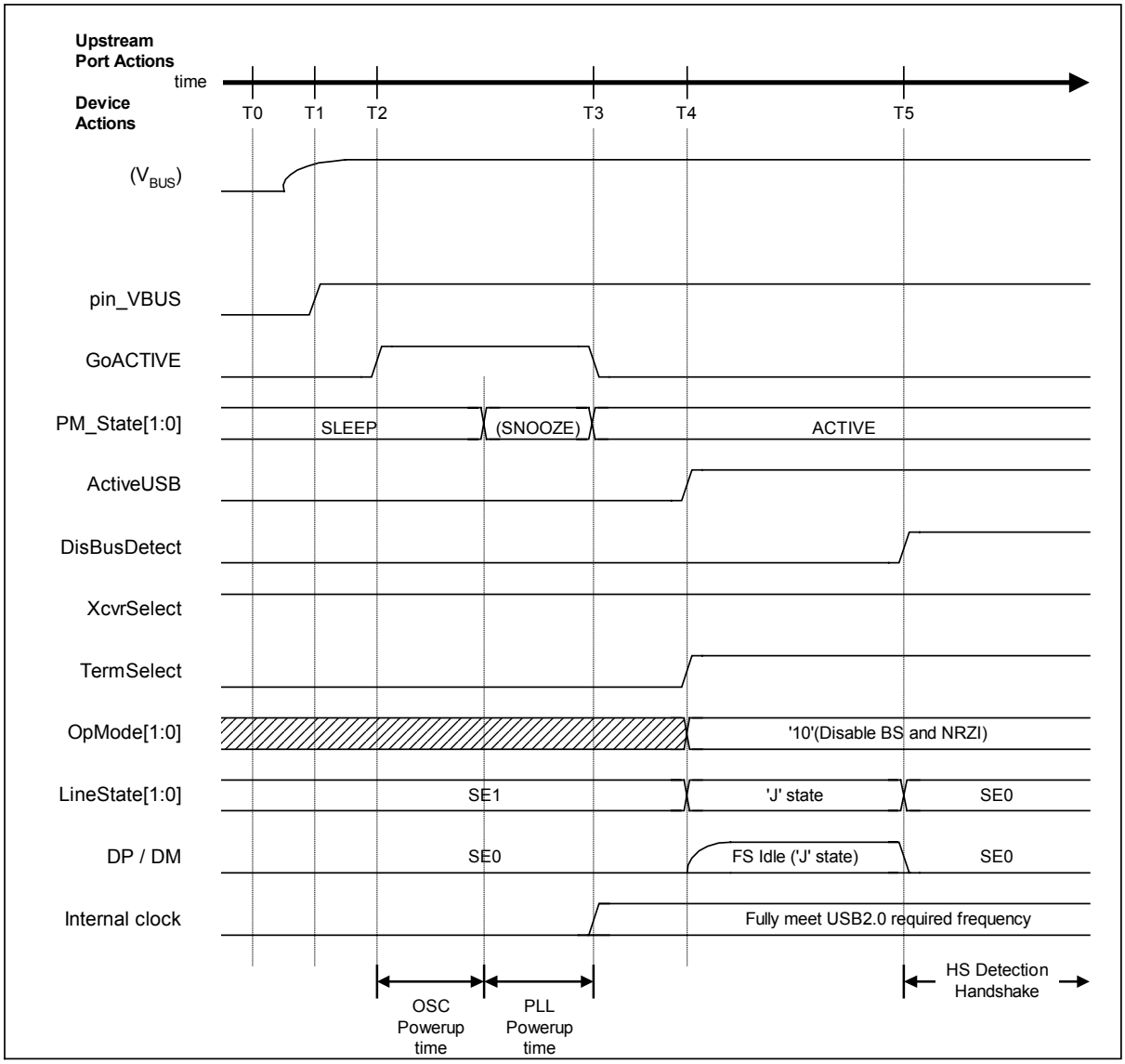



**Fig. 6.17 Device Attach timing**

**Table 6.18 Device Attach Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | No cable is inserted. | 0 (reference) |
| T1 | Cable is inserted, and the input pin VBUS shifts to 'high' . | T1 |
| T2 | Set GoActive to 1. | T2 |
| T3 | PM_State becomes active, completing GoACTIVE. Internal clock output stabilizes. | T2 + OSC Powerup + PLL Powerup < T3 |
| T4 | Set ActiveUSB to 1. Set TermSelect to 1. Set OpMode [1:0] to '00.'<br>The function is shifted to FS mode. FS termination is enabled. | T1 + 100ms {TSIGATT} < T4 |
| T5 | Reset is sent from the downstream port. Set DisBusDetect to 1. | T4 + 100ms {TATTDB} < T5 |

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

## 6.3 USB Host Control

### 6.3.1 Channels

#### 6.3.1.1 Channel Overview

The LSI stipulated herein has the host-side buffers corresponding one for one to the pipe and various setup registers used for transfers performed via those buffers, collectively referred to as "channels."

Transfer information is set in units of IRP (I/O Request Packet) for a channel. The channel divides the IRP into multiple transactions based on the set information as it executes transfer. Since a channel can switch over settings in IRP units, one channel can handle transfers for multiple endpoints.

Fig. 6.18 schematically shows a channel.

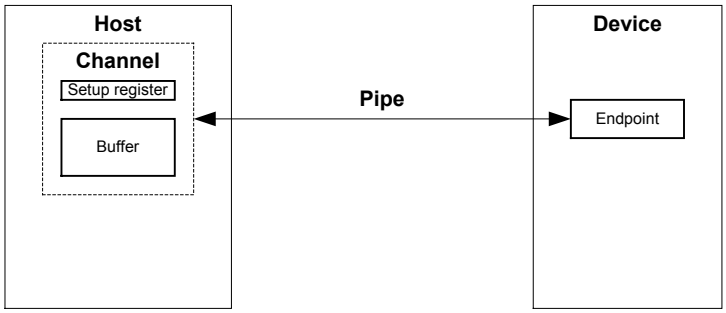



**Fig. 6.18 Conceptual diagram of a channel**

The firmware sets buffers and transfer information first and then sets transfer execution. After setting transfer execution, the firmware performs several processes by writing data to the buffer (for OUT transfer) or reading data from the buffer (for IN transfer) until processing for bytes of IRP data finishes. On the other hand, the hardware (channel) automatically divides a IRP into multiple transactions as it executes transfer. When the transfer finishes, it notifies the firmware to that effect via an interrupt.

The buffer for each channel can be allocated any memory space in the LSI's internal RAM by joining it to the FIFO area described later.

Fig. 6.19 shows the basic flow of operation for a transfer to be performed.

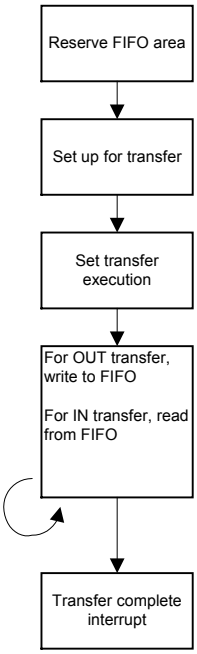



**Fig. 6.19 Basic procedure for transfer on a channel**

The LSI stipulated herein has a total of six channels, consisting of a channel that performs only a control transfer (CH0), a channel that performs only a bulk transfer (CHa), and channels that perform bulk/interrupt transfers (CHb, CHc, CHd, and CHe). Here, channel CH0 is referred to as a control-only channel, while channels CHa, CHb, CHc, CHd, and CHe are referred to as general-purpose channels.

Each channel has fixed basic setup items determined by the USB-defined interface and variable control items and status used for control of each transfer performed.The basic setup items should be set when, for example, the chip is initialized or USB-defined interfaces are switched over. Reserve memory space for the buffer for any channel in question by joining it to the FIFO area described later. Note that the maximum number of interrupt transfers that can be set at the same time is 4.

Table 6.19 lists the transfer types that can be handled by each channel.

**Table 6.19 Available Transfer Types**

| Channel | Available transfer type | Remark |
|---|---|---|
| CH0 | Control transfer | Control transfer support function (described later) may be used. |
| CHa | Bulk transfer | Bulk Only support function (described later) may be used. |
| CHb, CHc CHd, CHe | Bulk transfer<br>Interrupt transfer | |

#### 6.3.1.2 Control-only Channel

The LSI stipulated herein uses the control-only channel (CH0) to perform control transfers. Therefore, when performing a control transfer to or from multiple endpoints, it time-multiplexes the channel CH0 so that multiple packets can be sent on a single channel separated only in time.

Channel CH0 has fixed basic setup items determined by the USB-defined interface as well as variable control items and statuses to be controlled for each transfer performed. Set the basic setup items when, for example, initializing the chip or changing USB-defined interfaces.

Set AREAn{n=0-5}StartAdrs_H,L and AREAn{n=0-5}EndAdrs_H,L to reserve memory space for the FIFO area to be used, and after initializing the FIFO area with AREAn{n=0-5}FIFO_Clr, set up AREAn{n=0-5}Join_1.JoinEP0CH0 for the FIFO area used. No data transfers via a FIFO area can be performed until this joining process is executed.

Table 6.20 lists the basic setup items of the control-only channel (CH0).

**Table 6.20 Basic Setup Items of the Control.only Channel**

| Item | Register/Bit | Description |
|---|---|---|
| Transfer rate | H_CH0Config_0.SpeedMode | Sets the transfer rate (HS, FS or LS) of the endpoint corresponding to channel CH0. |
| Toggle sequence bit | H_CH0Config_0.Toggle | Sets the initial value of a toggle sequence bit with which a transaction is initiated. While the transaction is underway and after the transaction is completed, it indicates the state of the toggle sequence bit. |
| Transaction type | H_CH0Config_1.TID | Sets the transaction type (SETUP, IN or OUT) issued on channel CH0. |
| Max. packet size | H_CH0MaxPktSize | Sets the Max. packet size to 8 for operation in LS mode, or to 8, 16, 32, or 64 for operation in FS mode. Or set it to 64 for operation in HS mode. |
| USB address | H_CH0FuncAdrs.FuncAdrs | Sets the USB address of the function (including an endpoint) managed by channel CH0 to any value between 0x0 and 0xF. |
| Endpoint number | H_CH0FuncAdrs.EP_Number | Sets the endpoint number of the endpoint corresponding to channel CH0 to any value between 0x0 and 0xF. |
| Hub address | H_CH0HubAdrs.HubAdrs | Sets the USB address of the hub that performs split transactions. |
| Port number | H_CH0HubAdrs.Port | Sets the IRP data quantity on channel CH0 in byte units. |
| Number of IRP data bytes | H_CH0TotalSize_H,<br>H_CH0TotalSize _L | Sets the area allocated to channel CH0 by a FIFO address. |
| FIFO area | AREAn{n=0-5}StartAdrs_H,<br>AREAn{n=0-5}SStartAdrs _L,<br>AREAn{n=0-5}SEndAdrs_H,<br>AREAn{n=0-5}SEndAdrs _L | Sets the area allocated to channel CH0 by a FIFO address. Make sure the allocated FIFO is equal to or greater than Max. packet size of channel CH0.<br>For details on how to allocate the FIFO area, refer to the relevant section in Chapter 6 on FIFOs. |
| FIFO area join | AREAn{n=0-5}Join_1.JoinEP0 CH0 | Joins channel CH0 to the allocated area.<br>To use the control transfer support function, join channel CH0 to AREA0. |
| Setup data | H_CH0SETUP_x(x=0-7) | Sets 8 bytes of data to be transmitted by a setup transaction. |

#### 6.3.1.3 General-purpose Channels

The general-purpose channels permit the transaction direction, USB address, and endpoint number to be set as desired, so that channels can be corresponded one for one to a maximum of five endpoints at the same time. As for the control-only channel, these channels can be time-multiplexed in IRP units, making it possible to perform transfers to or from many more endpoints than the maximum five.

Each channel has fixed basic setup items determined by the USB-defined interface and variable control items and status used for control of each transfer performed. The basic setup items should be set when, for example, the chip is initialized or USB-defined interfaces are switched over. Set AREAn{n=0-5}StartAdrs_H,L and AREAn{n=0-5}EndAdrs_H,L to reserve memory space for the FIFO area to be used. After initializing the FIFO area with AREAn{n=0-5}FIFO_Clr, set up AREAn{n=0-5}Join_1.JoinEPxCHx{x=a-e} for the FIFO area used. No data transfers via a FIFO area can be performed until this joining process is executed.

Table 6.21 lists the basic setup items of the general-purpose channels. Set up these times as appropriate for the contents of definitions of the USB-defined interface. Also, enable the settings made, to configure a USB-defined interface.

**Table 6.21 Basic Setup Items of the General.purpose Channels**

| Item | Register/Bit | Description |
|---|---|---|
| Transfer rate | H_CHx{x=a-e}Config_0.SpeedMode | Sets the transfer rate (HS, FS or LS) of the endpoint corresponding to each channel. |
| Toggle sequence bit | H_CHx{x=a-e}Config_0.Toggle | Sets the initial value of a toggle sequence bit with which a transaction is initiated. While the transaction is underway and after the transaction is completed, it indicates the state of the toggle sequence bit. |
| Transaction type | H_CHx{x=a-e}Config_1.TID | Sets the transaction type (IN or OUT) issued on each channel. |
| Transfer type | H_CHx{x=b-e}Config_1.TranType | Sets the transfer type (bulk or interrupt) on each channel. |
| Max. packet size | H_CHx{x=a-e}MaxPktSize_H,<br>H_CHx{x=a-e}MaxPktSize_L | Sets the Max. packet size on each channel to any value between 1 byte to 512 bytes. |
| Hub address | H_CHx{x=a-e}HubAdrs.HubAdrs | Sets the USB address of the hub that performs split transactions. |
| Port number | H_CHx{x=a-e}HubAdrs.Port | Sets the port number of the hub that performs split transactions. |
| USB address | H_CHx{x=a-e}FuncAdrs.FuncAdrs | Sets the USB address of the function (including an endpoint) managed by each channel to any value between 0x0 and 0xF. |
| Endpoint number | H_CHx{x=a-e}FuncAdrs.EP_Number | Sets the endpoint number of the endpoint corresponding to each channel to any value between 0x0 and 0xF. |
| Number of IRP data bytes | H_CHx{x=a-e}TotalSize_HH,<br>H_CHx{x=a-e}TotalSize_HL,<br>H_CHx{x=a-e}TotalSize_LH,<br>H_CHx{x=a-e}TotalSize_LL | Sets the IRP data quantity on each channel in bytes. |
| Token issuance interval | H_CHx{x=b-e}Interval_H,<br>H_CHx{x=b-e}Interval_L | Sets the interrupt (period) at which tokens are issued in interrupt transfer. |
| FIFO area | AREAn{n=0-5}StartAdrs_H,<br>AREAn{n=0-5}StartAdrs_L,<br>AREAn{n=0-5}EndAdrs_H,<br>AREAn{n=0-5}EndAdrs_L | Sets the area allocated to each channel by a FIFO address.<br>Make sure the allocated FIFO is equal to or greater than Max. packet size of each channel. Also be aware that the size of the FIFO area affects the throughput of data transfer.<br>For details on how to allocate the FIFO area, refer to the relevant section in Chapter 6 on FIFOs. |
| FIFO area join | AREAn{n=0-5}Join_1.JoinEPxCHx{x=a-e} | Joins each channel to its allocated area.<br>To use the bulk-only support function, join channel CHa to AREA1. |

### 6.3.1.4 Example for Using Channels

#### 6.3.1.4.1 For One Storage Device Connected

Fig. 6.20 shows an example for using channels for the case where the system has a storage device compatible with USB Mass Storage Class (BulkOnly Transport Protocol) connected to it (e.g., a hard disk).

The bulk IN and bulk OUT transfers used in this class can be successively processed. The IRP for control transfer uses CH0. On the other hand, the IRP for bulk IN transfer and the IRP for bulk OUT transfer use CHa one after the other. CHa has the function to automatically manage a series of Mass Storage Class (BulkOnly Transport Protocol) transports (see 6.3.8.), such as command transport (CBW), data transport, and status transport (CSW). This function helps to reduce the transfer processing load of the CPU and increase the efficiency of transfer.

Note that if the bulk-only support function does not need to be used, the IRP for bulk IN transfer and the IRP for bulk OUT transfer can be assigned separately to other general-purpose channels (e.g., CHb and CHc).

The hardware schedules transfers for the IRPs set on channels (see 6.3.2) and executes transactions.

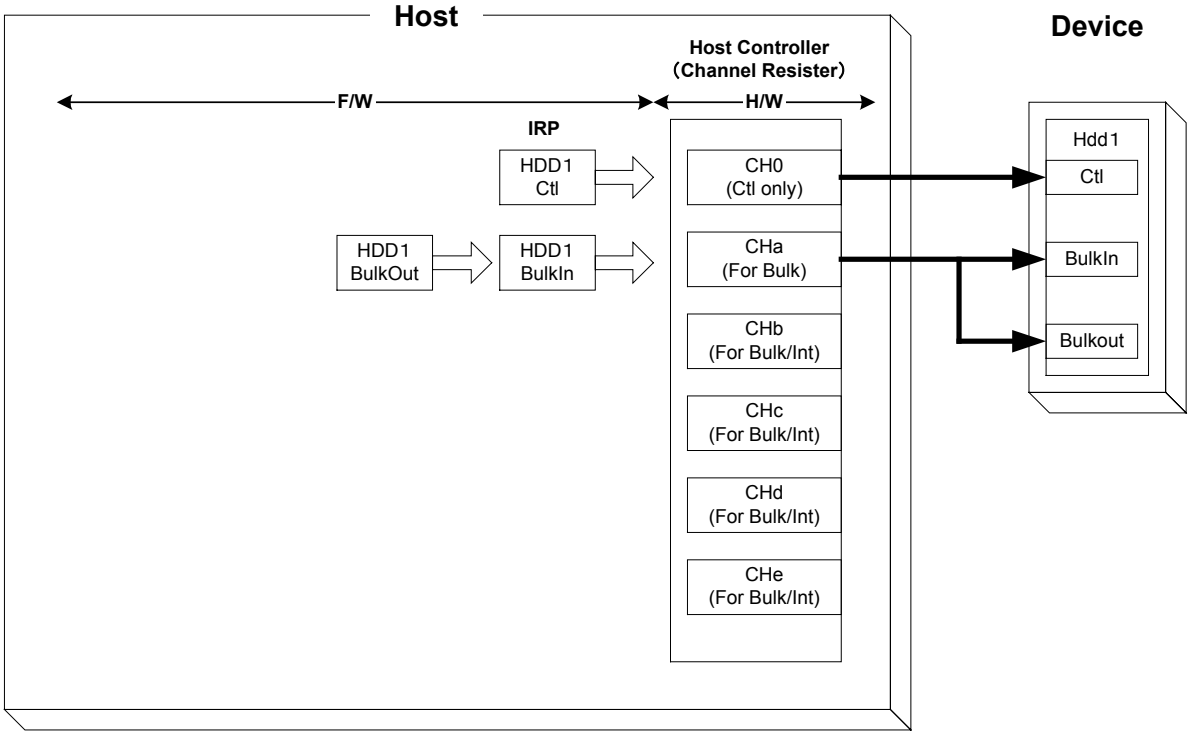



**Fig. 6.20 Example for using channels**
**(when the system has one storage device connected)**

#### 6.3.1.4.2 Connecting a Storage Device via a Hub

Fig. 6.21 shows an example for using channels for the case where the system has one storage device compatible with USB Mass Storage Class (BulkOnly Transport Protocol) connected to it via a hub (e.g., a hard disk and USB memory).

The bulk IN and bulk OUT transfers used in this class can be successively processed. The IRP for control transfer uses CH0. The IRP for interrupt IN transfer uses general-purpose channels that are assigned to it (e.g., CHd and CHe). On the other hand, the IRP for bulk IN transfer and the IRP for bulk OUT transfer use CHa. CHa has the function to automatically manage a series of Mass Storage Class (BulkOnly Transport Protocol) transports (see 6.3.8.), such as command transport (CBW), data transport, and status transport (CSW). This function helps to reduce the transfer processing load of the CPU and increase the efficiency of transfer.

Note that if the bulk-only support function need not to be used, the IRP for bulk IN transfer and the IRP for bulk OUT transfer can be assigned separately to other general-purpose channels (e.g., CHb and CHc).

The hardware schedules transfers for the IRPs set on channels and executes transactions (see 6.3.2).

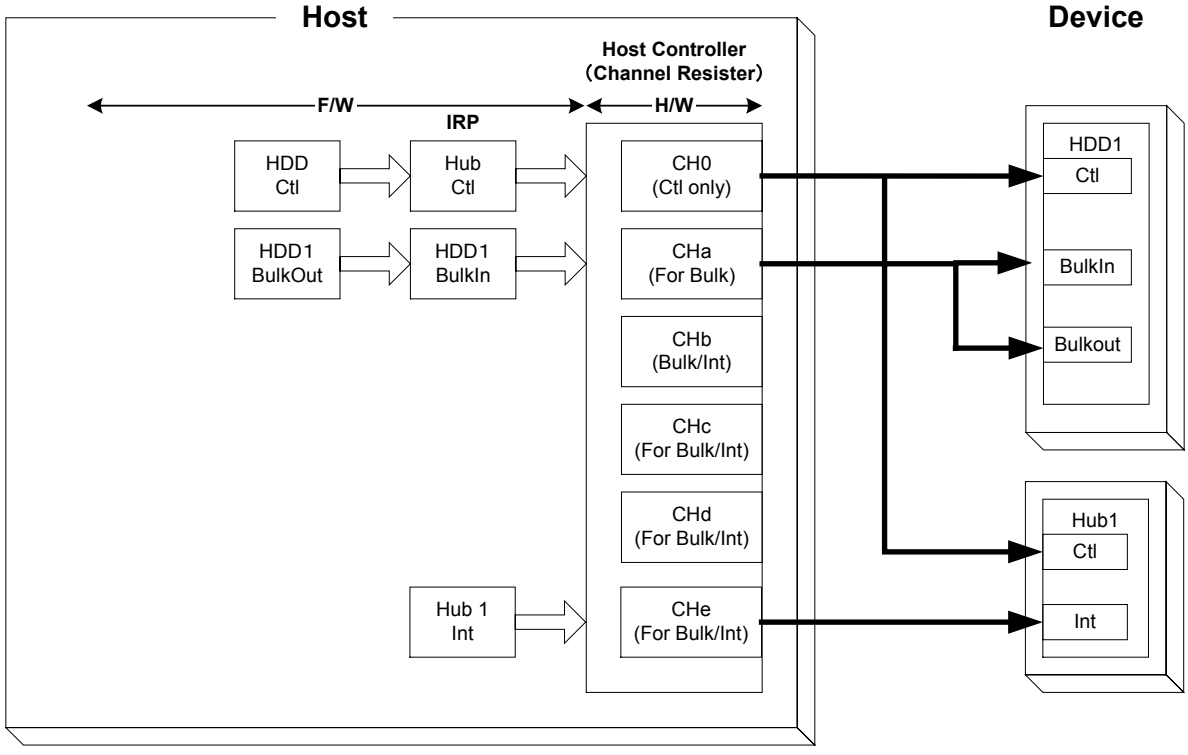



**Fig. 6.21 Example for using channels**
**(when system has one storage device connected via a hub)**

### 6.3.2 Scheduling

The hardware selects one of the channels which have had transfer executions set (H_CHx{x=0,a-e}Config_0.TranGo), and determines whether the transfer set for that channel can be executed. When found executable, it executes one transaction according to settings made. When the transaction finishes, the hardware selects another channel, determines whether execution is possible, and executes a transaction in the same way. That way, by selecting a channel, determining whether execution is possible, and executing a transaction repeatedly, the hardware performs transfers to or from multiple endpoints.

Table 6.22 lists the control items associated with scheduling control on channel CH0. Table 6.23 lists the scheduling setup items for general-purpose channels.

**Table 6.22 Setup Items for Scheduling on Channel CH0**

| Item | Register/Bit | Description |
|---|---|---|
| Transfer execution | H_CH0Config_0.TranGo | Sets transfer execution on channel CH0. Performs a transfer according to settings made on channel CH0. |

**Table 6.23 Setup Items for Scheduling on General.purpose Channels**

| Item | Register/Bit | Description |
|---|---|---|
| Transfer execution | H_CHx{x=a-e}Config_0.TranGo | Sets transfer execution on each channel. Performs a transfer according to settings made on each channel. |

### 6.3.3 Transactions

The LSI provides transaction execution functions in hardware and provides the firmware with the interfaces necessary to execute transactions. The interfaces for the firmware are implemented as control and status registers and the interrupt signals that are asserted by a status. For details about settings necessary to assert an interrupt by status, refer to the relevant section on registers.

The hardware selects the channel, and determine whether the transfer executions set channel can be executed. When found executable, it executes transaction according to setting mode. The LSI also issues a status to the firmware for each transaction executed. However, the firmware does not always need to manage each individual transaction. For an IN channel, for example, the firmware can read data from the FIFO through the CPU interface (DMA read or register read) to create a free space in the FIFO, thereby allowing IN transactions to be automatically executed in succession. For an OUT channel also, the firmware can write data to the FIFO through the CPU interface (DMA write or register write) to create valid data in the FIFO, thereby allowing OUT transactions to be automatically executed in succession.

Table 6.24 lists the control items and statuses associated with transaction control on channel CH0. Table 6.25 lists the control items and statuses associated with transaction processing on general-purpose channels (CHa, CHb, CHc, CHd, and CHe).

**Table 6.24 Control Items and status for Channel CH0**

| Item | Register/Bit | Description |
|---|---|---|
| Transaction status | H_CH0IntStat. TotalSizeCmp,<br>H_CH0IntStat.TranACK,<br>H_CH0IntStat.TranErr,<br>H_CH0IntStat.ChangeCondition | Shows the result of transaction. |
| Transaction condition code | H_CH0ConditionCode | Shows the result of transaction in detail. |

**Table 6.25 Control Items and status for General.purpose Channels**

| Item | Register/Bit | Description |
|---|---|---|
| Transaction status | H_CHx{x=a-e}IntStat.TotalSizeCmp,<br>H_CHx{x=a-e}IntStat.TranACK,<br>H_CHx{x=a-e}IntStat.TranErr,<br>H_CHx{x=a-e}IntStat.ChangeCondition | Shows the result of transaction. |
| Transaction condition code | H_CHx{x=a-e}ConditionCode | Shows the result of transaction in detail. |

#### 6.3.3.1 SETUP Transactions

In the CH0 basic setup register, set the transaction type (H_CH0Config_1.TID) to SETUP. Set other basic setup items as accordingly, write setup data (8 bytes) to the H_CH0SETUP_0–7 registers, and set transfer execution (H_CH0Config_0.TranGo). The channel will thereby be recognized as the target for which USB transfer is to be scheduled by the hardware, so that when this channel is selected, the hardware determines the remaining frame time and executes a SETUP transaction.

In SETUP transactions, the data in the H_CH0_SETUP_0–7 registers are used, where the data packets are 8 bytes in length. When ACK was received for a SETUP transaction, the hardware issues an ACK status (H_CH0IntStat.TranACK bit) to the firmware. If correct response was not received for a SETUP transaction, the hardware performs a retry process. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CH0Control.TranGo and sets condition code (H_CH0ConditionCode) accordingly. It then issues a ChangeCondition status (H-CH0IntStat.ChangeCondition bit) to the firmware.

Note that this SETUP transaction can be executed even when the channel CH0 is not joined to a FIFO area.

Fig. 6.22 shows how a SETUP transaction is performed in host mode. In (a), the LSI issues a SETUP token addressed to the endpoint 0 that resides in the local node. In (b), the LSI proceeds to send a data packet of 8 bytes in length. In (c), upon receiving ACK, the LSI automatically sets the relevant register and issues a status to the firmware.

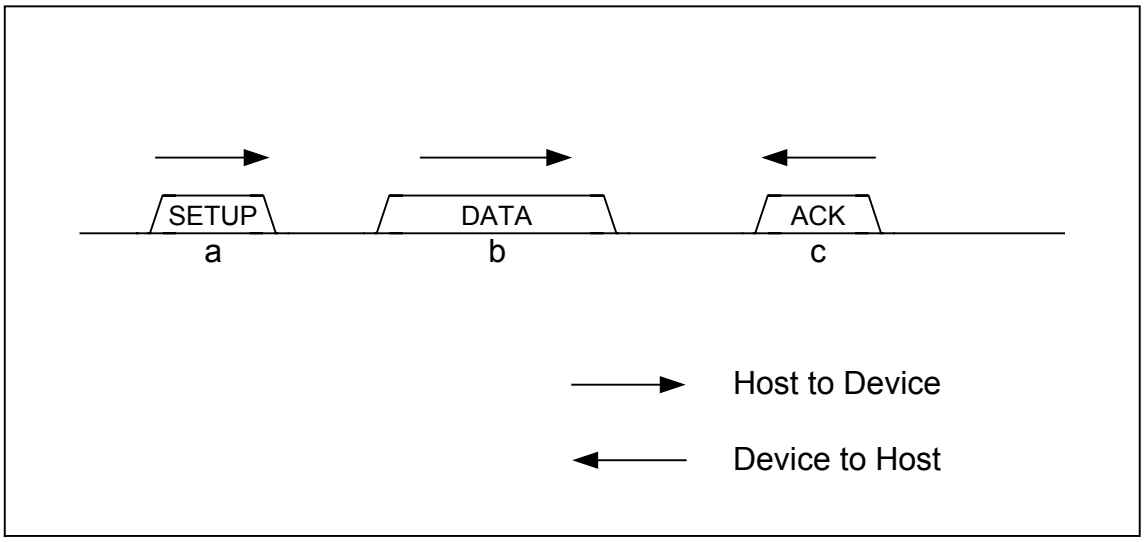



**Fig. 6.22 SETUP transaction in host mode**

#### 6.3.3.2 Bulk OUT Transaction

In the CHx basic setup register, set the transfer type (H_CHx{x=b-e}Config_1.TranType) to Bulk and the transaction type (H_CHx{x=b-e}Config_1.TID) to OUT. Set other basic setup items as accordingly, and set transfer execution (H_CHx{x=b-e}Config_0.TranGo). The channel will thereby be recognized as the target for which USB transfer is to be scheduled by the hardware, so that when this channel is selected, the hardware determines the number of valid data bytes in the FIFO and the remaining frame time and thereby executes a bulk OUT transaction. Note that CHa is usable for only Bulk transfer. There is no need to set the transfer type.

The data length of a data packet is H_CHx{x=a-e}Max. packet size_H,L or H_CHx{x=a-e}TotalSize_HH,HL,LH,LL whichever is smaller. When ACK was received for a bulk OUT transaction, the hardware issues an ACK status (H_CHx{x=a-e}IntStat.TranACK bit) to the firmware. It also updates the FIFO, and assuming the transmitted data as having been transmitted, frees the reserved area. If NAK was received for a bulk OUT transaction, the hardware does not update the FIFO, nor does it free the reserved area. Therefore, if the channel is selected again, the hardware executes the same transaction. If STALL was received for a bulk OUT transaction, the hardware terminates the transfer by automatically clearing H_CHx{x=0,a-e}Config_0.TranGo and sets the condition code (H_CHx{x=a-e}ConditionCode) to STALL. It then issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware. The FIFO is not updated, nor is the reserved area freed. If correct response was not received for a bulk OUT transaction, the hardware neither updates the FIFO nor frees the reserved area, sets the condition code (H_CHx{x=a-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=a-e}IntStat.TranErr bit) to the firmware. It then performs a retry process. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CHx{x=a-e}Control.TranGo and sets condition code (H_CHx{x=a-e}ConditionCode) as accordingly. It then issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware.

Fig. 6.23 shows how a Bulk OUT transaction is performed in host mode in cases in which the transaction is completed. In (a), the LSI issues an OUT token addressed to the OUT-direction endpoint that resides in the local node. In (b), the LSI proceeds to send a data packet within Max. packet size. In (c), upon receiving ACK, the LSI automatically sets the relevant register and issues a status to the firmware.

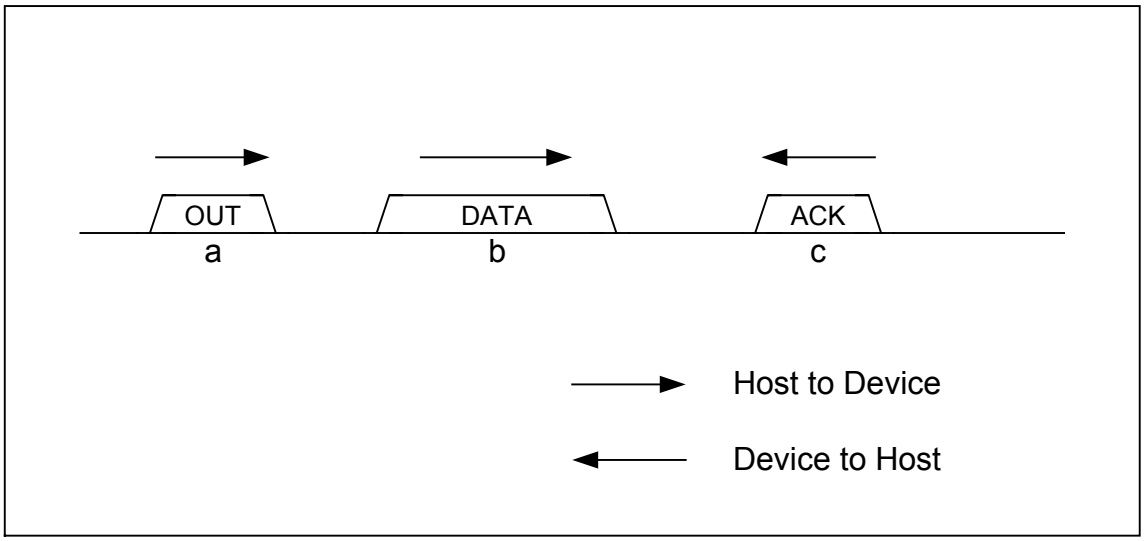



**Fig. 6.23 OUT transaction in host mode**

### 6.3.3.3 Interrupt OUT Transaction

In the CHx basic setup register, set the transfer type (H_CHx{x=b-e}Config_1.TranType) to Interrupt and the transaction type (H_CHx{x=b-e}Config_1.TID) to OUT. Furthermore, set a token issuance interval (H_CHx{x=b-e}Interval_H,L), and after setting other basic setup items as accordingly, set transfer execution (H_CHx{x=b-e}Config_0.TranGo). The channel will thereby be recognized as the target for which USB transfer is to be scheduled by the hardware, so that when this channel is selected, the hardware determines the token issuance interval (H_CHx{x=b-e}Interval_H,L), the number of valid data bytes in the FIFO, and the remaining frame time and thereby executes an interrupt OUT transaction.

The data length of a data packet is H_CHx{x=b-e}Max. packet size_H,L or H_CHx{x=b-e}TotalSize_HH,HL,LH,LL whichever is smaller. When ACK is received for an interrupt OUT transaction, the hardware issues an ACK status (H_CHx{x=b-e}IntStat.TranACK bit) to the firmware. It also updates the FIFO, and assuming the transmitted data as having been transmitted, frees the reserved area. If NAK is received for an interrupt OUT transaction, the hardware does not update the FIFO, nor does it free the reserved area. Therefore, if the channel is selected again, the hardware executes the same transaction. If STALL is received for an interrupt OUT transaction, the hardware terminates the transfer by automatically clearing H_CHx{x=b-e}Config_0.TranGo and sets the condition code (H_CHx{x=b-e}ConditionCode) to STALL. It then issues a ChangeCondition status (H-CHx{x=b-e}IntStat.ChangeCondition bit) to the firmware. The FIFO is not updated, nor is the reserved area freed. If correct response was not received for an interrupt OUT transaction, the hardware neither updates the FIFO nor frees the

reserved area, sets the condition code (H_CHx{x=a-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=a-e}IntStat.TranErr bit) to the firmware. It then performs a retry process. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CHx{x=b-e}Control.TranGo and sets condition code (H_CHx{x=d-h}ConditionCode) as accordingly. It then issues a ChangeCondition status (H-CHx{x=b-e}IntStat.ChangeCondition bit) to the firmware.

#### 6.3.3.4 Bulk IN Transaction

In the CHx basic setup register, set the transfer type (H_CHx{x=b-e}Config_1.TranType) to Bulk and the transaction type (H_CHx{x=a-e}Config_1.TID) to IN. Set other basic setup items as accordingly, and set transfer execution (H_CHx{x=a-e}Config_0.TranGo). The channel will thereby be recognized as the target for which USB transfer is to be scheduled by the hardware, so that when this channel is selected, the hardware determines the free space in the FIFO and the remaining frame time and thereby executes a bulk IN transaction. Note that CHa is usable for only Bulk transfer. There is no need to set the transfer type.

The expected data length of the data packet to be received is H_CHx{x=a-e}Max. packet size_H,L or H_CHx{x=a-e}TotalSize_HH,HL,LH,LL whichever is smaller. When all data bytes are received correctly in a bulk IN transaction, the hardware responds with ACK to complete the transaction. It also issues an ACK status (H_CHx{x=a-e}IntStat.TranACK bit) to the firmware. It further updates the FIFO, and assuming the data as having been received, reserves an area. If the received data length in a bulk IN transaction is less than the expected data length, the hardware terminates the transfer by automatically clearing H_CHx{x=a-e}Config_0.TranGo and responds with ACK. It also sets the condition code (H_CHx{x=a-e}ConditionCode) to DataUnderrun and then issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware. It further updates the FIFO, and assuming the data has been received, reserves an area. If NAK is received in a bulk IN transaction, the hardware does not issue a status. Nor does it update the FIFO. If STALL is received in a bulk IN transaction, the hardware terminates the transfer by automatically clearing H_CHx{x=a-e}Config_0.TranGo and sets the condition code (H_CHx{x=a-e}ConditionCode) to STALL. It then issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware. The FIFO is not updated. If the received data length in a bulk IN transaction is larger than the expected data length, the hardware terminates the transfer by automatically clearing H_CHx{x=a-e}Config_0.TranGo. It does not respond. It also sets the condition code (H_CHx{x=a-e}ConditionCode) to DataOverrun, and issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware. The FIFO is not updated. When a toggle mismatch occurs in a bulk IN transaction, the hardware responds with ACK. It also sets the condition code (H_CHx{x=a-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=a-e}IntStat.TranErr bit) to the firmware. The FIFO is not updated. When a time-out error, CRC error, bit stuffing error, or PID error (including unexpected PID) occurs in a bulk IN transaction, the hardware does not respond. It sets the condition code (H_CHx{x=a-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=a-e}IntStat.TranErr bit) to the firmware. The FIFO is not updated. When an error that would cause the condition code (H_CHx{x=a-e}ConditionCode) to be set to RetryError occurs, the

hardware performs a retry process. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CHx{x=a-e}Control.TranGo and issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware.

Fig. 6.24 shows how a Bulk IN transaction is performed in host mode in cases in which the transaction is completed. In (a), the LSI issues an IN token addressed to the IN-direction endpoint that resides in the local node. In (b), if the endpoint can respond to this IN transaction, it sends a data packet within Max. packet size. In (c), the LSI responds with ACK. It then automatically sets the relevant register and issues a status to the firmware.

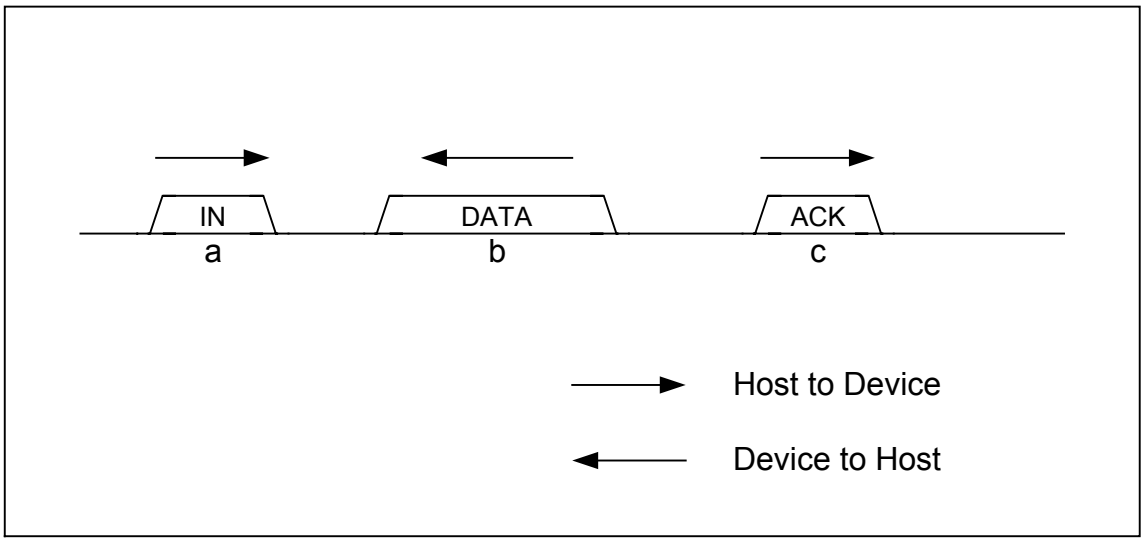



**Fig. 6.24 OUT transaction in host mode**

### 6.3.3.5 Interrupt IN Transaction

In the CHx basic setup register, set the transfer type (H_CHx{x=b-e}Config_1.TranType) to Interrupt and the transaction type (H_CHx{x=b-e}Config_1.TID) to IN. In addition, set a token issuance interval (H_CHx{x=b-e}Interval_H,L), and after setting other basic setup items as accordingly, set transfer execution (H_CHx{x=b-e}Config_0.TranGo). The channel will thereby be recognized as the target for which USB transfer is to be scheduled by the hardware, so that when this channel is selected, the hardware determines the token issuance interval (H_CHx{x=b-e}Interval_H,L), the free space in the FIFO, and the remaining frame time and thereby executes an interrupt IN transaction.

The expected data length of the data packet to be received is H_CHx{x=b-e}Max. packet size_H,L or H_CHx{x=b-e}TotalSize_HH,HL,LH,LL whichever is smaller. When all data bytes are received correctly in an interrupt IN transaction, the hardware responds with ACK to complete the transaction. It also issues an ACK status (H_CHx{x=b-e}IntStat.TranACK bit) to the firmware. It further updates the FIFO, and assuming the data has been received, reserves an area. If the received data length in an interrupt IN transaction is larger than the expected data length, the hardware terminates the transfer by automatically clearing H_CHx{x=b-e}Config_0.TranGo and responds with ACK. It also sets the condition code (H_CHx{x=b-e}ConditionCode) to DataUnderrun and then issues a ChangeCondition status (H-CHx{x=a-e}IntStat.ChangeCondition bit) to the firmware.

It further updates the FIFO, and assuming the data has been received, reserves an area. If NAK is received in an interrupt IN transaction, the hardware does not issue a status. Nor does it update the FIFO. The next transaction is performed in the next cycle. If STALL is received in an interrupt IN transaction, the hardware terminates the transfer by automatically clearing H_CHx{x=b-e}Config_0.TranGo and sets the condition code (H_CHx{x=b-e}ConditionCode) to STALL. It then issues a ChangeCondition status (H-CHx{x=b-e}IntStat.ChangeCondition bit) to the firmware. The FIFO is not updated. If the received data length in an interrupt IN transaction is larger than the expected data length, the hardware terminates the transfer by automatically clearing H_CHx{x=b-e}Config_0.TranGo. It does not respond. It also sets the condition code (H_CHx{x=b-e}ConditionCode) to DataOverrun, and issues a ChangeCondition status (H-CHx{x=b-e}IntStat.ChangeCondition bit) to the firmware. The FIFO is not updated. If a toggle mismatch occurrs in an interrupt IN transaction, the hardware responds with ACK. It also sets the condition code (H_CHx{x=a-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=a-e}IntStat.TranErr bit) to the firmware. The FIFO is not updated. If a time-out error, CRC error, bit stuffing error, or PID error (including unexpected PID) occurrs in an interrupt IN transaction, the hardware does not respond. It sets the condition code (H_CHx{x=b-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=b-e}IntStat.TranErr bit) to the firmware. The FIFO is not updated. If an error occurrs that causes the condition code (H_CHx{x=a-e}ConditionCode) to be set to RetryError, the hardware performs a retry process in the next cycle. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CHx{x=b-e}Control.TranGo and issues a ChangeCondition status (H-CHx{x=b-e}IntStat.ChangeCondition bit) to the firmware.

#### 6.3.3.6 PING Transaction

On channels in which bulk OUT transactions or control OUT transactions are performed, the LSI executes a PING transaction when it is operating in HS mode.

If NYET or NAK or no response is received for an OUT transaction, the hardware shifts to a state in which it can execute a PING transaction. If ACK is received for a PING transaction, the hardware returns to a state in which it can execute an OUT transaction. No status indications are issued to the firmware. If NAK is received for a PING transaction, the hardware remains in a state in which it can execute a PING transaction. It does not issue a status to the firmware. If STALL is received for a PING transaction, the hardware terminates the transfer by automatically clearing H_CHx{x=0,a-h}Control.TranGo and sets the condition code (H_CHx{x=0,a-h}ConditionCode) to STALL. It then issues a ChangeCondition status (H-CHx{x=0,a-h}IntStat.ChangeCondition bit) to the firmware. If correct response is not received for a PING transaction, the hardware sets the condition code (H_CHx{x=0,a-e}ConditionCode) to RetryError, and issues a TranErr status (H-CHx{x=0,a-e}IntStat.TranErr bit) to the firmware. In this case, it performs a retry process. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CHx{x=0,a-e}Control.TranGo and issues a ChangeCondition status (H-CHx{x=0,a-e}IntStat.ChangeCondition bit) to the firmware.

In no event will the FIFO be updated in a PING transaction.

Fig. 6.25 shows how a PING transaction is acknowledged by an ACK in host mode. In (a), the LSI issues a PING token addressed to the OUT-direction endpoint that resides in the local node. In (b), if the endpoint has a free space equivalent to Max. packet size, the device responds to this PING transaction with ACK.

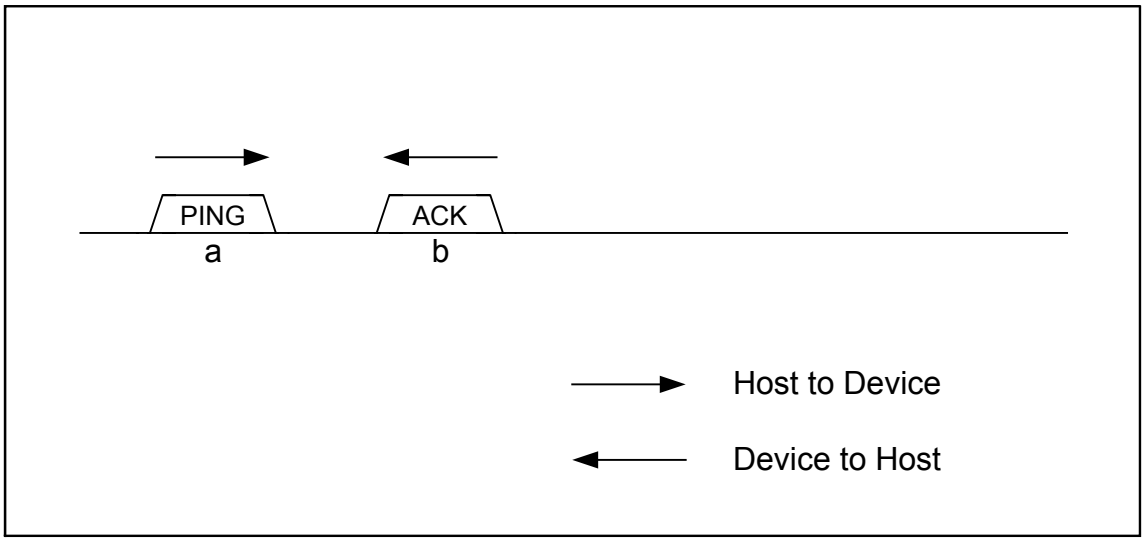



**Fig. 6.25 PING transaction in host mode**

### 6.3.3.7 Low-Speed (LS) Transaction

Transfers to or from LS devices are accomplished using control or interrupt transfers.

If the downstream port has an LS device connected, the host operates in LS mode. The host sets the transfer rate for the channel used (H_CHx{x=0,a-e}Config_0.SpeedMode) to LS, and thereby executes a transaction with LS bit time.

On the other hand, if the downstream port has a full-speed (FS) hub connected and the downstream port of the hub has a LS device connected, the host operates in FS mode. The host sets the transfer rate for the channel used (H_CHx{x=0,a-e}Config_0.SpeedMode) to LS, and thereby transmits downstream packets to the corresponding endpoint after attaching a preamble at the beginning of all packets. The preamble is transmitted with FS bit time, and the downstream packets following it are transmitted with LS bit time.

Fig. 6.26 shows how an interrupt OUT transaction will be performed for the case in which the transaction finishes without errorswithout errors. In (a), the LSI issues an OUT token addressed to the OUT-direction endpoint that resides in the local node after attaching a preamble at the beginning of it. In (b), the LSI transmits a data packet within Max. packet size after attaching a preamble at the beginning of it. In (c), upon receiving ACK, the LSI automatically sets the relevant register and issues a status to the firmware.

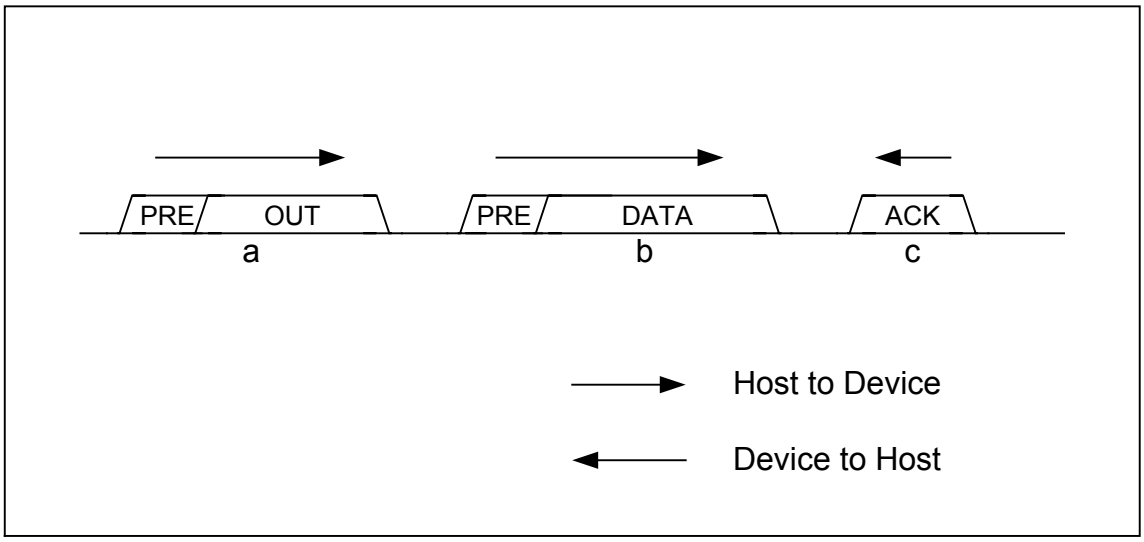



**Fig. 6.26 OUT transaction with preamble attached**

Fig. 6.27 shows how an interrupt IN transaction will be performed for the case where the transaction finishes without errors. In (a), the LSI issues an IN token addressed to the IN-direction endpoint that resides in the local node after attaching a preamble at the beginning of it. In (b), the device sends a data packet within Max. packet size. The LSI writes this data to the FIFO of the relevant channel. In (c), the LSI returns ACK for response after attaching a preamble at the beginning of it. It then automatically sets the relevant register and issues a status to the firmware.

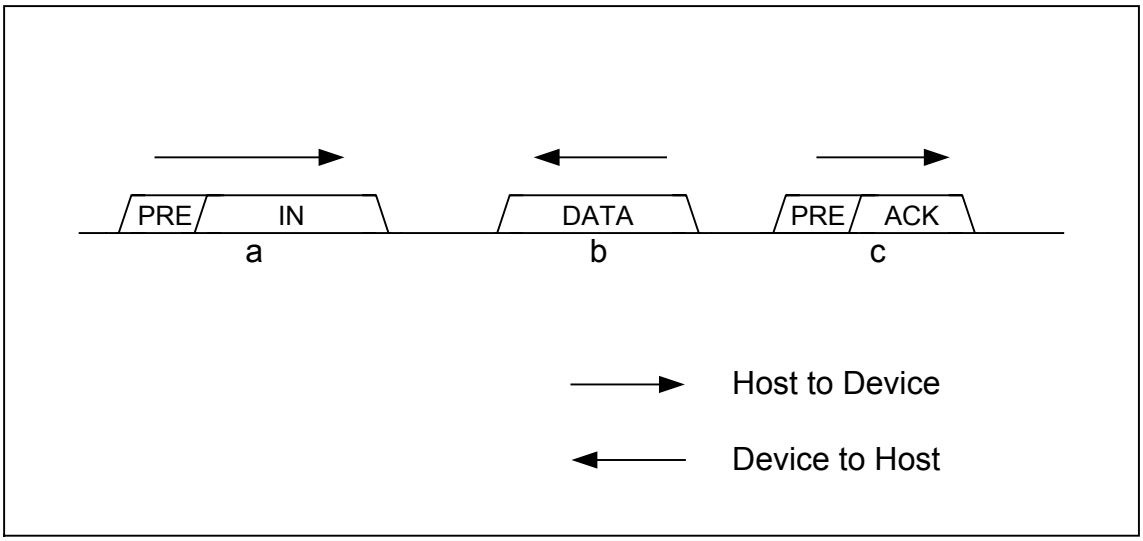



**Fig. 6.27 IN transaction with preamble attached**

#### 6.3.3.8 Split Transactions

If the downstream port has a high-speed (HS) hub connected and the downstream port of the hub has an FS or LS device connected, the host operates in HS mode. The host sets the transfer rate for the channel used (H_CHx{x=0,a-e}Config_0.SpeedMode) to FS or LS, and thereby executes a transaction for the corresponding endpoint in split transactions to or from the hub.

For the relevant channel, set the hub address (H_CHx{x=0,a-e}HubAdrs.HubAdrs) and port number (H_CHx{x=0,a-e}HubAdrs.Port) to appropriate values. The sequence of start split transaction through complete split transactions in split transactions is controlled by the hardware. The individual transactions in split transactions will not affect the firmware. If the last complete split transaction in the sequence of start split transaction through complete split transactions finishes without errors, the hardware issues an ACK status (H_CHx{x=0,a-e}IntStat.TranACK bit) to the firmware and updates the FIFO. On the other hand, for individual transactions other than the last complete split transaction, no status is issued to the firmware for their successful completion. If an error occurrs in any individual transaction in the sequence of start split transaction through complete split transactions, the hardware sets the condition code (H_CHx{x=0,a-e}ConditionCode) to RetryError and issues a TranErr status (H-CHx{x=0,a-e}IntStat.TranErr bit) to the firmware. In this case, the hardware does not update the FIFO, and performs a retry process. If the error continues three times consecutively, the hardware terminates the transfer by automatically clearing H_CHx{x=0,a-e}Control.TranGo and issues a ChangeCondition status (H-CHx{x=0,a-e}IntStat.ChangeCondition bit) to the firmware.

### 6.3.4 Control Transfer

Each stage of the transfer in a control transfer is controlled as an individual transaction.

Fig. 6.28 shows how a control transfer is controlled. The firmware sets each of the SETUP, DATA, and STATUS stages appropriately to ensure that the Control transfer is executed in hardware.

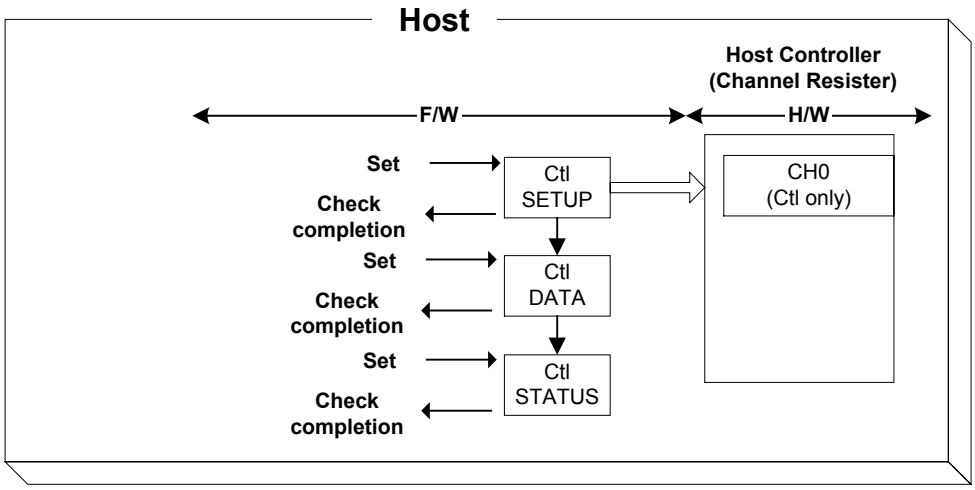



**Fig. 6.28 Controlling a control transfer**

Fig. 6.29 shows how a Control transfer is performed in host mode in cases in which the data stage is set in the OUT direction. In (a), the host starts a control transfer via a SETUP transaction. In (b), the host issues an OUT transaction to execute a data stage. In (c), the host issues an IN transaction to execute a status stage.

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

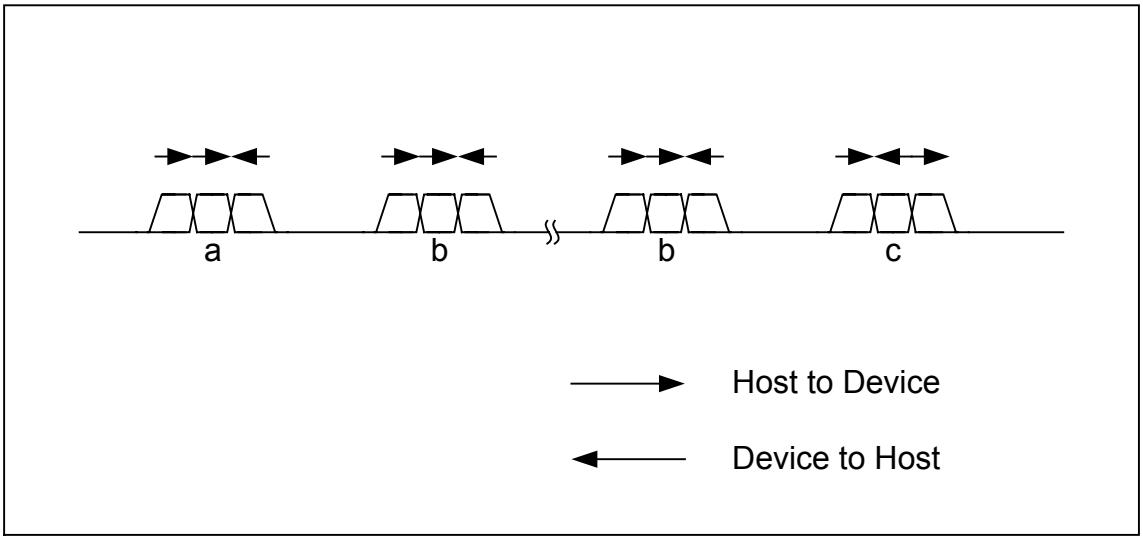



**Fig. 6.29 Control transfer in host mode when the data stage is set in the OUT direction**

Fig. 6.30 shows how a Control transfer is performed in host mode in cases in which the data stage is set in the IN direction. In (a), the host starts a control transfer via a SETUP transaction. In (b), the host issues an IN transaction to execute a data stage. In (c), the host issues an OUT transaction to execute a status stage.

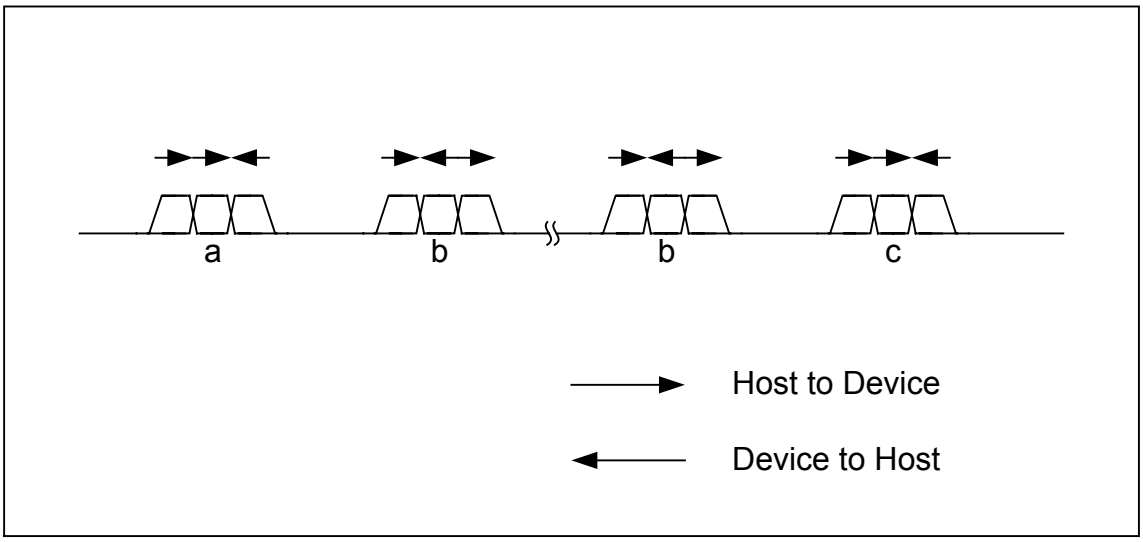



**Fig. 6.30 Control transfer in host mode when the data stage is set in the IN direction**

#### 6.3.4.1 Setup Stage

The setup stage is executed by a setup transaction. For details, refer to the relevant section in Chapter 6 on setup transactions.

#### 6.3.4.2 Data Stage and Status Stage

After the setup stage is complete, go to the next stage.

If the next stage is for an IN direction, set the transaction type (H_CH0Config_1.TID) to IN, and then set other basic setup registers as accordingly to execute a transaction.

On the other hand, if the next stage is for an OUT direction, set the transaction type (H_CH0Config_1.TID) to OUT, and then set other basic setup registers as accordingly to execute a transaction.

Note that if a status stage is that is executed, set the number of IRP data bytes (H_CH0TotalSize_H,L) to 0 before executing a transaction.

### 6.3.4.3 Control Transfer Support Function

The LSI stipulated herein has the function to automatically manage a series of stages executed in a control transfer. Use of this function relieves the firmware from the burden of managing each stage as an individual transaction. See Fig. 6.31.

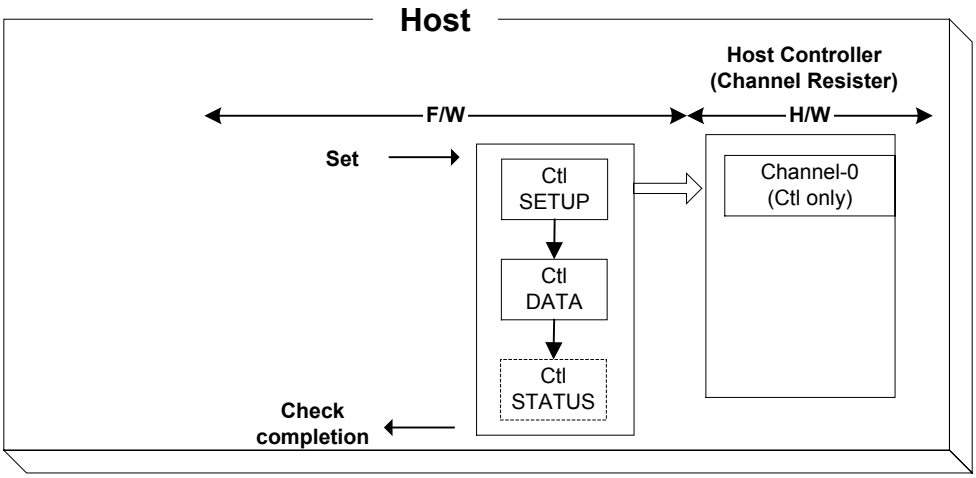



**Fig. 6.31 Control by the control transfer support function**

The control transfer support function is available only for channel CH0. When this function is used, a Control transfer is performed in the manner described in (1) to (10). The firmware performs processing in (1) to (4) and (7).

(1) Set the following basic setup registers for channel CH0 as accordingly.
Transfer rate (H_CH0Config_0.SpeedMode), max packet size (H_CH0MaxPktSize), USB address (H_CH0FuncAdrs.FuncAdrs), endpoint number (H_CH0FuncAdrs.EP_Number), FIFO area (AREA0StartAdrs-H,L, AREA0EndAdrs-H,L), FIFO area join (AREA0Join_1.JoinEP0CH0)

(2) Write setup data (8 bytes) to the setup registers (H_CH0SETUP_0 through 7).

(3) If the data stage is set in the OUT direction, write the transmit data to the FIFO joined to CH0.
If the data stage is set in the IN direction, clear the FIFO joined to CH0.

(4) Set the control transfer support execution (H_CTL_SupportControl.CTL_SupportGo).
At this point, set the control transfer stage (H_CTL_SupportControl.CTL_SupportState) to Idle (00b) by writing that value to the H_CTL_SupportControl register.

(5) A SETUP transaction (setup stage) is executed using the data (8 bytes) in the SETUP register.

(6) A data stage is executed based on the SETUP data.

- If bmRequestType in bit 7 of the SETUP data = 0, the data present in the channel CH0 area of the FIFO is transmitted by an OUT transaction (OUT-direction data stage).
- If bmRequestType in bit 7 of the SETUP data = 1, an IN transaction is issued and the received data is written to the channel CH0 area of the FIFO (IN-direction data stage).

- A data stage is executed by performing transactions for the number of data bytes indicated by wLength of the SETUP data.
- When a short packet is received while the data stage is set in the IN direction, issuance of the IN transaction is halted even when the received data is less than that specified by wLength in the SETUP data.
- A data stage in which wLength of the SETUP data = 0x0000 is not executed.

(7) If the FIFO area joined to CH0 is smaller than the value indicated by wLength in the SETUP data, the firmware should divide the data of the data stage as it processes the data.

- If while the data stage is set in the OUT direction the data to be sent to the FIFO area joined to CH0 runs out, no further transactions are issued. The firmware should check for FIFO free space as it writes the remaining transmit data to the FIFO area.
- If while the data stage is set in the IN direction the FIFO area joined to CH0 runs short of free space, no further transactions are issued. The firmware should check the valid bytes of data in the FIFO as it reads out the data received in sequence from the FIFO to create free space in the FIFO.

(8) A status stage is executed based on the SETUP data.

- If the status stage is directed for OUT, an IN transaction is issued (IN-direction status stage).
- If while the data stage is set in the IN direction the FIFO area joined to CH0 has all of the received data read out of it and is thereby emptied when no further transactions are issued for reasons that a quantity of data equal to wLength in the SETUP data or a short packet was received, an OUT transaction for a zero-length packet is issued.

(9) When the control transfer is completed without errors, the control transfer support execution (H_CTL_SupportControl.CTL_SupportGo) is automatically cleared and a control transfer complete status (H_CH0IntStat.CTL_SupportCmp) is issued.

(10) If a transaction error is detected in the middle of a control transfer, the control transfer support execution (H_CTL_SupportControl.CTL_SupportGo) is automatically cleared to abort the control transfer and a control transfer stopped status (H_CH0IntStat.CTL_SupportStop) is issued. The control transfer stage (H_CTL_SupportControl.CTL_SupportState) is flagged to indicate the stage in which the error occurred. In addition, the condition code (H_CH0ConditionCode) is set to a valid value and a ChangeCondition status (H_CH0IntStat.ChangeCondition bit) is issued.

To abort a control transfer, clear the control transfer support execution (H_CTL_SupportControl.CTL_SupportGo). A status will be issued when processing of abortion for the control transfer is complete. At this point, if the control transfer is completed upto the status stage by the time abortion processing is complete, a control transfer complete status (H_CH0IntStat.CTL_SupportCmp) is issued. If the control transfer is not completed by the time abortion processing is complete, a control transfer stopped status

(H_CH0IntStat.CTL_SupportStop) is issued. The stage in which the control transfer was aborted is indicated by the control transfer stage (H_CTL_SupportControl.CTL_SupportState). To resume a control transfer from the stage in which it was aborted, set the control transfer stage (H_CTL_SupportControl.CTL_SupportState) to a stage from which the operation is to be resumed (i.e., the aborted stage) and set the control transfer support execution (H_CTL_SupportControl.CTL_SupportGo).

On the other hand, to perform a new control transfer, set the control transfer stage (H_CTL_SupportControl.CTL_SupportState) to Idle (00b) and then set the control transfer support execution (H_CTL_SupportControl.CTL_SupportGo).

While the control transfer support function is being executed, the transfer execution bit (H_CH0Config_0.TranGo), toggle sequence bit (H_CH0Config_0.Toggle), transaction type (H_CH0Config_1.PID), and IRP data size (H_CH0TotalSize_H,L) are set and updated by the hardware. Therefore, do not write to these register bits during that time.

For details about transaction errors, refer to the relevant sections in Chapter 6 on individual transactions.

Table 6.26 lists the setup items and status of the control transfer support function.

**Table 6.26 Control Items and Status of the Control Transfer Support Function**

| Item | Register/Bit | Description |
|---|---|---|
| Control transfer support execution | H_CTL_SupportControl.CTL_SupportGo | Automatically executes control transfer stages.<br>For details, refer to the section in Chapter 6 on control transfer support function. |
| Control transfer stage | H_CTL_SupportControl.CTL_SupportState | Indicates the stage being executed by the control transfer support function. Or if a control transfer was aborted for an error, it indicates the stage in which the error occurred. |
| Control transfer execution result | H_CH0IntStat.CTL_SupportCmp<br>H_CH0IntStat.CTL_SupportStop | Indicates the result of a control transfer executed by the control transfer support function. |
| Transaction status | H_CH0IntStat.TotalSizeCmp,<br>H_CH0IntStat.TranACK,<br>H_CH0IntStat.TranErr,<br>H_CH0IntStat.ChangeCondition | Indicates the result of a transaction. |
| Transaction condition code | H_CH0ConditionCode.ConditionCode | Indicates details of transaction result. |

### 6.3.5 Bulk and Interrupt Transfers

Bulk transfers on CHa, as well as bulk and interrupt transfers on CHb, CHc, CHd, and CHe can be controlled either as a data flow (see 6.3.6) or as successive individual transactions (see 6.3.3).

### 6.3.6 Data Flow

This section describes control of a general data flow in OUT and IN transfers.

#### 6.3.6.1 OUT Transfer

Set the total number of OUT transfer data bytes in H_CH0TotalSize_H,L or CHx{x=a-e}TotalSize_HH,HL,LH,LL and write the data to be sent by an OUT transfer into the FIFO area joined to each corresponding channel. There are two methods for writing into the FIFO: a register write through the CPU interface or a DMA write through the CPU interface.

To write data into the FIFO by means of a register write through the CPU interface, select a single channel using the AREAn{n=0-5}Join_0.JoinCPU_Wr bit in the same area as the FIFO area joined to each corresponding channel. The FIFO of the selected channel can be written to by a FIFO_Wr register, and the data is transmitted in data packets in the order in which it was written. A free space in the FIFO can be inspected by FIFO_Remain_H,L registers. FIFOs in full state cannot be written to. Always be sure to inspect the FIFO_Remain_H,L registers to confirm the available size (in bytes) in the FIFO and make sure data is not written exceeding that size.

To write data into the FIFO by means of a DMA write through the CPU interface, select a single channel using the AREAn{n=0-5}Join_0.JoinDMA bit in the same area as the FIFO area joined to each corresponding channel. Set the DMA_Control.Dir bit to 0. Data is written to the FIFO for the selected channel a DMA procedure in the CPU interface. The data is transmitted in packets in the order written. When the FIFO is full, the CPU interface automatically pause the DMA for flow control.

The size of data packets transmitted by an OUT transaction is H_CH0TotalSize_H,L or smaller one of H_CHx{x=a-e}TotalSize_HH,HL,LH,LL and H_CHx{x=a-e}MaxPacketSize_H,L.

If data equal to or greater than the data size of data packet is present in the FIFO, an OUT transaction is executed to transmit the data. Also, H_CH0TotalSize_H,L or H_CHx{x=a-e}TotalSize_HH,HL,LH,LL is decremented by an amount equal to the transmitted data size. When TotalSize is reduced to 0, H_CHx{x=0,a-e}Config_0.TranGo is automatically cleared to terminate the transfer, and a TotalSizeCmp status (H_CHx{x=0,a-e}IntStat.TotalSizeCmp bit) is issued to the firmware.

That way, OUT transfers can be performed without the burden of controlling individual transactions by the firmware.

#### 6.3.6.2 IN Transfer

Set the total number of IN transfer data bytes in H_CH0TotalSize_H,L or CHx{x=a-e}TotalSize_HH,HL,LH,LL.

The expected data length of the data packet to be received in an IN transaction is H_CH0TotalSize_H,L or smaller one of H_CHx{x=a-e}TotalSize_HH,LH,LL and H_CHx{x=0,a-e}MaxPktSize_H,L. If the FIFO has a free space equal to or greater than MaxPacketSize, an IN transaction is executed to receive data. Also, the H_CH0TotalSize_H,L or H_CHx{x=a-e}TotalSize_HH,LH,LL is decremented by an amount equal to the received data size. When TotalSize is reduced to 0, H_CHx{x=0,a-e}Config_0.TranGo is automatically cleared to terminate the transfer, and a TotalSizeCmp status (H_CHx{x=0,a-e}IntStat.TotalSizeCmp bit) is issued to the firmware.

If the received data size is larger than the expected size of data packet, H_CHx{x=0,a-e}Config_0.TranGo is automatically cleared to terminate the transfer. No response is returned. The condition code (H_CHx{x=0,a-e}ConditionCode) is set to DataOverrun, and a ChangeCondition status (H-CHx{x=0,a-e}IntStat.ChangeCondition bit) is issued to the firmware. The FIFO is not updated.

If the received data size is less than the expected size of data packet, H_CHx{x=0,a-e}Config_0.TranGo is automatically cleared to terminate the transfer. The condition code (H_CHx{x=0,a-e}ConditionCode) is set to DataUnderrun, and a ChangeCondition status (H-CHx{x=0,a-e}IntStat.ChangeCondition bit) is issued to the firmware. In addition, the FIFO is updated, and assuming data has been received, an area is reserved.

That way, IN transfers can be performed without the burden of controlling individual transactions by the firmware.

The data received by an IN transfer is written to the FIFO area joined to each corresponding channel. There are two methods for reading out data from the FIFO: a register read through the CPU interface or a DMA read through the CPU interface.

To read out data from the FIFO by means of a register read through the CPU interface, select a single channel using the AREAn{n=0-5}Join_0.JoinCPU_Rd bit in the same area as the FIFO area joined to each corresponding channel. The FIFO of the selected channel can be read out in the order in which data was received by using the FIFO_Rd or FIFO_ByteRd register. The number of data bytes in the FIFO that can be read out is indicated by the FIFO_RrRemain_H,L registers. Since empty FIFOs cannot be read out, always be sure to inspect the FIFO_RrRemain_H,L registers to confirm the number of readable data bytes and make sure the FIFO is not accessed for read exceeding those bytes.

To read out data from the FIFO by means of a DMA read through the CPU interface, select a single channel using the AREAn{n=0-5}Join_0.JoinDMA bit in the same area as the FIFO area joined to each corresponding channel. Set the DMA_Control.Dir bit to 1. The FIFO for the selected channel will be read out in the order received by executing a DMA procedure in the CPU interface. The number of remaining data bytes in the FIFO can be known by the DMA_Remain_H,L registers. When the FIFO is emptied, the CPU interface automatically pause the DMA for flow control.

### 6.3.7 Zero-length Packet Auto Issue Function

A function to automatically issue a zero-length packet is enabled by setting the H_CHx{x=a-e}Config_1.AutoZeroLen bit on a channel where an OUT transfer is performed.

Table 6.27 lists the setup items of the zero-length packet automatic issue function.

**Table 6.27 Control Items of the Zero.length Packet Automatic Issue Function**

| Item | Register/Bit | Description |
|---|---|---|
| Zero-length packet automatic issue | H_CHx{x=a-e}Config_1.AutoZeroLen | Enables the zero-length packet automatic issue function. This bit is effective for only OUT transfers. |

#### 6.3.7.1 Zero-length Packet Auto Issue Function in Bulk/Interrupt OUT Transfers

On a channel where a Bulk/Interrupt OUT transfer is being executed, if the data size of the last transaction in a transfer for the data size set by the H_CHx{x=a-e}TotalSize_HH,HL,LH,LL register equals the max packet size, H_CHx{x=a-e}Config_0.TranGo is not automatically cleared. Then, when the channel is scheduled again, an OUT transaction is executed with zero-length packet. When this transaction is completed without errors, H_CHx{x=a-e}Config_0.TranGo is automatically cleared to terminate the transfer, and a TotalSizeCmp status (H_CHx{x=a-e}IntStat.TotalSizeCmp bit) is issued to the firmware.

### 6.3.8 Bulk Only Support Function

The LSI stipulated herein has a function to automatically manage a series of transport operations for command transport (CBW), data transport, and status transport (CSW) of the USB Mass Storage Class (BulkOnly Transport Protocol). Use of this function relieves the firmware of the burden of controlling each transport individually. Fig. 6.32 and Fig. 6.33 below show an example where transports are controlled by the Bulk-only Support function and an example where each transport is controlled as an individual transaction without using this function.

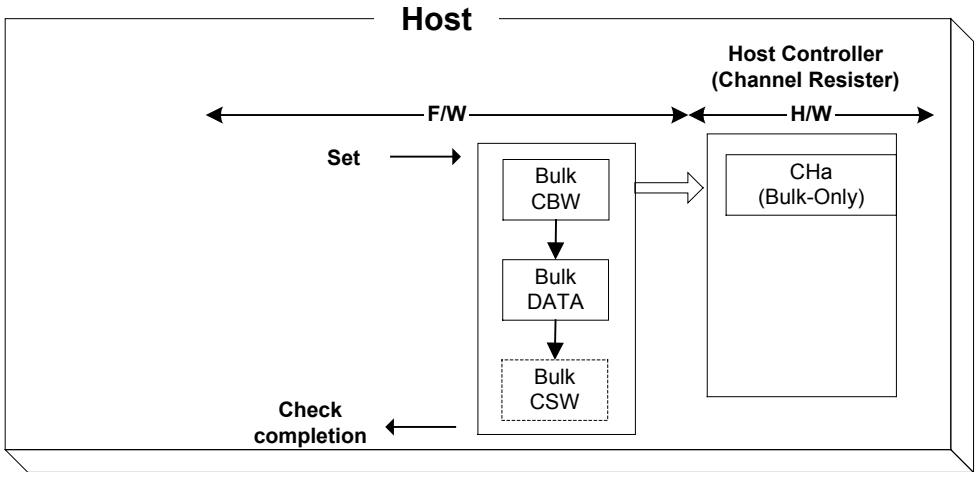



**Fig. 6.32 Control of transports by the bulk.only support function**

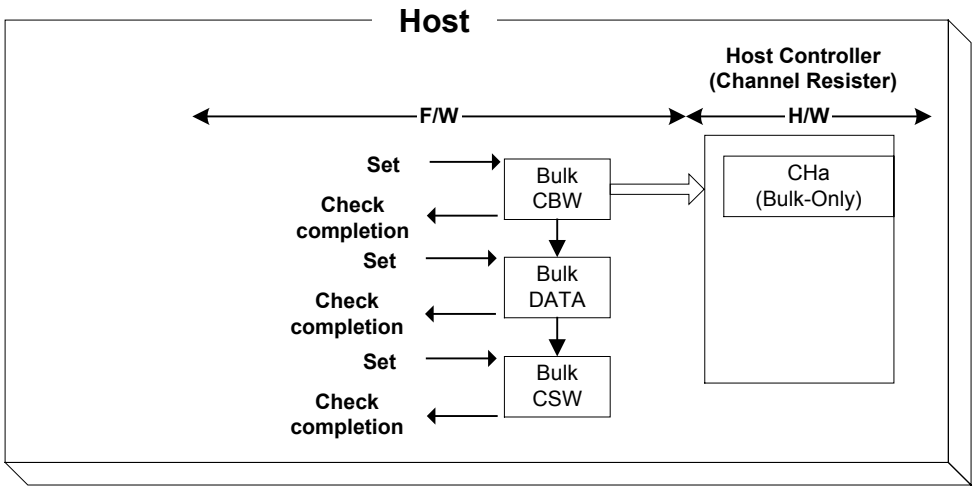



**Fig. 6.33 Control of transports without using the bulk.only support function (reference)**

The bulk-only support function is available only for channel CHa. This function processes transport in the manner described in (1) to (11). The firmware performs processing in (1) to (5).

(1) Set the following basic setup registers for channel CHa as accordingly.
Transfer rate (H_CHaConfig_0.SpeedMode), max packet size (H_CHaMaxPktSize), USB address (H_CHaFuncAdrs.FuncAdrs), FIFO area (AREA1StartAdrs_H,L, AREA1EndAdrs_H,L), FIFO area join (AREA1Join_1.JoinEPaCHa)

(2) Set the following control registers for the bulk-only support function as accordingly.
OUT endpoint toggle sequence (H_OUT_EP_Control.OUT_Toggle), OUT endpoint number (H_OUT_EP_Control.OUT_EP_Number), IN endpoint toggle sequence (H_IN_EP_Control.IN_Toggle), IN endpoint number (H_IN_EP_Control.IN_EP_Number)

(3) Write CBW data (31 bytes) to the CBW area of the FIFO.

(4) Set DMA for the FIFO area joined to CHa.

(5) Set bulk-only support execution (H_BO_SupportControl.BO_SupportGo).
At this point, set the transport state (H_BO_SupportControl.BO_TransportState) to Idle (00b) by writing that value to the H_BO_SupportControl register.

(6) The data (31 bytes) present in the CBW area is transmitted to the OUT-direction endpoint indicated by the OUT endpoint number (H_OUT_EP_Control.OUT_EP_Number) via a bulk OUT transaction (command transport).

(7) A data transport is executed based on the CBW data.

- If bmCBWFlags in bit 7 of the CBW data = 0, the data present in the FIFO area joined to CHa is transmitted to the OUT-direction endpoint indicated by the OUT endpoint number (H_CHaBO_OUT_EP_Ctl.OUT_EP_Number) by means of a Bulk OUT transaction (OUT-direction data transport).
- If bmCBWFlags in bit 7 of the CBW data = 1, a Bulk IN transaction is issued to the IN-direction endpoint indicated by the IN endpoint number

(H_CHaBO_IN_EP_Ctl.IN_EP_Number) to write the received data to the FIFO area joined to CHa (IN-direction data transport).

- A data transport is executed by performing transactions for the number of data bytes indicated by dCBWDataTransferLength of the CBW data.
- When a short packet is received while the data transport is set in the IN direction, issuance of the IN transaction is halted even when the data size of the received data is less than that specified by dCBWDataTransferLength in the CBW data.
- A data transport in which dCBWDataTransferLength of the CBW data = 0x00000000 is not executed.

(8) A bulk IN transaction is issued to the IN-direction endpoint indicated by the IN endpoint number (H_IN_EP_Control.IN_EP_Number) and the received data is written to the CSW area of the FIFO (status transport). The number of data bytes received in a status transport is reflected in the status transport received data size (H_CSW_RcvDataSize.CSW_RcvDataSize).

- If the data transport is set in the OUT direction, the LSI is placed in a state in which it can execute a status transport immediately after data transport is complete.
- If the data transport is set in the IN direction, the LSI is placed in a state in which it can execute a status transport once all received data has been read out of the FIFO area joined to CHa and is thereby emptied, while no further transactions are issued for reasons that a quantity of data equal to dCBWDataTransferLength in the CBW data or a short packet was received.

(9) The CSW received in a status transport is checked. This check is designed to confirm the following:

- The received data length of CSW is 13 bytes.
- dCSWSignature of CSW is 0x53425355.
- dCSWTag of CSW matches dCBWTag of CBW.
- The value of BCSWStatus is 0x00.

If any one of the above is not true, the bulk-only support execution (H_BO_SupportControl.BO_SupportGo) is automatically cleared to turn the bulk-only support function off. In addition a bulk-only support stopped status (H_CHaIntStat.BO_SupportStop) is issued. The data received in the CSW area can be read out using the RAM_Monitor function.

(10) When the status transport is completed normally, Bulk-only Support execution (H_CHaBO_SupportCtl.BO_SupportGo) is automatically cleared and a Bulk-only Support completed status indication (H_CHaIntStat.BO_SupportCmp) is issued.

(11) When a transaction error is detected during any of the transport processes, the bulk-only support execution (H_BO_SupportControl.BO_SupportGo) is automatically cleared to turn the bulk-only support function off, and a bulk-only support stopped status (H_CHaIntStat.BO_SupportStop) is issued. Then the transport state (H_BO_SupportControl.BO_TransportState) is flagged to indicate the transport in which the error occurred. In addition, the condition code (H_CHaConditionCode) is set to a valid value, and a ChangeCondition status (H-CHaIntStat.ChangeCondition bit) is issued.

To abort the bulk-only support function, clear the bulk-only support execution (H_BO_SupportControl.BO_SupportGo). A status is issued when abortion processing for the bulk-only support function is complete. At this point, if the transport had been completed way up to a status transport by the time abortion processing has finished, a bulk-only support complete status (H_CHaIntStat.BO_SupportCmp) is issued. At this point, if the transport is not completed upto the status transport by the time abortion processing is complete, a bulk-only support status (H_CHaIntStat.BO_SupportStop) is issued. The aborted transport is indicated by the transport state (H_BO_SupportControl.BO_TransportState). To resume a bulk-only support function from the aborted transport, set the transport state (H_BO_SupportControl.BO_TransportState) to a transport from which the operation is to be resumed (i.e., the aborted transport), and set the bulk-only support execution (H_BO_SupportControl.BO_SupportGo).

On the other hand, to execute the bulk-only support function anew, set the transport state (H_BO_SupportControl.BO_TransportState) to Idle (00b) and set the bulk-only support execution (H_BO_SupportControl.BO_SupportGo).

While the bulk-only support execution is being executed, the transfer execution bit (H_CHaConfig_0.TranGo), toggle sequence bit (H_CHaConfig_0.Toggle), transaction type (H_CHaConfig_1.TID), total size free bit (H_CHaConfig_1.TotalSizeFree), endpoint number (H_CHaFuncAdrs.EP_Number), and IRP data size (H_CHaTotalSize_HH,HL,LH,LL) are set and updated by the hardware. Therefore, do not write to these register bits during that time.

For details about transaction errors, refer to the relevant sections in Chapter 6 on individual transactions.

For details about the CBW and CSW areas of the FIFO, refer to the relevant section in Chapter 6 on FIFOs.

For details about the DMA, refer to the relevant section in Chapter 6 on DMAs.

Table 6.28 lists the setup items and status of the bulk-only support function.

**Table 6.28 Setup Items and Status of the Bulk Only Support Function**

| Item | Register/Bit | Description |
|---|---|---|
| Bulk Only support execution | H_BO_SupportControl.BO_SupportGo | Executes the bulk-only support function. For details, refer to the relevant section in Chapter 6 on bulk-only support function. |
| OUT endpoint toggle sequence | H_OUT_EP_Control.OUT_Toggle | Sets the initial value of the OUT endpoint toggle sequence bit. When a transaction is being executed or a transaction is completed, it indicates the state of the OUT endpoint toggle sequence bit. |
| OUT endpoint number | H_OUT_EP_Control.OUT_EP_Number | Sets the endpoint number of the OUT endpoint to any value between 0x0 and 0xF. |
| IN endpoint toggle sequence | H_IN_EP_Control.IN_Toggle | Sets the initial value of the IN endpoint toggle sequence bit. When a transaction is being executed or a transaction is completed, it indicates the state of the IN endpoint toggle sequence bit. |
| IN endpoint number | H_IN_EP_Control.IN_EP_Number | Sets the endpoint number of the IN endpoint to any value between 0x0 and 0xF. |
| Result of bulk-only support execution | H_CHaIntStat.BO_SupportCmp<br>H_CHaIntStat.BO_SupportStop | Indicates the result of the bulk-only support executed. |
| Transaction status | H_CHaIntStat.TotalSizeCmp,<br>H_CHaIntStat.TranACK,<br>H_ CHaIntStat.TranErr,<br>H_CHaIntStat.ChangeCondition | Indicates the result of a transaction. |
| Transaction condition code | H_CHaConditionCode | Indicates details of transaction result. |
| Transport status | H_BO_SupportControl.<br>BO_TransportState | Indicates the transport being executed under control of the bulk-only support function. If any transport was aborted for an error, it indicates the transport in which the error occurred. |
| Status transport received data size | H_CSW_RcvDataSize | Indicates the number of data bytes received in a status transport. |

### 6.3.9 Host State Management Support Function

#### 6.3.9.1 Host States

The host must have its state changed according to a request from a high-order system and the bus state. Therefore, the host states are managed by the firmware. The hardware supports various settings and negotiations in each state.

Fig. 6.34 shows a transition of the host states. Table 6.29 lists the setup items and statuses of the host state management support function.

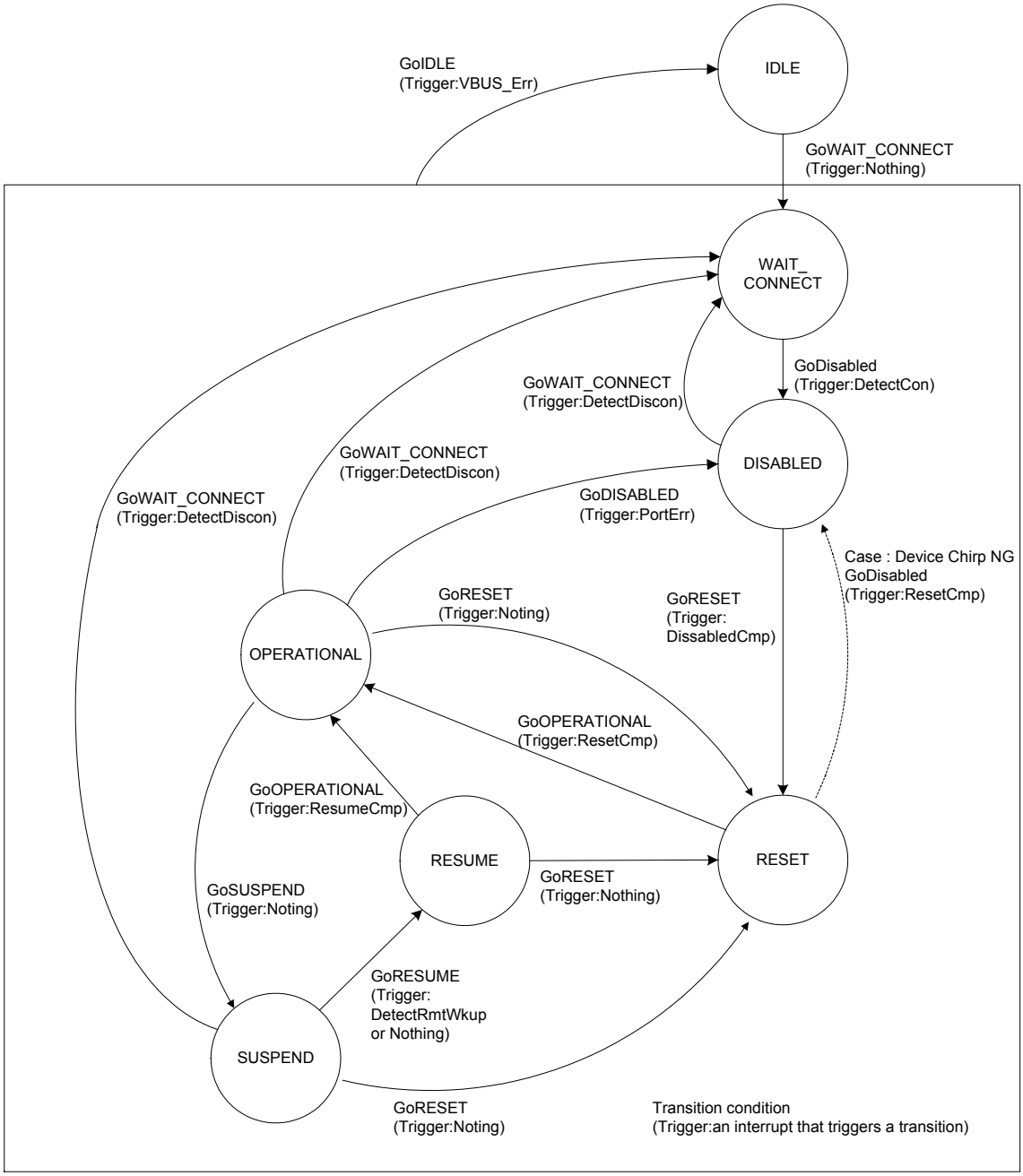



**Fig. 6.34 Host state transition diagram**

**Table 6.29 Setup Items and Status of the Host State Management Support Function**

| Item | Register/Bit | Description |
|---|---|---|
| Host state transition execution | H_NegoControl_0. AutoMode | Sets the host state to which to go. The state that is set here is one of the following:<br>GoIDLE<br>GoWAIT_CONNECT<br>GoDISABLED<br>GoRESET<br>GoOPERATIONAL<br>GoSUSPEND<br>GoRESUME<br>GoWAIT_CONNECTtoDIS<br>GoWAIT_CONNECTtoOP<br>GoRESETtoOP<br>GoRESUMEtoOP<br>GoSUSPENDtoOP |
| Host state transition execution cancel | H_NegoControl_0. AutoModeCancel | Stops processing of the current host state and ceases processing at that state. |
| Host state monitor | H_NegoControl_0. HostState | Indicates the current host state (shown below).<br>IDLE<br>WAIT_CONNECT<br>DISABLED<br>RESET<br>OPERATIONAL<br>SUSPEND<br>RESUME |
| VBUS state monitor | H_USB_Status. VBUS_State | Indicates the VBUS state (normal/erratic). |
| Remote wakeup acceptance enable | H_NegoControl_1. RmtWkupDetEnb | Enables remote wakeup acceptance. |
| Chirp complete disable | H_NegoControl_1.DisChirpFinish | Sets operation mode to be assumed when device chirp is not completed within a specified time. |
| VBUS error detection status | H_SIE_IntStat_0. VBUS_Err | Indicates that an error occurred in VBUS. |
| Connection detection status | H_SIE_IntStat_0. DetectCon | Indicates that a device has been connected to the downstream port. |
| Disconnection detection status | H_SIE_IntStat_0. DetectDisCon | Indicates that a device has been disconnected from the downstream port. |
| Remote wakeup detection status | H_SIE_IntStat_0. DetectRmtWkup | Indicates that a remote wakeup signal from the device has been detected. |
| Device chirp complete detection status | H_SIE_IntStat_0. DetectDevChirpOK | Indicates that a chirp signal from the device is normal. |
| Device chirp error detection status | H_SIE_IntStat_0. DetectDevChirpNG | Indicates that a chirp signal from the device is abnormal. |
| Reset complete status | H_SIE_IntStat_1. ResetCmp | Indicates that USB reset is completed without errors. |
| Suspend transition complete status | H_SIE_IntStat_1. SuspendCmp | Indicates that a transition to Suspend is completed without errors. |
| Resume completion status | H_SIE_IntStat_1. ResumeCmp | Indicates that Resume is completed without errors. |
| Port speed | H_NegoControl_1. PortSpeed | Indicates the operating speed (HS, FS or LS) of the downstream port, i.e., operation of the device connected to the port. |
| Line state | H_USB_Status. LineState | Indicates the signal state of USB cable. |
| Transceiver selection | H_XcvrControl. XcvrSelect | Selects and enables one of HS, FS or LS transceivers. |
| Terminal selection | H_XcvrControl. TermSelect | Selects and enables one of HS, FS or LS terminals. |
| Operation mode | H_XcvrControl. OpMode | Sets operation mode of the HTM. |

#### 6.3.9.1.1 IDLE

This is the state where the USB host function is initialized, and is the default state that is assumed when the USB host function is enabled.

