

***RK281X***  
***Technical Reference Manual***

Revision 0.1  
Feb 2010

## Revision History

Date	Revision	Description
2010-03-2	0.1	Initial Release

PRELIMINARY

# TABLE OF CONTENT

<b>TABLE OF CONTENT .....</b>	<b>3</b>
<b>FIGURE INDEX .....</b>	<b>9</b>
<b>TABLE INDEX .....</b>	<b>11</b>
<b>ACRONYM DESCRIPTIONS.....</b>	<b>12</b>
<b>CHAPTER 1 INTRODUCTION .....</b>	<b>13</b>
1.1 OVERVIEW .....	13
1.2 FEATURES .....	13
1.3 BLOCK DIAGRAM.....	20
<b>CHAPTER 2 PIN DESCRIPTION.....</b>	<b>22</b>
2.1 PIN PLACEMENT .....	22
<b>CHAPTER 3 SYSTEM CONFIGURATION .....</b>	<b>24</b>
3.1 BUS ARCHITECTURE .....	24
3.1.1 CPU system AHB Bus architecture.....	24
3.1.2 DSP system AHB Bus architecture .....	24
3.1.3 Data Path description for CPU and DSP .....	24
3.1.4 CPU ITCM/DTCM Application Notes .....	26
3.1.5 External Static/Dynamic Memory Interface Application Notes .....	27
3.2 SYSTEM ADDRESS MAP .....	28
3.2.1 System Memory Map for CPU .....	28
3.2.2 System Memory Map for DSP.....	29
3.3 SYSTEM MODE CONFIGURATION.....	30
3.3.1 Debug mode .....	30
3.3.2 CPU Boot mode .....	30
3.4 SYSTEM INTERRUPT CONNECTION.....	30
3.5 SYSTEM DMA HARDWARE REQUEST CONNECTION.....	32
<b>CHAPTER 4 STATIC/SDRAM MEMORY CONTROLLER.....</b>	<b>34</b>
4.1 DESIGN OVERVIEW .....	34
4.1.1 Overview.....	34
4.1.2 Features .....	34
4.2 ARCHITECTURE .....	34
4.2.1 Overview.....	35
4.2.2 Block Descriptions.....	35
4.3 REGISTERS .....	36
4.3.1 Registers Summary.....	36
4.3.2 Detail Register Description .....	36
4.4 FUNCTIONAL DESCRIPTION.....	44
4.4.1 Operation .....	44
<b>CHAPTER 5 STATIC/MOBILE SDRAM MEMORY CONTROLLER .....</b>	<b>52</b>
5.1 DESIGN OVERVIEW .....	52
5.1.1 Overview.....	52
5.1.2 Features .....	52
5.2 ARCHITECTURE .....	52
5.2.1 Overview.....	53
5.2.2 Block Descriptions .....	53
5.3 REGISTERS .....	54
5.3.1 Registers Summary.....	54
5.3.2 Detail Register Description .....	54
5.4 FUNCTIONAL DESCRIPTION .....	62
5.4.1 Operation .....	62
<b>CHAPTER 6 NAND FLASH CONTROLLER .....</b>	<b>70</b>
6.1 DESIGN OVERVIEW .....	70
6.1.1 Overview.....	70
6.1.2 Features .....	70

<b>CHAPTER 7 SD/MMC HOST CONTROLLER .....</b>	<b>71</b>
7.1 DESIGN OVERVIEW .....	71
7.1.1 Overview .....	71
7.1.2 Features .....	71
7.2 ARCHITECTURE .....	71
7.2.1 Block Diagram .....	71
7.2.2 Block Descriptions .....	71
7.3 REGISTERS .....	83
7.3.1 Registers Summary .....	83
7.3.2 Detail Register Description .....	83
7.4 FUNCTIONAL DESCRIPTION .....	94
7.4.1 Operation .....	94
7.4.2 Programming sequence .....	95
<b>CHAPTER 8 VIDEO INPUT PROCESSOR(VIP) .....</b>	<b>108</b>
8.1 DESIGN OVERVIEW .....	108
8.1.1 Overview .....	108
8.1.2 Features .....	108
8.2 ARCHITECTURE .....	108
8.2.1 Block Diagram .....	108
8.3 REGISTERS .....	108
8.3.1 Registers Summary .....	108
8.3.2 Detail Register Description .....	109
8.4 FUNCTIONAL DESCRIPTION .....	113
8.4.1 Operation .....	113
8.5 BYPASS FROM VIP TO LCDC FUNCTIONAL DESCRIPTION .....	116
<b>CHAPTER 9 LCD CONTROLLER.....</b>	<b>117</b>
9.1 DESIGN OVERVIEW .....	117
9.1.1 Overview .....	117
9.1.2 Features .....	117
9.2 ARCHITECTURE .....	118
9.2.1 Block Diagram .....	118
9.3 REGISTER DEFINITION .....	118
9.3.1 Registers Summary .....	118
9.3.2 Detail Register Description .....	120
9.4 FUNCTIONAL DESCRIPTION .....	134
9.4.1 Memory data formats .....	134
9.4.2 Virtual display .....	135
9.4.3 Rotation .....	137
9.4.4 De-interlace .....	137
9.4.5 Scaling .....	139
9.4.6 Overlay .....	141
9.4.7 Display Output .....	144
9.5 USE CASES AND TIPS .....	146
<b>CHAPTER 10 DW_DMA .....</b>	<b>147</b>
10.1 DESIGN OVERVIEW .....	147
10.1.1 Overview .....	147
10.1.2 Features .....	147
10.2 ARCHITECTURE .....	147
10.3 REGISTERS .....	147
10.3.1 Registers Summary .....	147
10.3.2 Configuration and Channel Enable Registers .....	149
10.3.3 Channel Registers .....	150
10.3.4 Interrupt Registers .....	160
10.4 REGISTER ACCESS .....	164
10.5 ILLEGAL REGISTER ACCESS .....	164
10.6 DW_DMA TRANSFER TYPES .....	164
<b>CHAPTER 11 INTERRUPT CONTROLLER (INTC) .....</b>	<b>168</b>
11.1 DESIGN OVERVIEW .....	168
11.1.1 Overview .....	168
11.1.2 Features .....	168

11.2 ARCHITECTURE .....	168
11.2.1 Block Diagram .....	168
11.3 REGISTERS .....	168
11.3.1 Registers Summary .....	168
11.3.2 Detail Register Description .....	169
11.4 FUNCTIONAL DESCRIPTION .....	174
11.4.1 Overview .....	174
11.4.2 Detail Description .....	174
<b>CHAPTER 12 HIGH-SPEED ADC INTERFACE .....</b>	<b>177</b>
12.1 DESIGN OVERVIEW .....	177
12.2 ARCHITECTURE .....	177
12.3 REGISTER .....	177
12.3.1 Register Summary .....	177
12.3.2 Detail Register Description .....	178
12.4 APPLICATION NOTES .....	180
<b>CHAPTER 13 HOST INTEFACE (HIF) .....</b>	<b>182</b>
13.1 DESIGN OVERVIEW .....	182
13.1.1 Overview .....	182
13.1.2 Features .....	182
13.2 ARCHITECTURE .....	182
13.2.1 Block Diagram .....	182
13.3 REGISTERS .....	182
13.3.1 Registers Summary .....	183
13.3.2 Detail Register Description .....	183
13.4 APPLICATION NOTES .....	184
<b>CHAPTER 14 USB OTG CONTROLLER .....</b>	<b>188</b>
14.1 DESIGN OVERVIEW .....	188
14.1.1 Overview .....	188
14.1.2 Features .....	188
<b>CHAPTER 15 SYSTEM CONTROL UNIT (SCU) .....</b>	<b>189</b>
15.1 DESIGN OVERVIEW .....	189
15.2 REGISTERS .....	189
15.2.1 Registers Summary .....	189
15.2.2 Detail Register Description .....	190
15.3 APPLICATION NOTES .....	198
15.3.1 PLL usage .....	198
15.3.2 Power mode management .....	199
15.3.3 HCLK frequency switch method .....	201
<b>CHAPTER 16 PMU IN CPU SYSTEM .....</b>	<b>202</b>
16.1 DESIGN OVERVIEW .....	202
16.1.1 Overviews .....	202
16.1.2 Features .....	202
16.2 POWER DOMAIN ARCHITECTURE .....	202
16.3 REGISTERS .....	203
<b>CHAPTER 17 UART .....</b>	<b>204</b>
17.1 DESIGN OVERVIEW .....	204
17.1.1 Overview .....	204
17.1.2 Features .....	204
17.2 ARCHITECTURE .....	204
17.2.1 Block Diagram .....	204
17.2.2 Block Descriptions .....	205
17.3 REGISTERS .....	205
17.3.1 Registers Summary .....	205
17.3.2 Detail Register Description .....	206
17.4 FUNCTIONAL DESCRIPTION .....	216
17.4.1 Operation .....	216
17.4.2 Programming sequence .....	219

<b>CHAPTER 18 SPI MASTER CONTROLLER</b>	<b>221</b>
18.1 DESIGN OVERVIEW	221
18.1.1 Overview	221
18.1.2 Features	221
18.2 ARCHITECTURE	221
18.2.1 Block Diagram	221
18.2.2 Block Descriptions	221
18.3 REGISTERS	222
18.3.1 Registers Summary	222
18.3.2 Detail Register Description	223
18.4 FUNCTIONAL DESCRIPTION	230
18.4.1 Operation	230
18.4.2 Programming sequence	231
<b>CHAPTER 19 SPI SLAVE CONTROLLER</b>	<b>233</b>
19.1 DESIGN OVERVIEW	233
19.1.1 Overview	233
19.1.2 Features	233
19.2 ARCHITECTURE	233
19.2.1 Block Diagram	233
19.2.2 Block Descriptions	233
19.3 REGISTERS	234
19.3.1 Registers Summary	234
19.3.2 Detail Register Description	235
19.4 FUNCTIONAL DESCRIPTION	241
19.4.1 Operation	241
19.4.2 Programming sequence	243
<b>CHAPTER 20 TIMERS IN CPU SYSTEM</b>	<b>244</b>
20.1 DESIGN OVERVIEW	244
20.1.1 Overview	244
20.1.2 Features	244
20.1.3 Block Diagram	244
20.2 REGISTERS	244
20.2.1 Registers Summary	244
20.2.2 Detail Register Description	245
20.3 FUNCTIONAL DESCRIPTION	248
20.3.1 Operation	248
20.3.2 Programming sequence	248
<b>CHAPTER 21 WATCHDOG TIMER (WDT)</b>	<b>249</b>
21.1 DESIGN OVERVIEW	249
21.1.1 Overview	249
21.1.2 Features	249
21.2 ARCHITECTURE	249
21.2.1 Block Diagram	249
21.2.2 Block Descriptions	249
21.3 REGISTERS	250
21.3.1 Registers Summary	250
21.3.2 Detail Register Description	250
21.4 FUNCTIONAL DESCRIPTION	252
21.4.1 Operation	252
21.4.2 Programming sequence	252
<b>CHAPTER 22 REAL TIME CLOCK (RTC)</b>	<b>254</b>
22.1 DESIGN OVERVIEW	254
22.1.1 Overview	254
22.1.2 Features	254
22.2 ARCHITECTURE	254
22.2.1 Block Diagram	254
22.3 REGISTERS	255
22.3.1 Registers Summary	255
22.3.2 Detail Register Description	255
22.4 FUNCTIONAL DESCRIPTION	259

22.4.1 Operation.....	259
<b>CHAPTER 23 I2C CONTROLLER.....</b>	<b>260</b>
23.1 DESIGN OVERVIEW .....	260
23.1.1 Overview .....	260
23.1.2 Features .....	260
23.2 ARCHITECTURE .....	260
23.2.1 Block Diagram.....	260
23.2.2 Block Descriptions.....	260
23.3 REGISTERS .....	261
23.3.1 Registers Summary.....	261
23.3.2 Detail Register Description .....	261
23.4 FUNCTIONAL DESCRIPTION .....	264
23.4.1 Operation.....	264
23.4.2 Programming sequence .....	269
<b>CHAPTER 24 I2S CONTROLLER .....</b>	<b>272</b>
24.1 DESIGN OVERVIEW .....	272
24.1.1 Overview .....	272
24.1.2 Features .....	272
24.2 ARCHITECTURE .....	272
24.2.1 Block Diagram.....	272
24.2.2 Block Descriptions.....	272
24.3 REGISTERS .....	273
24.3.1 Registers Summary.....	273
24.3.2 Detail Register Description .....	273
24.4 FUNCTIONAL DESCRIPTION .....	277
24.4.1 Operation.....	277
24.4.2 Programming sequence .....	279
<b>CHAPTER 25 PWM TIMER.....</b>	<b>282</b>
25.1 OVERVIEW .....	282
25.1.1 Key Features .....	282
25.2 ARCHITECTURE .....	282
25.2.1 Block Diagram.....	282
25.2.2 Block Descriptions.....	282
25.3 REGISTERS .....	282
25.3.1 Registers Summary.....	282
25.3.2 Detail Register Description .....	283
<b>CHAPTER 26 SAR-ADC CONTROLLER.....</b>	<b>285</b>
26.1 OVERVIEW .....	285
26.1.1 Key Features .....	285
26.2 ARCHITECTURE .....	285
26.2.1 Block Diagram.....	285
26.2.2 Block Descriptions.....	285
26.3 REGISTERS .....	285
26.3.1 Registers Summary.....	285
26.3.2 Detail Register Description .....	286
26.4 FUNCTION DESCRIPTION .....	286
<b>CHAPTER 27 GPIO IN CPU SYSTEM .....</b>	<b>287</b>
27.1 DESIGN OVERVIEW .....	287
27.1.1 Overview .....	287
27.1.2 Features .....	287
27.2 ARCHITECTURE .....	287
27.2.1 Block Diagram.....	287
27.2.2 Block Descriptions.....	287
27.3 REGISTERS .....	288
27.3.1 Registers Summary.....	288
27.3.2 Detail Register Description .....	288
27.4 FUNCTIONAL DESCRIPTION .....	292
27.4.1 Operation.....	292
27.4.2 Programming .....	294

<b>CHAPTER 28 GENERAL REGISTER FILE IN CPU SYSTEM.....</b>	<b>295</b>
28.1 OVERVIEW .....	295
28.2 REGISTERS .....	295
28.2.1 Registers Summary.....	295
28.2.2 Detail Registers Description.....	295
<b>CHAPTER 29 PORT MULTIPLEXER.....</b>	<b>309</b>
29.1 OVERVIEW .....	309
29.2 DETAILED DESCRIPTION FOR IO MUX.....	309
<b>CHAPTER 30 DDR2/MOBILE DDR SDRAM CONTROLLER.....</b>	<b>315</b>
39.1 DESIGN OVERVIEW .....	315
30.1.1 Overview .....	315
30.1.2 Features .....	315
30.2 ARCHITECTURE.....	316
30.2.1 Block Diagram.....	316
30.3 REGISTERS .....	316
30.3.1 Registers Summary.....	316
30.3.2 Detail Register Description .....	321
30.4 FUNCTIONAL DESCRIPTION .....	376
30.4.1 Initialization.....	376
30.4.2 Address Mapping .....	376
30.4.2.1 DDR SDRAM Address Mapping Options.....	376
30.4.2.2 Heterogeneous Memory Devices.....	376
<b>CHAPTER 31 HARDWARE INFORMATION.....</b>	<b>385</b>
31.1 OSCILLATOR CONNECTION .....	385
31.2 USB PHY CONNECTION .....	385
31.3 POWER UP SEQUENCE FOR POWER SUPPLY .....	386
31.4 POWER ON RESET DESCRIPTIONS .....	386
<b>CHAPTER 32 ELECTRICAL SPECIFICATION .....</b>	<b>387</b>
32.1 RECOMMENDED OPERATING CONDITIONS .....	387
32.2 DC CHARACTERISTICS .....	387
32.3 ABSOLUTE MAXIMUM RANGE .....	387
<b>APPENDIX A – ARM926EJS16K16K.....</b>	<b>388</b>
<b>APPENDIX B – DSP SYSTEM.....</b>	<b>389</b>



## Figure Index

Fig. 1-1 RK281x Block Diagram .....	21
Fig. 3-1 ITCM access timing with zero wait state .....	26
Fig. 3-2 DTCM access timing with zero wait state .....	26
Fig. 3-3 ITCM/DTCM access timing with one wait state .....	27
Fig. 3-4 External memory controller architecture .....	27
Fig. 3-5 System Memory Map for CPU .....	28
Fig. 3-6 Remap address description .....	29
Fig. 3-7 ITCM/DTCM address map .....	29
Fig. 3-8 System Memory Map for DSP .....	29
Fig. 4-1 Static Memory/SDRAM block diagrams .....	35
Fig. 4-2 SDRAM Page-Hit Single Write .....	44
Fig. 4-3 SDRAM Page-Miss Single Write .....	45
Fig. 4-4 SDRAM Page-Hit Busrt Write .....	46
Fig. 4-5 SDRAM Page-Hit Single Read .....	46
Fig. 4-6 SDRAM Page-Miss Single Read .....	47
Fig. 4-7 SDRAM Page-Hit Busrt Read .....	47
Fig. 4-8 Static Memory/SDRAM Controller power on sequence .....	48
Fig. 4-9 Static Memory/SDRAM Controller self_reflesh mode .....	50
Fig. 4-10 Static Memory/SDRAM Controller power_off mode .....	51
Fig. 5-1 Static Memory/Mobile SDRAM Controller Block Diagram .....	53
Fig. 5-2 Mobile SDRAM Page-Hit Single Write .....	63
Fig. 5-3 Mobile SDRAM Page-Miss Single Write .....	63
Fig. 5-4 Mobile SDRAM Page-Hit Busrt Write .....	64
Fig. 5-5 Mobile SDRAM Page-Hit Single Read .....	64
Fig. 5-6 Mobile SDRAM Page-Miss Single Read .....	65
Fig. 5-7 Mobile SDRAM Page-Hit Busrt Read .....	65
Fig. 5-8 Static Memory/Mobile SDRAM Controller power on sequence .....	66
Fig. 5-9 Mobile SDRAM self_reflesh flow .....	68
Fig. 5-10 Static Memory/Mobile SDRAM Controller power_off mode .....	69
Fig. 7-1 SD/MMC Host Controller Block Diagram .....	71
Fig. 7-2 SD/MMC Card Detect timing waveform .....	75
Fig. 7-3 SD/MMC Command Path State Machine .....	77
Fig. 7-4 SD/MMC Data Transmit State Machine .....	78
Fig. 7-5 SD/MMC Data Receive State Machine .....	80
Fig. 7-6 SD/MMC Initialization Sequence .....	95
Fig. 7-7 SD/MMC Command format for CMD52 .....	102
Fig. 8-1 RK281x VIP design architecture .....	108
Fig. 9-1 RK281x LCDC Block Diagram .....	118
Fig. 9-2 RK281x LCDC Frame buffer Data Format .....	134
Fig. 9-3 RK281x LCDC Hwc Data Format .....	135
Fig. 9-4 RK281x LCDC Hwc colors Look-up table .....	135
Fig. 9-5 RK281x LCDC Virtual display mode .....	136
Fig. 9-6 RK281x LCDC Virtual display roller mode .....	136
Fig. 9-7 RK281x LCDC Rotation .....	137
Fig. 9-8 RK281x LCDC De-interlace .....	138
Fig. 9-9 RK281x LCDC scaling down offset .....	140
Fig. 9-10 RK281x LCDC scaling up offset .....	140
Fig. 9-11 RK281x LCDC Interlace vertical filtering .....	141
Fig. 9-12 RK281x LCDC Overlay block diagram .....	141
Fig. 9-13 RK281x LCDC Overlay display .....	142
Fig. 9-14 RK281x LCDC Transparency color key .....	142
Fig. 9-15 RK281x LCDC Alpha blending .....	143
Fig. 9-16 RK281x LCDC Replicaion .....	143
Fig. 9-17 RK281x LCDC Dithering .....	144
Fig. 9-18 RK281x LCDC RGB LCD interface output timing .....	144
Fig. 9-19 RK281x LCDC MCU LCD interface output timing .....	145
Fig. 9-20 RK281x LCDC RGB delta LCD interface .....	145
Fig. 10-1 RK281x DW_DMA Architecture .....	147

Fig. 11-1 RK281x Interrupt Controller Architecture.....	168
Fig. 11-2 RK281x IRQ Interrupt Processing for INTC.....	174
Fig. 11-3 RK281x FIQ Interrupt Processing for INTC.....	176
Fig. 12-1 RK281x HS-ADC architecture diagram.....	177
Fig. 13-1 RK281x HIF block diagrams.....	182
Fig. 13-2 RK281x Timing Diagram for host interface.....	186
Fig. 15-1 RK281x clock architecture diagram.....	198
Fig. 15-2 RK281x system stop mode operation flow.....	200
Fig. 16-1 RK281x power domain architecture.....	203
Fig. 17-1 RK281x UART architecture diagram.....	205
Fig. 17-2 UART Serial protocol.....	217
Fig. 17-3 UART baud rate.....	217
Fig. 17-4 UART IrDA1.0 timing waveform.....	217
Fig. 17-5 UART Auto flow control block diagram.....	218
Fig. 17-6 UART Auto RTS timing waveform.....	219
Fig. 17-7 UART Auto CTS timing waveform.....	219
Fig. 17-8 Uart work flow in none fifo mode.....	219
Fig. 17-9 Uart fifo mode flow diagram.....	220
Fig. 18-1 RK281x SPI Master Controller Block diagram.....	221
Fig. 18-2 SPI Master Serial Format timing diagram(SCPH = 0).....	231
Fig. 18-3 SPI Master Serial Format timing diagram (SCPH = 1).....	231
Fig. 18-4 SPI Master transfer flow diagram.....	232
Fig. 19-1 RK281x SPI Slave Controller Block diagram.....	233
Fig. 19-2 SPI Slave Serial Format timing diagram (SCPH = 0).....	242
Fig. 19-3 SPI Slave Serial Format timing diagram (SCPH = 1).....	242
Fig. 19-4 SPI Slave transfer flow diagram.....	243
Fig. 20-1 Timers Block Diagram in CPU System.....	244
Fig. 20-2 Timers Ustage Flow in CPU System.....	248
Fig. 21-1 WDT Block Diagram.....	249
Fig. 21-2 WDT Operation Flow.....	253
Fig. 22-1 RK281x RTC design architecture.....	254
Fig. 23-1 I2C Controller design architecture.....	260
Fig. 23-2 I2C controller operation flow in Master/Transmitter mode.....	269
Fig. 23-3 I2C controller operation flow in Master/Receiver mode.....	270
Fig. 23-4 I2C controller operation flow in Slave/Transmitter mode.....	270
Fig. 23-5 I2C controller operation flow in Slave/Receiver mode.....	271
Fig. 24-1 RK281x I2S controller design architecture.....	272
Fig. 24-2 I2S Controller timing format for I2S interface.....	278
Fig. 24-3 I2S Controller timing format for Left-Justified interface.....	278
Fig. 24-4 I2S Controller timing format for Right-Justified interface.....	278
Fig. 24-5 I2S Controller TX operation flow chart.....	280
Fig. 24-6 I2S Controller RX operation flow chart.....	281
Fig. 25-1 PWM design architecture.....	282
Fig. 26-1 SAR-ADC Controller design architecture.....	285
Fig. 27-1 GPIO in CPU System Block Diagram.....	287
Fig. 27-2 GPIO in CPU System Interrupt RTL Block Diagram.....	293
Fig. 30-1 DDR2/mobile DDR SDRAM controller architecture.....	316
Fig. 30-2 RK281x DDR Various gate open timing waveform.....	381
Fig. 30-3 RK281x ddr_open_feedback signal connection diagram.....	382
Fig. 30-4 DDR2 Burst write operation waveform: RL = CL = 4, WL = 3, BL = 4.....	382
Fig. 30-5 DDR2 Burst read operation waveform: RL = CL = 4, BL = 4.....	382
Fig. 30-6 Mobile DDR Burst write operation waveform: BL = 4.....	383
Fig. 30-7 Mobile DDR Burst read operation waveform: CL = 3, BL = 4.....	383
Fig. 31-1 RK281x external oscillator connection diagram.....	385
Fig. 31-2 RK281x USB PHY connection diagram.....	385
Fig. 31-3 RK281x reset sequence timing waveform.....	386

## Table Index

Table 2-1 RK281x Pin Description .....	23
Table 3-1 Valid access path list for CPU and DSP .....	24
Table 3-2 RK281x Debug mode descriptions .....	30
Table 3-3 RK281x boot mode descriptions.....	30
Table 3-4 Interrupt sources connection for CPU .....	30
Table 3-5 Interrupt sources connection for DSP.....	31
Table 3-6 hardware request connection for DW_DMA .....	32
Table 3-7 hardware request connection for XDMA.....	33
Table 7-1 SD/MMC Bits in Interrupt Status Register .....	72
Table 7-2 SD/MMC Command Register Settings for No-Data Command .....	98
Table 7-3 SD/MMC Command Register Setting for Single-Block or Multiple-Block Read... 99	
Table 7-4 SD/MMC Command Register Settings for Single-Block or Multiple-Block Write 100	
Table 7-5 SD/MMC Parameters for CMDARG Registers .....	102
Table 7-6 SD/MMC CMDARG Bit Values .....	103
Table 7-7 SD/MMC Auto-Stop Generation condition list.....	106
Table 10-1 DW_DMA CTLx.SRC_MSIZ and DEST_MSIZ Decoding .....	155
Table 10-2 DW_DMA CTLx.SRC_TR_WIDTH and CTLx.DST_TR_WIDTH Decoding .....	155
Table 10-3 DW_DMA CTLx.TT_FC field Decoding .....	155
Table 10-4 DW_DMA PROTCTL field to HPROT Mapping .....	159
Table 10-5 DW_DMA Destination Scatter Register Description for Channel x.....	160
Table 10-6 Programming of Transfer Types and Channel Register Update Method .....	166
Table 13-1 Host interface address map table.....	184
Table 13-2 Pin mapping between HIF and LCDC interface.....	185
Table 29-1 RK281x IO MUX List .....	309

## Acronym descriptions

CXCLK	clock for DSP Core
XHCLK	clock for AHB bus inside DSP System
XPCLK	clock for APB bus inside DSP System
R	Read only
RW	Capable of both read and write
R/W	Capable of both read and write

PRELIMINARY

# Chapter 1 Introduction

## 1.1 Overview

RK281x is a highly-integrated , high-performance , low-power digital multimedia processor which is based on Dual Core(DSP+CPU) architecture with hardware accelerator . It is designed for high-end multimedia product applications such as PMP ,MID, AP, GPS and Mobile TV etc.

RK281x can support decode and encode for various types of video standards such as H.264/RMVB/MPEG-4/AVS/VC1/MPEG-2 by software and dedicated coprocessors. Specially, highest performance for video decode will reach fluent replay for video with H.264 @1280x720 format. RK281x also provides strong graphics/image ability with embedded GPU. By providing a complete set of peripheral interface, RK281x can support very flexible applications , including SDRAM/Mobile SDRAM/DDR2/Mobile DDR, Nor Flash, Nand Flash, LCD , Sensor, USB OTG 2.0/USB Host 1.0 ,SD/MMC/SDIO , Wi-Fi , High-speed ADC , I2C, I2S , UART , SPI , PWM etc.

This document will provide guideline on how to use RK281x correctly and efficiently. In them , the chapter 1 and chapter 2 will introduce the features, block diagram, signal descriptions and system configuration of RK281x, the chapter 3 through chapter 37 will describe the full function of each module in detail.

## 1.2 Features

- **System Operation**

- Dual Core Architecture (ARM9 + DSP) , including hardware accelerator
- Support system boot sequentially from ARM to DSP
- Support address remap function
- For two cores, all modules have unified address space
- Selectable JTAG debug method
  - ◆ ARM9 debug only (default)
  - ◆ DSP debug only
  - ◆ ARM9+DSP dual core debug
- Selectable CPU booting method
  - ◆ Boot from NOR Flash
  - ◆ Boot from Nand Flash
  - ◆ Boot from SPI nor flash
  - ◆ Boot from UART device
  - ◆ Boot from Host interface

- **Memory Organization**

- Internal memory space for ARM processor
  - ◆ Internal 16KB SRAM for ARM9 ICache
  - ◆ Internal 16KB SRAM for ARM9 DCache
  - ◆ Internal 8KB SRAM for ARM9 ITCM
  - ◆ Internal 16KB SRAM for ARM9 DTCM
- Internal memory space for DSP processor
  - ◆ Internal 96KB SRAM for DSP Instruction L1 Memory (also config as 32KB Memory+32KB ICache by software, another 32KB is switched by software)
  - ◆ Internal 64KB SRAM for DSP Data L1 Memory
  - ◆ Internal 48KB SRAM for DSP Instruction L2 Memory
  - ◆ Internal 32KB SRAM for DSP Data L2 Memory
- Embedded 8KB ROM for CPU Boot
- Embedded 4KB SRAM for communication between two cores
- Embedded 48KB SRAM for share among CPU,DSP and LCD rotator


- **Processors**
  - ARM926EJC
    - ◆ RISC architecture with 32bit ARM and 16bit Thumb instruction sets
    - ◆ Include efficient execution of Java byte codes
    - ◆ Built-in MMU to provide flexible memory management needed by many mainstream OS
    - ◆ Harvard cached architecture , separate ICache and DCache
    - ◆ Separate instruction and data TCM interfaces
    - ◆ Separate instruction and data AHB bus interface
    - ◆ Support ARM debug architecture
  - DSP
    - ◆ Based on VLIW instructions with SIMD concepts , reach high level of parallelism and high code density
    - ◆ Support 16bits and 32bits variable instruction sets
    - ◆ Based on a load/store architecture, have two load-store units
    - ◆ Support Nine-stage pipeline
    - ◆ Built-in two 16x16bit MAC units
- **Communication between two cores**
  - Support share memory and interactive interrupt method to complete communication
  - Processor Interface Unit (PIU)
    - ◆ Built-in three Command/reply protocols registers and three Semaphore registers to accessed by two cores
    - ◆ Support three semaphore-related interrupts and one command-reply-related interrupt between two cores
- **Clock & Power Management**
  - Three on-chip PLLs for ARM9 subsystem, DSP subsystem and Other logic
  - Support different DSP Core and internal AHB Bus clock ratio :  
1:1 , 1:2 , 1:3 , 1:4 , up to 1:16 mode
  - Support different DSP internal AHB Bus and internal APB Bus clock ratio :  
1:1 , 1:2 , 1:3 , 1:4 , up to 1:16 mode
  - Support different ARM9 core and AHB Bus clock ratio :  
1:1 , 1:2 , 1:3 and 1:4 mode
  - Support different ARM AHB Bus and ARM APB Bus clock ratio :  
1:1 , 1:2 and 1:4 mode
  - Max frequency of every key clock domain
    - ◆ 350MHz Max frequency for DSP Core
    - ◆ 330MHz Max frequency for ARM Core
  - 6 types of work modes by clock gating to save power :
    - ◆ Normal mode : Normal operating mode
    - ◆ Slow mode : Low frequency clock (24MHz) without PLL
    - ◆ Deep Slow mode : More Low frequency clock (32.768KHz) without PLL
    - ◆ Idle mode : The clock for only CPU is stopped ,  
wake up by any interrupts to CPU from idle mode
    - ◆ Sleep mode : The clock for only DSP is stopped ,  
wake up from sleep mode by some interrupts to DSP or register set from CPU
    - ◆ Stop mode : All clocks will be stopped , and SDRAM into  
self-refresh, all PLLs into power-down mode ,  
wake up from stop mode by external pin or RTC  
alarm interrupt
  - Support power supply shut down for 4 domain separately
- **Video hardware accellerator**
  - Deblocking



- ◆ Support RMVB and H.264 video format, max size is 1080p
- ◆ Support embedded DMA function with on-the-fly mode
- CABAC
  - ◆ Support H.264 video format, max size is 1080p
  - ◆ Support embedded DMA function
  - ◆ Tightly coprocessor in dsp system
- **Graphics hardware accellerator (GPU)**
  - 3D feature
    - ◆ Four times and 16 times Full Scene Anti-Aliasing (FSAA).
    - ◆ Lines, squares, triangles and points.
    - ◆ Flat and Gouraud shading.
    - ◆ Perspective correct texturing.
    - ◆ Point sampling, bilinear, and trilinear filtering.
    - ◆ Programmable mipmap level-of-detail biasing.
    - ◆ Multitexturing, with three textures.
    - ◆ Dot3 bump mapping.
    - ◆ Alpha blending.
    - ◆ Stencil buffering.
    - ◆ Point sprites.
    - ◆ 4-bit per texel texture compression, Ericsson Texture Compression (ETC)
  - 2D features
    - ◆ Lines, squares, triangles and points
    - ◆ ROP3/4
    - ◆ Arbitrary rotation and scaling
    - ◆ Alpha blending
    - ◆ Multitexture BitBLT
- **External Memory Interface**
  - Support SDRAM/Mobile SDRAM/DDRII/Mobile DDR separately
  - Support special SDRAM controller for high-performance video data transfer
  - Support Nor Flash/Nand Flash/SD/MMC/SDIO interface, Nor Flash interface is only available when use SDRAM or Mobile SDRAM
  - Static/SDRAM Memory controller
    - ◆ Dynamic memory interface support , including SDR-SDRAM and Mobile SDRAM
    - ◆ Asynchronous static memory device support including SRAM, ROM and Nor Flash with or without asynchronous page mode
    - ◆ Support 2 chip selects for (Mobile) SDRAM and 2 chip selects for static memory
    - ◆ Support 16bits or 32bits width data bus (Mobile) SDRAM and 8/16 bits data bus static memory, it is programmable.
    - ◆ Support industrial standard (Mobile) SDRAM with a maximum of 256MB of address space per chip select
    - ◆ 4Mbytes access space per static memory support
    - ◆ Support (Mobile) SDRAM and Static Memory power-down mode
    - ◆ Support (Mobile) SDRAM self-refresh mode
    - ◆ Programmable arbitration priority for 6 slave data ports
  - DDRII/Mobile DDR Memory controller
    - ◆ Programmable select for DDRII or Mobile DDR function

- ◆ Fully pipelined command, read and write data interface
- ◆ Advanced bank look-ahead features for high memory throughput
- ◆ Support one slave port for register set and 6 slave ports for data access
- ◆ Separate asynchronous FIFOs for every slave ports to support different frequency between AHB bus and DDR controller, and improve utility for bandwidth
- ◆ Support 32bit/16bit data width
- ◆ Support 2 chip selects , with a maximum of 256MB of address space per chip select
- ◆ DDRII data rate is 533M x 32bits , Mobile DDR data rate is 400M x 32bits
- Customized SDRAM controller for video
  - ◆ Support 32bit SDRAM data width
  - ◆ Special mechanism to improve little-block data transfer efficiency for video, especially when use together with MCDMA
  - ◆ Support one slave for register set and six data slave ports
- Nand Flash controller
  - ◆ Standard AMBA2.0 Slave interface
  - ◆ Support 8 chip selects for nand flash
  - ◆ Only support 8bit data width
  - ◆ Flexible CPU interface support
  - ◆ Embedded 2x1KB size buffer for DMA mode to improve performance
  - ◆ 512B 、 2KB 、 4KB page size support
  - ◆ Support hardware 24bit ECC
  - ◆ Support LBA nand
  - ◆ Support FF code auto correct process
- SD/MMC controller
  - ◆ Two Embedded SD/MMC Controllers, one is 4bit data bus , another is 8bit data bus
  - ◆ Compliant with SD Memory/SDIO with 1bit and 4bit data bus
  - ◆ Compliant with MMC V3.3 and V4.0 with 1/4/8bit data bus
  - ◆ Support combined single 32x32bits FIFO for both transmit and receive operations
  - ◆ Support FIFO over-run and under-run prevention by stopping card clock
  - ◆ Variable SD/MMC card clock rate 0 – 52 MHz which depends on AHB clock frequency
  - ◆ Controllable SD/MMC card clock to save power consumption
  - ◆ Support card detection and initialization , and write protection
  - ◆ Support transfer block size of 1 to 65365Bytes
  - ◆ DMA based or Interrupt based operation
- VIDEO/Image interface
  - Sensor controller
    - ◆ Embedded DMA function
    - ◆ Support 24MHz 、 48MHz 、 27MHz clock input
    - ◆ Support CCIR656 PAL/NTSC input
    - ◆ Support YUYV and UYVY format input
    - ◆ Support YUV 4:2:2 and YUV 4:2:0 format output
    - ◆ Programmable Hsync and Vsync porality
    - ◆ Support 10bit or 12bit raw data input
    - ◆ Support sensor bypass to LCDC interface
    - ◆ Support 8 MegaPixels
  - LCD controller
    - ◆ Embedded DMA function
    - ◆ Programmable transfer mode to meet different bus bandwidth and transfer efficiency.
    - ◆ Support two window with scale function
    - ◆ YUV422/YUV420/RGB565/RGB888 input format are supported in window0



- ◆ RGB565/RGB888 input format and 4 areas are supported in window1
- ◆ Support virtual display
- ◆ Built-in scaler engine from 1/8 to 8 
- ◆ Support 16 level alpha blending and transparent operation.
- ◆ Support Blank/Black Function
- ◆ Support LCD Panel resolution up to 1280x720
- ◆ Compatible with MCU panel
- ◆ Compatible with 8/16/18/24bits RGB Delta/no-Delta Panel
- ◆ Compatible with 8/16/18/24bits RGB Series/Parallel Output
- ◆ Support Interlace and Progressive Output
- ◆ Support hardware cursor
- ◆ Support rotation display
- ◆ Support video dither operation
- ◆ Support Interlace to progressive change for MPEG-2
- ◆ Support LCDC interface bypass from Host interface or VIP interface
- **DMA Controller**
  - Three DMA Controllers in chip
  - DW\_DMA Controller integrated inside ARM9 subsystem
    - ◆ Six DMA Channels support to use by audio , sd/mmc and system data transfer
    - ◆ 8 hardware request handshaking support
    - ◆ Support hardware and software trigger DMA transfer mode
    - ◆ Build-in 6 data FIFO : 64B/32B/16B/32B/16B/16B
    - ◆ Channel 0 & 1 support Scatter/Gather transfer
    - ◆ Channel 0 & 1 support LLP transfer
    - ◆ Two masters for on-the-fly support
    - ◆ The master interface only support defined length INCR transfer
  - 3D-DMA Controller(XDMA) integrated inside DSP subsystem
    - ◆ This DMA focus on data transfer for video process and mobile TV application
    - ◆ 16 configurable DMA channels , 4 channels support 3-dimensional data transfer
    - ◆ 8/16/32/64bit data transfer support and configurable burst length (INCR/INCR4/INCR8)
    - ◆ Programmable source and destination addresses with a post-modification option
    - ◆ Configurable external channel triggering (edge or level)
    - ◆ Support chaining-channels ,linked list-transfer and auto-channel initialization operating mode
    - ◆ Pause and resume operations supported to save power
    - ◆ Eight-stage memory buffer FIFO
  - MCDMA controller integrated inside ARM subsystem
    - ◆ This DMA focus on data transfer for video
    - ◆ Embedded DMA with one channel and three master interface
    - ◆ one master interface is directly for writing data to L1 Memory of DSP, which is fixed data transfer direction, two another master interface is general one
    - ◆ Support high-performance data transfer between ARM system and DSP system with asynchronous FIFO
- **Interrupt Controller**
  - Two Interrupt Controller in chip
  - DW\_INTC integrated inside ARM9 subsystem
    - ◆ Support 48 IRQ normal interrupt sources and 2 FIQ fast interrupt sources
    - ◆ Vectored interrupts support
    - ◆ Software interrupts support
    - ◆ Programmable interrupt priorities

- ◆ Fixed High Level sensitive triggered interrupts
- ICU (Interrupt Control unit) integrated inside DSP subsystem
  - ◆ 48 interrupt sources , each may be linked to different interrupt inputs for DSP core
  - ◆ Software triggering to all 48 interrupt sources
  - ◆ Configurable source interrupt polarity (low/high)
  - ◆ External interrupt source with software configuration to edge/level sensitive
- **USB interface**
  - USB OTG 2.0 interface
    - ◆ Complies with the OTG Supplement to the USB2.0 Specification
    - ◆ Operates in High-Speed and Full-Speed mode
    - ◆ Support Session Request Protocol(SRP) and Host Negotiation Protocol(HNP)
    - ◆ Support 6 channels in host mode
    - ◆ 6 endpoints , 3 in and 3 out
    - ◆ Built-in one 1777 x 35bits FIFO
  - USB HOST 1.0 interface
    - ◆ Complies with the USB1.1 Specification
    - ◆ Operates in Full-Speed mode
    - ◆ Operates in host mode
    - ◆ Support 2 channels in host mode
    - ◆ Built-in one 70 x 35 bits FIFO
- **High-speed ADC interface**
  - Max frequency is 64MHz
  - Standard AMBA2.0 Slave interface
  - Dual 8/10 bits A/D converter Interface
  - Support 2bit data bus from GPS tuner
  - Support TS stream data transfer
- **HOST interface**
  - Programmable 8bit/16bit data width
  - Embedded 4KB dual-port buffer for data transfer
  - Compatible with MCU interface timing
  - Interrupt request for data exchange
  - Support Host interface function disable
  - Support address self-increment when accessing buffer by MCU interface
  - Support LCD bypass function with 18bit data bus, ,bypass IO mapping relationship is programmable (totally 4 types)
  - Software or hardware control for LCD bypass enable
- **Low\_speed Peripheral interface**
  - Serial Peripheral Interface (SPI) Master Controller
    - ◆ Support two slave devices connection
    - ◆ Compatible with Motorola SPI , TI Synchronous Serial Protocol or National Semiconductor Microwire interface
    - ◆ Dynamic control of serial bit rate of data transfer by programmable sclk\_out frequency, which is half of PCLK in max mode
    - ◆ FIFO depth for transmit and receive are also 32x16bits
    - ◆ Programmable data item size ,from 4 to 16bits
    - ◆ Support DMA based and interrupt based operation
  - Serial Peripheral Interface (SPI) Slave Controller
    - ◆ Compatible with Motorola SPI , TI Synchronous Serial Protocol or National

- ◆ Semiconductor Microwire interface
- ◆ Dynamic control of serial bit rate of data transfer by sclk\_in from master device
- ◆ FIFO depth for transmit and receive are also 32x16bits
- ◆ Programmable data item size ,from 4 to 16bits
- ◆ DMA based and interrupt based operation
- UART
  - ◆ 4 UART support in chip
  - ◆ UART0 support modem function and IrDn function
  - ◆ UART1 support IrDA 1.0 SIR mode and Serial data transfer without auto flow-control function
  - ◆ UART2 only for Serial data transfer without auto flow-control function
  - ◆ UART3 only for Serial data transfer with auto flow-control function
  - ◆ Based on the 16550 industry standard
  - ◆ Programmable serial data baud rate, up to 3Mbps baud-rate , and main clock is 48MHz in max mode
  - ◆ Programmable baud rate generator. This enables division of the internal clock by (1 ~ 65535 x 16) and generates an internal x16 clock
  - ◆ Standard asynchronous communication bits (start, stop and parity).
  - ◆ DMA based and interrupt based operation
  - ◆ FIFO depth for data transfer is always 32x8bits
  - ◆ For UART1, In IrDA SIR mode, support configurable baud data rate up to 115.2K and a pulse duration as specified in the IrDA physical layer specification
- I2C controller
  - ◆ 2 I2C controllers integrated in chip
  - ◆ Multi masters operation support
  - ◆ Software programmable clock frequency and transfer rate up to 100Kbit/s in standard mode or up to 400Kbit/s in Fast mode
  - ◆ Supports 7 bits and 10 bits addressing modes
- I2S
  - ◆ Support mono/stereo audio file
  - ◆ Support audio resolution: 8, 16 bits
  - ◆ Support audio sample rate from 32KHz to 96 KHz
  - ◆ Support I2S, Left-Justified and Right-Justified digital serial data format
- PWM
  - ◆ Built-in three 32 bit timer modulers
  - ◆ Programmable counter
  - ◆ Chained timer for long period purpose
  - ◆ 4-channel 32-bit timer with Pulse Width Modulation (PWM)
  - ◆ Programmable duty-cycle, and frequency output
- General Purpose IO (GPIO)
  - ◆ Support 96 individually programmable input/output pins
  - ◆ 16 GPIOs with external interrupt capability
- Timers in CPU system
  - ◆ Built-in Three 32 bits timer modules
  - ◆ Support for two operation modes : free-running and user-defined count
- Timers in DSP system
  - ◆ Built-in two 32 bits timer modules
  - ◆ Support for 5 various counting modes : Single Count mode, Auto-restart mode , Free-running , Event Count mode and Watchdog Timer mode
  - ◆ Pulse Width Modulation(PWM) mechanism
  - ◆ Three possible input clock signals: internal , external and cascaded
- Watchdog Timer (WDT)
  - ◆ Watchdog function (Generate a system reset or an interrupt)
  - ◆ Built-in 32 bits programmable counter
- Real Time Clock (RTC)
  - ◆ Support perpetual RTC core power

- ◆ Programmable alarm with interrupt for system power wake up
- ◆ System power off sequence with output control pin
- **Analog IP interface**
  - ADC Converter
    - ◆ 4-channel single-ended 10-bit 1MSPS Successive Approximation Register (SAR) analog-to-digital converter
    - ◆ No off-chip components required
    - ◆ DNL less than +/-1 LSB , INL less than +/-1.5 LSB
    - ◆ Supply 2.8V to 3.6V for analog interface
  - eFuse
    - ◆ 64-bit serial eFuse macro
    - ◆ Be programmed one bit at a time, but all 64bits can be read at the same time.
    - ◆ 2.9V (+/-200mV) & 2.5V(+/-50mV) Programm voltage
    - ◆ 3.3V & 0V Sense voltage
- **Operation Temperature Range**
  - -40°C to +125°C
- **Operation Voltage Range**
  - Core: 1.2V
  - I/O : 3.3V/2.5V/1.8V
    - ◆ USB OTG : 2.5V/3.3V
    - ◆ LCD/VI/NandC : 1.8V/3.3V programmable
    - ◆ DDRII/Mobile DDR/Mobile SDRAM : 1.8V
    - ◆ Others : 3.3V)
- **Process**
  - Chartered 65nm LP
- **Package Type**
  - BGA (TBD)
- **Power**
  - TBD

### 1.3 Block Diagram

The following figure shows block diagram of RK281x.

RK281x can be divided into two sub system : DSP System and CPU System.

- DSP System
  - XDMA : three-dimensional DMA , used to data transfer for video decoder or other algorithm
  - High-Speed ADC Interface : focus on completing data reveiver from tuner in DVB-T,DAB, T-DMB,GPS application with software method.
  - Video hardware accelerator: work together with software algorithm to generate high-performance H.264 video decoder
  - ICU : Interrupt controller for DSP processor
  - PIU : processor interface unit, used to complete communication between DSP and CPU
  - PMU : power management unit, used to control clock and reset to save power for modules inside DSP system
  - General reg file : focus on general control on DSP system by software method, composed of some register groups
- CPU System
  - DW\_DMA : used to data transfer for audio and low-speed peripheral

- MC\_DMA: used for general data transfer, especially one data path from external memory directly to DSP L1 DMEM to improve video decoder efficiency
- SCU : focus on clock gating , clock frequency switch, reset control , power on/off and system mode switch for CPU system to save power
- PMU : used to complete power on/off switch control for RK281x
- INTC : Interrupt controller for CPU processor
- General reg file : focus on general control on CPU system by software method, composed of some register groups, including IO mux control, IO PAD pull up/down control and other system control signals .

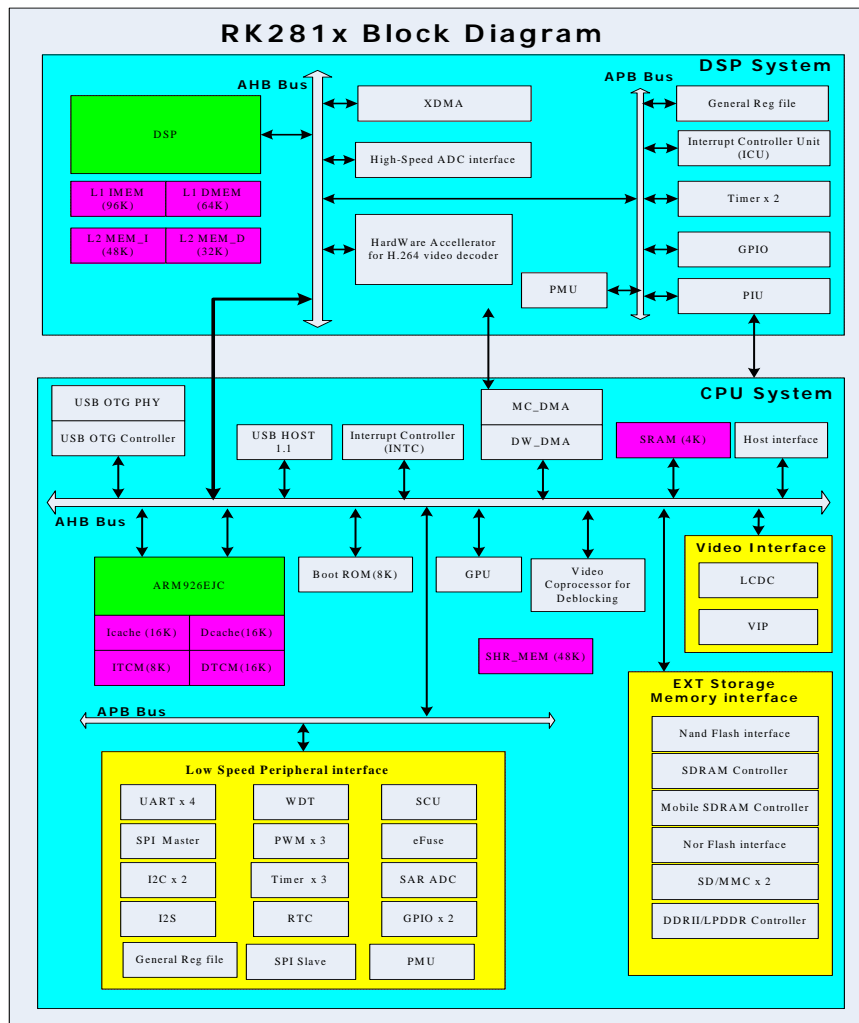


Fig. 1-1 RK281x Block Diagram

## Chapter 2 Pin Description

## 2.1 PIN Placement

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
A	MEM_D0	LCD_D6	PD3/LCD_D11	PC6/LCD_D22	PC5/LCD_D21	PD4/LCD_D12	PC0/LCD_D16	LCD_HSY_NC	SM_A15	TDO	PA3	PA2	PB1/SM_CS1/SDMMC0_PCA	USB11_D_VDD	USB11_D_M	RTC_DVD_D33	RTC_AVDD	VIP_CLKI	A
B	MEM_D3	MEM_D1	LCD_D4	LCD_D3	LCD_D7	PD1/LCD_D9	PD6/LCD_D14	LCD_DEN	LCDBP	TCK	ST0_CSN	VIP_D6	PF6/VIP_CLKO	USB11_D_VSS	USB11_D_P	RTCINT_OUT	PWR_GO_OD	VCC_VIP_1	B
C	MEM_DQ_S0P	MEM_DQ_S0M	MEM_D2	PC2/LCD_D18	LCD_D5	PD0/LCD_D8	PD7/LCD_D15	LCD_DCLK	SM_A10	OPMODE1	TDI	VCC_VIP_2	VIP_VSYN_C	PH7/VIP_LD3	PF5/PWM3/VIP_LD1	PE0/VIP_LD0	XIN32K	XOUT32K	C
D	MEM_D9	MEM_D7	MEM_D4	MEM_DM0	LCD_D0	LCD_D2	PD2/LCD_D10	PC1/LCD_D17	RTCK	TMS	PF4/PWM2/SDMMC0_WP	VIP_D7	VIP_D4	PH6/VIP_LD2	HOST_D0	HOST_D4	VSSA_CO_DECPLL	VDDA_CO_DECPLL	D
E	MEM_D11	MEM_D8	MEM_DM1	MEM_D6	MEM_VSS_Q0	PC3/LCD_D19	VCC_LCD_2	PD5/LCD_D13	OPMODE0	TRSTN	VIP_D3	VIP_D5	VIP_D2	HOST_D2	HOST_D6	AP2BB_INT	VSSA_AR_MPLL	VDDA_AR_MPLL	E
F	MEM_DQ_S1P	MEM_DQ_S1M	MEM_D5	MEM_D10	MEM_A14	MEM_VD_DQ0	PC4/LCD_D20	LCD_D1	LCD_VSYN_C	VCC_LCD_1	VIP_D1	VIP_D0	HOST_D1	HOST_A0	HOST_D7	HOST_A1	VSSA_DS_PPLL	VDDA_DS_PPLL	F
G	MEM_D14	MEM_D12	MEM_D15	MEM_D13	MEM_A13	MEM_VSS_Q1	VDDCORE	VDDCORE	VCCIO	PC7/LCD_D23	PF2/PWM0	VIP_HREF	HOST_D3	EXTDDR_SEL	NPOR	EXTMSDR_SEL	EWAKEUP_STOP	TEST	G
H	MEM_A9	MEM_A12	MEM_A6	MEM_A5	MEM_VD_DQ1	GND	GND	GND	GND	VCCIO	VCCIO	HOST_D5	EXTCLK	EWAKEUP_POWER	BTMODE	HSADC_I_D1	HSADC_Q_D1/HOST_D9	HSADC_I_D0	H
J	MEM_A3	MEM_A4	MEM_A8	MEM_VD_DQ2	VDDCORE	GND	GND	GND	GND	VDDCORE	VCCIO	HSADC_Q_D0/HOST_D8	HSADC_Q_D2/HOST_D10	HSADC_I_D5	HSADC_Q_D4/HOST_D12	HSADC_I_D3	HSADC_Q_D5/HOST_D13	HSADC_I_D2	J
K	MEM_A1	MEM_A2	MEM_A7	MEM_VSS_Q2	VDDCORE	GND	GND	GND	GND	VDDCORE	VCCIO	HSADC_Q_D7/HOST_D15	HOST_RDN	HSADC_Q_D9/SM_OEN	HSADC_I_D7	HSADC_I_D9	HSADC_Q_D8/SM_WEN	HSADC_Q_D3/HOST_D11	K
L	MEM_CLK	MEM_CLK_N	MEM_A11	MEM_A10	MEM_VREF	GND	GND	GND	GND	VCCIO	PB7/SPI0_RXD/SDMMC0_D7	PE6/UART1_SIR_IN/I2C1_SDA	PF0/UART1_RX/CX_T0_PWM	GPS_CLK/HSADC_CLKOUT	PB3/UART0_RTSN	HSADC_I_D6	HOST_CSN	HSADC_I_D4	L
M	MEM_A0	MEM_BA2	MEM_CSN0	MEM_RAS_N	MEM_VSS_Q3	MEM_VSS_Q4	VDDCORE	VDDCORE	FLASH_D7	FLASH_RDY	PG5/SDMMC1_D2	PB5/SPI0_CLKO/SDMMC0_D5	HSADC_I_D8	PA0/HOST_D16	PF1/UART1_TX/CX_T1_PWM	PG0/UART0_RX/SDMMC1_DET	HSADC_Q_D6/HOST_D14	XOUT24M	M
N	MEM_CKE	MEM_BA0	MEM_OPE_N1	MEM_D16	MEM_VD_DQ3	MEM_VSS_Q5	PE3/SPI_RXD/FLASH_CS6	PE2/SPI_SSIN/FLASH_CS5	FLASH_D5	PA5/FLASH_CS1	USBPHY_AVSS1	PH5/SDMMC0_CLK0	PG7/SDMMC1_CLK0	PG3/SDMMC1_D0	PF3/PWM1/SDMMC0_DET	PG1/UART0_TX/SDMMC1_WP	PA1/HOST_D17	XIN24M	N
P	MEM_CKE_LPDDR	MEM_OPE_N2	MEM_D19	MEM_D20	MEM_VD_DQ4	PE1/SPI_CLKI/FLASH_CS4	FLASH_D0	MEM_VD_DQ5	FLASH_D1	FLASH_RDN	USBPHY_DVSS	ID	PE5/I2C0_SCL	I2S_SCLK	PH3/SDMMC0_D2	PG6/SDMMC1_D3	PB0/SPI0_CS1/SDMMC1_PCA	HOST_WRN	P
R	MEM_BA1	MEM_ODT0	MEM_DM2	MEM_D21	MEM_D22	MEM_D26	MEM_D29	PF7/SPI_TXD/FLASH_CS7	VCC_NAND	FLASH_WP	USBPHY_DVDD	VBUS	VDDA_SARADC	I2S_SDI	PE4/I2C0_SDA	PG4/SDMMC1_D1	PB4/SPI0_CSN0/SDMMC0_D4	PE7/UART1_SIR_OUT/I2C1_SCL	R
T	MEM_CASN	MEM_CSN1	MEM_OPE_N3	MEM_D18	MEM_D27	MEM_D30	MEM_D17	FLASH_D6	PA6/FLASH_CS2	FLASH_ALE	USBPHY_AVDD25	USBPHY_AVDD33	ADC_AIN0	FSOURCE_EFUSE	OTG_DRV_VBUS	PH0/SDMMC0_CMD	PG2/SDMMC1_CMD	PB2/UART0_CTSN	T
U	MEM_WEN	MEM_DQ_S2P	MEM_ODT1	MEM_D25	MEM_DQ_S3P	MEM_D31	MEM_D23	FLASH_D3	FLASH_CS0	FLASH_CLE	DP	RKELVIN	ADC_AIN1	VGATE_EFUSE	I2S_SDO	I2S_CLK	PH1/SDMMC0_D0	PB6/SPI0_TXD/SDMMC0_D6	U
V	MEM_DQ_S2M	MEM_OPE_N0	MEM_D24	MEM_DQ_S3M	MEM_D28	MEM_DM3	FLASH_D2	FLASH_D4	PA7/FLASH_CS3	FLASH_WRN	DM	USBPHY_AVSS0	ADC_AIN2	VSSA_SARADC	I2S_LRCK	PA4/I2S_LRCK_RX	PH2/SDMMC0_D1	PH4/SDMMC0_D3	V
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

The following table shows all of the pins for RK281x. According to different application, part of pins will be bonded out, or double/triple bonded.

The first column in the pin function description is default function after power on reset, and function in the last two columns will be implemented by software set.

The detailed register descriptions are IOMUX\_A\_CON and IOMUX\_B\_CON in chapter 34.

As for GPIOm\_n[i] (m = 0,1 ; n = A~D ; i = 0~7), we can control Pull up or Pull Down or no resistor for them by software set. The value for Pull up/down type in the following table is default after power on reset. The detailed register descriptions are in chapter 34.

Notes    *I*    --- input pins  
          *O*    --- output pins  
          *B*    --- bidirectional pins  
          *P*    --- power supply pins (digital and analog)  
          *G*    --- ground supply pins (digital and analog)  
          *A*    --- Analog IO pins  
          *OSC* --- oscillator IO pins

Table 2-1 RK281x Pin Description



## Chapter 3 System Configuration

### 3.1 BUS Architecture

#### 3.1.1 CPU system AHB Bus architecture

In CPU system , the bus matrix is composed of three AHB-Lite bus and three multi-masters bus . RK281x will provide three arbiters on multi-masters bus named as EXP BUS ,ARMD BUS and VIDEO BUS. The default priority for every masters is as the following table. In them , the priority of every masters in ARMD BUS is programmable, however, the priority in EXP BUS and VIDEO BUS is fixed.

##### Priority of Masters in ARMD BUS

Master No.	Priority	Master
5	Highest	MC_DMA master1
4		AHB2AHB from DSP BUS
3		DW_DMA Master1
2		Deblocking(RV) Master1
1	Lowest	ARM DATA

##### Priority of Masters in EXP BUS

Master No.	Priority	Master
5	Highest	Deblocking(RV) Master0
4		USB HOST1.1 Master
3		DW_DMA Master0
2		USB OTG Master
1	Lowest	VIP Master

##### Priority of Masters in VIDEO BUS

Master No.	Priority	Master
2	Highest	MCDMA Master0
1	Lowest	GPU Master

#### 3.1.2 DSP system AHB Bus architecture

DSP system is based on a multi-layer AHB-Lite bus architecture . There are totally 6 masters . And fixed priority arbitration will be used in slave layer when different masters will access a same slave at the same time. The priority for 6 masters is as follows.

##### Priority of Masters in DSP system

Priority	Master
Highest	EXT AHB Master (CPU System)
	XDMA Data Port0
	XDMA Data Port1
	XDMA Manager
	DSP Data Port
Lowest	DSP Instruction Port

#### 3.1.3 Data Path description for CPU and DSP

The following list shows the valid access path for CPU and DSP.

Table 3-1 Valid access path list for CPU and DSP



	Device	CPU	DSP
CPU System	ITCM	V	X
	DTCM	V	X
	Boot ROM	V	V
	SRAM	V	V
	NAND Flash Interface	V	V
	Static/SDRAM Controller Register Port	V	V
	Nor Flash0/Nor Flash1	V	V
	DDRII/LPDDR	V	V
	SDRAM	V	V
	SD/MMC0	V	V
	SD/MMC1	V	V
	Host Interface	V	V
	USB OTG	V	V
	DW_DMA	V	V
	INTC	V	V
	LCDC	V	V
	VIP	V	V
	ARMD BUS Arbiter	V	V
	Video Coprocessor for Deblocking	V	V
	APB UART0/1/2/3	V	V
	APB Timer0/1/2	V	V
	APB eFuse	V	V
	APB GPIO0	V	V
	APB GPIO1	V	V
	APB I2S	V	V
	APB I2C0	V	V
	APB I2C1	V	V
	APB SPI Master	V	V
	APB SPI Slave	V	V
	APB WDT	V	V
	APB PWM	V	V
	APB RTC	V	V
	APB SAR-ADC	V	V
	APB SCU	V	V
	APB Reg File	V	V
DSP System	L1 DMEM	V	V
	L2 MEM_1	V	V
	L2 MEM_2	V	V
	XDMA	X	V
	Hardware Accelerator for Video Decoder	X	V
	High-Speed ADC	X	V
	PMU	V	V
	ICU	V	V
	TIMER0	V	V
	TIMER1	V	V

GPIO	V	V
ASHB MST	V	V
ASHB SLV	V	V
PIU	V	V
APB Reg file	V	V

### 3.1.4 CPU ITCM/DTCM Application Notes

As for ITCM/DTCM (Instruction tightly coupled memory and Data tightly coupled memory) , they are not devices in AHB bus , only accessed by CPU .They are always disabled at reset and can be accessed after enable them by software. As for the detailed information , you can refer to the Appendix A .

Pay more attention that ITCM/DTCM works in the same frequency as CPU , so it can get higher performance. In general , the cycle latency for read/write ITCM/DTCM of CPU is one cycle as illustration in Fig.2-4 and Fig.2-5. However, when CPU frequency is beyond 300MHz , reading operation will be inserted one wait cycle to meet timing requirement, which results in two cycles latency for reading operation as illustration in Fig.2-6.

Insertion one wait cycle will be completed by software set in bit 12 of CPU\_APB\_REG5. Refer to Chapter 34 (General Register File in CPU System) for detailed descriptions.

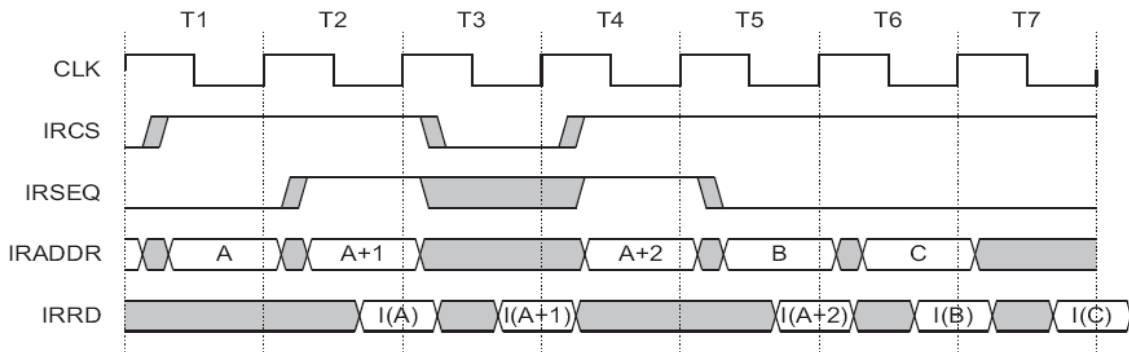


Fig. 3-1 ITCM access timing with zero wait state

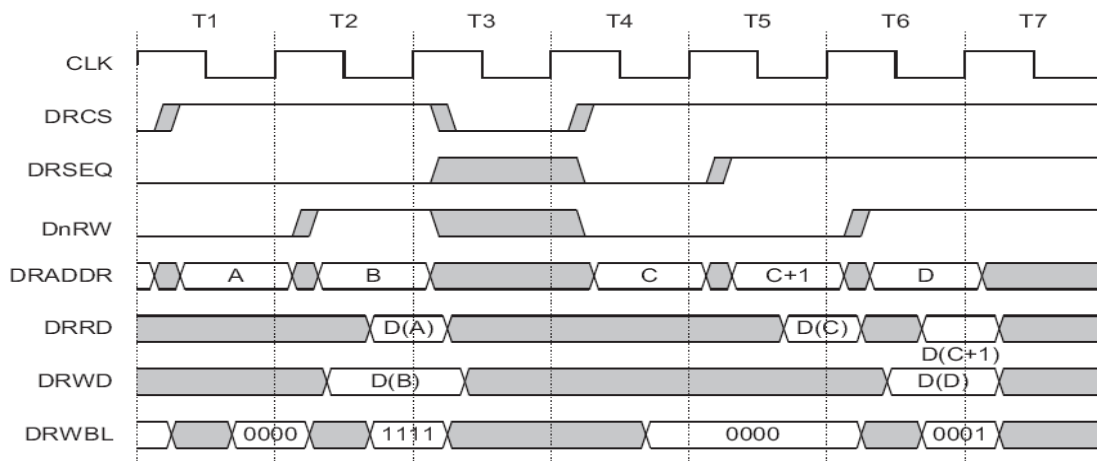


Fig. 3-2 DTCM access timing with zero wait state

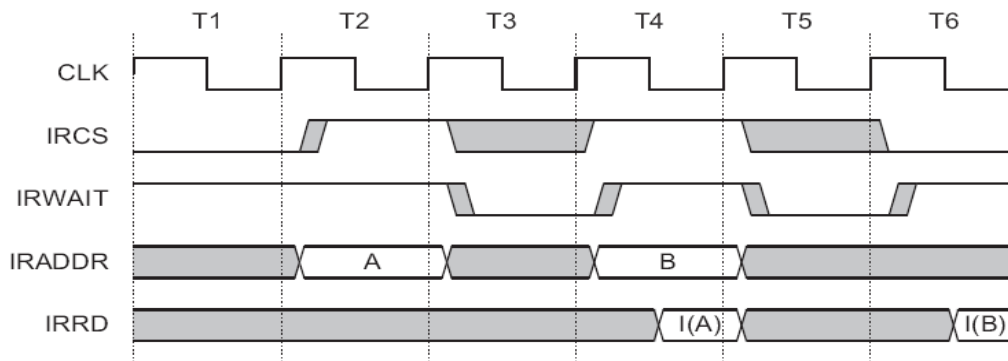


Fig. 3-3 ITCM/DTCM access timing with one wait state

### 3.1.5 External Static/Dynamic Memory Interface Application Notes

External memory controller in RK281x supports Static Memory, SDRAM and mobile SDRAM, DDRII and Mobile DDR interface. There are one register slave port and six data slave ports. In other words, it can support simultaneous read/write for six masters in different bus. Refer to the following diagram. However, we must pay more attention that we only do static select by software config or external IO pin, not dynamic switch for these different types of memory controller. Another, Static memory interface is only supported when use SDRAM memory controller or Mobile SDRAM memory controller.

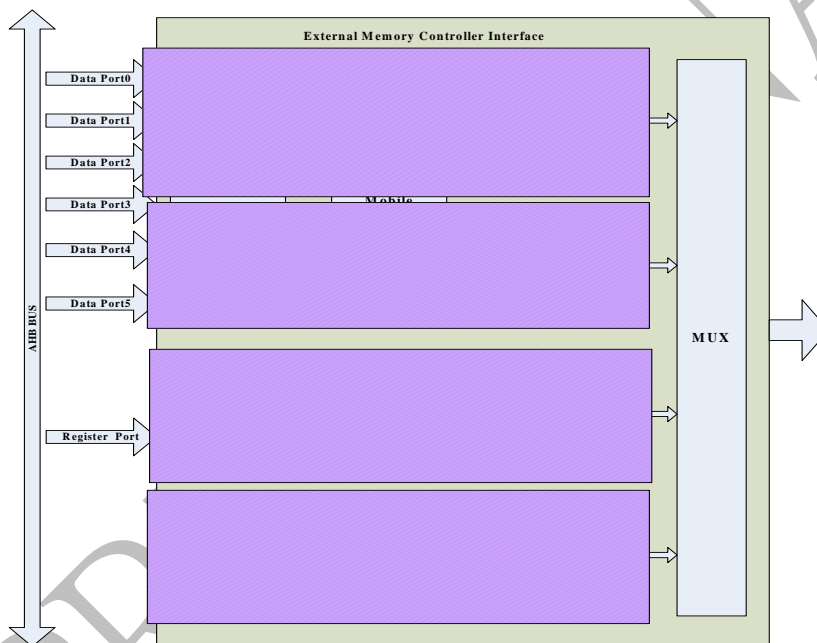


Fig. 3-4 External memory controller architecture

- **Switch for different types of memory**

The function switch for different external memory will be programmable or selected by IO pin, the detailed relationship is shown in the following list.

Memory type	IO Pin		Internal register in apb register groups			Internal register in DDRII/Mobile DDR controller
	IO_EXTDDR_SEL	IO_EXTMSDR_SEL	CPU_APB_REG5 [17]	CPU_APB_REG5 [19]	CPU_APB_REG4 [15]	pad_ctrl_reg_0[18]
DDRII Memory	1	0	x	x	x	0
	x	0	1	x	x	0
Mobile DDR Memory	1	1	x	x	x	x
	1	x	x	x	x	1
	x	1	1	x	x	x
	x	x	1	x	x	1

Customized SDR Memory	0	x	0	1	x	x
Mobile SDR and Static Memory	0	x	0	0	1	x
	0	1	0	0	x	x
SDR and Static Memory	0	0	0	0	0	x

Notes 1 --- high level for IO pins or 1'b1 value for related register bits  
 0 --- low level for IO pins or 1'b0 value for related register bits  
 x --- any level, do not care

### ● Priority for six Data Ports

In RK281x the six data ports of external memory interface are separately connected to six-layer bus. The default priority for six data ports is as follows when SDRAM or Mobile SDRAM is selected. And it is software programmable by bit[14:0] of CPU\_APB\_REG4 and bit[26:24] of CPU\_APB\_REG5 according to different requirement for every masters. Please refer to Chapter 34 (General Register File in CPU System) for detailed descriptions. For DDRII or Mobile DDR, please refer to Chapter 37 for priority switch of six data ports.

#### Priority of data ports for sdram/mobile sdram controller

Priority	Master	Port number
Highest	LCDC BUS	Data Port0
	EXP BUS	Data Port1
	ARMD BUS	Data Port2
	ARMI BUS	Data Port3
Lowest	DSP BUS	Data Port4
	VIDEO BUS	Data Port5

## 3.2 System Address Map

RK281x has fixed address maps for on-chip memory or registers and off-chip peripheral, and the 32-bit address bus can address up to 4GB of memory.

For CPU and DSP, unified address space is used except IPs inside DSP system, which have different address map between CPU and DSP.

For CPU System, RK281x will provide address decode re-map function. It can speed-up whole system performance. The detail memory map is as follows.

### 3.2.1 System Memory Map for CPU

The following list shows address space for every devices CPU can access.

Fig. 3-5 System Memory Map for CPU

Notes :

\* shows address space is after remap, really in default state (before remap) the address space for Boot ROM and Nor Flash0 are also 0x0000\_0000. Refer to the next detailed description

\*\* Before CPU will access Share Mem, some special bits in register CPU\_APB\_REG4 must be set. Refer to Chapter 34 (General Register File in CPU System) for detailed descriptions

RK281x provides remap function as the following diagram. Before remap (power-on-reset state), the address space for Boot ROM and Nor Flash0 are also 0x0000\_0000, which is the first instruction PC value for CPU. And the value for one input pin (IO\_BTMODE) will decide that the zero address is for Boot Rom or for Nor Flash0. After remap, the address space for them will changed to the real physical address. At this time the 0x0000\_0000 address will assigned to ITCM/DTCM after ITCM/DTCM is enabled as the following Fig. 2-3, since ITCM/DTCM is in disable state initially.

Remap operation is software programmable by setting a special bit in register CPU\_APB\_REG5. Refer to detailed description in Chapter 34 (General Register File in CPU

System) .

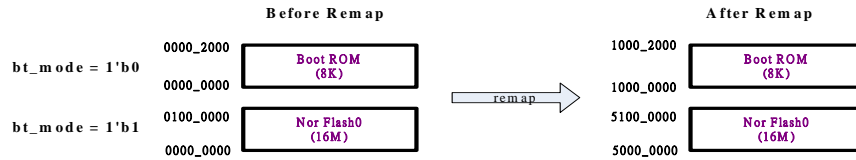


Fig. 3-6 Remap address description



Fig. 3-7 ITCM/DTCM address map

### 3.2.2 System Memory Map for DSP

The following list shows address space for every devices DSP can access .



Fig. 3-8 System Memory Map for DSP

### 3.3 System mode configuration

#### 3.3.1 Debug mode

Table 3-2 RK281x Debug mode descriptions

	op_mode[1:0] *	Description
OP_MODE0	2'b00	CPU Debug
OP_MODE1	2'b01	DSP Debug
OP_MODE2	2'b10	CPU + DSP Debug
OP_MODE3	2'b11	Reserved

Notes \* op\_mode[1:0] is input pins for RK281x

#### 3.3.2 CPU Boot mode

Table 3-3 RK281x boot mode descriptions

	bt_mode *	Description
BOOT_MODE0	1'b0	Boot from Embedded ROM
BOOT_MODE1	1'b1	Boot from NOR Flash bank0

Notes \* bt\_mode is input pin for RK281x

### 3.4 System Interrupt connection

RK281x provides an interrupt controller(INTC) for CPU processor, which has 48 general interrupt sources and 2 fast interrupt sources for internal blocks or external devices , and separately generates one IRQ and one FIQ to CPU. Each interrupts triggered type is high level, not programmable. The detailed interrupt sources connection is in the following table 2-4 . For detailed interrupt controller setting, please refer to Chapter 11 (Interrupt Controller).

Another , for DSP processor there are also an interrupt control unit (ICU), which has 48 maskable interrupt sources and one nmi interrupt , then generates INT0,INT1,INT2,VINT and NMI interrupt to DSP. The interrupt polarity(low/high) and triggered type(edge/level) for each interrupt sources are configurable. The detailed interrupt sources connection is in the following table 2-5 . For detailed ICU setting, please refer to Chapter 12 (Interrupt Control Unit).

Table 3-4 Interrupt sources connection for CPU

Type	Source #	Source Description	Polarity
FIQ	1	Software Interrupt	-
	0	APB GPIO0	High level
IRQ	47	USB HOST	High level
	46	APB UART3	High level
	45	APB UART2	High level
	44	Nand Flash RDY int	High level
	43	MC_DMA int	High level
	42	DDRII/Mobile DDR controller int	High level
	41	GPU MMU int	High level
	40	GPU M55 int	High level
	39	DSP system access error int *	High level
	38	DSP master interface error int **	High level
	37	Software Interrupt	-
	36	APB SCU	High level
	35	DSP interrupt by software set	High level
	34	DSP slave interface error int ***	High level

IRQ	33	SD/MMC 1	High level
	32	XDMA	High level
	31	PIU command/reply	High level
	30	PIU Semaphore 2	High level
	29	PIU Semaphore 1	High level
	28	PIU Semaphore 0	High level
	27	APB RTC	High level
	26	APB SAR-ADC	High level
	25	APB PWM3	High level
	24	APB PWM2	High level
	23	APB PWM1	High level
	22	APB PWM0	High level
	21	APB WDT	High level
	20	APB UART1	High level
	19	APB UART0	High level
	18	APB TIMER2 inside CPU system	High level
	17	APB TIMER1 inside CPU system	High level
	16	APB TIMER0 inside CPU system	High level
	15	APB SPI Slave	High level
	14	APB SPI Master	High level
	13	APB I2S	High level
	12	APB I2C1	High level
	11	APB I2C0	High level
	10	Arbiter in EXP BUS or VIDEO BUS	High level
	9	Arbiter in ARMD BUS	High level
	8	USB OTG	High level
	7	APB GPIO1	High level
	6	APB GPIO0	High level
	5	VIP	High level
	4	SD/MMC 0	High level
	3	LCDC	High level
	2	NANDC	High level
	1	Host Interface	High level
	0	DW_DMA	High level

Notes \* When Masters inside DSP system access any slave devices and hresp is error , DSP system access error int will be generated.

\*\* When Masters inside CPU system access address space inside DSP system and hresp is error , DSP master error int will be generated.

\*\*\* When Masters inside DSP system access address space inside CPU system and hresp is error , DSP slave error int will be generated.

In general, the three above interrupts will be used to debug only . Refer to Appendix B for the detailed descriptions .

Table 3-5 Interrupt sources connection for DSP

Source #	Source Description	Polarity
nmi	CPU interrupt by software set	High level
47	Video Coprocessor for Deblocking(RV)	High level
46	VIP	High level
45	LCDC	High level
44	DW_DMA	High level
43	APB GPIO1	High level
42	MC_DMA int	High level
41	Reserved	-
40	Reserved	-
39	Reserved	-

38	Reserved	-
37	Reserved	-
36	Reserved	-
35	High-Speed ADC Interface	High level
34	PIU command & reply	High level
33	PIU Semaphore 2	High level
32	PIU Semaphore 1	High level
31	PIU Semaphore 0	High level
30	XDMA channel 15	High level
29	XDMA channel 14	High level
28	XDMA channel 13	High level
27	XDMA channel 12	High level
26	XDMA channel 11	High level
25	XDMA channel 10	High level
24	XDMA channel 9	High level
23	XDMA channel 8	High level
22	XDMA channel 7	High level
21	XDMA channel 6	High level
20	XDMA channel 5	High level
19	XDMA channel 4	High level
18	XDMA channel 3	High level
17	XDMA channel 2	High level
16	XDMA channel 1	High level
15	XDMA channel 0	-
14	Software Interrupt	-
13	Software Interrupt	-
12	Software Interrupt	-
11	Software Interrupt	-
10	Software Interrupt	-
9	Software Interrupt	-
8	Software Interrupt	-
7	Software Interrupt	-
6	ASHB snoop inside DSP system	High level
5	Software Interrupt	-
4	ASHB error interrupt	High level
3	XDMA breakpoint interrupt	High level
2	XDMA error interrupt	High level
1	TIMER 1 inside DSP system	High level
0	TIMER 0 inside DSP system	High level

### 3.5 System DMA hardware request connection

RK281x provides 2 DMA controllers : XDMA inside DSP system and DW\_DMA inside CPU system. As for XDMA, there are 48 hardware request ports , the trigger type (edge or level )for each of them is programmable. Each XDMA channel is associated with a trigger source, so each channel has been assigned three hardware request ports really. Another, 16 hardware request ports are used in DW\_DMA, the trigger type for each of them is high level, not programmable.

Please refer to chapter 9 (DW\_DMA) and chapter 10 (XDMA) for detailed usage.

The following tables will describe hardware request connection.

Table 3-6 hardware request connection for DW\_DMA

Source #	Source Description	Polarity
----------	--------------------	----------



0	SD/MMC 0	HIGH level
1	APB UART2 txd	HIGH level
2	APB UART2 rxd	HIGH level
3	APB UART3 txd	HIGH level
4	APB UART3 rxd	HIGH level
5	SD/MMC 1	HIGH level
6	APB I2S txd	LOW level
7	APB I2S rxd	LOW level
8	APB SPI Master txd	HIGH level
9	APB SPI Master rxd	HIGH level
10	APB SPI Slave txd	HIGH level
11	APB SPI Slave rxd	HIGH level
12	APB UART0 txd	LOW level
13	APB UART0 rxd	LOW level
14	APB UART1 txd	LOW level
15	APB UART1 rxd	LOW level

Table 3-7 hardware request connection for XDMA

Channel #	Source Description *	Polarity
0	TIMER0 inside DSP system / XDMA manager port / Reserved	High level
1	TIMER1 inside DSP system / XDMA manager port / Reserved	High level
2	High-speed ADC interface	High level
3	Reserved	-
4	Reserved	-
5	Reserved	-
6	Reserved	-
7	Reserved	-
8	Reserved	-
9	Reserved	-
10	TIMER0 inside DSP system	High level
11	TIMER1 inside DSP system	High level
12	Reserved	-
13	Reserved	-
14	Reserved	-
15	Reserved	-

Notes \* Each channel is associated with three hardware trigger sources, which is software configurable, user can select one trigger source to its assigned channel.

## Chapter 4 Static/SDRAM Memory Controller

### 4.1 Design Overview

#### 4.1.1 Overview

The Static/SDRAM Controller is a memory controller that you can control Synchronous DRAMs – SDR-SDRAM, – as well as Static memories – SRAMs and FLASHes

#### 4.1.2 Features

##### AMBA AHB Interface Features

- AMBA AHB bus-compatible
- Supports all types of AMBA bursts
- Supports AHB data widths of 32 bits
- Supports AHB address width of 32 bits
- Supports busy and early terminations on AHB transactions
- Does not generate split, retry, or error responses on the AMBA bus
- Two-clock-cycle latency from AMBA bus hsel\_mem assertion to issue of memory command, depending on optional registering of memory control and data signals
- Supports shared memory address and data buses between SDRAM and Static memories

##### SDRAM Interface Features

- Glueless connection to all JEDEC-compliant SDRAM
- Supports up to 16 SDRAM address bits
- SDR-SDRAM data width is 32 bits
- Programmable row and column address bit widths up to:
  - ◆ 15-bit column address
  - ◆ 16-bit row address
  - ◆ 2-bit bank address
- Supports 2K to 64K rows, 256 to 32K columns, and 4 banks
- Supports up to 3 chip selects, with a maximum of 4 GB of address space per chip select
- SDRAM timing parameters – tRAS, tRCD, tRP, tWR, tWTR, tRCAR, tXSR, and tRC, – can be programmed to values supported by different SDRAM vendors
- Supports auto refresh with programmable refresh intervals
- Supports self-refresh
- Supports SDRAM power-down mode
- Programmable immediate precharge or delayed precharge modes
- Supports 1 to 4 (programmable) open banks for performance; pages can be non-contiguous –Least Recently Used (LRU) algorithm used during page miss replacements

##### Static Memory Interface Features

- Supports asynchronous SRAMs and page-mode FLASHes
- Supports up to three sets of timing registers
- Configurable address width of up to 32 bits
- Limited synchronous SRAM and FLASH interface support
- Synchronous SRAM and FLASH frequency could be 1, 1/2, 1/3, 1/4, and so on of the AHB frequency

### 4.2 Architecture

This section provides a description about the functions and behavior under various conditions.

### 4.2.1 Overview

The Static Memory/SDRAM controller can provide an interface between each of the following memory devices and an AMBA AHB 2.0 bus

- JEDEC-standard SDR-SDRAM
- Asynchronous SRAM, with or without page-mode
- Asynchronous FLASH, with or without page-mode
- Limited synchronous SRAM and FLASH interface support

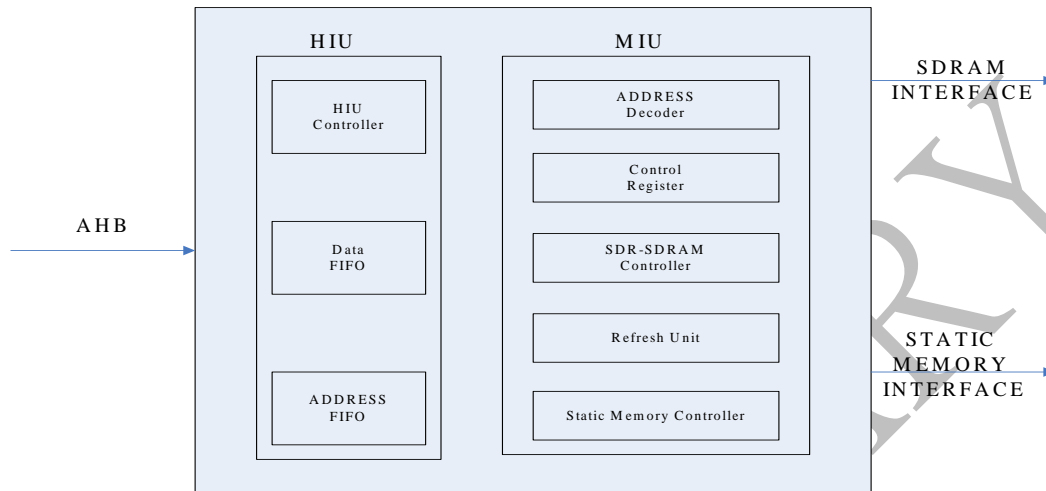


Fig. 4-1 Static Memory/SDRAM block diagrams

### 4.2.2 Block Descriptions

#### AMBA Host Interface Unit (HIU)

The AMBA Host Interface Unit (HIU) is the interface between the memctl and the AMBA Advanced High-performance Bus (AHB). The HIU generates memory read/write requests or control register read/write requests to the MIU block, which correspond to transfers on the AMBA bus; the HIU does not distinguish between an SDRAM request and an SRAM/FLASH request.

The HIU consists of the following sub-blocks:

- Address FIFO – Buffers the request of the AMBA AHB and sends memory/register access requests to the MIU; also contains some control information for a read/write transfer
- Write Data FIFO – Buffers write data to the memory and control registers
- Read Data FIFO – Buffers the read data from the memory
- HIU Control – Controls all the HIU sub-blocks by generating the control logic for read and write transfers

#### Memory Interface Unit (MIU)

The memory interface unit (MIU) is the interface for both SDRAM and Static memories; it generates appropriate address, data, and control signals corresponding to memory read/write transfers. The MIU contains two sets of modules, which are enabled depending on whether you choose the SDRAM or Static memory.

If you choose the SDRAM controller, the MIU includes the following modules:

- SDRAM controller – Generates the SDRAM control signals
- Refresh unit – Generates the SDRAM refresh request at appropriate intervals
- Address decoder – Generates the row, column, and bank addresses that correspond to the logical address provided by the host interface and Decodes and generates the address to SRAM/FLASH from the AHB address
- Control register – Holds the SDRAM control and configuration registers, and holds the control registers and timing registers for Static memories.
- Static control unit – Generates the SRAM/FLASH control signals

## 4.3 Registers

This section describes the control/status registers of the design.

### 4.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SDR_SCONR	0x00	W	0x1C4F68	SDRAM configuration register
SDR_STMG0R	0x04	W	0x1E5D696	SDRAM timing register0
SDR_STMG1R	0x08	W	0x7008	SDRAM timing register1
SDR_SCTLR	0x0C	W	0x3009	SDRAM control register
SDR_SREFR	0x10	W	0x410	SDRAM refresh register
SDR_SCSLR0_LOW	0x14	W	0x0	Chip select register0 (lower 32bits)
SDR_SCSLR1_LOW	0x18	W	0x5100	Chip select register1 (lower 32bits)
SDR_SCSLR2_LOW	0x1C	W	0x6000	Chip select register2 (lower 32bits)
SDR_SMSKR0	0x54	W	0x149	Mask register 0
SDR_SMSKR1	0x58	W	0x249	Mask register 1
SDR_SMSKR2	0x5C	W	0xC	Mask register 2
SDR_CSREMAP0_LOW	0x84	W	0x50000000	Remap register for chip select0 (lower 32 bits)
SDR_SMTMGR_SET0	0x94	W	0x1154C	Static memory timing register Set0
SDR_SMTMGR_SET1	0x98	W	0x791950	Static memory timing register Set1
SDR_SMTMGR_SET2	0x9C	W	0x1C1950	Static memory timing register Set2
SDR_FLASH_TRPDR	0xA0	W	0xC8	FLASH memory tRPD timing register
SDR_SMCTLR	0xA4	W	0x1201	Static memory control register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 4.3.2 Detail Register Description

#### SDR\_SCONR

Address: Operational Base + offset( 0x00)

SDRAM Config Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	RW	0x1	Reserved
19	RW	0x1	Reserved
18	RW	0x1	Reserved
17:15	RW	0x0	Reserved
14:13	RW	0x1	Specifies SDRAM data width in bits; fixed in 32bits; No use
12:9	RW	0x7	Number of address bits for column address; 15 – reserved 7-14 – correspond to 8-15 bits 0-6 – reserved

8:5	RW	0xb	Number of address bits for row address; 10-15 – correspond to 11-16 bits 0-10 – reserved
4:3	RW	0x1	Number of bank address bits; bit Values of 0-3 correspond to 1-4 bits, and therefore select 2-16 banks
2:0	-	-	Reserved.

**SDR\_STMG0R**

Address: Operational Base + offset( 0x04)

SDRAM Timing Register0

bit	Attr	Reset Value	Description
25:22	RW	0x7	Active-to-active command period; values of 0-15 correspond to t <sub>rc</sub> of 1-16 clocks
31:27 21:18	RW	0x0 0x9	Exit self-refresh to active or auto-refresh command time; minimum time controller should wait after taking SDRAM out of self-refresh mode before issuing any active or auto-refresh commands; values 0-511 correspond to t <sub>xsr</sub> of 1-512 clocks
17:14	RW	0x7	Auto-refresh period; minimum time between two auto-refresh commands; values 0-15 correspond to t <sub>rcar</sub> of 1-16 clocks.
13:12	RW	0x1	For writes, delay from last data in to next precharge command; values 0-3 correspond to t <sub>wr</sub> of 1-4 clocks
11:9	RW	0x3	Precharge period; values of 0-7 correspond to t <sub>rp</sub> of 1-8 clocks
8:6	RW	0x2	Minimum delay between active and read/write commands; values 0-7 correspond to t <sub>rcd</sub> values of 1-8 clocks
5:2	RW	0x5	Minimum delay between active and precharge commands; values of 0-15 correspond to T <sub>RAS_MIN</sub> of 1-16 clocks
26 1:0	RW	0x0 0x2	Delay in clock cycles between read command and availability of first data 0 – 1 clock 1 – 2 clocks 2 – 3 clocks 3 – 4 clocks 4 – Reserved 5 – Reserved 6, 7 – reserved

**SDR\_STMG1R**

Address: Operational Base + offset( 0x08)

SDRAM Timing Register1

bit	Attr	Reset Value	Description
31:22	-	-	Reserved.
21:20	RW	0x0	Reserved.
19:16	RW	0x7	Number of auto-refreshes during initialization; values 0-15 correspond to 1-16 auto-refreshes
15:0	RW	0x8	Number of clock cycles to hold SDRAM inputs stable after power up, before issuing any commands.

**SDR\_SCTLR**

Address: Operational Base + offset( 0x0C)

## SDRAM Control Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	-	-	Reserved.
19	-	-	Reserved.
18	-	-	Reserved.
17	RW	0x0	Reserved.
16:12	RW	0x3	Number of SDRAM internal banks to be open at any time; values of 1-16 correspond to 0-15 banks open.
11	R	0x0	Read only. When "1," indicates SDRAM is in self-refresh mode. When "self_refresh/deep_power_mode" bit (bit 1 of SCTL0) is set, it may take some time before SDRAM is put into self-refresh mode, depending on whether all rows or one row are refreshed before entering self-refresh mode defined by full_refresh_before_sr bit. Before gating clock in self-refresh mode, ensure this bit is set
9	RW	0x0	Set to 1, forces controller to do update of SDRAM mode register; bit is cleared by controller once it has finished mode register update
8:6	R/W	0x0	Indicates number of registers inserted in read data path for SDRAM in order to correctly latch data; values 0-7 indicate 0-7 registers
5	R/W	0x0	Controls number of refreshes done by SDR_memctl after SDRAM is taken out of self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just 1 row before entering self-refresh mode
4	R/W	0x0	Controls number of refreshes done by SDR_memctl before putting SDRAM into self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just one row before entering self-refresh mode
3	R/W	0x1	Determines when row is precharged: 0 – Immediate precharge; row precharged at end of read/write operation 1 – Delayed precharge; row kept open after read/write operations
2	R/W	0x0	Forces to put SDRAM in power-down mode;
1	R/W	0x0	Forces to put SDRAM in self-refresh mode. Bit can be cleared by writing to this bit or by clear_sr_dp pin, generated by external power management unit
0	R/W	0x1	Forces to initialize SDRAM; bit reset to 0 by SDR_memctl once initialization sequence is complete

## SDR\_SREFR

Address: Operational Base + offset( 0x10)

## SDRAM Refresh Interval Register

bit	Attr	Reset Value	Description
31:24	R	0x0	Reserved.
23:16	RW	0x0	Reserved.
15:0	RW	0x410	Number of clock cycles between consecutive refresh

			cycles;
--	--	--	---------

**SDR\_SCSLR0**

Address: Operational Base + offset( 0x14)

chip\_select\_register0

bit	Attr	Reset Value	Description
31:16	R	0x0000	Upper 16bits of base address for static memory bank0
15:0	-	-	Reserved.

**SDR\_SCSLR1**

Address: Operational Base + offset( 0x18)

chip\_select\_register1

bit	Attr	Reset Value	Description
31:16	R	0x5100	Upper 16bits of base address for static memory bank1
15:0	-	-	Reserved.

**SDR\_SCSLR2**

Address: Operational Base + offset( 0x1C)

chip\_select\_register0

bit	Attr	Reset Value	Description
31:16	R	0x6000	Upper 16bits of base address for SDRAM
15:0	-	-	Reserved.

**SDR\_SMSKR0**

Address: Operational Base + offset( 0x54)

Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	R/W	0x1	Register determines which timing parameters of memory connect to static memory bank0; primarily used for specifying static memories 0 – register set 0 , set in SDR_SMTMGR_SET0 1 – register set 1 , set in SDR_SMTMGR_SET1 2 – register set 2 , set in SDR_SMTMGR_SET2
7:5	R/W	0x2	Type of memory connected to static memory bank0: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R/W	0x9	size of memory connected to static memory bank0; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

**SDR\_SMSKR1**

Address: Operational Base + offset( 0x58)

Address Mask Registers

bit	Attr	Reset Value	Description
-----	------	-------------	-------------



31:11	-	-	Reserved.
10:8	R/W	0x2	Register determines which timing parameters of memory connect to static memory bank1; primarily used for specifying static memories 0 – register set 0 , set in SDR_SMTMGR_SET0 1 – register set 1 , set in SDR_SMTMGR_SET1 2 – register set 2 , set in SDR_SMTMGR_SET2
7:5	R/W	0x2	Type of memory connected to static memory bank1: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R/W	0x9	size of memory connected to static memory bank1; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

**SDR\_SMSKR2**

Address: Operational Base + offset( 0x5C)

Address Mask Registers

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:5	R/W	0x0	Type of memory connected to SDRAM 0 – SDRAM Others – Reserved
4:0	R/W	0xc	size of memory connected to SDRAM; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10 – 32MB 11 – 64MB 12 – 128MB 13~17 – Reserved

**SDR\_CSREMAPO\_LOW**

Address: Operational Base + offset(0x84)

REMAP REGISTER0

bit	Attr	Reset Value	Description
31:16	R	0x5000	Represent lower remap register bits for static memory bank0; compared with corresponding AHB address to generate chip select0; number of compared bits depends on size of memory selected by chip select0



			(specified SDR_SMSKR0): 64KB – bits 31:16 compared 128 KB – bits 31:17 compared
15:0	-	-	Reserved.

