Jz4725

Multimedia Application Processor

Programming Manual

Revision: 1.0
Date: May 2009

北京君正集成电路有限公司
Ingenic Semiconductor Co. Ltd

Jz4725 Multimedia Application Processor

Programming Manual

Copyright © Ingenic Semiconductor Co. Ltd 2007. All rights reserved.

Release history

Date	Revision	Change
May 2009	1.0	First release

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenicfor the current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Room 108, Building A, Information Center, Zhongguancun Software Park
8 Dongbeiwang West Road, Haidian District, Beijing, China,
Tel: 86-10-82826661
Fax: 86-10-82825845
Http: //www.ingenic.cn

CONTENTS

1 Overview 1
1.1 Block Diagram 2
1.2 Features 3
1.2.1 CPU core 3
1.2.2 Multimedia support 3
1.2.3 Memory sub-system 3
1.2.4 Clock generation and power management 4
1.2.5 On-chip peripherals 4
1.3 Characteristic 6
2 CPU Core 7
3 External Memory Controller 9
3.1 Overview 9
3.2 Pin Description 10
3.3 Physical Address Space Map 11
3.4 Static Memory Interface 13
3.4.1 Register Description 14
3.4.2 Example of Connection 19
3.4.3 Basic Interface 21
3.4.4 Byte Control 25
3.4.5 Burst ROM Interface 28
3.5 NAND Flash Interface 29
3.5.1 Register Description 29
3.5.2 NAND Flash Boot Loader 36
3.5.3 NAND Flash Operation 37
3.6 SDRAM Interface 41
3.6.1 Register Description 42
3.6.2 Refresh Time Constant Register (RTCOR) 49
3.6.3 Example of Connection 51
3.6.4 Address Multiplexing 53
3.6.5 SDRAM Command 56
3.6.6 SDRAM Timing 57
3.6.7 Power-Down Mode 71
3.6.8 Refreshing 72
3.6.9 Initialize Sequence 76
3.7 Bus Control Register (BCR) 80
4 DMA Controller 82

4.1 Features 82

4.2 Register Descriptions 83

4.2.1 DMA Source Address (DSAn, n = 0 ~ 5) 84

4.2.2 DMA Target Address (DTAn, n = 0 ~ 5) 84

4.2.3 DMA Transfer Count (DTCn, n = 0 ~ 5) 84

4.2.4 DMA Request Types (DRTn, n = 0 ~ 5) 85

4.2.5 DMA Channel Control/Status (DCSn, n = 0 ~ 5) 86

4.2.6 DMA Channel Command (DCMn, n = 0 ~ 5) 87

4.2.7 DMA Descriptor Address (DDAn, n = 0 ~ 5) 88

4.2.8 DMA Control 88

4.2.9 DMA Doorbell (DDR) 89

4.2.10 DMA Interrupt Pending (DIRQP) 90

4.3 DMA manipulation 91

4.3.1 Descriptor Transfer 91

4.3.2 No-Descriptor Transfer 94

4.4 DMA Requests 94

4.4.1 Auto Request 94

4.4.2 On-Chip Peripheral Request 94

4.5 DMA Transfer Modes 95

4.5.1 Single Mode 95

4.5.2 Block Mode 95

4.6 Channel Priorities 95

4.6.1 Fixed Mode 95

4.6.2 Round Robin Mode 95

4.7 Examples 96

4.7.1 Memory-to-memory auto request No-Descriptor Transfer 96

5 Clock Reset and Power Controller 97

5.1 Overview 97

5.2 Clock Generation UNIT 98

5.2.1 Pin Description 99

5.2.2 CGU Block Diagram 100

5.2.3 Clock Overview 101

5.2.4 CGU Registers 102

5.2.5 PLL Operation 109

5.2.6 Main Clock Division Change Sequence 110

5.2.7 Change Other Clock Frequencies 111

5.2.8 Change Clock Source Selection 111

5.2.9 EXCLK Oscillator 112

5.3 Power Manager 113

5.3.1 Low-Power Modes and Function 113

5.3.2 Register Description 114

5.3.3 Doze Mode 118

5.3.4 IDLE Mode 118
5.3.5 SLEEP Mode 119
5.4 Reset Control Module 119
5.4.1 Register Description 119
5.4.2 Power On Reset 120
5.4.3 WDT Reset 120

6 Real-Time Clock (RTC).................... 121

6.1 Overview 121
6.1.1 Features 121
6.1.2 Signal Descriptions 121
6.2 Register Description 123
6.2.1 RTC Control Register (RTCCR) 124
6.2.2 RTC Second Register (RTCSR) 126
6.2.3 RTC Second Alarm Register (RTCSAR) 127
6.2.4 RTC Regulator Register (RTCGR) 128
6.2.5 Hibernate Control Register (HCR) 129
6.2.6 HIBERNATE mode Wakeup Filter Counter Register (HWFCR) 130
6.2.7 Hibernate Reset Counter Register (HRCR) 131
6.2.8 HIBERNATE Wakeup Control Register (HWCR) 132
6.2.9 HIBERNATE Wakeup Status Register (HWRSR) 133
6.2.10 Hibernate Scratch Pattern Register (HSPR) 134
6.3 Time Regulation 135
6.3.1 HIBERNATE Mode 136

7 Interrupt Controller.................... 137

7.1 Overview 137
7.2 Register Description 138
7.2.1 Interrupt Controller Source Register (ICSR) 138
7.2.2 Interrupt Controller Mask Register (ICMR) 138
7.2.3 Interrupt Controller Mask Set Register (ICMSR) 139
7.2.4 Interrupt Controller Mask Clear Register (ICMCR) 139
7.2.5 Interrupt Controller Pending Register (ICPR) 140
7.3 Software Considerations 141

8 Timer/Counter Unit 142

8.1 Overview 142
8.2 Pin Description 143
8.3 Register Description 144
8.3.1 Timer Control Register (TCSR) 146
8.3.2 Timer Data FULL Register (TDFR) 147
8.3.3 Timer Data HALF Register (TDHR) 148
8.3.4 Timer Counter (TCNT) 148

8.3.5 Timer Counter Enable Register (TER) 148
8.3.6 Timer Counter Enable Set Register (TESR) 149
8.3.7 Timer Counter Enable Clear Register (TECR) 150
8.3.8 Timer Flag Register (TFR) 151
8.3.9 Timer Flag Set Register (TFSR) 152
8.3.10 Timer Flag Clear Register (TFCR) 153
8.3.11 Timer Mast Register (TMR) 153
8.3.12 Timer Mask Set Register (TMSR) 154
8.3.13 Timer Mask Clear Register (TMCR) 154
8.3.14 Timer Stop Register (TSR) 155
8.3.15 Timer Stop Set Register (TSSR) 156
8.3.16 Timer Stop Clear Register (TSCR) 157
8.4 Operation 158
8.4.1 Basic Operation 158
8.4.2 Disable and Shutdown Operation 159
8.4.3 Pulse Width Modulator (PWM) 159
8.4.4 The flow of using TCU5 159

9 Watchdog Timer 161

9.1 Overview 161
9.2 Register Description 162
9.2.1 Watchdog Control Register (TCSR) 162
9.2.2 Watchdog Enable Register (TCER) 163
9.2.3 Watchdog Timer Data Register (TDR) 163
9.2.4 Watchdog Timer Counter (TCNT) 164
9.3 Watchdog Timer Function 164

10 General-Purpose I/O Ports 165

10.1 Overview 165
10.2 Register Description 172
10.2.1 PORT PIN Level Register (PAPIN, PBPIN, PCPIN, PDPIN) 175
10.2.2 PORT Data Register (PADAT, PBDAT, PCDAT, PDDAT) 175
10.2.3 PORT Data Set Register (PADATS, PBDATS, PCDATS, PDDATS) 176
10.2.4 PORT Data Clear Register (PADATC, PBDATC, PCDATC, PDDATC) 176
10.2.5 PORT Mask Register (PAIM, PBIM, PCIM, PDIM) 177
10.2.6 PORT Mask Set Register (PAIMS, PBIMS, PCIMS, PDIMS) 177
10.2.7 PORT Mask Clear Register (PAIMC, GBPIMC, PCIMC, PDIMC) 178
10.2.8 PORT PULL Disable Register (PAPE, PBPE, PCPE, PDPE) 178
10.2.9 PORT PULL Set Register (PAPES, PBPES, PCPES, PDPES) 179
10.2.10 PORT PULL Clear Register (PAPEC, PBPEC, PCPEC, PDPEC) 179
10.2.11 PORT Function Register (PAFUN, PBFUN, PCFUN, PDFUN) 180
10.2.12 PORT Function Set Register (PAFUNS, PBFUNS, PCFUNS, PDFUNS) 180
10.2.13 PORT Function Clear Register (PAFUNC, PBFUNC, PCFUNC, PDFUNC) 181

10.2.14 PORT Select Register (PASEL, PBSEL, PCFSEL, PDSEL) 181
10.2.15 PORT Select Set Register (PASELS, PBSELS, PCSELS, PDSELS) 182
10.2.16 PORT Select Clear Register (PASELC, PBSELC, PCSELC, PDSELC) 182
10.2.17 PORT Direction Register (PADIR, PBDIR, PCDIR, PDDIR) 182
10.2.18 PORT Direction Set Register (PADIRS, PBDIRS, PCDIRS, PDDIRS) 183
10.2.19 PORT Direction Clear Register (PADIRC, PBDIRC, PCDIRC, PDDIRC) 184
10.2.20 PORT Trigger Register 0, 1, 2 and 3 (PATRG, PBTRG, PCTRG, PDTRG) 184
10.2.21 PORT Trigger Set Register (PATRGS, PBTRGS, PCTRGS, PDTRGS) 185
10.2.22 PORT Trigger Clear Register (PATRGC, PBTRGC, PCTRGC, PDTRGC) 185
10.2.23 PORT FLAG Register (PAFLG, PBFLG, PCFLG, PDFLG) 185
10.2.24 PORT FLAG Clear Register (PAFLGC, PBFLGC, PCFLGC, PDFLGC) 186
10.3 Program Guide 187
10.3.1 GPIO Function Guide 187
10.3.2 Alternate Function Guide 187
10.3.3 Interrupt Function Guide 187
10.3.4 Disable Interrupt Function Guide 188

11 LCD Controller 189

11.1 Overview 189
11.2 Pin Description 190
11.3 Block Diagram 191
11.4 LCD Display Timing 191
11.5 TV Encoder Timing 192
11.6 Register Description 194
11.6.1 Configure Register (LCDCFG) 195
11.6.2 Vertical Synchronize Register (LCDVSYNC) 197
11.6.3 Horizontal Synchronize Register (LCDHSYNC) 197
11.6.4 Virtual Area Setting (LCDVAT) 197
11.6.5 Display Area Horizontal Start/End Point (LCDDAH) 198
11.6.6 Display Area Vertical Start/End Point (LCDDAV) 198
11.6.7 PS Signal Setting (LCDPS) 199
11.6.8 CLS Signal Setting (LCDCLS) 199
11.6.9 SPL Signal Setting (LCDSPL) 200
11.6.10 REV Signal Setting (LCDREV) 200
11.6.11 Control Register (LCDCTRL) 200
11.6.12 Status Register (LCDSTATE) 202
11.6.13 Interrupt ID Register (LCDIID) 202
11.6.14 Descriptor Address Register0, 1 (LCDDA0, 1) 203
11.6.15 Source Address Register0, 1 (LCDSA0, 1) 204
11.6.16 Frame ID Register0 (LCDFID0,1) 204
11.6.17 DMA Command Register0, 1 (LCDCMD0, 1) 205
11.7 LCD Controller Pin Mapping 206
11.7.1 TFT and CCIR656 Pin Mapping 206

11.7.2 Single STN Pin Mapping....208
11.7.3 Dual Panel STN Pin Mapping....209
11.8 Display Timing....210
11.8.1 General 16-bit and 18-bit TFT Timing....210
11.8.2 8-bit Serial TFT Timing....211
11.8.3 Special TFT Timing....212
11.9 Format of Palette....214
11.9.1 STN....214
11.9.2 TFT....214
11.10 Format of Frame Buffer....215
11.10.1 16bpp....215
11.10.2 18bpp....215
11.10.3 24bpp....215
11.11 Format of Data Pin Utilization....215
11.11.1 Mono STN....215
11.11.2 Color STN....216
11.11.3 18-bit Parallel TFT....216
11.11.4 16-bit Parallel TFT....216
11.11.5 8-bit Serial TFT (24bpp)....216
11.12 LCD Controller Operation....216
11.12.1 Set LCD Controller Device Clock and Pixel Clock....216
11.12.2 Enabling the Controller....217
11.12.3 Disabling the Controller....217
11.12.4 Resetting the Controller....218
11.12.5 Frame Buffer & Palette Buffer....218

12 Smart LCD Controller....219

12.1 Overview....219
12.2 Structure....219
12.3 Pin Description....220
12.4 Register Description....220
12.4.1 SLCD Configure Register (MCFG)....221
12.4.2 SLCD Control Register (MCTRL)....222
12.4.3 SLCD Status Register (MSTATE)....223
12.4.4 SLCD Data Register (MDATA)....223
12.4.5 SLCD FIFO (MFIFO)....224
12.5 System Memory Format....225
12.5.1 Data format....225
12.5.2 Command Format....226
12.6 Transfer Mode....227
12.6.1 DMA Transfer Mode....227
12.6.2 Register Transfer Mode....227
12.7 Timing....227

12.7.1 Parallel Timing 227
12.7.2 Serial Timing 228
12.8 Operation Guide 228
12.8.1 DMA Operation 228
12.8.2 Register Operation 228

13 AC97/I2S Controller 230

13.1 Overview 230
13.1.1 Block Diagram 231
13.1.2 Features 231
13.1.3 Interface Diagram 232
13.1.4 Signal Descriptions 234
13.1.5 RESET# / SYS_CLK Pin 234
13.1.6 BIT_CLK Pin 234
13.1.7 SYNC Pin 234
13.1.8 SDATA_OUT Pin 235
13.1.9 SDATA_IN Pin 235
13.2 Register Descriptions 236
13.2.1 AIC Configuration Register (AICFR) 238
13.2.2 AIC Common Control Register (AICCR) 240
13.2.3 AIC AC-link Control Register 1 (ACCR1) 243
13.2.4 AIC AC-link Control Register 2 (ACCR2) 244
13.2.5 AIC I2S/MSB-justified Control Register (I2SCR) 246
13.2.6 AIC Controller FIFO Status Register (AICSR) 247
13.2.7 AIC AC-link Status Register (ACSR) 249
13.2.8 AIC I2S/MSB-justified Status Register (I2SSR) 251
13.2.9 AIC AC97 CODEC Command Address Register (ACCAR) & Data Register (ACCDR)252
13.2.10 AIC AC97 CODEC Status Address Register (ACSAR) & Data Register (ACSDR). 253
13.2.11 AIC I2S/MSB-justified Clock Divider Register (I2SDIV) 254
13.2.12 AIC FIFO Data Port Register (AICDR) 255
13.3 Serial Interface Protocol 256
13.3.1 AC-link serial data format 256
13.3.2 I2S and MSB-justified serial audio format 257
13.3.3 Audio sample data placement in SDATA_IN/SDATA_OUT 259
13.4 Operation 260
13.4.1 Initialization 261
13.4.2 AC ’97 CODEC Power Down 262
13.4.3 Cold and Warm AC ’97 CODEC Reset 262
13.4.4 External CODEC Registers Access Operation 264
13.4.5 Audio Replay 265
13.4.6 Audio Record 266
13.4.7 FIFOs operation 267
13.4.8 Data Flow Control 269

13.4.9 Serial Audio Clocks and Sampling Frequencies 270
13.4.10 Interrupts 274

14 Internal CODEC 275

14.1 Overview 275
14.1.1 Features 275
14.1.2 Signal Descriptions 275
14.1.3 Block Diagram 276
14.2 Register Descriptions 277
14.2.1 CODEC Control Register 1 (CDCCR1) 278
14.2.2 CODEC Control Register 2 (CDCCR2) 280
14.3 Operation 282
14.3.1 Initialization 282
14.3.2 CODEC controlling and typical operations 283
14.3.3 Power saving 285
14.3.4 Pop noise and the reduction of it 286
14.4 Timing parameters 293
14.5 AC & DC parameters 293

15 SAR A/D Controller 294

15.1 Overview 294
15.2 Pin Description 294
15.3 Register Description 295
15.3.1 ADC Enable Register (ADENA) 296
15.3.2 ADC Configure Register (ADCFG) 297
15.3.3 ADC Control Register (ADCTRL) 299
15.3.4 ADC Status Register (ADSTATE) 300
15.3.5 ADC Same Point Time Register (ADSAME) 301
15.3.6 ADC Wait Pen Down Time Register (ADWAIT) 301
15.3.7 ADC Touch Screen Data Register (ADTCH) 301
15.3.8 ADC PBAT Data Register (ADBDAT) 304
15.3.9 ADC SADCIN Data Register (ADSDAT) 305
15.3.10 ADC IP DATA PIN (ADDPIN) 305
15.4 SAR A/D Controller Guide 306
15.4.1 Single Operation (only used as a test mode to check the channel function) 306
15.4.2 A simple Touch Screen Operation 306
15.4.3 PBAT Sample Operation 307
15.4.4 SADCIN Sample Operation 307
15.4.5 Use TSC to support keypad 308

16 MultiMediaCard/Secure Digital Controller 312

16.1 Overview 312
16.2 Block Diagram 313

16.3 MMC/SD Controller Signal I/O Description 314
16.4 Register Description 316
  16.4.1 Start/stop MMC/SD clock Register (MSC_STRPCL) 316
  16.4.2 MSC Status Register (MSC_STAT) 317
  16.4.3 MSC Clock Rate Register (MSC_CLKRT) 319
  16.4.4 MMC/SD Command and Data Control Register (MSC_CMDAT) 320
  16.4.5 MMC/SD Response Time Out Register (MSC_RESTO) 321
  16.4.6 MMC/SD Read Time Out Register (MSC_RDTO) 322
  16.4.7 MMC/SD Block Length Register (MSC_BLKLEN) 322
  16.4.8 MSC/SD Number of Block Register (MSC_NOB) 322
  16.4.9 MMC/SD Number of Successfully-transferred Blocks Register (MSC_SNOB) 322
  16.4.10 MMC/SD Interrupt Mask Register (MSC_IMASK) 323
  16.4.11 MMC/SD Interrupt Register (MSC_IREG) 324
  16.4.12 MMC/SD Command Index Register (MSC_CMD) 325
  16.4.13 MMC/SD Command Argument Register (MSC_ARG) 325
  16.4.14 MMC/SD Response FIFO Register (MSC_RES) 325
  16.4.15 MMC/SD Receive Data FIFO Register (MSC_RXFIFO) 326
  16.4.16 MMC/SD Transmit Data FIFO Register (MSC_TXFIFO) 326
16.5 MMC/SD Functional Description 327
  16.5.1 MSC Reset 327
  16.5.2 MSC Card Reset 327
  16.5.3 Voltage Validation 327
  16.5.4 Card Registry 328
  16.5.5 Card Access 329
  16.5.6 Protection Management 330
  16.5.7 Card Status 334
  16.5.8 SD Status 337
  16.5.9 SDIO 338
  16.5.10 Clock Control 339
  16.5.11 Application Specified Command Handling 339
16.6 MMC/SD Controller Operation 341
  16.6.1 Data FIFOs 341
  16.6.2 DMA and Program I/O 342
  16.6.3 Start and Stop clock 342
  16.6.4 Software Reset 343
  16.6.5 Voltage Validation and Card Registry 343
  16.6.6 Single Data Block Write 345
  16.6.7 Single Block Read 346
  16.6.8 Multiple Block Write 346
  16.6.9 Multiple Block Read 347
  16.6.10 Stream Write (MMC) 348
  16.6.11 Stream Read (MMC) 348
  16.6.12 Erase, Select/Deselect and Stop 349

16.6.13 SDIO Suspend/Resume....349
16.6.14 SDIO ReadWait....350
16.6.15 Operation and Interrupt....350

17 I2C Bus Interface....352

17.1 Overview....352
17.2 Pin Description....353
17.3 Register Description....354
17.3.1 Data Register (I2CDR)....354
17.3.2 Control Register (I2CCCR)....354
17.3.3 Status Register (I2CSR)....355
17.3.4 Clock Generator Register (I2CGR)....355
17.4 I^{2}C-Bus Protocol....356
17.4.1 Bit Transfer....356
17.4.2 Data Validity....356
17.4.3 START and STOP Conditions....356
17.4.4 Byte Format....356
17.4.5 Data Transfer Format....358
17.5 I2C Operation....362
17.5.1 I2C Initialization....362
17.5.2 Write Operation....363
17.5.3 Read Operation....364

18 UART Interface....366

18.1 Overview....366
18.2 Pin Description....367
18.3 Register Description....367
18.3.1 UART Receive Buffer Register (URBR)....368
18.3.2 UART Transmit Hold Register (UTHR)....369
18.3.3 UART Divisor Latch Low/High Register (UDLLR / UDLHR)....369
18.3.4 UART Interrupt Enable Register (UIER)....370
18.3.5 UART Interrupt Identification Register (UIIR)....371
18.3.6 UART FIFO Control Register (UFCR)....372
18.3.7 UART Line Control Register (ULCR)....373
18.3.8 UART Line Status Register (ULSR)....374
18.3.9 UART Modem Control Register (UMCR)....376
18.3.10 UART Modem Status Register (UMSR)....377
18.3.11 UART Scratchpad Register....378
18.3.12 Infrared Selection Register (ISR)....378
18.3.13 Uart M Register (UMR)....379
18.3.14 Uart Add Cycle Register(UACR)....380
18.4 Operation....380
18.4.1 UART Configuration....380

18.4.2 Data Transmission 380
18.4.3 Data Reception 381
18.4.4 Receive Error Handling 381
18.4.5 Modem Transfer 382
18.4.6 DMA Transfer 382
18.4.7 Slow IrDA Asynchronous Interface 382
18.4.8 For any frequency clock to use the Uart 383

19 Image Process Unit 385

19.1 Overview 385
19.2 Features 385
19.3 Block Diagram 385
19.4 Data flow 386
19.4.1 Input Data 386
19.4.2 Output Data 386
19.4.3 Resize Coefficients LUT 386
19.5 Register definition 387
19.5.1 IPU Control Register 387
19.5.2 IPU Status Register 387
19.5.3 Data Format Register 388
19.5.4 Input Y Data Address Register 388
19.5.5 Input U Data Address Register 389
19.5.6 Input V Data Address Register 389
19.5.7 Input Geometric Size Register 389
19.5.8 Input Y Data Line Stride Register 390
19.5.9 Input UV Data Line Stride Register 390
19.5.10 Output Frame Start Address Register 390
19.5.11 Output Geometric Size Register 391
19.5.12 Output Data Line Stride Register 391
19.5.13 Resize Coefficients Table Index Register 391
19.5.14 CSC C0 Coefficient Register 392
19.5.15 CSC C1 Coefficient Register 392
19.5.16 CSC C2 Coefficient Register 393
19.5.17 CSC C3 Coefficient Register 393
19.5.18 CSC C4 Coefficient Register 395
19.5.19 Horizontal Resize Coefficients Look Up Table Register group 395
19.5.20 Vertical Resize Coefficients Look Up Table Register group 397
19.6 Calculation for Resized width and height 398
19.7 IPU Initialization Flow 399

20 Jz4725 Boot ROM Specification 405

20.1 Boot Select 405
20.2 Boot Sequence 405

20.3 NAND Boot Specification..405
20.4 USB Boot Specification 407

TABLES

TABLE 2-1 JzRISC CORE FEATURES 7
TABLE 3-1 EMC PIN DESCRIPTION 10
TABLE 3-2 PHYSICAL ADDRESS SPACE MAP 12
TABLE 3-3 DEFAULT CONFIGURATION OF EMC CHIP SELECT SIGNALS 12
TABLE 3-4 STATIC MEMORY INTERFACE REGISTERS 14
TABLE 3-5 NAND FLASH INTERFACE REGISTERS 29
TABLE 3-6 512-BYTE ECC PARITY CODE ASSIGNMENT TABLE FOR 8-BIT NAND FLASH 38
TABLE 3-7 256-BYTE ECC PARITY CODE ASSIGNMENT TABLE FOR 8-BIT NAND FLASH 38
TABLE 3-8 256-HALFWORD ECC PARITY CODE ASSIGNMENT TABLE FOR 16-BIT NAND FLASH 38
TABLE 3-9 SDRAM REGISTERS 42
TABLE 3-10 SDRAM ADDRESS MULTIPLEXING (32-BIT DATA WIDTH) *[4] 54
TABLE 3-11 SDRAM ADDRESS MULTIPLEXING (16-BIT DATA WIDTH) *[4] 55
TABLE 3-12 SDRAM COMMAND ENCODING (NOTES: 1) 56
TABLE 3-13 SDRAM MODE REGISTER SETTING ADDRESS EXAMPLE (32-BIT) 76
TABLE 3-14 SDRAM MODE REGISTER SETTING ADDRESS EXAMPLE (16-BIT) 76
TABLE 3-15 BOOT CONFIGURATION 80
TABLE 4-1 DMAC REGISTERS 83
TABLE 4-2 TRANSFER REQUEST TYPES 85
TABLE 4-3 DETECTION INTERVAL LENGTH 88
TABLE 4-4 DESCRIPTOR STRUCTURE 92
TABLE 4-5 RELATIONSHIP AMONG DMA TRANSFER CONNECTION, REQUEST MODE AND TRANSFER MODE 96
TABLE 5-1 CGU REGISTERS CONFIGURATION 102
TABLE 5-2 TYPICAL CL AND THE CORRESPONDING MAXIMUM ESR 112
TABLE 5-3 POWER/RESET MANAGEMENT CONTROLLER REGISTERS CONFIGURATION 114
TABLE 6-1 REGISTERS FOR REAL TIME CLOCK 123
TABLE 6-2 REGISTERS FOR HIBERNATING MODE 123
TABLE 7-1 INTC REGISTER 138
TABLE 8-1 PWM PINS DESCRIPTION 143
TABLE 10-1 GPIO PORT A SUMMARY 167
TABLE 10-2 GPIO REGISTERS 172
TABLE 11-1 LCD CONTROLLER PINS DESCRIPTION 190
TABLE 11-2 LCD CONTROLLER REGISTERS DESCRIPTION 194
TABLE 12-1 SLCD PINS DESCRIPTION 220
TABLE 13-1 AIC PINS DESCRIPTION 234
TABLE 13-2 AIC REGISTERS DESCRIPTION 236
TABLE 13-3 SAMPLE DATA BIT RELATE TO SDATA_IN/SDATA_OUT BIT 259
TABLE 13-4 COLD AC ’97 CODEC RESET TIMING PARAMETERS 262
TABLE 13-5 WARM AC ’97 CODEC RESET TIMING PARAMETERS 263
TABLE 13-6 AUDIO SAMPLING RATE, BIT_CLK AND SYS_CLK FREQUENCIES 270
TABLE 13-7 BIT_CLK DIVIDER SETTING 271

TABLES

TABLE 13-8 APPROXIMATE COMMON MULTIPLE OF SYS_CLK FOR ALL SAMPLE RATES 272
TABLE 13-9 CPM/AIC CLOCK DIVIDER SETTING FOR VARIOUS SAMPLING RATE IF PLL = 270.64MHZ............ 272
TABLE 13-10 PLL PARAMETERS AND AUDIO SAMPLE ERRORS FOR EXCLK=12MHZ....................................... 273
TABLE 14-1 CODEC SIGNAL IO PIN DESCRIPTION 275
TABLE 14-2 INTERNAL CODEC REGISTERS DESCRIPTION 277
TABLE 14-3 CODEC SETTINGS IN VARIOUS APPLICATIONS 283
TABLE 14-4 CHARGE VREF TO THE MIDDLE VOLTAGE VM... 286
TABLE 14-5 CHARGE HPOUT TO THE MIDDLE VOLTAGE VM... 286
TABLE 14-6 CHARGE VREF AND HPOUT TO THE MIDDLE VOLTAGE VM 286
TABLE 14-7 DISCHARGE HPOUT TO 0V 287
TABLE 14-8 DISCHARGE VREF TO 0V 287
TABLE 14-9 DISCHARGE VREF AND HPOUT TO 0V 287
TABLE 15-1 SADC PINS DESCRIPTION... 294
TABLE 16-1 MMC/SD CONTROLLER SIGNAL DESCRIPTION 314
TABLE 16-2 COMMAND TOKEN FORMAT.. 314
TABLE 16-3 MMC/SD DATA TOKEN FORMAT 315
TABLE 16-4 MMC/SD CONTROLLER REGISTERS DESCRIPTION 316
TABLE 16-5 COMMAND DATA BLOCK STRUCTURE 331
TABLE 16-6 CARD STATUS DESCRIPTION 334
TABLE 16-7 SD STATUS STRUCTURE.. 337
TABLE 16-8 HOW TO STOP MULTIPLE BLOCK WRITE 346
TABLE 16-9 HOW TO STOP MULTIPLE BLOCK READ.. 347
TABLE 16-10 THE MAPPING BETWEEN COMMANDS AND STEPS 350
TABLE 17-1 SMART CARD CONTROLLER PINS DESCRIPTION.. 353
TABLE 17-2 I2C REGISTERS DESCRIPTION 354
TABLE 18-1 UART PINS DESCRIPTION 367
TABLE 18-2 UART REGISTERS DESCRIPTION 367
TABLE 18-3 UART INTERRUPT IDENTIFICATION REGISTER DESCRIPTION 372
TABLE 20-1 BOOT CONFIGURATION OF JZ4725 405
TABLE 20-2 NAND FIRST DATA BYTE DEFINITION 406
TABLE 20-3 NAND SPARE AREA DEFINITION... 406
TABLE 20-4 JZ4725 NAND BOOT SEQUENCE 407
TABLE 20-5 TRANSFER TYPES USED BY THE BOOT PROGRAM 408
TABLE 20-6 VENDOR REQUEST 0 SETUP COMMAND DATA STRUCTURE 410
TABLE 20-7 VENDOR REQUEST 1 SETUP COMMAND DATA STRUCTURE 411
TABLE 20-8 VENDOR REQUEST 2 SETUP COMMAND DATA STRUCTURE 411
TABLE 20-9 VENDOR REQUEST 3 SETUP COMMAND DATA STRUCTURE 411
TABLE 20-10 VENDOR REQUEST 4 SETUP COMMAND DATA STRUCTURE 412
TABLE 20-11 VENDOR REQUEST 5 SETUP COMMAND DATA STRUCTURE 412

FIGURES

FIGURE 1-1 JZ4725 DIAGRAM 2
FIGURE 3-1 PHYSICAL ADDRESS SPACE MAP 11
FIGURE 3-2 EXAMPLE OF 32-BIT DATA WIDTH SRAM CONNECTION 19
FIGURE 3-3 EXAMPLE OF 16-BIT DATA WIDTH SRAM CONNECTION 20
FIGURE 3-4 EXAMPLE OF 8-BIT DATA WIDTH SRAM CONNECTION 20
FIGURE 3-5 BASIC TIMING OF NORMAL MEMORY READ 22
FIGURE 3-6 BASIC TIMING OF NORMAL MEMORY WRITE 22
FIGURE 3-7 NORMAL MEMORY READ TIMING WITH WAIT (SOFTWARE WAIT ONLY) 23
FIGURE 3-8 NORMAL MEMORY WRITE TIMING WITH WAIT (SOFTWARE WAIT ONLY) 23
FIGURE 3-9 NORMAL MEMORY READ TIMING WITH WAIT (WAIT CYCLE INSERTION BY WAIT# PIN) 24
FIGURE 3-10 EXAMPLE OF 32-BIT DATA WIDTH BYTE CONTROL SRAM CONNECTION 25
FIGURE 3-11 BYTE CONTROL SRAM READ TIMING 26
FIGURE 3-12 BYTE CONTROL SRAM WRITE TIMING 27
FIGURE 3-13 BURST ROM READ TIMING (SOFTWARE WAIT ONLY) 28
FIGURE 3-14 STRUCTURE OF NAND FLASH BOOT LOADER 36
FIGURE 3-15 STATIC BANK 2 PARTITION WHEN NAND FLASH IS USED (AN EXAMPLE) 37
FIGURE 3-16 EXAMPLE OF 8-BIT NAND FLASH CONNECTION 38
FIGURE 3-17 SYNCHRONOUS DRAM MODE REGISTER CONFIGURATION 47
FIGURE 3-18 EXAMPLE OF SYNCHRONOUS DRAM CHIP CONNECTION (1) 51
FIGURE 3-19 EXAMPLE OF SYNCHRONOUS DRAM CHIP CONNECTION (2) 52
FIGURE 3-20 SYNCHRONOUS DRAM 4-BEAT BURST READ TIMING (DIFFERENT ROW) 59
FIGURE 3-21 SYNCHRONOUS DRAM 4-BEAT BURST READ TIMING (SAME ROW) 60
FIGURE 3-22 SYNCHRONOUS DRAM 4-BEAT BURST WRITE TIMING (DIFFERENT ROW) 61
FIGURE 3-23 SYNCHRONOUS DRAM 4-BEAT BURST WRITE TIMING (SAME ROW) 62
FIGURE 3-24 SYNCHRONOUS DRAM 8-BEAT BURST READ TIMING (DIFFERENT ROW) 63
FIGURE 3-25 SYNCHRONOUS DRAM 8-BEAT BURST READ TIMING (SAME ROW) 64
FIGURE 3-26 SYNCHRONOUS DRAM 8-BEAT BURST WRITE TIMING (SAME ROW) 65
FIGURE 3-27 SYNCHRONOUS DRAM 8-BEAT BURST WRITE TIMING (DIFFERENT ROW) 66
FIGURE 3-28 SYNCHRONOUS DRAM SINGLE READ TIMING (DIFFERENT ROW) 67
FIGURE 3-29 SYNCHRONOUS DRAM SINGLE READ TIMING (SAME ROW) 68
FIGURE 3-30 SYNCHRONOUS DRAM SINGLE WRITE TIMING (DIFFERENT ROW) 69
FIGURE 3-31 SYNCHRONOUS DRAM SINGLE WRITE TIMING (SAME ROW) 70
FIGURE 3-32 SDRAM POWER-DOWN MODE TIMING (CKO STOPPED) 71
FIGURE 3-33 SDRAM POWER-DOWN MODE TIMING (CLOCK SUPPLIED) 71
FIGURE 3-34 SYNCHRONOUS DRAM AUTO-REFRESH OPERATION 72
FIGURE 3-35 SYNCHRONOUS DRAM AUTO-REFRESH TIMING 73
FIGURE 3-36 SYNCHRONOUS DRAM SELF-REFRESH TIMING 75
FIGURE 3-37 SDRAM MODE REGISTER WRITE TIMING 1 (PRE-CHARGE ALL BANKS) 78
FIGURE 3-38 SDRAM MODE REGISTER WRITE TIMING 2 (MODE REGISTER SET) 79
FIGURE 4-1 DESCRIPTOR TRANSFER FLOW 93

FIGURES

FIGURE 5-1 CGU BLOCK DIAGRAM 100
FIGURE 5-2 BLOCK DIAGRAM OF PLL 109
FIGURE 5-3 OSCILLATING CIRCUIT FOR FUNDAMENTAL MODE 112
FIGURE 11-1 BLOCK DIAGRAM 191
FIGURE 11-2 GENERAL 16-BIT AND 18-BIT TFT LCD TIMING 210
FIGURE 11-3 SPECIAL TFT LCD TIMING 1 212
FIGURE 11-4 SPECIAL TFT LCD TIMING 2 213
FIGURE 13-1 AIC BLOCK DIAGRAM 231
FIGURE 13-2 INTERFACE TO THE INTERNAL I2S CODEC DIAGRAM 232
FIGURE 13-3 INTERFACE TO AN EXTERNAL AC'97 CODEC DIAGRAM 232
FIGURE 13-4 INTERFACE TO AN EXTERNAL MASTER MODE I2S/MSB-JUSTIFIED CODEC DIAGRAM 232
FIGURE 13-5 INTERFACE TO AN EXTERNAL SLAVE MODE I2S/MSB-JUSTIFIED CODEC DIAGRAM 233
FIGURE 13-6 AC-LINK AUDIO FRAME FORMAT 256
FIGURE 13-7 AC-LINK TAG PHASE, SLOT 0 FORMAT 256
FIGURE 13-8 AC-LINK DATA PHASES, SLOT 1 ~ SLOT 12 FORMAT 256
FIGURE 13-9 I2S DATA FORMAT 257
FIGURE 13-10 MSB-JUSTIFIED DATA FORMAT 257
FIGURE 13-11 COLD AC '97 CODEC RESET TIMING 262
FIGURE 13-12 WARM AC '97 CODEC RESET TIMING 263
FIGURE 13-13 TRANSMITTING/RECEIVING FIFO ACCESS VIA APB BUS 267
FIGURE 13-14 SYS_CLK, BIT_CLK AND SYNC GENERATION SCHEME 271
FIGURE 14-1 CODEC BLOCK DIAGRAM 276
FIGURE 14-2 INTERNAL CODEC WORKS WITH AIC 276
FIGURE 15-1 6X5 KEYPAD CIRCUIT 308
FIGURE 15-2 WAIT FOR PEN-DOWN (C=1100) CIRCUIT 309
FIGURE 15-3 MEASURE X-POSITION (C=0010) CIRCUIT 310
FIGURE 15-4 MEASURE Y-POSITION (C=0011) CIRCUIT 310
FIGURE 16-1 MMC/SD CONTROLLER BLOCK DIAGRAM 313
FIGURE 17-1 I2C-BUS PROTOCOL 357
FIGURE 17-2 I^{2}C-BUS PROTOCOL (CONT.) 357
FIGURE 17-3 NORMAL 7 BIT ADDRESS AFTER START CONDITION 358
FIGURE 17-4 GENERAL CALL ADDRESS AFTER START CONDITION 359
FIGURE 17-5 START BYTE AFTER START CONDITION 360
FIGURE 17-6 A MASTER-TRANSMITTER ADDRESSES A SLAVE RECEIVER WITH A 7-BIT ADDRESS 361
FIGURE 17-7 A MASTER READS THE SLAVE IMMEDIATELY AFTER THE FIRST BYTE (MASTER-RECEIVER) 361
FIGURE 17-8 I2C INITIALIZATION 362
FIGURE 17-9 I2C WRITE OPERATION FLOWCHART 363
FIGURE 17-10 I2C READ OPERATION FLOWCHART 364
FIGURE 17-11 READ OPERATION FLOWCHART (CONT.) 365
FIGURE 20-1 USB COMMUNICATION FLOW 408
FIGURE 20-2 TYPICAL PROCEDURE OF USB BOOT 410

1 Overview

Jz4725 is a multimedia application processor targeting for PMP. Incorporate the XBurst® CPU core based on leading micro-architecture technology, this processor provides high integration, high performance and low power consumption solution for embedded device.

XBurst® is a high performance and power-efficient 32-bit RISC core with 16K instruction cache and 16K data cache, operating at speed up to 360MHz. The SIMD instruction set implemented by XBurst® core, in together with the on chip Image Processing Unit, provides strong decoding capability. The memory interface supports a variety of memory types that allow flexible design requirements, include the glueless connection to SLC/MLC NAND Flash for cost sensitive applications. On-chip modules such as LCD controller, audio CODEC, 2-channel SAR-ADC and AC97/I2S controller offer designers a rich suite of peripherals for multimedia application.

1.1 Block Diagram

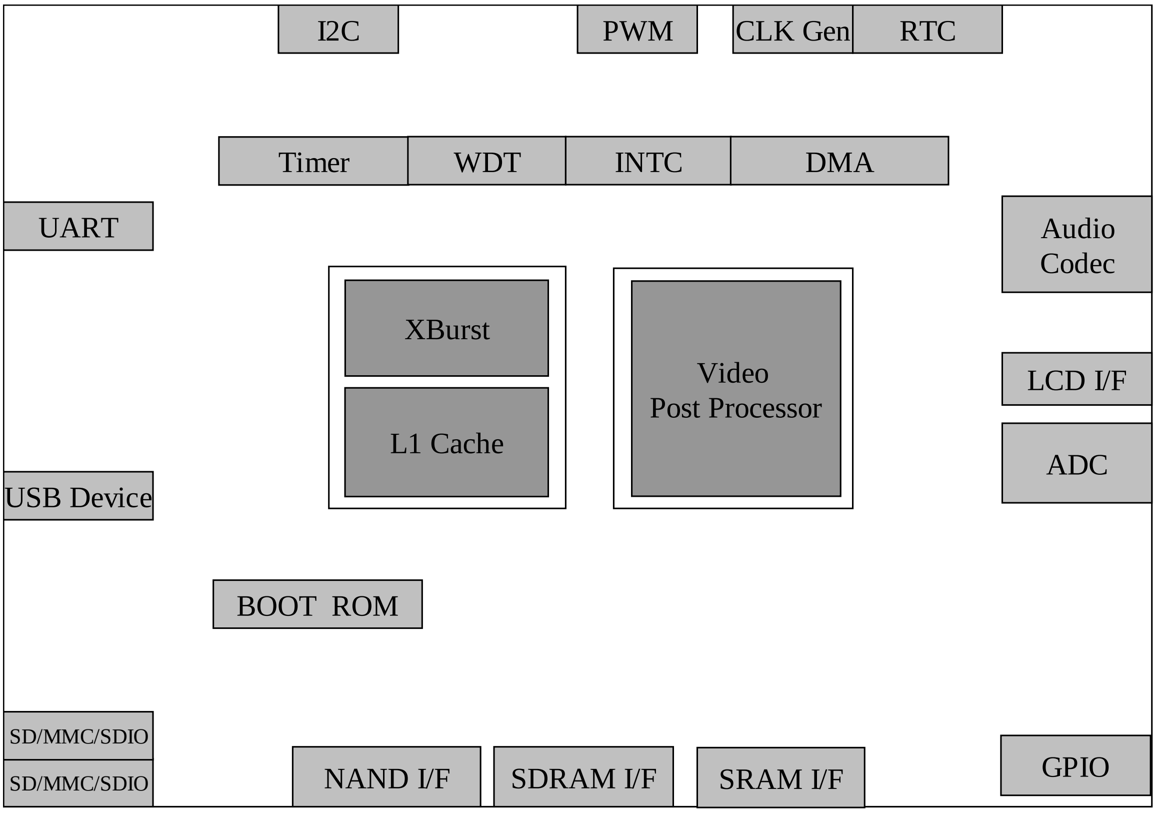

I2C
PWM
CLK Gen
RTC
Timer
WDT
INTC
DMA
UART
Audio Codec
XBurst
L1 Cache
Video Post Processor
LCD I/F
ADC
USB Device
BOOT ROM
SD/MMC/SDIO
SD/MMC/SDIO
NAND I/F
SDRAM I/F
SRAM I/F
GPIO

Figure 1-1 Jz4725 Diagram

1.2 Features

1.2.1 CPU core

- XBurst® RISC instruction set to support Linux and WinCE
- XBurst® SIMD instruction set to support multimedia acceleration
- XBurst® 8-stage pipeline micro-architecture up to 360MHz
- 16K I-Cache, 16K D-Cache
- 32-entry dual-pages joint-TLB, 4 entry Instruction TLB and 4 entry data TLB
- Smart prefetch to accelerate multimedia applications

1.2.2 Multimedia support

- IPU (Image Processing Unit)
  - Video frame resize
  - Color space conversion: 420/444/422 YUV to RGB convert

1.2.3 Memory sub-system

- NAND Flash interface
  - Support MLC NAND as well as SLC NAND
  - Support all 8-bit/16-bit NAND Flash devices regardless of density and organization
  - Hamming and Reed-Solomon Hardware ECC for error detection and correction
  - Support automatic boot up from NAND Flash devices
  - Two chip select
- Synchronous DRAM Interface
  - 1 banks with programmable size and base address
  - 16-bit data bus width
  - Multiplexes row/column addresses according to SDRAM capacity
  - Two-bank or four-bank SDRAM is supported
  - Supports auto-refresh and self-refresh functions
  - Supports power-down mode to minimize the power consumption of SDRAM
  - Supports page mode
- Direct Memory Access Controller
  - Six independent DMA channels
  - Descriptor supported
  - Transfer data units: 8-bit, 16-bit, 32-bit, 16-byte or 32-byte
  - Transfer requests can be: auto-request within DMA; and on-chip peripheral module request
  - Interrupt on transfer completion or transfer error
  - Supports two transfer modes: single mode or block mode
- The Jz4725 processor system supports little endian only

1.2.4 Clock generation and power management

- On-chip oscillator circuit for an 32768Hz clock and an 12MHz clock
- On-chip phase-locked loops (PLL) with programmable multiple-ratio. Internal counter are used to ensure PLL stabilize time
- PLL on/off is programmable by software
- ICLK, PCLK, SCLK, MCLK and LCLK frequency can be changed separately for software by setting division ratio
- Supports six low-power modes and function: NORMAL mode; DOZE mode; IDLE mode; SLEEP mode; HIBERNATE mode; and MODULE-STOP function.

1.2.5 On-chip peripherals

- General-Purpose I/O ports
  - Total GPIO pin number is 81
  - Each pin can be configured as general-purpose input or output or multiplexed with internal chip functions
  - Each pin can act as a interrupt source and has configurable rising/falling edge or high/low level detect manner, and can be masked independently
  - Each pin can be configured as open-drain when output
  - Each pin can be configured as internal resistor pull-up

- RTC (Real Time Clock)
  - 32-bit second counter
  - 1Hz from 32768hz
  - Alarm interrupt
  - Independent power
  - A 32-bits scratch register used to indicate whether power down happens for RTC power

- Interrupt controller
  - Total 28 maskable interrupt sources from on-chip peripherals and external request through GPIO ports
  - Interrupt source and pending registers for software handling
  - Unmasked interrupts can wake up the chip in sleep or standby mode

- Timer and counter unit with PWM output
  - Provide five separate channels
  - 16-bit A counter and 16-bit B counter with auto-reload function every channel
  - Support interrupt generation when the A counter underflows
  - Three clock sources: RTCLK (real time clock), EXCLK (external clock input), PCLK (APB Bus clock) selected with 1, 4, 16, 64, 256 and 1024 clock dividing selected
  - PWM output supported
- Watchdog timer
  - 16-bit counter in RTC clock with 1, 4, 16, 64, 256 and 1024 clock dividing selected

  - Generate power-on reset

- LCD controller
  - Single-panel display in active mode, and single- or dual-panel displays in passive mode
  - 2, 4, 16 grayscales and up to 4096 colors in STN mode
  - 2, 4, 16, 256, 4K, 32K, 64K, 256K and 16M colors in TFT mode
  - 18 bit data bus support 1,2,4,8 pins STN panel, 16bit and 18bit TFT and 8bit I/F TFT
  - Display size up to 800×600 pixels
  - 256×16 bits internal palette RAM
  - Support ITU601/656 data format
  - Support smart LCD (SRAM-like interface LCD module)

- AC97/I2S controller
  - Support the on-chip CODEC only

- On-chip audio CODEC
  - 18-bit DAC, SNR: 88dB
  - 16-bit ADC, SNR: 85dB
  - Sample rate: 8/11.025/12/16/22.05/24/32/44.1/48kHz
  - L/R channels line input
  - MIC input
  - L/R channels headphone output amplifier support up to 32ohm load

- SADC
  - 12-bit, 2Mbps, SNR@500kHz is 61dB, THD@500kHz is -71dB
  - Battery voltage input
  - 1 generic input Channel

- MMC/SD/SDIO controller
  - Compliant with “The MultiMediaCard System Specification version 3.3”
  - Compliant with “SD Memory Card Specification version 1.01” and “SDIO Card Specification version 1.0” with 1 command channel and 1 data channels
  - 5~20 Mbps maximum data rate
  - Supports up to 10 cards (including one SD card)
  - Maskable hardware interrupt for SD I/O interrupt, internal status, and FIFO status

- I2C bus interface
  - Only supports single master mode
  - Supports I2C standard-mode and F/S-mode up to 400 kHz
  - Double-buffered for receiver and transmitter
  - Supports general call address and START byte format after START condition

- UART

  - 5, 6, 7 or 8 data bit operation with 1 or 1.5 or 2 stop bits, programmable parity (even, odd, or none)
  - 16x8bit FIFO for transmit and 16x11bit FIFO for receive data
  - Interrupt support for transmit, receive (data ready or timeout), and line status
  - Supports DMA transfer mode
  - Provide complete serial port signal for modem control functions
  - Support slow infrared asynchronous interface (IrDA)
  - IrDA function up to 115200bps baudrate
  - UART function up to 921.6Kbps baudrate

- USB 2.0 device interface
  - Compliant with USB protocol revision 2.0
  - High speed and full speed supported
  - Embedded USB 2.0 PHY

1.3 Characteristic

Item	Characteristic
Process Technology	0.18um CMOS
Power supply voltage	I/O: 3.3 ± 0.3V Core: 1.8 ± 0.2
Package	LQFP 128 14mm * 14mm
Operating frequency	360MHz
Power consumption	200mw @ 360MHz

2 CPU Core

At the heart of the chip is the XBurst® CPU core. XBurst® CPU adopts a brand new micro-architecture which provides superior performance and power consumption than existent industry cores. Detailed description of XBurst® CPU core is specified in document titled “XBurst® Microprocessor Core User Manual”

Key features of XBurst® CPU core implemented in this chip are as following:

Table 2-1 XBurst® CPU Core Features

Item	Features
RISC ISA	• Industry standard Instruction set architecture • 32 32-bit general purpose registers
Ingenic Media ISA	• Implement 60 SIMD like instructions for multimedia acceleration • See document “Ingenic Media Instruction Set Architecture”
Ingenic Floating Point ISA	• Not implemented
Multiply-Divide Unit (MDU)	• Maximum issue rate of one 32x16 multiply every clock • Maximum issue rate of one 32x32 multiply every other clock • Minimum 2 clock cycle, maximum 34 clock cycles for divide
Memory Manager Unit (MMU)	• 4 G-Bytes of address space • 32/16 dual-entry full associative joint TLB plus 4 dual-entry ITLB and 4 dual-entry DTLB respectively • 7 different page size from 4Kb to 16MB supported in any entry • Support entry lock • Space identifier ASID: 8 bits
Data Cache	• Virtually-indexed, physically-tagged • 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes • LRU replacement algorithm • Write-back, write-through • 16-word depth write buffer
Instruction Cache	• Physically-indexed, physically-tagged • 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes • LRU replacement algorithm
Debug&JTAG	• JTAG interface to host machine • ACC mode to accelerate JTAG memory access • Two instruction and one data breakpoint
Branch Target Buffer (BTB)	• Virtally-tagged • Up to 64 entry direct mapped • 2-bit branch history maintained
Bus Interface	• compliance with AHB protocol

3 External Memory Controller

3.1 Overview

The External Memory Controller (EMC) divides the off-chip memory space and outputs control signals complying with specifications of various types of memory and bus interfaces. It enables the connection of static memory, NAND flash memory, synchronous DRAM, etc., to this processor.

- Static memory interface
  - Direct interface to ROM, Burst ROM, SRAM and NOR Flash.
  - Support 4 external chip selection CS4~1#. Each bank can be configured separately.
  - The size and base address of static memory banks are programmable.
  - Output of control signals allowing direct connection of memory to each bank. Write strobe setup time and hold time periods can be inserted in an access cycle to enable connection to low-speed memory
  - Wait state insertion can be controlled by program.
  - Wait insertion by WAIT pin.
  - Automatic wait cycle insertion to prevent data bus collisions in case of consecutive memory accesses to different banks, or a read access followed by a write access to the same bank

- NAND flash interface
  - Support on CS4~CS1, sharing with static memory bank4~bank1.
  - Support most types of NAND flashes, including 8-bit and 16-bit bus width, 512B and 2KB page size. For 512B page size, 3 and 4 address cycles are supported. For 2KB page size, 4 and 5 address cycles are supported.
  - Hardware ECC generation including Hamming and RS codes correction.
  - Support read/erase/program NAND flash memory.
  - Support boot from NAND flash.

- SDRAM Interface
  - Support 1 chip selection DCS#.
  - Support both 32-bit and 16-bit bus width.
  - Support both two-bank and four-bank type SDRAM.
  - Support burst operation.
  - Support both auto-refresh and self-refresh functions.
  - The size and base address of each bank is configurable.
  - Multiplexes row/column addresses according to SDRAM capacity
  - Controls timing of SDRAM direct-connection control signals according to register setting
  - Supports power-down mode to minimize the power consumption of SDRAM
  - Support page mode

3.2 Pin Description

Following table list the EMC pins.

Table 3-1 EMC Pin Description

Pin Name	I/O	Signal	Description
Data Bus	I/O	D31 – D0	Data I/O
Address bus	O	A22–A0	Address output
Static chip select 4 ~ 1	O	CS4~1#	Chip select signal that indicates the static bank being accessed
SDRAM chip select	O	DCS#	Chip select signal that indicates the SDRAM bank being accessed
Read enable	O	RD# /	For Static memory read enable signal
Write enable	O	WE# /	Static memory write enable signal
Column address strobe	O	CAS#	SDRAM column address strobe signal
Row address strobe	O	RAS#	SDRAM row address strobe signal
Read/write	O	RD/WR#	Data bus direction designation signal Also used as SDRAM write enable signal
Byte enable 0	O	WE0# / BE0# / DQM0 /	For non-byte-control static memory , D7-0 write enable signal, For byte-control static memory , D7-0 selection signal For SDRAM, D7–D0 selection signal
Byte enable 1	O	WE1# / BE1# / DQM1/	For non-byte-control static memory , D15-8 write enable signal For byte-control static memory , D15-8 selection signal For SDRAM, D15–D8 selection signal
Byte enable 2	O	WE2# / BE2# / DQM2 /	For non-byte-control static memory, D23-16 write enable signal For byte-control static memory, D23-16 selection signal For SDRAM , D23–D16 selection signal
Byte enable 3	O	WE3# / BE3# / DQM3	For static memory , D31-24 write enable signal For byte-control static memory , D31-24 selection signal For SDRAM, D31–D24 selection signal.
SDRAM Clock enable	O	CKE	Enable the SDRAM clock
Wait	I	Wait# /	External wait state request signal for memory-like devices
NAND flash read enable	O	FRE#	NAND flash read enable signal
NAND flash write enable	O	FWE#	NAND flash write enable signal
NAND flash ready/busy	I	FRB#	Indicates NAND flash is ready or busy (When Nand flash boot, GPC30 is used as FRB# of CS1#)

3.3 Physical Address Space Map

Both virtual spaces and physical spaces are 32-bit wide in this architecture. Virtual addresses are translated by MMU into physical address which is further divided into several partitions for static memory, SDRAM, and internal I/O.

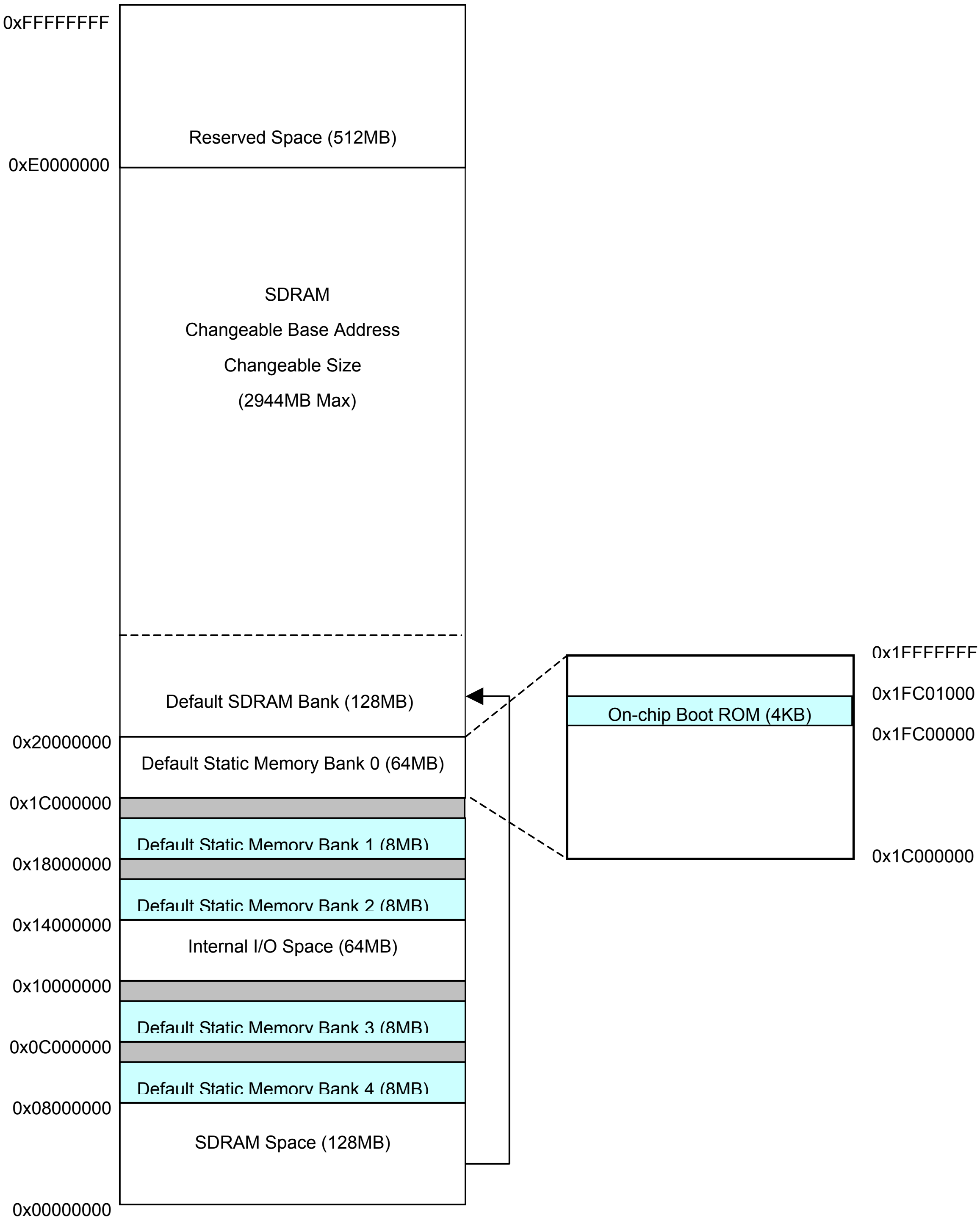

0xFFFFFFFF
Reserved Space (512MB)
0xE0000000
SDRAM
Changeable Base Address
Changeable Size
(2944MB Max)
Default SDRAM Bank (128MB)
0x20000000
Default Static Memory Bank 0 (64MB)
0x1C000000
Default Static Memory Bank 1 (8MB)
0x18000000
Default Static Memory Bank 2 (8MB)
0x14000000
Internal I/O Space (64MB)
0x10000000
Default Static Memory Bank 3 (8MB)
0x0C000000
Default Static Memory Bank 4 (8MB)
0x08000000
SDRAM Space (128MB)
0x00000000
0x1FFFFFFF
0x1FC01000
On-chip Boot ROM (4KB)
0x1FC00000
0x1C000000

Figure 3-1 Physical Address Space Map

Table 3-2 Physical Address Space Map

Start Address	End Address	Connectable Memory	Capacity
H’0000 0000	H’07FF FFF	SDRAM space	128 MB
H’0800 0000	H’0FFF FFFF	Static memory space	128 MB
H’1000 0000	H’13FF FFFF	Internal I/O space	64 MB
H’1400 0000	H’1BFF FFFF	Static memory space	128MB
H’1C00 0000	H’1FBF FFFF	Un-used	60MB
H’1FC0 0000	H’1FC0 0FFF	On-chip boot ROM	4KB
H’1FC0 1000	H’1FFF FFFF	Un-used	4095KB
H’2000 0000	H’BFFF FFFF	SDRAM space	2944 MB
H’D000 0000	H’FFFF FFFF	Reserved space	512 MB

The base address and size of each memory banks are configurable. Software can re-configure these memory banks according to the actual connected memories. Following table lists the default configuration after reset.

Table 3-3 Default Configuration of EMC Chip Select Signals

Chip-Select Signal	Connected Memory	Capacity	Memory Width [*1]	Start Address	End Address
CS1#	Static memory bank 1	8 MB	8, 16, 32	H'1800 0000	H'1BFF FFFF
CS2#	Static memory bank 2	8 MB	8, 16, 32	H'1400 0000	H'17FFFFFFF
CS3#	Static memory bank 3	8 MB	8, 16, 32	H'0C00 0000	H’0FFF FFFF
CS4#	Static memory bank 4	8 MB	8, 16, 32	H'0800 0000	H’0BFF FFFF
DCS#[*3]	SDRAM bank	128 MB	16, 32	H’2000 0000	H’27FF FFFF

Notes:

1. Data width of static memory banks can be configured to 8, 16 or 32 bits by software.

2. The 4KB address space from H’1FC00000 to H’1FC00FFF in bank 0 is mapped to on-chip boot ROM. The other memory spaces in bank 0 are not used.

3. To support large SDRAM space, EMC re-maps the physical address H’00000000-H07FFFFFF to H’20000000-H’27FFFFFF. Software must configure the SDRAM base address by the re-mapped address.

3.4 Static Memory Interface

The static memory controller provides a glueless interface to SRAM's, ROMs (PROMs/EPROMs/FLASH), dual port memory, IO devices, and many other peripherals devices. It can directly control up to 4 devices using four chip select lines. Additional devices may be supported through external decoding of the address bus. The Device Controller shares the data and address busses with the SDRAM controller. Thus, only one memory subsection (SDRAM, memory, or IO) can be active at any time.

Each chip select can directly access memory or IO devices that are 8-bits, 16-bits, or 32-bits wide. Each device connected to a chip select line has 2 associated registers that control its operation and the access timing to the external device. The Static Memory Control Register SMCRn specifies various configurations for the device. The Static Memory Address Configuration Register SACRn specifies the base address and size for each device, enabling any device to be located anywhere in the physical address range.

The static memory interface includes the following signals:

- Four chip selects, CS4~1#
- Twenty-three address signals, A22-A0
- One read enable, RD#
- One write enable, WE#
- Four byte enable, BE3~1#
- One wait pin, WAIT#

The SMT field in SMCRn registers specifies the type of memory and BW field specifies the bus width. BOOT_SEL[1:0] defines whether system boot from Nor or Nand flash and the page size when boot from Nand flash.

3.4.1 Register Description

Table 3-4 Static Memory Interface Registers

Name	Description	RW	Reset Value	Address	Access Width
SMCR1	Static memory control register 1	RW	0x0FFF7700	0x13010014	32
SMCR2	Static memory control register 2	RW	0x0FFF7700	0x13010018	32
SMCR3	Static memory control register 3	RW	0x0FFF7700	0x1301001C	32
SMCR4	Static memory control register 4	RW	0x0FFF7700	0x13010020	32
SACR1	Static memory bank 1 address configuration register	RW	0x000018FC	0x13010034	32
SACR2	Static memory bank 2 address configuration register	RW	0x000016FE	0x13010038	32
SACR3	Static memory bank 3 address configuration register	RW	0x000014FE	0x1301003C	32
SACR4	Static memory bank 4 address configuration register	RW	0x00000CFC	0x13010040	32

3.4.1.1 Static Memory Control Register (SMCR1~4)

SMCR1~4 are 32-bit read/write registers that contain control bits for static memory. On reset, SMCR1~4 are initialized to 0x0FFF7700.

SMCR1	**0x13010014**
SMCR2	**0x13010018**
SMCR3	**0x1301001C**
SMCR4	**0x13010020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					STRV				TAW				TBP					TAH				TAS			BW				BCM	BL		SMT
RST	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0/x	0/x	0	0	0	0	0	0

Bits	Name	Description	RW
31:28	Reserved	Writes to these bits have no effect and always read as 0.	R
27:24	STRV	**Static Memory Recovery Time:** Its value is the number of idle cycles (0~15 cycles) inserted between bus cycles when switching from one bank to another bank or between a read access to a write access in the same	RW

		bank. Its initial value is 0xF (15 cycles).	
23:20	TAW	**Access Wait Time:** For normal memory, these bits specify the number of wait cycles to be inserted in read strobe time. For burst ROM, these bits specify the number of wait cycles to be inserted in first data read strobe time. **TAW3~0 Wait cycle Wait# Pin** 0000 0 cycle Ignored 0001 1 cycle Enabled 0010 2 cycles Enabled 0011 3 cycles Enabled 0100 4 cycles Enabled 0101 5 cycles Enabled 0110 6 cycles Enabled 0111 7 cycles Enabled 1000 8 cycles Enabled 1001 9 cycles Enabled 1010 10 cycles Enabled 1011 12 cycles Enabled 1100 15 cycles Enabled 1101 20 cycles Enabled 1110 25 cycles Enabled 1111 31 cycles Enabled (Initial Value)	RW
19:16	TBP	**Burst Pitch Time:** For burst ROM, these bits specify the number of wait cycles to be inserted in subsequent access. For normal memory, these bits specify the number of wait cycles to be inserted in write strobe time. **TBP3~0 Wait cycle Wait# Pin** 0000 0 cycle Ignord 0001 1 cycle Enabled 0010 2 cycles Enabled 0011 3 cycles Enabled 0100 4 cycles Enabled 0101 5 cycles Enabled 0110 6 cycles Enabled 0111 7 cycles Enabled 1000 8 cycles Enabled 1001 9 cycles Enabled 1010 10 cycles Enabled 1011 12 cycles Enabled 1100 15 cycles Enabled 1101 20 cycles Enabled 1110 25 cycles Enabled 1111 31 cycles Enabled (Initial Value)	RW
15	Reserved	Writes to these bits have no effect and always read as 0.	R

14:12	TAH	**Address Hold Time:** These bits specify the number of wait cycles to be inserted from negation of read/write strobe to address. **TAH2~0 Wait cycle** 000 0 cycle 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101 5 cycles 110 6 cycles 111 7 cycles (Initial Value)	RW
11	Reserved	Writes to these bits have no effect and always read as 0.	R
10:8	TAS	**Address Setup Time:** These bits specify the number of wait cycles (0~7 cycles) to be inserted from address to assertion of read/write strobe. **TAS2~0 Wait cycle** 000 0 cycle 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101 5 cycles 110 6 cycles 111 7 cycles (Initial Value)	RW
7:6	BW	**Bus Width :** These bits specify the bus width. this filed is writeable and are initialized to 0 by a reset. **BW1~0 Bus Width** 00 8 bits (Initial Value) 01 16 bits 10 32 bits 11 Reserved	RW
5:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3	BCM	**SRAM Byte Control Mode (BCM):** When SRAM is connected; this bit specifies the type of SRAM. This bit is only valid when SMT is set to 0. **BCM Description** 0 SRAM is set to normal mode (Initial Value) 1 SRAM is set to byte control mode	RW
2:1	BL	**Burst Length (BL1, BL0):** When Burst ROM is connected; these bits specify the number of burst in an access. These bits are only valid when SMT is set to 1. **BL1~0 Burst Length** 00 4 consecutive accesses. Can be used with 8-, 16-, or 32-bit bus width (Initial Value). 01 8 consecutive accesses. Can be used with 8-, 16-, or	

		32-bit bus width 10 16 consecutive accesses. Can only be used with 8- or 16-bit bus width. Do not specify for 32-bit bus width 11 32 consecutive accesses. Can only be used with 8-bit bus width	
0	SMT	**Static Memory Type (SMT):** This bit specifies the type of static memory. **SMT** **Description** 0 Normal Memory (Initial Value) 1 Burst ROM	RW

3.4.1.2 Static Bank Address Configuration Register (SACR1~4)

SACR1~4 defines the physical address for static memory bank 1 to 4, respectively. Each register contains a base address and a mask. When the following equation is met:

$(physical\_address\ [31{:}24]\ \&\ \mathbf{MASK}_n) == \mathbf{BASE}_n$

The bank n is active. The *physical_address* is address output on internal system bus. Static bank regions must be programmed so that each bank occupies a unique area of the physical address space. Bank 0 base address must be 0 because it's system boot address. Programming overlapping bank regions will result in unpredictable error. These registers are initialized by a reset.

SACR1	**0x13010034**
SACR2	**0x13010038**
SACR3	**0x1301003C0x13010040**
SACR4	

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	BASE								MASK							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
15:8	BASE	**Address Base:** Defines the base address of Static Bank n (n = 1 to 4). The initial values are: SACR1.BASE 0x18 SACR2.BASE 0x14 SACR3.BASE 0x0C SACR4.BASE 0x08	RW
23:20	MASK	**Address Mask**: Defines the mask of Static Bank n (n = 1 to 4). The initial values are: SACR1.MASK 0xFC SACR2.MASK 0xFC SACR3.MASK 0xFC SACR4.MASK 0xFC	RW

3.4.2 Example of Connection

Following figures shows examples of connection to 32-, 16- and 8-bit data width normal memory.

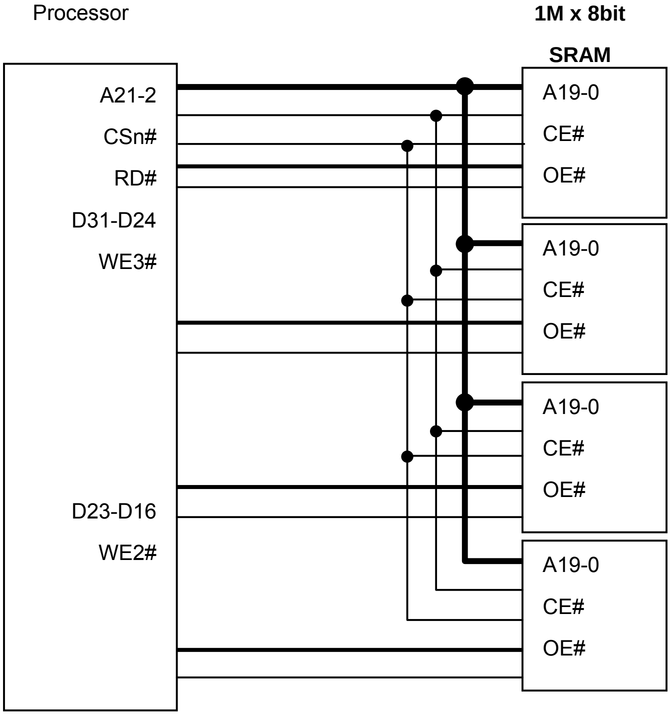

Processor
1M x 8bit
SRAM
A21-2
CSn#
RD#
D31-D24
WE3#
D23-D16
WE2#
A19-0
CE#
OE#
A19-0
CE#
OE#
A19-0
CE#
OE#
A19-0
CE#
OE#

Figure 3-2 Example of 32-Bit Data Width SRAM Connection

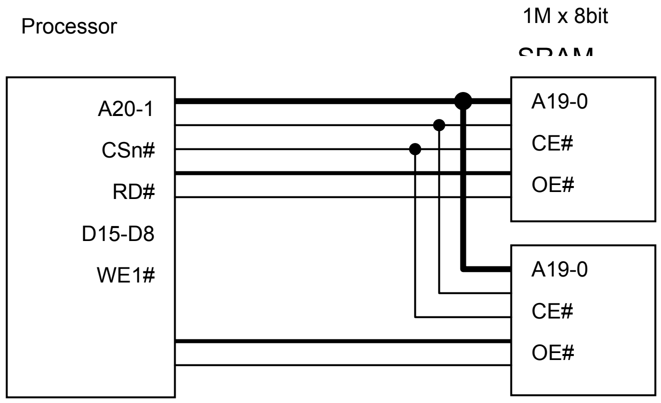

Processor
1M x 8bit
SRAM
A20-1
CSn#
RD#
D15-D8
WE1#
A19-0
CE#
OE#
A19-0
CE#
OE#

Figure 3-3 Example of 16-Bit Data Width SRAM Connection

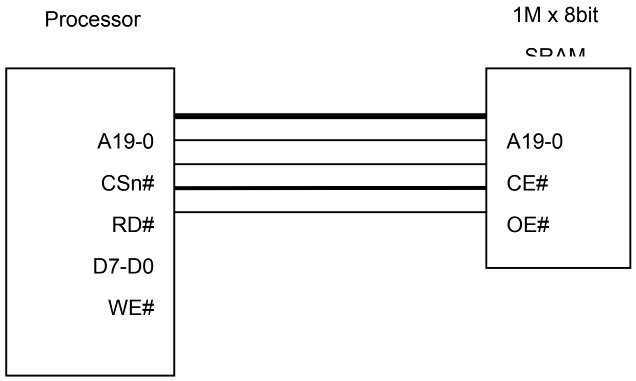

Processor
1M x 8bit
SRAM
A19-0
CSn#
RD#
D7-D0
WE#
A19-0
CE#
OE#

Figure 3-4 Example of 8-Bit Data Width SRAM Connection

3.4.3 Basic Interface

When SMT field in SMCRn (n = 1 to 4) is 0 and BCM field is 0, normal memory (non-burst ROM, Flash, normal SRAM or memory-like device) is connected to bank n. When bank n (n = 1 to 4) is accessed, CSn# is asserted as soon as address is output. In addition, the RD# signal, which can be used as OE#, and write control signals, WE0# to WE3#, are asserted.

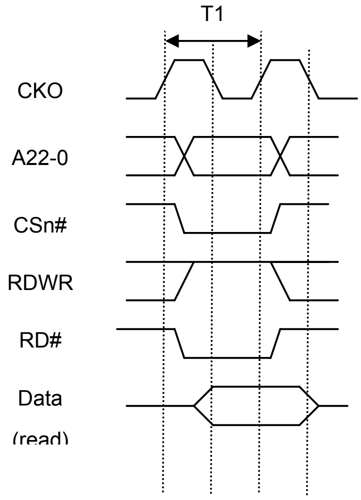
The TAS field in SMCRn is the latency from CSn# to read/write strobe. The TAW3 field is the delay time of RD# in read access. TBP3~0 field is the delay time of WE# and WEn# in write access. In addition, any number of waits can be inserted by means of the external pin (WAIT#). The TAH field is the latency from RD# and WEn# negation to CSn# negation, also the hold time to address and write data.

All kinds of normal memories (non-burst ROM, normal SRAM and Flash) have the same read and write timing. There are some requirements for writes to flash memory. Flash memory space must be un-cacheable and un-buffered. Writes must be exactly the width of the populated Flash devices on the data bus (no byte writes to a 32-bit bus or word writes to a 16-bit bus, and so on). Software is responsible for partitioning commands and data, and writing them out to Flash in the appropriate sequence.

Glossary

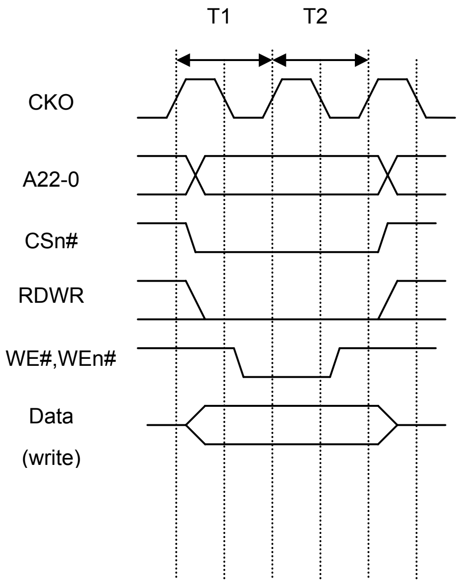
Th – hold cycle
Tw – wait cycle
Ts – setup cycle
T1 – read inherent cycle or first write inherent cycle
T2 – last write inherent cycle
Tb – burst read inherent cycle

Following figures show the timing of normal memory. A no-wait read access is completed in one cycle and a no-wait write access is completed in two cycles. Therefore, there is no negation period in case of access at minimum pitch.


T1
CKO
A22-0
CSn#
RDWR
RD#
Data
(read)

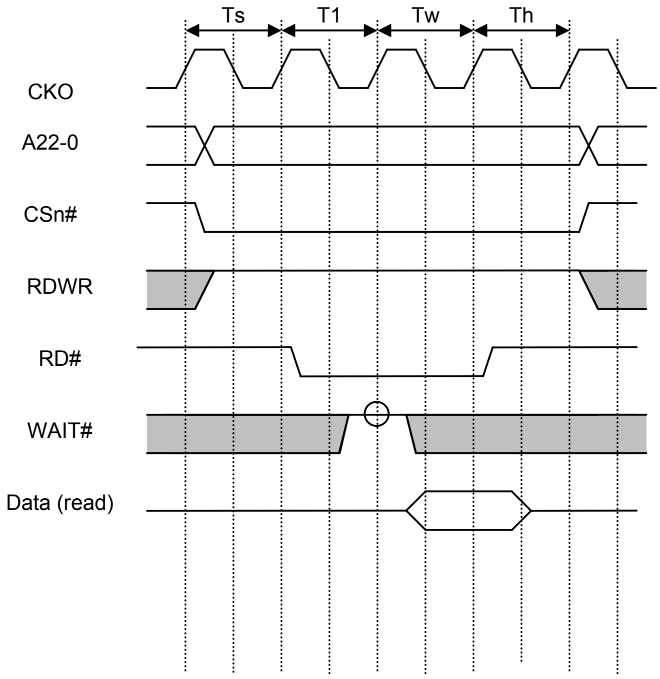
*In this example, SMCRn:MT = 0, BCM = 0,
TAS = 0, TAW = 0, TAH = 0

Figure 3-5 Basic Timing of Normal Memory Read


T1
T2
CKO
A22-0
CSn#
RDWR
WE#,WEn#
Data
(write)

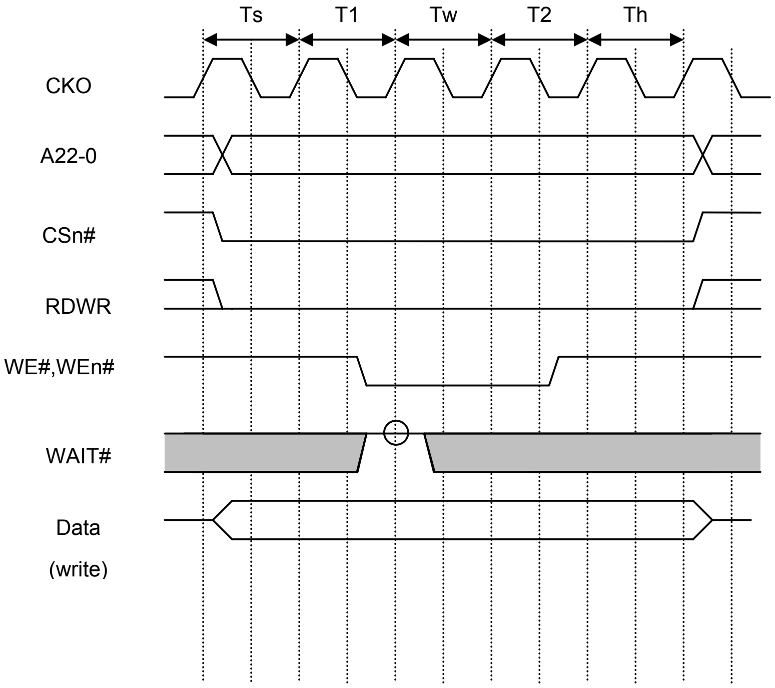
*In this example, SMCRn: SMT = 0, BCM = 0,
TAS = 0, TBP = 0, TAH = 0

Figure 3-6 Basic Timing of Normal Memory Write


Ts
T1
Tw
Th
CKO
A22-0
CSn#
RDWR
RD#
WAIT#
Data (read)

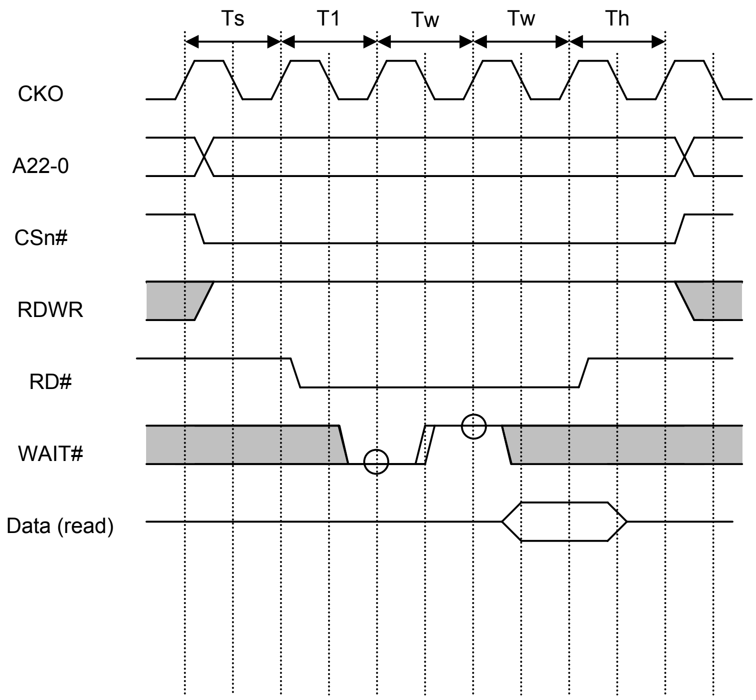
*In this example, SMCRn: SMT = 0, BCM = 0, TAS = 1, TAW = 1, TAH = 1

Figure 3-7 Normal Memory Read Timing With Wait (Software Wait Only)


Ts
T1
Tw
T2
Th
CKO
A22-0
CSn#
RDWR
WE#,WEn#
WAIT#
Data
(write)

*In this example, SMCRn: SMT = 0, BCM = 0, TAS = 1, TBP = 1, TAH = 1

Figure 3-8 Normal Memory Write Timing With Wait (Software Wait Only)


Ts
T1
Tw
Tw
Th
CKO
A22-0
CSn#
RDWR
RD#
WAIT#
Data (read)

*In this example, SMCRn: SMT = 0, BCM = 0, TAS = 1, TAW = 1, TAH=1

Figure 3-9 Normal Memory Read Timing With Wait (Wait Cycle Insertion by WAIT# pin)

3.4.4 Byte Control

The byte control SRAM interface is a memory interface that outputs a byte select strobe WEn# in both read and write bus cycles. It has 16 bit data pins, and can be directly connected to SRAM which has an upper byte select strobe and lower byte select strobe function such as UB# and LB#.

In read/write access, RD#/WE# is used as read/write strobe signal and WEn# are used as byte select signals.

Following figure shows an example of byte control SRAM connection to processor.

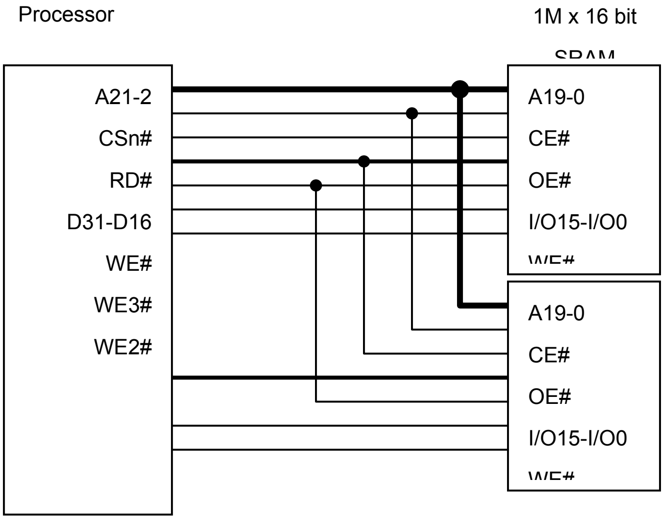
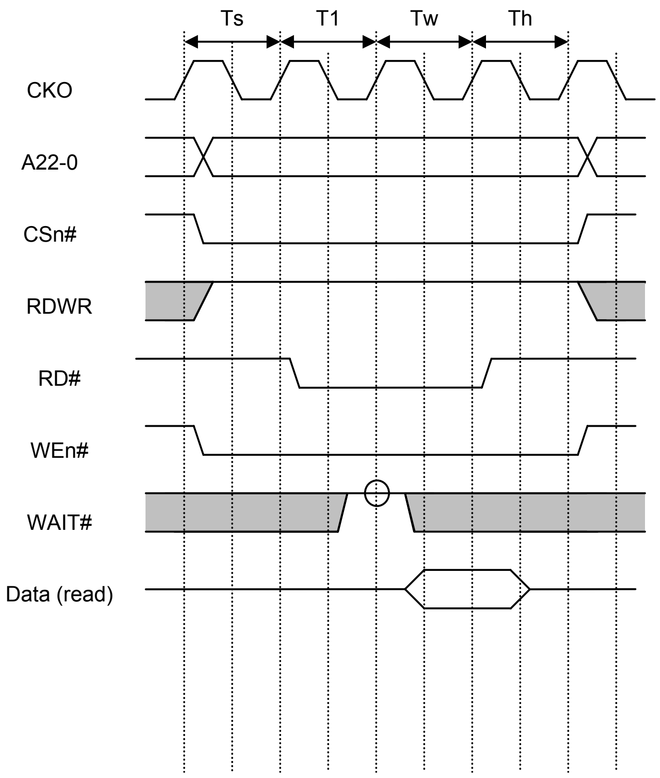

Processor
1M x 16 bit
SRAM
A21-2
CSn#
RD#
D31-D16
WE#
WE3#
WE2#
A19-0
CE#
OE#
I/O15-I/O0
WE#
A19-0
CE#
OE#
I/O15-I/O0
WE#

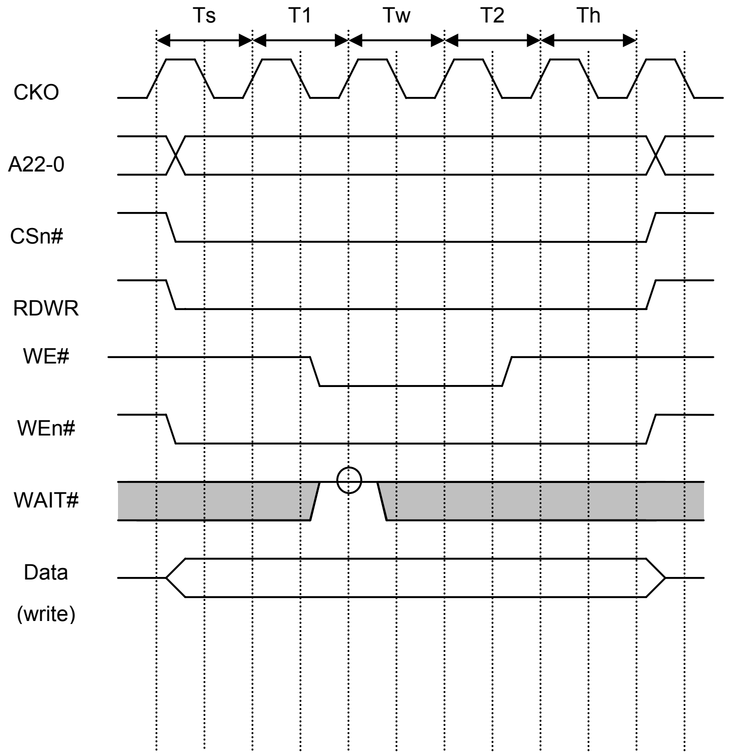
Figure 3-10 Example of 32-Bit Data Width Byte Control SRAM Connection

Following figures show examples of Byte Control SRAM timing.


Ts
T1
Tw
Th
CKO
A22-0
CSn#
RDWR
RD#
WEn#
WAIT#
Data (read)

*In this example, SMCRn: SMT = 0, BCM = 1, TAS = 1, TAW = 1, TAH = 1

Figure 3-11 Byte Control SRAM Read Timing


Ts
T1
Tw
T2
Th
CKO
A22-0
CSn#
RDWR
WE#
WEn#
WAIT#
Data
(write)

*In this example, SMCRn: SMT = 0, BCM = 1, TAS = 1, TBP = 1, TAH = 1

Figure 3-12 Byte Control SRAM Write Timing

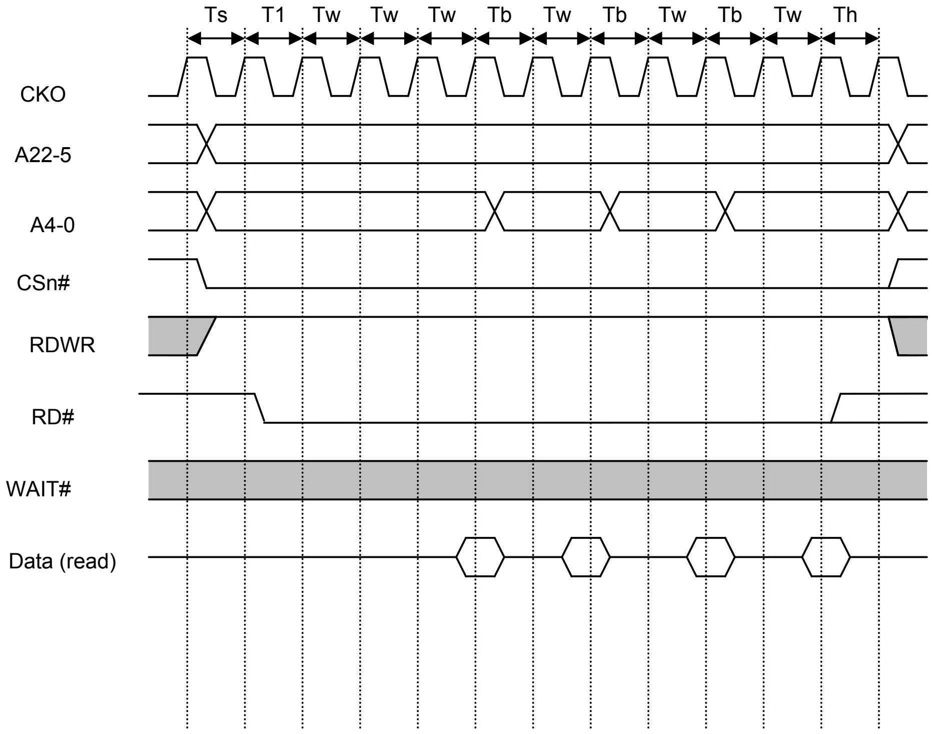
3.4.5 Burst ROM Interface

Setting SMT to 1 in SMCRn allows burst ROM to be connected to bank n (n = 1 to 4). The burst ROM interface provides high-speed access to ROM that has a nibble access function. Basically, access is performed in the same way as for normal memory, but when the first cycle ends, only the address is changed before the next access is executed. When 8-bit burst ROM is connected, the number of consecutive accesses can be set as 4, 8, 16, or 32 with bits BL1~0. When 16-bit ROM is connected, 4, 8, or 16 can be set in the same way. When 32-bit ROM is connected, 4 or 8 can be set.

For burst ROM read, TAW sets the delay time from read strobe to the first data, TBP sets the delay time from consecutive address to data. Burst ROM writes have the same timing as normal memory except TAW instead of TBP is used to set the delay time of write strobe.

WAIT# pin sampling is always performed when one or more wait states are set.

Following figures show the timing of burst ROM.



*In this example, SMT = 1, BL = 0, TAS = 1, TAW = 3,TBP = 1,TAH = 1

Figure 3-13 Burst ROM Read Timing (Software Wait Only)

3.5 NAND Flash Interface

NAND flash can be connected to static memory bank 4~ band 1. Both 8-bit and 16-bit NAND flashes are supported. Hardware ECC generator is implemented (including Hamming and RS codes correction). A mechanism for booting from NAND flash is also supported.

3.5.1 Register Description

Table 3-5 NAND Flash Interface Registers

Name	Description	RW	Reset Value	Address	Access Width
NFCSR	NAND flash control/status register	RW	0x00000000	0x13010050	32
NFECCR	NAND flash ECC control register	RW	0x00000000	0x13010100	32
NFECC	NAND flash ECC data register	R	Undefined	0x13010104	32
NFPAR0	NAND flash RS Parity 0 register	RW	0x00000000	0x13010108	32/16/8
NFPAR1	NAND flash RS Parity 1 register	RW	0x00000000	0x1301010C	32/16/8
NFPAR2	NAND flash RS Parity 2 register	RW	0x00000000	0x13010110	32/16/8
NFINTS	NAND flash Interrupt Status register	RW	0x00000000	0x13010114	32
NFINTE	NAND flash Interrupt Enable register	RW	0x00000000	0x13010118	32
NFERR0	NAND flash RS Error Report 0 register	R	0x00000000	0x1301011C	32/16
NFERR1	NAND flash RS Error Report 1 register	R	0x00000000	0x13010120	32/16
NFERR2	NAND flash RS Error Report 2 register	R	0x00000000	0x13010124	32/16
NFERR3	NAND flash RS Error Report 3 register	R	0x00000000	0x13010128	32/16

3.5.1.1 NAND Flash Control/Status Register (NFCSR)

NFCSR is a 32-bit read/write register that configure NAND flash. It is initialized by any reset.

NFCSR **0x13010050**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									NFCE4	NFE4	NFCE3	NFE3	NFCE2	NFE2	NFCE1	NFE1
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
1/3/5/7	FCEn (n=1,2,3, 4)	**NAND Flash FCE# Assertion Control :** Controls the assertion of NAND Flash FCEn#. When set, FCEn# is always asserted until this bit is cleared. When the NAND flash require FCEn# to be asserted during read busy time, this bit should be set	RW

		FCE **Description** 0 FCEn# is asserted as normal static chip enable(Initial value) 1 FCEn# is always asserted	
0/2/4/6	NFEn (n=1,2,3,4)	**NAND Flash Enable:** Specifies if NAND flash is connected to static bank n. When system is configured to boot from NAND flash, this bit is initialized to 1. **NFE** **Description** 0 Static bank n is not used as NAND flash. 1 Static bank n is used as NAND flash.	RW

3.5.1.2 NAND Flash ECC Control Register (NFECCR)

NFECCR is a 32-bit read/write register that is used to control ECC calculation. It is initialized by any reset.

NFECCR **0x13010100**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																												PRDY	ENCE	RSE	ERST	ECCE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and read always as 0.	R
4	PRDY	**PAR Ready**: It is used to indicate the parity data is ready in NFPAR0~2 register during RS decoding. It is automatically cleared by hardware and always read as 0. **PRDY** **Description** 0 Parity data is not available (Initial value) 1 Parity data is ready in NFPAR0~2 registers	W
3	ENCE	**RS Encoding/Decoding Select**: It is used to define whether in encoding or in decoding phase when RS is used. **ENCE** **Description** 0 Decoding (Initial value) 1 Encoding	RW
2	RSE	**Hamming and RS codes Select**: It is used to select the correction algorithm between Hamming and RS codes. **RSE** **Description** 0 Hamming (Initial value) 1 Reed-Solomn (RS)	RW
1	ERST	**NAND Flash ECC Reset:** It is used to reset ECC controller. This bit is cleared automatically by hardware and always read as 0.	W

		ERST **Description** 0 ECC controller is not reset(Initial value) 1 ECC controller is reset	
0	ECCE	**NAND Flash ECC Enable:** ECC correction is enable/disable. **ECCE** **Description** 0 ECC is disabled (initial value) 1 ECC is enabled	RW

3.5.1.3 NAND Flash ECC Data Register (NFECC)

NFECC is a 32-bit read only register that contains the result of ECC calculation. It is not initialized by any reset. When ERST of NFECCR is set, NFECC is initialized to 0.

NFECC **0x130100104**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	▨								ECC2								ECC1								ECC0							
RST	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	X	x	x	X

Bits	Name	Description	RW
31:24	Reserved	Writes to these bits have no effect and read always as 0.	R
23:16	ECC2	Byte 2 of ECC	R
15:8	ECC1	Byte 1 of ECC	R
7:0	ECC0	Byte 0 of ECC	R

3.5.1.4 NAND Flash Parity Register (NFPARn, n=0,1,2)

NFPAR0, NFPAR1 and NFPAR2 are all 32-bit read/write register that contains the encoding and decoding parity data during RS correction. It is initialized by any reset and ERST of NFECCR.

PARn (n=0~7), total 8 9-bit register together contains the parity data during RS correction. In encoding, they are writtern by hardware and software needs to read out and write into NAND flash spare space after NFINTS.ENCF bit is set to 1. In decoding, they are writtern by software. Software should first read out the 512B nand flash data and then 8 9-bit parity data and write the parity data into PARn registers.

NFPAR0 **0x13010108**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	PAR4(4~0 bits)					PAR5									PAR6									PAR7								
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NFPAR1 **0x1301010C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAR0(0 bit)	PAR1									PAR2									PAR3									PAR4(8~5 bits)			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NFPAR2 **0x13010110**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								PAR0 (8~1 bits)							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.5.1.5 NAND Flash Interrupt Status Register (NFINTS)

NFINTS is a 32-bit read-only register that contains the interrupt flag and error count information during RS correction. It is initialized by any reset. Software write 0 to clear the corresponding bit except ERRC.

NFINTS **0x13010114**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ERRC			Reserved																								PADF	DECF	ENCF	UNCOR	ERR
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:29	ERRC	**ERR Count:** It indicates the number of errors in the nand flash data block and these bits are also reset by NFECCR.ERST bit. **ERRC** **Description** 0 No errors or uncorrection error occurs (Initial value) 1 One error in the nand flash data block 2 Two errors in the nand flash data block	R

		3 Three errors 4 Four errors	
28:5	Reserved	Writes to these bits have no effect and read always as 0.	R
4	PADF	**Padding Finish:** It indicates that hardware finish padding zero after reading the 512B nand flash data block during RS decoding. **PADF Description** 0 Padding not finish (Initial value) 1 Padding finish	R
3	DECF	**Decoding Finish:** It indicates that hardware finish RS decoding. **PADF Description** 0 Decoding not Finish (Initial value) 1 Decoding Finish	R
2	ENCF	**Encoding Finish:** It indicates that hardware finish RS encoding. **PADF Description** 0 Encoding not Finish (Initial value) 1 Encoding Finish	R
1	UNCOR	**Uncorrection Error:** It indicates that hardware finish RS encoding. **UNCOR Description** 0 No uncorrectable error (Initial value) 1 Uncorrectable error occur	R
0	ERR	**Error:** It indicates that hardware detects error data in the 512B nand flash data block during RS decoding. **ERR Description** 0 No error (Initial value) 1 Error occur	R

3.5.1.6 NAND Flash Interrupt Enable Register (NFINTE)

NFINTE is a 32-bit read/write register that is used to enable/disable nand flash interrupt during RS correction. It is initialized by any reset.