The host must be moved to this state when VBUS_Err is detected in any other state.

To execute a transition, write 0x80 to the H_NegoControl_0 register (H_NegoControl_0.AutoModeCancel = 1 and H_NegoControl_0.AutoMode = 0x0) to cause operation of the state under execution to stop. The H_NegoControl_0.AutoModeCancel bit is cleared to 0 by a completion of the stop processing (6 cycles required when operating with 60 MHz clock). After confirming that the H_NegoControl_0.AutoModeCancel bit has been cleared to 0, write 0x01 to the same register (which sets the host state transition execution (H_NegoControl_0.AutoMode) to GoIDLE). The LSI thereby enters this state.

In this state, the following settings are automatically executed:

- Transaction execution function of the USB host put to immediate stop
- Port set to FS mode and to NonDriving
- VBUSEN turned off
- All of the connection detection, disconnection detection, remote wakeup detection, and device chirp detection functions turned off

#### 6.3.9.1.2 WAIT_CONNECT

This is the state where the host waits until a device is connected to the downstream port.

When a device disconnection is detected in the OPERATIONAL or RESET state due to a request from a high-order system, the host must be temporarily placed in this state and await connection by the other party.

The LSI shifts to this state when the host state transition execution (H_NegoControl_0.AutoMode) is set to GoWAIT_CONNECT.

In this state, the following initial settings are automatically executed:

- Transaction execution function of the USB host put to immediate stop
- Port set to FS mode and to PowerDown
- VBUSEN turned on
- All of the connection detection, disconnection detection, remote wakeup detection, and device chirp detection functions turned on

Next, after internal power supply stabilization time for the bus power device has elapsed, the hardware automatically turns the connection detection function on, and waits until a device is connected. Note that the duration of time from when VBUSEN turned on to when a device connection is detected is not controlled by the hardware. This time should be managed by the firmware as necessary If FS or HS device is connected, it can be referenced as line state “J.”

If an LS device is connected, it can be referenced as line state “K.” When either state continues for 2.5 µs or more, a device connection is detected. If the connected device is an LS device, the port is set to LS mode. When a connection is detected, the disconnection detection function is automatically turned on.

Thereafter, if a disconnection is not detected during the debounce interval period, a connection detected status indication (H_SIE_IntStat0.DetectCon) is issued to the firmware, and the connection detection function and disconnection detection function are automatically turned off. If a disconnection is detected, the disconnection detection function is automatically turned off, and the process restarts from the connection detection phase.

#### 6.3.9.1.3 DISABLED

This is the state where the downstream port has a device connected, with no signals sent or received on the bus.

The LSI shifts to this state when a connection is detected in the WAIT_CONNECT state, Chirp from an erratic device is detected in the RESET state, or a port error is detected in the OPERATIONAL state.

The LSI shifts to this state when the host state transition execution (H_NegoControl_0.AutoMode) is set to GoDISALBED.

In this state, the following initial settings are automatically executed.

- Transaction execution function of the USB host put to immediate stop
- Port set to FS mode (in HS mode before the LSI shifted to this state) or retain current mode (in FS or LS mode)
- Port set to PowerDown

Next, the following processes are automatically executed after the disconnection detection disabled period expires:

- Disconnection detection function is turned on
- Disabled transition complete status (H_SIE_IntStat_1.DisableCmp) is issued

#### 6.3.9.1.4 RESET

This is the state where a USB reset is issued to the downstream port.

When a disabled transition complete status is issued in the DISABLED state, the LSI shifts to this state and then issues a USB reset.

In addition, if a request is made from a high-order system, the LSI can shift to this state from whichever USB mode (OPERATIONAL, SUSPEND, or RESUME).

The LSI shifts to this state when the host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESET.

In this state, the following initial settings are automatically executed.

- The transaction execution function of the USB host is turned off after the current transaction is completed.
- Port set to HS mode and to NormalOperation (reset signal SE0 is driven onto the USB cable)
- Connection detection, disconnection detection, and remote wakeup detection functions turned on
- Device chirp detection function turned on

Chirp from a device is detected as "HS K" from the downstream port. It is detected by a line state "K" that continues for 2.5 µs or more, and when the line state "K" is terminated within a specified time after a USB reset is issued, it is detected as normal Chirp. If the state is not terminated within a specified time, it is detected as erratic Chirp.

The following processes are automatically executed according to the detection result.

(1) When Chirp from a normal device is detected:
Pursuant to completion of Chirp from the device, "HS K" (Chirp K) and "HS J" (Chirp J) are alternately set out in succession from the host. When the host completes sending out Chirp, a reset complete status (H_SIE_IntStat_1.ResetCmp) is issued to the firmware. The port remains in HS mode.

(2) When Chirp from an erratic device is detected:
After certain specified time elapses a device chirp error detected status (H_SIE_IntStat_0.DetectDevChirpNG) is issued to the firmware. Ensuing operation can be selected from two operation modes by setting the chirp complete disable (H_NegoControl_1.DisChirpFinish). For details, refer to Section 6.3.9.3.4.2, "Detection of Chirp from an Erratic Device."

(3) When Chirp from a device is not detected and the other connected party is FS:
The port is set to FS mode after issuing a USB reset of a designated duration.
A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued to the firmware.

(4) When the other connected party is LS:
The port is set to LS mode after issuing a USB reset of a designated duration.
A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued to the firmware.

#### 6.3.9.1.5 OPERATIONAL

This is the state where a USB transaction is executed.

The LSI shifts to this state after completion of RESET or RESUME and then executes a transaction.

The LSI shifts to this state when the host state transition execution (H_NegoControl_0.AutoMode) is set to GoOPERATIONAL.

In this state, the following settings are automatically executed:

- Port set to NormalOperation
- Transaction execution function of the USB host enabled
- Disconnection detection function turned on

#### 6.3.9.1.6 SUSPEND

This is the state where USB is suspended.

The LSI shifts to this state from OPERATIONAL when it is necessary to stop the use of the USB bus.

The LSI shifts to this state when the host state transition execution (H_NegoControl_0.AutoMode) is set to GoSUSPEND.

In this state, the following initial settings are automatically executed:

- Disconnection detection and remote wakeup detection functions turned off
- Transaction execution function of the USB host disabled after waiting for completion of the current transaction
- Port set to FS mode (in HS mode), or retain current mode (in FS or LS mode)
- Port set to PowerDown.

Next, the following processes are automatically executed after the disconnection and remote wakeup detection disable period has expired.

- Disconnection detection function turned on.
- Remote wakeup detection turned on if remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb) is enabled.
- Suspend transition complete status (H_SIE_IntStat_1.SuspendCmp) issued

In addition, if while the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb) is enabled, a remote wakeup signal (consecutive “K” for 2.5 μs or more) is detected, a remote wakeup detected status (H_SIE_IntStat_0.DetectRmtWkup) is issued to the firmware.

#### 6.3.9.1.7 RESUME

This is the state where a USB resume signal is issued to the downstream port.

The LSI shifts to this state from SUSPEND in order to restore the USB device from a suspend state.

The LSI shifts to this state when the host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESUME.

In this state, the disconnection and remote wakeup detection functions are automatically turned off, and a resume signal (K) of a designated duration is issued. When issuance of the resume signal is complete, the port is set back to its previous mode before shifting to SUSPEND, and is set to NormalOperation. In addition, a suspend complete status (H_SIE_IntStat_0.ResumeCmp) is issued to the firmware.

### 6.3.9.2 Detection Functions

#### 6.3.9.2.1 VBUS Error Detection

A VBUS error is detected by a level change (from high to low) at the VBUSFLG input pin. The procedure that is executed when a VBUS error is detected is described below. Procedure step (2) below is automatically executed by the LSI's hardware.

(1) Pull the VBUSFLG (error flag for external USB power switch) input pin low (error occurred) (T0).

(2) Issue a VBUS error detected status (HostIntStat.VBUS_Err) to the firmware (T0).

Note that when the host detects a VBUS error, the VBUS must be turned off immediately. Therefore, the firmware should write 0x80 to the H_NegoControl_0 register (H_NegoControl_0.AutoModeCancel = 1 and H_NegoControl_0.AutoMode = 0x0) to cause operation of the state under execution to stop. The H_NegoControl_0.AutoModeCancel bit is cleared to 0 by a completion of the stop process (6 cycles required when operating with 60 MHz clock). After confirming that the H_NegoControl_0.AutoModeCancel bit is cleared to 0, write 0x01 to the same register (which sets the host state transition execution (H_NegoControl_0.AutoMode) to GoIDLE). This will cause the host to shift to IDLE state and the VBUSEN pin logic to be disabled, with the result of VBUS being turned off.

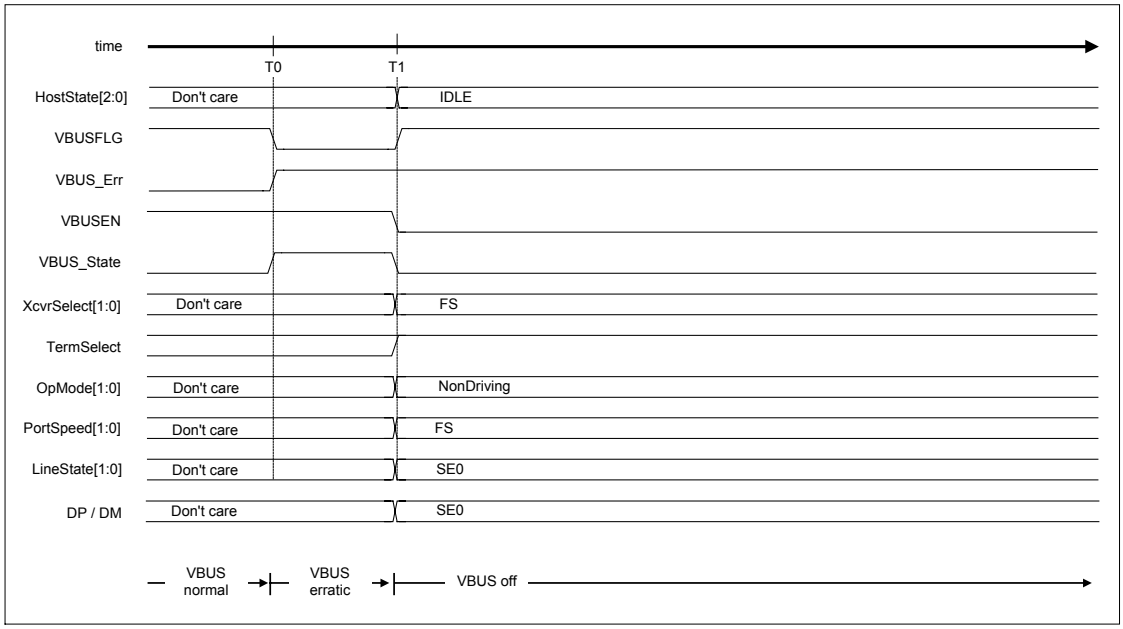



**Fig. 6.35 VBUS error detection timing**

**Table 6.30 VBUS Error Detection Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | The VBUSFLG (error flag for external USB power switch) input pin goes low (error occurred).<br>A VBUS error detected status (HostIntStat.VBUS_Err) is issued (by hardware). | 0 (reference) |
| T1 (reference) | The transition to the IDLE state is executed by first writing 0x08, then writing 0x01 to H_NegoControl_0 (by firmware). | T1 |

#### 6.3.9.2.2 Disconnection Detection

Disconnection of a device is detected in DISABLED, OPERATIONAL, and SUSPEND states.

To restart from the connection detection phase without turning VBUS off when a disconnection is detected, you must change the host state to WAIT_CONNECT. On the other hand, if the VBUS needs to be turned off, then change the host state to IDLE.

##### 6.3.9.2.2.1 When HS Device is Disconnected

Disconnection of an HS device is detected in the OPERATIONAL state.

The procedure that is executed when an HS device is disconnected is described below. Procedures (2) to (3) are automatically executed by the LSI's hardware.

(1) A device is disconnected (T0).

(2) A disconnection is detected in the EOP period of uSOF(HS_SOF). A disconnected state is assumed when a disconnection is detected three times in succession (T1).

(3) A disconnection detected status (H_SIE_IntStat_0.DetectDiscon) is issued to the firmware (T1).

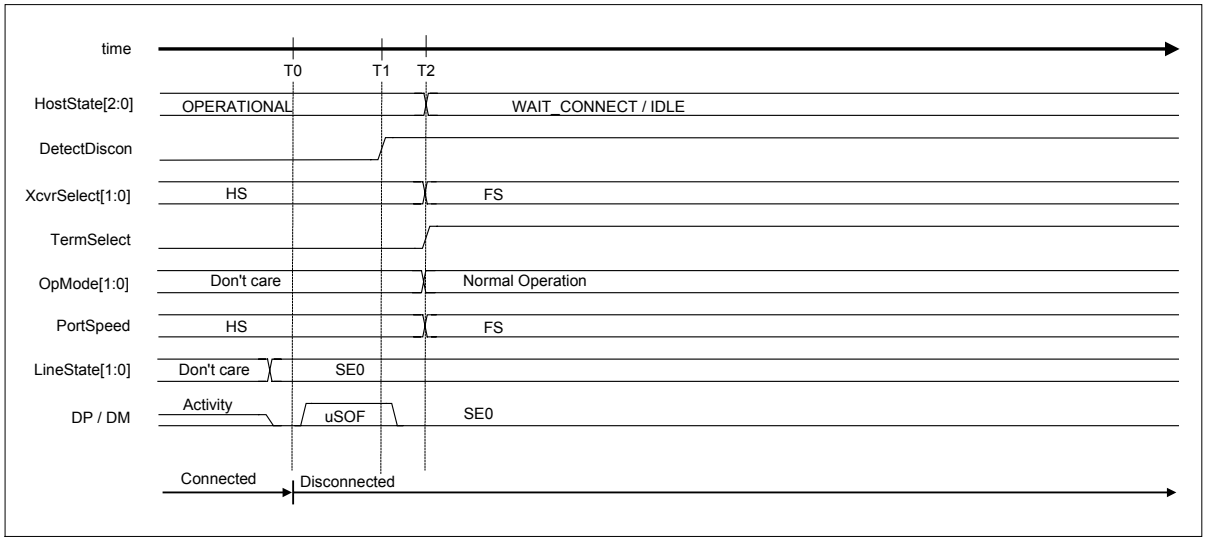



**Fig. 6.36 Disconnection detection timing (HS mode)**

**Table 6.31 Disconnection Detection Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | A device is disconnected. | 0 (reference) |
| T1 | A disconnection detected status (H_SIE_IntStat_0.DetectDiscon) is issued (by hardware). | T1 |
| T2 (reference) | Host state transition execution (H_NegoControl_0.AutoMode) is set to GoWAIT_CONNECT (by firmware). | T2 |

#### 6.3.9.2.2.2 When FS or LS Device is Disconnected

Disconnection of an FS or LS device is detected in the states DISABLED, OPERATIONAL, and SUSPEND.

The procedure that is executed when an FS or LS device is disconnected is described below. Procedures (2) to (3) are automatically executed by the LSI's hardware.

(1) A device is disconnected (T0).

(2) A disconnection is detected from the signal line state.

(3) A disconnection detected status (H_SIE_IntStat_0.DetectDiscon) is issued to the firmware (T1).

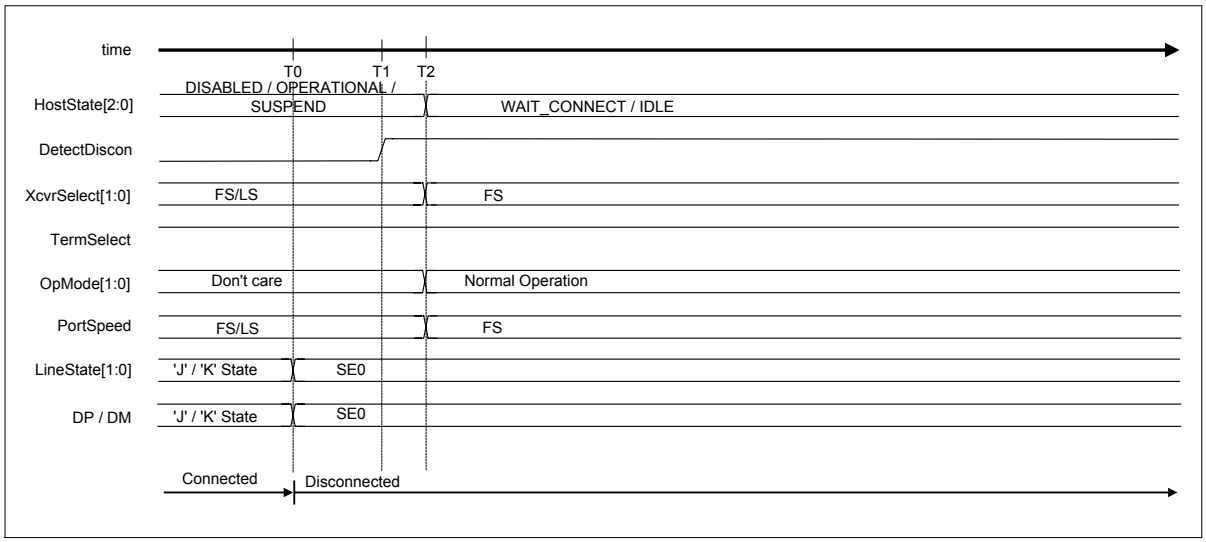



**Fig. 6.37 Disconnection detection timing (FS or LS mode)**

**Table 6.32 Disconnection Detection Timing Values (FS or LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | A device is disconnected. | 0 (reference) |
| T1 | A disconnection detected status (H_SIE_IntStat_0.DetectDiscon) is issued (by hardware). | T0 + 2.5us< T1 {$T_{DDIS}$} |
| T2 (reference) | Host state transition execution (H_NegoControl_0.AutoMode) is set to GoWAIT_CONNECT (by firmware). | Not stipulated |

#### 6.3.9.2.3 Remote Wakeup Detection

If the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb) is enabled, remote wakeup detection is performed in the SUSPEND state.

##### 6.3.9.2.3.1 When HS Device is Connected

The procedure that is executed for the case where an HS device is connected is described below. Procedures (2) to (3) are automatically executed by the LSI's hardware.

(1) The device starts sending out a remote wakeup signal (K) (T0).

(2) The host detects the remote wakeup signal (K) (T1).

(3) A remote wakeup detected status (H_SIE_IntStat_0.DetectRmtWkup) is issued to the firmware (T1).

Note that the host must issue a resume signal (K) within 1 ms after detecting a remote wakeup from the device. Therefore, the firmware should immediately set the host state transition execution (H_NegoControl_0.AutoMode) to GoRESUME within 900 µs after recognizing the remote wakeup detected status.

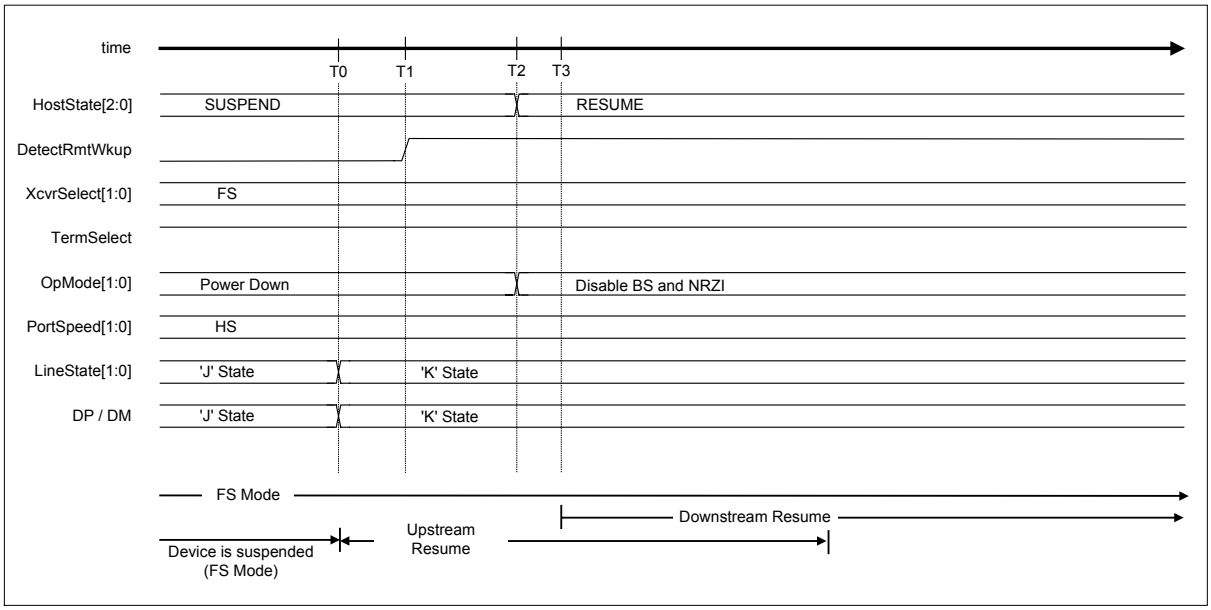



**Fig. 6.38 Remote wakeup timing (HS mode)**

**Table 6.33 Remote Wakeup Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | The device starts sending out a remote wakeup signal (K). | 0 (reference) |
| T1 | The remote wakeup signal (K) is detected.<br>A remote wakeup detected status is issued (by hardware). | T0 + 2.5us{TURLK} < T1 |
| T2<br>(reference) | Host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESUME<br>(by firmware). | T2< T1 + 900us |
| T3<br>(reference) | The host starts issuing a resume signal (K) (in hardware). | T3< T0 + 1ms{TURSM} |

Note: Names stipulated in the USB2.0 Standard are shown in { } .

#### 6.3.9.2.3.2 When FS Device is Connected

The procedure to be executed when an FS device is connected is the same as for the case when HS device is connected.

For details, refer to the preceding section.

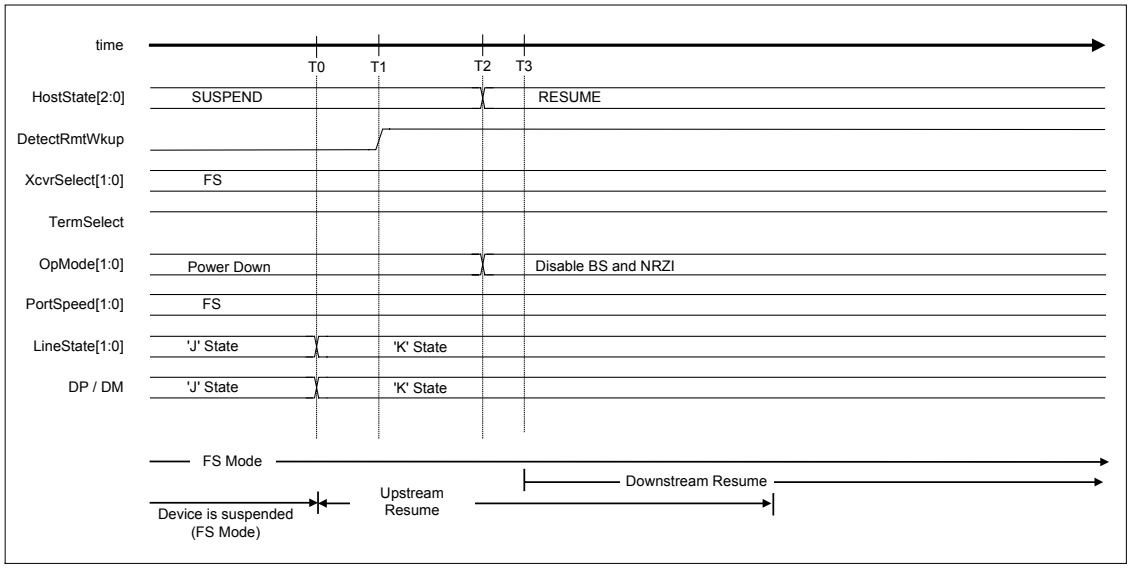



**Fig. 6.39 Remote wakeup timing (FS mode)**

**Table 6.34 Remote Wakeup Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | The device starts sending out a remote wakeup signal (K). | 0 (reference) |
| T1 | The remote wakeup signal (K) is detected.<br>A remote wakeup detected status is issued (by hardware). | T0 + 2.5us< T1 {TURLK} |
| T2 (reference) | Host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESUME (by firmware). | T2< T1 + 900us |
| T3 (reference) | The host starts issuing a resume signal (K) (in hardware). | T3< T0 + 1ms{TURSM} |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.2.3.3 When LS Device is Connected

The procedure to be executed when an LS device is connected is the same as for the case when an HS device is connected.

For details, refer to the preceding section.

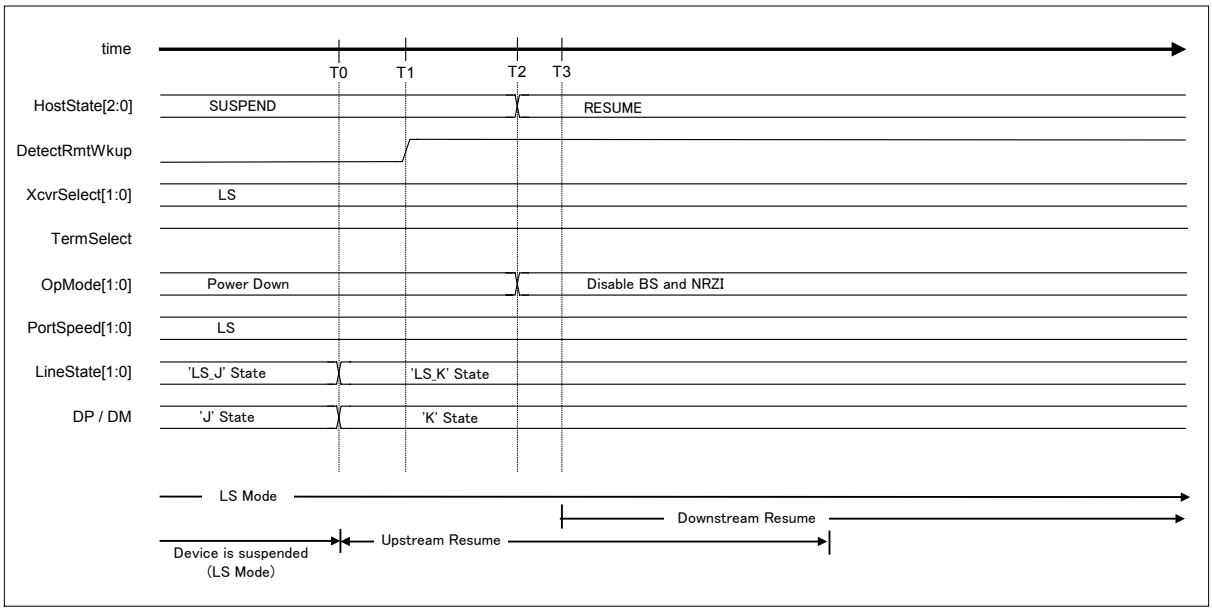



**Fig. 6.40 Remote wakeup timing (LS mode)**

**Table 6.35 Remote Wakeup Timing Values (LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | The device starts sending out a remote wakeup signal (K). | 0 (reference) |
| T1 | The remote wakeup signal (K) is detected.<br>A remote wakeup detected status is issued (by hardware). | T0 + 2.5us< T1 {TURLK} |
| T2<br>(reference) | Host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESUME (by firmware). | T2< T1 + 900us |
| T3<br>(reference) | The host starts issuing a resume signal (K) (in hardware). | T3< T0 + 1ms{TURSM} |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.2.4 Device Chirp Detection Function

A device Chirp is detected.

The device chirp detection function is turned on in the RESET state.

##### 6.3.9.2.4.1 When a Correct Device Chirp is Detected

The procedure for detecting a device Chirp is as described below:

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The device chirp detection function is turned on (T0).

(3) The device sends out a Chirp (T1).

(4) A device Chirp is recognized by a line state “K” (indicated by USB_Host_Status.LineState[1:0]) that continues for a specified time or more (T2).

(5) When the device Chirp is determined to have ended within a specified time after a reset started (i.e., the line state (USB_Host_Status.LineState[1:0]) has changed to ‘SE0’), a device chirp normal detection status (H_SIE_IntStat_0.DetectDevChirpOK) is issued (T3).

(6) When a device Chirp is detected, the device chirp detection function is turned off (T3).

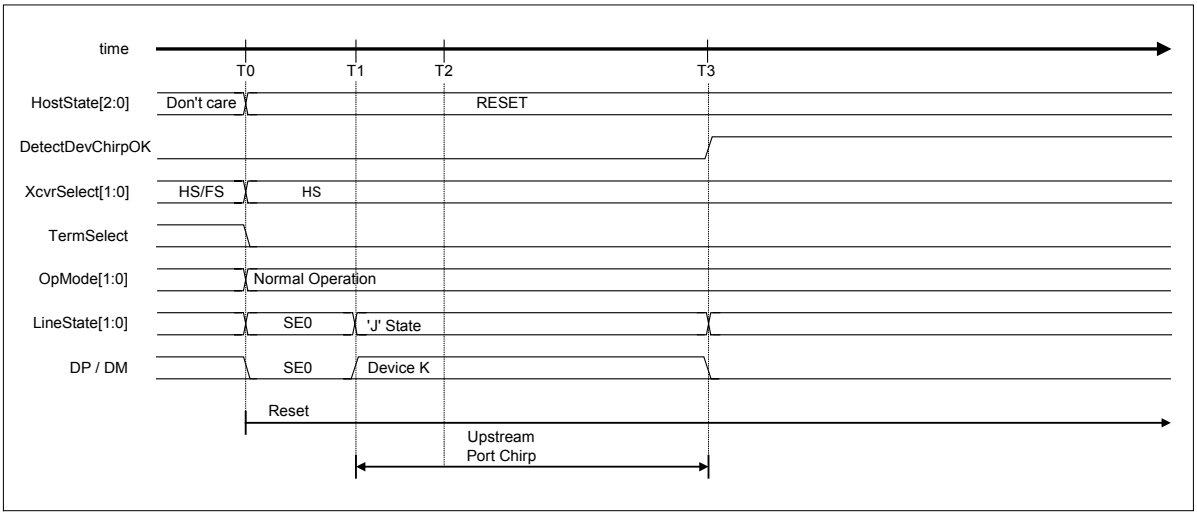



**Fig. 6.41 Device chirp timing**

**Table 6.36 Device Chirp Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESET (by firmware). | 0 (reference) |
| T1 | The device starts a Chirp. | T0 < T1 < T0 + 6.0ms |
| T2 | The device Chirp is recognized (by hardware). | T1 + 2.5us {TFILT} < T2 |
| T3 | The device completes the Chirp.<br>The device chirp detection function is turned off.<br>A device chirp normal detection status (DetectDevChirpOK) is issued (by hardware). | T1 + 1.0ms {TUCH} < T3 < T0 + 7.0ms {TUCHEND} |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.2.4.2 When an Erratic Device Chirp is Detected

The device chirp detection function assumes an error when a device Chirp does not end within a specified time, and issues a status accordingly.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The device chirp detection function is turned on (T0).

(3) The device sends out a Chirp (T1).

(4) A device Chirp is recognized by a line state "K" (indicated by USB_Host_Status.LineState[1:0]) that continues for a specified time or more (T2).

(5) Because the device Chirp does not end within a specified time after a reset started, an error is assumed and a device chirp abnormal detection status (H_SIE_IntStat_0.DetectDevChirpNG) is issued (T3).

(6) When a device Chirp is detected, the device chirp detection function is turned off (T3).

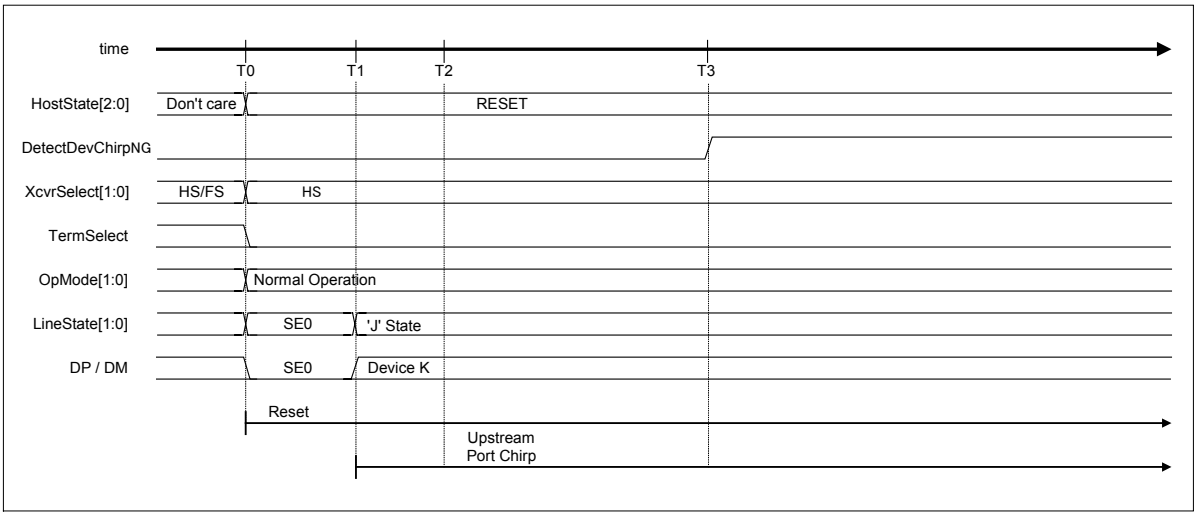



**Fig. 6.42 Device chirp timing (NG)**

**Table 6.37 Device Chirp Timing (NG) Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | USB_Control_0.AutoMode is set to GoRESET (by firmware). | 0 (reference) |
| T1 | The device starts a Chirp. | T0 < T1 < T0 + 6.0ms |
| T2 | The device Chirp is recognized (by hardware). | T1 + 2.5us {TFILT} < T2 |
| T3 | A device chirp abnormal detection status (DetectDevChirpNG) is issued (by hardware). | T0 + 7ms{TUCHEND} < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.2.5 Port Error Detection

In the OPERATIONAL state, port errors are detected.

A port error is raised when the LSI cannot detect the EOP even at the endpoint of the (micro) frame during a packet reception.

Once a port error is detected by the host, a port error detection status (H_FrameIntStat.PortErr) is issued to the firmware while transaction is stopped immediately. Thereafter, no transaction (including SOF) will be issued.

When a port error occurs, set the firmware as follows:

(1) Set H_NegoControl_0.AutoMode to GoDISABLED.

(2) Set H_NegoControl_0.ResetHTM to 1 and reset the host transceiver macro.

(3) After more than 3 cycles have elapsed with 60MHz clock, set H_NegoControl_0.ResetHTM to 0 and then cancel the reset on host transceiver macro.

### 6.3.9.3 Description of Individual Host State Management Support Function

#### 6.3.9.3.1 GoIDLE

Write 0x80 to the H_NegoControl_0 register (H_NegoControl_0.AutoModeCancel = 1 and H_NegoControl_0.AutoMode = 0x0) to cause operation of the state under execution to stop. The H_NegoControl_0.AutoModeCancel bit is cleared to 0 by a completion of the stop process (6 cycles required when operating with 60 MHz clock). After confirming that the H_NegoControl_0.AutoModeCancel bit is cleared to 0, write 0x01 to the same register (which sets the host state transition execution (H_NegoControl_0.AutoMode) to GoIDLE). This way, the processes required for a transition to IDLE will be automatically executed by the LSI's hardware.

Processes (3) to (8) are automatically executed by the LSI's hardware.

(1) Write 0x80 to the H_NegoControl_0 register (H_NegoControl_0.AutoModeCancel = 1 and H_NegoControl_0.AutoMode = 0x0) (T0).

(2) After confirming that the H_NegoControl_0.AutoModeCancel bit is cleared to 0, write 0x01 to the H_NegoControl_0 register (H_NegoControl_0.AutoMode = 0x1) (T1).

(3) The host state monitor (H_NegoControl_0.HostState) is set to IDLE (T1).

(4) VBUSEN is turned off (T1).

(5) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to FS mode (T1).

(6) Operation mode (H_XcvrControl.OpMode[1:0]) is set to NonDriving (T1).

(7) The transaction execution function of the USB host is immediately turned off (T1).

(8) All of the connection detection, disconnection detection, remote wakeup detection, and device chirp detection functions are turned off (T1).

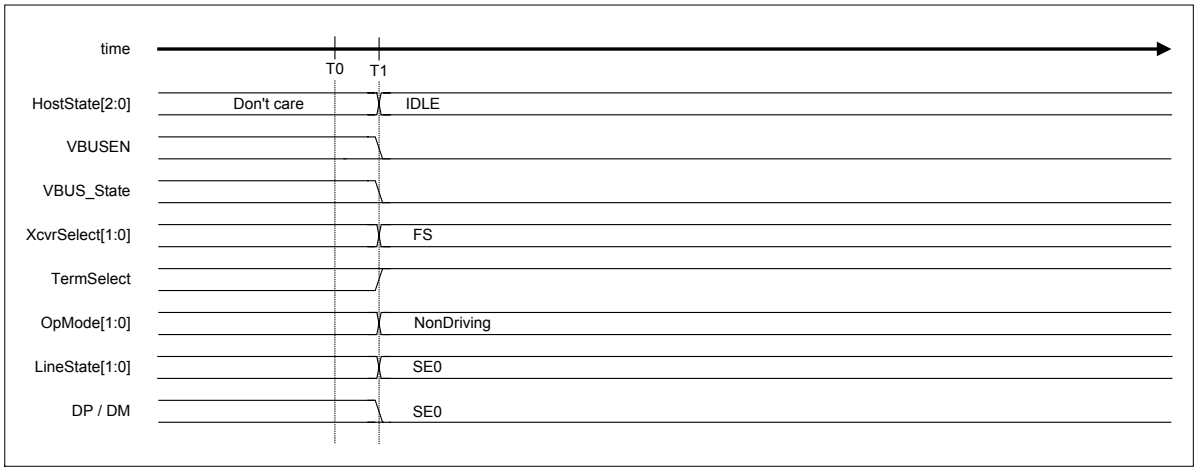



**Fig. 6.43 GoIDLE timing**

**Table 6.38 GoIDLE Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | The current state is suspended (by firmware). | 0 (reference) |
| T1 | Following confirmation that the host state transition execution cancel bit is cleared to 0, H_NegoControl_0.AutoMode is set to GoIDLE (by firmware).<br>VBUSEN is turned off.<br>Transceiver selection is set to FS mode.<br>Terminal selection is set to FS mode.<br>Operation mode is set to NonDriving.<br>The transaction execution function is immediately turned off.<br>The connection detection, disconnection detection, remote wakeup detection, and device chirp detection functions are turned off (by hardware). | T0 + 5cycle(60MHz) < T1 |

#### 6.3.9.3.2 GoWAIT_CONNECT

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoWAIT_CONNECT, the process required for transition to WAIT_CONNECT is automatically executed by the LSI's hardware.

Note that at this point, the HS device is connected as an FS device. It is made to operate as an HS device by HS Detection Handshaking that is performed in the subsequent reset operation.

##### 6.3.9.3.2.1 When FS Device is Connected

The procedure that is executed when an FS device is connected is described below. Processes (2) to (12) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoWAIT_CONNECT (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to WAIT_CONNECT (T0).

(3) VBUSEN is turned on (T0).

(4) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to FS mode (T0).

(5) Operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T0).

(6) The port speed (H_NegoControl_1.PortSpeed[1:0]) is set to FS (T0).

(7) After waiting a while for the internal device power supply to stabilize, the connection detection function is turned on (T1).

(8) When an FS device is connected, "J" appears in the line state (H_USB_Status.LineState[1:0]) (T2).

(9) A connection of an FS device is assumed by a fact that the line state "J" (indicated by H_USB_Status.LineState[1:0]) continues for 2.5 μs or more (T3).

(10) The disconnection detection function is turned on (T3).

(11) If a disconnection is not detected during the debounce interval period, a connection detected status (SIE_IntStat_0.DetectCon) is issued (T4). If a disconnection is detected during this period, the disconnection detection function is turned off and the process restarts from the connection detection phase beginning with (8). No disconnection detected status indication (H_SIE_IntStat_0.DetectDiscon) is issued.

(12) The disconnection detection function and connection detection function are turned off (T4).

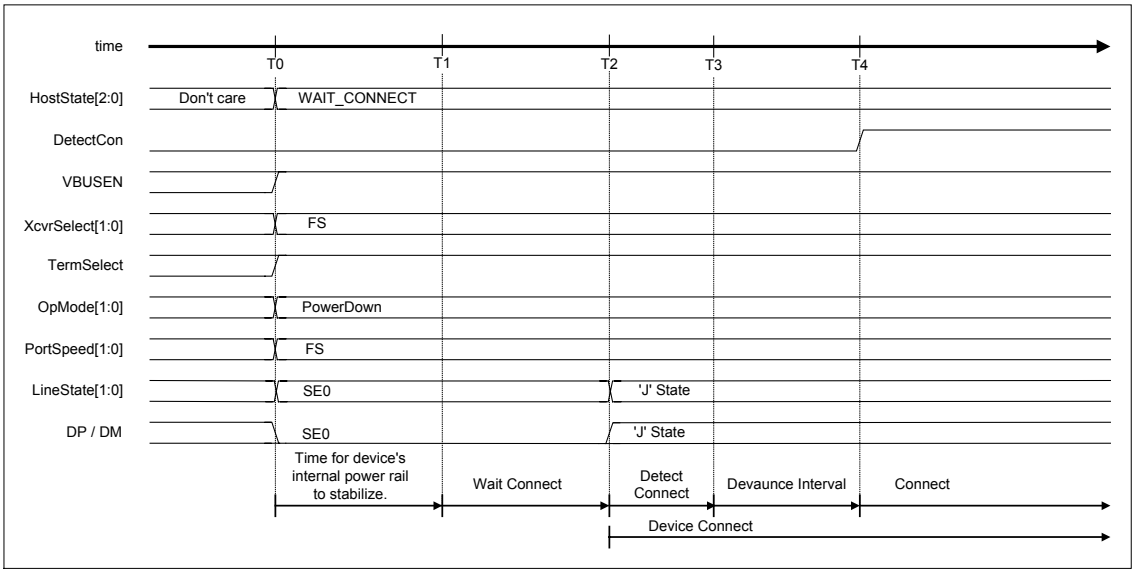



**Fig. 6.44 Device attach timing (FS mode)**

**Table 6.39 Device Attach Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoWAIT_CONNECT (by firmware). | 0 (reference) |
| T1 | The connection detection function is turned on (by hardware). | T0+100ms{TSIGATT} < T1 |
| T2 | A device is connected. | T2 |
| T3 | The disconnection detection function is turned on (by hardware). | T2+2.5us{TDCNN} < T3 |
| T4 | A connection detected status (DetectCon) is issued (by hardware).<br>The disconnection detection function and connection detection function are turned off (by hardware). | T3+100ms{TATTDB} <T4 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.2.2 When LS Device is Connected

The procedure that is executed when an LS device is connected is described below. Processes (2) to (14) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoWAIT_CONNECT (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to WAIT_CONNECT (T0).

(3) VBUSEN is turned on (T0).

(4) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to FS mode (T0).

(5) Operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T0).

(6) The port speed (H_NegoControl_1.PortSpeed[1:0]) is set to FS (T0).

(7) After waiting 100 ms for the internal device power supply to stabilize, the connection detection function is turned on (T1).

(8) When an LS device is connected, "K" appears in the line state (H_USB_Status.LineState[1:0]) (T2).

(9) A connection of an LS device is assumed by a fact that the line state "K" (indicated by H_USB_Status.LineState[1:0]) continues for 2.5 μs or more (T3).

(10) Transceiver selection (H_XcvrControl.XcvrSelect) is set to LS (T3). As a result, the polarity of the line state (H_USB_Status.LineState[1:0]) changes to LS, and "J" appears in the line state (H_USB_Status.LineState[1:0]).

(11) The port speed (H_NegoControl_1.PortSpeed[1:0]) is set to LS (T3).

(12) The disconnection detection function is turned on (T3).

(13) If a disconnection is not detected during the debounce interval period, a connection detected status (SIE_IntStat_0.DetectCon) is issued (T4). If a disconnection is detected during this period, the disconnection detection function is turned off and transceiver selection (H_XcvrControl.XcvrSelect) and port speed (H_NegoControl1.PortSpeed [1:0]) are both set to FS before the process restarts from the connection detection phase beginning with (8). No disconnection detected status indication (H_SIE_IntStat_0.DetectDiscon) is issued.

(14) The disconnection detection function and connection detection function are turned off (T4).

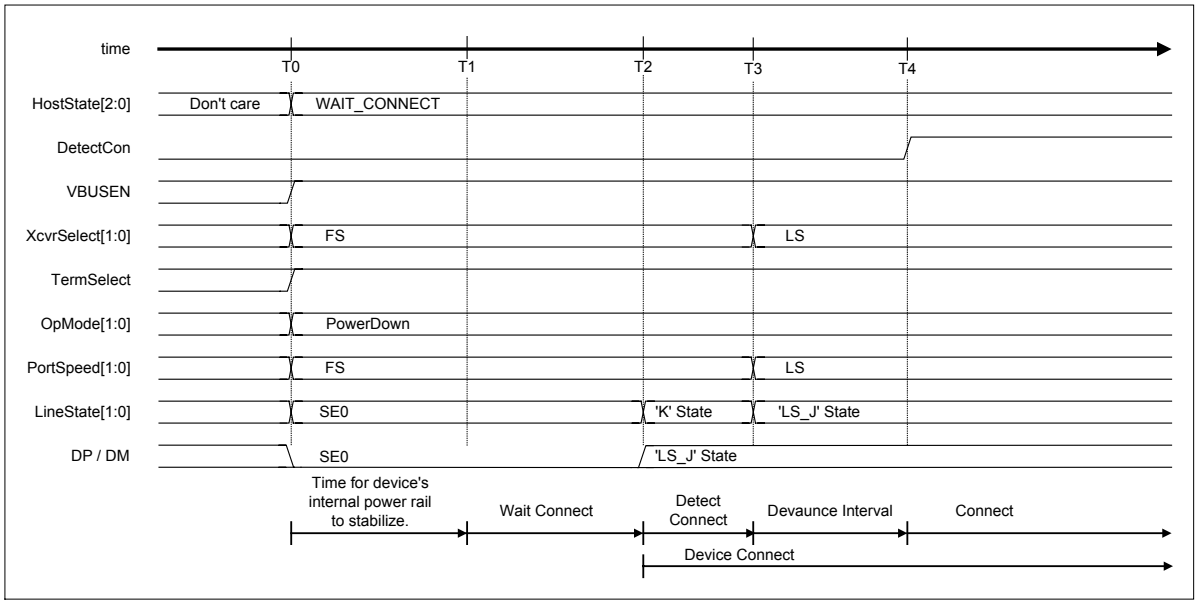



**Fig. 6.45 Device attach timing (LS mode)**

**Table 6.40 Device Attach Timing Values (LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoWAIT_CONNECT (by firmware). | 0 (reference) |
| T1 | The connection detection function is turned on (by hardware). | T0+100ms{TSIGATT} < T1 |
| T2 | A device is connected. | T2 |
| T3 | The disconnection detection function is turned on (by hardware). | T2+2.5us{TDCNN} < T3 |
| T4 | A connection detected status (DetectCon) is issued (by hardware).<br>The disconnection detection function and connection detection function are turned off (by hardware). | T3+100ms{TATTDB} <T4 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.3 GoDISABLED

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoDISABLED, the processes required for a transition to DISABLED is automatically executed by the LSI's hardware.

The LSI shifts to this state when a connection is detected in the WAIT_CONNECT state, a Chirp from an erratic device is detected in the RESET state, or a port error is detected in the OPERATIONAL state.

##### 6.3.9.3.3.1 When HS Device is Connected

The procedure that is executed when an HS device is connected is described below. Processes (2) to (6) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoDISABLED (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to DISABLED (T0).

(3) The disconnection detection function is turned off (T0).

(4) After waiting for the currently executed transaction to complete, transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) and the port speed (H_NegoControl_1.PortSpeed[1:0]) all are set to FS mode, and operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T1).

(5) The disconnection detection function is turned on (T3).

(6) A transition-to-disabled complete status (H_SIE_IntStat_1.DisabledCmp) is issued to the firmware (T3).

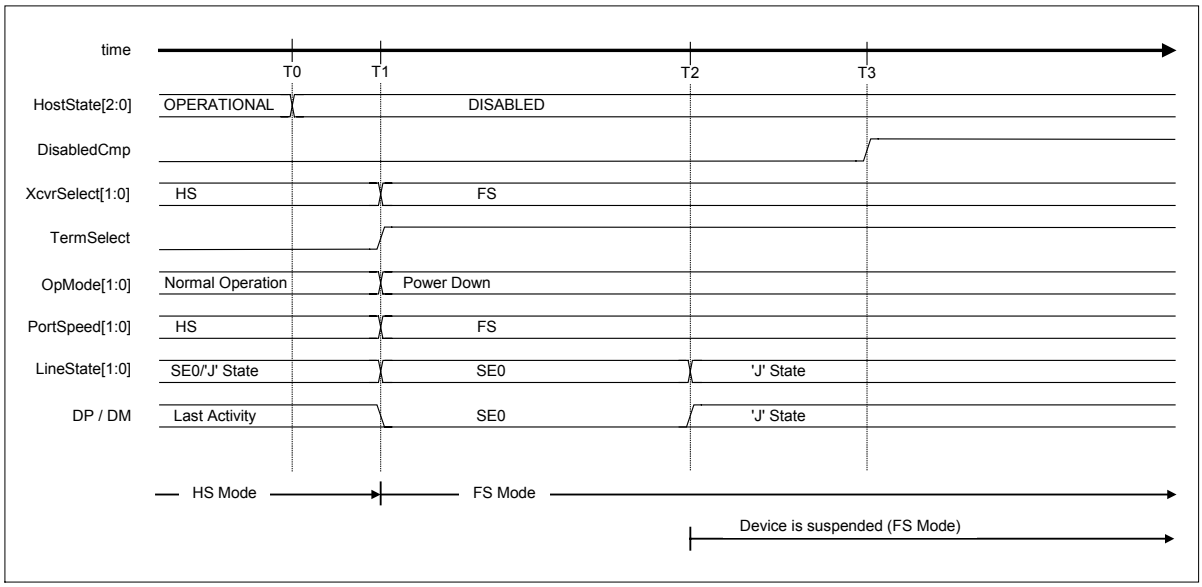



**Fig. 6.46 Disabled timing (HS mode)**

**Table 6.41 Disabled Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoDISABLED (by firmware).<br>The disconnection detection function is turned off (by hardware). | 0 (reference) |
| T1 | After last bus activity is complete, transceiver selection and terminal selection and port speed are set to FS mode, and operation mode (XcvrControl.OpMode[1:0]) is set to PowerDown (by hardware). | T1 |
| T2 | The device detects Suspend and shifts to FS mode. | T1 + 3.0ms < T2 {TWTREV} < T1 + 3.125ms |
| T3 | The disconnection detection function is turned on (by hardware).<br>A transition-to-disabled complete status is issued (by hardware). | T1 + 4ms < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.3.2 When FS Device is Connected

The procedure that is executed when an FS device is connected is described below. Processes (2) to (6) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoDISABLED (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to DISABLED (T0).

(3) The disconnection detection function is turned off (T0).

(4) After waiting for the currently executed transaction to complete, operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T1).

(5) The disconnection detection function is turned on (T3).

(6) A transition-to-disabled complete status (H_SIE_IntStat_1.DisabledCmp) is issued to the firmware (T3).

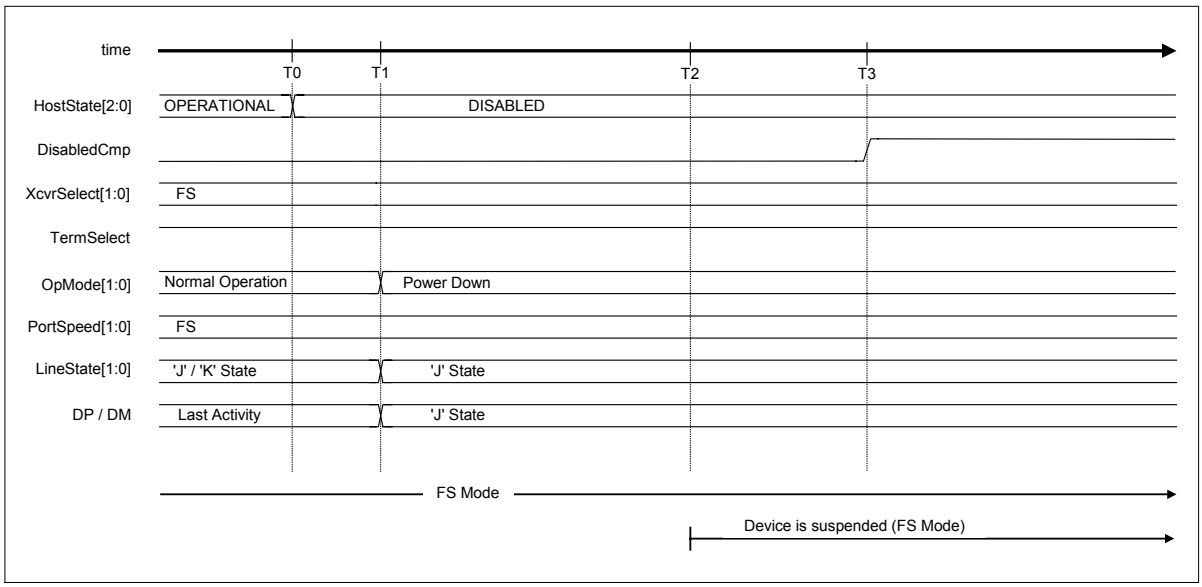



**Fig. 6.47 Disabled timing (FS mode)**

**Table 6.42 Disabled Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoDISABLED (by firmware).<br>The disconnection detection function is turned off (by hardware). | 0 (reference) |
| T1 | After last bus activity is complete, operation mode (XcvrControl.OpMode[1:0]) is set to PowerDown (by hardware). | T1 |
| T2 | The device detects Suspend. | T1 + 3.0ms < T2 {TWTREV} < T1 + 3.125ms |
| T3 | The disconnection detection function is turned on (by hardware).<br>A transition-to-disabled complete status is issued (by hardware). | T1 + 4ms < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.3.3 When LS Device is Connected

The procedure that is executed when an LS device is connected is described below. Processes (2) to (6) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoDISABLED (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to DISABLED (T0).

(3) The disconnection detection function is turned off (T0).

(4) After waiting for the currently executed transaction to complete, operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T1).

(5) The disconnection detection function is turned on (T3).

(6) A transition-to-disabled complete status (H_SIE_IntStat_1.DisabledCmp) is issued to the firmware (T3).

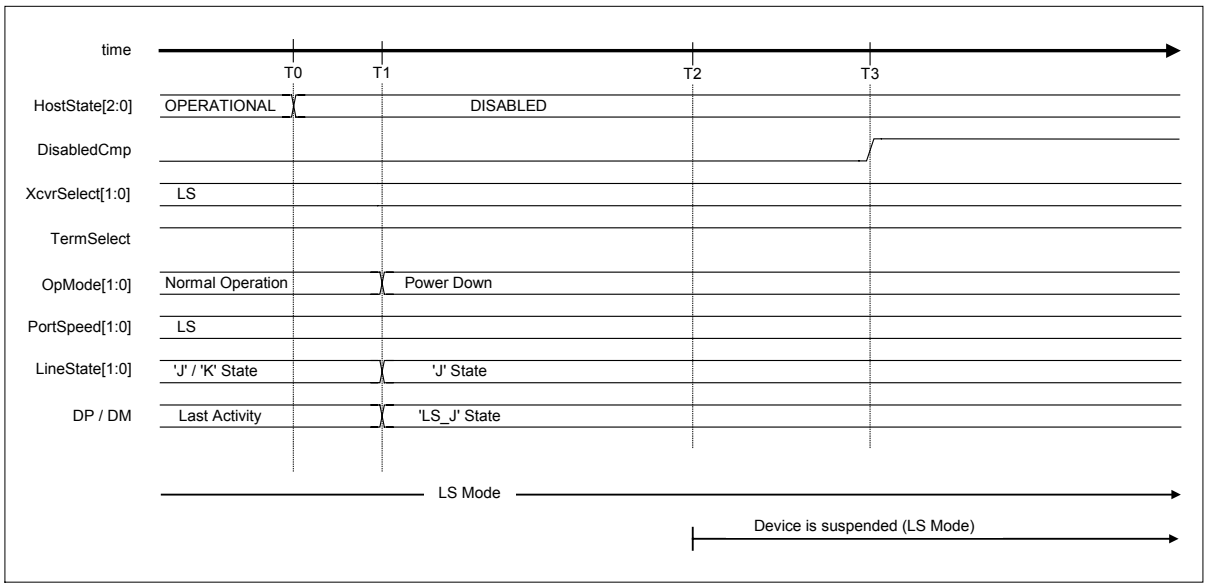



**Fig. 6.48 Disabled timing (LS mode)**

**Table 6.43 Disabled Timing Values (LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoDISABLED (by firmware).<br>The disconnection detection function is turned off (by hardware). | 0 (reference) |
| T1 | After last bus activity is complete, operation mode (XcvrControl.OpMode[1:0]) is set to PowerDown (by hardware). | T1 |
| T2 | The device detects Suspend. | T1 + 3.0ms < T2 {TWTREV} < T1 + 3.125ms |
| T3 | The disconnection detection function is turned on (by hardware).<br>A transition-to-disabled complete status is issued (by hardware). | T1 + 4ms < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.4 GoRESET

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESET, the processes required for a transition to RESET is automatically executed by the LSI's hardware. When transition to this state is to be executed from the OPERATIONAL state, the LSI will wait for the completion of the transaction under execution by the hardware and then start the RESET processing.

##### 6.3.9.3.4.1 Reset for an HS Device

The procedure that is executed when an HS device is to be reset is described below. Processes (2) to (14) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESET (T0).

(3) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to HS mode (T0).

(4) Operation mode (H_XcvrControl.OpMode[1:0]) is set to Normal (T0).

(5) The device chirp detection function is turned on (T0).

(6) A device Chirp is recognized by a line state (H_USB_Status.LineState[1:0]) that indicates presence of activity (seen as 'J' state) continuously for 2.5 μs or more. Then, when the device Chirp is determined to have ended within a specified time after a reset started (i.e., the line state (USB_Host_Status.LineState[1:0]) has changed to 'SE0'), a device chirp normal detection status (H_SIE_IntStat_0.DetectDevChirpOK) is issued (T2).

(7) The device chirp detection function is turned off (T3).

(8) After the device Chirp is complete, the host starts the output of Chirp K (T3).

(9) The host switches its output from Chirp K to Chirp J (T4)

(10) The host switches its output from Chirp J to Chirp K (T5). Thereafter, the host outputs Chirp K and Chirp J sequences alternately.

(11) Upon detecting a host Chirp, the device shifts to HS mode (T6). The fact that the Chirp level changed beginning with T7 means that the HS termination on the device side is enabled. Normally, Chirp is approximately 800 mV when the device is in FS mode, and approximately 400 mV when the device is in HS mode.

(12) The host completes Chirp (T8).

(13) Reset is complete (T9).

(14) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T9).

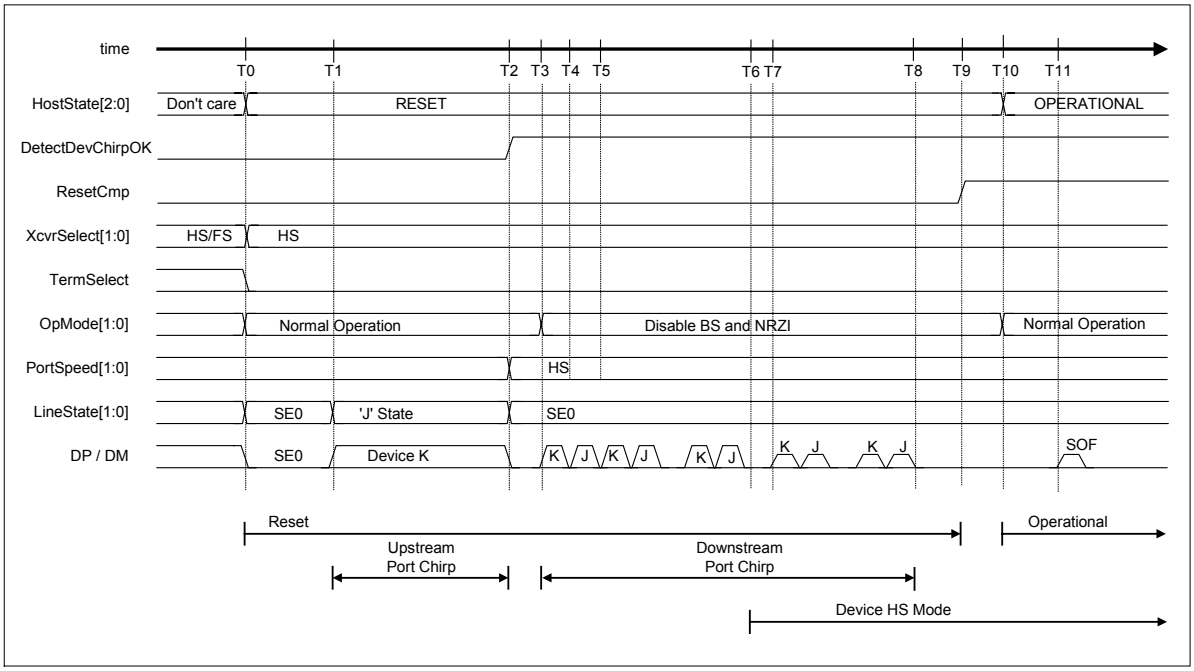



**Fig. 6.49 Reset timing (HS mode)**

**Table 6.44 Reset Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESET (by firmware).<br>The device chirp detection function is turned on (by hardware). | 0 (reference) |
| T1 | The device starts a Chirp. | T0 < T1 < T0 + 6.0ms |
| T2 | The device completes a Chirp.<br>The port speed is set to HS.<br>The device chirp detection function is turned off.<br>A device chirp normal detection status (DetectDevChirpOK) is issued (by hardware). | T1 + 1.0ms {TUCH} < T2 <<br>T0 + 7.0ms {TUCHEND} |
| T3 | The host outputs the first Chirp (Chirp K) (in hardware). | T2 < T3 < T2 + 100us {TWTDCH} |
| T4 | The host switches its output from Chirp K to Chirp J (in hardware). | T3 + 40us {TDCHBIT} < T4 <<br>T3 + 60us {TDCHBIT} |
| T5 | The host switches its output from Chirp J to Chirp K (in hardware). | T4 + 40us {TDCHBIT} < T5 <<br>T4 + 60us {TDCHBIT} |
| T6 | The device detects a host Chirp. | T6 |
| T7 | The device shifts to HS mode. | T6 < T7 < T6 + 500us |
| T8 | The host completes Chirp (in hardware). | T3+ 50ms{TDRSTR} < T8 |
| T9 | After reset, a reset completed status (ResetCmp) is issued (by hardware). | T8 < T9 <T8 + 150us |
| T10 (reference) | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware). | T10 < T9 + 200us |
| T11 (reference) | First SOF is sent out (by hardware). | T10 + 120us < T11 < T10 + 130us<br>T8 + 100us{TDCHSE0} < T11 <<br>T8 + 500us {TDCHSE0} |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.4.2 Erratic Device Chirp Detected

Behavior of the LSI when a device Chirp is found erratic in HS Detection Handshaking is shown below. Two operation modes are available to choose from depending on how the chirp complete disable (H_NegoControl_1.DisChirpFinish) is set.

##### 6.3.9.3.4.2.1 When Chirp Complete Disable (H_NegoControl_1.DisChirpFinish) = 0

A host Chirp is not performed after an error is detected. If a device chirp abnormal detection status is issued, the firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoDISABLED after waiting for a reset complete status (H_SIE_IntStat_1.ResetCmp) to be issued, thereby shifting the host into the DISABLED state. Processes (2) to (9) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESET (T0).

(3) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to HS mode (T0).

(4) Operation mode (H_XcvrControl.OpMode[1:0]) is set to Normal (T0).

(5) The device chirp detection function is turned on (T0).

(6) A device Chirp is recognized by a line state (H_USB_Status.LineState[1:0]) that indicates presence of activity (seen as 'J' state) continuously for 2.5 µs or more. However, because the device Chirp does not end within a specified time after a reset started, an error is assumed and a device chirp abnormal detection status (H_SIE_IntStat_0.DetectDevChirpNG) is issued (T2).

(7) The device chirp detection function is turned off (T2).

(8) Reset is complete (T3).

(9) A reset completed status (H_SIE_IntStat_1.ResetCmp) is issued (T3).

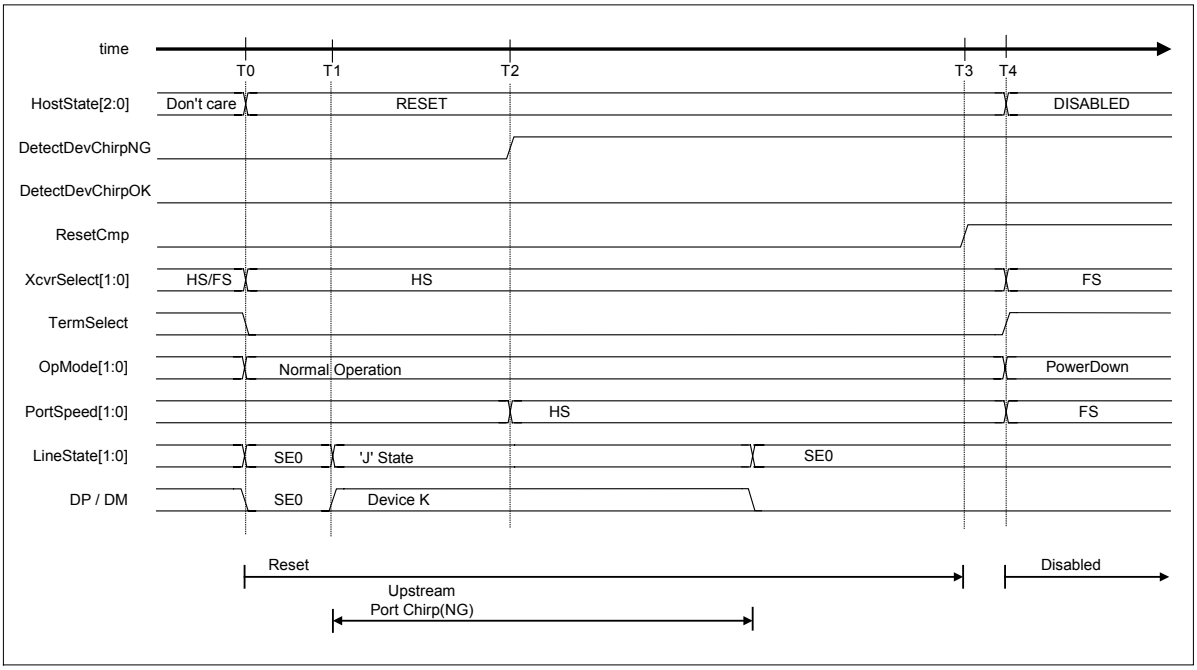



**Fig. 6.50 Detect device chirp NG timing (DisChirpFinish = 0)**

**Table 6.45 Detect Device Chirp NG Timing Values (DisChirpFinish = 0)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | USB_Control_0. AutoMode is set to GoRESET (by firmware).<br>The device chirp detection function is turned on (by hardware). | 0 (reference) |
| T1 | The device starts a Chirp. | T0 < T1 < T0 + 6.0ms |
| T2 | A device chirp abnormal detection status (DetectDevChirpNG) is issued.<br>The device chirp detection function is turned off (by hardware). | T0 + 7ms{TUCHEND} < T2 |
| T3 | Reset is complete. A reset complete status (ResetCmp) is issued (by hardware). | T2 + 50ms{TDRSTR }< T3 |
| T4 (reference) | H_NegoControl_0.AutoMode is set to GoDISABLED (by firmware). | |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.4.2.2 When Chirp Complete Disable (H_NegoControl_1.DisChirpFinish) = 1

A host Chirp is performed after an error is detected.

When transitioning the host state to DISABLED in this mode without waiting for the reset complete status signal (H_SIE_IntStat_1.ResetCmp) to be issued, write 0x80 to the H_NegoControl_0 register (H_NegoControl_0.AutoModeCancel = 1 and H_NegoControl_0.AutoMode = 0x0) to exit the current state. The H_NegoControl_0.AutoModeCancel bit is cleared to 0 by stop processing (about 6 cycles required when operating with 60 MHz clock). After confirming that the H_NegoControl_0.AutoModeCancel bit has been cleared to 0, write 0x03 to the same register (which sets the host state transition execution (H_NegoControl_0.AutoMode) to GoDISABLED).

Processes (2) to (15) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESET (T0).

(3) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to HS mode (T0).

(4) Operation mode (H_XcvrControl.OpMode[1:0]) is set to Normal (T0).

(5) The device chirp detection function is turned on (T0).

(6) A device Chirp is recognized by a line state (H_USB_Status.LineState[1:0]) that indicates presence of activity (shown as 'J' state) continuously for 2.5 μs or more. However, because the device Chirp does not end within a specified time after a reset started, an error is assumed and a device chirp abnormal detection status (H_SIE_IntStat_0.DetectDevChirpNG) is issued (T2).

(7) The device chirp detection function is turned off (T2).

(8) When the device Chirp is determined to have ended by a line state (USB_Host_Status.LineState[1:0]) that indicates absence of activity (shown as 'SE0'), a device chirp normal detection status (H_SIE_IntStat_0.DetectDevChirpOK) is issued (T3).

(9) After the device Chirp is complete, the host starts the output of Chirp K (T4).

(10) The host switches its output from Chirp K to Chirp J (T5).

(11) The host switches its output from Chirp J to Chirp K (T6). Thereafter, the host outputs Chirp K and Chirp J sequences alternately.

(12) Upon detecting a host Chirp, the device shifts to HS mode (T7). The fact that the Chirp level changed beginning with T8 means that the HS termination on the device side is enabled. Normally, Chirp is approximately 800 mV when the device is in FS mode, and approximately 400 mV when the device is in HS mode.

(13) The host completes Chirp (T9).

(14) Reset is complete (T10).

(15) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T10).

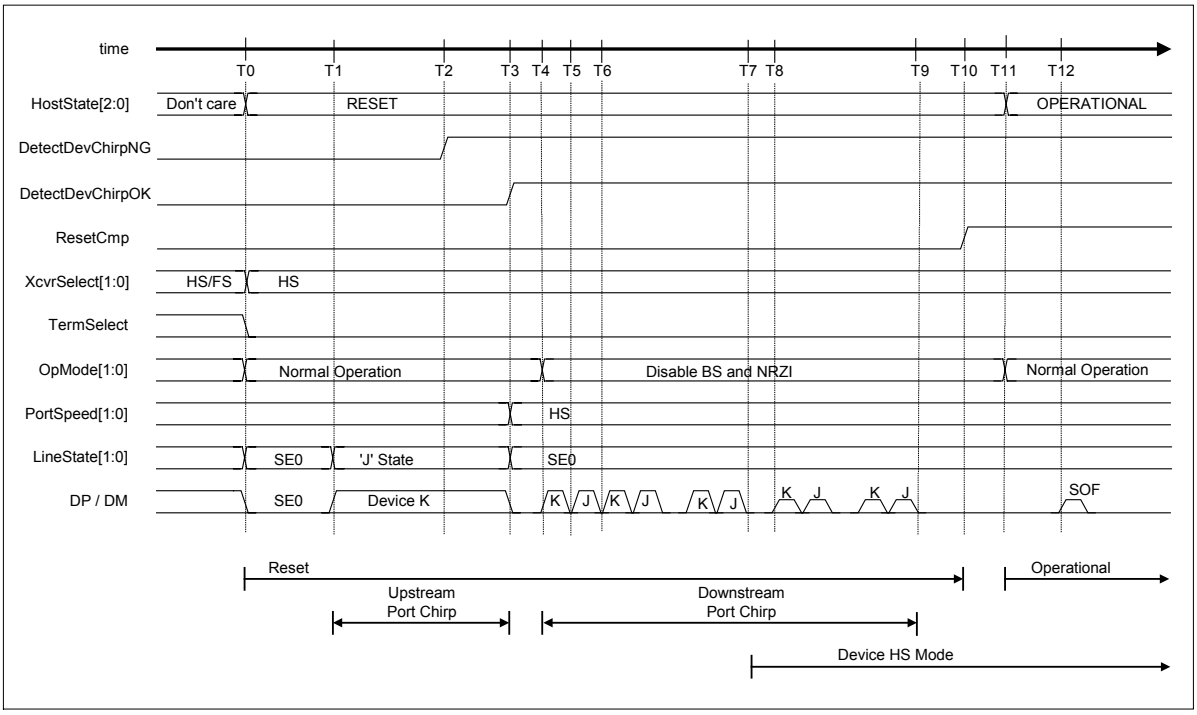



**Fig. 6.51 Detect device chirp NG timing (DisChirpFinish = 1)**

**Table 6.46 Detect Device Chirp NG Timing Values (DisChirpFinish = 1)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESET (by firmware).<br>The device chirp detection function is turned on (by hardware). | 0 (reference) |
| T1 | The device starts a Chirp. | T0 < T1 < T0 + 6.0ms |
| T2 | A device chirp abnormal detection status (DetectDevChirpNG) is issued.<br>The device chirp detection function is turned off (by hardware). | T0 + 7ms{TUCHEND} < T2 |
| T3 | The device completes a Chirp.<br>The port speed is set to HS.<br>A device chirp normal detection status (DetectDevChirpOK) is issued (by hardware). | T3 |
| T4 | The host outputs the first Chirp (Chirp K) (in hardware). | T3< T4 < T3 + 100us {TWTDCH} |
| T5 | The host switches its output from Chirp K to Chirp J (in hardware). | T4 + 40us {TDCHBIT} < T45<br>T4 + 60us {TDCHBIT} |
| T6 | The host switches its output from Chirp J to Chirp K (in hardware). | T5 + 40us {TDCHBIT} < T6 <<br>T5 + 60us {TDCHBIT} |
| T7 | The device detects a host Chirp. | T7 |
| T8 | The device shifts to HS mode. | T7 < T8 < T6 + 500us |
| T9 | The host completes Chirp (in hardware). | T4 + 50ms{TDRSTR} < T9 |
| T10 | Reset is complete. A reset complete status (ResetCmp) is issued (by hardware). | T9 < T10 <T9 + 150us |
| T11 (reference) | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware). | T11< T10 + 200us |
| T12 (reference) | First SOF is sent out (by hardware). | T11 + 120us < T12 < T11 + 130us<br>T9+ 100us{TDCHSE0} < T12 <<br>T9 + 500us {TDCHSE0} |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.4.3 Reset for an FS Device

The procedure that is executed when an FS device is to be reset is described below. Processes (2) to (9) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESET (T0).

(3) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to HS mode (T0).

(4) Operation mode (H_XcvrControl.OpMode[1:0]) is set to Normal (T0).

(5) The device chirp detection function is turned on (T0).

(6) Because no device Chirp is detected and the port speed (H_NegoControl_1.PortSpeed[1:0]) = HS/FS, the other party device is assumed to be an FS device, setting the transceiver selection (H_XcvrControl.XcvrSelect) and port speed (H_USB_Status.PortSpeed[1:0]) to FS (T1).

(7) The device chirp detection function is turned off (T1).

(8) Terminal selection (H_XcvrControl.TermSelect) is set to FS (T2).

(9) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T3).

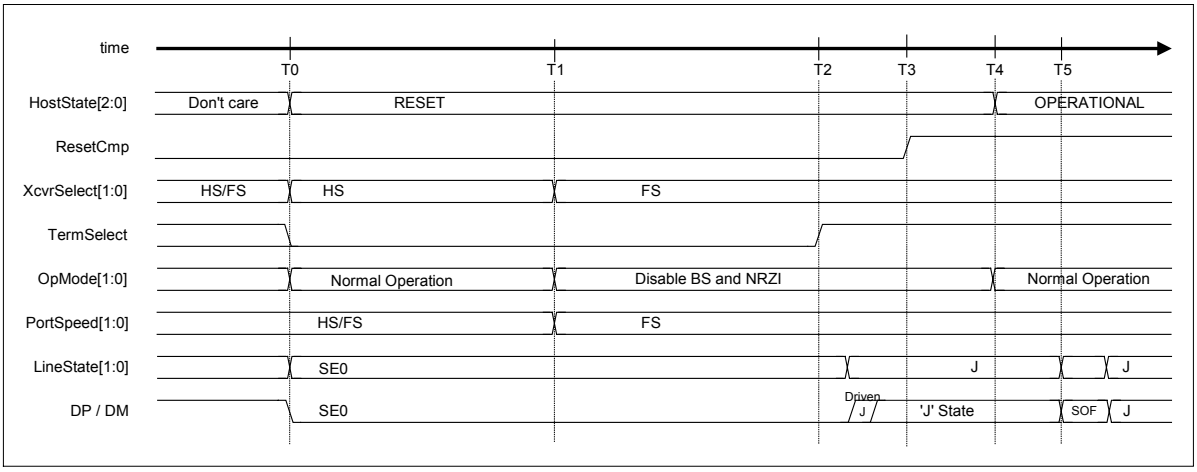



**Fig. 6.52 Reset timing (FS mode)**

**Table 6.47 Reset Timing Values (FS mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESET (by firmware).<br>The device chirp detection function is turned on (by hardware). | 0 (reference) |
| T1 | Transceiver selection is set to FS.<br>Port speed is set to FS.<br>The device chirp detection function is turned off (by hardware). | T0 + 7.0ms{TUCHEND} < T1 |
| T2 | Terminal selection is set to FS (by hardware). | T0 + 50ms{TDRSTR} < T2 |
| T3 | A reset complete status is issued (by hardware). | T2 + 150us < T3 |
| T4 (reference) | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware). | T4 |
| T5 (reference) | First SOF is sent out (by hardware). | T4 + 0.9ms < T5 < T4 + 1.1ms<br>(T5 < T2 + 3ms) |

Note: Names stipulated in the USB2.0 Standard are shown in { } .

#### 6.3.9.3.4.4 Reset for an LS Device

The procedure that is executed when an LS device is to be reset is described below. Processes (2) to (7) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESET (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESET (T0).

(3) Transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to HS mode (T0).

(4) Operation mode (H_XcvrControl.OpMode[1:0]) is set to Normal (T0).

(5) Because the port speed (H_NegoControl_1.PortSpeed[1:0]) = LS, the other party device is assumed to be an LS device, so that transceiver selection (H_XcvrControl.XcvrSelect) is set to LS (T1).

(6) Terminal selection (H_XcvrControl.TermSelect) is set to FS (T2).

(7) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T3).

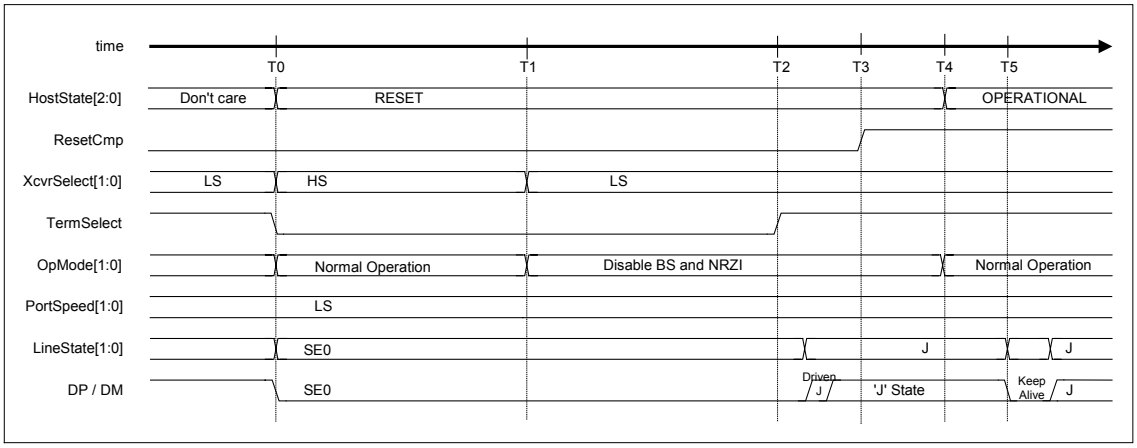



**Fig. 6.53 Reset timing (LS mode)**

**Table 6.48 Reset Timing Values (LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESET (by firmware). | 0 (reference) |
| T1 | Transceiver selection is set to LS (by hardware). | T0 + 7.0ms{TUCHEND} < T1 |
| T2 | Terminal selection is set to FS (by hardware). | T0 + 50ms{TDRSTR} < T2 |
| T3 | A reset complete status is issued (by hardware). | T2 + 150us < T3 |
| T4 (reference) | H_NegoControl_0. AutoMode is set to GoOPERATIONAL (by firmware). | T4 |
| T5 (reference) | First KeepAlive is sent out (by hardware). | T4 + 0.9ms < T5 < T4 + 1.1ms (T5 < T2 + 3ms) |

Note: Names stipulated in the USB2.0 Standard are shown in { } .

#### 6.3.9.3.5 GoOPERATIONAL

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoOPERATIONAL, the processes required for a transition to OPERATIONAL is automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoOPERATIONAL (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to OPERATIONAL (T0).

(3) Operation mode (H_XcvrControl.OpMode[1:0]) is set to Normal and a state is thereby entered in which USB transactions can be executed (T0).

(4) The disconnection detection function is turned on (T0).

(5) If the port speed (H_NegoControl_1.PortSpeed[1:0]) is set to HS or FS, the first SOF is issued; if the port speed is set to LS, the first KeepAlive is issued (T1). Thereafter, transfers are performed according to channel settings.

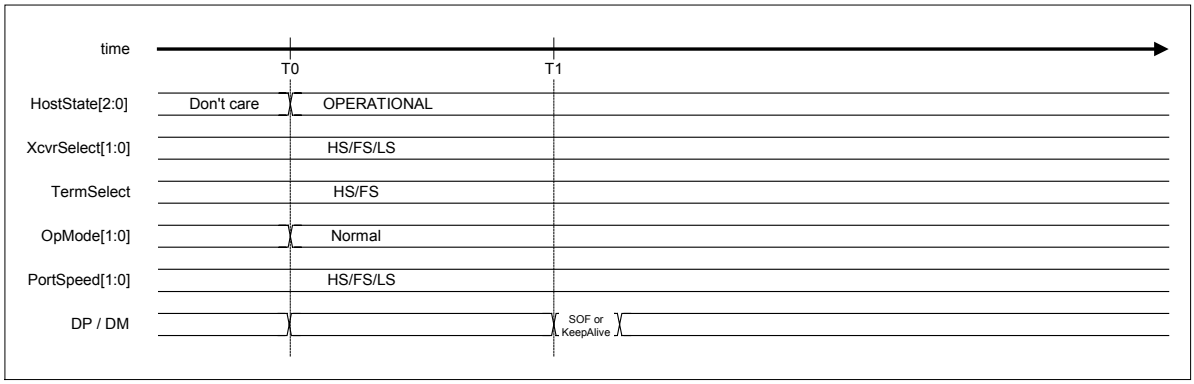



**Fig. 6.54 GoOPERATIONAL timing**

**Table 6.49 GoOPERATIONAL Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware). | 0 (reference) |
| T1 | First SOF (HS/FS) or first KeepAlive (LS) is issued. | T0+120us < T1(HS) < T0 + 130us<br>T0+0.9ms <T1(FS,LS) < T0 + 1.1ms |

#### 6.3.9.3.6 GoSUSPEND

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoSUSPEND, the processes required for a transition to SUSPEND is automatically executed by the LSI's hardware.

##### 6.3.9.3.6.1 When HS Device is Connected

The procedure that is executed when an HS device is connected is described below. Processes (2) to (7) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoSUSPEND (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to SUSPEND (T0).

(3) The disconnection detection and remote wakeup detection functions are turned off (T0).

(4) After waiting for the currently executed transaction to complete, transceiver selection (H_XcvrControl.XcvrSelect) and terminal selection (H_XcvrControl.TermSelect) are set to FS mode, and operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T1).

(5) The disconnection detection function is turned on (T3).

(6) If the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb) is enabled, the remote wakeup detection function is turned on (T3).

(7) A transition-to-suspend complete status (H_SIE_IntStat_1.SuspendCmp) is issued to the firmware (T3).

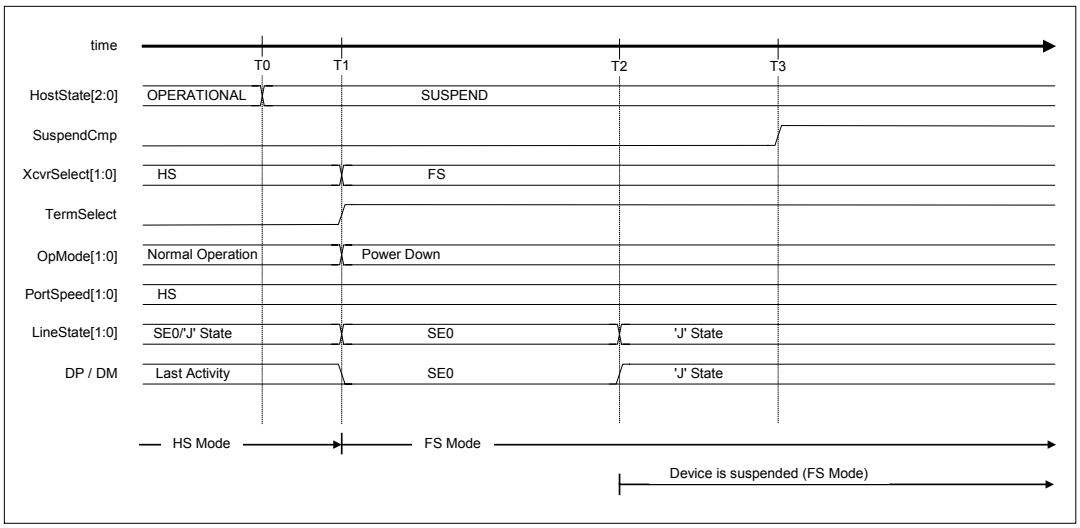



**Fig. 6.55 Suspend timing (HS mode)**

**Table 6.50 Suspend Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoSUSPEND (by firmware).<br>The disconnection detection and remote wakeup detection functions are turned off (by hardware). | 0 (reference) |
| T1 | After last bus activity is complete, transceiver selection and terminal selection are set to FS mode, and operation mode (XcvrControl.OpMode[1:0]) is set to PowerDown (by hardware). | T1 |
| T2 | The device detects Suspend and shifts to FS mode. | T1 + 3.0ms < T2 {TWTREV} < T1 + 3.125ms |
| T3 | The disconnection detection function is turned on.<br>If the remote wakeup acceptance enable is enabled, the remote wakeup detection function is turned on.<br>A transition-to-suspend complete status is issued (by hardware). | T1 + 5ms {TWTRSM} < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.6.2 When FS Device is Connected

The procedure that is executed when an FS device is connected is described below. Processes (2) to (7) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoSUSPEND (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to SUSPEND (T0).

(3) The disconnection detection and remote wakeup detection functions are turned off (T0).

(4) After waiting for the currently executed transaction to complete, operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T1).

(5) The disconnection detection function is turned on (T3).

(6) If the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb) is enabled, the remote wakeup detection function is turned on (T3).

(7) A transition-to-suspend complete status (H_SIE_IntStat_1.SuspendCmp) is issued to the firmware (T3).

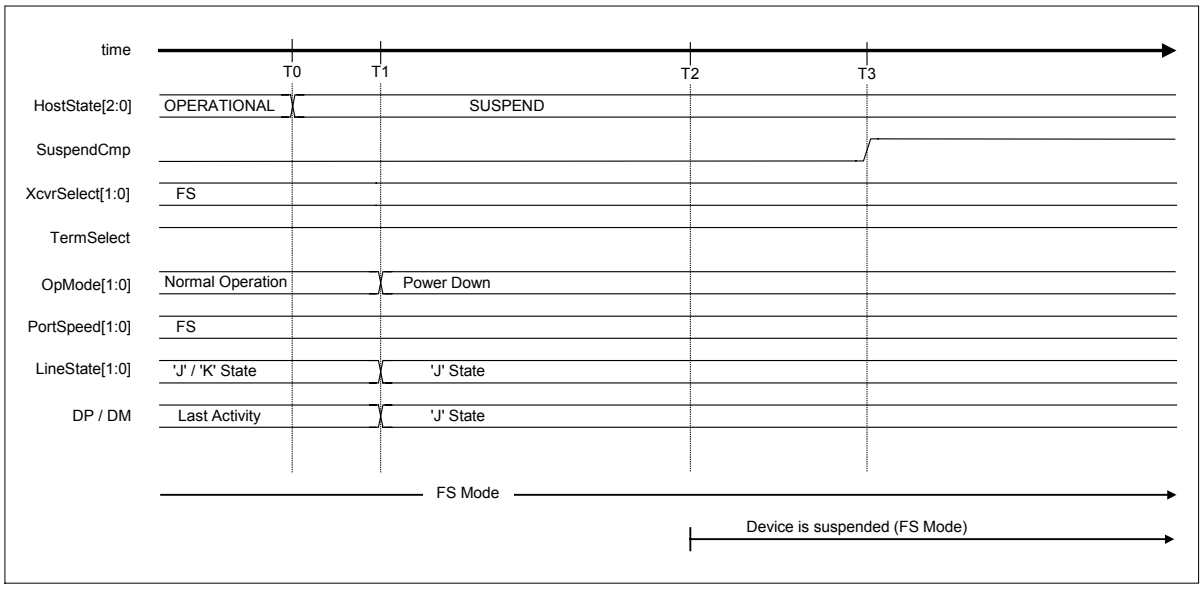



**Fig. 6.56 Suspend timing (FS mode)**

**Table 6.51 Suspend Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoSUSPEND (by firmware).<br>The disconnection detection and remote wakeup detection functions are turned off (by hardware). | 0 (reference) |
| T1 | After last bus activity is complete, operation mode (XcvrControl.OpMode[1:0]) is set to PowerDown (by hardware). | T1 |
| T2 | The device detects Suspend. | T1 + 3.0ms < T2 {TWTREV} < T1 + 3.125ms |
| T3 | The disconnection detection function is turned on.<br>If the remote wakeup acceptance enable is enabled, the remote wakeup detection function is turned on.<br>A transition-to-suspend complete status is issued (by hardware). | T1+ 5ms {TWTRSM} < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.6.3 When LS Device is Connected

The procedure that is executed when an LS device is connected is described below. Processes (2) to (7) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoSUSPEND (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to SUSPEND (T0).

(3) The disconnection detection and remote wakeup detection functions are turned off (T0).

(4) After waiting for the currently executed transaction to complete, operation mode (H_XcvrControl.OpMode[1:0]) is set to PowerDown (T1).

(5) The disconnection detection function is turned on (T3).

(6) If the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb) is enabled, the remote wakeup detection function is turned on (T3).

(7) A transition-to-suspend complete status (H_SIE_IntStat_1.SuspendCmp) is issued to the firmware (T3).