**SDR\_SMTMGR\_SET0**

Address: Operational Base + offset(0x94)

Static Memory Timing Register - Set0

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set0
27	RW	0x0	Valid if register set0 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x0	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x5	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0xc	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

**SDR\_SMTMGR\_SET1**

Address: Operational Base + offset(0x98)

Static Memory Timing Register – Set1

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set1

27	RW	0x0	Valid if register set1 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0xf	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

**SDR\_SMTMGR\_SET2**

Address: Operational Base + offset(0x9C)

Static Memory Timing Register – Set2

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set2
27	RW	0x0	Valid if register set2 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device:

			0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x3	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x4	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

**SDR\_FLASH\_TRPDR**

Address: Operational Base + offset(0xA0)

FLASH Timing Register

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11:0	RW	0xC8	FLASH reset/power-down high to read/write delay; values correspond to sm_rp_n high to read/write delay minus one

**SDR\_SMCTLR**

Address: Operational Base + offset(0xA4)

Static Memory Control Register

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:13	RW	0x0	Width of Static memory data bus controlled by Static memory register SET2: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
12:10	RW	0x4	Width of Static memory data bus controlled by Static memory register SET1: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
9:7	RW	0x4	Width of Static memory data bus controlled by Static memory register set 0: 000 – 16 bits 001 – 32 bits 010 – 64 bits

			011 – 128 bits 100 – 8 bits
6:4	-	-	Reserved.
3:1	RW	0x0	FLASH write-protection mode; writing 0 forces FLASH memory bootblock to write protect; the three bits correspond to three register sets
0	RW	0x1	FLASH reset/power-down mode; after reset, controller internally performs a power-down for FLASH and then sets this bit to 1 to force FLASH to power-down mode during normal operation: 0 – Forces FLASH to power-down mode 1 – Takes FLASH out of power-down mode

## 4.4 Functional Description

### 4.4.1 Operation

#### Basic Access Operation

1. A page-hit single-cycle write:

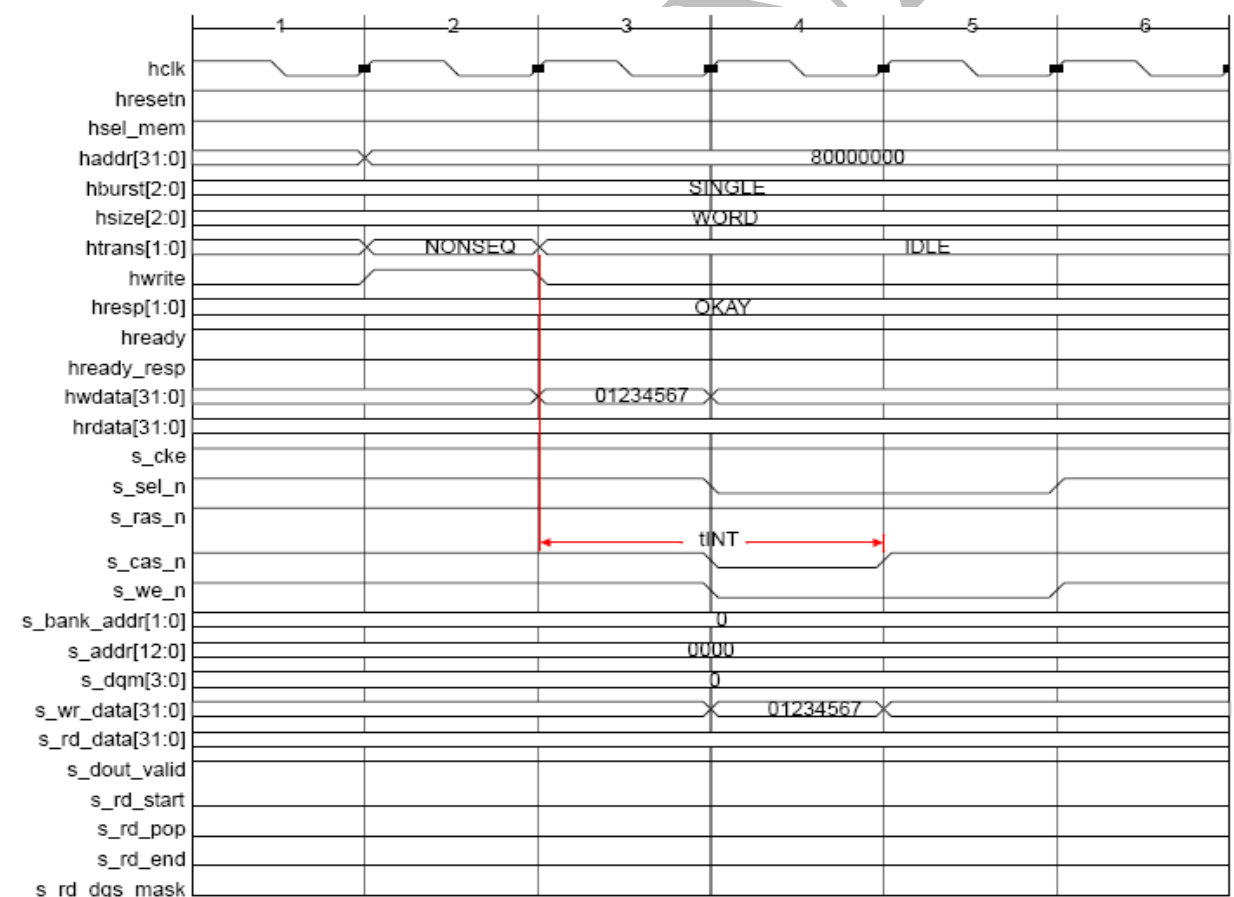


Fig. 4-2 SDRAM Page-Hit Single Write

2. A page-miss single-cycle write:

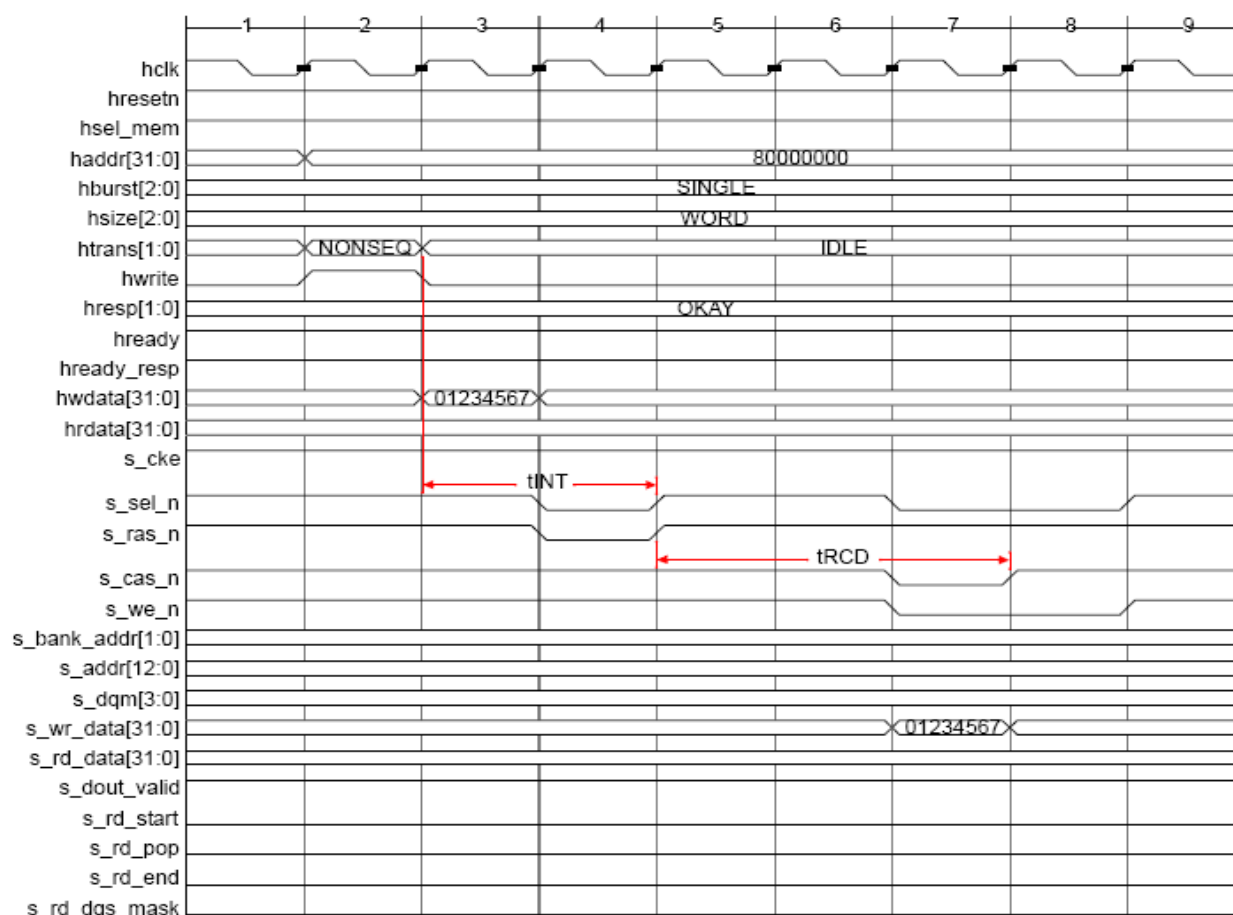


Fig. 4-3 SDRAM Page-Miss Single Write

### 3 、A page-hit burst write (hburst == INCR8)

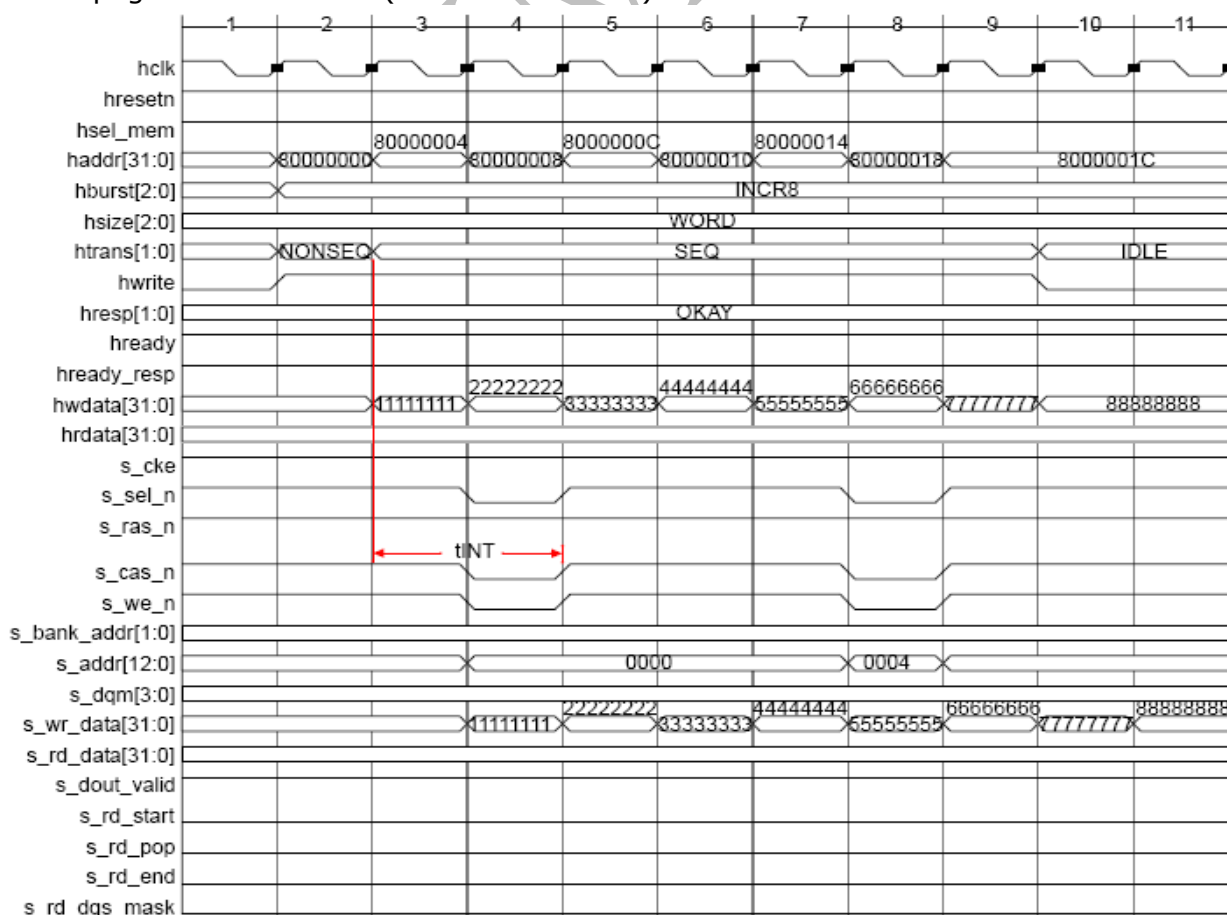


Fig. 4-4 SDRAM Page-Hit Busrt Write

4、A page-hit single-cycle read:

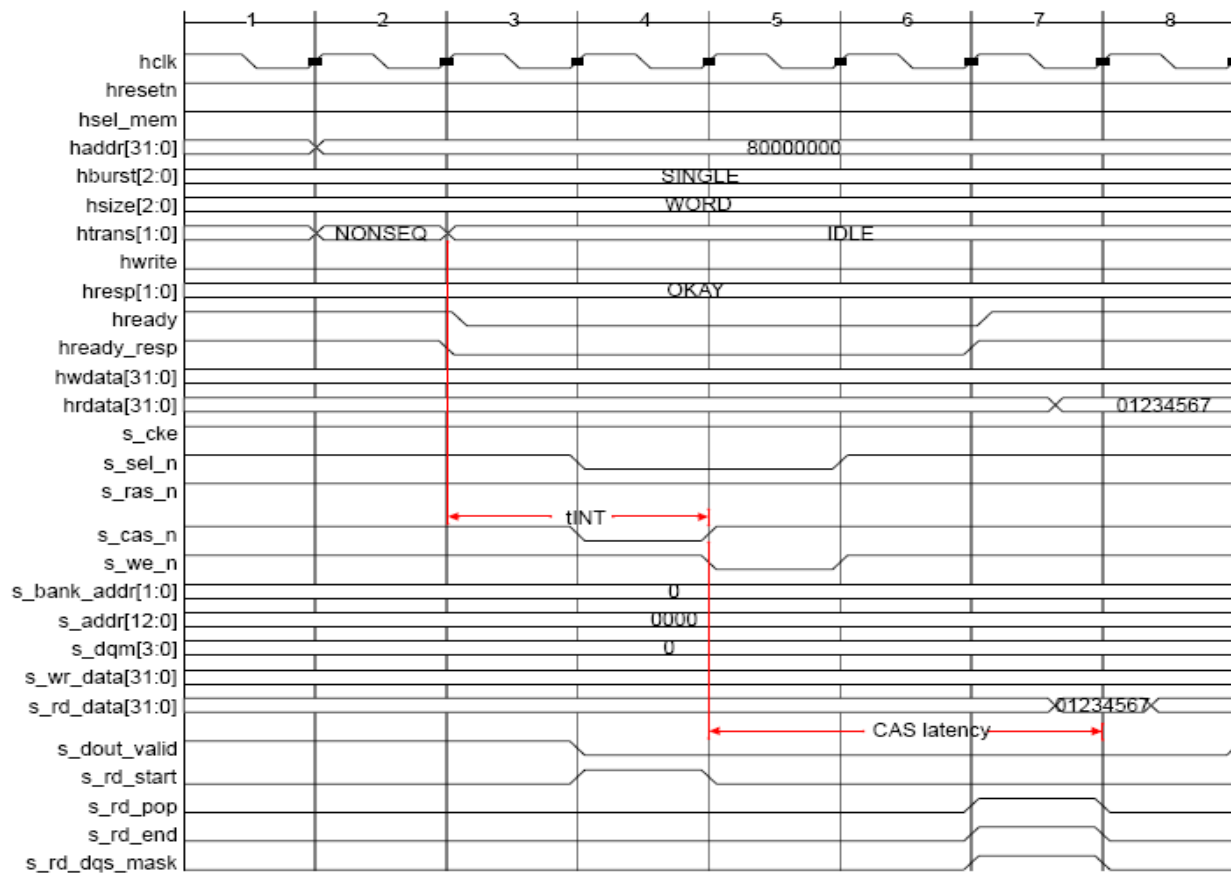


Fig. 4-5 SDRAM Page-Hit Single Read

5、a page-miss single-cycle read:

PRELIMINARY

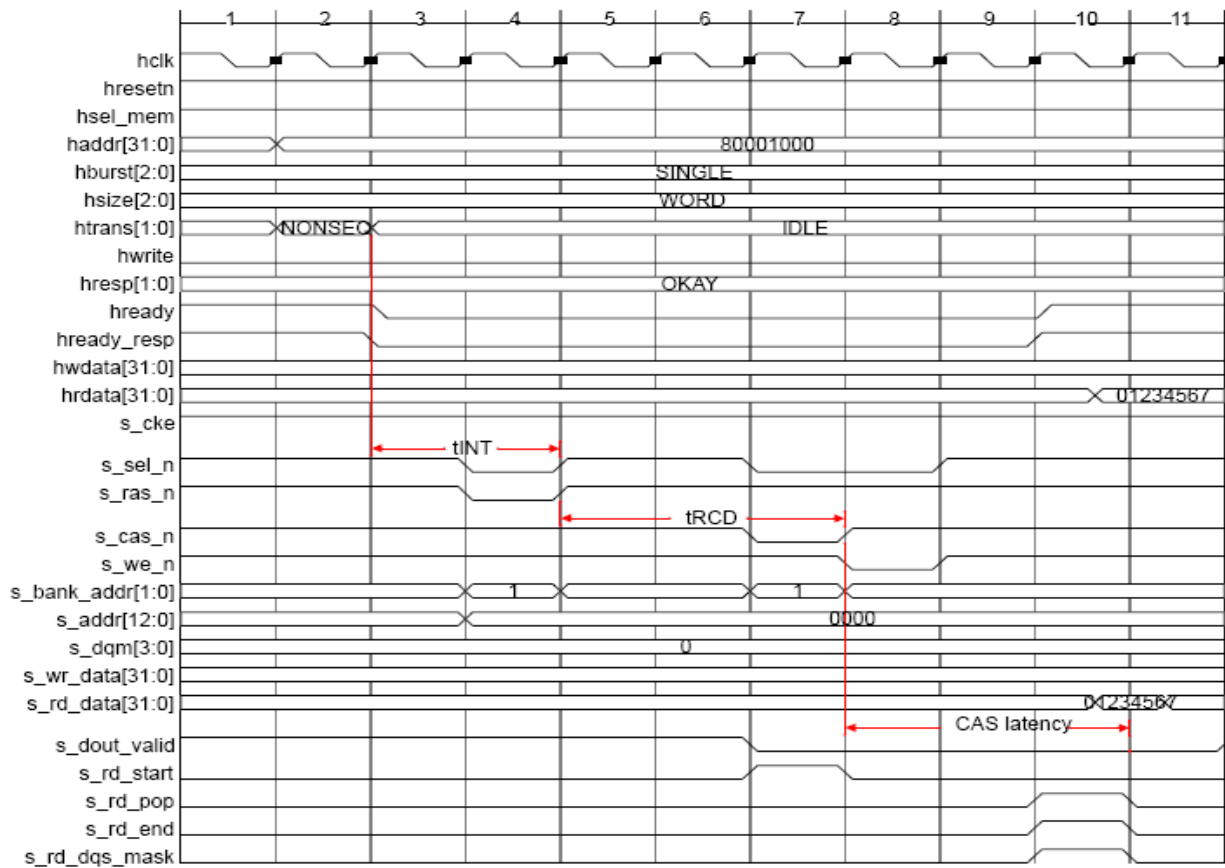


Fig. 4-6 SDRAM Page-Miss Single Read

## 6. A page-hit burst read:

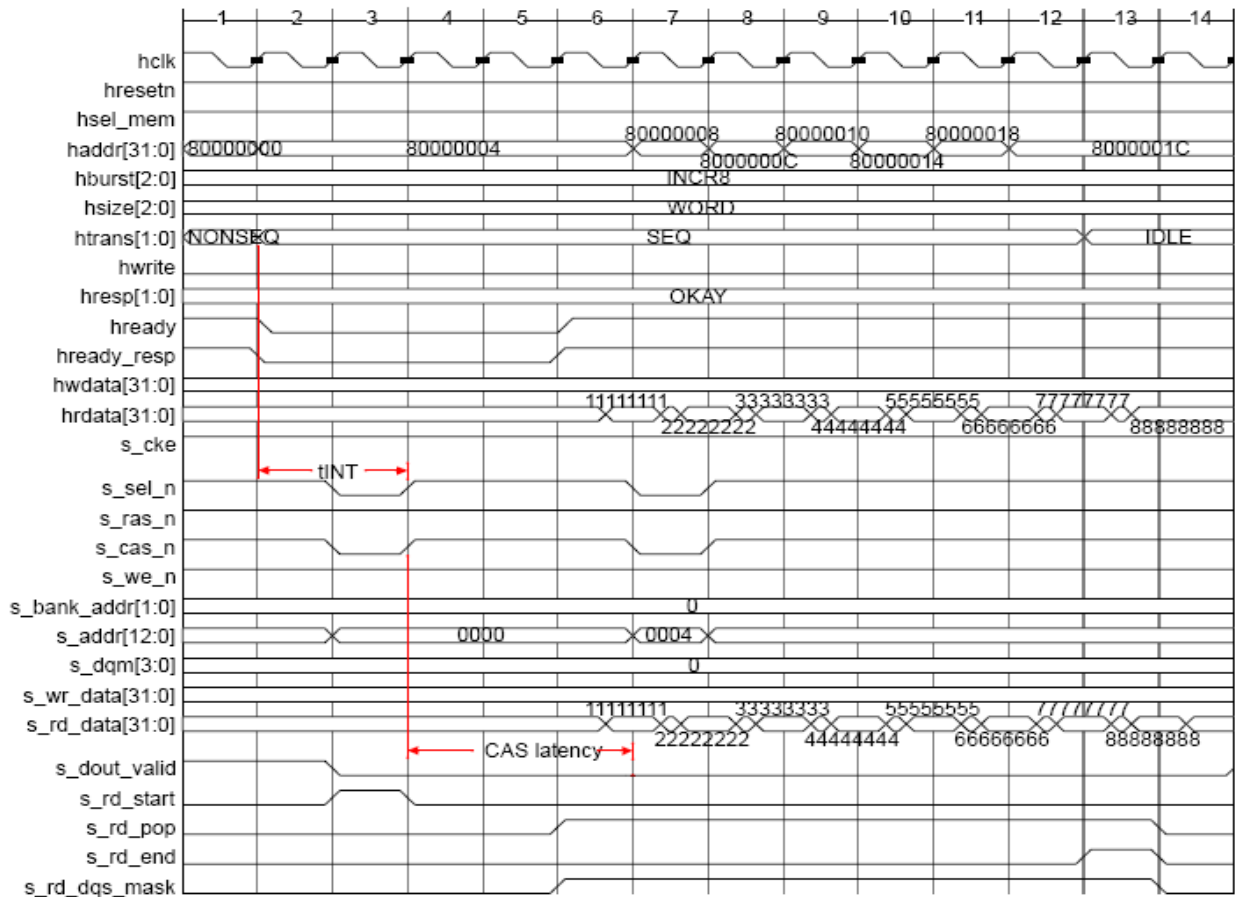


Fig. 4-7 SDRAM Page-Hit Busrt Read



## Power-On Initialization

The SDR-SDRAM controller follows the JEDEC-recommended SDR-SDRAM power-on initialization sequence as follows:

1. Apply power and start clock; maintain a NOP condition at the inputs
  2. Maintain stable power, stable clock, and NOP input conditions for a minimum of  $t_{init}$  clock cycles
  3. Issue precharge commands for all banks of the device
  4. Issue auto-refresh commands, depending on the value `num_init_ref` in the programmable register
  5. Issue a set-mode register command to initialize the mode register
- The commands issued to the SDRAM by the controller during the power-on initialization are shown as followed

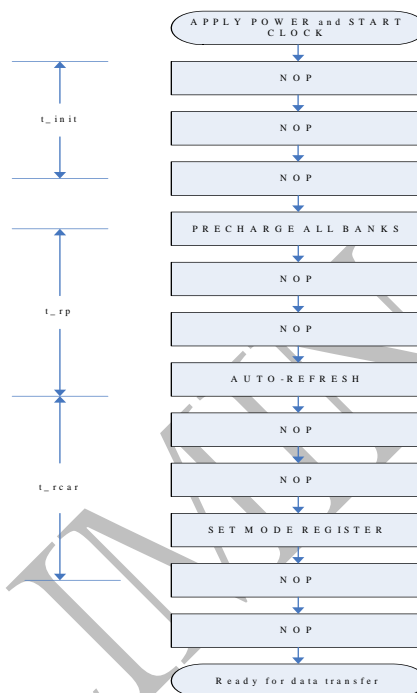


Fig. 4-8 Static Memory/SDRAM Controller power on sequence

## Read/Write Operation

The SDRAM Controller converts all AHB bursts to 4-word bursts on the SDRAM side. The memory bursts are concatenated to achieve continuous data flow for long AHB bursts. You can terminate the memory read/write burst with either a precharge command or terminate command, depending on which precharge mode – immediate precharge or delayed precharge – that you program. You can also terminate the write burst with a subsequent write burst

The SDRAM Controller supports two precharge modes – immediate precharge and delayed precharge. If you program for an immediate precharge mode, then the SDRAM Controller closes the open row after a read or write access. If you program for a delayed precharge mode, then the `SDR_memctl` keeps the row open after an access. The `SDR_memctl` can keep multiple banks open at the same time, depending on the value of `num_open_bank` in the programmable register. When the number of open banks reaches the `num_open_bank` and an access to a new bank comes, the SDRAM Controller will close the oldest bank (the bank opened first) before opening the new bank.

## Set-Mode Register

The SDRAM Controller automatically sets the SDR-SDRAM mode register during the power-up initialization. During normal operation, if you want to set the mode register, you need to set `set_mode_reg` (bit 9 of `SCTLR`) to 1. After the memory controller finishes the mode register setting, it clears the `set_mode_reg` to 0.

The “burst length” field and the “burst type” field of the SDR-SDRAM-mode register are fixed by the SDRAM Controller to “010” (burst length 4) and “0” (sequential burst), respectively. The SDRAM Controller programs the “CAS latency” field and the “operating mode” field of the mode register according to the values provided by the user in the control and timing registers.

### .Auto-Refresh

Auto-refresh commands are issued when the refresh control block issues refresh requests. During normal refresh operations, the SDRAM Controller always refreshes one row at a time. It is important for the user to program the tREF refresh interval register after a reset. If you need to refresh the SDRAM while a burst is active, normally the SDRAM Controller will issue the refresh command after the ongoing burst completes. However, if the ongoing burst is an AHB INCR burst, the SDRAM Controller will stop the burst, issue the refresh command, and then resume the burst.

The SDRAM Controller takes into account the maximum time it takes to complete a worst-case burst. This is the time to complete a read burst corresponding to an INCR16 burst on the AMBA bus, and with an AMBA-to-SDRAM data width ratio of 2:1. It is reasonable to assume 50 cycles for this worst-case burst, with 32 cycles for the data and the remaining 17 cycles for various latencies for the worst case.

The t<sub>ref</sub> value can be calculated using the following equation:

$$t_{ref} = \text{refresh\_period} / \text{clock\_period}$$

where refresh\_period = typically 7.8/15.6 s.

The tREF is the value of a free-running counter that the refresh logic in the SDRAM Controller operates on. When the count expires, the refresh logic gives a refresh request to the SDRAM controller.

Since the 64 ms refresh period is the same for most SDRAMs, the total number of rows in the SDRAM limits the minimum operating frequency for the SDR\_memctl. While calculating the minimum frequency, use the following equation:

$$t_{REF} > 50 * (1/f).$$

Typically, the refresh cycle is 15.6 s or 7.8 s, depending on the refresh rate; The table is summarized as followed:

Number of rows	tREF	Min Frequency
64K	(64ms-(50/f))/65536	51Mhz
32K	(64ms-(50/f))/32768	26Mhz
16K	(64ms-(50/f))/163904	13Mhz
8K	(64ms-(50/f))/8192	6Mhz
4K	(64ms-(50/f))/4096	3Mhz
2K	(64ms-(50/f))/2048	1.5Mhz

The refresh logic in the SDR\_memctl is inactive when the SDR\_memctl forces the SDRAM into self-refresh or power-down mode.

### Self-Refresh

You can put the SDRAM into self-refresh mode, at which point the SDRAM retains data without external clocking and auto-refresh. The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit the self-refresh mode

You can force the SDRAM Controller to enter self-refresh mode by programming bit 1 of the SDRAM control register (SCTLR) (Address 32'hxxxx\_xx0C). The SDRAM Controller forces the SDRAM to come out of self-refresh mode when bit 2 of the SCTLR is set to 0. You can set this bit to 0 by either programming the SDRAM control register or driving the clear\_sr\_dp pin high. You can use the clear\_sr\_dp pin when the code resides in the SDRAM, and the SDRAM itself is in self-refresh mode.

Bits 4 and 5 of the SCTLR specify the type of refresh done by the SDRAM Controller just prior to entering self-refresh mode and just after entering self-refresh mode.

Programming bit 4 of the SCTLR to 0 forces the SDRAM Controller to refresh only one row before putting the SDRAM into self-refresh mode. The default value of 1 forces the SDRAM

Controller to perform auto-refreshes for all rows. Bit 5 does the same, except that it controls the refresh pattern just after coming out of self-refresh mode.

Since it takes time between programming the control register bit to the SDRAM entering self-refresh mode, the SDRAM Controller provides a read-only register bit (bit 11 of the SDRAM control register) to indicate that the SDRAM is already in self-refresh mode. If you want to gate off the clock to the SDRAM Controller when the SDRAM is in self-refresh mode, you should ensure this bit is set to 1 before you stop the clock.

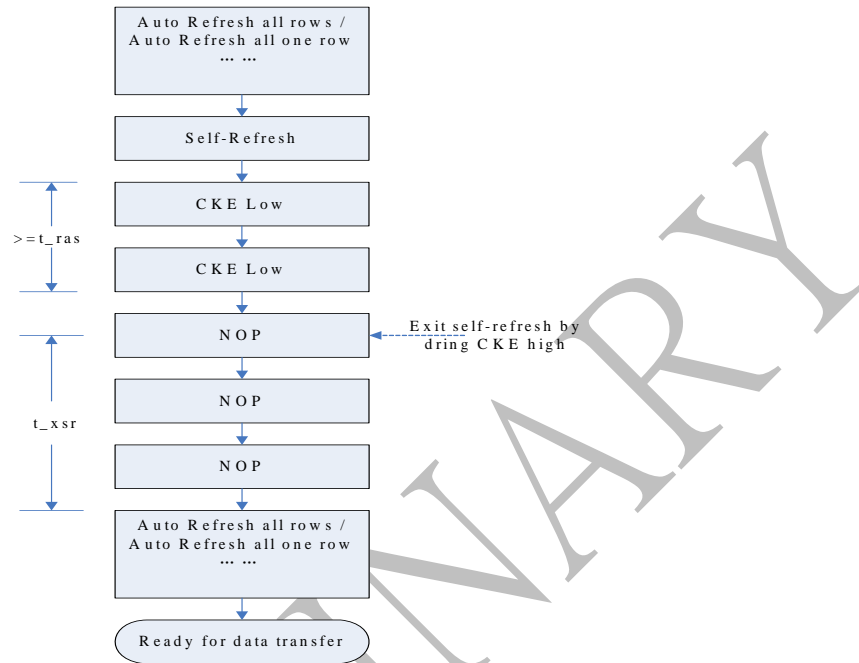


Fig. 4-9 Static Memory/SDRAM Controller self\_refresh mode

The SDRAM must remain in self-refresh mode for a minimum period of  $t_{ras}$  and can remain in self-refresh mode for an indefinite period of time. After the SDRAM exits self-refresh mode, the SDRAM Controller issues NOP commands for  $t_{xsr}$  before it issues any other command. The  $t_{ras}$  and  $t_{xsr}$  are programmable register values and have default values. These registers can be programmed only once after reset.

When an AHB read/write request to the SDRAM occurs while the SDRAM is in self-refresh mode, the SDRAM Controller generates dummy ready signals to the AHB without accessing external memory; no error response is generated on the AHB bus.

### Power-Down

The SDRAM can be put into power-down mode to save power. There are two ways to force the SDR\_memctl to put the SDRAM in power-down mode:

- Program bit 2 of SCTL to 1; should be 0 to bring the SDRAM out of power-down mode.
- Use the power-down input pin; can be driven by an external power management unit; the SDRAM will be in power-down mode as long as this signal stays high

The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit power-down mode

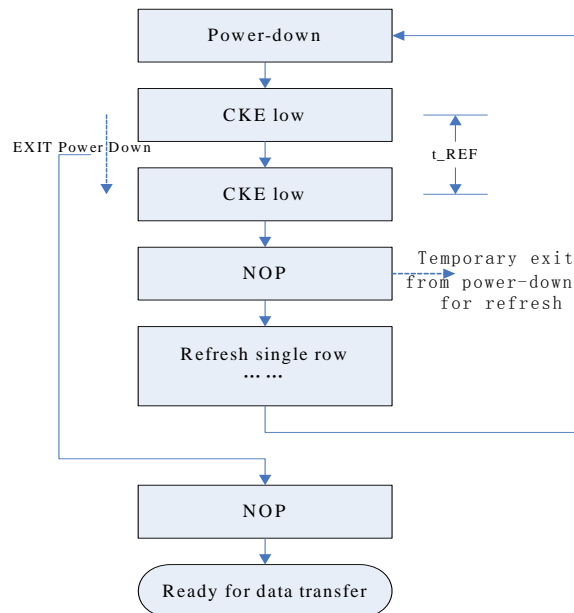


Fig. 4-10 Static Memory/SDRAM Controller power\_off mode

When in SDRAM power-down mode, the SDRAM Controller keeps switching the device back and forth between power-down and refresh mode. It remains in power-down for a  $t_{ref}$  period of time, then comes out of power-down and does a single-row refresh; then it again goes into power-down mode.

The SDRAM Controller keeps the SDRAM in this periodical power-down/refresh/power-down sequence until it is commanded to exit power-down mode (by programming bit 2 of SCTL0 to 0). When an AHB read/write request to the SDRAM occurs while the SDRAM is in power-down mode, the SDRAM Controller brings the SDRAM out of power-down mode and issues the read/write access to the SDRAM. The SDRAM Controller then puts the SDRAM back to power-down mode after the read/write access.

## Chapter 5 Static/Mobile SDRAM Memory Controller

### 5.1 Design Overview

#### 5.1.1 Overview

The Static Memory/Mobile SDRAM Controller is a memory controller that you can control Synchronous DRAMs – MSDR-SDRAM, – as well as Static memories – SRAMs and FLASHes

#### 5.1.2 Features

##### AMBA AHB Interface Features

- AMBA AHB bus-compatible
- Supports all types of AMBA bursts
- Supports AHB data widths of 32.
- Supports AHB address width of 32 bits
- Supports busy and early terminations on AHB transactions
- Does not generate split, retry, or error responses on the AMBA bus
- Two-clock-cycle latency from AMBA bus hsel\_mem assertion to issue of memory command, depending on optional registering of memory control and data signals
- Supports shared memory address and data buses between Mobile SDRAM and Static memories

##### MOBILE SDRAM Interface Features

- Glueless connection to all JEDEC-compliant Mobile SDRAM
- Supports up to 16 SDRAM address bits
- Mobile SDRAM data width is 32 bits
- Programmable row and column address bit widths up to:
  - ◆ 15-bit column address
  - ◆ 16-bit row address
  - ◆ 2-bit bank address
- Supports 2K to 64K rows, 256 to 32K columns, and 4 banks
- Supports up to 3 chip selects, with a maximum of 4 GB of address space per chip select
- Mobile SDRAM timing parameters – tRAS, tRCD, tRP, tWR, tWTR, tRCAR, tXSR, and tRC, – can be programmed to values supported by different SDRAM vendors
- Supports auto refresh with programmable refresh intervals
- Supports self-refresh
- Supports Mobile SDRAM power-down mode
- Programmable immediate precharge or delayed precharge modes
- Supports 4 (programmable) open banks for performance; pages can be non-contiguous –Least Recently Used (LRU) algorithm used during page miss replacements

##### Static Memory Interface Features

- Supports asynchronous SRAMs and page-mode FLASHes
- Supports up to three sets of timing registers
- Configurable address width of up to 32 bits
- Limited synchronous SRAM and FLASH interface support
- Synchronous SRAM and FLASH frequency could be 1, 1/2, 1/3, 1/4, and so on of the AHB frequency

### 5.2 Architecture

This section provides a description about the functions and behavior under various conditions

### 5.2.1 Overview

The MSDR\_memctl can provide an interface between each of the following memory devices and an AMBA AHB 2.0 bus

- JEDEC-standard Mobile-SDRAM
- Asynchronous SRAM, with or without page-mode
- Asynchronous FLASH, with or without page-mode
- Limited synchronous SRAM and FLASH interface support

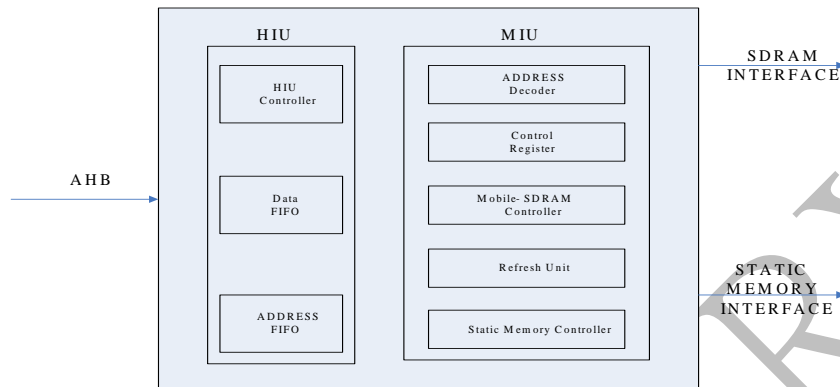


Fig. 5-1 Static Memory/Mobile SDRAM Controller Block Diagram

### 5.2.2 Block Descriptions

#### AMBA Host Interface Unit (HIU)

The AMBA Host Interface Unit (HIU) is the interface between the MSDR\_memctl and the AMBA Advanced High-performance Bus (AHB). The HIU generates memory read/write requests or control register read/write requests to the MIU block, which correspond to transfers on the AMBA bus; the HIU does not distinguish between an SDRAM request and an SRAM/FLASH request.

The HIU consists of the following sub-blocks:

- Address FIFO – Buffers the request of the AMBA AHB and sends memory/register access requests to the MIU; also contains some control information for a read/write transfer
- Write Data FIFO – Buffers write data to the memory and control registers
- Read Data FIFO – Buffers the read data from the memory
- HIU Control – Controls all the HIU sub-blocks by generating the control logic for read and write transfers

#### Memory Interface Unit (MIU)

The memory interface unit (MIU) is the interface for both SDRAM and Static memories; it generates appropriate address, data, and control signals corresponding to memory read/write transfers. The MIU contains two sets of modules, which are enabled depending on whether you choose the SDRAM or Static memory.

If you choose the SDRAM controller, the MIU includes the following modules:

- SDRAM controller – Generates the SDRAM control signals
- Refresh unit – Generates the SDRAM refresh request at appropriate intervals
- Address decoder – Generates the row, column, and bank addresses that correspond to the logical address provided by the host interface and Decodes and generates the address to SRAM/FLASH from the AHB address
- Control register – Holds the SDRAM control and configuration registers, and holds the control registers and timing registers for Static memories.
- Static control unit – Generates the SRAM/FLASH control signals

## 5.3 Registers

This section describes the control/status registers of the design.

### 5.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
MSDR_SCONR	0x00	W	0x1C4F68	SDRAM configuration register
MSDR_STMG0R	0x04	W	0x1E5D696	SDRAM timing register0
MSDR_STMG1R	0x08	W	0x7008	SDRAM timing register1
MSDR_SCTLR	0x0C	W	0x3009	SDRAM control register
MSDR_SREFR	0x10	W	0x410	SDRAM refresh register
MSDR_SCSLR0_LOW	0x14	W	0x0	Chip select register0 (lower 32bits)
MSDR_SCSLR1_LOW	0x18	W	0x5100	Chip select register1 (lower 32bits)
MSDR_SCSLR2_LOW	0x1C	W	0x6000	Chip select register2 (lower 32bits)
MSDR_SMSKR0	0x54	W	0x149	Mask register 0
MSDR_SMSKR1	0x58	W	0x249	Mask register 1
MSDR_SMSKR2	0x5C	W	0xC	Mask register 2
MSDR_CSREMAP0_LOW	0x84	W	0x50000000	Remap register for chip select0 (lower 32 bits)
MSDR_SMTMGR_SET0	0x94	W	0x1154C	Static memory timing register Set0
MSDR_SMTMGR_SET1	0x98	W	0x791950	Static memory timing register Set1
MSDR_SMTMGR_SET2	0x9C	W	0x1C1950	Static memory timing register Set2
MSDR_FLASH_TRPDR	0xA0	W	0xC8	FLASH memory tRPD timing register
MSDR_SMCTLR	0xA4	W	0x1201	Static memory control register
MSDR_EXN_MODE_REG	0xAC	W	0x0	Extended Mode Register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 5.3.2 Detail Register Description

#### MSDR\_SCONR

Address: Operational Base + offset( 0x00)

SDRAM Config Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	RW	0x1	Reserved.
19	RW	0x1	Reserved.
18	RW	0x1	Reserved.
17:15	RW	0x0	Reserved.
14:13	RW	0x1	Specifies SDRAM data width in bits; For all other SDRAMs, bits represent: 00 – 16 bits 01 – 32 bits 10 – 64 bits 11 – 128 bits
12:9	RW	0x7	Number of address bits for column address;



			15 – reserved 7-14 – correspond to 8-15 bits 0-6 – reserved
8:5	RW	0xb	Number of address bits for row address; 10-15 – correspond to 11-16 bits 0-10 – reserved
4:3	RW	0x1	Number of bank address bits; bit Values of 0-3 correspond to 1-4 bits, and therefore select 2-16 banks
2:0	-	-	Reserved.

**MSDR\_STMG0R**

Address: Operational Base + offset( 0x04)

SDRAM Timing Register0

bit	Attr	Reset Value	Description
25:22	RW	0x7	Active-to-active command period; values of 0-15 correspond to t <sub>rc</sub> of 1-16 clocks
31:27 21:18	RW	0x0 0x9	Exit self-refresh to active or auto-refresh command time; minimum time controller should wait after taking SDRAM out of self-refresh mode before issuing any active or auto-refresh commands; values 0-511 correspond to t <sub>xsr</sub> of 1-512 clocks
17:14	RW	0x7	Auto-refresh period; minimum time between two auto-refresh commands; values 0-15 correspond to t <sub>rcar</sub> of 1-16 clocks.
13:12	RW	0x1	For writes, delay from last data in to next precharge command; values 0-3 correspond to t <sub>wr</sub> of 1-4 clocks
11:9	RW	0x3	Precharge period; values of 0-7 correspond to t <sub>rp</sub> of 1-8 clocks
8:6	RW	0x2	Minimum delay between active and read/write commands; values 0-7 correspond to t <sub>rcd</sub> values of 1-8 clocks
5:2	RW	0x5	Minimum delay between active and precharge commands; values of 0-15 correspond to T <sub>RAS_MIN</sub> of 1-16 clocks
26 1:0	RW	0x0 0x2	Delay in clock cycles between read command and availability of first data 0 – 1 clock 1 – 2 clocks 2 – 3 clocks 3 – 4 clocks 4~7 – Reserved

**MSDR\_STMG1R**

Address: Operational Base + offset( 0x08)

SDRAM Timing Register1

bit	Attr	Reset Value	Description
31:22	-	-	Reserved.
21:20	RW	0x0	Reserved.
19:16	RW	0x7	Number of auto-refreshes during initialization; values 0-15 correspond to 1-16 auto-refreshes
15:0	RW	0x8	Number of clock cycles to hold SDRAM inputs stable after power up, before issuing any commands.

**MSDR\_SCTLR**

Address: Operational Base + offset( 0x0C)

SDRAM Control Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	R	0x0	Status of deep-power-down for mobile SDRAM. 0 – Mobile SDRAM not in deep-power-down 1 – Mobile SDRAM in deep power-down
19	RW	0x0	Reserved.
18	RW	0x0	Commands controller to update Mobile-SDRAM extended-mode register; once mode register update is done, controller automatically clears bit
17	RW	0x0	Reserved.
16:12	RW	0x3	Number of SDRAM internal banks to be open at any time; values of 1-16 correspond to 0-15 banks open.
11	R	0x0	Read only. When "1," indicates SDRAM is in self-refresh mode. When "self_refresh/deep_power_mode" bit (bit 1 of SCTLR) is set, it may take some time before SDRAM is put into self-refresh mode, depending on whether all rows or one row are refreshed before entering self-refresh mode defined by full_refresh_before_sr bit. Before gating clock in self-refresh mode, ensure this bit is set
9	RW	0x0	Set to 1, forces controller to do update of SDRAM mode register; bit is cleared by controller once it has finished mode register update
8:6	R/W	0x0	Indicates number of registers inserted in read data path for SDRAM in order to correctly latch data; values 0-7 indicate 0-7 registers
5	R/W	0x0	Controls number of refreshes after SDRAM is taken out of self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just 1 row before entering self-refresh mode
4	R/W	0x0	Controls number of refreshes before putting SDRAM into self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just one row before entering self-refresh mode
3	R/W	0x1	Determines when row is precharged: 0 – Immediate precharge; row precharged at end of read/write operation 1 – Delayed precharge; row kept open after read/write operations
2	R/W	0x0	Forces put SDRAM in power-down mode; bit 19 determines the type of power-down mode requested
1	R/W	0x0	Forces put SDRAM in self-refresh mode. Bit can be cleared by writing to this bit or by clear_sr_dp pin, generated by external power management unit
0	R/W	0x1	Forces initialize SDRAM; bit reset to 0 by MSDR_memctl once initialization sequence is complete

**MSDR\_SREFR**

Address: Operational Base + offset( 0x10)

SDRAM Refresh Interval Register

bit	Attr	Reset Value	Description
31:24	R	0x0	Reserved.
23:16	RW	0x0	Reserved.
15:0	RW	0x410	Number of clock cycles between consecutive refresh cycles;

### MSDR\_EXN\_MODE\_REG

Address: Operational Base + offset( 0xAC)

Extended Mode Register

bit	Attr	Reset Value	Description
31:13	-	-	Reserved.
12:7	RW	0x0	Always set to zero
6:5	RW	0x0	Driver strength of SDRAM output drivers. 0 - Full 1 - 1/2 2 - 1/4 (optional) 3 - 1/8 (optional)
4:3	RW	0x0	Maximum case temperature 0 - 70 1 - 45 2 - 15 3 - 85
2:0	RW	0x0	Self refresh coverage 0 - 4 banks 1 - 2 banks 2 - 1 bank 5 - 1/2 bank 6 - 1/4 bank Others - reserved

### MSDR\_SCSLR0

Address: Operational Base + offset( 0x14)

chip\_select\_register0

bit	Attr	Reset Value	Description
31:16	R	0x0000	Upper 16bits of base address for static memory bank0
15:0	-	-	Reserved.

### MSDR\_SCSLR1

Address: Operational Base + offset( 0x18)

chip\_select\_register1

bit	Attr	Reset Value	Description
31:16	R	0x5100	Upper 16bits of base address for static memory bank1
15:0	-	-	Reserved.

### MSDR\_SCSLR2

Address: Operational Base + offset( 0x1C)

chip\_select\_register0

bit	Attr	Reset Value	Description
31:16	R	0x6000	Upper 16bits of base address for Mobile SDRAM
15:0	-	-	Reserved.

### MSDR\_SMSKR0

Address: Operational Base + offset( 0x54)

Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	R	0x1	Register determines which timing parameters of memory connect to static memory bank0; primarily used for specifying static memories 0 – register set 0 , set in MSDR_SMTMGR_SET0 1 – register set 1 , set in MSDR_SMTMGR_SET1 2 – register set 2 , set in MSDR_SMTMGR_SET2
7:5	R	0x2	Type of memory connected to static memory bank0: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R	0x9	size of memory connected to static memory bank0; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

**MSDR\_SMSKR1**

Address: Operational Base + offset( 0x58)

Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	R	0x2	Register determines which timing parameters of memory connect to static memory bank1; primarily used for specifying static memories 0 – register set 0 , set in MSDR_SMTMGR_SET0 1 – register set 1 , set in MSDR_SMTMGR_SET1 2 – register set 2 , set in MSDR_SMTMGR_SET2
7:5	R	0x2	Type of memory connected to static memory bank1: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R	0x9	size of memory connected to static memory bank1; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

**MSDR\_SMSKR2**

Address: Operational Base + offset( 0x5C)

## Address Mask Registers

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:5	R	0x0	Type of memory connected to SDRAM 0 – SDRAM Others – Reserved
4:0	R	0xc	size of memory connected to SDRAM; 0 – No memory is connected to the chip select 1 – 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10 – 32MB 11 – 64MB 12 – 128MB 13~17 – Reserved

**MSDR\_CSREMAP0\_LOW**

Address: Operational Base + offset(0x84)

## REMAP REGISTER0

bit	Attr	Reset Value	Description
31:16	R	0x5000	Represent lower remap register bits for chip select0; compared with corresponding AHB address to generate chip select0; number of compared bits depends on size of memory selected by chip select0 (specified in mask register): 64KB – bits 31:16 compared 128 KB – bits 31:17 compared
15:0	-	-	Reserved.

**MSDR\_SMTMGR\_SET0**

Address: Operational Base + offset(0x94)

## Static Memory Timing Register - Set0

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set0
27	RW	0x0	Valid if register set0 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device:

			0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x0	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x5	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0xc	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

**MSDR\_SMTMGR\_SET1**

Address: Operational Base + offset(0x98)

Static Memory Timing Register – Set1

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set1
27	RW	0x0	Valid if register set1 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0xf	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold

			time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

**MSDR\_SMTMGR\_SET2**

Address: Operational Base + offset(0x9C)

Static Memory Timing Register – Set2

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set2
27	RW	0x0	Valid if register set2 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x3	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x4	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

**MSDR\_FLASH\_TRPDR**

Address: Operational Base + offset(0xA0)

FLASH Timing Register

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.



11:0	RW	0xC8	FLASH reset/power-down high to read/write delay; values correspond to sm_rp_n high to read/write delay minus one
------	----	------	------------------------------------------------------------------------------------------------------------------

**MSDR\_SMCTLR**

Address: Operational Base + offset(0xA4)

Static Memory Control Register

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:13	RW	0x0	Width of Static memory data bus controlled by Static memory register SET2: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
12:10	RW	0x4	Width of Static memory data bus controlled by Static memory register SET1: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
9:7	RW	0x4	Width of Static memory data bus controlled by Static memory register set 0: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
6:4	-	-	Reserved.
3:1	RW	0x0	FLASH write-protection mode; writing 0 forces FLASH memory bootblock to write protect; the three bits correspond to three register sets
0	RW	0x1	FLASH reset/power-down mode; after reset, controller internally performs a power-down for FLASH and then sets this bit to 1 to force FLASH to power-down mode during normal operation: 0 – Forces FLASH to power-down mode 1 – Takes FLASH out of power-down mode

**5.4 Functional Description****5.4.1 Operation****Basic Access Operation**

- 1、A page-hit sigle-cycle write:

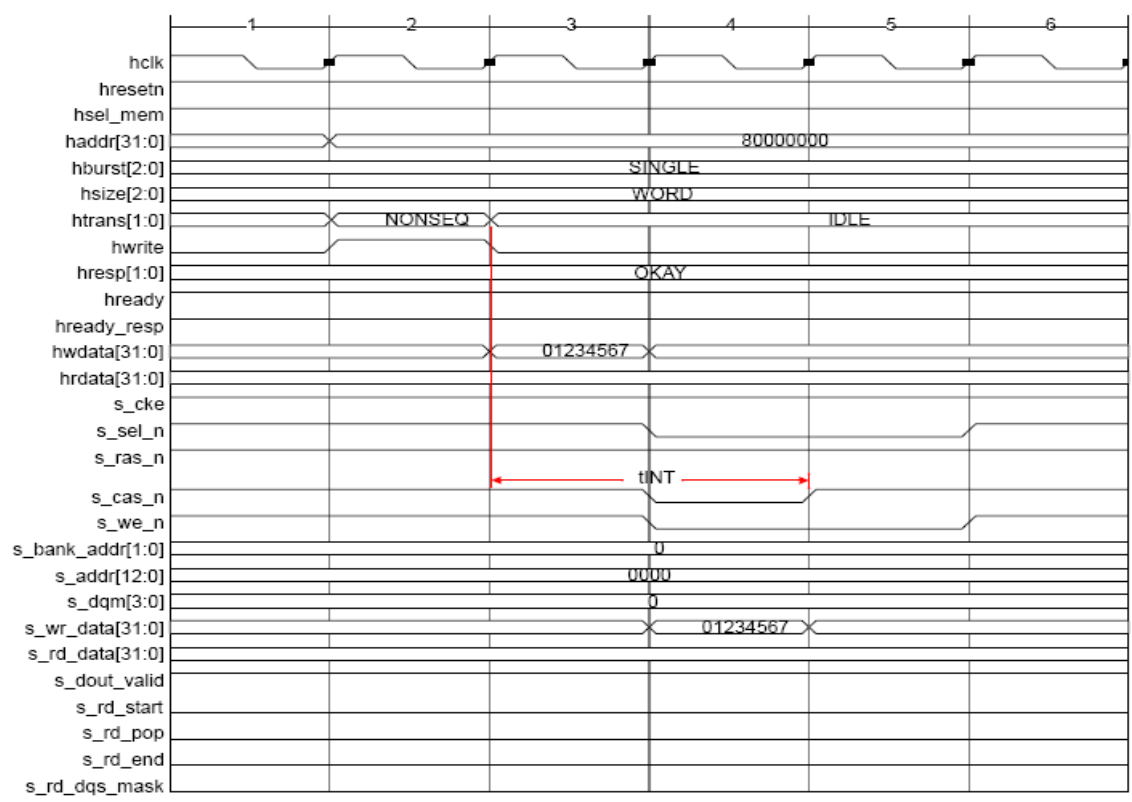


Fig. 5-2 Mobile SDRAM Page-Hit Single Write

## 2、A page-miss single-cycle write:

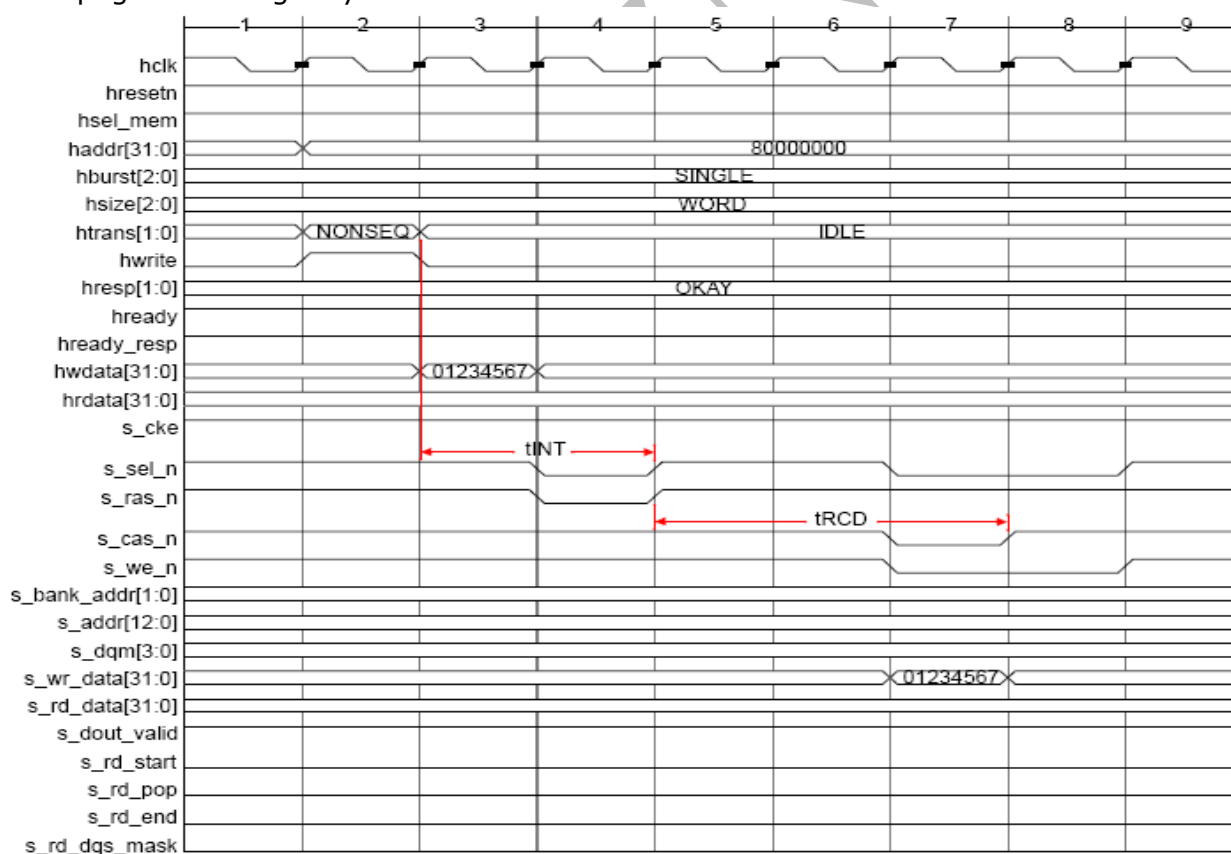


Fig. 5-3 Mobile SDRAM Page-Miss Single Write

## 3 、A page-hit burst write (hburst == INCR8)

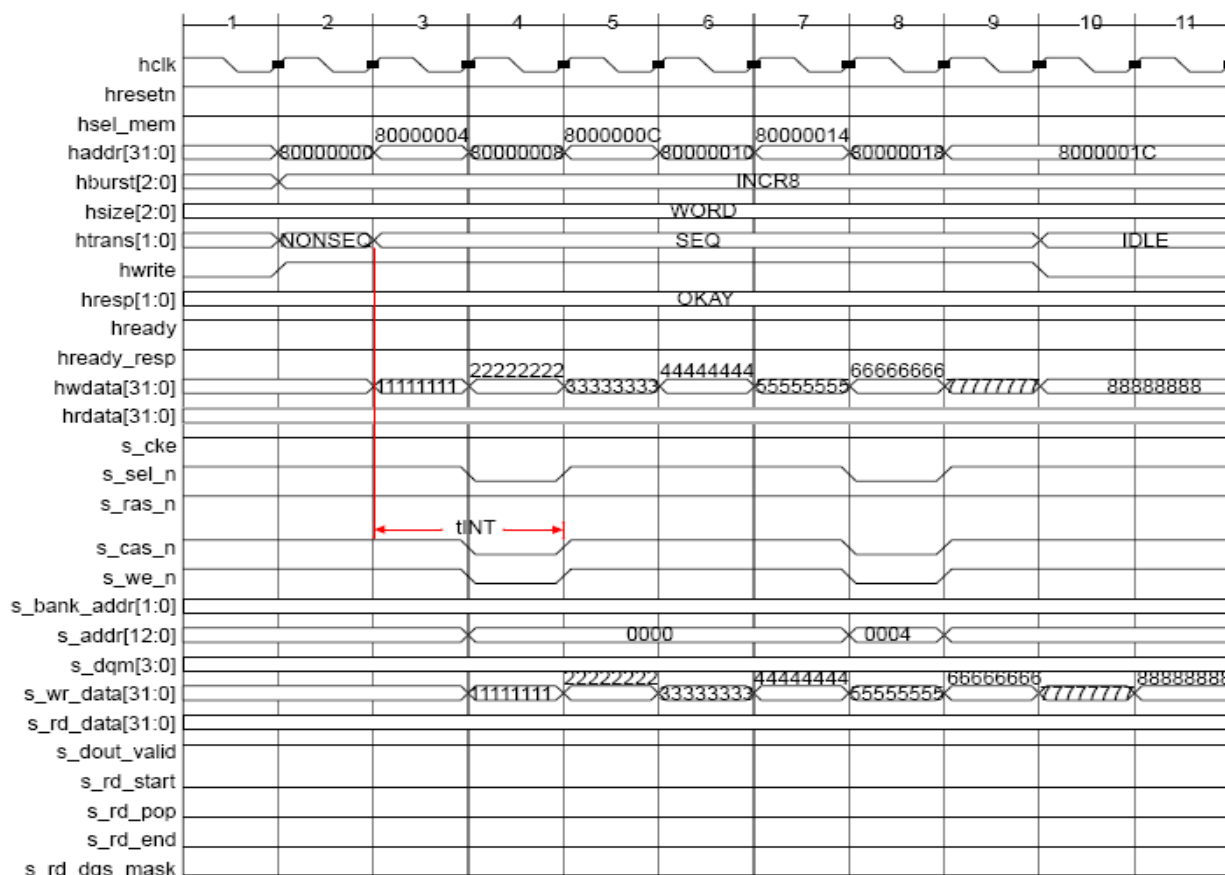


Fig. 5-4 Mobile SDRAM Page-Hit Burst Write

## 4、A page-hit single-cycle read:

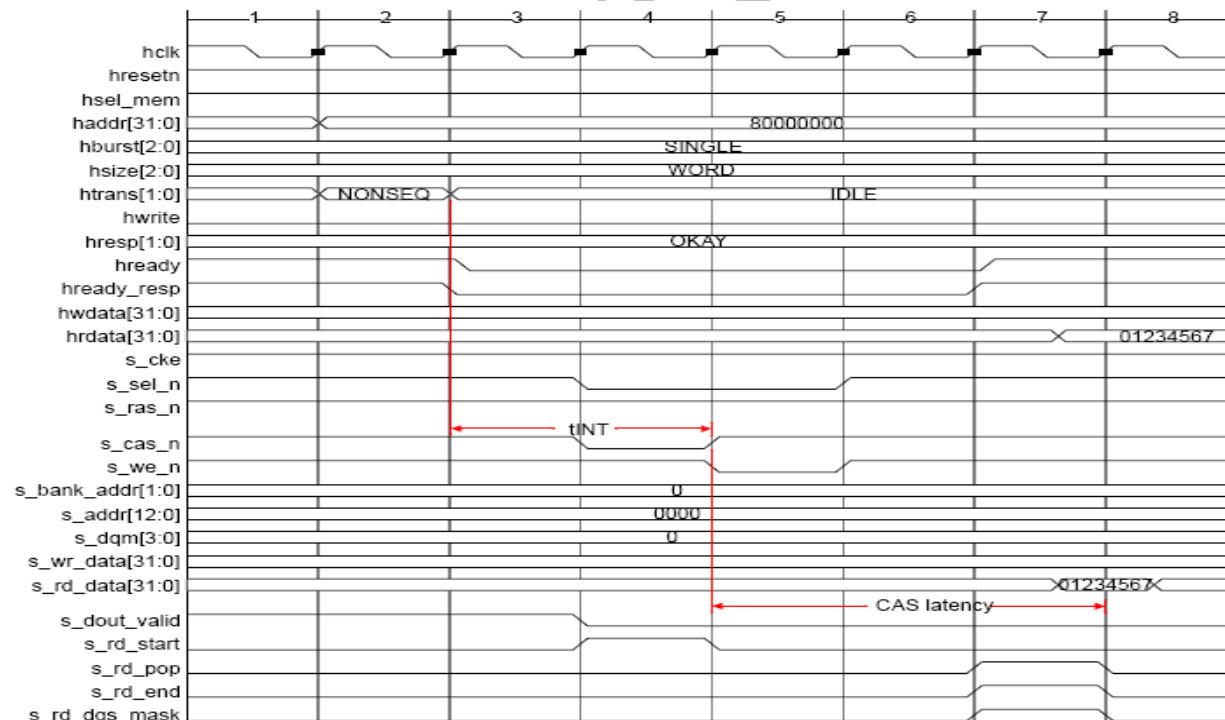


Fig. 5-5 Mobile SDRAM Page-Hit Single Read

## 5、a page-miss single-cycle read:

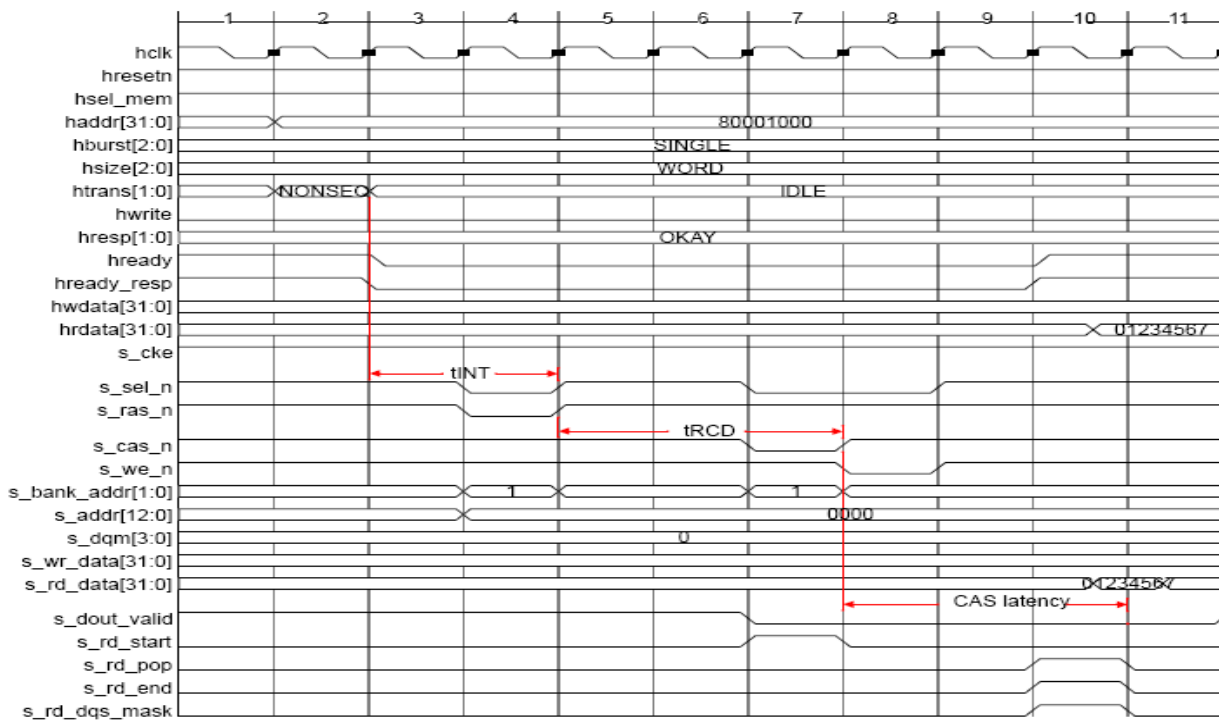


Fig. 5-6 Mobile SDRAM Page-Miss Single Read

## 6、A page-hit burst read:

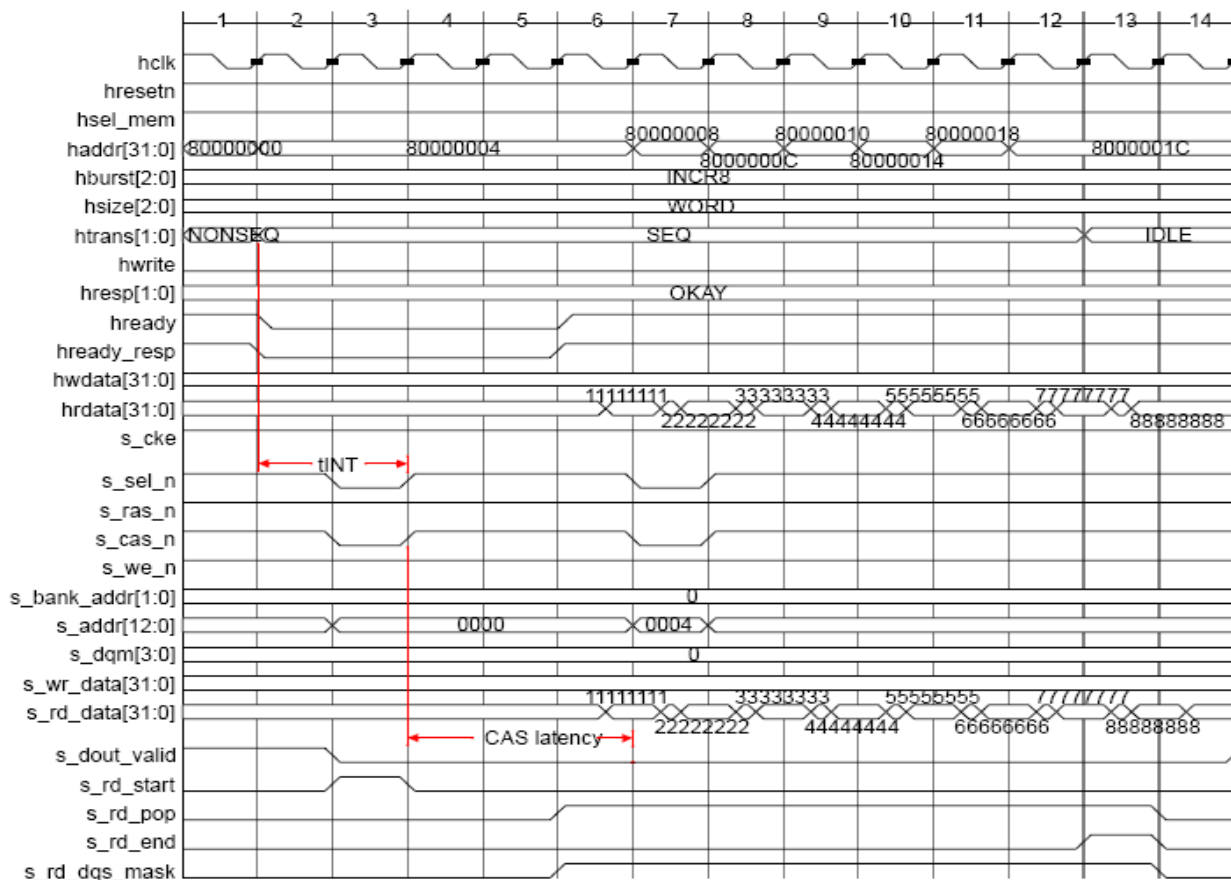


Fig. 5-7 Mobile SDRAM Page-Hit Burst Read

## Power-On Initialization

The SDR-SDRAM controller follows the JEDEC-recommended SDR-SDRAM power-on initialization sequence as follows:

1. Apply power and start clock; maintain a NOP condition at the inputs

2. Maintain stable power, stable clock, and NOP input conditions for a minimum of  $t_{init}$  clock cycles
3. Issue precharge commands for all banks of the device
4. Issue auto-refresh commands, depending on the value  $num\_init\_ref$  in the programmable register
5. Issue a set-mode register command to initialize the mode register the commands issued to the SDRAM by the controller during the power-on initialization are shown as followed
6. Issue a set-extended-mode register command to initialize the extended-mode register (valid only for Mobile-SDRAM)

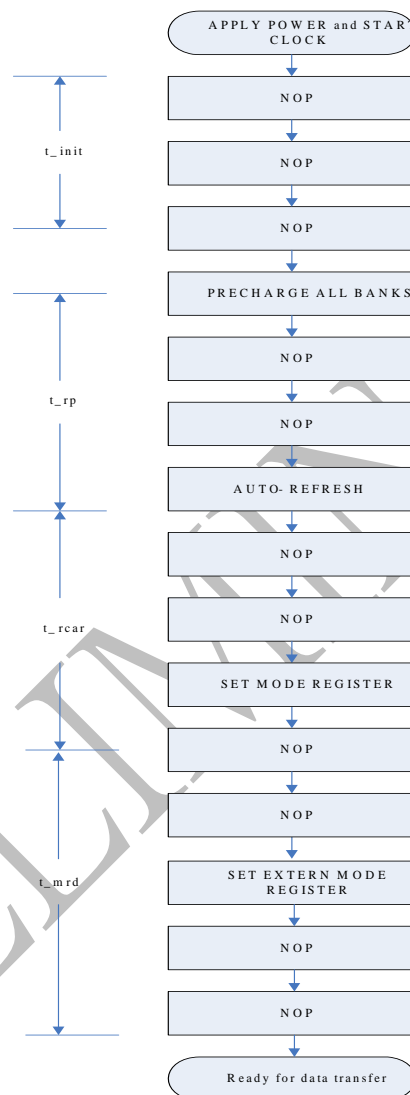


Fig. 5-8 Static Memory/Mobile SDRAM Controller power on sequence

## Read/Write Operation

The SDRAM Controller converts all AHB bursts to 4-word bursts on the SDRAM side. The memory bursts are concatenated to achieve continuous data flow for long AHB bursts. You can terminate the memory read/write burst with either a precharge command or terminate command, depending on which precharge mode – immediate precharge or delayed precharge – that you program. You can also terminate the write burst with a subsequent write burst

The SDRAM Controller supports two precharge modes – immediate precharge and delayed precharge. If you program for an immediate precharge mode, then the SDRAM Controller closes the open row after a read or write access. If you program for a delayed precharge mode, then the MSDR\_memctl keeps the row open after an access. The MSDR\_memctl can keep multiple banks open at the same time, depending on the value of

num\_open\_bank in the programmable register. When the number of open banks reaches the num\_open\_bank and an access to a new bank comes, the SDRAM Controller will close the oldest bank (the bank opened first) before opening the new bank.

### Set-Mode and Extended-Mode Register

The SDRAM Controller automatically sets the SDR-SDRAM mode register and extended-mode register during the power-up initialization. During normal operation, if you want to set the mode register or extended-mode register, you need to set set\_mode\_reg (bit 9 of SCTL) or exn\_mode\_reg\_update (bit 18 of SCTL) in the control register (SCTL) to 1. After the memory controller finishes the mode register setting, it clears the set\_mode\_reg or the exn\_mode\_reg\_update bit to 0.

The "burst length" field and the "burst type" field of the SDR-SDRAM-mode register are fixed by the SDRAM Controller to "010" (burst length 4) and "0" (sequential burst), respectively. The SDRAM Controller programs the "CAS latency" field and the "operating mode" field of the mode register according to the values provided by the user in the control and timing registers.

The MSDR\_memctl programs the extended-mode register of the Mobile-SDRAM according to the value provided by the user in the EXTN\_MODE\_REG (address 32'hxxxx\_xxAC).

### .Auto-Refresh

Auto-refresh commands are issued when the refresh control block issues refresh requests. During normal refresh operations, the SDRAM Controller always refreshes one row at a time. It is important for the user to program the tREF refresh interval register after a reset. If you need to refresh the SDRAM while a burst is active, normally the SDRAM Controller will issue the refresh command after the ongoing burst completes. However, if the ongoing burst is an AHB INCR burst, the SDRAM Controller will stop the burst, issue the refresh command, and then resume the burst.

The SDRAM Controller takes into account the maximum time it takes to complete a worst-case burst. This is the time to complete a read burst corresponding to an INCR16 burst on the AMBA bus, and with an AMBA-to-SDRAM data width ratio of 2:1. It is reasonable to assume 50 cycles for this worst-case burst, with 32 cycles for the data and the remaining 17 cycles for various latencies for the worst case.

The t\_ref value can be calculated using the following equation:

$$t_{ref} = \text{refresh\_period} / \text{clock\_period}$$

where refresh\_period = typically 7.8/15.6 s.

The tREF is the value of a free-running counter that the refresh logic in the SDRAM Controller operates on. When the count expires, the refresh logic gives a refresh request to the SDRAM controller.

Since the 64 ms refresh period is the same for most SDRAMs, the total number of rows in the SDRAM limits the minimum operating frequency for the MSDR\_memctl. While calculating the minimum frequency, use the following equation:

$$t_{REF} > 50 * (1/f)$$

Typically, the refresh cycle is 15.6 s or 7.8 s, depending on the refresh rate; The table is summarized as followed:

Number of rows	tREF	Min Frequency
64K	$(64\text{ms} - (50/f))/65536$	51Mhz
32K	$(64\text{ms} - (50/f))/32768$	26Mhz
16K	$(64\text{ms} - (50/f))/163904$	13Mhz
8K	$(64\text{ms} - (50/f))/8192$	6Mhz
4K	$(64\text{ms} - (50/f))/4096$	3Mhz
2K	$(64\text{ms} - (50/f))/2048$	1.5Mhz

The refresh logic in the MSDR\_memctl is inactive when the MSDR\_memctl forces the SDRAM into self-refresh or power-down mode.

### Self-Refresh

You can put the SDRAM into self-refresh mode, at which point the SDRAM retains data without external clocking and auto-refresh. The Figure as followed illustrates the

command sequence issued by the SDRAM controller to initiate, maintain, and exit the self-refresh mode

You can force the SDRAM Controller to enter self-refresh mode by programming bit 1 of the SDRAM control register (SCTLR) (Address 32'hxxxx\_xx0C). The SDRAM Controller forces the SDRAM to come out of self-refresh mode when bit 2 of the SCTLR is set to 0. You can set this bit to 0 by either programming the SDRAM control register or driving the clear\_sr\_dp pin high. You can use the clear\_sr\_dp pin when the code resides in the SDRAM, and the SDRAM itself is in self-refresh mode.

Bits 4 and 5 of the SCTLR specify the type of refresh done by the SDRAM Controller just prior to entering self-refresh mode and just after entering self-refresh mode.

Programming bit 4 of the SCTLR to 0 forces the SDRAM Controller to refresh only one row before putting the SDRAM into self-refresh mode. The default value of 1 forces the SDRAM Controller to perform auto-refreshes for all rows. Bit 5 does the same, except that it controls the refresh pattern just after coming out of self-refresh mode.

Since it takes time between programming the control register bit to the SDRAM entering self-refresh mode, the SDRAM Controller provides a read-only register bit (bit 11 of the SDRAM control register) to indicate that the SDRAM is already in self-refresh mode. If you want to gate off the clock to the SDRAM Controller when the SDRAM is in self-refresh mode, you should ensure this bit is set to 1 before you stop the clock.

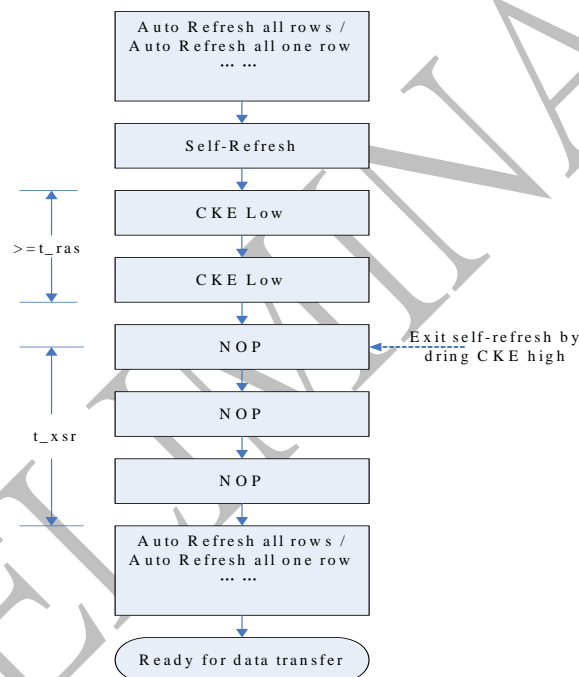


Fig. 5-9 Mobile SDRAM self\_refresh flow

The SDRAM must remain in self-refresh mode for a minimum period of  $t_{ras}$  and can remain in self-refresh mode for an indefinite period of time. After the SDRAM exits self-refresh mode, the SDRAM Controller issues NOP commands for  $t_{xsr}$  before it issues any other command. The  $t_{ras}$  and  $t_{xsr}$  are programmable register values and have default values. These registers can be programmed only once after reset.

When an AHB read/write request to the SDRAM occurs while the SDRAM is in self-refresh mode, the SDRAM Controller generates dummy ready signals to the AHB without accessing external memory; no error response is generated on the AHB bus.

### Power-Down

The SDRAM can be put into power-down mode to save power. There are two ways to force the MSDR\_memctl to put the SDRAM in power-down mode:

- Program bit 2 of SCTLR to 1; should be 0 to bring the SDRAM out of power-down mode.
- Use the power-down input pin; can be driven by an external power management unit; the SDRAM will be in power-down mode as long as this signal stays high

The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit power-down mode



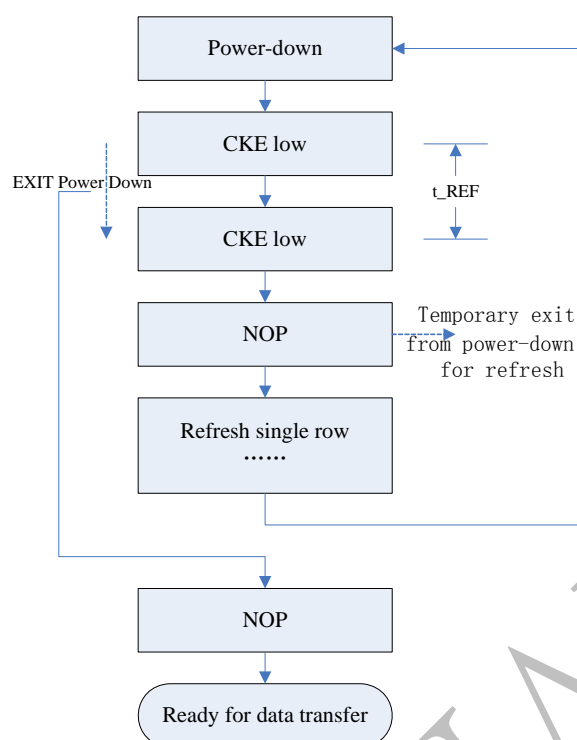


Fig. 5-10 Static Memory/Mobile SDRAM Controller power\_off mode

When in SDRAM power-down mode, the SDRAM Controller keeps switching the device back and forth between power-down and refresh mode. It remains in power-down for a  $t_{ref}$  period of time, then comes out of power-down and does a single-row refresh; then it again goes into power-down mode.

The SDRAM Controller keeps the SDRAM in this periodical power-down/refresh/power-down sequence until it is commanded to exit power-down mode (by programming bit 2 of SCTL0 to 0). When an AHB read/write request to the SDRAM occurs while the SDRAM is in power-down mode, the SDRAM Controller brings the SDRAM out of power-down mode and issues the read/write access to the SDRAM. The SDRAM Controller then puts the SDRAM back to power-down mode after the read/write access.

## Chapter 6 NAND Flash Controller

### 6.1 Design Overview

#### 6.1.1 Overview

The Nand Flash Controller (NANDC) is used for controlling data transfer to/from nand flash device. Control information is written from a master (CPU) over the AHB bus to the NANDC. It supports transferring data to/from flash in two ways: internal dma and directly bypass. Hardware ECC, which support 16bit/1kbyte (compatible with 8bit/512byte) or 24bit/1kbyte (compatible with 12bit/512byte) bch error correction, can correct error bits at any position in one codeword.

#### 6.1.2 Features

- AMBA AHB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation
- Support internal-dma-transfer-finish interrupt interface and flash-busy-to-ready interrupt interface to interrupt controller
- 2K sram memory space, divided to 2 sections, is used for internal dma transfer
- 8bits data interface to flash device
- Support 8 flash devices at most
- Selectable 16bit/1kbyte BCH error correction and 24bit/1kbyte BCH error correction. While the former mode is compatible with 8bit/512byte BCH ECC, and the latter mode is compatible with 12bit/512byte BCH ECC
- Support FF code auto correction
- Support LBA-NAND-FLASH which can detect and correct error by device
- Support configurable number of spare registers for LBA-NAND-FLASH
- Support data transferred in internal dma way or directly bypass way to the flash device
- Data can be transferred to/from sram that is free from internal dma use

For detailed information about NAND FLASH controller, please refer to **RK281x NAND Flash Controller.pdf**.

## Chapter 7 SD/MMC Host Controller

### 7.1 Design Overview

#### 7.1.1 Overview

The SD/MMC Host Controller is designed to support Secure Digital memory (SD mem - version 2.00), Secure Digital I/O (SDIO-version 1.10), Multimedia Cards (MMC-version 4.2). There are two SD/MMC Host Controllers connected with ARMD bus, SDMMC0 support SD Card(1/4bit), SDIO, MMC(1/4/8bit), and SDMMC1 support SD Card(1/4bit), SDIO, MMC(1/4bit).

#### 7.1.2 Features

- Supports AMBA AHB interface
- Supports DMA controller for data transfers
- Supports interrupt output
- Supports SD version2.0 except SPI mode
- Supports MMC version4.2 except SPI mode
- Supports SDIO version1.1
- Supports programmable baud rate.
- Provides individual clock control to selectively turn ON or OFF clock to a card
- Supports power management and power switch. Provides individual power control to selectively turn ON or OFF power to a card

### 7.2 Architecture

This section provides a description about the functions and behavior under various conditions.

#### 7.2.1 Block Diagram

The SD/MMC controller consists of the following main functional blocks, which are illustrated in Fig. 6-1.

- Bus Interface Unit (BIU) – Provides AMBA AHB and DMA interfaces for register and data read/writes.
- Card Interface Unit (CIU) – Takes care of the SD\_MMC protocols and provides clock management.

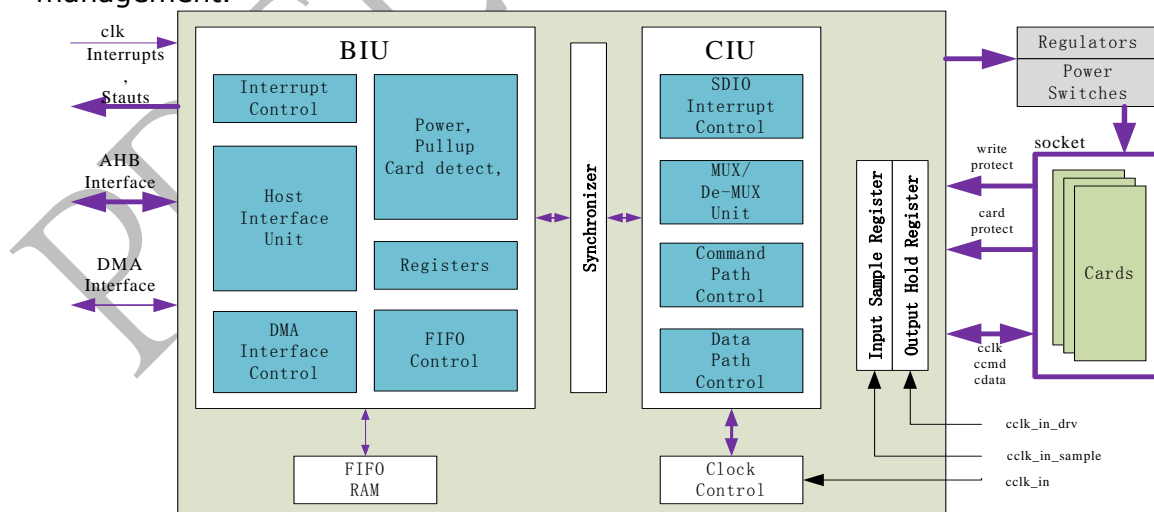


Fig. 7-1 SD/MMC Host Controller Block Diagram

#### 7.2.2 Block Descriptions

The SD/MMC comprises with:

##### **BIU(Bus Interface Unit)**

The BIU provides the following functions:

### Host Interface

The Host Interface Unit (HIU) is an AHB slave interface, which provides the interface between the SD/MMC controller and the host bus.

### DMA Interface

DMA signals interface the SD/MMC controller to an external AHB DMA controller to reduce the software overhead during FIFO data transfers. The DMA request/acknowledge handshake is used for only data transfer. The DMA interface provides a connection to the DW\_dmac.

On seeing the DMA request, the DMA controller initiates accesses through the host interface to read or write into the data FIFO. The SD/MMC controller has FIFO transmit/receive watermark registers that you can set, depending on system latency. The DMA interface asserts the request in the following case:

- Read from a card when the data FIFO word count exceeds the Rx-Watermark level
- Write to a card when the FIFO word count is less than or equal to the Tx-Watermark level

When the DMA interface is enabled, you can use normal host read/writes to access the data FIFOs.

### Interrupt Control

The interrupt controller unit generates an interrupt that depends on the controller raw interrupt status, the interrupt-mask register, and the global interrupt-enable register bit. Once an interrupt condition is detected, it sets the corresponding interrupt bit in the raw interrupt status register. The raw interrupt status bit stays on until the software clears the bit by writing a 1 to the interrupt bit; a 0 leaves the bit untouched.

The interrupt port is active only when any bit in the raw interrupt status register is active, the corresponding interrupt mask bit is 1, and the global interrupt enable bit is 1.

#### Notes:

\* Before enabling the interrupt, it is always recommended that you write 32'hffff\_ffff to the raw interrupt status register in order to clear any pending unserved interrupts. When clearing interrupts during normal operation, ensure that you clear only the interrupt bits that you serviced.

Table 7-1 SD/MMC Bits in Interrupt Status Register

bit	Interrupt	Description
31:16	SDIO Interrupts	Interrupts from SDIO cards; one bit for each card. Bit[31] corresponds to Card[15].
15	End Bit Error (read)/Write no CRC (EBE)	Error in end-bit during read operation, or no data CRC or negative CRC received during write operation. Note: For MMC CMD19, there may be no CRC status returned by the card. Hence, EBE is set for CMD19. The application should not treat this as an error.
14	Auto Command Done (ACD)	Stop/abort commands automatically sent by card unit and not initiated by host; similar to Command Done (CD) interrupt. Attention – Recommendation: Software typically need not enable this; Data Transfer Over (DTO) interrupt that comes after this interrupt determines whether data transfer has correctly completed.
13	Start Bit Error (SBE)	Error in data start bit when data is read from a card. In 4-bit mode, if all data bits do not have stat bit, then this error is set.
12	Hardware Locked write Error (HLE)	During hardware-lock period, write attempted to one of locked registers.
11	FIFO Underrun/Overrun Error (FRUN)	Host tried to push data when FIFO is full, or host tried to read data when FIFO was empty. Typically this should not happen, except due to error in software.

		Card unit never pushes data into FIFO when FIFO is full, and pop data when FIFO is empty.
10	Data Starvation by Host Timeout (HTO)	To avoid data loss, card clock out (cclk_out) is stopped if FIFO is empty when writing to card, or FIFO is full when reading from card. Whenever card clock is stopped to avoid data loss, data-starvation timeout counter is started with data-timeout value. This interrupt is set if host does not fill data FIFO during write to card, or does not read from FIFO during read from card before timeout period. Even after timeout, card clock stays in stopped state, with CIU state machines waiting. It is responsibility of host to push or pop data into FIFO upon interrupt, which automatically restarts cclk_out and card state machines. Even if host wants to send stop/abort command, it still needs to ensure it has to push or pop FIFO so that clock starts in order for stop/abort command to send on cmd signal along with data that is sent or received on data line.
9	Data Read Timeout(DRTO)	Data timeout occurred. Data Transfer Over (DTO) also set if data timeout occurs.
8	Response Timeout(RTO)	Response timeout occurred. Command Done (CD) also set if response timeout occurs. If command involves data transfer and when response times out, not data transfer is attempted by SD/MMC controller.
7	Data CRC Error (DCRC)	Received Data CRC does not match with locally-generated CRC in CIU.
6	Response CRC Error (RCRC)	Response CRC does not match with locally-generated CRC in CIU.
5	Receive FIFO Data Request (RXDR)	Interrupt set during read operation from card when FIFO level is greater than Receive-Threshold level. Attention – Recommendation: In DMA modes, this interrupt should not be enabled. ISR, in non-DMA mode: pop RX_WMark + 1 data from FIFO
4	Transmit FIFO Data Request (TXDR)	Interrupt set during write operation to card when FIFO level reaches less than or equal to Transmit-Threshold level. Attention – Recommendation: In DMA modes, this interrupt should not be enabled. ISR in non-DMA mode: if (pending_bytes > (FIFO_DEPTH - TX_WMark)) push (FIFO_DEPTH - TX_WMark) data into FIFO else push pending_bytes data into FIFO
3	Data Transfer Over (DTO)	Data transfer completed, even if there is Start Bit Error or CRC error. Attention – Recommendation: In non-DMA mode, when data is read from card, on seeing interrupt, host should read any pending data from FIFO. In DMA mode, DMA controllers guarantee FIFO is flushed before interrupt. Note – DTO bit is set at the end of the last data block, even if the device asserts MMC busy after the

		last data block.
2	Command Done (CD)	Command sent to card and got response from card, even if Response Error or CRC error occurs.
1	Response Error (RE)	Error in received response set if one of following occurs: <ul style="list-style-type: none"> <li>● Transmission bit !=0</li> <li>● Command index mismatch</li> <li>● End-bit !=1</li> </ul>
0	Card-Detect (CDT)	When one or more cards inserted or removed, this interrupt occurs. Software should read card-detect register (CDETECT, 0x50) to determine current card status. Attention – Recommendation: After power-on and before enabling interrupts, software should read card detect register and store it in memory. When interrupt occurs, it should read card detect register and compare it with value stored in memory to determine which card were removed/inserted. Before exiting ISR, software should update memory with new card-detect value.

**Note** – The SDIO Interrupts, Receive FIFO Data Request (RXDR), and Transmit FIFO Data Request (TXDR) are set by level-sensitive interrupt sources. Therefore, the interrupt source should be first cleared before you can clear the interrupt bit of the Raw Interrupt register. For example, on seeing the Receive FIFO Data Request (RXDR) interrupt, the FIFO should be emptied so that the “FIFO count greater than the RX-Watermark” condition, which triggers the interrupt, becomes inactive. The reset of the interrupts are triggered by a single clock-pulse-width source.

### Register Bank

The register unit is part of the bus interface unit (BIU); it provides read and write access to the registers.

All registers reside in the BIU clock domain. When a command is sent to a card by setting the start\_bit, which is bit[31] of the CMD register, all relevant registers needed for the CIU operation are transferred to the CIU block. During this time, the register that are transferred from the BIU to the CIU should not be written. The software should wait for the hardware to clear the start bit before writing to these registers again.

Once a command start is issued by setting the start\_bit of the CMD register, the following registers cannot be reprogrammed until the command is accepted by the CIU: CMD/CMDARG/BYTCNT/BLKSIZ/CLKDIV/CLKENA/CLKSRC/TMOU/CTYPE.

The hardware resets the start\_bit once the CIU accepts the command. If a host write to any of these registers is attempted during this lock time, then the write ignored and the hardware lock error bit is set in the raw interrupt status register. Additionally, if the interrupt is enabled and not masked for a hardware lock error, then an interrupt is sent to the host.

When the CIU is in an idle state, it typically takes the following number of clocks for the command handshake, where clk is the BIU clock and cclk\_in is the CIU clock:  $3(\text{clk}) + 3(\text{cclk\_in})$

Once a command is accepted, you can send another command to the CIU – which has a one-deep command queue – under the following conditions:

- If the previous command was not a data transfer command, the new command is sent to the SD\_MMC card once the previous command completes.
- If the previous command is a data transfer command and if wait\_prvdata\_complete (bit[13]) of the CMD register is set for the new command, the new command is sent to the SD\_MMC card only when the data transfer completes.
- If the wait\_prvdata\_complete is 0, then the new command is sent to the SD\_MMC card as soon as the previous command is sent. Typically, you should use this only to stop or abort a previous data transfer or query the card status in the middle of a data transfer.



### FIFO Controller Unit

The FIFO controller interfaces the internal FIFO to the host/DMA interface and the card controller unit.

The FIFO maps to an address offset that is greater than or equal to 0x100. While accessing the FIFO, you can set the address to any value 0x100 or greater; partial access to the FIFO is also supported. If the AHB is 32 bits, you can access the FIFO with two 16-bit accesses. The lower address should be accessed first, and then the higher address, such as 0x100, and then 0x101.

### Power/Pullup control and card detection Unit

The register unit has registers that control the power and MMC open-drain pullup. Power to each card can be selectively turned on or off. Additionally, there are two 4-bit card-voltage control signals that can control the voltages of two voltage regulators. The control register has an enable\_OD\_pullup bit, which is used to turn on the open-drain-mode pullup during MMC initialization.

The card detection unit looks for any changes in the card-detect signals for card insertion or card removal. It filters out the debounces associated with mechanical insertion or removal, and generates one interrupt to the host. You can program the debounce filter value.

On power-on, the controller should read in the card\_detect port and store the value in the memory. Upon receiving a card-detect interrupt, it should again read the card\_detect port and XOR with the previous card-detect status to find out which card has interrupted.

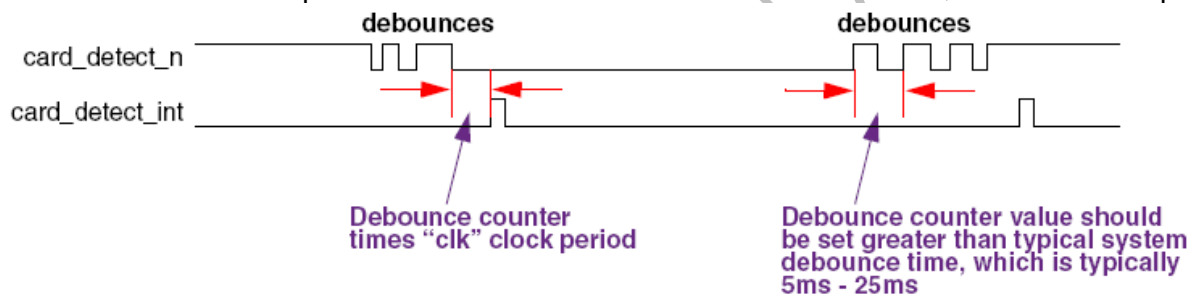


Fig. 7-2 SD/MMC Card Detect timing waveform

### CIU(Card Interface Unit)

The Card Interface Unit(CIU) interfaces with the BIU and the SD/MMC card or devices. The host writes command parameters to the SD/MMC controller BIU control registers, and these parameters are then passed to the CIU. Depending on control register values, the CIU generates SD/MMC command and data traffic on a selected card bus according to SD/MMC protocol.

The following software restrictions should be met for proper CIU operation:

- Only one data transfer command can be issued at a time.
- During an open-ended card write operation, if the card clock is stopped because the FIFO is empty, the software must first the data into the FIFO and start the card clock. It can then issue only a stop/abort command to the card.
- During an SDIO card transfer, if the card function is suspended and the software wants to resume the suspended transfer, it must first reset the FIFO and start the resume command as if it were a new data transfer command.
- When issuing card reset commands (CMD0, CMD15 or CMD52\_reset) while a card data transfer is in progress, the software must set the stop\_abort\_cmd bit in the CMD register so that the SD/MMC controller can stop the data transfer after issuing the card reset command.
- When the data end bit error is set in the RINTSTS register, the SD/MMC controller does not guarantee SDIO interrupts. The software should ignore the SDIO interrupts and issue the stop/abort command to the card, so that the card stops sending the read data.
- If the card clock is stopped because the FIFO is full during a card read, the software should read at two FIFO locations to start the card clock.



The CIU block consists of the following primary functional blocks:

#### **Command path**

The command path performs the following functions:

- Loads clock parameters
- Loads card command parameters
- Sends commands to card bus
- Receives response from card bus
- Sends responses to BIU
- Drives the P-bit on command line

A new command is issued to the SD/MMC controller by programming the BIU registers and setting the start\_cmd bit in the CMD register. The BIU asserts start\_cmd, which indicates that a new command is issued to the SD/MMC controller. The command path loads this new command (command. Command argument, timeout) and sends an acknowledge to the BIU by asserting cmd\_taken.

Once the new command is loaded, the command path state machine sends a command to the SD\_MMC bus-including the internally generated CRC7-and receives a response, if any. The state machine then sends the received response and signals to the BIU that the command is done, and then waits for eight clocks before loading a new command.

#### **1. Load Command Parameters**

One of the following commands or responses is loaded in the command path:

- New command from BIU – When start\_cmd is asserted, then the start\_cmd bit is set in the Command register.
- Internally-generated auto-stop command – When the data path ends, the stop command request is loaded.
- IRQ response with RCA 0x000 – When the command path is waiting for an IRQ response from the MMC card and a “send irq response” request is signaled by the BIU, then the send\_irq\_response bit is set in the control register.

Loading a new command from the BIU in the command path depends on the following Command register bit settings:

- update\_clock\_registers\_only – If this bit is set in the Command register, the command path updates only the clock enable, clock divider, and clock source registers. If this bit is not set, the command path loads the command, command argument, and timeout registers; it then starts processing the new command.
- wait\_prvdata\_complete – If this bit is set, the command path loads the new command under one of the following conditions:
  - 1) Immediately, if the data path is free (that is, there is no data transfer in progress), or if an open-ended data transfer is in progress (byte\_count = 0).
  - 2) After completion of the current data transfer, if a predefined data transfer is in progress.

#### **2. Send Command and Receive Response**

Once a new command is loaded in the command path – update\_clock\_registers\_only bit is unset – the command path state machine sends out a command on the SD\_MMC bus. The command path state machine is illustrated in Fig. 6-3.

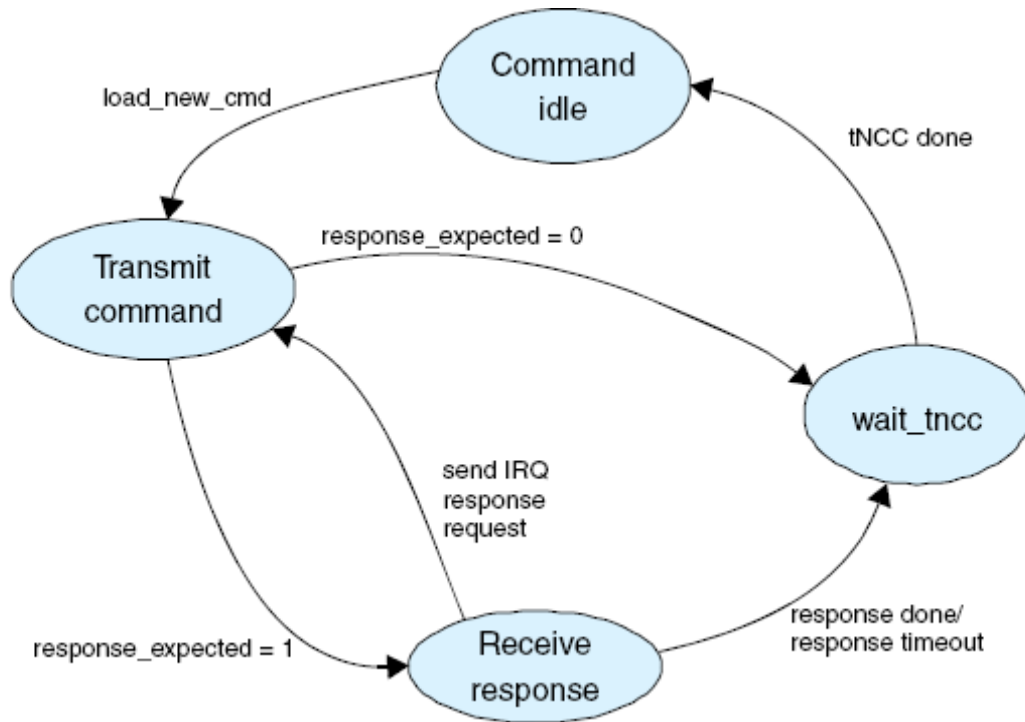


Fig. 7-3 SD/MMC Command Path State Machine

The command path state machine performs the following functions, according to Command register bit values:

- **send\_initialization** – Initialization sequence of 80 clocks is sent before sending the command.
- **response\_expected** – Response is expected for the command. After the command is sent out, the command path state machine receives a 48-bit or 136-bit response and sends it to the BIU. If the start bit of the card response is not received within the number of clocks programmed in the timeout register, then the response timeout and command done bit is set in the Raw Interrupt Status register as a signal to the BIU. If the response-expected bit is not set, the command path sends out a command and signals a response done to the BIU; that is, the command done bit is set in the Raw Interrupt Status register.
- **response\_length** – If this bit is set, a 136-bit response is received; if it is not set, a 48-bit response is received.
- **check\_response\_crc** – If this bit is set, the command path compares CRC7 received in the response with the internally-generated CRC7. If the two do not match, the response CRC error is signaled to the BIU; that is, the response CRC error bit is set in the Raw Interrupt Status register.

### 3. Send Command and Receive Response

If the **response\_expected** bit is set in the Command register, the received response is sent to the BIU. The Response0 register is updated for a short response, and the Response3, Response2, Response1, and Response0 registers are updated on a long response, after which the Command Done bit is set. If the response is for an **auto\_stop** command sent by the CIU, the response is saved in the Response1 register, after which the Auto Command Done bit is set.

Additionally, the command path checks for the following:

- Transmission bit = 0
- Command index matches command index of the sent command
- End bit = 1 in received card response

The command index is not checked for a 136-bit response or if the **check\_response\_crc** bit is unset. For a 136-bit response and reserved CRC 48-bit responses, the command index is reserved—that is, 111111.

### 4. Driving P-bit on CMD line

The command path drives a P-bit = 1 on the CMD line between two commands if a response is not expected. If a response is expected, the P-bit is driven after the response

is received and before the start of the next command; this is done by asserting both `ccmd_out` and `ccmd_out_en`.

During initialization, the software should set the `ccmd_od_pullup_en` bit, which indicates an open-drain mode, during which the controller drives only a 0 or high-impedance (Z) on the command bus; a hard 1 is never driven in open-drain mode.

### Data Path

The data path block pops the data FIFO and transmits data on `cdata_out` during a write data transfer, or it receives data on `cdata_in` and pushes it into the FIFO during a read data transfer. The data path loads new data parameters—that is, data expected, read/write data transfer, stream/block transfer, block size, byte count, card type, timeout registers—whenever a data transfer command is not in progress.

If the `data_expected` bit is set in the Command register, the new command is a data transfer command and the data path starts one of the following:

- Transmit data if the read/write bit = 1
- Data receive if read/write bit = 0

#### 1. Data Transmit

The data transmit state machine, illustrated in Fig. 6-4, starts data transmission two clocks after a response for the data write command is received; this occurs even if the command path detects a response error or response CRC error. If a response is not received from the card because of a response timeout, data is not transmitted. Depending upon the value of the `transfer_mode` bit in the Command register, the data transmit state machine puts data on the card data bus in a stream or in block(s).

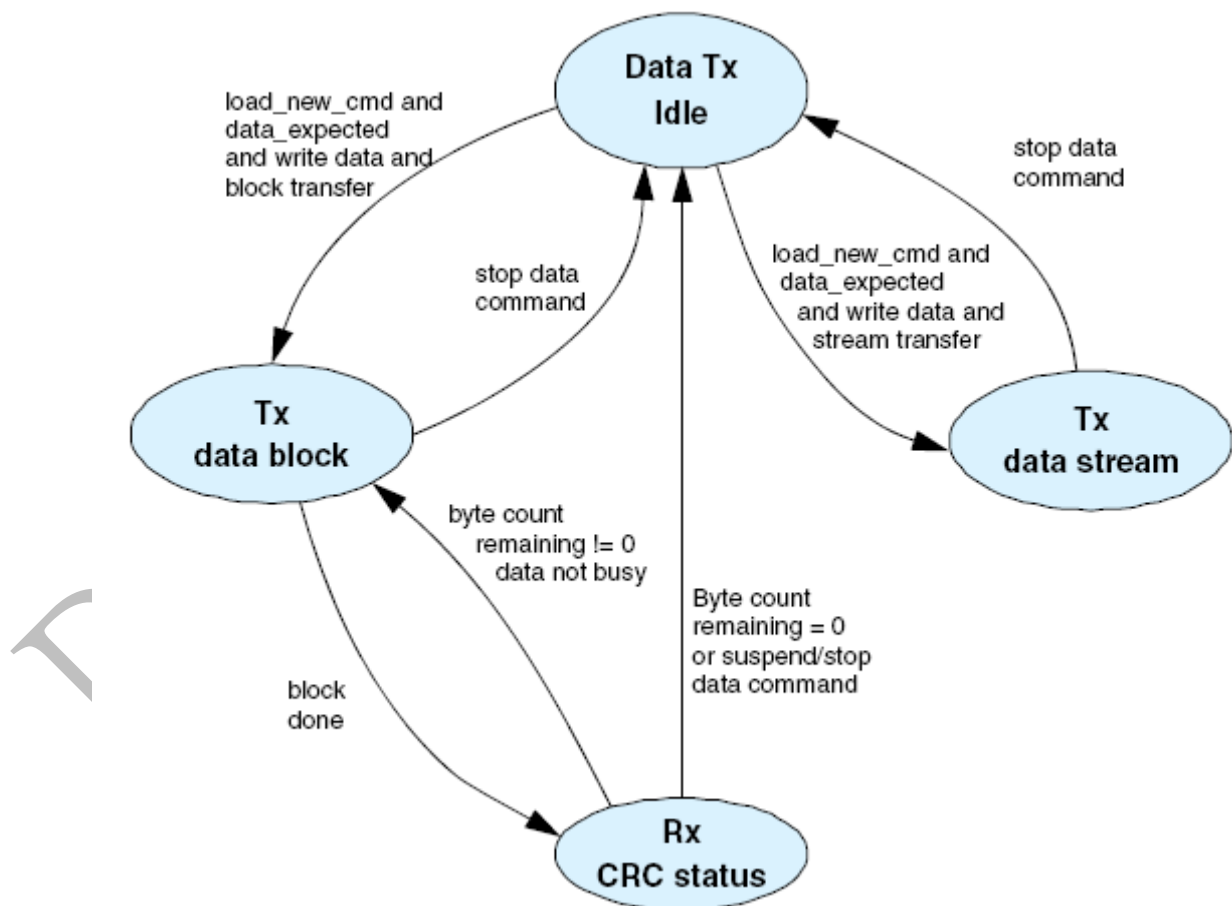


Fig. 7-4 SD/MMC Data Transmit State Machine

#### 2. Stream Data transmit

If the `transfer_mode` bit in the Command register is set to 1, it is a stream-write data transfer. The data path pops the FIFO from the BIU and transmits in a stream to the card data bus. If the FIFO becomes empty, the card clock is stopped and restarted once data is available in the FIFO.

If the `byte_count` register is programmed to 0, it is an open-ended stream-write data transfer. During this data transfer, the data path continuously transmits data in a stream until the host software issues a stop command. A stream data transfer is terminated when the end bit of the stop command and end bit of the data match over two clocks.

If the `byte_count` register is programmed with a non-zero value and the `send_auto_stop` bit is set in the Command register, the stop command is internally generated and loaded in the command path when the end bit of the stop command occurs after the last byte of the stream write transfer matches. This data transfer can also terminate if the host issues a stop command before all the data bytes are transferred to the card bus.

### 3. Single Block Data

If the `transfer_mode` bit in the Command register is set to 0 and the `byte_count` register value is equal to the value of the `block_size` register, a single-block write-data transfer occurs. The data transmit state machine sends data in a single block, where the number of bytes equals the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the `card_num` value in the Command register – is set for a 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted for 1, 4, or 8 data lines, respectively.

After a single data block is transmitted, the data transmit state machine receives the CRC status from the card and signals a data transfer to the BIU; this happens when the data-transfer-over bit is set in the RINTSTS register.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register.

Additionally, if the start bit of the CRC status is not received by two clocks after the end of the data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the RINTSTS register.

### 4. Multiple Block Data

A multiple-block write-data transfer occurs if the `transfer_mode` bit in the Command register is set to 0 and the value in the `byte_count` register is not equal to the value of the `block_size` register. The data transmit state machine sends data in blocks, where the number of bytes in a block equals the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the `card_num` value in the Command register – is set to 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted on 1, 4, or 8 data lines, respectively.

After one data block is transmitted, the data transmit state machine receives the CRC status from the card. If the remaining `byte_count` becomes 0, the data path signals to the BIU that the data transfer is done; this happens when the data-transfer-over bit is set in the RINTSTS register.

If the remaining data bytes are greater than 0, the data path state machine starts to transmit another data block.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register, and continues further data transmission until all the bytes are transmitted.

Additionally, if the CRC status start bit is not received by two clocks after the end of a data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the RINTSTS register; further data transfer is terminated.

If the `send_auto_stop` bit is set in the Command register, the stop command is internally generated during the transfer of the last data block, where no extra bytes are transferred to the card. The end bit of the stop command may not exactly match the end bit of the CRC status in the last data block.

If the block size is less than 4, 16, or 32 for card data widths of 1 bit, 4 bits, or 8 bits, respectively, the data transmit state machine terminates the data transfer when all the data is transferred, at which time the internally generated stop command is loaded in the command path.

If the `byte_count` is 0 – the block size must be greater than 0 – it is an open-ended

block transfer. The data transmit state machine for this type of data transfer continues the block-write data transfer until the host software issues a stop or abort command.

### 5. Data Receive

The data-receive state machine, illustrated in Fig. 6-5, receives data two clock cycles after the end bit of a data read command, even if the command path detects a response error or response CRC error. If a response is not received from the card because a response timeout occurs, the BIU does not receive a signal that the data transfer is complete; this happens if the command sent by the SD/MMC controller is an illegal operation for the card, which keeps the card from starting a read data transfer.

If data is not received before the data timeout, the data path signals a data timeout to the BIU and an end to the data transfer done. Based on the value of the transfer\_mode bit in the Command register, the data-receive state machine gets data from the card data bus in a stream or block(s).

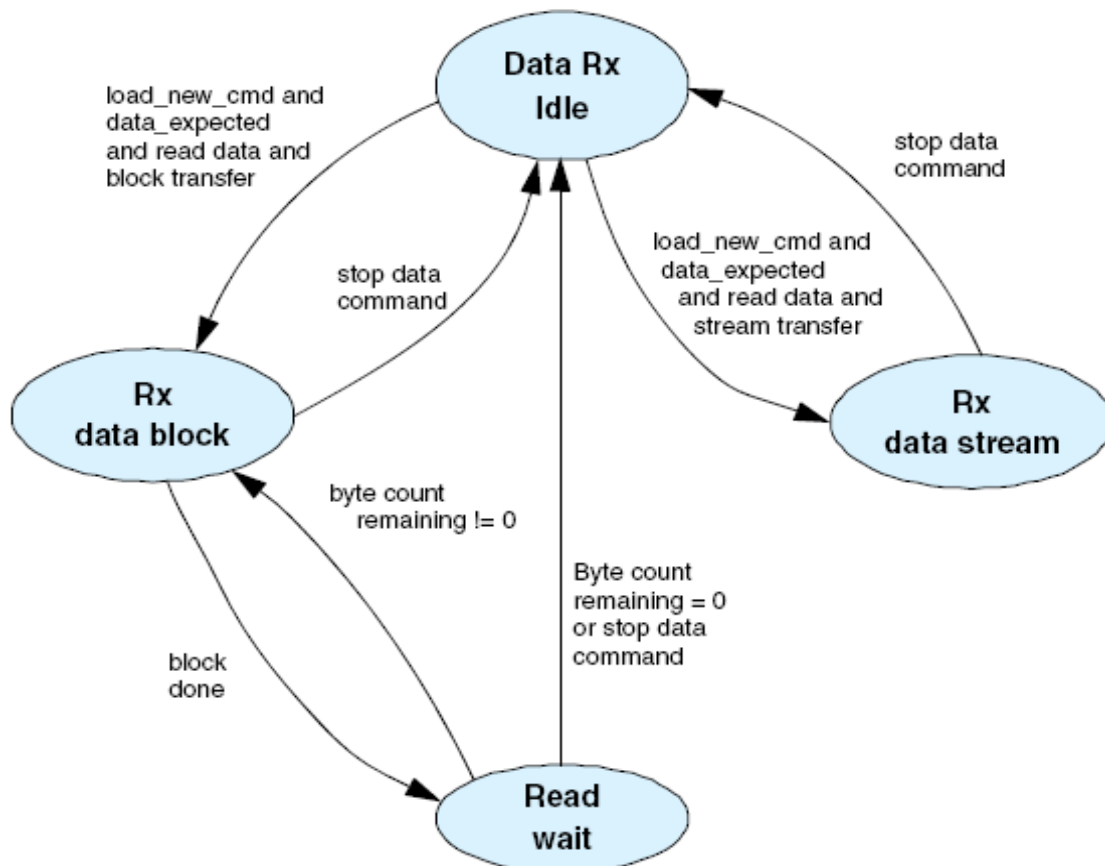


Fig. 7-5 SD/MMC Data Receive State Machine

### 6. Data Receive

A stream-read data transfer occurs if the transfer\_mode bit in the Command register equals 1, at which time the data path receives data from the card and pushes it to the FIFO. If the FIFO becomes full, the card clock stops and restarts once the FIFO is no longer full.

An open-ended stream-read data transfer occurs if the byte\_count register equals 0. During this type of data transfer, the data path continuously receives data in a stream until the host software issues a stop command. A stream data transfer terminates two clock cycles after the end bit of the stop command.

If the byte\_count register contains a non-zero value and the send\_auto\_stop bit is set in the Command register, a stop command is internally generated and loaded into the command path, where the end bit of the stop command occurs after the last byte of the stream data transfer is received. This data transfer can terminate if the host issues a stop or abort command before all the data bytes are received from the card.

### 7. Single-Block Data Receive

A single-block read-data transfer occurs if the transfer\_mode bit in the Command register is set to 0 and the value of the byte\_count register is equal to the value of the block\_size register. When a start bit is received before the data times out, data bytes



equal to the block size and CRC16 are received and checked with the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card\_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively. If there is a CRC16 mismatch, the data path signals a data CRC error to the BIU. If the received end bit is not 1, the BIU receives an end-bit error.

#### 8. Multiple-Block Data Receive

If the transfer\_mode bit in the Command register is set to 0 and the value of the byte\_count register is not equal to the value of the block\_size register, it is a multiple-block read-data transfer. The data-receive state machine receives data in blocks, where the number of bytes in a block is equal to the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card\_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively. After a data block is received, if the remaining byte\_count becomes 0, the data path signals a data transfer to the BIU.