NFINTE **0x13010118**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																											PADFE	DECFE	ENCFE	UNCORE	ERRE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and read always as 0.	R
4	PADFE	**Padding Finish Interrupt Enable:** It is used enable or disable padding finish interrupt. **PADFE Description**	RW

		0 Disable Padding finish interrupt (Initial value) 1 Enable Padding finish interrupt	
3	DECFE	**Decoding Finish Interrupt Enable:** It is used to enable or disable decoding finish interrupt. **DECFE Description** 0 Disable Decoding Finish Interrupt (Initial value) 1 Enable Decoding Finish Interrupt	RW
2	EDCFE	**Encoding/Decoding Finish Interrupt Enable:** It is used to enable or disable encoding and decoding finish interrupt. **EDCFE Description** 0 Disable Encoding/Decoding Finish Interrupt (Initial value) 1 Enable Encoding/Decoding Finish Interrupt	RW
1	UNCORE	**Uncorrection Error Interrupt Enable:** It is used to enable or disable uncorrection error interrupt. **UNCORE Description** 0 Disable Uncorrectable Error interrut (Initial value) 1 Enable Uncorrectable Error Interrupt	RW
0	ERRE	**Error Interrupt Enable:** It is used to enable or disable error interrupt. **ERRE Description** 0 Disable Error interrupt (Initial value) 1 Enable Error interrupt	RW

3.5.1.7 NAND Flash Error Report Register (NFERRn, n=0,1,2,3)

NFERRn is 32-bit read/write register that contains the index and error value for each error symbol after RS decoding. It is initialized by any reset and ERST of NFECCR.

NFERR0	**0x1301011C**
NFERR1	**0x13010120**
NFERR2	**0x13010124**
NFERR3	**0x13010128**

<table>
<tr><th>Bit</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td></td><td colspan="7">Reserved</td><td colspan="9">INDEXn(n=0,1,2,3)</td><td colspan="7">Reserved</td><td colspan="9">MASKn(n=0,1,2,3)</td></tr>
<tr><td>RST</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Bits	Name	Description	RW
31:25	Reserved	Writes to these bits have no effect and read always as 0.	R
24:16	INDEXn	**Error Symbol Index:** It is used to indicate the location of the error symbol in the 511 symbols. For example, INDEX=1, it means the first symbol has error bits.	R

15:9	Reserved	Writes to these bits have no effect and read always as 0.	R
8:0	MASKn	**Error Symbol Value:** It is used to indicate the error value of the indexed symbol. For example, INDEX=1, and MASK=3, it means the first two bits of the first symbol are wrong, and software need to XOR MASK and the indexed symbol to get the right data.	R

3.5.2 NAND Flash Boot Loader

To support boot from NAND flash, 4KB on-chip Boot ROM is implemented. Following figure illustrates the structure of NAND Flash Boot Loader.

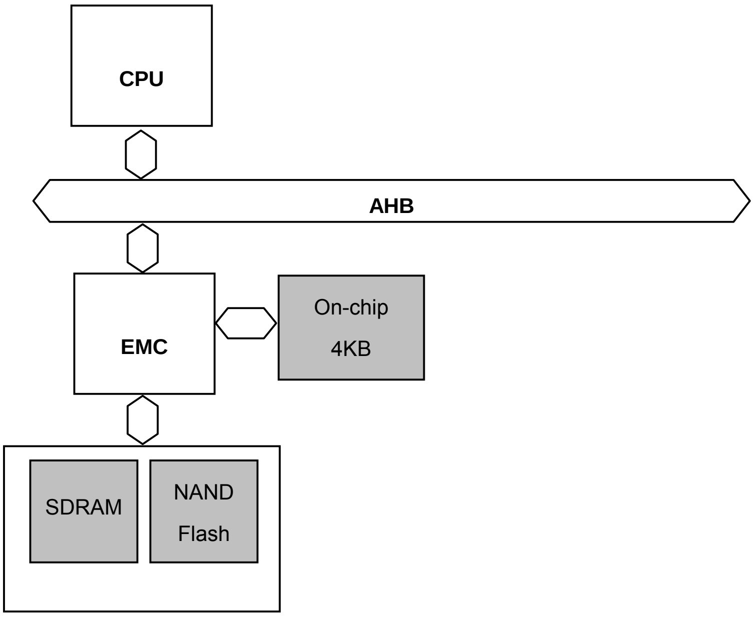


Figure 3-14 Structure of NAND Flash Boot Loader

When system is configured to boot from NAND flash, after reset, the program in Boot ROM is executed and the program will copy the first 4K bytes of NAND flash to internal memory for further initialization.

Generally, the boot code will copy more NAND flash content to SDRAM. Hardware ECC can be utilized to check the data validity. Then the main program will be executed on SDRAM.

When system is configured to boot from NAND flash, software may know the nand flash page size through BOOT_SEL[1:0] pin.

3.5.3 NAND Flash Operation

Set NFEn bit of NAND Flash Control/Status Register (NFCSR) will enable access to NAND flash. The partition of static bank n (n=1~4) is changed as following figure. Writes to any of address space will be translated to NAND flash address cycle. Writes to any of command space will be translated to NAND flash command cycle. Caution: don't read to address and command space, and these two partitions should be uncacheable. Reads and writes to any of data space will be translated to NAND flash data read/write cycle. DMA access to data space is supported to increase the speed of data read/write. The DMA access cannot exceed the page boundary (512 bytes or 2K bytes) of NAND flash.

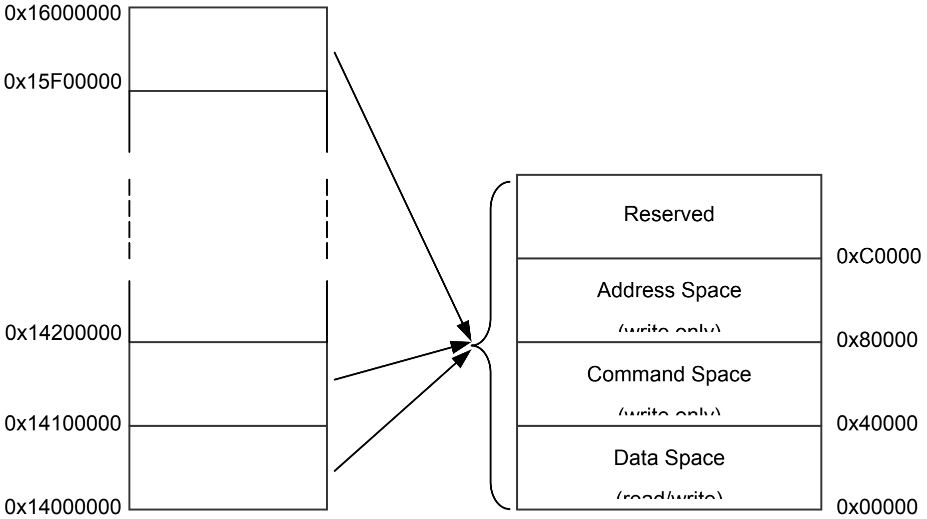

0x16000000
0x15F00000
0x14200000
0x14100000
0x14000000
Reserved
Address Space
(write only)
Command Space
(write only)
Data Space
(read/write)
0xC0000
0x80000
0x40000
0x00000

Figure 3-15 Static Bank 2 Partition When NAND Flash is Used (an example)

The timing of NAND flash access is configured by SMCRn and is same as normal static memory timing, except that CSn# is controlled by NFCE bit NFCSR. CSn# is always asserted when NFCE is 1. When NFCE is 0, CSn# is asserted as normal static memory access.

The control signals for direction connection of NAND flash are CSn#, FRE#, FWE#, FRB#(GPIO), A16 and A15. Following figure shows the connection between processor and NAND Flash.

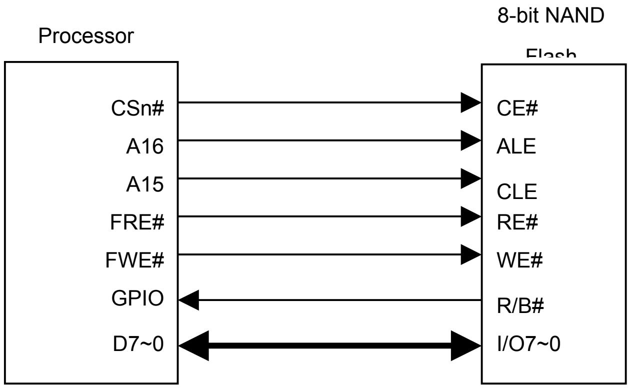


Figure 3-16 Example of 8-bit NAND Flash Connection

Hardware ECC generation for 8-/16-bit organization is implemented. There are two algorithm that could be used.

3.5.3.1 Hamming

When using Hamming algorithm, ECC parity code consists of 24 bits per 512 bytes (256 halfwords) and 22 bits per 256 bytes. Following table shows the ECC code assignment.

Table 3-6 512-Byte ECC Parity Code Assignment Table For 8-bit NAND Flash

	Data7	**Data6**	**Data5**	**Data4**	**Data3**	**Data2**	**Data1**	**Data0**
ECC0	P64	P64’	P32	P32’	P16	P16’	P8	P8’
ECC1	P1024	P1024’	P512	P512’	P256	P256’	P128	P128’
ECC2	P4	P4’	P2	P2’	P1	P1’	P2048	P2048’

24-bit ECC parity code = 18-bit line parity + 6-bit column parity

Table 3-7 256-Byte ECC Parity Code Assignment Table For 8-bit NAND Flash

	Data7	**Data6**	**Data5**	**Data4**	**Data3**	**Data2**	**Data1**	**Data0**
ECC0	P64	P64’	P32	P32’	P16	P16’	P8	P8’
ECC1	P1024	P1024’	P512	P512’	P256	P256’	P128	P128’
ECC2	P4	P4’	P2	P2’	P1	P1’	X	X

22-bit ECC parity code = 16-bit line parity + 6-bit column parity

Table 3-8 256-Halfword ECC Parity Code Assignment Table For 16-bit NAND Flash

	Data7	**Data6**	**Data5**	**Data4**	**Data3**	**Data2**	**Data1**	**Data0**
ECC0	P128	P128’	P64	P64’	P32	P32’	P16	P16’
ECC1	P2048	P2048’	P1024	P1024’	P512	P512’	P256	P256’

ECC2	P8	P8’	P4	P4’	P2	P2’	P1	P1’

24-bit ECC parity code = 16-bit line parity + 8-bit column parity

3.5.3.2 Reed-Solomn

RS controller uses RS(511, 503) codes. The total codes have 511 symbols, each symbol is 9-bit. The message has 503 9-bit symbols. So 512B nand flash data will be split into 9-bit symbols and it is 455 1/9 symbols, and then hardware will padding 47 8/9 zero symbol to build the 503 symbol message. During encoding, after padding zero, hardware will generate 8 9-bit parity symbol, software should read out the parity and write into nand flash spare space. During decoding, after reading out the 512B data and padding zero, hardware will using the parity data in NFPAR0~2 written by software to generate error information in NFERR0~3. Software should first read out the 512B nand flash data and then the parity data and write the parity data into NFPAR0~2.

NAND Flash Initialize Sequence

1. Configure SMCRn (n=1,2,3,4) according to the NAND flash AC characteristics.
2. Set NFEn bit of NFCSR to 1

NAND Flash Program Sequence

1. Set NFCEn bit of NFCSR to 1 to assert CSn# continuously
2. Write 0x80 to command space to issue Page Program command
3. Write 2 or 3 bytes of address to address space
4. Enable and reset the ECC generator by setting ECCE and ERST bits of NFECCR.
5. Select Haming or RS algorithm by setting NFECCR.RSE, 0: Hamming, 1: RS
6. If using RS algorithm, set NFECCR.ENCE to 1
7. Write 256 or 512 bytes data to data space using Hamming, write 512B data using RS
8. When using Hamming, clear ECCE bit of NFECCR to disable ECC generator and read out the ECC parity code from NFECC register, write 16-byte redundant data
9. When using RS, poll NFINTS.ENCF bit or wait for ENCF interrupt, and read out parity data from NFPAR0~2 register and write them into nand flash spare space
10. Write 0x10 to command space to issue Page Program command
11. Clear NFCEn bit of NFCSR to deassert CSn#
12. Check RB bit of NFCSR to wait the program complete

NAND Flash Read Sequence

1. Set NFCEn bit of NFCSR to 1 to assert CSn# continuously
2. Write 0x00 to command space to issue Read command
3. Write 2 or 3 bytes of address-to-address space

4. Enable and reset the ECC generator by setting ECCE and ERST bits of NFECCR.
5. Select Haming or RS algorithm by setting NFECCR.RSE, 0: Hamming, 1: RS
6. Check RB# pin (GPIO) to wait for NAND flash is not busy
7. When using Hamming, Read 256 or 512 bytes from NAND flash (DMA can be used), clear ECCE bit of NFECCR to disable ECC generator and Read 16-byte redundant data from NAND flash, then read out the ECC parity code from NFECC register, compare these two parity codes and correct the error bit or run the error routine
8. When using RS, Read 512 bytes data and 9 bytes parity data from NAND flash (DMA can be used), poll NFINTS.DECF bit or wait for decoding finish interrupt, and then analyze error status in NFINTS register, if there is any error, read NFERR0~3 registers, find the error byte according to NFERRn.INDEXn and XOR NFERRn.MASKn with the indexed bytes to get the correct data
9. If continuous pages are needed to be read, repeat steps from 5 to 9.
10. Clear NFCEn bit of NFCSR to deassert CSn#

ECC Generation Sequence for large page

In large page (2KBX8/1KHWX16 org.), 24-bit ECC code is generated for every 512 bytes or 256 halfwords data. Software gets ECC codes as following steps:

1. Set ERST bit of NFECCR to reset ECC generator
2. Read/write 512 bytes / 256 halfwords data
3. Read out 24-bit ECC code from NFECC register
4. Repeat step 1 to 3 until whole page is completed
5. Write spare area according to above ECC codes.

3.6 SDRAM Interface

The SDRAM controller provides a glueless interface to industry standard SDRAM chip. The SDRAM controller provides one chip selects DCS# supporting 16-bit or 32-bit wide SDRAM.

Both 2-bank and 4-bank SDRAM modules are supported. The bank select signals are always output from the A13 pin and A14 pin of processor.

The SDRAM interface includes the following signals:

- One chip selects, DCS#
- Four byte mask signals, DQM3~0#
- 15 multiplexed bank/row/column address signals, A14-A0
- One write enable, RD/WR#
- One column-address strobe CAS#
- One row-address strobe RAS#
- One clock enable CKE
- One clock CKO

The processor performs auto-refresh (CBR) during normal operation and supports self-refreshing SDRAM during sleep, hibernate, and frequency-change modes. An SDRAM power-down mode bit (DMCR[PDM]]) can be set so that the CKO and the clock-enable signal CKE to SDRAM are automatically deasserted whenever none of the corresponding banks is being accessed.

3.6.1 Register Description

Table 3-9 SDRAM Registers

Name	Description	RW	Reset Value	Address	Access Width
DMCR	DRAM control register	RW	0x0000 0000	0x13010080	32
RTCSR	Refresh time control/status register	RW	0x0000	0x13010084	16
RTCNT	Refresh timer counter	RW	0x0000	0x13010088	16
RTCOR	Refresh time constant register	RW	0x0000	0x1301008C	16
DMAR	SDRAM bank address configuration register	RW	0x000020F8	0x13010090	32
SDMR	Mode register of SDRAM bank	W	--	0x1301A000	8

3.6.1.1 SDRAM Control Register (DMCR)

DMCR is a 32-bit read/write register that specifies the timing, address multiplexing and refresh control of SDRAM. This enables direct connection of SDRAM without external circuits.

The DMCR is initialized to 0x00000000 by any resets. SDRAM bank should not be accessed until initialization is completed.

DMCR **0x13010080**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BW			CA			RMODE	RFSH	MRSET		RA		BA	PDM	EPIN		TRAS			RCD		TPC				TRWL		TRC			TCL	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	BW	Specifies the data bus width of SDRAM **BW** **Description** 0 Data width is 32 bits (Initial value) 1 Data width is 16 bits	RW
30:29	Reserved	Writes to these bits have no effect and always read as 0.	R
28:26	CA	**Column Address Width**: Specify the column address width of connected SDRAM chip. **CA** **Description** 000 8 bits column address 001 9 bits column address 010 10 bits column address	RW

		011 11 bits column address 100 12 bits column address 101 Reserved 110 Reserved 111 Reserved	
25	RMODE	**Refresh Mode.** **RMODE Description** 0 Auto-refresh 1 Self-refresh	RW
24	RFSH	**Refresh Control.** **RFSH Description** 0 No refresh is performed (Initial value) 1 Refresh is performed	RW
23	MRSET	**Mode Register Set:** Set when a SDRAM mode register setting is used. When this bit is 0 and SDRAM mode register is written, a Pre-charge all banks command (PALL) is performed. When this bit is 1 and SDRAM mode register is written, a Mode Register Set command (MRS) is performed. **MRSET Description** 0 All-bank pre-charge (Initial value) 1 Mode register setting	RW
22	Reserved	Writes to these bits have no effect and always read as 0.	R
21:20	RA	**Row Address Width:** Specify the row address width of connected SDRAM. **RA Description** 00 11-bit row address (Initial value) 01 12-bit row address 10 13-bit row address 11 Reserved	RW
19	BA	**Bank Address Width:** Specify the number of bank select signals for one chip select. **BA Description** **0** 1-bit bank address is used (2 banks each chip select) (Initial value) **1** 2-bit bank address is used (4 banks each chip select)	RW
18	PDM	**Power Down Mode:** Set power-down mode. When power-down mode is set, SDRAM will be driven to power-down mode when it is not accessing and refreshing. Clock supply to SDRAM will be stopped also. **PDM Description** 0 Non-power-down mode (Initial value) 1 Power-down mode	RW
17	EPIN	**CKE Pin Control:** Controls the level of CKE pin. Clearing this bit by software causes a power-down command (if CKOEN of CPM is 1).	RW

		Caution: after power-down command, all commands except power-down-exit are prohibited. Setting this bit by software causes a power-down-exit command. Setting EPIN is a part of initializes procedure for SDRAM. **EPIN** **Description** 0 CKE pin is deserted (Initial value) 1 CKE pin is asserted	
16	Reserved	Writes to these bits have no effect and always read as 0.	R
15:13	TRAS	**RAS Assertion Time:** When synchronous DRAM is connected, these bits set the minimum CKE negation time after self-refresh command is issued. **TRAS** **Description** 000 4 (Initial value) 001 5 010 6 011 7 100 8 101 9 110 10 111 11	RW
12:11	RCD	**RAS–CAS Delay**: Set the SDRAM bank active-read/write command delay time. **RCD** **Description** 00 1(Initial value) 01 2 10 3 11 4	RW
10:8	TPC	**RAS Precharge Time:** Specify the minimum number of cycles until the next bank active command is output after precharging. TPC Description 000 1 cycle (Initial value) 001 2 cycles 010 3 cycles 011 4 cycles 100 5 cycles 101 6 cycles 110 7 cycles 111 8 cycles	RW
7	Reserved	Writes to these bits have no effect and always read as 0.	R
6:5	TRWL	**Write Precharge Time:** Set the SDRAM write precharge delay time. In auto-precharge mode, they specify the time until the next bank active command is issued after a write cycle. After a write cycle, the next active command is not issued for a period of TRWL + TPC.	RW

		TRWL **Description** 00 1 cycle (Initial value) 01 2 cycles 10 3 cycles 11 4 cycles	
4:2	TRC	**RAS Cycle Time:** For SDRAM, no bank active command is issued during the period TRC after an auto-refresh command. In self-refresh, these bits also specify the delay cycles to be inserted after CKE assertion. **TRC** **Description** 000 1 cycle (Initial value) 001 3 cycle 010 5 cycle 011 7 cycle 100 9 cycle 101 11 cycle 110 13 cycle 111 15 cycle	RW
1:0	TCL	**CAS Latency:** Specify the delay from read command to data becomes available at the outputs. **TCL** **Description** 00 Inhibit (Initial value) 01 2 cycles 10 3 cycles 11 Inhibit	RW

3.6.1.2 SDRAM Mode Register (SDMR)

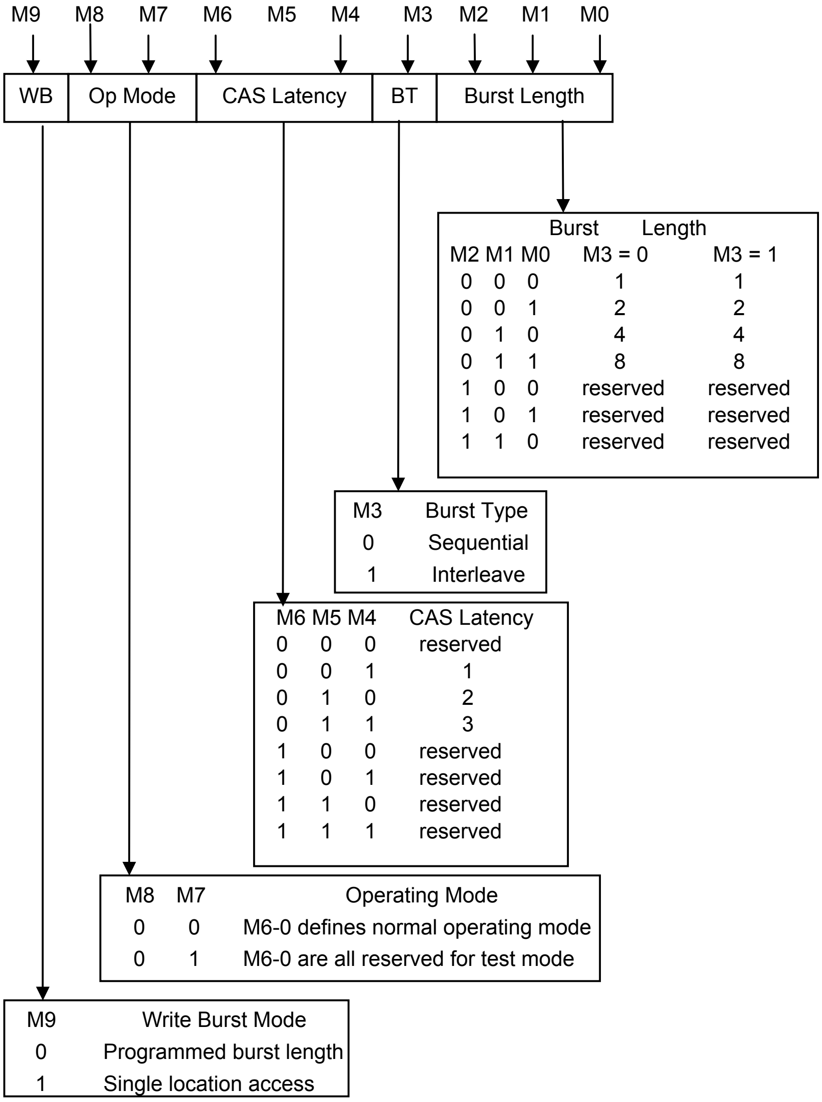
SDMR is written to via the SDRAM address bus and is a 10-bit write-only register. It sets SDRAM mode for SDRAM bank. SDMR is undefined after a reset.

Write to the SDRAM mode register use the address bus rather than the data bus. If the value to be set is X and the SDMR address is Y, the value X is written in the SDRAM mode register by writing in address X + Y. Here Y is 0xA000, X is value for SDRAM configuration. For example Y is 0x022, random data is writer to the address offset 0xA022, as a result, 0x022 is written to the SDMR register. The range for value X is 0x000 to 0x3FF.

SDMR **0x1301A000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		SDRAM address																														
RST																																

The Mode Register is used to define the specific mode of operation of the SDRAM. This definition includes the section of a burst length, a burst type, a CAS latency, an operating mode and a write burst mode, as shown in following figure.



M2 M1 M0			Burst Length M3 = 0	Burst Length M3 = 1
0	0	0	1	1
0	0	1	2	2
0	1	0	4	4
0	1	1	8	8
1	0	0	reserved	reserved
1	0	1	reserved	reserved
1	1	0	reserved	reserved

M3	Burst Type
0	Sequential
1	Interleave

M6	M5	M4	CAS Latency
0	0	0	reserved
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	reserved
1	0	1	reserved
1	1	0	reserved
1	1	1	reserved

M8	M7	Operating Mode
0	0	M6-0 defines normal operating mode
0	1	M6-0 are all reserved for test mode

M9	Write Burst Mode
0	Programmed burst length
1	Single location access

Figure 3-17 Synchronous DRAM Mode Register Configuration

3.6.1.3 Refresh Timer Control/Status Register (RTCSR)

RTCSR is a 16-bit readable/writable register that specifies the refresh cycle and the status of RTCNT.

RTCSR is initialized to 0x0000 by a reset.

RTSCR **0x13010084**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								CMF	Reserved				CKS		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved	These bits always read 0. Data written to these bits are ignored	R
7	CMF	**Compare-Match Flag (CMF):** Status flag that indicates a match between the refresh timer counter (RTCNT) and refresh time constant register (RTCOR) values. Writes to 1 of this bit have no effect. **CMF** **Description** 0 RTCNT and RTCOR values do not match (Initial value) Clear condition: When 0 is written 1 RTCNT and RTCOR values match Set condition: When RTCNT = RTCOR	
2:0	CKS	**Refresh Clock Select Bits:** These bits select the clock input to RTCNT. The source clock is the external bus clock (CKO). The RTCNT count clock is CKO divided by the specified ratio. **CKS** **Description** 000 Disable clock input (Initial value) 001 Bus lock CKO/4 010 CKO/16 011 CKO/64 100 CKO/256 101 CKO/1024 110 CKO/2048 111 CKO/4096	

3.6.1.4 Refresh Timer Counter (RTCNT)

RTCNT is a 16-bit read/write register. RTCNT is a 16-bit counter that counts up with input clocks. The clock select bits (CKS2–CKS0) of RTCSR select the input clock. When the refresh bit (RFSH) of the memory control register (DMCR) is set to 1 and the refresh mode is set to auto-refresh, a memory refresh cycle starts when RTCNT matches RTCOR. RTCNT is initialized to 0x0000 by a reset.

RTCNT **0x13010088**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.6.2 Refresh Time Constant Register (RTCOR)

RTCOR is a 16-bit read/write register. The values of RTCOR and RTCNT (bottom 8 bits) are constantly compared. When the refresh bit (RFSH) of the memory control register (DMCR) is set to 1 and the refresh mode bit (RMODE) is set to auto-refresh, a memory refresh cycle starts when RTCNT matches RTCOR. RTCOR is initialized to 0x0000 by a reset.

RTCOR **0x1301008C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.6.2.1 DRAM Bank Address Configuration Register (DMAR)

DMAR define the physical address for SDRAM bank, respectively. Each register contains a base address and a mask. When the following equation is met:

(*physical_address* [31:24] & $\mathbf{MASK_n}$) == $\mathbf{BASE_n}$

The bank n is active. The *physical_address* is address output on internal system bus. DRAM bank regions must be programmed so that each bank occupies a unique area of the physical address space. Programming overlapping bank regions will result in unpredictable error.

These registers are initialized by a reset.

DMAR **0x13010090**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	BASE								MASK							
RST																																

Bits	**Name**	**Description**	**RW**
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
15:8	BASE	**Address Base:** Defines the base address of SDRAM Bank. The initial values are: DMAR.BASE 0x20	RW
23:20	MASK	**Address Mask**: Defines the mask of SDRAM Bank. The initial values are: DMAR.MASK 0xF8	RW

3.6.3 Example of Connection

Following figure shows an example of connection of 512K x 16-bit x 2-bank SDRAM.

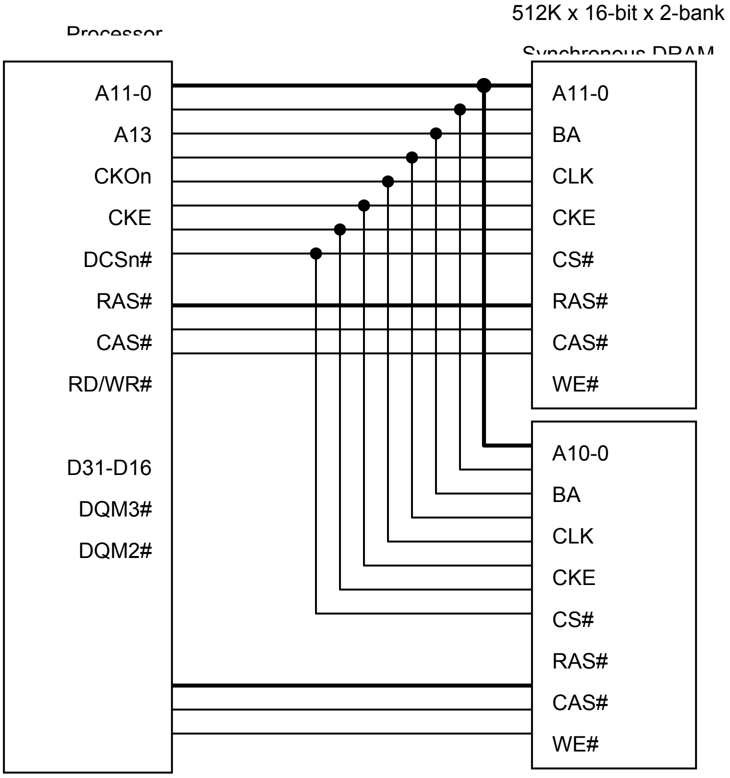

Processor
512K x 16-bit x 2-bank
Synchronous DRAM
A11-0
A13
CKOn
CKE
DCSn#
RAS#
CAS#
RD/WR#
D31-D16
DQM3#
DQM2#
A11-0
BA
CLK
CKE
CS#
RAS#
CAS#
WE#
A10-0
BA
CLK
CKE
CS#
RAS#
CAS#
WE#

Figure 3-18 Example of Synchronous DRAM Chip Connection (1)

Following figure shows an example of connection of 1M x 16-bit x 4-bank synchronous DRAM.

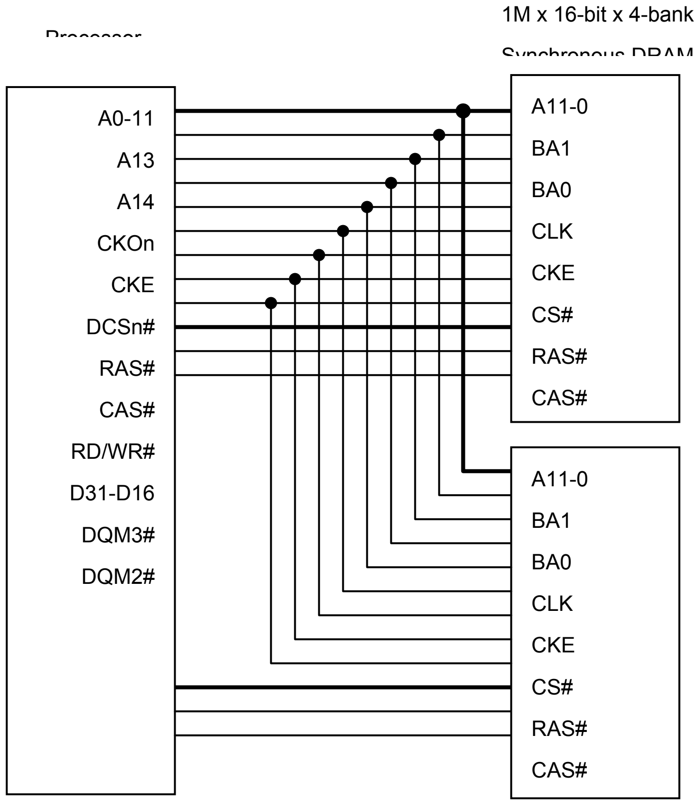

Processor
1M x 16-bit x 4-bank
Synchronous DRAM
A0-11
A13
A14
CKOn
CKE
DCSn#
RAS#
CAS#
RD/WR#
D31-D16
DQM3#
DQM2#
A11-0
BA1
BA0
CLK
CKE
CS#
RAS#
CAS#
A11-0
BA1
BA0
CLK
CKE
CS#
RAS#
CAS#

Figure 3-19 Example of Synchronous DRAM Chip Connection (2)

3.6.4 Address Multiplexing

SDRAM can be connected without external multiplexing circuitry in accordance the address multiplex specification bits CA2~0, RA1~0 and BA in DMCR. Table 3-10 shows the relationship between the address multiplex specification bits and the bits output at the address pins.

A14-0 is used as SDRAM address. The original values are always output at these pins.

Table 3-10 SDRAM Address Multiplexing (32-bit data width) *[4]

CA2~0	RA1~0	Output Timing	A0-A9, A10, A11, A12	A13	A14	Note
8 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A21	A22	3, 4
		Row	A10-A22			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A22	A23	3, 4
		Row	A10-A22			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A10-A22			
9 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A22	A23	3, 4
		Row	A11-A23			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A11-A23			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A11-A23			
10 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A12-A24			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A12-A24			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A12-A24			
11 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A13-A25,			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A13-A25,			
	13 bits	Column	A2-A11, L/H*[1], A12-A17	A26	A27	3, 4
		Row	A13-A25,			
12 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A14-A26			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A26	A27	3, 4
		Row	A14-A26			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A27	A28	3, 4
		Row	A14-A26			

Notes:

1. L/H is a bit used in the command specification; it is fixed at L or H according to the Access mode.
2. Bank address specification
3. If one bank select signal is used (BA = 0), take A13 as bank select signal. If two bank select signals are used (BA = 1), take A13 and A14 as bank select signals
4. The A0 to A14 in table head are output pins. The A2 to A28 in table body are physical address.

Table 3-11 SDRAM Address Multiplexing (16-bit data width) *[4]

CA2~0	RA1~0	Output Timing	A0-A9, A10, A11, A12	A13	A14	Note
8 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A21	A22	3, 4
		Row	A10-A22			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A22	A23	3, 4
		Row	A10-A22			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A10-A22			
9 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A22	A23	3, 4
		Row	A11-A23			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A11-A23			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A11-A23			
10 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A12-A24			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A12-A24			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A12-A24			
11 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A13-A25,			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A13-A25,			
	13 bits	Column	A2-A11, L/H*[1], A12-A17	A26	A27	3, 4
		Row	A13-A25,			
12 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A14-A26			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A26	A27	3, 4
		Row	A14-A26			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A27	A28	3, 4
		Row	A14-A26			

Notes:

1. L/H is a bit used in the command specification; it is fixed at L or H according to the Access mode.
2. Bank address specification
3. If one bank select signal is used (BA = 0), take A13 as bank select signal. If two bank select signals are used (BA = 1), take A13 and A14 as bank select signals
4. The A0 to A14 in table head are output pins. The A2 to A28 in table body are physical address.

3.6.5 SDRAM Command

Commands for SDRAM are specified by RAS#, CAS#, RD/WR and special address signals. The processor accesses SDRAM by using the following subset of standard interface commands.

- Mode Register Set (MRS)
- Bank Activate (ACTV)
- Read (READ)
- Write (WRIT)
- Burst Terminate
- Precharge All Banks (PALL)
- Auto-Refresh (CBR)
- Enter Self-Refresh (SLFRSH)
- No Operation (NOP)

Table 3-12 SDRAM Command Encoding (Notes: 1)

Command	**Processor Pins**							
	CS#	**RAS#**	**CAS#**	**RD/WR#**	**DQM**	**A14-11, A9-0**	**A10**	**Note**
INHIBIT	H	X	X	X	X	X	X	
NOP	L	H	H	H	X	X	X	
MRS	L	L	L	L	X	Op-Code		
ACTV	L	L	H	H	X	Bank, Row	X	2
READ	L	H	L	H	L/H	Bank, Col	L	3
WRIT	L	H	L	L	L/H	Bank, Col	L	3
Burst Terminate	L	H	H	L	X	X	X	
PRE	L	L	H	L	X	Bank	L	
PALL	L	L	H	L	X	X	H	
CBR/SLFRSH	L	L	L	H	X	X	X	4

Note:

- CKE is HIGH for all commands shown except SLFRSH
- A0-A12 provides row address, and A13-A14 determines which bank is active.
- A0-A9 provides column address, and A13-A14 determines which bank is being read from or written to.
- This command is CBR if CKE is HIGH, SLFRSH if CKE is LOW.

3.6.6 SDRAM Timing

The SDRAM bank function is used to support high-speed accesses to the same row address. As SDRAM is internally divided into two or four banks, it is possible to activate one row address in each bank.

When a de-active bank is accessed, an access is performed by issuing an ACTV command following by READ or WRIT command.

When an active bank is accessed and just hit the open row, an access is performed by issuing READ or WRIT command immediately without issuing an ACTV command.

When an active bank is accessed but hit a closed row, a PRE command is first issued to precharge the bank, then the access is performed by issuing an ACTV command followed by a READ or WRIT command.

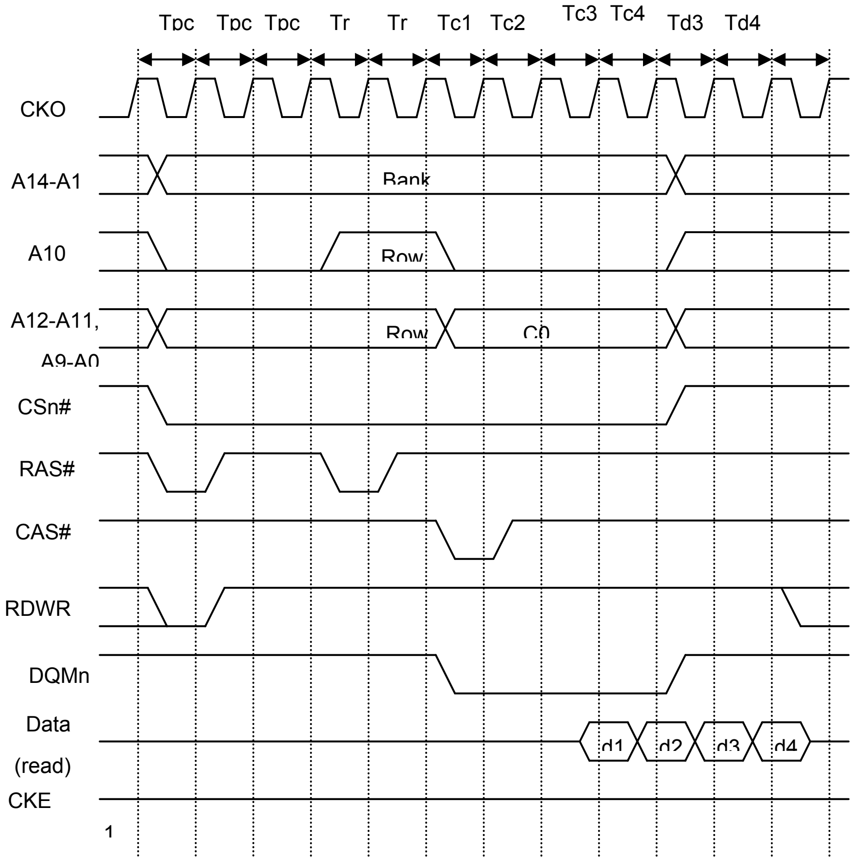
There is a limit on Tras , the time for placing each bank in the active state. If there is no guarantee that there will not be a cache hit and another row address will be accessed within the period in which this value is maintained by program execution, it is necessary to set auto-refresh and set the refresh cycle to no more than the maximum value of Tras. In this way, it is possible to observe the restrictions on the maximum active state time for each bank. If auto-refresh is not used, measures must be taken in the program to ensure that the banks do not remain active for longer than the prescribed time.

Glossary

Tr – row active cycle
Trw – row active wait cycle
Trwl – write latency cycle
Tpc – precharge cycle
TRr – refresh command cycle
Trc – RAS cycle
Trs1 – self refresh cycle 1
Trs2 – self refresh cycle 2
Trs3 – self refresh cycle 3
Trsw – self refresh wait cycle
Tc1 – command cycle 1
Tc2 – command cycle 2
Tc3 – command cycle 3
Tc4 – command cycle 4
Tc5 – command cycle 5
Tc6 – command cycle 6
Tc7 – command cycle 7
Tc8 – command cycle 8

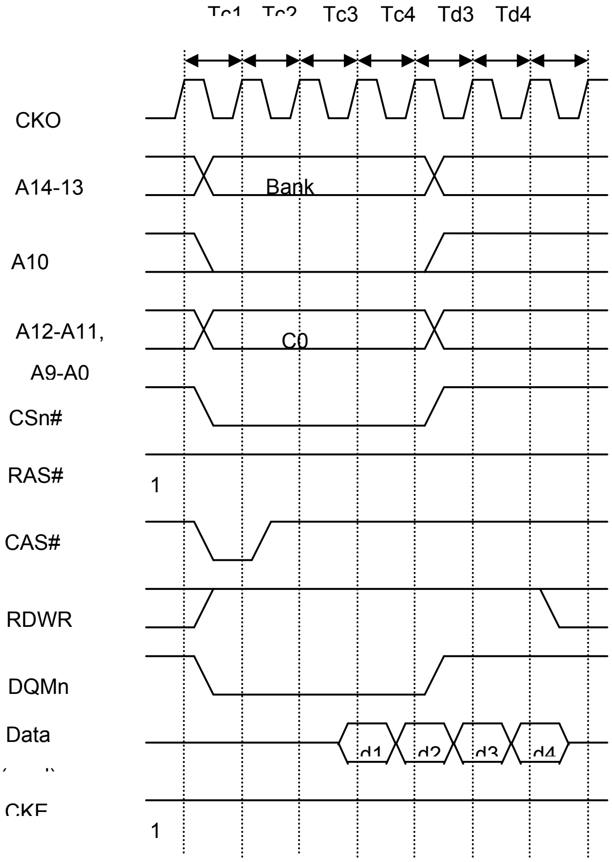
Td1 – data cycle 1
Td2 – data cycle 2
Td3 – data cycle 3
Td4 – data cycle 4
Td5 – data cycle 5
Td6 – data cycle 6
Td7 – data cycle 7
Td8 – data cycle 8
TRp1 – precharge-all cycle 1
TRp2 – precharge-all cycle 2
TRp3 – precharge-all cycle 3
TRp4 – precharge-all cycle 4
TMw1 – mode register set cycle 1
TMw2 – mode register set cycle 2
TMw3 – mode register set cycle 3
TMw4 – mode register set cycle 4

Following figures show the timing of 4-beat burst access, 8-beat burst access and single access.


Tpc
Tpc
Tpc
Tr
Tr
Tc1
Tc2
Tc3
Tc4
Td3
Td4
CKO
A14-A1
Bank
A10
Row
A12-A11,
A9-A0
Row
C0
CSn#
RAS#
CAS#
RDWR
DQMn
Data
(read)
d1
d2
d3
d4
CKE
1

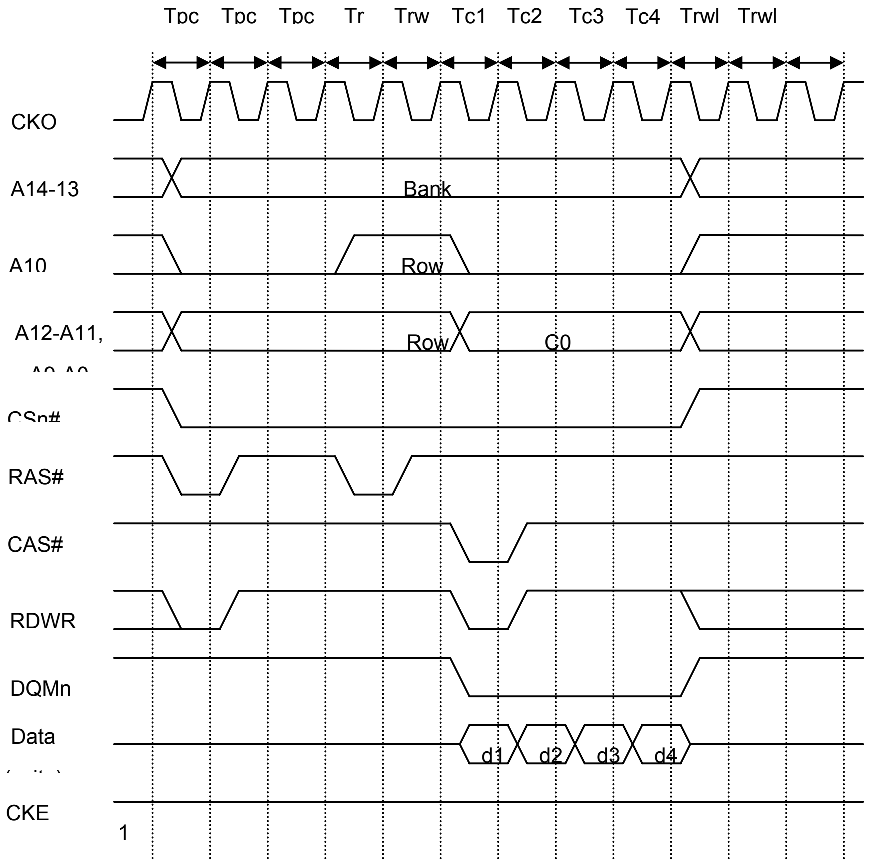
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-20 Synchronous DRAM 4-beat Burst Read Timing (Different Row)


Tc1
Tc2
Tc3
Tc4
Td3
Td4
CKO
A14-13
Bank
A10
A12-A11,
A9-A0
C0
CSn#
RAS#
1
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
CKE
1

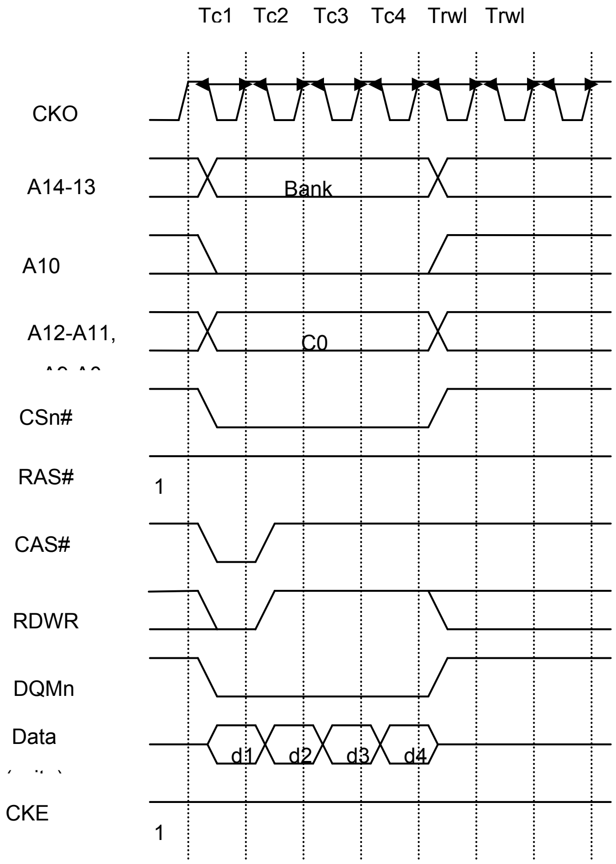
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-21 Synchronous DRAM 4-beat Burst Read Timing (Same Row)


Tpc
Tpc
Tpc
Tr
Trw
Tc1
Tc2
Tc3
Tc4
Trwl
Trwl
CKO
A14-13
Bank
A10
Row
A12-A11,
Row
C0
CSn#
RAS#
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
CKE
1

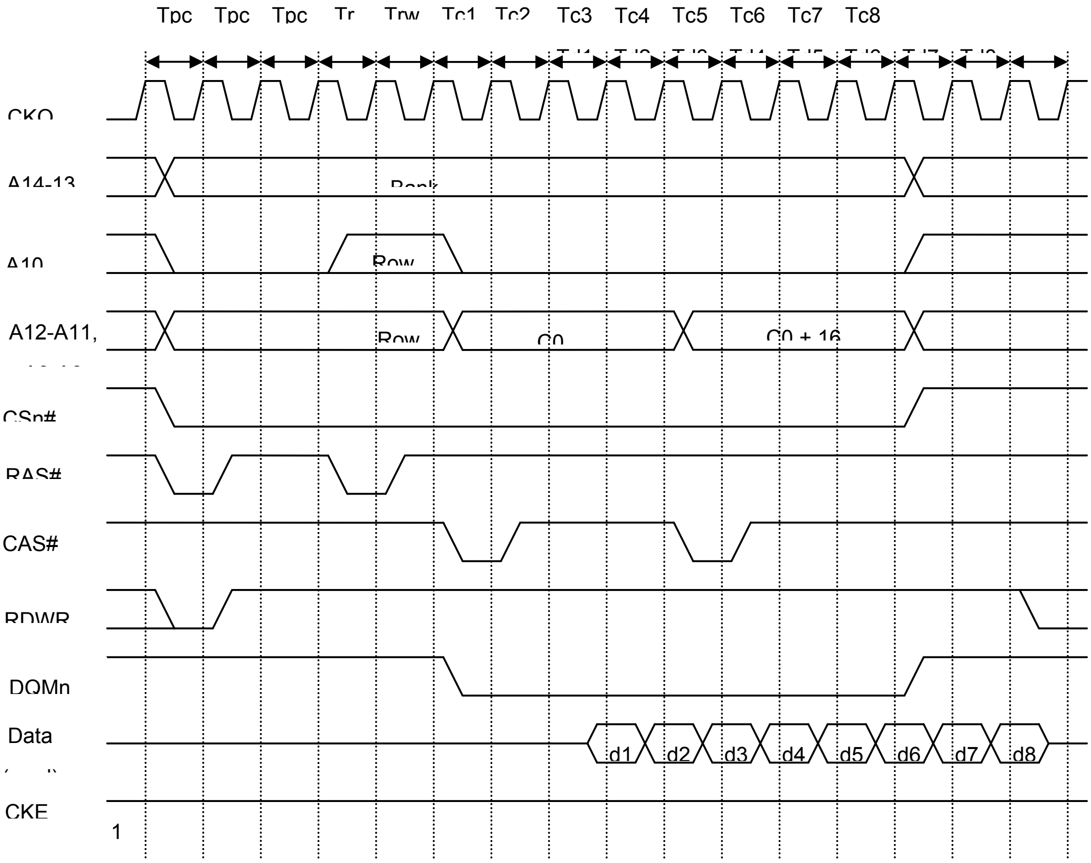
*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-22 Synchronous DRAM 4-beat Burst Write Timing (Different Row)


Tc1
Tc2
Tc3
Tc4
Trwl
Trwl
CKO
A14-13
Bank
A10
A12-A11,
C0
CSn#
RAS#
1
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
CKE
1

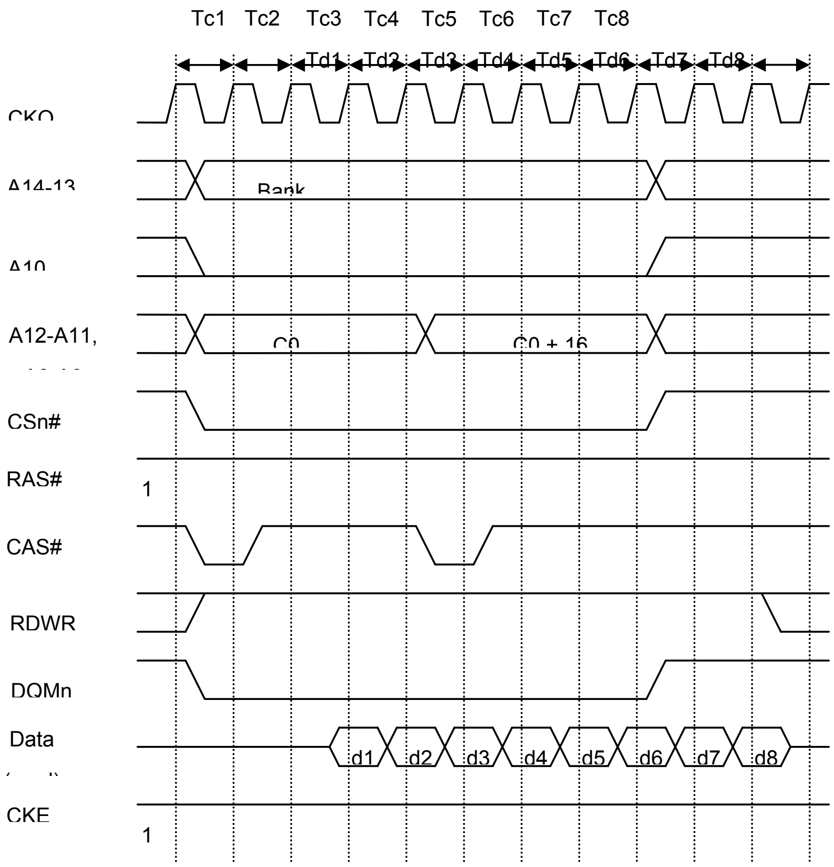
*DMCR: RCD = 1. TCL = 1. TPC = 2. TRWL = 1

Figure 3-23 Synchronous DRAM 4-beat Burst Write Timing (Same Row)


Tpc
Tpc
Tpc
Tr
Trw
Tc1
Tc2
Tc3
Tc4
Tc5
Tc6
Tc7
Tc8
CKO
A14-13
Bank
A10
Row
A12-A11,
Row
C0
C0 + 16
CSn#
RAS#
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
d5
d6
d7
d8
CKE
1

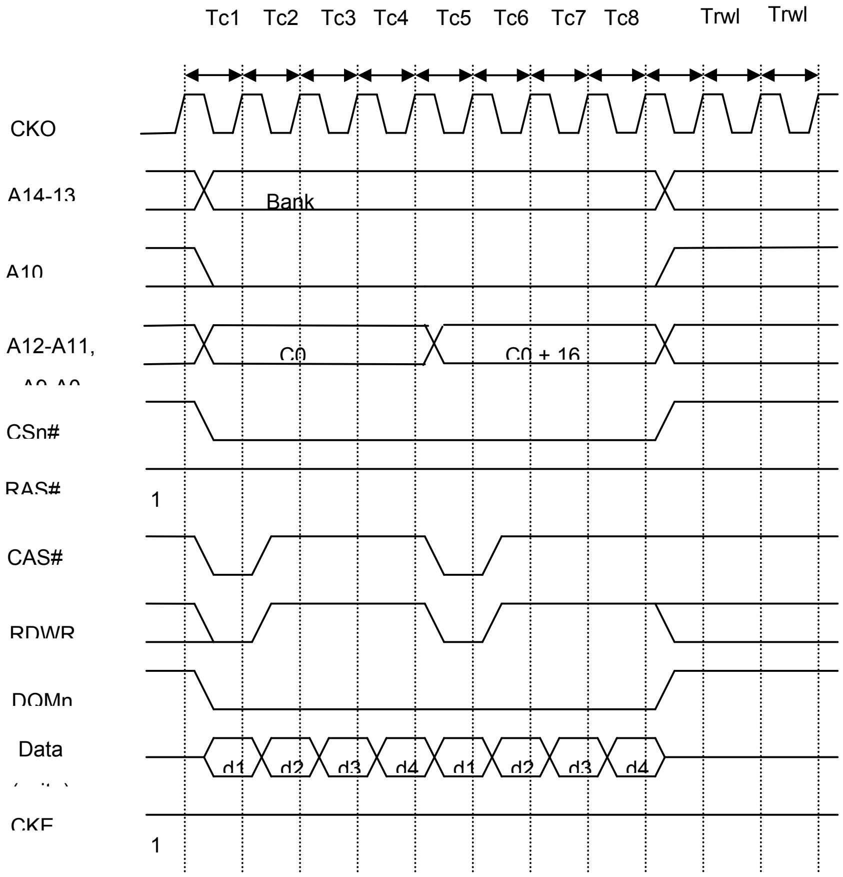
*DMCR: RCD = 1. TCL = 1. TPC = 2

Figure 3-24 Synchronous DRAM 8-beat Burst Read Timing (Different Row)


Tc1
Tc2
Tc3
Tc4
Tc5
Tc6
Tc7
Tc8
Td1
Td2
Td3
Td4
Td5
Td6
Td7
Td8
CKO
A14-13
Bank
A10
A12-A11,
C0
C0 + 16
CSn#
RAS#
1
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
d5
d6
d7
d8
CKE
1

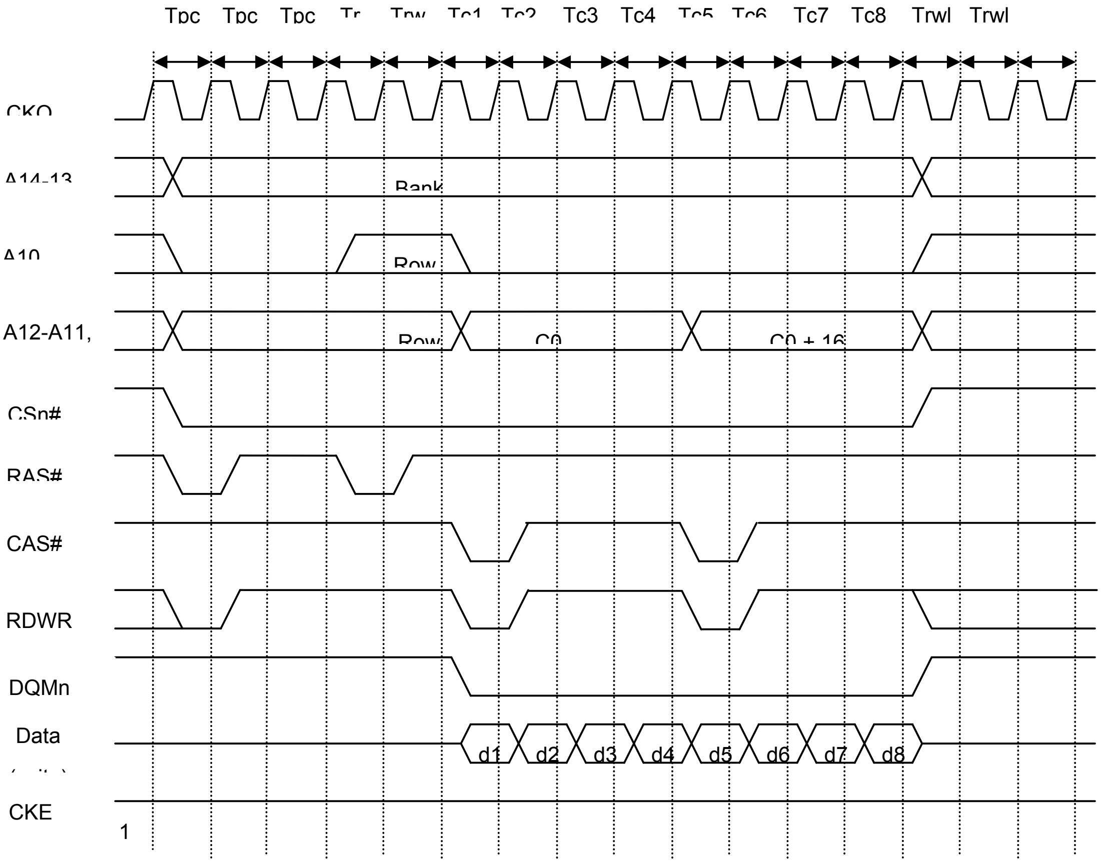
*DMCR: RCD = 1. TCL = 1. TPC = 2

Figure 3-25 Synchronous DRAM 8-beat Burst Read Timing (Same Row)


Tc1
Tc2
Tc3
Tc4
Tc5
Tc6
Tc7
Tc8
Trwl
Trwl
CKO
A14-13
Bank
A10
A12-A11,
C0
C0 + 16
CSn#
RAS#
1
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
d1
d2
d3
d4
CKE
1

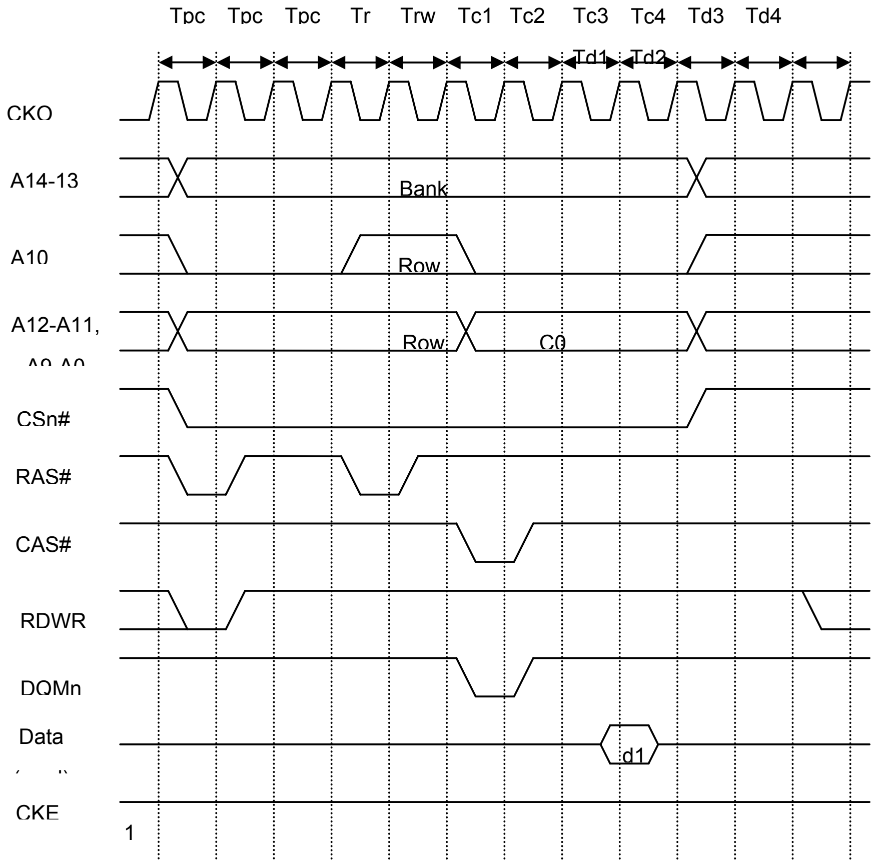
*DMCR: RCD = 1. TCL = 1. TPC = 2. TRWL = 1

Figure 3-26 Synchronous DRAM 8-beat Burst Write Timing (Same Row)


Tpc
Tpc
Tpc
Tr
Trw
Tc1
Tc2
Tc3
Tc4
Tc5
Tc6
Tc7
Tc8
Trwl
Trwl
CKO
A14-13
Bank
A10
Row
A12-A11,
Row
C0
C0 + 16
CSn#
RAS#
CAS#
RDWR
DQMn
Data
d1
d2
d3
d4
d5
d6
d7
d8
CKE
1

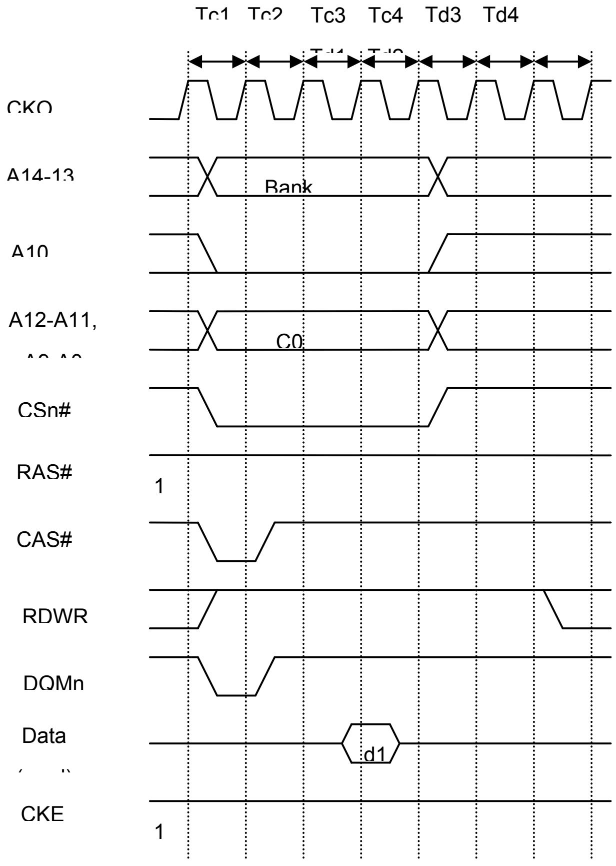
*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-27 Synchronous DRAM 8-beat Burst Write Timing (Different Row)


Tpc
Tpc
Tpc
Tr
Trw
Tc1
Tc2
Tc3
Tc4
Td3
Td4
Td1
Td2
CKO
A14-13
Bank
A10
Row
A12-A11,
Row
C0
CSn#
RAS#
CAS#
RDWR
DQMn
Data
d1
CKE
1

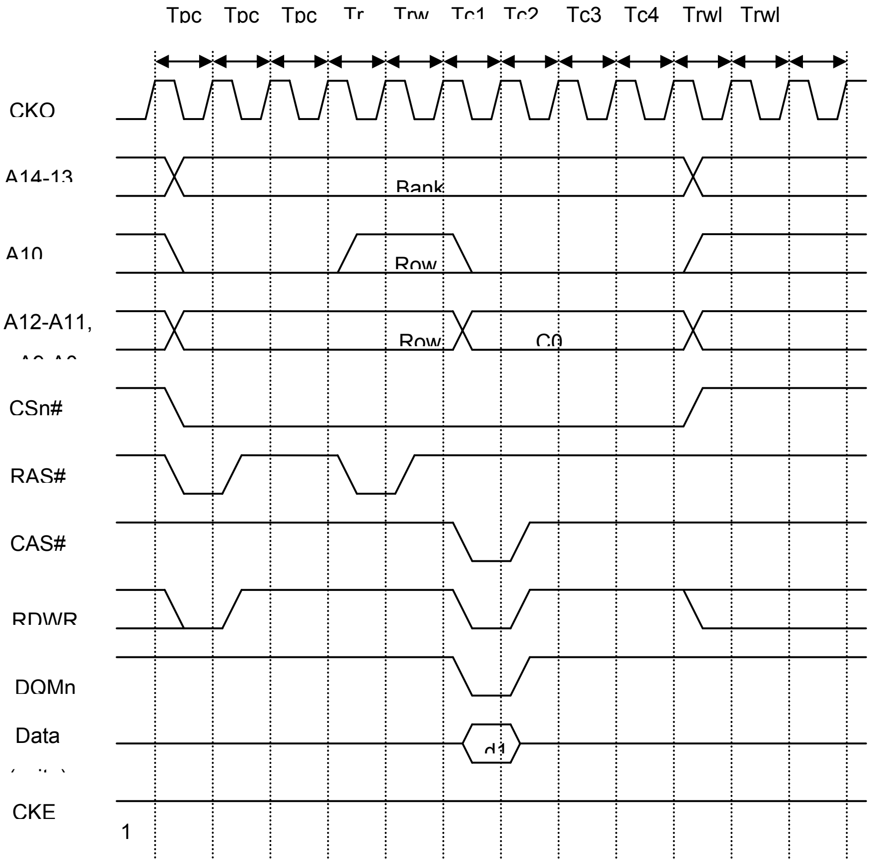
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-28 Synchronous DRAM Single Read Timing (Different Row)


Tc1
Tc2
Tc3
Tc4
Td3
Td4
CKO
A14-13
Bank
A10
A12-A11,
C0
CSn#
RAS#
1
CAS#
RDWR
DQMn
Data
d1
CKE
1

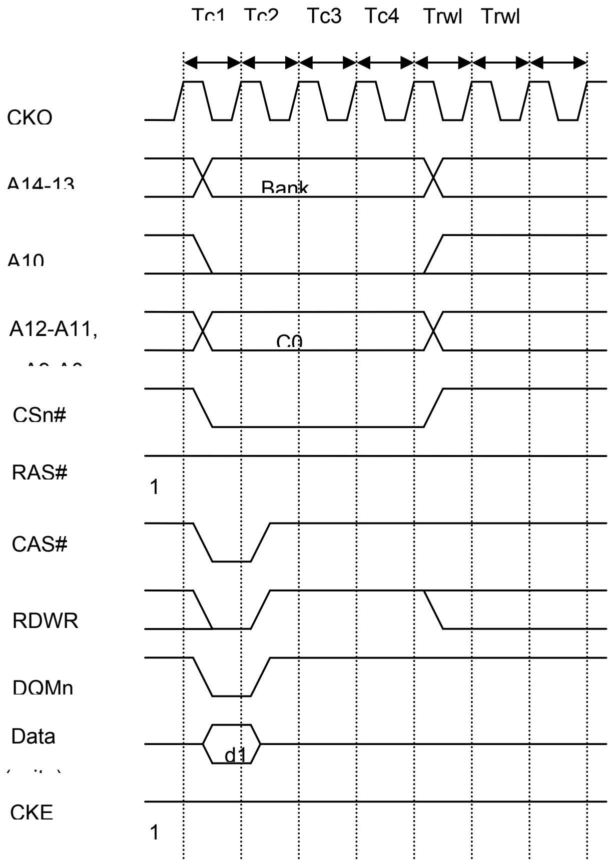
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-29 Synchronous DRAM Single Read Timing (Same Row)


Tpc
Tpc
Tpc
Tr
Trw
Tc1
Tc2
Tc3
Tc4
Trwl
Trwl
CKO
A14-13
Bank
A10
Row
A12-A11,
Row
C0
CSn#
RAS#
CAS#
RDWR
DQMn
Data
d1
CKE
1

*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-30 Synchronous DRAM Single Write Timing (Different Row)


Tc1
Tc2
Tc3
Tc4
Trwl
Trwl
CKO
A14-13
Bank
A10
A12-A11,
C0
CSn#
RAS#
1
CAS#
RDWR
DQMn
Data
d1
CKE
1

*DMCR: RCD = 1. TCL = 1. TPC = 2. TRWL = 1

Figure 3-31 Synchronous DRAM Single Write Timing (Same Row)

3.6.7 Power-Down Mode

The SDRAM power-down mode is supported to minimize the power consumption. CKE going to low level when SDRAM is idle/active state will drive SDRAM to precharge/active power-down mode. The clock supplies to SDRAM may be stopped also when CKE keep in low level more than two cycles. When a new access start or a refresh request, CKE is driven to high level and clock supplies is re-enabled. In power-down mode, clock of the accessed SDRAM bank pair is supplied. Clock of the other pair is stopped.

Following figures shows the timing of power-down mode and clock stopping.

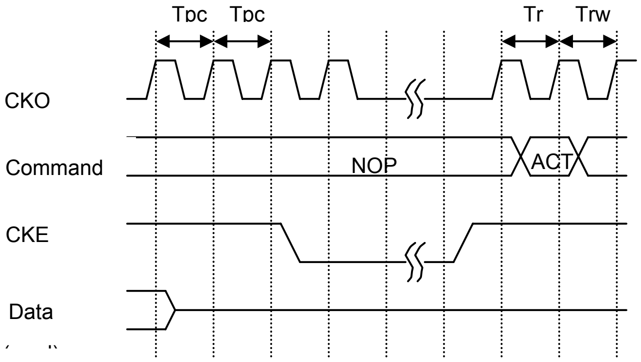


Figure 3-32 SDRAM Power-Down Mode Timing (CKO Stopped)

Following figure shows the power-down mode timing that CKE low level less than two cycles and clock is not stopped.

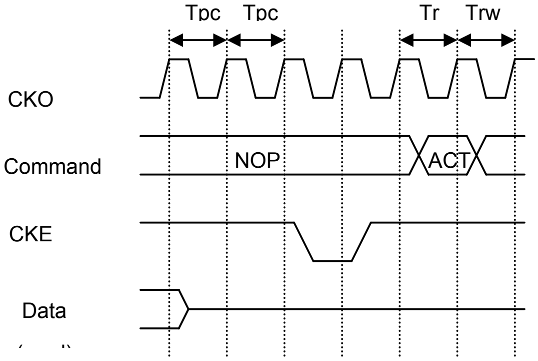


Figure 3-33 SDRAM Power-Down Mode Timing (Clock Supplied)

3.6.8 Refreshing

EMC provide a function for controlling the refresh of synchronous DRAM, Auto-refresh can be performed by clearing the RMODE bit to 0 and setting the RFSH bit to 1 in DMCR. If SDRAM is not accessed for a long period, self-refresh mode can be activated by set both the RMODE bit and the RFSH bit to 1.

3.6.8.1 AUTO-Refresh

Refreshing is performed at intervals determined by the input clock selected by bits CKS2-0 in RTCSR, and the value set in RTCOR. The value of bits CKS2-0 in RTCSR should be set so as to satisfy the refresh interval stipulation for the synchronous DRAM used. First make the settings for RTCOR, RTCNT, and the RMODE and RFSH bits in MCR, and then make the CKS2-CKS0 setting. When the clock is selected by CKS2-CKS0, RTCNT starts counting up from the value at that time. The RTCNT value is constantly compared with the RTCOR value, and if the two values are the same, a refresh request is generated and an auto-refresh is performed. At the same time, RTCNT is cleared to zero and the count-up is restarted. Figure 3-34 shows the auto-refresh cycle operation.

First, a REF command is issued in the TRr cycle. After the TRr cycle, new command output cannot be performed for the duration of the number of cycles specified by the TRC bits in DMCR. The TRC bits must be set so as to satisfy the synchronous DRAM refresh cycle time stipulation (active/active command delay time). Following figure shows the auto-refresh timing when TRC is set to 2.

Auto-refresh is performed in normal operation and sleep mode.

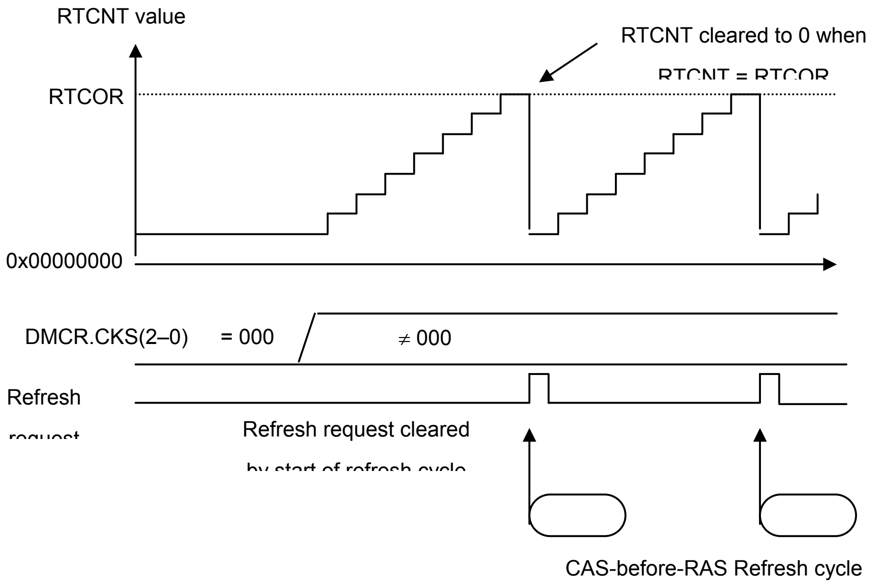
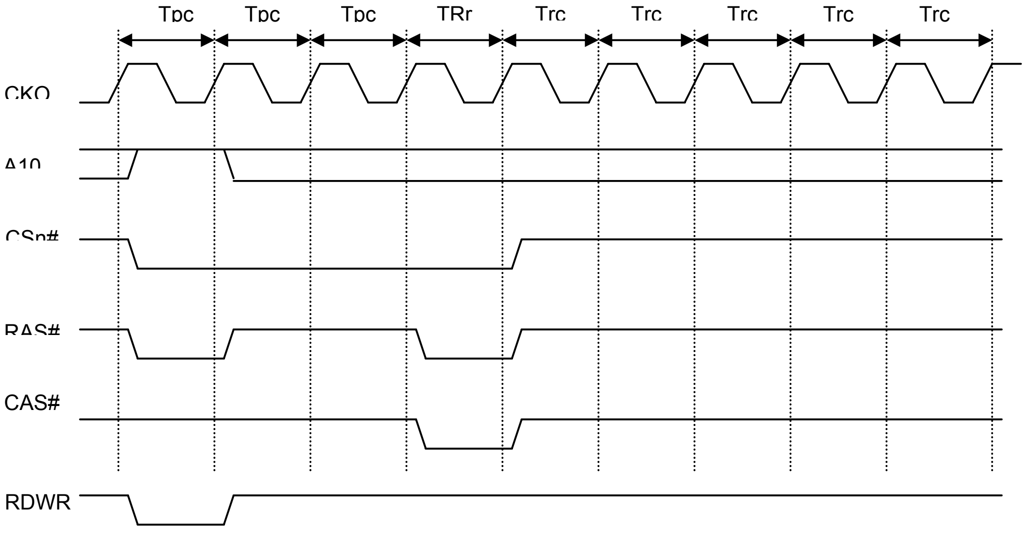


Figure 3-34 Synchronous DRAM Auto-Refresh Operation

A PALL command is issues firstly to precharge all banks. Then a REF command is issued in the TRr cycle.


Tpc
Tpc
Tpc
TRr
Trc
Trc
Trc
Trc
Trc
CKO
A10
CSn#
RAS#
CAS#
RDWR

* DMCR: TRC = 2, TPC = 2

Figure 3-35 Synchronous DRAM Auto-Refresh Timing

3.6.8.2 SELF-Refresh

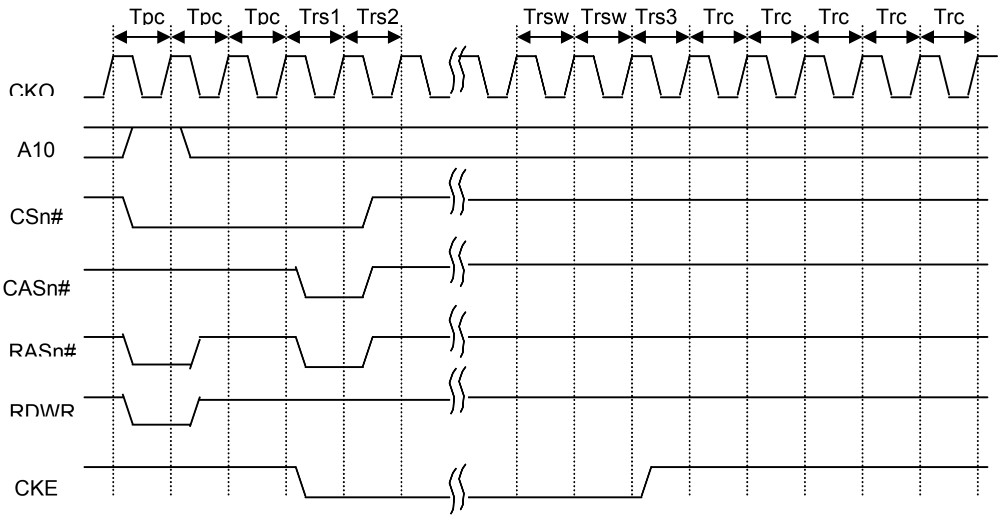
Self-refresh mode is a kind of sleep mode in which the refresh timing and refresh addresses are generated within the SDRAM. Self-refreshing is activated by setting both the RMODE bit and the RFSH bit to 1. The self-refresh state is maintained while the CKE signal is low. SDRAM cannot be accessed while in the self-refresh state. Self-refresh mode is cleared by clearing the RMODE bit to 0. After self-refresh mode has been cleared, command issuance is disabled for the number of cycles specified by the TRC bits in DMCR. Trsw cycles are inserted to meet the minimum CKE negation time specified by the TRAS bits in DMCR. Self-refresh timing is shown in following figure. Settings must be made so that self-refresh clearing and data retention are performed correctly, and auto-refresh is performed at the correct intervals. When self-refreshing is activated from the state in which auto-refreshing is set, or when exiting sleep mode other than through a reset, auto-refreshing is restarted if RFSH is set to 1 and RMODE is cleared to 0 when self-refresh mode is cleared. If the transition from clearing of self-refresh mode to the start of auto-refresh takes time, this time should be taken into consideration when setting the initial value of RTCNT. Making the RTCNT value 1 less than the RTCOR value will enable refreshing to be started immediately. After self-refreshing has been set, the self-refresh state continues even if the chip standby state is entered using the processor's sleep function, and is maintained even after recovery from sleep mode other than through a reset. In the case of a reset, the bus state controller's registers are initialized, and therefore the self-refresh state is cleared.

Self-refreshing is performed in normal operation, in idle mode and in sleep mode. In sleep mode, if RFSH bit in DMCR is 1, self-refresh is always performed in spite of RMODE field in DMCR until sleep mode is canceled.

Relationship between Refresh Requests and Bus Cycle Requests:

If a refresh request is generated during execution of a bus cycle, execution of the refresh is deferred until the bus cycle is completed. If a match between RTCNT and RTCOR occurs while a refresh is waiting to be executed, so that a new Refresh request is generated, the previous refresh request is eliminated. In order for refreshing to be performed normally, care must be taken to ensure that no bus cycle is longer than the refresh interval.

A PALL command is issued firstly to precharge all banks.


Tpc
Tpc
Tpc
Trs1
Trs2
Trsw
Trsw
Trs3
Trc
Trc
Trc
Trc
Trc
CKO
A10
CSn#
CASn#
RASn#
RDWR
CKE

* DMCR: TRAS = 0. TRC = 2

Figure 3-36 Synchronous DRAM Self-Refresh Timing

3.6.9 Initialize Sequence

In order to use SDRAM, mode setting must first be performed after powering on. To perform SDRAM initialization correctly, the EMC registers must first be set, followed by a write to the SDRAM mode register.

In SDRAM mode register setting, the address signal value at that time is latched by MRS command. If the value to be set is X, the bus state controller provides for value X to be written to the synchronous DRAM mode register by performing a write to address offset 0xA000 + X for bank 0. In this operation the data is ignored, but the mode write is performed as a byte-size access. To set burst read/write, CAS latency 2 to 3, wrap type = sequential, and burst length 4 supported by the processor, arbitrary data is written in a byte-size access to the following addresses.

Table 3-13 SDRAM Mode Register Setting Address Example (32-bit)

	Bank 0			
CAS latency 2	A022	B088	C088	D088
CAS latency 3	A032	B0C8	C0C8	D0C8

Table 3-14 SDRAM Mode Register Setting Address Example (16-bit)

	Bank 0			
CAS latency 2	A011	B044	C044	D044
CAS latency 3	A019	B064	C064	D064

The value set in DMCR.MRSET is used to select whether a Pre-charge All Banks command (PALL) or a Mode Register Set command (MRS) is issued. The timing for the Pre-charge All Banks command is shown in Figure 3-37, and the timing for the Mode Register Set command in Figure 3-38

Before mode register setting, a 200 µs idle time (depending on the memory manufacturer) must be guaranteed after powering on requested by the synchronous DRAM. If the reset signal pulse width is greater than this idle time, there is no problem in performing initialize sequence immediately.

First, a pre-CHARGE all bank (PALL) command must be issued by performing a write to address offset 0xA000 + X for bank 0, while DMCR.MRSET = 0.

Next the NUMBER of dummy auto-refresh cycles specified by the manufacturer (usually 8) or more must be executed. This is usually achieved automatically while various kinds of initialization are being performed after auto-refresh setting, but a way of carrying this out more dependably is to set a short refresh request generation interval just while these dummy cycles are being executed. With simple read or write access, the address counter in the synchronous DRAM used for auto-refreshing is not initialized, and so the cycle must always be an auto-refresh cycle.

After auto-REFRESH has been executed at least the prescribed number of times, a Mode Register Set command (MRS) is issued in the TMw1 cycle by setting DMCR.MRSET to 1 and performing a write to address offset 0xA000 + X.

An example of SDRAM operation flow is as the following:

1. Disable Bus release

   Write 0x00000000 to BCR
2. Initialize RTCOR and RTCNT for auto-refresh cycle

   Before configure SDRAM SDMR, SDRAM needs to execute auto-refresh, the number of times depends on the type of SDRAM. It's better to set a short refresh request generation interval here. For example, set RTCOR to 0x0000000F, and set RTCNT 0x00000000.
3. Initialize DMCR for Precharge all bank and auto-refresh

   When DMCR.RMODE=0 and DMCR.RFSH=1, enter auto-refresh mode;

   When DMCR.MRSET=0, write SDMR whill generates Precharge all bank cycle.

   DMCR.TPC must be defined for precharge.
4. Disable refresh counter clock

   Write 0x00000000 to RTCSR
5. Execute Precharge all bank before auto-refresh

   Because DMCR.MRSET=0, writing SDMR generates a Precharge all bank cycle, for example, write address (0x1301A000).
6. Enable fast refresh counter clock for auto-refresh cycle

   For example, write 0x00000001 to RTCSR
7. Wait for number of auto-refresh cycles (defined by SDRAM chip)

   When RTCSR.CMF=1, it indicates value of RTCOR and RTCNT match and an auto-refresh cycle occurs.
8. Configure DMCR for SDRAM MODE Register Set

   When DMCR.MRSET=1, write SDMR generate MRSET cycle.

   For example, write 0x059A5231 to DMCR, so that:

   Bus-width: 32-bit; Column Address: 9-bit; Row Address: 12-bit; Auto-refresh mode; SDMR Set mode; 4-bank; etc..
9. SDRAM Mode Register Set

   Because DMCR.MRSET=1, for example, write address 0x1301A022 to configure SDMR as:

   Burst Length: 4 burst

   Burst Type: Sequential

   CAS Latency: 2
10. Set normal auto-refresh counter clock

    For example, write 0x00000005 to RTCSR
11. Then Read/Write SDRAM can be executed

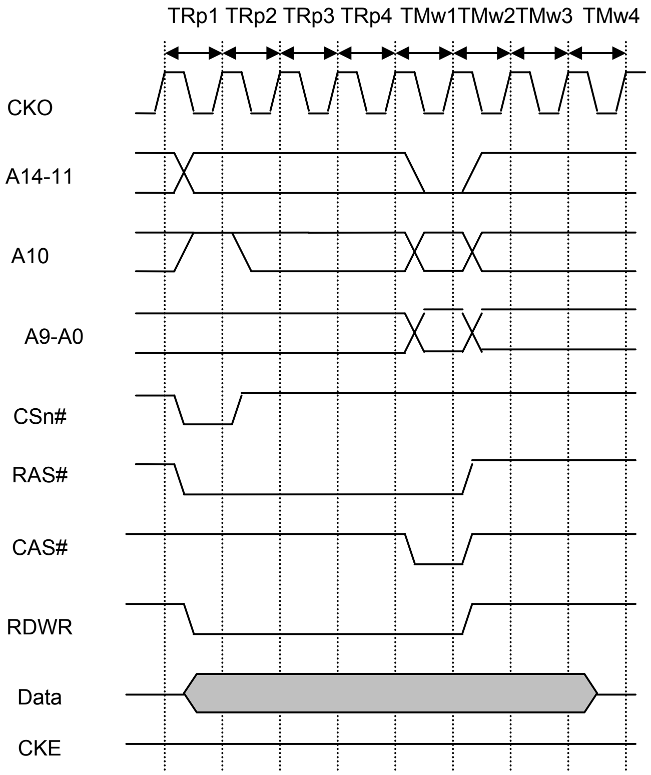

TRp1 TRp2 TRp3 TRp4 TMw1TMw2TMw3 TMw4
CKO
A14-11
A10
A9-A0
CSn#
RAS#
CAS#
RDWR
Data
CKE

Figure 3-37 SDRAM Mode Register Write Timing 1 (Pre-charge All Banks)

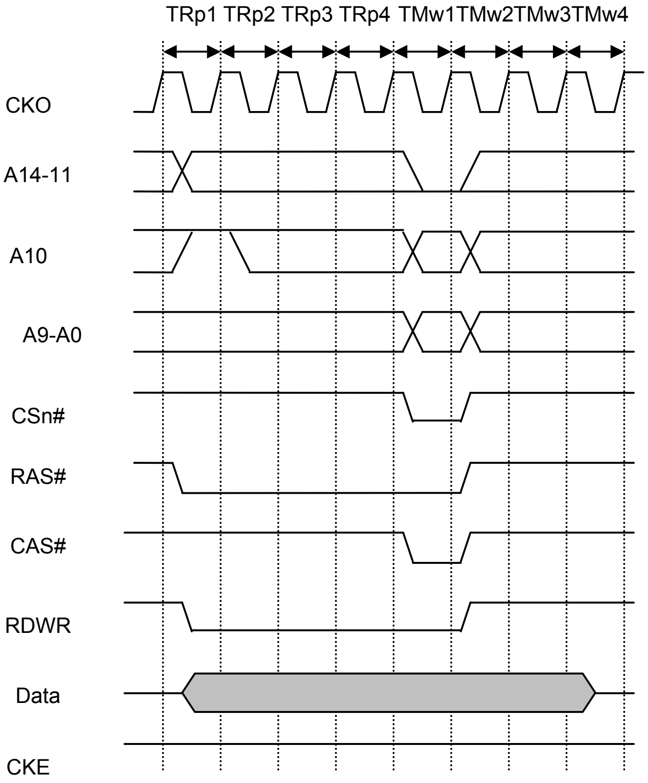

TRp1 TRp2 TRp3 TRp4 TMw1TMw2TMw3TMw4
CKO
A14-11
A10
A9-A0
CSn#
RAS#
CAS#
RDWR
Data
CKE

Figure 3-38 SDRAM Mode Register Write Timing 2 (Mode Register Set)

3.7 Bus Control Register (BCR)

BCR is used to specify the behavior of EMC on system bus and indicate the BOOT_SEL[1:0] status which defines the boot configure. It is initialized to 0x00000001 by any reset.

BOOT_SEL[1:0] pins define the boot time configurations as listed in the following table.

Table 3-15 Boot Configuration

Boot_sel[1]	Boot_sel[0]	Description
0	0	Boot from external ROM at CS3_
0	1	Boot from USB device
1	0	Boot from 512 Byte page NAND flash at CS1_
1	1	Boot from 2k Byte page NAND flash at CS1_

Name	Description	RW	Reset Value	Address	Access Width
BCR	Bus Control Register	RW	0x?0000001	0x13010000	32

BCR **0x13010000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BT_SEL		Reserved																												BRE	Endian
RST	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	Name	Description	RW
31:30	BT_SEL	**BOOT_SEL (BT_SEL[1:0]):** Status of BOOT_SEL pins that indicate the boot configure. See the above boot configuration table.	R
29:2	Reserved	Writes to these bits has no effect and always read as 0.	R
1	BRE	**Bus Release Enable**: When clear, once a transaction to EMC begins on the system bus; it must be completed before another transaction starts. When set, the system bus may be released to allow other transaction before EMC prepare the read data or be able to receipt the write data. If slow memory devices are used in the system, setting this bit will improve the efficiency of the whole system. The efficiency of SDRAM access may be improved by setting this bit. But the power consumption is increased if this bit is set. **BRE** **Description**	RW

		0 The system bus can not be released during an access (Initial value) 1 The system bus can be released during an access	
0	Endian	**Endian:** Indicates the system is little-endian.	R

4 DMA Controller

DMA controller (DMAC) is dedicated to transfer data between on-chip peripherals (MSC, AIC, UART, etc.), external memories, and memory-mapped external devices.

4.1 Features

- Support up to 6 independent DMA channels
- Descriptor or No-Descriptor Transfer
- Transfer data units: byte, 2-byte (half word), 4-byte (word), 16-byte or 32-byte
- Transfer number of data unit: 1 ~ 2^{24}
- Independent source and target port width: 8-bit, 16-bit, 32-bit
- Two channel priority modes: fixed, round robin.

4.2 Register Descriptions

Table 4-1 DMAC Registers

Name	Description	RW	Reset Value	Address	Access Size (bit)
DSA0	DMA Source Address 0	RW	0x0	0x13020000	32
DTA0	DMA Target Address 0	RW	0x0	0x13020004	32
DTC0	DMA Transfer Count 0	RW	0x0	0x13020008	32
DRT0	DMA Request Source 0	RW	0x0	0x1302000C	32
DCS0	DMA Channel Control/Status 0	RW	0x0	0x13020010	32
DCM0	DMA Command 0	RW	0x0	0x13020014	32
DDA0	DMA Descriptor Address 0	RW	0x0	0x13020018	32
DSA1	DMA Source Address 1	RW	0x0	0x13020020	32
DTA1	DMA Target Address 1	RW	0x0	0x13020024	32
DTC1	DMA Transfer Count 1	RW	0x0	0x13020028	32
DRT1	DMA Request Source 1	RW	0x0	0x1302002C	32
DCS1	DMA Channel Control/Status 1	RW	0x0	0x13020030	32
DCM1	DMA Command 1	RW	0x0	0x13020034	32
DDA1	DMA Descriptor Address 1	RW	0x0	0x13020038	32
DSA2	DMA Source Address 2	RW	0x0	0x13020040	32
DTA2	DMA Target Address 2	RW	0x0	0x13020044	32
DTC2	DMA Transfer Count 2	RW	0x0	0x13020048	32
DRT2	DMA Request Source 2	RW	0x0	0x1302004C	32
DCS2	DMA Channel Control/Status 2	RW	0x0	0x13020050	32
DCM2	DMA Command 2	RW	0x0	0x13020054	32
DDA2	DMA Descriptor Address 2	RW	0x0	0x13020058	32
DSA3	DMA Source Address 3	RW	0x0	0x13020060	32
DTA3	DMA Target Address 3	RW	0x0	0x13020064	32
DTC3	DMA Transfer Count 3	RW	0x0	0x13020068	32
DRT3	DMA Request Source 3	RW	0x0	0x1302006C	32
DCS3	DMA Channel Control/Status 3	RW	0x0	0x13020070	32
DCM3	DMA Command 3	RW	0x0	0x13020074	32
DDA3	DMA Descriptor Address 3	RW	0x0	0x13020078	32
DSA4	DMA Source Address 4	RW	0x0	0x13020080	32
DTA4	DMA Target Address 4	RW	0x0	0x13020084	32
DTC4	DMA Transfer Count 4	RW	0x0	0x13020088	32
DRT4	DMA Request Source 4	RW	0x0	0x1302008C	32
DCS4	DMA Channel Control/Status 4	RW	0x0	0x13020090	32
DCM4	DMA Command 4	RW	0x0	0x13020094	32
DDA4	DMA Descriptor Address 4	RW	0x0	0x13020098	32

DSA5	DMA Source Address 5	RW	0x0	0x130200A0	32
DDA5	DMA Target Address 5	RW	0x0	0x130200A4	32
DTC5	DMA Transfer Count 5	RW	0x0	0x130200A8	32
DRT5	DMA Request Source 5	RW	0x0	0x130200AC	32
DCS5	DMA Channel Control/Status 5	R/W	0x0	0x130200B0	32
DCM5	DMA Command 5	RW	0x0	0x130200B4	32
DDA5	DMA Descriptor Address 5	RW	0x0	0x130200B8	32
DMAC	DMA Control	R/W	0x0	0x13020300	32
DIRQP	DMA Interrupt Pending	R	0x0	0x13020304	32
DDR	DMA Doorbell	RW	0x0	0x13020308	32
DDRS	DMA Doorbell Set	W	0x0	0x1302030C	32

4.2.1 DMA Source Address (DSAn, n = 0 ~ 5)

DSA0, DSA1, DSA2, DSA3, DSA4, DSA5 **0x13020000, 0x13020020, 0x13020040, 0x13020060, 0x13020080, 0x130200a0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SA																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	SA	Source address	RW

4.2.2 DMA Target Address (DTAn, n = 0 ~ 5)

DTA0, DTA1, DTA2, DTA3, DTA4, DTA5 **0x13020004, 0x13020024, 0x13020044, 0x13020064, 0x13020084, 0x130200a4**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TA																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	TA	Target address	RW

4.2.3 DMA Transfer Count (DTCn, n = 0 ~ 5)

DTC0, DTC1, DTC2, DTC3, DTC4, DTC5 **0x13020008, 0x13020028, 0x13020048, 0x13020068, 0x13020088, 0x130200a8**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	Reserved								TC																							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	Write has no effect, read as zero	R
23:0	TC	Hold the number of data unit to transfer and it counts down to 0 at the end	RW

4.2.4 DMA Request Types (DRTn, n = 0 ~ 5)

DRT0, DRT1, DRT2, **0x1302000c, 0x1302002c, 0x1302004c,**

DRT3, DRT4, DRT5 **0x1302006c, 0x1302008c, 0x130200ac**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																											RT				
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:5	Reserved	Write has no effect, read as zero	R
4:0	RT	Transfer request type	RW

Table 4-2 Transfer Request Types

RT4-0	Description
0~7	Reserved
8	Auto-request (ignore RDIL3-0, external address → external address)
9~19	Reserved
20	UART transmit-fifo-empty transfer request (external address → UTHR)
21	UART receive-fifo-full transfer request (URBR → external address)
22	SSI transmit-fifo-empty transfer request
23	SSI receive-fifo-full transfer request
24	AIC transmit-fifo-empty transfer request
25	AIC receive-fifo-full transfer request
26	MSC transmit-fifo-empty transfer request
27	MSC receive-fifo-full transfer request
28	TCU channel n (overflow interrupt, external address→external address space)
29	SADC transfer request (SADC → external address)
30	SLCD transfer request (external address → SLCD)
31	Reserved

NOTES:

1. Only auto request can be concurrently selected in all channels with different source and target address.

2. For on-chip device DMA request except TCU, the corresponding source or target address that map to on-chip device must be set as fixed.

