SPACEMIT

Make RISC-V development for intelligent future easier

Key Stone® K1 User Manual

V6.3 - 2026.08.31



Revision history

Version	Date	Notes
V6.3	2026.08.31	Removed Secure Boot feature in Section 1.2
V6.2	2026.07.16	Updated Preface Added RX and TX descriptor details to Chapter 15, including descriptor layouts and bit-field definitions
V6.1	2025.08.06	1. Updated clock tree structure diagram 2. Updated register “SoC I2S CLOCK GENERATION CONTROL REGISTER (ISCCR0/1)”
V6.0	2025.08.04	1. Updated clock tree structure diagram 2. Added new register "FREQUENCY CHANGE CONTROL REGISTER (FCCR)" 3. Updated register "SSPAX CLOCK RESET CONTROL REGISTER (APBC_SSPAX_CLK_RST)"
V5.2	2025.07.11	Made some color adjustments to the overall layout to improve readability
V5.1	2025.06.25	Fixed some minor oversights — typos, grammar errors
V5.0	2025.05.20	1. Updated K1 architecture block diagram: - DDR clock rate change - RCPU modules change 2. Updated K1 general features: - RCPU modules related
V4.0	2025.05.16	1. Updated pinout 2. Fixed typos, missing/wrong words & sentences, cross-references in several sections 3. Edited PDF from online version
V3.0	2025.04.16	Removed the Audio Subsystem then performed related updates
V2.0	2025.03.27	Restructured and improved the quality of all content of the whole document
V1.4	2025.03.18	1. Updated SSPA function clock source data 2. Updated clock tree structure
V1.3	2025.02.25	1. Reviewed and updated the whole document for grammar, clarity and consistency 2. Updated security algorithm, in particular removed SM2, SM3, SM4
V1.2	2024.07.24	Added V2D content
V1.1	2024.06.21	Updated data
V1.0	2024.03.15	Initial release

Preface

Proprietary, Confidentiality & Disclaimer

© 2026 SPACEMIT (Hangzhou) Technology Co., Ltd. All rights reserved.

This document is confidential information of SpacemiT, provided "AS IS" without any warranties. For details, please refer to the Legal Notice.

SPACEMIT (Hangzhou) Technology Co., Ltd

Address: Room 701, Building B, Future Sky Center, Wuchang Street, Yuhang District, Hangzhou City, Zhejiang Province, China

Website: https://www.spacemit.com/

Email: business@spacemit.com

Telephone: (+86) 0571-89000775

About Document

I Purpose

This document provides a comprehensive guide to **SpacemiT Key Stone® K1**, detailing its characteristics, features, logic structure, architecture and system modules. It includes in-depth descriptions of the functions and operation of the key subsystems of K1, such as CPU, memory, video, audio and interface systems.

This document also provides information about package, pinout, boot modes, power management, address mapping, interrupt assignments and performance specifications. Additionally, it provides in detail all functions, working methods and related register definitions of each module of K1, providing the interface timing relationship and related parameters to support an efficient development and integration with K1.

II Target audience

- Product Managers/Designers
- System Engineers
- Hardware Engineers
- Software Developers
- Individuals seeking to understand the specifications and hardware features of SpacemiT Key Stone® K1 for efficient application and product development

III Content summary

- Chapter 1 - Overview
- Chapter 2 - Package
- Chapter 3 - Pinout
- Chapter 4 - Electrical Characteristic
- Chapter 5 - Boot Modes
- Chapter 6 - Address Mapping
- Chapter 7 - Interrupt Assignments
- Chapter 8 - CPU System
- Chapter 9 - Top System
- Chapter 10 - Memory & Storage
- Chapter 11 - Video & Graphics
- Chapter 12 - Display Subsystem
- Chapter 13 - Video Capture Subsystem
- Chapter 14 - RCPU Subsystem
- Chapter 15 - High-Speed Interface System
- Chapter 16 - Low-Speed Interface System

Contents index

Revision history I

Preface III

About Document IV

I Purpose IV
II Target audience IV
III Content summary IV

Contents index V

Chapter 1 *Overview* 11

1.1 Introduction 2
1.2 General Features 2
1.3 Multimedia Features 4
1.4 Block Diagram 6

Chapter 2 *Package* 7

2.1 Introduction 8
2.2 FCCSP Type 8
2.3 FCBGA Type 10

Chapter 3 *Pinout* 12

3.1 Introduction 13
3.2 Pinout Diagram & Description 13
3.2.1 (A~N, 1~13) 14
3.2.2 (A~N, 14~26) 22
3.2.3 (P~AF, 1~13) 30
3.2.4 (P~AF, 14~26) 36
3.3 I/O Pin Parameters 42
3.3.1 For 1.8V I/O Pins 42
3.3.2 For 3.3V I/O Pins 43
3.4 Multiplexed Signal/Pin Functions 43
3.4.1 JTAG 44
3.4.2 Keypad Controller 45
3.4.3 Miscellaneous 45
3.4.4 SPIx 46
3.4.5 TWSI 46
3.4.6 UARTx 46
3.4.7 USB 47
3.5 Multi-Function I/O Pin Assignments 47
3.6 Power Supply Pins 55
3.7 Multi-Function Pin Register (MFPRs) 57
3.7.1 MFPR Functional Description 61

Chapter 4 *Electrical Characteristics* 66

4.1 Pin AC/DC Operating Conditions 67
4.2 Absolute Max Ratings 70
4.2.1 For Pins 70
4.2.2 For Packages 71

4.3 Pin Max Currents 71
4.4 Power On/Off Sequence 73
4.4.1 Power On Sequence 73
4.4.2 Power Off Sequence 75
4.5 Power Consumption 76
4.5.1 In Typical Application Scenario 76
4.5.2 In Particular Application Scenario 76

Chapter 5 *Boot Modes* **77**

5.1 Introduction 78
5.2 Boot Strapping Pins 78
5.2.1 Boot Mode Selection 78
5.2.2 Download Mode Selection 78
5.2.3 Boot Download Mode Selection 79
5.2.4 SPI NAND/NOR Flash Boot Voltage Selection 79

Chapter 6 *Address Mapping* **80**

6.1 Introduction 81
6.2 Main CPU Domain Address Mapping 81
6.3 Real-Time CPU Domain Address Mapping 85

Chapter 7 *Interrupt Assignments* **87**

7.1 Introduction 88
7.2 Main CPU Interrupts 88
7.3 Sensor-Hub Subsystem Interrupts 92

Chapter 8 *CPU System* **95**

8.1 Overview 96
8.2 CPU Subsystem 96
8.2.1 Introduction 96
8.2.2 Features 96
8.2.3 Functional Description 96
8.2.4 Generic Counter Register Description 96
8.2.5 Dragon CIU Registers Description 98
8.3 Real-Time CPU 110
8.3.1 Introduction 110
8.3.2 Features 110
8.3.3 Functional Description 111
8.3.4 Register Description 119

Chapter 9 *Top System* **131**

9.1 Overview 132
9.2 Clock & Reset 132
9.2.1 Introduction 132
9.2.2 Features 132
9.2.3 Functional Description 132
9.2.4 Register Description 135
9.3 JTAG 219
9.3.1 Introduction 219
9.3.2 Features 219
9.3.3 Functional Description 220
9.4 DMA 220
9.4.1 Introduction 220

9.4.2 Features 221
9.4.3 Functional Description 221
9.4.4 Register Description 236
9.5 Crypto 251
9.5.1 Introduction 251
9.5.2 Features 251
9.5.3 Functional Description 251
9.5.4 Register Description 251
9.6 Timer & Watchdog 257
9.6.1 Introduction 257
9.6.2 Features 257
9.6.3 Functional Description 258
9.6.4 Register Description 260
9.7 RTC 269
9.7.1 Introduction 269
9.7.2 Features 269
9.7.3 Functional Description 269
9.7.4 Register Description 269
9.8 MailBox 271
9.8.1 Introduction 271
9.8.2 Features 272
9.8.3 Functional Description 272
9.8.4 Register Description 272
9.9 Power Management & Lower Power Mode Control 275
9.9.1 Introduction 275
9.9.2 Features 275
9.9.3 Functional Description 275
9.9.4 Register Description 277

Chapter 10 *Memory & Storage* **304**

10.1 Overview 305
10.2 On-Chip Memory 305
10.2.1 Introduction 305
10.3 DDR 305
10.3.1 Introduction 305
10.3.2 Features 306
10.3.3 Functional Description 306
10.4 QSPI Flash 309
10.4.1 Introduction 309
10.4.2 Features 309
10.4.3 Functional Description 310
10.5 SD/eMMC 311
10.5.1 Introduction 311
10.5.2 Features 311
10.5.3 Functional Description 312
10.5.4 Register Description 316

Chapter 11 *Video & Graphics* **379**

11.1 Video Subsystem 380
11.1.1 Video Processing Unit (VPU) 380
11.2 Image Subsystem 389
11.2.1 Graphics Processing Unit (GPU) 389

11.2.2 V2D 391

Chapter 12 *Display Subsystem* **406**

12.1 Display Controller 407
12.1.1 Introduction 407
12.1.2 Features 407
12.1.3 Block Diagram 408
12.2 HDMI Interface 409
12.2.1 Features 409
12.2.2 Block Diagram 409
12.3 MIPI-DSI 410
12.3.1 Introduction 410
12.3.2 Features 410
12.3.3 Functional Description 410
12.3.4 Register Description 411
12.3.5 SPI LCD Display Interface 452
12.3.6 Register Description 459

Chapter 13 *Video Capture Subsystem* **518**

13.1 Overview 519
13.2 MIPI Camera IN Interface 519
13.2.1 Introduction 519
13.2.2 Features 520
13.2.3 Register Description 520
13.3 ISP 520
13.3.1 Introduction 520
13.3.2 Features 520
13.3.3 Register Description 521

Chapter 14 *RCPU Subsystem* **522**

14.1 Overview 523
14.2 Features 523
14.2.1 General 523
14.2.2 Main Peripherals 523
14.2.3 Power & Clock Management 523
14.3 Register Description 524
14.3.1 AUD_PMU Registers 524
14.3.2 AUD_MCUSYSCTRL Registers 529
14.3.3 AUD_AUDCLOCK Registers 533
14.3.4 AUD_AHBDMA Registers 536
14.3.5 PWM Registers 542

Chapter 15 *High-Speed Interface System* **545**

15.1 USB 546
15.1.1 Introduction 546
15.1.2 Features 546
15.1.3 Block Diagram 548
15.1.4 Register Description 549
15.2 PCIe 557
15.2.1 Introduction 557
15.2.2 Features 557
15.2.3 Functional Description 558
15.3 EMAC 563

15.3.1 Introduction 563
15.3.2 Features 563
15.3.3 Block Diagram 564
15.3.4 Programming Model 565
15.3.5 Register Description 566
15.3.6 Descriptor Details 642

Chapter 16 *Low-Speed Interface System* **646**

16.1 I2C Bus Interface 647
16.1.1 Introduction 647
16.1.2 Features 647
16.1.3 Functional Description 647
16.1.4 Register Description 678
16.2 SPI/I2S 692
16.2.1 Introduction 692
16.2.2 Features 692
16.2.3 Functional Description 693
16.2.4 Register Description 704
16.3 UART 715
16.3.1 Introduction 715
16.3.2 Features 715
16.3.3 Functional Description 716
16.3.4 Register Description 725
16.4 GPIO 743
16.4.1 Introduction 743
16.4.2 Functional Description 743
16.4.3 Register Description 745
16.5 One-Wire Bus Master Interface 749
16.5.1 Introduction 749
16.5.2 Functional Description 750
16.5.3 Register Description 755
16.6 IR-RX 760
16.6.1 Overview 760
16.6.2 Features 760
16.6.3 Functional Description 760
16.6.4 Register Description 761
16.7 PWM 764
16.7.1 Introduction 764
16.7.2 Features 764
16.7.3 Register Description 764

Chapter 1

Overview



Key Stone® K1 User Manual

1.1 Introduction

SpacemiT Key Stone® K1 is a high-performance and ultra-low-power SoC that integrates 8 RISC-V CPU cores with SpacemiT® Daoyi™ AI computing power. It comes with the following most relevant advantages:

- Integration with SpacemiT® self-innovated X60™ RISC-V core processor which adheres to the RISC-V 64GCVB architecture and RVA22 standard
- Capable of delivering 2.0 TOPS AI computing power by leveraging customized RISC-V instructions to enable CPU AI fusion computing
- Support for the popular AI inference frameworks such as TensorFlow Lite, TensorFlow, and ONNX Runtime
- Attainment of ultra-low power consumption through the incorporation of multiple granular power islands and dynamic power state adjustments, making K1 highly competitive in energy efficiency
- Availability of full-feature interfaces for enabling innovative applications and products
- Compatibility with mainstream OS to meet the needs of various application scenarios
- Compliance with the industrial-grade reliability standards

1.2 General Features

- **Application Processor (AP)**
  - SpacemiT® X60™ RISC-V Dual-Cluster 8-Core Processor
  - Adherence to the RISC-V 64GCVB architecture and RVA22 standard
  - Cluster 0
    - Quad-Core with 2.0 TOPS AI computing power
    - 32K L1-Cache per core
    - 512K L2-Cache
    - 512KB TCM
    - 256bit vector
  - Cluster 1
    - Quad-Core
    - 32K L1-Cache per core
    - 512K L2-Cache
    - 256bit vector
  - DVFS with adaptive operating voltage from 0.6V to 1.05V

- **DDR Memory**
  - Dual-Chip selection, 32-bit LPDDR4/LPDDR4x SDRAM with 2666 Mbps transfer rate, supporting up to 16 GB of RAM
  - Dual-Chip selection, 32-bit LPDDR3 SDRAM with 1866 Mbps transfer rate, supporting up to 4 GB of RAM

- **RCPU (Real-Time CPU)**
  - SRAM 256KB x1
  - R_CAN-FD x1
  - R_I2C x1
  - R_SPI x2
  - HDMI Audio
  - R_Debug
  - R_UART x2
  - R_PWM x10
  - DMA x1
  - R_IR_RX x1

- **Peripheral Controller**
  - GPIO (×128)
    - 128 pins
    - Pull-up/pull-down programmable
    - 104x 1.8V IO8
    - 24x 1.8V/3.3V IO
  - UART (×10)
    - AP/BT/print
  - I2C (×10)
    - For camera, G-Sensor, E-COMPASS, Proximit-Sensor, Light-Sensor, Gyro, Fingerprint, NFC, PMIC, Touch, etc.
    - 8x AP_I2C (AP I2C0/1/7 dedicated for camera) + 1x HDMI I2C + 1x PWR I2C
  - SPI (×4)
    - Support for both master and slave mode
    - For IMU, codec etc.
    - Platform with 4 SPI (1x QSPI, 1x SPI LCD, 2x SPI)
  - USB (×3)
    - USB 2.0 OTG
    - USB 2.0 Host
    - USB 3.0 (combo PCIE PortA)
  - PCIE (×3)
    - PCIE PortA Gen2x1
    - PCIE PortB Gen2x2
    - PCIE PortC Gen2x2
  - GMAC (×2)
    - 10/100/1000 Mbps
    - RGMII
  - SDIO (×1 for WIFI)
    - Compatible with 4-bit SDIO 3.0 UHS-I protocol, up to SDR104 (208MHz)

- SD (×1 for TF card)
  - Compatible with 4-bit SD 3.0 UHS-I protocol, up to SDR104 (208MHz)
- eMMC (×1)
  - Compatible with 8bit eMMC5.1, up to HS400 (200MHz)
- MIPI CSI (CSI-2 v1.1) 4-Lane (×2)
  - 4-Lane + 4-Lane mode
  - 4-Lane + 2-Lane mode
  - 4-Lane + 2-Lane + 2-Lane mode (triple sensor)
- MIPI DSI (DSI v1.1) (×1)
  - 4-Lane DSI
- PWM (×20)
- CAN-FD (×1)
- IR-RX (×1)

- **Security System**
  - RISC-V PMP Security
  - Secure eFuse 4K bits
  - Cryptographic engine (TRNG, AES, RSA, ECC, SHA2, HMAC)

- **Debug System**
  - Two JTAGs for both CPU and MCU subsystem
  - UARTs
  - CPU/IO register snapshot after watchdog reboot

- **Boot System**
  - Initial AP boot from SPI-Nand/SPI-NorFlash/eMMC/SD
  - 128KB boot-ROM

- **Aided System**
  - Watchdog design for each CPU/MCU subsystem

- **Operating Temperature**
  - -40°C ~ +85°C (Industrial Standard)

1.3 Multimedia Features

- **GPU**

- IMG BXE-2-32@819MHz, 32KB SLC
- Support for OpenCL3.0/OpenGL ES 3.2/Vulkan1.3

- **VPU (Video Processing Unit)**

  - H.265/H.264/VP8/VP9/MPEG4/MPEG2 decoder 4K@60fps
  - H.265/H.264/VP8/VP9 encoder 4K@30fps
  - Support for simultaneous encoding and decoding at 1080P@60fps
  - Support for simultaneous H264/H265 encoding at 1080P@30fps and H264/H265 decoding at 4K@30fps

- **Display**

  - 1 MIPI DSI-4 lane or SPI interface
  - Support for up to Full HD (1920x1080@60fps)
  - Support for up to 4-full-size-layer composer and maximum 8-layer composer by up-down layer reuse in RDMA channel
  - Support for **cmdlist** mechanism which can configure register parameters by hardware
  - Support for concurrent write-back with both raw and AFBC format
  - Support for dither/crop/rotation in write-back path
  - Support for an advanced MMU (virtual address) mechanism with nearly no page missing in 90/270 degree rotation
  - Support for color key and solid color
  - Support for both advanced error diffusion and pattern based dither for panel
  - Support for both raw and AFBC format image source
  - Support for color saturation/contrast enhancement
  - Support for both video mode and **cmd** mode for panel
  - Support for DDR frequency dynamic changing with embedded DFC buffer
  - HDMI 1.4

- **Camera**

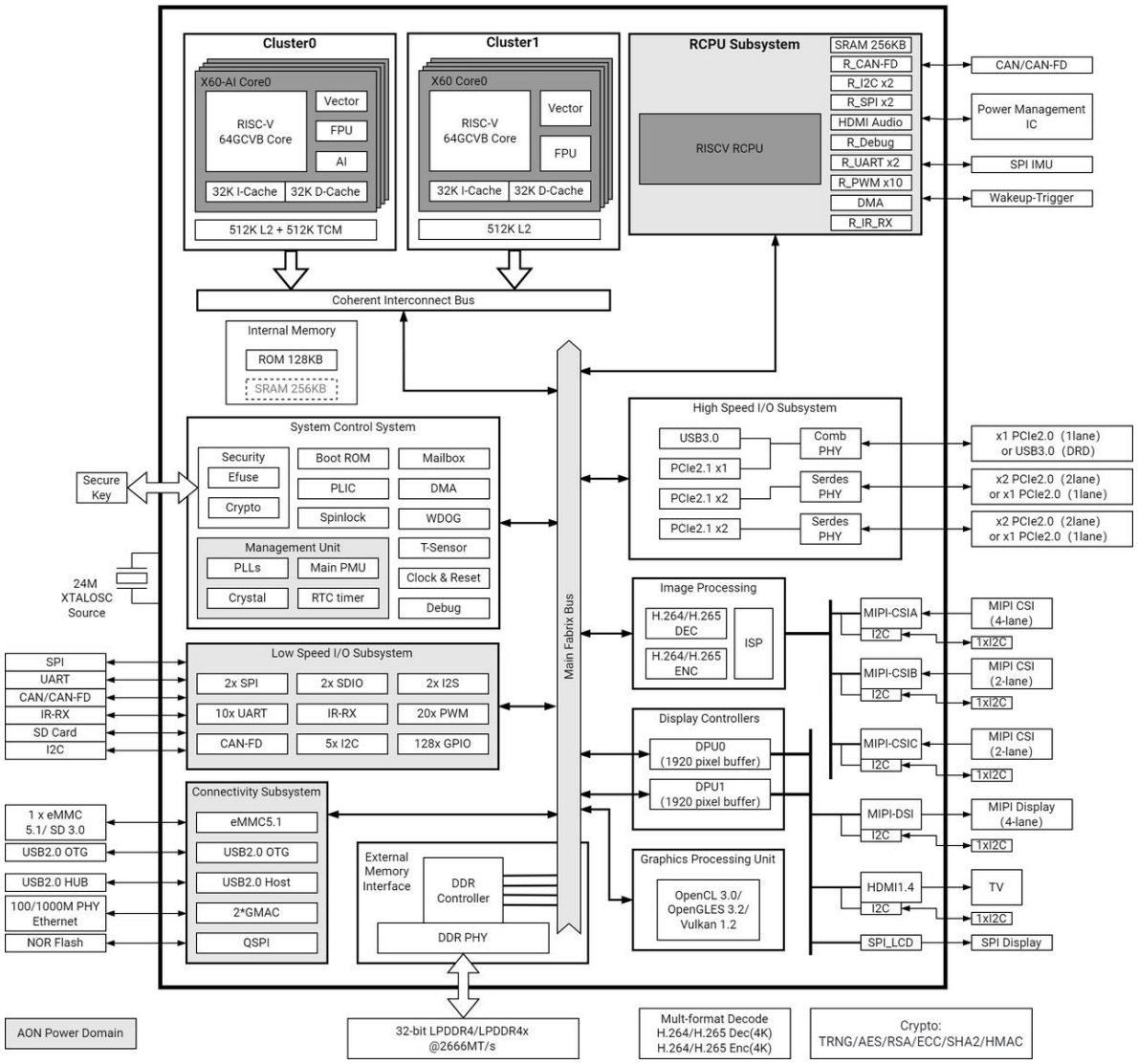
  - Dual-ISP
    - 16M (max) 30fps Dual ISP
    - One 4-Lane CSI + one 4-Lane CSI, or 4-Lane + 2-Lane + 2-Lane
    - RAW sensor, output YUV data to DRAM
    - Hardware JPEG encoder, supporting up to 23M
    - Support for YUV/EXIF/JFIF format
    - AF/AE/AWB
    - Face detection
    - Digital zoom, panorama view
    - PDAF

- ❖ PiP (Picture-in-Picture)
- ❖ Continuous video AF
- ❖ HW 3D denoise

- **Audio**
  - 2 × Full-Duplex I2S Interfaces
  - 1 × HDMI Audio Interface

1.4 Block Diagram

The architecture of K1 is depicted below.



Chapter 2

Package



Key Stone® K1 User Manual

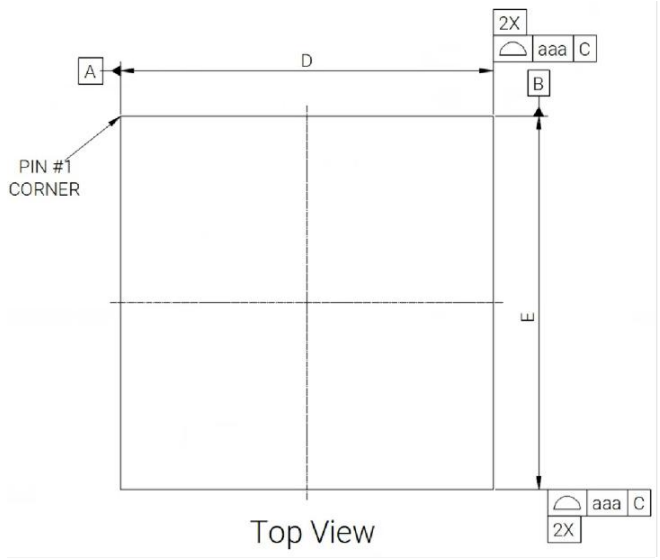
2.1 Introduction

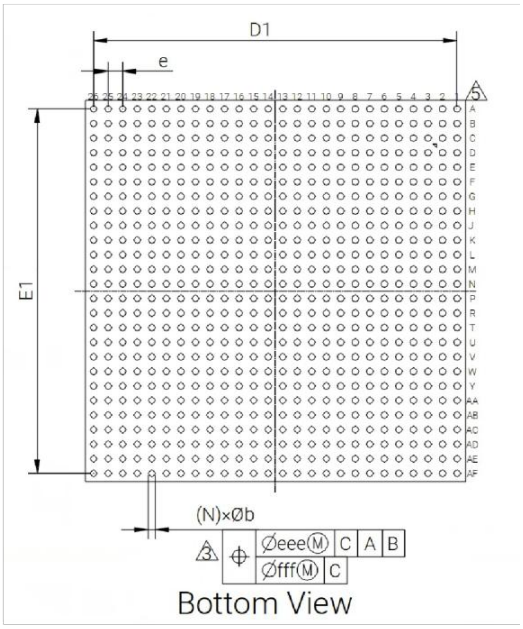
K1 is available in two packages as tabled below.

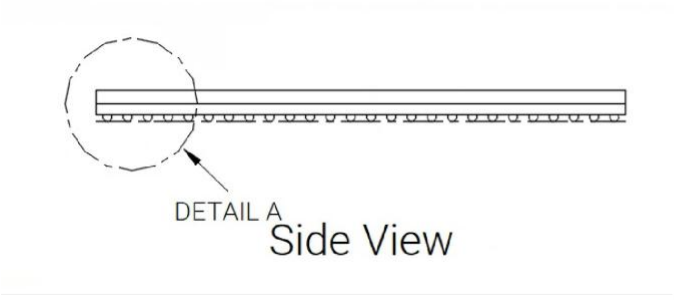
Type	Size	Pin Pitch	Pin Count
FCCSP	17×17 mm	0.65 mm	676 (26x26)
FCBGA	19×19 mm	0.65 mm	676 (26x26)

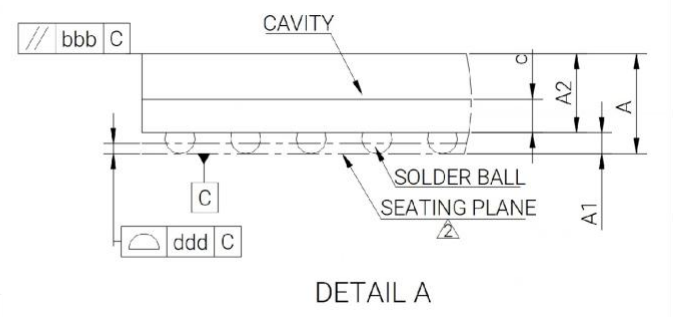
The related package outline drawing (POD) are depicted in the following sections.

2.2 FCCSP Type



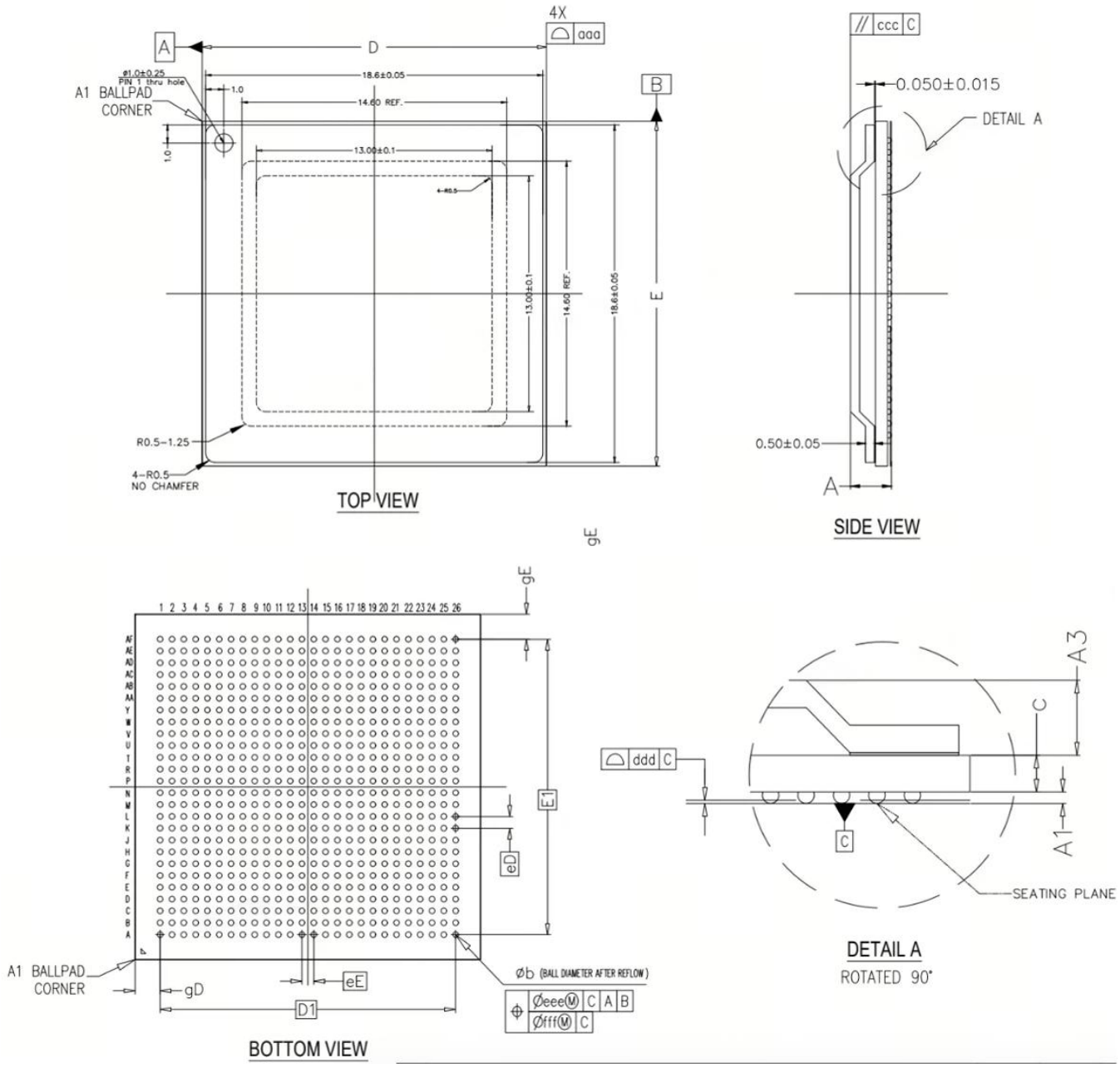






Item		Symbol	Dimension (in mm)		
			Min	Typ	Max
Total thickness		A	0.890	0.990	1.090
Pin stand off		A1	0.160	0.210	0.260
Substrate + Die + Mold		A2	0.710	0.780	0.850
Substrate + Die		c	0.290	0.330	0.370
Body size	X direction	D	16.900	17.000	17.100
	Y direction	E	16.900	17.000	17.100
Edge pin center to center	X direction	D1	—	16.250	—
	Y direction	E1	—	16.250	—
Pin pitch	X/Y direction	e	—	0.650	—
Pin width		b	0.250	0.300	0.350
Package edge tolerance		aaa	0.100		
HAT flatness		bbb	0.100		
Coplanarity		ddd	0.100		
Pin offset (package)		eee	0.150		
Pin offset (ball)		fff	0.080		
Pin diameter			0.300		
Pin count			676		
MD/ME			26/26		

2.3 FCBGA Type



Item		Symbol	Dimension (in mm)		
			Min	Typ	Max
Body size	X direction	D	18.900	19.000	19.100
	Y direction	E	18.900	19.000	19.100
Pin pitch	X direction	eD	0.650		
	Y direction	eE	0.650		
Total thickness		A	2.157	2.257	2.357

Hat + Adhesive		A3	1.322	1.375	1.428
Substrate thickness		c	0.602	0.672	0.742
Pin stand off		A1	0.169	0.210	0.260
Pin width		b	0.250	0.300	0.350
Package edge tolerance		aaa	0.150		
HAT flatness		ccc	0.350		
Coplanarity		ddd	0.080		
Pin offset (package)		eee	0.150		
Pin offset (ball)		fff	0.080		
Pin count		n	676		
Edge pin center to center	X direction	D1	16.250		
	Y direction	E1	16.250		
Edge pin center to package edge	X direction	gD	1.375		
	Y direction	gE	1.375		

Chapter 3

Pinout



Key Stone® K1 User Manual

3.1 Introduction

The two available packages of K1 as per **Chapter 2** are pin-to-pin.

3.2 Pinout Diagram & Description

The overall pinout diagram of K1 is depicted below

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	VSS	VSSQ_DDR	DQ_B_2	DMI0_B	VSSQ_DDR	DQ_B_6	DQ_B_4	DQ_B_13	DQ_B_15	VSSQ_DDR	DQ_B_9	DQ_B_12	DQ_B_11	DQS1_C_B	DQS1_C_A	DQ_A_12	DQ_A_9	DQ_A_8	DQ_A_15	VSSQ_DDR	DQ_A_5	DQ_A_7	DMI0_A	DQ_A_1	VSSQ_DDR	VSS
B	VSS	DQ_B_3	VSSQ_DDR	DQ_B_1	DQ_B_0	DQ_B_7	DQ_B_5	VDDQ_V1P2	DQ_B_14	DMI1_B	DQ_B_8	DQ_B_10	VSSQ_DDR	DQS1_T_B	DQS1_T_A	DQ_A_11	DQ_A_10	DMI1_A	DQ_A_14	DQ_A_13	DQ_A_4	DQ_A_6	VSSQ_DDR	DQ_A_2	DQ_A_3	VSS
C	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	DQS0_T_B	VSSQ_DDR	CS1_B	CA_B_1	CKE0_B	CKE1_B	VDDQ_V1P2	CA_B_5	VSSQ_DDR	VSSQ_DDR	CA_A_4	VSSQ_DDR	CKE1_A	CA_A_1	CS1_A	AVDD06_DDR	DQ_A_0	VSS	EMMC_DS	EMMC_D7	EMMC_D2
D	GPIO_114	GPIO_113	GPIO_112	GPIO_111	GPIO_53	DQS0_C_B	VSSQ_DDR	CA_B_0	VSSQ_DDR	DDR_LP4X_SEL	CK_C_B	CA_B_2	CA_B_4	VSSQ_DDR	AVDD06_DDR	CA_A_2	CK_C_A	CKE0_A	CA_A_0	DQS0_T_A	DQS0_C_A	VSS	EMMC_D4	EMMC_D1	VSS	EMMC_D0
E	GPIO_67	GPIO_65	GPIO_64	VSS	GPIO_63	VSS	VSSQ_DDR	VDDQ_V1P2	DDR_LP23_VREFDQ	VSSQ_DDR	CK_T_B	CA_B_3	AVSS18_DDR	AVDD18_DDR	CA_A_5	CS0_A	CK_T_A	AVDD06_DDR	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	EMMC_D6	AVSS_EMMC	EMMC_CLK	EMMC_D3	EMMC_D5
F	VSS	VSS	GPIO_69	GPIO_68	GPIO_66	VCC18_GPIO	VSS	VSSQ_DDR	VDDQ_V1P2	VDDQ_V1P2	VSSQ_DDR	CS0_B	DDR_RESET_N	ZQ_DDR_PHY	CA_A_3	VSSQ_DDR	DDR_LDO_CAP	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT2	QSPI_DAT1	EMMC_CMD	QSPI_DAT0
G	MIPI_CSI1_D1N	MIPI_CSI1_D1P	VSS	MIPI_CSI1_D0N	MIPI_CSI1_D0P	VSS	VSS	VSS	VSS	VSSQ_DDR	VDDQ_V1P2	AVDD11_DDR	VDDQ_V1P2	DDR_LP23_VREFCA	AVDD11_DDR	AVDD06_DDR	VSSQ_DDR	AVDD18_EFUSE	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT3	QSPI_CLK	QSPI_CS1	VSS	VSS
H	MIPI_CSI1_D2N	MIPI_CSI1_D2P	VSS	MIPI_CSI1_CLKN	MIPI_CSI1_CLKP	AVSS18_AFEAP	XI_PAD	AVSS18_AFEAP	VSS	VSSU_DDR	VSSU_DDR	AVDD18_PHY	AVDDU_DDR	AVSSU_DDR	AVDD18_PHY	VSSU_DDR	VSSU_DDR	VSSU_EMMC	AVDD18_EMMC	AVDD09_EMMC	VCC1833_QSPI	PCIEC_TX0P	PCIEC_TX0N	AVSS_PCIEC	PCIEC_RX0P	PCIEC_RX0N
J	MIPI_CSI3_D0N	MIPI_CSI3_D0P	AVSS_CSI	MIPI_CSI1_D3N	MIPI_CSI1_D3P	AVSS_CSI	XO_PAD	AVSS18_AFEAP	AVSS18_AFEAP	VCC_M1	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	VSS	VCC_M1	VSSU_EMMC	QSPI_VCC_CAP	AVDD09_EMMC	AVSS_PCIEC	PCIEC_REFCLK_P	PCIEC_REFCLK_N	AVSS_PCIEC	PCIEC_RX1P	PCIEC_RX1N
K	MIPI_CSI3_CLKN	MIPI_CSI3_CLKP	AVSS_CSI	MIPI_CSI3_D1N	MIPI_CSI3_D1P	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	VCC_M1	VSS	BG_OUT	AVDD18_AFEAP	MPLL_TST_CK	AVDD18_PLL	VCC_M1	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD09_PCIEB	AVSS_PCIEC	PCIEC_TX1P	PCIEC_TX1N	AVSS_PCIEC	PCIEB_RX0P	PCIEB_RX0N
L	MIPI_CSI3_D2N	MIPI_CSI3_D2P	AVSS_CSI	MIPI_CSI2_CLKN	MIPI_CSI2_CLKP	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	AVSS_CSI	VCC_M1	AVDD09_AFEAP	VSSU_AFEAP	AVSS_PLL	VSSU_PLL	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD18_PCIEC	AVDD09_PCIEB	AVDD09_PCIEB	PCIEB_TX0P	PCIEB_TX0N	AVSS_PCIEB	PCIEB_REFCLK_P	PCIEB_REFCLK_N
M	MIPI_CSI3_D3N	MIPI_CSI3_D3P	VSS	VSS	VSSU_PCIEA	AVDD18_USB	AVDD09_USB	VSSU_PCIEA	AVDD33_USB	VSS	AVDD09_PLL	VSS	VSS	VSS	VCC_M1	VSS	VSSU_PCIEB	VSSU_PCIEB	AVDD18_PCIEB	AVSS_PCIEB	AVSS_PCIEB	PCIEB_TX1P	PCIEB_TX1N	AVSS_PCIEB	PCIEB_RX1P	PCIEB_RX1N
N	USB2_DN	USB2_DP	AVSS_USB	PCIEA_TXN	PCIEA_TXP	AVDD18_PCIEA	AVDD09_PCIEA	AVSS_PCIEA	AVDD33_USB	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AVSS18_AUD	AVDD33_AUD	AVSS18_AUD	AVSS18_AUD	NA	NA	NA	NA	NA	NA
P	PCIEA_RXN	PCIEA_RXP	AVSS_USB	PCIEA_R_EXT	AVSS_USB	AVDD18_USB	AVDD09_USB	AVDD09_USB	AVDD33_USB	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	AUD_GNDSNS	AVDD18_AUD	AVDD18_AUD	NA	NA	NA	NA	NA	NA	NA
R	PCIEA_REFCLK_N	PCIEA_REFCLK_P	VSS	USB1_DN	USB1_DP	AVDD18_DSI1	AVSS_USB	VSS	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AUD_VSSU	AUD_VDDU09	AUD_REFGND	NA	AUD_AUREF10	NA	NA	VSS	NA	NA
T	MIPI_DSI1_D3N	MIPI_DSI1_D3P	VSS	USB0_DN	USB0_DP	VSS	AVDD09_DSI1	AVDD12_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	AVSS18_AUD	AVSS18_AUD	NA	VSS	NA	VSS
U	MIPI_DSI1_D2N	MIPI_DSI1_D2P	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1_FB	VSS_FB	VSS	GPIO_123	GPIO_125	NA	NA	GPIO_126	GPIO_127
V	MIPI_DSI1_CLKN	MIPI_DSI1_CLKP	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	GPIO_121	VSS	GPIO_124	GPIO_120	VSS	GPIO_122
W	VSS	VSS	VSS	MIPI_DSI1_D1N	MIPI_DSI1_D1P	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	GPIO3_VCC_CAP	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	GPIO_110	GPIO_117	GPIO_116	VSS	GPIO_119	GPIO_118
Y	PRI_TST_N	GPIO_74	VSS	MIPI_DSI1_D0N	MIPI_DSI1_D0P	VSS	AVDD33_HDMI	AVDD33_HDMI	AVDD09_HDMI	AVDD09_HDMI	VSS	VSS	VSS	MMC1_VCC_CAP	GPIO2_VCC_CAP	VSS	VSS	VSS	VSS	VSS	VCC18_GPIO	GPIO_26	GPIO_27	VSS	GPIO_28	GPIO_115
AA	PRI_TCK	PRI_TDO	VSS	VSS	VSS	VSS	AVSS_HDMI	HDMI_TX2N	AVDD18_HDMI	AVDD18_HDMI	VSS	VSS	VCC1833_GPIO3	VCC1833_MMC1	VCC1833_GPIO2	MMC1_DAT2	VSS	VSS	GPIO_32	GPIO_29	VCC18_GPIO	GPIO_21	GPIO_24	GPIO_23	GPIO_25	VSS
AB	PRI_TDI	PRI_TMS	VSS	HDMI_TXCN	HDMI_TX0N	AVSS_HDMI	HDMI_TX1N	HDMI_TX2P	AVSS_HDMI	VSS	VSS	VSS	GPIO_51	MMC1_DAT0	GPIO_78	GPIO_77	GPIO_02	GPIO_03	VSS	VSS	GPIO_41	GPIO_44	GPIO_19	VSS	GPIO_20	GPIO_22
AC	GPIO_61	GPIO_62	VCC18_GPIO	HDMI_TXCP	HDMI_TX0P	AVSS_HDMI	HDMI_TX1P	AVSS_HDMI	AVSS_HDMI	GPIO_86	VCC18_GPIO	GPIO_52	GPIO_47	VCC18_GPIO	GPIO_79	VSS	GPIO_05	GPIO_00	VSS	GPIO_31	GPIO_34	GPIO_42	GPIO_43	GPIO_17	VSS	GPIO_18
AD	GPIO_59	GPIO_60	VSS	VSS	VSS	VSS	VSS	GPIO_87	GPIO_85	PMIC_INT_N	VCC18_GPIO	GPIO_50	GPIO_48	MMC1_CMD	GPIO_76	VSS	GPIO_04	GPIO_01	GPIO_30	GPIO_33	VCC18_GPIO	VCC18_GPIO	GPIO_14	GPIO_12	GPIO_16	GPIO_15
AE	VSS	MPLL_TST_AD	VSS	GPIO_92	GPIO_90	GPIO_91	GPIO_89	GPIO_84	GPIO_81	DVL0	PWR_SCL	EXT_32K_IN	VSS	MMC1_CLK	MMC1_DAT3	GPIO_75	GPIO_11	GPIO_07	GPIO_10	GPIO_37	GPIO_35	GPIO_38	GPIO_46	VSS	GPIO_13	VSS
AF	VSS	VSS	RESET_IN_N	JTAG_SEL	VSS	GPIO_88	GPIO_82	GPIO_83	DVL1	VSS	SLEEP_OUT	PWR_SDA	GPIO_49	MMC1_DAT1	VSS	GPIO_80	GPIO_08	GPIO_06	GPIO_09	VSS	GPIO_40	GPIO_36	GPIO_39	GPIO_45	VSS	VSS

Note. Meaning of the different colors:

- Power supplies (different voltages):
  - Brown
  - Dark Blue
  - Grey
  - Light Blue
  - Orange
  - Purple
  - Red
  - Yellow

- Grounds:
  - Dark Green
  - Light Green
- Signals:
  - White

Let's consider the division into the quadrants

- (A~N, 1~13)
- (A~N, 14~26)
- (M~AF, 1~13)
- (M~AF, 14~26)

in order to provide conveniently the pinout description of K1 in the following subsections.

3.2.1 (A~N, 1~13)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	VSS	VSSQ_DDR	DQ_B_2	DMI0_B	VSSQ_DDR	DQ_B_6	DQ_B_4	DQ_B_13	DQ_B_15	VSSQ_DDR	DQ_B_9	DQ_B_12	DQ_B_11	DQS1_C_B	DQS1_C_A	DQ_A_12	DQ_A_9	DQ_A_8	DQ_A_15	VSSQ_DDR	DQ_A_5	DQ_A_7	DMI0_A	DQ_A_1	VSSQ_DDR	VSS
B	VSS	DQ_B_3	VSSQ_DDR	DQ_B_1	DQ_B_0	DQ_B_7	DQ_B_5	VDDQ_V1P2	DQ_B_14	DMI1_B	DQ_B_8	DQ_B_10	VSSQ_DDR	DQS1_T_B	DQS1_T_A	DQ_A_11	DQ_A_10	DMI1_A	DQ_A_14	DQ_A_13	DQ_A_4	DQ_A_6	VSSQ_DDR	DQ_A_2	DQ_A_3	VSS
C	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	DQS0_T_B	VSSQ_DDR	CS1_B	CA_B_1	CKE0_B	CKE1_B	VDDQ_V1P2	CA_B_5	VSSQ_DDR	VSSQ_DDR	CA_A_4	VSSQ_DDR	CKE1_A	CA_A_1	CS1_A	AVDD06_DDR	DQ_A_0	VSS	EMMC_DS	EMMC_D7	EMMC_D2
D	GPIO_114	GPIO_113	GPIO_112	GPIO_111	GPIO_53	DQS0_C_B	VSSQ_DDR	CA_B_0	VSSQ_DDR	DDR_LP4X_SEL	CK_C_B	CA_B_2	CA_B_4	VSSQ_DDR	AVDD06_DDR	CA_A_2	CK_C_A	CKE0_A	CA_A_0	DQS0_T_A	DQS0_C_A	VSS	EMMC_D4	EMMC_D1	VSS	EMMC_D0
E	GPIO_67	GPIO_65	GPIO_64	VSS	GPIO_63	VSS	VSSQ_DDR	VDDQ_V1P2	DDR_LP23_VREFDQ	VSSQ_DDR	CK_T_B	CA_B_3	AVSS18_DDR	AVDD18_DDR	CA_A_5	CS0_A	CK_T_A	AVDD06_DDR	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	EMMC_D6	AVSS_EMMC	EMMC_CLK	EMMC_D3	EMMC_D5
F	VSS	VSS	GPIO_69	GPIO_68	GPIO_66	VCC18_GPIO	VSS	VSSQ_DDR	VDDQ_V1P2	VDDQ_V1P2	VSSQ_DDR	CS0_B	DDR_RESET_N	ZQ_DDR_PHY	CA_A_3	VSSQ_DDR	DDR_LDO_CAP	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT2	QSPI_DAT1	EMMC_CMD	QSPI_DAT0
G	MIPI_CSI1_D1N	MIPI_CSI1_D1P	VSS	MIPI_CSI1_D0N	MIPI_CSI1_D0P	VSS	VSS	VSS	VSS	VSSQ_DDR	VDDQ_V1P2	AVDD11_DDR	VDDQ_V1P2	DDR_LP23_VREFCA	AVDD11_DDR	AVDD06_DDR	VSSQ_DDR	AVDD18_EFUSE	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT3	QSPI_CLK	QSPI_CS1	VSS	VSS
H	MIPI_CSI1_D2N	MIPI_CSI1_D2P	VSS	MIPI_CSI1_CLKN	MIPI_CSI1_CLKP	AVSS18_AFEAP	XI_PAD	AVSS18_AFEAP	VSS	VSSU_DDR	VSSU_DDR	AVDD18_PHY	AVDDU_DDR	AVSSU_DDR	AVDD18_PHY	VSSU_DDR	VSSU_DDR	VSSU_EMMC	AVDD18_EMMC	AVDD09_EMMC	VCC1833_QSPI	PCIEC_TX0P	PCIEC_TX0N	AVSS_PCIEC	PCIEC_RX0P	PCIEC_RX0N
J	MIPI_CSI3_D0N	MIPI_CSI3_D0P	AVSS_CSI	MIPI_CSI1_D3N	MIPI_CSI1_D3P	AVSS_CSI	XO_PAD	AVSS18_AFEAP	AVSS18_AFEAP	VCC_M1	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	VSS	VCC_M1	VSSU_EMMC	QSPI_VCC_CAP	AVDD09_EMMC	AVSS_PCIEC	PCIEC_REFCLK_P	PCIEC_REFCLK_N	AVSS_PCIEC	PCIEC_RX1P	PCIEC_RX1N
K	MIPI_CSI3_CLKN	MIPI_CSI3_CLKP	AVSS_CSI	MIPI_CSI3_D1N	MIPI_CSI3_D1P	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	VCC_M1	VSS	BG_OUT	AVDD18_AFEAP	MPLL_TST_CK	AVDD18_PLL	VCC_M1	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD09_PCIEC	AVSS_PCIEC	PCIEC_TX1P	PCIEC_TX1N	AVSS_PCIEC	PCIEB_RX0P	PCIEB_RX0N
L	MIPI_CSI3_D2N	MIPI_CSI3_D2P	AVSS_CSI	MIPI_CSI2_CLKN	MIPI_CSI2_CLKP	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	AVSS_CSI	VCC_M1	AVDD09_AFEAP	VSSU_AFEAP	AVSS_PLL	VSSU_PLL	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD18_PCIEC	AVDD09_PCIEB	AVDD09_PCIEB	PCIEB_TX0P	PCIEB_TX0N	AVSS_PCIEB	PCIEB_REFCLK_P	PCIEB_REFCLK_N
M	MIPI_CSI3_D3N	MIPI_CSI3_D3P	VSS	VSS	VSSU_PCIEA	AVDD18_USB	AVDD09_USB	VSSU_PCIEA	AVDD33_USB	VSS	AVDD09_PLL	VSS	VSS	VSS	VCC_M1	VSS	VSSU_PCIEB	VSSU_PCIEB	AVDD18_PCIEB	AVSS_PCIEB	AVSS_PCIEB	PCIEB_TX1P	PCIEB_TX1N	AVSS_PCIEB	PCIEB_RX1P	PCIEB_RX1N
N	USB2_DN	USB2_DP	AVSS_USB	PCIEA_TXN	PCIEA_TXP	AVDD18_PCIEA	AVDD09_PCIEA	AVSS_PCIEA	AVDD33_USB	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AVSS18_AUD	AVDD33V3_AUD	AVSS18_AUD	AVSS18_AUD	NA	NA	NA	NA	NA	NA
P	PCIEA_RXN	PCIEA_RXP	AVSS_USB	PCIEA_R_EXT	AVSS_USB	AVDD18_USB	AVDD09_USB	AVDD09_USB	AVDD33_USB	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	AUD_GNDSNS	AVDD18_AUD	AVDD18_AUD	NA	NA	NA	NA	NA	NA	NA
R	PCIEA_REFCLK_N	PCIEA_REFCLK_P	VSS	USB1_DN	USB1_DP	AVDD18_DSI1	AVSS_USB	VSS	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AUD_VSSU	AUD_VDDU09	AUD_REFGND	NA	AUD_AUREF10	NA	NA	VSS	NA	NA
T	MIPI_DSI1_D3N	MIPI_DSI1_D3P	VSS	USB0_DN	USB0_DP	VSS	AVDD09_DSI1	AVDD12_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	AVSS18_AUD	AVSS18_AUD	NA	VSS	NA	VSS
U	MIPI_DSI1_D2N	MIPI_DSI1_D2P	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1_FB	VSS_FB	VSS	GPIO_123	GPIO_125	NA	NA	GPIO_126	GPIO_127
V	MIPI_DSI1_CLKN	MIPI_DSI1_CLKP	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	GPIO_121	VSS	GPIO_124	GPIO_120	VSS	GPIO_122
W	VSS	VSS	VSS	MIPI_DSI1_D1N	MIPI_DSI1_D1P	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	GPIO3_VCC_CAP	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	GPIO_110	GPIO_117	GPIO_116	VSS	GPIO_119	GPIO_118
Y	PRI_TST_N	GPIO_74	VSS	MIPI_DSI1_D0N	MIPI_DSI1_D0P	VSS	AVDD33_HDMI	AVDD33_HDMI	AVDD09_HDMI	AVDD09_HDMI	VSS	VSS	VSS	MMC1_VCC_CAP	GPIO2_VCC_CAP	VSS	VSS	VSS	VSS	VSS	VCC18_GPIO	GPIO_26	GPIO_27	VSS	GPIO_28	GPIO_115
AA	PRI_TCK	PRI_TDO	VSS	VSS	VSS	VSS	AVSS_HDMI	HDMI_TX2N	AVDD18_HDMI	AVDD18_HDMI	VSS	VSS	VCC1833_GPIO3	VCC1833_MMC1	VCC1833_GPIO2	MMC1_DAT2	VSS	VSS	GPIO_32	GPIO_29	VCC18_GPIO	GPIO_21	GPIO_24	GPIO_23	GPIO_25	VSS
AB	PRI_TDI	PRI_TMS	VSS	HDMI_TXCN	HDMI_TX0N	AVSS_HDMI	HDMI_TX1N	HDMI_TX2P	AVSS_HDMI	VSS	VSS	VSS	GPIO_51	MMC1_DAT0	GPIO_78	GPIO_77	GPIO_02	GPIO_03	VSS	VSS	GPIO_41	GPIO_44	GPIO_19	VSS	GPIO_20	GPIO_22
AC	GPIO_61	GPIO_62	VCC18_GPIO	HDMI_TXCP	HDMI_TX0P	AVSS_HDMI	HDMI_TX1P	AVSS_HDMI	AVSS_HDMI	GPIO_86	VCC18_GPIO	GPIO_52	GPIO_47	VCC18_GPIO	GPIO_79	VSS	GPIO_05	GPIO_00	VSS	GPIO_31	GPIO_34	GPIO_42	GPIO_43	GPIO_17	VSS	GPIO_18
AD	GPIO_59	GPIO_60	VSS	VSS	VSS	VSS	VSS	GPIO_87	GPIO_85	PMIC_INT_N	VCC18_GPIO	GPIO_50	GPIO_48	MMC1_CMD	GPIO_76	VSS	GPIO_04	GPIO_01	GPIO_30	GPIO_33	VCC18_GPIO	VCC18_GPIO	GPIO_14	GPIO_12	GPIO_16	GPIO_15
AE	VSS	MPLL_TST_AD	VSS	GPIO_92	GPIO_90	GPIO_91	GPIO_89	GPIO_84	GPIO_81	DVL0	PWR_SCL	EXT_32K_IN	VSS	MMC1_CLK	MMC1_DAT3	GPIO_75	GPIO_11	GPIO_07	GPIO_10	GPIO_37	GPIO_35	GPIO_38	GPIO_46	VSS	GPIO_13	VSS
AF	VSS	VSS	RESET_IN_N	JTAG_SEL	VSS	GPIO_88	GPIO_82	GPIO_83	DVL1	VSS	SLEEP_OUT	PWR_SDA	GPIO_49	MMC1_DAT1	VSS	GPIO_80	GPIO_08	GPIO_06	GPIO_09	VSS	GPIO_40	GPIO_36	GPIO_39	GPIO_45	VSS	VSS

Note. Definition of symbols used for pin type:

- AO = Analog output
- AI = Analog input
- AIO = Analog input/output
- G = Ground
- I/O = Input/Output
- P = Power
- RO = Reference output

Pin ID	Name	Type	Power Domain	Function
A1	VSS	G	0V	Digital Core Ground
A2	VSSQ_DDR	G	0V	DDR Ground
A3	DQ_B_2	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ2 LPDDR3: DQ28
A4	DMI0_B	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Channel B DM0 LPDDR3: DQ25
A5	VSSQ_DDR	G	0V	DDR Ground
A6	DQ_B_6	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ6 LPDDR3: DQ24
A7	DQ_B_4	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ4 LPDDR3: DQ30
A8	DQ_B_13	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ13 LPDDR3: DQ15
A9	DQ_B_15	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ15 LPDDR3: DQ12
A10	VSSQ_DDR	G	0V	DDR Ground
A11	DQ_B_9	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ9 LPDDR3: DQ8
A12	DQ_B_12	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ12 LPDDR3: DQ10
A13	DQ_B_11	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ11 LPDDR3: DQ11
B1	VSS	G	0V	Digital Core Ground
B2	DQ_B_3	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ3 LPDDR3: DQM3
B3	VSSQ_DDR	G	0V	DDR Ground
B4	DQ_B_1	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ1 LPDDR3: DQ27

Pin ID	Name	Type	Power Domain	Function
B5	DQ_B_0	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ0 LPDDR3: DQ31
B6	DQ_B_7	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ7 LPDDR3: DQ29
B7	DQ_B_5	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ5 LPDDR3: DQ26
B8	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
B9	DQ_B_14	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ14 LPDDR3: DQ13
B10	DMI1_B	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Channel B DM1 LPDDR3: DQ14
B11	DQ_B_8	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ12 LPDDR3: DQM1
B12	DQ_B_10	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ10 LPDDR3: DQ9
B13	VSSQ_DDR	G	0V	DDR Ground
C1	GPIO_58	I/O	1.8V	General Purpose I/O 58
C2	GPIO_57	I/O	1.8V	General Purpose I/O 57
C3	GPIO_56	I/O	1.8V	General Purpose I/O 56
C4	GPIO_55	I/O	1.8V	General Purpose I/O 55
C5	GPIO_54	I/O	1.8V	General Purpose I/O 54
C6	DQS0_T_B	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Positive of CHB DQS0 LPDDR3: Positive of DQS3
C7	VSSQ_DDR	G	0V	DDR Ground
C8	CS1_B	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Active-low chip select 1 of CHB LPDDR3: N/A
C9	CA_B_1	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB CA1 LPDDR3: CA5
C10	CKE0_B	AO	lp3: 1.2V lp4x: 1.1V	LPDDR4X: clock enabling 0 of CHB LPDDR3: N/A
C11	CKE1_B	AO	lp3: 1.2V lp4x: 1.1V	LPDDR4X: clock enabling 1 of CHB LPDDR3: N/A
C12	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
C13	CA_B_5	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB CA5 LPDDR3: CA8

Pin ID	Name	Type	Power Domain	Function
D1	GPIO_114	I/O	1.8V	General Purpose I/O 114
D2	GPIO_113	I/O	1.8V	General Purpose I/O 113
D3	GPIO_112	I/O	1.8V	General Purpose I/O 112
D4	GPIO_111	I/O	1.8V	General Purpose I/O 111
D5	GPIO_53	I/O	1.8V	General Purpose I/O 53
D6	DQS0_C_B	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Negative of CHB DQS0 LPDDR3: Negtive of DQS3
D7	VSSQ_DDR	G	0V	DDR Ground
D8	CA_B_0	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB CA0
D9	VSSQ_DDR	G	0V	DDR Ground
D10	DDR_lp4x_SEL	AIO	1.8V	LPDDR4X: connect to 1.8V LP234: connect to Ground
D11	CK_C_B	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: negative LPDDR differential clock of CHB LPDDR3: negative LPDDR differential clock
D12	CA_B_2	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB CA2 LPDDR3: CA9
D13	CA_B_4	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA4 LPDDR3: CA7
E1	GPIO_67	I/O	1.8V	General Purpose I/O 67
E2	GPIO_65	I/O	1.8V	General Purpose I/O 65
E3	GPIO_64	I/O	1.8V	General Purpose I/O 64
E4	VSS	G	0V	Digital Core Ground
E5	GPIO_63	I/O	1.8V	General Purpose I/O 63
E6	VSS	G	0V	Digital Core Ground
E7	VSSQ_DDR	G	0V	DDR Ground
E8	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
E9	DDR_LP23_VREFDQ	P	lp3: 0.6V lp4: high-z	DQ VREF for lpddr23 , LP4/4x Keep the pin NC
E10	VSSQ_DDR	G	0V	DDR Ground
E11	CK_T_B	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: positive LPDDR differential clock of CHB LPDDR3: positive LPDDR differential clock
E12	CA_B_3	AO	lp3: 1.2V	LPDDR4X: CHB CA3

Pin ID	Name	Type	Power Domain	Function
			lp4x: 0.6V	LPDDR3: CA6
E13	AVSS18_DDR	G	0V	DDR Ground
F1	VSS	G	0V	Digital Core Ground
F2	VSS	G	0V	Digital Core Ground
F3	GPIO_69	I/O	1.8V	General Purpose I/O 69
F4	GPIO_68	I/O	1.8V	General Purpose I/O 68
F5	GPIO_66	I/O	1.8V	General Purpose I/O 66
F6	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
F7	VSS	G	0V	Digital Core Ground
F8	VSSQ_DDR	G	0V	DDR Ground
F9	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
F10	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
F11	VSSQ_DDR	G	0V	DDR Ground
F12	CS0_B	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: clock enabling 1 of CHB LPDDR3: N/A
F13	DDR_RESET_N	AO	lp3: 1.2V lp4x: 1.1V	LPDDR SDRAM reset
G1	MIPI_CSI1_D1N	AI	1.8V	CSI1 DATA1LANEN
G2	MIPI_CSI1_D1P	AI	1.8V	CSI1 DATA1LANEP
G3	VSS	G	0V	Digital Core Ground
G4	MIPI_CSI1_D0N	AI	1.8V	CSI1 DATA0LANEN
G5	MIPI_CSI1_D0P	AI	1.8V	CSI1 DATA0LANEP
G6	VSS	G	0V	Digital Core Ground
G7	VSS	G	0V	Digital Core Ground
G8	VSS	G	0V	Digital Core Ground
G9	VSS	G	0V	Digital Core Ground
G10	VSSQ_DDR	G	0V	DDR Ground
G11	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power

Pin ID	Name	Type	Power Domain	Function
G12	AVDD11_DDR	P	lp4x: 1.1V lp4: 1.1V lp3: 1.2V	LPDDR PHY power supply
G13	VDDQ_V1P2	P	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
H1	MIPI_CSI1_D2N	AI	1.8V	CSI1 DATA2LANEN
H2	MIPI_CSI1_D2P	AI	1.8V	CSI1 DATA2LANEP
H3	VSS	G	0V	Digital Core Ground
H4	MIPI_CSI1_CLKN	AO	1.8V	CSI1 CKLANEN
H5	MIPI_CSI1_CLKP	AO	1.8V	CSI1 CKLANEP
H6	AVSS18_AFEAP	G	0V	DCXO Ground
H7	XI_PAD	AI	1.8V	DCXO crystal input
H8	AVSS18_AFEAP	G	0V	DCXO Ground
H9	VSS	G	0V	Digital Core Ground
H10	VSSU_DDR	G	0V	system DDR Ground
H11	VSSU_DDR	G	0V	system DDR Ground
H12	AVDD18_PHY	P	1.8V	Analog 1.8V power
H13	AVDDU_DDR	P	0.9V	LPDDR PHY PLL logical power
J1	MIPI_CSI3_D0N	AI	1.8V	CSI3 DATA0LANEN
J2	MIPI_CSI3_D0P	AI	1.8V	CSI3 DATA0LANEP
J3	AVSS_CSI	G	0V	MIPI_CSI Ground
J4	MIPI_CSI1_D3N	AI	1.8V	CSI1 DATA3LANEN
J5	MIPI_CSI1_D3P	AI	1.8V	CSI1 DATA3LANEP
J6	AVSS_CSI	G	0V	MIPI_CSI Ground
J7	XO_PAD	AO	1.8V	DCXO crystal output
J8	AVSS18_AFEAP	G	0V	DCXO Ground

Pin ID	Name	Type	Power Domain	Function
J9	AVSS18_AFEAP	G	0V	DCXO Ground
J10	VCC_M1	P	0.9V	Digital Core power
J11	AVDDU_PHY	P	0.9V	LPDDR PHY core logical power
J12	AVDDU_PHY	P	0.9V	LPDDR PHY core logical power
J13	AVDDU_PHY	P	0.9V	LPDDR PHY core logical power
K1	MIPI_CSI3_CLKN	AO	1.8V	CSI3 CKLANEN for CSI3 DATALANE0/1 when CSI3 is configured as two 2ch CSI; CSI3 CKLANEN for CSI3 DATALANE0/1/2/3 when CSI3 is configured as 4ch CSI
K2	MIPI_CSI3_CLKP	AO	1.8V	CSI3 CKLANEP for CSI3 DATALANE0/1 when CSI3 is configured as two 2ch CSI; CSI3 CKLANEP for CSI3 DATALANE0/1/2/3 when CSI3 is configured as 4ch CSI
K3	AVSS_CSI	G	0V	MIPI_CSI Ground
K4	MIPI_CSI3_D1N	AI	1.8V	CSI3 DATA1LANEN
K5	MIPI_CSI3_D1P	AI	1.8V	CSI3 DATA1LANEP
K6	AVDD18_CSI	P	1.8V	MIPI_CSI analog power
K7	AVDD09_CSI	P	0.9V	MIPI_CSI digtial power
K8	AVSS_CSI	G	0V	MIPI_CSI Ground
K9	VCC_M1	P	0.9V	Digital Core power
K10	VSS	G	0V	Digital Core Ground
K11	BG_OUT	AO	1.8V	Bandgap output
K12	AVDD18_AFEAP	P	1.8V	1.8V power for DCXO
K13	MPLL_TST_CK	AIO	1.8V	Analog testpin
L2	MIPI_CSI3_D2P	AI	1.8V	CSI3 DATA2LANEP
L3	AVSS_CSI	G	0V	MIPI_CSI Ground
L4	MIPI_CSI2_CLKN	AO	1.8V	CKLANEN for CSI3 DATALANE2/3 when CSI3 is configured as two 2ch CSI; Disabled when CSI3 is configured as 4ch CSI
L5	MIPI_CSI2_CLKP	AO	1.8V	CKLANEP for CSI3 DATALANE2/3 when CSI3 is configured as two 2ch CSI; Disabled when CSI3 is configured as 4ch CSI

Pin ID	Name	Type	Power Domain	Function
L6	AVDD18_CSI	P	1.8V	MIPI_CSI analog power
L7	AVDD09_CSI	P	0.9V	MIPI_CSI digtial power
L8	AVSS_CSI	G	0V	MIPI_CSI Ground
L9	AVSS_CSI	G	0V	MIPI_CSI Ground
L10	VCC_M1	P	0.9V	Digital Core power
L11	AVDD09_AFEAP	P	0.9V	0.9V power for DCXO
L12	VSSU_AFEAP	G	0V	DCXO Ground
L13	AVSS_PLL	G	0V	Analog Core Ground
M1	MIPI_CSI3_D3N	AI	1.8V	CSI3 DATA3LANEN
M2	MIPI_CSI3_D3P	AI	1.8V	CSI3 DATA3LANEP
M3	VSS	G	0V	Digital Core Ground
M4	VSS	G	0V	Digital Core Ground
M5	VSSU_PCIEA	G	0V	PCIEA Ground
M6	AVDD18_USB	P	1.8V	USB2.0 1.8V power
M7	AVDD09_USB	P	0.9V	USB2.0 digital power
M8	VSSU_PCIEA	G	0V	PCIEA Ground
M9	AVDD33_USB	P	3.3V	USB2.0 3.3V power
M10	VSS	G	0V	Digital Core Ground
M11	AVDD09_PLL	P	0.9	System PLL power supply
M12	VSS	G	0V	Digital Core Ground
M13	VSS	G	0V	Digital Core Ground
N1	USB2_DN	AIO	3.3V	USB2.0_2 D- differential data line
N2	USB2_DP	AIO	3.3V	USB2.0_2 D+ differential data line
N3	AVSS_USB	G	0V	USB2.0 Ground
N4	PCIEA_TXN	AO	1.8V	PCIEA TXLANEN
N5	PCIEA_TXP	AO	1.8V	PCIEA TXLANEP
N6	AVDD18_PCIEA	P	1.8V	PCIEA analog power
N7	AVDD09_PCIEA	P	0.9V	PCIEA digital power
N8	AVSS_PCIEA	G	0V	PCIEA Ground

Pin ID	Name	Type	Power Domain	Function
N9	AVDD33_USB	P	3.3V	USB2.0 3.3V power
N10	VCC_M1	P	0.9V	Digital Core power
N11	VSS	G	0V	Digital Core Ground
N12	VCC_M1	P	0.9V	Digital Core power
N13	VSS	G	0V	Digital Core Ground

3.2.2 (A~N, 14~26)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	VSS	VSSQ_DDR	DQ_B_2	DMI0_B	VSSQ_DDR	DQ_B_6	DQ_B_4	DQ_B_13	DQ_B_15	VSSQ_DDR	DQ_B_9	DQ_B_12	DQ_B_11	DQS1_C_B	DQS1_C_A	DQ_A_12	DQ_A_9	DQ_A_8	DQ_A_15	VSSQ_DDR	DQ_A_5	DQ_A_7	DMI0_A	DQ_A_1	VSSQ_DDR	VSS
B	VSS	DQ_B_3	VSSQ_DDR	DQ_B_1	DQ_B_0	DQ_B_7	DQ_B_5	VDDQ_V1P2	DQ_B_14	DMI1_B	DQ_B_8	DQ_B_10	VSSQ_DDR	DQS1_T_B	DQS1_T_A	DQ_A_11	DQ_A_10	DMI1_A	DQ_A_14	DQ_A_13	DQ_A_4	DQ_A_6	VSSQ_DDR	DQ_A_2	DQ_A_3	VSS
C	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	DQS0_T_B	VSSQ_DDR	CS1_B	CA_B_1	CKE0_B	CKE1_B	VDDQ_V1P2	CA_B_5	VSSQ_DDR	VSSQ_DDR	CA_A_4	VSSQ_DDR	CKE1_A	CA_A_1	CS1_A	AVDD06_DDR	DQ_A_0	VSS	EMMC_DS	EMMC_D7	EMMC_D2
D	GPIO_114	GPIO_113	GPIO_112	GPIO_111	GPIO_53	DQS0_C_B	VSSQ_DDR	CA_B_0	VSSQ_DDR	DDR_LP4X_SEL	CK_C_B	CA_B_2	CA_B_4	VSSQ_DDR	AVDD06_DDR	CA_A_2	CK_C_A	CKE0_A	CA_A_0	DQS0_T_A	DQS0_C_A	VSS	EMMC_D4	EMMC_D1	VSS	EMMC_D0
E	GPIO_67	GPIO_65	GPIO_64	VSS	GPIO_63	VSS	VSSQ_DDR	VDDQ_V1P2	DDR_LP23_VREFDQ	VSSQ_DDR	CK_T_B	CA_B_3	AVSS18_DDR	AVDD18_DDR	CA_A_5	CS0_A	CK_T_A	AVDD06_DDR	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	EMMC_D6	AVSS_EMMC	EMMC_CLK	EMMC_D3	EMMC_D5
F	VSS	VSS	GPIO_69	GPIO_68	GPIO_66	VCC18_GPIO	VSS	VSSQ_DDR	VDDQ_V1P2	VDDQ_V1P2	VSSQ_DDR	CS0_B	DDR_RESET_N	ZQ_DDR_PHY	CA_A_3	VSSQ_DDR	DDR_LDQ_CAP	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT2	QSPI_DAT1	EMMC_CMD	QSPI_DAT0
G	MIPI_CSI1_D1N	MIPI_CSI1_D1P	VSS	MIPI_CSI1_D0N	MIPI_CSI1_D0P	VSS	VSS	VSS	VSS	VSSQ_DDR	VDDQ_V1P2	AVDD11_DDR	VDDQ_V1P2	DDR_LP23_VREFCA	AVDD11_DDR	AVDD06_DDR	VSSQ_DDR	AVDD18_EFUSE	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT3	QSPI_CLK	QSPI_CS1	VSS	VSS
H	MIPI_CSI1_D2N	MIPI_CSI1_D2P	VSS	MIPI_CSI1_CLKN	MIPI_CSI1_CLKP	AVSS18_AFEAP	XI_PAD	AVSS18_AFEAP	VSS	VSSU_DDR	VSSU_DDR	AVDD18_PHY	AVDDU_DDR	AVSSU_DDR	AVDD18_PHY	VSSU_DDR	VSSU_DDR	VSSU_EMMC	AVDD18_EMMC	AVDD09_EMMC	VCC1833_QSPI	PCIEC_TX0P	PCIEC_TX0N	AVSS_PCIEC	PCIEC_RX0P	PCIEC_RX0N
J	MIPI_CSI3_D0N	MIPI_CSI3_D0P	AVSS_CSI	MIPI_CSI1_D3N	MIPI_CSI1_D3P	AVSS_CSI	XO_PAD	AVSS18_AFEAP	AVSS18_AFEAP	VCC_M1	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	VSS	VCC_M1	VSSU_EMMC	QSPI_VCC_CAP	AVDD09_EMMC	AVSS_PCIEC	PCIEC_REFCLK_P	PCIEC_REFCLK_N	AVSS_PCIEC	PCIEC_RX1P	PCIEC_RX1N
K	MIPI_CSI3_CLKN	MIPI_CSI3_CLKP	AVSS_CSI	MIPI_CSI3_D1N	MIPI_CSI3_D1P	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	VCC_M1	VSS	BG_OUT	AVDD18_AFEAP	MPLL_TST_CK	AVDD18_PLL	VCC_M1	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD09_PCIEC	AVSS_PCIEC	PCIEC_TX1P	PCIEC_TX1N	AVSS_PCIEC	PCIEB_RX0P	PCIEB_RX0N
L	MIPI_CSI3_D2N	MIPI_CSI3_D2P	AVSS_CSI	MIPI_CSI2_CLKN	MIPI_CSI2_CLKP	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	AVSS_CSI	VCC_M1	AVDD09_AFEAP	VSSU_AFEAP	AVSS_PLL	VSSU_PLL	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD18_PCIEC	AVDD09_PCIEB	AVDD09_PCIEB	PCIEB_TX0P	PCIEB_TX0N	AVSS_PCIEB	PCIEB_REFCLK_P	PCIEB_REFCLK_N
M	MIPI_CSI3_D3N	MIPI_CSI3_D3P	VSS	VSS	VSSU_PCIEA	AVDD18_USB	AVDD09_USB	VSSU_PCIEA	AVDD33_USB	VSS	AVDD09_PLL	VSS	VSS	VSS	VCC_M1	VSS	VSSU_PCIEB	VSSU_PCIEB	AVDD18_PCIEB	AVSS_PCIEB	AVSS_PCIEB	PCIEB_TX1P	PCIEB_TX1N	AVSS_PCIEB	PCIEB_RX1P	PCIEB_RX1N
N	USB2_DN	USB2_DP	AVSS_USB	PCIEA_TXN	PCIEA_TXP	AVDD18_PCIEA	AVDD09_PCIEA	AVSS_PCIEA	AVDD33_USB	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AVSS18_AUD	AVDD33V3_AUD	AVSS18_AUD	AVSS18_AUD	NA	NA	NA	NA	NA	NA
P	PCIEA_RXN	PCIEA_RXP	AVSS_USB	PCIEA_REXT	AVSS_USB	AVDD18_USB	AVDD09_USB	AVDD09_USB	AVDD33_USB	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	AUD_GNDSNS	AVDD18_AUD	AVDD18_AUD	NA	NA	NA	NA	NA	NA	NA
R	PCIEA_REFCLK_N	PCIEA_REFCLK_P	VSS	USB1_DN	USB1_DP	AVDD18_DSI1	AVSS_USB	VSS	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AUD_VSSU	AUD_VDDU09	AUD_REFGND	NA	AUD_AUREF10	NA	NA	VSS	NA	NA
T	MIPI_DSI1_D3N	MIPI_DSI1_D3P	VSS	USB0_DN	USB0_DP	VSS	AVDD09_DSI1	AVDD12_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	AVSS18_AUD	AVSS18_AUD	NA	VSS	NA	VSS
U	MIPI_DSI1_D2N	MIPI_DSI1_D2P	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1_FB	VSS_FB	VSS	GPIO_123	GPIO_125	NA	NA	GPIO_126	GPIO_127
V	MIPI_DSI1_CLKN	MIPI_DSI1_CLKP	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	GPIO_121	VSS	GPIO_124	GPIO_120	VSS	GPIO_122
W	VSS	VSS	VSS	MIPI_DSI1_D1N	MIPI_DSI1_D1P	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	GPIO3_VCC_CAP	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	GPIO_110	GPIO_117	GPIO_116	VSS	GPIO_119	GPIO_118
Y	PRI_TRST_N	GPIO_74	VSS	MIPI_DSI1_D0N	MIPI_DSI1_D0P	VSS	AVDD33_HDMI	AVDD33_HDMI	AVDD09_HDMI	AVDD09_HDMI	VSS	VSS	VSS	MMC1_VCC_CAP	GPIO2_VCC_CAP	VSS	VSS	VSS	VSS	VSS	VCC18_GPIO	GPIO_26	GPIO_27	VSS	GPIO_28	GPIO_115
AA	PRI_TCK	PRI_TDO	VSS	VSS	VSS	VSS	AVSS_HDMI	HDMI_TX2N	AVDD18_HDMI	AVDD18_HDMI	VSS	VSS	VCC1833_GPIO3	VCC1833_MMC1	VCC1833_GPIO2	MMC1_DAT2	VSS	VSS	GPIO_32	GPIO_29	VCC18_GPIO	GPIO_21	GPIO_24	GPIO_23	GPIO_25	VSS
AB	PRI_TDI	PRI_TMS	VSS	HDMI_TXCN	HDMI_TX0N	AVSS_HDMI	HDMI_TX1N	HDMI_TX2P	AVSS_HDMI	VSS	VSS	VSS	GPIO_51	MMC1_DAT0	GPIO_78	GPIO_77	GPIO_02	GPIO_03	VSS	VSS	GPIO_41	GPIO_44	GPIO_19	VSS	GPIO_20	GPIO_22
AC	GPIO_61	GPIO_62	VCC18_GPIO	HDMI_TXCP	HDMI_TX0P	AVSS_HDMI	HDMI_TX1P	AVSS_HDMI	AVSS_HDMI	GPIO_86	VCC18_GPIO	GPIO_52	GPIO_47	VCC18_GPIO	GPIO_79	VSS	GPIO_05	GPIO_00	VSS	GPIO_31	GPIO_34	GPIO_42	GPIO_43	GPIO_17	VSS	GPIO_18
AD	GPIO_59	GPIO_60	VSS	VSS	VSS	VSS	VSS	GPIO_87	GPIO_85	PMIC_INT_N	VCC18_GPIO	GPIO_50	GPIO_48	MMC1_CMD	GPIO_76	VSS	GPIO_04	GPIO_01	GPIO_30	GPIO_33	VCC18_GPIO	VCC18_GPIO	GPIO_14	GPIO_12	GPIO_16	GPIO_15
AE	VSS	MPLL_TST_AD	VSS	GPIO_92	GPIO_90	GPIO_91	GPIO_89	GPIO_84	GPIO_81	DVL0	PWR_SCL	EXT_32K_IN	VSS	MMC1_CLK	MMC1_DAT3	GPIO_75	GPIO_11	GPIO_07	GPIO_10	GPIO_37	GPIO_35	GPIO_38	GPIO_46	VSS	GPIO_13	VSS
AF	VSS	VSS	RESET_IN_N	JTAG_SEL	VSS	GPIO_88	GPIO_82	GPIO_83	DVL1	VSS	SLEEP_OUT	PWR_SDA	GPIO_49	MMC1_DAT1	VSS	GPIO_80	GPIO_08	GPIO_06	GPIO_09	VSS	GPIO_40	GPIO_36	GPIO_39	GPIO_45	VSS	VSS