If the remaining data bytes are greater than 0, the data path state machine causes another data block to be received. If CRC16 of a received data block does not match the internally-generated CRC16, a data CRC error to the BIU and data reception continue further data transmission until all bytes are transmitted. Additionally, if the end of a received data block is not 1, data on the data path signals terminate the bit error to the CIU and the data-receive state machine terminates data reception, waits for data timeout, and signals to the BIU that the data transfer is complete. If the send\_auto\_stop bit is set in the Command register, the stop command is internally generated when the last data block is transferred, where no extra bytes are transferred from the card; the end bit of the stop command may not exactly match the end bit of the last data block.

If the requested block size for data transfers to cards is less than 4, 16, or 32 bytes for 1-bit, 4-bit, or 8-bit data transfer modes, respectively, the data-transmit state machine terminates the data transfer when all data is transferred, at which point the internally-generated stop command is loaded in the command path. Data received from the card after that are then ignored by the data path.

If the byte\_count is 0—the block size must be greater than 0—it is an open-ended block transfer. For this type of data transfer, the data-receive state machine continues the block-read data transfer until the host software issues a stop or abort command.

#### SDIO Interrupt Control

Interrupts for SD cards are reported to the BIU by asserting an interrupt signal for two clock cycles. SDIO cards signal an interrupt by asserting cdata\_in low during the interrupt period; an interrupt period for the selected card is determined by the interrupt control state machine. An interrupt period is always valid for non-active or non-selected cards, and 1-bit data mode for the selected card. An interrupt period for a wide-bus active or selected card is valid for the following conditions:

- Card is idle
- Non-data transfer command in progress
- Third clock after end bit of data block between two data blocks
- From two clocks after end bit of last data until end bit of next data transfer command

Bear in mind that, in the following situations, the SD/MMC controller does not sample the SDIO interrupt of the selected card when the card data width is 4 bits. Since the SDIO interrupt is level-triggered, it is sampled in a further interrupt period and the host does not lose any SDIO interrupt from the card.

1. Read/Write Resume – The CIU treats the resume command as a normal data transfer command. SDIO interrupts during the resume command are handled similarly to other data commands. According to the SDIO specification, for the normal data command the interrupt period ends after the command end bit of the

data command; for the resume command, it ends after the response end bit. In the case of the resume command, the SD/MMC controller stops the interrupt sampling period after the resume command end bit, instead of stopping after the response end bit of the resume command.

2. Suspend during read transfer – If the read data transfer is suspended by the host, the host sets the `abort_read_data` bit in the SD/MMC controller to reset the data state machine. In the CIU, the SDIO interrupts are handled such that the interrupt sampling starts after the `abort_read_data` bit is set by the host. In this case the SD/MMC controller does not sample SDIO interrupts between the period from response of the suspend command to setting the `abort_read_data` bit, and starts sampling after setting the `abort_read_data` bit.

### Clock Control

The clock control block provides different clock frequencies required for SD/MMC cards. The `cclk_in` is the source clock (`cclk_in`  $\geq$  card max operating frequency) for clock dividers of the clock control block. This source clock (`cclk_in`) is used to generate different card clock frequencies. Each card clock can have different clock frequencies, since the SD card can be a low-speed SD card or a full-speed SD card. The SD/MMC controller provides one clock signal (`cclk_out`) per card, which allows each card to operate at different clock frequencies.

The clock frequency of a card depends on the following clock control registers:

- Clock Divider register – Internal clock dividers are used to generate different clock frequencies required for cards; a configuration parameter determines the number of clock dividers (1-4). The division factor for each clock divider can be programmed by writing to the Clock Divider register. The clock divider is an 8-bit value that provides a clock division factor from 1 to 510; a value of 0 represents a clock-divider bypass, a value of 1 represents a divide by 2, a value of 2 represents a divide by 4, and so on.
- Clock Source register – One of the divided clocks from four clock dividers is selected for a card `cclk_out` by programming the Clock Source register.
- Clock Control register – `cclk_out` can be enabled or disabled for each card under the following conditions:
  1. `clk_enable` – `cclk_out` for a card is enabled if the `clk_enable` bit for a card in the Clock Control register is programmed (set to 1) or disabled (set to 0).
  2. Low-power mode – Low-power mode of a card can be enabled by setting the low-power mode bit of the Clock Control register to 1. If low-power mode is enabled to save card power, the `cclk_out` signal is disabled when the card is idle for at least 8 card clock cycles. It is enabled when a new command is loaded and the command path goes to a non-idle state.

Additionally, `cclk_out` of a selected or active card is disabled when an internal FIFO is full – card read (no more data can be received from card) – or when the FIFO is empty – card write (no data is available for transmission). This helps to avoid FIFO overrun and underrun conditions.

Under the following conditions, the card clock is stopped or disabled, along with the active `clk_en`, for the selected card:

- Clock can be disabled by writing to Clock Enable register (`clk_en` bit = 1).
- If low-power mode is selected and card is idle, or not selected for 8 clocks.
- FIFO is full and data path cannot accept more data from the card and data transfer is incomplete – to avoid FIFO overrun.
- FIFO is empty and data path cannot transmit more data to the card and data transfer is incomplete – to avoid FIFO underrun.

**Note:** Care should be taken by the host firmware while changing the Clock Divider register and Clock Source register values. The card clock must be disabled through the Clock Control register before changing the values of the Clock Divider and Clock Source registers.

### SD/MMC Mux/Demux Unit

A separate bus runs between the SD/MMC controller and each card. The demux logic



sends the command or data to only the selected card when commands or data are sent from the controller. The unselected cards see those on their command or data path; a card is selected by the card\_number value set in the Command register. Similarly, the response or data input from the selected card is multiplexed and sent to the SD/MMC controller.

## 7.3 Registers

This section describes the registers of the design.

### 7.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SDMMC_CTRL	0x00	W	0x0	SDMMC Control register
SDMMC_PWREN	0x04	W	0x0	Power-enable register
SDMMC_CLKDIV	0x08	W	0x0	Clock-divider register
SDMMC_CLKSRC	0x0C	W	0x0	Clock-source register
SDMMC_CLKENA	0x10	W	0x0	Clock-enable register
SDMMC_TMOUT	0x14	W	0xFFFFFFFF40	Time-out register(number of card clock output clocks)
SDMMC_CTYPE	0x18	W	0x0	Card-type register
SDMMC_BLKSI	0x1C	W	0x200	Block-size register
SDMMC_BYTCNT	0x20	W	0x200	Byte-count register
SDMMC_INTMASK	0x24	W	0x0	Interrupt-mask register
SDMMC_CMDARG	0x28	W	0x0	Command-argument register
SDMMC_CMD	0x2C	W	0x0	Command-register
SDMMC_RESP0	0x30	W	0x0	Response-0 register
SDMMC_RESP1	0x34	W	0x0	Response-1 register
SDMMC_RESP2	0x38	W	0x0	Response-2 register
SDMMC_RESP3	0x3C	W	0x0	Response-3 register
SDMMC_MINTSTS	0x40	W	0x0	Masked interrupt-status register
SDMMC_RINTSTS	0x44	W	0x0	Raw interrupt-status register
SDMMC_STATUS	0x48	W	0x6	Status register; mainly for debug purposes
SDMMC_FIFOTH	0x4C	W	0x001f0000	FIFO threshold register
SDMMC_CDETECT	0x50	W		Card-detect register
SDMMC_WRTPRT	0x54	W		Write-protect register
SDMMC_TBCNT	0x5C	W	0x0	Transferred CIU card byte count
SDMMC_TBBCNT	0x60	W	0x0	Transferred host/DMA to/from BIU_FIFO byte count
SDMMC_DEBNCE	0x64	W	0x00ffffff	Card detect debounce register (number of host clocks)

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 7.3.2 Detail Register Description

#### SDMMC\_CTRL

Address: Operational Base + offset( 0x00)

SDMMC Control Register

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x1	enable_OD_pullup: External open-drain pullup: 0- Disable 1- Enable

			When bit is set, command output always driven in open-drive mode.
23:20	RW	0x0	Card_voltage_b: Card regulator-B voltage setting; output to card_volt_b port.
19:16	RW	0x0	Card_voltage_a: Card regulator-A voltage setting; output to card_volt_a port.
15:9	-	-	Reserved
8	RW	0x0	abort_read_data: 0- No change 1- After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data state-machine resets to idle. Used in SDIO card suspend sequence.
7	RW	0x0	send_irq_response: 0- No change 1- Send auto IRQ response Bit automatically clears once response is sent. To wait for MMC card interrupts, host issues CMD40, and SDMMC Controller waits for interrupt response from MMC card. In meantime, if host wants controller to exit waiting for interrupt state, it can set this bit, at which time controller command state-machine sends CMD40 response on bus and returns to idle state.
6	RW	0x0	read_wait: 0- Clear read wait 1- Assert read wait
5	RW	0x0	dma_enable: 0- Disable DMA transfer mode 1- Enable DMA transfer mode
4	RW	0x0	int_enable: Global interrupt enable/disable bit: 0- Disable interrupts 1- Enable interrupts The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.
3:2	-	-	Reserved
1	RW	0x0	fifo_reset: 0- No change 1- Reset data FIFO to reset FIFO pointers To reset FIFO, firmware should set bit to 1. This bit is auto-cleared after completion of reset operation.
0	RW	0x0	controller_reset: 0- No change 1- Reset SDMMC controller To reset controller, firmware should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets: <ul style="list-style-type: none"> <li>● BIU/CIU interface</li> <li>● CIU and state machines</li> <li>● abort_read_data, send_irq_response, and read_wait bits of control register.</li> </ul>

			<ul style="list-style-type: none"> <li>Start_cmd bit of Command register</li> </ul> <p>Dose not affect any registers or DMA interface, or FIFO or host interrupts.</p>
--	--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**SDMMC\_PWREN**

Address: Operational Base + offset( 0x04)

Power Enable Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	Power_enable Once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card. 0- power off 1- power on

**SDMMC\_CLKDIV**

Address: Operational Base + offset( 0x08)

Clock Divider Register

bit	Attr	Reset Value	Description
31:24	RW	0x0	clk_divider3: Clock divider-3 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0=1$ (no division, by pass), a value of 1 means divide by $2^1=2$ , a value of "ff" means divide by $2^{255}=510$ , and so on.
23:16	RW	0x0	clk_divider2: Clock divider-2 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0=1$ (no division, by pass), a value of 1 means divide by $2^1=2$ , a value of "ff" means divide by $2^{255}=510$ , and so on.
15:8	RW	0x0	clk_divider1: Clock divider-1 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0=1$ (no division, by pass), a value of 1 means divide by $2^1=2$ , a value of "ff" means divide by $2^{255}=510$ , and so on.
7:0	RW	0x0	clk_divider0: Clock divider-0 value. Clock division is $2^n$ . For example, value of 0 means divide by $2^0=1$ (no division, by pass), a value of 1 means divide by $2^1=2$ , a value of "ff" means divide by $2^{255}=510$ , and so on.

**SDMMC\_CLKSRC**

Address: Operational Base + offset(0x0C)

SDMMC Clock source Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	clk_source: Clock divider source for SD/MMC card. Each card has two bits assigned to it. 00- Clock divider 0 01- Clock divider 1 10- Clock divider 2 11- Clock divider 3

**SDMMC\_CLKENA**

Address: Operational Base + offset(0x10)

Clock Enable Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	cclk_low_power: Low-power control for SD/MMC card clock. 0- Non-low-power mode 1- Low-power mode; stop clock when card in IDLE(should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).
15:1	-	-	Reserved.
0	RW	0x0	cclk_enable: Clock_enable control for SD/MMC card clock. 0- Clock disabled 1- Clock enabled

**SDMMC\_TMOUT**

Address: Operational Base + offset(0x14)

SDMMC Timeout Register

Bit	Attr	Reset Value	Description
31:8	W	0xffffffff	data_timeout: Value for card Data Read Timeout; same value also used for Data Starvation by Host timeout. Value is in number of card output clocks-cclk out of selected card.
7:0	W	0x40	response_timeout: Response timeout value. Value is in number of card output clocks-cclk_out.

**SDMMC\_CTYPE**

Address: Operational Base + offset(0x18)

Card Type Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	card_width: One bit indicates if card is 8-bit: 0- Non 8-bit mode 1- 8-bit mode
15:1	-	-	Reserved.
0	RW	0x0	card_width: One bit indicates if card is 1-bit or 4-bit: 0- 1-bit mode 1- 4-bit mode

The following examples use values for CTYPE[16]:

- If CTYPE[16]=1, the card at port0 is in 8-bit mode. Note that the CTYPE[0] value is ignored; it is recommended to keep this set to 0.
- If CTYPE[16]=0, the card at port0 is in either 1-bit or 4-bit mode, depending upon the value of CTYPE[0]; that is, if CTYPE[0]=1 -4-bit, CTYPE[0]=0 -1-bit.

**SDMMC\_BLKSI Z**

Address: Operational Base + offset(0x1C)

Block Size Register

bit	Attr	Reset Value	Description
31:16	-	-	Reserved
15:0	RW	0x200	Block size

**SDMMC\_BYTCNT**

Address: Operational Base + offset(0x20)

Byte Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x200	byte_count: Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.

**SDMMC\_INTMASK**

Address: Operational Base + offset(0x24)

Interrupt Mask Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	sdio_int_mask: Mask SDIO interrupts When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enable an interrupt.
15:0	RW	0x0	int_mask: Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt. bit15 - End-bit error(read)/Write no CRC(EBE) bit14 - Auto command done(ACD) bit13 - Start-bit error(SBE) bit12 - Hardware locked write error(HLE) bit11 - FIFO underrun/overflow error(FRUN) bit10 - Data starvation-by-host timeout(HTO) bit9 - Data read timeout(DRTO) bit8 - Response timeout(RTO) bit7 - Data CRC error(DCRC) bit6 - Response CRC error(RCRC) bit5 - Receive FIFO data request(RXDR) bit4 - Transmit FIFO data request(TXDR) bit3 - Data transfer over(DTO) bit2 - Command done(CD) bit1 - Response error(RE) bit0 - Card detect(CD)

**SDMMC\_CMDARG**

Address: Operational Base + offset(0x28)

Command Argument Register

bit	Attr	Reset Value	Description
31:0	RW	0x0	cmd_arg: Value indicates command argument to be passed to card.

**SDMMC\_CMD**

Address: Operational Base + offset(0x2C)

Command Register

bit	Attr	Reset Value	Description
31	RW	0x0	start_cmd: Start command. Once command is taken by CIU, bit is cleared. When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register.
30:22	-	-	Reserved
21	RW	0x0	update_clock_registers_only: 0- Normal command sequence 1- Do not send commands, just update clock register value into card clock domain Following register values transferred into card clock domain: CLKDIV, CLKSRC, CLKENA. Changes card clocks(change frequency, trun off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when update_clock_registers_only=0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card. When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC cards.
20:16	-	-	Reserved.
15	RW	0x0	send_initialization: 0- Do not send initialization sequence(80 clocks of 1) before sending this command 1- Send initialization sequence before sending this command. After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card to that controller will initialize clocks before sending command to card.
14	RW	0x0	stop_abort_cmd: 0- Neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0. 1- Stop or abort command intended to stop current data transfer in progress. When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state.
13	RW	0x0	wait_prvdata_complete: 0- Send command at once, even if previous data transfer has not completed. 1- Wait for previous data transfer completion before sending command. The wait_prvdata_complete=0 option typically used to query status of card during data transfer or to stop current data transfer; card_number should be same as



			in previous command.
12	RW	0x0	send_auto_stop: 0- No stop command sent at end of data transfer 1- Send stop command at end of data transfer When set, SDMMC controller sends stop command to SDMMC cards at end of data transfer. Refer to Table9 on page101 to determine: <ul style="list-style-type: none"> <li>when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands</li> <li>open-ended transfers that software should explicitly send to stop command</li> </ul> Don't care if no data expected from card.
11	RW	0x0	transfer_mode: 0- Block data transfer command 1- Stream data transfer command Don't care if no data expected.
10	RW	0x0	read/write: 0- Read from card 1- Write to card Don't care if no data expected from card.
9	RW	0x0	data_expected: 0- No data transfer expected(read/write) 1- Data transfer expected(read/write)
8	RW	0x0	check_response_crc: 0- Do not check response CRC 1- Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.
7	RW	0x0	response_length: 0- Short response expected from card 1- Long response expected from card
6	RW	0x0	response_expect: 0- No response expected from card 1- Response expected from card
5:0	RW	0x0	cmd_index: Command index

**SDMMC\_RESP0**

Address: Operational Base + offset(0x30)

Response Register 0

bit	Attr	Reset Value	Description
31:0	R	0x0	response 0: Bit[31:0] of response

**SDMMC\_RESP1**

Address: Operational Base + offset(0x34)

Response Register 1

bit	Attr	Reset Value	Description
31:0	R	0x0	response 1: Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always "short" for



			them. For information on when CIU sends auto-stop commands, refer to "Auto-Stop" on page 100.
--	--	--	-----------------------------------------------------------------------------------------------

**SDMMC\_RESP2**

Address: Operational Base + offset(0x38)

Response Register 2

bit	Attr	Reset Value	Description
31:0	R	0x0	response 2: Bit[95:64] of long response

**SDMMC\_RESP3**

Address: Operational Base + offset(0x3C)

Response Register 3

bit	Attr	Reset Value	Description
31:0	R	0x0	response 3: Bit[127:96] of long response

**SDMMC\_MINTSTS**

Address: Operational Base + offset(0x40)

Masked Interrupt Status Register

MINTSTS = RINTSTS and INTMASK

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	R	0x0	sdio_interrupt: Interrupt from SDIO card. SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register(mask bit 1 enable interrupt;0 masks interrupt). 0- No SDIO interrupt from card 1- SDIO interrupt from card
15:0	R	0x0	int_status: Interrupt enabled only if corresponding bit in interrupt mask register is set. bit15- End-bit error read/write no CRC (EBE) bit14- Auto command done(ACD) bit13- Start-bit error(SBE) bit12- Hardware locked write error(HLE) bit11- FIFO underrun/overflow error(FRUN) bit10- Data starvation by host timeout(HTO) bit9 - Data read timeout(DRTO) bit8 - Response timeout(RTO) bit7 - Data CRC error(DCRC) bit6 - Response CRC error(RCRC) bit5 - Receive FIFO data request(RXDR) bit4 - Transmit FIFO data request(TXDR) bit3 - Data transfer over(DTO) bit2 - Command done(CD) bit1 - Response error(RE) bit0 - Card detect(CD)

**SDMMC\_RINTSTS**

Address: Operational Base + offset(0x44)

Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	sdio_interrupt: Interrupt from SDIO card. Writes to this bit clear it.

			Value of 1 clears bit and 0 leaves bit intact. 0- No SDIO interrupt from card 1- SDIO interrupt from card Bit is logged regardless of interrupt-mask status.
15:0	RW	0x0	int_status: Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. bit15- End-bit error read/write no CRC (EBE) bit14- Auto command done(ACD) bit13- Start-bit error(SBE) bit12- Hardware locked write error(HLE) bit11- FIFO underrun/overflow error(FRUN) bit10- Data starvation by host timeout(HTO) bit9 - Data read timeout(DRTO) bit8 - Response timeout(RTO) bit7 - Data CRC error(DCRC) bit6 - Response CRC error(RCRC) bit5 - Receive FIFO data request(RXDR) bit4 - Transmit FIFO data request(TXDR) bit3 - Data transfer over(DTO) bit2 - Command done(CD) bit1 - Response error(RE) bit0 - Card detect(CD)

**SDMMC\_STATUS**

Address: Operational Base + offset(0x48)

Status Register

bit	Attr	Reset Value	Description
31	R	0x0	dma_req: DMA request signal state
30	R	0x0	dma_ack: DMA acknowledge signal state
29:17	R	0x0	fifo_count: FIFO count - Number of filled locations in FIFO
16:11	R	0x0	response_index: Index of previous response, including any auto-stop sent by core
10	R	0x0	data_state_mc_busy: Data transmit or receive state-machine is busy
9	R	0x0	data_busy: Inverted version of raw selected card_data[0] 0- card data not busy 1- card data busy
8	R	0x0	data_3_status: Raw selected card_data[3]; checks whether card is present 0- card not present 1- card present
7:4	R	0x0	command fsm states: 0- Idle 1- Send init sequence 2- Tx cmd start bit 3- Tx cmd tx bit 4- Tx cmd index + arg 5- Tx cmd crc7 6- Tx cmd end bit

			7- Rx resp start bit 8- Rx resp IRQ response 9- Rx resp tx bit 10- Rx resp cmd idx 11- Rx resp data 12- Rx resp crc7 13- Rx resp end bit 14- Cmd path wait NCC 15- Wait; CMD-to response turnaround
3	R	0x0	fifo_full: FIFO is full status
2	R	0x1	fifo_empty: FIFO is empty status
1	R	0x1	fifo_tx_watermark: FIFO reached Transmit watermark level; not qualified with data transfer.
0	R	0x0	fifo_rx_watermark: FIFO reached Receive watermark level; not qualified with data transfer.

**SDMMC\_FIFOTH**

Address: Operational Base + offset(0x4C)

FIFO Threshold Watermark Register

bit	Attr	Reset Value	Description
31	-	-	Reserved
30:28	RW	0x0	DW_DMA_Multiple_Transaction_Size: Burst size of multiple transaction; should be programmed same as DW_DMA controller multiple-transaction-size SRC/DEST_MSIZ. 000-1 transfers 001-4 010-8 011-16 100-32 101-64 110-128 111-256 Value should be sub-multiple of (RX_WMark+1) and (32-TX_WMark) Recommended: MSize=16, TX_WMask=16, RX_WMask=15
27:16	RW	0x1f	RX_WMark: FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold(RXDR) interrupt is endable, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single

			transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. Limitation: RX_WMark ≤ 30 Recommended: 15; (means greater than 15)
15:12	-	-	Reserved
11:0	RW	0x0	TX_WMark: FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated. Regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enable, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. Limitation: TX_WMark ≥ 1; Recommended: 16 (means less than or equal to 16)

**SDMMC\_CDETECT**

Address: Operational Base + offset(0x50)

Card Detect Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x1	card_detect_n: Value on card_detect input port 0 represents presence of card.

**SDMMC\_WRTprt**

Address: Operational Base + offset(0x54)

Write Protect Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	write_protect: Value on card_write_prt input port 1 represents write protection.

**SDMMC\_TCBCNT**

Address: Operational Base + offset(0x5C)

Transferred CIU Card Byte Count Register

bit	Attr	Reset Value	Description
31:0	R	0x0	trans_card_byte_count Number of bytes transferred by CIU unit to card.

**SDMMC\_TBBCNT**

Address: Operational Base + offset(0x60)

Transferred Host to BIU-FIFO Byte Count Register

bit	Attr	Reset Value	Description
31:0	R	0x0	trans_fifo_byte_count: Number of bytes transferred between Host/DMA

		memory and BIU FIFO.
--	--	----------------------

**SDMMC\_DEBNCE**

Address: Operational Base + offset(0x64)

Debounce Count Register

bit	Attr	Reset Value	Description
31:24	-	-	Reserved
23:0	RW	24'hff_ffff	Number of host clocks used by debounce filter logic; typical debounce time is 5-25ms.

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 7.4 Functional Description

### 7.4.1 Operation

#### Software/Hardware Restrictions

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

To avoid glitches in the card clock outputs (cclk\_out), the software should use the following steps when changing the card clock frequency:

1. Before disable the clocks, ensure that the card is not busy due to any previous data command. To determine this, check for 0 in bit9 of STATUS register.
2. Update the Clock Enable register to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:
  - start\_cmd bit
  - "update clock registers only" bits
  - "wait\_previous data complete" bit
 Wait for the CIU to take the command by polling for 0 on the start\_cmd bit.
3. Set the start\_cmd bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.
4. Set start\_cmd to update the Clock Enable register in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the Data Transfer Over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX\_WMark interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. While using the external DMA interface for reading from a card, the DTO interrupt occurs only after all the data is flushed to memory by the DMA interface unit.

While writing to a card in external DMA mode, if an undefined-length transfer is selected by setting the Byte Count Register to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a controller\_reset command by setting control register bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the host. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly
- DMA data read is ignored
- Write data is unknown(x)

Additionally, if dma\_reset is also issued, any pending DMA transfer is abruptly

terminated. When the DW-DMA is used, the DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.

One data-transfer requirement between the FIFO and host is that the number of transfers should be a multiple of the FIFO data width (32bits). For example, you want to write only 15 bytes to an SDMMC card (BYTCNT), the host should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers. The software can still program the Byte Count register to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the host should still read all 16 bytes from the FIFO.

It is recommended that you not change the FIFO threshold register in the middle of data transfers.

## 7.4.2 Programming sequence

### Initialization

Fig. 6-6 illustrates the initialization flow.

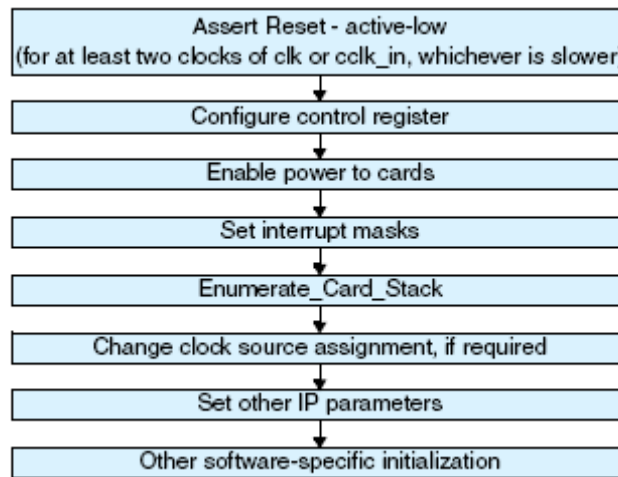


Fig. 7-6 SD/MMC Initialization Sequence

Once the power and clocks are stable, reset\_n should be asserted(active-low) for at least two clocks of clk or cclk\_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

1. Configure control register – For MMC mode, enable the open-drain pullup by setting enable\_OD\_pullup(bit24) in the control register.
2. Enable power to cards – Before enabling the power, confirm that the voltage setting to the voltage regulators is correct. Enable power to the connected cards by setting the corresponding bit to 1 in the Power Enable register. Wait for the power ramp-up time.
3. Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register. Set the global int\_enable bit of the Control register. It is recommended that you write 0xffff\_ffff to the Raw Interrupt register in order to clear any pending interrupts before setting the int\_enable bit.
4. Enumerate card stack – Each card is enumerated according to card type; for details, refer to “Enumerated Card Stack”. For enumeration, you should restrict the clock frequency to 400KHz.
5. Changing clock source assignment – set the card frequency using the clock-divider and clock-source registers; for details, refer to “Clock Programming”. MMC cards operate at a maximum of 20MHz (at maximum of 52MHz in high-speed mode). SD mode operates at a maximum of 25MHz (at maximum of 50MHz in high-speed mode).
6. Set other parameters, which normally do not need to be changed with every command, with a typical value such as timeout values in cclk\_out according to SDMMC



specifications.

- ResponseTimeout = 0x64
  - DataTimeout = highest of one of the following:  
(10\*((TAAC\*Fop)+(100\*NSAC))  
Host FIFO read/write latency from FIFO empty/full
  - Set the debounce value to 25ms(default:0x0fffff) in host clock cycle units in the DEBNCE register.
  - FIFO threshold value in bytes in the FIFOTH register. Typically, the threshold value can be set to half the FIFO depth; that is:  
RX\_WMark=15;  
TX\_WMark=16
7. According to MMC standards, the open-drain pullup resistor is required during only the enumeration phase. Therefore for MMC mode, disable the open-drain pullup by clearing the enable\_OD\_pullup (bit24) in the Control register.

### **Enumerated Card Stack**

The card stack does the following:

- Enumerates all connected cards
- Sets the RCA for the connected cards
- Reads card-specific information
- Stores card-specific information locally

Enumeration depends on the operating mode of the SDMMC Host Controller; the card type is first identified and the appropriate card enumeration routine is called.

1. Check if the card is connected.
2. Clear the card type register to set the card width as a single bit. For the given card number, clear the corresponding bits in the card\_type register. Clear the register bit for a 1-bit, 4-bit, or 8-bit bus width. For example, for card number=1, clear bit 0 and bit 16 of the card\_type register.
3. Set clock frequency to Fod=400KHz, maximum – Program clock divider0 (bits 0-7 in the CLKDIV register) value to one-half of the cclk\_in frequency divided by 400KHz. For example, if cclk\_in is 20MHz, then the value is 20,000/(2\*400)=25.
4. Identify the card type; that is, SD, MMC, or SDIO.
  - a. Send CMD5 first. If a response is received, then the card is SDIO
  - b. If not, send CMD8 with the following Argument  
 Bit[31:12] = 20'h0 //reserved bits  
 Bit[11:8] = 4'b0001 //VHS value  
 Bit[7:0] = 8'b10101010 //Preferred Check Pattern by SD2.0
  - c. If Response is received the card supports High Capacity SD2.0 then send ACMD41 with the following Argument  
 Bit[31] = 1'b0; //Reserved bits  
 Bit[30] = 1'b1; //High Capacity Status  
 Bit[29:24] = 6'h0; //Reserved bits  
 Bit[23:0] = Supported Voltage Range
  - d. If Response is received for ACMD41 then the card is SD. Otherwise the card is MMC.
  - e. If response is not received for initial CMD8 then card does not support High Capacity SD2.0, then issue CMD0 followed by ACMD41 with the following Argument  
 Bit[31] = 1'b0; //Reserved bits  
 Bit[30] = 1'b0; //High Capacity Status  
 Bit[29:24] = 6'h0; //Reserved bits  
 Bit[23:0] = Supported Voltage Range
5. Enumerate the card according to the card type.
6. Use a clock source with a frequency = Fod (that is, 400KHz) and use the following enumeration command sequence:
  - SD card – Send CMD0, CMD8, ACMD41, CMD2, CMD3.
  - SDIO – Send CMD5, CMD3.
  - MMC – Send CMD0, CMD1, CMD2, CMD3.



### **Power Control**

You can implement power control using the following registers, along with external circuitry:

- Control register bits `card_voltage_a` and `card_voltage_b` – Status of these bits is reflected at the IO pins. The bits can be used to generate or control the supply voltage that the memory cards require.
- Power enable register – Control power to individual cards.

Programming these two register depends on the implemented external circuitry. While turning on or off the power enable, you should confirm that power supply settings are correct. Power to all cards usually should be disable while switching off the power.

### **Clock Programming**

The SDMMC controller supports four clock sources, each of which can be programmed with a different frequency; software can select the clock source for each card. The clock to an individual card can be enabled or disabled. Registers that support this are:

- CLKDIV – Programs individual clock source frequency.
- CLKSRC – Assign clock source for each card.
- CLKENA – Enables or disables clock for individual card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The SDMMC Controller loads each of these registers only when the `start_cmd` bit and the `Update_clk_regs_only` bit in the CMD register are set. When a command is successfully loaded, the SDMMC Controller clears this bit, unless the SDMMC Controller already has another command in the queue, at which point it gives an HLE(Hardware Locked Error).

Software should look for the `start_cmd` and the `Update_clk_regs_only` bits, and should also set the `wait_prvdata_complete` bit to ensure that clock parameters do not change during data transfer. Note that even though `start_cmd` is set for updating clock registers, the SDMMC Controller does not raise a `command_done` signal upon command completion.

The following shows how to program these registers:

1. Confirm that no card is engaged in any transaction; if there is a transaction, wait until it finishes.
2. Stop all clocks by writing `xxxx0000` to the CLKENA register. Set the `start_cmd`, `Update_clk_regs_only`, and `wait_prvdata_complete` bits in the CMD register. Wait until `start_cmd` is cleared or an HLE is set; in case of an HLE, repeat the command.
3. Program the CLKDIV and CLKSRC registers, as required. Set the `start_cmd`, `Update_clk_regs_only`, and `wait_prvdata_complete` bits in the CMD register. Wait until `start_cmd` is cleared or an HLE is set; in case of an HLE, repeat the command.
4. Re-enable all clocks by programming the CLKENA register. Set the `start_cmd`, `Update_clk_regs_only`, and `wait_prvdata_complete` bits in the CMD register. Wait until `start_cmd` is cleared or an HLE is set; in case of an HLE, repeat the command.

### **No-Data Command With or Without Response Sequence**

To send any non-data command, the software needs to program the CMD register @0x2C and the CMDARG register @0x28 with appropriate parameters. Using these two registers, the SD/MMC controller forms the command and sends it to the command bus. The SD/MMC controller reflects the errors in the command response through the error bits of the RINTSTS register.

When a response is received – either erroneous or valid – the SD/MMC controller sets the `command_done` bit in the RINTSTS register. A short response is copied in Response Register0, while a long response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3 register bit 31 represents the MSB, and the Response0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

1. Program the Command register @0x28 with the appropriate command argument

parameter.

2. Program the Command register @0x2C with the settings in Table 6-2.

Table 7-2 SD/MMC Command Register Settings for No-Data Command

Parameter	Value	Description
<b>Default</b>		
start_cmd	1	-
Update_clk_regs_only	0	No clock parameters update command
data_expected	0	No data command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
cmd_index	command-index	-
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
<b>User-selectable</b>		
wait_prvdata_complete	1	Before sending command on command line, host should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
check_response_crc	1	If host should crosscheck CRC of response received

3. Wait for command acceptance by host. The following happens when the command is loaded into the SD/MMC controller:

- SD/MMC controller accepts the command for execution and clears the start\_cmd bit in the CMD register, unless one command is in process, at which point the SD/MMC controller can load and keep the second command in the buffer.
- If the SD/MMC controller is unable to load the command – that is, a command is already in progress, a second command is in the buffer, and a third command is attempted – then it generates an HLE (hardware-locked error).

4. Check if there is an HLE.

5. Wait for command execution to complete. After receiving either a response from a card or response timeout, the SD/MMC controller sets the command\_done bit in the RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.

6. Check if response\_timeout error, response\_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the RINTSTS register @0x44. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers @0x30-0x3C.

Software should not modify clock parameters while a command is being executed.

### Data Transfer Commands

Data transfer commands transfer data between the memory card and the SD/MMC controller. To send a data command, the SD/MMC controller needs a command argument, total data size, and block size. Software can receive or send data through the FIFO. Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively.

For the data transfer commands, it is important that the same bus width that is

programmed in the card should be set in the card type register @0x18.

The SD/MMC controller generates an interrupt for different conditions during data transfer, which are reflected in the RINTSTS register @0x44 as:

1. Data\_Transfer\_Over (bit 3) – When data transfer is over or terminated. If there is a response timeout error, then the SD/MMC Controller does not attempt any data transfer and the “Data Transfer Over” bit is never set.
2. Transmit\_FIFO\_Data\_request (bit 4) – FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO.
3. Receive\_FIFO\_Data\_request (bit 5) – FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
4. Data starvation by Host timeout (bit 10) – FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the SD/MMC controller cannot continue with data transfer. The clock to the card has been stopped.
5. Data read timeout error (bit 9) – Card has not sent data within the timeout period.
6. Data CRC error (bit 7) – CRC error occurred during data reception.
7. Start bit error (bit 13) – Start bit was not received during data reception.
8. End bit error (bit 15) – End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response timeout, then no data transfer occurred.

#### Single-Block or Multiple-Block Read

Steps involved in a single-block or multiple-block read are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C. The SD/MMC controller expects data from the card in blocks of size BLKSIZ each.
3. Program the CMDARG register @0x28 with the data address of the beginning of a data read.

Program the Command register with the parameters listed in Table 6-3. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

Table 7-3 SD/MMC Command Register Setting for Single-Block or Multiple-Block Read

Parameter	Value	Description
<b>Default</b>		
start_cmd	1	-
Update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0 or 1	Set according to Table xx
transfer_mode	0	Block transfer
read_write	0	Read from card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
<b>User-selectable</b>		
cmd_index	command -index	-
wait_prvdata_complete	1	0- Sends command immediately 1- Sends command after previous data transfer ends

check_response_crc	1	0- SD/MMC controller should not check response CRC 1- SD/MMC controller should check response CRC
--------------------	---	------------------------------------------------------------------------------------------------------

After writing to the CMD register, the SD/MMC controller starts executing the command; when the command is sent to the bus, the command\_done interrupt is generated.

4. Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
5. Software should look for Receive\_FIFO\_Data\_request and/or data starvation by host timeout conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
6. When a Data\_Transfer\_Over interrupt is received, the software should read the remaining data from the FIFO.

### Single-Block or Multiple-Block Write

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C; the SD/MMC controller sends data in blocks of size BLKSIZ each.
3. Program CMDARG register @0x28 with the data address to which data should be written.
4. Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.
5. Program the Command register with the parameters listed in [Table 6-4](#). For SD and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

Table 7-4 SD/MMC Command Register Settings for Single-Block or Multiple-Block Write

Parameter	Value	Description
<b>Default</b>		
start_cmd	1	-
Update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0 or 1	Set according to Table xx
transfer_mode	0	Block transfer
read_write	1	Write to card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
<b>User-selectable</b>		
cmd_index	command -index	-
wait_prvdata_complete	1	2- Sends command immediately 3- Sends command after previous data transfer ends
check_response_crc	1	2- SD/MMC controller should not check response CRC 3- SD/MMC controller should check response

		CRC
--	--	-----

After writing to the CMD register, SD/MMC controller starts executing a command; when the command is sent to the bus, a command\_done interrupt is generated.

6. Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending the STOP command.
7. Software should look for Transmit\_FIFO\_Data\_request and/or timeout conditions from data starvation by the host. In both cases, the software should write data into the FIFO.
8. When a Data\_Transfer\_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the SD/MMC controller should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto\_command\_done interrupt – bit 14 of the RINTSTS register. A response to AUTO\_STOP is stored in RESP1 @0x34.

### **Stream Read**

A stream read is like the block read mentioned in ["Single-Block or Multiple-Block Read"](#), except for the following bits in the Command register:

transfer\_mode = 1; //Stream transfer

cmd\_index = CMD20;

A stream transfer is allowed for only a single-bit bus width.

### **Stream Write**

A stream write is exactly like the block write mentioned in ["Single-Block or Multiple-Block Write"](#), except for the following bits in the Command register:

transfer\_mode = 1; //Stream transfer

cmd\_index = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a transfer, the SD/MMC controller sends the STOP command. Completion of this AUTO\_STOP command is reflected by the Auto\_command\_done interrupt. A response to an AUTO\_STOP is stored in the RESP1 register @0x34.

A stream transfer is allowed for only a single-bit bus width.

### **Sending Stop or Abort in Middle of Transfer**

The STOP command can terminate a data transfer between a memory card and the SD/MMC controller, while the ABORT command can terminate an I/O data transfer for only the SDIO\_IOONLY and SDIO\_COMBO cards.

- Send STOP command – Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer. For information on sending this command, refer to ["No-Data Command With or Without Response Sequence"](#).

You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (stop\_abort\_cmd) to 1. If stop\_abort\_cmd is not set to 1, the SD/MMC controller does not know that the user stopped a data transfer. Reset bit 13 of the Command register (wait\_prvdata\_complete) to 0 in order to make the SD/MMC controller send the command at once, even though there is a data transfer in progress.

- Send ABORT command – Can be used with only an SDIO\_IOONLY or SDIO\_COMBO card. To abort the function that is transferring data, program the function number in ASx bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

This is a non-data command. For information on sending this command, refer to ["No-Data Command With or Without Response Sequence"](#).

The command format for CMD52 is illustrated in Fig. 6-7:



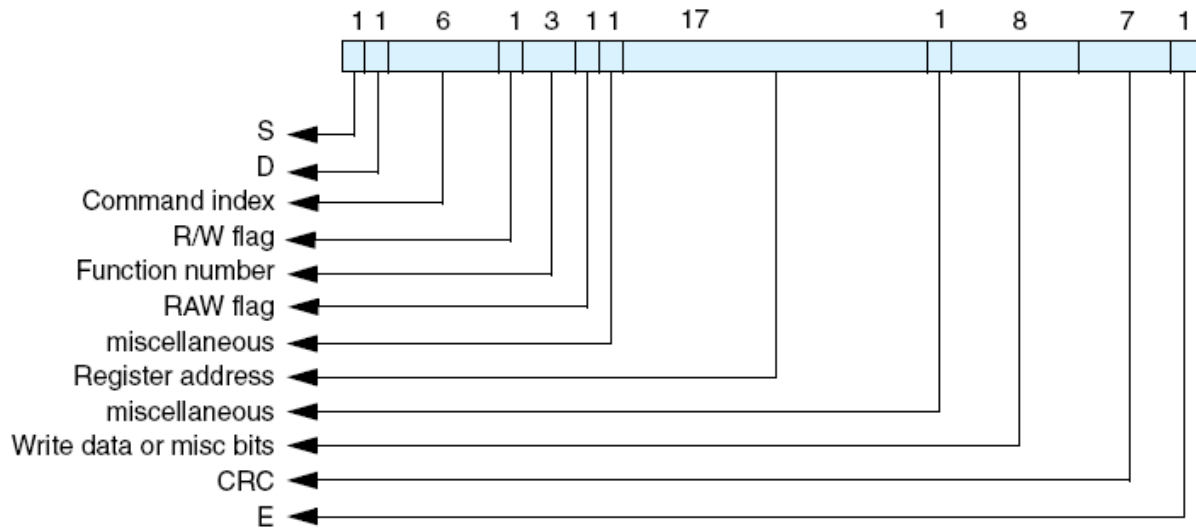


Fig. 7-7 SD/MMC Command format for CMD52

- Program the CMDARG register @0x28 with the appropriate command argument parameters listed in [Table 6-5](#).

Table 7-5 SD/MMC Parameters for CMDARG Registers

CMDARG Bits	Contents	Value
31	R/W flag	1
30-28	Function Number	0, for CCCR access
27	RAW flag	1, if needed to read after write
26	Don't care	-
25-9	Register address	0x06
8	Don't care	-
7-0	Write Data	Function number to be aborted

- Program the Command register using the command index as CMD52. Similar to the STOP command described, set bit 14 of the Command register (stop\_abort\_cmd) to 1, which must be done in order to inform the SD/MMC controller that the user aborted the data transfer. Reset bit 13 (wait\_prvdata\_complete) of the Command register to 0 in order to make the SD/MMC controller send the command at once, even though a data transfer is in progress.
- Wait for command\_transfer\_over.
- Check response (R5) for errors.

### **Suspend or Resume Sequence**

In an SDIO card, the data transfer between an I/O function and the SD/MMC controller can be temporarily halted using the SUSPEND command; this may be required in order to perform a high-priority data transfer with another function. When desired, the data transfer can be resumed using the RESUME command.

The following functions can be implemented by programming the appropriate bits in the CCCR register (Function 0) of the SDIO card. To read from or write to the CCCR register, use the CMD52 command.

- SUSPEND data transfer – Non-data command.
  - Check if the SDIO card supports the SUSPEND/RESUME protocol; this can be done through the SBS bit in the CCCR register @0x08 of the card.
  - Check if the data transfer for the required function number is in process; the function number that is currently active is reflected in bits 0-3 of the CCCR register @0x0D. Note that if the BS bit (address 0xc::bit 0) is 1, then only the function number given by the FSx bits is valid.
  - To suspend the transfer, set BR (bit 2) of the CCCR register @0x0C.
  - Poll for clear status of bits BR (bit 1) and BS (bit 0) of the CCCR @0x0C. The BS

- (Bus Status) bit is 1 when the currently-selected function is using the data bus; the BR (Bus Release) bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function has been suspended.
- e. During a read-data transfer, the SD/MMC controller can be waiting for the data from the card. If the data transfer is a read from a card, then the SD/MMC controller must be informed after the successful completion of the SUSPEND command. The SD/MMC controller then resets the data state machine and comes out of the wait state. To accomplish this, set `abort_read_data` (bit 8) in the Control register.
  - f. Wait for data completion. Get pending bytes to transfer by reading the TCBCNT register @0x5C.
2. RESUME data transfer – This is a data command.
- a. Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.
  - b. If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.
  - c. Check that a function to be resumed is ready for data transfer; this can be confirmed by reading the RFx flag in CCCR @0x0F. If RF = 1, then the function is ready for data transfer.
  - d. To resume transfer, use CMD52 to write the function number at FSx bits (0-3) in the CCCR register @0x0D. Form the command argument for CMD52 and write it in CMDARG @0x28; bit values are listed in [Table 6-6](#).

Table 7-6 SD/MMC CMDARG Bit Values

CMDARG Bits	Contents	Value
31	R/W flag	1
30-28	Function Number	0, for CCCR access
27	RAW flag	1, read after write
26	Don't care	-
25-9	Register address	0x0D
8	Don't care	-
7-0	Write Data	Function number to be resumed

- e. Write the block size in the BLKSIZ register @0x1C; data will be transferred in units of this block size.
- f. Write the byte count in the BYTCNT register @0x20. This is the total size of the data; that is, the remaining bytes to be transferred. It is the responsibility of the software to handle the data.
- g. Program Command registers; similar to a block transfer. For details, refer to ["Single-Block or Multiple-Block Read"](#) and ["Single-Block or Multiple-Block Write"](#).
- h. When the Command register is programmed, the command is sent and the function resumes data transfer. Read the DF flag (Resume Data Flag). If it is 1, then the function has data for the transfer and will begin a data transfer as soon as the function or memory is resumed. If it is 0, then the function has no data for the transfer.
- i. If the DF flag is 0, then in case of a read, the SD/MMC Controller waits for data. After the data timeout period, it gives a data timeout error.

### Read\_Wait Sequence

`Read_wait` is used with only the SDIO card and can temporarily stall the data transfer—either from function or memory—and allow the host to send commands to any function within the SDIO device. The host can stall this transfer for as long as required. The SD/MMC Controller provides the facility to signal this stall transfer to the card. The steps for doing this are:

1. Check if the card supports the `read_wait` facility; read SRW (bit 2) of the CCCR register @0x08. If this bit is 1, then all functions in the card support the `read_wait`



facility. Use CMD52 to read this bit.

2. If the card supports the read\_wait signal, then assert it by setting the read\_wait (bit 6) in the CTRL register @0x00.
3. Clear the read\_wait bit in the CTRL register.

### **Controller/DMA/FIFO Reset Usage**

Communication with the card involves the following:

- Controller – Controls all functions of the SD/MMC controller.
- FIFO – Holds data to be sent or received.
- DMA – If DMA transfer mode is enabled, then transfers data between system memory and the FIFO.
- Controller reset – Resets the controller by setting the controller\_reset bit (bit 0) in the CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset – Resets the FIFO by setting the fifo\_reset bit (bit 1) in the CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- DMA reset – Resets the internal DMA controller logic by setting the dma\_reset bit (bit 2) in the CTRL register, which abruptly terminates any DMA transfer in process. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

The following are recommended methods for issuing reset commands:

- Non-DMA transfer mode – Simultaneously sets controller\_reset and fifo\_reset; clears the RAWINTS register @0x44 using another write in order to clear any resultant interrupt.
- DMA mode – Sets controller\_reset and fifo\_reset; waits until dma\_req goes inactive (the Status register indicates the value of this signal). Resets the FIFO again. Clears the interrupts by clearing the RAWINTS register @0x44 using another write in order to clear any resultant interrupt. You also need to reset and reprogram the channel(s) of the DW\_dmac controller that are interfaced to the SD/MMC Controller.

In external DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the RAWINTS register caused by the DMA transfers after the FIFO was reset.

### **Error Handling**

The SD/MMC controller implements error checking; errors are reflected in the RAWINTS register @0x44 and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (int\_enable in the CTRL register is 0), and all the interrupts are masked (bits 0-31 of the INTMASK register; default is 0).

Error handling:

- Response and data timeout errors – For response timeout, software can retry the command. For data timeout, the SD/MMC controller has not received the data start bit – either for the first block or the intermediate block – within the timeout period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the TCBCNT later, the software can decide how many bytes remain to be copied.
- Response errors – Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors – Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.

- Hardware locked error – Set when the SD/MMC controller cannot load a command issued by software. When software sets the start\_cmd bit in the CMD register, the SD/MMC controller tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
- FIFO underrun/overflow error – If the FIFO is full and software tries to write data in the FIFO, then an overflow error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read the fifo\_empty or fifo\_full bits in the Status register.
- Data starvation by host timeout – Raised when the SD/MMC controller is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated timeout period. Under this condition and when a read transfer is in process, the software should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command – If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response timeout is expected from the SD/MMC controller. The ATA layer is notified that an MMC transport layer error occurred.

**Note:**

During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register. It then continues further data transmission until all the bytes are transmitted.

**Auto-Stop**

The SD/MMC controller internally generates a stop command and is loaded in the command path when the send\_auto\_stop bit is set in the Command register. The auto-stop command helps to send an exact number of data bytes using a stream read or write for the MMC, and a multiple-block read or write for SD memory transfer for SD cards.

The software should set the send\_auto\_stop bit according to details listed in Table 6-7.

Table 7-7 SD/MMC Auto-Stop Generation condition list

Card type	Transfer type	Byte Count	send_auto_stop bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	> 0	Yes	Auto-stop after all bytes transfer
MMC	Stream write	0	No	Open-ended stream
MMC	Stream write	> 0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	> 0	No	Byte count = 0 is illegal
MMC	Single-block write	> 0	No	Byte count = 0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	> 0	Yes **	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	> 0	Yes **	Pre-defined multiple block
SDMEM	Single-block read	> 0	No	Byte count = 0 is illegal
SDMEM	Single-block write	> 0	No	Byte count = 0 illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	> 0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block
SDMEM	Multiple-block write	> 0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	> 0	No	Byte count = 0 is illegal
SDIO	Single-block write	> 0	No	Byte count = 0 illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block
SDIO	Multiple-block read	> 0	No	Pre-defined multiple block
SDIO	Multiple-block write	0	No	Open-ended multiple block
SDIO	Multiple-block write	> 0	No	Per-defined multiple block

\*\* The condition under which the transfer mode is set to block transfer and *byte\_count* is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If *byte\_count* = *n\*block\_size* (*n* = 2, 3, ...), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the host software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue CMD18/CMD25 commands without setting the send\_auto\_stop bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the send\_auto\_stop bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop command after the programmed byte count.

The following list conditions for the auto-stop command.

- Stream read for MMC card with byte count greater than 0 – The SD/MMC controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 – The SD/MMC controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.
- Multiple-block read memory for SD card with byte count greater than 0 – If the block

size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.

- Multiple-block write memory for SD card with byte count greater than 0 – If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.
- Precaution for host software during auto-stop – Whenever an auto-stop command is issued, the host software should not issue a new command to the SD/MMC controller until the auto-stop is sent by the SD/MMC controller and the data transfer is complete. If the host issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the host wants to terminate the data transfer before the data transfer is complete, it can issue a stop or abort command, in which case the SD/MMC controller does not generate an auto-stop command.

## Chapter 8 Video Input Processor(VIP)

### 8.1 Design Overview

#### 8.1.1 Overview

The Video Input Processor receives the data from Camera or CCIR656 encoder, and transfers the data into system main memory by AHB bus.

#### 8.1.2 Features

- Support CMOS type image sensor interface
- Support CCIR656 interface
- Support CCIR-656 YCbCr 4:2:2 raster video input for 8bit mode in 525/60 NTSC and 625/50 PAL video system
- Data input clock is 27MHz for CCIR656 and 24MHz/48MHz for sensor, and max up to 96MHz for raw data
- Provide YUV 4:2:2/4:2:0 output
- Support up to (3856x2764) resolution, 10M pixel
- Support YUYV/UYVY format input
- Support 10/12-bit raw data input
- In Sensor Mode, support Vsync and Href High active or Low active configurable
- Support bypass path from VIP to LCDC

*Notes* vsync porality control is programmable by bit 7 of register CPU\_APB\_REG5, refer to Chapter 34 (General Register File in CPU System) for detailed information

### 8.2 Architecture

This section provides a description about the functions and behavior under various conditions.

#### 8.2.1 Block Diagram

The VIP comprises with:

- AHB Slave – Host configure the VIP\_Reg via the AHB Slave
- AHB Master – VIP transmit the data to chip memory via the AHB Master
- VIP\_REG – The register bank store the status and configuration information
- YUV Interface – translate the input video data into the requisite data format.
- DMA Control – Manage the memory buffer
- Buffer control and line buffer – store the translated video data.

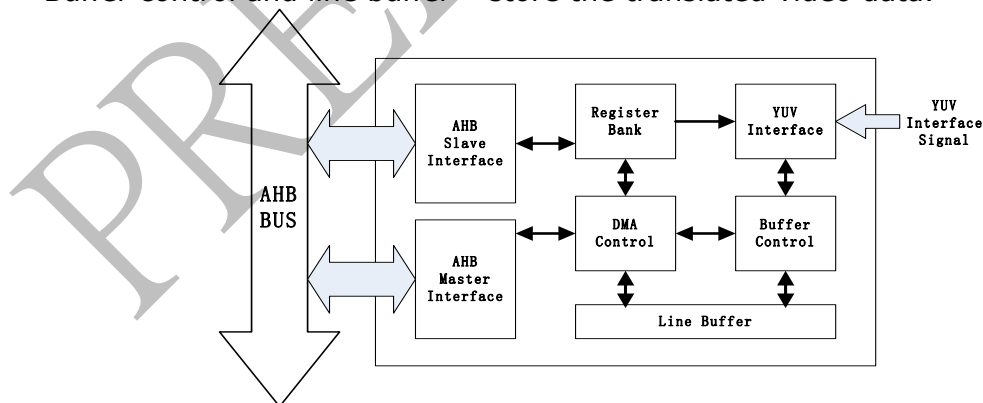


Fig. 8-1 RK281x VIP design architecture

### 8.3 Registers

This section describes the registers of the design.

#### 8.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
VIP_AHBR_CTRL	0x00	W	0x1	AHB write control register

VIP_INT_MASK	0x04	W	0x0	Interrupt Mask register
VIP_INT_STS	0x08	W	0x0	Interrupt status register
VIP_STS	0x0C	W	0x0	VIP Status register
VIP_CTRL	0x10	W	0x0	VIP control register
VIP_CAPTURE_F1S A_Y	0x14	W	0xFFFFFFFF	Capture data frame 1 start address for Y
VIP_CAPTURE_F1S A_UV	0x18	W	0xFFFFFFFF	Capture data frame 1 start address for UV
VIP_CAPTURE_F1S A_Cr	0x1C	W	0xFFFFFFFF	Capture data frame 1 start address for Cr
VIP_CAPTURE_F2S A_Y	0x20	W	0xFFFFFFFF	Capture data frame 2 start address for Y
VIP_CAPTURE_F2S A_UV	0x24	W	0xFFFFFFFF	Capture data frame 2 start address for UV
VIP_CAPTURE_F2S A_Cr	0x28	W	0xFFFFFFFF	Capture data frame 2 start address for Cr
VIP_FB_SR	0x2C	W	0xB	Frame buffer status register for capturing raw data
VIP_FS	0x30	W	0x02D001E6	Frame data size register
VIP_CROP	0x38	W	0x0	Cropping start upper left point to other little resolution
VIP_CRM	0x3C	W	0x0	Y/U/V color modification
VIP_RESET	0x40	W	0x0	Capture engine reset
VIP_L_SFT	0x44	W	0x0	Line shifter from first line

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 8.3.2 Detail Register Description

#### VIP\_AHBR\_CTRL

Address: Operational Base + offset (0x00)

AHB write control register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2:0	RW	0x5	<p>AHB Data Maximum Burst Length for Reading from H/W.</p> <p>This register will set the maximum length to transmit data to AHB bus. The actual data length to be transmitted will be decided by H/W automatically. For example, if INCR8 is set, only 8 or 4 will be the actual length.</p> <p>The following is the meaning.</p> <p>001: INCR 101: INCR8 111: INCR16 Other: unused</p>

#### VIP\_AHBR\_MASK

Address: Operational Base + offset (0x04)

Interrupt Mask register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Capture data line end happened interrupt enable

			1: enable (for debug, just a cycle pulse)
1	RW	0x0	Capture frame loss happened interrupt enable.(only for 656 mode) 0: Disable 1: Enable
0	RW	0x0	Capture complete interrupt enable 0: Disable 1: Enable

**VIP\_INT\_STS**

Address: Operational Base + offset (0x08)

Interrupt status register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	Capture data line end happened interrupt (Read Clear, just a cycle pulse)(for debug)
1	R	0x0	Capture frame loss happened interrupt. (Read Clear, only for 656 mode) 0: No interrupt happen 1: Interrupt happen
0	R	0x0	Capture complete interrupt (Read Clear) 0: No interrupt happen 1: Interrupt happen

**VIP\_STS**

Address: Operational Base + offset (0x0C)

Status register

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	FIFO overflow, 1:active (Read Clear)

**VIP\_CTRL**

Address: Operational Base + offset (0x10)

VIP control register

Bit	Attr	Reset Value	Description
31:15	-	-	Reserved
14:13	RW	0x0	Raw_input_edge select 00: bypass 01: posedge capture 10: negedge capture 11: default(by pass)
12	RW	0x0	Vsync sentive 0: low active 1: high active
11	RW	0x0	Little_end or big_end 0: little_end 1: big_end
10	RW	0x0	CCIR656 capture format 0: NTSC 1: PAL
9	RW	0x0	Posedge/Negedge capture by pixel clock



			0: Positive edge 1: Negative edge
8	RW	0x0	Ping-Pong mode enable 0: Continuous mode 1: Ping-Pong mode Note1: Bit5 priority > Bit8 Note2: Only both Bit5 and Bit8 be set to 0 can enable continuous mode
7	RW	0x0	Field capture for ccir format 0: Field 0 start 1: Field 1 start
6	RW	0x0	422 output enable 0: 420 output (write to memory) 1: 422 output (write to memory)
5	RW	0x0	One frame stop enable 0: continuous mode or ping-pong mode 1: one frame complete stop
4	RW	0x0	YUV or Raw: 0: YUV 1: RAW
3	RW	0x0	Input data order, Y or UV first 0: UYVY 1: YUYV
2	RW	0x0	Sensor_or_656 0: 656 1: sensor
1	RW	0x0	Href_sensitive 0: High active 1: Low active
0	RW	0x0	Enable capturing (set and clear by host) To set this bit to enable capturing, and clear it by host to disable capturing. (set and clear by host) 0: Disable 1: Enable

**VIP\_CAPTURE\_F1SA\_Y**

Address: Operational Base + offset (0x14)

Capture raw data frame 1 start address for Y

Bit	Attr	Reset Value	Description
31:0	RW	0xffffffff	Capture Frame 1 Starting Address Register for Y

**VIP\_CAPTURE\_F1SA\_UV**

Address: Operational Base + offset (0x18)

Capture raw data frame 1 start address for UV

Bit	Attr	Reset Value	Description
31:0	RW	0xffffffff	Capture Frame 1 Starting Address Register for UV

**VIP\_CAPTURE\_F2SA\_Y**

Address: Operational Base + offset (0x20)

Capture raw data frame 1 start address for Y

Bit	Attr	Reset Value	Description
31:0	RW	0xffffffff	Capture Frame 2 Starting Address Register for Y

**VIP\_CAPTURE\_F2SA\_UV**

Address: Operational Base + offset (0x24)

Capture raw data frame 1 start address for UV

Bit	Attr	Reset Value	Description
31:0	RW	0xffffffff	Capture Frame 2 Starting Address Register for UV

### VIP\_FB\_SR

Address: Operational Base + offset (0x2C)

Frame buffer status register for capturing raw data

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved
15: 8	RW	0x0	Frame number. Complete VIP number
7:4	-	-	Reserved
3	R	0x0	Indicate the latest used Frame buffer number, for example, if Bit0 and Bit1 are both 1 and Bit3 is 1, means that Frame 2 is capture finally. 0: Frame 1 1: Frame 2
2	RW	0x0	Indicate Frame loss (set by H/w and clear by HOST) 0: No frame loss 1: Frame loss occurred
1	RW	1x0	Status bit to indicate current status of frame 2 (set by H/W and clear by HOST) 0: data not ready 1: data ready Note: After reading this register, HOST shall assign new buffer addresses to "Capture Raw Frame 1/2 Starting Address Register for Y/Cb/Cr" and clear the status register for H/W capturing next frame to keep Ping-Pong mode enable.
0	RW	1x0	Status bit to indicate current status of frame 1 (set by H/W and clear by HOST) 0: data not ready 1: data ready Note: After reading this register, HOST shall assign new buffer addresses to "Capture Raw Frame 1/2 Starting Address Register for Y/Cb/Cr" and clear the status register for H/W capturing next frame to keep Ping-Pong mode enable.

### VIP\_FS

Address: Operational Base + offset (0x30)

Frame data size register

Bit	Attr	Reset Value	Description
31:16	RW	0x0	Pixel number per line width up to $2^{16}-1$
15: 0	RW	0x0	Line numbers per frame Height up to $2^{16}-1$

### VIP\_CROP

Address: Operational Base + offset (0x38)

Cropping start upper left point to other little resolution

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:16	RW	0x0	The X-coordinate of the cropping start point at up-left corner
15:10	-	-	Reserved.
9:0	RW	0x0	The Y-coordinate of the cropping start point at up-left corner

**VIP\_CRM**

Address: Operational Base + offset (0x3C)

Y/CB/CR color modification

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26	RW	0x0	Y direction, 0-decrease 1-increase
25	RW	0x0	Cb direction, 0-decrease 1-increase.
24	RW	0x0	Cr direction, 0-decrease 1-increase
23:16	RW	0x0	Y value
15:8	RW	0x0	Cb value
7:0	RW	0x0	Cr value

**VIP\_RESET**

Address: Operational Base + offset (0x40)

Capture engine reset

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Engine Reset (Refer to reset flow of video input processor) Value: 0x76543210 to reset

**VIP\_L\_SFT**

Address: Operational Base + offset (0x44)

Line Shifter from first line

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x0	Line shifter from first line, it is used in non-standard ccir656 input (not precisely 480/576 active line). Set this register can cut the lines at the first of both fields. Valid value: 0~15

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 8.4 Functional Description

This chapter is used to illustrate the operational behavior of how VIP module works. VIP module receive the sensor or ccir656 signal from external devices and translate it into YUV422/420 data, separate the data to Y and UV data, then store them to different memory via AHB bus separately. If the input data is raw data, VIP store them in the one memory with the input order.

### 8.4.1 Operation

#### Software/Hardware Reset

When RESETN pin is set to low, it will cause everything including both VIP and AHB modules to be reset to default state immediately.

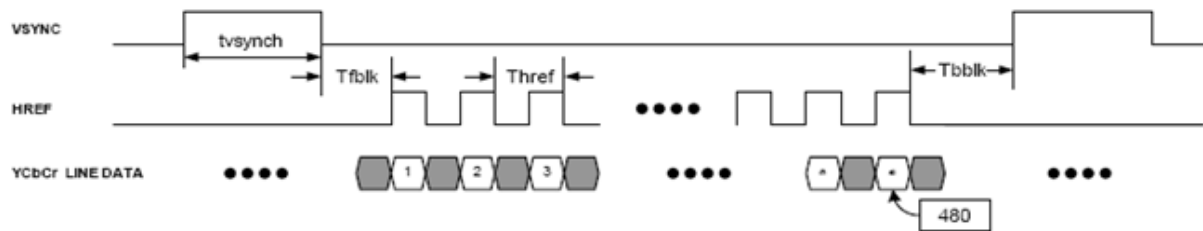
While configuring the VIP\_RESET(offset 0x40) register with 0x76543210, VIP will be software reset after 200 cycles of AHB.

#### Input data format

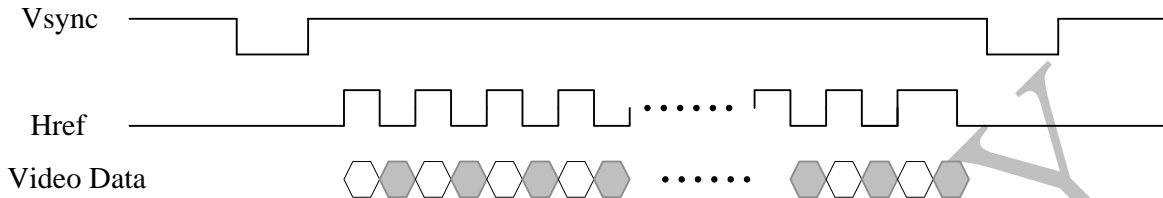
The VIP module support the 8bit YUV422 and CCIR656 ,10/12-bit raw data input.

1. Support Vsync high active or low active

- Vsync Low active as below:

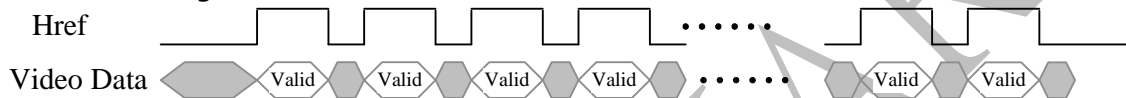
**Vertical sensor timing (line by line)**

- Vsync High active :

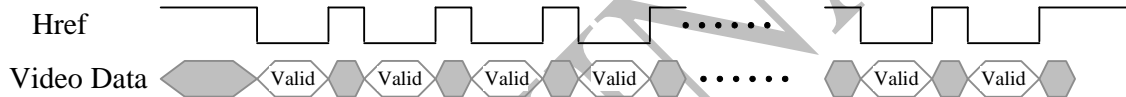


## 2. Support href high active or low active

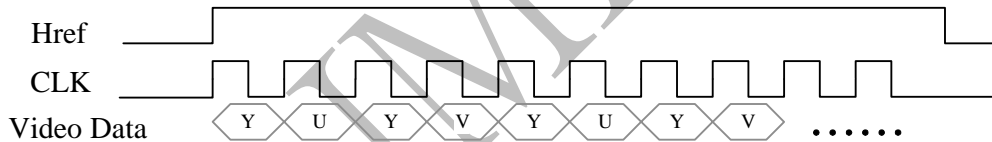
- Href high active:



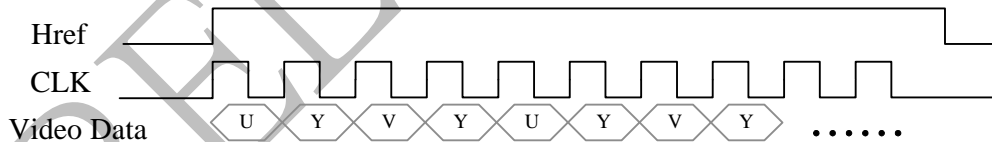
- Href Low active



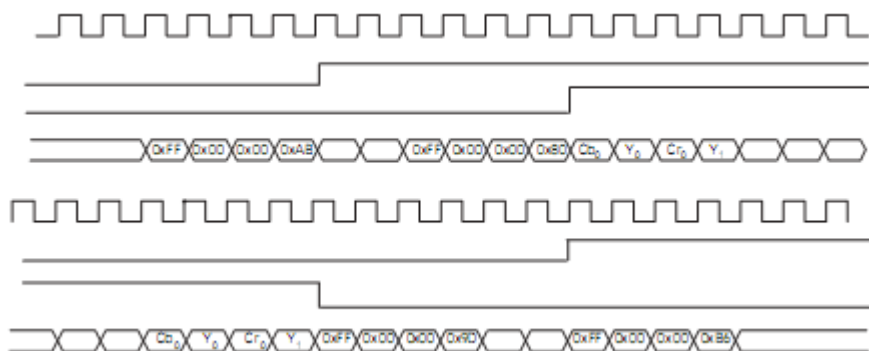
- Y first



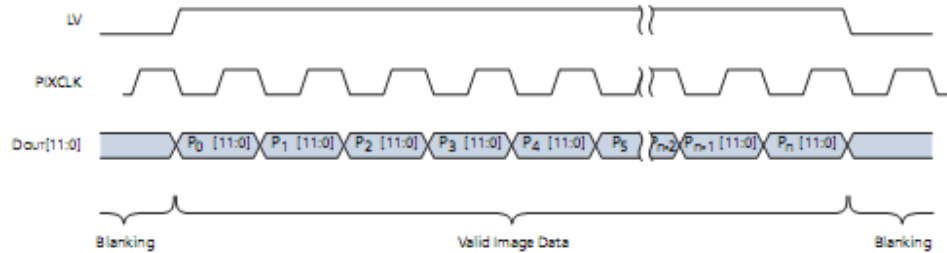
- U first



## 3. Support CCIR656 (NTSC and PAL)



## 4. Support Raw data(10/12-bit)

**Pixel Data Timing Example****YUV or CCIR656****Initial Configuration**

After HW/SW reset, host must initially configure VIP by control registers via AHB bus. The configurations include external device, frame data start address, frame size, output format... etc.

It is recommended that you can configure the VIP register as the follow the sequence:

1. Set the VIP\_AHBR\_CTRL to config the AHB Burst Length(INCR, INCR8 or INCR16)
2. Set the VIP\_INT\_MASK to config the VIP interrupt mask
3. Set the VIP\_CTRL, to config the parameter of the VIP, for example, ccir656 format, vip work mode(ping-pong/continue/one frame), input data order(y or uv first)...etc. Don't set the VIP\_enable bit (bit[0]) to 1 at this time.
4. Set the VIP\_CAPTURE\_F1SA\_Y, VIP\_CAPTURE\_F1SA\_UV, VIP\_CAPTURE\_F2SA\_Y, VIP\_CAPTURE\_F2SA\_UV to config the Y/UV start address of the frame1 and frame 2.
5. Set the VIP\_FS to config the frame size
6. Set the VIP\_FB\_SR to clear the frame buffer status
7. In the end, start the vip by setting the VIP\_enable bit(bit[0] in VIP\_CTRL) to 1.

VIP module can work in three modes: one frame stop mode、ping-pong mode、continuous mode.

**One frame stop mode (only frame1 can be used in this mode)**

In this mode, only need to set VIP\_CAPTURE\_F1SA\_<Y,UV> as start memory address of the continuous memory. Before trigger VIP to start capture, frame1 buffer status in VIP\_FB\_SR register(bit 0) must be clear to 1'b0. Then configure register VIP\_CTRL(bit0) to start one frame stop mode. After one frame captured, VIP will automatic stop. After capturing, the image Y, UV data will be stored at main memory location defined by VIP\_CAPTURE\_F1SA\_Y, VIP\_CAPTURE\_F1SA\_UV separately.

**Ping-Pong mode**

In this mode, need to set VIP\_CAPTURE\_F1SA\_<Y, UV> and VIP\_CAPTURE\_F2SA\_<Y, UV>. Before trigger VIP to start capture, frame1 & frame2 buffer status in VIP\_FB\_SR register(bit1:0) must be clear to 2'b00. Then configure VIP\_CTRL to start ping-pong mode. After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically, and host must assign new address pointer of frame1 and clear the frame1 status, thus VIP will capture the third frame automatically(by new F1 address) without any stop and so on for the following frames. But if host did not update the frame buffer address, the VIP will cover the pre-frame data stored in the memory with the following frame data. And if host did not clear the frame status, the VIP will stop after both 2 frame buffer (F1&F2) are at data ready state(bit[1:0] of VIP\_FB\_SR register is 2'b11).

**Continuous mode**

In this mode, need to set VIP\_CAPTURE\_F1SA\_<Y, UV> and VIP\_CAPTURE\_F2SA\_<Y, UV>. Before trigger VIP to start capture, frame1 & frame2 buffer status in VIP\_FB\_SR register(bit1:0) must be clear to 2'b00. Then configure VIP\_CTRL to start ping-pong mode. After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically. If you setup register VIP\_INT\_MASK to be value 0x1 you will get

a complete interrupt from VIP after every frame end. Note that the continuous mode will use both F1 & F2 pointer just like in ping-pong mode, but the difference between continuous mode and ping-pong mode is that continuous mode will never stop the VIP unless host disable VIP directly even the F1 & F2 status are both in data ready state.

## Raw Data

### Initial Configuration

VIP can support 10-bit or 12-bit raw data input, before configure the VIP register, set the iomux for 10/12-bit raw data input, refer the **IOMUX\_B\_CON** register.

Then you can configure the VIP register as the follow the sequence:

1. Set the VIP\_AHBR\_CTRL to configure the AHB Burst Length(INCR, INCR8 or INCR16)
2. Set the VIP\_INT\_MASK to configure the VIP interrupt mask
3. Set the VIP\_CTRL, you must set the 656\_or\_sensor(bit[2]), to 1 and set yuv\_or\_raw(bit[4]) to 1, you also can set the little\_end or big\_end(bit[11]) . Don't set the VIP\_enable bit (bit[0]) to 1 at this time.
4. Set the VIP\_CAPTURE\_F1SA\_Y, VIP\_CAPTURE\_F2SA\_Y to configure the Y/UV start address of the frame1 and frame 2.
5. Set the VIP\_FS to config the frame size
6. Set the VIP\_FB\_SR to clear the frame buffer status
7. In the end, start the vip by setting the VIP\_enable bit(bit[0] in VIP\_CTRL) to 1.

In the raw data mode, VIP module can work in three modes same as YUV mode : one frame stop mode、ping-pong mode、continuous mode.

**Note:** Difference between the YUV mode and raw mode is that in the YUV mode or ccir656 mode, data will be storage in the Y data buffer and UV data buffer; but in the raw mode, RGB data will be storage in the same buffer. In addition, in the yuv mode, four pixel data are storage in a word, and in the raw mode, two pixel data are storage in a word, one pixel is storage in the low 16 bits, other pixel is storage in the high 16 bits.

## 8.5 Bypass from VIP to LCDC Functional Description

VIP interface can be used to bypass directly to LCDC interface of RK281x , This function is enabled by programmable register set. As for the detailed information, please refer to bit 16 of CPU\_APB\_REG5 in Chapter 34 .

The following table will list pin mapping between VIP and LCDC interface.

Table 7-1 Pin mapping between VIP and LCDC interface

VIP Pin Name	VIP port name	LCDC port name	LCDC Pin Name
IO_VIP_DATAIN[11:4]	vip_datain[11:4]	lcdc_wdata[7:0]	IO_LCDC_DATA[7:0]
IO_VIP_VSYNC	vip_vsync	lcdc_vsync	IO_GPIO2[25]
IO_VIP_HREF	vip_href	lcdc_hsync	IO_LCDC_HSYNC
IO_VIP_CLKIN	vip_clkin	lcdc_dclk	IO_LCDC_DCLK

## Chapter 9 LCD Controller

### 9.1 Design Overview

#### 9.1.1 Overview

LCD Controller is the display interface from memory frame buffer to display device (LCD panel or TV set). LCDC is connected to LCDC\_AHB bus through an AHB slave and an AHB master. The register setting is configured through the AHB slave interface and the display frame data is read through the AHB master interface.

#### 9.1.2 Features

##### ◆ Display interface

- Parallel RGB LCD Interface: 24-bit(RGB888), 18-bit(RGB666), 15-bit(RGB565)
- Serial RGB LCD Interface: 3x8-bit(RGB delta support), 3x8-bit + dummy, 16-bit + 8-bit
- MCU LCD interface: i-8080(up to 24-bit RGB), Hold/Auto/Bypass modes
- TV Interface: ITU-R 656

##### ◆ Display process

- One Background layer: programmable 24-bit color
- One Video layer(win0):
  - RGB888, ARGB888, RGB565, YCbCr422, YCbCr420-0, YCbCr420-1, YCbCr420-M, YCbCr444
  - Maximum resolution is 1280x720
  - Virtual display(roller support)
  - 1/8 to 8 scaling-down and scaling-up engine
  - 256 level alpha blending(no scaling in ARGB mode)
  - Transparency color key
  - Rotation 270-degrees(YCbCr422, YCbCr420-0, YCbCr420-1, maximum resolution is 720x1280)
  - De-interlace support for YCbCr422, YCbCr420-0 mode
  - De-flicker support for interlace output
- One Graphic layer(win1):
  - RGB888, ARGB888, RGB565
  - Maximum resolution is 1024x768
  - Virtual display(roller support)
  - 1/4 to 4 scaling-down and scaling-up engine
  - 256 level alpha blending
  - Transparency color key
- Hardware cursor(HWC):
  - 32x32x2bpp (bit per pixel).
  - 3-color and transparent mode
  - 16 level alpha blending

##### ◆ Others

- Graphic layer and Video layer overlay exchangeable
- YCbCr2RGB(rec601-mpeg/ rec601-jpeg/rec709) and RGB2YCbCr modules
- Replication(16bpp to 24bpp) and Dithering(24bpp to 16bpp/18bpp)
- Blank and black display
- Standby mode



## 9.2 Architecture

This section provides a description about the functions and behavior under various conditions.

### 9.2.1 Block Diagram

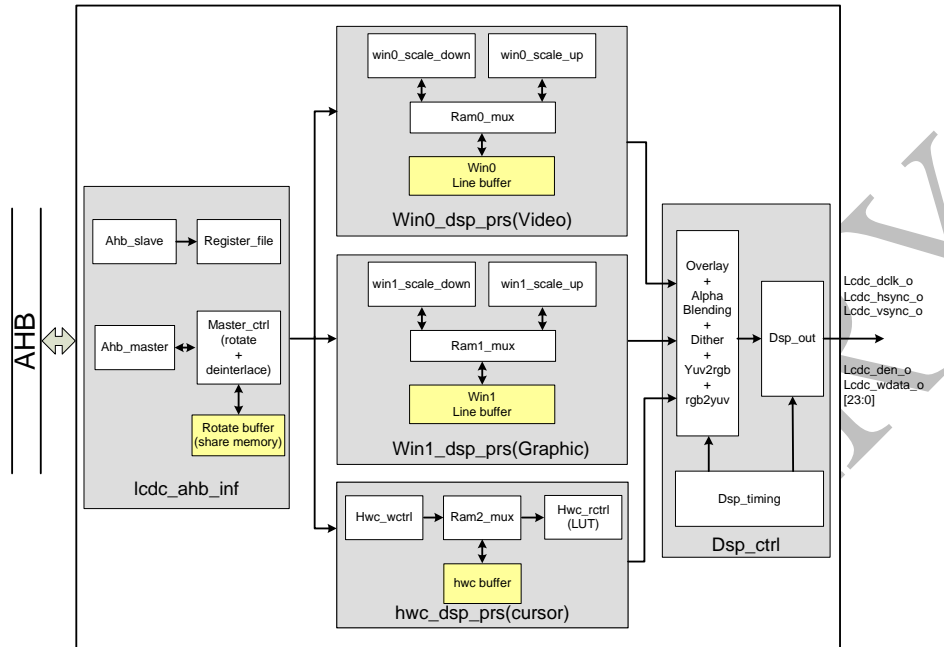


Fig. 9-1 RK281x LCDC Block Diagram

## 9.3 Register Definition

This section describes the control/status registers of the design. These registers should be accessed with 32-bit bus-width.

### 9.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
LCDC_SYS_CONFIG	0x00	W	0x0	SYSTEM configure register
LCDC_SWAP_CTRL	0x04	W	0x0	Data SWAP control
LCDC_MCU_TIMING_CTRL	0x08	W	0x0	MCU TIMING control register
LCDC_BLEND_CTRL	0x0C	W	0x0	Blending control register
LCDC_WIN0_COLOR_KEY_CTRL	0x10	W	0x0	Win0 blending control register
LCDC_WIN1_COLOR_KEY_CTRL	0x14	W	0x0	Win1 blending control register
LCDC_DEFLICKER_SCL_OFF SET	0x18	W	0x0	Deflick scaling start point offset
LCDC_DSP_CTRL0	0x1C	W	0x00041000	Display control register0
LCDC_DSP_CTRL1	0x20	W	0x01000000	Display control register1
LCDC_INT_STATUS	0x24	W	0x00000380	Interrupt status register
LCDC_WIN0_VIR	0x28	W	0x0	WIN0 virtual display width/height
LCDC_WIN0_YRGB_MST	0x2C	W	0x0	Win0 YRGB memory start address
LCDC_WIN0_CBR_MST	0x30	W	0x0	Win0 Cbr memory start

				address
LCDC_WIN0_ACT_INFO	0x34	W	0x0	Win0 active window width/height
LCDC_WIN0_ROLLER_INFO	0x38	W	0x0	Win0 x and y value of start point in roller mode
LCDC_WIN0_DSP_ST	0x3C	W	0x0	Win0 display start point on panel
LCDC_WIN0_DSP_INFO	0x40	W	0x0	Win0 display width/height on panel
LCDC_WIN1_VIR	0x44	W	0x0	Win1 virtual display width/height
LCDC_WIN1_CBR_MST	0x48	W	0x0	Win1 memory start address
LCDC_WIN1_ACT_INFO	0x4C	W	0x0	Win1 active width /height
LCDC_WIN1_ROLLER_INFO	0x50	W	0x0	Win1 x and y value of start point in roller mode
LCDC_WIN1_DSP_ST	0x54	W	0x0	Win1 display start point on panel
LCDC_WIN1_DSP_INFO	0x58	W	0x0	Win1 display width/height on panel
LCDC_HWC_MST	0x5C	W	0x0	HWC memory start address
LCDC_HWC_DSP_ST	0x60	W	0x0	HWC display start point on panel
LCDC_HWC_COLOR_LUT0	0x64	W	0x0	Hardware cursor color 2'b01 look up table 0
LCDC_HWC_COLOR_LUT1	0x68	W	0x0	Hardware cursor color 2'b10 look up table 1
LCDC_HWC_COLOR_LUT2	0x6C	W	0x0	Hardware cursor color 2'b11 look up table 2
LCDC_DSP_HTOTAL_HS_END	0x70	W	0x017F0017	Panel scanning horizontal width and hsync pulse end point
LCDC_DSP_HACT_ST_END	0x74	W	0x02B016A	Panel active horizontal scanning start/end point
LCDC_DSP_VTOTAL_VS_END	0x78	W	0x01070002	Panel scanning vertical height and vsync pulse end point
LCDC_DSP_VACT_ST_END	0x7C	W	0x000D00FC	Panel active vertical scanning start/end point
LCDC_DSP_VS_ST_END_F1	0x80	W	0x0	Vertical scanning start point and vsync pulse end point of even filed in interlace mode
LCDC_DSP_VACT_ST_END_F1	0x84	W	0x0	Vertical scanning active start/end point of even filed in interlace mode
LCDC_WIN0_SD_FACTOR_Y	0x88	W	0x10001000	Win0 YRGB scaling down factor setting
LCDC_WIN0_SP_FACTOR_Y	0x8C	W	0x10001000	Win0 YRGB scaling up factor setting
LCDC_WIN0_CBR_SCL_OFF SET	0x90	W	0x0	Win0 Cbr scaling start point offset
LCDC_WIN1_SCL_FACTOR	0x94	W	0x10001000	Win1 scaling factor setting
LCDC_I2P_REF0_MST_Y	0x98	W	0x0	I2P field 0 memory start address

LCDC_I2P_REF0_MST_CBR	0x9C	W	0x0	I2P field 0 memory start address
LCDC_I2P_REF1_MST_Y	0xA0	W	0x0	I2P field 2 memory start address
LCDC_I2P_REF1_MST_CBR	0xA4	W	0x0	I2P field 2 memory start address
LCDC_WIN0_YRGB_VIR_MST	0xA8	W	0x0	Win0 virtual memory start address
LCDC_WIN0_CBR_VIR_MST	0xAC	W	0x0	Win0 virtual memory start address
LCDC_WIN1_VIR_MST	0xB0	W	0x0	Win1 virtual memory start address
LCDC_WIN0_SD_FACTOR_CBR	0xB4	W	0x10001000	Win0 CBR scaling down factor setting
LCDC_WIN0_SP_FACTOR_CBR	0xB8	W	0x10001000	Win0 CBR scaling up factor setting
LCDC_REG_CFG_DONE	0xC0	W	0x0	REGISTER CONFIG FINISH
LCDC_MCU_BYPASS_WPOR	0x500	W	0x0	MCU BYPASS MODE, DATA Write Port

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

### 9.3.2 Detail Register Description

#### LCDC\_SYS\_CONFIG

Address: Operational Base + offset (0x00)

LCDC system config register

bit	Attr	Reset Value	Description
31:29	RW	0x0	WIN0 AHB Master Burst Control: 3'b001: incr; 3'b011: burst4; 3'b101: burst8; 3'b111: burst16;  * In YCbCr422/YCbCr420-0 Rotate mode, the transfer type of Y and CbCr would be Burst-4, regardless of this setting.  * In YCbCr420-1 Rotate mode, the transfer type of Y and CBR would be Burst-8 and the transfer type of CbCr would be Burst-4, regardless of this setting.  * In I2P mode, the transfer type would be Burst-16, regardless of this setting.
28:24	RW	0x0	WIN0 AHB Master INCR number setting: Incr num = (n+1)
23:21	RW	0x0	WIN1 and HWC AHB Master Burst Control: 3'b001: incr; 3'b011: burst4; 3'b101: burst8; 3'b111: burst16;
20:16	RW	0x0	WIN1 and HWC AHB Master number setting: Incr num = (n+1)
15	RW	0x0	LCDC stand by mode: Writing "1" to turn LCDC into standby mode, All the layer would disable and the data transfer from frame buffer memory would stop at the end of current frame. The output would be blank.

			<p>When writing "0" to this bit, standby mode would disable and the LCDC go back to work immediately. 0: disable; 1: enable;</p> <p><i>* Black display is recommended before setting standby mode enable.</i></p>
14	RW	0x0	Win0 AHB Master interlace read mode, used in interlace display mode(except YCbCr420-1/YCbCr420-M)
13	RW	0x0	Win1 AHB Master interlace read mode, used in interlace display mode
12	RW	0x0	Hardware cursor data reload enable, if enable, hwc refresh at the beginning of each frame.
11	RW	0x0	hwc enable control 1: enable 0: disable
10	RW	0x0	Win0 enable control 1: enable 0: disable
9	RW	0x0	Win1 enable control 1: enable 0: disable
8	RW	0x0	<p>Win0 video rotate enable 1: Win0 Rotation 270-degrees 0: Win0 normal display</p> <p><i>*The maximum resolution is 720x1280.</i></p> <p><i>*This mode is only valid in video mode (YCbCr422/YCbCr420-0/YCbCr420-1) of win0.</i></p> <p><i>*In rotation mode, the memory start address setting (LCDC_win0_YRGB_MST/ LCDC_win0_CBR_MST) should be setting to the left point of the last line of frame data. The width/height setting of LCDC_WIN0_ACT_INFO and LCDC_DSP_WIN0_INFO should also be reversed.</i></p>
7	RW	0x0	<p>MPEG2 Interlace to progressive(De-interlace) enable: 0: disable; 1: enable;</p> <p><i>*In I2P mode, The pre and current field data(even or odd) and the next field (even or odd) data is need . the memory start address registers are LCDC_I2P_PRE_F_MST_Y/ LCDC_I2P_PRE_F_MST_CBR LCDC_WIN0_YRGB_MST/ LCDC_WIN0_CBR_MST LCDC_I2P_NXT_F_MST_Y/ LCDC_I2P_NXT_F_MST_CBR</i></p> <p><i>* You must select the field polarity of current frame by setting LCDC_DSP_CTRL0[27],</i></p> <p><i>* The threshold and filter setting see the description of I2P block</i></p>
6	RW	0x0	<p>Interlace Display Control enable: This mode is related to the ITU-R656 output, the display timing of odd field must be set correctly. (LCDC_DSP_VS_ST_END_F1/LCDC_DSP_VCT_END_F1) 0: disable; 1: enable;</p>
5	RW	0x0	<p>Win0 roller display: In roller mode, The display of win0 would turn around if the display width/height overflows the</p>

			boundary of virtual frame. The x and y value of start point must be indicated in roller mode(LCDC_WIN1_ROLLER_INFO) 0: no roller display; 1: roller display;
4	RW	0x0	Win1 roller display: In roller mode, The display of win1 would turn around if the display width/height overflows the boundary of virtual frame. 0: no roller display; 1: roller display;
3:1	RW	0x0	Win0 Display data Format 3'b000 : RGB888 3'b001 : RGB565 3'b010 : YCbCr422 3'b011 : YCbCr4200 3'b100 : YCbCr4201 3'b101 : YCbCr420M 3'b110 : YCbCr444
0	RW	0x0	Win1 Display data Format : 0: RGB888; 1: RGB565

**LCDC\_SWAP\_CTRL**

Address: Operational Base + offset (0x04)

Frame data bit swap register

bit	Attr	Reset Value	Description
31:18	-	-	Reserved
17	RW	0x0	WIN0 YRGB ahb high-low 8bit swap enable 0: ABCD 1: DBCA
16	RW	0x0	Dummy swap enable 0: B+G+R+dummy 1: dummy+B+G+R
15	RW	0x0	DISPLAY DELTA SWAP ENABLE See detail in Delta display block.
14	RW	0x0	Display output red and green swap enable 0: RGB 1: GRB
13	RW	0x0	Display output red and blue swap enable 0: RGB 1: BGR
12	RW	0x0	Display output blue and green swap enable 0: RGB 1: RBG
11	RW	0x0	WIN1 8-bit right-shift swap enable 0: ABCD 1: DABC
10	RW	0x0	WIN1 8bit swap enable 0: ABCD 1: BADC
9	RW	0x0	WIN1 16bit swap enable 0: ABCD 1: CDAB
8	RW	0x0	WIN0 Cbr 8bit swap enable 0: ABCD 1: BADC

7	RW	0x0	WIN0 Cbr 16bit swap enable 0: ABCD 1: CDAB
6	RW	0x0	WIN0 YRGB 8bit swap enable 0: ABCD 1: BADC
5	RW	0x0	WIN0 YRGB 16bit swap enable 0: ABCD 1: CDAB
4	RW	0x0	WIN0 Cbr 8-bit right-shift swap enable 0: ABCD 1: DABC
3	RW	0x0	WIN0 YRGB 8-bit right-shift swap enable 0: ABCD 1: DABC
2	RW	0x0	WIN0 YRGB middle 8-bit swap enable 0: ABCD 1: ACBD
1	RW	0x0	WIN0 RGB565,Red and blue swap enable 0: R1G1B1R0G0B0 1: B1G1R1B0G0R0
0	RW	0x0	WIN1 RGB565,Red and Blue swap enable 0: R1G1B1R0G0B0 1: B1G1R1B0G0R0

**LCDC\_MCU\_TIMING\_CTRL**

Address: Operational Base + offset (0x08)

MCU LCD interface control register

bit	Attr	Reset Value	Description
31	RW	0x0	MCU LCD output SELECT
30	RW	0x0	MCU LCD BYPASS MODE Select
29	RW	0x0	MCU LCD RS Select
28	RW	0x0	MCU HOLD Mode Frame Start
27	RW	0x0	MCU HOLD Mode Select
26	R	0x0	MCU HOLD status
25	-	-	Reserved
24:20	RW	0x0	MCU_RW signal end point
19:15	RW	0x0	MCU_RW signal start point
14:10	RW	0x0	MCU_CS signal end point
9:5	RW	0x0	MCU_CS signal start point
4:0	RW	0x0	MCU LCD Interface writing period

**LCDC\_BLEND\_CTRL**

Address: Operational Base + offset (0x0C)

LCDC alpha blending control register

bit	Attr	Reset Value	Description
31:24	RW	0x0	WIN0 alpha blending factor
23:16	RW	0x0	WIN1 alpha blending factor
15:12	RW	0x0	HWC alpha blending factor
11:6	RW	-	reserved
5	RW	0x0	Win0 and Win1 overlay setting: 0: win1 on the top; 1: win0 on the top;
4	RW	0x0	WIN0 alpha blending factor selection: 0: alpha from register; 1: alpha in pixel data(ARGB888 mode);

3	RW	0x0	WIN1 alpha blending factor selection: 0: alpha from register; 1: alpha in pixel data(ARGB888 mode);
2	RW	0x0	HWC alpha blending enable: 0: disable; 1: enable;
1	RW	0x0	WIN0 alpha blending enable: 0: disable; 1: enable;
0	RW	0x0	WIN1 alpha blending enable: 0: disable; 1: enable;

**LCDC\_WIN0\_COLOR\_KEY\_CTRL**

Address: Operational Base + offset (0x10)

LCDC win0 color key control register

bit	Attr	Reset Value	Description
31:25	RW	0x0	reserved
24	RW	0x0	Win0 transparency color key enable: 0: disable; 1: enable;
23:16	RW	0x0	Win0 key color red[7:0];
15:8	RW	0x0	Win0 key color green[7:0];
7:0	RW	0x0	Win0 key color blue[7:0];

**LCDC\_WIN1\_COLOR\_KEY\_CTRL**

Address: Operational Base + offset (0x14)

LCDC win1 color key control register

bit	Attr	Reset Value	Description
31:25	RW	0x0	reserved
24	RW	0x0	Win1 transparency color key enable: 0: disable; 1: enable;
23:16	RW	0x0	Win1key color red[7:0];
15:8	RW	0x0	Win1 key color green[7:0];
7:0	RW	0x0	Win1 key color blue[7:0];

**LCDC\_DEFLICKER\_SCL\_OFFSET**

Address: Operational Base + offset (0x18)

LCDC scaling start point offset register for De-flicker

bit	Attr	Reset Value	Description
31:24	RW	0x0	Win1_vsp_offset: Vertical scaling up start point offset (0x00~0xff)/0x100 = 0~0.99
23:16	RW	0x0	Win1_vsd_offset Vertical scaling down start point offset (0x00~0xff)/0x100 = 0~0.99
15:8	RW	0x0	Win0_YRGB_vsp_offset Vertical scaling up start point offset (0x00~0xff)/0x100 = 0~0.99
7:0	RW	0x0	Win0_YRGB_vsd_offset Vertical scaling down start point offset (0x00~0xff)/0x100 = 0~0.99

**LCDC\_DSP\_CTRL\_REG0**

Address: Operational Base + offset (0x1C)

LCDC display control register 0



bit	Attr	Reset Value	Description
31	RW	0x0	I2p filter parameter choose: 0: #define K0 1 #define K1 2 #define K2 1 #define Kdis 2 #define Ks 2  1: #define K0 1 #define K1 6 #define K2 1 #define Kdis 4 #define Ks 3
30	RW	0x0	I2p filter enable: 0: filter disable 1: filter enable
29	RW	0x0	YCbCr clip mode 0: disable, YCbCr no clip 1: enable, YCbCr clip before YCbCr2RGB  <i>*Y clip: 16~235, CbCr clip: 16~239</i>
28	RW	0x0	Interlace field polarity Select the start field polarity of the interlace output. 0: Odd field 1: Even field  <i>*This bit is only valid in interlace mode</i>
27	RW	0x0	I2p current field polarity Select the current field of the interlace field data. So the other field would be generated (I2P algorithm) for one whole frame. 0: Odd field 1: Even field  <i>*This bit is only valid in De-interlace mode</i>
26	-	-	Reserved.
25	-	-	Reserved.
24	RW	0x0	Dithering enable: 0: disable; 1: enable;
23	RW	0x0	Dithering mode 0: RGB888 to RGB565 1: RGB888 to RGB666
22	RW	0x0	RGB565 to RGB888 Replication enable: 0: disable; 1: enable;  <i>*Replication would be valid only if win0 or win1's mode is RGB565, otherwise, this function would be ignored.</i>
21:16	RW	0x4	I2P threshold Cbr: CBR threshold for the judgment of static image or motion image 0x0~0xf
15:10	RW	0x4	I2P threshold Y: Y threshold for the judgment of static image or motion image 0x0~0xf
9:8	RW	0x00	Color space conversion:

			2'b00: YCbCr2RGB mpeg(ITU601) 2'b01: YCbCr2RGB jpeg(ITU601) 2'b10: YCbCr2RGB hdtv(REC709) 2'b11: YCbCr BYPASS  <i>*YCbCr bypass mode is used for ITU-656 output. If the pixel data of win0/win1 is RGB format, RGB2YCbCr conversion is active.</i>
7	RW	0x0	DSP OUTPUT PIN dclk polarity invert 0: normal; 1: invert;
6	RW	0x0	DSP OUTPUT PIN den polarity invert 0: positive; 1: negative;
5	RW	0x0	DSP OUTPUT PIN VSYNC polarity invert 0: negative; 1: positive;
4	RW	0x0	DSP OUTPUT PIN HSYNC polarity invert 0: negative; 1: positive;
3:0	RW	0x0	Display output format:  4'b0000: Parallel 24-bit RGB888 output {R[7:0],G[7:0],B[7:0]}  4'b0001: Parallel 16-bit RGB666 output {6'b0,R[5:0],G[5:0],B[5:0]}  4'b0010: Parallel 15-bit RGB565 output {8'b0,R[4:0],G[5:0],B[4:0]}  4'b0100: Serial 2x16-bit RGB888x {8'b0,G[7:0],B[7:0]} + {16'b0,R[7:0]}  4'b0110: ITU-656 output {16'b0,pixel_data[7:0]}  4'b1000: Serial 3x8-bit RGB888 {16'b0, B[7:0]}+{16'b0,G[7:0]}+{16'b0,R[7:0]}  4'b1100: Serial 3x8-bit RGB888 + dummy {16'b0, B[7:0]}+{16'b0,G[7:0]}+{16'b0,R[7:0]} + dummy  Others: Reserved.

**LCDC\_DSP\_CTRL\_REG1**

Address: Operational Base + offset (0x20)

LCDC display control register 1

bit	Attr	Reset Value	Description
31	RW	0x0	Win0 YRGB scaling up offset(De-flicker) enable: 0: disable; 1: enable;
30	RW	0x0	Win0 YRGB scaling down offset(De-flicker) enable: 0: disable; 1: enable;
29	RW	0x0	Win0 Cbr scaling up offset(De-flicker) enable: 0: disable; 1: enable;

28	RW	0x0	Win0 Cbr scaling down offset(De-flicker) enable: 0: disable; 1: enable;
27	RW	0x0	Win1 scaling up offset(De-flicker) enable: 0: disable; 1: enable;
26	RW	0x0	Win1 scaling down offset(De-flicker) enable: 0: disable; 1: enable;
25	RW	0x0	Black display mode. When this bit enable, the pixel data output is all black (0x000000)
24	RW	0x1	Blank display mode. When this bit enable, the hsync/vsync/den output is blank
23:16	RW	0x00	Background Red color
15:8	RW	0x00	Background Green color
7:0	RW	0x00	Background Blue color

**LCDC\_INT\_STATUS**

Address: Operational Base + offset (0x24)

LCDC interrupt status register

bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19:9	RW	0x0	Line number of the scanning flag interrupt The display line number when the flag interrupt occur, the range is (0~ DSP_VTOTAL-1).
8	W	0x0	Scanning flag Interrupt clear: After be set to 1, this bit will clear by itself 1 cycle later.
7	W	0x0	Frame start interrupt clear: After be set to 1, this bit will clear by itself 1 cycle later.
6	W	0x0	Horizontal start interrupt clear: After be set to 1, this bit will clear by itself 1 cycle later.
5	RW	0x1	Scanning flag Interrupt Mask: 0: unmask; 1: mask;
4	RW	0x1	Frame start interrupt mask: 0: unmask; 1: mask;
3	RW	0x1	Horizontal start interrupt mask: 0: unmask; 1: mask;
2	R	0x0	Scanning flag Interrupt status
1	R	0x0	Frame start interrupt status
0	R	0x0	Horizontal start interrupt status

**LCDC\_WINO\_VIR**

Address: Operational Base + offset (0x28)

LCDC win0 virtual width/height registers for Virtual display

bit	Attr	Reset Value	Description
31:16	RW	0x0	Win0 Virtual Display Height The max height of win0 Virtual display
15:0	RW	0x0	Win0 Virtual Display Width The max width of win0 Virtual display

**LCDC\_WINO\_YRGB\_MST**

Address: Operational Base + offset (0x2C)

LCDC win0 YRGB active frame buffer memory start address

bit	Attr	Reset Value	Description
31:0	RW	0x0000	win0 YRGB frame buffer memory start address

#### LCDC\_WINO\_CBR\_MST

Address: Operational Base + offset (0x30)

LCDC win0 CBR active frame buffer memory start address

bit	Attr	Reset Value	Description
31:0	RW	0x0000	win0 CBR frame buffer memory start address

#### LCDC\_WINO\_ACT\_INFO

Address: Operational Base + offset (0x34)

LCDC win0 active display width/height

bit	Attr	Reset Value	Description
31:16	W/R	0x0000	Win0 active(original) window height
15:0	W/R	0x0000	Win0 active(original) window width

#### LCDC\_WINO\_ROLLER\_INFO

Address: Operational Base + offset (0x38)

LCDC win0 roller display start point

bit	Attr	Reset Value	Description
31:16	W/R	0x00	Win0 start point row-number(y) on virtual frame
15:0	W/R	0x00	Win0 start point column-number(x) on virtual frame

#### LCDC\_WINO\_DSP\_ST

Address: Operational Base + offset (0x3C)

LCDC win0 display start point

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	W/R	0x00	Win0 vertical start point(y) of the Panel scanning
15:11	-	-	Reserved.
10:0	W/R	0x00	Win0 horizontal start point(x) of the Panel scanning

#### LCDC\_WINO\_DSP\_INFO

Address: Operational Base + offset (0x40)

LCDC win0 display width/height

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	W/R	0x00	Win0 display height on the Panel
15:11	-	-	Reserved.
10:0	W/R	0x00	Win0 display width on the Panel.

#### LCDC\_WIN1\_VIR

Address: Operational Base + offset (0x44)

LCDC win1 virtual width/height register for Virtual display

bit	Attr	Reset Value	Description
31:16	RW	0x0	Win1 Virtual Display Height The max height of win1 Virtual display
15:0	RW	0x0	Win1 Virtual Display Width The max width of win1 Virtual display

#### LCDC\_WIN1\_YRGB\_MST

Address: Operational Base + offset (0x48)

LCDC win1 active frame buffer memory start address

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Win1 RGB frame buffer memory start address

**LCDC\_WIN1\_ACT\_INFO**

Address: Operational Base + offset (0x4C)

LCDC win1 active display width/height

bit	Attr	Reset Value	Description
31:16	W/R	0x0000	Win1 active(original) window height
15:0	W/R	0x0000	Win1 active(original) window width

**LCDC\_WIN1\_ROLLER\_INFO**

Address: Operational Base + offset (0x50)

LCDC win1 roller display start point

bit	Attr	Reset Value	Description
31:16	W/R	0x00	Win1 start point row-number(y) on virtual frame
15:0	W/R	0x00	Win1 start point column-number(x) on virtual frame

**LCDC\_WIN1\_DSP\_ST**

Address: Operational Base + offset (0x54)

LCDC win1 display start point

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	W/R	0x00	Win1 vertical start point(y) of the Panel scanning
15:11	-	-	Reserved.
10:0	W/R	0x00	Win1 horizontal start point(x) of the Panel scanning

**LCDC\_WIN1\_DSP\_INFO**

Address: Operational Base + offset (0x58)

LCDC win1 display width/height

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	W/R	0x00	Win1 display height on the Panel
15:11	-	-	Reserved.
10:0	W/R	0x00	Win1 display width on the Panel.

**LCDC\_HWC\_MST**

Address: Operational Base + offset (0x5C)

LCDC HWC data memory start address

bit	Attr	Reset Value	Description
31:0	RW	0x0000	HWC data memory start address

**LCDC\_HWC\_DSP\_ST**

Address: Operational Base + offset (0x60)

LCDC HWC display start point

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x00	HWC vertical start point(y) of the Panel scanning
15:11	-	-	Reserved.
10:0	RW	0x00	HWC horizontal start point(x) of the Panel scanning

**LCDC\_HWC\_COLOR\_LUTO**

Address: Operational Base + offset (0x64)

LCDC HWC color 0 look-up table

bit	Attr	Reset Value	Description
31:24	RW	0x00	Win0 YRGB_hsp_offset Horizontal scaling up start point offset (0x00~0xff)/0x100 = 0~0.99
23:16	RW	0x00	Hardware cursor color red for 2'b01

15:8	RW	0x00	Hardware cursor color green for 2'b01
7:0	RW	0x00	Hardware cursor color blue for 2'b01

**LCDC\_HWC\_COLOR\_LUT1**

Address: Operational Base + offset (0x68)

LCDC HWC color 1 look-up table

bit	Attr	Reset Value	Description
31:24	RW	0x00	<i>Win0 YRGB_hsd_offset</i> Horizontal scaling down start point offset (0x00~0xff)/0x100 = 0~0.99
23:16	RW	0x00	Hardware cursor color red for 2'b10
15:8	RW	0x00	Hardware cursor color green for 2'b10
7:0	RW	0x00	Hardware cursor color blue for 2'b10

**LCDC\_HWC\_COLOR\_LUT2**

Address: Operational Base + offset (0x6C)

LCDC HWC color 2 look-up table

bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:16	RW	0x00	Hardware cursor color red for 2'b11
15:8	RW	0x00	Hardware cursor color green for 2'b11
7:0	RW	0x00	Hardware cursor color blue for 2'b11

**LCDC\_DSP\_HTOTAL\_HS\_END**

Address: Operational Base + offset (0x70)

LCDC Panel scanning timing register (Horizontal period, hsync pulse width)

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x17f	Panel display scanning horizontal period.
15:11	-	-	Reserved.
10:0	RW	0x17	Panel display scanning hsync pulse width.

**LCDC\_DSP\_HACT\_ST\_END**

Address: Operational Base + offset (0x74)

LCDC Panel scanning timing register (horizontal active start/end point)

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x2B	Panel display scanning horizontal active start point
15:11	-	-	Reserved.
10:0	RW	0x16A	Panel display scanning horizontal active end point

**LCDC\_DSP\_VTOTAL\_VS\_END**

Address: Operational Base + offset(0x78)

LCDC Panel scanning timing register (Vertical period, vsync pulse width)

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x107	Panel display scanning vertical period.
15:11	-	-	Reserved.
10:0	RW	0x002	Panel display scanning vsync pulse width.