4.2.5 DMA Channel Control/Status (DCSn, n = 0 ~ 5)

DCS0, DCS1, DCS2,	0x13020010, 0x13020030, 0x13020050,
DCS3, DCS4, DCS5	0x13020070, 0x13020090, 0x130200b0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NDES	Reserved							CDOA								Reserved									INV	Reserved	AR	TT	HLT	CT	CTE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	NDES	Descriptor or No-Descriptor Transfer Select: 0, Descriptor Transfer; 1, No-descriptor Transfer	RW
30:24	Reserved	Write has no effect, read as zero	R
23:16	CDOA	Copy of offset address of last completed descriptor from that in DMA command register. Software could know which descriptor is just completed combining with count terminate interrupt resulted by DCSn.CT (Ignored in No-Descriptor Transfer)	RW
15:7	Reserved	Write has no effect, read as zero	R
6	INV	Descriptor Invalid error: 0, no invalid error; 1, descriptor invalid, DCMn.V bit is loaded as 0 (Ignored in No-Descriptor Transfer)	RW
5	Reserved	Write has no effect, read as zero	R
4	AR	Address Error: 0, no address error; 1, address error	RW
3	TT	Transfer Terminate: 0, No-Link Descriptor or No-Descriptor DMA transfer does not end; 1, No-Link Descriptor or No-Descriptor DMA transfer end	RW
2	HLT	DMA halt: 0, DMA transfer is in progress; 1, DMA halt	RW
1	CT	Count Terminate: 0, Link DMA transfer does not end; 1, Link DMA transfer end (Ignored in No-Descriptor Transfer)	RW
0	CTE	Channel transfer enable: 0, disable; 1, enable	RW

4.2.6 DMA Channel Command (DCMn, n = 0 ~ 5)

DCM0, DCM1, DCM2,	0x13020014, 0x13020034, 0x13020054,
DCM3, DCM4, DCM5	0x13020074, 0x13020094, 0x130200b4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								SAI	DAI	Reserved		RDIL				SP		DP		Reserved	TSZ			TM	Reserved		V	VM	VIE	TIE	LINK
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	Write has no effect, read as zero	R
23	SAI	Source Address Increment: 0, no increment; 1, increment	RW
22	DAI	Target Address Increment: 0, no increment; 1, increment	RW
19:16	RDIL	Request Detection Interval Length: Set the number of transfer unit between two requests detection in single mode. Please refer to following Table 4-3	RW
15:14	SP	Source port width: 00, 32-bit; 01, 8-bit; 10, 16-bit; 11, reserved	RW
13:12	DP	Target port width: 00, 32-bit; 01, 8-bit; 10, 16-bit; 11, reserved	RW
11	Reserved	Write has no effect, read as zero	R
10:8	TSZ	Transfer Data Size of a data unit: 000, 32-bit; 001, 8-bit; 010, 16-bit; 011, 16-byte; 100, 32-byte; others, reserved	RW
7	TM	Transfer Mode: 0, single mode; 1, block mode	RW
6:5	Reserved	Write has no effect, read as zero	R
4	V	Descriptor Valid flag: 0, Descriptor Invalid; 1, Descriptor Valid for transfer (Ignored in No-Descriptor Transfer and in Descriptor Transfer with VM=0)	R
3	VM	Descriptor Valid Mode: 0, V bit is ignored; 1, Support V bit (Ignored in No-Descriptor Transfer)	RW
2	VIE	DMA Valid Error Interrupt Enable: 0, disable; 1, enable (Ignored in No-Descriptor Transfer)	RW
1	TIE	Transfer Interrupt Enable (TIE): 0, disable interrupt; 1, enable interrupt	RW
0	LINK	Descriptor Link Enable: 0, disable; 1, enable	RW

		(Ignored in No-Descriptor Transfer)	

Table 4-3 Detection Interval Length

RDIL	Description
0	Interval length is 0
1	Interval length is 2 transfer unit
2	Interval length is 4 transfer unit
3	Interval length is 8 transfer unit
4	Interval length is 12 transfer unit
5	Interval length is 16 transfer unit
6	Interval length is 20 transfer unit
7	Interval length is 24 transfer unit
8	Interval length is 28 transfer unit
9	Interval length is 32 transfer unit
10	Interval length is 48 transfer unit
11	Interval length is 60 transfer unit
12	Interval length is 64 transfer unit
13	Interval length is 124 transfer unit
14	Interval length is 128 transfer unit
15	Interval length is 200 transfer unit

4.2.7 DMA Descriptor Address (DDAn, n = 0 ~ 5)

This register is ignored in No-Descriptor Transfer.

DDA0, DDA1, DDA2, **0x13020018, 0x13020038, 0x13020058,**

DDA3, DDA4, DDA5 **0x13020078, 0x13020098, 0x130200b8**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DBA																				DOA								Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	DBA	Descriptor Base Address	RW
11:4	DOA	Descriptor Offset Address	RW
3:0	Reserved	Write has no effect, read as zero	R

4.2.8 DMA Control

DMAC **0x13020300**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	PM	Reserved	HLT	AR	Reserved	DMAE

RST 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bits	Name	Description	RW
31:10	Reserved	Write has no effect, read as zero.	R
9:8	PM	Channel priority mode: 00, CH0 > CH1 > CH2 > CH3 > CH4 > CH5 01, CH0 > CH2 > CH3 > CH1 > CH4 > > CH5 10, CH2 > CH0 > CH1 > CH3 > CH4 > CH5 11, round robin	RW
7:4	Reserve	Write has no effect, read as zero.	R
3	HLT	Global halt status, halt occurs in any channel, the bit should set to 1. 0, no halt 1, halt occurred	RW
2	AR	Global address error status, address error occurs in any channel, the bit should be set to 1. 0, no address error 1, address error occurred	RW
1	Reserved	Write has no effect, read as zero.	R
0	DMAE	Global DMA transfer enable. 0, disable DMA channel transfer 1, enable DMA channel transfer	RW

4.2.9 DMA Doorbell (DDR)

DDR **0x13020308**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Reserved	DB5	DB4	DB3	DB2	DB1	DB0

RST 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bits	Name	Description	RW
31:8	Reserved	Write has no effect, read as zero	R
7:0	DBn	DMA Doorbell for each channel, n=0~5, for example DB0 is for DMA channel 0. Software set it to 1 and hardware clears it to 0. 0, disable DMA controller to fetch the first descriptor or DMA controller clears it to 0 as soon as it starts to fetch the descriptor 1, Write 1 to DDS will set the corresponding DBn bit to 1 and enable DMA controller to fetch the first descriptor For example, write 0x00000001 to DDS, DB0 bit is set to 1 and enable	RW

		DMA channel 0 to fetch the first descriptor. Write 0 to DDS, no meaning.	

4.2.10 DMA Interrupt Pending (DIRQP)

DIRQP **0x13020304**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										CIRQ5	CIRQ4	CIRQ3	CIRQ2	CIRQ1	CIRQ0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:6	Reserved	Write has no effect, read as zero.	R
5:0	CIRQn	CIRQn (n=0~5) denotes pending status for corresponding channel 0, no abnormal situation or normal DMA transfer is in progress 1, abnormal situation occurred or normal DMA transfer done	RW

4.3 DMA manipulation

4.3.1 Descriptor Transfer

To do proper Descriptor DMA transfer, do as following steps:

1. First of all, check whether the status of DMA controller are available, that is, for global control (DMAC), ensure that DMAC.AR=0 and DMAC.HLT=0; while for expected channels, ensure that DCSn.AR=0, DCSn.HLT=0, DCSn.TT=0, DTCn=0 and DCSn.INV=0.
2. Set DMAC.DMAE=1 and expected DCSn.CTE=1 to launch DAM transfer
3. For Descriptor transfer, guarantee DCSn.NDES=0
4. Build descriptor in memory. Write the first descriptor address in DDAn and the address must be 16Bytes aligned. The descriptor address includes two parts: Base and Offset address. If the descriptor is linked, the 32-bit address of next descriptor is composed of 20-bit Base address in DDAn and 8-bit Offset address in DES3.DOA and the four LSB is 0x0. See Table 4-4 for the detailed 4-word descriptor structure.
5. Set 1 to the corresponding bit in DDR to initiate descriptor fetch
6. Hardware clears the corresponding bit in DDR as soon as it starts to fetch the descriptor.
7. If DES0.V =0 and DES0.VM=1, DMAC stops and set DCSn.INV=1. Otherwise, it waits for dma request from peripherals to start dma transfer
8. After DMAC completes the current descriptor dma transfer, if DES0.VM=1, it clears DES0.V to 0 and writes back to memory. If DES0.Link=1, it sets DCSn.CT to 1, otherwise it sets DCSn.TT to 1. If the interrupt enabled, it will generates the corresponding interrupts.
9. If DES0.LINK=1, after DMAC completes the current descriptor dma transfer and return to fetch the next descriptor and continues dma transfer until completes the descriptor dma transfer which DES0.LINK=0.

Table 4-4 Descriptor Structure

Word	Bit	Name	Function
1st (DES0)	31	EACKS	External DMA DACKn output polarity select
	30	EACKM	External DMA DACKn output Mode select
	29-28	ERDM	External DMA request detection Mode
	27	EOPM	External DMA End of process mode
	26-24	Reserved	
	23	SAI	Source Address Increment
	22	DAI	Target Address Increment
	21-20	Reserved	
	19-16	RDIL	Request Detection Interval Length
	15-14	SP	Source port width
	13-12	DP	Target port width
	11	Reserved	
	10-8	TSZ	Transfer Data Size
	7	TM	Transfer Mode
	6-5	Reserved	
	4	V	Descriptor Valid
	3	VM	Descriptor Valid Mode
	2	VIE	Descriptor Invalid Interrupt Enable
	1	TIE	Transfer Interrupt Enable
	0	LINK	Descriptor Link Enable
2st (DES1)	31-0	DSA	Source Address
3st (DES2)	31-0	DTA	Target Address
4st (DES3)	31-24	DOA	Descriptor Offset address
	23-0	DTC	Transfer Counter

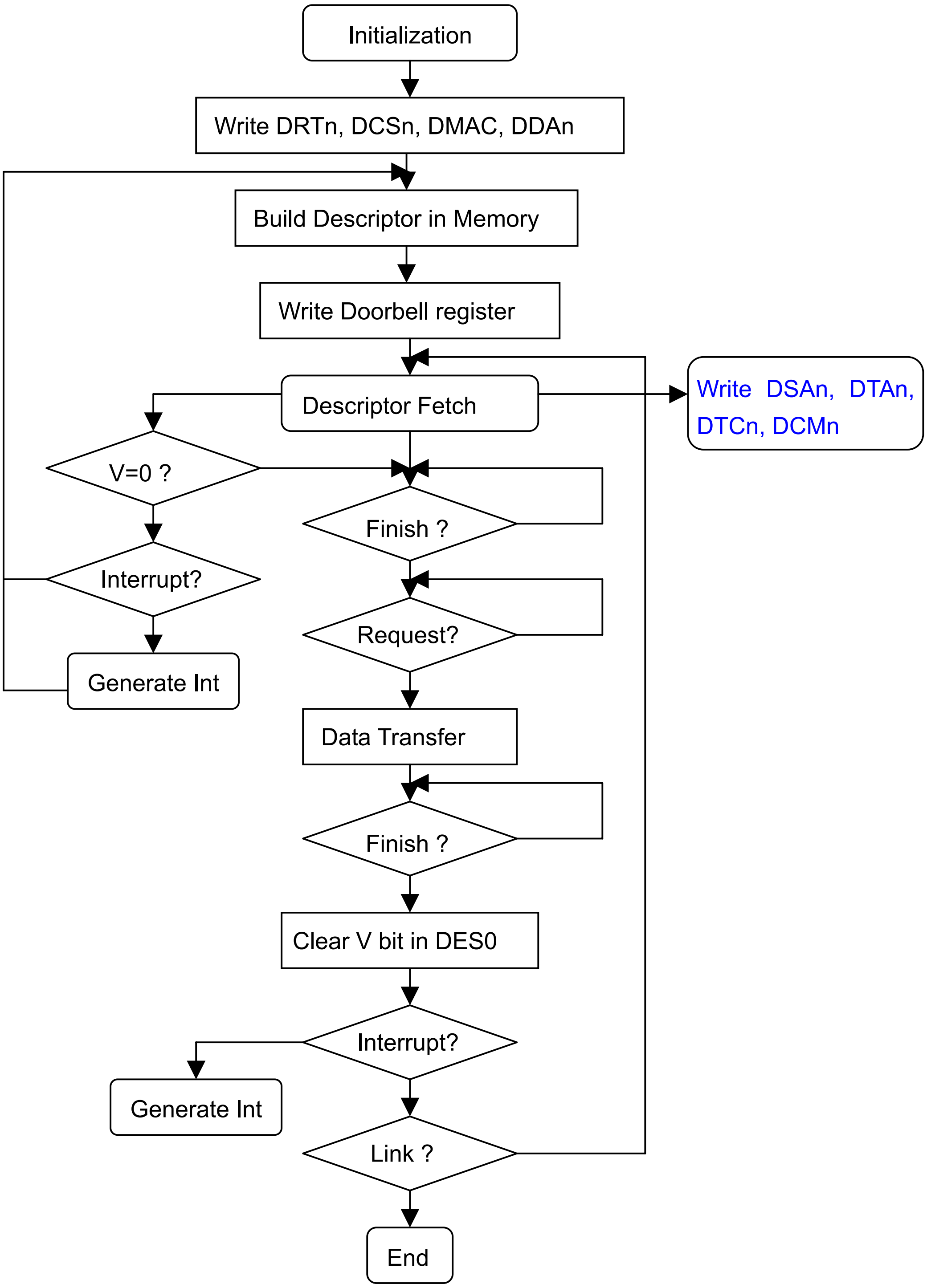

Initialization
Write DRTn, DCSn, DMAC, DDAn
Build Descriptor in Memory
Write Doorbell register
Descriptor Fetch
Write DSAn, DTAn, DTCn, DCMn
V=0 ?
Finish ?
Interrupt?
Request?
Generate Int
Data Transfer
Finish ?
Clear V bit in DES0
Interrupt?
Generate Int
Link ?
End

Figure 4-1 Descriptor Transfer Flow

4.3.2 No-Descriptor Transfer

To do proper DMA transfer, do as following steps:

1. First of all, check whether the status of DMA controller are available, that is, for global control (DMAC), ensure that DMAC.AR=0 and DMAC.HLT=0; while for expected channels, ensure that DCSn.AR=0, DCSn.HLT=0 and DCSn.TT=0 and DTCn=0.
2. For each channel n, initialize DSAn, DTAn, DTCn, DRTn, DCSn, DCMn properly
3. Set DMAC.DMAE=1 and expected DCSn.CTE=1 and DCSn.NDES=1 to launch DAM transfer

For a channel with auto-request (DRTn.RT=0x8), the transfer begins automatically when the DCSn.CTE bit and DMAC.DMAE bit are set to 1. While for a channel with other request types, the transfer does not start until a transfer request is issued and detected.

For any channel n, The DTCn value is decremented by 1 for each successful transaction of a data unit. When the specified number of transfer data unit has been completed (DTCn = 0), the transfer ends normally. Meanwhile corresponding bit of DIRQP is set to 1. If DCMn.TIE bit is set to 1, an interrupt request is sent to the CPU. However, during the transfer, if a DMA address error occurs, the transfer is suspended, both DCSn.AR and DMAC.AR are set to 1 as well as corresponding bit of DIRQP. Then an interrupt request is sent to the CPU despite of DCMn.TIE.

Sometimes, for example, an UART parity error occurs for a channel that is transferring data between such UART and another terminal. In the case, both DCSn.HLT and DMAC.HLT are set to 1 and the transfer is suspended. Software should identify halt status by checking such two bits and re-configure DMA to let DMA rerun properly later.

4.4 DMA Requests

DMA transfer requests are normally generated from either the data transfer source or target, but also they can be issued by on-chip peripherals that are neither the source nor the target. There are two DMA transfer request types: auto-request, and on-chip peripheral request. For any channel n, its transfer request type is determined through DRTn.

4.4.1 Auto Request

When there is no explicit transfer request signal available, for example, memory-to-memory transfer or memory to some on-chip peripherals like GPIO, the auto-request mode allows the DMA to automatically generate a transfer request signal internally. Therefore, when DMA initialization done, once the DMAC.DMAE and DCSn.CTE are set to 1, the transfer begins immediately in channel n which DRTn=0x8.

4.4.2 On-Chip Peripheral Request

In the mode, transfer request signals come from on-chip peripherals. All request types except 0x8 (value of DRT) belong to the mode. Both single and block transfer mode are available. Note that in single mode, the transfer byte number for one request detection according to DCMn.RDIL must be equal or less than the byte number according to receive or transmit trigger value of source or target devices.

4.5 DMA Transfer Modes

Each channel can toggles between two transfer modes: single and block

4.5.1 Single Mode

A channel with single mode will periodically detect the request signal according to presetting detection interval length (DCMn.RDIL). Moreover, during the transfer, after a transaction of a data unit (8-bit, 16-bit, 32-bit, 16-byte, or 32-byte), an internal arbitrator in the DMA will arbitrate all active channels again to select one to represent DMA's bus request to participate the AHB bus arbitration.

Above process will repeat when the channel captures the bus again until corresponding DCSn.TT bit equals to 1 or abnormal situation (address error, halt) occurs.

4.5.2 Block Mode

Once a channel with block mode captures the bus, it will do data transfer continuously until all data units are transferred or abnormal situation occurs. During the process, it does not release the bus so that neither other channels nor other bus masters can take up the bus. In the mode, the channel just detects the request signal once and corresponding DCMn.RDIL is ignored.

4.6 Channel Priorities

There are two priority modes: fixed, round robin

4.6.1 Fixed Mode

The relative channel priorities are unvaried in the mode.

- CH0 > CH1 > CH2 > CH3 > CH4 > CH5 > CH6 > CH7
- CH0 > CH2 > CH3 > CH1 > CH4 > CH6 > CH7 > CH5
- CH2 > CH0 > CH1 > CH3 > CH6 > CH4 > CH5 > CH7

4.6.2 Round Robin Mode

In the mode, there are two priority groups: CH0~CH3 and CH4~CH7. Round robin is performed in each group, and CH0~CH3 always has higher priority than CH4~CH7.

Table 4-5 Relationship among DMA Transfer connection, Request Mode and Transfer Mode

Transfer Connection	Request Mode	Transfer Mode	Data Size (bits)	Channel
External memory or memory-mapped external device and on-chip peripheral module	Auto on-chip	Block/Single	8/16/32 16-byte/32-byte	0~5

4.7 Examples

4.7.1 Memory-to-memory auto request No-Descriptor Transfer

Suppose you want to do memory move between two different memory regions through channel 3, for example, moving 1KB data from address 0x20001000 to 0x20011000, do as following steps:

a) Check if (DMAC.AR==0 && DMAC.HLT==0 && DCS3.AR==0 && DCS3.HLT==0 && DCS3.CT==0 && DCS3.NDES=1 && DTC3==0)

b) If above condition is true, set value 0 to DCS3.CTE to disable the channel 3 temporarily

c) Set source address 0x20001000 to DSA3 and target address 0x20011000 to DTA3

d) Suppose the data unit is word, set transfer count number 256 (1024/4) to DTC3

e) Set auto-request (0x8) to DRT3

f) Up to now, only the most important channel control register DCM3 is left, set it carefully:

- ➢ Set value 1 to SAI and DAI[*1]
- ➢ Ignore RDIL because in the case there is no explicit request signal can be detected
- ➢ Set word size (0) to SP and DP[*2]
- ➢ Set single mode (0) to TM[*3]
- ➢ Set value 1 to TIE to let CPU do some post process after the transfer done

g) Set value 1 to DCS3.CTE and DMAC.DMAE to launch the transfer in channels 3

h) When the transfer terminates normally (DTC3==0 && DCS3.TT==1), DIRQP.CIRQ3 will automatically be set value 1 and an interrupt request will be sent to CPU

i) When CPU grants the interrupt request, in the corresponding IRQ handler, software must clear the DCS3.CT to value 0, and the behavior will automatically clear DIRQP.CIRQ3.

NOTES:

1. Either source or target is a FIFO, must not enable corresponding address increment
2. When either source or target need be accessed through EMC (external memory controller), the real port with of the device is encapsulated by EMC, so you can set any favorite port with for it despite of the real one
3. Block mode may block bus for a long time, do not use the mode unless the data are emergency

5 Clock Reset and Power Controller

5.1 Overview

The Clock & Power management block consists of three parts: Clock control, PLL control, and Power control, Reset control.

The Clock control logic can generate the required clock signals including CCLK for CPU, HCLK for the AHB bus peripherals, and PCLK for the APB bus peripherals. This chip has one Phase Locked Loops (PLL): for CCLK, HCLK, and PCLK, MSCLK, UHCCLK, LDCLK, LPCLK. The clock control logic can make slow clocks without PLL and connect/disconnect the clock to each peripheral block by software, which will reduce the power consumption.

For the power control logic, there are various power management schemes to keep optimal power consumption for a given task. The power management block can activate four modes: NORMAL mode, DOZE mode, IDLE mode, SLEEP mode.

For reset control logic, the reset module controls or distributes all of the system reset signals.

5.2 Clock Generation UNIT

The clock generation unit (CGU) contains one PLL driven by an external oscillator and the clock generation circuit from which the following clocks are derived:

Signal	Description
CCLK	Fast clock for internal operations such as executing instructions from the cache. It can be gated during doze and idle mode when all the criteria to enter a low power are met.
HCLK	System clock—This signal appears as the HCLK input to the CPU and the HCLK to the system. This is a continuous clock (when the system is not in sleep mode) It can be gated during Sleep mode when all the criteria to enter a low power are met
PCLK	Peripheral clock – APB BUS device clock
MCLK	Clock for EMC controller
CKO	SDRAM Clock
LDCLK	LCD device clock
LPCLK	LCD pixel clock
CIM_MCLK	Clock output from CIM module
CIM_PCLK	Clock input to CIM module
I2SCLK	I2S codec clock
MSCCLK	MSC clock
SSICLK	SSI clock
EXCLK	12M clock output for UART I2C SSI TCU USB2.0-PHY

Feature:

- On-chip 2MHz~24MHZ oscillator circuit
- On-chip 32.768KHZ oscillator circuit
- One On-chip phase-locked loops (PLL) with programmable multiplier
- CCLK, PCLK, HCLK, MCLK , CKO and UHCCLK, LDCLK, LPCLK, I2SCLK, MSCCLK frequency can be changed separately for software by setting registers.
- SSI supports fast clock
- Switchable clock source
- Functional-unit clock gating

5.2.1 Pin Description

Name	I/O	Description
RTCLK_XI	Input	32.768KHZ Oscillator input signal
RTCLK_XO	Output	32.768KHZ Oscillator output signal
EXCLK	Input	Oscillator input signal
EXCLKO	Output	Oscillator output signal
CIM_MCLK	Output	Clock output from CIM module signal
CIM_PCLK	Input	Clock input to CIM module signal
LPCLK	Output	LCD pix clock signal
CKO	Output	SDRAM clock signal
MSC_CLK	Output	Clock output For MMC/SD Card signal
SSI_CLK	Output	Clock output from SSI module signal

5.2.2 CGU Block Diagram

Following figure illustrates a block diagram of CGU

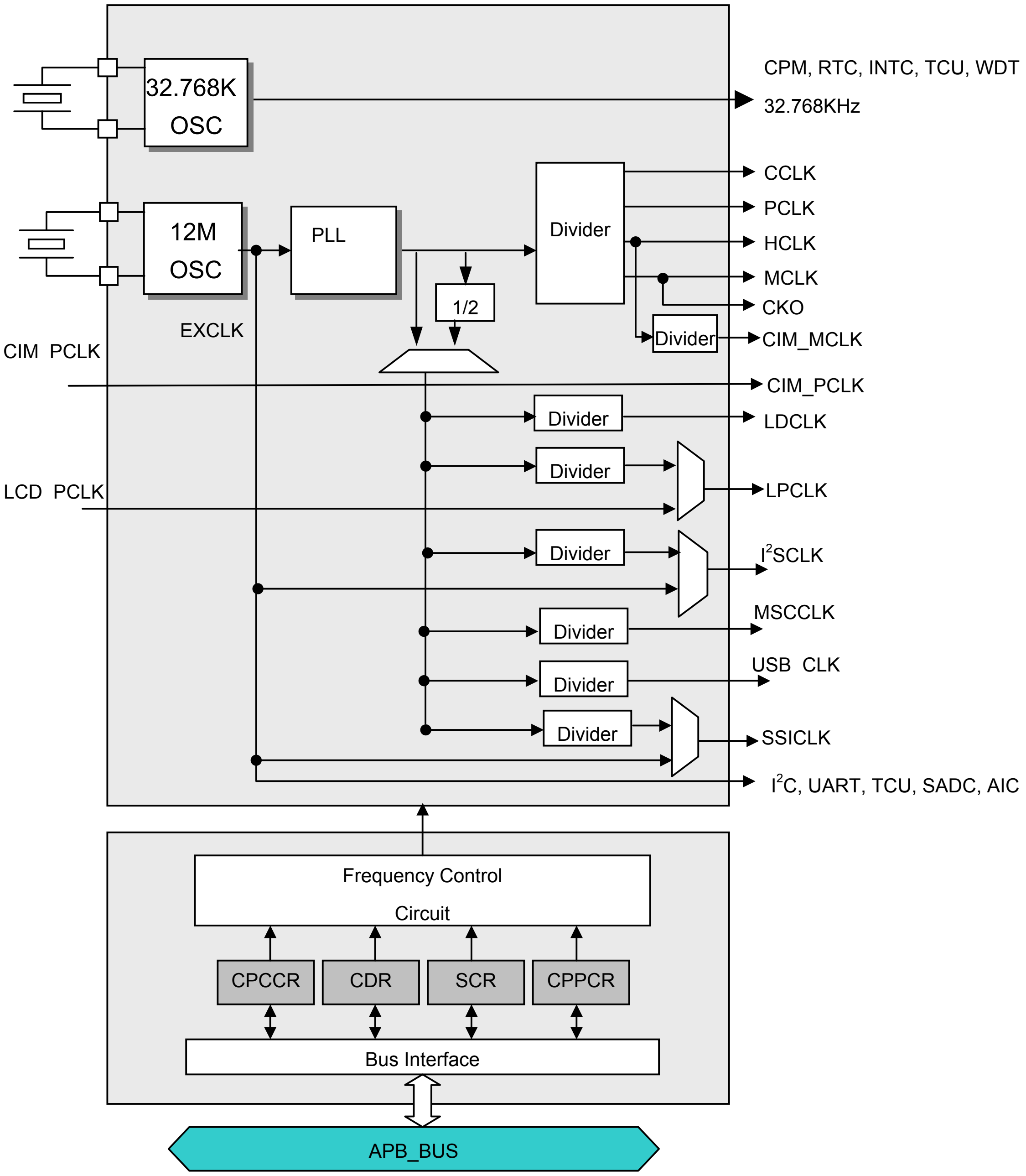

32.768K OSC
CPM, RTC, INTC, TCU, WDT
32.768KHz
12M OSC
PLL
Divider
1/2
EXCLK
CCLK
PCLK
HCLK
MCLK
CKO
Divider
CIM_MCLK
CIM PCLK
CIM_PCLK
Divider
LDCLK
Divider
LPCLK
LCD PCLK
Divider
I²SCLK
Divider
MSCCLK
Divider
USB CLK
Divider
SSICLK
I²C, UART, TCU, SADC, AIC
Frequency Control Circuit
CPCCR
CDR
SCR
CPPCR
Bus Interface
APB_BUS

Figure 5-1 CGU Block Diagram

5.2.3 Clock Overview

There is an internal PLL in this chip. PLL input clock is an external input clock EXCLK. Theoretically, EXCLK can be 2MHz ~ 20MHz.

CCLK is CPU clock. It is usually the fastest clock in the chip. This clock represents the chip speed.

HCLK is for on chip high speed peripherals connected to AHB bus.

PCLK is for on chip slow speed peripherals connected to APB bus.

MCLK is external memory bus clock. MCLK represents the SDRAM speed.

CCLK, HCLK, PCLK and MCLK are synchronous clocks that may have different frequencies. They are from the same clock source, the on chip PLL output clock in most cases. HCLK frequency can be equal to CCLK or divided CCLK by an integer. PCLK frequency can be equal to HCLK or divided HCLK by an integer. MCLK frequency can be equal to or half of HCLK.

USB device and host controllers need a 48MHz USB clock. USB clock can be selected by software divided PLL output clock.

AC97 in AIC module needs a 12.288MHz BIT clock. It is input from the external AC97 CODEC chip or other clock source.

Besides PLL input, EXCLK also provides device clock or one of device clocks for many peripherals, such as, UART, I2C, TCU, SSI and WDT

Device clock of MSC (MMC/SD) is taken from software divided PLL output clock.

LCD’s device clock and pixel clock are generated from PLL output clock, which are divided by two independent dividers.

The slowest clock is RTCLK, which is usually 32768Hz.

5.2.4 CGU Registers

All CGU register 32bit access address is physical address.

Table 5-1 CGU Registers Configuration

Name	description	RW	Reset Value	Address	Access Size
CPCCR	Clock Control Register	RW	0x42040000	0x10000000	32
CPPCR	PLL Control Register	RW	0x28080011	0x10000010	32
I2SCDR	I2S device clock divider Register	RW	0x00000004	0x10000060	32
LPCDR	LCD pix clock divider Register	RW	0x00000004	0x10000064	32
MSCCDR	MSC device clock divider Register	RW	0x00000004	0x10000068	32
UHCCDR	UHC 48M clock divider Register	RW	0x00000004	0x1000006C	32
UHCTST	UHC PHY test point register	RW	0x00000000	0x10000070	32
SSICDR	SSI clock divider Register	RW	0x00000004	0x10000074	32

5.2.4.1 Clock Control Register

The Clock Control Register (CPCCR) is a 32-bit read/write register, which controls CCLK, HCLK, PCLK, MCLK and LDCLK division ratios. It is initialized to 0x42040000 by any reset. Only word access can be used on CPCCR.

CPCCR **0x10000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I2CS	CKOEN	UCS	UDIV						CE	PCS	LDIV					MDIV				PDIV				HDIV				CDIV			
RST	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	I2CS	I2S Clock Source Selection. Selects the I2S clock source between PLL output and pin EXCLK. 0: I2S clock source is EXCLK 1: I2S clock source is PLL output divided by I2SDIV	RW
30	CKOEN	CKO Output Enable. Controls the output of CKO 0: Disable CKO output. CKO is Hi-Z 1: Enable CKO output	RW
29	UCS	USB Clock Source Selection. Selects the USB clock source between PLL output and pin EXCLK. 0: USB clock source is pin EXCLK 1: USB clock source is PLL output	RW

<table>
<tr><td>28:23</td><td>UDIV</td><td>Divider for USB Clock Frequency. When USB clock source is PLL (UCS bit is 1), this field specified the USB clock division ratio, which varies from 1 to 64 (division ratio = UDIV + 1).</td><td>RW</td></tr>
<tr><td>22</td><td>CE</td><td>change enable. If CE is 1, writes on CDIV, HDIV, PDIV, MDIV, UDIV, PXDIV or LDIV will start a frequency changing sequence immediately. When CE is 0, writes on CDIV, HDIV, PDIV, MDIV, UDIV, PXDIV and LDIV will not start a frequency changing sequence immediately. The division ratio is actually updated in PLL multiple ratio changing sequence or PLL Disable Sequence.
0: Division ratios are updated in PLL multiple ratio changing sequence or PLL Disable Sequence
1: Division ratios are updated immediately</td><td>RW</td></tr>
<tr><td>21</td><td>PCS</td><td>PLL out clock source clock selection. It supplies source clock for MSC I2S LCD USB
0: divider clock source is PLL output divided by 2
1: divider clock source is PLL output</td><td>RW</td></tr>
<tr><td>20:16</td><td>LDIV</td><td>Divider for LCD device Clock Frequency. Specified the LCLK division ratio, which varies from 1 to 32 (division ratio = LDIV + 1). The frequency of LCLK must be equal to or less than 150 MHz.</td><td>RW</td></tr>
<tr><td>15:12</td><td>MDIV</td><td>Divider for Memory Clock Frequency. Specified the MCLK division ratio.
<table>
<tr><th colspan="4">Bit 15~12: MDIV</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>X1/2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>X1/3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>X1/4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>X1/6</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>X1/8</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>X1/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>X1/16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>X1/24</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>X1/32</td></tr>
<tr><td colspan="4">Other Value</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
<tr><td>11:8</td><td>PDIV</td><td>Divider for Peripheral Clock Frequency. Specified the PCLK division ratio.
<table>
<tr><th colspan="4">Bit 11~8: PDIV</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>X1/2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>X1/3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>X1/4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>X1/6</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>X1/8</td></tr>
</table></td><td>RW</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>X1/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>X1/16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>X1/24</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>X1/32</td></tr>
<tr><td colspan="4">Other Value</td><td>Reserved</td></tr>
</table>
</td><td></td></tr>
<tr><td>7:4</td><td>HDIV</td><td>Divider for System Clock Frequency. Specified the HCLK division ratio.
<table>
<tr><td colspan="4">Bit 7~4: HDIV</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>X1/2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>X1/3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>X1/4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>X1/6</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>X1/8</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>X1/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>X1/16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>X1/24</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>X1/32</td></tr>
<tr><td colspan="4">Other Value</td><td>Reserved</td></tr>
</table>
</td><td>RW</td></tr>
<tr><td>3:0</td><td>CDIV</td><td>Divider for CPU Clock Frequency. Specifies the CCLK division ratio.
<table>
<tr><td>BIT 3~0</td><td>Description</td><td>BIT 3~0</td><td>description</td></tr>
<tr><td>0 0 0 0 :</td><td>X1</td><td>0 0 0 1 :</td><td>X1/2</td></tr>
<tr><td>0 0 1 0 :</td><td>X1/3</td><td>0 0 1 1 :</td><td>X1/4</td></tr>
<tr><td>0 1 0 0 :</td><td>X1/6</td><td>0 1 0 1 :</td><td>X1/8</td></tr>
<tr><td>0 1 1 0 :</td><td>X1/12</td><td>0 1 1 1 :</td><td>X1/16</td></tr>
<tr><td>1 0 0 0 :</td><td>X1/24</td><td>1 0 0 1 :</td><td>X1/32</td></tr>
<tr><td colspan="3">Other Value</td><td>Reserved</td></tr>
</table>
</td><td>RW</td></tr>
</table>

5.2.4.2 I2S device clock divider Register

I2S device clock divider Register (I2SCDR) is a 32-bit read/write register that specifies the divider of I2S device clock . This register is initialized to 0x00000004 only by any reset. Only word access can be used on I2SCDR

I2SCDR **0x10000060**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																							I2SCDR								
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
31:9	Reserved	Writes to these bits have no effect and always read as 0.	R
8:0	I2SCDR	Divider for I2S Frequency. Specified the I2S device clock division ratio, which varies from 1 to 512 (division ratio = I2SCDR + 1).	RW

5.2.4.3 LCD pix clock divider Register

LCD pix clock divider Register (LPCDR) is a 32-bit read/write register that specifies the divider of LCD pixel clock (LPCLK). This register is initialized to 0x00000004 only by any reset. Only word access can be used on LPCDR

LPCDR **0x10000064**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCS	Reserved																				LPCDR										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
31	LCS	LCD Pixel Clock Source Selection. Selects the LCD pixel clock source between divider and external clock input 0: LCD pixel clock source is divider output 1: LCD pixel clock source is LCD_PCLK pin	RW
30:11	Reserved	Writes to these bits have no effect and always read as 0.	R
10:0	LPCDR	Divider for Pixel Frequency. Specified the LCD pixel clock (LPCLK) division ratio, which varies from 1 to 2048 (division ratio = LPCDR + 1).	RW

5.2.4.4 MSC device clock divider Register

MSC device clock divider Register (MSCCDR) is a 32-bit read/write register that specifies the divider of MSC device clock . This register is initialized to 0x00000004 only by any reset. Only word access can be used on MSCCDR

MSCCDR **0x10000068**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																											MSCCDR				

RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and always read as 0.	R
4:0	MSCCDR	Divider for MSC Frequency. Specified the MSC device clock division ratio, which varies from 1 to 32 (division ratio = MSCCDR + 1).	RW

5.2.4.5 UHC device clock divider Register

UHC device clock divider Register (UHCCDR) is a 32-bit read/write register that specifies the divider of UHC 48M device clock . This register is initialized to 0x00000004 only by any reset. Only word access can be used on UHCCDR

UHCCDR **0x1000006C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												UHCCDR			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
31:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3:0	UHCCDR	Divider for UHC Frequency. Specified the UHC 48M device clock division ratio, which varies from 1 to 16 (division ratio = UHCCDR + 1).	RW

5.2.4.6 UHC PHY test point Register

UHC PHY test point Register (UHCTST) is a 32-bit read/write register. This register is initialized to 0x00000000. only by any reset. Only word access can be used on UHCTST

UHCTST **0x10000070**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								UHTST								
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	0	0	0	1	0	0

Bits	Name	Description	RW
31:9	Reserved	Writes to these bits have no effect and always read as 0.	R
8	rcvdata	UHC PHY rcvdata input	R

7	dm	UHC PHY dm input	R
6	dp	UHC PHY dp input	R
5	Testmod	UHC test mode	RW
4	txenl		RW
3	speed		RW
2	Txse0		RW
1	txdm		RW
0	txdp		RW

5.2.4.7 SSI device clock divider Register

SSI device clock divider Register (SSICDR) is a 32-bit read/write register that specifies the divider of SSI device clock . This register is initialized to 0x00000004 only by any reset. Only word access can be used on SSICDR

SSICDR **0x10000074**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCS	Reserved																											SSICDR			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
31	SCS	SSI Clock Source Selection. Selects the SSI clock source between PLL output and pin EXCLK. 0: SSI clock source is pin EXCLK 1: SSI clock source is PLL output	RW
30:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3:0	SSICDR	Divider for SSI Frequency. Specified the SSI device clock division ratio, which varies from 1 to 16 (division ratio = SSICDR + 1).	RW

5.2.4.8 PLL Control Register

The PLL Control Register (CPPCR) is a 32-bit read/write register, which controls PLL multiplier, on/off state and stabilize time. It is initialized to 0x28080011 only by any reset. Only word access can be used on CPPCR.

CPPCR **0x10000010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PLLM									PLLN					PLLOD		Reserved					PLLS	PLLBP	PLLEN	PLLST							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:23	PLLM	the PLL feedback 9-bit divider	RW
22:18	PLLN	the PLL input 5-bit divider	RW
17:16	PLLOD	00: divide by 1 01: divide by 2 10: divide by 2 11: divide by 4	RW
15:11	Reserved	Writes to these bits have no effect and always read as 0	R
10	PLLS	PLL Stabilize Flag 0: PLL is off or not stable 1: PLL is on and stable	R
9	PLLBP	PLL Bypass. If PLLEN is 1, set this bit to1 will bypass PLL. The PLL is still running background but the source of associated dividers is switched to 12-M. If PLLEN is 0, set this bit to 1 has no effect. If PLLEN is 1, clear this bit to 0 will switch the source of associated dividers to PLL output.	RW
8	PLLEN	PLL Enable. When PLLEN is set to 1, PLL starts to lock phase. After PLL stabilizes, PLLS bit is set. If PLLBP is 0, the source of associated dividers, is switched to PLL output. When PLLEN is clear to 0, PLL is shut off and the source of associated dividers is switched to 12-MHz in spite of PLLBP bit	RW
7:0	PLLST	PLL Stabilize Time. Specifies the PLL stabilize time by unit of RTCCLK (approximate 32kHz) cycles. It is used when change PLL multiplier or change PLL from off to on. It is initialized to H'11	RW

5.2.5 PLL Operation

The PLL developed as a macro cell for clock generator. It can generate a stable high-speed clock from a slower clock signal. The output frequency is adjustable and can be up to 500MHz. The PLL integrates a phase frequency detector (PFD), a low pass filter (LPF), a voltage controlled oscillator (VCO) and other associated support circuitry. All fundamental building blocks as well as fully programmable dividers are integrated on the core. It is useful for clock multiplication of stable crystal oscillator sources and for de-skew clock signals.

The PLL block diagram is shown in following figure

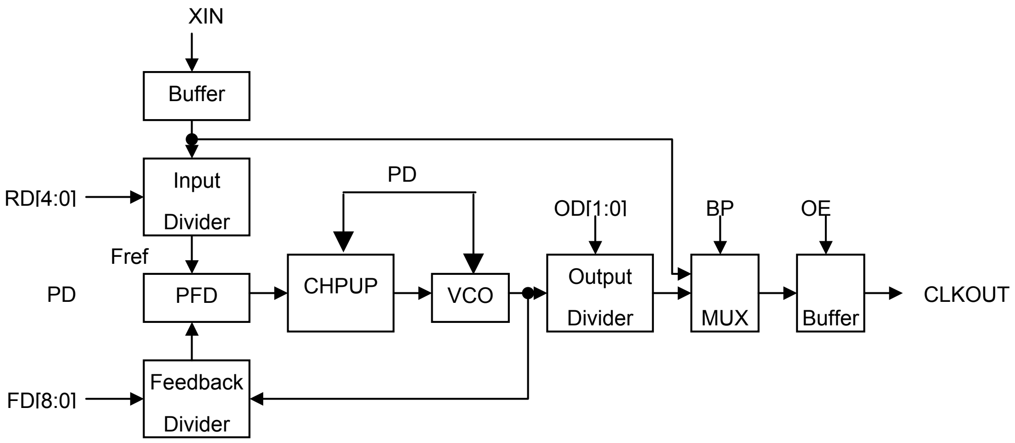

XIN
Buffer
RD[4:0]
Input Divider
Fref
PD
PFD
Feedback Divider
FD[8:0]
CHPUP
PD
VCO
OD[1:0]
Output Divider
BP
MUX
OE
Buffer
CLKOUT

Figure 5-2 Block Diagram of PLL

5.2.5.1 PLL Configuration

- PLL Divider Value Setting

  There are 3 divider values (N, M and NO) to set the PLL output clock frequency CLKOUT:

  - Input Divider Value N

    N = PLLN of CPPCR + 2

  - Feedback Divider Value M

    M = PLLM of CPPCR + 2

  - Output Divider Value NO

Output Divider Setting (OD)	Output Divider Value (NO)
0	1

1	2
2	2
3	4

- The PLL output frequency, CLK_OUT, is determined by the ratio set between the value set in the input divider and the feedback divider. PLL output frequency CLK_OUT is calculated from the following equations:

CLKOUT = XIN x (M / N) x (1/ NO)

M = F0 *1 + F1*2 + F2*4 + F3*8 + F4*16 + F5*32 + F6*64 + F7*128 + F8*256 + 2
N = R0*1 + R1*2 + R2*4 + R3*8 + R4*16 + 2
NO = $2^{od0+od1}$

Where:
CLK_OUT represents the output frequency
XIN represents PLL input frequency
N represents input divider value
M represents feedback divider value
NO represents output divider value

< Attention >
1. 1MHZ ≤ XIN/N ≤ 15MHZ
2. 100MHZ ≤ CLK_OUT x NO ≤ 500MHZ

5.2.5.2 PLL out clock frequency selection

PLL-freq = PLL-freq-raw / NO, where NO = 1, 2, 4
PLL-freq-raw = EXCLK * M / N, where M = integer of 2 ~ 513, N = integer of 2 ~ 33

So, to generate a specified PLL-freq, there are many valid sets of NO, M and N value.

Smaller PLL-freq-raw is better since it consumes less power. Reduce PLL-freq-raw from 200MHz to 100MHz saving a few milliwatts. Please beware not put PLL-freq-raw less than 100MHz.

If EXCLK is in small jitter, like a crystal-generated clock, a smaller N is better.

5.2.6 Main Clock Division Change Sequence

Main clock (CCLK, HCLK, PCLK and MCLK) frequencies can be changed separately or simultaneously by changing division ratio. Following conditions must be obeyed:

- CCLK must be integral multiple of HCLK
- The frequency ratio of CCLK and HCLK can not be 24 and 32

- HCLK must be equal to MCLK or twice of MCLK
- HCLK and MCLK must be integral multiple of PCLK.

Don't violate this limitation, otherwise unpredictable error may occurs.

In normal mode, if CE bit of CPCCR is 1, changing CDIV, HDIV, PDIV or MDIV will start a Division Change Sequence immediately. If CE bit of CPCCR is 0, changing CDIV, HDIV, PDIV or MDIV will not start Division Change Sequence.

5.2.7 Change Other Clock Frequencies

The divider of LCD device clock (LDCLK), LCD pixel clock (LPCLK), I2S device clock, SSI device clock, MSC device clock and USB clock can be changed by programming LDIV, LPCDR , I2SCDR, SSICDR, UHCDR, MSCCDR and UDIV, respectively.

Change LDIV LPCDR I2SCDR SSICDR UHCCDR MSCCDR and UDIV as following steps:

1.3 Stop related devices with clock-gate function. Clock supplies to the devices are stopped.
1.3 Change LDIV, LPCDR, I2SCDR, SSICDR , UHCCDR MSCCDR or UDIV. If CE is 1, clock frequencies are changed immediately. If CE is 0, clock frequencies are not changed until PLL Multiplier Change Sequence is started.
1.3 Cancel above clock-gate function.

5.2.8 Change Clock Source Selection

USB, I2S ,SSI device clocks and LCD pix clock can be selected from two sources. Before change clock source, corresponding devices should be stopped using clock-gate function.

- When USB clock source is changed (UCS bit of CPCCR), USB clock should be stopped.
- When I2S clock source is changed (I2CS bit of CPCCR), AIC should be stopped.
- When SSI clock source is changed (SCS bit of SSICDR), SSI should be stopped
- When LCD pix clock source is changed (LCS bit of LPCDR), LCD should be stopped

When UCS, I2CS ,SCS LCS bit is changed, clock source is changed immediately.

5.2.9 EXCLK Oscillator

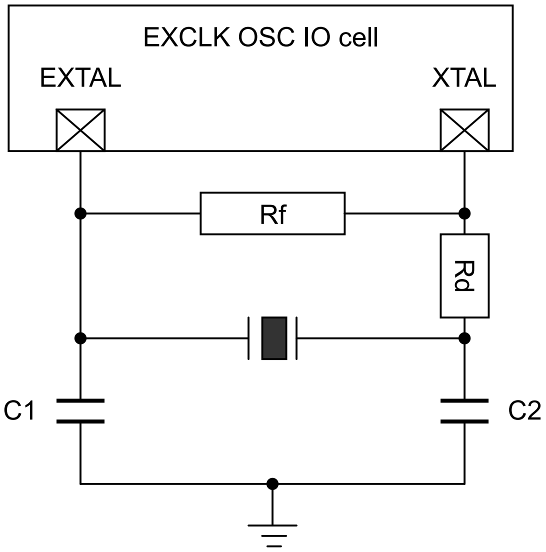


Figure 5-3 Oscillating circuit for fundamental mode

To turn on the oscillator, the oscillating circuit must provide the negative resistance (-Re) at least five times the equivalent series resistance (ESR) of the crystal sample. For larger -Re value, faster turn on the crystal. Higher gm provides larger -Re therefore can start-up the crystal with higher ESR for the same load capacitance (CL). However, it's required higher power consumption.

There are two key parameters to turn on oscillator. Which are CL and the maximum ESR at the target frequency. By reducing the CL, the -Re can be increased thus; shorter turn on time can be achieved. However, if CL is too small, the deviation from the target frequency will increase because of the capacitance variation. So, a trade-off relationship between short turn on time and small frequency deviation in deciding CL value. The smaller ESR of the crystal sample will reduce turn on time but the price is higher. The typical CL and ESR values for difference target frequencies are listed in Table 5-2.

Table 5-2 Typical CL and the corresponding maximum ESR

Target Frequency (Hz)	2M ~ 3M	3M ~ 6M	6M ~ 10M	10M ~ 20M
CL (pf)	25	20	16	12
Maximum ESR (ohm)	1K	400	100	80

Figure 5-3 shows the oscillating circuit is connected with the oscillator I/O cell. Components feedback resistor (Rf), damping resistor (Rd), C1 and C2 are used to adjust the turn on time, keep stability and accurate of the oscillator.

Rf is used to bias the inverter in the high gain region. It cannot be too low or the loop may not oscillate. For mega Hertz range applications, Rf of 1Mohm is applied.

Rd is used to increase stability, low power consumption, suppress the gain in high frequency region and also reduce -Re of the oscillator. Thus, proper Rd cannot be too large to cease the loop oscillating.

C1 and C2 are deciding regard to the crystal or resonator CL specification. In the steady state of oscillating, CL is defined as (C1*C2)/(C1+C2). Actually, the I/O ports, bond pad, and package pin all contribute the parasitic capacitance to C1 and C2. Thus, CL can be rewrite to (C1'*C2')/(C1'+C2'), where C1'=(C1+Cin,stray) and C2'=(C2+Cout,stray). In this case, the required C1 and C2 will be reduced.

Notice, this oscillating circuit is for parallel resonate but not series resonate. Because C1, C2, Rd and Rf are varying with the crystal specifications; therefore there is no single magic number of all the applications.

5.3 Power Manager

In the Low-Power mode, part or whole processor is halted. This will reduce power consumption. The Power Management Controller contains low-power mode control and reset sequence control

5.3.1 Low-Power Modes and Function

The processor supports six low-power modes and function:

- NORMAL mode

  In Normal mode, all peripherals and the basic blocks including power management block, the CPU core, the bus controller, the memory controller, the interrupt controller, DMA, and the external master may operate completely. But, the clock to each peripheral, except the basic blocks, can be stopped selectively by software to reduce the power consumption.

- DOZE mode

  DOZE mode is entered by setting DOZE bit of LCR to 1. In DOZE mode, clock is burst to CPU core and the clock duty is set by DUTY field of LCR. DOZE mode is canceled by reset, interrupt or clearing DOZE bit to 0. Continuous clock is supplied immediately after DOZE mode is canceled. The other Clocks except CCLK run continuously in DOZE mode.

- IDLE mode

  In IDLE mode, the clock to the CPU core is stopped except the bus controller, the memory controller, the interrupt controller, and the power management block. To exit the IDLE mode, the any interrupts should be activated.

- SLEEP mode

  In SLEEP mode, all clocks except RTC clock are disabled. PLL is disabled also. SLEEP mode is canceled by reset or interrupt. When SLEEP mode is canceled, PLL is restarted , the PLL needs clock stabilization time (PLL lock time). This PLL stabilization time is automatically inserted by the internal logic with lock time count register. and all clocks start operating after PLL stability time.

- CLOCK GATE function

  CLOCK GATE function is used to gate specified on-chip module when it is not used. Set specified CLKG0~15 bits in CLKGR will enter specified CLK gate function. CLOCK gate function is canceled by reset or clearing specified CLKGR0~15 to 0.

5.3.2 Register Description

All PMC register 32bit access address is physical address.

Table 5-3 Power/Reset Management Controller Registers Configuration

Name	description	RW	Initial Value	Address	Access Size
LCR	Low Power Control Register	RW	0x000000F8	0x10000004	32
CLKGR	Clock Gate Register	RW	0x00000000	0x10000020	32
SCR	Sleep Control Register	RW	0x00001500	0x10000024	32

5.3.2.1 Low Power Control Register

The Low Power Control Register (LCR) is a 32-bit read/write register that controls low-power mode status. It is initialized to 0x000000F8 by any reset.

LCR **0x10000004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								DUTY					DOZE	LPM	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0

Bits	Name	Description	RW
31:8	Reserved	Writes to these bits have no effect and always read as 0	R
7:3	DUTY	CPU Clock Duty. Control the CPU clock duty in doze mode. When the DUTY field is 0x1F, the clock is always on and when it is zero, the clock is always off. Set the DUTY field to 0 when the CPU will be disabled for an extended amount of time.	RW

		00000 = 0/31 duty-cycle 00001 = 1/31 duty-cycle 00010 = 2/31 duty-cycle … 11111 = 31/31 duty-cycle	
2	DOZE	Doze Mode. Control the doze mode. When doze mode is canceled, this bit is cleared to 0 automatically 0: Doze mode is off 1: Doze mode is on	RW
1:0	LPM	Low Power Mode. Specifies which low-power mode will be entered when SLEEP instruction is executed Bit 1~0: **00**: IDLE mode will be entered when SLEEP instruction is executed **01**: SLEEP mode will be entered when SLEEP instruction is executed **10**: Reserved **11**: Reserved	RW

5.3.2.2 Clock Gate Register

The Clock Gate Register (CLKGR) is a 32-bit read/write register that controls the CLOCK GATE function of peripherals. It is initialized to 0x00000000 by any reset.

CLKGR **0x10000020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																CLKGR															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0	R
15:0	CLKGR	Clock gate Bits. Controls the clock supplies to some peripherals. If set, clock supplies to associated devices are stopped, and registers of the device cannot be accessed also	RW

Bit	Module	Description
15	UART1	
14	UHC	

11	UDC	0: udc_hclk always running, don't stop 1: Only udc enters suspend mode, udc_hclk has been stopped . if the bit is 1 and udc doesn't enters suspend mode, udc_hclk always runs.
10	LCD	
9	CIM	
8	SADC	
7	MSC	
6	AIC	Affects both AC97 bitclk and I2S clock
5	AIC	Affects PCLK supply of AIC
4	SSI	
3	I2C	
2	RTC	The second counter is still counting
1	TCU	
0	UART0	

5.3.2.3 Sleep Control Register (SCR)

The Sleep Control Register is a 32-bit read/write register that specifies some special controls of SLEEP mode. It is initialized to 0x00001500 by reset.

SCR **0x10000024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TPE_ADC	TPE_CDC	Reserved														O1ST								SPENDH	SPENDD	Reserved	O1SE	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	TPE_ADC	Test SADC point .	RW
30	TPE_CDC	Test Codec point	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0	R
15:8	O1ST	12MHz Oscillator Stabilize Time. This filed specifies the 12Mhz oscillator stabilize time by unit of 16 RTCCLK periods (oscillator stable time O1ST × 16 / 32768) cycles. It is initialized to H'15.	RW
7	SPENDH	Force UHC phy to enter suspend mode 0: UHC phy hasn't forced to entered SUSPEND mode 1: UHC phy has forced to entered SUSPEND mode	RW
6	SPENDN	force UDC phy to enter suspend mode 0: UDC phy has forced to entered SUSPEND mode	RW

		1: UDC phy hasn’t forced to entered SUSPEND mode	
5	Reserved	Writes to these bits have no effect and always read as 0	R
4	O1SE	12MHz Oscillator Sleep Mode Enable. This filed controls the state of the 12Mhz oscillator in Sleep mode. 0: 12M oscillator is disabled in Sleep mode 1: 12M oscillator is enabled in Sleep mode	RW
3:0	Reserved	Writes to these bits have no effect and always read as 0	R

5.3.3 Doze Mode

Firstly, software should set the DUTY bits of LCR. Then set DOZE bit of LCR to 1 to enter doze mode. When slot controller of PMC indicates that the CPU clock's time-slot has expired, CPU is halted but its register contents are retained. During doze mode, program can modify clock duty-cycle according to core resource requirement. Clock control is in increments of approximately 3% (1/31).

Doze is exited by software, interrupt, reset or SLEEP instruction.

5.3.4 IDLE Mode

In normal or mode, when LPM bits in LCR are 0 and SLEEP instruction is executed, the processor enters idle mode. CPU is halted but its register contents are retained All critical application must be finished and peripherals must be configured to generate interrupts when they need CPU attention.

The procedure of entering sleep mode is shown blow:

1. Set LPM bits in LCR to 0.
2. Executes SLEEP instruction.
3. When current operation of CPU core has finished and CPU core is idle, CCLK supply to CPU core is stopped.

IDLE mode is exited by an interrupt (IRQ or on-chip devices) or a reset.

5.3.5 SLEEP Mode

In normal mode, when LPM bits in LCR is 1 and SLEEP instruction is executed, the processor enter SLEEP mode. CPU and on-chip devices are halted, except some wakeup-logic. PLL is shut off. Clock output from CKO pin is also stopped. SDRAM content is preserved by driving into self-refresh state. CPU registers and on-chip devices registers contents are retained

Before enter SLEEP mode, software should ensure that all peripherals are not running. The procedure of entering SLEEP mode is shown blow:

1. Set LPM bit in LCR to 1.
2. Execute a SLEEP instruction.
3. When current access on system bus complete, the arbiter will not grant any following request. EMC will drive SDRAM from auto-refresh mode to self-refresh mode.
4. When system bus is idle state and SDRAM is self-refresh mode, internal clock supplies are stopped.

SLEEP mode can be exited by an interrupt (IRQ or on-chip devices), WDT reset or a poweron reset via the RESETP pin.

5.4 Reset Control Module

5.4.1 Register Description

All RCM register 32bit access address is physical address.

Name	description	RW	Initial Value	Address	Access Size
RSR	Reset Status Register	RW	0x????????	0x10000008	32

5.4.1.1 Reset Status Register (RSR)

The Reset Status Register (RSR) is a 32-bit read/write register which records last cause of reset. Each RSR bit is set by a different source of reset. Please refer to Reset Sequence Control for reset sources description.

RSR **0x10000008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																														WR	PR
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?

Bits	Name	Description	RW
31:2	Reserved	Writes to these bits have no effect and always read as 0	R
1	WR	WDT Reset. When a WDT reset is detected, WR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 will be ignored 0: WDT reset has not occurred since the last time the software clears this bit 1: WDT reset has occurred since the last time the software clears this bit	RW
0	PR	Power On Reset. When a poweron reset via PRESET pin is detected, PR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 is ignored 0: Power on reset has not occurred since the last time the software clears this bit 1: Power on reset has occurred since the last time the software clears this bit	RW

5.4.2 Power On Reset

Power on reset is generated when PRESET pin is driven to low. Internal reset is asserted immediately. All pins return to their reset states.

PRESET pin must be held low until power stabilizes and the 12MHz oscillator stabilize. CPU and peripherals are clocked by 12MHz oscillator output directly. PLL is reset to off state. All internal modules are initialized to their predefined reset states.

5.4.3 WDT Reset

WDT reset is generated when WDT overflow. Internal reset is asserted within two RTCCLK cycles. All pins return to their reset states.

Then WDT reset source is cleared because of internal reset. The internal reset is asserted for about 10 milliseconds. CPU and peripherals are clocked by 12MHz oscillator output directly. PLL is reset to off state.

6 Real-Time Clock (RTC)

6.1 Overview

The Real-Time Clock (RTC) unit can be operated in either chip main power is on or the main power is down but the RTC power is still on. In this case, the RTC power domain consumes only a few micro watts power.

The RTC contains a 32768Hz oscillator, the real time and alarm logic, and the power down and wakeup control logic

6.1.1 Features

RTC module has following features:

- Embedded 32768Hz oscillator for 32k clock generation with an external 32k crystal
- 32-bits second counter
- Programmable and adjustable counter to generate accurate 1 Hz clock
- Alarm interrupt, 1Hz interrupt
- Stand alone power supply, work in hibernating mode
- Power down controller
- Alarm wakeup
- External pin wakeup with up to 2s glitch filter

6.1.2 Signal Descriptions

RTC has 5 signal IO pins and 1 power pin. They are listed and described in.

Pin Names	Pin Loc	IO	IO Cell Char.	Pin Description	Power
RTCLK		AI	32768Hz	RTCLK: 32768 clock input or OSC input	VDD$_{RTC}$
RTCLKO		AO		RTCLKO: OSC output	VDD$_{RTC}$
PWRON_		AO	~2mA, Open-Draw	PWRON_: Power on/off control of main power	VDD$_{RTC}$
WKUP_ PD29		AI AI	Schmitt	WKUP_: Wake signal after main power down PD29: GPIO group D bit 29, input/interrupt only	VDD$_{RTC}$
PPRST_		AI	Schmitt	PPRST_: RTC power on reset and RESET-KEY reset input	VDD$_{RTC}$
VDDRTC		P		VDDRTC: 3.3V power for RTC and hibernating mode controlling that never power down	-

- **RTCLK/RTCLKO** pins. We have an embedded oscillator for 32768Hz crystal. These two pins are the crystal XTALI and XTALO connection pins. If an input clock is used instead, please input it to RTCLKO pin.

- **PWRON_** pin: this pin is used to control the main power on/off. Output low voltage means on and high-Z means off.
- **WKUP_** pin: hibernating mode wakeup input
- **PPRST_** pin: This pin should be set to low voltage only in two cases

  (1) When RTC power is turned on (so that whole chip is power on)

  (2) A RESET-KEY is pressed

  Don't set this pin to low voltage when wakeup from hibernating mode. When entering/exiting to/from hibernating mode (in another word, in main power up/down procedure), please avoid putting both WKUP_ and PPRST_ in low voltage. Because the RTC registers, for instance, the second counter and others may be changed.

6.2 Register Description

Table 6-1 Registers for real time clock

Name	Description	RW	Reset Value	Address	Access Size
RTCCR	RTC Control Register	RW	0x00000081[1][2]	0x10003000	32
RTCSR	RTC Second Register	RW	0x????????	0x10003004	32
RTCSAR	RTC Second Alarm Register	RW	0x????????	0x10003008	32
RTCGR	RTC Regulator Register	RW	0x0???????	0x1000300C	32

Note:

1. Unless otherwise stated, the reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.
2. The reset value can be either of 0x00000081, 0x00000091, 0x00000089, 0x00000099

Table 6-2 Registers for hibernating mode

Name	Description	RW	Reset Value	Address	Access Size
HCR	Hibernate Control Register	RW	0x00000000[1]	0x10003020	32
HWFCR	Wakeup filter counter Register in Hibernate mode	RW	0x0000???0	0x10003024	32
HRCR	Hibernate reset counter Register in Hibernate mode	RW	0x00000??0	0x10003028	32
HWCR	Wakeup control Register in Hibernate mode	RW	0x00000000[1]	0x1000302C	32
HWRSR	Wakeup Status Register in Hibernate mode	RW	0x00000000[1]	0x10003030	32
HSPR	Scratch pattern register	RW	0x????????	0x10003034	32

Note: 1. Unless otherwise stated, the reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.

All these registers, include those for real time clock and for hibernating mode control, except otherwise stated, are implemented in 32k clock domain. When write to these registers, it needs about 35 ~ 65 us to actually change the register's value and allow the next write access. A bit RTCCR.WRDY is used to indicate it. When RCR.WRDY is 1, it means the previous write is finished, a right value can be read from the target register, and a new write access can be issued. So before any write access, please make sure RCR.WRDY = 1.

6.2.1 RTC Control Register (RTCCR)

RTCCR contains bits to configure the real time clock features. Unless otherwise stated, the reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.

RTCCR **0x10003000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								WRDY	1HZ	1HZIE	AF	AIE	AE	Reserved	RTCE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1[1]	0[1]	0[1]	?	?	0	0	1

Note: 1. These bits are reset in all resets: PPRST_ input pin reset, hibernating reset and WDT reset.

Bits	Name	Description	RW
31:7	Reserved	Writes to these bits have no effect and always read as 0	R
7	WRDY	Write ready flag. It is 0 when a write is currently processing and the value has not been written to the writing target register. No write to any RTC registers can be issued in this case, or the result is undefined. The read value from the target register is also undefined. The reading is meaningful and another write can be issued when it is 1. Please reference to descriptions in 6.2 for some more details. This bit is read only and write to it is ignored.	R
6	1HZ	1Hz flag. This bit is set by hardware once every 1 second through the 1Hz pulse if the real time clock is enabled (RTCCR.RTCE = 1). This bit can be cleared by software. Write 1 to this bit is ignored. Writing to this bit takes effect immediately without delay.	RW
5	1HZIE	1Hz interrupt enable. Writing to this bit takes effect immediately without delay. **1HZIE** / **Description** 0 / 1Hz interrupt is disabled 1 / 1Hz interrupt is enabled. RTC issues interrupt when 1HZ bit is set	RW
4	AF	Alarm flag. This bit is set by hardware when alarm match (RTCSR = RTCSAR) is found and alarm is enabled (RTCCR.AE = 1) and the real time clock is enabled (RTCCR.RTCE = 1). This bit can be cleared by software. Write 1 to this bit is ignored. Writing to this bit takes effect immediately.	RW
3	AIE	Alarm interrupt enable. **AIE** / **Description** 0 / Alarm interrupt is disabled 1 / Alarm interrupt is enabled. RTC issues interrupt when AF is set	RW

<table>
<tr><td>2</td><td>AE</td><td>Alarm enable.<table><tr><th>AE</th><th>Description</th></tr><tr><td>0</td><td>Alarm function is disabled</td></tr><tr><td>1</td><td>Alarm function is enabled</td></tr></table></td><td>RW</td></tr>
<tr><td>1</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>0</td><td>RTCE</td><td>Real time clock enable.<table><tr><th>RTCE</th><th>Description</th></tr><tr><td>0</td><td>Real time clock function is disabled</td></tr><tr><td>1</td><td>Real time clock function is enabled</td></tr></table></td><td>RW</td></tr>
</table>

6.2.2 RTC Second Register (RTCSR)

RTCSR is a 32-bit width second counter. It can be read and write by software. It is increased by 1 at every 1Hz pulse if the real time clock is enabled (RTCCR.RTCE = 1). When read, it should be read continued more than once and take the value if the adjacent results are the same. RTCSR is not initialized by any reset.

RTCSR **0x10003004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RTCSR																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

6.2.3 RTC Second Alarm Register (RTCSAR)

RTCSAR serves as a second alarm register. Alarm flag (RTCCR.AF) is set to 1 when the RTCSR equals the RTCSAR in the condition of alarm is enabled (RTCCR.AE = 1) and the real time clock is enabled (RTCCR.RTCE = 1). RTCSAR can be read and write by software and is not initialized by any reset.

RTCSAR **0x10003008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RTCSAR																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

6.2.4 RTC Regulator Register (RTCGR)

RTCGR is serves as the real time clock regulator, which is used to adjust the interval of the 1Hz pulse.

RTCGR **0x1000300C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LOCK	Reserved					ADJC										NC1HZ															
RST	0[1]	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Note: 1. This bit is reset in all resets: PPRST_ input pin reset, hibernating reset and WDT reset.

Bits	Name	Description	RW
31	LOCK	Lock bit. This bit is used to safeguard the validity of the data written into the RTCGR register. Once it is set, write to RTCGR is ignored. This bit can only be set by software and cleared by (any type of) resets. **LOCK** / **Description** 0 / Write to RTCGR is allowed 1 / Write to RTCGR is forbidden	RW
30:26	Reserved	Writes to these bits have no effect and always read as 0	R
25:16	ADJC	This field specifies how many times it needs to add one 32kHz cycle for the 1Hz pulse interval in every 1024 1Hz pulses. In other word, among every 1024 1Hz pulses, ADJC number of them are trigged in every (NC1HZ + 2) 32kHz clock cycles, (1024 – ADJC) number of them are trigged in every (NC1HZ + 1) 32kHz clock cycles.	RW
15:0	NC1HZ	This field specifies the number plus 1 of the working 32kHz clock cycles are contained in the 1Hz pulse interval. In other word, 1Hz pulse is trigged every (NC1HZ + 1) 32kHz clock cycles, if RTCGR.ADJC = 0	RW

6.2.5 Hibernate Control Register (HCR)

HCR contains the bit to control the main chip power on/off. This register is reset by any reset.

HCR **0x10003020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																															PD
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:1</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>0</td><td>PD</td><td>Power down or power on bit. Besides writing by CPU, this bit will be set to 1 if an unknown reason main power supply off is detected. This bit controls the PWRON_ pin level. When co-working with some external components, this bit is used for power management of this chip. It is supposed when 1 is written to this bit, the main power supply of the chip, except RTC power, will be shut down immediately. After this bit is set to 1, all registers in RTC module, except RTCCR.1HZ and RTCCR.1HZIE, cannot be changed by write access. This bit is cleared by reset pin reset and hibernating reset. The later one is asserted by wakeup procedure.
<table>
<tr><th>PD</th><th>PWRON_</th><th>Description</th></tr>
<tr><td>0</td><td>0 V</td><td>No power down, keep power on</td></tr>
<tr><td>1</td><td>VDDRTC</td><td>Power down enable, turn power off</td></tr>
</table>
</td><td>RW</td></tr>
</table>

6.2.6 HIBERNATE mode Wakeup Filter Counter Register (HWFCR)

The HIBERNATE mode Wakeup Filter Counter Register (HWFCR) is a 32-bit read/write register .It filter the glitch generated by a dedicated wakeup pin. The HRCR is initialized by PPRST_ and WDT reset.

HWFCR **0x10003024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																HWFCR											Reserved				
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0	R
15:5	HWFCR	Wakeup pin effective minimum time in number of 32 RTCLK cycles, used as glitch filter logic. Maximum of 2 seconds	RW
4:0	Reserved	Writes to these bits have no effect and always read as 0	R

6.2.7 Hibernate Reset Counter Register (HRCR)

The Hibernate Reset Counter Register is a 32-bit read/write register that specifies hibernate reset assertion time. The HRCR is initialized by PPRST_ and WDT reset

HRCR **0x10003028**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																				HRCR							Reserved				
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved	Writes to these bits have no effect and always read as 0	R
11:5	HRCR	HIBERNATE Reset waiting time. Number of 32 RTCLK cycles. Maximum 125 ms	RW
4:0	Reserved	Writes to these bits have no effect and always read as 0	R

6.2.8 HIBERNATE Wakeup Control Register (HWCR)

The HIBERNATE Wakeup Control Register is a 32-bit read/write register that controls real time clock alarm wake up enable. The reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.

HWCR **0x1000302C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																															EALM
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:1	Reserved	Writes to these bits have no effect and always read as 0	R
0	EALM	RTC Alarm wakeup enable 0: disable 1: enable	RW

6.2.9 HIBERNATE Wakeup Status Register (HWRSR)

The HIBERNATE Wakeup Status Register is a 32-bit read/write register that reflects wakeup status bits.

HWRSR **0x10003030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										HR	PPR	Reserved		PIN	ALM
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1[2]	1[1]	0	0	?	?

Note:

1. This reset value only for PPRST_. It is undefined in case of other resets.
2. This reset value only for HRST_. It is undefined in case of other resets.

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:6</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>5</td><td>HR</td><td>Hibernate Reset. When a Hibernate reset detected, HR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 is ignored
<table>
<tr><th>HR</th><th>Description</th></tr>
<tr><td>0</td><td>Hibernate reset has not occurred since the last time the software clears this bit</td></tr>
<tr><td>1</td><td>Hibernate reset has occurred since the last time the software clears this bit</td></tr>
</table></td><td>RW</td></tr>
<tr><td>4</td><td>PPR</td><td>PAD PIN Reset. When a PPRST_ is detected, PPR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 is ignored
<table>
<tr><th>PPR</th><th>Description</th></tr>
<tr><td>0</td><td>PPRST_ reset has not occurred since last time the software clears this bit</td></tr>
<tr><td>1</td><td>PPRST_ reset has occurred since last time the software clears this bit</td></tr>
</table></td><td>RW</td></tr>
<tr><td>3:2</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>1</td><td>PIN</td><td>Wakeup Pin Status bit. The bit is cleared when chip enters hibernating mode. It is set when exit the hibernating mode by wakeup pin. This bit can only be written with 0. Write with 1 is ignored.</td><td>RW</td></tr>
<tr><td>0</td><td>ALM</td><td>RTC Alarm Status bit. The bit is cleared when chip enters hibernating mode. It is set when exit the hibernating mode by alarm. This bit can only be written with 0. Write with 1 is ignored.</td><td>RW</td></tr>
</table>

6.2.10 Hibernate Scratch Pattern Register (HSPR)

This is a scratch register used to hold a pattern. The software can check the pattern is kept to know whether RTC power has ever been down and whether it is needed to setup the real time clock.

HSPR **0x10003034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PAT																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	**Name**	**Description**	**RW**
31:0	PAT	The pattern	RW

6.3 Time Regulation

Because of the inherent inaccuracy of crystal and other variables, the time counter may be inaccurate. This requires a slight adjustment. The application processor, through the RTCGR, lets you adjust the 1Hz time base to an error of less than 1ppm. Such that if the Hz clock were set to be 1Hz, there would be an error of less than 5 seconds per month.

To determine the value programmed into the RTCGR, you must first measure the output frequency at the oscillator multiplex (approximately 32 kHz) using an accurate time base, such as a frequency counter. This clock is externally visible by selecting the alternate function of GPIO[?]
To gain access to the clock, program this pin as an output and then switch to the alternate function. To trim the clock, divide the output of the oscillator by an integer value and fractional adjust it by periodically deleting clocks from the stream driving this integer divider.

After the true frequency of the oscillator is known, it must be split into integer and fractional portions. The integer portion of the value (minus one) is loaded into the DIV field of the RTCGR.

The fractional part of the adjustment is done by periodically deleting clocks from the clock stream driving the Hz divider. The trim interval period is hardwired to be 1024 1Hz clock cycles (approximately 17 minutes). The number of clocks (represented by ADC field of RTCGR) are deleted from the input clock stream per trim interval. If ADC is programmed to be zero, then no trim operations occur and the RTC is clocked with the raw 32 kHz clock. The relationship between the Hz clock frequency and the nominal 32 kHz clock (f1 and f32K, respectively) is shown in the following equation.

$$\mathrm{f1} = \frac{2\text{^}10 \times (\mathrm{DIV} + 1)}{2\text{^}10 \times (\mathrm{DIV} + 1) + \mathrm{ADC}} \times \frac{\mathrm{f32k}}{\mathrm{DIV} + 1}$$

f1 = actual frequency of 1Hz clock
f32k = frequency of either 32.768KHz crystal output or 3.6864MHz crystal output further divided down to 32.914KHz

6.3.1 HIBERNATE Mode

When Software writes 1 to PD bit of HCR, the system at once enters HIBERNATE mode. The powers of CORE and IO are disconnected by PWRON_ pin, no power consumption to core and IO. When a wakeup event occurs, the core enters through a hibernate reset. Only CPM wake up logic and RTC is operating in HIBERNATE mode.

6.3.1.1 Procedure to Enter HIBERNATE mode

Before enter HIBERNATE mode, software must complete following steps:

- Finish the current operation and preserve all data to flash
- Configure the wake-up sources properly by configure HWCSR
- Set HIBERNATE MODE (Set PD bit in HCR to 1.)

6.3.1.2 Procedure to Wake-up from HIBERNATE mode

- The internal hibernate reset signal will be asserted if one of the wake-up sources is issued.
- Check RSR to determine what caused the reset
- Check PIN/ALM bits of HWCSR in order to know whether or not the power-up is caused by which wake-up from HIBERNATE mode.
- Configure the SDRAM memory controller.
- Recover the data from flash

7 Interrupt Controller

7.1 Overview

This chapter describes the interrupt controller included in the JZ4XX Processor, explains its modes of operation, and defines its registers. The interrupt controller controls the interrupt sources available to the processor and contains the location of the interrupt source to allow software to determine source of all interrupts. It also determines whether the interrupts cause an IRQ to occur and masks the interrupts.

Features:

- Total 32 interrupt sources
- Each interrupt source can be independently enabled
- Priority mechanism to indicate highest priority interrupt
- All the registers are accessed by CPU.
- Unmasked interrupts can wake up the chip in sleep mode.

7.2 Register Description

Table 7-1 INTC Register lists the registers of Interrupt Controller. All of these registers are 32bit, and each bit of the register represents or controls one interrupt source that list in Table 7-1 INTC Register.

All INTC register 32bit access address is physical address.

Name	Description	RW	Reset Value	Address	Access Size
ICSR	Interrupt controller Source Register	R	0x00000000	0x10001000	32
ICMR	Interrupt controller Mask Register	RW	0xFFFFFFFF	0x10001004	32
ICMSR	Interrupt controller Mask Set Register	W	0x????????	0x10001008	32
ICMCR	Interrupt controller Mask Clear Register	W	0x????????	0x1000100C	32
ICPR	Interrupt controller Pending Register	R	0x00000000	0x10001010	32

Table 7-1 INTC Register

7.2.1 Interrupt Controller Source Register (ICSR)

This register contains all the interrupts' status. A "1" indicates that the corresponding interrupt is pending . A "0" indicates that the interrupt is not pending now. The register is read only.

ICSR **0x10001000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	LCD	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICSR	Description
0	The corresponding interrupt source is not pending
1	The corresponding interrupt source is pending

7.2.2 Interrupt Controller Mask Register (ICMR)

This register is used to mask the interrupt input sources and defines which active sources are allowed to generate interrupt requests to the processor. Its value can be changed either by writing ICMSR and ICMCR or by writing itself. The masked interrupts are invisible to the processor.

ICMR **0x10001004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	LCD	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits Of ICMR	Description
0	The corresponding interrupt is not masked.
1	The corresponding interrupt is masked

7.2.3 Interrupt Controller Mask Set Register (ICMSR)

This register is used to set bits in the interrupt mask register. This register is write only

ICMSR **0x10001008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	LCD	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits Of ICMSR	Description
0	ignore
1	Will set the corresponding interrupt mask bit

7.2.4 Interrupt Controller Mask Clear Register (ICMCR)

This register is used to clear bits in the interrupt mask register. This register is write only.

ICMCR **0x1000100C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	LCD	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits Of ICMCR	Description
0	ignore
1	Will clear the corresponding interrupt mask bit

7.2.5 Interrupt Controller Pending Register (ICPR)

This register contains the status of the interrupt sources after masking. This register is read only.

ICPR **0x10001010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	LCD	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICPR	Description
0	The corresponding interrupt is not active or is masked
1	The corresponding interrupt is active and is not masked to the processor.

Note: Reserved bits in ICMR, ICMSR and ICMCR are normal bits to be written into and read out. Reserved bits in ICSR and ICPR are read-only and always 0.

7.3 Software Considerations

The interrupt controller is reflecting the status of interrupts sources in the peripheral .

Software should perform the task - determine the interrupt source from in ICPR. In this chip, pending interrupts have two levels in structure. Interrupting module in the system that contains more than one interrupt sources need software to determine how to service it by reading interrupt status registers within it.

In the interrupt handler, the serviced interrupt source needs to be cleared in the interrupting device. In order to make certain the cleared source request status has been reflected at the corresponding ICPR bit, software should wait enough time before exiting interrupt state.

The procedure is described following:

1. Interrupt generated.
2. CPU query interrupt sources, saves the current environment and then goes to interrupt common service routine.
3. Get ICPR.
4. Find the highest priority interrupt and vector it. (The software decides which one has the highest priority).
5. Mask the chosen interrupt by writing the register ICMSR.
6. Enable the system interrupt to allow the interrupt nesting.(software decided)
7. Execute the interrupt handler and unmask it by writing the register ICMCR when exit the handler.
8. CPU restores the saved environment and exit the interrupt state.

8 Timer/Counter Unit

8.1 Overview

The TCU (Timer/Counter with PWM output) contains 8 channels of 16-bit programmable timers (timers 0 to 7). They can be used as Timer or PWM.

TCU has the following features:

- Six independent channels, each consisting of
  - — Counter
  - — Data register (FULL and HALF)
  - — Control register
- Independent clock for each counter, selectable by software
  - — PCLK, EXTAL and RTCCLK can be used as the clock for counter
  - — The division ratio of the clock can be set to 1, 4, 16, 64, 256 and 1024 by software
- FULL interrupt and HALF interrupt can be generated for each channel using the compare data registers
  - — Timer 0 and Timer 1 have separated interrupt.
  - — Timer 2-7 has one interrupt in common.
  - — Timer 0-7 can be used as PWM (Set the initial signal level)

8.2 Pin Description

Table 8-1 PWM Pins Description

Name	I/O	Description
PWM [7:0]	Output	PWM channel output signals.

8.3 Register Description

In this section, we will describe the registers in timer. Following table lists all the registers definition. All timer register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
TSR	Timer STOP Register	R	0x00	0x1000201C	8
TSSR	Timer STOP Set Register	W	0x00	0x1000202C	8
TSCR	Timer STOP Clear Register	W	0x00	0x1000203C	8
TER	Timer Counter Enable Register	R	0x00	0x10002010	8
TESR	Timer Counter Enable Set Register	W	0x??	0x10002014	8
TECR	Timer Counter Enable Clear Register	W	0x??	0x10002018	8
TFR	Timer Flag Register	R	0x00000000	0x10002020	32
TFSR	Timer Flag Set Register	W	0x????????	0x10002024	32
TFCR	Timer Flag Clear Register	W	0x????????	0x10002028	32
TMR	Timer Mask Register	R	0x00000000	0x10002030	32
TMSR	Timer Mask Set Register	W	0x????????	0x10002034	32
TMCR	Timer Mask Clear Register	W	0x????????	0x10002038	32
TDFR0	Timer Data FULL Register 0	RW	0x????	0x10002040	16
TDHR0	Timer Data HALF Register 0	RW	0x????	0x10002044	16
TCNT0	Timer Counter 0	RW	0x????	0x10002048	16
TCSR0	Timer Control Register 0	RW	0x0000	0x1000204C	16
TDFR1	Timer Data FULL Register 1	RW	0x????	0x10002050	16
TDHR1	Timer Data HALF Register 1	RW	0x????	0x10002054	16
TCNT1	Timer Counter 1	RW	0x????	0x10002058	16
TCSR1	Timer Control Register 1	RW	0x0000	0x1000205C	16
TDFR2	Timer Data FULL Register 2	RW	0x????	0x10002060	16
TDHR2	Timer Data HALF Register 2	RW	0x????	0x10002064	16
TCNT2	Timer Counter 2	RW	0x????	0x10002068	16
TCSR2	Timer Control Register 2	RW	0x0000	0x1000206C	16
TDFR3	Timer Data FULL Register 3	RW	0x????	0x10002070	16
TDHR3	Timer Data HALF Register 3	RW	0x????	0x10002074	16
TCNT3	Timer Counter 3	RW	0x????	0x10002078	16
TCSR3	Timer Control Register 3	RW	0x0000	0x1000207C	16
TDFR4	Timer Data FULL Register 4	RW	0x????	0x10002080	16
TDHR4	Timer Data HALF Register 4	RW	0x????	0x10002084	16
TCNT4	Timer Counter 4	RW	0x????	0x10002088	16

TCSR4	Timer Control Register 4	RW	0x0000	0x1000208C	16
TDFR5	Timer Data FULL Register 5	RW	0x????	0x10002090	16
TDHR5	Timer Data HALF Register 5	RW	0x????	0x10002094	16
TCNT5	Timer Counter 5	RW	0x????	0x10002098	16
TCSR5	Timer Control Register 5	RW	0x0000	0x1000209C	16
TDFR6	Timer Data FULL Register 6	RW	0x????	0x100020A0	16
TDHR6	Timer Data HALF Register 6	RW	0x????	0x100020A4	16
TCNT6	Timer Counter 6	RW	0x????	0x100020A8	16
TCSR6	Timer Control Register 6	RW	0x0000	0x100020AC	16
TDFR7	Timer Data FULL Register 7	RW	0x????	0x100020B0	16
TDHR7	Timer Data HALF Register 7	RW	0x????	0x100020B4	16
TCNT7	Timer Counter 7	RW	0x????	0x100020B8	16
TCSR7	Timer Control Register 7	RW	0x0000	0x100020BC	16

8.3.1 Timer Control Register (TCSR)

The TCSR is a 16-bit read/write register. It contains the control bits for each channel. It is initialized to 0x00 by any reset.

TCSR0, TCSR1, TCSR2, TCSR3, TCSR4, TCSR5, TCSR6, TCSR7 — **0x1000204C, 0x1000205C, 0x1000206C, 0x1000207C, 0x1000208C, 0x1000209C, 0x100020AC, 0x100020BC**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved						SD	INITL	PWM_EN	Reserved	PRESCALE			EXT_EN	RTC_EN	PCK_EN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:10	Reserved	These bits always read 0, and written are ignored.	R
9	SD	Shut Down (SD) the PWM output. 0: Graceful shutdown 1: Abrupt shutdown Graceful shutdown: The output level for PWM output will keep the level after the comparison match of FULL. Abrupt shutdown: The output level for PWM output will keep the level.	RW
8	INITL	Selects an initial output level for PWM output. 1: High 0: Low	RW
7	PWM_EN	PWM output pin control bit 1: PWM pin output enable 0: PWM pin output disable, and the PWM pin will be set to the initial level according to INITL.	RW
6	Reserved	These bits always read 0, and written are ignored.	R
5:3	PRESCALE	These bits select the TCNT count clock frequency. Don't change this field when the channel is running	RW

Bit 2	Bit1	Bit 0	Description
0	0	0	Internal clock: CLK/1
0	0	1	Internal clock: CLK/4
0	1	0	Internal clock: CLK/16

<table>
<tr><td></td><td></td><td><table><tr><td>0</td><td>1</td><td>1</td><td>Internal clock: CLK/64</td></tr><tr><td>1</td><td>0</td><td>0</td><td>Internal clock: CLK/256</td></tr><tr><td>1</td><td>0</td><td>1</td><td>Internal clock: CLK/1024</td></tr><tr><td colspan="3">110~111</td><td>Reserved</td></tr></table></td><td></td></tr>
<tr><td>2</td><td>EXT_EN</td><td>Select EXTAL as the timer clock input.
1: Enable
0: Disable</td><td>RW</td></tr>
<tr><td>1</td><td>RTC_EN</td><td>Select RTCCLK as the timer clock input.
1: Enable
0: Disable</td><td>RW</td></tr>
<tr><td>0</td><td>PCK_EN</td><td>Select PCLK as the timer clock input.
1: Enable
0: Disable</td><td>RW</td></tr>
</table>

Note:

The input clock of timer and the PCLK should keep to the rules as follows:

Input clock of timer: IN_CLK	**Clock generated from the frequency divider (PRESCALE): DIV_CLK**
PCK_EN == 0, RTC_EN == 1 and EXT_EN == 0 (IN_CLK = RTCCLK)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 0, RTC_EN == 0 and EXT_EN == 1 (IN_CLK = EXTAL)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 1, RTC_EN == 0 and EXT_EN == 0 (IN_CLK = PCLK)	ANY

8.3.2 Timer Data FULL Register (TDFR)

The comparison data FULL registers TDFR is used to store the data to be compared with the content of the up-counter TCNT. This register can be directly read and written. (Default: indeterminate)

TDFR0, TDFR1, TDFR2,	**0x10002040, 0x10002050, 0x10002060,**
TDFR3, TDFR4, TDFR5,	**0x10002070, 0x10002080, 0x10002090,**
TDFR6, TDFR7	**0x100020A0, 0x100020B0**

Bit		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		TDFR															
RST		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

8.3.3 Timer Data HALF Register (TDHR)

The comparison data HALF registers TDHR is used to store the data to be compared with the content of the up-counter TCNT. This register can be directly read and written. (Default: indeterminate)

TDHR0, TDHR1, TDHR2, **0x10002044, 0x10002054, 0x10002064,**
TDHR3, TDHR4, TDHR5, **0x10002074, 0x10002084, 0x10002094,**
TDHR6, TDHR7 **0x100020A4, 0x100020B4**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDHR															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

8.3.4 Timer Counter (TCNT)

TCNT is a 16-bit read/write register. The up-counter TCNT can be reset to 0 by software and counts up using the prescaler output clock. When TCNT count up to equal to TDFR, it will reset to 0 and continue to count up. The data can be read out at any time. The counter data can be written at any time. This makes it possible to change the interrupt and/or clock output cycles temporarily. (Default: indeterminate)

TCNT0, TCNT1, TCNT2, **0x10002048, 0x10002058, 0x10002068,**
TCNT3, TCNT4, TCNT5, **0x10002078, 0x10002088, 0x10002098,**
TCNT6, TCNT7 **0x100020A8, 0x100020B8**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCNT															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

8.3.5 Timer Counter Enable Register (TER)

The TER is an 8-bit read-only register. It contains the counter enable control bits for each channel. It is initialized to 0x00 by any reset. It can only be set by register TESR and TECR. Since the timer enable control bits are located in the same addresses, two or more timers can be started at the same time.

TER **0x10002010**

Bit	7	6	5	4	3	2	1	0

		TCEN 7	TCEN 6	TCEN 5	TCEN 4	TCEN 3	TCEN 2	TCEN 1	TCEN 0
RST		0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	TCEN 7	Enable the counter in timer 7. 1: Begin counting up 0: Stop counting up	R
6	TCEN 6	Enable the counter in timer 6. 1: Begin counting up 0: Stop counting up	R
5	TCEN 5	Enable the counter in timer 5. 1: Begin counting up 0: Stop counting up	R
4	TCEN 4	Enable the counter in timer 4. 1: Begin counting up 0: Stop counting up	R
3	TCEN 3	Enable the counter in timer 3. 1: Begin counting up 0: Stop counting up	R
2	TCEN 2	Enable the counter in timer 2. 1: Begin counting up 0: Stop counting up	R
1	TCEN 1	Enable the counter in timer 1. 1: Begin counting up 0: Stop counting up	R
0	TCEN 0	Enable the counter in timer 0. 1: Begin counting up 0: Stop counting up	R

8.3.6 Timer Counter Enable Set Register (TESR)

The TCCSR is an 8-bit write-only register. It contains the counter enable set bits for each channel. Since the timer enable control set bits are located in the same addresses, two or more timers can be started at the same time.

TESR **0x10002014**

Bit		7	6	5	4	3	2	1	0

		TCST 7	TCST 6	TCST 5	TCST 4	TCST 3	TCST 2	TCST 1	TCST 0
RST		?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7	TCST 7	Set TCEN 7 bit of TER. 1: Set TCEN 7 bit to 1 0: Ignore	W
6	TCST 6	Set TCEN 6 bit of TER. 1: Set TCEN 6 bit to 1 0: Ignore	W
5	TCST 5	Set TCEN 5 bit of TER. 1: Set TCEN 5 bit to 1 0: Ignore	W
4	TCST 4	Set TCEN 4 bit of TER. 1: Set TCEN 4 bit to 1 0: Ignore	W
3	TCST 3	Set TCEN 3 bit of TER. 1: Set TCEN 3 bit to 1 0: Ignore	W
2	TCST 2	Set TCEN 2 bit of TER. 1: Set TCEN 2 bit to 1 0: Ignore	W
1	TCST 1	Set TCEN 1 bit of TER. 1: Set TCEN 1 bit to 1 0: Ignore	W
0	TCST 0	Set TCEN 0 bit of TER. 1: Set TCEN 0 bit to 1 0: Ignore	W

8.3.7 Timer Counter Enable Clear Register (TECR)

The TECR is an 8-bit write-only register. It contains the counter enable clear bits for each channel. Since the timer enable clear bits are located in the same addresses, two or more timers can be stop at the same time.

TECR **0x10002018**

Bit		7	6	5	4	3	2	1	0

	TCCL 7	TCCL 6	TCCL 5	TCCL 4	TCCL 3	TCCL 2	TCCL 1	TCCL 0
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7	TCCL 7	Set TCEN 7 bit of TER. 1: Set TCEN 7 bit to 0 0: Ignore	W
6	TCCL 6	Set TCEN 6 bit of TER. 1: Set TCEN 6 bit to 0 0: Ignore	W
5	TCCL 5	Set TCEN 5 bit of TER. 1: Set TCEN 5 bit to 0 0: Ignore	W
4	TCCL 4	Set TCEN 4 bit of TER. 1: Set TCEN 4 bit to 0 0: Ignore	W
3	TCCL 3	Set TCEN 3 bit of TER. 1: Set TCEN 3 bit to 0 0: Ignore	W
2	TCCL 2	Set TCEN 2 bit of TER. 1: Set TCEN 2 bit to 0 0: Ignore	W
1	TCCL 1	Set TCEN 1 bit of TER. 1: Set TCEN 1 bit to 0 0: Ignore	W
0	TCCL 0	Set TCEN 0 bit of TER. 1: Set TCEN 0 bit to 0 0: Ignore	W

8.3.8 Timer Flag Register (TFR)

The TFR is a 32-bit read-only register. It contains the comparison match flag bits for all the channels. It can also be set by register TFSR and TFCR. It is initialized to 0x00000000 by any reset.

TFR **0x10002020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HFLAG 7	HFLAG 6	HFLAG 5	HFLAG 4	HFLAG 3	HFLAG 2	HFLAG 1	HFLAG 0	Reserved								FFLAG 7	FFLAG 6	FFLAG 5	FFLAG 4	FFLAG 3	FFLAG 2	FFLAG 1	FFLAG 0

RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	These bits always read 0, and written are ignored.	R
23:16	HFLAG 7~0	HALF comparison match flag. (TCNT = TDHR) 1: Comparison match 0: Comparison not match	R
15:8	Reserved	These bits always read 0, and written are ignored.	R
7:0	FFLAG 7~0	FULL comparison match flag. (TCNT = TDFR) 1: Comparison match 0: Comparison not match	R

8.3.9 Timer Flag Set Register (TFSR)

The TFSR is a 32-bit write-only register. It contains the comparison match flag set bits for all the channels.

TFSR **0x10002024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HFST 7	HFST 6	HFST 5	HFST 4	HFST 3	HFST 2	HFST 1	HFST 0	Reserved								FFST 7	FFST 6	FFST 5	FFST 4	FFST 3	FFST 2	FFST 1	FFST 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:22	Reserved	-	-
23:16	HFST 7~0	Set HFLAG n bit of TFR. 1: Set HFLAG n bit to 1 0: Ignore	W
15:8	Reserved	-	-
7:0	FFST 7~0	Set FFLAG n bit of TFR. 1: Set FFLAG n bit to 1 0: Ignore	W

8.3.10 Timer Flag Clear Register (TFCR)

The TFCR is a 32-bit write-only register. It contains the comparison match flag clear bits for all the channels.

TFCR **0x10002028**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HFCL 7	HFCL 6	HFCL 5	HFCL 4	HFCL 3	HFCL 2	HFCL 1	HFCL 0	Reserved								FFCL 7	FFCL 6	FFCL 5	FFCL 4	FFCL 3	FFCL 2	FFCL 1	FFCL 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:24	Reserved	-	-
23:16	HFCL 7~0	Set HFLAG n bit of TFR. 1: Set FFLAG n bit to 0 0: Ignore	W
15:8	Reserved	-	-
7:0	FFCL 7~0	Set FFLAG n bit of TFR. 1: Set FFLAG n bit to 0 0: Ignore	W

8.3.11 Timer Mast Register (TMR)

The TMR is a 32-bit read-only register. It contains the comparison match flag bits for all the channels. It is initialized to 0x00000000 by any reset. It can only be set by register TMSR and TMCR.

TMR **0x10002030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HMASK 7	HMASK 6	HMASK 5	HMASK 4	HMASK 3	HMASK 2	HMASK 1	HMASK 0	Reserved								FMASK 7	FMASK 6	FMASK 5	FMASK 4	FMASK 3	FMASK 2	FMASK 1	FMASK 0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:24	Reserved	These bits always read 0, and written are ignored.	R
23:16	HMASK 7~0	HALF comparison match interrupt mask. 1: Comparison match interrupt mask 0: Comparison match interrupt not mask	R
15:8	Reserved	These bits always read 0, and written are ignored.	R
7:0	FMASK 7~0	FULL comparison match interrupt mask.	R

		1: Comparison match interrupt mask 0: Comparison match interrupt not mask	

8.3.12 Timer Mask Set Register (TMSR)

The TMSR is a 32-bit write-only register. It contains the comparison match flag set bits for all the channels.

TMSR **0x10002034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HFST 7	HFST 6	HFST 5	HFST 4	HFST 3	HFST 2	HFST 1	HFST 0	Reserved								FFST 7	FFST 6	FFST 5	FFST 4	FFST 3	FFST 2	FFST 1	FFST 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:24	Reserved	-	-
23:16	HMST 7~0	Set HMASK n bit of TMR. 1: Set HMASK n bit to 1 0: Ignore	W
15:8	Reserved	-	-
7:0	FMST 7~0	Set FMASK n bit of TMR. 1: Set FMASK n bit to 1 0: Ignore	W

8.3.13 Timer Mask Clear Register (TMCR)

The TMCR is a 32-bit write-only register. It contains the comparison match flag clear bits for all the channels.

TMCR **0x10002038**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								HFCL 7	HFCL 6	HFCL 5	HFCL 4	HFCL 3	HFCL 2	HFCL 1	HFCL 0	Reserved								FFCL 7	FFCL 6	FFCL 5	FFCL 4	FFCL 3	FFCL 2	FFCL 1	FFCL 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:24	Reserved	-	-

25:16	HMCL 7~0	Set HMASK n bit of TMR. 1: Set HMASK n bit to 0 0: Ignore	W
15:8	Reserved	-	-
7:0	FMCL 7~0	Set FMASK n bit of TMR. 1: Set FMASK n bit to 0 0: Ignore	W

8.3.14 Timer Stop Register (TSR)

The TSR is a 32-bit read-only register. It contains the timer stop control bits for each channel and WDT timer. It is initialized to 0x00000000 by any reset. It can only be set by register TSSR and TSCR. If set, clock supplies to timer n / WDT timer is stopped, and registers of the timer/WDT cannot be accessed also.

TSR **0x1000201C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															WDTS	Reserved								STOP 7	STOP 6	STOP 5	STOP 4	STOP 3	STOP 2	STOP 1	STOP 0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:17	Reserved	These bits always read 0, and written are ignored.	R
16	WDTS	1: The clock supplies to WDT is stopped. 0: The clock supplies to WDT is supplied.	R
15:8	Reserved	These bits always read 0, and written are ignored.	R
7	STOP 7	1: The clock supplies to timer 7 is stopped. 0: The clock supplies to timer 7 is supplied.	R
6	STOP 6	1: The clock supplies to timer 6 is stopped. 0: The clock supplies to timer 6 is supplied.	R
5	STOP 5	1: The clock supplies to timer 5 is stopped. 0: The clock supplies to timer 5 is supplied.	R
4	STOP 4	1: The clock supplies to timer 4 is stopped. 0: The clock supplies to timer 4 is supplied.	R
3	STOP 3	1: The clock supplies to timer 3 is stopped.	R

		0: The clock supplies to timer 3 is supplied.	
2	STOP 2	1: The clock supplies to timer 2 is stopped. 0: The clock supplies to timer 2 is supplied.	R
1	STOP 1	1: The clock supplies to timer 1 is stopped. 0: The clock supplies to timer 1 is supplied.	R
0	STOP 0	1: The clock supplies to timer 0 is stopped. 0: The clock supplies to timer 0 is supplied.	R

8.3.15 Timer Stop Set Register (TSSR)

The TCSR is an 8-bit write-only register. It contains the timer stop set bits for each channel and WDT timer. Since the timer stop control set bits are located in the same addresses, two or more timers can be started at the same time.