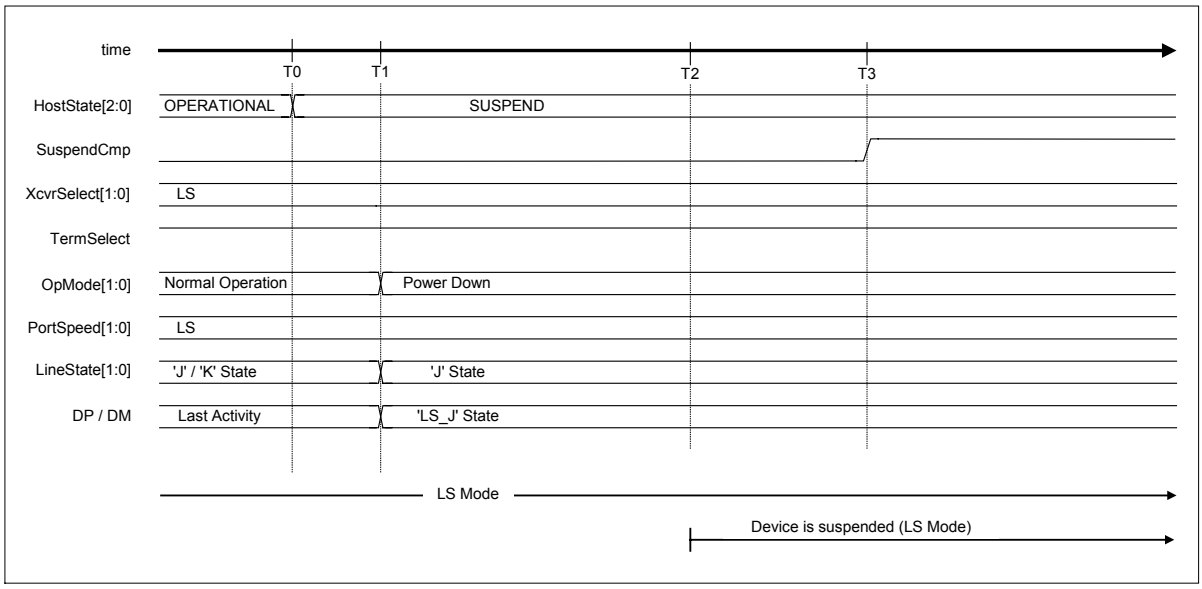



**Fig. 6.57 Suspend timing (LS mode)**

**Table 6.52 Suspend Timing Values (LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoSUSPEND (by firmware).<br>The disconnection detection and remote wakeup detection functions are turned off (by hardware). | 0 (reference) |
| T1 | After last bus activity is complete, operation mode (XcvrControl.OpMode[1:0]) is set to PowerDown (by hardware). | T1 |
| T2 | The device detects Suspend. | T1 + 3.0ms < T2 {TWTREV} < T1 + 3.125ms |
| T3 | The disconnection detection function is turned on.<br>If the remote wakeup acceptance enable is enabled, the remote wakeup detection function is turned on.<br>A transition-to-suspend complete status is issued (by hardware). | T1+ 5ms {TWTRSM} < T3 |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.7 GoRESUME

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESUME, the processes required for a transition to RESUME is automatically executed by the LSI's hardware.

##### 6.3.9.3.7.1 When HS Device is Connected

The procedure that is executed when an HS device is connected is described below. Processes (2) to (8) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESUME (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESUME (T0).

(3) The disconnection detection and remote wakeup detection functions are turned off (T0).

(4) Operation mode (H_XcvrControl.OpMode) is set to Disable BS and NRZI, and the hardware thereby starts issuing a resume "K" signal (T0).

(5) The hardware finishes issuing the resume "K" signal (T1).

(6) Terminal selection (H_XcvrControl.TermSelect) is set to HS (T2).

(7) Transceiver selection (H_XcvrControl.XcvrSelect) is set to HS (T3).

(8) A resume complete status (SIE_IntStat_1.ResumeCmp) is issued to the firmware (T3).

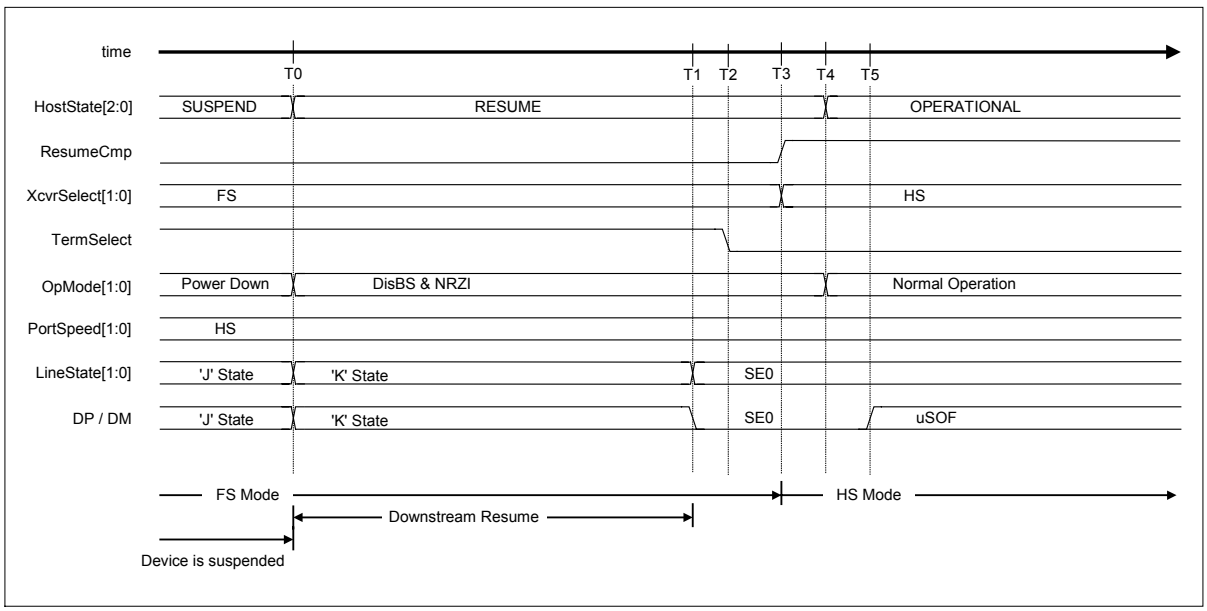



**Fig. 6.58 Resume timing (HS mode)**

**Table 6.53 Resume Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoSUSPEND (by firmware).<br>The disconnection detection and remote wakeup detection functions are turned off.<br>Operation mode is set to Disable BS and NRZI and the hardware starts issuing a resume “K” signal. | 0 (reference) |
| T1 | The hardware finishes issuing the resume “K” signal.<br>Terminal selection is set to HS (by hardware). | T0 + 20ms{$T_{DRSMDN}$}< T1 |
| T2 | Transceiver selection is set to HS (by hardware). | T1 + 100us < T2 < T1 + 2.0us |
| T3 | A resume complete status (SIE_IntStat_1.ResumeCmp) is issued (by hardware). | T1 + 90us < T3 < T1 + 110us |
| T4 (reference) | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware)<br>Operation mode is set to NormalOperation (by hardware). | T4 |
| T5 (reference) | First micro SOF is issued (by hardware). | T5 < T1 + 3ms<br>T4 + 120us < T5 < T4 + 130us |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.7.2 When FS Device is Connected

The procedure that is executed when an FS device is connected is described below. Processes (2) to (6) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESUME (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESUME (T0).

(3) The disconnection detection and remote wakeup detection functions are turned off (T0).

(4) Operation mode (H_XcvrControl.OpMode) is set to Disable BS and NRZI, and the hardware thereby starts issuing a resume "K" signal (T0).

(5) The hardware finishes issuing the resume "K" signal (T1), with EOP in LS bit time appended at the end.

(6) A resume complete status (SIE_IntStat_1.ResumeCmp) is issued to the firmware (T2).

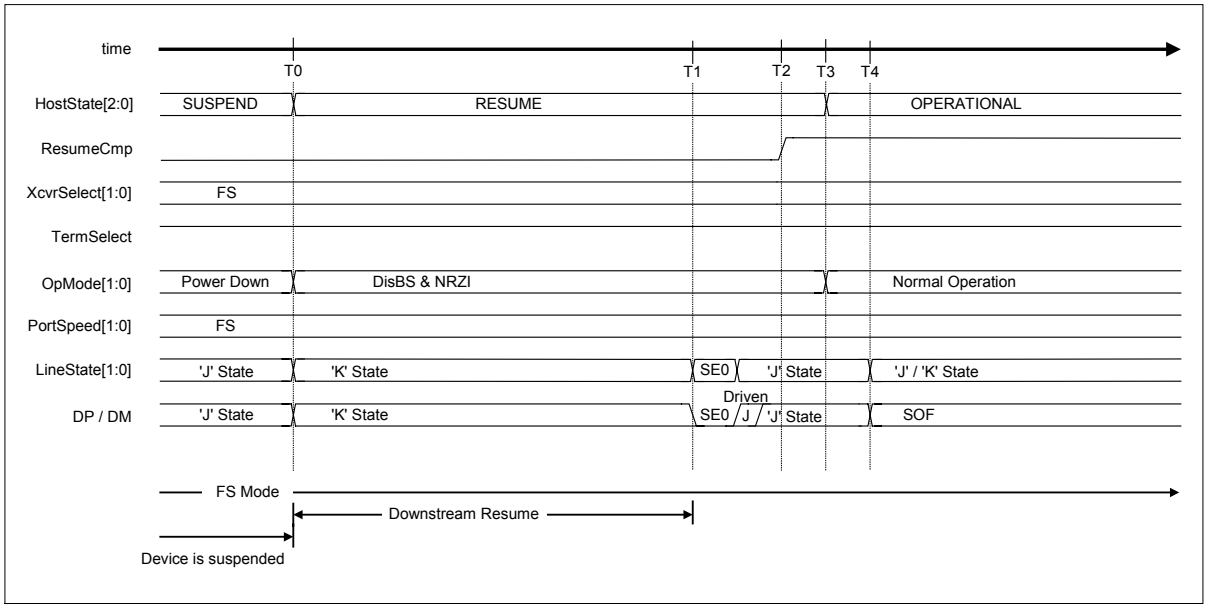



**Fig. 6.59 Resume timing (FS mode)**

**Table 6.54 Resume Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESUME (by firmware).<br>The disconnection detection and remote wakeup detection functions are turned off.<br>Operation mode is set to Disable BS and NRZI, and the hardware starts issuing a resume “K” signal. | 0 (reference) |
| T1 | The hardware finishes issuing the resume “K” signal, with EOP in LS bit time appended at the end. | T0 + 20ms {TDRSMDN} < T1 |
| T2 | A resume complete status (SIE_IntStat_1.ResumeCmp) is issued (by hardware). | T1 + 90us < T2 < T1 + 110us |
| T3 (reference) | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware)<br>Operation mode is set to NormalOperation (by hardware). | T3 |
| T4 (reference) | First SOF is issued (by hardware). | T4 < T1 + 3ms<br>T3 + 0.9ms < T4 < T3 + 1.1ms |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.7.3 When LS Device is Connected

The procedure that is executed when an LS device is connected is described below. Processes (2) to (6) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESUME (T0).

(2) The host state monitor (H_NegoControl_0.HostState) is set to RESUME (T0).

(3) The disconnection detection and remote wakeup detection functions are turned off (T0).

(4) Operation mode (H_XcvrControl.OpMode) is set to Disable BS and NRZI, and the hardware thereby starts issuing a resume "K" signal (T0).

(5) The hardware finishes issuing the resume "K" signal (T1), with EOP in LS bit time appended at the end.

(6) A resume complete status (SIE_IntStat_1.ResumeCmp) is issued to the firmware (T2).

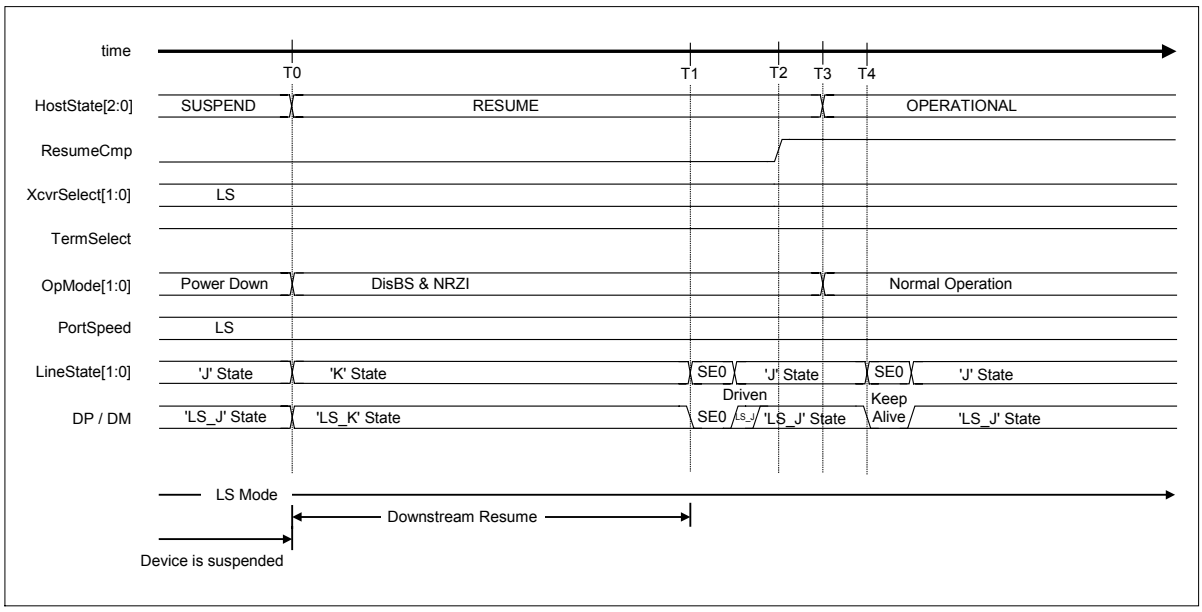



**Fig. 6.60 Resume timing (LS mode)**

**Table 6.55 Resume Timing Values (FS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESUME (by firmware).<br>The disconnection detection and remote wakeup detection functions are turned off.<br>Operation mode is set to Disable BS and NRZI, and the hardware starts issuing a resume “K” signal. | 0 (reference) |
| T1 | The hardware finishes issuing the resume “K” signal, with EOP in LS bit time appended at the end. | T0 + 20ms {TDRSMDN} < T1 |
| T2 | A resume complete status (SIE_IntStat_1.ResumeCmp) is issued (by hardware). | T1 + 90us < T2 < T1 + 110us |
| T3 (reference) | H_NegoControl_0.AutoMode is set to GoOPERATIONAL (by firmware)<br>Operation mode is set to NormalOperation (by hardware). | T3 |
| T4 (reference) | First KeepAlive is issued (by hardware). | T4 < T1 + 3ms<br>T3 + 0.9ms < T4 < T3 + 1.1ms |

Note: Names stipulated in the USB2.0 Standard are shown in { }.

#### 6.3.9.3.8 GoWAIT_CONNECTtoDIS

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoWAIT_CONNECTtoDIS, the processes required for a transition from WAIT_CONNECT state to DISABLED state is automatically executed by the LSI's hardware.

The execution procedure is described below. Processes (2) to (5) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoWAIT_CONNECTtoDIS (T0).

(2) Process equivalent to GoWAIT_CONNECT is executed (T0).

(3) A connection detection is performed and a connection detected status (H_SIE_IntStat_0.DetectCon) is issued (T1).

(4) Process equivalent to GoDISABLED is executed (T1).

(5) A disabled complete status (H_SIE_IntStat_1.DisabledCmp) is issued (T2).

Note that the timing in each state here is the same as when GoWAIT_CONNECT and GoDISABLED are executed. For details about the timing, refer to the relevant sections on GoWAIT_CONNECT and GoDISABLED.

For the execution procedure and timing in cases when an error (disconnection or VBUS error) is detected during the process, refer to the relevant sections on disconnection detection and VBUS error.

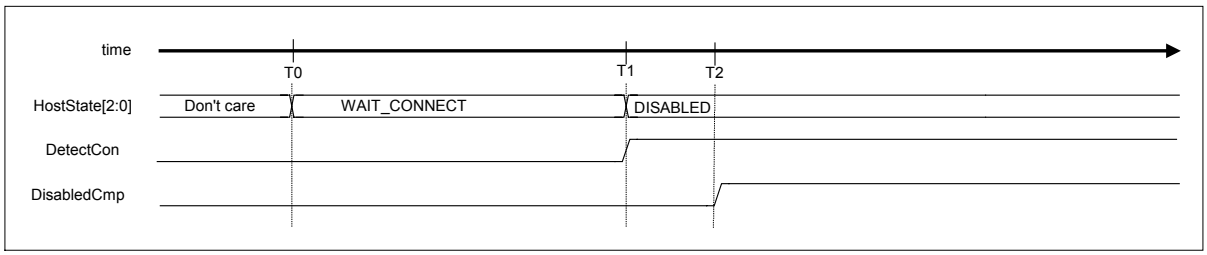



**Fig. 6.61 GoWAIT_CONNECTtoDIS timing (HS mode)**

**Table 6.56 GoWAIT_CONNECTtoDIS Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoWAIT_CONNECTtoDIS (by firmware). Process equivalent to GoWAIT_CONNECT is executed (by hardware). | 0 (reference) |
| T1 | A connection detection is performed and a connection detected status is issued. Process equivalent to GoDISABLED is executed (by hardware). | T1 |
| T2 | A disabled complete status is issued (by hardware). | T2 |

#### 6.3.9.3.9 GoWAIT_CONNECTtoOP

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoWAIT_CONNECTtoOP, the processes required for a transition from WAIT_CONNECT state to OPERATIONAL state are automatically executed by the LSI's hardware.

##### 6.3.9.3.9.1 When HS Device is Connected

The procedure that is executed when an HS device is connected is described below. Processes (2) to (9) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoWAIT_CONNECTtoOP (T0).

(2) A process equivalent to GoWAIT_CONNECT is executed (T0).

(3) A connection detection is performed and a connection detected status (H_SIE_IntStat_0.DetectCon) is issued (T1).

(4) Process equivalent to GoDISABLED is executed (T1).

(5) A disabled complete status (H_SIE_IntStat_1.DisabledCmp) is issued (T2).

(6) Process equivalent to GoRESET is executed (T2).

(7) A device Chirp is detected, and a device chirp normal detection status (H_SIE_IntStat_0.DetectDevChirpOK) is issued (T3).

(8) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T4).

(9) Process equivalent to GoOPERATIONAL is executed (T4).

Note that the timing in each state here is the same as when GoWAIT_CONNECT, GoDISABLED, GoRESET, and GoOPERATIONAL are executed. For details about the timing, refer to the relevant sections on GoWAIT_CONNECT, GoDISABLED, GoRESET, and GoOPERATIONAL.

For the execution procedure and timing in cases when an error (disconnection, VBUS error, or device chirp error) is detected during the process, refer to the relevant sections on disconnection detection, VBUS error, and GoRESET.

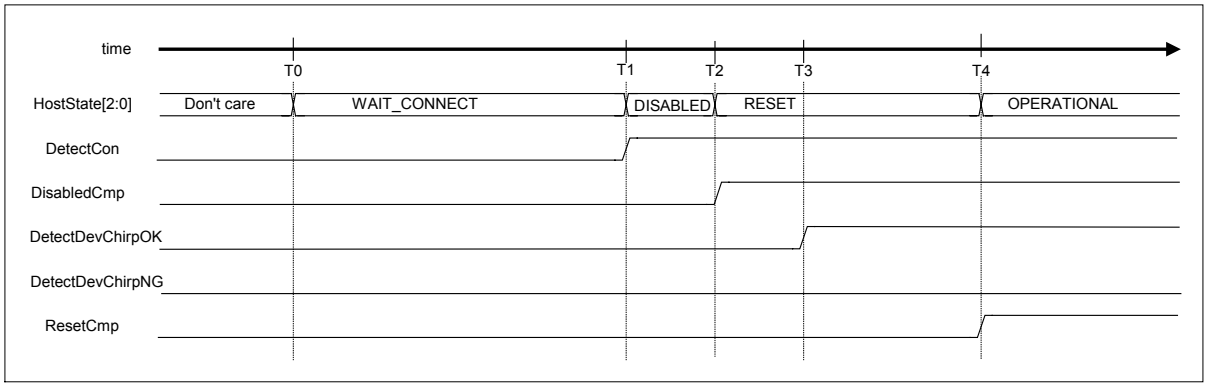



**Fig. 6.62 GoWAIT_CONNECTtoOP timing (HS mode)**

**Table 6.57 GoWAIT_CONNECTtoOP Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoWAIT_CONNECTtoOP (by firmware).<br>A process equivalent to GoWAIT_CONNECT is executed (by hardware). | 0 (reference) |
| T1 | A connection detection is performed and a connection detected status is issued.<br>A process equivalent to GoDISABLED is executed (by hardware). | T1 |
| T2 | A disabled complete status is issued.<br>A process equivalent to GoRESET is executed (by hardware). | T2 |
| T3 | A device Chirp is detected, and a device chirp normal detection status is issued (by hardware). | T3 |
| T4 | A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued.<br>A process equivalent to GoOPERATIONAL is executed (by hardware). | T4 |

#### 6.3.9.3.9.2 When FS or LS Device is Connected

The procedure that is executed when an FS or LS device is connected is described below. Processes (2) to (9) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoWAIT_CONNECTtoOP (T0).

(2) A process equivalent to GoWAIT_CONNECT is executed (T0).

(3) A connection detection is performed and a connection detected status (H_SIE_IntStat_0.DetectCon) is issued (T1).

(4) A process equivalent to GoDISABLED is executed (T1).

(5) A disabled complete status (H_SIE_IntStat_1.DisabledCmp) is issued (T2).

(6) A process equivalent to GoRESET is executed (T1).

(7) Because a device Chirp is not detected, a device chirp normal/abnormal detection (H_SIE_IntStat_0.DetectDevChirpOK/NG) is not issued (T3).

(8) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T4).

(9) A process equivalent to GoOPERATIONAL is executed (T4).

Note that the timing in each state here is the same as when GoWAIT_CONNECT, GoDISABLED, GoRESET, and GoOPERATIONAL are executed. For details about the timing, refer to the relevant sections on GoWAIT_CONNECT, GoDISABLED, GoRESET, and GoOPERATIONAL.

For the execution procedure and timing in cases when an error (disconnection or VBUS error) is detected during the process, refer to the relevant sections on disconnection detection and VBUS error.

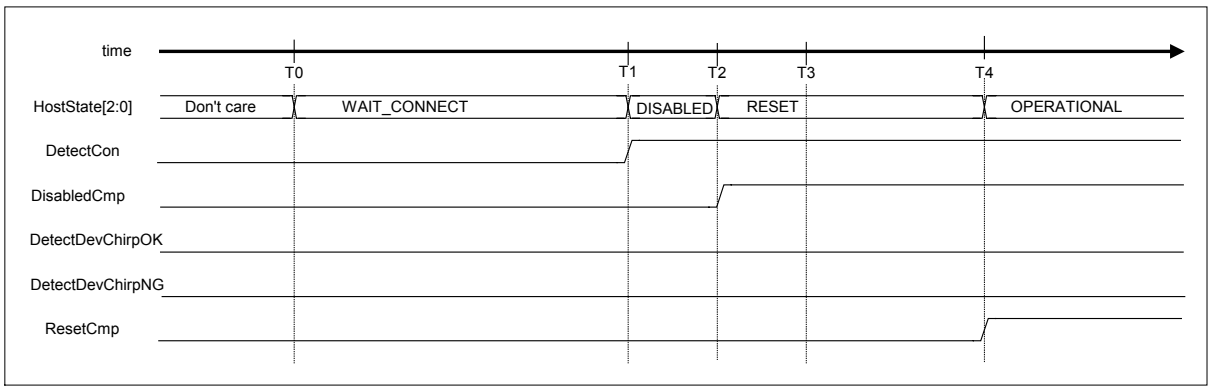



**Fig. 6.63 GoWAIT_CONNECTtoOP timing (FS or LS mode)**

**Table 6.58 GoWAIT_CONNECTtoOP Timing Values (FS or LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoWAIT_CONNECTtoOP (by firmware).<br>A process equivalent to GoWAIT_CONNECT is executed (by hardware). | 0 (reference) |
| T1 | A connection detection is performed and a connection detected status is issued.<br>A process equivalent to GoDISABLED is executed (by hardware). | T1 |
| T2 | A disabled complete status is issued.<br>A process equivalent to GoRESET is executed (by hardware). | T2 |
| T3 | Because a device Chirp is not detected, a device chirp normal/abnormal detection is not issued (by hardware). | T3 |
| T4 | A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued.<br>A process equivalent to GoOPERATIONAL is executed (by hardware). | T4 |

#### 6.3.9.3.10 GoRESETtoOP

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESETtoOP, the processes required for a transition from RESET state to OPERATIONAL state are automatically executed by the LSI's hardware.

##### 6.3.9.3.10.1 When HS Device is Connected

The procedure that is executed when an HS device is connected is described below. Processes (2) to (5) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESETtoOP (T0).

(2) A process equivalent to GoRESET is executed (T0).

(3) A device Chirp is detected, and a device chirp normal detection status (H_SIE_IntStat_0.DetectDevChirpOK) is issued (T1).

(4) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T2).

(5) A process equivalent to GoOPERATIONAL is executed (T2).

Note that the timing in each state here is the same as when GoRESET and GoOPERATIONAL are executed. For details about the timing, refer to the relevant sections on GoRESET and GoOPERATIONAL.

For the execution procedure and timing in cases when an error (VBUS error or device chirp error) is detected during the process, refer to the relevant sections on VBUS error and GoRESET.

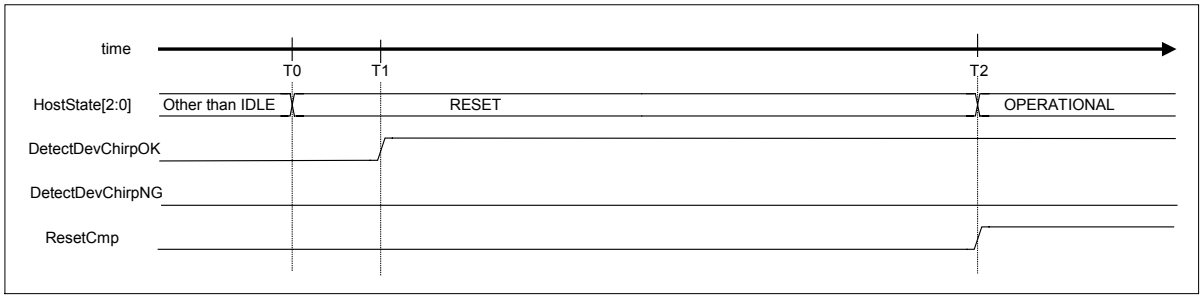



**Fig. 6.64 GoRESETtoOP timing (HS mode)**

**Table 6.59 GoRESETtoOP Timing Values (HS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESETtoOP (by firmware).<br>A process equivalent to GoRESET is executed (by hardware). | 0 (reference) |
| T1 | A device Chirp is detected, and a device chirp normal detection status is issued (by hardware). | T1 |
| T2 | A reset complete status is issued.<br>A process equivalent to GoOPERATIONAL is executed (by hardware). | T2 |

#### 6.3.9.3.10.2 When FS or LS Device is Connected

The procedure that is executed when an FS or LS device is connected is described below. Processes (2) to (5) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESETtoOP (T0).

(2) A process equivalent to GoRESET is executed (T0).

(3) Because a device Chirp is not detected, a device chirp normal/abnormal detection (H_SIE_IntStat_0.DetectDevChirpOK) is not issued (T1).

(4) A reset complete status (H_SIE_IntStat_1.ResetCmp) is issued (T2).

(5) A process equivalent to GoOPERATIONAL is executed (T2).

Note that the timing in each state here is the same as when GoRESET and GoOPERATIONAL are executed. For details about the timing, refer to the relevant sections on GoRESET and GoOPERATIONAL.

For the execution procedure and timing in cases when an error (VBUS error) is detected during the process, refer to the relevant section on VBUS error.

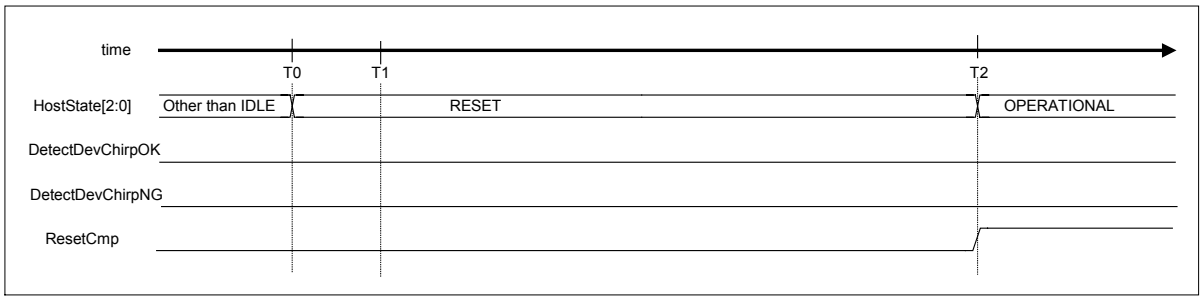



**Fig. 6.65 GoRESETtoOP timing (FS or LS mode)**

**Table 6.60 GoRESETtoOP Timing Values (FS or LS Mode)**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESETtoOP (by firmware).<br>A process equivalent to GoRESET is executed (by hardware). | 0 (reference) |
| T1 | Because a device Chirp is not detected, a device chirp normal/abnormal detection is not issued (by hardware). | T1 |
| T2 | A reset complete status is issued.<br>A process equivalent to GoOPERATIONAL is executed (by hardware). | T2 |

#### 6.3.9.3.11 GoSUSPENDtoOP

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoSUSPENDtoOP, the processes required for a transition from SUSPEND state to OPERATIONAL state is automatically executed by the LSI's hardware.

When GoSUSPENDtoOP is set, the remote wakeup detection function is automatically turned on/off. Note, however, that since this on/off is not reflected in the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb), the firmware is not required to manipulate the remote wakeup acceptance enable (H_NegoControl_1.RmtWkupDetEnb).

When this setting is used, do not use the power management function.

The procedure that is executed for this setting is described below. Processes (2) to (7) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoSUSPENDtoOP (T0).

(2) A process equivalent to GoSUSPEND is executed (T0).

(3) A transition-to-suspend complete status (H_SIE_IntStat_1.SuspendCmp) is issued (T1).

(4) A remote wakeup is detected, and a remote wakeup detected status (H_SIE_IntStat_0.DetectRmtWkup) is issued (T2).

(5) A process equivalent to GoRESUME is executed (T2).

(6) A resume complete status (H_SIE_IntStat_1.ResumeCmp) is issued (T3).

(7) A process equivalent to GoOPERATIONAL is executed (T3).

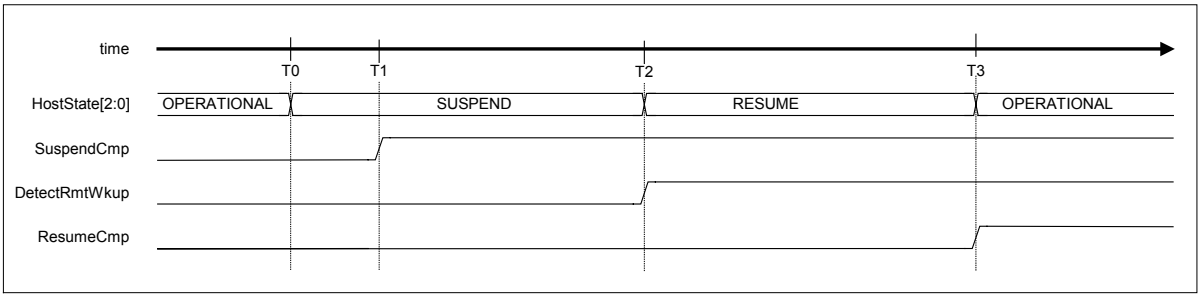



**Fig. 6.66 GoSUSPENDtoOP timing**

**Table 6.61 GoSUSPENDtoOP Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoSUSPENDtoOP (by firmware).<br>A process equivalent to GoSUSPEND is executed (by hardware). | 0 (reference) |
| T1 | A transition-to-suspend complete status is issued (by hardware). | T1 |
| T2 | A remote wakeup is detected, and a remote wakeup detected status is issued. A process equivalent to GoRESUME is executed (by hardware). | T2 |
| T3 | A resume complete status is issued. A process equivalent to GoOPERATIONAL is executed (by hardware). | T3 |

#### 6.3.9.3.12 GoRESUMEtoOP

When the host state transition execution (H_NegoControl_0.AutoMode) is set to GoRESUMEtoOP, the processes required for a transition from RESUME state to OPERATIONAL state is automatically executed by the LSI's hardware.

The procedure that is executed for this setting is described below. Processes (2) to (4) below are automatically executed by the LSI's hardware.

(1) The firmware sets the host state transition execution (H_NegoControl_0.AutoMode) to GoRESUMEtoOP (T0).

(2) A process equivalent to GoRESUME is executed (T0).

(3) A resume complete status (H_SIE_IntStat_1.ResumeCmp) is issued (T1).

(4) A process equivalent to GoOPERATIONAL is executed (T1).

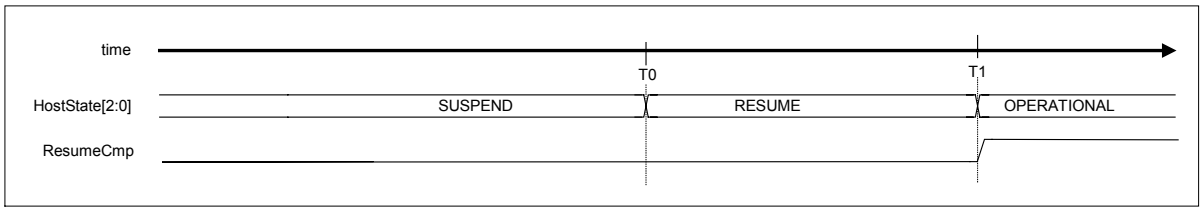



**Fig. 6.67 GoRESUMEtoOP timing**

**Table 6.62 GoRESUMEtoOP Timing Values**

| Timing Parameter | Description | Value |
|---|---|---|
| T0 | H_NegoControl_0.AutoMode is set to GoRESUMEtoOP (by firmware).<br>A process equivalent to GoRESUME is executed (by hardware). | 0 (reference) |
| T1 | A resume complete status is issued.<br>A process equivalent to GoOPERATIONAL is executed (by hardware). | T1 |

## 6.4 Power Management Function

The power management function controls the operation of the oscillator and the PLL, manipulating a transition between two states: Sleep and Active. Note that the Snooze state serves only as a transition state through which transitions between Sleep and Active states occur. To enter other states, set the GoSLEEP or GoACTIVE bit of the PM_Control register. This will initiate a state transition. This state transition completes after the given processes are performed. To confirm the LSI's current state, check the PM_Control.PM_State [1:0]. A MainIntStat.FinishedPM event is generated after a transition is complete. At this time, if the MainIntEnb.EnFinishedPM bit has been set, an XINT interrupt is generated. After the GoSLEEP or GoACTIVE bit of the PM_Control register is set, note that PM_Control.PM_State [1:0] will not indicate the precise state until a MainIntStat.FinishedPM event occurs.

If the PM_Control.GoSLEEP bit is set during the Active state, the LSI shifts to the Sleep state via the Snooze state. When the state transition is fully complete, a MainIntStat.FinishedPM event is generated. Furthermore, if the PM_Control.GoActive bit is set during the sleep state, the LSI shifts to the Active state via the Snooze state. When the state transition is fully complete, a MainIntStat.FinishedPM event is generated.

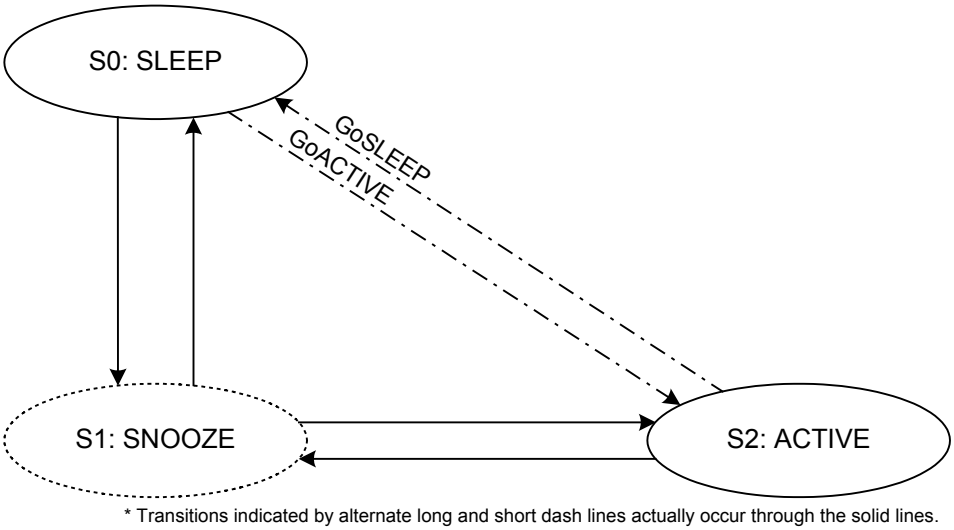



**Fig. 6.68 Power management**

### 6.4.1 SLEEP (Sleep)

In this state, CLK input from the CLKIN pin is gated, or the oscillator does not oscillate. In this state, therefore, the PLL are not oscillating either. The registers and bits in ***bold italic*** type can be read and written even in Sleep state. No other registers can be read and written unless in Active state, in which case all register bits are read as 0 when read-accessed.

To initiate Sleep by setting the PM_Control.GoSLEEP bit during the Active state, shut down the operating internal PLL to turn off OSCCLK output to the internal circuit, then shut down the oscillator.

Conversely, if the LSI shifts to the Snooze state after leaving Sleep by setting the PM_Control.GoACTIVE bit during 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's oscillation is stablized. Since the oscillation stabilization time varies with the oscillator cell, resonator, peripheral circuits, and the board involved, use the WakeUpTim_H, L registers to set it.

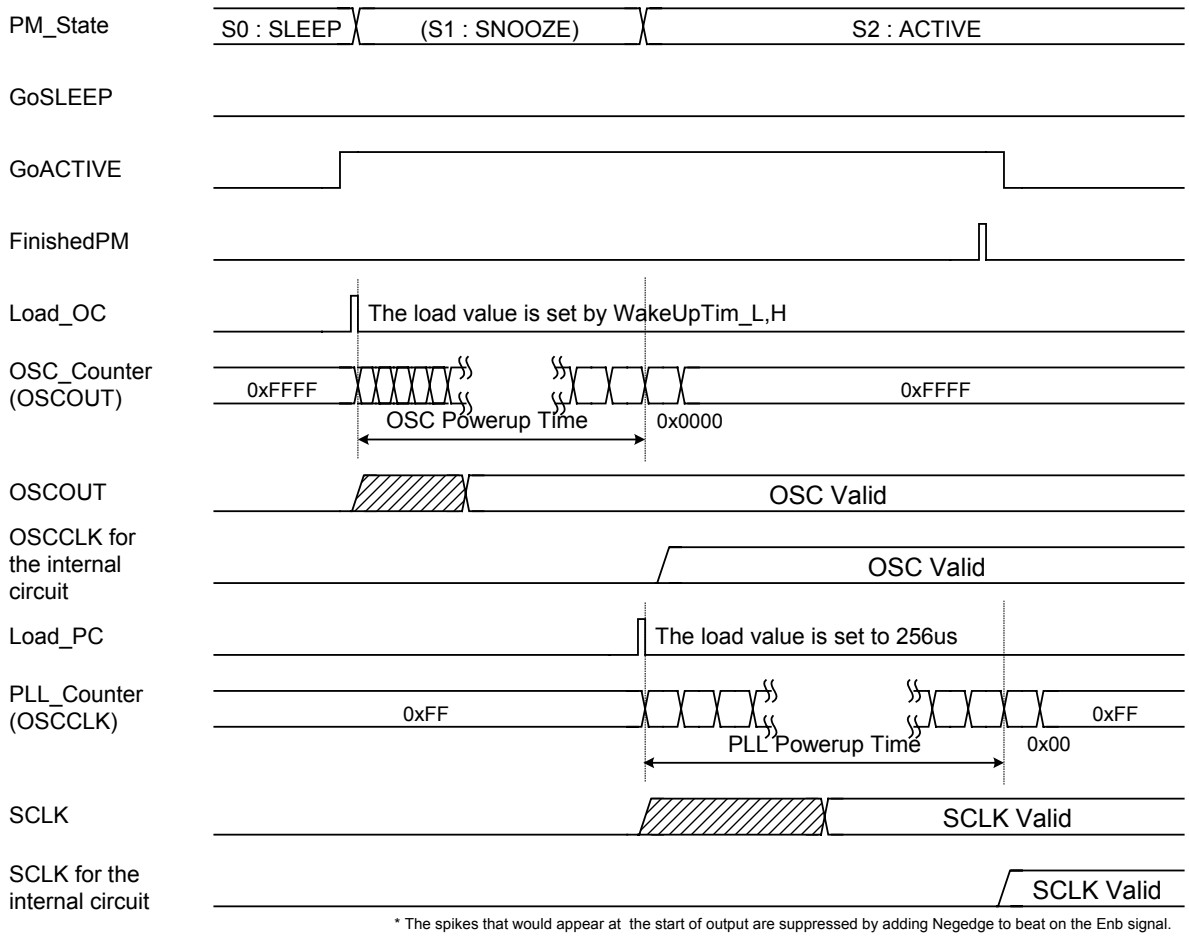



**Fig. 6.69 Leaving SLEEP state (during GoACTIVE)**

### 6.4.2 SNOOZE (Snooze)

This is the state in which the oscillator is oscillating but the PLL are not. Note that the Snooze state serves only as a transition state through which transitions between Sleep and Active states occur.

If the LSI shifts to the Active state after leaving Snooze, SCLK is gated for a PLL stabilization time (approx. 250 us) to ensure that it will not be supplied to the internal circuit until after the PLL's oscillation is stablized.

### 6.4.3 ACTIVE (Active)

This is the state in which the oscillator and PLL are operating. All types of data transfers are permitted, including USB transfers.

### 6.4.4 CPU_Cut Mode

Input of all CPU interface signals except XCS is shut off from the initial IC stage on by setting the PM_State.GoCPU_Cut bit to Sleep state. In this mode, the LSI's input pins do not switch on or off even if the CPU interface is in the Sleep state. This reduces chip power consumption to a level nearly equal to that when shut down (IQ state).

To resume from CPU_Cut mode, the LSI requires a dummy read of the PM_Control register (data = 0x0). The LSI is then restored from CPU_Cut mode at the same time the read operation finishes. Since this restore operation is achieved by a high-going edge of the XCS signal, always make sure the XCS signal is negated (returned high) after a dummy read. If the XCS signal is not negated and another register is read immediately after this dummy read, CPU_Cut mode will persist.

## 6.5 FIFO Management

This section describes FIFO management.

### 6.5.1 FIFO Memory Map

The following shows the FIFO memory map.

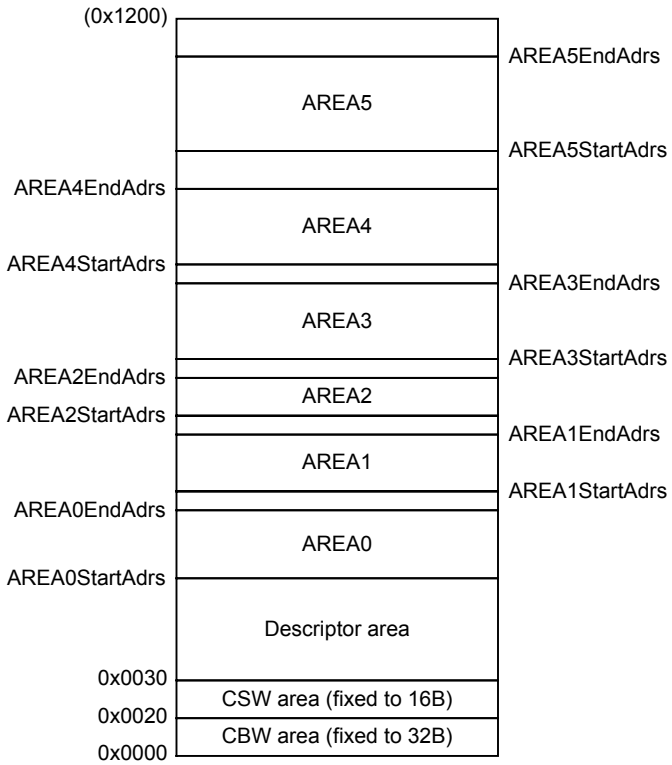



**Fig. 6.70 Device FIFO memory map**

The FIFO memory can be divided for use into nine areas— CBW area, CSW area, descriptor area, AREA0, AREA1, AREA2, AREA3, AREA4, and AREA5. Of these areas, CBW, and CSW areas have a fixed amount of storage allocated to each, as shown in Fig. 6.70. The other AREAn{n=0-5} can have any amount of storage allocated by setting the FIFO area setup registers (AREAn{n=0-5}StartAdrs_H,L and AREAn{n=0-5}EndAdrs_H,L) as desired. Any unused memory space can be used as the descriptor area.

The descriptor area is provided for use by the descriptor reply function during USB device mode. Any unused FIFO memory space may be used. The actual method for using this area is described later in Section 6.5.. Although any FIFO area can be set to be usable by the descriptor reply function, to avoid contention, we recommend allocating the descriptor area within the address space shown here.

The CBW area is used for CBW support of the Bulk-only Support function during USB device mode. This area has 32 bytes of storage reserved for it, of which 31 bytes of storage beginning with the address 0x0000 is used. The actual method for using this area is described later in Section 6.5.3.1. This

CBW area is used by the Bulk-only Support function for CHa during USB host mode. The method for using this area is described later in Section 6.5.3.2.

The CSW area is used for CSW support of the Bulk-only Support function during USB device mode. This area has 16 bytes of storage reserved for it, of which 13 bytes of storage beginning with the address 0x0020 is used. The actual method for using this area is described later in Section 6.5.4.1. This CSW area is used by the Bulk-only Support function for CHa during USB host mode. The method for using this area is described later in Section 6.5.4.2.

AREA0, AREA1, AREA2, AREA3, AREA4, and AREA5 are the general-purpose endpoint areas that can be made usable when endpoints EPx{x=0,a-e} are joined during USB device mode. Similarly, in USB host mode, they can be made usable when channels CHx{x=0,a-e} are joined. To join any area and an endpoint or channel, set the JoinEPxCHx{x=0,a-e} bit of the AREA join setting register (AREAn{n=0-5}Join_1). For the various possible combinations in which the endpoints and channels can be joined to the FIFO areas, refer to Appendix E.

The AREA0, AREA1, AREA2, AREA3, AREA4, and AREA5 areas, respectively, are controlled as FIFO, so that the number of data bytes stored in each is retained. To clear this retained status, set the AREAnFIFO_Clr.ClearAREAn{n=0-5} bits.

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, so that the information recorded in the descriptor area will never be lost and there is no need to write data back again after clearing the status.

### 6.5.2 Descriptor Area

The descriptor area is provided for use by the descriptor reply function during USB device mode. 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 for 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 an request is accepted. That way, requests can be responded promptly because there is no need to write data into the EP0 area for each request.

#### 6.5.2.1 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 in the RAM_WrDoor_H,L registers. The RAM_WrArs_H,L register values are updated every write data bytes each time the registers are written to, so that 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.

#### 6.5.2.2 Executing a Data Stage (IN) in the Descriptor Area

To use the 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, and then set the D_EP0Control.ReplyDescriptor bit to 1. Furthermore, set the D_EP0Control.INxOUT bit to 1 to permit an IN transaction to be performed. To ensure that data packets will be sent back to an IN transaction of the data stage, make sure the D_SETUP_Control.ProtectEP0 is cleared before the D_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 D_EP0MaxSize) until the number of data bytes set in the D_DescSize_H,L registers is reached. If the D_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 D_EP0Control.ReplyDescriptor bit is cleared and the D_EP0_IntStat.DescriptorCmp is set. The firmware should go to processing of the status stage.

### 6.5.3 CBW Area

#### 6.5.3.1 CBW Area (during USB Device Mode)

The CBW area is used for CBW support of the Bulk-only Support function during USB device mode. When a command transport of Bulk Only Transport Protocol is to be performed at the Bulk Only endpoint (endpoint EPa, EPb, EPc or EPe), data can be received in this area. This enables only the data element to be received by the endpoint FIFO.

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 more than 31 bytes in size, an error status is issued and the data is discarded.

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

#### 6.5.3.2 CBW Area (during USB Host Mode)

The CBW area is provided for use by the Bulk-only Support function in USB host mode. When a command transport of the Bulk Only Transport Protocol is performed on channel CHa, the CBW data is transmitted from this area as data packets.

The CBW area must have 31 bytes of CBW data prepared before data packets can be transmitted, starting with address 0x0000.

Use the RAM_WrDoor function to write data to the CBW area. Write the start address of the CBW area (0x0000) in the RAM_WrAdr_H,L registers, then write 31 bytes of valid data via the RAM_WrDoor_0,1 registers. Since 32 bytes of space is reserved for the CBW area, 32 bytes of data can be written to the CBW area without problems by accessing it wordwise.

### 6.5.4 CSW Area

#### 6.5.4.1 CSW Area (during USB Device Mode)

The CSW area is used for CSW support of the Bulk-only Support function during USB device mode. When a command transport of Bulk Only Transport Protocol is to be performed at the Bulk IN endpoint (endpoint Epa, EPb, EPc, EPd, or EPe), data can be transmitted from this area. This enables only the data element to be received by the endpoint FIFO.

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

To write data into the CSW area, use the RAM_WrDoor function, write the start address of the CSW area (0x0020) in the RAM_WrAdr_H,L registers and write 13 bytes of valid data via the RAM_WrDoor_H,L registers. Since the CSW area has a 16 bytes of storage reserved for it, there is no possibility of affecting other areas even when 14 bytes of data are written into this area wordwise.

#### 6.5.4.2 CSW Area (during USB Host Mode)

The CSW area is provided for use by the Bulk-only Support function in USB host mode. When a status transport for the Bulk Only Transport Protocol is performed on channel CHa, the CSW data is received in this area. This enables only the data element to be received by the endpoint FIFO.

Use the RAM_Rd function to read out data received in the CSW area. Set the RAM_RdControl.RAM_GoRdCBW_CSW bit. The data in the CSW area will be read out and copied to the RAM_Rd_00 through the RAM_Rd_0C registers. A complete status (CPU_IntStat.RAM_RdCmp bit) is issued when the operation is finished.

### 6.5.5 Method for Accessing the FIFO

There are several methods to access the FIFO, including the CPU (registers), CPU (DMA), IDE, and USB.

#### 6.5.5.1 Method for Accessing the FIFO (RAM_Rd)

To access the FIFO for read 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 of the specified FIFO area is ready for read from the RAM_Rd register, the CPU_IntStat.RAM_RdCmp bit is set to 1. After confirming the RAM_RdCmp bit, read data from the RAM_Rd_00 through RAM_Rd_1F registers. If the data size set in the RAM_RdCount register is smaller than 32 bytes, the data bytes in the RAM_Rd registers exceeding the set size are ignored.

FIFO data read via the RAM_Rd registers can be performed regardless of the settings for FIFO area of a relevant channel.

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

#### 6.5.5.2 Method for Accessing the FIFO (RAM_WrDoor)

To access the FIFO for write via the RAM_WrDoor_0,1 registers of the CPUIF, set the write start address in the RAM_WrAdr_H,L registers and write data via the RAM_WrDoor_0,1 registers. The RAM_WrAdr_H,L registers are automatically incremented by an amount equal to written bytes each time the FIFO is accessed for write, so that if data needs to be written to contiguous addresses, the data can be written to the RAM_WrDoor_0,1 registers successively.

Write to the FIFO via the RAM_WrDoor_0,1 registers can be performed regardless of the settings for FIFO area of a relevant channel.

#### 6.5.5.3 Method for Accessing the FIFO (Register Access)

To access the FIFO for read via a register access of the CPU, set AREAn{n=0-5}Join_0.JoinCPU_Rd for any one of the channels to 1 and read data via the FIFO_Rd_0,1 or FIFO_ByteRd register.

To access the FIFO for write via a register access of the CPU, set AREAn{n=0-5}Join_0.JoinCPU_Wr for any one of the channels to 1 and write data in the FIFO_Wr_0,1 registers.

The FIFO_RdRemain_H,L registers indicate the remaining number of data bytes that can be read from the FIFO for only one channel that is set by JoinCPU_Rd. Similarly, the FIFO_WrRemain_H,L registers indicate the remaining number of bytes in the FIFO area to which data can be written for only one channel that is set by JoinCPU_Wr.

Be aware that in cases where registers are to be dumped when debugging the firmware using an ICE or other tool, if any JoinCPU_Rd register is set, data may inadvertently be read from the FIFO when registers are dumped.

#### 6.5.5.4 Method for Accessing the FIFO (DMA)

To access the FIFO for read via a DMA access of the CPU, select only one endpoint using the AREAn{n=0-5}Join_0.JoinDMA bit and set the DMA_Control.Dir bit to 1, and then execute a DMA procedure to read data.

To access the FIFO for write via a DMA access of the CPU, select only one endpoint on either DMA channel using the AREAn{n=0-5}Join_0.JoinDMA bit and set the DMA_Control.Dir bit to 0, and then execute a DMA procedure to write data.

The DMA_Remain_H,L registers indicate the remaining number of data bytes that can be read from the FIFO for only one endpoint on either DMA channel that is selected by the AREAn{n=0-5}Join_0.JoinDMA bit. Similarly, they indicate the remaining number of bytes in the

FIFO area to which data can be written for only one endpoint on either DMA channel that is selected by the AREAn{n=0-5}Join_0.JoinDMA bit.

#### 6.5.5.5 Limitations on FIFO Access

The FIFO of the LSI stipulated herein is designed so that transmission/reception with the USB and register or DMA read/write from the CPU bus are performed simultaneously. Furthermore, read from the CPU bus is accomplished by look-ahead processing.

For the above reasons, the setup method (Join) for access to the FIFO on respective channels is subject to the following exclusion rules:

- Only one of JoinCPU_Wr, JoinDMA, or JoinEPxCHx{x=0,a-e} can be set for one area as the FIFO write source.
- Only one of JoinCPU_Rd, JoinDMA, or JoinEPxCHx{x=0,a-e} can be set for one area as the FIFO read source.
- Only one of JoinCPU_Wr, JoinCPU_Rd, or JoinDMA can be set for one area at any given time.
- Only one of JoinCPU_Wr or JoinCPU_Rd can be set at any given time.

Furthermore, the following prohibitions apply for FIFO access from the USB:

- The FIFO area being written to from the USB cannot be accessed for write from other sources.
- The FIFO area being read into the USB cannot be accessed for read from other sources.

For example, although it is possible to write to an FIFO area with an OUT endpoint joined during USB device mode after setting JoinCPU_Wr, there must be no OUT transactions being performed when the FIFO is written to from the CPU. Similarly, although it is possible to read from the FIFO area with an IN endpoint joined after setting JoinCPU_Rd, there must be no IN transactions being performed when the FIFO is read from the CPU. The situation where no transactions are being performed can be confirmed by the fact that the ActiveUSB bit is cleared, Endpoint concerned is not joined to any FIFO area, or ForceNAK is set.

## 6.6 CPUIF

### 6.6.1 Mode Switching

The CPUIF of the S1R72V17 accommodates asynchronous CPUs, and has the following three operation modes.

**Table 6.63 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 CPU_Config register. The value of the CPU_Config register can be protected against erroneous writes by setting the ModeProtect register.

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

In addition, the CPUIF of the S1R72V17 has a bus swap function. To use this function, set the CPU_Config.CPU_Endian bit when initially setting up the CPU_Config register. Furthermore, it is possible to set the XINT logic level and output mode, the logic levels of XDREQ and XDACK, and the CS_Mode of DMA during initial setting of the CPU_Config register.

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

### 6.6.2. Notes on Mode Switchover

The S1R72V17 CPU bus operation mode can be set to one suitable for the CPU used by setting the CPU_Config register. In its initial state, the LSI operates in 16-bit Strobe mode. If the operation mode needs to be changed to 16-bit BE mode or 8-bit mode, observe the precautions described below.

#### 6.6.2.1. When Using 16-bit BE Mode

If the CPU is to be used in 16-bit BE mode, first set the CPU_Config register as described in Section 6.6.1. When changing operation modes, be sure to write data in bytes to address 075h as shown in Fig. 6.71. Since the S1R72V17 is operating at its initial state of 16-bit Strobe mode, if the chip select and byte mask high signals of the CPU (XCS and XWRH) have a skew like the one shown below, the LSI may mistake it for a valid write period and proceed internally on this assumption. Although the S1R72V17 incorporates a filter circuit (min: 1ns) to eliminate such skews, be sure to confirm the AC characteristics of the CPU used and process the board, etc. as necessary to prevent such skews from occurring.

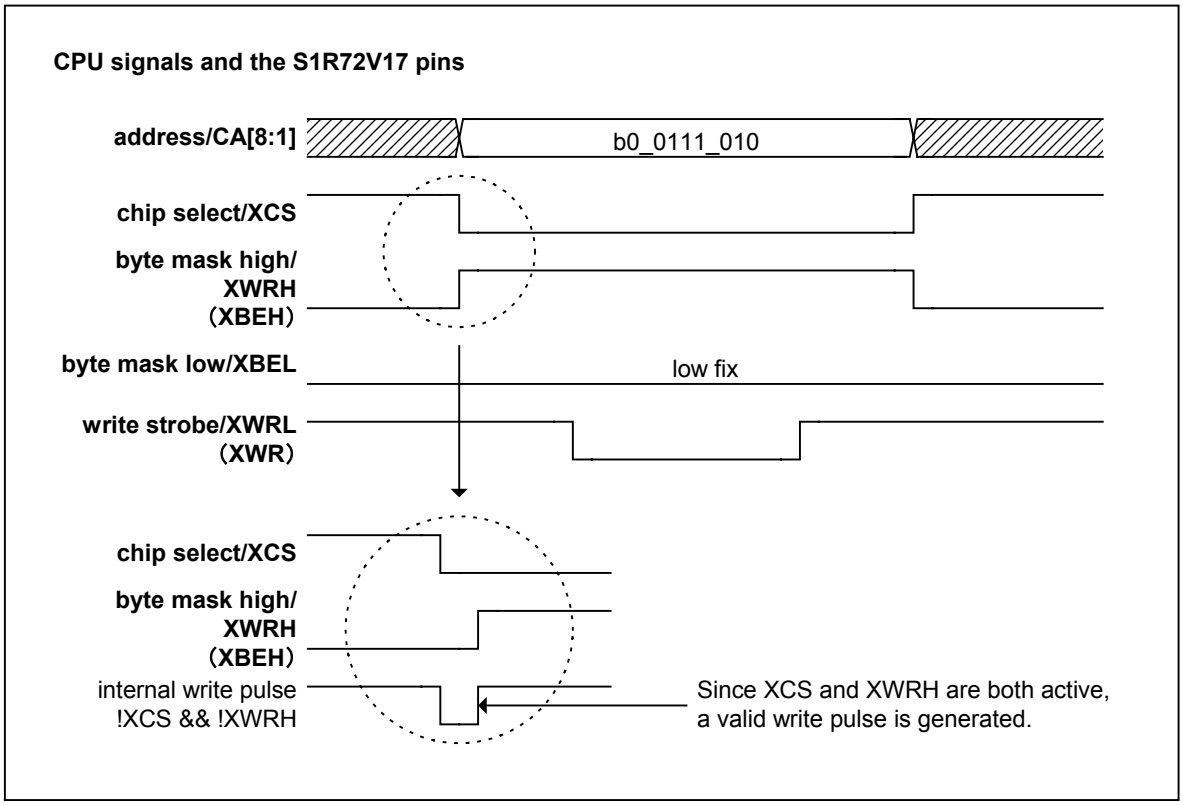



**Fig. 6.71 Initializing the CPU_Config register**

Once operating mode settings are completed, this limitation no longer applies, since the internal write pulse generation conditions are updated for 16-bit BE mode.

If the S1R72V17 is accessed for reads before the CPU_Config register is set, a problem may arise in the chip on which read and write operations happen to be performed simultaneously, as shown in Fig. 6.72. In such cases, LSI operations cannot be guaranteed. Be sure to set the CPU_Config register first.

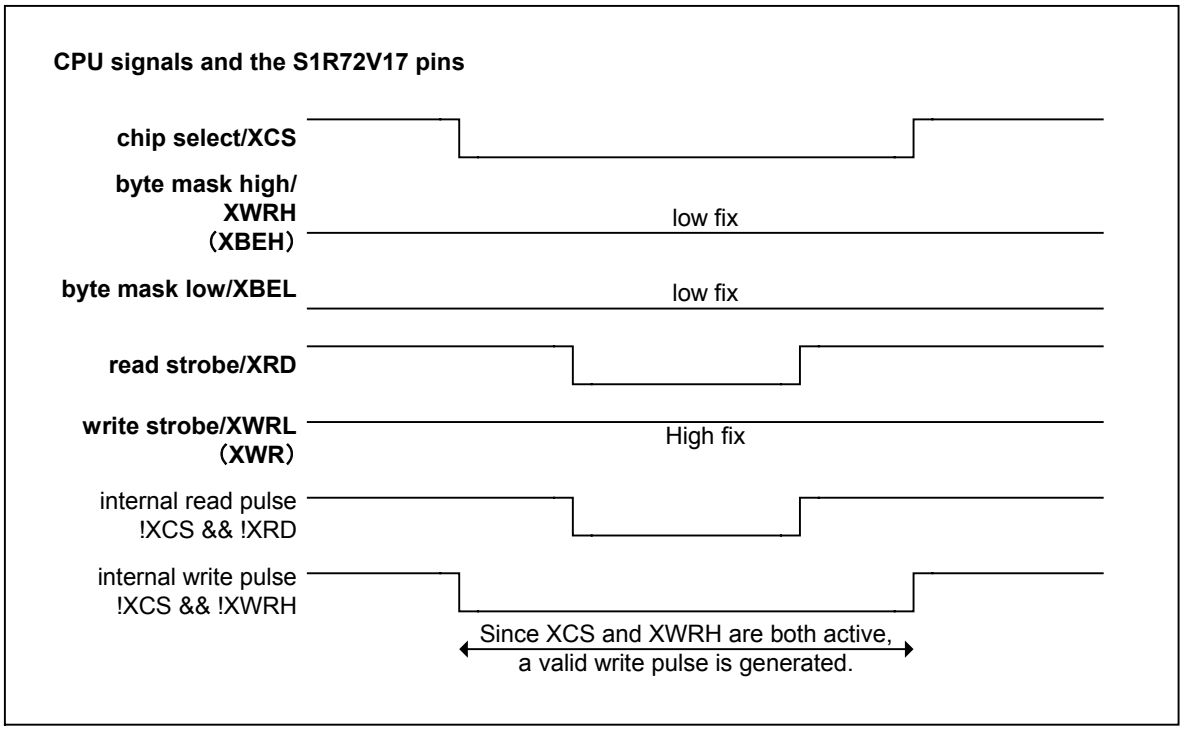



**Fig. 6.72 Read access before the CPU_Config register is initialized**

### 6.6.2.2. When Using 8-bit Mode

If the CPU is to be used in 8-bit mode, first set up the CPU_Config register as described in Section 6.6.1. If the S1R72V17 is accessed for reads before the CPU_Config register is set, all CD[15:0] pins will be placed in an output state because the S1R72V17 is operating in its initial state of 16-bit Strobe mode. While having CD[15:8] pins internally pulled up or down through resistors will not present particular problems, connecting these pins directly to VDD or GND will significantly increase chip current consumption. Be sure to set the CPU_Config register first.

### 6.6.3 Block Configuration

The block configuration of the CPUIF of the S1R72V17 (hereafter referred to as "CPUIF") is shown in Fig. 6.71.

The CPUIF is comprised of two blocks—REG, and DMA.

- REG: Controls access to the S1R72V17 register area
- DMA: DMA channel

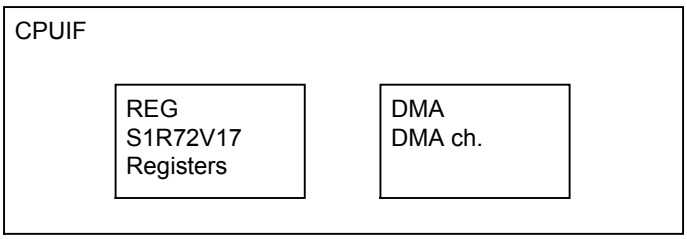



**Fig. 6.71 Block Configuration**

#### 6.6.3.1 REG (S1R72V17 Registers)

Controls access to the S1R72V17 register area. This includes the following access functions:

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

##### 6.6.3.1.1 Synchronous Register Access (Write)

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

##### 6.6.3.1.2 Synchronous Register Access (Read)

In this access, the register data is output to the external bus during a read period in which XCS and XRD both are asserted as an output enable period.

In a register read operation, the 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, because it is possible that an erroneous count value will be read due to a carry of the count 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 to the external bus when the lower bytes are read.

#### 6.6.3.1.3 FIFO Access (Write)

The FIFO write access refers to writing data to the FIFO_Wr_0,1 and RAM_WrDoor_0,1 registers. When operating in 8-bit mode, either one of the FIFO_Wr_0,1 registers can be accessed for write to the FIFO without causing any problem. The same applies to the RAM_WrDoor_0,1 registers.

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

- After setting the AREAn{n=0-5}Join_0.JoinCPU_Wr bit, inspect the FIFO_WrRemain_H,L registers to confirm the number of writable data bytes before accessing the FIFO. This limitation does not apply to the RAM_WrDoor_0,1 registers.
- When using a 16-bit CPU, the FIFO must basically be accessed by word (in 2-byte units). For writes of odd bytes, take the byte boundaries of the FIFO into consideration when controlling the strobe signal. For details, refer to Section 6.7.2.1.5, “Processing Odd Bytes in FIFO Access.”
- The FIFO_WrRemain_H,L registers, if inspected immediately after writing to the FIFO_Wr_0,1 registers, 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 the FIFO_WrRemain_H,L registers.
- The RAM_WrDoorAdr_H,L registers, if inspected immediately after writing to the RAM_WrDoor_0,1 registers, will not show the exact address. Be sure to insert an interval equal to 1 CPU cycle or more before reading the RAM_WrDoorAdr_H,L registers.

#### 6.6.3.1.4 FIFO Access (Read)

The FIFO read access refers to reading out data from the FIFO_Rd_0,1 or FIFO_ByteRd registers. When operating in 8-bit mode, either FIFO_Rd_0,1 or FIFO_ByteRd register can be accessed for read from the FIFO without causing any problem.

The FIFO read access is subject to the following limitations:

- After setting the AREAn{n=0-5}Join_0.JoinCPU_Wr bit, inspect the FIFO_RdRemain_H,L registers to confirm the number of readable data bytes and also check the RdRemainValid bit before accessing the FIFO.
- When operating in 16-bit mode, use the FIFO_Rd_0,1 registers to read data by word. Use the FIFO_ByteRd register to read data bytewise. If byte boundaries exist, read data bytewise. In this case, if data is read by word using the FIFO_Rd_0,1 registers, valid data is output only one side of the registers. For details, refer to Section 6.6.2.1.5, “Processing Odd Bytes in FIFO Access.”

#### 6.6.2.1.5 Processing Odd Bytes in FIFO Access

This section describes the relationship between the manner of how data is stored in the FIFO and the FIFO access made when handling odd bytes of data. 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 simplicity purpose. 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 data is found present after data was written wordwise from a byte boundary nonexistent state by setting the AREAnFIFO_Clr register bit, etc., 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 Fig. 6.72. The data is output from the USB, etc. 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 (write of data K) to eliminate the byte boundary and then write data wordwise (data L and M). This state is shown in (2) of Fig. 6.72.

Described above are the normal write operations.

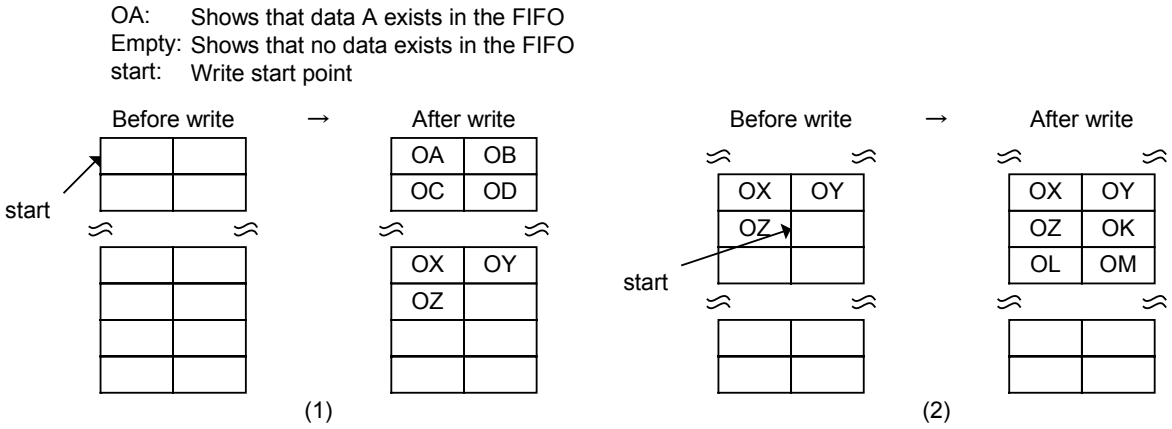



**Fig. 6.72 FIFO write processing (normal operations)**

Described below are the write operations that require caution.

If data is written wordwise while the FIFO has a byte boundary in it, a write to the High side is ignored, and data is written to only the Low side ((3) in Fig. 6.73). This is the same as writing data 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 write operation performed is ignored ((4) in Fig. 6.73).

If data is written to only the Low side while the FIFO has no byte boundary in it, the write operation performed is ignored ((5) in Fig. 6.73). Furthermore, if data is written wordwise while the FIFO has no byte boundary in it and the number of writable bytes is 1, a write to the Low side is ignored and data is written to only the High side ((6) in Fig. 6.73). This is the same as writing data to the High side bytewise.

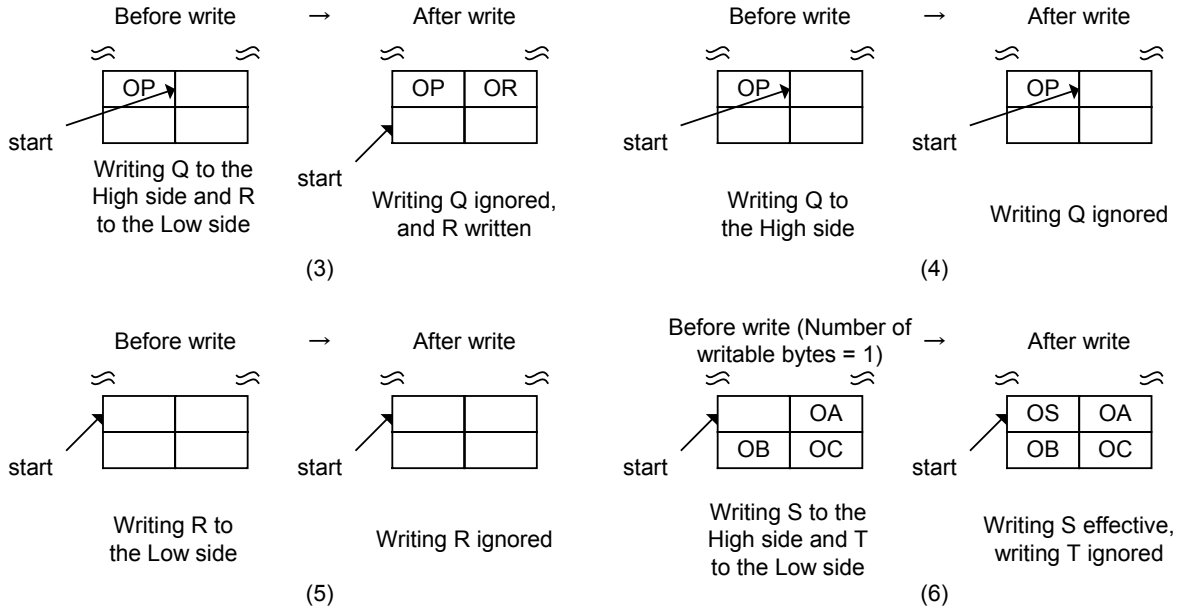



**Fig. 6.73 FIFO write processing (operations that require caution)**

[Read operation]

If no byte boundaries exist, data can be read wordwise using the FIFO_Rd_0,1 registers or can be read bytewise using the FIFO_ByteRd register without causing any problem. If any byte boundary exists, data must be read bytewise using the FIFO_ByteRd register. Once the byte boundary is eliminated, data can be read either wordwise or bytewise without causing any problem.

The manner of how data is read wordwise when no byte boundaries exist is shown in (1) of Fig. 6.74. Data A,B and then data C,D are read each time the FIFO is accessed. The manner of how data is read bytewise is shown in (2) of Fig. 6.74. Data A, data B, data C, and data D are read each time the FIFO is accessed. Described above are the normal read operations.

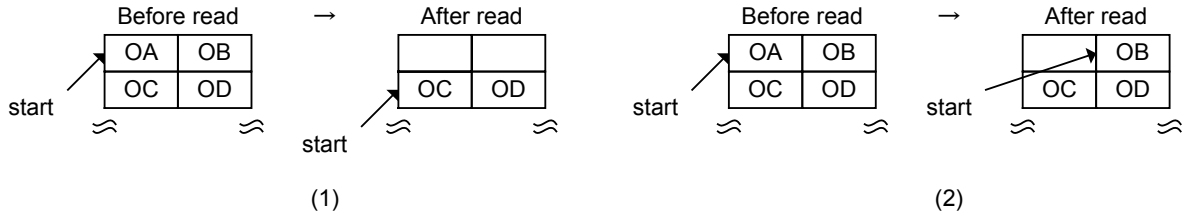



**Fig. 6.74 FIFO read processing (normal operations)**

Described below are the read operations that require caution.

Shown in (3) of Fig. 6.75 is an operation in which data is read wordwise using the FIFO_Rd_0,1 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 Fig. 6.75 is an operation in which data is read wordwise using the FIFO_Rd_0,1 registers while no byte boundary exists 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.

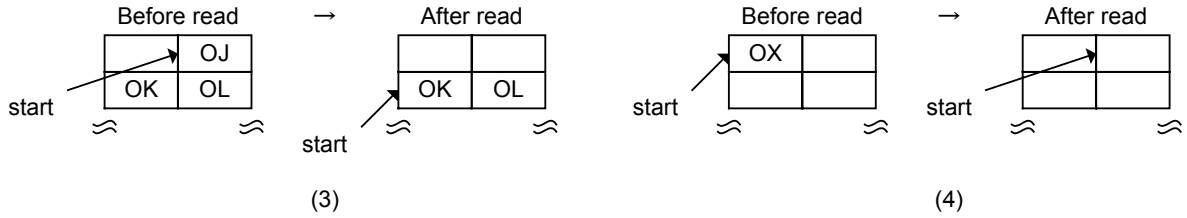



**Fig. 6.75 FIFO read processing (operations that require caution)**

Based on the above, the following shows an example read operation in odd bytes processing.

1) To read 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 FIFO_Rd_0,1 registers or read bytewise using the FIFO_ByteRd register.
   (3) The 31st byte of data is read bytewise using the FIFO_ByteRd register. ->A byte boundary is created.
   (4) The 32nd byte of data is read bytewise. In this case, it is recommended that the FIFO_ByteRd register be used for byte read. If the data is read wordwise using the FIFO_Rd_0,1 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 FIFO_Rd_0,1 registers or read bytewise using the FIFO_ByteRd register.
2) To read 64 bytes of data sent separately 31 bytes and then 33 bytes from the USB while JoinCPU_Rd is set, all wordwise by using the FIFO_Rd_H,L registers
   (1) When 31 bytes of data is 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.

#### 6.6.2.1.6 RAM_Rd Access

As with synchronous register read, data is output to the external bus during a read period in XCS and XRD both are asserted as an output enable period. For details, refer to Section 6.6.1.5.1 or 6.6.2.4.1, “Method for Accessing the FIFO (RAM_Rd).”

#### 6.6.2.1.7 Asynchronous Register Access (Write)

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

#### 6.6.2.1.8 Asynchronous Register Access (Read)

As with synchronous register read, the register data is output to the external bus during a read period in which XCS and XRD both are asserted as an output enable period.

### 6.6.2.2 DMA (DMA Channel)

#### 6.6.2.2.1 Basic Functionality

The basic operations of the DMA are described below.

[Write operation]

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

[Read operation]

When the FIFO has valid readable data and is readable, 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 counts set.
  When the internal FIFO has a writable free space or valid readable data and there is a remaining count in the DMA_Count_HH,HL,LH,LL registers, XDREQ is asserted to enable DMA transfers to be performed.
- Free-running mode
  When the internal FIFO has a 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/write by the CPU. This option can be used in either count mode or free-running mode. If the FIFO has a writable free space or valid readable data more than the assert counts set by the DMA_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 free space or data in the FIFO is less than the set assert count, if

count mode is selected and the said free space or data in the FIFO is greater than the remaining count, then XDREQ is asserted. In this case, the guaranteed transfer bytes equal the remaining count.

In 16-bit mode, DMA basically is data processed in word units. Data processing in byte units can be performed only when DMA is operating in count mode and the remaining count = 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 6.64 Operation Modes and Option vs. Transfer Start Conditions**

**Count mode with the ReqAssertCount option used (when operating in 16-bit or 8-bit mode)**

| Condition | Count mode (Count > 0) | | | |
|---|---|---|---|---|
| | Count≧Ready | | Count < Req | |
| | Ready≧Req | Ready < Req | Ready≧Count | Ready < Count |
| XDREQ | Asserted | Negated | Asserted | Negated |
| Transfers performed | Req | - | Count | - |

**Free-running mode with the ReqAssertCount option used (when operating in 16-bit or 8-bit mode)**

| Condition | Free-running mode | |
|---|---|---|
| | - | |
| | Ready≧Req | Ready < Req |
| XDREQ | Asserted | Negated |
| Transfers performed | Req | - |

**Count mode with the ReqAssertCount option unused (when operating in 16-bit mode)**

| Condition | Count mode (Count > 0) | | |
|---|---|---|---|
| | Count≧Ready | | Count < Ready |
| | Ready≧2 | Ready < 2 | Ready≧Count |
| XDREQ | Asserted | Negated | Asserted |
| Transfers performed | Ready (if Ready is an odd number, Ready - 1) | - | Count |

**Free-running mode with the ReqAssertCount option unused (when operating in 16-bit mode)**

| Condition | Free-running mode | |
|---|---|---|
| | - | |
| | Ready≧2 | Ready < 2 |
| XDREQ | Asserted | Negated |
| Transfers performed | Ready (if Ready is an odd number, Ready - 1) | - |

**Count mode with the ReqAssertCount option unused (when operating in 8-bit mode)**

| Condition | Count mode (Count > 0) | | |
|---|---|---|---|
| | Count≧Ready | | Count < Ready |
| | Ready≧1 | Ready < 1 | Ready≧Count |
| XDREQ | Asserted | Negated | Asserted |
| Transfers performed | Ready | - | Count |

**Free-running mode with the ReqAssertCount option unused (when operating in 8-bit mode)**

| Condition | Free-running mode | |
|---|---|---|
| | - | |
| | Ready≧1 | Ready < 1 |
| XDREQ | Asserted | Negated |
| Transfers performed | Ready | - |

* In the above table, Req indicates the set value of DMA_Config.ReqAssertCount, Ready indicates the free space/data bytes in the FIFO, and Count indicates the value of DMA_Count_HH,HL,LH,LL.

#### 6.6.2.2.2 Pin Settings

It is possible to set the XDREQ and XDACK logic levels each by setting up the CPU_Config register. In the explanation below, XDREQ and XDACK both are described as being active-low (negative logic) unless otherwise noted.

#### 6.6.2.2.3 Count Mode (Write)

[Starting operation]

After setting a count value in the DMA_Count_HH,HL,LH,LL registers, set the DMA_Control_DMA_Go bit to 1. If the internal FIFO has 2 bytes or more of writable free space (DMA_Ready) (for 8-bit mode, 1 byte or more) and has any remaining count, XDREQ is asserted to enable DMA transfers to be performed. If the free space available in the FIFO is only 1 byte, count mode is selected, in which case XDREQ is asserted only when the remaining count = 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. to eliminate the byte boundary before starting the next write operation. To transfer data from the USB 31 bytes each time after writing data from the DMA 31 bytes each 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 DMA_Control.DMA_Running bit reads “1.”

[Stopping operation]

There are two conditions to stop the operation:

- DMA transfers equal to the counts set in the DMA_Count_HH,HL,LH,LL registers are completed.
- The DMA_Control.DMA_Stop bit is set by writing 1 in software.

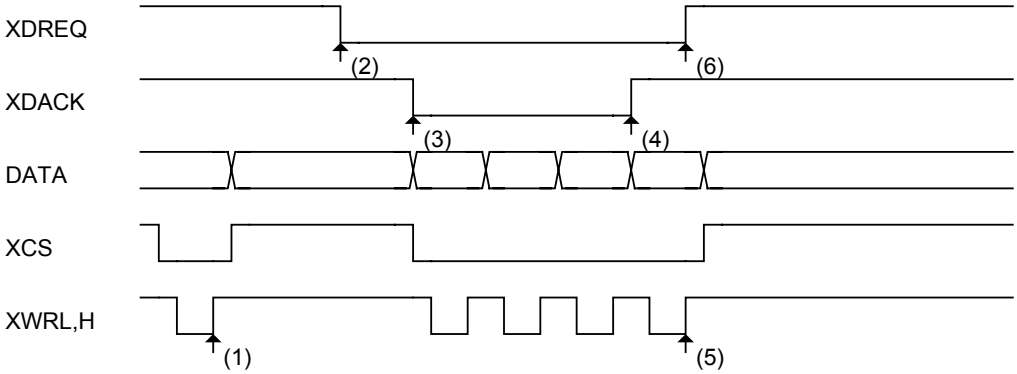
When the DMA operation stops, the CPU_IntStat.DMA_Cmp bit is set.

When the transfer stops due to expiration of counts set in the DMA_Count_HH,HL,LH,LL registers, XDREQ is negated during the strobe assert period of the last access.

When the transfer stops due to DMA_Stop bit, the chip’s internal operation is halted in synch 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.

Fig. 6.76 shows an operation timing for the case where a transfer is started in count mode and the transfer is stopped by setting the DMA_Control.DMA_Stop bit before transfers for the set count are completed.

Example 1: [Transfer start condition] Count (8 bytes) < free space in FIFO (16 bytes)
[Transfer stop condition] DMA_Stop

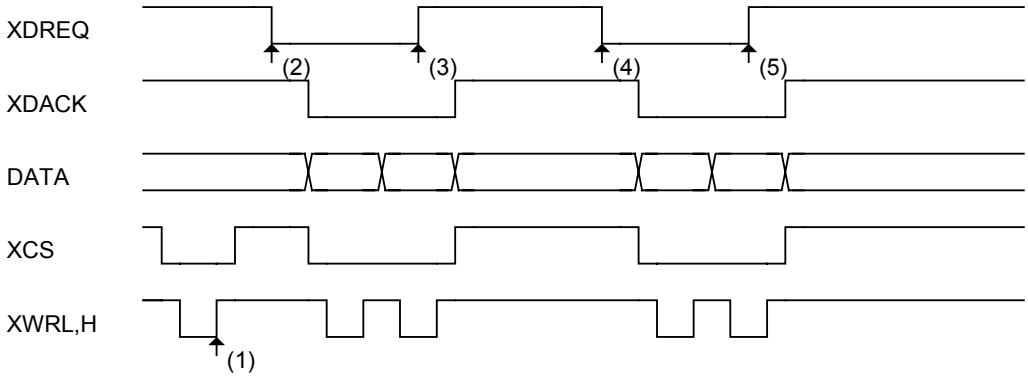



(1) The DMA circuit is activated by writing 1 to the DMA_Control.DMA_Go bit.
(2) A free space is created in the FIFO (DMA_Ready) due to transfer of data from the USB, etc., and XDREQ is asserted in response to DMA_Ready.
(3) XDACK is asserted causing a DMA transfer to start.
(4) The master is stopped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

**Fig. 6.76 Count mode write timing 1**

Fig. 6.77 shows an operation timing for the case where a transfer is started in count mode and when the DMA transfer finishes due to completion of transfers for the set count.

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) A free space is created in the FIFO (DMA_Ready) due to the transfer of data from the USB, etc., and XDREQ is asserted in response to DMA_Ready.
(3) XDREQ is negated in synch with the disappearance of DMA_Ready.
(4) A free space is created in the FIFO (DMA_Ready) due to the transfer of data from the USB, etc., and XDREQ is asserted in response to DMA_Ready.
(5) XDREQ is negated in synch with the last data timing of DMA_Count. The DMA circuit stops due to completion of transfers equal to DMA_Count.

**Fig. 6.77 Count mode write timing 2**

#### 6.6.2.2.4 Count Mode (Read)

[Starting operation]

After setting a count value in the DMA_Count_HH,HL,LH,LL registers, set the DMA_Control_DMA_Go bit to 1. If the internal FIFO has 2 bytes or more of valid readable data (for 8-bit mode, 1 byte or more) and has any remaining count, and while in that state, the FIFO is prepared for read from an external device, XDREQ is asserted to enable DMA transfers to be performed. If the remaining data available in the FIFO is only 1 byte, count mode is selected, in which case XDREQ is asserted only when the remaining count = 1.

For read in count mode during USB device operation for example, when bytes of data more than the counts set in the DMA_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 to return a NAK response. Furthermore, even when a short packet is received from the USB, unless the DisAF_NAK_Short is set, the ForceNAK bit for the relevant endpoint is automatically set to 1 to return a 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 data from the DMA 31 bytes each time after data is transferred from the USB 31 bytes each time, for example, (1) receive 31 bytes of data from the USB (at this point, the ForceNAK is set and the relevant endpoint returns a NAK response), (2) read 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 DMA_Control.DMA_Running bit reads “1.”

[Stopping operation]

There are two conditions to stop the operation:

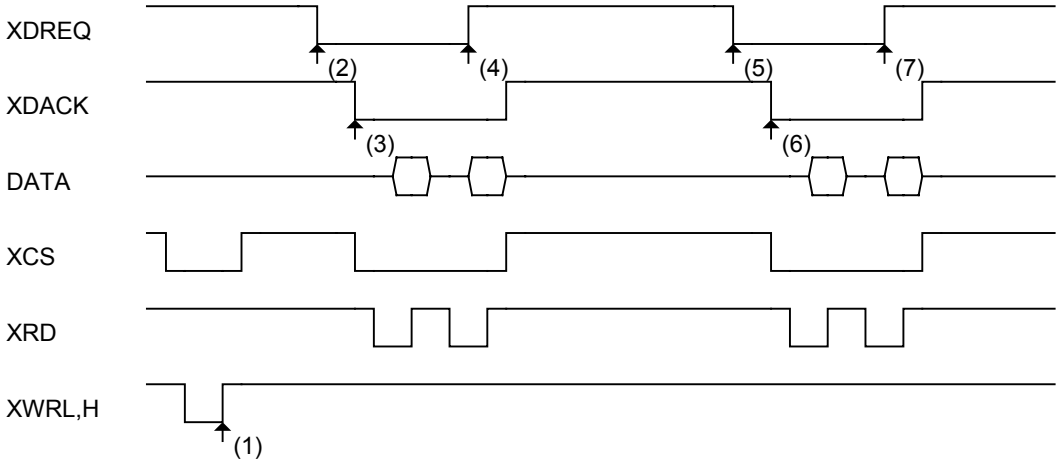
- DMA transfers equal to the counts set in the DMA_Count_HH,HL,LH,LL registers are completed.
- The DMA_Control.DMA_Stop bit is set by writing 1 in software.

When the transfer stops due to expiration of counts set in the DMA_Count_HH,HL,LH,LL registers, XDREQ is negated during the strobe signal assert period of the last access.

When the transfer stops due to the DMA_Stop bit, 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.

Fig. 6.78 shows an operation timing for the case where a transfer is started in count mode, and when the DMA transfer finishes due to completion of transfers for the set count.

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 to the FIFO from the USB, etc., and the FIFO is readable from an external device, XDREQ is asserted.
(3) XDACK is asserted causing a DMA transfer to start.
(4) XDREQ is negated in synch 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 readable from an external device, XDREQ is asserted.
(6) XDACK is asserted causing a DMA transfer to start.
(7) XDREQ is negated in synch with the last data timing of DMA_Count.

**Fig. 6.78 Count mode read timing**

#### 6.6.2.2.5 Free-running Mode (Write)

[Starting operation]

After setting the DMA_Config.FreeRun bit, set the DMA_Control_DMA_Go bit by writing 1. If the internal FIFO has 2 bytes or more of writable free space (DMA_Ready) (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 is only 1 byte, XDREQ is not asserted when 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 DMA_Control.DMA_Running bit reads "1."

[Stopping operation]

There is one condition to stop the operation:

- The DMA_Control.DMA_Stop bit is set by writing 1 in software.

When the transfer stops due to the DMA_Stop bit, the chip's internal operation is halted in synch 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 DMA_Count_HH,HL,LH,LL register value overflows after reaching terminal count during a DMA transfer in free-running mode, the CPU_IntStat.DMA_Countup bit is set.

Even in this case, the DMA transfer is continued and the count in the DMA_Count_HH,HL,LH,LL restarts incrementing.

The operation timing in free-running mode is the same as in count mode, except that the limitations due to DMA_Count_HH,HL,LH,LL are nonexistent.

#### 6.6.2.2.6 Free-running Mode (Read)

[Starting operation]

After setting the DMA_Config.FreeRun bit, set the DMA_Control_DMA_Go bit by writing 1. If the internal FIFO has 2 bytes or more of valid readable data (for 8-bit mode, 1 byte or more) and is readable from an external device, XDREQ is asserted. If the remaining valid data in the FIFO is only 1 byte, DMA transfer is not started. If transfers need to be performed, refer to the explanation of count mode in the preceding section.

Until the operation stops, the DMA_Control.DMA_Running bit reads "1."

[Stopping operation]

There is one condition to stop the operation:

- The DMA_Control.DMA_Stop bit is set by writing 1 in software.

When the transfer stops due to the DMA_Stop bit, the chip's internal operation is halted in synch 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 DMA_Count_HH,HL,LH,LL register value overflows after reaching terminal count during a DMA transfer in free-running mode, the CPU_IntStat.DMA_Countup bit is set. Even in this case, the DMA transfer is continued and the count in the DMA_Count_HH,HL,LH,LL restarts incrementing.

The operation timing in free-running mode is the same as in count mode, except that the limitations due to DMA_Count_HH,HL,LH,LL are nonexistent.

#### 6.6.2.2.7 REQ Assert Count Option (Write)

[Starting operation]

After setting an assert count with the DMA_Config.ReqAssertCount [1:0] bits, set the DMA_Control_DMA_Go bit to 1. If the internal FIFO has more bytes of writable free space than the set assert count, 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 count are guaranteed. However, even when the free space in the FIFO is less than the set assert count, if count mode is selected and the said free space in the FIFO is greater than the remaining count, then XDREQ is asserted. In this case, the guaranteed transfer bytes equal the remaining count.

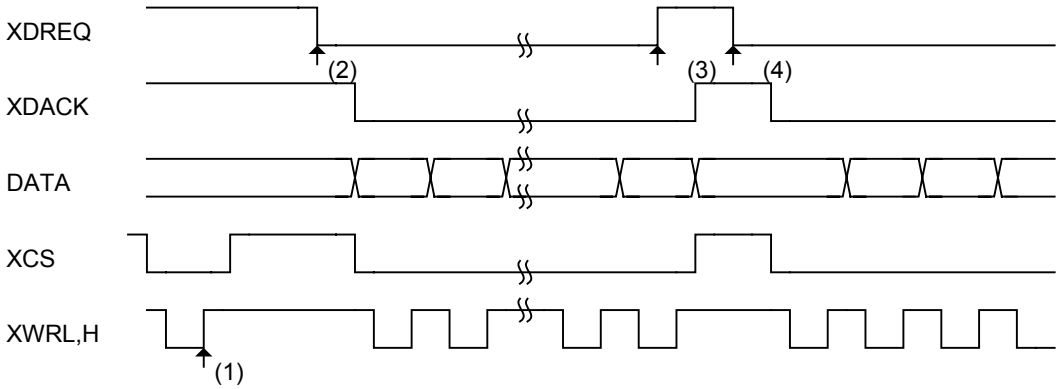
In this mode, XDREQ is temporarily negated every transfer bytes set in the ReqAssertCount [1:0] bits.

Until the operation stops, the DMA_Control.DMA_Running bit reads "1."

[Stopping operation]

For condition(s) that stop the operation, 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.
Since the DMA_Ready value is less than required for successive transfers, XDREQ is not asserted.
(2) When data is transferred from the USB, etc., and a valid free space greater than required for successive transfers is created in the FIFO (DMA_Ready).
XDREQ is asserted in response to DMA_Ready.
(3) XDREQ is negated in synch upon completion of successive transfers (REQ assert count).
(4) When the first round of successive transfers is complete, a free space for the next successive transfers is available (DMA_Ready).
XDREQ is asserted in response to DMA_Ready.

**Fig. 6.79 REQ assert count option write timing**

#### 6.6.2.2.8 REQ Assert Count Option (Read)

[Starting operation]

After setting an assert count with the DMA_Config.ReqAssertCount [1:0] bits, set the DMA_Control_DMA_Go bit to 1. If the internal FIFO has more bytes of valid readable data than the set assert count and is readable from an external device, 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 count are guaranteed. However, even when the data in the FIFO is less than the REQ assert count, if count mode is selected and the said data in the FIFO is greater than the remaining count, then XDREQ is asserted. In this case, the guaranteed transfer bytes equal the remaining count.

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

Until the operation stops, the DMA_Control.DMA_Running bit reads “1.”

[Stopping operation]

For the condition(s) that stop the operation, refer to the explanation of count mode and free-running mode in the preceding sections.

For the operation timing, refer to Fig. 6.78 and Fig. 6.79.

#### 6.6.2.2.9 FIFO Access Odd Bytes Processing in DMA

Refer to Section 6.6.2.1.5“Processing Odd Bytes in FIFO Access.” Note that the DMA has no entries for byte read.

# 7. Registers

The registers in the S1R72V17 are classified into three groups: shared device/host registers, device registers, and host registers.

Do not write 1 to the reserved register bits.