**LCDC\_DSP\_VACT\_ST\_END**

Address: Operational Base + offset (0x7C)

LCDC Panel scanning timing register (Vertical active start/end point)

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x00D	Panel display scanning vertical active start point

15:11	-	-	Reserved.
10:0	RW	0x0FC	Panel display scanning vertical active end point

**LCDC\_DSP\_VS\_ST\_END\_F1**

Address: Operational Base + offset (0x80)

LCDC Panel scanning timing register (Vertical vsync pulse start and end of interlace mode)

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x00D	Panel display scanning vertical vsync start point of 2nd field (interlace display mode)
15:11	-	-	Reserved.
10:0	RW	0x0FC	Panel display scanning vertical vsync end point of 2nd field(interlace display mode)

**LCDC\_DSP\_VACT\_ST\_END\_F1**

Address: Operational Base + offset (0x84)

LCDC Panel scanning timing register (Vertical active start and end of interlace mode)

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x00D	Panel display scanning vertical active start point of 2nd field (interlace display mode)
15:11	-	-	Reserved.
10:0	RW	0x0FC	Panel display scanning vertical active end point of 2nd field (interlace display mode)

**LCDC\_WIN0\_SD\_FACTOR\_Y**

Address: Operational Base + offset (0x88)

LCDC win0 scaling down factor of Y or RGB

bit	Attr	Reset Value	Description
31:16	RW	0x1000	Win0 YRGB vertical scaling down factor: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$
15:0	RW	0x1000	Win0 YRGB horizontal scaling down factor: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$

**LCDC\_WIN0\_SP\_FACTOR\_Y**

Address: Operational Base + offset (0x8C)

LCDC win0 scaling up factor of Y or RGB

bit	Attr	Reset Value	Description
31:16	RW	0x1000	Win0 YRGB vertical scaling up factor: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$
15:0	RW	0x1000	Win0 YRGB horizontal scaling up factor: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$

**LCDC\_WIN0\_CBR\_SCL\_OFFSET**

Address: Operational Base + offset (0x90)

LCDC win0 CBR scaling start point offset

bit	Attr	Reset Value	Description
31:24	RW	0x00	Win0 Cbr_vsp_offset Vertical scaling up start point offset (0x00~0xff)/0x100 = 0~0.99



23:16	RW	0x00	Win0 Cbr_vsd_offset Vertical scaling down start point offset (0x00~0xff)/0x100 = 0~0.99
15:8	RW	0x00	Win0 Cbr_hsp_offset Horizontal scaling up start point offset (0x00~0xff)/0x100 = 0~0.99
7:0	RW	0x00	Win0 Cbr_hsd_offset Horizontal scaling down start point offset (0x00~0xff)/0x100 = 0~0.99

**LCDC\_WIN1\_SCL\_FACTOR**

Address: Operational Base + offset(0x94)

LCDC win1 scaling up/down factor

bit	Attr	Reset Value	Description
31:16	RW	0x1000	Win1 vertical scaling up/down factor: $factor = \left( \frac{LCDC\_WIN1\_ACT\_INFO[31:16]}{LCDC\_WIN1\_DSP\_INFO[31:16]} \right) \times 2^{12}$
15:0	RW	0x1000	Win1 horizontal scaling up/down factor: $factor = \left( \frac{LCDC\_WIN1\_ACT\_INFO[15:0]}{LCDC\_WIN1\_DSP\_INFO[15:0]} \right) \times 2^{12}$

**LCDC\_I2P\_REFO\_MST\_Y**

Address: Operational Base + offset (0x98)

LCDC I2P pre field Ydata start address in Memory

bit	Attr	Reset Value	Description
31:0	W/R	0x00	I2P pre field Y data Start Point in Memory

**LCDC\_I2P\_REF1\_MST\_CBR**

Address: Operational Base + offset (0x9C)

LCDC I2P pre field CBR data start address in Memory

bit	Attr	Reset Value	Description
31:0	W/R	0x00	I2P pre field CBR data Start Point in Memory

**LCDC\_I2P\_REF1\_MST\_Y**

Address: Operational Base + offset (0xA0)

LCDC I2P next field Ydata start address in Memory

bit	Attr	Reset Value	Description
31:0	W/R	0x00	I2P next field Y data Start Point in Memory

**LCDC\_I2P\_REF1\_MST\_CBR**

Address: Operational Base + offset (0xA4)

LCDC I2P next field CBR data start address in Memory

bit	Attr	Reset Value	Description
31:0	W/R	0x00	I2P next field CBR data Start Point in Memory

**LCDC\_WINO\_YRGB\_VIR\_MST**

Address: Operational Base + offset (0xA8)

LCDC win0 YRGB virtual frame buffer memory start address

bit	Attr	Reset Value	Description
31:0	W/R	0x00	win0 virtual frame buffer memory start address

**LCDC\_WINO\_CBR\_VIR\_MST**

Address: Operational Base + offset (0xAC)

LCDC win0 CBR virtual frame buffer memory start address

bit	Attr	Reset Value	Description
31:0	W/R	0x00	win0 virtual frame buffer memory start address

**LCDC\_WIN1\_VIR\_MST**

Address: Operational Base + offset (0xB0)

LCDC win1 virtual frame buffer memory start address

bit	Attr	Reset Value	Description
31:0	W/R	0x00	Win1 virtual frame buffer memory start address

**LCDC\_WIN0\_SD\_FACTOR\_CBR**

Address: Operational Base + offset (0xB4)

LCDC win0 scaling down factor of CBR

bit	Attr	Reset Value	Description
31:16	RW	0x1000	Win0 CBR vertical scaling down factor: YCbCr420: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]/2}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$ YCbCr422,YCbCr444: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$
15:0	RW	0x1000	Win0 CBR horizontal scaling down factor: YCbCr422,YCbCr420: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]/2}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$ YCbCr444: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$

**LCDC\_WIN0\_SP\_FACTOR\_CBR**

Address: Operational Base + offset (0xB8)

LCDC win0 scaling up factor of CBR

bit	Attr	Reset Value	Description
31:16	RW	0x1000	Win0 CBR vertical scaling up factor: YCbCr420: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]/2}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$ YCbCr422,YCbCr444: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$
15:0	RW	0x1000	Win0 CBR horizontal scaling up factor: YCbCr422,YCbCr420: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]/2}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$ YCbCr444: $factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$

**LCDC\_REG\_CFG\_DONE**

Address: Operational Base + offset (0xC0)

LCDC register config done flag

bit	Attr	Reset Value	Description
31:1	-	-	Reserved.

0	W	0x0	In the first setting of the register, the new value was saved into the mirror register. When all the register config finish, writing this register to enable the copyright of the mirror register to real register. Then register would be updated at the start of every frame.
---	---	-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**LCDC\_MCU\_BYPASS\_WPORT**

Address: Operational Base + offset(0x500)

LCDC MCU bypass mode data write port

bit	Attr	Reset Value	Description
31:0	W	0x0	When MCU is in BYPASS Mode, BYPASS data is written through this Port

**9.4 Functional Description****9.4.1 Memory data formats**

LCDC master read the frame data from the frame buffer in the system memory (SDR or DDR). There are total 9 formats supported in three layers.

- Graphic layer(win1): RGB888, ARGB888, RGB565,
- Video layer(win0): RGB888, ARGB888, RGB565, YCbCr422, YCbCr420-0, YCbCr420-1, YCbCr420-M, YCbCr444
- Hardware cursor (hwc): 2-bit LUT-colors

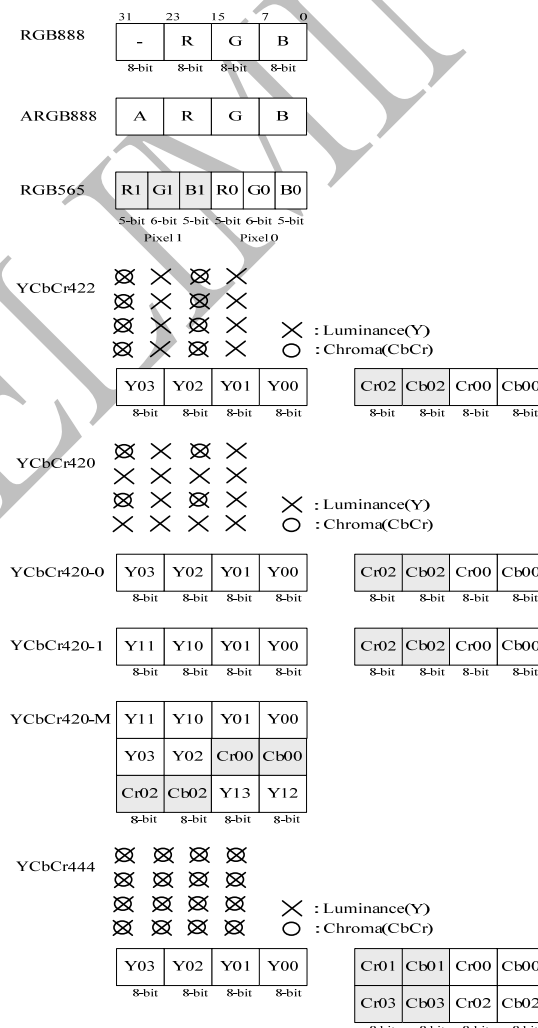


Fig. 9-2 RK281x LCDC Frame buffer Data Format

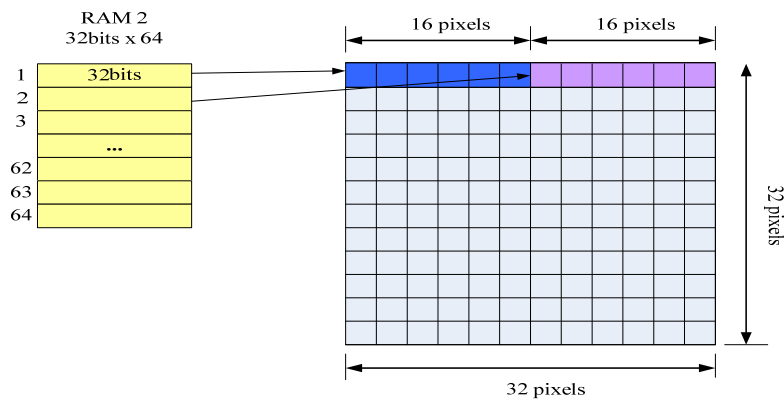


Fig. 9-3 RK281x LCDc Hwc Data Format

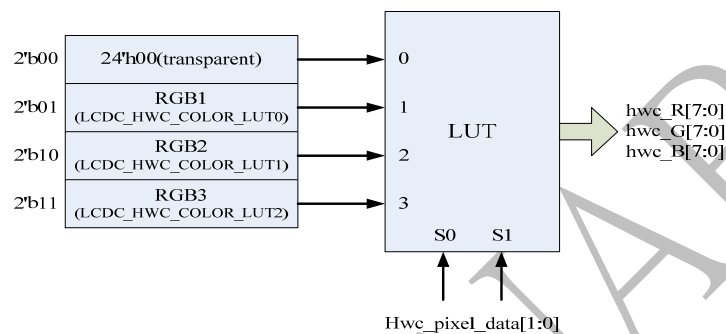


Fig. 9-4 RK281x LCDc Hwc colors Look-up table

### Data SWAP function

There are several data-swap modes for flexible application. Those registers are LCDC\_SWAP\_CTRL[11:0] and LCDC\_SWAP\_CTRL[17].

For RGB data, there are 6 types of data-swap for WIN0 and 5 types for WIN1:

Table. 8-1 LCDc RGB/Y Data swap of WIN0 and WIN1

Data-swap (RGB/Y)	8bit swap	16bit swap	middle 8-bit swap	right-shift 8-bit swap	RB swap	high-low 8-bit swap
WIN0	yes	yes	yes	yes	yes	yes
WIN1	yes	yes	yes	yes	yes	No

Table. 8-2 LCDc CbCr Data swap of WIN0

Data-swap (CbCr)	8bit swap	16bit swap	right-shift 8-bit swap
WIN0	yes	yes	yes

## 9.4.2 Virtual display

### 1. Normal Mode

Virtual display is supported in WIN0 and WIN1. The active frame is part of the virtual (original) frame in frame buffer memory. First, Vir\_width/Vir\_height and the memory start of virtual (Vir\_mst) frame must be set to LCDC\_WIN0\_VIR/LCDC\_WIN1\_VIR, LCDC\_WIN0\_YRGB\_VIR\_MST/LCDC\_WIN0\_CBR\_VIR\_MST/LCDC\_WIN1\_VIR\_MST.

The display frame on the panel is the active frame, whose Act\_width/ Act\_height and Act\_mst is indicated in LCDC\_WIN0\_ACT\_INFO/LCDC\_WIN1\_ACT\_INFO,

LCDC\_WIN0\_YRGB\_ACT\_MST/LCDC\_WIN0\_CBR\_ACT\_MST/LCDC\_WIN1 \_ACT\_MST.

The register setting of Vir\_width/Vir\_height and Act\_width/Act\_height should be multiples of 2 when the data format is RGB565 or YCbCr420-1, It should be multiples of 4 when the data format is YCbCr422, YCbCr420-0, YCbCr420-M or YCbCr444.

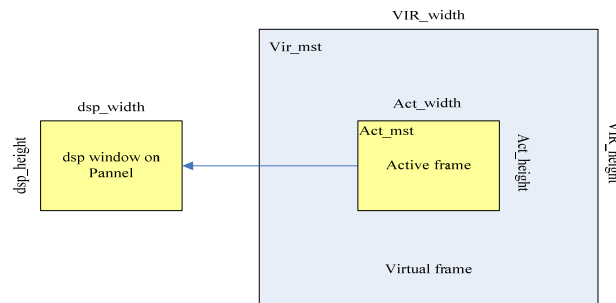
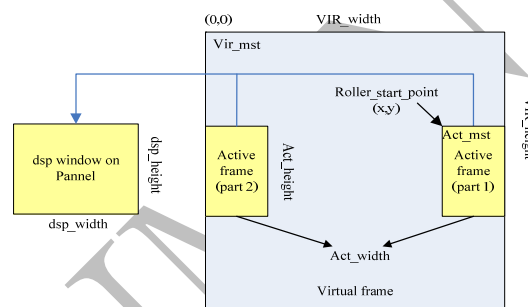


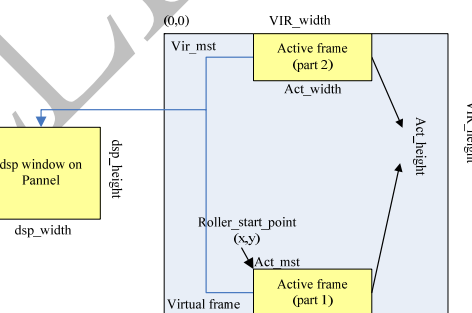
Fig. 9-5 RK281x LCDC Virtual display mode

## 2. Roller Mode

In normal virtual mode, the boundary of active frame should not overflow the virtual frame. In roller mode, the area of the overflow frame will continue with the other side of virtual frame.



a. horizontal roller mode



b. Vertical roller mode

Fig. 9-6 RK281x LCDC Virtual display roller mode

### Configuration flow:

1. Normal display configuration.
2. Assert roller enable bit in LCDC\_SYS\_CONFIG register.
3. Configure LCDC\_WIN0\_ROLLER\_INFO/LCDC\_WIN1\_ROLLER\_INFO register to set the active window's starting location in the virtual window.
4. WIN0/1\_VIR\_MST configuration is needed when vertical roller occurs.
5. Config done.

\*Note: roller\_x must be less than win0/1\_act\_width, roller\_y must be less than win0/1\_act\_height.

### 9.4.3 Rotation

In rotation mode, the image rotates 270 degrees clockwise.

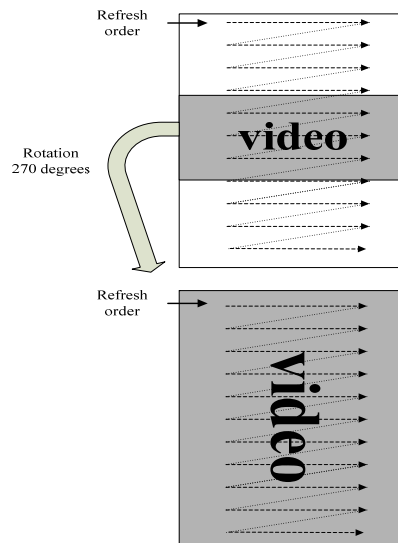


Fig. 9-7 RK281x LCDC Rotation

#### Configuration flow:

1. Assert rotate enable bit in LCDC\_SYS\_CONFIG register.
2. Exchange the active width and height setting value of the active image.
3. Configure LCDC\_WIN0\_MST register. In rotate mode, the calculation of LCDC\_WIN0\_MST is as follow:

(1) Y memory start address of YCbCr422/YCbCr444/YCbCr4200:

$$LCDC\_WIN0\_MST\_Y = WIN0\_MST\_Y\_ori + win0\_vir\_width * (win0\_act\_width - 1) \quad (2)$$

Y memory start address of YCbCr4201:

$$LCDC\_WIN0\_MST\_Y = WIN0\_MST\_Y\_ori + win0\_vir\_width * 2 * \left( \frac{win0\_act\_width}{2} - 1 \right)$$

(3) Cb/Cr memory start address of YCbCr422:

$$LCDC\_WIN0\_MST\_Cbr = WIN0\_MST\_Cbr\_ori + win0\_vir\_width * (win0\_act\_width - 1)$$

(4) Cb/Cr memory start address of YCbCr4200/YCbCr4201:

$$LCDC\_WIN0\_MST\_Cbr = WIN0\_MST\_Cbr\_ori + win0\_vir\_width * \left( \frac{win0\_act\_width}{2} - 1 \right)$$

4. Config done.

\*Note:

1. In the equation, WIN0\_MST\_ori means the WIN0\_MST before rotation; win0\_vir\_width means a virtual win0 line's address length.

2. when source format is YCbCr422, the scaling factors should be configured according YCbCr444.

### 9.4.4 De-interlace

De-interlace is a video process method dealing with the grid problem when displaying interlace video on progressive panel.

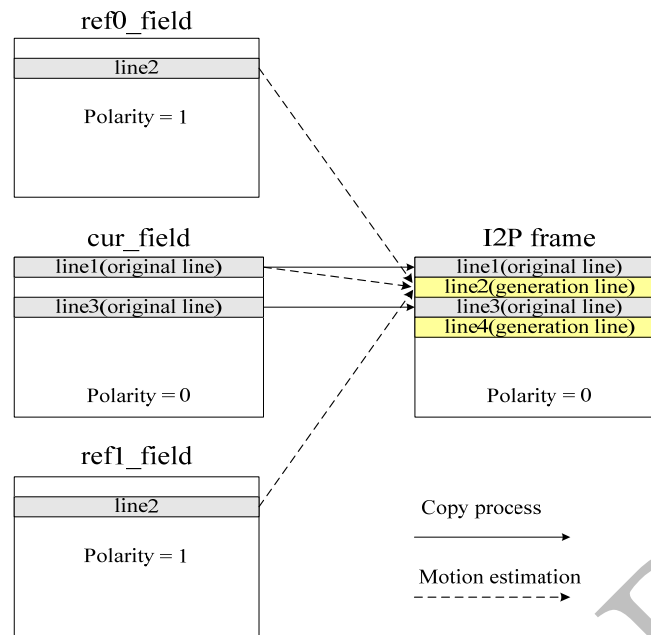


Fig. 9-8 RK281x LCD De-interlace

**Configuration flow:**

1. Assert De-interlace enable bit in LCDDC\_SYS\_CONFIG register.
2. Configure current field's polarity in LCDDC\_DSP\_CTRL\_REG0 register.
3. Configure LCDDC\_WIN0\_MST、LCDDC\_I2P\_REF0\_MST、LCDDC\_I2P\_REF1\_MST registers.  
In De-interlace mode, the calculations are as follow:

## A. i2p\_polarity is 0:

$$\text{LCDDC\_WIN0\_MST} = \text{ref0\_mst}$$

$$\text{LCDDC\_REF0\_MST} = \text{ref0\_mst} + \text{win0\_vir\_width}$$

$$\text{LCDDC\_REF1\_MST} = \text{ref1\_mst} + \text{win0\_vir\_width}$$

## B. i2p\_polarity is 1:

$$\text{LCDDC\_WIN0\_MST} = \text{ref0\_mst} + \text{win0\_vir\_width}$$

$$\text{LCDDC\_REF0\_MST} = \text{ref0\_mst}$$

$$\text{LCDDC\_REF1\_MST} = \text{ref1\_mst}$$

In De-interlace and rotate mode, the calculations are as follow:

## A. i2p\_polarity is 0:

$$\text{LCDDC\_WIN0\_MST} = \text{ref0\_mst} + \text{win0\_vir\_width} * (\text{win0\_act\_width} - 2)$$

$$\text{LCDDC\_REF0\_MST} = \text{ref0\_mst} + \text{win0\_vir\_width} * (\text{win0\_act\_width} - 1)$$

$$\text{LCDDC\_REF1\_MST} = \text{ref1\_mst} + \text{win0\_vir\_width} * (\text{win0\_act\_width} - 1)$$

## B. i2p\_polarity is 1:

$$\text{LCDDC\_WIN0\_MST} = \text{ref0\_mst} + \text{win0\_vir\_width} * (\text{win0\_act\_width} - 1)$$

$$\text{LCDDC\_REF0\_MST} = \text{ref0\_mst} + \text{win0\_vir\_width} * (\text{win0\_act\_width} - 2)$$

$$\text{LCDDC\_REF1\_MST} = \text{ref1\_mst} + \text{win0\_vir\_width} * (\text{win0\_act\_width} - 2)$$

4. Configure i2p filter switch in LCDDC\_DSP\_CTRL\_REG0 register when filter is needed.  
Filter parameters could be selected by asserting i2p\_para\_sel bit in



LCDC\_DSP\_CTRL\_REG0 register.

5. Config done.

*\*Note: In the above equations, mst\_ref0 is reference field0's memory start address; mst\_ref1 is reference field1's memory start address; win0\_vir\_width means a virtual win0 line's address length.*

### 9.4.5 Scaling

The scaling operation is the image resizing process of data transfer from the frame buffer memory to LCD panel or TV set. There are two scaling units: Scaling down and scaling up.

If the display frame's width or height is smaller than the active frame's width or height. Scaling down operation is done by setting the scaling down factor (vertical/horizontal). Similarly, If the display frame's width or height is greater than the active frame's width or height. Scaling up operation is done by setting the scaling up factor (vertical/horizontal).

The virtual scaling and horizontal scaling, scaling down and scaling up units are independent, so neither of them, only one, or both can be used simultaneously (except the scaling down and scaling up operation of Win1).

The scaling units in layer Graphic (WIN1) and layer Video (WIN0) are also independent. So WIN0 and WIN1's scaling can be enabled simultaneously.

#### 1. Scaling factor

Because the chroma data may have different sampling rate with Luma data in the memory format of YCbCr422/YCbCr420-0/YCbCr420-1/YCbCr420-M. The scaling factor of WIN0 has two couples of factor registers:

LCDC\_WIN0\_SD\_FACTOR\_Y/ LCDC\_WIN0\_SP\_FACTOR\_Y  
LCDC\_WIN0\_SD\_FACTOR\_CBR/ LCDC\_WIN0\_SP\_FACTOR\_CBR

The software calculates the scaling up/down factor value using the following equations:

For WIN0,

$$y\_rgb\_vertical\_factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$$

$$y\_rgb\_horizontal\_factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$$

$$yuv422\_yuv444\_Cbr\_vertical\_factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$$

$$yuv420\_Cbr\_vertical\_factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[31:16]/2}{LCDC\_WIN0\_DSP\_INFO[31:16]} \right) \times 2^{12}$$

$$yuv444\_Cbr\_horizontal\_factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$$

$$yuv422\_yuv420\_Cbr\_horizontal\_factor = \left( \frac{LCDC\_WIN0\_ACT\_INFO[15:0]/2}{LCDC\_WIN0\_DSP\_INFO[15:0]} \right) \times 2^{12}$$

For WIN1,

$$vertical\_factor = \left( \frac{LCDC\_WIN1\_ACT\_INFO[31:16]}{LCDC\_WIN1\_DSP\_INFO[31:16]} \right) \times 2^{12}$$

$$horizontal\_factor = \left( \frac{LCD\_WIN1\_ACT\_INFO[15:0]}{LCD\_WIN1\_DSP\_INFO[15:0]} \right) \times 2^{12}$$

Win1 only supports RGB data format, there are only one scaling down register and one scaling up factor register (LCD\\_WIN1\\_SD\\_FACTOR and LCD\\_WIN1\\_SP\\_FACTOR).

## 2. Scaling start point offset

The x and y start point of the generated pixels can be offset, the offset value can be from 0 to 0.99.

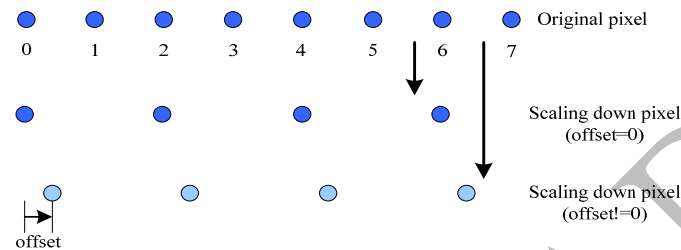


Fig. 9-9 RK281x LCD scaling down offset

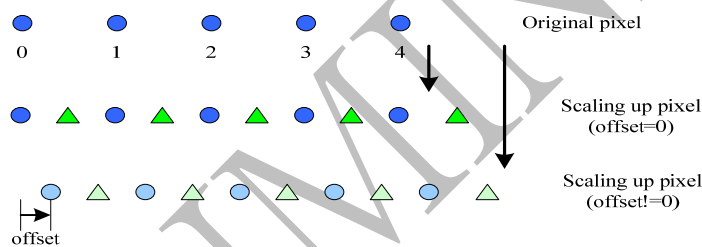


Fig. 9-10 RK281x LCD scaling up offset

Table. 8-3 LCD scaling down/up start point offset registers

<b>scaling down/up start point offset</b>	<b>Offset variable</b>	<b>Register</b>
Win1 vertical scaling up offset	Win1_vsp_offset	LDC_DEFLICKER_SCL_OFFSET[31:24]
Win1 vertical scaling down offset	Win1_vsd_offset	LDC_DEFLICKER_SCL_OFFSET[23:16]
Win0 YRGB vertical scaling up offset	Win0_YRGB_vsp_offset	LDC_DEFLICKER_SCL_OFFSET[15:8]
Win0 YRGB vertical scaling down offset	Win0_YRGB_vsd_offset	LDC_DEFLICKER_SCL_OFFSET[7:0]
Win0 YRGB horizontal scaling up offset	Win0_YRGB_hsp_offset	HWC_LUT0 [31:24]
Win0 YRGB horizontal scaling down offset	Win0_YRGB_hsd_offset	HWC_LUT1 [31:24]
Win0 Cbr vertical scaling up offset	Win0_CBR_vsp_offset	LDC_WIN0_CBR_SCL_OFFSET[31:24]
Win0 Cbr vertical scaling down offset	Win0_CBR_vsd_offset	LDC_WIN0_CBR_SCL_OFFSET[23:16]
Win0 Cbr horizontal scaling up offset	Win0_CBR_hsp_offset	LDC_WIN0_CBR_SCL_OFFSET[15:8]
<b>Win0 Cbr horizontal scaling down offset</b>	<b>Win0_CBR_hsd_offset</b>	<b>LDC_WIN0_CBR_SCL_OFFSET[7:0]</b>

\*There is only vertical scaling offset for WIN1.

### 3. De-flicker (Interlace vertical filtering)

It is necessary to display a non-interlaced video signal on an interlaced display (such as TV set). Thus some form of "non-interlaced-to-interlaced conversion" may be required.

The easiest approach is to throw away every other active scan line in each non-interlaced frame. Although the cost is minimal, there are problems with this approach. If there is a sharp vertical transition of color or intensity. It will flicker at one-half the refresh rate.

A better solution is to use two lines of non-interlaced data to generate one line of interlace data. Fast vertical transitions are smoothed out over several interlace lines.

The vertical filtering of two non-interlaced lines can be done by enabling the vertical scaling offset dynamic change in different field (even/odd). The dynamic change value of scaling offset is half of the scaling factor. You should enable the scaling down vertical offset in scaling down mode; enable the scaling up vertical offset in scaling up mode, or one of it in no-scaling mode.

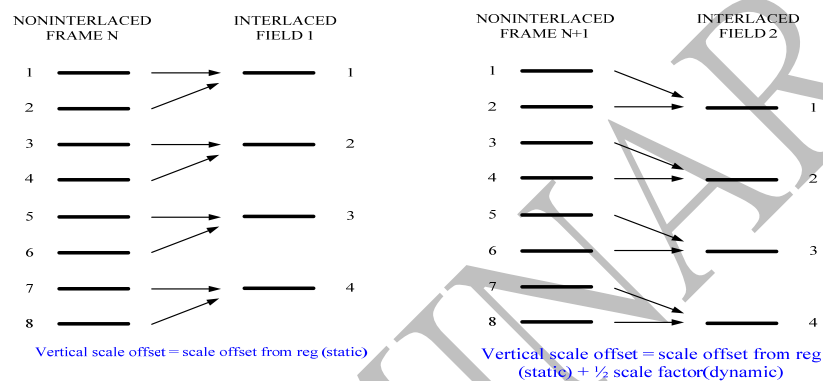


Fig. 9-11 RK281x LCDC Interlace vertical filtering

### 9.4.6 Overlay

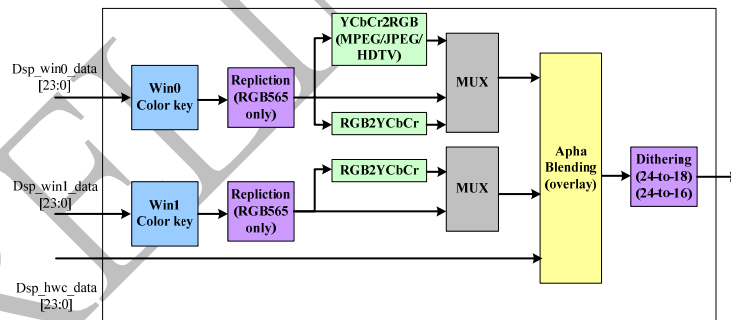


Fig. 9-12 RK281x LCDC Overlay block diagram

#### 1. Overlay display

There are totally 4 layers for overlay display: Background, Win0, Win1 and Hwc.

The background is a programmable solid color layer, which is always the bottom of the display screen.

Hwc is a 32x32 3-LUT-colors layer, which is on the top layer of the display screen.

The two middle layers are WIN0 and WIN1. WIN1 is on the top of WIN0 in default setting, setting LCDC\_BLEND\_CTRL [5] to '1' can let WIN0 be on the top of WIN1.

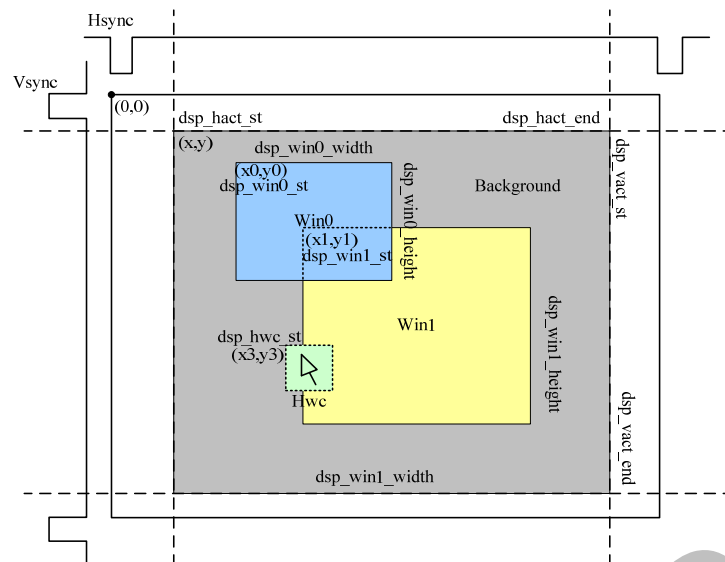


Fig. 9-13 RK281x LCDC Overlay display

## 2. Transparency color key

There are two transparency color keys in Graphic (WIN1) layer and Video (WIN0) layer. The two transparency color key can be active at the same time.

The pixel color value is compared to the transparency color key before final display. The transparency color key value defines the pixel data considered as the transparent pixel. The pixel values with the source color key value are pixels not visible on the screen, and the under layer pixel values or solid background color are visible.

Transparency color key is done after the scaling module and before the YCbCr2RGB color space converter. So transparency color key can only be used in non-scaling mode and YCbCr color key is available in WIN0 layer.

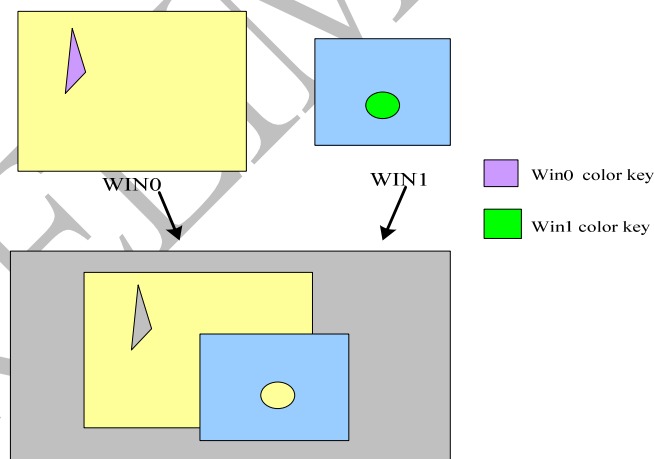


Fig. 9-14 RK281x LCDC Transparency color key

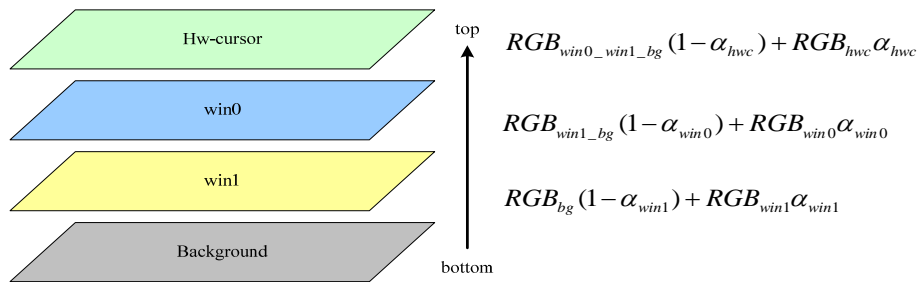
## 3. Alpha Blending

There are 3 alpha for blending between the 4 overlay layers: alpha\_win0 [7:0], alpha\_win1 [7:0], alpha\_hwc [3:0].

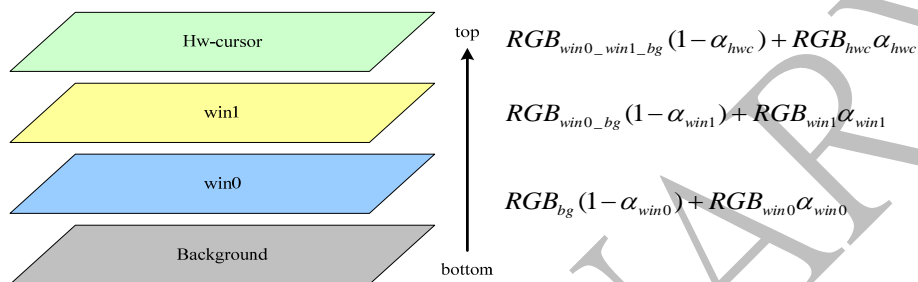
There are two blending mode. One is per-pixel (ARGB) mode; the other is user-specified mode. In ARGB mode, the alpha value is in the ARGB data (WIN0 and WIN1 normal mode only). In user-specified mode, the alpha value comes from the register (LCDC\_BLEND\_CTRL [31:24], LCDC\_BLEND\_CTRL[23:16], LCDC\_BLEND\_CTRL[15:12]).

ARGB blending mode is enabled after writing '1' to LCDC\_BLEND\_CTRL [4] and LCDC\_BLEND\_CTRL [5].

In HWC layer, if the data of the hwc pixel is 2'b00, then this pixel is transparent (alpha = 0), regardless the alpha setting of alpha value.



LCDC\_BLEND\_CTRL[5](win0\_top) = 1



LCDC\_BLEND\_CTRL[5](win0\_top) = 0

Fig. 9-15 RK281x LCDC Alpha blending

#### 4. Replication and Dithering

If the interface data bus is wider than the pixel format size, by programming the pixel components replication active/inactive, the MSB is replicated to the LSB of the interface data bus or the LSB is filled with 0s.

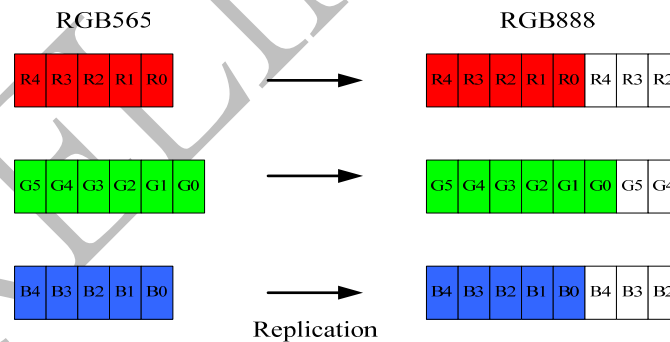


Fig. 9-16 RK281x LCDC Replication

Dithering is an intentionally applied form of noise, used to randomize quantization error, thereby preventing large-scaling patterns such as "banding".

The pixel values are used by Dithering logic to display the data in a lower color depth on the LCD panel. The Dithering algorithm is based on the (x,y) pixel position and the value of removed bits. The picture quality is improved when enabling the Dithering logic. When Dithering is not enabled, the MSBs of the pixel color components are output on the interface data bus if the interface data bus is smaller than the pixel format size.

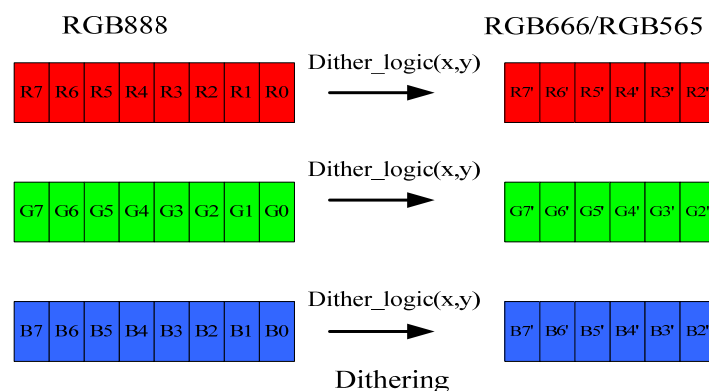


Fig. 9-17 RK281x LCDC Dithering

## 9.4.7 Display Output

### 1. Pin definition

Table. 8-4 RK281x LCDC output pin definition

Pin	RGB mode	MCU mode	CCIR656 mode
DCLK	DCLK	RS	DCLK
HSYNC	HSYNC	WEN	-
VSYNC	VSYNC	CSN	-
DEN	DEN	-	-
DATA	DATA[23:0]	DATA[23:0]	DATA[7:0]

### 2. RGB LCD interface timing

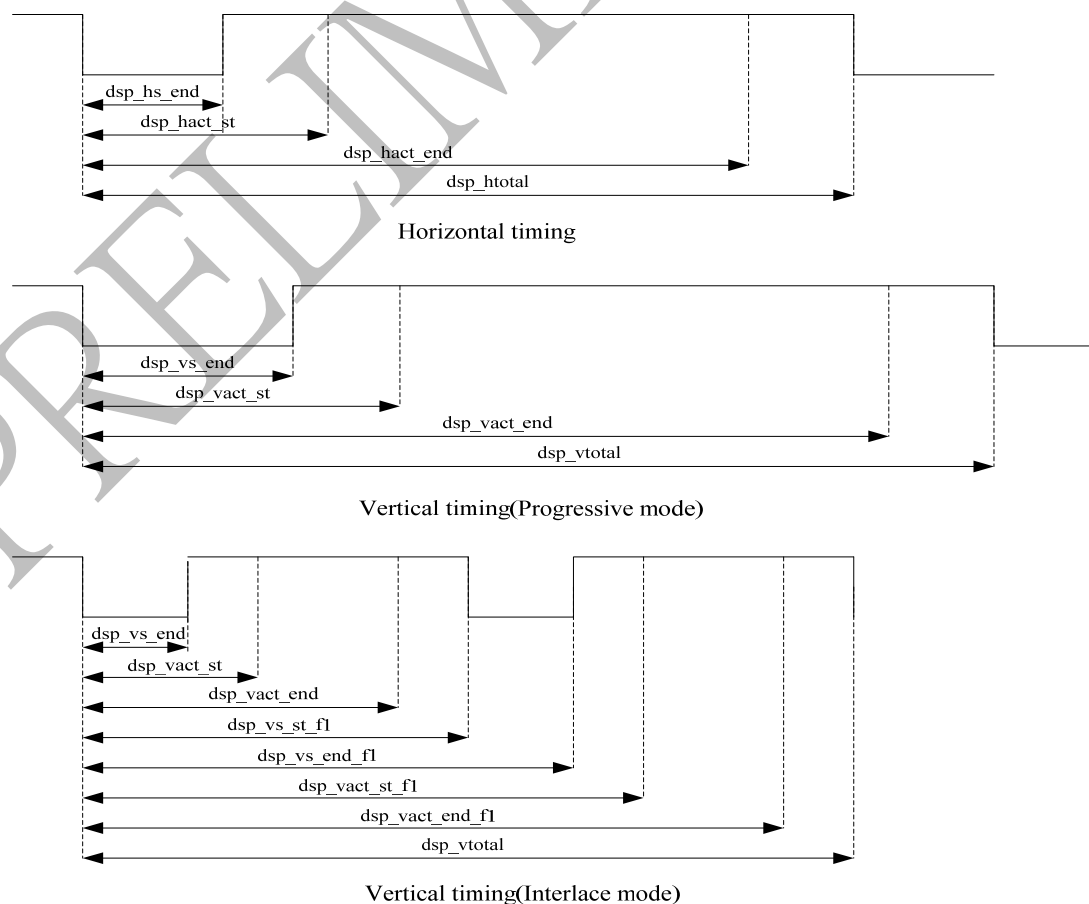


Fig. 9-18 RK281x LCDC RGB LCD interface output timing

## 2. MCU LCD interface timing (i8080)

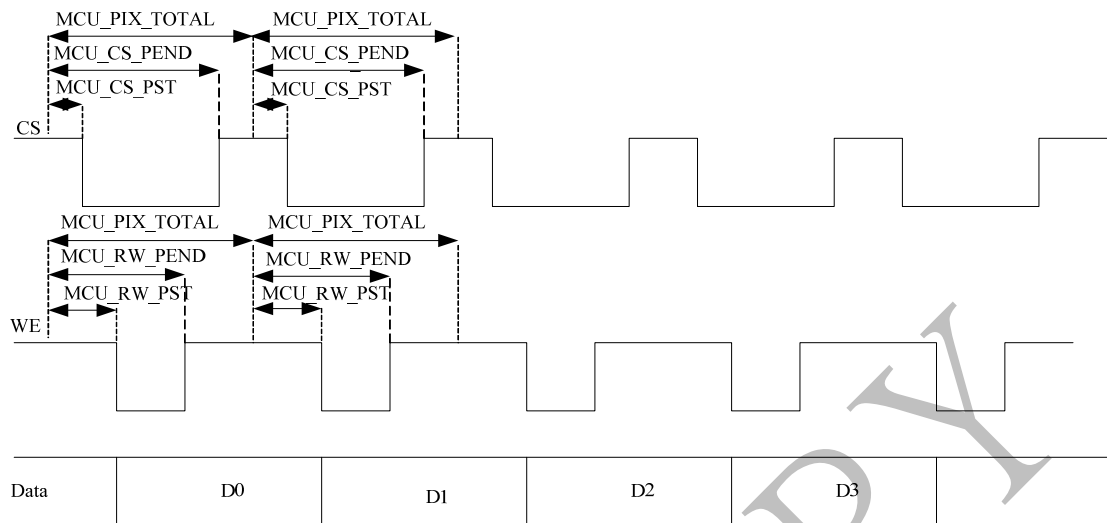


Fig. 9-19 RK281x LCDC MCU LCD interface output timing

## 3. RGB delta LCD interface

RGB delta LCD handles serial 8bit data. In the case of RGB 8bit serial, there are four scanning modes for the RGB delta data. See the detail in the following figure.

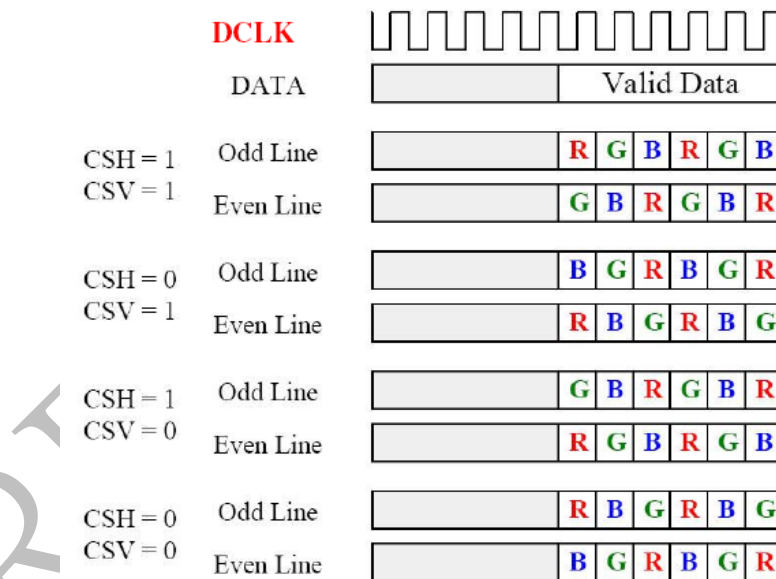


Fig. 9-20 RK281x LCDC RGB delta LCD interface

There are four setting modes for the four scanning modes of RGB delta LCD in LCD controller.

Table. 8-5 RK281x LCDC delta and swap setting for RGB delta LCD

RGB delta LCD scanning mode	Delta_en (LCDC_SWAP_CTRL[15])	Dsp_rg_swap (LCDC_SWAP_CTRL[14])	Dsp_rb_swap (LCDC_SWAP_CTRL[13])	Dsp_bg_swap (LCDC_SWAP_CTRL[12])
CSH=1,CSV=1	1	0	1	0
CSH=0,CSV=1	1	0	0	0
CSH=1,CSV=0	1	0	0	1
CSH=0,CSV=0	1	0	1	1



## 9.5 Use Cases and Tips

1. Do not enable De-flicker and i2p in the same mode;
2. Only win0 support rotation and De-interlace;
3. Vertical width is needed only in vertical roller mode;
4. The minimum active width of win0/win1 is 16;
5. The roller\_x and roller\_y should not overflow the virtual width and height;
6. In De-interlace mode, the reference 0 field is always in current frame;
7. The LCDC share memory can only be used as share memory when LCDC's mode is neither rotation mode nor De-interlace mode.

PRELIMINARY

## Chapter 10 DW\_DMA

### 10.1 Design Overview

#### 10.1.1 Overview

The DW\_DMA controller provides an interface to translate data between AHB and AHB. It can support all kinds of burst type and data size, which define in AHB protocol. It can support on-the-fly DMA transfer on ARMD AHB bus devices and EXP AHB bus data transfer.

#### 10.1.2 Features

- Provide AHB-to-AHB bus protocol translation
- Support AHB-to-AHB DMA or Single AHB DMA
- Six DMA channels supported
- Support byte, half word and word data transfer sizes
- Support incremental and fixed addressing mode
- Support block and software DMA transfer mode
- Support all burst type transfer for bridge
- Support SINGLE and all incremental burst type transfer for DMA
- Support fixed channel priority arbitration
- Build-in 6 data FIFO : 64B/32B/16B/32B/16B/16B'
- Channel 0 & 1 support Scatter/Gather transfer
- Channel 0 & 1 support LLP transfer

### 10.2 Architecture

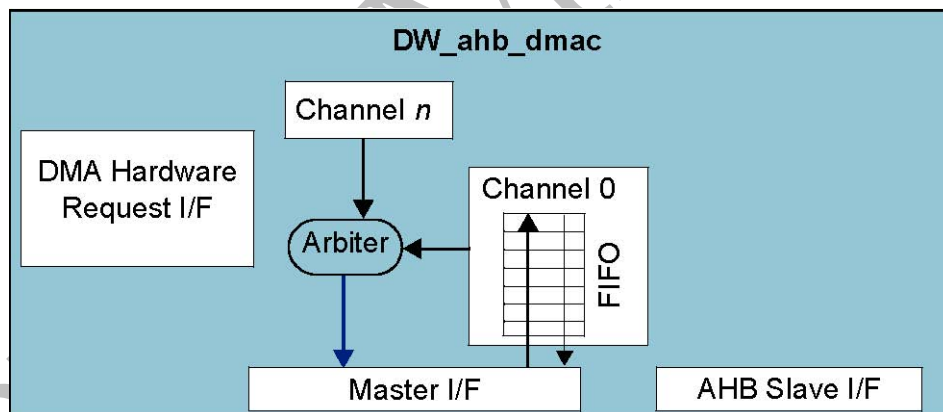


Fig. 10-1 RK281x DW\_DMA Architecture

### 10.3 Registers

#### 10.3.1 Registers Summary

Name	Address Offset	Width	R/W	Description
<b>Channel Registers, x = channels number , (0 ~ 2 )</b>				
SARx	0x000	64	R/W	Channel x Source Address Register Reset Value: 0x0
DARx	0x008	64	R/W	Channel x Destination Address Register Reset Value: 0x0
LLPx	0x010	64	R/W	Channel x Linked List Pointer Register Reset Value: 0x0

CTLx	0x018	64	R/W	Channel x Control Register Reset Value: Configuration dependent
CFGx	0x040	64	R/W	Channel x Configuration Register Reset Value: 0x0000000400000c00
SGRx	0x048	64	R/W	Channel x Source Gather Register Reset Value: 0x0
DSRx	0x050	64	R/W	Channel x Destination Scatter Register Reset Value: 0x0
<b>Interrupt Registers</b>				
RawTfr	0x2c0	64	R	Raw Status for IntTfr Interrupt Reset Value: 0x0
RawBlock	0x2c8	64	R	Raw Status for IntBlock Interrupt Reset Value: 0x0
RawSrcTran	0x2d0	64	R	Raw Status for IntSrcTran Interrupt Reset Value: 0x0
RawDstTran	0x2d8	64	R	Raw Status for IntDstTran Interrupt Reset Value: 0x0
RawErr	0x2e0	64	R	Raw Status for IntErr Interrupt Reset Value: 0x0
<b>StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr</b>				
StatusTfr	0x2e8	64	R	Status for IntTfr Interrupt Reset Value: 0x0
StatusBlock	0x2f0	64	R	Status for IntBlock Interrupt Reset Value: 0x0
StatusSrcTran	0x2f8	64	R	Status for IntSrcTran Interrupt Reset Value: 0x0
StatusDstTran	0x300	64	R	Status for IntDstTran Interrupt Reset Value: 0x0
StatusErr	0x308	64	R	Status for IntErr Interrupt Reset Value: 0x0
<b>MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr</b>				
MaskTfr	0x310	64	R/W	Mask for IntTfr Interrupt Reset Value: 0x0
MaskBlock	0x318	64	R/W	Mask for IntBlock Interrupt Reset Value: 0x0
MaskSrcTran	0x320	64	R/W	Mask for IntSrcTran Interrupt Reset Value: 0x0
MaskDstTran	0x328	64	R/W	Mask for IntDstTran Interrupt Reset Value: 0x0
MaskErr	0x330	64	R/W	Mask for IntErr Interrupt Reset Value: 0x0
<b>ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr</b>				
ClearTfr	0x338	64	W	Clear for IntTfr Interrupt Reset Value: 0x0
ClearBlock	0x340	64	W	Clear for IntBlock Interrupt Reset Value: 0x0
ClearSrcTran	0x348	64	W	Clear for IntSrcTran Interrupt Reset Value: 0x0
ClearDstTran	0x350	64	W	Clear for IntDstTran Interrupt Reset Value: 0x0
ClearErr	0x358	64	W	Clear for IntErr Interrupt Reset Value: 0x0
StatusInt	0x360	64	W	Status for each interrupt type Reset Value: 0x0
<b>Miscellaneous Registers</b>				

DmaCfgReg	0x398	64	R/W	DMA Configuration Register Reset Value: 0x0
ChEnReg	0x3a0	64	R/W	DMA Channel Enable Register Reset Value: 0x0

### 10.3.2 Configuration and Channel Enable Registers

#### DmaCfgReg

- **Name:** DW\_DMA Configuration Register
- **Size:** 64 bits
- **Address Offset:** 0x398
- **Read/Write Access:** Read/Write

This register is used to enable the DW\_DMA, which must be done before any channel activity can begin.



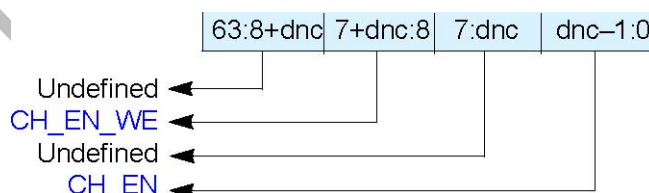
Bits	Name	R/W	Reset	Description
63:1	Undefined	N/A	0x0	Reserved
0	DMA_EN	R/W	0x0	<b>DW_DMA Enable bit.</b> 0 = DW_DMA Disabled 1 = DW_DMA Enabled.

If the global channel enable bit is cleared while any channel is still active, then DmaCfgReg.DMA\_EN still returns 1 to indicate that there are channels still active until hardware has terminated all activity on all channels, at which point the DmaCfgReg.DMA\_EN bit returns 0.

#### ChEnReg

- **Name:** DW\_DMA Channel Enable Register
- **Size:** 64 bits
- **Address Offset:** 0x3a0
- **Read/Write Access:** Read/Write

This is the DW\_DMA Channel Enable Register. If software needs to set up a new channel, then it can read this register in order to find out which channels are currently inactive; it can then enable an inactive channel with the required priority.



Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	CH_EN_WE	W	0x0	<b>Channel enable write enable.</b>
7:4	Undefined	N/A	0x0	Reserved

3:0	CH_EN	R/W	0x0	<b>Enables/Disables the channel.</b> 0 = Disable the Channel 1 = Enable the Channel The ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer.
-----	-------	-----	-----	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

All bits of this register are cleared to 0 when the global DW\_DMA channel enable bit, DmaCfgReg[0], is 0. When the global channel enable bit is 0, then a write to the ChEnReg register is ignored and a read will always read back 0.

The channel enable bit, ChEnReg.CH\_EN, is written only if the corresponding channel write enable bit, ChEnReg.CH\_EN\_WE, is asserted on the same AHB write transfer. For example, writing hex 01x1 writes a 1 into ChEnReg[0], while ChEnReg[7:1] remains unchanged. Writing hex 00xx leaves ChEnReg[7:0] unchanged. Note that a read-modified write is not required.

### 10.3.3 Channel Registers

The channel registers consist of the following, where x = 0 to 2:

- **CFGx** – Configuration register for channel x
- **CTLx** – Control register for channel x
- **DARx** – Destination address register for channel x
- **DSRx** – Destination scatter register for channel x
- **LLPx** – Linked list pointer register for channel x
- **SARx** – Source address register for channel x
- **SGRx** – Source gather register for channel x

The SARx, DARx, LLPx, CTLx, and CFGx channel registers should be programmed prior to enabling the channel. However, if an LLI update occurs before commencing data transfer, SARx and DARx may not need to be programmed prior to enabling the channel; refer to rows 6 to 10 in Table 5. It is an Illegal Register Access when a write to the SARx, DARx, LLPx, CTLx, SSTATx, DSTATx, SSTATARx, DSTATARx, SGRx, or DSRx registers occurs when the channel is enabled.

#### SARx

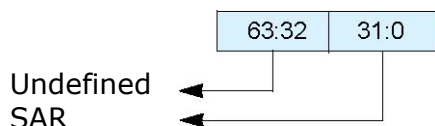
- Name: Source Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:  
 SAR0 – 0x000  
 SAR1 – 0x058  
 SAR2 – 0x0b0
- Read/Write Access: Read/Write

The starting source address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the source address of the current AHB transfer.



Note

You must program the SAR address to be aligned to CTLx.SRC\_TR\_WIDTH.



Bits	Name	R/W	Reset	Description
------	------	-----	-------	-------------

63:32	Undefined	N/A	0x0	Reserved
31:0	SAR	R/W	0x0	<b>Current Source Address of DMA transfer.</b> Updated after each source transfer. The SINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

## DARx

- Name: Destination Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:  
DAR0 – 0x008  
DAR1 – 0x060  
DAR2 – 0x0b8

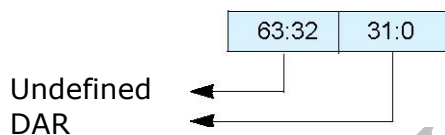
- Read/Write Access: Read/Write

The starting destination address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the destination address of the current AHB transfer.



Note

**You must program the DAR to be aligned to CTLx.DST\_TR\_WIDTH.**



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31:0	DAR	R/W	0x0	<b>Current Destination address of DMA transfer.</b> Updated after each destination transfer. The DINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

## Hardware Realignment of SAR/DAR Registers

In a particular circumstance, during contiguous multi-block DMA transfers, the destination address can become misaligned between the end of one block and the start of the next block. When this situation occurs, DW\_DMA re-aligns the destination address before the start of the next block.

Consider the following example. If the block length is 9, the source transfer width is 16 (halfword), and the destination transfer width is 32 (word)—the destination is programmed for contiguous block transfers—then the destination performs four word transfers followed by a halfword transfer to complete the block transfer to the destination. At the end of the destination block transfer, the address is aligned to a 16-bit transfer as the last AMBA transfer is halfword. This is misaligned to the programmed transfer size of 32 bits for the destination. However, for contiguous destination multi-block transfers, DW\_DMA re-aligns the DAR address to the nearest 32-bit address (next 32-bit address upwards if address control is incrementing or next address downwards if address control is decrementing).

This only occurs when the following is the DMA transfer setup:

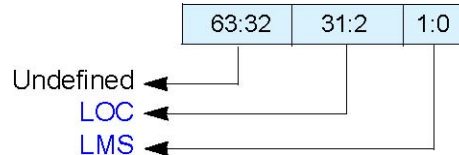
- When on the DAR address, OR
- Contiguous multi-block transfers on destination side, OR
- DST\_TR\_WIDTH > SRC\_TR\_WIDTH, OR
- $(\text{BLOCK\_TS} * \text{SRC\_TR\_WIDTH}) / \text{DST\_TR\_WIDTH} \neq \text{integer}$  (where SRC\_TR\_WIDTH, DST\_TR\_WIDTH is byte width of transfer)

**LLPx**

- Name: Linked List Pointer Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:  
LLP0 – 0x010  
LLP1 – 0x068  
LLP2 – 0x0c0
- Read/Write Access: Read/Write

**Note**

**You need to program this register to point to the first Linked List Item (LLI) in memory prior to enabling the channel if block chaining is enabled.**



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31:2	LOC	R/W	0x0	<b>Starting Address</b> In Memory of next LLI if block chaining is enabled. Note that the two LSBs of the starting address are not stored because the address is assumed to be aligned to a 32-bit boundary.
1:0	LMS	R/W	0x0	<b>List Master Select.</b> Identifies the AHB layer/interface where the memory device that stores the next linked list item resides. 00 = AHB master 1 01 = AHB master 2

The LLP register has two functions:

- The logical result of the equation  $LLP.LOC \neq 0$  is used to set up the type of DMA transfer—single or multi-block. If LLP.LOC is set to 0x0, then transfers using linked lists are not enabled. This register must be programmed prior to enabling the channel in order to set up the transfer type.
- $LLP.LOC \neq 0$  contains the pointer to the next LLI for block chaining using linked lists; The LLPx register can also point to the address where write-back of the control and source/destination status information occur after block completion.

**CTLx**

- Name: Control Register for Channel x
- Size: 64 bits
- Address Offset: for x = 0 to 2:  
CTL0 – 0x018  
CTL1 – 0x070  
CTL2 – 0x0c8
- Read/Write Access: Read/Write

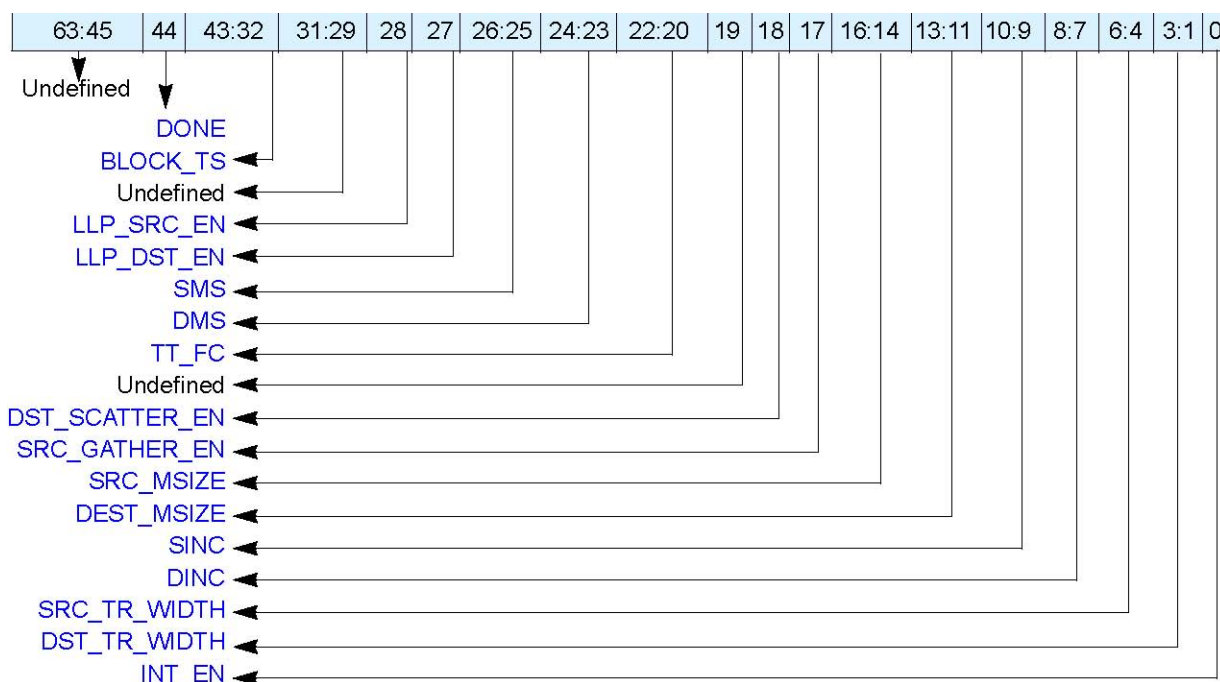
This register contains fields that control the DMA transfer.

The CTLx register is part of the block descriptor (linked list item – LLI) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled. If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the LLI in system memory at the end of the block transfer.

**Note**

**You need to program this register prior to enabling the channel.**





Bits	Name	R/W	Description
63:45	Undefined	N/A	Reserved
44	DONE	R/W	<b>Done bit</b> If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set. Software can poll the LLI CTLx.DONE bit to see when a block transfer is complete. The LLI CTLx.DONE bit should be cleared when the linked lists are set up in memory prior to enabling the channel.
b:32	BLOCK_TS	R/W	<b>Block Transfer Size.</b> When the DW_DMA is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat. Width: The width of the single transaction is determined by CTLx.SRC_TR_WIDTH. Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller. When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at DMAH_CHx_MAX_BLK_SIZE, but the actual block size can be greater. $b = \log_2(\text{DMAH\_CHx\_MAX\_BLK\_SIZE} + 1) + 31$ Bits 43:b+1 do not exist and return 0 on a read. <b>Reset Value:</b> 0x2
31:29	Undefined	N/A	Reserved
28	LLP_SRC_EN	R/W	Block chaining is enabled on the source side only if the LLP_SRC_EN field is high and LLPx.LOC is non-zero; <b>Reset Value:</b> 0x0
27	LLP_DST_EN	R/W	Block chaining is enabled on the destination side only if the LLP_DST_EN field is high and LLPx.LOC is non-zero. <b>Reset Value:</b> 0x0

26:25	SMS	R/W	Source Master Select. Identifies the Master Interface layer from which the source device (peripheral or memory) is accessed. 00 = AHB master 1 01 = AHB master 2 <b>Reset Value:</b> DMAH_CHx_SMS[1:0]
24:23	DMS	R/W	Destination Master Select. Identifies the Master Interface layer where the destination device (peripheral or memory) resides. 00 = AHB master 1 01 = AHB master 2 <b>Reset Value:</b> DMAH_CHx_DMS[1:0]
22:20	TT_FC	R/W	Transfer Type and Flow Control. The following transfer types are supported. <ul style="list-style-type: none"> <li>• Memory to Memory</li> <li>• Memory to Peripheral</li> <li>• Peripheral to Memory</li> <li>• Peripheral to Peripheral</li> </ul> Flow Control can be assigned to the DW_DMA, the source peripheral, or the destination peripheral. Table 3 lists the decoding for this field. <b>Reset Value:</b> Configuration dependent: TT_FC[0] = 1'b1 TT_FC[1] = DMAH_CHx_FC[1]    (!DMAH_CHx_FC[0]) TT_FC[2] = DMAH_CHx_FC[1] ^ DMAH_CHx_FC[0] <b>Dependencies:</b> If the configuration parameter DMAH_CHx_FC is set to DMA_FC_ONLY, then TT_FC[2] does not exist and TT_FC[2] always reads back 0. If DMAH_CHx_FC is set to SRC_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b10. If DMAH_CHx_FC is set to DST_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b11.
19	Undefined	N/A	Reserved
18	DST_SCATTER_EN	R/W	Destination scatter enable bit: 0 = Scatter disabled 1 = Scatter enabled Scatter on the destination side is applicable only when the CTLx.DINC bit indicates an incrementing or decrementing address control. <b>Reset Value:</b> 0x0
17	SRC_GATHER_EN	R/W	Source gather enable bit: 0 = Gather disabled 1 = Gather enabled Gather on the source side is applicable only when the CTLx.SINC bit indicates an incrementing or decrementing address control. <b>Reset Value:</b> 0x0
16:14	SRC_MSIZ	R/W	Source Burst Transaction Length. Number of data items, each of width CTLx.SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 1 lists the decoding for this field; <b>NOTE:</b> This value is not related to the AHB bus master HBURST bus. <b>Reset Value:</b> 0x1
13:11	DEST_MSIZ	R/W	Destination Burst Transaction Length. Number of data items, each of width CTLx.DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface. <b>NOTE:</b> This value is not related to the AHB bus master HBURST bus. <b>Reset Value:</b> 0x1

10:9	SINC	R/W	Source Address Increment. Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change <b>NOTE:</b> Incrementing or decrementing is done for alignment to the next CTLx.SRC_TR_WIDTH boundary. <b>Reset Value:</b> 0x0
8:7	DINC	R/W	Destination Address Increment. Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change <b>NOTE:</b> Incrementing or decrementing is done for alignment to the next CTLx.DST_TR_WIDTH boundary. <b>Reset Value:</b> 0x0
6:4	SRC_TR_WIDTH	R/W	Source Transfer Width. Table 2 lists the decoding for this field. Mapped to AHB bus "hsize." For a non-memory peripheral, typically the peripheral (source) FIFO width. This value must be less than or equal to DMAH_Mx_HDATA_WIDTH, where x is the AHB layer 1 to 2 where the source resides. <b>Reset Value:</b> Encoded value; refer to Table 2.
3:1	DST_TR_WIDTH	R/W	Destination Transfer Width. Table 2 lists the decoding for this field. Mapped to AHB bus "hsize." For a non-memory peripheral, typically rgw peripheral (destination) FIFO width. This value must be less than or equal to DMAH_Mk_HDATA_WIDTH, where k is the AHB layer 1 to 2 where the destination resides. <b>Reset Value:</b> Encoded value; refer to Table 2.
0	INT_EN	R/W	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. <b>Reset Value:</b> 0x1

Table 10-1 DW\_DMA CTLx.SRC\_MSIZ and DEST\_MSIZ Decoding

CTLx.SRC_MSIZ / CTLx.DEST_MSIZ	Number of data items to be transferred (of width CTLx.SRC_TR_WIDTH or CTLx.DST_TR_WIDTH)
000	1
001	4
010	8
011	16
100	32

Table 10-2 DW\_DMA CTLx.SRC\_TR\_WIDTH and CTLx.DST\_TR\_WIDTH Decoding

CTLx.SRC_TR_WIDTH / CTLx.DST_TR_WIDTH	Size (bits)
000	8
001	16
010	32

Table 10-3 DW\_DMA CTLx.TT\_FC field Decoding

CTLx.TT_FC Field	Transfer Type	Flow Controller
000	Memory to Memory	DW_DMA
001	Memory to Peripheral	DW_DMA
010	Peripheral to Memory	DW_DMA
011	Peripheral to Peripheral	DW_DMA

100	Peripheral to Memory	Peripheral
101	Peripheral to Peripheral	Source Peripheral
110	Memory to Peripheral	Peripheral
111	Peripheral to Peripheral	Destination Peripheral

**CFGx**

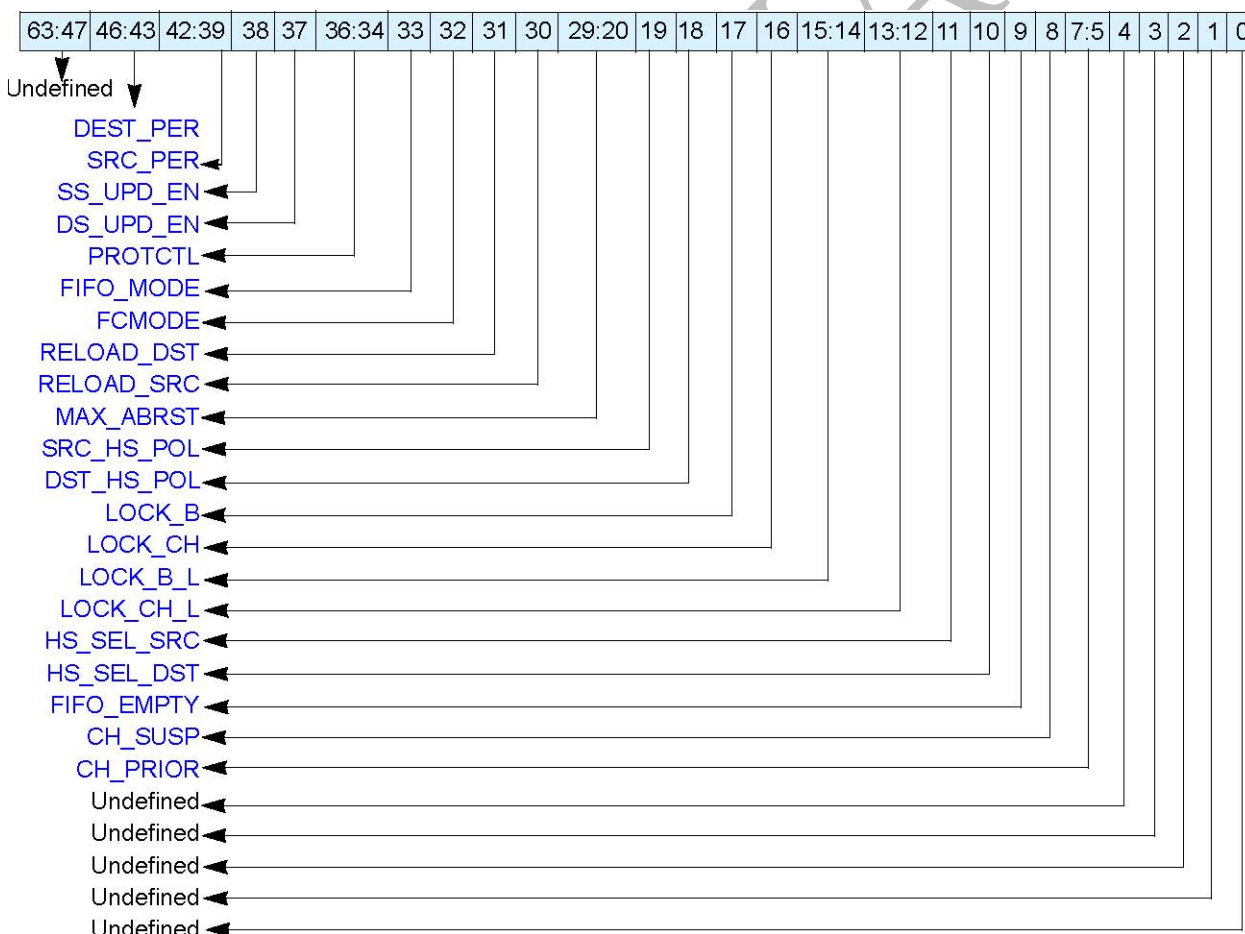
- Name: Configuration Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:  
CFG0 – 0x040  
CFG1 – 0x098  
CFG2 – 0x0f0

- Read/Write Access: Read/Write

This register contains fields that configure the DMA transfer. The channel configuration register remains fixed for all blocks of a multi-block transfer.

**Note**

You need to program this register prior to enabling the channel.



Bits	Name	R/W	Reset	Description
63:47	Undefined	N/A	0x0	Reserved

46:43	DEST_PER	R/W	0x0	<p>Assigns a hardware handshaking interface (0 -DMAH_NUM_HS_INT-1) to the destination of channel x if the CFGx.HS_SEL_DST field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface.</p> <p><b>NOTE:</b> For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.</p>
42:39	SRC_PER	R/W	0x0	<p>Assigns a hardware handshaking interface (0 -DMAH_NUM_HS_INT-1) to the source of channel x if the CFGx.HS_SEL_SRC field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface.</p> <p><b>NOTE:</b> For correct DW_DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.</p>
38	SS_UPD_EN	R/W	0x0	<p><b>Source Status Update Enable.</b> Source status information is fetched only from the location pointed to by the SSTATARx register, stored in the SSTATx register and written out to the SSTATx location of the LLI if SS_UPD_EN is high.</p> <p><b>NOTE:</b> This enable is applicable only if DMAH_CHx_STAT_SRC is set to True.</p>
37	DS_UPD_EN	R/W	0x0	<p><b>Destination Status Update Enable.</b> Destination status information is fetched only from the location pointed to by the DSTATARx register, stored in the DSTATx register and written out to the DSTATx location of the LLI if DS_UPD_EN is high.</p>
36:34	PROTCTL	R/W	0x1	<p>Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals.</p> <p>Table 4 shows the mapping of bits in this field to the AHB HPROT[3:1] bus.</p>
33	FIFO_MODE	R/W	0x0	<p><b>FIFO Mode Select.</b> Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.</p> <p>0 = Space/data available for single AHB transfer of the specified transfer width.</p> <p>1 = Space/data available is greater than or equal to half the FIFO depth for destination transfers and less than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.</p>
32	FCMODE	R/W	0x0	<p><b>Flow Control Mode.</b> Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.</p> <p>0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled.</p> <p>1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is</p>

				disabled.
31	RELOAD_DST	R/W	0x0	<b>Automatic Destination Reload.</b> The DArx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. For conditions under which this occurs, refer to Table 5.
30	RELOAD_SRC	R/W	0x0	<b>Automatic Source Reload.</b> The SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. For conditions under which this occurs.
29:20	MAX_ABRST	R/W	0x0	<b>Maximum AMBA Burst Length.</b> Maximum AMBA burst length that is used for DMA transfers on this channel. A value of 0 indicates that software is not limiting the maximum AMBA burst length for DMA transfers on this channel.
19	SRC_HS_POL	R/W	0x0	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
18	DST_HS_POL	R/W	0x0	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
17	LOCK_B	R/W	0x0	<b>Bus Lock Bit.</b> When active, the AHB bus master signal hlock is asserted for the duration specified in CFGx.LOCK_B_L.
16	LOCK_CH	R/W	0x0	<b>Channel Lock Bit.</b> When the channel is granted control of the master bus interface and if the CFGx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CFGx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CFGx.LOCK_CH_L.
15:14	LOCK_B_L	R/W	0x0	<b>Bus Lock Level.</b> Indicates the duration over which CFGx.LOCK_B bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction
13:12	LOCK_CH_L	R/W	0x0	<b>Channel Lock Level.</b> Indicates the duration over which CFGx.LOCK_CH bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction



11	HS_SEL_SRC	R/W	0x1	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
10	HS_SEL_DST	R/W	0x1	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
9	FIFO_EMPTY	R	0x0	Indicates if there is data left in the channel FIFO. Can be used in conjunction with CFGx.CH_SUSP to cleanly disable a channel. 1 = Channel FIFO empty 0 = Channel FIFO not empty
8	CH_SUSP	R/W	0x0	Channel Suspend. Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMA transfer from the source.
7:5	CH_PRIOR	R/W	Channel Number example: Chan0=0 Chan1=1	Channel priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range: 0: (DMAH_NUM_CHANNELS – 1) A programmed value outside this range will cause erroneous behavior.
4:0	Undefined	N/A	0x0	Reserved

Table 10-4 DW\_DMA PROTCTL field to HPROT Mapping

1'b1	HPROT[0]
CFGx.PROTCTL[1]	HPROT[1]
CFGx.PROTCTL[2] ->	HPROT[2]
CFGx.PROTCTL[3] ->	HPROT[3]

**SGRx**

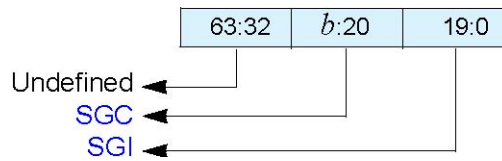
- Name: Source Gather Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 1:  
SGR0 – 0x048  
SGR1 – 0x0a0
- Read/Write Access: Read/Write

The Source Gather register contains two fields:

- Source gather count field (SGRx.SGC) – Specifies the number of contiguous source transfers of CTLx.SRC\_TR\_WIDTH between successive gather intervals. This is defined as a gather boundary.
- Source gather interval field (SGRx.SGI) – Specifies the source address increment/decrement in multiples of CTLx.SRC\_TR\_WIDTH on a gather boundary when gather mode is enabled for the source transfer.



The CTLx.SINC field controls whether the address increments or decrements. When the CTLx.SINC field indicates a fixed-address control, then the address remains constant throughout the transfer and the SGRx register is ignored. This register does not exist if the configuration parameter DMAH\_CHx\_SRC\_GAT\_EN is set to False.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved.
b:20 See description	SGC	R/W	0x0	Source gather count. Source contiguous transfer count between successive gather boundaries. b = log2 (DMAH_CHx_MAX_BLK_SIZE + 1) + 19 Bits 31:b+1 do not exist and read back as 0.
19:0	SGI	R/W	0x0	Source gather interval.

### DSRx

- Name: Destination Scatter Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 1:  
DSR0 – 0x050  
DSR1 – 0x0a8
- Read/Write Access: Read/Write

The Destination Scatter register contains two fields:

- Destination scatter count field (DSRx.DSC) – Specifies the number of contiguous destination transfers of CTLx.DST\_TR\_WIDTH between successive scatter boundaries.
- Destination scatter interval field (DSRx.DSI) – Specifies the destination address increment/ decrement in multiples of CTLx.DST\_TR\_WIDTH on a scatter boundary when scatter mode is enabled for the destination transfer.

The CTLx.DINC field controls whether the address increments or decrements. When the CTLx.DINC field indicates a fixed address control, then the address remains constant throughout the transfer and the DSRx register is ignored. This register does not exist if the configuration parameter DMAH\_CHx\_DST\_SCA\_EN is set to False.

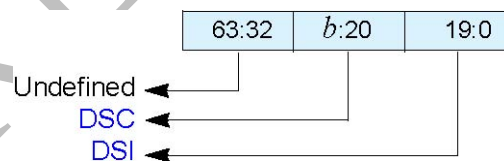


Table 10-5 DW\_DMA Destination Scatter Register Description for Channel x

Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved.
b:20 See description	DSC	R/W	0x0	Destination scatter count. Destination contiguous transfer count between successive scatter boundaries. b = log2 (DMAH_CHx_MAX_BLK_SIZE + 1) + 19 Bits 31:b+1 do not exist and read 0.
19:0	DSI	R/W	0x0	Destination scatter interval.

### 10.3.4 Interrupt Registers

The following sections describe the registers pertaining to interrupts, their status, and

how to clear them. For each channel, there are five types of interrupt sources:

- IntBlock – Block Transfer Complete Interrupt This interrupt is generated on DMA block transfer completion to the destination peripheral.
- IntDstTran – Destination Transaction Complete Interrupt

This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the destination side.



#### Note

**If the destination for a channel is memory, then that channel will never generate the IntDstTran interrupt. Because of this, the corresponding bit in this field will not be set.**

- IntErr – Error Interrupt  
This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.
- IntSrcTran – Source Transaction Complete Interrupt

This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the source side.



#### Note

**If the source for a channel is memory, then that channel will never generate a IntSrcTran interrupt. Because of this, the corresponding bit in this field will not be set.**

- IntTfr – DMA Transfer Complete Interrupt

This interrupt is generated on DMA transfer completion to the destination peripheral.

There are several groups of interrupt-related registers:

- RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr
- StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr
- MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr
- ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr
- StatusInt

When a channel has been enabled to generate interrupts, the following is true:

- Interrupt events are stored in the Raw Status registers.
- The contents of the Raw Status registers are masked with the contents of the Mask registers.
- The masked interrupts are stored in the Status registers.
- The contents of the Status registers are used to drive the int\_\* port signals.
- Writing to the appropriate bit in the Clear registers clears an interrupt in the Raw Status registers and the Status registers on the same clock cycle.

The contents of each of the five Status registers is ORed to produce a single bit for each interrupt type in the Combined Status register; that is, StatusInt.



#### Note

The CTLx.INT\_EN bit must be set for an enabled channel to generate any interrupts.

#### RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr

- Name: Interrupt Raw Status Registers
- Size: 64 bits
- Address Offset:  
RawTfr – 0x2c0  
RawBlock – 0x2c8  
RawSrcTran – 0x2d0

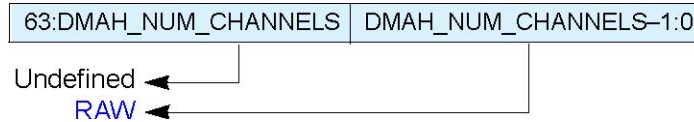
RawDstTran – 0x2d8

RawErr – 0x2e0

- Read/Write Access: Read

Interrupt events are stored in these Raw Interrupt Status registers before masking: RawBlock, RawDstTran, RawErr, RawSrcTran, and RawTfr. Each Raw Interrupt Status register has a bit allocated per channel; for example, RawTfr[2] is the Channel 2 raw transfer complete interrupt.

Each bit in these registers is cleared by writing a 1 to the corresponding location in the ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr registers.

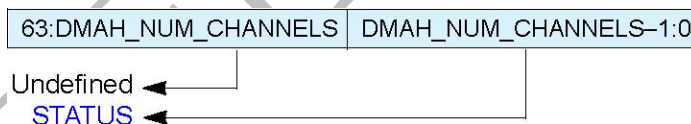


Bits	Name	R/W	Reset	Description
63:4	Undefined	N/A	0x0	Reserved
3:0	RAW	R	0x0	Raw interrupt status.

#### StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr

- Name: Interrupt Status Registers
- Size: 64 bits
- Address Offset:
  - StatusTfr – 0x2e8
  - StatusBlock – 0x2f0
  - StatusSrcTran – 0x2f8
  - StatusDstTran – 0x300
  - StatusErr – 0x308
- Read/Write Access: Read

All interrupt events from all channels are stored in these Interrupt Status registers after masking: StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, and StatusTfr. Each Interrupt Status register has a bit allocated per channel; for example, StatusTfr[2] is the Channel 2 status transfer complete interrupt. The contents of these registers are used to generate the interrupt signals (int or int\_n bus, depending on interrupt polarity) leaving the DW\_DMA.



Bits	Name	R/W	Reset	Description
63:4	Undefined	N/A	0x0	Reserved
3:0	STATUS	R	0x0	Interrupt status.

#### MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr

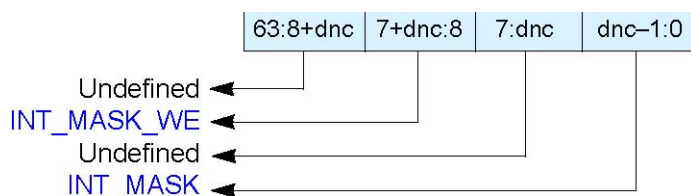
- Name: Interrupt Mask Registers
- Size: 64 bits
- Address Offset:
  - MaskTfr – 0x310
  - MaskBlock – 0x318
  - MaskSrcTran – 0x320
  - MaskDstTran – 0x328
  - MaskErr – 0x330
- Read/Write Access: Read/Write

The contents of the Raw Status registers are masked with the contents of the Mask registers: MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, and MaskTfr. Each Interrupt

Mask register has a bit allocated per channel; for example, MaskTfr[2] is the mask bit for the Channel 2 transfer complete interrupt.

When the source peripheral of DMA channel  $i$  is memory, then the source transaction complete interrupt, MaskSrcTran[ $i$ ], must be masked to prevent an erroneous triggering of an interrupt on the int\_combined signal. Similarly, when the destination peripheral of DMA channel  $i$  is memory, then the destination transaction complete interrupt, MaskDstTran[ $i$ ], must be masked to prevent an erroneous triggering of an interrupt on the int\_combined( $_n$ ) signal.

A channel INT\_MASK bit will be written only if the corresponding mask write enable bit in the INT\_MASK\_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation. For example, writing hex 01x1 to the MaskTfr register writes a 1 into MaskTfr[0], while MaskTfr[7:1] remains unchanged. Writing hex 00xx leaves MaskTfr[7:0] unchanged. Writing a 1 to any bit in these registers unmasks the corresponding interrupt, thus allowing the DW\_DMA to set the appropriate bit in the Status registers and int\_\* port signals.

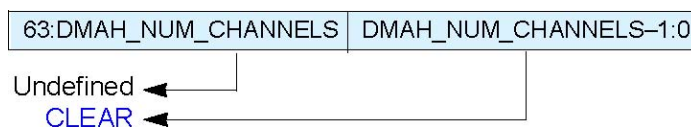


Bits	Name	R/W	Description
63:12	Undefined	N/A	<b>Reset Value:</b> 0x0
11:8	INT_MASK_WE	W	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled <b>Reset Value:</b> 0x0
7:4	Undefined	N/A	<b>Reset Value:</b> 0x0
3:0	INT_MASK	R/W	Interrupt Mask 0 = masked 1 = unmasked <b>Reset Value:</b> 0x0

#### ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr

- Name: Interrupt Clear Registers
- Size: 64 bits
- Address Offset:
  - ClearTfr – 0x338
  - ClearBlock – 0x340
  - ClearSrcTran – 0x348
  - ClearDstTran – 0x350
  - ClearErr – 0x358
- Read/Write Access: Write

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, and ClearTfr. Each Interrupt Clear register has a bit allocated per channel; for example, ClearTfr[2] is the clear bit for the Channel 2 transfer complete interrupt. Writing a 0 has no effect. These registers are not readable.



Bits	Name	R/W	Reset	Description
------	------	-----	-------	-------------

63:4	Undefined	N/A	0x0	Reserved
3:0	CLEAR	W	N/A	Interrupt clear. 0 = no effect 1 = clear interrupt

### StatusInt

- Name: Combined Interrupt Status Register
- Size: 64 bits
- Address Offset: 0x360
- Read/Write Access: Read

## 10.4 Register Access

All registers are aligned to a 64-bit boundary and are 64 bits wide. In general, the upper 32 bits of a register are reserved. A write to reserved bits within the register is ignored. A read from reserved bits in the register reads back 0. To avoid address aliasing, do one of the following:

The DW\_dma should not be allocated more than 1 KB of address space in the system memory map. If it is, then addresses selected above 1 KB from the base address are aliased to an address within the 1 KB space, and a transfer takes place involving this register.

Software should not attempt to access non-register locations when hsel is asserted.



### Note

The hsel signal is asserted by the system decoder when the address on the bus is within the system address assigned for DW\_DMA.

## 10.5 Illegal Register Access

An illegal access can be any of the following:

1. A AHB transfer of hsize greater than 64 is attempted.
2. The hsel signal is asserted, but the address does not decode to a valid address.
3. A write to the SARx, DARx, LLPx, CTLx, SSTATx, DSTATx, SSTATARx, DSTATARx, SGRx, or DSRx registers occurs when the channel is enabled.
4. A read from the ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr is attempted.
5. A write to the StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr is attempted.
6. A write to the StatusInt register is attempted.
7. A write to either the DmaIdReg or DMA Component ID Register register is attempted.

The response to an illegal access is configured using the configuration parameter DMAH\_RETURN\_ERR\_RESP. When DMAH\_RETURN\_ERR\_RESP is set to True, an illegal access (read/write) returns an error response.

If DMAH\_RETURN\_ERR\_RESP is set to False, an OKAY response is returned, a read reads back 0x0, and a write is ignored.

## 10.6 DW\_DMA Transfer Types

A DMA transfer may consist of single or multi-block transfers. On successive blocks of a multi-block transfer, the SARx/DARx register in the DW\_DMA is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the CTLx register in the DW\_DMA is

reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using Linked Lists is the multi-block method of choice. On successive blocks, the LLPx register in the DW\_DMA is reprogrammed using block chaining with linked lists.

A block descriptor consists of six registers: SARx, DARx, LLPx, CTLx, SSTATx, and DSTATx. The first four registers, along with the CFGx register, are used by the DW\_DMA to set up and describe the block transfer.



Note

**The term Link List Item (LLI) and block descriptor are synonymous.**

### Multi-Block Transfers

Multi-block transfers are enabled by setting the DMAH\_CHX\_MULTI\_BLK\_EN configuration parameter to True.

#### Block Chaining Using Linked Lists

To enable multi-block transfers using block chaining, you must set the configuration parameter DMAH\_CHx\_MULTI\_BLK\_EN to True and the DMAH\_CHx\_HC\_LLP parameter to False.

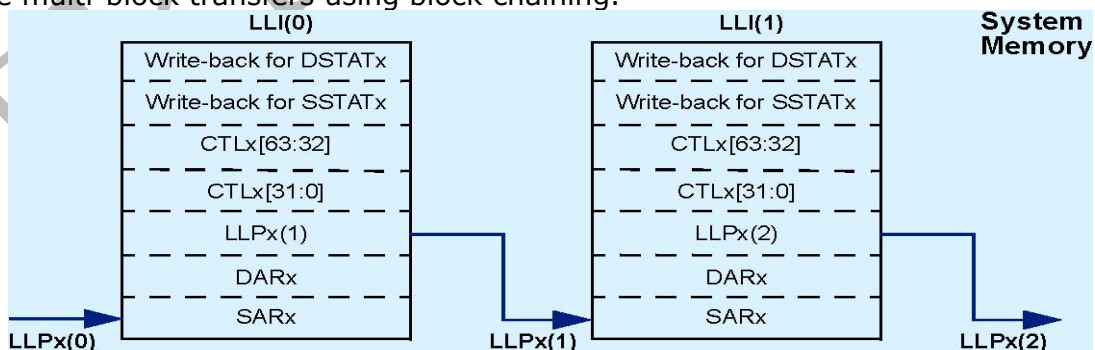
In this case, the DW\_DMA reprograms the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

DW\_DMA block chaining uses a Linked List Pointer register (LLPx) that stores the address in memory of the next linked list item. Each LLI contains the corresponding block descriptors:

1. SARx
2. DARx
3. LLPx
4. CTLx
5. SSTATx
6. DSTATx

To set up block chaining, you program a sequence of Linked Lists in memory.

The SARx, DARx, LLPx, and CTLx registers are fetched from system memory on an LLI update. If configuration parameter DMAH\_CHx\_CTL\_WB\_EN = True, then the updated contents of the CTLx, SSTATx, and DSTATx registers are written back to memory on block completion. Figure 2 and Figure 3 show how you use chained linked lists in memory to define multi-block transfers using block chaining.

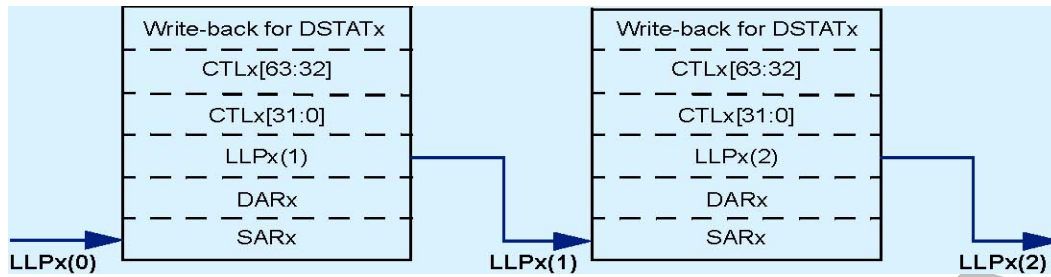


It is assumed that no allocation is made in system memory for the source status when the configuration parameter DMAH\_CHx\_STAT\_SRC is set to False. If this parameter is False, then the order of a Linked List item is as follows:

1. SARx
2. DARx
3. LLPx
4. CTLx



## 5. DSTATx

**Note**

In order to not confuse the SARx, DARx, LLPx, CTLx, STATx, and DSTATx register locations of the LLI with the corresponding DW\_DMA memory mapped register locations, the LLI register locations are prefixed with LLI; that is, LLI.SARx, LLI.DARx, LLI.LLPx, LLI.CTLx, LLI.SSTATx, and LLI.DSTATx.

**Note**

For rows 6 through 10 of Table 5, the LLI.CTLx, LLI.LLPx, LLI.SARx, and LLI.DARx register locations of the LLI are always affected at the start of every block transfer. The LLI.LLPx and LLI.CTLx locations are always used to reprogram the DW\_DMA LLPx and CTLx registers. However, depending on the Table 5 row number, the LLI.SARx/LLI.DARx address may or may not be used to reprogram the DW\_DMA SARx/DARx registers.

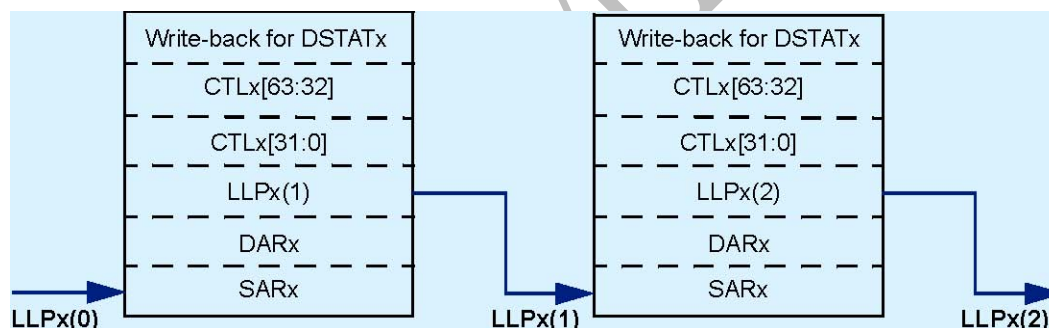
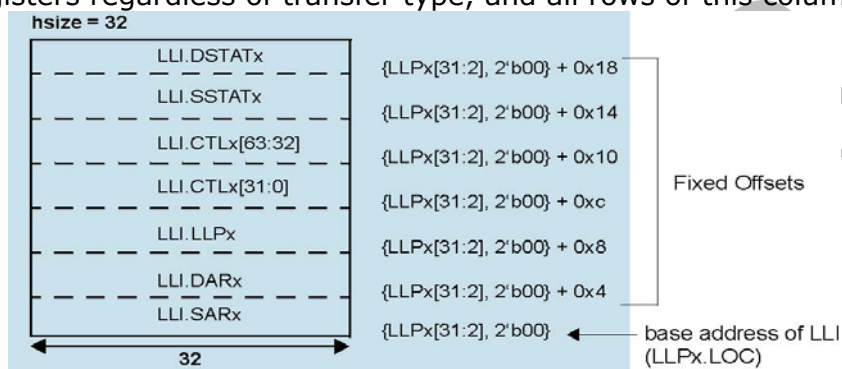
Table 10-6 Programming of Transfer Types and Channel Register Update Method

Transfer Type	LLP. LOC =0	LLP_ SRC_EN (CTLx)	RELOAD _SRC (CFGx)	LLP_ DST_EN (CTLx)	RELOAD _DST (CFGx)	CTLx, LLPx Update Method	SARx Update Method	DARx Update Method	Write Backa
1. Single-block or last transfer of multi-block.	Yes	0	0	0	0	None, user reprograms	None (single)	None (single)	No
2. Auto-reload multi-block transfer with contiguous SAR	Yes	0	0	0	1	CTLx, LLPx are reloaded from initial values.	Con-tiguous	Auto-Reload	No
3. Auto-reload multi-block transfer with contiguous DAR.	Yes	0	1	0	0	CTLx, LLPx are reloaded from initial values	Auto-Reload	Con-tiguous	No
4. Auto-reload multi-block transfer	Yes	0	1	0	1	CTLx, LLPx are reloaded from initial values	Auto-reload	Auto-Reload	No
5. Single-block or last transfer of multi-block.	No	0	0	0	0	None, user reprograms	None (single)	None (single)	Yes
6. Linked list multi-block transfer with contiguous SAR	No	0	0	1	0	CTLx, LLPx loaded from next Linked List item.	Con-tiguous	Linked List	Yes
7. Linked list multi-block transfer with	No	0	1	1	0	CTLx, LLPx loaded from next Linked	Auto-Reload	Linked List	Yes



auto-reload SAR						List item.			
8. Linked list multi-block transfer with contiguous DAR	No	1	0	0	0	CTLx, LLPx loaded from next Linked List item.	Linked List	Contiguous	Yes
9. Linked list multi-block transfer with auto-reload DAR	No	1	0	0	1	CTLx, LLPx loaded from next Linked List item.	Linked List	Auto-Reload	Yes
10. Linked list multi-block transfer	No	1	0	1	0	CTLx, LLPx loaded from next Linked List item.	Linked List	Linked List	Yes

a. This column assumes that the configuration parameter DMAH\_CHx\_CTL\_WB\_EN = True. If DMAH\_CHx\_CTL\_WB\_EN = False, then there is never writeback of the control and status registers regardless of transfer type, and all rows of this column are "No".



## Chapter 11 Interrupt Controller (INTC)

### 11.1 Design Overview

#### 11.1.1 Overview

The INTC is a configurable, vectored interrupt controller for AMBA-based systems. It is an AMBA 2.0-compliant Advanced High-speed Bus (AHB) slave device.

#### 11.1.2 Features

- 48 IRQ normal interrupt sources
- 2 FIQ fast interrupt sources
- Software interrupts
- Priority filtering
- Masking
- Scan mode
- Programmable interrupt priorities
- Configuration ID registers
- Encoded parameters

### 11.2 Architecture

This section describes the functional operation of AHB Interrupt Controller.

#### 11.2.1 Block Diagram

The INTC comprises with:

- Slave I/F – AHB bus interface
- IRQ\_Generation – IRQ generation module
- FIQ\_Generation – FIQ generation module
- Mask – Interrupt Mask module

The diagram is shown as followed

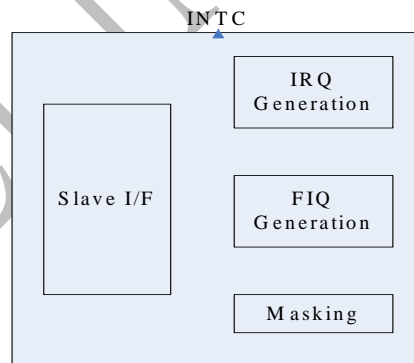


Fig. 11-1 RK281x Interrupt Controller Architecture

### 11.3 Registers

This section describes the control/status registers of the design

#### 11.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
IRQ_INTEN_L	0x00	W	0x0	IRQ interrupt source enable register (low)
IRQ_INTEN_H	0x04	W	0x0	IRQ interrupt source enable register (high).
IRQ_INTMASK_L	0x08	W	0x0	IRQ interrupt source mask

				register (low).
IRQ_INTMASK_H	0x0C	W	0x0	IRQ interrupt source mask register (high).
IRQ_INTFORCE_L	0x10	W	0x0	IRQ interrupt force register (low).
IRQ_INTFORCE_H	0x14	W	0x0	IRQ interrupt force register (high).
IRQ_RAWSTATUS_L	0x18	W	0x0	IRQ raw status register (low).
IRQ_RAWSTATUS_H	0x1c	W	0x0	IRQ raw status register (high)
IRQ_STATUS_L	0x20	W	0x0	IRQ status register (low)
IRQ_STATUS_H	0x24	W	0x0	IRQ status register (high)
IRQ_MASKSTATUS_L	0x28	W	0x0	IRQ interrupt mask status register (low)
IRQ_MASKSTATUS_H	0x2c	W	0x0	IRQ interrupt mask status register (high)
IRQ_FINALSTATUS_L	0x30	W	0x0	IRQ interrupt final status (low)
IRQ_FINALSTATUS_H	0x34	W	0x0	IRQ interrupt final status (high)
FIQ_INTEN	0xc0	W	0x0	Fast interrupt enable register
FIQ_INTMASK	0xc4	W	0x0	Fast interrupt mask register
FIQ_INTFORCE	0xc8	W	0x0	Fast interrupt force register
FIQ_RAWSTATUS	0xcc	W	0x0	Fast interrupt source raw status register
FIQ_STATUS	0xd0	W	0x0	Fast interrupt status register
FIQ_FINALSTATUS	0xd4	W	0x0	Fast interrupt final status register
IRQ_PLEVEL	0xd8	W	0x0	IRQ System Priority Level Register
IRQ_PN_OFFSET	0xe8 + N*4	W	N	Interrupt N priority level register(s), where N is from 0 to 15
IRQ_PN_OFFSET	0xe8 + N*4	W	N-16	Interrupt N priority level register(s), where N is from 16 to 31
IRQ_PN_OFFSET	0xe8 + N*4	W	N-32	Interrupt N priority level register(s), where N is from 32 to 39
AHB_ICTL_COMP_VERSION	0x3f8	W	0x3230342a	Version register
ICTL_COMP_TYPE	0x3fc	W	0x44571120	Component Type Register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 11.3.2 Detail Register Description

#### IRQ\_INTEN\_L

Address: Operational Base + offset( 0x00)

Interrupt Source Enable (Low) Register

bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:0	RW	0x0	Interrupt enable bits for lower 32 interrupt sources. A 1 in any bit position enables the corresponding interrupt. 0: disable interrupt 1: enable interrupt
------	----	-----	-------------------------------------------------------------------------------------------------------------------------------------------------------------------

**IRQ\_INTEN\_H**

Address: Operational Base + offset( 0x04)

Interrupt Source Enable (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Interrupt enable bit for upper 8 interrupt sources. A 1 in any bit position enables the corresponding interrupt. 0: disable interrupt 1: enable interrupt

**IRQ\_INTMASK\_L**

Address: Operational Base + offset( 0x08)

Interrupt Source Mask (Low) Register

bit	Attr	Reset Value	Description
31:0	RW	0x0	Interrupt mask bits for the lower 32 interrupt sources. A 1 in any bit position masks (disables) the corresponding interrupt. By default, all bits are unmasked. 0: unmask interrupt 1: mask interrupt

**IRQ\_INTMASK\_H**

Address: Operational Base + offset( 0x0c)

Interrupt Source Mask (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Interrupt mask bits for the upper 8 interrupt sources. 0: unmask interrupt 1: mask interrupt

**IRQ\_INTFORCE\_L**

Address: Operational Base + offset( 0x10)

Interrupt Force (Low) Register

bit	Attr	Reset Value	Description
31:0	RW	0x0	Interrupt force bits for the lower 32 interrupt sources. Each bit in this register corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register correspond to the polarity of the associated irq_intsrc input. If the interrupt input is configured to be active high, the corresponding bit in the register is also active high. 0: active low 1: active high

**IRQ\_INTFORCE\_H**

Address: Operational Base + offset( 0x14)

Interrupt Force (High) Register

bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:8	-	-	Reserved.
7:0	RW	0x0	Interrupt force bits for the upper 8 interrupt sources. Each bit in this register corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register correspond to the polarity of the associated irq_intsrc input. If the interrupt input is configured to be active high, the corresponding bit in the register is also active high. The reset state of the force bits is always inactive. 0: active low 1: active high

**IRQ\_RAWSTATUS\_L**

Address: Operational Base + offset( 0x18)

Interrupt Raw Status (Low) Register

bit	Attr	Reset Value	Description
31:0	R	0x0	Actual interrupt source.

**IRQ\_RAWSTATUS\_H**

Address: Operational Base + offset( 0x1c)

Interrupt Raw Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Actual interrupt source. These are the upper 8 interrupt sources.

**IRQ\_STATUS\_L**

Address: Operational Base + offset( 0x20)

Interrupt Status (Low) Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Interrupt status after the forcing and interrupt enabling stage. These are the interrupt status signals for the lower 32 interrupt sources.

**IRQ\_STATUS\_H**

Address: Operational Base + offset( 0x24)

Interrupt Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Interrupt status after the forcing and interrupt enabling stage. These are the interrupt status signals for the upper 8 interrupt sources.

**IRQ\_MASKSTATUS\_L**

Address: Operational Base + offset( 0x28)

Interrupt Mask Status (Low) Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Interrupt status after the masking stage. These are the interrupt status signals for the lower 32 interrupt sources.

**IRQ\_MASKSTATUS\_H**

Address: Operational Base + offset( 0x2c)

Interrupt Mask Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Interrupt status after the masking stage. These are the interrupt status signals for the upper 8 interrupt sources.

**IRQ\_FINALSTATUS\_L**

Address: Operational Base + offset( 0x30)

Interrupt Final Status (Low) Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Interrupt status after the priority level filtering stage. These are the interrupt status signals for the lower 32 interrupt sources.

**IRQ\_FINALSTATUS\_H**

Address: Operational Base + offset( 0x34)

Interrupt Final Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Interrupt status after the priority level filtering stage. These are the interrupt status signals for the upper 8 interrupt sources.

**FIQ\_INTEN**

Address: Operational Base + offset( 0xc0)

Fast Interrupt Enable Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Fast interrupt enable bits. A 1 in any bit position enables the corresponding interrupt. 0: disable interrupt 1: enable interrupt

**FIQ\_INTMASK**

Address: Operational Base + offset( 0xc4)

Fast Interrupt Mask Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Fast interrupt mask bits. A 1 in any bit position masks the corresponding interrupt. This register does not exist when ICT_HAS_FIQ = 0. 0: unmask interrupt 1: mask interrupt

**FIQ\_INTFORCE**

Address: Operational Base + offset( 0xc8)

Fast Interrupt Force Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Fast interrupt force bits. Each bit in this register corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register correspond to the polarity of the associated fiq_intsrc input. If the interrupt input is configured to be active high, the corresponding bit in the register is also active high. 0: active low

			1: active high
--	--	--	----------------

**FIQ\_RAWSTATUS**

Address: Operational Base + offset( 0xcc)

Fast Interrupt Source Raw Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	Fast interrupt source raw input status..

**FIQ\_STATUS**

Address: Operational Base + offset( 0xd0)

Fast Interrupt Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	Fast interrupt status after the forcing and interrupt enabling stage. 1: active 0: inactive

**FIQ\_FINALSTATUS**

Address: Operational Base + offset( 0xd4)

Fast Interrupt Final Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	Fast interrupt status after the masking stage. 1: active 0: inactive

**IRQ\_PLEVEL**

Address: Operational Base + offset( 0xd8)

IRQ System Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x0	Interrupt controller system priority level for normal interrupt sources. The default state can be configured so that after reset the interrupt controller will accept only interrupts that are enabled and have a priority the same or greater than the system level priority setting.

**IRQ\_PN\_OFFSET**

Address: Operational Base + offset(0xe8 + 4 \* n)

IRQ Individual Interrupt Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	n	Individual interrupt priority level. The range of N or n (number of registers) is from 0 to 15. A register's value must be an integer from 0x0 to 0xf.

**IRQ\_PN\_OFFSET**

Address: Operational Base + offset(0xe8 + 4 \* n)

IRQ Individual Interrupt Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	n-16	Individual interrupt priority level. The range of N or n (number of registers) is from 16 to 31. A register's



		value must be an integer from 0x0 to 0xf.
--	--	-------------------------------------------

**irq\_pN\_offset**

Address: Operational Base + offset(0xe8 + 4 \* n)

IRQ Individual Interrupt Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	n-32	Individual interrupt priority level. The range of N or n (number of registers) is from 32 to 39. A register's value must be an integer from 0x0 to 0xf.