TSSR **0x1000202C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															WDTSS	Reserved								STPS 7	STPS 6	STPS 5	STPS 4	STPS 3	STPS 2	STPS 1	STPS 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:17	Reserved	-	-
16	WDTSS	Set WDTS bit of TSR. 1: Set WDTS bit to 1 0: Ignore	W
15:8	Reserved	-	-
7	STPS 7	Set STOP 7 bit of TSR. 1: Set STOP 7 bit to 1 0: Ignore	W
6	STPS 6	Set STOP 6 bit of TSR. 1: Set STOP 6 bit to 1 0: Ignore	W
5	STPS 5	Set STOP 5 bit of TSR. 1: Set STOP 5 bit to 1	W

		0: Ignore	
4	STPS 4	Set STOP 4 bit of TSR. 1: Set STOP 4 bit to 1 0: Ignore	W
3	STPS 3	Set STOP 3 bit of TSR. 1: Set STOP 3 bit to 1 0: Ignore	W
2	STPS 2	Set STOP 2 bit of TSR. 1: Set STOP 2 bit to 1 0: Ignore	W
1	STPS 1	Set STOP 1 bit of SR. 1: Set STOP 1 bit to 1 0: Ignore	W
0	STPS 0	Set STOP 0 bit of TSR. 1: Set STOP 0 bit to 1 0: Ignore	W

8.3.16 Timer Stop Clear Register (TSCR)

The TSCR is an 8-bit write-only register. It contains the timer stop clear bits for each channel and WDT timer. Since the timer stop clear bits are located in the same addresses, two or more timers can be stop at the same time.

TSCR **0x1000203C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															WDTSC	Reserved								STPC 7	STPC 6	STPC 5	STPC 4	STPC 3	STPC 2	STPC 1	STPC 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:17	Reserved	-	-
16	WDTSC	Set WDTS bit of TSR. 1: Set WDTS bit to 0 0: Ignore	W
15:8	Reserved	-	-

7	STPC 7	Set STOP 7 bit of TSR. 1: Set STOP 7 bit to 0 0: Ignore	W
6	STPC 6	Set STOP 6 bit of TSR. 1: Set STOP 6 bit to 0 0: Ignore	W
5	STPC 5	Set STOP 5 bit of TSR. 1: Set STOP 5 bit to 0 0: Ignore	W
4	STPC 4	Set STOP 4 bit of TSR. 1: Set STOP 4 bit to 0 0: Ignore	W
3	STPC 3	Set STOP 3 bit of TSR. 1: Set STOP 3 bit to 0 0: Ignore	W
2	STPC 2	Set STOP 2 bit of TSR. 1: Set STOP 2 bit to 0 0: Ignore	W
1	STPC 1	Set STOP 1 bit of TSR. 1: Set STOP 1 bit to 0 0: Ignore	W
0	STPC 0	Set STOP 0 bit of TSR. 1: Set STOP 0 bit to 0 0: Ignore	W

8.4 Operation

8.4.1 Basic Operation

The value of TDFR should be bigger than TDHR, and the minimum settings are TDHR = 0 and TDFR = 1. In this case, the timer output clock cycle is the input clock × 1/2. If TDHR > TDFR, no comparison TFHR signal is generated.

Before the timer counter begin to count up, we need to do as follows:
If you want to use PWM you should keep TCSR.PWM_EN to be 0 before you initial TCU.

1. Setting TCSR.
   (1) Writing TCSR.INITL to initialize PWM output level.
   (2) Writing TCSR.SD to setting the shutdown mode (Abrupt shutdown or Graceful shutdown).
   (3) Writing TCSR.PRESCALE to set TCNT count clock frequency.

2. Setting TCNT, TDHR and TDFR.

3. Setting TCSR.

(1) Writing TCSR.PWM_EN to set whether enable PWM or disable PWM.
(2) Writing TCSR.EXT_EN, TCSR.RTC_EN or TCSR.PCK_EN to 1 to select the input clock and enable the input clock. Only one of TCSR.EXT_EN, TCSR.RTC_EN and TCSR.PCK_EN can be set to 1.

After initialize the register of timer, we should start the counter as follows:

4. Setting the TESR.TCST bit to 1 to enable the TCNT.

Note: The input clock and PCLK should follows the rules advanced before.

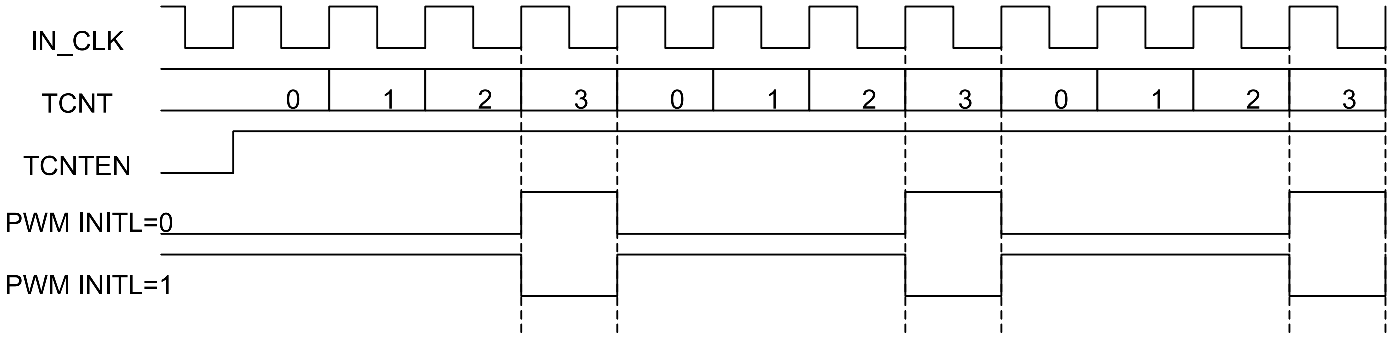
8.4.2 Disable and Shutdown Operation

Setting the TECR.TCCL bit to 1 to disable the TCNT.

8.4.3 Pulse Width Modulator (PWM)

Timer 0~7 can be used as Pulse Width Modulator (PWM). The PWM can be used to control the back light inverter or adjust bright or contrast of LCD panel.

FULL comparison match signal and HALF comparison match signal can determine an attribute of the PWM_OUT waveform. FULL comparison match signal specifies the clock cycle for the PWM module clock. HALF comparison match signal specifies the duty ratio for the PWM module clock.



8.4.4 The flow of using TCU5

Here is the basic flow of using TCU5

Step 1: make sure the counter clock is stop
Set TMR.FMASK5 = 1, TMR.HMASK5 = 1; TCSR5.PRESCALE = 0;
TCSR5.RTC_EN = 0, TCSR5.EXT_EN = 0, TCSR5.PCK_EN = 0;

Step 2: Initial TDFR5 and TDHR5. These registers must be initialed in case of the counter clock is stopped. Only abrupt shutdown mode can be used, which needs set TCSR5.SD to 1. To use PWM output, set TCSR5.INITL to choose start level

Step 3: Initial TCNT5 in case of select and enable pclk without dividing as the counter clock

TCSR5.RTC_EN = 0, TCSR5.EXT_EN = 0, TCSR5.PCK_EN = 1, TCSR5.PRESCALE = 0

Step4: If PWM5 is used, set TCSR5.PWM_EN = 1, in the same condition of above

Step 5: stop the pclk as counter clock by, set TCSR5.PCK_EN = 0

Step 6: set clock divider parameter TCSR.PRESCALE

Step 7: enable the selected clock by set one of TCSR.RTC_EN, TCSR.EXT_EN or TCSR.PCK_EN to 1

Step 8: start counter by write 1 to TESR.TCST5 and enable the corresponding interrupt if needed

Step 9: after counter matching interrupt, please do

- Stop counter by write 1 to TESR.TCCL5
- Disable the selected counter clock by clear the TCSR.RTC_EN, TCSR.EXT_EN or TCSR.PCK_EN
- Disable more interrupt by change TDFR or TDHR to make them not equal to TCNT
- Clear interrupt flag

Note: In TCU5, in any case, if TDFR or TDHR is equal to TCNT, the interrupt flag will be set.

9 Watchdog Timer

9.1 Overview

The watchdog timer is used to resume the controller operation whenever it is disturbed by malfunctions such as noise and system errors. The watchdog timer can generate the reset signal.

Features:

- Generates WDT reset.
- A 16-bit Data register and a 16-bit counter.
- Counter clock uses the input clock selected by software.
  - — PCLK, EXTAL and RTCCLK can be used as the clock for counter
  - — The division ratio of the clock can be set to 1, 4, 16, 64, 256 and 1024 by software

9.2 Register Description

In this section, we will describe the registers in WDT. Following table lists all the registers definition. All WDT register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
TDR	Watchdog Timer Data Register	RW	0x????	0x10002000	16
TCER	Watchdog Counter Enable Register	RW	0x00	0x10002004	8
TCNT	Watchdog Timer Counter	RW	0x????	0x10002008	16
TCSR	Watchdog Timer Control Register	RW	0x0000	0x1000200C	16

9.2.1 Watchdog Control Register (TCSR)

The TCSR is a 16-bit read/write register. It contains the control bits for WDT. It is initialized to 0x00 by any reset.

TCSR **0x1000200C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										PRESCALE			EXT_EN	RTC_EN	PCK_EN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:6	Reserved	These bits always read 0, and written are ignored.	R
5:3	PRESCALE	These bits select the TCNT count clock frequency.	RW

Bit 2	Bit1	Bit 0	Description
0	0	0	Internal clock: CLK/1
0	0	1	Internal clock: CLK/4
0	1	0	Internal clock: CLK/16
0	1	1	Internal clock: CLK/64
1	0	0	Internal clock: CLK/256
1	0	1	Internal clock: CLK/1024
110~111			Reserved

Bits	Name	Description	RW
2	EXT_EN	Select EXTAL as the timer clock input. 1: Enable 0: Disable	RW
1	RTC_EN	Select RTCCLK as the timer clock input.	RW

		1: Enable 0: Disable	
0	PCK_EN	Select PCLK as the timer clock input. 1: Enable 0: Disable	RW

Note:

The input clock of timer and the PCLK should keep to the rules as follows:

Input clock of timer: IN_CLK	**Clock generated from the frequency divider (PRESCALE): DIV_CLK**
PCK_EN == 0, RTC_EN == 1 and EXT_EN == 0 (IN_CLK = RTCCLK)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 0, RTC_EN == 0 and EXT_EN == 1 (IN_CLK = EXTAL)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 1, RTC_EN == 0 and EXT_EN == 0 (IN_CLK = PCLK)	ANY

9.2.2 Watchdog Enable Register (TCER)

The TCER is an 8-bit read/write register. It contains the counter enable control bits for watchdog. It is initialized to 0x00 by any reset.

TCER **0x10002004**

Bit	7	6	5	4	3	2	1	0
	Reserved							TCEN
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:1	Reserved	These bits always read 0, and written are ignored.	R
0	TCEN	Counter enable control. 0: Timer stop. 1: Timer running.	RW

9.2.3 Watchdog Timer Data Register (TDR)

The watchdog timer data register TDR is used to store the data to be compared with the content of the watchdog timer up-counter TCNT. This register can be directly read and written. (Default: indeterminate)

TDR **0x10002000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDR															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

9.2.4 Watchdog Timer Counter (TCNT)

The watchdog timer counter (TCNT) is a 16-bit read/write counter. The up-counter TCNT can be reset to 0 by software and counts up using the prescaler output clock. When TCNT count up to equal to TDR, the comparison match signal will be generated and a WDT reset is generated. The data can be read out at any time. The counter data can be written at any time. (Default: indeterminate)

TCNT **0x10002008**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCNT															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

9.3 Watchdog Timer Function

The following describes steps of using WDT:

1. Setting the PRESCALE of input clock in register TCSR.
2. Set register TDR and TCNT.
3. Select the input clock and enable the input clock in register TCSR.

After initialize the register of timer, we should start the counter as follows:

4. Set TCEN bit in TCER to 1. The counter TCNT begins to count.
5. If TCNT = TDR, a WDT reset will be generated.

Note1: The input clock and PCLK should follows the rules advanced before.

Note2: The clock of WDT can be stopped by setting register TSR, and register TSR can only be set by register TSSR or TSCR. The content of register TSR, TSSR and TSCR can be found in TCU spec.

10 General-Purpose I/O Ports

10.1 Overview

General Purpose I/O Ports (GPIO) is used in generating and capturing application-specific input and output signals. Each port can be programmed as an output, an input or function port that serves certain peripheral. As input, pull up/down can be enabled/disabled for the port and the port also can be configured as level or edge tripped interrupt source.

Features:

- Each port can be configured as an input, an output or an alternate function port.
- Each port can be configured as an interrupt source of low/high level or rising/falling edge triggering. Every interrupt source can be masked independently.
- Each port has an internal pull-up or pull-down resistor connected. The pull-up/down resistor can be disabled.

The GPIO ports, named PA00~31, PB00~31, PC00~31 and PD00~31 are divided into 4 GPIO groups with maximum of 32 GPIO in each group. Group A includes PA00~PA31. Group B includes PB00~31; Group C includes PC00~PC31 and Group D include PD00~PD31. GPIO output 4 interrupts, 1 for every group, to INTC.

For every group, 23 memory-mapped 32-bit registers can be used to operate the GPIO ports:

- PAPIN, PBPIN, PCPIN, PDPIN - PORT PIN Level Register
- PADAT, PBDAT, PCDAT, PDDAT - PORT Data Register
- PADATS, PBDATS, PCDATS, PDDATS - PORT Data Set Register
- PADATC, PBDATC, PCDATC, PDDATC - PORT Data Clear Register
- PAIM, PBIM, PCIM, PDIM - PORT Interrupt Mask Register
- PAIMS, PBIMS, PCIMS, PDIMS - PORT Interrupt Mask Set Register
- PAIMC, PBIMC, PCIMC, PDIMC - PORT Interrupt Mask Clear Register
- PAPE, PBPE, PCPE, PDPE - PORT PULL Disable Register
- PAPES, PBPES, PCPES, PDPES - PORT PULL Disable Set Register
- PAPEC, PBPEC, PCPEC, PDPEC - PORT PULL Disable Clear Register
- PAFUN, PBFUN, PCFUN, PDFUN - PORT Function Register
- PAFUNS, PBFUNS, PCFUNS, PDFUNS - PORT Function Set Register
- PAFUNC, PBFUNC, PBFUNC, PDFUNC - PORT Function Clear Register
- PASEL, PBSEL, PCSEL, PDSEL - PORT Select Register
- PASELS, PBSELS, PCSELS, PDSELS - PORT Select Set Register
- PASELC, PBSELC, PCSELC, PDSELC - PORT Select Clear Register
- PADIR, PBDIR, PCDIR, PDDIR - PORT Direction Register
- PADIRS, PBDIRS, PCDIRS, PDDIRS - PORT Direction Set Register

- PADIRC, PBDIRC, PCDIRC, PDDIRC - PORT Direction Clear Register
- PATRG, PBTRG, PCTRG, PDTRG - PORT Trigger Mode Register
- PATRGS, PBTRGS, PCTRGS, PDTRGS - PORT Trigger Mode Set Register
- PATRGC, PBTRGC, PCTRGC, PDTRGC - PORT Trigger Mode Clear Register
- PAFLG, PBFLG, PCFLG, PDFLG - PORT FLAG Register

Table 10-1 ~ Table 10-4 summarized pull resistor direction and shared function ports for all GPIO.

Table 10-1 GPIO Port A summary

Bit N	PA N	Pull (U/D)	Shared Function Port Selected by		
			PFUN = 1 & PSEL = 0	PFUN = 1 & PSEL = 1	Note
0	00	U	D[0] (io)	-	7
1	01	U	D[1]/D[0] (io)	-	
2	02	U	D[2] (io)	-	7
3	03	U	D[3]/D[1] (io)	-	
4	04	U	D[4] (io)	-	7
5	05	U	D[5]/D[2] (io)	-	
6	06	U	D[6] (io)	-	7
7	07	U	D[7]/D[3] (io)	-	
8	08	U	D[8]/D[4] (io)	-	
9	09	U	D[9]/D[5] (io)	-	
10	10	U	D[10]/D[6] (io)	-	
11	11	U	D[11]/D[7] (io)	-	
12	12	U	D[12]/D[8] (io)	-	
13	13	U	D[13]/D[9] (io)	-	
14	14	U	D[14] (io)	-	7
15	15	U	D[15]/D[10] (io)	-	
16	16	U	D[16] (io)	-	6, 7
17	17	U	D[17]/D[11] (io)	-	6
18	18	U	D[18] (io)	-	6, 7
19	19	U	D[19] (io)	-	6, 7
20	20	U	D[20] (io)	-	6, 7
21	21	U	D[21] (io)	-	6, 7
22	22	U	D[22]/D[12] (io)	-	6
23	23	U	D[23] (io)	-	6, 7
24	24	U	D[24] (io)	-	6, 7
25	25	U	D[25] (io)	-	6, 7
26	26	U	D[26]/D[13] (io)	-	6
27	27	U	D[27] (io)	-	6, 7
28	28	U	D[28]/D[14] (io)	-	6
29	29	U	D[29] (io)	-	6, 7
30	30	U	D[30]/D[15] (io)	-	6
31	31	U	D[31] (io)	-	6, 7

Table 10-2 GPIO Port B summary

Bit N	PB N	Pull (U/D)	Shared Function Port Selected by		
			PFUN = 1 & PSEL = 0	PFUN = 1 & PSEL = 1	Note
0	00	U	A [0] (out)	-	
1	01	U	A [1] (out)	-	
2	02	U	A [2] (out)	-	
3	03	U	A [3] (out)	-	
4	04	U	A [4] (out)	-	
5	05	U	A [5] (out)	-	
6	06	U	A [6] (out)	-	
7	07	U	A [7] (out)	-	
8	08	U	A [8] (out)	-	
9	09	U	A [9] (out)	-	
10	10	U	A [10] (out)	-	
11	11	U	A [11] (out)	-	
12	12	U	A [12] (out)	-	
13	13	U	A [13] (out)	-	
14	14	U	A [14] (out)	-	
15	15	U	A [15] (out)	-	
16	16	U	A [16] (out)	-	
17	17	U	CLS (out)	A [21] (out)	
18	18	U	SPL (out)	A [22] (out)	
19	19	U	DCS_ (out)	-	
20	20	U	RAS_ (out)	-	
21	21	U	CAS_ (out)	-	
22	22	U	SDWE_ & BUFD_ (out)	-	
23	23	U	CKE (out)	-	
24	24	U	CKO (out)	-	
25	25	U	CS1_ (out)	-	
26	26	U	CS2_ (out)	-	
27	27	U	CS3_ (out)	-	6
28	28	U	CS4_ (out)	-	6
29	29	U	RD_ (out)	-	6
30	30	U	WR_ (out)	-	6
31	31	U	WE0_ (out)	-	

Table 10-3 GPIO Port C summary

Bit N	PC N	Pull (U/D)	Shared Function Port Selected by		
			PFUN = 1 & PSEL = 0	PFUN = 1 & PSEL = 1	Note
0	00	U	LCD_D [0] (out)	-	
1	01	U	LCD_D [1] (out)	-	
2	02	U	LCD_D [2] (out)	-	
3	03	U	LCD_D [3] (out)	-	
4	04	U	LCD_D [4] (out)	-	
5	05	U	LCD_D [5] (out)	-	
6	06	U	LCD_D [6] (out)	-	
7	07	U	LCD_D [7] (out)	-	
8	08	U	LCD_D [8] (out)	-	
9	09	U	LCD_D [9] (out)	-	
10	10	U	LCD_D [10] (out)	-	
11	11	U	LCD_D [11] (out)	-	
12	12	U	LCD_D [12] (out)	-	
13	13	U	LCD_D [13] (out)	-	
14	14	U	LCD_D [14] (out)	-	
15	15	U	LCD_D [15] (out)	-	
16	16	U	LCD_D [16] (out)	-	
17	17	U	LCD_D [17] (out)	-	
18	18	U	LCD_PCLK (io)	-	
19	19	U	LCD_HSYNC (io)	-	
20	20	U	LCD_VSYNC (io)	-	
21	21	U	LCD_DE (out)	-	
22	22	U	LCD_PS (out)	A [19] (out)	
23	23	U	LCD_REV (out)	A [20] (out)	
24	24	U	WE1_ (out)	-	
25	25	U	WE2_ (out)	-	6, 7
26	26	U	WE3_ (out)	-	6, 7
27	27	U	WAIT_ (in)	-	
28	28	U	FRE_ (out)	-	
29	29	U	FWE_ (out)	-	
30	30	U	-	-	1
31	31	U		-	2

Table 10-4 GPIO Port D summary

Bit N	PD N	Pull (U/D)	Shared Function Port Selected by			
			PFUN = 1 & PSEL = 0 & PTRG = 0	PFUN = 1 & PSEL = 1 & PTRG = 0	PFUN = 1 & PSEL = 0 & PTRG = 1	Note
0	00	U	CIM_D0 (in)	-		6
1	01	U	CIM_D1 (in)	-		6, 7
2	02	U	CIM_D2 (in)	-		6
3	03	U	CIM_D3 (in)	-		6, 7
4	04	U	CIM_D4 (in)	-		6
5	05	U	CIM_D5 (in)	-		6, 7
6	06	U	CIM_D6 (in)	-		6
7	07	U	CIM_D7 (in)	-		6, 7
8	08	U	MSC_CMD (io)	-		
9	09	U	MSC_CLK (out)	-		
10	10	U	MSC_D0 (io)	-		
11	11	U	MSC_D1 (io)	-		6
12	12	U	MSC_D2 (io)	-		6
13	13	U	MSC_D3 (io)	-		6
14	14	U	CIM_MCLK (out)	-		3, 6, 7
15	15	U	CIM_PCLK (in)	-		6
16	16	U	CIM_VSYN (in)	-		4, 6, 7
17	17	U	CIM_HSYN (in)	-		6, 7
18	18	U	SSI_CLK (out)	SCLK_RSTN (out)		6
19	19	U	SSI_CE0_ (out)	BCLK (io)		6
20	20	U	SSI_DT (out)	SDATO (out)		6
21	21	U	SSI_DR (in)	SDATI (in)		6
22	22	U	SSI_CE1_ & SSI_GPC (out)	SYNC (io)		6
23	23	U	PWM0 (out)	I2C_SDA (io)		
24	24	U	PWM1 (out)	I2C_SCK (io)		
25	25	U	PWM2 (out)	UART0_TXD (out)		
26	26	U	PWM3 (out)	UART0_RXD (in)		
27	27	U	PWM4 (out)	A [17] (out)		6
28	28	U	PWM5 (out)	A [18] (out)		
29	29	-	-	-		5
30	30	U	PWM6 (out)	UART0_CTS_ (in)	UART1_RXD (in)	6, 7
31	31	U	PWM7 (out)	UART0_RTS_ (out)	UART1_TXD (out)	6, 7

Notes:

1. PC30: GPIO group C bit 30. If NAND flash is used, this pin must be used as NAND FRB. (NAND flash ready/busy).
2. PC31: GPIO group C bit 31. No corresponding pin exists for this GPIO. It is only used to select the function between UART and JTAG, which share the same set of pins, by using register PCSEL [31]
   When PCSEL [31]=0, select JTAG function.
   When PCSEL [31]=1, select UART function
3. PD14 is output 0 during the reset (PPRST_, WDT-reset and hibernating-reset) period.
4. The input/output direction of GPD16 pin is controlled by PDDIR[15] when GPD16 is used as GPIO function
5. PD29 is only used as input and interrupt only, and with no pull-up and pull-down.
6. This GPIO pin is not available in Jz4725
7. This GPIO pin is not available in Jz4720

10.2 Register Description

Table 10-2 summarized all memory-mapped registers, which can be programmed to operate GPIO port and alternate function port sharing configuration.

All registers are in 32-bits width. Usually, 1 bit in the register affects a corresponding GPIO port and every GPIO port can be operated independently.

Table 10-2 GPIO Registers

Name	Description	RW	Reset Value	Address	Size
GPIO PORT A					
PAPIN	PORT A PIN Level Register	R	0x00000000	0x10010000	32
PADAT	PORT A Data Register	R	0x00000000	0x10010010	32
PADATS	PORT A Data Set Register	W	0x????????	0x10010014	32
PADATC	PORT A Data Clear Register	W	0x????????	0x10010018	32
PAIM	PORT A Interrupt Mask Register	R	0xFFFFFFFF	0x10010020	32
PAIMS	PORT A Interrupt Mask Set Register	W	0x????????	0x10010024	32
PAIMC	PORT A Interrupt Mask Clear Register	W	0x????????	0x10010028	32
PAPE	PORT A PULL Disable Register	R	0x00000000	0x10010030	32
PAPES	PORT A PULL Disable Set Register	W	0x????????	0x10010034	32
PAPEC	PORT A PULL Disable Clear Register	W	0x????????	0x10010038	32
PAFUN	PORT A Function Register	R	0x00000000	0x10010040	32
PAFUNS	PORT A Function Set Register	W	0x????????	0x10010044	32
PAFUNC	PORT A Function Clear Register	W	0x????????	0x10010048	32
PASEL	PORT A Select Register	R	0x00000000	0x10010050	32
PASELS	PORT A Select Set Register	W	0x????????	0x10010054	32
PASELC	PORT A Select Clear Register	W	0x????????	0x10010058	32
PADIR	PORT A Direction Register	R	0x00000000	0x10010060	32
PADIRS	PORT A Direction Set Register	W	0x????????	0x10010064	32
PADIRC	PORT A Direction Clear Register	W	0x????????	0x10010068	32
PATRG	PORT A Trigger Register	R	0x00000000	0x10010070	32
PATRGS	PORT A Trigger Set Register	W	0x????????	0x10010074	32
PATRGC	PORT A Trigger Clear Register	W	0x????????	0x10010078	32
PAFLG	PORT A FLAG Register	R	0x00000000	0x10010080	32
PAFLGC	PORT A FLAG Clear Register	W	0x????????	0x10010014	32
GPIO PORT B					
PBPIN	PORT B PIN Level Register	R	0x00000000	0x10010100	32
PBDAT	PORT B Data Register	R	0x00000000	0x10010110	32
PBDATS	PORT B Data Set Register	W	0x????????	0x10010114	32
PBDATC	PORT B Data Clear Register	W	0x????????	0x10010118	32

PBIM	PORT B Interrupt Mask Register	R	0xFFFFFFFF	0x10010120	32
PBIMS	PORT B Interrupt Mask Set Register	W	0x????????	0x10010124	32
PBIMC	PORT B Interrupt Mask Clear Register	W	0x????????	0x10010128	32
PBPE	PORT B PULL Enable Register	R	0x00000000	0x10010130	32
PBPES	PORT B PULL Enable Set Register	W	0x????????	0x10010134	32
PBPEC	PORT B PULL Enable Clear Register	W	0x????????	0x10010138	32
PBFUN	PORT B Function Register	R	0x00000000	0x10010140	32
PBFUNS	PORT B Function Set Register	W	0x????????	0x10010144	32
PBFUNC	PORT B Function Clear Register	W	0x????????	0x10010148	32
PBSEL	PORT B Select Register	R	0x00000000	0x10010150	32
PBSELS	PORT B Select Set Register	W	0x????????	0x10010154	32
PBSELC	PORT B Select Clear Register	W	0x????????	0x10010158	32
PBDIR	PORT B Direction Register	R	0x00000000	0x10010160	32
PBDIRS	PORT B Direction Set Register	W	0x????????	0x10010164	32
PBDIRC	PORT B Direction Clear Register	W	0x????????	0x10010168	32
PBTRG	PORT B Trigger Register	R	0x00000000	0x10010170	32
PBTRGS	PORT B Trigger Set Register	W	0x????????	0x10010174	32
PBTRGC	PORT B Trigger Clear Register	W	0x????????	0x10010178	32
PBFLG	PORT B FLAG Register	R	0x00000000	0x10010180	32
PBFLGC	PORT B FLAG Clear Register	W	0x????????	0x10010114	32
GPIO PORT C					
PCPIN	PORT C PIN Level Register	R	0x00000000	0x10010200	32
PCDAT	PORT C Data Register	R	0x00000000	0x10010210	32
PCDATS	PORT C Data Set Register	W	0x????????	0x10010214	32
PCDATC	PORT C Data Clear Register	W	0x????????	0x10010218	32
PCIM	PORT C Interrupt Mask Register	R	0xFFFFFFFF	0x10010220	32
PCIMS	PORT C Interrupt Mask Set Register	W	0x????????	0x10010224	32
PCIMC	PORT C Interrupt Mask Clear Register	W	0x????????	0x10010228	32
PCPE	PORT C PULL Enable Register	R	0x00000000	0x10010230	32
PCPES	PORT C PULL Enable Set Register	W	0x????????	0x10010234	32
PCPEC	PORT C PULL Enable Clear Register	W	0x????????	0x10010238	32
PCFUN	PORT C Function Register	R	0x00000000	0x10010240	32
PCFUNS	PORT C Function Set Register	W	0x????????	0x10010244	32
PCFUNC	PORT C Function Clear Register	W	0x????????	0x10010248	32
PCSEL	PORT C Select Register	R	0x00000000	0x10010250	32
PCSELS	PORT C Select Set Register	W	0x????????	0x10010254	32
PCSELC	PORT C Select Clear Register	W	0x????????	0x10010258	32
PCDIR	PORT C Direction Register	R	0x00000000	0x10010260	32
PCDIRS	PORT C Direction Set Register	W	0x????????	0x10010264	32
PCDIRC	PORT C Direction Clear Register	W	0x????????	0x10010268	32
PCTRG	PORT C Trigger Register	R	0x00000000	0x10010270	32

PCTRGS	PORT C Trigger Set Register	W	0x????????	0x10010274	32
PCTRGC	PORT C Trigger Clear Register	W	0x????????	0x10010278	32
PCFLG	PORT C FLAG Register	R	0x00000000	0x10010280	32
PCFLGC	PORT C FLAG Clear Register	W	0x????????	0x10010214	32
GPIO PORT D					
PDPIN	PORT D PIN Level Register	R	0x00000000	0x10010300	32
PDDAT	PORT D Data Register	R	0x00000000	0x10010310	32
PDDATS	PORT D Data Set Register	W	0x????????	0x10010314	32
PDDATC	PORT D Data Clear Register	W	0x????????	0x10010318	32
PDIM	PORT D Interrupt Mask Register	R	0xFFFFFFFF	0x10010320	32
PDIMS	PORT D Interrupt Mask Set Register	W	0x????????	0x10010324	32
PDIMC	PORT D Interrupt Mask Clear Register	W	0x????????	0x10010328	32
PDPE	PORT D PULL Enable Register	R	0x00000000	0x10010330	32
PDPES	PORT D PULL Enable Set Register	W	0x????????	0x10010334	32
PDPEC	PORT D PULL Enable Clear Register	W	0x????????	0x10010338	32
PDFUN	PORT D Function Register	R	0x00000000	0x10010340	32
PDFUNS	PORT D Function Set Register	W	0x????????	0x10010344	32
PDFUNC	PORT D Function Clear Register	W	0x????????	0x10010348	32
PDSEL	PORT D Select Register	R	0x00000000	0x10010350	32
PDSELS	PORT D Select Set Register	W	0x????????	0x10010354	32
PDSELC	PORT D Select Clear Register	W	0x????????	0x10010358	32
P\DDIR	PORT D Direction Register	R	0x00000000	0x10010360	32
PDDIRS	PORT D Direction Set Register	W	0x????????	0x10010364	32
PDDIRC	PORT D Direction Clear Register	W	0x????????	0x10010368	32
PDTRG	PORT D Trigger Register	R	0x00000000	0x10010370	32
PDTRGS	PORT D Trigger Set Register	W	0x????????	0x10010374	32
PDTRGC	PORT D Trigger Clear Register	W	0x????????	0x10010378	32
PDFLG	PORT D FLAG Register	R	0x00000000	0x10010380	32
PDFLGC	PORT D FLAG Clear Register	W	0x????????	0x10010314	32

Note: PX**** in the description of register as follows means PA****, PB****, PC**** and PD****.

10.2.1 PORT PIN Level Register (PAPIN, PBPIN, PCPIN, PDPIN)

PAPIN, PBPIN, PCPIN and PDPIN are four 32-bit PORT PIN level registers. They are read-only registers.

PAPIN, PBPIN, PCPIN, PDPIN **0x10010000, 0x10010100, 0x10010200, 0x10010300**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PINL31	PINL30	PINL29	PINL28	PINL27	PINL26	PINL25	PINL24	PINL23	PINL22	PINL21	PINL20	PINL19	PINL18	PINL17	PINL16	PINL15	PINL14	PINL13	PINL12	PINL11	PINL10	PINL09	PINL08	PINL07	PINL06	PINL05	PINL04	PINL03	PINL02	PINL01	PINL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	PINL n	Where n = 0 ~ 31 and PINL n = PINL0 ~ PINL31. The PORT PIN level can be read by reading PINL n bit in register PXPIN.	R

PAPIN bits 31-0 correspond to PA31-0; PBPIN to PB31-0; PCPIN to PC31-0 and PDPIN to PD 31-0.

10.2.2 PORT Data Register (PADAT, PBDAT, PCDAT, PDDAT)

PADAT, PBDAT, PCDAT and PDDAT are four 32-bit PORT DATA registers. They are read-only registers.

PADAT, PBDAT, PCDAT, PDDAT **0x10010010, 0x10010110, 0x10010210, 0x10010310**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA31	DATA30	DATA29	DATA28	DATA27	DATA26	DATA25	DATA24	DATA23	DATA22	DATA21	DATA20	DATA19	DATA18	DATA17	DATA16	DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA09	DATA08	DATA07	DATA06	DATA05	DATA04	DATA03	DATA02	DATA01	DATA00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	DATA n	Where n = 0 ~ 31 and DATA n = DATA0 ~ DATA31. The register is used as GPIO data register. When GPIO is used as interrupt the register is no used.	R

PADAT bits 31-0 correspond to PA31-0; PBDAT to PB31-0; PCDAT to PC31-0 and PDDAT to PD 31-0.

10.2.3 PORT Data Set Register (PADATS, PBDATS, PCDATS, PDDATS)

PADATS, PBDATS, PCDATS and PDDATA are four 32-bit PORT DATA set registers. They are write-only registers.

PADATS, PBDATS, PCDATS, PDDATS **0x10010014, 0x10010114, 0x10010214, 0x10010314**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATAS31	DATAS30	DATAS29	DATAS28	DATAS27	DATAS26	DATAS25	DATAS24	DATAS23	DATAS22	DATAS21	DATAS20	DATAS19	DATAS18	DATAS17	DATAS16	DATAS15	DATAS14	DATAS13	DATAS12	DATAS11	DATAS10	DATAS09	DATAS08	DATAS07	DATAS06	DATAS05	DATAS04	DATAS03	DATAS02	DATAS01	DATAS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DATAS n	Writing 1 to DATAS n will set DATA n to 1 in register PXDAT. Writing 0 to DATAS n will no use.	W

PADATS bits 31-0 correspond to PA31-0; PBDATS to PB31-0; PCDATS to PC31-0 and PDDATS to PD 31-0.

10.2.4 PORT Data Clear Register (PADATC, PBDATC, PCDATC, PDDATC)

PADATC, PBDATC, PCDATC and PDDATC are four 32-bit PORT DATA clear registers. They are write-only registers.

PADATC, PBDATC, PCDATC, PDDATC **0x10010018, 0x10010118, 0x10010218, 0x10010318**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATAC31	DATAC30	DATAC29	DATAC28	DATAC27	DATAC26	DATAC25	DATAC24	DATAC23	DATAC22	DATAC21	DATAC20	DATAC19	DATAC18	DATAC17	DATAC16	DATAC15	DATAC14	DATAC13	DATAC12	DATAC11	DATAC10	DATAC09	DATAC08	DATAC07	DATAC06	DATAC05	DATAC04	DATAC03	DATAC02	DATAC01	DATAC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DATAC n	Writing 1 to DATAC n will set DATA n to 0 in register PXDAT. Writing 0 to DATAC n will no use.	W

PADATC bits 31-0 correspond to PA31-0; PBDATC to PB31-0; PCDATC to PC31-0 and PDDATC to PD 31-0.

10.2.5 PORT Mask Register (PAIM, PBIM, PCIM, PDIM)

PAIM, PBIM, PCIM and PDIM are four 32-bit PORT MASK registers. They are read-only registers.

PAIM, PBIM, PCIM, PDIM **0x10010020, 0x10010120, 0x10010220, 0x10010320**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MASK31	MASK30	MASK29	MASK28	MASK27	MASK26	MASK25	MASK24	MASK23	MASK22	MASK21	MASK20	MASK19	MASK18	MASK17	MASK16	MASK15	MASK14	MASK13	MASK12	MASK11	MASK10	MASK09	MASK08	MASK07	MASK06	MASK05	MASK04	MASK03	MASK02	MASK01	MASK00
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W
n	MASK n	Where n = 0 ~ 31 and MASK n = MASK0 ~ MASK31. MASK n is used for mask the interrupt of GPIO n. 0: Enable the pin as an interrupt source. 1: Disable the pin as an interrupt source.	R

PAIM bits 31-0 correspond to PA31-0; PBIM to PB31-0; PCIM to PC31-0 and PDIM to PD 31-0.

10.2.6 PORT Mask Set Register (PAIMS, PBIMS, PCIMS, PDIMS)

PAIMS, PBIMS, PCIMS and PIMS are four 32-bit PORT MASK set registers. They are write-only registers.

PAIMS, PBIMS, PCIMS, PDIMS **0x10010024, 0x10010124, 0x10010224, 0x10010324**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MASKS31	MASKS30	MASKS29	MASKS28	MASKS27	MASKS26	MASKS25	MASKS24	MASKS23	MASKS22	MASKS21	MASKS20	MASKS19	MASKS18	MASKS17	MASKS16	MASKS15	MASKS14	MASKS13	MASKS12	MASKS11	MASKS10	MASKS09	MASKS08	MASKS07	MASKS06	MASKS05	MASKS04	MASKS03	MASKS02	MASKS01	MASKS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	MASKS n	Writing 1 to MASKS n will set MASK n to 1 in register PXIM. Writing 0 to MASKS n will no use.	W

PAIMS bits 31-0 correspond to PA31-0; PBIMS to PB31-0; PCIMS to PC31-0 and PDIMS to PD 31-0.

10.2.7 PORT Mask Clear Register (PAIMC, GBPIMC, PCIMC, PDIMC)

PAIMC, PBIMC, PCIMC and PDIMC are four 32-bit PORT MASK clear registers. They are write-only registers.

PAIMS, PBIMC, PCIMC, PDIMC **0x10010028, 0x10010128, 0x10010228, 0x10010328**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MASKC31	MASKC30	MASKC29	MASKC28	MASKC27	MASKC26	MASKC25	MASKC24	MASKC23	MASKC22	MASKC21	MASKC20	MASKC19	MASKC18	MASKC17	MASKC16	MASKC15	MASKC14	MASKC13	MASKC12	MASKC11	MASKC10	MASKC09	MASKC08	MASKC07	MASKC06	MASKC05	MASKC04	MASKC03	MASKC02	MASKC01	MASKC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	MASKC n	Writing 1 to MASKC n will set MASK n to 0 in register PXIM. Writing 0 to MASKC n will no use.	W

PAIMC bits 31-0 correspond to PA31-0; PBIMC to PB31-0; PCIMC to PC31-0 and PDIMC to PD 31-0.

10.2.8 PORT PULL Disable Register (PAPE, PBPE, PCPE, PDPE)

PAPE, PBPE, PCPE and PDPE are four 32-bit PORT PULL disable registers. They are read-only registers.

PAPE, PBPE, PCPE, PDPE **0x10010030, 0x10010130, 0x10010230, 0x10010330**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULL31	PULL30	PULL29	PULL28	PULL27	PULL26	PULL25	PULL24	PULL23	PULL22	PULL21	PULL20	PULL19	PULL18	PULL17	PULL16	PULL15	PULL14	PULL13	PULL12	PULL11	PULL10	PULL09	PULL08	PULL07	PULL06	PULL05	PULL04	PULL03	PULL02	PULL01	PULL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	PULL n	Where n = 0 ~ 31 and PULL n = PULL0 ~ PULL31. PULL n is used for setting the port to be PULL UP or PULL DOWN enable. 1: No pull up or pull down resistor connects to the port. 0: An internal pull up or pull down resistor connects to the port. Up or down is pin dependence. Please reference to Table 10-1 ~ Table 10-4 for it.	R

PAPE bits 31-0 correspond to PA31-0; PBPE to PB31-0; PCPE to PC31-0 and PDPE to PD 31-0.

10.2.9 PORT PULL Set Register (PAPES, PBPES, PCPES, PDPES)

PAPES, PBPES, PCPES and PDPES are four 32-bit PORT PULL set registers. They are write-only registers.

PAPES, PBPES, PCPES, PDPES **0x10010034, 0x10010134, 0x10010234, 0x10010334**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULLS31	PULLS30	PULLS29	PULLS28	PULLS27	PULLS26	PULLS25	PULLS24	PULLS23	PULLS22	PULLS21	PULLS20	PULLS19	PULLS18	PULLS17	PULLS16	PULLS15	PULLS14	PULLS13	PULLS12	PULLS11	PULLS10	PULLS09	PULLS08	PULLS07	PULLS06	PULLS05	PULLS04	PULLS03	PULLS02	PULLS01	PULLS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PULLS n	Writing 1 to PULLS n will set PULL n to 1 in register PXPE. Writing 0 to PULLS n will no use.	W

PAPES bits 31-0 correspond to PA31-0; PBPES to PB31-0; PCPES to PC31-0 and PDPES to PD 31-0.

10.2.10 PORT PULL Clear Register (PAPEC, PBPEC, PCPEC, PDPEC)

PAPEC, PBPEC, PCPEC and PDPEC are four 32-bit PORT PULL clear registers. They are write-only registers.

PAPES, PBPEC, PCPEC, PDPEC **0x10010038, 0x10010138, 0x10010238, 0x10010338**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULLC31	PULLC30	PULLC29	PULLC28	PULLC27	PULLC26	PULLC25	PULLC24	PULLC23	PULLC22	PULLC21	PULLC20	PULLC19	PULLC18	PULLC17	PULLC16	PULLC15	PULLC14	PULLC13	PULLC12	PULLC11	PULLC10	PULLC09	PULLC08	PULLC07	PULLC06	PULLC05	PULLC04	PULLC03	PULLC02	PULLC01	PULLC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PULLC n	Writing 1 to PULLC n will set PULL n to 0 in register PXPE. Writing 0 to PULLC n will no use.	W

PAPEC bits 31-0 correspond to PA31-0; PBPEC to PB31-0; PCPEC to PC31-0 and PDPEC to PD 31-0.

10.2.11 PORT Function Register (PAFUN, PBFUN, PCFUN, PDFUN)

PAFUN, PBFUN, PCFUN and PDFUN are four 32-bit PORT function registers. They are read-only registers.

PAFUN, PBFUN, PCFUN, PDFUN **0x10010040, 0x10010140, 0x10010240, 0x10010340**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FUN31	FUN30	FUN29	FUN28	FUN27	FUN26	FUN25	FUN24	FUN23	FUN22	FUN21	FUN20	FUN19	FUN18	FUN17	FUN16	FUN15	FUN14	FUN13	FUN12	FUN11	FUN10	FUN09	FUN08	FUN07	FUN06	FUN05	FUN04	FUN03	FUN02	FUN01	FUN00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FUN n	Where n = 0 ~ 31 and FUN n = FUN0 ~ FUN31 In most cases, port is shared with one or more peripheral functions. FUN n controls the owner of the port n. 0: GPIO or Interrupt 1: Alternate Function (Function 0 *1 or Function 1*1) Note: 1. Please reference to Table 10-1 ~ Table 10-4 for the details.	R

PAFUN bits 31-0 correspond to PA31-0; PBFUN to PB31-0; PCFUN to PC31-0 and PDFUN to PD 31-0.

10.2.12 PORT Function Set Register (PAFUNS, PBFUNS, PCFUNS, PDFUNS)

PAFUNS, PBFUNS, PCFUNS and PDFUNS are four 32-bit PORT function set registers. They are write-only registers.

PAFUNS, PBFUNS, PCFUNS, PDFUNS **0x10010044, 0x10010144, 0x10010244, 0x10010344**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FUNS31	FUNS30	FUNS29	FUNS28	FUNS27	FUNS26	FUNS25	FUNS24	FUNS23	FUNS22	FUNS21	FUNS20	FUNS19	FUNS18	FUNS17	FUNS16	FUNS15	FUNS14	FUNS13	FUNS12	FUNS11	FUNS10	FUNS09	FUNS08	FUNS07	FUNS06	FUNS05	FUNS04	FUNS03	FUNS02	FUNS01	FUNS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	FUNS n	Writing 1 to FUNS n will set FUN n to 1 in register PXFUN. Writing 0 to FUNS n will no use.	W

PAFUNS bits 31-0 correspond to PA31-0; PBFUNS to PB31-0; PCFUNS to PC31-0 and PDFUNS to PD 31-0.

10.2.13 PORT Function Clear Register (PAFUNC, PBFUNC, PCFUNC, PDFUNC)

PAFUNC, PBFUNC, PCFUNC and PDFUNC are four 32-bit PORT function clear registers. They are write-only registers.

PAFUNC, PBFUNC, PCFUNC, PDFUNC **0x10010048, 0x10010148, 0x10010248, 0x10010348**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FUNC31	FUNC30	FUNC29	FUNC28	FUNC27	FUNC26	FUNC25	FUNC24	FUNC23	FUNC22	FUNC21	FUNC20	FUNC19	FUNC18	FUNC17	FUNC16	FUNC15	FUNC14	FUNC13	FUNC12	FUNC11	FUNC10	FUNC09	FUNC08	FUNC07	FUNC06	FUNC05	FUNC04	FUNC03	FUNC02	FUNC01	FUNC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	FUNC n	Writing 1 to FUNC n will set FUN n to 0 in register PXFUN. Writing 0 to FUNC n will no use.	W

PAFUNC bits 31-0 correspond to PA31-0; PBFUNC to PB31-0; PCFUNC to PC31-0 and PDFUNC to PD 31-0.

10.2.14 PORT Select Register (PASEL, PBSEL, PCFSEL, PDSEL)

PASEL, PBSEL, PCSEL and PDSEL are four 32-bit PORT select registers. They are read-only registers.

PASEL, PBSEL, PCSEL, PDSEL **0x10010050, 0x10010150, 0x10010250, 0x10010350**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SEL31	SEL30	SEL29	SEL28	SEL27	SEL26	SEL25	SEL24	SEL23	SEL22	SEL21	SEL20	SEL19	SEL18	SEL17	SEL16	SEL15	SEL14	SEL13	SEL12	SEL11	SEL10	SEL09	SEL08	SEL07	SEL06	SEL05	SEL04	SEL03	SEL02	SEL01	SEL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	SEL n	Where n = 0 ~ 31 and SEL n = SEL0 ~ SEL31 SEL n is used for selecting the function of GPIO. When PXFUN = 0: 0: GPIO 1: Interrupt When PXFUN = 1: 0: Alternate Function 0[*1] 1: Alternate Function 1[*1] Note: 1. Please reference to Table 10-1 ~ Table 10-4 for the details.	R

PASEL bits 31-0 correspond to PA31-0; PBSEL to PB31-0; PCSEL to PC31-0 and PDSEL to PD 31-0.

10.2.15 PORT Select Set Register (PASELS, PBSELS, PCSELS, PDSELS)

PASELS, PBSELS, PCSELS and PDSELS are four 32-bit PORT select set registers. They are write-only registers.

PASELS, PBSELS, PCSELS, PFDSELS **0x10010054, 0x10010154, 0x10010254, 0x10010354**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SELS31	SELS30	SELS29	SELS28	SELS27	SELS26	SELS25	SELS24	SELS23	SELS22	SELS21	SELS20	SELS19	SELS18	SELS17	SELS16	SELS15	SELS14	SELS13	SELS12	SELS11	SELS10	SELS09	SELS08	SELS07	SELS06	SELS05	SELS04	SELS03	SELS02	SELS01	SELS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	SELS n	Writing 1 to SELS n will set SEL n to 1 in register PXSEL. Writing 0 to SELS n will no use.	W

PASELS bits 31-0 correspond to PA31-0; PBSELS to PB31-0; PCSELS to PC31-0 and PDSELS to PD 31-0.

10.2.16 PORT Select Clear Register (PASELC, PBSELC, PCSELC, PDSELC)

PASELC, PBSELC, PCSELC and PDSELC are four 32-bit PORT select clear registers. They are write-only registers.

PASELC, PBSELC, PCSELC, PDSELC **0x10010058, 0x10010158, 0x10010258, 0x10010358**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SELC31	SELC30	SELC29	SELC28	SELC27	SELC26	SELC25	SELC24	SELC23	SELC22	SELC21	SELC20	SELC19	SELC18	SELC17	SELC16	SELC15	SELC14	SELC13	SELC12	SELC11	SELC10	SELC09	SELC08	SELC07	SELC06	SELC05	SELC04	SELC03	SELC02	SELC01	SELC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	SELC n	Writing 1 to SELC n will set SEL n to 0 in register PXSEL. Writing 0 to SELC n will no use.	W

PASELC bits 31-0 correspond to PA31-0; PBSELC to PB31-0; PCSELC to PC31-0 and PDSELC to PD 31-0.

10.2.17 PORT Direction Register (PADIR, PBDIR, PCDIR, PDDIR)

PADIR, PBDIR, PCDIR and PDDIR are four 32-bit PORT direction registers. They are read-only registers.

PADIR, PBDIR, PCDIR, PDDIR **0x10010060, 0x10010160, 0x10010260, 0x10010360**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIR31	DIR30	DIR29	DIR28	DIR27	DIR26	DIR25	DIR24	DIR23	DIR22	DIR21	DIR20	DIR19	DIR18	DIR17	DIR16	DIR15	DIR14	DIR13	DIR12	DIR11	DIR10	DIR09	DIR08	DIR07	DIR06	DIR05	DIR04	DIR03	DIR02	DIR01	DIR00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	DIR n	Where n = 0 ~ 31 and DIR n = DIR0 ~ DIR31 DIR n is used for setting the direction of port or setting the trigger direction of interrupt trigger. GPIO Direction: (GPIO Function) 0: INPUT 1: OUTPUT Interrupt Trigger Direction: (Interrupt Function) PXTRG = 0: 0: Low Level Trigger 1: High Level Trigger PXTRG =1: 0: Falling Edge Trigger 1: Rising Edge Trigger	R

PADIR bits 31-0 correspond to PA31-0; PBDIR to PB31-0; PCDIR to PC31-0 and PDDIR to PD 31-0.

10.2.18 PORT Direction Set Register (PADIRS, PBDIRS, PCDIRS, PDDIRS)

PADIRS, PBDIRS, PCDIRS and PDDIRS are four 32-bit PORT direction set registers. They are write-only registers.

PADIRS, PBDIRS, PCDIRS, PDDIRS **0x10010064, 0x10010164, 0x10010264, 0x10010364**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIRS31	DIRS30	DIRS29	DIRS28	DIRS27	DIRS26	DIRS25	DIRS24	DIRS23	DIRS22	DIRS21	DIRS20	DIRS19	DIRS18	DIRS17	DIRS16	DIRS15	DIRS14	DIRS13	DIRS12	DIRS11	DIRS10	DIRS09	DIRS08	DIRS07	DIRS06	DIRS05	DIRS04	DIRS03	DIRS02	DIRS01	DIRS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DIRS n	Writing 1 to DIRS n will set DIR n to 1 in register PXDIR. Writing 0 to DIRS n will no use.	W

PADIRS bits 31-0 correspond to PA31-0; PBDIRS to PB31-0; PCDIRS to PC31-0 and PDDIRS to PD 31-0.

10.2.19 PORT Direction Clear Register (PADIRC, PBDIRC, PCDIRC, PDDIRC)

GPDIRC0, GPDIRC1, GPDIRC2 and GPDIRC3 are four 32-bit PORT direction clear registers. They are write-only registers.

PADIRS, PBDIRC, PCDIRC, PDDIRC **0x10010068, 0x10010168, 0x10010268, 0x10010368**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIRC31	DIRC30	DIRC29	DIRC28	DIRC27	DIRC26	DIRC25	DIRC24	DIRC23	DIRC22	DIRC21	DIRC20	DIRC19	DIRC18	DIRC17	DIRC16	DIRC15	DIRC14	DIRC13	DIRC12	DIRC11	DIRC10	DIRC09	DIRC08	DIRC07	DIRC06	DIRC05	DIRC04	DIRC03	DIRC02	DIRC01	DIRC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DIRC n	Writing 1 to DIRC n will set DIR n to 0 in register PXDIR. Writing 0 to DIRC n will no use.	W

PADIRC bits 31-0 correspond to PA31-0; PBDIRC to PB31-0; PCDIRC to PC31-0 and PDDIRC to PD 31-0.

10.2.20 PORT Trigger Register 0, 1, 2 and 3 (PATRG, PBTRG, PCTRG, PDTRG)

PATRG, PBTRG, PCTRG and PDTRG are four 32-bit PORT trigger registers. They are read-only registers.

PATRG, PBTRG, PCTRG, PDTRG **0x10010070, 0x10010170, 0x10010270, 0x10010370**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIG31	TRIG30	TRIG29	TRIG28	TRIG27	TRIG26	TRIG25	TRIG24	TRIG23	TRIG22	TRIG21	TRIG20	TRIG19	TRIG18	TRIG17	TRIG16	TRIG15	TRIG14	TRIG13	TRIG12	TRIG11	TRIG10	TRIG09	TRIG08	TRIG07	TRIG06	TRIG05	TRIG04	TRIG03	TRIG02	TRIG01	TRIG00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	TRIG n	Where n = 0 ~ 31 and TRIG n = TRIG00 ~ TRIG31 TRIG n is used for setting the trigger mode for interrupt. When GPIO is used as interrupt function: 0: Level Trigger Interrupt. 1: Edge Trigger Interrupt. When GPIO is used as alternate function: 0: Alternate Function Group 0. 1: Alternate Function Group 1.	R

PATRG bits 31-0 correspond to PA31-0; PBTRG to PB31-0; PCTRG to PC31-0 and PDTRG to PD 31-0.

10.2.21 PORT Trigger Set Register (PATRGS, PBTRGS, PCTRGS, PDTRGS)

PATRGS, PBTRGS, PCTRGS and PDTRGS are four 32-bit PORT trigger set registers. They are write-only registers.

PATRGS, PBTRGS, PCTRGS, PDTRGS **0x10010074, 0x10010174, 0x10010274, 0x10010374**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIGS31	TRIGS30	TRIGS29	TRIGS28	TRIGS27	TRIGS26	TRIGS25	TRIGS24	TRIGS23	TRIGS22	TRIGS21	TRIGS20	TRIGS19	TRIGS18	TRIGS17	TRIGS16	TRIGS15	TRIGS14	TRIGS13	TRIGS12	TRIGS11	TRIGS10	TRIGS09	TRIGS08	TRIGS07	TRIGS06	TRIGS05	TRIGS04	TRIGS03	TRIGS02	TRIGS01	TRIGS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	TRIGS n	Writing 1 to TRIGS n will set TRIG n to 1 in register PXTRG. Writing 0 to TRIGS n will no use.	W

PATRGS bits 31-0 correspond to PA31-0; PBTRGS to PB31-0; PCTRGS to PC31-0 and PDTRGS to PD 31-0.

10.2.22 PORT Trigger Clear Register (PATRGC, PBTRGC, PCTRGC, PDTRGC)

PATRGC, PBTRGC, PCTRGC and PDTRGC are four 32-bit PORT trigger clear registers. They are write-only registers.

PATRGC, PBTRGC, PCTRGC, PDTRGC **0x10010078, 0x10010178, 0x10010278, 0x10010378**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIGC31	TRIGC30	TRIGC29	TRIGC28	TRIGC27	TRIGC26	TRIGC25	TRIGC24	TRIGC23	TRIGC22	TRIGC21	TRIGC20	TRIGC19	TRIGC18	TRIGC17	TRIGC16	TRIGC15	TRIGC14	TRIGC13	TRIGC12	TRIGC11	TRIGC10	TRIGC09	TRIGC08	TRIGC07	TRIGC06	TRIGC05	TRIGC04	TRIGC03	TRIGC02	TRIGC01	TRIGC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	TRIGC n	Writing 1 to TRIGC n will set TRIG n to 0 in register PXTRG. Writing 0 to TRIGC n will no use.	W

PATRGC bits 31-0 correspond to PA31-0; PBTRGC to PB31-0; PCTRGC to PC31-0 and PDTRGC to PD 31-0.

10.2.23 PORT FLAG Register (PAFLG, PBFLG, PCFLG, PDFLG)

PAFLG, PBFLG, PCFLG and PDFLG are four 32-bit GPIO FLAG registers. They are read-only registers.

PAFLG, PBFLG, PCFLG, PDFLG **0x10010080, 0x10010180, 0x10010280, 0x10010380**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLAG31	FLAG30	FLAG29	FLAG28	FLAG27	FLAG26	FLAG25	FLAG24	FLAG23	FLAG22	FLAG21	FLAG20	FLAG19	FLAG18	FLAG17	FLAG16	FLAG15	FLAG14	FLAG13	FLAG12	FLAG11	FLAG10	FLAG09	FLAG08	FLAG07	FLAG06	FLAG05	FLAG04	FLAG03	FLAG02	FLAG01	FLAG00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FLAG n	Where n = 0 ~ 31 and FLAG n = FLAG00 ~ FLAG31 FLAG n is interrupt flag bit for checking the interrupt whether to happen. When GPIO is used as interrupt function and the interrupt happened, the FLAG n in PXFLG will be set to 1.	R

PAFLG bits 31-0 correspond to PA31-0; PBFLG to PB31-0; PCFLG to PC31-0 and PDFLG to PD 31-0.

10.2.24 PORT FLAG Clear Register (PAFLGC, PBFLGC, PCFLGC, PDFLGC)

PAFLGC, PBFLGC, PCFLGC and PDFLGC are four 32-bit GPIO FLAG Clear registers. They are read-only registers.

PAFLGC, PBFLGC, PCFLGC, PDFLGC **0x10010014, 0x10010114, 0x10010214, 0x10010314**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLAGC31	FLAGC30	FLAGC29	FLAGC28	FLAGC27	FLAGC26	FLAGC25	FLAGC24	FLAGC23	FLAGC22	FLAGC21	FLAGC20	FLAGC19	FLAGC18	FLAGC17	FLAGC16	FLAGC15	FLAGC14	FLAGC13	FLAGC12	FLAGC11	FLAGC10	FLAGC09	FLAGC08	FLAGC07	FLAGC06	FLAGC05	FLAGC04	FLAGC03	FLAGC02	FLAGC01	FLAGC00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FLAGC n	When GPIO is used as interrupt function and when write 1 to the bit, the bit FLAG n in PXFLG will be cleared.	R

PAFLGC bits 31-0 correspond to PA31-0; PBFLGC to PB31-0; PCFLGC to PC31-0 and PDFLGC to PD 31-0.

10.3 Program Guide

10.3.1 GPIO Function Guide

1. Set PXFUN to choose the function of GPIO / Interrupt by writing 1 to register PXFUNC.
2. Set PXSEL to choose the function of GPIO by writing 1 to register PXSELC.
3. Set PXDIR to choose the direction of GPIO by writing 1 to register PXDIRS or PXDIRC.
4. Others.

(1) You can read the PORT PIN level by reading register PXPIN.

(2) You can use register PXDAT as normal data register. The register can be set by register PXDATS and PXDATC.

(3) You can set PXPE by writing 1 to register PXPES or PXPE to use Internal pull-up/down resistor or not.

10.3.2 Alternate Function Guide

1. Set PXFUN to 0 by writing 1 to register PXFUNC. (Ready state)

2. Set PXTRG to choose the alternate function group 0 by writing 1 to register PXTRGC.
   Set PXTRG to choose the alternate function group 1 by writing 1 to register PXTRGS.

3. Set PXSEL to choose the alternate function 0 by writing 1 to register PXSELC.
   Set PXSEL to choose the alternate function 1 by writing 1 to register PXSELS.

4. Set PXFUN to choose the function of alternate function by writing 1 to register PXFUNS.

10.3.3 Interrupt Function Guide

First you should keep GPIO status.

1. Set PXIM by writing 1 to register PXIMS.

2. Set PXTRG to choose the interrupt trigger mode by writing 1 to register PXTRGS or PXTRGC.

3. Set PXFUN to choose the function of GPIO / Interrupt by writing 1 to register or PXFUNC.
4. Set PXSEL to choose the Interrupt function by writing 1 to register PXSELS.

5. Set PXDIR to choose the direction of interrupt trigger by writing 1 to register PXDIRS or PXDIRC.

6. Set the PXFLGC register to clear the interrupt flag.
7. Clear PXIM by writing 1 to register PXIMC to enable the GPIO interrupt.

8. Others.

You should check the level interrupt whether to happen as follows:

(1) When the PIN level read from register PXPIN is the same with what you have set in register PXTRG and PXDIR, then the level interrupt happened.

(2) When the PIN level read from register PXPIN is different from what you have set in register PXTRG and PXDIR, then the level interrupt did not happen.

10.3.4 Disable Interrupt Function Guide

1. Set PXIM by writing 1 to register PXIMS.

2. Set PXTRG to 0 by writing 1 to register PXTRGC.
3. Set PXDIR to 0 by writing 1 to register PXDIRC.

4. Set PXFUN to 0 by writing 1 to register or PXFUNC.
5. Set PXSEL to 0 by writing 1 to register PXSELC.

11 LCD Controller

11.1 Overview

The JZ integrated LCD controller has the capabilities to driving the latest industry standard STN and TFT LCD panels. It also supports some special TFT panels used in consuming electronic products. The controller performs the basic memory based frame buffer and palette buffer to LCD panel data transfer through use of a dedicated DMA controller. Temporal dithering (frame rate modulation) is supported for STN LCD panels.

Features:

(1) Basic Features:

- Support ITU601/656 data format.
- Single and Dual panel displays in STN mode.
- Single panel displays in TFT mode.
- Display size up to 800x600.
- Internal palette RAM 256x16 bits.

(2) Colors Supports:

- Encoded pixel data of 1, 2, 4, 8 or 16 BPP in STN mode.
- Support 2, 4, 16 grayscales and up to 4096 colors in STN mode.
- Encoded pixel data of 1, 2, 4, 8, 16, 18 or 24 BPP in TFT mode.
- Support 65,536(65K), 262,144(260K) and up to 16,777,216 (16M) colors in TFT mode.

(3) Panel Supports:

- Support 1, 2, 4, 8 data output pins in STN mode.
- Support 8-bit serial data output for 1bpp, 2bpp, 4bpp, 8bpp, 16bpp, 18bpp and 24bpp in TFT mode.
- Support 16-bit parallel data output for 1bpp, 2bpp, 4bpp, 8bpp and 16bpp in TFT mode.
- Support 18-bit parallel data output for 1bpp, 2bpp, 4bpp, 8bpp, 16bpp, 18bpp and 24bpp in TFT mode.

11.2 Pin Description

Table 11-1 LCD Controller Pins Description

Name	I/O	Description
Lcd_pclk	Input/Output	Display device pixel clock
Lcd_vsync	Input/Output	Display device vertical synchronize pulse
Lcd_hsync	Input/Output	Display device horizontal synchronize pulse
Lcd_de	Output	Display device is STN: AC BIAS Pin Display device is NOT STN: data enable Pin
Lcd_d[17:0]	Output	Display device data pins
Lcd_spl*[1]	Output	Programmable special pin for generating control signals
Lcd_cls*[1]	Output	Programmable special pin for generating control signals
Lcd_ps*[1]	Output	Programmable special pin for generating control signals
Lcd_rev*[1]	Output	Programmable special pin for generating control signals

Note1:

The mode and timing of special pin Lcd_spl, Lcd_cls, Lcd_ps and Lcd_rev can be seen in **part 1.7 LCD Controller Pin Mapping.**

11.3 Block Diagram

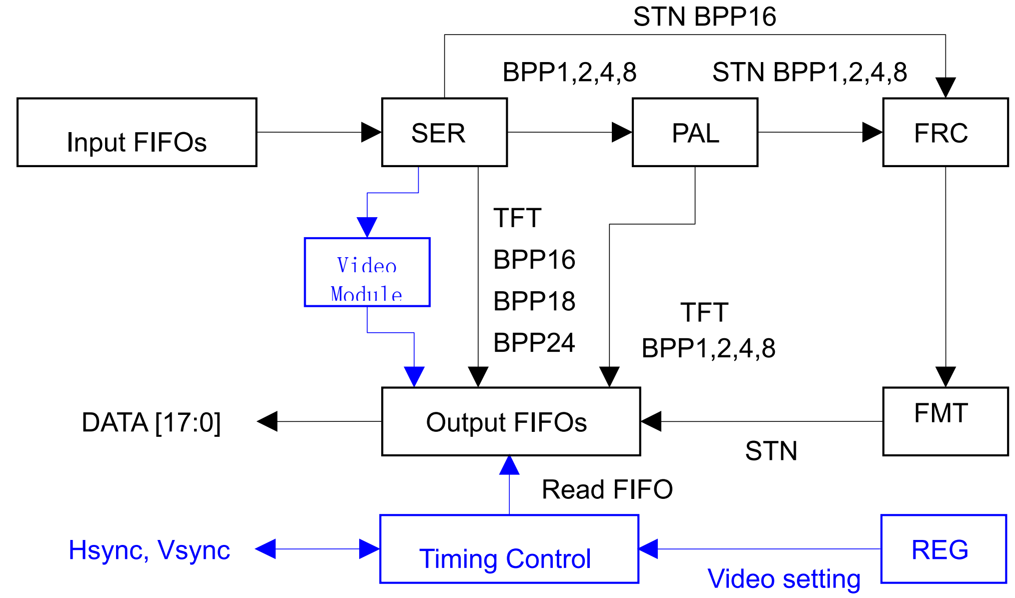

STN BPP16
BPP1,2,4,8
STN BPP1,2,4,8
Input FIFOs
SER
PAL
FRC
Video Module
TFT
BPP16
BPP18
BPP24
TFT
BPP1,2,4,8
DATA [17:0]
Output FIFOs
FMT
STN
Read FIFO
Hsync, Vsync
Timing Control
REG
Video setting

Figure 11-1 Block Diagram

11.4 LCD Display Timing

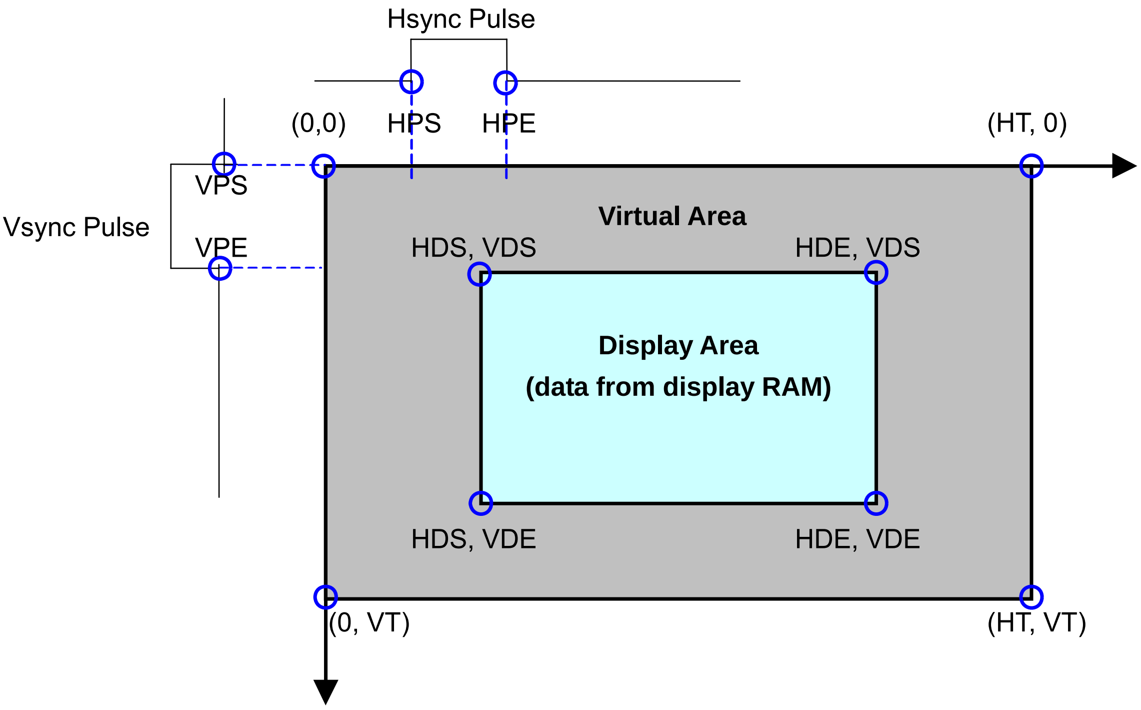

Hsync Pulse
(0,0)
HPS
HPE
(HT, 0)
Vsync Pulse
VPS
VPE
Virtual Area
HDS, VDS
HDE, VDS
Display Area
(data from display RAM)
HDS, VDE
HDE, VDE
(0, VT)
(HT, VT)

Figure 11-2 Display Parameters

Note1:

VPS === 0
VSYNC pulse always start at point (0,0)

Note2:

H: Horizontal	V: Vertical	T: Total
D: Display Area	P: Pulse	
S: Start point	E: End point	

In the (H, V) Coordinates:

1. The gray rectangle (0, 0) to (HT, VT) is "Virtual Area";
2. The blue rectangle (HDS, VDS) to (HDE, VDE) is "Display Area";
3. VPS, VPE defines the VSYNC signal timing; (VPS always be zero)
4. HPS, HPE defines the HSYNC signal timing;

All timing parameters start with "H" is measured in lcd_pclk ticks.
All timing parameters start with "V" is measured in lcd_hsync ticks.

This diagram describes the general LCD panel parameters, these can be set via the registers that describes in next section.

11.5 TV Encoder Timing

Some of Video Encoders for TV (Tele Vision) require interlaced timing interface.

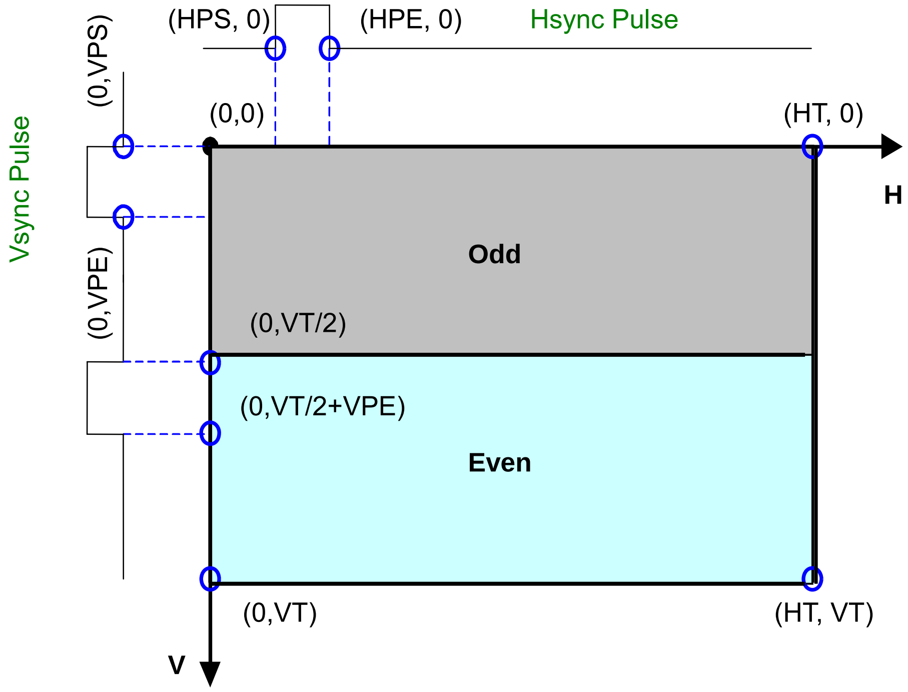


Figure 11-2 TV-Encoder Display Parameters

Note1:

Even Field contains one more blank line.

e.g. For standard PAL timing, Odd filed has 312 lines while even field has 313 lines.

Note2:

Interlace mode generate 2 vsync pulse for each field. The second vsync start at (VT/2), end at (VT/2 + VPE)

Note3:

Display Area & Virtual Area has the same size. VDS=HDS=0, VDE=VT, HDE=HT

11.6 Register Description

Table 11-2 LCD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
LCDCFG	RW	0x00000000	0x13050000	32
LCDVSYNC	RW	0x00000000	0x13050004	32
LCDHSYNC	RW	0x00000000	0x13050008	32
LCDVAT	RW	0x00000000	0x1305000C	32
LCDDAH	RW	0x00000000	0x13050010	32
LCDDAV	RW	0x00000000	0x13050014	32
LCDPS*1	RW	0x00000000	0x13050018	32
LCDCLS*1	RW	0x00000000	0x1305001C	32
LCDSPL*1	RW	0x00000000	0x13050020	32
LCDREV*1	RW	0x00000000	0x13050024	32
LCDCTRL	RW	0x00000000	0x13050030	32
LCDSTATE	RW	0x00000000	0x13050034	32
LCDIID	R	0x00000000	0x13050038	32
LCDDA0	RW	0x00000000	0x13050040	32
LCDSA0	R	0x00000000	0x13050044	32
LCDFID0	R	0x00000000	0x13050048	32
LCDCMD0	R	0x00000000	0x1305004C	32
LCDDA1*2	RW	0x00000000	0x13050050	32
LCDSA1*2	R	0x00000000	0x13050054	32
LCDFID1*2	R	0x00000000	0x13050058	32
LCDCMD1*2	R	0x00000000	0x1305005C	32

Note: *1: These registers are only used for SPECIAL TFT panels.

*2: These registers are only used for Dual Panel STN panels.

11.6.1 Configure Register (LCDCFG)

LCDCFG **0x13050000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCDPIN	Reserved							PSM	CLSM	SPLM	REVM	HSYNM	PCLKM	INVDAT	SYNDIR	PSP	CLSP	SPLP	REVP	HSP	PCP	DEP	VSP	18/16	Reserved	PDW		MODE			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	LCDPIN[*1]	LCD PIN Select bit. These two bits are used to choose the function of LCD PINS or SLCD PINS. The function of pins is as follows: **LCDPIN** – **PIN SELECT** 0 – LCD PIN 1 – SLCD PIN	RW
30:24	Reserved	These bits always read 0, and written are ignored.	R
23	PSM	PS signal mode bit: 1 – disabled, 0 – enabled.	RW
22	CLSM	CLS signal mode bit: 1 – disabled, 0 – enabled.	RW
21	SPLM	SPL signal mode bit: 1 – disabled, 0 – enabled.	RW
20	REVM	REV signal mode bit: 1 – disabled, 0 – enabled.	RW
19	HSYNM	H-Sync signal polarity choice function: 1 – disabled, 0 – enabled.	RW
18	PCLKM	Dot clock signal polarity choice function: 1 – disabled, 0 – enabled.	RW
17	INVDAT	Inverse output data: 0 – normal, 1 – inverse.	RW
16	SYNDIR	V-Sync and H-Sync direction: 0 – output, 1 – input.	RW
15	PSP	PS pin reset state	RW
14	CLSP	CLS pin reset state	RW
13	SPLP	SPL pin reset state	RW
12	REVP	REV pin reset state	RW
11	HSP	H-Sync polarity: 0 – active high, 1 – active low	RW
10	PCP	Pix-clock polarity: 0 – data translations at rising edge. 1 – data translations at falling edge.	RW
9	DEP	Data Enable polarity: 0 – active high, 1 – active low	RW
8	VSP	V-Sync polarity: 0 – leading edge is rising edge. 1 – leading edge is falling edge.	RW
7	18/16	18-bit TFT Panel or 16-bit TFT Panel. This bit will be available when MODE [3:2] is equal to 0, 0 - 16-bit TFT Panel. 1 - 18-bit TFT Panel.	RW

<table>
<tr><td>6</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>5:4</td><td>PDW</td><td>STN pins utilization
<table>
<tr><td></td><td>Signal Panel</td></tr>
<tr><td>00</td><td>Lcd_d[0]</td></tr>
<tr><td>01</td><td>Lcd_d[0:1]</td></tr>
<tr><td>10</td><td>Lcd_d[0:3]</td></tr>
<tr><td>11</td><td>Lcd_d[0:7]</td></tr>
<tr><td></td><td>Dual-Monochrome Panel</td></tr>
<tr><td>00</td><td>Reserved</td></tr>
<tr><td>01</td><td>Reserved</td></tr>
<tr><td>10</td><td>Upper panel: lcd_d[3:0], lower panel: lcd_d[11:8]</td></tr>
<tr><td>11</td><td>Upper panel: lcd_d[7:0], lower panel: lcd_d[15:8]</td></tr>
</table></td><td>RW</td></tr>
<tr><td>3:0</td><td>MODE</td><td>Display Device Mode Select
<table>
<tr><td></td><td>LCD Panel</td></tr>
<tr><td>0000</td><td>Generic 16-bit/18-bit Parallel TFT Panel</td></tr>
<tr><td>0001</td><td>Special TFT Panel Mode1</td></tr>
<tr><td>0010</td><td>Special TFT Panel Mode2</td></tr>
<tr><td>0011</td><td>Special TFT Panel Mode3</td></tr>
<tr><td>0100</td><td>Non-Interlaced CCIR656</td></tr>
<tr><td>0101</td><td>Reserved</td></tr>
<tr><td>0110</td><td>Interlaced CCIR656</td></tr>
<tr><td>0111</td><td>Reserved</td></tr>
<tr><td>1000</td><td>Single-Color STN Panel</td></tr>
<tr><td>1001</td><td>Single-Monochrome STN Panel</td></tr>
<tr><td>1010</td><td>Dual-Color STN Panel</td></tr>
<tr><td>1011</td><td>Dual-Monochrome STN Panel</td></tr>
<tr><td>1100</td><td>8-bit Serial TFT</td></tr>
<tr><td>1101</td><td>Reserved</td></tr>
<tr><td>1110</td><td>Reserved</td></tr>
<tr><td>1111</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
</table>

Note[*1]

LCDPIN	PIN25	PIN24	PIN23	PIN22	PIN21	PIN20	PIN19	PIN18	PIN17-0
0	LCD PCLK	LCD VSYNC	LCD HSYNC	LCD DE	LCD REV	LCD PS	LCD CLS	LCD SPL	LCD D [17:0]
1	SLCD CLK	SLCD CS	SLCD RS						SLCD D [17:0]

The direction of PIN25 is set by register LPCDR.LCS in CPM SPEC.

The direction of PIN23 and PIN23 are set by register LCDCFG.SYNDIR

11.6.2 Vertical Synchronize Register (LCDVSYNC)

LCDVSYNC **0x13050004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					VPS											Reserved					VPE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	VPS	V-Sync Pulse start position, fixed to 0 (in line clock)	R
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	VPE	V-Sync Pulse end position (in line clock)	RW

11.6.3 Horizontal Synchronize Register (LCDHSYNC)

LCDHSYNC **0x13050008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					HPS											Reserved					HPE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	HPS	H-Sync pulse start position (in dot clock)	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	HPE	H-Sync pulse end position (in dot clock)	RW

11.6.4 Virtual Area Setting (LCDVAT)

LCDVAT **0x1305000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					HT											Reserved					VT										

RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	HT	Horizontal Total size (in dot clock, sum of display area and blank space)	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	VT	Vertical Total size (in line clock, sum of display area and blank space)	RW

11.6.5 Display Area Horizontal Start/End Point (LCDDAH)

LCDDAH **0x13050010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					HDS											Reserved					HDE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	HDS	Horizontal display area start (in dot clock)	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	HDE	Horizontal display area end (in dot clock)	RW

11.6.6 Display Area Vertical Start/End Point (LCDDAV)

LCDDAV **0x13050014**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					VDS											Reserved					VDE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	VDS	Vertical display area start position (in line clock)	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	VDE	Vertical display area end position (in line clock)	RW

11.6.7 PS Signal Setting (LCDPS)

LCDPS **0x13050018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					PSS											Reserved					PSE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	PSS	PS signal start position (in dot clock). In STN mode, PS signal is ignored. But this register is used to define the AC BIAs signal. AC BIAs signal will toggle very N lines per frame. PSS defines the Toggle position.	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	PSE	PS signal end position (in dot clock). In STN mode, PSE defines N, which described in PSS.	RW

11.6.8 CLS Signal Setting (LCDCLS)

LCDCLS **0x1305001C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					CLSS											Reserved					CLSE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	CLSS	CLS signal start position (in dot clock)	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	CLSE	CLS signal end position (in dot clock)	RW

11.6.9 SPL Signal Setting (LCDSPL)

LCDSPL **0x13050020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					SPLS											Reserved					SPLE										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	SPLS	SPL signal start position (in dot clock)	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10:0	SPLE	SPL signal end position (in dot clock)	RW

11.6.10 REV Signal Setting (LCDREV)

LCDREV **0x13050024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					REVS											Reserved															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
26:16	REVS	REV signal start position (in dot clock)	RW
15:0	Reserved	These bits always read 0, and written are ignored.	R

11.6.11 Control Register (LCDCTRL)

LCDCTRL **0x13050030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved		BST		RGB	OFUP	FRC		PDD								Reserved		EOFM	SOFM	OFUM	IFUM0	IFUM1	LDDM	QDM	BEDN	PEDN	DIS	ENA	BPP		

RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:30</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>29:28</td><td>BST</td><td>Burst Length Selection
<table>
<tr><th></th><th>Burst Length</th></tr>
<tr><td>00</td><td>4 word</td></tr>
<tr><td>01</td><td>8 word</td></tr>
<tr><td>10</td><td>16 word</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
<tr><td>27</td><td>RGB</td><td>RGB mode: 0 – RGB565, 1 – RGB555.</td><td>RW</td></tr>
<tr><td>26</td><td>OFUP</td><td>Output FIFO under run protection: 0 – disable, 1 – enable.</td><td>RW</td></tr>
<tr><td>25:24</td><td>FRC</td><td>STN FRC Algorithm Selection
<table>
<tr><th></th><th>Grayscale</th></tr>
<tr><td>00</td><td>16 grayscale</td></tr>
<tr><td>01</td><td>4 grayscale</td></tr>
<tr><td>10</td><td>2 grayscale</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
<tr><td>23:16</td><td>PDD</td><td>Load Palette Delay Counter</td><td>RW</td></tr>
<tr><td>15:14</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>13</td><td>EOFM</td><td>Mask end of frame interrupt: 0 – INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>12</td><td>SOFM</td><td>Mask start of frame interrupt: 0 –INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>11</td><td>OFUM</td><td>Mask out FIFO under run interrupt: 0 –INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>10</td><td>IFUM0</td><td>Mask in FIFO 0 under run interrupt: 0 –INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>9</td><td>IFUM1</td><td>Mask in FIFO 1 under run interrupt: 0 –INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>8</td><td>LDDM</td><td>Mask LCD disable done interrupt: 0 –INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>7</td><td>QDM</td><td>Mask LCD quick disable done interrupt: 0 –INT-disabled, 1 –INT-enabled</td><td>RW</td></tr>
<tr><td>6</td><td>BEDN</td><td>Endian selection: 0 – same as system Endian, 1 – reverse endian format</td><td>RW</td></tr>
<tr><td>5</td><td>PEDN</td><td>Endian in byte: 0 – msb first, 1 – lsb first</td><td>RW</td></tr>
<tr><td>4</td><td>DIS</td><td>Disable controller indicate bit: 0 – enable, 1 – in disabling or disabled</td><td>RW</td></tr>
<tr><td>3</td><td>ENA</td><td>Enable controller: 0 – disable, 1 – enable</td><td>W</td></tr>
</table>

<table>
<tr><td>2:0</td><td>BPP</td><td>Bits Per Pixel
<table>
<tr><th></th><th>Bits Per Pixel</th></tr>
<tr><td>000</td><td>1 bpp</td></tr>
<tr><td>001</td><td>2 bpp</td></tr>
<tr><td>010</td><td>4 bpp</td></tr>
<tr><td>011</td><td>8 bpp</td></tr>
<tr><td>100</td><td>15/16 bpp</td></tr>
<tr><td>101</td><td>18 bpp/24bpp</td></tr>
<tr><td>110</td><td>Reserved</td></tr>
<tr><td>111</td><td>Reserved</td></tr>
</table>
</td><td>RW</td></tr>
</table>

11.6.12 Status Register (LCDSTATE)

LCDSTATE **0x13050034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									QD	Reserved	EOF	SOF	OUF	IFU0	IFU1	LDD
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	QD	LCD Quick disable: 0 – not been quick disabled, 1 – quick disabled done.	RW
6	Reserved	These bits always read 0, and written are ignored.	R
5	EOF	End of Frame indicate bit.	RW
4	SOF	Start of Frame indicate bit.	RW
3	OUF	Out FIFO under run.	RW
2	IFU0	In FIFO 0 under run.	RW
1	IFU1	In FIFO 1 under run.	RW
0	LDD	LCD disable: 0 – not been normal disabled, 1 – been normal disabled	RW

11.6.13 Interrupt ID Register (LCDIID)

LCDIID is a read-only register that contains a copy of the Frame ID register (LCDFID) from the descriptor currently being processed when a start of frame (SOF) or end of frame (EOF) interrupt is generated. LCDIID is written to only when an unmasked interrupt of the above type is signaled and there are no other unmasked interrupts in the LCD controller pending. As such, the register is considered to be sticky and will be overwritten only when the signaled interrupt is cleared by writing the LCD controller status register. For dual-panel displays, LCDIID is written only when both channels have reached a given state.

LCDIID is written with the last channel to reach that state. (i.e. LCDFID of the last channel to reach SOF would be written in LCDIID if SOF interrupts are enabled). Reserved bits must be written with zeros and reads from them must be ignored.

LCDIID **0x13050038**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IID																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	IID	A copy of Frame ID register, which transferred from Descriptor.	RW

11.6.14 Descriptor Address Register0, 1 (LCDDA0, 1)

A frame descriptor is a 4-word block, aligned on 4-word (16-byte) boundary, in external memory:

WORD [0] contains the physical address for next LCDDAx
WORD [1] contains the physical address for LCDSAx
WORD [2] contains the value for LCDFIDx
WORD [3] contains the value for LCDCMDx

Software must write the physical address of the first descriptor to LCDDAx before enabling the LCD Controller. Once the LCD Controller is enabled, the first descriptor is read, and all 4 registers are written by the DMAC. The next frame descriptor pointed to by LCDDAx is loaded into the registers for the associated DMA channel after all data for the current descriptor has been transferred.

Note: If only one frame buffer is used in external memory, the LCDDAx field (word [0] of the frame descriptor) must point back to itself. That is to say, the value of LCDDAx is the physical address of itself.

Read/write registers LCDDA0 and LCDDA1, corresponding to DMA channels 0 and 1, contain the physical address of the next descriptor in external memory. The DMAC fetches the descriptor at this location after finishing the current descriptor. On reset, the bits in this register are zero. The target address must be aligned to 16-byte boundary. Bits [3:0] of the address must be zero.

LCDDA0, LCDDA1 **0x13050040, 0x13050050**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DA0, 1																															

RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	DA0, 1	Next descriptor physical address. And descriptor structure as following: WORD [0]: next descriptor physical address. WORD [1]: the buffer physical address. WORD [2]: the buffer ID value. (Only for debug) WORD [3]: the buffer property. The value is same as LCDCMD.	RW

11.6.15 Source Address Register0, 1 (LCDSA0, 1)

Registers LCDSA0 and LCDSA1, corresponding to DMA channels 0 and 1, contain the **physical** address of frame buffer or palette buffer in external memory. The address must be aligned on an 4, 8, or 16 word boundary according to register LCDCTRL .BST. If this descriptor is for palette data, LCDSA0 points to the memory location of the palette buffer. If this descriptor is for frame data, LCDSAx points to the memory location of the frame buffer. This address is incremented by hardware as the DMAC fetches data from memory. If desired, the Frame ID Register can be used to hold the initial frame source address.

LCDSA0, LCDSA1 **0x13050044, 0x13050054**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SA0, 1																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	SA0, 1	Buffer start address. (Only for driver debug)	R

11.6.16 Frame ID Register0 (LCDFID0,1)

Registers LCDFID0 and LCDFID1, corresponding to DMA channels 0 and 1, contain an ID field that describes the current frame. The particular use of this field is up to the software. This ID register is copied to the LCD Controller Interrupt ID Register when an interrupt occurs.

LCDFID0, LCDFID1 **0x13050048, 0x13050058**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	FID0, 1																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	FID0, 1	Frame ID. (Only for debug)	R

11.6.17 DMA Command Register0, 1 (LCDCMD0, 1)

LCDCMD0, LCDCMD1 **0x1305004C, 0x1305005C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SOFINT	EOFINT	Reserved	PAL	Reserved				LEN																							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	SOFINT	Enable start of frame interrupt. When SOFINT =1, the DMAC sets the start of frame bit (LCDSTATE.SOF) when starting a new frame. The SOF bit is set after a new descriptor is loaded from memory and before the palette/frame data is fetched. In dual-panel mode, LCDSTATE.SOF is set only when both channels reach the start of frame and both frame descriptors have SOFINT set. SOFINT must not be set for palette descriptors in dual-panel mode, since only one channel is ever used to load the palette descriptor.	R
30	EOFINT	Enable end of frame interrupt. When EOFINT =1, the DMAC sets the end of frame bit (LCDSTATE.EOF) after fetching the last word in the frame buffer. In dual-panel mode, LCDSTATE.EOF is set only when both channels reach the end of frame and both frame descriptors have EOFINT set. EOFINT must not be set for palette descriptors in dual-panel mode, since only one channel is ever used to load the palette descriptor.	R
29	Reserved	These bits always read 0, and written are ignored.	R

28	PAL	The descriptor contains a palette buffer. PAL indicates that data being fetched will be loaded into the palette RAM. If PAL =1, the palette RAM data is loaded via DMA channel 0 as follows: In bpp1, 2, 4, 8 mode, software must load the palette at least once after enabling the LCD. In bpp16 mode, PAL must be 0.	R
27:24	Reserved	These bits always read 0, and written are ignored.	R
23:0	LEN	The buffer length value (in WORD). The LEN bit field determines the number of bytes of the buffer size pointed by LCDSAx. LEN = 0 is not valid. DMAC fetch data according to LEN. Each time one or more word(s) been fetched, LEN is decreased automatically. Software can read LEN.	R

11.7 LCD Controller Pin Mapping

There are several mapping schemes for different LCD panels.

11.7.1 TFT and CCIR656 Pin Mapping

Pin	Generic 8-bit Serial TFT	Generic 18/16-bit Parallel TFT		Special TFT 1 18/16-bit Parallel		Special TFT 2 18/16-bit Parallel		Special TFT 3 18/16-bit Parallel		CCIR656 8-bit	CCIR656 16-bit
Lcd_pclk/ Slcd_clk	CLK	CLK		DCLK		CLK		HCLK		CLK	CLK
Lcd_vsync/ Slcd_cs	VSYNC	VSYNC		SPS		GSRT		STV		VSYNC	VSYNC
Lcd_hsync/ Slcd_rs	HSYNC	HSYNC		LP		GPCK		STH		HSYNC	HSYNC
Lcd_de	DE	DE		-		-		-		-	-
Lcd_ps	-	-		Pulse mode		Toggle mode		Toggle mode		-	-
Lcd_cls	-	-		Pulse mode		Pulse mode		Pulse mode		-	-
Lcd_rev	-	-		Toggle mode		Toggle mode		Toggle mode		-	-
Lcd_spl	-	-		Pulse mode		Pulse mode		**Toggle mode**		-	-
Lcd_dat17	-	R5	-	R5	-	R5	-	R5	-	-	-
Lcd_dat16	-	R4	-	R4	-	R4	-	R4	-	-	-
Lcd_dat15	-	R3	R5	R3	R5	R3	R5	R3	R5	-	D15
Lcd_dat14	-	R2	R4	R2	R4	R2	R4	R2	R4	-	D14

Lcd_dat13	-	R1	R3	R1	R3	R1	R3	R1	R3	-	D13
Lcd_dat12	-	R0	R2	R0	R2	R0	R2	R0	R2	-	D12
Lcd_dat11	-	G5	R1	G5	R1	G5	R1	G5	R1	-	D11
Lcd_dat10	-	G4	G5	G4	G5	G4	G5	G4	G5	-	D10
Lcd_dat9	-	G3	G4	G3	G4	G3	G4	G3	G4	-	D9
Lcd_dat8	-	G2	G3	G2	G3	G2	G3	G2	G3	-	D8
Lcd_dat7	R7/G7/B7	G1	G2	G1	G2	G1	G2	G1	G2	D7	D7
Lcd_dat6	R6/G6/B6	G0	G1	G0	G1	G0	G1	G0	G1	D6	D6
Lcd_dat5	R5/G5/B5	B5	G0	B5	G0	B5	G0	B5	G0	D5	D5
Lcd_dat4	R4/G4/B4	B4	B5	B4	B5	B4	B5	B4	B5	D4	D4
Lcd_dat3	R3/G3/B3	B3	B4	B3	B4	B3	B4	B3	B4	D3	D3
Lcd_dat2	R2/G2/B2	B2	B3	B2	B3	B2	B3	B2	B3	D2	D2
Lcd_dat1	R1/G1/B1	B1	B2	B1	B2	B1	B2	B1	B2	D1	D1
Lcd_dat0	R0/G0/B0	B0	B1	B0	B1	B0	B1	B0	B1	D0	D0

11.7.2 Single STN Pin Mapping

Pin	**Color STN**	**Mono STN**			
	PDW=3	PDW=0	PDW=1	PDW=2	PDW=3
Lcd_pclk	CLK	CLK	CLK	CLK	CLK
Lcd_vsync	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC
Lcd_hsync	HSYNC	HSYNC	HSYNC	HSYNC	HSYNC
Lcd_de	BIAS	BIAS	BIAS	BIAS	BIAS
Lcd_ps	-	-	-	-	-
Lcd_cls	-	-	-	-	-
Lcd_rev	-	-	-	-	-
Lcd_spl	-	-	-	-	-
Lcd_dat17	-	-	-	-	-
Lcd_dat16	-	-	-	-	-
Lcd_dat15	-	-	-	-	-
Lcd_dat14	-	-	-	-	-
Lcd_dat13	-	-	-	-	-
Lcd_dat12	-	-	-	-	-
Lcd_dat11	-	-	-	-	-
Lcd_dat10	-	-	-	-	-
Lcd_dat9	-	-	-	-	-
Lcd_dat8	-	-	-	-	-
Lcd_dat7	D7	-	-	-	D7
Lcd_dat6	D6	-	-	-	D6
Lcd_dat5	D5	-	-	-	D5
Lcd_dat4	D4	-	-	-	D4
Lcd_dat3	D3	-	-	D3	D3
Lcd_dat2	D2	-	-	D2	D2
Lcd_dat1	D1	-	D1	D1	D1
Lcd_dat0	D0	D0	D0	D0	D0

11.7.3 Dual Panel STN Pin Mapping

Pin	Color STN	Mono STN			
	PDW=3	PDW=0	PDW=1	PDW=2	PDW=3
Lcd_pclk	CLK	-	-	CLK	CLK
Lcd_vsync	VSYNC	-	-	VSYNC	VSYNC
Lcd_hsync	HSYNC	-	-	HSYNC	HSYNC
Lcd_de	BIAS	-	-	BIAS	BIAS
Lcd_ps	-	-	-	-	-
Lcd_cls	-	-	-	-	-
Lcd_rev	-	-	-	-	-
Lcd_spl	-	-	-	-	-
Lcd_dat17	-	-	-	-	-
Lcd_dat16	-	-	-	-	-
Lcd_dat15	UD7	-	-	-	UD7
Lcd_dat14	UD6	-	-	-	UD6
Lcd_dat13	UD5	-	-	-	UD5
Lcd_dat12	UD4	-	-	-	UD4
Lcd_dat11	UD3	-	-	UD3	UD3
Lcd_dat10	UD2	-	-	UD2	UD2
Lcd_dat9	UD1	-	-	UD1	UD1
Lcd_dat8	UD0	-	-	UD0	UD0
Lcd_dat7	LD7	-	-	-	LD7
Lcd_dat6	LD6	-	-	-	LD6
Lcd_dat5	LD5	-	-	-	LD5
Lcd_dat4	LD4	-	-	-	LD4
Lcd_dat3	LD3	-	-	LD3	LD3
Lcd_dat2	LD2	-	-	LD2	LD2
Lcd_dat1	LD1	-	-	LD1	LD1
Lcd_dat0	LD0	-	-	LD0	LD0

11.8 Display Timing

11.8.1 General 16-bit and 18-bit TFT Timing

This section shows the general 16-bit and 18-bit TFT LCD timing diagram, the polarity of signal "Vsync", "Hsync", and "PCLK" can be programmed correspond to the LCD panel specification.