## 7.1 Device/Host Shared Register Map

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000 | ***MainIntStat*** | R/(W) | 0x00 | ***USB_DeviceIntStat*** | ***USB_HostIntStat*** | CPU_IntStat | FIFO_IntStat | | | | ***FinishedPM*** |
| 0x001 | ***USB_DeviceIntStat*** | R/(W) | 0x00 | ***VBUS_Changed*** | | ***D_SIE_IntStat*** | D_BulkIntStat | RcvEP0SETUP | | D_EP0IntStat | D_EPrIntStat |
| 0x002 | ***USB_HostIntStat*** | R/(W) | 0x00 | ***VBUS_Err*** | | H_SIE_IntStat_1 | H_SIE_IntStat_0 | H_FrameIntStat | | H_CH0IntStat | H_CHrIntStat |
| 0x003 | CPU_IntStat | R/(W) | 0x00 | RAM_RdCmp | | | | | | DMA_Countup | DMA_Cmp |
| 0x004 | FIFO_IntStat | R/(W) | 0x00 | FIFO_DMA_Cmp | | | | | FIFO_NotEmpty | FIFO_Full | FIFO_Empty |
| 0x005 | | | 0xXX | | | | | | | | |
| 0x006 | | | 0xXX | | | | | | | | |
| 0x007 | | | 0xXX | | | | | | | | |
| 0x008 | ***MainIntEnb*** | R/W | 0x00 | ***EnUSB_DeviceIntStat*** | ***EnUSB_HostIntStat*** | EnCPU_IntStat | EnFIFO_IntStat | | | | ***EnFinishedPM*** |
| 0x009 | ***USB_DeviceIntEnb*** | R/W | 0x00 | ***EnVBUS_Changed*** | | ***EnD_SIE_IntStat*** | EnD_BulkIntStat | EnRcvEP0SETUP | | EnD_EP0IntStat | EnD_EPrIntStat |
| 0x00A | ***USB_HostIntEnb*** | R/W | 0x00 | ***EnVBUS_Err*** | | EnH_SIE_IntStat_1 | EnH_SIE_IntStat_0 | EnH_FrameIntStat | | EnH_CH0IntStat | EnH_CHrIntStat |
| 0x00B | CPU_IntEnb | R/W | 0x00 | EnRAM_RdCmp | | | | | | EnDMA_Countup | EnDMA_Cmp |
| 0x00C | FIFO_IntEnb | R/W | 0x00 | EnFIFO_DMA_Cmp | | | | | EnFIFO_NotEmpty | EnFIFO_Full | EnFIFO_Empty |
| 0x00D | | | 0xXX | | | | | | | | |
| 0x00E | | | 0xXX | | | | | | | | |
| 0x00F | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x010 | ***RevisionNum*** | R | 0x10 | ***RevisionNumber*** | | | | | | | |
| 0x011 | ***ChipReset*** | R/W | 0x80 | ***ResetMTM*** | | | | | | | ***AllReset*** |
| 0x012 | ***PM_Control*** | R/W | 0x00 | ***GoSLEEP*** | ***GoACTIVE*** | ***GoCPU_Cut*** | | | | ***PM_State[1:0]*** | |
| 0x013 | | | 0xXX | | | | | | | | |
| 0x014 | ***WakeupTim_H*** | R/W | 0x00 | ***WakeupTim [15:8]*** | | | | | | | |
| 0x015 | ***WakeupTim_L*** | R/W | 0x00 | ***WakeupTim [7:0]*** | | | | | | | |
| 0x016 | ***H_USB_Control*** | R/W | 0x00 | ***VBUS_Enb*** | | | | | | | |
| 0x017 | ***H_XcvrControl*** | R/W | 0x91 | ***TermSelect*** | ***RemoveRPD*** | ***XcvrSelect[1:0]*** | | | | ***OpMode[1:0]*** | |
| 0x018 | ***D_USB_Status*** | R/W | 0xXX | ***VBUS*** | FSxHS | | | | | ***LineState[1:0]*** | |
| 0x019 | ***H_USB_Status*** | R | 0xXX | ***VBUS_State*** | | | | | | ***LineState[1:0]*** | |
| 0x01A | | | 0xXX | | | | | | | | |
| 0x01B | MTM_Config | R/W | 0x00 | | | MTM_SlopeValue [1:0] | | | | MTM_TermValue [1:0] | |
| 0x01C | | | 0xXX | | | | | | | | |
| 0x01D | | | 0xXX | | | | | | | | |
| 0x01E | | | 0xXX | | | | | | | | |
| 0x01F | ***HostDeviceSel*** | R/W | 0x00 | | | | | | | | ***HOSTxDEVICE*** |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x020 | FIFO_Rd_0 | R | 0xXX | FIFO_Rd_0[7:0] | | | | | | | |
| 0x021 | FIFO_Rd_1 | R | 0xXX | FIFO_Rd_1[7:0] | | | | | | | |
| 0x022 | FIFO_Wr_0 | W | 0xXX | FIFO_Wr_0[7:0] | | | | | | | |
| 0x023 | FIFO_Wr_1 | W | 0xXX | FIFO_Wr_1[7:0] | | | | | | | |
| 0x024 | FIFO_RdRemain_H | R | 0x00 | RdRemainValid | | | RdRemain[12:8] | | | | |
| 0x025 | FIFO_RdRemain_L | R | 0x00 | RdRemain[7:0] | | | | | | | |
| 0x026 | FIFO_WrRemain_H | R | 0x00 | | | | WrRemain[12:8] | | | | |
| 0x027 | FIFO_WrRemain_L | R | 0x00 | WrRemain[7:0] | | | | | | | |
| 0x028 | FIFO_ByteRd | R | 0xXX | FIFO_ByteRd[7:0] | | | | | | | |
| 0x029 | | | 0xXX | | | | | | | | |
| 0x02A | | | 0xXX | | | | | | | | |
| 0x02B | | | 0xXX | | | | | | | | |
| 0x02C | | | 0xXX | | | | | | | | |
| 0x02D | | | 0xXX | | | | | | | | |
| 0x02E | | | 0xXX | | | | | | | | |
| 0x02F | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x030 | RAM_RdAdrs_H | R/W | 0x00 | | | | RAM_RdAdrs[12:8] | | | | |
| 0x031 | RAM_RdAdrs_L | R/W | 0x00 | RAM_RdAdrs[7:2] | | | | | | | |
| 0x032 | RAM_RdControl | R/W | 0x00 | RAM_GoRdCBW_CSW | RAM_GoRd | | | | | | |
| 0x033 | | | 0xXX | | | | | | | | |
| 0x034 | | | 0xXX | | | | | | | | |
| 0x035 | RAM_RdCount | R/W | 0x00 | | | RAM_RdCount[5:2] | | | | | |
| 0x036 | | | 0xXX | | | | | | | | |
| 0x037 | | | 0xXX | | | | | | | | |
| 0x038 | RAM_WrAdrs_H | R/W | 0x00 | | | | RAM_WrAdrs[12:8] | | | | |
| 0x039 | RAM_WrAdrs_L | R/W | 0x00 | RAM_WrAdrs[7:0] | | | | | | | |
| 0x03A | RAM_WrDoor_0 | W | 0xXX | RAM_WrDoor_0[7:0] | | | | | | | |
| 0x03B | RAM_WrDoor_1 | W | 0xXX | RAM_WrDoor_1[7:0] | | | | | | | |
| 0x03C | | | 0xXX | | | | | | | | |
| 0x03D | | | 0xXX | | | | | | | | |
| 0x03E | | | 0xXX | | | | | | | | |
| 0x03F | | | 0xXX | | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x040 | RAM_Rd_00 | R | 0x00 | RAM_Rd_00[7:0] | | | | | | | |
| 0x041 | RAM_Rd_01 | R | 0x00 | RAM_Rd_01[7:0] | | | | | | | |
| 0x042 | RAM_Rd_02 | R | 0x00 | RAM_Rd_02[7:0] | | | | | | | |
| 0x043 | RAM_Rd_03 | R | 0x00 | RAM_Rd_03[7:0] | | | | | | | |
| 0x044 | RAM_Rd_04 | R | 0x00 | RAM_Rd_04[7:0] | | | | | | | |
| 0x045 | RAM_Rd_05 | R | 0x00 | RAM_Rd_05[7:0] | | | | | | | |
| 0x046 | RAM_Rd_06 | R | 0x00 | RAM_Rd_06[7:0] | | | | | | | |
| 0x047 | RAM_Rd_07 | R | 0x00 | RAM_Rd_07[7:0] | | | | | | | |
| 0x048 | RAM_Rd_08 | R | 0x00 | RAM_Rd_08[7:0] | | | | | | | |
| 0x049 | RAM_Rd_09 | R | 0x00 | RAM_Rd_09[7:0] | | | | | | | |
| 0x04A | RAM_Rd_0A | R | 0x00 | RAM_Rd_0A[7:0] | | | | | | | |
| 0x04B | RAM_Rd_0B | R | 0x00 | RAM_Rd_0B[7:0] | | | | | | | |
| 0x04C | RAM_Rd_0C | R | 0x00 | RAM_Rd_0C[7:0] | | | | | | | |
| 0x04D | RAM_Rd_0D | R | 0x00 | RAM_Rd_0D[7:0] | | | | | | | |
| 0x04E | RAM_Rd_0E | R | 0x00 | RAM_Rd_0E[7:0] | | | | | | | |
| 0x04F | RAM_Rd_0F | R | 0x00 | RAM_Rd_0F[7:0] | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x050 | RAM_Rd_10 | R | 0x00 | RAM_Rd_10[7:0] | | | | | | | |
| 0x051 | RAM_Rd_11 | R | 0x00 | RAM_Rd_11[7:0] | | | | | | | |
| 0x052 | RAM_Rd_12 | R | 0x00 | RAM_Rd_12[7:0] | | | | | | | |
| 0x053 | RAM_Rd_13 | R | 0x00 | RAM_Rd_13[7:0] | | | | | | | |
| 0x054 | RAM_Rd_14 | R | 0x00 | RAM_Rd_14[7:0] | | | | | | | |
| 0x055 | RAM_Rd_15 | R | 0x00 | RAM_Rd_15[7:0] | | | | | | | |
| 0x056 | RAM_Rd_16 | R | 0x00 | RAM_Rd_16[7:0] | | | | | | | |
| 0x057 | RAM_Rd_17 | R | 0x00 | RAM_Rd_17[7:0] | | | | | | | |
| 0x058 | RAM_Rd_18 | R | 0x00 | RAM_Rd_18[7:0] | | | | | | | |
| 0x059 | RAM_Rd_19 | R | 0x00 | RAM_Rd_19[7:0] | | | | | | | |
| 0x05A | RAM_Rd_1A | R | 0x00 | RAM_Rd_1A[7:0] | | | | | | | |
| 0x05B | RAM_Rd_1B | R | 0x00 | RAM_Rd_1B[7:0] | | | | | | | |
| 0x05C | RAM_Rd_1C | R | 0x00 | RAM_Rd_1C[7:0] | | | | | | | |
| 0x05D | RAM_Rd_1D | R | 0x00 | RAM_Rd_1D[7:0] | | | | | | | |
| 0x05E | RAM_Rd_1E | R | 0x00 | RAM_Rd_1E[7:0] | | | | | | | |
| 0x05F | RAM_Rd_1F | R | 0x00 | RAM_Rd_1F[7:0] | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x060 | | | 0xXX | | | | | | | | |
| 0x061 | DMA_Config | R/W | 0x00 | FreeRun | DMA_Mode | | | ActiveDMA | | ReqAssertCount [1:0] | |
| 0x062 | DMA_Control | R/W | 0x00 | DMA_Running | | | CounterClr | Dir | | DMA_Stop | DMA_Go |
| 0x063 | | | 0xXX | | | | | | | | |
| 0x064 | DMA_Remain_H | R | 0x00 | | | | DMA_Remain [12:8] | | | | |
| 0x065 | DMA_Remain_L | R | 0x00 | DMA_Remain [7:0] | | | | | | | |
| 0x066 | | | 0xXX | | | | | | | | |
| 0x067 | | | 0xXX | | | | | | | | |
| 0x068 | DMA_Count_HH | R/W | 0x00 | DMA_Count [31:24] | | | | | | | |
| 0x069 | DMA_Count_HL | R/W | 0x00 | DMA_Count [23:16] | | | | | | | |
| 0x06A | DMA_Count_LH | R/W | 0x00 | DMA_Count [15:8] | | | | | | | |
| 0x06B | DMA_Count_LL | R/W | 0x00 | DMA_Count [7:0] | | | | | | | |
| 0x06C | DMA_RdData_0 | R | 0xXX | DMA_RdData_0[7:0] | | | | | | | |
| 0x06D | DMA_RdData_1 | R | 0xXX | DMA_RdData_1[7:0] | | | | | | | |
| 0x06E | DMA_WrData_0 | W | 0xXX | DMA_WrData_0[7:0] | | | | | | | |
| 0x06F | DMA_WrData_1 | W | 0xXX | DMA_WrData_1[7:0] | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x070 | | | 0xXX | | | | | | | | |
| 0x071 | ***ModeProtect*** | R/W | 0x56 | ***Protected[7:0](writing other than 56 enables protect; 0x56 disables protect)*** | | | | | | | |
| 0x072 | | | 0xXX | | | | | | | | |
| 0x073 | ***ClkSelect*** | R/W | 0x00 | ***ClkSource*** | | | | | | ***ClkFreq[1:0]*** | |
| 0x074 | | | 0xXX | | | | | | | | |
| 0x075 | ***CPU_Config*** | R/W | 0x00 | ***IntLevel*** | ***IntMode*** | ***DREQ_Level*** | ***DACK_Level*** | ***CS_Mode*** | ***CPU_Endian*** | ***BusMode*** | ***Bus8x16*** |
| 0x076 | | | 0xXX | | | | | | | | |
| 0x077 | ***CPU_ChgEndian*** | R | 0xXX | ***Performing a dummy read of this register bits causes the endian set by CPU_Config.CPU_Endian to take effect.*** | | | | | | | |
| 0x078 | | | 0xXX | | | | | | | | |
| 0x079 | | | 0xXX | | | | | | | | |
| 0x07A | | | 0xXX | | | | | | | | |
| 0x07B | | | 0xXX | | | | | | | | |
| 0x07C | | | 0xXX | | | | | | | | |
| 0x07D | | | 0xXX | | | | | | | | |
| 0x07E | | | 0xXX | | | | | | | | |
| 0x07F | | | 0xXX | | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x080 | AREA0StartAdrs_H | R/W | 0x00 | | | | StartAdrs[12:8] | | | | |
| 0x081 | AREA0StartAdrs_L | R/W | 0x00 | StartAdrs[7:2] | | | | | | | |
| 0x082 | AREA0EndAdrs_H | R/W | 0x00 | | | | EndAdrs[12:8] | | | | |
| 0x083 | AREA0EndAdrs_L | R/W | 0x00 | EndAdrs[7:2] | | | | | | | |
| 0x084 | AREA1StartAdrs_H | R/W | 0x00 | | | | StartAdrs[12:8] | | | | |
| 0x085 | AREA1StartAdrs_L | R/W | 0x00 | StartAdrs[7:2] | | | | | | | |
| 0x086 | AREA1EndAdrs_H | R/W | 0x00 | | | | EndAdrs[12:8] | | | | |
| 0x087 | AREA1EndAdrs_L | R/W | 0x00 | EndAdrs[7:2] | | | | | | | |
| 0x088 | AREA2StartAdrs_H | R/W | 0x00 | | | | StartAdrs[12:8] | | | | |
| 0x089 | AREA2StartAdrs_L | R/W | 0x00 | StartAdrs[7:2] | | | | | | | |
| 0x08A | AREA2EndAdrs_H | R/W | 0x00 | | | | EndAdrs[12:8] | | | | |
| 0x08B | AREA2EndAdrs_L | R/W | 0x00 | EndAdrs[7:2] | | | | | | | |
| 0x08C | AREA3StartAdrs_H | R/W | 0x00 | | | | StartAdrs[12:8] | | | | |
| 0x08D | AREA3StartAdrs_L | R/W | 0x00 | StartAdrs[7:2] | | | | | | | |
| 0x08E | AREA3EndAdrs_H | R/W | 0x00 | | | | EndAdrs[12:8] | | | | |
| 0x08F | AREA3EndAdrs_L | R/W | 0x00 | EndAdrs[7:2] | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x090 | AREA4StartAdrs_H | R/W | 0x00 | | | | StartAdrs[12:8] | | | | |
| 0x091 | AREA4StartAdrs_L | R/W | 0x00 | StartAdrs[7:2] | | | | | | | |
| 0x092 | AREA4EndAdrs_H | R/W | 0x00 | | | | EndAdrs[12:8] | | | | |
| 0x093 | AREA4EndAdrs_L | R/W | 0x00 | EndAdrs[7:2] | | | | | | | |
| 0x094 | AREA5StartAdrs_H | R/W | 0x00 | | | | StartAdrs[12:8] | | | | |
| 0x095 | AREA5StartAdrs_L | R/W | 0x00 | StartAdrs[7:2] | | | | | | | |
| 0x096 | AREA5EndAdrs_H | R/W | 0x00 | | | | EndAdrs[12:8] | | | | |
| 0x097 | AREA5EndAdrs_L | R/W | 0x00 | EndAdrs[7:2] | | | | | | | |
| 0x098 | | | 0xXX | | | | | | | | |
| 0x099 | | | 0xXX | | | | | | | | |
| 0x09A | | | 0xXX | | | | | | | | |
| 0x09B | | | 0xXX | | | | | | | | |
| 0x09C | | | 0xXX | | | | | | | | |
| 0x09D | | | 0xXX | | | | | | | | |
| 0x09E | | | 0xXX | | | | | | | | |
| 0x09F | AREAnFIFO_Clr | W | 0xXX | | | ClrAREA5 | ClrAREA4 | ClrAREA3 | ClrAREA2 | ClrAREA1 | ClrAREA0 |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0A0 | AREA0Join_0 | R/W | 0x00 | JoinFIFO_Stat | | | | | JoinDMA | JoinCPU_Rd | JoinCPU_Wr |
| 0x0A1 | AREA0Join_1 | R/W | 0x00 | | | JoinEPeCHe | JoinEPdCHd | JoinEPcCHc | JoinEPbCHb | JoinEPaCHa | JoinEP0CH0 |
| 0x0A2 | AREA1Join_0 | R/W | 0x00 | JoinFIFO_Stat | | | | | JoinDMA | JoinCPU_Rd | JoinCPU_Wr |
| 0x0A3 | AREA1Join_1 | R/W | 0x00 | | | JoinEPeCHe | JoinEPdCHd | JoinEPcCHc | JoinEPbCHb | JoinEPaCHa | JoinEP0CH0 |
| 0x0A4 | AREA2Join_0 | R/W | 0x00 | JoinFIFO_Stat | | | | | JoinDMA | JoinCPU_Rd | JoinCPU_Wr |
| 0x0A5 | AREA2Join_1 | R/W | 0x00 | | | JoinEPeCHe | JoinEPdCHd | JoinEPcCHc | JoinEPbCHb | JoinEPaCHa | JoinEP0CH0 |
| 0x0A6 | AREA3Join_0 | R/W | 0x00 | JoinFIFO_Stat | | | | | JoinDMA | JoinCPU_Rd | JoinCPU_Wr |
| 0x0A7 | AREA3Join_1 | R/W | 0x00 | | | JoinEPeCHe | JoinEPdCHd | JoinEPcCHc | JoinEPbCHb | JoinEPaCHa | JoinEP0CH0 |
| 0x0A8 | AREA4Join_0 | R/W | 0x00 | JoinFIFO_Stat | | | | | JoinDMA | JoinCPU_Rd | JoinCPU_Wr |
| 0x0A9 | AREA4Join_1 | R/W | 0x00 | | | JoinEPeCHe | JoinEPdCHd | JoinEPcCHc | JoinEPbCHb | JoinEPaCHa | JoinEP0CH0 |
| 0x0AA | AREA5Join_0 | R/W | 0x00 | JoinFIFO_Stat | | | | | JoinDMA | JoinCPU_Rd | JoinCPU_Wr |
| 0x0AB | AREA5Join_1 | R/W | 0x00 | | | JoinEPeCHe | JoinEPdCHd | JoinEPcCHc | JoinEPbCHb | JoinEPaCHa | JoinEP0CH0 |
| 0x0AC | | | | | | | | | | | |
| 0x0AD | | | | | | | | | | | |
| 0x0AE | ClrAREAnJoin_0 | W | 0x00 | ClrJoinFIFO_Stat | | | | | ClrJoinDMA | ClrJoinCPU_Rd | ClrJoinCPU_Wr |
| 0x0AF | ClrAREAnJoin_1 | W | 0x00 | | | ClrJoinEPeCHe | ClrJoinEPdCHd | ClrJoinEPcCHc | ClrJoinEPbCHb | ClrJoinEPaCHa | ClrJoinEP0CH0 |

## 7.2 Device Register Map

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0B0 | ***D_SIE_IntStat*** | R/(W) | 0x00 | | ***NonJ*** | RcvSOF | DetectRESET | DetectSUSPEND | ChirpCmp | RestoreCmp | SetAddressCmp |
| 0x0B1 | | | 0xXX | | | | | | | | |
| 0x0B2 | | R/(W) | 0x00 | | | | | | | | |
| 0x0B3 | D_BulkIntStat | R/(W) | 0x00 | CBW_Cmp | CBW_LengthErr | CBW_Err | | CSW_Cmp | CSW_Err | | |
| 0x0B4 | D_EPrIntStat | R | 0x00 | D_AlarmIN_IntStat | D_AlarmOUT_IntStat | | D_EPeIntStat | D_EPdIntStat | D_EPcIntStat | D_EPbIntStat | D_EPaIntStat |
| 0x0B5 | D_EP0IntStat | R/(W) | 0x00 | DescriptorCmp | OUT_ShortACK | IN_TranACK | OUT_TranACK | IN_TranNAK | OUT_TranNAK | IN_TranErr | OUT_TranErr |
| 0x0B6 | D_EPaIntStat | R/(W) | 0x00 | | OUT_ShortACK | IN_TranACK | OUT_TranACK | IN_TranNAK | OUT_TranNAK | IN_TranErr | OUT_TranErr |
| 0x0B7 | D_EPbIntStat | R/(W) | 0x00 | | OUT_ShortACK | IN_TranACK | OUT_TranACK | IN_TranNAK | OUT_TranNAK | IN_TranErr | OUT_TranErr |
| 0x0B8 | D_EPcIntStat | R/(W) | 0x00 | | OUT_ShortACK | IN_TranACK | OUT_TranACK | IN_TranNAK | OUT_TranNAK | IN_TranErr | OUT_TranErr |
| 0x0B9 | D_EPdIntStat | R/(W) | 0x00 | | OUT_ShortACK | IN_TranACK | OUT_TranACK | IN_TranNAK | OUT_TranNAK | IN_TranErr | OUT_TranErr |
| 0x0BA | D_EPeIntStat | R/(W) | 0x00 | | OUT_ShortACK | IN_TranACK | OUT_TranACK | IN_TranNAK | OUT_TranNAK | IN_TranErr | OUT_TranErr |
| 0x0BB | | | 0xXX | | | | | | | | |
| 0x0BC | D_AlarmIN_IntStat_H | R/(W) | 0x00 | AlarmEP15IN | AlarmEP14IN | AlarmEP13IN | AlarmEP12IN | AlarmEP11IN | AlarmEP10IN | AlarmEP9IN | AlarmEP8IN |
| 0x0BD | D_AlarmIN_IntStat_L | R/(W) | 0x00 | AlarmEP7IN | AlarmEP6IN | AlarmEP5IN | AlarmEP4IN | AlarmEP3IN | AlarmEP2IN | AlarmEP1IN | |
| 0x0BE | D_AlarmOUT_IntStat_H | R/(W) | 0x00 | AlarmEP15OUT | AlarmEP14OUT | AlarmEP13OUT | AlarmEP12OUT | AlarmEP11OUT | AlarmEP10OUT | AlarmEP9OUT | AlarmEP8OUT |
| 0x0BF | D_AlarmOUT_IntStat_L | R/(W) | 0x00 | AlarmEP7OUT | AlarmEP6OUT | AlarmEP5OUT | AlarmEP4OUT | AlarmEP3OUT | AlarmEP2OUT | AlarmEP1OUT | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0C0 | ***D_SIE_IntEnb*** | R/W | 0x00 | | ***EnNonJ*** | EnRcvSOF | EnDetectRESET | EnDetectSUSPEND | EnChirpCmp | EnRestoreCmp | EnSetAddressCmp |
| 0x0C1 | | | 0xXX | | | | | | | | |
| 0x0C2 | | R/W | 0x00 | | | | | | | | |
| 0x0C3 | D_BulkIntEnb | R/W | 0x00 | EnCBW_Cmp | EnCBW_LengthErr | EnCBW_Err | | EnCSW_Cmp | EnCSW_Err | | |
| 0x0C4 | D_EPrIntEnb | R/W | 0x00 | EnD_AlarmIN_IntStat | EnD_AlarmOUT_IntStat | | EnD_EPeIntStat | EnD_EPdIntStat | EnD_EPcIntStat | EnD_EPbIntStat | EnD_EPaIntStat |
| 0x0C5 | D_EP0IntEnb | R/W | 0x00 | EnDescriptorCmp | EnOUT_ShortACK | EnIN_TranACK | EnOUT_TranACK | EnIN_TranNAK | EnOUT_TranNAK | EnIN_TranErr | EnOUT_TranErr |
| 0x0C6 | D_EPaIntEnb | R/W | 0x00 | | EnOUT_ShortACK | EnIN_TranACK | EnOUT_TranACK | EnIN_TranNAK | EnOUT_TranNAK | EnIN_TranErr | EnOUT_TranErr |
| 0x0C7 | D_EPbIntEnb | R/W | 0x00 | | EnOUT_ShortACK | EnIN_TranACK | EnOUT_TranACK | EnIN_TranNAK | EnOUT_TranNAK | EnIN_TranErr | EnOUT_TranErr |
| 0x0C8 | D_EPcIntEnb | R/W | 0x00 | | EnOUT_ShortACK | EnIN_TranACK | EnOUT_TranACK | EnIN_TranNAK | EnOUT_TranNAK | EnIN_TranErr | EnOUT_TranErr |
| 0x0C9 | D_EPdIntEnb | R/W | 0x00 | | EnOUT_ShortACK | EnIN_TranACK | EnOUT_TranACK | EnIN_TranNAK | EnOUT_TranNAK | EnIN_TranErr | EnOUT_TranErr |
| 0x0CA | D_EPeIntEnb | R/W | 0x00 | | EnOUT_ShortACK | EnIN_TranACK | EnOUT_TranACK | EnIN_TranNAK | EnOUT_TranNAK | EnIN_TranErr | EnOUT_TranErr |
| 0x0CB | | | 0xXX | | | | | | | | |
| 0x0CC | D_AlarmIN_IntEnb_H | R/W | 0x00 | EnAlarmEP15IN | EnAlarmEP14IN | EnAlarmEP13IN | EnAlarmEP12IN | EnAlarmEP11IN | EnAlarmEP10IN | EnAlarmEP9IN | EnAlarmEP8IN |
| 0x0CD | D_AlarmIN_IntEnb_L | R/W | 0x00 | EnAlarmEP7IN | EnAlarmEP6IN | EnAlarmEP5IN | EnAlarmEP4IN | EnAlarmEP3IN | EnAlarmEP2IN | EnAlarmEP1IN | |
| 0x0CE | D_AlarmOUT_IntEnb_H | R/W | 0x00 | EnAlarmEP15OUT | EnAlarmEP14OUT | EnAlarmEP13OUT | EnAlarmEP12OUT | EnAlarmEP11OUT | EnAlarmEP10OUT | EnAlarmEP9OUT | EnAlarmEP8OUT |
| 0x0CF | D_AlarmOUT_IntEnb_L | R/W | 0x00 | EnAlarmEP7OUT | EnAlarmEP6OUT | EnAlarmEP5OUT | EnAlarmEP4OUT | EnAlarmEP3OUT | EnAlarmEP2OUT | EnAlarmEP1OUT | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0D0 | D_NegoControl | R/W | 0x00 | DisBusDetect | EnAutoNego | InSUSPEND | DisableHS | SendWakeup | RestoreUSB | GoChirp | ActiveUSB |
| 0x0D1 | | | 0xXX | | | | | | | | |
| 0x0D2 | | | 0xXX | | | | | | | | |
| 0x0D3 | D_XcvrControl | R/W | 0x41 | TermSelect | XcvrSelect | | | | | OpMode [1:0] | |
| 0x0D4 | D_USB_Test | R/W | 0x00 | EnHS_Test | | | | Test_SE0_NAK | Test_J | Test_K | Test_Packet |
| 0x0D5 | | | 0xXX | | | | | | | | |
| 0x0D6 | D_EPnControl | W | 0xXX | AllForceNAK | EPrForceSTALL | | | | | | |
| 0x0D7 | | | 0xXX | | | | | | | | |
| 0x0D8 | D_BulkOnlyControl | R/W | 0x00 | AutoForceNAK_CBW | | | | | GoCBW_Mode | GoCSW_Mode | |
| 0x0D9 | D_BulkOnlyConfig | R/W | 0x00 | | | | EPeBulkOnly | EPdBulkOnly | EPcBulkOnly | EPbBulkOnly | EPaBulkOnly |
| 0x0DA | | | 0xXX | | | | | | | | |
| 0x0DB | | | 0xXX | | | | | | | | |
| 0x0DC | | | 0xXX | | | | | | | | |
| 0x0DD | | | 0xXX | | | | | | | | |
| 0x0DE | | | 0xXX | | | | | | | | |
| 0x0DF | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0E0 | D_EP0SETUP_0 | R | 0x00 | SETUP 0[7:0] | | | | | | | |
| 0x0E1 | D_EP0SETUP_1 | R | 0x00 | SETUP 1[7:0] | | | | | | | |
| 0x0E2 | D_EP0SETUP_2 | R | 0x00 | SETUP 2[7:0] | | | | | | | |
| 0x0E3 | D_EP0SETUP_3 | R | 0x00 | SETUP 3[7:0] | | | | | | | |
| 0x0E4 | D_EP0SETUP_4 | R | 0x00 | SETUP 4[7:0] | | | | | | | |
| 0x0E5 | D_EP0SETUP_5 | R | 0x00 | SETUP 5[7:0] | | | | | | | |
| 0x0E6 | D_EP0SETUP_6 | R | 0x00 | SETUP 6[7:0] | | | | | | | |
| 0x0E7 | D_EP0SETUP_7 | R | 0x00 | SETUP 7[7:0] | | | | | | | |
| 0x0E8 | D_USB_Address | R/(W) | 0x00 | SetAddress | USB_Address [6:0] | | | | | | |
| 0x0E9 | | | 0xXX | | | | | | | | |
| 0x0EA | D_SETUP_Control | R/W | 0x00 | | | | | | | | ProtectEP0 |
| 0x0EB | | | 0xXX | | | | | | | | |
| 0x0EC | | | 0xXX | | | | | | | | |
| 0x0ED | | | 0xXX | | | | | | | | |
| 0x0EE | D_FrameNumber_H | R | 0x80 | Fn_Invalid | | | | | FrameNumber [10:8] | | |
| 0x0EF | D_FrameNumber_L | R | 0x00 | FrameNumber [7:0] | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0F0 | D_EP0MaxSize | R/W | 0x00 | EP0MaxSize[6:3] | | | | | | | |
| 0x0F1 | D_EP0Control | R/W | 0x00 | INxOUT | | | | | | | ReplyDescriptor |
| 0x0F2 | D_EP0ControlIN | R/W | 0x00 | | EnShortPkt | | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x0F3 | D_EP0ControlOUT | R/W | 0x00 | AutoForceNAK | | | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x0F4 | | | 0xXX | | | | | | | | |
| 0x0F5 | | | 0xXX | | | | | | | | |
| 0x0F6 | | | 0xXX | | | | | | | | |
| 0x0F7 | | | 0x00 | | | | | | | | |
| 0x0F8 | D_EPaMaxSize_H | R/W | 0x00 | | | | | | EPaMaxSize[10:8] | | |
| 0x0F9 | D_EPaMaxSize_L | R/W | 0x00 | EPaMaxSize[7:0] | | | | | | | |
| 0x0FA | D_EPaConfig | R/W | 0x00 | INxOUT | IntEP_Mode | ISO | | EndpointNumber[3:0] | | | |
| 0x0FB | | | 0xXX | | | | | | | | |
| 0x0FC | D_EPaControl | R/W | 0x00 | AutoForceNAK | EnShortPkt | DisAF_NAK_Short | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x0FD | | | 0xXX | | | | | | | | |
| 0x0FE | | | 0xXX | | | | | | | | |
| 0x0FF | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x100 | D_EPbMaxSize_H | R/W | 0x00 | | | | | | EPbMaxSize[10:8] | | |
| 0x101 | D_EPbMaxSize_L | R/W | 0x00 | EPbMaxSize[7:0] | | | | | | | |
| 0x102 | D_EPbConfig | R/W | 0x00 | INxOUT | IntEP_Mode | ISO | | EndpointNumber[3:0] | | | |
| 0x103 | | | 0xXX | | | | | | | | |
| 0x104 | D_EPbControl | R/W | 0x00 | AutoForceNAK | EnShortPkt | DisAF_NAK_Short | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x105 | | | 0xXX | | | | | | | | |
| 0x106 | | | 0xXX | | | | | | | | |
| 0x107 | | | 0xXX | | | | | | | | |
| 0x108 | D_EPcMaxSize_H | R/W | 0x00 | | | | | | EPcMaxSize[10:8] | | |
| 0x109 | D_EPcMaxSize_L | R/W | 0x00 | EPcMaxSize[7:0] | | | | | | | |
| 0x10A | D_EPcConfig | R/W | 0x00 | INxOUT | IntEP_Mode | ISO | | EndpointNumber[3:0] | | | |
| 0x10B | | | 0xXX | | | | | | | | |
| 0x10C | D_EPcControl | R/W | 0x00 | AutoForceNAK | EnShortPkt | DisAF_NAK_Short | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x10D | | | 0xXX | | | | | | | | |
| 0x10E | | | 0xXX | | | | | | | | |
| 0x10F | | | 0xXX | | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x110 | D_EPdMaxSize_H | R/W | 0x00 | | | | | | EPdMaxSize[10:8] | | |
| 0x111 | D_EPdMaxSize_L | R/W | 0x00 | EPdMaxSize[7:0] | | | | | | | |
| 0x112 | D_EPdConfig | R/W | 0x00 | INxOUT | IntEP_Mode | ISO | | EndpointNumber[3:0] | | | |
| 0x113 | | | 0xXX | | | | | | | | |
| 0x114 | D_EPdControl | R/W | 0x00 | AutoForceNAK | EnShortPkt | DisAF_NAK_Short | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x115 | | | 0xXX | | | | | | | | |
| 0x116 | | | 0xXX | | | | | | | | |
| 0x117 | | | 0xXX | | | | | | | | |
| 0x118 | D_EPeMaxSize_H | R/W | 0x00 | | | | | | EPeMaxSize[10:8] | | |
| 0x119 | D_EPeMaxSize_L | R/W | 0x00 | EPeMaxSize[7:0] | | | | | | | |
| 0x11A | D_EPeConfig | R/W | 0x00 | INxOUT | IntEP_Mode | ISO | | EndpointNumber[3:0] | | | |
| 0x11B | | | 0xXX | | | | | | | | |
| 0x11C | D_EPeControl | R/W | 0x00 | AutoForceNAK | EnShortPkt | DisAF_NAK_Short | ToggleStat | ToggleSet | ToggleClr | ForceNAK | ForceSTALL |
| 0x11D | | | 0xXX | | | | | | | | |
| 0x11E | | | 0xXX | | | | | | | | |
| 0x11F | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x120 | D_DescAdrs_H | R/W | 0x00 | | | | DescAdrs[12:8] | | | | |
| 0x121 | D_DescAdrs_L | R/W | 0x00 | DescAdrs [7:0] | | | | | | | |
| 0x122 | D_DescSize_H | R/W | 0x00 | | | | | | | DescSize [9:8] | |
| 0x123 | D_DescSize_L | R/W | 0x00 | DescSize [7:0] | | | | | | | |
| 0x124 | | | 0xXX | | | | | | | | |
| 0x125 | | | 0xXX | | | | | | | | |
| 0x126 | D_EP_DMA_Ctrl | R/W | 0xXX | FIFO_Running | AutoEnShort | | | | | | |
| 0x127 | | | 0xXX | | | | | | | | |
| 0x128 | D_EnEP_IN_H | R/W | 0x00 | EnEP15IN | EnEP14IN | EnEP13IN | EnEP12IN | EnEP11IN | EnEP10IN | EnEP9IN | EnEP8IN |
| 0x129 | D_EnEP_IN_L | R/W | 0x00 | EnEP7IN | EnEP6IN | EnEP5IN | EnEP4IN | EnEP3IN | EnEP2IN | EnEP1IN | |
| 0x12A | D_EnEP_OUT_H | R/W | 0x00 | EnEP15OUT | EnEP14OUT | EnEP13OUT | EnEP12OUT | EnEP11OUT | EnEP10OUT | EnEP9OUT | EnEP8OUT |
| 0x12B | D_EnEP_OUT_L | R/W | 0x00 | EnEP7OUT | EnEP6OUT | EnEP5OUT | EnEP4OUT | EnEP3OUT | EnEP2OUT | EnEP1OUT | |
| 0x12C | D_EnEP_IN_ISO_H | R/W | 0x00 | EnEP15IN_ISO | EnEP14IN_ISO | EnEP13IN_ISO | EnEP12IN_ISO | EnEP11IN_ISO | EnEP10IN_ISO | EnEP9IN_ISO | EnEP8IN_ISO |
| 0x12D | D_EnEP_IN_ISO_L | R/W | 0x00 | EnEP7IN_ISO | EnEP6IN_ISO | EnEP5IN_ISO | EnEP4IN_ISO | EnEP3IN_ISO | EnEP2IN_ISO | EnEP1IN_ISO | |
| 0x12E | D_EnEP_OUT_ISO_H | R/W | 0x00 | EnEP15OUT_ISO | EnEP14OUT_ISO | EnEP13OUT_ISO | EnEP12OUT_ISO | EnEP11OUT_ISO | EnEP10OUT_ISO | EnEP9OUT_ISO | EnEP8OUT_ISO |
| 0x12F | D_EnEP_OUT_ISO_L | R/W | 0x00 | EnEP7OUT_ISO | EnEP6OUT_ISO | EnEP5OUT_ISO | EnEP4OUT_ISO | EnEP3OUT_ISO | EnEP2OUT_ISO | EnEP1OUT_ISO | |

For detailed information on the registers listed below, refer to Appendix D.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x130 | (Reserved) | | 0xXX | | | | | | | | |
| 0x131 | D_ModeControl | W | 0xXX | (Reserved) | (Reserved) | (Reserved) | SetAddressMode | (Reserved) | (Reserved) | (Reserved) | (Reserved) |

0x132–0x1FF are reserved.

## 7.3 Host Register Map

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x140 | H_SIE_IntStat_0 | R/(W) | 0x00 | | | | DetectCon | DetectDiscon | DetectRmtWkup | DetectDevChirpOK | DetectDevChirpNG |
| 0x141 | H_SIE_IntStat_1 | R/(W) | 0x00 | | | | | DisabledCmp | ResumeCmp | SuspendCmp | ResetCmp |
| 0x142 | | R/(W) | 0x00 | | | | | | | | |
| 0x143 | H_FrameIntStat | R/(W) | 0x00 | | | | | | PortErr | FrameNumOver | SOF |
| 0x144 | H_CHrIntStat | R | 0x00 | | | | H_CHeIntStat | H_CHdIntStat | H_CHcIntStat | H_CHbIntStat | H_CHaIntStat |
| 0x145 | H_CH0IntStat | R/(W) | 0x00 | TotalSizeCmp | TranACK | TranErr | ChangeCondition | | | CTL_SupportCmp | CTL_SupportStop |
| 0x146 | H_CHaIntStat | R/(W) | 0x00 | TotalSizeCmp | TranACK | TranErr | ChangeCondition | | | BO_SupportCmp | BO_SupportStop |
| 0x147 | H_CHbIntStat | R/(W) | 0x00 | TotalSizeCmp | TranACK | TranErr | ChangeCondition | | | | |
| 0x148 | H_CHcIntStat | R/(W) | 0x00 | TotalSizeCmp | TranACK | TranErr | ChangeCondition | | | | |
| 0x149 | H_CHdIntStat | R/(W) | 0x00 | TotalSizeCmp | TranACK | TranErr | ChangeCondition | | | | |
| 0x14A | H_CHeIntStat | R/(W) | 0x00 | TotalSizeCmp | TranACK | TranErr | ChangeCondition | | | | |
| 0x14B | | | 0xXX | | | | | | | | |
| 0x14C | | | 0xXX | | | | | | | | |
| 0x14D | | | 0xXX | | | | | | | | |
| 0x14E | | | 0xXX | | | | | | | | |
| 0x14F | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x150 | H_SIE_IntEnb_0 | R/W | 0x00 | | | | EnDetectCon | EnDetectDiscon | EnDetectRmtWkup | EnDetectDevChirpOK | EnDetectDevChirpNG |
| 0x151 | H_SIE_IntEnb_1 | R/W | 0x00 | | | | | EnDisabledCmp | EnResumeCmp | EnSuspendCmp | EnResetCmp |
| 0x152 | | R/W | 0x00 | | | | | | | | |
| 0x153 | H_FrameIntEnb | R/W | 0x00 | | | | | | EnPortErr | EnFrameNumOver | EnSOF |
| 0x154 | H_CHrIntEnb | R/W | 0x00 | | | | EnH_CHeIntStat | EnH_CHdIntStat | EnH_CHcIntStat | EnH_CHbIntStat | EnH_CHaIntStat |
| 0x155 | H_CH0IntEnb | R/W | 0x00 | EnTotalSizeCmp | EnTranACK | EnTranErr | EnChangeCondition | | | EnCTL_SupportCmp | EnCTL_SupportStop |
| 0x156 | H_CHaIntEnb | R/W | 0x00 | EnTotalSizeCmp | EnTranACK | EnTranErr | EnChangeCondition | | | EnBO_SupportCmp | EnBO_SupportStop |
| 0x157 | H_CHbIntEnb | R/W | 0x00 | EnTotalSizeCmp | EnTranACK | EnTranErr | EnChangeCondition | | | | |
| 0x158 | H_CHcIntEnb | R/W | 0x00 | EnTotalSizeCmp | EnTranACK | EnTranErr | EnChangeCondition | | | | |
| 0x159 | H_CHdIntEnb | R/W | 0x00 | EnTotalSizeCmp | EnTranACK | EnTranErr | EnChangeCondition | | | | |
| 0x15A | H_CHeIntEnb | R/W | 0x00 | EnTotalSizeCmp | EnTranACK | EnTranErr | EnChangeCondition | | | | |
| 0x15B | | | 0xXX | | | | | | | | |
| 0x15C | | | 0xXX | | | | | | | | |
| 0x15D | | | 0xXX | | | | | | | | |
| 0x15E | | | 0xXX | | | | | | | | |
| 0x15F | | | 0xXX | | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x160 | H_NegoControl_0 | R/W | 0x1X | AutoModeCancel | HostState[2:0] | | | AutoMode[3:0] | | | |
| 0x161 | | | 0xXX | | | | | | | | |
| 0x162 | H_NegoControl_1 | R/W | 0x10 | | | PortSpeed[1:0] | | | | DisChirpFinish | RmtWkupDetEnb |
| 0x163 | | | 0xXX | | | | | | | | |
| 0x164 | H_USB_Test | R/W | 0x00 | EnHS_Test | | | Test_Force_Enable | Test_SE0_NAK | Test_J | Test_K | Test_Packet |
| 0x165 | | | 0xXX | | | | | | | | |
| 0x166 | | | 0xXX | | | | | | | | |
| 0x167 | | | 0xXX | | | | | | | | |
| 0x168 | | | 0xXX | | | | | | | | |
| 0x169 | | | 0xXX | | | | | | | | |
| 0x16A | | | 0xXX | | | | | | | | |
| 0x16B | | | 0xXX | | | | | | | | |
| 0x16C | | | 0xXX | | | | | | | | |
| 0x16D | | | 0xXX | | | | | | | | |
| 0x16E | | | 0xXX | | | | | | | | |
| 0x16F | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x170 | H_CH0SETUP_0 | R/W | 0x00 | SETUP 0[7:0] | | | | | | | |
| 0x171 | H_CH0SETUP_1 | R/W | 0x00 | SETUP 1[7:0] | | | | | | | |
| 0x172 | H_CH0SETUP_2 | R/W | 0x00 | SETUP 2[7:0] | | | | | | | |
| 0x173 | H_CH0SETUP_3 | R/W | 0x00 | SETUP 3[7:0] | | | | | | | |
| 0x174 | H_CH0SETUP_4 | R/W | 0x00 | SETUP 4[7:0] | | | | | | | |
| 0x175 | H_CH0SETUP_5 | R/W | 0x00 | SETUP 5[7:0] | | | | | | | |
| 0x176 | H_CH0SETUP_6 | R/W | 0x00 | SETUP 6[7:0] | | | | | | | |
| 0x177 | H_CH0SETUP_7 | R/W | 0x00 | SETUP 7[7:0] | | | | | | | |
| 0x178 | | | 0xXX | | | | | | | | |
| 0x179 | | | 0xXX | | | | | | | | |
| 0x17A | | | 0xXX | | | | | | | | |
| 0x17B | | | 0xXX | | | | | | | | |
| 0x17C | | | 0xXX | | | | | | | | |
| 0x17D | | | 0xXX | | | | | | | | |
| 0x17E | H_FrameNumber_H | R | 0x07 | | | | | | FrameNumber[10:8] | | |
| 0x17F | H_FrameNumber_L | R | 0xFF | FrameNumber[7:0] | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x180 | H_CH0Config_0 | R/W | 0x00 | ACK_Cnt[3:0] | | | | SpeedMode[1:0] | | Toggle | TranGo |
| 0x181 | H_CH0Config_1 | R/W | 0x00 | TID[1:0] | | | | | | | |
| 0x182 | | | 0xXX | | | | | | | | |
| 0x183 | H_CH0MaxPktSize | R/W | 0x00 | MaxPktSize[6:0] | | | | | | | |
| 0x184 | | | 0xXX | | | | | | | | |
| 0x185 | | | 0xXX | | | | | | | | |
| 0x186 | H_CH0TotalSize_H | R/W | 0x00 | TotalSize[15:8] | | | | | | | |
| 0x187 | H_CH0TotalSize_L | R/W | 0x00 | TotalSize[7:0] | | | | | | | |
| 0x188 | H_CH0HubAdrs | R/W | 0x00 | HubAdrs[3:0] | | | | | Port[2:0] | | |
| 0x189 | H_CH0FuncAdrs | R/W | 0x00 | FuncAdrs[3:0] | | | | EP_Number[3:0] | | | |
| 0x18A | | | 0xXX | | | | | | | | |
| 0x18B | H_CTL_SupportControl | R/W | 0x00 | | | CTL_SupportState[1:0] | | | | | CTL_SupportGo |
| 0x18C | | | 0xXX | | | | | | | | |
| 0x18D | | | 0xXX | | | | | | | | |
| 0x18E | H_CH0ConditionCode | R | 0x00 | | ConditonCode[2:0] | | | | | | |
| 0x18F | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x190 | H_CHaConfig_0 | R/W | 0x00 | ACK_Cnt[3:0] | | | | SpeedMode[1:0] | | Toggle | TranGo |
| 0x191 | H_CHaConfig_1 | R/W | 0x00 | TID[1:0] | | | | AutoZerolen | | | TotalSizeFree |
| 0x192 | H_CHaMaxPktSize_H | R/W | 0x00 | | | | | | (MaxPktSize[10]) | MaxPktSize[9:8] | |
| 0x193 | H_CHaMaxPktSize_L | R/W | 0x00 | MaxPktSize[7:0] | | | | | | | |
| 0x194 | H_CHaTotalSize_HH | R/W | 0x00 | TotalSize[31:24] | | | | | | | |
| 0x195 | H_CHaTotalSize_HL | R/W | 0x00 | TotalSize[23:16] | | | | | | | |
| 0x196 | H_CHaTotalSize_LH | R/W | 0x00 | TotalSize[15:8] | | | | | | | |
| 0x197 | H_CHaTotalSize_LL | R/W | 0x00 | TotalSize[7:0] | | | | | | | |
| 0x198 | H_CHaHubAdrs | R/W | 0x00 | HubAdrs[3:0] | | | | | Port[2:0] | | |
| 0x199 | H_CHaFuncAdrs | R/W | 0x00 | FuncAdrs[3:0] | | | | EP_Number[3:0] | | | |
| 0x19A | H_CHaBO_SupportCtl | R/W | 0x00 | | | BO_TransportState[1:0] | | | | | BO_SupportGo |
| 0x19B | H_CHaBO_CSW_RcvSize | R | 0x00 | | | | | CSW_RcvDataSize[3:0] | | | |
| 0x19C | H_CHaBO_OUT_EP_Ctl | R/W | 0x00 | | | | OUT_Toggle | OUT_EP_Number[3:0] | | | |
| 0x19D | H_CHaBO_IN_EP_Ctl | R/W | 0x00 | | | | IN_Toggle | IN_EP_Number[3:0] | | | |
| 0x19E | H_CHaConditionCode | R | 0x00 | | ConditonCode[2:0] | | | | | | |
| 0x19F | | | 0xXX | | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1A0 | H_CHbConfig_0 | R/W | 0x00 | ACK_Cnt[3:0] | | | | SpeedMode[1:0] | | Toggle | TranGo |
| 0x1A1 | H_CHbConfig_1 | R/W | 0x00 | TID[1:0] | | TranType[1:0] | | AutoZerolen | | | TotalSizeFree |
| 0x1A2 | H_CHbMaxPktSize_H | R/W | 0x00 | | | | | | (MaxPktSize[10]) | MaxPktSize[9:8] | |
| 0x1A3 | H_CHbMaxPktSize_L | R/W | 0x00 | MaxPktSize[7:0] | | | | | | | |
| 0x1A4 | H_CHbTotalSize_HH | R/W | 0x00 | TotalSize[31:24] | | | | | | | |
| 0x1A5 | H_CHbTotalSize_HL | R/W | 0x00 | TotalSize[23:16] | | | | | | | |
| 0x1A6 | H_CHbTotalSize_LH | R/W | 0x00 | TotalSize[15:8] | | | | | | | |
| 0x1A7 | H_CHbTotalSize_LL | R/W | 0x00 | TotalSize[7:0] | | | | | | | |
| 0x1A8 | H_CHbHubAdrs | R/W | 0x00 | HubAdrs[3:0] | | | | | Port[2:0] | | |
| 0x1A9 | H_CHbFuncAdrs | R/W | 0x00 | FuncAdrs[3:0] | | | | EP_Number[3:0] | | | |
| 0x1AA | H_CHbInterval_H | R/W | 0x00 | | | | | | Interval[10:8] | | |
| 0x1AB | H_CHbInterval_L | R/W | 0x00 | Interval[7:0] | | | | | | | |
| 0x1AC | | | 0xXX | | | | | | | | |
| 0x1AD | | | 0xXX | | | | | | | | |
| 0x1AE | H_CHbConditionCode | R | 0x00 | | ConditonCode[2:0] | | | | | | |
| 0x1AF | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1B0 | H_CHcConfig_0 | R/W | 0x00 | ACK_Cnt[3:0] | | | | SpeedMode[1:0] | | Toggle | TranGo |
| 0x1B1 | H_CHcConfig_1 | R/W | 0x00 | TID[1:0] | | TranType[1:0] | | AutoZerolen | | | TotalSizeFree |
| 0x1B2 | H_CHcMaxPktSize_H | R/W | 0x00 | | | | | | (MaxPktSize[10]) | MaxPktSize[9:8] | |
| 0x1B3 | H_CHcMaxPktSize_L | R/W | 0x00 | MaxPktSize[7:0] | | | | | | | |
| 0x1B4 | H_CHcTotalSize_HH | R/W | 0x00 | TotalSize[31:24] | | | | | | | |
| 0x1B5 | H_CHcTotalSize_HL | R/W | 0x00 | TotalSize[23:16] | | | | | | | |
| 0x1B6 | H_CHcTotalSize_LH | R/W | 0x00 | TotalSize[15:8] | | | | | | | |
| 0x1B7 | H_CHcTotalSize_LL | R/W | 0x00 | TotalSize[7:0] | | | | | | | |
| 0x1B8 | H_CHcHubAdrs | R/W | 0x00 | HubAdrs[3:0] | | | | | Port[2:0] | | |
| 0x1B9 | H_CHcFuncAdrs | R/W | 0x00 | FuncAdrs[3:0] | | | | EP_Number[3:0] | | | |
| 0x1BA | H_CHcInterval_H | R/W | 0x00 | | | | | | Interval[10:8] | | |
| 0x1BB | H_CHcInterval_L | R/W | 0x00 | Interval[7:0] | | | | | | | |
| 0x1BC | | | 0xXX | | | | | | | | |
| 0x1BD | | | 0xXX | | | | | | | | |
| 0x1BE | H_CHcConditionCode | R | 0x00 | | ConditonCode[2:0] | | | | | | |
| 0x1BF | | | 0xXX | | | | | | | | |

The registers shown in ***bold face italic*** can be read and written even in the SLEEP state.

All other registers can be read and written in the ACTIVE state.

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1C0 | H_CHdConfig_0 | R/W | 0x00 | ACK_Cnt[3:0] | | | | SpeedMode[1:0] | | Toggle | TranGo |
| 0x1C1 | H_CHdConfig_1 | R/W | 0x00 | TID[1:0] | | TranType[1:0] | | AutoZerolen | | | TotalSizeFree |
| 0x1C2 | H_CHdMaxPktSize_H | R/W | 0x00 | | | | | | (MaxPktSize[10]) | MaxPktSize[9:8] | |
| 0x1C3 | H_CHdMaxPktSize_L | R/W | 0x00 | MaxPktSize[7:0] | | | | | | | |
| 0x1C4 | H_CHdTotalSize_HH | R/W | 0x00 | TotalSize[31:24] | | | | | | | |
| 0x1C5 | H_CHdTotalSize_HL | R/W | 0x00 | TotalSize[23:16] | | | | | | | |
| 0x1C6 | H_CHdTotalSize_LH | R/W | 0x00 | TotalSize[15:8] | | | | | | | |
| 0x1C7 | H_CHdTotalSize_LL | R/W | 0x00 | TotalSize[7:0] | | | | | | | |
| 0x1C8 | H_CHdHubAdrs | R/W | 0x00 | HubAdrs[3:0] | | | | | Port[2:0] | | |
| 0x1C9 | H_CHdFuncAdrs | R/W | 0x00 | FuncAdrs[3:0] | | | | EP_Number[3:0] | | | |
| 0x1CA | H_CHdInterval_H | R/W | 0x00 | | | | | | Interval[10:8] | | |
| 0x1CB | H_CHdInterval_L | R/W | 0x00 | Interval[7:0] | | | | | | | |
| 0x1CC | | | 0xXX | | | | | | | | |
| 0x1CD | | | 0xXX | | | | | | | | |
| 0x1CE | H_CHdConditionCode | R | 0x00 | | ConditonCode[2:0] | | | | | | |
| 0x1CF | | | 0xXX | | | | | | | | |

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1D0 | H_CHeConfig_0 | R/W | 0x00 | ACK_Cnt[3:0] | | | | SpeedMode[1:0] | | Toggle | TranGo |
| 0x1D1 | H_CHeConfig_1 | R/W | 0x00 | TID[1:0] | | TranType[1:0] | | AutoZerolen | | | TotalSizeFree |
| 0x1D2 | H_CHeMaxPktSize_H | R/W | 0x00 | | | | | | (MaxPktSize[10]) | MaxPktSize[9:8] | |
| 0x1D3 | H_CHeMaxPktSize_L | R/W | 0x00 | MaxPktSize[7:0] | | | | | | | |
| 0x1D4 | H_CHeTotalSize_HH | R/W | 0x00 | TotalSize[31:24] | | | | | | | |
| 0x1D5 | H_CHeTotalSize_HL | R/W | 0x00 | TotalSize[23:16] | | | | | | | |
| 0x1D6 | H_CHeTotalSize_LH | R/W | 0x00 | TotalSize[15:8] | | | | | | | |
| 0x1D7 | H_CHeTotalSize_LL | R/W | 0x00 | TotalSize[7:0] | | | | | | | |
| 0x1D8 | H_CHeHubAdrs | R/W | 0x00 | HubAdrs[3:0] | | | | | Port[2:0] | | |
| 0x1D9 | H_CHeFuncAdrs | R/W | 0x00 | FuncAdrs[3:0] | | | | EP_Number[3:0] | | | |
| 0x1DA | H_CHeInterval_H | R/W | 0x00 | | | | | | Interval[10:8] | | |
| 0x1DB | H_CHeInterval_L | R/W | 0x00 | Interval[7:0] | | | | | | | |
| 0x1DC | | | 0xXX | | | | | | | | |
| 0x1DD | | | 0xXX | | | | | | | | |
| 0x1DE | H_CHeConditionCode | R | 0x00 | | ConditonCode[2:0] | | | | | | |
| 0x1DF | | | 0xXX | | | | | | | | |

For details of the following registers, see “Appendix C.”

| Address | Register Name | R/W | Reset | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1F4 | (Reserved) | | 0xXX | | | | | | | | |
| 0x1F5 | H_Protect | R/W | 0x00 | | | | | PortSpeedWrEnb | | TranEnb[1:0] | |
| 0x1F6 | H_Monitor | R | 0x00 | | | | | | | | TranRunning |
| 0x1F7 | (Reserved) | | 0xXX | | | | | | | | |

The addresses 0x1E0–0x1F3 and 0x1F8–0x1FF are reserved.

## 7.4 Detailed Description of Device/Host Shared Registers

### 7.4.1 000h *MainIntStat (Main Interrupt Status)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 000h | ***MainIntStat*** | R | 7: ***USB_DeviceIntStat*** | 0: None | 1: USB Device Interrupts | 00h |
| | | | R | 6: ***USB_HostIntStat*** | 0: None | 1: USB Host Interrupts | |
| | | | R | 5: CPU_IntStat | 0: None | 1: CPU Interrupts | |
| | | | R | 4: FIFO_IntStat | 0: None | 1: FIFO Interrupts | |
| | | | | 3: | 0: | 1: MediaFIFO Interrupts | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R (W) | 0: ***FinishedPM*** | 0: None | 1:Detect FinishedPM | |

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. That way, it is possible to trace back to the bit that indicates the reason that caused the interrupt to occur. 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 that 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 was 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 USB_DeviceIntStat**

This bit indicates the cause of interrupt indirectly.

If the USB_DeviceIntStat register has the cause of interrupt and the bit corresponding to that interrupt cause in the USB_DeviceIntEnb register is enabled, this bit is set to 1. This bit is effective even during Sleep.

**Bit6 USB_HostIntStat**

This bit indicates the cause of interrupt indirectly.

If the USB_HostIntStat register has the cause of interrupt and the bit corresponding to that interrupt cause in the USB_HostIntEnb register is enabled, this bit is set to 1. This bit is effective even during Sleep.

**Bit5 CPU_IntStat**

This bit indicates the cause of interrupt indirectly.

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

**Bit4 FIFO_IntStat**

This bit indicates the cause of interrupt indirectly.

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

**Bits3-1 Reserved**

**Bit0 FinishedPM**

This bit indicates the cause of interrupt directly.

If GoSLEEP or GoActive is set by the PM_Control register and the instructed state is reached, this bit is set to 1. This bit is effective even during Sleep.

### 7.4.2 001h *USB_DeviceIntStat (USB Device Interrupt Status)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 001h | ***USB_DeviceIntStat*** | R (W) | 7: ***VBUS_Changed*** | 0: None | 1: VBUS is Changed | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R | 5: ***D_SIE_IntStat*** | 0: None | 1: SIE Interrupts | |
| | | | R | 4: D_BulkIntStat | 0: None | 1: Bulk Interrupts | |
| | | | R (W) | 3: RcvEP0SETUP | 0: None | 1: Receive EP0 SETUP | |
| | | | | 2: | 0: | 1: | |
| | | | R | 1: D_EP0IntStat | 0: None | 1: EP0 Interrupts | |
| | | | R | 0: D_EPrIntStat | 0: None | 1: EPr Interrupts | |

This register shows the device-related interrupts.

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. That way, it is possible to trace back to the bit that indicates the reason that caused the interrupt to occur. 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 that the cause of an interrupt indicated by a bit can be cleared by writing 1 to that bit.

**Bit7 VBUS_Changed**

This bit indicates the cause of interrupt directly.

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

Check the VBUS bit in the D_USB_Status register to confirm the VBUS status. If VBUS = 0, it means that the cable is removed. This bit is effective even during Sleep.

**Bit6 Reserved**

**Bit5 D_SIE_IntStat**

This bit indicates the cause of interrupt indirectly.

If the D_SIE_IntStat register has the cause of interrupt and the bit corresponding to that interrupt cause in the D_SIE_IntEnb register is enabled, this bit is set to 1. This bit is effective even during Sleep.

**Bit4 D_BulkIntStat**

This bit indicates the cause of interrupt indirectly.

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

**Bit3 RcvEP0SETUP**

This bit indicates the cause of interrupt directly.

When the setup stage for a control transfer is complete and the received data is stored in the D_EP0Setup_0 through D_EP0Setup_7 registers, this bit is set to 1. At the same time, the ForceSTALL bit in the D_EP0ControlIN, D_EP0ControlOUT registers is cleared to 0, and the ForceNAK and ToggleStat bits in the D_EP0ControlIN, D_EP0ControlOUT registers and the ProtectEP0 bit in the D_SETUP Control register are set to 1, all automatically. SetAddress() requests are automatically responded by the AutoSetAddress function, and this status is not set.

**Bit2 Reserved**

**Bit1 D_EP0IntStat**

This bit indicates the cause of interrupt indirectly.

If the D_EP0IntStat register has the cause of interrupt and the bit corresponding to that interrupt cause in the D_EP0IntEnb register is enabled, this bit is set to 1.

**Bit0 D_EPrIntStat**

This bit indicates the cause of interrupt indirectly.

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

### 7.4.3 002h *USB_HostIntStat (USB Host Interrupt Status)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 002h | ***USB_HostIntStat*** | R (W) | 7: ***VBUS_Err*** | 0: None | 1: VBUS Error | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R | 5: H_SIE_IntStat1 | 0: None | 1: SIE Interrupts1 | |
| | | | R | 4: H_SIE_IntStat0 | 0: None | 1: SIE Interrupts0 | |
| | | | R | 3: H_FrameIntStat | 0: None | 1: Frame Interrupts | |
| | | | | 2: | 0: | 1: | |
| | | | R | 1: H_CH0IntStat | 0: None | 1: CH0 Interrupts | |
| | | | R | 0: H_CHrIntStat | 0: None | 1: CHr Interrupts | |

This register shows the host-related interrupts.

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. That way, it is possible to trace back to the bit that indicates the reason that caused the interrupt to occur. 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 that the cause of an interrupt indicated by a bit can be cleared by writing 1 to that bit.

**Bit7** ***VBUS_Err***

This bit indicates the cause of interrupt directly. This bit is effective even during Sleep.

When a VBUS error signal (high to low-going edge) is input from an external VBUS power switch connected to the VBUSFLG pin externally to the chip, this bit is set to 1.

Check the VBUS_State bit in the H_USB_Status register to confirm the VBUSFLG pin state.

The above-mentioned error signal differs with specifications of an externally connected power switch, so consult specifications of the power switch used in your system.

**Bit6** **Reserved**

**Bit5** **H_SIE_IntStat1**

This bit indicates the cause of interrupt indirectly.

If the H_SIE_IntStat1 register has the cause of interrupt and the bit corresponding to that interrupt cause in the H_SIE_IntEnb1 register is enabled, this bit is set to 1.

**Bit4** **H_SIE_IntStat0**

This bit indicates the cause of interrupt indirectly.

If the H_SIE_IntStat0 register has the cause of interrupt and the bit corresponding to that interrupt cause in the H_SIE_IntEnb0 register is enabled, this bit is set to 1.

**Bit3** **H_FrameIntStat**

This bit indicates the cause of interrupt indirectly.

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

**Bit2** **Reserved**

**Bit1** **H_CH0IntStat**

This bit indicates the cause of interrupt indirectly.

If the has the cause of interrupt H_CH0IntStat register and the bit corresponding to that interrupt cause in the H_CH0IntEnb register is enabled, this bit is set to 1.

**Bit0** **H_CHrIntStat**

This bit indicates the cause of interrupt indirectly.

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

### 7.4.4 003h CPU_IntStat (CPU Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 003h | CPU_IntStat | R (W) | 7: RAM_RdCmp | 0: None | 1: RAM Read Complete | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R (W) | 1: DMA_CountUp | 0: None | 1: DMA Counter Overflow | |
| | | | R (W) | 0: DMA_Cmp | 0: None | 1: DMA Complete | |

This register shows the interrupts associated with the CPU interface.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 RAM_RdCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when the data placed in the RAM_Rd_XX after being read from the RAM is prepared in the RAM_Rd function.

**Bits6-2 Reserved**

**Bit1 DMA_CountUp**

This bit indicates the cause of interrupt directly.

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

**Bit0 DMA_Cmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when the DMA transfer in progress is stopped or when processing for termination of transfer is complete after completion of a specified number of transfers.

### 7.4.5 004h FIFO_IntStat (FIFO Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 004h | FIFO_IntStat | R (W) | 7: FIFO_DMA_Cmp | 0: None | 1: DMA Compete on FIFO | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R (W) | 2: FIFO_NotEmpty | 0: None | 1: FIFO NotEmpty | |
| | | | R (W) | 1: FIFO_Full | 0: None | 1: FIFO Full | |
| | | | R (W) | 0: FIFO_Empty | 0: None | 1: FIFO Empty | |

This register shows the interrupts associated with the FIFO.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bits7 FIFO_DMA_Cmp**

This bit indicates the cause of the interrupt.

If while HostDeviceSel.HOSTxDEVICE = 0 the endpoint joined to DMA is directed for IN, this bit is set to 1 when the FIFO is emptied after a DMA transfer finishes. If the endpoint joined to DMA is directed for OUT, this bit is set to 1 when a DMA transfer finishes.

If while HostDeviceSel.HOSTxDEVICE = 1 the channel joined to DMA is directed for OUT, this bit is set to 1 when the FIFO is emptied after an IDE transfer. If the channel joined to DMA is directed for IN, this bit is set to 1 when an IDE transfer finishes.

**Bits6-3 Reserved**

**Bit2 FIFO_NotEmpty**

This bit indicates the cause of the interrupt.

This bit is set to 1 when data in the FIFO area for the relevant channel is detected as present while AREAn{n=0-5}Join_0.JoinFIFO_Stat bit = 1.

**Bits1 FIFO_Full**

This bit indicates the cause of the interrupt.

This bit is set to 1 when the FIFO area for the relevant channel is found to be full while AREAn{n=0-5}Join_0.JoinFIFO_Stat bit = 1.

**Bit0 FIFO_Empty**

This bit indicates the cause of the interrupt.

This bit is set to 1 when the FIFO area for the relevant channel is found to be empty while AREAn{n=0-5}Join_0.JoinFIFO_Stat bit = 1.

### 7.4.6 008h *MainIntEnb (Main Interrupt Enable)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 008 | ***MainIntEnb*** | R / W | 7: ***EnUSB_DeviceIntStat*** | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: ***EnUSB_HostIntStat*** | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnCPU_IntStat | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnFIFO _IntStat | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1 | |
| | | | R / W | 0: ***EnFinishedPM*** | 0: Disable | 1: Enable | |

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

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

The EnUSB_DeviceIntStat, EnUSB_HostIntStat, and EnFinishedPM bits are effective even during Sleep.

### 7.4.7 009h *USB_DeviceIntEnb (Device Interrupt Enable)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 009 | ***USB_DeviceIntEnb*** | R / W | 7: ***EnVBUS_Changed*** | 0: Disable | 1: Enable | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R / W | 5: ***EnD_SIE_IntStat*** | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnD_BulkIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnRcvEP0SETUP | 0: Disable | 1: Enable | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: EnD_EP0IntStat | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnD_EPrIntStat | 0: Disable | 1: Enable | |

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

The EnVBUS_Changed and EnD_SIE_IntStat bits are effective even during Sleep.

### 7.4.8 00Ah *USB_HostIntEnb (Host Interrupt Enable)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 00Ah | ***USB_HostIntEnb*** | R / W | 7: ***EnVBUS_Err*** | 0: Disable | 1: Enable | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R / W | 5: EnH_SIE_IntStat1 | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnH_SIE_IntStat0 | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnH_FrameIntStat | 0: Disable | 1: Enable | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: EnH_CH0IntStat | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnH_CHrIntStat | 0: Disable | 1: Enable | |

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

The EnVBUS_Err bit are effective even during Sleep.

### 7.4.9 00Bh CPU_IntEnb (CPU Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 00Bh | CPU_IntEnb | R / W | 7: EnRAM_RdCmp | 0: Disable | 1: Enable | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: EnDMA_CountUp | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnDMA_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 accommodated in the CPU_IntStat register.

### 7.4.10 00Ch FIFO_IntEnb (FIFO Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 00Ch | FIFO_IntEnb | R / W | 7: FIFO_DMA_Cmp | 0: Disable | 1: Enable | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: FIFO_NotEmpty | 0: Disable | 1: Enable | |
| | | | R / W | 1: FIFO_Full | 0: Disable | 1: Enable | |
| | | | R / W | 0: FIFO_Empty | 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 accommodated in the FIFO_IntStat register.

### 7.4.11 010h *RevisionNum (Revision Number)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 010h | ***RevisionNum*** | R | 7: ***RevisionNum [7]*** | Revision Number | 10h |
| | | | | 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 of the LSI stipulated herein. This register can be accessed even during Sleep.

The revision number relating to the current specifications of the LSI is 0x10.

### 7.4.12 011h *ChipReset (Chip Reset)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 011h | ***ChipReset*** | R/W | 7: ***ResetMTM*** | 0: None | 1: MTM Reset | 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.

**Bits7 Reserved**

Setting this bit to 1 initializes the transceiver macro (MTM) of the LSI.

To deassert the reset, clear this bit to 0.

**Bits6-1 Reserved**

**Bit0 AllReset**

This bit resets the entire circuit of the LSI. It works 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 a write to this register is attempted for other than rsesetting the LSI in violation of A.C. characteristics, the LSI may operate erratically.

### 7.4.13 012h *PM_Control (Power Management Control)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 012h | ***PM_Control*** | R / W | 7: ***GoSLEEP*** | 0: Do nothing | 1: Go to SLEEP | 00h |
| | | | R / W | 6: ***GoACTIVE*** | 0: Do nothing | 1: Go to ACTIVE | |
| | | | W | 5: ***GoCPU_Cut*** | 0: Do nothing | 1: Go to CPU Cut mode | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R | 1: ***PM_State[1]*** | Power Management State<br>00: SLEEP, 01: (SNOOZE), 11: ACTIVE | | |
| | | | | 0: ***PM_State[0]*** | | | |

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.

**Bit7 GoSLEEP**

This bit causes the LSI to start shifting to Sleep state from Active state.

When this bit is set to 1 during the Active state, the LSI first turns off the PLL and then the oscillator, thereby shifting to Sleep state.

Upon completion of a state transition no matter which state it has occurred from, this bit is automatically cleared and at the same time, the MainIntStat.FinishedPM bit is set.

**Bit6 GoActive**

This bit causes the LSI to start shifting to Active state from Sleep state.

When this bit is set to 1 during the Sleep state, the LSI turns on the oscillator and after the oscillator's oscillation stabilization time (set by WakeupTim_H,L) has elapsed, turns on the PLL and after the PLL480 oscillation stabilization time (approx. 250 µs) has elapsed, shifts to the Active state.

Upon completion of a state transition no matter which state it has occurred from, this bit is automatically cleared and at the same time, the MainIntStat.FinishedPM bit is set.

**Bit5 GoCPU_Cut**

Setting this bit in the SLEEP state places the LSI into CPU_Cut mode, which further reduces chip current consumption.

When this bit is set by writing 1 after the SLEEP state is fully entered, all input pins of the CPU interface except the XCS pin are turned off from the initial IC stage on when this CPU write state terminates. Consequently, even when signal lines except XCS are driven high or low, chip power consumption can be reduced to the absolute minimum necessary because the initial stage drivers of the CPU interface are turned off.

To resume from CPU_Cut mode, perform a dummy read of this register. Keep in mind, however, that the data read out at this time is indeterminate. Note also that since this return operation is performed at the same time a dummy read finishes, always make sure that XCS is negated once

(pulled low and then back high). For general CPUs, such a XCS negation can be achieved by accessing any address space other than those allocated to this LSI.

* The LSI stipulated herein has the XINT signal masked not to be asserted during Sleep by an interrupt status that cannot be accessed during Sleep (hereafter 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 Sleep, the firmware should execute the processing described below.

<Before entering Sleep>

Process the synchronous status to clear it (–IntStat).

Disable the synchronous status (–IntEnb).

<After exiting Sleep>

Clear the synchronous status (–IntStat).

Re-enable the synchronous status (–IntEnb).

**Bits4-2 Reserved**

**Bits1-0 PM_State [1:0]**

Indicates the state of power mode.

00: Sleep state (OSC off, PLL off)

01: (Snooze state) (OSC on, PLL off)

11: Active state (OSC on, PLL on)

Note that this state is unstable during a transition period from the time PM_Control.GoSLEEP/GoACTIVE is set until the MainIntStat.FinishedPM interrupt status is set. Do not refer to this state during this interval.

### 7.4.14 014h *WakeupTim_H (Wakeup Time High)*

### 7.4.15 015h *WakeupTim_L (Wakeup Time Low)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 014h -015h | ***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 waited for when the LSI returns from the Sleep state to the Snooze state. These registers can be accessed even during Sleep.

When the PM_Control.GoActive bit is set by writing 1 during 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 is complete 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 with the resonator, oscillator cell, circuit board, and load capacitance. If the LSI needs to be dropped into 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 + PLL stabilization time (less than 250 µs)

### 7.4.16 016h *H_USB_Control (Host USB Control )*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 016h | ***H_USB_Control*** | R / W | 7: ***VBUS_Enb*** | 0: Disable | 1: 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 is used to set the operations associated with the host.

This register is effective even in the SLEEP state.

**Bit7 *VBUS_Enb***

This bit sets the VBUSEN pin (output) state. The default output state of this pin is low.

**Bits6-0 Reserved**

### 7.4.17 017h *H_XcvrControl (Host Xcvr Control)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 017h | ***H_XcvrControl*** | R / W | 7: ***TermSelect*** | 0: HS Termination | 1: FS Termination | 91h |
| | | | R / W | 6: ***RemoveRPD*** | 0: RPD ON | 1: RPD OFF | |
| | | | R / W | 4: ***XcvrSelect[1]*** | XcvrSelect[1:0] | | |
| | | | R / W | 4: ***XcvrSelect[0]*** | | | |
| | | | | 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 host transceiver macro.

This register is effective even in the SLEEP state.

**Bit7 TermSelect**

This bit selects HS or FS termination to enable it.

Do not set this bit manually because the operation mode is automatically set to H_NegoControl_0.AutoMode by the hardware.

**Bit6 RemoveRPD**

This bit turns on/off the internal pulldown resistors for DP_A aand DM_A that are host data lines.

0: RPD is turned on

1: RPD is turned off

Normally, use this bit as “0” (on). In particular, always use "0” (on) during USB host operation (including SUSPEND). If you use values other than “0” the charcteristics of the host data line changes and it may cause the USB to operate erratically.

**Bits5-4 XcvrSelect[1:0]**

These bits select the HS, FS, or LS transceiver to enable it.

00: High Speed transceiver

01: Full Speed transceiver

10: Reserved

11: Low Speed transceiver

Do not set this bit manually because the operation mode is automatically set to H_NegoControl_0.AutoMode by the hardware.

**Bits3-2 Reserved**

**Bits1-0 OpMode**

These bits set the operation mode of the HTM.

Do not set this bit manually because the operation mode is automatically set to H_NegoControl_0.AutoMode by the hardware.

However, if the signal line change status of the host port is to be detected by state other than the ACT_HOST state, refer to the section on signal line change status (6.1.2.1.2) and then set this bit.

| OpMode | | |
|---|---|---|
| 00 | "Normal Operation" | Normal operating state |
| 01 | "Non-Driving" | Unused state |
| 10 | "Disable Bitstuffing and NRZI encoding" | Bitstuffing and NRZI encoding function disabled state in normal operating state |
| 11 | "Power-Down" | State where only single-end receiver is used |

### 7.4.18 018h *D_USB_Status (Device USB Status)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device /Host | 018h | ***D_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 indicates the device-related status.

**Bit7** ***VBUS***

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

**Bit6** **FSxHS**

This bit indicates the current operation mode. This bit is automatically set when HS Detection Handshaking is executed via the D_NegoControl.GoChirp bit (see "Functional Description"). Although operation mode can be forcibly changed by writing to this bit, it is recommended that this bit be manipulated only when operation mode needs to be changed without performing HS Detection Handshaking during a simulation, etc.

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

Although this bit can be read even in the SLEEP state, it can only be written when ACTIVE.

**Bits5-2** **Reserved**

**Bits1-0** ***LineState [1:0]***

These bits indicate the signal status on the USB cable. These bits are effective even during Sleep.

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

When TermSelect = 0, these bits indicate a 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 |

### 7.4.19 019h *H_USB_Status (Host USB Status)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 019h | ***H_USB_Status*** | R | 7: ***VBUS_State*** | 0: VBUSFLG = High | 1: VBUSFLG = Low | XXh |
| | | | | 6: | 0: | 1: | |
| | | | | 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 indicates the host-related status.

This register is effective even in the SLEEP state.

**Bit7** ***VBUS_State***

This bit indicates the status of the VBUSFLG pin.

**Bits6-2** **Reserved**

**Bits1-0** ***LineState [1:0]***

These bits indicate the signal status on the USB cable.

If the D_XcvrControl register's XcvrSelect[1:0] = 01 (FS transceiver selected), these bits indicate the received value of the FS receiver of the DP/DM. If XcvrSelect[1:0] = 11 (LS transceiver selected), these bits indicate the received value of the LS receiver.

When XcvrSelect[1:0] = 00 (HS transceiver selected), these bits indicate a USB bus activity.

| LineState | | |
|---|---|---|
| **XcvrSelect[1:0]** | **DP / DM** | **LineState [1:0]** |
| 00 | Don't Care | Bus activity<br>Bus activity detected: 0b01<br>No bus activity detected: 0b00 |
| 01 or 11 | SE0 | 0b00 |
| 01 or 11 | J | 0b01 |
| 01 or 11 | K | 0b10 |
| 01 or 11 | SE1 | 0b11 |

Note: XcvrSelect[1:0] = 10 is reserved, so that when this code is set, operation of the LSI cannot be guaranteed.

### 7.4.20 01Bh MTM_Config (Multi Transceiver Macro Config)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 01Bh | MTM_Config | | 7: | 0: | 1: | XXh |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: MTM_SlopeValue[1] | MTM Slope Value[1:0] | | |
| | | | R/W | 4: MTM_SlopeValue[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R/W | 1: MTM_TermValue[1] | MTM Termination Value[1:0] | | |
| | | | R/W | 0: MTM_TermValue[0] | | | |

This register is used to set adjustment values for the transceiver macro.

**Bits7-6 Reserved**

**Bits5-4 MTM_SlopeValue[1:0]**

These bits adjust the slew rate of the HS transmitter. The slew rate can be adjusted in four increments:

00: Mild

01: ↑

10: ↓

11: Sharp

**Bits3-2 Reserved**

**Bits1-0 MTM_TermValue[1:0]**

These bits adjust the termination of the HS transmission path. The termination can be adjusted in four increments:

00: High

01: ↑

10: ↓

11: Low

### 7.4.21 01Fh *HostDeviceSel (Host Device Select)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 01Fh | ***HostDeviceSel*** | | 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: ***HOSTxDEVICE*** | 0: Device mode | 1: Host mode | |

This register is used to select USB device or host mode.

This register is effective even in the SLEEP state.

**Bits7-1 Reserved**

**Bit0 HOSTxDEVICE**

This bit selects USB device or host mode.

0: USB device mode

1: USB host mode

The control of the internal system clock is based on this bit. When HOSTxDEVICE = 0 (i.e., device mode), the system clock is supplied to the shared block and the USB device block while the system clock for the USB host block is turned off. When HOSTxDEVICE = 1 (i.e., host mode), the system clock is supplied to the shared block and the USB host block while the system clock for the USB device block is turned off.

### 7.4.22 020h FIFO_Rd_0 (FIFO Read 0)

### 7.4.23 021h FIFO_Rd_1 (FIFO Read 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 020h | FIFO_Rd_0 | R | 7: FIFO_Rd_0 [7] | FIFO Read | XXh |
| | | | | 6: FIFO_Rd_0 [6] | | |
| | | | | 5: FIFO_Rd_0 [5] | | |
| | | | | 4: FIFO_Rd_0 [4] | | |
| | | | | 3: FIFO_Rd_0 [3] | | |
| | | | | 2: FIFO_Rd_0 [2] | | |
| | | | | 1: FIFO_Rd_0 [1] | | |
| | | | | 0: FIFO_Rd_0 [0] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 021h | FIFO_Rd_1 | R | 7: FIFO_Rd_1 [7] | FIFO Read | XXh |
| | | | | 6: FIFO_Rd_1 [6] | | |
| | | | | 5: FIFO_Rd_1 [5] | | |
| | | | | 4: FIFO_Rd_1 [4] | | |
| | | | | 3: FIFO_Rd_1 [3] | | |
| | | | | 2: FIFO_Rd_1 [2] | | |
| | | | | 1: FIFO_Rd_1 [1] | | |
| | | | | 0: FIFO_Rd_1 [0] | | |

**020h.Bit7-0, 021h.Bit7-0 FIFO_Rd_0 [7:0], FIFO_Rd_1[7:0]**

These bits allow data to be read from a FIFO that has had the AREAn{n=0-5}Join_0.JoinCPU_Rd bit set.

If these registers are accessed for read while a byte boundary exists in the FIFO, valid data will be output to only one side of the registers. For details, refer to Section 6.7.2.1.5, “Processing Odd Bytes in FIFO Access.”

To read FIFO data using these registers, always be sure to read the FIFO_RdRemain_H,L registers first to confirm the number of data bytes that can be read.

### 7.4.24 022h FIFO_Wr_0(FIFO Write 0)

### 7.4.25 023h FIFO_Wr_1(FIFO Write 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 022h | FIFO_Wr_0 | W | 7: FIFO_Wr_0 [7] | FIFO Write | XXh |
| | | | | 6: FIFO_Wr_0 [6] | | |
| | | | | 5: FIFO_Wr_0 [5] | | |
| | | | | 4: FIFO_Wr_0 [4] | | |
| | | | | 3: FIFO_Wr_0 [3] | | |
| | | | | 2: FIFO_Wr_0 [2] | | |
| | | | | 1: FIFO_Wr_0 [1] | | |
| | | | | 0: FIFO_Wr_0 [0] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 023h | FIFO_Wr_1 | W | 7: FIFO_Wr_1 [7] | FIFO Write | XXh |
| | | | | 6: FIFO_Wr_1 [6] | | |
| | | | | 5: FIFO_W_1r [5] | | |
| | | | | 4: FIFO_Wr_1 [4] | | |
| | | | | 3: FIFO_Wr_1 [3] | | |
| | | | | 2: FIFO_Wr_1 [2] | | |
| | | | | 1: FIFO_Wr_1 [1] | | |
| | | | | 0: FIFO_Wr_1 [0] | | |

**022h.Bit7-0, 023h.Bit7-0 FIFO_Wr_0 [7:0], FIFO_Wr_1[7:0]**

These bits allow data to be read from a FIFO that has had the AREAn{n=0-5}Join_0.JoinCPU_Wr bit set.

If these registers are accessed for write while a byte boundary exists in the FIFO, the data will be written to only one side of the registers. For details, refer to Section 6.7.2.1.5, “Processing Odd Bytes in FIFO Access.”

To write data into the FIFO using these registers, always be sure to read the FIFO_WrRemain_H,L registers first to confirm the number of data bytes that can be written to.

### 7.4.26 024h FIFO_RdRemain_H (FIFO Read Remain High)

### 7.4.27 025h FIFO_RdRemain_L (FIFO Read Remain Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 024h | FIFO_RdRemain_H | R | 7: RdRemainValid | 0:None | 1: Read Remain Valid | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R | 4: RdRemain [12] | Read Remain High | | |
| | | | | 3: RdRemain [11] | | | |
| | | | | 2: RdRemain [10] | | | |
| | | | | 1: nRdRemain [9] | | | |
| | | | | 0: RdRemain [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 025h | FIFO_RdRemain_L | R | 7: RdRemain [7] | Read Remain Low | 00h |
| | | | | 6: RdRemain [6] | | |
| | | | | 5: RdRemain [5] | | |
| | | | | 4: RdRemain [4] | | |
| | | | | 3: RdRemain [3] | | |
| | | | | 2: RdRemain [2] | | |
| | | | | 1: RdRemain [1] | | |
| | | | | 0: RdRemain [0] | | |

#### 024h.Bit7 RdRemainValid

This bit is set to 1 when FIFO is joined to the CPU I/F by the AREAn{n=0-5}Join_0.JoinCPU_Rd bit and the value of FIFO_RdRemain is valid.

#### 024h.Bit6-5 Reserved

#### 024h.Bit4-0, 025h.Bit7-0 RdRemain [12:0]

These bits indicate the number of readable data bytes in the FIFO joined to the CPU I/F by the AREAn{n=0-5}Join_0.JoinCPU_Rd bit. To get the number of readable data bytes in the FIFO, it is necessary to access the FIFO_RdRemain_H and FIFO_RdRemain_L registers in pairs. Be sure to access the FIFO_RdRemain_H register first.

### 7.4.28 026h FIFO_WrRemain_H (FIFO Write Remain High)

### 7.4.29 027h FIFO_WrRemain_L (FIFO Write Remain Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 026h | WrRemain_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R | 4: WrRemain [12] | FIFO Write Remain High | | |
| | | | | 3: WrRemain [11] | | | |
| | | | | 2: WrRemain [10] | | | |
| | | | | 1: WrRemain [9] | | | |
| | | | | 0: WrRemain [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 027h | WrRemain_L | R | 7: nWrRemain [7] | FIFO Write Remain Low | 00h |
| | | | | 6: WrRemain [6] | | |
| | | | | 5: WrRemain [5] | | |
| | | | | 4: WrRemain [4] | | |
| | | | | 3: WrRemain [3] | | |
| | | | | 2: WrRemain [2] | | |
| | | | | 1: WrRemain [1] | | |
| | | | | 0: WrRemain [0] | | |

**026h.Bit7-5 Reserved**

**026h.Bit4-0, 027h.Bit7-0 WrRemain [12:0]**

These bits indicate the amount of free space in the FIFO connected to the CPU interface by the AREAn{n=0-5}Join_0.JoinCPU_Wr bit. An 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 checking the amount of free space in the FIFO. To get the amount of free space in the FIFO, it is necessary to access the FIFO_WrRemain_H and FIFO_WrRemain_L registers in pairs. Be sure to access the FIFO_WrRemain_H register first.

### 7.4.30 028h FIFO_ByteRd(FIFO Byte Read)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 028h | FIFO_ByteRd | R | 7: FIFO_ByteRd [7] | FIFO Byte Read | XXh |
| | | | | 6: FIFO_ByteRd [6] | | |
| | | | | 5: FIFO_ByteRd [5] | | |
| | | | | 4: FIFO_ByteRd [4] | | |
| | | | | 3: FIFO_ByteRd [3] | | |
| | | | | 2: FIFO_ByteRd [2] | | |
| | | | | 1: FIFO_ByteRd [1] | | |
| | | | | 0: FIFO_ByteRd [0] | | |

**Bits7-0 FIFO_ByteRd [7:0]**

These bits allow data to be read in bytes from FIFO with the AREAn{n=0-5}Join_0.JoinCPU_Rd bit set. To read FIFO data using this register, always be sure to read the FIFO_RdRemain_H,L registers first to confirm the number of data bytes that can be read.

### 7.4.31 030h RAM_RdAdrs_H (RAM Read Address High)

### 7.4.32 031h RAM_RdAdrs_L (RAM Read Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 030h | RAM_RdAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: RAM_RdAdrs [12] | RAM Read Address | | |
| | | | | 3: RAM_RdAdrs [11] | | | |
| | | | | 2: RAM_RdAdrs [10] | | | |
| | | | | 1: RAM_RdAdrs [9] | | | |
| | | | | 0: RAM_RdAdrs [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 031h | RAM_RdAdrs_L | R / W | 7: RAM_RdAdrs [7] | RAM Read Address | 00h |
| | | | | 6: RAM_RdAdrs [6] | | |
| | | | | 5: RAM_RdAdrs [5] | | |
| | | | | 4: RAM_RdAdrs [4] | | |
| | | | | 3: RAM_RdAdrs [3] | | |
| | | | | 2: RAM_RdAdrs [2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

**030h.Bit7-5 Reserved**

**030h.Bit4-0, 031h.Bit7-2 RAM_RdAdrs[12:2]**

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 read until the CPU_IntStat.RAM_RdCmp bit is set. If these registers are accessed for read while the RAM_Rd function is active, the read value cannot be guaranteed. Note also that if data is written to these registers while the RAM_Rd function is active, such an operation will cause the LSI to operate erratically.

**031h.Bit1-0 Reserved**

### 7.4.33 032h RAM_RdControl (RAM Read Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 032h | RAM_RdControl | R / W | 7: RAM_GoRdCBW_CSW | 0: Do nothing | 1: RAM Read CBW_CSW 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_CSW**

This bit activates the RAM_Rd function to read the data that is received in the CBW area during USB device operation or the data that is received in the CSW area during USB host operation.

When this bit is set by writing 1 during USB device operation, the RAM_Rd function is activated, reading data from the CBW area. When the data stored in the RAM_Rd_00 through RAM_Rd_1E registers become valid, the CPU_IntStat.RAM_RdCmp bit is set to 1, at which time this bit is automatically cleared.

When this bit is set by writing 1 during USB host operation, the RAM_Rd function is activated, reading data from the CSW area. When the data stored in the RAM_Rd_00 through RAM_Rd_0C registers become valid, the CPU_IntStat.RAM_RdCmp bit is set to 1, at which time this bit is automatically cleared.

In either case, setting the RAM_RdAdrs_H,L registers and RAM_RdCount register is not necessary.

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 to activate the RAM_Rd function. Bytes of data equal to the specified count are read beginning with the specified start address, and when the data stored in the RAM_Rd_xx{xx=00–1F} registers become valid, the CPU_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_GoRdCBW_CSW bit, the function of the RAM_GoRdCBW_CSW bit is given priority.

**Bits5-0 Reserved**

### 7.4.34 035h RAM_RdCount (RAM Read Counter)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 035h | RAM_RdCount | R / W | 7: | RAM Read Counter | 00h |
| | | | | 6: | | |
| | | | | 5: RAM_RdCount [5] | | |
| | | | | 4: RAM_RdCount [4] | | |
| | | | | 3: RAM_RdCount [3] | | |
| | | | | 2: RAM_RdCount [2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

**Bits7-6 Reserved**

**Bits5-2 RAM_RdCount [5:2]**

These bits are used to set the number of data bytes to be read into the RAM_Rd_xx{xx=00 through 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 read until the CPU_IntStat.RAM_RdCmp bit is set. If this register is accessed for read while the RAM_Rd function is active, the read value cannot be guaranteed. Note also that if data is written to this register while the RAM_Rd function is active, such an operation will cause the LSI to operate erratically.

The maximum value that can be set in this register is 32 bytes. Be aware that if any number of data bytes exceeding this limit is set, the LSI will operate erratically.

**Bits1-0 Reserved**

### 7.4.35 038h RAM_WrAdrs_H (RAM Write Address High)

### 7.4.36 039h RAM_WrAdrs_L (RAM Write Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 038h | RAM_WrAdrs_H | | 7: | | 00h |
| | | | | 6: | | |
| | | | | 5: | | |
| | | | R / W | 4: RAM_WrAdrs [12] | RAM Write Address High | |
| | | | | 3: RAM_WrAdrs [11] | | |
| | | | | 2: RAM_WrAdrs [10] | | |
| | | | | 1: RAM_WrAdrs [9] | | |
| | | | | 0: RAM_WrAdrs [8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 039h | RAM_WrAdrs_L | R / W | 7: RAM_WrAdrs [7] | RAM Write Address Low | 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.

**038h.Bit7-5 Reserved**

**038h.Bit4-0, 039h.Bit7-0 RAM_WrAdrs[12:0]**

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. Since 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 checking RAM_WrAdrs. For details on how to write data, refer to the section on RAM_WrDoorH,L registers.

To inspect RAM_WrAdrs for confirmation, access the registers in order of RAM_WrAdrs_H and RAM_WrAdrs_L.

### 7.4.37 03Ah RAM_WrDoor_0 (RAM Write Door 0)

### 7.4.38 03Bh RAM_WrDoor_1 (RAM Write Door 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 03Ah | RAM_WrDoor_0 | W | 7: RAM_WrDoor_0 [7] | RAM Write Door 0 | XXh |
| | | | | 6: RAM_WrDoor_0 [6] | | |
| | | | | 5: RAM_WrDoor_0 [5] | | |
| | | | | 4: RAM_WrDoor_0 [4] | | |
| | | | | 3: RAM_WrDoor_0 [3] | | |
| | | | | 2: RAM_WrDoor_0 [2] | | |
| | | | | 1: RAM_WrDoor_0 [1] | | |
| | | | | 0: RAM_WrDoor_0 [0] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 03Bh | RAM_WrDoor_1 | W | 7: RAM_WrDoor_1 [7] | RAM Write Door 1 | XXh |
| | | | | 6: RAM_WrDoor_1 [6] | | |
| | | | | 5: RAM_WrDoor_1 [5] | | |
| | | | | 4: RAM_WrDoor_1 [4] | | |
| | | | | 3: RAM_WrDoor_1 [3] | | |
| | | | | 2: RAM_WrDoor_1 [2] | | |
| | | | | 1: RAM_WrDoor_1 [1] | | |
| | | | | 0: RAM_WrDoor_1 [0] | | |

**03Ah.Bit7-0, 03Bh.Bit7-0 RAM_WrDoor_0 [7:0], RAM_WrDoor_1[7:0]**

These registers are a write-only register, which is 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, allowing data to be written to successively.

During USB device mode, the RAM_WrDoor_0,1 registers may be used to write data for the descriptor or CSW_0 area. The data written to the descriptor area via these registers can be used as much as necessary by the ReplyDescriptor function. In other words, this data is neither erased nor overwritten by the descriptor reply function. However, if the area to which descriptor data is written overlaps any area reserved for another endpoint, the data in it may be overwritten.

During USB host mode, the RAM_WrDoor_0,1 registers may be used to write data for the CBW0,1 area.