**AHB\_ICTL\_COMP\_VERSION**

Address: Operational Base + offset(0x3f8)

Component Version Register

bit	Attr	Reset Value	Description
31:0	R	0x3230342a	Specific values for this register are described in the Releases Table

**ICTL\_COMP\_TYPE**

Address: Operational Base + offset(0x3fc)

Component Type Register

bit	Attr	Reset Value	Description
31:0	R	0x44571120	Type number = 0x44_57_11_20

## 11.4 Functional Description

### 11.4.1 Overview

The INTC supports from two to 40 normal interrupt (IRQ) sources that are processed to produce a single IRQ interrupt to the processor. It supports from one to 2 fast interrupt (FIQ) sources that are processed to produce a single FIQ interrupt to the processor. All interrupt processing is combinational so that interrupts are propagated if the bus interface of the INTC is powered down. This means that reading any of the interrupt status registers (raw, status, or final\_status) is simply returning the status of the combinational logic, since there are no flip-flops associated with these registers. It is the user's responsibility to ensure that the interrupts stay asserted until they are serviced

### 11.4.2 Detail Description

#### IRQ Interrupt Processing

The INTC processes these interrupt sources to produce a single IRQ interrupt to the processor; irq or irq\_n. The processing of the interrupt sources is shown as followed

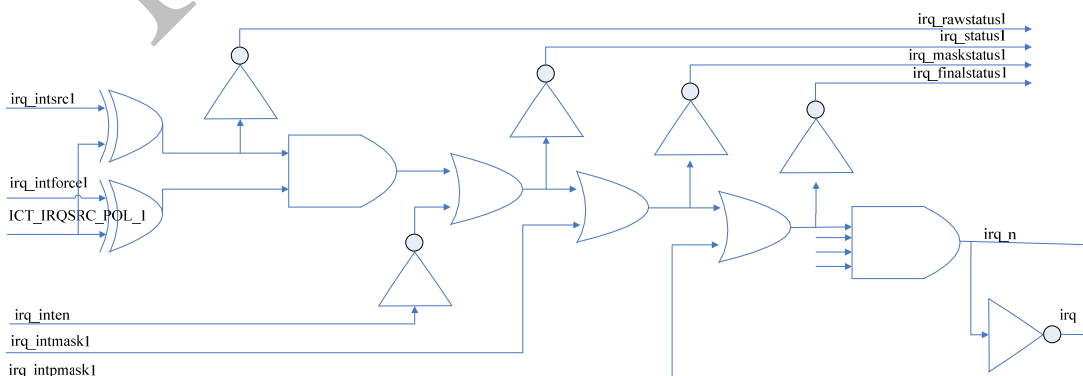


Fig. 11-2 RK281x IRQ Interrupt Processing for INTC

## IRQ Software-Programmable Interrupts

The INTC supports forcing interrupts from software. To force an interrupt to be active, write to the corresponding bit in the irq\_intforce registers (IRQ\_INTFORCE\_L or IRQ\_INTFORCE\_H).

## IRQ Enable and Masking

To enable each interrupt source independently, write a 1 to the corresponding bit of the irq\_inten registers (IRQ\_INTEN\_L or IRQ\_INTEN\_H).

To mask each interrupt source independently, write a 1 to the corresponding bit of the interrupt mask register (IRQ\_MASKSTATUS\_L/IRQ\_MASKSTATUS\_H). The reset value for each mask bit is 0 (unmasked)

## IRQ Software-Programmable Priority Levels

The INTC supports optional software programmable priority levels. To change the priority level of an interrupt, you write the priority value to the corresponding priority level register in the memory map. There is a priority register for each of the interrupt sources, which can be programmed to one of 16 values from 0x0 to 0xf. Priority registers only exist for available interrupt sources.

## IRQ Priority Filter

The INTC supports optional priority filtering. The function of the priority filtering logic is described as follows:

- Each interrupt source is configured to one of 16 priority levels. where 0 is the lowest priority.
- A system priority level can be programmed into the IRQ\_PLEVEL\_REGISTER, which holds values from 0 to 15.
- The INTC filters out any interrupt source with a configured priority level less than the priority currently programmed in the irq\_plevel register

## IRQ Interrupt Status Registers

The INTC includes up to four status registers used for querying the current status of any interrupt at various stages of the processing. All of the following status registers have the same polarity; a 1 indicates that an interrupt is active, a 0 indicates it is inactive:

- irq\_rawstatus

The irq\_rawstatus register (IRQ\_RAWSTATUS\_L/IRQ\_RAWSTATUS\_H) contains the state of the interrupt sources after being adjusted for input polarity. Each bit of this register is set to 1 if the corresponding interrupt source bit is active and is set to 0 if it is inactive.

- irq\_status

The irq\_status register (IRQ\_STATUS\_L/IRQ\_STATUS\_H) contains the state of all interrupts after the enabling stage, meaning that an active-high bit indicates that particular interrupt source is active and enabled.

- irq\_maskstatus

The irq\_maskstatus register (IRQ\_MASKSTATUS\_L/IRQ\_MASKSTATUS\_H) contains the state of all interrupts after the masking stage, meaning that an active-high bit indicates that particular interrupt source is active, enabled, and not masked.

- irq\_finalstatus

This register (IRQ\_FINALSTATUS\_L/IRQ\_FINALSTATUS\_H) contains the state of all interrupts after the priority filtering stage, meaning an active-high bit indicates that particular interrupt source is active, enabled, not masked, and its configured priority level is greater or equal to the value programmed in the irq\_plevel register. If priority filtering has not been selected, this register will contain the same value as the irq\_maskstatus register (the final stage of processing).

## FIQ Interrupt Processing

FIQ interrupt processing is similar to IRQ interrupt processing, except that priority filtering and interrupt vectors are not supported for the FIQ interrupts. This section describes how the INTC handles the FIQ interrupt processing. the processing of the interrupt sources is described as followed.

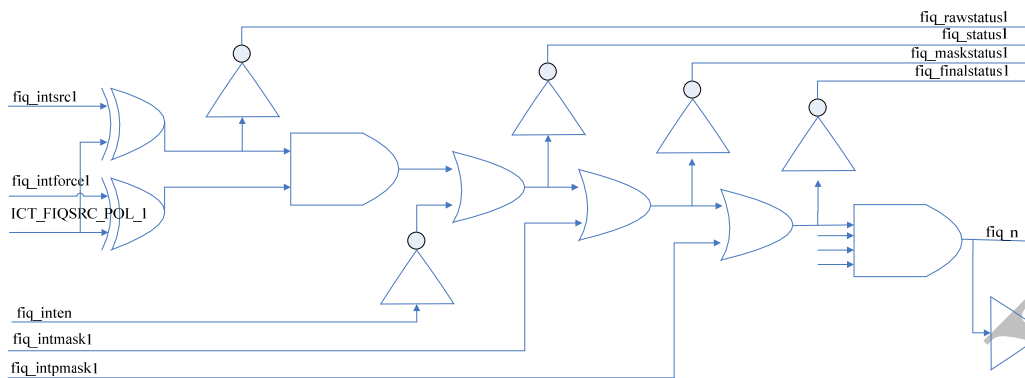


Fig. 11-3 RK281x FIQ Interrupt Processing for INTC

### FIQ Software-Programmable Interrupts

The INTC supports forcing interrupts from software. You force an interrupt to be active by writing to the corresponding bit in the **FIQ\_INTFORCE** register. The polarity of each bit in this register is the same as the polarity of the corresponding interrupt source signal.

### FIQ Enable and Masking

You can enable each interrupt source independently by writing a 1 to the corresponding bit of the **FIQ\_INTEN** register.

You can mask each interrupt source independently by writing a 1 to the corresponding bit of the **FIQ\_INTMASK** register. The reset value for each mask bit is 0; that is, unmasked.

### FIQ Interrupt Status Registers

The INTC includes three status registers that you can use to query the current status of any FIQ interrupt at various stages of the processing:

- **fiq\_rawstatus**

The **fiq\_rawstatus** register contains the state of the interrupt sources after being adjusted for input polarity. Each bit of this register is set to 1 if the corresponding interrupt source bit is active and is set to 0 if it is inactive.

- **fiq\_status**

The **fiq\_status** register contains the state of all interrupts after the enabling stage, meaning that an active-high bit indicates that particular interrupt source is active and enabled.

- **fiq\_finalstatus**

The **fiq\_finalstatus** register contains the state of all interrupts after the masking, meaning that an active-high bit indicates that particular interrupt source is active, enabled, and unmasked.

## Chapter 12 High-Speed ADC Interface

### 12.1 Design Overview

HS-ADC Interface Unit is interface unit of connected the High Speed AD Converter to AMBA AHB bus. That implement bus speed convert at low speed AD Converter bus to high speed AHB bus. HS-ADC Interface Unit fetch the bus data by the AD converter and store that to asynchronous FIFO after the AD clock is active when OS configure completion by DMA and HS-ADC Interface Unit. The HS-ADC Interface Unit generates the DMA request signal When data length of the asynchronous FIFO over then almost full level or almost empty level.

#### Key features

- Support the burst transfers and that type include SINGLE, INCR4, INCR8, INCR16.
- Support HS-ADC Interface Unit Enable and Disable. Notice that controller register can be modified when HS-ADC Interface Unit Disabled.
- Support 8-bit/10-bit data bus by the AD converter.
- Support two channel 8-bit/10-bit data input
- Support the most significant bit negation.
- Support store to high 8-bit/10-bit or low 8-bit/10-bit at a haft word width. Sample the 8-bit data by the AD converter store to high 8-bit is between the data[15] to data[8] by a haft word width. And that have sign extend if store to low 8-bit/10-bit by a haft word width.
- Support 2-bit GPS data input
- Support MPEG transport stream data input
- Support DMA transfers mode and that generate DMA request from the event of almost full or almost empty in the asynchronous FIFO. The almost full/almost empty level can be configuration.

### 12.2 Architecture

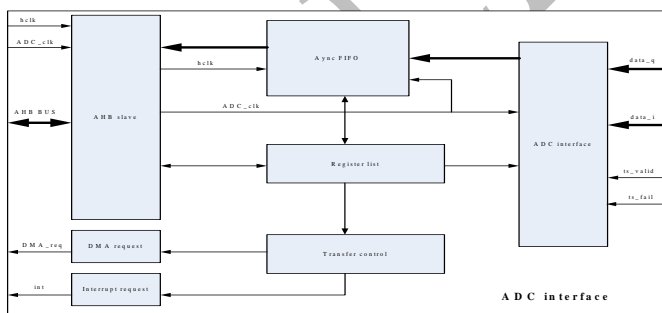


Fig. 12-1 RK281x HS-ADC architecture diagram

### 12.3 Register

#### 12.3.1 Register Summary

Name	Offset	Width	Access	Reset Value	Description
HSADC_CTRL	0x00	64	R/W	0x0000_0000	control register
HSADC_IER	0x08	64	R/W	0x0000_0000	interrupt enable/mask register
HSADC_ISR	0x10	64	R/W	0x0000_0000	interrupt status register

HSADC_TS_FAIL	0x18	64	R	0x0000_0000	ts stream fail signal indication
HSADC_DATA	0x20	64	R	--	data register

### 12.3.2 Detail Register Description

#### HSADC\_CTRL

Address: Operational Base + offset(0x00)

The controls register of ADC interface

Bit#	Access	Reset value	Description
31:28			Reserved
27:24	R/W	0	Define almost full trigger level: 0x0~"0xf" - configure valid range (Notes: 1 level indicate 4 entries data in the async FIFO. and this configure range mapping to 64 - 124 entries data in the async FIFO.)
23:20			Reserved
19:16	R/W	0	Define almost empty trigger level: 0x0~"0xf" - configure valid range (Notes: 1 level indicate 4 entries data in the async FIFO. and this configure range mapping to 0 - 60 entries data in the async FIFO.)
15:9			Reserved
8	R/W	0	MPEG transport stream data input select 0: not select MPEG transport stream 1: select MPEG transport steam data input
7	R/W	0	data input channel select 0 : single channel, ADC input from adc_data_i 1 : double channel data input (must select ADC input)
6	R/W	0	sample data rate select 0 : sample data rate is 30MHz 1 : sample data rate is 60MHz , when use two channel AD data input,this bit must set high
5	R/W	0	DMA request mode: 1 - almost full generate DMA request signal (Notes: this mode generate DMA request signal from almost full condition and cancel DMA request signal from almost empty condition. so you need configure two level by almost full level and almost empty level) 0 - almost empty generate DMA request signal (Notes: this mode generate DMA request signal from almost empty condition and that only once DMA request.)
4	R/W	0	control the most significant bit negation: "1" - negation "0" - not negation
3	R/W	0	fetch the bus data by AD converter and that store to high 8-bit/10-bit or low 8-bit/10-bit at a haft word width before push to Async FIFO : "1" - store to high 8-bit/10-bit "0" - store to low 8-bit/10-bit (Notes: have sign extend if that configure of store to low 8-bit/10-bit)

2	R/W	0	The data bus width of AD converter : "1" - 10-bit "0" - 8-bit
1	R/W	0	GPS data input select 0 : ADC input 1 : GPS data input adc_data_i[1:0]
0	R/W	0	HS-ADC Interface Unit Enable Bit: "1" - enable (Notes: will return 1 when the hardware started transfer) "0" - disable (Notes: other bit can be modify only the hardware return 0)

**HSADC\_IER**

Address: Operational Base + offset(0x08)

The interrupt enable control register of ADC interface.

Bit	Access	Reset value	Description
31:2			
1	R/W	0	Interrupt enable/mask bit for enable/disable the empty interrupt flag of Async FIFO "1" - enable "0" - disable
0	R/W	0	Interrupt enable/mask bit for enable/disable the full interrupt flag of Async FIFO "1" - enable "0" - disable

**HSADC\_ISR**

Address: Operational Base + offset(0x10)

The interrupt statuses register of ADC interface.

Bit	Access	Reset value	Description
31:2			
1	RW	0	Async FIFO empty interrupt flag. "1(R)" - This bit will be set to "1" when Async FIFO empty status and that only to read operation. "0(W)" - Write "0" to bit for clear the interrupt flag and that only to write operation.
0	RW	0	Async FIFO full interrupt flag. "1(R)" - This bit will be set to "1" when Async FIFO full status and that only to read operation. "0(W)" - Write "0" to bit for clear the interrupt flag and that only to write operation.

**HSADC\_TS\_FAIL**

Address: Operational Base + offset(0x18)

MPEG transport stream packet fail indication.

Bit#	Access	Reset value	Description
31:2			
0	R	0	TS stream fail indication, this signal only valid when select TS stream input. Clean by read 0 : TS stream decode successfully 1: TS stream decode fail

**HSADC\_DATA**

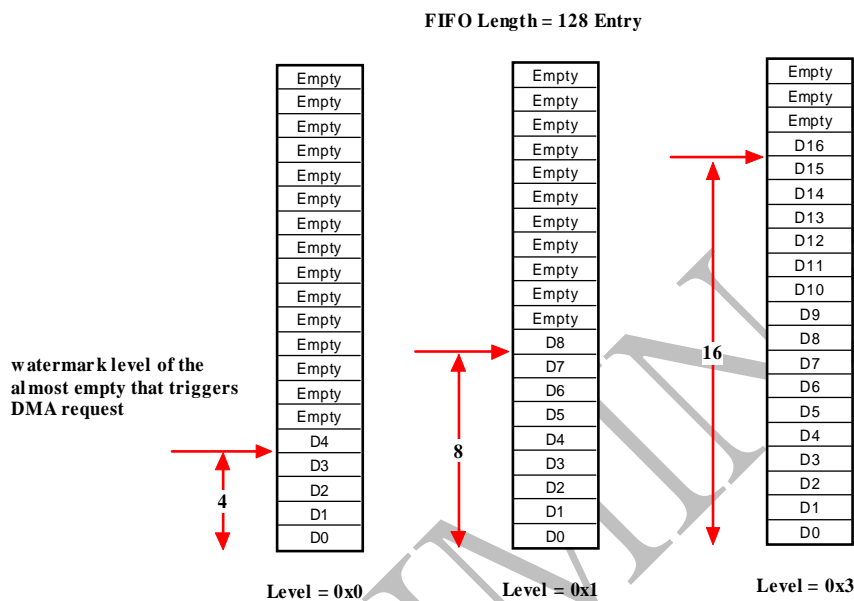
Address: Operational Base + offset(0x20)

The data register of ADC interface.

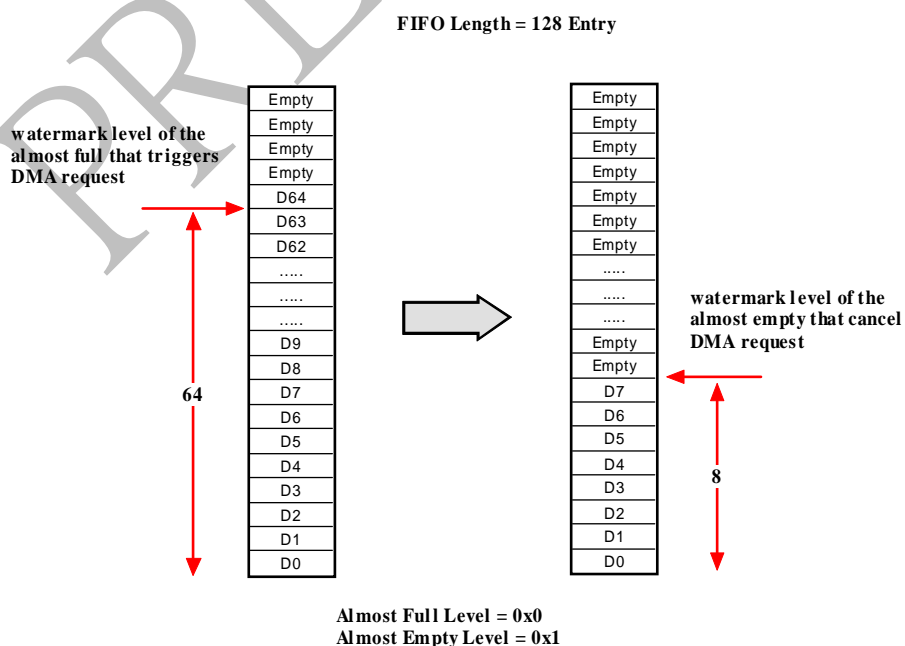
Bit	Access	Reset value	Description
63:0	R	--	the data of async FIFO for the DMA read.

**12.4 Application Notes**

The following sections will describe the operation of DMA request and DMA transfers:  
**Almost empty triggers a DMA request by DMA request mode**



The DMA request signal will generate from a watermark level triggers when store to FIFO data over the watermark level of almost empty and that watermark level can be configuration to HSADC\_CTRL[19:16] by controls register . This DMA request mod doesn't care the watermark level of almost full.

**Almost full triggers a DMA request by DMA request mode**



The DMA request signal will generate from a watermark level triggers when store to FIFO data over the watermark level of almost full and at this DMA request mode that continue generates request signal when the number of FIFO data great then watermark level of almost empty. This DMA request mode need configuration double watermark level and that is watermark level of almost empty at the HSADC\_CTRL[19:16] and watermark level of almost full at the HSADC\_CTRL[27:24].

PRELIMINARY

## Chapter 13 Host Interface (HIF)

### 13.1 Design Overview

#### 13.1.1 Overview

Host Interface(HIF) will focus on high-speed data transfer between RK281x and Modem chip. There is a 4KB size dual-port SRAM buffer, which can be used to complete data exchange by interactive interrupt for each other.

Another, HIF function can be disabled by software, and buffer will become share memory between CPU and DSP.

#### 13.1.2 Features

- 8bits / 16 bits parallel bus for data transfer, it is programmable
- Configurable MCU interface signal valid polarity
- 4KB internal Dual Port SRAM buffer
- Interrupt request for data exchange
- Support HIF function disable
- Two AHB slave interface for memory share of two processors
- MCU interface to communicate between RK281x and Modem chip
- Support address self-increment for burst transfer when accessing buffer by MCU interface
- Support LCDC interface bypass from HIF interface

### 13.2 Architecture

#### 13.2.1 Block Diagram

The following diagram illustrates the block diagram for HIF module.

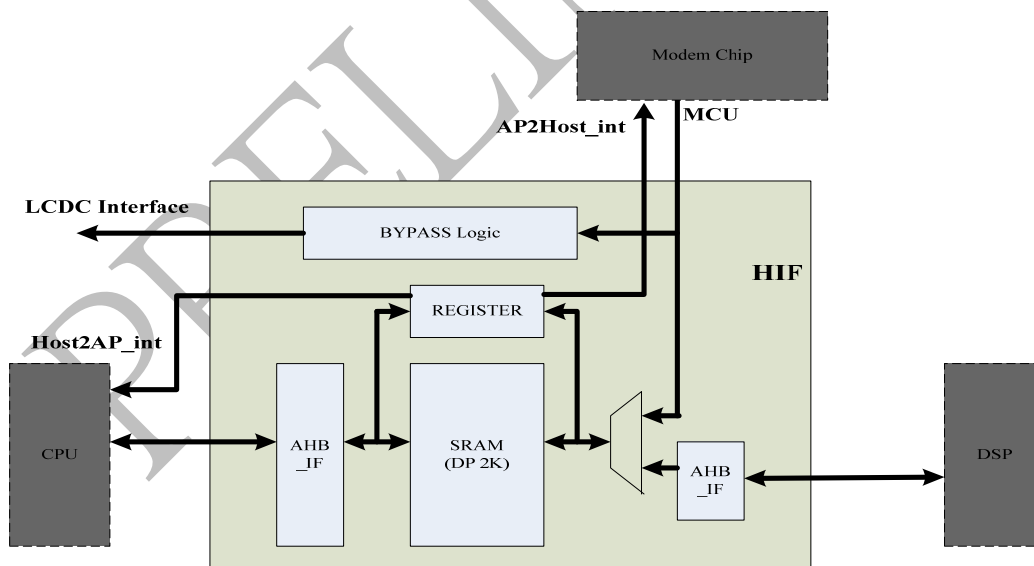


Fig. 13-1 RK281x HIF block diagrams

### 13.3 Registers

This section describe the registers of the HIF.

### 13.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
HIF_CON	0x00	b	0x0	HIF Control register
HIF_INITCOUNT	0x04	hw	0x0	HIF Initial Transfer Count register
HIF_INITADDR	0x08	hw	0x0	HIF Initial Address register
HIF_ADDR	0x0c	hw	0x0	HIF real Address register, only read by AP
HIF_COUNT	0x10	hw	0x0	HIF real Transfer Count register, only read by AP

### 13.3.2 Detail Register Description

#### HIF\_CON

Address : Base Addr+0x00

Hif control register

bit	Attr	Reset Value	Description
7:6	R/W	0x0	Reserved
5	R/W	0x0	Modem chip to RK281x interrupt , driven by Modem chip , only write by Modem chip
4	R/W	0x0	RK281x to Modem chip interrupt , driven by RK281x , only write by RK281x
3	R/W	0x0	Valid level select for MCU interface signal 0 : low level 1 : high level Only write by RK281x
2	R/W	0x0	Register access select 0: HIF_ADDR register 1: HIF_COUNT register
1	R/W	0x0	Byte Control Bit for count register: 0 : low byte 1 : high byte Modem chip must set this bit if necessary before every access, and do automatic refresh, only write by modem chip
0	R/W	0x0	Byte Control Bit for addr register: 0 : low byte 1 : high byte Modem chip must set this bit if necessary before every access, and do automatic refresh, only write by modem chip

#### HIF\_INITCOUNT

Address : Base Addr+0x04

Hif transfer data byte register

bit	Attr	Reset Value	Description
15:0	R/W	0x0	Total byte numbers for data transfered

#### HIF\_INITADDR

Address : Base Addr+0x08

Hif start address register

bit	Attr	Reset Value	Description
15:0	RW	0x0	Initial Address value for modem chip accessing RK281x SRAM

#### HIF\_ADDR

Address : Base Addr+0x0c

Hif address register

bit	Attr	Reset Value	Description
15:0	R	0x0	Access Address value for Modem chip accessing RK281x SRAM, updated during accessing, only read by RK281x

#### HIF\_COUNT

Address : Base Addr+0x10

Hif transfer byte register

bit	Attr	Reset Value	Description
15:0	R	0x0	real rest byte numbers for data transferred, updated during accessing, only read by RK281x

## 13.4 Application Notes

### ■ Host interface address map

- ◆ Modem chip can read/write registers and buffer inside RK281x HIF module
- ◆ Modem chip can access buffer with single mode or continuous burst mode
- ◆ The above access type is decided by host\_addr[1:0] from host interface, the detailed address map is shown as follows :

Table 13-1 Host interface address map table

host_addr[1:0]	Access type
2'b00	HIF_CON register
2'b01	HIF_INITCOUNT or HIF_INITADDR register
2'b10	4K SRAM with single mode
2'b11	4K SRAM with continuous burst mode, Internal address can be self-increased

### ■ Host interface function

- ◆ HIF support 8bits (default) and 16bits data bus width, which can be programmable by bit 26 in CPU\_APB\_REG4. Refer to Chapter 34 (General Register File in CPU System) for detailed information.
- ◆ It must have at least 2 cycles hclk interval between read and write operation of host interface

### ■ Share Memory function

HIF function can be disabled (default) by software set. Refer to bit 27 of CPU\_APB\_REG4 in Chapter 34 (General Register File in CPU System) for detailed

information. After that, the 4KB size buffer will be used for share memory between DSP and CPU.

#### ■ LCD interface bypass function

HIF interface can be used to bypass to LCDC interface of RK281x, then modem chip will control LCD panel by HIF interface, which is controlled by RK281x before. Remind that the panel type for this application scene is only MCU panel. As for the detailed information, please refer to bit 25 of CPU\_APB\_REG4 in Chapter 34. In fact, RK281x LCDC bypass function is controlled by IO pin named as IO\_LcdbP, and also can be masked by software (bit 25 of register CPU\_APB\_REG4). In other words, it is controlled by both hardware and software. RK281x can support 4 different RGB component when bypass function is enabled, which is programmable by bit[15:14] of register CPU\_APB\_REG5.

The following table will list pin mapping between HIF and LCDC interface.

Table 13-2 Pin mapping between HIF and LCDC interface

- When CPU\_APB\_REG5[15:14] = 2'b00, the bypass pin relationship is as follows:

HIF Pin Name	HIF port name	LCDC port name	LCDC Pin Name
IO_GPIO0_A[0]	host_data16	lcdc_wdata[0]	IO_LCDC_DATA[0]
IO_GPIO2[6:0]	host_wdata[6:0]	lcdc_wdata[7:1]	IO_LCDC_DATA[7:1]
IO_GPIO2[7]	host_wdata[7]	lcdc_wdata[8]	IO_GPIO0_D[0]
IO_GPIO2[16:14]	host_wdata[10:8]	lcdc_wdata[11:9]	IO_GPIO0_D[3:1]
IO_GPIO0_A[1]	host_data17	lcdc_wdata[12]	IO_GPIO0_D[4]
IO_GPIO2[19:17]	host_wdata[13:11]	lcdc_wdata[15:13]	IO_GPIO0_D[7:5]
IO_GPIO2[20]	host_wdata[14]	lcdc_wdata[16]	IO_GPIO0_C[0]
IO_GPIO2[21]	host_wdata[15]	lcdc_wdata[17]	IO_GPIO0_C[1]
IO_GPIO2[10]	host_csn	lcdc_vsync/lcdc_csn	IO_GPIO2[25]
IO_GPIO2[12]	host_wrn	lcdc_hsync/lcdc_wen	IO_LCDC_HSYNC
IO_GPIO2[8]	host_addr[0]	lcdc_dclk /lcdc_rs	IO_LCDC_DCLK

- When CPU\_APB\_REG5[15:14] = 2'b01, the bypass pin relationship is as follows:

HIF Pin Name	HIF port name	LCDC port name	LCDC Pin Name
IO_GPIO2[7:0]	host_wdata[7:0]	lcdc_wdata[7:0]	IO_LCDC_DATA[7:0]
IO_GPIO2[21:14]	host_wdata[15:8]	lcdc_wdata[15:8]	IO_GPIO0_D[7:0]
IO_GPIO0_A[0]	host_data16	lcdc_wdata[16]	IO_GPIO0_C[0]
IO_GPIO0_A[1]	host_data17	lcdc_wdata[17]	IO_GPIO0_C[1]
IO_GPIO2[10]	host_csn	lcdc_vsync/lcdc_csn	IO_GPIO2[25]
IO_GPIO2[12]	host_wrn	lcdc_hsync/lcdc_wen	IO_LCDC_HSYNC
IO_GPIO2[8]	host_addr[0]	lcdc_dclk /lcdc_rs	IO_LCDC_DCLK

- When CPU\_APB\_REG5[15:14] = 2'b10, the bypass pin relationship is as follows:

HIF Pin Name	HIF port name	LCDC port name	LCDC Pin Name
IO_GPIO0_A[0]	host_data16	lcdc_wdata[0]	IO_LCDC_DATA[0]
IO_GPIO0_A[1]	host_data17	lcdc_wdata[1]	IO_LCDC_DATA[1]
IO_GPIO2[5:0]	host_wdata[5:0]	lcdc_wdata[7:2]	IO_LCDC_DATA[7:2]

IO_GPIO2[7:6]	host_wdata[7:6]	lcdc_wdata[9:8]	IO_GPIO0_D[1:0]
IO_GPIO2[19:14]	host_wdata[13:8]	lcdc_wdata[15:10]	IO_GPIO0_D[7:2]
IO_GPIO2[20]	host_wdata[14]	lcdc_wdata[16]	IO_GPIO0_C[0]
IO_GPIO2[21]	host_wdata[15]	lcdc_wdata[17]	IO_GPIO0_C[1]
IO_GPIO2[10]	host_csn	lcdc_vsync/lcdc_csn	IO_GPIO2[25]
IO_GPIO2[12]	host_wrn	lcdc_hsync/lcdc_wen	IO_LCDC_HSYNC
IO_GPIO2[8]	host_addr[0]	lcdc_dclk /lcdc_rs	IO_LCDC_DCLK

- When CPU\_APB\_REG5[15:14] = 2'b11, the bypass pin relationship is as follows :

HIF Pin Name	HIF port name	LCDC port name	LCDC Pin Name
IO_GPIO0_A[0]	host_data16	lcdc_wdata[0]	IO_LCDC_DATA[0]
IO_GPIO2[6:0]	host_wdata[6:0]	lcdc_wdata[7:1]	IO_LCDC_DATA[7:1]
IO_GPIO2[7]	host_wdata[7]	lcdc_wdata[8]	IO_GPIO0_D[0]
IO_GPIO0_A[1]	host_data17	lcdc_wdata[9]	IO_GPIO0_D[1]
IO_GPIO2[19:14]	host_wdata[13:8]	lcdc_wdata[15:10]	IO_GPIO0_D[7:2]
IO_GPIO2[20]	host_wdata[14]	lcdc_wdata[16]	IO_GPIO0_C[0]
IO_GPIO2[21]	host_wdata[15]	lcdc_wdata[17]	IO_GPIO0_C[1]
IO_GPIO2[10]	host_csn	lcdc_vsync/lcdc_csn	IO_GPIO2[25]
IO_GPIO2[12]	host_wrn	lcdc_hsync/lcdc_wen	IO_LCDC_HSYNC
IO_GPIO2[8]	host_addr[0]	lcdc_dclk /lcdc_rs	IO_LCDC_DCLK

### ■ Host interface timing requirement

The following waveform has shown the requirement for host interface timing. In them, the T is period for AHB bus clock inside RK281x.

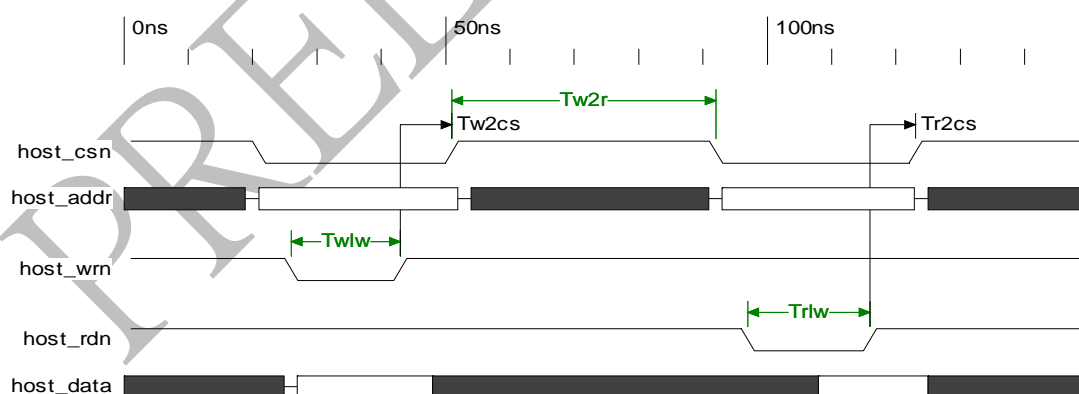


Fig. 13-2 RK281x Timing Diagram for host interface

Symbol	Description	Min	Max
Twlw	Write low width	2T	N/A
Trlw	Read low width	3T	N/A
Tw2cs	width from wrn invalid to csn invalid	T	N/A

Tr2cs	width from rdn invalid to csn invalid	T	N/A
Tw2r	interval width between read and write	3T	N/A

PRELIMINARY



## Chapter 14 USB OTG Controller

### 14.1 Design Overview

#### 14.1.1 Overview

USB OTG Controller is a Dual-Role Device controller, which supports both device and host functions and is fully compliant with OTG Supplement to USB2.0 specification, and support high-speed(480Mbps),full-speed(12Mbps),low-speed(1.5Mbps) transfer. This controller will support UTMI+ Level 3 PHY interface. It connects to the industry-standard AMBA AHB for communication with the application and system memory. And it is optimized for portable electronic devices , point-to-point applications(no hub, direct connection to device) and multi-point applications to devices.

#### 14.1.2 Features

- Compliant with the OTG Supplement to the USB2.0 Specification
- Operates in High-Speed and Full-Speed mode
- Supports UTMI+ Level 3 interface , and 16bit data bus will be used.
- Support Session Request Protocol(SRP) and Host Negotiation Protocol(HNP)
- Support 6 channels in host mode
- 6 Device mode endpoints in addition to control endpoint 0 , 3 in and 3 out
- Built-in one 1777 x 35bits FIFO
- Internal DMA with scatter/gather function
- Supports packet-based, dynamic FIFO memory allocation for endpoints for flexible,efficient use of RAM
- Provides support to change an endpoint`s FIFO memory size during transfers

For detailed information about USB OTG controller, please refer to **RK281x USB OTG Controller.pdf**.

## Chapter 15 System Control Unit (SCU)

### 15.1 Design Overview

The SCU is an APB slave module that is designed for generating all of the internal and system clocks, resets of chip and power domain control. SCU generates system clock from PLL output clock or external clock source , and generate system reset from external power-on-reset or watchdog timer reset. The SCU also provide power management mechanism for system power saving and general control bit.

#### Key Features

- Compliance to the AMBA APB interface
- Centralization of clock and reset sources control
- Five power management mode --- normal , slow , idle , stop, power off
- General peripheral control bit and chip status record
- Power domain control

### 15.2 Registers

This section describes the control/status registers of the design.

#### 15.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SCU_APLL_CON	0x0000	w	0x01850412	ARM PLL output frequency control register (133MHz)
SCU_DPLL_CON	0x0004	w	0x01830310	DSP PLL output frequency control register (300MHz)
SCU_CPLL_CON	0x0008	w	0x01980ff2	CODEC PLL output frequency control register (122.88MHz)
SCU_MODE_CON	0x000c	w	0x00700207	System work mode control register
SCU_PMU_CON	0x0010	w	0x00000000	Power management control register
SCU_CLKSEL0_CON	0x0014	w	0x00000734	Clock divider frequency and select control register
SCU_CLKSEL1_CON	0x0018	w	0x0003000c	Clock divider frequency and select control register
SCU_CLKGATE0_CON	0x001c	w	0x00000000	Clock gating control register
SCU_CLKGATE1_CON	0x0020	w	0x00000000	Clock gating control register
SCU_CLKGATE2_CON	0x0024	w	0x00000000	Clock gating control register
SCU_SOFT_RST_CON	0x0028	w	0x00000010	Soft reset control register
SCU_CHIPCFG_CON	0x002c	w	0x0bb80000	Chip general configuration register
SCU_CPUPD	0x0030	w	0x00000000	ARM926E Power down control register
SCU_CLKSEL2_CON	0x0034	W	0x00000330	Clock divider frequency

				and select control register
--	--	--	--	-----------------------------

## 15.2.2 Detail Register Description

### SCU\_APLL\_CON

Address : Base Addr+0x00

ARM PLL configuration register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved
25	RW	0x0	ARM PLL test control 0 : normal 1 : test
24	RW	0x1	ARM PLL saturation behavior enable 0: disable 1 : enable
23	RW	0x1	ARM PLL Enables fast locking circuit 0 : disable 1 : enable
22	RW	0x0	ARM PLL Power down control 1: powerdown
21:16	RW	0x5	ARM PLL CLKR factor control
15:4	RW	0x41	ARM PLL CLKF factor control
3:1	RW	0x1	ARM PLL CLKOD factor control
0	RW	0x0	ARM PLL Bypass mode control 1: bypass 0: no bypass

### SCU\_DPLL\_CON

Address : Base Addr+0x04

DSP PLL configuration register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved
25	RW	0x0	DSP PLL test control 0 : normal 1 : test
24	RW	0x1	DSP PLL saturation behavior enable 0: disable 1 : enable
23	RW	0x1	DSP PLL Enables fast locking circuit 0 : disable 1 : enable
22	RW	0x0	DSP PLL Power down control
21:16	RW	0x3	DSP PLL CLKR factor control
15:4	RW	0x31	DSP PLL CLKF factor control
3:1	RW	0x0	DSP PLL CLKOD factor control
0	RW	0x0	DSP PLL Bypass mode control

### SCU\_CPLL\_CON

Address : Base Addr+0x08

CODEC PLL configuration register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved

25	RW	0x0	CODEC PLL test control 0 : normal 1 : test
24	RW	0x1	CODEC PLL saturation behavior enable 0: disable 1 : enable
23	RW	0x1	CODEC PLL Enables fast locking circuit 0 : disable 1 : enable
22	RW	0x0	CODEC PLL Power down control
21:16	RW	0x18	CODEC PLL CLKR factor control
15:4	RW	0xff	CODEC PLL CLKF factor control
3:1	RW	0x1	CODEC PLL CLKOD factor control
0	RW	0x0	CODEC PLL Bypass mode control

**SCU\_MODE\_CON**

Address : Base Addr+0x0c

System work mode control register

bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	SCU interrupt clear bit 0 pending 1 clear
7	RW	0x0	Wakeup pin type selection 0 positive polarity 1 negative polarity
6	RW	0x0	Disable RTC alarm or interrupt wakeup stop mode 0 Enable RTC alarm or interrupt wakeup 1 Disable RTC alarm or interrupt wakeup
5	RW	0x0	Disable external wakeup stop mode 0 Enable external wakeup pin 1 Disable external wakeup pin
4	RW	0x0	Stop mode enable 0: disable 1: stop mode
3:2	RW	0x00	CPU work mode 00: CPU subsys slow mode 01: Normal mode 10: CPU subsys deep slow mode 11: CPU subsys slow mode
1:0	RW	0x00	DSP work mode 00: DSP subsys slow mode 01: Normal mode 10: DSP subsys deep slow mode 11: DSP subsys slow mode

**SCU\_PMU\_MODE**

Address : Base Addr+0x10

Power domain control register

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	R	0x0	LCDC subsystem (Power Domain4) power on/off status 0 : power on

			1 : power off
7	R	0x0	DDR PHY (Power Domain3) power on/off status 0 : power on 1 : power off
6	R	0x0	CPU Subsys (Power Domain2) power on/off status 0 : power on 1 : power off
5	R	0x0	DSP Subsys (Power Domain1) power on/off status 0 : power on 1 : power off
4	RW	0x0	Enable CPU subsys(Power Domain2) power switch by external pin 0 Disable (default) 1 Enable
3	RW	0x0	Control LCDC subsystem (Power Domain4) Power down or on 0 : Power on (default) 1 : Power down
2	RW	0x0	Control DDR PHY (Power Domain3) Power down or on 0 : Power on (default) 1 : Power down
1	RW	0x0	Control CPU subsys (Power Domain2) Power down or on 0 : Power on (default) 1 : Power down
0	RW	0x0	Control DSP subsys (Power Domain1) Power down or on 0 : Power on (default) 1 : Power down

**SCU\_CLKSELO\_CON**

Address : Base Addr+0x14

Internal clock select and divide register0

bit	Attr	Reset Value	Description
31:30	RW	2'b0	DDR clock divide frequency select: 00 : divide 1 (default) 01 : divide 2 10 : divide 4 11 : divide 8
29:28	RW	2'b00	Control DDR clock source 00 : select codecpll_clk (default) 01 : select armppll_clk 10 : select dsppll_clk
27:25	RW	3'b00	Reserved
24:23	RW	2'b00	Select Sensor CLK 00: select 24MHz (default) 01 : select 27MHz 10 : select 48MHz
22:20	RW	3'b00	Reserved
19:18	RW	2'b00	Select USB PHY clk 00 : 24MHz (default) 01 : 12MHz 10 : 48MHz
17:16	RW	2'b00	Control LCDC CLK divider frequency source 00 : select armppll_clk (default)

			01 : select dsppll_clk 10 : select codecpll_clk
15:8	RW	8'b00000111	Control LCDDC CLK divider frequency value (pllclk/(1~128)) lcdclk = pllclk/(lcdclk_div_sel+1) default : pllclk/8
7	RW	1'b0	Select LCDDC CLK 0 : select divider output 1 : select 27MHz from external clock
6:4	RW	3'b011	Control sd/mmc0 clk frequency (hclk/(1~6)) mmc0_clk = arm_hclk/(mmc0clk_sel+1) default : arm_hclk/4
3:2	RW	2'b01	Control arm subsys pclk frequency 00 : hclk:pclk = 1:1 01 : hclk:pclk = 2:1 (default) 10 : hclk:pclk = 4:1
1:0	RW	2'b00	Control arm subsys hclk frequency 00 : armclk:hclk=1:1 (default) 01 : armclk:hclk=2:1 10 : armclk:hclk=3:1 11 : armclk:hclk=4:1

Notes : The detailed clock architecture, please refer to the following diagram.

### SCU\_CLKSEL1\_CON

Address : Base Addr+0x18

Internal clock select and divide register1

bit	Attr	Reset Value	Description
31	RW	1'b0	Select UART clk 0 : 24MHz(default) 1 : 48MHz
30	RW	1'b0	Reserved
29	RW	1'b0	Reserved
28	RW	1'b0	Select HS_ADC clock output 0 : select demod_clock/2 (default) 1 : select demod_clock/2 inverted
27	RW	1'b0	Select GPS tuner input clock or not to hsadc interface 0 : not from GPS input clock (default) 1 : from GPS tuner input
26	RW	1'b0	Select demodulator clk from external clock or not 0 : internal divider out (default) 1 : external clock input
25:24	RW	2'b00	Control demodulator CLK divider frequency source 00 : select codecpll_clk (default) 01 : select armpll_clk 10 : select dsppll_clk
23:16	RW	8'b00000011	Control demodulator CLK divider frequency (xppll_clk/(1~128)) demod_clk = xppll_clk/(demod_clk_divcon+1) default : xppll_clk/4
15:8	RW	8'b0	Control LS_ADC CLK divider frequency (pclk/(1~128)) ladcclk = pclk/(ladcclk_sel+1)
7:3	RW	5'b00001	Control CODECCLK divider frequency (cppll_clk/(1~32)) codeccclk = codecpll_clk/(codeccclk_sel+1) default : cppll_clk/2

2	RW	1'b1	codeclk12m_sel Control CODECLK work frequency 0 : select divider output from codec pll 1 : select 12MHz from osc input (default)
1:0	RW	2'b00	Codec PLL slow mode select 00 : slow mode, clock from external 24m osc (default) 01 : normal mode, clock from PLL 10 : deep slow mode Other:Reserved

**SCU\_CLKGATE0\_CON**

Address : Base Addr+0x1c

Internal clock gating control register0

Bit	Attr	Reset Value	Description
31	RW	0x0	SD/MMC1 clock and hclk disable. When HIGH,disable clock
30	RW	0x0	Uart3 clock disable. When HIGH,disable clock
29	RW	0x0	Uart2 clock disable. When HIGH,disable clock
28	RW	0x0	SAR-ADC clock and pclk disable. When HIGH,disable clock
27	RW	0x0	Rtc pclk disable. When HIGH,disable clock
26	RW	0x0	WDT pclk disable. When HIGH,disable clock
25	RW	0x0	Timer pclk disable. When HIGH,disable clock
24	RW	0x0	PWM pclk disable. When HIGH,disable clock
23	RW	0x0	Spi1(slave) clock disable. When HIGH,disable clock
22	RW	0x0	spi0(master) clock disable. When HIGH,disable clock
21	RW	0x0	I2c1 clock disable. When HIGH, disable clock
20	RW	0x0	i2c0 clock disable. When HIGH,disable clock
19	RW	0x0	uart1 clock disable. When HIGH,disable clock
18	RW	0x0	uart0 clock disable. When HIGH,disable clock
17	RW	0x0	Gpio1 clock disable. When HIGH,disable clock
16	RW	0x0	Gpio0 pclk disable. When HIGH,disable clock
15	RW	0x0	Embedded Rom clock disable. When HIGH,disable clock
14	RW	0x0	SD/MMC0 clock and hclk disable. When HIGH,disable clock
13	RW	0x0	i2s clock and pclk disable. When HIGH,disable clock
12	RW	0x0	viu clock and hclk disable. When HIGH,disable clock
11	RW	0x0	lcdc clock disable. When HIGH,disable clock
10	RW	0x0	Deblocking(RV) hclk clock disable. When HIGH,disable clock
9	RW	0x0	intc hclk clock disable. When HIGH,disable clock
8	RW	0x0	nandc hclk clock disable. When HIGH,disable clock
7	RW	0x0	usb otg phy clock disable. When HIGH,disable clock
6	RW	0x0	usb otg bus side clock disable. When HIGH,disable clock
5	RW	0x0	HIF&SRAM block hif clock disable. When HIGH,disable clock
4	RW	0x0	HIF&SRAM block dsp bus clock disable. When HIGH,disable clock
3	RW	0x0	HIF&SRAM block arm bus clock disable. When HIGH,disable clock
2	RW	0x0	dma clock disable. When HIGH,disable clock



1	RW	0x0	dsp clock disable. When HIGH,disable clock
0	RW	0x0	arm core clock disable . When HIGH,disable clock

**SCU\_CLKGATE1\_CON**

Address : Base Addr+0x20

Internal clock gating control register1

Bit	Attr	Reset Value	Description
31	RW	0x0	USB Host clock disable. When HIGH,disable clock
30	RW	0x0	DSP master interface bridge clock disable. When HIGH,disable clock
29	RW	0x0	DSP Slave interface bridge clock disable. When HIGH,disable clock
28	RW	0x0	DSP timer clock disable. When HIGH,disable clock
27	RW	0x0	DDR AXI bus clock disable. When LOW,disable clock
26	RW	0x0	SDRAM clock out disable. When HIGH,disable clock
25	RW	0x0	MCDMA clock disable. When HIGH,disable clock
24	RW	0x0	Customized SDRAM Controller HCLK clock disable. When HIGH,disable clock
23	RW	0x0	DDR clock disable. When HIGH,disable clock
22	RW	0x0	DDR HCLK clock disable. When HIGH,disable clock
21	RW	0x0	GPU clock disable. When HIGH,disable clock
20	RW	0x0	Deblocking(H.264) hclk clock disable. When HIGH,disable clock
19	RW	0x0	LCDC hclk clock disable. When HIGH,disable clock
18	RW	0x0	LCDC Share Memory clock disable. When HIGH,disable clock
17	RW	0x0	Mobile SDRAM Controller hclk clock disable. When HIGH,disable clock
16	RW	0x0	SDRAM Controller hclk clock disable. When HIGH,disable clock
15	RW	0x0	Mobile SDRAM/SDRAM common hclk clock disable. When HIGH,disable clock
14	RW	0x0	Reserved
13	RW	0x0	Reserved
12	RW	0x0	Reserved
11	RW	0x0	Reserved
10	RW	0x0	Reserved
9	RW	0x0	Reserved
8	RW	0x0	Reserved
7	RW	0x0	Reserved
6	RW	0x0	Reserved
5	RW	0x0	Reserved
4	RW	0x0	Reserved
3	RW	0x0	Reserved
2	RW	0x0	Reserved
1	RW	0x0	Reserved
0	RW	0x0	HS-ADC interface and logic clock disable. When HIGH,disable clock

**SCU\_CLKGATE2\_CON**

Address : Base Addr+0x24

Internal clock gating control register2

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:10	-	-	Reserved
9	RW	0x0	Video AHB bus clock disable. When HIGH, disable clock
8	RW	0x0	ARM ITCM clock disable. When HIGH, disable clock
7	RW	0x0	ARM DTCM0 clock disable. When HIGH, disable clock logic
6	RW	0x0	ARM DTCM1 clock disable. When HIGH, disable clock
5	RW	0x0	EFUSE IP clock disable. When HIGH, disable clock
4	RW	0x0	APB bus logic clock disable. When HIGH, disable clock
3	RW	0x0	EXP AHB bus clock disable. When HIGH, disable clock
2	RW	0x0	DSP AHB bus clock disable. When HIGH, disable clock
1	RW	0x0	ARMD bus clock disable. When High, disable clock
0	RW	0x0	ARMI bus clock disable. When HIGH, disable clock

**SCU\_SOFT\_RST\_CON**

Address : Base Addr+0x28

Internal soft reset control register

Bit	Attr	Reset Value	Description
31	RW	0x0	DDR bus logic reset request When HIGH, reset relative logic
30	RW	0x0	DDR core logic reset request When HIGH, reset relative logic
29	RW	0x0	GPU logic reset request When HIGH, reset relative logic
28	RW	0x0	SDRAM controller reset request When HIGH, reset relative logic
27	RW	0x0	Reserved
26	RW	0x0	Reserved
25	RW	0x0	DSP A2A bridge soft reset request. When HIGH, reset relative logic
24	RW	0x0	SD/MMC1 soft reset request. When HIGH, reset relative logic
23	RW	0x0	ARM core soft reset request. When HIGH, reset relative logic
22	RW	0x0	Reserved
21	RW	0x0	USB Host soft reset request for 48MHz clock logic. When HIGH, reset relative logic
20	RW	0x0	USB Host soft reset request for hclk clock logic. When HIGH, reset relative logic
19	RW	0x0	UART3 soft reset request. When HIGH, reset relative logic
18	RW	0x0	UART2 soft reset request. When HIGH, reset relative logic
17	RW	0x0	Customized Video SDRAM Controller logic reset request. When HIGH, reset relative logic
16	RW	0x0	SPI1(Slave) soft reset request. When HIGH, reset relative logic
15	RW	0x0	SPI0(Master) soft reset request. When HIGH, reset relative logic
14	RW	0x0	UART1 soft reset request. When HIGH, reset relative logic

13	RW	0x0	UART0 soft reset request. When HIGH, reset relative logic
12	RW	0x0	USB PHY reset request. When HIGH, reset relative logic
11	RW	0x0	USB controller logic soft reset request. When HIGH, reset relative logic
10	RW	0x0	Reserved
9	RW	0x0	SD/MMC0 soft reset request. When HIGH, reset relative logic
8	RW	0x0	Deblocking(RV) soft reset request. When HIGH, reset relative logic
7	RW	0x0	SAR_ADC soft reset request. When HIGH, reset relative logic
6	RW	0x0	I2S soft reset request. When HIGH, reset relative logic
5	RW	0x0	DSP peripheral module soft reset request. When HIGH, reset relative logic
4	RW	0x1	DSP CORE soft reset request. When HIGH, reset relative logic
3	RW	0x0	NandC soft reset request. When HIGH, reset relative logic
2	RW	0x0	VIP soft reset request. When HIGH, reset relative logic
1	RW	0x0	LCDC soft reset request. When HIGH, reset relative logic
0	RW	0x0	USB OTG soft reset request in HCLK domain. When HIGH, reset relative logic

**SCU\_CHIPCFG\_CON**

Address : Base Addr+0x2c

Chip config register

bit	Attr	Reset Value	Description
31:16	RW	0x0bb8	pll lock period control
15:0	-	-	reserved

**SCU\_CPUPD**

Address : Base Addr+0x30

ARM power down control register

bit	Attr	Reset Value	Description
31:0	RW	0x0	If write " 0xdeed_babe" will stop ARM926 clock

**SCU\_CLKSEL2\_CON**

Address : Base Addr+0x34

Internal clock select and divide register2

bit	Attr	Reset Value	Description
31:11	RW	24'b0	Reserved
10:8	RW	3'b011	Control sd/mmc1 clk frequency (hclk/(1~8)) mmc1_clk = arm_hclk /(mmc1clk_sel+1) default : arm_hclk/4
7:4	RW	4'b011	Control 48MHz clock divider frequency (armclk/(1~16)) clk48m = armclk/(clk48m_sel+1) default : armclk/4
3:0	RW	4'b0	Control arm clk divide frequency (1~16) arm_clk = arm_pll_clk /(armclk_sel+1)

default : arm\_pll\_clk/1

## 15.3 Application Notes

The following diagram shows clock architecture ( mux or divider information) in RK281x .

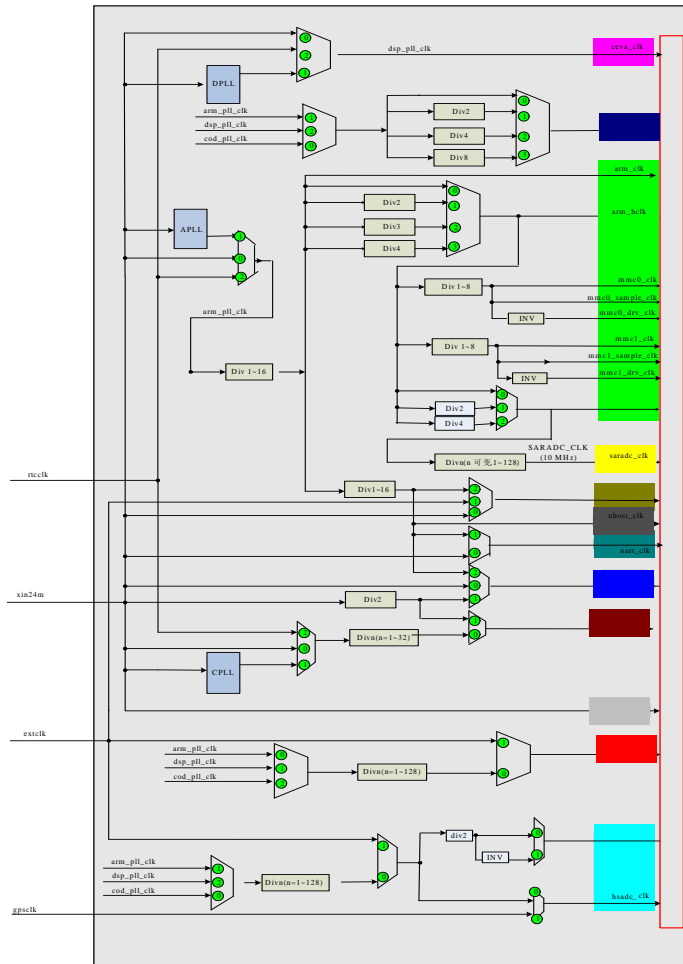


Fig. 15-1 RK281x clock architecture diagram

### 15.3.1 PLL usage

#### ● PLL output frequency configuration

The output frequency  $F_{out}$  is related to the reference frequency  $F_{ref}$  by:

$$F_{out} = F_{ref} * NF / NR / OD$$

$F_{out}$  is clk output of PLL , and  $F_{ref}$  is clk input of PLL from external oscillator (24MHz). Another , other factors such as NF, NR, OD can be configured by programming SCU\_APLL\_CON, SCU\_DPLL\_CON and SCU\_CPLL\_CON register , and their value will affect  $F_{out}$  as follows .

(1) CLKR: A 6-bit bus that selects the values 1-64 for the reference divider

$$NR = CLKR[5:0] + 1$$

Example:

/1 pgm 000000

/4 pgm 000011

/8 pgm 000111

(2) CLKF: A 12-bit bus that selects the values 1-4096 for the PLL multiplication factor

$$NF = CLKF[11:0] + 1$$

Example:

```
X1      pgm 000000000000
X2      pgm 000000000001
X4096   pgm 111111111111
```

(3) CLKOD: A 3-bit bus that selects the values 1-8 for the PLL post VCO divider

$OD = CLKOD[2:0] + 1$

Example:

```
/1      pgm 000
/4      pgm 011
/8      pgm 111
```

- **PLL frequency range requirement**

Fref/NR value range requirement : 97.7KHz - 800MHz

Fref/NR \* NF value range requirement: 160MHz - 800MHz

If different CLKR and CLKF configuration value cause internal out of range, unpredicted result will be caused.

- **PLL frequency change method**

Before set some factors such NR/NF/OD to change PLL output frequency, you must change PLL from normal to bypass mode by programming SCU\_APLL\_CON, SCU\_DPLL\_CON and SCU\_CPLL\_CON register or change chip from normal to slow mode by programming SCU\_MODE\_CON and SCU\_CLKSEL1\_CON register. The later method is recommended. Then until PLL is in lock state by check CPU\_APB\_REG0 register you can change PLL into normal mode, or after delay about 0.3ms.

- **PLL powerdown**

You can make PLL into or out of powerdown mode by programming SCU\_APLL\_CON, SCU\_DPLL\_CON and SCU\_CPLL\_CON register. After PLL will be out of powerdown mode, you can check CPU\_APB\_REG0 register to confirm PLL in lock state.

### 15.3.2 Power mode management

The SCU provide five power management mode for system power saving and system can enter each power saving mode by setting appropriate control registers and programming sequence.

Mode	CPU	System IP	Peripheral IP	Power	Frequency
Normal	Run	Stop unused IP clock by software setting	Stop unused IP clock by software setting	On	133MHz (ARM PLL) 300MHz (DSP PLL) 122.88MHz (CODEC PLL)
Slow	Run	Stop unused IP clock by software setting	Stop unused IP clock by software setting	On	24MHz Low speed
IDLE	Halt	Stop unused IP clock by software setting	Stop unused IP clock by software setting	On	Normal frequency or slow
Stop	Halt	Off	Off	On	Off
Power off	Off	Off	Off	RTC battery	Off

#### Normal mode :

In normal mode, CPU, system IPs and all peripheral IPs should work normally. The power consumption will be maximum when all IPs are turn on. Software allow to stop unused IP clock by programming SCU\_CLKGATE<sub>x</sub>\_CON( $x=0\sim2$ ), register to reduce the power consumption.

#### Slow mode:

In SLOW mode, the system clock source is switching from high speed clock (PLL) to external lower speed clock source, and then power down PLL for further power saving. Enter by setting SCU\_CLKSEL<sub>x</sub>\_CON( $x=0,1$ ) register to select system clock source from PLL to the external OSC and set SCU\_APLL\_CON, SCU\_DPLL\_CON and SCU\_CPLL\_CON registers to turn off PLL. Exit by turning on PLL and wait for PLL locked, switch system clock source back to PLL clock.

**IDLE mode:**

In IDLE mode, the CPU is expected to be idle and just wait for interrupts. In this case, software will make CPU to power down state. The peripheral IP will keep running and wake up the CPU by external interrupt or external wakeup.

**Stop mode:**

In STOP mode, the operation of CPU and all IP should be halted. The clock of all IP is stop since PLL is power down. The system can release the stop mode from external wakeup pin.

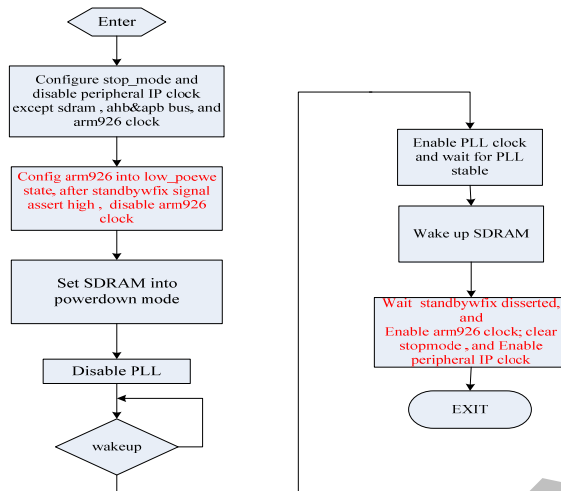


Fig. 15-2 RK281x system stop mode operation flow

**Power Off mode:**

In power off mode, the system power is shut down. And the RTC is switch to battery power and keep running. The system can be power on again from RTC alarm or manually. For detail programming sequence please refer to RTC specification.

**Programming Sequence:**

- Normal mode:
  - ◆ Disable unused IP clock by setting SCU\_CLKGATE<sub>x</sub>\_CON(x=0~2) register.
- Slow mode:
  - ◆ SCU\_MODE\_CON[3:0] register to "1111" to select clock source to low speed clock.
  - ◆ Turn off the PLL by setting SCU\_APLL\_CON[22], SCU\_DPLL\_CON[22]
  - ◆ Before switch to PLL clock, turn on PLL by setting SCU\_APLL\_CON[22], SCU\_DPLL\_CON[22] and read CPU\_APB\_REG0[7] to check PLL lock status or delay 0.3ms.
- IDLE mode:
  - ◆ Program ARM926 to low\_power state by instruction MCR p15,0 <Rd>,c7,c0,4
  - ◆ Idle mode will exit by external interrupt or wakeup pin
- STOP mode
  - ◆ Set SCU\_CLKGATE<sub>x</sub>\_CON(x=0~2) to disable all peripheral IP clock except sdram, ahb and apb bus, and arm926 clock
  - ◆ Set SCU\_MODE\_CON[6:5] to select wakeup stop mode method
  - ◆ Set RTC alarm time in RTC control register if use RTC alarm to wakeup
  - ◆ Set SCU\_MODE\_CON[7] to select ewakeup signal polarity if use external wakeup pin to wakeup
  - ◆ Set SCU\_MODE\_CON[4] and set SCU\_CPUPD to 0xdeed\_babe to enable stop mode
  - ◆ Program ARM926 to low\_power state by instruction MCR p15,0 <Rd>,c7,c0,4

- ◆ Check the interrupt status when wakeup from stop mode and use SCU\_MODE\_CON[8] to clear SCU\_INT if system wakeup by external pin.

### 15.3.3 HCLK frequency switch method

In RK281x, since clock frequency ratio between armclk and hclk can support from 1:1 to 1:4, before we do different ratio select, we must make system enter into slow mode, in other word, system must work in lower frequency, or else, if switch operation for hclk is finished in normal mode, it will generate unpredictable result for system. After switch operation is done, we can make system work in normal mode. The detailed steps are shown as follows:

- Make system enter into slow mode, and armclk frequency is 24MHz
- Set SCU\_CLKSEL0\_CON bit[1:0] to get correct ratio value for armclk and hclk
- Change ARM PLL output frequency to make system requirement
- Make system enter into normal mode after PLL is in lock state



## Chapter 16 PMU in CPU System

### 16.1 Design Overview

#### 16.1.1 Overviews

The Power Management Unit (PMU) focuses on power on/off switch for different power domain in RK281x. RK281x has been divided into 6 independent power domain such as CPU System, DSP System, LCDC System, DDR PHY and SCU, RTC. In them, SCU and RTC is always on power domain, not be switched off, the four other modules can be switched on/off for their power by software method. Therefore, when one module is not used in some application, we can make it power off to save power, even leakage power.

Another, after CPU system will be powered off, we can wake up it by external pin.

#### 16.1.2 Features

The PMU has the following main feature:

- Support power off/on switch by software for 4 power domain
- Support external wake up for main module (CPU System)

### 16.2 Power Domain Architecture

The following diagram shows the different power domain in different colors.

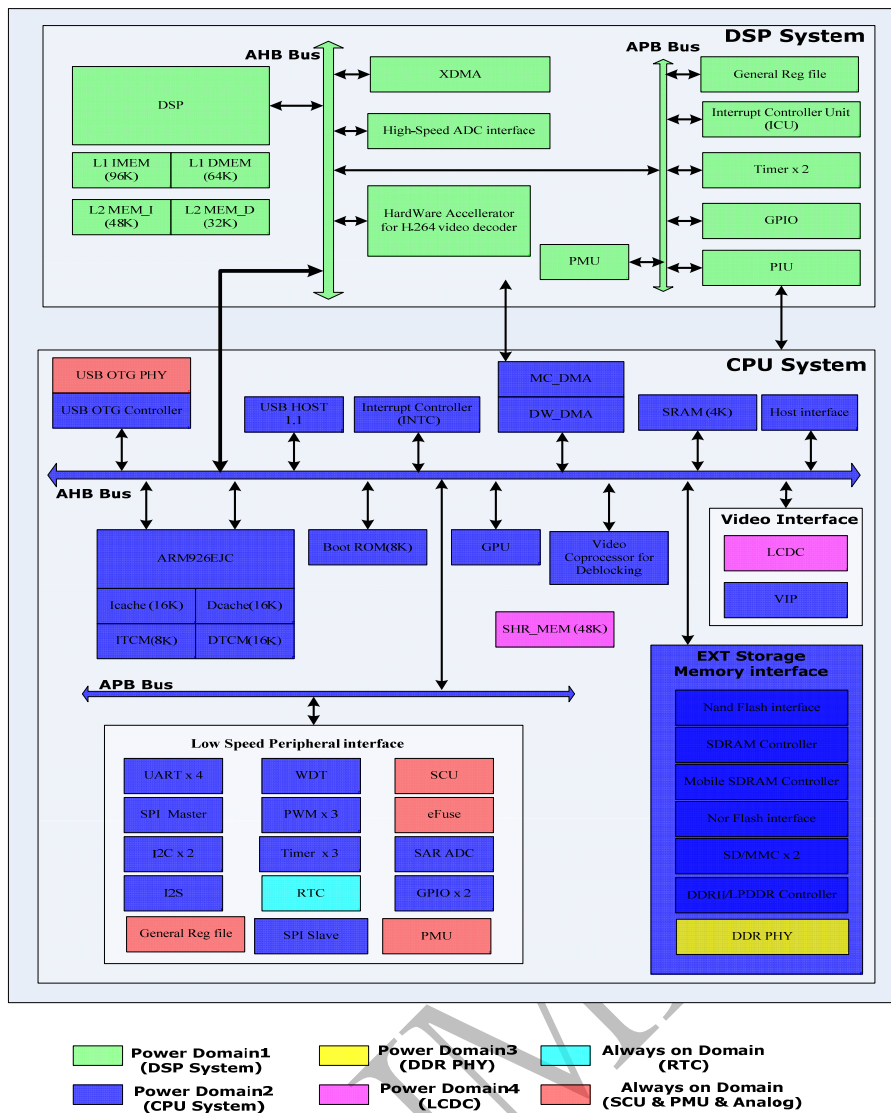


Fig. 16-1 RK281x power domain architecture

## 16.3 Registers

As for PMU register, please refer to register SCU\_PMU\_MODE in Chapter 16(System Control Unit) for more detailed descriptions.

## Chapter 17 UART

### 17.1 Design Overview

#### 17.1.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) is used for serial communication with a peripheral, modem (data carrier equipment, DCE) or data set. Data is written from a master (CPU) over the APB bus to the UART and it is converted to serial form and transmitted to the destination device. Serial data is also received by the UART and stored for the master (CPU) to read back.

#### 17.1.2 Features

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- DMA Controller Interface – Enables interface to a DMA controller over the AMBA bus using a handshaking interface for transfer requests.
- Support interrupt interface to interrupt controller.
- Two 32x8bits fifos for transferring and receiving use respectively.
- Programmable serial data baud rate as calculated by the following:  $\text{baud rate} = (\text{serial clock frequency}) / (16 \times \text{divisor})$ .
- IrDA 1.0 SIR mode support with up to 115.2 Kbaud data rate and a pulse duration (width) as follows:  $\text{width} = 3/16 \times \text{bit period}$  as specified in the IrDA physical layer specification.
- UART0 supports modem function, UART1 supports IrDA function.

### 17.2 Architecture

This section provides a description about the functions and behavior under various conditions.

#### 17.2.1 Block Diagram

The UART comprises with:

- ◆ AMBA APB interface
- ◆ FIFO controllers
- ◆ Register block
- ◆ Modem synchronization block and baud clock generation block
- ◆ Serial receiver and serial transmitter

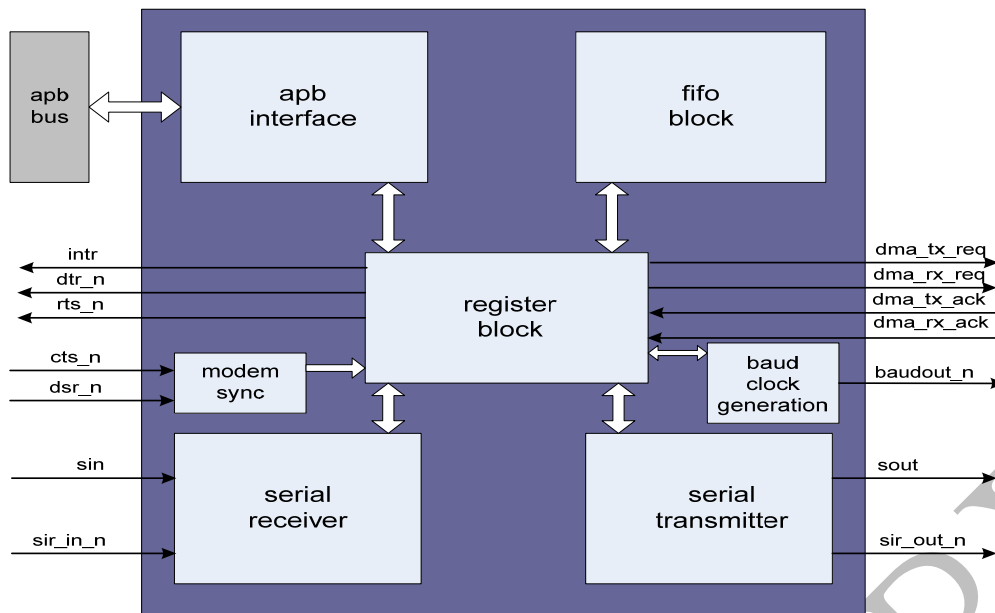


Fig. 17-1 RK281x UART architecture diagram

## 17.2.2 Block Descriptions

### APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

### Register block

Be responsible for the main UART functionality including control, status and interrupt generation.

### Modem Synchronization block

Synchronizes the modem input signal.

### FIFO block

Be responsible for FIFO control and storage (when using internal RAM) or signaling to control external RAM (when used).

### Baud Clock Generator

Produces the transmitter and receiver baud clock along with the output reference clock signal (baudout\_n).

### Serial Transmitter

Converts the parallel data, written to the UART, into serial form and adds all additional bits, as specified by the control register, for transmission. This makeup of serial data, referred to as a character can exit the block in two forms, either serial UART format or IrDA 1.0 SIR format.

### Serial Receiver

Converts the serial data character (as specified by the control register) received in either the UART or IrDA 1.0 SIR format to parallel form. Parity error detection, framing error detection and line break detection is carried out in this block.

## 17.3 Registers

This section describes the control/status registers of the design.

### 17.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
UART_RBR	0x0000	W	0x0000_0000	Receive Buffer Register
UART_THR				Transmit Holding Register
UART_DLL				Divisor Latch (Low)
UART_DLH	0x0004	W	0x0000_0000	Divisor Latch (High)
UART_IER				Interrupt Enable Register

UART_IIR	0x0008	W	0x0000_0000	Interrupt Identification Register
UART_FCR				FIFO Control Register
UART_LCR	0x000C	W	0x0000_0000	Line Control Register
UART_MCR	0x0010	W	0x0000_0000	Modem Control Register
UART_LSR	0x0014	W	0x0000_0060	Line Status Register
UART_MSR	0x0018	W	0x0000_0000	Modem Status Register
UART_SCR	0x001c	W	0x0000_0000	Scratchpad Register
Reserved	0x0020-2C	W	0x0000_0000	--
UART_SRBR	0x0030	W	0x0000_0000	Shadow Receive Buffer Register
UART_STHR	-6C	W	0x0000_0000	Shadow Transmit Holding Register
UART_FAR	0x0070	W	0x0000_0000	FIFO Access Register
UART_TFR	0x0074	W	0x0000_0000	Transmit FIFO Read
UART_RFW	0x0078	W	0x0000_0000	Receive FIFO Write
UART_USR	0x007C	W	0x0000_0006	UART Status Register
UART_TFL	0x0080	W	0x0000_0000	Transmit FIFO Level
UART_RFL	0x0084	W	0x0000_0000	Receive FIFO Level
UART_SRR	0x0088	W	0x0000_0000	Software Reset Register
UART_SRTS	0x008C	W	0x0000_0000	Shadow Request to Send
UART_SBCR	0x0090	W	0x0000_0000	Shadow Break Control Register
UART_SDMAM	0x0094	W	0x0000_0000	Shadow DMA Mode
UART_SFE	0x0098	W	0x0000_0000	Shadow FIFO Enable
UART_SRT	0x009C	W	0x0000_0000	Shadow RCVR Trigger
UART_STET	0x00A0	W	0x0000_0000	Shadow TX Empty Trigger
UART_HTX	0x00A4	W	0x0000_0000	Halt TX
UART_DMASA	0x00A8	W	0x0000_0000	DMA Software Acknowledge
Reserved	0x00AC-F0	W	0x0000_0000	--
UART_CPR	0x00F4	W	0x0000_0000	Component Parameter Register
UART_UCV	0x00F8	W	0x3330_372a	UART Component Version
UART_CTR	0x00FC	W	0x4457_0110	Component Type Register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 17.3.2 Detail Register Description

#### UART\_RBR

Address: Operational Base + offset( 0x00)

Receive Buffer Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	Reserved
7:0	RW	0x0	Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LSR) is set. If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error. If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data

			character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.
--	--	--	---------------------------------------------------------------------------------------------------------------------------------

**UART\_THR**

Address: Operational Base + offset( 0x00)

Transmit Holding Register

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If in non-FIFO mode or FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If in FIFO mode and FIFOs are enabled (FCR[0] = 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.

**UART\_DLL**

Address: Operational Base + offset( 0x00)

Divisor Latch (Low)

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The output baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor). Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLH is set, at least 8 clock cycles of the slowest Uart clock should be allowed to pass before transmitting or receiving data.

**UART\_DLH**

Address: Operational Base + offset( 0x04)

Divisor Latch (High)

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Upper 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.

**UART\_IER**

Address: Operational Base + offset( 0x04)

## Interrupt Enable Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	Reserved and read as zero
7	RW	0x0	Programmable THRE Interrupt Mode Enable This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled
6:4			Reserved and read as zero
3	RW	0x0	Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0 = disabled 1 = enabled
2	RW	0x0	Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0 = disabled 1 = enabled
1	RW	0x0	Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = disabled 1 = enabled
0	RW	0x0	Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled). These are the second highest priority interrupts. 0 = disabled 1 = enabled

## UART\_IIR

Address: Operational Base + offset( 0x08)

## Interrupt Identification Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	
7:6	R	0x01	FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00 = disabled 11 = enabled
5:4			Reserved and read as zero
3:0	R	0x01	Interrupt ID. This indicates the highest priority pending interrupt which can be one of the following types: 0000 = modem status 0001 = no interrupt pending 0010 = THR empty 0100 = received data available 0110 = receiver line status 0111 = busy detect 1100 = character timeout

## UART\_FCR

Address: Operational Base + offset( 0x08)

## FIFO Control Register

bit	Attr	Reset Value	Description
-----	------	-------------	-------------



31:8	RW	0x0	Reserved and read as zero
7:6	W	0x0	RCVR Trigger. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. In auto flow control mode it is used to determine when the rts_n signal is de-asserted. It also determines when the dma_rx_req_n signal is asserted in certain modes of operation. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO ¼ full 10 = FIFO ½ full 11 = FIFO 2 less than full
5:4	W	0x0	TX Empty Trigger. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. It also determines when the dma_tx_req_n signal is asserted when in certain modes of operation. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO ¼ full 11 = FIFO ½ full
3	W	0x0	DMA Mode. This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected . 0 = mode 0 1 = mode 1 1100 = character timeout
2	W	0x0	XMIT FIFO Reset. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are selected . Note that this bit is 'self-clearing'. It is not necessary to clear this bit.
1	W	0x0	RCVR FIFO Reset. This resets the control portion of the receive FIFO and treats the FIFO as empty. This also de-asserts the DMA RX request and single signals when additional DMA handshaking signals are selected . Note that this bit is 'self-clearing'. It is not necessary to clear this bit
0	W	0x0	FIFO Enable. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.

**UART\_LCR**

Address: Operational Base + offset(0x0C)

Line Control Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	Reserved and read as zero
7	RW	0x0	Divisor Latch Access Bit. Writeable only when UART is not busy (USR[0] is zero), always readable. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate

			setup in order to access other registers
6	RW	0x0	Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If MCR[6] set to one, the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low
5	Reserved and read as zero		
4	RW	0x0	Even Parity Select. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic 1s is transmitted or checked. If set to zero, an odd number of logic 1s is transmitted or checked.
3	RW	0x0	Parity Enable. Writeable only when UART is not busy (USR[0] is zero), always readable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively. 0 = parity disabled 1 = parity enabled
2	RW	0x0	Number of stop bits. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit
1:0	RW	0x0	Data Length Select. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bit that may be selected areas follows: 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits

**UART\_MCR**

Address: Operational Base + offset(0x10)

Modem Control Register

bit	Attr	Reset Value	Description
31:7	RW	0x0	Reserved and read as zero
6	RW	0x0	SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode . 0 = IrDA SIR Mode disabled

			1 = IrDA SIR Mode enabled
5	RW	0x0	Auto Flow Control Enable. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled
4	RW	0x0	LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes.
3	RW	0x0	OUT2. This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is: 0 = out2_n de-asserted (logic 1) 1 = out2_n asserted (logic 0)
2	RW	0x0	OUT1
1	RW	0x0	Request to Send. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data
0	RW	0x0	Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is: 0 = dtr_n de-asserted (logic 1) 1 = dtr_n asserted (logic 0)

**UART\_LSR**

Address: Operational Base + offset(0x14)

Line Status Register

bit	Attr	Reset Value	Description
31:8	Reserved and read as zero		
7	R	0x0	Receiver FIFO Error bit. This bit is relevant if FIFOs are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0 = no error in RX FIFO 1 = error in RX FIFO
6	R	0x1	Transmitter Empty bit. If FIFOs enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.
5	R	0x1	Transmit Holding Register Empty bit. If THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If IER[7] set to one and FCR[0] set to one respectively, the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.
4	R	0x0	Break Interrupt bit. This is used to indicate the

			detection of a break sequence on the serial input data.
3	R	0x0	Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data.
2		0x0	Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.
1	R	0x0	Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.
0	R	0x0	Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. 0 = no data ready 1 = data ready

**UART\_MSR**

Address: Operational Base + offset(0x18)

Modem Status Register

bit	Attr	Reset Value	Description
31:8	Reserved		
7	R	0x0	Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n.
6	R	0x0	Ring Indicator. This is used to indicate the current state of the modem control line ri_n.
5	R	0x0	Data Set Ready. This is used to indicate the current state of the modem control line dsr_n.
4	R	0x0	Clear to Send. This is used to indicate the current state of the modem control line cts_n.
3	R	0x0	Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.
2	R	0x0	Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.
1	R	0x0	Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read.
0	R	0x0	Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read.

**UART\_SCR**

Address: Operational Base + offset(0x1C)

Scratchpad Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	This register is for programmers to use as a temporary storage space.

**UART\_SRB**

Address: Operational Base + offset(0x30-6C)

Shadow Receive Buffer Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	<p>This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set.</p> <p>If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun error.</p> <p>If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO are preserved, but any incoming data is lost. An overrun error also occurs</p>

**UART\_STHR**

Address: Operational Base + offset(0x30-6C)

Shadow Transmit Holding Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	This is a shadow register for the THR.

**UART\_FAR**

Address: Operational Base + offset(0x70)

FIFO Access Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	<p>This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled. When FIFOs are not enabled it allows the RBR to be written by the master and the THR to be read by the master.</p> <p>0 = FIFO access mode disabled 1 = FIFO access mode enabled</p>

**UART\_TFR**

Address: Operational Base + offset(0x74)

Transmit FIFO Read

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	<p>Transmit FIFO Read. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are implemented and enabled, reading this register gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO.</p> <p>When FIFOs are not implemented or not enabled, reading this register gives the data in the THR.</p>

**UART\_RFW**

Address: Operational Base + offset(0x78)

Receive FIFO Write

bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9	W	0x0	Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).
8	W	0x0	Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).
7:0	W	0x0	Receive FIFO Write Data. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are enabled, the data that is written to the RFW is pushed into the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFOs not enabled, the data that is written to the RFW is pushed into the RBR.

**UART\_USR**

Address: Operational Base + offset(0x7C)

UART Status Register

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4	R	0x0	Receive FIFO Full. This is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full
3	R	0x0	Receive FIFO Not Empty. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty
2	R	0x1	Transmit FIFO Empty. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty
1	R	0x1	Transmit FIFO Not Full. This is used to indicate that the transmit FIFO is not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full.
0	R	0x0	UART Busy. This indicates that a serial transfer is in progress, when cleared indicates that the uart is idle or inactive. 0 = Uart is idle or inactive 1 = Uart is busy (actively transferring data)

**UART\_TFL**

Address: Operational Base + offset(0x80)

Transmit FIFO Level

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4:0	R	0x0	Transmit FIFO Level. This indicates the number



			of data entries in the transmit FIFO.
--	--	--	---------------------------------------

**UART\_RFL**

Address: Operational Base + offset(0x84)

Receive FIFO Level

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4:0	R	0x0	Receive FIFO Level. This indicates the number of data entries in the receive FIFO.

**UART\_SRR**

Address: Operational Base + offset(0x88)

Software Reset Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	W	0x0	XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]).
1	W	0x0	RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]).
0	W	0x0	UART Reset. This asynchronously resets the Uart and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset.

**UART\_SRTS**

Address: Operational Base + offset(0x8C)

Shadow Request to Send

bit	Attr	Reset Value	Description
31:1	-	-	
0	RW	0x0	Shadow Request to Send. This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to performing a read-modify-write on the MCR.

**UART\_SBCR**

Address: Operational Base + offset(0x90)

Shadow Break Control Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	Shadow Break Control Bit. This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR.

**UART\_SDMAM**

Address: Operational Base + offset(0x94)

Shadow DMA Mode

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
	RW	0x0	Shadow DMA Mode. This is a shadow register for the DMA mode bit (FCR[3]).

**UART\_SFE**

Address: Operational Base + offset(0x98)

Shadow FIFO Enable

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]).

**UART\_SRT**

Address: Operational Base + offset(0x9C)



## Shadow RCVR Trigger

bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1:0	RW	0x0	Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]).

**UART\_STET**

Address: Operational Base + offset(0xa0)

## Shadow TX Empty Trigger

bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1:0	RW	0x0	Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]).

**UART\_HTX**

Address: Operational Base + offset(0xa4)

## Halt TX

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
	RW	0x0	This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 0 = Halt TX disabled 1 = Halt TX enabled

**UART\_DMASA**

Address: Operational Base + offset(0xa8)

## RTC counter reset register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	W	0x0	This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition.

**UART\_UCV**

Address: Operational Base + offset(0xf8)

## UART Component Version

bit	Attr	Reset Value	Description
31:0	R	0x330372a	ASCII value for each number in the version

**UART\_CTR**

Address: Operational Base + offset(0xfc)

## Component Type Register

bit	Attr	Reset Value	Description
31:0	R	0x44570110	This register contains the peripherals identification code.

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 17.4 Functional Description

### 17.4.1 Operation

**● UART (RS232) Serial Protocol**

Because the serial communication is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure to perform simple error checking on the received data., as shown in Figure.

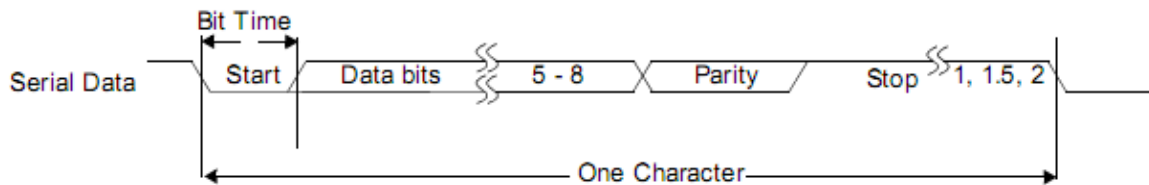


Fig. 17-2 UART Serial protocol

### ● Baud Clock

The baud rate controlled by the serial clock (sclk or pclk in a single clock implementation) and the Divisor Latch Register (DLH and DLL). As the exact number of baud clocks that each bit was transmitted for is known, calculating the mid point for sampling is not difficult, that is every 16 baud clocks after the mid point sample of the start bit.

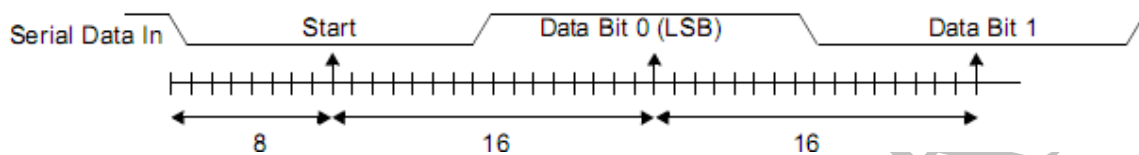


Fig. 17-3 UART baud rate

### ● IrDA 1.0 SIR Protocol

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. IrDA 1.0 SIR mode specifies a maximum baud rate of 115.2 Kbaud. Transmitting a single infrared pulse signals a logic zero, while a logic one is represented by not sending a pulse. The width of each pulse is 3/16ths of a normal serial bit time. data transfers can only occur in half-duplex fashion when IrDA SIR mode is enabled.

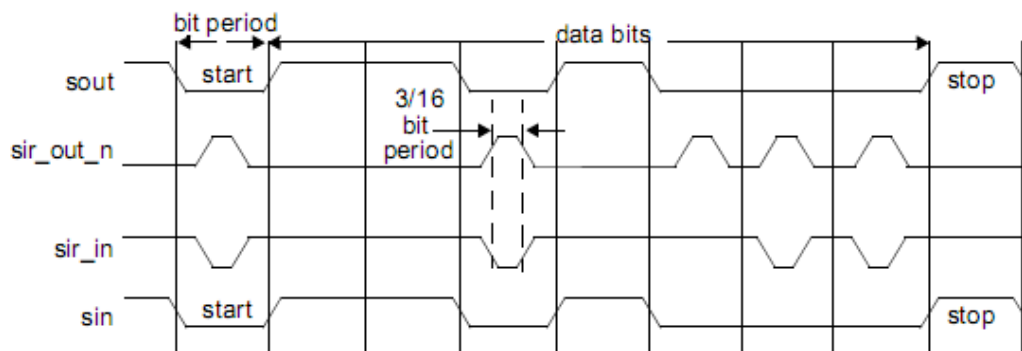


Fig. 17-4 UART IrDA1.0 timing waveform

### ● FIFO Support

#### 1. NONE FIFO MODE

If FIFO support is not selected, then no FIFOs are implemented and only a single receive data byte and transmit data byte can be stored at a time in the RBR and THR.

#### 2. FIFO MODE

The FIFO depth is 32, enabled by register FCR[0].

### ● Interrupts

The following interrupt types can be enabled with the IER register.

Receiver Error;

Receiver Data Available;

Character Timeout (in FIFO mode only);

Transmitter Holding Register Empty at/below threshold (in Programmable THRE Interrupt mode);

Modem Status;

### ● DMA Support

The uart supports DMA signalling with the use of two output signals (dma\_tx\_req\_n and dma\_rx\_req\_n) to indicate when data is ready to be read or when the transmit FIFO is empty.

The dma\_tx\_req\_n signal is asserted under the following conditions:

- When the Transmitter Holding Register is empty in non-FIFO mode
- When the transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled
- When the transmitter FIFO is at, or below the programmed threshold with Programmable THRE interrupt mode enabled.

The dma\_rx\_req\_n signal is asserted under the following conditions:

- When there is a single character available in the Receive Buffer Register in non-FIFO mode
- When the Receiver FIFO is at or above the programmed trigger level in FIFO mode

### ● Auto Flow Control

The uart can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available. If FIFOs are not implemented, then this mode cannot be selected. When Auto Flow Control mode has been selected it can be enabled with the Modem Control Register (MCR[5]). Following figure shows a block diagram of the Auto Flow Control functionality.

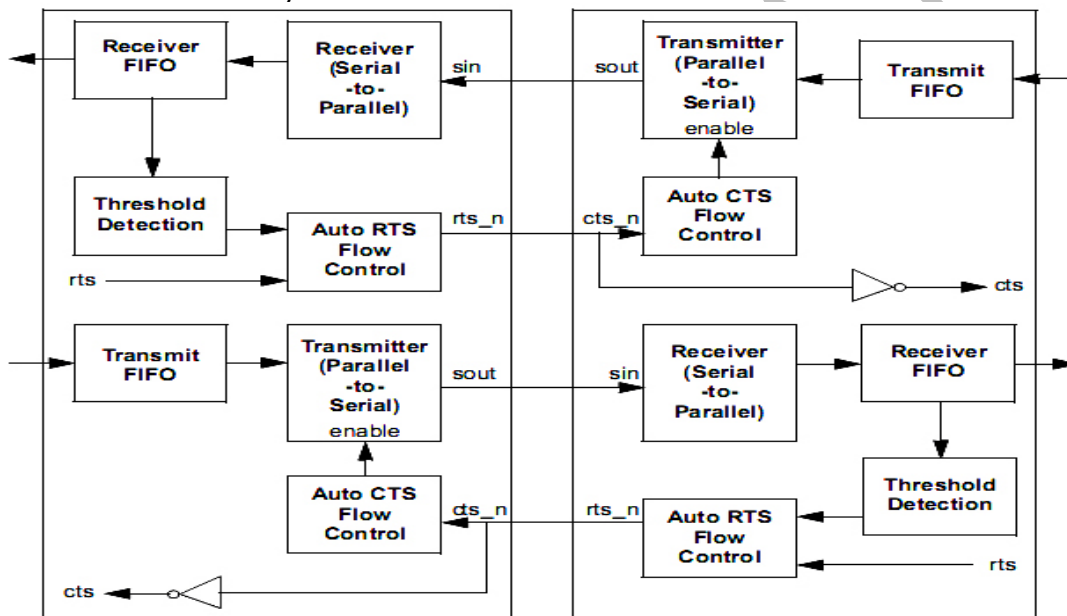


Fig. 17-5 UART Auto flow control block diagram

Auto RTS – Becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- RTS (MCR[1] bit and MCR[5]bit are both set)
- FIFOs are enabled (FCR[0] bit is set)
- SIR mode is disabled (MCR[6] bit is not set)

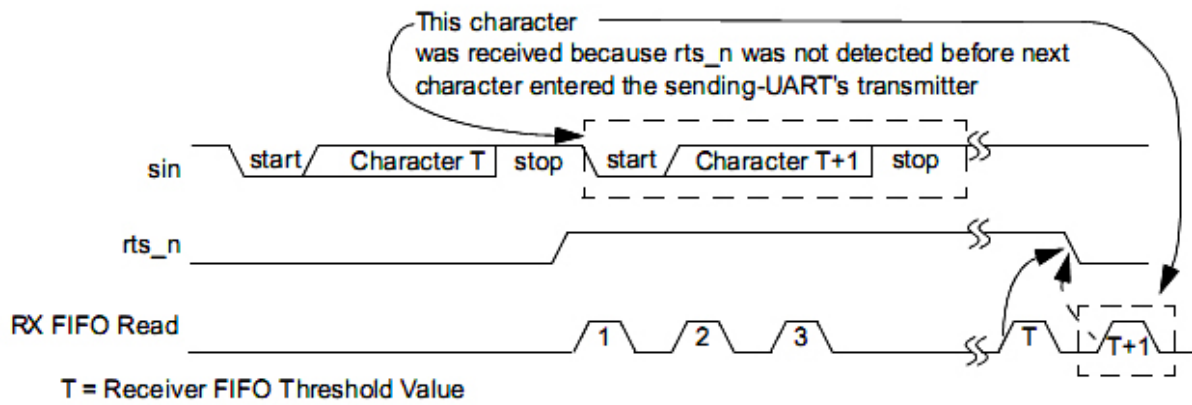


Fig. 17-6 UART Auto RTS timing waveform

Auto CTS – becomes active when the following occurs:

- Auto Flow Control is selected during configuration
- FIFOs are implemented
- AFCE (MCR[5] bit is set)
- FIFOs are enabled through FIFO Control Register FCR[0] bit
- SIR mode is disabled (MCR[6] bit is not set)

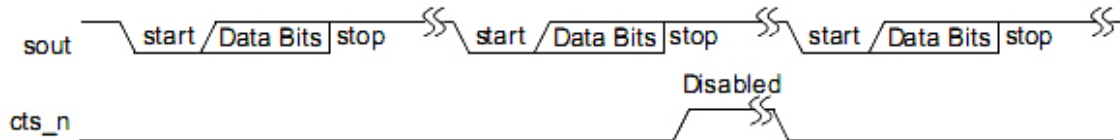


Fig. 17-7 UART Auto CTS timing waveform

## 17.4.2 Programming sequence

### ● None FIFO Mode Transfer Flow

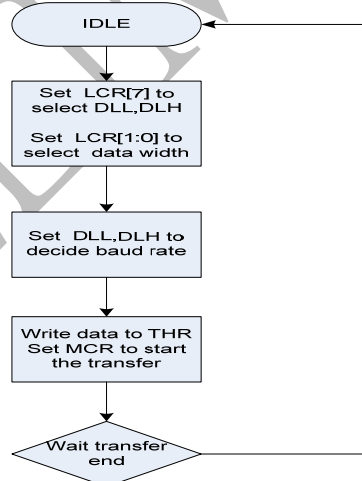


Fig. 17-8 Uart work flow in none fifo mode

### ● FIFO Mode Transfer Flow

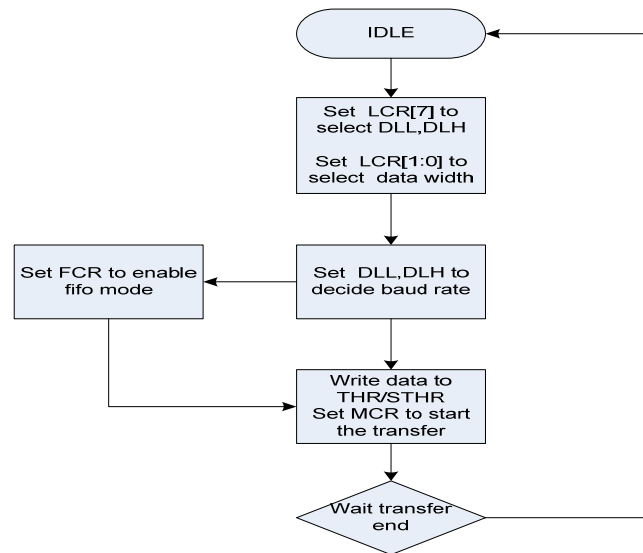


Fig. 17-9 Uart fifo mode flow diagram

The UART is an APB slave performing:

Serial-to-parallel conversion on data received from a peripheral device

Parallel-to-serial conversion on data transmitted to the peripheral device

The CPU reads and writes data and control/status information through the APB interface. The transmitting and receiving paths are buffered with internal FIFO memories enabling up to 16-bytes to be stored independently in both transmit and receive modes. A baud rate generator can generate a common transmit and receive internal clock input. The baud rates will depend on the internal clock frequency. The UART will also provide transmit, receive and exception interrupts to system. A DMA interface is implemented for improving the system performance.

## Chapter 18 SPI Master Controller

### 18.1 Design Overview

#### 18.1.1 Overview

The Serial Peripheral Interface 0(SPI0) is an APB slave device. A four-wire, full-duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The slave select line is held high when the spi is idle or disabled. SPI master controller only work as master mode.

#### 18.1.2 Features

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- Serial-master operation – Enables serial communication with serial-slave peripheral devices.
- DMA Controller Interface – Enables interface to a DMA controller over the AMBA bus using a handshaking interface for transfer requests.
- Support interrupt interface to interrupt controller, and independently masking of interrupts.
- Dedicated 2 hardware slave-select lines.
- Dynamic control of the serial bit rate of the data transfer.
- Two 16x16 fifos for transferring and receiving use respectively.

### 18.2 Architecture

This section provides a description about the functions and behavior under various conditions.

#### 18.2.1 Block Diagram

The SPI comprises with:

- ◆ AMBA APB interface and DMA Controller Interface
- ◆ Transmit and receive FIFO controllers and an FSM controller
- ◆ Register block
- ◆ Shift control and interrupt

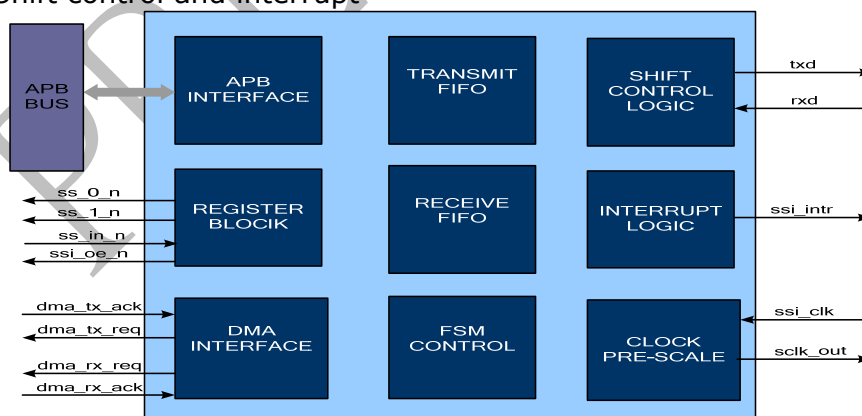


Fig. 18-1 RK281x SPI Master Controller Block diagram

#### 18.2.2 Block Descriptions

##### APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

**DMA INTERFACE**

This block has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA Controller.

**FIFO LOGIC**

For transmit and receive transfers, data transmitted from the SPI to the external serial device is written into the transmit FIFO. Data received from the external serial device into the SPI is pushed into the receive FIFO. Both fifos are 16x16bits.

**FSM CONTROL**

Control the state's transformation of the design.

**REGISTER BLOCK**

All registers in the SPI are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bits wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

**SHIFT CONTROL**

Shift control logic shift the data from the transmit fifo or to the receive fifo. This logic automatically right-justifies receive data in the receive FIFO buffer

**INTERRUPT CONTROL**

The SPI supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other SPI interrupts after masking.

**18.3 Registers**

This section describes the control/status registers of the design.

**18.3.1 Registers Summary**

Name	Offset	Size	Reset Value	Description
SPIM_CTRLR0	0x0000	W	0x0000_0007	Control register 0
SPIM_CTRLR1	0x0004	W	0x0000_0000	Control register 1
SPIM_SPIENR	0x0008	W	0x0000_0000	Ssi enable register
SPIM_MWCR	0x000C	W	0x0000_0000	Microwire control register
SPIM_SER	0x0010	W	0x0000_0000	Slave enable register
SPIM_BAUDR	0x0014	W	0x0000_0000	Baud rate select
SPIM_TXFTLR	0x0018	W	0x0000_0000	Transmit FIFO Threshold Level
SPIM_RXFTLR	0x001c	W	0x0000_0000	Receive FIFO Threshold Level
SPIM_TXFLR	0x0020	W	0x0000_0000	Transmit FIFO Level Register
SPIM_RXFLR	0x0024	W	0x0000_0000	Receive FIFO Level Register
SPIM_SR	0x0028	W	0x0000_0006	Status Register
SPIM_IMR	0x002c	W	0x0000_003f	Interrupt Mask Register
SPIM_ISR	0x0030	W	0x0000_0000	Interrupt Status Register
SPIM_RISR	0x0034	W	0x0000_0000	Raw Interrupt Status Register
SPIM_TXOICR	0x0038	W	0x0000_0000	Transmit FIFO Overflow Interrupt Clear Register
SPIM_RXOICR	0x003c	W	0x0000_0000	Receive FIFO Overflow Interrupt Clear Register
SPIM_RXUICR	0x0040	W	0x0000_0000	Receive FIFO Underflow Interrupt Clear Register
SPIM_MSTICR	0x0044	W	0x0000_0000	Multi-Master Interrupt Clear



				Register
SPIM_ICR	0x0048	W	0x0000_0000	Interrupt Clear Register
SPIM_DMACR	0x004c	W	0x0000_0000	DMA Control Register
SPIM_DMATDLR	0x0050	W	0x0000_0000	DMA Transmit Data Level
SPIM_DMARDLR	0x0054	W	0x0000_0000	DMA Receive Data Level
SPIM_IDR	0x0058	W	0xffff_ffff	Identification Register
SPIM_SPI_COMP_VERSION	0x005c	W	0x3331_302a	coreKit version ID register
SPIM_DR	0x0060-9C	W	0x0000_0000	Data Register

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

### 18.3.2 Detail Register Description

#### SPIM\_CTRL0

Address: Operational Base + offset( 0x00)

Control register 0

bit	Attr	Reset Value	Description
15:12	RW	0x0	Control Frame Size. Selects the length of the control word for the Microwire frame format. For the field decode,
11	RW	0x0	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 – Normal Mode Operation 1 – Test Mode Operation
10	RW	0x0	Slave Output Enable.
9:8	RW	0x0	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. 00 -- Transmit & Receive 01 -- Transmit Only 10 -- Receive Only 11 -- EEPROM Read
7	RW	0x0	Serial Clock Polarity 0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high
6	RW	0x0	Serial Clock Phase 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit
5:4	RW	0x0	Frame Format. Selects which serial protocol transfers the data. 00 -- Motorola SPI 01 -- Texas Instruments SSP 10 -- National Semiconductors Microwire 11 -- Reserved
3:0	RW	0x7	Data Frame Size. Selects the data frame length . When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the

			upper unused bits when transmitting the data. 0000: Reserved – undefined operation 0001: Reserved – undefined operation 0010: Reserved – undefined operation 0011: 4-bit serial data transfer 0100: 5-bit serial data transfer 0101: 6-bit serial data transfer 0110: 7-bit serial data transfer 0111: 8-bit serial data transfer 1000: 9-bit serial data transfer 1001: 10-bit serial data transfer 1010: 11-bit serial data transfer 1011: 12-bit serial data transfer 1100: 13-bit serial data transfer 1101: 14-bit serial data transfer 1110: 15-bit serial data transfer 1111: 16-bit serial data transfer
--	--	--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**SPIM\_CTRL1**

Address: Operational Base + offset( 0x04)

Control register 1

bit	Attr	Reset Value	Description
15:0	RW	0x0	Number of Data Frames. When TMOD = 10, this register field sets the number of data frames to be continuously received by the SPI. The Spi continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. When the SPI is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the Spi is configured as a serial slave

**SPIM\_SPIENR**

Address: Operational Base + offset( 0x08)

Spi enable register

bit	Attr	Reset Value	Description
0	RW	0x0	SPI Enable. Enables and disables all Spi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the Spi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

**SPIM\_MWCR**

Address: Operational Base + offset(0x0C)

Microwire control register

bit	Attr	Reset Value	Description
2	RW	0x0	Microwire Handshaking.
1	RW	0x0	Microwire Control.
0	RW	0x0	Microwire Transfer Mode.

**SPIM\_SER**

Address: Operational Base + offset(0x10)

Slave enable register

bit	Attr	Reset Value	Description
1:0	RW	0x0	Slave Select Enable Flag. Each bit in this register corresponds to a slave select line (ss_x_n) from the Spi master.

**SPIM\_BAUDR**

Address: Operational Base + offset(0x14)

Baud rate select

Bit	Attr	Reset Value	Description
15:0	RW	0x0	SPI Clock Divider. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation: $F_{sclk\_out} = F_{ssi\_clk} / SCKDV$ where SCKDV is any even value between 2 and 65534. For example: for $F_{ssi\_clk} = 3.6864\text{MHz}$ and $SCKDV = 2$ $F_{sclk\_out} = 3.6864 / 2 = 1.8432\text{MHz}$

**SPIM\_TXFTLR**

Address: Operational Base + offset(0x18)

Transmit FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.

**SPIM\_RXFTLR**

Address: Operational Base + offset(0x1C)

Receive FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

**SPIM\_TXFLR**

Address: Operational Base + offset(0x20)

Transmit FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

**SPIM\_RXFLR**

Address: Operational Base + offset(0x24)

Receive FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO

**SPIM\_SR**

Address: Operational Base + offset(0x28)

Status Register

bit	Attr	Reset Value	Description
6	R	0x0	Data Collision Error. Relevant only when the Spi is configured as a master device. This bit is set if the Spi master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error
5	R	0x0	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the Spi is configured as a slave device. Data from the previous transmission is resent on the txd line. This bit is cleared when read. 0 – No error 1 – Transmission error
4	R	0x0	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 – Receive FIFO is not full 1 – Receive FIFO is full
3	R	0x0	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty
2	R	0x1	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty
1	R	0x1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full
0	R	0x0	SPI Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that the Spi is idle or disabled. 0 – Spi is idle or disabled 1 – Spi is actively transferring data

**SPIM\_IMR**

Address: Operational Base + offset(0x2C)

Interrupt Mask Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	RW	0x1	Multi-Master Contention Interrupt Mask. This bit field is not present if the spi is configured as a serial-slave device. 0 – ssi_mst_intr interrupt is masked

			1 – ssi_mst_intr interrupt is not masked
4	RW	0x1	Receive FIFO Full Interrupt Mask 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked
3	RW	0x1	Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked
2	RW	0x1	Receive FIFO Underflow Interrupt Mask 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked
1	RW	0x1	Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked
0	RW	0x1	Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked

**SPIM\_ISR**

Address: Operational Base + offset(0x30)

Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt not active after masking 1 = ssi_mst_intr interrupt is active after masking
4	R	0x0	Receive FIFO Full Interrupt Status 0 = ssi_rxf_intr interrupt is not active after masking 1 = ssi_rxf_intr interrupt is full after masking
3	R	0x0	Receive FIFO Overflow Interrupt Status 0 = ssi_rxo_intr interrupt is not active after masking 1 = ssi_rxo_intr interrupt is active after masking
2	R	0x0	Receive FIFO Underflow Interrupt Status 0 = ssi_rxu_intr interrupt is not active after masking 1 = ssi_rxu_intr interrupt is active after masking
1	R	0x0	Transmit FIFO Overflow Interrupt Status 0 = ssi_txo_intr interrupt is not active after masking 1 = ssi_txo_intr interrupt is active after masking
0	R	0x0	Transmit FIFO Empty Interrupt Status 0 = ssi_txe_intr interrupt is not active after masking 1 = ssi_txe_intr interrupt is active after masking

**SPIM\_RISR**

Address: Operational Base + offset(0x34)

Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt is not active prior to masking 1 = ssi_mst_intr interrupt is active prior masking
4	R	0x0	Receive FIFO Full Raw Interrupt Status 0 = ssi_rxf_intr interrupt is not active prior to masking 1 = ssi_rxf_intr interrupt is active prior to masking

3	R	0x0	Receive FIFO Overflow Raw Interrupt Status 0 = ssi_rxo_intr interrupt is not active prior to masking 1 = ssi_rxo_intr interrupt is active prior masking
2	R	0x0	Receive FIFO Underflow Raw Interrupt Status 0 = ssi_rxu_intr interrupt is not active prior to masking 1 = ssi_rxu_intr interrupt is active prior to masking
1	R	0x0	Transmit FIFO Overflow Raw Interrupt Status 0 = ssi_txo_intr interrupt is not active prior to masking 1 = ssi_txo_intr interrupt is active prior masking
0	R	0x0	Transmit FIFO Empty Raw Interrupt Status 0 = ssi_txe_intr interrupt is not active prior to masking 1 = ssi_txe_intr interrupt is active prior masking

**SPIM\_TXOICR**

Address: Operational Base + offset(0x38)

Transmit FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

**SPIM\_RXOICR**

Address: Operational Base + offset(0x3C)

Receive FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.