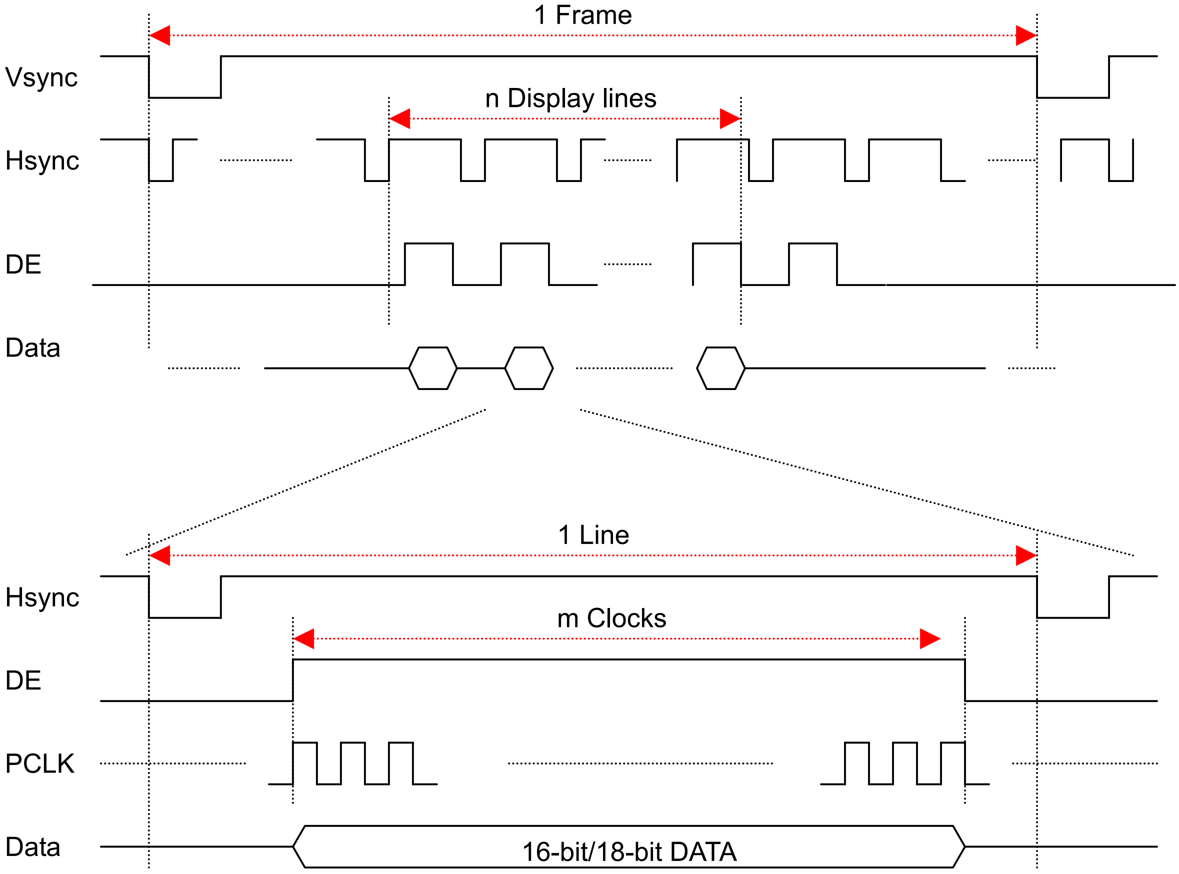

1 Frame
Vsync
n Display lines
Hsync
DE
Data
1 Line
Hsync
m Clocks
DE
PCLK
Data
16-bit/18-bit DATA

Figure 11-2 General 16-bit and 18-bit TFT LCD Timing

11.8.2 8-bit Serial TFT Timing

This section shows the 8-bit serial TFT LCD timing diagram, the polarity of signal “Vsync”, “Hsync”, and “PCLK” can be programmed correspond to the LCD panel specification.

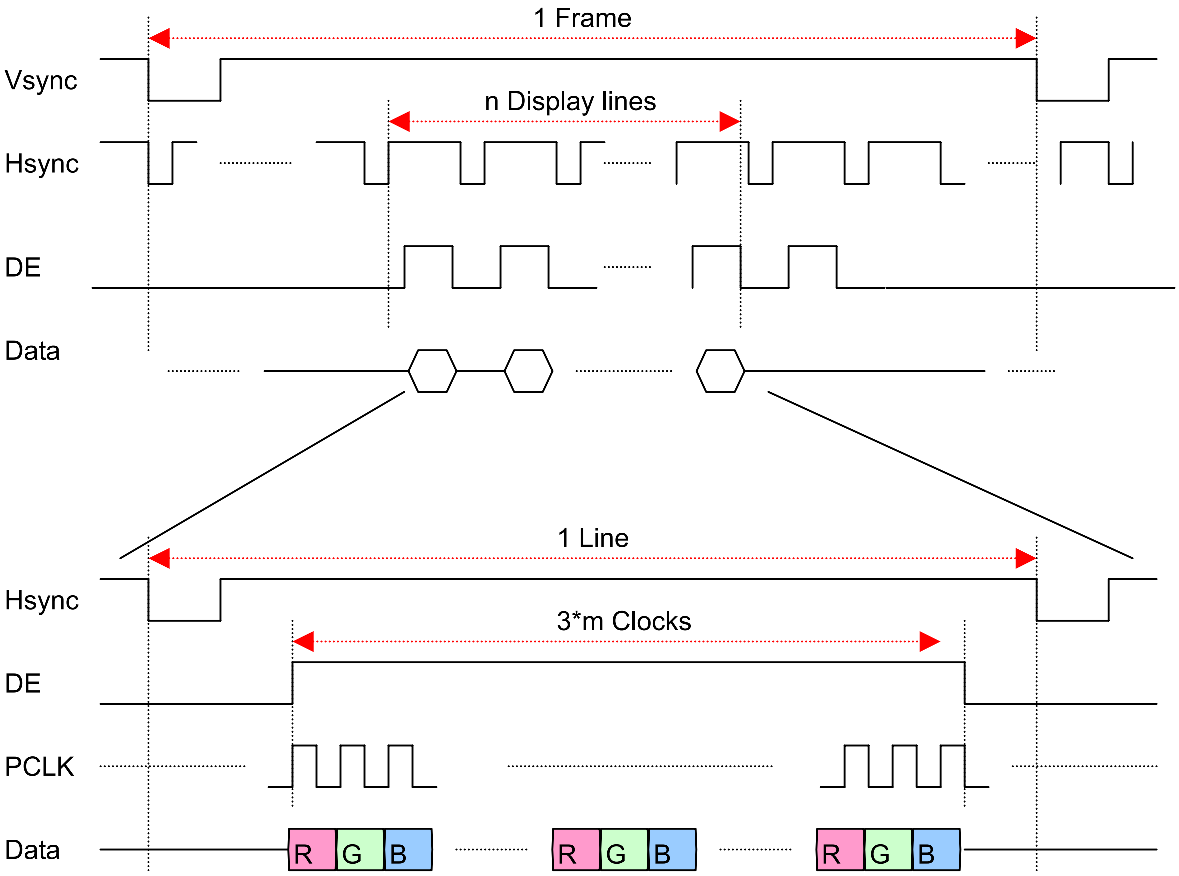

1 Frame
Vsync
n Display lines
Hsync
DE
Data
1 Line
Hsync
3*m Clocks
DE
PCLK
Data
R
G
B
R
G
B
R
G
B

Figure 11-3 8-bit serial TFT LCD Timing (24bpp)

11.8.3 Special TFT Timing

Based on the general TFT LCD support, this controller also provides 4 special signals that can be programmed to general some special timing used for some panel. All 4 signals are worked in two modes: pulse mode and toggle mode. Signal "CLS" is fixed in pulse mode, and "REV" in toggle mode. The work mode of signals "SPL" and "PS" are defined in the special TFT LCD mode 1 to mode 3, either pulse mode or toggle mode. The position and polarity of these 4 signals can be programmed via registers. The Figures show the two modes as follows: (The toggle mode of signal "SPL" is different with the others signal. "SPL" does toggle after display line.)

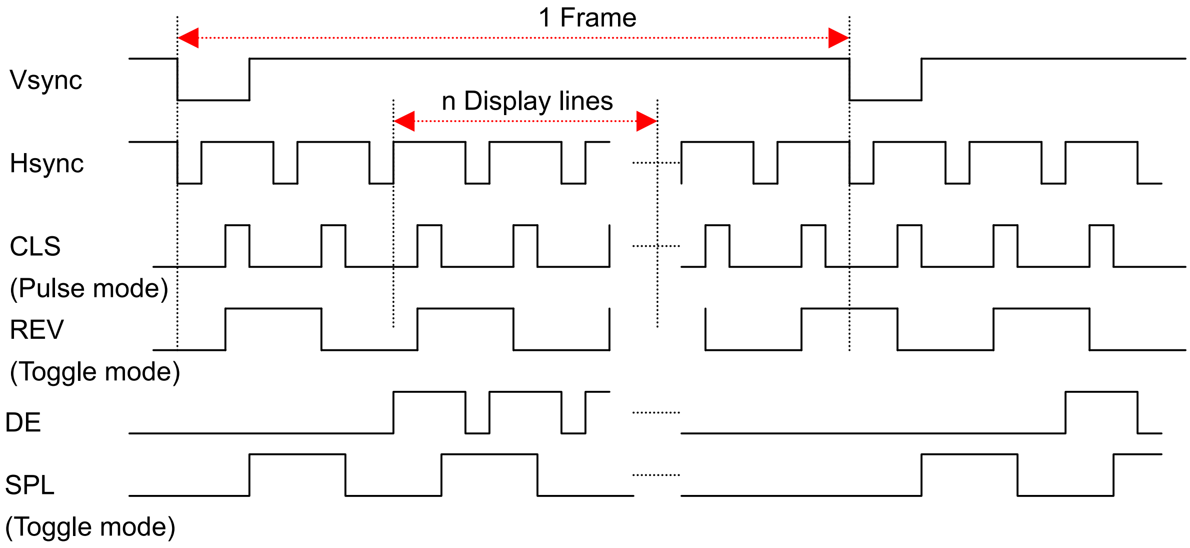

1 Frame
n Display lines
Vsync
Hsync
CLS
(Pulse mode)
REV
(Toggle mode)
DE
SPL
(Toggle mode)

Figure 11-3 Special TFT LCD Timing 1

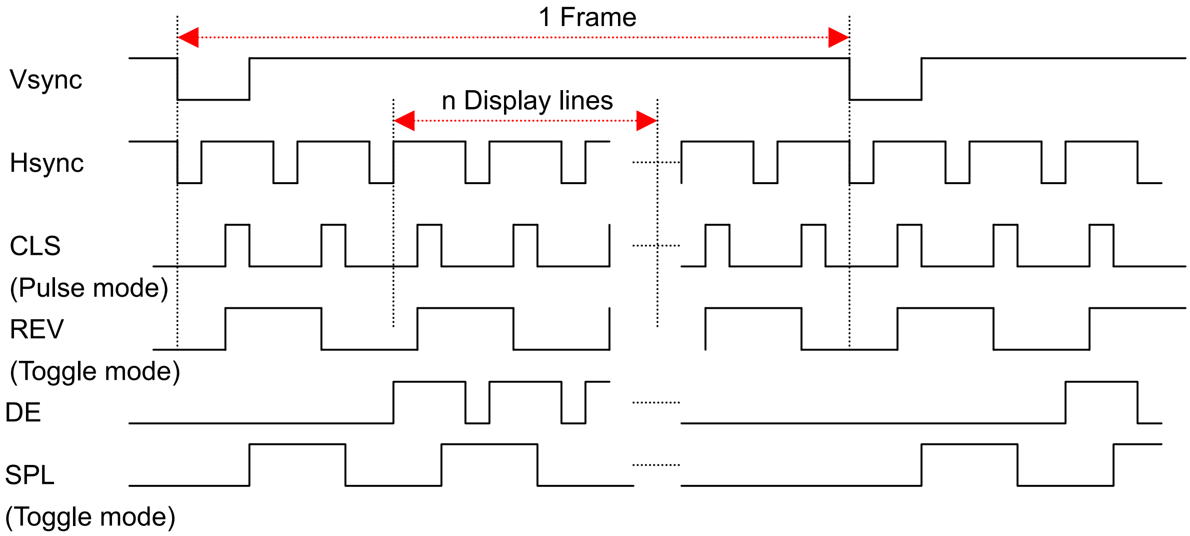

1 Frame
n Display lines
Vsync
Hsync
CLS
(Pulse mode)
REV
(Toggle mode)
DE
SPL
(Toggle mode)

Figure 11-4 Special TFT LCD Timing 2

These two Figures show the timing of pulse mode and toggle mode, the pulse mode timing is same and the toggle mode timing is different. Timing 1 shows the condition when the total lines in 1 frame is odd (the number of display is even and the number of blank is odd), so the phase of REV inverse at the first line of each frame and the phase of SPL dose not inverse at the first line of each frame. Timing 2 shows the condition when the total lines in 1 frame is even (the number of display is even and the number of blank is even), so the phase of REV and SPL dose not inverse at the first line of each frame.

When LCDC is enabled ,there will be a null line to be add before transferring data to LCD panel. So the toggle mode exept SPL signal of special 3 TFT mode is when reset level is high,the first valid edge will be rising edge. SPL signal of special 3 TFT mode is when reset level is high,the first valid edge will be falling edge.

11.9 Format of Palette

This LCD controller contains a palette RAM with 256-entry x 16-bit used only for BPP8, BPP4, BPP2 and BPP1. Palette RAM data is loaded directly from the external memory palette buffer by DMAC channel 0. Each word of palette buffer contains 2 palette entries.

- In 8-bpp modes, palette buffer size is128 words.
- In 4-bpp modes, palette buffer size is 8 words.
- In 2-bpp modes, palette buffer size is 2 words.
- In 1-bpp modes, palette buffer size is 1 word.
- In 16/18/24-bpp modes, has no palette buffer.

Palette buffer base address	Bit: 31 . . . 16	Bit: 15 . . . 0
Palette entry	Entry-1 bit: 15 . . . 0	Entry-0 bit: 15 . . . 0
Palette buffer base address + 4	Bit: 31 . . . 16	Bit: 15 . . . 0
Palette entry	Entry-3 bit: 15 . . . 0	Entry-2 bit: 15 . . . 0
Palette buffer base address + 8	Bit: 31 . . . 16	Bit: 15 . . . 0
Palette entry	Entry-5 bit: 15 . . . 0	Entry-4 bit: 15 . . . 0

11.9.1 STN

For STN Panel, 16-bpp pixel data is encoded with RGB 565 or RGB 555.
Please refer to register LCDCTRL.RGB.

BPP 16, RGB 565, pixel encoding for STN Panel

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

BPP 16, RGB 555, pixel encoding for STN Panel

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

11.9.2 TFT

BPP 16, RGB 565, pixel encoding for TFT Panel

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

Note: For BPP 16, 18, 24, palette is bypass.

11.10 Format of Frame Buffer

11.10.1 16bpp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

11.10.2 18bpp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	R5	R4	R3	R2	R1	R0	0	0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G5	G4	G3	G2	G1	G0	0	0	B5	B4	B3	B2	B1	B0	0	0

11.10.3 24bpp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	R7	R6	R5	R4	R3	R2	R1	R0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

11.11 Format of Data Pin Utilization

11.11.1 Mono STN

In Mono STN mode, data pin pixel ordering of one LCD screen row. Column 0 is the first pixel of a screen row.

Upper panel								
Panel data width	**Col0**	**Col1**	**Col2**	**Col3**	**Col4**	**Col5**	**Col6**	**Col7**
1 bit	D0	D0	D0	D0	D0	D0	D0	D0
2 bit	D1	D0	D1	D0	D1	D0	D1	D0

4 bit	D3	D2	D1	D0	D3	D2	D1	D0
8 bit	D7	D6	D5	D4	D3	D2	D1	D0
Lower panel (dual-panel mode)								
4 bit	D11	D10	D9	D8	D11	D10	D9	D8
8 bit	D15	D14	D13	D12	D11	D10	D9	D8

11.11.2 Color STN

In Color STN mode, data pin pixel ordering of one LCD screen row. Column 0 is the first pixel of a screen row.

Upper panel							
Col0 (R)	**Col0 (G)**	**Col0 (B)**	**Col1 (R)**	**Col1 (G)**	**Col1 (B)**	**Col2 (R)**	**Col2 (G)**
D7	D6	D5	D4	D3	D2	D1	D0
Lower panel (dual-panel mode)							
D15	D14	D13	D12	D11	D10	D9	D8

11.11.3 18-bit Parallel TFT

Col0 (RGB)																	
D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

11.11.4 16-bit Parallel TFT

Col0 (RGB)															
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

11.11.5 8-bit Serial TFT (24bpp)

Col0 (R)							
D7	D6	D5	D4	D3	D2	D1	D0
Col0 (G)							
D7	D6	D5	D4	D3	D2	D1	D0
Col0 (B)							
D7	D6	D5	D4	D3	D2	D1	D0

11.12 LCD Controller Operation

11.12.1 Set LCD Controller Device Clock and Pixel Clock

The LCD Controller has 2 clock input: device clock and pixel clock. The both clocks are generated by CPM (Clock and Power Manager). The frequency of the 2 clocks can be set by CPM registers. CPM registers CPCCR.LDIV and CPCCR.PCS set LCD device clock division ratio, and LPCDR set LCD pixel clock division ratio. Please refer to CPM spec for detail.

LCD device clock is the LCD controller’s internal clock while LCD pixel clock is output to drive LCD panel. There have 2 rules for LCD clocks:

(1) For TFT Panel, the frequency of LCD device clock must be at least 1.5 times of LCD pixel clock.
(2) For STN Panel, the frequency of LCD device clock must be at least 3 times of LCD pixel clock.

LCD panel determines the frequency of LCD pixel clock.

11.12.2 Enabling the Controller

If the LCD controller is being enabled for the first time after system reset or sleep reset, all of the LCD registers must be programmed as follows:

(1) Write the frame descriptors and, if needed, the palette descriptor to memory.

(2) Program the entire LCD configuration registers except the Frame Descriptor Address Registers (LCDDAx) and the LCD Controller enable bit (LCDCTRL.ENA).

(3) Program LCDDAx with the memory address of the palette/frame descriptor.

(4) Enable the LCD controller by writing to LCDCTRL.ENA.

If the LCD controller is being re-enabled, there has not been a reset since the last programming; only the registers LCDDAx and LCDCTRL.ENA need to be reprogrammed. The LCD Controller Status Register (LCDSTATE) must also be written to clear any old status flags before re-enabling the LCD controller.

Once the LCD controller has been enabled, do not write new values to LCD registers except LCDCTRL.ENA or DIS or LCDDA0/1.

11.12.3 Disabling the Controller

The LCD controller can be disabled in two ways: regular and quick.

(1) Regular disabling:
Regular disabling is accomplished by setting the disable bit, LCDCTRL.DIS. The other bits in LCDCTRL must not be changed — read the register, set the DIS bit, and rewrite the register. This method causes the LCD controller to stop cleanly at the end of a frame. The LCD Disable Done bit, LCDSTATE.LDD, is set when the LCD controller finishes displaying the last frame, and the enable bit, LCDCTRL.ENA, is cleared automatically by hardware.
LCDCTRL.DIS must be set zero when enabling the controller.

(2) Quick disabling:

Quick disabling is accomplished by clearing the enable bit, LCDCTRL.ENA. The LCD controller will finish any current DMA transfer, stop driving the panel, setting the LCD Quick Disable bit (LCDSTATE.QD) and shut down immediately. This method is intended for situations such as a battery fault, where system bus traffic has to be minimized immediately so the processor can have enough time to store critical data to memory before the loss of power. The LCD controller must not be re-enabled until the QD bit is set, indicating that the quick shutdown is complete. Do not set the DIS bit when a quick disabling command has been issued.

Note: It is strongly recommended that software set the "LCD Module Stop Bit" in PMC to shut down LCDC clock supply to save power consumption after disable LCDC. Please refer to PMC for detailed information.

11.12.4 Resetting the Controller

At reset, the LCD Controller is disabled. All LCD Controller Registers are reset to the conditions shown in the register descriptions.

11.12.5 Frame Buffer & Palette Buffer

The starting address of frame buffer stored in external memory must be aligned to 4, 8 or 16 words boundary according to register LCDCTRL.BST. The length of buffer must be multiple of word (32-bit).

If LCDCTRL .BST = 0, align frame and palette buffer to 16 word boundary
If LCDCTRL .BST = 1, align frame and palette buffer to 8 word boundary
If LCDCTRL .BST = 2, align frame and palette buffer to 4 word boundary

One frame buffer contains encoded pixel data of multiple of screen lines; each line of encoded pixel data must be aligned to word boundary. If the length of a line is not the multiple of word, extra bits must be applied to reach a word boundary. It is suggested that the extra bits to be set zero.

12 Smart LCD Controller

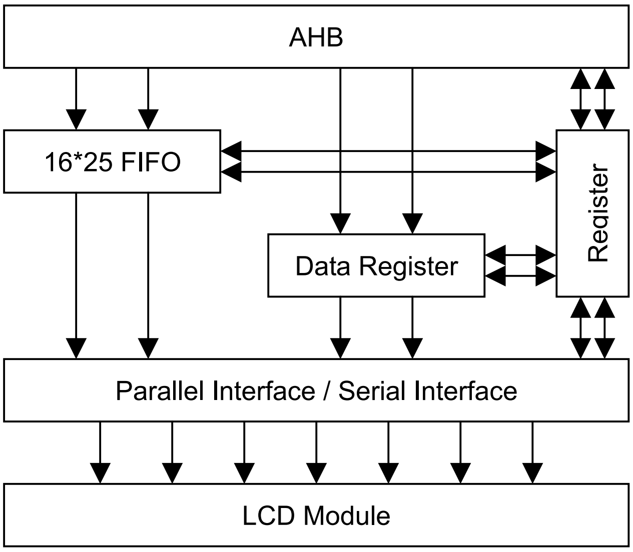
12.1 Overview

The Smart LCD Controller transfers data from the display buffer to the LCD Module. It supports DMA operation and register operation.

Features:

- Supports a large variety of LCD Module from different vendors.
- Supports parallel and serial interfaces.
- Supports different size of display panel.
- Supports different width of pixel data.
- Supports DMA operation and register operation.
- Supports Write Operation. Read Operation is not supported.

12.2 Structure


AHB
16*25 FIFO
Register
Data Register
Parallel Interface / Serial Interface
LCD Module

12.3 Pin Description

Table 12-1 SLCD Pins Description

Name	I/O	Description	Interface
SLCD_RS	O	Command/Data Select Signal. The polarity of the signal can be programmable.	Serial: RS Parallel: RS
SLCD_CS	O	Chip Select Signal. The polarity of the signal can be programmable.	Serial: CS Parallel: Sample Data with the edge of CS
SLCD_CLK	O	The clock of SLCD. The polarity of the clock can be programmable.	Serial or not used
SLCD_DAT[*1] [17:0]	O	The data of SLCD.	Serial: SLCD_DAT [15] Parallel: SLCD_DAT [17:0] SLCD_DAT [15:0] SLCD_DAT [7:0]

Note*[1]: SLCD_DAT [15] is also use as data pin for serial. The SLCD pins are shared with LCDC. You can see the set of register LCDCFG.LCDPIN in LCDC spec.

12.4 Register Description

In this section, we will describe the registers in Smart LCD controller. Following table lists all the registers definition. All register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
MCFG	SLCD Configure Register	RW	0x0000	0x130500A0	32
MCTRL	SLCD Control Register	RW	0x00	0x130500A4	8
MSTATE	SLCD Status Register	RW	0x00	0x130500A8	8
MDATA	SLCD Data Register	RW	0x00000000	0x130500AC	32
MFIFO	SLCD FIFO	RW	0x00000000	0x130500B0	32

12.4.1 SLCD Configure Register (MCFG)

The register MCFG is used to configure SLCD.

MCFG **0x130500A0**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIGGER		Reserved	DWIDTH			CWIDTH		Reserved			CSPLY	RSPLY	Reserved	CLKPLY	TYPE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:14	TRIGGER	FIFO trigger for DMA Operation. Trigger Length Selection **TRIGGER** – Trigger Length 00 – 4 word 01 – 8 word 10 – Reserved 11 – Reserved	RW
13	Reserved	These bits always read 0, and written are ignored.	R
12:10	DWIDTH[*1]	Data Width. **DWIDTH** – **Data Width** 000 – 18-bit once Parallel/Serial 001 – 16-bit once Parallel/Serial 010 – 8-bit third time Parallel 011 – 8-bit twice Parallel 100 – 8-bit once Parallel/Serial 111 – 9-bit twice Parallel 101~110 – Reserved	RW
9:7	CWIDTH[*1]	Command Width. **CWIDTH** – **Command Width** 00 – 16-bit once 01 – 8-bit once 10 – 18-bit once 11 – Reserved	RW
7:5	Reserved	These bits always read 0, and written are ignored.	R
4	CSPLY	CS Polarity. (CS initial level will be different from CS Polarity) 0: Active Level is Low 1: Active Level is High	RW

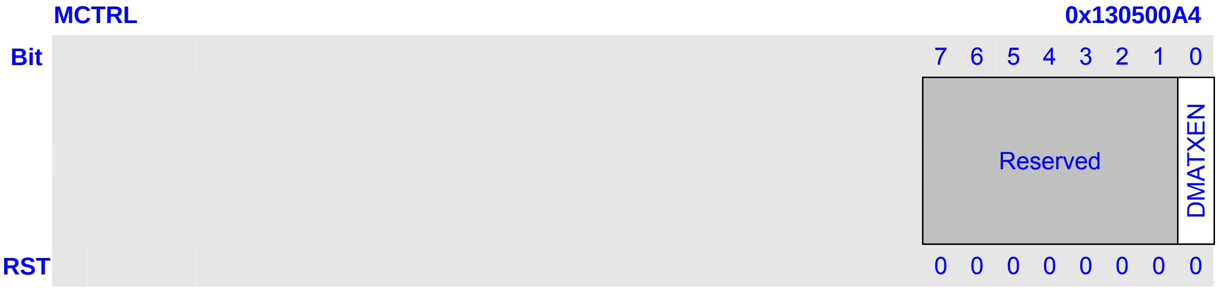
3	RSPLY	RS Polarity. 0: Command RS = 0, Data RS = 1 1: Command RS = 1, Data RS = 0	RW
2	Reserved	These bits always read 0, and written are ignored.	R
1	CLKPLY	LCD_CLK Polarity. 0: Active edge is Falling 1: Active edge is Rising	RW
0	TTYPE	Transfer Type: 0: Parallel 1: Serial	RW

Note*1: The set of DWIDTH and CWIDTH should keep to the rules as follows:

Interface Mode	Command Width	Data Width	Color
Parallel	18-bit	18-bit once	
	16-bit	16-bit once	
		9-bit twice	
	8-bit	8-bit once	
		8-bit twice	
		8-bit third times	
Serial	18-bit	18-bit once	
	16-bit	16-bit once	
	8-bit	8-bit once	
		8-bit twice	
		8-bit third times	

12.4.2 SLCD Control Register (MCTRL)

MCTRL is SLCD Control Register.



Bits	Name	Description	RW
7:1	Reserved	These bits always read 0, and written are ignored.	R

0	DMATXEN	SLCD DMA Transfer Enable. This bit is only used for DMA automatic transfer. (1) This bit starts the automatic transfer of image data from system memory to LCDM. (2) When DMAC finishes transferring the data, and the MSTATE.BUSY bit is 0, you can clear DMATXEN bit to stop DMA mode.	RW

12.4.3 SLCD Status Register (MSTATE)

The register of MSTATE is SLCD status register.

MSTATE **0x130500A8**

Bit	7	6	5	4	3	2	1	0
	Reserved							BUSY
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:1	Reserved	These bits always read 0, and written are ignored.	R
0	BUSY	Transfer is working or not. This bit will be set to 1 when transfer is working. It will be cleared by hardware when transfer is finished. 0: not busy 1: busy	RW

12.4.4 SLCD Data Register (MDATA)

The register MDATA is used to send command or data to LCM. When RS=0, the low 24-bit is used as command. When RS=1, the low 24-bit is used as data.

MDATA **0x130500AC**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RS	Reserved							DATA / CMD																							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	RS	The RS bit of data register is used to decide the meanings of the low 24-bit. 0: data 1: command	RW
30:24	Reserved	These bits always read 0, and written are ignored.	R
23:0	DATA/CMD	Data or Command Register.	RW

12.4.5 SLCD FIFO (MFIFO)

The FIFO is used to send command or data to LCM. When RS=0, the low 24-bit is used as command. When RS=1, the low 24-bit is used as data.

MFIFO **0x130500B0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RS	Reserved							DATA / CMD																							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	RS	The RS bit of FIFO is used to decide the meanings of the low 24-bit. 0: data 1: command	RW
30:24	Reserved	These bits always read 0, and written are ignored.	R
23:0	DATA/CMD	Data or Command Register.	RW

12.5 System Memory Format

The format of Command and Data in system memory is as follows:

Command	RS [31] = 1	XXX [30:n+1]	Command [n:0]
Data	RS [31] = 0	XXX [30:24]	Data [23:0]
	RS [31] = 0	XXX [30:16]	Data [15:0]
	RS [31] = 0	XXX [30:8]	Data [7:0]

12.5.1 Data format

(1) 24-bit color

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	X	X	X	X	X	X	X	R7	R6	R5	R4	R3	R2	R1	R0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

(2) 18-bit color

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	X	X	X	X	X	X	X	R5	R4	R3	R2	R1	R0	X	X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G5	G4	G3	G2	G1	G0	X	X	B5	B4	B3	B2	B1	B0	X	X

(3) 16-bit color

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

(4) 8-bit color

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	C7	C6	C5	C4	C3	C2	C1	C0

12.5.2 Command Format

(1) 18-bit command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	C17	C16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

(2) 16-bit command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

(3) 8-bit command once

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	X	X	X	X	X	X	C7	C6	C5	C4	C3	C2	C1	C0

(4) 8-bit command twice (Command = command part + data part, twice transfer)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	X	X	X	X	X	X	C7	C6	C5	C4	C3	C2	C1	C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	X	X	X	X	X	X	D7	D6	D5	D4	D3	D2	D1	D0

Note: The command is made up of command part and data part, but need twice transfer, and the first transfer is command part and the second transfer is data part. You need to divide the command into two parts by software.)

12.6 Transfer Mode

Two transfer modes can be used: DMA Transfer Mode and Data Register Transfer Mode.

12.6.1 DMA Transfer Mode

Command and data can be recognized by RS bit coming from memory. The format of DMA transfer can be as follows:

(1) Command and Data

C	D	D	C	C	D	D	D	C	C	C	D	D	D	D

(2) Only Data

D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

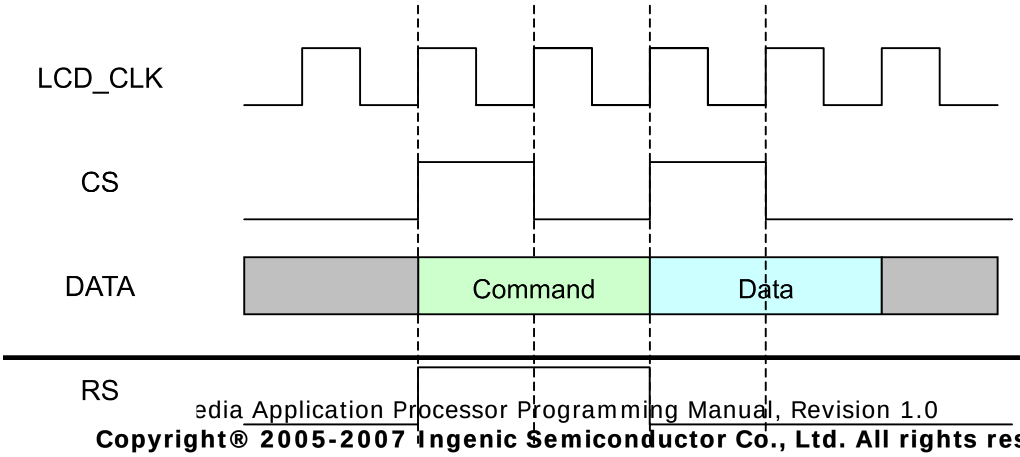
12.6.2 Register Transfer Mode

Each time you can write a command or a data to the register, then it will transfer the RS signal and data or command to LCM. Command and data can be recognized by RS bit coming from data register. The format of data register transfer can be as follows:

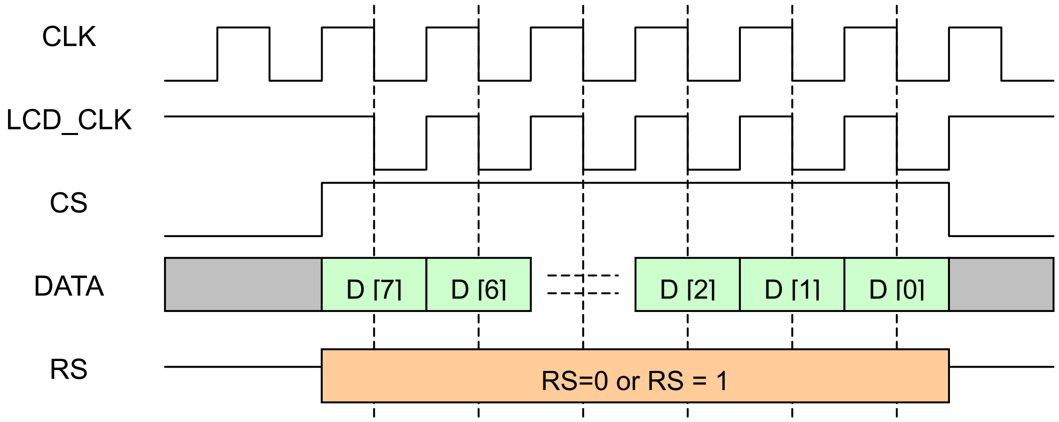
RS	Data Or Command

12.7 Timing

12.7.1 Parallel Timing



12.7.2 Serial Timing



12.8 Operation Guide

12.8.1 DMA Operation

(1) Start DMA transfer

(1) Initial DMAC.

(2) Set MCFG to configure SLCD.

(3) Before starting DMA, Wait for MSTATE.BUSY == 0.

(4) Set MCTRL.DMATXEN to 1 to start DMA transfer. (If you don't want to stop DMA transfer, you need not to check MSTATE.BUSY.)

(2) Stop DMA transfer

(1) Check the status of DMAC, and stop DMAC.

(2) Wait MSTATE.BUSY == 0

(3) Set MCTRL.DMATXEN to 0 to stop DMA transfer.

(3) Restart DMA transfer

When MCTRL.DMATXEN is set to 0, and then you want to restart DMA transfer at once, you should ensure that MCTRL.DMATXEN must keep low level at least three cycles of PIXCLK.

12.8.2 Register Operation

(1) Set MCFG to configure SLCD.

(2) Wait for MSTATE.BUSY == 0.

(3) Set MDATA register.

(4) Wait for MSTATE.BUSY == 0.

(5) Set MDATA register.

(6) Wait for MSTATE.BUSY == 0.

(7) … …

13 AC97/I2S Controller

13.1 Overview

This chapter describes the AIC (AC'97 and I^2S Controller) included in this processor.

The AIC supports the Audio Codec '97 Component Specification 2.3 for AC-link format and I2S or IIS (for inter-IC sound), a protocol defined by Philips Semiconductor. Both normal I2S and the MSB-justified I2S formats are supported by AIC.

AIC consists of buffers, status registers, control registers, serializers, and counters for transferring digitized audio between the processor's processor system memory and an internal I2S CODEC, an external AC97 or I2S CODEC. AIC can record digitized audio by storing the samples in system memory. For playback of digitized audio or production of synthesized audio, the AIC retrieves digitized audio samples from system memory and sends them to a CODEC through the serial connection with AC-link or I2S formats. The internal or external digital-to-analog converter in the CODEC then converts the audio samples into an analog audio waveform. The audio sample data can be stored to and retrieved from system memory either by the DMA controller or by programmed I/O.

The AC-link is a synchronous, fixed-rate serial bus interface for transferring CODEC register control and status information in addition to digital audio. Where both normal I2S and MSB-justified-I2S work with a variety of clock rates, which can be obtained either by dividing the PLL clock by two programmable dividers or from an external clock source.

For I2S systems that support the L3 control bus protocol, additional pins are required to control the external CODEC. CODECs that use an L3 control bus require 3 signals: L3_CLK, L3_DATA, and L3_MODE for writing bytes into the L3 bus register. The AIC supports the L3 bus protocol via software control of the general-purpose I/O (GPIO) pins. The AIC does not provide hardware control for the L3 bus protocol.

To control the internal CODEC, internal CODEC spec can be referenced.

This chapter describes the programming model for the AIC. The information in this chapter requires an understanding of the AC'97 specification, Revision 2.3.

13.1.1 Block Diagram

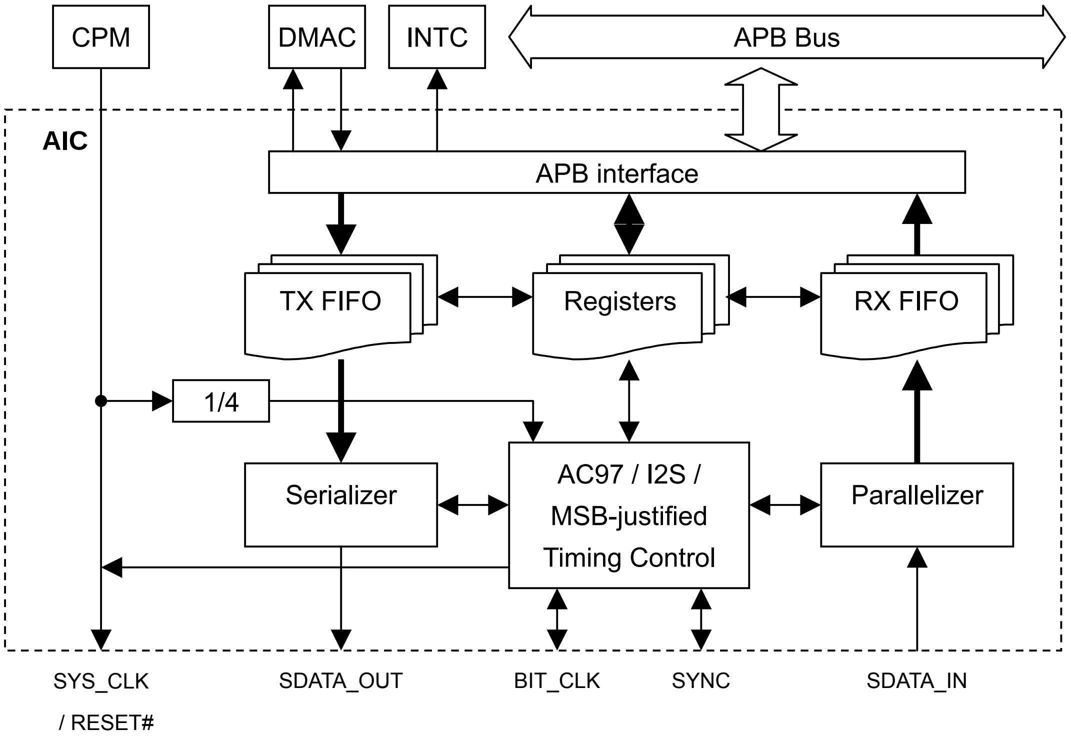

CPM
DMAC
INTC
APB Bus
AIC
APB interface
TX FIFO
Registers
RX FIFO
1/4
Serializer
AC97 / I2S / MSB-justified Timing Control
Parallelizer
SYS_CLK / RESET#
SDATA_OUT
BIT_CLK
SYNC
SDATA_IN

Figure 13-1 AIC Block Diagram

13.1.2 Features

AIC support following AC97/I2S features:

- 8, 16, 18, 20 and 24 bit audio sample data sizes supported
- DMA transfer mode supported
- Stop serial clock supported
- Programmable Interrupt function supported
- Support mono PCM data to stereo PCM data expansion on audio play back
- Support endian switch on 16-bits audio samples play back
- Support variable sample rate in AC-link format
- Multiple channel output and double rated supported for AC-link format
- Power Down Mode and two Wake-Up modes Supported for AC-link format
- Internal programmable or external serial clock and optional system clock supported for I2S or MSB-Justified format
- Internal I2S CODEC supported
- Two FIFOs for transmit and receive respectively with 32 samples capacity in every direction.

13.1.3 Interface Diagram

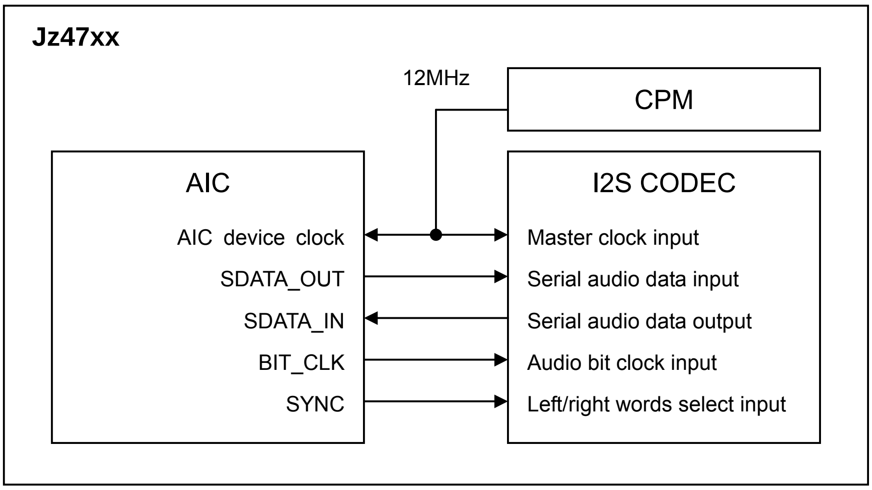

Jz47xx
12MHz
CPM
AIC
AIC device clock
SDATA_OUT
SDATA_IN
BIT_CLK
SYNC
I2S CODEC
Master clock input
Serial audio data input
Serial audio data output
Audio bit clock input
Left/right words select input

Figure 13-2 Interface to the Internal I2S CODEC Diagram

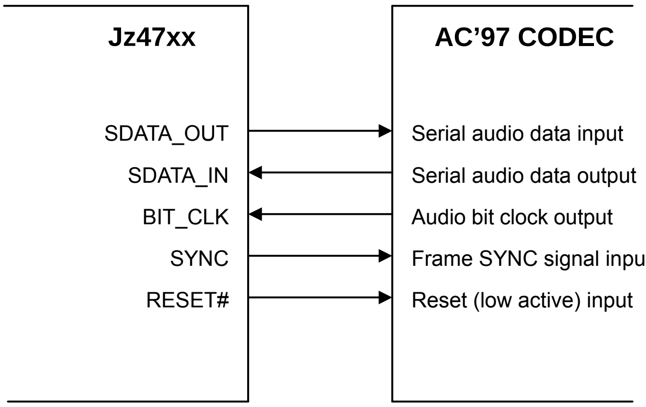

Jz47xx
SDATA_OUT
SDATA_IN
BIT_CLK
SYNC
RESET#
AC'97 CODEC
Serial audio data input
Serial audio data output
Audio bit clock output
Frame SYNC signal input
Reset (low active) input

Figure 13-3 Interface to an External AC'97 CODEC Diagram

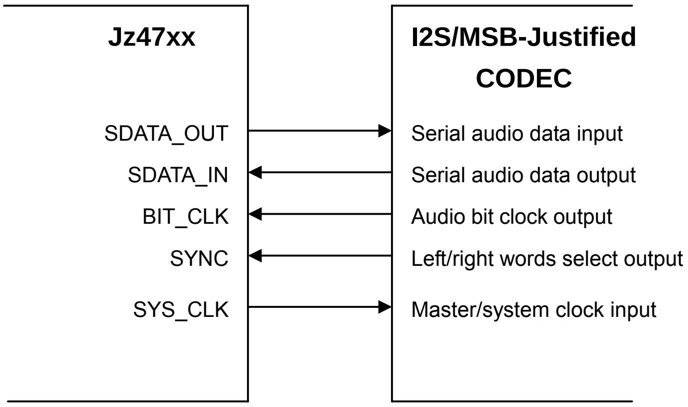

Jz47xx
SDATA_OUT
SDATA_IN
BIT_CLK
SYNC
SYS_CLK
I2S/MSB-Justified CODEC
Serial audio data input
Serial audio data output
Audio bit clock output
Left/right words select output
Master/system clock input

Figure 13-4 Interface to an External Master Mode I2S/MSB-Justified CODEC Diagram

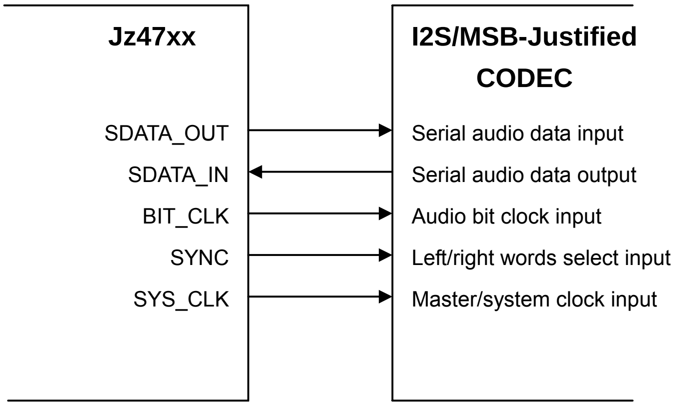

Jz47xx
SDATA_OUT
SDATA_IN
BIT_CLK
SYNC
SYS_CLK
I2S/MSB-Justified CODEC
Serial audio data input
Serial audio data output
Audio bit clock input
Left/right words select input
Master/system clock input

Figure 13-5 Interface to an External Slave Mode I2S/MSB-Justified CODEC Diagram

13.1.4 Signal Descriptions

There are all 5 pins used to connect between AIC and an external audio CODEC device. If an internal CODEC is used, these pins are not needed. They are listed and described in Table 13-1.

Table 13-1 AIC Pins Description

Name	I/O	Description
RESET# SYS_CLK	O	RESET#: AC-link format, active-low CODEC reset. SYS_CLK: I2S/MSB-Justified formats, supply system clock to CODEC
BIT_CLK	I I/O	12.288 MHz bit-rate clock input for AC-link, and sample rate dependent bit-rate clock input/output for I2S/MSB-Jistified.
SYNC	O	48-kHz frame indicator and synchronizer for AC-link format
	I/O	Indicates the left- or right-channel for I2S/MSB-Justified format
SDATA_OUT	O	Serial audio output data to CODEC
SDATA_IN	I	Serial audio input data from CODEC

13.1.5 RESET# / SYS_CLK Pin

RESET# is AC97 active-low CODEC reset, which outputs to CODEC. The CODEC's registers are reset when this RESET# is asserted. This pin is useful only in AC-link format. If AIC is disabled, it retains the high.

SYS_CLK outputs the system clock to CODEC. This pin is useful only in I2S/MSB-justified format. It generates a frequency between approximately 2.048 MHz and 24.576 MHz by dividing down the PLL clock with a programmable divisor. This frequency can be 256, 384, 512 and etc. times of the audio sampling frequency. Or it can be set to a wanted frequency. If AIC is disabled, it retains the high.

13.1.6 BIT_CLK Pin

BIT_CLK is the serial data bit rate clock, at which AC97/I2S data moves between the CODEC and the processor. One bit of the serial data is transmitted or received each BIT_CLK period. It is fixed to 12.288 MHz in AC-link format, which inputs from the CODEC. In I2S and MSB-justified format it inputs from the CODEC in slave mode and outputs to CODEC in master mode. In the master mode, the clock is generated internally that is 64 times the sampling frequency. Table 13-7 lists the available sampling frequencies based on an internal clock source. If AIC is disabled, AICFR.AUSEL and AICFR.BCKD determine the direction. And it retains the low if it is output and the state is undefined if it is input.

13.1.7 SYNC Pin

In AC-link format, SYNC provides frame synchronization, fixed to 48kHz, by specifying beginning of an audio sample frame and outputs to CODEC. In I2S/MSB-Justified formats, SYNC is used to indicate left- or right-channel sample data and toggled in sample rate frequency. It outputs to CODEC in master mode and inputs from CODEC in slave mode. If AIC is disabled, AICFR.AUSEL and AICFR.BCKD determine the direction. And it retains the low if it is output and the state is undefined if it is input.

13.1.8 SDATA_OUT Pin

SDATA_OUT is AIC output data pin, which outputs serial audio data or data of AC97 CODEC register control to an external audio CODEC device. If AIC is disabled, it retains the low.

13.1.9 SDATA_IN Pin

SDATA_IN is AIC inputs data pin, which inputs serial audio data or data of AC97 CODEC register status from an external audio CODEC device. If AIC is disabled, its state is undefined.

13.2 Register Descriptions

AIC software interface includes 13 registers and 1 FIFO data port. They are mapped in IO memory address space so that program can access them to control the operation of AIC and the outside CODEC.

Table 13-2 AIC Registers Description

Name	Description	RW	Reset value	Address	Size
AICFR	AIC Configuration Register	RW	0x00007800	0x10020000	32
AICCR	AIC Common Control Register	RW	0x00000000	0x10020004	32
ACCR1	AIC AC-link Control Register 1	RW	0x00000000	0x10020008	32
ACCR2	AIC AC-link Control Register 2	RW	0x00000000	0x1002000C	32
I2SCR	AIC I2S/MSB-justified Control Register	RW	0x00000000	0x10020010	32
AICSR	AIC FIFO Status Register	RW	0x00000008	0x10020014	32
ACSR	AIC AC-link Status Register	RW	0x00000000	0x10020018	32
I2SSR	AIC I2S/MSB-justified Status Register	RW	0x00000000	0x1002001C	32
ACCAR	AIC AC97 CODEC Command Address Register	RW	0x00000000	0x10020020	32
ACCDR	AIC AC97 CODEC Command Data Register	RW	0x00000000	0x10020024	32
ACSAR	AIC AC97 CODEC Status Address Register	R	0x00000000	0x10020028	32
ACSDR	AIC AC97 CODEC Status Data Register	R	0x00000000	0x1002002C	32
I2SDIV	AIC I2S/MSB-justified Clock Divider Register	RW	0x00000003	0x10020030	32
AICDR	AIC FIFO Data Port Register	RW	0x????????	0x10020034	32

- AICFR is used to control FIFO threshold, AC-link or I2S/MSB-justified selection, AIC reset, master/slave selection, and AIC enable.
- AICCR is used to control DMA mode, FIFO flush, interrupt enable, internal loop-back, play back and recording enable. It also controls sample size and signed/unsigned data transfer.
- ACCR1 is used to reflect/control valid incoming/outgoing slots of AC97.
- ACCR2 is used to control interrupt enable, output/input sample size, and alternative control of RESET#, SYNC and SDATA_OUT pins in AC-link.
- I2SCR is used to control BIT_CLK stop, audio sample size, I2S or MSB-justified selection in I2S/MSB-justified.
- AICSR is used to reflect FIFOs status
- ACSR is used to reflect the status of the connected external CODEC in AC-link.

- I2SSR is used to reflect AIC status in I2S/MSB-justified.
- ACCAR and ACCDR are used to hold address and data for AC-link CODEC register read/write.
- ACSAR and ACSDR are used to receive AC-link CODEC registers address and data
- I2SDIV is used to set clock divider for BIT_CLK generating in I2S/MSB-justified format.
- AICDR is act as data input/output port to/from transmit/receive FIFO when write/read.

13.2.1 AIC Configuration Register (AICFR)

AICFR contains bits to control FIFO threshold, AC-link or I2S/MSB-justified selection, AIC reset, master/slave selection, and AIC enable.

AICFR **0x10020000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																RFTH				TFTH				Reserved	LSMP	ICDC	AUSEL	RST	BCKD	SYNCD	ENB
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0	R
15:12	RFTH	Receive FIFO threshold for interrupt or DMA request. The RFTH valid value is 0 ~ 15. This value represents a threshold value of (RFTH + 1) * 2. When the sample number in receive FIFO, indicated by AICSR.RFL, is great than or equal to the threshold value, AICSR.RFS is set. Larger RFTH value provides lower DMA/interrupt request frequency but have more risk to involve receive FIFO overflow. The optimum value is system dependent.	RW
11:8	TFTH	Transmit FIFO threshold for interrupt or DMA request. The TFTH valid value 0 ~ 15. This value represents a threshold value of TFTH * 2. When the sample number in transmit FIFO, indicated by AICSR.TFL, is less than or equal to the threshold value, AICSR.TFS is set. Smaller TFTH value provides lower DMA/interrupt request frequency but have more risk to involve transmit FIFO underflow. The optimum value is system dependent.	RW
7	Reserved	Writes to these bits have no effect and always read as 0	R
6	LSMP	Select between play last sample or play ZERO sample in TX FIFO underflow. ZERO sample means sample value is zero. This bit is better be changed while audio replay is stopped. **LSMP** / **CODEC used** 0: Play ZERO sample when TX FIFO underflow 1: Play last sample when TX FIFO underflow	RW
5	ICDC	Internal CODEC used. Select between internal or external CODEC. **ICDC** / **CODEC used** 0: External CODEC 1: Internal CODEC	RW
4	AUSEL	Audio Unit Select. Select between AC-link and I2S/MSB-justified. Change this bit in case of BIT_CLK is stopped (I2SCR.STPBK = 1)	RW

<table>
<tr><td></td><td></td><td><table><tr><th>AUSEL</th><th>Selected</th></tr><tr><td>0</td><td>Select AC-link format</td></tr><tr><td>1</td><td>Select I2S/MSB-justified format</td></tr></table></td><td></td></tr>
<tr><td>3</td><td>RST</td><td>Reset AIC. Write 1 to this bit reset AIC registers and FIFOs except AICFR and I2SDIV register. Writing 0 to this bit has no effect and this bit is always reading 0.</td><td>W</td></tr>
<tr><td>2</td><td>BCKD</td><td>BIT_CLK Direction. This bit specifies input/output direction of BIT_CLK. It is only valid in I2S/MSB-justified format. When AC-link format is selected, BIT_CLK is always input and this bit is ignored. Change this bit in case of BIT_CLK is stopped (I2SCR.STPBK = 1)<table><tr><th>BCKD</th><th>BIT_CLK Direction</th></tr><tr><td>0</td><td>BIT_CLK is input from an external source.</td></tr><tr><td>1</td><td>BIT_CLK is generated internally and driven out to the CODEC.</td></tr></table></td><td>RW</td></tr>
<tr><td>1</td><td>SYNCD</td><td>SYNC Direction. This bit specifies input/output direction of SYNC in I2S/MSB-justified format. When AC-link format is selected, SYNC is always output and this bit is ignored. Change this bit in case of BIT_CLK is stopped (I2SCR.STPBK = 1)<table><tr><th>SYNCD</th><th>SYNC Direction</th></tr><tr><td>0</td><td>SYNC is input from an external source.</td></tr><tr><td>1</td><td>SYNC is generated internally and driven out to the CODEC.</td></tr></table></td><td>RW</td></tr>
<tr><td>0</td><td>ENB</td><td>Enable AIC function. This bit is used to enable or disable the AIC function.<table><tr><th>ENB</th><th>Description</th></tr><tr><td>0</td><td>Disable AIC Controller</td></tr><tr><td>1</td><td>Enable AIC Controller</td></tr></table></td><td>RW</td></tr>
</table>

The BCKD bit (bit 2) and SYNCD bit (bit 1) configure the mode of I2S/MSB-justified interface. This is compliant with I2S specification.

BCKD	SYNCD	Description
0 (input)	0 (input)	AIC roles the slave of I2S/MSB-justified interface.
	1 (output)	AIC roles the master with external serial clock source of I2S/MSB-justified interface.
1 (output)	0 (input)	Reserved
	1 (output)	AIC roles the master of I2S/MSB-justified interface

13.2.2 AIC Common Control Register (AICCR)

AICCR contains bits to control DMA mode, FIFO flush, interrupt enable, internal loop-back, play back and recording enable. It also controls sample size and signed/unsigned data transfer.

AICCR **0x10020004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										OSS			ISS			RDMS	TDMS	Reserved	Reserved	M2S	ENDSW	ASVTSU	FLUSH	Reserved	EROR	ETUR	ERFS	ETFS	ENLBF	ERPL	EREC
RST	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:22	Reserved	Writes to these bits have no effect and always read as 0	R
21:19	OSS	Output Sample Size. These bits reflect output sample data size from memory or register. The data sizes supported are: 8, 16, 18, 20 and 24 bits. The sample data is LSB-justified in memory/register.	RW
18:16	ISS	Input Sample Size. These bits reflect input sample data size to memory or register. The data sizes supported are: 8, 16, 18, 20 and 24 bits. The sample data is LSB-justified in memory/register.	RW
15	RDMS	Receive DMA enable. This bit is used to enable or disable the DMA during receiving audio data.	RW
14	TDMS	Transmit DMA enable. This bit is used to enable or disable the DMA	RW

OSS	Sample Size
0x0	8 bit
0x1	16 bit
0x2	18 bit
0x3	20 bit
0x4	24 bit
0x5~0x7	Reserved

ISS	Sample Size
0x0	8 bit
0x1	16 bit
0x2	18 bit
0x3	20 bit
0x4	24 bit
0x5~0x7	Reserved

RDMS	Receive DMA
0	Disabled
1	Enabled

<table>
<tr><td></td><td></td><td>during transmit audio data.<table><tr><th>TDMS</th><th>Transmit DMA</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table></td><td></td></tr>
<tr><td>13:12</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>11</td><td>M2S</td><td>Mono To Stereo. This bit control whether to do mono to stereo sample expansion in play back. When this bit is set, every outgoing sample data in the steam plays in both left and right channels. This bit should only be set in 2 channels configuration. It takes effective immediately when the bit is changed. Change this before replay started<table><tr><th>M2S</th><th>Description</th></tr><tr><td>0</td><td>No mono to stereo expansion</td></tr><tr><td>1</td><td>Do mono to stereo expansion</td></tr></table></td><td>RW</td></tr>
<tr><td>10</td><td>ENDSW</td><td>Endian Switch. This bit control endian change on outgoing 16-bits size audio sample by swapping high and low bytes in the sample data.<table><tr><th>ENDSW</th><th>Description</th></tr><tr><td>0</td><td>No change on outgoing sample data</td></tr><tr><td>1</td><td>Swap high and low byte for outgoing 16-bits size sample data</td></tr></table></td><td>RW</td></tr>
<tr><td>9</td><td>ASVTSU</td><td>Audio Sample Value Transfer between Signed and Unsigned data format. This bit is used to control the signed ←→unsigned data transfer. If it is 1, the incoming and outgoing audio sample data will be transferred by toggle its most significant bit.<table><tr><th>ASVTSU</th><th>Description</th></tr><tr><td>0</td><td>No audio sample value signed ←→unsigned transfer</td></tr><tr><td>1</td><td>Do audio sample value signed ←→unsigned transfer</td></tr></table></td><td>RW</td></tr>
<tr><td>8</td><td>FLUSH</td><td>FIFO Flush. Write 1 to this bit flush transmit/receive FIFOs to empty. Writing 0 to this bit has no effect and this bit is always reading 0.</td><td>W</td></tr>
<tr><td>7</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>6</td><td>EROR</td><td>Enable ROR Interrupt. This bit is used to control the ROR interrupt enable or disable.<table><tr><th>EROR</th><th>ROR Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table></td><td>RW</td></tr>
<tr><td>5</td><td>ETUR</td><td>Enable TUR Interrupt. This bit is used to control the TUR interrupt enable or disable.<table><tr><th>ETUR</th><th>TUR Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table></td><td>RW</td></tr>
<tr><td>4</td><td>ERFS</td><td>Enable RFS Interrupt. This bit is used to control the RFS interrupt enable or disable.<table><tr><th>ERFS</th><th>RFS Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr></table></td><td>RW</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table></td><td></td></tr>
<tr><td>3</td><td>ETFS</td><td>Enable TFS Interrupt. This bit is used to control the TFS interrupt enable or disable.<table><tr><th>ETFS</th><th>TFS Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table></td><td>RW</td></tr>
<tr><td>2</td><td>ENLBF</td><td>Enable AIC Loop Back Function. This bit is used to enable or disable the internal loop back function of AIC, which is used for test only. When the AIC loop back function is enabled, normal audio replay/record functions are disabled.<table><tr><th>ENLBF</th><th>Description</th></tr><tr><td>0</td><td>AIC Loop Back Function is Disabled</td></tr><tr><td>1</td><td>AIC Loop Back Function is Enabled</td></tr></table></td><td>RW</td></tr>
<tr><td>1</td><td>ERPL</td><td>Enable Playing Back function. This bit is used to disable or enable the audio sample data transmitting.<table><tr><th>ERPL</th><th>Description</th></tr><tr><td>0</td><td>AIC Playing Back Function is Disabled</td></tr><tr><td>1</td><td>AIC Playing Back Function is Enabled</td></tr></table></td><td>RW</td></tr>
<tr><td>0</td><td>EREC</td><td>Enable Recording Function. This bit is used to disable or enable the audio sample data receiving.<table><tr><th>EREC</th><th>Description</th></tr><tr><td>0</td><td>AIC Recording Function is Disabled</td></tr><tr><td>1</td><td>AIC Recording Function is Enabled</td></tr></table></td><td>RW</td></tr>
</table>

13.2.3 AIC AC-link Control Register 1 (ACCR1)

ACCR1 contains bits to reflect/control valid incoming/outgoing slots of AC97. It is used only in AC-link format.

ACCR1 **0x10020008**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved						RS										Reserved						XS									
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:26</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>25:16</td><td>RS</td><td>Receive Valid Slots. These bits are used to indicate which incoming slots are valid. Slot 3 is mapped to bit 16 or RS[0], slot 4 to bit 17 or RS[1] and so on. When write to this field, a bit 1 means we expect a PCM data in the corresponding slot, a bit 0 means the corresponding slot PCM data will be discarded. When read from this field, a bit 1 means we receive an expected valid PCM data in the corresponding slot. This field should be written before record started.
<table>
<tr><th>RS[n] Value</th><th>Description</th></tr>
<tr><td>0</td><td>Slot n+3 is invalid.</td></tr>
<tr><td>1</td><td>Slot n+3 has valid PCM data.</td></tr>
</table></td><td>RW</td></tr>
<tr><td>15:10</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>9:0</td><td>XS</td><td>Transmit Valid Slots. These bits making up slots map to the valid bits in the AC'97 tag (slot 0 on SDATA_OUT) and indicate which outgoing slots have valid PCM data. Bit 0 or XS[0] maps to slot 3, bit 1 or XS[1] to slot 4 and so on. Setting the corresponding bit indicates to AIC to take an audio sample from transmit FIFO to fill the respective slot. And it indicates to the CODEC that valid PCM data will be in the respective slot. The number of valid bits will designate how many words will be pulled out of the FIFO per audio frame. This field should be written before record and replay started.
<table>
<tr><th>XS[n] Value</th><th>Description</th></tr>
<tr><td>0</td><td>Slot n+3 is invalid.</td></tr>
<tr><td>1</td><td>Slot n+3 has valid PCM data.</td></tr>
</table></td><td>RW</td></tr>
</table>

13.2.4 AIC AC-link Control Register 2 (ACCR2)

ACCR2 contains bits to control interrupt enable, output/input sample size, and alternative control of RESET#, SYNC and SDATA_OUT pins in AC-link. It is valid only in AC-link format.

ACCR2 **0x1002000C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved													ERSTO	ESADR	ECADT	Reserved												SO	SR	SS	SA
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:19</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>18</td><td>ERSTO</td><td>Enable RSTO Interrupt. This bit is used to control the RSTO interrupt enable or disable.
<table>
<tr><th>ERSTO</th><th>RSTO Interrupt</th></tr>
<tr><td>0</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
</table></td><td>RW</td></tr>
<tr><td>17</td><td>ESADR</td><td>Enable SADR Interrupt. This bit is used to control the SADR interrupt enable or disable.
<table>
<tr><th>ESADR</th><th>SADR Interrupt</th></tr>
<tr><td>0</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
</table></td><td>RW</td></tr>
<tr><td>16</td><td>ECADT</td><td>Enable CADT Interrupt. This bit is used to control the CADT interrupt enable or disable.
<table>
<tr><th>ECADT</th><th>CADT Interrupt</th></tr>
<tr><td>0</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
</table></td><td>RW</td></tr>
<tr><td>15:4</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>3</td><td>SO</td><td>SDATA_OUT output value. When SA is 1, this bit controls SDATA_OUT pin voltage level, 0 for low, 1 for high; otherwise, it is ignored.</td><td>RW</td></tr>
<tr><td>2</td><td>SR</td><td>RESET# pin level. When AC-link is selected, this bit is used to drive the RESET# pin.
<table>
<tr><th>SR</th><th>RESET# Pin Voltage Level</th></tr>
<tr><td>0</td><td>High</td></tr>
<tr><td>1</td><td>Low</td></tr>
</table></td><td>RW</td></tr>
<tr><td>1</td><td>SS</td><td>SYNC value. When this bit is read, it returns the actual value of SYNC. When SA is 1, write value controls SYNC pin value. When SA is 0, write to it is ignored.</td><td>RW</td></tr>
<tr><td>0</td><td>SA</td><td>SYNC and SDATA_OUT Alternation. This bit is used to determine the</td><td>RW</td></tr>
</table>

<table>
<tr><td></td><td></td><td>driven signal of SYNC and SDATA_OUT. When SA is 0, SYNC and SDATA_OUT being driven AIC function logic; otherwise, SYNC is controlled by the SS and SDATA_OUT is controlled by the SO. The true table of SYNC is described in following.
<table>
<tr><th>SA</th><th>SS</th><th>Description</th></tr>
<tr><td rowspan="4">0</td><td rowspan="2">0</td><td>When read, indicated SYNC is 0</td></tr>
<tr><td>When write, not effect</td></tr>
<tr><td rowspan="2">1</td><td>When read, indicated SYNC is 1</td></tr>
<tr><td>When write, not effect</td></tr>
<tr><td rowspan="4">1</td><td rowspan="2">0</td><td>When read, indicated SYNC is 0</td></tr>
<tr><td>When write, SYNC is driven to 0</td></tr>
<tr><td rowspan="2">1</td><td>When read, indicated SYNC is 1</td></tr>
<tr><td>When write, SYNC is driven to 1</td></tr>
</table>
</td><td></td></tr>
</table>

13.2.5 AIC I2S/MSB-justified Control Register (I2SCR)

I2SCR contains bits to control BIT_CLK stop, audio sample size, I2S or MSB-justified selection in I2S/MSB-justified. It is valid only in I2S/MSB-justified format.

I2SCR **0x10020010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																			STPBK	Reserved							ESCLK	Reserved			AMSL
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:13</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>12</td><td>STPBK</td><td>Stop BIT_CLK. It is used to stop the BIT_CLK in I2S/MSB-justified format. When AC-link is selected, all of its operations are ignored.
<table>
<tr><th>STPBK</th><th>Description</th></tr>
<tr><td>0</td><td>BIT_CLK is not stopped</td></tr>
<tr><td>1</td><td>BIT_CLK is stopped</td></tr>
</table>
Please set this bit to 1 to stop BIT_CLK when change AICFR.AUSEL and AICFR.BCKD</td><td>RW</td></tr>
<tr><td>11:5</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>4</td><td>ESCLK</td><td>Enable SYSCLK output. When this bit is 1, the SYSCLK outputs to chip outside is enabled. Else, the clock is disabled.</td><td>RW</td></tr>
<tr><td>0</td><td>AMSL</td><td>Specify Alternate Mode (I2S or MSB-Justified) Operation.
<table>
<tr><th>AMSL</th><th>Description</th></tr>
<tr><td>0</td><td>Select I2S Operation Mode</td></tr>
<tr><td>1</td><td>Select MSB-Justified Operation Mode</td></tr>
</table></td><td>RW</td></tr>
</table>

13.2.6 AIC Controller FIFO Status Register (AICSR)

AICSR contains bits to reflect FIFOs status. Most of the bits are read-only except two, which can be written a 0.

AICSR **0x10020014**

<table>
<tr><th>Bit</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td></td><td colspan="2">Reserved</td><td colspan="6">RFL</td><td colspan="10">Reserved</td><td colspan="6">TFL</td><td>Reserved</td><td>ROR</td><td>TUR</td><td>RFS</td><td>TFS</td><td colspan="3">Reserved</td></tr>
<tr><td>RST</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:30</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>29:24</td><td>RFL</td><td>Receive FIFO Level. The bits indicate the amount of valid PCM data in Receive FIFO.
<table>
<tr><th>RFL Value</th><th>Description</th></tr>
<tr><td>0x00 ~ 0x20</td><td>RFL valid PCM data in receive FIFO</td></tr>
<tr><td>0x21 ~ 0x3F</td><td>Reserved</td></tr>
</table></td><td>R</td></tr>
<tr><td>23:14</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>13:8</td><td>TFL</td><td>Transmit FIFO Level. The bits indicate the amount of valid PCM data in Transmit FIFO.
<table>
<tr><th>TFL Value</th><th>Description</th></tr>
<tr><td>0x00 ~ 0x20</td><td>TFL valid PCM data in transmit FIFO</td></tr>
<tr><td>0x21 ~ 0x3F</td><td>Reserved</td></tr>
</table></td><td>R</td></tr>
<tr><td>7</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>6</td><td>ROR</td><td>Receive FIFO Over Run. This bit indicates that receive FIFO has or has not experienced an overrun.
<table>
<tr><th>ROR</th><th>Description</th></tr>
<tr><td rowspan="2">0</td><td>When read, indicates over-run has not been found</td></tr>
<tr><td>When write, clear itself</td></tr>
<tr><td rowspan="2">1</td><td>When read, indicates data has even been written to full receive FIFO</td></tr>
<tr><td>When write, not effects</td></tr>
</table></td><td>RW</td></tr>
<tr><td>5</td><td>TUR</td><td>Transmit FIFO Under Run. This bit indicates that transmit FIFO has or has not experienced an under-run.
<table>
<tr><th>TUR</th><th>Description</th></tr>
<tr><td rowspan="2">0</td><td>When read, indicates under-run has not been found</td></tr>
<tr><td>When write, clear itself</td></tr>
<tr><td>1</td><td>When read, indicates data has even been read from empty transmit FIFO</td></tr>
</table></td><td>RW</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td></td><td>When write, not effects</td></tr></table></td><td></td></tr>
<tr><td>4</td><td>RFS</td><td>Receive FIFO Service Request. This bit indicates that receive FIFO level is or not below RFL threshold which is controlled by AICFR.RFTH. When RFS is 1, it may trigger interrupt or DMA request depends on the interrupt enable and DMA setting.<table><tr><th>RFS</th><th>Description</th></tr><tr><td>0</td><td>Receive FIFO level below RFL threshold</td></tr><tr><td>1</td><td>Receive FIFO level at or above RFL threshold</td></tr></table></td><td>R</td></tr>
<tr><td>3</td><td>TFS</td><td>Transmit FIFO Service Request. This bit indicates that transmit FIFO level exceeds TFL threshold which is controlled by AICFR.TFTH When TFS is 1, it may trigger interrupt or DMA request depends on the interrupt enable and DMA setting.<table><tr><th>TFS</th><th>Description</th></tr><tr><td>0</td><td>Transmit FIFO level exceeds TFL threshold</td></tr><tr><td>1</td><td>Transmit FIFO level at or below TFL threshold</td></tr></table></td><td>R</td></tr>
<tr><td>2:0</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
</table>

13.2.7 AIC AC-link Status Register (ACSR)

ACSR contains bits to reflect the status of the connected external CODEC in AC-link format. Bits in this register are read-only in general, except some of them can be written a 0.

ACSR **0x10020018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										SLTERR	CRDY	CLPM	RSTO	SADR	CADT	Reserved															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:22</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>21</td><td>SLTERR</td><td>Hardware detects a Slot Error. This bit indicates an error in SLOTREQ bits on incoming data from external CODEC is detected. The error can be: (1) find 1 in a SLOTREQ bit, which corresponding to an inactive slot; (2) all active slots should be request in the same time by SLOTREQ, but an exception is found. All errors are accumulated to ACSR.SLTERR by hardware until software clears it. Software writes 0 clear this bit and write 1 has no effect.</td><td>RW</td></tr>
<tr><td>20</td><td>CRDY</td><td>External CODEC Ready. This bit is derived from the CODEC Ready bit of Slot 0 in SDATA_IN, and it indicates the external AC97 CODEC is ready or not.
<table>
<tr><th>CRDY</th><th>Description</th></tr>
<tr><td>0</td><td>CODEC is not ready</td></tr>
<tr><td>1</td><td>CODEC is ready</td></tr>
</table></td><td>R</td></tr>
<tr><td>19</td><td>CLPM</td><td>External CODEC Low Power Mode. This bit indicates the external CODEC is switched to low power mode or BIT_CLK is active from CODEC after wake up.
<table>
<tr><th>CLPM</th><th>Description</th></tr>
<tr><td>0</td><td>BIT_CLK is active</td></tr>
<tr><td>1</td><td>CODEC is switched to low power mode</td></tr>
</table></td><td>R</td></tr>
<tr><td>18</td><td>RSTO</td><td>External CODEC Registers Read Status Time Out. This bit indicates that the read status time out is detected or not. It is set to 1 if the data not return in 4 frames after a CODEC registers read command issued.
<table>
<tr><th>RSTO</th><th>Description</th></tr>
<tr><td>0</td><td>When read, indicates time out has not occurred</td></tr>
<tr><td>1</td><td>When read, indicates read status time out found</td></tr>
</table>
Write 0 clear this bit and write 1 is ignored. When RSTO is 1, it may trigger an interrupt depends on the interrupt enable setting.</td><td>RW</td></tr>
</table>

<table>
<tr><td>17</td><td>SADR</td><td>External CODEC Registers Status Address and Data Received. This bit indicates that address and data of an external AC ’97 CODEC register has or has not been received.
<table>
<tr><th>SADR</th><th>Description</th></tr>
<tr><td>0</td><td>When read, indicates no register address/data received.</td></tr>
<tr><td>1</td><td>When read, indicates address/data received.</td></tr>
</table>
Write 0 clear this bit and write 1 is ignored. When SADR is 1, it may trigger an interrupt depends on the interrupt enable setting.</td><td>RW</td></tr>
<tr><td>16</td><td>CADT</td><td>Command Address and Data Transmitted. This bit indicates that a CODEC register reading/writing command transmission has completed or not.
<table>
<tr><th>CADT</th><th>Description</th></tr>
<tr><td>0</td><td>When read, indicates the command has not done.</td></tr>
<tr><td>1</td><td>When read, indicates the command has done.</td></tr>
</table>
Write 0 clear this bit and write 1 is ignored. When CADT is 1, it may trigger an interrupt depends on the interrupt enable setting.</td><td>RW</td></tr>
<tr><td>15:0</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
</table>

13.2.8 AIC I2S/MSB-justified Status Register (I2SSR)

I2SSR is used to reflect AIC status in I2S/MSB-justified. It is a read-only register.

I2SSR **0x1002001c**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																													BSY	Reserved	Reserved
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:3</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>2</td><td>BSY</td><td>AIC busy in I2S/MSB-justified format.
<table>
<tr><th>BSY</th><th>Description</th></tr>
<tr><td>0</td><td>AIC controller is idle or disabled</td></tr>
<tr><td>1</td><td>AIC controller currently is transmitting or receiving a frame</td></tr>
</table></td><td>R</td></tr>
<tr><td>1:0</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
</table>

13.2.9 AIC AC97 CODEC Command Address Register (ACCAR) & Data Register (ACCDR)

ACCAR and ACCDR are used to hold register address and data for external AC-link CODEC register read/write operation through SDATA_OUT. The format of ACCAR.CAR and ACCDR.CDR is compliant with AC’97 Component Specification 2.3 where ACCAR.CAR[19] of “1” specifies CODEC register read operation, of “0” specifies CODEC register write operation. The write access to ACCAR and ACCDR signals AIC to issue this operation. Please reference to 13.4.4 for software flow. These registers are valid only in AC-link. It is ignored in I2S/MSB-justified format.

ACCAR **0x10020020**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												CAR																			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0	R
19:0	CAR	Command Address Register. This is used to hold 20-bit AC ’97 CODEC register address transmitted in SDATA_OUT slot 1. After this field is write, it should not be write again until the operation is finished.	RW

ACCDR **0x10020024**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												CDR																			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0	R
19:0	CDR	Command Data Register. This is used to hold 20-bit AC’97 CODEC register data transmitted in SDATA_OUT slot 2. After this field is write, it should not be write again until the operation is finished.	RW

13.2.10 AIC AC97 CODEC Status Address Register (ACSAR) & Data Register (ACSDR)

ACSAR and ACSDR are used to receive the external AC-link CODEC registers address and data from SDATA_IN. When AIC receives CODEC register status from SDATA_IN, it set ACSR.SADR bit and put the address and data to ACSAR.SAR and ACSDR.SDR. Please reference to 13.4.4 for software flow. These registers are valid only in AC-link format and are ignored in I2S/MSB-justified format.

ACSAR **0x10020028**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												SAR																			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0	R
19:0	SAR	CODEC Status Address Register. This is used to receive 20-bit AC '97 CODEC status address from SDATA_IN slot 1. Which reflect the register index for which data is being returned. The write operation is ignored.	R

ACSDR **0x1002002C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												SDR																			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0	R
19:0	SDR	CODEC Status Data Register. This is used to receive 20-bit AC '97 CODEC status data from SDATA_IN slot 2. The register data of external CODEC is returned. The write operation is ignored.	R

13.2.11 AIC I2S/MSB-justified Clock Divider Register (I2SDIV)

I2SDIV is used to set clock divider to generated BIT_CLK from SYS_CLK in I2S/MSB-justified format.

I2SDIV **0x10020030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												DV			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bits	Name	Description	RW
31:4	Reserved	Writes to these bits have no effect and always read as 0	R
3:0	DV	Audio BIT_CLK clock divider value minus 1. I2SDIV.DV is used to control the generating of BIT_CLK from dividing SYS_CLK. The dividing value should be even and I2SDIV.DV should be set to the dividing value minus 1. So I2SDIV.DV bit0 is fixed to 1. BIT_CLK frequency is fixed to 64 f_S in AIC, where f_S is the audio sample frequency. I2SDIV.DV depends on SYS_CLK frequency f_{SYS_CLK}, which is selected according to external CODEC's requirement and internal PLL frequency. Please reference to 13.4.9 "Serial Audio Clocks and Sampling Frequencies" for further description.	RW

13.2.12 AIC FIFO Data Port Register (AICDR)

AICDR is act as data input port to transmit FIFO when write and data output port from receive FIFO when read, one audio sample every time. The FIFO width is 24 bits. Audio sample with size N that is less than 24 is located in LSB N-bits. The sample size is specified by ACCR2.OASS and ACCR2.IASS in AC-link, and by I2SCR.WL in I2S/MSB-justified. The sample order is specified by ACCR1.XS and ACCR1.RS in AC-link. In I2S/MSB-justified, the left channel sample is prior to the right channel sample.

Care should be taken to monitor the status register to insure that there is room for data in the FIFO when executing a program read or write transaction. This is taken care automatically in DMA.

AICDR **0x10020034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								DATA																							
RST	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:24	Reserved	Writes to these bits have no effect and always read as 0	R
23:0	DATA	FIFO port. When write to it, data is push to the transmit FIFO. When read from it, data is pop from the receiving FIFO.	RW

13.3 Serial Interface Protocol

13.3.1 AC-link serial data format

Following figures are AC-link serial data format. Audio data is MSB adjusted, regardless of 8, 16, 18, 20, 24 bits sample size. When a 24-bits sample is transmitted, the LSB 4-bits are truncated. When try to record 24-bits sample, 4-bits of 0 are appended in LSB. Please reference to "AC '97 Component Specification Revision 2.3, 2002", provided by Intel Corporation, for details of AC '97 architecture and AC-link specification.

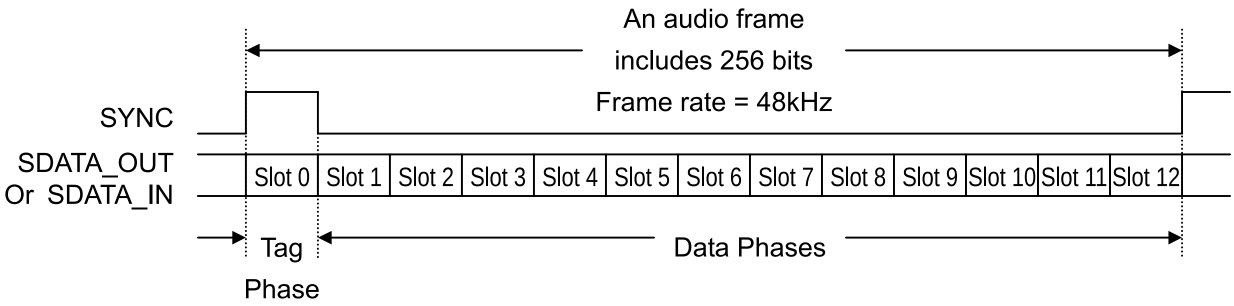

An audio frame
includes 256 bits
Frame rate = 48kHz
SYNC
SDATA_OUT
Or SDATA_IN
Slot 0
Slot 1
Slot 2
Slot 3
Slot 4
Slot 5
Slot 6
Slot 7
Slot 8
Slot 9
Slot 10
Slot 11
Slot 12
Tag
Phase
Data Phases

Figure 13-6 AC-link audio frame format

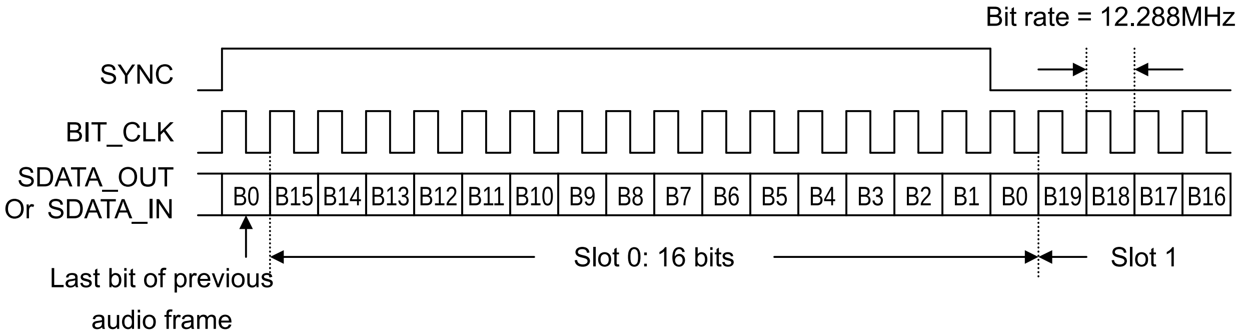

Bit rate = 12.288MHz
SYNC
BIT_CLK
SDATA_OUT
Or SDATA_IN
B0
B15
B14
B13
B12
B11
B10
B9
B8
B7
B6
B5
B4
B3
B2
B1
B0
B19
B18
B17
B16
Slot 0: 16 bits
Slot 1
Last bit of previous
audio frame

Figure 13-7 AC-link tag phase, slot 0 format

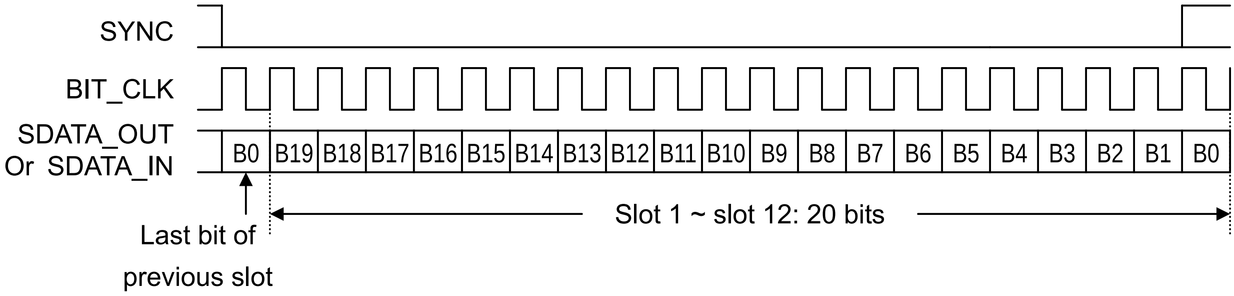

SYNC
BIT_CLK
SDATA_OUT
Or SDATA_IN
B0
B19
B18
B17
B16
B15
B14
B13
B12
B11
B10
B9
B8
B7
B6
B5
B4
B3
B2
B1
B0
Slot 1 ~ slot 12: 20 bits
Last bit of
previous slot

Figure 13-8 AC-link data phases, slot 1 ~ slot 12 format

13.3.2 I2S and MSB-justified serial audio format

Normal I2S and MSB-justified are similar protocols for digitized stereo audio transmitted over a serial path.

The BIT_CLK supplies the serial audio bit rate, the basis for the external CODEC bit-sampling logic. Its frequency is 64 times the audio sampling frequency. Divided by 64, the resulting 8 kHz to 48 kHz or even higher signal signifies timing for left and right serial data samples passing on the serial data paths. This left/right signal is sent to the CODEC on the SYNC pin. Each phase of the left/right signal is accompanied by one serial audio data sample on the data pins SDATA_IN and SDATA_OUT.

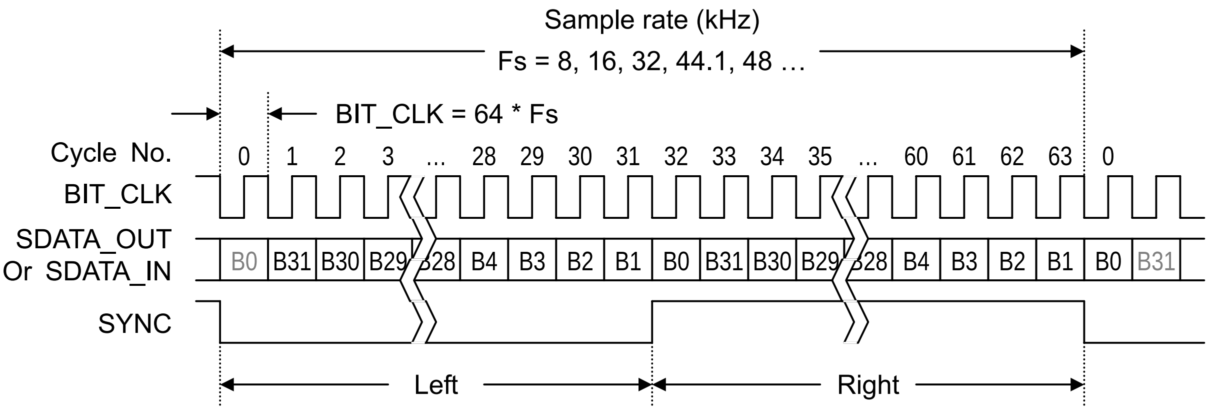

Sample rate (kHz)
Fs = 8, 16, 32, 44.1, 48 …
BIT_CLK = 64 * Fs
Cycle No.
0 1 2 3 … 28 29 30 31 32 33 34 35 … 60 61 62 63 0
BIT_CLK
SDATA_OUT
Or SDATA_IN
B0 B31 B30 B29 B28 B4 B3 B2 B1 B0 B31 B30 B29 B28 B4 B3 B2 B1 B0 B31
SYNC
Left
Right

Figure 13-9 I2S data format

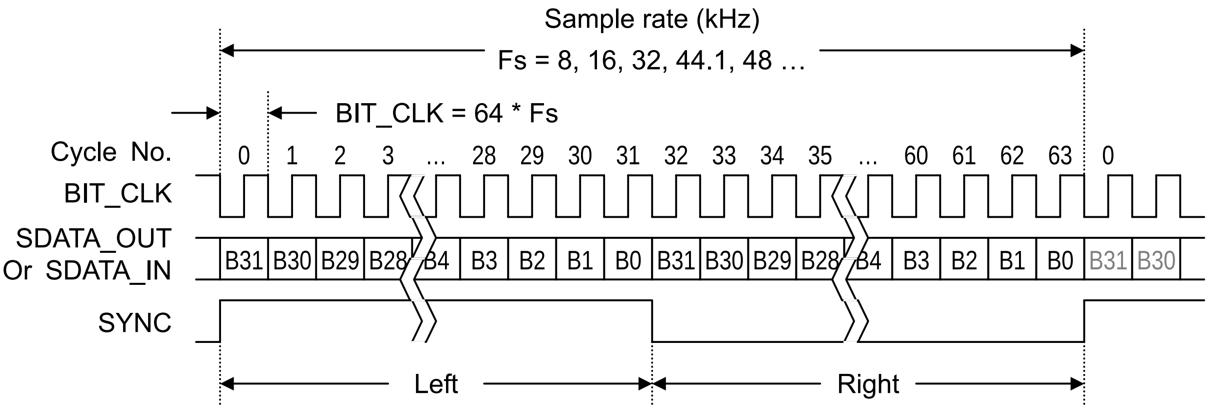

Sample rate (kHz)
Fs = 8, 16, 32, 44.1, 48 …
BIT_CLK = 64 * Fs
Cycle No.
0 1 2 3 … 28 29 30 31 32 33 34 35 … 60 61 62 63 0
BIT_CLK
SDATA_OUT
Or SDATA_IN
B31 B30 B29 B28 B4 B3 B2 B1 B0 B31 B30 B29 B28 B4 B3 B2 B1 B0 B31 B30
SYNC
Left
Right

Figure 13-10 MSB-justified data format

Figure 13-9 and Figure 13-10 provide timing diagrams that show formats for the normal I2S and MSB-justified modes of operations. Data is sampled on the rising edge of the BIT_CLK and data is sent out on the falling edge of the BIT_CLK.

Data is transmitted and received in frames of 64 BIT_CLK cycles. Each frame consists of a left sample and a right sample. Each sample holds 8, 16, 18, 20 or 24 bits of valid data. The LSB other bits of each sample is padded with zeroes.

- In the normal I2S mode, the SYNC is low for the left sample and high for the right sample. Also, the MSB of each data sample lags behind the SYNC edges by one BIT_CLK cycle.
- In the MSB-justified mode, the SYNC is high for the left sample and low for the right sample. Also, the MSB of each data sample is aligned with the SYNC edges.

When use with the internal CODEC, the BIT_CLK and SYNC signals are provided by the internal CODEC from the 12MHz clock.

13.3.3 Audio sample data placement in SDATA_IN/SDATA_OUT

The placement of audio sample in incoming/outgoing serial data stream for all formats support in AIC is MSB (Most Significant Bit) justified. Suppose n bit sample composed by

S_{n-1}	S_{n-2}	…	S_2	S_1	S_0

Table 13-3 described the how sample data bits are transferred.

Table 13-3 Sample data bit relate to SDATA_IN/SDATA_OUT bit

AC-link Format						I2S/MSB-Justified Format					
SDATA IN/OUT	**Audio Sample Size (bit)**					**SDATA IN/OUT**					
	8	**16**	**18**	**20**	**24**		**8**	**16**	**18**	**20**	**24**
B19	S7	S15	S17	S19	S23	B31	S7	S15	S17	S19	S23
B18	S6	S14	S16	S18	S22	B30	S6	S14	S16	S18	S22
B17	S5	S13	S15	S17	S21	B29	S5	S13	S15	S17	S21
B16	S4	S12	S14	S16	S20	B28	S4	S12	S14	S16	S20
B15	S3	S11	S13	S15	S19	B27	S3	S11	S13	S15	S19
B14	S2	S10	S12	S14	S18	B26	S2	S10	S12	S14	S18
B13	S1	S9	S11	S13	S17	B25	S1	S9	S11	S13	S17
B12	S0	S8	S10	S12	S16	B24	S0	S8	S10	S12	S16
B11	0	S7	S9	S11	S15	B23	0	S7	S9	S11	S15
B10	0	S6	S8	S10	S14	B22	0	S6	S8	S10	S14
B9	0	S5	S7	S9	S13	B21	0	S5	S7	S9	S13
B8	0	S4	S6	S8	S12	B20	0	S4	S6	S8	S12
B7	0	S3	S5	S7	S11	B19	0	S3	S5	S7	S11
B6	0	S2	S4	S6	S10	B18	0	S2	S4	S6	S10
B5	0	S1	S3	S5	S9	B17	0	S1	S3	S5	S9
B4	0	S0	S2	S4	S8	B16	0	S0	S2	S4	S8
B3	0	0	S1	S3	S7	B15	0	0	S1	S3	S7
B2	0	0	S0	S2	S6	B14	0	0	S0	S2	S6
B1	0	0	0	S1	S5	B13	0	0	0	S1	S5
B0	0	0	0	S0	S4	B12	0	0	0	S0	S4
						B11	0	0	0	0	S3
						B10	0	0	0	0	S2
						B9	0	0	0	0	S1
						B8	0	0	0	0	S0
						B7~ B0	0	0	0	0	0

13.4 Operation

The AIC can be accessed either by the processor using programmed I/O instructions or by the DMA controller. The processor uses programmed I/O instructions to access the AIC and can access the following types of data

- The AIC memory mapped registers data—All registers are 32 bits wide and are aligned to word boundaries.
- AIC controller FIFO data—An entry is placed into the transmit FIFO by writing to the I2S controller's Serial Audio Data register (AICDR). Writing to AICDR updates a transmit FIFO entry. Reading AICDR flushes out a receive FIFO entry.
- The external CODEC registers for I2S CODEC—CODEC registers can be accessed through the L3 bus. The L3 bus operation is emulated by software controlling three GPIO pins.
- The external CODEC registers for AC97 CODEC—An AC97 audio CODEC can contain up to sixty-four 16-bit registers. A CODEC uses a 16-bit address boundary for registers. The AIC supplies access to the CODEC registers through several registers.
- The internal CODEC registers can be accessed via memory mapped registers in the CODEC.

The DMA controller can only access the FIFOs. Accesses are made through the data registers, as explained in the previous paragraph. The DMA controller responds to the following DMA requests made by the I2S controller:

- The transmit FIFO request is based on the transmit trigger-threshold (AICFR.TFTH) setting. See 13.2.1 for further details regarding AICFR.TFTH.
- The receive FIFO request is based on the receive trigger-threshold (AICFR.RFTH) setting. See 13.2.1 for further details regarding AICFR.RFTH.

Before operation to AIC, you may need to set proper PIN function selection from GPIO using if the pin is shared with GPIO.

Please also reference to "AC '97 Component Specification Revision 2.3, 2002" when deal with AIC AC-link operations.

13.4.1 Initialization

At power-on or other hardware reset (WDT and etc), AIC is disabled. Software must initiate AIC and the internal or external CODEC after power-on or reset. If errors found in data transferring, or in other places, software must initial AIC and optional, the internal or external CODEC. Here is the initial flow.

1. Select internal or external CODEC (AICFR.ICDC).
2. If external CODEC is selected, select AC-link or I2S/MSB-Justified (AICFR.AUSEL). If internal CODEC is used, select I2S/MSB-Justified format (AICFR.AUSEL=1). If the resettlement without involving link format and architecture changing, this step can be skip.
3. If I2S/MSB-Justified is selected, select between I2S and MSB-Justified (I2SCR.AMSL), decide BIT_CLK direction (AICFR.BCKD) and SYNC direction (AICFR.SYNCD). If BIT_CLK is configured as output, BIT_CLK divider I2SDIV.DV must be set to what correspond with the values as shown in Table 13-7. And the clock selection and the divider between PLL clock out and AIC also must be set (CFCR.I2S and I2SCDR in CPM). If internal CODEC is used, select 12MHz clock input (via set proper value in CFCR.I2S and I2SCDR), I2S format (I2SCR.AMSL=0), input BIT_CLK (AICFR.BCKD=0), input SYNC (AICFR.SYNCD=0).
4. Enable AIC by write 1 to AICFR.ENB
5. If it needs to reset AIC registers and flush FIFOs, write 1 to AICFR.RST. If it need only flush FIFOs, write 1 to AICCR.FLUSH. BIT_CLK must exist here and after.
6. In AC-link format, issue a warm or cold CODEC reset.
7. In AC-link format, configure AC ’97 CODEC via ACCAR and ACCDR registers. If the resettlement doesn’t involving AC’97 CODEC registers changing, this step can be skip.
8. In case of external CODEC with I2S/MSB-Justified format, configure I2S/MSB-justified CODEC via the control bus connected to the CODEC, for instance I2C or L3, depends on CODEC. In case of internal CODEC, configure CODEC via CODEC’s memory mapped registers. If the resettlement without involving I2S/MSB-justified CODEC or ADC/DAC function changing, this step can be skip.

13.4.2 AC ’97 CODEC Power Down

AC ‘97 CODEC can be placed in a low power mode. When the CODEC’s power-down register (26h), is programmed to the appropriate value, the CODEC will be put in a low power mode and both BIT_CLK and SDATA_IN will be brought to and held at a logic low voltage level.

Once powered down, re-activation of the AC-link via re-assertion of the SYNC signal must not occur for a minimum of four audio frame times following the frame in which the power down was triggered. When AC-link powers up it indicates readiness via the CODEC Ready bit (input slot 0, bit 15).

13.4.3 Cold and Warm AC ’97 CODEC Reset

AC-link reset operations occur when the system is initially powered up, when resuming from a lower powered sleep state, and in response to critical subsystem failures that can only be recovered from with a reset.

13.4.3.1 Cold AC ‘97 CODEC Reset

A cold reset is achieved by asserting RESET# for the minimum specified time. By driving RESET# low, BIT_CLK, and SDATA_IN will be activated, or re-activated as the case may be, and all AC ‘97 CODEC registers will be initialized to their default power on reset values.

RESET# is an asynchronous AC ‘97 CODEC input.