**7.4.39 040h RAM_Rd_00 (RAM Read 00)**
**7.4.40 041h RAM_Rd_01 (RAM Read 01)**
**7.4.41 042h RAM_Rd_02 (RAM Read 02)**
**7.4.42 043h RAM_Rd_03 (RAM Read 03)**
**7.4.43 044h RAM_Rd_04 (RAM Read 04)**
**7.4.44 045h RAM_Rd_05 (RAM Read 05)**
**7.4.45 046h RAM_Rd_06 (RAM Read 06)**
**7.4.46 047h RAM_Rd_07 (RAM Read 07)**
**7.4.47 048h RAM_Rd_08 (RAM Read 08)**
**7.4.48 049h RAM_Rd_09 (RAM Read 09)**
**7.4.49 04Ah RAM_Rd_0A (RAM Read 0A)**
**7.4.50 04Bh RAM_Rd_0B (RAM Read 0B)**
**7.4.51 04Ch RAM_Rd_0C (RAM Read 0C)**
**7.4.52 04Dh RAM_Rd_0D (RAM Read 0D)**
**7.4.53 04Eh RAM_Rd_0E (RAM Read 0E)**
**7.4.54 04Fh RAM_Rd_0F (RAM Read 0F)**
**7.4.55 050h RAM_Rd_10 (RAM Read 10)**
**7.4.56 051h RAM_Rd_11 (RAM Read 11)**
**7.4.57 052h RAM_Rd_12 (RAM Read 12)**
**7.4.58 053h RAM_Rd_13 (RAM Read 13)**
**7.4.59 054h RAM_Rd_14 (RAM Read 14)**
**7.4.60 055h RAM_Rd_15 (RAM Read 15)**
**7.4.61 056h RAM_Rd_16 (RAM Read 16)**
**7.4.62 057h RAM_Rd_17 (RAM Read 17)**
**7.4.63 058h RAM_Rd_18 (RAM Read 18)**
**7.4.64 059h RAM_Rd_19 (RAM Read 19)**
**7.4.65 05Ah RAM_Rd_1A (RAM Read 1A)**
**7.4.66 05Bh RAM_Rd_1B (RAM Read 1B)**
**7.4.67 05Ch RAM_Rd_1C (RAM Read 1C)**
**7.4.68 05Dh RAM_Rd_1D (RAM Read 1D)**
**7.4.69 05Eh RAM_Rd_1E (RAM Read 1E)**

### 7.4.70 05Fh RAM_Rd_1F (RAM Read 1F)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 040h -05Fh | RAM_Rd_00 through 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] | | |

**040h-05Fh.Bit7-0 RAM_Rd_xx [7:0]**

These registers are used to store the data that is 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 of data bytes set in the RAM_RdCount register (e.g., if the count = 16, those stored in RAM_Rd_10 through RAM_Rd_1F) are ignored.

### 7.4.71 061h DMA_Config (DMA Config)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 061h | DMA_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: DMA Inactive | 1: DMA0 Active | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: ReqAssertCount [1] | Request Assert Count | | |
| | | | | 0: ReqAssertCount [0] | | | |

This register is used to set the operation mode of DMA.

**Bit7 FreeRun**

This bit sets DMA mode.

0: Count mode

1: Free-running mode

**Bit6 DMA_Mode**

This bit sets DMA mode.

0: The DMA operates in response to XDACK from the host as acknowledge.

1: The DMA operates in response to an access to the DMA_RdData/DMA_WrData register from the host as acknowledge.

**Bits5-4 Reserved**

**Bit3 ActiveDMA**

This bit enables DACK for DMA.

0: Disables DMA (DACK).

1: Enables DMA (DACK).

**Bit2 Reserved**

**Bits1-0 ReqAssertCount [1:0]**

These bits set the REQ Assert Count option provided to support burst read/writes by the CPU.

Set an assert count for XDREQ (number of transfer bytes). If the FIFO has a writable free space or readable valid data greater than the set assert count, XDREQ will be asserted. When DMA transfers for bytes equal to the set assert count is complete, XDREQ is temporarily negated, and when the FIFO is confirmed to have a writable free space or readable valid data greater than the set assert count again, XDREQ is reasserted. This means that when XDREQ is asserted once, transfers for bytes equal to the set assert count are guaranteed.

However, if DMA is set to count mode and the count of remaining bytes in DMA_Count_HH,HL,LH,LL is smaller than the set assert count, the remaining count in DMA_Count_HH,HL,LH,LL has priority, so that XDREQ is asserted when the FIFO has a writable free space or readable valid data greater than the remaining count in DMA_Count_HH,HL,LH,LL.

The table below shows the relationship between DMA_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) vs. the XDREQ signal and the number of transferable bytes.

The remaining count in DMA_Count_HH,HL,LH,LL must be greater than or equal to 1 for the REQ Assert Count option to be able to work.

| | **Count>=Req** | | **Count<Req** | |
|---|---|---|---|---|
| | **Ready>=Req** | **Ready<Req** | **Ready>=Count** | **Ready<Count** |
| XDREQ | 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.

### 7.4.72 062h DMA_Control (DMA Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 062h | DMA_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 | |
| | | | R / W | 3: Dir | 0: CPU-IF -> FIFO RAM | 1: CPU-IF <- FIFO RAM | |
| | | | | 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 DMA and shows the status of DMA.

**Bit7 DMA_Running**

This bit is set to 1 and remains set while a transfer on DMA is underway. While this bit remains set, the AREAn{n=0-5}Join_0.JoinDMA bit cannot be overwritten.

**Bits6-5 Reserved**

**Bit4 CounterClr**

Setting this bit to 1 clears the DMA_Count_HH,HL,LH,LL registers to 0x00. A write to this bit is ignored while the DMA_Running bit = 1.

**Bit3 Dir**

This bit sets the transfer direction of DMA.

0: CPU IF to FIFO RAM (DMA write)

1: FIFO RAM to CPU IF (DMA read)

**Bit2 Reserved**

**Bit1 DMA_Stop**

Setting this bit to 1 stops the transfer on DMA. When the transfer on DMA stops, the DMA_Running bit is cleared to 0. Furthermore, the DMA_Cmp bit in the CPU_IntStat register is set to 1. To restart a transfer on DMA, check the DMA_Running or the DMA_Cmp bit to confirm the status and wait until the DMA terminates.

**Bit0 DMA_Go**

Setting this bit to 1 starts a transfer on DMA.

Setting this bit to 1 star

### 7.4.73 064h DMA_Remain_H (DMA FIFO Remain High)

### 7.4.74 065h DMA_Remain_L (DMA0 FIFO Remain Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 064h | DMA_Remain_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R | 4: DMA_Remain [12] | DMA FIFO Remain High | | |
| | | | | 3: DMA_Remain [11] | | | |
| | | | | 2: DMA_Remain [10] | | | |
| | | | | 1: DMA_Remain [9] | | | |
| | | | | 0: DMA_Remain [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 065h | DMA_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] | | |

#### 064h.Bit7-5 Reserved

#### 064h.Bit4-0, 065h.Bit7-0 DMA_Remain [12:0]

For read, these bits indicate the number of data bytes remaining in the FIFO connected to the DMA by the AREAn{n=0-5}Join_0.JoinDMA bit.

For write, these bits indicate the amount of free space remaining in the FIFO connected to the DMA by the AREAn{n=0-5}Join_0.JoinDMA bit. Since an exact amount of free space in the FIFO cannot be known through 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 DMA_Remain_H and L.

### 7.4.75 068h DMA_Count_HH (DMA Transfer Byte Counter High/High)

### 7.4.76 069h DMA_Count_HL (DMA Transfer Byte Counter High/Low)

### 7.4.77 06Ah DMA_Count_LH (DMA Transfer Byte Counter Low/High)

### 7.4.78 06Bh DMA_Count_LL (DMA Transfer Byte Counter Low/Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 068h | DMA_Count_HH | R / W | 7: DMA_Count [31] | DMA Transfer Byte Counter High-HIgh | 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] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 069h | DMA_Count_HL | R / W | 7: DMA_Count [23] | DMA Transfer Byte Counter High-Low | 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] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 06Ah | DMA_Count_LH | R / W | 7: DMA_Count [15] | DMA Transfer Byte Counter Low-High | 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] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 06Bh | DMA_Count_LL | R / W | 7: DMA_Count [7] | DMA Transfer Byte Counter Low-Low | 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 in bytes for a transfer on DMA during count mode. 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 DMA_Control.DMA_Go bit to 1 to start a DMA transfer. When transfers for the transfer bytes set in these registers is complete, the DMA transfer is terminated.

During free-running mode, the counter counts up from the set value. When the value in the DMA_Count_HH,HL,LH,LL registers overflows, the DMA_CountUp bit in the CPU_IntStat register is set to 1. The counter continues counting even after an overflow. In this mode, the number of bytes that have been DMA transferred can be inspected.

Since an exact byte count cannot be known through 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 DMA_Count_HH, HL, LH, and LL.

### 7.4.79 06Ch DMA_RdData_0 (DMA Read Data 0)

### 7.4.80 06Dh DMA_RdData_1 (DMA Read Data 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 06Ch | DMA_RdData_0 | R | 7: DMA_RdData_0 [7] | DMA Read Data 0 | XXh |
| | | | | 6: DMA_RdData_0 [6] | | |
| | | | | 5: DMA_RdData_0 [5] | | |
| | | | | 4: DMA_RdData_0 [4] | | |
| | | | | 3: DMA_RdData_0 [3] | | |
| | | | | 2: DMA_RdData_0 [2] | | |
| | | | | 1: DMA_RdData_0 [1] | | |
| | | | | 0: DMA_RdData_0 [0] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 06Dh | DMA_RdData_1 | R | 7: DMA_RdData_1 [7] | DMA Read Data 1 | XXh |
| | | | | 6: DMA_RdData_1 [6] | | |
| | | | | 5: DMA_RdData_1 [5] | | |
| | | | | 4: DMA_RdData_1 [4] | | |
| | | | | 3: DMA_RdData_1 [3] | | |
| | | | | 2: DMA_RdData_1 [2] | | |
| | | | | 1: DMA_RdData_1 [1] | | |
| | | | | 0: DMA_RdData_1 [0] | | |

**06Ch.Bit7-0, 06Dh.Bit7-0 DMA_RdData_0[7:0], DMA_RdData_1[7:0]**

By accessing these registers when the DMA_Config.DMA_Mode bit = 1, it is possible to read data from the FIFO connected to the DMA by the AREAn{n=0-5}Join_0.JoinDMA bit. Before this operation can be performed, the DMA_Control.Dir bit must be set for DMA read.

When operating in 8-bit mode, this DMA access can be achieved by accessing either DMA_RdData_0 or DMA0_RdData_1.

### 7.4.81 06Eh DMA_WrData_0 (DMA Write Data 0)

### 7.4.82 06Fh DMA_WrData_1 (DMA Write Data 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 06Eh | DMA_WrData_0 | W | 7: DMA_WrData_0 [7] | DMA Write Data 0 | XXh |
| | | | | 6: DMA_WrData_0 [6] | | |
| | | | | 5: DMA_WrData_0 [5] | | |
| | | | | 4: DMA_WrData_0 [4] | | |
| | | | | 3: DMA_WrData_0 [3] | | |
| | | | | 2: DMA_WrData_0 [2] | | |
| | | | | 1: DMA_WrData_0 [1] | | |
| | | | | 0: DMA_WrData_0 [0] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 06Fh | DMA_WrData_1 | W | 7: DMA_WrData_1 [7] | DMA Write Data 1 | XXh |
| | | | | 6: DMA_WrData_1 [6] | | |
| | | | | 5: DMA_WrData_1 [5] | | |
| | | | | 4: DMA_WrData_1 [4] | | |
| | | | | 3: DMA_WrData_1 [3] | | |
| | | | | 2: DMA_WrData_1 [2] | | |
| | | | | 1: DMA_WrData_1 [1] | | |
| | | | | 0: DMA_WrData_1 [0] | | |

**06Eh.Bit7-0, 06Fh.Bit7-0 DMA_WrData_0[7:0], DMA_WrData_1[7:0]**

By accessing these registers when the DMA_Config.DMA_Mode bit = 1, it is possible to write data into the FIFO connected to the DMA by the AREAn{n=0-5}Join_0.JoinDMA, bit. Before this operation can be performed, the DMA_Control.Dir bit must be set for DMA write.

When operating in 8-bit mode, this DMA access can be achieved by accessing either DMA_WrData_0 or DMA_WrData_1.

### 7.4.83 071h *ModeProtect(Mode Protection)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 071h | ***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]*** | | |

**Bits7-0 ModeProtect [7:0]**

This register protects the values of the CPU-Config register and ClkSelect register. Writing "56h" to this register removes protection, allowing the CPU-Config register and ClkSelect register to be accessed for write.

During normal use, after setting the CPU-Config register and ClkSelect register to any value, write other than "56h" (e.g., 00h) to this register to protect the CPU-Config register and ClkSelect register.

This bit can be accessed even during Sleep.

### 7.4.84 073h *ClkSelect (Clock Select)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 073h | ***ClkSelect*** | R / W | 7: ***ClkSource*** | 0: Xtal | 1: CLKIN | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: ***ClkFreq[1]*** | Input clock frequency<br>00:12MHz, 01: 24MHz, 11: 48MHz | | |
| | | | | 0: ***ClkFreq[0]*** | | | |

This register initializes the clock for the LSI.

This register must be set before operating the LSI.

Note that this register is effective even in the SLEEP state.

**Bit7 ClkSource**

This bit selects the clock source for the LSI.

The relationship between ClkSource and ClkFreq is shown below.

In CPU_Cut mode, input at the CLKIN pin is turned off from the initial stage on. This is also true for the other CPU interface pins.

**Bits6-2 Reserved**

**Bits1-0 ClkFreq[1:0]**

These bits select the clock frequency for the LSI.

The relationship between ClkFreq and ClkSource is shown below.

| | | ClkSource | |
|---|---|---|---|
| ClkFreq | | 0: Crystal resonator | 1: CLKIN pin |
| 00 | 12MHz | ○ | ○ |
| 01 | 24MHz | ○ | ○ |
| 11 | 48MHz | × | ○ |

○: Usable.

×: Cannot be used.

### 7.4.85 075h *CPU_Config (CPU Configuration)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 075h | ***CPU_Config*** | 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_Endian*** | 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 operating modes of the LSI.

The bits in this register can be accessed even in the SLEEP state.

**Bit7 IntLevel**

This bit sets the XINT logic level.

0: Active low

1: Active high

**Bit6 IntMode**

This bit sets the XINT output mode.

0: 1/0 mode

1: Hi-Z/0 mode

**Bit5 DREQ_Level**

This bit sets the XDREQ logic level.

0: Active low

1: Active high

**Bit4 DACK_Level**

This bit sets the XDACK logic level.

0: Active low

1: Active high

**Bit3 CS_Mode**

This bit sets the DMA operating mode.

0: Operated as a valid DMA access when XDACK is asserted.

1: Operated as a valid DMA access when XCS and XDACK are both asserted.

**Bit2 CPU_Endian**

This bit sets the CPU bus when operating in 16-bit mode. Do not set this bit when operating in 8-bit mode.

0: The even and the odd addresses of the bus are used for the upper and the lower bytes, respectively.

1: The even and the odd addresses of the bus are used for the lower and the upper bytes, respectively.

This bit setting is enabled by accessing the address 077h for a dummy read after writing to the register. If the circuit is reset by the ChipReset.ResetAll bit, the register value is initialized, in which case the initialized register value does not take effect until after the address 077h is accessed for a dummy read, as in the preceding case.

**Bits1-0 BusMode and Bus8x16**

These bits set the CPU operating mode.

| Operating mode | bit1.BusMode | bit0.Bus8x16 |
|---|---|---|
| 16bit Strobe mode | 0 | 0 |
| 16bit BE mode | 1 | * |
| 8bit mode | 0 | 1 |

### 7.4.86 077h *CPU_ChgEndian(CPU Change Endian)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 077h | ***CPU_ChgEndian*** | R | 7: ***CPU_ChgEndian[7]*** | CPU Change Endian | XXh |
| | | | | 6: ***CPU_ChgEndian[6]*** | | |
| | | | | 5: ***CPU_ChgEndian[5]*** | | |
| | | | | 4: ***CPU_ChgEndian[4]*** | | |
| | | | | 3: ***CPU_ChgEndian[3]*** | | |
| | | | | 2: ***CPU_ChgEndian[2]*** | | |
| | | | | 1: ***CPU_ChgEndian[1]*** | | |
| | | | | 0: ***CPU_ChgEndian[0]*** | | |

**Bits7-0 CPU_ChgEndian [7:0]**

Performing a dummy read of these register bits causes the endian set by CPU_Config.CPU_Endian to take effect.

These bits can be accessed even in the SLEEP state.

### 7.4.87 080h AREA0StartAdrs_H (AREA0 Start Address High)

### 7.4.88 081h AREA0StartAdrs_L (AREA0 Start Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 080h | AREA0StartAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: StartAdrs[12] | AREA0 Start Address High | | |
| | | | | 3: StartAdrs[11] | | | |
| | | | | 2: StartAdrs[10] | | | |
| | | | | 1: StartAdrs[9] | | | |
| | | | | 0: StartAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 081h | AREA0StartAdrs_L | R/W | 7: StartAdrs[7] | AREA0 Start Address Low | 00h |
| | | | | 6: StartAdrs[6] | | |
| | | | | 5: StartAdrs[5] | | |
| | | | | 4: StartAdrs[4] | | |
| | | | | 3: StartAdrs[3] | | |
| | | | | 2: StartAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA0.

**080h.Bits7-5 Reserved**

**080h.Bits4-0, 081h.Bits7-2 StartAdrs[12:2]**

These bits set the start address of the FIFO to be allocated to the AREA0 FIFO area.

Since address values are set using the high-order bits 12 to 2, the start address here should be set in four-byte units.

The memory space allocated to the AREA0 FIFO area ranges from this address to a location preceding by one byte the address set by AREA0EndAdr.

After setting AREA0StartAdrs and AREA0EndAdrs, always set the ClrAREA0 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA0 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Additionally, if the AREA0 FIFO area and another FIFO area overlap, the LSI may behave erratically.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**081h.Bits1-0 Reserved**

### 7.4.89 082h AREA0EndAdrs_H (AREA0 End Address High)

### 7.4.90 083h AREA0EndAdrs_L (AREA0 End Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 082h | AREA0EndAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: EndAdrs[12] | AREA0 End Address High | | |
| | | | | 3: EndAdrs[11] | | | |
| | | | | 2: EndAdrs[10] | | | |
| | | | | 1: EndAdrs[9] | | | |
| | | | | 0: EndAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 083h | AREA0EndAdrs_L | R/W | 7: EndAdrs[7] | AREA0 End Address Low | 00h |
| | | | | 6: EndAdrs[6] | | |
| | | | | 5: EndAdrs[5] | | |
| | | | | 4: EndAdrs[4] | | |
| | | | | 3: EndAdrs[3] | | |
| | | | | 2: EndAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA0.

**082h.Bits7-5 Reserved**

**082h.Bits4-0, 083h.Bits7-2 EndAdrs[12:2]**

These bits set the end address of the FIFO to be allocated to the AREA0 FIFO area. The actual end address to be set here is one byte less than the intended end address.

Since address values are set using the high-order bits 12 to 2, the end address here should be set in four-byte units.

The memory space allocated to the AREA0 FIFO area ranges from the start address to a location preceding by one byte the address set by AREA0EndAdr.

After setting AREA0StartAdrs and AREA0EndAdrs, always set the ClrAREA0 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA0 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Furthermore, if the AREA0 FIFO area and another FIFO area overlap, the LSI may behave erratically either.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**083h.Bits1-0 Reserved**

### 7.4.91 084h AREA1StartAdrs_H (AREA1 Start Address High)

### 7.4.92 085h AREA1StartAdrs_L (AREA1 Start Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 084h | AREA1StartAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: StartAdrs[12] | AREA1 Start Address High | | |
| | | | | 3: StartAdrs[11] | | | |
| | | | | 2: StartAdrs[10] | | | |
| | | | | 1: StartAdrs[9] | | | |
| | | | | 0: StartAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 085h | AREA1StartAdrs_L | R/W | 7: StartAdrs[7] | AREA1 Start Address Low | 00h |
| | | | | 6: StartAdrs[6] | | |
| | | | | 5: StartAdrs[5] | | |
| | | | | 4: StartAdrs[4] | | |
| | | | | 3: StartAdrs[3] | | |
| | | | | 2: StartAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA1.

**084h.Bits7-5 Reserved**

**084h.Bits4-0, 085h.Bits7-2 StartAdrs[12:2]**

These bits set the start address of the FIFO to be allocated to the AREA1 FIFO area.

Since address values are set using the high-order bits 12 to 2, the start address here should be set in four-byte units.

The memory space allocated to the AREA1 FIFO area ranges from this address to a location preceding by one byte the address set by AREA1EndAdr.

After setting AREA1StartAdrs and AREA1EndAdrs, always set the ClrAREA1 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA1 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Additionally, if the AREA1 FIFO area and another FIFO area overlap, the LSI may behave erratically.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**085h.Bits1-0 Reserved**

### 7.4.93 086h AREA1EndAdrs_H (AREA1 End Address High)

### 7.4.94 087h AREA1EndAdrs_L (AREA1 End Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 086h | AREA1EndAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: EndAdrs[12] | AREA1 End Address High | | |
| | | | | 3: EndAdrs[11] | | | |
| | | | | 2: EndAdrs[10] | | | |
| | | | | 1: EndAdrs[9] | | | |
| | | | | 0: Enddrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 087h | AREA1EndAdrs_L | R/W | 7: EndAdrs[7] | ARE1 End Address Low | 00h |
| | | | | 6: EndAdrs[6] | | |
| | | | | 5: EndAdrs[5] | | |
| | | | | 4: EndAdrs[4] | | |
| | | | | 3: EndAdrs[3] | | |
| | | | | 2: EndAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA1.

**086h.Bits7-5 Reserved**

**086h.Bits4-0, 087h.Bits7-2 EndAdrs[12:2]**

These bits set the end address of the FIFO to be allocated to the AREA1 FIFO area. The actual end address to be set here is one byte less than the intended end address.

Since address values are set using the high-order bits 12 to 2, the end address here should be set in four-byte units.

The memory space allocated to the AREA1 FIFO area ranges from the start address to a location preceding by one byte the address set by AREA1EndAdr.

After setting AREA1StartAdrs and AREA1EndAdrs, always set the ClrAREA1 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA1 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Furthermore, if the AREA1 FIFO area and another FIFO area overlap, the LSI may behave erratically either.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**087h.Bits1-0 Reserved**

### 7.4.95 088h AREA2StartAdrs_H (AREA2 Start Address High)

### 7.4.96 089h AREA2StartAdrs_L (AREA2 Start Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 088h | AREA2StartAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: StartAdrs[12] | AREA2 Start Address High | | |
| | | | | 3: StartAdrs[11] | | | |
| | | | | 2: StartAdrs[10] | | | |
| | | | | 1: StartAdrs[9] | | | |
| | | | | 0: StartAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 089h | AREA2StartAdrs_L | R/W | 7: StartAdrs[7] | AREA2 Start Address Low | 00h |
| | | | | 6: StartAdrs[6] | | |
| | | | | 5: StartAdrs[5] | | |
| | | | | 4: StartAdrs[4] | | |
| | | | | 3: StartAdrs[3] | | |
| | | | | 2: StartAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA2.

**088h.Bits7-5 Reserved**

**088h.Bits4-0, 089h.Bits7-2 StartAdrs[12:2]**

These bits set the start address of the FIFO to be allocated to the AREA2 FIFO area.

Since address values are set using the high-order bits 12 to 2, the start address here should be set in four-byte units.

The memory space allocated to the AREA2 FIFO area ranges from this address to a location preceding by one byte the address set by AREA2EndAdr.

After setting AREA2StartAdrs and AREA2EndAdrs, always set the ClrAREA2 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA2 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Additionally, if the AREA2 FIFO area and another FIFO area overlap, the LSI may behave erratically.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**089h.Bits1-0 Reserved**

### 7.4.97 08Ah AREA2EndAdrs_H (AREA2 End Address High)

### 7.4.98 08Bh AREA2EndAdrs_L (AREA2 End Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 08Ah | AREA2EndAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: EndAdrs[12] | AREA2 End Address High | | |
| | | | | 3: EndAdrs[11] | | | |
| | | | | 2: EndAdrs[10] | | | |
| | | | | 1: EndAdrs[9] | | | |
| | | | | 0: Enddrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 08Bh | AREA2EndAdrs_L | R/W | 7: EndAdrs[7] | ARE2 End Address Low | 00h |
| | | | | 6: EndAdrs[6] | | |
| | | | | 5: EndAdrs[5] | | |
| | | | | 4: EndAdrs[4] | | |
| | | | | 3: EndAdrs[3] | | |
| | | | | 2: EndAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA2.

**08Ah.Bits7-5 Reserved**

**08Ah.Bits4-0, 08Bh.Bits7-2 EndAdrs[12:2]**

These bits set the end address of the FIFO to be allocated to the AREA2 FIFO area. The actual end address to be set here is one byte less than the intended end address.

Since address values are set using the high-order bits 12 to 2, the end address here should be set in four-byte units.

The memory space allocated to the AREA2 FIFO area ranges from the start address to a location preceding by one byte the address set by AREA2EndAdr.

After setting AREA2StartAdrs and AREA2EndAdrs, always set the ClrAREA2 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA2 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Furthermore, if the AREA2 FIFO area and another FIFO area overlap, the LSI may behave erratically either.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**08Bh.Bits1-0 Reserved**

### 7.4.99 08Ch AREA3StartAdrs_H (AREA3 Start Address High)

### 7.4.100 08Dh AREA3StartAdrs_L (AREA3 Start Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 08Ch | AREA3StartAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: StartAdrs[12] | AREA3 Start Address High | | |
| | | | | 3: StartAdrs[11] | | | |
| | | | | 2: StartAdrs[10] | | | |
| | | | | 1: StartAdrs[9] | | | |
| | | | | 0: StartAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 08Dh | AREA3StartAdrs_L | R/W | 7: StartAdrs[7] | AREA3 Start Address Low | 00h |
| | | | | 6: StartAdrs[6] | | |
| | | | | 5: StartAdrs[5] | | |
| | | | | 4: StartAdrs[4] | | |
| | | | | 3: StartAdrs[3] | | |
| | | | | 2: StartAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA3.

**08Ch.Bits7-5 Reserved**

**08Ch.Bits4-0, 08Dh.Bits7-2 StartAdrs[12:2]**

These bits set the start address of the FIFO to be allocated to the AREA3 FIFO area.

Since address values are set using the high-order bits 12 to 2, the start address here should be set in four-byte units.

The memory space allocated to the AREA3 FIFO area ranges from this address to a location preceding by one byte the address set by AREA3EndAdr.

After setting AREA3StartAdrs and AREA3EndAdrs, always set the ClrAREA3 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA3 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Additionally, if the AREA3 FIFO area and another FIFO area overlap, the LSI may behave erratically.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**08Dh.Bits1-0 Reserved**

### 7.4.101 08Eh AREA3EndAdrs_H (AREA3 End Address High)

### 7.4.102 08Fh AREA3EndAdrs_L (AREA3 End Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 08Eh | AREA3EndAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: EndAdrs[12] | AREA3 End Address High | | |
| | | | | 3: EndAdrs[11] | | | |
| | | | | 2: EndAdrs[10] | | | |
| | | | | 1: EndAdrs[9] | | | |
| | | | | 0: Enddrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 08Fh | AREA3EndAdrs_L | R/W | 7: EndAdrs[7] | ARE3 End Address Low | 00h |
| | | | | 6: EndAdrs[6] | | |
| | | | | 5: EndAdrs[5] | | |
| | | | | 4: EndAdrs[4] | | |
| | | | | 3: EndAdrs[3] | | |
| | | | | 2: EndAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA3.

**08Eh.Bits7-5 Reserved**

**08Eh.Bits4-0, 08Fh.Bits7-2 EndAdrs[12:2]**

These bits set the end address of the FIFO to be allocated to the AREA3 FIFO area. The actual end address to be set here is one byte less than the intended end address.

Since address values are set using the high-order bits 12 to 2, the end address here should be set in four-byte units.

The memory space allocated to the AREA3 FIFO area ranges from the start address to a location preceding by one byte the address set by AREA3EndAdr.

After setting AREA3StartAdrs and AREA3EndAdrs, always set the ClrAREA3 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA3 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Furthermore, if the AREA3 FIFO area and another FIFO area overlap, the LSI may behave erratically either.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**08Fh.Bits1-0 Reserved**

### 7.4.103 090h AREA4StartAdrs_H (AREA4 Start Address High)

### 7.4.104 091h AREA4StartAdrs_L (AREA4 Start Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 090h | AREA4StartAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: StartAdrs[12] | AREA4 Start Address High | | |
| | | | | 3: StartAdrs[11] | | | |
| | | | | 2: StartAdrs[10] | | | |
| | | | | 1: StartAdrs[9] | | | |
| | | | | 0: StartAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 091h | AREA4StartAdrs_L | R/W | 7: StartAdrs[7] | AREA4 Start Address Low | 00h |
| | | | | 6: StartAdrs[6] | | |
| | | | | 5: StartAdrs[5] | | |
| | | | | 4: StartAdrs[4] | | |
| | | | | 3: StartAdrs[3] | | |
| | | | | 2: StartAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA4.

**090h.Bits7-5 Reserved**

**090h.Bits4-0, 091h.Bits7-2 StartAdrs[12:2]**

These bits set the start address of the FIFO to be allocated to the AREA4 FIFO area.

Since address values are set using the high-order bits 12 to 2, the start address here should be set in four-byte units.

The memory space allocated to the AREA4 FIFO area ranges from this address to a location preceding by one byte the address set by AREA4EndAdr.

After setting AREA4StartAdrs and AREA4EndAdrs, always set the ClrAREA4 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA4 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Additionally, if the AREA4 FIFO area and another FIFO area overlap, the LSI may behave erratically.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**091h.Bits1-0 Reserved**

### 7.4.105 092h AREA4EndAdrs_H (AREA4 End Address High)

### 7.4.106 093h AREA4EndAdrs_L (AREA4 End Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 092h | AREA4EndAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: EndAdrs[12] | AREA4 End Address High | | |
| | | | | 3: EndAdrs[11] | | | |
| | | | | 2: EndAdrs[10] | | | |
| | | | | 1: EndAdrs[9] | | | |
| | | | | 0: Enddrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 093h | AREA4EndAdrs_L | R/W | 7: EndAdrs[7] | ARE4 End Address Low | 00h |
| | | | | 6: EndAdrs[6] | | |
| | | | | 5: EndAdrs[5] | | |
| | | | | 4: EndAdrs[4] | | |
| | | | | 3: EndAdrs[3] | | |
| | | | | 2: EndAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA4.

**092h.Bits7-5 Reserved**

**092h.Bits4-0, 093h.Bits7-2 EndAdrs[12:2]**

These bits set the end address of the FIFO to be allocated to the AREA4 FIFO area. The actual end address to be set here is one byte less than the intended end address.

Since address values are set using the high-order bits 12 to 2, the end address here should be set in four-byte units.

The memory space allocated to the AREA4 FIFO area ranges from the start address to a location preceding by one byte the address set by AREA4EndAdr.

After setting AREA4StartAdrs and AREA4EndAdrs, always set the ClrAREA4 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA4 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Furthermore, if the AREA4 FIFO area and another FIFO area overlap, the LSI may behave erratically either.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**093h.Bits1-0 Reserved**

### 7.4.107 094h AREA5StartAdrs_H (AREA5 Start Address High)

### 7.4.108 095h AREA5StartAdrs_L (AREA5 Start Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 094h | AREA5StartAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: StartAdrs[12] | AREA5 Start Address High | | |
| | | | | 3: StartAdrs[11] | | | |
| | | | | 2: StartAdrs[10] | | | |
| | | | | 1: StartAdrs[9] | | | |
| | | | | 0: StartAdrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 095h | AREA5StartAdrs_L | R/W | 7: StartAdrs[7] | AREA5 Start Address Low | 00h |
| | | | | 6: StartAdrs[6] | | |
| | | | | 5: StartAdrs[5] | | |
| | | | | 4: StartAdrs[4] | | |
| | | | | 3: StartAdrs[3] | | |
| | | | | 2: StartAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA5.

**094h.Bits7-5 Reserved**

**094h.Bits4-0, 095h.Bits7-2 StartAdrs[12:2]**

These bits set the start address of the FIFO to be allocated to the AREA5 FIFO area.

Since address values are set using the high-order bits 12 to 2, the start address here should be set in four-byte units.

The memory space allocated to the AREA5 FIFO area ranges from this address to a location preceding by one byte the address set by AREA5EndAdr.

After setting AREA5StartAdrs and AREA5EndAdrs, always set the ClrAREA5 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA5 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Additionally, if the AREA5 FIFO area and another FIFO area overlap, the LSI may behave erratically.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**095h.Bits1-0 Reserved**

### 7.4.109 096h AREA5EndAdrs_H (AREA5 End Address High)

### 7.4.110 097h AREA5EndAdrs_L (AREA5 End Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 096h | AREA5EndAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R/W | 4: EndAdrs[12] | AREA5 End Address High | | |
| | | | | 3: EndAdrs[11] | | | |
| | | | | 2: EndAdrs[10] | | | |
| | | | | 1: EndAdrs[9] | | | |
| | | | | 0: Enddrs[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device / Host | 097h | AREA5EndAdrs_L | R/W | 7: EndAdrs[7] | ARE5 End Address Low | 00h |
| | | | | 6: EndAdrs[6] | | |
| | | | | 5: EndAdrs[5] | | |
| | | | | 4: EndAdrs[4] | | |
| | | | | 3: EndAdrs[3] | | |
| | | | | 2: EndAdrs[2] | | |
| | | | | 1: | | |
| | | | | 0: | | |

These registers set the FIFO area used in AREA5.

**096h.Bits7-5 Reserved**

**096h.Bits4-0, 097h.Bits7-2 EndAdrs[12:2]**

These bits set the end address of the FIFO to be allocated to the AREA5 FIFO area. The actual end address to be set here is one byte less than the intended end address.

Since address values are set using the high-order bits 12 to 2, the end address here should be set in four-byte units.

The memory space allocated to the AREA5 FIFO area ranges from the start address to a location preceding by one byte the address set by AREA5EndAdr.

After setting AREA5StartAdrs and AREA5EndAdrs, always set the ClrAREA5 bit of the AREAnFIFO_Clr register to 1 to clear the FIFO in the AREA5 FIFO area.

Note that if MaxSize of the joined USB device or host exceeds the area set here, the LSI may behave erratically. Furthermore, if the AREA5 FIFO area and another FIFO area overlap, the LSI may behave erratically either.

Since the LSI's internal RAM is 4.5 KB, addresses up to 0x1200 are supported.

**097h.Bits1-0 Reserved**

### 7.4.111 09Fh AREAnFIFO_Clr (AREAn FIFO Clear)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 09Fh | AREAnFIFO_Clr | | 7: | 0: | 1: | XXh |
| | | | | 6: | 0: | 1: | |
| | | | W | 5: AREA5FIFO_Clr | 0: Do nothing | 1: Clear AREA5 FIFO | |
| | | | W | 4: AREA4FIFO_Clr | 0: Do nothing | 1: Clear AREA4 FIFO | |
| | | | W | 3: AREA3FIFO_Clr | 0: Do nothing | 1: Clear AREA3 FIFO | |
| | | | W | 2: AREA2FIFO_Clr | 0: Do nothing | 1: Clear AREA2 FIFO | |
| | | | W | 1: AREA1FIFO_Clr | 0: Do nothing | 1: Clear AREA1 FIFO | |
| | | | W | 0: AREA0FIFO_Clr | 0: Do nothing | 1: Clear AREA0 FIFO | |

This register clears the FIFO in the relevant FIFO area of AREAn{n=0-5}. This is a write-only register.

When set to 1, each bit in this register clears the FIFO without retaining the value set in it.

While DMA is joined to the FIFO area of AREAn{n=0-5} and the relevant DMA is active (while DMA_Running bit = 1), do not set the bit for the corresponding endpoint to 1.

Note that this register only initializes data retention information and does not write or clear the data itself. Therefore, in no case will the data in the RAM be cleared by the assertion of any bit in this register.

### 7.4.112 0A0h AREA0Join_0 (AREA0 Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A0h | AREA0Join_0 | R/W | 7: JoinFIFO_Stat | 0: Do nothing | 1: Join to FIFO Status | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: JoinDMA | 0: Do nothing | 1: Join to DMA | |
| | | | R/W | 1: JoinCPU_Rd | 0: Do nothing | 1: Join to CPU Read | |
| | | | R/W | 0: JoinCPU_Wr | 0: Do nothing | 1: Join to CPU Write | |

This register sets the port to be connected to the AREA0 FIFO area.

**Bit7 JoinFIFO_Stat**

This bit enables the full and empty status of the FIFO in the AREA0 FIFO area to be monitored by means of FIFO_IntStat.FIFO_NotEmpty, FIFO_IntStat.FIFO_Full, and FIFO_IntStat.FIFO_Empty.

**Bits6-3 Reserved**

**Bit2 JoinDMA**

This bit causes a DMA transfer to be performed with the FIFO in the AREA0 FIFO area. The transfer direction is determined by the DMA_Control.Dir bit.

**Bit1 JoinCPU_Rd**

This bit causes a read transfer for CPU register access to be performed with the FIFO in the AREA0 FIFO area. In other words, when a read of the FIFO_Rd_0,1 or FIFO_ByteRd register is performed, data is read from this FIFO area.

**Bit0 JoinCPU_Wr**

This bit causes a write transfer for CPU register access to be performed with the FIFO in the AREA0 FIFO area. That is, when a write is performed to the FIFO_Wr_0,1 registers, data is written to this FIFO area.

If the JoinDMA bit is set, it is possible to ascertain the number of remaining data bytes when DMA_Control.Dir bit = 1 or the free space when DMA_Control.Dir bit = 0 by checking the DMA_Remain_H,L registers.

If the JoinCPU_Rd or JoinCPU_Wr bit is set, data can be read or written to or from the FIFO_Rd_0,1, FIFO_ByteRd, or FIFO_Wr_0,1 registers after checking FIFO_RdRemain_H,L or FIFO_WrRemain_H,L.

The JoinDMA, JoinCPU_Rd, and JoinCPU_Wr bits can be set to 1 only one bit at a time. If two or more of these bits are set by writing 1 at the same time, the register may not function properly.

### 7.4.113 0A1h AREA0Join_1 (AREA0 Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A1h | AREA0Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: JoinEPeCHe | 0: Do nothing | 1: Join to EPe / CHe | |
| | | | R/W | 4: JoinEPdCHd | 0: Do nothing | 1: Join to EPd / CHd | |
| | | | R/W | 3: JoinEPcCHc | 0: Do nothing | 1: Join to EPc / CHc | |
| | | | R/W | 2: JoinEPbCHb | 0: Do nothing | 1: Join to EPb / CHb | |
| | | | R/W | 1: JoinEPaCHa | 0: Do nothing | 1: Join to EPa / CHa | |
| | | | R/W | 0: JoinEP0CH0 | 0: Do nothing | 1: Join to EP0 / CH0 | |

This register sets the endpoint and channel to be connected to the AREA0 FIFO area.

**Bits7-6 Reserved**

**Bit5 JoinEPeCHe**

This bit connects the endpoint EPe or channel CHe to the AREA0 FIFO area. When the endpoint EPe or channel CHe is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit4 JoinEPdCHd**

This bit connects the endpoint EPd or channel CHd to the AREA0 FIFO area. When the endpoint EPd or channel CHd is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit3 JoinEPcCHc**

This bit connects the endpoint EPc or channel CHc to the AREA0 FIFO area. When the endpoint EPc or channel CHc is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit2 JoinEPbCHb**

This bit connects the endpoint EPb or channel CHb to the AREA0 FIFO area. When the endpoint EPb or channel CHb is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit1 JoinEPaCHa**

This bit connects the endpoint EPa or channel CHa to the AREA0 FIFO area. When the endpoint EPa or channel CHa is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit0 JoinEP0CH0**

This bit connects the endpoint EP0 or channel CH0 to the AREA0 FIFO area. When the endpoint EP0 or channel CH0 is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

Note that if two or more of JoinEPxCHx{x=0,a-e} bits are set for the same FIFO area at the same time, an unexpected operation may be performed depending on the order of transactions executed. In general, we recommend against setting two or more JoinEPxCHx{x=0,a-e} bits for the same FIFO area at the same time.

To use the USB device function, make sure endpoint EP0 is joined to the FIFO area of AREA0. To use the USB host function along with the control transfer support function, make sure channel CH0 is joined to the FIFO area of AREA0.

### 7.4.114 0A2h AREA1Join_0 (AREA1 Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A2h | AREA1Join_0 | R/W | 7: JoinFIFO_Stat | 0: Do nothing | 1: Join to FIFO Status | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: JoinDMA | 0: Do nothing | 1: Join to DMA | |
| | | | R/W | 1: JoinCPU_Rd | 0: Do nothing | 1: Join to CPU Read | |
| | | | R/W | 0: JoinCPU_Wr | 0: Do nothing | 1: Join to CPU Write | |

This register sets the port to be connected to the AREA0 FIFO area.

**Bit7 JoinFIFO_Stat**

This bit enables the full and empty status of the FIFO in the AREA1 FIFO area to be monitored by means of FIFO_IntStat.FIFO_NotEmpty, FIFO_IntStat.FIFO_Full, and FIFO_IntStat.FIFO_Empty.

**Bits6-3 Reserved**

**Bit2 JoinDMA**

This bit causes a DMA transfer to be performed with the FIFO in the AREA1 FIFO area. The transfer direction is determined by the DMA_Control.Dir bit.

**Bit1 JoinCPU_Rd**

This bit causes a read transfer for CPU register access to be performed with the FIFO in the AREA1 FIFO area. In other words, when a read of the FIFO_Rd_0,1 or FIFO_ByteRd register is performed, data is read from this FIFO area.

**Bit0 JoinCPU_Wr**

This bit causes a write transfer for CPU register access to be performed with the FIFO in the AREA1 FIFO area. That is, when a write is performed to the FIFO_Wr_0,1 registers, data is written to this FIFO area.

If the JoinDMA bit is set, it is possible to ascertain the number of remaining data bytes when DMA_Control.Dir bit = 1 or the free space when DMA_Control.Dir bit = 0 by checking the DMA_Remain_H,L registers.

If the JoinCPU_Rd or JoinCPU_Wr bit is set, data can be read or written to or from the FIFO_Rd_0,1, FIFO_ByteRd, or FIFO_Wr_0,1 registers after checking FIFO_RdRemain_H,L or FIFO_WrRemain_H,L.

The JoinDMA, JoinCPU_Rd, and JoinCPU_Wr bits can be set to 1 only one bit at a time. If two or more of these bits are set by writing 1 at the same time, the register may not function properly.

### 7.4.115 0A3h AREA1Join_1 (AREA1 Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A3h | AREA1Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: JoinEPeCHe | 0: Do nothing | 1: Join to EPe / CHe | |
| | | | R/W | 4: JoinEPdCHd | 0: Do nothing | 1: Join to EPd / CHd | |
| | | | R/W | 3: JoinEPcCHc | 0: Do nothing | 1: Join to EPc / CHc | |
| | | | R/W | 2: JoinEPbCHb | 0: Do nothing | 1: Join to EPb / CHb | |
| | | | R/W | 1: JoinEPaCHa | 0: Do nothing | 1: Join to EPa / CHa | |
| | | | R/W | 0: JoinEP0CH0 | 0: Do nothing | 1: Join to EP0 / CH0 | |

This register sets the endpoint and channel to be connected to the AREA1 FIFO area.

**Bits7-6 Reserved**

**Bit5 JoinEPeCHe**

This bit connects the endpoint EPe or channel CHe to the AREA1 FIFO area. When the endpoint EPe or channel CHe is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit4 JoinEPdCHd**

This bit connects the endpoint EPd or channel CHd to the AREA1 FIFO area. When the endpoint EPd or channel CHd is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit3 JoinEPcCHc**

This bit connects the endpoint EPc or channel CHc to the AREA1 FIFO area. When the endpoint EPc or channel CHc is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit2 JoinEPbCHb**

This bit connects the endpoint EPb or channel CHb to the AREA1 FIFO area. When the endpoint EPb or channel CHb is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit1 JoinEPaCHa**

This bit connects the endpoint EPa or channel CHa to the AREA1 FIFO area. When the endpoint EPa or channel CHa is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit0 JoinEP0CH0**

This bit connects the endpoint EP0 or channel CH0 to the AREA1 FIFO area. When the endpoint EP0 or channel CH0 is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

Note that if two or more of JoinEPxCHx{x=0,a-e} bits are set for the same FIFO area at the same time, an unexpected operation may be performed depending on the order of transactions executed. In general, we recommend against setting two or more JoinEPxCHx{x=0,a-e} bits for the same FIFO area at the same time.

To use the USB device function, make sure endpoint EPa is joined to the FIFO area of AREA1. To use the USB host function along with the bulk-only support function, make sure channel CHa is joined to the FIFO area of AREA1.

### 7.4.116 0A4h AREA2Join_0 (AREA2 Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A4h | AREA2Join_0 | R/W | 7: JoinFIFO_Stat | 0: Do nothing | 1: Join to FIFO Status | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: JoinDMA | 0: Do nothing | 1: Join to DMA | |
| | | | R/W | 1: JoinCPU_Rd | 0: Do nothing | 1: Join to CPU Read | |
| | | | R/W | 0: JoinCPU_Wr | 0: Do nothing | 1: Join to CPU Write | |

This register sets the port to be connected to the AREA2 FIFO area.

**Bit7 JoinFIFO_Stat**

This bit enables the full and empty status of the FIFO in the AREA2 FIFO area to be monitored by means of FIFO_IntStat.FIFO_NotEmpty, FIFO_IntStat.FIFO_Full, and FIFO_IntStat.FIFO_Empty.

**Bits6-3 Reserved**

**Bit2 JoinDMA**

This bit causes a DMA transfer to be performed with the FIFO in the AREA2 FIFO area. The transfer direction is determined by the DMA_Control.Dir bit.

**Bit1 JoinCPU_Rd**

This bit causes a read transfer for CPU register access to be performed with the FIFO in the AREA2 FIFO area. In other words, when a read of the FIFO_Rd_0,1 or FIFO_ByteRd register is performed, data is read from this FIFO area.

**Bit0 JoinCPU_Wr**

This bit causes a write transfer for CPU register access to be performed with the FIFO in the AREA2 FIFO area. That is, when a write is performed to the FIFO_Wr_0,1 registers, data is written to this FIFO area.

If the JoinDMA bit is set, it is possible to ascertain the number of remaining data bytes when DMA_Control.Dir bit = 1 or the free space when DMA_Control.Dir bit = 0 by checking the DMA_Remain_H,L registers.

If the JoinCPU_Rd or JoinCPU_Wr bit is set, data can be read or written to or from the FIFO_Rd_0,1, FIFO_ByteRd, or FIFO_Wr_0,1 registers after checking FIFO_RdRemain_H,L or FIFO_WrRemain_H,L.

The JoinDMA, JoinCPU_Rd, and JoinCPU_Wr bits can be set to 1 only one bit at a time. If two or more of these bits are set by writing 1 at the same time, the register may not function properly.

### 7.4.117 0A5h AREA2Join_1 (AREA2 Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A5h | AREA2Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: JoinEPeCHe | 0: Do nothing | 1: Join to EPe / CHe | |
| | | | R/W | 4: JoinEPdCHd | 0: Do nothing | 1: Join to EPd / CHd | |
| | | | R/W | 3: JoinEPcCHc | 0: Do nothing | 1: Join to EPc / CHc | |
| | | | R/W | 2: JoinEPbCHb | 0: Do nothing | 1: Join to EPb / CHb | |
| | | | R/W | 1: JoinEPaCHa | 0: Do nothing | 1: Join to EPa / CHa | |
| | | | R/W | 0: JoinEP0CH0 | 0: Do nothing | 1: Join to EP0 / CH0 | |

This register sets the endpoint and channel to be connected to the AREA2 FIFO area.

**Bits7-6 Reserved**

**Bit5 JoinEPeCHe**

This bit connects the endpoint EPe or channel CHe to the AREA2 FIFO area. When the endpoint EPe or channel CHe is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit4 JoinEPdCHd**

This bit connects the endpoint EPd or channel CHd to the AREA2 FIFO area. When the endpoint EPd or channel CHd is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit3 JoinEPcCHc**

This bit connects the endpoint EPc or channel CHc to the AREA2 FIFO area. When the endpoint EPc or channel CHc is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit2 JoinEPbCHb**

This bit connects the endpoint EPb or channel CHb to the AREA2 FIFO area. When the endpoint EPb or channel CHb is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit1 JoinEPaCHa**

This bit connects the endpoint EPa or channel CHa to the AREA2 FIFO area. When the endpoint EPa or channel CHa is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit0 JoinEP0CH0**

This bit connects the endpoint EP0 or channel CH0 to the AREA2 FIFO area. When the endpoint EP0 or channel CH0 is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

Note that if two or more of JoinEPxCHx{x=0,a-e} bits are set for the same FIFO area at the same time, an unexpected operation may be performed depending on the order of transactions executed. In general, we recommend against setting two or more JoinEPxCHx{x=0,a-e} bits for the same FIFO area at the same time.

To use the USB device function, make sure endpoint EPb is joined to the FIFO area of AREA2.

### 7.4.118 0A6h AREA3Join_0 (AREA3 Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A6h | AREA3Join_0 | R/W | 7: JoinFIFO_Stat | 0: Do nothing | 1: Join to FIFO Status | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: JoinDMA | 0: Do nothing | 1: Join to DMA | |
| | | | R/W | 1: JoinCPU_Rd | 0: Do nothing | 1: Join to CPU Read | |
| | | | R/W | 0: JoinCPU_Wr | 0: Do nothing | 1: Join to CPU Write | |

This register sets the port to be connected to the AREA3 FIFO area.

**Bit7 JoinFIFO_Stat**

This bit enables the full and empty status of the FIFO in the AREA3 FIFO area to be monitored by means of FIFO_IntStat.FIFO_NotEmpty, FIFO_IntStat.FIFO_Full, and FIFO_IntStat.FIFO_Empty.

**Bits6-3 Reserved**

**Bit2 JoinDMA**

This bit causes a DMA transfer to be performed with the FIFO in the AREA3 FIFO area. The transfer direction is determined by the DMA_Control.Dir bit.

**Bit1 JoinCPU_Rd**

This bit causes a read transfer for CPU register access to be performed with the FIFO in the AREA3 FIFO area. In other words, when a read of the FIFO_Rd_0,1 or FIFO_ByteRd register is performed, data is read from this FIFO area.

**Bit0 JoinCPU_Wr**

This bit causes a write transfer for CPU register access to be performed with the FIFO in the AREA3 FIFO area. That is, when a write is performed to the FIFO_Wr_0,1 registers, data is written to this FIFO area.

If the JoinDMA bit is set, it is possible to ascertain the number of remaining data bytes when DMA_Control.Dir bit = 1 or the free space when DMA_Control.Dir bit = 0 by checking the DMA_Remain_H,L registers.

If the JoinCPU_Rd or JoinCPU_Wr bit is set, data can be read or written to or from the FIFO_Rd_0,1, FIFO_ByteRd, or FIFO_Wr_0,1 registers after checking FIFO_RdRemain_H,L or FIFO_WrRemain_H,L.

The JoinDMA, JoinCPU_Rd, and JoinCPU_Wr bits can be set to 1 only one bit at a time. If two or more of these bits are set by writing 1 at the same time, the register may not function properly.

### 7.4.119 0A7h AREA3Join_1 (AREA3 Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A7h | AREA3Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: JoinEPeCHe | 0: Do nothing | 1: Join to EPe / CHe | |
| | | | R/W | 4: JoinEPdCHd | 0: Do nothing | 1: Join to EPd / CHd | |
| | | | R/W | 3: JoinEPcCHc | 0: Do nothing | 1: Join to EPc / CHc | |
| | | | R/W | 2: JoinEPbCHb | 0: Do nothing | 1: Join to EPb / CHb | |
| | | | R/W | 1: JoinEPaCHa | 0: Do nothing | 1: Join to EPa / CHa | |
| | | | R/W | 0: JoinEP0CH0 | 0: Do nothing | 1: Join to EP0 / CH0 | |

This register sets the endpoint and channel to be connected to the AREA3 FIFO area.

**Bits7-6 Reserved**

**Bit5 JoinEPeCHe**

This bit connects the endpoint EPe or channel CHe to the AREA3 FIFO area. When the endpoint EPe or channel CHe is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit4 JoinEPdCHd**

This bit connects the endpoint EPd or channel CHd to the AREA3 FIFO area. When the endpoint EPd or channel CHd is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit3 JoinEPcCHc**

This bit connects the endpoint EPc or channel CHc to the AREA3 FIFO area. When the endpoint EPc or channel CHc is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit2 JoinEPbCHb**

This bit connects the endpoint EPb or channel CHb to the AREA3 FIFO area. When the endpoint EPb or channel CHb is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit1 JoinEPaCHa**

This bit connects the endpoint EPa or channel CHa to the AREA3 FIFO area. When the endpoint EPa or channel CHa is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit0 JoinEP0CH0**

This bit connects the endpoint EP0 or channel CH0 to the AREA3 FIFO area. When the endpoint EP0 or channel CH0 is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

Note that if two or more of JoinEPxCHx{x=0,a-e} bits are set for the same FIFO area at the same time, an unexpected operation may be performed depending on the order of transactions executed. In general, we recommend against setting two or more JoinEPxCHx{x=0,a-e} bits for the same FIFO area at the same time.

To use the USB device function, make sure endpoint EPc is joined to the FIFO area of AREA3.

### 7.4.120 0A8h AREA4Join_0 (AREA4 Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A8h | AREA4Join_0 | R/W | 7: JoinFIFO_Stat | 0: Do nothing | 1: Join to FIFO Status | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: JoinDMA | 0: Do nothing | 1: Join to DMA | |
| | | | R/W | 1: JoinCPU_Rd | 0: Do nothing | 1: Join to CPU Read | |
| | | | R/W | 0: JoinCPU_Wr | 0: Do nothing | 1: Join to CPU Write | |

This register sets the port to be connected to the AREA4 FIFO area.

**Bit7 JoinFIFO_Stat**

This bit enables the full and empty status of the FIFO in the AREA4 FIFO area to be monitored by means of FIFO_IntStat.FIFO_NotEmpty, FIFO_IntStat.FIFO_Full, and FIFO_IntStat.FIFO_Empty.

**Bits6-3 Reserved**

**Bit2 JoinDMA**

This bit causes a DMA transfer to be performed with the FIFO in the AREA4 FIFO area. The transfer direction is determined by the DMA_Control.Dir bit.

**Bit1 JoinCPU_Rd**

This bit causes a read transfer for CPU register access to be performed with the FIFO in the AREA4 FIFO area. In other words, when a read of the FIFO_Rd_0,1 or FIFO_ByteRd register is performed, data is read from this FIFO area.

**Bit0 JoinCPU_Wr**

This bit causes a write transfer for CPU register access to be performed with the FIFO in the AREA4 FIFO area. That is, when a write is performed to the FIFO_Wr_0,1 registers, data is written to this FIFO area.

If the JoinDMA bit is set, it is possible to ascertain the number of remaining data bytes when DMA_Control.Dir bit = 1 or the free space when DMA_Control.Dir bit = 0 by checking the DMA_Remain_H,L registers.

If the JoinCPU_Rd or JoinCPU_Wr bit is set, data can be read or written to or from the FIFO_Rd_0,1, FIFO_ByteRd, or FIFO_Wr_0,1 registers after checking FIFO_RdRemain_H,L or FIFO_WrRemain_H,L.

The JoinDMA, JoinCPU_Rd, and JoinCPU_Wr bits can be set to 1 only one bit at a time. If two or more of these bits are set by writing 1 at the same time, the register may not function properly.

### 7.4.121 0A9h AREA4Join_1 (AREA4 Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0A9h | AREA4Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: JoinEPeCHe | 0: Do nothing | 1: Join to EPe / CHe | |
| | | | R/W | 4: JoinEPdCHd | 0: Do nothing | 1: Join to EPd / CHd | |
| | | | R/W | 3: JoinEPcCHc | 0: Do nothing | 1: Join to EPc / CHc | |
| | | | R/W | 2: JoinEPbCHb | 0: Do nothing | 1: Join to EPb / CHb | |
| | | | R/W | 1: JoinEPaCHa | 0: Do nothing | 1: Join to EPa / CHa | |
| | | | R/W | 0: JoinEP0CH0 | 0: Do nothing | 1: Join to EP0 / CH0 | |

This register sets the endpoint and channel to be connected to the AREA4 FIFO area.

**Bits7-6 Reserved**

**Bit5 JoinEPeCHe**

This bit connects the endpoint EPe or channel CHe to the AREA4 FIFO area. When the endpoint EPe or channel CHe is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit4 JoinEPdCHd**

This bit connects the endpoint EPd or channel CHd to the AREA4 FIFO area. When the endpoint EPd or channel CHd is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit3 JoinEPcCHc**

This bit connects the endpoint EPc or channel CHc to the AREA4 FIFO area. When the endpoint EPc or channel CHc is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit2 JoinEPbCHb**

This bit connects the endpoint EPb or channel CHb to the AREA4 FIFO area. When the endpoint EPb or channel CHb is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit1 JoinEPaCHa**

This bit connects the endpoint EPa or channel CHa to the AREA4 FIFO area. When the endpoint EPa or channel CHa is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit0 JoinEP0CH0**

This bit connects the endpoint EP0 or channel CH0 to the AREA4 FIFO area. When the endpoint EP0 or channel CH0 is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

Note that if two or more of JoinEPxCHx{x=0,a-e} bits are set for the same FIFO area at the same time, an unexpected operation may be performed depending on the order of transactions executed. In general, we recommend against setting two or more JoinEPxCHx{x=0,a-e} bits for the same FIFO area at the same time.

To use the USB device function, make sure endpoint EPd is joined to the FIFO area of AREA4.

### 7.4.122 0AAh AREA5Join_0 (AREA5 Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0AAh | AREA5Join_0 | R/W | 7: JoinFIFO_Stat | 0: Do nothing | 1: Join to FIFO Status | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: JoinDMA | 0: Do nothing | 1: Join to DMA | |
| | | | R/W | 1: JoinCPU_Rd | 0: Do nothing | 1: Join to CPU Read | |
| | | | R/W | 0: JoinCPU_Wr | 0: Do nothing | 1: Join to CPU Write | |

This register sets the port to be connected to the AREA5 FIFO area.

**Bit7 JoinFIFO_Stat**

This bit enables the full and empty status of the FIFO in the AREA5 FIFO area to be monitored by means of FIFO_IntStat.FIFO_NotEmpty, FIFO_IntStat.FIFO_Full, and FIFO_IntStat.FIFO_Empty.

**Bits6-3 Reserved**

**Bit2 JoinDMA**

This bit causes a DMA transfer to be performed with the FIFO in the AREA5 FIFO area. The transfer direction is determined by the DMA_Control.Dir bit.

**Bit1 JoinCPU_Rd**

This bit causes a read transfer for CPU register access to be performed with the FIFO in the AREA5 FIFO area. In other words, when a read of the FIFO_Rd_0,1 or FIFO_ByteRd register is performed, data is read from this FIFO area.

**Bit0 JoinCPU_Wr**

This bit causes a write transfer for CPU register access to be performed with the FIFO in the AREA5 FIFO area. That is, when a write is performed to the FIFO_Wr_0,1 registers, data is written to this FIFO area.

If the JoinDMA bit is set, it is possible to ascertain the number of remaining data bytes when DMA_Control.Dir bit = 1 or the free space when DMA_Control.Dir bit = 0 by checking the DMA_Remain_H,L registers.

If the JoinCPU_Rd or JoinCPU_Wr bit is set, data can be read or written to or from the FIFO_Rd_0,1, FIFO_ByteRd, or FIFO_Wr_0,1 registers after checking FIFO_RdRemain_H,L or FIFO_WrRemain_H,L.

The JoinDMA, JoinCPU_Rd, and JoinCPU_Wr bits can be set to 1 only one bit at a time. If two or more of these bits are set by writing 1 at the same time, the register may not function properly.

### 7.4.123 0ABh AREA5Join_1 (AREA5 Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0ABh | AREA5Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | R/W | 5: JoinEPeCHe | 0: Do nothing | 1: Join to EPe / CHe | |
| | | | R/W | 4: JoinEPdCHd | 0: Do nothing | 1: Join to EPd / CHd | |
| | | | R/W | 3: JoinEPcCHc | 0: Do nothing | 1: Join to EPc / CHc | |
| | | | R/W | 2: JoinEPbCHb | 0: Do nothing | 1: Join to EPb / CHb | |
| | | | R/W | 1: JoinEPaCHa | 0: Do nothing | 1: Join to EPa / CHa | |
| | | | R/W | 0: JoinEP0CH0 | 0: Do nothing | 1: Join to EP0 / CH0 | |

This register sets the endpoint and channel to be connected to the AREA5 FIFO area.

**Bits7-6 Reserved**

**Bit5 JoinEPeCHe**

This bit connects the endpoint EPe or channel CHe to the AREA5 FIFO area. When the endpoint EPe or channel CHe is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit4 JoinEPdCHd**

This bit connects the endpoint EPd or channel CHd to the AREA5 FIFO area. When the endpoint EPd or channel CHd is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit3 JoinEPcCHc**

This bit connects the endpoint EPc or channel CHc to the AREA5 FIFO area. When the endpoint EPc or channel CHc is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit2 JoinEPbCHb**

This bit connects the endpoint EPb or channel CHb to the AREA5 FIFO area. When the endpoint EPb or channel CHb is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit1 JoinEPaCHa**

This bit connects the endpoint EPa or channel CHa to the AREA5 FIFO area. When the endpoint EPa or channel CHa is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

**Bit0 JoinEP0CH0**

This bit connects the endpoint EP0 or channel CH0 to the AREA5 FIFO area. When the endpoint EP0 or channel CH0 is connected in this way, a transaction that accompanies data transfers can be executed using the connected endpoint or channel.

Note that if two or more of JoinEPxCHx{x=0,a-e} bits are set for the same FIFO area at the same time, an unexpected operation may be performed depending on the order of transactions executed. In general, we recommend against setting two or more JoinEPxCHx{x=0,a-e} bits for the same FIFO area at the same time.

To use the USB device function, make sure endpoint EPe is joined to the FIFO area of AREA5.

### 7.4.124 0AEh ClrAREAnJoin_0 (Clear AREA n Join 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0AEh | ClrAREAnJoin_0 | W | 7: ClrJoinFIFO_Stat | 0: Do nothing | 1: Clear JoinFIFO_Stat | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | W | 2: ClrJoinDMA | 0: Do nothing | 1: Clear JoinDMA | |
| | | | W | 1: ClrJoinCPU_Rd | 0: Do nothing | 1: Clear JoinCPU_Rd | |
| | | | W | 0: ClrJoinCPU_Wr | 0: Do nothing | 1: Clear JoinCPU_Wr | |

This register clears the connection of each FIFO area and a relevant port. This is a write-only register.

Any bit in this register is automatically cleared to 0 after the relevant FIFO and port connection is cleared.

While the FIFO area is connected to a relevant port (AREAn{n=0-5}Join_0 register's corresponding bit = 1) and the port is active, do not set the corresponding bit in this register to 1. The LSI may behave erratically.

### 7.4.125 0AFh ClrAREAnJoin_1 (Clear AREA n Join 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device / Host | 0AFh | ClrAREA n Join_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | W | 5: ClrJoinEPeCHe | 0: Do nothing | 1: Clear JoinEPeCHe | |
| | | | W | 4: ClrJoinEPdCHd | 0: Do nothing | 1: Clear JoinEPdCHd | |
| | | | W | 3: ClrJoinEPcCHc | 0: Do nothing | 1: Clear JoinEPcCHc | |
| | | | W | 2: ClrJoinEPbCHb | 0: Do nothing | 1: Clear JoinEPbCHb | |
| | | | W | 1: ClrJoinEPaCHa | 0: Do nothing | 1: Clear JoinEPaCHa | |
| | | | W | 0: ClrJoinEP0CH0 | 0: Do nothing | 1: Clear JoinEP0CH0 | |

This register clears the connection of each FIFO area and a relevant endpoint or channel. This is a write-only register.

Any bit in this register is automatically cleared to 0 after the relevant FIFO and port connection is cleared.

While the FIFO area is connected to a relevant endpoint or channel (AREAn{n=0-5}Join_1 register's corresponding bit = 1) and a transaction is being executed at the endpoint or channel, do not set the corresponding bit in this register to 1. The LSI may behave erratically.

## 7.5 Detailed Description of Device Registers

### 7.5.1 0Bh *D_SIE_IntStat (Device SIE Interrupt Status)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0Bh | ***D_SIE_IntStat*** | | 7: | 0: | 1: | 00h |
| | | | R (W) | 6: ***NonJ*** | 0: None | 1: Detect Non J state | |
| | | | R (W) | 5: RcvSOF | 0: None | 1: Received SOF | |
| | | | 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 device SIE.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 Reserved**

**Bit6 NonJ**

This bit indicates the cause of interrupt directly.

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

**Bit5 RcvSOF**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an SOF token is received.

**Bit4 DetectReset**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a reset state of the USB is 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 D_NegoControl register remains set.

During HS operation mode, when no bus activity is detected for a predetermined time, the 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 not using the AutoNegotiation function, it is necessary to prevent erroneous detection of subsequent resets after this bit is set to 1. Therefore, set the DisBusDetect bit in the D_NegoControl register to 1 to disable the detection of reset/suspend states of the USB. After processing for the reset is complete, clear the DisBusDetect bit to 0 to reenable the detection of reset/suspend states of the USB.

**Bit3 DetectSuspend**

This bit indicates the cause of interrupt directly.

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

During HS operation mode, when no bus activity is detected for a predetermined 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 Sleep mode (internal PLL and Oscillator turned off) by setting the GoSleep bit in the PM_Control register to 1.

**Bit2 ChirpCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when HS Detection Handshaking that was started by the GoChirp bit in the D_NegoControl register is complete.

The current operation mode (FS or HS) can be determined by reading the FSxHS bit in the D_USB_Status register after the interrupt occurred.

**Bit1 RestoreCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when the Restore processing that was started by RestoreUSB bit in the D_NegoControl register is complete. 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 interrupt directly.

When a SetAddress() request is received, the AutoSetAddress function (see "USB_Address register") automatically performs processing for the control transfer needed. 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 D_USB_Address register.

Even when power management is ACTIVE, the synchronous bits (bits 5 to 0) cannot be read out or written to (to clear the cause of the interrupt) unless the HostDeviceSel.HOSTxDEVICE bit = 0 (i.e., device mode). Therefore, if a transition from this state is required, the processing described below should be executed from firmware to ensure that the XINT interrupt signal is not inadvertently asserted by these interrupt statuses.

<For transition from device mode while ACTIVE>

1) Process the interrupt status and clear it (D_SIE_IntStat.Bit5 through 0)

2) Disable the interrupt status (D_SIE_IntEnb.Bit5 through 0)

<For transition to device mode while ACTIVE>

3) Clear the interrupt status (D_SIE_IntStat.Bit5 through 0)

4) Reenable the interrupt status (D_SIE_IntEnb.Bit5 through 0)

### 7.5.2 0Bh D_BulkIntStat (Device Bulk Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0Bh | D_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 interrupt directly.

This bit is set to 1 when 31 bytes of CBW is received without errors.

**Bit6 CBW_LengthErr**

This bit indicates the cause of interrupt directly.

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

**Bit5 CBW_Err**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a CRC error or other transaction error is detected in the received CBW.

**Bit4 Reserved**

**Bit3 CSW_Cmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when 31 bytes of CSW is received without errors.

**Bit2 CSW_Err**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a CSW transmit error (not responded by ACK) occurrs.

**Bits1-0 Reserved**

### 7.5.3 0Bh D_EPrIntStat (Device EPr Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0Bh | D_EPrIntStat | R | 7: D_AlarmIN_IntStat | 0: None | 1: Alarm IN Interrupt | 00h |
| | | | R | 6: D_AlarmOUT_IntStat | 0: None | 1: Alarm OUT Interrupt | |
| | | | | 5: | 0: | 1: | |
| | | | R | 4: D_EPeIntStat | 0: None | 1: EPe Interrupt | |
| | | | R | 3: D_EPeIntStat | 0: None | 1: EPd Interrupt | |
| | | | R | 2: D_EPcIntStat | 0: None | 1: EPc Interrupt | |
| | | | R | 1: D_EPbIntStat | 0: None | 1: EPb Interrupt | |
| | | | R | 0: D_EPaIntStat | 0: None | 1: EPa Interrupt | |

This register shows the interrupts of the endpoint Epr{r=a-e}, and AlarmIN/AlarmOUT.

**Bit7 D_AlarmIN_IntStat**

This bit indicates the cause of interrupt indirectly.

If the D_AlarmIN_IntStat_H, L register has the cause of interrupt and the bit corresponding to that interrupt cause in the D_AlarmIN_IntEnb_H, L register is enabled, this bit is set to 1.

**Bit6 D_AlarmOUT_IntStat**

This bit indicates the cause of interrupt indirectly.

If the D_AlarmOUT_IntStat_H, L register has the cause of interrupt and the bit corresponding to that interrupt cause in the D_AlarmOUT_IntEnb_H, L register is enabled, this bit is set to 1.

**Bits5 Reserved**

**Bit4 D_EPeIntStat**

This bit indicates the cause of interrupt indirectly.

If the D_EPeIntStat_H, L register has the cause of interrupt and the bit corresponding to that interrupt cause in the D_EPeIntEnb_H, L register is enabled, this bit is set to 1.

**Bit3 D_EPdIntStat**

This bit indicates the cause of interrupt indirectly.

If the D_EPdIntStat_H, L register has the cause of interrupt and the bit corresponding to that interrupt cause in the D_EPdIntEnb_H, L register is enabled, this bit is set to 1.

**Bit2 D_EPcIntStat**

This bit indicates the cause of interrupt indirectly.

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

**Bit1** **D_EPbIntStat**

This bit indicates the cause of interrupt indirectly.

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

**Bit0** **D_EPaIntStat**

This bit indicates the cause of interrupt indirectly.

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

### 7.5.4 0B5h D_EP0IntStat (Device EP0 Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0B5h | D_EP0IntStat | R (W) | 7: DescriptorCmp | 0: None | 1: Descriptor Complete | 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 DescriptorCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in the descriptor reply function, the LSI has finished sending a number of data bytes set by the DescriptorSize register back to the host.

If a transition to the status stage occurs (OUT token received) before the data bytes set by the DescriptorSize register have all been returned, the OUT_TranNAK bit and this status bit are both set to 1.

**Bit6 OUT_ShortACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 at the same time OUT_TranACK is set when ACK is returned for the short packet received in an OUT transaction.

**Bit5 IN_TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when ACK is received in an IN transaction.

**Bit4 OUT_TranACK**

This bit indicates the cause of interrupt directly.

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

**Bit3 IN_TranNAK**

This bit indicates the cause of interrupt directly.

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

**Bit2** **OUT_TranNAK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when NAK is returned to the host for the OUT or PING transaction attempted.

**Bit1** **IN_TranErr**

This bit indicates the cause of interrupt directly.

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

**Bit0** **OUT_TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in an OUT transaction, STALL is returned or an error in the packet is found.

### 7.5.5 0B6h D_EPaIntStat (Device EPa Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0B6h | D_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 interrupt directly.

This bit is set to 1 at the same time OUT_TranACK is set when ACK is returned for the short packet received in an OUT transaction.

**Bit5 IN_TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when ACK is received in an IN transaction.

**Bit4 OUT_TranACK**

This bit indicates the cause of interrupt directly.

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

**Bit3 IN_TranNAK**

This bit indicates the cause of interrupt directly.

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

**Bit2 OUT_TranNAK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when NAK is returned to the host for the OUT or PING transaction attempted.

**Bit1 IN_TranErr**

This bit indicates the cause of interrupt directly.

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

**Bit0 OUT_TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in an OUT transaction, STALL is returned or an error in the packet is found.

### 7.5.6 0B7h D_EPbIntStat (Device EPb Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0B7h | D_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 status 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 interrupt directly.

This bit is set to 1 at the same time OUT_TranACK is set when ACK is returned for the short packet received in an OUT transaction.

**Bit5 IN_TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when ACK is received in an IN transaction.

**Bit4 OUT_TranACK**

This bit indicates the cause of interrupt directly.

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

**Bit3 IN_TranNAK**

This bit indicates the cause of interrupt directly.

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

**Bit2 OUT_TranNAK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when NAK is returned to the host for the OUT or PING transaction attempted.

**Bit1 IN_TranErr**

This bit indicates the cause of interrupt directly.

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

**Bit0 OUT_TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in an OUT transaction, STALL is returned or an error in the packet is found.

### 7.5.7 0B8h D_EPcIntStat (D_EPc Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0B8h | D_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 status 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 interrupt directly.

This bit is set to 1 at the same time OUT_TranACK is set when ACK is returned for the short packet received in an OUT transaction.

**Bit5 IN_TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when ACK is received in an IN transaction.

**Bit4 OUT_TranACK**

This bit indicates the cause of interrupt directly.

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

**Bit3 IN_TranNAK**

This bit indicates the cause of interrupt directly.

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

**Bit2 OUT_TranNAK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when NAK is returned to the host for the OUT or PING transaction attempted.

**Bit1 IN_TranErr**

This bit indicates the cause of interrupt directly.

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

**Bit0 OUT_TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in an OUT transaction, STALL is returned or an error in the packet is found.

### 7.5.8 0B9h D_EPdIntStat (D_EPd Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0B9h | D_EPdIntStat | | 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 status of the endpoint EPd. 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 interrupt directly.

This bit is set to 1 at the same time OUT_TranACK is set when ACK is returned for the short packet received in an OUT transaction.

**Bit5 IN_TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when ACK is received in an IN transaction.

**Bit4 OUT_TranACK**

This bit indicates the cause of interrupt directly.

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

**Bit3 IN_TranNAK**

This bit indicates the cause of interrupt directly.

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

**Bit2 OUT_TranNAK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when NAK is returned to the host for the OUT or PING transaction attempted.

**Bit1 IN_TranErr**

This bit indicates the cause of interrupt directly.

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

**Bit0 OUT_TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in an OUT transaction, STALL is returned or an error in the packet is found.

### 7.5.9 0BAh D_EPeIntStat (D_EPe Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0BAh | D_EPeIntStat | | 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 status of the endpoint EPe. 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 interrupt directly.

This bit is set to 1 at the same time OUT_TranACK is set when ACK is returned for the short packet received in an OUT transaction.

**Bit5 IN_TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when ACK is received in an IN transaction.

**Bit4 OUT_TranACK**

This bit indicates the cause of interrupt directly.

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

**Bit3 IN_TranNAK**

This bit indicates the cause of interrupt directly.

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

**Bit2 OUT_TranNAK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when NAK is returned to the host for the OUT or PING transaction attempted.

**Bit1 IN_TranErr**

This bit indicates the cause of interrupt directly.

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

**Bit0 OUT_TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when in an OUT transaction, STALL is returned or an error in the packet is found.

### 7.5.10 0BCh D_AlarmIN_IntStat_H (Device AlarmIN Interrupt Status High)

### 7.5.11 0BDh D_AlarmIN_IntStat_L (Device AlarmIN Interrupt Status Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0BCh | D_AlarmIN | R(W) | 7: AlarmEP15IN | 0: Do nothing | 1: EP15 received IN token | 00h |
| | | _IntStat_H | R(W) | 6: AlarmEP14IN | 0: Do nothing | 1: EP14 received IN token | |
| | | | R(W) | 5: AlarmEP13IN | 0: Do nothing | 1: EP13 received IN token | |
| | | | R(W) | 4: AlarmEP12IN | 0: Do nothing | 1: EP12 received IN token | |
| | | | R(W) | 3: AlarmEP11IN | 0: Do nothing | 1: EP11 received IN token | |
| | | | R(W) | 2: AlarmEP10IN | 0: Do nothing | 1: EP10 received IN token | |
| | | | R(W) | 1: AlarmEP9IN | 0: Do nothing | 1: EP9 received IN token | |
| | | | R(W) | 0: AlarmEP8IN | 0: Do nothing | 1: EP8 received IN token | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0BDh | D_AlarmIN | R(W) | 7: AlarmEP7IN | 0: Do nothing | 1: EP7 received IN token | 00h |
| | | _IntStat_L | R(W) | 6: AlarmEP6IN | 0: Do nothing | 1: EP6 received IN token | |
| | | | R(W) | 5: AlarmEP5IN | 0: Do nothing | 1: EP5 received IN token | |
| | | | R(W) | 4: AlarmEP4IN | 0: Do nothing | 1: EP4 received IN token | |
| | | | R(W) | 3: AlarmEP3IN | 0: Do nothing | 1: EP3 received IN token | |
| | | | R(W) | 2: AlarmEP2IN | 0: Do nothing | 1: EP2 received IN token | |
| | | | R(W) | 1: AlarmEP1IN | 0: Do nothing | 1: EP1 received IN token | |
| | | | R(W) | 0: AlarmEP0IN | 0: Do nothing | 1: EP0 received IN token | |

These registers indicate the Alarm IN interrupt status. The cause of an interrupt can be cleared by writing 1 to the corresponding bit in these registers.

All bits in these registers indicate the cause of the interrupt.

If a transaction is issued (IN token received) from the host to one of the endpoints enabled by D_EnEP_IN_H,L that are appropriately set for the D_EPx{x=0,a-e}-related registers, except for those joined by the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit, the response described below is returned, and the corresponding bit in these registers is set to 1. However, note that since the endpoint EP0 is always enabled, the process described above applies even when the registers related to D_EP0 are not appropriately set or when the AREAn{n=0-5}Join.JoinEP0CH0 bit is not set for any FIFO area.

The response returned to the host for the IN token depends on how D_EnEP_IN_ISO_H,L is set. For endpoints whose relevant bit is set to 1, the response to the IN token is a zero-length packet; for endpoints whose relevant bit is cleared to 0, the response to the IN token is NAK.

If any bit is set in these registers corresponding to an endpoint, set the registers related to D_EPx{x=0,a-e} appropriately. Join the endpoint to the FIFO area using the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit to make a transaction executable.

### 7.5.12 0BEh D_AlarmOUT_IntStat_H (Device AlarmOUT Interrupt Status High)

### 7.5.13 0BFh D_AlarmOUT_IntStat_L (Device AlarmOUT Interrupt Status Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0BCh | D_AlarmOUT | R(W) | 7: AlarmEP15OUT | 0: Do nothing | 1: EP15 received OUT token | 00h |
| | | _IntStat_H | R(W) | 6: AlarmEP14IOUT | 0: Do nothing | 1: EP14 received OUT token | |
| | | | R(W) | 5: AlarmEP13OUT | 0: Do nothing | 1: EP13 received OUT token | |
| | | | R(W) | 4: AlarmEP12OUT | 0: Do nothing | 1: EP12 received OUT token | |
| | | | R(W) | 3: AlarmEP11OUT | 0: Do nothing | 1: EP11 received OUT token | |
| | | | R(W) | 2: AlarmEP10OUT | 0: Do nothing | 1: EP10 received OUT token | |
| | | | R(W) | 1: AlarmEP9OUT | 0: Do nothing | 1: EP9 received OUT token | |
| | | | R(W) | 0: AlarmEP8OUT | 0: Do nothing | 1: EP8 received OUT token | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0BDh | D_AlarmIN | R(W) | 7: AlarmEP7OUT | 0: Do nothing | 1: EP7 received OUT token | 00h |
| | | _IntStat_L | R(W) | 6: AlarmEP6OUT | 0: Do nothing | 1: EP6 received OUT token | |
| | | | R(W) | 5: AlarmEP5OUT | 0: Do nothing | 1: EP5 received OUT token | |
| | | | R(W) | 4: AlarmEP4OUT | 0: Do nothing | 1: EP4 received OUT token | |
| | | | R(W) | 3: AlarmEP3OUT | 0: Do nothing | 1: EP3 received OUT token | |
| | | | R(W) | 2: AlarmEP2OUT | 0: Do nothing | 1: EP2 received OUT token | |
| | | | R(W) | 1: AlarmEP1OUT | 0: Do nothing | 1: EP1 received OUT token | |
| | | | R(W) | 0: AlarmEP0OUT | 0: Do nothing | 1: EP0 received OUT token | |

These registers indicate the Alarm OUT interrupt status. The cause of interrupt can be cleared by writing 1 to the corresponding bit in these registers.

All bits in these registers indicate the cause of the interrupt.

If a transaction is issued (IN token received) from the host to one of the endpoints enabled by D_EnEP_OUT_H,L that are appropriately set for the D_EPx{x=0,a-e}-related registers, except for those joined by the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit, the response described below is returned, and the corresponding bit in these registers is set to 1. However, note that since the endpoint EP0 is always enabled, what is described above applies even when the registers related to D_EP0 are not appropriately set or when the AREAn{n=0-5}Join.JoinEP0CH0 bit is not set for any FIFO area.

The response returned to the host for the OUT token depends on how D_EnEP_OUT_ISO_H,L is set. For endpoints whose relevant bit is set to 1, the data sent from the host is not received, and no response is issued for the handshake sequence; for endpoints whose relevant bit is cleared to 0, an NAK response is issued for the OUT token. If a PING token is issued from the host while the device is set to HS, an NAK response is issued for the token.

If any bit in these registers corresponding to an endpoint is set, set the D_EPx{x=0,a-e}-related registers appropriately. Join the endpoint to the FIFO area using the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit to make a transaction executable.

### 7.5.14 0C0h *D_SIE_IntEnb (Device SIE Interrupt Enable)*

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C0h | ***D_SIE_IntEnb*** | | 7: | 0: | 1: | 00h |
| | | | R / W | 6: ***EnNonJ*** | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnRcvSOF | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnDetectRESET | 0: Disable | 1: Enable | |
| | | | R / W | 3: 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 D_SIE_IntStat bit in the MainIntStat register for the interrupt causes accommodated in the D_SIE_IntStat register.

The EnNonJ bit in this register is effective even during Sleep.

### 7.5.15 0C3h D_BulkIntEnb (Device Bulk Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C3h | D_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 | 1: Enable | |
| | | | | 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 D_BulkIntStat bit in the MainIntStat register for the interrupt causes accommodated in the D_BulkIntStat register.

### 7.5.16 0C4h D_EPrIntEnb (Device EPr Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C4h | D_EPrIntEnb | R / W | 7: EnD_AlarmIN_IntStat | 0: | 1: | 00h |
| | | | R / W | 6: EnD_AlarmOUT_IntStat | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: EnD_EPeIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnD_EPdIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnD_EnEPcIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnD_EnEPbIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnD_EnEPaIntStat | 0: Disable | 1: Enable | |

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

### 7.5.17 0C5h D_EP0IntEnb (Device EP0 Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C5h | D_EP0IntEnb | R / W | 7: EnDescriptorCmp | 0: Disable | 1: Enable | 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 D_EP0IntStat bit in the MainIntStat register for the interrupt causes accommodated in the D_EP0IntStat register.

### 7.5.18 0C6h D_EPaIntEnb (DeviceEPa Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C6h | D_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 D_EPrIntStat register for the interrupt causes accommodated in the D_EPaIntStat register.

### 7.5.19 0C7h D_EPbIntEnb (Device EPb Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C7h | D_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 D_EPrIntStat register for the interrupt causes accommodated in the D_EPbIntStat register.

### 7.5.20 C8h D_EPcIntEnb (Device EPc Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | C8h | D_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 D_EPrIntStat register for the interrupt causes accommodated in the D_EPcIntStat register.

### 7.5.21 0C9h D_EPdIntEnb (Device EPd Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0C9h | D_EPdIntEnb | | 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 EPdIntStat bit in the D_EPrIntStat register for the interrupt causes accommodated in the D_EPdIntStat register.

### 7.5.22 0CAh D_EPeIntEnb (Device EPe Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0CAh | D_EPeIntEnb | | 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 D_EPeIntStat bit in the D_EPrIntStat register for the interrupt causes accommodated in the D_EPeIntStat register.

### 7.5.23 0CCh D_AlarmIN_IntEnb_H (Device AlarmIN Interrupt Enable High)

### 7.5.24 0CDh D_AlarmIN_IntEnb_L (Device AlarmIN Interrupt Enable Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0CCh | D_AlarmIN | R / W | 7: EnAlarmEP15IN | 0: Disable | 1: Enable | 00h |
| | | _IntEnb_H | R / W | 6: EnAlarmEP14IN | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnAlarmEP13IN | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnAlarmEP12IN | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnAlarmEP11IN | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnAlarmEP10IN | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnAlarmEP9IN | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnAlarmEP8IN | 0: Disable | 1: Enable | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0CDh | D_AlarmIN | R / W | 7: EnAlarmEP7IN | 0: Disable | 1: Enable | 00h |
| | | _IntEnb_L | R / W | 6: EnAlarmEP6IN | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnAlarmEP5IN | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnAlarmEP4IN | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnAlarmEP3IN | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnAlarmEP2IN | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnAlarmEP1IN | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnAlarmEP0IN | 0: Disable | 1: Enable | |

This register is used to enable or disable the assertion of the AlarmIN_IntStat bit in the D_EPrIntStat register for the interrupt causes accommodated in the D_AlarmIN_IntStat register.

### 7.5.25 0CEh D_AlarmOUT_IntEnb_H (Device AlarmOUT Interrupt Enable High)

### 7.5.26 0CFh D_AlarmOUT_IntEnb_L (Device AlarmOUT Interrupt Enable Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0CEh | D_AlarmOUT | R / W | 7: EnAlarmEP15OUT | 0: Disable | 1: Enable | 00h |
| | | _IntEnb_H | R / W | 6: EnAlarmEP14OUT | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnAlarmEP13OUT | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnAlarmEP12OUT | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnAlarmEP11OUT | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnAlarmEP10OUT | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnAlarmEP9OUT | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnAlarmEP8OUT | 0: Disable | 1: Enable | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0CFh | D_AlarmOUT | R / W | 7: EnAlarmEP7OUT | 0: Disable | 1: Enable | 00h |
| | | _IntEnb_L | R / W | 6: EnAlarmEP6OUT | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnAlarmEP5OUT | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnAlarmEP4OUT | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnAlarmEP3OUT | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnAlarmEP2OUT | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnAlarmEP1OUT | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnAlarmEP0OUT | 0: Disable | 1: Enable | |

This register is used to enable or disable the assertion of the AlarmOUT_IntStat bit in the D_EPrIntStat register for the interrupt causes accommodated in the D_AlarmOUT_IntStat register.

### 7.5.27 0D0h D_NegoControl (Device Nego Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0D0h | D_NegoControl | 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 operations associated with device negotiation.

**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.

During HS mode, if no bus activities are detected for a 3 ms period, operation mode is automatically switched to FS mode and then determination is made to see if the USB is in a reset or suspend state, according to which the relevant cause of interrupt (DetectReset or DetectSuspend) is set. During FS mode, if no bus activities are detected for a 3 ms period, a suspend state of the USB is assumed. Furthermore, if SE0 in duration of 2.5 μs or more is detected, a reset is assumed, and the relevant cause of interrupt is set.

When the DetectReset or DetectSuspend bit is set, set the DisBusDetect bit to 1 to disable the detection of states while the reset or suspend state of the USB continues. When using the AutoNegotiation function, do not set this bit to 1.

**Bit6 EnAutoNego**

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

**Bit5 InSUSPEND**

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

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

**Bit4 DisableHS**

If this bit is 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 1 ms or more but within 15 ms have elapsed after the LSI started sending out the RemoteWakeup signal, clear this bit to 0 to stop the transmission.

**Bit2 RestoreUSB**

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

This bit is automatically cleared to 0 after the operation is complete.

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

**Bit1 GoChirp**

If this bit is set to 1 while the USB bus is in a reset status, 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 interrupt (ChirpCmp) is set at the same time the operation is complete.

This bit is automatically cleared to 0 after the operation is complete. 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 because the function of this bit is automatically controlled during that time.

**Bit0 ActiveUSB**

Since this bit remains cleared to 0 after the LSI is reset in hardware, the entire function of the USB is disabled. Therefore, after setting up the LSI, set this bit to 1 to enable the USB.

### 7.5.28 0D3h D_XcvrControl (Device Xcvr Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0D3 | D_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 device transceiver macro.

**Bit7 TermSelect**

This bit selects FS or HS termination to enable it. This bit is automatically set when HS Detection Handshaking is executed via the GoChirp bit in the USB_Control register or when the AutoNegotiation function is executed after the EnAutoNego bit in the D_NegoControl register is set.

**Bit6 XcvrSelect**

This bit selects the FS or HS transceiver to enable it. This bit is automatically set when HS Detection Handshaking is executed via the GoChirp bit in the USB_Control register or when the AutoNegotiation function is executed after the EnAutoNego bit in the D_NegoControl register is set.

**Bits5-2 Reserved**

**Bits1-0 OpMode**

These bits set the operation mode of the MTM.

These bits are normally not required to be set unless the USB cable is removed (*), or unless the USB is in a suspend state or during test mode.

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

* It is recommended to set this register to “41h” if the USB cable is not connected.

### 7.5.29 0D4h D_USB_Test (Device USB_Test)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0D4h | D_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 during USB device operation. To perform any test mode defined in the USB2.0 standard, set the bit corresponding to the test mode specified in a SetFeature request and after the status stage is complete, set the EnHS_Test bit to 1.

**Bit7 EnHS_Test**

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

Also make sure that shift to the test mode is made after the status stage in a SetFeature request is complete.

**Bits6-4 Reserved**

**Bit3 Test_SE0_NAK**

The LSI can be shifted to the Test_SE0_NAK test mode by setting this bit to 1 and then the EnHS_Test bit to 1.

**Bit2 TEST_J**

The LSI can be shifted to the Test_J test mode by setting this bit to 1 and then the EnHS_Test 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**

The LSI can be shifted to the Test_K test mode by setting this bit to 1 and then the EnHS_Test 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**

The LSI can be shifted to the Test_Packet test mode by setting this bit to 1 and then the EnHS_Test bit to 1.

Since this test mode can be used at any endpoint other than EP0, following settings need to be made:

1) Set MaxPacketSize and the direction of transfer for the endpoint EPx{x=a-e} to 64 or more and IN, respectively, and then set EndpointNumber to "0xF" to make the endpoint usable. Remember to allocate 64 bytes or more of storage to the FIFO of the endpoint EPx{x=a-e}.
2) Make sure that other endpoint settings do not overlap the above EPx{x=a-e} setting. Or clear the AREAx{x=1-5}Join_1.JoinEPxCHx{x=a-e} bit.
3) Clear the FIFO for EPx{x=a-e} and write the data for test packet shown below into the FIFO. Clear the IN_TranErr bit in the D_EPx{x=a-e}IntStat register to 0.
4) Each time test packet transmission is complete, the IN_TranErr status is set to 1.

The following 53 bytes are the data to be written to the FIFO during 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

Since the SIE adds PID and CRC 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 next to DATA0 PID to those that follow but not including CRC16.

### 7.5.30 0D6h D_EPnControl (Device Endpoint Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0D6h | D_EPnControl | W | 7: AllForceNAK | 0: Do nothing | 1: Set All ForceNAK | XXh |
| | | | W | 6: EPrForceSTALL | 0: Do nothing | 1: Set EP's ForceSTALL | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

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

**Bit7 AllForceNAK**

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

**Bit6 EPrForceSTALL**

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

**Bits5-0 Reserved**

### 7.5.31 0D8h D_BulkOnlyControl (Device BulkOnly Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0D8h | D_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: None | 1: Begin CBW Mode | |
| | | | R / W | 1: GoCSW_Mode | 0: None | 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 be received for CBW support is complete.

**Bits6-3 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 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 BulkOnlyConfig register.

**Bit0 Reserved**

### 7.5.32 0D9h D_BulkOnlyConfig (Device BulkOnly Configuration)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0D9h | D_BulkOnlyConfig | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: EPeBulkOnly | 0: None | 1: Enable BulkOnly on EPe | |
| | | | R / W | 3: EPdBulkOnly | 0: None | 1: Enable BulkOnly on EPd | |
| | | | R / W | 2: EPcBulkOnly | 0: None | 1: Enable BulkOnly on EPc | |
| | | | R / W | 1: EPbBulkOnly | 0: None | 1: Enable BulkOnly on EPb | |
| | | | R / W | 0: EPaBulkOnly | 0: None | 1: Enable BulkOnly on EPa | |

This register enables the Bulk Only Support function.

**Bits7-5 Reserved**

**Bit4 EPeBulkOnly**

Setting this bit to 1 enables the Bulk Only Support function for the endpoint EPe. If the endpoint EPe is an OUT endpoint when the Bulk Only Support function is enabled this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Or, if the endpoint EPe 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.

**Bit3 EPdBulkOnly**

Setting this bit to 1 enables the Bulk Only Support function for the endpoint EPd. If the endpoint EPd is an OUT endpoint when the Bulk Only Support function is enabled this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Or, if the endpoint EPd 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.

**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 this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Or, 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 this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Or, 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 EPaBulkOnly**

Setting this bit to 1 enables the Bulk Only Support function for the endpoint EPa. If the endpoint EPa is an OUT endpoint when the Bulk Only Support function is enabled this way, CBW support can be executed by setting the BulkOnlyControl.GoCBW_Mode bit. Or, if the endpoint EPa 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.

### 7.5.33 0E0h D_EP0SETUP_0 (Device EP0 SETUP 0)
### 7.5.34 0E1h D_EP0SETUP_1 (Device EP0 SETUP 1)
### 7.5.35 0E2h D_EP0SETUP_2 (Device EP0 SETUP 2)
### 7.5.36 0E3h D_EP0SETUP_3 (Device EP0 SETUP 3)
### 7.5.37 0E4h D_EP0SETUP_4 (Device EP0 SETUP 4)
### 7.5.38 0E5h D_EP0SETUP_5 (Device EP0 SETUP 5)
### 7.5.39 0E6h D_EP0SETUP_6 (Device EP0 SETUP 6)
### 7.5.40 0E7h D_EP0SETUP_7 (Device EP0 SETUP 7)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 0E0h<br>-0E7h | D_EP0SETUP_0<br>-D_EP0SETUP_7 | R | 7: EP0SETUP_n [7] | Endpoint 0 SETUP Data 0<br>-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.

### 7.5.41 0E8h D_USB_Address (Device USB Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0E8h | D_USB_Address | | 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 complete, the AutoSetAddress function sets USB_Address and then issues a SetAddressCmp status.

**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.

**Bits6-0 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 software, but when a SetAddress() request is received, the value will be automatically rewritten.

### 7.5.42 0EAh D_SETUP_Control(Device SETUP Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0EAh | D_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 settings associated with control transfers.

**Bits7-1 Reserved**

**Bit0 ProtectEP0**

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

At the same time, the ForceSTALL bits in the D_EP0ControlIN and D_EP0ControlOUT registers are cleared to 0 and the ForceNAK and ToggleStat bits in those registers both are set to 1, all automatically.