**SPIM\_RXUICR**

Address: Operational Base + offset(0x40)

Receive FIFO Underflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect.

**SPIM\_MSTICR**

Address: Operational Base + offset(0x44)

Multi-Master Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect.

**SPIM\_ICR**

Address: Operational Base + offset(0x48)

Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the

			ssi_mst_intr interrupts. Writing to this register has no effect.
--	--	--	------------------------------------------------------------------

**SPIM\_DMCCR**

Address: Operational Base + offset(0x4C)

DMA Control Register

bit	Attr	Reset Value	Description
1	RW	0x0	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
0	RW	0x0	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 = Receive DMA disabled 1 = Receive DMA enabled

**SPIM\_DMATDLR**

Address: Operational Base + offset(0x50)

DMA Transmit Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

**SPIM\_DMARDLR**

Address: Operational Base + offset(0x54)

DMA Receive Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.

**SPIM\_IDR**

Address: Operational Base + offset(0x58)

Identification Register

bit	Attr	Reset Value	Description
31:0	R	0xffffffff	Identification Code.

**SPIM\_SPI\_COMP\_VERSION**

Address: Operational Base + offset(0x5C)

coreKit version ID register

bit	Attr	Reset Value	Description
31:0	R	0x3331302a	Contains the hex representation of the component version.

**SPIM\_DR**

Address: Operational Base + offset(0x60-9C)

Data Register

bit	Attr	Reset Value	Description
-----	------	-------------	-------------



15:0	RW	0x0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer
------	----	-----	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 18.4 Functional Description

### 18.4.1 Operation

#### ◆ Operation Modes

The Spi can be configured in the following two fundamental modes of operation: Master Mode, Slave Mode. In our design, SPI0 work as master mode with ID 0xffff\_ffff and SPI1 work as slave mode with ID 0xffff\_ffff1.

#### ◆ Transfer Modes

The Spi operates in the following four modes when transferring data on the serial bus:

##### 1. Transmit and Receive:

When TMOD = 2'b00, both transmit and receive logic are valid.

##### 2. Transmit Only:

When TMOD = 2'b01, the receive data are invalid and should not be stored in the receive FIFO.

##### 3. Receive Only:

When TMOD = 2'b10, the transmit data are invalid.

##### 4. EEPROM Read:

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). When the transmit FIFO becomes empty (all control information has been sent), data on the receive line (rxd) is valid and is stored in the receive FIFO. The serial transfer continues until the number of data frames received by the Spi master matches the value of the NDF field in the CTRLR1 register + 1

#### ◆ Clock Ratios

A summary of the frequency ratio restrictions between the bit-rate clock (sclk\_out/sclk\_in) and the Spi peripheral clock (ssi\_clk) are described as:

Master:  $F_{ssi\_clk} \geq 2 \times (\text{maximum } F_{sclk\_out})$

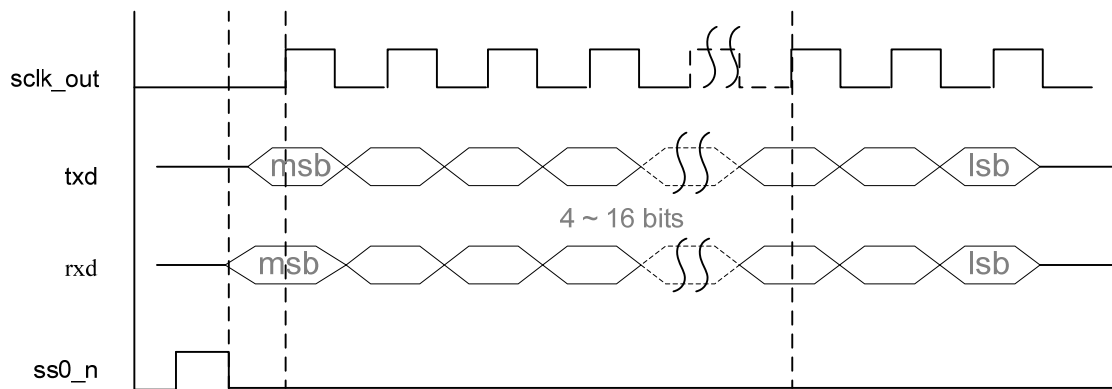
Slave (receive only):  $F_{ssi\_clk} \geq 6 \times (\text{maximum } F_{sclk\_in})$

Slave:  $F_{ssi\_clk} \geq 8 \times (\text{maximum } F_{sclk\_in})$

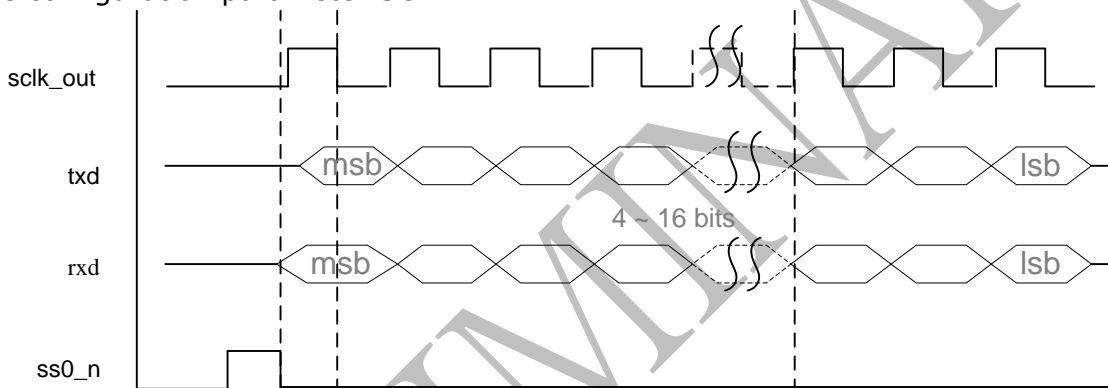
#### ◆ SPI Operation

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 16 bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. Figure 2 shows a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPOL = 1.

Fig. 18-2 SPI Master Serial Format timing diagram( $SCPH = 0$ )

When the configuration parameter  $SCPH = 1$ , both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. Figure 3 shows the timing diagram for the SPI format when the configuration parameter  $SCPH = 1$ .

Fig. 18-3 SPI Master Serial Format timing diagram ( $SCPH = 1$ )

### 18.4.2 Programming sequence

#### ● Master Transfer Flow

When configured as a serial-master device, the spi initiates and controls all serial transfers. The serial bit-rate clock, generated and controlled by the spi, is driven out on the `sclk_out` line. When the spi is disabled ( $SPI\_EN = 0$ ), no serial transfers can occur and `sclk_out` is held in "inactive" state, as defined by the serial protocol under which it operates.

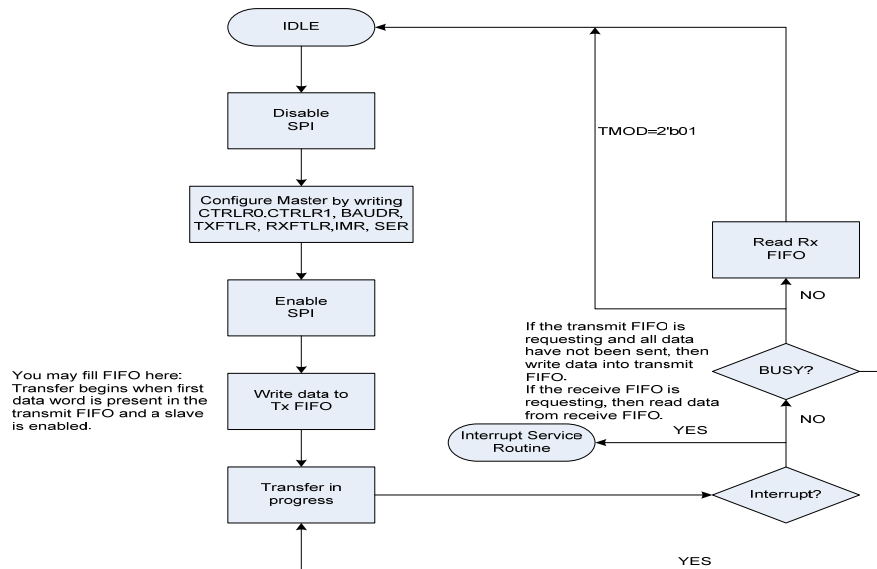


Fig. 18-4 SPI Master transfer flow diagram

## Chapter 19 SPI Slave Controller

### 19.1 Design Overview

#### 19.1.1 Overview

The Serial Peripheral Interface(SPI) is an APB slave device. A four-wire, full-duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The slave select line is held high when the spi is idle or disabled. SPI slave controller only work as slave mode.

#### 19.1.2 Features

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- Serial-slave operation – Enables serial communication with serial-master peripheral devices.
- DMA Controller Interface – Enables interface to a DMA controller over the AMBA bus using a handshaking interface for transfer requests.
- Support interrupt interface to interrupt controller, and independently masking of interrupts.
- Dynamic control of the serial bit rate of the data transfer.
- Two 16x16 fifos for transferring and receiving use respectively.

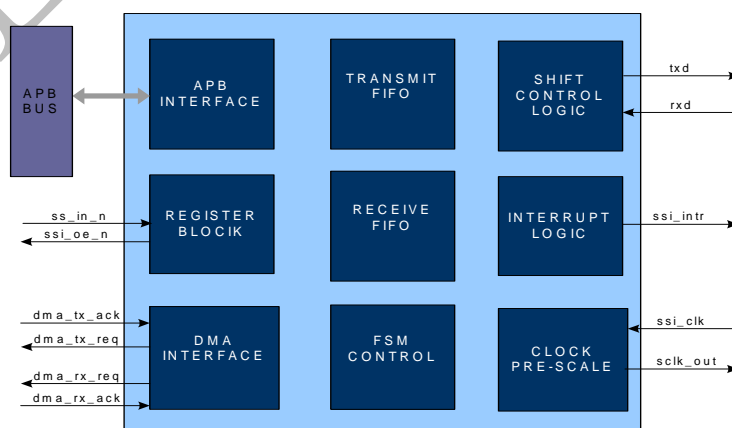
### 19.2 Architecture

This section provides a description about the functions and behavior under various conditions.

#### 19.2.1 Block Diagram

The SPI comprises with:

- ◆ AMBA APB interface and DMA Controller Interface
- ◆ Transmit and receive FIFO controllers and an FSM controller
- ◆ Register block



- ◆ Shift control and interrupt

Fig. 19-1 RK281x SPI Slave Controller Block diagram

#### 19.2.2 Block Descriptions

##### APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

##### DMA INTERFACE

This block has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA Controller.

### FIFO LOGIC

For transmit and receive transfers, data transmitted from the SPI to the external serial device is written into the transmit FIFO. Data received from the external serial device into the SPI is pushed into the receive FIFO. Both fifos are 16x16bits.

### FSM CONTROL

Control the state's transformation of the design.

### REGISTER BLOCK

All registers in the SPI are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bits wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

### SHIFT CONTROL

Shift control logic shift the data from the transmit fifo or to the receive fifo. This logic automatically right-justifies receive data in the receive FIFO buffer

### INTERRUPT CONTROL

The SPI supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the Ored result of all other SPI interrupts after masking.

## 19.3 Registers

This section describes the control/status registers of the design.

### 19.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SPIS_CTRLR0	0x0000	W	0x0000_0007	Control register 0
SPIS_CTRLR1	0x0004	W	0x0000_0000	Control register 1
SPIS_SPIENR	0x0008	W	0x0000_0000	Ssi enable register
SPIS_MWCR	0x000C	W	0x0000_0000	Microwire control register
SPIS_TXFTLR	0x0018	W	0x0000_0000	Transmit FIFO Threshold Level
SPIS_RXFTLR	0x001c	W	0x0000_0000	Receive FIFO Threshold Level
SPIS_TXFLR	0x0020	W	0x0000_0000	Transmit FIFO Level Register
SPIS_RXFLR	0x0024	W	0x0000_0000	Receive FIFO Level Register
SPIS_SR	0x0028	W	0x0000_0006	Status Register
SPIS_IMR	0x002c	W	0x0000_001f	Interrupt Mask Register
SPIS_ISR	0x0030	W	0x0000_0000	Interrupt Status Register
SPIS_RISR	0x0034	W	0x0000_0000	Raw Interrupt Status Register
SPIS_TXOICR	0x0038	W	0x0000_0000	Transmit FIFO Overflow Interrupt Clear Register
SPIS_RXOICR	0x003c	W	0x0000_0000	Receive FIFO Overflow Interrupt Clear Register
SPIS_RXUICR	0x0040	W	0x0000_0000	Receive FIFO Underflow Interrupt Clear Register
SPIS_MSTICR	0x0044	W	0x0000_0000	Multi-Master Interrupt Clear Register
SPIS_ICR	0x0048	W	0x0000_0000	Interrupt Clear Register
SPIS_DMACR	0x004c	W	0x0000_0000	DMA Control Register
SPIS_DMATDLR	0x0050	W	0x0000_0000	DMA Transmit Data Level

SPIS_DMARDL R	0x0054	W	0x0000_0000	DMA Receive Data Level
SPIS_IDR	0x0058	W	0xffff_fff1	Identification Register
SPIS_COMP_V ERSION	0x005c	W	0x3331_302a	coreKit version ID register
SPIS_DR	0x0060 -9C	W	0x0000_0000	Data Register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 19.3.2 Detail Register Description

#### SPIS\_CTRL0

Address: Operational Base + offset( 0x00)

Control register 0

bit	Attr	Reset Value	Description
15:12	RW	0X0	Control Frame Size. Selects the length of the control word for the Microwire frame format. For the field decode,
11	RW	0x0	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 – Normal Mode Operation 1 – Test Mode Operation
10	RW	0x0	Slave Output Enable.
9:8	RW	0x0	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. 00 -- Transmit & Receive 01 -- Transmit Only 10 -- Receive Only 11 -- EEPROM Read
7	RW	0x0	Serial Clock Polarity 0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high
6	RW	0x0	Serial Clock Phase 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit
5:4	RW	0x0	Frame Format. Selects which serial protocol transfers the data. 00 -- Motorola SPI 01 -- Texas Instruments SSP 10 -- National Semiconductors Microwire 11 -- Reserved
3:0	RW	0x7	Data Frame Size. Selects the data frame length . When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. 0000: Reserved – undefined operation 0001: Reserved – undefined operation 0010: Reserved – undefined operation

			0011: 4-bit serial data transfer 0100: 5-bit serial data transfer 0101: 6-bit serial data transfer 0110: 7-bit serial data transfer 0111: 8-bit serial data transfer 1000: 9-bit serial data transfer 1001: 10-bit serial data transfer 1010: 11-bit serial data transfer 1011: 12-bit serial data transfer 1100: 13-bit serial data transfer 1101: 14-bit serial data transfer 1110: 15-bit serial data transfer 1111: 16-bit serial data transfer
--	--	--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**SPIS\_CTRL1**

Address: Operational Base + offset( 0x04)

Control register 1

bit	Attr	Reset Value	Description
15:0	RW	0x0	Number of Data Frames. When TMOD = 10, this register field sets the number of data frames to be continuously received by the SPI. The Spi continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. When the SPI is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the Spi is configured as a serial slave

**SPIS\_SPIENR**

Address: Operational Base + offset( 0x08)

Spi enable register

bit	Attr	Reset Value	Description
0	RW	0x0	SPI Enable. Enables and disables all Spi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the Spi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

**SPIS\_MWCR**

Address: Operational Base + offset(0x0C)

Microwire control register

bit	Attr	Reset Value	Description
2	RW	0x0	Microwire Handshaking.
1	RW	0x0	Microwire Control.
0	RW	0x0	Microwire Transfer Mode.

**SPIS\_TXFTLR**

Address: Operational Base + offset(0x18)

Transmit FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved



3:0	RW	0x0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.
-----	----	-----	--------------------------------------------------------------------------------------------------------------------------------

**SPIS\_RXFTLR**

Address: Operational Base + offset(0x1C)

Receive FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

**SPIS\_TXFLR**

Address: Operational Base + offset(0x20)

Transmit FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

**SPIS\_RXFLR**

Address: Operational Base + offset(0x24)

Receive FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO

**SPIS\_SR**

Address: Operational Base + offset(0x28)

Status Register

bit	Attr	Reset Value	Description
6	R	0x0	Data Collision Error. Relevant only when the Spi is configured as a master device. This bit is set if the Spi master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error
5	R	0x0	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the Spi is configured as a slave device. Data from the previous transmission is resent on the txd line. This bit is cleared when read. 0 – No error 1 – Transmission error
4	R	0x0	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 – Receive FIFO is not full 1 – Receive FIFO is full
3	R	0x0	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by

			software to completely empty the receive FIFO. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty
2	R	0x1	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty
1	R	0x1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full
0	R	0x0	SPI Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that the Spi is idle or disabled. 0 – Spi is idle or disabled 1 – Spi is actively transferring data

**SPIS\_IMR**

Address: Operational Base + offset(0x2C)

Interrupt Mask Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	RW	0x1	Multi-Master Contention Interrupt Mask. This bit field is not present if the spi is configured as a serial-slave device. 0 – ssi_mst_intr interrupt is masked 1 – ssi_mst_intr interrupt is not masked
4	RW	0x1	Receive FIFO Full Interrupt Mask 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked
3	RW	0x1	Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked
2	RW	0x1	Receive FIFO Underflow Interrupt Mask 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked
1	RW	0x1	Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked
0	RW	0x1	Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked

**SPIS\_ISR**

Address: Operational Base + offset(0x30)

Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt not active after masking 1 = ssi_mst_intr interrupt is active after masking
4	R	0x0	Receive FIFO Full Interrupt Status

			0 = ssi_rxf_intr interrupt is not active after masking 1 = ssi_rxf_intr interrupt is full after masking
3	R	0x0	Receive FIFO Overflow Interrupt Status 0 = ssi_rxo_intr interrupt is not active after masking 1 = ssi_rxo_intr interrupt is active after masking
2	R	0x0	Receive FIFO Underflow Interrupt Status 0 = ssi_rxu_intr interrupt is not active after masking 1 = ssi_rxu_intr interrupt is active after masking
1	R	0x0	Transmit FIFO Overflow Interrupt Status 0 = ssi_txo_intr interrupt is not active after masking 1 = ssi_txo_intr interrupt is active after masking
0	R	0x0	Transmit FIFO Empty Interrupt Status 0 = ssi_txe_intr interrupt is not active after masking 1 = ssi_txe_intr interrupt is active after masking

**SPIS\_RISR**

Address: Operational Base + offset(0x34)

Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt is not active prior to masking 1 = ssi_mst_intr interrupt is active prior masking
4	R	0x0	Receive FIFO Full Raw Interrupt Status 0 = ssi_rxf_intr interrupt is not active prior to masking 1 = ssi_rxf_intr interrupt is active prior to masking
3	R	0x0	Receive FIFO Overflow Raw Interrupt Status 0 = ssi_rxo_intr interrupt is not active prior to masking 1 = ssi_rxo_intr interrupt is active prior masking
2	R	0x0	Receive FIFO Underflow Raw Interrupt Status 0 = ssi_rxu_intr interrupt is not active prior to masking 1 = ssi_rxu_intr interrupt is active prior to masking
1	R	0x0	Transmit FIFO Overflow Raw Interrupt Status 0 = ssi_txo_intr interrupt is not active prior to masking 1 = ssi_txo_intr interrupt is active prior masking
0	R	0x0	Transmit FIFO Empty Raw Interrupt Status 0 = ssi_txe_intr interrupt is not active prior to masking 1 = ssi_txe_intr interrupt is active prior masking

**SPIS\_TXOICR**

Address: Operational Base + offset(0x38)

Transmit FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

**SPIS\_RXOICR**

Address: Operational Base + offset(0x3C)

Receive FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this

			register clears the ssi_rxo_intr interrupt; writing has no effect.
--	--	--	--------------------------------------------------------------------

**SPIS\_RXUICR**

Address: Operational Base + offset(0x40)

Receive FIFO Underflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect

**SPIS\_MSTICR**

Address: Operational Base + offset(0x44)

Multi-Master Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect.

**SPIS\_ICR**

Address: Operational Base + offset(0x48)

Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.

**SPIS\_DMACR**

Address: Operational Base + offset(0x4C)

DMA Control Register

bit	Attr	Reset Value	Description
1	RW	0x0	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
0	RW	0x0	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 = Receive DMA disabled 1 = Receive DMA enabled

**SPIS\_DMATDLR**

Address: Operational Base + offset(0x50)

DMA Transmit Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

**SPIS\_DMARDLR**

Address: Operational Base + offset(0x54)

DMA Receive Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.

### SPIS\_IDR

Address: Operational Base + offset(0x58)

Identification Register

bit	Attr	Reset Value	Description
31:0	R	0xffffffff	Identification Code.

### SPIS\_COMP\_VERSION

Address: Operational Base + offset(0x5C)

coreKit version ID register

bit	Attr	Reset Value	Description
31:0	R	0x3331302a	Contains the hex representation of the component version.

### SPIS\_DR

Address: Operational Base + offset(0x60-9C)

Data Register

bit	Attr	Reset Value	Description
15:0	RW	0x0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 19.4 Functional Description

### 19.4.1 Operation

#### ◆ Operation Modes

The Spi can be configured in the following two fundamental modes of operation: Master Mode, Slave Mode. In our design, SPI0 work as master mode with ID 0xffff\_ffff and SPI1 work as slave mode with ID 0xffff\_ffff1.

#### ◆ Transfer Modes

The Spi operates in the following four modes when transferring data on the serial bus:

##### 1. Transmit and Receive:

When TMOD = 2'b00, both transmit and receive logic are valid.

##### 2. Transmit Only:

When TMOD = 2'b01, the receive data are invalid and should not be stored in the receive FIFO.

##### 3. Receive Only:

When TMOD = 2'b10, the transmit data are invalid.

##### 4. EEPROM Read:

When TMOD = 2'b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). When the transmit FIFO becomes empty (all control information has been sent), data on the receive line (rxd) is

valid and is stored in the receive FIFO. The serial transfer continues until the number of data frames received by the Spi master matches the value of the NDF field in the CTRLR1 register + 1

#### ◆ Clock Ratios

A summary of the frequency ratio restrictions between the bit-rate clock (sclk\_out/sclk\_in) and the Spi peripheral clock (ssi\_clk) are described as:

Master:  $F_{ssi\_clk} \geq 2 \times (\text{maximum } F_{sclk\_out})$

Slave (receive only):  $F_{ssi\_clk} \geq 6 \times (\text{maximum } F_{sclk\_in})$

Slave:  $F_{ssi\_clk} \geq 8 \times (\text{maximum } F_{sclk\_in})$

#### ◆ SPI Operation

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 16 bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. Figure 2 shows a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPOL = 1.

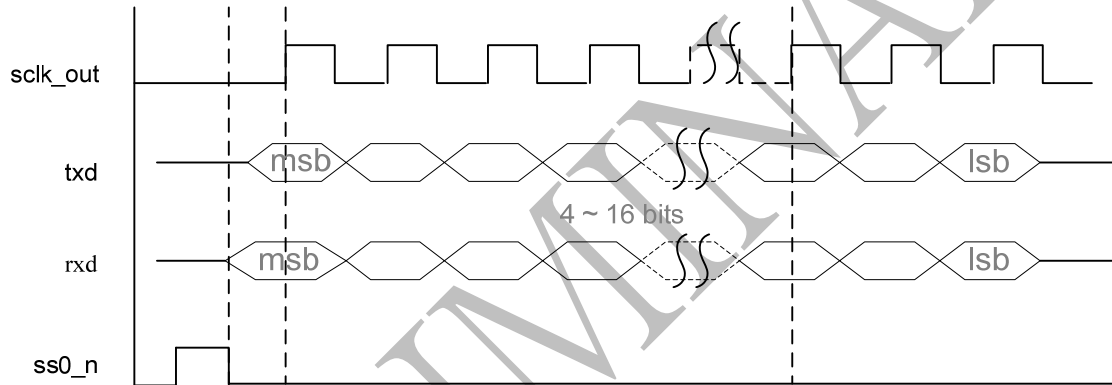


Fig. 19-2 SPI Slave Serial Format timing diagram (SCPH = 0)

When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. Figure 3 shows the timing diagram for the SPI format when the configuration parameter SCPH = 1.

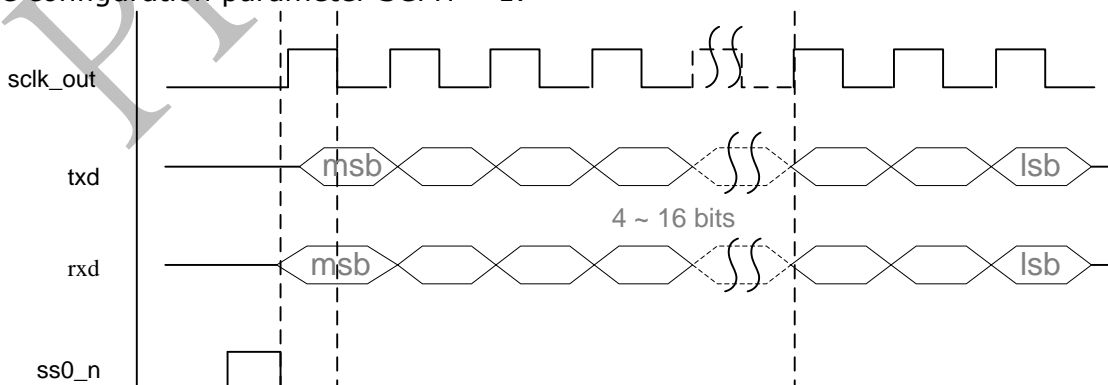


Fig. 19-3 SPI Slave Serial Format timing diagram (SCPH = 1)

### 19.4.2 Programming sequence

- **Slave Transfer Flow**

When the SPI is configured as a slave device, all serial transfers are initiated and controlled by the serial bus master.

When the SPI serial slave is selected during configuration, it enables its txd data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line (sclk\_in), driven from the serial-master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

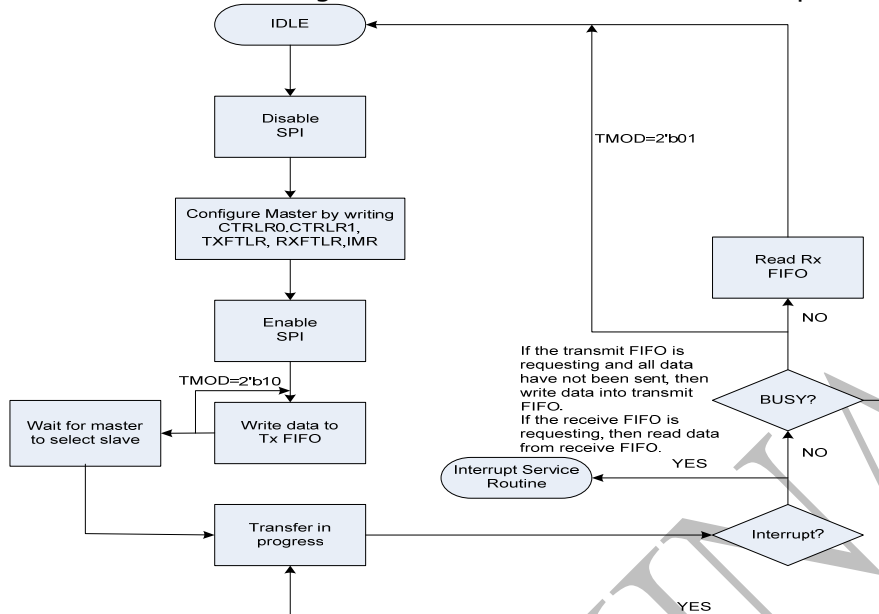


Fig. 19-4 SPI Slave transfer flow diagram



## Chapter 20 Timers in CPU system

### 20.1 Design Overview

#### 20.1.1 Overview

Timers is a programmable timers peripheral. This component is an APB slave device. Timers count down from a programmed value and generate an interrupt when the count reaches zero.

#### 20.1.2 Features

- Three programmable 32 bits timers
- Two operation modes: free-running and user-defined count
- The clock of all timers is pclk
- Maskable for each individual interrupt

#### 20.1.3 Block Diagram

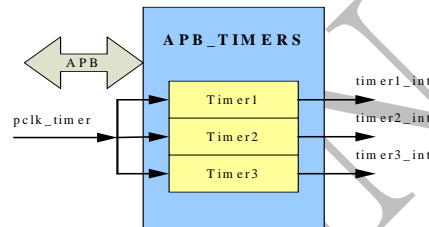


Fig. 20-1 Timers Block Diagram in CPU System

### 20.2 Registers

This chapter describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses.

#### 20.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
Timer1LoadCount	0x0000	W	0x00000000	Timer1 Load Count Register
Timer1CurrentValue	0x0004	W	0x00000000	Timer1 Current Value Register
Timer1ControlReg	0x0008	W	0x00000000	Timer1 Control Register
Timer1EOI	0x000C	W	0x00000000	Timer1 End-of-Interrupt Register
Timer1IntStatus	0x0010	W	0x00000000	Timer1 Interrupt Status Register
Timer2LoadCount	0x0014	W	0x00000000	Timer2 Load Count Register
Timer2CurrentValue	0x0018	W	0x00000000	Timer2 Current Value Register
Timer2ControlReg	0x001c	W	0x00000000	Timer2 Control Register
Timer2EOI	0x0020	W	0x00000000	Timer2 End-of-Interrupt Register
Timer2IntStatus	0x0024	W	0x00000000	Timer2 Interrupt Status Register
Timer3LoadCount	0x0028	W	0x00000000	Timer3 Load Count Register
Timer3CurrentValue	0x002c	W	0x00000000	Timer3 Current Value Register
Timer3ControlReg	0x0030	W	0x00000000	Timer3 Control Register
Timer3EOI	0x0034	W	0x00000000	Timer3 End-of-Interrupt Register

Timer3IntStatus	0x0038	W	0x00000000	Timer3 Interrupt Status Register
TimersIntStatus	0x00a0	W	0x00000000	Timers Interrupt Status Register
TimersEOI	0x00a4	W	0x00000000	Timers End-of-Interrupt Register
TimersRawIntStatus	0x00a8	W	0x00000000	Timers Raw Interrupt Status Register

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

## 20.2.2 Detail Register Description

### Timer1LoadCount

Address: Operational Base + offset(0x00)

Timer1 Load Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Value to be loaded into Timer1. This is the value from which counting commences.

### Timer1CurrentValue

Address: Operational Base + offset(0x04)

Timer1 Current Value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000	Current Value of Timer1.

### Timer1ControlReg

Address: Operational Base + offset(0x08)

Timer1 Control Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 0: not mask 1: mask
1	RW	0x0	Timer mode. 0: free-running mode 1: user-defined count mode
0	RW	0x0	Timer enable. 0: disable 1: enable

### Timer1EOI

Address: Operational Base + offset(0x0C)

Timer1 End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from timer1

### Timer1IntStatus

Address: Operational Base + offset(0x10)

Timer1 Interrupt Status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register contains the interrupt status for timer1

**Timer2LoadCount**

Address: Operational Base + offset(0x14)

Timer2 Load Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Value to be loaded into Timer2. This is the value from which counting commences.

**Timer2CurrentValue**

Address: Operational Base + offset(0x18)

Timer2 Current Value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000	Current Value of Timer2.

**Timer2ControlReg**

Address: Operational Base + offset(0x1c)

Timer2 Control Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 0: not mask 1: mask
1	RW	0x0	Timer mode. 0: free-running mode 1: user-defined count mode
0	RW	0x0	Timer enable. 0: disable 1: enable

**Timer2EOI**

Address: Operational Base + offset(0x20)

Timer2 End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from timer2

**Timer2IntStatus**

Address: Operational Base + offset(0x24)

Timer2 Interrupt Status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register contains the interrupt status for timer2

**Timer3LoadCount**

Address: Operational Base + offset(0x28)

Timer3 Load Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Value to be loaded into Timer3. This is the value from which counting commences.

**Timer3CurrentValue**

Address: Operational Base + offset(0x2c)

Timer3 Current Value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000	Current Value of Timer3.

**Timer3ControlReg**

Address: Operational Base + offset(0x30)

Timer3 Control Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 0: not mask 1: mask
1	RW	0x0	Timer mode. 0: free-running mode 1: user-defined count mode
0	RW	0x0	Timer enable. 0: disable 1: enable

**Timer3EOI**

Address: Operational Base + offset(0x34)

Timer3 End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from timer3

**Timer3IntStatus**

Address: Operational Base + offset(0x38)

Timer3 Interrupt Status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register contains the interrupt status for timer3

**TimersIntStatus**

Address: Operational Base + offset(0xa0)

Timers Interrupt Status Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	This register contains the interrupt status for timer3
1	R	0x0	This register contains the interrupt status for timer2
0	R	0x0	This register contains the interrupt status for timer1

**TimersEOI**

Address: Operational Base + offset(0xa4)

Timers End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2:0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from all timers

**TimersRawIntStatus**

Address: Operational Base + offset(0xa8)

Timers Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	This register contains the interrupt status for timer3 prior to masking
1	R	0x0	This register contains the interrupt status for timer2

			prior to masking
0	R	0x0	This register contains the interrupt status for timer1 prior to masking

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 23.3 Functional Description

### 23.3.1 Operation

#### Timer operation

Timers count down from a programmed value and generate an interrupt when the count reaches zero.

The input clock for each timer is pclk\_timer from SCU block(Chapter16).

The initial value for each timer – that is, the value from which it counts down – is loaded using the appropriate load count register (TimerNLoadCount). Two events can cause a timer to load the initial count from its TimerNLoadCount register:

- Timer is enabled after being reset or disabled
- Timer counts down to 0

All interrupt status registers and end-of-interrupt registers can be accessed at any time.

#### Configuration

Timers contain three identical but separately-programmable timers, which are accessed through a single AMBA APB interface.

### 23.3.2 Programming sequence

1. Initialize the timer through the TimerNControlReg register (where N is in the range 1 to 3):
  - a. Disable the timer by writing a "0" to the timer enable bit (bit 0); accordingly, the timer\_en output signal is de-asserted.
  - b. Program the timer mode—user-defined or free-running—by writing a "0" or "1," respectively, to the timer mode bit (bit 1).
  - c. Set the interrupt mask as either masked or not masked by writing a "0" or "1," respectively, to the timer interrupt mask bit (bit 2).
2. Load the timer counter value into the TimerNLoadCount register (where N is in the range 1 to 3).
3. Enable the timer by writing a "1" to bit 0 of TimerNControlReg.

#### Timers Ustage flow

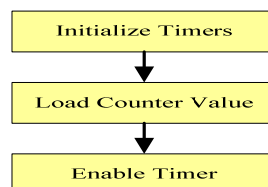


Fig. 20-2 Timers Ustage Flow in CPU System

## Chapter 21 Watchdog Timer (WDT)

### 21.1 Design Overview

#### 21.1.1 Overview

Watchdog Timer (WDT) is an APB slave peripheral that can be used to prevent system lockup that may be caused by conflicting parts or programs in an SoC. The WDT would generate an interrupt or reset signal when its counter reaches zero, then a reset controller would reset the system.

#### 21.1.2 Features

- 32 bits APB bus width
- WDT counter's clock is pclk
- 32 bits WDT counter width
- Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
- WDT can perform two types of operations when timeout occurs:
  - (1) Generate a system reset;
  - (2) First generate an interrupt and if this is not cleared by the service routine by the time a second timeout occurs then generate a system reset
- Programmable reset pulse length
- Total 16 defined-ranges of main timeout period

### 21.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

#### 21.2.1 Block Diagram

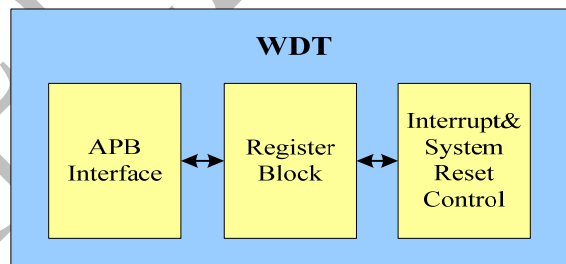


Fig. 21-1 WDT Block Diagram

#### 21.2.2 Block Descriptions

##### APB Interface

The APB Interface implements the APB slave operation. Its bus width is 32 bits.

##### Register Block

A register block with read coherency for the current count register.

##### Interrupt & system reset control

An interrupt/system reset generation block comprising of a decrementing counter and control logic.

## 21.3 Registers

This chapter describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses.

### 25.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
WDT_CR	0x0000	W	0x0000000a	Control Register
WDT_TORR	0x0004	W	0x00000000	Timeout range Register
WDT_CCVR	0x0008	W	0x0000ffff	Current counter value Register
WDT_CRR	0x000C	W	0x00000000	Counter restart Register
WDT_STAT	0x0010	W	0x00000000	Interrupt status Register
WDT_EOI	0x0014	W	0x00000000	Interrupt clear Register

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 21.3.2 Detail Register Description

#### WDT\_CR

Address: Operational Base + offset(0x00)

Control Register

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4:2	RW	0x2	Reset pulse length. This is used to select the number of pclk cycles for which the system reset stays asserted. 000: 2 pclk cycles 001: 4 pclk cycles 010: 8 pclk cycles 011: 16 pclk cycles 100: 32 pclk cycles 101: 64 pclk cycles 110: 128 pclk cycles 111: 256 pclk cycles
1	RW	0x1	Response mode. Selects the output response generated to a timeout. 0: Generate a system reset. 1: First generate an interrupt and if it is not cleared by the time a second timeout occurs then generate a system reset.
0	RW	0x0	WDT enable. 0: WDT disabled. 1: WDT enabled.

#### WDT\_TORR

Address: Operational Base + offset(0x04)

Timeout range Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	Timeout period. This field is used to select the timeout period from which the watchdog counter restarts. A change of the timeout period takes effect only after the next counter restart (kick). The range of values available for a 32-bit watchdog



			counter are: 0000: 0x0000ffff 0001: 0x0001ffff 0010: 0x0003ffff 0011: 0x0007ffff 0100: 0x000fffff 0101: 0x001fffff 0110: 0x003fffff 0111: 0x007fffff 1000: 0x00ffffff 1001: 0x01ffffff 1010: 0x03ffffff 1011: 0x07ffffff 1100: 0x0fffffff 1101: 0x1fffffff 1110: 0x3fffffff 1111: 0x7fffffff
--	--	--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**WDT\_CCVR**

Address: Operational Base + offset(0x08)

Current counter value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000ffff	This register, when read, is the current value of the internal counter. This value is read coherently when ever it is read

**WDT\_CRR**

Address: Operational Base + offset(0x0C)

Counter restart Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	This register is used to restart the WDT counter. As a safety feature to prevent accidental restarts, the value 0x76 must be written. A restart also clears the WDT interrupt. Reading this register returns zero.

**WDT\_STAT**

Address: Operational Base + offset(0x10)

Interrupt status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register shows the interrupt status of the WDT. 1: Interrupt is active regardless of polarity. 0: Interrupt is inactive.

**WDT\_EOI**

Address: Operational Base + offset(0x14)

Interrupt clear Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Clears the watchdog interrupt. This can be used to clear the interrupt without restarting the watchdog counter.

## 21.4 Functional Description

### 21.4.1 Operation

#### Counter

The WDT counts from a preset (timeout) value in descending order to zero. When the counter reaches zero, depending on the output response mode selected, either a system reset or an interrupt occurs. When the counter reaches zero, it wraps to the selected timeout value and continues decrementing. The user can restart the counter to its initial value. This is programmed by writing to the restart register at any time. The process of restarting the watchdog counter is sometimes referred to as kicking the dog. As a safety feature to prevent accidental restarts, the value 0x76 must be written to the Current Counter Value Register (WDT\_CRR).

#### Interrupts

The WDT can be programmed to generate an interrupt (and then a system reset) when a timeout occurs. When a 1 is written to the response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT\_CR), the WDT generates an interrupt. If it is not cleared by the time a second timeout occurs, then it generates a system reset. If a restart occurs at the same time the watchdog counter reaches zero, an interrupt is not generated.

#### System Resets

When a 0 is written to the output response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT\_CR), the WDT generates a system reset when a timeout occurs.

#### Reset Pulse Length

The reset pulse length is the number of pclk cycles for which a system reset is asserted. When a system reset is generated, it remains asserted for the number of cycles specified by the reset pulse length or until the system is reset. A counter restart has no effect on the system reset once it has been asserted.

### 21.4.2 Programming sequence

#### Operation Flow Chart (Response mode=1)

- 1 Select required timeout period.
- 2 Set reset pulse length, response mode, and enable WDT.
- 3 Write 0x76 to WDT\_CRR.
- 4 Starts back to selected timeout period.
- 5 Can clear by reading WDT\_EOI or restarting (kicking) the counter by writing 0x76 to WDT\_CRR.

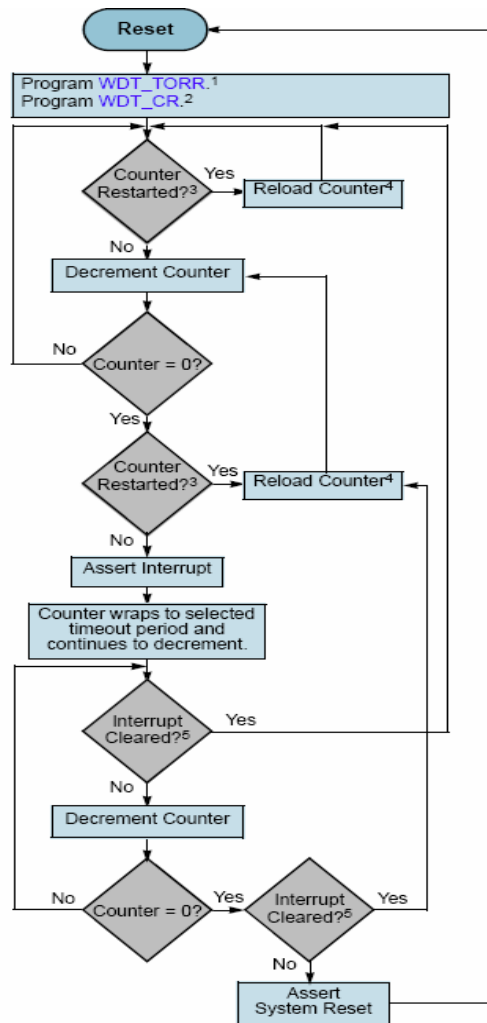


Fig. 21-2 WDT Operation Flow

## Chapter 22 Real Time Clock (RTC)

### 22.1 Design Overview

#### 22.1.1 Overview

The Real Time Clock (RTC) is an APB slave device. It can be used to provide a basic alarm function. This is achieved by generating an interrupt signal at a programmed time. Counting base on 1 Hz clock . The RTC also provides a system wakeup. The RTC core power and IO power is isolated from system power, so the RTC counter can running continuously during system power off.

#### 22.1.2 Features

- 24 hour time mode with highest precision of one second
- Provides Year, Month, Day, Weekday, Hours, Minutes and Seconds Information based on 32.768KHz quartz crystal
- Century Flag
- Programmable alarm with interrupt generation
- Maskable interrupt
- The interrupt mode configurable
- Programmable alarm wake up system power with output control pin
- Storage system information on the power off mode

### 22.2 Architecture

This section provides a description about the functions and behavior under various conditions.

#### 22.2.1 Block Diagram

The RTC comprises with:

- RTC\_REG - RTC/alarm registers with APB slave interface
- Clock divider for 1Hz and counter
- Counter based on the 1Hz clock
- Alarm and interrupt generator

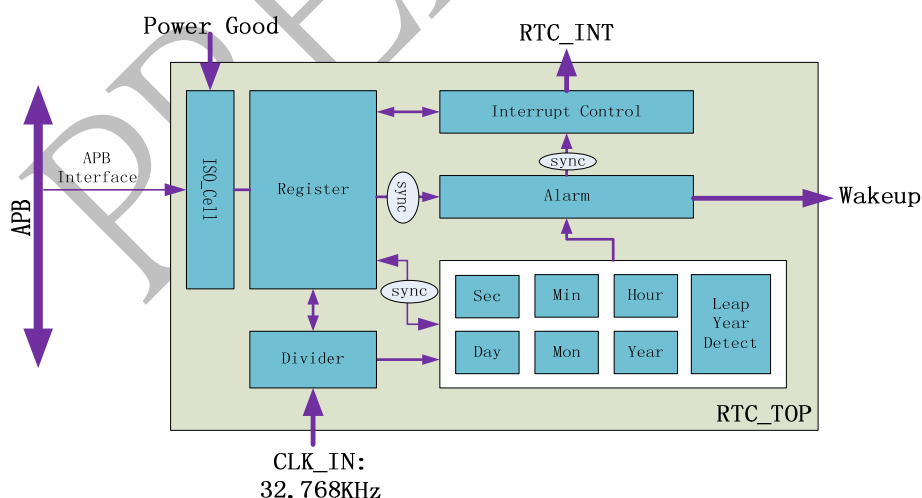


Fig. 22-1 RK281x RTC design architecture

## 22.3 Registers

This section describes the control/status registers of the design.

### 22.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
RTC_CTRL	0x0000	W	0x00000000	RTC control register.
RTC_INTS	0x0004	W	0x00000000	RTC_INT Status
RTC_TIME	0x0008	W	0x00000000	RTC time register.
RTC_DATE	0x000C	W	0x10000100	RTC date register.
RTC_TALARM	0x0010	W	0x00000000	RTC time alarm register.
RTC_DALARM	0x0014	W	0x00000000	RTC date alarm register.
RTC_ALARMEN	0x0018	W	0x00000000	RTC alarm en register
SYS_TEMP_STORAGE0	0x001C	W	0x00000000	System temp storage0
SYS_TEMP_STORAGE1	0x0020	W	0x00000000	System temp storage1
SYS_TEMP_STORAGE2	0x0024	W	0x00000000	System temp storage2
SYS_TEMP_STORAGE3	0x0028	W	0x00000000	System temp storage3
SYS_TEMP_STORAGE4	0x002C	W	0x00000000	System temp storage4
SYS_TEMP_STORAGE5	0x0030	W	0x00000000	System temp storage5

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 22.3.2 Detail Register Description

#### RTC\_CTRL

Address: Operational Base + offset( 0x00)

RTC Control register

bit	Attr	Reset Value	Description
31:7	-	-	Reserve
6	RW	0x0	WAKEUP_MODE 0: high level wakeup 1: low level wakeup
5	RW	0x0	INT_MODE 0: level sensitive 1: edge sensitive
4	RW	0x0	CNT_RST Count reset 0: No reset 1: reset (auto_clear)
3	RW	0x0	WAKEUP_EN 0: system wakeup disable 1: system wakeup enable
2	RW	0x0	INT_EN 0: interrupt disable 1: interrupt enable
1	RW	0x0	RTC_WEN(TIME/DATE) 0: write disable 1: write enable
0	RW	0x0	RTC_EN 0: RTC halt 1: RTC run

#### RTC\_INTS

Address: Operational Base + offset( 0x04)

RTC Alarm Interrupt Status register

bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:2	-	-	Reserve
0	RW	0x0	RTC_INT 0: No int 1: int, write 1 clear

**RTC\_TIME**

Address: Operational Base + offset( 0x08)

RTC Time register

bit	Attr	Reset Value	Description
31:22	-	-	Reserve
21:20	RW	0x0	TEN_HOUR Ten hours, with valid value with 0-2
19:16	RW	0x0	HOUR Hours, with valid value with 0-9
15	-	-	Reserve
14:12	RW	0x0	TEN_MIN Ten minutes, with valid value with 0-5
11:8	RW	0x0	Min Minutes, with valid value with 0-9
7	-	-	Reserve
6:4	RW	0x0	TEN_SEC Ten second, with valid value with 0-5
3:0	RW	0x0	SEC Second, with valid value with 0 to 9

\*Note: write the register, must be set RTC\_EN=0 and RTC\_WEN=1 first.

**RTC\_DATE**

Address: Operational Base + offset( 0x0C)

RTC Date register

bit	Attr	Reset Value	Description
31	-	-	Reserve
30:28	RW	0x1	WEEK Week, with valid value with 1-7
27:24	RW	0x0	TEN_CENTURY Ten century, with valid value with 0-9
23:20	RW	0x0	CENTURY Century, with valid value with 0-9
19:16	RW	0x0	TEN_YEAR Ten years, with valid value with 0-9
15:12	RW	0x0	YEAR Year, with valid value with 0-9
11:8	RW	0x1	Mon Month, with valid value with 1-12
7:6	-	-	Reserve
5:4	RW	0x0	TEN_DAY Ten days, with valid value with 0-3
3:0	RW	0x0	DAY DAY, with valid value with 0 to 9

\*Note: write the register, must be set RTC\_EN=0 and RTC\_WEN=1 first.

**RTC\_TALARM**

Address: Operational Base + offset( 0x10)

RTC Time Alarm register

bit	Attr	Reset Value	Description
31:22	-	-	Reserve
21:20	RW	0x0	TEN_HOUR Ten hours, with valid value with 0-2

19:16	RW	0x0	HOUR Hours, with valid value with 0-9
15	-	-	Reserve
14:12	RW	0x0	TEN_MIN Ten minutes, with valid value with 0-5
11:8	RW	0x0	Min Minutes, with valid value with 0-9
7	-	-	Reserve
6:4	RW	0x0	TEN_SEC Ten second, with valid value with 0-5
3:0	RW	0x0	SEC Second, with valid value with 0 to 9

**RTC\_DALARM**

Address: Operational Base + offset( 0x14)

RTC Date register

bit	Attr	Reset Value	Description
31	-	-	Reserve
30:28	RW	0x1	WEEK Week, with valid value with 1-7
27:24	RW	0x0	TEN_CENTURY Ten century, with valid value with 0-9
23:20	RW	0x0	CENTURY Century, with valid value with 0-9
19:16	RW	0x0	TEN_YEAR Ten years, with valid value with 0-9
15:12	RW	0x0	YEAR Year, with valid value with 0-9
11:8	RW	0x0	Mon Month, with valid value with 0-12
7:6	-	-	Reserve
5:4	RW	0x0	TEN_DAY Ten days, with valid value with 0-3
3:0	RW	0x0	DAY DAY, with valid value with 0 to 9

**RTC\_ALARMEN**

Address: Operational Base + offset( 0x18)

RTC Alarm en register

bit	Attr	Reset Value	Description
31:8	-	-	Reserve
7	RW	0x0	WEEK_ALARM Week alarm enable 0: disable 1: enable
6	RW	0x0	CEN_ALARM Century alarm enable 0: disable 1: enable
5	RW	0x0	YEAR_ALARM Year alarm enable 0: disable 1: enable
4	RW	0x0	MON_ALARM Month alarm enable 0: disable



			1: enable
3	RW	0x0	DAY_ALARM Days alarm enable 0: disable 1: enable
2	RW	0x0	HOUR_ALARM Hours alarm enable 0: disable 1: enable
1	RW	0x0	MIN_ALARM Minutes alarm enable 0: disable 1: enable
0	RW	0x0	SEC_ALARM Second alarm enable 0: disable 1: enable

**SYS\_TEMP\_STORAGE0**

Address: Operational Base + offset( 0x1C)

System temp storage0

bit	Attr	Reset Value	Description
31:0	RW	0x0	System temp storage0 when system power down

**SYS\_TEMP\_STORAGE1**

Address: Operational Base + offset( 0x20)

System temp storage1

bit	Attr	Reset Value	Description
31:0	RW	0x0	System temp storage1 when system power down

**SYS\_TEMP\_STORAGE2**

Address: Operational Base + offset( 0x24)

System temp storage2

bit	Attr	Reset Value	Description
31:0	RW	0x0	System temp storage2 when system power down

**SYS\_TEMP\_STORAGE3**

Address: Operational Base + offset( 0x28)

System temp storage3

bit	Attr	Reset Value	Description
31:0	RW	0x0	System temp storage3 when system power down

**SYS\_TEMP\_STORAGE4**

Address: Operational Base + offset( 0x2C)

System temp storage4

bit	Attr	Reset Value	Description
31:0	RW	0x0	System temp storage4 when system power down

**SYS\_TEMP\_STORAGE5**

Address: Operational Base + offset( 0x30)

System temp storage5

bit	Attr	Reset Value	Description
31:0	RW	0x0	System temp storage5 when system power down

## 22.4 Functional Description

### 22.4.1 Operation

#### Initialization/Changing time and date

User has one second to initialize/change all the time/date counters as following:

1. Set RTC\_WEN(bit[1] of RTC\_CTRL) to 1 and RTC\_EN(bit[0] of RTC\_CTRL) to 0.
2. Set RTC\_TIME and RTC\_DATE
3. Wait 2 seconds, the change will be effective.
4. Then set the RTC\_EN (bit[0] of RTC\_CTRL) to 1 and set RTC\_WEN ( bit[1] of RTC\_CTRL) to 0, RTC counter begin working.

#### Alarm

An Alarm can be set with any combination of century, year, month, day, day of week, hour, minute or second. To set an alarm, as following:

1. Set RTC\_WEN(bit[1] of RTC\_CTRL) to 1 and RTC\_EN(bit[0] of RTC\_CTRL) to 0.
2. Set RTC\_TIME and RTC\_DATE
3. Set RTC\_TALARM and RTC\_DALARM
4. Set the corresponding bit of RTC\_ALARMEN
5. Wait 2 seconds, the change will be effective.
6. Then set the RTC\_EN (bit[0] of RTC\_CTRL) to 1 and set RTC\_WEN ( bit[1] of RTC\_CTRL) to 0, RTC counter begin working. At the step, user can can the INT\_EN (bit[2] of RTC\_CTRL) or WAKEUP\_EN (bit[3] of RTC\_CTRL) to enable the alarm interrupt or system wakeup function.

#### Alarm interrupt

If user set the INT\_EN (bit[3] of RTC\_CTRL) to 1, the alarm wakeup function is enable. User also can set the WAKEUP\_MODE (bit[6] of RTC\_CTRL) at the step 6 of alarm to select wakeup mode, high level or low level. When the time and date set with any combination of century, year, month, day, day of week, hour, minute or second in the RTC\_ALARMEN register is same with the value set in the RTC\_TALARM and RTC\_DALARM register, the RTC module will generate an interrupt to the system. Then user can read the status in the RTC\_INTS register, and clean the status by write 1 to the bit[0].

#### System Wakeup

If user set the WAKEUP\_EN (bit[2] of RTC\_CTRL) to 1, the alarm interrupt function is enable. User also can set the INT\_MODE (bit[5] of RTC\_CTRL) at the step 6 of alarm to select interrupt mode, level or edge sensitive. Because the RTC core power and IO power is isolated from system power, so the RTC counter can running continuously during system power off.

When the time and date set with any combination of century, year, month, day, day of week, hour, minute or second in the RTC\_ALARMEN register is same with the value set in the RTC\_TALARM and RTC\_DALARM register, the RTC module will generate an wakeup signal by the output pin.

#### System Storage

Because the RTC core power and IO power is isolated from system power, so the RTC counter can running continuously during system power off. User can storage some information in the SYS\_TEMP\_STORAGE0~5 before the system power off.

## Chapter 23 I2C Controller

### 23.1 Design Overview

#### 23.1.1 Overview

The Inter-Integrated Circuit (I2C) is a two wired (SCL and SDA), bi-directional serial bus that provides an efficient and simple method of information exchange between devices. The I2C bus is a multi-master bus protocol using arbitration to avoid bus collision if two or more masters attempt to control the bus at the same time. This I2C bus controller supports both master and slave modes acting as a bridge between AMBA protocol and generic I2C bus system.

#### 23.1.2 Features

- Compatible with I2C-bus
- AMBA APB slave interface
- Supports master and slave modes of I2C bus
- Multi masters operation
- Software programmable clock frequency and transfer rate up to 400Kbit/sec
- Supports 7 bits and 10 bits addressing modes
- Interrupt or polling driven byte-by-byte data transfer

### 23.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

#### 23.2.1 Block Diagram

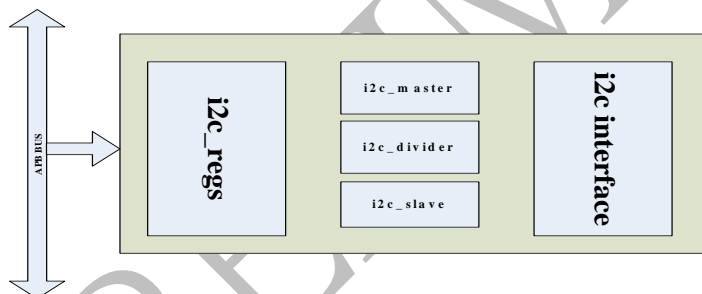


Fig. 23-1 I2C Controller design architecture

#### 23.2.2 Block Descriptions

##### i2c\_regs – Control and Status Registers

The i2c\_regs component is used to control the I2C controller operation by the host with APB interface. It implements the register set and the interrupt functionality. The CSR component operates synchronously with the pclk clock.

##### i2c\_master – I2C Master Control and State Machine

The I2C master controller implements the I2C master operation for transmit data to and receive data from other I2C devices. The I2C master controller operates synchronously with the pclk.

##### i2c\_slave – I2C Slave Control and State Machine

The I2C slave controller implements the I2C master operation for transmit data to and receive data from other I2C devices. The I2C slave controller operates synchronously with the pclk.

**i2c\_divider – Clock Divider**

The clock divider module generates I2C clock SCL output signals from pclk at frequency according the given equation.

**i2c\_interface – I2C interface**

SDA output enable from I2C master controller and slave controller are ANDed together as the output ports. Similarly, SCL output enable from I2C master controller and slave controller are ANDed together. SDA output and SCL output are actually ties to the ground since I2C is an open drain architecture. So once enabled, SDA/ SCL on I2C will be pulled low.

**23.3 Registers**

This chapter describes the control/status registers of the design.  
Software should read and write these registers using 32-bits accesses.

**23.3.1 Registers Summary**

Name	Offset	Size	Reset Value	Description
I2C_MTXR	0x0000	W	0x00000000	Master transmit register input
I2C_MRXR	0x0004	W	0x00000000	Master receive register output
I2C_STXR	0x0008	W	0x00000000	Slave transmit register input
I2C_SRXR	0x000C	W	0x00000000	Slave receive register output
I2C_SADDR	0x0010	W	0x000003FF	I2C controller slave address
I2C_IER	0x0014	W	0x00000000	Enable/Mask interrupts generated by the I2C controller
I2C_ISR	0x0018	W	0x00000000	Interrupt status register
I2C_LCMR	0x001C	W	0x00000000	I2C line command register
I2C_LSR	0x0020	W	0x00000000	I2C core status
I2C_CONR	0x0024	W	0x00000000	I2C operation register 1
I2C_OPR	0x0028	W	0x00000000	I2C operation register 2

Notes: **Size: B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

**23.3.2 Detail Register Description****I2C\_MTXR**

Address: Operational Base + offset(0x00)

This register contains data to be transmitted on the I2C for master purpose.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	I2C master transmit data register.

**I2C\_MRXR**

Address: Operational Base + offset(0x04)

This register contains data to be received on the I2C for master purpose.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved

**I2C\_STXR**

Address: Operational Base + offset(0x08)

This register contains data to be transmitted on the I2C for slave purpose.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	I2C slave transmit data register.

**I2C\_SRXR**

Address: Operational Base + offset(0x0C)

This register contains data to be received on the I2C for slave purpose.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	I2C slave receive data register.

**I2C\_SADDR**

Address: Operational Base + offset(0x10)

This register contains address to be matched on the I2C for slave purpose.

bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9:0	RW	0x3FF	Slave address.

**I2C\_IER**

Address: Operational Base + offset(0x14)

This register contains the bits to control the interrupt generation of I2C controller.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7	RW	0x0	Arbitration lose interrupt enable bit. "1" - enable. "0" - disable.
6	RW	0x0	Abnormal stop interrupt enable bit. "1" - enable. "0" - disable.
5	RW	0x0	Broadcast address matches (address zero) interrupt enable bit. "1" - enable. "0" - disable.
4	RW	0x0	Slave address matches interrupt enable bit. "1" - enable. "0" - disable.
3	RW	0x0	Slave ACK period interrupt enable bit (SRX mode). "1" - enable. "0" - disable.
2	RW	0x0	Slave receives ACK interrupt enable bit (STX mode). "1" - enable. "0" - disable.
1	RW	0x0	Master ACK period interrupt enable bit (MRX mode). "1" - enable. "0" - disable.
0	RW	0x0	Master receives ACK interrupt enable bit (MTX mode). "1" - enable. "0" - disable.

**I2C\_ISR**

Address: Operational Base + offset(0x18)

I2C interrupt status register.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7	RW	0x0	Arbitration lose status bit. "1" - Arbitration lose occurs "0" - No Arbitration lose occurs Write this bit "0" to clear. Write "1" will not change this bit.
6	RW	0x0	Abnormal stop status bit. "1" - Abnormal stop occurs "0" - No abnormal stop occurs Write this bit "0" to clear. Write "1" will not change this bit.
5	RW	0x0	Broadcast address (address zero) matches status bit. "1" - Broadcast address matches. "0" - No broadcast address matches.

			Write this bit "0" to clear. Write "1" will not change this bit.
4	RW	0x0	Slave address matches status bit. "1" - Slave address matches. "0" - No slave address matches (When read). Clear slave address matches interrupt (When write). Write this bit "0" to clear. Write "1" will not change this bit.
3	RW	0x0	Slave ACK period interrupt status bit (SRX mode). "1" - interrupt generation "0" - no interrupt generation Write this bit "0" to clear. Write "1" will not change this bit.
2	RW	0x0	Slave receives ACK interrupt status bit (STX mode). "1" - interrupt generation "0" - no interrupt generation Write this bit "0" to clear. Write "1" will not change this bit.
1	RW	0x0	Master ACK period interrupt status bit (MRX mode). "1" - interrupt generation "0" - no interrupt generation Write this bit "0" to clear. Write "1" will not change this bit.
0	RW	0x0	Master receives ACK interrupt status bit (MTX mode). "1" - interrupt generation "0" - no interrupt generation Write this bit "0" to clear. Write "1" will not change this bit.

**I2C\_LCMR**

Address: Operational Base + offset(0x1C)

This register contains the bits to generate start and stop commands of I2C controller.

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	"RESUME" condition generation bit. "1" - enable. "0" - disable. Write "1" to start "RESUME" action. It cannot be cancelled by write "0". This bit is self-cleared after "RESUME" action. Write "0" is undefined.
1	RW	0x0	"STOP" condition generation bit. "1" - enable. "0" - disable. Write "1" to start "STOP" action. It cannot be cancelled by write "0". This bit is self-cleared after "STOP" action. Write "0" is undefined.
0	RW	0x0	"START" condition generation bit. "1" - enable. "0" - disable. Write "1" to start "START" action. It cannot be cancelled by write "0". This bit is self-cleared after "START" action. Write "0" is undefined.

**I2C\_LSR**

Address: Operational Base + offset(0x20)

This register is used to read I2C core status.

bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1	R	0x0	I2C receives ACK status bit (MTX and STX modes). "0" – I2C bus receives ACK "1" – I2C bus receives NAK.
0	R	0x0	I2C core busy status bit. "1" – After START condition detect. "0" – After STOP condition detect.

**I2C\_CONR**

Address: Operational Base + offset(0x24)

This register is used to set the operation modes and ACK enable bit of I2C controller.

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4	RW	0x0	I2C bus acknowledge enable register "0" – enable (ACK). "1" – disable (NAK). When enable, the SDA is free (TX mode), and is LOW (RX mode) in acknowledge time.
3	RW	0x0	Master receive/transmit mode select bit "0": receive. "1": transmit.
2	RW	0x0	Master port enable bit "0": disable. "1": enable.
1	RW	0x0	Slave receive/transmit mode select bit "0": receive. "1": transmit.
0	RW	0x0	Slave port enable bit "0": disable. "1": enable.

**I2C\_OPR**

Address: Operational Base + offset(0x28)

This register is used to set I2C core enable bit, frequency divider factor, internal state machine reset and slave address length modes.

bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	I2C slave address mode bit. "0" – 7 bits address mode. "1" – 10 bits address mode.
7	RW	0x0	I2C state machine (both master/slave) reset bit. "0" – don't reset state machine "1" – reset state machine
6	RW	0x0	I2C core enable bit "0" – disable I2C controller. "1" – enable I2C controller.
5:0	RW	0x0	I2C clock divisor bits (I2CCDVR). The value of I2CCDVR is used to generate the transmit and receive bit rate of the I2C master part. And the bit rate equation will be described more detail in section below.

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 23.4 Functional Description

### 23.4.1 Operation

**I2C bus terminology**

TERM	DESCRIPTION
Master	The device initiates/stops a transfer and



	generates SCL clock signals.
Slave	The device addressed by a master.
Transmitter	The device which sends data to SDA line.
Receiver	The device which receives data from SDA line.
Multi-master	More than one master can attempt to control the bus without corrupting the message.
Arbitration	If multi-master condition occurs, only one master is allowed to own the bus during this procedure.
Synchronization	Procedure to synchronize the clock signals of two or more devices.

The I2C controller supports both the Master and Slave functions. It also supports the 7-bits/10-bits addressing mode and support general call address. The maximum clock frequency and transfer rate can be up to 400Kbit/sec.

The operations of I2C controller is divided to 3 parts and described separately: initialization, master mode programming, and slave mode programming.

More details are listed in the Program Sequence section.

### Initialization

The I2C controller is based on AMBA APB bus architecture and usually is part of a SOC. So before I2C operates, some system setting & configuration must be conformed, which includes:

I2C Register memory mapping

- I2C interrupt connection type: CPU interrupt scheme should be considered. If the I2C interrupt is connected to extra Interrupt Controller module, we need decide the INTC vector.

I2C Clock Rate: The I2C controller uses the APB clock/5 as the system clock so the APB clock will determine the I2C bus clock. The correct register setting is subject to the system requirement.

### Master Mode Programming

**SCL Clock:** When the I2C controller is programmed in Master mode, the SCL frequency is determine by I2C\_OPR register. The SCL frequency is calculated by the following formula

$$\text{SCL Divisor} = (\text{I2CCDVR}[5:3] + 1) \times 2^{(\text{I2CCDVR}[2:0] + 1)}$$

$$\text{SCL} = \text{PCLK} / 5 * \text{SCLK Divisor}$$

The I2CCDVR[2:0] is coarse factor and I2CCDVR[5:3] is fine tune factor for the SCL clock.

**Data Receiver Register Access:** The Master data receive register (I2C\_MRXR) can only be correctly accessed at Master Receiver Mode. When accessing the I2C\_MRXR register, make sure the I2C\_CONR[3:2] is set to receiver mode.

**Start Command and 1'st Byte Address:** The I2C controller combines the start command and 1'st byte address data output together. SW cannot issue start command only. So before issuing start command, SW must prepare the correct address data (include R/W bit) to the I2C\_MTXR register. Since the I2C protocol allows the repeated start command, the resume command needs to be issued with repeated start.

**Interrupt and Resume:** I2C controller interrupt status is generated by HW and cleared by SW. The clear scheme is to write 0 to the correspond bit. Writing 1 to interrupt status bits will not get any affect. The interrupt clear affect only the interrupt status bits. For reasons of flexibility and interrupt latency reduction, the interrupt clear will not resume the I2C function. The I2C function resumes when I2C\_LCMR register resume bit is set to 1. This separates the I2C service routine from the ISR (Interrupt Service Routine). SW must carefully design the I2C service routine because the I2C controller may be locked in some special state. When operating at Transmit mode, SW should prepare the next

transmit data on the I2C\_MTXR register before issue the resume command.

**Read/Write Command:** The I2C Read/Write command depends on the last bit of address. SW should take the responsibility of Read/Write control. The Read/Write control and the Master Receive/Transmit mode setting must be correctly set before resume the I2C function.

**Multi-Master Arbitration:** When I2C controller works on Multi-Master I2C bus, HW will detect the bus busy condition and arbitration loss. When it happens, HW will stop the transaction and notify SW. SW should take the responsibility of re-transmit and time-out handling.

**Master Interrupt Condition:** There are 3 interrupt bits in I2C\_ISR register related to master mode.

**Master ACK (Bit 0):** The bit is asserted when Master receives ACK. In other words, the interrupt happens only at Master Transmit Mode.

**Master ACK Period (Bit 1):** The bit is asserted when Master needs to send out ACK. In other words, the interrupt happens only at Master Receive Mode.

**Arbitration Loss (Bit 7):** The bit is asserted when Master starts a transaction but lose the bus arbitration.

**Stop Command:** Master can issue Stop command when receive Master ACK or Master ACK Period interrupt. Because the Stop command is attached at the end of a transaction, the resume command needs to be issued with stop command. According to the I2C spec, the NAK must be sent out at Receive Mode in Master ACK Period before Stop.

### Slave Mode Programming

**Data Receiver Register Access:** The Slave data receive register (I2C\_SRXR) can only be correctly accessed at Slave Receiver Mode. When accessing the I2C\_SRXR register, make sure the I2C\_CONR[1:0] is set to slave mode.

**7 Bits and 10 Bits Address:** I2C Slave transaction starts when Slave address is matched. The I2C controller supports both the standard 7 bits address mode and 10 bits address mode. The I2C controller filters out the transaction of which address is not matched with the I2C\_SADDR register. However, I2C controller only filters the 1st address mode, SW should take care the 2nd address for 10 bits address mode. The 7 or 10 bits address mode is set with I2C\_OPR[8].

**7 Bits Address Setting:** Slave function begins upon detecting a received address matched with I2C\_SADDR register. The I2C\_SADDR does not include the Read/Write bit. The I2C\_SADDR[9:7] is ignored at 7 bits address mode.

**10 Bits Address Setting:** The I2C\_SADDR register must be correctly initialized before slave function start to work. The I2C\_SADDR does not include the Read/Write bit. The I2C controller detects the received 1st address by the I2C\_SADDR[9:8] combined with the 10 bits mode address prefix(0b11110xx).

**Address Matching:** The 1st transaction received by I2C controller in slave mode is the address. When the address matched with the slave address of the I2C controller, it will notify SW with an interrupt. I2C\_ISR[4] high represents the incoming slave address matched with the specific slave address of the I2C controller. I2C\_ISR[5] high represents the broadcast, the general call (0x00), is detected.

When address matched, SW should read the I2C\_SRXR register to figure out the transaction is a read or write and set the slave mode accordingly before resume ACK. If the next transaction is a read, the read data needs to be prepared to the I2C\_STXR before resume.

**10 Bits Address 2nd Phase:** Because the I2C controller takes care only the 1st address matching at 10 bits address mode, SW should take care the 2nd address

comparison. When the 2nd byte address comparison fails, SW should issue a reset, I2C\_OPR[7], and issue a resume. After reset and resume, HW will ignore the rest coming transactions until next start detected.

**Interrupt and Resume:** the interrupt is generated by I2C controller and cleared by SW. The clear scheme is to write 0 to the corresponding bit. Writing 1 to interrupt status bits will not get any effect. The interrupt clear affects only the interrupt status bits. For reasons of flexibility and interrupt latency reduction, the interrupt clear will not resume the I2C function. The I2C function resumes when I2C\_LCMR register resume bit is set to 1. This separates the I2C service routine from the ISR (Interrupt Service Routine). SW must carefully design the I2C service routine because the I2C controller may be locked in some special state. When operating at Transmit mode, SW should prepare the next transmit data on the I2C\_STXR register before issue the resume command.

**Read/Write Command:** The I2C Read/Write command depends on the last bit of address. SW should take the responsibility of Read/Write control. The Read/Write control and the Master Receive/Transmit mode setting must be correctly set before resume the I2C function.

**Slave Interrupt Condition:** There are 5 interrupt bits in I2C\_ISR register related to slave mode.

**Slave ACK (Bit 2):** The bit is asserted when Slave receives ACK. In other words, this interrupt happens only at Slave Transmit Mode.

**Slave ACK Period (Bit 3):** The bit is asserted when Slave needs to send out ACK. In other words, this interrupt happens only at Slave Receive Mode.

**Slave address match (bit 4):** The bit is asserted when the coming address is matched with the slave address of the I2C controller. Slave ACK interrupt is not asserted when Slave address match interrupt is asserted.

**Broadcast address match (bit 5):** The bit is asserted when the coming address matched with general call (0x00) address. Slave ACK interrupt is not asserted when general call address match interrupt is asserted.

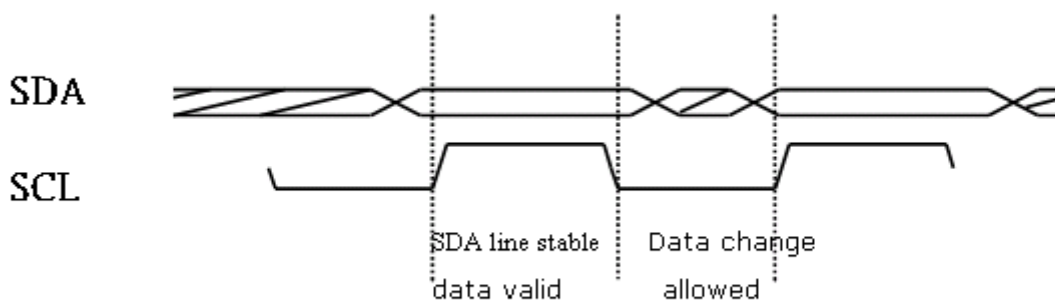
**Abnormal stop occurs (bit 6):** The bit is asserted when Slave receive abnormal stop.

## I2C controller data transfer waveform

- **Bit transferring**

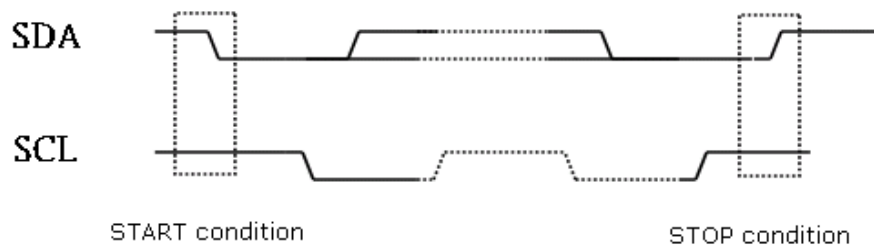
- (a) **Data Validity**

The SDA line must be stable during the high period of SCL, and the data on SDA line can only be changed when SCL is in low state.



- (b) **START and STOP conditions**

START condition occurs when SDA goes low while SCL is in high period. STOP condition is generated when SDA line goes high while SCL is in high state.



### ● Data transfer

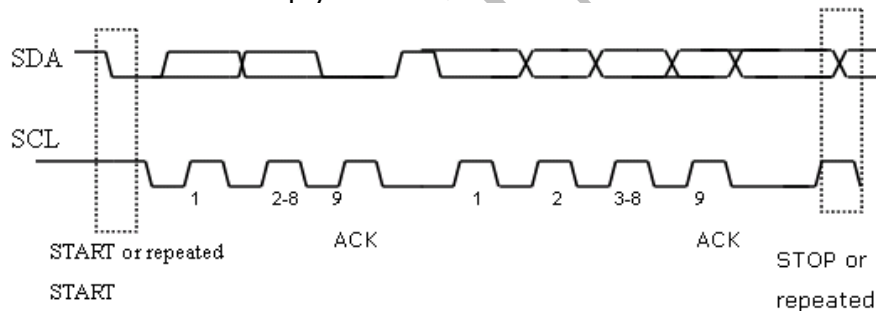
#### (a) Acknowledge

After a byte of data transferring (clocks labeled as 1~8), in 9<sup>th</sup> clock the receiver must assert an ACK signal on SDA line, if the receiver pulls SDA line to low, it means "ACK", on the contrary, it's "NOT ACK".



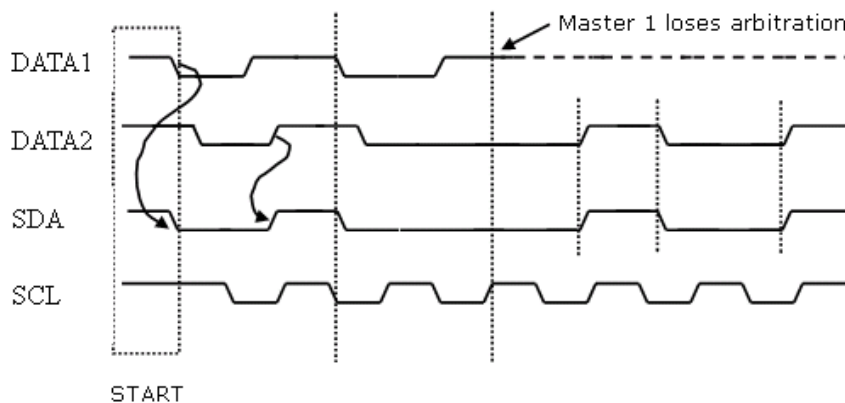
#### (b) Byte transfer

The master own I2C bus might initiate multi byte or transfers to a slave, the transfers starts from a "START" command and ends in a "STOP" command. After every byte transfer, the receiver must reply an ACK to transmitter.



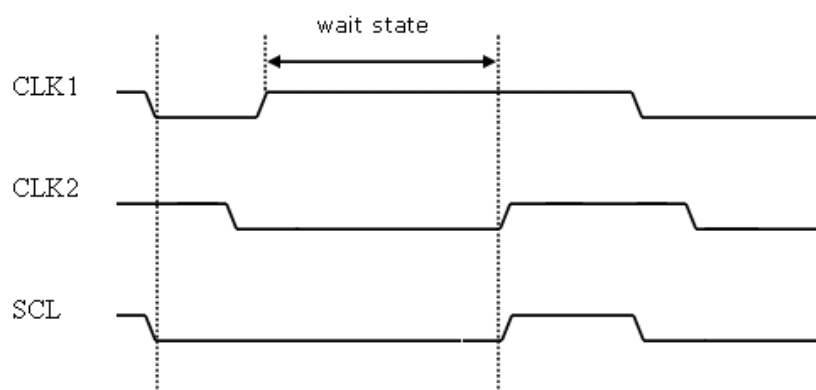
#### (c) Arbitration

Arbitration takes place at SDA line, while the SCL line is at high level. The master transmits a high level, while another master transmits a low level will lose arbitration.



#### (d) Synchronization

Clock synchronization is performed using the wired-and connexion of I2C interface to the SCL line.



### 23.4.2 Programming sequence

Control/Status Register programming sequence

The I2C controller core operation flow chart below is to describe how the software configures and performs an I2C transaction through this I2C controller core. Descriptions are divided into 4 sections, master transmit mode, master receive mode, slave transmit mode and slave receive mode. Users are strongly advised to following.

#### Operations for Master/Transmitter Mode

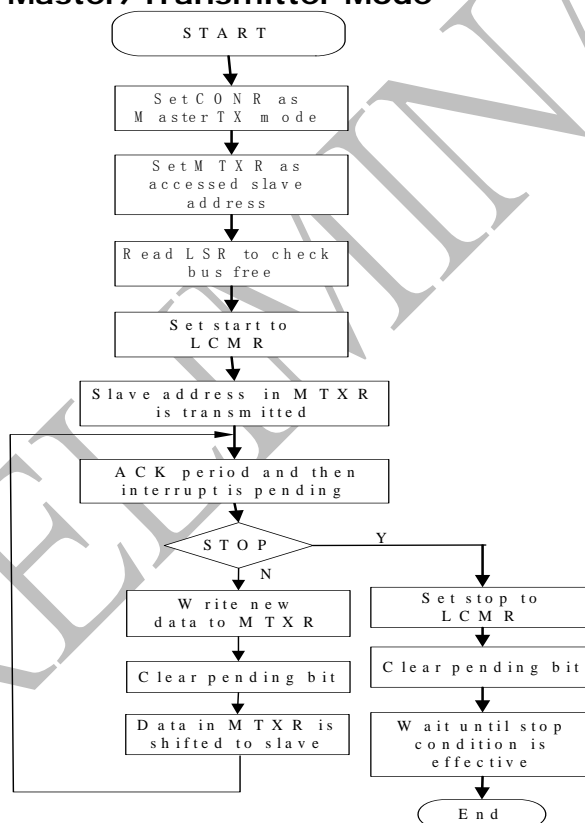


Fig. 23-2 I2C controller operation flow in Master/Transmitter mode

## Operations for Master/Receiver Mode

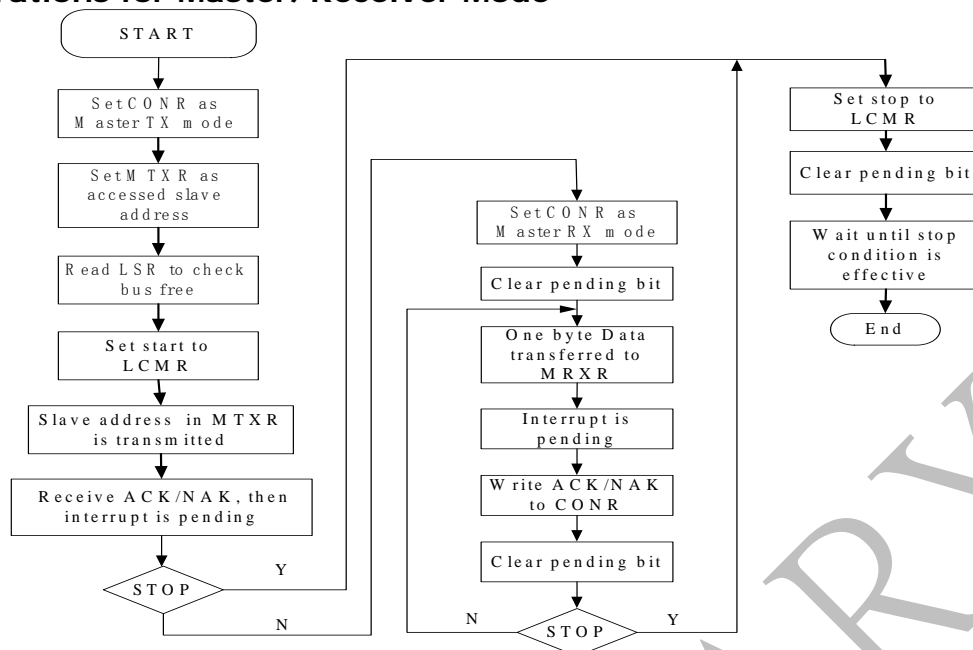


Fig. 23-3 I2C controller operation flow in Master/Receiver mode

## Operations for Slave/Transmitter Mode

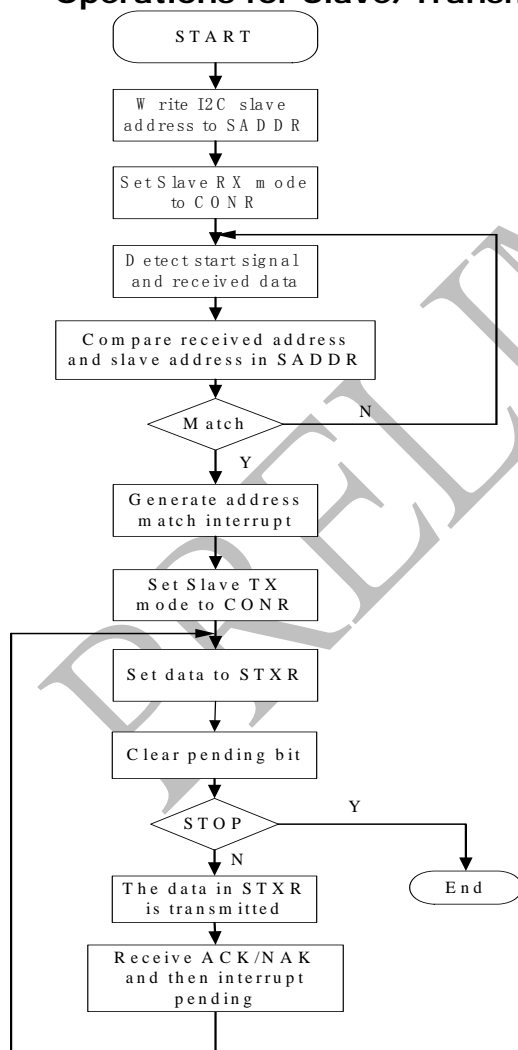


Fig. 23-4 I2C controller operation flow in Slave/Transmitter mode

## Operations for Slave/Receiver Mode

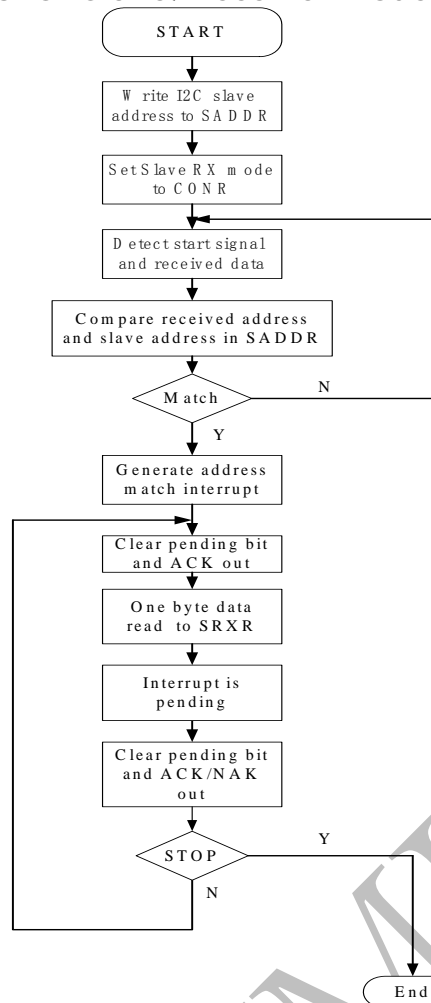


Fig. 23-5 I2C controller operation flow in Slave/Receiver mode



## Chapter 24 I2S Controller

### 24.1 Design Overview

#### 24.1.1 Overview

The I2S Controller is designed for interfacing between the APB bus and the I2S bus.

The I2S bus (Inter-IC sound bus) is a serial link for digital audio data transfer between devices in the system and be invented by Philips Semiconductor. Now it is widely used by many semiconductor manufacturers.

Devices often use the I2S bus are ADC, DAC, DSP, CPU, etc. With the I2S interface, we can connect audio devices and the embedded SoC platform together and provide an audio interface solution for the system.

#### 24.1.2 Features

- Have both transmitter and receiver
- Support mono/stereo audio file
- Support audio resolution: 8, 16 bits
- Support audio sample rate from 32 to 96 KHz
- Support I2S, Left-Justified, Right-Justified digital serial audio data interface
- Have 2 FIFOs with hardware configurable size for Tx and Rx transfer
- Support Master and Slave mode function For Tx and Rx Transfer.

### 24.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

#### 24.2.1 Block Diagram

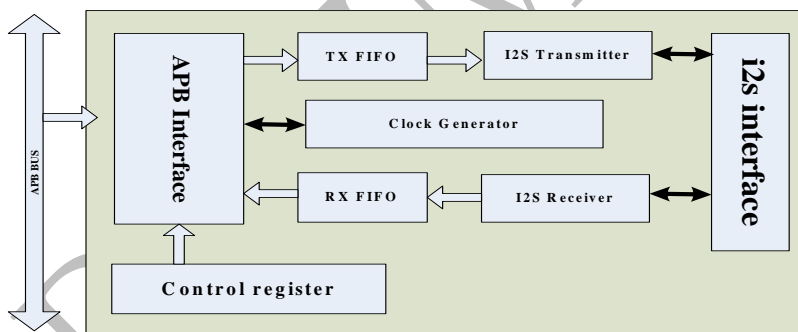


Fig. 24-1 RK281x I2S controller design architecture

#### 24.2.2 Block Descriptions

##### APB Interface/Control Register

The APB Interface implements the APB slave operation. It contains control registers of APB slave, transmitter and receiver inside. Through the APB slave, system can control this I2S design.

##### Clock Generator

The Clock Generator implements clock generation function. The input source clock to the module is MCLK\_I2S, and by the divider of the module, the clock generator generates SCLK and LRCK to transmitter and receiver.

##### I2S Transmitter

The I2S Transmitter implements transmission operation. The transmitter can act as either master or slave, with I2S, Left-Justified, and right-Justified serial audio interface.

The digital data input is through TX FIFO, and output serial data to I2S Interface.

### I2S Receiver

The I2S Receiver implements Receive operation. The receiver can act as either master or slave, with I2S, Left-Justified, and right-Justified serial audio interface. The serial data input is from I2S interface, and input digital data to RX FIFO.

### TX FIFO/RX FIFO

The TX FIFO/RX FIFO is the buffer to store audio data. Both FIFOs have their FIFO control circuit. The size of each FIFO can be programmable, which the default size is 32 bits x 32.

### I2S Interface

The I2S Interface is used to connect I2S bus and both transmitter and receiver of the design. For transmitter, it has four stereo output channels to connect devices, like audio DAC. It is only one of four stereo channels active at one time. For receiver, it has one stereo input channel. The I2S Interface also implements loop-back mode. At loop-back mode, the TX channel 0 is connected directly to RX channel.

## 24.3 Registers

This chapter describes the control/status registers of the design.

### 24.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
I2S_OPR	0x0000	W	0x10000018	I2S version control and operation start register.
I2S_TXR	0x0004	W	0x00000000	I2S transmitter FIFO input.
I2S_RXR	0x0008	W	0x00000000	I2S receiver FIFO output.
I2S_TXCTL	0x000C	W	0x00010811	I2S transmitter control register.
I2S_RXCTL	0x0010	W	0x00010811	I2S receiver control register.
I2S_FIFOSTS	0x0014	W	0x00010055	I2S transmit and receive FIFO control register.
I2S_IER	0x0018	W	0x00000000	I2S interrupt Enable/Mask register.
I2S_ISR	0x001C	W	0x00000000	I2S interrupt status register.

#### 1. Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 24.3.2 Detail Register Description

#### I2S\_OPR

Address: Operational Base + offset(0x00)

This register contains I2S Controller's version and transmit/receive operation bit.

bit	Attr	Reset Value	Description
31:24	R	0x1	I2S version
23:18	-	-	Reserved
17	W	0x0	Reset Tx logic. Writing to this bit will reset Tx logic and its FSM.
16	W	0x0	Reset Rx logic. Writing to this bit will reset Rx logic and its FSM.
15:7	-	-	Reserved
6	W	0x0	HDMA REQ1 Disable 0 : Enable HDMA REQ1 1 : Disable (HDMA REQ1 Always 1)

5	W	0x0	HDMA REQ2 Disable 0 : Enable HDMA REQ2 1 : Disable (HDMA REQ2 Always 1)
4	RW	0x1	HDMA_REQ1_CH This bit is to indicate the Hardware DMA IF1 is used for which FIFO 0 : TX 1 : RX
3	RW	0x1	HDMA_REQ2_CH This bit is to indicate the Hardware DMA IF2 is used for which FIFO 0 : TX 1 : RX
2	RW	0x0	I2S loop-back mode. This bit is to indicate the operation mode of the I2S Controller is in a normal operation mode or in a loop-back mode. 0 : Normal operation mode. 1 : Loop-back mode.
1	RW	0x0	I2S transmit-operation start. The transmitter starts to send SCLK and LRCK signals and transmit data stored in the Tx FIFO to receiver after this bit is set to 1. (Transmitter acts as a master)
0	RW	0x0	I2S receive-operation start. The receiver starts to send SCLK and LRCK signals and receive data from transmitter after this bit is set to 1. (Receiver acts as a master)

**I2S\_TXR**

Address: Operational Base + offset(0x04)

I2S transmit FIFO input.

bit	Attr	Reset Value	Description
31:0	W	0x0	Written data in this register will be transmitted on the I2S bus through the Transmit FIFO.

**I2S\_RXR**

Address: Operational Base + offset(0x08)

I2S receive FIFO output.

bit	Attr	Reset Value	Description
31:0	R	0x0	Received data from I2S bus through the Receive FIFO will be read in this register.

**I2S\_TXCTL**

Address: Operational Base + offset(0x0C)

This register controls the setting of the transmitter.

bit	Attr	Reset Value	Description
31:20	-	-	Reserved
19:18	RW	0x0	Transmitter devices select. This value is used to select which device of the transmitter is active now (including loop-back mode). 0x0 : device 0                      0x1 : device 1 0x2 : device 2                      0x3 : device 3
17:16	RW	0x1	Oversampling rate select bits. 0x0 : 32fs                          0x1 : 64fs 0x2 : 128fs                        0x3 : reserved

			Oversampling rate = LRCK / SCLK
15:8	RW	0x8	Ratio bits. (MCLK / Ratio) = oversampling rate = SCLK frequency. This value is from 1 ~ 255. Default value is 8.
7:6	-	-	Reserved
5:4	RW	0x1	Sample data resolution. Number of bits that are transmitted from each audio word. 0x0 : 8 bits                      0x2 : 16 bits 0x2~0x3: Reserved
3	RW	0x0	Mono/Stereo mode. When the bit is set to 1, transmitter is at Mono mode, and data output from left channel. Default is stereo mode.
2:1	RW	0x0	Bus Interface mode Choose the type of the bus interface. 0x0 : I2S                      0x1 : Left - Justified 0x2 : Right - Justified    0x3 : reserved
0	RW	0x1	Master/Slave mode select. This bit decides that transmitter acts as a master or slave. 1 : Master mode 0 : Slave mode

**I2S\_RXCTL**

Address: Operational Base + offset(0x10)

This register controls the setting of the receiver.

bit	Attr	Reset Value	Description
31:25	-	-	Reserved
24	W	0x0	Clear Rx FIFO bit. Writing a '1' to this bit clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, i.e. receiving of the current character continues.
23:18	-	-	Reserved
17:16	RW	0x1	Oversampling rate select bits. 0x0 : 32fs                      0x1 : 64fs 0x2 : 128fs                    0x3 : reserved Oversampling rate = LRCK / SCLK
15:8	RW	0x8	Ratio bits. (MCLK / Ratio) = oversampling rate = SCLK frequency. This value is from 1 ~ 255. Default value is 8.
7:6	-	-	Reserved
5:4	RW	0x1	Sample data resolution. Number of bits that are transmitted from each audio word. (20 and 24 bits is not available now) 0x0 : 8 bits                      0x1 : 16 bits 0x2 : 20 bits                    0x3 : 24 bits
3	RW	0x0	Mono/Stereo mode. When the bit is set to 1, transmitter is at Mono mode, and data output from left channel. Default is stereo mode.
2:1	RW	0x0	Bus Interface mode Choose the type of the bus interface. 0x0 : I2S                      0x1 : Left - Justified 0x2 : Right - Justified    0x3 : reserved

0	RW	0x1	Master/Slave mode select. This bit decides that transmitter acts as a master or slave. 1 : Master mode 0 : Slave mode
---	----	-----	--------------------------------------------------------------------------------------------------------------------------------

**I2S\_FIFOSTS**

Address: Operational Base + offset(0x14)

This register shows FIFO status and interrupts trigger level.

bit	Attr	Reset Value	Description
31:20	-	-	Reserved
19:18	RW	0x0	Tx interrupt trigger level. 0x0 : almost empty      0x1 : half full 0x2 : almost full      0x3 : reserved
17:16	RW	0x1	Rx interrupt trigger level. 0x0 : almost empty      0x1 : half full 0x2 : almost full      0x3 : reserved
15:10	-	-	Reserved
9	R	0x0	Tx FIFO half full flag. This bit is set whenever Tx FIFO is half full.
8	R	0x0	Rx FIFO half full flag. This bit is set whenever Rx FIFO is half full.
7	R	0x0	Tx FIFO almost full flag. This bit is set whenever Tx FIFO is almost full.
6	R	0x1	Tx FIFO almost empty flag. This bit is set whenever Tx FIFO is almost empty.
5	R	0x0	Rx FIFO almost full flag. This bit is set whenever Rx FIFO is almost full.
4	R	0x1	Rx FIFO almost empty flag. This bit is set whenever Rx FIFO is almost empty.
3	R	0x0	Tx FIFO full flag. This bit is set whenever Tx FIFO is full.
2	R	0x1	Tx FIFO empty flag. This bit is set whenever Tx FIFO is empty.
1	R	0x0	Rx FIFO full flag. This bit is set whenever Rx FIFO is full.
0	R	0x1	Rx FIFO empty flag. This bit is set whenever Rx FIFO is empty.

**I2S\_IER**

Address: Operational Base + offset(0x18)

This register contains the bits to control the interrupt generation of this I2S Controller.

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Tx FIFO data trigger interrupt enable bit. This bit enables the interrupt when Tx FIFO's trigger level is reached. 0x0 : Disable. 0x1 : Enable.
1	RW	0x0	Rx FIFO data trigger interrupt enable bit. This bit enables the interrupt when Rx FIFO's trigger level is reached. 0x0 : Disable. 0x1 : Enable.
0	RW	0x0	Rx FIFO overrun interrupt enable bit. This bit enables the interrupt when Rx FIFO <b>overrun</b>

			<b>condition</b> is occurred. 0x0 : Disable. 0x1 : Enable.
--	--	--	------------------------------------------------------------------

**I2S\_ISR**

Address: Operational Base + offset(0x1C)

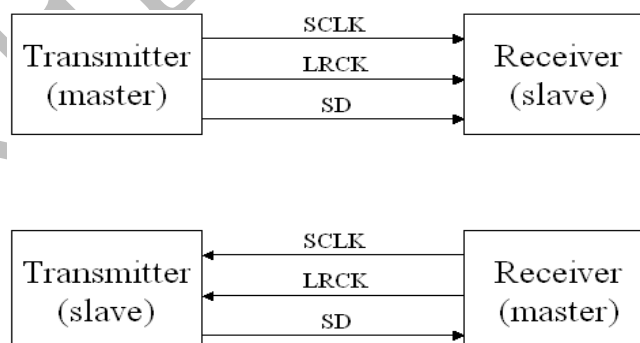
I2S interrupt status register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	Tx FIFO almost empty interrupt. This bit is set when Tx FIFO's trigger level is reached, and CPU wishes to keep transmitting data to the device. The bit is cleared when data in Tx FIFO is above trigger level.
1	R	0x0	Rx FIFO data trigger interrupt. This bit is set when Rx FIFO's trigger level is reached. The bit is cleared when data in Rx FIFO is below trigger level.
0	R	0x0	Rx FIFO overrun interrupt. This bit is set when <b>Rx FIFO is full and another character has been received in the receiver shift register</b> . If another character is starting to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. The bit is cleared after Rx FIFO is cleared by software simultaneously.

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only**24.4 Functional Description****24.4.1 Operation**

Digital audio serial data interface format

The I2S interface core supports three digital serial data interface formats for audio data transfer: I2S, Left-Justified, Right-justified. All these formats have SCLK, LRCK, and SD signals. The signal's direction is as below:



I2S Interface format

This is the waveform of I2S interface. For LRCK signal, it goes "low" to indicate left channel and "high" to right channel. For SD signal, it transfers MSB-first and sends the MSB bit one SCLK clock cycle after LRCK goes low.

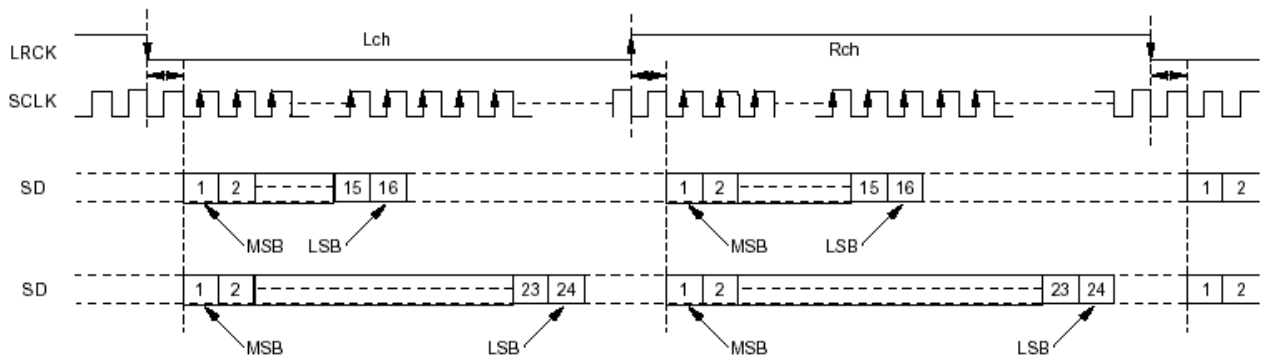


Fig. 24-2 I2S Controller timing format for I2S interface

#### Left-Justified Interface format

This is the waveform of Left-Justified interface. For LRCK signal, it goes "high" to indicate left channel and "low" to right channel. For SD signal, it transfers MSB-first and sends the MSB bit at the same time when LRCK goes high.

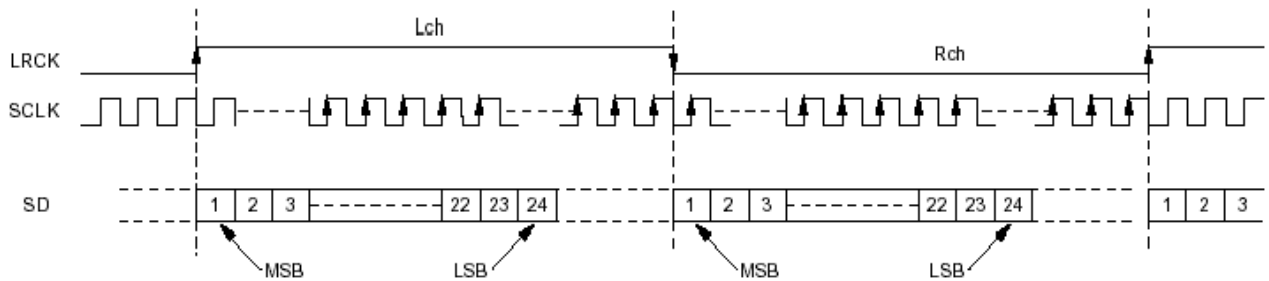


Fig. 24-3 I2S Controller timing format for Left-Justified interface

#### Right-justified Interface format

This is the waveform of Right-Justified interface. For LRCK signal, it goes "high" to indicate left channel and "low" to right channel. For SD signal, it transfers MSB-first; but different from I2S or Left-Justified interface, its data is aligned to LSB at falling edge of the LRCK signal.

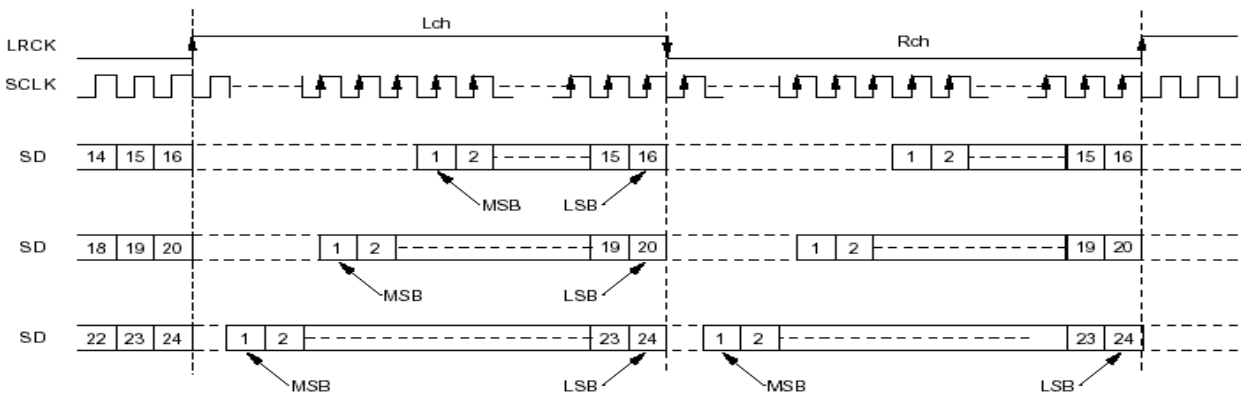


Fig. 24-4 I2S Controller timing format for Right-Justified interface

#### I2S Normal Operation

Referring to Figure 6-1, in the I2S Controller, there are four conditions: transmitter-master & receiver-master; transmitter-master & receiver-slave; transmitter-slave & receiver-master; transmitter-slave & receiver-slave.

When transmitter acts as a master, it sends all signals to receiver (slave), and CPU control when to send clock and data to the receiver. When acting as a slave, SD signal still goes from transmitter to receiver, but SCLK and LRCK signals are from receiver (master) to transmitter. Based on three interface specifications, transmitting data should be ready before transmitter receives SCLK and LRCK signals. CPU should know when the receiver to



initialize a transaction and when to send data.

When the receiver acts as a master, it sends SCLK and LRCK signals to the transmitter (slave) and receives serial data. So CPU must tell the transmitter when to start a transaction for it to prepare transmitting data then start a transfer and send clock and channel-select signals. When the receiver acts as a slave, CPU should only do initial setting and wait for all signals and then start reading data.

#### I2S initial setting

Before transmitting or receiving data, CPU need do initial setting to the I2S register. These includes CPU settings, I2S interface registers settings, and maybe the embedded SoC platform settings. These registers must be set before starting data transfer. Detail settings about the registers of the I2S interface refer to chapter 5 "Function Registers".

#### I2S Loop-back Test

The I2S interface has two operation modes: Normal mode and Loop-back mode. In the Loop-back mode operation, transmitter acts as a master and receiver acts as a slave. Transmitter device 0's output is connected to receiver's input. That is, TX\_SCLKOUT[0], TX\_LRCKOUT[0] and SDO[0] is connected to RX\_SCLKIN, RX\_LRCKIN and SDI.

### 24.4.2 Programming sequence

#### I2S Tx Operation Flow Chart

The flow chart below is to describe how the software to configure and perform an I2S transmitting transaction from transmitter's view.