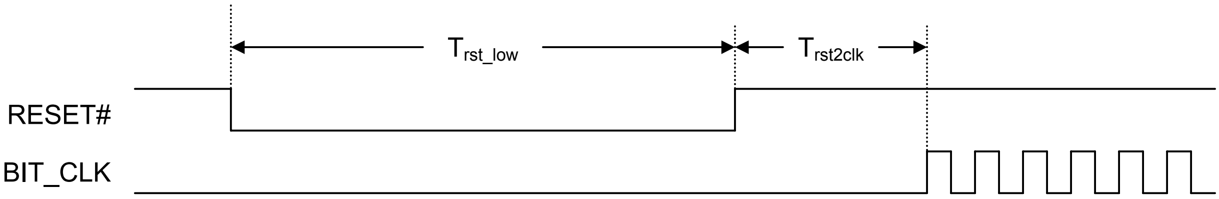


Figure 13-11 Cold AC ’97 CODEC Reset Timing

Table 13-4 Cold AC ’97 CODEC Reset Timing parameters

Parameter	Symbol	Min	Type	Max	Units
RESET# active low pulse width	T_{rst_low}	1.0	-	-	μs
RESET# inactive to BIT_CLK startup delay	$T_{rst2clk}$	162.8	-	-	ns

13.4.3.2 Warm AC ‘97 CODEC Reset

A warm AC‘97 reset will re-activate the AC-link without altering the current AC‘97 register values. Driving SYNC high for a minimum of 1 µs in the absence of BIT_CLK signals a warm reset.

Within normal audio frames SYNC is a synchronous AC ‘97 CODEC input. However, in the absence of BIT_CLK, SYNC is treated as an asynchronous input used in the generation of a warm reset to AC ’97 CODEC.

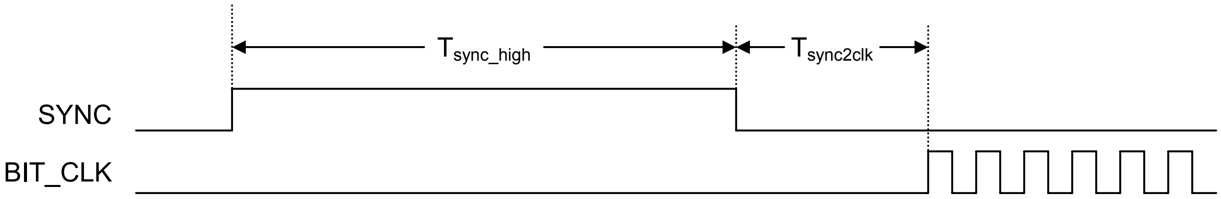


Figure 13-12 Warm AC ’97 CODEC Reset Timing

Table 13-5 Warm AC ’97 CODEC Reset Timing Parameters

Parameter	Symbol	Min	Type	Max	Units
SYNC active high pulse width	T_{sync_high}	1.0	-	-	µs
SYNC inactive to BIT_CLK startup delay	$T_{sync2clk}$	162.8	-	-	ns

13.4.4 External CODEC Registers Access Operation

The external audio CODEC can be configured/controlled by its internal registers. To access these registers, an I2S/MSB-justified CODEC usually employs L3 bus, SPI bus, I2C bus or other control bus. The L3 bus operation can be emulated by software controlling 3 GPIO pins of the chip. For AC ’97, “AC ’97 Component Specification” defines the CODEC register access protocol. Several registers are provided in AIC to accomplish this task.

The ACCAR and ACCDR are used to send a register accessing request command to external AC ’97 CODEC. The ACSAR and ACSDR are used to receive a register’s content from external AC’97 CODEC. The register accessing request and the register’s content returning is asynchronous.

The AC’97 CODEC register accessing request flow:

1. If ACSR.CADT is 0, wait for 25.4μs. If no previous accessing request, this step can be skip.
2. Clear ACSR.CADT.
3. If read access, write read-command and register address to ACCAR, if write access, write write-command and register address to ACCAR and write data to ACCDR. Any order of write ACCAR and ACCDR is OK.
4. Polling for ACSR.CADT changing to 1, which means the request has been send to CODEC via AC-link.

The AC’97 CODEC register content receiving flow by polling:

1. Polling for ACSR.SADR changing to 1
2. Read the CODEC register’s address from ACSAR and content from ACSDR
3. Clear ACSR.SADR

The AC’97 CODEC register content receiving flow by interrupt:

1. Before accessing request, clear ACSR.SADR and set ACCR2.ESADR.
2. Waiting for the interrupt. When the interrupt is found, check if ACSR.SADR is 1, if not, repeat this step again.
3. Read the CODEC register’s address from ACSAR and content from ACSDR
4. Clear ACSR.SADR

13.4.5 Audio Replay

Outgoing audio sample data (from AIC to CODEC) is written to AIC transmit FIFO from processor via store instruction or from memory via DMA. AIC then takes the data from the FIFO, serializes it, and sends it over the serial wire SDATA_OUT to an external CODEC or over an internal wire to an internal CODEC.

The audio transmission is enabled automatically when the AIC is enabled by set AICFR.ENB. But all replay data is zero at this time except both of the following conditions are true:

1. AICCR.ERPL must be 1. If AICCR.ERPL is 0, value of zero is send to CODEC even if there are samples in transmit FIFO.
2. At least one audio sample data in the transmit FIFO. If the transmit FIFO is empty, value of zero or last sample depends on AICFR.LSMP, is send to CODEC even if AICCR.ERPL is 1.

Here is the audio replay flow:

1. Configure the CODEC as needed.
2. Configure sample size by AICCR.OSS
3. Configure sample rate by clock dividers (for I2S/MSB-Justified format with BIT_CLK is provided internally) or by CODEC registers (for AC-link or BIT_CLK provided by external CODEC) or by CODEC memory mapped registers (for internal CODEC)
4. For AC-link, configure replay channels by ACCR1.XS
5. Some other configurations: mono to stereo, endian switch, signed/unsigned data transfer, transmit FIFO configuration, play ZERO or last sample when TX FIFO under-run, and etc.
6. Write 1 to AICCR.ERPL. It is suggested that at least a frame of PCM data is pre-filled in the transmit FIFO to prevent FIFO under-run flag (AICSR.TUR).
7. Fill sample data to the transmit FIFO. Repeat this till finish all sample data. In this procedure, please control the FIFO to make sure no FIFO under-run and other errors happen. When the transmit FIFO under-run, noise or pause may be heard in the audio replay, AICSR.TUR is 1, and if AICCR.ETUR is 1, AIC issues an interrupt. Please reference to 13.4.7 for detail description on FIFO.
8. Waiting for AICSR.TFL change to 0. So that all samples in the transmit FIFO has been replayed, then we can have a clean start up next time
9. Write 0 to AICCR.ERPL.

13.4.6 Audio Record

Incoming audio sample data (from CODEC to AIC) is received from SDATA_IN (for an external CODEC) or an internal wire (for an internal CODEC) serially and converted to parallel word and stored in AIC receive FIFO. Then the data can be taken from the FIFO to processor via load instruction or to memory via DMA.

The audio recording is enabled automatically when the AIC is enabled by set AICFR.ENB. But all received data is discarded at this time except both of the following conditions are true:

1. AICCR.EREC must be 1. If AICCR.EREC is 0, the received data is discarded even if there are rooms in the receive FIFO.
2. At least one room left in the receive FIFO. If the receive FIFO is full, the received data is discarded even if AICCR.EREC is 1.

Here is the audio record flow:

1. Configure the CODEC as needed.
2. Configure sample size by AICCR.ISS
3. Configure sample rate by clock dividers (for I2S/MSB-Justified format with BIT_CLK is provided internally) or by CODEC registers (for AC-link or BIT_CLK provided by external CODEC) or by CODEC memory mapped registers (for internal CODEC)
4. Some other configurations: signed/unsigned data transfer, receive FIFO configuration, and etc.
5. Write 1 to AICCR.EREC. Make sure there are rooms available in the receive FIFO before set AICCR.EREC. Usually, it should empty the receive FIFO by fetch data from it before set AICCR.EREC
6. Take sample data form the receive FIFO. Repeat this till the audio finished. In this procedure, please control the FIFO to make sure no FIFO over-run and other errors happen. When the receive FIFO over-run, same recorded audio samples will be lost, AICSR.ROR is 1, and if AICCR.EROR is 1, AIC issues an interrupt. Please reference to 13.4.7 for detail description on FIFO. For AC-link, ACCR1.RS tells which channels are recorded.
7. Write 0 to AICCR.EREC.
8. Take sample data from the receive FIFO until AICSR.RFL change to 0. So that all samples in the receive FIFO has been taken away, then we can have a clean start up next time. When the receive FIFO is empty, read from it returns zero.

13.4.7 FIFOs operation

AIC has two FIFOs, one for transmit audio sample and one for receive. All AIC played/recorded audio sample data is taken from/send to transmit/receive FIFOs. The FIFOs are in 24 bits width and 32 entries depth, one entry for keep one audio sample regardless of the sample size. AICDR.DATA provides the access point for processor/DMA to write to transmit FIFO and read from receive FIFO. One time access to AICDR.DATA process one sample. The sample data should be put in LSB (Least Significant Bit) in memory or processor registers. For transmitting, bits exceed sample are discarded. For receiving, these bits are set to 0. Figure 13-13 illustrates the FIFOs access.

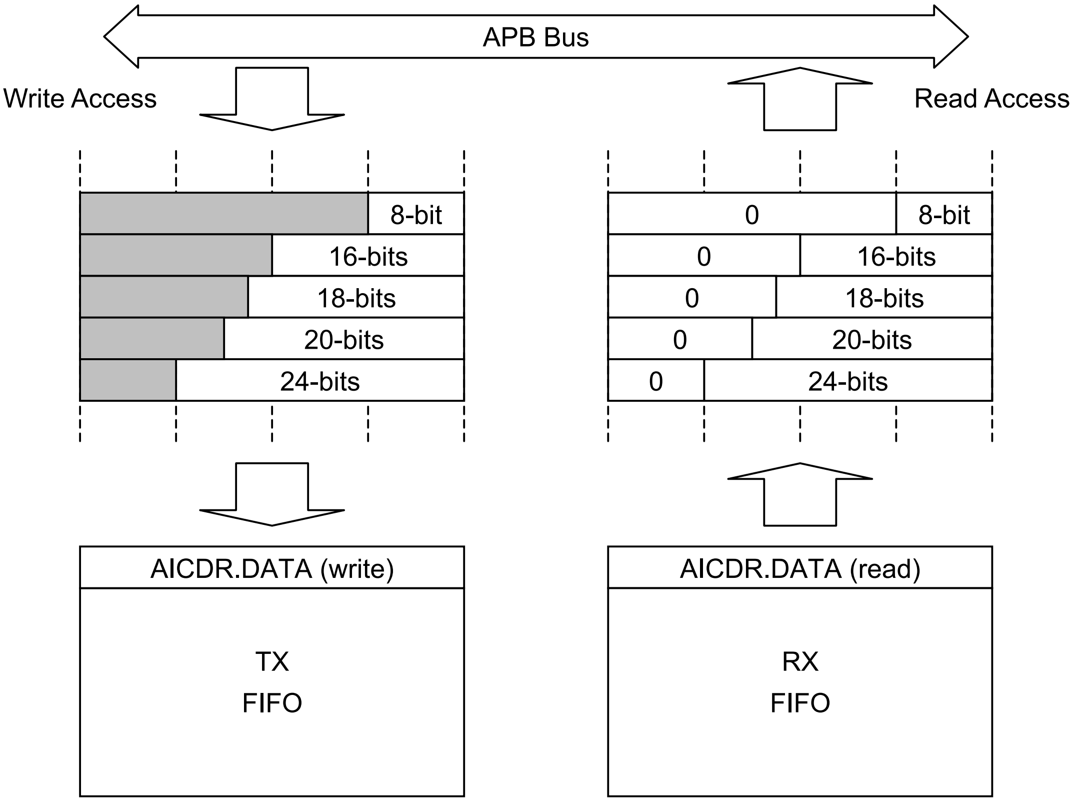

APB Bus
Write Access
Read Access
8-bit
16-bits
18-bits
20-bits
24-bits
0
8-bit
0
16-bits
0
18-bits
0
20-bits
0
24-bits
AICDR.DATA (write)
TX
FIFO
AICDR.DATA (read)
RX
FIFO

Figure 13-13 Transmitting/Receiving FIFO access via APB Bus

The software and bus initiator must guarantee the right sample placement at the bus.

In case of DMA bus initiator, one 24, 20, 18 bits audio sample must occupies one 32-bits word in memory, so 32-bits width DMA must be used. One 16 bits sample occupies one 16-bits half word in memory, so 16-bits width DMA must be used. One 8-bits sample occupies one byte in memory, and use 8-bits width DMA.

In case of processor bus initiator, any type of the audio sample must occupies one CPU general-purpose register at LSB, and read/write from/to AICDR.DATA with 32-bits load/store instruction. When process small sample size, 16-bits or 8-bits, software may need to do the data pack/unpack.

The AICFR.TFTH and AICFR.RFTH are used to set the FIFO level thresholds, which are the trig levels of DMA request and/or FIFO service interrupt. The AICFR.TFTH and AICFR.RFTH should be set to proper values, too small or too big are not good. When it is too small, the DMA burst length or the number of sample can be processed by processor is too small, which harms the bus or processor efficiency. When it is too big, the bus or the interrupt latency left for under-run/over-run is too small, which may causes replay/record errors.

AICSR.TUR is set to 1 during transmit under-run conditions. If AICCR.ETUR is 1, this can trigger an interrupt. During transmit under-run conditions, zero or last sample is continuously sent out across the serial link. Transmit under-run can occur under the following conditions:

1. Valid transmit data is still available in memory, but the DMA controller/processor starves the transmit FIFO, as it is busy servicing other higher-priority tasks.
2. The DMA controller/processor has transferred all valid data from memory to the transmit FIFO.

AICSR.ROR is set to 1 during receive over-run conditions. If AICCR.EROR is 1, this can trigger an interrupt. During receive over-run conditions, data sent by the CODEC is lost and is not recorded.

When replay/record two channels data, the left channel is always the first data in FIFOs and in the serial link. If multiple channels in AC-link are used, the channel sample order is follows the slot order.

13.4.8 Data Flow Control

There are three approaches provided to control/synchronize the audio incoming/outgoing data flow.

13.4.8.1 Polling and Processor Access

AICSR.RFL and AICSR.TFL reflect how many samples exist in receiving and transmitting FIFOs. Through read these register fields, processor can detect when there are samples in receiving FIFO in audio record and then load them from the RX-FIFO, and when there are rooms in transmitting FIFO in audio replay and then store samples to the TX-FIFO.

Polling approach is in very low efficiency and is not recommended.

13.4.8.2 Interrupt and Processor Access

Set proper values to AICFR.TFTH and AICFR.RFTH, the FIFO interrupts trig thresholds. Set AICCR.ETFS and/or AICCR.ERFS to 1 to enable transmitting and/or receiving FIFO level trigger interrupts. When the interrupt found, it means there are rooms or samples in the TX or RX FIFO, and processor can store or load samples to or from the FIFO.

Interrupt approach is more efficient than polling approach.

13.4.8.3 DMA Access

Audio data is real time stream, though it is in low data bandwidth, usually less than 1.2Mbps. DMA approach is the most efficient and is the recommended approach.

To enable DMA operation, set AICCR.TDMS and AICCR.RDMS to 1 for transmit and receive respectively. It also needs to allocate two channels in DMA controller for data transmitting and receiving respectively. Please reference to the processor's DMAC spec for the details.

The AICFR.TFTH and AICFR.RFTH are used to set the transmitting and receiving FIFO level thresholds, which determine the issuing of DMA request to DMA controller. To respond the request, DMAC initiator and controls the data movement between memory and TX/RX FIFO.

13.4.9 Serial Audio Clocks and Sampling Frequencies

For internal CODEC, CODEC module containing the audio CODEC circuit/logic and corresponding controlling registers. CODEC needs a 12MHz clock from CPM and provides BIT_CLK and LR_CLK (left-right clock which is the sample rate as SYNC) to AIC for outgoing and incoming audio respectively. These clocks change when change the sample rate in CODEC controlling registers.

For AC-link, the bit clock is input from chip external and is fixed to 12.288MHz. The sample frequency of 48kHz is supported in nature. Variable Sample Rate feature is supported in this AIC. If the CODEC supports this feature, sample rate other than 48kHz audio data can be replay directly. Otherwise, software has to do the rate transfer to replay other sample rate audio data. Double rate, 96kHz or even 88.2kHz audio is also supported with proper CODEC.

Following are for BIT_CLK/SYS_CLK configuration in I2S/MSB-Justified format with external CODEC.

The BIT_CLK is the rate at which audio data bits enter or leave the AIC. BIT_CLK can be supplied either by the CODEC or an internally PLL. If it is supplied internally, BIT_CLK is configured as output pins, and is supplied to the CODEC. If BIT_CLK is supplied by the CODEC, then it is configured as an input pin. Register bit AICFR.BCKD is used to select BIT_CLK direction.

The audio sampling frequency is the frequency of the SYNC signal, which must be 1/64 of BIT_CLK, $f_{BIT\_CLK} = 64\ f_S$.

SYS_CLK is only for CODEC. It usually takes one of the two roles, as CODEC master clock input or as CODEC over-sampling clock input. If SYS_CLK roles as CODEC master clock input, it usually should be set to a fixed frequency according to CODEC requirement but independent to audio sample rate. In this case, usually there is a PLL in the CODEC and CODEC roles master mode. See Figure 13-4 for the interface diagram. This is the recommended AIC CODEC system configuration.

If SYS_CLK roles as CODEC over-sampling clock, its frequency is usually 4, 6, 8 or 12 times of BIT_CLK frequency, which are 256, 384, 512 and 768 times of audio sample rates. Table 13-6 lists the relation between sample rate, BIT_CLK and SYS_CLK frequencies.

Table 13-6 Audio Sampling rate, BIT_CLK and SYS_CLK frequencies

Sample Rate f_S (kHz)	**BIT_CLK (MHz) $f_{BIT\_CLK} = 64\ f_S$**	**SYS_CLK (MHz)**			
		256 f_S	**384 f_S**	**512 f_S**	**768 f_S**
48	3.072	12.288	18.432	24.576	36.864
44.1	2.8224	11.2896	16.9344	22.5792	33.8688
32	2.048	8.192	12.288	16.384	24.576
24	1.536	6.144	9.216	12.288	18.432
22.05	1.4112	5.6448	8.4672	11.2896	16.9344

16	1.024	4.096	6.144	8.192	12.288
11.025	0.7056	2.8224	4.2336	5.6448	8.4672
8	0.512	2.048	3.072	4.096	6.144

In this processor, SYS_CLK can be selected from EXCLK or generated by dividing the PLL output clock in a CPM divider controlled by I2SCDR. If BIT_CLK is chosen as an output, another divider in AIC is used to divide SYS_CLK for it. Figure 13-14 illustrates the AIC clock generation scheme.

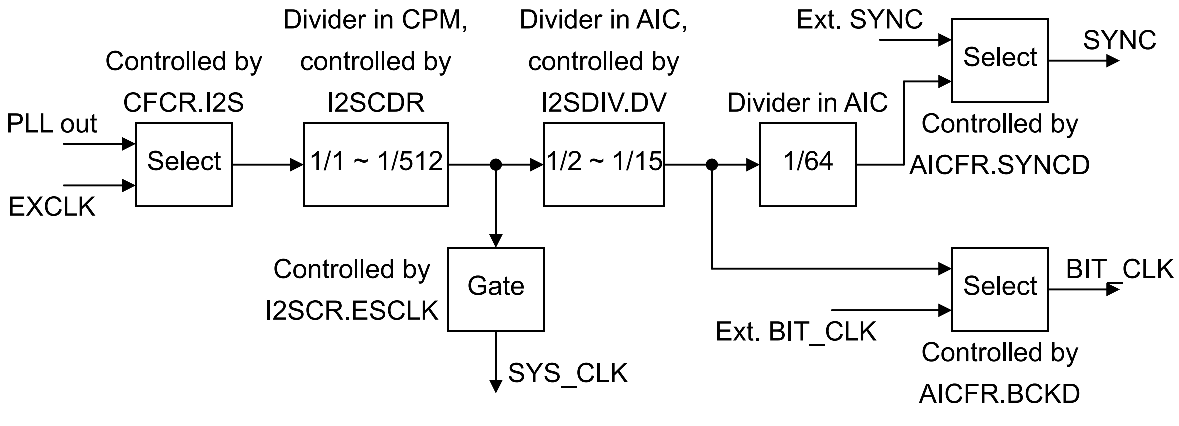

PLL out
EXCLK
Controlled by
CFCR.I2S
Select
Divider in CPM,
controlled by
I2SCDR
1/1 ~ 1/512
Divider in AIC,
controlled by
I2SDIV.DV
1/2 ~ 1/15
Controlled by
I2SCR.ESCLK
Gate
SYS_CLK
Divider in AIC
1/64
Ext. SYNC
Select
SYNC
Controlled by
AICFR.SYNCD
Select
BIT_CLK
Ext. BIT_CLK
Controlled by
AICFR.BCKD

Figure 13-14 SYS_CLK, BIT_CLK and SYNC generation scheme

The setting of I2SDIV.DV is shown in Table 13-7.

Table 13-7 BIT_CLK divider setting

I2SDIV.DV	f_{SYS_CLK}	f_{BIT_CLK}	f_{SYS_CLK} / f_{BIT_CLK}
0x1	128 f_S	64 f_S	2
0x2	196 f_S	64 f_S	3
0x3	256 f_S	64 f_S	4
0x5	384 f_S	64 f_S	6
0x7	512 f_S	64 f_S	8
0xB	768 f_S	64 f_S	12

As we observe in Table 13-6, if SYS_CLK is taken as over-sampling clock by CODEC, the common multiple of all SYS_CLK frequencies is much bigger than the PLL output clock frequency. To generate all different SYS_CLK frequencies, one approach is change PLL frequency according to sample rate. This is not realistic, since frequently change PLL frequency during normal operation is not recommended.

Another approach is to found some approximate common multiples of all SYS_CLK frequencies according to the fact that there tolerance in audio sample rate. Take f_{SYS_CLK} = 256 f_S, Table 13-8 list most frequencies, which are less than 400MHz, with relatively small sample rate errors. It is suggested to set PLL frequency as close to the frequencies listed as possible, then use clock dividers to generate different SYS_CLK/BIT_CLK for different sample rate.

Table 13-8 Approximate common multiple of SYS_CLK for all sample rates

Approximate Common Frequency (MHz)	Max Error Caused in Audio Sample Rate (%)
123.53	0.53
147.11	0.24
170.68	0.79
235.5	0.87
247.06	0.53
270.64	0.11
280.56	0.73
294.22	0.24
305.14	0.67
317.79	0.53
329.57	0.66
341.35	0.79
347	0.85
353.13	0.90
358.79	0.69
370.59	0.53
382.96	0.54
394.17	0.24

Take PLL = 270.64 MHz as an example, Table 13-9 lists the divider settings for various sample rates.

Table 13-9 CPM/AIC clock divider setting for various sampling rate if PLL = 270.64MHz

Sample Rate (kHz)	I2SCDR	I2SDIV.DV	Sample Rate Error (%)
48	1	11	0.11
44.1	1	12	-0.11
32	0	33	0.11
24	1	22	0.11
22.05	1	24	-0.11
16	1	33	0.11
12	1	44	0.11
11.025	1	48	-0.11
8	1	66	0.11

For an EXCLK clock frequency, try to generate PLL frequencies as close to the frequencies listed in Table 13-8 as possible. Table 13-10 lists the PLL parameters and audio sample errors at different PLL frequencies for EXCLK at 12MHz.

Table 13-10 PLL parameters and audio sample errors for EXCLK=12MHz

PLL			Max Sample Rate Error
M	**N**	**Freq (MHz)**	
103	10	123.6	0.59%
49	4	147	0.31%
128	9	170.67	0.79%
157	8	235.5	0.87%
103	5	247.2	0.59%
65	3	260	0.82%
45	2	270	0.35%
203	9	270.67	0.12%
113	5	271.2	0.32%
187	8	280.5	0.75%
237	10	284.4	0.81%
49	2	294	0.31%
178	7	305.14	0.67%
53	2	318	0.60%
302	11	329.45	0.70%
256	9	341.33	0.79%
318	11	346.91	0.88%
206	7	353.14	0.90%
299	10	358.8	0.69%
247	8	370.5	0.55%
351	11	382.91	0.55%
230	7	394.29	0.27%

The BIT_CLK should be stopped temporary when change the divider settings, or when change BIT_CLK source (from internal or external), to prevent clock glitch. Register I2SCR.STPBK is provided to assist the task. When I2SCR.STPBK = 1, BIT_CLK is disabled no matter whether it is generated internally or inputted from the external source. The operation flow is described in following.

1. Stop all replay/record by clear AICCR.ERPL and AICCR.EREC.
2. Polling I2SSR.BSY till it is 0
3. Stop the BIT_CLK by write 1 to I2SCR.STPBK
4. Operations concerning BIT_CLK
5. Resume the BIT_CLK by write 0 to I2SCR.STPBK

13.4.10 Interrupts

The following status bits, if enabled, interrupt the processor:

- Receive FIFO Service (AICSR.RFS). It's also DMA Request
- Transmit FIFO Service (AICSR.TFS). It's also DMA Request
- Transmit Under-Run (AICSR.TUR)
- Receive Over-Run (AICSR.ROR)
- Command Address and Data Transmitted, AC-link only (ACSR.CADT)
- External CODEC Registers Status Address and Data Received, AC-link only (ACSR.SADR)
- External CODEC Registers Read Status Time Out, AC-link only (ACSR.RSTO)

For further details, see the corresponding register description sections.

14 Internal CODEC

14.1 Overview

This chapter describes internal audio CODEC embedded in the processor and related software interface.

The internal CODEC is an I2S audio CODEC with 18 bits DAC and 16 bits ADC. It also has several memory mapped registers used to control and configure the CODEC. AIC is used to interface to the CODEC for audio data replaying and recording.

14.1.1 Features

The following are internal CODEC features:

- DAC: 18 bits sample size, SNR 90dB
- ADC: 16 bits sample size, SNR 85dB
- Sample rate: 8kHz, 11.025kHz, 12kHz, 16kHz, 22.05kHz, 24kHz, 32kHz, 44.1kHz and 48kHz
- Head phone amplifier to support up to 16ohm load.
- Anti-pop for head phone out
- Low power dissipation mode
- Digital volume control

14.1.2 Signal Descriptions

CODEC has 5 ~ 7 signal IO pins depending on various chips. They are listed and described in Table 14-1.

Table 14-1 CODEC signal IO pin description

Pin Names	IO	4740 Loc	4720 Loc	IO Cell Char.	Pin Description	Power
LHPO	AO	E14			LHPO: Left headphone out	VDD_{CDC}
RHPO	AO	E13			RHPO: Right headphone out	VDD_{CDC}
MICIN	AI	D14			MICIN: Microphone input	VDD_{CDC}
MICBIAS	AO	E15			MICBIAS: Microphone bias	VDD_{CDC}
LLINEIN	AI	D12	▨		LLINEIN: Left line input	VDD_{CDC}
RLINEIN	AI	D13	▨		RLINEIN: Right line input	VDD_{CDC}
VREF	AO	E12			VREF: Voltage Reference Output. An electrolytic capacitor more than 10μF in parallel with a 0.1μF ceramic capacitor attached from this pin to VSSCDC eliminates the effects of high frequency noise	VDD_{CDC}
VDDHP	P	G12			VDDHP: Headphone amplifier power, 3.3V	-
VSSHP	P	G10			VSSHP: Headphone amplifier ground	-

Pin Names	IO	4740 Loc	4720 Loc	IO Cell Char.	Pin Description	Power
VDDCDC	P	D11			VDDCDC: CODEC analog power, 3.3V	-
VSSCDC	P	F9			VSSCDC: CODEC analog ground	-

14.1.3 Block Diagram

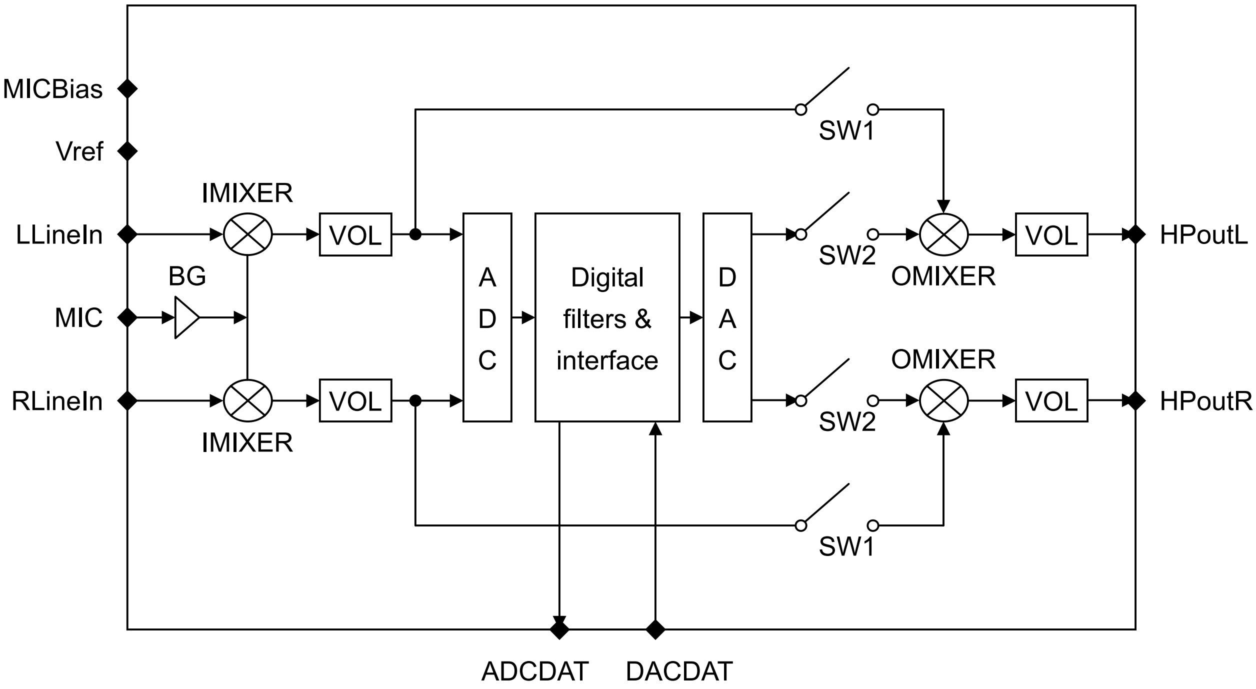

MICBias
Vref
LLineIn
MIC
RLineIn
BG
IMIXER
IMIXER
VOL
VOL
A
D
C
Digital
filters &
interface
D
A
C
SW1
SW2
SW2
SW1
OMIXER
OMIXER
VOL
VOL
HPoutL
HPoutR
ADCDAT
DACDAT

Figure 14-1 CODEC block diagram

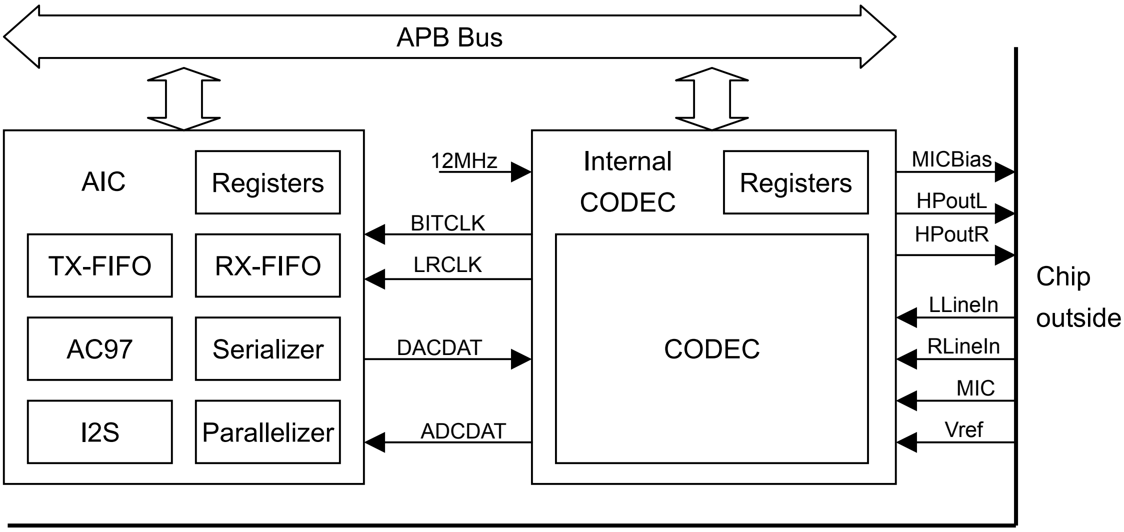

APB Bus
AIC
Registers
TX-FIFO
RX-FIFO
AC97
Serializer
I2S
Parallelizer
12MHz
BITCLK
LRCLK
DACDAT
ADCDAT
Internal
CODEC
Registers
CODEC
MICBias
HPoutL
HPoutR
LLineIn
RLineIn
MIC
Vref
Chip
outside

Figure 14-2 Internal CODEC works with AIC

14.2 Register Descriptions

The internal CODEC software interface includes 2 registers. They are mapped in IO memory address space so that program can access them to control the operations of the CODEC.

Table 14-2 Internal CODEC Registers Description

Name	Description	RW	Reset value	Address	Size
CDCCR1	CODEC Control Register 1	RW	0x021B2302	0x10020080	32
CDCCR2	CODEC Control Register 2	RW	0x00170803	0x10020084	32

- CDCCR1 is used to control MIC input, LINE input, headphone out, ADC, DAC, CODEC suspend/reset and anti-pop procedures.
- CDCCR2 is used to control values or gains of MIC input, LINE input and headphone, and audio sample rate.

14.2.1 CODEC Control Register 1 (CDCCR1)

CDCCR1 contains bits to control MIC input, LINE input, headphone out, ADC, DAC, CODEC suspend/reset and anti-pop procedures. Set AICFR.ICDC to 1 before write to this register, or the effect is undefined.

CDCCR1 **0x10020080**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved		ELININ	EMIC	SW1ON	EADC	SW2ON	EDAC	Reserved			PDVR	PDVRA	VRPLD	VRCGL	VRCGH	Reserved	HPMUTE	HPOV0	HPCG	HPPLDM	HPPLDR	PDHPM	PDHP	Reserved						SUSPD	RST
RST	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31:30</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>29</td><td>ELININ</td><td>LINE input enabled.
<table>
<tr><th>ELININ</th><th>Description</th></tr>
<tr><td>0</td><td>LINE input is disabled</td></tr>
<tr><td>1</td><td>LINE input is enabled</td></tr>
</table></td><td>RW</td></tr>
<tr><td>28</td><td>EMIC</td><td>MIC input enabled.
<table>
<tr><th>EMIC</th><th>Description</th></tr>
<tr><td>0</td><td>MIC input is disabled</td></tr>
<tr><td>1</td><td>MIC input is enabled</td></tr>
</table></td><td>RW</td></tr>
<tr><td>27</td><td>SW1ON</td><td>Switch 1 (SW1) in CODEC is on. When switch 1 is on, the input audio is taken by the output audio mixer and sends to the headphone output.
<table>
<tr><th>SW1ON</th><th>Description</th></tr>
<tr><td>0</td><td>SW1 is off</td></tr>
<tr><td>1</td><td>SW1 is on</td></tr>
</table></td><td>RW</td></tr>
<tr><td>26</td><td>EADC</td><td>Enable ADC.
<table>
<tr><th>EADC</th><th>Description</th></tr>
<tr><td>0</td><td>The ADC is disabled. No AD convert can be down</td></tr>
<tr><td>1</td><td>The ADC is enabled.</td></tr>
</table></td><td>RW</td></tr>
<tr><td>25</td><td>SW2ON</td><td>Switch 2 (SW2) in CODEC is on. When switch 2 is on, the audio from DAC is taken by the output audio mixer and sends to the headphone output. If this switch is off, DAC audio cannot be heard
<table>
<tr><th>SW2ON</th><th>Description</th></tr>
<tr><td>0</td><td>SW2 is off</td></tr>
<tr><td>1</td><td>SW2 is on</td></tr>
</table></td><td>RW</td></tr>
<tr><td>24</td><td>EDAC</td><td>Enable ADC.
<table>
<tr><th>EADC</th><th>Description</th></tr>
</table></td><td>RW</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>0</td><td>The DAC is disabled. No DA convert can be down</td></tr><tr><td>1</td><td>The DAC is enabled.</td></tr></table></td><td></td></tr>
<tr><td>23:21</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>20</td><td>PDVR</td><td>Power down Vref.</td><td>RW</td></tr>
<tr><td>19</td><td>PDVRA</td><td>Power down Vref amplifier.</td><td>RW</td></tr>
<tr><td>18</td><td>VRPLD</td><td>Vref pull-down</td><td>RW</td></tr>
<tr><td>17</td><td>VRCGL</td><td>Charge Vref capacitors with lower current.</td><td>RW</td></tr>
<tr><td>16</td><td>VRCGH</td><td>Charge Vref capacitors with high current.</td><td>RW</td></tr>
<tr><td>15</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>14</td><td>HPMUTE</td><td>Headphone Mute.<table><tr><th>HPMUTE</th><th>Description</th></tr><tr><td>0</td><td>Headphone is not mute</td></tr><tr><td>1</td><td>Headphone is mute</td></tr></table></td><td>RW</td></tr>
<tr><td>13</td><td>HPOV0</td><td>Headphone amplifier value changed at the audio over zero. This bit should be set to 1 in most cases to prevent noise when change HP amplifier value by change HPVOL. It must be set to 0 in linear anti pop procedure.<table><tr><th>HPOV0</th><th>Description</th></tr><tr><td>0</td><td>Headphone amplifier value changed at any time</td></tr><tr><td>1</td><td>Headphone amplifier value changed at the audio over zero.</td></tr></table></td><td>RW</td></tr>
<tr><td>12</td><td>HPCG</td><td>Change HP</td><td>RW</td></tr>
<tr><td>11</td><td>HPPLDM</td><td>Pull-down HP in M mode</td><td>RW</td></tr>
<tr><td>10</td><td>HPPLDR</td><td>Pull-down HP in R mode</td><td>RW</td></tr>
<tr><td>9</td><td>PDHPM</td><td>Power down HP in M mode</td><td>RW</td></tr>
<tr><td>8</td><td>PDHP</td><td>Power down HP</td><td>RW</td></tr>
<tr><td>7:2</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0</td><td>R</td></tr>
<tr><td>1</td><td>SUSPD</td><td>CODEC suspend. When this bit is 1, CODEC is forced to suspend mode, which consumes minimum power. Before headphone amplifier turn on procedure, change this bit to 0. If HP amplifier is on, please turn it off before set this bit to 1. Otherwise, pop noise will be found. Please reference to 14.3.3 for more details about CODEC power consumption.</td><td>RW</td></tr>
<tr><td>0</td><td>RST</td><td>Reset the CODEC. The RST should be kept for at least 2us. It should not reset CODEC for too long time. Don't reset CODEC during replay/record, or noise will be found. The CODEC should be reset every time before it wakes up or power on.<table><tr><th>RST</th><th>Description</th></tr><tr><td>0</td><td>CODEC is not reset</td></tr><tr><td>1</td><td>CODEC is reset</td></tr></table></td><td>RW</td></tr>
</table>

14.2.2 CODEC Control Register 2 (CDCCR2)

CDCCR2 contains bits to control values or gains of MIC input, LINE input and headphone, and audio sample rate. Set AICFR.ICDC to 1 before write to this register, or the effect is undefined.

CDCCR2 **0x10020084**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved											AINVOL					Reserved				SMPR				Reserved		MICBG		Reserved		HPVOL	
RST	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1

Bits	Name	Description	RW
31:21	Reserved	Writes to these bits have no effect and always read as 0	R
20:16	AINVOL	The audio input programmable gain amplifier volume control. It influences both MIC input and line input volume. AINVOL | Gain 0 | -34.5 dB 1 | -33.0 dB … | (AINVOL * 1.5) – 34.5 23 | 0 dB … | … 30 | +10.5 dB 31 | +12.0 dB	RW
15:12	Reserved	Writes to these bits have no effect and always read as 0	R
11:8	SMPR	The audio sample rate. There is an error of 0.04% exists for some sample rates. The sample rate for ADC and DAC is the same if they work in the same time. SMPR should be changed during AIC register AICCR.EREC=0 and AICCR.ERPL=0, or noise may be recorded/heard. SMPR | Nominal Sample Rate (kHz) | Actual Sample Rate (kHz) 0000 | 8 | 8 0001 | 11.025 | 11.029 0010 | 12 | 12 0011 | 16 | 16 0100 | 22.05 | 22.059 0101 | 24 | 24 0110 | 32 | 32 0111 | 44.1 | 44.118 1000 | 48 | 48	RW
7:6	Reserved	Writes to these bits have no effect and always read as 0	R

5:4	MICBG	MIC Boost Gain. **MICBG** | **MIC Boost Gain** 00 | 0 dB 01 | 6 dB 10 | 12 dB 11 | 20 dB	RW
3:2	Reserved	Writes to these bits have no effect and always read as 0	R
1:0	HPVOL	Headphone amplifier volume control. In case of HPVOL=’b11, the maximum positive sample data produces 0 voltage in HPOUT while maximum negative sample data produces VDDHP voltage. The maximum peak-to-peak voltage is VDDHP. In case of HPVOL=’b00, The maximum HPOUT peak-to-peak voltage is half of VDDHP and the middle of the wave is still VREF voltage. **HPVOL** | **Gain** 00 | 0 dB 01 | 2 dB 10 | 4 dB 11 | 6 dB	RW

14.3 Operation

The internal CODEC can be accessed by the processor using programmed I/O instructions via memory mapped registers. CODEC memory mapped registers are only for the CODEC controlling. The audio data transferring, i.e. audio replaying and recording, is down by AIC. AIC still takes the role of I2S controller where CODEC memory mapped registers take the role of CODEC controlling interface just like L3 bus or I2C bus for an external CODEC. We will refer to many AIC operations and registers in the following audio operation descriptions. Please reference to AIC spec for the details.

14.3.1 Initialization

At power-on or other hardware reset (WDT, wakeup from hibernating mode and etc), The CODEC is reset and is put in suspend mode. The CODEC is also be reset at the time it leaves suspend mode. So if there's error found in the CODEC, set CDCCR.SUSPD to 1 and then set it to 0 will reset CODEC from the error.

To use the internal CODEC with AIC, several AIC registers should be set as,

AICFR.ICDC = 1
AICFR.AUSEL = 1
AICFR.BCKD = 0
AICFR.SYNCD = 0
I2SCR.AMSL = 0

14.3.2 CODEC controlling and typical operations

Table 14-3 CODEC settings in various applications

SUSPD	ELININ	EMIC	Set MICBG	Set AINVOL	EADC	SW1ON	EDAC	SW2ON	Set SMPR	HP Amp	HPMUTE	Set HPVOL	Applications
0	0	0	N	N	0	0	1	1	Y	ON	0	Y	Audio data replay
0	1	0	N	Y	0	1	1	1	Y	ON	0	Y	Audio data replay mixed with LINE input
0	0	1	Y	Y	0	1	1	1	Y	ON	0	Y	Audio data replay mixed with MIC input
0	1	1	Y	Y	0	1	1	1	Y	ON	0	Y	Audio data replay mixed with MIC and LINE input
0	0	1	Y	Y	1	0	1	1	Y	ON	0	Y	Audio data replay while record MIC input
0	1	0	N	Y	0	1	0	0	N	ON	0	Y	Playback LINE input audio
0	1	0	N	Y	1	0	0	0	Y	ON	1	N	Record LINE input audio without playback 1[1]
0	1	0	N	Y	1	?	0	?	Y	OFF	?	N	Record LINE input audio without playback 2[2]
0	1	0	N	Y	1	1	0	0	Y	ON	0	Y	Record LINE input audio with playback
0	0	1	Y	Y	0	1	0	0	N	ON	0	Y	Playback MIC input audio
0	0	1	Y	Y	1	0	0	0	Y	ON	1	N	Record MIC input audio without playback 1[1]
0	0	1	Y	Y	1	?	0	?	Y	OFF	?	N	Record MIC input audio without playback 2[2]
0	0	1	Y	Y	1	1	0	0	Y	ON	0	Y	Record MIC input audio with playback
0	1	1	Y	Y	0	1	0	0	N	ON	0	Y	Playback MIC/LINE mixed input audio
0	1	1	Y	Y	1	0	0	0	Y	ON	1	N	Record MIC/LINE mixed input audio without playback 1[1]
0	1	1	Y	Y	1	?	0	?	Y	OFF	?	N	Record MIC/LINE mixed input audio without playback 2[2]
0	1	1	Y	Y	1	1	0	0	Y	ON	0	Y	Record MIC/LINE mixed input audio with playback
1	?	?	?	?	?	?	?	?	?	?	?	?	CODEC is off, no action can be taken

Notes:

1. HP amplifier turn on/turn off procedures are complicate and may produce noise.
2. Turn off HP amplifier can save power

Table 14-3 lists the CODEC settings in many applications. Following are more details for some typical operations.

14.3.2.1 Audio data replay

To replay audio data to the internal CODEC, please consult the following steps.

1. Turn HP amplifier on if it is off or the CODEC is suspended.
2. Set CDCCR1.ELININ=0, CDCCR1.EMIC=0, CDCCR1.EADC=0, CDCCR1.SW1ON=0, CDCCR1.EDAC=1, CDCCR1.SW2ON=1, CDCCR1.HPMUTE=0

3. Set proper HP amplifier volume CDCCR2.HPVOL
4. Set proper sample rate CDCCR2.SMPR
5. Set proper sample size AICCR.OSS
6. Configure other audio replaying features
7. Configure AIC TX-FIFO, interrupt
8. Setup DMA and interrupt for audio data
9. Set AICCR.ERPL=1 to replay
10. After finished the data replaying, set AICCR.ERPL=0

14.3.2.2 Audio data replay while record MIC input without playback

To replay audio data to the internal CODEC, in the same time, record audio from the internal CODEC MIC input without playback them, please consult the following steps.

1. Turn HP amplifier on if it is off or the CODEC is suspended.
2. Set CDCCR1.ELININ=0, CDCCR1.EMIC=1, CDCCR1.EADC=1, CDCCR1.SW1ON=0, CDCCR1.EDAC=1, CDCCR1.SW2ON=1, CDCCR1.HPMUTE=0
3. Set proper MIC boost gain volume CDCCR2.MICBG, input amplifier volume CDCCR2.AINVOL and HP amplifier volume CDCCR2.HPVOL
4. Set proper ADC and DAC sample rate CDCCR2.SMPR
5. Set proper DAC sample size AICCR.OSS and ADC sample size AICCR.ISS to 16 bits
6. Configure other audio replaying features
7. Configure AIC TX-FIFO, RX-FIFO, interrupt
8. Setup DMA and interrupt for both incoming and outgoing audio data
9. Set AICCR.ERPL=1 to replay and AICCR.EREC=1 to record
10. After finished the data replaying, set AICCR.ERPL=0
11. After finished the record, set AICCR.EREC=0

14.3.2.3 Playback LINE input audio

To playback audio LINE input in the internal CODEC, please consult the following steps.

1. Turn HP amplifier on if it is off or the CODEC is suspended.
2. Set CDCCR1.ELININ=1, CDCCR1.EMIC=0, CDCCR1.EADC=0, CDCCR1.SW1ON=1, CDCCR1.EDAC=0, CDCCR1.SW2ON=0, CDCCR1.HPMUTE=0
3. Set proper input amplifier volume CDCCR2.AINVOL and HP amplifier volume CDCCR2.HPVOL

14.3.2.4 Record LINE input audio with playback

To record audio from the internal CODEC LINE input, in the same time playback them, please consult the following steps.

1. Turn HP amplifier on if it is off or the CODEC is suspended.
2. Set CDCCR1.ELININ=1, CDCCR1.EMIC=0, CDCCR1.EADC=1, CDCCR1.SW1ON=1, CDCCR1.EDAC=0, CDCCR1.SW2ON=0, CDCCR1.HPMUTE=0
3. Set proper input amplifier volume CDCCR2.AINVOL and HP amplifier volume CDCCR2.HPVOL
4. Set proper sample rate CDCCR2.SMPR
5. Set sample size AICCR.ISS to 16 bits
6. Configure other audio record features
7. Configure AIC RX-FIFO, interrupt
8. Setup DMA and interrupt for audio data
9. Set AICCR.EREC=1 to record
10. After finished the record, set AICCR.EREC=0

14.3.2.5 Record MIC input audio without playback 2

To record audio from the internal CODEC MIC input without playback them, please consult the following steps.

1. If HP amplifier is on, turn it off.
2. Set CDCCR1.SUSPD=0, CDCCR1.ELININ=0, CDCCR1.EMIC=1, CDCCR1.EADC=1, CDCCR1.EDAC=0
3. Set proper MIC boost gain volume CDCCR2.MICBG and input amplifier volume CDCCR2.AINVOL
4. Set proper sample rate CDCCR.SMPR
5. Set sample size AICCR.ISS to 16 bits
6. Configure other audio record features
7. Configure AIC RX-FIFO, interrupt
8. Setup DMA and interrupt for audio data
9. Set AICCR.EREC=1 to record
10. After finished the record, set AICCR.EREC=0

14.3.3 Power saving

14.3.4 Pop noise and the reduction of it

The internal CODEC includes an amplifier for headphone output. The pop noise when headphone amplifies turning on and turning off (power on/off) is normally an issue for an audio CODEC. In this processor, we provide several approaches to reduce the pop noise.

The sound of the headphone depends on the voltage change of the HPOUT pins. The HPOUT voltage changing speed decides the sound tune and the changing range decides the sound magnitude. VREF is the reference voltage. It will be Vm, the half of VDDCDC and is the middle voltage of HPOUT when CODEC is working. The HPOUT and VREF voltage should be bring to Vm after this chip and the CODEC power on and before play audio from it. No noise will be found when start play audio in case of HPOUT and VREF are Vm. So the key point is bring VREF and HPOUT to Vm in a noise free and fast procedure.

14.3.4.1 Charge and discharge of VREF and HPOUT

The timing values listed in the following tables are measured in the condition of:

- Two 220uF capacitors are connected to HPOUTL and HPOUTR pins respectively
- 0.1uF + 10uF capacitors are connected to VREF pin
- Room temperature

The standard deviation (σ) of the timing is also listed as well as the average timing value. According to statistical theory, the average timing ±3σ covers 99.7% cases, ±4σ covers 99.994% cases. So the (average + 4σ) timing value is recommended.

Table 14-4 Charge VREF to the middle voltage Vm

Operation	Timing	No Noise Condition
Set CDCCR1 = 0x000b2302	VREF 0→Vm: 0.8±?s	
Set CDCCR1 = 0x00002302?	VREF 0→Vm: 50±?ms?	

Note:

In case of CDCCR1 = 0x000b2302, the whole CODEC power consumption is about 410uA.

In case of CDCCR1 = 0x00002302, the whole CODEC power consumption is about ?uA.

Table 14-5 Charge HPOUT to the middle voltage Vm

Operation	Timing	No Noise Condition
Replay data from 0x1FFFF to 0	HPOUT 0→Vm: 500±?ms	VREF = Vm, HPOUT start from 0V, CDCCR2.HPVOL=3, CDCCR1.HPOV0=0

Table 14-6 Charge VREF and HPOUT to the middle voltage Vm

Operation	Timing	No Noise Condition
Set CDCCR1 = 0x00033302	VREF 0→Vm: 0.8±?s HPOUT 0→Vm: 1.5±?s	Both VREF and HPOUT start from 0V
	HPOUT (Vm – 0.1V)→Vm: 0.7±?s	VREF = Vm, HPOUT > (Vm – 0.1V)

Note:

In case of CDCCR1 = 0x00033302, the whole CODEC power consumption is about 1mA.

Table 14-7 Discharge HPOUT to 0V

Operation	Timing	No Noise Condition
Set CDCCR1 = 0x001b2102	HPOUT Vm→0: 1.7±?s	Change to CDCCR1 = 0x001b2302 first

Note:

In case of CDCCR1 = 0x001b2102, the whole CODEC power consumption is < 0.1uA.

Table 14-8 Discharge VREF to 0V

Operation	Timing	No Noise Condition
Set CDCCR1 = 0x001f2302	VREF Vm→0: 3±?ms	

Note:

In case of CDCCR1 = 0x001f2302, the whole CODEC power consumption is < 0.1uA.

Table 14-9 Discharge VREF and HPOUT to 0V

Operation	Timing	No Noise Condition
Set CDCCR1 = 0x001f2102	HPOUT Vm→0: 1.7±?s VREF Vm→0: 3±?ms	Change to CDCCR1 = 0x001b2302 first

Note:

In case of CDCCR1 = 0x001f2102, the whole CODEC power consumption is about 500uA.

14.3.4.2 Leakage of HPOUT and VREF

Set CDCCR1 to 0x001b2302 after HPOUT and VREF are charged Vm, the electrons in HPOUT and VREF capacitors will leak slowly and the voltage will be down slowly. The voltage down speed is greatly depends on the system environment, the HPOUT/VREF capacitance, PCB, capacitor leakage characteristics, temperature, humidity and etc.

Please measure the HPOUT leak 0.1V time, TL01, for your system at the worst case: the highest working temperature and humidity. The steps are

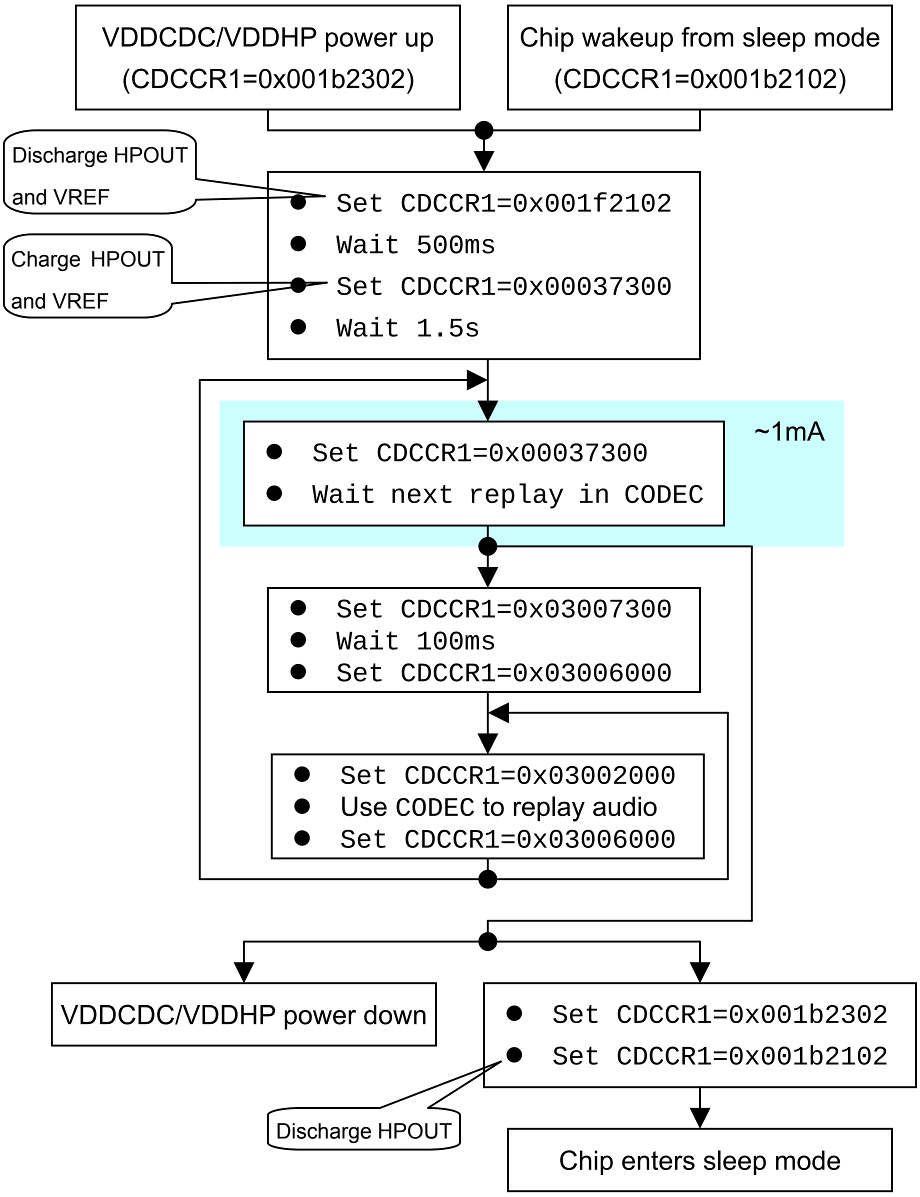
1. Set CDCCR1 = 0x00033302, which charge HPOUT to Vm
2. Set CDCCR1 = 0x001b2302 and start count the time
3. When HPOUT = Vm – 0.1V, stop count the time. (caution: please notice the voltage measurement

instrument causes leakage)

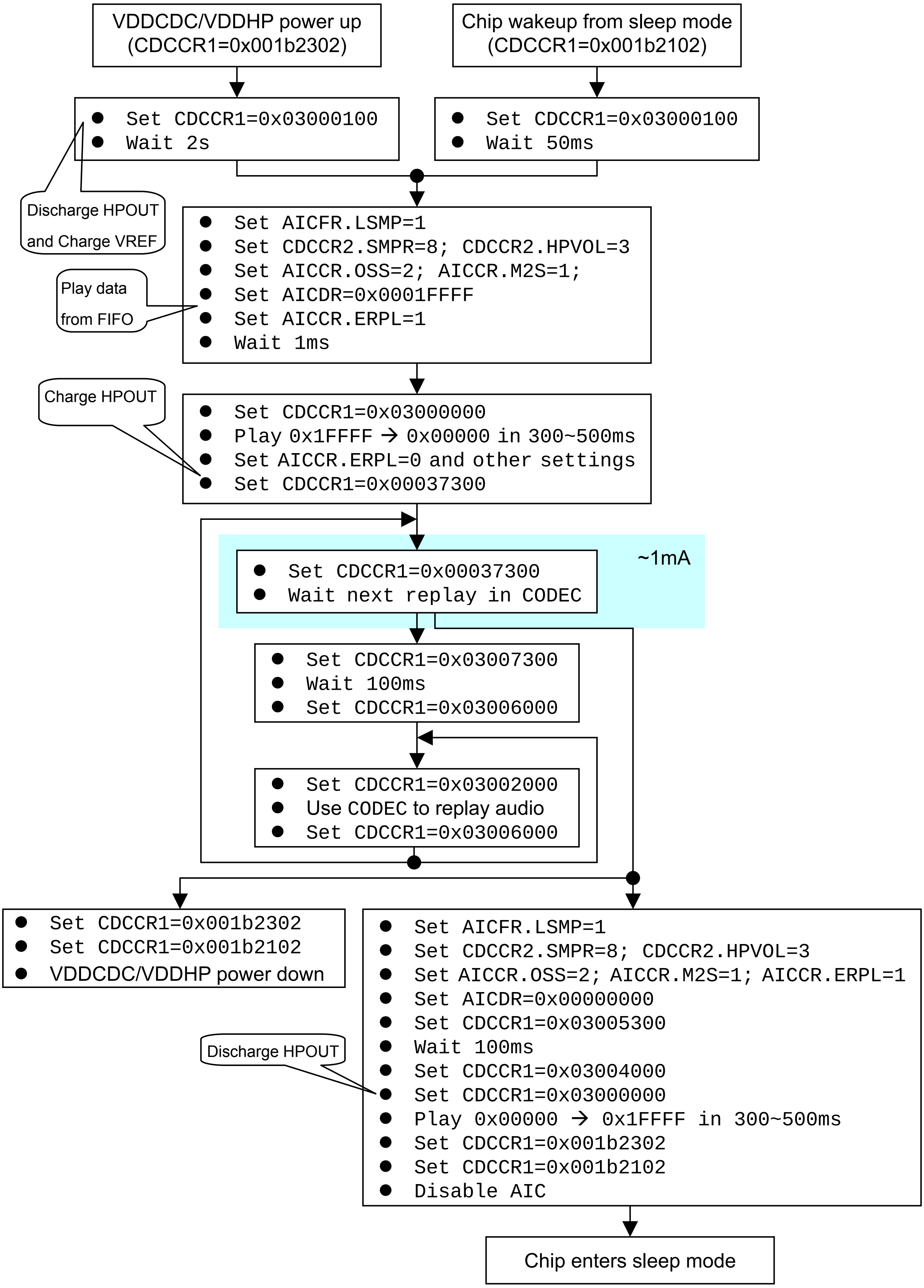
TL01 is the time counted.

This measurement is not needed if user doesn't use complex anti-pop approach.

14.3.4.3 Simple-slow anti-pop approach


VDDCDC/VDDHP power up
(CDCCR1=0x001b2302)
Chip wakeup from sleep mode
(CDCCR1=0x001b2102)
Discharge HPOUT and VREF
Charge HPOUT and VREF
Set CDCCR1=0x001f2102
Wait 500ms
Set CDCCR1=0x00037300
Wait 1.5s
~1mA
Set CDCCR1=0x00037300
Wait next replay in CODEC
Set CDCCR1=0x03007300
Wait 100ms
Set CDCCR1=0x03006000
Set CDCCR1=0x03002000
Use CODEC to replay audio
Set CDCCR1=0x03006000
VDDCDC/VDDHP power down
Set CDCCR1=0x001b2302
Set CDCCR1=0x001b2102
Discharge HPOUT
Chip enters sleep mode

14.3.4.4 Simple-fast anti-pop approach



VDDCDC/VDDHP power up (CDCCR1=0x001b2302)

- Set CDCCR1=0x03000100
- Wait 2s

Chip wakeup from sleep mode (CDCCR1=0x001b2102)

- Set CDCCR1=0x03000100
- Wait 50ms

- Set AICFR.LSMP=1
- Set CDCCR2.SMPR=8; CDCCR2.HPVOL=3
- Set AICCR.OSS=2; AICCR.M2S=1;
- Set AICDR=0x0001FFFF
- Set AICCR.ERPL=1
- Wait 1ms

- Set CDCCR1=0x03000000
- Play 0x1FFFF → 0x00000 in 300~500ms
- Set AICCR.ERPL=0 and other settings
- Set CDCCR1=0x00037300

- Set CDCCR1=0x00037300
- Wait next replay in CODEC

~1mA

- Set CDCCR1=0x03007300
- Wait 100ms
- Set CDCCR1=0x03006000

- Set CDCCR1=0x03002000
- Use CODEC to replay audio
- Set CDCCR1=0x03006000

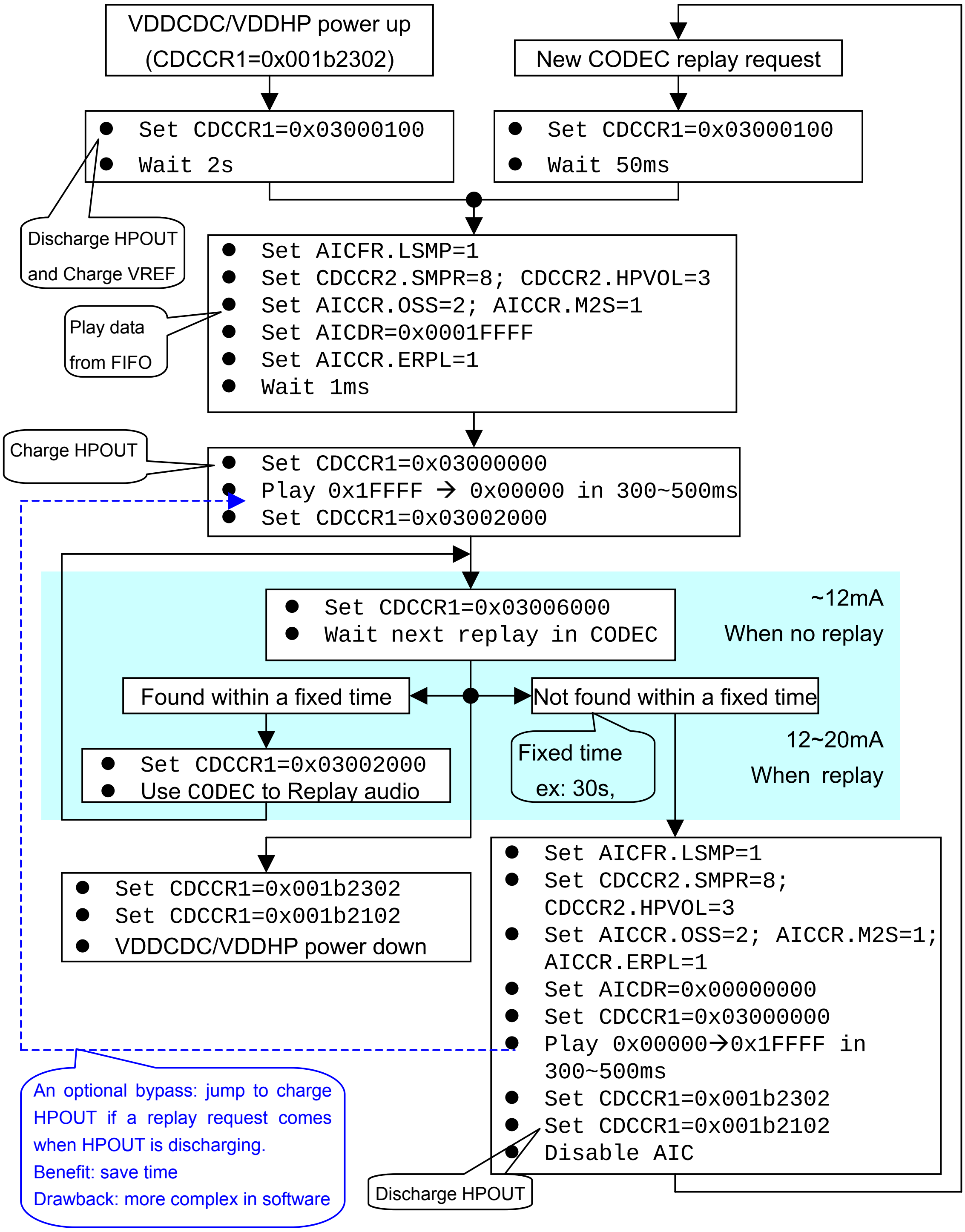
- Set CDCCR1=0x001b2302
- Set CDCCR1=0x001b2102
- VDDCDC/VDDHP power down

- Set AICFR.LSMP=1
- Set CDCCR2.SMPR=8; CDCCR2.HPVOL=3
- Set AICCR.OSS=2; AICCR.M2S=1; AICCR.ERPL=1
- Set AICDR=0x00000000
- Set CDCCR1=0x03005300
- Wait 100ms
- Set CDCCR1=0x03004000
- Set CDCCR1=0x03000000
- Play 0x00000 → 0x1FFFF in 300~500ms
- Set CDCCR1=0x001b2302
- Set CDCCR1=0x001b2102
- Disable AIC

Chip enters sleep mode

14.3.4.5 Improved anti-pop approach

There is a small pop (about 20~30mV voltage change in HPOUT) when CDCCR1 changed from 0x00033302 to 0x03002000. This approach eliminates this pop.


VDDCDC/VDDHP power up
(CDCCR1=0x001b2302)
New CODEC replay request
● Set CDCCR1=0x03000100
● Wait 2s
● Set CDCCR1=0x03000100
● Wait 50ms
Discharge HPOUT and Charge VREF
● Set AICFR.LSMP=1
● Set CDCCR2.SMPR=8; CDCCR2.HPVOL=3
● Set AICCR.OSS=2; AICCR.M2S=1
● Set AICDR=0x0001FFFF
● Set AICCR.ERPL=1
● Wait 1ms
Play data from FIFO
Charge HPOUT
● Set CDCCR1=0x03000000
● Play 0x1FFFF → 0x00000 in 300~500ms
● Set CDCCR1=0x03002000
● Set CDCCR1=0x03006000
● Wait next replay in CODEC
~12mA
When no replay
Found within a fixed time
Not found within a fixed time
● Set CDCCR1=0x03002000
● Use CODEC to Replay audio
Fixed time ex: 30s,
12~20mA
When replay
● Set CDCCR1=0x001b2302
● Set CDCCR1=0x001b2102
● VDDCDC/VDDHP power down
● Set AICFR.LSMP=1
● Set CDCCR2.SMPR=8; CDCCR2.HPVOL=3
● Set AICCR.OSS=2; AICCR.M2S=1; AICCR.ERPL=1
● Set AICDR=0x00000000
● Set CDCCR1=0x03000000
● Play 0x00000→0x1FFFF in 300~500ms
● Set CDCCR1=0x001b2302
● Set CDCCR1=0x001b2102
● Disable AIC
An optional bypass: jump to charge HPOUT if a replay request comes when HPOUT is discharging.
Benefit: save time
Drawback: more complex in software
Discharge HPOUT

14.3.4.6 Complex anti-pop approach

14.4 Timing parameters

14.5 AC & DC parameters

15 SAR A/D Controller

15.1 Overview

The A/D embedded in this processor is a CMOS low-power dissipation 12bit SAR analog to digital converter. It operates with 3.3/1.8V power supply. Circuits needed by touch screen function and battery voltage measurement are also included.

The SAR A/D controller is dedicated to control A/D to work at three different modes: Touch Screen (measure pen position and pen down pressure), Battery (check the battery power), and SADCIN (external ADC input). Touch Screen can transfer the data to memory though the DMA or CPU. Battery and SADCIN can transfer the data to memory though CPU.

Features:

- 6 Channel
- Resolution: 12-bit
- Integral nonlinearity: ±0.5 LSB
- Differential nonlinearity: ±0.4 LSB
- Resolution/speed: up to12bit 187.5ksps
- Max Frequency: 6.0MHz
- Power-down current: 1uA
- Support touch screen measurement (Through pin XP, XN, YP, YN)
- Support voltage measurement (Through pin PBAT)
- Support external SAR-ADC input (Through pin SADCIN)
- Separate Channel Conversion Mode
- Single-end and Differential Conversion Mode
- Auto X/Y, X/Y/Z and X/Y/Z1/Z2 position measurement

15.2 Pin Description

Table 15-1 SADC Pins Description

Name	I/O	Description
XN	AI	Touch screen analog differential X- position input
YN	AI	Touch screen analog differential Y- position input
XP	AI	Touch screen analog differential X- position input
YP	AI	Touch screen analog differential Y- position input
ADIN0 (PBAT)	AI	Analog input for VBAT measurement.
ADIN1 (SADCIN)	AI	External SAR-ADC input

15.3 Register Description

In this section, we will describe the registers in SAR A/D controller. Following table lists all the registers definition. All register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
ADENA	ADC Enable Register	RW	0x00	0x10070000	8
ADCFG	ADC Configure Register	RW	0x0002002C	0x10070004	32
ADCTRL	ADC Control Register	RW	0x00	0x10070008	8
ADSTATE	ADC Status Register	RW	0x00	0x1007000C	8
ADSAME	ADC Same Point Time Register	RW	0x0000	0x10070010	16
ADWAIT	ADC Wait Time Register	RW	0x0000	0x10070014	16
ADTCH	ADC Touch Screen Data Register	RW	0x00000000	0x10070018	32
ADBDAT	ADC PBAT Data Register	RW	0x0000	0x1007001C	16
ADSDAT	ADC SADCIN Data Register	RW	0x0000	0x10070020	16
ADDPIN	ADC IP DATA PIN	R	0x0000	0x10070024	16

15.3.1 ADC Enable Register (ADENA)

The register ADENA is used to trigger A/D to work.

ADENA **0x10070000**

Bit	7	6	5	4	3	2	1	0
	ADEN	Reserved				TCHEN	PBATEN	SADCINEN
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	ADEN[*1]	A/D Enable Control. (Only used in test mode) Check the channel function of ADC. When A/D finish sampling the data, ADEN will be cleared by hardware auto. 0: disable 1: enable	RW
6:3	Reserved	These bits always read 0, and written are ignored.	R
2	TCHEN[*2]	Touch Screen Enable Control. 0: disable 1: enable	RW
1	PBATEN[*2]	PBAT Enable Control. Sample the voltage of battery, PBATEN can be set to 1 no matter TCHEN is disable or enable, and when the voltage of battery is ready. PBATEN will be cleared by hardware auto.	RW
0	SADCINEN[*2]	SADCIN Enable Control. Sample SADCIN, SADCINEN can be set to 1 no matter TCHEN is disable or enable, and when SADCIN is ready, SADCINEN will be cleared by hardware auto.	RW

Note:

*1. When ADEN is set to 1, other bits cannot be set to 1 at the same time. This mode only used in test mode.

*2. TCHEN, PBATEN and SADCINEN can be set to 1 at the same time. The priority of the three mode is SADCIN > PBAT > TCH.

15.3.2 ADC Configure Register (ADCFG)

The register ADCFG is used to configure the A/D.

ADCFG **0x10007004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SPZZ	EX_IN	Reserved											CLKOUTD	NUM		DMA_EN	XYZ		SNUM			CLKDIV					BAT_MD	CMD			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>31</td><td>SPZZ[*1]</td><td>The $X_dY_dZ_mZ_n$ of different point measure can be different.
But the $X_dY_dZ_mZ_n$ of the same point measure can be same or different.
0:The $X_dY_dZ_mZ_n$ of the same point measure is all the same.
(X_dY_dZ1Z2, X_dY_dZ1Z2, X_dY_dZ1Z2, X_dY_dZ1Z2 … X_dY_dZ1Z2)
1: The $X_dY_dZ_mZ_n$ of the same point measure maybe different.
(X_dY_dZ1Z2, X_dY_dZ3Z4, X_dY_dZ3Z4, X_dY_dZ1Z2 … X_dY_dZ1Z2)</td><td>RW</td></tr>
<tr><td>30</td><td>EX_IN</td><td>Choose external driver or internal driver.
0: X_sY_s or X_sY_sZ
1: X_dY_d or X_dY_dZ
It is no use for $X_dY_dZ_mZ_n$.

It is no use when ADCFG.XYZ = 10.
It is useful when ADCFG.XYZ = 00/01.</td><td>RW</td></tr>
<tr><td>29:19</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>18:16</td><td>DNUM</td><td>This will set which is the sampled data is the virtual value.
Default: = 3'b010
<table>
<tr><td>DNUM</td><td>Number</td></tr>
<tr><td>3'b000</td><td>Reserved</td></tr>
<tr><td>3'b001</td><td>The virtual value is the 2nd sampled data</td></tr>
<tr><td>3'b010</td><td>The virtual value is the 3rd sampled data</td></tr>
<tr><td>3'b011</td><td>The virtual value is the 4th sampled data</td></tr>
<tr><td>3'b100</td><td>The virtual value is the 5th sampled data</td></tr>
<tr><td>3'b101</td><td>The virtual value is the 6th sampled data</td></tr>
<tr><td>3'b110</td><td>The virtual value is the 7th sampled data</td></tr>
<tr><td>3'b111</td><td>The virtual value is the 8th sampled data</td></tr>
</table></td><td>RW</td></tr>
<tr><td>15</td><td>DMA_EN</td><td>When A/D is used as Touch Screen (CMD=1100), DMA_EN is used as follows:
0: The sample data is read by CPU
1: The sample data is read by DMA</td><td>RW</td></tr>
<tr><td>14:13</td><td>XYZ</td><td>When A/D is used in Touch Screen mode (CMD=1100), XYZ is used as</td><td>RW</td></tr>
</table>

<table>
<tr><td></td><td></td><td>follows:
<table>
<tr><td>XYZ</td><td>Measure (EX_IN = 1)</td><td>Measure (EX_IN = 0)</td></tr>
<tr><td>00</td><td>$X_d \rightarrow Y_d$</td><td>$X_s \rightarrow Y_s$</td></tr>
<tr><td>01</td><td>$X_d \rightarrow Y_d \rightarrow Z_s$</td><td>$X_s \rightarrow Y_s \rightarrow Z_s$</td></tr>
<tr><td>10</td><td>$X_d \rightarrow Y_d \rightarrow Z1_d \rightarrow Z2_d$
or
$X_d \rightarrow Y_d \rightarrow Z3_d \rightarrow Z4_d$</td><td>$X_d \rightarrow Y_d \rightarrow Z1_d \rightarrow Z2_d$
or
$X_d \rightarrow Y_d \rightarrow Z3_d \rightarrow Z4_d$</td></tr>
<tr><td>11</td><td>Reserved</td><td>Reserved</td></tr>
</table></td><td></td></tr>
<tr><td>12:10</td><td>SNUM</td><td>The number of repeated sampling one point. When A/D is used as Touch Screen (CMD=1100), SNUM is used as follows:
<table>
<tr><td>SNUM</td><td>Number</td></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>010</td><td>3</td></tr>
<tr><td>011</td><td>4</td></tr>
<tr><td>100</td><td>5</td></tr>
<tr><td>101</td><td>6</td></tr>
<tr><td>110</td><td>8</td></tr>
<tr><td>111</td><td>9</td></tr>
</table></td><td>RW</td></tr>
<tr><td>9:5</td><td>CLKDIV</td><td>A/D converter frequency.
A/D works at the frequency between 500KHz and 6MHz.
If CLKDIV =N, Then the frequency divide number = 12MHz/N+1.
0< N < 24</td><td>RW</td></tr>
<tr><td>4</td><td>BAT_MD</td><td>When AD is used as PBAT measure the following channel mode can be chose to measure the battery power.
0: PBAT (full battery voltage>=2.5V)
1: PBAT (full battery voltage<2.5V)</td><td>RW</td></tr>
</table>

<table>
<tr><td>3:0</td><td>CMD</td><td>CMD is used to choose the current sample command when adc_en_r is set to 1 (single channel test mode).
<table>
<tr><td>CMD</td><td>Function</td><td>Reference mode</td></tr>
<tr><td>0000</td><td>Measure X Position (X-plate is driven by external DC power)</td><td>Single-end</td></tr>
<tr><td>0001</td><td>Measure Y Position (Y-plate is driven by external DC power)</td><td>Single-end</td></tr>
<tr><td>0010</td><td>Measure X Position</td><td>Differential</td></tr>
<tr><td>0011</td><td>Measure Y Position</td><td>Differential</td></tr>
<tr><td>0100</td><td>Measure Z1 Position</td><td>Differential</td></tr>
<tr><td>0101</td><td>Measure Z2 Position</td><td>Differential</td></tr>
<tr><td>0110</td><td>Measure Z3 Position</td><td>Differential</td></tr>
<tr><td>0111</td><td>Measure Z4 Position</td><td>Differential</td></tr>
<tr><td>1000</td><td>Measure Touch Pressure Z</td><td>Single-end</td></tr>
<tr><td>1001</td><td>Measure PBAT (>=2.5V)</td><td>Single-end</td></tr>
<tr><td>1010</td><td>Measure PBAT (<2.5V)</td><td>Single-end</td></tr>
<tr><td>1011</td><td>Measure SADCIN</td><td>Single-end</td></tr>
<tr><td>1100</td><td colspan="2">INT_PEN enable, this mode is the default value.</td></tr>
<tr><td>1101~1111</td><td colspan="2">Reserved</td></tr>
</table>
</td><td>RW</td></tr>
</table>

Note[*1]:

X_s, Y_s, Z_s means the reference mode of X, Y, Z is single-end mode.

X_d, Y_d, $Z1_d$, $Z2_d$, $Z3_d$, Z4d means the reference mode of X, Y, Z1, Z2, Z3, Z4 is differential mode.

When you measure Xs you need to make sure that X-plate is driven by external DC power.

When you measure Ys you need to make sure that Y-plate is driven by external DC power.

15.3.3 ADC Control Register (ADCTRL)

The register ADCTRL is used to control A/D to work.

ADCTR **0x10070008**

Bit	7	6	5	4	3	2	1	0
	Reserved			PENDM	PENUM	DTCHM	DRDYM	SRDYM
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	These bits always read 0, and written are ignored.	R
4	PENDM	Pen down interrupt mask. 0= enabled 1= masked	RW

3	PENUM	Pen up interrupt mask. 0= enabled 1= masked	RW
2	DTCHM	Touch Screen Data Ready interrupt mask. 0= enabled 1= masked	RW
1	DRDYM	Data ready interrupt mask. (ADCEN = 1) PBAT data ready interrupt mask. 0= enabled 1= masked	RW
0	SRDYM	SADCIN Data Ready interrupt mask. 0= enabled 1= masked	RW

15.3.4 ADC Status Register (ADSTATE)

The register ADCSTATE is used to keep the status of A/D.

ADSTS **0x1007000C**

Bit	7	6	5	4	3	2	1	0
	Reserved			PEND	PENU	DTCH	DRDY	SRDY
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	These bits always read 0, and written are ignored.	R
4	PEND	Pen down interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
3	PENU	Pen up interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
2	DTCH	Touch screen data ready interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW

1	DRDY	Data ready interrupt flag when ADCEN = 1. PBAT data ready interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
0	SRDY	SADCIN Data ready interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW

15.3.5 ADC Same Point Time Register (ADSAME)

The register ADSAME is used to store the interval time between repeated sampling the same point. The clock frequency of the counter is 12M/128.

ADSAME **0x10070010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCNT															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15.3.6 ADC Wait Pen Down Time Register (ADWAIT)

The register ADWAIT is used to store the interval time of wait pen down. And the register can be used as the interval time among the different point. The clock frequency of the counter is 12M/128.

ADDIFF **0x10070014**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WCNT															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15.3.7 ADC Touch Screen Data Register (ADTCH)

The read-only ADTCH is corresponded to 2x32 bit FIFO, it keep the sample data for touch screen. 0~11 bits are data, 15 bit is data type. 16~27 bits are data, 31 bit is data type. When write to the register, DATA will be clear to 0.

ADTCH **0x10070018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	TYPE1	Reserved	TDATA1	TYPE0	Reserved	TDATA0
RST	0	0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0	0 0 0	0 0 0 0 0 0 0 0 0 0 0 0

Bits	Name	Description	RW
31	TYPE1	Type of the Touch Screen Data1. When A/D is used as Touch Screen, ADCFG.XYZ=10 TYPE1=1: $X_d \rightarrow Y_d \rightarrow Z1 \rightarrow Z2$ TYPE1=0: $X_d \rightarrow Y_d \rightarrow Z3 \rightarrow Z4$ When A/D is used as Touch Screen, ADCFG.XYZ=00 or XYZ=01, TYPE1=0.	RW
30:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	TDATA1	The concert data of touch screen A/D.	RW
15	TYPE0	Type of the Touch Screen Data2. When A/D is used as Touch Screen, ADCFG.XYZ=10 TYPE0=1: $X_d \rightarrow Y_d \rightarrow Z1 \rightarrow Z2$ TYPE0=0: $X_d \rightarrow Y_d \rightarrow Z3 \rightarrow Z4$ When A/D is used as Touch Screen, ADCFG.XYZ=00 or XYZ=01, TYPE0=0.	RW
14:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	TDATA0	The concert data of touch screen A/D.	RW

Note:

(1) When A/D is used as Touch Screen, EX_IN=0 and ADCFG.XYZ=00.

The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_s	0	000	X_s

(2) When A/D is used as Touch Screen, EX_IN=1 and ADCFG.XYZ=00.

The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_d	0	000	X_d

(3) When A/D is used as Touch Screen, EX_IN=0 and ADCFG.XYZ=01.

The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_s	0	000	X_s

0	000	000000000000	0	000	Z_s

Users need to read twice to get the whole data. The first time reading gets the data Y_s and X_s. The second time reading gets the data Z_s. The relation between “touch pressure” and “Z_s” are inverse ratio.

(4) When A/D is used as Touch Screen, EX_IN=1 and ADCFG.XYZ=01
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_d	0	000	X_d
0	000	000000000000	0	000	Z_s

Users need to read twice to get the whole data. The first time reading gets the data Y_d and X_d. The second time reading gets the data Z_s. The relation between “touch pressure” and “Z_s” are inverse ratio.

(5) When A/D is used as Touch Screen, ADCFG.XYZ=11,TYPE=1
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
1	000	Y_d	1	000	X_d
1	000	$Z2_d$	1	000	$Z1_d$

Users need to read twice to get the whole data. The first time reading gets the data Y_d and X_d. The second time reading gets the data $Z2_d$ and $Z1_d$.

The touch pressure measurement formula is as follows: (You can use formula 1 or formula 2.)

$$R_{\text{TOUCH}} = R_{\text{X-Plate}} \bullet \frac{\text{X-Position}}{4096}\left(\frac{Z_2}{Z_1}-1\right) \qquad (1)^{*1}$$

$$R_{\text{TOUCH}} = \frac{R_{\text{X-Plate}} \bullet \text{X-Position}}{4096}\left(\frac{4096}{Z_1}-1\right) - R_{\text{Y-Plate}} \bullet \left(1-\frac{\text{Y-Position}}{4096}\right) \qquad (2)^{*1}$$

(6) When A/D is used as Touch Screen, ADCFG.XYZ=11,TYPE=0
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_d	0	000	X_d
0	000	$Z4_d$	0	000	$Z3_d$

Users need to read twice to get the whole data. The first time reading gets the data Y_d and X_d. The second time reading gets the data $Z4_d$ and $Z3_d$.

The touch pressure measurement formula is as follows: (You can use formula 3 or formula 4.)

$$R_{\text{TOUCH}} = R_{\text{Y-Plate}} \bullet \frac{\text{Y-Position}}{4096}\left(\frac{Z_4}{Z_3} - 1\right) \qquad (3)^{*1}$$

$$R_{\text{TOUCH}} = \frac{R_{\text{Y-Plate}} \bullet \text{Y-Position}}{4096}\left(\frac{4096}{Z_3} - 1\right) - R_{\text{X-Plate}} \bullet \left(1 - \frac{\text{X-Position}}{4096}\right) \qquad (4)^{*1}$$

Note[*1]:

To determine pen or finger touch, the pressure of the touch needs to be determined. Generally, it is not necessary to have very high performance for this test; therefore, the 8-bit resolution mode is recommended (however, calculations will be shown here are in 12-bit resolution mode).

$R_{\text{X-plate}}$: Total X-axis resistor value (about 200Ω~ 600Ω)
$R_{\text{Y-plate}}$: Total Y-axis resistor value (about 200Ω~ 600Ω)
X-Position: X-axis voltage sample value
Y-Position: Y-axis voltage sample value
Z1, Z2: Z1, Z2 voltage sample value
Z3, Z4: Z3, Z4 voltage sample value

15.3.8 ADC PBAT Data Register (ADBDAT)

The read-only ADBDAT is a 16-bit register, it keep the sample data of both “PBAT mode” and “Single channel check” mode. 0~11 bits are data.

ADBDAT **0x1007001C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				BDATA											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	BDATA	Data of A/D convert when ADCEN = 1. Data of PBAT A/D convert. When write to the register, DATA will be clear to 0.	RW

When ADCCFG.BAT_MD = 0 (full battery voltage>=2.5V), the measured voltage V_{BAT} is as follows:

$$V_{\text{BAT}} = \frac{\text{BDATA}}{4096} \bullet 7.5V * 0.986 + 0.033V$$

When ADCCFG.BAT_MD = 1 (full battery voltage<2.5V), the measured voltage V_{BAT} is as follows:

$$V_{BAT} = \frac{BDATA}{4096} \bullet 2.5V$$

It is recommended to connect a capacitance of about 0.1uF near to pin ADIN0 to have a more stable battery measurement and better ESD protection.

15.3.9 ADC SADCIN Data Register (ADSDAT)

The read-only ADSDAT is a 16-bit register, it keep the sample data. 0~11 bits are data.

ADDAT **0x10070020**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				SDATA											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	SDATA	Data of SADCIN A/D convert. When write to the register, DATA will be clear to 0.	RW

15.3.10 ADC IP DATA PIN (ADDPIN)

The read-only ADDPIN is a 16-bit fake register. The transfer data is come from SADC IP output data directly. This is used to check IP data.

ADDPIN **0x10070024**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				DATA											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	DATA	Data of transfer.	R

15.4 SAR A/D Controller Guide

The following describes steps of using SAR-ADC.

15.4.1 Single Operation (only used as a test mode to check the channel function)

(1) Set ADTCTL to 0x1f to mask all the interrupt of SADC.
(2) Set ADCFG.CMD to choose one CMD;(0000~1011)
(3) Set ADCFG.CLKDIV to set A/D clock frequency;
(4) Set ADTCTL.PDEN to 0;
(5) Set ADENA.ADEN to 1 to start A/D;
(6) When ADSTATE.DRDY to 1, you can read the sample data from ADBDAT and ADENA.ADEN will be set to 0 auto.

15.4.2 A simple Touch Screen Operation

(Pen Down → Sample some data of several points → Pen Up)

(1) Set ADTCTL to 0x1f to mask all the interrupt of SADC.
(2) Set DMA_EN to choose whether to use DMA to read the sample data out or to use CPU to read the sample data out;
(3) Set ADCFG.SPZZ, ADCFG.EX_IN and ADCFG.XYZ to choose sample mode
    1: X_s→Y_s (Single-end X→Single-end Y).
    2: X_d→Y_d (Differential X→Differential Y).
    3: X_s→Y_s→Z_s (Single-end X→Single-end Y→Single-end Z)
    4: X_d→Y_d→Z_s (Differential X→Differential Y→Single-end Z)
    5:X_d→Y_d→$Z1_d$→$Z2_d$ or X_d→Y_d→$Z3_d$→$Z4_d$ (Reference register ADCFG.SPZZ)
    (Differential X→Differential Y→Differential Z1→Differential Z2 or
      Differential X→Differential Y→Differential Z3→Differential Z4,)
(4) Set ADCFG.CLKDIV to set A/D clock frequency;
(5) Set ADWAIT to decide the wait time of pen down and the interval time between sampling different points. This time delay is necessary because when pen is put down or pen position change, there should be some time to wait the pen down signal to become stable.
(6) Set ADSAME to decide the interval time between repeated sampling the same point. User can repeat sampling one point to get the most accurate data.
(7) Set ADTCTL.PENDM to 0 to enable the pen down interrupt of touch panel;
(8) Set ADENA.TCHEN to 1 to start touch panel;
(9) When pen down interrupt is happened, you should set ADTCTL.PENDM to 1 and clear ADSTATE.PEND to close pen down interrupt. Then you should clear ADSTATE.PENDU and set ADTCTL.PENUM to 0 to enable pen up interrupt.
(10) When pen down interrupt is happened, the SAR ADC is sampling data. When ADSTATE.DTCH to 1, user must read the sample data from ADTCH. The SAR ADC will not sample the next point until the whole data of the one point are read (no matter by CPU or DMA). If ADCFG.XYZ is mode one and mode two, user only needs to read once to get the whole data. In other modes, user needs to read twice to get the whole data.
(11) Repeat 10 till pen up interrupt happened.

(12) When pen up interrupt is happened, you should set ADTCTL.PENUM to 1 and clear ADSTATE.PENU. Then you should clear ADSTATE.PENDD and set ADTCTL.PENPM to 0 to enable pen down interrupt.
(13) Wait pen down interrupt and repeat from 9.
(14) When you want to shut down the touch screen, user can set the ADENA.TCHEN to 0. If the last point is not sampled completely, user can abandon it.

15.4.3 PBAT Sample Operation

(1) Set ADCFG.CLKDIV to set A/D clock frequency;
(2) Set ADCFG.CH_MD to choose PBAT test mode channel.
(3) Set ADENA.PBATEN to 1 to enable the channel.
(4) When ADSTATE.DRDY = 1, you can read the sample data from ADPBAT. And the PBATEN will be set to 0 auto.

15.4.4 SADCIN Sample Operation

(1) Set ADCFG.CLKDIV to set A/D clock frequency;
(2) Set ADENA.SADCINEN to 1 to enable the channel.
(3) When ADSTATE. SRDY = 1, you can read the sample data from ADSDAT. And the SADCINEN will be set to 0 auto.

Note:

Touch Screen mode can be interrupt by the PBAT and SADCIN mode. And the priority is SADCIN > PBAT > TOUCH. If SADCINEN or PBATEN is set to 1 before or at the same time with TCHEN, SAR ADC will first work in SADCIN mode then in PBAT mode at last in touch screen mode. If SADCINEN and PBATEN are set to 1 after the TCHEN, the SAR ADC will work in touch screen mode first and finish sampling the same point completely then turn to the SADCIN or PBAT mode. And return to touch screen mode.

15.4.5 Use TSC to support keypad

SADC TSC function can apply to a keypad, if touch screen is not used. Suppose the keypad is a NxM matrix, where X direction has N key columns and Y direction has M key rows. Kij is used to indicate the key in ith column from left to right and jth row from bottom to top, where i=0~(N-1) and j=0~(M-1). Figure 15-1 is a 6x5 keypad circuit. The blue color is for X direction network and pink color is for Y. The networks are composed by resistors and metal line. These two networks should be connected to SADC 4 pins: XP/XN/YP/YN as illustrated in the figure. The gray circle is the key. When no key pressing, X network and Y network is open circuit. When a key is pressed, the X network and Y network is shorted under the key position.

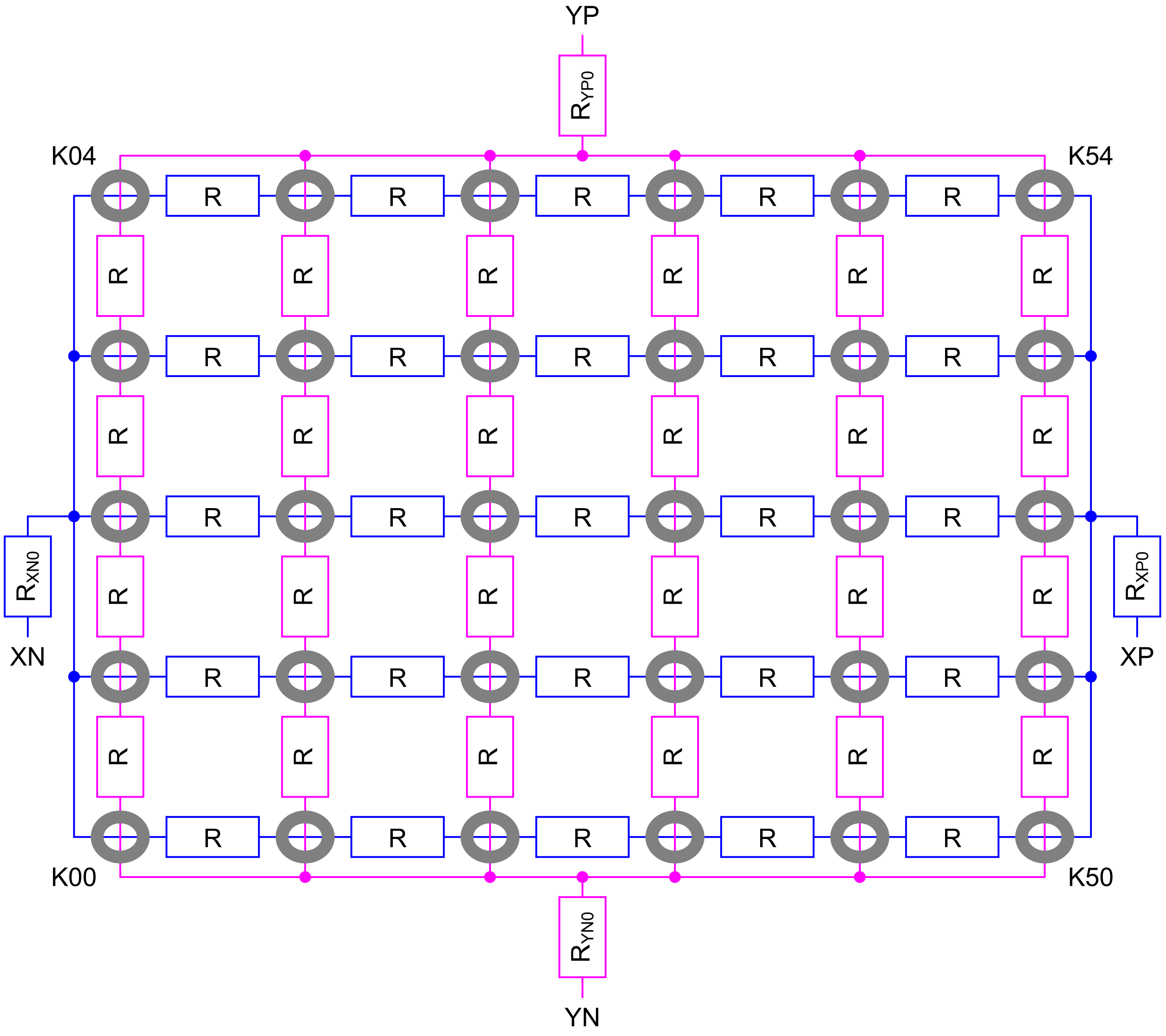


Figure 15-1 6x5 keypad circuit

When SADC is in waiting for pen-down status (C=1100), the equivalent circuit is show in Figure 15-2. When the key is not pressed, XP is open and the PEN is pulled to VDDADC, which is logic 1. When the key Kij is pressed, the circuit is: VDDADC→(10kΩ resistor)→R_{XP}→R_{YN}→VSSADC.