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

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

### 7.5.43 0EEh D_FrameNumber_H (Device FrameNumber High)

### 7.5.44 0EFh D_FrameNumber_L (Device FrameNumber Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0EEh | D_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] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 0EFh | D_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 get a frame number, the FrameNumber_H and FrameNumber_L registers must be accessed in pairs. In that case, be sure to access the FrameNumber_H register first.

**0EEh.Bit7 FnInvalid**

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

**0EEh.Bit6-3 Reserved**

**0EEh.Bit2-0, 0EFh.Bit7-0 FrameNumber [10:0]**

These bits show the FrameNumber of the received SOF packet.

### 7.5.45 0F0h D_EP0MaxSize (Device EP0 Max Packet Size)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0F0h | D_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**

**Bits6-3 EP0MaxSize [6:3]**

These bits set the MaxPacketSize of the endpoint EP0.

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

During FS 8, 16, 32, or 64 bytes

During HS 64 bytes

**Bits2-0 Reserved**

### 7.5.46 0F1h D_EP0Control (Device EP0 Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0F1h | D_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 is involved, set the direction of transfer at the data stage in this bit. When a setup stage is complete, the ForceNAK bits in the D_EP0ControlIN and D_EP0ControlOUT registers are set, so be sure to clear these bits before executing the data and status stages.

When the data stage is complete, 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 for OUT, so set this bit to 0. Conversely, if the direction of the data stage is OUT or a data stage is not involved, the status stage is directed for IN, so clear the FIFO for the endpoint EP0 and set this bit to 1.

An IN or OUT transaction in the direction opposite to the set value of this bit is attempted, a NAK response is returned. However, if the ForceSTALL bit in the D_EP0ControlIN or D_EP0ControlOUT register for that transaction is set, the transaction is responded with STALL.

**Bits6-1 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 an IN transaction at the endpoint EP0. The descriptor data here refers to the data that starts from the address set in the D_DescAdr_H,L registers and whose size is set in the D_DescSize_H,L registers. Since these set values are updated during execution of the descriptor reply function, they need to be set each time the ReplyDescriptor bit is set.

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

When the transmit operation is complete after the bytes of data set by D_DescSizeH,L have been transmitted, or 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, the DescriptorCmp bit in the D_EP0IntStat register and the IN_TranACK bit in the D_EP0IntStat register both are set to 1.

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

### 7.5.47 0F2h D_EP0ControlIN (Device EP0 Control IN)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0F2h | D_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 the data in the FIFO less than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EP0. When the IN transaction in which a short packet has been transmitted is complete, 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 of zero-length can be transmitted in response to an IN token from the host. If while a packet is being transmitted, data is written to the relevant FIFO after this bit was set, the written data may inadvertently be included in the transmitted packet depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission is complete 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 the NAK response for an IN transaction at the endpoint EP0 irrespective of the number of data bytes in the FIFO.

When the RcvEP0SETUP bit in the USB_DeviceIntStat register is set to 1 upon completion of a setup stage, this bit is set to 1 and while the RcvEP0SETUP bit remains set, cannot be cleared to 0. Furthermore, this bit is set to 1 when an IN transaction in which a short packet was transmitted is complete.

If a transaction has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

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

When the RcvEP0SETUP bit in the USB_DeviceIntStat register is set to 1 upon completion of a setup stage, this bit is cleared to 0 and while the RcvEP0SETUP bit remains set, cannot be set to 1.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.48 0F3h D_EP0ControlOUT (Device EP0 Control OUT)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0F3h | D_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 while this bit is set an OUT transaction at the endpoint EP0 is completed without errors, the ForceNAK bit in this register is set to 1.

**Bits6-5 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 as response for an OUT transaction at the endpoint EP0 irrespective of the amount of free space in the FIFO.

When the RcvEP0SETUP bit in the USB_DeviceIntStat register is set to 1 upon completion of a setup stage, this bit is set to 1 and while the RcvEP0SETUP bit remains set, cannot be cleared to 0.

If a transaction has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion of it. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

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

When the RcvEP0SETUP bit in the USB_DeviceIntStat register is set to 1 upon completion of a setup stage, this bit is cleared to 0 and while the RcvEP0SETUP bit remains set, cannot be set to 1.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.49 0F8h D_EPaMaxSize_H (Device EPa Max Packet Size High)

### 7.5.50 0F9h D_EPaMaxSize_L (Device EPa Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0F8h | D_EPaMaxSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: MaxSize [10] | 0: | 1: | |
| | | | R / W | 1: MaxSize [9] | Endpoint [a] Max Packet Size | | |
| | | | | 0: MaxSize [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 0F9h | D_EPaMaxSize_L | R / W | 7: MaxSize [7] | Endpoint [a] Max Packet Size | 00h |
| | | | | 6: MaxSize [6] | | |
| | | | | 5: MaxSize [5] | | |
| | | | | 4: MaxSize [4] | | |
| | | | | 3: MaxSize [3] | | |
| | | | | 2: MaxSize [2] | | |
| | | | | 1: MaxSize [1] | | |
| | | | | 0: MaxSize [0] | | |

These registers set MaxPacketSize.

#### 0F8h.Bit7-3Reserved

#### 0F8h.Bit2-0, 0F9h.Bit7-0 EPaMaxSize [10:0]

These bits set MaxPacketSize for the endpoint EPa.

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 transfer bytes within the limits given below:

During FS Up to 64 bytes

During HS Up to 512 bytes

To use this endpoint for isochronous transfers, set MaxSise to any desired transfer bytes within the limits given below:

During FS 1 to 1023 bytes

During HS 1 to 1024 bytes

### 7.5.51 0FAh D_EPaConfig (Device EPa Configuration)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0FAh | D_EPaConfig | R / W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | R / W | 6: IntEP_Mode | 0: Normal Toggle (IN)<br>0: Bulk OUT (OUT) | 1: Always Toggle (IN)<br>1: Interrupt OUT (OUT) | |
| | | | R / W | 5: ISO | 0: Not Isochronous | 1: Isochronous | |
| | | | | 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 that the combination of EndpointNumber and INxOUT set for this endpoint does not overlap 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.

Setting of this bit varies 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 about 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 — Set this for a Bulk OUT endpoint.

1: Interrupt OUT — Set this for an Interrupt OUT endpoint.

**Bit5 ISO**

To perform an isochronous transfer, set this bit to 1. For endpoints at which bulk or interrupt transfers are performed, set this bit to 0.

**Bit4** **Reserved**

**Bits3-0** **EndpointNumber**

Set any endpoint number in the range of 0x1 to 0xF.

### 7.5.52 0FCh D_EPaControl (Device EPa Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 0FCh | D_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 operation of the endpoint EPa.

#### Bit7 AutoForceNAK

When transaction at the endpoint EPa is completed without errors, the ForceNAK bit in this register is set to 1.

#### Bit6 EnShortPkt

Setting this bit to 1 allows the data in the FIFO less than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPa. When the IN transaction in which a short packet has been transmitted is complete, 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 zero-length can be transmitted in response to an IN token from the host. If while a packet is being transmitted, data is written to the relevant FIFO after this bit was set, the written data may inadvertently be included in the transmitted packet depending on the timing of the write. Therefore, do not write data to the FIFO until after a packet transmission is completed and this bit is cleared.

#### Bit5 DisAF_NAK_Short

This bit enable/disables the Auto Force NAK Short function (hereonafter referred to as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that completed without errors is a short packet.

The AF_NAK_Short function is by default enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit is 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 as response 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 has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion of it. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

Setting this bit to 1 returns STALL as response for a transaction at the endpoint EPa. This bit has priority over the ForceNAK bit.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.53 100h D_EPbMaxSize_H (Device EPb Max Packet Size High)

### 7.5.54 101h D_EPbMaxSize_L (Device EPb Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 100h | D_EPbMaxSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: MaxSize [10] | 0: | 1: | |
| | | | R / W | 1: MaxSize [9] | Endpoint [b] Max Packet Size | | |
| | | | | 0: MaxSize [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 101h | D_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: MaxSize [2] | | |
| | | | | 1: MaxSize [1] | | |
| | | | | 0: MaxSize [0] | | |

These registers set MaxPacketSize.

**100h.Bit7-3 Reserved**

**100h.Bit2-0, 101h.Bit7-0 EPbMaxSize [10:0]**

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 transfer bytes within the limits given below:

During FS Up to 64 bytes

During HS Up to 512 bytes

To use this endpoint for isochronous transfers, set MaxSise to any desired transfer bytes within the limits given below:

During FS 1 to 1023 bytes

During HS 1 to 1024 bytes

### 7.5.55 102h D_EPbConfig (Devie EPb Configuration)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 102h | D_EPbConfig | R / W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | R / W | 6: IntEP_Mode | 0: Normal Toggle (IN)<br>0: Bulk OUT (OUT) | 1: Always Toggle (IN)<br>1: Interrupt OUT (OUT) | |
| | | | R / W | 5: ISO | 0: Not Isochronous | 1: Isochronous | |
| | | | | 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 that the combination of EndpointNumber and INxOUT set for this endpoint does not overlap 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.

Setting of this bit varies with 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 about 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 — Set this for a Bulk OUT endpoint.

1: Interrupt OUT — Set this for an Interrupt OUT endpoint.

**Bit5 ISO**

To perform an isochronous transfer, set this bit to 1. For endpoints at which bulk or interrupt transfers are performed, set this bit to 0.

**Bit4** **Reserved**

**Bits3-0** **EndpointNumber**

Set any endpoint number in the range of 0x1 to 0xF.

### 7.5.56 104h D_EPbControl (Device EPb Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 104h | D_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 operation of the endpoint EPb.

#### Bit7 AutoForceNAK

When while this bit is set a transaction at the endpoint EPb is completed without errors, the ForceNAK bit in this register is set to 1.

#### Bit6 EnShortPkt

Setting this bit to 1 allows the data in the FIFO less 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 is complete, 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 zero-length packet can be transmitted in response to an IN token from the host. If while a packet is being transmitted, data is written to the relevant FIFO after this bit was set, the written data may inadvertently be included in the transmitted packet depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission is completed and this bit is cleared.

#### Bit5 DisAF_NAK_Short

This bit enable/disables the Auto Force NAK Short function (hereonafter referred to as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that completed without errors is a short packet.

The AF_NAK_Short function is by default enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit is 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 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 EPb 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 as response 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 has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion of it. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

Setting this bit to 1 returns STALL as response for a transaction at the endpoint EPb. This bit has priority over the ForceNAK bit.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.57 108h D_EPcMaxSize_H (Device EPc Max Packet Size High)

### 7.5.58 109h D_EPcMaxSize_L (Device EPc Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 108h | D_EPcMaxSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: MaxSize [10] | 0: | 1: | |
| | | | R / W | 1: MaxSize [9] | Endpoint [c] Max Packet Size | | |
| | | | | 0: MaxSize [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 109h | D_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: MaxSize [2] | | |
| | | | | 1: MaxSize [1] | | |
| | | | | 0: MaxSize [0] | | |

These registers set MaxPacketSize.

#### 108h.Bit7-3 Reserved

#### 108h.Bit2-0, 109h.Bit7-0 EPcMaxSize [10:0]

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 transfer bytes within the limits given below:

During FS Up to 64 bytes

During HS Up to 512 bytes

To use this endpoint for isochronous transfers, set MaxSise to any desired transfer bytes within the limits given below:

During FS 1 to 1023 bytes

During HS 1 to 1024 bytes

### 7.5.59 10Ah D_EPcConfig (Device EPc Configuration)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 10Ah | D_EPcConfig | R / W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | R / W | 6: IntEP_Mode | 0: Normal Toggle (IN)<br>0: Bulk OUT (OUT) | 1: Always Toggle (IN)<br>1: Interrupt OUT (OUT) | |
| | | | R / W | 5: ISO | 0: Not Isochronous | 1: Isochronous | |
| | | | | 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 that the combination of EndpointNumber and INxOUT set for this endpoint does not overlap 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.

Setting of this bit differs with 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 about 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 — Set this for a Bulk OUT endpoint.

1: Interrupt OUT — Set this for an Interrupt OUT endpoint.

**Bit5 ISO**

To perform an isochronous transfer, set this bit to 1. For endpoints at which bulk or interrupt transfers are performed, set this bit to 0.

**Bit4** **Reserved**

**Bits3-0** **EndpointNumber**

Set any endpoint number in the range of 0x1 to 0xF.

### 7.5.60 10Ch D_EPcControl (Device EPc Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 10Ch | D_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 operation of the endpoint EPc.

**Bit7 AutoForceNAK**

When while this bit is set and transaction at the endpoint EPc is completed without errors, the ForceNAK bit in this register is set to 1.

**Bit6 EnShortPkt**

Setting this bit to 1 allows the data in the FIFO less than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPc. When the IN transaction in which a short packet has been transmitted is complete, 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 zero-length packet can be transmitted in response to an IN token from the host. If while a packet is being transmitted, data is written to the relevant FIFO after this bit was set, the written data may inadvertently be included in the transmitted packet depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission is completed and this bit is cleared.

**Bit5 DisAF_NAK_Short**

This bit enable/disables the Auto Force NAK Short function (hereonafter referred to as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that completed without errors is a short packet.

The AF_NAK_Short function is by default enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit is 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 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 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 as response 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 has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion of it. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

Setting this bit to 1 returns STALL as response for a transaction at the endpoint EPc. This bit has priority over the ForceNAK bit.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.61 110h D_EPdMaxSize_H (Device EPd Max Packet Size High)

### 7.5.62 111h D_EPdMaxSize_L (Device EPd Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 110h | D_EPdMaxSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: MaxSize [10] | 0: | 1: | |
| | | | R / W | 1: MaxSize [9] | Endpoint [d] Max Packet Size | | |
| | | | | 0: MaxSize [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 111h | D_EPdMaxSize_L | R / W | 7: MaxSize [7] | Endpoint [d] Max Packet Size | 00h |
| | | | | 6: MaxSize [6] | | |
| | | | | 5: MaxSize [5] | | |
| | | | | 4: MaxSize [4] | | |
| | | | | 3: MaxSize [3] | | |
| | | | | 2: MaxSize [2] | | |
| | | | | 1: MaxSize [1] | | |
| | | | | 0: MaxSize [0] | | |

These registers set MaxPacketSize.

#### 110h.Bit7-3 Reserved

#### 110h.Bit2-0, 111h.Bit7-0 EPdMaxSize [10:0]

These bits set MaxPacketSize for the endpoint EPd.

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 transfer bytes within the limits given below:

During FS Up to 64 bytes

During HS Up to 512 bytes

To use this endpoint for isochronous transfers, set MaxSise to any desired transfer bytes within the limits given below:

During FS 1 to 1023 bytes

During HS 1 to 1024 bytes

### 7.5.63 112h D_EPdConfig (Device EPd Configuration)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 112h | D_EPdConfig | R / W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | R / W | 6: IntEP_Mode | 0: Normal Toggle (IN)<br>0: Bulk OUT (OUT) | 1: Always Toggle (IN)<br>1: Interrupt OUT (OUT) | |
| | | | R / W | 5: ISO | 0: Not Isochronous | 1: Isochronous | |
| | | | | 4: | 0: | 1: | |
| | | | R / W | 3: EndpointNumber [3] | Endpoint Number | | |
| | | | | 2: EndpointNumber [2] | | | |
| | | | | 1: EndpointNumber [1] | | | |
| | | | | 0: EndpointNumber [0] | | | |

This register sets the endpoint Epd.

Make sure that the combination of EndpointNumber and INxOUT set for this endpoint does not overlap 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.

Setting of this bit differs with 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 about 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 — Set this for a Bulk OUT endpoint.

1: Interrupt OUT — Set this for an Interrupt OUT endpoint.

**Bit5 ISO**

To perform an isochronous transfer, set this bit to 1. For endpoints at which bulk or interrupt transfers are performed, set this bit to 0.

**Bit4** **Reserved**

**Bits3-0** **EndpointNumber**

Set any endpoint number in the range of 0x1 to 0xF.

### 7.5.64 114h D_EPdControl (Device EPd Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 114h | D_EPdControl | 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 operation of the endpoint EPd.

#### Bit7 AutoForceNAK

When while this bit is set and transaction at the endpoint EPd is completed without errors, the ForceNAK bit in this register is set to 1.

#### Bit6 EnShortPkt

Setting this bit to 1 allows the data in the FIFO less than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPd. When the IN transaction in which a short packet has been transmitted is complete, 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 zero-length packet can be transmitted in response to an IN token from the host. If while a packet is being transmitted, data is written to the relevant FIFO after this bit was set, the written data may inadvertently be included in the transmitted packet depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission is completed and this bit is cleared.

#### Bit5 DisAF_NAK_Short

This bit enable/disables the Auto Force NAK Short function (hereonafter referred to as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that completed without errors is a short packet.

The AF_NAK_Short function is by default enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit is 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 EPd.

**Bit3 ToggleSet**

This bit sets the toggle sequence bit for the endpoint EPd 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 EPd 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 as response for a transaction at the endpoint EPd irrespective of the number of data bytes or the amount of free space in the FIFO.

If a transaction has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion of it. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

Setting this bit to 1 returns STALL as response for a transaction at the endpoint EPd. This bit has priority over the ForceNAK bit.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.65 118h D_EPeMaxSize_H (Device EPe Max Packet Size High)

### 7.5.66 119h D_EPeMaxSize_L (Device EPe Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 118h | D_EPeMaxSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: MaxSize [10] | 0: | 1: | |
| | | | R / W | 1: MaxSize [9] | Endpoint [e] Max Packet Size | | |
| | | | | 0: MaxSize [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 119h | D_EPeMaxSize_L | R / W | 7: MaxSize [7] | Endpoint [e] Max Packet Size | 00h |
| | | | | 6: MaxSize [6] | | |
| | | | | 5: MaxSize [5] | | |
| | | | | 4: MaxSize [4] | | |
| | | | | 3: MaxSize [3] | | |
| | | | | 2: MaxSize [2] | | |
| | | | | 1: MaxSize [1] | | |
| | | | | 0: MaxSize [0] | | |

These registers set MaxPacketSize.

#### 118h.Bit7-3 Reserved

#### 118h.Bit2-0, 119h.Bit7-0 EPeMaxSize [10:0]

These bits set MaxPacketSize for the endpoint EPe.

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 transfer bytes within the limits given below:

During FS Up to 64 bytes

During HS Up to 512 bytes

To use this endpoint for isochronous transfers, set MaxSise to any desired transfer bytes within the limits given below:

During FS 1 to 1023 bytes

During HS 1 to 1024 bytes

### 7.5.67 11Ah D_EPeConfig (Device EPe Configuration)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 112h | D_EPeConfig | R / W | 7: INxOUT | 0: OUT | 1: IN | 00h |
| | | | R / W | 6: IntEP_Mode | 0: Normal Toggle (IN)<br>0: Bulk OUT (OUT) | 1: Always Toggle (IN)<br>1: Interrupt OUT (OUT) | |
| | | | R / W | 5: ISO | 0: Not Isochronous | 1: Isochronous | |
| | | | | 4: | 0: | 1: | |
| | | | R / W | 3: EndpointNumber [3] | Endpoint Number | | |
| | | | | 2: EndpointNumber [2] | | | |
| | | | | 1: EndpointNumber [1] | | | |
| | | | | 0: EndpointNumber [0] | | | |

This register sets the endpoint Epe.

Make sure that the combination of EndpointNumber and INxOUT set for this endpoint does not overlap 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.

Setting of this bit differs with 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 about 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 — Set this for a Bulk OUT endpoint.

1: Interrupt OUT — Set this for an Interrupt OUT endpoint.

**Bit5 ISO**

To perform an isochronous transfer, set this bit to 1. For endpoints at which bulk or interrupt transfers are performed, set this bit to 0.

**Bit4** **Reserved**

**Bits3-0** **EndpointNumber**

Set any endpoint number in the range of 0x1 to 0xF.

### 7.5.68 11Ch D_EPeControl (Device EPe Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 11Ch | D_EPeControl | 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 operation of the endpoint EPe.

**Bit7 AutoForceNAK**

When while this bit is set and transaction at the endpoint EPe is completed without errors, the ForceNAK bit in this register is set to 1.

**Bit6 EnShortPkt**

Setting this bit to 1 allows the data in the FIFO less than MaxPacketSize to be transmitted as a short packet for an IN transaction at the endpoint EPe. When the IN transaction in which a short packet has been transmitted is complete, 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 zero-length packet can be transmitted in response to an IN token from the host. If while a packet is being transmitted, data is written to the relevant FIFO after this bit was set, the written data may inadvertently be included in the transmitted packet depending on the write timing. Therefore, do not write data to the FIFO until after a packet transmission is completed and this bit is cleared.

**Bit5 DisAF_NAK_Short**

This bit enable/disables the Auto Force NAK Short function (hereonafter referred to as "AF_NAK_Short"*).

* This function automatically sets the ForceNAK bit to 1 if the packet received in an OUT transaction that completed without errors is a short packet.

The AF_NAK_Short function is by default enabled.

Setting this bit to 1 disables the AF_NAK_Short function.

If the AutoForceNAK bit is 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 EPe.

**Bit3 ToggleSet**

This bit sets the toggle sequence bit for the endpoint EPd 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 EPe 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 as response for a transaction at the endpoint EPe irrespective of the number of data bytes or the amount of free space in the FIFO.

If a transaction has already been underway when an attempt was made to set this bit to 1, the bit is not set until the transaction is complete, and is set to 1 upon completion of it. If no transactions are under execution, the bit is set to 1 immediately.

**Bit0 ForceSTALL**

Setting this bit to 1 returns STALL as response for a transaction at the endpoint EPe. This bit has priority over the ForceNAK bit.

If there is any transaction currently executed and this bit is set a certain time after the transaction has started, setting of this bit takes effect beginning with the next transaction.

### 7.5.69 120h D_DescAdrs_H (Device Descriptor Address High)

### 7.5.70 121h D_DescAdrs_L (Device Descriptor Address Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 120h | D_DescAdrs_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: DescAdrs[12] | Descriptor Address | | |
| | | | | 3: DescAdrs [11] | | | |
| | | | | 2: DescAdrs [10] | | | |
| | | | | 1: DescAdrs [9] | | | |
| | | | | 0: DescAdrs [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 121h | D_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.

#### 120h.Bit7-5 Reserved

#### 120h.Bit4-0, 121h.Bit7-0 DescAdrs [12:0]

These bits specify the start address of the FIFO from which a descriptor reply operation of the Descriptor Reply function is to start.

Descriptor Address is not intended to allocate a FIFO area to the Descriptor Reply function. No matter how FIFO areas have been set, any address in the entire FIFO area from 0x000 to 0x11FF (4.5 kbytes) can be specified for Descriptor Address.

During descriptor reply, DescAdrs is updated by an amount equal to the transmitted bytes of data each time an IN transaction at the endpoint EP0 is complete. For details about the Descriptor Reply function, refer to the section on ReplyDescriptor of the D_EP0Control register.

Since the FIFO area used for the Descriptor Reply function is not explicitly allocated, make sure this FIFO does not overlap those of other endpoints by setting up the D_DescAdrs_H,L and the D_DescSize_H,L registers properly. An address range from the reserved end address of the CSW area (0x0030) to the start address of the area to be allocated for AREA0–5 is appropriate.

To inspect Descriptor Address, read D_DescAdrs_H and D_DescAdrs_L in that order.

### 7.5.71 122h D_DescSize_H (Device Descriptor Size High)

### 7.5.72 123h D_DescSize_L (Device Descriptor Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 122h | D_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] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Device | 123h | D_DescSize_L | R / W | 7: DescSize [7] | DescriptorSize | 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.

#### 122h.Bit7-2 Reserved

#### 122h.Bit1-0, 123h.Bit7-0 DescSize [9:0]

For Descriptor Size, specify the total number of data bytes to be sent back to the host in the Descriptor Reply function. For details about the Descriptor Reply function, refer to the section on ReplyDescriptor of the D_EP0Control register.

No matter how FIFO areas have been set, any size from 0x000 to 0x3FF can be specified for Descriptor Size. During descriptor reply, DescSize is updated by an amount equal to the transmitted bytes of data each time an IN transaction at the endpoint EP0 is complete.

Since the FIFO area used for the Descriptor Reply function is not explicitly allocated, make sure this FIFO does not overlap those of other endpoints by setting up the D_DescAdrs_H,L and the D_DescSize_H,L registers properly. An address range from the reserved end address of the CSW area (0x0030) to the start address of the area to be allocated for AREA0–5 is appropriate.

To inspect Descriptor Size, read D_DescSize_H and D_DescSize_L in that order.

### 7.5.73 126h D_EP_DMA_Ctrl (Device EP DMA Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 126h | D_EP_DMA_Ctrl | 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 indicates and sets the FIFO status during a DMA transfer.

**Bit7 FIFO_Running**

This bit indicates that the FIFO for the endpoint connected to DMA is currently operating. This bit is set to 1 when DMA is activated and cleared to 0 when the FIFO is emptied after DMA completion.

**Bit6 AutoEnShort**

If an odd number of data bytes of fewer than max packet size remains in the FIFO for any endpoint at DMA completion, this bit sets the EnShortPkt bit for that endpoint to 1.

This bit is effective even when the endpoint connected to DMA is directed for IN.

**Bits5-0 Reserved**

### 7.5.74 128h D_EnEP_IN_H (Device Enable Endpoint-IN High)

### 7.5.75 129h D_EnEP_IN_L (Device Enable Endpoint-IN Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 128h | D_EnEP_IN_H | R / W | 7: EnEP15IN | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnEP14IN | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP13IN | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP12IN | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP11IN | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP10IN | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP9IN | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnEP8IN | 0: Disable | 1: Enable | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 129h | D_EnEP_IN_L | R / W | 7: EnEP7IN | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnEP6IN | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP5IN | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP4IN | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP3IN | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP2IN | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP1IN | 0: Disable | 1: Enable | |
| | | | | 0: | 0: | 1: | |

These registers set the endpoints to be enabled as an IN endpoint during device mode.

If a transaction is issued (IN token received) from the host to one of the endpoints enabled by D_EnEP_IN_H,L that are appropriately set for the D_EPx{x=0,a-e}-related registers, except for those joined by the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit, the response described below is returned.

The response returned to the host for the IN token depends on how D_EnEP_IN_ISO_H,L is set. For endpoints whose relevant bit is set to 1, the response to the IN token is a zero-length packet. For endpoints whose relevant bit is cleared to 0, the response to the IN token is NAK.

### 7.5.76 12Ah D_EnEP_OUT_H (Device Enable Endpoint-IN High)

### 7.5.77 12Bh D_EnEP_OUT_L (Device Enable Endpoint-IN Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 12Ah | D_EnEP_OUT_H | R / W | 7: EnEP15OUT | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnEP14OUT | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP13OUT | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP12OUT | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP11OUT | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP10OUT | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP9OUT | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnEP8OUT | 0: Disable | 1: Enable | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 12Bh | D_EnEP_OUT_L | R / W | 7: EnEP7OUT | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnEP6OUT | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP5OUT | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP4OUT | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP3OUT | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP2OUT | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP1OUT | 0: Disable | 1: Enable | |
| | | | | 0: | 0: | 1: | |

These registers set the endpoints to be enabled as an OUT endpoint during device mode.

If a transaction is issued (IN token received) from the host to one of the endpoints enabled by D_EnEP_OUT_H,L that are appropriately set for the D_EPx{x=0,a-e}-related registers, except for those joined by the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit, the response described below is returned. The corresponding bit in these registers is set to 1.

The response returned to the host for the OUT token depends on how D_EnEP_OUT_ISO_H,L is set. For endpoints whose relevant bit is set to 1, the data sent from the host is not received, and no response is issued for the handshake sequence. For endpoints whose relevant bit is cleared to 0, an NAK response is issued for the OUT token. If a PING token is issued from the host while the device is set to HS, an NAK response is issued for the token.

If any bit in these registers corresponding to an endpoint is set, set the D_EPx{x=0,a-e}-related registers appropriately. Join the endpoint to the FIFO area using the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit to make a transaction executable.

### 7.5.78 12Ch D_EnEP_IN_H (Device Enable Endpoint-IN High)

### 7.5.79 12Dh D_EnEP_IN_L (Device Enable Endpoint-IN Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 12Ch | D_EnEP_IN | R / W | 7: EnEP15IN_ISO | 0: Disable | 1: Enable | 00h |
| | | _ISO_H | R / W | 6: EnEP14IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP13IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP12IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP11IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP10IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP9IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnEP8IN_ISO | 0: Disable | 1: Enable | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 12Dh | D_EnEP_IN | R / W | 7: EnEP7IN_ISO | 0: Disable | 1: Enable | 00h |
| | | _ISO_H | R / W | 6: EnEP6IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP5IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP4IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP3IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP2IN_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP1IN_ISO | 0: Disable | 1: Enable | |
| | | | | 0: | 0: | 1: | |

These registers set the endpoints to be enabled as an IN endpoint during device mode to ISO mode.

If a transaction is issued (IN token received) from the host to one of the endpoints enabled by D_EnEP_IN_H,L that are appropriately set for the D_EPx{x=0,a-e}-related registers, except for those joined by the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit, the response described below is returned.

The response returned to the host for the IN token depends on how D_EnEP_IN_ISO_H,L is set. For endpoints whose relevant bit is set to 1, the response to the IN token is a zero-length packet. For endpoints whose relevant bit is cleared to 0, the response to the IN token is NAK.

### 7.5.80 12Eh D_EnEP_OUT_ISO_H (Device Enable Endpoint-OUT Isochronous High)

### 7.5.81 12Fh D_EnEP_OUT_ISO_L (Device Enable Endpoint-OUT Isochronous Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 12Eh | D_EnEP_OUT | R / W | 7: EnEP15OUT_ISO | 0: Disable | 1: Enable | 00h |
| | | _ISO_H | R / W | 6: EnEP14OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP13OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP12OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP11OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP10OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP9OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnEP8OUT_ISO | 0: Disable | 1: Enable | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 12Fh | D_EnEP_OUT | R / W | 7: EnEP7OUT_ISO | 0: Disable | 1: Enable | 00h |
| | | _ISO_L | R / W | 6: EnEP6OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnEP5OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnEP4OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnEP3OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnEP2OUT_ISO | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnEP1OUT_ISO | 0: Disable | 1: Enable | |
| | | | | 0: | 0: | 1: | |

These registers set the endpoints to be enabled as an OUT endpoint during device mode.

If a transaction is issued (IN token received) from the host to one of the endpoints enabled by D_EnEP_OUT_H,L that are appropriately set for the D_EPx{x=0,a-e}-related registers, except for those joined by the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit, the response described below is returned. The corresponding bit in these registers is set to 1.

The response returned to the host for the OUT token depends on how D_EnEP_OUT_ISO_H,L is set. For endpoints whose relevant bit is set to 1, the data sent from the host is not received, and no response is issued for the handshake sequence. For endpoints whose relevant bit is cleared to 0, an NAK response is issued for the OUT token. If a PING token is issued from the host while the device is set to HS, an NAK response is issued for the token.

If any bit in these registers corresponding to an endpoint is set, set the D_EPx{x=0,a-e}-related registers appropriately. Join the endpoint to the FIFO area using the AREAn{n=0-5}Join.JoinEPxCHx{x=0,a-e} bit to make a transaction executable.

## 7.6 Detailed Description of Host Registers

### 7.6.1 140h H_SIE_IntStat_0(Host SIE Interrupt Status 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 140h | H_SIE_IntStat_0 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R (W) | 4: DetectCon | 0: None | 1: Detect Connect | |
| | | | R (W) | 3: DetectDiscon | 0: None | 1: Detect Disconnect | |
| | | | R (W) | 2: DetectRmtWkup | 0: None | 1: Detect Remote WakeUp | |
| | | | R (W) | 1: DetectDevChirpOK | 0: None | 1: Detect Device Chirp OK | |
| | | | R (W) | 0: DetectDevChirpNG | 0: None | 1: Detect Device Chirp NG | |

This register shows the SIE-related interrupts of the host.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bits7-5 Reserved**

**Bit4 DetectCon**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a connection of USB cable is detected.

**Bit3 DetectDiscon**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a disconnection of USB cable is detected.

This detective function is not available when H_NegoControl_1.PortSpeed = = “HS” and H_NegoControl_0.HostState = = “SUSPEND” are set.

**Bit2 DetectRmtWkup**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a Remote Wakeup signal from the USB device is detected during a Suspend state.

**Bit1 DetectDevChirpOK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when the chirp signal from the USB device is normal.

**Bit0 DetectDevChirpNG**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when the chirp signal from the USB device is erratic.

Even when power management is ACTIVE, the synchronous bits (bits 4 to 0) cannot be read out or written to (to clear the cause of interrupt) unless the HostDeviceSel.HOSTxDEVICE bit = 1 (i.e., host mode). Therefore, if a transition from this state is required, the processing described below should be executed from firmware to ensure that the interrupt signal XINT will not be asserted inadvertently by these interrupt statuses.

<For transition from host mode while ACTIVE>

1) Process and clear the interrupt status (H_SIE_IntStat_0.Bit4 through 0).

2) Disable the interrupt status (H_SIE_IntEnb_0.Bit4 through 0).

<For transition to host mode while ACTIVE>

3) Clear the interrupt status (H_SIE_IntStat_0.Bit4 through 0).

4) Reenable the interrupt status (H_SIE_IntEnb_0.Bit4 through 0).

### 7.6.2 141h H_SIE_IntStat_1(SIE Host Interrupt Status 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 141h | H_SIE_IntStat_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1 | |
| | | | | 4: | 0: | 1: | |
| | | | R (W) | 3: Disabled Cmp | 0: None | 1: Disabled Complete | |
| | | | R (W) | 2: ResumeCmp | 0: None | 1: Resume Complete | |
| | | | R (W) | 1: SuspendCmp | 0: None | 1: Suspend Cmplete | |
| | | | R (W) | 0: ResetCmp | 0: None | 1: Reset Cmplete | |

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

**Bits7-4 Reserved**

**Bit 3 DisabledCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 if a transition to DISABLED state is completed without errors when the state management function is executed after setting GoDISABLED in H_NegoControl_0.AutoMode[3:0].

**Bit2 ResumeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 if a Resume is completed without errors when the state management function is executed after setting GoRESUME in H_NegoControl_0.AutoMode[3:0].

**Bit1 SuspendCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 if a transition to Suspend is completed without errors when the state management function is executed after setting GoSUSPEND in H_NegoControl_0.AutoMode[3:0].

**Bit0 ResetCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 if a USB reset is completed without errors when the state management function is executed after setting GoRESET in H_NegoControl_0.AutoMode[3:0].

Even when power management is ACTIVE, the synchronous bits (bits 3–0) cannot be accessed for reads or writes (to clear the cause of an interrupt) unless the HostDeviceSel.HOSTxDEVICE bit = 1 (i.e., HOST mode).

To change states from this mode, be sure to perform the following processing in firmware to ensure that the interrupt signal XINT will not be asserted by one of the interrupt statuses.

When shifting out from the ACT_HOST state:

1) Process the interrupt status and then clear (H_SIE_IntStat_1.Bit3 through 0).

2) Disable the interrupt status (H_SIE_IntEnb_1.Bit3 through 0).

When shifting in to the ACT_HOST state:

3) Clear the interrupt status (H_SIE_IntStat_1.Bit3 through 0).

4) Enable the interrupt status (H_SIE_IntEnb_1.Bit3 through 0)

### 7.6.3 143h H_FrameIntStat(Host Frame Interrupt Status )

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 143h | H_FrameIntStat | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R (W) | 2: PortErr | 0: None | 1: Port Error | |
| | | | R (W) | 1: FrameNumOver | 0: None | 1: Frame Number Over | |
| | | | R (W) | 0: SOF | 0: None | 1: SOF | |

This register shows the frame-related interrupts of the host.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bits7-3 Reserved**

**Bit2 PortErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a port is detected during USB host operation.

**Bit1 FrameNumOver**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when the frame number counter is overflowed (FrameNumber_H register's MSB (bit 2) changed state from 1 to 0). If the FrameNumber_H,L registers are insufficient for the necessary digits of counts, the deficiency can be compensated for by counting occurrences of this interrupt.

**Bit0 SOF**

This bit indicates the cause of interrupt directly.

This bit is set to 1 in the following cases depending on the transfer speed:

HS: When the host controller transmitted an SOF token in microframe 0

FS: When the host controller transmitted an SOF token

LS: When the host controller transmitted Keepalive

### 7.6.4 144h H_CHrIntStat (Host CHr Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 144h | H_CHrIntStat | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R | 4: H_CHeIntStat | 0: None | 1: CHe Interrupt | |
| | | | R | 3: H_CHdIntStat | 0: None | 1: CHd Interrupt | |
| | | | R | 2: H_CHcIntStat | 0: None | 1: CHc Interrupt | |
| | | | R | 1: H_CHbIntStat | 0: None | 1: CHb Interrupt | |
| | | | R | 0: H_CHaIntStat | 0: None | 1: CHa Interrupt | |

This register shows the interrupts of channel CHr.

**Bits7-5 Reserved**

**Bit4 H_CHeIntStat**

This bit indicates the cause of interrupt indirectly.

This bit is set to 1 when any bit in the H_CHeIntEnb register corresponding to an interrupt cause in the H_CHeIntStat register is enabled.

**Bit3 H_CHdIntStat**

This bit indicates the cause of interrupt indirectly.

This bit is set to 1 when any bit in the H_CHdIntEnb register corresponding to an interrupt cause in the H_CHdIntStat register is enabled.

**Bit2 H_CHcIntStat**

This bit indicates the cause of interrupt indirectly.

This bit is set to 1 when any bit in the H_CHcIntEnb register corresponding to an interrupt cause in the H_CHcIntStat register is enabled.

**Bit1 H_CHbIntStat**

This bit indicates the cause of interrupt indirectly.

This bit is set to 1 when any bit in the H_CHbIntEnb register corresponding to an interrupt cause in the H_CHbIntStat register is enabled.

**Bit0 H_CHaIntStat**

This bit indicates the cause of interrupt indirectly.

This bit is set to 1 when any bit in the H_CHaIntEnb register corresponding to an interrupt cause in the H_CHaIntStat register is enabled.

### 7.6.5 145h H_CH0IntStat (Host CH0 Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 145h | H_CH0IntStat | R (W) | 7: TotalSizeCmp | 0: None | 1: TotalSize Complete | 00h |
| | | | R (W) | 6: TranACK | 0: None | 1: Tran ACK | |
| | | | R (W) | 5: TranErr | 0: None | 1: Tran Error | |
| | | | R (W) | 4: ChangeCondition | 0: None | 1: Change Condition | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R (W) | 1: CTL_SupportCmp | 0: None | 1: CTL_Support Cmplete | |
| | | | R (W) | 0: CTL_SupportStop | 0: None | 1: CTL_Support Stop | |

This register shows the interrupt statuses of channel CH0.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 TotalSizeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a transfer in a packet transfer in IRP units is completed without errors.

When the control transfer support function is active, this bit is set to 1 when the setup stage, data stage, and status stage each has finished normallyis completed without errors.

**Bit6 TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when transfers for as many individual transactions as set in the H_CH0Config_0.ACK_Cnt bits are completed without errors.

When the control transfer support function is active, this bit is set to 1 at only the data stage.

**Bit5 TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an individual transaction has terminated in an error such as a retry error (i.e, time-out error), CRC error, bit stuffing error, PID error (including unexpected PID), or toggle mismatch error.

**Bit4 ChangeCondition**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a condition code STALL, Data Overrun or Data Underrun or three consecutive retry errors occurred in a transaction.

This bit is also set to 1 when the H_CH0Config_0.TranGo bit was cleared by the firmware. In that case, ConditionCode indicates the result of the last transaction.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-2 Reserved**

**Bit1** **CTL_SupportCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when all stages of a control transfer initiated by the control transfer function are completed without errors.

Furthermore, this bit is set to 1 when in a process to deactivate the control transfer support function by clearing the CTL_SupportGo bit in the H_CTL_SupportControl register, the status stage has finished normallyis completed without errors, resulting in a termination of the deactivation process.

**Bit0** **CTL_SupportStop**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a control transfer initiated by the control transfer function is abnormally terminated during the process.

Furthermore, this bit is set to 1 when in a process to deactivate the control transfer support function by clearing the CTL_SupportGo bit in the H_CTL_SupportControl register, the deactivation process has terminated at other than the status stage or the transaction has terminated in error at the status stage.

### 7.6.6 146h H_CHaIntStat (Host CHa Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 146h | H_CHaIntStat | R (W) | 7: TotalSizeCmp | 0: None | 1: Tran Complete | 00h |
| | | | R (W) | 6: TranACK | 0: None | 1: Tran ACK | |
| | | | R (W) | 5: TranErr | 0: None | 1: Tran Error | |
| | | | R (W) | 4: ChangeCondition | 0: None | 1: Change Condition | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R (W) | 1: BO_SupportCmp | 0: None | 1: BO Support Cmplete | |
| | | | R (W) | 0: BO_SupportStop | 0: None | 1: BO Support Stop | |

This register shows the interrupt statuses of channel CHa.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 TotalSizeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a transfer in a packet transfer in IRP units is completed without errors.

When the bulk-only support function is active, this bit is set to 1 when a CBW transport, data transport, or CSW transport has finished normallyis completed without errors.

**Bit6 TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when transfers for as many individual transactions as set in the H_CH0Config_0.ACK_Cnt bits are completed without errors.

When the bulk-only support function is active, this bit is set to 1, only in a data transport.

**Bit5 TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an individual transaction has terminated in an error such as a retry error (i.e, time-out error), CRC error, bit stuffing error, PID error (including unexpected PID), or toggle mismatch error.

**Bit4 ChangeCondition**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a condition code STALL, Data Overrun or Data Underrun or three consecutive retry errors occurred in a transaction.

This bit is also set to 1 when the H_ChaConfig_0.TranGo bit is cleared by the firmware. In that case, ConditionCode indicates the result of the last transaction.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-2 Reserved**

**Bit1** **BO_SupportCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a status transport in transfers initiated by the bulk-only support function is completed without errors.

Furthermore, this bit is set to 1 when in a process to deactivate the bulk-only support function by clearing the BO_SupportGo bit in the H_BO_SupportControl register, a CSW transport has finished normallyis completed without errors, resulting in a termination of the deactivation process.

**Bit0** **BO_SupportStop**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when any transfer in transfers initiated by the bulk-only support function is abnormally terminated.

Furthermore, this bit is set to 1 when in a process to deactivate the bulk-only support function by clearing the BO_SupportGo bit in the H_BO_SupportControl register, the deactivation process has terminated in other than a CSW transport or an error was detected in a CSW transport.

### 7.6.7 147h H_CHbIntStat (Host CHb Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 147h | H_ChbIntStat | R (W) | 7: TotalSizeCmp | 0: None | 1: Tran Complete | 00h |
| | | | R (W) | 6: TranACK | 0: None | 1: Tran ACK | |
| | | | R (W) | 5: TranErr | 0: None | 1: Tran Error | |
| | | | R (W) | 4: ChangeCondition | 0: None | 1: Change Condition | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register shows the interrupt statuses of channel CHb.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 TotalSizeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a transfer in a packet transfer in IRP units is completed without errors.

**Bit6 TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when transfers for as many individual transactions as set in the H_CH0Config_0.ACK_Cnt bits are completed without errors.

**Bit5 TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an individual transaction has terminated in an error such as a retry error (i.e, time-out error), CRC error, bit stuffing error, PID error (including unexpected PID), or toggle mismatch error.

**Bit4 ChangeCondition**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a condition code STALL, Data Overrun or Data Underrun or three consecutive retry errors occurred in a transaction.

This bit is also set to 1 when the H_ChbConfig_0.TranGo bit is cleared by the firmware. In that case, ConditionCode indicates the result of the last transaction.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.8 148h H_CHcIntStat (Host CHc Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 148h | H_CHcIntStat | R (W) | 7: TotalSizeCmp | 0: None | 1: Tran Complete | 00h |
| | | | R (W) | 6: TranACK | 0: None | 1: Tran ACK | |
| | | | R (W) | 5: TranErr | 0: None | 1: Tran Error | |
| | | | R (W) | 4: ChangeCondition | 0: None | 1: Change Condition | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register shows the interrupt statuses of channel CHc.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 TotalSizeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a transfer in a packet transfer in IRP units is completed without errors.

**Bit6 TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when transfers for as many individual transactions as set in the H_CH0Config_0.ACK_Cnt bits are completed without errors.

**Bit5 TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an individual transaction has terminated in an error such as a retry error (i.e, time-out error), CRC error, bit stuffing error, PID error (including unexpected PID), or toggle mismatch error.

**Bit4 ChangeCondition**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a condition code STALL, Data Overrun or Data Underrun or three consecutive retry errors occurred in a transaction.

This bit is also set to 1 when the H_ChcConfig_0.TranGo bit is cleared by the firmware. In that case, ConditionCode indicates the result of the last transaction.

| Code | Meaning | Description |
|---|---|---|
| 000 | NoError | The transaction is completed without error. |
| 001 | Stall | The endpoint has returned stall PID. |
| 010 | DataOverrun | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DataUnderrun | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |
| 100 | RetryError | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.9 149h H_CHdIntStat (Host CHd Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 149h | H_CHdIntStat | R (W) | 7: TotalSizeCmp | 0: None | 1: TotalSize Complete | 00h |
| | | | R (W) | 6: TranACK | 0: None | 1: Tran ACK | |
| | | | R (W) | 5: TranErr | 0: None | 1: Tran Error | |
| | | | R (W) | 4: ChangeCondition | 0: None | 1: Change Condition | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

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

**Bit7 TotalSizeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a transfer in a packet transfer in IRP units is completed without errors.

**Bit6 TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when transfers for as many individual transactions as set in the H_CH0Config_0.ACK_Cnt bits are completed without errors.

**Bit5 TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an individual transaction has terminated in an error such as a retry error (i.e, time-out error), CRC error, bit stuffing error, PID error (including unexpected PID), or toggle mismatch error.

**Bit4 ChangeCondition**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a condition code STALL, Data Overrun or Data Underrun or three consecutive retry errors occurred in a transaction.

This bit is also set to 1 when the H_ChdConfig_0.TranGo bit is cleared by the firmware. In that case, ConditionCode indicates the result of the last transaction.

| Code | Meaning | Description |
|---|---|---|
| 000 | NoError | The transaction is completed without error. |
| 001 | Stall | The endpoint has returned stall PID. |
| 010 | DataOverrun | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DataUnderrun | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |
| 100 | RetryError | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.10 14Ah H_CHeIntStat (Host CHe Interrupt Status)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 14Ah | H_CHeIntStat | R (W) | 7: TotalSizeCmp | 0: None | 1: TotalSize Complete | 00h |
| | | | R (W) | 6: TranACK | 0: None | 1: Tran ACK | |
| | | | R (W) | 5: TranErr | 0: None | 1: Tran Error | |
| | | | R (W) | 4: ChangeCondition | 0: None | 1: Change Condition | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register shows the interrupt statuses of channel CHe.

The cause of interrupt indicated by any bit in this register can be cleared by writing 1 to that bit.

**Bit7 TotalSizeCmp**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a transfer in a packet transfer in IRP units is completed without errors.

**Bit6 TranACK**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when transfers for as many individual transactions as set in the H_CH0Config_0.ACK_Cnt bits are completed without errors.

**Bit5 TranErr**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when an individual transaction has terminated in an error such as a retry error (i.e, time-out error), CRC error, bit stuffing error, PID error (including unexpected PID), or toggle mismatch error.

**Bit4 ChangeCondition**

This bit indicates the cause of interrupt directly.

This bit is set to 1 when a condition code STALL, Data Overrun or Data Underrun or three consecutive retry errors occurred in a transaction.

This bit is also set to 1 when the H_CheConfig_0.TranGo bit is cleared by the firmware. In that case, ConditionCode indicates the result of the last transaction.

| Code | Meaning | Description |
|---|---|---|
| 000 | NoError | The transaction is completed without error. |
| 001 | Stall | The endpoint has returned stall PID. |
| 010 | DataOverrun | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DataUnderrun | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |
| 100 | RetryError | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.11 150h H_SIE_IntEnb_0 (Host SIE Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 150h | H_SIE_IntEnb_0 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: EnDetectCon | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnDetectDiscon | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnDetectRmtWkup | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnDetectDevChirpOK | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnDetectDevChirpNG | 0: Disable | 1: Enable | |

This register is used to enable or disable the assertion of the H_SIE_IntStat_0 bit in the MainIntStat register for the interrupt causes accommodated in the H_SIE_IntStat_0 register.

### 7.6.12 151h H_SIE_IntEnb_1(SIE Host Interrupt Enable 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 151h | H_SIE_IntEnb_1 | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1 | |
| | | | | 4: | 0: | 1: F | |
| | | | R / W | 3: EnDisabledCmp | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnResumeCmp | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnSuspendCmp | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnResetCmp | 0: Disable | 1: Enable | |

This register is used to enable or disable the assertion of the H_SIE_IntStat_1 bit in the MainIntStat register for the interrupt causes accommodated in the H_SIE_IntStat_1 register.

### 7.6.13 152h Reserved

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 152h | Reserved | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

### 7.6.14 153h H_FrameIntEnb(Host Frame Interrupt Enable )

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 153h | H_FrameIntEnb | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: EnPortErr | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnFrameNumOver | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnSOF | 0: Disable | 1: Enable | |

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

### 7.6.15 154h H_CHrIntEnb(Host CHr Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 154h | H_CHrIntEnb | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: EnH_EnCHeIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 3: EnH_EnCHdIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 2: EnH_EnCHcIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 1: EnH_EnCHbIntStat | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnH_EnCHaIntStat | 0: Disable | 1: Enable | |

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

### 7.6.16 155h H_CH0IntEnb(Host CH0 Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 155h | H_CH0IntEnb | R / W | 7: EnTotalSizeCmp | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnTranACK | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnTranErr | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnChangeCondition | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: EnCTL_SupportCmp | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnCTL_SupportStop | 0: Disable | 1: Enable | |

This register is used to enable or disable the assertion of the H_CH0IntStat bit in the MainIntStat register for the interrupt causes accommodated in the H_CH0IntStat register.

### 7.6.17 156h H_CHaIntEnb (Host CHa Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 156h | H_CHaIntEnb | R / W | 7: EnTotalSizeCmp | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnTranACK | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnTranErr | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnChangeCondition | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: Disable | 1: | |
| | | | R / W | 1: EnBO_Support_Cmp | 0: Disable | 1: Enable | |
| | | | R / W | 0: EnBO_Support_Stop | 0: Disable | 1: Enable | |

This register is used to enable or disable the assertion of the CHaIntStat bit in the H_CHrIntStat register for the interrupt causes accommodated in the H_CHaIntStat register.

### 7.6.18 157h H_CHbIntEnb (Host CHb Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 157h | H_CHbIntEnb | R / W | 7: EnTotalSizeCmp | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnTranACK | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnTranErr | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnChangeCondition | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register is used to enable or disable the assertion of the CHbIntStat bit in the H_CHrIntStat register for the interrupt causes accommodated in the H_CHbIntStat register.

### 7.6.19 158h H_CHcIntEnb (Host CHc Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 158h | H_CHcIntEnb | R / W | 7: EnTotalSizeCmp | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnTranACK | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnTranErr | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnChangeCondition | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register is used to enable or disable the assertion of the CHcIntStat bit in the H_CHrIntStat register for the interrupt causes accommodated in the H_CHcIntStat register.

### 7.6.20 159h H_CHdIntEnb (Host CHd Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 159h | H_CHdIntEnb | R / W | 7: EnTotalSizeCmp | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnTranACK | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnTranErr | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnChangeCondition | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register is used to enable or disable the assertion of the CHdIntStat bit in the H_CHrIntStat register for the interrupt causes accommodated in the H_CHdIntStat register.

### 7.6.21 15Ah H_CHeIntEnb (Host CHe Interrupt Enable)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 15Ah | H_CHeIntEnb | R / W | 7: EnTotalSizeCmp | 0: Disable | 1: Enable | 00h |
| | | | R / W | 6: EnTranACK | 0: Disable | 1: Enable | |
| | | | R / W | 5: EnTranErr | 0: Disable | 1: Enable | |
| | | | R / W | 4: EnChangeCondition | 0: Disable | 1: Enable | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register is used to enable or disable the assertion of the CHeIntStat bit in the H_CHrIntStat register for the interrupt causes accommodated in the H_CHeIntStat register.

### 7.6.22 160h H_NegoControl_0 (Host NegoControl 0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 160h | H_NegoControl_0 | R / W | 7: AutoModeCancel | 0: None | 1: Cancel | 1Xh |
| | | | R | 6: HostState[2] | HostState[2:0] | | |
| | | | R | 5: HostState[1] | | | |
| | | | R | 4: HostState[0] | | | |
| | | | W | 3: AutoMode[3] | AutoMode[3:0] | | |
| | | | W | 2: AutoMode[2] | | | |
| | | | W | 1: AutoMode[1] | | | |
| | | | W | 0: AutoMode[0] | | | |

This register sets the operations associated with host negotiation.

**Bit7 AutoModeCancel**

Setting this bit to 1 halts execution of the host state management support function, maintaining the LSI in that state (in which the settings of H_NegoControl_0.AutoMode and H_XcvrControl are retained, signal line state is retained, internal timer is deactivated, and the connect/disconnect, device Chirp and remote wakeup detection functions each are turned off).

Before the following operations can be performed, halt execution of the host state management support function by setting this bit.

- To change the host state to the IDLE state
- To change the host state to the DISABLED state without awaiting a reset complete status (H_SIE_IntStat_1.ResetCmp) to be issued after detecting an error in Chirp from the device
- To execute test mode by setting H_USB_Test.EnHS_Test

Setting this bit to 1 causes execution of the host state management support function to stop, and this bit is cleared to 0 upon completion of the stopping process (approximately 6 clock cycles required when operating at 60 MHz). In the above case, be sure to confirm that this bit has beenis cleared to 0 before setting GoIDLE or GoDISABLE in H_NegoControl_0.AutoMode or setting H_USB_Test.EnHS_Test.

**Bits6-4 HostState[2:0]**

These bits indicate the current host state while the host state management support function is under execution. The host state is one of the following:

000: Reserved

001: IDLE

010: WAIT_CONNECT

011: DISABLED

100: USB_RESET

101: USB_OPERATIONAL

110: USB_SUSPEND

111: USB_RESUME

**Bits3-0 AutoMode[3:0]**

These bits set a host state to which the LSI is to be placed in by execution of the host state management support function.

These write-only register bits can be used to set one of the following host states:

0001: GoIDLE (causes a transition to IDLE state)

0010: GoWAIT_CONNECT (causes a transition to WAIT_CONNECT state)

0011: GoDISABLED (causes a transition to DISABLED state)

0100: GoRESET (causes a transition to RESET state)

0101: GoOPERATIONAL (causes a transition to OPERATIONAL state)

0110: GoSUSPEND (causes a transition to SUSPEND state)

0111: GoRESUME (causes a transition to RESUME state)

1001: GoWAIT_CONNECTtoDIS (causes successive transitions from WAIT_CONNECT state to DISABLED state)

1010: GoWAIT_CONNECTtoOP (causes successive transitions from WAIT_CONNECT state to OPERATIONAL state)

1100: GoRESETtoOP (causes successive transitions from RESET state to OPERATIONAL state)

1110: GoSUSPENDtoOP (causes successive transitions from SUSPEND state to OPERATIONAL state)

1111: GoRESUMEtoOP (causes successive transitions from RESUME state to OPERATIONAL state)

Other than the above: Reserved

To place the LSI from a given state into the IDLE state (by executing GoIDLE), execute the procedure described below:

- Write 0x80 to the H_NegoControl_0 register
- Check to see that the H_NegoControl_0.AutoModeCancel bit is cleared to 0
- Write 0x01 to the H_NegoControl_0 register

### 7.6.23 162h H_NegoControl_1 (Host NegoControl 1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 162h | H_NegoControl_1 | | 7: | 0: | 1: | 10h |
| | | | | 6: | 0: | 1: | |
| | | | R / W | 5: PortSpeed[1] | PortSpeed[1:0] | | |
| | | | R / W | 4: PortSpeed[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: DisChirpFinish | 0: Normal | 1: DisableChirpFinish | |
| | | | R / W | 0: RmtWkupDetEnb | 0: Disable | 1: Enable | |

This register sets the operations associated with host negotiation.

Note: The Reset value of this register can only be read out when the power management state is ACTIVE. In other states, the Reset value will always read 00h.

**Bits7-6 Reserved**

**Bits5-4 PortSpeed[1:0]**

These bits set and indicate the transfer speed.

00: High Speed

01: Full Speed

10: Reserved

11: Low Speed

**Bits3-2 Reserved**

**Bit1 DisChirpFinish**

This bit sets an operation mode to be assumed when a device Chirp is not completed in a designated time.

0: After flagging a device Chirp error status, the LSI sends out USB Reset for a designated duration to complete USB Reset.

1: After flagging a device Chirp error status, the LSI keeps waiting for the device Chirp to complete, and upon completion of it, finishes USB Reset after executing a host Chirp.

**Bit0 RmtWkupDetEnb**

This bit enable/disables the remote wakeup detection function.

### 7.6.24 164h H_USB_Test (Host USB_Test)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 164h | H_USB_Test | R / W | 7: EnHS_Test | 0: Do nothing | 1: EnHS_Test | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: Test_Force_Enable | 0: Do nothing | 1: Test_Force_Enable | |
| | | | 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 sets the operations relating to USB2.0 test mode during USB host operation.

Test mode can be executed in any of WAIT_CONNECT, DISABLED, or SUSPEND states.

Before the LSI can be shifted from these states into test mode, processing in either state must be terminated. To shift to test mode, execute the procedure described below.

- Set the TranGo bit for all channels (H_CHx{x=0,a-e}Config_0.TranGo), H_CTL_SupportControl.CTL_SupportGo, and H_BO_SupportControl.BOSupportGo all to 0.
- Write 0x80 to the H_NegoControl_0 register.
- Check to see that the H_NegoControl_0.AutoModeCancel bit is cleared to 0.
- Set any of the 5 low-order bits in this register. At the same time, set EnHS_Test to 1.

Furthermore, to switch from one test mode to another or terminate a test mode, write 0x00 to this register. Test mode will be terminated, and the host state will shift to IDLE.

**Bit7 EnHS_Test**

If this bit and any of the 5 low-order bits in the H_USB_Test register are set to 1 simultaneously, the LSI enters the test mode corresponding to that bit, the LSI shifts to the test mode corresponding to that bit.

**Bits6-5 Reserved**

**Bit4 TestForceEnable**

The LSI can shift to the TestForceEnable test mode by setting this bit and the EnHS_Test bit to 1 simultaneously. In this test mode, the host port can be disconnected by sending out an SOF in HS mode.

**Bit3 Test_SE0_NAK**

The LSI shifts to the Test_SE0_NAK test mode by setting this bit to 1 and then the EnHS_Test bit to 1. In this test mode, the host port becomes ready to receive in HS mode.

**Bit2** **TEST_J**

The LSI shifts to the Test_J test mode by setting this bit to 1 and then the EnHS_Test bit to 1. In this test mode, the host port sends out “J” in HS mode.

**Bit1** **TEST_K**

The LSI shifts to the Test_K test mode by setting this bit to 1 and then the EnHS_Test bit to 1. In this test mode, the host port sends out “K” in HS mode.

**Bit0** **Test_Packet**

The LSI shifts to the Test_Packet test mode by setting this bit to 1 and then the EnHS_Test bit to 1. This test mode can only be used on channel CH0. Before entering this test mode, set the FIFO area for CH0 to 64 bytes, clear the FIFO, and write the following data for a test packet to the FIFO. There are no other settings required for CH0.

The following 53 bytes are the data to be written to the FIFO during 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

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

### 7.6.25 170h H_CH0SETUP_0 (Host CH0 SETUP 0)
### 7.6.26 171h H_CH0SETUP_1 (Host CH0 SETUP 1)
### 7.6.27 172h H_CH0SETUP_2 (Host CH0 SETUP 2)
### 7.6.28 173h H_CH0SETUP_3 (Host CH0 SETUP 3)
### 7.6.29 174h H_CH0SETUP_4 (Host CH0 SETUP 4)
### 7.6.30 175h H_CH0SETUP_5 (Host CH0 SETUP 5)
### 7.6.31 176h H_CH0SETUP_6 (Host CH0 SETUP 6)
### 7.6.32 177h H_CH0SETUP_7 (Host CH0 SETUP 7)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 170h<br>-177h | H_CH0SETUP_0<br>-H_CH0SETUP_7 | R/W | 7: CH0SETUP_n [7] | Channel 0 SETUP Data 0<br>- Channel 0 SETUP Data 7 | 00h |
| | | | | 6: CH0SETUP_n [6] | | |
| | | | | 5: CH0SETUP _n [5] | | |
| | | | | 4: CH0SETUP _n [4] | | |
| | | | | 3: CH0SETUP _n [3] | | |
| | | | | 2: CH0SETUP _n [2] | | |
| | | | | 1: CH0SETUP _n [1] | | |
| | | | | 0: CH0SETUP _n [0] | | |

The 8 bytes of data received in the setup stage of channel CH0 are stored in these registers sequentially beginning with CH0SETUP_0.

**CH0SETUP_0**

BmRequestType is set in this register.

**CH0SETUP_1**

BRequest is set in this register.

**CH0SETUP_2**

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

**CH0SETUP_3**

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

**CH0SETUP_4**

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

**CH0SETUP_5**

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

**CH0SETUP_6**

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

**CH0SETUP_7**

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

### 7.6.33 17Eh H_FrameNumber_H (Host FrameNumber High)

### 7.6.34 17Fh H_FrameNumber_L (Host FrameNumber Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 17Eh | H_FrameNumber_H | | 7: | 0: | 1: | 80h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R/W | 2: FrameNumber [10] | Frame Number High | | |
| | | | | 1: FrameNumber [9] | | | |
| | | | | 0: FrameNumber [8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 17Fh | H_FrameNumber_L | R/W | 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 transmitted. To get a frame number, the FrameNumber_H and FrameNumber_L registers must be accessed in pairs. At that time, be sure to access the FrameNumber_H register first.

Note: The Reset value of this register can only be read out when the power management state is ACTIVE. In other states, the Reset value will always read 0000h.

**17Eh.Bit7-3 Reserved**

**17Eh.Bit2-0, 17Fh.Bit7-0 FrameNumber [10:0]**

These bits show the FrameNumber of the transmitted SOF packet.

### 7.6.35 180h H_CH0Config_0(Host Channel 0 Configuration0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 180h | H_CH0Config_0 | R / W | 7: ACK_Cnt[3] | Channel 0 ACK Count | | 10h |
| | | | | 6: ACK_Cnt[2] | | | |
| | | | | 5: ACK_Cnt[1] | | | |
| | | | | 4: ACK_Cnt[0] | | | |
| | | | R / W | 3: SpeedMode[1] | Channel 0 Speed Mode | | |
| | | | | 2: SpeedMode[0] | | | |
| | | | R / W | 1: Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 0: TranGo | 0: Stand by | 1: Transaction Start | |

This register is used to make basic settings of channel CH0 during host operation.

#### Bit7-4 ACK_Cnt [3:0]

These bits set the number of ACK count in a transfer performed on channel CH0.

When ACK count reaches the set value, the TranACK bit in the H_CH0IntStat register is set.

0000: ACK is counted up to 16 times.

0001 to 1111: ACK is counted anywhere between 1 to 15 times.

During the execution of the control transfer support function, only the transactions in the data stage are counted and transactions from setup stage and status stage are not counted.

#### Bits3-2 SpeedMode [1:0]

These bits set the operation mode of the device that performs a transfer on channel CH0.

00: HS mode — Set this for HS devices.

01: FS mode — Set this for FS devices.

10: Reserved — Use of this value is prohibited.

11: LS mode — Set this for FS devices.

#### Bit1 Toggle

This bit sets the initial value of the toggle sequence bit with which a transaction is to be started. While a transaction is under execution and after a transaction is complete, this bit indicates the status of the toggle sequence bit.

0: Toggle 0

1: Toggle 1

**Bit0 TranGo**

Setting this bit to 1 causes a transaction on channel CH0 to start. Transaction process can be stopped by clearing this bit to 0 after a transaction started. This bit also serves as a status indicating whether a transaction on channel CH0 is under execution or not.

0: Stops a transaction (transaction stopped)

1: Starts a transaction (transaction under execution)

When a transfer for the number of bytes set by the H_CH0TotalSize_H through L registers is complete, the TotalSizeCmp bit in the H-CH0IntStat register is set to 1, at which time this bit is automatically reset to 0. Furthermore, this bit is reset to 0 when the ChangeCondition bit in the H_CH0IntStat register is set. In that case, inspect the H_CH0ConditionCode register to find a reason for which the condition bit was set.

If a transaction is stopped by clearing this bit, the ChangeCondition bit in the H_CH0IntStat register is set when the transaction being processed is complete. Even when a transaction is stopped, the data present in the FIFO, the (remaining) total size, and channel-related settings are left intact. Therefore, a transaction can be resumed from where it stopped by setting this bit to 1 again. (To perform a new transaction, clear the FIFO and set channel information over again.)

This setting is not required when using the control transfer support function.

### 7.6.36 181h H_CH0Config_1(Host Channel 0 Configuration1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 181h | H_CH0Config_1 | R / W | 7: TID[1] | Channel 0 Transaction ID | | 00h |
| | | | | 6: TD[0] | | | |
| | | | | 5: | 0: | | |
| | | | | 4: | 0: | | |
| | | | | 3: | 0: | | |
| | | | | 2: | 0: | | |
| | | | | 1: | 0: | | |
| | | | | 0: | 0: | | |

This register is used to make basic settings of channel CH0 during host operation.

**Bits7-6 TID[1:0]**

These bits set the type of token (SETUP, OUT, or IN) that is issued on channel CH0. Settings of these bits have no effect when a transaction is started by setting the CTL_SupportGo bit in the CTL_SupportControl register to 1.

00: SETUP — Issues a SETUP token.

01: OUT — Issues an OUT token.

10: IN — Issues an IN token.

11: Reserved — Use of this value is prohibited.

This setting is not required when using the control transfer support function.

**Bits5-0 Reserved**

### 7.6.37 183h H_CH0MaxPktSize (Host Channel 0 Max Packet Size)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 183h | H_CH0MaxPktSize | | 7: | 0: | 1: | 00h |
| | | | R / W | 6: MaxPktSize[6] | Channel 0 Max Packet Size | | |
| | | | | 5: MaxPktSize[5] | | | |
| | | | | 4: MaxPktSize[4] | | | |
| | | | | 3: MaxPktSize[3] | | | |
| | | | | 2: MaxPktSize[2] | | | |
| | | | | 1: MaxPktSize[1] | | | |
| | | | | 0: MaxPktSize[0] | | | |

This register sets MaxPacketSize of channel CH0 during host operation.

**Bit7 Reserved**

**Bits6-0 MaxPktSize[6:0]**

These bits set the MaxPacketSize of channel CH0.

During LS 8 bytes

During FS 8, 16, 32, or 64 bytes

During HS 64 bytes

Set one of the above transfer sizes.

Setting any other transfer size is prohibited.

### 7.6.38 186h H_CH0TotalSize_H (Host Channel 0 Total Size High)

### 7.6.39 187h H_CH0TotalSize_L (Host Channel 0 Total Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 186h | H_CH0TotalSize_H | R / W | 7: TotalSize[15] | Channel 0 Total Size High | 00h |
| | | | | 6: TotalSize[14] | | |
| | | | | 5: TotalSize[13] | | |
| | | | | 4: TotalSize[12] | | |
| | | | | 3: TotalSize[11] | | |
| | | | | 2: TotalSize[10] | | |
| | | | | 1: TotalSize[9] | | |
| | | | | 0: TotalSize[8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 187h | H_CH0TotalSize_L | R / W | 7: TotalSize[7] | Channel 0 Total Size Low | 00h |
| | | | | 6: TotalSize[6] | | |
| | | | | 5: TotalSize[5] | | |
| | | | | 4: TotalSize[4] | | |
| | | | | 3: TotalSize[3] | | |
| | | | | 2: TotalSize[2] | | |
| | | | | 1: TotalSize[1] | | |
| | | | | 0: TotalSize[0] | | |

This register sets the Total Size of the data to be transferred on channel CH0 during host operation.

#### 186h.Bit7-0, 187h.Bit7-0 TotalSize[15:0]

These bits set a total number of data bytes to be transferred on channel CH0 (maximum 65,535 bytes: approx. 64 Kbytes).

Once a transaction is started by the TranGo bit in the H_CH0Config_0 register, the remaining number of bytes to be transferred can be read through this register.

When bits 15 to 8 (H_CH0TotalSize_H register) are read, the value of bits 7 to 0 (H_CH0TotalSize_L register) is fixed. (Even when the read value is fixed, the internal counter of the LSI continues counting.)

Therefore, to read the remaining number of transfer bytes through 8-bit register accesses, read H_CH0TotalSize_H and H_CH0TotalSize_L in that order.

Note also that if an OUT transaction is executed with TotalSize = 0, a zero-length packet is transmitted.

The setting of this register is not required when executing a SETUP transaction or using the control transfer support function.

### 7.6.40 188h H_CH0HubAdrs (Host Channel 0 Hub Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 188h | H_CH0HubAdrs | R / W | 7: HubAdrs[3] | Channel 0 Hub Address | | 00h |
| | | | | 6: HubAdrs[2] | | | |
| | | | | 5: HubAdrs[1] | | | |
| | | | | 4: HubAdrs[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Port[2] | Channel 0 Port Number | | |
| | | | | 1: Port[1] | | | |
| | | | | 0: Port[0] | | | |

This register sets the hub that is connected to channel CH0 during host operation.

**Bits7 HubAdrs[3:0]**

These bits set the USB address of the hub to which a function is connected that performs a transfer on channel CH0. Any value in the range of 0 to 15 can be set.

**Bit3 Reserved**

**Bits2-0 Port[2:0]**

These bits set the port number of the hub to which a function is connected that performs a transfer on channel CH0. Any value in the range of 0 to 7 can be set.

### 7.6.41 189h H_CH0FuncAdrs (Host Channel 0 Function Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 189h | H_CH0FuncAdrs | R / W | 7: FuncAdrs[3] | Channel 0 Function Address | 00h |
| | | | | 6: FuncAdrs[2] | | |
| | | | | 5: FuncAdrs[1] | | |
| | | | | 4: FuncAdrs[0] | | |
| | | | R / W | 3: EP_Number[3] | Channel 0 Endpoint Number | |
| | | | | 2: EP_Number[2] | | |
| | | | | 1: EP_Number[1] | | |
| | | | | 0: EP_Number[0] | | |

This register sets the address of a function that performs a transfer on channel CH0 during host operation.

**Bits7-4 FuncAdrs[3:0]**

These bits set the USB address of the function that includes the endpoint managed by channel CH0. Any value in the range of 0 to 15 can be set.

**Bits3-0 EP_Number[3:0]**

This bit sets the endpoint number that performs a transfer on channel CH0. Any value in the range of 0 to 15 can be set.

### 7.6.42 18Bh CTL_SupportControl (Host ControlTransfer Support Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 18Bh | H_CTL_Support- | | 7: | 0: | 1: | XXh |
| | | Control | | 6: | 0: | 1: | |
| | | | R | 5: CTL_SupportState[1] | ControlTransfer Support State | | |
| | | | | 4: CTL_SupportState[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: CTL_SupportGo | 0: Stand by | 1: Control Transfer Go | |

This register sets the support functions used for control transfers on channel CH0 during host operation.

**Bits7-6 Reserved**

**Bits5-4 CTL_SupportState[1:0]**

These bits indicate which stage is under execution when a transfer is performed using the control transfer support function after setting the CTL_SupportGo bit to 1.

00: Idle — Indicates that transfer is not executed yet or a transfer has completed without errors.

01: Setup Stage — Indicates that a setup stage is under execution.

10: Data Stage — Indicates that a data stage is under execution.

11: Status Stage — Indicates that a status stage is under execution.

**Bits3-1 Reserved**

**Bit0 CTL_SupportGo**

Setting this bit to 1 enables the control transfer support function so that a control transfer on channel CH0 is automatically performed, ranging from the setup stage to the status stage with or without a data stage included.

In the setup stage, a SETUP token is automatically sent out, and the requests set in H_CH0SETUP_0 through 7 are transmitted.

Next, if a data stage is involved, a transaction is automatically executed in a specified direction with a specified size.

Finally in the status stage, an appropriate PID token is issued depending on whether a data stage is involved and the direction of a data stage, and a zero-length packet is transmitted/received between the host and device.

When the above transition and stage sequence is completed without errors, the CTL_SupportCmp bit in the H_CH0IntStat register is set. If a packet error was detected during the sequence, the CTL_SupportStop bit in the H_CH0IntStat register is set, causing the transaction to stop. In that case, inspect the ConditionCode register to find the cause of the error.

When a control transfer has finished (whether terminated normally or in error), this bit is automatically cleared.

A control transfer can be aborted by clearing this bit during execution of the control transfer support function. If a control transfer finishes normally in the status stage, the CTL_SupportCmp bit is set. Otherwise, the CTL_SupportStop bit is set. Inspect CTL_SupportState to find the stage in which a control transfer stopped.

### 7.6.43 18Eh H_CH0ConditionCode (Host Channel 0 Condition Code)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 18Eh | H_CH0ConditionCode | | 7: | 0: | 1: | 00h |
| | | | R | 6: ConditionCode[2] | Channel 0 Condition Code | | |
| | | | | 5: ConditionCode[1] | | | |
| | | | | 4: ConditionCode[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register indicates the result of a transfer that was completed on channel CH0 during host operation.

**Bit7 Reserved**

**Bits6-4 ConditionCode[2:0]**

These bits indicate the result of a transfer that was completed on channel CH0.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* In the event the CRC error and the bit stuffing error is detected simultaneously, the errors will be processed as retry errors.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* In the event the CRC error and the bit stuffing error is detected simultaneously, the errors will be processed as retry errors.<br>* If the size of the received data packet is less than the MaxPacketSize and the data toggle included in the data packet does not match the expected value, the error will be treated as Toggle Mismatch Error and not Data Overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* In the event the CRC error and the bit stuffing error is detected simultaneously, the errors will be processed as retry errors. |

| Code | Meaning | Description |
|---|---|---|
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• The data toggle included in the data packet from the endpoint does not match the expected value. (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.44 190h H_CHaConfig_0(Host Channel a Configuration0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 190h | H_CHaConfig_0 | R / W | 7: ACK_Cnt[3] | Channel [a] ACK Count | | 00h |
| | | | | 6: ACK_Cnt[2] | | | |
| | | | | 5: ACK_Cnt[1] | | | |
| | | | | 4: ACK_Cnt[0] | | | |
| | | | R / W | 3: SpeedMode[1] | Channel [a] Speed Mode | | |
| | | | | 2: SpeedMode[0] | | | |
| | | | R / W | 1: Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 0: TranGo | 0: Stand by | 1: Transaction Start | |

This register is used to make basic settings of channel CHa during host operation.

#### Bits7-4 ACK_Cnt [3:0]

These bits set the number of ACK count in a transfer performed on channel CHa.

When ACK count reaches the set value, the TranACK bit in the H_CHaIntStat register is set.

0000: ACK is counted up to 16 times.

0001 to 1111: ACK is counted anywhere between 1 to 15 times.

During the execution of the Bulk Only Transfer Support function, only the data transport transactions are counted and CBW/CSW transactions are not counted.

#### Bits3-2 SpeedMode [1:0]

These bits set the operation mode of the device that performs a transfer on channel CHa.

00: HS mode — Set this for HS devices.

01: FS mode — Set this for FS devices.

10 to 11: Reserved — Use of this value is prohibited.

#### Bit1 Toggle

This bit sets the initial value of the toggle sequence bit with which a transaction is to be started. While a transaction is under execution and after a transaction is complete, this bit indicates the status of the toggle sequence bit.

0: Toggle 0

1: Toggle 1

The setting of this bit is not required when using the Bulk Only Support function.

**Bit0 TranGo**

Setting this bit to 1 causes a transaction on channel CHa to start. Transaction processing can be stopped by clearing this bit to 0 after a transaction started. This bit also serves as a status indicating whether a transaction on channel CHa is under execution or not.

0: Stops a transaction (transaction stopped)

1: Starts a transaction (transaction under execution)

When a transfer for the number of bytes set by the H_CHaTotalSize_HH through LL registers is complete, the TranCmp bit in the H-CHaIntStat register is set to 1, at which time this bit is automatically reset to 0. Furthermore, this bit is reset to 0 when the ChangeCondition bit in the H_CHaIntStat register is set. In that case, inspect the H_CHaConditionCode register to find a reason for which the condition bit was set.

If a transaction is stopped by clearing this bit, the ChangeCondition bit in the H_CHaIntStat register is set when the transaction being processed is complete. Even when a transaction is stopped, the data present in the FIFO, the (remaining) total size, and channel-related settings are left intact. Therefore, a transaction can be resumed from where it stopped by setting this bit to 1 again. (To perform a new transaction, clear the FIFO and set channel information over again.)

The setting of this bit is not required when using the Bulk Only Support function.

### 7.6.45 191h H_CHaConfig_1(Host Channel a Configuration1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 191h | H_CHaConfig_1 | R / W | 7: TID[1] | Channel a Transaction ID | | 00h |
| | | | | 6: TID[0] | | | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | R / W | 3: AutoZerolen | 0: Do nothing | 1: Add Zerolen | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: TotalSizeFree | 0: Do nothing | 1: Total Size Free | |

This register is used to make basic settings of channel CHa during host operation.

**Bits7-6 TID[1:0]**

These bits set the type of token (OUT or IN) that is issued on channel CHa. Settings of these bits have no effect when a transaction is started by setting the CTL_SupportGo bit in the CTL_SupportControl register to 1.

00: Reserved — Use of this value is prohibited.

01: OUT — Issues an OUT token.

10: IN — Issues an IN token.

11: Reserved — Use of this value is prohibited.

The setting of this bit is not required when using the Bulk Only support function

**Bits5-4 Reserved**

**Bit3 AutoZerolen**

Setting this bit to 1 enables the auto zero-length function, so that when a transfer for the data bytes set by H_ChaTotalSizeHH through LL registers is completed exactly with MaxPacketSize, a zero-length packet is automatically attached at the end of the transfer. This bit is effective for only OUT transfers.

**Bits2-1 Reserved**

**Bit0 TotalSizeFree**

Setting this bit to 1 makes the transfer size infinite regardless of the value set in the H_CHaTotalSizeHH through LL registers.

### 7.6.46 192h H_CHaMaxPktSize_H (Host Channel a Max Packet Size High)

### 7.6.47 193h H_CHaMaxPktSize_L (Host Channel a Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 192h | H_CHaMaxPktSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: MaxPktSize[9] | Channel a Max Packet Size High | | |
| | | | | 0: MaxPktSize[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 193h | H_CHaMaxPktSize_L | R / W | 7: MaxPktSize[7] | Channel a Max Packet Size Low | 00h |
| | | | | 6: MaxPktSize[6] | | |
| | | | | 5: MaxPktSize[5] | | |
| | | | | 4: MaxPktSize[4] | | |
| | | | | 3: MaxPktSize[3] | | |
| | | | | 2: MaxPktSize[2] | | |
| | | | | 1: MaxPktSize[1] | | |
| | | | | 0: MaxPktSize[0] | | |

This register sets MaxPacketSize of channel CHa during host operation.

#### 192h.Bit7-2 Reserved

Do not write 1 to the reserved bits.

#### 192h.Bit1-0, 193h.Bit7-0 MaxPktSize[9:0]

These bits set the MaxPacketSize of channel CHa.

During FS (either 32 or 64 bytes when the bulk-only support function is used)

During HS 512 bytes

Set one of the above transfer sizes.

Setting any other transfer size is prohibited.

### 7.6.48 194h H_CHaHubAdrs (Host Channel a Hub Address)

### 7.6.49 195h H_CHaTotalSize_HL (Host Channel a Total Size High-Low)

### 7.6.50 196h H_CHaTotalSize_LH (Host Channel a Total Size Low-High)

### 7.6.51 197h H_CHaTotalSize_LL (Host Channel a Total Size Low-Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 194h | H_CHaTotalSize_HH | R / W | 7: TotalSize[31] | Channel a Total Size High-High | 00h |
| | | | | 6: TotalSize[30] | | |
| | | | | 5: TotalSize[29] | | |
| | | | | 4: TotalSize[28] | | |
| | | | | 3: TotalSize[27] | | |
| | | | | 2: TotalSize[26] | | |
| | | | | 1: TotalSize[25] | | |
| | | | | 0: TotalSize[24] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 195h | H_CHaTotalSize_HL | R / W | 7: TotalSize[23] | Channel a Total Size High-Low | 00h |
| | | | | 6: TotalSize[22] | | |
| | | | | 5: TotalSize[21] | | |
| | | | | 4: TotalSize[20] | | |
| | | | | 3: TotalSize[19] | | |
| | | | | 2: TotalSize[18] | | |
| | | | | 1: TotalSize[17] | | |
| | | | | 0: TotalSize[16] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 196h | H_CHaTotalSize_LH | R / W | 7: TotalSize[15] | Channel a Total Size Low-High | 00h |
| | | | | 6: TotalSize[14] | | |
| | | | | 5: TotalSize[13] | | |
| | | | | 4: TotalSize[12] | | |
| | | | | 3: TotalSize[11] | | |
| | | | | 2: TotalSize[10] | | |
| | | | | 1: TotalSize[9] | | |
| | | | | 0: TotalSize[8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 197h | H_CHaTotalSize_LL | R / W | 7: TotalSize[7] | Channel a Total Size Low-Low | 00h |
| | | | | 6: TotalSize[6] | | |
| | | | | 5: TotalSize[5] | | |
| | | | | 4: TotalSize[4] | | |
| | | | | 3: TotalSize[3] | | |
| | | | | 2: TotalSize[2] | | |
| | | | | 1: TotalSize[1] | | |
| | | | | 0: TotalSize[0] | | |

These registers set the Total Size of the data to be transferred on channel CHa during host operation.

**194h.Bit7-0, 195h.Bit7-0, 196h.Bit7-0, 197h.Bit7-0 TotalSize[31:0]**

These bits set the total number of data bytes to be transferred on channel CHa (maximum 4,294,967,295 bytes: approx. 4 Gbytes).

Once a transaction is started by the TranGo bit in the H_CHaConfig_0 register, the remaining number of bytes to be transferred can be read through this register.

When bits 31 to 24 (H_CHaTotalSize_HH register) are read, the values of bits 23 to 16 (H_CHaTotalSize_HL register), bits 15 to 8 (H_CHaTotalSize_LH register), and bits 7 to 0 (H_CHaTotalSize_LL register) all are fixed. (Even when the read value is fixed, the internal counter of the LSI continues counting.)

Therefore, to read the remaining number of transfer bytes through 8-bit register accesses, read H_CHaTotalSize_HH, H_CHaTotalSize_HL, H_CHaTotalSize_LH, and H_CHaTotalSize_LL in that order.

Note also that if an OUT transaction is executed with TotalSize = 0, a zero-length packet is transmitted.

The setting of this register is not required when using the Bulk Only Support function.

### 7.6.52 198h H_CHaHubAdrs (Host Channel a Hub Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 198h | H_CHaHubAdrs | R / W | 7: HubAdrs[3] | Channel a Hub Address | | 00h |
| | | | | 6: HubAdrs[2] | | | |
| | | | | 5: HubAdrs[1] | | | |
| | | | | 4: HubAdrs[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Port[2] | Channel a Port Number | | |
| | | | | 1: Port[1] | | | |
| | | | | 0: Port[0] | | | |

This register sets the hub that is connected to channel CHa during host operation.

**Bit7 HubAdrs[3:0]**

This bit set sthe USB address of the hub to which a function is connected that performs a transfer on channel CHa.

Any value in the range of 0 to 15 can be set.

**Bit3 Reserved**

**Bits2-0 Port[2:0]**

These bits set the port number of the hub to which a function is connected that performs a transfer on channel CHa.

Any value in the range of 0 to 7 can be set.

### 7.6.53 199h H_CHaFuncAdrs (Host Channel a Function Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 199h | H_CHaFuncAdrs | R / W | 7: FuncAdrs[3] | Channel a Function Address | 00h |
| | | | | 6: FuncAdrs[2] | | |
| | | | | 5: FuncAdrs[1] | | |
| | | | | 4: FuncAdrs[0] | | |
| | | | R / W | 3: EP_Number[3] | Channel a Endpoint Number | |
| | | | | 2: EP_Number[2] | | |
| | | | | 1: EP_Number[1] | | |
| | | | | 0: EP_Number[0] | | |

This register sets the address of a function that performs a transfer on channel CHa during host operation.

**Bits7-4 FuncAdrs[3:0]**

These bits set the USB address of the function that includes the endpoint managed by channel CHa. Any value in the range of 0 to 15 can be set.

**Bits3-0 EP_Number[3:0]**

These bits set the endpoint number that performs a transfer on channel CHa. Any value in the range of 0 to 15 can be set.

The setting of this bit is not required when using the Bulk Only support function

### 7.6.54 19Ah H_CHaBO_SupporotCtl (Host CHa Bulk Only Transfer Supporot Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 19Ah | H_CHaBO_ | | 7: | 0: | 1: | 00h |
| | | Supporot-Ctl | | 6: | 0: | 1: | |
| | | | R / W | 5: BO_TransportState[1] | Bulk Only Transfer Transport State | | |
| | | | | 4: BO_TransportState[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: BO_SupportGo | 0: Stand by | 1: BO Transfer Go | |

This register sets the Bulk Only transfer support functions on channel CHa during host operation.

**Bits7-6 Reserved**

**Bits5-4 BO_TransportState[1:0]**

These bits indicate which transport is under execution when a transfer is performed using the Bulk Only transfer support function after setting the BO_SupportGo bit to 1.

00: Idle — Indicates that transfer is not executed yet or a transfer is completed without errors.

01: CBW Transport — Indicates that a CBW transport is under execution.

10: Data Transport — Indicates that a data transport is under execution.

11: CSW Transport — Indicates that a CSW transport is under execution.

**Bits3-1 Reserved**

**Bit0 BO_SupportGo**

Setting this bit to 1 enables the Bulk Only transfer support function so that a Bulk Only transfer on channel CHa is automatically performed, ranging from the CBW transport to CSW transport with or without a data transport included.

In the CBW transport, an OUT token is automatically sent out, and the data set in the CBW area of the FIFO is transmitted.

Next, if a data transport is involved, a data transport is automatically executed in a specified direction with a specified size.

Finally in the CSW transport, an IN token is automatically sent out, and data is received in the CSW area of the FIFO.

When the above transports are completed without errors, the BO_SupportCmp bit in the H_BO_SupportIntStat register is set. If a packet error is detected during the transport or the CSW value is found inappropriate, the BO_SupportStop bit in the H_CHaIntStat register is set, causing the transaction to stop. In that case, inspect the H_CHaConditionCode register to find the cause of the error. If the value of ConditionCode = 000 when the BO_SupportStop bit is set to 1, it means

that the CSW value is inappropriate. Inspect the stopped transport by shifting to the BO_TranportState.

This bit is automatically cleared when a series of transports has finished (whether terminated normally or in error).

Transport processing can be aborted by clearing this bit during execution of the bulk-only support function. In this case, the BO_SupportCmp bit is set when CSW transport finishes normally, or the BO_SupportStop bit is set otherwise. Inspect BO_TransportState to determine if the transport has halted.

### 7.6.55 19Bh H_CHaBO_CSW_RcvDataSize (Host CHa Bulk Only Transfer Support CSW Receive Data Size)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 19Bh | H_ChaBO_CSW_RcvDataSize | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | R | 3: CSW_RcvDataSize[3] | CSW Resceive Data Size | | |
| | | | | 2: CSW_RcvDataSize[2] | | | |
| | | | | 1: CSW_RcvDataSize[1] | | | |
| | | | | 0: CSW_RcvDataSize[0] | | | |

This register indicates the number of data bytes received during execution of the CSW transport function when using the Bulk Only transfer support function on channel CHa during host operation.

**Bits7-4 Reserved**

**Bits3-0 CSW_RcvDataSize[3:0]**

These bits indicate the number of transmitted data bytes of CSW.

If less than 13 bytes of data is received in a CSW transport, this register will show the number of trasmitted data bytes.

If a hanshake is received in a CSW transport, the values on this register will have no significance in cases other than CSW transport.

### 7.6.56 19Ch H_ChaBQ_OUT_EP_Ctl (Host CHa Bulk Only Transfer Support OUT Endpoint Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 19Ch | H_ChaBO_OUT_EP_Ctl | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: OUT_Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 3: OUT_EP_Number[3] | OUT EP Number | | |
| | | | | 2: OUT_EP_Number[2] | | | |
| | | | | 1: OUT_EP_Number[1] | | | |
| | | | | 0: OUT_EP_Number[0] | | | |

This register sets the Bulk Only transfer support functions used on channel CHa during host operation.

**Bits7-5 Reserved**

**Bit4 OUT_Toggle**

This bit sets the initial value of the toggle sequence bit for an OUT direction transfer (CBW transport or Data OUT transport) to be performed using the Bulk Only transfer support function after setting the BO_SupportGo bit in the H_CBW_Control register to 1.

0: Toggle 0

1: Toggle 1

In addition, when an OUT-direction transport is completed without errors, this bit will retain the toggle sequence bit automatically.

**Bits3-0 OUT_EP_Number[3:0]**

These bits set the endpoint number of the destination device for an OUT direction transfer (CBW transport or Data OUT transport) to be performed using the Bulk Only transfer support function after setting the BO_SupportGo bit in the H_CBW_Control register to 1.

Any number in the range of 0 to 15 can be set.

### 7.6.57 19Dh H_ChaBO_IN_EP_Ctl (Host CHa Bulk Only Transfer Support IN Endpoint Control)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 19Dh | H_ChaBO_IN_EP_Ctl | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | R / W | 4: IN_Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 3: IN_EP_Number[3] | IN EP Number | | |
| | | | | 2: IN_EP_Number[2] | | | |
| | | | | 1: IN_EP_Number[1] | | | |
| | | | | 0: IN_EP_Number[0] | | | |

This register sets the Bulk Only transfer support functions used on channel CHa during host operation.

**Bits7-5 Reserved**

**Bit4 IN_Toggle**

This bit sets the initial value of the toggle sequence bit for an IN direction transfer (CBW transport or Data IN transport) to be performed using the Bulk Only transfer support function after setting the BO_SupportGo bit in the H_CBW_Control register to 1.

0: Toggle 0

1: Toggle 1

In addition, when an IN-direction transport is completed without errors, this bit will retain the toggle sequence bit automatically.

**Bits3-0 IN_EP_Number[3:0]**

These bits set the endpoint number of the destination device for an IN direction transfer (CBW transport or Data IN transport) to be performed using the Bulk Only transfer support function after setting the BO_SupportGo bit in the H_CBW_Control register to 1.

Any number in the range of 0 to 15 can be set.

### 7.6.58 19Eh H_CHaConditionCode (Channel a Condition Code)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 19Eh | H_CHaConditionCode | | 7: | 0: | 1: | 00h |
| | | | R | 6: ConditionCode[2] | Channel a Condition Code | | |
| | | | | 5: ConditionCode[1] | | | |
| | | | | 4: ConditionCode[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register indicates the result of a transfer that was completed on channel CHa during host operation.

**Bit7 Reserved**

**Bits6-4 ConditionCode[2:0]**

These bits indicate the result of a transfer that was completed on channel CHa.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* In the event the CRC error and the bit stuffing error is detected simultaneously, the errors will be processed as retry errors.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* In the event the CRC error and the bit stuffing error is detected simultaneously, the errors will be processed as retry errors.<br>* If the size of the received data packet is less than the MaxPacketSize and the data toggle included in the data packet does not match the expected value, the error will be treated as Toggle Mismatch Error and not Data Overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* In the event the CRC error and the bit stuffing error is detected simultaneously, the errors will be processed as retry errors. |

| Code | Meaning | Description |
|---|---|---|
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.59 1A0h H_CHbConfig_0(Host Channel b Configuration0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1A0h | CHbConfig_0 | R / W | 7: ACK_Cnt[3] | Channel b ACK Count | | 00h |
| | | | | 6: ACK_Cnt[2] | | | |
| | | | | 5: ACK_Cnt[1] | | | |
| | | | | 4: ACK_Cnt[0] | | | |
| | | | R / W | 3: SpeedMode[1] | Channel b Speed Mode | | |
| | | | | 2: SpeedMode[0] | | | |
| | | | R / W | 1: Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 0: TranGo | 0: Stand by | 1: Transaction Start | |

This register is used to make basic settings of channel CHb during host operation.

#### Bits7-4 ACK_Cnt [3:0]

These bits set a number of ACK counts in a transfer performed on channel CHb.

When ACK count reaches the set value, the TranACK bit in the H_CHbIntStat register is set.

0000: ACK is counted up to 16 times.

0001 to 1111: ACK is counted anywhere between 1 to 15 times.

#### Bits3-2 SpeedMode [1:0]

These bits set the operation mode of the device that performs a transfer on channel CHb.

00: HS mode — Set this for HS devices.

01: FS mode — Set this for FS devices.

10: Reserved — Use of this value is prohibited.

11: LS mode — Set this for LS devices.

#### Bit1 Toggle

This bit sets the initial value of the toggle sequence bit with which a transaction is to be started. While a transaction is under execution and after a transaction is complete, this bit indicates the status of the toggle sequence bit.

0: Toggle 0

1: Toggle 1

#### Bit0 TranGo

Setting this bit to 1 causes a transaction on channel CHb to start. Transaction processing can be stopped by clearing this bit to 0 after a transaction started. This bit also serves as a status indicating whether a transaction on channel CHb is under execution or not.

0: Stops a transaction (transaction stopped)

1: Starts a transaction (transaction under execution)

When a transfer for the number of bytes set by the H_CHbTotalSize_HH through LL registers is complete, the TranCmp bit in the H-CHbIntStat register is set to 1, at which time this bit is automatically reset to 0. Furthermore, this bit is reset to 0 when the ChangeCondition bit in the H_CHbIntStat register is set. In that case, inspect the H_CHbConditionCode register to find a reason for which the condition bit was set.

If a transaction is stopped by clearing this bit, the ChangeCondition bit in the H_CHbIntStat register is set when the transaction being processed is complete. Even when a transaction is stopped, the data present in the FIFO, the (remaining) total size, and channel-related settings are left intact. Therefore, a transaction can be resumed from where it stopped by setting this bit to 1 again. (To perform a new transaction, clear the FIFO and set channel information over again.)