Note. Definition of symbols used for pin type:

- AO = Analog output
- AI = Analog input
- AIO = Analog input/output

- G = Ground
- I/O = Input/Output
- P = Power
- RO = Reference output

Pin ID	Name	Type	Power Domain	Function
A14	DQS1_C_B	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Negative of CHB DQS1 LPDDR3: Negtive of DQS1
A15	DQS1_C_A	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Negative of CHA DQS1 LPDDR3: Negtive of DQS0
A16	DQ_A_12	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ12 LPDDR3: DQM0
A17	DQ_A_9	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ9 LPDDR3: DQ7
A18	DQ_A_8	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ8 LPDDR3: DQ5
A19	DQ_A_15	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ15 LPDDR3: DQ3
A20	VSSQ_DDR	G	0V	DDR Ground
A21	DQ_A_5	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ5 LPDDR3: DQ21
A22	DQ_A_7	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ7 LPDDR3: DQ17
A23	DMI0_A	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Channel A DM0 LPDDR3: DQ22
A24	DQ_A_1	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ1 LPDDR3: DQ16
A25	VSSQ_DDR	G	0V	DDR Ground
A26	VSS	G	0V	Digital Core Ground
B14	DQS1_T_B	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Positive of CHB DQS1 LPDDR3: Positive of DQS1
B15	DQS1_T_A	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Positive of CHA DQS1 LPDDR3: Positive of DQS0
B16	DQ_A_11	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ11 LPDDR3: DQ4
B17	DQ_A_10	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ10 LPDDR3: DQ6
B18	DMI1_A	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Channel A DM1 LPDDR3: DQ2

Pin ID	Name	Type	Power Domain	Function
B19	DQ_A_14	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ14 LPDDR3: DQ1
B20	DQ_A_13	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ13 LPDDR3: DQ0
B21	DQ_A_4	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ4 LPDDR3: DQ18
B22	DQ_A_6	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ6 LPDDR3: DQ23
B23	VSSQ_DDR	G	0V	DDR Ground
B24	DQ_A_2	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ2 LPDDR3: DQ19
B25	DQ_A_3	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHB DQ3 LPDDR3: DQM2
B26	VSS	G	0V	Digital Core Ground
C14	VSSQ_DDR	G	0V	DDR Ground
C15	VSSQ_DDR	G	0V	DDR Ground
C16	CA_A_4	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA4 LPDDR3: CA3
C17	VSSQ_DDR	G	0V	DDR Ground
C18	CKE1_A	AO	lp3: 1.2V lp4x: 1.1V	LPDDR4X: clock enabling 1 of CHA LPDDR3: clock enabling 1
C19	CA_A_1	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA1 LPDDR3: CA2
C20	CS1_A	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Active-low chip select 1 of CHA LPDDR3: Active-low chip select 1
C21	AVDD06_DDR	P	lp4x: 0.6V lp4: TBD/lp3: TBD	LPDDR4X IO power
C22	DQ_A_0	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA DQ0 LPDDR3: DQ20
C23	VSS	G	0V	Digital Core Ground
C24	EMMC_DS	I/O	1.8V	eMMC data strobe
C25	EMMC_D7	I/O	1.8V	eMMC data7
C26	EMMC_D2	I/O	1.8V	eMMC data2
D14	VSSQ_DDR	G	0V	DDR Ground
D15	AVDD06_DDR	P	lp4x: 0.6V lp4: TBD	LPDDR4X IO power

Pin ID	Name	Type	Power Domain	Function
			lp3: TBD	
D16	CA_A_2	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA2
D17	CK_C_A	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: negative LPDDR differential clock of CHA LPDDR3: N/A
D18	CKE0_A	AO	lp3: 1.2V lp4x: 1.1V	LPDDR4X: clock enabling 0 of CHA LPDDR3: clock enabling 0
D19	CA_A_0	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA0 LPDDR3: CA4
D20	DQS0_T_A	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Positive of CHA DQS0 LPDDR3: Positive of DQS2
D21	DQS0_C_A	AIO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Negative of CHA DQS0 LPDDR3: Negative of DQS2
D22	VSS	G	0V	Digital Core Ground
D23	EMMC_D4	I/O	1.8V	eMMC data4
D24	EMMC_D1	I/O	1.8V	eMMC data1
D25	VSS	G	0V	Digital Core Ground
D26	EMMC_D0	I/O	1.8V	eMMC data0
E14	AVDD18_DDR	P	1.8V	LPDDR PHY PLL 1.8V power
E15	CA_A_5	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA5 LPDDR3: CA1
E16	CS0_A	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: Active-low chip select 0 of CHA LPDDR3: Active-low chip select 0
E17	CK_T_A	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: positive LPDDR differential clock of CHA LPDDR3: N/A
E18	AVDD06_DDR	P	lp4x: 0.6V lp4: TBD lp3: TBD	LPDDR4X IO power
E19	AVDD06_DDR	P	lp4x: 0.6V lp4: TBD lp3: TBD	LPDDR4X IO power
E20	VSSQ_DDR	G	0V	DDR Ground
E21	AVSS_EMMC	G	0V	eMMC Ground
E22	EMMC_D6	I/O	1.8V	eMMC data6
E23	AVSS_EMMC	G	0V	eMMC Ground

Pin ID	Name	Type	Power Domain	Function
E24	EMMC_CLK	I/O	1.8V	eMMC Clock
E25	EMMC_D3	I/O	1.8V	eMMC data3
E26	EMMC_D5	I/O	1.8V	eMMC data5
F14	ZQ_DDR_PHY	AIO	lp3: 1.2V lp4x: 0.6V	DDR ZQ calibration
F15	CA_A_3	AO	lp3: 1.2V lp4x: 0.6V	LPDDR4X: CHA CA3 LPDDR3: CA0
F16	VSSQ_DDR	G	0V	DDR Ground
F17	DDR_LDO_CAP	RO	0.7~0.9V	External LDO output ball; Connect to a 100nF capacitor on PCB board
F18	AVDD06_DDR	P	lp4x: 0.6V lp4: TBD lp3: TBD	LPDDR4X IO power
F19	VSSQ_DDR	G	0V	DDR Ground
F20	AVSS_EMMC	G	0V	eMMC Ground
F21	AVSS_EMMC	G	0V	eMMC Ground
F22	AVSS_EMMC	G	0V	eMMC Ground
F23	QSPI_DAT2	I/O	1.8V/3.3V	QSPI data2
F24	QSPI_DAT1	I/O	1.8V/3.3V	QSPI data1
F25	EMMC_CMD	I/O	1.8V	eMMC command
F26	QSPI_DAT0	I/O	1.8V/3.3V	QSPI data0
G14	DDR_LP23_VREF CA	P	lp3: 0.6V lp4: high-z	CA VREF for lpddr23, LP4/4x Keep the pin NC
G15	AVDD11_DDR	P	lp4x: 1.1V lp4: 1.1V lp3: 1.2V	LPDDR PHY power supply
G16	AVDD06_DDR	P	lp4x: 0.6V lp4: TBD lp3: TBD	LPDDR4X IO power
G17	VSSQ_DDR	G	0V	DDR Ground
G18	AVDD18_EFUSE	P	1.8V	ANAGRP
G19	AVSS_EMMC	G	0V	eMMC Ground
G20	AVSS_EMMC	G	0V	eMMC Ground
G21	AVSS_EMMC	G	0V	eMMC Ground
G22	QSPI_DAT3	I/O	1.8V/3.3V	QSPI data3

Pin ID	Name	Type	Power Domain	Function
G23	QSPI_CLK	I/O	1.8V/3.3V	QSPI CLK
G24	QSPI_CS1	I/O	1.8V/3.3V	QSPI CS
G25	VSS	G	0V	Digital Core Ground
G26	VSS	G	0V	Digital Core Ground
H14	AVSSU_DDR	G	0V	DDR Ground
H15	AVDD18_PHY	P	1.8V	Analog 1.8V power
H16	VSSU_DDR	G	0V	System DDR Ground
H17	VSSU_DDR	G	0V	System DDR Ground
H18	VSSU_EMMC	G	0V	eMMC Ground
H19	AVDD18_EMMC	P	1.8V	eMMC analog power
H20	AVDD09_EMMC	P	0.9V	eMMC digtial power
H21	VCC1833_QSPI	P	1.8V/3.3V	QSPI IO power
H22	PCIEC_TX0P	AO	1.8V	PCIEC TX0LANEP
H23	PCIEC_TX0N	AO	1.8V	PCIEC TX0LANEN
H24	AVSS_PCIEC	G	0V	PCIEC Ground
H25	PCIEC_RX0P	AI	1.8V	PCIEC RX0LANEP
H26	PCIEC_RX0N	AI	1.8V	PCIEC RX0LANEN
J14	AVDDU_PHY	P	0.9V	LPDDR PHY core logical power
J15	AVDDU_PHY	P	0.9V	LPDDR PHY core logical power
J16	VSS	G	0V	Digital Core Ground
J17	VCC_M1	P	0.9V	Digital Core power
J18	VSSU_EMMC	G	0V	eMMC Ground
J19	QSPI_VCC_CAP	RO	1.8V	QSPI 1.8V LDO cap
J20	AVDD09_EMMC	P	0.9V	eMMC digtial power
J21	AVSS_PCIEC	G	0V	PCIEC Ground
J22	PCIEC_REFCLK_P	AIO	1.8V	PCIEC CKLANEP
J23	PCIEC_REFCLK_N	AIO	1.8V	PCIEC CKLANEN
J24	AVSS_PCIEC	G	0V	PCIEC Ground
J25	PCIEC_RX1P	AI	1.8V	PCIEC RX1LANEP
J26	PCIEC_RX1N	AI	1.8V	PCIEC RX1LANEN
K14	AVDD18_PLL	P	1.8	System PLL power supply

Pin ID	Name	Type	Power Domain	Function
K15	VCC_M1	P	0.9V	Digital Core power
K16	VSS	G	0V	Digital core Ground
K17	VCC_M1	P	0.9V	Digital Core power
K18	VSSU_PCIEC	G	0V	PCIEC Ground
K19	VSSU_PCIEC	G	0V	PCIEC Ground
K20	AVDD09_PCIEC	P	0.9V	PCIEC digital power
K21	AVSS_PCIEC	G	0V	PCIEC Ground
K22	PCIEC_TX1P	AO	1.8V	PCIEC TX1LANEP
K23	PCIEC_TX1N	AO	1.8V	PCIEC TX1LANEN
K24	AVSS_PCIEC	G	0V	PCIEC Ground
K25	PCIEB_RX0P	AI	1.8V	PCIEB RX0LANEP
K26	PCIEB_RX0N	AI	1.8V	PCIEB RX0LANEN
L14	VSSU_PLL	G	0V	System PLL Ground
L15	VSS	G	0V	Digital core Ground
L16	VCC_M1	P	0.9V	Digital Core power
L17	VSSU_PCIEC	G	0V	PCIEC Ground
L18	VSSU_PCIEC	G	0V	PCIEC Ground
L19	AVDD18_PCIEC	P	1.8V	PCIEC analog power
L20	AVDD09_PCIEB	P	0.9V	PCIEB digital power
L21	AVDD09_PCIEB	P	0.9V	PCIEB digital power
L22	PCIEB_TX0P	AO	1.8V	PCIEB TX0LANEP
L23	PCIEB_TX0N	AO	1.8V	PCIEB TX0LANEN
L24	AVSS_PCIEB	G	0V	PCIEB Ground
L25	PCIEB_REFCLK_P	AIO	1.8V	PCIEB CKLANEP
L26	PCIEB_REFCLK_N	AIO	1.8V	PCIEB CKLANEN
M14	VSS	G	0V	Digital Core Ground
M15	VCC_M1	P	0.9V	Digital Core power
M16	VSS	G	0V	Digital Core Ground
M17	VSSU_PCIEB	G	0V	PCIEB Ground
M18	VSSU_PCIEB	G	0V	PCIEB Ground
M19	AVDD18_PCIEB	P	1.8V	PCIEB analog power

Pin ID	Name	Type	Power Domain	Function
M20	AVSS_PCIEB	G	0V	PCIEB Ground
M21	AVSS_PCIEB	G	0V	PCIEB Ground
M22	PCIEB_TX1P	AO	1.8V	PCIEB TX1LANEP
M23	PCIEB_TX1N	AO	1.8V	PCIEB TX1LANEN
M24	AVSS_PCIEB	G	0V	PCIEB Ground
M25	PCIEB_RX1P	AI	1.8V	PCIEB RX1LANEP
M26	PCIEB_RX1N	AI	1.8V	PCIEB RX1LANEN
N14	VCC_M1	P	0.9V	Digital Core power
N15	VSS	G	0V	Digital Core Ground
N16	VCC_M1	P	0.9V	Digital Core power
N17	AVSS18_AUD	G	0V	Audio Ground
N18	AVDD3V3_AUD	P	3.3V	3.3V power for earphone driver
N19	AVSS18_AUD	G	0V	Audio Ground
N20	AVSS18_AUD	G	0V	Audio Ground
N21	NA	P	1.8V	NA
N22	NA	P	-1.8V	NA
N23	NA	AO	+/-1.8V	NA
N24	NA	AO	+/-1.8V	NA
N25	NA	AO	3.3V	NA
N26	NA	AO	3.3V	NA

3.2.3 (P~AF, 1~13)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	VSS	VSSQ_DDR	DQ_B_2	DMI0_B	VSSQ_DDR	DQ_B_6	DQ_B_4	DQ_B_13	DQ_B_15	VSSQ_DDR	DQ_B_9	DQ_B_12	DQ_B_11	DQS1_C_B	DQS1_C_A	DQ_A_12	DQ_A_9	DQ_A_8	DQ_A_15	VSSQ_DDR	DQ_A_5	DQ_A_7	DMI0_A	DQ_A_1	VSSQ_DDR	VSS
B	VSS	DQ_B_3	VSSQ_DDR	DQ_B_1	DQ_B_0	DQ_B_7	DQ_B_5	VDDQ_V1P2	DQ_B_14	DMI1_B	DQ_B_8	DQ_B_10	VSSQ_DDR	DQS1_T_B	DQS1_T_A	DQ_A_11	DQ_A_10	DMI1_A	DQ_A_14	DQ_A_13	DQ_A_4	DQ_A_6	VSSQ_DDR	DQ_A_2	DQ_A_3	VSS
C	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	DQS0_T_B	VSSQ_DDR	CS1_B	CA_B_1	CKE0_B	CKE1_B	VDDQ_V1P2	CA_B_5	VSSQ_DDR	VSSQ_DDR	CA_A_4	VSSQ_DDR	CKE1_A	CA_A_1	CS1_A	AVDD06_DDR	DQ_A_0	VSS	EMMC_DS	EMMC_D7	EMMC_D2
D	GPIO_114	GPIO_113	GPIO_112	GPIO_111	GPIO_53	DQS0_C_B	VSSQ_DDR	CA_B_0	VSSQ_DDR	DDR_L P4X_SEL	CK_C_B	CA_B_2	CA_B_4	VSSQ_DDR	AVDD06_DDR	CA_A_2	CK_C_A	CKE0_A	CA_A_0	DQS0_T_A	DQS0_C_A	VSS	EMMC_D4	EMMC_D1	VSS	EMMC_D0
E	GPIO_67	GPIO_65	GPIO_64	VSS	GPIO_63	VSS	VSSQ_DDR	VDDQ_V1P2	DDR_L P23_VREFDQ	VSSQ_DDR	CK_T_B	CA_B_3	AVSS18_DDR	AVDD18_DDR	CA_A_5	CS0_A	CK_T_A	AVDD06_DDR	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	EMMC_D6	AVSS_EMMC	EMMC_CLK	EMMC_D3	EMMC_D5
F	VSS	VSS	GPIO_69	GPIO_68	GPIO_66	VCC18_GPIO	VSS	VSSQ_DDR	VDDQ_V1P2	VDDQ_V1P2	VSSQ_DDR	CS0_B	DDR_RESET_N	ZQ_DDR_PHY	CA_A_3	VSSQ_DDR	DDR_L DO_CAP	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT2	QSPI_DAT1	EMMC_CMD	QSPI_DAT0
G	MIPI_CSI1_D1N	MIPI_CSI1_D1P	VSS	MIPI_CSI1_D0N	MIPI_CSI1_D0P	VSS	VSS	VSS	VSS	VSSQ_DDR	VDDQ_V1P2	AVDD11_DDR	VDDQ_V1P2	DDR_L P23_VREFCA	AVDD11_DDR	AVDD06_DDR	VSSQ_DDR	AVDD18_EFUSE	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT3	QSPI_CLK	QSPI_CS1	VSS	VSS
H	MIPI_CSI1_D2N	MIPI_CSI1_D2P	VSS	MIPI_CSI1_CLKN	MIPI_CSI1_CLKP	AVSS18_AFEAP	XI_PAD	AVSS18_AFEAP	VSS	VSSU_DDR	VSSU_DDR	AVDD18_PHY	AVDDU_DDR	AVSSU_DDR	AVDD18_PHY	VSSU_DDR	VSSU_DDR	VSSU_EMMC	AVDD18_EMMC	AVDD09_EMMC	VCC1833_QSPI	PCIEC_TX0P	PCIEC_TX0N	AVSS_PCIEC	PCIEC_RX0P	PCIEC_RX0N
J	MIPI_CSI3_D0N	MIPI_CSI3_D0P	AVSS_CSI	MIPI_CSI1_D3N	MIPI_CSI1_D3P	AVSS_CSI	XO_PAD	AVSS18_AFEAP	AVSS18_AFEAP	VCC_M1	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	VSS	VCC_M1	VSSU_EMMC	QSPI_VCC_CAP	AVDD09_EMMC	AVSS_PCIEC	PCIEC_REFCLK_P	PCIEC_REFCLK_N	AVSS_PCIEC	PCIEC_RX1P	PCIEC_RX1N
K	MIPI_CSI3_CLKN	MIPI_CSI3_CLKP	AVSS_CSI	MIPI_CSI3_D1N	MIPI_CSI3_D1P	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	VCC_M1	VSS	BG_OUT	AVDD18_AFEAP	MPLL_TST_CK	AVDD18_PLL	VCC_M1	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD09_PCIEC	AVSS_PCIEC	PCIEC_TX1P	PCIEC_TX1N	AVSS_PCIEC	PCIEB_RX0P	PCIEB_RX0N
L	MIPI_CSI3_D2N	MIPI_CSI3_D2P	AVSS_CSI	MIPI_CSI2_CLKN	MIPI_CSI2_CLKP	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	AVSS_CSI	VCC_M1	AVDD09_AFEAP	VSSU_AFEAP	AVSS_PLL	VSSU_PLL	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD18_PCIEC	AVDD09_PCIEB	AVDD09_PCIEB	PCIEB_TX0P	PCIEB_TX0N	AVSS_PCIEB	PCIEB_REFCLK_P	PCIEB_REFCLK_N
M	MIPI_CSI3_D3N	MIPI_CSI3_D3P	VSS	VSS	VSSU_PCIEA	AVDD18_USB	AVDD09_USB	VSSU_PCIEA	AVDD33_USB	VSS	AVDD09_PLL	VSS	VSS	VSS	VCC_M1	VSS	VSSU_PCIEB	VSSU_PCIEB	AVDD18_PCIEB	AVSS_PCIEB	AVSS_PCIEB	PCIEB_TX1P	PCIEB_TX1N	AVSS_PCIEB	PCIEB_RX1P	PCIEB_RX1N
N	USB2_DN	USB2_DP	AVSS_USB	PCIEA_TXN	PCIEA_TXP	AVDD18_PCIEA	AVDD09_PCIEA	AVSS_PCIEA	AVDD33_USB	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AVSS18_AUD	AVDD33V3_AUD	AVSS18_AUD	AVSS18_AUD	NA	NA	NA	NA	NA	NA
P	PCIEA_RXN	PCIEA_RXP	AVSS_USB	PCIEA_R_EXT	AVSS_USB	AVDD18_USB	AVDD09_USB	AVDD09_USB	AVDD33_USB	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	AUD_GNDSNS	AVDD18_AUD	AVDD18_AUD	NA	NA	NA	NA	NA	NA	NA
R	PCIEA_REFCLK_N	PCIEA_REFCLK_P	VSS	USB1_DN	USB1_DP	AVDD18_DSI1	AVSS_USB	VSS	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AUD_VSSU	AUD_VDDU09	AUD_REFGND	NA	AUD_AUREF10	NA	NA	VSS	NA	NA
T	MIPI_DSI1_D3N	MIPI_DSI1_D3P	VSS	USB0_DN	USB0_DP	VSS	AVDD09_DSI1	AVDD12_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	AVSS18_AUD	AVSS18_AUD	NA	VSS	NA	VSS
U	MIPI_DSI1_D2N	MIPI_DSI1_D2P	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1_FB	VSS_FB	VSS	GPIO_123	GPIO_125	NA	NA	GPIO_126	GPIO_127
V	MIPI_DSI1_CLKN	MIPI_DSI1_CLKP	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	GPIO_121	VSS	GPIO_124	GPIO_120	VSS	GPIO_122
W	VSS	VSS	VSS	MIPI_DSI1_D1N	MIPI_DSI1_D1P	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	GPIO3_VCC_CAP	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	GPIO_110	GPIO_117	GPIO_116	VSS	GPIO_119	GPIO_118
Y	PRI_TST_N	GPIO_74	VSS	MIPI_DSI1_D0N	MIPI_DSI1_D0P	VSS	AVDD33_HDMI	AVDD33_HDMI	AVDD09_HDMI	AVDD09_HDMI	VSS	VSS	VSS	MMC1_VCC_CAP	GPIO2_VCC_CAP	VSS	VSS	VSS	VSS	VSS	VCC18_GPIO	GPIO_26	GPIO_27	VSS	GPIO_28	GPIO_115
AA	PRI_TCK	PRI_TDO	VSS	VSS	VSS	VSS	AVSS_HDMI	HDMI_TX2N	AVDD18_HDMI	AVDD18_HDMI	VSS	VSS	VCC1833_GPIO3	VCC1833_MMC1	VCC1833_GPIO2	MMC1_DAT2	VSS	VSS	GPIO_32	GPIO_29	VCC18_GPIO	GPIO_21	GPIO_24	GPIO_23	GPIO_25	VSS
AB	PRI_TDI	PRI_TMS	VSS	HDMI_TXCN	HDMI_TX0N	AVSS_HDMI	HDMI_TX1N	HDMI_TX2P	AVSS_HDMI	VSS	VSS	VSS	GPIO_51	MMC1_DAT0	GPIO_78	GPIO_77	GPIO_02	GPIO_03	VSS	VSS	GPIO_41	GPIO_44	GPIO_19	VSS	GPIO_20	GPIO_22
AC	GPIO_61	GPIO_62	VCC18_GPIO	HDMI_TXCP	HDMI_TX0P	AVSS_HDMI	HDMI_TX1P	AVSS_HDMI	AVSS_HDMI	GPIO_86	VCC18_GPIO	GPIO_52	GPIO_47	VCC18_GPIO	GPIO_79	VSS	GPIO_05	GPIO_00	VSS	GPIO_31	GPIO_34	GPIO_42	GPIO_43	GPIO_17	VSS	GPIO_18
AD	GPIO_59	GPIO_60	VSS	VSS	VSS	VSS	VSS	GPIO_87	GPIO_85	PMIC_INT_N	VCC18_GPIO	GPIO_50	GPIO_48	MMC1_CMD	GPIO_76	VSS	GPIO_04	GPIO_01	GPIO_30	GPIO_33	VCC18_GPIO	VCC18_GPIO	GPIO_14	GPIO_12	GPIO_16	GPIO_15
AE	VSS	MPLL_TST_AD	VSS	GPIO_92	GPIO_90	GPIO_91	GPIO_89	GPIO_84	GPIO_81	DVL0	PWR_SCL	EXT_32K_IN	VSS	MMC1_CLK	MMC1_DAT3	GPIO_75	GPIO_11	GPIO_07	GPIO_10	GPIO_37	GPIO_35	GPIO_38	GPIO_46	VSS	GPIO_13	VSS
AF	VSS	VSS	RESET_IN_N	JTAG_SEL	VSS	GPIO_88	GPIO_82	GPIO_83	DVL1	VSS	SLEEP_OUT	PWR_SDA	GPIO_49	MMC1_DAT1	VSS	GPIO_80	GPIO_08	GPIO_06	GPIO_09	VSS	GPIO_40	GPIO_36	GPIO_39	GPIO_45	VSS	VSS

Note. Definition of symbols used for pin type:

- AO = Analog output
- AI = Analog input
- AIO = Analog input/output
- G = Ground
- I/O = Input/Output
- P = Power
- RO = Reference output

Pin ID	Name	Type	Power Domain	Function
P1	PCIEA_RXN	AI	1.8V	PCIEA RXLANEN
P2	PCIEA_RXP	AI	1.8V	PCIEA RXLANEP

Pin ID	Name	Type	Power Domain	Function
P3	AVSS_USB	G	0V	USB2.0 Ground
P4	PCIEA_R_EXT	AO	1.8V	PCIEA External calibration resistor
P5	AVSS_USB	G	0V	USB2.0 Ground
P6	AVDD18_USB	P	1.8V	USB2.0 1.8V power
P7	AVDD09_USB	P	0.9V	USB2.0 digital power
P8	AVDD09_USB	P	0.9V	USB2.0 digital power
P9	AVDD33_USB	P	3.3V	USB2.0 3.3V power
P10	VSS	G	0V	Digital Core Ground
P11	VCC_M1	P	0.9V	Digital Core power
P12	VSS	G	0V	Digital Core Ground
P13	VCC_M1	P	0.9V	Digital Core power
R1	PCIEA_REFCLK_N	AIO	1.8V	PCIEA CKLANEN
R2	PCIEA_REFCLK_P	AIO	1.8V	PCIEA CKLANEP
R3	VSS	G	0V	Digital core Ground
R4	USB1_DN	AIO	3.3V	USB2.0_1 D- differential data line
R5	USB1_DP	AIO	3.3V	USB2.0_1 D+ differential data line
R6	AVDD18_DSI1	P	1.8V	DSI analog power
R7	AVSS_USB	G	0V	USB2.0 Ground
R8	VSS	G	0V	Digital Core Ground
R9	VSS	G	0V	Digital Core Ground
R10	VCC_M1	P	0.9V	Digital Core power
R11	VSS	G	0V	Digital Core Ground
R12	VCC_M1	P	0.9V	Digital Core power
R13	VSS	G	0V	Digital Core Ground
T1	MIPI_DSI1_D3N	AO	1.2V	DSI DATA3LANEN
T2	MIPI_DSI1_D3P	AO	1.2V	DSI DATA3LANEP
T3	VSS	G	0V	Digital core ground
T4	USB0_DN	AIO	3.3V	USB2.0_0 D- differential data line
T5	USB0_DP	AIO	3.3V	USB2.0_0 D+ differential data line
T6	VSS	G	0V	Digital core ground
T7	AVDD09_DSI1	P	0.9V	DSI digital power
T8	AVDD12_DSI1	P	1.2V	DSI driver power

Pin ID	Name	Type	Power Domain	Function
T9	VCC_M1	P	0.9V	Digital Core power
T10	VSS	G	0V	Digital Core ground
T11	VCC_M1	P	0.9V	Digital Core power
T12	VSS	G	0V	Digital Core ground
T13	VCC_M1	P	0.9V	Digital Core power
U1	MIPI_DSI1_D2N	AO	1.2V	DSI DATA2LANEN
U2	MIPI_DSI1_D2P	AO	1.2V	DSI DATA2LANEP
U3	AVSS_DSI1	G	0V	DSI Ground
U4	AVSS_DSI1	G	0V	DSI Ground
U5	AVSS_DSI1	G	0V	DSI Ground
U6	VCC_M1	P	0.9V	Digital Core power
U7	AVSS_DSI1	G	0V	DSI Ground
U8	VCC_M1	P	0.9V	Digital Core power
U9	VSS	G	0V	Digital Core ground
U10	VCC_M1	P	0.9V	Digital Core power
U11	VSS	G	0V	Digital Core ground
U12	VCC_M1	P	0.9V	Digital Core power
U13	VSS	G	0V	Digital Core ground
V1	MIPI_DSI1_CLKN	AO	1.2V	DSI CKLANEN
V2	MIPI_DSI1_CLKP	AO	1.2V	DSI CKLANEP
V3	AVSS_DSI1	G	0V	DSI Ground
V4	AVSS_DSI1	G	0V	DSI Ground
V5	AVSS_DSI1	G	0V	DSI Ground
V6	AVSS_DSI1	G	0V	DSI Ground
V7	VCC_M1	P	0.9V	Digital Core power
V8	VSS	G	0V	Digital Core ground
V9	VCC_M1	P	0.9V	Digital Core power
V10	VSS	G	0V	Digital Core ground
V11	VCC_M1	P	0.9V	Digital Core power
V12	VSS	G	0V	Digital Core ground
V13	VCC_M1	P	0.9V	Digital Core power
W1	VSS	G	0V	Digital Core ground

Pin ID	Name	Type	Power Domain	Function
W2	VSS	G	0V	Digital Core ground
W3	VSS	G	0V	Digital Core ground
W4	MIPI_DSI1_D1N	AO	1.2V	DSI DATA1LANEN
W5	MIPI_DSI1_D1P	AO	1.2V	DSI DATA1LANEP
W6	VCC_M1	P	0.9V	Digital Core power
W7	VSS	G	0V	Digital Core ground
W8	VCC_M1	P	0.9V	Digital Core power
W9	VSS	G	0V	Digital Core ground
W10	VCC_M1	P	0.9V	Digital Core power
W11	VSS	G	0V	Digital Core ground
W12	VCC_M1	P	0.9V	Digital Core power
W13	GPIO3_VCC_CAP	RO	1.8V	GPIO3 1.8V LDO cap
Y1	PRI_TRST_N	I/O	1.8V	JTAG reset
Y2	GPIO_74	I/O	1.8V	General Purpose I/O 74
Y3	VSS	G	0V	Digital Core ground
Y4	MIPI_DSI1_D0N	AO	1.2V	DSI DATA0LANEN
Y5	MIPI_DSI1_D0P	AO	1.2V	DSI DATA0LANEP
Y6	VSS	G	0V	Digital Core ground
Y7	AVDD33_HDMI	P	3.3V	HDMI 3.3V power
Y8	AVDD33_HDMI	P	3.3V	HDMI 3.3V power
Y9	AVDD09_HDMI	P	0.9V	HDMI digtial power
Y10	AVDD09_HDMI	P	0.9V	HDMI digtial power
Y11	VSS	G	0V	Digital Core ground
Y12	VSS	G	0V	Digital Core ground
Y13	VSS	G	0V	Digital Core ground
AA1	PRI_TCK	I/O	1.8V	JTAG clock
AA2	PRI_TDO	I/O	1.8V	JTAG output data
AA3	VSS	G	0V	Digital Core ground
AA4	VSS	G	0V	Digital Core ground
AA5	VSS	G	0V	Digital Core ground
AA6	VSS	G	0V	Digital Core ground
AA7	AVSS_HDMI	G	0V	HDMI Ground

Pin ID	Name	Type	Power Domain	Function
AA8	HDMI_TX2N	AO	1.8V	HDMI data2n
AA9	AVDD18_HDMI	P	1.8V	HDMI 1.8V power
AA10	AVDD18_HDMI	P	1.8V	HDMI 1.8V power
AA11	VSS	G	0V	Digital Core ground
AA12	VSS	G	0V	Digital Core ground
AA13	VCC1833_GPIO3	P	1.8V/3.3V	GPIO3 IO power
AB1	PRI_TDI	I/O	1.8V	JTAG input data
AB2	PRI_TMS	I/O	1.8V	JTAG mode selection
AB3	VSS	G	0V	Digital Core ground
AB4	HDMI_TXCN	AO	1.8V	HDMI clkn
AB5	HDMI_TX0N	AO	1.8V	HDMI data0n
AB6	AVSS_HDMI	G	0V	HDMI Ground
AB7	HDMI_TX1N	AO	1.8V	HDMI data1n
AB8	HDMI_TX2P	AO	1.8V	HDMI data2p
AB9	AVSS_HDMI	G	0V	HDMI Ground
AB10	VSS	G	0V	Digital Core ground
AB11	VSS	G	0V	Digital Core ground
AB12	VSS	G	0V	Digital Core ground
AB13	GPIO_51	I/O	1.8V/3.3V	General purpose I/O 51
AC1	GPIO_61	I/O	1.8V	General Purpose I/O 61
AC2	GPIO_62	I/O	1.8V	General Purpose I/O 62
AC3	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AC4	HDMI_TXCP	AO	1.8V	HDMI clkp
AC5	HDMI_TX0P	AO	1.8V	HDMI data0p
AC6	AVSS_HDMI	G	0V	HDMI Ground
AC7	HDMI_TX1P	AO	1.8V	HDMI data1p
AC8	AVSS_HDMI	G	0V	HDMI Ground
AC9	AVSS_HDMI	G	0V	HDMI Ground
AC10	GPIO_86	I/O	1.8V	General Purpose I/O 86
AC11	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AC12	GPIO_52	I/O	1.8V/3.3V	General Purpose I/O 52
AC13	GPIO_47	I/O	1.8V/3.3V	General Purpose I/O 47

Pin ID	Name	Type	Power Domain	Function
AD1	GPIO_59	I/O	1.8V	General Purpose I/O 59
AD2	GPIO_60	I/O	1.8V	General Purpose I/O 60
AD3	VSS	G	0V	Digital Core ground
AD4	VSS	G	0V	Digital Core ground
AD5	VSS	G	0V	Digital Core ground
AD6	VSS	G	0V	Digital Core ground
AD7	VSS	G	0V	Digital Core ground
AD8	GPIO_87	I/O	1.8V	General Purpose I/O 87
AD9	GPIO_85	I/O	1.8V	General Purpose I/O 85
AD10	PMIC_INT_N	I/O	1.8V	PMIC interrupt
AD11	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AD12	GPIO_50	I/O	1.8V/3.3V	General Purpose I/O 50
AD13	GPIO_48	I/O	1.8V/3.3V	General Purpose I/O 48
AE1	VSS	G	0V	Digital Core ground
AE2	MPLL_TST_AD	AIO	1.8V	Analog testpin
AE3	VSS	G	0V	Digital Core ground
AE4	GPIO_92	I/O	1.8V	General Purpose I/O 92
AE5	GPIO_90	I/O	1.8V	General Purpose I/O 90
AE6	GPIO_91	I/O	1.8V	General Purpose I/O 91
AE7	GPIO_89	I/O	1.8V	General Purpose I/O 89
AE8	GPIO_84	I/O	1.8V	General Purpose I/O 84
AE9	GPIO_81	I/O	1.8V	General Purpose I/O 81
AE10	DVL0	I/O	1.8V	Hardware dynamic voltage regulation signal0
AE11	PWR_SCL	I/O	1.8V	PMIC I2C bus clock
AE12	EXT_32K_IN	I/O	1.8V	32K clock input
AE13	VSS	G	0V	Digital Core ground
AF1	VSS	G	0V	Digital Core ground
AF2	VSS	G	0V	Digital Core ground
AF3	RESET_IN_N	I/O	1.8V	Reset input
AF4	JTAG_SEL	I/O	1.8V	Primary JTAG selection
AF5	VSS	G	0V	Digital Core ground
AF6	GPIO_88	I/O	1.8V	General Purpose I/O 88

Pin ID	Name	Type	Power Domain	Function
AF7	GPIO_82	I/O	1.8V	General Purpose I/O 82
AF8	GPIO_83	I/O	1.8V	General Purpose I/O 83
AF9	DVL1	I/O	1.8V	Hardware dynamic voltage regulation signal1
AF10	VSS	G	0V	Digital Core ground
AF11	SLEEP_OUT	I/O	1.8V	VCXO enabling
AF12	PWR_SDA	I/O	1.8V	PMIC I2C bus data/address
AF13	GPIO_49	I/O	1.8V/3.3V	General Purpose I/O 49

3.2.4 (P~AF, 14~26)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	VSS	VSSQ_DDR	DQ_B_2	DMI0_B	VSSQ_DDR	DQ_B_6	DQ_B_4	DQ_B_13	DQ_B_15	VSSQ_DDR	DQ_B_9	DQ_B_12	DQ_B_11	DQS1_C_B	DQS1_C_A	DQ_A_12	DQ_A_9	DQ_A_8	DQ_A_15	VSSQ_DDR	DQ_A_5	DQ_A_7	DMI0_A	DQ_A_1	VSSQ_DDR	VSS
B	VSS	DQ_B_3	VSSQ_DDR	DQ_B_1	DQ_B_0	DQ_B_7	DQ_B_5	VDDQ_V1P2	DQ_B_14	DMI1_B	DQ_B_8	DQ_B_10	VSSQ_DDR	DQS1_T_B	DQS1_T_A	DQ_A_11	DQ_A_10	DMI1_A	DQ_A_14	DQ_A_13	DQ_A_4	DQ_A_6	VSSQ_DDR	DQ_A_2	DQ_A_3	VSS
C	GPIO_58	GPIO_57	GPIO_56	GPIO_55	GPIO_54	DQS0_T_B	VSSQ_DDR	CS1_B	CA_B_1	CKE0_B	CKE1_B	VDDQ_V1P2	CA_B_5	VSSQ_DDR	VSSQ_DDR	CA_A_4	VSSQ_DDR	CKE1_A	CA_A_1	CS1_A	AVDD06_DDR	DQ_A_0	VSS	EMMC_DS	EMMC_D7	EMMC_D2
D	GPIO_114	GPIO_113	GPIO_112	GPIO_111	GPIO_53	DQS0_C_B	VSSQ_DDR	CA_B_0	VSSQ_DDR	DDR_LP4X_SEL	CK_C_B	CA_B_2	CA_B_4	VSSQ_DDR	AVDD06_DDR	CA_A_2	CK_C_A	CKE0_A	CA_A_0	DQS0_T_A	DQS0_C_A	VSS	EMMC_D4	EMMC_D1	VSS	EMMC_D0
E	GPIO_67	GPIO_65	GPIO_64	VSS	GPIO_63	VSS	VSSQ_DDR	VDDQ_V1P2	DDR_LP23_VREFDQ	VSSQ_DDR	CK_T_B	CA_B_3	AVSS18_DDR	AVDD18_DDR	CA_A_5	CS0_A	CK_T_A	AVDD06_DDR	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	EMMC_D6	AVSS_EMMC	EMMC_CLK	EMMC_D3	EMMC_D5
F	VSS	VSS	GPIO_69	GPIO_68	GPIO_66	VCC18_GPIO	VSS	VSSQ_DDR	VDDQ_V1P2	VDDQ_V1P2	VSSQ_DDR	CS0_B	DDR_RESET_N	ZQ_DDR_PHY	CA_A_3	VSSQ_DDR	DDR_LDO_CAP	AVDD06_DDR	VSSQ_DDR	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT1	QSPI_DAT2	EMMC_CMD	QSPI_DAT0
G	MIPI_CSI1_D1N	MIPI_CSI1_D1P	VSS	MIPI_CSI1_D0N	MIPI_CSI1_D0P	VSS	VSS	VSS	VSS	VSSQ_DDR	VDDQ_V1P2	AVDD11_DDR	VDDQ_V1P2	DDR_LP23_VREFCA	AVDD11_DDR	AVDD06_DDR	VSSQ_DDR	AVDD18_EFUSE	AVSS_EMMC	AVSS_EMMC	AVSS_EMMC	QSPI_DAT3	QSPI_CLK	QSPI_CS1	VSS	VSS
H	MIPI_CSI1_D2N	MIPI_CSI1_D2P	VSS	MIPI_CSI1_CLKN	MIPI_CSI1_CLKP	AVSS18_AFEAP	XI_PAD	AVSS18_AFEAP	VSS	VSSU_DDR	VSSU_DDR	AVDD18_PHY	AVDDU_DDR	AVSSU_DDR	AVDD18_PHY	VSSU_DDR	VSSU_DDR	VSSU_EMMC	AVDD18_EMMC	AVDD09_EMMC	VCC1833_QSPI	PCIEC_TX0P	PCIEC_TX0N	AVSS_PCIEC	PCIEC_RX0P	PCIEC_RX0N
J	MIPI_CSI3_D0N	MIPI_CSI3_D0P	AVSS_CSI	MIPI_CSI1_D3N	MIPI_CSI1_D3P	AVSS_CSI	XO_PAD	AVSS18_AFEAP	AVSS18_AFEAP	VCC_M1	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	AVDDU_PHY	VSS	VCC_M1	VSSU_EMMC	QSPI_VCC_CAP	AVDD09_EMMC	AVSS_PCIEC	PCIEC_REFCLK_P	PCIEC_REFCLK_N	AVSS_PCIEC	PCIEC_RX1P	PCIEC_RX1N
K	MIPI_CSI3_CLKN	MIPI_CSI3_CLKP	AVSS_CSI	MIPI_CSI3_D1N	MIPI_CSI3_D1P	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	VCC_M1	VSS	BG_OUT	AVDD18_AFEAP	MPLL_TST_CK	AVDD18_PLL	VCC_M1	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD09_PCIEC	AVSS_PCIEC	PCIEC_TX1P	PCIEC_TX1N	AVSS_PCIEC	PCIEB_RX0P	PCIEB_RX0N
L	MIPI_CSI3_D2N	MIPI_CSI3_D2P	AVSS_CSI	MIPI_CSI2_CLKN	MIPI_CSI2_CLKP	AVDD18_CSI	AVDD09_CSI	AVSS_CSI	AVSS_CSI	VCC_M1	AVDD09_AFEAP	VSSU_AFEAP	AVSS_PLL	VSSU_PLL	VSS	VCC_M1	VSSU_PCIEC	VSSU_PCIEC	AVDD18_PCIEC	AVDD09_PCIEB	AVDD09_PCIEB	PCIEB_TX0P	PCIEB_TX0N	AVSS_PCIEB	PCIEB_REFCLK_P	PCIEB_REFCLK_N
M	MIPI_CSI3_D3N	MIPI_CSI3_D3P	VSS	VSS	VSSU_PCIEA	AVDD18_USB	AVDD09_USB	VSSU_PCIEA	AVDD33_USB	VSS	AVDD09_PLL	VSS	VSS	VSS	VCC_M1	VSS	VSSU_PCIEB	VSSU_PCIEB	AVDD18_PCIEB	AVSS_PCIEB	AVSS_PCIEB	PCIEB_TX1P	PCIEB_TX1N	AVSS_PCIEB	PCIEB_RX1P	PCIEB_RX1N
N	USB2_DN	USB2_DP	AVSS_USB	PCIEA_TXN	PCIEA_TXP	AVDD18_PCIEA	AVDD09_PCIEA	AVSS_PCIEA	AVDD33_USB	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AVSS18_AUD	AVDD33V3_AUD	AVSS18_AUD	AVSS18_AUD	NA	NA	NA	NA	NA	NA
P	PCIEA_RXN	PCIEA_RXP	AVSS_USB	PCIEA_R_EXT	AVSS_USB	AVDD18_USB	AVDD09_USB	AVDD09_USB	AVDD33_USB	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	AUD_GNDSNS	AVDD18_AUD	AVDD18_AUD	NA	NA	NA	NA	NA	NA	NA
R	PCIEA_REFCLK_N	PCIEA_REFCLK_P	VSS	USB1_DN	USB1_DP	AVDD18_DSI1	AVSS_USB	VSS	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	AUD_VSSU	AUD_VDDU09	AUD_REFGND	NA	AUD_AUREF10	NA	NA	VSS	NA	NA
T	MIPI_DSI1_D3N	MIPI_DSI1_D3P	VSS	USB0_DN	USB0_DP	VSS	AVDD09_DSI1	AVDD12_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	AVSS18_AUD	AVSS18_AUD	NA	VSS	NA	VSS
U	MIPI_DSI1_D2N	MIPI_DSI1_D2P	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1_FB	VSS_FB	VSS	GPIO_123	GPIO_125	NA	NA	GPIO_126	GPIO_127
V	MIPI_DSI1_CLKN	MIPI_DSI1_CLKP	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	AVSS_DSI1	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	VSS	GPIO_121	VSS	GPIO_124	GPIO_120	VSS	GPIO_122
W	VSS	VSS	VSS	MIPI_DSI1_D1N	MIPI_DSI1_D1P	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	GPIO3_VCC_CAP	VCC_M1	VSS	VCC_M1	VSS	VCC_M1	VSS	VSS	GPIO_110	GPIO_117	GPIO_116	VSS	GPIO_119	GPIO_118
Y	PRI_TST_N	GPIO_74	VSS	MIPI_DSI1_D0N	MIPI_DSI1_D0P	VSS	AVDD33_HDMI	AVDD33_HDMI	AVDD09_HDMI	AVDD09_HDMI	VSS	VSS	VSS	MMC1_VCC_CAP	GPIO2_VCC_CAP	VSS	VSS	VSS	VSS	VSS	VCC18_GPIO	GPIO_26	GPIO_27	VSS	GPIO_28	GPIO_115
AA	PRI_TCK	PRI_TDO	VSS	VSS	VSS	VSS	AVSS_HDMI	HDMI_TX2N	AVDD18_HDMI	AVDD18_HDMI	VSS	VSS	VCC1833_GPIO3	VCC1833_MMC1	VCC1833_GPIO2	MMC1_DAT2	VSS	VSS	GPIO_32	GPIO_29	VCC18_GPIO	GPIO_21	GPIO_24	GPIO_23	GPIO_25	VSS
AB	PRI_TDI	PRI_TMS	VSS	HDMI_TXCN	HDMI_TX0N	AVSS_HDMI	HDMI_TX1N	HDMI_TX2P	AVSS_HDMI	VSS	VSS	VSS	GPIO_51	MMC1_DAT0	GPIO_78	GPIO_77	GPIO_02	GPIO_03	VSS	VSS	GPIO_411	GPIO_44	GPIO_19	VSS	GPIO_20	GPIO_22
AC	GPIO_61	GPIO_62	VCC18_GPIO	HDMI_TXCP	HDMI_TX0P	AVSS_HDMI	HDMI_TX1P	AVSS_HDMI	AVSS_HDMI	GPIO_86	VCC18_GPIO	GPIO_52	GPIO_47	VCC18_GPIO	GPIO_79	VSS	GPIO_05	GPIO_00	VSS	GPIO_31	GPIO_34	GPIO_42	GPIO_43	GPIO_17	VSS	GPIO_18
AD	GPIO_59	GPIO_60	VSS	VSS	VSS	VSS	VSS	GPIO_87	GPIO_85	PMIC_INT_N	VCC18_GPIO	GPIO_50	GPIO_48	MMC1_CMD	GPIO_76	VSS	GPIO_04	GPIO_01	GPIO_30	GPIO_33	VCC18_GPIO	VCC18_GPIO	GPIO_14	GPIO_12	GPIO_16	GPIO_15
AE	VSS	MPLL_TST_AD	VSS	GPIO_92	GPIO_90	GPIO_91	GPIO_89	GPIO_84	GPIO_81	DVL0	PWR_SCL	EXT_32K_IN	VSS	MMC1_CLK	MMC1_DAT3	GPIO_75	GPIO_11	GPIO_07	GPIO_10	GPIO_37	GPIO_35	GPIO_38	GPIO_46	VSS	GPIO_13	VSS
AF	VSS	VSS	RESET_IN_N	JTAG_SEL	VSS	GPIO_88	GPIO_82	GPIO_83	DVL1	VSS	SLEEP_OUT	PWR_SDA	GPIO_49	MMC1_DAT1	VSS	GPIO_80	GPIO_08	GPIO_06	GPIO_09	VSS	GPIO_40	GPIO_36	GPIO_39	GPIO_45	VSS	VSS

Note. Definition of symbols used for pin type:

- AO = Analog output
- AI = Analog input
- AIO = Analog input/output
- G = Ground
- I/O = Input/Output
- P = Power
- RO = Reference output

Pin ID	Name	Type	Power Domain	Function
P14	VSS	G	0V	Digital Core Ground
P15	VCC_M1	P	0.9V	Digital Core power
P16	VSS	G	0V	Digital Core Ground
P17	AUD_GNDSNS	G	0V	Headphone sense_Ground
P18	AVDD18_AUD	P	1.8V	1.8V power for audio
P19	AVDD18_AUD	P	1.8V	1.8V power for audio
P20	NA	AO	1.8V	NA
P21	NA	AO	1.8V	NA
P22	NA	AO	1.8V	NA
P23	NA	AI	1.8V	NA
P24	NA	AI	1.8V	NA
P25	NA	AI	1.8V	NA
P26	NA	AI	1.8V	NA
R14	VCC_M1	P	0.9V	Digital Core power
R15	VSS	G	0V	Digital Core Ground
R16	VCC_M1	P	0.9V	Digital Core power
R17	AUD_VSSU	G	0V	Audio Ground
R18	AUD_VDDU09	P	0.9V	0.9V power for audio
R19	AUD_REFGND	G	0V	Audio Reference Ground
R20	NA	AO	1.8V	NA
R21	AUD_AUREF10	RO	1.8V	Audio reference voltage
R22	NA	AI	1.8V	NA
R23	NA	AI	1.8V	NA
R24	VSS	G	0V	Digital core ground
R25	NA	AI	1.8V	NA
R26	NA	AI	1.8V	NA

Pin ID	Name	Type	Power Domain	Function
T14	VSS	G	0V	Digital Core ground
T15	VCC_M1	P	0.9V	Digital Core power
T16	VSS	G	0V	Digital Core ground
T17	VCC_M1	P	0.9V	Digital Core power
T18	VSS	G	0V	Digital Core ground
T19	VSS	G	0V	Digital Core ground
T20	VSS	G	0V	Digital Core ground
T21	AVSS18_AUD	G	0V	Audio Ground
T22	AVSS18_AUD	G	0V	Audio Ground
T23	NA	AI	1.8V	NA
T24	VSS	G	0V	Digital Core ground
T25	NA	AO	3.3V	NA
T26	VSS	G	0V	Digital core ground
U14	VCC_M1	P	0.9V	Digital Core power
U15	VSS	G	0V	Digital Core ground
U16	VCC_M1	P	0.9V	Digital Core power
U17	VSS	G	0V	Digital Core ground
U18	VCC_M1_FB	P	0.9V	Digital Core power FeedBack
U19	VSS_FB	G	0V	Digital Core ground FeedBack
U20	VSS	G	0V	Digital core ground
U21	GPIO_123	I/O	1.8V	General Purpose I/O 123
U22	GPIO_125	I/O	1.8V	General Purpose I/O 125
U23	NA	AI	1.8V	NA
U24	NA	AO	3.3V	NA
U25	GPIO_126	I/O	1.8V	General Purpose I/O 126
U26	GPIO_127	I/O	1.8V	General Purpose I/O 127
V14	VSS	G	0V	Digital Core ground
V15	VCC_M1	P	0.9V	Digital Core power
V16	VSS	G	0V	Digital Core ground
V17	VCC_M1	P	0.9V	Digital Core power
V18	VSS	G	0V	Digital Core ground
V19	VSS	G	0V	Digital Core ground

Pin ID	Name	Type	Power Domain	Function
V20	VSS	G	0V	Digital Core ground
V21	GPIO_121	I/O	1.8V	General Purpose I/O 121
V22	VSS	G	0V	Digital Core ground
V23	GPIO_124	I/O	1.8V	General Purpose I/O 124
V24	GPIO_120	I/O	1.8V	General Purpose I/O 120
V25	VSS	G	0V	Digital Core ground
V26	GPIO_122	I/O	1.8V	General purpose I/O 122
W14	VCC_M1	P	0.9V	Digital Core power
W15	VSS	G	0V	Digital Core ground
W16	VCC_M1	P	0.9V	Digital Core power
W17	VSS	G	0V	Digital Core ground
W18	VCC_M1	P	0.9V	Digital Core power
W19	VSS	G	0V	Digital Core ground
W20	VSS	G	0V	Digital Core ground
W21	GPIO_110	I/O	1.8V	General Purpose I/O 110
W22	GPIO_117	I/O	1.8V	General Purpose I/O 117
W23	GPIO_116	I/O	1.8V	General Purpose I/O 116
W24	VSS	G	0V	Digital Core ground
W25	GPIO_119	I/O	1.8V	General Purpose I/O 119
W26	GPIO_118	I/O	1.8V	General Purpose I/O 118
Y14	MMC1_VCC_CAP	RO	1.8V	SD card 1.8V LDO cap
Y15	GPIO2_VCC_CAP	RO	1.8V	GPIO2 1.8V LDO cap
Y16	VSS	G	0V	Digital Core ground
Y17	VSS	G	0V	Digital Core ground
Y18	VSS	G	0V	Digital Core ground
Y19	VSS	G	0V	Digital Core ground
Y20	VSS	G	0V	Digital Core ground
Y21	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
Y22	GPIO_26	I/O	1.8V	General Purpose I/O 26
Y23	GPIO_27	I/O	1.8V	General Purpose I/O 27
Y24	VSS	G	0V	Digital Core ground
Y25	GPIO_28	I/O	1.8V	General Purpose I/O 28

Pin ID	Name	Type	Power Domain	Function
Y26	GPIO_115	I/O	1.8V	General Purpose I/O 115
AA14	VCC1833_MMC1	P	1.8V/3.3V	SD card IO power
AA15	VCC1833_GPIO2	P	1.8V/3.3V	GPIO2 IO power
AA16	MMC1_DAT2	I/O	1.8V/3.3V	SD card data 2
AA17	VSS	G	0V	Digital Core ground
AA18	VSS	G	0V	Digital Core ground
AA19	GPIO_32	I/O	1.8V	General Purpose I/O 32
AA20	GPIO_29	I/O	1.8V	General Purpose I/O 29
AA21	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AA22	GPIO_21	I/O	1.8V	General Purpose I/O 21
AA23	GPIO_24	I/O	1.8V	General Purpose I/O 24
AA24	GPIO_23	I/O	1.8V	General Purpose I/O 23
AA25	GPIO_25	I/O	1.8V	General Purpose I/O 25
AA26	VSS	G	0V	Digital Core ground
AB14	MMC1_DAT0	I/O	1.8V/3.3V	SD card data 0
AB15	GPIO_78	I/O	1.8V/3.3V	General Purpose I/O 78
AB16	GPIO_77	I/O	1.8V/3.3V	General Purpose I/O 77
AB17	GPIO_02	I/O	1.8V	General Purpose I/O 02
AB18	GPIO_03	I/O	1.8V	General Purpose I/O 03
AB19	VSS	G	0V	Digital Core ground
AB20	VSS	G	0V	Digital Core ground
AB21	GPIO_41	I/O	1.8V	General Purpose I/O 41
AB22	GPIO_44	I/O	1.8V	General Purpose I/O 44
AB23	GPIO_19	I/O	1.8V	General Purpose I/O 19
AB24	VSS	G	0V	Digital Core ground
AB25	GPIO_20	I/O	1.8V	General Purpose I/O 20
AB26	GPIO_22	I/O	1.8V	General Purpose I/O 22
AC14	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AC15	GPIO_79	I/O	1.8V/3.3V	General Purpose I/O 79
AC16	VSS	G	0V	Digital Core ground
AC17	GPIO_05	I/O	1.8V	General Purpose I/O 05
AC18	GPIO_00	I/O	1.8V	General Purpose I/O 00

Pin ID	Name	Type	Power Domain	Function
AC19	VSS	G	0V	Digital Core ground
AC20	GPIO_31	I/O	1.8V	General Purpose I/O 31
AC21	GPIO_34	I/O	1.8V	General Purpose I/O 34
AC22	GPIO_42	I/O	1.8V	General Purpose I/O 42
AC23	GPIO_43	I/O	1.8V	General Purpose I/O 43
AC24	GPIO_17	I/O	1.8V	General Purpose I/O 17
AC25	VSS	G	0V	Digital Core ground
AC26	GPIO_18	I/O	1.8V	General Purpose I/O 18
AD14	MMC1_CMD	I/O	1.8V/3.3V	SD card command
AD15	GPIO_76	I/O	1.8V/3.3V	General Purpose I/O 76
AD16	VSS	G	0V	Digital Core ground
AD17	GPIO_04	I/O	1.8V	General Purpose I/O 04
AD18	GPIO_01	I/O	1.8V	General Purpose I/O 01
AD19	GPIO_30	I/O	1.8V	General Purpose I/O 30
AD20	GPIO_33	I/O	1.8V	General Purpose I/O 33
AD21	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AD22	VCC18_GPIO	P	1.8V	GPIO1/4/5/PMIC I/O power
AD23	GPIO_14	I/O	1.8V	General Purpose I/O 14
AD24	GPIO_12	I/O	1.8V	General Purpose I/O 12
AD25	GPIO_16	I/O	1.8V	General Purpose I/O 16
AD26	GPIO_15	I/O	1.8V	General Purpose I/O 15
AE14	MMC1_CLK	I/O	1.8V/3.3V	SD card clock
AE15	MMC1_DAT3	I/O	1.8V/3.3V	SD card data 3
AE16	GPIO_75	I/O	1.8V/3.3V	General Purpose I/O 75
AE17	GPIO_11	I/O	1.8V	General Purpose I/O 11
AE18	GPIO_07	I/O	1.8V	General Purpose I/O 07
AE19	GPIO_10	I/O	1.8V	General Purpose I/O 10
AE20	GPIO_37	I/O	1.8V	General Purpose I/O 37
AE21	GPIO_35	I/O	1.8V	General Purpose I/O 35
AE22	GPIO_38	I/O	1.8V	General Purpose I/O 38
AE23	GPIO_46	I/O	1.8V	General Purpose I/O 46
AE24	VSS	G	0V	Digital Core ground

Pin ID	Name	Type	Power Domain	Function
AE25	GPIO_13	I/O	1.8V	General Purpose I/O 13
AE26	VSS	G	0V	Digital Core ground
AF14	MMC1_DAT1	I/O	1.8V/3.3V	SD card data 1
AF15	VSS	G	0V	Digital Core ground
AF16	GPIO_80	I/O	1.8V/3.3V	General Purpose I/O 80
AF17	GPIO_08	I/O	1.8V	General Purpose I/O 08
AF18	GPIO_06	I/O	1.8V	General Purpose I/O 06
AF19	GPIO_09	I/O	1.8V	General Purpose I/O 09
AF20	VSS	G	0V	Digital Core ground
AF21	GPIO_40	I/O	1.8V	General Purpose I/O 40
AF22	GPIO_36	I/O	1.8V	General Purpose I/O 36
AF23	GPIO_39	I/O	1.8V	General Purpose I/O 39
AF24	GPIO_45	I/O	1.8V	General Purpose I/O 45
AF25	VSS	G	0V	Digital Core ground
AF26	VSS	G	0V	Digital Core ground

3.3 I/O Pin Parameters

3.3.1 For 1.8V I/O Pins

Power Domain	Symbol	Description	Min	Typ	Max
1.8V Input	Vih	High level input	VCC×0.7V	1.8V	VCC+0.2V
	Vil	Low level input	-0.3V	0V	VCCx0.3V
	Rpu	Pull up resister	55kOhm	79KOhm	121kOhm
	Rpd	Pull down resister	51kOhm	87kOhm	169kOhm
	Iil	Input leakage current Pad in input mode			10uA
1.8V Ouput	Voh	High level output	VCC-0.2V		
	Vol	Low level output			0.2V
	Iol DCS[1:0]= 00 01 10 11	Low level output current when Vpad=0.2V	 13mA 25mA 37mA 49mA		

Power Domain	Symbol	Description	Min	Typ	Max
	Ioh DCS[1:0]= 00 01 10 11	High level output current when Vpad=VCC-0.2V	 11mA 21mA 32mA 42mA		

3.3.2 For 3.3V I/O Pins

Power Domain	Symbol	Description	Min	Typ	Max
3.3V Input	Vih	High level input	2V		VCC+0.3V
	Vil	Low level input	-0.3V	0V	0.8V
	Rpu	Pull up resister	26kOhm	47kOhm	72kOhm
	Rpd	Pull down resister	27kOhm	54kOhm	267kOhm
	Iil	Input leakage current			10uA
3.3V Ouput	Voh	High level output	2.4V		
	Vol	Low level output			0.4V
	Iol DS[2:0]= 000 001 010 011 100 101 110 111	Low level output current when Vpad=0.4V	 7mA 10mA 14mA 18mA 21mA 24mA 28mA 31mA		
	Ioh DS[2:0]= 000 001 010 011 100 101 110 111	High level output current when Vpad=VCC-0.5V	 7mA 10mA 13mA 16mA 19mA 23mA 26mA 29mA		

3.4 Multiplexed Signal/Pin Functions

The **Function 0** through 7 signals is assigned to the I/O pins of K1.

Most I/O pins of K1 are multi-function allowing them to be configured for one of several available functions using Multi-Function Pin Registers (MFPRs). Additionally, some functions can be configured to be present on several different pins.

The assigned signals are organized by their functions (e.g. power supply, clock, etc.) which are arranged in groups according to their interfaces (e.g. JTAG, SPIx, etc.) as per description in the following subsections (sorted alphabetically for user convenience).

Note. Definition of symbols used for signal/pin type:

- I = Input
- O = Output
- I/O = Input/Output
- OD = Open-Drain
- RO = Reference output

3.4.1 JTAG

3.4.1.1 Primary

Signal/Pin		Description
Name	**Type**	
PRI_TCK	I	Primary JTAG interface 1 test clock. Used for all transfers on the JTAG test interface.
PRI_TDI	I	Primary JTAG interface 1 test data input. Used to send data from the JTAG controller to the K1 processor. This pin has an internal pullup resistor.
PRI_TDO	O	Primary JTAG Interface 1 test data output Used to return data from the K1 processor to the JTAG controller.
PRI_TMS	I	Primary JTAG Interface 1 test mode select. Used to select the test mode required from the JTAG controller. This pin has an internal pullup resistor.
PRI_TRSTn	I	Primary JTAG Interface 1 test reset. Used for IEEE 1194.1 test reset.
VCXO_OUT	O	24 MHz VCXO output clock
VCXO_REQ	I	OCLK1 request

3.4.1.2 Secondary

Signal/Pin		Description
Name	**Type**	
SEC2_TCK	I	Secondary JTAG Interface 2 test clock. Used for all transfers on the JTAG test interface.

Signal/Pin		Description
Name	Type	
SEC2_TDI	I	Secondary JTAG Interface 2 test data input. Used to send data from the JTAG controller to the K1 processor. This pin has an internal pullup resistor.
SEC2_TDO	O	Secondary JTAG Interface 2 test data output. Used to return data from the K1 processor to the JTAG controller.
SEC2_TMS	I	Secondary JTAG Interface 2 test mode select. Used to select the test mode required from the JTAG controller. This pin has an internal pullup resistor.
SEC2_TRSTn	I	Secondary JTAG Interface 2 test reset. Used for IEEE 1194.1 test reset.

3.4.2 Keypad Controller

Signal/Pin		Description
Name	Type	
KP_DK[4: 0]	I	Keypad direct key inputs [4: 0]
KP_MKIN[3: 0]	I	Keypad matrix key inputs [3: 0]
KP_MKOUT[3: 0]	O	Keypad matrix key outputs [3: 0]

3.4.3 Miscellaneous

Signal/Pin		Description
Name	Type	
MPLL_TST_CK		PLL test pin
MN_CLK_OUT	O	Fractional (M/N) divided clock. Main PMU general purpose M/N fractional clock divider clock output. CLK_REQ must be set as Function 0 and pulled high for the 13 MHz clock to be output on GPIO[122] (MN_CLK_OUT).
Sleep_OUT	O	PMIC sleep setting

3.4.4 SPIx

Signal/Pin		Description
Name	Type	
SPIx_FRM	I/O	Synchronous serial port frame 0/2. The serial frame sync can be configured as an output (master mode operation) or an input (slave mode operation).
SPIx_RXD	I	Synchronous serial port receive data 0/2. Serial data latched using the bit clock.
SPIx_SCLK	I/O	Synchronous serial port clock 0/2. The serial bit clock can be configured as an output (master mode operation) or an input (slave mode operation).
SPIx_TXD	O	Synchronous serial port transmit data 0/2. Serial data driven out synchronously with the bit clock.

3.4.5 TWSI

3.4.5.1 Dedicated

Signal/Pin		Description
Name	Type	
PWR_SDA	I/O	TWSI serial data/address signal
PWR_SCL	I/O	TWSI serial clock line signal

3.4.5.2 Common

Signal/Pin		Description
Name	Type	
I2Cx_SCL	I/O,OD	TWSIx clock
I2Cx_SDA	I/O,OD	TWSIx data

3.4.6 UARTx

Signal/Pin		Description
Name	Type	
UARTx_CTSn	I	UARTx clear-to-send
UARTx_RTSn	O	UARTx request-to-send

Signal/Pin		Description
Name	Type	
UARTx_RXD	I	UARTx receive data
UARTx_TXD	O	UARTx transmit data

3.4.7 USB

Signal/Pin		Description
Name	Type	
USBx_N	I/O	USB D±
USBx_P	I/O	
VBUS_ON	I	USB VBUS present indicator

3.5 Multi-Function I/O Pin Assignments

All functions that are assigned to a pin as its primary functions are tabled below.