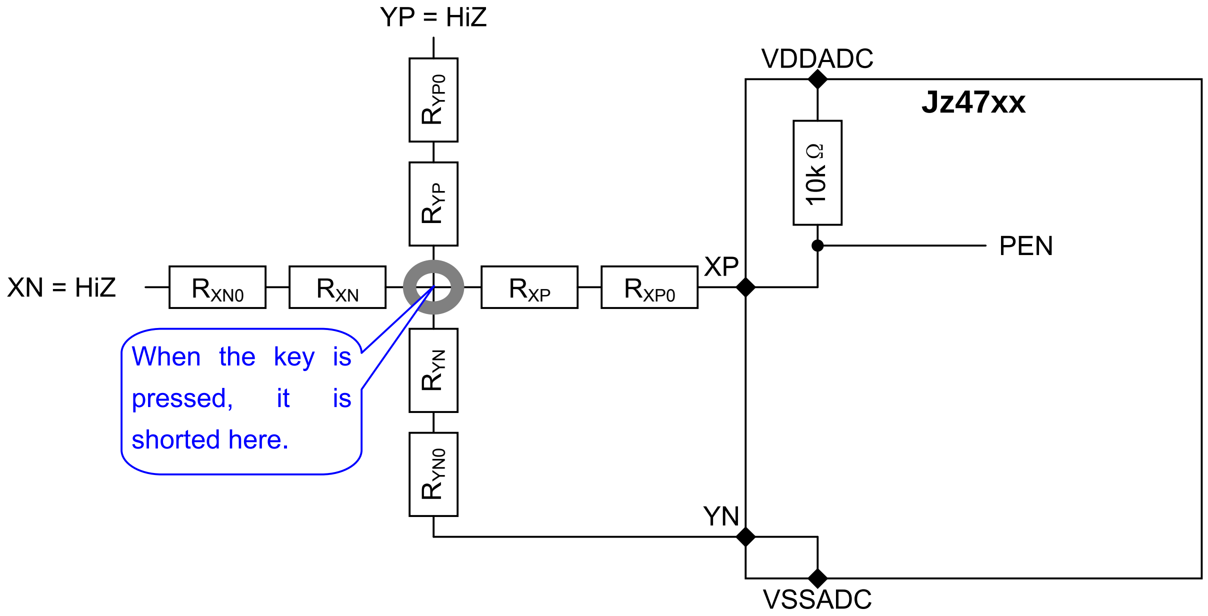

YP = HiZ
VDDADC
Jz47xx
R_{YP0}
R_{YP}
10k Ω
PEN
XP
XN = HiZ
R_{XN0}
R_{XN}
R_{XP}
R_{XP0}
When the key is pressed, it is shorted here.
R_{YN}
R_{YN0}
YN
VSSADC

Figure 15-2 Wait for pen-down (C=1100) circuit

Where

$$R_{XP} = \frac{(N-1)^2 - i^2}{M \times (N-1-i) + 2i} \times R$$

$$R_{YN} = \frac{j \times (2M-2-j)}{N \times j + 2M - 2 - 2j} \times R$$

To ensure logic 0 at PEN in this case, following formula should be obeyed.

$$R_{XP} + R_{YN} + R_{XP0} + R_{YN0} \le 3k\Omega \qquad (1)$$

It is suggested the value of N and M is as close to each other as possible. For N=2~20, M=2~20 and M=(N-1, N or N+1), we found

$$R_{XP} + R_{YN} < 2.7 \times R \qquad (2)$$

After key pressing is found, the key Kij location, columns and row, should be measured by using C=0010 and C=0011 respectively. The equivalent circuits are show in Figure 15-3 and Figure 15-4, where

$$R_{X0} = \frac{N-1}{M-1} \times R$$

$$R_{Y0} = \frac{M-1}{N-1} \times R$$

$$R_{XNi} = i \times R$$

$$R_{XPi} = (N-1-i) \times R$$

$$R_{YNj} = j \times R$$

$$R_{YPj} = (M-1-j) \times R$$

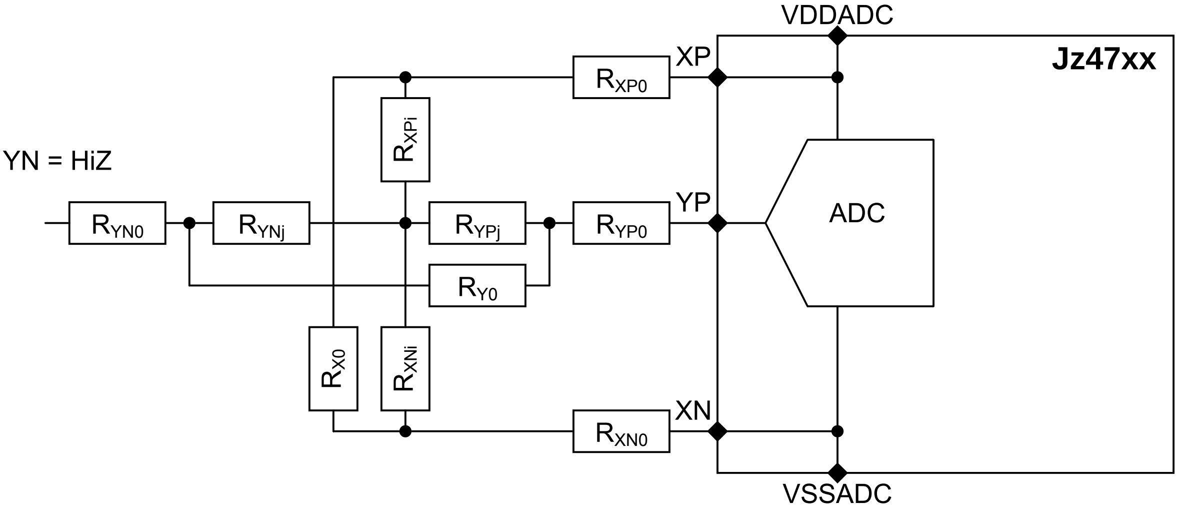

VDDADC
Jz47xx
R_{XP0}
XP
R_{XPi}
YN = HiZ
R_{YN0}
R_{YNj}
R_{YPj}
R_{YP0}
YP
ADC
R_{Y0}
R_{X0}
R_{XNi}
R_{XN0}
XN
VSSADC

Figure 15-3 Measure X-position (C=0010) circuit

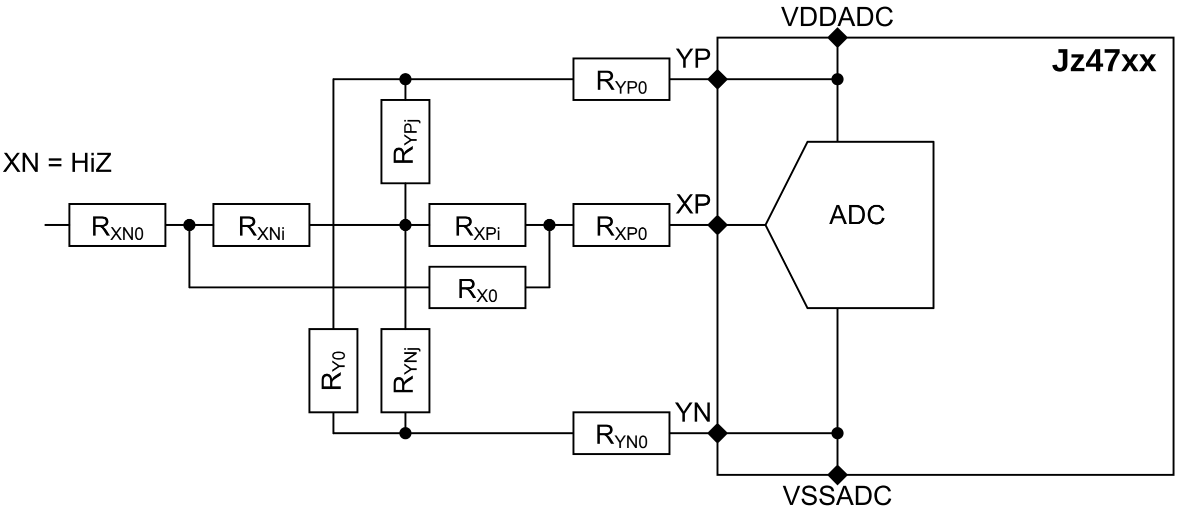

VDDADC
Jz47xx
R_{YP0}
YP
R_{YPj}
XN = HiZ
R_{XN0}
R_{XNi}
R_{XPi}
R_{XP0}
XP
ADC
R_{X0}
R_{Y0}
R_{YNj}
R_{YN0}
YN
VSSADC

Figure 15-4 Measure Y-position (C=0011) circuit

So for Kij pressing, we should get ADC converted number Ni and Nj for i and j respectively.

$$Ni = \frac{R_{XN0} + \frac{i}{M}R}{R_{XN0} + \frac{N-1}{M}R + R_{XP0}} \times 4096$$

$$Nj = \frac{R_{YN0} + \frac{j}{N}R}{R_{YN0} + \frac{M-1}{N}R + R_{YP0}} \times 4096$$

It is required the resistor between XP and XN in case of C=0010, between YP and YN in case of C=0011, must be $\geqslant 200\Omega$ and it better be $\geqslant 500\Omega$. Also consider the requirement in formula (1) and

(2) above, we suggest to put $R_{XP0} = R_{XN0} = R_{YP0} = R_{YN0}$ = 50Ω or 100Ω, put R = 500Ω ~ 1kΩ.

To use the keypad, the software should set:

```
ADENA.TCHEN = 1
ADCFG.EX_IN = 1
ADCFG.XYZ = 00
```

The operation is similar to touch screen.

16 MultiMediaCard/Secure Digital Controller

16.1 Overview

The MultiMediaCard (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications such as electronic toys, organizers, PDAs, smart phones, and so on. The MMC communication is base on an advanced 7 pin serial bus designed to operate in a low voltage range, at medium speed (20 Mbps).

The Secure Digital (SD) card is an evolution of MMC, with an additional 2 pins and the same form factors. It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the MultiMediaCard with some additions. An SD card can be categorized as SD memory or SD I/O card, commonly known as SDIO. A memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard and is faster and capable of higher memory capacity. The SDIO card provides high-speed data I/O with low-power consumption for mobile electronic devices.

Features of the MultimediaCard/Secure Digital Controller include the following:

- Fully compatible with the *MMC System Specification version 3.3*
- Fully compatible with the *SD Memory Card Specification 1.01* and *SD I/O Specification 1.0* with 1 command channel and 4 data channels
- 20-80 Mbps maximum data rate
- Built-in programmable frequency divider for MMC/SD bus
- Maskable hardware interrupt for SDIO interrupt, internal status and FIFO status
- 16-entry x 32-bit built-in data FIFO
- Password protection of cards
- Multi-SD function support including multiple I/O and combined I/O and memory
- Up to 7 I/O functions plus one memory supported on single SDIO card
- IRQ supported enable card to interrupt MMC/SD controller
- Single or multi block access to the card including erase operation
- Stream access to the card
- Supports SDIO read wait, interrupt detection during 1-bit or 4-bit access
- The maximum block length is 2048 bytes

16.2 Block Diagram

MMC/SD Controller Block Diagram

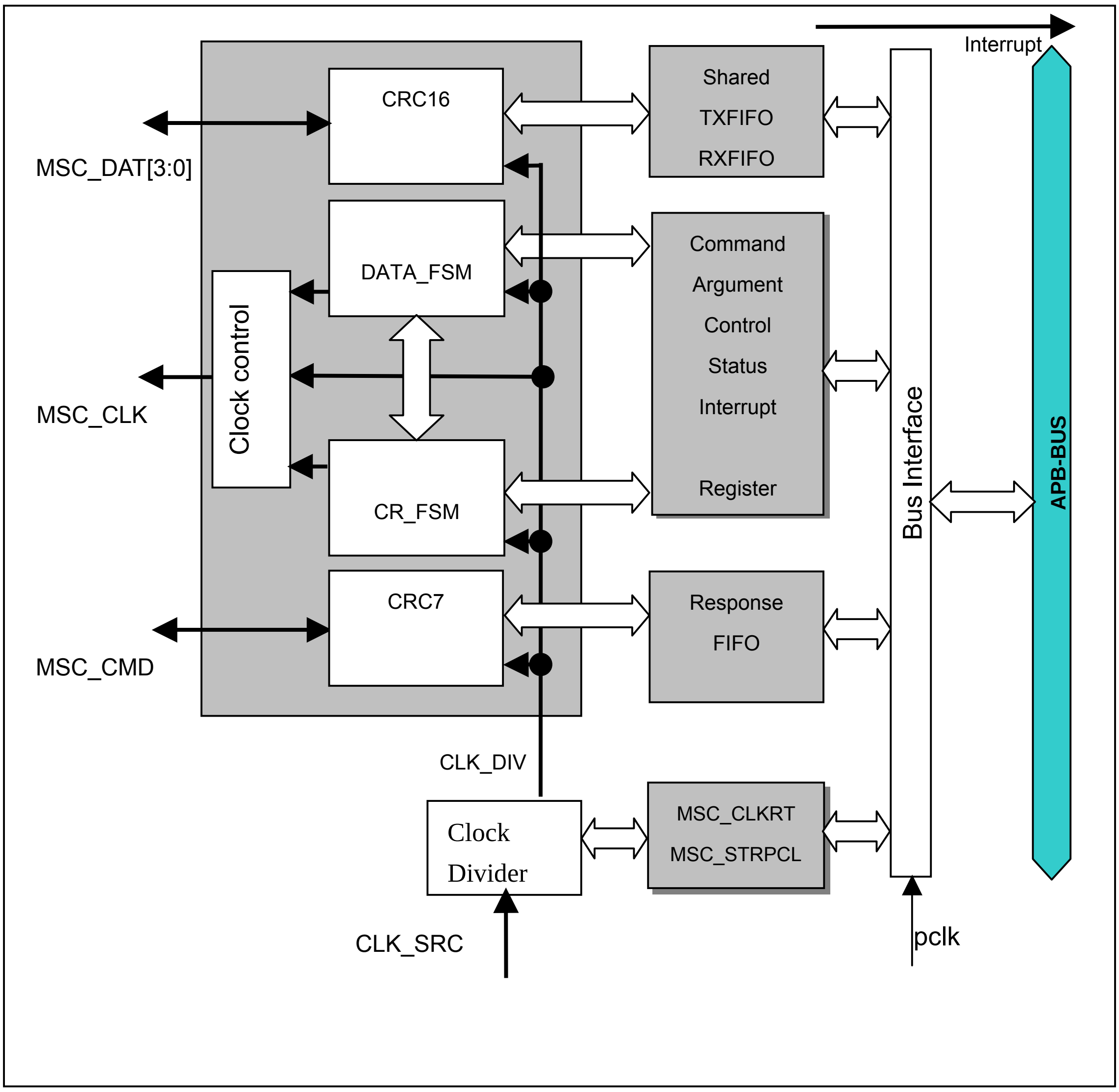
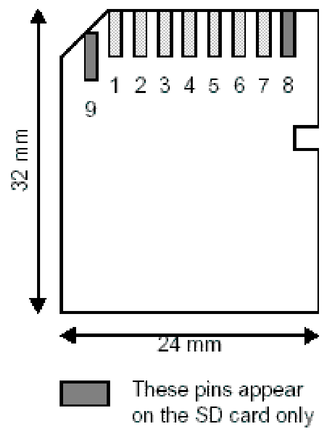

Interrupt
CRC16
Shared
TXFIFO
RXFIFO
MSC_DAT[3:0]
DATA_FSM
Command
Argument
Control
Status
Interrupt
Register
Clock control
MSC_CLK
CR_FSM
Bus Interface
APB-BUS
CRC7
Response
FIFO
MSC_CMD
CLK_DIV
Clock
Divider
MSC_CLKRT
MSC_STRPCL
CLK_SRC
pclk

Figure 16-1 MMC/SD Controller Block Diagram

16.3 MMC/SD Controller Signal I/O Description

The MMC and SD cards are 7- or 9- pin cards that operate as external memory storage. The pin assignment and form factor are shown in Table 16-1.

Table 16-1 MMC/SD Controller Signal Description

Form factor and pinout	Pin number	MMC card	SD card	
			1-bit mode	4-bit mode
 1 2 3 4 5 6 7 8 9 32 mm 24 mm These pins appear on the SD card only	1	Reserved	Not Used	Data Line [Bit 3]
	2	Command/Response (CMD)		
	3	Supply Voltage Ground (Vss1)		
	4	Supply Voltage (Vdd)		
	5	Clock (CLK)		
	6	Supply Voltage Ground (Vss2)		
	7	Data Line [Bit 0]		
	8	▨	Interrupt (IRQ)	Data Line [Bit 1] or Interrupt (IRQ)
	9	▨	ReadWait(RW)	Data Line [Bit 2] or ReadWait (RW)

MSC and the card communication over the CMD and DATA line is base on command and data bit streams which are initiated by a start bit and terminated by a stop bit.

- **Command**: a command is a token, which starts an operation. A command is sent from MSC either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line. Each command token is preceded by a start bit ('0') and succeeded by an end bit ('1'). The total length is 48 bits and protected by CRC bits.

Table 16-2 Command Token Format

Bit position	47	46	[45 : 40]	[39 : 8]	[7 : 1]	0
Width (bits)	1	1	6	32	7	1
Value	0	1	X	X	x	1
Description	Start bit	Transmission bit	Command index	argument	CRC7	End bit

- **Response**: a response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to MSC as an answer to a previously received command. A response is transferred serially on the CMD line. Response tokens have varies coding schemes depending on their content.

- **Data:** data can be transferred from the card to MSC or vice versa. Data is transferred via the data line. Data transfers to/from the SD Memory Card are done in blocks. Data blocks always succeeded by CRC bits. Single and multiple block operations are defined. Note that the Multiple Block operation mode is better for faster write operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the MSC to use single or multiple data lines.

Table 16-3 MMC/SD Data Token Format

Description	Start bit	Data	CRC16	End bit
Stream Data	0	X	no CRC	1
Block Data	0	X	X	1

16.4 Register Description

The MMC/SD controller is controlled by a set of registers that the application configures before every operation. The Table 16-4 lists all the MSC registers.

Table 16-4 MMC/SD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
MSC_STRPCL	W	0x0000	0x10021000	16
MSC_STAT	R	0x00000040	0x10021004	32
MSC_CLKRT	RW	0x0000	0x10021008	16
MSC_CMDAT	RW	0x00000000	0x1002100C	32
MSC_RESTO	RW	0x40	0x10021010	16
MSC_RDTO	RW	0xFFFF	0x10021014	16
MSC_BLKLEN	RW	0x0000	0x10021018	16
MSC_NOB	RW	0x0000	0x1002101C	16
MSC_SNOB	R	0x????	0x10021020	16
MSC_IMASK	RW	0x00FF	0x10021024	16
MSC_IREG[*1]	RW	0x0000	0x10021028	16
MSC_CMD	RW	0x00	0x1002102C	8
MSC_ARG	RW	0x00000000	0x10021030	32
MSC_RES	R	0x????	0x10021034	16
MSC_RXFIFO	R	0x????????	0x10021038	32
MSC_TXFIFO	W	0x????????	0x1002103C	32

Note: *1: Writing MSC_IREG is used to clear interrupt source and only three interrupt sources can be cleared by this way.

16.4.1 Start/stop MMC/SD clock Register (MSC_STRPCL)

MSC_STRPCL **0x10021000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								EXIT_MULTIPLE	EXIT_TRANSFER	START_READWAIT	STOP_READWAIT	RESET	START_OP	STRPCL	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved		R
7	EXIT_MULTIPLE	If CMD12 or CMD52 (I/O abort) is to be sent to terminate multiple block read/write in advance, set this bit to 1.	W

		0 – No effect. 1 – Exit from multiple block read/write.	
6	EXIT_TRANSFER	**Only used for SDIO suspend/resume and MMC stream read.** For SDIO, after suspend is accepted, set this bit with 1. For MMC, after the expected number of data are received, set this bit with 1. 0 – No effect. 1 – Exit from multiple block read/write after suspend is accepted, or exit from stream read.	W
5	START_READWAIT	Only used for SDIO ReadWait. Start the ReadWait cycle. 0 – No effect. 1 – Start ReadWait.	W
4	STOP_READWAIT	Only used for SDIO ReadWait. Stop the ReadWait cycle. 0 – No effect. 1 – Start ReadWait.	W
3	RESET	Resets the MMC/SD controller. 0 – No effect. 0 – Reset the MMC/SD controller.	W
2	START_OP	This bit is used to start the new operation. When starting the clock, this bit can be 1. When stopping the clock, this bit can only be 0. 0 – Do nothing. 1 – Start the new operation.	W
1:0	CLOCK_CONTROL	These bits are used to start or stop clock. 00 – Do nothing. 01 – Stop MMC/SD clock 10 – Start MMC/SD clock 11 – Reserved	W

16.4.2 MSC Status Register (MSC_STAT)

MSC_STAT **0x10021004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																IS_RESETTING	SDIO_INT_ACTIVE	PRG_DONE	DATA_TRAN_DONE	END_CMD_RES	DATA_FIFO_AFULL	IS_READWAIT	CLK_EN	DATA_FIFO_FULL	DATA_FIFO_EMPTY	CRC_RES_ERR	CRC_READ_ERROR	CRC_WRITE_ERROR		TIME_OUT_RES	TIME_OUT_READ
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved		R
15	IS_RESETTING	MSC is resetting after power up or MSC_STRPCL[RESET] is written with 1 0 – Reset has been finished. 1 – Reset has not been finished.	R
14	SDIO_INT_ACTIVE	Indicates whether an interrupt is detected at the SD I/O card. A separate acknowledge command to the card is required to clear this interrupt. 0 – No interrupt detected. 1 – The interrupt from SDIO is detected.	R
13	PRG_DONE	Indicates whether card has finished programming. 0 – Card has not finished programming and is busy. 1 – Card has finished programming and is not busy.	R
12	DATA_TRAN_DONE	Indicates whether data transmission to card has completed. 0 – Data transmission to card has not completed. 1 – Data transmission to card has completed.	R
11	END_CMD_RES	End command-response sequence or command sequence. 0 – Command and response/no-response sequence has not completed. 1 – Command and response/no-response sequence has completed.	R
10	DATA_FIFO_AFULL	Indicates whether data FIFO is almost full (The number of words >= 15). For reading data from card, use this bit. 0 – Data FIFO is not full. 1 – Data FIFO is full.	R
9	IS_READWAIT	Indicates whether SDIO card has entered ReadWait State 0 – Card has not enter ReadWait. 1 – Card has enter ReadWait.	R
8	CLK_EN	Clock enabled. 0 – Clock is off. 1 – Clock is on.	R
7	DATA_FIFO_FULL	Indicates whether data FIFO is full. For reading data from card, do not use this bit, because it almost keeps to be 0. 0 – Data FIFO is not full. 1– Data FIFO is full.	R
6	DATA_FIFO_EMPTY	Indicates whether data FIFO is empty. 0 – Data FIFO is not empty. 1– Data FIFO is empty.	R

5	CRC_RES_ERR	Response CRC error. 0 – No error on the response CRC. 1– CRC error occurred on the response.	R
4	CRC_READ_ERROR	CRC read error. 0 – No error on received data. 1– CRC error occurred on received data	R
3:2	CRC_WRITE_ERROR	CRC write error. 00 – No error on transmission of data. 01 – Card observed erroneous transmission of data. 10 – No CRC status is sent back. 11 – Reserved	R
1	TIME_OUT_RES	Response time out. 0 – Card response has not timed out. 1 – Card response has time out.	R
0	TIME_OUT_READ	Read time out. 0 – Card read data has not timed out. 1 – Card read data has timed out.	R

16.4.3 MSC Clock Rate Register (MSC_CLKRT)

The MSC_CLKRT register specifies the frequency division of the MMC/SD bus clock. The software is responsible for setting this register.

MSC_CLKRT																**0x10021008**
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved													CLK_RATE		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:3	Reserved		R
2:0	CLK_RATE	Clock rate. 000 – CLK_SRC. 001 – 1/2 of CLK_SRC. 010 – 1/4 of CLK_SRC. 011 – 1/8 of CLK_SRC. 100 – 1/16 of CLK_SRC. 101 – 1/32 of CLK_SRC. 110 – 1/64 of CLK_SRC. 111 – 1/128 of CLK_SRC.	WR

16.4.4 MMC/SD Command and Data Control Register (MSC_CMDAT)

MSC_CMDAT **0x1002100C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																Reserved				IO_ABORT	BUS_WIDTH		DMA_EN	INIT	BUSY	STREAM_BLOCK	WRITE_READ	DATA_EN	RESPONSE_FORMAT		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:12	Reserved		R
11	IO_ABORT	Specifies the current command is used to abort data transfer. **Only used for SDIO.** 0 – Nothing. 1 – The current command is used to abort transfer.	WR
10:9	BUS_WIDTH	Specifies the width of the data bus. 00 – 1-bit. 01 – Reserved. 10 – 4-bit. 11 – Reserved.	WR
8	DMA_EN	DMA mode enables. When DMA mode is used, this bit is also a mask on RXFIFO_RD_REQ and TXFIFO_WR_REQ interrupts. 0 – Program I/O. 1 – DMA mode.	WR
7	INIT	80 initialization clocks 0 – Do not precede command sequence with 80 clocks. 1 – Precede command sequence with 80 clocks.	W
6	BUSY	Specifies whether a busy signal is expected after the current command. This bit is for no data command/response transactions only. 0 – Not expect a busy signal. 0 – Expects a busy signal. If the response is R1b, then set it.	WR
5	STREAM_BLOCK	Stream mode 0 – Data transfer of the current command sequence is not in stream mode. 1– Data transfer of the current command sequence is in	WR

		stream mode.	
4	WRITE_READ	Specifies that the data transfer of the current command is a read or write operation. 0 – Specifies that the data transfer of the current command is a read operation. 1 – Specifies that the data transfer of the current command is a write operation.	WR
3	DATA_EN	Specifies whether the current command includes a data transfer. It is also used to reset RX_FIFO and TX_FIFO. 0 – No data transfer with current command. 1 – Has data transfer with current command. It is also used to reset RX_FIFO and TX_FIFO.	WR
2:0	RESPONSE_FORMAT	These bit specify the response format for the current command. 000 – No response. 001 – Format R1 and R1b. 010 – Format R2. 011 – Format R3. 100 – Format R4. 101 – Format R5. 110 – Format R6. 111 – Reserved.	WR

16.4.5 MMC/SD Response Time Out Register (MSC_RESTO)

MSC_RESTO **0x10021010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								RES_TO							
RST	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved		R
7:0	RES_TO	Specifies the number of MSC_CLK clock counts between the command and when the MMC/SD controller turns on the time-out error for the received response. The default value is 64.	WR

16.4.6 MMC/SD Read Time Out Register (MSC_RDTO)

MSC_RDTO **0x10021014**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	READ_TO															
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
15:0	READ_TO	Specifies the number of clocks between the command and when the MMC/SD host controller turns on the time-out error for the received data. The unit is CLK_SRC / 256.	WR

16.4.7 MMC/SD Block Length Register (MSC_BLKLEN)

MSC_BLKLEN **0x10021018**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				BLK_LEN											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved		R
11:0	BLK_LEN	Specifies the number of bytes in a block, and is normally set to 0x200 for MMC/SD data transactions. The value Specified in the cards CSD.	WR

16.4.8 MSC/SD Number of Block Register (MSC_NOB)

MSC_NOB **0x1002101C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOB															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	NOB	Specifies the number of blocks in a data transfer. One block is a possibility.	WR

16.4.9 MMC/SD Number of Successfully-transferred Blocks Register (MSC_SNOB)

In block mode, the MSC_SNOB register records the number of successfully transferred blocks. If the last block has CRC error, this register also summaries it. It is used to query blocks for multiple block transfer.

MSC_SNOB **0x10021020**

Bit		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		MSC_SNOB															
RST		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	MSC_SNOB	Specify the number of successfully transferred blocks for a multiple block transfer.	R

16.4.10 MMC/SD Interrupt Mask Register (MSC_IMASK)

MSC_IMASK **0x10021024**

Bit		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Reserved								SDIO	TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved		END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
15:8	Reserved		R
7	SDIO	Mask the interrupt from the SD I/O card. 0 – Not masked. 1 – Masked.	WR
6	TXFIFO_WR_REQ	Mask the Transmit FIFO write request interrupt. 0 – Not masked. 1 – Masked.	WR
5	RXFIFO_RD_REQ	Mask the Receive FIFO read request interrupt. 0 – Not masked. 1 – Masked.	WR
4:3	Reserved		R
2	END_CMD_RES	Mask the End command response interrupt. 0 – Not masked. 1 – Masked.	WR
1	PRG_DONE	Mask the Programming done interrupt. 0 – Not masked. 1 – Masked.	WR
0	DATA_TRAN_DONE	Mask the Data transfer done interrupt. 0 – Not masked.	WR

		1 – Masked.	

16.4.11 MMC/SD Interrupt Register (MSC_IREG)

The MSC_IREG register shows the currently requested interrupt. The FIFO request interrupts, TXFIFO_WR_REQ, and RXFIFO_RD_REQ are masked off with the DMA_EN bit in the MSC_CMDAT register. The software is responsible for monitoring these bit in program I/O mode.

MSC_IREG **0x10021028**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								SDIO	TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved		END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved		R
7	SDIO	Indicates whether the interrupt from SDIO is detected. 0 – The interrupt from SDIO is not detected. 1 – The interrupt from SDIO is detected.	R
6	TXFIFO_WR_REQ	Transmit FIFO write request. Set if data FIFO becomes half empty (the number of words is < 8). 0 – No Request for data Write to MSC_TXFIFO. 1 – Request for data write to MSC_TXFIFO.	R
5	RXFIFO_RD_REQ	Receive FIFO read request. Set if data FIFO becomes half full (the number of words is >= 8) or the entries in data FIFO are the last read data. 0 – No Request for data read from MSC_RXFIFO. 1 – Request for data read from MSC_RXFIFO.	R
4:3	Reserved		R
2	END_CMD_RES	Indicates whether the command/response sequence has been finished. 0 – The command/response sequence has not been finished. 1 – The command/response sequence has bee finished. Write 1 to clear.	WR
1	PRG_DONE	Indicates whether card has finished programming. 0 – Card has not finished programming and is busy. 1 – Card has finished programming and is no longer busy. Write 1 to clear.	WR

0	DATA_TRAN_DONE	Indicates whether data transfer is done. Note that for stream read/write, only when CMD12 (STOP_TRANS) has been sent, is this bit set. 0 – Data transfer is not complete. 1 – Data transfer has completed or an error has occurred. Write 1 to clear.	WR

16.4.12 MMC/SD Command Index Register (MSC_CMD)

MSC_CMD **0x1002102C**

Bit	7	6	5	4	3	2	1	0
	Reserved		CMD_INDEX					
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	Reserved		R
5:0	CMD_INDEX	Specifies the command index to be executed.	WR

16.4.13 MMC/SD Command Argument Register (MSC_ARG)

MSC_ARG **0x10021030**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARG																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	ARG	Specifies the argument for the current command.	WR

16.4.14 MMC/SD Response FIFO Register (MSC_RES)

The read-only MMC/SD Response FIFO register (RES_FIFO) holds the response sent back to the MMC/SD controller after every command. The size of this FIFO is 8 x 16-bit. The RES FIFO does not contain the 7-bit CRC for the response. The Status for CRC checking and response time-out status is in the status register, MSC_STAT.

The first halt-word read from the response FIFO is the most significant halt-word of the received response.

MSC_RES **0x10021034**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

		DATA
RST		? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?

Bits	Name	Description	RW
15:0	DATA	Contains the response to every command that is sent by the MMC/SD controller. The size of this FIFO register is 8 x 16-bit.	R

16.4.15 MMC/SD Receive Data FIFO Register (MSC_RXFIFO)

The MSC_RXFIFO is used to read the data from a card. It is read-only to the software, and is read on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_RXFIFO **0x10021038**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of read data. The size of this FIFO is 16 x 32-bit.	R

16.4.16 MMC/SD Transmit Data FIFO Register (MSC_TXFIFO)

The MSC_TXFIFO is used to write the data to a card. It is write-only to the software, and is written on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_TXFIFO **0x1002103C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of write data. The size of this FIFO is 16 x 32-bit.	W

16.5 MMC/SD Functional Description

All communication between system and cards is controlled by the MSC. The MSC sends commands of two type: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, command like "Go_Idle_State", "Send_Op_Cond", "All_send_CID" and "Set_relative_Addr" are using way of broadcasting. During Broadcast mode, all cards are in open-drain mode, to avoid bus contention.

After Broadcast commands "Set_relative_Addr" issue, cards are enter standby mode, and Addressed command will be used from now on, in this mode, CMD/DAT will return to push-pull mode, to have maximum driving for maximum operation frequency.

The MMC and the SD are similar product. Besides the 4x bandwidth and the built-in encryption, they are being programmed similarly.

The MMC/SD controller (MSC) is the interface between the software and the MMC/SD bus. It is responsible for the timing and protocol between the software and the MMC/SD bus. It consists of control and status registers, a 16-bit response FIFO that is 8 entries deep, and one 32-bit receive/transmit data FIFOs that are 16 entries deep. The registers and FIFOs are accessible by the software.

MSC also enable minimal data latency by buffering data and generating and checking CRCs.

16.5.1 MSC Reset

The MMC/SD controller (MSC) can be reset by a hardware reset or software reset. All registers and FIFO controls are set to their default values after any reset.

16.5.2 MSC Card Reset

The command Go_Idle_State, CMD0 is the software reset command for MMC and SD Memory Card, and sets each card into Idle State regardless of the current card state; while in SDIO card, CMD52 is used to write IO reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

16.5.3 Voltage Validation

All cards shall be able to establish communication with the host using any operation voltage in the maximal allowed voltage range specified in this standard. However, the support minimum and maximum values for Vdd are defined in Operation Conditions register (OCR) and many not cover the whole range. Cards that store the CID and CSD data in the payload memory would be able to communicate these information only under data transfer Vdd conditions. That means if host and card have non compatible Vdd ranges, the card will not be able to complete the identification cycle, nor to send CSD data.

Therefore, a special command Send_Op_cont (CMD1 for MMC), SD_Send_Op_Cont (CMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SDIO) are designed to provide a mechanism to identify and reject cards which do not match the Vdd range desired by the host. This is accomplished by the host sending the required Vdd voltage window as the operand of this command. Cards which can not perform data transfer in the specified range must discard themselves from further bus operations and go into Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification to the application of non usable cards in the stack is desired.

16.5.4 Card Registry

Card registry on MCC and SD card are different.

For SD card, Identification process start at clock rate Fod, while CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated the host will request the cards to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are sent into Inactive State. The host then issue the command All_Send_CID (CMD2) to each card and get its unique card indentification (CID) number. Card that is unidentified, that is, which is in Ready State, send its CID number as the response. After the CID was sent by the card it goes into Identification State. Thereafter, the host issues Send_Relative_Addr (CMD3) asks the card to publish a new relative card address (RCA), which is shorter that CID and which will be used to address the card in the future data transfer mode. Once the RCA is received the card state changes to the Stand-by State. At this point, if the host wants that the card will have another RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card. The host repeats the identification process, that is, the cycles with CMD2 and CMD3 for each card in the system.

In MMC, the host starts the card identification process in open-drain mode with the identification clock rate Fod. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is actived the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the ‘wired or’ operation on the condition restrictions of all cards in the system. Incompatible cards are sent into Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards, that is, those which are in Ready State, simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bitstream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately and must wait for the next identification cycle. Since CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to this card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by

State, and the card does not react to further identification cycles, and its output switches from open-drain to push-pull. The host repeat the process, that is CM2 and CMD3, until the host receive time-out condition to recognize completion of the identification process.

16.5.5 Card Access

16.5.5.1 Block Access, Block Write and Block Read

During block write (CMD24-27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line; the transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the writte buffer is unavailable.

Block read is similar to stream read, except the basic unit of data transfer is a block whose maximizes is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. Unlike stream read, a CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks will be continuously transferred until a stop command is issued. If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first mis-aligned block, set the ADDRESS_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.l

16.5.5.2 Stream Access, Stream Write and Stream Read (MMC Only)

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. Since the amount of data to be transferred is not determined in advance, CRC can not be used. If the end of the memory range is reached while sending data and no stop command has been sent by the host, all further transferred data is discarded.

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its payload, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command. If the end of the memory range is reached while sending data and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined.

16.5.5.3 Erase, Group Erase and Sector Erase (MMC Only)

It is desirable to erase many sectors simultaneously in order to enhance the data throughput. Identification of these sectors is accomplished with the TAG_* commands. Either an arbitrary set of sectors within a single erase group, or an arbitrary selection of erase groups may be erase at one time, but not both together. That is, the unit of measure for determining an erase is either a sector or an erase group. If a set of sectors must be erased, all selected sectors must lie within the same erase group. To facilitate selection, a first command with the starting address is followed by a second command with the final address, and all sectors (or groups) within this range will be selected for erase.

16.5.5.4 Wide Bus Selection/Deselection

Wide Bus (4 bit bus width) operation mode may be selected / deselected using ACMD6. The default bus width after power up or GO_IDLE (CMD0) is 1 bit bus width. ACMD6 command is valid in ‘trans state’ only. That means the bus width may be changed only after a card was selected (CMD7).

16.5.6 Protection Management

Three write protect methods are supported in the host for Cards, Card internal write protect (Card’s responsibility), Mechanical write protect switch (Host responsibility only) and Password protection card lock operation.

16.5.6.1 Card Internal Write Protection

Card data may be protected against either erase or write. The entire card may be permanently write protected by the manufacturer or content provider by setting the permanent or temporary write protect bits in the CSD. For cards which support write protection of groups of sectors by setting the WP_GRP_SIZE sectors as specified in the CSD), and the write protection may be changed by the application. The SET_WRITE_PROT command sets the write protection of the addressed write-protect group, and the CLR_WRITE_PROT command clears the write protection of the addressed write-protect group.

The SEND_WRITE_PROT command is similar to a single block read command. The card shall send a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card will ignore all LSB’s below the group size.

16.5.6.2 Mechanical write protect switch

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open that means the card is write protected. If the window is close the card is not write protected.

A proper, matched, switch on the socket side will indicated to the host that the card is write protected or not. It is the responsibility of the host to protect the card. The position of the write protect switch is un-known to the internal circuitry of the card.

16.5.6.3 Password Protect

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size is kept in an 128-bit PWD and 8-bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0) and “lock card” command class. Thus the host is allowed to reset, initialize, select, query for status, etc., but not to access data on the card. If the password was previously set (the value of PWD_LEN is not 0) will be locked automatically after power on. Similar to the existing CSD and CID register write commands the lock/unlock command is available in “trans_state” only. This means that it does not include an address argument and the card must be selected before using it. The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). The following table describes the structure of the command data block.

Table 16-5 Command Data Block Structure

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Rsv	Rsv	Rsv	Rsv	ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2	Password Data							
…								
PWDS_LEN + 1								

- **ERASE** – 1 Defines Forced Erase Operation (all other bits shall be 0) and only the command byte is sent.
- **LOCK/UNLOCK** – 1=Locks the card. 0=Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).
- **CLR_PWD** – 1=Clears PWD.
- **SET_PWD** – 1=Set new password to PWD.
- **PWD_LEN** – Defines the following password length (in bytes).
- **PWD** – The password (new or currently used depending on the command).

The data block size shall be defined by the host before it send the card lock/unlock command. This will allow different password sizes.

The following paragraphs define the various lock/unlock command sequences:

- Setting the Password:
    - Select a card (CMD7), if not previously selected already
    - Define the block length (CMD16), given by the 8bit card lock/unlock mode, the 8 bits password size (in bytes), and the number of bytes of the new password. In case that a password replacement is done, then the block size shall consider that both passwords, the old and the new one, are sent with the command.
    - Send Card Lock/Unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWD_LEN) and the password itself. In case that a password replacement is done, then the length value (PWD_LEN) shall include both passwords, the old and the new one, and the PWD field shall include the old password (currently used) followed by the new password.
    - In case that the sent old password is not correct (not equal in size and content) then LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In case that PWD matches the sent old password then the given new password and its size will be saved in the PWD and PWD_LEN fields, respectively.

  Note that the password length register (PWD_LEN) indicates if a password is currently set. When it equals 0 there is no password set. If the value of PWD_LEN is not equal to zero the card will lock itself after power up. It is possible to lock the card immediately in the current power session by setting the LOCK/UNLOCK bit (while setting the password) or sending additional command for card lock.

- Reset the password:
    - Select a card (CMD7), if not previously selected already
    - Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
    - Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWD_LEN) and the password (PWD) itself (LOCK/UNLOCK bit is don't care). If the PWD and PWD_LEN is set to 0. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

- Locking a card:
    - Select a card (CMD7), if not previously selected already
    - Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of currently used password.
    - Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct then LOCK_UNLOCK_FAILED error bit will be set in the status register.

Note that it is possible to set the password and to lock the card in the same sequence. In such case the host shall perform all the required steps for setting the password (as described above) including the bit LOCK set while the new password command is sent. If the password was previously set (PWD_LEN is not 0), then the card will be locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

- Unlocking the card
  - Select a card (CMD7), if not previously selected already.
  - Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
  - Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Note that the unlocking is done only for the current power session. As long as the PWD is not cleared the card will be locked automatically on the next power up. The only way to unlock the card is by clearing the password. An attempt to unlock an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

- Forcing Erase:
  In case that the user forgot the password (the PWD content) it is possible to erase all the card data content along with the PWD content. This operation is called Forced Erase.
  - Select a card (CMD7), if not previously selected already.
  - Define the block length (CMD16) to 1 byte (8bit card lock/unlock command). Send the card lock/unlock command with the appropriate data block of one byte on the data line including 16-bit CRC. The data block shall indicate the mode ERASE (the ERASE bit shall be the only bit set).

If the ERASE bit is not the only bit in the data field then the LCOK_UNLOCK_FAILED error bit will be set in the status register and the erase request is rejected. If the command was accepted then ALL THE CARD CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked card will get unlocked.

An attempt to force erase on an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

16.5.7 Card Status

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

Table below defines the different entries of the status. The type and clear condition fields in the table are abbreviate as follows:

Type:

E: Error bit.
S: Status bit.
R: Detected and set for the actual command response.
X: Detected and set during command execution. The host must poll the card by issuing the status command in order to read these bits.

Clear Condition:

A: According to the card current state.
B: Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).
C: Clear by read.

Table 16-6 Card Status Description

Bits	Identifier	Type	Description	Clear Condition
31	OUT_OF_RAGE	E R	The commmand's argument was out of the allowed range for this card. 0 – No Error 1 – Error	C
30	ADDRESS_ERROR	E R X	A misaligned address which did not match the block length was used in the command. 0 – No Error 1 – Error	C
29	BLOCK_LEN_ERROR	E R	The transferred block length is not allowed for this, or the number of transferred bytes does not match the block length. 0 – No Error 1 – Error	C

28	ERASE_SEQ_ERROR	E R	An error in the sequence of erase commands occurred. 0 – No Error 1 – Error	C
27	ERASE_PARAM	E X	An invalid selection of sectors or groups for erase occurred. 0 – No Error 1 – Error	C
26	WP_VIOLATION	E R X	Attempt to program a write protected block. 0 – No Protected 1 – Protected	C
25	CARD_IS_LOCKED	S X	When set, signals that the card is locked by the host. 0 – Card unlocked 1 – Card locked	A
24	LOCK_UNLOCK_FAILED	E R X	Set when a sequence or password error has been detected in lock/unlock card command or if there was an attempt to access a locked card. 0 – No Error. 1 – Error.	C
23	COM_CRC_ERROR	E R	The CRC check of the previous command failed. 0 – No Error. 1 – Error.	B
22	ILLEGAL_COMMAND	E R	Command not legal for the card state. 0 – No Error. 1 – Error.	B
21	CARD_ECC_FAILED	E X	Card internal ECC was applied but failed to correct the data. 0 – normal. 1 – failure.	C
20	CC_ERROR	E R X	Internal card controller error. 0 – No Error. 1 – Error.	C
19	ERROR	E R X	A general or an unknown error occurred during the operation. 0 – No Error. 1 – Error.	C

18	UNDERRUN	E X	The card could not sustain data transfer in stream read mode. 0 – No Error. 1 – Error.	C
17	OVERRUN	E X	The card could not sustain data programming in stream write mode. 0 – No Error. 1 – Error.	C
16	CID/CSD_OVERWRITE	E R X	Can be either one of the following errors: 0 – No Error. 1 – Error.	C
15	WP_ERASE_SKIP	S X	Only partial address space was erased due to existing write protected blocks. 1 – No Protected. 1 – Protected.	C
14	CARD_ECC_DISABLED	S X	The command has been executed without using the internal ECC. 0 – enabled. 1 – disabled.	A
13	ERASE_RESET	S R	An erase sequence was cleared before executing because an out of erase sequence command was received. 0 – normal. 1 – set.	C

12:9	CURRENT_STATE	S X	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as binary coded number between 0 and 15. 0 – idle 1 – ready 2 – ident 3 – stby 4 – tran 5 – data 6 – rcv 7 – prg 8 – dis (9 – 15) – rsv	B
8	READY_FOR_DATA	S X	Corresponds to buffer empty signaling on the bus. 0 – No Ready. 1 – Ready.	A
7:6	Reserved	-	-	-
5	APP_CMD	S R	The card will expect ACMD, or indication that the command has been interpreted as ACMD 0 – Disable. 1 – Enable.	C
4:0	Reserved	-	-	-

16.5.8 SD Status

The SD status contains status bits that are related to the SD card proprietary features and may be used for future application specific usage. The size of the SD status is one data block of 512bit. The content of this register is transmitted to the Host over the DAT bus along with 16-bit CRC. The SD status is sent to the host over the DAT bus if ACMD13 is sent (CMD55 followed with CMD13). ACMD13 can be sent to a card only in 'tran_state' (card is selected). SD status structure is described in below.

The same abbreviation for *type* and *clear condition* were used as for the Card Status above.

Table 16-7 SD Status Structure

Bits	Identifier	Type	Description	Clear Condition
511:510	DAT_BUS_WIDTH	S R	Shows the currently defined data	A

			bus width that was defined by SET_BUS_WIDTH command. 00 – 1 (default). 01 – Reserved. 10 – 4 bit width. 11 – Reserved.	
509	SECURED_MODE	S R	Card is in Secured Mode of operation. 0 – Not in the Mode. 10 – In the mode.	A
508:496	Reserved			
495:480	SD_CARD_TYPE	S R	All 0, is SD Memory cards.	A
479:448	SIZE_OF_PROTECTED_AREA	S R	Size of protected area.	A
447:312	Reserved			
311:0	Reserved for manufacturer			

16.5.9 SDIO

I/O access differs from memory in that the registers can be written and read individually and directly without a FAT file structure or the concept of blocks (although block access is supported). These registers allow access to the IO data, control of the IO function, and report on status or transfer I/O data to and from the host.

Each SDIO card may have from 1 to 7 functions plus one memory function built into it. A function is a self contained I/O device. I/O functions may be identical or completely different from each other. All I/O functions are organized as a collection of registers, and there is a maximum of 131,072 registers possible for each I/O function.

16.5.9.1 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, and interrupt function is added to a pin on the SD interface. Pin number 8 which is used as DAT[1] when operating in the 4 bit SD mode is used to signal the card’s interrupt to the host. The use of interrupt is optional for each card or function within a card. The SDIO interrupt is “level sensitive”, that is, the interrupt line must be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period. Once the host has serviced the interrupt, it is cleared via an IO write to the appropriate bit in the CCCR. The interrupt output of all SDIO cards is active low. This host controller provides pull-up resistors on all data lines DAT[3:0].

As Pin 8 of the card is shared between the IRQ and DAT[1] use in the 4 bit SD mode, and interrupt shall only be sent by the card and recognized by the host during a specific time. The time that a low on Pin 8 will be recognized as an interrupt is defined as the Interrupt Period.

The host here will only sample the level of Pin 8 (DAT[1]/IRQ) into the interrupt detector during the Interrupt Period. At all other times, the host will ignore the level on Pin 8. Note that the Interrupt Period is applicable for both memory and IO operations. The definition of the Interrupt Period is different for operations with single block and multiple block data transfer.

16.5.9.2 SDIO Suspend/Resume

Within a multi-function SDIO or a Combo (Mix IO and Memory) card, there are multiple devices (I/O and memory) that must share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume. In a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function of memory. Once this higher-priority transfer is complete, the original transfer is re-started where it left off (resume). The host controller here is supported by all IO functions except zero, and the memory of a combo card, and can suspend multiple transactions and resume them in any order desired. IO function zero does not support suspend/resume.

The procedure used to perform the Suspend/Resume operation on the SD bus is:

1. The host determines which function currently used the DAT[] line(s).
2. The host requests the lower priority or slower transaction to suspend.
3. The host checks for the transaction suspension to complete.
4. The host begins the higher priority transaction.
5. The host waits for the completion of the higher priority transaction.
6. The host restores the suspended transaction.

16.5.9.3 SDIO Read Wait

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The read wait operation allows a host to signal a card that it is doing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SDIO device. To determine if a card supports the Read Wait protocol, the host must test capability bits in CCCR. The timing for Read Wait is base on the Interrupt Period.

16.5.10 Clock Control

The software should guarantee that the card identification process starts in open-drain mode with the clock rate fod (0 ~ 400khz). In addition, the software should also make the card into interrupt mode with fod (only for MMC). The commands that require fod are CMD0, CMD1, CMD2, CMD3, CMD5, CMD40 and ACMD41. In data transfer mode, the MSC controller can operate card with clock rate fpp (0 ~ 25Mhz).

16.5.11 Application Specified Command Handling

The MultiMediaCard/SD system is designed to provide a standard interface for a variety applications types. In this environment it is anticipate that there will be a need for specific customers/applications features. To enable a common way of implementing these features, two types of generic commands are defined in the standard: Application Specific Command, ACMD, and General Command, GEN_CMD.

GEN_CMD, this command, when received by the card, will cause the card to interpret the following command as an application specific command, ACMD. The ACMD has the same structure as of regular MultiMediaCard standard commands and it may have the same CMD number. The card will recognize it as ACMD by the fact that it appears after APP_CMD.

The only effect of the APP_CMD is that if the command index of the, immediately, following command has an ACMD overloading, the none standard version will used. If, as an example, a card has a definition for ACMD13 but not for ACMD7 then, if received immediately after APP_CMD command, Command 13 will be interpreted as the non standard ACMD13 but, command 7 as the standard CMD7.

In order to use one of the manufacturer specific ACMD's the host will:

- Send APP_CMD. The response will have the APP_CMD bit (new status bit) set signaling to the host that ACMD is now expected.
- Send the required ACMD. The response will have the APP_CMD bit set, indicating that the accepted command was interpreted as ACMD. If a non-ACMD is sent then it will be respected by the card as normal MultiMediaCard command and the APP_CMD bit in the Card Status stays clear.

If a non valid command is sent (neither ACMD nor CMD) then it will be handled as a standard MultiMediaCard illegal command error.

The bus transaction of the GEN_CMD is the same as the single block read or write commands (CMD24 or CMD17). The difference is that the argument denotes the direction of the data transfer (rather than the address) and the data block is not a memory payload data but has a vendor specific format and meaning.

The card shall be selected ('tran_state') before sending CMD56. The data block size is the BLOCK_LEN that was defined with CMD16. The response to CMD56 will be R1b (card status + busy indication).

16.6 MMC/SD Controller Operation

16.6.1 Data FIFOs

The controller FIFOs for the response tokens, received data, and transmitted data are MSC_RES, MSC_RXFIFO, and MSC_TXFIFO, respectively. These FIFOs are accessible by the software and are described in the following paragraphs.

16.6.1.1 Response FIFO (MSC_RES)

The response FIFO, MSC_RES, contains the response received from an MMC/SD card after a command is sent from the controller. MSC_RES is a read-only, 16-bit, and 8-entry deep FIFO.

The FIFO will hold all possible response lengths. Responses that are only one byte long are located on the LSB of the 16-bit entry in the FIFO. The first half-word read from the response FIFO is the most significant half-word of the received response. For example, if the response format is R1, then the response read from RES_FIFO is bit [47:32], bit[31:16], bit[15:0] and in the third half-word only the low 8-bit is effective response [15:8] and the high 8-bit is ignored. If the response format is R2, then the response read from MSC_RES is bit [135:8] and needs reading 8 times.

The FIFO does not contain the response CRC. The status of the CRC check is in the status register, MSC_STAT.

16.6.1.2 Receive/Transmit Data FIFO (MSC_RXFIFO/MSC_TXFIFO)

The receive data FIFO and transmit data FIFO share one 16-entry x 32-bit FIFO, because at one time data are only received or are only transmitted. If it is used to receive data, it is called MSC_RXFIFO and read-only. If it is used to transmit data, it is called MSC_TXFIFO and write-only.

Data FIFO and its controls are cleared to a starting state after a system reset or at the beginning of the operations which include data transfer (MSC_CMDAT[DATA_EN] == 1).

If at any time MSC_RXFIFO becomes full and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any overflows. When the clock is off, data transmission from the card stops until the clock is turned back on. After MSC_RXFIFO is not full, the controller turns the clock on to continue data transmission. The full status of the FIFO is registered in the MSC_STAT [DATA_FIFO_FULL] bit.

If at any time MSC_TXFIFO becomes empty and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any underrun. When the clock is off, data transmission to the card stops until the clock is turned back on. When MSC_TXFIFO is no longer empty, the controller automatically restarts the clock. The empty status of the FIFO is registered in the MSC_STAT [DATA_FIFO_EMPTY] bit.

The FIFO is readable on word (32-bit) boundaries. The max read/written number is 16 words. The controller can correctly process big-endian an little-endian data.

Because at the beginning of the operation which include data transfer (MSC_CMDAT [DATA_EN] == 1), Data FIFO and its controls are cleared, software should guarantee data in FIFO have been read/written before beginning a new command.

16.6.2 DMA and Program I/O

Software may communicate to the MMC controller via the DMA or program I/O.

To access MSC_RXFIFO/MSC_TXFIFO with the DMA, the software must program the DMA to read or write the FIFO with source port width 32-bit, destination port width 32-bit, transfer data size 32-byte, transfer mode single. For example, to write 64 bytes of data to the MSC_TXFIFO, the software must program the DMA as follows:

```
DMA_DCTRn = 2                  // Write 2 32-bytes (64 bytes)
DMA_DCCRn[SWDH] = 0            // source port width is 32-bit
DMA_DCCRn[DWDH] = 0            // destination port width is 32-bit
DMA_DCCRn[DS] = 4              // transfer data size is 32-byte
DMA_DCCRn[TM] = 4              // transfer mode is single
DMA_DCCRn[RDIL] = 0            // request detection interval length is 0
```

The number of 32-bytes should be calculated from the number of transferred bytes as follows:

The number of words = (The number of bytes + 31) / 32

If the number of transferred bytes is not the multiple of 4, the controller can correctly process endian.

The DMA trigger level is 8 words, that is to say, the DMA read trigger is when data words in MSC_RXFIFO is >= 8 and the DMA write trigger is when data words in MSC_TXFIFO is < 8. Software can also configure DMA registers based on requirements, but the above 32-byte transfer data size is most efficient.

With program I/O, the software waits for the MSC_IREG [RXFIFO_RD_REQ] or MSC_IREG [TXFIFO_WR_REQ] interrupts before reading or writing the respective FIFO.

Note:

1. The MSC_CMDAT [DMA_EN] bit must be set to a 1 to enable communication with the DMA and it must be set to a 0 to enable program I/O.
2. DMA can be enabled only after MSC_CMDAT is written, because MSC_CMDAT [DATA_EN] is used to reset TX/RXFIFO.

16.6.3 Start and Stop clock

The software stops the clock as follows:

1. Write MSC_STRPCL with 0x01 to stop the MMC/SD bus clock.
2. Wait until MSC_STAT[CLK_EN] becomes zero.

To start the clock the software writes MSC_STRPCL with 0x02.

16.6.4 Software Reset

Reset includes the MSC reset and the card reset.

The MSC reset is through MSC_STRPCL [RESET] bit.

The card reset is to make the card into idle state. CMD0 (GO_IDLE_STATE) sets the MMC and SD memory cards into idle state. CMD52 (IO_RW_DIRECT, with argument 0x88000C08) reset the SD I/O card. The MMC/SD card are initialized with a default relative card address (RCA = 0x0001 for MMC and RCA = 0x0000 for SD) and with a default driver stage register setting (lowest speed, highest driving current capability).

The following registers must be set before the clock is started:

1. STOP THE CLOCK.
2. Set MSC_STRPCL register to 0x08 to reset MSC.
3. Wait while MSC_STAT [IS_RESETTING] is 1.
4. Set MSC_CMD with CMD0.
5. Update the MSC_CMDAT register as follows:
   a) Write 0x0000 to MSC_CMDAT [RESPONSE_FORMAT]
   b) Clear the MSC_CMDAT [DATA_EN] bit.
   c) Clear the MSC_CMDAT [BUSY] bit.
   d) Clear the MSC_CMDAT [INIT] bit.
6. Start the clock.
7. Start the operation (write MSC_STRPCL with 0x04)
8. Wait for the END_CMD_RES interrupt.

9. Set MSC_CMD with CMD52.
10. Set MSC_ARG with 0x88000C08
11. Update the MSC_CMDAT register as follows:
    a) Write 0x005 to MSC_CMDAT [RESPONSE_FORMAT]
    b) Clear the MSC_CMDAT [DATA_EN] bit.
    c) Clear the MSC_CMDAT [BUSY] bit.
    d) Clear the MSC_CMDAT [INIT] bit.
12. Start the operation.
13. Wait for the END_CMD_RES interrupt.

16.6.5 Voltage Validation and Card Registry

At most 10 MMC and 1 SD (either SDMEM or SDIO) can be inserted MMC/SD bus at the same time, and their voltage validation and card registry steps are different, so the software should be programmed as follows:

1. CHECK WHETHER SDIO CARD IS INSERTED.
2. Check whether SDMEM card is inserted.
3. Check whether MMC cards are inserted.

16.6.5.1 Check SDIO

The commands are sent as follows:

1. (Optional) Send CMD52 (IO_RW_DIRECT) with argument 0x88000C08 to reset SDIO card.
2. Send CMD5 (IO_SEND_OP_CMD) to validate voltage.
3. If the response is correct and the number of IO functions > 0, then continue, else go to check SDMEM.
4. If C-bit in the response is ready (the initialization has finished), go to 6.
5. Send CMD5 (IO_SEND_OP_CMD) to validate voltage, then go to 4.
6. If memory-present-bit in the response is true, then it is a combo card (SDIO + Memory), else it is only a SDIO card.
7. If it is a combo card, go to check SDMEM to initialize the memory part.
8. Send CMD3 (SET_RELATIVE_ADDR) to let the card publish a RCA. The RCA is returned from the response.
9. If do not accept the new RCA, go to 8, else record the new RCA.
10. Go to check MMC, because we can assure that there is no SDMEM card.

16.6.5.2 Check SDMEM

If there is no SDIO card or there is a combo card, continue to check SDMEM.

The commands are sent as follows:

1. (Optional) Send CMD0 (GO_IDLE_STATE) to reset MMC and SDMEM card. This command has no response.
2. Send CMD55. Here the default RCA 0x0000 is used for CMD55.
3. If the response is correct (CMD55 has response), then continue, else go to check MMC.
4. Send ACMD41 (SD_SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF8000).
5. If the initialization has finished, go to 7. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)
6. Send CMD55 and ACMD41 to validate voltage, and then go to 5.
7. Send CMD2 (ALL_SEND_CID) to get the card CID.
8. Send CMD3 (SET_RELATIVE_ADDR) to let card publish a RCA. The RCA is returned from the response.
9. If do not accept the new RCA, go to 8, else record the new RCA.
10. Go to check MMC.

16.6.5.3 Check MMC

Because there may be several MMC card, so some steps (5 ~ 8) should be repeated several times.

The commands are sent as follows:

1. Send CMD1 (SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF88000).
2. If the response is correct, then continue, else goto 9.

3. If the initialization has finished, go to 5. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)
4. Send CMD1 (SEND_OP_CMD) to validate voltage, and then go to 3.
5. Send CMD2 (ALL_SEND_CID) to get the card CID.
6. If the response timeout occurs, goto 9.
7. Send CMD3 (SET_RELATIVE_ADDR) to assign the card a RCA.
8. If there are other MMC cards, then go to 5.
9. Finish.

16.6.6 Single Data Block Write

In a single block write command, the following registers must be set before the operation is started:

1. Set MSC_NOB register to 0x0001.
2. Set MSC_BLKLEN to the number of bytes per block.
3. Update the MSC_CMDAT register as follows:
   a) Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT]
   b) Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.
   c) Set the MSC_CMDAT [DATA_EN] bit.
   d) Set the MSC_CMDAT [WRITE_READ] bit.
   e) Clear the MSC_CMDAT [STREAM_BLOCK] bit.
   f) Clear the MSC_CMDAT [BUSY] bit.
   g) Clear the MSC_CMDAT [INIT] bit.
4. Start the operation.
5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
   At the same time write data to the MSC_TXFIFO and continue until all of the data have been written to the FIFO.
3. Wait for MSC_IREG [PROG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.
4. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

To address a different card, the software sends a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PROG_DONE] interrupt. This ensures that the card is not in the busy state.

In addition, CMD26 (PROGRAM_CID), CMD27 (PROGRAM_CSD), CMD42 (LOCK/UNLOCK), CMD56 (GEN_CMD: write) and CMD53 (single_block_write) operations are similar to single block write.

16.6.7 Single Block Read

In a single block read command, the following registers must be set before the operation is started:

1. SET MSC_NOB REGISTER TO 0X0001.
2. Set MSC_BLKLEN register to the number of bytes per block.
3. Update the following bits in the MSC_CMDAT register:
    a) Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT].
    b) Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.
    c) Set the MSC_CMDAT [DATA_EN] bit.
    d) Clear the MSC_CMDAT [WRITE_READ] bit.
    e) Clear the MSC_CMDAT [STREAM_BLOCK] bit.
    f) Clear the MSC_CMDAT [BUSY] bit.
    g) Clear the MSC_CMDAT [INIT] bit.
4. Start the operation.
5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
   At the same time read data from the MSC_RXFIFO as data becomes available in the FIFO, and continue reading until all data is read from the FIFO.
3. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

In addition, CMD30 (SEND_WRITE_PROT), ACMD13 (SD_STATUS), CMD56 (GEN_CMD-read), ACMD51 (SEND_SCR) and CMD53 (single_block_read) are similar to single block read.

16.6.8 Multiple Block Write

The multiple block write mode is similar to the single block write mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block write, except that the MSC_NOB register is set to the number of blocks to be written.

The multiple block write mode also requires a stop transmission command, CMD12, after the data is transferred to the card. After the MSC_IREG [DATA_TRAN_DONE] interrupt occurs, the software must program the controller register to send a stop data transmission command.

If multiple block write with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_write) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 16-8 How to stop multiple block write

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After write MSC_NOB blocks	1. Wait for DATA_TARN_DONE interrupt 2. Send CMD12 or CMD52 (IO abort)

		3. Wait for END_CMD_RES and PRG_DONE interrupt
Open-ended or SDIO infinite	Stop writing in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES and PRG_DONE interrupt.
Predefined block or SDIO finite	After writing MSC_NOB blocks	1. Wait for DATA_TRAN_DONE interrupt
Predefined block or SDIO finite	Stop writing in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES and PRG_DONE interrupt

16.6.9 Multiple Block Read

The multiple blocks read mode is similar to the single block read mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block read, except that the MSC_NOB register is set to the number of blocks to be read.

The multiple blocks read mode requires a stop transmission command, CMD12, after the data from the card is received. After the MSC_IREG [DATA_TRAN_DONE] interrupt has occurred, the software must program the controller registers to send a stop data transmission command.

If multiple block read with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_read) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 16-9 How to stop multiple block read

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After reading MSC_NOB blocks	1. Wait for DATA_TRAN_DONE interrupt 2. Send CMD12 or CMD52 (IO abort) 3. Wait for END_CMD_RES interrupt
Open-ended or SDIO infinite	Stop reading in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt 3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES interrupt
Predefined block or SDIO finite	After reading MSC_NOB blocks	1. Wait for DATA_TRAN_DONE interrupt
Predefined block or SDIO finite	Stop reading in advance (not write MSC_NOB blocks)	1. Set MSC_STRPCL [EXIT_MULTIPLE] 2. Wait for DATA_TRAN_DONE interrupt

		3. Send CMD12 or CMD52 (IO abort) 4. Wait for END_CMD_RES interrupt

16.6.10 Stream Write (MMC)

In a stream write command, the following registers must be set before the operation is started:

1. UPDATE MSC_CMDAT REGISTER AS FOLLOWS:
   a) Write 0x001 to the MSC_CMDAT [RESPONSE_FORMAT].
   b) Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream write
   c) Set the MSC_CMDAT [DATA_EN] bit.
   d) Set the MSC_CMDAT [WRITE_READ] bit.
   e) Set the MSC_CMDAT [STREAM_BLOCK] bit.
   f) Clear the MSC_CMDAT [BUSY] bit.
   g) Clear the MSC_CMDAT [INIT] bit.
2. Start the operation.
3. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. WAIT FOR THE MSC_IREG [END_CMD_RES] INTERRUPT.
2. Write data to the MSC_TXFIFO and continue until all of the data is written to the Data FIFO.
3. Stop clock. Wait until MSC_STAT[CLK_EN] becomes 0. The clock must be stopped.
4. Set the command registers for a stop transaction command (CMD12) and other registers.
5. Start the clock and start the operation.
6. Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
8. Wait for the MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.
9. Read the MSC_STAT register to verify the status of the transaction.

To address a different card, the software must send a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PRG_DONE] interrupt. This ensures that the card is not in the busy state.

If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. If WRITE_BL_PARTIAL is not set, 16 more stuff bytes need to be written after the useful written data, otherwise only write the useful written data.

16.6.11 Stream Read (MMC)

In a stream read command, the following registers must be set before the operation is turned on:

1. UPDATE THE MSC_CMDAT REGISTER AS FOLLOWS:
   a) Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT]
   b) Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream read.
   c) Clear the MSC_CMDAT [WRITE_READ] bit.

   d) Set the MSC_CMDAT [STREAM_BLOCK] bit.
   e) Clear the MSC_CMDAT [BUSY] bit.
   f) Clear the MSC_CMDAT [INIT] bit.
2. Start the operation.
3. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. WAIT FOR THE MSC_IREG [END_CMD_RES] INTERRUPT.
2. Read data from the MSC_RXFIFO and continue until all of the expected data has been read from the FIFO.
3. Write MSC_STRPCL [EXIT_TRANSER] with 1. If MSC_STAT[DATA_FIFO_FULL] is 1, then read MSC_RXFIFO to make it not full. Because if data FIFO is full, MSC_CLK is stopped. Here, the data FIFO contains useless data.
4. Set the command registers for a stop transaction command (CMD12) and send it. There is no need to stop the clock.
5. Wait for the MSC_IREG [END_CMD_RES]
6. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
7. Read the MSC_STAT register to verify the status of the transaction.

16.6.12 Erase, Select/Deselect and Stop

For CMD7 (SELECT/DESELECT_CARD), CMD12 (STOP_TRANSMISSION) and CMD38 (ERASE), the following registers must be set before the operation is started:

1. UPDATE THE MSC_CMDAT REGISTER AS FOLLOWS:
   a) Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT]
   b) Clear the MSC_CMDAT [DATA_EN] bit.
   c) Clear the MSC_CMDAT [WRITE_READ] bit.
   d) Clear the MSC_CMDAT [STREAM_BLOCK] bit.
   e) Set the MSC_CMDAT [BUSY] bit.
   f) Clear the MSC_CMDAT [INIT] bit.
2. Start the operation.
3. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1. WAIT FOR THE MSC_IREG [END_CMD_RES] INTERRUPT.
2. Wait for the MSC_IREG [PRG_DONE] interrupt. If CMD12 is sent to terminate data read operation, then there is no need to wait for MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.

16.6.13 SDIO Suspend/Resume

The actual suspend/resume steps are as follows:

1. DURING DATA TRANSFER, SEND CMD52 TO REQUIRE SUSPEND. BR AND RAW FLAG SHOULD BE 1.
2. If BS flag in the response is 0, then suspend has been accepted and goto 4

3. Send CMD52 to query card status. R flag should be 1. Go to 2.
4. Write MSC_STRPCL [EXIT_TRANSFER] with 1.
5. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6. Read MSC_NOB, MSC_SNOB and etc, save them into variables.
7. Set registers for high priority transfer and start it.
8. Wait until high priority transfer is finished.
9. Restore registers from variables, but MSC_NOB should be (MSC_NOB – MSC_SNOB).
10. Send CMD52 to require resume. FSx should be resumed function number.

16.6.14 SDIO ReadWait

The actual ReadWait steps are as follows

1. During multiple block read, read MSC_SNOB. If MSC_SNOB is nearby or equal to MSC_NOB, no need to use ReadWait.
2. Write MSC_STRPCL [START_READWAIT] with 1.
3. Wait until MSC_STAT [IS_READWAIT] becomes 1.
4. Send CMD52 to query card status.
5. Write MSC_STRPCL [STOP_READWAIT] with 1.

16.6.15 Operation and Interrupt

The software can use polling-status method to operate the MMC/SD card, but this is not the proposed method, because its performance is very low. The proposed method is to use interrupt. Generally there are fixed necessary steps to finish each command. The steps are as follows.

1. (Optional) Stop clock. Poll CLK_EN.
2. Fill the registers (MSC_CMD, MSC_CMDAT, MSC_ARG, MSC_CLKRT, and etc).
3. (Optional) Start clock.
4. Start the operation. Wait for the MSC_IREG [END_CMD_RES] interrupt.
5. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6. Send STOP_TRANS (CMD12) or I/O abort (CMD52). Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt
8. Wait for the MSC_IREG [PRG_DONE] interrupt.

Table 16-10 The mapping between Commands and Steps

Index	Abbreviation	1	2	3	4	5	6	7	8	Comments
CMD0	GO_IDLE_STATE	Y	Y	Y	Y					
CMD1	SEND_OP_COND	Y	Y	Y	Y					
CMD2	ALL_SEND_CID	Y	Y	Y	Y					
CMD3	SET_RELATIVE_ADDR	Y	Y	Y	Y					
CMD4	SET_DSR	Y	Y	Y	Y					
CMD7	SELECT/DSELECT_CARD	Y	Y	Y	Y				Y	
CMD9	SEND_CID	Y	Y	Y	Y					
CMD10	SEND_CSD	Y	Y	Y	Y					
CMD11	READ_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y		

CMD12	STOP_TRANSMISSION	Y	Y	Y	Y				Y	
CMD13	SEND_STATUS	Y	Y	Y	Y					
CMD15	GO_INACTIVE_STATE	Y	Y	Y	Y					
CMD16	SET_BLOCKLEN	Y	Y	Y	Y					
CMD17	READ_SINGLE_BLOCK	Y	Y	Y	Y	Y				
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y			Open-ended
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y				Predefine blocks
CMD20	WRITE_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y	Y	
CMD23	SET_BLOCK_COUNT	Y	Y	Y	Y					
CMD24	WRITE_SINGLE_BLOCK	Y	Y	Y	Y	Y			Y	
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y		Y	Open-ended
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y			Y	Predefine blocks
CMD26	PROGRAM_CID	Y	Y	Y	Y	Y			Y	
CMD27	PROGRAM_CSD	Y	Y	Y	Y	Y			Y	
CMD28	SET_WRITE_PROT	Y	Y	Y	Y				Y	
CMD29	CLR_WRITE_PROT	Y	Y	Y	Y				Y	
CMD30	SEND_WRITE_PROT	Y	Y	Y	Y	Y				
CMD32	ERASE_WR_BLOCK_START	Y	Y	Y	Y					
CMD33	ERASE_WR_BLOCK_END	Y	Y	Y	Y					
CMD35	ERASE_GROUP_START	Y	Y	Y	Y					
CMD36	ERASE_GROUP_END	Y	Y	Y	Y					
CMD38	ERASE	Y	Y	Y	Y				Y	
CMD39	FAST_IO	Y	Y	Y	Y					
CMD40	GO_IRQ_STATE	Y	Y	Y	Y					
CMD42	LOCK/UNLOCK	Y	Y	Y	Y	Y			Y	
CMD55	APP_CMD	Y	Y	Y	Y					
CMD56	GEN_CMD	Y	Y	Y	Y	Y				Read
CMD56	GEN_CMD	Y	Y	Y	Y	Y			Y	Write
ACMD6	SET_BUS_WIDTH	Y	Y	Y	Y					
ACMD13	SD_STATUS	Y	Y	Y	Y	Y				
ACMD22	SEND_NUM_WR_BLOCKS	Y	Y	Y	Y					
ACMD23	SET_WR_BLOCK_COUNT	Y	Y	Y	Y					
ACMD41	SD_SEND_OP_COND	Y	Y	Y	Y					
ACMD42	SET_CLR_CARD_DETECT	Y	Y	Y	Y					
ACMD51	SEND_SCR	Y	Y	Y	Y	Y				

Note: For stream read/write, STOP_CMD is sent after finishing data transfer. For write, STOP_CMD is with the last six bytes. For read, STOP_CMD is sent after receiving data and card sends some data which MSC ignores.

17 I2C Bus Interface

17.1 Overview

The I2C bus was created by the Phillips Corporation and is a serial bus with a two-pin interface. The SDA data pin is used for input and output functions and the SCL clock pin is used to control and reference the I2C bus. The I2C unit allows the processor to serve as a master and slave device that resides on the I2C bus. The I2C unit enables the processor to communicate with I2C peripherals and microcontrollers for system management functions. The I2C bus requires a minimum amount of hardware to relay status and reliability information concerning the processor subsystem to an external device. The I2C unit is a peripheral device that resides on the processor internal bus. Data is transmitted to and received from the I2C bus via a buffered interface. Control and status information is relayed through a set of memory-mapped registers. Refer to ***The I2C-Bus Specification*** for complete details on I2C bus operation.

The I2C has the following features:

- Supports only single master mode.
- Supports I2C standard-mode and F/S-mode up to 400 kHz.
- I2C receiver and transmitter are double-buffered.
- Supports burst reading or writing of data.
- Supports random writing access of data.
- Supports general call address and START byte format after START condition.
- Independent, programmable serial clock generator.
- Supports slave coping with fast master during data transfers by holding the SCL line on a bit level.
- The number of devices that you can connect to the same I2C-bus is limited only by the maximum bus capacitance of 400pF.

17.2 Pin Description

Table 17-1 Smart Card Controller Pins Description

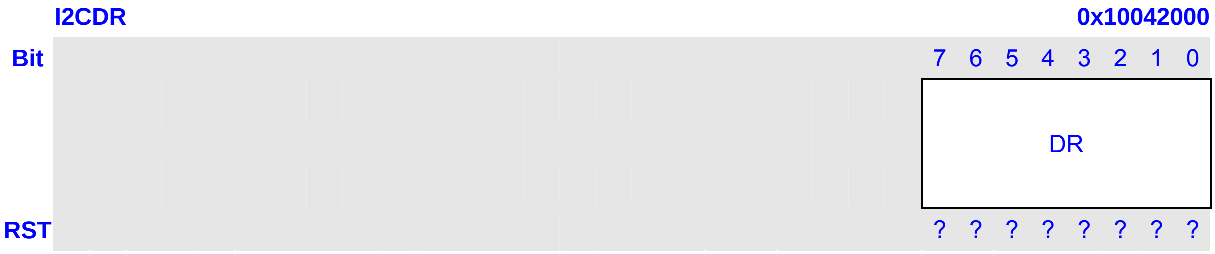
Name	I/O	Description
SDA	Input/Output	I2C Serial Clock Line signal.
SCL	Input/Output	I2C Serial Data/Address signal.

17.3 Register Description

Table 17-2 I2C Registers Description

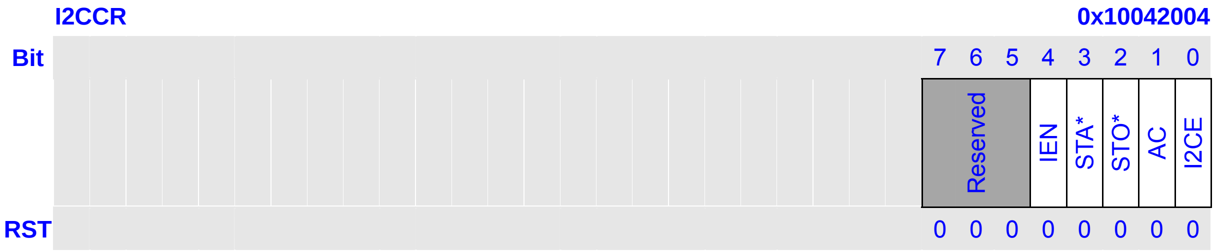
Name	RW	Reset Value	Address	Access Size
I2CDR	RW	0x??	0x10042000	8
I2CCR	RW	0x00	0x10042004	8
I2CSR	RW	0x04	0x10042008	8
I2CGR	RW	0x0000	0x1004200C	16

17.3.1 Data Register (I2CDR)


I2CDR
0x10042000
Bit
7 6 5 4 3 2 1 0
DR
RST
? ? ? ? ? ? ? ?

Bits	Name	Description	RW
7:0	DR	Data port of HW FIFO.	RW

17.3.2 Control Register (I2CCCR)


I2CCR
0x10042004
Bit
7 6 5 4 3 2 1 0
Reserved
IEN
STA*
STO*
AC
I2CE
RST
0 0 0 0 0 0 0 0

***Note**: STA and STO can only be written with 1.

Bits	Name	Description	RW
7:5	Reserved	These bits always read as 0. Write data to these bits are ignored.	R
4	IEN	I2C interrupt bit. 0 – Disable I2C interrupt. 1 – Enable I2C interrupt.	RW
3	STA	I2C START bit. 0 – START condition will not be sent to I^{2}C bus. 1 – START condition will be sent to I^{2}C bus.	RW
2	STO	I2C STOP bit. 0 – STOP condition won't be sent to I^{2}C bus. 1 – STOP condition will be sent to I^{2}C bus.	RW
1	AC	I2C Acknowledge Control Bit. 0 – will be sent to I^{2}C bus as LOW level acknowledge signal. 1 – will be sent to I^{2}C bus as HIGH level	RW

		acknowledge signal.	
0	I2CE	Enable of I2C. 0 – I2C module is disabled. 1 – I2C module is enabled.	RW

17.3.3 Status Register (I2CSR)

I2CSR **0x10042008**

Bit	7	6	5	4	3	2	1	0
	Reserved			STX	BUSY	TEND	DRF	ACKF
RST	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
7:5	Reserved	These bits always read as 0. Write data to these bits are ignored.	R
4	STX	STA/STO Command is On. 0 – STA/STO FIFO buffer is empty. 1 – STA/STO FIFO buffer is not empty.	R
3	BUSY	I2C Bus Busy. 0 – I2C bus is free. 1 – I2C bus is busy.	R
2	TEND	Transmission End Flag. 0 – Byte transmission or acknowledge bit for that byte has not completed. 1 – The I2C is in transmission idle state.	R
1	DRF	Data Register Valid Flag. 0 – Data in I2CDR is invalid. 1 – Data in I2CDR is valid.	RW
0	ACKF	Acknowledge Level Flag. 0 – The acknowledge signal from I^2C-bus is “0”. 1 – The acknowledge signal from I^2C-bus is “1”.	R

17.3.4 Clock Generator Register (I2CGR)

I2CGR **0x1004200C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GR															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:01	GR	Sets the frequency of serial clock. The serial clocks frequency is calculated as follows: [Value of I2CGR] = [Frequency of Device_clock] / (16 * [SCL clock rate]) – 1	RW

Note: To make the I2C operate normally, frequency of PCLK (APB-bus clock) should not lower than transfer 2 * [byte rate].

17.4 I^{2}C-Bus Protocol

17.4.1 Bit Transfer

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I^{2}C-bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of VDD. One clock pulse is generated for each data bit transferred.

17.4.2 Data Validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW states of the data line can only change when the clock signal on the SCL line is LOW.

17.4.3 START and STOP Conditions

A HIGH to LOW transition on the SDA line while SCL is HIGH indicates a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

17.4.4 Byte Format

1. Every byte put on the SDA line must be 8-bits width
2. The number of bytes that can be transmitted/received per transfer is unrestricted.
3. Each byte has to be followed by an acknowledge (ack/nack) bit.
4. Data is transferred with the most significant bit (MSB) first.
5. Data transfer with an acknowledge signal (acknowledge or not-acknowledge) is obligatory.
6. The acknowledge_ related clock pulse is generated by the master.
7. The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse. The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse.
8. Slave can hold the SCL line LOW during the SCL in LOW level at any bit to force the master to proceed a lower speed of transfer.

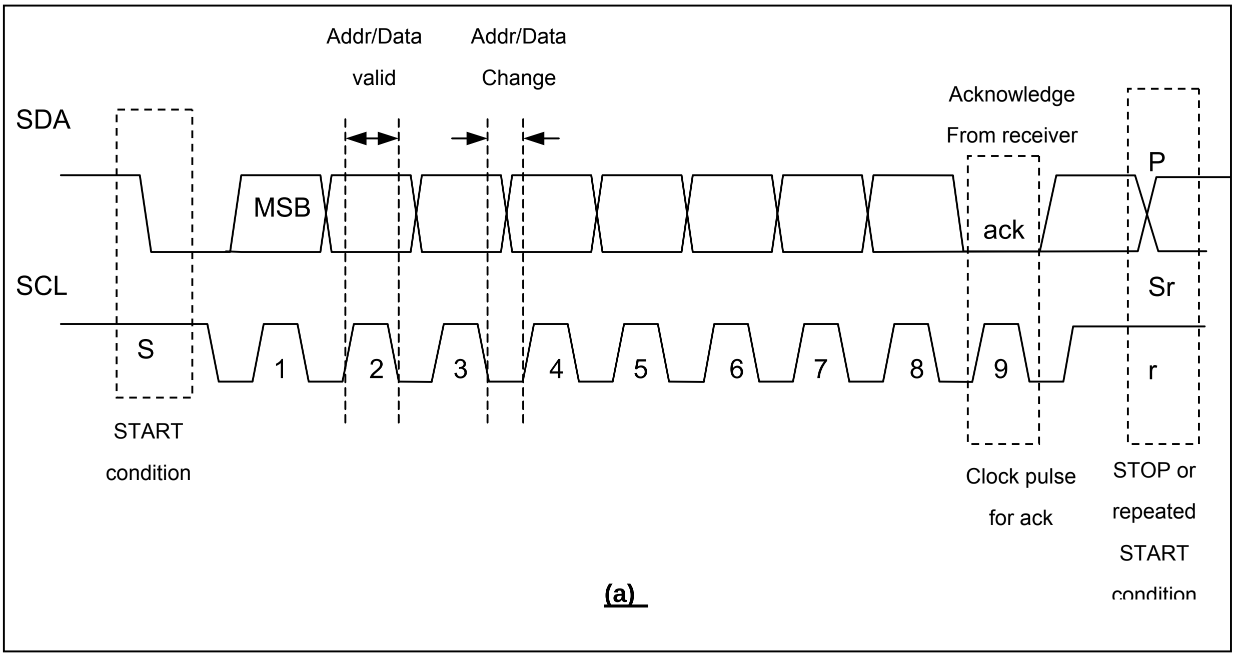

Addr/Data
valid
Addr/Data
Change
Acknowledge
From receiver
SDA
SCL
MSB
ack
P
Sr
S
1
2
3
4
5
6
7
8
9
r
START
condition
Clock pulse
for ack
STOP or
repeated
START
condition
(a)

Figure 17-1 I2C-bus Protocol

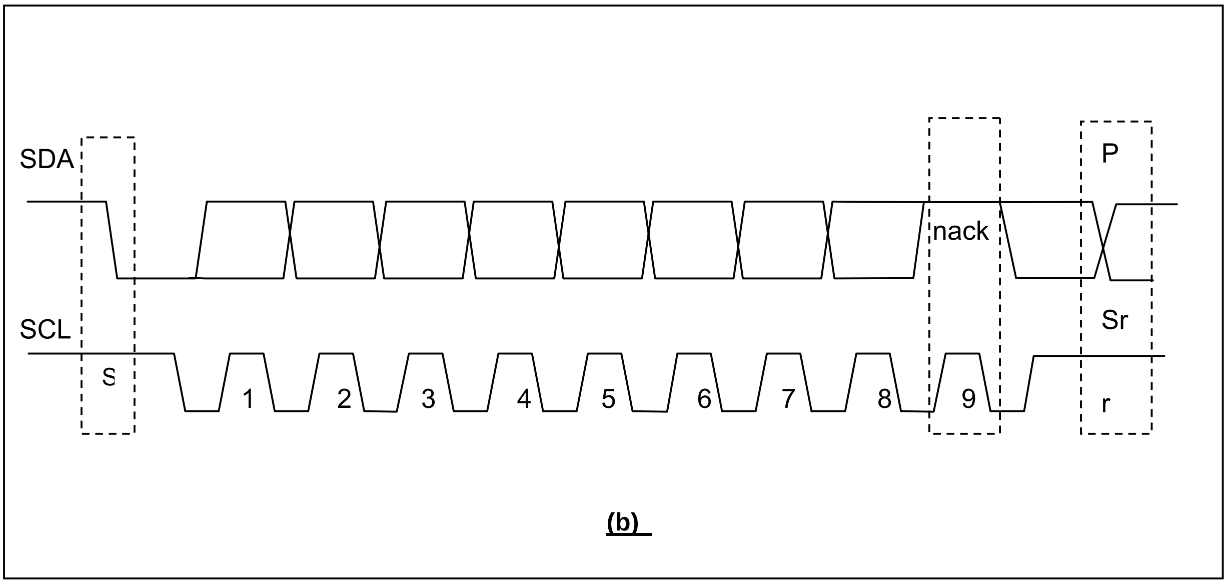

SDA
SCL
nack
P
Sr
S
1
2
3
4
5
6
7
8
9
r
(b)

Figure 17-2 I^{2}C-bus Protocol (cont.)

Notes:

1. Sr means repeated START condition. P means STOP condition.

2. In Fig (a), if the master does not generate Sr or P, the next data byte follows the ack.
3. In Fig (b), nack is received, the master generates Sr or P and the transfer terminates.

17.4.5 Data Transfer Format

17.4.5.1 First Byte

The first byte is a term indicates the address byte after START condition.

1) Normal 7-bit Address:

After the START condition, the addressing procedure for the I^2C-bus is such that the first byte usually determines which slave will been selected by the master.

The first seven bits of the first byte make up the slave address. The eighth bit is the LSB (least significant bit). It determines the direction of the message. A ‘zero’ in the least significant position of the first byte means that the master will write information to a selected slave. A ‘one’ in this position means that the master will read information from the slave.

When an address is sent, each device in a system compares the first seven bits after the START condition with its address. If they match, the device considers itself addressed by the master as a slave-receiver or slave-transmitter, depending on the R/W bit.

A slave address can be made-up of a fixed and a programmable part. Since it’s likely that there will be several identical devices in a system, the programmable part of the slave address enables the maximum possible number of such devices to be connected to the I^2C-bus. The number of programmable address bits of a device depends on the number of pins available. For example, if a device has 4 fixed and 3 programmable address bits, a total of 8 identical devices can be connected to the same bus.

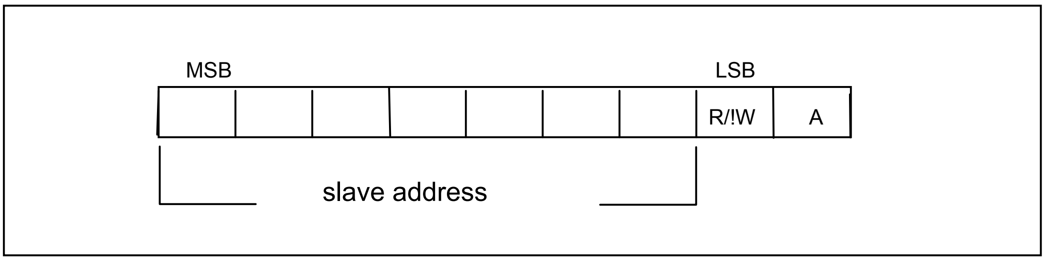


Figure 17-3 Normal 7 Bit Address after START Condition

2) General Call Address:

Address byte with all bits are “0” is defined as “general call address”. When this address is used, all devices should, in theory, respond with an acknowledge. However, if a device doesn’t need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgment. If a device does require data from a general call address, it will acknowledge this address and behave as a slave- receiver. The second and following bytes will be acknowledged by every slave-receiver capable of handling this data. A slave that cannot process one of these bytes must ignore it by not-acknowledging.

The second byte of the general call address then defines the action to be taken.

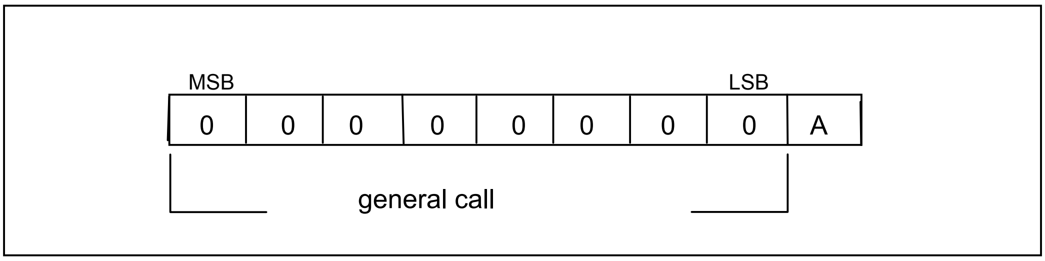


Figure 17-4 General Call Address after START Condition

3) START Byte Address:

START Byte:

After the START condition S has been transmitted by the master, data transfer can be preceded by a start procedure which is much longer than normal. The start procedure consists of:

- A START condition (S)
- A START byte (00000001)
- An acknowledge clock pulse (ACK)*
- A repeated START condition (Sr)

Note: An acknowledge-related clock pulse is generated after the START byte. This is present only to conform to the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

When the START byte (00000001) is transmitted, another microcontroller (the slave) can therefore sample the SDA line at a low sampling rate (also determined by the I2CGR) until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr which is then used for synchronization.

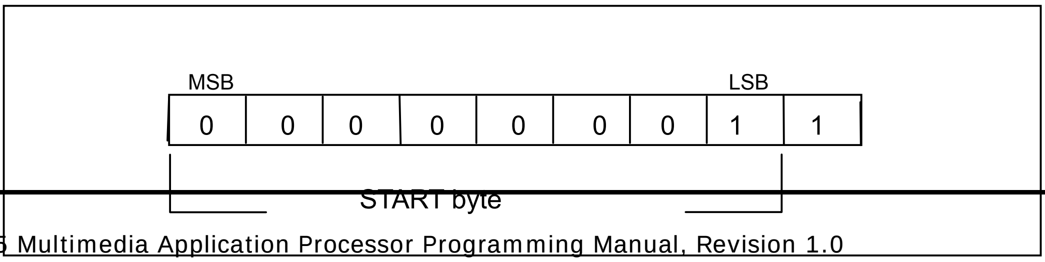


Figure 17-5 START Byte after START Condition

17.4.5.2 Transfer Format

A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed.
- Master reads slave immediately after first byte. At the moment of the first acknowledge, the master-transmitter becomes a master- receiver and the slave-receiver becomes a slave-transmitter.
- This first acknowledge is still generated by the slave. The STOP condition is generated by the master, which has previously sent a not-acknowledge.

Notes:

1. Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the START condition and slave address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations etc. are taken by the designer of the device.
3. Each byte is followed by an acknowledgment bit as indicated by the 'A 'or '!A ' blocks in the sequence.

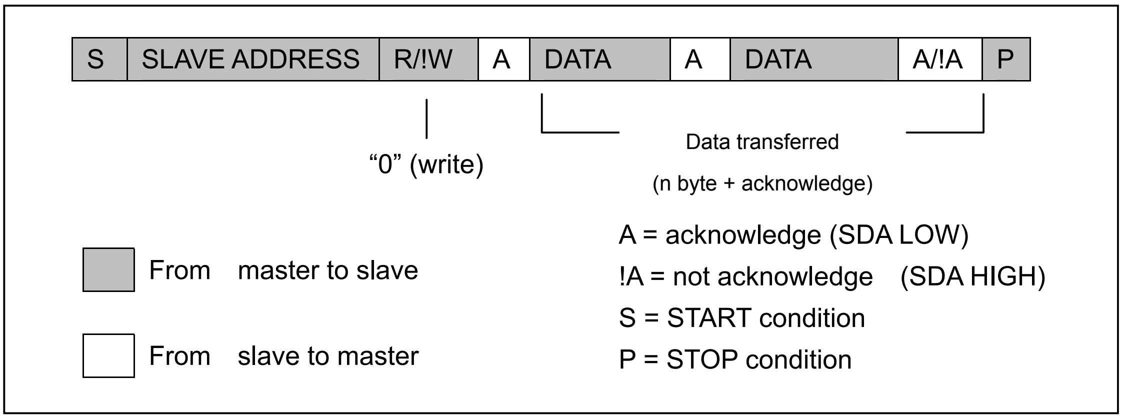

S
SLAVE ADDRESS
R/!W
A
DATA
A
DATA
A/!A
P
"0" (write)
Data transferred
(n byte + acknowledge)
From master to slave
From slave to master
A = acknowledge (SDA LOW)
!A = not acknowledge (SDA HIGH)
S = START condition
P = STOP condition

Figure 17-6 A Master-Transmitter Addresses a Slave Receiver with a 7-Bit Address

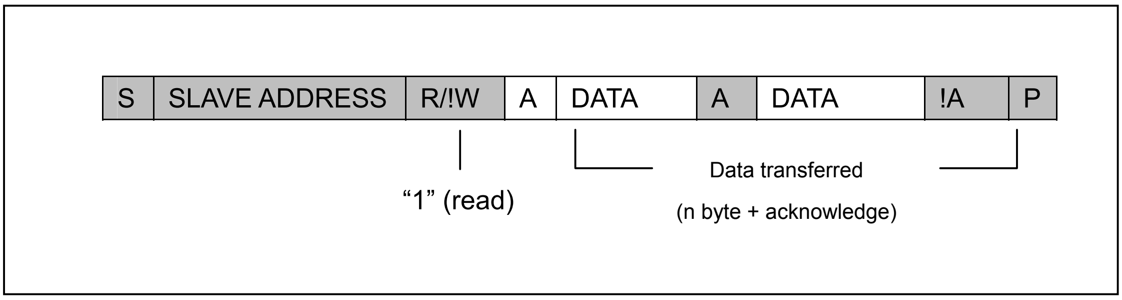

S
SLAVE ADDRESS
R/!W
A
DATA
A
DATA
!A
P
“1” (read)
Data transferred
(n byte + acknowledge)

Figure 17-7 A Master Reads the Slave Immediately after the First Byte (Master-Receiver)

17.5 I2C Operation

17.5.1 I2C Initialization

Before transmitting and receiving data, set the I2CE bit in I2CCR to 1 to enable I2C operation and set I2CGR for proper serial clock frequency. Set the I2CE bit to 0 after transmitting or receiving data for low power dissipation.

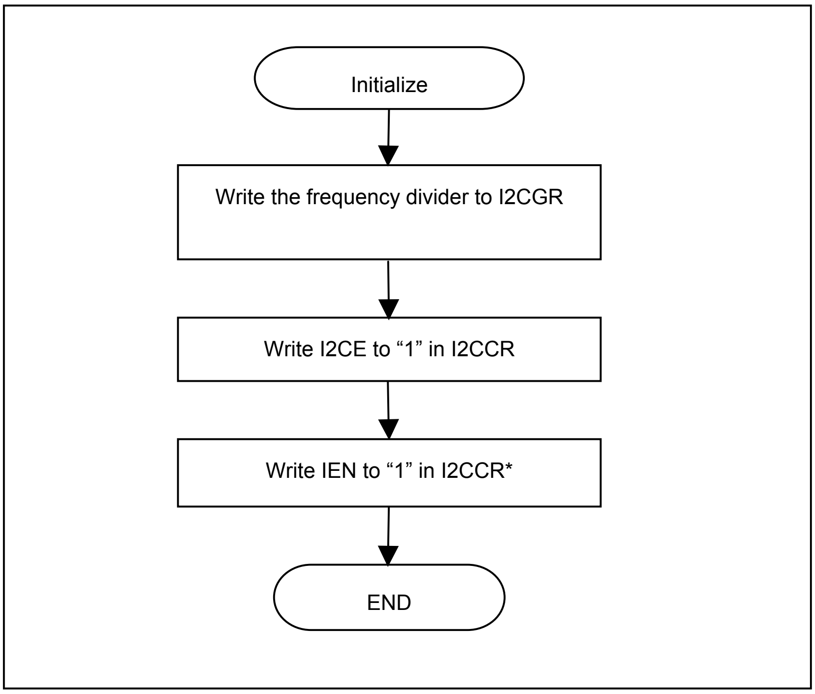
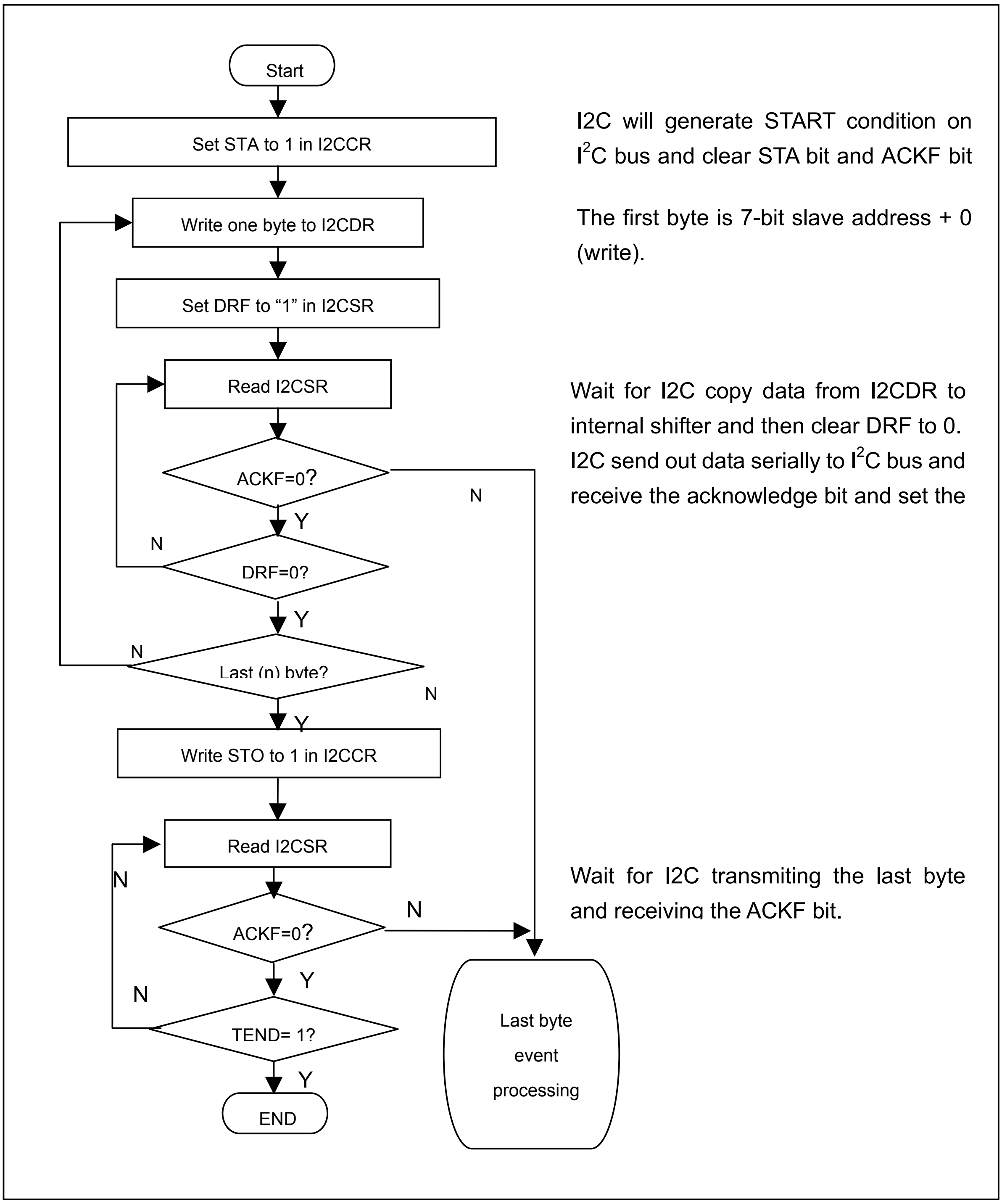

Initialize
Write the frequency divider to I2CGR
Write I2CE to “1” in I2CCR
Write IEN to “1” in I2CCR*
END

Figure 17-8 I2C Initialization

Note: This step is selectable.