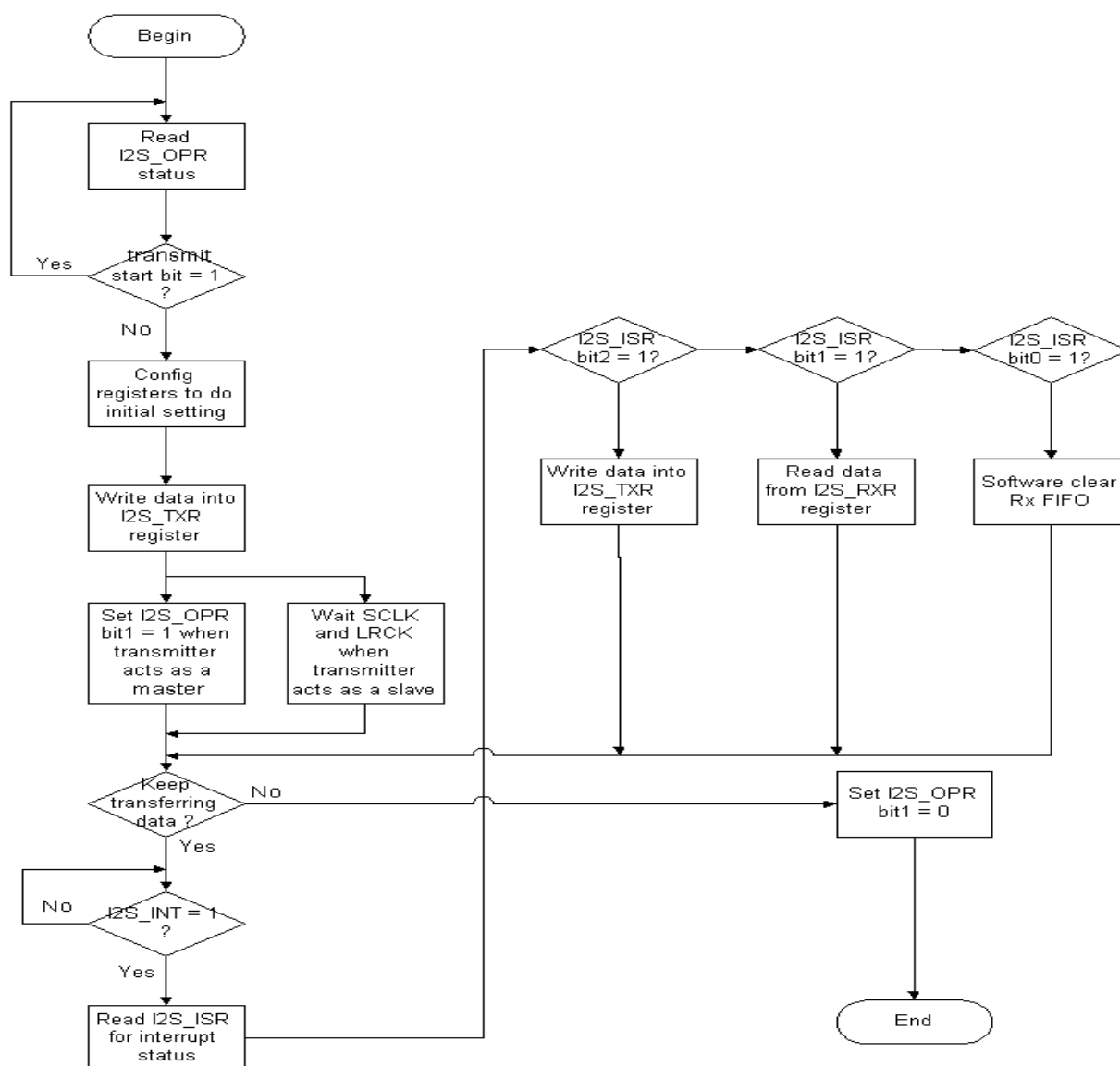


Fig. 24-5 I2S Controller TX operation flow chart

### I2S Rx Operation Flow Chart

The flow chart below is to describe how the software to configure and perform an I2S receiving transaction from receiver's view.

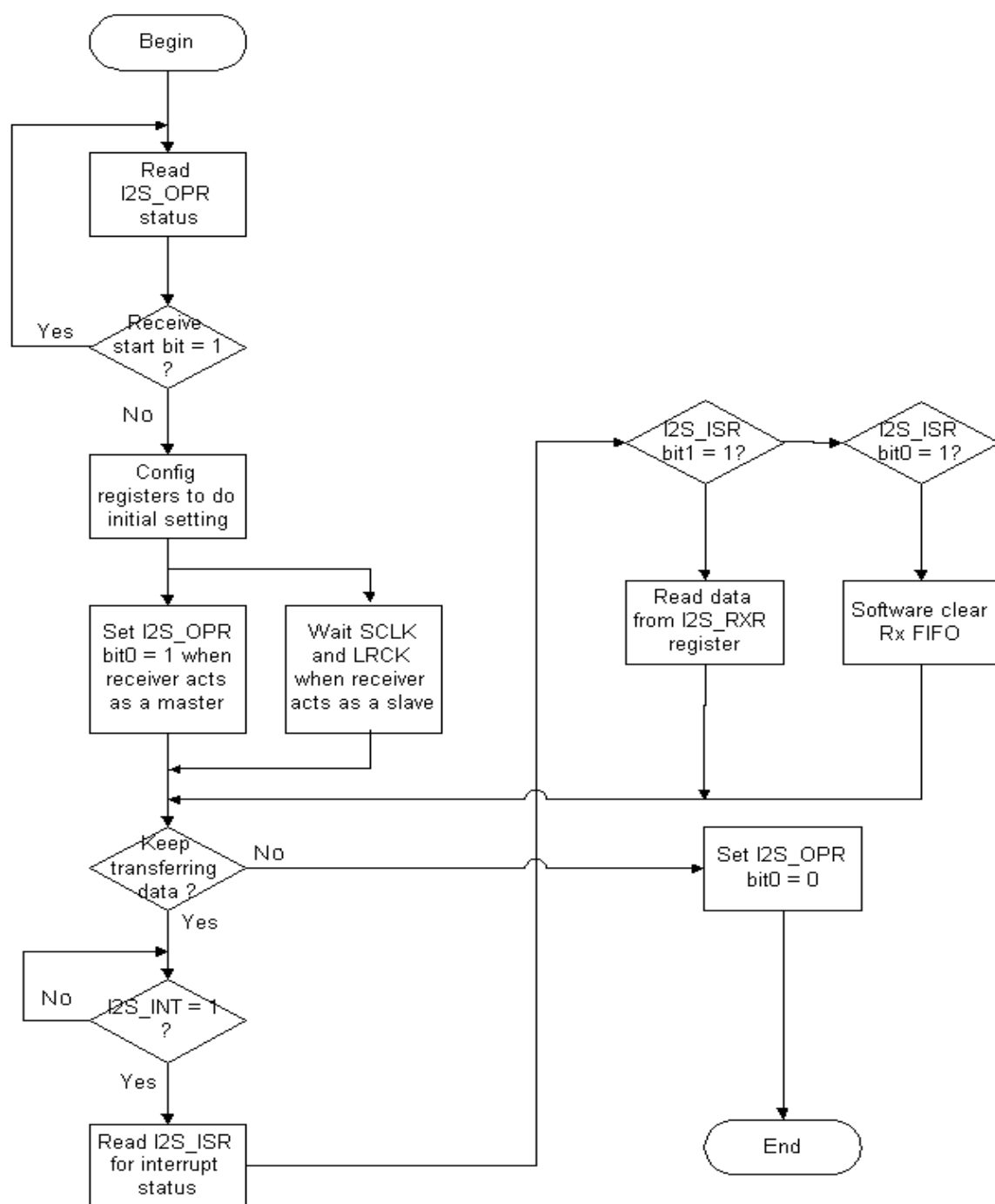


Fig. 24-6 I2S Controller RX operation flow chart

## Chapter 25 PWM Timer

### 25.1 Overview

There are four PWM blocks in PWM Timer (PWM0, PWM1, PWM2 and PWM3). Each PWM block built-in 4-bit pre-scalar from PCLK. The PWM Timer supports both reference mode, which can output various duty-cycle waveforms, and capture, which can measure the duty-cycle of input waveform.

#### 25.1.1 Key Features

- Programmable 4-bit pre-scalar
- 32-bit timer/counter facility
- Single-run or continues-run PWM mode
- Support maskable interrupt

### 25.2 Architecture

#### 25.2.1 Block Diagram

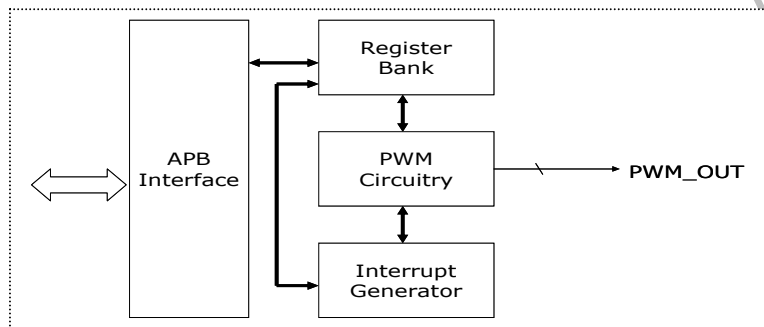


Fig. 25-1 PWM design architecture

#### 25.2.2 Block Descriptions

##### PWM Register Block

This block controls the setting of PWM mode.

##### PWM Circuitry

This block includes clock pre-scalar and reference comparator for PWM timer.

##### Interrupt Generator

This block handles the interrupt generation, masking, and clearing.

### 25.3 Registers

#### 25.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
PWMT0_CNTR	0x0000	W	0x00000000	Main counter register
PWMT0_HRC	0x0004	W	0x00000000	PWM HIGH Reference/Capture register
PWMT0_LRC	0x0008	W	0x00000000	PWM LOW Reference/Capture register
PWMT0_CTRL	0x000C	W	0x00000000	Current value register
PWMT1_CNTR	0x0010	W	0x00000000	Main counter register

PWMT1_HRC	0x0014	W	0x00000000	PWM HIGH Reference/Capture register
PWMT1_LRC	0x0018	W	0x00000000	PWM LOW Reference/Capture register
PWMT1_CTRL	0x001C	W	0x00000000	Current value register
PWMT2_CNTR	0x0020	W	0x00000000	Main counter register
PWMT2_HRC	0x0024	W	0x00000000	PWM HIGH Reference/Capture register
PWMT2_LRC	0x0028	W	0x00000000	PWM LOW Reference/Capture register
PWMT2_CTRL	0x002C	W	0x00000000	Current value register
PWMT3_CNTR	0x0030	W	0x00000000	Main counter register
PWMT3_HRC	0x0034	W	0x00000000	PWM HIGH Reference/Capture register
PWMT3_LRC	0x0038	W	0x00000000	PWM LOW Reference/Capture register
PWMT3_CTRL	0x003C	W	0x00000000	Current value register

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

### 25.3.2 Detail Register Description

#### PWMTn\_CNTR (n=0~3)

Address: Operational Base + offset(0x00, 0x10, 0x20, 0x30)

PWM0 timer counter.

bit	Attr	Reset Value	Description
31:0	RW	0	Main PWM timer counter. Counting value ranges from 0 ~ ( $2^{32} - 1$ ).

#### PWMTn\_HRC (n=0~3)

Address: Operational Base + offset(0x04, 0x14, 0x24, 0x34)

PWM0 HIGH reference or capture register

bit	Attr	Reset Value	Description
31:0	RW	0	PWM HIGH reference/capture registers

#### PWMTn\_LRC (n=0~3)

Address: Operational Base + offset(0x08, 0x18, 0x28, 0x38)

PWM0 LOW reference or capture register

bit	Attr	Reset Value	Description
31:0	RW	0	PWM LOW reference/capture registers

#### PWMTn\_CTRL (n=0~3)

Address: Operational Base + offset(0x0C, 0x1C, 0x2C, 0x3C)

This control register of PWM0 Timer.

bit	Attr	Reset Value	Description
31:13	-	-	Reserved.
12:9	R/W	0	Prescale factor. <div style="display: flex; justify-content: space-between;"> <div> 0000: 1/2  0010: 1/8  0100: 1/32  0110: 1/128  1000: 1/512  1010: 1/2048  1100: 1/8192  1110: 1/32768 </div> <div> 0001: 1/4  0011: 1/16  0101: 1/64  0111: 1/256  1001: 1/1024  1011: 1/4096  1101: 1/16384  1111: 1/65536 </div> </div>
8	R/W	0	Capture mode enable/disable

			0: Disable 1: Enable
7	R/W	0	PWM reset. 0: Normal operation 1: Reset PWM
6	R/W	0	Interrupt status and clear bit. Write "1" to clear interrupt status.
5	R/W	0	PWM timer interrupt enable/disable. PWM timer will assert an interrupt when PWMTx_CNTR value is equal to the value of PWMTx_LRC or PWMTx_HRC. 0: Disable 1: Enable
4	R/W	0	Single counter mode. 0: PWMTx_CNTR is restarted after it reaches value equal to the PWMTx_LRC value. 1: PWMTx_CNTR is not increased anymore after it reaches value equal to the PWMTx_LRC value.
3	R/W	0	PWM output enable/disable. 0: Disable 1: Enable
2:1	-	-	Reserved
0	R/W	0	PWM timer enable/disable. 0: Disable 1: Enable

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## Chapter 26 SAR-ADC Controller

### 26.1 Overview

The ADC is an 4-channel signal-ended 10-bit Successive Approximation Register (SAR) A/D Converter. It converts the analog input signal into 10-bit binary digital codes at maximum conversion rate of 100KSPS with 1MHz A/D converter clock.

#### 26.1.1 Key Features

- 4 input channels
- Maximum 10-bit resolution
- Maximum conversion rate of 100KSPS
- Channel 3 is tied to 1.2V

### 26.2 Architecture

#### 26.2.1 Block Diagram

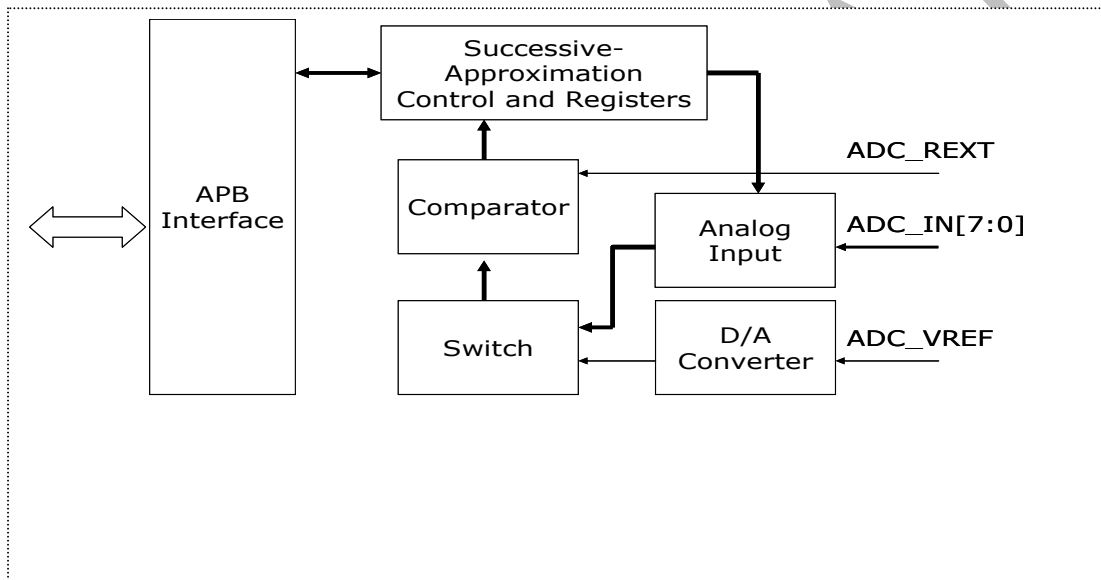


Fig. 26-1 SAR-ADC Controller design architecture

#### 26.2.2 Block Descriptions

##### Successive-Approximate Register and Control Logic Block

This block is exploited to realize binary search algorithm, storing the intermediate result and generate control signal for analog block.

##### Comparator Block

This block compares the analog input ADC\_CH[3:0] with the voltage generated from D/A Converter, and output the comparison result to SAR and Control Logic Block for binary search. Three level amplifiers are employed in this comparator to provide enough gain.

### 26.3 Registers

#### 26.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
ADC_DATA	0x0000	W	0x00000000	ADC data registers
ADC_STAS	0x0004	W	0x00000000	ADC status register
ADC_CTRL	0x0008	W	0x00000000	ADC controller register



Notes: **Size**: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

### 26.3.2 Detail Register Description

#### ADC\_DATA

Address: Operational Base + offset(0x00)

This register contains the data after A/D Conversion.

bit	Attr	Reset Value	Description
31:10	-	-	Reserved.
9:0	R	0x00000000	A/D value of the last conversion.

#### ADC\_STAS

Address: Operational Base + offset(0x04)

The status register of A/D Converter.

bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0	ADC status. 0: ADC stop      1: Conversion in progress

#### ADC\_CTRL

Address: Operational Base + offset(0x08)

The control register of A/D Converter.

bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0	Interrupt status. This bit will be set to 1 when end-of-conversion. Set 0 to clear the interrupt.
5	RW	0	Interrupt enable. 0: Disable      1: Enable
4	RW	0	Start of Conversion(SOC) Set this bit to 1 to start an ADC conversion. This bit will reset to 0 by hardware when ADC conversion has started.
3	RW	0	ADC power down control bit 0: ADC power down 1: ADC power up and reset
2:0	RW	0	ADC input source selection. 000 : Input source 0 (ADC_CH[0]) 001 : Input source 1 (ADC_CH[1]) 010 : Input source 2 (ADC_CH[2]) 011 : Input source 3 (ADC_CH[3])

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 26.4 Function Description

#### A/D Conversion Sequence

The following is an example sequence of setting up A/D Converter, starting of conversion, and acquiring the result value.

- Power-up A/D Converter in ADC\_CTRL[3]
- Select input channel of A/D Converter in ADC\_CTRL[2:0] bit
- Set ADC start conversion in ADC\_CTRL[4]
- Wait an A/D interrupt or poll the ADC\_STAS register to determine when the conversion is completed
- Read the conversion result in the ADC\_DATA register

## Chapter 27 GPIO in CPU System

### 27.1 Design Overview

#### 27.1.1 Overview

GPIO is a programmable General Purpose Programming I/O peripheral. This component is a APB slave device. GPIO controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers.

There are two GPIOs in CPU system: GPIO\_0 and GPIO\_1. Each GPIO has 32 I/O pins, which is divided into 4 group from A to D. Pay more attention that only 8 ports in group A can support interrupt function, and connected to interrupt controller.

#### 27.1.2 Features

- 32 bits APB bus width
- 32 independently configurable signals
- Four ports, A to D, which are separately configurable
- Configurable interrupt mode for Port A
- Separate data registers and data direction registers for each signal
- Software control for each bit of each signal

### 27.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

#### 27.2.1 Block Diagram

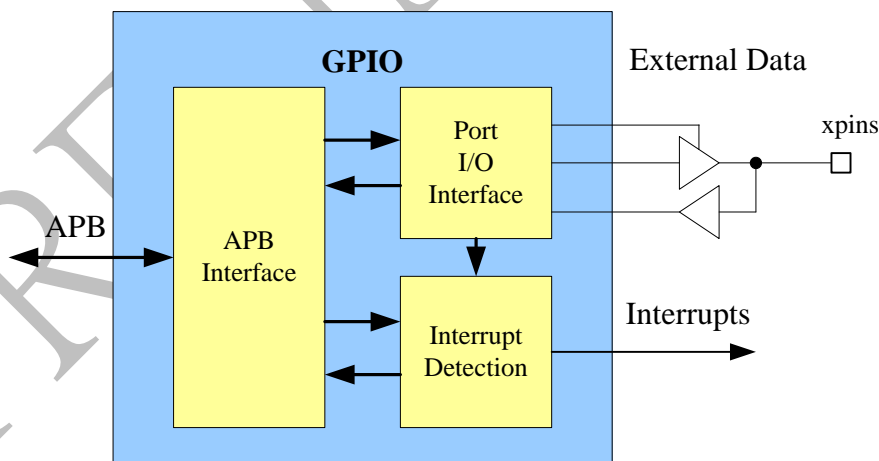


Fig. 27-1 GPIO in CPU System Block Diagram

#### 27.2.2 Block Descriptions

##### APB Interface

The APB Interface implements the APB slave operation. It's bus width is 32 bits.

##### Port I/O Interface

External data Interface to or from I/O pads.

##### Interrupt Detection

Interrupt interface to or from interrupt controller.

## 27.3 Registers

This chapter describes the control/status registers of the design. Software should read and write these registers using 32-bits accesses.

### 27.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
GPIO_SWPORTA_DR	0x0000	W	0x00000000	Port A data register
GPIO_SWPORTA_DDR	0x0004	W	0x00000000	Port A data direction register
GPIO_SWPORTB_DR	0x000C	W	0x00000000	Port B data register
GPIO_SWPORTB_DDR	0x0010	W	0x00000000	Port B data direction register
GPIO_SWPORTC_DR	0x0018	W	0x00000000	Port C data register
GPIO_SWPORTC_DDR	0x001C	W	0x00000000	Port C data direction register
GPIO_SWPORTD_DR	0x0024	W	0x00000000	Port D data register
GPIO_SWPORTD_DDR	0x0028	W	0x00000000	Port D data direction register
GPIO_INTEN	0x0030	W	0x00000000	Port A Interrupt enable register
GPIO_INTMASK	0x0034	W	0x00000000	Port A Interrupt mask register
GPIO_INTTYPE_LEVEL	0x0038	W	0x00000000	Port A Interrupt level register
GPIO_INT_POLARITY	0x003C	W	0x00000000	Port A Interrupt polarity register
GPIO_INT_STATUS	0x0040	W	0x00000000	Interrupt status of port A
GPIO_INT_RAWSTATUS	0x0044	W	0x00000000	Raw Interrupt status of port A
GPIO_DEBOUNCE	0x0048	W	0x00000000	Debounce enable register
GPIO_PORTA_EOI	0x004C	W	0x00000000	Port A clear interrupt register
GPIO_EXT_PORTA	0x0050	W	0x00000000	Port A external port register
GPIO_EXT_PORTB	0x0054	W	0x00000000	Port B external port register
GPIO_EXT_PORTC	0x0058	W	0x00000000	Port C external port register
GPIO_EXT_PORTD	0x005C	W	0x00000000	Port D external port register
GPIO_LS_SYNC	0x0060	W	0x00000000	Level sensitive synchronization enable register

Notes: Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

### 27.3.2 Detail Register Description

#### GPIO\_SWPORTA\_DR

Address: Operational Base + offset(0x00)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port A if the corresponding data direction bits for Port A are set to Output mode. The value read back is equal to the last value written to this register.

#### GPIO\_SWPORTA\_DDR

Address: Operational Base + offset(0x04)

Port A data register

bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port A. 0: Input (default) 1: Output

**GPIO\_SWPORTB\_DR**

Address: Operational Base + offset(0x0C)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port B if the corresponding data direction bits for Port B are set to Output mode. The value read back is equal to the last value written to this register.

**GPIO\_SWPORTB\_DDR**

Address: Operational Base + offset(0x10)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port B. 0: Input (default) 1: Output

**GPIO\_SWPORTC\_DR**

Address: Operational Base + offset(0x18)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port C if the corresponding data direction bits for Port C are set to Output mode. The value read back is equal to the last value written to this register.

**GPIO\_SWPORTC\_DDR**

Address: Operational Base + offset(0x1C)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port C. 0: Input (default) 1: Output

**GPIO\_SWPORTD\_DR**

Address: Operational Base + offset(0x24)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port D if the corresponding data direction bits for Port D are set to Output mode.

			The value read back is equal to the last value written to this register.
--	--	--	--------------------------------------------------------------------------

**GPIO\_SWPORTD\_DDR**

Address: Operational Base + offset(0x28)

Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port D. 0: Input (default) 1: Output

**GPIO\_INTEN**

Address: Operational Base + offset(0x30)

Interrupt enable register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Allows each bit of Port A to be configured for interrupts. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on Port A to become an interrupt; otherwise, Port A operates as a normal GPIO signal. Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output. 0: Configure Port A bit as normal GPIO signal (default) 1: Configure Port A bit as interrupt

**GPIO\_INTMASK**

Address: Operational Base + offset(0x34)

Interrupt mask register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. 0: Interrupt bits are unmasked (default) 1: Mask interrupt

**GPIO\_INTTYPE\_LEVEL**

Address: Operational Base + offset(0x38)

Interrupt level register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls the type of interrupt that can occur on Port A. 0: Level-sensitive (default) 1: Edge-sensitive

**GPIO\_INT\_POLARITY**

Address: Operational Base + offset(0x3C)

Interrupt polarity register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved

7:0	RW	0x00	Controls the polarity of edge or level sensitivity that can occur on input of Port A. 0: Active-low (default) 1: Active-high
-----	----	------	------------------------------------------------------------------------------------------------------------------------------------

**GPIO\_INTSTATUS**

Address: Operational Base + offset(0x40)

Interrupt status register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	Interrupt status of Port A

**GPIO\_RAWINTSTATUS**

Address: Operational Base + offset(0x44)

Raw Interrupt status register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	Raw interrupt of status of Port A (premasking bits)

**GPIO\_DEBOUNCE**

Address: Operational Base + offset(0x48)

Debounce enable register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. Writing a 1 to a bit in this register enables the debouncing circuitry. A signal must be valid for two periods of an external clock before it is internally processed. 0: No debounce (default) 1: Enable debounce

**GPIO\_PORTS\_EOI**

Address: Operational Base + offset(0x4C)

Port A clear interrupt register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	W	0x00	Controls the clearing of edge type interrupts from Port A. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts. 0: No interrupt clear (default) 1: Clear interrupt

**GPIO\_EXT\_PORTA**

Address: Operational Base + offset(0x50)

Port A external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port A is configured as Input, then reading this location reads the values on the signal. When the data direction of Port A is set as Output, reading this location reads the data register for Port A.

**GPIO\_EXT\_PORTB**

Address: Operational Base + offset(0x54)

Port B external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port B is configured as Input, then reading this location reads the values on the signal. When the data direction of Port B is set as Output, reading this location reads the data register for Port B.

#### GPIO\_EXT\_PORTC

Address: Operational Base + offset(0x58)

Port C external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port C is configured as Input, then reading this location reads the values on the signal. When the data direction of Port C is set as Output, reading this location reads the data register for Port C.

#### GPIO\_EXT\_PORTD

Address: Operational Base + offset(0x5C)

Port D external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port D is configured as Input, then reading this location reads the values on the signal. When the data direction of Port D is set as Output, reading this location reads the data register for Port D.

#### GPIO\_LS\_SYNC

Address: Operational Base + offset(0x60)

Level\_sensitive synchronization enable register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr. 0: No synchronization to pclk_intr (default) 1: Synchronize to pclk_intr

## 27.4 Functional Description

### 27.4.1 Operation

#### Control Mode(software)

Under software control, the data and direction control for the signal are sourced from the data register (GPIO\_SWPORTX\_DR) and direction control register (GPIO\_SWPORTX\_DDR), where X is either A, B, C, or D.

The direction of the external I/O pad is controlled by a write to the Portx data direction register (GPIO\_SWPORTX\_DDR). The data written to this memory-mapped register gets mapped onto an output signal, GPIO\_PORTX\_DDR, of the GPIO peripheral. This output signal controls the direction of an external I/O pad.

The data written to the Portx data register (GPIO\_SWPORTX\_DR) drives the output buffer of the I/O pad. External data are input on the external data signal, GPIO\_EXT\_PORTX. Reading the external signal register (GPIO\_EXT\_PORTX) shows the value on the signal, regardless of the direction. This register is read-only.



## Reading External Signals

The data on the GPIO\_EXT\_PORTX external signal can always be read. The data on the external gpio signal is read by an APB read of the memory-mapped register, GPIO\_EXT\_PORTX.

An APB read to the GPIO\_EXT\_PORTX register yields a value equal to that which is on the GPIO\_EXT\_PORTX signal.

## Interrupts

Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge

The interrupts can be masked by programming the GPIO\_INTMASK register. The interrupt status can be read before masking (called raw status) and after masking.

The interrupts are combined into a single interrupt output signal, which has the same polarity as the individual interrupts. In order to mask the combined interrupt, all individual interrupts have to be masked. The single combined interrupt does not have its own mask bit. GPIO\_STATUS register must be read in the interrupt service routine (ISR) to find the source of the interrupt.

Whenever Port A is configured for interrupts, the data direction must be set to Input. If the data direction register is reprogrammed to Output, then any pending interrupts are not lost. However, no new interrupts are generated.

For edge-detected interrupts, the ISR can clear the interrupt by writing a 1 to the GPIO\_PORTA\_EOI register for the corresponding bit to disable the interrupt. This write also clears the interrupt status and raw status registers.

Writing to the GPIO\_PORTA\_EOI register has no effect on level-sensitive interrupts. If level-sensitive interrupts cause the processor to interrupt, then the ISR can poll the GPIO\_RAWINT status register until the interrupt source disappears, or it can write to the GPIO\_INTMASK register to mask the interrupt before exiting the ISR. If the ISR exits without masking or disabling the interrupt prior to exiting, then the level-sensitive interrupt repeatedly requests an interrupt until the interrupt is cleared at the source.

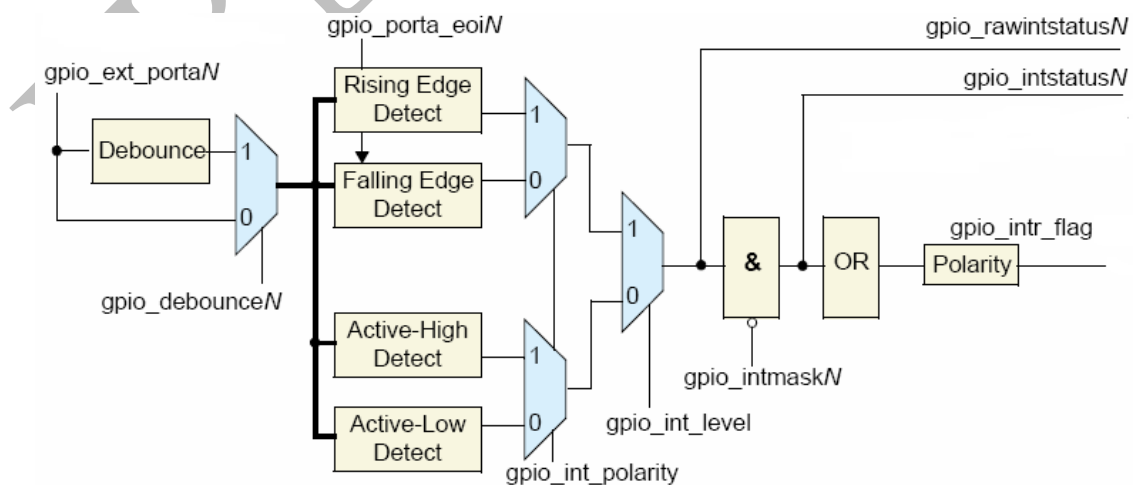


Fig. 27-2 GPIO in CPU System Interrupt RTL Block Diagram

### Debounce operation

Port A has been configured to include the debounce capability interrupt feature. The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock.

When input interrupt signals are debounced using a debounce clock(pclk), the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered.

### Synchronization of Interrupt Signals to the System Clock

Interrupt signals are internally synchronized to pclk. Synchronization to pclk must occur for edge-detect signals. With level-sensitive interrupts, synchronization is optional and under software control(GPIO\_LS\_SYNC).

## 27.4.2 Programming

### Programming Considerations

- Reading from an unused location or unused bits in a particular register always returns zeros. There is no error mechanism in the APB.
- Programming the GPIO registers for interrupt capability, edge-sensitive or level-sensitive interrupts, and interrupt polarity should be completed prior to enabling the interrupts on Port A in order to prevent spurious glitches on the interrupt lines to the interrupt controller.
- Writing to the interrupt clear register clears an edge-detected interrupt and has no effect on a level-sensitive interrupt.

## Chapter 28 General Register File in CPU System

### 28.1 Overview

In CPU system, the General Register File will be used to do static set by software, which is composed of many registers for system control.

### 28.2 Registers

This section describe the registers for this module. In them , the CPU\_APB\_REG $i(i=0\sim3)$  is only for read and used to record the system status inside CPU sub-system. Another, the CPU\_APB\_REG $i(i=4\sim7)$  will focus on the general control inside CPU sub-system.

#### 28.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
CPU_APB_REG0	0x0000	w	0x0	General controller register0
CPU_APB_REG1	0x0004	w	0x0	General controller register1
CPU_APB_REG2	0x0008	w	0x0	General controller register2
CPU_APB_REG3	0x000c	w	0x0	General controller register3
CPU_APB_REG4	0x0010	w	0x00040000	General controller register4
CPU_APB_REG5	0x0014	w	0x0	General controller register5
CPU_APB_REG6	0x0018	w	0x0	General controller register6
CPU_APB_REG7	0x001c	w	0x0	General controller register7
IOMUX_A_CON	0x0020	w	0x0	IO MUX controller register
IOMUX_B_CON	0x0024	w	0x0	IO MUX controller register
GPIO0_AB_PU_CON	0x0028	w	0x55555555	Pull up or down control for GPIO0 A and B group
GPIO0_CD_PU_CON	0x002c	w	0x55555555	Pull up or down control for GPIO0 C and D group
GPIO1_AB_PU_CON	0x0030	w	0xaaaa55aa	Pull up or down control for GPIO1 A and B group
GPIO1_CD_PU_CON	0x0034	w	0xaaaaaaaa	Pull up or down control for GPIO1 C and D group
OTGPHY_CON0	0x0038	w	0x16fbc963	OTG PHY Control signals
OTGPHY_CON1	0x003c	w	0x00000008	OTG PHY Control signals

#### 28.2.2 Detail Registers Description

##### CPU\_APB\_REG0

Address : Base Addr+0x00

bit	Attr	Reset Value	Description
31:14	R	-	Reserved
13	R	0x0	Lcd bypass enable status, also is value from

			IO_LCDDBP pin
12:11	R	0x0	External memory select status , also is value from IO_EXTDDR_SEL and IO_EXTMSDR_SEL pins 00 : SDRAM 01 : Mobile SDRAM 10 : DDRII 11 : Mobile DDR
10	R	0x0	cx_ocm_core_rst status
9	R	0x0	codec pll lock status
8	R	0x0	dsp pll lock status
7	R	0x0	cpu pll lock status
6:4	R	0x0	timer2/1/0 en status
3	R	0x0	Spi slave controller sleep status
2	R	0x0	Spi master controller sleep status
1	R	0x0	utmi_linestate[1]
0	R	0x0	utmi_linestate[0]

**CPU\_APB\_REG1**

Address : Base Addr+0x04

bit	Attr	Reset Value	Description
31:15	R	0x0	Reserved
14	R	0x0	DDRII/MDDR idle drive enable state
13	R	0x0	DDRII/MDDR cke state
12	R	0x0	DDRII/MDDR interrupt state
11	R	0x0	DDRII/MDDR self-refresh state
10	R	0x0	DDRII/MDDR into self-refresh ack state
9	R	0x0	DDRII/MDDR controller quene almost full state
8	R	0x0	DDRII/MDDR controller busy state
7:0	R	0x0	DDRII/MDDR 8 data ports busy state

**CPU\_APB\_REG2**

Address : Base Addr+0x08

bit	Attr	Reset Value	Description
31:0	-	-	Reserved

**CPU\_APB\_REG3**

Address : Base Addr+0x0c

bit	Attr	Reset Value	Description
31:0	-	-	Reserved

**CPU\_APB\_REG4**

Address : Base Addr+0x10

bit	Attr	Reset Value	Description
31	RW	0x0	LCDC share memory enable 0: disable , memory is used inside LCDC 1: enable, memory is not used for LCDC, but for system as embedded sram

30	RW	0x0	Spi master controller interface type select
29	RW	0x0	arbiter in exp bus enter pause mode control. Active high
28	RW	0x0	arbiter in armd bus enter pause mode control. Active high
27	RW	0x0	host interface enable. Active high 0: disable , internal buffer is used as share memory between two cores 1: enable, dsp can not access internal buffer
26	RW	0x0	Host inerafce data bus bit control 0 : 8bit 1 : 16bit
25	RW	0x0	LCDC bypass mask bit. 0 : disable 1 : enable
24	RW	0x0	VIP IO voltage 1.8V enable 0 : disable , 2.5V or 3.3V 1 : enable , 1.8V
23	RW	0x0	Sdram or Mobile SDRAM controller write pipe control signals. Active low 0 : enable 1 : disable <i>*notes : it is better not to change this bit in general application.</i>
22	RW	0x0	SDRAM or Mobile SDRAM read pipe control signals. Active high 0 : disable 1 : enable <i>*notes : it is better to change this bit in general application.</i>
21	RW	0x0	SDRAM or Mobile SDRAM exit self refresh control. Active high
20	RW	0x0	static memory power down control. Active high
19	RW	0x0	SDRAM or Mobile SDRAM power down control. Active high
18:16	RW	0x100	nor flash(Static Memory) data bus width select signals 3'b000 : 16bit 3'b100 : 8bit Others : reserved
15	RW	0x0	Mobile Sdram controller select signals 0 : sdr sdram 1 : mobile sdram

14:0	RW	0x0	priority set for 6 data ports for SDRAM or Mobile SDRAM controller,  *Notes : operation together with CPU_APB_REG5 bits[26:24] to control 6 data port
------	----	-----	-------------------------------------------------------------------------------------------------------------------------------------------------------------

**CPU\_APB\_REG5**

Address : Base Addr+0x14

bit	Attr	Reset Value	Description
31	RW	0x0	Reserved
30	RW	0x0	DDR PHY dfi_odt signal source select 0: use register version of dfi_odt from DDRII/Mobile DDR controller 1: use special reserved0 signal to support CAS 3 with ODT control  *Notes : In general , we can set this bit as 1'b1
29	RW	0x0	Deblocking(RV) master0 (In EXP Bus) bridge wrapper select 0: no bridge wrapper 1: bridge wrapper enable  *Notes : In general , we do not change this bits
28	RW	0x0	LCDC IO voltage 1.8V enable 0 : disable , 2.5V or 3.3V 1 : enable , 1.8V
27	RW	0x0	Arbiter2 in VIDEO BUS enter pause mode control. Active high
26:24	RW	0x0	Extended priority set for 6 data ports in SDRAM or Mobile SDRAM controller  *Notes : operation together with CPU_APB_REG4 bits[14:0] to control 6 data port
23	RW	0x0	Method select for force External memory out of self-refresh state in stop mode 0 : software control 1 : hardware control  *Notes : If software method is selected, we must set another related register inside controller to make it out of self-refresh state
22	RW	0x0	Method select for force External memory into self-refresh state in stop mode 0 : software control 1 : hardware control  *Notes : If software method is selected, we must set another related register inside controller to make it enter into self-refresh state
21	RW	0x0	Different IO MUX solution for I2S lrck signal 0 : share lrck signal for tx and rx 1 : separate lrck signal for tx and rx  *Notes : If this bit is enable, lrck for tx and rx is separate, rx lrck is mux with IO_GPIO0_A[4], and tx lrck is original lrck ; If this bit is disable, only one lrck is connected to IO pin.

20	RW	0x0	Flash IO voltage 1.8V enable 0 : disable , 2.5V or 3.3V 1 : enable , 1.8V
19	RW	0x0	Customized Video SDRAM Memory controller select 0 : disable 1: enable
18	RW	0x0	Force DDRII or Mobile DDR enter into self-refresh state. Active high
17	RW	0x0	DDRII/Mobile DDR controller select by software set 0 : select SDRAM/Mobiel SDRAM controller 1 : select DDRII/Mobile DDR controller
16	RW	0x0	VIP bypass to LCD select 0 : disable 1 : enable
15:14	RW	0x0	LCD panel type select for bypass  <i>* Notes : The detailed mapping relationship between LCDC IO pins and Host IO pins, please refer to Chapter 14 Host Interface</i>
13	RW	0x0	Deblocking(RV) master1(In ARMD BUS) bridge wrapper select 0: no bridge wrapper 1: bridge wrapper enable  <i>* Notes : In general , we do not change this bits</i>
12	RW	0x0	ARM itcm & dtcm wait select: 0 : zero wait cycle 1 : one wait cycle
11:10	RW	0x0	Reserved
9:8	RW	0x0	Reserved
7	RW	0x0	VIP Vsync valid porality control: 0 : low valid (default) 1 : high valid
6	RW	0x0	Reserved
5	RW	0x0	CPU to DSP interrupt (nmi) .
4	RW	0x0	DSP boot control. Active high
3	RW	0x0	DSP external wait control . Active high
2	RW	0x0	DSP wake up control . Active high
1	RW	0x0	fiq interrupt control to CPU
0	RW	0x0	remap control . Active high

**CPU\_APB\_REG6**

Address : Base Addr+0x18

bit	Attr	Reset Value	Description
-----	------	-------------	-------------



31:0	RW	0x0	DSP boot vector address
------	----	-----	-------------------------

**CPU\_APB\_REG7**

Address : Base Addr+0x1c

bit	Attr	Reset Value	Description
31:30	RW	0x0	IO pullup/pulldown control for 18bits host data interface 00 : pull down 11 : pull up Others : reserved
29	RW	0x0	External dynamic memory cke signal level control: (used for different types of memory) 0 : keep original level 1 : force cke in low level
28:12	RW	0x0	Reserved
11:10	RW	0x0	undefined length INCR read strategy select for high-speed ahb2ahb bridge in DSP slave interface(When CPU subsystem access internal slave inside DSP) 00 : keep incr transfer 01 : change to INCR4 read 10: change to INCR8 read 11 : change to INCR16 read
9:8	RW	0x0	block size for write operation of high-performace mode for high-speed ahb2ahb bridge in DSP slave interface(When CPU subsystem access internal slave inside DSP) 00 : equal to burst length 01 : fixed INCR4
7	RW	0x0	write strategy select for high-speed ahb2ahb bridge in DSP slave interface(When CPU subsystem access internal slave inside DSP) 0 : low-latency, data transfer will start at once after receive one operation 1 : high-performance , data transfer start after receive one block operation
6	RW	0x0	DSP slave interface bridge select (When CPU subsystem access internal slave inside DSP) 0 : low-speed ahb2ahb bridge 1 : high-speed ahb2ahb bridge
5:4	RW	0x0	undefined length INCR read strategy for high-speed ahb2ahb bridge in DSP master interface(When DSP subsystem access slave inside CPU system) 00 : keep INCR transfer

			01 : change to INCR4 read 10: change to INCR8 read 11 : change to INCR16 read
3:2	RW	0x0	block size for write operation of high-performace mode for high-speed ahb2ahb bridge in DSP master interface(When DSP subsystem access slave inside CPU system)  00 : equal to burst length 01 : fixed INCR4
1	RW	0x0	write strategy select for high-speed ahb2ahb bridge in DSP master interface(When DSP subsystem access slave inside CPU system)  0 : low-latency, data transfer will start at once after receive one operation 1 : high-performance , data transfer start after receive one block operation
0	RW	0x0	DSP master interface bridge select in DSP master interface(When DSP subsystem access slave inside CPU system)  0 : low-speed ahb2ahb bridge 1 : high-speed ahb2ahb bridge

**IOMUX\_A\_CON**

Address : Base Addr+0x20

bit	Attr	Reset Value	Description
31	RW	0x0	Reserved
30	RW	0x0	gpio1a_i2c0_sel 0 : i2c0_sda/scl 1 : gpio1_a4/a5
29:28	RW	0x0	gpio1a_u1ir_i2c1 00 : gpio1_a6/a7 01 : uart1_sir_in/sir_out_n 10 : i2c1_sda/scl
27:26	RW	0x0	gpio1b1_uart1_cpwm1 00 : gpio1_b1 01 : uart1_sout 10 : cx_timer1_pwm
25:24	RW	0x0	gpio1b0_uart1_cpwm0 00 : gpio1_b0 01 : uart1_sin 10 : cx_timer0_pwm
23	RW	0x0	gpio1c_mmc1_sel 0 : gpio1_c2/c3/c7

			1: sdmmc1_cmd/data0/clkout
22	RW	0x0	gpio1c_mmc1d_sel 0 : gpio1_c4/c5/c6 1 : sdmmc1_data1/data2/data3
21:20	RW	0x0	gpio1a_spi1_flash_sel_2 00 : gpio1_a3/b7 01 : spi1_rxd/spi1_txd 11 : flash_cs6/flash_cs7
19:18	RW	0x0	gpio1a_spi1_flash_sel 00 : gpio1_a1/a2 01 : spi1_clkin/spi1_ss_in_n 10 : flash_cs4 / flash_cs5
17:16	RW	0x0	gpio0b0_spi0csn1_mmc1pca 00 : gpio0_b0 01 : spi0_csn1 10 : sdmmc1_pwr_en
15:14	RW	0x0	gpio1c1_uart0_mmc1wpt 00 : gpio1_c1 01 : uart0_sout 10 : sdmmc1_write_prt
13:12	RW	0x0	gpio1c0_uart0_mmc1det 00 : gpio1_c0 01 : uart0_sin 10 : sdmmc1_detect_n
11:10	RW	0x0	gpio1b4_apwm2_mmc0wpt 00 : gpio1_b4 01 : pwm2 10 : sdmmc0_write_prt
9:8	RW	0x0	gpio1b3_apwm1_mmc0detn 00 : gpio1_b3 01 : pwm1 10 : sdmmc0_detect_n
7:6	RW	0x0	gpio0b1_smcs1_mmc0pca 00 : gpio0_b1 01 : sm_cs1_n 10 : sdmmc0_pwr_en
5	RW	0x0	gpio1d_mmc0d_sel 0 : gpio1_d2/d3/d4 1 : sdmm0_data1/data2/data3

4	RW	0x0	gpio1d_mmc0_sel 0 : gpio1_d0/d1/d5 1 : sdmmc0_cmd/data0/clkout
3:2	RW	0x0	gpio0b_spi0_mmc0 00 : gpio0_b5/b6/b7 01 : spi0_clkout/spi0_txd/spi0_rxd 10 : sdmmc0_data5/data6/data7
1:0	RW	0x0	gpio0b4_spi0cs0_mmc0d4 00 : gpio0_b4 01 : spi0_csn0 10 : sdmmc0_data4

**IOMUX\_B\_CON**

Address : Base Addr+0x24

bit	Attr	Reset Value	Description
31	RW	0x0	gpio1b34_uart3_sel 0 : gpio1_b3 / gpio1_b4 function is decided by gpio1b3_apwm1_mmc0detn(IOMUX_A_CON[9:8]) and gpio1b4_apwm2_mmc0wpt(IOMUX_A_CON[11:10]) separately 1 : uart3_sin / uart3_sout
30	RW	0x0	gpio0b01_uart2_sel 0 : gpio0_b0 / gpio0_b1 function is decided by gpio0b0_spi0csn1_mmc1pca(IOMUX_A_CON [17:16]) and gpio0b1_smcs1_mmc0pca(IOMUX_A_CON [7:6]) separately 1 : uart2_sin / uart2_sout
29	RW	0x0	gpio0a23_uart2_sel 0 : gpio0_a2/gpio0_a3 1 : uart2_cts_n / uart2_rts_n
28:27	RW	0x00	cxgpio_hsadc_norflash_sel 00 : gpio2_22 ~ gpio2_23 01 : hsadc_data_q[9:8] 10 : sm_we_n/sm_oe_n
26:25	RW	0x0	cxgpio_hsadc_hif_sel 00 : gpio2_14 ~ gpio2_21 01 : hsadc_data_q[7:0] 10 : host_data[15:8]
24	RW	0x0	gpio0a1_hostdata17_sel 0 : gpio0_a1 1: host data 17
23	RW	0x0	gpio0a0_hostdata16_sel 0 : gpio0_a0 1: host data 16

22	RW	0x0	gpio1a_vipdata0_sel 0: gpio1_a0 1: vip_data0
21:20	RW	0x0	cxgpio_gpssclk_hsadcclkout 00 : gpio2_24 01 : gps clk 10 : hsadc_clkout
19	RW	0x0	hsadcdata_tscon_sel 0: hsadc_data_i[9:8] 1: ts_fail / ts_valid
18	RW	0x0	gpio0a7_flashcs3_sel 0 : gpio0_a7 1 : flash_cs3
17	RW	0x0	gpio0a6_flashcs2_sel 0 : gpio0_a6 1 : flash_cs2
16	RW	0x0	gpio0a5_flashcs1_sel 0 : gpio0_a5 1 : flash_cs1
15:14	RW	0x0	gpio1b5_apwm3_vipdata1 00 : gpio1_b5 01 : pwm3 10 : vip_data 1
13	RW	0x0	gpio0b3_u0rtsn_sel 0 : gpio0_b3 1 : uart0_rts_n
12	RW	0x0	gpio0b2_u0ctsn_sel 0 : gpio0_b2 1 : uart0_cts_n
11	RW	0x0	gpio1b2_apwm0_sel 0 : gpio1_b2 1 : pwm0
10	RW	0x0	gpio0d_lcdc16bit_sel 0 : gpio0_d0 ~ gpio0_d7 1 : lcdc_data8 ~ lcdc_data15
9	RW	0x0	gpio0c_lcdc24bit_sel 0 : gpio0_c2 ~ gpio0_c7 1 : lcdc_data18 ~ lcdc_data23
8	RW	0x0	gpio0c_lcdc18bit_sel

			0 : gpio0_c0/c1 1 : lcdc_data16 ~ lcdc_data17
7	RW	0x0	cxgpio_lcddden_sel 0 : gpio2_26 1 : lcdc_denable
6	RW	0x0	cxgpio_lcdvsync_sel 0 : gpio2_25 1 : lcdc_vsync
5	RW	0x0	gpio0a4_i2slrck_sel 0: gpio0_a4 1: i2s lrck_rx
4	RW	0x0	cxgpio_host_sel 0 : gpio2_0 ~ gpio2_13 1 : host interface
3	RW	0x0	gpio1d7_vipdata3_sel 0 : gpio1_d7 1 : vip data 3
2	RW	0x0	gpio1d6_vipdata2_sel 0 : gpio1_d6 1 : vip data 2
1	RW	0x0	cxgpio_i2s_sel 0 : i2s interface 1 : gpio2_27 ~ gpio2_31
0	RW	0x0	gpio1b6_vipclk_sel 0 : gpio1_b6 1 : vip clkout

**GPIO0\_AB\_PU\_CON**

Address : Base Addr+0x28

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO0 A0 [3:2] GPIO0 A1 ... [17:16] GPIO0 B0 ... [31:30] GPIO0 B7 00 : Normal 01 : Pull UP 10 : Pull Down 11 : Reserved

**GPIO0\_CD\_PU\_CON**

Address : Base Addr+0x2c

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO0 C0 [3:2] GPIO0 C1 ... [17:16] GPIO0 D0 ... [31:30] GPIO0 D7  00 : Normal 01 : Pull Up 10 : Pull Down 11 : Reserved

**GPIO1\_AB\_PU\_CON**

Address : Base Addr+0x30

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO1 A0 [3:2] GPIO1 A1 ... [17:16] GPIO1 B0 ... [31:30] GPIO1 B7  00 : Normal 01 : Pull Up 10 : Pull Down 11 : Reserved

**GPIO1\_CD\_PU\_CON**

Address : Base Addr+0x34

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO1 C0 [3:2] GPIO1 C1 ... [17:16] GPIO1 D0 ... [31:30] GPIO1 D7  00 : Normal 01 : Pull Up 10 : Pull Down



			11 : Reserved
--	--	--	---------------

**OTGPHY\_CON0**

Address : Base Addr+0x38

*\*Notes : In general, we do not use this register*

bit	Attr	Reset Value	Description
31	RW	0x0	Reserved
30	RW	0x0	usbphy_txrise_tune
29:28	RW	01	usbphy_txhsxv_tune
27:24	RW	0110	usbphy_txvref_tune
23:20	RW	1111	usbphy_txfsls_tune
19	RW	1	usbphy_txfreemphases_tune
18:16	RW	011	usbphy_sqrxtune
15	RW	1	usbphy_txbitstuff_enh
14	RW	1	usbphy_txbitstuff_en
13	RW	0	usbphy_siddq
12	RW	0	usbphy_port_reset
11:10	RW	10	usbphy_refclk_sel
9:8	RW	01	usbphy_refclk_div
7:5	RW	011	usbphy_otg_tune
4	RW	0	usbphy_otg_disable
3:1	RW	001	usbphy_compdistune
0	RW	1	usb phy_common_on_n

**OTGPHY\_CON1**

Address : Base Addr+0x3c

*\*Notes: This register will be used to control the IO status or function for USB HOST 1.1 and debug for USB OTG 2.0*

bit	Attr	Reset Value	Description
31	RW	0x0	USB HOST low power enable 0 : disable 1 : enable
30	RW	0x0	USB HOST DP/DN pulldown control 0 : disable 1 : enable
29	RW	0x0	USB HOST DP pulldown control 0 : enable 1 : disable
28	RW	0x0	USB HOST DN pulldown control 0 : enable 1 : disable
27	RW	0x0	USB HOST DP pullup master control

			0 : disable 1 : enable
26	RW	0x0	USB HOST DP pullup slave control 0 : disable 1 : enable
25	RW	0x0	USB HOST DN pullup master control 0 : disable 1 : enable
24	RW	0x0	USB HOST DN pullup slave control 0 : disable 1 : enable
23:9	RW	0x0	Reserved
8	RW	0	USB OTG utmi_termselect
7:6	RW	00	USB OTG utmi_xcvsselect[1:0]
5:4	RW	00	USB OTG utmi_opmode[1:0]
3	RW	1	USB OTG utmi_suspend_n
2	RW	0	USB OTG usbphy_soft_con_sel : 0 : software control usb phy disable 1 : software control usb phy enable
1	RW	0	USB OTG usbphy_vbus_vld_extsel
0	RW	0	USB OTG usbphy_vbus_vld_ext

## Chapter 29 Port Multiplexer

### 29.1 Overview

RK281x has a lot of general purpose IOs which have been described in Chapter 31 and Chapter 32. All of them are set to input mode at reset.

Most of IOs have the multiple functions shared by programmable register set. And can also be pulled-up or pulled-down by reconfigurable register. As for the detailed description for these registers, please refer to register IOMUX\_A\_CON / IOMUX\_B\_CON / GPIO0\_AB\_PU\_CON / GPIO0\_CD\_PU\_CON / GPIO1\_AB\_PU\_CON / GPIO1\_CD\_PU\_CON in Chapter 34 .

### 29.2 Detailed description for IO MUX

The following table shows the detailed multiplexer for all GPIOs.

Table 29-1 RK281x IO MUX List

PAD NAME	PORT Name	PAD Direction	Pin Description
CPU GPIO0 A			
IO_GPIO0_A[0]	gpio0_a[0]	B Pull Up	gpio
	host_data[16]	I	host data bit16
IO_GPIO0_A[1]	gpio0_a[1]	B Pull Up	gpio
	host_data[17]	I	host data bit17
IO_GPIO0_A[2]	gpio0_a[2]	B Pull Up	gpio
	uart2_cts_n	I	uart2 modem signal
IO_GPIO0_A[3]	gpio0_a[3]	B Pull Up	gpio
	uart2_rts_n	O	uart2 modem signal
IO_GPIO0_A[4]	gpio0_a[4]	B Pull Up	gpio
	i2s_lrck_rx	B	i2s lrck rx
IO_GPIO0_A[5]	gpio0_a[5]	B Pull Up	gpio
	flash_cs1	O	nand flash cs1
IO_GPIO0_A[6]	gpio0_a[6]	B Pull Up	gpio
	flash_cs2	O	nand flash cs2
IO_GPIO0_A[7]	gpio0_a[7]	B Pull Up	gpio
	flash_cs3	O	nand flash cs3
CPU GPIO0 B			
IO_GPIO0_B[0]	gpio0_b[0]	B Pull Up	gpio
	spi0_csn1	O	spi0 second chip select
	sdmmc1_pwr_en	O	sdmmc1 power control
	uart2_sin	I	uart2 serial in
IO_GPIO0_B[1]	gpio0_b[1]	B Pull Up	gpio
	sm_cs1_n	O	nor flash second chip select
	sdmmc0_pwr_en	O	sdmmc0 power control
	uart2_sout	O	uart2 serial out
IO_GPIO0_B[2]	gpio0_b[2]	B Pull Up	gpio
	uart0_cts_n	I	uart0 modem signal
IO_GPIO0_B[3]	gpio0_b[3]	B Pull Up	gpio
	uart0_rts_n	O	uart0 modem signal
IO_GPIO0_B[4]	gpio0_b[4]	B Pull Up	gpio
	spi0_csn0	O	spi0 first chip select
	sdmmc0_data[4]	B	sdmmc0 data bit4

IO_GPIO0_B[5]	gpio0_b[5]	B Pull Up	gpio
	spi0_clkout	O	spi0 clk out
	sdmmc0_data[5]	B	sdmmc0 data bit5
IO_GPIO0_B[6]	gpio0_b[6]	B Pull Up	gpio
	spi0_txd	O	spi0 txd
	sdmmc0_data[6]	B	sdmmc0 data bit6
IO_GPIO0_B[7]	gpio0_b[7]	B Pull Up	gpio
	spi0_rxd	I	spi0 rxd
	sdmmc0_data[7]	B	sdmmc0 data bit7

## CPU GPIO0 C

IO_GPIO0_C[0]	gpio0_c[0]	B Pull Up	gpio
	lcdc_data16	O	lcdc data bit16
IO_GPIO0_C[1]	gpio0_c[1]	B Pull Up	gpio
	lcdc_data17	O	lcdc data bit17
IO_GPIO0_C[2]	gpio0_c[2]	B Pull Up	gpio
	lcdc_data18	O	lcdc data bit18
IO_GPIO0_C[3]	gpio0_c[3]	B Pull Up	gpio
	lcdc_data19	O	lcdc data bit19
IO_GPIO0_C[4]	gpio0_c[4]	B Pull Up	gpio
	lcdc_data20	O	lcdc data bit20
IO_GPIO0_C[5]	gpio0_c[5]	B Pull Up	gpio
	lcdc_data21	O	lcdc data bit21
IO_GPIO0_C[6]	gpio0_c[6]	B Pull Up	gpio
	lcdc_data22	O	lcdc data bit22
IO_GPIO0_C[7]	gpio0_c[7]	B Pull Up	gpio
	lcdc_data23	O	lcdc data bit23

## CPU GPIO0 D

IO_GPIO0_D[0]	gpio0_d[0]	B Pull Up	gpio
	lcdc_data8	O	lcdc data bit8
IO_GPIO0_D[1]	gpio0_d[1]	B Pull Up	gpio
	lcdc_data9	O	lcdc data bit9
IO_GPIO0_D[2]	gpio0_d[2]	B Pull Up	gpio
	lcdc_data10	O	lcdc data bit10
IO_GPIO0_D[3]	gpio0_d[3]	B Pull Up	gpio
	lcdc_data11	O	lcdc data bit11
IO_GPIO0_D[4]	gpio0_d[4]	B Pull Up	gpio
	lcdc_data12	O	lcdc data bit12
IO_GPIO0_D[5]	gpio0_d[5]	B Pull Up	gpio
	lcdc_data13	O	lcdc data bit13
IO_GPIO0_D[6]	gpio0_d[6]	B Pull Up	gpio
	lcdc_data14	O	lcdc data bit14
IO_GPIO0_D[7]	gpio0_d[7]	B Pull Up	gpio
	lcdc_data15	O	lcdc data bit15

## CPU GPIO1 A

IO_GPIO1_A[0]	gpio1_a[1]	B Pull Down	gpio
	vip_data[0]	I Pull Down	vip input data bit0
IO_GPIO1_A[1]	gpio1_a[1]	B Pull Down	gpio
	spi1_clkin	I	spi1 slave mode clock signal
	flash_cs4	O	nand flash cs4
IO_GPIO1_A[2]	gpio1_a[2]	B Pull Down	gpio
	spi1_ss_n	I	spi1 slave mode select signal
	flash_cs5	O	nand flash cs5

IO_GPIO1_A[3]	gpio1_a[3]	B Pull Down	gpio
	spi1_rxd	I	spi1 rxd
	flash_cs6	O	nand flash cs6
IO_GPIO1_A[4]	i2c0_sda	B Pull Up	i2c0 sda
	gpio1_a[4]	B Pull UP	gpio
IO_GPIO1_A[5]	i2c0_scl	B Pull UP	i2c0 scl
	gpio1_a[5]	B Pull UP	gpio
IO_GPIO1_A[6]	gpio1_a[6]	B Pull UP	gpio
	uart1_sir_in	I	uart1 IR data in
	i2c1_sda	B Pull UP	i2c1 sda
IO_GPIO1_A[7]	gpio1_a[7]	B Pull UP	gpio
	uart1_sir_out_n	O	uart1 IR data out
	i2c1_scl	B Pull UP	i2c1 scl

## CPU GPIO1 B

IO_GPIO1_B[0]	gpio1_b[0]	B Pull Down	gpio
	uart1_sin	I	uart1 serial data in
	cx_timer0_pwm	O	pwm out from dsp
IO_GPIO1_B[1]	gpio1_b[1]	B Pull Down	gpio
	uart1_sout	O	uart1 serial data out
	cx_timer1_pwm	O	pwm out from dsp
IO_GPIO1_B[2]	gpio1_b[2]	B Pull Down	gpio
	pwm0	B	pwm
IO_GPIO1_B[3]	gpio1_b[3]	B Pull Down	gpio
	pwm1	B	pwm
	sdmmc0_detect_n	I	sdmmc0 detect signal
	uart3_sin	I	uart3 serial in
IO_GPIO1_B[4]	gpio1_b[4]	B Pull Down	gpio
	pwm2	B	pwm
	sdmmc0_write_prt	I	sdmmc0 write protect
	uart3_sout	O	uart3 serial out
IO_GPIO1_B[5]	gpio1_b[5]	B Pull Down	gpio
	pwm3	B	pwm
	vip_data[1]	I Pull Down	vip input data bit1
IO_GPIO1_B[6]	gpio1_b[6]	B Pull Down	gpio
	vip_clkout	O	sensor clk out
IO_GPIO1_B[7]	gpio1_b[7]	B Pull Down	gpio
	spi1_txd	O	spi1 txd
	flash_cs7	O	nand flash cs7

## CPU GPIO1 C

IO_GPIO1_C[0]	gpio1_c[0]	B Pull Down	gpio
	uart0_sin	I	uart0 serial data in
	sdmmc1_detect_n	I	sdmmc1 card detect
IO_GPIO1_C[1]	gpio1_c[1]	B Pull Down	gpio
	uart0_sout	O	uart0 serial data out
	sdmmc1_write_prt	I	sdmmc1 card write protect
IO_GPIO1_C[2]	gpio1_c[2]	B Pull Down	gpio
	sdmmc1_cmd	B	sdmmc1 command
IO_GPIO1_C[3]	gpio1_c[3]	B Pull Down	gpio
	sdmmc1_data[0]	B	sdmmc1 data bit0
IO_GPIO1_C[4]	gpio1_c[4]	B Pull Down	gpio
	sdmmc1_data[1]	B	sdmmc1 data bit1
IO_GPIO1_C[5]	gpio1_c[5]	B Pull Down	gpio

	sdmmc1_data[2]	B	sdmmc1 data bit2
IO_GPIO1_C[6]	gpio1_c[6]	B Pull Down	gpio
	sdmmc1_data[3]	B	sdmmc1 data bit3
IO_GPIO1_C[7]	gpio1_c[7]	B Pull Down	gpio
	sdmmc1_clkout	O	sdmmc1 clk out

## CPU GPIO1 D

IO_GPIO1_D[0]	gpio1_d[0]	B Pull Down	gpio
	sdmmc0_cmd	B	sdmmc0 command
IO_GPIO1_D[1]	gpio1_d[1]	B Pull Down	gpio
	sdmmc0_data[0]	B	sdmmc0 data bit0
IO_GPIO1_D[2]	gpio1_d[2]	B Pull Down	gpio
	sdmmc0_data[1]	B	sdmmc0 data bit1
IO_GPIO1_D[3]	gpio1_d[3]	B Pull Down	gpio
	sdmmc0_data[2]	B	sdmmc0 data bit2
IO_GPIO1_D[4]	gpio1_d[4]	B Pull Down	gpio
	sdmmc0_data[3]	B	sdmmc0 data bit3
IO_GPIO1_D[5]	gpio1_d[5]	B Pull Down	gpio
	sdmmc0_clkout	O	sdmmc0 clock out
IO_GPIO1_D[6]	gpio1_d[6]	B Pull Down	gpio
	vip_data[2]	I Pull Down	vip input data bit2
IO_GPIO1_D[7]	gpio1_d[7]	B Pull Down	gpio
	vip_data[3]	I Pull Down	vip input data bit3

## DSP GPIO

IO_GPIO2[0]	gpio2[0]	B Pull UP	gpio
	host_data0	B	host interface data bit0
IO_GPIO2[1]	gpio2[1]	B Pull UP	gpio
	host_data1	B	host interface data bit1
IO_GPIO2[2]	gpio2[2]	B Pull UP	gpio
	host_data2	B	host interface data bit2
IO_GPIO2[3]	gpio2[3]	B Pull UP	gpio
	host_data3	B	host interface data bit3
IO_GPIO2[4]	gpio2[4]	B Pull UP	gpio
	host_data4	B	host interface data bit4
IO_GPIO2[5]	gpio2[5]	B Pull UP	gpio
	host_data5	B	host interface data bit5
IO_GPIO2[6]	gpio2[6]	B Pull UP	gpio
	host_data6	B	host interface data bit6
IO_GPIO2[7]	gpio2[7]	B Pull UP	gpio
	host_data7	B	host interface data bit7
IO_GPIO2[8]	gpio2[8]	B Pull UP	gpio
	host_addr0	I	host interface addr bit0
IO_GPIO2[9]	gpio2[9]	B Pull UP	gpio
	host_addr1	I	host interface addr bit1
IO_GPIO2[10]	gpio2[10]	B Pull UP	gpio
	host_csn	I	host interface chip select
IO_GPIO2[11]	gpio2[11]	B Pull UP	gpio
	host_rdn	I	host interface read valid
IO_GPIO2[12]	gpio2[12]	B Pull UP	gpio
	host_wrn	I	host interface write valid
IO_GPIO2[13]	gpio2[13]	B Pull UP	gpio
	ap2bb_int	O	host interface interrupt from

			chip to host
IO_GPIO2[14]	gpio2[14]	B Pull UP	gpio
	host_data[8]	B Pull Down	host interface data bit8
	hsadc_data_q[0]	I	hsadc data bit0 for Q path
IO_GPIO2[15]	gpio2[15]	B Pull UP	gpio
	host_data[9]	B Pull Down	host interface data bit8
	hsadc_data_q[1]	I	hsadc data bit1 for Q path
IO_GPIO2[16]	gpio2[16]	B Pull Down	gpio
	host_data[10]	B Pull Down	host interface data bit8
	hsadc_data_q[2]	I	hsadc data bit2 for Q path
IO_GPIO2[17]	gpio2[17]	B Pull Down	gpio
	host_data[11]	B Pull Down	host interface data bit8
	hsadc_data_q[3]	I	hsadc data bit3 for Q path
IO_GPIO2[18]	gpio2[18]	B Pull Down	gpio
	host_data[12]	B Pull Down	host interface data bit8
	hsadc_data_q[4]	I	hsadc data bit4 for Q path
IO_GPIO2[19]	gpio2[19]	B Pull Down	gpio
	host_data[13]	B Pull Down	host interface data bit8
	hsadc_data_q[5]	I	hsadc data bit5 for Q path
IO_GPIO2[20]	gpio2[20]	B Pull Down	gpio
	host_data[14]	B Pull Down	host interface data bit8
	hsadc_data_q[6]	I	hsadc data bit6 for Q path
IO_GPIO2[21]	gpio2[21]	B Pull Down	gpio
	host_data[15]	B Pull Down	host interface data bit8
	hsadc_data_q[7]	I	hsadc data bit7 for Q path
IO_GPIO2[22]	gpio2[22]	B Pull Down	gpio
	sm_we_n	O	SRAM wen
	hsadc_data_q[8]	I	hsadc data bit8 for Q path
IO_GPIO2[23]	gpio2[23]	B Pull Down	gpio
	sm_oe_n	O	SRAM oen
	hsadc_data_q[9]	I	hsadc data bit9 for Q path
IO_GPIO2[24]	gpio2[24]	B Pull Down	gpio
	gps_clk	I Pull Down	clock input for gps application
	hsadc_clkout	O	clock out to hsadc analog
IO_GPIO2[25]	gpio2[25]	B Pull Down	gpio
	lcdc_vsync	O	lcdc vertical sync signal
IO_GPIO2[26]	gpio2[26]	B Pull Down	gpio
	lcdc_denable	O	lcdc data valid signal
IO_GPIO2[27]	i2s_sdi	I Pull Down	i2s sdi from codec
	gpio2[27]	B Pull Down	gpio
IO_GPIO2[28]	i2s_sdo	O	i2s sdo to codec
	gpio2[28]	B Pull Down	gpio
IO_GPIO2[29]	i2s_clk	O	i2s clock out to codec
	gpio2[29]	B Pull Down	gpio
IO_GPIO2[30]	i2s_lrck	B Pull Down	i2s lrck
	gpio2[30]	B Pull Down	gpio
IO_GPIO2[31]	i2s_sclk	B Pull Down	i2s serial clock
	gpio2[31]	B Pull Down	gpio



Notes :    *B* --- Bidirectional IO  
              *I* --- Input IO  
              *O* --- Output IO

PRELIMINARY

## Chapter 30 DDR2/mobile DDR SDRAM controller

### 39.1 Design Overview

#### 30.1.1 Overview

The DDR2/mobile DDR SDRAM controller is a memory controller that you can control DDR2 SDRAM, – as well as mobile DDR SDRAM – LPDDR1

#### 30.1.2 Features

##### AMBA AHB Interface Features

- AMBA AHB bus-compatible
- Supports all types of AMBA bursts
- Supports byte/half word/word transfer size
- Supports AHB data widths of 32 bits
- Supports AHB address width of 32 bits
- Supports early terminations on AHB transactions
- Does not generate split, retry, or error responses on the AMBA bus
- Supports one AHB slave interface for register configuration and six AHB slave interfaces for data access
- Supports the priority of six AHB slave interfaces configurable

##### DDR2/mobile DDR Interface Feature

- Supports connection to JEDEC-compliant DDR2 SDRAM
- Supports connection to JEDEC-compliant mobile DDR SDRAM(LPDDR1)
- Supports up to 15 DDR2/mobile DDR SDRAM address bits
- DDR2/mobile DDR SDRAM data width is 32 bits
- Supports reduced data path mode, only use low 16 bits data width
- Programmable row and column address bit widths up to:
  - ◆ 13-bit column address
  - ◆ 15-bit row addressAnd sum of column address bit widths, row address bit widths, bank address bit widths and number of chip selects up to 29
- Supports 3-bit bank address for DDR2 SDRAM and 2-bit bank address for mobile DDR SDRAM
- Supports up to 2 chip selects, with a maximum of 4 Gb of address space per chip select
- Supports DQS single-ended or differential configurable
- Supports CAS latency equal to 2/3/4/5/6 for DDR2 SDRAM
- Supports CAS latency equal to 2/3 for mobile DDR SDRAM
- Only supports burst length 4
- Supports DDR2 SDRAM clock frequency up to 266MHz
- Supports mobile DDR SDRAM clock frequency up to 200MHz
- Supports DLL bypass mode when in low clock frequency
- There are four user-selectable modes of clock relationship for AHB bus and controller core:
  - ◆ Synchronous
  - ◆ 1:2 Bus:core pseudo-synchronous
  - ◆ 2:1 Bus:core pseudo-synchronous
  - ◆ Asynchronous
- Supports self-refresh
- There are five low power modes available in the Memory Controller, The memory controller may enter and exit the various low power modes with automatic mode or manual mode:

- ◆ Memory power-down
- ◆ Memory power-down with memory clock gating
- ◆ Memory self-refresh
- ◆ Memory self-refresh with memory clock gating
- ◆ Memory self-refresh with memory and controller clock gating
- DDR2/mobile DDR SDRAM timing parameters can be programmed to values supported by different vendors

## 30.2 Architecture

This section provides a description about the functions and behavior under various conditions.

### 30.2.1 Block Diagram

The DDR2/mobile DDR SDRAM controller can provide an interface between DDR2/mobile DDR SDRAM memory devices and an AMBA AHB 2.0 bus.

The block comprises with:

- DDR2/mobile DDR memory controller core logic
- DDR2/mobile DDR memory phy core logic
- DDR2/mobile DDR memory pad logic

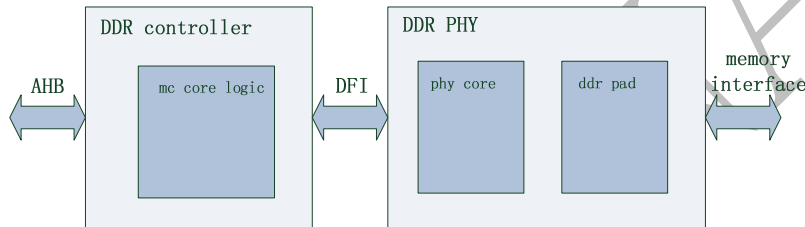


Fig. 30-1 DDR2/mobile DDR SDRAM controller architecture

## 30.3 Registers

This section describes the control/status registers of the design.

### 30.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
CTRL_REG_00	0x00	W	0x0	AUTO_REFRESH_MODE/AREFRESH/AP/ADDR_CMP_EN.
CTRL_REG_01	0x04	W	0x0	BDW_OVFLOW_0.
CTRL_REG_02	0x08	W	0x0	BDW_OVFLOW_1.
CTRL_REG_03	0x0C	W	0x0	CONCURRENTAP_WR_ONLY/CONCURRENTAP/CKE_STATUS/BANK_SPLIT_EN.
CTRL_REG_04	0x10	W	0x0	DQS_N_EN/DLL_BYPASS_MODE/DLLLOCKREG/CONFIG_ERROR.
CTRL_REG_05	0x14	W	0x0	ENABLE_QUICK_SREFRESH/EIGHT_BANK_MODE_1/EIGHT_BANK_MODE_0/DRIVE_DQ_DQS.
CTRL_REG_06	0x18	W	0x0	INTRPTWRITEA/INTRPTREADA/INTRPTAPBURST/FAST_WRITE.
CTRL_REG_07	0x1c	W	0x0	POWER_DOWN/PLACEMENT_EN/ODT_ALT_EN/NO_CMD_INIT.
CTRL_REG_08	0x20	W	0x0	REG_DIMM_ENABLE/REDUC/PWRUP_SREFRESH_EXIT/PRIORITY_EN.
CTRL_REG_09	0x24	W	0x0	SREFRESH/RW_SAME_EN/RESY

				NC_DLL_PER_AREF_EN/RESYNC_DLL.
CTRL_REG_10	0x28	W	0x0	TRAS_LOCKOUT/SWAP_PORT_R W_SAME_EN/SWAP_EN/START.
CTRL_REG_11	0x2c	W	0x0	AXI0_FIFO_TYPE_REG/WRITE_ MODEREG/WRITEINTERP/TREF_ ENABLE.
CTRL_REG_12	0x30	W	0x0	AXI1_R_PRIORITY/AXI1_FIFO_T YPE_REG/AXI0_W_PRIORITY/A XI0_R_PRIORITY.
CTRL_REG_13	0x34	W	0x0	AXI2_W_PRIORITY/AXI2_R_PRI ORITY/AXI2_FIFO_TYPE_REG/A XI1_W_PRIORITY.
CTRL_REG_14	0x38	W	0x0	AXI4_FIFO_TYPE_REG/AXI3_W _PRIORITY/AXI3_R_PRIORITY/A XI3_FIFO_TYPE_REG.
CTRL_REG_15	0x3c	W	0x0	AXI5_R_PRIORITY/AXI5_FIFO_T YPE_REG/AXI4_W_PRIORITY/A XI4_R_PRIORITY.
CTRL_REG_16	0x40	W	0x0	AXI6_W_PRIORITY/AXI6_R_PRI ORITY/AXI6_FIFO_TYPE_REG/A XI5_W_PRIORITY.
CTRL_REG_17	0x44	W	0x0	CS_MAP/AXI7_W_PRIORITY/AXI 7_R_PRIORITY/AXI7_FIFO_TYP E_REG.
CTRL_REG_18	0x48	W	0x00020000	ODT_RD_MAP_CS0/MAX_CS_RE G/LOWPOWER_REFRESH_ENAB LE/DRAM_CLK_DISABLE.
CTRL_REG_19	0x4c	W	0x0	ADDR_PINS_0/ODT_WR_MAP_C S1/ODT_WR_MAP_CS0/ODT_R D_MAP_CS1.
CTRL_REG_20	0x50	W	0x0	CKE_DELAY/CASLAT/ARB_CMD _Q_THRESHOLD/ADDR_PINS_1 .
CTRL_REG_21	0x54	W	0x0	Q_FULLNESS/PORT_DATA_ERR OR_TYPE/COLUMN_SIZE_1/COL UMN_SIZE_0.
CTRL_REG_22	0x58	W	0x0	R2W_SAMECS_DLY/R2W_DIFFC S_DLY/R2R_SAMECS_DLY/R2R_ DIFFCS_DLY.
CTRL_REG_23	0x5c	W	0x0	TRRD/TDFI_DRAM_CLK_DISABL E/TCKE/TBST_INT_INTERVAL.
CTRL_REG_24	0x60	W	0x0	W2W_DIFFCS_DLY/W2R_SAME CS_DLY/W2R_DIFFCS_DLY/TRT P.
CTRL_REG_25	0x64	W	0x0	ADD_ODT_CLK_SAMETYPE_DIF FCS/ADD_ODT_CLK_DIFFTYPE_ SAMECS/ADD_ODT_CLK_DIFFT YPE_DIFFCS/W2W_SAMECS_DL Y.
CTRL_REG_26	0x68	W	0x0	AHB3_RDLEN/AHB2_WRLLEN/AH B2_RDLEN/AGE_COUNT.
CTRL_REG_27	0x6c	W	0x0	AHB5_RDLEN/AHB4_WRLLEN/AH B4_RDLEN/AHB3_WRLLEN.
CTRL_REG_28	0x70	W	0x0	AHB7_RDLEN/AHB6_WRLLEN/AH B6_RDLEN/AHB5_WRLLEN.

CTRL_REG_29	0x74	W	0x0	CASLAT_LIN/APREBIT_1/APREBIT_0/AHB7_WRLLEN.
CTRL_REG_30	0x78	W	0x0	COMMAND_AGE_COUNT/CKSRX_CKSRE/CASLAT_LIN_GATE.
CTRL_REG_31	0x7c	W	0x0f0d0000	MAX_ROW_REG/MAX_COL_REG/INITAREF/DRAM_CLASS.
CTRL_REG_32	0x80	W	0x00040000	TDFI_CTRL_DELAY/TDFI_CTRLUPD_MIN/RDLAT_ADJ/PORT_CMD_ERROR_TYPE.
CTRL_REG_33	0x84	W	0x0	TDFI_PHY_WRLAT_BASE/TDFI_PHY_WRLAT/TDFI_PHY_RDLAT/TDFI_DRAM_CLK_ENABLE.
CTRL_REG_34	0x88	W	0x0	TRP_AB_0/TRP/TDFI_RDDATA_EN_BASE/TDFI_RDDATA_EN.
CTRL_REG_35	0x8c	W	0x0	WRLAT_ADJ/WRLAT/TWTR/TRP_AB_1.
CTRL_REG_36	0x90	W	0x0	OCD_ADJUST_PUP_CS_0/OCD_ADJUST_PDN_CS_0/LOWPOWER_CONTROL/LOWPOWER_AUTO_ENABLE.
CTRL_REG_37	0x94	W	0x0	TMRD/TDAL/TCKESR/TCCD.
CTRL_REG_38	0x98	W	0x0	TRC/TFW/OUT_OF_RANGE_TYPE/TWR_INT.
CTRL_REG_39	0x9c	W	0x0	BDW_0.
CTRL_REG_40	0xa0	W	0x0	BDW_1.
CTRL_REG_41	0xa4	W	0x0	BDW_2.
CTRL_REG_42	0xa8	W	0x0	BDW_3.
CTRL_REG_43	0xac	W	0x0	TMOD/DLL_RST_ADJ_DLY/DLL_LOCK/OUT_OF_RANGE_LENGTH.
CTRL_REG_44	0xb0	W	0x0	TRFC/TRCD_INT/TRAS_MIN.
CTRL_REG_45	0xb4	W	0x0	INT_MASK/INT_ACK.
CTRL_REG_46	0xb8	W	0x0	OUT_OF_RANGE_SOURCE_ID/INT_STATUS.
CTRL_REG_47	0xbc	W	0x0	PORT_DATA_ERROR_ID/PORT_CMD_ERROR_ID.
CTRL_REG_48	0xc0	W	0x0	TDFI_PHYUPD_RESP/TDFI_CTRLUPD_MAX.
CTRL_REG_49	0xc4	W	0x0	TDFI_PHYUPD_TYPE_0.
CTRL_REG_50	0xc8	W	0x0	TDFI_PHYUPD_TYPE_1.
CTRL_REG_51	0xcc	W	0x0	MR0_DATA_0/TREF.
CTRL_REG_52	0xd0	W	0x0	MR1_DATA_0/MR0_DATA_1.
CTRL_REG_53	0xd4	W	0x0	MR2_DATA_0/MR1_DATA_1.
CTRL_REG_54	0xd8	W	0x0	MR3_DATA_0/MR2_DATA_1.
CTRL_REG_55	0xdc	W	0x0	AXI0_EN_SIZE_LT_WIDTH_INSTR/MR3_DATA_1.
CTRL_REG_56	0xe0	W	0x0	AXI2_EN_SIZE_LT_WIDTH_INSTR/AXI1_EN_SIZE_LT_WIDTH_INSTR.
CTRL_REG_57	0xe4	W	0x0	AXI4_EN_SIZE_LT_WIDTH_INSTR/AXI3_EN_SIZE_LT_WIDTH_INSTR.
CTRL_REG_58	0xe8	W	0x0	AXI6_EN_SIZE_LT_WIDTH_INSTR/AXI5_EN_SIZE_LT_WIDTH_INSTR.
CTRL_REG_59	0xec	W	0x0	CS_MSK_0/AXI7_EN_SIZE_LT

				WIDTH INSTR.
CTRL_REG_60	0xf0	W	0x0	CS_VAL_0/CS_MSK_1.
CTRL_REG_61	0xf4	W	0x0	DLL_RST_DELAY/CS_VAL_1.
CTRL_REG_62	0xf8	W	0x0	LOWPOWER_INTERNAL_CNT/LOWPOWER_EXTERNAL_CNT.
CTRL_REG_63	0xfc	W	0x0	LOWPOWER_REFRESH_HOLD/LOWPOWER_POWER_DOWN_CNT.
CTRL_REG_64	0x100	W	0x0	TCPD/LOWPOWER_SELF_REFRESH_CNT.
CTRL_REG_65	0x104	W	0x0	TPDEX/TDLL.
CTRL_REG_66	0x108	W	0x0	TXSNR/TRAS_MAX.
CTRL_REG_67	0x10c	W	0x20440000	VERSION/TXSR.
CTRL_REG_68	0x110	W	0x0	TINIT.
CTRL_REG_69	0x114	W	0x0	DFT_CTRL_REG.
CTRL_REG_70	0x118	W	0x0	DLL_CTRL_REG_0_0.
CTRL_REG_71	0x11c	W	0x0	DLL_CTRL_REG_0_1.
CTRL_REG_72	0x120	W	0x0	DLL_CTRL_REG_0_2.
CTRL_REG_73	0x124	W	0x0	DLL_CTRL_REG_0_3.
CTRL_REG_74	0x128	W	0x0	DLL_CTRL_REG_1_0.
CTRL_REG_75	0x12c	W	0x0	DLL_CTRL_REG_1_1.
CTRL_REG_76	0x130	W	0x0	DLL_CTRL_REG_1_2.
CTRL_REG_77	0x134	W	0x0	DLL_CTRL_REG_1_3.
CTRL_REG_78	0x138	W	0x0	DLL_OBS_REG_0_0.
CTRL_REG_79	0x13c	W	0x0	DLL_OBS_REG_0_1.
CTRL_REG_80	0x140	W	0x0	DLL_OBS_REG_0_2.
CTRL_REG_81	0x144	W	0x0	DLL_OBS_REG_0_3.
CTRL_REG_82	0x148	W	0x0	PAD_CTRL_REG_0.
CTRL_REG_83	0x14c	W	0x0	PHY_CTRL_REG_0_0.
CTRL_REG_84	0x150	W	0x0	PHY_CTRL_REG_0_1.
CTRL_REG_85	0x154	W	0x0	PHY_CTRL_REG_0_2.
CTRL_REG_86	0x158	W	0x0	PHY_CTRL_REG_0_3.
CTRL_REG_87	0x15c	W	0x0	PHY_CTRL_REG_1_0.
CTRL_REG_88	0x160	W	0x0	PHY_CTRL_REG_1_1.
CTRL_REG_89	0x164	W	0x0	PHY_CTRL_REG_1_2.
CTRL_REG_90	0x168	W	0x0	PHY_CTRL_REG_1_3.
CTRL_REG_91	0x16c	W	0x0	PHY_CTRL_REG_2.
CTRL_REG_92	0x170	W	0x0	PHY_OBS_REG_0_0.
CTRL_REG_93	0x174	W	0x0	PHY_OBS_REG_0_1.
CTRL_REG_94	0x178	W	0x0	PHY_OBS_REG_0_2.
CTRL_REG_95	0x17c	W	0x0	PHY_OBS_REG_0_3.
CTRL_REG_96	0x180	W	0x0	OUT_OF_RANGE_ADDR [31:0].
CTRL_REG_97	0x184	W	0x0	OUT_OF_RANGE_ADDR [33:32].
CTRL_REG_98	0x188	W	0x0	PORT_CMD_ERROR_ADDR [31:0].
CTRL_REG_99	0x18c	W	0x0	PORT_CMD_ERROR_ADDR [33:32].
CTRL_REG_100	0x190	W	0x0	DLL_OBS_REG_1_0 [31:0].
CTRL_REG_101	0x194	W	0x0	DLL_OBS_REG_1_0 [63:32].
CTRL_REG_102	0x198	W	0x0	DLL_OBS_REG_1_0 [95:64].
CTRL_REG_103	0x19c	W	0x0	DLL_OBS_REG_1_0 [127:96].
CTRL_REG_104	0x1a0	W	0x0	DLL_OBS_REG_1_0 [156:128].
CTRL_REG_105	0x1a4	W	0x0	DLL_OBS_REG_1_1 [31:0].
CTRL_REG_106	0x1a8	W	0x0	DLL_OBS_REG_1_1 [63:32].
CTRL_REG_107	0x1ac	W	0x0	DLL_OBS_REG_1_1 [95:64].



CTRL_REG_108	0x1b0	W	0x0	DLL_OBS_REG_1_1 [127:96].
CTRL_REG_109	0x1b4	W	0x0	DLL_OBS_REG_1_1 [156:128].
CTRL_REG_110	0x1b8	W	0x0	DLL_OBS_REG_1_2 [31:0].
CTRL_REG_111	0x1bc	W	0x0	DLL_OBS_REG_1_2 [63:32].
CTRL_REG_112	0x1c0	W	0x0	DLL_OBS_REG_1_2 [95:64].
CTRL_REG_113	0x1c4	W	0x0	DLL_OBS_REG_1_2 [127:96].
CTRL_REG_114	0x1c8	W	0x0	DLL_OBS_REG_1_2 [156:128].
CTRL_REG_115	0x1cc	W	0x0	DLL_OBS_REG_1_3 [31:0].
CTRL_REG_116	0x1d0	W	0x0	DLL_OBS_REG_1_3 [63:32].
CTRL_REG_117	0x1d4	W	0x0	DLL_OBS_REG_1_3 [95:64].
CTRL_REG_118	0x1d8	W	0x0	DLL_OBS_REG_1_3 [127:96].
CTRL_REG_119	0x1dc	W	0x0	DLL_OBS_REG_1_3 [156:128].
CTRL_REG_120	0x1e0	W	0x0	DLL_OBS_REG_2_0 [31:0].
CTRL_REG_121	0x1e4	W	0x0	DLL_OBS_REG_2_0 [63:32].
CTRL_REG_122	0x1e8	W	0x0	DLL_OBS_REG_2_0 [95:64].
CTRL_REG_123	0x1ec	W	0x0	DLL_OBS_REG_2_0 [127:96].
CTRL_REG_124	0x1f0	W	0x0	DLL_OBS_REG_2_0 [156:128].
CTRL_REG_125	0x1f4	W	0x0	DLL_OBS_REG_2_1 [31:0].
CTRL_REG_126	0x1f8	W	0x0	DLL_OBS_REG_2_1 [63:32].
CTRL_REG_127	0x1fc	W	0x0	DLL_OBS_REG_2_1 [95:64].
CTRL_REG_128	0x200	W	0x0	DLL_OBS_REG_2_1 [127:96].
CTRL_REG_129	0x204	W	0x0	DLL_OBS_REG_2_1 [156:128].
CTRL_REG_130	0x208	W	0x0	DLL_OBS_REG_2_2 [31:0].
CTRL_REG_131	0x20c	W	0x0	DLL_OBS_REG_2_2 [63:32].
CTRL_REG_132	0x210	W	0x0	DLL_OBS_REG_2_2 [95:64].
CTRL_REG_133	0x214	W	0x0	DLL_OBS_REG_2_2 [127:96].
CTRL_REG_134	0x218	W	0x0	DLL_OBS_REG_2_2 [156:128].
CTRL_REG_135	0x21c	W	0x0	DLL_OBS_REG_2_3 [31:0].
CTRL_REG_136	0x220	W	0x0	DLL_OBS_REG_2_3 [63:32].
CTRL_REG_137	0x224	W	0x0	DLL_OBS_REG_2_3 [95:64].
CTRL_REG_138	0x228	W	0x0	DLL_OBS_REG_2_3 [127:96].
CTRL_REG_139	0x22c	W	0x0	DLL_OBS_REG_2_3 [156:128].
CTRL_REG_140	0x230	W	0x0	DLL_OBS_REG_3_0 [31:0].
CTRL_REG_141	0x234	W	0x0	DLL_OBS_REG_3_0 [63:32].
CTRL_REG_142	0x238	W	0x0	DLL_OBS_REG_3_0 [95:64].
CTRL_REG_143	0x23c	W	0x0	DLL_OBS_REG_3_0 [127:96].
CTRL_REG_144	0x240	W	0x0	DLL_OBS_REG_3_0 [156:128].
CTRL_REG_145	0x244	W	0x0	DLL_OBS_REG_3_1 [31:0].
CTRL_REG_146	0x248	W	0x0	DLL_OBS_REG_3_1 [63:32].
CTRL_REG_147	0x24c	W	0x0	DLL_OBS_REG_3_1 [95:64].
CTRL_REG_148	0x250	W	0x0	DLL_OBS_REG_3_1 [127:96].
CTRL_REG_149	0x254	W	0x0	DLL_OBS_REG_3_1 [156:128].
CTRL_REG_150	0x258	W	0x0	DLL_OBS_REG_3_2 [31:0].
CTRL_REG_151	0x25c	W	0x0	DLL_OBS_REG_3_2 [63:32].
CTRL_REG_152	0x260	W	0x0	DLL_OBS_REG_3_2 [95:64].
CTRL_REG_153	0x264	W	0x0	DLL_OBS_REG_3_2 [127:96].
CTRL_REG_154	0x268	W	0x0	DLL_OBS_REG_3_2 [156:128].
CTRL_REG_155	0x26c	W	0x0	DLL_OBS_REG_3_3 [31:0].
CTRL_REG_156	0x270	W	0x0	DLL_OBS_REG_3_3 [63:32].
CTRL_REG_157	0x274	W	0x0	DLL_OBS_REG_3_3 [95:64].
CTRL_REG_158	0x278	W	0x0	DLL_OBS_REG_3_3 [127:96].
CTRL_REG_159	0x27c	W	0x0	DLL_OBS_REG_3_3 [156:128].

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access



### 30.3.2 Detail Register Description

#### CTRL\_REG\_00

Address: Operational Base + offset( 0x00)

Controller register 00

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Sets the mode for when the automatic refresh will occur. If the auto_refresh_mode parameter is set and a refresh is required to memory, the memory controller will delay this refresh until the end of the current transaction (if the transaction is fully contained inside a single page), or until the current transaction hits the end of the current page. <ul style="list-style-type: none"> <li>• 'b0 = Issue refresh on the next DRAM burst boundary, even if the current command is not complete.</li> <li>• 'b1 = Issue refresh on the next command boundary.</li> </ul>
23:17	-	-	Reserved.
16	W	0x0	Initiates an automatic refresh to the DRAM devices based on the setting of the auto_refresh_mode parameter. If there are any open banks when this parameter is set, the memory controller will automatically close these banks before issuing the auto-refresh command. This parameter will always read back as 0x0. <ul style="list-style-type: none"> <li>• 'b0 = No action</li> <li>• 'b1 = Issue refresh to the DRAM devices</li> </ul>
15:9	-	-	Reserved.
8	RW	0x0	Enables auto pre-charge mode for DRAM devices. Note: This parameter may not be modified after the start parameter has been asserted. <ul style="list-style-type: none"> <li>• 'b0 = Auto pre-charge mode disabled. Memory banks will stay open until another request require this bank, the maximum open time (tras_max) ha elapsed, or a refresh command closes all the banks.</li> <li>• 'b1 = Auto pre-charge mode enabled. All read and write transactions must be terminated by an auto pre-charge command. If a transaction consists of multiple read or write bursts, only the last command is issued with an auto pre-charge.</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	Enables address collision/data coherency detection as a condition when using the placement logic to fill the command queue. <ul style="list-style-type: none"> <li>• 'b0 = Disabled</li> <li>• 'b1 = Enabled</li> </ul>

#### CTRL\_REG\_01

Address: Operational Base + offset( 0x04)

Controller register 01

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.

24	RW	0x0	Port 3 behavior when bandwidth maximized.
23:17	-	-	Reserved.
16	RW	0x0	Port 2 behavior when bandwidth maximized.
15:9	-	-	Reserved.
8	RW	0x0	Port 1 behavior when bandwidth maximized.
7:1	-	-	Reserved.
0	RW	0x0	Port 0 behavior when bandwidth maximized.

**CTRL\_REG\_02**

Address: Operational Base + offset( 0x08)

Controller register 02

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Port 7 behavior when bandwidth maximized.
23:17	-	-	Reserved.
16	RW	0x0	Port 6 behavior when bandwidth maximized.
15:9	-	-	Reserved.
8	RW	0x0	Port 5 behavior when bandwidth maximized.
7:1	-	-	Reserved.
0	RW	0x0	Port 4 behavior when bandwidth maximized.

**CTRL\_REG\_03**

Address: Operational Base + offset( 0x0c)

Controller register 03

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Limit concurrent auto-precharge by waiting for the write recovery time, tWR, to complete after a write before issuing a read. <ul style="list-style-type: none"> <li>• 'b0 = Do not restrict concurrent auto-precharge.</li> <li>• 'b1 = Wait tWR after a write before issuing a read.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Enables concurrent auto pre-charge. Some DRAM devices do not allow one bank to be auto pre-charged while another bank is reading or writing. The JEDEC standard allows concurrent auto pre-charge. The user should set this parameter if the DRAM device supports this feature. <ul style="list-style-type: none"> <li>• 'b0 = Concurrent auto pre-charge disabled.</li> <li>• 'b1 = Concurrent auto pre-charge enabled.</li> </ul>
15:9	-	-	Reserved.
8	R	0x0	Register access to cke_status signal.
7:1	-	-	Reserved.
0	RW	0x0	Enables bank splitting as a condition when using the placement logic to fill the command queue. <ul style="list-style-type: none"> <li>• 'b0 = Disabled</li> <li>• 'b1 = Enabled</li> </ul>

**CTRL\_REG\_04**

Address: Operational Base + offset( 0x10)

Controller register 04

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Enables differential data strobe signals from the DRAM. This parameter is only relevant for the DDR

			PHY. <ul style="list-style-type: none"> <li>• 'b0 = Single-ended DQS signal from the DRAM.</li> <li>• 'b1 = Differential DQS signal from the DRAM.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Enable the DLL bypass feature of the controller.
15:9	-	-	Reserved.
8	R	0x0	Shows status of the dfi_init_complete signal from the PHY. For the DDR PHY, this signal indicates whether the DLL is locked or unlocked. <ul style="list-style-type: none"> <li>• 'b0 = The dfi_init_complete signal is not asserted</li> <li>• 'b1 = The dfi_init_complete signal is asserted.</li> </ul>
7:1	-	-	Reserved.
0	R	0x0	Configuration error on CS_VAL or CS_MSK settings.

**CTRL\_REG\_05**

Address: Operational Base + offset( 0x14)

Controller register 05

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	When this bit is set to 'b1, the memory initialization sequence may be interrupted and the memory may enter self-refresh mode. This is used to place the memory devices into self-refresh mode when a power loss is detected during the initialization process. <ul style="list-style-type: none"> <li>• 'b0 = Continue memory initialization.</li> <li>• 'b1 = Interrupt memory initialization and enter self-refresh mode.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Indicates that the memory devices for chip select 1 have eight banks. <ul style="list-style-type: none"> <li>• 'b0 = Memory devices have 4 banks.</li> <li>• 'b1 = Memory devices have 8 banks.</li> </ul>
15:9	-	-	Reserved.
8	RW	0x0	Indicates that the memory devices for chip select 0 have eight banks. <ul style="list-style-type: none"> <li>• 'b0 = Memory devices have 4 banks.</li> <li>• 'b1 = Memory devices have 8 banks.</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	Selects if the DQ output enables and DQS output enables will be driven active when the memory controller is in an idle state. <ul style="list-style-type: none"> <li>• 'b0 = Leave the output enables in their current state when idle.</li> <li>• 'b1 = Drive the idle_drive_enable signal high when idle.</li> </ul>

**CTRL\_REG\_06**

Address: Operational Base + offset( 0x18)

Controller register 06

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Enables interrupting of a combined write with auto pre-charge command with another read or write

			command to the same bank before the first write command is completed. <ul style="list-style-type: none"> <li>• 'b0 = Disable interrupting a combined write with auto pre-charge command with another read or write command to the same bank.</li> <li>• 'b1 = Enable interrupting a combined write with auto pre-charge command with another read or write command to the same bank.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Enables interrupting of a combined read with auto pre-charge command with another read command to the same bank before the first read command is completed. <ul style="list-style-type: none"> <li>• 'b0 = Disable interrupting the combined read with auto pre-charge command with another read command to the same bank.</li> <li>• 'b1 = Enable interrupting the combined read with auto pre-charge command with another read command to the same bank.</li> </ul>
15:9	-	-	Reserved.
8	RW	0x0	Enables interrupting an auto pre-charge command with another command for a different bank. If enabled, the current operation will be interrupted. However, the bank will be pre-charged as if the current operation were allowed to continue. <ul style="list-style-type: none"> <li>• 'b0 = Disable interrupting an auto pre-charge operation on a different bank.</li> <li>• 'b1 = Enable interrupting an auto pre-charge operation on a different bank.</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	Controls when the write commands are issued to the DRAM devices. <ul style="list-style-type: none"> <li>• 'b0 = The memory controller will issue a write command to the DRAM devices when it has received enough data for one DRAM burst. In this mode, write data can be sent in any cycle relative to the write command. This mode also allows for multi-word write command data to arrive in non-sequential cycles.</li> <li>• 'b1 = The memory controller will issue a write command to the DRAM devices after the first word of the write data is received by the memory controller. The first word can be sent at any time relative to the write command. In this mode, multi-word write command data must be available to the memory controller in sequential cycles.</li> </ul>

**CTRL\_REG\_07**

Address: Operational Base + offset( 0x1c)

Controller register 07

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	When this parameter is set to 'b1, the memory controller will complete processing of the current burst for the current transaction (if any), issue a pre-charge all command and then disable the clock

			enable signal to the DRAM devices. Any subsequent commands in the command queue will be suspended until this parameter is cleared to 'b0. <ul style="list-style-type: none"> <li>• 'b0 = Enable full power state</li> <li>• 'b1 = Disable the clock enable and power down the memory controller.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Enables using the placement logic to fill the command queue. <ul style="list-style-type: none"> <li>• 'b0 = Placement logic is disabled. The command queue is a straight FIFO.</li> <li>• 'b1 = Placement logic is enabled. The command queue will be filled according to the placement logic factors.</li> </ul>
15:9	-	-	Reserved.
8	RW	0x0	Enables the use of the non-DFI compliant alternative ODT internal signal odt_alt, which is externally viewed as the signal reserved0. This signal is only required if the user intends to use a CAS latency of 3 with ODT support. <ul style="list-style-type: none"> <li>• 'b0 = ODT support with CAS latency 3 is not supported.</li> <li>• 'b1 = ODT support with CAS latency 3 is supported but is not DFI compliant. This disables the interrupt bit for ODT-with-CAS3 and disables the OVL error.</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	Disables DRAM commands until DLL initialization is complete and tdll has expired. <ul style="list-style-type: none"> <li>• 'b0 = Issue only REF and PRE commands during DLL initialization of the DRAM devices. If PRE commands are issued before DLL initialization is complete, the command will be executed immediately, and then the DLL initialization will continue.</li> <li>• 'b1 = Do not issue any type of command during DLL initialization of the DRAM devices. If any other commands are issued during the initialization time, they will be held off until DLL initialization is complete.</li> </ul>

**CTRL\_REG\_08**

Address: Operational Base + offset( 0x20)

Controller register 08

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Enables registered DIMM operations to control the address and command pipeline of the memory controller. <ul style="list-style-type: none"> <li>• 'b0 = Normal operation</li> <li>• 'b1 = Enable registered DIMM operation</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Controls the width of the memory datapath. When enabled, the upper half of the memory buses (DQ, DQS and DM) are unused and relevant data only exists in the lower half of the buses. This parameter

			expands the memory controller for use with memory devices of the configured width or half of the configured width. <ul style="list-style-type: none"> <li>• 'b0 = Standard operation using full memory bus.</li> <li>• 'b1 = Memory datapath width is half of the maximum size. The upper half of the data_byte_disable bus will be driven to 'b1.</li> </ul>
15:9	-	-	Reserved.
8	RW	0x0	Allows controller to exit power-down mode by executing a self-refresh exit instead of the full memory initialization. This parameter provides a means to skip full initialization when the DRAM devices are in a known self-refresh state. <ul style="list-style-type: none"> <li>• 'b0 = Disabled</li> <li>• 'b1 = Enabled</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	Enables priority as a condition when using the placement logic to fill the command queue. <ul style="list-style-type: none"> <li>• 'b0 = Disabled</li> <li>• 'b1 = Enabled</li> </ul>

**CTRL\_REG\_09**

Address: Operational Base + offset( 0x24)

Controller register 09

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	When this parameter is set to 'b1, the DRAM device(s) will be placed in self-refresh mode. For this, the current burst for the current transaction (if any) will complete, all banks will be closed, the self-refresh command will be issued to the DRAM, and the clock enable signal will be de-asserted. The system will remain in self-refresh mode until this parameter is cleared to 'b0. The DRAM devices will return to normal operating mode after the self-refresh exit time (the txsr parameter) of the device and any DLL initialization time for the DRAM is reached. The memory controller will resume processing of the commands from the interruption point. This parameter will be updated with an assertion of the srefresh_enter pin, regardless of the behavior on the register interface. To disable self-refresh again after a srefresh_enter pin assertion, the user will need to clear the parameter to 'b0. <ul style="list-style-type: none"> <li>• 'b0 = Disable self-refresh mode</li> <li>• 'b1 = Initiate self-refresh of the DRAM devices.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	Enables read/write grouping as a condition when using the placement logic to fill the command queue. <ul style="list-style-type: none"> <li>• 'b0 = Disabled</li> <li>• 'b1 = Enabled</li> </ul>
15:9	-	-	Reserved.
8	RW	0x0	Enables automatic DLL resyncs after every refresh.
7:1	-	-	Reserved.

0	W	0x0	Initiate a DLL resync.
---	---	-----	------------------------

**CTRL\_REG\_10**

Address: Operational Base + offset( 0x28)

Controller register 10

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	Defines the tRAS lockout setting for the DRAM device. tRAS lockout allows the memory controller to execute auto pre-charge commands before the tras_min parameter has expired. <ul style="list-style-type: none"> <li>• 'b0 = tRAS lockout not supported by memory device.</li> <li>• 'b1 = tRAS lockout supported by memory device.</li> </ul>
23:17	-	-	Reserved.
16	RW	0x0	No meaning for this MC.
15:9	-	-	Reserved.
8	RW	0x0	Enables swapping of the active command for a new higher-priority command when using the placement logic. <ul style="list-style-type: none"> <li>• 'b0 = Disabled</li> <li>• 'b1 = Enabled</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	With this parameter is cleared to 'b0, the memory controller will not issue any commands to the DRAM devices or respond to any signal activity except for reading and writing parameters. Once this parameter is set to 'b1, the memory controller will respond to inputs from the ASIC. When set, the memory controller begins its initialization routine. Note: Until the initialization complete bit is set in the int_status parameter and the dfi_init_complete signal is asserted from the PHY, commands will not be accepted into the Databahn core command queue. <ul style="list-style-type: none"> <li>• 'b0 = Controller is not in active mode.</li> <li>• 'b1 = Initiate active mode for the memory controller.</li> </ul>

**CTRL\_REG\_11**

Address: Operational Base + offset( 0x2c)

Controller register 11

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Sets the relativity of the clock domains between AXI port 0 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>
23:17	-	-	Reserved.
16	W	0x0	Writes mode register information into the memory devices. The user should program the appropriate mrY_data_X parameters with valid information based on the memory type being used. All of the mode registers that are relevant for the memory type



			<p>specified in the dram_class parameter will be written on each write_modereg setting. This parameter will always read back as 'b0.</p> <p>The mode registers are automatically written at initialization of the memory controller. There is no need to initiate a mode register write after setting the start parameter in the memory controller unless some value in these registers needs to be changed after initialization.</p> <p>Note: This parameter may not be changed when the memory is in power-down mode (when the CKE input is de-asserted).</p>
15:9	-	-	Reserved.
8	RW	0x0	<p>Defines whether the memory controller can interrupt a write burst with a read command. Some memory devices do not allow this functionality. For LPDDR1 memory devices, consult the memory specification for the setting for this parameter. For DDR2 memory devices, this parameter must be cleared to 'b0.</p> <ul style="list-style-type: none"> <li>• 'b0 = The device does not support read commands interrupting write commands.</li> <li>• 'b1 = The device does support read commands interrupting write commands.</li> </ul>
7:1	-	-	Reserved.
0	RW	0x0	<p>Enables refresh commands. If command refresh mode is configured, then refresh commands will be automatically issued based on the tref parameter value and any refresh commands sent through the command interface or the register interface. Refreshes will still occur even if the DRAM devices have been placed in power down state by the assertion of the power_down parameter.</p> <ul style="list-style-type: none"> <li>• 'b0 = Refresh commands disabled.</li> <li>• 'b1 = Refresh commands enabled.</li> </ul>

**CTRL\_REG\_12**

Address: Operational Base + offset( 0x30)

Controller register 12

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Priority of read cmds from AXI port 1.
23:18	-	-	Reserved.
17:16	RW	0x0	<p>Sets the relativity of the clock domains between AXI port 1 and the memory controller core clock.</p> <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>
15:10	-	-	Reserved.
9:8	RW	0x0	Priority of write cmds from AXI port 0.
7:2	-	-	Reserved.
1:0	RW	0x0	Priority of read cmds from AXI port 0.