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
QSPI	QSPI_DAT3	DOWN	ENABLE	QSPI_DAT[3]/strap[3]	GPIO[98]		UART1_TXD <secure domain>			
	QSPI_DAT2	DOWN	ENABLE	QSPI_DAT[2]/strap[2]	GPIO[99]		UART1_RXD <secure domain>			
	QSPI_DAT1	DOWN	ENABLE	QSPI_DAT[1]/strap[1]	GPIO[100]		UART1_CTS <secure domain>	UART4_TXD		
	QSPI_DAT0	DOWN	ENABLE	QSPI_DAT[0]/strap[0]	GPIO[101]		UART1_RTS <secure domain>	UART4_RXD		
	QSPI_CLK	DOWN	ENABLE	QSPI_CLK	GPIO[102]		UART5_TXD			
	QSPI_CS1	UP	ENABLE	QSPI_CS1	GPIO[103]		UART5_RXD			
SD/MMC	MMC1_DAT3	UP	ENABLE	MMC1_DAT[3]	R_I2S2_SCLK	SEC2_TMS	UART0_TXD	GPIO[104]	PWM0	
	MMC1_DAT2	UP	ENABLE	MMC1_DAT[2]	R_I2S2_LRCK	SEC2_TDI	UART0_RXD	GPIO[105]	PWM1	
	MMC1_DAT1	UP	ENABLE	MMC1_DAT[1]	R_I2S2_TXD	SEC2_TDO		GPIO[106]	PWM2	

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	MMC1_DAT0	UP	ENABLE	MMC1_DAT[0]	R_I2S2_RXD	SEC2_TRSTn		GPIO[107]	PWM3	
	MMC1_CMD	UP	ENABLE	MMC1_CMD	UART0_TXD	CPU_SEL	R_UART0_TXD	GPIO[108]	PWM4	
	MMC1_CLK	DOWN	ENABLE	MMC1_CLK	R_I2S2_SYSCLK	SEC2_TCK		GPIO[109]	PWM5	
PMIC	RESET_IN_N	UP	NO	RESET_IN_N						
	EXT_32K_IN	DOWN	NO	EXT_32K_IN						
	PWR_SCL	UP	ENABLE	PWR_SCL	GPIO[93]					
	PWR_SDA	UP	ENABLE	PWR_SDA	GPIO[94]					
	SLEEP_OUT	NO	ENABLE	SLEEP_OUT	GPIO[95]					
	DVL0	DOWN	ENABLE	DVL0	GPIO[96]		VCXO_REQ			
	DVL1	DOWN	ENABLE	DVL1	GPIO[97]	IR_RX	VCXO_OUT			
	PMIC_INT_N	UP	ENABLE	PMIC_INT_N						
	GPIO[81]	UP	ENABLE	GPIO[81]	R_I2S3_SCLK	UART3_TXD	UART4_CTS_N	MN_CLK	AP_I2C5_SCL	
	GPIO[82]	UP	ENABLE	GPIO[82]	R_I2S3_LRCK	UART3_RXD	UART4_RTS_N	UART8_TXD	AP_I2C5_SDA	
	GPIO[83]	UP	ENABLE	GPIO[83]	R_I2S3_TXD	UART3_CTS_N	UART4_TXD	UART8_RXD	AP_I2C6_SCL	
	GPIO[84]	UP	ENABLE	GPIO[84]	R_I2S3_RXD	UART3_RTS_N	UART4_RXD	AP_I2C2_SCL		
	GPIO[85]	UP	ENABLE	GPIO[85]	R_I2S3_SYSCLK	UART6_CTS_N	MN_CLK2	AP_I2C2_SDA		
	GPIO[86]	UP	ENABLE	GPIO[86]	HDMI_TX_HSCL	UART6_TXD	DCLK <SPI_LCD>	UART7_CTS_N		

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	GPIO[87]	UP	ENABLE	GPIO[87]	HDMI_TX_HSDA	UART6_RXD	DCX/DOUT1 <SPI_LCD>	UART7_RTS_N		
	GPIO[88]	DOWN	ENABLE	GPIO[88]	HDMI_TX_HCEC	UART7_TXD	DIN <SPI_LCD>	PWM6		
	GPIO[89]	DOWN	ENABLE	GPIO[89]	HDMI_TX_PDP	UART7_RXD	DOUT0 <SPI_LCD>	VCXO_REQ		
	GPIO[90]	DOWN	ENABLE	GPIO[90]/strap[4]		UART6_RTS_N	CS<SPI_LCD>	VCXO_OUT	AP_I2C6_SDA	
	GPIO[91]	UP	ENABLE	GPIO[91]	MN_CLK2	VCXO_OUT	DSI_TE	R_I2C0_SCL		
	GPIO[92]	UP	ENABLE	GPIO[92]	MN_CLK	PWM7		R_I2C0_SDA		
	JTAG_SEL	DOWN	NO	JTAG_SEL						
GPIO1	GPIO[0]	DOWN	ENABLE	GPIO[0]	GMAC0_RXDV	UART6_TXD	PWM8			
	GPIO[1]	DOWN	ENABLE	GPIO[1]	GMAC0_RX_D0	UART6_RXD	PWM9			
	GPIO[2]	DOWN	ENABLE	GPIO[2]	GMAC0_RX_D1	UART6_CTS_N	PWM10			
	GPIO[3]	DOWN	ENABLE	GPIO[3]	GMAC0_RX_CLK	UART6_RTS_N	PWM11			
	GPIO[4]	DOWN	ENABLE	GPIO[4]	GMAC0_RX_D2	UART7_TXD	PWM12			
	GPIO[5]	DOWN	ENABLE	GPIO[5]	GMAC0_RX_D3	UART7_RXD	PWM13			
	GPIO[6]	DOWN	ENABLE	GPIO[6]	GMAC0_TX_D0	UART7_CTS_N	PWM14			
	GPIO[7]	DOWN	ENABLE	GPIO[7]	GMAC0_TX_D1	UART7_RTS_N	PWM15			
	GPIO[8]	DOWN	ENABLE	GPIO[8]	GMAC0_TX	UART8_TXD				
	GPIO[9]	DOWN	ENABLE	GPIO[9]	GMAC0_TX_D2	UART8_RXD	PWM16			

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	GPIO[10]	DOWN	ENABLE	GPIO[10]	GMAC0_TX_D3	UART8_CTS_N	PWM17			
	GPIO[11]	DOWN	ENABLE	GPIO[11]	GMAC0_TX_EN	UART8_RTS_N	PWM18			
	GPIO[12]	DOWN	ENABLE	GPIO[12]	GMAC0_MDC	UART9_TXD	VCXO_OUT			
	GPIO[13]	DOWN	ENABLE	GPIO[13]	GMAC0_MDIO	UART9_RXD	PWM19			
	GPIO[14]	DOWN	ENABLE	GPIO[14]	GMAC0_INT_N		PWM0			
	GPIO[15]	UP	ENABLE	GPIO[15]	MMC2_DATA3	PCIe0_PERSTN		PCIe1_PERSTN		
	GPIO[16]	UP	ENABLE	GPIO[16]	MMC2_DATA2	PCIe0_WAKEN	VCXO_REQ	PCIe1_WAKEN		
	GPIO[17]	UP	ENABLE	GPIO[17]	MMC2_DATA1	PCIe0_CLKREQN	VCXO_OUT	PCIe1_CLKREQN		
	GPIO[18]	UP	ENABLE	GPIO[18]	MMC2_DATA0	UART3_TXD		PCIe2_PERSTN		
	GPIO[19]	UP	ENABLE	GPIO[19]	MMC2_CMD	UART3_RXD		PCIe2_WAKEN		
	GPIO[20]	UP	ENABLE	GPIO[20]	MMC2_CLK	UART3_CTS_N	MN_CLK	PCIe2_CLKREQN		
	GPIO[21]	DOWN	ENABLE	GPIO[21]	UART2_TXD	UART3_RTS_N	32K_OUT			
	GPIO[22]	DOWN	ENABLE	GPIO[22]	UART2_RXD	PWM2		PWM0		
	GPIO[23]	DOWN	ENABLE	GPIO[23]	UART2_CTS_N	UART4_TXD	MN_CLK	PWM1		
	GPIO[24]	DOWN	ENABLE	GPIO[24]	UART2_RTS_N	UART4_RXD	I2S1_SYSCLK	PWM2		
	GPIO[25]	DOWN	ENABLE	GPIO[25]	I2S1_SCLK	UART5_TXD		PWM3		
	GPIO[26]	DOWN	ENABLE	GPIO[26]	I2S1_LRCK	UART5_RXD				

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	GPIO[27]	DOWN	ENABLE	GPIO[27]	I2S1_TXD	UART5_CTS_N				
	GPIO[28]	DOWN	ENABLE	GPIO[28]	I2S1_RXD	UART5_RTS_N		32K_OUT		
	GPIO[29]	DOWN	ENABLE	GPIO[29]	GMAC1_RXDV	UART1_TXD <secure domain>	PWM1	PCIe0_PERSTN		
	GPIO[30]	DOWN	ENABLE	GPIO[30]	GMAC1_RX_D0	UART1_RXD <secure domain>	PWM2	PCIe0_WAKEN		
	GPIO[31]	DOWN	ENABLE	GPIO[31]	GMAC1_RX_D1	UART1_CTS_N <secure domain>	32K_OUT	PCIe0_CLKREQN		
	GPIO[32]	DOWN	ENABLE	GPIO[32]	GMAC1_RX_CLK	UART1_RTS_N <secure domain>	MN_CLK	PCIe1_PERSTN		
	GPIO[33]	DOWN	ENABLE	GPIO[33]	GMAC1_RX_D2	UART4_TXD	PWM3	PCIe1_WAKEN		
	GPIO[34]	DOWN	ENABLE	GPIO[34]	GMAC1_RX_D3	UART4_RXD	PWM4	PCIe1_CLKREQN		
	GPIO[35]	DOWN	ENABLE	GPIO[35]	GMAC1_TX_D0	UART4_CTS_N	PWM5	PCIe2_PERSTN		
	GPIO[36]	DOWN	ENABLE	GPIO[36]	GMAC1_TX_D1	UART4_RTS_N	PWM6	PCIe2_WAKEN		
	GPIO[37]	DOWN	ENABLE	GPIO[37]	GMAC1_TX	PWM7		PCIe2_CLKREQN		
	GPIO[38]	UP	ENABLE	GPIO[38]	GMAC1_TX_D2	AP_I2C3_SCL <secure domain>	R_I2S3_SCLK	PWM8		
	GPIO[39]	UP	ENABLE	GPIO[39]	GMAC1_TX_D3	AP_I2C3_SDA <secure domain>	R_I2S3_LRCK	PWM9		
	GPIO[40]	UP	ENABLE	GPIO[40]	GMAC1_TX_EN	AP_I2C4_SCL	R_I2S3_TXD	PWM10		
	GPIO[41]	UP	ENABLE	GPIO[41]	GMAC1_MDC	AP_I2C4_SDA	R_I2S3_RXD	PWM11		
	GPIO[42]	DOWN	ENABLE	GPIO[42]	GMAC1_MDIO	UART5_TXD	R_I2S3_SYSCLK	PWM12		
	GPIO[43]	DOWN	ENABLE	GPIO[43]	GMAC1_INT_N	UART5_RXD		PWM13		

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	GPIO[44]	DOWN	ENABLE	GPIO[44]	MN_CLK	UART5_CTS_N	R_IR_RX	PWM14		
	GPIO[45]	DOWN	ENABLE	GPIO[45]	GMAC0_CLK_REF	UART5_RTS_N		PWM15		
	GPIO[46]	DOWN	ENABLE	GPIO[46]	GMAC1_CLK_REF			PWM16		
	GPIO[110]	DOWN	ENABLE	GPIO[110]	R_CAN_TX0	R_UART1_TXD	UART9_CTS_N	PCIe0_PERSTN	ONE_WIRE	
	GPIO[115]	DOWN	ENABLE	GPIO[115]	R_CAN_RX0	R_UART1_RXD	UART9_RTS_N	PCIe0_WAKEN		
	GPIO[116]	DOWN	ENABLE	GPIO[116]	R_PWM1	R_UART1_CTS_N	UART9_TXD	PCIe0_CLKREQN	VCXO_REQ[1]	
	GPIO[117]	DOWN	ENABLE	GPIO[117]	R_PWM2	R_UART1_RTS_N	UART9_RXD	PCIe2_CLKREQN	VCXO_CLK_OUT	
	GPIO[118]	UP	ENABLE	GPIO[118]	AP_I2C7_SCL (CAM)	AP_I2C6_SCL	I2S0_SCLK	R_PWM8	KP_MKIN[0]	
	GPIO[119]	UP	ENABLE	GPIO[119]	AP_I2C7_SDA (CAM)	AP_I2C6_SDA	I2S0_LRCK	R_PWM9	KP_MKOUT[0]	
	GPIO[120]	DOWN	ENABLE	GPIO[120]	CAM_MCLK2		I2S0_TXD	R_PWM6	KP_MKIN[1]	
	GPIO[121]	DOWN	ENABLE	GPIO[121]	CAMERA2_RST	VBUS_ON2	I2S0_RXD	R_PWM7	KP_MKOUT[1]	
	GPIO[122]	DOWN	ENABLE	GPIO[122]	CAMERA2_PDN	USB_ID2	I2S0_SYSCLK		KP_MKIN[2]	
	GPIO[123]	DOWN	ENABLE	GPIO[123]	DRIVE_VBUS2_ISO	KP_DKIN[0]	KP_MKIN[0]			
	GPIO[124]	DOWN	ENABLE	GPIO[124]	DRIVE_VBUS1_ISO	KP_DKIN[1]	KP_MKOUT[0]			
	GPIO[125]	DOWN	ENABLE	GPIO[125]	VBUS_ON0	KP_DKIN[2]	KP_MKIN[1]			
	GPIO[126]	DOWN	ENABLE	GPIO[126]	USB_ID0	KP_DKIN[3]	KP_MKOUT[1]			
	GPIO[127]	DOWN	ENABLE	GPIO[127]	DRIVE_VBUS0_ISO	KP_DKIN[4]	KP_MKIN[2]			

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
GPIO 2	GPIO[75]	UP	ENABLE	GPIO[75]	SPI2_SCLK <secure domain>	SPI3_SCLK	CAN_TX0	UART8_TXD	AP_I2C4_SCL	
	GPIO[76]	UP	ENABLE	GPIO[76]	SPI2_FRM <secure domain>	SPI3_FRM	CAN_RX0	UART8_RXD	AP_I2C4_SDA	
	GPIO[77]	UP	ENABLE	GPIO[77]	SPI2_TXD <secure domain>	SPI3_TXD	AP_I2C3_SCL <secure domain>	UART8_CTS_N	R_PWM0	KP_MKOUT[2]
	GPIO[78]	UP	ENABLE	GPIO[78]	SPI2_RXD <secure domain>	SPI3_RXD	AP_I2C3_SDA <secure domain>	UART8_RTS_N	R_PWM1	KP_MKIN[3]
	GPIO[79]	DOWN	ENABLE	GPIO[79]	IR_RX	R_PWM2				KP_MKOUT[3]
	GPIO[80]	DOWN	ENABLE	GPIO[80]	MMC_Card_detect	R_PWM3	UART0_RXD	R_UART0_RXD		
GPIO 3	GPIO[47]	UP	ENABLE	GPIO[47]	R_UART0_TXD	R_CAN_TX0	R_PWM8	AP_I2C3_SCL<secure domain>	ONE_WIRE	
	GPIO[48]	UP	ENABLE	GPIO[48]	R_UART0_RXD	R_CAN_RX0	R_IR_RX	AP_I2C3_SDA<secure domain>	KP_MKOUT[2]	
	GPIO[49]	UP	ENABLE	GPIO[49]	R_SPI_SCLK	R_UART1_CTS_N	R_PWM4	R_I2C0_SCL	KP_MKIN[3]	
	GPIO[50]	UP	ENABLE	GPIO[50]	R_SPI_FRM	R_UART1_RTS_N	R_PWM5	R_I2C0_SDA	KP_MKOUT[3]	
	GPIO[51]	UP	ENABLE	GPIO[51]	R_SPI_TXD	R_UART1_TXD	R_PWM6	AP_I2C4_SCL		
	GPIO[52]	UP	ENABLE	GPIO[52]	R_SPI_RXD	R_UART1_RXD	R_PWM7	AP_I2C4_SDA		
GPIO 4	GPIO[53]	DOWN	ENABLE	GPIO[53]	CAM_MCLK0	PWM17	PCIe0_CLKREQN	UART3_TXD		
	GPIO[54]	UP	ENABLE	GPIO[54]	AP_I2C0_SCL (CAM)	CAN_TX0	PCIe0_PERSTN	UART3_RXD	AP_I2C5_SCL	
	GPIO[55]	UP	ENABLE	GPIO[55]	AP_I2C0_SDA (CAM)	CAN_RX0	PCIe0_WAKEN	UART3_CTS_N	AP_I2C5_SDA	
	GPIO[56]	UP	ENABLE	GPIO[56]	AP_I2C1_SCL (CAM)	UART6_TXD	PCIe1_PERSTN	UART3_RTS_N	AP_I2C6_SCL	
	GPIO[57]	UP	ENABLE	GPIO[57]	AP_I2C1_SDA (CAM)	UART6_RXD	PCIe1_WAKEN	PWM18	AP_I2C6_SDA	
	GPIO[58]	DOWN	ENABLE	GPIO[58]	CAM_MCLK1	I2S0_SYSCLK	PCIe1_CLKREQN	IR_RX		

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	GPIO[111]	DOWN	ENABLE	GPIO[111]	CAMERA0_RST	I2S0_SCLK	PCIe2_PERSTN	UART4_TXD		
	GPIO[112]	DOWN	ENABLE	GPIO[112]	CAMERA1_RST	I2S0_LRCK	PCIe2_WAKEN	UART4_RXD		
	GPIO[113]	DOWN	ENABLE	GPIO[113]	CAMERA0_PDN	I2S0_TXD	PCIe2_CLKREQN	UART4_CTS_N		
	GPIO[114]	DOWN	ENABLE	GPIO[114]	CAMERA1_PDN	I2S0_RXD	DSI_TE	UART4_RTS_N		
	GPIO[63]	DOWN	ENABLE	GPIO[63]	DRIVE_VBUS0_ISO	R_I2S2_SYSCLK		PWM19	KP_DKIN[0]	
	GPIO[64]	DOWN	ENABLE	GPIO[64]	VBUS_ON0	R_I2S2_SCLK	SPI2_SCLK <secure domain>	R_PWM0	KP_DKIN[1]	
	GPIO[65]	UP	ENABLE	GPIO[65]	USB_ID0	R_I2S2_LRCK	SPI2_FRM <secure domain>	R_PWM1	KP_DKIN[2]	
	GPIO[66]	DOWN	ENABLE	GPIO[66]	DRIVE_VBUS1_ISO	R_I2S2_TXD	SPI2_TXD <secure domain>	R_PWM2	KP_DKIN[3]	
	GPIO[67]	DOWN	ENABLE	GPIO[67]	DRIVE_VBUS2_ISO	R_I2S2_RXD	SPI2_RXD <secure domain>	R_PWM3	KP_DKIN[4]	
	GPIO[68]	DOWN	ENABLE	GPIO[68]	VBUS_ON2	UART0_TXD	AP_I2C2_SCL	R_PWM4		
	GPIO[69]	UP	ENABLE	GPIO[69]	USB_ID2	UART0_RXD	AP_I2C2_SDA	R_PWM5		
GPIO5	GPIO[59]	UP	ENABLE	GPIO[59]	HDMI_TX_HSCL	SPI3_SCLK	UART1_TXD <secure domain>	PCIe1_PERSTN		
	GPIO[60]	UP	ENABLE	GPIO[60]	HDMI_TX_HSDA	SPI3_FRM	UART1_RXD <secure domain>	PCIe1_WAKEN		
	GPIO[61]	UP	ENABLE	GPIO[61]	HDMI_TX_HCEC	SPI3_TXD	UART1_CTS_N <secure domain>	PCIe1_CLKREQN		
	GPIO[62]	UP	ENABLE	GPIO[62]	HDMI_TX_PDP	SPI3_RXD	UART1_RTS_N <secure domain>	PCIe2_PERSTN		
	PRI_TDI	UP	NO	PRI_TDI	GPIO[70]	AP_I2C2_SCL	DCLK <SPI_LCD>	UART5_TXD		
	PRI_TMS	UP	NO	PRI_TMS	GPIO[71]	AP_I2C2_SDA	DCX/DOUT1 <SPI_LCD>	UART5_RXD		

Group	Pad Name	Default Pulling	Pad Edge Detected	Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6
	PRI_TCK	DOWN	NO	PRI_TCK	GPIO[72]	UART9_TXD	DIN<SPI_LCD>	UART5_CTS_N		
	PRI_TDO	UP	NO	PRI_TDO	GPIO[73]	UART9_RXD	DOUT0 <SPI_LCD>	UART5_RTS_N		
	PRI_TRSTn	UP	NO	PRI_TRSTn						
	GPIO[74]	UP	ENABLE	GPIO[74]		PWM9	CS<SPI_LCD>	PCIe2_WAKEN		
EMMC5.1	EMMC_D0			EMMC_D0	GPIO[93]					
	EMMC_D1			EMMC_D1	GPIO[94]					
	EMMC_D2			EMMC_D2	GPIO[95]					
	EMMC_D3			EMMC_D3	GPIO[96]					
	EMMC_D4			EMMC_D4	GPIO[97]					
	EMMC_D5			EMMC_D5	GPIO[98]					
	EMMC_D6			EMMC_D6	GPIO[99]					
	EMMC_D7			EMMC_D7	GPIO[100]					
	EMMC_DS			EMMC_DS	GPIO[101]					
	EMMC_CLK			EMMC_CLK	GPIO[102]					
	EMMC_CMD			EMMC_CMD	GPIO[103]					

3.6 Power Supply Pins

Pin Name	Domain Name	Domain Voltage	Description
AUD_VDDU09	AUDIO	0.9V	0.9V power for audio

Pin Name	Domain Name	Domain Voltage	Description
AUD_VNEG	AUDIO	-1.8V	Negative voltage for headphone driver
AUD_VPOS	AUDIO	1.8V	Positive voltage for headphone driver
AVDD18_AUD	AUDIO	1.8V	1.8V power for audio
AVDD3V3_AUD	AUDIO	3.3V	3.3V power for earphone driver
VCC_M1	CORE	0.9V	Digital core power
AVDD09_CSI	CSI	0.9V	MIPI_CSI digital power
AVDD18_CSI	CSI	1.8V	MIPI_CSI analog power
AVDD09_AFEAP	DCXO	0.9V	0.9V power for DCXO
AVDD18_AFEAP	DCXO	1.8V	1.8V power for DCXO
AVDD06_DDR	DDR	lp4x: 0.6V lp4: TBD lp3: TBD	LPDDR4X IO power
AVDD11_DDR	DDR	lp4x:1.1V lp4:1.1V lp3: 1.2V	LPDDR PHY power supply
AVDD18_DDR	DDR	1.8V	LPDDR PHY PLL 1.8V power
AVDD18_PHY	DDR	1.8V	Analog 1.8V power
AVDDU_DDR	DDR	0.9V	LPDDR PHY PLL logical power
AVDDU_PHY	DDR	0.9V	LPDDR PHY core logical power
DDR_LDO_CAP	DDR	0.7~0.9V	External LDO output ball. Connect to a 100nF capacitor on PCB board.
DDR_LP23_VREFCA	DDR	lp3:0.6V lp4: high-z	CA VREF for lpddr23. LP4/4x, Keep the pin NC.
DDR_LP23_VREFDQ	DDR	lp3: 0.6V lp4: high-z	DQ VREF for lpddr23. LP4/4x, keep the pin NC.
VDDQ_V1P2	DDR	lp3: 1.2V lp4x: 0.6V	LPDDR3 IO power
AVDD09_DSI1	DSI	0.9V	DSI digital power
AVDD12_DSI1	DSI	1.2V	DSI driver power
AVDD18_DSI1	DSI	1.8V	DSI analog power
AVDD18_EFUSE	EFUSE	1.8V	ANAGRP
AVDD09_EMMC	EMMC	0.9V	eMMC digital power
AVDD18_EMMC	EMMC	1.8V	eMMC analog power
VCC18_GPIO	GPIO1/4/5/PMIC	1.8V	GPIO1/4/5/PMIC I/O power
VCC1833_GPIO2	GPIO2	1.8V/3.3V	GPIO2 IO power

Pin Name	Domain Name	Domain Voltage	Description
VCC1833_GPIO3	GPIO3	1.8V/3.3V	GPIO3 IO power
AVDD09_HDMI	HDMI	0.9V	HDMI digital power
AVDD18_HDMI	HDMI	1.8V	HDMI 1.8V power
AVDD33_HDMI	HDMI	3.3V	HDMI 3.3V power
AVDD09_PCIEA	PCIEA	0.9V	PCIEA digital power
AVDD18_PCIEA	PCIEA	1.8V	PCIEA analog power
AVDD09_PCIEB	PCIEB	0.9V	PCIEB digital power
AVDD18_PCIEB	PCIEB	1.8V	PCIEB analog power
AVDD09_PCIEC	PCIEC	0.9V	PCIEC digital power
AVDD18_PCIEC	PCIEC	1.8V	PCIEC analog power
AVDD09_PLL	PLL	0.9V	System PLL power supply
AVDD18_PLL	PLL	1.8V	System PLL power supply
VCC1833_QSPI	QSPI	1.8V/3.3V	QSPI IO power
VCC1833_MMC1	SD card	1.8V/3.3V	SD card IO power
AVDD09_USB	USB2.0	0.9V	USB2.0 digital power
AVDD18_USB	USB2.0	1.8V	USB2.0 1.8V power
AVDD33_USB	USB2.0	3.3V	USB2.0 3.3V power

3.7 Multi-Function Pin Register (MFPRs)

In K1 are defined and implemented Multi-Function Pin Registers (MFPRs). In particular, there are 129 MFPR in total, starting from the base address 0xD401_E000 with a stride of 0x4, as tabled below.

MFPR ID	Address	Offset
GPIO_00	0xD401E004	0x4
GPIO_01	0xD401E008	0x8
GPIO_02	0xD401E00C	0xC
GPIO_03	0xD401E010	0x10
GPIO_04	0xD401E014	0x14
GPIO_05	0xD401E018	0x18
GPIO_06	0xD401E01C	0x1C
GPIO_07	0xD401E020	0x20
GPIO_08	0xD401E024	0x24
GPIO_09	0xD401E028	0x28

MFPR ID	Address	Offset
GPIO_10	0xD401E02C	0x2C
GPIO_11	0xD401E030	0x30
GPIO_12	0xD401E034	0x34
GPIO_13	0xD401E038	0x38
GPIO_14	0xD401E03C	0x3C
GPIO_15	0xD401E040	0x40
GPIO_16	0xD401E044	0x44
GPIO_17	0xD401E048	0x48
GPIO_18	0xD401E04C	0x4C
GPIO_19	0xD401E050	0x50
GPIO_20	0xD401E054	0x54
GPIO_21	0xD401E058	0x58
GPIO_22	0xD401E05C	0x5C
GPIO_23	0xD401E060	0x60
GPIO_24	0xD401E064	0x64
GPIO_25	0xD401E068	0x68
GPIO_26	0xD401E06C	0x6C
GPIO_27	0xD401E070	0x70
GPIO_28	0xD401E074	0x74
GPIO_29	0xD401E078	0x78
GPIO_30	0xD401E07C	0x7C
GPIO_31	0xD401E080	0x80
GPIO_32	0xD401E084	0x84
GPIO_33	0xD401E088	0x88
GPIO_34	0xD401E08C	0x8C
GPIO_35	0xD401E090	0x90
GPIO_36	0xD401E094	0x94
GPIO_37	0xD401E098	0x98
GPIO_38	0xD401E09C	0x9C
GPIO_39	0xD401E0A0	0xA0
GPIO_40	0xD401E0A4	0xA4
GPIO_41	0xD401E0A8	0xA8

MFPR ID	Address	Offset
GPIO_42	0xD401E0AC	0xAC
GPIO_43	0xD401E0B0	0xB0
GPIO_44	0xD401E0B4	0xB4
GPIO_45	0xD401E0B8	0xB8
GPIO_46	0xD401E0BC	0xBC
GPIO_47	0xD401E0C0	0xC0
GPIO_48	0xD401E0C4	0xC4
GPIO_49	0xD401E0C8	0xC8
GPIO_50	0xD401E0CC	0xCC
GPIO_51	0xD401E0D0	0xD0
GPIO_52	0xD401E0D4	0xD4
GPIO_53	0xD401E0D8	0xD8
GPIO_54	0xD401E0DC	0xDC
GPIO_55	0xD401E0E0	0xE0
GPIO_56	0xD401E0E4	0xE4
GPIO_57	0xD401E0E8	0xE8
GPIO_58	0xD401E0EC	0xEC
GPIO_59	0xD401E0F0	0xF0
GPIO_60	0xD401E0F4	0xF4
GPIO_61	0xD401E0F8	0xF8
GPIO_62	0xD401E0FC	0xFC
GPIO_63	0xD401E100	0x100
GPIO_64	0xD401E104	0x104
GPIO_65	0xD401E108	0x108
GPIO_66	0xD401E10C	0x10C
GPIO_67	0xD401E110	0x110
GPIO_68	0xD401E114	0x114
GPIO_69	0xD401E118	0x118
PRI_TDI	0xD401E11C	0x11C
PRI_TMS	0xD401E120	0x120
PRI_TCK	0xD401E124	0x124
PRI_TDO	0xD401E128	0x128

MFPR ID	Address	Offset
GPIO_74	0xD401E12C	0x12C
GPIO_75	0xD401E130	0x130
GPIO_76	0xD401E134	0x134
GPIO_77	0xD401E138	0x138
GPIO_78	0xD401E13C	0x13C
GPIO_79	0xD401E140	0x140
GPIO_80	0xD401E144	0x144
GPIO_81	0xD401E148	0x148
GPIO_82	0xD401E14C	0x14C
GPIO_83	0xD401E150	0x150
GPIO_84	0xD401E154	0x154
GPIO_85	0xD401E158	0x158
QSPI_DAT0	0xD401E168	0x168
QSPI_DAT1	0xD401E16C	0x16C
QSPI_DAT2	0xD401E170	0x170
QSPI_DAT3	0xD401E174	0x174
QSPI_CS1	0xD401E178	0x178
QSPI_CLK	0xD401E17C	0x17C
MMC1_DAT3	0xD401E1B8	0x1B8
MMC1_DAT2	0xD401E1BC	0x1BC
MMC1_DAT1	0xD401E1C0	0x1C0
MMC1_DAT0	0xD401E1C4	0x1C4
MMC1_CMD	0xD401E1C8	0x1C8
MMC1_CLK	0xD401E1CC	0x1CC
GPIO_110	0xD401E1D0	0x1D0
PWR_SCL	0xD401E1D4	0x1D4
PWR_SDA	0xD401E1D8	0x1D8
VCXO_EN	0xD401E1DC	0x1DC
DVL0	0xD401E1E0	0x1E0
DVL1	0xD401E1E4	0x1E4
PMIC_INT_N	0xD401E1E8	0x1E8
GPIO_86	0xD401E1EC	0x1EC

MFPR ID	Address	Offset
GPIO_87	0xD401E1F0	0x1F0
GPIO_88	0xD401E1F4	0x1F4
GPIO_89	0xD401E1F8	0x1F8
GPIO_90	0xD401E1FC	0x1FC
GPIO_91	0xD401E200	0x200
GPIO_92	0xD401E204	0x204
GPIO_111	0xD401E20C	0x20C
GPIO_112	0xD401E210	0x210
GPIO_113	0xD401E214	0x214
GPIO_114	0xD401E218	0x218
GPIO_115	0xD401E21C	0x21C
GPIO_116	0xD401E220	0x220
GPIO_117	0xD401E224	0x224
GPIO_118	0xD401E228	0x228
GPIO_119	0xD401E22C	0x22C
GPIO_120	0xD401E230	0x230
GPIO_121	0xD401E234	0x234
GPIO_122	0xD401E238	0x238
GPIO_123	0xD401E23C	0x23C
GPIO_124	0xD401E240	0x240
GPIO_125	0xD401E244	0x244
GPIO_126	0xD401E248	0x248
GPIO_127	0xD401E24C	0x24C

3.7.1 MFPR Functional Description

3.7.1.1 I/O PAD Paramenter Definition

The input thresholds of Buffer Mode of I/O PADs are tabled below.

ST1:ST0==2'b00				
Input Threshold	**Min**	**Typ**	**Max**	**Unit**
VT	0.75	0.91	1.09	V
VT PU	0.74	0.90	1.08	V

ST1:ST0==2'b00				
Input Threshold	**Min**	**Typ**	**Max**	**Unit**
VT PD	0.76	0.92	1.10	V

Instead, the input thresholds of Schmitt Trigger Mode of I/O PADs are tabled below.

ST1:ST0==2'b01				
Input Threshold	**Min**	**Typ**	**Max**	**Unit**
VT+	0.82	0.97	1.13	V
VT-	0.72	0.85	1.02	V
VT+PU	0.81	0.96	1.12	V
VT-PU	0.71	0.84	1.01	V
VT+PD	0.82	0.98	1.14	V
VT-PD	0.73	0.86	1.03	V

ST1:ST0==2'b10/2'b11				
Input Threshold	**Min**	**Typ**	**Max**	**Unit**
VT+	0.87	1.04	1.19	V
VT-	0.69	0.80	0.95	V
VT+PU	0.86	1.03	1.18	V
VT-PU	0.68	0.79	0.94	V
VT+PD	0.88	1.05	1.20	V
VT-PD	0.69	0.81	0.96	V

3.7.1.2 MFPR Field Description

Bit(s)	Field	Type	Reset	Description
31:16	RSVD	RO	0	This field is reserved for future use
15	PULL SEL	RW	0x1	This field selects between two sets of controls for the pull-up and pull-down functionality as follows: - 0: The pull-up and pull-down resistors are controlled by the selected alternate function for the pin - 1: The pull-up and pull-down resistors are controlled by the <PULLUP

Bit(s)	Field	Type	Reset	Description
				EN> and <PULLDN EN> fields in this register, overriding the function indicated by the selected alternate function. During low-power states, this field is overridden to 1 and controlled by the <PULLUP EN> and <PULLDN EN> fields. In these low-power states, this field is effectively 1, although the register value is not changed (refer to low-power (sleep) mode operation for more information).
14	PULLUP EN	RW	0x0	This field controls the output function while the <PULL SEL> field is set to 1 (or is effectively 1) as follows: - 0: The internal pull-up resistor of the pin is disabled - 1: The internal pull-up resistor of the pin is enabled The address and reset value is on a pin-by-pin basis. Do not rely on the reset value of this field. It must be configured by software to the desired settings.
13	PULLDN EN	RW	0x0	This field controls the output function while <PULL SEL> is set to 1 (or is effectively 1) as follows: - 0: The internal pull-down resistor of the pin is disabled - 1: The internal pull-down resistor of the pin is enabled The address and reset value is on a pin-by-pin basis . Do not rely on the reset value of this field. It must be configured by software to the desired settings.
12:11	DRIVE[1:0]	RW	0x2	This field defines the drive strength and slew rate for this pin (in functional mode when the pin is driving HIGH or LOW value) as follows: - 2'b00: SLOW - 2'b01: SLOW - 2'b10: MEDIUM - 2'b11: FAST They are the DS1 and DS0 bit of the drive strength in the current table.
10	DRIVE[2]	RW	0x0	This is the DS2 bit to program for higher level of driving strength in the current table. The address and reset value is on a pin-by-pin basis. Do not rely on the reset value of this field. It must be configured by software to the desired settings. For Medium (all GPIOs except for SD card), it is 010. For Fast (SD card I/O), it is 110.
9:8	ST[1:0]	RW	0x0	This field controls the Schmitt trigger input threshold as follows: - 2'b00: buffer input, threshold is 0.9v - 2'b01/10/11: enabled the Schmitt trigger with larger hysteresis for VT- and VT+ threshold (refer to **Section 3.7.1.1**)
7	SLE	RW	0x0	This field enables/disables the slew rate output control as follows: - 1'b1: Enabled - 1'b0: Disabled Enabling the slew rate output control will slow down the output ramp for EMI considerations.

Bit(s)	Field	Type	Reset	Description
6	EDGE_CLEAR	RW	0x1	This field enable/disable the edge-detection logic as follows: - 1'b0: Enabled and ready to detect an edge - 1'b1: Disabled and no edge is detected This is an enable for the <EDGE_FALL_EN> and <EDGE_RISE_EN> control fields. This field is only present when a pin has been defined as potentially waking up on an edge. If the device is not configured in this manner, this field is not present (i.e. reserved) and writing to it has no effect (refer to **Section 3.5** for more information about which MFPRs include or not include these bits).
5	EDGE_FALL_EN	RW	0x0	This field enables/disable to detect a falling edge as follows: - 1'b0: Disabled - 1'b1: Enable To detect a falling edge on this pin, - The pin needs not be an output - This field must be set to 1 - The <EDGE_CLEAR> field must be set to 0 This field is only present when a pin has been defined as potentially waking up on an edge. If the device is not configured in this manner, this field is not present (i.e. reserved) and writing to it has no effect (refer to **Section 3.5** for more information about which MFPRs include or not include these bits).
4	EDGE_RISE_EN	RW	0x0	This field enables/disable to detect a rising edge as follows: - 1'b0: Disables - 1'b1: Enabled To detect a rising edge on this pin,, - The pin need not be an output - This field must be set to 1 - The <EDGE_CLEAR> field must be set to 0 This field is only present when a pin has been defined as potentially waking up on an edge. If the device is not configured in this manner, this field is not present (i.e. reserved). and writing to it has no effect (refer to **Section 3.5** for more information about which MFPRs include or not include these bits).
3	SPU	RW	0x0	This field enables/disables a strong pull resistor as follows: - 1'b0: Disabled - 1'b1: Enabled This field is used for I2C or SD card PADs which require a strong pull resistor.
2:0	AF SEL	RW	0x0	This field is used for the selection of an alternate function for a pin between eight possible options as follows: - 0x0: Alternate function 0 (always as the primary at reset) - 0x1: Alternate function 1 - 0x2: Alternate function 2 - 0x3: Alternate function 3 - 0x4: Alternate function 4

Bit(s)	Field	Type	Reset	Description
				- 0x5: Alternate function 5 - 0x6: Alternate function 6 - 0x7: Alternate function 7

Chapter 4

Electrical Characteristics



Key Stone® K1 User Manual

4.1 Pin AC/DC Operating Conditions

Item	Symbol/Pin	Min	Typ	Max	Unit	Note
Digital Power	VCC_M1	0.85	0.9	1.0	V	
PLL	AVDD09_PLL	0.855	0.9	0.945	V	
	AVDD18_PLL	1.71	1.8	1.89	V	
OSC	AVDD09_AFEAP	0.855	0.9	0.945	V	
	AVDD18_AFEAP	1.71	1.8	1.89	V	
PCIeC	AVDD18_PCIEC	1.71	1.8	1.89	V	
	AVDD09_PCIEC	0.855	0.9	0.945	V	
PCIeB	AVDD18_PCIEB	1.71	1.8	1.89	V	
	AVDD09_PCIEB	0.855	0.9	0.945	V	
PCIeA	AVDD18_PCIEA	1.71	1.8	1.89	V	
	AVDD09_PCIEA	0.855	0.9	0.945	V	
USB IO	AVDD33_USB	3.135	3.3	3.465	V	
USB PHY	AVDD18_USB	1.71	1.8	1.89	V	
	AVDD09_USB	0.855	0.9	0.945	V	

Item	Symbol/Pin	Min	Typ	Max	Unit	Note
MIPI DSI PHY	AVDD09_DSI1	0.855	0.9	0.945	V	
	AVDD18_DSI1	1.71	1.8	1.89	V	
MIPI DSI IO	AVDD12_DSI1	1.14	1.2	1.26	V	
MIPI CSI PHY	AVDD09_CSI	0.855	0.9	0.945	V	
	AVDD18_CSI	1.71	1.8	1.89	V	
HDMI	AVDD09_HDMI	0.855	0.9	0.945	V	
	AVDD18_HDMI	1.71	1.8	1.89	V	
	AVDD33_HDMI	3.135	3.3	3.465	V	
eMMC	VDD09_EMMC	0.855	0.9	0.945	V	
	V18_EMMC	1.71	1.8	1.89	V	
QSPI	VCC1833_QSPI	1.71	1.8	1.89	V	Dual power domain
		3.135	3.3	3.465	V	
SD	VCC1833_MMC1	1.71	1.8	1.89	V	Dual power domain
		3.135	3.3	3.465	V	
DDR PHY	AVDD18_PHY	1.71	1.8	1.89	V	
	AVDD18_DDR	1.71	1.8	1.89	V	
	AVDD11_DDR	1.045	1.1	1.155	V	LP4/4X
		1.14	1.2	1.26	V	LP3

Item	Symbol/Pin	Min	Typ	Max	Unit	Note
	AVDDU_PHY	0.855	0.9	0.945	V	
	AVDDU_DDR	0.855	0.9	0.945	V	
DDR IO	AVDD06_DDR	0.57	0.6	0.63	V	
	VDDQ_V1P2	1.14	1.2	1.26	V	
eFuse	AVDD18_EFUSE	1.71	1.8	1.89	V	
Audio Logic	AUD_VDDU09	0.855	0.9	0.945	V	
Audio Power NEG	AUD_VNEG	-1.71	-1.8	-1.89	V	
Audio Power POS	AUD_VPOS	1.71	1.8	1.89	V	
Audio Analog	AVDD18_AUD	1.71	1.8	1.89	V	
	AVDD3V3_AUD	3.135	3.3	3.465	V	
GPIO	VCC18_GPIO	1.71	1.8	1.89	V	
GIOP3	VCC1833_GPIO3	1.71	1.8	1.89	V	Dual power domain
		3.135	3.3	3.465	V	
GIOP2	VCC1833_GPIO2	1.71	1.8	1.89	V	Dual power domain
		3.135	3.3	3.465	V	

4.2 Absolute Max Ratings

4.2.1 For Pins

Item	Symbol/Pin	Min	Max	Unit
Digital Power	VCC_M1	-0.1	1.035	V
PLL	AVDD09_PLL	-0.1	1.035	V
	AVDD18_PLL	-0.1	2.07	V
OSC	AVDD09_AFEAP	-0.1	1.035	V
	AVDD18_AFEAP	-0.1	2.07	V
PCIeC	AVDD18_PCIEC	-0.1	2.07	V
	AVDD09_PCIEC	-0.1	1.035	V
PCIeB	AVDD18_PCIEB	-0.1	2.07	V
	AVDD09_PCIEB	-0.1	1.035	V
PCIeA	AVDD18_PCIEA	-0.1	2.07	V
	AVDD09_PCIEA	-0.1	1.035	V
USB IO	AVDD33_USB	-0.1	3.795	V
USB PHY	AVDD18_USB	-0.1	2.07	V
	AVDD09_USB	-0.1	1.035	V
MIPI DSI IO	AVDD12_DSI1	-0.1	1.38	V
MIPI DSI PHY	AVDD09_DSI1	-0.1	1.035	V
	AVDD18_DSI1	-0.1	2.07	V
MIPI CSI PHY	AVDD09_CSI	-0.1	1.035	V
	AVDD18_CSI	-0.1	2.07	V
HDMI	AVDD09_HDMI	-0.1	1.035	V
	AVDD18_HDMI	-0.1	2.07	V
	AVDD33_HDMI	-0.1	3.795	V
eMMC	VDD09_EMMC	-0.1	1.035	V
	V18_EMMC	-0.1	2.07	V
QSPI	VCC1833_QSPI	-0.1	2.07	V
		-0.1	3.795	V
SD	VCC1833_MMC1	-0.1	2.07	V
		-0.1	3.795	V
DDR PHY	AVDD18_PHY	-0.1	2.07	V

Item	Symbol/Pin	Min	Max	Unit
	AVDD18_DDR	-0.1	2.07	V
	AVDD11_DDR	-0.1	1.265	V
	AVDD11_DDR	-0.1	1.38	V
	AVDDU_PHY	-0.1	1.035	V
	AVDDU_DDR	-0.1	1.035	V
DDR IO	AVDD06_DDR	-0.1	0.69	V
	VDDQ_V1P2	-0.1	1.38	V
eFuse	AVDD18_EFUSE	-0.1	2.07	V
Audio Logic	AUD_VDDU09	-0.1	1.035	V
Audio Power NEG	AUD_VNEG	N/A	-2.07	V
Audio Power POS	AUD_VPOS	-0.1	2.07	V
Audio Analog	AVDD18_AUD	-0.1	2.07	V
	AVDD3V3_AUD	-0.1	3.795	V
GPIO	VCC18_GPIO	-0.1	2.07	V
GPIO3	VCC1833_GPIO3	-0.1	2.07	V
		-0.1	3.795	V
GPIO2	VCC1833_GPIO2	-0.1	2.07	V
		-0.1	3.795	V

4.2.2 For Packages

Item	Symbol	Min	Max	Unit
Operating Temperature (Industrial Standard)	Ta	-40	+85	℃
Junction Temperature	Tj	N/A	125	℃
Storage Temperature	Tstg	-40	125	℃

4.3 Pin Max Currents

Item	Symbol/Pin	Max	Unit
Digital Power	VCC_M1	10000	mA
PLL	AVDD09_PLL	5	mA

Item	Symbol/Pin	Max	Unit
	AVDD18_PLL	5	mA
OSC	AVDD09_AFEAP	5	mA
	AVDD18_AFEAP	5	mA
PCIeC	AVDD18_PCIEC	50	mA
	AVDD09_PCIEC	100	mA
PCIeB	AVDD18_PCIEB	50	mA
	AVDD09_PCIEB	100	mA
PCIeA	AVDD18_PCIEA	50	mA
	AVDD09_PCIEA	100	mA
USB IO	AVDD33_USB	90	mA
USB PHY	AVDD18_USB	90	mA
	AVDD09_USB	15	mA
MIPI DSI PHY	AVDD09_DSI1	20	mA
	AVDD18_DSI1	50	mA
MIPI DSI IO	AVDD12_DSI1	50	mA
MIPI CSI PHY	AVDD09_CSI	70	mA
	AVDD18_CSI	100	mA
HDMI	AVDD09_HDMI	10	mA
	AVDD18_HDMI	10	mA
	AVDD33_HDMI	10	mA
eMMC	VDD09_EMMC	50	mA
	V18_EMMC	50	mA
QSPI	VCC1833_QSPI	150	mA
SD	VCC1833_MMC1	150	mA
DDR PHY	AVDD18_PHY	200	mA
	AVDD18_DDR	20	mA
	AVDD11_DDR	100	mA
	AVDDU_PHY	100	mA
	AVDDU_DDR	100	mA
DDR IO	AVDD06_DDR	100	mA
	VDDQ_V1P2	600	mA
eFuse	AVDD18_EFUSE	150	mA

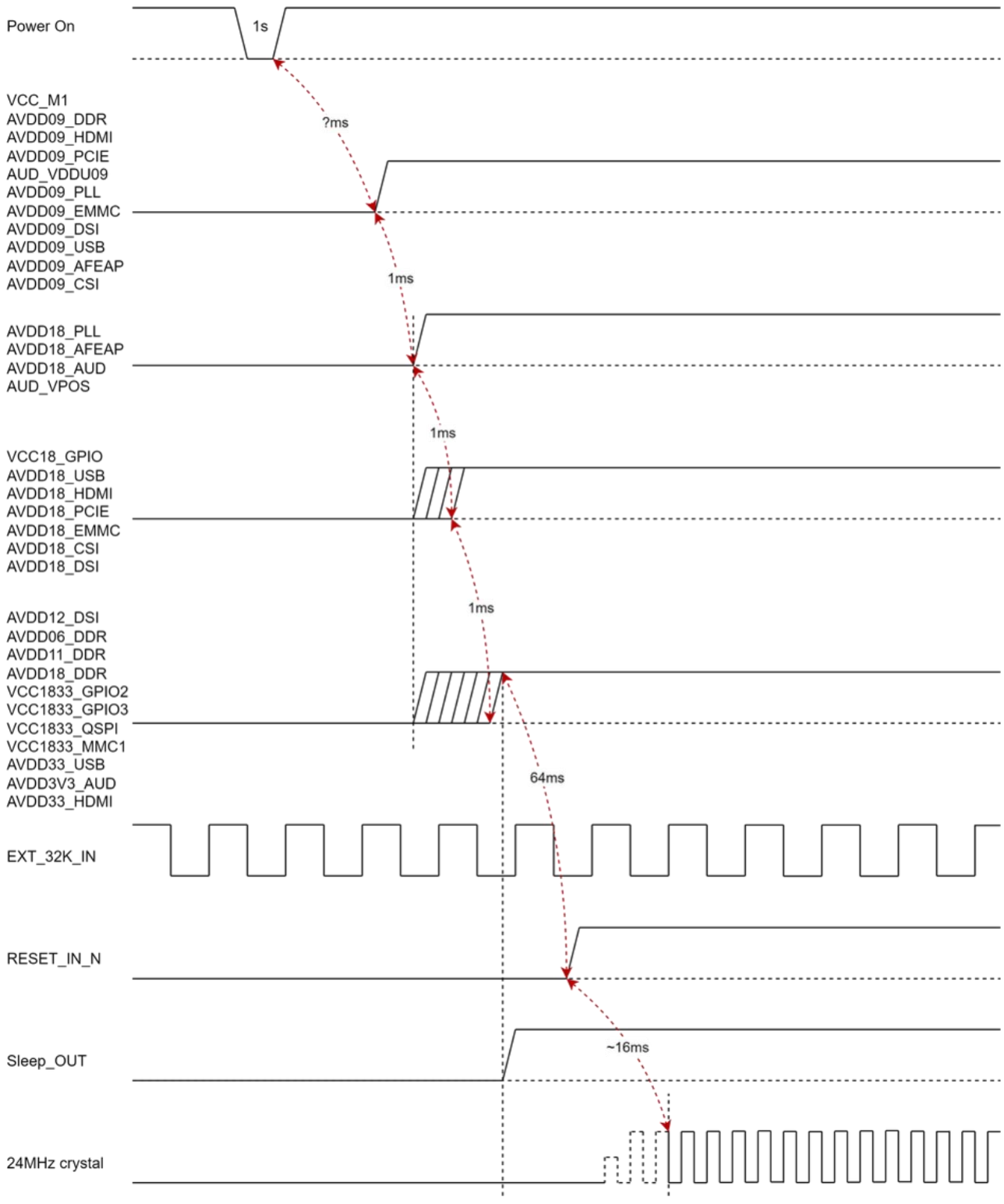
Item	Symbol/Pin	Max	Unit
Audio Logic	AUD_VDDU09	1	mA
Audio Power NEG	AUD_VNEG	102	mA
Audio Power POS	AUD_VPOS	102	mA
Audio Analog	AVDD18_AUD	10	mA
	AVDD3V3_AUD	100	mA

4.4 Power On/Off Sequence

4.4.1 Power On Sequence

- A short pressure (i.e. 1 second) of the power button will turn on the K1 processor automatically if it was off before (cold start)
- The Power Management IC (PMIC) will turn on firstly the core logic then the external I/O to ensure proper initialization
- PMIC will asserts a Power-On-Reset (POR) to initialize the system and ensure a defined starting state

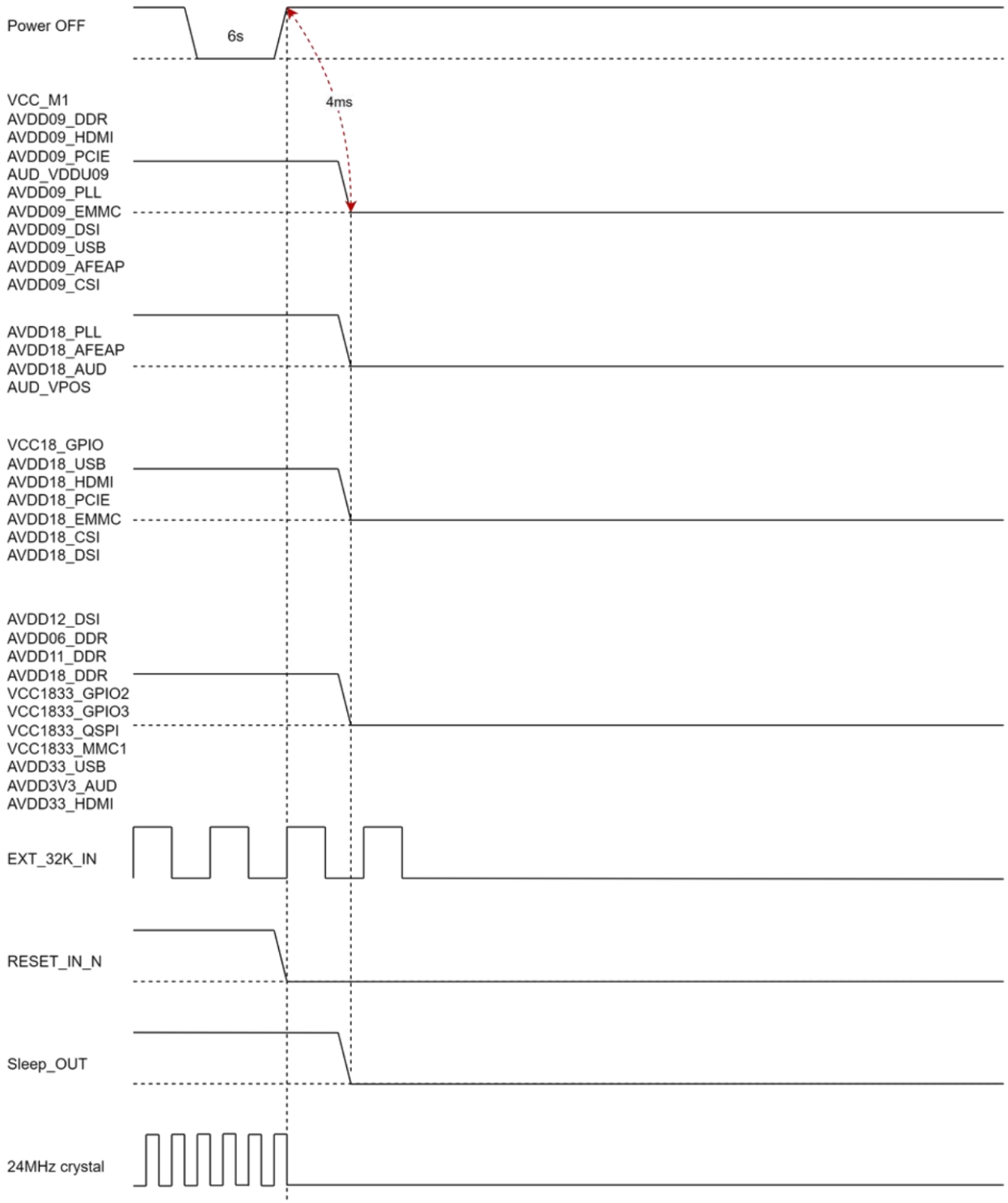
The order of the involved pins with state change during the power on sequence is depicted below.


Power On
1s
VCC_M1
AVDD09_DDR
AVDD09_HDMI
AVDD09_PCIE
AUD_VDDU09
AVDD09_PLL
AVDD09_EMMC
AVDD09_DSI
AVDD09_USB
AVDD09_AFEAP
AVDD09_CSI
?ms
AVDD18_PLL
AVDD18_AFEAP
AVDD18_AUD
AUD_VPOS
1ms
VCC18_GPIO
AVDD18_USB
AVDD18_HDMI
AVDD18_PCIE
AVDD18_EMMC
AVDD18_CSI
AVDD18_DSI
1ms
AVDD12_DSI
AVDD06_DDR
AVDD11_DDR
AVDD18_DDR
VCC1833_GPIO2
VCC1833_GPIO3
VCC1833_QSPI
VCC1833_MMC1
AVDD33_USB
AVDD3V3_AUD
AVDD33_HDMI
64ms
EXT_32K_IN
RESET_IN_N
Sleep_OUT
~16ms
24MHz crystal

4.4.2 Power Off Sequence

- A long pressure (i.e. 6 seconds) of the power button will turn off the K1 processor.

The order of the involved pins with state change during the power off sequence is depicted below.



4.5 Power Consumption

4.5.1 In Typical Application Scenario

To be defined soon.

4.5.2 In Particular Application Scenario

To be defined soon.

Chapter 5

Boot Modes



Key Stone® K1 User Manual

5.1 Introduction

K1 supports booting from

- SPI NAND Flash
- SPI NOR Flash
- eMMC
- SD/TF Card

5.2 Boot Strapping Pins

The boot process of K1 is controlled by a set of boot strapping pins which can be configured to determine the boot mode and other operational parameters. The definitions of the boot strapping pins are tabled below.

Pin Name	Boot Strapping Pin	Default Value
QSPI_DAT0	STRAP[0]	0
QSPI_DAT1	STRAP[1]	0
QSPI_DAT2	STRAP[2]	0
QSPI_DAT3	STRAP[3]	0
GPIO90	STRAP[4]	0

Details about the purpose of the different configurations of the boot strapping pins are provided in the following subsections.

5.2.1 Boot Mode Selection

The boot mode is selected basing on the state of **STRAP[0]** and **STRAP[1]** pins as tabled below.

No.	STRAP[1]	STRAP[0]	Boot Mode
1	Down	Down	SD/TF Card -> eMMC (default)
2	Up	Down	SD/TF Card -> SPI NAND Flash
3	Down	Up	SD/TF Card -> SPI NOR Flash
4	Up	Up	SD/TF Card

5.2.2 Download Mode Selection

The download mode is selected basing on the state of the **STRAP[2]** pin as tabled below**.**

No.	STRAP[2]	Download Mode
1	Down	USB (default)
2	Up	UART

5.2.3 Boot Download Mode Selection

The boot download mode is selected basing on the state of the **STRAP[3]** pin as tabled below**.**

No.	STRAP[3]	Boot Download Mode
1	Down	Boot Mode as per **Section 5.2.1** (default)
2	Up	Download Mode as per **Section 5.2.2**

5.2.4 SPI NAND/NOR Flash Boot Voltage Selection

The boot voltage of SPI NAND/NOR Flash is selected basing on the state of the **STRAP[4]** pin as tabled below.

No.	STRAP[4]	SPI NAND/NOR Flash Boot Voltage
1	Up	3.3V I/O
2	Down	1.8V I/O

Chapter 6

Address Mapping



Key Stone® K1 User Manual

6.1 Introduction

K1 includes

- One SpacemiT® X60™ RISC-V Main CPU Domain
- One RISC-V Real-Time CPU Domain

The different SoC address mappings from each CPU perspective are provided in the following sections.

6.2 Main CPU Domain Address Mapping

Module	Address	Size	Note
DRAM_0	0x0000_0000	0x8000_0000	
PCIe PortA Data	0x8000_0000	0x1000_0000	
PCIe PortB Data	0x9000_0000	0x1000_0000	
PCIe PortC Data	0xA000_0000	0x1800_0000	
QSPI	0xB800_0000	0x0800_0000	
DDRPHY Config	0xC000_0000	0x0010_0000	
V2D	0xC010_0000	0x0010_0000	
ASR ISP	0xC020_0000	0x000f_0000	
ASR CPP	0xC02f_0000	0x0001_0000	
ASR LCD DSI	0xC030_0000	0x0010_0000	
FBC-DEC0	0xC040_0000	0x0000_0100	
FBC-DEC1	0xC040_0100	0x0000_0100	
ASTC-DEC	0xC040_0200	0x0000_0100	
LCD_TOP	0xC040_0300	0x0000_0100	
LCD_MMU	0xC040_0400	0x0000_0100	
Reserved	0xC040_0500	0x0003_FB00	
Saturn	0xC044_0000	0x0004_0000	
VPU	0xC050_0000	0x0008_0000	
Reserved	0xC058_0000	0x0028_0000	
AUDIO SRAM	0xC080_0000	0x0004_0000	
Reserved	0xC084_0000	0x0004_0000	
AUDIO Peripherals	0xC088_0000	0x0008_0000	
USB20 Controller	0xC090_0000	0x0004_0000	
USB20 PHY	0xC094_0000	0x0004_0000	
USB20 Host Controller	0xC098_0000	0x0004_0000	
USB20 Host PHY	0xC09C_0000	0x0004_0000	

Module	Address	Size	Note
USB30	0xC0A0_0000	0x0010_0000	
PCIe PortA PHY	0xC0B0_0000	0x0010_0000	
PCIe PortB PHY	0xC0C0_0000	0x0010_0000	
PCIe PortC PHY	0xC0D0_0000	0x0010_0000	
Reserved	0xC0E0_0000	0x0920_0000	
PCIe PortA Config	0xCA00_0000	0x0040_0000	
PCIe PortB Config	0xCA40_0000	0x0040_0000	
PCIe PortC Config	0xCA80_0000	0x0040_0000	
GPU	0xCAC0_0000	0x0008_0000	
GMAC0	0xCAC8_0000	0x0000_1000	
GMAC1	0xCAC8_1000	0x0000_1000	
Reserved	0xCAC8_2000	0x0937_E000	
PDMA Controller	0xD400_0000	0x0001_0000	
RTC	0xD401_0000	0x0000_0800	
IIC0	0xD401_0800	0x0000_0800	
IIC1	0xD401_1000	0x0000_0800	
OneWire	0xD401_1800	0x0000_0800	
IIC2	0xD401_2000	0x0000_0800	
IIC4	0xD401_2800	0x0000_0800	
DRO	0xD401_3000	0x0000_0400	
IPC (mailbox)	0xD401_3400	0x0000_0400	Real CPU to Main X60™ CPU
IIC5	0xD401_3800	0x0000_0800	
Timer1	0xD401_4000	0x0000_1000	
APB Bus Clock Unit	0xD401_5000	0x0000_1000	
Timer2	0xD401_6000	0x0000_1000	
UART0	0xD401_7000	0x0000_0100	
UART2	0xD401_7100	0x0000_0100	
UART3	0xD401_7200	0x0000_0100	
UART4	0xD401_7300	0x0000_0100	
UART5	0xD401_7400	0x0000_0100	
UART6	0xD401_7500	0x0000_0100	
UART7	0xD401_7600	0x0000_0100	
UART8	0xD401_7700	0x0000_0100	
UART9	0xD401_7800	0x0000_0100	
Reserved	0xD401_7900	0x0000_0600	

Module	Address	Size	Note
IR	0xD401_7F00	0x0000_0100	
Tsensor	0xD401_8000	0x0000_0800	
IIC6	0xD401_8800	0x0000_0800	
GPIO	0xD401_9000	0x0000_0800	
GPIO Edge	0xD401_9800	0x0000_0800	
PWM0	0xD401_A000	0x0000_0400	
PWM1	0xD401_A400	0x0000_0400	
PWM2	0xD401_A800	0x0000_0400	
PWM3	0xD401_AC00	0x0000_0400	
PWM4	0xD401_B000	0x0000_0400	
PWM5	0xD401_B400	0x0000_0400	
PWM6	0xD401_B800	0x0000_0400	
PWM7	0xD401_BC00	0x0000_0400	
SSP3(SPI)	0xD401_C000	0x0000_2000	
IIC7	0xD401_D000	0x0000_0800	
IIC8	0xD401_D800	0x0000_0800	
Pad Configuration（Pinmux)	0xD401_E000	0x0000_0C00	
PWM8	0xD402_0000	0x0000_0400	
PWM9	0xD402_0400	0x0000_0400	
PWM10	0xD402_0800	0x0000_0400	
PWM11	0xD402_0C00	0x0000_0400	
PWM12	0xD402_1000	0x0000_0400	
PWM13	0xD402_1400	0x0000_0400	
PWM14	0xD402_1800	0x0000_0400	
PWM15	0xD402_1C00	0x0000_0400	
PWM16	0xD402_2000	0x0000_0400	
PWM17	0xD402_2400	0x0000_0400	
PWM18	0xD402_2800	0x0000_0400	
PWM19	0xD402_2C00	0x0000_0400	
Reserved	0xD402_3000	0x0000_2000	
SSPA0 (I2S0)	0xD402_6000	0x0000_0800	
SSPA1 (I2S1)	0xD402_6800	0x0000_0800	
Reserved	0xD402_7000	0x0000_1000	
CAN0	0xD402_8000	0x0000_0400	
Reserved	0xD402_8400	0x0002_7C00	

Module	Address	Size	Note
Main PMU (NDR)	0xD405_0000	0x0001_0000	
Reserved	0xD406_0000	0x0002_0000	
PMU Timer & WDT (NDR)	0xD408_0000	0x0001_0000	
Extra Logic (NDR)	0xD409_0000	0x0001_0000	
Reserved	0xD40A_0000	0x0001_0000	
Resource IPC (NDR)	0xD40B_0000	0x0001_0000	
Reserved	0xD40C_0000	0x0014_6000	
IPE3(mipi-csi)	0xD420_6000	0x0000_0800	
Reserved	0xD420_6800	0x0000_2800	
SPI_LCD	0xD420_9000	0x0000_1000	
IPE1 (mipi-csi)	0xD420_A000	0x0000_0800	
IPE2 (mipi-csi)	0xD420_A800	0x0000_0800	
Reserved	0xD420_B000	0x0000_1000	
QSPI reg	0xD420_C000	0x0000_3000	
ISP MMU	0xD420_F000	0x0000_1000	
Reserved	0xD421_0000	0x0000_A800	
LCD_DSI	0xD421_A800	0x0000_5800	
Reserved	0xD427_F000	0x0000_1000	
SDH1	0xD428_0000	0x0000_0800	
SDH2	0xD428_0800	0x0000_0800	
SDH3	0xD428_1000	0x0000_1000	
AP PMU	0xD428_2800	0x0000_0100	
Reserved	0xD428_2900	0x0000_0300	
CPU Config Unit	0xD428_2C00	0x0000_0400	
Reserved	0xD428_3000	0x00D7_E000	
Reserved	0xD500_3000	0x02FF_D000	
RSICV TCM(512KB)	0xD800_0000	0x0008_0000	
Reserved	0xD808_0000	0x003C_0000	
CIU Dragon	0xD844_0000	0x000C_0000	CIUDRAGON regsheet
CCI500	0xD850_0000	0x0010_0000	
REE（AES Engine）	0xD860_0000	0x0010_0000	
Reserved	0xD870_0000	0x0790_0000	
RISCV_APB	0xE000_0000	0x1000_0000	
Secure DDRC Config	0xF000_0000	0x0030_0000	
Secure LCD_DSI	0xF030_0000	0x0010_0000	

Module	Address	Size	Note
Secure LCD_HDMI	0xF040_0000	0x0010_0000	
Secure VPU	0xF050_0000	0x0008_0000	
Secure Configuration Unit	0xF058_0000	0x0008_0000	
Secure DMA Controller Config	0xF060_0000	0x0001_0000	
SEC APB2 Bus Clock Unit	0xF061_0000	0x0000_2000	
SEC UART1	0xF061_2000	0x0000_1000	
SEC SSP2 (SPI)	0xF061_3000	0x0000_1000	
SEC_IIC3	0xF061_4000	0x0000_1000	
SEC_RTC	0xF061_5000	0x0000_1000	
SEC_Timer 0	0xF061_6000	0x0000_1000	
SEC_Keypad Controller	0xF061_7000	0x0000_1000	
SEC_JTAG Software	0xF061_8000	0x0000_1000	
SEC_GPIO	0xF061_9000	0x000E_7000	
Secure BCM config（Crypto)	0xF070_0000	0x0000_3800	
Reserved	0xF800_0000	0x07E0_0000	
ROM	0xFFE0_0000	0x0020_0000	
DRAM_1	0x1_0000_0000	0x3_8000_0000	

6.3 Real-Time CPU Domain Address Mapping

Module	Address	Size	Note
SRAM	0x0000_0000	0x0004_0000	
Reserved	0x0004_0000	0x2FFC_0000	
DDR	0x3000_0000	0x1000_0000	
Reserved	0x4000_0000	0x8087_0000	
R CAN	0xC087_0000	0x0000_4000	
Reserved	0xC087_4000	0x0000_4000	
sys_ctrl_regs	0xC088_0000	0x0000_1000	
SHUB_UART0	0xC088_1000	0x0000_1000	
Audio_ctrl_reg	0xC088_2000	0x0000_1000	
CODEC ADMA	0xC088_3000	0x0000_0100	
CODEC SSPA	0xC088_3100	0x0000_0300	
I2S1 ADMA	0xC088_3400	0x0000_0100	
I2S1 SSPA	0xC088_3500	0x0000_0300	
HDMI ADMA	0xC088_3800	0x0000_0100	

Module	Address	Size	Note
HDMI SSPA	0xC088_3900	0x0000_0300	
I2S0 ADMA	0xC088_3C00	0x0000_0100	
I2S0 SSPA	0xC088_3D00	0x0000_0300	
AHBDMA	0xC088_4000	0x0000_1000	
SHUB_SSP0 (SPI)	0xC088_5000	0x0000_1000	
SHUB_SSP1 (SPI)	0xC088_6000	0x0000_1000	
SHUB_I2C0	0xC088_7000	0x0000_1000	
SHUB_PWM0-9	0xC088_8000	0x0000_1000	
AON _TIMER	0xC088_9000	0x0000_1000	
AON_IPC2AP	0xC088_A000	0x0000_1000	
Reserved	0xC088_B000	0x0000_0800	
Reserved	0xC088_B800	0x0000_0800	
AON _PMU_REG	0xC088_C000	0x0000_0800	
Reserved	0xC088_C800	0x0000_0800	
SHUB_UART1	0xC088_D000	0x0000_1000	
R_IR_RX	0xC088_E000	0x0000_1000	
Reserved	0xC089_0000	0x0004_0000	
Audio Buffer	0xC08D_0000	0x0000_4000	
Reserved	0xC08D_4000	0x1372_C000	
To AP APB	0xD400_0000	0x0040_0000	

Chapter 7

Interrupt Assignments



Key Stone® K1 User Manual

7.1 Introduction

K1 contains

- One Processor Core Local Interrupt Controller (**CLINT**)
- One Platform Level Interrupt Controller (**PLIC**)

CLINT is a single functional module embedded in the **SpacemiT®X60™ RISC-V Main CPU**. It is responsible for centralizing all Main CPU interrupt sources before dispatching them to **each individual Main CPU hart.**

PLIC, instead, is a single functional module embedded in the **Sensor-Hub Subsystem**. It is responsible for centralizing all Sensor-Hub Subsystem interrupt sources before dispatching them to the **RISC-V Real-Time CPU**.

7.2 Main CPU Interrupts

Interrupt ID	Interrupt Source
sys_int_ap[16]	CAN_0_INT
sys_int_ap[17]	CAN_1_INT
sys_int_ap[18]	I2C_7_int
sys_int_ap[19]	I2C_8_int
sys_int_ap[20]	**Reserved**
sys_int_ap[21]	rtc_hzclk_int_ndr
sys_int_ap[22]	rtc_slp_alarm_ndr
sys_int_ap[23]	timer0_1_irq
sys_int_ap[24]	timer0_2_irq
sys_int_ap[25]	timer0_3_irq
sys_int_ap[26]	timer1_1_irq
sys_int_ap[27]	timer1_2_irq
sys_int_ap[28]	timer1_3_irq
sys_int_ap[29]	new_timer_1_irq
sys_int_ap[30]	new_timer_2_irq
sys_int_ap[31]	new_timer_3_irq
sys_int_ap[32]	ndr_timer_1_irq
sys_int_ap[33]	ndr_timer_2_irq
sys_int_ap[34]	ndr_timer_3_irq
sys_int_ap[35]	wdt_irq
sys_int_ap[36]	I2C_0_int
sys_int_ap[37]	I2C_1_int

Interrupt ID	Interrupt Source
sys_int_ap[38]	I2C_2_int
sys_int_ap[39]	I2C_3_int
sys_int_ap[40]	I2C_4_int
sys_int_ap[41]	I2C_5_int
sys_int_ap[42]	UART0_int
sys_int_ap[43]	UART1_int
sys_int_ap[44]	UART2_int
sys_int_ap[45]	UART3_int
sys_int_ap[46]	UART4_int
sys_int_ap[47]	UART5_int
sys_int_ap[48]	UART6_int
sys_int_ap[49]	UART7_int
sys_int_ap[50]	UART8_int
sys_int_ap[51]	UART9_int
sys_int_ap[52]	ipc_adsp2ap_int_comb
sys_int_ap[53]	ripc0_int0
sys_int_ap[54]	ssp2_int_req(SSP2(0xF0613000))
sys_int_ap[55]	ssp3_int_req(SSP3(0xD401C000))
sys_int_ap[56]	ssp4_int_req(SSPA0(0xD4026000))
sys_int_ap[57]	ssp5_int_req(SSPA1(0xD4026800))
sys_int_ap[58]	GPIO_INT_AP
sys_int_ap[59]	GPIO_INT_AP_SEC
sys_int_ap[60]	gpio_edge_det_wakeup
sys_int_ap[61]	tsen_int
sys_int_ap[62]	onewire_int
sys_int_ap[63]	m1_kp_int | m1_keypad_wakeup
sys_int_ap[64]	PMIC_INT_edge_detected
sys_int_ap[65]	pm_xsc_wakeup_int_mux
sys_int_ap[66]	sec_rtc_hzclk_int_ndr
sys_int_ap[67]	sec_rtc_slp_alarm_ndr
sys_int_ap[68]	dro_int
sys_int_ap[69]	irc_int

Interrupt ID	Interrupt Source
sys_int_ap[70]	I2C_6_int
sys_int_ap[71]	**Reserved**
sys_int_ap[72]	dma_int_ap_non_sec
sys_int_ap[73]	dma_int_ap_sec
sys_int_ap[74]	vpu_irq
sys_int_ap[75]	gpu_irqgpu
sys_int_ap[76]	gpu_irqjob
sys_int_ap[77]	gpu_irqevent
sys_int_ap[78]	gpu_irqmmu
sys_int_ap[79]	isp_irq
sys_int_ap[80]	isp_mmu_irq
sys_int_ap[81]	ipe_irq
sys_int_ap[82]	ipe2_irq
sys_int_ap[83]	ipe3_irq
sys_int_ap[84]	cpp_irq
sys_int_ap[85]	isp_dma_irq
sys_int_ap[86]	v2d_irq
sys_int_ap[87]	jpg_irq
sys_int_ap[88]	dpu_offl1_int_sat
sys_int_ap[89]	dpu_offl0_int_sat
sys_int_ap[90]	dpu_onl2_int_sat
sys_int_ap[91]	lcd_de_irq
sys_int_ap[92]	RSVD for future AP
sys_int_ap[93]	lcd_sec_irq
sys_int_ap[94]	dmmu_irq
sys_int_ap[95]	dsi_irq_a
sys_int_ap[96]	ad_irq
sys_int_ap[97]	aud_wakeup
sys_int_ap[98]	RSVD for future AP
sys_int_ap[99]	mmc1_int
sys_int_ap[100]	mmc2_int
sys_int_ap[101]	mmc3_int

Interrupt ID	Interrupt Source
sys_int_ap[102]	sdh2icu_wakeup_int1
sys_int_ap[103]	sdh2icu_wakeup_int2
sys_int_ap[104]	sdh2icu_wakeup_int3
sys_int_ap[105]	usb_int
sys_int_ap[106]	usb_vbus_id_wakeup
sys_int_ap[107]	aeu_int
sys_int_ap[108]	fabric0_timeout
sys_int_ap[109]	ddr_arm_int
sys_int_ap[110]	wtm_hst_int_out
sys_int_ap[111]	wtm_sp_int_out
sys_int_ap[112]	pmu_irq2
sys_int_ap[113]	bcm0_sp_int_out|bcm0_hst_int_out
sys_int_ap[114]	aud_wdt_irq
sys_int_ap[115]	mc_irq
sys_int_ap[116]	ddrphy_irq
sys_int_ap[117]	qspi_int
sys_int_ap[118]	usbp1_int
sys_int_ap[119]	RSVD for future AP
sys_int_ap[120]	arb_timeout1_int_level
sys_int_ap[121]	apb_mon_int
sys_int_ap[122]	mc_irq_scy
sys_int_ap[123]	aclk_off_fab_int
sys_int_ap[124]	**Reserved**
sys_int_ap[125]	usb3_interrupt
sys_int_ap[126]	hook_key_int
sys_int_ap[127]	aud_plug_int
sys_int_ap[128]	sm2_pmdone_int
sys_int_ap[129]	sm2_padone_int
sys_int_ap[130]	lcd_spi_irq
sys_int_ap[131]	emac0_intr
sys_int_ap[132]	emac0_wkuprcvd|emac0_phy_int
sys_int_ap[133]	emac1_intr

Interrupt ID	Interrupt Source
sys_int_ap[134]	emac1_wkuprcvd|emac1_phy_int
sys_int_ap[135]	hdmi_irq[0]
sys_int_ap[136]	hdmi_irq[1]
sys_int_ap[137]	hdmi_dpu_offl1_int_sat
sys_int_ap[138]	hdmi_dpu_offl0_int_sat
sys_int_ap[139]	hdmi_dpu_onl2_int_sat
sys_int_ap[140]	hdmi_sec_irq
sys_int_ap[141]	pcie_porta_interrupt
sys_int_ap[142]	pcie_portb_interrupt
sys_int_ap[143]	pcie_portc_interrupt
sys_int_ap[144]	CCI_nERRORIRQ_ap
sys_int_ap[145]	pcie_porta_wakeup
sys_int_ap[146]	pcie_portb_wakeup
sys_int_ap[147]	pcie_portc_wakeup
sys_int_ap[148]	usbp1_vbus_id_wakeup
sys_int_ap[149]	usb3_vbus_id_wakeup
sys_int_ap[150]	dma2_int_ap_non_sec
sys_int_ap[151]	dma2_int_ap_sec

7.3 Sensor-Hub Subsystem Interrupts

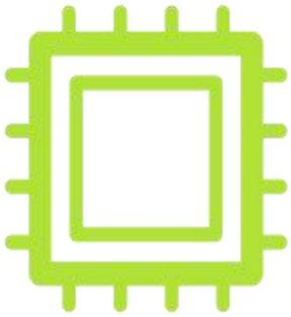
Interrupt ID	Interrupt Source
rcpu_int[0]	Dmac_int
rcpu_int[1]	**Reserved**
rcpu_int[2]	ap_wakeup_audio_int
rcpu_int[3]	voice_dection_int
rcpu_int[4]	rsm_irq
rcpu_int[5]	adma_ch0_int
rcpu_int[6]	adma_ch1_int
rcpu_int[7]	dfe_int | sspa_int
rcpu_int[8]	timer_1_irq
rcpu_int[9]	timer_2_irq
rcpu_int[10]	timer_3_irq

Interrupt ID	Interrupt Source
rcpu_int[11]	ipc_ap2aud_int
rcpu_int[12]	**Reserved**
rcpu_int[13]	shubi2c0_int
rcpu_int[14]	shubi2c1_int
rcpu_int[15]	shubssp0_int
rcpu_int[16]	shubssp1_int
rcpu_int[17]	uart_int_req
rcpu_int[18]	aud_ocp_int | classg_shortpwr_int
rcpu_int[19]	hook_key_int
rcpu_int[20]	aud_plug_int
rcpu_int[21]	IPC_MSA2AUD
rcpu_int[22]	aud_ap_wakeup
rcpu_int[23]	sensor_hub_irpc_int
rcpu_int[24]	GPIO_0_INT or pmic_int
rcpu_int[25]	GPIO_1_INT
rcpu_int[26]	GPIO_2_INT
rcpu_int[27]	GPIO_3_INT
rcpu_int[28]	GPIO_4_INT
rcpu_int[29]	GPIO_5_INT
rcpu_int[30]	GPIO_6_INT
rcpu_int[31]	GPIO_7_INT
rcpu_int[32]	**Reserved**
rcpu_int[33]	adma1_ch1_int
rcpu_int[34]	Adma1_ch0_int
rcpu_int[35]	sspa1_int
rcpu_int[36]	hdmi_adma_ch_int
rcpu_int[37]	**Reserved**
rcpu_int[38]	adma0_ch1_int
rcpu_int[39]	adma0_ch0_int
rcpu_int[40]	sspa0_int
rcpu_int[41]	can0_int0
rcpu_int[42]	can0_int1

Interrupt ID	Interrupt Source
rcpu_int[43]	irc_int
rcpu_int[44]	uart1_int_req
rcpu_int[45]	shub_edge_det_int

Chapter 8

CPU System



Key Stone® K1 User Manual

8.1 Overview

The CPU System ok K1 consists of

- A SpacemiT®X60™ RISC-V Main CPU (hereafter **CPU Subsystem**)
- A RISC-V Real-Time CPU (hereafter **RCPU**)

8.2 CPU Subsystem

8.2.1 Introduction

The **CPU Subsystem** of K1 includes the following components:

- Two X60™ clusters
- A Cache Coherency Unit
- A Dragon CIU for configuration
- Additional components to ensure high performance and system efficiency

8.2.2 Features

- **X60™ Clusters**
  - Each cluster contains four cores with 512KB L2 cache
  - The BOOST cluster also includes 512KB TCM
  - Each X60™ core has 32KB L1-I cache and 32KB L1-D cache
- **Cache Coherency Unit**
  - Maintenance of data coherency between X60™ clusters
- **Dragon CIU Registers**
  - Availability of multiple configuration registers for X60™ clusters and cores
- **Generic Counter Register**
  - SoC Counter Registers

8.2.3 Functional Description

Reserved

8.2.4 Generic Counter Register Description

Starting from the **base address 0xD500_1000**, there are several **Generic Counter Registers** which configuration details are provided in the following subsections

8.2.4.1 CNTCR_REG

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved for future use
1	HDBG	RW	0x0	Halt on debug. The possible values are: - 0: HLTDBG signal into the counter has no effect - 1: HLTDBG signal into the counter halts the counter
0	EN	RW	0x0	Enable/Disable counter. The possible values are: - 0: The counter is disabled and not incrementing - 1: The counter is enabled and is incrementing

8.2.4.2 CNTSR_REG

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved for future use
1	HDBG	RW	0x0	Debug halted
0	Reserved	RO	0x0	Reserved for future use

8.2.4.3 CNTCVLW_REG

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CNTCVLW	RW	0x0	Value of counter [31:0]

8.2.4.4 CNTCVUP_REG

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CNTCVUP	RW	0x0	Value of counter [63:32]

8.2.4.5 CNTFID_REG

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	FREQ	RW	0x0	Frequency in number of ticks per second

8.2.5 Dragon CIU Registers Description

Starting from the **base address is 0xD844_0000**, there are many **Dragon CIU Registers** which configuration details are provided in the following subsections

8.2.5.1 CLUSTER0_CPU_SRAM_CTRL0_REG

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CLUSTER0_MEM_EMA_CFG_31_0	RW	0x00035102	MEM_EMA_CFG_31_0 for Cluster0 CPU memory

8.2.5.2 CLUSTER0_CPU_SRAM_CTRL1_REG

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CLUSTER0_MEM_EMA_CFG_63_32	RW	0x59454000	MEM_EMA_CFG_63_32 for Cluster0 CPU memory

8.2.5.3 CLUSTER1_CPU_SRAM_CTRL0_REG

Offset: 0x54				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CLUSTER1_MEM_EMA_CFG_31_0	RW	0x00035102	MEM_EMA_CFG_31_0 for Cluster1 CPU memory

8.2.5.4 CLUSTER1_CPU_SRAM_CTRL1_REG

Offset: 0x58				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CLUSTER1_MEM_EMA_CFG_63_32	RW	0x59454000	MEM_EMA_CFG_63_32 for Cluster1 CPU memory

8.2.5.5 CKG_CTRL_REG

Offset: 0x88				
Bits	**Field**	**Type**	**Reset**	**Description**
31:26	Reserved	RO	0x0	Reserved for future use
25:16	CCI dynamic clock gate timer	RW	0x0	When dynamic clock gating is enabled, the CCI clock turns off after the number of cycles set by **CCI_CLKOFF_TIMER**
15:4	Reserved	RO	0x0	Reserved for future use
3	Dragon CIU dynamic gate enable	RW	0x0	- 0: Clock is always on - 1: Dynamic clock gating enabled
2	CCI clock dynamic gate enable	RW	0x0	- 0: Clock is always on - 1: Dynamic clock gating enabled
1	Reserved	RO	0x0	Reserved for future use
0	nic400_cci550_2x5 clock dynamic gate enable	RW	0x0	- 0: Clock is always on - 1: Dynamic clock gating enabled

8.2.5.6 CCI_DBG_CTRL_REG

Offset: 0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0x0	Reserved for future use
3	Invasive debug enable	RW	0x0	If HIGH, the counting and export of PMU events are enabled
2	Secure invasive debug enable	RW	0x0	If both SPIDEN and DBGEN are HIGH, the counting of both Non-Secure and Secure events are enabled
1	Non-Invasive debug enable	RW	0x0	If HIGH, the counting and export of PMU events are enabled
0	Secure privileged non-invasive	RW	0x0	If both SPNIDEN and NIDEN are HIGH, the

Offset: 0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
	debug enable			counting of both Non-Secure and Secure events are enabled

8.2.5.7 CCI_INF_QOS_CTRL_REG

Offset: 0x98				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15:12	C1_ARQOS_CFG	RW	0x0	ALZO_ARQOS configuration
11:8	C1_AWQOS_CFG	RW	0x0	ALZO_AWQOS configuration
7:4	C0_ARQOS_CFG	RW	0x0	CPU ARQOS configuration
3:0	C0_AWQOS_CFG	RW	0x0	CPU AWQOS configuration

8.2.5.8 CCI_SFRAM_CTL_REG

Offset: 0x9C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	MEM_EMA_CFG_31_0	RW	0x00035102	MEM_EMA_CFG_31_0 for CCI CPU memory

8.2.5.9 X60_OUTER_COH_EN_CTRL_REG

Offset: 0xF0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0	Reserved for future use
1	C1_cpu_outer_coh_en	RW	0x1	0: Cluster 1 outer cache coherency disabled 1: Cluster 1 outer cache coherency enabled
0	C0_cpu_outer_coh_en	RW	0x1	0: Cluster 1 outer cache coherency disabled 1: Cluster 0 outer cache coherency enabled

8.2.5.10 FAB_TOM_RD_STS0_REG

Offset: 0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Ranking number of AXI read to Monitor 1	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
27:24	Ranking number of AXI read to Monitor 0	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
23:20	Ranking number of AXI Write to Monitor 4	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
19:16	Ranking number of AXI Write to Monitor 3	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
15:12	Ranking number of AXI Write to Monitor 2	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
11:8	Ranking number of AXI Write to Monitor 1	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
7:4	Ranking number of AXI Write to Monitor 0	RO	0x0	0: No timeout occurred, write clear 1: Timeout occurred
3:0	Total timeout number	RO	0x0	It Indicates the timeout status of various ranks as follows: - RANK4: s7_sec - RANK3: s5_axidec2 - RANK2: s4_audio - RANK1: s0_mc - RANK0: s2_axidec **Note.** 0 indicates no timeout. Write clear is required before using this register and fab_tom_wr_sts

8.2.5.11 FAB_TOM_RD_STS1_REG

Offset: 0x11C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	Reserved	RO	0	Reserved for future use
11:8	Ranking number of AXI Read to Monitor 4	RO	0x0	0: No timeout, write clear by FAB_TOM_RD_STS 1: Timeout occurred
7:4	Ranking number of AXI Read to Monitor 3	RO	0x0	0: No timeout, write clear by FAB_TOM_RD_STS 1: Timeout occurred

Offset: 0x11C				
Bits	**Field**	**Type**	**Reset**	**Description**
3:0	Ranking number of AXI Read to Monitor 2	RO	0x0	0: No timeout, write clear by FAB_TOM_RD_STS 1: Timeout occurred

8.2.5.12 FAB_TIMEOUT_CTRL_X_REG

This register controls the timeout for the AXI bus monitor on the Fabric#1 S2 port.

If there is a pending request on the AXI Fabric#1 S2 port and the request duration exceeds the timeout value, a timeout interrupt will be asserted. In particular,

- The interrupt can be de-asserted by writing a “0” to TIMEOUT_VAL
- The default timeout value is 0xC000 where one unit represents one AXI clock cycle

Notes.