17.5.2 Write Operation

Following figure illustrates the flow of a write operation.


Start
Set STA to 1 in I2CCR
Write one byte to I2CDR
Set DRF to "1" in I2CSR
Read I2CSR
ACKF=0?
Y
N
DRF=0?
Y
N
Last (n) byte?
N
Y
Write STO to 1 in I2CCR
Read I2CSR
N
ACKF=0?
N
Y
N
TEND= 1?
Y
END
Last byte event processing

I2C will generate START condition on I^2C bus and clear STA bit and ACKF bit

The first byte is 7-bit slave address + 0 (write).

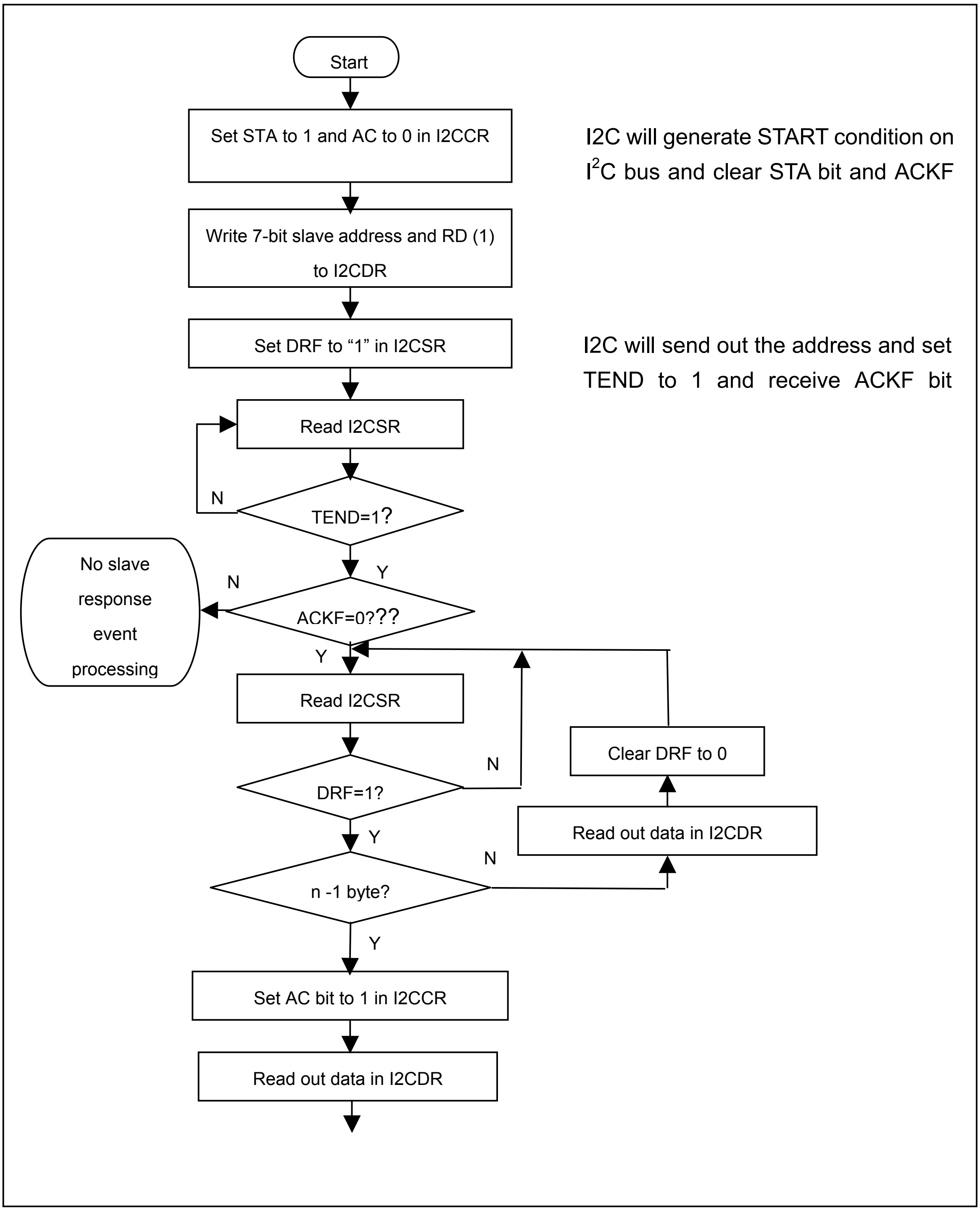
Wait for I2C copy data from I2CDR to internal shifter and then clear DRF to 0. I2C send out data serially to I^2C bus and receive the acknowledge bit and set the

Wait for I2C transmiting the last byte and receiving the ACKF bit.

Figure 17-9 I2C Write Operation Flowchart

17.5.3 Read Operation

Following figure illustrates the flow of read operation.


Start
Set STA to 1 and AC to 0 in I2CCR
Write 7-bit slave address and RD (1) to I2CDR
Set DRF to “1” in I2CSR
Read I2CSR
N
TEND=1?
Y
No slave response event processing
N
ACKF=0???
Y
Read I2CSR
N
DRF=1?
Y
Clear DRF to 0
Read out data in I2CDR
N
n -1 byte?
Y
Set AC bit to 1 in I2CCR
Read out data in I2CDR

I2C will generate START condition on I^2C bus and clear STA bit and ACKF

I2C will send out the address and set TEND to 1 and receive ACKF bit

Figure 17-10 I2C Read Operation Flowchart

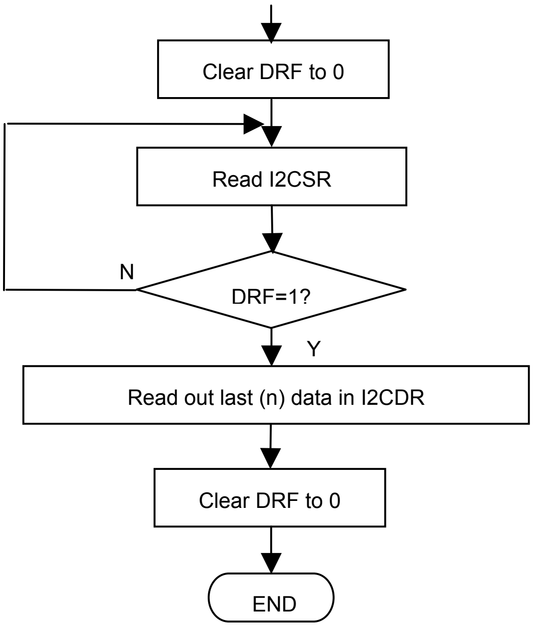

Clear DRF to 0
Read I2CSR
N
DRF=1?
Y
Read out last (n) data in I2CDR
Clear DRF to 0
END

Figure 17-11 Read Operation Flowchart (cont.)

18 UART Interface

This chapter describes the universal asynchronous receiver/transmitter (UART) serial ports. There are two UARTs: All UARTs use the same programming model. Each of the serial ports can operate in interrupt based mode or DMA-based mode.

The Universal asynchronous receiver/transmitter (UART) is compatible with the 16550 industry standard and can be used as slow infrared asynchronous interface that conforms to the Infrared Data Association (IrDA) serial infrared specification 1.1.

18.1 Overview

- Full-duplex operation
- 5-, 6-, 7- or 8-bit characters with optional no parity or even or odd parity and with 1, 1½, or 2 stop bits
- 16x8bit transmit FIFO and 16x11bit receive FIFO
- Independently controlled transmit, receive (data ready or timeout), line status interrupts
- Baud rate generation allows up to 230.4Kbps
- Internal diagnostic capability Loopback control and break, parity, overrun and framing-error is provided
- Separate DMA requests for transmit and receive data services in FIFO mode
- Modem Control Functions are provided
- Slow infrared asynchronous interface that conforms to IrDA specification

18.2 Pin Description

Table 18-1 UART Pins Description

Name	Type	Description
RxD	Input	Receive data input
TxD	Output	Transmit data output
CTS_	Input	Clear to Send — Modem Transmission enabled
RTS_	Output	Request to Send — UART Transmission request

18.3 Register Description

All UART register 32-bit access address is physical address. When ULCR.DLAB is 0, URBR, UTHR and UIER can be accessed; When ULCR.DLAB is 1, UDLLR and UDLHR can be accessed.

Table 18-2 UART Registers Description

Name	Description	RW	Reset Value	Address	Access Size
URBR0	UART Receive Buffer Register 0	R	0x??	0x10030000	8
UTHR0	UART Transmit Hold Register 0	W	0x??	0x10030000	8
UDLLR0	UART Divisor Latch Low Register 0	RW	0x00	0x10030000	8
UDLHR0	UART Divisor Latch High Register 0	RW	0x00	0x10030004	8
UIER0	UART Interrupt Enable Register 0	RW	0x00	0x10030004	8
UIIR0	UART Interrupt Identification Register 0	R	0x01	0x10030008	8
UFCR0	UART FIFO Control Register 0	W	0x00	0x10030008	8
ULCR0	UART Line Control Register 0	RW	0x00	0x1003000C	8
UMCR0	UART Modem Control Register 0	RW	0x00	0x10030010	8
ULSR0	UART Line Status Register 0	R	0x00	0x10030014	8
UMSR0	UART Modem Status Register 0	R	0x00	0x10030018	8
USPR0	UART ScratchPad Register 0	RW	0x00	0x1003001C	8
ISR0	Infrared Selection Register 0	RW	0x00	0x10030020	8
URBR1	UART Receive Buffer Register 1	R	0x??	0x10031000	8
UTHR1	UART Transmit Hold Register 1	W	0x??	0x10031000	8
UDLLR1	UART Divisor Latch Low Register 1	RW	0x00	0x10031000	8
UDLHR1	UART Divisor Latch High Register 1	RW	0x00	0x10031004	8
UIER1	UART Interrupt Enable Register 1	RW	0x00	0x10031004	8
UIIR1	UART Interrupt Identification Register 1	R	0x01	0x10031008	8
UFCR1	UART FIFO Control Register 1	W	0x00	0x10031008	8
ULCR1	UART Line Control Register 1	RW	0x00	0x1003100C	8
UMCR1	UART Modem Control Register 1	RW	0x00	0x10031010	8
ULSR1	UART Line Status Register 1	R	0x00	0x10031014	8

UMSR1	UART Modem Status Register 1	R	0x00	0x10031018	8
USPR1	UART ScratchPad Register 1	RW	0x00	0x1003101C	8
ISR1	Infrared Selection Register 1	RW	0x00	0x10031020	8
URBR2	UART Receive Buffer Register 2	R	0x??	0x10032000	8
UTHR2	UART Transmit Hold Register 2	W	0x??	0x10032000	8
UDLLR2	UART Divisor Latch Low Register 2	RW	0x00	0x10032000	8
UDLHR2	UART Divisor Latch High Register 2	RW	0x00	0x10032004	8
UIER2	UART Interrupt Enable Register 2	RW	0x00	0x10032004	8
UIIR2	UART Interrupt Identification Register 2	R	0x01	0x10032008	8
UFCR2	UART FIFO Control Register 2	W	0x00	0x10032008	8
ULCR2	UART Line Control Register 2	RW	0x00	0x1003200C	8
UMCR2	UART Modem Control Register 2	RW	0x00	0x10032010	8
ULSR2	UART Line Status Register 2	R	0x00	0x10032014	8
UMSR2	UART Modem Status Register 2	R	0x00	0x10032018	8
USPR2	UART ScratchPad Register 2	RW	0x00	0x1003201C	8
ISR2	Infrared Selection Register 2	RW	0x00	0x10032020	8
URBR3	UART Receive Buffer Register 3	R	0x??	0x10033000	8
UTHR3	UART Transmit Hold Register 3	W	0x??	0x10033000	8
UDLLR3	UART Divisor Latch Low Register 3	RW	0x00	0x10033000	8
UDLHR3	UART Divisor Latch High Register 3	RW	0x00	0x10033004	8
UIER3	UART Interrupt Enable Register 3	RW	0x00	0x10033004	8
UIIR3	UART Interrupt Identification Register 3	R	0x01	0x10033008	8
UFCR3	UART FIFO Control Register 3	W	0x00	0x10033008	8
ULCR3	UART Line Control Register 3	RW	0x00	0x1003300C	8
UMCR3	UART Modem Control Register 3	RW	0x00	0x10033010	8
ULSR3	UART Line Status Register 3	R	0x00	0x10033014	8
UMSR3	UART Modem Status Register 3	R	0x00	0x10033018	8
USPR3	UART ScratchPad Register 3	RW	0x00	0x1003301C	8
ISR3	Infrared Selection Register 3	RW	0x00	0x10033020	8

18.3.1 UART Receive Buffer Register (URBR)

The read-only URBR is corresponded to one level 11bit buffer in non-FIFO mode and a 16x11bit FIFO that holds the character(s) received by the UART. Bits in URBR are right-justified when being configured to use fewer than eight bits, and the rest of most significant data bits are zeroed and the most significant three bits of each buffer are the status for the character in the buffer. If ULSR.DRY is 0, don't read URBR, otherwise wrong operation may occur.

URBR0, URBR1, URBR2, URBR3

0x10030000, 0x10031000,
0x10032000, 0x10033000 (DLAB = 0)

Bit		7	6	5	4	3	2	1	0

	URBR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	URBR	8-bit UART receive read data	R

18.3.2 UART Transmit Hold Register (UTHR)

The write-only UTHR is corresponded to one leve 8 bit buffer in non-FIFO mode and a 16x8bit FIFO in FIFO mode that holds the data byte(s) to be transmitted next.

UTHR0, UTHR1, UTHR2, UTHR3

0x10030000, 0x10031000,
0x10032000, 0x10033000 (DLAB = 0)

Bit	7	6	5	4	3	2	1	0
	UTHR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	UTHR	8-bit UART transmit write hold data	W

18.3.3 UART Divisor Latch Low/High Register (UDLLR / UDLHR)

UART Divisor Latch registers, UDLLR/UDLHR together compose the divisor for the programmable baud rate generator that can take the 3.6864-MHz fixed-input clock and divide it by 1 to (2^{16} – 1). UDLHR/UDLLR stores the high/low 8-bit of the divisor respectively. Load these divisor latches during initialization to ensure that the baud rate generator operates properly. If both Divisor Latch registers are 0, the 16X clock stops. The maximum baud rate is 230.4kbps.

The relationship of baud rate and the value of Divisor is shown by the formula:

Baud Rate = 3.6864 Mhz / (16 * **Divisor**)

UDLLR0, UDLLR1, UDLLR2, UDLLR3

0x10030000, 0x10031000,
0x10032000, 0x10033000 (DLAB = 1)

Bit	7	6	5	4	3	2	1	0

	7	6	5	4	3	2	1	0
	Divisor Latch Low 8-bit							
RST	0	0	0	0	0	0	0	0

UDLHR0, UDLHR1, UDLHR2, UDLHR3

0x10030004, 0x10031004,
0x10032004, 0x10033004 (DLAB = 1)

Bit	7	6	5	4	3	2	1	0
	Divisor Latch High 8-bit							
RST	0	0	0	0	0	0	0	0

18.3.4 UART Interrupt Enable Register (UIER)

The UART Interrupt Enable Register (UIER) contains the interrupt enable bits for the five types of interrupts (receive data ready, timeout, line status, and transmit data request, and modem status) that set a value in UIIR.

UIER0, UIER1, UIER2, UIER3

0x10030004, 0x100301004,
0x10032004, 0x10033004 (DLAB = 0)

Bit	7	6	5	4	3	2	1	0
	Reserved			RTOIE	MSIE	RLSIE	TDRIE	RDRIE
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Always read 0, write is ignored	R
4	RTOIE	**Receive Timeout Interrupt Enable** 0 = Disable the receive timeout interrupt 1 = Enable the receive timeout interrupt Timeout means the URDR (FIFO mode) is not empty but no character has received for a period of time T: T (bits) = 4 X Word length + 12	RW
3	MSIE	**Modem Status Interrupt Enable** 0 = Disable the modem status interrupt 1 = Enable the modem status interrupt	RW
2	RLSIE	**Receive Line Status Interrupt Enable** 0 = Disable receive line status interrupt 1 = Enable receive line status interrupt	RW

1	TDRIE	**Transmit Data Request Interrupt Enable** 0 = Disable the transmit data request interrupt 1 = Enable the transmit data request interrupt	RW
0	RDRIE	**Receive Data Ready Interrupt Enable** 0 = Disable the receive data ready interrupt 1 = Enable the receive data ready interrupt	RW

18.3.5 UART Interrupt Identification Register (UIIR)

The read-only UART Interrupt Identification Register (UIIR) records the prioritized pending interrupt source information. Its initial value after power-on reset is 0x01.

UIIR0, UIIR1, UIIR2, UIIR3 **0x10030008, 0x10031008, 0x10032008, 0x10033008**

Bit	7	6	5	4	3	2	1	0
	FFMSEL		Reserved		INID			INPEND
RST	0	0	0	0	0	0	0	1

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th><th>RW</th></tr>
<tr><td>7:6</td><td>FFMSEL</td><td>FIFO Mode Select
0b00 = Non-FIFO mode
0b01 = Reserved
0b10 = Reserved
0b11 = FIFO mode</td><td>R</td></tr>
<tr><td>5:4</td><td>Reserved</td><td>Always read 0, write is ignored</td><td>R</td></tr>
<tr><td>3:1</td><td>INID</td><td>Interrupt Identifier
These bits identify the current highest priority pending interrupt.
<table>
<tr><td>INID</td><td>Description</td></tr>
<tr><td>0b000</td><td>Modem Status</td></tr>
<tr><td>0b001</td><td>Transmit Data Request</td></tr>
<tr><td>0b010</td><td>Receive Data Ready</td></tr>
<tr><td>0b011</td><td>Receive Line Status</td></tr>
<tr><td>0b100</td><td>Reserved</td></tr>
<tr><td>0b101</td><td>Reserved</td></tr>
<tr><td>0b110</td><td>Receive Time Out</td></tr>
<tr><td>0b111</td><td>Reserved</td></tr>
</table>
See Table 18-3 for details</td><td>R</td></tr>
</table>

0	INPEND	**Interrupt Pending** 0 = interrupt is pending 1 = No interrupt pending	R

Table 18-3 UART Interrupt Identification Register Description

UIIR.INID	Interrupt Set/Clear Cause			
	Priority	**Type**	**Source**	**Clear Condition**
0b0001	—	None	No pending interrupt	—
0b0110	1st Highest	Receive Line Status	Overrun, Parity, Frame Error, Break Interrupt, and FIFO Error (DMA mode only)	Reading ULSR or empty all the error characters in DMA mode
0b0100	2nd Highest	Receive Data Ready	FIFO mode: Trigger threshold was reached Non-FIFO mode: URBR full	FIFO mode: Reading URBR till below trigger threshold. Non-FIFO mode: Empty URBR
0b1100	2nd Highest	Receive Timeout	FIFO mode only: URBR not empty but no data read in for a period of time	Reset receive buffer by setting UFCR.RFRT to 1 or Reading URBR
0b0010	3rd Highest	Transmit Data Request	FIFO mode: Empty location in UTHR equal to half or more than half Non-FIFO mode: UTHR empty	FIFO mode: Data number in UTHR more than half Non-FIFO mode: Writing UTHR
0b0000	4th Highest	Modem Status	Modem CTS_ pin status change	Reading UMSR

18.3.6 UART FIFO Control Register (UFCR)

The write-only register UFCR contains the control bits for receive and transmit FIFO.

UFCR0, UFCR1, UFCR2, UFCR3 **0x10030008, 0x10031008, 0x10032008, 0x1003300**

Bit	7	6	5	4	3	2	1	0
	RDTR		Reserved	UME	DME	TFRT	RFRT	FME
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	RDTR	**Receive Buffer Data Number Trigger** These bits are used to select the trigger level for the receive data ready	W

		interrupt in FIFO mode. 0b00 = 1 0b01 = 4 0b10 = 8 0b11 = 15	
5	Reserved	Always read 0, write is ignored	R
4	UME	**UART Module Enable** 0 = Disable UART 1 = Enable UART	W
3	DME	**DMA Mode Enable** 0 = Disable DMA mode 1 = Enable DMA mode	W
2	TFRT	**Transmit Holding Register Reset** 0 = Not reset 1 = Reset transmit FIFO	W
1	RFRT	**Receive Buffer Reset** 0 = Not reset 1 = Reset receive FIFO	W
0	FME	**FIFO Mode Enable** Set this bit before the trigger levels. 0 = non-FIFO mode 1 = FIFO mode	W

18.3.7 UART Line Control Register (ULCR)

The ULCR defines the format for UART data transmission.

ULCR0. ULCR1, ULCR2, ULCR3

0x1003000C, 0x1003100C
0x1003200C, 0x1003300C

Bit	7	6	5	4	3	2	1	0
	DLAB	SBK	STPAR	PARM	PARE	SBLS	WLS	
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	DLAB	**Divisor Latch Access Bit** 0 = Enable to access URBR, UTHR or UIER 1 = Enable to access UDLLR or UDLHR	W
6	SBK	**Set Break** Causes a break condition (at least one 0x00 data) to be transmitted to the	W

		receiving UART. Acts only on the TXD pin and has no effect on the transmit logic. 0 = No effect on TXD output 1 = Forces TXD output to 0	
5	STPAR	**Sticky Parity** Setting this bit forces parity location to be opposite of PARM bit when PARE is 1 (it is ignored when PARE is 0). 0 = Disable Sticky parity 1 = Enable Sticky parity (opposite o f PARM bit)	W
4	PARM	**Parity Odd/Even Mode Select** If PARE = 0, PARM is ignored 0 = Odd parity 1 = Even parity	W
3	PARE	**Parity Enable** Enables a parity bit to be generated on transmission or checked on reception. 0 = No parity 1 = Parity	W
2	SBLS	**Stop Bit Length Select** Specifies the number of stop bits transmitted and received in each character. When receiving, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 2 stop bits, except for 5-bit character then 1-1/2 bits	W
1:0	WLS	**Word Length Select** 0b00 = 5-bit character 0b01 = 6-bit character 0b10 = 7-bit character 0b11 = 8-bit character	W

18.3.8 UART Line Status Register (ULSR)

The read-only ULSR indicates status information during the data transfer. Receive error information in ULSR[4:1] remains set until software reads ULSR and it must be read before the error character is read.

ULSR0, ULSR1, ULSR2, ULSR3 **0x10030014, 0x10031014, 0x10032014, 0x10033014**

Bit	7	6	5	4	3	2	1	0
	FIFOE	TEMP	TDRQ	BI	FMER	PARER	OVER	DRY
RST	0	1	1	0	0	0	0	0

Bits	Name	Description	RW
7	FIFOE	**FIFO Error Status (FIFO mode only)** FIFOE is set when there is at least one kind of receive error (parity, frame, overrun, break) for any of the characters in receive buffer. FIFOE is reset when all error characters are read out of the buffer. During DMA transfer, the error interrupt generates when FIFOE is 1, and no receive DMA request generates even when data in receive buffer reaches the trigger threshold until all the error characters are read out. In non-DMA mode, FIFOE set does not generate error interrupt. 0 = No error data in receive buffer or non-FIFO mode 1 = One or more error character in receive buffer	R
6	TEMP	**Transmit Holding Register Empty** Set when both UTHR and shift register are empty. It is cleared when either the UTHR or the shift register contains a data character. 0 = There is data in the transmit shifter and UTHR 1 = All the data in the transmit shifter and UTHR has been shifted out	R
5	TDRQ	**Transmit Data Request** Set when UTHR has half or more empty location (FIFO mode) or empty (non-FIFO mode). When both UIER.TDRIE and TDRQ are 1, transmit data request interrupt generates or during DMA transfer, DMA request to the DMA controller generates when UIER.TDRIE is 0 and TDRQ is 1. 0 = There is one (non-FIFO mode) or more than half data (FIFO mode) in UTHR 1 = None data (non-FIFO mode) or half or less than half data (FIFO mode) in UTHR	R
4	BI	**Break Interrupt** BI is set when the received data input is held low for longer than a full-word transmission time (the total time of start bit + data bits + parity bit + stop bits). BI is cleared when the processor reads the ULSR. In FIFO mode, only one character equal to 0x00 is loaded into the FIFO regardless of the length of the break condition. BI shows the break condition for the character at the front of the FIFO, not the most recently received character. 0 = No break signal has been received 1 = Break signal received	R
3	FMER	**Framing Error** Set when the bit following the last data bit or parity bit is detected to be 0. If the ULCR had been set for two or one and half stop bits, the other stop	R

		bits are not checked except the first one. In FIFO mode, FMER shows a framing error for the character at the front of the receive buffer, not for the most recently received character. Cleared when the processor reads the ULSR. 0 = No framing error 1 = Invalid stop bit has been detected	
2	PARER	**Parity Error** Indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. PARER is set upon detection if a parity error and is cleared when the processor reads the ULSR. In FIFO mode, PARER shows a parity error for the character at the front of the FIFO, not the most recently received character. 0 = No parity error 1 = Parity error has occurred	R
1	OVER	**Overrun Error** Set when both receive buffer and shifter are full and new data is received which will be lost. Cleared when the processor reads the ULSR. 0 = No data has been lost 1 = Receive data has been lost	R
0	DRY	**Data Ready** Set when a complete incoming character has been received into the Receive Buffer registers. DRY is cleared when the receive buffer is read (non-FIFO mode) or when the buffer is empty or when the buffer is reset by setting UFCR.RFRT to 1. 0 = No data has been received 1 = Data is available in URBR	R

18.3.9 UART Modem Control Register (UMCR)

The UMCR uses the modem control pins RTS_ and CTS_ to control the interface with a modem or data set. UMCR also controls the loopback mode. Loopback mode must be enabled before the UART is enabled.

UMCR0, UMCR1, UMCR2, UMCR3 **0x10030010, 0x10031010, 0x10032010, 0x10033010**

Bit	7	6	5	4	3	2	1	0
	MDCE	Reserved		LOOP	Reserved	Reserved	RTS	Reserved
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW

7	MDCE	**Modem Control Enable** 0 = Modem function is disabled 1 = Modem function is enabled	W
6:5	Reserved	Always read 0, write is ignored	R
4	LOOP	**Loop Back** This bit is used for diagnostic testing of the UART. When LOOP is 1, TXD output pin is set to a logic 1 state, RXD is disconnected from the pin, and the output of the transmitter shifter register is looped back into the receiver shift register input internally, similar to CTS_ and RTS_ pins and the RTS bit of the UMCR is connected to CTS bit of UMSR respectively. Loopback mode must be selected before the UART is enabled. 0 = Normal operation mode 1 = Loopback-mode UART operation	W
3	Reserved	Always read 0, write is ignored	R
2	Reserved	Always read 0, write is ignored	R
1	RTS	**Request To Send** This bit can control the RTS_ output state. 0 = RTS_ force to high 1 = RTS_ force to low	W
0	Reserved	Always read 0, write is ignored	R

18.3.10 UART Modem Status Register (UMSR)

The read-only UMSR provides the current state of the control lines from the modem to the processor. They are cleared when the processor reads UMSR.

UMSR0, UMSR1, UMSR2, UMSR3 **0x10030018, 0x10031018, 0x10032018, 0x10033018**

Bit	7	6	5	4	3	2	1	0
	Reserved			CTS	Reserved			CCTS
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	Reserved	Always read 0, write is ignored	R
6	Reserved	Always read 0, write is ignored	R
5	Reserved	Always read 0, write is ignored	R
4	CTS	**Status of Clear To Send** When MDCE bit is 1, this bit is the complement of CTS_ input. If Loop bit of UMCR is 1, this bit is equivalent to RTS bit of UMCR.	R

		0 = CTS_ pin is 1 1 = CTS_ pin is 0	
3	Reserved	Always read 0, write is ignored	R
2	Reserved	Always read 0, write is ignored	R
1	Reserved	Always read 0, write is ignored	R
0	CCTS	**Change status of CTS_** When MDCE bit is 1, this bit indicates the state change on CTS_ pin. 0 = No state change on CTS_ pin since last read of UMSR 1 = A change occurs on the state of CTS_ pin	R

18.3.11 UART Scratchpad Register

This Scratchpad register is used as a scratch register for the programmer and has no effect on the UART.

USPR **0x1003001C**

Bit	7	6	5	4	3	2	1	0
	Scratch Data							
RST	?	?	?	?	?	?	?	?

18.3.12 Infrared Selection Register (ISR)

The ISR is used to configure the slow-infrared (SIR) interface that is provided in each UART to support two-way wireless communication using infrared transmission that conforms to the IrDA serial infrared specification 1.1. The maximum frequency is up to 115.2kbps.

ISR **0x10030020**

Bit	7	6	5	4	3	2	1	0
	Reserved			RDPL	TDPL	XMODE	RCVEIR	XMITIR
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Always read 0, write is ignored	R
4	RDPL	**Receive Data Polarity** 0 = Slow-infrared (SIR) interface decoder takes positive pulses as zeros. 1 = SIR decoder takes negative pulses as zeros.	W

3	TDPL	**Transmit Data Polarity** 0 = SIR encoder generates a positive pulse for a data bit of zero. 1 = SIR encoder generates a negative pulse for a data bit of zero.	W
2	XMODE	**Transmit Pulse Width Mode** Set when the transmit encoder needs to generate 1.6us pulses (that are 3/16 of a bit-time at 115.2 kbps). Cleared when the transmit encoder needs to generate 3/16 of a bit-time wide according to current baud rate. 0 = Transmit pulse width is 3/16 of a bit-time wide. 1 = Transmit pulse width is 1.6 us.	W
1	RCVEIR	**Receiver SIR Enable** This bit is used to select the signal from the RXD pin is processed by the IrDA decoder before it is fed to the UART (RCVEIR = 1) or bypass IrDA decoder and is fed directly to the UART (RCVEIR = 0). 0 = Receiver is in UART mode. 1 = Receiver is in SIR mode.	W
0	XMITIR	**Transmitter SIR Enable** This bit is used to select TXD output pin is processed by the IrDA encoder before it is fed to the device pin (XMITIR = 1) or bypass IrDA encoder and is fed directly to the device pin (XMITIR = 0). Note: disable infrared LED before XMITIR is set, otherwise a false start bit may occur. 0 = Transmitter is in UART mode. 1 = Transmitter is in SIR mode.	W

18.3.13 Uart M Register (UMR)

UMSR0, UMSR1, UMSR2, UMSR3 **0x10030024, 0x10031024, 0x10032024, 0x10033024**

Bit	7	6	5	4	3	2	1	0
	▨	▨	▨		▨	▨	▨	
RST	0	0	0	0	0	0	0	0

M is the value of UMR register.

It will take UART at least M cycles for transmitter to transmit one bit and receiver to receive one bit ,

It will take UART at most M+1 cycles for transmitter to transmit one bit and receiver to receive one bit,

18.3.14 Uart Add Cycle Register(UACR)

UMSR0, UMSR1, UMSR2, UMSR3

Bit	12	11	10	9	8	7	6	5	3	2	1
RST						0	0	0	0	0	0

If Nth bit of the register is 1 ,It will take Uart M+1 cycles to transmit or receive the bit of date for transmit or receive.

If the register is 12'h0 ,UART will receive or transmit a bit by M cycle .

If the register is 12'hfff ,UART will receive or transmit a bit by M+1 cycle .

For the detail to see 1.4.8For any frequency clock to use the Uart

18.4 Operation

The following sections describe the UART operations that include flow of configuration, data transmission, data reception, and Infra-red mode.

18.4.1 UART Configuration

Before UART starts to transfer data or changing transfer format, configuration must be done to define the transfer format. The sample flow is as the following:

In FIFO mode, set FME bit of UFCR to 1, reset receive and transmit FIFO, then initialize the UART as described below.

1. Clear UFCR.UME to 0
2. Set value in UDLL/UDHR to generate the baud rate clock
3. Set data format in ULCR
4. If it is in FIFO MODE, set FME bit and other FIFO control in UFCR, reset receive and transmit FIFO, otherwise skip item 4
5. Set each interrupt enable bit in UIER in interrupt-based transfer or set UFCR.DME in DMA-based transfer (DMA transfer is FIFO mode only), then set UFCR.UME

18.4.2 Data Transmission

After configuration, UART is ready for data transfer. For data transmission, refer to the following procedure:

1. Read ULSR.TDRQ (interrupt disable) or wait for transmit data request interrupt (interrupt enable), if TDRQ = 1 or transmit data request interrupt generates, that means there is enough empty location in UTHR for new data
2. If ULSR.TDRQ is 1 or get the transmit data request interrupt, write transmit data to UTHR to start transmission
3. Do item 1 and item 2 if there are more data waiting for transmit
4. After all necessary data are written to UTHR, wait ULSR.TEMP = 1, that means all data completely transmitted
5. If it is necessary to send break, set ULCR.SBK and at least wait for 1-bit interval time to send a valid break, then clear ULCR.SBK
6. Clear UME bit to finish UART transmission

18.4.3 Data Reception

After configuration, UART is ready for data transfer. For data reception, refer to the following sample procedure:

1. Read ULSR.DRY (interrupt disable) or wait for receive data request interrupt (interrupt enable), if ULSR.DRY =1 or receive data request interrupt generates, that means URBR has one data (non-FIFO mode) or data in URBR reaches the trigger value (FIFO mode)
2. If ULSR.DRY = 1 or receive data request interrupt generates, then read ULSR.FIFOE or see if there is error interrupt, if FIFOE = 1, it means received data has receive error, then go to error handler, other wise go to item 3
3. Read one received data in URBR (non-FIFO mode) or data equal to trigger value in URBR (FIFO mode)
4. Check whether all data received: check whether ULSR.DRY = 0, in FIFO mode and interrupt is enabled, timeout interrupt may generate, when timeout interrupt generates, read URBR till ULSR.DRY = 0
5. Clear UFCR.UME to end data reception when all data are received and ULSR.DRY = 0

18.4.4 Receive Error Handling

A sample error handling flow is as the following:

1. If ULSR.FIFOE = 1, it means there is receive error in received data, then check what error it is
2. If ULSR.OVER = 1, go to OVER error handling
3. If ULSR.BI = 1, go to Break handling
4. If ULSR.FMER = 1, go to Frame error handling
5. If PARER = 1, go to PARER error handling

18.4.5 Modem Transfer

When UMCR.MDCE = 1, modem control is enabled. Transfer flow can be stopped and restarted by software through RTS_ and CTS_ pin. When UART transmitter detects low level on CTS_ pin, it stops transmission and TxD pin goes to mark state after finishing transmitting the current character until it detects CTS_ pin goes back to high level. RTS_ pin is output to receiving UART and its state can be controlled by setting UMCR.RTS bit, that is, setting UMCR.RTS to 1, RTS_ pin is low level output that means UART is ready to receive data, on the contrary, it means UART currently can’t receive more data.

18.4.6 DMA Transfer

UART can operate in DMA-based (UFCR.DME = 1, FIFO mode only), that is, dma request initiated by UART takes the place of interrupt request and transmission/reception is carried out using DMA instead of CPU. Be sure that software guarantee to disable transmit and receive interrupt except timeout and error interrupts.

During DMA transfer, if an interrupt occurs, software must first read the ULSR to see if an error interrupt exists, then check the UIIR for the source of the interrupt and if DMA channel is already halt because of the error indicator from UART, then disable DMA channel and read out all the error data from receive FIFO. Software re-set and re-enable DMA and data transfer by DMA will re-start.

18.4.7 Slow IrDA Asynchronous Interface

Each UART supports slow infra-red (SIR) transmission and reception by setting ISR.XMITIR and ISR.RCVEIR to 1 (make sure the two bits are not set to 1 at the same time because SIR can’t operate full-duplex). According to the IrDA 1.1, data rate is limited at a maximum value of 115.2Kbps.

In SIR transmit mode, the transmit pulse comes out at a rate of 3/16 (when the transmit data bit is zero); in SIR receive mode, the receiver must detect the 3/16 pulsed period to recognize a zero value (an active high or low pulse is demodulation to 0, and no pulse is demodulation to 1).

Compared to normal UART, there are some limitations to SIR, that is, each character is fixed to 8-bit data width, no parity and 1 stop bit and modem function is ignored. The IrDA 1.1 specifies a minimum 10ms latency after an optical node ceases transmitting before its receiver recovers its receiving function and software must guarantee this delay.

In the IrDA 1.1 specification, communication must start up at the rate of 9600bps, but then allows the link to negotiate higher (or lower) data rates if supported by both ends. However, the communication rate will not automatically change. Change, if necessary, is performed by software.

18.4.8 For any frequency clock to use the Uart

Note: if you don't set M register and UACR the uart work at normal mode with the specified frequencies. To use other frequency you should to set Mregister and UACR to right value .

1.The Improving

Following changes are made

- One bit is composed by M CLK_{BR} cycles, which can be 4~1024
- Some extra CLK_{BR} cycles can be inserted in some bits in one frame, so that like M has fraction.

For instance:

$CLK_{BR} = CLK_{DEV} / N$ N = 1, 2, …
$CLK_{BR} = CLK_{DEV} = 4MHz$
Band rate = 460800

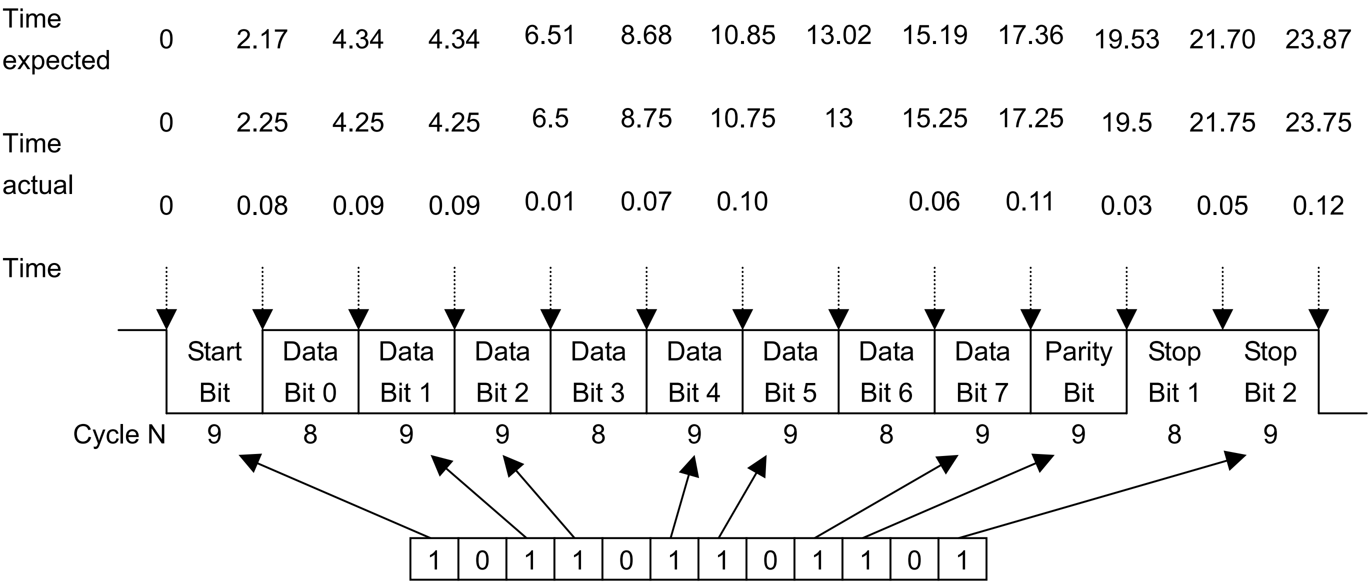
In accurate

$M_a = 8.681$

We take

M = 8, with 8 extra cycles in every frame

A 12-bit register is used to indicate where to insert the extra cycles


Time expected
0 2.17 4.34 4.34 6.51 8.68 10.85 13.02 15.19 17.36 19.53 21.70 23.87
Time actual
0 2.25 4.25 4.25 6.5 8.75 10.75 13 15.25 17.25 19.5 21.75 23.75
0 0.08 0.09 0.09 0.01 0.07 0.10 0.06 0.11 0.03 0.05 0.12
Time
Start Bit
Data Bit 0
Data Bit 1
Data Bit 2
Data Bit 3
Data Bit 4
Data Bit 5
Data Bit 6
Data Bit 7
Parity Bit
Stop Bit 1
Stop Bit 2
Cycle N
9 8 9 9 8 9 9 8 9 9 8 9
1 0 1 1 0 1 1 0 1 1 0 1

For transmission, in theory, the biggest error is half of CLK_{BR} cycle, which is 0.125us here.

2 To set UMR register

$CLK_{BR} = CLK_{DEV} / N$

$M_a = CLK_{BR}/\text{band rate}$

M is modem of M_a。

Write M to Mregister.

Considering the power and the robust quality, for M form 6 to 32 is you better select by set the UDLR.

The max error

$$\frac{0.5 / CLK_{BR}}{M_a / CLK_{BR}} = 0.5/M_a < 0.5/M$$

M	4	8	16	32	64
error/W_{bit}	12.5%	6.25%	3.125%	1.56%	0.78%

Table 1

3. To set UACR value

for each bit of it means:

0 :means not to add additional cycle to the the bit that uart is prepare to transmit or receive , in another word ,you will to use M cycles to transmit or receive the bit .

1 :means to add additional cycle to the the bit that uart is prepare to transmit or receive,in another word, you will to use M+1 cycles to transmit or receive the bit .

To set UACR value you must ensure that the max error of each bit should be less than $0.5P_{BR}$.

For example：M_a-M =0.15 ; M+1-M_a=0.85;

Write UMR 8

Write UMR 408

cycle/bit	：	M，	M，	M，	M+1，	M，	M，	M，	M，	M，	M，	M+1，	M
UACR	：	0	0	0	1	0	0	0	0	0	0	1	0

19 Image Process Unit

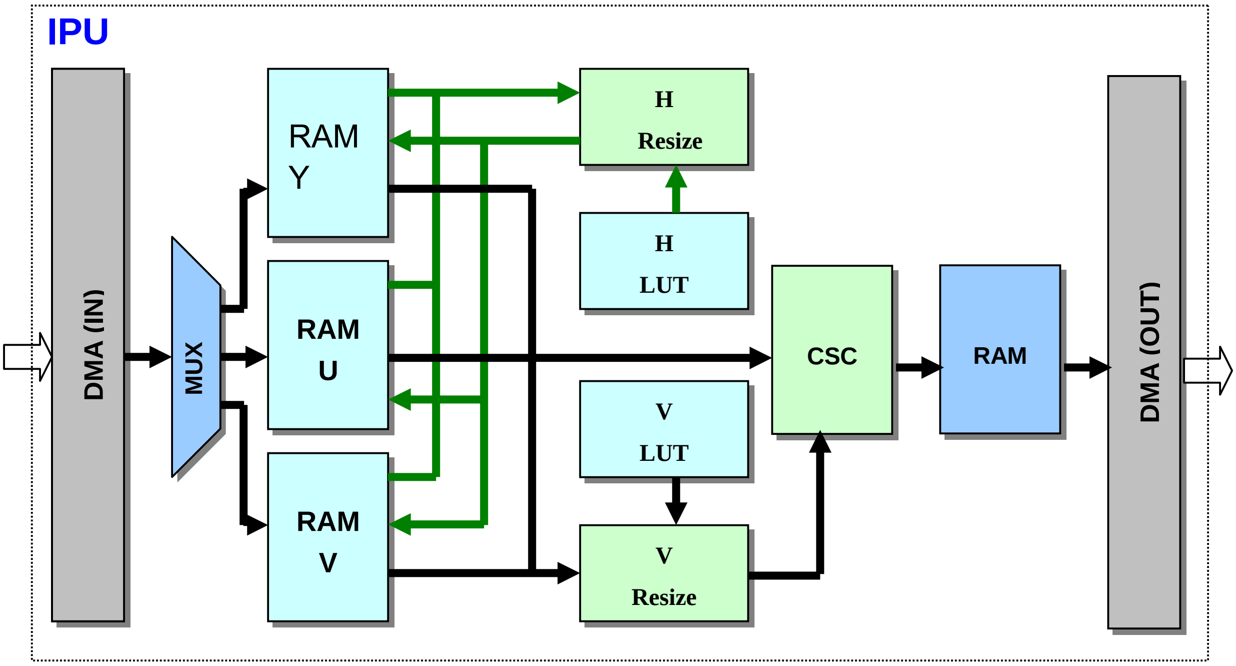
19.1 Overview

IPU (Image process unit) contains Resize and CSC (color space conversion), which is used for image post processing.

19.2 Features

- Input data: from external memory
- Input format: YUV /YCbCr (4:2:0, 4:2:2, 4:4:4, 4:1:1)
- Output format: RGB (565, 555, 888)
- Minimum input image size: 33x33
- Maximum input image size: 2047x2047
- Image resizing:
- ■ Up scaling ratios up to 1:2 in fractional steps
- ■ Down scaling ratios up to 20:1 in fractional steps

19.3 Block Diagram


IPU
DMA (IN)
MUX
RAM Y
RAM U
RAM V
H Resize
H LUT
V LUT
V Resize
CSC
RAM
DMA (OUT)

19.4 Data flow

19.4.1 Input Data

Y, U, V (or Y, Cb, Cr; following chapters use YUV for convenience) data would be load from external memory by DMA respectively.

19.4.2 Output Data

The data format after CSC is RGB (565, 555, 888), and the data would be stored back to the external memory by DMA.

19.4.3 Resize Coefficients LUT

The resize coefficients look up tables are preset by software according to specific format (see later chapter for detail). There are 2 tables support independent horizontal and vertical scaling. Each table has 20 entries that can accommodate up to 20 coefficients.

19.5 Register definition

19.5.1 IPU Control Register

IPU_CONTROL **0x0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							V_SCALE	H_SCALE					IPU_RST	FM_IRQ_EN	RSZ_EN	IPU_EN
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0	0

Bits	Name	Description	R/W
31:10	Reserved	Writing has no effect, read as zero.	R
9	V_SCALE	Vertical direction scale flag. 0: down scaling; 1: up scaling	RW
8	H_SCALE	Horizontal direction scale flag. 0: down scaling; 1: up scaling	RW
7:4	Reserved	Writing has no effect, read as zero.	R
3	IPU_RST	Reset IPU. Writing 1: reset IPU; 0: no effect. Read as zero	W
2	FM_IRQ_EN	Frame process finish interrupt enable. 1: enable; 0: disable	RW
1	RSZ_EN	Resize enable. 1: enable; 0: disable	RW
0	IPU_EN	IPU enable. 1: enable; 0: disable Once IPU is enabled, IPU keeps working until the flag OUT_END in the status register IPU_STATUS is set value 1.	RW

NOTES:

Setting value 1 to IPU_RST will reset all software visible IPU registers immediately. It is not recommended that stopping IPU abruptly by clearing IPU_EN when IPU is running (IPU_EN=1, OUT_END=0), or writing value 1 to IPU_RST when DMA (in or out) is working (IPU_EN=1, OUT_END=0), otherwise, the result is unpredictable.

19.5.2 IPU Status Register

IPU_STATUS **0x4**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																																OUT_END
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0

Bits	Name	Description	R/W
31:1	Reserved	Writing has no effect, read as zero.	R
0	OUT_END	Output DMA termination flag. 1: finished; 0: not finished HW can only set 1 to it, while SW can only clear it to 0	R/W

NOTES:

If IPU_CONTROL.FM_IRQ_EN has been set 1, once OUT_END is set value 1 which denotes a frame's post process done, an low level active interrupt request will be issued until corresponding software handler clear OUT_END to value 0.

19.5.3 Data Format Register

D_FMT **0x8**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															OUT_FMT															IN_FMT		
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:18	Reserved	Writing has no effect, read as zero.	R
17:16	OUT_FMT	Indicates the destination data format: 00: RGB555 01: RGB565 10: RGB888 11: reserved	RW
15:3	Reserved	Writing has no effect, read as zero.	R
2:0	IN_FMT	Indicates the source data format: 000: YUV 4:2:0 001: YUV 4:2:2 010: YUV 4:4:4 011: YUV 4:1:1 100: YCbCr 4:2:0 101: YCbCr 4:2:2 110: YCbCr 4:4:4 111: YCbCr 4:1:1	RW

19.5.4 Input Y Data Address Register

Y_ADDR **0xc**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Y_ADDR																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W

31:0	Y_ADDR	The source Y data buffer's start address	RW

19.5.5 Input U Data Address Register

U_ADDR **0x10**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

U_ADDR

RST ?

Bits	Name	Description	R/W
31:0	U_ADDR	The source U data buffer's start address	RW

19.5.6 Input V Data Address Register

V_ADDR **0x14**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

V_ADDR

RST ?

Bits	Name	Description	R/W
31:0	V_ADDR	The source V data buffer's start address	RW

19.5.7 Input Geometric Size Register

IN_FM_GS **0x18**

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

IN_FM_W IN_FM_H

RST ?

Bits	Name	Description	R/W
31:27	Reserved	Writing has no effect, read as zero.	R
26:16	IN_FM_W	The width of the input frame (unit: byte). Y's width equals this value while the width of U/V or Cb/Cr is calculated by IPU automatically in terms of the source data format.	RW
15:11	Reserved	Writing has no effect, read as zero.	R

10:0	IN_FM_H	The height of the input frame (unit: byte). Y's height equals this value while the height of U/V or Cb/Cr is calculated by IPU automatically in terms of the source data format.	RW

19.5.8 Input Y Data Line Stride Register

Y_STRIDE **0x1c**

<table>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="20"></td><td colspan="12">Y_S</td></tr>
<tr><td>RST</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td></tr>
</table>

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	Y_S	The row's stride of the source Y data in the external memory. (unit: byte)	RW

19.5.9 Input UV Data Line Stride Register

UV_STRIDE **0x20**

<table>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="4"></td><td colspan="12">U_S</td><td colspan="4"></td><td colspan="12">V_S</td></tr>
<tr><td>RST</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td><td>?</td></tr>
</table>

Bits	Name	Description	R/W
31:28	Reserved	Writing has no effect, read as zero.	R
27:16	U_S	The row's stride of the source U data in the external memory. (unit: byte)	RW
15:12	Reserved	Writing has no effect, read as zero.	R
11:0	V_S	The row's stride of the source V data in the external memory. (unit: byte)	RW

19.5.10 Output Frame Start Address Register

OUT_ ADDR **0x24**

<table>
<tr><td>Bit</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

	OUT_ADDR																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:0	OUT_ ADDR	The output buffer’s start address.	RW

19.5.11 Output Geometric Size Register

OUT_GS **0x28**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			OUT_FM_W																OUT_FM_H													
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:30	Reserved	Writing has no effect, read as zero.	R
29:16	OUT_FM_W	The width of the output destination frame (unit: byte).	RW
15:14	Reserved	Writing has no effect, read as zero.	R
13:0	OUT_FM_H	The height of the output destination frame (unit: byte).	RW

19.5.12 Output Data Line Stride Register

OUT_STRIDE **0x2c**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	OUT_S															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	OUT_S	The row’s stride of the destination data buffer in the external memory. (unit: byte)	RW

19.5.13 Resize Coefficients Table Index Register

RSZ_COEF_INDEX **0x30**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												HE_IDX																VE_IDX				
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:21	Reserved	Writing has no effect, read as zero.	R
20:16	HE_IDX	Indicates the last valid entry number of the horizontal resize LUT.	RW
15:5	Reserved	Writing has no effect, read as zero.	R
4:0	VE_IDX	Indicates the last valid entry number of the vertical resize LUT.	RW

19.5.14 CSC C0 Coefficient Register

CSC_C0_COEF **0x34**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C0_COEF											
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C0_COEF	The C0 coefficient of the YUV/YCbCr to RGB conversion. C0_COEF = [C0 * 1024 + 0.5]	RW

Note:

R = C0*(Y – X0) + C1*(Cr-128)

G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128)

B = C0*(Y – X0) + C4*(Cb-128)

19.5.15 CSC C1 Coefficient Register

CSC_C1_COEF **0x38**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C1_COEF											
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W

31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C1_COEF	The C1 coefficient of the YUV/YCbCr to RGB conversion. C1_COEF = [C1 * 1024 + 0.5]	RW
Note: R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128)			

19.5.16 CSC C2 Coefficient Register

CSC_C2_COEF **0x3c**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C2_COEF											
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C2_COEF	The C2 coefficient of the YUV/YCbCr to RGB conversion. C2_COEF = [C2 * 1024 + 0.5]	RW
Note: R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128)			

19.5.17 CSC C3 Coefficient Register

CSC_C3_COEF **0x40**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C3_COEF											
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C3_COEF	The C3 coefficient of the YUV/YCbCr to RGB conversion. C3_COEF = [C3 * 1024 + 0.5]	RW

Note:

R = C0*(Y – X0) + C1*(Cr-128)

G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128)

B = C0*(Y – X0) + C4*(Cb-128)

19.5.18 CSC C4 Coefficient Register

CSC_C4_COEF **0x44**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C4_COEF											
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C4_COEF	The C4 coefficient of the YUV/YCbCr to RGB conversion. C4_COEF = [C4 * 1024 + 0.5]	RW
Note: R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128)			

19.5.19 Horizontal Resize Coefficients Look Up Table Register group

HRSZ_COEF_LUT **0x48 ~ 0x94**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							W_COEF								IN_EN	OUT_EN
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:13	Reserved	Writing has no effect, read as zero.	R

9:2	W_COEF	Weighting coefficients, 8 bits length, that is to say the precision is 1/128.
		For up-scaling,
		W_k = 1 – (k*n/m – [k*n/m]), k = 0, 1, … m-1.
		For down-scaling,
		for (t=0, k=0; k < n; k++) {
		If ([(t*n+1)/m] – k >=1) { W_k = 0;}
		else if ((t*n+1)/m – k == 0) { W_k = 1; t++;}
		else { W_k = 1 – ((t*n+1)/m – [t*n/m]); t++;}
		}
		W_COEF $_k$ = [128 * W_k]
		Here n stands for original pixel points, m stands for pixel points after resize. For example down-scaling 5:3, n = 5, m = 3.
			RW

		Moreover, m and n are prime, that is, for example 8:2 should be converted to 4:1. When IPU_CONTROL.RSZ_EN set as 1 and m:n = 1:1, all coefficients should be calculated as up-scale case.	

1	IN_EN	Flag for whether new pixel would be used.
		IN_EN = 0, no new pixel
		IN_EN = 1, one new pixel
		In down scale case, IN_EN always equals 1.
		In up scale case,
		For (i=0, k=0; k < m; k++) {
		If(i<= k*n/m) { IN_EN $_k$ = 1; i++;}
		else { IN_EN $_k$ =0;}
		}
			RW

0	OUT_EN	Flag for whether current interpolation would be output.
		OUT_EN = 0, current interpolation would not be output
		OUT_EN = 1, current interpolation would be output
		In up scale case, OUT_EN always equals 1.
		In down scale case,
		For (t=0, k=0; k < n; k++) {
		If([(t*n+1)/m] – k >=1)
		OUT_EN $_k$ = 0;
		else {OUT_EN $_k$ =1; t++;}
		}
			RW

NOTES:

The coefficient number equals to max (m, n). HLUT (horizontal look up table) and VLUT (vertical look up table) are independent, so the two tables may have different coefficient number. Therefore,

RSZ_COEF_INDEX.VIDX = The coefficient number of VLUT – 1

RSZ_COEF_INDEX.HIDX = The coefficient number of HLUT – 1

Moreover, when m=1 for down-scaling, discard above formula and use following rules:

1. W_COEF_0 = 64 (W_0 = 0.5), and $W\_COEF_{1 \sim n-1}$ = 0
2. IN_EN always equals 1
3. OUT_EN_0 = 1, and $OUT\_EN_{1 \sim n-1}$ = 0

Following are two examples of setting LUT

Resize coefficients for 7:3

W	W_COEF	IN_EN	OUT_EN	Pixel1	Pixel2	OUT
2/3	85	1	1	P [0]	P [1]	P [0] * 2/3 + P [1] * 1/3
0	0	1	0	P [1]	P [2]	No new pixel out
1/3	42	1	1	P [2]	P [3]	P [2] * 1/3 + P [3] * 2/3
0	0	1	0	P [3]	P [4]	No new pixel out
0	0	1	0	P [4]	P [5]	No new pixel out
1	128	1	1	P [5]	P [6]	P [5] * 1 + P [6] * 0
0	0	1	0	P [6]	P [7]	No new pixel out

Resize coefficients for 3:5

W	W_COEF	IN_EN	OUT_EN	Pixel1	Pixel2	OUT
1	128	1	1	P [0]	P [1]	P [0] * 1 + P [1] * 0
2/5	51	0	1	P [0]	P [1]	P [0] * 2/5 + P [1] * 3/5
4/5	102	1	1	P [1]	P [2]	P [1] * 4/5 + P [2] * 1/5
1/5	25	0	1	P [1]	P [2]	P [1] * 1/5 + P [2] * 4/5
3/5	76	1	1	P [2]	P [3]	P [2] * 3/5 + P [3] * 2/5

19.5.20 Vertical Resize Coefficients Look Up Table Register group

VRSZ_COEF_LUT **0x98 ~ 0xe4**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																							W_COEF								IN_EN	OUT_EN
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Function descriptions are same as horizontal LUT.

19.6 Calculation for Resized width and height

For software, to preset correct value for register OUT_GS, please refer to following formula.

Set IW stand for original input frame width, IH stand for original input frame height, OW stand for new frame width after resize, OH stand for new frame height after resize.

In Up-scale case (n < m):

If [(IW - 1) * (m/n)] * (n/m) ==(IW-1) then
    OW = [(IW - 1) * (m/n)] + 1;
Else OW = [(IW - 1) * (m/n)] + 2;
If [(IH - 1) * (m/n)] * (n/m) == (IH-1) then
    OH = [(IH - 1) * (m/n)] + 1;
Else OH = [(IH - 1) * (m/n)] + 2;

In Down-scale case (n>m):

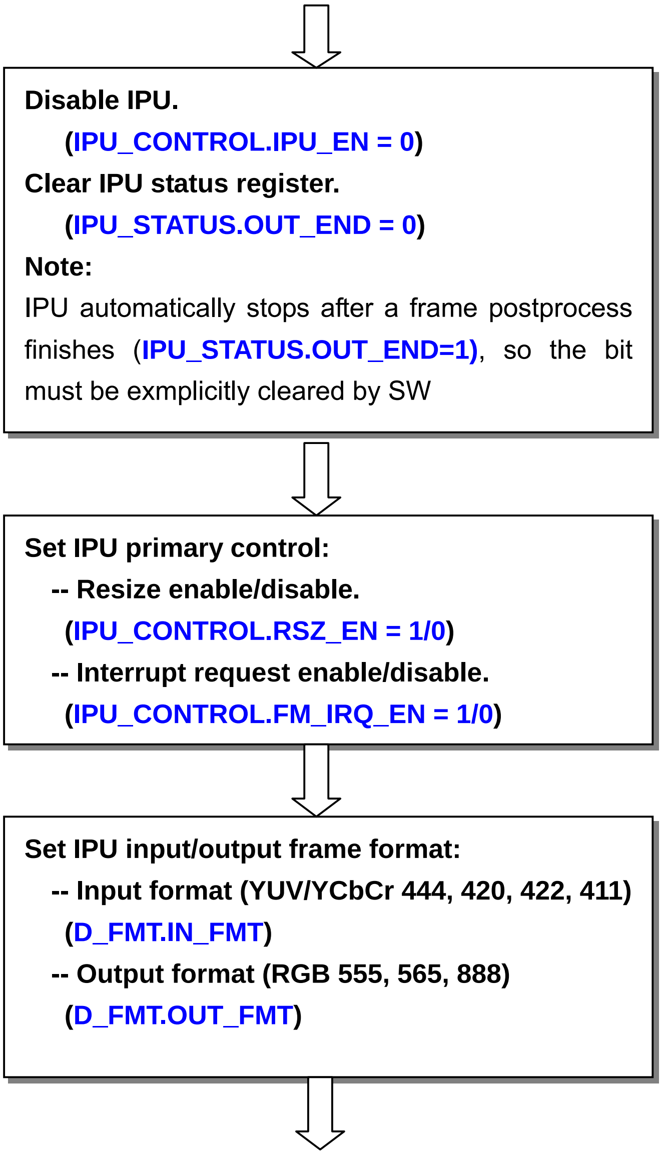
If [(IW - 1) * (m/n)] * (n/m) ==(IW-1) then
    OW = [(IW - 1) * (m/n)];
Else OW = [(IW - 1) * (m/n)] + 1;
If [(IH - 1) * (m/n)] * (n/m) ==(IH-1) then
    OH = [(IH - 1) * (m/n)];
Else OH = [(IH - 1) * (m/n)] + 1;

For example:

A 36x46 frame with the horizontal resize ratio of 4:5 (up-scale) and vertical resize ratio of 7:6 (down-scale), by the expressions above we get its new size after resize from the following process.

For Width: [(36 - 1) * (5/4)] * (4/5) = 34.4 ≠ (36-1)
    So OW = [(36 - 1) * (5/4)] + 2 = 45
For Height: [(46 - 1) * (6/7)] * (7/6) = 44.33 ≠ (46 - 1)
    So OH = [(46 - 1) * (6/7)] + 1 = 39

19.7 IPU Initialization Flow



Disable IPU.

(IPU_CONTROL.IPU_EN = 0)

Clear IPU status register.

(IPU_STATUS.OUT_END = 0)

Note:

IPU automatically stops after a frame postprocess finishes (**IPU_STATUS.OUT_END=1)**, so the bit must be exmplicitly cleared by SW

Set IPU primary control:

- **Resize enable/disable.** (IPU_CONTROL.RSZ_EN = 1/0)
- **Interrupt request enable/disable.** (IPU_CONTROL.FM_IRQ_EN = 1/0)

Set IPU input/output frame format:

- **Input format (YUV/YCbCr 444, 420, 422, 411)** (D_FMT.IN_FMT)
- **Output format (RGB 555, 565, 888)** (D_FMT.OUT_FMT)

Set input frame size:

- **-- Input frame width (Eg: 288x188 frame)** **(IN_FM_GS.IN_FM_W = 288)**
- **-- Input frame height (Eg: 288x188 frame)** **(IN_FM_GS.IN_FM_H = 188)**
- **-- Y frame stride (Y_STRIDE.Y_S)**
- **-- U frame stride (UV_STRIDE.U_S)**
- **-- V frame stride (UV_STRIDE.V_S)**

Note: Frame width/height value should be restricted according to frame format (ensure it is a legal size). In the case of 411 the value should be multiple of 4. In the case of 420/422 the value should be multiple of 2. In the case of 444 the value can be any integer in the legal range (33 ~ 2047). Moreover, the stride value should be set to ensure frames' every line start address are word aligned.

Set input/output data start address:

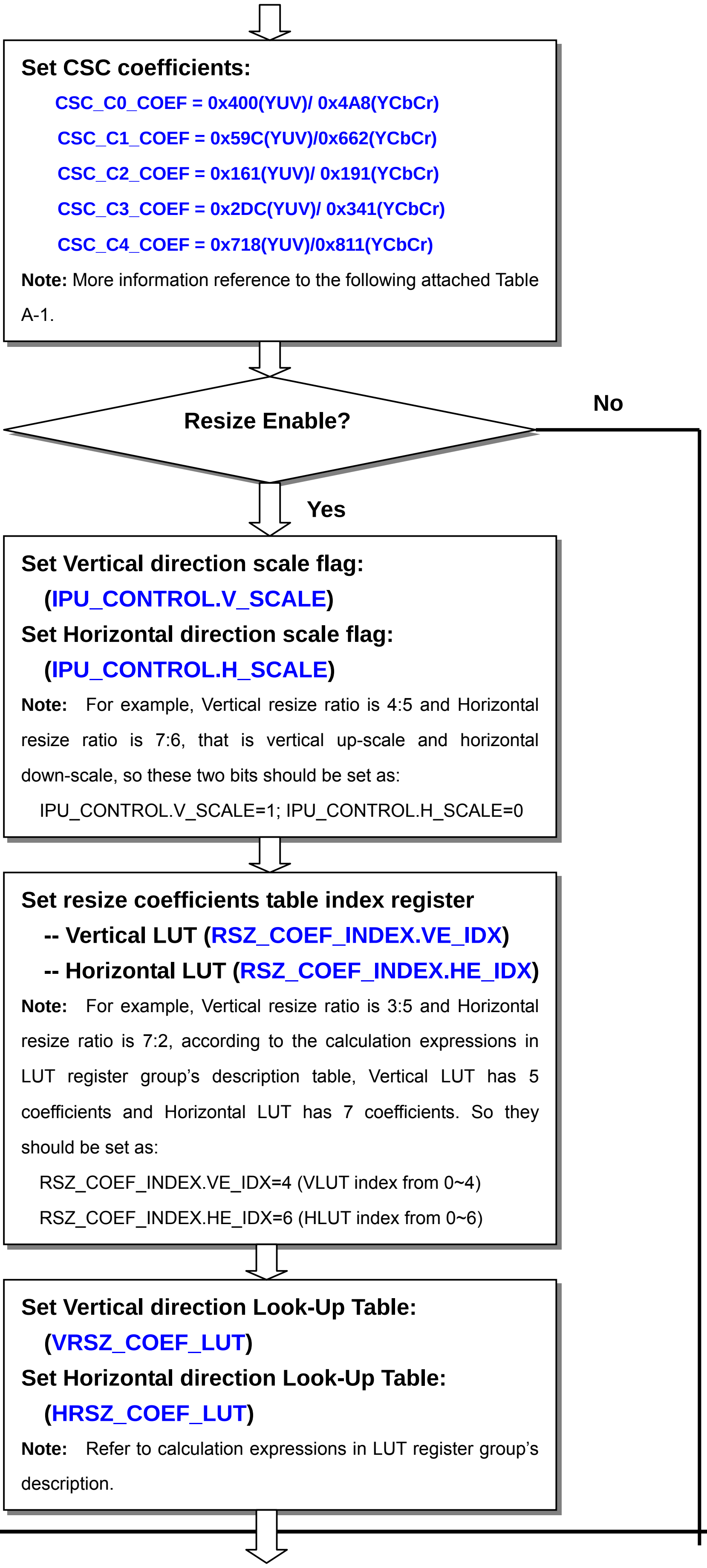
- **-- Y frame start address (Y_ADDR.Y_ADDR)**
- **-- U frame start address (U_ADDR.U_ADDR)**
- **-- V frame start address (V_ADDR.V_ADDR)**
- **-- Output frame start address** **(OUT_ADDR.OUT_ADDR)**

Note: Y/U/V frame start address must be word aligned. Output frame start address can be half-word or word aligned except RGB888 format, which should be word aligned.

Set output frame size:

- **-- Output frame width** **(OUT_GS.OUT_FM_W)**
- **-- Output frame height** **(OUT_GS.OUT_FM_H)**
- **-- Output frame stride** **(OUT_STRIDE.OUT_S)**

Note: Since the unit is byte, the exact values filled in are the pixel numbers left shifted by 1 or 2 according to output format. For example: RGB888, each pixel takes 4 bytes, so the width value is pixel numbers * 4. Hence, a RGB888 format output frame with size of 120x80 and stride of 124, their value should be filled as: OUT_GS.OUT_FM_W=120<<2;
OUT_GS.OUT_FM_H=80;OUT_STRIDE.OUT_S=124<<2



Set CSC coefficients:

CSC_C0_COEF = 0x400(YUV)/ 0x4A8(YCbCr)
CSC_C1_COEF = 0x59C(YUV)/0x662(YCbCr)
CSC_C2_COEF = 0x161(YUV)/ 0x191(YCbCr)
CSC_C3_COEF = 0x2DC(YUV)/ 0x341(YCbCr)
CSC_C4_COEF = 0x718(YUV)/0x811(YCbCr)

Note: More information reference to the following attached Table A-1.

Set Vertical direction scale flag:
(IPU_CONTROL.V_SCALE)
Set Horizontal direction scale flag:
(IPU_CONTROL.H_SCALE)

Note: For example, Vertical resize ratio is 4:5 and Horizontal resize ratio is 7:6, that is vertical up-scale and horizontal down-scale, so these two bits should be set as:

IPU_CONTROL.V_SCALE=1; IPU_CONTROL.H_SCALE=0

Set resize coefficients table index register
-- Vertical LUT (RSZ_COEF_INDEX.VE_IDX)
-- Horizontal LUT (RSZ_COEF_INDEX.HE_IDX)

Note: For example, Vertical resize ratio is 3:5 and Horizontal resize ratio is 7:2, according to the calculation expressions in LUT register group's description table, Vertical LUT has 5 coefficients and Horizontal LUT has 7 coefficients. So they should be set as:

RSZ_COEF_INDEX.VE_IDX=4 (VLUT index from 0~4)
RSZ_COEF_INDEX.HE_IDX=6 (HLUT index from 0~6)

Set Vertical direction Look-Up Table:
(VRSZ_COEF_LUT)
Set Horizontal direction Look-Up Table:
(HRSZ_COEF_LUT)

Note: Refer to calculation expressions in LUT register group's description.

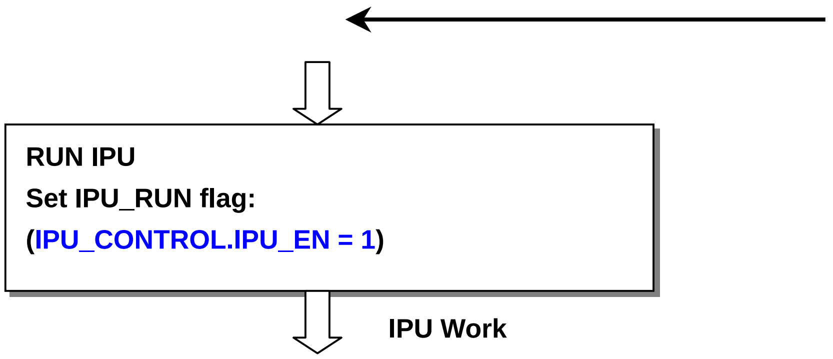
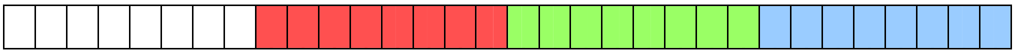
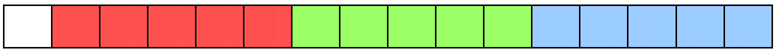
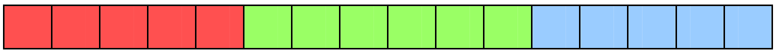


Table A-1. YUV/YCbCr to RGB CSC Equations

Input data	Matrix	CSC_COEF register values
YUV	R = C0*(Y – X0) + C1*(V-128) G = C0*(Y – X0) – C2*(U-128) – C3*(V-128) B = C0*(Y – X0) + C4*(U-128) X0: 0 C0: 1 C1: 1.4026 C2: 0.3444 C3: 0.7144 C4: 1.7730	CSC_C0_COEF = 0x400 CSC_C1_COEF= 0x59C CSC_C2_COEF = 0x161 CSC_C3_COEF = 0x2DC CSC_C4_COEF = 0x718
YCbCr	R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128) X0: 16 C0: 1.164 C1: 1.596 C2: 0.391 C3: 0.813 C4: 2.018	CSC_C0_COEF = 0x4A8 CSC_C1_COEF = 0x662 CSC_C2_COEF = 0x191 CSC_C3_COEF = 0x341 CSC_C4_COEF = 0x811

Table A-2. Output data package format

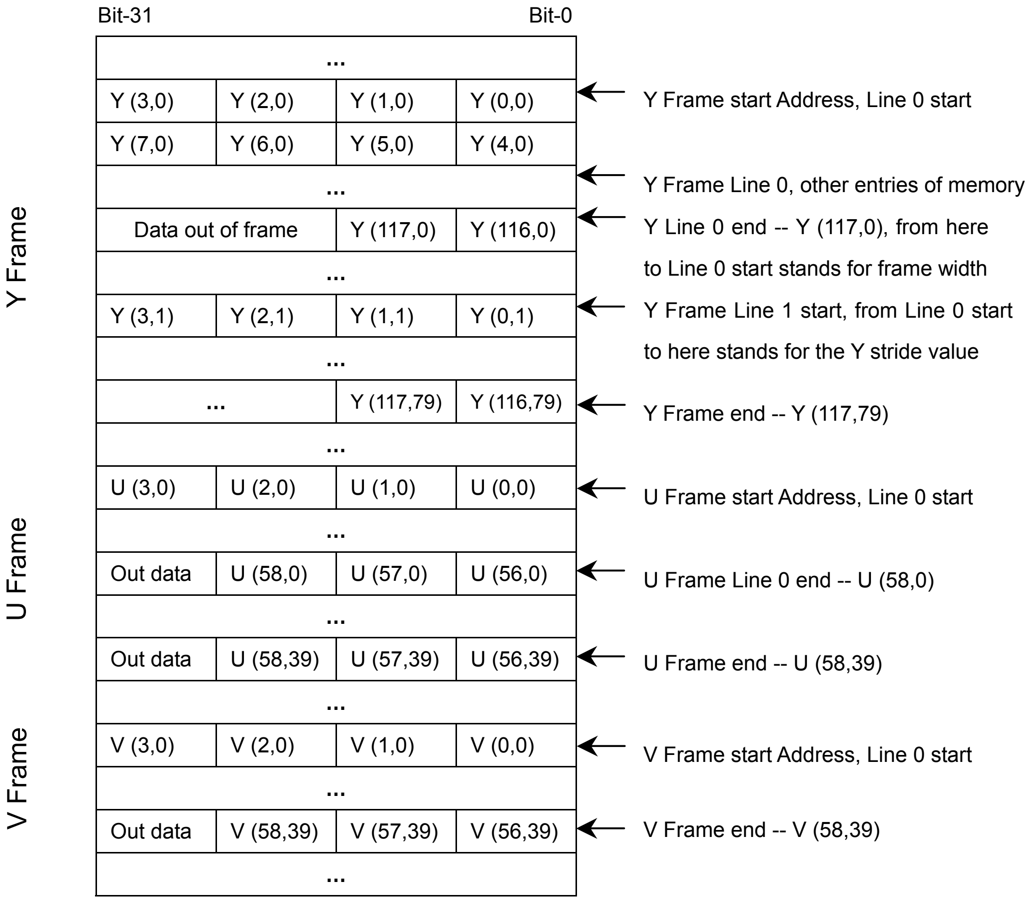
Format	Package
RGB888	Bit-31 24 23 16 15 8 7 0  EMPTY R7 R0 G7 G0 B7 B0
RGB555	15 14 10 9 5 4 0  Empty R7 R3 G7 G3 B7 B3
RGB565	15 11 10 5 4 0  R7 R3 G7 G2 B7 B3

Note:

All R/G/B data are little-endian type; all the empty bits in the above figure are filled with 0.

Figure A-1 Source Data storing format in external memory.

Example: YUV420 118x80 frame



	Bit-31			Bit-0	
	...				
Y Frame	Y (3,0)	Y (2,0)	Y (1,0)	Y (0,0)	Y Frame start Address, Line 0 start
	Y (7,0)	Y (6,0)	Y (5,0)	Y (4,0)	
	...				Y Frame Line 0, other entries of memory
	Data out of frame		Y (117,0)	Y (116,0)	Y Line 0 end -- Y (117,0), from here to Line 0 start stands for frame width
	...				
	Y (3,1)	Y (2,1)	Y (1,1)	Y (0,1)	Y Frame Line 1 start, from Line 0 start to here stands for the Y stride value
	...				
	...		Y (117,79)	Y (116,79)	Y Frame end -- Y (117,79)
	...				
U Frame	U (3,0)	U (2,0)	U (1,0)	U (0,0)	U Frame start Address, Line 0 start
	...				
	Out data	U (58,0)	U (57,0)	U (56,0)	U Frame Line 0 end -- U (58,0)
	...				
	Out data	U (58,39)	U (57,39)	U (56,39)	U Frame end -- U (58,39)
	...				
V Frame	V (3,0)	V (2,0)	V (1,0)	V (0,0)	V Frame start Address, Line 0 start
	...				
	Out data	V (58,39)	V (57,39)	V (56,39)	V Frame end -- V (58,39)
	...				

Note:
1. Every line's start address should be word aligned.
2. All pixel data should be stored as little-endian format.
3. Destination data (RGB) storing format in external memory is similar with above figure, but RGB555 and RGB565 frame's every line start address can be half-word aligned (RGB888 frame still need word aligned).

20 Jz4725 Boot ROM Specification

The Jz4725 contains an internal 8KB boot ROM. The CPU boots from the boot ROM after reset.

20.1 Boot Select

The boot sequence of the Jz4725 is controlled by boot_sel[1:0]. The configuration of boot_sel[1:0] is showed as below:

Table 20-1 Boot Configuration of Jz4725

boot_sel[1:0]	
00	Not used
01	Boot from USB device
10	Boot from 512 page size NAND flash at CS1
11	Boot from 2048 page size NAND flash at CS1

20.2 Boot Sequence

After reset, the boot program on the internal boot ROM executes as bellows:

1. Read boot_sel[1:0] and branch to proper programs according to it.
2. If it is boot from USB (“01”), connect USB cable to PC host, receive a block of data from USB and store it in internal SRAM. Then branch to this area in SRAM.
3. If it is boot from NAND flash (“10”, or “11”), set proper NAND flash page size, read first byte from the NAND to know if it is 8 or 16 bits width, page cycle is 2 or 3 cycles. Set EMC according to it and read up to 8KB from NAND to internal SRAM. Then branch to SRAM at 4 bytes offset.

Note: The Jz4725 internal SRAM is 16KB, its address is from 0x80000000 to 0x80004000.

20.3 NAND Boot Specification

If CPU boots from NAND flash (CS1), the boot program will read the first byte from NAND flash to know if it is 8 or 16 bits width, page cycle is 2 or 3 cycles. If bit[7:4] of the first byte is zero, it is a 16-bit flash, else it is a 8-bit flash. If bit[3:0] of the first byte is zero, it is 2 page cycles, else it is 3 page cycles. The boot ROM will configure EMC and GPIO pins according to the first byte data, then continue to load up to 8KB from NAND to internal SRAM and branch to internal SRAM at 4 bytes offset.

The boot program can load two areas of data from NAND flash to internal SRAM, one is the normal area up to 8KB starting from NAND flash address 0, the other is the backup area up to 8KB starting from NAND flash address 0x2000. After reset, the boot program will first read the normal area data from NAND flash using hardware Reed-Solomon ECC. If no ECC error is detected or ECC error is correctable, the boot program then branches to internal SRAM at 4 bytes offset. If it detects an uncorrectable ECC error, it will continue to read the backup area of data from NAND flash using hardware Reed-Solomon ECC. If no ECC error is detected or ECC error is correctable, the boot program will then branch to internal SRAM at 4 bytes offset. If it detects an uncorrectable ECC error again, it will continue to start booting from NOR flash at CS4.

Every time the boot program starts reading a page, it first checks if the page contains valid data. If the page contains valid data, it will read the page to internal SRAM. If the page data is not valid, the boot program knows that it has finished reading all the data and then branches to internal SRAM at 4 bytes offset. The boot program will decide the page valid flag according to the 3rd, 4th, 5th byte of the spare area of each page. If one of the three bytes is zero, the boot program knows that this page data is valid, else this page data is not valid.

The boot program enables hardware RS ECC when reading NAND flash data. When a 512-byte data is read, it will check the calculated ECC with stored ECC. The calculated and stored ECC are both 9 bytes per 512-byte data. And the 9-byte stored ECC is starting from the 7th byte of the spare area of each page.

The NAND boot specification is showed as below:

Table 20-2 NAND First Data Byte Definition

First Byte of NAND	Data Bus Width of Flash
0xff	3 page cycles, 8-bit
0xf0	2 page cycles, 8-bit
0x0f	3 page cycles, 16-bit
0x00	2 page cycles, 16-bit

Table 20-3 NAND Spare Area Definition

Spare Area Offset	Description
0 – 1	Reserved
2 – 4	Page is valid if one of the three bytes is zero
5	Reserved
6 - 14	Stored 9-byte ECC of data0 – data511
15 - 23	Stored 9-byte ECC of data512 – data1023 (2KB page NAND only)
24 – 32	Stored 9-byte ECC of data1024 – data1535 (2KB page NAND only)

33 - 41	Stored 9-byte ECC of data1536 – data2047 (2KB page NAND only)

The procedure of the Jz4725 NAND boot is showed as below:

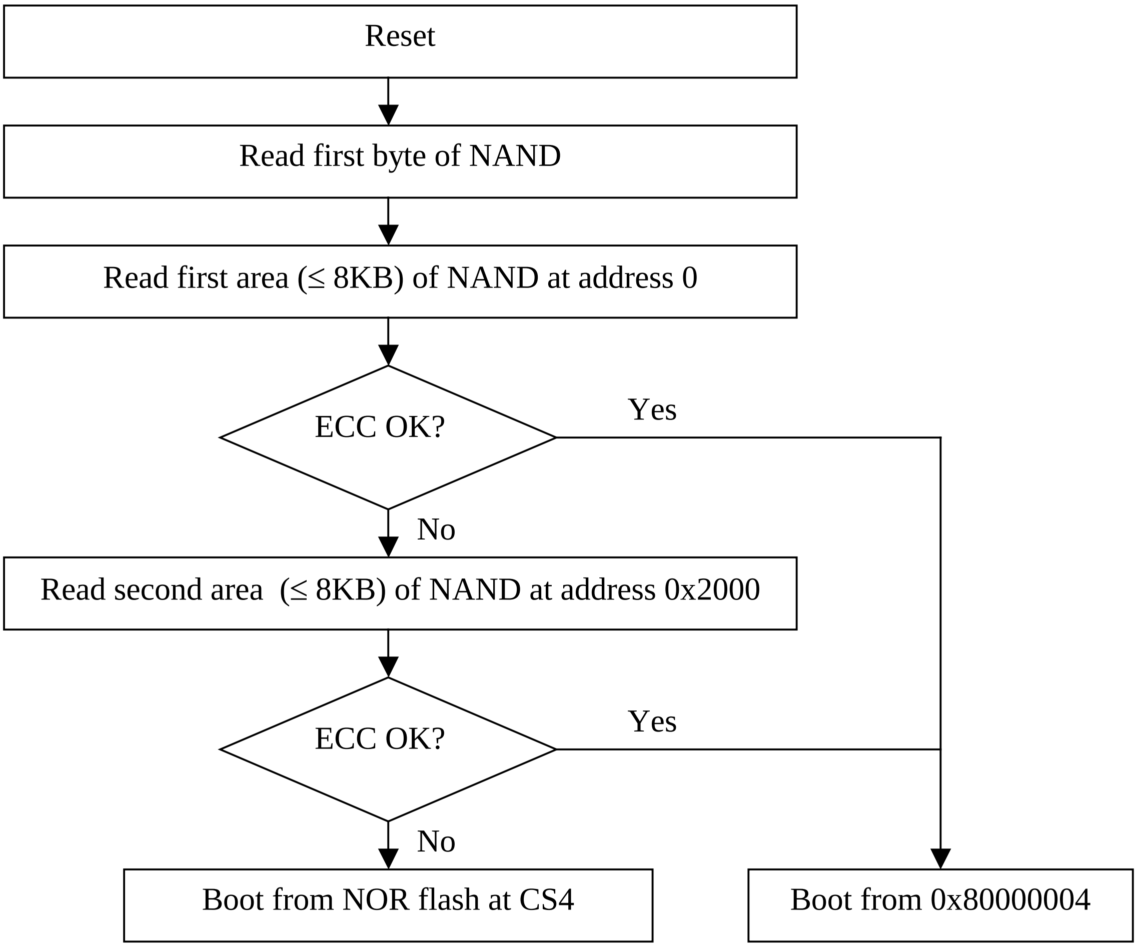

Reset
Read first byte of NAND
Read first area (≤ 8KB) of NAND at address 0
ECC OK?
Yes
No
Read second area (≤ 8KB) of NAND at address 0x2000
ECC OK?
Yes
No
Boot from NOR flash at CS4
Boot from 0x80000004

Table 20-4 Jz4725 NAND Boot Sequence

20.4 USB Boot Specification

When boot_sel[1:0] is selected from USB boot, the internal boot ROM downloads user program from the USB port to internal SRAM and branches to the internal SRAM to execute the program.

The boot program supports both high-speed (480MHz) and full-speed (12MHz) transfer modes. The boot program uses the following two transfer types.

Table 20-5 Transfer Types Used by the Boot Program

Transfer Type	Description
Control Transfer	Used for transmitting standard requests and vendor requests.
Bulk Transfer	Used for responding to vendor requests and transmitting a user program.

The following figure shows an overview of the USB communication flow.

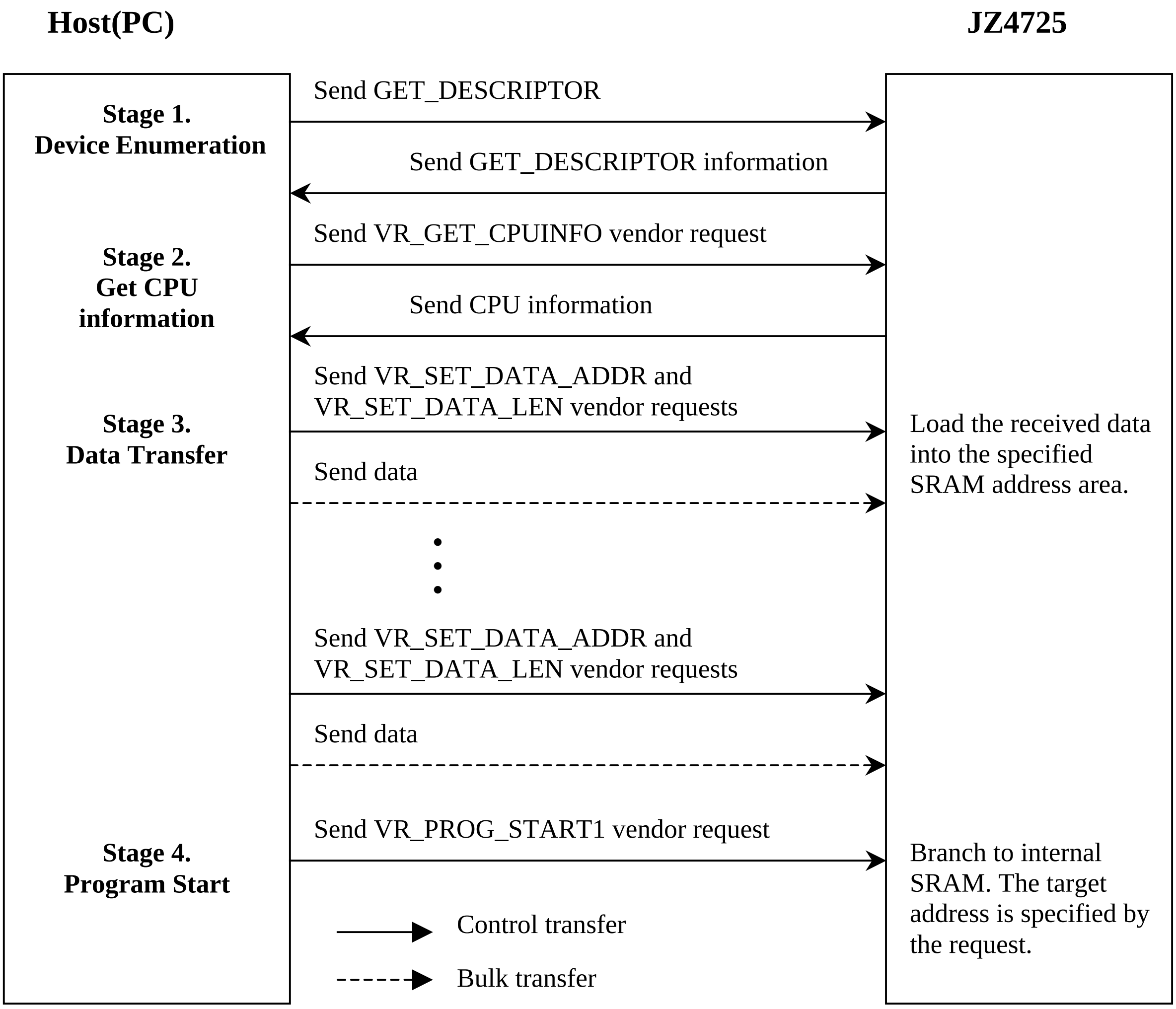

Host(PC)
JZ4725
Stage 1.
Device Enumeration
Stage 2.
Get CPU information
Stage 3.
Data Transfer
Stage 4.
Program Start
Send GET_DESCRIPTOR
Send GET_DESCRIPTOR information
Send VR_GET_CPUINFO vendor request
Send CPU information
Send VR_SET_DATA_ADDR and VR_SET_DATA_LEN vendor requests
Send data
Send VR_SET_DATA_ADDR and VR_SET_DATA_LEN vendor requests
Send data
Send VR_PROG_START1 vendor request
Load the received data into the specified SRAM address area.
Branch to internal SRAM. The target address is specified by the request.
Control transfer
Bulk transfer

Figure 20-1 USB Communication Flow

The vendor ID and product ID for the USB boot device are 0x601A and 0x4740 respectively. The Configuration for USB is for Control Endpoint 0 with Max Packet Size equals 64 bytes, Bulk IN at Endpoint 1 with Max Packet Size equals 512 bytes in high-speed and 64 bytes in full-speed, Bulk OUT at Endpoint 1 with Max Packet Size equals 512 bytes in high-speed and 64 bytes in full-speed.

The USB boot program provides six vendor requests through control endpoint for user to download/upload data to/from device, and to branch to a target address to execute user program. The six vendor requests are VR_GET_CPU_INFO (0x00), VR_SET_DATA_ADDRESS (0x01), VR_SET_DATA_LENGTH (0x02), VR_FLUSH_CACHES (0x03), VR_PROGRAM_START1 (0x04) and VR_PROGRAM_START2 (0x05). User program is transferred through Bulk IN or Bulk OUT endpoint.

When Jz4725 is reset with boot_sel[1:0] equals 01b, the internal boot ROM will switch to USB boot mode and wait for USB requests from host. After connecting the USB device port to host, host will recognise the connection of a USB device, and start device enumeration. After finishing the device enumeration, user can send VR_GET_CPU_INFO (0x00) to query the device CPU information. If user wants to download/upload a program to/from device, two vendor requests VR_SET_DATA_ADDRESS (0x01) and VR_SET_DATA_LENGTH (0x02) should be sent first to tell the device the address and length in byte of the subsequent transferring data. Then data can be transferred through bulk-out/bulk-in endpoint. After this first stage program has been transferred to device, user can send vendor request VR_PROGRAM_START1 (0x04) to let the CPU to execute the program. This first stage program must not greater than 16KB and is normally used to init GPIO and SDRAM of the target board. At the end of the first stage program, it can return back to the internal boot ROM by jumping to ra ($31) register. Thus user can download a new program to the SDRAM of the target board like the first stage, and send vendor request VR_FLUSH_CACHES (0x03) and VR_PROGRAM_START2 (0x05) to let the CPU to execute the new program. Next figure is the typical procedure of USB boot.

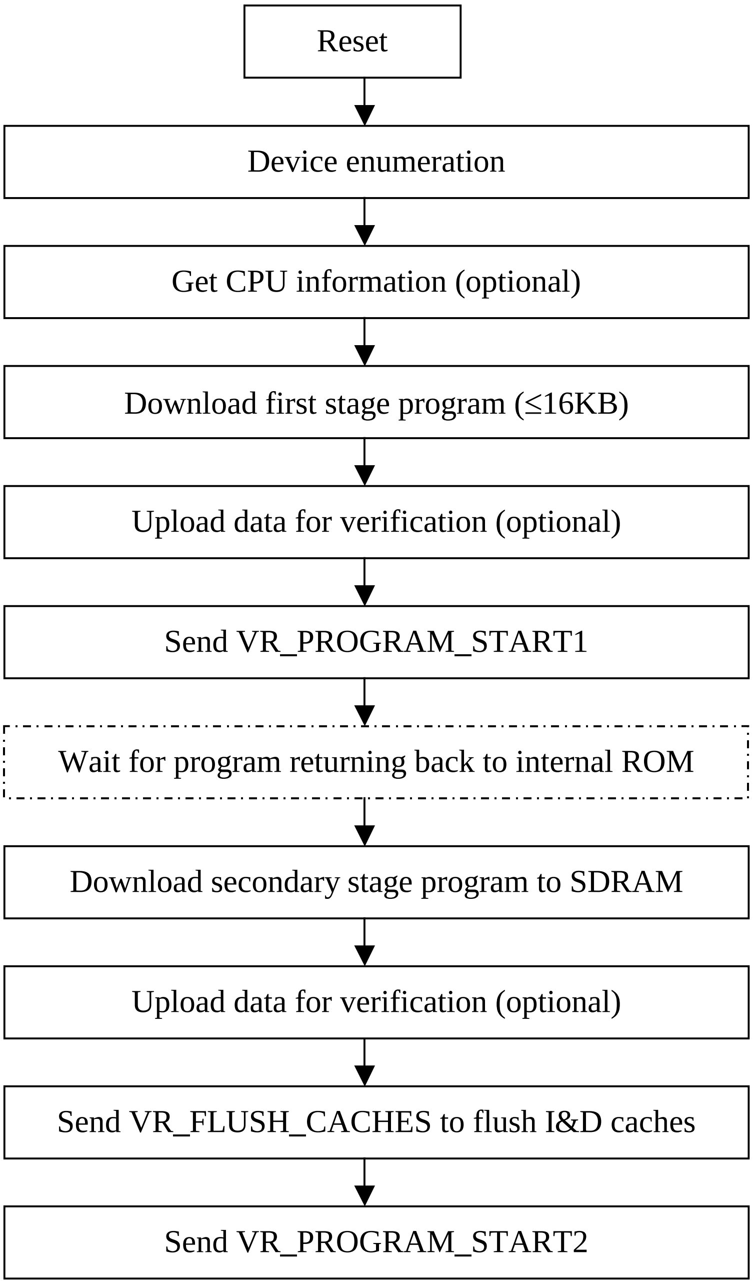

Reset
Device enumeration
Get CPU information (optional)
Download first stage program (≤16KB)
Upload data for verification (optional)
Send VR_PROGRAM_START1
Wait for program returning back to internal ROM
Download secondary stage program to SDRAM
Upload data for verification (optional)
Send VR_FLUSH_CACHES to flush I&D caches
Send VR_PROGRAM_START2

Figure 20-2 Typical Procedure of USB Boot

Following tables list all the vendor requests that USB boot program supports:

Table 20-6 Vendor Request 0 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	00H	VR_GET_CPU_INFO: get CPU information
2	wValue	2	0000H	Not in used
4	wIndex	2	0000H	Not in used
6	wLength	2	0008H	8 bytes

Table 20-7 Vendor Request 1 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	01H	VR_SET_DATA_ADDRESS: set address for next bulk-in/bulk-out transfer
2	wValue	2	xxxxH	MSB (bit[31:16]) of the data address
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the data address
6	wLength	2	0000H	Not in used

Table 20-8 Vendor Request 2 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	02H	VR_SET_DATA_LENGTH: set length in byte for next bulk-in/bulk-out transfer
2	wValue	2	xxxxH	MSB (bit[31:16]) of the data length
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the data length
6	wLength	2	0000H	Not in used

Table 20-9 Vendor Request 3 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device

1	bRequest	1	03H	VR_FLUSH_CACHES: flush I-Cache and D-Cache
2	wValue	2	0000H	Not in used
4	wIndex	2	0000H	Not in used
6	wLength	2	0000H	Not in used

Table 20-10 Vendor Request 4 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	04H	VR_PROGRAM_START1: transfer data from D-Cache to I-Cache and branch to address in I-Cache. Note: After downloading program from host to device for the first time, you can only use this request to start the program. Since the USB boot program will download data to D-Cache after reset. This request will transfer data from D-Cache to I-Cache and execute the program in I-Cache.
2	wValue	2	xxxxH	MSB (bit[31:16]) of the program entry point
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the program entry point
6	wLength	2	0000H	Not in used

Table 20-11 Vendor Request 5 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device D6-D5 2: Vendor D4-D0 0: Device
1	bRequest	1	05H	VR_PROGRAM_START2: branch to target address directly
2	wValue	2	xxxxH	MSB (bit[31:16]) of the program entry point
4	WIndex	2	xxxxH	LSB (bit[15:0]) of the program entry point
6	WLength	2	0000H	Not in used