**CTRL\_REG\_13**

Address: Operational Base + offset( 0x34)

## Controller register 13

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Priority of write cmds from AXI port 2.
23:18	-	-	Reserved.
17:16	RW	0x0	Priority of read cmds from AXI port 2.
15:10	-	-	Reserved.
9:8	RW	0x0	Sets the relativity of the clock domains between AXI port 2 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>
7:2	-	-	Reserved.
1:0	RW	0x0	Priority of write cmds from AXI port 1.

## CTRL\_REG\_14

Address: Operational Base + offset( 0x38)

## Controller register 14

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Sets the relativity of the clock domains between AXI port 4 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>
23:18	-	-	Reserved.
17:16	RW	0x0	Priority of write cmds from AXI port 3.
15:10	-	-	Reserved.
9:8	RW	0x0	Priority of read cmds from AXI port 3.
7:2	-	-	Reserved.
1:0	RW	0x0	Sets the relativity of the clock domains between AXI port 3 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>

## CTRL\_REG\_15

Address: Operational Base + offset( 0x3c)

## Controller register 15

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Priority of read cmds from AXI port 5.
23:18	-	-	Reserved.
17:16	RW	0x0	Sets the relativity of the clock domains between AXI port 5 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>
15:10	-	-	Reserved.
9:8	RW	0x0	Priority of write cmds from AXI port 4.

7:2	-	-	Reserved.
1:0	RW	0x0	Priority of read cmds from AXI port 4.

**CTRL\_REG\_16**

Address: Operational Base + offset( 0x40)

Controller register 16

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Priority of write cmds from AXI port 6.
23:18	-	-	Reserved.
17:16	RW	0x0	Priority of read cmds from AXI port 6.
15:10	-	-	Reserved.
9:8	RW	0x0	Sets the relativity of the clock domains between AXI port 6 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>
7:2	-	-	Reserved.
1:0	RW	0x0	Priority of write cmds from AXI port 5.

**CTRL\_REG\_17**

Address: Operational Base + offset( 0x44)

Controller register 17

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Sets the mask that determines which chip select pins are active, with each bit representing a different chip select. The user address chip select field will be mapped into the active chip selects indicated by this parameter in ascending order from lowest to highest. This allows the memory controller to map the entire contiguous user address into any group of chip selects. Bit [0] of this parameter corresponds to chip select [0], bit [1] corresponds to chip select [1], etc. For heterogeneous memory systems, none of the chip selects may be disabled. As a result, the user is not allowed to clear any of the cs_map parameter bits to 'b0 in this mode.
23:18	-	-	Reserved.
17:16	RW	0x0	Priority of write cmds from AXI port 7.
15:10	-	-	Reserved.
9:8	RW	0x0	Priority of read cmds from AXI port 7.
7:2	-	-	Reserved.
1:0	RW	0x0	Sets the relativity of the clock domains between AXI port 7 and the memory controller core clock. <ul style="list-style-type: none"> <li>• 'b00 = Asynchronous</li> <li>• 'b01 = 2:1 Port:Core Pseudo-Synchronous</li> <li>• 'b10 = 1:2 Port:Core Pseudo-Synchronous</li> <li>• 'b11 = Synchronous</li> </ul>

**CTRL\_REG\_18**

Address: Operational Base + offset( 0x48)

Controller register 18

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	Sets up which (if any) chip(s) will have their ODT termination active while a read occurs on chip select 0. <ul style="list-style-type: none"> <li>• Bit [1] = CS1 will have active ODT termination when chip select 0 is performing a read.</li> <li>• Bit [0] = CS0 will have active ODT termination when chip select 0 is performing a read.</li> </ul>
23:18	-	-	Reserved.
17:16	R	0x2	Maximum number of chip selects available.
15:10	-	-	Reserved.
9:8	RW	0x0	Sets whether refreshes will occur while the memory controller is in any of the low power modes. There is one bit for each chip select. Note: This parameter is active low. <ul style="list-style-type: none"> <li>• 'b0 = Refreshes still occur</li> <li>• 'b1 = Refreshes do not occur</li> </ul>
7:2	-	-	Reserved.
1:0	RW	0x0	Sets value for the DFI output signal dfi_dram_clk_disable. Bit [0] controls CS0, Bit [1] controls CS1. For each bit: <ul style="list-style-type: none"> <li>• 'b0 = Memory clock/s should be active.</li> <li>• 'b1 = Memory clock/s should be disabled.</li> </ul>

**CTRL\_REG\_19**

Address: Operational Base + offset( 0x4c)

Controller register 19

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	Difference between number of addr pins available and number being used for chip select 0.
23:18	-	-	Reserved.
17:16	RW	0x0	Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select 1. <ul style="list-style-type: none"> <li>• Bit [1] = CS1 will have active ODT termination when chip select 1 is performing a write.</li> <li>• Bit [0] = CS0 will have active ODT termination when chip select 1 is performing a write.</li> </ul>
15:10	-	-	Reserved.
9:8	RW	0x0	Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select 0. <ul style="list-style-type: none"> <li>• Bit [1] = CS1 will have active ODT termination when chip select 0 is performing a write.</li> <li>• Bit [0] = CS0 will have active ODT termination when chip select 0 is performing a write.</li> </ul>
7:2	-	-	Reserved.
1:0	RW	0x0	Sets up which (if any) chip(s) will have their ODT termination active while a read occurs on chip select 1. <ul style="list-style-type: none"> <li>• Bit [1] = CS1 will have active ODT termination when chip select 1 is performing a read.</li> <li>• Bit [0] = CS0 will have active ODT termination</li> </ul>

			when chip select 1 is performing a read.
--	--	--	------------------------------------------

**CTRL\_REG\_20**

Address: Operational Base + offset( 0x50)

Controller register 20

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	Sets the number of additional cycles of delay to include in the CKE signal cke_status for status reporting. The default delay is 0 cycles.
23:19	-	-	Reserved.
18:16	RW	0x0	Sets the CAS (Column Address Strobe) latency encoding that the memory uses. The binary value programmed into this parameter is dependent on the memory device, since the same caslat value may have different meanings to different memories. This will be programmed into the DRAM devices at initialization. The CAS encoding will be specified in the DRAM spec sheet, and should correspond to the caslat_lin parameter.
15:11	-	-	Reserved.
10:8	RW	0x0	Sets the command queue fullness that determines if ports will be allowed to overflow. This parameter is used in conjunction with the axiY_bdw_ovflow parameters.
7:3	-	-	Reserved.
2:0	RW	0x0	Difference between number of addr pins available and number being used for chip select 1.

**CTRL\_REG\_21**

Address: Operational Base + offset( 0x54)

Controller register 21

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	Defines quantity of data that will be considered full for the command queue. When this value is reached, the q_almost_full signal will be driven to the user interface.
23:19	-	-	Reserved.
18:16	R	0x0	Type of error and access type that caused the PORT data error.
15:11	-	-	Reserved.
10:8	RW	0x0	Difference between number of column pins available and number being used for chip select 1.
7:3	-	-	Reserved.
2:0	RW	0x0	Difference between number of column pins available and number being used for chip select 0.

**CTRL\_REG\_22**

Address: Operational Base + offset( 0x58)

Controller register 22

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	Defines the number of additional clocks of delay to insert from a read command to a write command to the same chip select.

			The minimum allowable programming of this parameter is 1 cycle.
23:19	-	-	Reserved.
18:16	RW	0x0	Defines the number of additional clocks of delay to insert from a read command to one chip select to a write command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.
15:11	-	-	Reserved.
10:8	RW	0x0	Additional delay to insert between reads and reads to the same chip select.
7:3	-	-	Reserved.
2:0	RW	0x0	Defines the number of additional clocks of delay to insert from a read command to one chip select to a read command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.

**CTRL\_REG\_23**

Address: Operational Base + offset( 0x5c)

Controller register 23

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	DRAM TRRD parameter, Defines the DRAM activate to activate delay for different banks, in cycles.
23:19	-	-	Reserved.
18:16	RW	0x0	Holds the DFI tdram_clk_disable timing parameter. This parameter should be programmed with the number of cycles that the PHY requires to disable the clock after the dfi_dram_clk_disable signal is asserted. The user should determine these values based on their pad types. For example, if the pad disable time and the clock period are both 2.5 ns, then the tdfi_dram_clk_disable parameter would be set to 0x1. If the dfi_dram_clk_disable signal is being used to turn off the pad output enables, this parameter should be programmed with the amount of time expected for the pad to enable itself again.
15:11	-	-	Reserved.
10:8	RW	0x0	Minimum CKE pulse width.
7:3	-	-	Reserved.
2:0	RW	0x0	Defines the burst interrupt interval. This parameter is only relevant if the burst has not completed. This value is loaded into a parameter when a burst is issued and another command may only interrupt the current burst when this counter value hits 0. If the counter value hits 0 and the burst has not completed, the counter will be reset with the tbst_int_interval value. If a command is in progress and the burst has not completed, another command may only be issued on cycles after the parameter tccd value cycles have elapsed since the last CAS command and this counter value hits 0.

**CTRL\_REG\_24**



Address: Operational Base + offset( 0x60)

Controller register 24

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	Defines the number of additional clocks of delay to insert from a write command to one chip select to a write command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.
23:19	-	-	Reserved.
18:16	RW	0x0	Defines the number of additional clocks of delay to insert from a write command to a read command to the same chip select. The minimum allowable programming of this parameter is 1 cycle.
15:11	-	-	Reserved.
10:8	RW	0x0	Defines the number of additional clocks of delay to insert from a write command to one chip select to a read command to a different chip select. The minimum allowable programming of this parameter is 1 cycle.
7:3	-	-	Reserved.
2:0	RW	0x0	DRAM TRTP (read to pre-charge time) parameter in cycles.

**CTRL\_REG\_25**

Address: Operational Base + offset( 0x64)

Controller register 25

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Additional delay to insert between same transaction types to different chip selects to meet ODT timing requirements.
23:20	-	-	Reserved.
19:16	RW	0x0	Additional delay to insert between different transaction types to the same chip select to meet ODT timing requirements.
15:12	-	-	Reserved.
11:8	RW	0x0	Additional delay to insert between different transaction types to different chip selects to meet ODT timing requirements.
7:3	-	-	Reserved.
2:0	RW	0x0	Additional delay to insert between writes and writes to the same chip select.

**CTRL\_REG\_26**

Address: Operational Base + offset( 0x68)

Controller register 26

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Defines the number of beats of data to return to AHB-bridged port 3 for an INCR READ AHB command. The AHB logic will send (ahbX_rdlen +1) beats of read data to the AHB bus.
23:20	-	-	Reserved.
19:16	RW	0x0	Defines the number of beats of data to expect from



			AHB-bridged port 2 for an INCR WRITE AHB command. The AHB logic will wait for (ahbX_wrlen + 1) beats of write data from the AHB bus.
15:12	-	-	Reserved.
11:8	RW	0x0	Defines the number of beats of data to return to AHB-bridged port 2 for an INCR READ AHB command. The AHB logic will send (ahbX_rdlen + 1) beats of read data to the AHB bus.
7:4	-	-	Reserved.
3:0	RW	0x0	Holds the initial value of the master aging-rate counter. When using the placement logic to fill the command queue, the command aging counters will be decremented one each time the master aging-rate counter counts down age_count cycles.

**CTRL\_REG\_27**

Address: Operational Base + offset( 0x6c)

Controller register 27

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Defines the number of beats of data to return to AHB-bridged port 5 for an INCR READ AHB command. The AHB logic will send (ahbX_rdlen + 1) beats of read data to the AHB bus.
23:20	-	-	Reserved.
19:16	RW	0x0	Defines the number of beats of data to expect from AHB-bridged port 4 for an INCR WRITE AHB command. The AHB logic will wait for (ahbX_wrlen + 1) beats of write data from the AHB bus.
15:12	-	-	Reserved.
11:8	RW	0x0	Defines the number of beats of data to return to AHB-bridged port 4 for an INCR READ AHB command. The AHB logic will send (ahbX_rdlen + 1) beats of read data to the AHB bus.
7:4	-	-	Reserved.
3:0	RW	0x0	Defines the number of beats of data to expect from AHB-bridged port 3 for an INCR WRITE AHB command. The AHB logic will wait for (ahbX_wrlen + 1) beats of write data from the AHB bus.

**CTRL\_REG\_28**

Address: Operational Base + offset( 0x70)

Controller register 28

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Defines the number of beats of data to return to AHB-bridged port 7 for an INCR READ AHB command. The AHB logic will send (ahbX_rdlen + 1) beats of read data to the AHB bus.
23:20	-	-	Reserved.
19:16	RW	0x0	Defines the number of beats of data to expect from AHB-bridged port 6 for an INCR WRITE AHB command. The AHB logic will wait for (ahbX_wrlen + 1) beats of write data from the AHB bus.
15:12	-	-	Reserved.
11:8	RW	0x0	Defines the number of beats of data to return to

			AHB-bridged port 6 for an INCR READ AHB command. The AHB logic will send (ahbX_rdlen + 1) beats of read data to the AHB bus.
7:4	-	-	Reserved.
3:0	RW	0x0	Defines the number of beats of data to expect from AHB-bridged port 5 for an INCR WRITE AHB command. The AHB logic will wait for (ahbX_wrlen + 1) beats of write data from the AHB bus.

**CTRL\_REG\_29**

Address: Operational Base + offset( 0x74)

Controller register 29

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Sets the CAS latency linear value in 1/2 cycle increments. This sets an internal adjustment for the delay from when the read command is sent from the memory controller to when data will be received back. The window of time in which the data is captured is a fixed length. The caslat_lin parameter adjusts the start of this data capture window. Not all linear values will be supported for the memory devices being used. Refer to the specification for the memory devices being used. Note: For optimal synthesis behavior, the ODT path for a CAS latency of three is clocked at a 200 MHz clock regardless of configured maximum speed. <ul style="list-style-type: none"> <li>• 'b0000 - 'b0001 = Reserved</li> <li>• 'b0010 = 1 cycle</li> <li>• 'b0011 = 1.5 cycles</li> <li>• 'b0100 = 2 cycles</li> <li>• 'b0101 = 2.5 cycles</li> <li>• 'b0110 = 3 cycles</li> <li>• 'b0111 = 3.5 cycles</li> <li>• 'b1000 = 4 cycles</li> <li>• 'b1001 = 4.5 cycles</li> <li>• 'b1010 = 5 cycles</li> <li>• 'b1011 = 5.5 cycles</li> <li>• 'b1100 = 6 cycles</li> <li>• 'b1101 = 6.5 cycles</li> <li>• 'b1110 = 7 cycles</li> <li>• 'b1111 = 7.5 cycles</li> </ul>
23:20	-	-	Reserved.
19:16	RW	0x0	Location of auto pre-charge bit in DRAM address for chip select 1.
15:12	-	-	Reserved.
11:8	RW	0x0	Location of auto pre-charge bit in DRAM address for chip select 0.
7:4	-	-	Reserved.
3:0	RW	0x0	Defines the number of beats of data to expect from AHB-bridged port 7 for an INCR WRITE AHB command. The AHB logic will wait for (ahbX_wrlen + 1) beats of write data from the AHB bus.

**CTRL\_REG\_30**

Address: Operational Base + offset( 0x78)

Controller register 30

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Holds the initial value of the command aging counters associated with each command in the command queue. When using the placement logic to fill the command queue, the command aging counters decrement one each time the master aging-rate counter counts down the number of cycles in the age_count parameter.
23:20	-	-	Reserved.
19:16	RW	0x0	Sets the number of cycles to hold the clock stable before exiting self-refresh mode. The clock will run for a minimum of cksrx cycles before CKE rises.
15:12	-	-	Reserved.
11:8	RW	0x0	Sets the number of cycles to hold the clock stable after entering self-refresh mode. The clock will run for a minimum of cksre cycles after CKE falls.
7:4	-	-	Reserved.
3:0	RW	0x0	Adjusts the data capture gate open time by 1/2 cycle increments. This parameter is programmed differently than caslat_lin when there are fixed offsets in the flight path between the memories and the memory controller for clock gating. When caslat_lin_gate is a larger value than caslat_lin, the data capture window will become shorter. A caslat_lin_gate value smaller than caslat_lin may have no effect on the data capture window, depending on the fixed offsets in the ASIC and the board. Note: For optimal synthesis behavior, the ODT path for a CAS latency of three is clocked at a 200 MHz clock regardless of configured maximum speed. <ul style="list-style-type: none"> <li>• 'b0000 - 'b0001 = Reserved</li> <li>• 'b0010 = 1 cycle</li> <li>• 'b0011 = 1.5 cycles</li> <li>• 'b0100 = 2 cycles</li> <li>• 'b0101 = 2.5 cycles</li> <li>• 'b0110 = 3 cycles</li> <li>• 'b0111 = 3.5 cycles</li> <li>• 'b1000 = 4 cycles</li> <li>• 'b1001 = 4.5 cycles</li> <li>• 'b1010 = 5 cycles</li> <li>• 'b1011 = 5.5 cycles</li> <li>• 'b1100 = 6 cycles</li> <li>• 'b1101 = 6.5 cycles</li> <li>• 'b1110 = 7 cycles</li> <li>• 'b1111 = 7.5 cycles</li> </ul>

**CTRL\_REG\_31**

Address: Operational Base + offset( 0x7c)

Controller register 31

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	R	0xf	Maximum width of memory address bus.

23:20	-	-	Reserved.
19:16	R	0xd	Maximum width of column address in DRAMs.
15:12	-	-	Reserved.
11:8	RW	0x0	Number of auto-refresh cmds to execute during DRAM initialization.
7:4	-	-	Reserved.
3:0	RW	0x0	<p>Selects the mode of operation for the memory controller.</p> <ul style="list-style-type: none"> <li>• 'b0000 = DDR1</li> <li>• 'b0001 = DDR1 with Mobile (LPDDR1)</li> <li>• 'b0100 = DDR2</li> <li>• All other settings reserved</li> </ul>

**CTRL\_REG\_32**

Address: Operational Base + offset( 0x80)

Controller register 32

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Holds the DFI tctrl_delay timing parameter. This parameter should be programmed with the number of cycles that the PHY requires to send a power-down or self-refresh command to the DRAM devices.
23:20	-	-	Reserved.
19:16	R	0x4	Holds the DFI tCTRLUPD_MIN timing parameter.
15:12	-	-	Reserved.
11:8	RW	0x0	Adjusts the relative timing between DFI read commands and the dfi_rddata_en signal to conform to PHY timing requirements. When this parameter is programmed to 0x0, dfi_rddata_en will assert one cycle after the dfi_address. The sum of the tdfi_rddata_en_base parameter and this parameter must be at least 0x4. These two parameters work together to set the actual PHY read latency (tdfi_rddata_en). This parameter only affects the DFI.
7:4	-	-	Reserved.
3:0	R	0x0	<p>Defines the type of error and the access type that caused the port command error condition. If multiple bits are set to 'b1, then multiple errors were found. This parameter is read-only.</p> <ul style="list-style-type: none"> <li>• Bit [3] = Narrow transfer requested for a requestor Y whose axiY_en_size_lt_width_instr parameter is clear.</li> <li>• Bit [2:0] = Reserved</li> </ul>

**CTRL\_REG\_33**

Address: Operational Base + offset( 0x84)

Controller register 33

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Used to adjust the tdfi_phy_wrlat parameter for the difference between the PHY's command path delay and data path delay. The sum of the wrlat_adj parameter and this parameter must be at least 0x3. These two parameters work together to set the actual PHY write latency (tdfi_phy_wrlat). This parameter only affects the DFI.

			<ul style="list-style-type: none"> <li>• 'b0000 = Command path delay is 2 cycles shorter than the data path delay.</li> <li>• 'b0001 = Command path delay is 1 cycle shorter than the data path delay.</li> <li>• 'b0010 = Command path delay and data path delay are equivalent.</li> <li>• 'b0011 = Command path delay is 1 cycle longer than the data path delay.</li> <li>• 'b0100 = Command path delay is 2 cycles longer than the data path delay.</li> <li>• 'b0101 = Command path delay is 3 cycles longer than the data path delay.</li> <li>• 'b0110, etc.</li> </ul>
23:20	-	-	Reserved.
19:16	R	0x0	Holds the calculated value of the tphy_wrlat timing parameter and is used to adjust the dfi_wrdata_en signal timing. $\text{tdfi\_phy\_wrlat} = \text{tdfi\_phy\_wrlat\_base} + \text{wrlat\_adj} + \text{reg\_dimm\_enable} - \text{WRLAT\_WIDTH}'\text{h}3$ Note: Values of $(\text{tdfi\_phy\_wrlat\_base} + \text{wrlat\_adj}) < 3$ are not supported.
15:12	-	-	Reserved.
11:8	RW	0x0	Holds the tPHY_RDLAT timing parameter.
7:4	-	-	Reserved.
3:0	RW	0x0	Holds the DFI tdram_clk_enable timing parameter. This parameter is used to indicate the number of cycles that the PHY requires to respond to a de-assertion of the dfi_dram_clk_disable signal. The user should determine these values based on their pad types. For example, if the pad enable time and the clock period are both 2.5 ns, then the tdfi_dram_clk_enable parameter would be set to 0x1.

**CTRL\_REG\_34**

Address: Operational Base + offset( 0x88)

Controller register 34

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	DRAM TRP All Bank parameter in cycles for chip select 0.
23:20	-	-	Reserved.
19:16	RW	0x0	DRAM TRP(pre-charge command time) parameter in cycles.
15:12	-	-	Reserved.
11:8	RW	0x0	Used to adjust the tdfi_rddata_en parameter if the PHY requires greater delay from read command to read data enable. The CAS latency is defined in the caslat parameter. The sum of the rdlat_adj parameter and this parameter must be at least 0x4. These two parameters work together to set the actual PHY read latency (tdfi_rddata_en). A value of 2 sets the dfi_rddata_en signal to the value "CAS Latency-1." Values less than 2 pull the dfi_rddata_en signal earlier in time. Values greater than 2 delay the dfi_rddata_en signal relative to the read command.

7:4	-	-	Reserved.
3:0	R	0x0	Holds the calculated value of the trddata_en timing parameter. $\text{tdfi\_rddata\_en} = \text{tdfi\_rddata\_en\_base} + \text{rdlat\_adj} + \text{reg\_dimm\_enable} - \text{RDLAT\_WIDTH}'\text{h3}$ Note: Values of $(\text{tdfi\_rddata\_en\_base} + \text{rdlat\_adj}) < 4$ are not supported.

**CTRL\_REG\_35**

Address: Operational Base + offset( 0x8c)

Controller register 35

bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	Adjusts the relative timing between DFI write commands and the dfi_wrdata_en signal to conform to PHY timing requirements. When this parameter is programmed to 0x0, dfi_wrdata_en will assert on the same cycle as the dfi_address. The sum of the tdfi_phy_wrlat_base parameter and this parameter must be at least 0x3. These two parameters work together to set the actual PHY write latency (tdfi_phy_wrlat). This parameter only affects the DFI.
23:20	-	-	Reserved.
19:16	RW	0x0	Defines the write latency from when the write command is issued to the time the write data is presented to the DRAM devices, in cycles.
15:12	-	-	Reserved.
11:8	RW	0x0	Sets the number of cycles needed to switch from a write to a read operation, as dictated by the DDR SDRAM specification.
7:4	-	-	Reserved.
3:0	RW	0x0	DRAM TRP All Bank parametein cycles for chip select 1.

**CTRL\_REG\_36**

Address: Operational Base + offset( 0x90)

Controller register 36

bit	Attr	Reset Value	Description
31:13	-	-	Reserved.
12:8	RW	0x0	Enables the individual low power modes of the device. <ul style="list-style-type: none"> <li>• Bit [4] = Controls memory power-down mode (Mode 1).</li> <li>• Bit [3] = Controls memory power-down with memory clock gating mode (Mode 2).</li> <li>• Bit [2] = Controls memory self-refresh mode (Mode 3).</li> <li>• Bit [1] = Controls memory self-refresh with memory clock gating mode (Mode 4).</li> <li>• Bit [0] = Controls memory self-refresh with memory and controller clock gating mode (Mode 5).</li> </ul> For all bits: 'b0 = Disabled 'b1 = Enabled
7:5	-	-	Reserved.
4:0	RW	0x0	Enables automatic entry into the low power modes of the memory controller. <ul style="list-style-type: none"> <li>• Bit [4] = Controls memory power-down mode</li> </ul>



			<p>(Mode 1).</p> <ul style="list-style-type: none"> <li>• Bit [3] = Controls memory power-down with memory clock gating mode (Mode 2).</li> <li>• Bit [2] = Controls memory self-refresh mode (Mode 3).</li> <li>• Bit [1] = Controls memory self-refresh with memory clock gating mode (Mode 4).</li> <li>• Bit [0] = Controls memory self-refresh with memory and controller clock gating mode (Mode 5).</li> </ul> <p>Note: It is not possible to enter Mode 5 manually. Setting bit [0] of lowpower_control with bit [0] of this parameter cleared will not result in any change.</p> <p>Note: The user should not use both automatic and manual modes for the various low power modes. All modes should be entered automatically or all entered manually.</p> <p>For all bits:</p> <p>'b0 = Automatic entry into this mode is disabled. The user may enter this mode manually by setting the associated lowpower_control bit.</p> <p>'b1 = Automatic entry into this mode is enabled. The mode will be entered automatically when the proper counters expire, and only if the associated lowpower_control bit is set.</p>
--	--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_37**

Address: Operational Base + offset( 0x94)

Controller register 37

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:24	RW	0x0	Defines the minimum number of cycles required between two mode register write commands. This is the time required to complete the write operation to the mode register.
23:21	-	-	Reserved.
20:16	RW	0x0	Defines the auto pre-charge write recovery time when auto pre-charge is enabled (the ap parameter is set to 'b1), in cycles. This is defined internally as tRP (pre-charge time) + auto pre-charge write recovery time. Not all memories use this parameter. If tDAL is defined in the memory specification, then program this parameter to the specified value. If the memory does not specify a tDAL time, then program this parameter to tWR + tRP. DO NOT program this parameter with a value of 0x0 or the memory controller will not function properly when auto pre-charge is enabled.
15:13	-	-	Reserved.
12:8	RW	0x0	Defines the minimum number of cycles that CKE must be held low during self-refresh. pulse width, in cycles. If the memory specification does not define a tckesr, then the tckesr parameter should be programmed with the tcke value.
7:5	-	-	Reserved.
4:0	RW	0x0	DRAM CAS-to-CAS parameter in cycles.



**CTRL\_REG\_38**

Address: Operational Base + offset( 0x98)

Controller register 38

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:24	RW	0x0	DRAM TRC parameter, Defines the DRAM period between active commands for the same bank, in cycles.
23:22	-	-	Reserved.
21:16	RW	0x0	DRAM TFAW parameter in cycles.
15:14	-	-	Reserved.
13:8	R	0x0	Type of cmd that caused an Out-of-Range interrupt.
7:5	-	-	Reserved.
4:0	RW	0x0	DRAM TWR(write recovery time) parameter in cycles.

**CTRL\_REG\_39**

Address: Operational Base + offset( 0x9c)

Controller register 39

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	R	0x0	Current bandwidth usage percentage for port 1.
23	-	-	Reserved.
22:16	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 1. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.
15	-	-	Reserved.
14:8	R	0x0	Current bandwidth usage percentage for port 0.
7	-	-	Reserved.
6:0	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 0. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.

**CTRL\_REG\_40**

Address: Operational Base + offset( 0xa0)

Controller register 40

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	R	0x0	Current bandwidth usage percentage for port 3.
23	-	-	Reserved.
22:16	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 3. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.
15	-	-	Reserved.
14:8	R	0x0	Current bandwidth usage percentage for port 2.
7	-	-	Reserved.
6:0	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 2. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.

**CTRL\_REG\_41**

Address: Operational Base + offset( 0xa4)

Controller register 41

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	R	0x0	Current bandwidth usage percentage for port 5.
23	-	-	Reserved.
22:16	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 5. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.
15	-	-	Reserved.
14:8	R	0x0	Current bandwidth usage percentage for port 4.
7	-	-	Reserved.
6:0	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 4. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.

**CTRL\_REG\_42**

Address: Operational Base + offset( 0xa8)

Controller register 42

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	R	0x0	Current bandwidth usage percentage for port 7.
23	-	-	Reserved.
22:16	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 7. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.
15	-	-	Reserved.
14:8	R	0x0	Current bandwidth usage percentage for port 6.
7	-	-	Reserved.
6:0	RW	0x0	Sets the maximum bandwidth allocation percentage for AXI port 6. The percentage must be specified as a hex value (0x01 - 0x64) representing a decimal percentage value from 1 - 100.

**CTRL\_REG\_43**

Address: Operational Base + offset( 0xac)

Controller register 43

bit	Attr	Reset Value	Description
31:24	RW	0x0	Number of clock cycles after MRS command and before any other command.
23:16	RW	0x0	Minimum number of cycles after setting master delay in DLL until reset is released.
15:8	R	0x0	Defines the actual number of delay elements used to capture one full clock cycle. This parameter is automatically updated every time a refresh operation is performed.
7	-	-	Reserved.
6:0	R	0x0	Length of cmd that caused an Out-of-Range interrupt.

**CTRL\_REG\_44**

Address: Operational Base + offset( 0xb0)

Controller register 44

bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:16	RW	0x0	DRAM TRFC(refresh command time) parameter in

			cycles.
15:8	RW	0x0	DRAM TRCD(RAS to CAS delay) parameter in cycles.
7:0	RW	0x0	DRAM TRAS_MIN(minimum row activate time) parameter in cycles.

**CTRL\_REG\_45**

Address: Operational Base + offset( 0xb4)

Controller register 45

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	RW	0x0	Active-high mask bits that control the value of the memory controller_int signal on the ASIC interface. Unless the user has suppressed interrupt reporting (by setting the most significant bit of this parameter to 'b1), all lower bits of the int_mask parameter will be inverted and logically AND'ed with the corresponding bits of the int_status parameter and the result is reported on the controller_int signal.
15:10	-	-	Reserved.
9:0	W	0x0	Controls the clearing of the int_status parameter. If any of the int_ack bits are set to 'b1, the corresponding bit in the int_status parameter will be cleared to 'b0. Any int_ack bits cleared to 'b0 will not alter the corresponding bit in the int_status parameter. This parameter will always read back as 0x0.

**CTRL\_REG\_46**

Address: Operational Base + offset( 0xb8)

Controller register 46

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	R	0x0	Source ID of cmd that caused an Out-of-Range interrupt.
15:11	-	-	Reserved.
10:0	R	0x0	Shows the status of all possible interrupts generated by the memory controller. The MSB is the result of a logical OR of all the lower bits. The int_status bits correspond to these interrupts: <ul style="list-style-type: none"> <li>• Bit [10] = Logical OR of all lower bits.</li> <li>• Bit [9] = User-initiated DLL resync is finished.</li> <li>• Bit [8] = DLL lock state change condition detected. (i.e. lock to unlock or unlock to lock)</li> <li>• Bit [7] = Indicates that a read DQS gate error occurred.</li> <li>• Bit [6] = ODT enabled and CAS Latency 3 programmed error detected. This is an unsupported programming option.</li> <li>• Bit [5] = Both DDR2 and Mobile modes have been enabled.</li> <li>• Bit [4] = DRAM initialization complete.</li> <li>• Bit [3] = Error was found with command data channel in a port.</li> <li>• Bit [2] = Error was found with command channel in a port.</li> <li>• Bit [1] = Multiple accesses outside the defined</li> </ul>

			PHYSICAL memory space detected. • Bit [0] = A single access outside the defined PHYSICAL memory space detected.
--	--	--	--------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_47**

Address: Operational Base + offset( 0xbc)

Controller register 47

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	R	0x0	Source ID of cmd that caused the PORT data error.
15:11	-	-	Reserved.
10:0	R	0x0	Source ID of cmd that caused the PORT cmd error.

**CTRL\_REG\_48**

Address: Operational Base + offset( 0xc0)

Controller register 48

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	Holds the DFI tPHYUPD_RESP timing parameter.
15:14	-	-	Reserved.
13:0	RW	0x0	Holds the DFI tCTRLUPD_MAX timing parameter.

**CTRL\_REG\_49**

Address: Operational Base + offset( 0xc4)

Controller register 49

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	Holds the DFI tPHYUPD_TYPE1 timing parameter.
15:14	-	-	Reserved.
13:0	RW	0x0	Holds the DFI tPHYUPD_TYPE0 timing parameter.

**CTRL\_REG\_50**

Address: Operational Base + offset( 0xc8)

Controller register 50

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	Holds the DFI tPHYUPD_TYPE3 timing parameter.
15:14	-	-	Reserved.
13:0	RW	0x0	Holds the DFI tPHYUPD_TYPE2 timing parameter.

**CTRL\_REG\_51**

Address: Operational Base + offset( 0xcc)

Controller register 51

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:16	RW	0x0	Holds the memory mode register 0 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register. This parameter correlates to the memory mode register (MR). The use of this parameter varies based on the memory type connected to this memory controller: • For DDR2 memories: The memory controller does not support interleaving and therefore the A3 bit should be cleared to '0'. In addition, the DLL Reset bit

			<p>(A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the memory controller only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010.</p> <ul style="list-style-type: none"> <li>For LPDDR1 memories: The memory controller does not support interleaving and therefore the A3 bit should be cleared to 'b0. Also, the memory controller only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. This data will be programmed into the memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>
15:14	-	-	Reserved.
13:0	RW	0x0	<p>Defines the DRAM cycles between refresh commands. This parameter sets the average interval between refreshes. The actual interval may vary by up to 8 cycles from refresh to refresh, depending on other activities that are occurring in the controller at the time.</p> <p>Over an infinite time period, the average interval will be equal to the number of clocks set by this parameter; however, if the user sets this parameter to be exactly equal to the specification value tREFi, then local variations might mean that the memory tREF value may be violated by a very small amount. The amount of this violation would be <math>8tCK/tREF</math>.</p> <p>For example, for a 400MHz memory and tREF=64ms, the amount of the violation would be <math>2.5ns/64ms = 0.000004\%</math>, but this violation may still be detected by some memory models.</p> <p>If the warning from the memory model is concerning, the user should program the tref parameter to the value <math>(tREFi/tCK)-8</math>, which will result in slightly lower overall bandwidth and slightly higher memory power usage.</p>

**CTRL\_REG\_52**

Address: Operational Base + offset( 0xd0)

Controller register 52

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:16	RW	0x0	<p>Holds the memory mode register 1 data for chip Select0 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The memory controller does not support additive latency and therefore the A5:A3 bits should be cleared to 'b000.</li> <li>For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg</li> </ul>

			parameter is set to 'b1.
15	-	-	Reserved.
14:0	RW	0x0	<p>Holds the memory mode register 0 data for chip select 1 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>This parameter correlates to the memory mode register (MR). The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>For DDR2 memories: The memory controller does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the memory controller only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010.</li> <li>For LPDDR1 memories: The memory controller does not support interleaving and therefore the A3 bit should be cleared to 'b0. Also, the memory controller only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. This data will be programmed into the memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>

**CTRL\_REG\_53**

Address: Operational Base + offset( 0xd4)

Controller register 53

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:16	RW	0x0	<p>Holds the memory mode register 2 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2).</li> <li>For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>
15	-	-	Reserved.
14:0	RW	0x0	<p>Holds the memory mode register 1 data for chip Select 1 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The memory controller does not support additive latency and therefore the A5:A3 bits should be cleared to 'b000.</li> </ul>



			<ul style="list-style-type: none"> <li>For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>
--	--	--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_54**

Address: Operational Base + offset( 0xd8)

Controller register 54

bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:16	RW	0x0	<p>Holds the memory mode register 3 data for chip select 0 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3).</li> <li>For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>
15	-	-	Reserved.
14:0	RW	0x0	<p>Holds the memory mode register 2 data for chip select 1 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2).</li> <li>For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>

**CTRL\_REG\_55**

Address: Operational Base + offset( 0xdc)

Controller register 55

bit	Attr	Reset Value	Description
31:16	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 0 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 0 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 0's axi0_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the</p>



			<p>axi0_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 0 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 0 can issue size equal to width or size less than width instructions.</li> </ul>
15	-	-	Reserved.
14:0	RW	0x0	<p>Holds the memory mode register 3 data for chip select 1 written during memory initialization. Consult the memory specification for the fields of this mode register.</p> <p>The use of this parameter varies based on the memory type connected to this memory controller:</p> <ul style="list-style-type: none"> <li>• For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3).</li> <li>• For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.</li> </ul>

**CTRL\_REG\_56**

Address: Operational Base + offset( 0xe0)

Controller register 56

bit	Attr	Reset Value	Description
31:16	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 2 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 2 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 2's axi2_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi2_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 2 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 2 can issue size equal to</li> </ul>

			width or size less than width instructions.
15:0	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 1 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 1 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 1's axi1_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi1_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 1 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 1 can issue size equal to width or size less than width instructions.</li> </ul>

**CTRL\_REG\_57**

Address: Operational Base + offset( 0xe4)

Controller register 57

bit	Attr	Reset Value	Description
31:16	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 4 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 4 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 4's axi4_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi4_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 4 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 4 can issue size equal to</li> </ul>

			width or size less than width instructions.
15:0	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 3 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 3 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 3's axi3_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi3_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 3 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 3 can issue size equal to width or size less than width instructions.</li> </ul>

**CTRL\_REG\_58**

Address: Operational Base + offset( 0xe8)

Controller register 58

bit	Attr	Reset Value	Description
31:16	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 6 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 6 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 6's axi6_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi6_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 6 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 6 can issue size equal to</li> </ul>

			width or size less than width instructions.
15:0	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 5 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 5 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 5's axi5_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi5_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 5 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 5 can issue size equal to width or size less than width instructions.</li> </ul>

**CTRL\_REG\_59**

Address: Operational Base + offset( 0xec)

Controller register 59

bit	Attr	Reset Value	Description
31:16	RW	0x0	Masks certain bits of the cs_val_0 parameter from the incoming user address. When used together, the cs_msk_0 and cs_val_0 parameters allow the memory controller to interface with memory devices with different row and column addresses.
15:0	RW	0x0	<p>Allows the port to accept size less than width transactions on AXI port 7 from requestors with the lowest four bits set to Z. Each bit Z corresponds to the lowest four bits of requestor Z, meaning that if bit [0] is set, then requestors with the lowest four bits of 0 for AXI port 7 will be able to send size less than width transactions. Example: if there are 5 requestors on a port and requestors 1, 3 and 4 were able to send size less than width transactions, then port 7's axi7_en_size_lt_width_instr parameter would be set to 0x001A.</p> <p>Note: Only the lowest 4 bits of the source ID are compared against the value in the axi7_en_size_lt_width_instr parameter.</p> <p>Note: While this Memory Controller is connected to AHB and AXI buses, the Databahn core only interfaces with the AXI-Databahn interface blocks. Therefore, all port parameters are represented in terms of AXI.</p> <p>Note: Attempts to perform a narrow transfer for an ID that is not enabled to accept narrow transfers</p>

			<p>will result in undefined behavior. No external indication will be made that this error has occurred.</p> <ul style="list-style-type: none"> <li>• 'b0 = Requestor Z of port 7 may only issue size equal to width instructions.</li> <li>• 'b1 = Requestor Z of port 7 can issue size equal to width or size less than width instructions.</li> </ul>
--	--	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_60**

Address: Operational Base + offset( 0xf0)

Controller register 60

bit	Attr	Reset Value	Description
31:16	RW	0x0	Holds the comparison value to match against the incoming user address as the maximum address for chip select 0. When used together, the cs_msk_0 and cs_val_0 parameters allow the memory controller to interface with memory devices with different row and column addresses.
15:0	RW	0x0	Masks certain bits of the cs_val_1 parameter from the incoming user address. When used together, the cs_msk_1 and cs_val_1 parameters allow the memory controller to interface with memory devices with different row and column addresses.

**CTRL\_REG\_61**

Address: Operational Base + offset( 0xf4)

Controller register 61

bit	Attr	Reset Value	Description
31:16	RW	0x0	Minimum number of cycles required for DLL reset.
15:0	RW	0x0	Holds the comparison value to match against the incoming user address as the maximum address for chip select 1. When used together, the cs_msk_1 and cs_val_1 parameters allow the memory controller to interface with memory devices with different row and column addresses.

**CTRL\_REG\_62**

Address: Operational Base + offset( 0xf8)

Controller register 62

bit	Attr	Reset Value	Description
31:16	RW	0x0	Counts the number of idle cycles before memory self-refresh with memory and controller clock gating low power mode. This parameter must be programmed to a non-zero value for proper operation.
15:0	RW	0x0	Counts the number of idle cycles before memory self-refresh with memory clock gating low power mode. This parameter must be programmed to a non-zero value for proper operation.

**CTRL\_REG\_63**

Address: Operational Base + offset( 0xfc)

Controller register 63

bit	Attr	Reset Value	Description
31:16	RW	0x0	Sets the number of cycles that the memory controller will wait before attempting to re-lock the DLL when using the controller clock gating mode low power mode. This counter will ONLY be used

			in this mode, the deepest low power mode. When this counter expires, the DLL will be un-gated for at least 16 cycles during which the DLL will attempt to re-lock. After 16 cycles have elapsed and the DLL has locked, then the DLL controller clock will be gated again and the counter will reset to this value. If the DLL requires more than 16 cycles to re-lock, then the un-gated time will be longer.
15:0	RW	0x0	Counts the number of idle cycles before memory power-down or power-down with memory clock gating low power mode. This parameter must be programmed to a non-zero value for proper operation.

**CTRL\_REG\_64**

Address: Operational Base + offset( 0x100)

Controller register 64

bit	Attr	Reset Value	Description
31:16	RW	0x0	DRAM TCPD parameter, defines the clock enable to pre-charge delay time for the DRAM devices, in cycles.
15:0	RW	0x0	Counts the number of cycles to the next memory self-refresh low power mode. This parameter must be programmed to a non-zero value for proper operation.

**CTRL\_REG\_65**

Address: Operational Base + offset( 0x104)

Controller register 65

bit	Attr	Reset Value	Description
31:16	RW	0x0	DRAM TPDEX(power-down exit command period) parameter in cycles.
15:0	RW	0x0	DRAM TDLL(DLL lock time) parameter in cycles.

**CTRL\_REG\_66**

Address: Operational Base + offset( 0x108)

Controller register 66

bit	Attr	Reset Value	Description
31:16	RW	0x0	DRAM TXSNR parameter, defines the DRAM time from a self-refresh exit to a command that does not require the memory DLL to be locked.
15:0	RW	0x0	DRAM TRAS_MAX parameter, defines the DRAM maximum row active time, in cycles. The user should set this value to the timing parameter in the memory specification. In practical use, the memory controller will use the value in this parameter with adjustments made for write recovery time, burst length and any internal delays if required.

**CTRL\_REG\_67**

Address: Operational Base + offset( 0x10c)

Controller register 67

bit	Attr	Reset Value	Description
31:16	R	0x2044	Controller version number.
15:0	RW	0x0	DRAM TXSR parameter, defines the DRAM time from a self-refresh exit to a command that requires the memory DLL to be locked.



**CTRL\_REG\_68**

Address: Operational Base + offset( 0x110)

Controller register 68

bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	RW	0x0	DRAM TINIT(initialization time) parameter in cycles.

**CTRL\_REG\_69**

Address: Operational Base + offset( 0x114)

Controller register 69

bit	Attr	Reset Value	Description
31:0	RW	0x0	Currently Not Functional.

**CTRL\_REG\_70**

Address: Operational Base + offset( 0x118)

Controller register 70

bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>DLL_CTRL_REG_0_0 Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 0.</p> <ul style="list-style-type: none"> <li>• Bit [28] = DLL Bypass Control            'b0 = Normal operational mode. In this mode, the DLL uses dll_ctrl_reg_0_0 [14:8] for the read DQS and dll_ctrl_reg_1_0 [14:8] for clk_wr.            'b1 = Bypass Mode is on. In this mode, the DLL uses dll_ctrl_reg_0_0 [23:15] for the read DQS and dll_ctrl_reg_1_0 [23:15] for clk_wr.</li> <li>• Bits [23:15] = Holds the read DQS delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_0 [28] = 'b1)</li> <li>• Bits [14:8] = Holds the read DQS delay setting when the DLL is operating in normal mode. (dll_ctrl_reg_0_0 [28] = 'b0) Typically, this value is 1/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle.</li> <li>• Bits [7:0] = DLL Start Point Control. This value is loaded into the DLL at initialization and is the value at which the DLL will begin searching for a lock.</li> <li>• All other bits undefined</li> </ul>

**CTRL\_REG\_71**

Address: Operational Base + offset( 0x11c)

Controller register 71

bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>DLL_CTRL_REG_0_1 Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 1.</p> <p>All bits are defined samely as DLL_CTRL_REG_0_0.</p>

**CTRL\_REG\_72**

Address: Operational Base + offset( 0x120)

Controller register 72

bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_0_2



			Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 2.  All bits are defined samely as DLL_CTRL_REG_0_0.
--	--	--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_73**

Address: Operational Base + offset( 0x124)

Controller register 73

bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_0_3 Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 3.  All bits are defined samely as DLL_CTRL_REG_0_0.

**CTRL\_REG\_74**

Address: Operational Base + offset( 0x128)

Controller register 74

bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_0 Holds the clk_wr settings and the Dll increment value for data slice 0.  <ul style="list-style-type: none"> <li>• Bits [23:15] = Holds the clk_wr delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_0 [28] = 'b1)</li> <li>• Bits [14:8] = Holds the clk_wr delay setting in normal mode. (dll_ctrl_reg_0_0 [28] = 'b0) Typically, this value is 3/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle.</li> <li>• Bits [7:0] = DLL Increment Value. This sets the increment used by the DLL when searching for a lock. We recommends keeping this field small (around 0x4) to keep the steps gradual.</li> <li>• All other bits undefined</li> </ul>

**CTRL\_REG\_75**

Address: Operational Base + offset( 0x12c)

Controller register 75

bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_1 Holds the clk_wr settings and the Dll increment value for data slice 1.  All bits are defined samely as DLL_CTRL_REG_1_0.

**CTRL\_REG\_76**

Address: Operational Base + offset( 0x130)

Controller register 76

bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_2 Holds the clk_wr settings and the Dll increment value for data slice 2.  All bits are defined samely as DLL_CTRL_REG_1_0.

**CTRL\_REG\_77**

Address: Operational Base + offset( 0x134)

Controller register 77

bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_3 Holds the clk_wr settings and the Dll increment value for data slice 3.  All bits are defined samely as DLL_CTRL_REG_1_0.

**CTRL\_REG\_78**

Address: Operational Base + offset( 0x138)

Controller register 78

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_0_0 Reports the lock status and the encoder value for data slice 0.  <ul style="list-style-type: none"> <li>• Bits [31:1] = Reports the DLL encoder value from the master DLL to the slave DLL's. The slaves use this value to set up their delays for the clk_wr and read DQS signals.</li> <li>• Bit [0] = DLL Lock Indicator  'b0 = DLL has not locked.  'b1 = DLL is locked.</li> </ul>

**CTRL\_REG\_79**

Address: Operational Base + offset( 0x13c)

Controller register 79

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_0_1 Reports the lock status and the encoder value for data slice 1.  All bits are defined samely as DLL_OBS_REG_0_0.

**CTRL\_REG\_80**

Address: Operational Base + offset( 0x140)

Controller register 80

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_0_2 Reports the lock status and the encoder value for data slice 2.  All bits are defined samely as DLL_OBS_REG_0_0.

**CTRL\_REG\_81**

Address: Operational Base + offset( 0x144)

Controller register 81

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_0_3 Reports the lock status and the encoder value for data slice 3.  All bits are defined samely as DLL_OBS_REG_0_0.

**CTRL\_REG\_82**

Address: Operational Base + offset( 0x148)

Controller register 82

bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>PAD_CTRL_REG_0 Controls pad termination, type and IDDQ settings.</p> <ul style="list-style-type: none"> <li>• Bits [31] = DDR PHY test selection.  `b0 = not select DDR PHY test  `b1 = select DDR PHY test</li> <li>• Bits [23:20] = DDR PAD driver strength.  `b1111 = Driver resistance 28 ohm  `b1110 = Driver resistance 32 ohm  `b1100 = Driver resistance 37 ohm  `b1010 = Driver resistance 45 ohm  `b1000 = Driver resistance 56 ohm  `b0110 = Driver resistance 74 ohm  `b0100 = Driver resistance 111 ohm  `b0010 = Driver resistance 222 ohm  other bits invalid</li> <li>• Bits [19] = DQS differential selection.  `b0 = select DQS single-ended  `b1 = select DQS differential</li> <li>• Bits [18] = mobile DDR selection.  `b0 = select DDR2  `b1 = select mobile DDR</li> <li>• Bits [17:16] = select ODT value for open feedback PAD, must be programmed as `b00.</li> <li>• Bits [15:14] = select ODT value for odt PAD.</li> <li>• Bits [13:12] = select ODT value for clk PAD.</li> <li>• Bits [11:10] = select ODT value for command signal PAD.</li> <li>• Bits [9:8] = select ODT value for address signal PAD.</li> <li>• Bits [7:6] = select ODT value for dq/dqs/dm of slice3</li> <li>• Bits [5:4] = select ODT value for dq/dqs/dm of slice2</li> <li>• Bits [3:2] = select ODT value for dq/dqs/dm of slice1</li> <li>• Bits [1:0] = select ODT value for dq/dqs/dm of slice0  ODT value as follow:  `b00 = invalid  `b01 = 150 ohm  `b10 = 75 ohm  `b11 = 50 ohm</li> </ul>

**CTRL\_REG\_83**

Address: Operational Base + offset( 0x14c)

Controller register 83

bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>PHY_CTRL_REG_0_0 Controls pad output enable times and other PHY parameters for data slice 0.</p> <ul style="list-style-type: none"> <li>• Bit [31] = Enables dynamic termination select in the PHY for the DM pads.  `b0 = Disabled  `b1 = Enabled</li> </ul>

			<ul style="list-style-type: none"> <li>• Bit [29] = Controls termination enable for the DM pads. Set to 'b1 to disable termination.  'b0 = Enabled  'b1 = Disabled</li> <li>• Bits [28] = Echo gate control for data slice 0. Default 0x0.  'b0 = Uses the dfi_rddata_en signal to create a gate.  'b1 = Creates an echo_gate signal.</li> <li>• Bit [27] = Gather FIFO Enable  'b0 = Disabled  'b1 = Enabled</li> <li>• Bits [26:24] = Defines the read data delay. Holds the number of cycles to delay the dfi_rddata_en signal prior to enabling the read FIFO. After this delay, the read pointers begin incrementing the read FIFO. Default 0x3.</li> <li>• Bit [20] = Sets the pad output enable polarity. Default 0x0.  'b0 = OEN pad  'b1 = OE pad</li> <li>• Bit [16] = Subtracts 1/2 cycle from the DQS gate value programmed into phy_ctrl_reg_1_0 [2:0] by 1/2 cycle. Default 0x0. This is used when the gate is being aligned to the first DQS, and then is removed to move the gate back into the center of the preamble. This parameter is controlled by the hardware logic when hardware gate training is enabled but should be controlled by software when software leveling is used.  'b0 = Do not adjust  'b1 = Adjust the DQS gate by 1/2 clock forward.</li> <li>• Bits [15:12] = Adjusts the starting point of the DQS pad output enable window. Lower numbers pull the rising edge earlier in time, and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x2.</li> <li>• Bits [11:8] = Adjusts the ending point of the DQS pad output enable window. Lower numbers pull the falling edge earlier in time, and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. This field must be set to at least the value of bits [14:12]+2 to prevent disabling the pad before the data is completely written. Default 0x7.</li> <li>• Bits [6:4] = Adjusts the starting point of the DQ pad output enable window. Lower numbers pull the rising edge earlier in time, and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x1.</li> <li>• Bits [2:0] = Adjusts the ending point of the DQ pad output enable window. Lower numbers pull the falling edge earlier in time, and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. This field must be set to at least the value of bits [6:4]+2 to prevent</li> </ul>
--	--	--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

			disabling the pad before the data is completely written. Default 0x4. • All other bits undefined
--	--	--	-----------------------------------------------------------------------------------------------------

**CTRL\_REG\_84**

Address: Operational Base + offset( 0x150)

Controller register 84

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_1 Controls pad output enable times and other PHY parameters for data slice 1.  All bits are defined samely as PHY_CTRL_REG_0_0.

**CTRL\_REG\_85**

Address: Operational Base + offset( 0x154)

Controller register 85

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_2 Controls pad output enable times and other PHY parameters for data slice 2.  All bits are defined samely as PHY_CTRL_REG_0_0.

**CTRL\_REG\_86**

Address: Operational Base + offset( 0x158)

Controller register 86

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_3 Controls pad output enable times and other PHY parameters for data slice 3.  All bits are defined samely as PHY_CTRL_REG_0_0.

**CTRL\_REG\_87**

Address: Operational Base + offset( 0x15c)

Controller register 87

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_0 Controls pad termination and loopback for data slice 0.  • Bits [31:28] = Defines the pad dynamic termination select enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution. • Bits [27:24] = Defines the pad dynamic termination select disable time. Larger values reduce the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution. • Bit [23] = Enables dynamic termination select in the PHY for the DQS pads. 'b0 = Disabled 'b1 = Enabled • Bit [22] = Controls the polarity of the tsel signal for the DQS and DM pads. 'b0 = Negative Polarity

			<ul style="list-style-type: none"> <li>'b1 = Positive Polarity</li> <li>• Bit [21] = Triggers a data return to the memory controller.</li> <li>'b0 = No action</li> <li>'b1 = Sends loopback data on the dfi_rddata signal.</li> <li>• Bit [20] = Selects data output type for phy_obs_reg_0_0 [23:8].</li> <li>'b0 = Return the expected data.</li> <li>'b1 = Return the actual data.</li> <li>• Bits [19:18] = Loopback control</li> <li>'b00 = Normal operational mode</li> <li>'b01 = Enables loopback write mode</li> <li>'b10 = Stop loopback to check the error register</li> <li>'b11 = Clear loopback registers</li> <li>• Bits [17] = Controls the loopback read multiplexer.</li> <li>• Bits [16] = Controls the internal write multiplexer.</li> <li>• Bits [14:12] = Sets the cycle delay between the LFSR and loopback error check logic. Note that 'h7 is not a valid selection and will result in a false passing result.</li> <li>• Bits [10:8] = Gate Error Delay. Controls whether the dqs_err check is enabled, and if so, allows the user to adjust the length of time from the dfi_rddata_en signal assertion to the maximum time in which all of the data should have been returned. If the data is not returned in the time expected by the MC, an error condition may occur. The MC will auto-close the gate and generate an interrupt. In the event of an error, the user should check the connections between the SOC and DRAM devices. This field can be useful for debugging the I/O during RTL integration. The default value is based on the delay information provided at configuration. Setting these bits to 0x0 disables error checking.</li> <li>• Bits [5:4] = Adjusts the closing of the gate. This field default value is based on the delay information provided at configuration.</li> <li>• Bits [2:0] = Coarse adjust of gate open time. This value is the number of cycles to delay the dfi_rddata_en signal prior to opening the gate in full cycle increments. Decreasing this value pulls the gate earlier in time. This field, along with the phy_ctrl_reg_0_0 [16] bit should be programmed such that the gate signal lands in the valid DQS gate window.</li> <li>• All other bits undefined</li> </ul>
--	--	--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_88**

Address: Operational Base + offset( 0x160)

Controller register 88

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_1 Controls pad termination and loopback for data slice 1.  All bits are defined samely as PHY_CTRL_REG_1_0.

**CTRL\_REG\_89**

Address: Operational Base + offset( 0x164)

## Controller register 89

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_2 Controls pad termination and loopback for data slice 2.  All bits are defined samely as PHY_CTRL_REG_1_0.

## CTRL\_REG\_90

Address: Operational Base + offset( 0x168)

## Controller register 90

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_3 Controls pad termination and loopback for data slice 3.  All bits are defined samely as PHY_CTRL_REG_1_0.

## CTRL\_REG\_91

Address: Operational Base + offset( 0x16c)

## Controller register 91

bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_2 Selects the dfi_rddata_valid delay.  <ul style="list-style-type: none"> <li>• Bit [25] = Selects the address / command path.  `b0 = DFI control signals are captured 1 cycle after being sent from the MC.  `b1 = DFI control signals are captured cycle after being sent from the MC.  Note: The Low Latency option is not functional at this time.</li> <li>• Bit [24] = Selects the write latency path.  `b0 = Write Data is captured 1 cycle after being sent from the MC  `b1 = Write data is captured cycle after being sent from the MC.  Note: The Low Latency option is not functional at this time.</li> <li>• Bit [23] = DFI control for Mobile MCs.</li> <li>• Bit [5] = Enables the pad inputs specifically for external loopback. This bit is tied to the internal signal lpbk_rddata_en.  `b0 = Normal Operation  `b1 = Loopback Mode</li> <li>• Bit [4] = Enables the pad outputs specifically for external loopback. This bit is tied to the internal signal lpbk_wrdata_en.  `b0 = Normal Operation  `b1 = Loopback Mode</li> <li>• Bits [3:0] = Sets the dfi_rddata_valid delay relative to dfi_rddata_en. At default, all phy_ctrl_reg_0_X [26:24] parameter fields have the same default values, and this field is set to phy_ctrl_reg_0_X [26:24] + 1.</li> <li>• All other bits undefined</li> </ul>

## CTRL\_REG\_92

Address: Operational Base + offset( 0x170)



## Controller register 92

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>PHY_OBS_REG_0_0 Controls loopback status, data and masking info for slice 0.</p> <ul style="list-style-type: none"> <li>• Bit [24] = Status signal to indicate that the logic gate had to be forced closed.            'b0 = Normal operation            'b1 = Gate close was forced</li> <li>• Bits [23:8] = Loopback data. Reports the actual data or expected data, depending on the setting of the phy_ctrl_reg_1_0 [20] parameter bit.</li> <li>• Bits [5:4] = Loopback mask data. Reports the actual data mask or the expected data mask, depending on the setting of the phy_ctrl_reg_1_0 [20] parameter bit.</li> <li>• Bit [1] = Reports status of loopback errors.            'b0 = Last Loopback test had no errors.            'b1 = Last Loopback test contained data errors.</li> <li>• Bit [0] = Defines the status of the loopback mode.            'b0 = Not in loopback mode            'b1 = In Loopback mode</li> <li>• All other bits Reserved</li> </ul>

**CTRL\_REG\_93**

Address: Operational Base + offset( 0x174)

## Controller register 93

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>PHY_OBS_REG_0_1 Controls loopback status, data and masking info for slice 1.</p> <p>All bits are defined samely as PHY_OBS_REG_0_0.</p>

**CTRL\_REG\_94**

Address: Operational Base + offset( 0x178)

## Controller register 94

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>PHY_OBS_REG_0_2 Controls loopback status, data and masking info for slice 2.</p> <p>All bits are defined samely as PHY_OBS_REG_0_0.</p>

**CTRL\_REG\_95**

Address: Operational Base + offset( 0x17c)

## Controller register 95

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>PHY_OBS_REG_0_3 Controls loopback status, data and masking info for slice 3.</p> <p>All bits are defined samely as PHY_OBS_REG_0_0.</p>

**CTRL\_REG\_96**

Address: Operational Base + offset( 0x180)

Controller register 96

bit	Attr	Reset Value	Description
31:0	R	0x0	OUT_OF_RANGE_ADDR[31:0] Holds the address of the command that caused an out-of-range interrupt request to the memory devices.  This register must be combined with OUT_OF_RANGE_ADDR[33:32].

**CTRL\_REG\_97**

Address: Operational Base + offset( 0x184)

Controller register 97

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	OUT_OF_RANGE_ADDR[33:32] Holds the address of the command that caused an out-of-range interrupt request to the memory devices.  This register must be combined with OUT_OF_RANGE_ADDR[31:0].

**CTRL\_REG\_98**

Address: Operational Base + offset( 0x188)

Controller register 98

bit	Attr	Reset Value	Description
31:0	R	0x0	PORT_CMD_ERROR_ADDR[31:0] Holds the address of the command that caused a port command error condition.  This register must be combined with PORT_CMD_ERROR_ADDR[33:32].

**CTRL\_REG\_99**

Address: Operational Base + offset( 0x18c)

Controller register 99

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	PORT_CMD_ERROR_ADDR[33:32] Holds the address of the command that caused a port command error condition.  This register must be combined with PORT_CMD_ERROR_ADDR[31:0].

**CTRL\_REG\_100**

Address: Operational Base + offset( 0x190)

Controller register 100

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_0[31:0] Reports master DLL info for delay line 0. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.

			This register is part of DLL_OBS_REG_1_0[156:0].
--	--	--	--------------------------------------------------

**CTRL\_REG\_101**

Address: Operational Base + offset( 0x194)

Controller register 101

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_0[63:32] Reports master DLL info for delay line 0. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_0[156:0].

**CTRL\_REG\_102**

Address: Operational Base + offset( 0x198)

Controller register 102

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_0[95:64] Reports master DLL info for delay line 0. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_0[156:0].

**CTRL\_REG\_103**

Address: Operational Base + offset( 0x19c)

Controller register 103

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_0[127:96] Reports master DLL info for delay line 0. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_0[156:0].

**CTRL\_REG\_104**

Address: Operational Base + offset( 0x1a0)

Controller register 104

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_1_0[156:128] Reports master DLL info for delay line 0. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_0[156:0].

**CTRL\_REG\_105**

Address: Operational Base + offset( 0x1a4)

## Controller register 105

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_1[31:0] Reports master DLL info for delay line 1. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_1[156:0].

**CTRL\_REG\_106**

Address: Operational Base + offset( 0x1a8)

## Controller register 106

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_1[63:32] Reports master DLL info for delay line 1. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_1[156:0].

**CTRL\_REG\_107**

Address: Operational Base + offset( 0x1ac)

## Controller register 107

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_1[95:64] Reports master DLL info for delay line 1. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_1[156:0].

**CTRL\_REG\_108**

Address: Operational Base + offset( 0x1b0)

## Controller register 108

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_1[127:96] Reports master DLL info for delay line 1. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_1[156:0].

**CTRL\_REG\_109**

Address: Operational Base + offset( 0x1b4)

## Controller register 109

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_1_1[156:128]

			<p>Reports master DLL info for delay line 1. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.</p> <p>This register is part of DLL_OBS_REG_1_1[156:0].</p>
--	--	--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_110**

Address: Operational Base + offset( 0x1b8)

Controller register 110

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>DLL_OBS_REG_1_2[31:0] Reports master DLL info for delay line 2. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.</p> <p>This register is part of DLL_OBS_REG_1_2[156:0].</p>

**CTRL\_REG\_111**

Address: Operational Base + offset( 0x1bc)

Controller register 111

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>DLL_OBS_REG_1_2[63:32] Reports master DLL info for delay line 2. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.</p> <p>This register is part of DLL_OBS_REG_1_2[156:0].</p>

**CTRL\_REG\_112**

Address: Operational Base + offset( 0x1c0)

Controller register 112

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>DLL_OBS_REG_1_2[95:64] Reports master DLL info for delay line 2. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.</p> <p>This register is part of DLL_OBS_REG_1_2[156:0].</p>

**CTRL\_REG\_113**

Address: Operational Base + offset( 0x1c4)

Controller register 113

bit	Attr	Reset Value	Description
31:0	R	0x0	<p>DLL_OBS_REG_1_2[127:96] Reports master DLL info for delay line 2. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.</p>

			This register is part of DLL_OBS_REG_1_2[156:0].
--	--	--	--------------------------------------------------

**CTRL\_REG\_114**

Address: Operational Base + offset( 0x1c8)

Controller register 114

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_1_2[156:128] Reports master DLL info for delay line 2. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_2[156:0].

**CTRL\_REG\_115**

Address: Operational Base + offset( 0x1cc)

Controller register 115

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_3[31:0] Reports master DLL info for delay line 3. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_3[156:0].

**CTRL\_REG\_116**

Address: Operational Base + offset( 0x1d0)

Controller register 116

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_3[63:32] Reports master DLL info for delay line 3. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_3[156:0].

**CTRL\_REG\_117**

Address: Operational Base + offset( 0x1d4)

Controller register 117

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_3[95:64] Reports master DLL info for delay line 3. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_3[156:0].

**CTRL\_REG\_118**

Address: Operational Base + offset( 0x1d8)

Controller register 118

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_1_3[127:96] Reports master DLL info for delay line 3. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_3[156:0].

**CTRL\_REG\_119**

Address: Operational Base + offset( 0x1dc)

Controller register 119

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_1_3[156:128] Reports master DLL info for delay line 3. One-hot register that identifies which multiplexer controls the master delay line. As an example, 0x200 = 'b1000000000, so this identifies the 10th multiplexer.  This register is part of DLL_OBS_REG_1_3[156:0].

**CTRL\_REG\_120**

Address: Operational Base + offset( 0x1e0)

Controller register 120

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_0[31:0] Reports the read DQS delay value for data slice 0.  This register is part of DLL_OBS_REG_2_0[156:0].

**CTRL\_REG\_121**

Address: Operational Base + offset( 0x1e4)

Controller register 121

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_0[63:32] Reports the read DQS delay value for data slice 0.  This register is part of DLL_OBS_REG_2_0[156:0].

**CTRL\_REG\_122**

Address: Operational Base + offset( 0x1e8)

Controller register 122

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_0[95:64] Reports the read DQS delay value for data slice 0.  This register is part of DLL_OBS_REG_2_0[156:0].

**CTRL\_REG\_123**

Address: Operational Base + offset( 0x1ec)

Controller register 123



bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_0[127:96] Reports the read DQS delay value for data slice 0.  This register is part of DLL_OBS_REG_2_0[156:0].

**CTRL\_REG\_124**

Address: Operational Base + offset( 0x1f0)

Controller register 124

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_2_0[156:128] Reports the read DQS delay value for data slice 0.  This register is part of DLL_OBS_REG_2_0[156:0].

**CTRL\_REG\_125**

Address: Operational Base + offset( 0x1f4)

Controller register 125

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_1[31:0] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_2_1[156:0].

**CTRL\_REG\_126**

Address: Operational Base + offset( 0x1f8)

Controller register 126

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_1[63:32] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_2_1[156:0].

**CTRL\_REG\_127**

Address: Operational Base + offset( 0x1fc)

Controller register 127

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_1[95:64] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_2_1[156:0].

**CTRL\_REG\_128**

Address: Operational Base + offset( 0x200)

Controller register 128

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_1[127:96] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_2_1[156:0].

**CTRL\_REG\_129**

Address: Operational Base + offset( 0x204)

Controller register 129

bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_2_1[156:128] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_2_1[156:0].

**CTRL\_REG\_130**

Address: Operational Base + offset( 0x208)

Controller register 130

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_2[31:0] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_2_2[156:0].

**CTRL\_REG\_131**

Address: Operational Base + offset( 0x20c)

Controller register 131

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_2[63:32] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_2_2[156:0].

**CTRL\_REG\_132**

Address: Operational Base + offset( 0x210)

Controller register 132

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_2[95:64] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_2_2[156:0].

**CTRL\_REG\_133**

Address: Operational Base + offset( 0x214)

Controller register 133

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_2[127:96] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_2_2[156:0].

**CTRL\_REG\_134**

Address: Operational Base + offset( 0x218)

Controller register 134

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_2_2[156:128] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_2_2[156:0].

**CTRL\_REG\_135**

Address: Operational Base + offset( 0x21c)

Controller register 135

bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:0	R	0x0	DLL_OBS_REG_2_3[31:0] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_2_3[156:0].
------	---	-----	-------------------------------------------------------------------------------------------------------------------------------------

**CTRL\_REG\_136**

Address: Operational Base + offset( 0x220)

Controller register 136

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_3[63:32] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_2_3[156:0].

**CTRL\_REG\_137**

Address: Operational Base + offset( 0x224)

Controller register 137

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_3[95:64] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_2_3[156:0].

**CTRL\_REG\_138**

Address: Operational Base + offset( 0x228)

Controller register 138

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_2_3[127:96] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_2_3[156:0].

**CTRL\_REG\_139**

Address: Operational Base + offset( 0x22c)

Controller register 139

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_2_3[156:128] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_2_3[156:0].

**CTRL\_REG\_140**

Address: Operational Base + offset( 0x230)

Controller register 140

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_0[31:0] Reports the clk_wr delay value for data slice 0.  This register is part of DLL_OBS_REG_3_0[156:0].

**CTRL\_REG\_141**

Address: Operational Base + offset( 0x234)

Controller register 141

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_0[63:32]

			Reports the read DQS delay value for data slice 0. This register is part of DLL_OBS_REG_3_0[156:0].
--	--	--	--------------------------------------------------------------------------------------------------------

**CTRL\_REG\_142**

Address: Operational Base + offset( 0x238)

Controller register 142

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_0[95:64] Reports the read DQS delay value for data slice 0. This register is part of DLL_OBS_REG_3_0[156:0].

**CTRL\_REG\_143**

Address: Operational Base + offset( 0x23c)

Controller register 143

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_0[127:96] Reports the read DQS delay value for data slice 0. This register is part of DLL_OBS_REG_3_0[156:0].

**CTRL\_REG\_144**

Address: Operational Base + offset( 0x240)

Controller register 144

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_3_0[156:128] Reports the read DQS delay value for data slice 0. This register is part of DLL_OBS_REG_3_0[156:0].

**CTRL\_REG\_145**

Address: Operational Base + offset( 0x244)

Controller register 145

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_1[31:0] Reports the clk_wr delay value for data slice 1. This register is part of DLL_OBS_REG_3_1[156:0].

**CTRL\_REG\_146**

Address: Operational Base + offset( 0x248)

Controller register 146

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_1[63:32] Reports the read DQS delay value for data slice 1. This register is part of DLL_OBS_REG_3_1[156:0].

**CTRL\_REG\_147**

Address: Operational Base + offset( 0x24c)

Controller register 147

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_1[95:64] Reports the read DQS delay value for data slice 1.

			This register is part of DLL_OBS_REG_3_1[156:0].
--	--	--	--------------------------------------------------

**CTRL\_REG\_148**

Address: Operational Base + offset( 0x250)

Controller register 148

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_1[127:96] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_3_1[156:0].

**CTRL\_REG\_149**

Address: Operational Base + offset( 0x254)

Controller register 149

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_3_1[156:128] Reports the read DQS delay value for data slice 1.  This register is part of DLL_OBS_REG_3_1[156:0].

**CTRL\_REG\_150**

Address: Operational Base + offset( 0x258)

Controller register 150

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_2[31:0] Reports the clk_wr delay value for data slice 2.  This register is part of DLL_OBS_REG_3_2[156:0].

**CTRL\_REG\_151**

Address: Operational Base + offset( 0x25c)

Controller register 151

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_2[63:32] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_3_2[156:0].

**CTRL\_REG\_152**

Address: Operational Base + offset( 0x260)

Controller register 152

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_2[95:64] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_3_2[156:0].

**CTRL\_REG\_153**

Address: Operational Base + offset( 0x264)

Controller register 153

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_2[127:96] Reports the read DQS delay value for data slice 2.

		This register is part of DLL_OBS_REG_3_2[156:0].
--	--	--------------------------------------------------

**CTRL\_REG\_154**

Address: Operational Base + offset( 0x268)

Controller register 154

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_3_2[156:128] Reports the read DQS delay value for data slice 2.  This register is part of DLL_OBS_REG_3_2[156:0].

**CTRL\_REG\_155**

Address: Operational Base + offset( 0x26c)

Controller register 155

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_3[31:0] Reports the clk_wr delay value for data slice 3.  This register is part of DLL_OBS_REG_3_3[156:0].

**CTRL\_REG\_156**

Address: Operational Base + offset( 0x270)

Controller register 156

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_3[63:32] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_3_3[156:0].

**CTRL\_REG\_157**

Address: Operational Base + offset( 0x274)

Controller register 157

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_3[95:64] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_3_3[156:0].

**CTRL\_REG\_158**

Address: Operational Base + offset( 0x278)

Controller register 158

bit	Attr	Reset Value	Description
31:0	R	0x0	DLL_OBS_REG_3_3[127:96] Reports the read DQS delay value for data slice 3.  This register is part of DLL_OBS_REG_3_3[156:0].

**CTRL\_REG\_159**

Address: Operational Base + offset( 0x27c)

Controller register 159

bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:0	R	0x0	DLL_OBS_REG_3_3[156:128] Reports the read DQS delay value for data slice 3.

		This register is part of DLL_OBS_REG_3_3[156:0].
--	--	--------------------------------------------------

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

## 30.4 Functional Description

### 30.4.1 Initialization

The memory controller is designed such that it requires a sequence for correct operation after all power to the chip and to the memory devices is stable. The memory controller does not include circuitry to control the activation of power and ground to the system. Once the power to the memory devices and the chip is stable, the memory controller must be initialized and it will then automatically initialize the memory devices. For memory initialization, please refer to JEDEC specifications and data sheets of memory devices. The procedure to initialize the memory controller is as follows:

1. Clear the rst\_n signal by driving it to 'b0. All programmable registers will be cleared.
2. Set the rst\_n signal synchronously with the memory controller clock by driving the signal to 'b1.
3. Issue write register commands to configure the registers. Keep the start parameter de-asserted during this initialization step.
4. Assert the start parameter. This triggers the memory controller to execute the initialization sequence using the parameters written into the registers.

The memory controller will wait for the PHY to assert the dfi\_init\_complete signal, which indicates that the PHY and memory devices are ready to accept commands.

### 30.4.2 Address Mapping

#### 30.4.2.1 DDR SDRAM Address Mapping Options

The address structure of DDR SDRAM devices contains five fields. Each of these fields can be individually addressed when accessing the DRAM. The address map for this memory controller is ordered as follows:

Chip Select -- Row -- Bank -- Column -- Datapath

The maximum widths of the fields are based on the configuration settings. The actual widths of the fields may be smaller if the device address width parameters (addr\_pins\_X, eight\_bank\_mode\_X and column\_size\_X) are programmed differently.

In our system, the sum of number of chip select, bank bit widths, row address bit widths and column address bit widths is up to 29.

#### 30.4.2.2 Heterogeneous Memory Devices

This memory controller can support different-sized devices on each rank of memory, where a rank is defined as a group of memory banks that all use one chip select. The mapping of the user address to the available memory is controlled by the addr\_pins\_X, aprebit\_X, column\_size\_X, eight\_bank\_mode\_X, cs\_msk\_X and cs\_val\_X parameters, where X represents the chip select number.

Since the row and column sizes for the attached memory devices are not identical, the memory controller must be programmed for the row/column sizes of each device. The parameters cs\_val\_X and cs\_msk\_X work together to control this interface.

The cs\_val\_X parameter is the value that each incoming user address is compared against, and the cs\_msk\_X parameter determines relevant bits for this comparison. Together, these parameters identify which device to access for each operation.

The memory map must be defined with the largest devices (most row/column bits) at the lowest address space. The mapping of the memory space is defined with the cs\_val\_X and cs\_msk\_X parameter settings and it involves these two steps:

1. Determine the cs\_msk\_X parameter value for each chip select based on the number of chip selects, the maximum number of row and column bits, and the actual address size of the device.



2. Determine the correct value for the cs\_val\_X parameters to keep the largest devices mapped to the lowest memory addresses.

### Setting the cs\_msk\_X Parameter

If there was a single homogeneous device, then the mask value would be 'b1111\_1111\_1111\_1111. For heterogeneous devices, this value must be shifted to the right with 0's shifted in to the left to fill the empty bits. This mask must be adjusted based on these factors:

- Number of Chip Selects
- Column Bits of the Devices
- Number of Banks of the Devices
- Row Bits of the Devices

For chip select adjustment, all of the cs\_msk\_X parameter values will be adjusted by the same number. Two chip selects require a single bit for encoding and therefore all cs\_msk\_X parameter values must be adjusted by a single bit. The following table shows the cs\_msk\_X parameter values with this adjustment.

Initial cs_msk_X value	Number of Chip Selects	Bits Required to Encode Chip Selects	cs_msk_X values, Shifted by Bit Select Encoding
'b 1111_1111_1111_1111	1	0	'b 1111_1111_1111_1111
'b 1111_1111_1111_1111	2	1	'b 0111_1111_1111_1111

For column, bank, and row adjustment, the adjustment in these cases is specific for the particular device used on each particular chip select. The differences to accommodate are:

- Difference between the maximum number of row bits for this configuration (15) and the actual number of row pins for the device on chip select X.
- Difference between the maximum number of column bits for this onfiguration (13) and the actual number of column bits for the device on chip select X.
- Unused bank of a DDR2 four-bank device (if used) on chip select X.

As an example, consider a memory controller interfacing with two heterogeneous devices. The values shown in the following table show the effect of column, bank and row adjustments.

cs_msk_X After CS Shifting	Chip Select	Row Bit Diff + Column Bit Diff	Bank Mode Shift	cs_msk_X Value
'b 0111_1111_1111_1111	0	3	8 bank device (0 bits)	'b 0000_1111_1111_1111
'b 0111_1111_1111_1111	1	5	4 bank device (1 bit)	'b 0000_0001_1111_1111

### Setting the cs\_val\_X Parameter

Once the values for the cs\_msk\_X parameters have been determined, the values for the cs\_val\_X parameters can be calculated. In general, the cs\_val\_X parameter will be a sum of the previous calculation's cs\_msk\_X and cs\_val\_X parameters plus one.

1. Find the cs\_msk\_X parameter with the large value and set all bits of its cs\_val\_X parameter to 'b0. If two cs\_msk\_X parameters equivalent, choose one parameter and set all bits of its cs\_val\_X parameter to 'b0. In the example shown in Table 1-2, since the large cs\_msk\_X parameter value is associated with chip select 0, the cs\_val\_0 parameter will be 0x0000.

2. Find the small cs\_msk\_X parameter value, or another cs\_msk\_X parameter equal to the first. Add the cs\_val\_X parameter and the cs\_msk\_X parameter of the previous calculation. Now add a "1" to that sum. This value is the cs\_val\_X parameter value for the chip select that had the small cs\_msk\_X parameter value.

Using the same example, the cs\_val\_1 parameter will be the sum of the cs\_msk\_0

parameter and the `cs_val_0` parameter, plus 1.

### 39.4.3 Changing the Input Clock Frequency

The operating frequency of the memory controller is dependent on an chip-level input clock. There are situations in which the user may wish to modify the frequency of the clock without resetting the memory controller. To change the clock frequency at which the memory controller operates, the memory controller must stop processing requests, the clock must be adjusted, the memory controller's timing parameters must be reprogrammed and then the memory controller can be restarted. The procedure to follow for changing the clock frequency is as follows:

1. Ensure that the memory controller is idle, that is when the `controller_busy` signal is low.
2. Put the memory devices into self-refresh mode by asserting the `srefresh` parameter to 'b1. It is imperative that the memory devices be placed into self-refresh through this parameter, and not through any other means. If the devices have been placed into self-refresh mode through one of the low power modes (programming of the `lowpower_control` parameter), then the user should bring the devices out of that low power mode manually, then program the `srefresh` parameter with a 'b1.
3. Wait until the memory devices have been placed into self-refresh mode, indicated by the `cke_status` signal.
4. Stop the memory controller by writing a 'b0 to the `start` parameter.
5. The clock frequency may now be changed. Once the clock frequency has stabilized, the user should program the following parameters with updated values: The user should also modify any other parameters that may be affected by the frequency change such as `caslat`, `caslat_lin`, `caslat_lin_gate`, any of the timing parameters, etc. The user should review the parameters carefully to ensure that all parameters that require modification are changed.
6. After all parameters have been updated, restart the memory controller by writing a 'b1 to the `start` parameter. This forces the DLL to lock to the new frequency.
7. Once the DLL has locked and the PHY has initialized, the memory controller input signal `dfi_init_complete` will be asserted. At this point, the user may bring the memory devices out of self-refresh by clearing the `srefresh` parameter to 'b0. The user does not need to wait to send commands to the memory controller after clearing the `srefresh` parameter; the memory controller will adjust for self-refresh exit time before processing memory commands.
8. If any of the memory mode registers require updating at this point, the values must be set in the EMRS parameters, and then written to the memory devices by setting the `write_modereg` parameter to 'b1.

### 39.4.3 Low Power Operation

#### 39.4.3.1 Low Power Modes

There are five low power modes available in the memory controller. The low power modes are listed from least to most power saving.

It is not possible to exit one low power mode and enter another low power mode simultaneously. The user should plan for a minimum delay between exit and entry between the two low power modes of 15 cycles in which the memory controller must remain stable.

#### Memory Power-Down

The memory controller sets the memory devices into power-down, which reduces the overall power consumption of the system, but has the least effect of all the low power modes. In this mode, the memory controller and memory clocks are fully operational, but the CKE input bit to the memory devices is de-asserted. The memory controller will continue to monitor memory refresh needs and will automatically bring the memory out of power-down to perform these refreshes. When a refresh is required, the CKE input bit to the memory devices will be re-enabled. This action brings the memory devices out of power-down. Once the refresh has been completed, the memory devices will be returned to power-down by de-asserting the CKE input bit.

**Memory Power-Down with Memory Clock Gating**

The memory controller sets the memory devices into power-down and gates off the clock to the memory devices. Refreshes will be handled as in the Memory Power-Down mode (Mode 1), with the exception that gating on the memory clock will be removed before asserting the CKE pin. After the refresh has been completed, the memory devices will be returned to power-down with the clock gated. Before the memory devices are removed from power-down, the clock will be gated on again.

**Memory Self-Refresh**

The memory controller sets the memory devices into self-refresh. In this mode, the memory controller and memory clocks are fully operational and the CKE input bit to the memory devices is de-asserted. Since the memory automatically refreshes its contents, the memory controller does not need to send explicit refreshes to the memory.

**Memory Self-Refresh with Memory Clock Gating**

The memory controller sets the memory devices into self-refresh and gates off the clock to the memory devices. Before the memory devices are removed from self-refresh, the clock will be gated on again.

**Memory Self-Refresh with Memory and Controller Clock Gating**

This is the deepest low power mode of the memory controller. The memory controller sets the memory devices into self-refresh and gates off the clock to the memory devices. In addition, the clock to the memory controller and the programming parameters will be gated off, except to a small portion of the DLL, which must remain active to maintain the lock. Before the memory devices are removed from self-refresh, the memory controller and memory clocks will be gated on. When the memory controller is in this low power mode, transactions on the port interface will be ignored.

**39.4.3.2 Low Power Mode Control**

The memory controller may enter and exit the various low power modes in the following ways:

**Automatic Entry**

Automatic Entry will occur if all of the following conditions are true:

- The mode is programmed for automatic entry by setting the relevant bit in the `lowpower_auto_enable` parameter to 'b1.
- The particular mode is enabled in the `lowpower_control` parameter.
- The memory controller is idle.
- The counter associated with this mode expires.

There are four counters in all to cover the five low power modes. There are separate counters for each of the three memory self-refresh low power modes (Modes 3, 4 and 5). Memory Power-Down mode (Mode 1) and Memory Power-Down with Memory Clock Gating mode (Mode 2) share the same counter.

All five low power modes can be entered through automatic entry, and will be exited automatically when any of the following conditions occur:

- A new read or write transaction appears at the memory controller interface.
- The memory controller must refresh the memory when in either of the power-down modes (Modes 1 or 2). After completing the memory refresh, the memory controller re-enters power-down.
- The counter for a deeper low power mode expires. The memory controller must exit the current low power mode in order to enter the deeper low power mode. A minimum of 15 cycles occur between exit from one low power mode before entering into the next low power mode, even if the counters expire within 15 cycles of each other. Note that the memory controller will not enter a less deep low power mode, regardless of which counters expire.

**Manual Entry**

Manual Entry will occur if all of the following conditions are true:

- The mode is programmed for manual entry by clearing the relevant bit in the

lowpower\_auto\_enable parameter to 'b0.

- The particular mode is set to 'b1 in the lowpower\_control parameter.

For manual entry, the lowpower\_control parameter triggers entry into the low power modes. The memory controller does not need to be idle when the a low power mode bit is enabled. When a particular mode that is programmed for manual entry is enabled, the memory controller will complete the current memory burst access, and then, regardless of the activity inside the memory

controller or at the memory interface, it will enter the selected low power mode.

Exit from a manually-entered low power mode is also manual. Clearing the lowpower\_control parameter bits to 'b0 will trigger the memory controller to pull the memory devices out of power-down or self-refresh, and command processing will resume.

*Note: The user should not use both automatic and manual modes for the various low power modes. All modes should be entered automatically or all entered manually.*

*Note: It is not possible to enter Mode 5 manually. Setting bit [0] of the lowpower\_control parameter with bit [0] of the lowpower\_auto\_enable parameter cleared will not result in any change.*

#### 39.4.4 Self-Refresh Handshaking Protocol

The user may manually trigger the memory devices to enter self-refresh mode by setting the srefresh parameter to 'b1 or by driving the user-interface signal srefresh\_enter high. Either of these methods will cause the memory controller to complete the active processes inside the memory controller and then put the memory devices into self-refresh. The CKE input will be de-asserted in this mode.

In some circumstances, the user may require confirmation that the memory devices have entered self-refresh mode. For this, the srefresh\_ack acknowledge signal has been implemented. This acknowledge is only available when self-refresh mode is triggered by driving the srefresh\_enter pin, and only if the pin is held high until the acknowledge is received. Pulsing the srefresh\_enter signal is enough to trigger entry to the self-refresh mode, but the acknowledge requires the signal to be held.

Once asserted, the srefresh\_ack signal will be de-asserted when the srefresh\_enter pin is de-asserted.

The srefresh parameter will be updated with an assertion of the srefresh\_enter pin, regardless of the behavior on the register interface. To disable self-refresh again after a srefresh\_enter pin assertion, the user will need to clear the srefresh parameter to 'b0.

#### 39.4.5 Mobile Devices DQS

For Mobile memory controller, the user must add pull-down resistors on the DRAM boundary to the DQS and DQS\_n pins. These resistors enable the system to be able to open the gate early without receiving bad data. Both resistors are very important and the system can not function accurately without them.

#### 39.4.6 Echo Gating for DDR Devices

Memory controller employs an echo-gating scheme for its DDR PHY design. The goal is to allow the PHY to reliably open the DQS gate during the pre-amble of the read DQS so that a clean DQS signal is used on the capture flop's clock pin to reliably capture data (Fig.1-2a). If the gate is turned on too early the hi-Z value of DQS could lead to the capture flops triggering and capturing erroneous data (Fig.1-2b). Opening the DQS gate too late causes problems as well (Fig.1-2c). This will potentially cause clipping on the DQS and result in a narrower first DQS pulse or even lead to the first DQS pulse being missed.

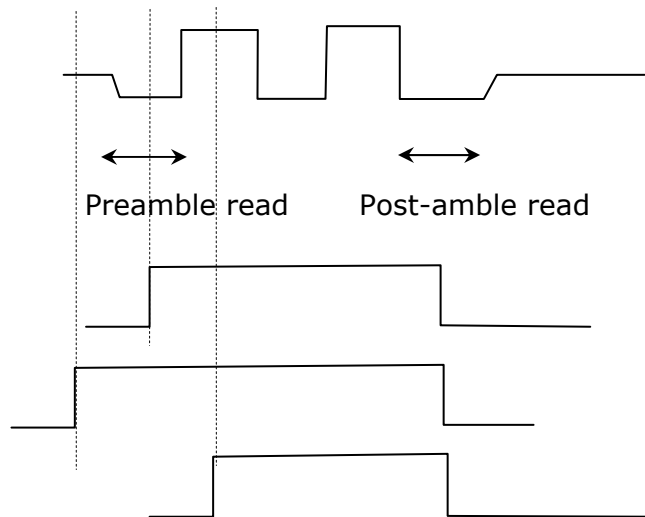


Fig.1-2a: Proper gating

Fig.1-2b: Gate too early

Fig.1-2c: Gate too late

Fig. 30-2 RK281x DDR Various gate open timing waveform

The timing for the arrival read DQS will be affected by the I/O pad delay and PCB flight delay (trace delay from SOC to DRAM devices + trace delay from DRAM devices to SOC). Because of this, it is important that the gate signal goes through the same route and gets tracks equally as VT changes.

To accomplish this, the DDR PHY has a signal called `ddr_open` which is sent to an I/O pad and then back into the PHY as feedback to create this gate signal (`ddr_open_feedback`). Since the `ddr_open` signal is routed to the top level and through the pad layer, the delay in the pads is automatically compensated for in the opening of the DQS gate. In designs where very long board traces exist (SOC to DRAM devices PCB trace), there may still be cases where the read DQS arrives even later which means that the DQS gate must open later. Also note that there is delay in the package.

To account for these delays the `ddr_open` signal is programmable with half-clock granularity.

- `phy_ctrl_reg_0_X[16]` is used to enable or disable `halfcas` to provide cycle shifts
- `phy_ctrl_reg_1_X[2:0]` adjusts the gate signal by full clock cycles

By using the two registers above, we can either adjust 1clk or 1/2clk granularity.

e.g. if `phy_ctrl_reg_1_X[2:0] = 3'b10` and `phy_ctrl_reg_0_X[16] = 1'b0`, it introduce 2clks delay;

e.g. if `phy_ctrl_reg_1_X[2:0] = 3'b10` and `phy_ctrl_reg_0_X[16] = 1'b1`, it introduce 1.5clks delay.

For  $CL=3$  and where the  $(IO + flight\_delay) < 1clk$ , program `phy_ctrl_reg_1_X[2:0] = 3'b011` and `phy_ctrl_reg_0_X[16] = 'b0`.

If  $IO + flight\_delay > 1clk$ , program `phy_ctrl_reg_1_X[2:0] = 3'b100` and `phy_ctrl_reg_0_X[16] = 'b1`, this sets 3.5 clocks of delay for the gate.

For better granularity, the feedback pad can be pinned out to be able to control the delay externally at the PCB level. Controller uses a bi-directional I/O pad (preferable the same one as the DQS pad) and pin it out to the PCB. For tuning, put a load on the pad and it will delay the gate signal usually by less than 1/2 a clock. How much loading to add can be determined with a SPICE simulation.

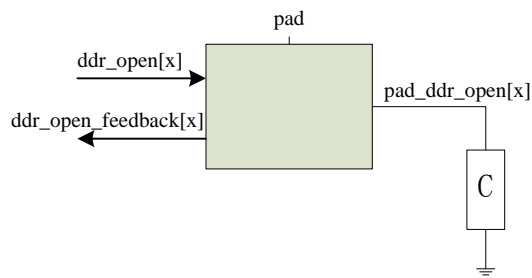


Fig. 30-3 RK281x ddr\_open\_feedback signal connection diagram

The loading connected to pad\_ddr\_open[x] will cause changes of the delay from ddr\_open[x] to ddr\_open\_feedback[x].

Because the routing delay for the ddr\_open\_feedback are too far, even if we program the GATE signal to be 0 (earliest gate turn-on), it cannot turn on on-time or earlier to prevent a clip on the incoming read DQS. The only way to solve this is to delay the incoming read DQS and this can be done by making the PCB board trace abit longer. It is recommended about 800ps trace delay which should be more than sufficient (round trip will be 1.6ns). The skew has to be as small as possible for the PCB trace. The PCB pads tool should allow you to measure and keep every pin trace length the same when doing routing. As mention, 800ps is something recommended and usually it should be sufficiently long for PCB layout engineer and yet not too long. How long contribute to Xps is something PCB tool can measure.

### 39.4.7 Read/Write Access Timing

#### 39.4.7.1 DDR2 Read/Write Access Timing

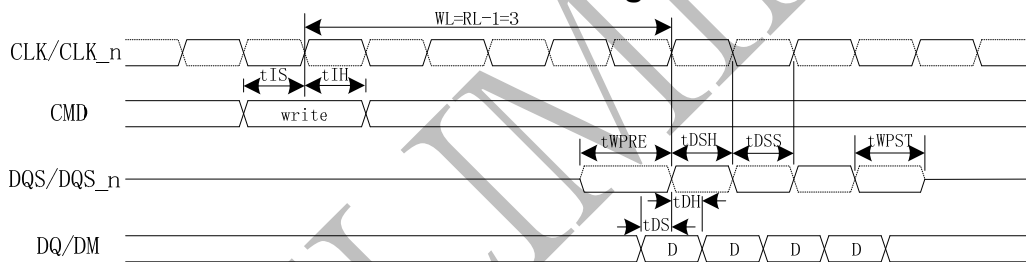


Fig. 30-4 DDR2 Burst write operation waveform: RL = CL = 4, WL = 3, BL = 4

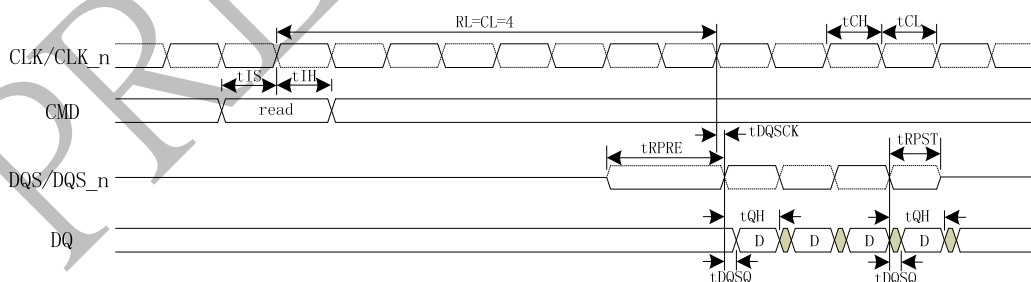


Fig. 30-5 DDR2 Burst read operation waveform: RL = CL = 4, BL = 4

Parameter	Description	DDR2-533		unit
		min	max	
tCH	CK HIGH pulse width	0.48	0.52	tCK
tCL	CK LOW pulse width	0.48	0.52	tCK
tDS	DQ and DM input setup time (differential strobe)	100	-	ps
tDH	DQ and DM input hold time	225	-	ps



	(differential strobe)			
tDSS	DQS falling edge to CK setup time	0.2	-	tCK
tDSH	DQS falling edge hold time from CK	0.2	-	tCK
tIS	Address and control input setup time	250	-	ps
tIH	Address and control input hold time	375	-	ps
tWPRE	Write preamble	0.35	-	tCK
tWPST	Write postamble	0.4	0.6	tCK
tRPRE	Read preamble	0.9	1.1	tCK
tRPST	Read postamble	0.4	0.6	tCK
tDQSK	DQS output access time from CK/CK_n	-450	+450	ps
tDQSQ	DQS-DQ skew for DQS and associated DQ signals	-	300	ps
tQH	DQ/DQS output hold time from DQS	min(tCL, tCH) - 400ps	-	ps
WL	Write command to DQS associated clock edge	RL-1		tCK

Table 39-3. Meaning of the parameter in Fig.1-2 and Fig.1-3

### 39.4.7.1 mobile DDR Read/Write Access Timing

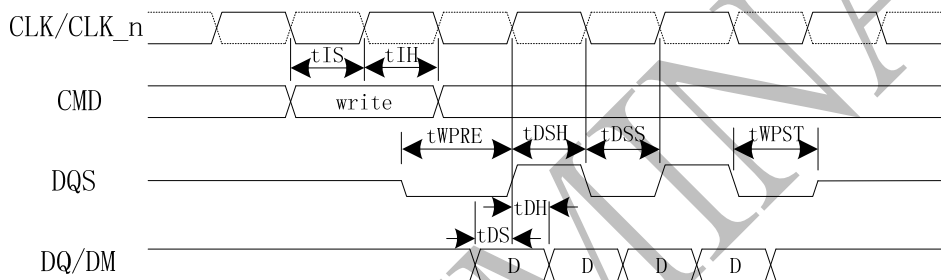


Fig. 30-6 Mobile DDR Burst write operation waveform: BL = 4

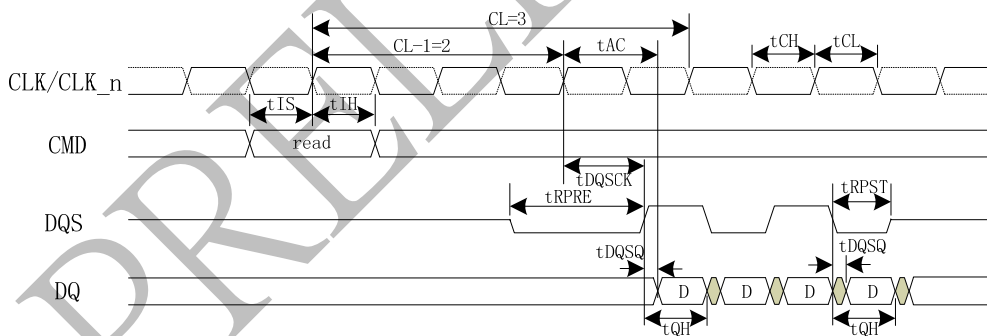


Fig. 30-7 Mobile DDR Burst read operation waveform: CL = 3, BL = 4

Parameter	Description	LPDDR-400		unit
		min	max	
tCH	CK HIGH pulse width	0.48	0.52	tCK
tCL	CK LOW pulse width	0.48	0.52	tCK
tDS	DQ and DM input setup time	fast slew rate	1.1	ns
		slow slew rate	1.2	
tDH	DQ and DM input hold time	fast slew rate	1.1	ns
		slow slew rate	1.2	
tDSS	DQS falling edge to CK setup time	0.2	-	tCK
tDSH	DQS falling edge hold time from CK	0.2	-	tCK



tIS	Address and control input setup time	fast slew rate	1.5	-	ns
		slow slew rate	1.7	-	
tIH	Address and control input hold time	fast slew rate	1.5	-	ns
		slow slew rate	1.7	-	
tWPRE	Write preamble		0.25	-	tCK
tWPST	Write postamble		0.4	0.6	tCK
tRPRE	Read preamble	CL = 2	0.5	1.1	tCK
		CL = 3	0.9	1.1	
tRPST	Read postamble		0.4	0.6	tCK
tDQSCK	DQS output access time from CK/CK_n		2.0	7.0	ns
tDQSQ	DQS-DQ skew for DQS and associated DQ signals		-	0.7	ns
tQH	DQ/DQS output hold time from DQS		min(tCL, tCH) - 1ns	-	ns
tAC	DQ output access time from CK/CK_n		2.0	7.0	ns

## Chapter 31 Hardware Information

### 31.1 Oscillator Connection

RK281x will use two oscillators, one is for input of three on-chip PLLs , for USB OTG PHY, and for I2S main clock, which should be 24MHz, another is for RTC function, which should be 32.768KHz. The design for oscillator pad has been optimized for stability and minimum jitter, and characterized to allow a variation of 4pF to 18pF on both XI and XO pins for crystal stability. In the Fig. 39-1 , the variation range for C value is 4pF to 18pF.

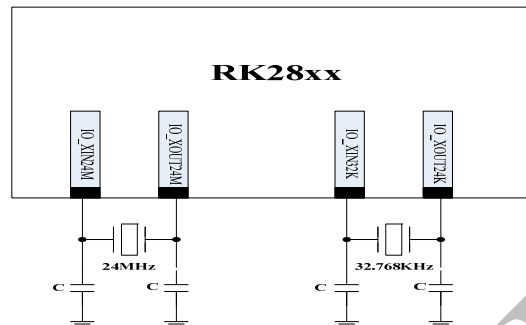


Fig. 31-1 RK281x external oscillator connection diagram

### 31.2 USB PHY Connection

USB2.0 OTG PHY is used in RK281x for USB host, USB device and otg functions. The following figure shows external connection for USB PHY interface.

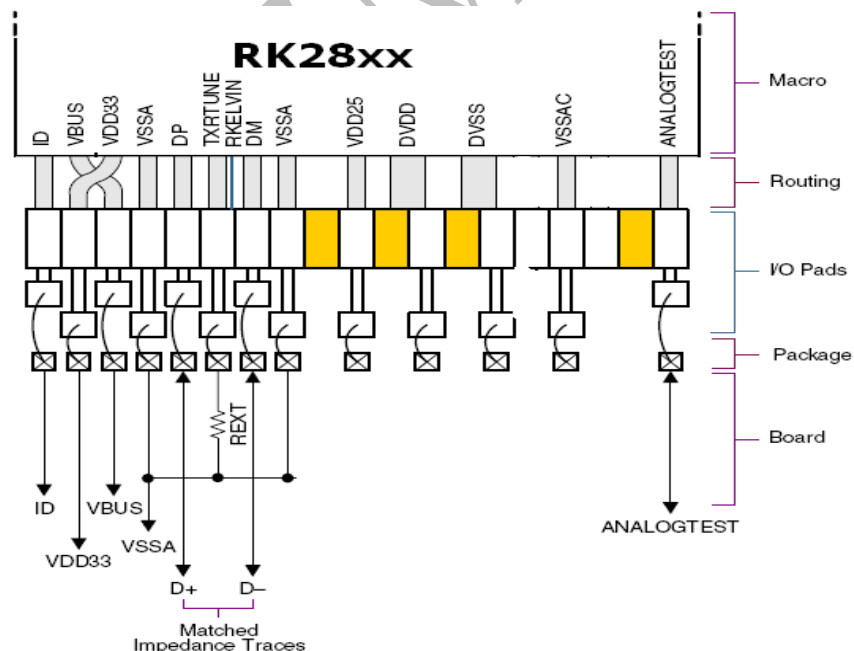


Fig. 31-2 RK281x USB PHY connection diagram

In the above diagram, some parameters and its variant will be shown in the following table.

External resistor (REXT)	44.2 $\Omega$ ( $\pm 1\%$ )
Analog power supplies	3.3 V (+ 10%, – 10%) at the macro pins with respect to VSSA and VSSAC 2.5 V (+ 10%, – 10%) at the macro pins with respect to VSSA and VSSAC
Digital power supply	1.2 V (+ 10%, – 12.5%) at the macro pins with respect to DVSS
Junction temperature	–40° C through +125° C

### 31.3 Power up Sequence for power supply

For IO and core power supply of RK281x , there are no power sequence requirements, since IO is 3-state when core power is not valid .

### 31.4 Power on reset Descriptions

The following figure shows power-on-reset sequence and relative clock behavior. When npor (power-on-reset) is released after stabilization of oscillator clock xin24m. After about T1 timing length, power supply for on-chip PLLs will be in stable state and pll\_rstn (internal reset signal for PLL) is released. Then after (T2-T1) timing length, chip\_rstn (internal reset signal for chip logic) is released. Then clock for IP module inside chip will be valid . After about 15 clocks , ip\_rstn (internal reset signal for all IPs) will be released, which can meet some special requirements for some IPs , " reset signal will be kept valid no less than 15 clock cycles" .

Notes : T1 is about 5us ; T2 is about 139us

Another, RK281x can filter out 5 clock cycles for low pulse of npor, the clock cycle is xin24m clock, so about 208ns low pulse of npor will not be recognized as valid power-on-reset signal for RK281x.

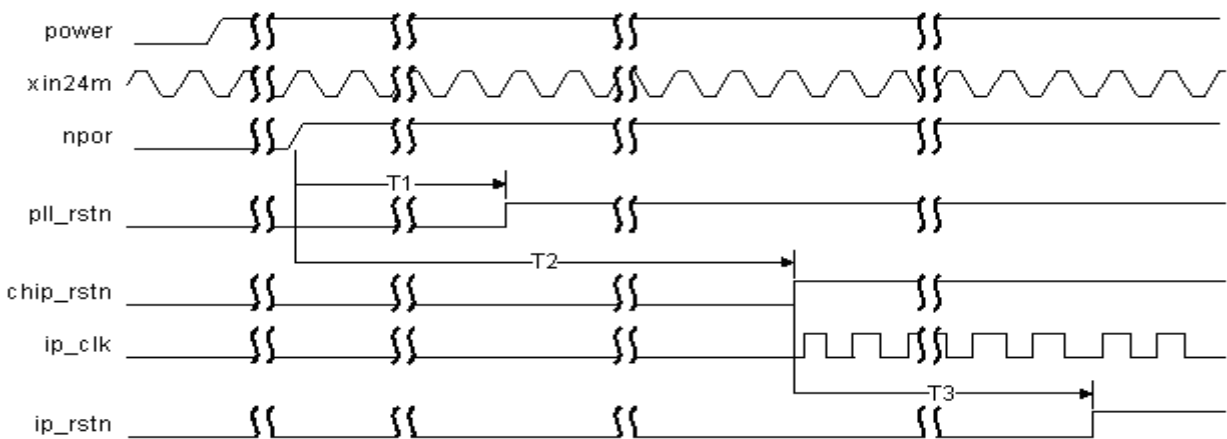


Fig. 31-3 RK281x reset sequence timing waveform

## Chapter 32 Electrical Specification

### 32.1 Recommended Operating Conditions

Symbol	Parameters	Min	Typ	Max	Unit
VDD	Power Supply Voltage (Digital Core)	1.08	1.2	1.32	V
VCCIO	Power Supply Voltage (Digital IO)	2.97	3.3	3.63	V
GND	Ground Voltage (Digital Core and IO)	N/A	0	N/A	V
VDDSDR	IO Power Supply Voltage (SDRAM IO)	2.97	3.3	3.63	V
	IO Power Supply Voltage (Mobile SDRAM IO)	1.62	1.8	1.98	V
VSSSDR	Ground Voltage (SDRAM IO)	N/A	0	N/A	V
VDDA_xPLL	Power Supply Voltage (PLL)	1.08	1.2	1.32	V
VSSA_xPLL	Ground Voltage (PLL)	N/A	0	N/A	V
AVDD25_USB	Power Supply Voltage (USB Analog part)	2.25	2.5	2.75	V
AVDD33_USB	Power Supply Voltage (USB Analog part)	2.97	3.3	3.63	V
AVSS_USB	Ground Voltage (USB Analog part)	N/A	0	N/A	V
DVDD_USB	Power Supply Voltage (USB digital part)	1.08	1.2	1.32	V
DVSS_USB	Ground Voltage (USB digital part)	N/A	0	N/A	V
VDDA_ADC	Power Supply Voltage (SAR-ADC Analog)	2.8	3	3.6	V
VSSA_ADC	Ground Voltage (SAR-ADC Analog)	0	0	0	V
XIN24M	PLL Input clock frequency	N/A	24	N/A	MHz
T	Operating Temperature	-10	25	60	0C

### 32.2 DC Characteristics

Symbol	Parameters	Min	Typ	Max	Unit
Vil	Input Low Voltage	-0.3	N/A	0.3*VCCIO	V
Vih	Input High Voltage	0.7*VCCIO	N/A	VCCIO+0.3	V
Vol	Output Low Voltage			0.4	V
Voh	Output High Voltage	VCCIO-0.4		3.6	V

### 32.3 Absolute Maximum Range

Symbol	Parameters	Min	Max	Unit
Vvdd	Core supply voltage range	-0.5	1.32	V
Vvccio	I/O supply voltage range	-0.5	3.6	V
Vpad	Voltage range at digital IO	-0.5	VCCIO+0.5	V
Vanalog	Voltage range at analog IO	0	VDDanalog(max)	V

## Appendix A – ARM926EJS16K16K

TBD

PRELIMINARY

## Appendix B – DSP System

TBD

PRELIMINARY