### 7.6.60 1A1h H_CHbConfig_1(Host Channel b Configuration1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1A1h | H_CHbConfig_1 | R / W | 7: TID[1] | Channel b Transaction ID | | 00h |
| | | | | 6: TID[0] | | | |
| | | | R / W | 5: TranType[1] | Channel b Transfer Type | | |
| | | | | 4: TranType[0] | | | |
| | | | R / W | 3: AutoZerolen | 0: Do nothing | 1: Add Zerolen | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: TotalSizeFree | 0: Do nothing | 1: Total Size Free | |

This register is used to make basic settings of channel CHb during host operation.

**Bits7-6 TID[1:0]**

These bits set the type of token (OUT or IN) that is issued on channel CHb.

00: Reserved — Use of this value is prohibited.

01: OUT — Issues an OUT token.

10: IN — Issues an IN token.

11: Reserved — Use of this value is prohibited.

**Bits5-4 TranType[1:0]**

These bits set the type of transfer that is performed on channel CHb.

00: Reserved — Use of this value is prohibited.

01: Reserved — Use of this value is prohibited.

10: Bulk — Performs a bulk transfer.

11: Interrupt — Performs an interrupt transfer.

**Bit3 AutoZerolen**

Setting this bit to 1 enables the auto zero-length function, so that when a transfer for the data bytes set by H_CHbTotalSizeHH through LL registers is completed exactly with MaxPacketSize, a zero-length packet is automatically attached at the end of the transfer. This bit is effective for only OUT transfers.

**Bits2-1 Reserved**

**Bit0 TotalSizeFree**

Setting this bit to 1 makes the transfer size infinite regardless of the value set in the H_CHbTotalSizeHH through LL registers.

### 7.6.61 1A2h H_CHbMaxPktSize_H (Host Channel b Max Packet Size High)

### 7.6.62 1A3h H_CHbMaxPktSize_L (Host Channel b Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1A2h | H_CHbMaxPktSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: MaxPktSize[9] | Channel b Max Packet Size High | | |
| | | | | 0: MaxPktSize[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1A3h | H_CHbMaxPktSize_L | R / W | 7: MaxPktSize[7] | Channel b Max Packet Size Low | 00h |
| | | | | 6: MaxPktSize[6] | | |
| | | | | 5: MaxPktSize[5] | | |
| | | | | 4: MaxPktSize[4] | | |
| | | | | 3: MaxPktSize[3] | | |
| | | | | 2: MaxPktSize[2] | | |
| | | | | 1: MaxPktSize[1] | | |
| | | | | 0: MaxPktSize[0] | | |

This register sets MaxPacketSize of channel CHb during host operation.

**1A2h.Bit7-2 Reserved**

Do not write 1 to the reserved bits.

**1A2h.Bit1-0, 1A3h.Bits7-0 MaxPktSize[9:0]**

These bits set the MaxPacketSize of channel CHb.

When using this channel for bulk transfers, set one of the following transfer sizes.

During FS 8, 16, 32, or 64 bytes

During HS 512 bytes

When using this channel for interrupt transfers, set any desired transfer size within the limits given below.

During LS Up to 8 bytes

During FS Up to 64 bytes

During HS Up to 512 bytes

Setting any other transfer size is prohibited.

### 7.6.63 1A4h H_ChbTotalSize_HH (Host Channel b Total Size High-High)

### 7.6.64 1A5h H_CHbTotalSize_HL (Host Channel b Total Size High-Low)

### 7.6.65 1A6h H_CHbTotalSize_LH (Host Channel b Total Size Low-High)

### 7.6.66 1A7h H_CHbTotalSize_LL (Host Channel b Total Size Low-Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1A4h | H_CHbTotalSize_HH | R / W | 7: TotalSize[31] | Channel b Total Size High-High | 00h |
| | | | | 6: TotalSize[30] | | |
| | | | | 5: TotalSize[29] | | |
| | | | | 4: TotalSize[28] | | |
| | | | | 3: TotalSize[27] | | |
| | | | | 2: TotalSize[26] | | |
| | | | | 1: TotalSize[25] | | |
| | | | | 0: TotalSize[24] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1A5h | H_CHbTotalSize_HL | R / W | 7: TotalSize[23] | Channel b Total Size High-Low | 00h |
| | | | | 6: TotalSize[22] | | |
| | | | | 5: TotalSize[21] | | |
| | | | | 4: TotalSize[20] | | |
| | | | | 3: TotalSize[19] | | |
| | | | | 2: TotalSize[18] | | |
| | | | | 1: TotalSize[17] | | |
| | | | | 0: TotalSize[16] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1A6h | H_CHbTotalSize_LH | R / W | 7: TotalSize[15] | Channel b Total Size Low-High | 00h |
| | | | | 6: TotalSize[14] | | |
| | | | | 5: TotalSize[13] | | |
| | | | | 4: TotalSize[12] | | |
| | | | | 3: TotalSize[11] | | |
| | | | | 2: TotalSize[10] | | |
| | | | | 1: TotalSize[9] | | |
| | | | | 0: TotalSize[8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1A7h | H_CHbTotalSize_LL | R / W | 7: TotalSize[7] | Channel b Total Size Low-Low | 00h |
| | | | | 6: TotalSize[6] | | |
| | | | | 5: TotalSize[5] | | |
| | | | | 4: TotalSize[4] | | |
| | | | | 3: TotalSize[3] | | |
| | | | | 2: TotalSize[2] | | |
| | | | | 1: TotalSize[1] | | |
| | | | | 0: TotalSize[0] | | |

These registers set the Total Size of the data to be transferred on channel CHb during host operation.

#### 1A4h.Bit7-0, 1A5h.Bit7-0, 1A6h.Bit7-0, 1A7h.Bit7-0 TotalSize[31:0]

These bits set a total number of data bytes to be transferred on channel CHb (maximum 4,294,967,295 bytes: approx. 4 Gbytes).

Once a transaction is started by the TranGo bit in the H_CHbConfig_0 register, the remaining number of bytes to be transferred can be read through this register.

When bits 31 to 24 (H_CHbTotalSize_HH register) are read, the values of bits 23 to 16 (H_CHbTotalSize_HL register), bits 15 to 8 (H_CHbTotalSize_LH register), and bits 7 to 0 (H_CHbTotalSize_LL register) all are fixed. (Even when the read value is fixed, the internal counter of the LSI continues counting.)

Therefore, to read the remaining number of transfer bytes through 8-bit register accesses, read H_CHaTotalSize_HH, H_CHaTotalSize_HL, H_CHaTotalSize_LH, and H_CHaTotalSize_LL in that order.

Note also that if an OUT transaction is executed with TotalSize = 0, a zero-length packet is transmitted.

### 7.6.67 1A8h H_CHbHubAdrs (Host Channel b Hub Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1A8h | H_CHbHubAdrs | R / W | 7: HubAdrs[3] | Channel b Hub Address | | 00h |
| | | | | 6: HubAdrs[2] | | | |
| | | | | 5: HubAdrs[1] | | | |
| | | | | 4: HubAdrs[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Port[2] | Channel b Port Number | | |
| | | | | 1: Port[1] | | | |
| | | | | 0: Port[0] | | | |

This register sets the hub that is connected to channel CHb during host operation.

**Bits7-4 HubAdrs[3:0]**

These bits set the USB address of the hub to which a function is connected that performs a transfer on channel CHb.

Any value in the range of 0 to 15 can be set.

**Bit3 Reserved**

**Bits2-0 Port[2:0]**

These bits set the port number of the hub to which a function is connected that performs a transfer on channel CHb.

Any value in the range of 0 to 7 can be set.

### 7.6.68 1A9h H_CHbFuncAdrs (Host Channel b Function Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1A9h | H_CHbFuncAdrs | R / W | 7: FuncAdrs[3] | Channel b Function Address | 00h |
| | | | | 6: FuncAdrs[2] | | |
| | | | | 5: FuncAdrs[1] | | |
| | | | | 4: FuncAdrs[0] | | |
| | | | R / W | 3: EP_Number[3] | Channel b Endpoint Number | |
| | | | | 2: EP_Number[2] | | |
| | | | | 1: EP_Number[1] | | |
| | | | | 0: EP_Number[0] | | |

This register sets the address of a function that performs a transfer on channel CHb during host operation.

**Bits7-4 FuncAdrs[3:0]**

These bits set the USB address of the function that includes the endpoint managed by channel CHb.

Any value in the range of 0 to 15 can be set.

**Bits3-0 EP_Number[3:0]**

These bits set the endpoint number that performs a transfer on channel CHb.

Any value in the range of 0 to 15 can be set.

### 7.6.69 1AAh CHbInterval_H(Channel b Interval High)

### 7.6.70 1ABh CHbInterval_L(Channel b Interval Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1AAh | H_CHbInterval_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Interval[10] | Channel b Interrupt Transfer Interval High | | |
| | | | | 1: Interval[9] | | | |
| | | | | 0: Interval[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1ABh | H_CHdInterval_L | R / W | 7: Interval[7] | Channel b Interrupt Transfer Interval Low | 00h |
| | | | | 6: Interval[6] | | |
| | | | | 5: Interval[5] | | |
| | | | | 4: Interval[4] | | |
| | | | | 3: Interval[3] | | |
| | | | | 2: Interval[2] | | |
| | | | | 1: Interval[1] | | |
| | | | | 0: Interval[0] | | |

This register sets an interval value for the interrupt transfers to be performed on channel CHb during host operation.

**1AAh.Bit7-3 Reserved**

**1AAh.Bit2-0, 1ABh.Bit7-0 Interval[10:0]**

These bits set the interval (period) at which intervals tokens for interrupt transfers will be issued. The 3 low-order bits specify the interval time in microframes (125 µs), and the 7 high-order bits specify the interval time in frames (ms). Settings of this register are effective only when the H_CHbConfig1 register's TranType bits = 11 (interrupt transfer). Setting the value "0d" in this register has no effect.

The interval time set in this register is also used when transactions are retransmitted.

Interval[2:0] µFrame — Specifies an interval time in 125 µs. Set this interval time to 1, 2, or 4 microframes. Setting any other value is prohibited. When setting these bits, make sure Interval[10:3] are all set to 0s.

Interval[10:3] Frame — Specifies an interval time in ms. This interval time can be set to any value within the range 1 to 255 frames. When setting these bits, make sure Interval[2:0] are all set to 0s.

### 7.6.71 1AEh H_CHbConditionCode (Host Channel b Condition Code)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1AEh | H_CHbConditionCode | | 7: | 0: | 1: | 00h |
| | | | R | 6: ConditionCode[2] | Channel b Condition Code | | |
| | | | | 5: ConditionCode[1] | | | |
| | | | | 4: ConditionCode[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register indicates the result of a transfer that was completed on channel CHb during host operation.

**Bit7 Reserved**

**Bits6-4 ConditionCode[2:0]**

These bits indicate the result of a transfer that was completed on channel CHb.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |

| Code | Meaning | Description |
|---|---|---|
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.72 1B0h H_CHcConfig_0(Host Channel c Configuration0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1B0h | CHcConfig_0 | R / W | 7: ACK_Cnt[3] | Channel c ACK Count | | 10h |
| | | | | 6: ACK_Cnt[2] | | | |
| | | | | 5: ACK_Cnt[1] | | | |
| | | | | 4: ACK_Cnt[0] | | | |
| | | | R / W | 3: SpeedMode[1] | Channel c Speed Mode | | |
| | | | | 2: SpeedMode[0] | | | |
| | | | R / W | 1: Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 0: TranGo | 0: Stand by | 1: Transaction Start | |

This register is used to make basic settings of channel CHc during host operation.

#### Bits7-4 ACK_Cnt [3:0]

These bits set the number of ACK count in a transfer performed on channel CHc.

When ACK count reaches the set value, the TranACK bit in the H_CHcIntStat register is set.

0000: ACK is counted up to 16 times.

0001 to 1111: ACK is counted anywhere between 1 to 15 times.

#### Bits3-2 SpeedMode [1:0]

These bits set the operation mode of the device that performs a transfer on channel CHc.

00: HS mode — Set this for HS devices.

01: FS mode — Set this for FS devices.

10: Reserved — Use of this value is prohibited.

11: LS mode — Set this for LS devices.

#### Bit1 Toggle

This bit sets the initial value of the toggle sequence bit with which a transaction is to be started. While a transaction is under execution and after a transaction is complete, this bit indicates the status of the toggle sequence bit.

0: Toggle 0

1: Toggle 1

#### Bit0 TranGo

Setting this bit to 1 causes a transaction on channel CHc to start. Transaction processing can be stopped by clearing this bit to 0 after a transaction started. This bit also serves as a status indicating whether a transaction on channel CHc is under execution or not.

0: Stops a transaction (transaction stopped)

1: Starts a transaction (transaction under execution)

When a transfer for the number of bytes set by the H_CHcTotalSize_HH through LL registers is complete, the TranCmp bit in the H-CHcIntStat register is set to 1, at which time this bit is automatically reset to 0. Furthermore, this bit is reset to 0 when the ChangeCondition bit in the H_CHcIntStat register is set. In that case, inspect the H_CHcConditionCode register to find a reason for which the condition bit was set.

If a transaction is stopped by clearing this bit, the ChangeCondition bit in the H_CHcIntStat register is set when the transaction being processed is complete. Even when a transaction is stopped, the data present in the FIFO, the (remaining) total size, and channel-related settings are left intact. Therefore, a transaction can be resumed from where it stopped by setting this bit to 1 again. (To perform a new transaction, clear the FIFO and set channel information over again.)

### 7.6.73 1B1h H_CHcConfig_1(Host Channel c Configuration1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1B1h | H_CHcConfig_1 | R / W | 7: TID[1] | Channel c Transaction ID | | 00h |
| | | | | 6: TID[0] | | | |
| | | | R / W | 5: TranType[1] | Channel c Transfer Type | | |
| | | | | 4: TranType[0] | | | |
| | | | R / W | 3: AutoZerolen | 0: Do nothing | 1: Add Zerolen | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: TotalSizeFree | 0: Do nothing | 1: Total Size Free | |

This register is used to make basic settings of channel CHc during host operation.

**Bits7-6 TID[1:0]**

These bits set the type of token (OUT or IN) that is issued on channel CHc.

00: Reserved — Use of this value is prohibited.

01: OUT — Issues an OUT token.

10: IN — Issues an IN token.

11: Reserved — Use of this value is prohibited.

**Bits5-4 TranType[1:0]**

These bits set the type of transfer that is performed on channel CHc.

00: Reserved — Use of this value is prohibited.

01: Reserved — Use of this value is prohibited.

10: Bulk — Performs a bulk transfer.

11: Interrupt — Performs an interrupt transfer.

**Bit3 AutoZerolen**

Setting this bit to 1 enables the auto zero-length function, so that when a transfer for the data bytes set by H_ChcTotalSizeHH through LL registers is completed exactly with MaxPacketSize, a zero-length packet is automatically attached at the end of the transfer. This bit is effective for only OUT transfers.

**Bits2-1 Reserved**

**Bit 0 TotalSizeFree**

Setting this bit to 1 makes the transfer size infinite regardless of the value set in the H_CHcTotalSizeHH through LL registers.

### 7.6.74 1B2h H_CHcMaxPktSize_H (Host Channel c Max Packet Size High)

### 7.6.75 1B3h H_CHcMaxPktSize_L (Host Channel c Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1B2h | H_CHcMaxPktSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: MaxPktSize[9] | Channel c Max Packet Size High | | |
| | | | | 0: MaxPktSize[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1B3h | H_CHcMaxPktSize_L | R / W | 7: MaxPktSize[7] | Channel c Max Packet Size Low | 00h |
| | | | | 6: MaxPktSize[6] | | |
| | | | | 5: MaxPktSize[5] | | |
| | | | | 4: MaxPktSize[4] | | |
| | | | | 3: MaxPktSize[3] | | |
| | | | | 2: MaxPktSize[2] | | |
| | | | | 1: MaxPktSize[1] | | |
| | | | | 0: MaxPktSize[0] | | |

This register sets MaxPacketSize of channel CHc during host operation.

#### 1B2h.Bit7-2 Reserved

Do not write 1 to the reserved bits.

#### 1B2h.Bit1-0, 1B3h.Bit7-0 MaxPktSize[9:0]

These bits set the MaxPacketSize of channel CHc.

When using this channel for bulk transfers, set one of the following transfer sizes.

During FS 8, 16, 32, or 64 bytes

During HS 512 bytes

When using this channel for interrupt transfers, set any desired transfer size within the limits given below.

During LS Up to 8 bytes

During FS Up to 64 bytes

During HS Up to 512 bytes

Setting any other transfer size is prohibited.

### 7.6.76 1B4h H_CHcTotalSize_HH (Host Channel c Total Size High-High)

### 7.6.77 1B5h H_CHcTotalSize_HL (Host Channel c Total Size High-Low)

### 7.6.78 1B6h H_CHcTotalSize_LH (Host Channel c Total Size Low-High)

### 7.6.79 1B7h H_CHcTotalSize_LL (Host Channel c Total Size Low-Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 154h | H_CHcTotalSize_HH | R / W | 7: TotalSize[31] | Channel c Total Size High-High | 00h |
| | | | | 6: TotalSize[30] | | |
| | | | | 5: TotalSize[29] | | |
| | | | | 4: TotalSize[28] | | |
| | | | | 3: TotalSize[27] | | |
| | | | | 2: TotalSize[26] | | |
| | | | | 1: TotalSize[25] | | |
| | | | | 0: TotalSize[24] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 155h | H_CHcTotalSize_HL | R / W | 7: TotalSize[23] | Channel c Total Size High-Low | 00h |
| | | | | 6: TotalSize[22] | | |
| | | | | 5: TotalSize[21] | | |
| | | | | 4: TotalSize[20] | | |
| | | | | 3: TotalSize[19] | | |
| | | | | 2: TotalSize[18] | | |
| | | | | 1: TotalSize[17] | | |
| | | | | 0: TotalSize[16] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 156h | H_CHcTotalSize_LH | R / W | 7: TotalSize[15] | Channel c Total Size Low-High | 00h |
| | | | | 6: TotalSize[14] | | |
| | | | | 5: TotalSize[13] | | |
| | | | | 4: TotalSize[12] | | |
| | | | | 3: TotalSize[11] | | |
| | | | | 2: TotalSize[10] | | |
| | | | | 1: TotalSize[9] | | |
| | | | | 0: TotalSize[8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 157h | H_CHcTotalSize_LL | R / W | 7: TotalSize[7] | Channel c Total Size Low-Low | 00h |
| | | | | 6: TotalSize[6] | | |
| | | | | 5: TotalSize[5] | | |
| | | | | 4: TotalSize[4] | | |
| | | | | 3: TotalSize[3] | | |
| | | | | 2: TotalSize[2] | | |
| | | | | 1: TotalSize[1] | | |
| | | | | 0: TotalSize[0] | | |

These registers set the Total Size of the data to be transferred on channel CHc during host operation.

#### 1B4h.Bit7-0, 1B5h.Bit7-0, 1B6h.Bit7-0, 1B7h.Bit7-0 TotalSize[31:0]

These bits set a total number of data bytes to be transferred on channel CHc (maximum 4,294,967,295 bytes: approx. 4 Gbytes).

Once a transaction is started by the TranGo bit in the H_CHcConfig_0 register, the remaining number of bytes to be transferred can be read through this register.

When bits 31 to 24 (H_CHcTotalSize_HH register) are read, the values of bits 23 to 16 (H_CHcTotalSize_HL register), bits 15 to 8 (H_CHcTotalSize_LH register), and bits 7 to 0 (H_CHcTotalSize_LL register) all are fixed. (Even when the read value is fixed, the internal counter of the LSI continues counting.)

Therefore, to read the remaining number of transfer bytes through 8-bit register accesses, read H_CHaTotalSize_HH, H_CHaTotalSize_HL, H_CHaTotalSize_LH, and H_CHaTotalSize_LL in that order.

Note also that if an OUT transaction is executed with TotalSize = 0, a zero-length packet is transmitted.

### 7.6.80 1B8h H_CHcHubAdrs (Host Channel c Hub Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1B8h | H_CHcHubAdrs | R / W | 7: HubAdrs[3] | Channel c Hub Address | | 00h |
| | | | | 6: HubAdrs[2] | | | |
| | | | | 5: HubAdrs[1] | | | |
| | | | | 4: HubAdrs[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Port[2] | Channel c Port Number | | |
| | | | | 1: Port[1] | | | |
| | | | | 0: Port[0] | | | |

This register sets the hub that is connected to channel CHc during host operation.

**Bits7-4 HubAdrs[3:0]**

These bits set the USB address of the hub to which a function is connected that performs a transfer on channel CHc.

Any value in the range of 0 to 15 can be set.

**Bit3 Reserved**

**Bits2-0 Port[2:0]**

These bits set the port number of the hub to which a function is connected that performs a transfer on channel CHc.

Any value in the range of 0 to 7 can be set.

### 7.6.81 1B9h H_CHcFuncAdrs (Host Channel c Function Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1B9h | H_CHcFuncAdrs | R / W | 7: FuncAdrs[3] | Channel c Function Address | 00h |
| | | | | 6: FuncAdrs[2] | | |
| | | | | 5: FuncAdrs[1] | | |
| | | | | 4: FuncAdrs[0] | | |
| | | | R / W | 3: EP_Number[3] | Channel c Endpoint Number | |
| | | | | 2: EP_Number[2] | | |
| | | | | 1: EP_Number[1] | | |
| | | | | 0: EP_Number[0] | | |

This register sets the address of a function that performs a transfer on channel CHc during host operation.

**Bits7-4 FuncAdrs[3:0]**

These bits set the USB address of the function that includes the endpoint managed by channel CHc.

Any value in the range of 0 to 15 can be set.

**Bits3-0 EP_Number[3:0]**

These bits set the endpoint number that performs a transfer on channel CHc.

Any value in the range of 0 to 15 can be set.

### 7.6.82 1BAh H_CHcInterval_H(Host Channel c Interval High)

### 7.6.83 1BBh H_CHcInterval_L(Host Channel c Interval Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1BAh | H_CHcInterval_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Interval[10] | Channel c Interrupt Transfer Interval High | | |
| | | | | 1: Interval[9] | | | |
| | | | | 0: Interval[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1BBh | H_CHcInterval_L | R / W | 7: Interval[7] | Channel c Interrupt Transfer Interval Low | 00h |
| | | | | 6: Interval[6] | | |
| | | | | 5: Interval[5] | | |
| | | | | 4: Interval[4] | | |
| | | | | 3: Interval[3] | | |
| | | | | 2: Interval[2] | | |
| | | | | 1: Interval[1] | | |
| | | | | 0: Interval[0] | | |

This register sets an interval value for the interrupt transfers to be performed on channel CHc during host operation.

**1BAh.Bit7-3 Reserved**

**1BAh.Bit2-0, 1BBh.Bit7-0 Interval[10:0]**

These bits set the interval (period) at which intervals tokens for interrupt transfers will be issued. The 3 low-order bits specify the interval time in microframes (125 µs), and the 7 high-order bits specify the interval time in frames (ms). Settings of this register are effective only when the H_CHcConfig1 register's TranType bits = 11 (interrupt transfer). Setting the value "0d" in this register has no effect.

The interval time set in this register is also used when transactions are retransmitted.

Interval[2:0] µFrame — Specifies an interval time in 125 µs units. Set this interval time to 1, 2, or 4 microframes. Setting any other value is prohibited. When setting these bits, make sure Interval[10:3] are all set to 0s.

Interval[10:3] Frame — Specifies an interval time in ms units. This interval time can be set to any value within the range 1 to 255 frames. When setting these bits, make sure Interval[2:0] are all set to 0s.

### 7.6.84 1BEh H_CHcConditionCode (Host Channel c Condition Code)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1BEh | H_CHcConditionCode | | 7: | 0: | 1: | 00h |
| | | | R | 6: ConditionCode[2] | Channel c Condition Code | | |
| | | | | 5: ConditionCode[1] | | | |
| | | | | 4: ConditionCode[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register indicates the result of a transfer that was completed on channel CHc during host operation.

**Bit7 Reserved**

**Bits6-4 ConditionCode[2:0]**

These bits indicate the result of a transfer that was completed on channel CHc.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |

| Code | Meaning | Description |
|---|---|---|
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.85 1C0h H_CHdConfig_0(Host Channel d Configuration0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1C0h | CHdConfig_0 | R / W | 7: ACK_Cnt[3] | Channel d ACK Count | | 00h |
| | | | | 6: ACK_Cnt[2] | | | |
| | | | | 5: ACK_Cnt[1] | | | |
| | | | | 4: ACK_Cnt[0] | | | |
| | | | R / W | 3: SpeedMode[1] | Channel d Speed Mode | | |
| | | | | 2: SpeedMode[0] | | | |
| | | | R / W | 1: Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 0: TranGo | 0: Stand by | 1: Transaction Start | |

This register is used to make basic settings of channel CHd during host operation.

#### Bits7-4 ACK_Cnt [3:0]

These bits set the number of ACK count in a transfer performed on channel CHd.

When ACK count reaches the set value, the TranACK bit in the H_CHdIntStat register is set.

0000: ACK is counted up to 16 times.

0001 to 1111: ACK is counted anywhere between 1 to 15 times.

#### Bits3-2 SpeedMode [1:0]

These bits set the operation mode of the device that performs a transfer on channel CHd.

00: HS mode — Set this for HS devices.

01: FS mode — Set this for FS devices.

10: Reserved — Use of this value is prohibited.

11: LS mode — Set this for LS devices.

#### Bit1 Toggle

This bit sets the initial value of the toggle sequence bit with which a transaction is to be started. While a transaction is under execution and after a transaction is complete, this bit indicates the status of the toggle sequence bit.

0: Toggle 0

1: Toggle 1

#### Bit0 TranGo

Setting this bit to 1 causes a transaction on channel CHd to start. Transaction processing can be stopped by clearing this bit to 0 after a transaction started. This bit also serves as a status indicating whether a transaction on channel CHd is under execution or not.

0: Stops a transaction (transaction stopped)

1: Starts a transaction (transaction under execution)

When a transfer for the number of bytes set by the H_CHdTotalSize_HH through LL registers is complete, the TranCmp bit in the H-CHdIntStat register is set to 1, at which time this bit is automatically reset to 0. Furthermore, this bit is reset to 0 when the ChangeCondition bit in the H_CHdIntStat register is set. In that case, inspect the H_CHdConditionCode register to find a reason for which the condition bit was set.

If a transaction is stopped by clearing this bit, the ChangeCondition bit in the H_CHdIntStat register is set when the transaction being processed is complete. Even when a transaction is stopped, the data present in the FIFO, the (remaining) total size, and channel-related settings are left intact. Therefore, a transaction can be resumed from where it stopped by setting this bit to 1 again. (To perform a new transaction, clear the FIFO and set channel information over again.)

### 7.6.86 1C1h H_CHdConfig_1(Host Channel d Configuration1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1C1h | H_CHdConfig_1 | R / W | 7: TID[1] | Channel d Transaction ID | | 00h |
| | | | | 6: TID[0] | | | |
| | | | R / W | 5: TranType[1] | Channel d Transfer Type | | |
| | | | | 4: TranType[0] | | | |
| | | | R / W | 3: AutoZerolen | 0: Do nothing | 1: Add Zerolen | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: TotalSizeFree | 0: Do nothing | 1: Total Size Free | |

This register is used to make basic settings of channel CHd during host operation.

**Bits7-6 TID[1:0]**

These bits set the type of token (OUT or IN) that is issued on channel CHd.

00: Reserved — Use of this value is prohibited.

01: OUT — Issues an OUT token.

10: IN — Issues an IN token.

11: Reserved — Use of this value is prohibited.

**Bits5-4 TranType[1:0]**

These bits set the type of transfer that is performed on channel CHd.

00: Reserved — Use of this value is prohibited.

01: Reserved — Use of this value is prohibited.

10: Bulk — Performs a bulk transfer.

11: Interrupt — Performs an interrupt transfer.

**Bit3 AutoZerolen**

Setting this bit to 1 enables the auto zero-length function, so that when a transfer for the data bytes set by H_ChdTotalSizeHH through LL registers is completed exactly with MaxPacketSize, a zero-length packet is automatically attached at the end of the transfer. This bit is effective for only OUT transfers.

**Bits2-1 Reserved**

**Bit0 TotalSizeFree**

Setting this bit to 1 makes the transfer size infinite regardless of the value set in the H_ChdTotalSizeHH through LL registers.

### 7.6.87 1C2h H_CHdMaxPktSize_H (Host Channel d Max Packet Size High)

### 7.6.88 1C3h H_CHdMaxPktSize_L (Host Channel d Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1C2h | H_CHdMaxPktSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: MaxPktSize[9] | Channel d Max Packet Size High | | |
| | | | | 0: MaxPktSize[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1C3h | H_CHdMaxPktSize_L | R / W | 7: MaxPktSize[7] | Channel d Max Packet Size Low | 00h |
| | | | | 6: MaxPktSize[6] | | |
| | | | | 5: MaxPktSize[5] | | |
| | | | | 4: MaxPktSize[4] | | |
| | | | | 3: MaxPktSize[3] | | |
| | | | | 2: MaxPktSize[2] | | |
| | | | | 1: MaxPktSize[1] | | |
| | | | | 0: MaxPktSize[0] | | |

This register sets MaxPacketSize of channel CHd during host operation.

**1C2h.Bit7-2 Reserved**

Do not write 1 to the reserved bits.

**1C2h.Bit1-0, 1C3h.Bit7-0 MaxPktSize[9:0]**

These bits set the MaxPacketSize of channel CHd.

When using this channel for bulk transfers, set one of the following transfer sizes.

During FS 8, 16, 32, or 64 bytes

During HS 512 bytes

When using this channel for interrupt transfers, set any desired transfer size within the limits given below.

During LS Up to 8 bytes

During FS Up to 64 bytes

During HS Up to 512 bytes

Setting any other transfer size is prohibited.

### 7.6.89 1C4h H_CHdTotalSize_HH (Host Channel d Total Size High-High)

### 7.6.90 1C5h H_CHdTotalSize_HL (Host Channel d Total Size High-Low)

### 7.6.91 1C6h H_CHdTotalSize_LH (Host Channel d Total Size Low-High)

### 7.6.92 1C7h H_CHdTotalSize_LL (Host Channel d Total Size Low-Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1C4h | H_CHdTotalSize_HH | R / W | 7: TotalSize[31] | Channel d Total Size High-High | 00h |
| | | | | 6: TotalSize[30] | | |
| | | | | 5: TotalSize[29] | | |
| | | | | 4: TotalSize[28] | | |
| | | | | 3: TotalSize[27] | | |
| | | | | 2: TotalSize[26] | | |
| | | | | 1: TotalSize[25] | | |
| | | | | 0: TotalSize[24] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1C5h | H_CHdTotalSize_HL | R / W | 7: TotalSize[23] | Channel d Total Size High-Low | 00h |
| | | | | 6: TotalSize[22] | | |
| | | | | 5: TotalSize[21] | | |
| | | | | 4: TotalSize[20] | | |
| | | | | 3: TotalSize[19] | | |
| | | | | 2: TotalSize[18] | | |
| | | | | 1: TotalSize[17] | | |
| | | | | 0: TotalSize[16] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1C6h | H_CHdTotalSize_LH | R / W | 7: TotalSize[15] | Channel d Total Size Low-High | 00h |
| | | | | 6: TotalSize[14] | | |
| | | | | 5: TotalSize[13] | | |
| | | | | 4: TotalSize[12] | | |
| | | | | 3: TotalSize[11] | | |
| | | | | 2: TotalSize[10] | | |
| | | | | 1: TotalSize[9] | | |
| | | | | 0: TotalSize[8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1C7h | H_CHdTotalSize_LL | R / W | 7: TotalSize[7] | Channel d Total Size Low-Low | 00h |
| | | | | 6: TotalSize[6] | | |
| | | | | 5: TotalSize[5] | | |
| | | | | 4: TotalSize[4] | | |
| | | | | 3: TotalSize[3] | | |
| | | | | 2: TotalSize[2] | | |
| | | | | 1: TotalSize[1] | | |
| | | | | 0: TotalSize[0] | | |

These registers set the Total Size of the data to be transferred on channel CHd during host operation.

**1C4h.Bit7-0, 1C5h.Bit7-0, 1C6h.Bit7-0, 1C7h.Bit7-0 TotalSize[31:0]**

These bits set a total number of data bytes to be transferred on channel CHd (maximum 4,294,967,295 bytes: approx. 4 Gbytes).

Once a transaction is started by the TranGo bit in the H_CHdConfig_0 register, the remaining number of bytes to be transferred can be read through this register.

When bits 31 to 24 (H_CHdTotalSize_HH register) are read, the values of bits 23 to 16 (H_CHdTotalSize_HL register), bits 15 to 8 (H_CHdTotalSize_LH register), and bits 7 to 0 (H_CHdTotalSize_LL register) all are fixed. (Even when the read value is fixed, the internal counter of the LSI continues counting.)

Therefore, to read the remaining number of transfer bytes through 8-bit register accesses, read H_CHaTotalSize_HH, H_CHaTotalSize_HL, H_CHaTotalSize_LH, and H_CHaTotalSize_LL in that order.

Note also that if an OUT transaction is executed with TotalSize = 0, a zero-length packet is transmitted.

### 7.6.93 1C8h H_CHdHubAdrs (Host Channel d Hub Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1C8h | H_CHdHubAdrs | R / W | 7: HubAdrs[3] | Channel d Hub Address | | 00h |
| | | | | 6: HubAdrs[2] | | | |
| | | | | 5: HubAdrs[1] | | | |
| | | | | 4: HubAdrs[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Port[2] | Channel d Port Number | | |
| | | | | 1: Port[1] | | | |
| | | | | 0: Port[0] | | | |

This register sets the hub that is connected to channel CHd during host operation.

**Bits7-4 HubAdrs[3:0]**

These bits set the USB address of the hub to which a function is connected that performs a transfer on channel CHd.

Any value in the range of 0 to 15 can be set.

**Bit3 Reserved**

**Bits2-0 Port[2:0]**

These bits set the port number of the hub to which a function is connected that performs a transfer on channel CHd.

Any value in the range of 0 to 7 can be set.

### 7.6.94 1C9h H_CHdFuncAdrs (Host Channel d Function Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1C9h | H_CHdFuncAdrs | R / W | 7: FuncAdrs[3] | Channel d Function Address | 00h |
| | | | | 6: FuncAdrs[2] | | |
| | | | | 5: FuncAdrs[1] | | |
| | | | | 4: FuncAdrs[0] | | |
| | | | R / W | 3: EP_Number[3] | Channel d Endpoint Number | |
| | | | | 2: EP_Number[2] | | |
| | | | | 1: EP_Number[1] | | |
| | | | | 0: EP_Number[0] | | |

This register sets the address of a function that performs a transfer on channel CHd during host operation.

**Bits7-4 FuncAdrs[3:0]**

These bits set the USB address of the function that includes the endpoint managed by channel CHd.

Any value in the range of 0 to 15 can be set.

**Bits3-0 EP_Number[3:0]**

These bits set the endpoint number that performs a transfer on channel CHd.

Any value in the range of 0 to 15 can be set.

### 7.6.95 1CAh H_CHdInterval_H(Host Channel d Interval High)

### 7.6.96 1CBh H_CHdInterval_L(Host Channel d Interval Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1CAh | H_CHdInterval_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Interval[10] | Channel d Interrupt Transfer Interval High | | |
| | | | | 1: Interval[9] | | | |
| | | | | 0: Interval[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1CBh | H_CHdInterval_L | R / W | 7: Interval[7] | Channel d Interrupt Transfer Interval Low | 00h |
| | | | | 6: Interval[6] | | |
| | | | | 5: Interval[5] | | |
| | | | | 4: Interval[4] | | |
| | | | | 3: Interval[3] | | |
| | | | | 2: Interval[2] | | |
| | | | | 1: Interval[1] | | |
| | | | | 0: Interval[0] | | |

This register sets an interval value for the interrupt transfers to be performed on channel CHd during host operation.

**1CAh.Bit7-3 Reserved**

**1CAh.Bit2-0, 1CBh.Bit7-0 Interval[10:0]**

These bits set the interval (period) at which intervals tokens for interrupt transfers will be issued. The 3 low-order bits specify the interval time in microframes (125 µs), and the 7 high-order bits specify the interval time in frames (ms). Settings of this register are effective only when the H_CHdConfig1 register's TranType bits = 11 (interrupt transfer). Setting the value "0d" in this register has no effect.

The interval time set in this register is also used when transactions are retransmitted.

Interval[2:0] µFrame — Specifies an interval time in 125 µs. Set this interval time to 1, 2, or 4 microframes. Setting any other value is prohibited. When setting these bits, make sure Interval[10:3] are all set to 0s.

Interval[10:3] Frame — Specifies an interval time in ms units. This interval time can be set to any value within the range of 1 to 255 frames. When setting these bits, make sure Interval[2:0] are all set to 0s.

### 7.6.97 1CEh H_CHdConditionCode (Host Channel d Condition Code)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1CEh | H_CHdConditionCode | | 7: | 0: | 1: | 00h |
| | | | R | 6: ConditionCode[2] | Channel d Condition Code | | |
| | | | | 5: ConditionCode[1] | | | |
| | | | | 4: ConditionCode[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register indicates the result of a transfer that was completed on channel CHd during host operation.

**Bit7 Reserved**

**Bit6-4 ConditionCode[2:0]**

These bits indicate the result of a transfer that was completed on channel CHd.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |

| Code | Meaning | Description |
|---|---|---|
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

### 7.6.98 1D0h H_CHeConfig_0(Host Channel e Configuration0)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1D0h | CHeConfig_0 | R / W | 7: ACK_Cnt[3] | Channel e ACK Count | | 00h |
| | | | | 6: ACK_Cnt[2] | | | |
| | | | | 5: ACK_Cnt[1] | | | |
| | | | | 4: ACK_Cnt[0] | | | |
| | | | R / W | 3: SpeedMode[1] | Channel e Speed Mode | | |
| | | | | 2: SpeedMode[0] | | | |
| | | | R / W | 1: Toggle | 0: Toggle0 | 1: Toggle1 | |
| | | | R / W | 0: TranGo | 0: Stand by | 1: Transaction Start | |

This register is used to make basic settings of channel CHe during host operation.

**Bits7-4 ACK_Cnt [3:0]**

These bits set the number of ACK count in a transfer performed on channel CHe.

When ACK count reaches the set value, the TranACK bit in the H_CHeIntStat register is set.

0000: ACK is counted up to 16 times.

0001 to 1111: ACK is counted anywhere between 1 to 15 times.

**Bits3-2 SpeedMode [1:0]**

These bits set the operation mode of the device that performs a transfer on channel CHe.

00: HS mode — Set this for HS devices.

01: FS mode — Set this for FS devices.

10: Reserved — Use of this value is prohibited.

11: LS mode — Set this for LS devices.

**Bit1 Toggle**

This bit sets the initial value of the toggle sequence bit with which a transaction is to be started. While a transaction is under execution and after a transaction is complete, this bit indicates the status of the toggle sequence bit.

0: Toggle 0

1: Toggle 1

**Bit0 TranGo**

Setting this bit to 1 causes a transaction on channel CHe to start. Transaction processing can be stopped by clearing this bit to 0 after a transaction started. This bit also serves as a status indicating whether a transaction on channel CHe is under execution or not.

0: Stops a transaction (transaction stopped)

1: Starts a transaction (transaction under execution)

When a transfer for the number of bytes set by the H_CHeTotalSize_HH through LL registers is complete, the TranCmp bit in the H-CHeIntStat register is set to 1, at which time this bit is automatically reset to 0. Furthermore, this bit is reset to 0 when the ChangeCondition bit in the H_CHeIntStat register is set. In that case, inspect the H_CHeConditionCode register to find a reason for which the condition bit was set.

If a transaction is stopped by clearing this bit, the ChangeCondition bit in the H_CHeIntStat register is set when the transaction being processed is complete. Even when a transaction is stopped, the data present in the FIFO, the (remaining) total size, and channel-related settings are left intact. Therefore, a transaction can be resumed from where it is stopped by setting this bit to 1 again. (To perform a new transaction, clear the FIFO and set channel information over again.)

### 7.6.99 1D1h H_CHeConfig_1(Host Channel e Configuration1)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1D1h | CHeConfig_1 | R / W | 7: TID[1] | Channel e Transaction ID | | 00h |
| | | | | 6: TID[0] | | | |
| | | | R / W | 5: TranType[1] | Channel e Transfer Type | | |
| | | | | 4: TranType[0] | | | |
| | | | R / W | 3: AutoZerolen | 0: Do nothing | 1: Add Zerolen | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | R / W | 0: TotalSizeFree | 0: Do nothing | 1: Total Size Free | |

This register is used to make basic settings of channel CHe during host operation.

**Bits7-6 TID[1:0]**

These bits set the type of token (OUT or IN) that is issued on channel CHe.

00: Reserved — Use of this value is prohibited.

01: OUT — Issues an OUT token.

10: IN — Issues an IN token.

11: Reserved — Use of this value is prohibited.

**Bits5-4 TranType[1:0]**

These bits set the type of transfer that is performed on channel CHe.

00: Reserved — Use of this value is prohibited.

01: Reserved — Use of this value is prohibited.

10: Bulk — Performs a bulk transfer.

11: Interrupt — Performs an interrupt transfer.

**Bit3 AutoZerolen**

Setting this bit to 1 enables the auto zero-length function, so that when a transfer for the data bytes set by H_CheTotalSizeHH through LL registers is completed exactly with MaxPacketSize, a zero-length packet is automatically attached at the end of the transfer. This bit is effective for only OUT transfers.

**Bits2-1 Reserved**

**Bit0 TotalSizeFree**

Setting this bit to 1 makes the transfer size infinite regardless of the value set in the H_CheTotalSizeHH through LL registers.

### 7.6.100 1D2h H_CHeMaxPktSize_H (Host Channel e Max Packet Size High)

### 7.6.101 1D3h H_CHeMaxPktSize_L (Host Channel e Max Packet Size Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1D2h | H_CHeMaxPktSize_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: MaxPktSize[9] | Channel e Max Packet Size High | | |
| | | | | 0: MaxPktSize[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1D3h | H_CHeMaxPktSize_L | R / W | 7: MaxPktSize[7] | Channel e Max Packet Size Low | 00h |
| | | | | 6: MaxPktSize[6] | | |
| | | | | 5: MaxPktSize[5] | | |
| | | | | 4: MaxPktSize[4] | | |
| | | | | 3: MaxPktSize[3] | | |
| | | | | 2: MaxPktSize[2] | | |
| | | | | 1: MaxPktSize[1] | | |
| | | | | 0: MaxPktSize[0] | | |

This register sets MaxPacketSize of channel CHe during host operation.

**1D2h.Bit7-2 Reserved**

Do not write 1 to the reserved bits.

**1D2h.Bit1-0, 1D3h.Bit7-0 MaxPktSize[9:0]**

These bits set the MaxPacketSize of channel CHe.

When using this channel for bulk transfers, set one of the following transfer sizes.

During FS 8, 16, 32, or 64 bytes

During HS 512 bytes

When using this channel for interrupt transfers, set any desired transfer size within the limits given below.

During LS Up to 8 bytes

During FS Up to 64 bytes

During HS Up to 512 bytes

Setting any other transfer size is prohibited.

### 7.6.102 1D4h H_CHeTotalSize_HH (Host Channel e Total Size High-High)

### 7.6.103 1D5h H_CHeTotalSize_HL (Host Channel e Total Size High-Low)

### 7.6.104 1D6h H_CHeTotalSize_LH (Host Channel e Total Size Low-High)

### 7.6.105 1D7h H_CHeTotalSize_LL (Host Channel e Total Size Low-Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1D4h | H_CHeTotalSize_HH | R / W | 7: TotalSize[31] | Channel e Total Size High-High | 00h |
| | | | | 6: TotalSize[30] | | |
| | | | | 5: TotalSize[29] | | |
| | | | | 4: TotalSize[28] | | |
| | | | | 3: TotalSize[27] | | |
| | | | | 2: TotalSize[26] | | |
| | | | | 1: TotalSize[25] | | |
| | | | | 0: TotalSize[24] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1D5h | H_CHeTotalSize_HL | R / W | 7: TotalSize[23] | Channel e Total Size High-Low | 00h |
| | | | | 6: TotalSize[22] | | |
| | | | | 5: TotalSize[21] | | |
| | | | | 4: TotalSize[20] | | |
| | | | | 3: TotalSize[19] | | |
| | | | | 2: TotalSize[18] | | |
| | | | | 1: TotalSize[17] | | |
| | | | | 0: TotalSize[16] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1D6h | H_CHeTotalSize_LH | R / W | 7: TotalSize[15] | Channel e Total Size Low-High | 00h |
| | | | | 6: TotalSize[14] | | |
| | | | | 5: TotalSize[13] | | |
| | | | | 4: TotalSize[12] | | |
| | | | | 3: TotalSize[11] | | |
| | | | | 2: TotalSize[10] | | |
| | | | | 1: TotalSize[9] | | |
| | | | | 0: TotalSize[8] | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1D7h | H_CHeTotalSize_LL | R / W | 7: TotalSize[7] | Channel e Total Size Low-Low | 00h |
| | | | | 6: TotalSize[6] | | |
| | | | | 5: TotalSize[5] | | |
| | | | | 4: TotalSize[4] | | |
| | | | | 3: TotalSize[3] | | |
| | | | | 2: TotalSize[2] | | |
| | | | | 1: TotalSize[1] | | |
| | | | | 0: TotalSize[0] | | |

These registers set the Total Size of the data to be transferred on channel CHe during host operation.

#### 1D4h.Bit7-0, 1D5h.Bit7-0, 1D6h.Bit7-0, 1D7h.Bit7-0 TotalSize[31:0]

These bits set a total number of data bytes to be transferred on channel CHe (maximum 4,294,967,295 bytes: approx. 4 Gbytes).

Once a transaction is started by the TranGo bit in the H_CHeConfig_0 register, the remaining number of bytes to be transferred can be read through this register.

When bits 31 to 24 (H_CHeTotalSize_HH register) are read, the values of bits 23 to 16 (H_CHeTotalSize_HL register), bits 15 to 8 (H_CHeTotalSize_LH register), and bits 7 to 0 (H_CHeTotalSize_LL register) all are fixed. (Even when the read value is fixed, the internal counter of the LSI continues counting.)

Therefore, to read the remaining number of transfer bytes through 8-bit register accesses, read H_CHaTotalSize_HH, H_CHaTotalSize_HL, H_CHaTotalSize_LH, and H_CHaTotalSize_LL in that order.

Note also that if an OUT transaction is executed with TotalSize = 0, a zero-length packet is transmitted.

### 7.6.106 1D8h H_CHeHubAdrs (Host Channel e Hub Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1D8h | H_CHeHubAdrs | R / W | 7: HubAdrs[3] | Channel e Hub Address | | 00h |
| | | | | 6: HubAdrs[2] | | | |
| | | | | 5: HubAdrs[1] | | | |
| | | | | 4: HubAdrs[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Port[2] | Channel e Port Number | | |
| | | | | 1: Port[1] | | | |
| | | | | 0: Port[0] | | | |

This register sets the hub that is connected to channel CHe during host operation.

**Bits7-4 HubAdrs[3:0]**

This bit sets the USB address of the hub to which a function is connected that performs a transfer on channel CHe.

Any value in the range of 0 to 15 can be set.

**Bit3 Reserved**

**Bits2-0 Port[2:0]**

These bits set the port number of the hub to which a function is connected that performs a transfer on channel CHe.

Any value in the range of 0 to 7 can be set.

### 7.6.107 1D9h H_CHeFuncAdrs (Host Channel e Function Address)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1D9h | H_CHeFuncAdrs | R / W | 7: FuncAdrs[3] | Channel e Function Address | 00h |
| | | | | 6: FuncAdrs[2] | | |
| | | | | 5: FuncAdrs[1] | | |
| | | | | 4: FuncAdrs[0] | | |
| | | | R / W | 3: EP_Number[3] | Channel e Endpoint Number | |
| | | | | 2: EP_Number[2] | | |
| | | | | 1: EP_Number[1] | | |
| | | | | 0: EP_Number[0] | | |

This register sets the address of a function that performs a transfer on channel CHe during host operation.

**Bits7-4 FuncAdrs[3:0]**

These bits set the USB address of the function that includes the endpoint managed by channel CHe.

Any value in the range of 0 to 15 can be set.

**Bits3-0 EP_Number[3:0]**

This bit sets the endpoint number that performs a transfer on channel CHe.

Any value in the range of 0 to 15 can be set.

### 7.6.108 1DAh H_CHeInterval_H(Host Channel e Interval High)

### 7.6.109 1DBh H_CHeInterval_L(Host Channel e Interval Low)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1DAh | H_CHeInterval_H | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | | 3: | 0: | 1: | |
| | | | R / W | 2: Interval[10] | Channel e Interrupt Transfer Interval High | | |
| | | | | 1: Interval[9] | | | |
| | | | | 0: Interval[8] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | Reset |
|---|---|---|---|---|---|---|
| Host | 1DBh | H_CHeInterval_L | R / W | 7: Interval[7] | Channel e Interrupt Transfer Interval Low | 00h |
| | | | | 6: Interval[6] | | |
| | | | | 5: Interval[5] | | |
| | | | | 4: Interval[4] | | |
| | | | | 3: Interval[3] | | |
| | | | | 2: Interval[2] | | |
| | | | | 1: Interval[1] | | |
| | | | | 0: Interval[0] | | |

This register sets an interval value for the interrupt transfers to be performed on channel CHe during host operation.

**1DAh.Bit7-3 Reserved**

**1DAh.Bit2-0, 1DBh.Bit7-0 Interval[10:0]**

These bits set the interval (period) at which intervals tokens for interrupt transfers will be issued. The 3 low-order bits specify the interval time in microframes (125 µs), and the 7 high-order bits specify the interval time in frames (ms). Settings of this register are effective only when the H_CHeConfig1 register's TranType bits = 11 (interrupt transfer). Setting the value "0d" in this register has no effect.

The interval time set in this register is also used when transactions are retransmitted.

Interval[2:0] µFrame — Specifies an interval time in 125 µs units. Set this interval time to 1, 2, or 4 microframes. Setting any other value is prohibited. When setting these bits, make sure Interval[10:3] are all set to 0s.

Interval[10:3] Frame — Specifies an interval time in ms units. This interval time can be set to any value within the range 1 to 255 frames. When setting these bits, make sure Interval[2:0] are all set to 0s.

### 7.6.110 1DEh H_CHeConditionCode (Host Channel e Condition Code)

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1DEh | H_CHeConditionCode | | 7: | 0: | 1: | 00h |
| | | | R | 6: ConditionCode[2] | Channel e Condition Code | | |
| | | | | 5: ConditionCode[1] | | | |
| | | | | 4: ConditionCode[0] | | | |
| | | | | 3: | 0: | 1: | |
| | | | | 2: | 0: | 1: | |
| | | | | 1: | 0: | 1: | |
| | | | | 0: | 0: | 1: | |

This register indicates the result of a transfer that was completed on channel CHe during host operation.

**Bit7 Reserved**

**Bits6-4 ConditionCode[2:0]**

These bits indicate the result of a transfer that was completed on channel CHe.

| Code | Meaning | Description |
|---|---|---|
| 000 | NOERROR | The transaction is completed without error. |
| 001 | STALL | The endpoint has returned stall PID. |
| 010 | DATAOVERRUN | • A data packet exceeding MaxPacketSize was received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>• Bytes of data exceeding IRP (TotalSize) were received.<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error.<br>* If the received data packet is equal to or less than MaxPacketSize and the data toggle included in it does not match the expected value, this is processed as a toggle mismatch, and not as a data overrun. |
| 011 | DATAUNDERRUN | • A data packet smaller than MaxPacketSize was received and the bytes of data in it were less than IRP (TotalSize).<br>* If a CRC error or bit stuffing error was detected at the same time, this is processed as a retry error. |

| Code | Meaning | Description |
|---|---|---|
| 100 | RETRYERROR | • The device does not respond to a token within a designated time (IN) or does not transmit a handshaking signal within a designated time (OUT).<br>• The data packet from the endpoint contains a CRC error.<br>• The data packet from the endpoint contains a bit stuffing error.<br>• The PID inspection bit from the endpoint failed in data PID (IN) or handshaking (OUT).<br>• The received PID is invalid or has no PID values defined.<br>• An ERR handshaking signal was received in a split transaction of interrupt transfer.<br>• Three NYET handshaking signals were received consecutively in a split transaction of interrupt transfer.<br>• The data toggle included in the data packet from the endpoint does not match the expected value (toggle mismatch). |
| Other | Reserved | |

**Bits3-0 Reserved**

# 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 |
| Input Voltage | HVI | VSS-0.3 to HVDD+0.5 | V |
| | CVI *1 | VSS-0.3 to CVDD+0.5 | V |
| | LVI *2 | VSS-0.3 to LVDD+0.5 | V |
| | VVI *3 | VSS-0.3 to 6.0 | V |
| Output Voltage | HVO | VSS-0.3 to HVDD+0.5 | V |
| | CVO *1 | VSS-0.3 to CVDD+0.5 | V |
| Output Current per Pin | IOUT | ±10 | mA |
| Storage Temperature | Tstg | -65 to 150 | °C |

*1: CPU I/F

*2: TESTEN, ATPGEN, BURNIN, X1

*3: VBUS

## 8.2 Recommended Operating Conditions

| Parameter | Symbol | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|
| Supply Voltage | HVDD | 3.00 | 3.30 | 3.60 | V |
| | CVDD | 1.65 | 1.8 to 3.30 | 3.60 | V |
| | LVDD | 1.65 | 1.80 | 1.95 | V |
| Input Voltage | HVI | -0.3 | - | HVDD+0.3 | V |
| | CVI *1 | -0.3 | - | CVDD+0.3 | V |
| | LVI *2 | -0.3 | - | LVDD+0.3 | V |
| Ambient Temperature | $T_a$ | -40 | 25 | 85 | °C |

*1: CPU/IF

*2: TESTEN, ATPGEN, BURNIN, X1

Engage power following the procedure below:

**LVDD (int.) → HVDD,CVDD (I/O)**

Power to the IC must be turned off in the following sequence:

**HVDD, CVDD (IO unit) -> LVDD (internal)**

NOTE: Due to the chip's limited reliability, do not apply HVDD,CVDD continuously (for over 1 sec) while the LVDD is disconnected.

## 8.3 D.C. Characteristics

Input Characteristics in the D.C. State (under Recommended Operating Conditions)

| Parameter | Symbol | Condition | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|---|
| Power Supply Current | | | | | | |
| | IDDH | HVDD=3.3V *1 | - | 7.8 | - | mA |
| | IDDCH | CVDD=3.3V *1 | - | 1.4 | - | mA |
| | IDDCL | CVDD=1.8V *1 | - | 0.7 | - | mA |
| | IDDL | LVDD=1.8V * 1 | - | 39.3 | - | mA |
| Quiescent Current*2 | | | | | | |
| Supply Current | IDDS | VIN = HVDD, CVDD, LVDD or VSS<br>HVDD=3.6V<br>CVDD=3.6V<br>LVDD=1.95V | - | - | 20 | µA |
| Input Leakage | | | | | | |
| Input Leakage Current | IL | HVDD=3.6V<br>CVDD=3.6V<br>LVDD=1.95V<br>HVIH=HVDD<br>CVIH=CVDD<br>LVIH=LVDD<br>VIL=VSS | -5 | - | 5 | µA |

*1: Approximate current values when operating under the recommended operating conditions (Ta = 25°C).

*2: Quiescent current for the case in which Ta = 25°C and the bidirectional pins are set for input.

Under our test environment, the power consumption measurement value in each power management state (Ta = 25°C)

| Parameter | Condition | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|
| CPU Cut (At the CPU bus has activities *1) *2 | | | | | |
| Power Comsumption | HVDD=3.3V<br>CVDD=1.8V<br>LVDD=1.8V | - | 3.9 | - | µW |
| SLEEP (At the CPU bus has activities *1) *2 | | | | | |
| Power Comsumption | HVDD=3.3V<br>CVDD=1.8V<br>LVDD=1.8V | - | 10.0 | - | µW |
| ACTIVE (At it acts as USB device) *3 | | | | | |
| Power Comsumption | HVDD=3.3V<br>CVDD=1.8V<br>LVDD=1.8V | - | 92.2 | - | mW |
| ACTIVE (At it acts as USB host) *4 | | | | | |
| Power Comsumption | HVDD=3.3V<br>CVDD=1.8V<br>LVDD=1.8V | - | 97.6 | - | mW |

*1: The condition where the CPU is accessing to the memory (the SRAM and ROM etc.) that is connected on the CPU bus.

*2: At it acts as USB device, it is excepted the current consumption value (about 200uA) by VBUS at the DP pull-up resistance that S1R72V17 is containing.

*3: It connects to the PC as USB device, and it is condition where is transfering data (real transfer rate 13.5MB/s).

*4: It connects to the HDD as USB host, and it is condition where is transfering data (real transfer rate 12.6MB/s).

Input Characteristics in the D.C. State (under Recommended Operating Conditions) Cont'd from the preceding page

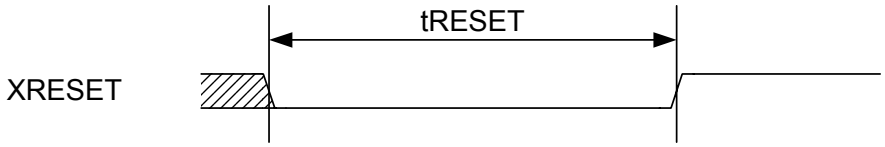
| Parameter | Symbol | Condition | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|---|
| Input Characteristics (LVCMOS) | Pin name: | TESTEN, ATPGEN, BURNIN | | | | |
| High Level Input Voltage | VIH1 | HVDD = 1.95V | 1.27 | - | - | V |
| Low Level Input Voltage | VIL1 | HVDD = 1.65V | - | - | 0.57 | V |
| Input Characteristics (LVCMOS) | Pin name: | CA[8:1], CD[15:0], XRD, XWRL, XWRH, XBEL, XDACK, CLKIN | | | | |
| High Level Input Voltage | VIH2 | CVDD=3.6V | 2.2 | - | - | V |
| Low Level Input Voltage | VIL2 | CVDD=3.0 | - | - | 0.8 | V |
| High Level Input Voltage | VIH3 | CVDD=1.95V | 1.27 | - | - | V |
| Low Level Input Voltage | VIL3 | CVDD=1.65V | - | - | 0.57 | 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 and DM in pairs | | | | |
| Sensitivity of Differential Input | VDS (USB) | HVDD = 3.0V<br>Differential input voltage<br>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 |
| Input Characteristics (Schmitt) | Pin name: | VBUSFLG | | | | |
| High Level Trigger Voltage | VT1+ | HVDD = 3.6V | 1.4 | - | 2.7 | V |
| Low Level Trigger Voltage | VT1- | HVDD = 3.0V | 0.6 | - | 1.8 | V |
| Hysteresis Voltage | ΔV | HVDD= 3.0V | 0.3 | - | - | V |
| Input Characteristics (Schmitt) | Pin name: | XCS, XRESET | | | | |
| High Level Trigger Voltage | VT1+ | HVDD = 3.6V | 1.4 | - | 2.7 | V |
| Low Level Trigger Voltage | VT1- | HVDD = 3.0V | 0.6 | - | 1.8 | V |
| Hysteresis Voltage | ΔV1 | HVDD= 3.0V | 0.3 | - | - | V |
| High Level Trigger Voltage | VT2+ | CVDD = 1.95V | 0.6 | - | 1.4 | V |
| Low Level Trigger Voltage | VT2- | CVDD = 1.65V | 0.3 | - | 1.1 | V |
| Hysteresis Voltage | ΔV2 | CVDD = 1.65V | 0.2 | - | - | V |
| Input Characteristics | Pin name: | VBUSFLG | | | | |
| Pullup Resistance | RPLU2H | VI=VSS | 50 | 100 | 240 | kΩ |
| Input Characteristics | Pin name: | ATPGEN, BURNIN | | | | |
| Pulldown Resistance | RPLD1L | VI=LVDD | 24 | 60 | 150 | kΩ |
| Input Characteristics | Pin name: | TESTEN | | | | |
| Pulldown Resistance | RPLD2L | VI= LVDD | 48 | 120 | 300 | kΩ |

| Parameter | Symbol | Condition | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|---|
| Input Characteristics | Pin name: | VBUS | | | | |
| Pulldown Resistance | RPLD3L | VI= 5.0V | 110 | 125 | 150 | kΩ |
| Output Characteristics | Pin name: | CD [15:0], XDREQ, XINT | | | | |
| High Level Output Voltage | VOH1 | CVDD = 3.0V<br>IOH = -0.2mA | CVDD-0.4 | - | - | V<br>V |
| Low Level Output Voltage | VOL1 | CVDD = 3.0V<br>IOL = 2.0mA | - | - | VSS+0.4 | V |
| High Level Output Voltage | VOH2 | CVDD = 1.65V<br>IOH = -1.0mA | CVDD-0.4 | - | - | V |
| Low Level Output Voltage | VOL2 | CVDD = 1.65V<br>IOL = 1.0mA | - | - | VSS+0.4 | V |
| Output Characteristics | Pin name: | VBUSEN | | | | |
| High Level Output Voltage | VOH3 | HVDD = 3.0V<br>IOH = -2.0mA | HVDD-0.4 | - | - | V |
| Low Level Output Voltage | VOL3 | HVDD = 3.0V<br>IOL = 2.0mA | - | - | 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 |
| 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: | CD [15:0], XINT | | | | |
| OFF-State Leakage Current | IOZ | CVDD=3.6V<br>VOH=CVDD<br>VOL=VSS | -5 | - | 5 | µA |

| Parameter | Symbol | Condition | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|---|
| Pin Capacitance | Pin name: | All input pins | | | | |
| Input Pin Capacitance | CI | f = 1MHz<br>HVDD=CVDD=LVDD=VSS | - | - | 8 | pF |
| Pin Capacitance | Pin name: | All output pins | | | | |
| Output Pin Capacitance | CO | f = 1MHz<br>HVDD=CVDD=LVDD=VSS | - | - | 8 | pF |
| Pin Capacitance | Pin name: | All input/output pins (not including DP and DM) | | | | |
| Input/Output Pin Capacitance 1 | CIO1 | f = 1MHz<br>HVDD=UVDD=LVDD=VSS | - | - | 8 | pF |
| Pin Capacitance | Pin name: | DP, DM | | | | |
| Input/Output Pin Capacitance 2 | CIO2 | f = 1MHz<br>HVDD=CVDD=LVDD=VSS | - | - | 11 | pF |

## 8.4 A.C. Characteristics

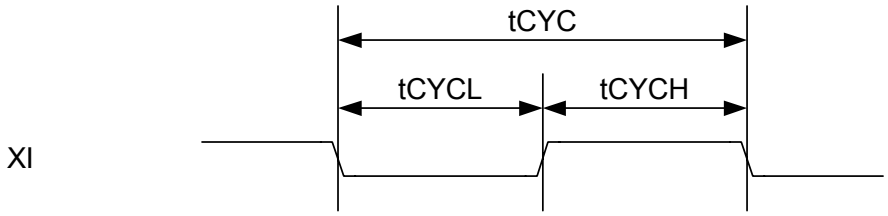
### 8.4.1 RESET Timing




| Symbol | Description | min | typ | max | Unit |
|---|---|---|---|---|---|
| tRESET | Reset pulse width | 40 | - | - | ns |

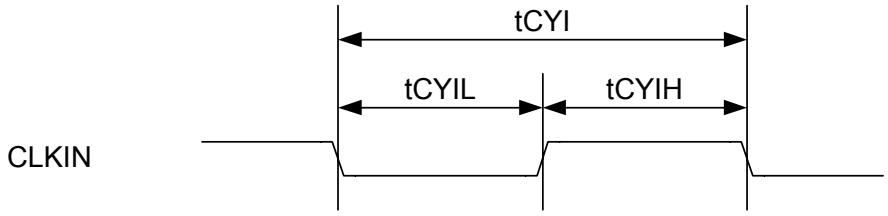
### 8.4.2 Clock Timing

<Internal oscillator>




| Symbol | Description | min | typ | max | Unit |
|---|---|---|---|---|---|
| tCYC(*) | Clock cycle(ClkFreq=0b00) | 11.9988 | 12 | 12.0012 | MHz |
| tCYC(*) | Clock cycle(ClkFreq=0b01) | 23.9976 | 24 | 24.0024 | MHz |
| tCYCH<br>tCYCL | Clock duty | 45 | - | 55 | % |

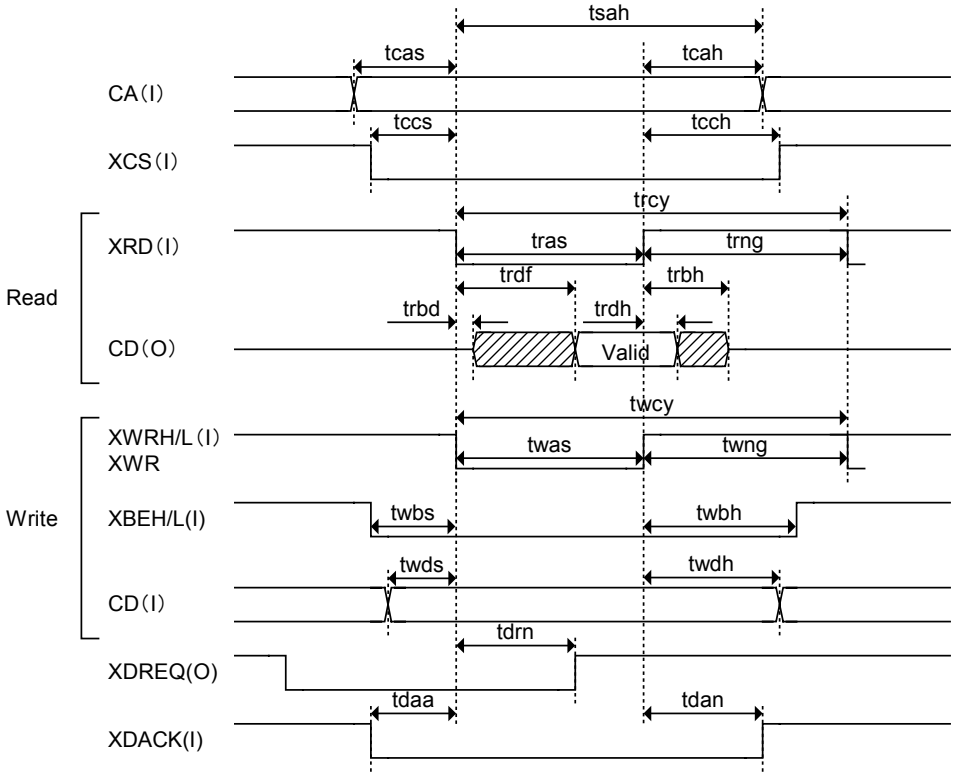
<External input>




| Symbol | Description | min | typ | max | Unit |
|---|---|---|---|---|---|
| tCYI | Clock cycle(ClkFreq=0b00) | 11.9988 | 12 | 12.0012 | MHz |
| tCYI | Clock cycle(ClkFreq=0b01) | 23.9976 | 24 | 24.0024 | MHz |
| tCYI | Clock cycle(ClkFreq=0b11) | 47.9952 | 48 | 48.0048 | MHz |
| tCYIH<br>tCYIL | Clock duty | 45 | - | 55 | % |

### 8.4.3 CPU and DMA I/F Access Timing

#### 8.4.3.1 Basic Cycles



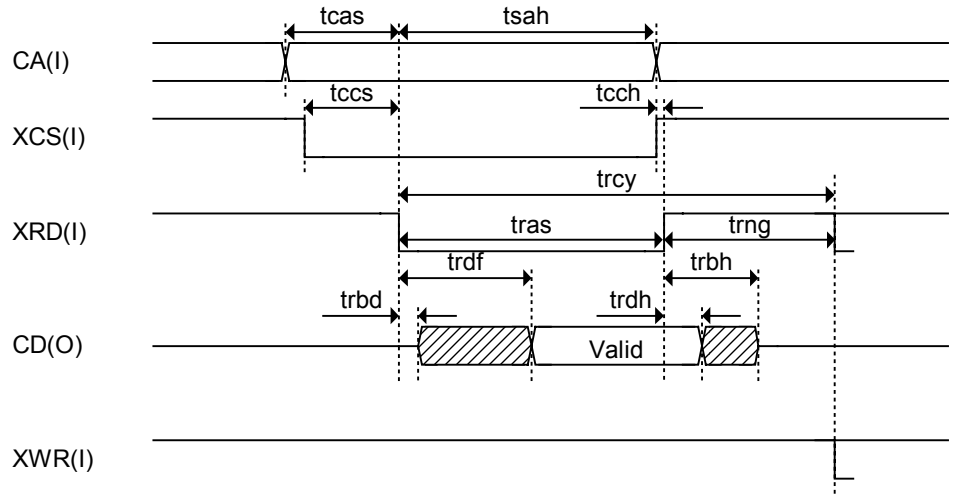

(CL=30pF)

| Symbol | Description | min | typ | max | unit |
|---|---|---|---|---|---|
| tcas | Address setup time | 6 | - | - | ns |
| tcah | Address hold time (from nagation of the strobe) | 6 | - | - | ns |
| tsah | Address hold time ( from assertion of the strobe) | 55 | - | - | 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 | 3 | - | - | ns |
| trbh | Read data output dielay time | - | - | 9 | ns |
| twcy | Write cycle | 80 | - | - | ns |
| twas | Write strobe assert time | 40 | - | - | ns |
| twng | Write strobe negate time | 25 | - | - | ns |
| twbs | Write byte enable seetup 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 | XDERQ negate delay time | - | - | 35 | ns |
| tdaa | XDACK setup time | 6 | - | - | ns |
| tdan | XDACK hold time | 6 | - | - | ns |

### 8.4.3.2 BE Mode Read Timing (when Not Using DMA)

If DMA is not used, the AC characteristics stipulated for read operation are partly eased.

BE mode read timing (when not using DMA: DMA_Config.ActivreDMA and DMA_Mode bits = 0)




($C_L$=30pF)

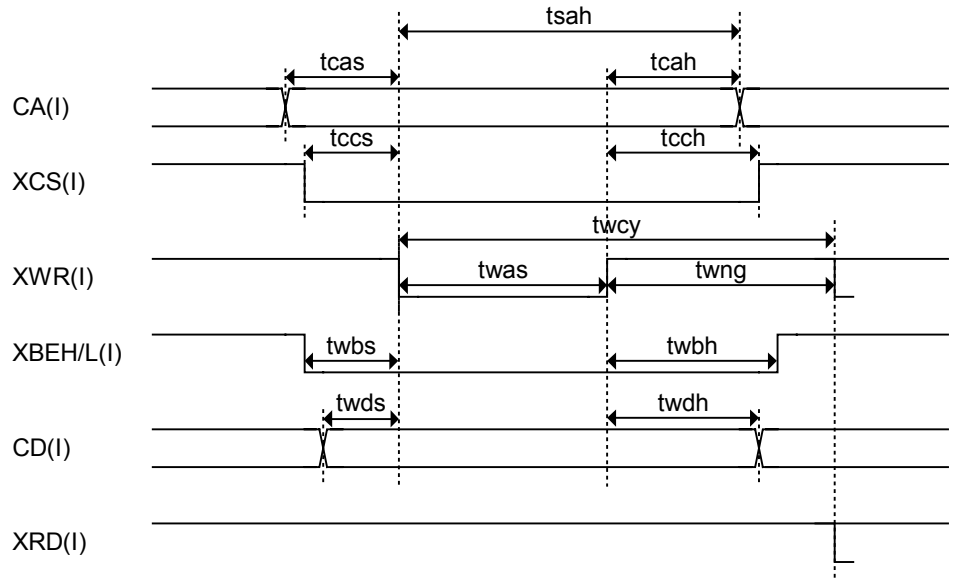
| Symbol | Parameter | min | typ | max | unit |
|---|---|---|---|---|---|
| tcas | Address setup time | 6 | - | - | ns |
| tsah | Address hold time (from strobe assertion) | 55 | - | - | ns |
| tccs | XCS setup time | 6 | - | - | ns |
| tcch | XCS hold time* | - | 0 | - | ns |
| trcy | Read cycle | 80 | - | - | 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 | 3 | - | - | ns |
| trbh | Read data output delay time | - | - | 9 | ns |

*: If XCS is negated before XRD is negated, tras and trdh are regulated by XCS negation.

### 8.4.3.3 BE Mode Write Timing (when Not Using DMA)

If DMA is not used, the AC characteristics stipulated for write operations are partially eased.

BE mode write timing (when not using DMA: DMA_Config.ActivreDMA and DMA_Mode bits = 0)




($C_L$=30pF)

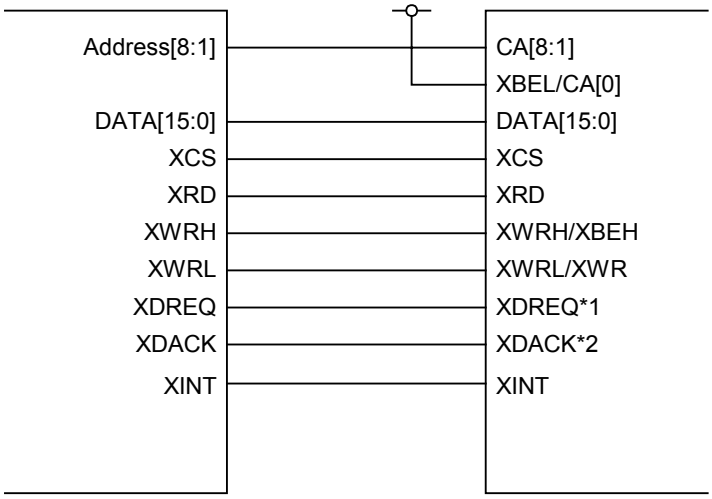
| Symbol | Parameter | min | typ | max | unit |
|---|---|---|---|---|---|
| tcas | Address setup time | 6 | - | - | ns |
| tcah | Address hold time (from strobe negation) | 6 | - | - | ns |
| tsah | Address hold time (from strobe assertion) | 55 | - | - | ns |
| tccs | XCS setup time | 6 | - | - | ns |
| tcch | XCS hold time | 6 | - | - | ns |
| twcy | Write cycle | 80 | - | - | 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 |

### 8.4.4 USB I/F Timing

The USB I/F timing conforms to the USB 2.0 Standard.

# 9. Connection Examples

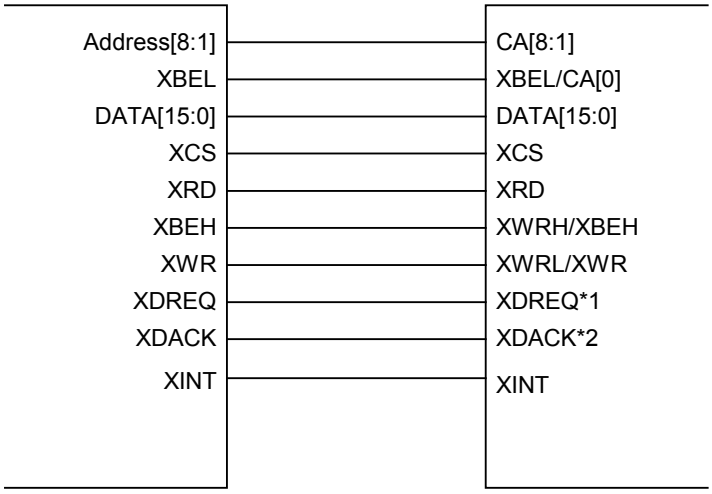
## 9.1 CPU I/F Connection Example




*1: Leave these pins open when not using DMA.

*2: Fix these pins either high or low when not using DMA.

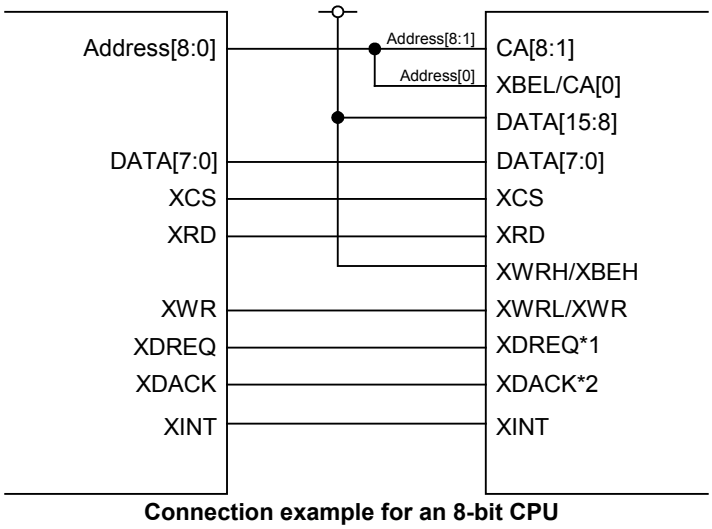
**Connection example for a 16bit CPU(XWRH/XWRL)**




*1: Leave these pins open when not using DMA.

*2: Fix these pins either high or low when not using DMA.

**Connection example for a 16bit CPU(XBEH/XBEL)**



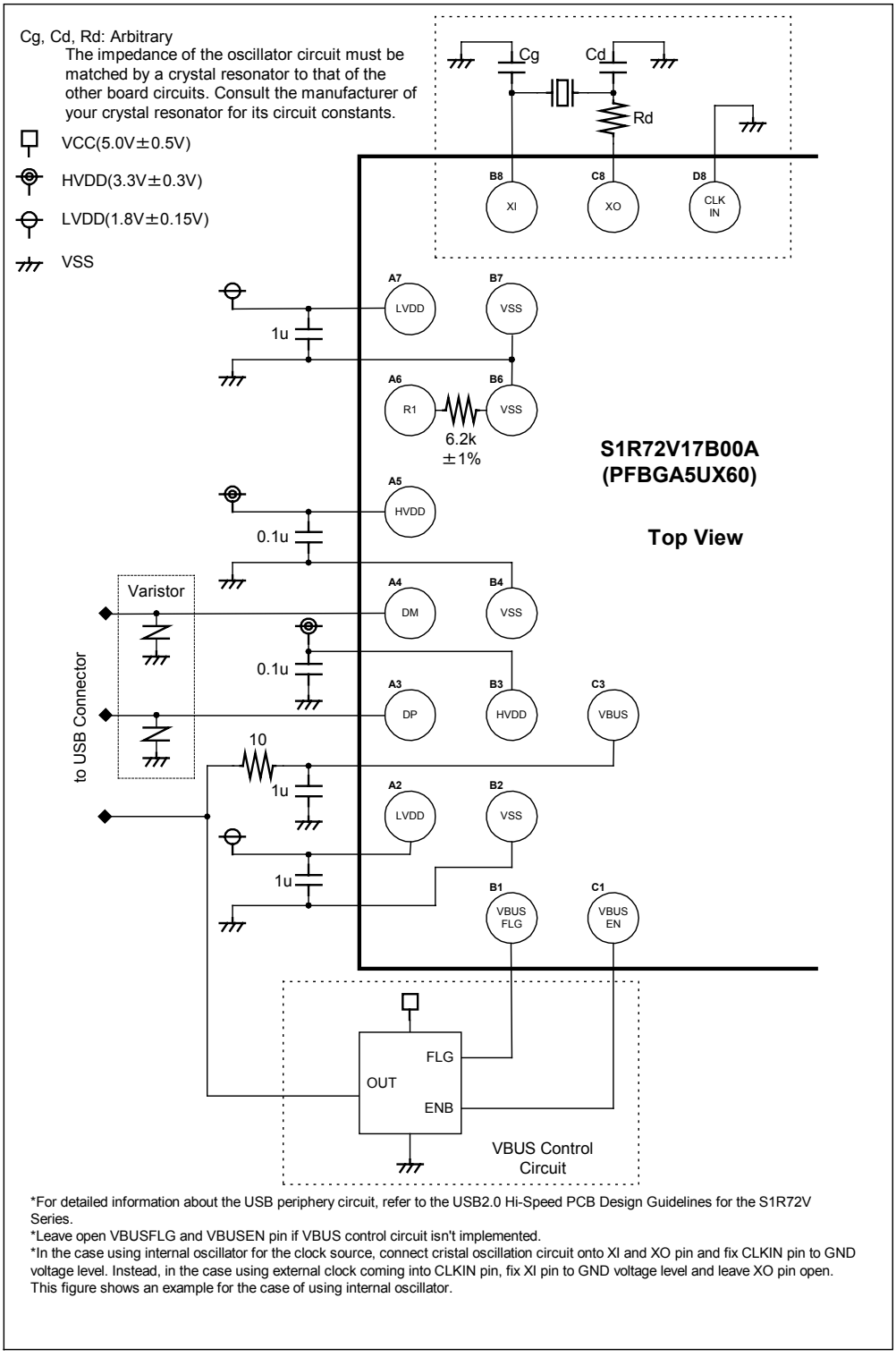

*1: Leave these pins open when not using DMA.

*2: Fix these pins either high or low when not using DMA.

**Connection example for an 8-bit CPU**

## 9.2 USB I/F Connection Example

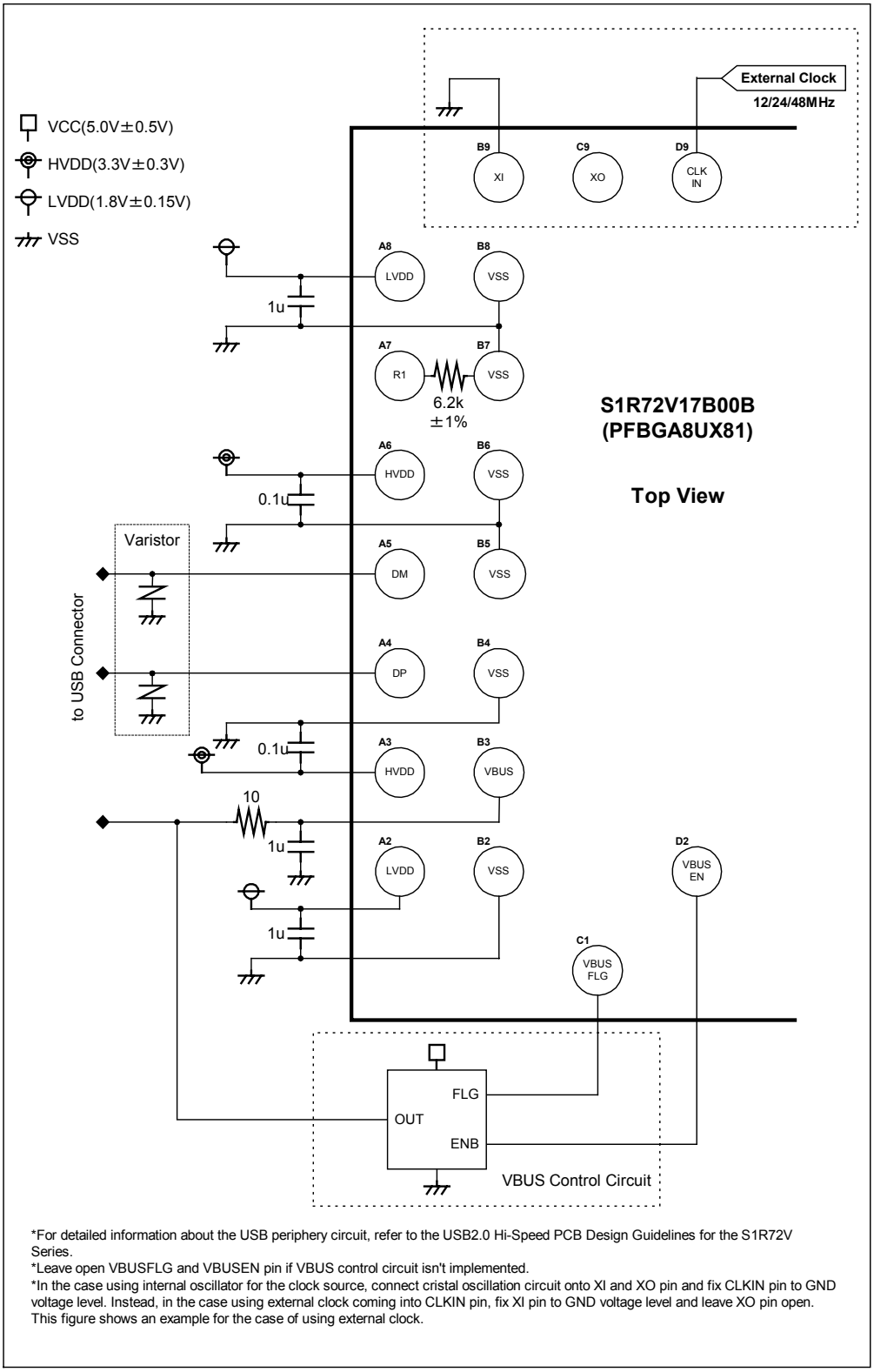
### 9.2.1 For the PFBGA5UX60




*For detailed information about the USB periphery circuit, refer to the USB2.0 Hi-Speed PCB Design Guidelines for the S1R72V Series.
*Leave open VBUSFLG and VBUSEN pin if VBUS control circuit isn't implemented.
*In the case using internal oscillator for the clock source, connect cristal oscillation circuit onto XI and XO pin and fix CLKIN pin to GND voltage level. Instead, in the case using external clock coming into CLKIN pin, fix XI pin to GND voltage level and leave XO pin open. This figure shows an example for the case of using internal oscillator.

Select power supply devices carefully. The performance of such devices will affect the quality of USB signal waveforms.

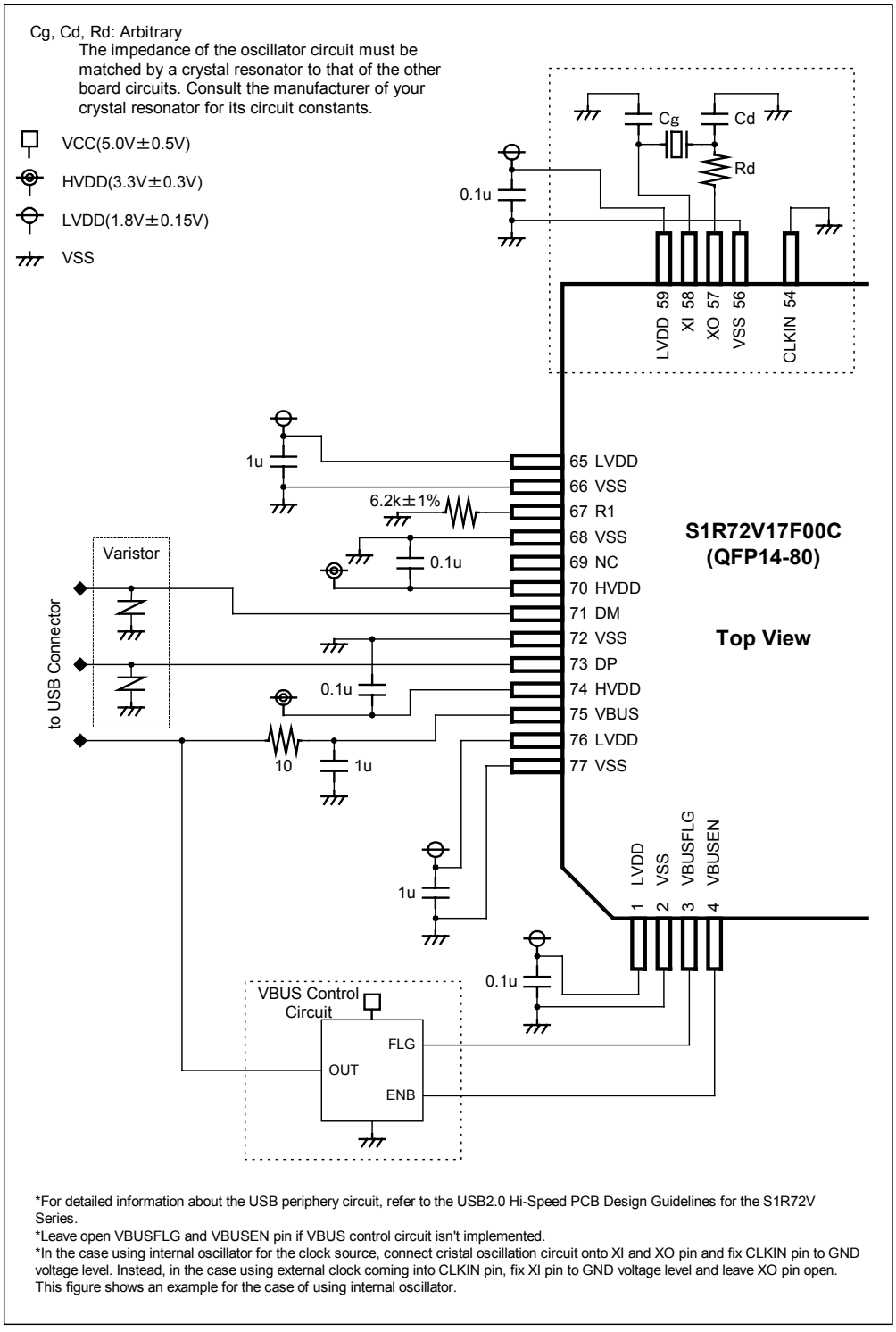
### 9.2.2 For the PFBGA8UX81




*For detailed information about the USB periphery circuit, refer to the USB2.0 Hi-Speed PCB Design Guidelines for the S1R72V Series.
*Leave open VBUSFLG and VBUSEN pin if VBUS control circuit isn't implemented.
*In the case using internal oscillator for the clock source, connect cristal oscillation circuit onto XI and XO pin and fix CLKIN pin to GND voltage level. Instead, in the case using external clock coming into CLKIN pin, fix XI pin to GND voltage level and leave XO pin open. This figure shows an example for the case of using external clock.

Select power supply devices carefully. The performance of such devices will affect the quality of USB signal waveforms.

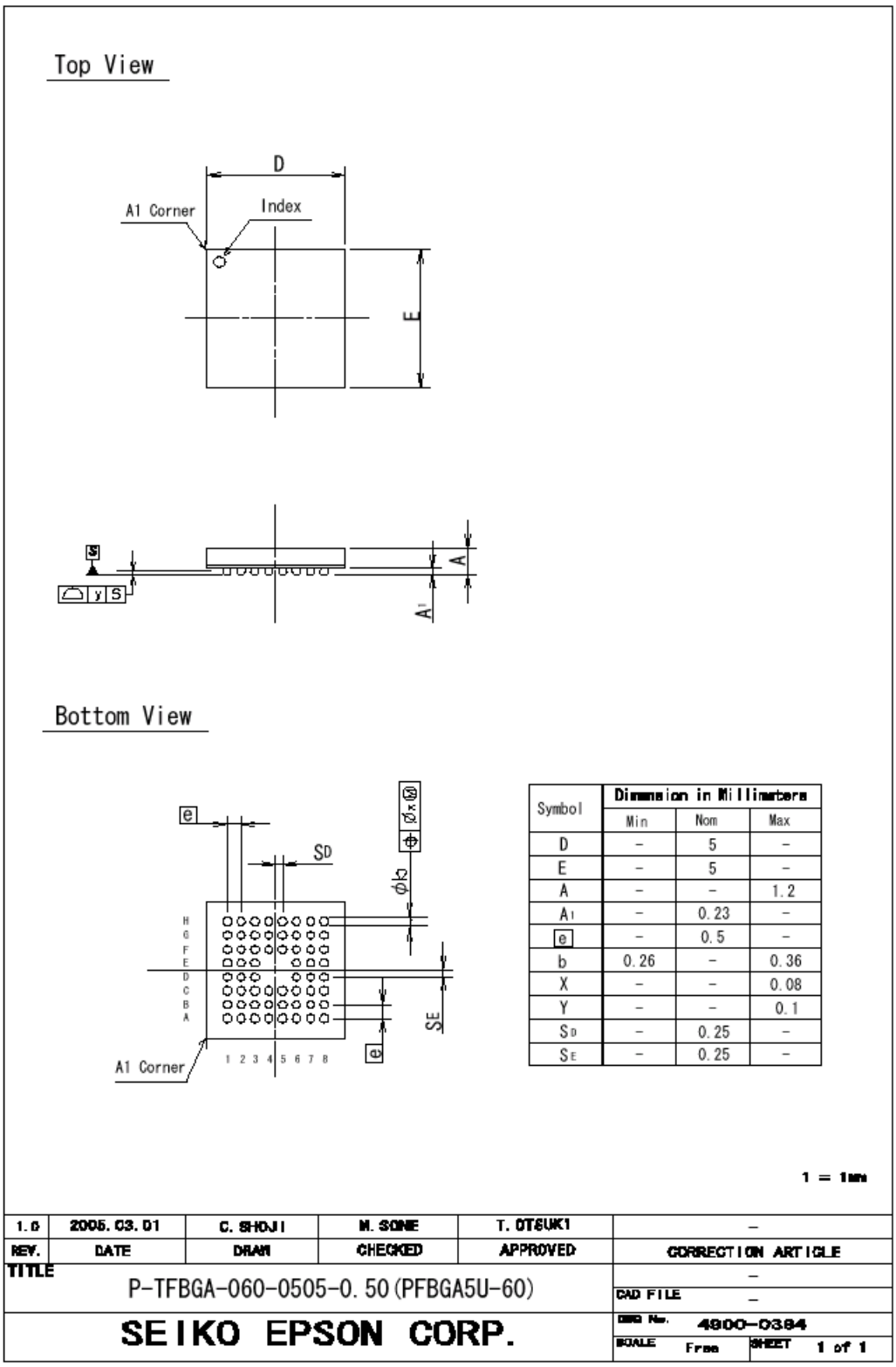
### 9.2.3 For the QFP14-80




Select power supply devices carefully. The performance of such devices will affect the quality of USB signal waveforms.