- Only the Fabric#1 S2 port is monitored, and this feature is intended for debugging purposes
- To ensure a proper access to the registers when a timeout occurs on the Fabric#1 S2 port, these timeout control registers should be placed alongside the SQU registers on the Fabric#1 S4 port

Offset: 0x120~0x160（0x10）				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Timeout auto response type	RW	0x0	- 0: Auto response type is SLVERR - 1: Auto response type is OKAY
30	Timeout auto response enable	RW	0x0	- 0: Disable auto response when fabric timeout happens - 1: Enable auto response when fabric timeout happens (this can help taking CPU out of a hanging state)
29	Timeout interrupt mask	RW	0x0	This bit determines whether a timeout interrupt is triggered when a timeout event occurs, in particular: - 0: Interrupt is not masked (an interrupt will be asserted when the timeout event happens) - 1: Interrupt is masked (no interrupt will be asserted when the timeout event happens)
28	Fabric monitor reset	RW	0x0	- 0: Reset fabric monitor - 1: Release fabric monitor
27	Fabric monitor register clear	RW	0x0	Clear the information saved when the latest timeout occurred, in particular: - 0: The relevant registers are kept - 1: The relevant registers are cleared
26	Timeout interrupt clear	RW	0x0	- 0: No effect - 1: Auto-clear (by hardware) AXI fabric S2 port timeout

Offset: 0x120~0x160（0x10）				
Bits	**Field**	**Type**	**Reset**	**Description**
				interrupt
25	no_ready_check_en	RW	0x0	Enable/Disable checking AWREADY or ARREADY signals as follows: 0: Disable 1: Enable
24	new_feature_en	RW	0x0	Enable/Disable the AXI fabric timeout monitor new feature as follows: 0: Disable 1: Enable
23:16	Reserved	RO	0	Reserved for future use
15	Clock gating disable	RW	0x0	[DOVE_A0] Disable monitor clock gating, in particular: - 1: Clock runs freely (i.e. clock gating for monitor disabled) - 0: Clock is gated if FAB_MON_RST==0
14:0	AXI fabric timeout value	RW	0xC000	This field indicates the timeout value for AXI fabric. The real timeout time is 2048*TIMEOUT_VAL. One unit represents one AXI clock cycle.

8.2.5.13 FAB_TIMEOUT_ID_X_REG

This register provides information gathered by AXI bus monitor on the Fabric#1 S2 port, and it is updated when the first timeout event occurs.

Offset: 0x124~0x164（0x10）				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Write timeout indicator	RO	0x0	- 0: No write request timeout - 1: Write request timeout occurs
30:22	Reserved	RO	0	Reserved for future use
21:16	Timeout transaction WID	RO	0x0	It is the Write ID (WID) of a Write Transaction that triggers the first timeout event, and is valid only when bit[31], WR_TIMEOUT_IND is asserted
15	Read timeout indicator	RO	0x0	- 0: No read request timeout - 1: Read request timeout occurs
14:6	Reserved	RO	0	Reserved for future use
5:0	Timeout transaction RID	RO	0x0	It is the Read ID (RID) of a Read Transaction that triggers the first timeout event, and is valid only when bit[15], RD_TIMEOUT_IND is asserted

8.2.5.14 FAB_TIMEOUT_STATUS0_X_REG

This register provides information gathered by AXI bus monitor on Fabric#1 S2 port, and it is updated when the first timeout event occurs.

Offset: 0x128~0x168（0x10）				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Write timeout Address	RO	0x0	The write address being accessed during the first write timeout event
1	Reserved	RO	0	Reserved for future use
0	Write timeout Indicator	RO	0x0	- 0: No write request timeout occurred - 1: Write request timeout occurred

8.2.5.15 FAB_TIMEOUT_STATUS1_X_REG

This register provides information gathered by AXI bus monitor on Fabric#1 S2 port, and is updated when the first timeout event occurs.

Offset: 0x12C~0x16C（0x10）				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Read timeout address	RO	0x0	It is the address being accessed during the first read timeout event
1	Reserved	RO	0	Reserved for future use
0	Read timeout indicator	RO	0x0	- 0: No read request timeout occurred - 1: Read request timeout occurred

8.2.5.16 FAB_DEBUG_REG_X_REG

Offset: 0x170~0x17C（0x4）				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	AXI fabric debug register	RO	0x0	- [0] = debug_axifab_cp_dec - [1] = debug_axifab_axi_dec2 - [2] = debug_axifab_csap_fab - [3] = debug_axifab_vpu_lte - [4] = debug_axifab_isp_lcd - [5] = debug_axifab_cp_mfab - [6] = debug_axifab_apfab - [7] = 32'h0 - [8] = debug_axifab_cpu_dec - [9] = debug_axifab_cpu_gpu

Offset: 0x170~0x17C（0x4）				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. debug_axifab = mas_bready, mas_rready, slv_awready, slv_arready, slv_wready, clk_axi_en_nd

8.2.5.17 PAD_PIC_PLIC_INT_0_CTL_REG

Offset: Reserved				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pad_pic_plic_int_0_ctl_	RW	0x0	This register controls the interrupts for [31:0]. For each bit, - 0: Level-triggered interrupt - 1: Edge-triggered interrupt

8.2.5.18 PAD_PIC_PLIC_INT_1_CTL_REG

Offset: Reserved				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pad_pic_plic_int_1_ctl_	RW	0x0	This register controls the interrupts for [63:32]. For each bit: - 0: Level-triggered interrupt - 1: Edge-triggered interrupt

8.2.5.19 PAD_PIC_PLIC_INT_2_CTL_REG

Offset: Reserved				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pad_pic_plic_int_2_ctl_	RW	0x0	This register controls the interrupts for [95:64]. For each bit: - 0: Level-triggered interrupt - 1: Edge-triggered interrupt

8.2.5.20 PAD_PIC_PLIC_INT_3_CTL_REG

Offset: Reserved				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pad_pic_plic_int_3_ctl_	RW	0x0	This register controls the interrupts for [127:96]. For each bit: - 0: Level-triggered interrupt

Offset: Reserved				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1: Edge-triggered interrupt

8.2.5.21 PAD_PIC_PLIC_INT_4_CTL_REG

Offset: Reserved				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pad_pic_plic_int_4_ctl_	RW	0x0	This register controls the interrupts for [159:128]. For each bit: - 0: Level-triggered interrupt - 1: Edge-triggered interrupt

8.2.5.22 C0_INT_WAKEUP_MASK_REG

Offset: 0x1A4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0	Reserved for future use
9	C0_ms_int_core_mask	RW	0x0	Mask for C0_ms_int to core: - 1: Mask - 0: Unmask
8	C0_mt_int_core_mask	RW	0x0	Mask for C0_mt_int to core: - 1: Mask - 0: Unmask
7	C0_ss_int_core_mask	RW	0x0	Mask for C0_ss_int to core: - 1: Mask - 0: Unmask
6	C0_me_int_core_mask	RW	0x0	Mask for C0_me_int to core: - 1: Mask - 0: Unmask
5	C0_se_int_core_mask	RW	0x0	Mask for C0_se_int to core: - 1: Mask - 0: Unmask
4	C0_ms_int_pmu_wakeup_mask	RW	0x0	Mask for C0_ms_int to PMU wakeup: - 1: Mask - 0: Unmask
3	C0_mt_int_pmu_wakeup_mask	RW	0x0	Mask for C0_mt_int to PMU wakeup: - 1: Mask - 0: Unmask

Offset: 0x1A4				
Bits	**Field**	**Type**	**Reset**	**Description**
2	C0_ss_int_pmu_wakeup_mask	RW	0x0	Mask for C0_ss_int to PMU wakeup: - 1: Mask - 0: Unmask
1	C0_me_int_pmu_wakeup_mask	RW	0x0	Mask for C0_me_int to PMU wakeup: - 1: Mask - 0: Unmask
0	C0_se_int_pmu_wakeup_mask	RW	0x0	Mask for C0_se_int to PMU wakeup: - 1: Mask - 0: Unmask

8.2.5.23 C1_INT_WAKEUP_MASK_REG

Offset: 0x1A8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0	Reserved for future use
9	C1_ms_int_core_mask	RW	0x0	Mask for C1_ms_int to core: - 1: Mask - 0: Unmask
8	C1_mt_int_core_mask	RW	0x0	Mask for C1_mt_int to core: - 1: Mask - 0: Unmask
7	C1_ss_int_core_mask	RW	0x0	Mask for C1_ss_int to core: - 1: Mask - 0: Unmask
6	C1_me_int_core_mask	RW	0x0	Mask for C1_me_int to core: - 1: Mask - 0: Unmask
5	C1_se_int_core_mask	RW	0x0	Mask for C1_se_int to core: - 1: Mask - 0: Unmask
4	C1_ms_int_pmu_wakeup_mask	RW	0x0	Mask for C1_ms_int to PMU wakeup: - 1: Mask - 0: Unmask
3	C1_mt_int_pmu_wakeup_mask	RW	0x0	Mask for C1_mt_int to PMU wakeup: - 1: Mask - 0: Unmask
2	C1_ss_int_pmu_wakeup_mask	RW	0x0	Mask for C1_ss_int to PMU wakeup: - 1: Mask

Offset: 0x1A8				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 0: Unmask
1	C1_me_int_pmu_wakeup_mask	RW	0x0	Mask for C1_me_int to PMU wakeup: - 1: Mask - 0: Unmask
0	C1_se_int_pmu_wakeup_mask	RW	0x0	Mask for C1_se_int to PMU wakeup: - 1: Mask - 0: Unmask

8.2.5.24 CPU_SYS_SW_RESET_REG

Offset: 0x1AC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0	Reserved for future use
1	hap_top_sw_reset	RW	0x0	Perform a software reset on hap_top by first writing 1, then writing 0
0	pic_top_sw_reset	RW	0x0	Perform a software reset on pic_top by first writing 1, then writing 0

8.2.5.25 C0_L2_FLUSH_CTL_RESET_REG

Offset: 0x1B0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	Reserved	RO	0	Reserved for future use
4	C0_l2_hw_flush_done_mask	RW	0x0	- 0: L2 flush completion status not masked - 1: L2 flush completion status masked (i.e. C0_l2_flush_done_status =1)
3	C0_l2_flush_done_status	RO	0x0	Status of the L2 flush operation: - 0: L2 flush not completed - 1: L2 flush completed
2	C0_l2_flush_hw_en	RO	0x0	Enable of hardware L2 flush: - 0: Hardware L2 flush disabled - 1: Hardware L2 flush enabled (only valid if C0_l2_flush_ctl[0] is 1)
1	C0_l2_flush_sw_req	RW	0x0	Software request for L2 flush: - 0: Not request a software-initiated L2 flush - 1: Request a software-initiated L2 flush (clear this bit (i.e. set to 0) after completed the flush (when C0_l2_flush_done_status=1))

Offset: 0x1B0				
Bits	**Field**	**Type**	**Reset**	**Description**
0	C0_l2_flush_type	RW	0x0	- 1: Hardware PMU mode - 0: Software mode

8.2.5.26 C1_L2_FLUSH_CTL_RESET_REG

Offset: 0x1B4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	Reserved	RO	0	Reserved for future use
4	C1_l2_hw_flush_done_mask	RW	0x0	- 0: L2 flush completion status not masked - 1: L2 flush completion status masked (i.e. C1_l2_flush_done_status =1)
3	C1_l2_flush_done_status	RO	0x0	Status of the L2 flush operation: - 0: L2 flush not completed - 1: L2 flush completed
2	C1_l2_flush_hw_en	RO	0x0	Enable/Disable hardware L2 flush as follows: - 0: Disabled - 1: Enabled (only valid if C1_l2_flush_ctl[0] is 1)
1	C1_l2_flush_sw_req	RW	0x0	Software request for L2 flush: - 0: Not request a software-initiated L2 flush - 1: Request a software-initiated L2 flush (clear this bit (i.e. set to 0) after completed the flush (when C1_l2_flush_done_status is 1))
0	C1_l2_flush_type	RW	0x0	- 1: Hardware PMU mode - 0: Software mode

8.2.5.27 CPU_ACCESS_DDR_REMAP_EN_RESET_REG

Offset: 0x1B8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0	Reserved for future use
0	cpu_access_ddr_remap_en_	RW	0x1	- 0: DDR remap disabled - 1: DDR remap enabled

8.2.5.28 HAP_DM_CTL_REG

Offset: 0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	Reserved	RO	0	Reserved for future use
4	C01_hap_dm_ndmreset_n_mask	RO	0x0	Enable/Disable preventing hap_sys_apb_rst_n and power-on cluster's sys_apb_rst_n from being set to 0 during SoC WDT reset caused by hap_dm_ndmreset_n as follow: - 1: Enabled - 0: Disabled
3	core_iso_hold_en	RW	0x0	Enable/Disable holding the isolation of cores 1 to 7 during the SoC WDT reset caused by hap_dm_ndmreset_n as follows: - 1: Enabled - 0: Disabled
2	C1_pwr_hold_en	RW	0x0	Enable/Disable holding the power to cluster 1 during the SoC WDT reset caused by hap_dm_ndmreset_n as follows: - 1: Enabled - 0: Disabled
1	hap_dm_ndmreset_clr	RW	0x0	Used to clear C01_hap_dm_ndmreset_n_mask = 1 (from hap_dm_ctl[4]) as per the following steps: - First, write 1 to this bit - Then, write 0 to this bit
0	hap_dm_ndmreset_n_dis	RW	0x1	Enable/Disable the SoC WDT reset caused by hap_dm_ndmreset_n as follows: - 0: Enabled - 1: Disabled

8.3 Real-Time CPU

8.3.1 Introduction

The **Real-Time CPU (RCPU)** of K1 is a **32-bit RISC-V N308 High-Efficiency Processor Core** designed by **Nuclei System Technology Co. Ltd.**, and is used for IoT and low-power applications.

8.3.2 Features

- Support for 3-stage pipeline, single-issue micro-architecture
- Support for RV32I/M/A/F/D/C/P/B instruction extensions
- Support for privileged mode
- Support for low power management as follows:
  - WFI and WFE schemes to enter sleep mode

- Two-level sleep modes: shallow & deep

- Support for 16KB D-Cache
- Support for a 64-bit Real-Time Core-Private Timer (TIMER)
  - Support for an Enhanced Core-Level Interrupt Controller (ECLIC) which
  - Support for RISC-V architecturally defined software, timer and external interrupts
  - Support for configurable interrupt levels and priorities
  - Support for vectored interrupt processing mode
- Support for Non-Maskable Interrupt (NMI)
- Support for 8 PMP entries with 4KB minimal granularity
- Support for RISC-V debug framework

8.3.3 Functional Description

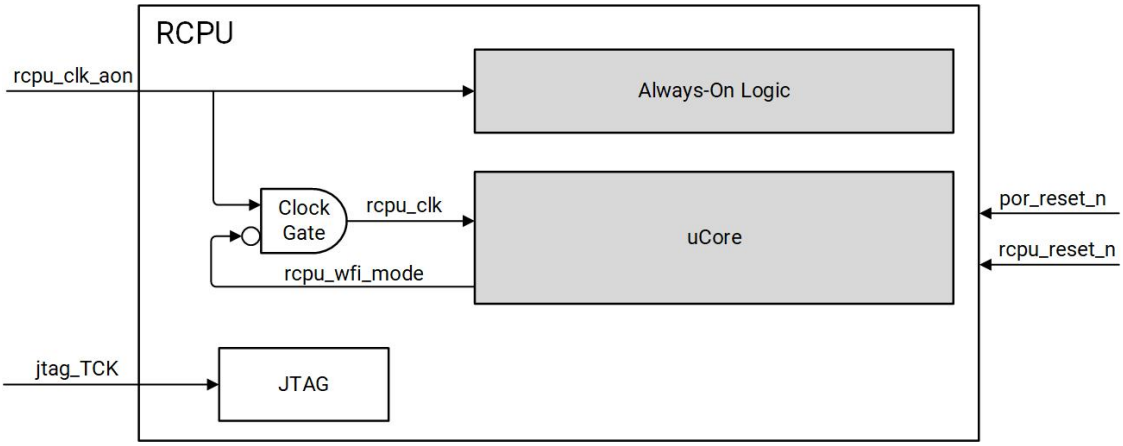
8.3.3.1 Clock Domains

The RCPU core has two asynchronous clock domains as follows:

- **Main clock domain**
  It drives most of the functional logic in RCPU, and consists of
  - The **rcpu_clk** clock as the primary working clock that drives the main domain inside RCPU, and is automatically clock-gated during sleep mode
  - The **rcpu_clk_aon** clock as an always-on clock that drives the always-on logic in RCPU, including **DEBUG**, **ECLIC**, **TIMER**
- **JTAG clock domain**
  It drives the JTAG logic of the DEBUG unit, and consist of
  - The **tck_s** clock created inside JTAG dividing by 3 the **jtag_TCK** clock

As said, both clock domains are asynchronous and an **asynchronous cross-clock domain process** has been implemented within RCPU to ensure proper operations.

The RCPU clock domains are depicted below.



8.3.3.2 Private Peripherals

In addition to the uCore, the RCPU hierarchy includes the following private peripherals:

- A handler of the JTAG interface and related debugging features (**DEBUG)**
- An Enhanced Core-Level Interrupt Controller (**ECLIC**)
- A Real-Time Core-Private Timer (**TIMER**)

Details are provided in the following subsections.

8.3.3.2.1 Address Spaces

Each private peripheral has its own address space as tabled below.

Private Peripherals	Address Space Size	Offset	Comments
DEBUG	64KB	0x000~0xFFF	This address space is used for debugging functionality, regular application software should NOT access it.
ECLIC	64KB	0x0000~0xFFFF	
TIMER	64KB	0x000~0xFFF	

8.3.3.2.2 TIMER

8.3.3.2.2.1 Introduction

It is used to generate the timer and software interrupts in the RCPU core. The configuration details of TIMER registers are provided in **Section 8.3.4.3.** Its key functionality are provided in the following sections.

8.3.3.2.2.2 Time counting

TIMER implements a 64-bit register (**mtime**) consisting of the registers

- **mtime_hi (high)**
- **mtime_lo (low)**

for counting time. In particular, **mtime** is turned on by default and continues counting time after the reset de-assertion. The increase of its frequency is controlled by

- Either the core's input signal **mtime_toggle_a**
- Or the core's always-on clock **rcpu_aon_clk**

8.3.3.2.2.3 Timer interrupts generation

TIMER implements a 64-bit register (**mtimecmp**) consisting of the registers

- **mtimecmp_hi (high)**
- **mtimecmp_lo (low)**

for comparing the value of **mtime** with the value of **mtimecmp** to generate or not a timer interrupt. In particular,

- If **mtimectl.CMPCLREN = 1**, then **mtime** will be automatically cleared to zero when its value is greater than the value of **mtimecmp**, and then the timer will restart from zero.
- If **mtimectl.CMPCLREN = 0**, then **mtime** will continue to increase normally. The software can clear the timer interrupt by modifying either the value of **mtimecmp** or **mtime**, in order to ensure that the value of **mtimecmp** is greater than the value of **mtime**.

Note. The timer interrupt is connected to ECLIC for unified interrupt management.

8.3.3.2.2.4 Software interrupts generation

TIMER implements a register (**msip**) to generate software interrupts. With reference to its configuration details tabled in **Section 8.3.4.3.9,** only the least significant bit (LSB) is effective for this purpose, which

- Is written to **1** by the software to generate a software interrupt
- Is written to **0** by the software to clear a software interrupt

Note. The software interrupt is connected to **ECLIC** as unified interrupt management.

8.3.3.2.2.5 Software-Reset request generation

TIMER implement a register **msftrst** to generate a software-reset request. With reference to its configuration details provided later in **Section 8.3.4.3.7**, only the most significant bit (MSB) is effective for this purpose, which

- Is written to **0x80000a5f** to generate a software-reset request (this specific value is used to avoid the random mis-operation of software writing)
- Can be clear only by a reset (when the SoC resets the core in response to a software-reset request, **msftrst** register will be reset and cleared to zero)

Note. The core's output signal **sysrstreq** (active high) is used to carry out the software-reset request, and is response to it, the SoC should reset the core by asserting **rcpu_reset_n** instead of **por_reset_n**.

8.3.3.2.3 ECLIC

8.3.3.2.3.1 Introduction

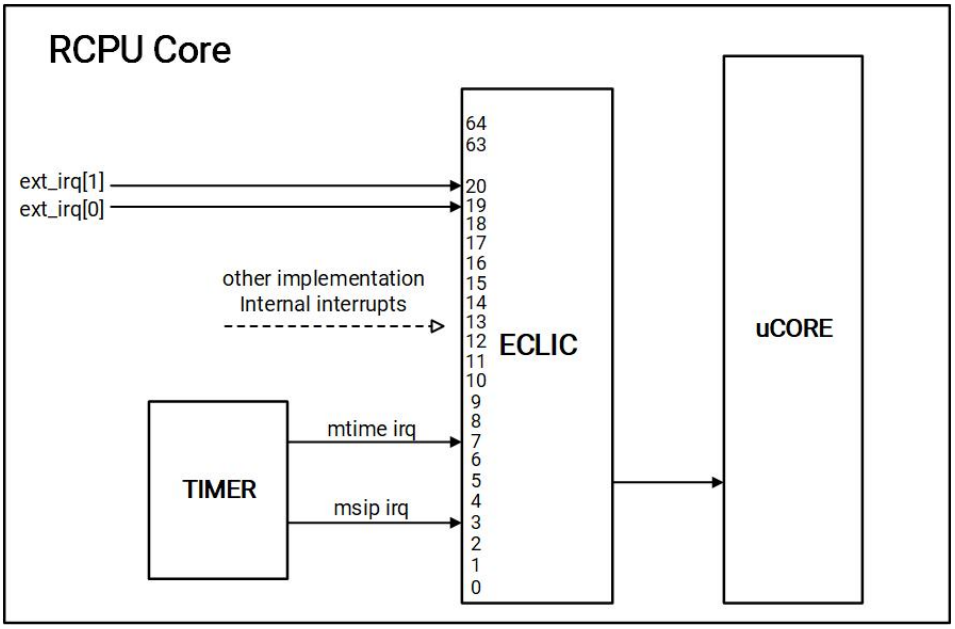
For real-time or microcontroller applications, the fast interrupt handling scheme is very important. To address this, RCPU includes an **Enhanced Core Local Interrupt Controller (ECLIC)**, which is the optimized version of the RISC-V standard CLIC, to manage all interrupt sources. To be highlighted:

- ECLIC only serves one core and privately operates inside the core
- To enable ECLIC, set the LSB bit of CSR registers **mtvec** as ECLIC mode
- ECLIC is functionally exclusive to PLIC
- ECLIC is used to manage multiple internal and external interrupts, send interrupt requests to core, and support interrupt preemption.

The configuration details of ECLIC registers are provided in **Section 8.3.4.4.** Its key functionality are provided in the following sections.

8.3.3.2.3.2 Interrupt target

When ECLIC is enabled, it arbitrates interrupt sources and sends them to the processor core (the interrupt target) through a dedicated line as depicted below



8.3.3.2.3.3 Interrupt Source ID

ECLIC assigns a unique ID to each interrupt source for supporting up to 64 interrupts, where

- Interrupt ID from 0 to 18 are reserved for the core-specified internal interrupts. Among these, ID 3 & 7 are fixed for software interrupt and timer interrupt respectively.
- Interrupt ID greater than 18 can be used by user to connect external interrupt sources.

Details are tabled below.

ECLIC Interrupt ID	Function	ECLIC **Interrupt Source Description**
0	Reserved	This source is not used
1	Reserved	This source is not used
2	Reserved	This source is not used
3	Software interrupt	Software interrupt generated by TIMER
4	Reserved	This source is not used
5	Reserved	This source is not used
6	Reserved	This source is not used
7	Timer interrupt	Timer interrupt generated by TIMER
8	Reserved	This source is not used
9	Reserved	This source is not used
10	Reserved	This source is not used
11	Reserved	This source is not used
12	Reserved	This source is not used
13	Reserved	This source is not used
14	Reserved	This source is not used
15	Reserved	This source is not used
16	Reserved	This source is not used
17	Reserved	This source is not used
18	Reserved	This source is not used
19~63	External interrupt	Normal external interrupt defined by user. **Note.** Although ECLIC can support up to 4096 interrupt sources under programming mode, the actual number of supported interrupt sources is indicated in the field **clicinfo.NUM_INTERRUPT**

8.3.3.2.3.4 Interrupt Level & Priority

As depicted in **Section 8.3.3.2.3.2,** each interrupt source of **ECLIC** can be configured with specified level and priority. The key points are as follows:

- The **clicintctl** register of each interrupt source is 8-bit width, and the number of effective bits actually implemented by the hardware is specified by the **CLICINTCTLBITS** field in the **clicinfo** register.

[Example] If the value of the clicinfo.CLICINTCTLBITS field is 6, it means that only the upper 6-bit ofthe clicintctl register are true valid bits, and the lowest 2 bits are tied to 1, as depicted below.

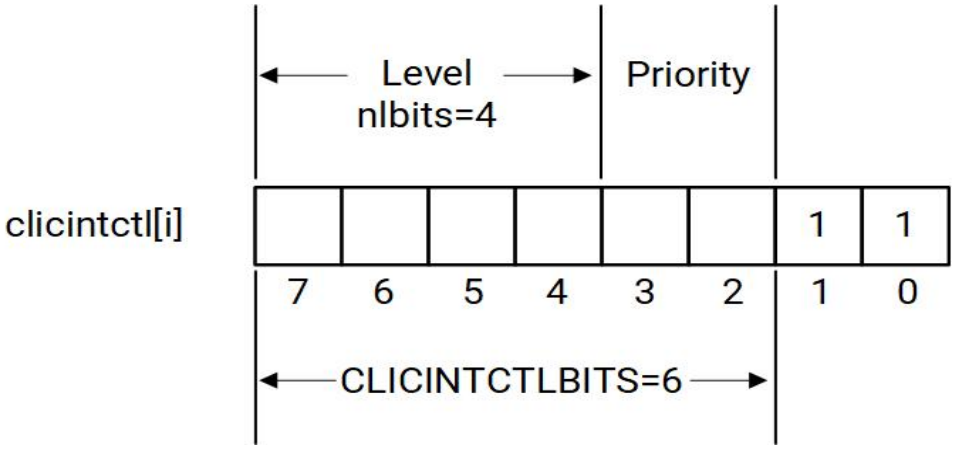
Note. The **CLICINTCTLBITS** field is a constant readable value, and software cannot overwrite it. Its valid value range is **2 ≤ CLICINTCTLBITS ≤ 8**.

- The effective bits of the **clicintctl** register are divided into
  - One dynamic field for the **interrupt level**
  - One dynamic field for the **interrupt priority**

  The width of the level field is defined by **cliccfg.nlbits**.

  [Example] If the value of **cliccfg.nlbits** is 4, it means that the upper 4-bit of the effective bits in **clicintctl** represent the level field, while the remaining lower effective bits represent the priority field, as depicted below.

  Note. The **cliccfg.nlbits** field is both readable and writeable, which means the software can modify its value.



About **interrupt level:**

- The value of level is read in a left-aligned manner. Except the effective bits, defined by the value of **cliccfg.nlbits**, the lower ineffective bits are all filled with the constant **1**, in particular
  - If **cliccfg.nlbits > clicinfo.CLICINTCTLBITS**, the number of bits indicated by **nlbits** exceeds the effective bits of the **clicintctl** register, and the excess bits are filled with constant **1**
  - If **cliccfg.nlbits = 0**, the value of level will be regarded as a fixed value of **255**
  - A greater value of level corresponds to higher priority
- **Interrupt preemption**: Higher-level interrupts can preempt lower-level interrupts. When multiple interrupts are pending (**IP = 1**), **ECLIC** performs arbitration to determine which interrupt needs to be sent to the core. The level of each interrupt source is a key factor in this arbitration process.
  An example of how the level value is decoded is depicted below.

#nlbits	Encoding	The value of level
1	L.....	127, 255
2	LL.....	127, 191, 255
3	LLL.....	31, 63, 95, 127, 159, 191, 233, 255
4	LLLL....	15,31,47,63,79,95,111,127,143,159,175,191,207,233,239,255

"L" indicates the field of level

"." indicated the rest bits and are filled with the constant 1

Instead, about **interrupt priority:**

- The priority value is also read in a left-aligned manner. Except for the effective bits, defined by **clicinfo.CLICINTCTLBITS - cliccfg.nlbits**, the lower ineffective bits are filled with the constant **1**, in particular
  - A greater value of priority corresponds to higher priority

  Note. The priority of the interrupt does not influence interrupt preemption, which means whether an interrupt can preempt another interrupt is independent of its priority value.

  When multiple interrupts are simultaneously pending, **ECLIC** needs to perform arbitration to determine which interrupt can be sent to the core. This arbitration needs to consider both the value of level and the value of priority of each interrupt source.

8.3.3.3 Control & Status Registers (CSRs)

Some Control & Status Registers (CSRs) are defined in the RISC-V architecture to configure or record the status of execution, where

- CSRs are internal registers within the core which uses a 12-bit encoding space to access their values
- Extended CSR registers are defined for **D-Cache Control and Maintenance (CCM)** operations, including
  - FLUSH
  - INVAL
  - LOCK
  - UNLOCK

These operations can be performed on specific **addresses** (ADDR) or applied **globally** (ALL).

8.3.3.4 Performance Monitor

The RISC-V architecture defines two performance counters as follows:

- **Cycle Counter**:
  - It is a 64-bit wide clock cycle counter that tracks the number of clock cycles the core has executed. This counter continuously increases as long as the core's **rcpu_clk_aon** is on.
  - The CSR **mcycle** reflects the lower 32 bits of this counter, while the CSR **mcycleh**_ _reflects the upper 32 bits.

- **Instruction Retirement Counter**:
  - It is a 64-bit counter that tracks the number of instructions the core has executed successfully. The counter will increase if the processor executes an instruction successfully.
  - The CSR **minstret** reflects the lower 32 bits of this counter, while the CSR **minstreth** reflects the upper 32 bits.

Both the **Cycle Counter** and **Instruction Retirement Counter** are typically used to measure performance.

By default, the counter value starts from zero after a reset and increases continuously. In order to save power, an extra bit in the customized CSR **mcountinhibi**t can pause the counter to save power when performance monitoring is not needed.

8.3.3.5 Low-Power Mechanism

The low-power mechanism of the RCPU core operates as follows:

- The clocks of the core's main units are automatically gated off when they are idle for reducing static power consumption.
- The core supports different sleep modes (**shallow sleep mode** or **deep sleep mode**) through **WFI (Wait For Interrupt)** and **WFE (Wait For Event)** mechanisms to achieve lower dynamic and static power consumption.

8.3.3.6 Clock Control

Two cases are distinguished:

- For the case that the core is waiting for an interrupt to wake up, because the core can only handle the interrupt processed and distributed by **ECLIC**, then only the interrupt, which is enabled and has a greater interrupt level than the interrupt threshold level, can wake up the core. Furthermore, whether to enable the **rcpu_clk_aon** inside the core needs to be handled carefully:
  - **TIMER** is clocked by **rcpu_clk_aon**, so if the SoC system has disabled the always-on clock **rcpu_clk_aon**, **TIMER** cannot generate timer or software interrupt because it has no working clock, and the core cannnot be woken up.
  - **ECLIC** is clocked by **rcpu_clk_aon**, so if the SoC system has disabled the always-on clock **rcpu_clk_aon**, then the external interrupt signal must keep asserted until the SoC enables the **rcpu_clk_aon** again. Otherwise, **ECLIC** cannot sample the external interrupt signal because it has no working clock, and the core cannot be woken up.
- For the case that the core is waiting for an event or NMI to wake up, if the core sampled (by the **rcpu_clk_aon**) the input signal **rx_evt** (which indicates there is one event) or the input signal **nmi** (which indicates there is one NMI), the core will be woken up. Furthermore, whether to enable the **rcpu_clk_aon** inside the core needs to be handled carefully:
  - If the SoC system has disabled the always-on clock **rcpu_clk_aon**, then the input signal **rx_evt** or **nmi** must be kept as 1 until the SoC turns on the clock **rcpu_aon_clk**. Otherwise, the core cannot sample the **rx_evt** or **nmi** as the sample logic has no working clock, and the core will not wake up.

8.3.3.7 Physical Memory Protection

Since the RCPU is a low-power core designed for micro-controllers, it does not support the Memory Management Unit (MMU), so all the address access operations use physical addresses. In order to perform memory access protection and isolation according to memory physical address and execution privilege mode, the RISC-V standard architecture defines a physical memory protection (PMP) mechanism.

RCPU is configured to support

- 8 PMP entries, each with 4KB minimal granularity
- The PMP TOR (Top of Range) mode

8.3.4 Register Description

8.3.4.1 Core Control & Status Register (CSRs)

8.3.4.1.1 mnvec

mnvec is a customized CSR holding the NMI entry address. When an NMI is triggered while the processor is running a program, the program is forced to jump into a new PC address, which is determined by the value of **mnvec.** And the value of **mnvec** register is controlled by the **mmisc_ctl** register.

8.3.4.1.2 msubm

msubm holds:

- The current machine sub-mode
- The machine sub-mode before the current trap

Offset: 0x07C4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	N/A	N/A	Reserved, tied to 0
9:8	PTYP	RW	0x0	Machine sub-mode before entering the trap: - 0: Normal Machine Mode - 1: Interrupt Handling Mode - 2: Exception Handling Mode - 3: NMI Handling Mode
7:6	TYP	RW	0x0	Current sub-mode: - 0: Normal Machine Mode - 1: Interrupt Handling Mode - 2: Exception Handling Mode - 3: NMI Handling Mode
5:0	Reserved	N/A	N/A	Reserved, ties to 0.

8.3.4.1.3 mdcause

mdcause is used to further record the exception difference.

Offset: 0x07C9				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	N/A	N/A	Reserved, tied to 0
2:0	PTYP	RW	0x0	Further record the detailed information about the exception. When mcause.EXCCODE=1 (instruction access fault): - 0: Reserved - 1: PMP permission violation - 2: Bus error - 3-7: Reserved When mcause.EXCCODE=5 (load access fault): - 0: Reserved - 1: PMP permission violation bus error - 2: Bus error caused by core memory read - 4-7: Reserved When mcause.EXCCODE=7 (store/AMO access fault): - 0: Reserved - 1: PMP permission violation bus error - 2: Bus error caused by core memory write - 3-7: Reserved

8.3.4.1.4 mcache_ct

mcache_ctl is used to control the I-Cache and D-Cache features.

Offset: 0x07CA				
Bits	**Field**	**Type**	**Reset**	**Description**
31:21	Reserved	N/A	N/A	Reserved, tied to 0.
20:17	Reserved	N/A	N/A	Reserved, tied to 0.
16	DC_EN	RW	0x0	D-Cache enable control: - 0: D-Cache disabled (default) - 1: D-Cache enabled **Note.** When D-Cache is disabled in runtime, it just bypasses D-Cache. Users should writeback and invalidate all D-Cache before disabling it.
15:1	Reserved	N/A	N/A	Reserved, tied to 0.
0	IC_EN	RW	0x0	I-Cache enable control: - 0: I-Cache disable (default) - 1: I-Cache enable

Offset: 0x07CA				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. When I-Cache is disabled in runtime, it just bypasses I-Cache. Users should writeback and invalidate all I-Cache before disabling it.

8.3.4.1.5 mmisc_ctl

mmisc_ctl is used to control many micro-architecture implementation-related features.

Offset: 0x07D0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:13	Reserved	N/A	N/A	Reserved, tied to 0.
12	LDSPEC_ENABLE	RW	0x0	Enable or disable the load speculative going to memory interface: - 0: Disable - 1: Enable
11	SIJUMP_ENABLE	RW	0x0	Control the SIJUMP mode of trace: - 0: SIJUMP mode is off - 1: SIJUMP mode is on
10	IMRETURN_ENABLE	RW	0x0	Control the IMRETURN mode of trace: - 0: IMRETURN mode is off - 1: IMRETURN mode is on
9	NMI_CAUSE_FFF	RW	0x0	Control mnvec and mcause.EXCCODE of NMI: - 0: The value of nmvec equals the PC address after reset, mcause.EXCCODE of NMI is 0x1 - 1: The value of mnvec is the same as the value of mtvec, mcause.EXCCODE of NMI is 0xfff
8	CORE_BUS_ERR	RW	0x0	Control the bus error caused by core is exception or interrupt: - 0: Core bus error caused by core is an exception - 1: Core bus error caused by core is an interrupt
7	Reserved	N/A	N/A	Reserved, tied to 0.
6	UNALGN_ENABLE	RW	0x0	Enable or disable misaligned load/store access: - 0: Disable, then accessing misaligned memory locations will trigger an Address Misaligned exception: - 1: Enable **Note.** This field only takes effect on load/store specified in I/F/D extension. For load/store specified in A extension, misaligned accesses always trigger an address misaligned exception.
5:4	Reserved	N/A	N/A	Reserved, tied to 0.

Offset: 0x07D0				
Bits	**Field**	**Type**	**Reset**	**Description**
3	BPU_ENABLE	RW	0x1	Enable or disable the BPU Unit: - 0: Disable - 1: Enable **Note.** If BPU is disabled by software, all branches are predicted to jump statically until the BPU is enabled again.
2:0	Reserved	N/A	N/A	Reserved, tied to 0.

8.3.4.2 Performance Monitor CSRs

8.3.4.2.1 Introduction

RCPU supports configurable performance monitor CSR used for counting on various events for performance profiling, and their implementation follow exactly RISC-V privilege specification (user can refer to it for a detailed description).

8.3.4.2.2 mhpmcounter[i] (3 ≤ i ≤ 6)

mhpmcounter[i] (3 ≤ i ≤ 6) is used to record specific micro-architecture event numbers.

Offset: 0xB00+i				
Bits	**Field**	**Type**	**Reset**	**Description**
MXLEN-1:32	Reserved	RW	0x0	Tied to 0 for RV32, applicable only for RV64
31:0	mhpmcounteri	RW	0x0	Machine performance monitor counter i. (where 3 ≤ i ≤ 6)

8.3.4.2.3 mhpmcounter[i] (7 ≤ i ≤ 31)

Offset: 0xB00+i				
Bits	**Field**	**Type**	**Reset**	**Description**
MXLEN-1:32	Reserved	R	0x0	Tied to 0, applicable only for RV64.
31:0	mhpmcounteri	RW	0x0	Tie 0. (where 7 ≤ i ≤ 31)

8.3.4.2.4 mhpmcounterh[i]

mhpmcounterh[i] is used to record the upper 32 bit numbers of specific micro-architecture event. It is only for RV32.

Offset: 0xB80+i				
Bits	**Field**	**Type**	**Reset**	**Description**
MXLEN-1:0	mhpmcounterhi	R	0x0	Tied to 0, applicable only for RV32. (where 0 ≤ i ≤ 31)

8.3.4.2.5 mcountinhibit

The counter-inhibit **mcountinhibit** CSR is a **MXLEN-bit WARL** register that controls which hardware performance-monitoring counters increment. The settings in this CSR register affect only the increment behavior of the counters, and they do not influence the accessibility of the counters, in particular:

- When the **CY**, **IR** or **HPMn** bit in the **mcountinhibit** register is clear (set to 0), the corresponding counter (**cycle**, **instret**, or **hpmcountern**) increments as usual.
- When the **CY**, **IR**, or **HPMn** bit is set (set to 1), the corresponding counter (**cycle**, **instret**, or **hpmcountern**) does not increment.

Offset: 0x320				
Bits	**Field**	**Type**	**Reset**	**Description**
MXLEN-1:7	Reserved	R	0x0	-
6	HPM6	RW	0x0	If set, the increment of the performance monitor counter 6 is stopped
5	HPM5	RW	0x0	If set, the increment of the performance monitor counter 5 is stopped
4	HPM4	RW	0x0	If set, the increment of the performance monitor counter 4 is stopped
3	HPM3	RW	0x0	If set, the increment of the performance monitor counter 3 is stopped
2	IR	RW	0x0	If set, the increment of the instret counter is stopped
1	Reserved	R	0x0	-
0	CY	RW	0x0	If set, the increment of the cycle counter is stopped

8.3.4.2.6 mhpmevent3~6

The event selector **mhpmevent3~mhpmevent31** CSR, are **MXLEN-bit WARL** registers that control which event causes the corresponding counter to increase.

Offset: 0x320+i				
Bits	**Field**	**Type**	**Reset**	**Description**
MXLEN	Reserved	R	0x0	-

Offset: 0x320+i				
Bits	**Field**	**Type**	**Reset**	**Description**
-1:32				
31	mevent_enable	RW	0x1	Enable the corresponding performance monitor counter increment for events in Machine Mode.
30	Reserved	R	0x0	0
29	sevent_enable	RW	0x1	Enable the corresponding performance monitor counter increment for events in Supervisor Mode.
28	uevent_enable	RW	0x1	Enable the corresponding performance monitor counter increment for events in User Mode.
27:9	Reserved	R	0x0	-
8:4	event_idx	RW	0x0	Detailed event selector. For instruction commit events, see **Section 8.3.4.2.8** for more details. For memory access events, see **Section 8.3.4.2.9** for more details.
3:0	event_sel	RW	0x0	0: Select the instruction commit events, such as load, store, bjp ect. (see **Section 8.3.4.2.8** for more details). 1: Select the memory access events, such as icache miss, dcache miss ect. (see **Section 8.3.4.2.9** for more details).

8.3.4.2.7 mhpmevent7~31

Offset: 0x320+i				
Bits	**Field**	**Type**	**Reset**	**Description**
MXLEN-1:0	Reserved	R	0x0	-

8.3.4.2.8 mhpeventi[3:0]==0

Event selection value for instruction commit events are tabled below.

event_idx (mhpeventi[3:0]==0)	Event Name
1	Cycle count
2	Retired instruction count
3	Integer load instruction (includes LR)
4	Integer store instruction (includes SC)
5	Atomic memory operation (do not include LR and SC)
6	System instruction

event_idx (mhpeventi[3:0]==0)	Event Name
7	Integer computational instruction (excluding multiplication/division/remainder)
8	Conditional branch
9	Taken conditional branch
10	JAL instruction
11	JALR instruction
12	Return instruction
13	Control transfer instruction (CBR+JAL+JALR)
14	-
15	Integer multiplication instruction
16	Integer division/remainder instruction
17	Floating-point load instruction
18	Floating-point store instruction
19	Floating-point addition/substraction
20	Floating-point multiplication
21	Floating-point fused multiply-add (FMADD, FMSUB, FNMSUB, FNMADD)
22	Floating-point division or square-root
23	Other floating-point instruction
24	Conditional branch prediction fail
25	JAL prediction fail
26	JALR prediction fail

8.3.4.2.9 mhpeventi[3:0]==1

Event selection value for instruction commit events are tabled below.

event_idx (mhpeventi[3:0]==1)	Event Name
1	Icache miss
2	Dcache miss
3	ITLB miss
4	DLTB miss
5	Main TLB miss

8.3.4.3 TIMER Registers

8.3.4.3.1 Introduction

The **base address** of **TIME** registers is **0xE003_0000**

8.3.4.3.2 mtime_lo

Offset: 0x0000				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	mtime	RW	0x0	Reflect the lower 32-bit value of mtime. Shadow copy of MTIME in CLINT mode

8.3.4.3.3 mtime_hi

Offset: 0x0004				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	mtime	RW	0x0	Reflect the upper 32-bit value of mtime. Shadow copy of MTIME in CLINT mode

8.3.4.3.4 mtimecmp_lo

Offset: 0x0008				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	mtimecmp	RW	0xFFFFFFFF	Set the lower 32-bit value of mtimecmp. Shadow copy of MTIMECMP for Hart-0 in CLINT mode.

8.3.4.3.5 mtimecmp_hi

Offset: 0x000C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	mtimecmp	RW	0xFFFFFFFF	Set the upper 32-bit value of mtimecmp. Shadow copy of MTIMECMP for Hart-0 in CLINT mode.

8.3.4.3.6 mtime_srw_ctrl

Offset: 0x0FEC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	N/A	N/A	Control S-mode can access this timer or not.
0	mtime_srw_ctrl	RW	0x0	Control S-mode can read or write timer registers or not: - 0: S-Mode can read/write all timer registers - 1: S-Mode can not read/write timer registers

8.3.4.3.7 msftrst

Offset: 0x0FF0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	MSFTRST	RW	0x0	Generate a software-reset request
30:0	Reserved	N/A	N/A	Reserved, tied to 0.

8.3.4.3.8 mtimectl

Offset: 0x0FF8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	N/A	N/A	Reserved, tied to 0
2	CLKSRC	RW	0x0	Select the source of increment frequency. If this field is 1, then the increment frequency is rcpu_aon_clk, otherwise the increment frequency is controlled by mtime_toggle_a.
1	CMPCLREN	RW	0x0	Control the timer count to clear-to-zero or not. If this field is 1, then the mtime register will be cleared to zero after generating timer interrupt, otherwise it increases normally. **Note.** If CMPCLREN is enabled, the timer interrupt request will be a pulse request. In this case, timer interrupt should be set to edge-trigger mode.
0	TIMESTOP			Control the timer count or pause. If this field is 1, then the timer pauses, otherwise it increases normally.

8.3.4.3.9 msip

Offset: 0x0FFC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Reserved	N/A	N/A	Reserved, tied to 0.
0	MSIP	RW	0x0	This bit is used to generate a software interrupt. Shadow copy of MSIP for Hart-0 in CLINT mode

8.3.4.4 ECLIC Registers

8.3.4.4.1 Introduction

The **base address** of **ECLIC** registers is **0xE002_0000**

8.3.4.4.2 cliccfg

cliccfg is a global configurations setting register, the software can write into it.

Offset: 0x0000				
Bits	**Field**	**Type**	**Reset**	**Description**
7:5	Reserved	R	N/A	Reserved, tied to 0
4:1	nlbits	RW	0x0	Used to specify the bit-width of level and priority in the register clicintctl[i]
0	Reserved	R	N/A	Reserved, tied to 1

8.3.4.4.3 clicinfo

clicinfo is a global parameters register, the software can query it.

Offset: 0x0004				
Bits	**Field**	**Type**	**Reset**	**Description**
31:25	Reserved	R	N/A	Reserved, tied to 0
24:21	CLICINTCTLBITS	R	N/A	Used to specify the effective bit-width of the register clicintctl[i]
20:13	VERSION	R	N/A	Hardware implementation version number
12:0	NUM_INTERRUPT	R	N/A	Number of interrupt sources supported by the hardware

8.3.4.4.4 mth

mth is used to set the target interrupt threshold level, the software can write into it.

Offset: 0x000b				
Bits	**Field**	**Type**	**Reset**	**Description**
7:0	mth	RW	N/A	Target threshold level register. **Note.** Only when the level of the interrupt finally arbitrated by ECLIC is higher than the level in this register, the interrupt can be sent to the processor core.

8.3.4.4.5 clicintip[i]

clicintip[i] is the pending flag register for the interrupt source.

Offset: 0x1000+4*i				
Bits	**Field**	**Type**	**Reset**	**Description**
7:1	Reserved	RO	N/A	Reserved, tied to 0.
0	IP	RW	0x0	Interrupt source pending flag: - 0: interrupt is not triggered - 1: interrupt is triggered **Note.** For edge-triggered interrupt source, the IP bit may be cleared by the hardw

8.3.4.4.6 clicintie[i]

clicintie[i] is the enable bit register for the interrupt source.

Offset: 0x1001+4*i				
Bits	**Field**	**Type**	**Reset**	**Description**
7:1	Reserved	R	N/A	Reserved, tied to 0.
0	IE	RW	0x0	Interrupt enable bit: - 0: interrupt source is masked - 1: interrupt is enabled

8.3.4.4.7 clicintattr[i]

clicintattr[i] is used to indicate the attribute of the interrupt source, the software can write into it.

Offset: 0x1002+4*i				
Bits	**Field**	**Type**	**Reset**	**Description**
7:6	Reserved	R	N/A	Reserved, tied to 2'b11
5:3	Reserved	R	N/A	Reserved, tied to 0
2:1	trig	RW	0x0	Used to set the level or edge triggered attribute of the interrupt source: - 2'b00: level-triggered interrupt. The IP bit of the interrupt source will reflect the level of the interrupt source in real time. The software writing to this IP bit is ignored. The IP bit can only be cleared by clearing the original source of the interrupt. - 2'b01: Rising edge-triggered interrupt the software can set or clear the IP bit by writing operations. - 2'b11: Falling edge-triggered interrupt, the software can set or clear the IP bit by writing operations. **Note.** In order to improve the efficiency of the interrupt processing, when the interrupt is taken and the core jumps to the ISR (Interrupt Service Routine), the hardware of ECLIC will clear the IP bit automatically, and the software needs not to clear the IP bit in ISR.
0	shv	RW	0x0	Used to set whether the interrupt is vectored or non-vectored: - 0: Non-vectored mode. The core will jump to the common base entry shared by all interrupts when the interrupt is taken. - 1: Vectored mode. The core will directly jump to the target address stored in the vector table entry when the interrupt is taken.

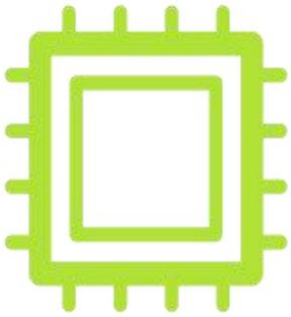
8.3.4.4.8 clicintctl[i]

clicintctl[i] is the control register of the interrupt source, the software can set the level and priority into it. The level and priority field are dynamically allocated based on the value of **cliccfg.nlbits**.

Offset: 0x1003+4*i				
Bits	**Field**	**Type**	**Reset**	**Description**
7:7-(nlbits-1)	Level	RW	N/A	Interrupt source level, a greater value of level results in a higher ECLIC arbitration priority.
7-(nlbits-1):7-(CLICINTCTLBITS-1)	Priority	RW	N/A	Interrupt source priority, a greater value of level results in a higher ECLIC arbitration priority.
7-(CLICINTCTLBITS-1):0	Reserved	RO	N/A	Reserved, tied to 1.

Chapter 9

Top System



Key Stone® K1 User Manual

9.1 Overview

The Top System of K1 consists of

- Clock & Reset
- JTAG
- DMA
- Crypto
- Timer & Watchdog
- RTC
- Mailbox
- Power Management & Lower Power Mode Control

9.2 Clock & Reset

9.2.1 Introduction

K1 comes with the following clocks:

- One 32K RTC clock
- One 24M OSC clock

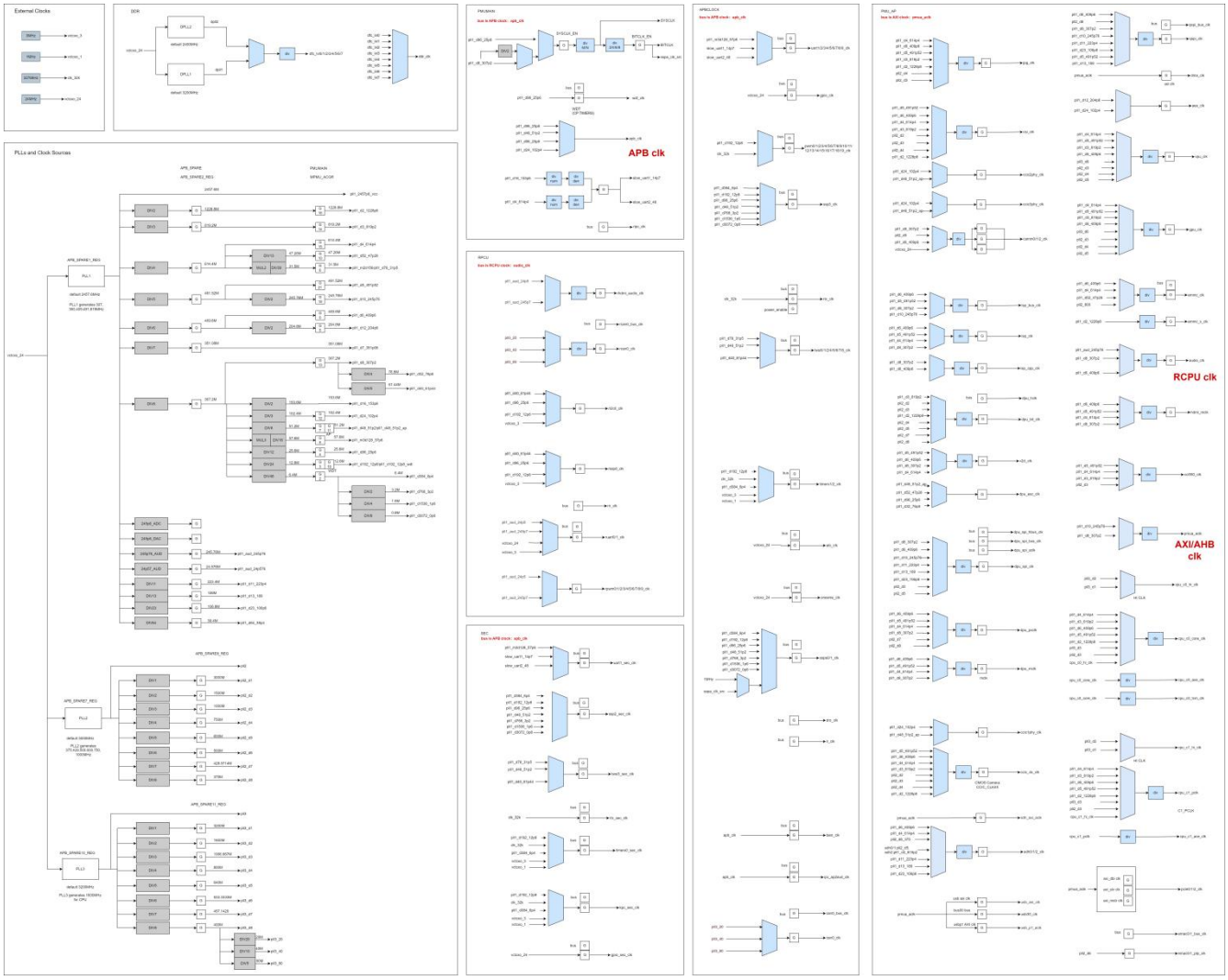
9.2.2 Features

- Three PLLs implemented inside to provide various frequencies to meet different scenario requirements
- DVFS feature supported to balance the tradeoff between power and performance
- Glitch-free clock switches and clock dividers implemented to provide all required frequencies with limited PLLs cost
- Clock gating and software reset schemes applied to modules in fine granularity to achieve power saving and flexible management

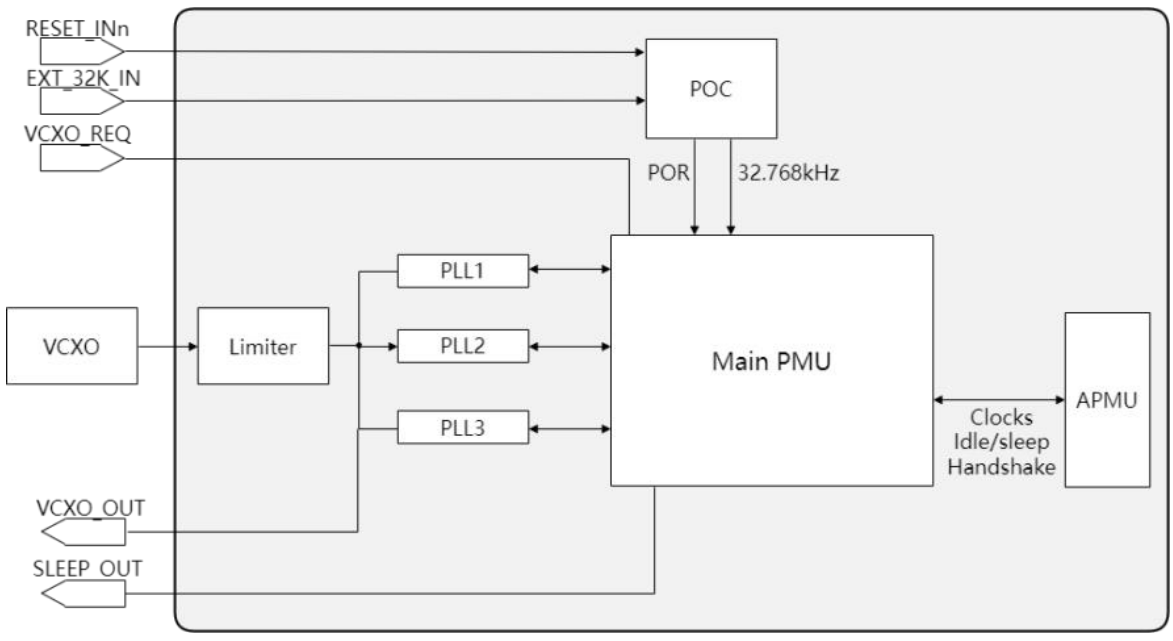
9.2.3 Functional Description

9.2.3.1 Clock System

The detailed clock tree structure is depicted below, where is highlighted how the clock signals are generated, managed and distributed across the system to support various modules and functions.



Instead, the high-level architecture of the clock system is depicted below.



VCXO_OUT is driven with the OSC frequency if either of the following occurs:

- VCXO_REQ is asserted, and the relevant REQ_EN bit field is set in the VCXO software request control register
- Software request bit field is enabled in the VCXO software request control register
  There are three Phase-Locked Loop (PLL) designed to accept a wide range of input frequencies, and generate a broad range of output frequencies to all modules for functioning properly in different application scenario. Details for each PLL are provided in the following subsections.

9.2.3.1.1 PLL1

PLL1 is designed to generate fixed frequency points for the CPU cores and other peripherals, where

- Changes of the run-time frequency in the PLL1 output are only available for debugging purposes and should not be used in production systems
- PLL1 is enabled by default at system reset and shutdown only when the entire chip entered sleep mode with VCXO shutdown enabled
- The settings configured in the PLL1 and oscillator control registers of the Main PMU control the delay required for the PLL1 output clocks to stabilize after system reset or shutdown
- Updating the PLL1 configuration registers to change frequency during normal operations is not recommended

9.2.3.1.2 PLL2

PLL2 is designed to generate various fixed frequencies, working alongside PLL1 to provide a full range of frequencies required for different modules, where

- Changes of run-time frequency in the PLL2 output are only available for debugging purposes and should not be used in production systems
- PLL2 is disabled at system reset and must be enabled through software when required
- The settings configured in the PLL2 and oscillator control registers of the Main PMU control the delay required for the PLL2 output clocks to stabilize after system reset or shutdown
- Updating the PLL2 configuration registers to change frequency during normal operations is not recommended

9.2.3.1.3 PLL3

PLL3 is designed to provide frequencies for CPU frequency scaling and switching, where

- PLL3 is disabled at system reset and must be enabled through software when required
- The settings configured in the PLL3 and oscillator control register of the Main PMU control the delay required for the PLL3 output clocks to stabilize after system reset or shutdown
- Updating the PLL3 configuration registers to change frequency during normal operations is not recommended

9.2.3.2 Resource Reset Schemes

K1 allows applying different schemes of resource reset as tabled below.

No.	Resource Reset Scheme	Description
1	Power-On-Reset	Reset the whole chip during power-on sequence
2	WatchDog Reset	Reset the whole chip excluding pinmux registers and debug registers
3	Module Software Reset	Reset each module individually through software
4	Power Island POR Reset	Reset the whole power island during its power-on sequence

9.2.4 Register Description

9.2.4.1 Basing on <PMUMAIN(D4050000>

9.2.4.1.1 FREQUENCY CHANGE CONTROL REGISTER (FCCR)

Offset: 0xD4050000+0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	Reserved	RSVD	0xX	Reserved. Always write 0, ignore read value.
29	I2SCLK307M	RW	0x0	12S Divider Clock 307.2M 12S Divider Clock 307.2M selection: - 1 = 307.2M PLL1 Clock - 0 = PLL1 307.2M/2=153.5M clock
28:9	Reserved	RSVD	0xX	Reserved. Always write 0, ignore read value.
8:0	PLL1 FBD	RW	0x0	PLL1 FBDIV Actually this register bits are not used in Auiqla, it's fixed to Ox30 now.

9.2.4.1.2 PLL & OSCILLATOR CONTROL REGISTER (POCR)

Offset: 0xD4050000+0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	FORCE	RW	0x0	Force all clocks to be free running: - **1'b0**: The 'FORCE' field does not affect clock gating. Standard clock gating behavior is maintained. - **1'b1**: All output clocks are continuously running and not gated.
30:24	RSVD	RO	0	Reserved

Offset: 0xD4050000+0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
23:16	VCXOST	RW	0xE5	Determines the wait time for VCTCXO and clock limiter stabilization: - **0x0**： wait time = one 32.768 KHz clock cycle - **0x1**： wait time = two 32.768 KHz clock cycles .... - **0xE5**: wait time = 230 32.768 KHz clock cycles - **0xE6 - 0xFF**: Reserved
15:12	RSVD	RO	0	Reserved
11:0	PLLLOCK	RW	0x0	Determines the wait time for PLL lock. This value determines lock time for both PLLs (PLL1 and PLL2): - **0x0**： Main PLL lock time = VCTCXO/2 cycle - **0x1**: Main PLL lock time = 2 VCTCXO/2 cycles - **0xFFF**: Main PLL lock time = 4096 VCTCXO/2 cycles

9.2.4.1.3 VCTCXO SW REQUEST CONTROL REGISTER (VRCR)

Offset: 0xD4050000+0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	RO	0x0	Reserved
15	VCXO_OUT_REQ_EN3	RW	0x0	Enable VCXO_REQ2 for VCXO_OUT
14	VCXO_OUT_REQ_EN2	RW	0x0	Enable VCXO_REQ2 for VCXO_OUT
13	VCXO_OUT_REQ_EN1	RW	0x0	Enable VCXO_REQ1 for VCXO_OUT
12	VCXO_OUT_REQ_EN0	RW	0x0	Enable VCXO_REQ0 for VCXO_OUT
11:9	RSVD	RO	0x0	Reserved
8	SW_REQ	RW	0x0	SW request of VCTCXO
7	REQ_POL3	RW	0x0	VCXO_REQ3 request polarity
6	REQ_POL2	RW	0x0	VCXO_REQ2 request polarity
5	REQ_POL1	RW	0x0	VCXO_REQ1 request polarity
4	REQ_POL0	RW	0x0	VCXO_REQ0 request polarity
3	RSVD	RO	0x0	Reserved
2	REQ_EN2	RW	0x0	Enable VCXO_REQ2 for sysclk_en
1	Reserved	RO	0x0	Reserved
0	REQ_EN0	RW	0x0	Enable VCXO_REQ0 for OCLK1

9.2.4.1.4 SoC I2S CLOCK GENERATION CONTROL REGISTER (ISCCR0/1)

Offset: 0xD4050000+0x40/0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31	SYSCLK_EN	RW	0x0	Enables the I2S clock input to SYSCLKn Generator and its output
30	SYSCLK_SRC	RW	0x1	Selects the I2S M/N divider input clock frequency: - **1'b0**: Clock rate is 25.6 MHz - **1'b1**: Clock rate is based on 153.6 MHz
29	BITCLK_EN	RW	0x0	Enables the I2S clock input to the bit clock generator
28:27	BITCLK_DIV_468	RW	0x3	Determines BITCLK1 to SYSCLK1 relation: - 2'b00: BITCLK1 rate = SYSCLK1 rate / 2 - 2'b01: BITCLK1 rate = SYSCLK1 rate / 4 - 2'b10: BITCLK1 rate = SYSCLK1 rate / 6 - 2'b11: BITCLK1 rate = SYSCLK1 rate / 8
26:15	DENOM	RW	0x40	- I2S clock generation programmable divider denominator value - The I2S sysclk is generated using a fractional divider.
14:0	NOM	RW	0x130B	- I2S clock generation programmable divider numerator value. - The I2S sysclk is generated using a fractional divider.

9.2.4.1.5 WDT (CP TIMERS) CONTROL REGISTER (WDTPCR)

Offset: 0xD4050000+0x200				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0	Reserved
2	RST	RW	0x0	Timers Hardware reset generation (resets both APB & Functional domains): 1'b0: No reset 1'b1: Reset
1	FNCLK	RW	0x0	Timers and WDT functional Clock enable/disable: 1'b0: Clock off 1'b1: Clock on
0	ABCLK	RW	0x0	CP Timers and WDT APB Bus Clock enable/disable: 1'b0: Clock off 1'b1: Clock on

9.2.4.1.6 RIPC CONTROL REGISTER (RIPCCR)

Offset: 0xD4050000+0x210				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RSVD	0xX	Reserved Always write 0. Ignore read value.
2	RST	RW	0x0	RST R-IPC Hardware reset generation 1'b0: No reset 1'b1: Reset
1	Reserved	RSVD	0xX	Reserved Always write 0. Ignore read value.
0	APBCLK	RW	0x0	APBCLK R-IPC APB Bus Clock enable/disable 1'b0: Clock off 1'b1: Clock on

9.2.4.1.7 CLOCK GATING REGISTER (CGR)

Offset: 0xD4050000+0x1024				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	RO	0	Reserved
21	CLK_491P5M	RW	0x1	Enable the functional 491.52 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
20	RSVD	RO	0	Reserved
19	WDT_12P8M	RW	0x0	Enable the functional 12.8 MHz clock output of the main to the WatchDogTimer. - 1'b0: Clock gated - 1'b1: Clock enabled
18	CLK_245P7M	RW	0x1	Enable the functional 245.76 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
17	RSVD	RO	0	Reserved
16	CLK_1228P8M	RW	0x0	Enable the functional 1228.8 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
15	CLK_6	RW	0x1	Enable the functional 614.4 MHz clock output.

Offset: 0xD4050000+0x1024				
Bits	**Field**	**Type**	**Reset**	**Description**
	14P4M			- 1'b0: Clock gated - 1'b1: Clock enabled
14	CLK_819P2M	RW	0x1	Enable the functional 819.2 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
13	CLK_307P2M	RW	0x1	Enable the functional 307.2 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
12	CLK_102P4M	RW	0x0	Enable the functional 102.4 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
11	CLK_51P2_AP	RW	0x0	Enable the functional 51.2 MHz clock output for AP PMU and AP Peripherals - 1'b0: Clock gated - 1'b1: Clock enabled
10	CLK_47P2M	RW	0x0	Enable the functional 47.26 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
9	GPC	RW	0x0	Enable the M/N clock generator of the VCXO clock configured through GPCR, the clock is output to VCXO_OUT PAD func3 - 1'b0: Clock gated - 1'b1: Clock enabled
8	AP_FUART	RW	0x0	Enable the functional fast UART clock output (57.6 MHz) of the main to the Application Processor APB portion. - 1'b0: Clock gated - 1'b1: Clock enabled
7	CLK_51P2M	RW	0x0	Enable the functional 51.2 MHz clock output for APB Peripherals - 1'b0: Clock gated - 1'b1: Clock enabled
6	AP_TWSI	RW	0x0	Enable the 31.5M clock of the functional TWSI clock output of the main to the Application Processor APB portion. - 1'b0: Clock gated - 1'b1: Clock enabled
5	CLK_204P8M	RW	0x0	Enable the functional 204.8 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
4	CLK_25P6M	RW	0x1	Enable the functional 25.6 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
3	CLK_1	RW	0x0	Enable the functional 12.8 MHz clock output.

Offset: 0xD4050000+0x1024				
Bits	**Field**	**Type**	**Reset**	**Description**
	2P8M			- 1'b0: Clock gated - 1'b1: Clock enabled
2	CLK_6P4M	RW	0x0	Enable the functional 6.4 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled
1	AP_SUART	RW	0x0	Enable the functional M/N slow UART clock output (configured through SUCCR) of the main to the Application Processor APB portion. It is used to enable UART slow clock(14.74M) source. - 1'b0: Clock gated - 1'b1: Clock enabled
0	CLK_409P6M	RW	0x1	Enable the functional 409.6 MHz clock output. - 1'b0: Clock gated - 1'b1: Clock enabled

9.2.4.1.8 APB CLOCK SOURCE CONTROL REGISTER (APBCSCR)

Offset: 0xD4050000+0x1050				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	RSVD	RO	0x0	Reserved
1:0	APB_52M	RW	0x0	System APB Clock Source Selection - 2'b00/2'b10: 25.6Mhz - 2'b01: 51.2Mhz - 2'b11: 102.4MHz

9.2.4.1.9 PM_MN_CLK CONTROL REGISTER (PM_MN_CLK)

Offset: 0xD4050000+0x10A4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:5	PM_MN_CLK2_SEL	RW	0x0	PM_MN_CLK2 Selector: - 1'b0: Select vctcxo clock - 1'b1: Select PLL2 divided by 5 (600MHz)
4	PM_MN_CLK2_SW_EN	RW	0x0	PM_MN_CLK2_SW_EN - 1'b1: Software enable for PM_MN_CLK2
3:2	RSVD	RO	0x0	Reserved

Offset: 0xD4050000+0x10A4				
Bits	**Field**	**Type**	**Reset**	**Description**
1	PM_MN_CLK_SEL	RW	0x0	PM_MN_CLK Selector: - 1'b0: Select vctcxo clock - 1'b1: Select PLL1 (614MHz)
0	PM_MN_CLK_SW_EN	RW	0x0	PM_MN_CLK_SW_EN - 1'b1: Software enable for PM_MN_CLK

9.2.4.1.10 SLOW UART (UART 1) CLOCK GENERATION CONTROL REGISTER (SUCCR_1)

Offset: 0xD4050000+0x10B0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:29	RSVD	RO	0	Reserved
28:16	UARTDIVN_1	RW	0x1800	- Programmable numerator value for UART clock generation. - The UART clock is generated using a fractional divider.
15:13	RSVD	RO	0	Reserved
12:0	UARTDIVD_1	RW	0x3C0	- Programmable denominator value for UART clock generation. - The UART clock is generated using a fractional divider

9.2.4.2 Basing on <PMU_BASE(0xD4282800)>

9.2.4.2.1 AP CLOCK CONTROL REGISTER (AP CLOCK CONTROL REGISTER)

Offset: 0xD4282800+0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31	AP_RD_ST_CLEAR	RW	0x0	RD_ST clears AP_RD_STATUS by writing logic High to this bit
30:23	RSVD	RO	0x0	Reserved
22	DDR_FREQ_CHG_REQ	RW	0x0	DDR CLK Frequency Change Request - Used to request a change in the DDR clock frequency.
21:19	RSVD	RO	0x0	Reserved
18	AP_ALLOW_SPD_CHG	RW	0x0	AP Speed Change Voting - Indicates whether the AP is allowed to change speed.
17:0	RSVD	RO	0x0	Reserved

9.2.4.2.2 DUMMY AP CLOCK CONTROL REGISTER (PMU_DM_CC_AP)

Offset: 0xD4282800+0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RSVD	RO	0x0	Reserved
30	AP_C1_FC_DONE	RO	0x0	AP Cluster1 Frequency Change Done Status - 1'b0: Frequency change is not done - 1'b1: Frequency change is done
29	ACLK_FC_DONE	RO	0x0	ACLK Frequency Change Done Status - 1'b0: Frequency change is not done - 1'b1: Frequency change is done
28	DCLK_FC_DONE	RO	0x0	DDR CLK Frequency Change Done Status - 1'b0: Frequency change is not done - 1'b1: Frequency change is done
27	AP_C0_FC_DONE	RO	0x0	AP Cluster0 Frequency Change Done Status - 1'b0: Frequency change is not done - 1'b1: Frequency change is done
26	RSVD	RO	0x0	Reserved
25	AP_RD_STATUS	RO	0x0	AP Read Status： - Indicates that the dummy AP clock register is being read by AP. - Reading the status is the beginning of the frequency update process. - If this bit is set, it indicates that one of the cores is currently updating its frequency. The other core must wait until this bit is cleared (by using the <RD_ST Clear> field in the AP Clock Control Register), which means that the frequency change has been completed.
24:12	RSVD	RO	0x0	Reserved
11:9	C1_ACLK_DIV	RO	0x1	Clock Divider Selection for Cluster1 AXI Interface Clock: - C1_ACLK is divided from PCLK. - Formula: C1_ACLK= PCLK / (this field +1)
8:6	C1_CLK_DIV	RO	0x1	Clock Divider Selection for Cluster1 PCLK. If (FCAP.C1_PLLSEL<=3), PCLK= (FCAP.C1_PLLSEL selectin Clock)/(C1_CLK_DIV +1); else PCLK =FCAP.C1_PLLSEL selectin Clock
5:3	C0_ACLK_DIV	RO	0x1	Clock Divider Selection for Cluster0 AXI Interface: - Clock C0_ACLK is divided from PCLK. - Formula: C0_ACLK= PCLK / (this field +1)

Offset: 0xD4282800+0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
2:0	C0_CLK_DIV	RO	0x1	Clock Divider Selection for Cluster0 PCLK. If (FCAP.C0_PLLSEL<=3), PCLK= (FCAP.C0_PLLSEL selectin Clock)/(C0_CLK_DIV +1); else PCLK =FCAP.C0_PLLSEL selectin Clock

9.2.4.2.3 JPEG CLOCK/RESET CONTROL REGISTER (PMU_JPEG_CLK_RES_CTRL)

Offset: 0xD4282800+0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	RO	0x0	Reserved
18	AFBC_2K_RESET	RW	0x0	2K AFBC reset: - 1'b1: release - 1'b0: reset
17:16	RSVD	RO	0x0	Reserved
15	JPG_CLK_FC_REQ	W1C	0x0	JPEG Function Clock FC Request - 1'b1: Triggers a frequency change; - This field is automatically cleared by hardware once the frequency change is complete.
14:8	RSVD	RO	0x0	Reserved
7:5	JPG_CLK_DIV	RW	0x0	JPEG Function Clock Divide Ratio: - isp_clk = JPEG_CLK_DIV / (this field +1). **Note.** The divider only applicable for clock source 0~3
4:2	JPG_CLK_SEL	RW	0x0	JPEG Function Clock Source Select - 3'b000: PLL1_614MHz - 3'b001: PLL1_409MHz - 3'b010: PLL1_491MHz - 3'b011: PLL1_819MHz - 3'b100: PLL1_1228MHz - 3'b101: PLL2_div4 - 3'b110: PLL2_div3 - 3'b111: 0
1	JPG_CLK_EN	RW	0x0	JPEG Function Clock Enable - 1'b1: Clock enabled - 1'b0: Clock disabled
0	JPG_CLK_RSTN	RW	0x0	JPEG Function Clock Reset

Offset: 0xD4282800+0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b1: Release - 1'b0: Reset

9.2.4.2.4 CSI CCIC2 CLOCK/RESET CONTROL REGISTER (PMU_CSI_CCIC2_CLK_RES_CTRL)

Offset: 0xD4282800+0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CCIC3_PHY_CLK_SEL	RW	0x0	CCIC3 PHY Clock Select - 1‘b0: 104 MHz - 1'b1: 52 MHz
30	CCIC3_PHY_CLK_EN	RW	0x0	CCIC3 PHY Clock Enable - 1'b1: PHY clock enabled - 1'b0: PHY clock disabled
29	CCIC3_PHY_CLK_RST	RW	0x0	CCIC3 PHY Clock Reset. - 1'b0: Reset **Note.** This clock is also used for DPHY reset.
28	CAM_MCLK0_EN	RW	0x0	CAM MCLK0 ENABLE
27	CAM_MCLK2_EN	RW	0x0	CAM MCLK ENABLE
26:23	ISIM_VCLK_OUT_DIV	RW	0xB	Clock Divider for ISIM_VCLK_OUT The source sensor reference clock is selected by bit [9:8]. The divider of the sensor reference clock is controlled by this register. ISIM_VCLK_OUT = CAM_MCLK / (this field +1)， except that 0xf is (this field +2) actually The default value is 312/(11+1) = 26 MHz
22:20	CSI_FNC_CLK_DIV	RW	0x1	Sets the divide ratio for CSI Controller Function Clock: CSI_FNC_CLK = CSI_CLK / (this field +1)
19	RSVD	RO	0	Reserved
18:16	CSI_CLK_SEL	RW	0x0	Selects the source for CSI Controller Function Clock - 3'b000: PLL1_491MHz - 3'b001: PLL1_409MHz - 3'b010: PLL1_614MHz - 3'b011: PLL1_819MHz - 3'b100: PLL2_Div2 - 3'b101: PLL2_Div3 - 3'b110: PLL2_Div4 - 3'b111: PLL1_1248MHz

Offset: 0xD4282800+0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
15	CSI_CLK_FC_REQ	W1C	0x0	CSI Controller Function Clock Request - 1'b1: Trigger frequency change process; This field is automatically cleared when the frequency change is complete.
14:10	RSVD	RO	0	Reserved
9:8	CAM_MCLK_SEL	RW	0x0	Selects CAM MCLK - 2'b00: 312 MHz - 2'b01: PLL2_DIV5 - 2'b10: 416Mhz - 2'b11: 24Mhz vctcxo
7	CCIC2_PHYCLK_SEL	RW	0x0	Selects clock CCIC2 PHY - 1'b0: 104 MHz - 1'b1: 52 MHz
6	CAM_MCLK1_EN	RW	0x0	Enables CAM MCLK1
5	CCIC2_PHYCLK_EN	RW	0x0	CCIC2 PHY Clock Enable - 1'b1: PHY clock enabled - 1'b0: PHY clock disabled
4	CSI_CLK_EN	RW	0x0	Enables CSI Controller Function Clock - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3	CAM_MCLK2_EN	RW	0x0	Enables CAM MCLK2
2	CCIC2_PHYCLK_RST	RW	0x0	CCIC2 PHY Clock Reset - 1'b0: Reset **Note.** This clock is also used for DPHY reset.
1	CSI_CLK_RST	RW	0x0	Reset for CSI Controller Function Clock. - 1'b0: Reset
0	RSVD	RO	0	Reserved

9.2.4.2.5 CMOS CAMERA INTERFACE CONTROLLER 1 DYNAMIC CLOCK GATE CONTROL REGISTER (PMU_CCIC1_CLK_GATE_CTRL)

Offset: 0xD4282800+0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	CCIC1_GATE_CSI_CLK_STATIC	RW	0x3	CCIC1 CSI Static Clock Gate Control - 0x0 = Hardware dynamic control - 2'b00/2'b01/2'b10: Stop clock

Offset: 0xD4282800+0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 2'b11: Free running
29:28	CCIC1_GATE_CLK4X_STATIC	RW	0x3	CCIC1 CLK4x Static Clock Gate Control - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
27:26	CCIC1_GATE_CLK1X_STATIC	RW	0x3	CCIC1 CLK1 Static Clock Gate Control - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
25:24	CCIC1_GATE_HCLK_STATIC	RW	0x3	CCIC1 HCLK Static Clock Gate Control - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
23:22	CCIC1_GATE_ACLK_STATIC	RW	0x3	CCIC1 aclk Static Clock Gate Control - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
21:20	CCIC1_GATE_ISP_PIP1_CLK1X_STATIC	RW	0x3	ISP PIPE1 clk1x Static Clock Gate Control - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
19:18	CCIC1_GATE_DMA_ACLK_STATIC	RW	0x3	CCIC DMA aclk Static Clock Gate Contro - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
17:16	CCIC1_GATE_AXI_BRIDGE_CLK_STATIC	RW	0x3	CCIC DMA (AXI Bridge 64 to128) CLK Static Clock Gate Control - 2'b00/2'b01/2'b10: Stop clock - 2'b11: Free running
15:14	CCIC1_GATE_LANE3_CLK_DYNAMIC	RW	0x3	CCIC1 CSI Lane3 Clock Dynamic Clock Gate Control - 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
13:12	CCIC1_GATE_LANE2_CLK_DYNAMIC	RW	0x3	CCIC1 CSI Lane 2 Clock Dynamic Clock Gate Control - 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
11:10	CCIC1_GATE_LANE1_CLK_DYNAMIC	RW	0x3	CCIC1 CSI Lane 1 Clock Dynamic Clock Gate Control - 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
9:8	CCIC1_GATE_LANE0_CLK_DYNAMIC	RW	0x3	CCIC1 CSI Lane 0 Clock Dynamic Clock Gate Control - 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
7:6	CCIC1_GATE_C	RW	0x3	CCIC1 CSI Clock Dynamic Clock Gate Control

Offset: 0xD4282800+0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
	SI_CLK_DYNAMIC			- 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
5:4	CCIC1_GATE_AHB_CLK_DYNAMIC	RW	0x3	CCIC1 ahb Clock Dynamic Clock Gate Control - 0x0 = Hardware dynamic control - 0x1 = Hardware dynamic control - 0x2 = Stop clock - 0x3 = Free running
3:2	CCIC1_GATE_PIP_CLK_DYNAMIC	RW	0x3	CCIC1 pipeline Clock Dynamic Clock Gate Control - 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
1:0	CCIC1_GATE__AXI_CLK_DYNAMIC	RW	0x3	CCIC1 axi Clock Dynamic Clock Gate Control - 2'b00/2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running

9.2.4.2.6 ISP CLOCK/RESET CONTROL REGISTER (PMU_ISP_CLK_RES_CTRL)

Offset: 0xD4282800+0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RSVD	RO	0x0	Reserved
30	ISP_REPAIR_MEM_CTRL_TRIG	W1C	0x0	ISP Repairable Memory Control Trigger - 1'b1: Triggers ISP repairable memory control - The field is cleared by hardware once the repair is done.
29	ISP_REPAIR_MEM_CTRL_DONE_BYPASS	RW	0x0	ISP Repairable Memory Control Done Bypass in ISP Hardware Mode. - This field is valid when the ISP is in hardware mode (ISP_HW_MODE = 1).
28	ISP_CPP_CLK_EN	RW	0x0	ISP CPP Function Clock Enable - 1'b1: Clock enabled - 1'b0：Clock disabled
27	ISP_CPP_CLK_RSTN	RW	0x0	ISP CPP Clock Reset - 1'b0: Reset
26	ISP_CPP_CLK_SEL	RW	0x0	ISP CPP Clock Select - 1'b0: 312Mhz - 1'b1: 416Mhz
25:24	ISP_CPP_CLK_DIV	RW	0x1	ISP CPP Clock Divider. - isp_mcu_clk = CLK/ (this field +1)

Offset: 0xD4282800+0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
23	ISP_CI_BUS_CLK_FC_REQ	W1C	0x0	ISP_CI Bus Clock FC Request - 1'b1: Triggers a frequency change - The field is cleared by hardware once the frequency change is done.
22:21	ISP_CI_BUS_CLK_SEL	RW	0x0	ISP_CI Bus Clock Select - 2'b00: PLL1_409 MHz - 2'b01: PLL1_491 MHz - 2'b10: PLL1_307MHz - 2'b11: PLL1_245Mhz
20:18	ISP_CI_BUS_CLK_DIV	RW	0x1	ISP_CI BUS Clock Divide Ratio. Isp_ci_divided_bus_clk = isp_ci_bus_clk / (this field +1)
17	ISP_CI_BUS_CLK_EN	RW	0x0	ISP_CI Bus Clock Enable. - This field enables the DMA clock for CCIC and ISP. - It controls the first level AXI clock gating for CCIC and ISP. - The second level AXI clock gating for CCIC and ISP is managed by separate gating control registers. - 1'b1: Clock enabled
16	ISP_CI_BUS_CLK_RST	RW	0x0	isp_ci bus clk Reset - 1'b0: Reset. - This reset will reset all the AXI logic in SC2 except ISP AXI logic.
15:10	RSVD	RO	0x0	Reserved
9:8	ISP_CLK_SEL	RW	0x0	ISP Function Clock Source Select: - 2'b00: 416 MHz - 2'b01: 499 MHz - 2'b10: 624 MHz - 2'b11: PLL1_307MHz
7	ISP_CLK_FC_REQ	W1C	0x0	ISP Function Clock Frequency Change Request. - 1'b1: Triggers a frequency change. - The field is cleared by hardware once the frequency change is done.
6:4	ISP_CLK_DIV	RW	0x1	ISP Function Clock Divide Ratio. isp_clk = ISP_CLK_DIV/(this field +1)
3	ISP_AHB_RESETN	RW	0x0	ISP AHB Resetn - 1'b0: Reset
2	RSVD	RO	0x0	Reserved
1	ISP_CLK_EN	RW	0x0	ISP Function Clock Enable - 1'b1: Clock enabled - 1'b0: Clock disabled
0	ISP_CLK_RSTN	RW	0x0	ISP Function Clock Reset

Offset: 0xD4282800+0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b0: Reset

9.2.4.2.7 LCD CLOCK/RESET CONTROL REGISTER1 (PMU_LCD_CLK_RES_CTRL1)

Offset: 0xD4282800+0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31	MIPI_BIT_CLK_FC_REQ	W1C	0x0	MIPI BIT CLK FC Request. - Write 1 to trigger a frequency change. - The field is cleared by hardware once the frequency change is done.
30	LCD_PXCLK_FC_REQ	W1C	0x0	LCD PXCLK FC Request. - Write 1 to trigger a frequency change. - The field is cleared by hardware once the frequency change is done.
29	LCD_MCLK_FC_REQ	W1C	0x0	LCD MCLK FC Request. - Write 1 to trigger a frequency change. - The field is cleared by hardware once the frequency change is done.
28	V2D_FCLK_FC_REQ	W1C	0x0	V2D FCLK FC Request. - Write 1 to trigger a frequency change. - The field is cleared by hardware once the frequency change is done.
27	V2D_SW_RST	RW	0x0	v2d clk domain reset - 1'b0: Reset assert - 1'b1: Reset release
26:24	RSVD	RO	0x0	Reserved
23	MIPI_BIT_BLANK_MSK	RW	0x1	MIPI BIT CLK FC wait BLANK signal mask: - 0: Wait for the LCD BLANK signal - 1: Do not wait for the LCD BLANK signal
22:20	MIPI_BIT_CLK_SEL	RW	0x0	MIPI BIT Clock Select - 3'b000: 832Mhz - 3'b001: 1.5GHz(PLL2_DIV2) - 3'b010: 1GHz(PLL2_DIV3) - 3'b011: 1248MHz - 3'b100: 750MHz - 3'b101: 600MHz - 3'b110: 429MHz - 3'b111: 375MHz
19:17	MIPI_BIT_CLK_D	RW	0x0	MIPI BIT Clock Divide Ratio.

Offset: 0xD4282800+0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
	IV			MIPI_BIT_CLK = clock source/ (this field +1)
16	MIPI_BIT_CLK_EN	RW	0x0	MIPI BIT CLK Enable. - 1'b0: Disabled - 1'b1: Enabled
15	MIPI_BIT_CLK_RST	RW	0x0	MIPI BIT CLK Reset - 1'b0: Reset - 1'b1: No Reset
14	RSVD	RO	0x0	Reserved
13:12	V2D_FCLK_SEL	RW	0x0	V2D FCLK Clock Select - 2'b00: 499 MHz - 2'b01: 416 MHz - 2'b10: 312 MHz - 2'b11: 624 MHz
11:9	V2D_FCLK_DIV	RW	0x2	V2D FCLK Clock Divide Ratio. V2D_FCLK = clock source / (this field +1)
8	V2D_FCLK_EN	RW	0x0	V2D FCLK Enable - 1'b0: Disabled - 1'b1: Enabled
7	RSVD	RO	0x0	Reserved
6	LCD_HCLK_SWAP_CTRL	RW	0x0	LCD HCLK Swap This field is used to control the HCLK source for the LCD in D1P mode. - 1'b0: Use System fabric clock as the LCD HCLK source in D1P mode - 1'b1: Use Bypass VCTXO clock as LCD HCLK source in D1P mode
5	LCD_HCLK_EN	RW	0x0	LCD HCLK Enable - 1'b0: Disabled - 1'b1: Enabled
4	LCD_SW_RST	RW	0x0	LCD Software Reset - 1'b0: Reset - 1'b1: No reset
3	DSI_ESCCLK_RESET	RW	0x0	DSI ESC Clock Reset - 1'b0: Reset
2	DSI_ESC_EN	RW	0x0	DSI ESC Clock Enable - 1'b1: REF clock enabled - 1'b0: REF clock disabled
1:0	DSI_ESC_SEL	RW	0x0	DSI ESC Clock Select - 2'b00: 52 MHz - 2'b01: 48 MHz

Offset: 0xD4282800+0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 2'b10: 26 MHz - 2'b11: 78 MHz

9.2.4.2.8 LCD_SPI CLOCK/RESET CONTROL REGISTER (PMU_LCD_SPI_CLK_RES_CTRL)

Offset: 0xD4282800+0x48				
Bits	**Field**	**Type**	**Reset**	**Description**
31:15	RSVD	RO	0x0	Reserved
14:12	LCD_SPI_CLK_SEL	RW	0x0	- 3'b000: clk_312m, //div8 - 3'b001: clk_in_416m_pll1, //div6 - 3'b010: clk_in_249m_pll1, //div10 - 3'b011: clk_pll_div11 - 3'b100: clk_pll_div13 - 3'b101: clk_pll_div23 - 3'b110: clk_in_pll2_div3 - 3'b111: clk_in_pll2_div5
11	RSVD	RO	0x0	Reserved
10:8	LCD_SPI_CLK_DIV	RW	0x0	LCD_SPI CLK = clock source/ (this field +1)
7	LCD_SPI_PXCLK_REQ	W1C	0x0	- Write 1 to trigger a frequency change. - The field is cleared by hardware once the frequency change is done.
6	LCD_SPI_ACLK_EN	RW	0x0	- 1'b0: Disabled - 1'b1: Enabled
5	LCD_SPI_BUS_CLK_EN	RW	0x0	- 1'b0: Disabled - 1'b1: Enabled
4	LCD_SPI_BUS_RESET	RW	0x0	- 1'b0: Reset
3	LCD_SPI_HBUS_CLK_EN	RW	0x0	- 1'b0: Disabled - 1'b1: Enabled
2	LCD_SPI_HBUS_RESET	RW	0x0	- 1'b0: Reset
1	LCD_SPI_CLK_EN	RW	0x0	- 1'b0: Disabled - 1'b1: Enabled
0	LCD_SPI_RESET	RW	0x0	- 1'b0: Reset

9.2.4.2.9 LCD CLOCK/RESET CONTROL REGISTER2 (PMU_LCD_CLK_RES_CTRL2)

Offset: 0xD4282800+0x4C

Bits	Field	Type	Reset	Description
31:25	RSVD	RO	0x0	Reserved
24	LCD_PXCLK_BLANK_MSK	RW	0x1	LCD PXCLK Frequency Change wait BLANK signal mask: - 1'b0: Wait for the LCD BLANK signal - 1'b1: Do not wait for the LCD BLANK signal
23:21	LCD_PXCLK_SEL	RW	0x0	LCD PXCLK Select - 3'b000: PLL1_409MHz - 3'b001: PLL1_491MHz - 3'b010: PLL1_614MHz - 3'b011: PLL1_312MHz - 3'b100: 428MHz(PLL2_DIV7) - 3'b101: 375MHz(PLL2_DIV8)
20:17	LCD_PXCLK_DIV	RW	0x2	LCD PXCLK Divide Ratio LCD_PXCLK = clock source/ (this field +1)
16	LCD_PXCLK_EN	RW	0x0	LCD PXCLK Enable - 1'b0: Disable - 1'b1: Enabled
15:10	RSVD	RO	0x0	Reserved
9	LCD_MCLK_RESET	RW	0x1	- 1'b0: Reset
8	RSVD	RO	0x0	Reserved
7:5	LCD_MCLK_SEL	RW	0x0	LCD MCLK Select - 3'b000: PLL1_409 MHz - 3'b001: PLL1_491 MHz - 3'b010: PLL1_614 MHz - 3'b011: PLL1_307 MHz - 3'b100~3'b111: N/A
4:1	LCD_MCLK_DIV	RW	0x2	LCD MCLK Clock Divide Ratio. LCD_MCLK = clock source/ (this field +1)
0	LCD_MCLK_EN	RW	0x0	LCD MCLK Enable. - 1'b0: Disabled - 1'b1: Enabled > **Note.** This clock is also used for camera AHB Clock. The camera can only enable this clock but cannot change its frequency.

9.2.4.2.10 CCIC CLOCK/RESET CONTROL REGISTER (PMU_CCIC_CLK_RES_CTRL)

Offset: 0xD4282800+0x50

Bits	Field	Type	Reset	Description
31	CCIC_ISP_HCLK_SWAP_CTRL	RW	0x0	CCIC ISP HCLK Swap Control. This field is used to control the HCLK source for CCIC and ISP in D1P mode. - 1'b0: Use System fabric clock as the HCLK source - 1'b1: Use Bypass VCTXO clock as the HCLK source
30	MASK_ISP_BLANK_CHECK	RW	0x0	ISP FCLK Frequency Change Mask ISP blank Check. This field is used to mask the ISP blank indication check during ISP FCLK frequency change. - 1'b1: The ISP FCLK FC will wait for the blank signal - 1'b0: Mask the ISP blank check (i.e., does not wait for the blank signal)
29	ISP_BLANK_CHECK_MODE	RW	0x0	ISP FCLK Frequency Change ISP blank Check Mode. This field is used to select the ISP blank mode for ISP FCLK frequency change. - 1'b1: Use V blank flag - 1'b0: Use H blank flag
28:26	RSVD	RO	0x0	Reserved
25:23	CCICI_CLK4X_SEL	RW	0x00	CCIC4x Controller Function Clock Source Select. - 3'b000: PLL1_491 MHz - 3'b001: PLL1_409 MHz - 3'b010: PLL1_614 MHz - 3'b011: PLL1_819 MHz - 3'b100: PLL2_div2 - 3'b101: PLL2_div3 - 3'b110: PLL2_div4 - 3'b111: PLL1_1248 MHz
22:21	RSVD	RO	0x0	Reserved
20:18	CICIC_CLK4X_DIV	RW	0x1	CI Function Clock Divide Ratio. ci_fnc_clk = CI_FNC_CLK_DIV / (this field +1)
17:16	RSVD	RO	0x0	Reserved
15	CCIC_CLK4X_FC_REQ	W1C	0x0	CCIC Function clk4x Frequency Change Request. - Write 1 to trigger a frequency change. - The field is cleared by hardware once the frequency change is done.
14:8	RSVD	RO	0x0	Reserved
7	CCIC1_PHYCLK_SEL	RW	0x0	CCIC1 PHY Clock Select. - 1'b0: 104 MHz - 1'b1: 52 MHz
6	RSVD	RO	0x0	Reserved

Offset: 0xD4282800+0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
5	CCIC1_PHYCLK_EN	RW	0x0	CCIC1 PHY Clock Enable - 1'b1: PHY clock enabled - 1'b0: PHY clock disabled
4	CCIC_CLK4X_EN	RW	0x0	CMOS Camera Interface Controller Peripheral Clock Enable - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3	RSVD	RO	0x0	Reserved
2	CCIC1_PHYCLK_RST	RW	0x0	CCIC1 PHY Clock Reset - 1'b0: Reset Note: This clock is also used for DPHY reset.
1	CCIC_CLK4X_RST	RW	0x0	CMOS Camera Interface Controller Peripheral Reset - 1'b0: Reset
0	RSVD	RO	0x0	Reserved

9.2.4.2.11 SDH0 CLOCK/RESET CONTROL REGISTER (PMU_SDH0_CLK_RES_CTRL)

Offset: 0xD4282800+0x54				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	RO	0x0	Reserved
11	SDH0_CLK_FC_REQ	W1C	0x0	SDH0 Clock Frequency Change Request. - Write 1 to force SDH0_CLK_DIV to work. - The field is cleared by hardware once the clock switch is done.
10:8	SDH0_CLK_DIV	RW	0x1	SDH0 Clock Frequency Divisor. SDH0_CLK = SDH0 source clock/(SDH0_CLK_DIV + 1)
7:5	SDH0_CLK_SEL	RW	0x0	SDH0 Clock source Select. - 3'b000: 409MHz(clk_in_416m_pll1) - 3'b001: 614MHz(clk_in_pll1_2x) - 3'b010: PLL2_div8 - 3'b011: PLL2_div5 - 3'b100: PLL1_div11 - 3'b101: PLL1_div13 - 3'b110: PLL1_div23 - 3'b111: Reserved
4	SDH0_CLK_EN	RW	0x0	SDH0 Peripheral Clock Enable. - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3	SDH_AXICLK_EN	RW	0x0	All SDH AXI Clock Enable

Offset: 0xD4282800+0x54				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field is used to enable the AXI click for all 3 SDH modules. - 1: AXI clock enabled - 0: AXI clock disabled
2	RSVD	RO	0x0	Reserved
1	SDH0_RST	RW	0x0	SDH0 Peripheral Reset. - 1'b0: Reset
0	SDH_AXI_RST	RW	0x0	All SDH AXI Reset This field is used to perform an AXI reset for all 3 SDH modules.

9.2.4.2.12 SDH1 CLOCK/RESET CONTROL REGISTER (PMU_SDH1_CLK_RES_CTRL)

Offset: 0xD4282800+0x58				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	RO	0x0	Reserved
11	SDH1_CLK_FC_REQ	W1C	0x0	SDH1 Clock Frequency Change Request. - Write 1 to force SDH1_CLK_DIV to work. - The field is cleared by hardware once the clock switch is done.
10:8	SDH1_CLK_DIV	RW	0x1	SDH1 Clock Frequency Divisor. SDH1_CLK = SDH1 source clock/(SDH0_CLK_DIV + 1)
7:5	SDH1_CLK_SEL	RW	0x0	SDH1 Clock source Select. - 3'b000: 409MHz(clk_in_416m_pll1) - 3'b001: 614MHz(clk_in_pll1_2x) - 3'b010: PLL2_div8 - 3'b011: PLL2_div5 - 3'b100: PLL1_div11 - 3'b101: PLL1_div13 - 3'b110: PLL1_div23 - 3'b111: Reserved
4	SDH1_CLK_EN	RW	0x0	SDH1 Peripheral Clock Enable. - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3:2	RSVD	RO	0x0	Reserved
1	SDH1_RST	RW	0x0	SDH1 Peripheral Reset. - 1'b0: Reset
0	RSVD	RO	0x0	Reserved

9.2.4.2.13 USB CLOCK/RESET CONTROL REGISTER (PMU_USB_CLK_RES_CTRL)

Offset: 0xD4282800+0x5C

Bits	Field	Type	Reset	Description
31	VBUS_DLY_CNT_EN	RW	0x0	Enable for VBUS fall edge debounce
30:23	DLY_CNT_REG	RW	0x0	Debounce Counter Register: - Configures the debounce duration to detect the VBUS fall edge signal. - Valid only when vbus_dly_cnt_en is set to 1.
22:18	RSVD	RO	0x0	Reserved
17	USBP1_AHB_PROT_IN_SUSP	RW	0x0	- 1'b1: Gated access to the USB port controller and PHY registers - Software sets this bit to 1 before suspending USB, and clear it before reactivating USB
16	USB_AHB_PROT_IN_SUSP	RW	0x0	- 1'b1: Gated access to the USB port controller and PHY registers - Software sets this bit to 1 before suspending USB, and clear it before reactivating USB
15:12	RSVD	RO	0x0	Reserved
11	USB3_0_PHY_RESETN	RW	0x0	dedicated now.
10	USB3_0_VCC_RESETN	RW	0x0	usb3_0_vcc_resetn
9	USB3_0_AHB_RSTN	RW	0x0	usb3_0_ahb_rstn
8	USB3_0_BUS_CLK_EN	RW	0x0	usb3_0_bus_clk_en
7:6	RSVD	RO	0x0	Reserved
5	USBP1_AXICLK_EN	RW	0x0	USBP1 AXI Clock Enable - 1'b1: AXI clock enabled - 1'b0: AXI clock disabled
4	USBP1_AXI_RST	RW	0x0	USBP1 AXI Reset - 1'b0: Reset - 1'b1: De-reset
3:2	RSVD	RO	0x0	Reserved
1	USB_AXICLK_EN	RW	0x0	USB AXI Clock Enable - 1'b1: AXI clock enabled - 1'b0: AXI clock disabled
0	USB_AXI_RST	RW	0x0	USB AXI Reset. - 1'b0: Reset

Offset: 0xD4282800+0x5C				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b1: De-reset

9.2.4.2.14 QSPI CLOCK/RESET CONTROL REGISTER (PMU_QSPI_CLK_RES_CTRL)

Offset: 0xD4282800+0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31:13	RSVD	RO	0	Reserved
12	QSPI_CLK_FC_REQ	RWAC	0x0	- Write 1 to force QSPI_CLK_SEL to work. - The field is cleared by hardware once the clock switch is done.
11:9	QSPI_CLK_DIV	RW	0x7	QSPI Clock division ratio. QSPI_Clock_Freq = QSPI_CLK_SEL Freq/(QSPI_CLK_DIV+1)
8:6	QSPI_CLK_SEL	RW	0x5	- 3'b000: 409 MHz(PLL1_div6) - 3'b001: 375 MHz(PLL2_div8) - 3'b010: 307 MHz(PLL1_div8) - 3'b011: 245 MHz(PLL1_div10) - 3'b100: 223 MHz (PLL1_div11) - 3'b101: 106 MHz(PLL1_div23) - 3'b110: 495 MHz(PLL1_div5) - 3'b111: 189 MHz(PLL1_div13)
5	RSVD	RO	0	Reserved
4	QSPI_CLK_EN	RW	0x1	QSPI Function Clock Enable - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3	QSPI_BUS_CLK_EN	RW	0x1	QSPI Bus Clock Enable - 1'b1: Bus clock enabled - 1'b0: Bus clock disabled
2	RSVD	RO	0	Reserved
1	QSPI_CLK_RST	RW	0x1	QSPI clk Reset - 1'b0: Reset
0	QSPI_BUS_RST	RW	0x1	QSPI_BUS_CLK Reset. - 1'b0: Reset

9.2.4.2.15 DMA CLOCK/RESET CONTROL REGISTER (PMU_DMA_CLK_RES_CTRL)

Offset: 0xD4282800+0x64				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	RO	0x0	Reserved
3	DMA_AXICLK_EN	RW	0x0	DMA AXI Clock Enable - 1'b1: AXI clock enabled - 1'b0: AXI clock disabled
2:1	RSVD	RO	0x0	Reserved
0	DMA_AXI_RST	RW	0x0	DMA AXI Reset

9.2.4.2.16 AES CLOCK/RESET CONTROL REGISTER (PMU_AES_CLK_RES_CTRL)

Offset: 0xD4282800+0x68				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6	WTM_CLK_SEL	RW	0x0	WTM Clock Select - 1'b0: 208 MHz - 1'b1: 104 MHz
5	WTM_CLK_EN	RW	0x0	WTM Clock Enable - 1'b0: WTM clock disabled - 1'b1: WTM clock enabled
4	WTM_RST	RW	0x0	WTM Clock Reset
3:0	RSVD	RO	0x0	Reserved

9.2.4.2.17 VPU CLOCK/RESET CONTROL REGISTER (PMU_VPU_CLK_RES_CTRL)

Offset: 0xD4282800+0xA4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	RO	0x0	Reserved
21	VPU_CLK_FC _REQ	W1C	0x0	VPU Clock Frequency Change Request - Write 1 to trigger a frequency change. - The field is cleared by hardware once the trigger a frequency change is done.
20:16	RSVD	RO	0x0	Reserved
15:13	VPU_CLK_DIV	RW	0x1	VPU Function Clock Divide Ratio. VPU_CLK = VPU_CLK_SEL / (this field +1).

Offset: 0xD4282800+0xA4				
Bits	**Field**	**Type**	**Reset**	**Description**
				> **Note.** Divider only used for clock source 0, 1, 2 and 3.
12:10	VPU_CLK_SEL	RW	0x0	VPU Function Clock Select. - 3'b000: PLL1_614MHz - 3'b001: PLL1_491MHz - 3'b010: PLL1_819MHz - 3'b011: PLL1_409Mhz - 3'b100: PLL3_DIV6 - 3'b101: PLL2_DIV3 - 3'b110: PLL2_DIV4 - 3'b111: PLL2_DIV5
9:4	RSVD	RO	0x0	Reserved
3	VPU_CLK_EN	RW	0x0	VPU Function Clock Enable - 1'b1: clock enabled - 1'b0: clock disabled
2:1	RSVD	RO	0x0	Reserved
0	VPU_RST	RW	0x0	VPU Reset. - 1'b1: Release reset - 1'b0: Reset

9.2.4.2.18 DDR MEMORY CONTROLLER HARDWARE SLEEP TYPE REGISTER (PMU_MC_HW_SLP_TYPE)

Offset: 0xD4282800+0xB0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	RO	0x0	Reserved
23	DCLK_BYPASS_FC_REQ	W1C	0x0	DCLK Bypass Clock Frequency Change Request. - Write 1 to trigger a frequency change. - The field is cleared by hardware once the trigger a frequency change is done.
22	DCLK_BYPASS_CLK_EN	RW	0x1	DCLK Bypass Clock Enable. - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
21	DCLK_BYPASS_RST	RW	0x1	DCLK Bypass Clock Reset. - 1'b0: Reset
20:19	DCLK_BYPASS_SEL	RW	0x0	DCLK Bypass Clock Select. - 2'b00: PLL1 312 MHz

Offset: 0xD4282800+0xB0				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 2'b01: PLL1 416 MHz - 2'b10: 24MHz - 2'b11: Reserved
18:16	DCLK_BYPASS_DIV	RW	0x1	DCLK Bypass Clock Divider. DCLK Bypass Clock = DCLK_BYPASS_SEL / (this field +1). > **Note.** Divider only used for Clock source 0 and 1.
15:11	RSVD	RO	0x0	Reserved
10	MC_REG_TABLE_EN	RW	0x1	Frequency Change table mask. - 1'b0: Enabled - 1'b1: Disabled
9	FREQ_PLL_CHG_MODE	RW	0x0	Determines the behavior of the DDRPHY PLL during a frequency change. - 1'b0: Perform a full PLL switch, including VCO reconfiguration, to transition to a new frequency. - 1'b1: Keep the VCO frequency unchanged and only update the clock divider or input clock source.
8:7	RSVD	RO	0x0	Reserved
6:3	MC_REG_TABLE_NUM	RW	0x0	Memory Controller Register Table Number: - bit[3]: 1'b1: Frequency change occurs within the same timing table - bit [2]: 1'b1: The target frequency is a high frequency - bit[1:0]: Specifies the target timing table number for the memory controller.
2:0	RSVD	RO	0x0	Reserved

9.2.4.2.19 PLL CLOCK SELECT STATUS REGISTER (PMU_PLL_SEL_STATUS)

Offset: 0xD4282800+0xC4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:14	RSVD	RO	0x0	Reserved
13:11	AP_C1_PLL_SEL	RO	0x0	Core Clock selection - 3'b000: 614M. - 3'b001: 819M. - 3'b010: 409M. - 3'b011: 491M. - 3'b100: 1228M. - 3'b101: PLL3_DIV3(1066M) - 3'b110: PLL2_DIV3(1000M)

Offset: 0xD4282800+0xC4				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 3'b111: PLL3_DIV2(1600M)
10:8	AP_C0_PLL_SEL	RO	0x0	Core Clock selection - 3'b000: 614MHz. - 3'b001: 819MHz. - 3'b010: 409MHz. - 3'b011: 491MHz. - 3'b100: 1228MHz. - 3'b101: PLL3_DIV3(1066MHz) - 3'b110: PLL2_DIV3(1000MHz) - 3'b111: PLL3_DIV2(1600MHz)
7:6	ACLK_PLL_SEL	RO	0x0	ACLK source selection - 1'b0: 249MHz - 1'b1: 312MHz
5:0	RSVD	RO	0x0	Reserved

9.2.4.2.20 GPU CLOCK/RESET CONTROL REGISTER (PMU_GPU_CLK_RES_CTRL)

Offset: 0xD4282800+0xCC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:21	RSVD	RO	0x0	Reserved
20:18	GPU_CLK_SEL	RW	0x0	GPU Clock Select - 3'b000: PLL1_614MHz - 3'b001: PLL1_491MHz - 3'b010: PLL1_819MHz - 3'b011: PLL1_409MHz - 3'b100: PLL3_DIV6 - 3'b101: PLL2_DIV3 - 3'b110: PLL2_DIV4 - 3'b111: PLL2_DIV5
17:16	RSVD	RO	0x0	Reserved
15	GPU_FNC_FC_REQ	W1C	0x0	GPU Clock Frequency Change Request - Write 1 to trigger a frequency change. - The field is cleared by hardware once the trigger a frequency change is done.
14:12	GPU_CLK_DIV	RW	0x1	GPU Clock Divider. GPU_fnc_clk = GPU_CLK_SEL / (this field +1). **Note**. Divider only used for clock source 0, 1 and 2.
11:5	RSVD	RO	0x0	Reserved
4	GPU_CLK_EN	RW	0x0	GPU Clock Enable.

Offset: 0xD4282800+0xCC				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3:2	RSVD	RO	0x0	Reserved
1	GPU_RST	RW	0x0	GPU Reset. - 1'b1: Release Reset - 1'b0: Reset
0	RSVD	RO	0x0	Reserved

9.2.4.2.21 SDH2 CLOCK/RESET CONTROL REGISTER (PMUA_SDH2_CLK_RES_CTRL)

Offset: 0xD4282800+0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	RO	0x0	Reserved
11	SDH2_CLK_FC_REQ	W1C	0x0	SDH2 Clock Frequency Change Request. When this field is set to 1, it will force SDH2_CLK_DIV to work. This field will be automatically cleared by hardware when the clock switch is done. - Write 1 to force SDH2_CLK_DIV to work. - The field is cleared by hardware once forcing SDH2_CLK_DIV to work is done.
10:8	SDH2_CLK_DIV	RW	0x1	SDH2 Clock Frequency Divisor 0-7. SDH2_CLK = SDH2 source clock/(SDH2_CLK_DIV + 1)
7:5	SDH2_CLK_SEL	RW	0x0	SDH2 Clock source Select - 3'b000: 409MHz - 3'b001: 614MHz - 3'b010: PLL2_div8 - 3'b011: 819MHz - 3'b100: PLL1_div11 - 3'b101: PLL1_div13 - 3'b110: PLL1_div23 - 3'b111: Reserved
4	SDH2_CLK_EN	RW	0x0	SDH2 Peripheral Clock Enable - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3:2	RSVD	RO	0x0	Reserved
1	SDH2_RST	RW	0x0	SDH2 Peripheral Reset - 1'b0: Reset
0	RSVD	RO	0x0	Reserved

9.2.4.2.22 MEMORY CONTROLLER AHB REGISTER (PMUA_MC_CTRL)

Offset: 0xD4282800+0xE8				
Bits	**Field**	**Type**	**Reset**	**Description**
31	DFC_D1P_BLOCK	RW	0x0	- 1'b1: Disable Dynmanic Frequency Change during D1P - 1'b0: Enable Dynmanic Frequency Change during D1P
30	DDR_DPHY_PU	RW	0x1	DDR DPHY PU control. > **Note.** Do not modify this value; it must always remain **high**.
29	MC_CLK_GATE_BYPS	RW	0x0	Memory Controller Clock Gating Bypass: - 1'b1: Bypass MCK_root clock gating during low power state. - 1'b0: No Bypass, MCK_root is gated during low power state (for power saving)
28:02	RSVD	RO	0	Reserved
1	MC_AHBCLK_EN	RW	0x0	Memory Controller AHB Clock Enable. - 1'b1: AHB clock enabled - 1'b0: AHB clock disabled
0	MC_HCLK_RST	RW	0x0	Memory Controller HCLK Reset. - 1'b0: Reset

9.2.4.2.23 AP CLOCK CONTROL REGISTER2 (PMU_CC2_AP)

This register is used to trigger CPU CORE reset as follows:

- **Core Power-On Reset**
  - Initializes all processor logic, including:
    - CPU Debug logic
    - Breakpoint and watchpoint logic
  - Affects all logic within the processor power domains
- **Core Software Reset**
  - Resets processor logic within the processor power domains, but excludes:
    - Debug logic
    - Breakpoint and watchpoint logic
- **Reset at different levels:**
  - CORE Level:
    - Resets only the debug, breakpoint, and watchpoint logic within the processor power domain
  - MP (Multiprocessor) Level:
    - Additionally resets the debug logic for each processor in the debug power domain.

Offset: 0xD4282800+0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	RO	0x0	Reserved
29	MPSUB_DBG_RST	RW	0x0	MP Debug Reset. This field is used to reset the MP debug/Coresight logic, including core debug logic. - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
28	C1_MPSUB_SW_RST	RW	0x0	Cluster1 Reset. This field is used to reset Cluster1 logic except debug/Coresight logic. - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
27	RSVD	RO	0x0	Reserved
26	CPU7_SW_RST	RW	0x0	CPU7 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
25	CPU7_POR_RST	RW	0x0	CPU7 Core Power on Reset. This field is used to reset CPU7 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted
24	RSVD	RO	0x0	Reserved
23	CPU6_SW_RST	RW	0x0	CPU6 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
22	CPU6_POR_RST	RW	0x0	CPU6 Core Power on Reset. This field is used to reset CPU6 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted
21	RSVD	RO	0x0	Reserved
20	CPU5_SW_RST	RW	0x0	CPU5 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
19	CPU5_POR_RST	RW	0x0	CPU5 Core Power on Reset. This field is used to reset CPU5 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted
18	RSVD	RO	0x0	Reserved
17	CPU4_SW_RST	RW	0x0	CPU4 Core Software Reset

Offset: 0xD4282800+0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
16	CPU4_POR_RST	RW	0x0	CPU4 Core Power on Reset. This field is used to reset CPU4 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted
15:13	RSVD	RO	0x0	Reserved
12	C0_MPSUB_SW_RST	RW	0x0	Cluster0 Reset. This field is used to reset Cluster0 logic except debug/Coresight logic. - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
11	RSVD	RO	0x0	Reserved
10	CPU3_SW_RST	RW	0x0	CPU3 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
9	CPU3_POR_RST	RW	0x0	CPU3 Core Power on Reset. This field is used to reset CPU3 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted
8	RSVD	RO	0x0	Reserved
7	CPU2_SW_RST	RW	0x0	CPU2 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
6	CPU2_POR_RST	RW	0x0	CPU2 Core Power on Reset. This field is used to reset CPU2 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted
5	RSVD	RO	0x0	Reserved
4	CPU1_SW_RST	RW	0x0	CPU1 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
3	CPU1_POR_RST	RW	0x0	CPU1 Core Power on Reset. This field is used to reset CPU1 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted

Offset: 0xD4282800+0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
2	RSVD	RO	0x0	Reserved
1	CPU0_SW_RST	RW	0x0	CPU0 Core Software Reset - 1'b1: Reset is asserted - 1'b0: Reset is de-asserted
0	CPU0_POR_RST	RW	0x0	CPU0 Core Power on Reset. This field is used to reset CPU0 all logic, including debug logic. - 1'b1:Reset is asserted - 1'b0:Reset is de-asserted

9.2.4.2.24 EMMC5.0 CLOCK/RESET CONTROL REGISTER (PMUA_EM_CLK_RES_CTRL)

Offset: 0xD4282800+0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
31	EM_PHY_TMS_SW	RW	0x0	Control the bypass/test mode for EM5.0 EPHY pad: - 1'b1: bypass/test mode enabled - 1'b0: bypass/test disabled
30	EM_PHY_TOE_SW	RW	0x0	Control the selection of Test Data Output (TDO) ports: - 1'b1: TDO ports are sampled with FCLK (low speed only). - 1'b0: TDO ports are directly from receiver
29	EM_PHY_VREF	RW	0x0	SW controls the EPHY VREF port value
28	EM_PHY_V18EN	RW	0x0	EPHY IO 1.8 enable SW control. This bit controls the 1.8V power supply for EPHY IO: - 1'b1: 1.8V enabled - 1'b0: 1.8V disabled
27	EM_PHY_LP_SEL	RW	0x0	EPHY Low Power (LP) mode SW control. This bit controls the Low Power Mode for EPHY: - 1'b1: LP mode enabled - 1'b0: LP mode disabled
26:16	RSVD	RO	0	Reserved
15	EM_1248M_CLK_EN	RW	0x0	EM5.0 1248 MHz Input Clock Enable This bit Controls the 1248 MHz clock source for the EM clock divider: - 1'b0: Disable EM 1248M clock divider clock source - 1'b1: Enable EM 1248M clock divider clock source
14:12	EM_1248M_CLK_DIV	RW	0x0	EMMC5.0 1248 MHz Input Clock Frequency Divisor 0x0 to 0x7. EM_1248M_DIV5 = 1248Mhz source clock/(EM_1248_CLK_DIV + 1)

Offset: 0xD4282800+0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
11	EM_CLK_FC_REQ	RW	0x0	- Write 1 to force EM_CLK_DIV to work. - The field is cleared by hardware once forcing EM_CLK_DIV to work is done.
10:8	EM_CLK_DIV	RW	0x0	EM_CLK_DIV 0-7. EM_CLK = EM source clock/(EM_CLK_DIV + 1)
7:6	EM_CLK_SEL	RW	0x0	EM Clock source Selection - 1'b0: 416MHz(clk_in_416m_PLL1) - 1'b1: 624MHz(clk_in_PLL1_2x) - 2'b10: 48MHz - 2'b11: 800MHz(clk_in_PLL2)
5	RSVD	RO	0	Reserved
4	EM_CLK_EN	RW	0x0	EM Peripheral Clock Enable. This bit controls the peripheral clock for EM: - 1'b1: Peripheral clock enabled - 1'b0: Peripheral clock disabled
3	EM_AXICLK_EN	RW	0x0	EM AXI Clock Enable. This bit controls the AXI Clock for EM - 1'b1: AXI clock enabled - 1'b0: AXI clock disabled
2	RSVD	RO	0	Reserved
1	EM_RST	RW	0x0	EM Peripheral Reset - 1'b0: Reset > **Note.** The eMMC5.0 controller primarily uses EM_AXI_RST instead of this reset.
0	EM_AXI_RST	RW	0x0	EM AXI Reset.

9.2.4.2.25 USB PHY CONTROL REGISTER0 (PMUA_USB_PHY_CTRL0)

Offset: 0xD4282800+0x110				
Bits	Field	Type	Reset	Description
31:4	RSVD	RO	0x0	Reserved
3	COMBO_PHY_SEL	RW	0x0	It specifies the function of the shared PUPHY - 1'b0: Indicate the shared PUPHY is used for PCIe. - 1'b1 : Indicate the shared PUPHY is used for USB3.
2:0	RSVD	RO	0x0	Reserved

9.2.4.2.26 AUDIO CLOCK RESET ENABLE REGISTER (PMU_AUDIO_CLK_RES_CTRL)

Offset: 0xD4282800+0x14C

Bits	Field	Type	Reset	Description
31	AUDIO_PWR_STATUS	RO	0x1	Audio Power domain status It indicates the power status of the Audio PMU: - 1'b1: Audio PMU is in Power-On status - 1'b0: Audio PMU is in Power-Off status
30	USE_SOFT_RST	RW	0x0	This bit set to force audio use soft reset
29	AUDIO_AUTO_POWER_ON_OFF_TRIGGER_IN_HARDWARE_MODE	RW	0x1	Audio Auto Power On/Off Trigger It controls the automatic power-on or power-off request for Audio Power Island: - 1'b1: Triggers request to power-up the Audio Power Island - 1'b0: Triggers request to power-down Audio Power Island
28	AP_POWER_CTL_AUDIO_AUTHO	RW	0x1	AP Power Control Audio Authority. It specifies whether the AP (Application Processor) can control Audio power - 1'b1: AP has control over Audio power - 1'b0: AP can not control Audio power, and Audio PMU manages the audio power switch.
27:24	RSVD	RO	0x0	Reserved
23	AUDIO_PWR_STATUS	RO	0x1	Status for Audio PMU. It indicates the power status of the Audio PMU - 1'b0: Audio PMU is in Power-Off status. - 1'b1: Audio PMU is in Power-On status
22:20	RSVD	RO	0x0	Reserved
19	LOG_EN	RW	0x0	Debug Information Recording Enable: It controls whether debug information (e.g., PC value, bus status) is recorded. - 1'b0: Disabled - 1'b1: Enabled
18	ADSP_EN	RW	0x0	It controls whether the DSP is active or not 1'b0: Hold DSP (DSP is in wait status) 1'b1: Active DSP (DSP is running)
17:16	RSVD	RO	0x0	Reserved
15	AUDIO_FC_REQ	W1C	0x0	Audio island main clock Frequency Change Request. - 1'b1: Triggers a frequency change. - The field is cleared by hardware once the frequency change is done
14	FORCE_AUD_PWR_OFF	RW	0x0	It controls whether to force the audio power to turn

Offset: 0xD4282800+0x14C				
Bits	**Field**	**Type**	**Reset**	**Description**
				off - 1'b1: Force audio power off - 1'b0: Do not force audio power off
13	FORCE_AUD_PWR_ON	RW	0x0	It controls whether to force the audio power to turn on - 1'b1: Force audio power on - 1'b0: Do not force audio power on
12	AUDIO_CLK_EN	RW	0x0	Audio clock enable - 1'b1: Enabled
11	CUR_PWR_MST	RO	0x0	Audio Power Control Authority Indicator It indicates which component controls the audio power switch - 1'b1: **AP** controls audio power switch - 1'b0: **Audio PMU** controls audio power switch
10	AUDIO_HW_CKG_BYPASS	RW	0x0	Audio Always On Domain reset This bit should be always set to 1 after the silicon power up
9:7	AUDIO_CLK_SEL	RW	0x0	Audio Main Clock Selection It controls the clock source for the audio main clock - 3'b000: PLL1 245.76MHz - 3'b001: PLL1 312MHz - 3'b010: Reserved - 3'b011: PLL1 416MHz - 3'b100-3'b111: Reserved
6:4	AUDIO_CLK_DIV	RW	0x0	Clock divider control for audio main clock. Audio Main Clock = selected Clock/(AUDIO_CLK_DIV+1) when AUDIO_CLK_SEL=1 or 3, AUDIO_CLK_DIV must be greater than 0
3	AUDIO_APMU_RESET	RW	0x0	Audio APMU reset This bit should be always set to 1 after the silicon power up
2	AUD_MCU_CORE_RESET	RW	0x0	Soft AUD_MCU core_rst. It controls the reset state of the AUD_MCU core - 1'b0: Reset - 1'b1: Release reset
1	RSVD	RO	0x0	Reserved
0	AUDIO_SYS_RESET	RW	0x0	Soft audio island system reset It controls the reset state of the audio island system - 1'b0: Reset - 1'b1: Release reset

9.2.4.2.27 MP DCLK DYNAMIC FREQ CHANGE CONTROL REGISTER (DFC_AP)

Offset: 0xD4282800+0x180				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	RO	0x0	Reserved
3:1	FL	RW	0x0	DCLK Frequency Level in Active Mode It specifies the required frequency level for DCLK in active mode, as determined by MP.
0	DFC_REQ	RW	0x0	DCLK HFC Request in Active Mode. It is used by software to signal that a DCLK HFC is required for the MP in active mode. - Writing 1 to this bit to trigger this request, and the hardware automatically clears it once the request action is done. - Writing 0 to this bit has no effect.

9.2.4.2.28 MP DCLK HARDWARE FREQ CHANGE STATUS REGISTER (DFC_STATUS)

Offset: 0xD4282800+0x188				
Bits	**Field**	**Type**	**Reset**	**Description**
31:11	RSVD	RO	0x0	Reserved
10:7	DFC_CAUSE	RO	0x0	DCLK DFC Cause. This field indicates the reason for the current DFC of the DCLK. This field is valid only when the <DCLK_DFC_STATUS> field is set to 1. - 4'b0001: MP Triggered DFC - Other values: Low power mode entry/exit triggered DFC
6:4	TFL	RO	0x0	Target Frequency Level of DCLK
3:1	CFL	RO	0x0	Current Frequency Level of DCLK
0	DFC_STATUS	RW	0x0	DCLK DFC Status It indicates the status of DFC for the DCLK, specifying whether a DFC is active: - 1'b0: No ongoing DFC, or the DFC has just completed - 1'b1: A DFC is currently active

9.2.4.2.29 DCLK FREQ LEVEL 0 CONTROL REGISTER (DFC_LEVEL_X)

Offset: 0xD4282800+0x190/0x194/0x198/0x19C/0x1A0/0x1A4/0x1A8/0x1AC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	RO	0x0	Reserved
15:14	PLL_CLK_DIV	RW	0x0	PLL Clock Divider Configuration - 2'b11: DIV4 - 2'b10: DIV3 - 2'b01: DIV2 - 2'b00: DIV1
13	DCLK_POSTDIV	RW	0x0	PLL bypass Clock Divider Configuration. 1'b1: Divide the clock by 2 1'b0: No division (bypass)
12	PLL_BYPASS_SEL	RW	0x0	PLL Bypass Selection - 1'b1: Bypass PLL - 1'b0: Enable PLL
11:9	RSVD	RO	0x0	Reserved
8	PLL_SEL	RW	0x0	PLL Selection It indicates which PLL (PLL1 or PLL2) is selected for operation. It is effective only when FREQ_PLL_CHG_MODE is set to 1. - 1'b0: PLL2 is selected - 1'b1：PLL1 is selected
7	PLL_SEL_OFF	RW	0x0	PLL Selection (for disabling Unused PLL) It controls the disabling of one of the PLLs. It is effective only when FREQ_PLL_CHG_MODE is set to 1. - 1’b0: Do not disable the PLL - 1'b1: Disable unused PLL. 1. If PLL_SEL is 1, it will disable PLL2. 2. If PLL_SEL is 0, it will disable PLL1.
6:4	MC_TABLE_NUM	RW	0x0	This field is used to control frequency changes by selecting from pre-defined configuration - bit[6]: 1'b1: The target frequency is high frequency - bit[5:4]: Specifies the target timing table number
3:0	VL	RW	0x0	Required Voltage Level

9.2.4.2.30 HDMI CLOCK/RESET CONTROL REGISTER (PMU_HDMI_CLK_RES_CTRL)

Offset: 0xD4282800+0x1B8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	RO	0x0	Reserved
29	HDMI_MCLK_FC_REQ	W1C	0x0	HDMI MCLK FC Request - Write 1 to trigger the frequency change. - The field is cleared by hardware once the frequency change is done.
28:10	RSVD	RO	0x0	Reserved
9	HDMI_MCLK_RESET	RW	0x1	- 1'b0: Reset
8	RSVD	RO	0x0	Reserved
7:5	HDMI_MCLK_SEL	RW	0x0	HDMI MCLK Clock Selection - 3'b000: PLL1_409MHz - 3'b001: PLL1_491MHz - 3'b010: PLL1_614MHz - 3‘b011: PLL1_307MHz - 3'b100-3'b111: NA
4:1	HDMI_MCLK_DIV	RW	0x2	HDMI MCLK Clock Divide Ratio. HDMI_MCLK = clock source/ (this field +1)
0	HDMI_MCLK_EN	RW	0x0	HDMI MCLK Enable. - 1'b0: Disabled - 1'b1: Enabled

9.2.4.2.31 CMOS CAMERA INTERFACE CONTROLLER 2 DYNAMIC CLOCK GATE CONTROL REGISTER (PMU_CCIC2_CLK_GATE_CTRL)

Offset: 0xD4282800+0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	CCIC2_GATE_CSI_CLK_STATIC	RW	0x3	CCIC2 CSI Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
29:28	CCIC2_GATE_CLK4X_STATIC	RW	0x3	CCIC2 clk4x Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
27:26	CCIC2_GATE_CLK1X_	RW	0x3	CCIC2 clk1x Static Clock Gate Control

Offset: 0xD4282800+0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
	STATIC			- 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
25:24	CCIC2_GATE_HCLK_STATIC	RW	0x3	CCIC2 hclk Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
23:22	CCIC2_GATE_ACLK_STATIC	RW	0x3	CCIC2 aclk Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
21:16	RSVD	RO	0x0	Reserved
15:14	CCIC2_GATE_LANE3_CLK_DYNAMIC	RW	0x3	CCIC2 CSI Lane 3 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
13:12	CCIC2_GATE_LANE2_CLK_DYNAMIC	RW	0x3	CCIC2 CSI Lane 2 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
11:10	CCIC2_GATE_LANE1_CLK_DYNAMIC	RW	0x3	CIC2 CSI Lane 1 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
9:8	CCIC2_GATE_LANE0_CLK_DYNAMIC	RW	0x3	CCIC2 CSI Lane 0 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
7:6	CCIC2_GATE_CSI_CLK_DYNAMIC	RW	0x3	CCIC2 CSI Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock

Offset: 0xD4282800+0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 2'b11: Free running
5:4	CCIC2_GATE_AHB_CLK_DYNAMIC	RW	0x3	CCIC2 ahb Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
3:2	CCIC2_GATE_PIP_CLK_DYNAMIC	RW	0x3	CCIC2 axi Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
1:0	CCIC2_GATE__AXI_CLK_DYNAMIC	RW	0x3	CCIC2 axi Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running

9.2.4.2.32 CMOS CAMERA INTERFACE CONTROLLER 3 DYNAMIC CLOCK GATE CONTROL REGISTER (PMU_CCIC3_CLK_GATE_CTRL)

Offset: 0xD4282800+0x1C0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	CCIC3_GATE_CSI_CLK_STATIC	RW	0x3	CCIC3 CSI Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
29:28	CCIC3_GATE_CLK4X_STATIC	RW	0x3	CCIC3 clk4x Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
27:26	CCIC3_GATE_CLK1X_STATIC	RW	0x3	CCIC3 clk1x Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
25:24	CCIC3_GATE_HCLK_STATIC	RW	0x3	CCIC3 hclk Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock

Offset: 0xD4282800+0x1C0				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 2'b10: Stop clock - 2'b11: Free running
23:22	CCIC3_GATE_ACLK_STATIC	RW	0x3	CCIC3 aclk Static Clock Gate Control - 2'b00: Stop clock - 2'b01: Stop clock - 2'b10: Stop clock - 2'b11: Free running
21:16	RSVD	RO	0x0	Reserved
15:14	CCIC3_GATE_LANE3_CLK_DYNAMIC	RW	0x3	CCIC3 CSI Lane 3 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
13:12	CCIC3_GATE_LANE2_CLK_DYNAMIC	RW	0x3	CCIC3 CSI Lane 2 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
11:10	CCIC3_GATE_LANE1_CLK_DYNAMIC	RW	0x3	CIC2 CSI Lane 1 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
9:8	CCIC3_GATE_LANE0_CLK_DYNAMIC	RW	0x3	CCIC3 CSI Lane 0 Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
7:6	CCIC3_GATE_CSI_CLK_DYNAMIC	RW	0x3	CCIC3 CSI Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
5:4	CCIC3_GATE_AHB_CLK_DYNAMIC	RW	0x3	CCIC3 ahb Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running

Offset: 0xD4282800+0x1C0				
Bits	**Field**	**Type**	**Reset**	**Description**
3:2	CCIC3_GATE_PIP_CLK_DYNAMIC	RW	0x3	CCIC3 axi Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running
1:0	CCIC3_GATE__AXI_CLK_DYNAMIC	RW	0x3	CCIC3 axi Clock Dynamic Clock Gate Control - 2'b00: Hardware dynamic control - 2'b01: Hardware dynamic control - 2'b10: Stop clock - 2'b11: Free running

9.2.4.2.33 CCI550 CLOCK CONTROL REGISTER (PMU_CCI_CLK_CTRL)

Offset: 0xD4282800+0x300				
Bits	**Field**	**Type**	**Reset**	**Description**
31:17	RSVD	RO	0x0	Reserved
16	MASK_DRAGON_ADB400_CCI_SIDE_IDLE	RW	0x0	Mask Dragon ADB400 CCI Side Idle Flag: It controls whether to mask the CCI side idle flag for cci_idle_clk_off_req - 1'b1: Mask the idle flag - 1'b0: Unmask the idle flag
15	CCI_CLK_SMOOTH_MUX_DIS	RW	0x0	CCI Clock Smooth Multiplexer Disable It controls whether a smooth transition for the CCI clock is enabled or disabled. - 1'b0: Enable CCI clock smooth MUX 1. Hardware automatically switches to the VCXO clock when the CPU enters M2 mode and the GPU shuts down. - 1'b1: Disable CCI clock smooth MUX 1. The clock for the CCI will only come from cci_clock_gen 2. The clock for the CCI is gated when the CPU enters M2 mode and the GPU shuts down
14	CCI_CLKEN_BY_INT_AP	RW	0x0	CCI Clock Enable by SYS_INT_AP It controls whether the CCI clock can be enabled by sys_int_ap[127:0] - 1'b0: The CCI clock is only controlled by the status of CPU Clusters and GPU - 1'b1: Enable this function
13	CCI550_CLKGEN_AUT	RW	0x0	CCI Clock Generator Automatic Gating Control

Offset: 0xD4282800+0x300				
Bits	**Field**	**Type**	**Reset**	**Description**
	O_CG_EN			It manages automatic gating for the CCI clock generator working clock - 1'b0: The automatic clock gating for the CCI550 clock generator is disabled, and the clock is free-running - 1'b1: The automatic clock gating for the CCI550 clock generator is enabled
12	CCI550_FC_REQ	RW	0x0	CCI frequency change request. - The field is cleared by hardware once the frequency change is done.
11	RSVD	RO	0x0	Reserved
10:8	CCI550_BIU_CLK_DIV	RW	0x1	Clock Divider Selection for CCI AXI_M0 port to fabric. ACLK_M0 = ACLKM1/ (this field +1)
7:2	RSVD	RO	0x0	Reserved
1:0	CCI550_PLLSEL	RW	0x0	CCI Clock Selection. - 2'b00: PLL1_491MHz. - 2'b01: PLL1 614MHz. - 2'b10: 819MHz. - 3'b11: PLL2_div3(1000MHz), as a backup for increasing voltage scenario

9.2.4.2.34 AP ACLK CONTROL REGISTER (PMUA_ACLK_CTRL)

Offset: 0xD4282800+0x388				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	ACLK_FC_REQ	RW	0x0	ACLK frequency change request. - 1'b1: Enable ACLK Frequency Change. - The field is cleared by hardware once the frequency change is done.
3	RSVD	RO	0x0	Reserved
2:1	ACLK_DIV	RW	0x0	ACLK_DIV ACLK=<ACLK_SEL>/(<ACLK_DIV>+1)
0	ACLK_SEL	RW	0x0	ACLK source selection - 1'b0: 249MHz - 1'b1: 312MHz

9.2.4.2.35 AP CPU CLUSTER0 CLK CONTROL REGISTER (PMUA_CPU_C0_CLK_CTRL)

Offset: 0xD4282800+0x38C

Bits	Field	Type	Reset	Description
31:14	RSVD	RO	0x0	Reserved
13	C0_HI_CLK_SEL	RW	0x0	CPU cluster0 highest Clock Frequecny Selection. It controls the selection of the highest clock frequency from CPU Cluster 0 based on the configuration of PLL3 - 1'b0: PLL3_div2(1600MHz) if PLL3 VCO is 3200M - 1'b1: PLL3_div1(1600MHz) if PLL3 VCO is 1600M
12	C0_CLK_FC_REQ	RW	0x0	CPU cluster0 clk frequency change request. - 1'b1: Enable clock frequency change. - The field is cleared by hardware once the frequency change is done.
11:9	C0_TCM_AXI_DIV	RW	0x1	Clock Divider Selection for Cluster0 TCM AXI slave Clock. Formula: C0_TCM_AXI = C0_CORE_CLK / (this field +1)
8:6	C0_ACE_CLK_DIV	RW	0x1	Clock Divider Selection for Cluster0 ACE Interface Clock Formula: C0_ACE_CLK = C0_CORE_CLK / (this field +1)
5:3	C0_CORE_CLK_DIV	RW	0x0	Clock Divider Selection for C0_CORE_CLK Formula: C0_CORE_CLK = Clock Selection / (this field +1)
2:0	C0_CLK_SEL	RW	0x0	CPU Cluster0 Clock Selection - 3'b000: 614MHz - 3'b001: 819MHz - 3'b010: 409MHz - 3'b011: 491MHz - 3'b100: 1228MHz - 3'b101: PLL3_div3(1066MHz) - 3'b110: PLL2_div3_gated(1000MHz) - 3'b111: depends on bit[13] 1. 1'b0: PLL3_div2(1600MHz) 2. 1'b1: PLL3_div1(1600MHz)

9.2.4.2.36 AP CPU CLUSTER1 CLK CONTROL REGISTER (PMUA_CPU_C1_CLK_CTRL)

Offset: 0xD4282800+0x390

Bits	Field	Type	Reset	Description
31:14	RSVD	RO	0x0	Reserved

Offset: 0xD4282800+0x390				
Bits	**Field**	**Type**	**Reset**	**Description**
13	C1_HI_CLK_SEL	RW	0x0	CPU cluster1 highest Clock frequecny selection. It controls the selection of the highest clock frequency from CPU Cluster 1 based on the configuration of PLL3 - 1'b0: PLL3_div2(1600MHz) if PLL3 VCO is 3200M - 1'b1: PLL3_div1(1600MHz) if PLL3 VCO is 1600M
12	C1_CLK_FC_REQ	RW	0x0	CPU cluster1 Clock frequency change request. - 1'b1: Enable clock frequency change. - The field is cleared by hardware once the frequency change is done.
11:9	C1_TCM_AXI_DIV	RW	0x1	Clock Divider Selection for Cluster0 TCM AXI slave Clock. Formula: C1_TCM_AXI = C1_CORE_CLK / (this field +1)
8:6	C1_ACE_CLK_DIV	RW	0x1	Clock Divider Selection for Cluster1 ACE Interface Clock Formula: C1_ACE_CLK = C1_CORE_CLK / (this field +1)
5:3	C1_CORE_CLK_DIV	RW	0x0	Clock Divider Selection for C1_CORE_CLK Formula: C1_CORE_CLK = Clock Selection / (this field +1)
2:0	C1_CLK_SEL	RW	0x0	CPU cluster1 clock selection - 3'b000: 614MHz - 3'b001: 819MHz - 3'b010: 409MHz - 3'b011: 491MHz - 3'b100: 1228MHz - 3'b101: PLL3_div3(1066MHz) - 3'b110: PLL2_div3_gated(1000MHz) - 3'b111: depends on bit[13] 1. 1'b0: PLL3_div2(1600MHz) 2. 1'b1: PLL3_div1(1600MHz)

9.2.4.2.37 PCIE PORTA CLK RESET CONTROL REGISTER (PCIE_CLK_RES_CTRL_PORTA)

Offset: 0xD4282800+0x3CC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PCIE_DEVICE_TYPE_SEL	RW	0x0	PCIe mode selection: - 1'b0: EP - 1'b1: RC
30	PCIE_APP_H	RW	0x1	Set this signal to 1 before the de-assertion of power-on reset

Offset: 0xD4282800+0x3CC				
Bits	**Field**	**Type**	**Reset**	**Description**
	OLD_PHY_RST			sequence to hold the PHY in reset. This can be used for PHY configuration.
29	PCIE_APP_SRIS_MODE	RW	0x0	Used to enable or disable SRIS mode for PCIe controller
28:24	PCIE_APP_DEV_NUM	RW	0x0	Device number for RC mode
23:16	PCIE_APP_BUS_NUM	RW	0x0	Bus number for RC mode
15	PCIE_APPS_PM_XMT_PME	RW	0x0	Wake Up. Used to wake the PCIe controller from low-power states (L1/L2) and restore active operation. - When the PME is enabled and configured in the PMCSR, asserting this signal wakes the controller from L1 or L2 states; once the controller transitions back to the L0 state, it sends a PME message and sets the PME_Status. The root complex then clears PME_Status and changes the D-state back to D0.
14	PCIE_APP_DBI_RO_WR_DISABLE	RW	0x0	DBI Read-only Write Disabled Controls the write access behavior of the DBI_RO_WR_EN register field. - 1'b0: MISC_CONTROL_1_OFF 1. DBI_RO_WR_EN register field is read-write. - 1'b1: MISC_CONTROL_1_OFF 1. DBI_RO_WR_EN register field is forced to 0 and is read-only.
13	PCIE_EP_WAKE_SW	RW	0x0	In EP mode, SE can program this bit to 1 to drive WAKE# to low. This is a wakeup event for RC side
12	PCIE_RC_PERST	RW	0x0	In RC mode, SW can program this bit to 1 to drive PERST# to low. This is a WARM reset for EP side
11	PCIE_PORTA_CLKREQ_OE	RW	0x0	If this bit is set to 1, the chip drives the CLKREQ# signal for PortA to 0
10	PCIE_PORTA_CLKREQ_IN	RO	0x1	Show the value of portA CLKREQ# IO input value
9	PCIE_SYS_AUX_PWR_DET	RW	0x1	Auxiliary Power Detected Used to report to the host software that auxiliary power (Vaux) is present
8	PCIE_GLB_RST	RW	0x1	Global reset. Software (SW) must clear this bit to 0 while simultaneously asserting the following reset signals: - pcie_axi_dbi_resetn - pcie_axi_slv_resetn - pcie_axi_mstr_resetn

Offset: 0xD4282800+0x3CC				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. This reset signal is high-level-valid
7	PCIE_PERSTN_IN	RO	0x1	PERST value form PAD for EP mode
6	PCIE_LTSSM_EN	RW	0x0	Enable the PCIe controller to start training - 1'b1: Enable - 1'b0: Hold the LTSSM in detect state.
5	PCIE_AXI_MSTR_RESETN	RW	0x0	PCIe AXI data master port reset-n. - 1'b1: Non-Reset - 1'b0: Reset
4	PCIE_AXI_SLV_RESETN	RW	0x0	PCIe AXI data slave port reset-n - 1'b1: Non-Reset - 1'b0: Reset
3	PCIE_AXI_DBI_RESETN	RW	0x0	PCIe AXI DBI slave port reset-n - 1'b1: Non-Reset - 1'b0: Reset
2	PCIE_AXI_MSTR_CLK_EN	RW	0x0	PCIe AXI data master port clock enable - 1'b1: Enable - 1’b0: Disable
1	PCIE_AXI_SLV_CLK_EN	RW	0x0	PCIe AXI data slave port clock enable - 1'b1: Enable - 1’b0: Disable
0	PCIE_AXI_DBI_CLK_EN	RW	0x0	PCIe AXI DBI slave port clock enable - 1'b1: Enable - 1’b0: Disable

9.2.4.2.38 PCIE PORTA CONTROL LOGIC REGISTER (PCIE_CTRL_LOGIC_PORTA)

Offset: 0xD4282800+0x3D0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	RO	0x0	Reserved
21:20	PCIE_RC_WAKEN_DEB_CFG	RW	0x3	Used to configure the debounce settings for the PCIe Root Complex (RC) WAKE_N signal
19:18	PCIE_PERSTN_IN_DEB_CFG	RW	0x3	Used to configure the debounce settings for the PCIe PERST# input signal
17:16	PCIE_RXELECIDLE_DEB_CFG	RW	0x3	Used to configure the debounce settings for the PCIe RX Electrical Idle signal
15	PCIE_WAKEUP	RW	0x0	Used to enable the PCIe wake-up interrupt

Offset: 0xD4282800+0x3D0				
Bits	**Field**	**Type**	**Reset**	**Description**
	_INT_EN			
14	PCIE_WAKEUP_EN	RW	0x0	Used to enable the wake-up functionality of the PCIe device itself
13:11	PCIE_WAKEUP_INT_REG	RO	0x0	Used to indicate the PCIe wakeup interrupt status: - bit 13: PCIe RC wakeup event - bit 12: PCIe EP perstn wakeup event - bit 11: PCIe RX Electrical Idle wakeup event
10:8	PCIE_WAKEUP_INT_CLR	RW	0x0	Used to clear the wake-up interrupt status bits for various PCIe wake-up events - bit 10: PCIe RC wakeup event - bit 9: PCIe EP perstn wakeup event - bit 8: PCIe RX Electrical Idle wakeup event
7	RSVD	RO	0x0	Reserved
6:4	PCIE_WAKEUP_MASK	RW	0x0	PCIe wakeup interrupt mask 1'b1: Enable - bit 6: PCIe RC wakeup event - bit 5: PCIe EP perstn wakeup event - bit 4: PCIe RX Electrical Idle wakeup event
3	PCIE_WAKE_SOURCE_SEL	RW	0x0	Wake# Source Selection in EP mode - 1'b1: the WAKE# pad is driven by pcie_ep_wake_sw bit of PCIe CLK Reset Control Register - 1'b0: the WAKE# pad is driven by PCIe controller
2	PCIE_IGNORE_PERSTN	RW	0x0	Used to control whether the PCIe controller and PHY in EP mode respond to the PERSTN signal from the RC. - When this bit is set to 1, The PCIe controller and PHY ignore the PERSTN signal from the RC
1	PCIE_FORCE_PERSTN	RW	0x0	In EP mode, SW can set this bit to 1 to force the PERST# signal to be asserted
0	PCIE_SOFT_RESET	RW	0x0	PCIe soft reset - 1'b1: Reset - 1'b0: Non-Reset

9.2.4.2.39 PCIE PORTB CLK RESET CONTROL REGISTER (PCIE_CLK_RES_CTRL_PORTB)

Offset: 0xD4282800+0x3D4				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PCIE_DEVICE	RW	0x0	PCIe mode selection:

Offset: 0xD4282800+0x3D4				
Bits	**Field**	**Type**	**Reset**	**Description**
	_TYPE_SEL			- 1'b0: EP - 1'b1: RC
30	PCIE_APP_HOLD_PHY_RST	RW	0x1	Set this signal to 1 before the de-assertion of power-on reset sequence to hold the PHY in reset. This can be used for PHY configuration.
29	PCIE_APP_SRIS_MODE	RW	0x0	Used to enable or disable SRIS mode for PCIe controller
28:24	PCIE_APP_DEV_NUM	RW	0x0	Device number for RC mode
23:16	PCIE_APP_BUS_NUM	RW	0x0	Bus number for RC mode
15	PCIE_APPS_PM_XMT_PME	RW	0x0	Wake Up. Used to wake the PCIe controller from low-power states (L1/L2) and restore active operation. - When the PME is enabled and configured in the PMCSR, asserting this signal wakes the controller from L1 or L2 states; once the controller transitions back to the L0 state, it sends a PME message and sets the PME_Status. The root complex then clears PME_Status and changes the D-state back to D0.
14	PCIE_APP_DBI_RO_WR_DISABLE	RW	0x0	DBI Read-only Write Disabled Controls the write access behavior of the DBI_RO_WR_EN register field. - 1'b0: MISC_CONTROL_1_OFF 1. DBI_RO_WR_EN register field is read-write. - 1'b1: MISC_CONTROL_1_OFF 1. DBI_RO_WR_EN register field is forced to 0 and is read-only.
13	PCIE_EP_WAKE_SW	RW	0x0	In EP mode, SE can program this bit to 1 to drive WAKE# to low. This is a wakeup event for RC side
12	PCIE_RC_PERST	RW	0x0	In RC mode, SW can program this bit to 1 to drive PERST# to low. This is a WARM reset for EP side
11	PCIE_PORTB_CLKREQ_OE	RW	0x0	If this bit is set to 1, the chip drives the CLKREQ# signal for PortB to 0
10	PCIE_PORTB_CLKREQ_IN	RO	0x1	Show the value of PortB CLKREQ# IO input value
9	PCIE_SYS_AUX_PWR_DET	RW	0x1	Auxiliary Power Detected Used to report to the host software that auxiliary power (Vaux) is present
8	PCIE_GLB_RST	RW	0x1	Global reset Software (SW) must clear this bit to 0 while simultaneously asserting the following reset signals:

Offset: 0xD4282800+0x3D4				
Bits	**Field**	**Type**	**Reset**	**Description**
				- pcie_axi_dbi_resetn - pcie_axi_slv_resetn - pcie_axi_mstr_resetn **Note**. This reset signal is high-level-valid
7	PCIE_PERSTN_IN	RO	0x1	PERST value form PAD for EP mode
6	PCIE_LTSSM_EN	RW	0x0	Enable the PCIe controller to start training - 1'b1: enable - 1'b0: hold the ltssm in detect.
5	PCIE_AXI_MSTR_RESETN	RW	0x0	PCIe AXI data master port reset-n. - 1'b1: Non-Reset - 1'b0: Reset
4	PCIE_AXI_SLV_RESETN	RW	0x0	PCIe AXI data slave port reset-n - 1'b1: Non-Reset - 1'b0: Reset
3	PCIE_AXI_DBI_RESETN	RW	0x0	PCIe AXI DBI slave port resetn - 1'b1: Non-Reset - 1'b0: Reset
2	PCIE_AXI_MSTR_CLK_EN	RW	0x0	PCIe AXI data master port clock enable - 1'b1: Enable - 1'b0: Disable
1	PCIE_AXI_SLV_CLK_EN	RW	0x0	PCIe AXI data slave port clock enable - 1'b1: Enable - 1'b0: Disable
0	PCIE_AXI_DBI_CLK_EN	RW	0x0	PCIe AXI dbi slave port clock enable - 1'b1: Enable - 1'b0: Disable

9.2.4.2.40 PCIE PORTB CONTROL LOGIC REGISTER (PCIE_CTRL_LOGIC_PORTB)

Offset: 0xD4282800+0x3D8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	RO	0x0	Reserved
21:20	PCIE_RC_WAKEN_DEB_CFG	RW	0x3	Used to configure the debounce settings for the PCIe Root Complex (RC) WAKE_N signal
19:18	PCIE_PERSTN_IN_DEB_C	RW	0x3	Used to configure the debounce settings for the PCIe PERST# input signal

Offset: 0xD4282800+0x3D8				
Bits	**Field**	**Type**	**Reset**	**Description**
	FG			
17:16	PCIE_RXELECIDLE_DEB_CFG	RW	0x3	Used to configure the debounce settings for the PCIe RX Electrical Idle signal
15	PCIE_WAKEUP_INT_EN	RW	0x0	Used to enable the PCIe wake-up interrupt
14	PCIE_WAKEUP_EN	RW	0x0	Used to enable the wake-up functionality of the PCIe device itself
13:11	PCIE_WAKEUP_INT_REG	RO	0x0	Used to indicate the PCIe wakeup interrupt status: - bit 13: PCIe RC wakeup event - bit 12: PCIe EP perstn wakeup event - bit 11: PCIe RX Electrical Idle wakeup event
10:8	PCIE_WAKEUP_INT_CLR	RW	0x0	Used to clear the wake-up interrupt status bits for various PCIe wake-up events - bit 10: PCIe RC wakeup event - bit 9: PCIe EP perstn wakeup event - bit 8: PCIe RX Electrical Idle wakeup event
7	RSVD	RO	0x0	Reserved
6:4	PCIE_WAKEUP_MASK	RW	0x0	PCIe wakeup interrupt mask 1'b1: Enable - bit 6: PCIe RC wakeup event - bit 5: PCIe EP perstn wakeup event - bit 4: PCIe RX Electrical Idle wakeup event
3	PCIE_WAKE_SOURCE_SEL	RW	0x0	Wake# Source Selection in EP mode - 1'b1: the WAKE# pad is driven by pcie_ep_wake_sw bit of PCIe CLK Reset Control Register - 1'b0: the WAKE# pad is driven by PCIe controller
2	PCIE_IGNORE_PERSTN	RW	0x0	Used to control whether the PCIe controller and PHY in EP mode respond to the PERSTN signal from the RC. - When this bit is set to 1, The PCIe controller and PHY ignore the PERSTN signal from the RC
1	PCIE_FORCE_PERSTN	RW	0x0	In EP mode, SW can set this bit to 1 to force the PERST# signal to be asserted
0	PCIE_SOFT_RESET	RW	0x0	PCIe soft reset - 1' b1: Reset - 1'b0: Non-Reset

9.2.4.2.41 PCIE PORTC CLK RESET CONTROL REGISTER (PCIE_CLK_RES_CTRL_PORTC)

Offset: 0xD4282800+0x3DC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PCIE_DEVICE_TYPE_SEL	RW	0x0	PCIe mode selection: - 1'b0: EP - 1'b1: RC
30	PCIE_APP_HOLD_PHY_RST	RW	0x1	Set this signal to 1 before the de-assertion of power-on reset sequence to hold the PHY in reset. This can be used for PHY configuration.
29	PCIE_APP_SRIS_MODE	RW	0x0	Used to enable or disable SRIS mode for PCIe controller
28:24	PCIE_APP_DEV_NUM	RW	0x0	Device number for RC mode
23:16	PCIE_APP_BUS_NUM	RW	0x0	Bus number for RC mode
15	PCIE_APPS_PM_XMT_PME	RW	0x0	Wake Up. Used to wake the PCIe controller from low-power states (L1/L2) and restore active operation. - When the PME is enabled and configured in the PMCSR, asserting this signal wakes the controller from L1 or L2 states; once the controller transitions back to the L0 state, it sends a PME message and sets the PME_Status. The root complex then clears PME_Status and changes the D-state back to D0.
14	PCIE_APP_DBI_RO_WR_DISABLE	RW	0x0	DBI Read-only Write Disabled Controls the write access behavior of the DBI_RO_WR_EN register field. - 1'b0: MISC_CONTROL_1_OFF 1. DBI_RO_WR_EN register field is read-write. - 1'b1: MISC_CONTROL_1_OFF 1. DBI_RO_WR_EN register field is forced to 0 and is read-only.
13	PCIE_EP_WAKE_SW	RW	0x0	In EP mode, SE can program this bit to 1 to drive WAKE# to low. This is a wakeup event for RC side
12	PCIE_RC_PERST	RW	0x0	In RC mode, SW can program this bit to 1 to drive PERST# to low. This is a WARM reset for EP side
11	PCIE_PORTC_CLKREQ_OE	RW	0x0	If this bit is set to 1, the chip drives the CLKREQ# signal for PortC to 0
10	PCIE_PORTC	RO	0x1	Show the value of PortC CLKREQ# IO input value

Offset: 0xD4282800+0x3DC				
Bits	**Field**	**Type**	**Reset**	**Description**
	_CLKREQ_IN			
9	PCIE_SYS_AUX_PWR_DET	RW	0x1	Auxiliary Power Detected Used to report to the host software that auxiliary power (Vaux) is present
8	PCIE_GLB_RST	RW	0x1	Global reset Software (SW) must clear this bit to 0 while simultaneously asserting the following reset signals: - pcie_axi_dbi_resetn - pcie_axi_slv_resetn - pcie_axi_mstr_resetn **Note.** This reset signal is high-level-valid
7	PCIE_PERSTN_IN	RO	0x1	PERST value form PAD for EP mode
6	PCIE_LTSSM_EN	RW	0x0	Enable the PCIe controller to start training - 1'b1: Enable - 1'b0: Hold the LTSSM in detect state.
5	PCIE_AXI_MSTR_RESETN	RW	0x0	PCIe AXI data master port reset-n. - 1'b1: Non-Reset - 1'b0: Reset
4	PCIE_AXI_SLV_RESETN	RW	0x0	PCIe AXI data slave port reset-n - 1'b1: Non-Reset - 1'b0: Reset
3	PCIE_AXI_DBI_RESETN	RW	0x0	PCIe AXI DBI slave port reset-n - 1'b1: Non-Reset - 1'b0: Reset
2	PCIE_AXI_MSTR_CLK_EN	RW	0x0	PCIe AXI data master port clock enable - 1'b1: Enable - 1’b0: Disable
1	PCIE_AXI_SLV_CLK_EN	RW	0x0	PCIe AXI data slave port clock enable - 1'b1: Enable - 1’b0: Disable
0	PCIE_AXI_DBI_CLK_EN	RW	0x0	PCIe AXI DBI slave port clock enable - 1'b1: Enable - 1’b0: Disable

9.2.4.2.42 PCIE PORTC CONTROL LOGIC REGISTER (PCIE_CTRL_LOGIC_PORTC)

Offset: 0xD4282800+0x3D8

Bits	Field	Type	Reset	Description
31:22	RSVD	RO	0x0	Reserved
21:20	PCIE_RC_WAKEN_DEB_CFG	RW	0x3	pcie_rc_waken debounce configuration Used to configure the debounce settings for the PCIe Root Complex (RC) WAKE_N signal
19:18	PCIE_PERSTN_IN_DEB_CFG	RW	0x3	pcie_perstn_in debounce configuration Used to configure the debounce settings for the PCIe PERST# input signal
17:16	PCIE_RXELECIDLE_DEB_CFG	RW	0x3	pcie_rxelecidle debounce configuration Used to configure the debounce settings for the PCIe RX Electrical Idle signal
15	PCIE_WAKEUP_INT_EN	RW	0x0	PCIE wake up enable Used to enable the PCIe wake-up interrupt
14	PCIE_WAKEUP_EN	RW	0x0	PCIE wake up event interrupt enable Used to enable the wake-up functionality of the PCIe device itself
13:11	PCIE_WAKEUP_INT_REG	RO	0x0	Used to indicate the PCIe wakeup interrupt status: - bit 13: PCIe RC wakeup event - bit 12: PCIe EP perstn wakeup event - bit 11: PCIe RX Electrical Idle wakeup event
10:8	PCIE_WAKEUP_INT_CLR	RW	0x0	Used to clear the wake-up interrupt status bits for various PCIe wake-up events - bit 10: PCIe RC wakeup event - bit 9: PCIe EP perstn wakeup event - bit 8: PCIe RX Electrical Idle wakeup event
7	RSVD	RO	0x0	Reserved
6:4	PCIE_WAKEUP_MASK	RW	0x0	PCIe wakeup interrupt mask 1'b1: Enable - bit 6: PCIe RC wakeup event - bit 5: PCIe EP perstn wakeup event - bit 4: PCIe RX Electrical Idle wakeup event
3	PCIE_WAKE_SOURCE_SEL	RW	0x0	Wake# Source Selection in EP mode - 1'b1: the WAKE# pad is driven by pcie_ep_wake_sw bit of PCIe CLK Reset Control Register - 1'b0: the WAKE# pad is driven by PCIe controller
2	PCIE_IGNORE_PERSTN	RW	0x0	Used to control whether the PCIe controller and PHY in EP mode respond to the PERSTN signal from the RC. - When this bit is set to 1, The PCIe controller and PHY ignore the PERSTN signal from the RC
1	PCIE_FORCE_PERSTN	RW	0x0	In EP mode, SW can set this bit to 1 to force the PERST# signal to be asserted
0	PCIE_SOFT_RE	RW	0x0	PCIe soft reset

Offset: 0xD4282800+0x3D8				
Bits	**Field**	**Type**	**Reset**	**Description**
	SET			- 1'b1: Reset - 1'b0: Non-Reset

9.2.4.2.43 EMAC0_CLK_RST_CTRL REGISTER (EMAC0_CLK_RST_CTRL)

Offset: 0xD4282800+0x3E4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	RO	0x0	Reserved
15	EMAC0_1588_CLK_EN	RW	0x0	- 1'b1: EMAC0 1588 clock enable - 1'b0: EMAC0 1588 clock disable
14	EMAC0_CLK_REF_SELECT	RW	0x0	- 1'b1: TX refclk Select 125MHz Clock - 1'b0: TX refclk Select 25MHz Clock
13	EMAC0_AXI_MST_ID	RW	0x0	- 1'b1: EMAC0 AXI MST interface uses single ID to issue transfer - 1'b0: EMAC0 AXI MST interface uses multiple IDs to issue transfer
12	EMAC0_PHY_INTR_EN	RW	0x0	- 1'b1: EMAC0 PHY interrupt enable - 1'b0: EMAC0 PHY interrupt disable
11:9	RSVD	RO	0x0	Reserved
8	EMAC0_RGMII_TXC_SRC_SEL	RW	0x0	This bit is only valid in RGMII mode. EMAC RGMII tx clock source selection. - 1'b1: TX clock source from SoC - 1'b0: TX clock source from RX clock
7:2	RSVD	RO	0x0	Reserved
1	EMAC0_BUS_RST	RW	0x0	EMAC0 AXI Bus Reset - 1'b1: Reset Release (deasserted) - 1'b0: Reset
0	EMAC0_BUS_EN	RW	0x0	EMAC0 AXI Bus Clock Enable - 1'b1: AXI clock enabled - 1'b0: AXI clock disabled

9.2.4.2.44 EMAC0_RGMII_DLINE REGISTER (EMAC0_RGMII_DLINE)

Offset: 0xD4282800+0x3E8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	EMAC0_RGMII_TXC_DLINE_ADJ	RW	0x0	Delay code
23:17	RSVD	RO	0x0	Reserved
16	EMAC0_RGMII_TXC_DLINE_PU	RW	0x0	Delay line enable
15:8	EMAC0_RGMII_RXC_DLINE_ADJ	RW	0x0	Delay code
7:1	RSVD	RO	0x0	Reserved
0	EMAC0_RGMII_RXC_DLINE_PU	RW	0x0	Delay line enable

9.2.4.2.45 EMAC1_CLK_RST_CTRL REGISTER (EMAC1_CLK_RST_CTRL)

Offset: 0xD4282800+0x3EC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	RO	0x0	Reserved
15	EMAC1_1588_CLK_EN	RW	0x0	- 1'b1: EMAC1 1588 clock enable - 1'b0: EMAC1 1588 clock disable
14	EMAC1_CLK_REF_SELECT	RW	0x0	- 1'b1: TX refclk Select 125MHz Clock - 1'b0: TX refclk Select 25MHz Clock
13	EMAC1_AXI_MST_ID	RW	0x0	- 1'b1: EMAC1 AXI MST interface uses single ID to issue transfer - 1'b0: EMAC1 AXI MST interface uses multiple IDs to issue transfer
12	EMAC1_PHY_INTR_EN	RW	0x0	- 1'b1: EMAC1 PHY interrupt enable - 1'b0: EMAC1 PHY interrupt disable
11:9	RSVD	RO	0x0	Reserved
8	EMAC1_RGMII_TXC_SRC_SEL	RW	0x0	This bit is only valid in RGMII mode. EMAC RGMII tx clock source selection. - 1'b1: TX clock source from SoC - 1'b0: TX clock source from RX clock
7:2	RSVD	RO	0x0	Reserved
1	EMAC1_BUS_RST	RW	0x0	EMAC1 AXI Bus Reset - 1'b1: Reset Release (deasserted) - 1'b0: Reset
0	EMAC1_BUS_EN	RW	0x0	EMAC1 AXI Bus Clock Enable - 1'b1: AXI clock enabled - 1'b0: AXI clock disabled

9.2.4.2.46 EMAC1_RGMII_DLINE REGISTER (EMAC1_RGMII_DLINE)

Offset: 0xD4282800+0x3E8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	EMAC1_RGMII_TXC_DLINE_ADJ	RW	0x0	Delay code
23:17	RSVD	RO	0x0	Reserved
16	EMAC1_RGMII_TXC_DLINE_PU	RW	0x0	Delay line enable
15:8	EMAC1_RGMII_RXC_DLINE_ADJ	RW	0x0	Delay code
7:1	RSVD	RO	0x0	Reserved
0	EMAC1_RGMII_RXC_DLINE_PU	RW	0x0	Delay line enable

9.2.4.3 Basing on <APBCLOCK(0xD4015000)>

9.2.4.3.1 UARTX CLOCK RESET CONTROL REGISTER (APBC_UARTX_CLK_RST)

X=0/1/2/3/4/5/6/7/8/9

Offset: 0xD4015000+0x0/0x4/0x24/0x70/0x74/0x78/0x94/0x98/0x9C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 57.6 MHz - 3'b001: 14.7456 MHz - 3'b010: 48 MHz - 3'b011: UART_LP - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	UART Reset Generation. This field resets both APB and Functional domains. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	UART Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	UART APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.2 GPIO CLOCK RESET CONTROL REGISTER (APBC_GPIO_CLK_RST)

Offset: 0xD4015000+0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	UART Reset Generation. This field resets both APB and Functional domains. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	GPIO Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	GPIO APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.3 PWMX CLOCK RESET CONTROL REGISTER (APBC_PWMX_CLK_RST)

X=0/1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19

Offset: 0xD4015000+0xC/0x10/0x14/0x18/0xA8/0xAC/0xB0/0xB4/0xB8/0xBC/0xC0/0xC4/0xC8/0xCC/0xD0/0xD4/0xD8/0xDC/0xE0/0xE4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 12.8 MHz - 3'b001: 32 KHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	PWMX Reset Generation. This field resets both APB and Functional domains. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	PWMX Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	PWMX APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.4 SSPX CLOCK RESET CONTROL REGISTER (APBC_SSPX_CLK_RST)

X=3

Offset: 0xD4015000+0x7C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	RO	0x0	Reserved
7	SEL_SSP_FUNC_CLK	RW	0x0	AC97 Clock Switch This bit enables the SSP module to switch clocks internally.
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 6.4 MHz - 3'b001: 12.8 MHz - 3'b010: 25.6 MHz - 3'b011: 51.2 MHz - 3'b100: 3.2 MHz - 3'b101: 1.6 MHz - 3'b110: 800 kHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	SSP 3 Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	SSP 3 Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	SSP 3 APB Bus Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.5 RTC CLOCK RESET CONTROL REGISTER (APBC_RTC_CLK_RST)

Offset: 0xD4015000+0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	RO	0x0	Reserved
7	PM_POWER_SENSOR	RW	0x0	Power Enabled. This field enables the register read/write for the RTC module by indicating power is enabled. Set this field to 0x1 before enabling RTC operations.
6:4	FNCLKSEL	RW	0x0	Functional Clock Select.