# 10. Package Dimensions

## 10.1 PFBGA5UX60



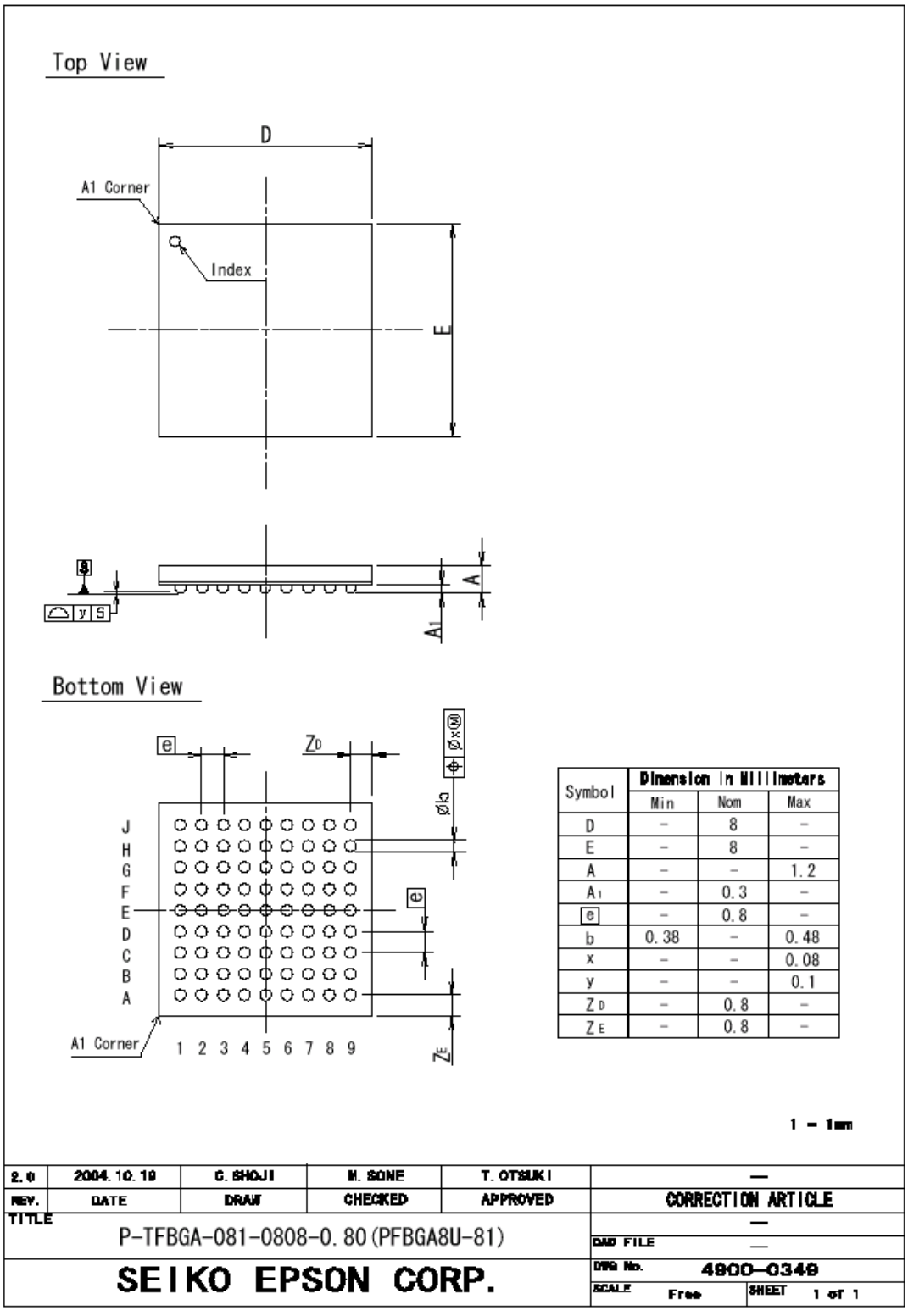

| Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | – | 5 | – |
| E | – | 5 | – |
| A | – | – | 1.2 |
| A1 | – | 0.23 | – |
| e | – | 0.5 | – |
| b | 0.26 | – | 0.36 |
| X | – | – | 0.08 |
| Y | – | – | 0.1 |
| SD | – | 0.25 | – |
| SE | – | 0.25 | – |

1 = 1mm

| | | | | | |
|---|---|---|---|---|---|
| 1.6 | 2005.03.01 | C. SHOJI | M. SONE | T. OTSUKI | – |
| REV. | DATE | DRAW | CHECKED | APPROVED | CORRECTION ARTICLE |
| TITLE | P-TFBGA-060-0505-0.50 (PFBGA5U-60) | | | | – |
| | | | | | CAD FILE – |
| SEIKO EPSON CORP. | | | | | DRG No. 4900-0384 |
| | | | | | SCALE Free SHEET 1 of 1 |

2900-0002-01 (Rev. 1.1)

## 10.2 PFBGA8UX81



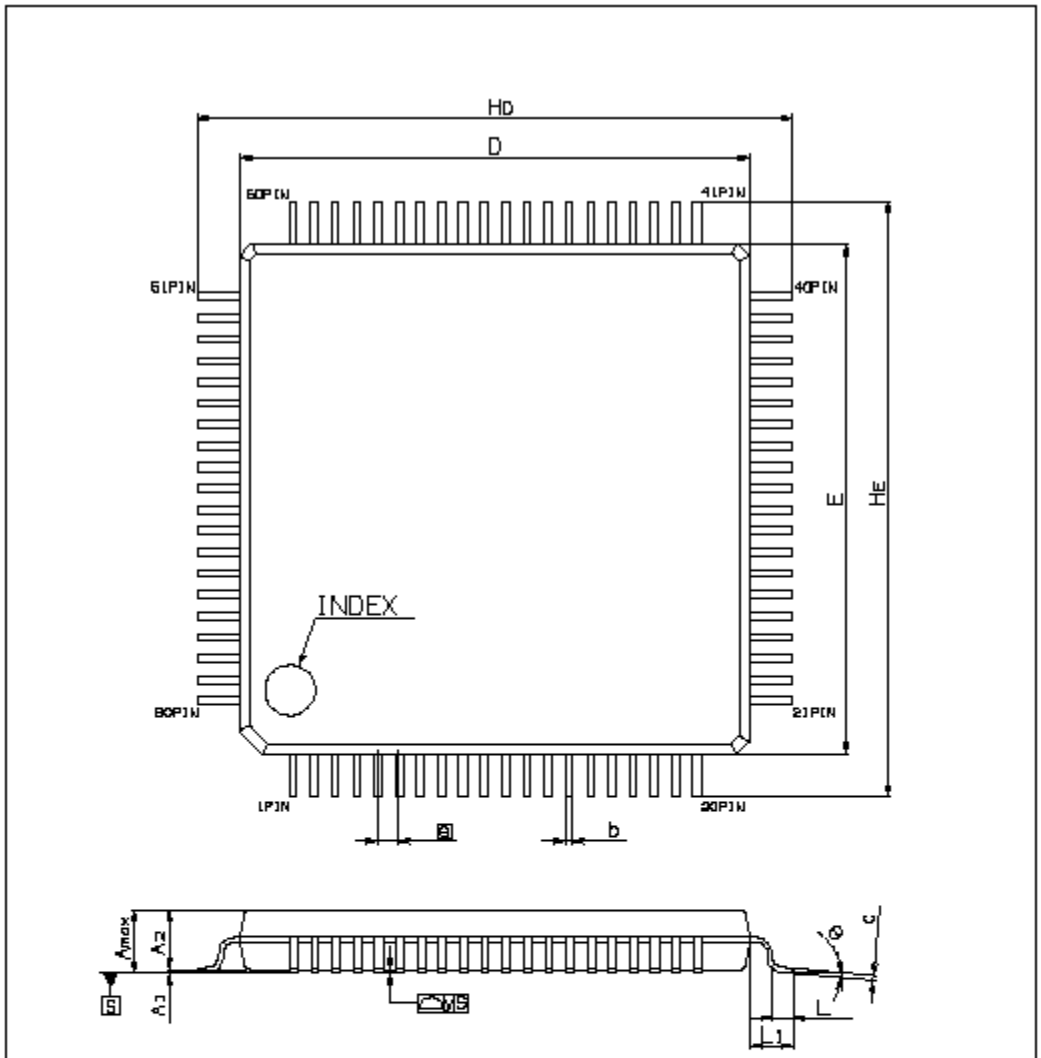

| Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| D | – | 8 | – |
| E | – | 8 | – |
| A | – | – | 1.2 |
| $A_1$ | – | 0.3 | – |
| e | – | 0.8 | – |
| b | 0.38 | – | 0.48 |
| x | – | – | 0.08 |
| y | – | – | 0.1 |
| $Z_D$ | – | 0.8 | – |
| $Z_E$ | – | 0.8 | – |

1 – 1mm

| | | | | | |
|---|---|---|---|---|---|
| 2.0 | 2004.10.19 | G.SHOJI | M.SONE | T.OTSUKI | — |
| REV. | DATE | DRAW | CHECKED | APPROVED | CORRECTION ARTICLE |
| TITLE: P-TFBGA-081-0808-0.80(PFBGA8U-81) | | | | | — |
| | | | | | DAD FILE — |
| SEIKO EPSON CORP. | | | | | DWG No. 4900-0349 |
| | | | | | SCALE Free SHEET 1 of 1 |

2900-0002-01(Rev.1.1)

## 10.3 QFP14-80




| Symbol | Dimension in Millimters | | |
|---|---|---|---|
| | Min | Nom | Max |
| E | - | 12 | - |
| D | - | 12 | - |
| Amax | - | - | 1.7 |
| A1 | - | 0.1 | - |
| A2 | - | 1.4 | - |
| e | - | 0.5 | - |
| b | 0.13 | - | 0.27 |
| c | 0.08 | - | 0.2 |
| θ | 0° | - | 10° |
| L | 0.3 | - | 0.75 |
| L1 | - | 1 | - |
| HE | - | 14 | - |
| HD | - | 14 | - |
| y | - | - | 0.08 |

1 = 1mm

| | | | | | |
|---|---|---|---|---|---|
| 2.0 | 2004.08.03 | C.SAITO | M.SONE | T.OTSUKI | - |
| REV. | DATE | DRAW | CHECKED | APPROVED | CORRECTION ARTICLE |
| TITLE P-LQFP080-1212-0.50(QFP14-80PIN) | | | | | - |
| | | | | CAD FILE | - |
| SEIKO EPSON CORP. | | | | DWG No. | 4900-0017 |
| | | | | SCALE Free | SHEET 1 of 1 |

2900-0002-01(Rev.1.1)

# Appendix A. Connecting to Little Endian CPUs

The internal buses of the S1R72V17 are configured with the big endian, with even and the odd addresses comprising the upper and lower bytes, respectively. If the S1R72V17 is to be used with a little endian CPU, follow the procedures given below for connecting to the little endian CPU.

**<Circuit board>**

The little endian CPU pins and the S1R72V17 pins for data bus and control signals can be connected directly one-to-one, as indicated by pin names. Connect CD15–CD8 of the S1R72V17 to data bus bits 15–8 or the upper byte of the CPU, then connect CD7–CD0 of the S1R72V17 to data bus bits 7–0, or the lower byte of the CPU. Similarly, write signals can be connected directly, high for high and low for low.

Keep in mind that write signal specifications for a specific CPU used may differ from those for other CPUs.

**<Firmware>**

If the S1R72V17 must be operated in a little endian CPU, follow the procedure given below.

(1) Set the CPU_Config.CPU_Endian bit to 1.
Although this register in the S1R72V17 is mapped to address 0x075, when operating in a little endian CPU, it will behave as if this register is mapped to address 0x074 until the operation in (2) is performed. This is because the S1R72V17 in its initial state is a big endian device, in which the upper and lower bytes of the write signals are reversed.

(2) Read the address 0x077.
This read operation causes the S1R72V17 to reverse the upper and lower bytes of its CPU buses. Keep in mind that performing (1) alone will not affect the byte order of the CPU buses. Once this read operation is performed, all registers are mapped to the addresses shown in the register map in the next pages.

Once the above settings are made, all internal registers can be accessed in Char or Short. No problems should arise, even when the registers are accessed using the DMAC of the CPU (see the table below).

Example: Access to the FIFO_Rd_0/1 register when data is received from USB sequentially in order of 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 |

However, use registers larger than Short by separately accessing in Short units and casting types in CPU memory.

**Big Endian**

**Common registers**

| | |
|---|---|
| 0x000 | ***MainIntStat*** |
| 0x001 | ***USB_DeviceIntStat*** |
| 0x002 | ***USB_HostIntStat*** |
| 0x003 | CPU_IntStat |
| 0x004 | FIFO_IntStat |
| 0x005 | |
| 0x006 | |
| 0x007 | |
| 0x008 | ***MainIntEnb*** |
| 0x009 | ***USB_DeviceIntEnb*** |
| 0x00A | ***USB_HostIntEnb*** |
| 0x00B | CPU_IntEnb |
| 0x00C | FIFO_IntEnb |
| 0x00D | |
| 0x00E | |
| 0x00F | |

| | |
|---|---|
| 0x010 | ***RevisionNum*** |
| 0x011 | ***ChipReset*** |
| 0x012 | ***PM_Control*** |
| 0x013 | |
| 0x014 | ***WakeupTim_H*** |
| 0x015 | ***WakeupTim_L*** |
| 0x016 | ***H_USB_Control*** |
| 0x017 | ***H_XcvrControl*** |
| 0x018 | ***D_USB_Status*** |
| 0x019 | ***H_USB_Status*** |
| 0x01A | |
| 0x01B | MTM_Config |
| 0x01C | |
| 0x01D | |
| 0x01E | |
| 0x01F | ***HostDeviceSel*** |

| | |
|---|---|
| 0x020 | FIFO_Rd_0 |
| 0x021 | FIFO_Rd_1 |
| 0x022 | FIFO_Wr_0 |

**Little Endian**

Registers whose upper and lower bytes change during little endian

**Common registers**

| | |
|---|---|
| 0x000 | ***MainIntStat*** |
| 0x001 | ***USB_DeviceIntStat*** |
| 0x002 | ***USB_HostIntStat*** |
| 0x003 | CPU_IntStat |
| 0x004 | FIFO_IntStat |
| 0x005 | |
| 0x006 | |
| 0x007 | |
| 0x008 | ***MainIntEnb*** |
| 0x009 | ***USB_DeviceIntEnb*** |
| 0x00A | ***USB_HostIntEnb*** |
| 0x00B | CPU_IntEnb |
| 0x00C | FIFO_IntEnb |
| 0x00D | |
| 0x00E | |
| 0x00F | |

| | |
|---|---|
| 0x010 | ***RevisionNum*** |
| 0x011 | ***ChipReset*** |
| 0x012 | ***PM_Control*** |
| 0x013 | |
| 0x014 | ***WakeupTim_L*** |
| 0x015 | ***WakeupTim_H*** |
| 0x016 | ***H_USB_Control*** |
| 0x017 | ***H_XcvrControl*** |
| 0x018 | ***D_USB_Status*** |
| 0x019 | ***H_USB_Status*** |
| 0x01A | |
| 0x01B | MTM_Config |
| 0x01C | |
| 0x01D | |
| 0x01E | |
| 0x01F | ***HostDeviceSel*** |

| | |
|---|---|
| 0x020 | FIFO_Rd_0 |
| 0x021 | FIFO_Rd_1 |
| 0x022 | FIFO_Wr_0 |

| | |
|---|---|
| 0x023 | FIFO_Wr_1 |
| 0x024 | FIFO_RdRemain_H |
| 0x025 | FIFO_RdRemain_L |
| 0x026 | FIFO_WrRemain_H |
| 0x027 | FIFO_WrRemain_L |
| 0x028 | FIFO_ByteRd |
| 0x029 | |
| 0x02A | |
| 0x02B | |
| 0x02C | |
| 0x02D | |
| 0x02E | |
| 0x02F | |

| | |
|---|---|
| 0x023 | FIFO_Wr_1 |
| 0x024 | FIFO_RdRemain_L |
| 0x025 | FIFO_RdRemain_H |
| 0x026 | FIFO_WrRemain_L |
| 0x027 | FIFO_WrRemain_H |
| 0x028 | FIFO_ByteRd |
| 0x029 | |
| 0x02A | |
| 0x02B | |
| 0x02C | |
| 0x02D | |
| 0x02E | |
| 0x02F | |

| | |
|---|---|
| 0x030 | RAM_RdAdrs_H |
| 0x031 | RAM_RdAdrs_L |
| 0x032 | RAM_RdControl |
| 0x033 | |
| 0x034 | |
| 0x035 | RAM_RdCount |
| 0x036 | |
| 0x037 | |
| 0x038 | RAM_WrAdrs_H |
| 0x039 | RAM_WrAdrs_L |
| 0x03A | RAM_WrDoor_0 |
| 0x03B | RAM_WrDoor_1 |
| 0x03C | |
| 0x03D | |
| 0x03E | |
| 0x03F | |

| | |
|---|---|
| 0x030 | RAM_RdAdrs_L |
| 0x031 | RAM_RdAdrs_H |
| 0x032 | RAM_RdControl |
| 0x033 | |
| 0x034 | |
| 0x035 | RAM_RdCount |
| 0x036 | |
| 0x037 | |
| 0x038 | RAM_WrAdrs_L |
| 0x039 | RAM_WrAdrs_H |
| 0x03A | RAM_WrDoor_0 |
| 0x03B | RAM_WrDoor_1 |
| 0x03C | |
| 0x03D | |
| 0x03E | |
| 0x03F | |

| | |
|---|---|
| 0x040 | RAM_Rd_00 |
| 0x041 | RAM_Rd_01 |
| 0x042 | RAM_Rd_02 |
| 0x043 | RAM_Rd_03 |
| 0x044 | RAM_Rd_04 |
| 0x045 | RAM_Rd_05 |
| 0x046 | RAM_Rd_06 |
| 0x047 | RAM_Rd_07 |
| 0x048 | RAM_Rd_08 |

| | |
|---|---|
| 0x040 | RAM_Rd_00 |
| 0x041 | RAM_Rd_01 |
| 0x042 | RAM_Rd_02 |
| 0x043 | RAM_Rd_03 |
| 0x044 | RAM_Rd_04 |
| 0x045 | RAM_Rd_05 |
| 0x046 | RAM_Rd_06 |
| 0x047 | RAM_Rd_07 |
| 0x048 | RAM_Rd_08 |

| | |
|---|---|
| 0x049 | RAM_Rd_09 |
| 0x04A | RAM_Rd_0A |
| 0x04B | RAM_Rd_0B |
| 0x04C | RAM_Rd_0C |
| 0x04D | RAM_Rd_0D |
| 0x04E | RAM_Rd_0E |
| 0x04F | RAM_Rd_0F |

| | |
|---|---|
| 0x049 | RAM_Rd_09 |
| 0x04A | RAM_Rd_0A |
| 0x04B | RAM_Rd_0B |
| 0x04C | RAM_Rd_0C |
| 0x04D | RAM_Rd_0D |
| 0x04E | RAM_Rd_0E |
| 0x04F | RAM_Rd_0F |

| | |
|---|---|
| 0x050 | RAM_Rd_10 |
| 0x051 | RAM_Rd_11 |
| 0x052 | RAM_Rd_12 |
| 0x053 | RAM_Rd_13 |
| 0x054 | RAM_Rd_14 |
| 0x055 | RAM_Rd_15 |
| 0x056 | RAM_Rd_16 |
| 0x057 | RAM_Rd_17 |
| 0x058 | RAM_Rd_18 |
| 0x059 | RAM_Rd_19 |
| 0x05A | RAM_Rd_1A |
| 0x05B | RAM_Rd_1B |
| 0x05C | RAM_Rd_1C |
| 0x05D | RAM_Rd_1D |
| 0x05E | RAM_Rd_1E |
| 0x05F | RAM_Rd_1F |

| | |
|---|---|
| 0x050 | RAM_Rd_10 |
| 0x051 | RAM_Rd_11 |
| 0x052 | RAM_Rd_12 |
| 0x053 | RAM_Rd_13 |
| 0x054 | RAM_Rd_14 |
| 0x055 | RAM_Rd_15 |
| 0x056 | RAM_Rd_16 |
| 0x057 | RAM_Rd_17 |
| 0x058 | RAM_Rd_18 |
| 0x059 | RAM_Rd_19 |
| 0x05A | RAM_Rd_1A |
| 0x05B | RAM_Rd_1B |
| 0x05C | RAM_Rd_1C |
| 0x05D | RAM_Rd_1D |
| 0x05E | RAM_Rd_1E |
| 0x05F | RAM_Rd_1F |

| | |
|---|---|
| 0x060 | |
| 0x061 | DMA_Config |
| 0x062 | DMA_Control |
| 0x063 | |
| 0x064 | DMA_Remain_H |
| 0x065 | DMA_Remain_L |
| 0x066 | |
| 0x067 | |
| 0x068 | DMA_Count_HH |
| 0x069 | DMA_Count_HL |
| 0x06A | DMA_Count_LH |
| 0x06B | DMA_Count_LL |
| 0x06C | DMA_RdData_0 |
| 0x06D | DMA_RdData_1 |

| | |
|---|---|
| 0x060 | |
| 0x061 | DMA_Config |
| 0x062 | DMA_Control |
| 0x063 | |
| 0x064 | DMA_Remain_L |
| 0x065 | DMA_Remain_H |
| 0x066 | |
| 0x067 | |
| 0x068 | DMA_Count_HL |
| 0x069 | DMA_Count_HH |
| 0x06A | DMA_Count_LL |
| 0x06B | DMA_Count_LH |
| 0x06C | DMA_RdData_0 |
| 0x06D | DMA_RdData_1 |

| | |
|---|---|
| 0x06E | DMA_WrData_0 |
| 0x06F | DMA_WrData_1 |

| | |
|---|---|
| 0x06E | DMA_WrData_0 |
| 0x06F | DMA_WrData_1 |

| | |
|---|---|
| 0x070 | |
| 0x071 | ***ModeProtect*** |
| 0x072 | |
| 0x073 | ***ClkSelect*** |
| 0x074 | |
| 0x075 | ***CPU_Config*** |
| 0x076 | |
| 0x077 | ***CPU_ChgEndian*** |
| 0x078 | |
| 0x079 | |
| 0x07A | |
| 0x07B | |
| 0x07C | CPU_ChacheErr |
| 0x07D | CPU_BufWrErr |
| 0x07E | TestStatus |
| 0x07F | |

| | |
|---|---|
| 0x070 | |
| 0x071 | ***ModeProtect*** |
| 0x072 | |
| 0x073 | ***ClkSelect*** |
| 0x074 | |
| 0x075 | ***CPU_Config*** |
| 0x076 | |
| 0x077 | ***CPU_ChgEndian*** |
| 0x078 | |
| 0x079 | |
| 0x07A | |
| 0x07B | |
| 0x07C | CPU_ChacheErr |
| 0x07D | CPU_BufWrErr |
| 0x07E | TestStatus |
| 0x07F | |

| | |
|---|---|
| 0x080 | AREA0StartAdrs_H |
| 0x081 | AREA0StartAdrs_L |
| 0x082 | AREA0EndAdrs_H |
| 0x083 | AREA0EndAdrs_L |
| 0x084 | AREA1StartAdrs_H |
| 0x085 | AREA1StartAdrs_L |
| 0x086 | AREA1EndAdrs_H |
| 0x087 | AREA1EndAdrs_L |
| 0x088 | AREA2StartAdrs_H |
| 0x089 | AREA2StartAdrs_L |
| 0x08A | AREA2EndAdrs_H |
| 0x08B | AREA2EndAdrs_L |
| 0x08C | AREA3StartAdrs_H |
| 0x08D | AREA3StartAdrs_L |
| 0x08E | AREA3EndAdrs_H |
| 0x08F | AREA3EndAdrs_L |

| | |
|---|---|
| 0x080 | AREA0StartAdrs_L |
| 0x081 | AREA0StartAdrs_H |
| 0x082 | AREA0EndAdrs_L |
| 0x083 | AREA0EndAdrs_H |
| 0x084 | AREA1StartAdrs_L |
| 0x085 | AREA1StartAdrs_H |
| 0x086 | AREA1EndAdrs_L |
| 0x087 | AREA1EndAdrs_H |
| 0x088 | AREA2StartAdrs_L |
| 0x089 | AREA2StartAdrs_H |
| 0x08A | AREA2EndAdrs_L |
| 0x08B | AREA2EndAdrs_H |
| 0x08C | AREA3StartAdrs_L |
| 0x08D | AREA3StartAdrs_H |
| 0x08E | AREA3EndAdrs_L |
| 0x08F | AREA3EndAdrs_H |

| | |
|---|---|
| 0x090 | AREA4StartAdrs_H |

| | |
|---|---|
| 0x090 | AREA4StartAdrs_L |

| | |
|---|---|
| 0x091 | AREA4StartAdrs_L |
| 0x092 | AREA4EndAdrs_H |
| 0x093 | AREA4EndAdrs_L |
| 0x094 | AREA5StartAdrs_H |
| 0x095 | AREA5StartAdrs_L |
| 0x096 | AREA5EndAdrs_H |
| 0x097 | AREA5EndAdrs_L |
| 0x098 | |
| 0x099 | |
| 0x09A | |
| 0x09B | |
| 0x09C | |
| 0x09D | |
| 0x09E | |
| 0x09F | AREAnFIFO_Clr |

| | |
|---|---|
| 0x091 | AREA4StartAdrs_H |
| 0x092 | AREA4EndAdrs_L |
| 0x093 | AREA4EndAdrs_H |
| 0x094 | AREA5StartAdrs_L |
| 0x095 | AREA5StartAdrs_H |
| 0x096 | AREA5EndAdrs_L |
| 0x097 | AREA5EndAdrs_H |
| 0x098 | |
| 0x099 | |
| 0x09A | |
| 0x09B | |
| 0x09C | |
| 0x09D | |
| 0x09E | |
| 0x09F | AREAnFIFO_Clr |

| | |
|---|---|
| 0x0A0 | AREA0Join_0 |
| 0x0A1 | AREA0Join_1 |
| 0x0A2 | AREA1Join_0 |
| 0x0A3 | AREA1Join_1 |
| 0x0A4 | AREA2Join_0 |
| 0x0A5 | AREA2Join_1 |
| 0x0A6 | AREA3Join_0 |
| 0x0A7 | AREA3Join_1 |
| 0x0A8 | AREA4Join_0 |
| 0x0A9 | AREA4Join_1 |
| 0x0AA | AREA5Join_0 |
| 0x0AB | AREA5Join_1 |
| 0x0AC | |
| 0x0AD | |
| 0x0AE | ClrAREAnJoin_0 |
| 0x0AF | ClrAREAnJoin_1 |

| | |
|---|---|
| 0x0A0 | AREA0Join_0 |
| 0x0A1 | AREA0Join_1 |
| 0x0A2 | AREA1Join_0 |
| 0x0A3 | AREA1Join_1 |
| 0x0A4 | AREA2Join_0 |
| 0x0A5 | AREA2Join_1 |
| 0x0A6 | AREA3Join_0 |
| 0x0A7 | AREA3Join_1 |
| 0x0A8 | AREA4Join_0 |
| 0x0A9 | AREA4Join_1 |
| 0x0AA | AREA5Join_0 |
| 0x0AB | AREA5Join_1 |
| 0x0AC | |
| 0x0AD | |
| 0x0AE | ClrAREAnJoin_0 |
| 0x0AF | ClrAREAnJoin_1 |

**Device registers (HOSTxDEVICE == 0)**

| | |
|---|---|
| 0x0B0 | ***D_SIE_IntStat*** |
| 0x0B1 | |
| 0x0B2 | |
| 0x0B3 | D_BulkIntStat |
| 0x0B4 | D_EPrIntStat |

**Device registers (HOSTxDEVICE == 0)**

| | |
|---|---|
| 0x0B0 | ***D_SIE_IntStat*** |
| 0x0B1 | |
| 0x0B2 | |
| 0x0B3 | D_BulkIntStat |
| 0x0B4 | D_EPrIntStat |

| | |
|---|---|
| 0x0B5 | D_EP0IntStat |
| 0x0B6 | D_EPaIntStat |
| 0x0B7 | D_EPbIntStat |
| 0x0B8 | D_EPcIntStat |
| 0x0B9 | D_EPdIntStat |
| 0x0BA | D_EPeIntStat |
| 0x0BB | |
| 0x0BC | D_AlarmIN_IntStat_H |
| 0x0BD | D_AlarmIN_IntStat_L |
| 0x0BE | D_AlarmOUT_IntStat_H |
| 0x0BF | D_AlarmOUT_IntStat_L |

| | |
|---|---|
| 0x0B5 | D_EP0IntStat |
| 0x0B6 | D_EPaIntStat |
| 0x0B7 | D_EPbIntStat |
| 0x0B8 | D_EPcIntStat |
| 0x0B9 | D_EPdIntStat |
| 0x0BA | D_EPeIntStat |
| 0x0BB | |
| 0x0BC | D_AlarmIN_IntStat_L |
| 0x0BD | D_AlarmIN_IntStat_H |
| 0x0BE | D_AlarmOUT_IntStat_L |
| 0x0BF | D_AlarmOUT_IntStat_H |

| | |
|---|---|
| 0x0C0 | ***D_SIE_IntEnb*** |
| 0x0C1 | |
| 0x0C2 | — |
| 0x0C3 | D_BulkIntEnb |
| 0x0C4 | D_EPrIntEnb |
| 0x0C5 | D_EP0IntEnb |
| 0x0C6 | D_EPaIntEnb |
| 0x0C7 | D_EPbIntEnb |
| 0x0C8 | D_EPcIntEnb |
| 0x0C9 | D_EPdIntEnb |
| 0x0CA | D_EPeIntEnb |
| 0x0CB | |
| 0x0CC | D_AlarmIN_IntEnb_H |
| 0x0CD | D_AlarmIN_IntEnb_L |
| 0x0CE | D_AlarmOUT_IntEnb_H |
| 0x0CF | D_AlarmOUT_IntEnb_L |

| | |
|---|---|
| 0x0C0 | ***D_SIE_IntEnb*** |
| 0x0C1 | |
| 0x0C2 | — |
| 0x0C3 | D_BulkIntEnb |
| 0x0C4 | D_EPrIntEnb |
| 0x0C5 | D_EP0IntEnb |
| 0x0C6 | D_EPaIntEnb |
| 0x0C7 | D_EPbIntEnb |
| 0x0C8 | D_EPcIntEnb |
| 0x0C9 | D_EPdIntEnb |
| 0x0CA | D_EPeIntEnb |
| 0x0CB | |
| 0x0CC | D_AlarmIN_IntEnb_L |
| 0x0CD | D_AlarmIN_IntEnb_H |
| 0x0CE | D_AlarmOUT_IntEnb_L |
| 0x0CF | D_AlarmOUT_IntEnb_H |

| | |
|---|---|
| 0x0D0 | D_NegoControl |
| 0x0D1 | |
| 0x0D2 | |
| 0x0D3 | D_XcvrControl |
| 0x0D4 | D_USB_Test |
| 0x0D5 | |
| 0x0D6 | D_EPnControl |
| 0x0D7 | |
| 0x0D8 | D_BulkOnlyControl |
| 0x0D9 | D_BulkOnlyConfig |
| 0x0DA | — |

| | |
|---|---|
| 0x0D0 | D_NegoControl |
| 0x0D1 | |
| 0x0D2 | |
| 0x0D3 | D_XcvrControl |
| 0x0D4 | D_USB_Test |
| 0x0D5 | |
| 0x0D6 | D_EPnControl |
| 0x0D7 | |
| 0x0D8 | D_BulkOnlyControl |
| 0x0D9 | D_BulkOnlyConfig |
| 0x0DA | — |

| | |
|---|---|
| 0x0DB | — |
| 0x0DC | — |
| 0x0DD | — |
| 0x0DE | — |
| 0x0DF | — |

| | |
|---|---|
| 0x0DB | — |
| 0x0DC | — |
| 0x0DD | — |
| 0x0DE | — |
| 0x0DF | — |

| | |
|---|---|
| 0x0E0 | D_EP0SETUP_0 |
| 0x0E1 | D_EP0SETUP_1 |
| 0x0E2 | D_EP0SETUP_2 |
| 0x0E3 | D_EP0SETUP_3 |
| 0x0E4 | D_EP0SETUP_4 |
| 0x0E5 | D_EP0SETUP_5 |
| 0x0E6 | D_EP0SETUP_6 |
| 0x0E7 | D_EP0SETUP_7 |
| 0x0E8 | D_USB_Address |
| 0x0E9 | |
| 0x0EA | D_SETUP_Control |
| 0x0EB | |
| 0x0EC | |
| 0x0ED | |
| 0x0EE | D_FrameNumber_H |
| 0x0EF | D_FrameNumber_L |

| | |
|---|---|
| 0x0E0 | D_EP0SETUP_0 |
| 0x0E1 | D_EP0SETUP_1 |
| 0x0E2 | D_EP0SETUP_2 |
| 0x0E3 | D_EP0SETUP_3 |
| 0x0E4 | D_EP0SETUP_4 |
| 0x0E5 | D_EP0SETUP_5 |
| 0x0E6 | D_EP0SETUP_6 |
| 0x0E7 | D_EP0SETUP_7 |
| 0x0E8 | D_USB_Address |
| 0x0E9 | |
| 0x0EA | D_SETUP_Control |
| 0x0EB | |
| 0x0EC | |
| 0x0ED | |
| 0x0EE | D_FrameNumber_L |
| 0x0EF | D_FrameNumber_H |

| | |
|---|---|
| 0x0F0 | D_EP0MaxSize |
| 0x0F1 | D_EP0Control |
| 0x0F2 | D_EP0ControlIN |
| 0x0F3 | D_EP0ControlOUT |
| 0x0F4 | |
| 0x0F5 | |
| 0x0F6 | |
| 0x0F7 | |
| 0x0F8 | D_EPaMaxSize_H |
| 0x0F9 | D_EPaMaxSize_L |
| 0x0FA | D_EPaConfig |
| 0x0FB | |
| 0x0FC | D_EPaControl |
| 0x0FD | |
| 0x0FE | |
| 0x0FF | |

| | |
|---|---|
| 0x0F0 | D_EP0MaxSize |
| 0x0F1 | D_EP0Control |
| 0x0F2 | D_EP0ControlIN |
| 0x0F3 | D_EP0ControlOUT |
| 0x0F4 | |
| 0x0F5 | |
| 0x0F6 | |
| 0x0F7 | |
| 0x0F8 | D_EPaMaxSize_L |
| 0x0F9 | D_EPaMaxSize_H |
| 0x0FA | D_EPaConfig |
| 0x0FB | |
| 0x0FC | D_EPaControl |
| 0x0FD | |
| 0x0FE | |
| 0x0FF | |

| | |
|---|---|
| 0x100 | D_EPbMaxSize_H |
| 0x101 | D_EPbMaxSize_L |
| 0x102 | D_EPbConfig |
| 0x103 | |
| 0x104 | D_EPbControl |
| 0x105 | |
| 0x106 | |
| 0x107 | |
| 0x108 | D_EPcMaxSize_H |
| 0x109 | D_EPcMaxSize_L |
| 0x10A | D_EPcConfig |
| 0x10B | |
| 0x10C | D_EPcControl |
| 0x10D | |
| 0x10E | |
| 0x10F | |

| | |
|---|---|
| 0x100 | D_EPbMaxSize_L |
| 0x101 | D_EPbMaxSize_H |
| 0x102 | D_EPbConfig |
| 0x103 | |
| 0x104 | D_EPbControl |
| 0x105 | |
| 0x106 | |
| 0x107 | |
| 0x108 | D_EPcMaxSize_L |
| 0x109 | D_EPcMaxSize_H |
| 0x10A | D_EPcConfig |
| 0x10B | |
| 0x10C | D_EPcControl |
| 0x10D | |
| 0x10E | |
| 0x10F | |

| | |
|---|---|
| 0x110 | D_EPdMaxSize_H |
| 0x111 | D_EPdMaxSize_L |
| 0x112 | D_EPdConfig |
| 0x113 | |
| 0x114 | D_EPdControl |
| 0x115 | |
| 0x116 | |
| 0x117 | |
| 0x118 | D_EPeMaxSize_H |
| 0x119 | D_EPeMaxSize_L |
| 0x11A | D_EPeConfig |
| 0x11B | |
| 0x11C | D_EPeControl |
| 0x11D | |
| 0x11E | |
| 0x11F | |

| | |
|---|---|
| 0x110 | D_EPdMaxSize_L |
| 0x111 | D_EPdMaxSize_H |
| 0x112 | D_EPdConfig |
| 0x113 | |
| 0x114 | D_EPdControl |
| 0x115 | |
| 0x116 | |
| 0x117 | |
| 0x118 | D_EPeMaxSize_L |
| 0x119 | D_EPeMaxSize_H |
| 0x11A | D_EPeConfig |
| 0x11B | |
| 0x11C | D_EPeControl |
| 0x11D | |
| 0x11E | |
| 0x11F | |

| | |
|---|---|
| 0x120 | D_DescAdrs_H |
| 0x121 | D_DescAdrs_L |
| 0x122 | D_DescSize_H |

| | |
|---|---|
| 0x120 | D_DescAdrs_L |
| 0x121 | D_DescAdrs_H |
| 0x122 | D_DescSize_L |

| | |
|---|---|
| 0x123 | D_DescSize_L |
| 0x124 | |
| 0x125 | |
| 0x126 | D_EP_DMA_Ctrl |
| 0x127 | |
| 0x128 | D_EnEP_IN_H |
| 0x129 | D_EnEP_IN_L |
| 0x12A | D_EnEP_OUT_H |
| 0x12B | D_EnEP_OUT_L |
| 0x12C | D_EnEP_IN_ISO_H |
| 0x12D | D_EnEP_IN_ISO_L |
| 0x12E | D_EnEP_OUT_ISO_H |
| 0x12F | D_EnEP_OUT_ISO_L |

| | |
|---|---|
| 0x123 | D_DescSize_H |
| 0x124 | |
| 0x125 | |
| 0x126 | D_EP_DMA_Ctrl |
| 0x127 | |
| 0x128 | D_EnEP_IN_L |
| 0x129 | D_EnEP_IN_H |
| 0x12A | D_EnEP_OUT_L |
| 0x12B | D_EnEP_OUT_H |
| 0x12C | D_EnEP_IN_ISO_L |
| 0x12D | D_EnEP_IN_ISO_H |
| 0x12E | D_EnEP_OUT_ISO_L |
| 0x12F | D_EnEP_OUT_ISO_H |

| | |
|---|---|
| 0x130 | |
| 0x131 | D_ModeControl |
| 0x132 | |
| 0x133 | |
| 0x134 | |
| 0x135 | |
| 0x136 | |
| 0x137 | |
| 0x138 | |
| 0x139 | |
| 0x13A | |
| 0x13B | |
| 0x13C | |
| 0x13D | |
| 0x13E | |
| 0x13F | |

| | |
|---|---|
| 0x130 | |
| 0x131 | D_ModeControl |
| 0x132 | |
| 0x133 | |
| 0x134 | |
| 0x135 | |
| 0x136 | |
| 0x137 | |
| 0x138 | |
| 0x139 | |
| 0x13A | |
| 0x13B | |
| 0x13C | |
| 0x13D | |
| 0x13E | |
| 0x13F | |

Blank addresses without indicated register names are reserved.

**Host registers (HOSTxDEVICE == 1)**

| | |
|---|---|
| 0x140 | H_SIE_IntStat_0 |
| 0x141 | H_SIE_IntStat_1 |
| 0x142 | — |
| 0x143 | H_FrameIntStat |
| 0x144 | H_CHrIntStat |
| 0x145 | H_CH0IntStat |

**Host registers (HOSTxDEVICE == 1)**

| | |
|---|---|
| 0x140 | H_SIE_IntStat_0 |
| 0x141 | H_SIE_IntStat_1 |
| 0x142 | — |
| 0x143 | H_FrameIntStat |
| 0x144 | H_CHrIntStat |
| 0x145 | H_CH0IntStat |

| | |
|---|---|
| 0x146 | H_CHaIntStat |
| 0x147 | H_CHbIntStat |
| 0x148 | H_CHcIntStat |
| 0x149 | H_CHdIntStat |
| 0x14A | H_CHeIntStat |
| 0x14B | |
| 0x14C | |
| 0x14D | |
| 0x14E | |
| 0x14F | |

| | |
|---|---|
| 0x146 | H_CHaIntStat |
| 0x147 | H_CHbIntStat |
| 0x148 | H_CHcIntStat |
| 0x149 | H_CHdIntStat |
| 0x14A | H_CHeIntStat |
| 0x14B | |
| 0x14C | |
| 0x14D | |
| 0x14E | |
| 0x14F | |

| | |
|---|---|
| 0x150 | H_SIE_IntEnb_0 |
| 0x151 | H_SIE_IntEnb_1 |
| 0x152 | — |
| 0x153 | H_FrameIntEnb |
| 0x154 | H_CHrIntEnb |
| 0x155 | H_CH0IntEnb |
| 0x156 | H_CHaIntEnb |
| 0x157 | H_CHbIntEnb |
| 0x158 | H_CHcIntEnb |
| 0x159 | H_CHdIntEnb |
| 0x15A | H_CHeIntEnb |
| 0x15B | |
| 0x15C | |
| 0x15D | |
| 0x15E | |
| 0x15F | |

| | |
|---|---|
| 0x150 | H_SIE_IntEnb_0 |
| 0x151 | H_SIE_IntEnb_1 |
| 0x152 | — |
| 0x153 | H_FrameIntEnb |
| 0x154 | H_CHrIntEnb |
| 0x155 | H_CH0IntEnb |
| 0x156 | H_CHaIntEnb |
| 0x157 | H_CHbIntEnb |
| 0x158 | H_CHcIntEnb |
| 0x159 | H_CHdIntEnb |
| 0x15A | H_CHeIntEnb |
| 0x15B | |
| 0x15C | |
| 0x15D | |
| 0x15E | |
| 0x15F | |

| | |
|---|---|
| 0x160 | H_NegoControl_0 |
| 0x161 | |
| 0x162 | H_NegoControl_1 |
| 0x163 | |
| 0x164 | H_USB_Test |
| 0x165 | |
| 0x166 | |
| 0x167 | |
| 0x168 | |
| 0x169 | |
| 0x16A | |
| 0x16B | |

| | |
|---|---|
| 0x160 | H_NegoControl_0 |
| 0x161 | |
| 0x162 | H_NegoControl_1 |
| 0x163 | |
| 0x164 | H_USB_Test |
| 0x165 | |
| 0x166 | |
| 0x167 | |
| 0x168 | |
| 0x169 | |
| 0x16A | |
| 0x16B | |

| | |
|---|---|
| 0x16C | |
| 0x16D | |
| 0x16E | |
| 0x16F | |

| | |
|---|---|
| 0x16C | |
| 0x16D | |
| 0x16E | |
| 0x16F | |

| | |
|---|---|
| 0x170 | H_CH0SETUP_0 |
| 0x171 | H_CH0SETUP_1 |
| 0x172 | H_CH0SETUP_2 |
| 0x173 | H_CH0SETUP_3 |
| 0x174 | H_CH0SETUP_4 |
| 0x175 | H_CH0SETUP_5 |
| 0x176 | H_CH0SETUP_6 |
| 0x177 | H_CH0SETUP_7 |
| 0x178 | |
| 0x179 | |
| 0x17A | |
| 0x17B | |
| 0x17C | |
| 0x17D | |
| 0x17E | H_FrameNumber_H |
| 0x17F | H_FrameNumber_L |

| | |
|---|---|
| 0x170 | H_CH0SETUP_0 |
| 0x171 | H_CH0SETUP_1 |
| 0x172 | H_CH0SETUP_2 |
| 0x173 | H_CH0SETUP_3 |
| 0x174 | H_CH0SETUP_4 |
| 0x175 | H_CH0SETUP_5 |
| 0x176 | H_CH0SETUP_6 |
| 0x177 | H_CH0SETUP_7 |
| 0x178 | |
| 0x179 | |
| 0x17A | |
| 0x17B | |
| 0x17C | |
| 0x17D | |
| 0x17E | H_FrameNumber_L |
| 0x17F | H_FrameNumber_H |

| | |
|---|---|
| 0x180 | H_CH0Config_0 |
| 0x181 | H_CH0Config_1 |
| 0x182 | |
| 0x183 | H_CH0MaxPktSize |
| 0x184 | |
| 0x185 | |
| 0x186 | H_CH0TotalSize_H |
| 0x187 | H_CH0TotalSize_L |
| 0x188 | H_CH0HubAdrs |
| 0x189 | H_CH0FuncAdrs |
| 0x18A | |
| 0x18B | H_CTL_SupportControl |
| 0x18C | |
| 0x18D | |
| 0x18E | H_CH0ConditionCode |
| 0x18F | |

| | |
|---|---|
| 0x180 | H_CH0Config_0 |
| 0x181 | H_CH0Config_1 |
| 0x182 | H_CH0MaxPktSize |
| 0x183 | |
| 0x184 | |
| 0x185 | |
| 0x186 | H_CH0TotalSize_L |
| 0x187 | H_CH0TotalSize_H |
| 0x188 | H_CH0HubAdrs |
| 0x189 | H_CH0FuncAdrs |
| 0x18A | |
| 0x18B | H_CTL_SupportControl |
| 0x18C | |
| 0x18D | |
| 0x18E | H_CH0ConditionCode |
| 0x18F | |

| | |
|---|---|
| 0x190 | H_CHaConfig_0 |
| 0x191 | H_CHaConfig_1 |
| 0x192 | H_CHaMaxPktSize_H |
| 0x193 | H_CHaMaxPktSize_L |
| 0x194 | H_CHaTotalSize_HH |
| 0x195 | H_CHaTotalSize_HL |
| 0x196 | H_CHaTotalSize_LH |
| 0x197 | H_CHaTotalSize_LL |
| 0x198 | H_CHaHubAdrs |
| 0x199 | H_CHaFuncAdrs |
| 0x19A | H_CHaBO_SupportCtl |
| 0x19B | H_CHaBO_CSW_RcvSize |
| 0x19C | H_CHaBO_OUT_EP_Ctl |
| 0x19D | H_CHaBO_IN_EP_Ctl |
| 0x19E | H_CHaConditionCode |
| 0x19F | |

| | |
|---|---|
| 0x190 | H_CHaConfig_0 |
| 0x191 | H_CHaConfig_1 |
| 0x192 | H_CHaMaxPktSize_L |
| 0x193 | H_CHaMaxPktSize_H |
| 0x194 | H_CHaTotalSize_HL |
| 0x195 | H_CHaTotalSize_HH |
| 0x196 | H_CHaTotalSize_LL |
| 0x197 | H_CHaTotalSize_LH |
| 0x198 | H_CHaHubAdrs |
| 0x199 | H_CHaFuncAdrs |
| 0x19A | H_CHaBO_SupportCtl |
| 0x19B | H_CHaBO_CSW_RcvSize |
| 0x19C | H_CHaBO_OUT_EP_Ctl |
| 0x19D | H_CHaBO_IN_EP_Ctl |
| 0x19E | H_CHaConditionCode |
| 0x19F | |

| | |
|---|---|
| 0x1A0 | H_CHbConfig_0 |
| 0x1A1 | H_CHbConfig_1 |
| 0x1A2 | H_CHbMaxPktSize_H |
| 0x1A3 | H_CHbMaxPktSize_L |
| 0x1A4 | H_CHbTotalSize_HH |
| 0x1A5 | H_CHbTotalSize_HL |
| 0x1A6 | H_CHbTotalSize_LH |
| 0x1A7 | H_CHbTotalSize_LL |
| 0x1A8 | H_CHbHubAdrs |
| 0x1A9 | H_CHbFuncAdrs |
| 0x1AA | H_CHbInterval_H |
| 0x1AB | H_CHbInterval_L |
| 0x1AC | |
| 0x1AD | |
| 0x1AE | H_CHbConditionCode |
| 0x1AF | |

| | |
|---|---|
| 0x1A0 | H_CHbConfig_0 |
| 0x1A1 | H_CHbConfig_1 |
| 0x1A2 | H_CHbMaxPktSize_L |
| 0x1A3 | H_CHbMaxPktSize_H |
| 0x1A4 | H_CHbTotalSize_HL |
| 0x1A5 | H_CHbTotalSize_HH |
| 0x1A6 | H_CHbTotalSize_LL |
| 0x1A7 | H_CHbTotalSize_LH |
| 0x1A8 | H_CHbHubAdrs |
| 0x1A9 | H_CHbFuncAdrs |
| 0x1AA | H_CHbInterval_L |
| 0x1AB | H_CHbInterval_H |
| 0x1AC | |
| 0x1AD | |
| 0x1AE | H_CHbConditionCode |
| 0x1AF | |

| | |
|---|---|
| 0x1B0 | H_CHcConfig_0 |
| 0x1B1 | H_CHcConfig_1 |
| 0x1B2 | H_CHcMaxPktSize_H |
| 0x1B3 | H_CHcMaxPktSize_L |
| 0x1B4 | H_CHcTotalSize_HH |

| | |
|---|---|
| 0x1B0 | H_CHcConfig_0 |
| 0x1B1 | H_CHcConfig_1 |
| 0x1B2 | H_CHcMaxPktSize_L |
| 0x1B3 | H_CHcMaxPktSize_H |
| 0x1B4 | H_CHcTotalSize_HL |

| | |
|---|---|
| 0x1B5 | H_CHcTotalSize_HL |
| 0x1B6 | H_CHcTotalSize_LH |
| 0x1B7 | H_CHcTotalSize_LL |
| 0x1B8 | H_CHcHubAdrs |
| 0x1B9 | H_CHcFuncAdrs |
| 0x1BA | H_CHcInterval_H |
| 0x1BB | H_CHcInterval_L |
| 0x1BC | |
| 0x1BD | |
| 0x1BE | H_CHcConditionCode |
| 0x1BF | |

| | |
|---|---|
| 0x1B5 | H_CHcTotalSize_HH |
| 0x1B6 | H_CHcTotalSize_LL |
| 0x1B7 | H_CHcTotalSize_LH |
| 0x1B8 | H_CHcHubAdrs |
| 0x1B9 | H_CHcFuncAdrs |
| 0x1BA | H_CHcInterval_L |
| 0x1BB | H_CHcInterval_H |
| 0x1BC | |
| 0x1BD | |
| 0x1BE | H_CHcConditionCode |
| 0x1BF | |

| | |
|---|---|
| 0x1C0 | H_CHdConfig_0 |
| 0x1C1 | H_CHdConfig_1 |
| 0x1C2 | H_CHdMaxPktSize_H |
| 0x1C3 | H_CHdMaxPktSize_L |
| 0x1C4 | H_CHdTotalSize_HH |
| 0x1C5 | H_CHdTotalSize_HL |
| 0x1C6 | H_CHdTotalSize_LH |
| 0x1C7 | H_CHdTotalSize_LL |
| 0x1C8 | H_CHdHubAdrs |
| 0x1C9 | H_CHdFuncAdrs |
| 0x1CA | H_CHdInterval_H |
| 0x1CB | H_CHdInterval_L |
| 0x1CC | |
| 0x1CD | |
| 0x1CE | H_CHdConditionCode |
| 0x1CF | |

| | |
|---|---|
| 0x1C0 | H_CHdConfig_0 |
| 0x1C1 | H_CHdConfig_1 |
| 0x1C2 | H_CHdMaxPktSize_L |
| 0x1C3 | H_CHdMaxPktSize_H |
| 0x1C4 | H_CHdTotalSize_HL |
| 0x1C5 | H_CHdTotalSize_HH |
| 0x1C6 | H_CHdTotalSize_LL |
| 0x1C7 | H_CHdTotalSize_LH |
| 0x1C8 | H_CHdHubAdrs |
| 0x1C9 | H_CHdFuncAdrs |
| 0x1CA | H_CHdInterval_L |
| 0x1CB | H_CHdInterval_H |
| 0x1CC | |
| 0x1CD | |
| 0x1CE | H_CHdConditionCode |
| 0x1CF | |

| | |
|---|---|
| 0x1D0 | H_CHeConfig_0 |
| 0x1D1 | H_CHeConfig_1 |
| 0x1D2 | H_CHeMaxPktSize_H |
| 0x1D3 | H_CHeMaxPktSize_L |
| 0x1D4 | H_CHeTotalSize_HH |
| 0x1D5 | H_CHeTotalSize_HL |
| 0x1D6 | H_CHeTotalSize_LH |
| 0x1D7 | H_CHeTotalSize_LL |
| 0x1D8 | H_CHeHubAdrs |
| 0x1D9 | H_CHeFuncAdrs |

| | |
|---|---|
| 0x1D0 | H_CHeConfig_0 |
| 0x1D1 | H_CHeConfig_1 |
| 0x1D2 | H_CHeMaxPktSize_L |
| 0x1D3 | H_CHeMaxPktSize_H |
| 0x1D4 | H_CHeTotalSize_HL |
| 0x1D5 | H_CHeTotalSize_HH |
| 0x1D6 | H_CHeTotalSize_LL |
| 0x1D7 | H_CHeTotalSize_LH |
| 0x1D8 | H_CHeHubAdrs |
| 0x1D9 | H_CHeFuncAdrs |

| | | | | |
|---|---|---|---|---|
| 0x1DA | H_CHeInterval_H | | 0x1DA | H_CHeInterval_L |
| 0x1DB | H_CHeInterval_L | | 0x1DB | H_CHeInterval_H |
| 0x1DC | | | 0x1DC | |
| 0x1DD | | | 0x1DD | |
| 0x1DE | H_CHeConditionCode | | 0x1DE | H_CHeConditionCode |
| 0x1DF | | | 0x1DF | |

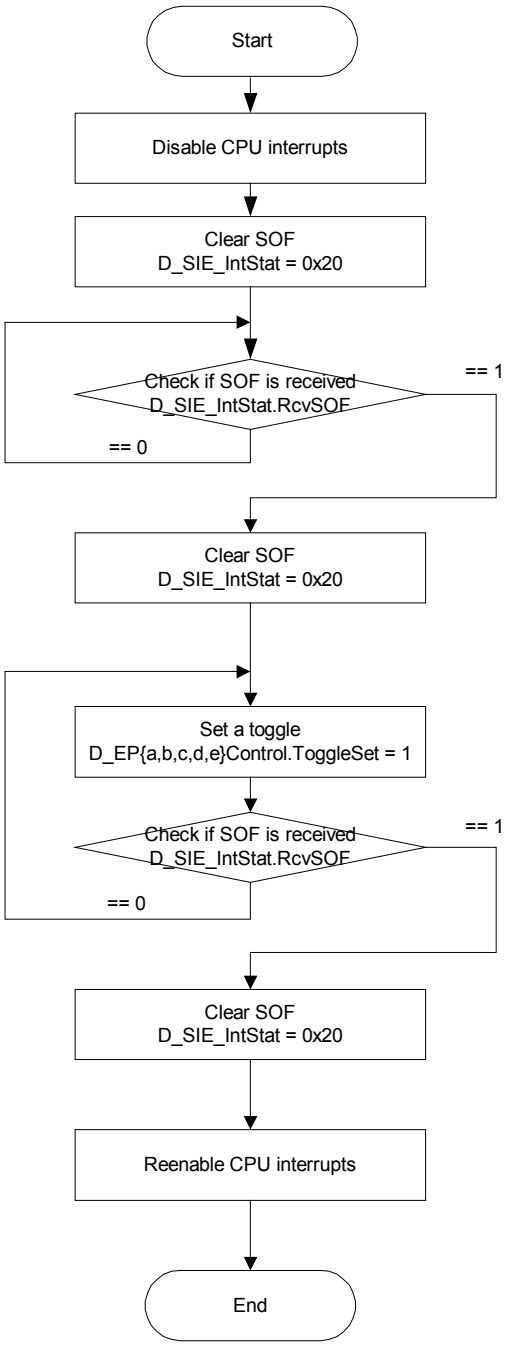
| | | | | |
|---|---|---|---|---|
| 0x1F5 | H_Protect | | 0x1F5 | H_Protect |
| 0x1F6 | H_Monitor | | 0x1F6 | H_Monitor |

Blank addresses without indicated register names are reserved.

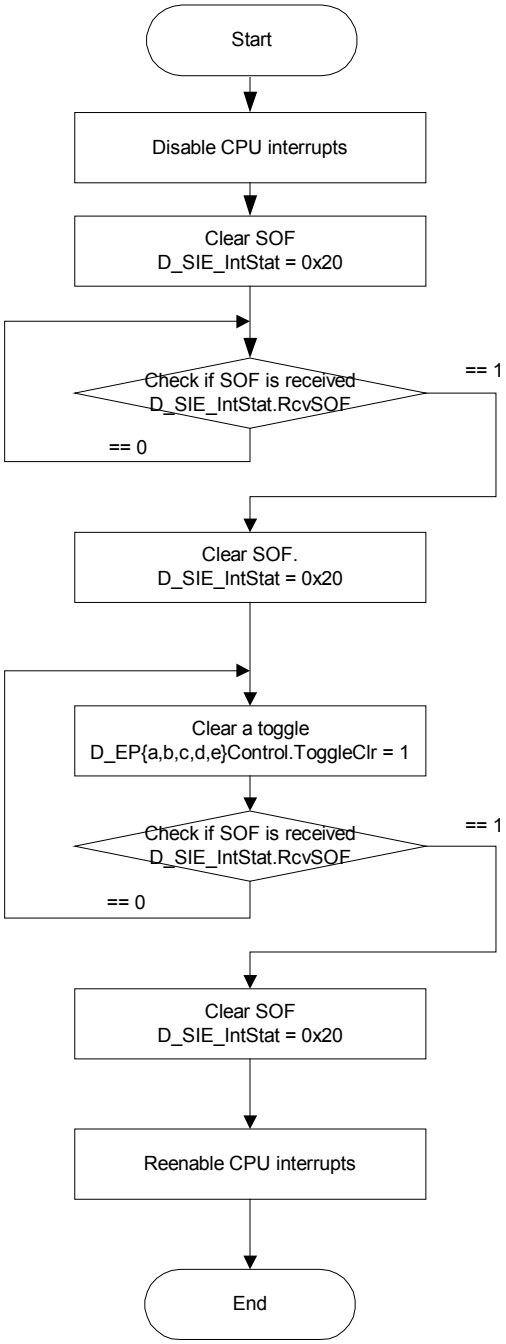
# Appendix B. Toggle Settings for Endpoint Changeover

In certain cases – e.g., when the host PC returns an NAK – endpoints will change over while a transaction is issued from the host PC to a general-purpose endpoint (EPa-e). In such cases, although a toggle must be set for the new endpoint to be used after the endpoint changeover, keep in mind that toggle settings made during transaction execution may be ignored, depending on the timing set. Follow the toggle setting operational flow shown below.

**Setting a toggle**




**Clearing a toggle**



# Appendix C. SUSPEND during HOST High-Speed Operation

If the USB bus is in a SUSPEND state while the S1R72V17's USB Host Port operates in HS mode, the disconnect detection function is unavailable. No H_SIE_IntStat.DetectDiscon interrupt status will be issued, even when the cable is disconnected.

* Here, the USB Host Port refers to the following:

USB Port when HostDeviceSel.HOSTxDEVICE == "1"

* The USB Host Port will operate in HS mode when the value of the following parameter is as shown below:

H_NegoControl_1.PortSpeed == "HS (0b00)"

This register is automatically set in hardware following execution of the auto-negotiation function.

* The USB bus will enter a SUSPEND state if the value of the following parameter is as shown below:

H_NegoControl_0.HostState == "USB_SUSPEND (0b110)"

This register is automatically set in hardware following execution of the auto-negotiation function.

Take one of the following corrective measures.

1. Avoid the SUSPEND state.

   Avoiding SUSPEND generally means keeping the bus active. For an "embedded" host, system power-savings and control may be facilitated by forcing the LSI into a disconnect state through software, rather than keeping it connected in SUSPEND. The S1R72V17 is designed so that the LSI can be placed in a power-save mode like SNOOZE or SLEEP when disconnected in software.

   If the LSI needs to be placed by the host in a disconnect state in software, switch off the power supply for the VBUS. Execute a state transition to H_NegoControl_0.HostState = "IDLE." This will negate output at the VBUSEN pin.

   Do the following to enter the IDLE state:

   - Set H_NegoControl_0.AutoModeCancel = "1".

   To do this, write "0x80" to H_NegoControl_0.

   - Wait until H_NegoControl_0.AutoModeCancel = 0.

   This change may take up to 100 ns.

   - Set H_NegoControl_0.AutoMode = "GoIDLE (0b0001)."

2. Enter a SUSPEND state after forcing the LSI into FS mode in firmware.

   Enable the disconnect detection function during a SUSPEND state by rewriting the value of H_NegoControl_1.PortSpeed to FS before placing the USB bus into SUSPEND state.

   With this method, keep in mind that firmware processing must be subject to time constraints to support Remote Wakeup from USB devices. When Remote Wakeup is detected, the hardware issues a RESUME signal. PortSpeed must be rewritten to "HS (0b00)" during this hardware processing.

   Do the following to enter a SUSPEND state:

   - Terminate issuance of all transactions.

   Do not issue any new transactions.

   - Set H_Protect.TranEnb = "STOP (0b01)."

   This stops issuance of SOF transactions. Write "0x01" to H_Protect.

   - Wait until H_Monitor.TranRunning = "0."

   Wait until issuance of SOF transactions stops. This may take up to 1 us.

   - Set H_Protect.PortSpeedWrEnb = "1."

   This allows rewriting of PortSpeed. To avoid rewriting other bits, access this register in read-modify-write mode.

   - Set H_NegoControl_1.PortSpeed = "FS (0b01)."
   - Clear H_Protect.PortSpeedWrEnb = "0."

   To avoid rewriting other bits, access this register in read-modify-write mode

   - Set H_NegoControl_0.AutoMode = "GoSUSPEND (0b1110)."

   Disconnect detection is performed in the manner described below.

   - A device disconnected interrupt (H_SIE_IntStat.DetectDiscon) is generated.
   - Set H_NegoControl_0.AutoModeCancel = "1."

   This clears the setting for GoSUSPENDtoOP. Write "0x80" to H_NegoControl_0.

   - Wait until H_NegoControl_0.AutoModeCancel = "0."

   This may take up to 100 ns.

   - Set H_NegoControl_0.AutoMode = "GoWAIT_CONNECTtoDIS (0b1001)."

   The LSI waits until a connection is detected.

   Do the following to perform RESUME:

   - Set H_Protect.PortSpeedWrEnb = "1."

Do this in read-modify-write mode.

- Set the operation speed in H_NegoControl_1.PortSpeed.
- Clear H_Protect.PortSpeedWrEnb = "0."

Do this in read-modify-write mode.

- Set H_NegoControl_0.AutoModeCancel = "1."

This clears the setting for GoSUSPENDtoOP. Write "0x80" to H_NegoControl_0.

- Wait until H_NegoControl_0.AutoModeCancel = "0."

This may take up to 100 ns.

- Set H_NegoControl_0.AutoMode = "GoRESUMEtoOP (0b1111)."

Do the following to respond to Remote Wakeup:

- A Remote Wakeup detected interrupt (H_SIE_IntStat_0.DetectRmtWkup) is generated.

Remote Wakeup is thereby detected.

- Set H_Protect.PortSpeedWrEnb = "1."

Do this in read-modify-write mode.

- Set the operation speed in H_NegoControl_1.PortSpeed.
- Clear H_Protect.PortSpeedWrEnb = "0."

Do this in read-modify-write mode.

When responding to Remote Wakeup, make sure the above processing is executed within 20 ms following Remote Wakeup detection.

Do the following to perform a RESET during RESUME:

- Set H_NegoControl_0.AutoModeCancel = "1." Write "0x80" to H_NegoControl_0.

This clears the setting for GoRESUMEtoOP.

- Wait until H_NegoControl_0.AutoModeCancel = "0."

This may take up to 100 ns.

- Set H_NegoControl_0.AutoMode = "GoRESETtoOP (0b1100)."

There are no departures from the conventional procedure.

3. Monitor LineState in firmware.

   Detect disconnections by polling H_USB_Status.LineState.

   If the USB device remains connected during SUSPEND, the following will be monitored.

   - H_USB_Status.LineState == "J (0b01)"

If the USB device is disconnected during SUSPEND, the following will be monitored.

- H_USB_Status.LineState == "SE0 (0b00)"

Note that of the registers included in the above procedures, the following have been additionally defined in Technical Manual Rev 1.30.

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1F5h | H_Protect | | 7: | 0: | 1: | 00h |
| | | | | 6: | 0: | 1: | |
| | | | | 5: | 0: | 1: | |
| | | | | 4: | 0: | 1: | |
| | | | R / W | 3: PortSpeedWrEnb | Enable to replace PortSpeed | | |
| | | | | 2: | 0: | 1: | |
| | | | R / W | 1: TranEnb[1] | Transaction Control | | |
| | | | R / W | 0: TranEnb[0] | | | |

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Host | 1F6h | H_Monitor | | 7: | | | 00h |
| | | | | 6: | | | |
| | | | | 5: | | | |
| | | | | 4: | | | |
| | | | | 3: | | | |
| | | | | 2: | | | |
| | | | | 1: | Monitor transaction | | |
| | | | R | 0: TranRunning | | | |

**1F5h.Bit7-4Reserved**

**1F5h.Bit3 PortSpeedWrEnb**

Writing to H_NegoControl_1.PortSpeed is enabled.

**1F5h.Bit2 Reserved**

**1F5h.Bit1-0TranEnb**

Transfer of SOF is stopped. Setting is completed before writing to H_NegoControl_1.PortSpeed.

**1F6h.Bit7-1Reserved**

**1F6h.Bit0 TranRunning**

SOF transfer stop is monitored.

# Appendix D. 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:<br>D_SETUP_Control.ProtectEP0 = 0<br>D_EP0Control.INxOUT = 1<br>D_EP0ControlIN = 0x40*<br>* 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 D_EP0SETUP0 and 1 registers* to determine the values of bmRequestType and bRequest. If bmRequestType==0 and bRequest==0x05, a SetAddress request is assumed.

* In register definitions of the S1R72V03, this is “RcvEP0SETUP.”

(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.

- D_SETUP_Control.ProtectEP0="0"
- D_EP0Control.INxOUT="1"
- D_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. 1.50. Bit 7 of the D_SUB_Address register is also defined in Development Specifications Rev. 1.50.

| Mode | Address | Register Name | R / W | Bit Symbol | Description | | Reset |
|---|---|---|---|---|---|---|---|
| Device | 131h | 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**

# Appendix E. Joining Endpoints/Channels to FIFO Areas

- Joining during USB device operations
  During USB device operations, the combinations for which endpoints can be joined to the FIFO areas are subject to limitations. When using endpoints, make sure they are joined to the appropriate FIFO areas as shown below. Do not join unused endpoints to FIFO areas.

| Endpoints used | FIFO areas to which joined |
|---|---|
| EP0 | AREA0 |
| EPa | AREA1 |
| EPb | AREA2 |
| EPc | AREA3 |
| EPd | AREA4 |
| EPe | AREA5 |

- Joining during USB host operations
  If the control transfer support and the bulk-only support functions are used during USB host operations, the possible combinations for which the channels can be joined to the FIFO areas are subject to limitations. When using channels, make sure they are joined to the appropriate FIFO areas, as shown below. Unused channels if any do not need to be joined to the FIFO areas.

| Channels used | FIFO areas to which joined |
|---|---|
| CH0 | AREA0 |
| CHa | AREA1 |
| CHb | AREA2, AREA3, AREA4, AREA5 |
| CHc | AREA2, AREA3, AREA4, AREA5 |
| CHd | AREA2, AREA3, AREA4, AREA5 |
| CHe | AREA2, AREA3, AREA4, AREA5 |

Note that if the control transfer support and the bulk-only support functions are not used, no limitations apply, and the channels can be joined to any desired FIFO area.

## Revision History

| Date | Content of Revision | | | | | |
|---|---|---|---|---|---|---|
| | Rev. | Page or section | Category | Content | | |
| 05/8/12 | 0.79 | All pages | New | Newly created. | | |
| 05/9/8 | 0.80 | - | Revision | Wholly revised (deletion is shown in blue and addition is shown in red). | | |
| 06/01/11 | 1.0 | | Revision | Description | Before correction | After correction |
| | | 2 | | Scope of application<br>Model name added | "S1R72V17B00A***/S1R72V17B00B***/S1R72V17F00A***" | "S1R72V17B00A***/ S1R72V17B00B***/ S1R72V17F00C***/ S1R72V17B00S***/ S1R72V17B00T***/ S1R72V17F00U***" |
| | | P1 | Addition | 1. Overview | This LSI incorporates host ports and device ports independently, | The host ports and device ports of this LSI are shared, |
| | | P2 | Addition | 2. Features<br><USB2.0 device functions><br><USB2.0 host functions><br><Other> Package type | Supports control, bulk, and interrupt transfers<br>Supports five general-purpose (Bulk, Interrupt, and Isochronous transfer) endpoints and Endpoint 0.<br>PFBGA5UX60(S1R72V17B00A***)<br><br>PFBGA8UX81(S1R72V17B00B***)<br><br>QFP14-80 (S1R72V17F00A***) | Supports control, bulk, interrupt, and isochronous transfers.<br>Includes five general purpose (Bulk and Interrupt transfer) channels<br>PFBGA5UX60(S1R72V17B00A***/ S1R72V17B00S***)<br>PFBGA8UX81(S1R72V17B00B***/ S1R72V17B00T***)<br>QFP14-80 (S1R72V17F00C***/ S1R72V17F00U***) |
| | | P6 | Correction | Pin Layout Diagram of the QFP14-80 package<br>Pin name corrected | TESTMODE | TESTEN |
| | | | Revision | Mode number shown in the pin layout diagram changed | S1R72V17F00A | S1R72V17F00C |
| | | P7 | | Pin Description | | |
| | | | | Description | GENERAL | OSC |
| | | | | XRESET | Described on "GENERAL" | Moved to "CPU I/F" (p. 8) |
| | | | Addition | TESTEN, ATPGEN, BURNIN | RESET: -<br>Pin Description: Test pin | RESET: (PD)<br>Pin Description: Test pin (fixed low) |
| | | | | VBUSFLG | RESET: - | RESET: (PU) |
| | | P9 | Addition | Book note below CPU I/F | - | The internal register can be set to operate the XINT pin in 1/0 mode or in Hi-Z/0 mode. |
| | | | | LVDD | Power supply for the internal circuit | Power supply for TEST I/O, power supply for OSC |
| | | | | CVDD | Power supply for the CPU I/O | Power supply for CPU interface I/O |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P67 | Correction | Errors in description corrected (2) | a bulk IN transaction | an IN transaction |
| | | | | Errors in description corrected (2) | a bulk OUT transaction | an OUT transaction |
| | | P125 – P129 | Revision | Paragraph changed | 6.5.1 FIFO Management<br>6.5.1.1 – 6.5.1.4 | 6.5.1. FIFO Memory Map<br>-<br>6.5.5 Method for Accessing the FIFO |
| | | P126 | Correction | Errors in description corrected (3) | {x=0, a-d} | {x=0, a-e} |
| | | P129 | Correction | Errors in description corrected (2) | JoinEPxCHx{x=0,a-d} | JoinEPxCHx{x=0,a-e} |
| | | P130 – P132 | Addition | Newly added | - | 6.6.2. Notes on Mode Switchover<br>6.6.2.1. When Using 16-bit BE Mode<br>6.6.2.2. Read access before the CPU_Config register is initialized |
| | | P203 | Correction | Errors in explanations of Bits7-0 corrected | ChipConfig | CPU_Config |
| | | P205 | Correction | Errors in the register name corrected | (Chip Configuration) | (CPU Configuration) |
| | | P304 | Revision | Explanation of Bit1 FIFO_Full<br>Explanation of Bit0 FIFO_Empty | H_CHx{x=a-e}Join.JoinFIFO_Stat<br>H_CHx{x=a-e}Join.JoinFIFO_Stat | AREAx{x=0-5}Join_0.JoinFIFO_Stat<br>AREAx{x=0-5}Join_0.JoinFIFO_Stat |
| | | P343, P354, P367, P377, P387, P397 | Correction | ○Errors in description corrected<br>Bit Symbol | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0] |
| | | P401 | Deletion | Absolute Maximum Ratings<br>Supply Voltage | CVDD*1<br>LVDD*2 | CVDD<br>LVDD |
| | | | Addition | *2 | TESTEN,ATPGEN, BURNIN | TESTEN,ATPGEN, BURNIN,XI |
| | | | Deletion | Recommended Operating Conditions<br>Supply Voltage | CVDD*1<br>LVDD*2 | CVDD<br>LVDD |
| | | | Addition | *2 | TESTEN,ATPGEN, BURNIN | TESTEN,ATPGEN, BURNIN,XI |
| | | P402 | Revision | Supply Voltage (TYP)<br>IDDH<br>IDDCH<br>IDCL<br>IDDL | <br>TBD<br>TBD<br>TBD<br>TBD | <br>7.8 mA<br>1.4 mA<br>0.7 mA<br>39.3 mA |
| | | | | Quiescent Current (MAX) | TBD | 300 uA |
| | | P404 | | Pin capacitance (MAX)<br>CI<br>CO<br>CIO1<br>CIO2 | <br>TBD<br>TBD<br>TBD<br>TBD | <br>8 pF<br>8 pF<br>8 pF<br>11 pF |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P403 | Correction | ○Errors in description corrected<br>Input Capacitance (LVCMOS)<br>VIH1<br>VIL1<br>Input Capacitance (LVCMOS)<br>VIH2<br>VIL2<br>VIH3<br>VIL3<br>Input Characteristics (Schmitt)<br>VT1+<br>VT1-<br>ΔV1 | MIN 1.22V<br>HVDD=1.65V MAX 0.62V<br><br>MIN 2.1V<br>MAX 0.9V<br>MIN 1.22V<br>MAX 0.62V<br><br>Condition: CVDD=1.95V to 3.6V<br>Condition: CVDD=1.65V to 3V<br>Condition: CVDD=1.65V to 3V | MIN 1.27V<br>LVDD = 1.65V MAX 0.57V<br><br>MIN 2.2V<br>MAX 0.8V<br>MIN 1.27V<br>MAX 0.57V<br><br>Condition: CVDD=3.6V<br>Condition: CVDD=3.0V<br>Condition: CVDD=3.0V |
| | | | Addition | ○Newly added<br>Input Characteristics (Schmitt)<br>VT2+<br>VT2-<br>ΔV2 | -<br>-<br>- | CVDD=1.95V MIN0.6V MAX1.4V<br>CVDD=1.65V MIN0.3V MAX1.1V<br>CVDD=1.65V MIN0.2V |
| | | | Correction | ○Errors in description corrected<br>Output Characteristics<br>Pin name: CD[15:0], etc.<br>VOH1<br>VOL1<br>VOH2<br>VOL2 | Condition: IOH=-2.6mA<br>Condition: IOL=2.7mA<br>Condition: IOH=-1.3mA<br>Condition: IOH=1.4mA | Condition: IOH=-2.0mA<br>Condition: IOL=2.0mA<br>Condition: IOH=-1.0mA<br>Condition: IOH=1.0mA |
| | | P404 | Correction | Output Characteristics<br>Pin name: VBUSEN<br>High Level Output Voltage<br>Low Level Output Voltage<br><br>○Errors in description corrected<br>Output Characteristics<br>IOZ<br>○Page of description moved<br>Pullup resistance<br>RPLO2H<br>RPLD1L<br>RPLD2L | Symbol: VOH2<br>Condition: IOH=-2.6mA<br>Symbol: VOL2<br>Condition: IOH=2.7mA<br><br><br><br>Pin name: CA[15:0]<br>Condition: CVDD=1.8V to 3.3V<br><br>Described on the page (p. 402) about “Output Characteristics” | Symbol: VOH3<br>Condition: IOH=-2.0mA<br>Symbol: VOL3<br>Condition: IOH=2.0mA<br><br><br><br>Pin name: CD[15:0]<br>Condition: CVDD=3.6V<br><br>Moved to the previous page (p. 403)<br>Standard values are not changed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P406 | | CPU and DMA I/F Access Timing<br>trcy min<br>tras min<br>trdf max<br>trdh min<br>trbh max<br>twcy min<br>twas min<br>trdn max<br>tdrn Description<br>tdaa Description<br>tdan Description | <br><br>75<br>35<br>25<br>1<br>5<br>75<br>35<br>50<br>XDREQ0/1 negate delay time<br>XDREQ0/1 setup time<br>XDREQ0/1 hold time | <br><br>80<br>40<br>35<br>3<br>9<br>80<br>40<br>35<br>XDREQ negate delay time<br>XDREQ setup time<br>XDREQ hold time |
| | | P410 | Addition | PFBGA5UX60 | TBD | New figure added |
| | | P411 | Addition | PFBGA8UX81 | TBD | New figure added |
| | | P412 | Addition | QFP14-80 | TBD | New figure added |
| | | P420 | Addition | Appendix A added | - | Connecting to Little Endian CPUs |
| | | P433 | Addition | Appendix B added | - | Toggle Settings for Endpoint Changeover |
| 06/01/23 | 1.1 | 1 | Revision | | | |
| | | 2 | | Scope of Application | This specification applies to the USB2.0 Controller “S1R72V17B00A***/S1R72V17B00B***/S1R72V17F00C***/S1R72V17B00S***/S1R72V17B00T***/S1R72V17F00U***” manufactured by the Semiconductor Operations Division of Seiko Epson Corporation. | This specification applies to the USB2.0 Controller “S1R72V17B00A***/S1R72V17B00B***/S1R72V17F00C***” manufactured by the Semiconductor Operations Division of Seiko Epson Corporation. |
| | | P2 | | Features<br>Package type | PFBGA5UX60 (S1R72V17B00A***/ S1R72V17B00S***)<br>PFBGA8UX81 (S1R72V17B00B***/ S1R72V17B00T***)<br>QFP14-80 (S1R72V17F00C***/ S1R72V17F00U***) | PFBGA5UX60 (S1R72V17B00A***)<br><br>PFBGA8UX81 (S1R72V17B00B***)<br><br>QFP14-80 (S1R72V17F00C***) |
| | | P410 | | Connection example for the PFBGA5UX60 Product type number in the diagram | S1R72V17B00A / S1R72V17B00S | S1R72V17B00A |
| | | P411 | | Connection example for the PFBGA8UX81 Product type number in the diagram | S1R72V17B00B/ S1R72V17B00T | S1R72V17B00B |
| | | P412 | | Connection example for the QFP14-80 Product type number in the diagram | S1R72V17F00C /S1R72V17F00U | S1R72V17F00C |

| | | | | | | |
|---|---|---|---|---|---|---|
| 06/02/10 | 1.2 | P144 | Revision | Description in 7, "Registers" changed | The registers in the S1R72V17 are classified into three groups: shared device/host registers, device registers, and host registers. The register maps between the device and host registers are switched by the HostDeviceSel.HOSTx DEVICE bit. When this bit = 0, the device register map is selected. When this bit = 1, the host register map is selected. Changing the setting for this bit does not clear values set in either register map. | The registers in the S1R72V17 are classified into three groups: shared device/host registers, device registers, and host registers. |
| | | P150-153 | | 7.2. Device Register | While the HostDeviceSel.HOSTx DEVICE bit = 0, the registers shown in ***bold face italic*** can be read and written even in the SLEEP state. All other registers can be read and written in the ACTIVE state. | The registers shown in ***bold face italic*** can be read and written even in the SLEEP state. All other registers can be read and written in the ACTIVE state. |
| | | P151 | | Register list<br>0x0EE D_FrameNumber_H<br>0x0EF D_FrameNumber_L | Reset<br>0x00<br>0x80 | Reset<br>0x80<br>0x00 |
| | | P154-158 | | 7.3. Host Register | While the HostDeviceSel.HOSTx DEVICE bit = 1, the registers shown in ***bold face italic*** can be read and written even in the SLEEP state. All other registers can be read and written in the ACTIVE state. | The registers shown in ***bold face italic*** can be read and written even in the SLEEP state. All other registers can be read and written in the ACTIVE state. |
| | | P155 | | Register list<br>0x17E H_FrameNumber_H<br>0x17F H_FrameNumber_L | Reset<br>0xFF<br>0x07 | Reset<br>0x07<br>0xFF |
| | | Table on page 173 | | 1: PM_State[1]<br>0: PM_State[0] | Power Management State<br>00: SLEEP,<br>01: (SNOOZE),<br>11: SLEEP | Power Management State<br>00: SLEEP,<br>01: (SNOOZE),<br>11: ACTIVE |
| | | (Formerly page 304) | | 142h | H_FIFO_IntStat(Host FIFO Interrupt Status) | Delete this page. (142H Reserved) (The page number of each page after p.273 needs to be reduced by 1.) |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P320 (Formerly page 321) | | Description of 152h H_FIFO_IntEnb register | This register is used to enable or disable the assertion of the H_FIFO_IntStat bit | This register is used to enable or disable the assertion of the FIFO_IntStat bit |
| | | P331 (Formerly page 332) | | 162h H_NegoControl_1 Precautions | None | Note: The Reset value of this register can only be read out when the power management state is ACTIVE. In other states, the Reset value will always read 00h. |
| | | P335 (Formerly page 336) | | 17Eh-17Fh H_FrameNumber_H / L<br>Reset value<br>Precautions | 8000h<br>None | 07FFh<br>Note: The Reset value of this register can only be read out when the power management state is ACTIVE. In other states, the Reset value will always read 0000h. |
| | | P336 (Formerly page 337) | | 180h H_CH0Config_0 Bit 4 changed | 4: ACK_Cnt[1] | 4: ACK_Cnt[0] |
| | | P336 (Formerly page 337) | | Description of Bit3-2 SpeedMode[1:0] | This setting is not required when using the control transfer support function. | Deleted<br>(also need to be set for the control transfer support function) |
| | | P346 (Formerly page 347) | | 190h H_CHaConfig_0 Bit 4 changed | 4: ACK_Cnt[1] | 4: ACK_Cnt[0] |
| | | P360 (Formerly page 361) | | 1A0h H_CHbConfig_0 Bit 4 changed | 4: ACK_Cnt[1] | 4: ACK_Cnt[0] |
| | | P370 (Formerly page 371) | | 1B0h H_CHcConfig_0 Bit 4 changed | 4: ACK_Cnt[1] | 4: ACK_Cnt[0] |
| | | P380 (Formerly page 381) | | 1C0h H_CHdConfig_0 Bit 4 changed | 4: ACK_Cnt[1] | 4: ACK_Cnt[0] |
| | | P390 (Formerly page 391) | | 1D0h H_CHeConfig_0 Bit 4 changed | 4: ACK_Cnt[1] | 4: ACK_Cnt[0] |
| | | P342 (Formerly page 343) | | 189h H_CH0FuncAdrs<br><br><br>Bit description | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3]<br>Bit5 EP_Number[3:0] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0]<br>Bit3-0 EP_Number[3:0] |
| | | P353 (Formerly page 354) | | 199h H_CHaFuncAdrs<br><br><br>Bit description | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3]<br>Bit5 EP_Number[3:0] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0]<br>Bit3-0 EP_Number[3:0] |
| | | P366 (Formerly page 367) | | 1A9h H_CHbFuncAdrs<br><br><br>Bit description | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3]<br>Bit5 EP_Number[3:0] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0]<br>Bit3-0 EP_Number[3:0] |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P376 (Formerly page 377) | | 1B9h H_CHcFuncAdrs<br><br><br>Bit description | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3]<br>Bit5 EP_Number[3:0] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0]<br>Bit3-0 EP_Number[3:0] |
| | | P386 (Formerly page 387) | | 1C9h H_CHdFuncAdrs<br><br><br>Bit description | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3]<br>Bit5 EP_Number[3:0] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0]<br>Bit3-0 EP_Number[3:0] |
| | | P396 (Formerly page 397) | | 1D9h H_CHeFuncAdrs<br><br><br>Bit description | 6: FuncAdrs[3]<br>5: FuncAdrs[3]<br>4: FuncAdrs[3]<br>Bit5 EP_Number[3:0] | 6: FuncAdrs[2]<br>5: FuncAdrs[1]<br>4: FuncAdrs[0]<br>Bit3-0 EP_Number[3:0] |
| | | P408 (Formerly page 409) | | CPU I/F Connection Examples<br>8-bit CPU connection | Data Bus 16Bit | Corrected to Data Bus 8 bits |
| 06/4/10 | 1.3 | P11 | Revision | 6.1.2 Correction | This function can be used in both SLEEP and ACTIVE states (see the section on Power Management). Since this function can be used in either USB device or host mode, it allows to detect a change of the state of the Downstream port (host port) during device mode or that of the Upstream port (device port) during host mode. | This function can be used in both SLEEP and ACTIVE states (see the section on Power Management). |
| | | | | 6.1.2.1 Correction | The following shows an example usage of VBUS pin change status and signal line change status. | Example of use of device port change status and host port change status are shown. |
| | | P103<br>P104<br>P105 | | 6.3.9.3.4.3 Correction<br>6.3.9.3.4.4 Correction<br>6.3.9.3.5 Correction | H_USB_Status.<br>PortSpeed[1:0] | H_NegoControl_1.<br>PortSpeed[1:0] |
| | | P185 | | Deletion | If no data exists for the relevant and endpoint or channel, this bit is not set until data is written to the endpoint or channel from the USB(not RdRemain Valid=0, however). If this bit=0, the value of RdRemain has no effect. | This description deleted because of error. |
| | | P302 | | Addition | | Description of Bit 3 Restrictions added to the description of DetectDiscon. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P306<br>P308<br>P310<br>P312<br>P314<br>P316 | | Addition | | Description of Bit6 Relation of H_CHx{x=0,a-e}Config_0.ACK_Cnt bit is added to the description of TranACK bit. |
| | | P332 | | Correction | Set one of the 5 low-order bits in this register to 1 and then EnHS_Test to 1. | Set any of the 5 low-order bits in this register. At the same time, set EnHS_Test to 1. |
| | | | | | Description of Bit 7 If when this bit is set to 1, any of the 5 low-order bits in the H_USB_Test register is set to 1. | If this bit and any of the 5 low-order bits in the H_USB_Test register are set to 1 simultaneously, the LSI enters the test mode corresponding to that bit. |
| | | | | | Description of Bit4,Bit3,Bit2,Bit1,Bit0 by setting this bit to 1 and then the EnHS_Test bit to 1. | Description of Bit4, Bit3, Bit2, Bit1, Bit0 by setting this bit and the EnHS_Test bit to 1 simulfaneously |
| | | P408<br>P409<br>p410 | | Correction of connection examples | LM3525M-H is used in the VUBS control circuit. | MAX8586ETA is used in the VUBS control circuit(including changes of notes) |
| | | | | | 10µF condensers are used between the HVDD terminals and VSS terminals. | 10µF condensers between the HVDD terminals and VSS terminals. |
| | | | | | | Varistors (electrostatic protective device) added to the DP/DM lines. |
| | | P158<br>P428 | | Addition | Appendix C added | 0x1F5 H_Protect register added |
| | | | | | | 0x1F6 H_Monitor register added |
| | | P275<br>P279<br>P283<br>P287<br>P291 | | Correction | Description of Bit 140 in EPx{x=a-e}MaxSize omitted | Added |
| 06/05/19 | 1.4 | P1 | Revision | Correction | Dual-power supply | Triple-power supply |
| | | P4 | | Addition | | Specify clock source of the PLL at clause 3.1 |
| | | P7 | | Correction | Reset value and Pin type of VBUS are not specified | Reset value and Pin type of VBUS are specified as PD. |
| | | P234 | | Correction | Title of the clause Clear AREA 1 Join 1 | Title of the clause Clear AREA n Join 1 |
| | | xii, P320 | | Deletion | H_FIFO_IntEnb is present at 152h | Fix it as Reserved |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P159 | | Correction | Description of bit 4 on the table<br>IDE Interrupts | Description of bit 4 on the table<br>FIFO Interrupts |
| | | P402 | | Addition | | Input Characteristics for Pulldown Resistance on VBUS pin is added |
| | | P331 | | Correction | Ability of bit 5 and 4 are "R" | Fixes it to "R / W" |
| | | P401 | | Addition | | Adds actual power consumption measurement value from examination |
| | | P405 | | Addition | | Adds a timing restriction "tsah" |
| | | P2 | | Deletion | | Unapplicable product name such as "S1R72V17B00S" and "S1R72V17F00U" on Package type feature are removed |
| | | vi, P171,<br>P172,<br>P182,<br>P203,<br>P204,<br>P205,<br>P207,<br>P235 | | Correction | Clause titles are not written in Italic | Fixes title to Italic |
| | | vi, P175 | | Correction | Clause titles are not written in Italic<br>Address 0124h/0125h is wrong | Fixes titles to Italic<br>Fixes address for WakeupTim_H to 014h<br>Fixes address for WakeupTim_L to 015h |
| | | vi, P176 | | Correction | Clause title is not written in Italic<br>Address 0126h is wrong | Fixes title to Italic<br>Fixes address to 016h |
| | | vi, P177 | | Correction | Clause title is not written in Italic<br>Address 0127h is wrong | Fixes title to Italic<br>Fixes address to 017h |
| | | vi, P179 | | Correction | Address 0128h is wrong | Fixes address to 018h |
| | | vi, P180 | | Correction | Address 0129h is wrong | Fixes address to 019h |
| | | P243,<br>P244 | | Correction | Register name for 0B9h and 0BA are wrong | Fixes them to D_EPdIntStat and D_EPeIntStat |
| | | P247 | | Correction | Register name in the table is wrong | Fix it to D_SIE_IntEnb |
| | | P301 | | Correction | Title is wrong | Correct title to D_EnEP_OUT_ISO_H and D_EnEP_OUT_ISO_L |

| | | | | | | |
|---|---|---|---|---|---|---|
| 06/07/10 | 1.5 | P3 | Revision | Correction | | Move CLKIN and XRESET on block diagram |
| | | P7, P9 | | Addition | | Pin Description<br>Pin Description of CLKIN, XI added |
| | | P12, P17, P159, P272, P274 | | Correction | HostIntStat<br>Host_IntStat<br>HostIntEnb<br>DeviceIntStat<br>Device_IntStat<br>DeviceIntEnb | USB_HostIntStat<br>USB_HostIntEnb<br>USB_DeviceIntStat<br>USB_DeviceIntEnb |
| | | P13, P222 | | Addition | | Description of AREA0 and EP0 joint Restrictions added |
| | | P14, P15, P224, P226, P228, P230, P232 | | Addition | | Description of AREA1-5 and EPa-e joint Restrictions added |
| | | P20 | | Correction | returns an ACK response | returns an ACK or a NYET response |
| | | P25, P271 | | Correction | D_FIFO_IntStat | D_EP0IntStat |
| | | P26, P128 | | Deletion | | Description of uninstalled FIFO_ByteWr deleted |
| | | P51, P67, P222 | | Addition | | Description of AREA0 and CH0 joint Restrictions added |
| | | P53, P72, P224 | | Addition | | Description of AREA1-5 and CHa joint Restrictions added |
| | | P124 | | Addition | | Description of joining the endpoint or channel to FIFO |
| | | P129 | | | EnEndpoint for the endpoint concerned is cleared | Endpoint concerned is not joined to any FIFO area |
| | | P148, P153, P295 | | Correction | | Correct bit width for AREAx{x=0-5}StartAdrs_H,L, AREAx{x=0-5}EndAdrs_H,L and DescAdrs_H |
| | | P159, P160, P162, P167, P168, P171 | | Correction | | Description of registers, bits corrected and show in bold face and italic |
| | | P151, P267 | | Addition | | Bit of D_USB_Address.SetAddress added |
| | | P153, P424 | | Addition | | D_ModeControl register added |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P158 | | Addition | | Description of Reserved added |
| | | P159 | | Correction | | Description of uninstalled GoCPU_Cut removed |
| | | P165 | | Addition | | Description for FIFO_DMA_Cmp added |
| | | P203 | | Correction | ClkSelect.ClkSelect bit | ClkSelect register |
| | | P236 | | Correction | | Snooze is replaced by Sleep |
| | | P255 | | Correction | EPeIntStat bit | D_EPeIntStat bit |
| | | P260 | | Correction | DTM | MTM |
| | | P261 | | Correction | EPx{x=a-e}Config.EnEndpoint | AREAx{x=1-5}Join_1.JoinEPxCHx{x=a-e} |
| | | P261 | | Correction | EPx{x=a-e}IntStat | D_EPx{x=a-e}IntStat |
| | | P271, P296 | | Correction | DescAdrs_H,L DescSize_H,L | D_DescAdrs_H,L D_DescSize_H,L |
| | | P272 | | Correction | RcvEP0SETUP bit in the MainIntStat register | RcvEP0SETUP bit in the USB_DeviceIntStat register |
| | | P276, P280, P284, P288, P292 | | Correction | D_EPaConfig_0 D_EPbConfig_0 D_EPcConfig_0 D_EPdConfig_0 D_EPeConfig_0 | D_EPaConfig D_EPbConfig D_EPcConfig D_EPdConfig D_EPeConfig |
| | | P279 | | Correction | 101h.Bit2-0 | 100h.Bit2-0 |
| | | P295, P296 | | Correction | | Address corrected |
| | | P296 | | Correction | EP0Control | D_EP0Control |
| | | P303 | | Addition | | Description of processing method of Interrupt Status added |
| | | P400 | | Addition | | Description of power-off procedure added |
| | | P401 | | Correction | | Quiescent Current value corrected |
| | | P404 | | Correction | | Oscillator input characteristics of ClkFreq setting corrected |
| | | P404 | | Addition | | Clock input characteristics added |
| | | P405, P406, P407 | | Addition | | Description of CPU and DMA I/F Access timing divided into standard cycle and specific mode cycle |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | P369, P370, P371 | | Addition | | Description of pin connection when USB Host is not used, and pin connection of CLKIN added<br>VBUS feed parts symbolized |
| | | P424 | | Correction | | Register names corrected<br>Description of Reserved added |
| | | P426, P427, P428 | | Correction | | H_CHx{x=b-e}Interval_ H,L shaded<br>Endian fixed |
| | | P428 | | Addition | | H_Protect, H_Monitor register added<br>Description of Reserved added |
| | | P434-P436 | | Addition | | Appendix D added |
| | | P437 | | Addition | | Appendix E added |
| 07/11/5 | 1.6 | P149 | Revision | Addition | | Description of Limitations on FIFO Access added |
| | | P457 | | Correction | VBUS | Pin name corrected<br>VBUSFLG |
| | | P457 | | Correction | VBUS | Pin names corrected<br>XCS, XRESET |
| | | P458 | | Correction | VOH2, VOL2 | Symbols corrected<br>VOH3, VOL3 |
| | | P459 | | Correction | min.11.999, max.12.001<br>min.23.998, max.24.002<br>min.47.996, max.48.004 | Limitations eased<br>min.11.9988, max.12.0012<br>min.23.9976, max.24.0024<br>min.47.9952, max.48.0048 |

EPSON

# International Sales Operations

## AMERICA

**EPSON ELECTRONICS AMERICA, INC.**
**HEADQUARTERS**
2580 Orchard Parkway
San Jose , CA 95131,USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

**SALES OFFICES**
**Northeast**
301 Edgewater Place, Suite 210
Wakefield, MA 01880, U.S.A.
Phone: +1-800-922-7667 FAX: +1-781-246-5443

## EUROPE

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

## 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: +886-2-8786-6660

**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 International Sales Group**
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 410457804
First Issue July 2007
Printed November 2007 in JAPAN ©