Offset: 0xD4015000+0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 3'b000: 32 KHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	RTC Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	RTC Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	RTC APB Bus Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.6 TWSI0 CLOCK RESET CONTROL REGISTER (APBC_TWSI0_CLK_RST)

Offset: 0xD4015000+0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI0 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI0 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI0 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.7 TWSI1 CLOCK RESET CONTROL REGISTER (APBC_TWSI1_CLK_RST)

Offset: 0xD4015000+0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI1 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI1 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI1 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.8 TWSI2 CLOCK RESET CONTROL REGISTER (APBC_TWSI2_CLK_RST)

Offset: 0xD4015000+0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI2 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI2 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on

Offset: 0xD4015000+0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
0	APBCLK	RW	0x0	TWSI2 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.9 TWSI4 CLOCK RESET CONTROL REGISTER (APBC_TWSI4_CLK_RST)

Offset: 0xD4015000+0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI4 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI4 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI4 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.10 TWSI5 CLOCK RESET CONTROL REGISTER (APBC_TWSI5_CLK_RST)

Offset: 0xD4015000+0x4C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use

Offset: 0xD4015000+0x4C				
Bits	**Field**	**Type**	**Reset**	**Description**
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI5 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI5 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI5 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.11 TWSI6 CLOCK RESET CONTROL REGISTER (APBC_TWSI6_CLK_RST)

Offset: 0xD4015000+0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI6 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI6 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI6 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.12 TWSI7 CLOCK RESET CONTROL REGISTER (APBC_TWSI7_CLK_RST)

Offset: 0xD4015000+0x68

Bits	Field	Type	Reset	Description
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI7 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI7 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI7 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.13 TWSI8 CLOCK RESET CONTROL REGISTER (APBC_TWSI8_CLK_RST)

Offset: 0xD4015000+0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	WO	0	Reserved for future use
6:4	FNCLKSEL	WO	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	WO	0	Reserved for future use
2	RST	WO	0x1	TWSI8 Reset Generation This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	WO	0x0	TWSI8 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	WO	0x0	TWSI8 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.3.14 TIMERX CLOCK RESET CONTROL REGISTER (APBC_TIMERX_CLK_RST)

X=1/2

Offset: 0xD4015000+0x34/0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 12.8 MHz - 3'b001: 32 KHz - 3'b010: 6.4 MHz - 3'b011: 3.00 MHz - 3'b100: 1 MHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	TIMER Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	TIMER Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	TIMER APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.15 AIB CLOCK RESET CONTROL REGISTER (APBC_AIB_CLK_RST)

Offset: 0xD4015000+0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Reserved for future use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	AIB Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	AIB Functional Clock Enable/ disable - 1'b0: Clock off - 1'b1: Clock on
0	APBCLK	RW	0x0	AIB APB Bus Clock Enable/Disable

Offset: 0xD4015000+0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b0: Clock off - 1'b1: Clock on

9.2.4.3.16 SSPAX CLOCK RESET CONTROL REGISTER (APBC_SSPAX_CLK_RST)

X=1/2

Offset: 0xD4015000+0x80/0x84				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0 = 6.5 MHz - 0x1 = 13MHz - 0x2 = 26 MHz - 0x3 = 52 MHz - 0x4 = 3.25 MHz - 0x5=1.625 MHz - 0x6=812.5 kHz - 0x7=1 MHz clock or i2s bit clock (MN divided from PLL DIV8) - All other values: Reserved, do not use
3	SEL_1MHz	RW	0	- 0x0 = 1 MHz - 0x1 = i2s bit clock MN divided from PLL_div8
2	RST	RW	0x1	SSPA Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	SSPA Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	SSPA APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.17 ONEWIRE CLOCK RESET CONTROL REGISTER (APBC_ONEWIRE_CLK_RST)

Offset: 0xD4015000+0x49				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0	Reserved

Offset: 0xD4015000+0x49				
Bits	**Field**	**Type**	**Reset**	**Description**
6:4	FNCLKSEL	RW	0x0	IR Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
3	RSVD	RO	0	Reserved
2	RST	RW	0x1	IR Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	One-Wire Functional Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	One-Wire APB Bus Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.18 DRO CLOCK RESET CONTROL REGISTER (APBC_DRO_CLK_RST)

Offset: 0xD4015000+0x58				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	DRO Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	RSVD	RO	0x0	Reserved
0	APBCLK	RW	0x0	DRO APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.19 IR CLOCK RESET CONTROL REGISTER (APBC_IR_CLK_RST)

Offset: 0xD4015000+0x5C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0x0	Reserved

Offset: 0xD4015000+0x5C				
Bits	**Field**	**Type**	**Reset**	**Description**
2	RST	RW	0x1	IR Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	RSVD	RO	0x0	Reserved
0	APBCLK	RW	0x0	IR APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.20 COUNTER CLOCK RESET CONTROL REGISTER (APBC_COUNTER_CLK_RST)

Offset: 0xD4015000+0x64				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	LOW_FREQ_STEP	RW	0x2DC	Generic Counter Step of Low Frequency This value represents the Generic Counter step when operating at low frequency. Default: 24 MHz / 32768 = 0x2DC
15:2	RSVD	RO	0x0	Reserved
1	FREQ_SW_SEL	RW	0x0	Generic Counter Frequency Software Select. - 1'b0: 24 MHz - 1'b1: 32 KHz
0	FREQ_HW_CTRL	RW	0x0	Generic Counter Frequency Controlled by Hardware. - 1'b0: Software FREQ_SW_SEL - 1'b1: Hardware VCTCXO_EN signal. 1. If VCTCXO_EN=1, Generic Counter clock frequency is 24 MHz. 2. If VCTCXO_EN=0, Generic Counter clock frequency is 32 kHz

9.2.4.3.21 TEMPERATURE SENSOR CLOCK RESET CONTROL REGISTER (APBC_TSEN_CLK_RST)

Offset: 0xD4015000+0x6C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0x0	Reserved
2	TSEN_RST_EN	RW	0x1	Temperature Sensor Reset Enable - 1'b0: Release reset

Offset: 0xD4015000+0x6C				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b1: Reset Temperature Sensor
1	TSEN_FCLK_EN	RW	0x0	Temperature Sensor Function Clock Enable - 1'b0: Disable temperature sensor function clock - 1'b1: Enable temperature sensor function clock
0	TSEN_PCLK_EN	RW	0x0	Temperature Sensor APB Clock Enable - 1'b0: Disable temperature sensor APB clock - 1'b1: Enable temperature sensor APB clock

9.2.4.3.22 INTER-PROCESSOR COMMUNICATION AP TO AUDIO CLOCK RESET CONTROL REGISTER (APBC_IPC_AP2AUD_CLK_RST)

Offset: 0xD4015000+0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	IPC Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	IPC Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	IPC APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.3.23 CAN CLOCK RESET CONTROL REGISTER (APBC_CAN_CLK_RST)

Offset: 0xD4015000+0xA0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 20 MHz - 3'b001: 40 MHz - 3'b010: 80 MHz - All other values: Reserved, do not use
2	RST	RW	0x1	CAN Reset Generation

Offset: 0xD4015000+0xA0				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	CAN Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	CAN APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.4 Basing on <APB_SPARE(0xD4090000)>

9.2.4.4.1 PLLX SW1 CONTROL REGISTER (PLLX_SW1_CTRL)

X=1/2/3

Offset: 0xD4090000+0x100/0x118/0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	PLL_REG8	RW	0x00	PLL test control. - [31]: Not in use - [30]: Input frequency selection between 24 and 19.2/38.4 MHz. 1. 1'b0: Input frequency = 24 MHz 2. 1'b1: Input frequency = 19.2 or 38.4 MHz - [29]: FVCO configuration sources. 1. 1'b0: External value from digital value 2. 1'b1: Internal value from default value in force - [28]: Selection on input frequency between 19.2 and 38.4 MHz. 1. 1'b0: Input frequency = 38.4 MHz 2. 1'b1: Input frequency = 19.2 MHz - [27:24]: CK_test driving capability and clock configuration 1. <27:26> CK_test driving capability - 2'b00: 2 driver cell - 2'b01: 3 driver cell - 2'b10: 4 driver cell - 2'b11: 5 driver cell 2. <25:24> CK input select. - 2'b00: ckin_1 (div200_aud) - 2'b01: ckin_2(div3_soc) - 2'b10: ckin_3(div5_soc) - 2'b11: ckin_4(clk_dac)
23:16	PLL_REG7	RW	0x50	- [23]: Bypass PLL power down 1. 1'b1: PD is bypassed and always on

Offset: 0xD4090000+0x100/0x118/0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
				2. 1'b0: Controlled by PD - [22]: SSC enable select 1. 1'b1: Pre_lock 2. 1'b0: LDO_rdy - [21]: choose vreg caliration period 1. 1'b1: 256*Tref 2. 1'b0: 128*Tref - [20]: Enable PLL fast lock - [19]: Force PLL lock - [18:16]: ATEST/DTEST select
15:8	PLL_REG6	RW	0xDD	- [15:14]: select LPF proportionality factor 1. 2'b00: fref = 38.4 MHz 2. 2'b01/2'b10: fref = 30/27/26/25/24 MHz 3. 2'b11: fref = 19.2 MHz - [13:12]: PLL pre-divider select 1. 2'b00: pre_div = div1 (vco range: 1G~2G) 2. 2'b01: pre_div = div2 (vco range: 2G~4G) 3. 2'b10: pre_div = div3 (default) (vco range: 4G~6G) 4. 2'b11: pre_div = div4 - [11]: High kvco enable 1. 1'b1: Enable 2. 1'b0: Disable - [10]: Enable regulator calibration 1. 1'b1: Enable 2. 1'b0: Disable - [9:8]: regulator calibration vref select 1. 2'b00: vrefh = 727mV, vrefl = 626mV 2. 2'b01: vrefh = 750mV, vrefl = 650mV 3. 2'b10: vrefh = 776mV, vrefl = 679mV 4. 2'b11: vrefh = 802mV, vrefl = 707mV
7:0	PLL_REG5	RW	0x64	- [7:5]: Charge-bump current select 1. 3'b000: 0.5 μA 2. 3'b100: 0.75 μA (24/25/26/27/30 MHz) 3. 3'b010/3'b001: 1.0 μA 4. 3'b101: 1.25 μA (19.2 MHz) 5. 3'b011/3'b010: 1.5 μA 6. 3'b110: 1.75 μA (38.4 MHz) 7. 3'b001/3'b011: 2.0 μA 8. 3'b111: 2.25 μA - [4]: config DAC clock 1. 1'b0: DAC's T_dac = 10xTvco 2. 1'b1: DAC's T_dac = 12xTvco - [3]: config ADC clock 1. 1'b0: ADC's T_adc = 10xTvco 2. 1'b1: ADC's T_adc = 12xTvco

Offset: 0xD4090000+0x100/0x118/0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
				- [2:0]: vco frequency-band (GHz) select when reg6<3>(bit[11]) = 0: 1. 3'b000: 0.61~0.76~0.92 2. 3'b001: 0.85~1.07~1.24 3. 3'b010: 1.24~1.37~1.67 4. 3'b011: 1.34~1.67~2.01 5. 3'b100: 1.68~1.97~2.27 6. 3'b101: 1.94~2.28~2.62 7. 3'b110: 2.19~2.58~2.96 8. 3'b111: 2.45~2.88~3.32 - [2:0]: vco frequency-band (GHz) select when reg6<3>(bit[11]) = 1: 1. 3'b000: 0.61~0.76~0.92 2. 3'b001: 0.85~1.07~1.24 3. 3'b010: 1.24~1.37~1.67 4. 3'b011: 1.34~1.67~2.01 5. 3'b100: 1.68~1.97~2.27 6. 3'b101: 1.94~2.28~2.62 7. 3'b110: 2.19~2.58~2.96 8. 3'b111: 2.45~2.88~3.32

9.2.4.4.2 PLLX SW2 CONTROL REGISTER (PLLX_SW2_CTRL)

X=1/2/3

Offset: 0xD4090000+0x104/0x11C/0x128				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	BG_REG	RW	0x58	- internal pin name: {nc,nc,nc, bgsel<2:0>, rtemp<1:0>} bg_reg1<7:5>: Reserved - bg_reg1<4:2>: bandgap output control bits - bg_reg1<1:0>: bandgap output temperature coefficient control bits
23	BG_EN	RW	0x1	Bandgap enable - 1'b1: Enable
22	REFBUF2_EN	RW	0x0	REFBUF SW enable - 1'b1: Enable. - 1'b0: HW control
21	PLL_UPDATE_EN	RW	0x1	PLLX divider update enable - 1'b1: Enable - 1'b0: Disable
20	PLL_DIV23_EN	RW	0x0	PLLX_DIV23_EN - 1'b1: Enable
19:17	PLL1_MON_CFG	RW	0x4	PLLX_MON_CFG.

Offset: 0xD4090000+0x104/0x11C/0x128				
Bits	**Field**	**Type**	**Reset**	**Description**
				- [19]: Monitor enable - [18:17]: Monitor divider
16	PLL_DIV13_EN	RW	0x0	PLLX_DIV13_EN - 1'b1: Enable
15	PLL_DIV11_EN	RW	0x0	PLLX_DIV11_EN - 1'b1: Enable
14	EN_DTEST	RW	0x0	DTEST enable
13	EN_CKTEST	RW	0x0	CKTEST enable
12	EN_ATEST	RW	0x0	ATEST enable
11	PLL_24P576_AUD_EN	RW	0x1	PLL1_24p576_AUD_EN - 1'b1: Enable
10	PLL_245P76_AUD_EN	RW	0x1	PLL1_245p76_AUD_EN - 1'b1: Enable
9	PLL_245P6_DAC_EN	RW	0x1	If APBaux/PLL_ADDA_OVRD_EN=1, this bit controls PLLX_245p6_DAC_EN - 1'b1: Enable
8	PLL_245P6_ADC_EN	RW	0x1	If APBaux/PLL_ADDA_OVRD_EN=1, this bit controls PLLX_245p6_ADC_EN - 1'b1: Enable
7	PLL_DIV8_EN	RW	0x1	PLLX_DIV8_EN - 1'b1: Enable
6	PLL_DIV7_EN	RW	0x1	PLLX_DIV7_EN - 1'b1: Enable
5	PLL_DIV6_EN	RW	0x1	PLLX_DIV6_EN - 1'b1: Enable
4	PLL_DIV5_EN	RW	0x1	PLLX_DIV5_EN - 1'b1: Enable
3	PLL_DIV4_EN	RW	0x1	PLLX_DIV4_EN - 1'b1: Enable
2	PLL_DIV3_EN	RW	0x1	PLLX_DIV3_EN - 1'b1: Enable
1	PLL_DIV2_EN	RW	0x1	PLLX_DIV2_EN - 1'b1: Enable
0	PLL_DIV1_EN	RW	0x1	PLLX_DIV1_EN - 1'b1: Enable

9.2.4.4.3 PLLX SW3 CONTROL REGISTER (PLLX_SW3_CTRL)

X=1/2/3

Offset: 0xD4090000+0x108/0x120/0x12C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PLL_SW_EN	RW	0x0	- 1'b0: PLL enable controlled by PMU HW - 1'b1: SW force enabled
30:0	RSVD	RO	0x0	Reserved

9.2.4.5 Basing on <APB_SPARE(0xF0610000)>

9.2.4.5.1 APB2 UART1 CLOCK RESET CONTROL REGISTER (APB2_UART1_CLK_RST)

Offset: 0xF0610000+0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 57.6 MHz - 3'b001: 14.7456 MHz - 3'b010: 48MHz - 3'b011: UART_LP - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	UART Reset Generation. This field resets both APB and Functional domains. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	UART Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	UART APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.5.2 APB2 SSP2 CLOCK RESET CONTROL REGISTER (APB2_SSP2_CLK_RST)

Offset: 0xF0610000+0x4

Bits	Field	Type	Reset	Description
31:8	RSVD	RO	0x0	Reserved
7	SEL_SSP_FUNC_CLK	RW	0x0	AC97 Clock Switch This bit enables the SSP module to switch clocks internally.
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 6.4 MHz - 3'b001: 12.8 MHz - 3'b010: 25.6 MHz - 3'b011: 51.2 MHz - 3'b100: 3.2 MHz - 3'b101: 1.6 MHz - 3'b110: 800 kHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	SSP 3 Reset Generation This field resets both the APB and functional domain - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	SSP 3 Functional Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	SSP 3 APB Bus Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.5.3 TWSI3 CLOCK RESET CONTROL REGISTER (APBC_TWSI3_CLK_RST)

Offset: 0xF0610000+0x8

Bits	Field	Type	Reset	Description
31:7	RSVD	RO	0	Reserved for future use
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 0x0: 31.5 MHz - 0x1: 51.2 MHz - 0x2: 61.44 MHz - All other values: Reserved, do not use
3	RSVD	RO	0	Reserved for future use
2	RST	RW	0x1	TWSI3 Reset Generation

Offset: 0xF0610000+0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field resets both the APB and functional domain. - 0: No Reset - 1: Reset
1	FNCLK	RW	0x0	TWSI3 Functional Clock Enable/Disable. - 0: Clock off - 1: Clock on
0	APBCLK	RW	0x0	TWSI3 APB Bus Clock Enable/Disable. - 0: Clock off - 1: Clock on

9.2.4.5.4 APB2 SECURE RTC CLOCK RESET CONTROL REGISTER (APB2_SEC_RTC_CLK_RST)

Offset: 0xF0610000+0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	RO	0x0	Reserved
7	PM_POWER_SENSOR	RW	0x0	Power Enabled. This field enables the register read/write for the RTC module by indicating power is enabled. Set this field to 0x1 before enabling RTC operations.
6:4	FNCLKSEL	RW	0x0	Functional Clock Select. - 3'b000: 32 KHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	RTC Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	RTC Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	RTC APB Bus Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.5.5 APB2 TIMER0 CLOCK RESET CONTROL REGISTER (APB2_TIMER0_CLK_RST)

Offset: 0xF0610000+0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select - 3'b000: 12.8 MHz - 3'b001: 32 KHz - 3'b010: 6.4 MHz - 3'b011: 3.00 MHz - 3'b100: 1 MHz - All other values: Reserved, do not use
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	TIMER Reset Generation This field resets both the APB and functional domain. - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	TIMER Functional Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	TIMER APB Bus Clock Enable/Disable. - 1'b0: Clock gating - 1'b1: Clock on

9.2.4.5.6 APB2 KPC CLOCK RESET CONTROL REGISTER (APB2_KPC_CLK_RST)

Offset: 0xF0610000+0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	RO	0x0	Reserved
6:4	FNCLKSEL	RW	0x0	Functional Clock Select
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	KPC Reset Generation This field resets both APB and Functional domains - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	KPC Functional Clock Enable/Disable - 1'b0: Clock gating - 1'b1: Clock on
0	APBCLK	RW	0x0	KPC APB Bus Clock Enable/Disable.

Offset: 0xF0610000+0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 1'b0: Clock gating - 1'b1: Clock on

9.2.4.6 Basing on <RCPU(0xC0880000)>

9.2.4.6.1 RCPU SSP0 CLOCK RESET CONTROL REGISTER (RCPU_SSP0_CLK_RST)

Offset: 0xC0880000+0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	RO	0x0	Reserved
18:8	SHUBSSP0_FCLK_DIV	RW	0x0	RCPU SSP0 clk div fclk= source_clk/(rcpussp0_fclk_div + 1)
7:6	RSVD	RO	0x0	Reserved
5:4	SHUBSSP0_FCLK_SEL	RW	0x0	RCPU SSP0 fclk selection: - 2'b00: clk_61.44MHz - 2'b01: clk_25.6MHz - 2'b10: clk_12.8MHz - 2'b11: clk_3MHz
3	RSVD	RO	0x0	Reserved
2	SHUBSSP0_PCLK_EN	RW	0x0	Enable bit for RCPU SSP0 pclk. - 1'b0: Disable - 1'b1: Enable
1	SHUBSSP0_FCLK_EN	RW	0x0	Enable bit for RCPU SSP0 fclk - 1'b0: Disable - 1'b1: Enable
0	SHUBSSP0_SW_RSTN	RW	0x0	Enable bit for RCPU SSP0 reset - 1'b0: Disable - 1'b1: Enable

9.2.4.6.2 RCPU I2C0 CLOCK RESET CONTROL REGISTER (RCPU_I2C0_CLK_RST)

Offset: 0xC0880000+0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	RO	0x0	Reserved
18:8	SHUBI2C0_FCLK_DIV	RW	0x0	RCPU I2C0 clk div

Offset: 0xC0880000+0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
				fclk= source_clk/(rcpui2c0_fclk_div + 1)
7:6	RSVD	RO	0x0	Reserved
5:4	SHUBI2C0_FCLK_SEL	RW	0x0	RCPU I2C0 fclk selection - 2'b00: clk_61.44MHz - 2'b01: clk_25.6MHz - 2'b10: clk_12.8MHz - 2'b11: clk_3MHz
3	RSVD	RO	0x0	Reserved
2	SHUBI2C0_PCLK_EN	RW	0x0	Enable bit for RCPU I2C0 pclk. - 1'b0: Disable - 1'b1: Enable
1	SHUBI2C0_FCLK_EN	RW	0x0	Enable bit for RCPU I2C0 fclk - 1'b0: Disable - 1'b1: Enable
0	SHUBI2C0_SW_RSTN	RW	0x0	Enable bit for RCPU I2C0 reset - 1'b0: Disable - 1'b1: Enable

9.2.4.6.3 RCPU UARTX CLOCK RESET CONTROL REGISTER (RCPU_UARTX_CLK_RST)

X=1/0

Offset: 0xC0880000+0x3C/0xD8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	RO	0x0	Reserved
18:8	UART_FCLK_DIV	RW	0x0	uart clk div fclk= source_clk/(uart_fclk_div + 1)
7:6	RSVD	RO	0x0	Reserved
5:4	UART_FCLK_SEL	RW	0x0	UART fclk selection - 2'b00: clk_61.44MHz - 2'b01: clk_25.6MHz - 2'b10: clk_12.8MHz - 2'b11: clk_3MHz
3	RSVD	RO	0x0	Reserved
2	UART_PCLK_EN	RW	0x0	Enable bit for UART pclk - 1'b0: Disable - 1'b1: Enable

Offset: 0xC0880000+0x3C/0xD8				
Bits	**Field**	**Type**	**Reset**	**Description**
1	UART_FCLK_EN	RW	0x0	Enable bit for UART fclk - 1'b0: Disable - 1'b1: Enable
0	UART_SW_RSTN	RW	0x0	Enable bit for UART reset - 1'b0: Disable - 1'b1: Enable

9.2.4.6.4 RCPU CAN CLOCK RESET CONTROL REGISTER (RCPU_CAN_CLK_RST)

Offset: 0xC0880000+0x48				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	RO	0x0	Reserved
18:8	CAN_FCLK_DIV	RW	0x0	CAN clk div fclk= source_clk/(can_fclk_div + 1)
7:6	RSVD	RO	0x0	Reserved
5:4	CAN_FCLK_SEL	RW	0x0	CAN fclk selection: - 2'b00: 20 MHz - 2'b01: 40 MHz - 2'b10: 80 MHz - 2'b11: Reserved
3	RSVD	RO	0x0	Reserved
2	CAN_PCLK_EN	RW	0x0	Enable bit for CAN pclk - 1'b0: Disable - 1'b1: Enable
1	CAN_FCLK_EN	RW	0x0	Enable bit for CAN clk - 1'b0: Disable - 1'b1: Enable
0	CAN_SW_RSTN	RW	0x0	Enable bit for CAN reset - 1'b0: Disable - 1'b1: Enable

9.2.4.6.5 RCPU IR CLOCK RESET CONTROL REGISTER (RCPU_IR_CLK_RST)

Offset: 0xC0880000+0x4C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	RO	0x0	Reserved

Offset: 0xC0880000+0x4C				
Bits	**Field**	**Type**	**Reset**	**Description**
2	R_IR_PCLK_EN	RW	0x0	Enable bit for R_IR pclk - 1'b0: Disable - 1'b1: Enable
1	RSVD	RO	0	Reserved
0	R_IR_SW_RSTN	RW	0x0	Enable bit for R_IR reset - 1'b1: Enable

9.2.4.7 Basing on <AUD_AUDCLOCK (0xC0882000)>

9.2.4.7.1 AUDIO TX RX CLOCK CONTROL REGISTER (AUDIO_CODEC_TX_RX_CLK_CTRL)

This register controls the audio SSPA and ADMA clock. Both share the same bus clock, while SSPA has a separate functional clock.

Offset: 0xC0882000+0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	RSVD	RO	0x0	Reserved
17:16	SSPA_FCLK_SRC_SEL	RW	0x0	SSPA function clock source - 2'b0: SSPA_FCLK_SRC = 24.576M - 2'b1: SSPA_FCLK_SRC = 245.76M
15	RSVD	RO	0x0	Reserved
14:4	SSPA_FCLK_DIV	RW	0x9f	SSPA function clock divider sspa_fclk = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	RSVD	RO	0x0	Reserved
2	SSPA_FCLK_EN	RW	0x0	- 1'b0: SSPA function clock gated - 1'b1: SSPA function clock enable
1	TXRX_BUS_CLK_EN	RW	0x0	- 1'b0: bus clock gated - 1'b1: bus clock enable
0	TXRX_SW_RSTN	RW	0x0	Reset control for audio SSPA and ADMA - 1'b0: Software reset - 1'b1: Release reset

9.2.4.7.2 AUDIO DEF CLOCK CONTROL REGISTER (AUDIO_DFE_CLK_CTRL)

This register is for the Audio Codec DFE clock control.

Offset: 0xC0882000+0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	RSVD	RO	0x0	Reserved
5	DFE_SW_RST	RW	0x0	- 1'b0: Software reset - 1'b1: Release reset
4	DFE_FUNC_CLK_EN	RW	0x0	- 1'b0: Disable - 1'b1: Enable
3	DAC_SW_RST	RW	0x0	- 1'b0: Software reset - 1'b1: Release reset
2	DAC_CLK_INV_EN	RW	0x0	DAC clock can use the original clock from analog or the invert clock - 1'b0: DAC uses the original clock - 1'b1: DAC uses the invert clock
1	ADC_SW_RST	RW	0x0	- 1'b0: Software reset - 1'b1: Release reset
0	ADC_CLK_INV_EN	RW	0x0	ADC clock can use the original clock from analog or the invert clock - 1'b0: ADC uses the original clock - 1'b1: ADC uses the invert clock

9.2.4.7.3 AUDIO FM TX RX CLOCK CONTROL REGISTER (AUDIO_I2S1_TX_RX_CLK_CTRL)

This register controls the audio SSPA and ADMA clock. Both share the same bus clock, while SSPA has a separate functional clock.

Offset: 0xC0882000+0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	RSVD	RO	0x0	Reserved
17:16	I2S1_SSPA_FCLK_SRC_SEL	RW	0x0	SSPA function clock source - 2'b0: SSPA_FCLK_SRC = 24.576M - 2'b1: SSPA_FCLK_SRC = 245.6M
15	RSVD	RO	0x0	Reserved
14:4	I2S1_SSPA_FCLK_DIV	RW	0x9f	SSPA function clock divider sspa_fclk = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	RSVD	RO	0x0	Reserved
2	I2S1_SSPA_FCLK_EN	RW	0x0	- 1'b0: SSPA function clock gated - 1'b1: SSPA function clock enable

Offset: 0xC0882000+0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
1	I2S1_TXRX_BUS_CLK_EN	RW	0x0	- 1'b0: bus clock gated - 1'b1: bus clock enable
0	I2S1_TXRX_SW_RSTN	RW	0x0	Reset control for audio SSPA and ADMA - 1'b0: Software reset

9.2.4.7.4 AUDIO I2S TX RX CLOCK CONTROL REGISTER (AUDIO_HDMI_CLK_CTRL)

This register controls the audio SSPA and ADMA clock. Both share the same bus clock, while SSPA has a separate functional clock.

Offset: 0xC0882000+0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	RSVD	RO	0x0	Reserved
17:16	HDMI_SSPA_FCLK_SRC_SEL	RW	0x0	SSPA function clock source - 2'b0: SSPA_FCLK_SRC = 24.576M - 2'b1: SSPA_FCLK_SRC = 245.76M
15	RSVD	RO	0x0	Reserved
14:4	HDMI_SSPA_FCLK_DIV	RW	0x9f	SSPA function clock divider sspa_fclk = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	RSVD	RO	0x0	Reserved
2	HDMI_SSPA_FCLK_EN	RW	0x0	- 1'b0: SSPA function clock gated - 1'b1: SSPA function clock enable
1	HDMI_BUS_CLK_EN	RW	0x0	- 1'b0: bus clock gated - 1'b1: bus clock enable
0	HDMI_SW_RSTN	RW	0x0	Reset control for audio SSPA and ADMA - 1'b0: Software reset - 1'b1: Release reset

9.2.4.7.5 AUDIO FM TX RX CLOCK CONTROL REGISTER (AUDIO_I2S0_TX_RX_CLK_CTRL)

This register controls the audio SSPA and ADMA clock. Both share the same bus clock, while SSPA has a separate functional clock.

Offset: 0xC0882000+0x60				
Bits	**Field**	**Type**	**Reset**	**Description**

Offset: 0xC0882000+0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	RSVD	RO	0x0	Reserved
17:16	I2S0_SSPA_FCLK_SRC_SEL	RW	0x0	SSPA function clock source - 2'b0: SSPA_FCLK_SRC = 24.576M - 2'b1: SSPA_FCLK_SRC = 245.76M
15	RSVD	RO	0x0	Reserved
14:4	I2S0_SSPA_FCLK_DIV	RW	0x9f	SSPA function clock divider sspa_fclk = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	RSVD	RO	0x0	Reserved
2	I2S0_SSPA_FCLK_EN	RW	0x0	- 1'b0: SSPA function clock gated - 1'b1: SSPA function clock enable
1	I2S0_TXRX_BUS_CLK_EN	RW	0x0	- 1'b0: bus clock gated - 1'b1: bus clock enable
0	I2S0_TXRX_SW_RSTN	RW	0x0	Reset control for audio SSPA and ADMA - 1'b0: Software reset - 1'b1: Release reset

9.2.4.7.6 RCPU PWMx CLOCK CONTROL REGISTER (R_PWMx_CLK_RST)

x=0/1/2/3/4/5/6/7/8/9

Offset: 0xC0888000+0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	RO	0x0	Reserved
18:8	FNCLKDIV	RW	0x0	This field defines the division factor for the functional clock source.
7:6	RSVD	RO	0x0	Reserved
5:4	FNCLKSEL	RW	0x0	Functional Clock Selection: - 2'b0: 245.76 MHz - 2'b1: 24.576 MHz - All others are reserved
3	RSVD	RO	0x0	Reserved
2	RST	RW	0x1	PWM Reset Generation This field resets both APB and Functional domains - 1'b0: No Reset - 1'b1: Reset
1	FNCLK	RW	0x0	PWM Functional Clock Enable/Disable - 1'b0: Clock off - 1'b1: Clock on

Offset: 0xC0888000+0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
0	APBCLK	RW	0x0	PWM APB Bus Clock Enable/Disable - 1'b0: Clock off - 1'b1: Clock on

9.3 JTAG

9.3.1 Introduction

JTAG provides a way to view the states and access all resources of the chip, including processors and peripherals.

JTAG logic consists of

- Test-Access Port (TAP) Controller
- TAP pins
- Instruction Register
- Test Data Registers (TDRs)

The registers include

- Boundary Scan Register (BSR) to control the IO pins directly
- Bypass Register
- Device Identification (ID) Register
- Data-specific registers

Data is shifted into all registers serially, most significant bit (MSB) first. The JTAG interface is controlled through five dedicated TAP controller pins:

- TDI
- TMS
- TCK
- TRSTn
- TDO

9.3.2 Features

- Provide of access to IEEE Std. 1149.1 compatible registers such as IDCODE, BYPASS, EXTEST, etc. through the JTAG port
- Support for hardware/software debugging via the core TAP controller in Concatenation mode

Note. For detailed explanations of these terms and TAP controller states, refer to IEEE Std. 1149.1

9.3.3 Functional Description

There are two JTAG interfaces on the chip named

- **Primary JTAG**
- **Secondary JTAG**

and they can be selected and configured to connect to either X60™ or RCPU processors, in particular

- **Primary JTAG** is enabled at default
- **Secondary JTAG** can be configured as follows:

	CORE_SEL=0	**CORE_SEL=1**
JTAG_SEL=0	Secondary JTAG disabled	Secondary JTAG disabled
JTAG_SEL=1	Secondary JTAG routes to X60™	Secondary JTAG routes to RCPU

The typical scenario for both JTAG usage is that one JTAG connects to X60™ and the other connects to RCPU for debugging simultaneously.

9.4 DMA

9.4.1 Introduction

The Direct-Memory Access Controller (DMAC) is used to transfer data directly between peripheral devices and memory without involving the central processing unit (CPU) allowing the latter to focus on other processing asks.

DMA enables faster and more efficient data transfer bypassing the involvement of CPU in the data transfer process. Instead, a dedicated DMA controller takes over the data transfer operation, accessing the system's memory and the peripheral device directly. This reduces CPU overhead and improves overall system performance.

In DMA transactions, every DMA request from a peripheral device generates a bus transaction. The DMA controller can manage different data transfer types in DMA Flow-Through Mode through 16 configurable DMA channels as tabled below.

	Internal Memory	**External Memory**	**Internal Peripheral**	**External Peripheral**
Internal Memory	Flow-Through Mode	___	___	___
External Memory	Flow-Through Mode	Flow-Through Mode	___	___
Internal Peripheral	Flow-Through Mode	Flow-Through Mode	___	___
External Peripheral	Flow-Through Mode	Flow-Through Mode	___	___

9.4.2 Features

- Two instances of the DMA Controller, one for secure and one for non-secure domains, to handle data transfers
- Support for memory-to-memory, peripheral-to-memory and memory-to-peripheral transfers in Flow-through mode
- Support for Flow-through mode for transfers between Flash and DDR
- Implementation of a priority mechanism to process active channels (4 channels with outstanding DMA requests at any time)
- Each of the 16 channels is allow to operate for descriptor-fetch or non-descriptor-fetch transfers
- Support for special descriptor modes (descriptor comparison and descriptor branching)
- Retrieval of trailing bytes from the receive peripheral device buffers
- Support for programmable data-burst sizes (8, 16, 32 or 64 bytes) and configurable peripheral device data widths (byte, half-word or word)
- Support for up to 8191 bytes of data transfer per descriptor, larger transfers can be performed by chaining multiple descriptors
- Support for flow-control bits to manage requests from peripheral devices, requests are not processed unless the flow-control bit is set.

9.4.3 Functional Description

9.4.3.1 DMA channel

Each one of the 16 DMA channels can be controlled by eight 32-bit registers which are categorized as follows:

- DMA Descriptor Address Registers
- DMA Source Address Registers
- DMA Target Address Registers
- DMA Command Registers

Each channel can be configured for different types of transfers, and processes data in increments of the device's burst size and delivers it according to the device port width, both of which are set in the channel registers based on FIFO depth and bandwidth requirements.

When multiple channels are active, the DMA Controller services them in bursts. After each burst, DMA Controller switches context to another active channel. The switching is determined by whether the channel is active, if its target device is requesting service, and by its priority level.

9.4.3.1.1 DMA Channel-Priority Scheme

The DMA channel-priority scheme ensures peripherals are serviced based on their bandwidth requirements. Higher-priority channels handle high-bandwidth peripherals more frequently, while lower-priority channels manage lower-bandwidth peripherals.

DMA channels are divided into **4 sets**, each containing **4 channels**, as follows:

- **Set 0** → Highest priority (for peripherals with strict latency needs)

- **Set 1** → Higher priority (for memory-to-memory transfer and peripherals with latency needs)
- **Set 2** → Lower priority (for memory-to-memory transfers and low-bandwidth peripherals)
- **Set 3** → Lowest priority (for memory-to-memory transfers and low-bandwidth peripherals)

Details about channel priority are tabled below.

Set	Channels	Priority	Number of Times Served
0	0, 1, 2, 3	Highest	4 / 8
1	4, 5, 6, 7	Higher than 2 and 3. Lower than 0.	2 / 8
2	8, 9, 10,11	Higher than 3. Lower than 0 and 1.	1 / 8
3	12, 13, 14, 15	Lowest	1/ 8

Channels within each set follow a **round-robin priority**. When all channels are active:

- **Set 0** is serviced **four** times in every **eight servicing cycles**
- **Set 1** is serviced **twice**
- **Sets 2 and 3** are each serviced **once**

[Example] If all channels request data transfers, the servicing pattern is as follows:

Set 0 → **Set 1** → **Set 0** → **Set 2** → **Set 0** → **Set 1** → **Set 0** → **Set 3**, then repeats.

9.4.3.1.2 Channel States

The following states apply to the DMA channels:

- **Uninitialized**
  Occurs after a reset. The<STOPINTR> field in the DMA Channel Control/Status Registers is set.
- **Not Running**
  The channel is configured but not yet active because the <RUN> field in the DMA Channel Control/Status Registers is not set, then two transfers are possible as follows:
  - **Descriptor-fetch transfer**
    A valid descriptor is loaded into the DMA Descriptor Address Registers. The <STOPINTR> field is cleared when the DMA Controller updates the DMA Descriptor Address Registers.
  - **No-Descriptor-fetch transfer**
    The DMA Source Address Registers, DMA Target Address Registers and DMA Command Registers are programmed, but <STOPINTR> remains set until the channel starts running.
- **Running**, then two transfers are possible as follows:
  - **Descriptor-fetch transfer**
    After programming DDADR_H/DDADR_L and setting <RUN>, the DMA fetches eight words of descriptors from memory, keeping <STOPINTR> clear.
  - **No-Descriptor-fetch transfer**
    After programming the DMA Source Address Registers, DMA Target Address Registers, DMA Command Registers (if accessing memory beyond the 4GB limit, then configure the high-level address registers in the same sequence as the first four address registers) and setting <RUN>, the channel

clears <STOPINTR>, skips the Descriptor-fetch Running state and enters either the “Wait for Request” or “Transfer Data” state.

- **Wait for a request**
  Occurs as the channel waits for a request before starting data transfer. <STOPINTR> is clear.
- **Transfer data**
  Data is transferred between the source and the target. <STOPINTR> is clear.
- **Channel error**
  The channel with the error remains in the stopped state until software clears the error condition, re-initializes the channel and sets the <RUN> field and the <BUSERRINTR> field in the DMA Channel Control/Status Registers.
- **Stopped**
  The channel is stopped. The <STOPINTR> field is set. Then two transfers are possible as follows:
  - **No-Descriptor-fetch transfer**
    A stopped channel is re-initialized by updating the DMA Source Address Registers, DMA Target Address Registers, DMA Command Registers (including the next four registers (i.e. high-address registers) if required), then setting the <RUN> field.
  - **Descriptor-fetch transfer**
    A stopped channel is re-initialized by updating the DMA Descriptor Address Registers and setting<RUN>.

The summary of the DMA channel states is tabled below.

Descriptor Mode	Software Configuration	<Run>	<Stop interrupt>	Resulting Channel State
Descriptor-Fetch mode	Power-up	0	1	Uninitialized
	Write to DDADR before DCSR[RUN] is set **(recommended)**	0	0	Valid Descriptor, not running
	Set DCSR[RUN] after writing to DDADR **(recommended)**	1	0	Running
	Set DCSR[RUN] before writing to DDADR（**not recommended)**	1	1	Invalid
	Write to DDADR after DCSR[RUN] is set **(not recommended)**	1	0	Descriptor fetch, running.
	Stop running channel by clearing DCSR[RUN] and DCSR[MASKRUN]	0	0 -> 1	Channel, if not immediately, eventually switches to a stopped state (identified by DCSR[STOPINTR] toggling from low to high).
No-Descriptor-Fetch mode	Power-on	0	1	Uninitialized
	Write to DSADR, DTADR and DCMD before DCSR[RUN] is	0	1	Valid Descriptor, not running

Descriptor Mode	Software Configuration	<Run>	<Stop interrupt>	Resulting Channel State
	set**(recommended)**			
	Set DCSR[RUN] after configuring DSADR, DTADR, and DCMD**(recommended)**	1	0	Running
	Set DCSR[RUN] before configuring DSADR, DTADR, and DCMD **(not recommended)**	1	0	Wait for Request, running. Channel uses current DSADR, DTADR and DCMD for the transfer, potentially leading to unpredictable results.
	Stop running channel, by clearing DCSR[RUN] and DCSR[MASKRUN	0	0 -> 1	Channel, if not immediately, eventually switches to a stopped state (identified by DCSR[STOPINTR] toggling from low to high)

9.4.3.2 DMA Descriptors

A DMA Descriptor is an 8-word block (32-bits per word) aligned to a 32-byte boundary in memory. Details are tabled below.

Word Index	Description
Word [0]	DMA Descriptor Address Register + Flag Bit
Word [1]	DMA Source Address Register
Word [2]	DMA Target Address Register
Word [3]	DMA Command Register
Word [4]	High 32-bit Descriptor Address Register
Word [5]	High 32-bit Source Address Register
Word [6]	High 32-bit Target Address Register
Word [7]	Reserved

The DMAC can operate in two distinct modes based on the DCSR[NODESCFETCH] bit as follows:

- DCSR[NODESCFETCH] = 0 - Descriptor-fetch transfer
- DCSR[NODESCFETCH] = 1 - No-Descriptor-fetch transfer

Instead, about DCSR[LPAE_EN] bit - Long Physical Address Extension Enable:

- DCSR[LPAE_EN] = 0 - 4 words Descriptor
- DCSR[LPAE_EN] = 1 - 8 words Descriptor

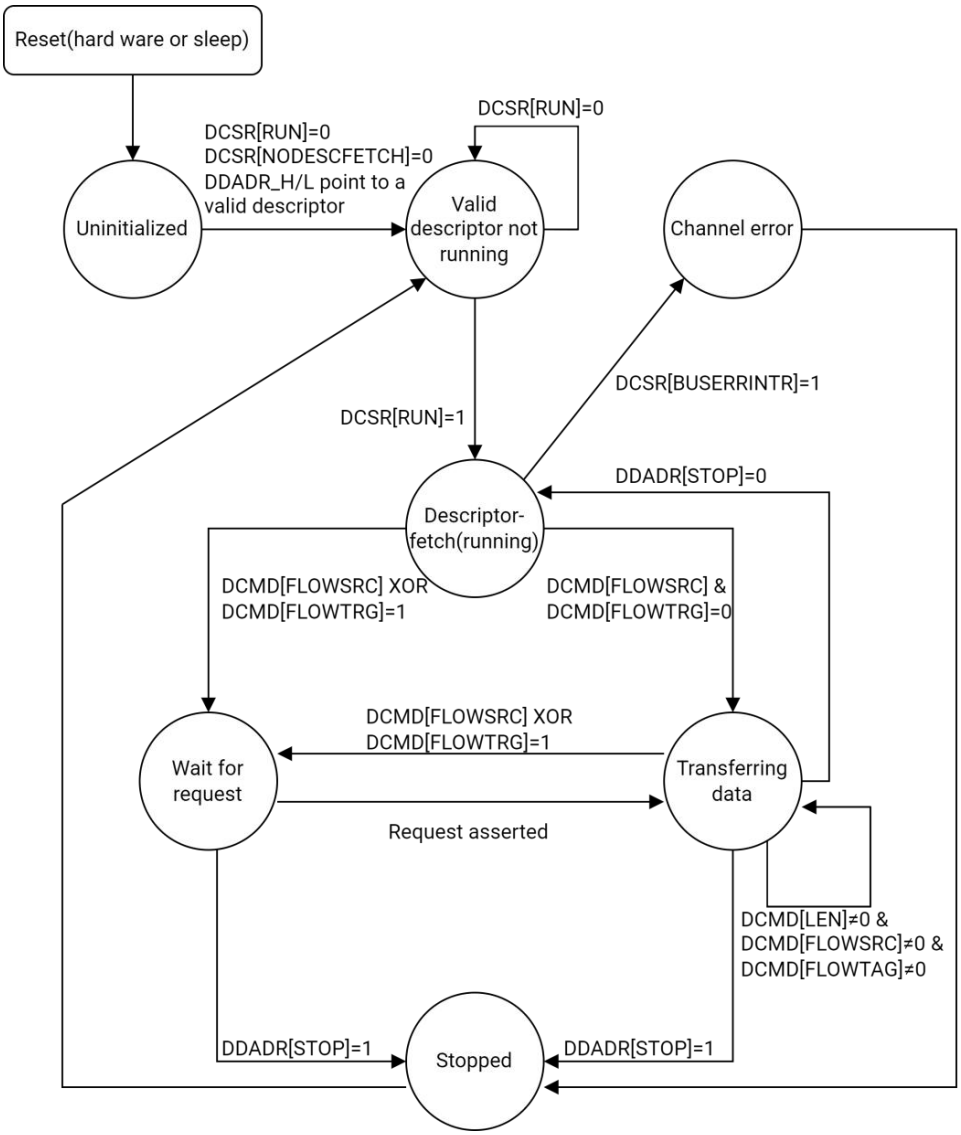
9.4.3.2.1 Descriptor-Fetch Transfer Operation

The descriptor-fetch transfer (<NODESCFETCH> field in the DMA Channel Control/Status Registers = 0) operates in the following manner:

- **Setting up the descriptor fetch**
  - First, the software must clear the <RUN> field and then clear the <NODESCFETCH> field
  - Next, the software writes a valid Descriptor address to the DMA Descriptor Address Register
  - Finally, the software sets <RUN>, allowing the DMAC to fetch the Descriptor (four-word or eight-word) from memory, as indicated by the DMA Descriptor Address Register.
- **Starting the data transfer**
  - The channel either waits for a request or starts the transfer, depending on DCMD[FLOWSRC] or DCMD[FLOWTRG].
  - The channel transfers data until reaching the smaller of <DMA_SIZE> or <LEN> (from the DMA Command Registers).
  - After reaching this limit, the channel will
    - Either wait for the next request
    - Or continue transferring until <LEN> reaches zero.
- **Fetching the next descriptor or stopping**
  - Once the transfer completes, the channel will
    - Either fetch a new descriptor from memory
    - Or stop, based on the <STOP> field in the DMA Descriptor Address Registers
- **Handling Errors**
  - If an error occurs during descriptor fetching, the channel enters the “Stopped” state
  - To resume, software must:
    - Clear the error condition
    - Re-initialize the channel
    - Set the <RUN> field again
- **Switching Modes**
  - If switching between Descriptor-Fetch Mode and No-Descriptor-Fetch Mode, the channel must be stopped first before changing the mode.
  - A Descriptor-fetch transfer will only occur if the **DMA Descriptor Address Register** is loaded and the <RUN> field is set.
- **Loading DMA Descriptors**
  - Although the DMA Descriptor Address Register is loaded by software, other registers (DMA Source Address, Target Address and Command Registers) are loaded indirectly from the DMA Descriptors
  - When the <RUN> field is set, the DMA Descriptors are transferred into the corresponding DMA channel registers
- **Special Stop Condition**
  - Bit [0] (<STOP>) of word [0] in a DMA Descriptor marks the final Descriptor in the list

- When a Descriptor with the stop bit is loaded into a Channel register, the channel will stop after completing the transfer, but the stop bit itself does not affect how Descriptor fields are loaded

The summary of the operations is depicted below.



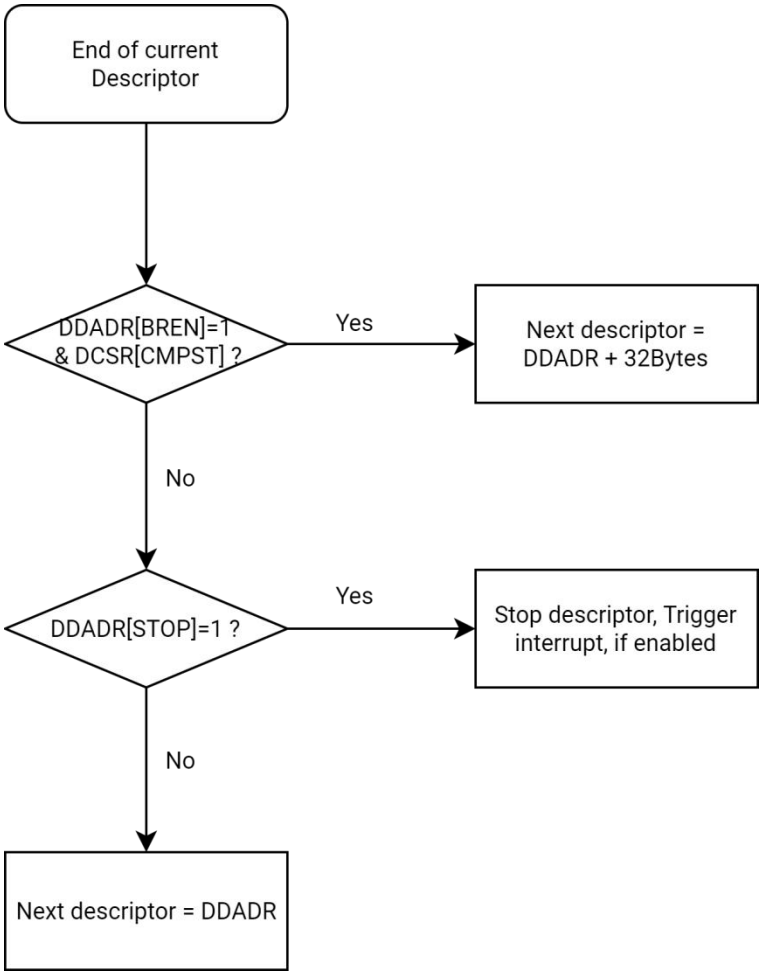
9.4.3.2.1.1 Descriptor Branching

The Descriptor Branching operates in the following manner:

- **Determining the Next Descriptor Address**
  - If both <BREN> (in the DMA Descriptor Address Registers) and <CMPEN> (in the DMA Channel Control/Status Registers) are set, the DMAC fetches the next descriptor from Current Descriptor Address + 32 bytes.
  - If these bits are cleared, the DMAC fetches the next Descriptor from the same address in the DMA Descriptor Address Register

- **Applicability of <BREN>**
  - The <BREN> field is only relevant for Descriptor-fetch transfers when the <NODESCFETCH> field (in the DMA Channel Control/Status Registers) is cleared.

The summary of the operations is depicted below.



9.4.3.2.2 No-Descriptor-Fetch Transfer Operation

The typical no-Descriptor-fetch transfer (<NODESCFETCH> = 1) operates in the following manner:

- **Initialization**
  - After a reset, the channel is in an uninitialized state
  - The software must:
    - Clear the <RUN> field
    - Set the <NODESCFETCH> field
    - Write a valid source physical address to the DMA Source Address Registers
    - Write a target physical address to the DMA Target Address Registers
    - Write a command to the DMA Command Registers
  - Finally, software must set the <RUN> field to start the operation

Note. The DMA Descriptor Address Registers are reserved in this mode and must not be written.

- **Data Transfer**
  - No Descriptor fetch occurs
  - Based on the <FLOWSRC> and <FLOWTAG> fields in the DMA Command Registers, the channel will
    - Either wait for a request
    - Or start the data transfer immediately
  - The channel transfers data until it reaches the smaller value between <DMA_SIZE> and <LEN>
  - After this, the channel will
    - Either wait for the next request
    - Or continue the data transfer (depending on the priority of enabled DMA channels).
  - The channel stops automatically when <LEN> reaches zero.

- **Handling Stop Interrupts (<STOPIRQEN>)**
  - Setting the <STOPIRQEN> field in the DMA Channel Control/Status Registers may cause the DMAC to trigger a stop interrupt prematurely, potentially even before the channel enters active-run mode
  - Refer to <STOPIRQEN> and <STOPINTR> fields in the DMA Channel Control/Status Registers for details on the RUN condition

- **End-of-Receive (<EOR>) Detection**
  - To detect if a channel stops due to an End-of-Receive (EOR) from a peripheral, software must check the <EORINT> field in the DMA Channel Control/Status Registers
  - The EOR signal, sent from the peripheral when transferring the last byte, informs the DMA to stop once the trailing byte is completely transferred
  - If an EOR condition stops the channel, <EORINT> is set

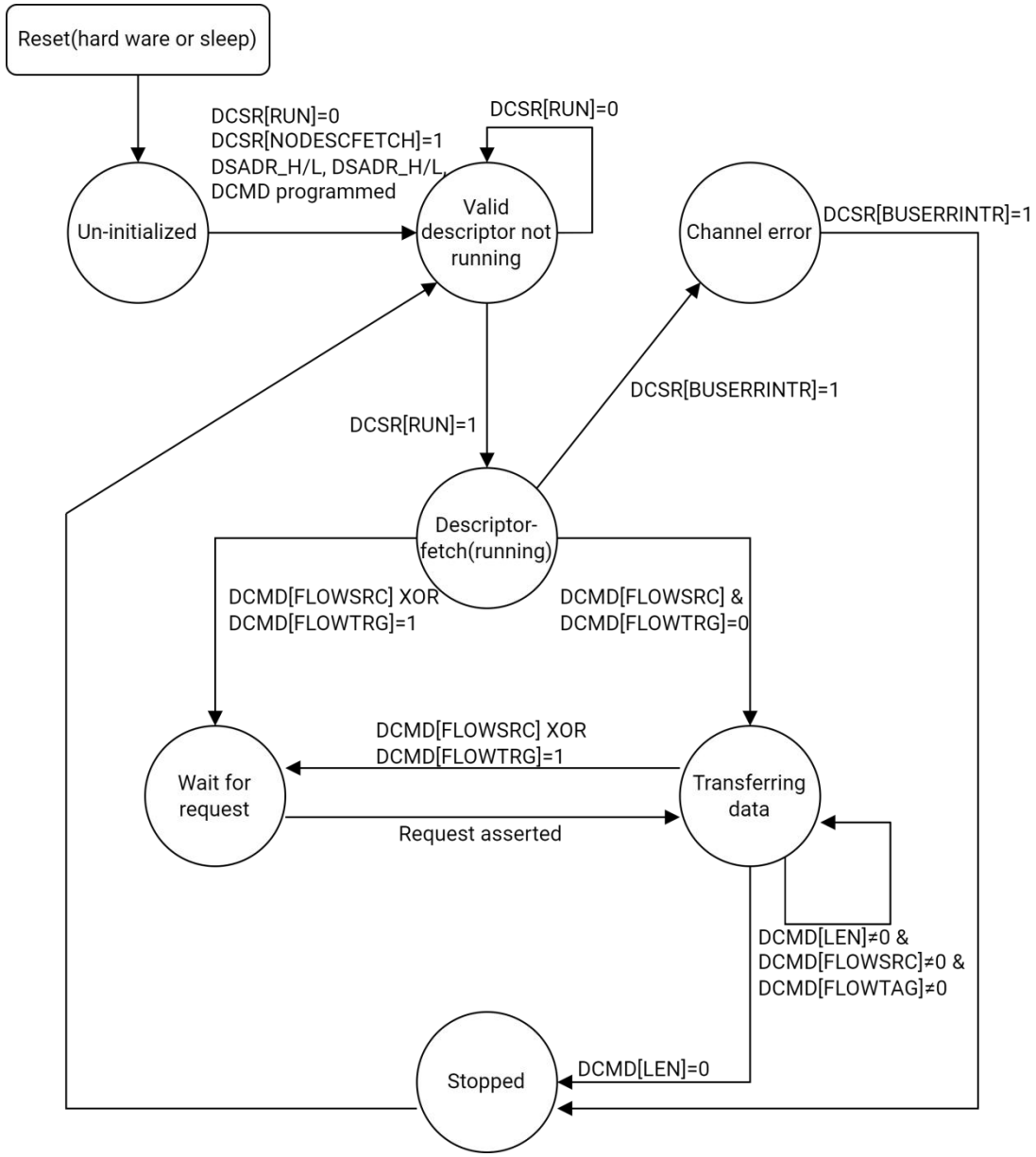
  Note. Refer to DMA Channel Control/Status Registers for more details

- **Normal Stop Detection**
  - To detect a normal stop, use the end interrupt (<ENDINTR>) instead of the stop interrupt (<STOPINTR>).

  Note. Refer to DMA Channel Control/Status Registers for more details

- **Handling Errors**
  - If an **error occurs**, the channel enters the **Stopped state** and remains there **until** software:
    - **Clears** the error condition
    - **Sets** the <RUN> field again

The summary of the operations is depicted below



9.4.3.3 Transferring Data

The on-chip peripherals connected to the DMA via the peripheral bus operate as flowthrough transfers. Although the source or destination of a DMA transfer is usually a peripheral intended to be used as a source or sink of DMA data, the DMAC can transfer data to or from any memory location through memory-to-memory moves.

9.4.3.3.1 Servicing Internal Peripherals

The DMAC provides DMA requests to the DMA Request to Channel Map Registers (0-63 and 64-99), each containing five-bit channel number for every possible DMA request.

These possible peripheral requests are mapped to 16 available channels.

- If the on-chip peripheral address is located in the DMA Source Address Registers, the <FLOWSRC> field must be set to allow the processor to wait for the request before it initiates the transfer.
- If the on-chip peripheral address is located in the DMA Target Address Registers, the <FLOWTAG> field must be set.

Additionally, if the <ENDIRQEN> field is set, a DMA interrupt will be requested at the end of the last cycle, corresponding to the byte that caused the <LEN> field to decrease to zero.

9.4.3.3.2 Servicing Internal Peripherals Using Flowthrough DMA Read Cycles

A flowthrough DMA Read begins when an on-chip peripheral sends a request to a channel in the DMAC while the channel is running. The number of bytes to be transferred is specified using the <LEN>. The following process begins when the request is recognized:

- The DMAC instructs the Memory Controller to read the required number of bytes addressed by DMA Source Address Registers into a 32-byte buffer in the DMAC.
- The DMAC transfers the data to the peripheral device addressed in the DMA Target Address Registers. The <WIDTH> field in the DMA Command Registers specifies the width of the internal peripheral to which the transfer is being made.
- At the end of the transfer, DMA Source Address Registers is incremented, and the <LEN> field is decreased by the smaller of <LEN> and <DMA_SIZE>.

Use the following settings for the DMAC register bits for a flowthrough DMA Read from an internal peripheral:

- <SRCADDR_H> field and <SRCADDR_L> in the DMA Source Address High/Low register = memory address
- <TRGADDR_H> field and <TRGADDR_L> in the DMA Target Address High/Low register = internal peripheral address
- <INCSRCADDR> field in the DMA Command Registers = 1
- <INCTAGADDR> field in the DMA Command Registers = 0
- <FLOWSRC> field in the DMA Command Registers = 0
- <FLOWTAG> field in the DMA Command Registers = 1

9.4.3.3.3 Servicing Internal Peripherals Using Flowthrough DMA Write Cycles

A flowthrough-DMA Write begins when an on-chip peripheral sends a request to a channel in the DMAC while the channel is running. The number of bytes to be transferred is specified using the <DMA_SIZE> field.

When the request is recognized, the following process begins:

- The DMAC processes the request by transferring the required number of bytes from the peripheral device addressed by DMA Source Address Registers into a DMAC buffer
- The DMAC transfers the data to the Memory Controller. The <WIDTH> field specifies the width of the internal peripheral from which the transfer is being made
- At the end of the transfer, DTADRx is increased and <Length of the transfer in bytes> is decreased by the smaller of <Length of the transfer in bytes> and <Maximum burst size>

Use the following settings for the DMAC register bits for a flowthrough-DMA Write from an internal peripheral:

- <SRCADDR_H> field and <SRCADDR_L> = internal peripheral address
- <TRGADDR_H> field and <TRGADDR_L> = memory address
- <INCSRCADDR> = 0
- <INCSRCADDR> = 1
- <FLOWSRC> = 1
- <FLOWTAG> = 0

Memory-to-memory moves do not involve request signals. For a memory-to-memory move, the processor writes to the <Run> field indicated by the channel that is configured to perform a memory-to-memory move. The <FLOWSRC> and <FLOWTAG> fields must be cleared by software once the Descriptor is fetched. The transfer then begins.

If <ENDIRQEN> is set, a DMA interrupt is requested at the end of the last cycle, corresponding to the byte that causes the <LEN> field to decrease to zero.

9.4.3.3.4 Memory-to-Memory Moves: Flowthrough DMA Read/Write Cycles

A flowthrough DMA memory-to-memory Read or Write begins when the processor the DCSR[RUN] bit. If the channel is in a Descriptor-fetch transfer, it fetches the four-word Descriptor. The <FLOWSRC> and <FLOWTAG> fields must be cleared for a memory-to-memory move. The channel starts transferring data without waiting for a PREQ or DREQ assertion. The number of bytes to be transferred is specified using <LEN>. Processing proceeds as follows:

- The DMAC instructs the Memory Controller to read the required number of bytes addressed by the DMA Source Address Registers into a 16-byte buffer in the DMAC
- The DMAC generates a Write cycle to the location addressed by the DMA Target Address Registers
- At the end of the transfer, both DMA Source Address Registers and DMA Target Address Registers are incremented, and the <LEN> field is decreased by the smaller of <LEN> and <DMA_SIZE>

Use the following settings for the DMAC register bits for flowthrough memory-to-memory moves:

- <SRCADDR_H> field and <SRCADDR_L> = internal peripheral address
- <TRGADDR_H> field and <TRGADDR_L> = memory address
- <INCSRCADDR> = 1
- <INCSRCADDR> = 1
- <FLOWSRC> = 0
- <FLOWTAG> = 0

9.4.3.4 Programming Tips

9.4.3.4.1 Software Management Requirements

Information that must be maintained on a per-stream basis (such as the memory address, the peripheral address, the transfer count, and the implied direction of data flow) is stored in Descriptor registers in the DMAC. These Descriptor registers are loaded from memory locations specified by the software. Multiple DMA Descriptors can be chained together in a list, allowing a DMA channel to transfer data to and from multiple separate locations.

The Descriptor-based DMA design allows Descriptors to be added dynamically to an active DMA channel Descriptor chain, which is particularly useful in applications that involve network-transmit lists and network-receiver buffer-free lists.

Each data demand generated by a peripheral involves either a memory data Read or Write operation. A peripheral must not request a DMA transfer unless it is ready to read or write the entire data block (8, 16, or 32 bytes) and can handle any trailing bytes that may occur at the end of a DMA transfer.

9.4.3.4.2 Programmed I/O Operations

The processor can read from and write to the peripheral registers and FIFOs on the peripheral bus. Internal registers of the peripheral must be accessed using word-access loads and stores. Both the internal register space and FIFO space must be mapped as non-cacheable. Byte and half-word accesses to internal registers are not allowed.

However, some peripherals on the peripheral bus allow their FIFOs to be accessed using byte, half-word, or word-access loads and stores. For specific details, refer to the individual peripheral sections.

9.4.3.4.3 Instruction Ordering

The DMAC executes programmed I/O instructions in the order specified by the software. References to internal addresses generally complete faster than those issued to external addresses. This means that memory accesses can be sent in one order and completed in a different order.

The DMAC ensures that memory references made by a single DMA channel are presented to memory in the specified order, with Descriptor fetches occurring between data blocks. However, the order in which accesses are completed cannot be guaranteed unless the channels refer to only one type of memory (either external memory or internal SRAM).
The channel references must not involve both internal and external memory in a DMA Descriptor chain for the following operations:

- Self-modifying DMA Descriptor chains.
- Channels that write data blocks followed by status blocks while another channel (typically the processor) polls a field in the status block.

9.4.3.4.4 Misaligned Memory Accesses

The DMAC is a 64-bit device that can access memory on byte-aligned boundaries. The DMAC may encounter misaligned addresses (i.e., addresses not aligned to a 64-bit boundary) while it accesses memory.

Only the following type of data transfers may involve misaligned addresses:

- Memory-to-memory transfers
- Memory-to-peripheral transfers or peripheral-to-memory transfers. In this case, the peripheral addresses are 32-bit aligned

In compare-descriptor mode, addresses must be 64-bit aligned.

To handle misaligned data, the DMAC uses channel-specific alignment buffers, which hold either the leading or lagging misaligned data. These buffers must be empty when the DMAC performs a context switch to service the

next pending channel. Once a Descriptor transfer is completed, the DMAC ensures that all data in the alignment buffers is properly flushed to its respective targets.

Because the DMAC incurs overhead when working with misaligned data, it is recommended to restrict memory addresses to 8-byte boundaries. For optimal DMAC and Memory Controller performance, align the source and target addresses to 32-byte boundaries.

By default, during data transfers, the DMA Controller forces the least significant 3 bits of all external addresses and the least significant 2 bits of all peripheral addresses to zero. To enable byte-aligned addressing, software must activate the Alignment Register. Refer to the DMA Alignment Register for further details.

9.4.3.5 How DMA Handles Trailing Bytes

DMA normally transfers bytes equal to the transaction size specified by <DMA_SIZE>. However, when the Descriptor is reaching its end, the number of trailing bytes in the <LEN> field could be smaller than the transfer size. In this case, the DMA can transfer the exact number of trailing bytes if both the <FLOWSRC> and <FLOWTAG> fields are 0, or if it receives a corresponding request from the on-chip/off-chip peripheral or companion chip. The following cases are possible:

- **Memory-to-memory moves**
  The DMA transfers bytes equal to the smaller of <LEN> or <DMA_SIZE>.
- **Companion-chip-related transfers (flowthrough)**
  The companion chip must assert the request to allow the DMA to handle the trailing bytes. If the request is asserted, the DMA transfers a number of bytes equal to the smaller of <LEN> and <DMA_SIZE>.
- **Memory-to-on-chip-peripheral transfers**
  Most of the on-chip peripherals send a request for trailing bytes. The DMA transfers a number of bytes equal to the smaller of <LEN> and <DMA_SIZE>.
- **On-chip-peripheral-to-memory transfers**
  Special handshaking signals and interrupts are employed for transferring trailing bytes from an on-chip peripheral to memory. The conditions that use the handshaking signals and interrupts are explained below:
  - **End of Packet (EOP)**
    The peripheral receives its last data sample from an external codec and detects an EOP based on its Receive protocol. Any remaining data samples in the peripheral Receive FIFO are treated as trailing bytes. The peripheral can be programmed to initiate a DMA request, even if it has fewer bytes than its Receive trigger threshold. The DMA responds to this request and reads out the trailing bytes. CPU intervention is not required as long as the Descriptor chain has not ended.
  - **Time Out (TO)**
    Peripherals that do not support EOP protocols use a time-out mechanism to determine if they have received their last data sample. Any remaining data samples in the peripheral Receive FIFO are treated as trailing bytes. The peripheral can be programmed to initiate a DMA request, even if it has fewer bytes than its Receive trigger threshold. The DMA responds to the DMA request and reads out the trailing bytes. CPU intervention is not required as long as the Descriptor chain has not ended.
  - **End-of-Descriptor chain (EOC)**
    Indicates that a DMA channel is at the end of its last Descriptor. After the current transfer, <Length of the transfer in bytes> = 0 and <Stop> = 1. DMA signals the peripheral on an EOC and the peripheral interrupts the CPU to retrieve any trailing bytes. EOC is the only trailing-bytes case that requires programmed I/O to retrieve data.

- **Request-after-channel-stops (RAS)**
  Status bit in the DMA Channel Control/Status Register 0-15. This bit is set when a peripheral asserts a DMA request after the channel to which the peripheral is mapped has stopped. Refer to **Section 9.4.4.1** for DMA Channel Control/Status Register 0-15 for details.

  Note. When a peripheral signals either an EOP or a TO from an external device, the DMAC sets the end-of-receive (EOR) status bit in the corresponding channel Control Status register (DCSR).

[Example]

Handling of various trailing bytes using EOR, EOC, and RAS. The peripheral signals a DMA request to service trailing bytes in its Receive FIFO (RxFIFO). The current Descriptor <LEN> is equal to or greater than the trailing-bytes count, then

- The peripheral signals a Receive DMA request
- The DMAC responds and reads out all trailing bytes including the last byte
- The peripheral signals an EOR
- The DMAC transfers all trailing bytes to the channel target and then updates the <EORINT> field
- The DMA channel can be configured to stop, jump, or just wait for another request after receiving EOR, depending on the <EORSTOPEN> and <EORJMPEN> fields in the DMA Channel Control/Status Registers. The <EORSTOPEN> field must be cleared before restarting a channel
- The <EORSTOPEN> field set indicates that all trailing bytes were read and transferred to the required target

9.4.3.6 DMA Connectivity & Assignments

There are two DMAs in K1 SoC as follows:

- Non-secure DMA
- Secure DMA

Both have the same features, however, secure DMA is used in the secure environment, while non-secure DMA is used in the non-secure environment.

The Direct-Memory Access Controller (DMAC) transfers data to and from memory in response to requests generated by peripheral devices or companion chips. Peripheral devices do not directly provide addresses or commands to the Memory Controller. Instead, the states required to manage a data stream are maintained within DMA Channels. Each DMA request from a peripheral device triggers a memory-bus transaction. The processor can directly access the peripheral bus by using the DMA controller which acts as a DMA bridge to bypass the DMA of the system.

The **DMA request numbers** for **non-secure DMA** peripherals are tabled below.

DRQ	Description	Comments
3	Request for UART0 TxReq	UART0 (0xD4017000)
4	Request for UART0 RxReq	UART0 (0xD4017000)
5	Request for UART2 TxReq	UART2 (0xD4017100)
6	Request for UART2 RxReq	UART2 (0xD4017100)
7	Request for UART3 TxReq	UART3 (0xD4017200)
8	Request for UART3 RxReq	UART3 (0xD4017200)

DRQ	Description	Comments
9	Request for UART4 TxReq	UART4 (0xD4017300)
10	Request for UART4 RxReq	UART4 (0xD4017300)
11	Request for I2C0 TxReq	I2C0 (0xD4010800)
12	Request for I2C0 RxReq	I2C0 (0xD4010800)
13	Request for I2C1 TxReq	I2C1 (0xD4011000)
14	Request for I2C1 RxReq	I2C1 (0xD4011000)
15	Request for I2C2 TxReq	I2C2 (0xD4012000)
16	Request for I2C2 RxReq	I2C2 (0xD4012000)
17	Request for I2C4 TxReq	I2C4 (0xD4012800)
18	Request for I2C4 RxReq	I2C4 (0xD4012800)
19	Request for SPI3 TxReq	SPI3 (0xD401C000)
20	Request for SPI3 RxReq	SPI3 (0xD401C000)
21	Request for I2S0 TxReq	I2S0 (0xD4026000)
22	Request for I2S0 RxReq	I2S0 (0xD4026000)
23	Request for I2S1 TxReq	I2S1 (0xD4026800)
24	Request for I2S1 RxReq	I2S1 (0xD4026800)
25	Request for UART5 TxReq	UART5 (0xD4017400)
26	Request for UART5 RxReq	UART5 (0xD4017400)
27	Request for UART6 TxReq	UART6 (0xD4017500)
28	Request for UART6 RxReq	UART6 (0xD4017500)
29	Request for UART7 TxReq	UART7 (0xD4017600)
30	Request for UART7 RxReq	UART7 (0xD4017600)
31	Request for UART8 TxReq	UART8 (0xD4017700)
32	Request for UART8 RxReq	UART8 (0xD4017700)
33	Request for UART9 TxReq	UART9 (0xD4017800)
34	Request for UART9 RxReq	UART9 (0xD4017800)
35	Request for I2C5 TxReq	I2C5 (0xD4013800)
36	Request for I2C5 RxReq	I2C5 (0xD4013800)
37	Request for I2C6 TxReq	I2C6 (0xD4018800)
38	Request for I2C6 RxReq	I2C6 (0xD4018800)
39	Request for I2C7 TxReq	I2C7 (0xD401d000)
40	Request for I2C7 RxReq	I2C7 (0xD401d000)

DRQ	Description	Comments
41	Request for I2C8 TxReq	I2C8 (0xD401d800)
42	Request for I2C8 RxReq	I2C8 (0xD401d800)
43	Request for CAN0 RxReq	CAN0 (0xD4028000)
44	Request for QSPI RxReq	QSPI (0xD420_C000)
45	Request for QSPI TxReq	QSPI (0xD420_C000)

Instead, the **DMA request numbers** for **secure DMA** peripherals are tabled below.

DRQ	Description	Comments
0	Request for UART1 RxReq	UART1 (0xF0612000)
1	Request for UART1 TxReq	UART1 (0xF0612000)
2	Request for SPI2 RxReq	SPI2(0xF0613000)
3	Request for SPI2 TxReq	SPI2 (0xF0613000)
4	Request for I2C3 TxReq	I2C3 (0xF0614000)
5	Request for I2C3 RxReq	I2C3 (0xF0614000)

9.4.4 Register Description

The base addresses of DMA registers are tabled below.

Name	Address
DMA_BASE	0xD4000000
DMA2_BASE	0xF0600000

9.4.4.1 DCSR_x REGISTER

DMA channel control/status registers. These read/write registers contain the control and status bits for the channels.

Offset: 0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24/0x28/0x2C/0x30/0x34/0x38/0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RUN	R/W	0x0	This field allows software to start or stop the DMA channel. 0: Stop the channel 1: Start the channel. If it is cleared during a burst transfer, the burst completes before stopping. If the channel is in a descriptor-fetch transfer and this field is set

Offset: 0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24/0x28/0x2C/0x30/0x34/0x38/0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
				before writing a valid descriptor address to the DMA Descriptor Address Registers, no descriptor fetch occurs. This bit automatically resets when cleared or when the channel stops normally. After stopping, the <STOPINTR> field is set. Software must poll <STOPINTR> to check the channel status or set <STOPIRQEN> to receive an interrupt when the channel stops.
30	NODESCFETCH	R/W	0x0	No-Descriptor Fetch 0: Descriptor-fetch transfer 1: No-descriptor-fetch transfer This bit determines whether the channel operates with or without descriptors: - When set (1), the channel functions as a simple channel with no descriptors. The DMA does not fetch descriptors when the <RUN> field is set or when the current transfer's byte count reaches zero. 1. In this mode, software must manually configure the channel by writing to the DMA Source Address Registers, DMA Target Address Registers, and DMA Command Registers. 2. The DMA Descriptor Address Registers are not used and must not be written. 3. The <RUN> field must be set to start the transfer. - When cleared (0), the DMAC initiates descriptor fetches: 1. When software writes to the DMA Descriptor Address Registers. 2. When the byte count for the current transfer reaches zero.
29	STOPIRQEN	R/W	0x0	Stop Interrupt Enabled This field controls whether an interrupt is generated when the <STOPINTR> field is set. 0: No interrupt is generated if the channel is uninitialized or stopped. 1: An interrupt is generated when the channel is uninitialized or stopped. **Note.** After a system reset, <STOPINTR> is set. If <STOPIRQEN> is already enabled before starting the channel, an interrupt is triggered immediately.
28	EORIRQEN	R/W	0x0	Setting the End-of-Receive interrupt enable This field triggers an interrupt on an EOR condition. Clearing this bit does not generate an EOR-related interrupt. 0: No interrupt is triggered even if the <EORINT> field is set 1: An Interrupt is triggered when <EORINT> is set
27	EORJMPEN	R/W	0x0	Jump to the next descriptor on EOR This field controls the descriptor flow when the mapped

Offset: 0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24/0x28/0x2C/0x30/0x34/0x38/0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
				peripheral signals an EOR to the DMAC. See Descriptor Behavior on End-of-Receive (EOR) for the behavior of the descriptor during this condition. **Note.** This control bit has no effect on the channel for no-descriptor-fetch transfers (<NODESCFETCH> set). The DMAC completes the peripheral-to-memory data transfer on an EOR, regardless of this field. 0: DMAC holds the current descriptor and waits for the mapped peripheral to make another receive request. 1 = DMAC jumps to the channel's next descriptor on receiving an EOR from the mapped peripheral.
26	EORSTOPEN	R/W	0x0	Stop channel on EOR **Note.** This field has no effect on the channel for no-descriptor-fetch transfers (<NODESCFETCH> set). The DMAC completes the peripheral-to-memory data transfer on an EOR, regardless of this field. Setting this field causes the DMAC to stop the channel on an EOR and set the corresponding <STOPINTR> field. If the <STOPIRQEN> field is set when this field is set, an interrupt occurs. 0: DMAC holds the current descriptor and waits for the mapped peripheral to make another receive request. 1: DMAC stops the channel that receives an EOR from the mapped peripheral.
25	SETCMPST	W	0x0	Set descriptor Compare Status 0: No effect on <CMPST> 1: Set <CMPST>，regardless of whether the descriptor is in compare mode (<CMPEN> = 0 in DMA Command Registers).
24	CLRCMPST	W	0x0	Clear descriptor Compare Status 0: No effect on <CMPST> 1: Clear <CMPST>，regardless of compare mode configuration
23	RASIRQEN	R/W	0x0	Request after channel stopped interrupt enable 0: No interrupt when a peripheral requests DMA after the channel stops 1: Triggers an interrupt in <CHLINTR> (DMA Interrupt Register) when a peripheral requests DMA after the channel stops.
22	MASKRUN	W	0x0	Mask <RUN> during a programmed I/O write to this register 0: Software (programmed I/O write) can modify <RUN> during a write transaction 1: Software (programmed I/O write) can not modify <RUN> during a write transaction

Offset: 0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24/0x28/0x2C/0x30/0x34/0x38/0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
21	LPAE_EN	R/W	0x0	Long Physical Address Extension (LPAE) enable This bit enable Long Physical Address Extension feature for both descriptor mode and nondescriptor modes. 0: LPAE feature is disabled. - For Descriptor mode, no need to program DDADR_H register. Descriptors should remain 4 words (32bits per word, aligned on a 16-byte boundary in memory). - For Non-descriptor mode, no need to program DTADR_H and DSADR_H registers. No software SW changes required. 1: LPAE feature is enabled. - For Descriptor mode, Software must program DADR_H register and prepare the 8 words (32bits per word, aligned on a 32-byte boundary in memory). - For Non-descriptor mode, Software must program DTADR_H and DSADR_H registers. - LPAE is a feature that can enable or disable DMA transfer. LPAE and non-LPAE transfers can be interleaved.
20:11	RSVD	R	0	Reserved for future use
10	CMPST	R	0x0	Descriptor Compare Status This field reflects the result of the most recent source and target compare operation in descriptor compare mode (CMPEN = 1 in the DMA Command Registers) 0: Indicates an unsuccessful address compare in descriptor-compare mode. 1: Indicates a successful compare of the current descriptor source and target addresses in descriptor-compare mode.
9	EORINT	R/W1C	0x0	End of Receive Interrupt EORINT pertains only to internal peripherals. This field indicates the status of the mapped peripheral's receive data. It is set after the DMAC reads out the last trailing sample from the peripheral's receive FIFO. The Descriptor Behavior on End-of-Receive (EOR) figure illustrates the behavior of the descriptor during this condition. 0 = DMA continues with current descriptor because the internal peripheral is still actively receiving data 1 = Channel mapped internal peripheral has no data remaining in its receive FIFO and has completed all receive transactions. Refer to the description of <EORJMPEN> for the behavior of the DMAC during this condition. - CMPST is updated only when CMPEN = 1. - This field can be manually set by SETCMPST and cleared by CLRCMPST. - If both SETCMPST and CLRCMPST are written simultaneously, SETCMPST takes priority. - Do not modify this field while the channel is running (RUN = 1),

Offset: 0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24/0x28/0x2C/0x30/0x34/0x38/0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
				as it may cause faulty descriptor behavior. Always stop the channel before updating this field.
8	REQPEND	R	0x0	Request Pending This field indicates a pending request for the DMA channel. 0: No request is pending for the channel 1: A request is pending for the channel - REQPEND is cleared for a channel if that channel has no pending request or the request has just been issued to the memory interface in case of a read or write from the external companion chip to memory. - If the DREQ assertion sets REQPEND and <RUN> is cleared to stop that channel, REQPEND and the internal registers that hold the DREQ assertion information, do not remain set. - If the channel is restarted, REQPEND must be reset by a descriptor that transfers dummy data (for example, a memory-to-memory transfer from a temporary location to another temporary location).
7:5	RSVD	R	0	Reserved for future use
4	RASINTR	R/W	0x0	Request after channel stopped 0: No interrupt 1: Interrupt occurred due to a peripheral request after the channel stopped - This bit is reset by writing a 1.
3	STOPINTR	R	0x1	Stop Interrupt This bit indicates the current state of the channel: 0: Channel is running 1: Channel is in uninitialized or stopped state This is a read-only bit that reflects the channel state. - Software must clear <STOPIRQEN> to reset the interrupt. - Reprogramming the DMA Descriptor Address Registers and setting <RUN> restarts the channel. - If <STOPIRQEN> is set, the DMAC generates an interrupt.
2	ENDINTR	R/W1C	0x0	End Interrupt This field indicates that the current descriptor finished successfully and <ENDIRQEN> in the DMA Command Registers is set. This field indicates the successful completion of the current descriptor in a DMA operation 0: No interrupt 1: An nterrupt occured due to the successful completion of the current transaction, and <LEN> field in DMA Command Registers is set to 0
1	STARTINTR	R/W1	0x0	Start Interrupt

Offset: 0x0/0x4/0x8/0xC/0x10/0x14/0x18/0x1C/0x20/0x24/0x28/0x2C/0x30/0x34/0x38/0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
		C		This field indicates the successful loading of the current descriptor in a DMA operation 0: No interrupt 1: An interrupt occurred due to the successful descriptor fetching, and <STARTIRQEN> in the DMA Command Registers is set
0	BUSERRINTR	R/W1C	0x0	Bus Error Interrupt This field indicates an error during data transfer on the internal bus, potentially caused by an invalid descriptor source or target address (any address that is in the non-burstable or reserved space can cause a bus error on the system bus). Only one error per channel is logged, and the affected channel will not be updated until it is reprogrammed and the corresponding <RUN> field is set. 0 = No interrupt 1 = An Interrupt occurred due to a bus error

9.4.4.2 DALGN REGISTER

DMA Alignment Register. This register activates byte alignment for source and target addresses. Each bit in this register corresponds to a DMA channel. By default, during data transfers, the DMAC

- Forces the least-significant three bits of all external addresses to zero
- Forces the least-significant two bits of all peripheral addresses to zero

Setting a channel-specific bit in this register causes the corresponding channel to access the complete user-specified address (none of the LSB bits of the address will be forced to zeros). For example, if channel 15 is programmed to transfer data involving a misaligned address, software must write 1 to bit 15 of this register.

Clearing a bit position in this register will cause the DMAC to treat the corresponding channel as the default, a 64-bit aligned channel, the source and target addresses will be forced to zeros as explained earlier.

This register must be updated before setting the <RUN> field in the DMA Channel Control/Status Registers and then must not be altered until the channel stops.

Offset: 0xA0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	R	0	Reserved for future use
15:0	DALGNX	R/W	0x0	Alignment control for channel x 0: Source and target addresses of channel x follow the default alignment (internal peripherals default to 4 byte alignment, external bus addresses default to 8 byte alignment) 1: Source and target addresses of channel x follow user-defined alignment (byte aligned)

9.4.4.3 DPCSR REGISTER

DMA programmed I/O control status register.

Offset: 0xA4				
Bits	**Field**	**Type**	**Reset**	**Description**
31	BRGSPLIT	R/W	0x1	Activate posted writes and split reads. Don’t care
30:1	RSVD	R	0	Reserved for future use
0	BRGBUSY	R	0x0	Bridge busy status. Don‘t care

9.4.4.4 DRQSR REGISTER

DMA Request Status Register. This register tracks the number of pending requests made by an external companion chip on the corresponding DREQ pin. The register reflects the status of a 5-bit counter that is controlled by the DMAC in the following manner:

- The DMAC increments the counter each time the external companion chip toggles the DREQ pin from low to high (positive edge trigger).
- For a write to an external peripheral, the DMAC decreases the counter after it completes the write.
- For a read from an external peripheral, the DMAC decreases the counter after it sends the corresponding read request to the memory controller.
- The external companion chip must not exceed 31 pending requests at a given time.
- This is a read/write register. Ignore reads from reserved bits. Write 0x0 to reserved bits.

Offset: 0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	RSVD	R	0	Reserved for future use
8	CLR	W	0x0	Clearing pending request This field clears all pending requests registered in <REQPEND>, which were made by the external DMA request pin (DREQ). - Writing 0x1 to this field clears the <REQPEND> field to remove all pending requests. - Writing 0x0 to this field has no effect. Notes: - This field can be used for clearing the requests if the channel mapped to DREQ was prematurely stopped by software. - This field must be set only after the mapped channel has stopped (<STOPINTR> field in the DMA Channel Control/Status Registers is set). - Clearing the requests of a running channel can cause unpredictable behavior. 0 = No effect on <REQPEND> 1 = Clear all pending requests registered in <REQPEND>
7:5	RSVD	R	0	Reserved for future use

Offset: 0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
4:0	REQPEND	R	0x0	Request pending Indicates the number of pending requests on DREQ.

9.4.4.5 DINT REGISTER

DMA Interrupt Register. This read-only register tracks the interrupt information for each channel. An interrupt is generated if any of the following conditions occurs:

- Any transaction error occurs on the internal bus associated with the relevant channel.
- The current transfer finishes successfully and the <ENDIRQEN> field in the DMA Command Registers is set.
- The current descriptor is loaded successfully and the <STARTIRQEN> field in the DMA Command Registers 0-15 is set.
- The <STOPIRQEN> field in the DMA Channel Control/Status Registers is set and the channel is in an uninitialized or stopped state.
- The <EORIRQEN> and <EORINT> (EOR signaled by a peripheral) fields in the DMA Channel Control/Status Registers are set.
- The <RASINTR> field in the DMA Channel Control/Status Registers is set and the peripheral makes a DMA request after the channel has stopped.

All DMAC interrupts, except the one that corresponds to the <STOPINTR> field in the DMA Channel Control/Status Registers, are cleared by writing 1 to the corresponding interrupt bit in the DMA Channel Control/Status Registers.

Offset: 0xF0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	R	0	Reserved for future use
15:0	CHLINTRX	R	0x0	Channel interrupt This field indicates that DMA channel x has been interrupted. 0: No interrupt 1: Interrupt

9.4.4.6 DRCMR_x REGISTER

DMA request to channel mapping registers. These registers map the DMA request to a channel.

Offset: 0x100/0x104/0x108/0x10C/0x110/0x114/0x118/0x11C/0x120/0x124/0x128/0x12C/ 0x130/0x134/0x138/0x13C/0x140/0x144/0x148/0x14C/0x150/0x154/0x158/0x15C/ 0x160/0x164/0x168/0x16C/0x170/0x174/0x178/0x17C/0x180/0x184/0x188/0x18C/ 0x190/0x194/0x198/0x19C/0x1A0/0x1A4/0x1A8/0x1AC/0x1B0/0x1B4/0x1B8/0x1BC/ 0x1C0/0x1C4/0x1C8/0x1CC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	R	0	Reserved for future use
7	MAPVLD	R/W	0x0	Map valid channel Defines whether the request is mapped to a valid channel. 0: Request is unmapped 1: Request is mapped to a valid channel indicated by <Channel number> This bit can also be used to mask the request.
6:5	RSVD	R	0	Reserved for future use
4:0	CHLNUM	R/W	0x0	Channel number Indicates the valid channel number if <Map valid channel> is set. Note: Do not map two active requests to the same channel since it produces unpredictable results

9.4.4.7 DDADR_L_x REGISTER

DMA Descriptor Address Registers. These registers store the memory address of the next descriptor for a given DMA channel, in particular:

- The fields in this register (except <STOP>) are undefined on power up
- <STOP> is cleared on power up
- The address must be aligned to either a 128-bit (4-word) boundary or 256-bit (8-word) boundary that depends on <LPAE_EN>

Note. These registers must not contain the address of any other internal peripheral register or DMA register as this causes a bus error.

These registers are reserved if the channel is performing a no-descriptor-fetch transaction.

Offset: 0x200/0x210/0x220/0x230/0x240/0x250/0x260/0x270/0x280/0x290/0x2A0/0x2B0/0x2C0/0x2D0/0x2E0/0x2F0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	DDADR_L	R/W	0x0	Descriptor address It contains the address of the next descriptor.
3:2	RSVD	R	0	Reserved for future use
1	BREN	R/W	0x0	Enable Descriptor Branch This field controls descriptor branching and works with the <Descriptor

Offset: 0x200/0x210/0x220/0x230/0x240/0x250/0x260/0x270/0x280/0x290/0x2A0/0x2B0/0x2C0/0x2D0/0x2E0/0x2F0				
Bits	**Field**	**Type**	**Reset**	**Description**
				compare status> field in the DMA Channel Control/Status Registers (0-31) to determine which descriptor is fetched next. - If both this field and <Descriptor compare status> are set, the DMAC fetches the next descriptor from (DDADRx + 32 bytes). - If either of the bits is cleared, DMAC fetches the next descriptor from the DMA Descriptor Address Registers. - This field is relevant only for descriptor-fetch transactions (when <No-Descriptor Fetch> field in the DMA Channel Control/Status Registers 0-31 = 0). 0: Disable descriptor branching. Fetch the next descriptor from DDADRx. 1: Enable descriptor branching. Fetch the next descriptor from DDADRx + 32 bytes
0	STOP	R/W	0x0	Stop It controls whether the channel stops after processing the current descriptor 0: Continue running. 1: Stop after completing the current descriptor (when <Length of the transfer in bytes> field in DMA Command Registers 0-31 = 0).

9.4.4.8 DSADR_L_x REGISTER

DMA source address registers. These registers are read-only for descriptor-fetch transactions and read/write for no-descriptor-fetch transactions. These registers store the source address of the current descriptor for a channel. The source address can refer to:

- An on-chip peripheral
- An external peripheral
- A companion chip
- A memory location

Note. These registers can not contain addresses of any other internal DMA registers, as this will cause a bus error.

If the source address refers to a memory location and the Alignment register is properly configured, it can be aligned to a byte boundary (refer to **Section 9.4.4.2** DMA Alignment Register for more details). Otherwise, if the Alignment register is not configured correctly, the source address defaults to an 8-byte boundary.

Offset: 0x204/0x214/0x224/0x234/0x244/0x254/0x264/0x274/0x284/0x294/0x2A4/0x2B4/0x2C4/0x2D4/0x2E4/0x2F4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	SRCADDR	R/W	0x0	Source address of the on-chip peripheral, external peripheral, companion chip, or address of a memory location

Offset: 0x204/0x214/0x224/0x234/0x244/0x254/0x264/0x274/0x284/0x294/0x2A4/0x2B4/0x2C4/0x2D4/0x2E4/0x2F4				
Bits	**Field**	**Type**	**Reset**	**Description**
2	SRCADDR2	R/W	0x0	Relevant if <Source address> is a memory location and alignment register is configured. Refer to **Section 9.4.4.2** for DMA Alignment Register for programming details and restrictions.
1:0	SRCADDR0	R/W	0x0	Relevant if <Source address> is a memory location and alignment register is configured. Refer to **Section 9.4.4.2** for DMA Alignment Register for programming details and restrictions.

9.4.4.9 DTADR_L_x REGISTER

DMA Target Address registers. These registers are read-only for descriptor-fetch transfers and read/write for no-descriptor-fetch transfers. These registers store the source address of the current descriptor for a channel. The source address can refer to:

- An on-chip peripheral
- An external peripheral
- A companion chip
- A memory location

Note. These registers can not contain addresses of any other internal DMA registers, as this will cause a bus error.

If the source address refers to a memory location and the Alignment register is properly configured, it can be aligned to a byte boundary (refer to **Section 9.4.4.2** for DMA Alignment Register for more details). Otherwise, if the Alignment register is not configured correctly, the source address defaults to an 8-byte boundary.

Offset: 0x208/0x218/0x228/0x238/0x248/0x258/0x268/0x278/0x288/0x298/0x2A8/0x2B8/0x2C8/0x2D8/0x2E8/0x2F8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	TRGADDR	R/W	0x0	Target address of the on-chip peripheral, external peripheral, companion chip, or address of a memory location
2	TRGADDR2	R/W	0x0	Relevant if <Target address> is a memory location and alignment register is configured. Refer to **Section 9.4.4.2** for DMA Alignment Register for programming details and restrictions.
1:0	TRGADDR0	R/W	0x0	Relevant if <Target address> is a memory location and alignment register is configured. Refer to **Section 9.4.4.2** for DMA Alignment Register for programming details and restrictions.

9.4.4.10 DCMD_x REGISTER

DMA Command registers. These read-only registers are for descriptor-fetch transfers and read/write for no-descriptor-fetch transfers.

Offset: 0x20C/0x21C/0x22C/0x23C/0x24C/0x25C/0x26C/0x27C/0x28C/0x29C/0x2AC/0x2BC/0x2CC/0x2DC/0x2EC/0x2FC

Bits	Field	Type	Reset	Description
31	INCSRCADDR	R/W	0x0	Source address increment Controls whether the source address increments on each successive access. 0: No increment (use for internal peripheral FIFO or external I/O addresses). 1: Increment source address.
30	INCTRGADDR	R/W	0x0	Target address increment Controls whether the target address increments on each successive access. 0: No increment (use for internal peripheral FIFO or external I/O addresses). 1: Increment target address.
29	FLOWSRC	R/W	0x0	Source flow control Use when the source is an on-chip peripheral or external companion chip. 0: Start data transfer without waiting for request signals 1: Wait for a request signal before initiating the data transfer **Note.** Do not set this field if <FLOWSRC> is already set , as it may cause unpredictable behavior.
28	FLOWTRG	R/W	0x0	Target flow control Use when the target is an on-chip peripheral or external companion chip. 0: Start data transfer without waiting for request signals 1: Wait for a request signal before initiating the data transfer **Note.** Do not set this field if <FLOWTAG> is already set, as it may cause unpredictable behavior.
27:26	RSVD	R	0	Reserved for future use
25	CMPEN	R/W	0x0	Descriptor Compare enable This field must be cleared for normal DMA operations. Setting the field enables the descriptor-compare mode, in which the DMAC treats the current descriptor as a special case and compares data that corresponds to the source and target fields. <ADDRMODE> is used to determine the addressing mode before the Compare operation. 0: DMA does not perform any address-compare operations 1: DMA recognizes the current descriptor as a special case and compares data based on the source address and target address fields.

Offset: 0x20C/0x21C/0x22C/0x23C/0x24C/0x25C/0x26C/0x27C/0x28C/0x29C/0x2AC/0x2BC/0x2CC/0x2DC/0x2EC/0x2FC

Bits	Field	Type	Reset	Description
				- If the compare is true, the channel's <CMPST> field in the DMA Channel Control/Status Registers is set. - If the compare is false, <CMPST> is cleared.
24	RSVD	R	0	Reserved for future use
23	ADDRMODE	R/W	0x0	Addressing mode This field controls the addressing mode for descriptor comparison and is valid only in the descriptor compare mode (<CMPEN> = 1). - Reserved if <CMPEN> = 0. - If <CMPEN> is set, this bit specifies the addressing modes of the source address and target address fields. - If either field contains an address, the DMAC fetches the data at that address and uses it for the compare operation. 0: Source address field contains address, and target address field contains address 1: Source address field contains address, and target address field contains data - If DALGN is clear, then the lowest three bits of immediate data are forced to be 0 before comparison. - If DALGN is set, then the lowest three bits of immediate data are not forced to be 0 before comparison.
22	STARTIRQEN	R/W	0x0	Start interrupt enable This field indicates that the interrupt is enabled when the descriptor is loaded. In no-descriptor-fetch transfers, this field is reserved. 0: Interrupt is not triggered after descriptor is loaded 1: Sets interrupt bit for the channel in the <CHLINTRX> field in the DMA Interrupt Register when the descriptor for the channel is loaded
21	ENDIRQEN	R/W	0x0	End interrupt enable 0: Interrupt is not triggered when LENGTH decrements to zero. 1: Sets the DINT interrupt bit for the channel when LENGTH decrements to zero.
20:19	RSVD	R	0	Reserved for future use
18:16	DMA_SIZE	R/W	0x0	Maximum burst size Maximum burst size of each data transfer - 0x0 = Reserved - 0x1 = 8 bytes - 0x2 = 16 bytes - 0x3 = 32 bytes - 0x4 = 64 bytes The size must be less than or equal to the serviced peripheral FIFO trigger threshold to properly handle the respective FIFO trailing bytes.
15:14	WIDTH	R/W	0x0	Width of the on-chip peripheral

Offset: 0x20C/0x21C/0x22C/0x23C/0x24C/0x25C/0x26C/0x27C/0x28C/0x29C/0x2AC/0x2BC/0x2CC/0x2DC/0x2EC/0x2FC				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field is reserved for operations that do not involve on-chip peripherals, such as memory-to-memory moves and companion-chip-related operations. 0x0 = Reserved for on-chip peripheral-related transactions 0x1 = 1 byte 0x2 = half-word (2 bytes) 0x3 = word (4 bytes) Note: For memory-to-memory moves or companion-chip-related operations, this field must be set to 0x0.
13	RSVD	R	0	Reserved for future use
12:0	LEN	R/W	0x0	Length of the transfer in bytes This field is the length of transfer in bytes. - **LEN = 0**: 1. In descriptor-fetch mode**,** LEN = 0 signifies no byte transfer. when <CMPEN> is clear (normal data transfer mode) causes the channel to immediately discard the descriptor after it is fetched from memory. If the descriptor chain has more descriptors, the channel fetches the next valid descriptor. The channel stops if the descriptor chain has no more descriptors. 2. In no-descriptor-fetch mode, LEN = 0 is an invalid setting. - **Maximum Transfer Length**: 1. The maximum transfer length is (8K-1) bytes. 2. If the transfer is of the memory-to-memory type, the length of the transfer may be any value (except for the LEN = 0 restriction in no-descriptor-fetch mode) up to a maximum of (8K -1) bytes. - **For memory-to-memory:** 1. The length of the transfer may be any value (except for the LEN = 0 restriction in no-descriptor-fetch mode) up to a maximum of (8K -1) bytes. - **For Peripherals**: 1. The length of the transfer must be an integer multiple of the peripheral FIFO threshold (or water-mark).

9.4.4.11 DDADR_H_x REGISTER

DMA descriptor address higher bits registers. These registers contain the higher 8 bits of memory address of the next descriptor for a channel. The address must be aligned to a 128-bit (4-word) boundary when <LPAE_EN> is clear, 256-bit (8-word) boundary when <LPAE_EN> is set.

Note. These registers can not contain addresses of any other internal peripheral register or DMA register, as this will cause a bus error.

Offset: 0x300/0x310/0x320/0x330/0x340/0x350/0x360/0x370/0x380/0x390/0x3A0/0x3B0/0x3C0/0x3D0/0x3E0/0x3F0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	R	0	Reserved for future use
7:0	DDADR_H	R/W	0x0	Descriptor address higher bits [39:32]. Contains address of next descriptor higher bits [39:32]

9.4.4.12 DSADR_H_x REGISTER

DMA source address higher bits registers. These registers are read-only for descriptor-fetch transactions and read/write for no-descriptor-fetch transactions. These registers store the source address higher bits [39:32] of the current descriptor for a channel. The source address can refer to:

- An on-chip peripheral
- An external peripheral
- A companion chip
- A memory location

Note. These registers can not contain addresses of any other internal peripheral register or DMA register, as this will cause a bus error.

Offset: 0x304/0x314/0x324/0x334/0x344/0x354/0x364/0x374/0x384/0x394/0x3A4/0x3B4/0x3C4/0x3D4/0x3E4/0x3F4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	R	0	Reserved for future use
7:0	SOURCE_ADD_RESS_H	R/W	0x0	Source address higher bits[39:32] Contains address of source higher bits[39:32].

9.4.4.13 DTADR_H_x REGISTER

DMA target address higher bits registers. These registers are read-only for descriptor-fetch transactions and read/write for no-descriptor-fetch transactions. These registers store the source address higher bits [39:32] of the current descriptor for a channel. The source address can refer to:

- An on-chip peripheral
- An external peripheral
- A companion chip
- A memory location

Note. These registers can not contain addresses of any other internal peripheral register or DMA register, as this will cause a bus error.

Offset: 0x308/0x318/0x328/0x338/0x348/0x358/0x368/0x378/0x388/0x398/0x3A8/0x3B8/0x3C8/0x3D8/0x3E8/0x3F8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	R	0	Reserved for future use
7:0	TARGET_ADD_RESS_H	R/W	0x0	Target address higher bits[39:32] Contains address of target higher bits[39:32].

9.5 Crypto

9.5.1 Introduction

The crypto subsystem consists of True Random Number Generator (TRNG) and hardware accelerating engines for multiple mainstream cryptography algorithms.

9.5.2 Features

- TRNG for various kinds of security applications
- Symmetric encryption algorithms including AES
- Public key algorithms including RSA/ECC
- HASH algorithms including SHA2

9.5.3 Functional Description

The crypto module provides high-quality true random number generation and supports multiple mainstream cryptography algorithms. It may be utilized in a broad range of security applications.

9.5.4 Register Description

The base address of TRNG registers is **0xF0703800**.

9.5.4.1 RNG BYTE COUNT REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved
15:0	RNG Byte Count	RW	0x0	Bytes of data that the DMA channel must transfer.

9.5.4.2 RNG SOURCE ADDRESS REGISTER

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RNG sourceaddress	RW	0x0	Not used. All data is from the RNG generator engine inside the RNG module.

9.5.4.3 RNG DESTINATION ADDRESS REGISTER

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RNG destination address	RW	0x0	RNG data output destination address.

9.5.4.4 RNG NEXT DESCRIPTOR POINTER REGISTER

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RNG next descriptor pointer address	RW	0x0	Channel 0 next descriptor pointer address. - This must be 16 bytes aligned. - The register contains a 32-bit address where the value for the next DMA channel descriptor can be loaded for chain operation. - The next descriptor can be stored in either SDRAM or SRAM. - The byte count, source address, destination address, and next descriptor pointer must be located at sequential addresses. - This register is only used when the channel is configured for Chain mode.

9.5.4.5 RNG CONTROL REGISTER

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Reserved	RO	0x0	Reserved
30	RNG fifo clear	WO	0x0	Write to clear RNG RX FIFO
29:22	RNG fifo threshold	RW	0xf	The threshold of FIFO for data transmit request

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
21:17	Reserved	RO	0x0	Reserved
16	Channel abort	RW	0x0	- Setting this bit to 1 forces the DMA to abort the current transfer immediately. - This bit is automatically cleared by the DMA hardware after the abort operation is completed.
15	Close descriptor enable	RW	0x0	If enabled, the DMA writes the remainder byte count into bits [31:16] of the byte count field. 0: Disable 1: Enable
14	Source/Destination address alignment	RW	0x0	0: Alignment towards the source. After the DMA's first read, all reads will be to 128-bit aligned address. 1: Alignment towards the destination. After the first write, the following writes will be with all Byte Enables asserted.
13	DMA channel active	RO	0x0	0: Channel is not active 1: Channel is active
12	Fetch next descriptor	RW	0x0	1: Force a fetch of the next descriptor. This field is automatically cleared after the fetch completes.
11	TransMod	RW	0x0	0: External DMA_REQ access mode
10	Interrupt mode	RW	0x0	0: Interrupt asserted every time the DMA byte count reaches 0 1: Interrupt asserted only when the next descriptor pointer value is NULL and the DMA byte count reaches 0
9	Chain mode	RW	0x0	0: Chain mode 1: Non-chain mode
8:6	Burst Limit in each DMA access	RW	0x0	- 0x0: 1 byte - 0x1: 2 byte - 0x2: 4 byte - 0x3: 8 byte - 0x4: 16 byte - 0x5: 32 byte - 0x6: 64 byte - 0x7: 128 byte
5:4	Destination direction	RW	0x0	- 0x0: Increment destination address - 0x1: Decrement destination address - 0x2: Hold in the same value - 0x3: Reserved
3:2	Source direction	RW	0x0	- 0x0: Increment source address - 0x1: Decrement source address - 0x2: Hold in the same value - 0x3: Reserved
1	SSPMod	RW	0x0	1: SSP FIFO access

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
0	Channel Enable	RW	0x0	1: Activates the channel. This bit is automatically cleared when the DMA transfer is complete. 0: Suspends the channel. **Note.** If the channel is suspended, setting this bit to 1 again allows the DMA transfer to continue.

9.5.4.6 RNG CURRENT DESCRIPTOR POINTER REGISTER

Offset: 0x74				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RNG current descriptor pointer address	RO	0x0	This register is used for closing the current descriptor before fetching the next descriptor. - When the processor writes the next descriptor pointer register, the current descriptor pointer register will reload itself with the same value written to the next descriptor pointer register. - After processing a descriptor, the DMA channel: 1. updates the current descriptor using the current descriptor pointer register. 2. saves the next descriptor pointer register into the current descriptor pointer register. 3. and fetches a new descriptor for processing.

9.5.4.7 RNG INTERRUPT MASK REGISTER

Offset: 0x84				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved
1	RNG DMA aborted interrupt mask	RW	0x0	1: Channel 1 DMA aborted interrupt is enabled
0	RNG interrupt done mask	RW	0x0	1: Channel 1 Comp interrupt is enabled

9.5.4.8 RNG RESET SELECT REGISTER

Offset: 0x94				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved
1:0	RNG reset select address	RW	0x0	The bits in this register correspond to the bits in the RNT_INT_STATUS register. Setting a bit to 1 enables its corresponding RNT_INT_STATUS bit read clear function.

9.5.4.9 RNG INTERRUPT STATUS REGISTER

Offset: 0xA4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved
1	DMA abort interrupt status and clear	RW	0x0	Read clear in this field is enabled through the corresponding RSR bit setting in the RNG_RSR register.
0	Interrupt done	RW	0x0	Read clear in this field is enabled through the corresponding RSR bit setting in the RNG_RSR register.

9.5.4.10 PRNG CTRL REGISTER

Offset: 0xC0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PRNG valid	RO	0x0	1'b1: PRNG is valid, and PRNG data can be read from PRNG result low and high registers.
30:9	Reserved	RO	0x0	Reserved
8:1	PRNG Register CTRL	RW	0x1A	Please keep the default value for current design.
0	PRNG enable	RW	0x0	Enable PRNG

9.5.4.11 PRNG LOW REGISTER

Offset: 0xC4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	PRNG data low bits	RO	0x0	PRNG result data bit [31:0].

9.5.4.12 PRNG HIGH REGISTER

Offset: 0xC8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved
7:0	PRNG data high bits	RO	0x0	PRNG result data bit [39:32].

9.5.4.13 TRNG CTRL REGISTER

Offset: 0xD0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	Reserved	RO	0x0	Reserved
19	TRNG data search done	RO	0x0	TRNG data search done.
18:16	TRNG data search value	RO	0x0	TRNG data search value.
15:8	Reserved	RO	0x0	Reserved
7	TRNG_REG_LDO_CAL_START	RW	0x0	TRNG LDO calibration start
6	TRNG_REG_LDO_PU	RW	0x0	LDO PU on
5	TRNG_REG_LDO_RECAL	RW	0x0	LDO re-calibration
4	TRNG_REG_SEL	RW	0x0	0: Use PRNG (Default) 1: Use TRNG
3:2	TRNG_REG_MODE	RW	0x0	- 0: 4 bit mode - 1: 2 bit mode - 2: 1 bit mode - 3: debug mode
1	TRNG CLK ON	RW	0x0	TRNG 24M reference clock turn on
0	TRNG enable	RW	0x0	Enable TRNG

9.5.4.14 TRNG REG REGISTER

Offset: 0xD4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0x0	Reserved
23:16	TRNG_REG_REG3	RW	0xD9	TRNG_REG_REG3
15:8	TRNG_REG_REG2	RW	0x81	TRNG_REG_REG2

Offset: 0xD4				
Bits	**Field**	**Type**	**Reset**	**Description**
7:0	TRNG_REG_REG1	RW	0x81	TRNG_REG_REG1

9.5.4.15 TRNG DATA REGISTER

Offset: 0xD8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TRNG data register value	RO	0x0	TRNG result data bit [31:0]

9.6 Timer & Watchdog

9.6.1 Introduction

The K1 includes:

- Three 32-bit general-purpose timers with programmable clock frequency
- One 16-bit Watchdog timer with programmable clock frequency

9.6.2 Features

[General-Purpose Timers]

- Each one consists of a 32-bit Timer Clock Control Register (TCCRn) and operates as an up counter
- Programmable clock frequency with two clock inputs as follows:
  - [For fast clock] 12.8 MHz, 6.4 MHz, 3 MHz, 1 MHz
  - [For slow clock] 32.768 kHz

- **[Watchdog Timer]**
  - Operative as an up counter
  - Programmable clock frequency with two clock inputs as follows:
    - [For fast clock] 12.8 MHz, 6.4 MHz, 3 MHz, 1 MHz
    - [For slow clock] 32.768 kHz

9.6.3 Functional Description

9.6.3.1 Timer Unit

- **Fast Clock Selection**
  The fast clock is selectable via the Functional Clock Select field in the Clock/Reset Control Register for Timers (APBC_TIMERS_CLK_RST). All three timers (Timer 0/1/2) can operate using either the fast or slow clock

- **Timer Count Registers (TCRn)**
  Each of the three Timer Count Registers (TCRn, where n = 0, 1, or 2) is associated with three 32-bit Timer Match Registers (TMR_Tn_Mm). When the value in the Operating System Count Register (TCRn) matches a value in any of the match registers, and the interrupt-enable bit is set, the corresponding bit in the Timer Status Register (TSR) is set.

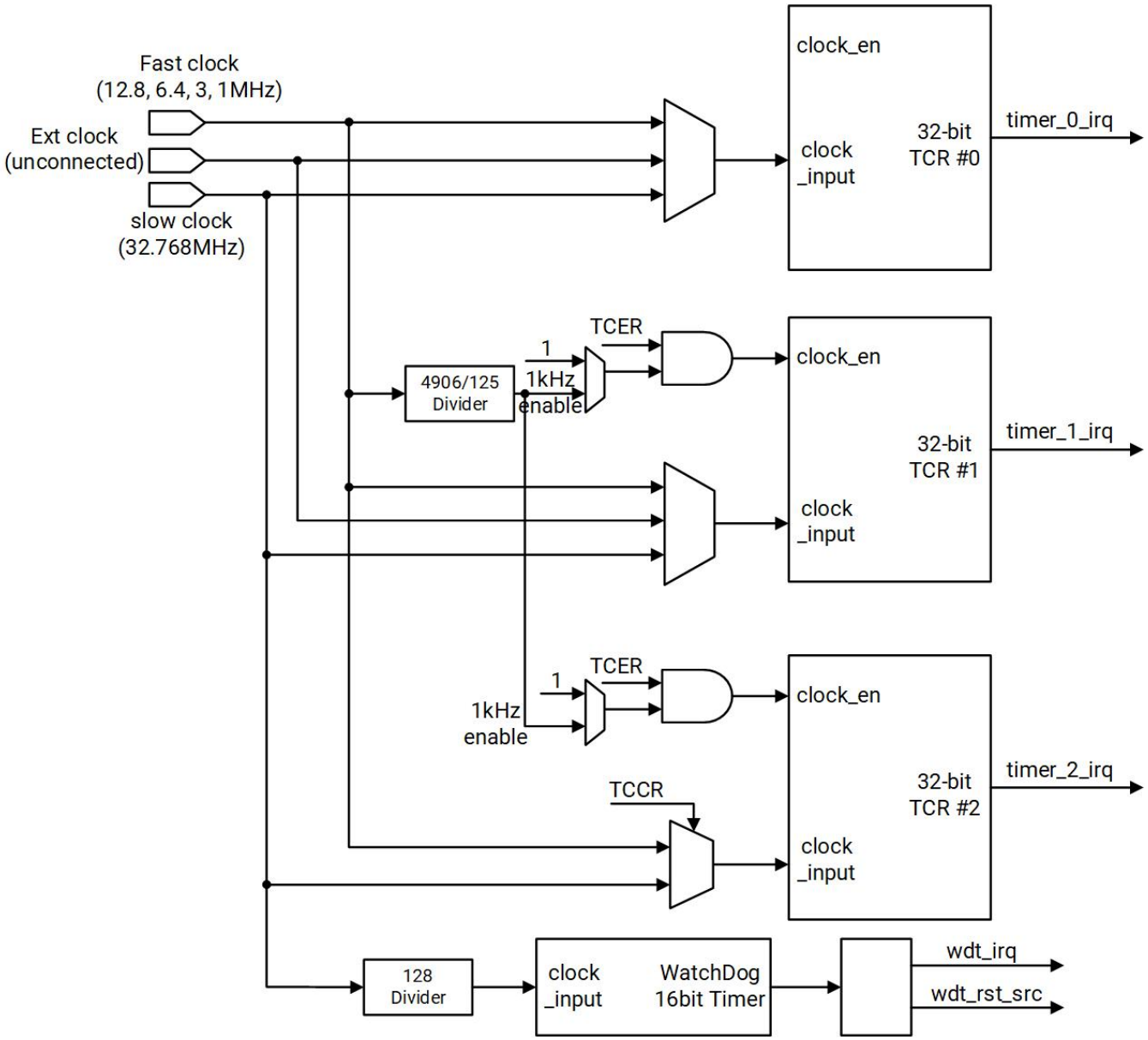
- **Interrupt Generation**
  These bits in the TSR are routed to the Interrupt Controller, which can be programmed to generate an interrupt when triggered.

- **Reprogramming Timers**
  Reprogramming any of the timer control registers (TCCR, TPLVRn, TMR_Tn_Mm, TPLCRn, TCMRn, or TILRn) while the timer is operating is not guaranteed to be valid. To safely reprogram these registers,

  - Disable the timer
  - Re-program the timer
  - Re-enable the timer

  The functionality of one enabled timer is not affected by the programming of another timer.

The architecture of the timer unit is depicted below.



9.6.3.2 Watchdog Timer

The 16-bit Watchdog timer operates using a timer module-derived clock with a frequency of 256 Hz.

The Watchdog timer initiates a reset event when its value matches the value in the TWMR and if the TWMER[WE] bit is set. This causes the wdt_rst_src# signal to be asserted, initiating a Watchdog Timer Reset event in the system.

- Watchdog Timer Reset Mode: This mode is activated when software does not properly manage the WDT timeout, indicating potential software malfunction or data corruption. In this mode, most internal system registers are reset to their default values, with the exception of the real-time clock, which remains unaffected.
- To avoid a Watchdog Timer Reset, the software must restart the WDT before it reaches the match value, by setting the TWCR[WCR] field to restart the WDT.

Note. The TWSR Register (and some other internal WDT flops), which determines whether a match event has occurred, is reset upon the assertion of power-on-reset or external-master reset. All other registers are reset upon the assertion of power-on-reset, external-master reset, or watchdog-timer reset.

When a WDT reset event occurs, the WDT generates a 4 msec-wide pulse on the wdt_rst_src# output. This keeps the system in a reset state during that period. The system goes through a reset sequence once the reset signal is de-asserted.

Writing to all WDT registers is protected by 2 access registers: TWFAR and TWSAR. Follow the steps below to enable Writes to any WDT register:

- Write the proper key value to the TWFAR register
- Write the proper key value to the TWSAR register
- Perform the write operation on the preferred WDT register.

Once this write operation is completed, the WDT registers are locked again. This process must be repeated for each subsequent write operation.

9.6.4 Register Description

The base addresses of timer/watchdog registers are tabled below.

Name	Address
APB_TIMERS1_BASE	0xD4014000
APB_TIMERS2_BASE	0xD4016000
PMU_TIMERS_BASE	0xD4080000
SEC_TIMERS_BASE	0xF0616000

9.6.4.1 Timer Count Enable Register

This register contains a count enable bit for each timer. After being enabled, the corresponding TCR restarts the count from the value prescribed by the TPLVR.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use.
2	Timer #2 count enable	RW	0x0	Controls whether Timer #2 counts: 0: Counting is disabled. 1: Counting is enabled. **Note.** Changes do not take effect immediately due to synchronization across clock domains.
1	Timer #1 count enable	RW	0x0	Controls whether Timer #1 counts: 0: Counting is disabled.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				1: Counting is enabled. **Note.** Changes do not take effect immediately due to synchronization across clock domains.
0	Timer #0 count enable	RW	0x0	Controls whether Timer #0 counts: 0: Counting is disabled. 1: Counting is enabled. **Note**. Changes do not take effect immediately due to synchronization across clock domains.

9.6.4.2 Timer Count Mode Register

The TCMR contains a count mode bit for each timer. The processor TCR operates only in periodic timer mode, it does not operate in one-shot mode.

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use.
2	Timer #2 count mode	RW	0x0	Defines the mode of Timer #2: 0: Periodic timer mode: Timer re-loads if a match occurs and PLCR != 0 1: Free-run mode: Timer wraps to 0 when it reaches 0xFFFFFFFF.
1	Timer #1 count mode	RW	0x0	Defines the mode of Timer #1: 0: Periodic timer mode: Timer re-loads if a match occurs and PLCR != 0 1: Free-run mode: Timer wraps to 0 when it reaches 0xFFFFFFFF.
0	Timer #0 count mode	RW	0x0	Defines the mode of Timer #0: 0: Periodic timer mode: Timer re-loads if a match occurs and PLCR != 0 1: Free-run mode: Timer wraps to 0 when it reaches 0xFFFFFFFF.

9.6.4.3 Timer Count Restart Register

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use.
2	T2RS	RW	0x0	Timer #2 count restart 0: No effect 1: Counting is restarted **Note.** Configure other registers before setting this bit to 1.
1	T1RS	RW	0x0	Timer #1 count restart 0: No effect 1: Counting is restarted **Note.** Configure other registers before setting this bit to 1.
0	T0RS	RW	0x0	Timer #0 count restart 0: No effect 1: Counting is restarted **Note.** Configure other registers before setting this bit to 1

9.6.4.4 Timer Clock Control Register

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	Reserved	RO	0x0	Reserved for future use.
6:5	CS_2	RW	0x0	Clock Source for Timer #2. - 0x0: Fast clock (AP APB Timer clock depending on APBC_TIMERSx_CLK_RST[6:4], CP APB Timer fast can only be 12.8M) - 0x1: 32.768 kHz - 0x2: 32.768 kHz - 0x3: fast clock
4	Reserved	RO	0x0	Reserved for future use.
3:2	CS_1	RW	0x0	Clock Source for Timer #1 - 0x0: Fast clock (AP APB Timer clock depending on APBC_TIMERSx_CLK_RST[6:4], CP APB Timer fast can only be 12.8M) - 0x1: 32.768 kHz - 0x2: 32.768 kHz - 0x3: Reserved
1:0	CS_0	RW	0x0	Clock Source for Timer #0

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 0x0: fast clock (AP APB Timer clock depending on APBC_TIMERSx_CLK_RST[6:4], CP APB Timer fast can only be 12.8M) - 0x1: 32.768 kHz - 0x2: Reserved - 0x3: Fast clock

9.6.4.5 Timer Match Register

Offset: 0x10~0x18(0x4)/0x20~0x28(0x4)/0x30~0x38(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TMR_TN_MM	RW	0xFFFFFFFF	Timer n Match register m value. This register holds the value used for the match comparison for Timer n (where n is the timer number, e.g., Timer 0, Timer 1, or Timer 2). When the timer counter reaches this value, a match event occurs.

9.6.4.6 Timer Preload Value Register

Each TCR has a 32-bit-wide Preload Value register that loads the TCRn when a match occurs between TMR_Tn_Mm and TCRn. The corresponding TPLCRn register selects the match comparator.

Offset: 0x40~0x48(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TPLVRn	RW	0x0	Timer n preload value that is loaded into TCRn when a match occurs between TMR_Tn_Tm and TCRn. The corresponding TPLCRn register selects the match comparator.

9.6.4.7 Timer Preload Control Register

Each TCR has a Preload Control register.

Offset: 0x50~0x58(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use
2	CRPD	RW	0x0	Disable preload when counter restar - 0x0: Preload the PLCR to counter when the restart bit is set;

Offset: 0x50~0x58(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
				- 0x1: Disable preload of the PLCR to counter when the restart bit is set.
1:0	MCS	RW	0x0	Match comparator selection: - 0x0: Free running mode (up to max value) - 0x1: Enable preload with match comparator 0 - 0x2: Enable preload with match comparator 1 - 0x3: Enable preload with match comparator 2

9.6.4.8 Timer Interrupt Enable Register

Each of these three counter registers contain one enable bit, which determines whether a match between a Match register & the operating-system timer counter will set a status bit in the Timer Status Register (TSR) and assert the corresponding timer#_irq output.

Note. Clearing an enable bit does not reset the corresponding interrupt status bit if it has already been set.

Offset: 0x60~0x68(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use.
2	IE2	RW	0x0	Interrupt enable for match comparator 2. 0: Disable interrupt for match between Match register 2 and its OS timer (no interrupt assertion). 1: Enable interrupt for match between Match register 2 and its OS timer (interrupt assertion in TSRn or timer#_irq output).
1	IE1	RW	0x0	Interrupt enable for match comparator 1. 0: Disable interrupt for match between Match register 1 and its OS timer (no interrupt assertion). 1: Enable interrupt for match between Match register 1 and its OS timer (interrupt assertion in TSRn or timer#_irq output).
0	IE0	RW	0x0	Interrupt enable for match comparator 0. 0: Disable interrupt for match between Match register 0 and its OS timer (no interrupt assertion). 1: Enable interrupt for match between Match register 0 and its OS timer (interrupt assertion in TSRn or timer#_irq output).

9.6.4.9 Timer Interrupt Clear Register

These three registers contain a separate clear bit for each interrupt source, which is used to reset the level-sensitive interrupt request directed to the interrupt controller. Each match register has its own corresponding clear bit. The interrupt is cleared by writing to the respective bit position.

Note. This register is not applicable for edge-sensitive interrupts.

Offset: 0x70~0x78(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use.
2	TCLR2	WO	0x0	Interrupt clear for match comparator 2 0: No affect 1: Clear the level interrupt and corresponding status bit
1	TCLR1	WO	0x0	Interrupt clear for match comparator 1 0: No affect 1: Clear the level interrupt and corresponding status bit
0	TCLR0	WO	0x0	Interrupt clear for match comparator 0 0: No affect 1: Clear the level interrupt and corresponding status bit

9.6.4.10 Timer Status Register

These three Status registers contain status bits indicating whether a match has occurred on any of the three Match registers of a given Timer Count register, in particular:

- These bits are set when a match event occurs on the next rising edge of the respective clock
- They are cleared by writing a logical one to the corresponding bit position of TICLRn

This register reflects level-sensitive interrupt status only, edge-sensitive interrupts are not captured in this register.

Offset: 0x80~0x88(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0x0	Reserved for future use.
2	M2	RO	0x0	Match status of TMR_Tn_M2. 0: Timer Match register TMR_Tn_M2 has not matched the counter since the last interrupt clear. 1: Timer Match register TMR_Tn_M2 has matched the counter since the last interrupt clear
1	M1	RO	0x0	Match status of TMR_Tn_M1. 0: Timer Match register TMR_Tn_M1 has not matched the counter since the last interrupt clear. 1: Timer Match register TMR_Tn_M1 has matched the counter since the last interrupt clear
0	M0	RO	0x0	Match status of TMR_Tn_M0. 0: Timer Match register TMR_Tn_M0 has not matched the counter since the last interrupt clear. 1: Timer Match register TMR_Tn_M0 has matched the counter since the last interrupt clear.

9.6.4.11 Timer Count Register

Three read-only Timer Count registers (TCRn, where n = 0, 1, 2) are 32-bit counters that increment on the rising edge of their selected clocks, in particular:

- The Timer Count registers have been synchronized from timer clock domain to APB clock domain. Software can directly read these registers to use.
- The counters are pre-loaded with a value from the TPLVR register. When enabled, counters start from pre-loaded values (defined in the corresponding TPLCRn register), and count up to either a maximum or matched value.
- This request requires up to three timer clock cycles. If the selected timer is working at a slow clock, the request could take longer.

Offset: 0x90~0x98(0x4)				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TCRN	RO	0x0	Timer n count register. - The counter is incremented on the rising edge of the selected clock. - These registers have been synchronized to APB clock domain.

9.6.4.12 Timer Watchdog First Access Register

Offset: 0xB0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	KEY	WO	0x0	Watchdog access key. Writing the value of 0xBABA to this register matches the key.

9.6.4.13 Timer Watchdog Second Access Register

Offset: 0xB4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	KEY	WO	0x0	Watchdog access key. Writing the value of 0xEB10 to this register matches the key.

9.6.4.14 Timer Watchdog Match Enable Register

The Watchdog Enable register contains a WDT enable bit that can only be set by the user. The write access to this register is protected by the TWFAR and TWSAR Access Registers.

Offset: 0xB8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved for future use.
1	WRIE	RW	0x0	Watchdog reset/interrupt enable. 0: Watchdog timer expiration generates a watchdog interrupt (no watchdog timer reset) 1: Watchdog timer expiration generates a watchdog timer reset, (no watchdog interrupt)
0	WE	RW	0x0	WDT count enable. 0 = Disable WDT count, reset WDT's value to zero. 1 = Enable counting, the WDT always starts from zero. **Note.** Due to the chain of synchronizers that transform this signal from domain to domain, the WDT timer enable and disable operation do not occur immediately.

9.6.4.15 Timer Watchdog Match Register

This Match register is compared to the watchdog timer. The watchdog timer resets the processor when a match occurs and the TWER[WRIE] bit is set.

Offset: 0xBC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	WTM	RW	0xFFFF	16-bit watchdot timer match value

9.6.4.16 Timer Watchdog Status Register

This register indicates whether a WDT reset has occurred and caused a system reset, in particular:

- This bit is set when wdt_src_rst# is asserted
- It is cleared by writing a logical 0 to this register
- Clearing this bit is not required for the WDT to be re-activated after a WDT reset event

Offset: 0xC0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0x0	Reserved for future use.
0	WTS	RW	0x0	Watchdog timer reset indication. Indicates that reset was caused by the WDT. Read:

Offset: 0xC0				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Watchdog timer did not cause reset because this bit was cleared. 1: Watchdog timer caused a reset. Write: 0: Clears the WDT reset status. 1: No affect.

9.6.4.17 Timer Watchdog Interrupt Clear Register

Offset: 0xC4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0x0	Reserved for future use.
0	WICLR	WO	0x0	WDT Interrupt clear. Write: 0: No affect. 1: Clear interrupt.

9.6.4.18 Timer Watchdog Counter Reset Register

Offset: 0xC8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0x0	Reserved for future use.
0	WCR	WO	0x0	Watchdog timer counter value reset. Write: 0: No effect. 1: Clears the value of WDT counter.

9.6.4.19 Timer Watchdog Value Register

Offset: 0xCC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	WTV	RO	0x0	Watchdog timer value. Read the current value of WDT. Since the register may be in transition during a read operation, software must perform a double read and compare the two values

Offset: 0xCC				
Bits	**Field**	**Type**	**Reset**	**Description**
				to ensure accuracy.

9.7 RTC

9.7.1 Introduction

The RTC module is capable of counting real time in the unit of seconds independent of the system clock frequency.

9.7.2 Features

- Count of the number of seconds basing on the internal 1-Hz clock
- Possibility to calibrate the frequency of the internal oscillator
- Support for an alarm interrupt and 1-Hz interrupt

9.7.3 Functional Description

The RTC module includes a 32-bit counter for recording real time in the unit of seconds. The system clock is divided into an internal 1-Hz clock for real time counting. Users can configure the initial RTC value based on their preferred settings. The counter value is not affected by transitions into and out of sleep or idle modes.

The RTC is configurable to generate two kinds of interrupt as follows:

- Alarm interrupt
- 1-Hz interrupt

There are two RTC modules in the system:

- Non-secure RTC
- Secure RTC

The secure RTC is only accessible by the secured core, and is reseted by power-on reset, not APB bus reset.

9.7.4 Register Description

The base addresses of RTC registers are tabled below.

Name	Address
Non-secure RTC	0xD401_0000

Name	Address
Secure RTC	0xD401_0400

9.7.4.1 RTC COUNTER REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Time Count	RW	0x0	Time Count of the real time counter, updated at a 1-Hz clock rate.

9.7.4.2 RTC ALARM REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Alarm Time	RW	0x0	Alarm Time for interrupt generation.

9.7.4.3 RTC STATUS REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0x0	Reserved
3	1-Hz interrupt enable	RW	0x0	Enables 1-Hz interrupt: 0x0: Not Enabled 0x1: Enabled
2	RTC alarm interrupt enable	RW	0x0	Enables RTC alarm interrupt: 0x0: Not Enabled 0x1: Enabled
1	1-Hz rising edge detected	W1C	0x0	Flag indicating detection of an 1-Hz rising edge. Writing 1 clears the 1-Hz interrupt.
0	RTC alarm detected	W1C	0x0	Flag indicating detection of an RTC alarm. Writing 1 clears the RTC alarm interrupt.

9.7.4.4 RTC TRIM REGISTER

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Locking bit for the trim value	RW	0x0	Locking bit for the trim value.
30:26	Reserved	RO	0x0	Reserved
25:16	Trim delete count	RW	0x0	This value represents the number of 32-kHz clocks to delete when clock trimming begins.
15:0	Clock divider count	RW	0x7FFF	This value is the integer portion of the clock trim logic.

9.7.4.5 RTC CONTROL REGISTER

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0x0	Reserved
0	Controls the ALARM signal	RW	0x0	This bit enables software control on the ALARM signal that is generated by the processor, and signals any external device that is currently running. So the ALARM signal could be asserted by either software or hardware. 0x0: off (ALARM negated) 0x1: on (ALARM asserted)

9.7.4.6 RTC BACKUP REGISTERS

Offset: 0x14~0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Backup data	RW	0x0	The processor RTC has five back-up registers that store erasable data. The processor can read and write all five 32-bit registers.

9.8 MailBox

9.8.1 Introduction

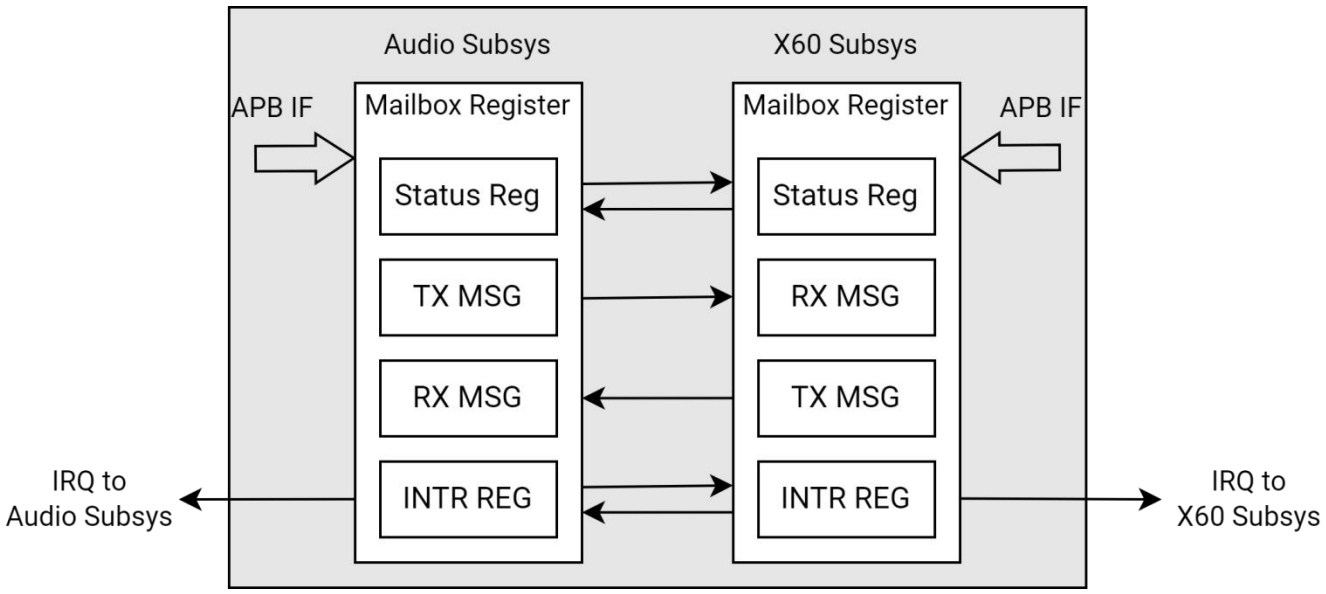
The Mailbox is designed to deliver messages or signals between SoC and MCU subsystem

9.8.2 Features

- A processor is allow to generate an interrupt for another processor
- Support for a polling word to enable signaling an event from one party to another without the need of interrupts
- Reception of an ACK interrupt indicates that the other party is active
- A processor can wake up another processor (supported)

9.8.3 Functional Description

The architecture of the Mailbox is depicted below.



9.8.4 Register Description

With reference to the previous **Chapter 6** & **Chapter 7**

- In X60™ subsystem, the mailbox base address is 0xD4013400 and the interrupt number is 52
- In Audio subsystem, the mailbox base address is 0xC088A000 and the interrupt number is 30

9.8.4.1 ISRR REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	reserved	R	0x0	Reserved
15:0	APB_MBOX_ISRR	R	0x0	Used to obtain the ISRW value of the other side Mailbox.

9.8.4.2 WDR REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	APB_MBOX_WDR	W	0x0	Write Data Includes a 32-bit control word to be transferred to the other side (that will poll it). The content of this word is defined by application.

9.8.4.3 ISRW REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	reserved	R	0x0	Reserved
15:0	APB_MBOX_ISRW	W	0x0	Interrupt Set This register allows setting 4 different interrupts in the other side's interrupt controller. An interrupt can be set by writing 1 to the corresponding interrupt bit in this register, as described below: - Setting bit [10] would generate a SET_GP_INT in the other side's interrupt controller. - Setting bit [9] would generate a SET_MSG_INT in the other side's interrupt controller. - Setting bit [8] would generate a SET_CMD_INT in the other side's interrupt controller. - Setting any one of bits [7:0] would generate a single interrupt called DATA_ACK interrupt. Once an interrupt is set, the corresponding bit is automatically cleared by hardware.

9.8.4.4 ICR REGISTER

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	reserved	R	0x0	Reserved
15:0	APB_MBOX_ICR	W	0x0	Interrupt Clear Used to clear an interrupt asserted by the other side. - Writing **1** to a specific bit clears the corresponding interrupt. - No need to write **0** after **1** — the bit is cleared automatically.

9.8.4.5 IIR REGISTER

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	reserved	RSVD	0xX	Reserved Reserved. Always write 0. Ignore read value.
15:0	APB_MBOX_IIR	RO	0x0	This register is used to read the interrupt source received from the other side. If the Mailbox was identified as the interrupt source, the Interrupt Identification Register can be read to identify which of the 11 possible Mailbox interrupts was asserted. **Notes.** - When [10:8] were the only bits to be asserted, reading the Interrupt Identification Register is redundant, since there is a one-to-one mapping between the interrupt source and interrupt representation on the Interrupt Identification Register. However, when the received interrupt is DATA_ACK interrupt, the Interrupt Identification Register must be read to identify the interrupt cause. - Before reading this register, a dummy write operation (to any Mailbox address range) must be performed to latch the data to the read register. Without performing the write before the read operation, data will not be updated and old data will be read.

9.8.4.6 RDR REGISTER

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	APB_MBOX_RDR	RO	0x0	This register is used to poll the Write Data Register of the other side. **Note.** Before reading this register, a dummy write operation (to any Mailbox address range) must be performed to latch the data to the read register. Without performing the write before the read operation, data will not be updated and old data will be read.

9.9 Power Management & Lower Power Mode Control

9.9.1 Introduction

Various power domains and fine-granularity power states are defined and implemented on the chip to achieve significant power-savings.

9.9.2 Features

- Two power management units distributed on the chip to take control of power-on/off sequence
- Different power domains implemented to achieve power-savings for different scenarios
- Various power states and wake-up events defined to meet various product applications

9.9.3 Functional Description

9.9.3.1 Power Management Unit (PMU)

A two-level power management strategy is implemented to control various granularities of power consumption. Different power domains and power states are also defined to achieve ultra-low power consumption. The two power management units are as follows:

- **Main PMU (MPMU)**
  The Main PMU is responsible for chip-level power management. Once the Application Subsystem PMU (APMU) enters the lowest power state, the Main PMU takes control of the chip-level low-power states.
  For details of the power on/off sequence, refer to **Section 4.4**
- **Application Subsystem PMU (APMU)**
  The Application Subsystem PMU consists of the following major parts:
  - RISCV X60™ Processor clock and low-power mode state machine.
    This state machine controls clock generation for the RISCV X60™ processors. It also generates the sequence of entry into and exit of the RISCV X60™ processor low-power modes.
  - AXI Fabric and DDR clocking and low power mode state machine.
    This state machine controls clock generation for both bus-matrix and DDR. Once the RISCV X60™ processor state machine is in low-power mode, it controls the entry and exit sequence into and out of the bus and DDR low-power modes.

The Application Subsystem PMU takes in charge of X60™ and AXI/DDR power management. Once the Application Subsystem PMU enters into low power mode, the Main PMU then takes over control and may place K1 in chip-level sleep mode according to the Main PMU controls.

9.9.3.2 Power Domains & States

A total of 9 power domains are implemented, and they are for

- CPU cores

  Note. Each CPU core has its own power domain independently controlled

- CPU clusters

  Note. Each CPU cluster has its own power domain independently controlled
- Video Encoder/Decoder
- GPU
- HDMI Display Subsystem
- MIPI DSI Subsystem
- Video Input Subsystem
- RCPU (including N308, Audio Codec, RCPU Peripherals)
- Always-On-Domain (AON)

All those power domains, except AON, can be powered off depending on specific application scenarios.

In order to achieve the minimal power consumption, different power states are designed as tabled below:

No.	Power State Name	Description
1	ACTIVE	The system is alive and active, with all power domains on, except those power domains with power switches that can be turned off selectively and independently.
2	CORE-IDLE	Each core stops executing instructions and enters an idle state, with clock gating automatically after a Wait-for-Interrupt (WFI) execution. The core exits this state when receiving an interrupt routed to it and continues execution.
3	Core-Power-Off	Each core, when voted, enters a power-off state after Core-Idle sleep mode. The core exits this state when receiving an interrupt, with power turned on and reset released.
4	CPU-Cluster-Power-Off	Each CPU cluster, when voted, enters this low-power state after all cores within this cluster have entered the Core-Power-Off state, with L2/TCM memory also shut down. Any active interrupt routing to CPU cores in this cluster would bring CPU cluster out of this state, then power on, clock resume and reset release.
5	Home-Screen	The main bus fabric AXI clock is gated off (if voted) after both CPU clusters enter CPU-Cluster-Power-Off mode. Any interrupt will wake up the chip from this state by resuming the main bus AXI clock, and powering up the corresponding CPU cluster and CPU core to which the interrupt is routed, resuming the CPU clock, and releasing the reset to service the interrupt routine.
6	Chip-Sleep	This is the most ultra-low power state, with all PLLs/Power islands off. Only 32K RTC clock remains alive, and the 24M VCXO can be configured to be on or off. In this state only the logic/IO in AON domain alives, and a pin named SLEEP_OUT connected to PMIC would be deasserted to signal PMIC to lower the VCC power supply voltage to reduce lower power comsumption.
7	RCPU with SOC LP	RCPU power domain is an independent power island and can function in any

No.	Power State Name	Description
		of above PMU states. RCPU can vote for different SoC low-power states according to its specific scenario requirements. The RCPU itself has four low-power states as follows: - Active Mode: Clock running - ClkGate Mode: Clock gating - PLL Off Mode: PLL powered off - Power Off Mode: RCPU power is shut down, but the RCPU AON domain remains alive

Note. VPU, GPU, ISP, DPU power islands can be turned on or off by software, and are independent of the power states No. 1~5 in the table above

9.9.3.3 Wake-Up Resource

- In the **Chip-Sleep Low Power State**, the following interrupts or events can wake-up the chip:
- Pad edge detection
- Keypad press
- RTC / Timer / WDT
- USB / RCPU / AP2AUDIO_IPC
- SD / EMMC / PCIE
- PMIC
- In the **RCPU Power-Off State**, the following interrupts or events can wake-up RCPU PMU to resume its power supply:
- Audio plug interrupt / Hook key interrupt / Class-G short power interrupt / Audio OCP interrupt
- AP IPC power-on request
- RCPU AON Timer wakeup request
- Sensor-Hub GPIO wakeup request

9.9.4 Register Description

9.9.4.1 POWER MODE STATUS REGISTER

Offset: 0xD4050000+0x1030				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	PWRMODE_STATUS	RO	0x0	This field indicates which low power state the system has entered or exited. Each bit corresponds to a different low power mode. If set to 1, it indicates that the respective low power mode has occurred. The status is cleared by setting the corresponding bit in CLR_PWRMODE_STATUS to 1. - bit0: D1P mode - bit1: D1PP mode - bit2: D1 mode

Offset: 0xD4050000+0x1030				
Bits	**Field**	**Type**	**Reset**	**Description**
				- bit3: D2 mode - bit4: D2P - bit5: D2PP mode - bit6: cluster0 M2 - bit7: cluster1 M2 - bit8: cluster2 M2 - bit9: cr5 C2 - bit10: comm_top D2
15:0	CLR_PWRMODE_STATUS	RW	0x0	Clear Power Mode Status. Set 1 to clear the corresponding PWRMODE_STATUS bit.

9.9.4.2 WAKEUP AND CLOCK RESUME LINES STATUS REGISTER (AWUCRS)

Offset: 0xD4050000+0x1048				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Audio	RW	0x0	Audio wake-up
30	RSVD	RO	0x0	Reserved
29	RSVD	RO	0x0	Reserved
28	RSVD	RO	0x0	Reserved
27	RSVD	RO	0x0	Reserved
26	RSVD	RO	0x0	Reserved
25	RSVD	RO	0x0	Reserved
24	RSVD	RO	0x0	Reserved
23	SDH1_AUDIO	RO	0x0	SDH1 wake-up
22	SDH2_SDH3	RO	0x0	SDH2/SDH3 wake-up
21	KEYPRESS	RO	0x0	Key press
20	RSVD	RO	0x0	Reserved
19	RSVD	RO	0x0	Reserved
18	WDT	RO	0x0	WDT
17	RTC_ALARM	RO	0x0	RTC ALARM
16	PMU_TIMER_3	RO	0x0	PMU Timer 3
15	PMU_TIMER_2	RO	0x0	PMU Timer 2
14	PMU_TIMER_1	RO	0x0	PMU Timer 1

Offset: 0xD4050000+0x1048				
Bits	**Field**	**Type**	**Reset**	**Description**
13	AP2_TIMER_3	RO	0x0	AP2 Timer 3
12	AP2_TIMER_2	RO	0x0	AP2 Timer 2
11	AP2_TIMER_1	RO	0x0	AP2 Timer 1
10	AP1_2_TIMER_3	RO	0x0	AP1 Timer 3 and SEC Timer 3
9	AP1_2_TIMER_2	RO	0x0	AP1 Timer 2 and SEC Timer 2
8	AP1_2_TIMER_1	RO	0x0	AP1 Timer 1 and SEC Timer 1
7	WAKEUP7	RO	0x0	Wakeup7 line in status
6	WAKEUP6	RO	0x0	Wakeup6 line in status
5	WAKEUP5	RO	0x0	Wakeup5 line in status
4	WAKEUP4	RO	0x0	Wakeup4 line in status
3	WAKEUP3	RO	0x0	Wakeup3 line in status
2	WAKEUP2	RO	0x0	Wakeup2 line in status
1	WAKEUP1	RO	0x0	Wakeup1 line in status
0	WAKEUP0	RO	0x0	Wakeup0 line in status

9.9.4.3 WAKEUP AND CLOCK RESUME LINES MASK REGISTER (AWUCRM1)

Offset: 0xD4050000+0x1044				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	RSVD	RO	0x0	Reserved
0	IPC_AP2AUD_INT	RW	0x0	Enable ipc_ap2aud_int

9.9.4.4 WAKEUP AND CLOCK RESUME LINES STATUS REGISTER (AAWUCRM0)

Offset: 0xD4050000+0x104C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	AUDIO_WAKEUP	RW	0x0	Enable audio wake-up
30	RSVD	RW	0x0	Reserved
29	RSVD	RW	0x0	Reserved
28	RSVD	RW	0x0	Reserved
27	RSVD	RW	0x0	Reserved

Offset: 0xD4050000+0x104C				
Bits	**Field**	**Type**	**Reset**	**Description**
26	RSVD	RW	0x0	Reserved
25	RSVD	RW	0x0	Reserved
24	RSVD	RW	0x0	Reserved
23	SDH1_AUDIO	RW	0x0	Enable SDH1 wake-up
22	SDH2_SDH3	RW	0x0	Enable SDH2/SDH3 wake-up
21	KEYPRESS	RW	0x0	Enable key press wake-up
20	RSVD	RW	0x0	Reserved
19	RSVD	RW	0x0	Reserved
18	WDT	RW	0x0	Enable WDT
17	RTC_ALARM	RW	0x0	Enable RTC ALARM
16	PMU_TIMER_3	RW	0x0	Enable PMU Timer 3
15	PMU_TIMER_2	RW	0x0	Enable PMU Timer 2
14	PMU_TIMER_1	RW	0x0	Enable PMU Timer 1
13	AP2_TIMER_3	RW	0x0	Enable AP2 Timer 3
12	AP2_TIMER_2	RW	0x0	Enable AP2 Timer 2
11	AP2_TIMER_1	RW	0x0	Enable AP2 Timer 1
10	AP1_2_TIMER_3	RW	0x0	Enable AP1 Timer 3 and SEC Timer 3
9	AP1_2_TIMER_2	RW	0x0	Enable AP1 Timer 2 and SEC Timer 2
8	AP1_2_TIMER_1	RW	0x0	Enable AP1 Timer 1 and SEC Timer 1
7	WAKEUP7	RW	0x0	Enable Wakeup7 input to Pm_clkres
6	WAKEUP6	RW	0x0	Enable Wakeup6 input to Pm_clkres
5	WAKEUP5	RW	0x0	Enable Wakeup5 input to Pm_clkres
4	WAKEUP4	RW	0x0	Enable Wakeup4 input to Pm_clkres
3	WAKEUP3	RW	0x0	Enable Wakeup3 input to Pm_clkres
2	WAKEUP2	RW	0x0	Enable Wakeup2 input to Pm_clkres
1	WAKEUP1	RW	0x0	Enable Wakeup1input to Pm_clkres
0	WAKEUP0	RW	0x0	Enable Wakeup0 input to Pm_clkres

9.9.4.5 WAKEUP AND CLOCK RESUME LINES STATUS REGISTER (AWUCRS1)

Offset: 0xD4050000+0x1064				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	RSVD	RO	0x0	Reserved for future use
0	IPC_AP2AUD_INT	RO	0x0	IPC_AP2AUD_INT

9.9.4.6 CLUSTER 0 POWER CONTROL REGISTER (APCR_CLUSTER0)

Offset: 0xD4050000+0x1090				
Bits	**Field**	**Type**	**Reset**	**Description**
31	AXISDD	RW	0x0	Allows AXI bus and agents to shut down after ASR<var Processor: Application> cores enters idle state. 1'b0: AXI shutdown not allowed 1'b1: AXI shutdown allowed
30	RSVD	RO	0	Must be 1
29	RSVD	RO	0	Must be 1
28	RSVD	RO	0	Reserved
27	DDRCORSD	RW	0x0	Allow ASR <var Processor: Application MP> core and TC DDR clocks to shut down. The clocks are halted when CPCR[DDRCORSD], APCR[DDRCORSD] & DPCR[DDRCORSD] are set, and ASR <var Processor: Application MP> core is in idle mode. 1'b0: ASR <var Processor: Application MP> core and TC DDR clocks shutdown not allowed 1'b1: ASR <var Processor: Application MP> core and TC DDR clocks shutdown allowed
26	APBSD	RW	0x0	Allow PMU to shut down APB clocks to all of its recipients, overriding other per-module fields. The APB clock is shut down once the ASR <var Processor: Application> cores are idle and CPCR[APBSD], APCR[APBSD] & DPCR[APBSD] are set. 1'b0: APB clock shutdown not allowed 1'b1: APB clock shutdown allowed
25	RSVD	RO	0	Must be 1
24:20	RSVD	RO	0	Reserved
19	VCTCXOSD	RW	0x0	Allow VCTCXO shutdown when the system is in sleep mode. VCTCXO is shutdown when CPCR[VCTCXOSD], APCR[VCTCXOSD] & DPCR[VCTCXOSD] are set & the system enters sleep mode

Offset: 0xD4050000+0x1090				
Bits	**Field**	**Type**	**Reset**	**Description**
				1'b0: VCTCXO shutdown not allowed 1'b1: VCTCXO shutdown allowed
18:15	RSVD	RO	0	Reserved
14	RSVD	RO	0	must be 1
13	STBYEN	RW	0x1	Allow Apps Subsystem to shutdown and go into UDR-mode when AP subsystem is in sleep mode. UDR is enabled when CPCR[STBYEN], APCR[STBYEN] are both set & AP subsystem enters AP Sleep.
12:4	RSVD	RO	0	Reserved
3	C0_VOTE_AP_SLPEN	RW	0x1	Cluster1 vote APMU sleep enable
2:0	RSVD	RO	0	Reserved

9.9.4.7 CLUSTER 1 POWER CONTROL REGISTER (APCR_CLUSTER0)

Offset: 0xD4050000+0x1094				
Bits	**Field**	**Type**	**Reset**	**Description**
31	AXISDD	RW	0x0	Allows AXI bus and agents to shut down after ASR<var Processor: Application> cores enters idle state. 1'b0: AXI shutdown not allowed 1'b1: AXI shutdown allowed
30	RSVD	RO	0	Must be 1
29	RSVD	RO	0	Must be 1
28	RSVD	RO	0	Reserved
27	DDRCORSD	RW	0x0	Allow ASR <var Processor: Application MP> core and TC DDR clocks shutdown. The clocks are halted when CPCR[DDRCORSD], APCR[DDRCORSD] & DPCR[DDRCORSD] are set and ASR <var Processor: Application MP> core is in idle mode. 1'b0: ASR <var Processor: Application MP> core and TC DDR clocks shutdown not allowed 1'b1: ASR <var Processor: Application MP> core and TC DDR clocks shutdown allowed
26	APBSD	RW	0x0	Allow PMU to shut down APB clocks to all of its recipients, overriding other per-module fields. The APB clock is actually shut down once the ASR <var Processor: Application> cores are idle and CPCR[APBSD], APCR[APBSD] & DPCR[APBSD] are set.

Offset: 0xD4050000+0x1094				
Bits	**Field**	**Type**	**Reset**	**Description**
				1'b0: APB clock shutdown not allowed 1'b1: APB clock shutdown allowed
25	RSVD	RO	0	must be 1
24:20	RSVD	RO	0	Reserved
19	VCTCXOSD	RW	0x0	Allow VCTCXO shutdown when the system is in sleep mode. VCTCXO is shutdown when CPCR[VCTCXOSD], APCR[VCTCXOSD] & DPCR[VCTCXOSD] are set & the system enters sleep mode 1'b0: VCTCXO shutdown not allowed 1'b1: VCTCXO shutdown allowed
18:15	RSVD	RO	0	Reserved
14	RSVD	RO	0	Must be 1
13	STBYEN	RW	0x1	Allow Apps Subsystem to shutdown and go into UDR-mode when AP subsystem is in sleep mode. UDR is enabled when CPCR[STBYEN], APCR[STBYEN] are both set & AP subsystem enters AP Sleep.
12:4	RSVD	RO	0	Reserved
3	C1_VOTE_AP_SLPEN	RW	0x1	Cluster1 vote APMU sleep enable
2:0	RSVD	RO	0	Reserved

9.9.4.8 ASR PERIPHERAL 1 POWER CONTROL REGISTER (APCR_PER)

Offset: 0xD4050000+0x1098				
Bits	**Field**	**Type**	**Reset**	**Description**
31	AXISDD	RW	0x0	Allow AXI bus and agents to shut down after ASR<var Processor: Application> cores enters idle state. 1'b0: AXI shutdown not allowed 1'b1: AXI shutdown allowed
30	RSVD	RO	0x0	must be 1
29	RSVD	RO	0x0	must be 1
28	RSVD	RO	0x0	Reserved
27	DDRCORSD	RW	0x0	Allow ASR <var Processor: Application MP> core and TC DDR clocks shutdown. The clocks are halted when CPCR[DDRCORSD], APCR[DDRCORSD] & DPCR[DDRCORSD] are set and

Offset: 0xD4050000+0x1098				
Bits	**Field**	**Type**	**Reset**	**Description**
				ASR <var Processor: Application MP> core is in idle mode. 1'b0: ASR <var Processor: Application MP> core and TC DDR clocks shutdown not allowed 1'b1: ASR<var Processor: Application MP> core and TC DDR clocks shutdown allowed
26	APBSD	RW	0x0	Allow PMU to shut down APB clocks to all of its recipients, overriding other per-module fields. The APB clock is actually shut down once the ASR <var Processor: Comm>/ <var Processor: Application> cores are idle and CPCR[APBSD], APCR[APBSD] & DPCR[APBSD] are set. 1'b0: APB clock shutdown not allowed 1'b1: APB clock shutdown allowed
25	RSVD	RO	0x0	must be 1
24:20	RSVD	RO	0x0	Reserved
19	VCTCXOSD	RW	0x0	Allow VCTCXO shutdown when the system is in sleep mode. VCTCXO is shutdown when CPCR[VCTCXOSD], APCR[VCTCXOSD] & DPCR[VCTCXOSD] are set & the system enters sleep mode 0: VCTCXO shutdown not allowed 1: VCTCXO shutdown allowed
18:15	RSVD	RO	0x0	Reserved
14	RSVD	RO	0x0	Must be 1
13	STBYEN	RW	0x1	Allow Apps Subsystem to shutdown and go into UDR-mode when AP subsystem is in sleep mode. UDR is enabled when CPCR[STBYEN], APCR[STBYEN] are both set & AP subsystem enters AP Sleep.
12:4	RSVD	RO	0x0	Reserved
3	PE_VOTE_AP_SLPEN	RW	0x1	PE vote APMU sleep enable
2:0	RSVD	RO	0x0	Reserved

9.9.4.9 Basing on <PMU_BASE(0xD4282800)>

9.9.4.9.1 SDIO/ROTARY WAKE CLEAR REGISTER (PMU_USB_SD_ROT_WAKE_CLR)

Offset: 0xD4282800+0x7C

Bits	Field	Type	Reset	Description
31	USB_WK_INT_STATUS	RO	0x0	USB Wake up status
30:29	RSVD	RO	0x0	Reserved0
28	USB_CHGDET_WK_STATUS	RO	0x0	USB Line charge detect wake up status
27	USB_ID_WK_STATUS	RO	0x0	USB Line ID wake up status
26	USB_VBUS_WK_STATUS	RO	0x0	USB Line vbus valid wake up status
25	USB_LINE1_WK_STATUS	RO	0x0	USB Line state1 wake up status
24	USB_LINE0_WK_STATUS	RO	0x0	USB Line state0 wake up status
23	USB_IDDIG_OVRD_VALUE	RO	0x0	USB IDDIG OVERRIDE VALUE
22	USB_IDDIG_OVRD_EN	RO	0x0	USB IDDIG OVERRIDE ENABLE
21	USB_VBUS_DRV	RO	0x0	USB VBUS DRV
20	USB_CHGDET_WK_CLR	RW	0x0	USB Line charge detect wake up Clear. 1'b1: Clear This bit is self-cleared by hardware
19	USB_ID_WK_CLR	RW	0x0	USB Line ID wake up Clear. 1'b1: Clear This bit is self-cleared by hardware
18	USB_VBUS_WK_CLR	RW	0x0	USB Line vbus valid wake up Clear. 1'b1: Clear This bit is self-cleared by hardware
17	USB_LINE1_WK_CLR	RW	0x0	USB Line state1 wake up Clear 1'b1: Clear This bit is self-cleared by hardware
16	USB_LINE0_WK_CLR	RW	0x0	USB Line state0 wake up Clear 1'b1: Clear This bit is self-cleared by hardware
15	USB_WK_INT_MASK	RW	0x0	USB Wakeup Interrupt Enable. 1'b1: Enable
14:13	RSVD	RO	0	Reserved

Offset: 0xD4282800+0x7C				
Bits	**Field**	**Type**	**Reset**	**Description**
12	USB_CHGDET_WK_MASK	RW	0x0	USB Line charge detect wake up Enable. 1'b1: Enable
11	USB_ID_WK_MASK	RW	0x0	USB Line ID wake up Enable. 1'b1: Enable
10	USB_VBUS_WK_MASK	RW	0x0	USB Line vbus valid wake up Enable 1'b1: Enable
9	USB_LINE1_WK_MASK	RW	0x0	USB Line state1 wake up Enable 1'b1: Enable
8	USB_LINE0_WK_MASK	RW	0x0	USB Line state0 wake up Enable 1'b1: Enable
7	CS_WK_STATUS	RO	0x0	CS wake up status
6	SDH2_WK_CLR	RW	0x1	SDH2 Wake Clear. 1'b1: SDH2 wake event clear This bit is self-cleared by hardware
5	CS_WK_CLR	RW	0x0	Clear of DAP Power Wake Up Request (DAP CSYSPWRUPREQ). 1'b1: Clear DAP_REQ wakeup
4	CS_WK_MASK	RW	0x0	DAP Power Wake Up Enable (DAP CSYSPWRUPREQ) 1'b1: Enable
3	KB_WK_CLR	RW	0x1	Keypad Wake Clear. 1'b1: ROT wake event clear This bit is self-cleared by hardware
2	ROT_WK_CLR	RW	0x1	Rotary Wake Clear. 1'b1: ROT wake event clear This bit is self-cleared by hardware
1	SDH1_WK_CLR	RW	0x1	SDH1 Wake Clear. 1'b1: SDH1 wake event clear This bit is self-cleared by hardware
0	SDH0_WK_CLR	RW	0x1	SDH0 Wake Clear. 1'b1: SDH0 wake event clear This bit is self-cleared by hardware

9.9.4.9.2 POWER STABLE TIMER REGISTER (PMU_PWR_STBL_TIMER)

Offset: 0xD4282800+0x7C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	RO	0x0	Reserved
23:16	PWR_CLK_PRE	RW	0x02	Clock Prescaler for Timer Count. 0x0 = Divide by 1 0x1 = Divide by 1 0x2 = Divide by 2 All other values use an incremental divider.
15:8	PWR_UP_STBL_TIMER	RW	0x40	Power-Up Stable Timer Defines the stable time required for power-up during core idle mode in 24 MHz unit
7:0	PWR_DWN_STBL_TIMER	RW	0x00	Power Down Stable Timer Defines the stable time required for power-down during core idle mode in 24 MHz unit

9.9.4.9.3 CORE STATUS REGISTER (PMU_CORE_STATUS)

Offset: 0xD4282800+0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
31	AP_CORE7_C2	RO	0x1	CORE7 in C2 Mode Indication. Indicates whether CORE7 is in C2 mode. 1'b1: CORE7 is in C2 mode
30	AP_CORE7_C1	RO	0x0	CORE7 in C1 Mode Indication. Indicates whether CORE7 is in C1 mode (external mode). 1'b1: CORE7 is in C1 mode
29	AP_CORE7_WFI_FLAG	RO	0x1	CORE7 WFI Flag Reflects the WFI flag generated by CORE7 When CORE7 enters WFI, this field is set
28	AP_CORE6_C2	RO	0x1	CORE6 in C2 Mode Indication. Indicates whether CORE6 is in C2 mode. 1'b1: CORE6 is in C2 mode
27	AP_CORE6_C1	RO	0x0	CORE6 in C1 Mode Indication. Indicates whether CORE6 is in C1 mode (external mode). 1'b1: CORE6 is in C1 mode
26	AP_CORE6_WFI_FLAG	RO	0x1	CORE6 WFI Flag Reflects the WFI flag generated by CORE6 When CORE6 enters WFI, this field is set

Offset: 0xD4282800+0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
25	AP_CORE5_C2	RO	0x1	CORE5 in C2 Mode Indication. Indicates whether CORE5 is in C2 mode. 1'b1: CORE5 is in C2 mode
24	AP_CORE5_C1	RO	0x0	CORE5 in C1 Mode Indication. Indicates whether CORE5 is in C1 mode (external mode). 1'b1: CORE5 is in C1 mode
23	AP_CORE5_WFI_FLAG	RO	0x1	CORE5 WFI Flag Reflects the WFI flag generated by CORE5 When CORE5 enters WFI, this field is set
22	AP_CORE4_C2	RO	0x1	CORE4 in C2 Mode Indication. Indicates whether CORE4 is in C2 mode. 1'b1: CORE4 is in C2 mode
21	AP_CORE4_C1	RO	0x0	CORE4 in C1 Mode Indication. Indicates whether CORE4 is in C1 mode (external mode). 1'b1: CORE4 is in C1 mode
20	AP_CORE4_WFI_FLAG	RO	0x1	CORE4 WFI Flag Reflects the WFI flag generated by CORE4 When CORE4 enters WFI, this field is set
19	AP_C1_MPSUB_M2	RO	0x1	Cluster1 Subsystem idle mode Indication. Indicates whether cluster1 Subsystem is in M2 mode. 1'b1: Cluster1 Subsystem is in M2 mode
18	AP_C1_MPSUB_M1	RO	0x0	Cluster1 Subsystem in M1 Mode Indication Indicates whether Cluster1 Subsystem is in M1 mode (external idle mode). 1'b1: Cluster1 Subsystem is in M1 mode
17	AP_C1_MPSUB_IDLE_FLAG	RO	0x1	Cluster1 Subsystem Idle Flag Reflects the AND logic value of SCU_IDLE and L2CLKSTOPPED generated for Cluster1 Subsystem. 1'b1: core0/core1/core2/core3/scu/L2 are all in idle state
16	SP_IDLE	RO	0x1	Core idle mode Indication Indicates whether Core is in "core idle mode" mode. 1'b1: Core is in core idle mode
15	AP_CORE3_C2	RO	0x1	CORE3 in C2 Mode Indication. Indicate whether CORE3 is in C2 mode. 1'b1: CORE3 is in C2 mode
14	AP_CORE3_C1	RO	0x0	CORE3 in C1 Mode Indication. Indicate whether CORE3 is in C1 mode (external idle mode).

Offset: 0xD4282800+0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
				1'b1: CORE3 is in C1 mode
13	AP_CORE3_WFI_FLAG	RO	0x1	CORE3 WFI Flag Reflects the WFI flag generated by CORE3. When CORE3 enters WFI, this field is set.
12	AP_CORE2_C2	RO	0x1	CORE2 in C2 Mode Indication. Indicate whether CORE2 is in C2 mode. 1'b1: CORE2 is in C2 mode
11	AP_CORE2_C1	RO	0x0	CORE2 in C1 Mode Indication. Indicate whether CORE2 is in C1 mode (external idle mode). 1'b1: CORE2 is in C1 mode
10	AP_CORE2_WFI_FLAG	RO	0x1	CORE2 WFI Flag Reflects the WFI flag generated by CORE2. When CORE2 enters WFI, this field is set.
9	AP_CORE1_C2	RO	0x1	CORE1 in C2 Mode Indication. Indicate whether CORE1 is in C2 mode. 1'b1: CORE1 is in C2 mode
8	AP_CORE1_C1	RO	0x0	CORE1 in C1 Mode Indication. Indicate whether CORE1 is in C1 mode (external idle mode). 1'b1: CORE1 is in C1 mode
7	AP_CORE1_WFI_FLAG	RO	0x1	CORE1 WFI Flag Reflects the WFI flag generated by CORE1. When CORE1 enters WFI, this field is set.
6	AP_CORE0_C2	RO	0x0	CORE0 in C2 Mode Indication. Indicate whether CORE0 is in C2 mode. 1'b1: CORE0 is in C2 mode
5	AP_CORE0_C1	RO	0x0	CORE0 in C1 Mode Indication. Indicate whether CORE0 is in C1 mode (external idle mode). 1'b1: CORE0 is in C1 mode
4	AP_CORE0_WFI_FLAG	RO	0x0	CORE0 WFI Flag Reflects the WFI flag generated by CORE0. When CORE0 enters WFI, this field is set.
3	AP_C0_MPSUB_M2	RO	0x0	Cluster0 Subsystem idle mode Indication. Indicates whether Cluster1 MP Subsystem is in M2 mode. 1'b1: Cluster0 MP Subsystem is in M2 mode
2	AP_C0_MPSUB_M1	RO	0x0	Cluster0 Subsystem in M1 Mode Indication Indicates whether Cluster0 MP Subsystem is in M1

Offset: 0xD4282800+0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
				mode (external idle mode). 1'b1: Cluster0 MP Subsystem is in M1 mode
1	AP_C0_MPSUB_IDLE_FLAG	RO	0x0	Cluster0 Subsystem Idle Flag. Reflects the AND logic value of SCU_IDLE and L2CLKSTOPPED generated in the Cluster0 MP Subsystem. 1'b1: core0/core1/core2/core3/scu/L2 are all in idle state
0	RSVD	RO	0x0	Reserved

9.9.4.9.4 RESUME FROM SLEEP CLEAR REGISTER (PMU_RES_FRM_SLP_CLR)

Offset: 0xD4282800+0x94				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	RSVD	RO	0x0	Reserved
0	CLR_RSM_FRM_SLP	RW	0x0	Clear Resume from Sleep Indication 1'b1: Clear the status signal in CIU sys_boot_cntr[14]

9.9.4.9.5 VPU POWER CONTROL REGISTER (PMU_VPU_PWR_CTRL)

Offset: 0xD4282800+0xA8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	RSVD	RW	0x0	Reserved
3	VPU_SLEEP2	RW	0x0	VPU Power Switch Sleep2
2	VPU_SLEEP1	RW	0x0	VPU Power Switch Sleep1
1	VPU_ISOB	RW	0x0	VPU Isolation Wrapper 1'b0: Enable isolation (VPU power-down mode) 1'b1: disable isolation (VPU active mode)
0	RSVD	RW	0x0	Reserved

9.9.4.9.6 GPU POWER CONTROL REGISTER (PMU_GPU_PWR_CTRL)

Offset: 0xD4282800+0xD0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	RSVD	RW	0x0	Reserved
3	GPU_SLEEP2	RW	0x0	GPU Power Switch Sleep2
2	GPU_SLEEP1	RW	0x0	GPU Power Switch Sleep1
1	GPU_ISOB	RW	0x0	GPU Isolation Wrapper 1'b0: Enable isolation (GPU power-down mode) 1'b1: disable isolation (GPU active mode)
0	RSVD	RW	0x0	Reserved

9.9.4.9.7 BLOCK POWER TIMER REGISTER (PMUA_PWR_BLK_TMR_REG)

Offset: 0xD4282800+0xDC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	PWR_ON1_TIMER	RW	0x8	Delay for GPU/VPU/ISP/Audio auto-power on from Sleep1 to Sleep2
15:8	PWR_ON2_TIMER	RW	0x64	Delay for GPU/VPU/ISP/Audio auto-power on from Sleep2 to clock_en
7:0	PWR_OFF_TIMER	RW	0x0	Delay for GPU/VPU/ISP/Audio auto-power off from Sleep2 to Sleep1

9.9.4.9.8 CLUSTER0 MP IDLE CONFIGURATION REGISTER FOR CORE X (PMU_C0_CAPMP_IDLE_CFGX_X;X=0/1/2/3)

This register is used by cluster0 core x to vote MP subsystem low power mode.

PMU_C0_IDLE_CFGx (x=0,1,2,3) are fully symmetric. Each bit of the register takes effect only when all corresponding bits of PMU_C0__IDLE_CFGx are set to 1.

Offset: 0xD4282800+0x120/0xE4/0x150/0x154				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	RSVD	RO	0	Reserved
19	DIS_MP_L2_SLP	RW	0x0	Disable L2 Power Switch Disables the MP L2 power switch sleep power down during MP power down mode. 1'b1: Disable MP L2 power switch
18	DIS_MP_SLP	RW	0x0	Disable MP Power Switch.

Offset: 0xD4282800+0x120/0xE4/0x150/0x154				
Bits	**Field**	**Type**	**Reset**	**Description**
				Disables the MP power switch sleep power down during MP subsystem power down mode. 1'b1: Disable MP power switch sleep
17	RSVD	RO	0	
16	FRC_L2_SRAM_Off	RW	0x0	Frequency Change L2 SRAM Off. 1'b1: L2 Frequency Change is off
15:14	RSVD	RO	0	Reserved
13	L2_HW_CACHE_FLUSH_EN	RW	0x0	L2 Hardware Cache Flush Enable. 1'b1: Enable
12	MASK_SRAM_REPAIR_DONE_CHECK	RW	0x0	Mask SRAM Repair Done Check. 1'b1: Mask SRAM repair done check
11	MASK_CLK_OFF_CHECK	RW	0x0	Mask the MP Clock Off State Check. Masks the MP clock off check during the MP idle process.
10	MASK_CLK_STBL_CHECK	RW	0x0	Mask MP clock stable State Check. Masks the MP clock stable check during MP wakeup.
9	MASK_JTAG_IDLE_CHECK	RW	0x0	Mask JTAG Idle State Check. Masks the JTAG idle check during the MP idle entry. 1'b1: Mask JTAG idle check
8	MASK_IDLE_CHECK	RW	0x0	Mask the MP Idle State Check.
7	ACINACTM_HW_CTRL	RW	0x0	ACINACTM Hardware Control. 1'b0: low power state machine does not control ACINACTM port. 1'b1: low power state machine controls ACINACTM port of MP. When M2/M1 low power mode is entered, ACINACTM port will be high
6	RSVD	RO	0	Reserved
5	DIS_MC_SW_REQ	RW	0x0	Disable Memory Controller Software Request. Disables the Memory Controller entry to idle mode using the Memory Controller sleep request bits. The Memory Controller will always enter into idle mode based on the hardware state machine
4	MP_WAKE_MC_EN	RW	0x0	MP Wake Memory Controller Enable. Wakes up the Memory Controller when the MP wakes up from idle mode. The Memory Controller is woken up before the interrupt to the core is

Offset: 0xD4282800+0x120/0xE4/0x150/0x154				
Bits	**Field**	**Type**	**Reset**	**Description**
				released. 1'b0: Memory Controller wakes up if MP wakes up from idle mode
3	MP_SCU_SRAM_PWRDWN	RW	0x0	No Used. SCU SRAM does not support retention
2	L2_SRAM_PWRDWN	RW	0x0	L2 Cache SRAM Power Down This field does not take effect if MP_PWRDWN is 0. 1'b1: When MP is idle, L2 SRAM power will be off. 1'b0: When MP is idle, L2 SRAM is in retention mode
1	MP_PWRDWN	RW	0x0	MP Power Down. This field does not take effect if MP_IDLE is 0. 1'b1: When MP is idle, MP will go into deep sleep mode and the MP logic will be power-gated
0	MP_IDLE	RW	0x0	MP Idle. 1'b1: When MP is idle, the MP clocks will be gated externally

9.9.4.9.9 POWER STATUS REGISTER (PMUA_PWR_STATUS_REG)

Offset: 0xD4282800+0xF0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:13	RSVD	RO	0x0	Reserved
12	LCD_HW_PWR_STAT	RO	0x0	HW Power Mode updates the status when the LCD enters/exits its LPM.
11	AUDIO_HW_PWR_STAT	RO	0x0	HW Power Mode updates the status when the Audio enters/exits its LPM. 1'b0: Audio is powered off 1'b1: Audio is powered on
10	ISP_HW_PWR_STAT	RO	0x0	HW Power Mode updates the status when the ISP enters/exits its LPM.
9	VPU_HW_PWR_STAT	RO	0x0	HW Power Mode updates the status when the VPU enters/exits its LPM.
8	GPU_HW_PWR_STAT	RO	0x0	HW Power Mode updates the status when the GPU enters/exits its LPM.
7:5	RSVD	RO	0	Reserved
4	LCD_PWR_STATUS	RO	0x0	LCD power state including both HW and SW power

Offset: 0xD4282800+0xF0				
Bits	**Field**	**Type**	**Reset**	**Description**
				mode. 1'b1: Power on 1'b0: Power off
3	AUDIO_PWR_STATUS	RO	0x0	Audio Power State including both HW and SW power mode. 1'b1: Power on 1'b0: Power off
2	ISP_PWR_STATUS	RO	0x0	ISP power state including both HW and SW power mode. 1'b1: Power on 1'b0: Power off
1	VPU_PWR_STATUS	RO	0x0	VPU power state including both HW and SW power mode. 1'b1: Power on. 1'b0: Power off
0	GPU_PWR_STATUS	RO	0x0	GPU power state including both HW and SW power mode. 1'b1: Power on. 1'b0: Power off

9.9.4.9.10 USB PHY READ REGISTER (PMUA_USB_PHY_READ)

Offset: 0xD4282800+0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD3	RO	0x0	Reserved
21	USB3_PHY_RXELECIDLE	RO	0x0	USB3 PHY RXELECIDLE output
20:19	USB3_PHY_LINESTATE	RO	0x0	USB3 PHY line_state[1:0] output
18	USB3_PHY_VBUSVALID	RO	0x0	USB3 PHY Vbus Valid output
17	USB3_PHY_IDDIG	RO	0x0	USB3 PHY ID DIG output
16	USB3_PHY_CHGDECT	RO	0x0	USB3 PHY CHGDECT output
15:13	RSVD1	RO	0x0	Reserved
12:11	USBP1_PHY_LINESTATE	RO	0x0	USBP1 PHY line_state[1:0] output
10	USBP1_PHY_VBUSVALID	RO	0x0	USBP1 PHY VbusValid output
9	USBP1_PHY_IDDIG	RO	0x0	USBP1 PHY ID DIG output
8	USBP1_PHY_CHGDECT	RO	0x0	USBP1 PHY CHGDECT output

Offset: 0xD4282800+0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
7:5	RSVD0	RO	0x0	Reserved
4:3	USB_PHY_LINESTATE	RO	0x0	USB PHY line_state[1:0] output
2	USB_PHY_VBUSVALID	RO	0x0	USB PHY VbusValid output
1	USB_PHY_IDDIG	RO	0x0	USB PHY ID DIG output
0	USB_PHY_CHGDECT	RO	0x0	USB PHY CHGDECT output

9.9.4.9.11 CORE X IDLE CONFIGURATION REGISTER (PMU_CAP_COREX_IDLE_CFG_X)

Offset: 0xD4282800+0x124/0x128/0x160/0x164/0x304/0x308/0x30C/0x310				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	RO	0x0	Reserved
11	MASK_CLK_OFF_CHECK	RW	0x0	Mask core clock off check during core idle process.
10	MASK_CLK_STBL_CHECK	RW	0x0	Mask core clock stable check during core wakeup.
9	MASK_JTAG_IDLE_CHECK	RW	0x0	Mask the JTAG idle check during MP idle entry. 1'b1: Mask the JTAG idle check.
8	MASK_CORE_WFI_IDLE_CHECK	RW	0x0	Mask the Core WFI IDLE check during MP idle entry. 1'b1: Mask the core wait for interrupt idle check
7:5	RSVD	RO	0x0	Reserved
4	MASK_GIC_NFIQ_TO_CORE	RW	0x0	Mask nFIQ generated in GIC for CORE. Software can set this bit before CORE enter C2. APMU hardware will automatically clear this bit when CORE enters C2
3	MASK_GIC_NIRQ_TO_CORE	RW	0x0	Mask nIRQ generated in GIC for CORE. Software can set this bit before CORE enter C2. APMU hardware will automatically clear this bit when CORE enters C2
2	RSVD	RO	0x0	Reserved
1	CORE_PWRDWN	RW	0x0	Core Power Down. This bit does not takes effect if CORE_IDLE is 0. 1'b1: When core issues WFI idle, core goes into deep sleep mode and power is off.

Offset: 0xD4282800+0x124/0x128/0x160/0x164/0x304/0x308/0x30C/0x310				
Bits	**Field**	**Type**	**Reset**	**Description**
				This bit will not take effect if dbgnopwrdwn is set
0	CORE_IDLE	RW	0x0	Core Idle. 1'b1: When core issues WFI idle, the core clock will be gated externally. This bit will not take effect if dbgnopwrdwn is set

9.9.4.9.12 CORE X WAKEUP REGISTER (PMU_CAP_COREX_WAKEUP_X)

This register is used by Core x software to wake up other cores in Cluster0/1.

To avoid software lock issues, other cores in the MP subsystem should not write to this register.

Offset: 0xD4282800+0x12C/0x130/0x134/0x138/0x324/0x328/0x32C/0x330				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	RO	0	Reserved
7	WAKEUP_CORE7	RW	0x0	Wakeup Core7 - Writing 1 to this field: 1. Wakes up Core 7 2. Has no effect if Core 7 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
6	WAKEUP_CORE6	RW	0x0	Wakeup Core6 - Writing 1 to this field: 1. Wakes up Core 6 2. Has no effect if Core 6 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
5	WAKEUP_CORE5	RW	0x0	Wakeup Core5 - Writing 1 to this field: 1. Wakes up Core 5 2. Has no effect if Core 5 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
4	WAKEUP_CORE4	RW	0x0	Wakeup Core 4 - Writing 1 to this field: 1. Wakes up Core 4 2. Has no effect if Core 4 is in C0 mode - Writing 0 to this field: No effect

Offset: 0xD4282800+0x12C/0x130/0x134/0x138/0x324/0x328/0x32C/0x330				
Bits	**Field**	**Type**	**Reset**	**Description**
				The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
3	WAKEUP_CORE3	RW	0x0	Wakeup Core3 - Writing 1 to this field: 1. Wakes up Core 3 2. Has no effect if Core 3 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
2	WAKEUP_CORE2	RW	0x0	Wakeup Core2 - Writing 1 to this field: 1. Wakes up Core 2 2. Has no effect if Core 2 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
1	WAKEUP_CORE1	RW	0x0	Wakeup Core1 - Writing 1 to this field: 1. Wakes up Core 1 2. Has no effect if Core 1 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.
0	WAKEUP_CORE0	RW	0x0	Wakeup Core 0 - Writing 1 to this field: 1. Wakes up Core 0 2. Has no effect if Core 0 is in C0 mode - Writing 0 to this field: No effect The PMU hardware automatically clears this bit when Core 3 exits C1/C2 mode.

9.9.4.9.13 CLUSTER1 MP IDLE CONFIGURATION REGISTER FOR CORE X (PMU_C1_CAPMP_IDLE_CFGX_X)

This register is used by Cluster 1 - Core x to vote for MP subsystem low power mode, where:

- PMU_C1_IDLE_CFGx (x=0,1,2,3) registers are fully symmetric.
- Each bit in the register takes effect only when all corresponding bits of PMU_C1__IDLE_CFGx are set to 1.

Offset: 0xD4282800+0x314/0x318/0x31C/0x320				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	RSVD	RO	0	Reserved
19	DIS_MP_L2_SLP	RW	0x0	Disable L2 Power Switch. Disables the MP L2 power switch sleep power down during MP power down mode. 1'b1: Disables MP L2 power switch
18	DIS_MP_SLP	RW	0x0	Disable MP Power Switch. Disables the MP power switch sleep power down during MP subsystem power down mode. 1'b1: Disables MP power switch sleep
17	RSVD	RO	0	
16	FRC_L2_SRAM_Off	RW	0x0	Frequency Change L2 SRAM Off. 1'b1: L2 Frequency change is off
15:14	RSVD	RO	0	Reserved
13	L2_HW_CACHE_FLUSH_EN	RW	0x0	L2 Hardware Cache Flush Enable 1'b1: Enable
12	MASK_SRAM_REPAIR_DONE_CHECK	RW	0x0	Mask SRAM Repair Done Check 1'b1: Mask SRAM repair done check
11	MASK_CLK_OFF_CHECK	RW	0x0	Mask the MP Clock Off State Check. Masks the MP clock off check during the MP idle process
10	MASK_CLK_STBL_CHECK	RW	0x0	Mask MP clock stable State Check. Masks the MP clock stable check during MP wakeup.
9	MASK_JTAG_IDLE_CHECK	RW	0x0	
8	MASK_IDLE_CHECK	RW	0x0	Mask the MP Idle State Check.
7	ACINACTM_HW_CTRL	RW	0x0	ACINACTM Hardware Control 1'b0: The low-power state machine does not control the ACINACTM port. 1'b1: The low-power state machine controls the ACINACTM port of MP. When M2/M1 low-power mode is entered, ACINACTM port will be set high
6	RSVD	RO	0	Reserved
5	DIS_MC_SW_REQ	RW	0x0	Disable Memory Controller Software Request Disables the Memory Controller entry to idle mode using the Memory Controller sleep request bits.

Offset: 0xD4282800+0x314/0x318/0x31C/0x320				
Bits	**Field**	**Type**	**Reset**	**Description**
				The Memory Controller will always enter into idle mode based on the hardware state machine
4	MP_WAKE_MC_EN	RW	0x0	MP Wake Memory Controller Enable The Memory Controller will wake up before the interrupt to the core is released. 1'b0: Memory Controller will wake up when MP wakes up from idle mode
3	MP_SCU_SRAM_PWRDWN	RW	0x0	No Used SCU SRAM does not support retention
2	L2_SRAM_PWRDWN	RW	0x0	L2 Cache SRAM Power Down This field does not take effect if MP_PWRDWN is 0. 1'b1: When MP is idle, L2 SRAM power will be off 1'b0: When MP is idle, L2 SRAM is in retention mode
1	MP_PWRDWN	RW	0x0	MP Power Down This field does not take effect if MP_IDLE is 0. 1'b1: When the MP is idle, MP enters deep sleep mode and the MP logic will be power-gated
0	MP_IDLE	RW	0x0	MP Idle 1'b1: When MP is idle, the MP clocks will be gated externally

9.9.4.9.14 AUDIO POWER CONTROL REGISTER (PMUA_PWR_CTRL_AUDIO)

Offset: 0xD4282800+0x378				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	HW_MODE	RW	0x0	Hardware Control Mode for AUD Power Up and Power Down
3	AUD_SLEEP2	RW	0x0	AUD Power Switch Sleep2
2	AUD_SLEEP1	RW	0x0	AUD Power Switch Sleep1
1	AUD_ISOB	RW	0x0	AUD Isolation Wrapper 1'b0: Enable isolation (AUD power-down mode) 1'b1: Disable isolation (AUD active mode)
0	AUD_AUTO_PWR_ON	RW	0x0	AUD Auto Power On 1: Triggers a request to power on the ISP power island. 0: Triggers a request to power down the ISP power island, but only if PWR_CTRL_AUD[4] is set.

9.9.4.9.15 ISP POWER CONTROL REGISTER (PMUA_PWR_CTRL_ISP)

Offset: 0xD4282800+0x37C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	HW_MODE	RW	0x0	Hardware Control Mode for ISP Power Up and Power Down
3	ISP_SLEEP2	RW	0x0	ISP Power Switch Sleep2
2	ISP_SLEEP1	RW	0x0	ISP Power Switch Sleep1
1	ISP_ISOB	RW	0x0	ISP Isolation Wrapper. 1'b0: Enable isolation (ISP power-down mode). 1'b1: Disable isolation (ISP active mode)
0	ISP_AUTO_PWR_ON	RW	0x0	ISP Auto Power On 1: Triggers a request to power on the ISP power island. 0: Triggers a request to power down the ISP power island, but only if PWR_CTRL_ISP[4] is set.

9.9.4.9.16 LCD POWER CONTROL REGISTER (PMUA_PWR_CTRL_LCD)

Offset: 0xD4282800+0x380				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	HW_MODE	RW	0x0	Hardware Control Mode for LCD Power Up and Power Down
3	LCD_SLEEP2	RW	0x0	LCD Power Switch Sleep2
2	LCD_SLEEP1	RW	0x0	LCD Power Switch Sleep1
1	LCD_ISOB	RW	0x0	LCD Isolation Wrapper. 1'b0: Enable isolation (LCD power-down mode). 1'b1: Disable isolation (LCD active mode)
0	LCD_AUTO_PWR_ON	RW	0x0	LCD Auto Power On 1: Triggers a request to power on the ISP power island. 0: Triggers a request to power down the ISP power island, but only if PWR_CTRL_LCD[4] is set.

9.9.4.9.17 HDMI POWER CONTROL REGISTER (PMUA_PWR_CTRL_HDMI)

Offset: 0xD4282800+0x3F4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	RO	0x0	Reserved
4	HW_MODE	RW	0x0	Hardware Control Mode for HDMI Power Up and Power Down
3	HDMI_SLEEP2	RW	0x0	HDMI Power Switch Sleep2
2	HDMI_SLEEP1	RW	0x0	HDMI
1	HDMI_ISOB	RW	0x0	HDMI Isolation Wrapper. 1'b0: Enable isolation (HDMI power-down mode) 1'b1: Disable isolation (HDMI active mode)
0	HDMI_AUTO_PWR_ON	RW	0x0	HDMI Auto Power On 1: Triggers a request to power on the ISP power island. 0: Triggers a request to power down the ISP power island, but only if PWR_CTRL_HDMI[4] is set.

9.9.4.9.18 USB WAKE CLEAR REGISTER (PMU_USBP1_WAKE_CLR)

Offset: 0xD4282800+0x3C4				
Bits	**Field**	**Type**	**Reset**	**Description**
31	USBP1_WK_INT_STATUS	RO	0x0	USBP1 Wake up status
30:29	RSVD	RO	0x0	Reserved
28	USBP1_CHGDET_WK_STATUS	RO	0x0	USBP1 Line charge detect wake up status
27	USBP1_ID_WK_STATUS	RO	0x0	USBP1 Line ID wake up status
26	USBP1_VBUS_WK_STATUS	RO	0x0	USBP1 Line vbus valid wake up status
25	USBP1_LINE1_WK_STATUS	RO	0x0	USBP1 Line state1 wake up status
24	USBP1_LINE0_WK_STATUS	RO	0x0	USBP1 Line state0 wake up status
23	USBP1_IDDIG_OVRD_VALUE	RO	0x0	USBP1 IDDIG OVERRIDE VALUE
22	USBP1_IDDIG_OVRD_EN	RO	0x0	USBP1 IDDIG OVERRIDE ENABLE
21	USBP1_VBUS_DRV	RO	0x0	USBP1 VBUS DRV
20	USBP1_CHGDET_WK_CLR	RW	0x0	USBP1 Line charge detect wake up Clear 1'b1: Clear This bit is self-cleared by hardware
19	USBP1_ID_WK_CLR	RW	0x0	USBP1 Line ID wake up Clear 1'b1: Clear This bit is self-cleared by hardware

Offset: 0xD4282800+0x3C4				
Bits	**Field**	**Type**	**Reset**	**Description**
18	USBP1_VBUS_WK_CLR	RW	0x0	USBP1 Line vbus valid wake up Clear 1'b1: Clear. This bit is self-cleared by hardware
17	USBP1_LINE1_WK_CLR	RW	0x0	USBP1 Line state1 wake up Clear 1'b1: Clear This bit is self-cleared by hardware
16	USBP1_LINE0_WK_CLR	RW	0x0	USBP1 Line state0 wake up Clear 1'b1: Clear This bit is self-cleared by hardware
15	USBP1_WK_INT_MASK	RW	0x0	USBP1 Wakeup Interrupt Enable 1’b1: Enable
14:13	RSVD	RO	0x0	Reserved
12	USBP1_CHGDET_WK_MASK	RW	0x0	USBP1 Line charge detect wake up Enable 1’b1: Enable
11	USBP1_ID_WK_MASK	RW	0x0	USBP1 Line ID wake up Enable 1’b1: Enable
10	USBP1_VBUS_WK_MASK	RW	0x0	USBP1 Line vbus valid wake up Enable 1’b1: Enable
9	USBP1_LINE1_WK_MASK	RW	0x0	USBP1 Line state1 wake up Enable 1’b1: Enable
8	USBP1_LINE0_WK_MASK	RW	0x0	USBP1 Line state0 wake up Enable 1’b1: Enable
7:0	RSVD	RO	0x0	Reserved

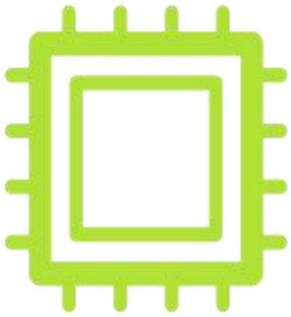
9.9.4.9.19 USB3 WAKE CLEAR REGISTER (PMU_USB3_WAKE_CLR)

Offset: 0xD4282800+0x3C8				
Bits	**Field**	**Type**	**Reset**	**Description**
31	USB3_WK_INT_STATUS	RO	0x0	USB3 Wake up status
30	RSVD	RO	0x0	Reserved
29	USB3_LFPS_WK_STATUS	RO	0x0	USB3 LFPS wake up status
28	USB3_CHGDET_WK_STATUS	RO	0x0	USB3 Line charge detect wake up status
27	USB3_ID_WK_STATUS	RO	0x0	USB3 Line ID wake up status
26	USB3_VBUS_WK_STATUS	RO	0x0	USB3 Line vbus valid wake up status
25	USB3_LINE1_WK_STATUS	RO	0x0	USB3 Line state1 wake up status

Offset: 0xD4282800+0x3C8				
Bits	**Field**	**Type**	**Reset**	**Description**
24	USB3_LINE0_WK_STATUS	RO	0x0	USB3 Line state0 wake up status
23	USB3_IDDIG_OVRD_VALUE	RO	0x0	USB3 IDDIG OVERRIDE VALUE
22	USB3_IDDIG_OVRD_EN	RO	0x0	USB3 IDDIG OVERRIDE ENABLE
21	USB3_VBUS_DRV	RO	0x0	USB3 VBUS DRV
20	USB3_CHGDET_WK_CLR	RW	0x0	USB3 Line charge detect wake up Clear 1'b1: Clear This bit is self-cleared by hardware.
19	USB3_ID_WK_CLR	RW	0x0	USB3 Line ID wake up Clear. 1'b1: Clear This bit is self-cleared by hardware.
18	USB3_VBUS_WK_CLR	RW	0x0	USB3 Line vbus valid wake up Clear 1'b1: Clear This bit is self-cleared by hardware.
17	USB3_LINE1_WK_CLR	RW	0x0	USB3 Line state1 wake up Clear 1'b1: Clear This bit is self-cleared by hardware.
16	USB3_LINE0_WK_CLR	RW	0x0	USB3 Line state0 wake up Clear 1'b1: Clear This bit is self-cleared by hardware.
15	USB3_WK_INT_MASK	RW	0x0	USB3 Wakeup Interrupt Enable 1'b1: Enable
14	USB3_LFPS_WK_CLR	RW	0x0	USB3 LFPS wake up Clear 1'b1: Clear This bit is self-cleared by hardware.
13	USB3_LFPS_WK_MASK	RW	0x0	USB3 LFPS wake up Enable 1'b1: Enable
12	USB3_CHGDET_WK_MASK	RW	0x0	USB3 Line charge detect wake up Enable 1'b1: Enable
11	USB3_ID_WK_MASK	RW	0x0	USB3 Line ID wake up Enable 1'b1: Enable
10	USB3_VBUS_WK_MASK	RW	0x0	USB3 Line vbus valid wake up Enable 1'b1: Enable
9	USB3_LINE1_WK_MASK	RW	0x0	USB3 Line state1 wake up Enable 1'b1: Enable
8	USB3_LINE0_WK_MASK	RW	0x0	USB3 Line state0 wake up Enable 1'b1: Enable
7:0	RSVD	RO	0x0	Reserved

Chapter 10

Memory & Storage



Key Stone® K1 User Manual

10.1 Overview

K1 supports on-chip memory and various peripheral interfaces connecting to different memory devices such as DDR and NAND/NOR Flash.

10.2 On-Chip Memory

10.2.1 Introduction

K1 includes the following on-chip memory types.

- 128KB boot-ROM
- 256KB SRAM shared by Main CPU and Real CPU

10.3 DDR

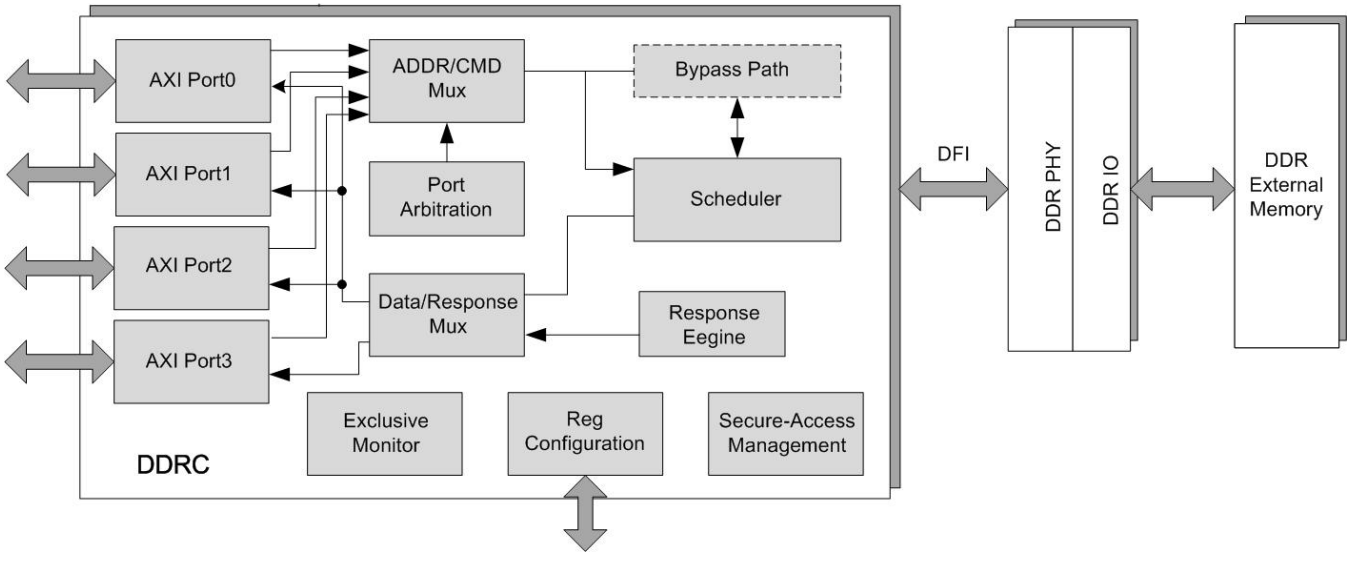
10.3.1 Introduction

The DDR controller features a cutting-edge design that optimizes DRAM access by rearranging requests into an efficient order, rather than processing them in their original sequence. It uses re-ordering buffers (ROBs) to reorganize accesses to the SRAM device for improving performance, while maintaining the original transaction order for requests with the same ID on the AXI interface.

Additionally, the DDR controller includes a unified write pool to temporarily store write transactions. Such write pool minimizes write latency and reduces the performance penalty due to switching between read and write operation at the DRAM interface. With a built-in heuristic write buffer control and user-programmable write buffer control, the DDR controller dynamically balances read and write operation performance in real-time.

The DDR controller is also designed to support AMBA AXI4 bus protocols. It is fully scalable and supports up to 4 AXI ports.

The architecture of the DDR controller interface is depicted below.



10.3.2 Features

- Priority-based arbitration with a starvation prevention scheme
- Merge of write operations to the same address by using a write buffer to reduce DDR write operation traffic
- Direct forward of read operations of the write buffer to the ROB without accessing DDR
- Two levels dynamic scheduling with bandwidth guarantee
- Support for power-saving features, including active/pre-charge power-off and self-refresh, with control options available automatically (via idle timer), manually (through registers) or externally (via dedicated ports)
- Support for dynamic frequency change
- Support for JEDEC compliant LPDDR3 and LPDDR4 devices
- Support for DRAM size from 64MB to 16GB
- One DRAM channel with a x32 DDR PHY, programmable by software to support x32, x16 or x8 data width
- Support for x16, x32 DRAM devices (1 DQS per 8 DQ)
- Support for up to 2 Chip Select (CS) or Rank per channel
- Support for up to 8 banks per CS for LPDDRx
- Each CS can be mapped to a different starting address
- Each CS can be programmed for 8MB to 16GB
- DRAM banks can be kept open after access (no auto-pre-charge)
- Support for burst length of 8 and 16 for the applicable DDR type
- Programmable address order
- Flexible bank placement between CS and data width
- Implementation of memory controller performance counters
- Global monitors for RISC-V exclusive load/store access
- Secure access management for DDR transactions
- Frequency change register update: implementation of a register table for hardware-triggered sequence update after frequency changes

10.3.3 Functional Description

10.3.3.1 DDR Controller Ports Allocation

The DDR Controller supports 4 AXI slave ports with arbitration among them. Details about the port connections to the DDR Controller are tabled below.

Port0	Port1	Port2	Port3
RISCV X60 Clusters and GPU	GMAC0, GMAC1 and AXI Fabric1 Port	VPU and PCIE PortA/B/C	Display and ISP

10.3.3.2 Dynamic Scheduling

The DDR Controller uses a priority-based arbitration with a starvation prevention scheme to manage the latency of important accesses.

The DDR Controller schedules transactions based on

- Current DRAM page status
- Transaction priority, determined by the AxQoS[2:0] signals on the AXI interface

The AxQoS are decoded as

- b100: CRITICAL
- b011: TOP
- b010: HIGH
- b001: MED (Guaranteed Bandwidth Feature)
- b000: LOW

The DDR controller can interleave between banks to achieve 100% bus utilization for streaming transactions to different banks, effectively hiding the precharge and activate overheads between read and write commands.

A CRITICAL priority transaction is always served as quickly as possible. It does not check the DRAM page status and is the next transaction to be processed. However, if there are multiple CRITICAL priority transactions, the page status will be considered.

Transactions are served when no higher priority transactions are pending. If lower priority transactions are directed to an already open page, the current higher priority transaction will close that page first.

10.3.3.3 Starvation Prevention

An aging mechanism increases the priority of a transaction when its counter expires to prevent starvation. The starvation counter values for the reading pool and scheduling pool can be configured in the following registers:

- RPP Starvation Control Register (RPP_Starvation_Control)
- Spool Control Register (Spool_Control)

10.3.3.4 Scheduling Algorithm

Scheduling algorithm is implemented in DDR Controller to grant different weights to the following events:

- DRAM pages status regarding to this transaction, it depends on if
  - Page is open
  - Page is closed
  - Page is missing (it is open on another page)
- CS
- Same CS
- Different CS

The scheduler allows out-of-order DRAM execution by re-ordering of data to fit with AXI ordering model. The scheduler would make a decision based on both the queue status and bank status to utilize the DDR bus bandwidth at most.

10.3.3.5 Ports Arbitration

When multi-ports configuration is used, DDR controller uses AXI QoS signal to arbitrate between different AXI ports. If ports have the same QoS priority, then a round-robin arbitration scheme is used.

10.3.3.6 Power-Down Mode

Power-down mode deactivates most I/O buffers on the DRAM module, in particular:

- **Precharge Power-Down**
  Occurs when all banks are closed. If any banks are open, a precharge-all command is issued before entering this mode.
- **Active Power-Down**
  Occurs when at least one bank remains open. If all banks are already closed, an active power-down request behaves as a precharge power-down.

Precharge power-down saves more power than active power-down but closes all banks upon exit, which may reduce performance.

The power-down duration is limited by the auto-refresh interval. If no data requests are pending, the DDR Memory Controller will wake up the DRAM, perform an auto-refresh, and re-enter power-down mode.

For DDR3, there is an option to select fast-exit or slow-exit power-down, which determines whether the DRAM DLL remains active or is frozen during power-down.

10.3.3.7 Self-Refresh Mode

In self-refresh mode, the DRAM automatically refreshes its banks, in particular:

- Within a refresh period, precharge power-down and self-refresh provide similar power savings, but self-refresh has a longer exit delay. Over extended periods, self-refresh is more efficient since precharge power-down requires periodic wake-ups and auto-refresh commands.
- LPDDR and DDR3 support Partial-Array Self-Refresh (PASR), which controls how many banks are refreshed during self-refresh. Each chip-select has a separate PASR configuration in the SDRAM Configuration Register.
- Since PASR allows certain banks to remain unrefreshed, software must track memory usage to prevent data loss in unrefreshed banks.

10.3.3.8 Automatic Clock-Stop

LPDDR allows the DRAM clock to be stopped when there is no pending DRAM operations.

Automatic Clock Stop (ACS) operates separately and can be used alongside other power-saving modes.

10.3.3.9 Power Saving Control

The DDR Memory Controller supports all power-saving features such as clock stop, pre-charge power-down, active power-down, self-refresh and deep power-down.

Power-saving commands can be issued in three ways as follows:

- **External control**
- **Manual issue**
- **Automatic issue**

Power-saving commands can be manually triggered through the User-Initiated Command Register, with each request executed only once. A new request will wake the Memory Controller from its current power-saving mode.

The Memory Controller can also enter a power-saving state (active or precharge power-down) automatically after a set number of idle cycles. However, deep power-down must be entered manually since it shuts off power to the memory array, causing a reset.

10.3.3.10 Dynamic Frequency Change (DFC)

DDR frequency change can be achieved by either hardware (boundary pin handshake) or software (AHB programming), in particular:

- **Hardware DFC**
  - The system must preload a DFC table (a predefined frequency sequence) into SRAM. Each table corresponds to a specific target frequency (e.g. Table 1 for 800MHz, Table 2 for 300MHz).
  - To change frequency, the SoC PMU drives the boundary pin and selects a DFC table. The Memory Controller then executes the DFC sequence automatically and signals the PMU when it is safe to switch clocks and when the process is complete.
- **Software DFC**
  - The DFC sequence is initiated by programming Memory Controller registers, without using SRAM
  - Software must execute the sequence step by step and poll for completion

10.4 QSPI Flash

10.4.1 Introduction

Quad-SPI acts as an interface to external serial flash devices with up to four bidirectional data lines.

10.4.2 Features

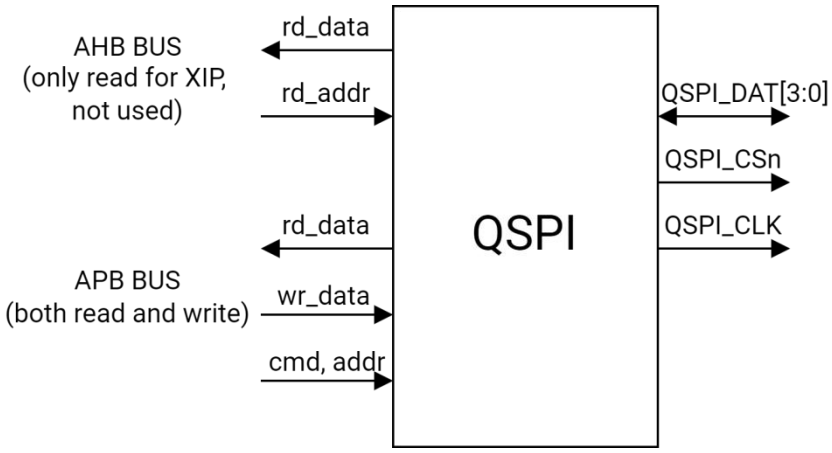
- Flexible sequence engine to support various flash vendor devices
- Single, dual and quad mode operation
- DMA supports reading RX buffer data via AMBA AHB bus (64-bit width interface) or IP register space (32-bit access), and filling TX buffer via IP register space (32-bit access)
- Configurable DMA inner loop size

- Fifteen interrupt conditions
- Memory-mapped read access for connected flash devices
- Programmable sequence engine for future command/protocol changes, and able to support all existing vendor commands and operations
- Support for all types of addressing
- Support for standard SPI, Fast, Dual, Dual I/O, Quad, Quad I/O mode
- Operation up to 104MHz clock frequency

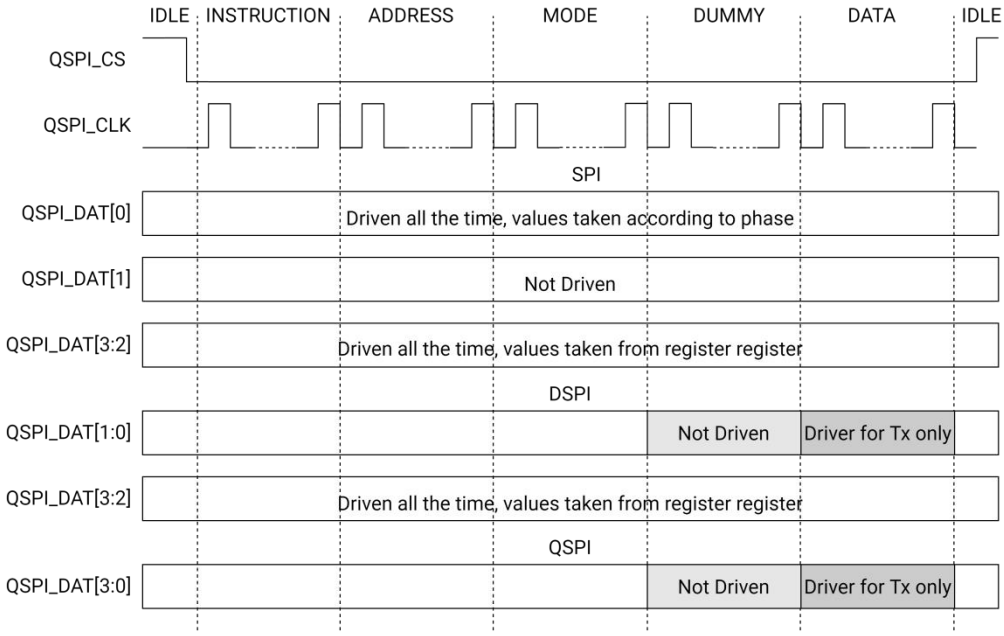
10.4.3 Functional Description

The QSPI block diagram is depicted below, where

- AHB BUS is used for XIP transfer (not used in current design)
- APB BUS is used to configure registers and write/read/erase external serial flash



The different phases of the serial flash access scheme are depicted below.



The different phases and the I/O driving characteristics of the QuadSPI module are characterized in the following way:

- **IDLE**
  Serial flash device is not selected, and there is no interaction. All QSPI_DATx signals remain undriven.
- **INSTRUCTION**
  Serial flash device is selected. And the instruction is sent to the serial flash device.
- **ADDRESS**
  Serial Flash Address is sent to the device.

  Note. This phase is not applicable for all SFM Commands.
- **MODE**
  Mode bytes are sent to the serial flash device, and all QSPI_DATx signals are driven.

  Note. This phase is not applicable for all SFM Commands.
- **DUMMY**
  Dummy clocks are provided to the serial flash device.

  Note. This phase is not applicable for all SFM Commands.
- **DATA**
  Serial flash data is sent to or received from the serial flash device.
- **Note.** This phase is not applicable for all SFM Commands.

QSPI_CS and QSPI_CLK signals are driven permanently throughout all phases.

10.5 SD/eMMC

10.5.1 Introduction

The SD/eMMC Host Controller is a hardware block that acts as a host of the SD/eMMC interface to transfer data between MMC, SDIO, SD cards and the internal bus master.

10.5.2 Features

- Compliance with MMC/eMMC 5.1 specification with support for socket 3 (eMMC)
- Support for PIO mode and SDMA mode data transfer
- Support for ADMA1 and ADMA 2 (64-bit addressing) data transfer
- Support for 1-bit/4-bit SD memory and SDIO
- Support for 1-bit/8-bit MMC and CE-ATA cards
- Support for SPI mode for eMMC card
- The following modes are supported for the eMMC device:
  - Legacy (up to 26MB/s, 1.8V signal)
  - High-speed SDR (up to 52MB/s, 1.8V signal)

- High-speed DDR (up to 52MB/s, 1.8V signal)
- HS200 (up to 200MB/s, 1.8V signal)
- HS400 (up to 400MB/s, 1.8V signal)

- The following modes are supported for the SD device:
  - Default Speed (up to 12.5MB/s, 3.3V signal)
  - High Speed (up to 25MB/s, 3.3V signal)
  - SDR12 (up to 25 MHz, 1.8V signal)
  - SDR25 (up to 50 MHz, 1.8V signal)
  - SDR50 (up to 100 MHz, 1.8V signal)
  - SDR104 (up to 208 MHz, 1.8V signal)
  - DDR50 (up to 50 MHz, 1.8V signal)
  - Support for read-wait control in SDIO cards
  - Support for suspend resume in SDIO cards
  - Hardware generation/checking of CRC on all command and data transaction on the card bus
  - Card insertion/removal detection for sockets 1 (SD)

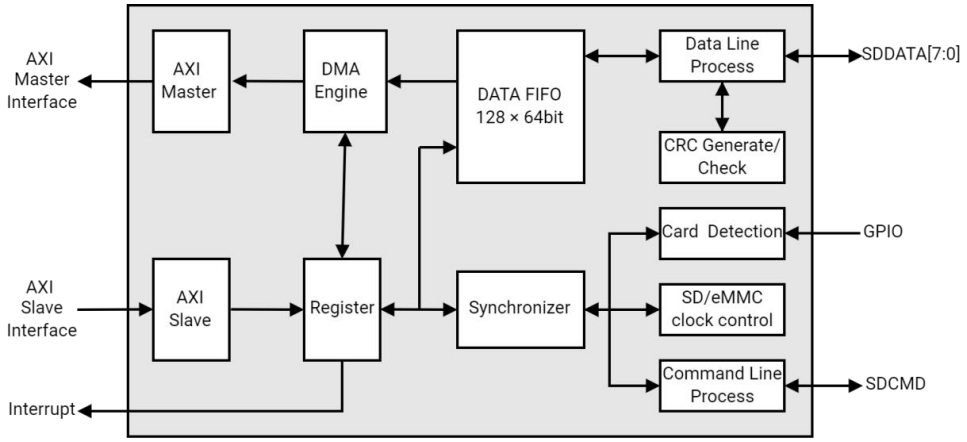
Note. For more information about AXI Fabric Specification, please check the description below:

- SD AXI master always generates an INCR (AWBURST/ARBURST = 0x1) burst and 8-bytes burst size (AWSIZE/ARSIZE = 0x3)
- SD AXI master bursts are normal, non-secure data accesses (AWPROT/ARPROT = 0x2)
- SD AXI master never generates atomic accesses (AWLOCK/ARLOCK = 0x0)
- SD AXI slave does not support exclusive, protected or atomic accesses. The response signaling is always “OKAY” (RRESP/BRESP = 0x0) or “DECERR” (RRESP/BRESP = 0x2). The “DECERR” response is signaled if an address is accessed where no registers exist.

10.5.3 Functional Description

10.5.3.1 Block Diagram

The architecture of SD/eMMC Host Controller is depicted below.



To be highlighted:

- The system fabric connects through two AXI interfaces:
  - AXI Slave
  - AXI Master
- AHB Slave is used for configuring registers and handling read/write operations in PIO mode
- AHB Master manages data transmission in SDMA/ADMA mode
- A 128 × 64-bit FIFO stores up to two 512-byte packets for data buffering
- Configuration registers synchronize with the SD/eMMC clock domain via a synchronizer to configure the clock, data width, etc.
- Card insertion/removal is detected through GPIO, triggering an interrupt to notify the CPU

10.5.3.2 SD/eMMC Bus Protocol Description

Communication over the SD bus is based on commands and data bit streams that are initiated by a start bit and terminated by a stop bit as follows:

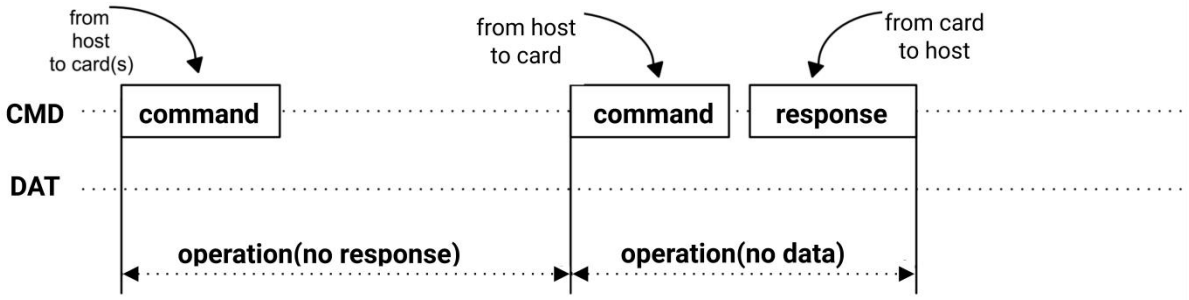
- **Command**
  A command is a token that starts an operation. It is sent from the host to the card(s) and is transferred serially via the SDCMD line (1 bit).

- **Command Response**
  A command response is a token that is sent from an addressed card to the host as a reply to a previously received host command. It is transferred serially via the SDCMD line since the SDCMD line is a bidirectional signal.

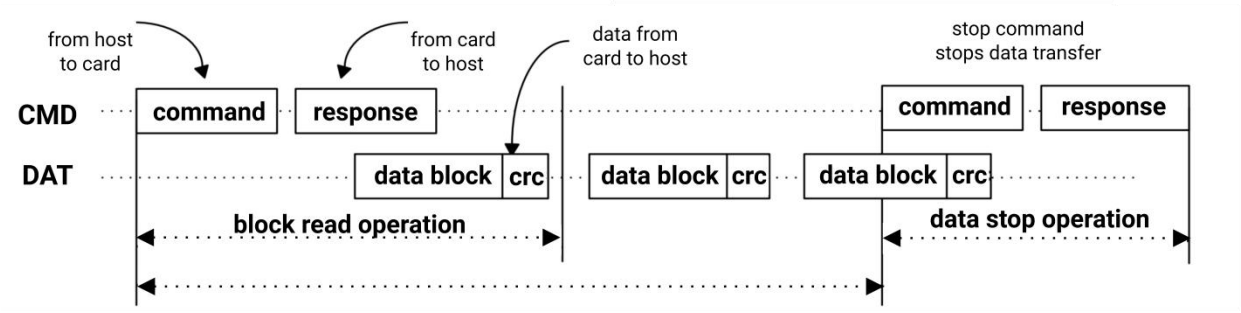
- **Data**
  There are 4 data lines. Data can be transferred from the card to the host or vice versa. Data is transferred via Data line, and they are bidirectional signals. In 1-bit mode, SDDATA[0] is used, and the other is in Z-state.

The basic operations of the SD/eMMC cards are depicted below.

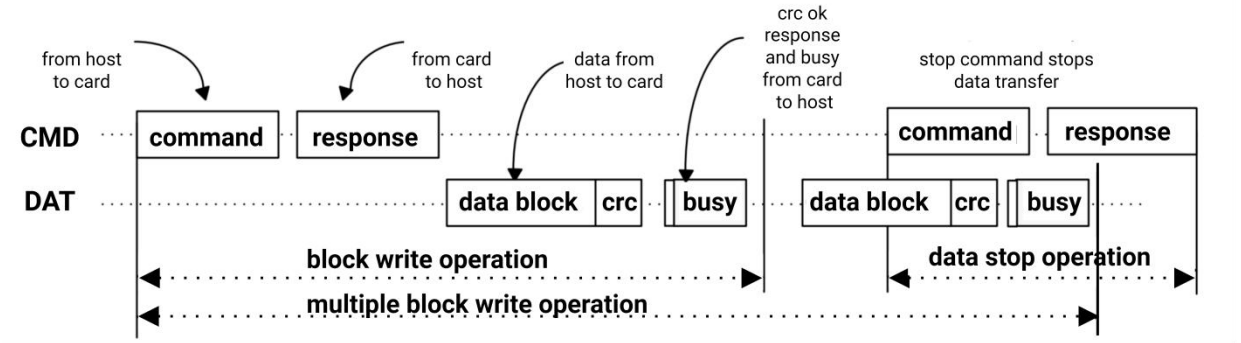
[No Response & No Data Operations]



[Multiple Block Read Operation]



[Multiple Block Write Operation]



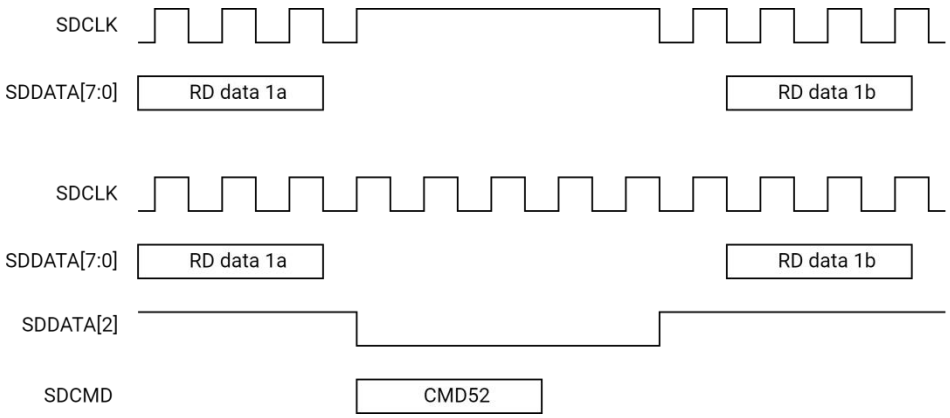
10.5.3.3 Special Bus Transactions

10.5.3.3.1 Read Wait Command

During a Read operation, if the host cannot accept more data, it usually stops the clock to pause the Read data output from the card. However, this also prevents the host from issuing any commands.

To avoid this limitation, a Read Wait command allows the host to pause the Read data while keeping the clock running. This command is issued after the data block ends, ensuring that the host can manage data flow without completely stopping communication.

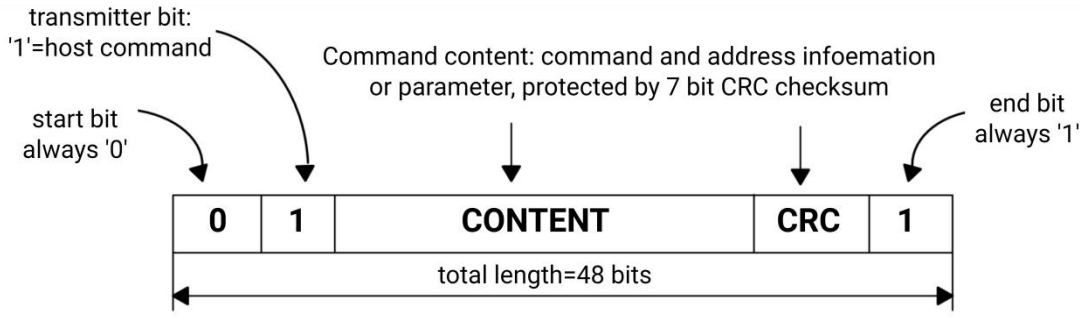
The Read Wait Controlled by Stopping Clock is depicted below.



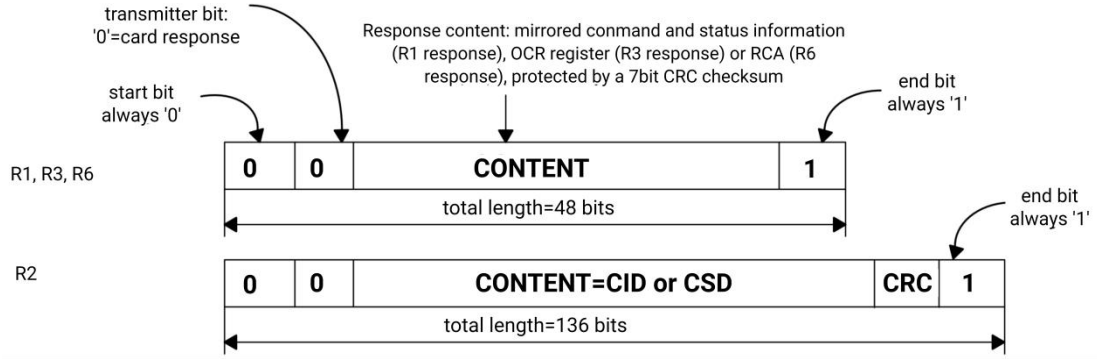
10.5.3.3.2 Packet Format

The formats of commands from the host and responses from cards are depicted below.

[Command Token Format]



[Response Token Format]



10.5.3.3.3 Sequences of Host & Card Interaction

SD cards are connected to the host with a dedicated interface, such as its own SDCLK, SDCMD, or SDDATA[3:0] lines.

After a **power-on reset, software reset or new card insertion**, the host and SD cards go through two phases as follows:

- **Phase 1 - Card Identification Phase:**
  - The host looks for new cards on the bus
  - While in this phase, the host resets all the cards that are in card identification mode. Any card already identified will not be reset.
  - Then the host:
    - Sends the command to validate operation voltage range
    - Identifies each card and requests its Relative Card Address (RCA)
    - Communicates with each card separately using the **SDCMD line** (in the case of SD memory)
  - All data communication in this phase uses the SDCMD line only

- **Phase 2- Data Transfer Phase:**
  - Once all cards are identified, the host enters this phase that is ready to transfer data
  - After the host driver sets up the Host Controller, it starts the transfer by writing to the Command Register which is written last
  - For R1b commands, the Command Inhibit will stay set as long as the busy signal is held low
  - CMD_ COMPLETE will then be set when Command Inhibit changes from 1 to 0

Note. For more information, please refer to SD Host Controller Simplified Specification V3.00

10.5.3.4 Card Detection

The SD/eMMC Host Controller supports card detection (insertion/removal) via the GPIO toggle with software operation.

The card detection features can be accessed though the Normal Interrupt Status Enable bits. When enabled, an interrupt is generated if a logic change is detected on the card-detect switch input. To avoid false triggers, this logic change is debounced before generating an interrupt.

10.5.3.5 SPI Mode

SPI mode is enabled by setting the <spi_en> field in the Legacy Register (0x10C). The rest of the flow is similar to the SD flow except for error interrupts.

If an error occurs in SPI mode, then the <spi_err> field in the Error Interrupt Status Register will be set. The cause of this error can be identified in the <spi_err_token> field in the Legacy Register.

10.5.4 Register Description

The base addresses of SD/eMMC Host Controller Registers are tabled below.

Name	Address
SD1_BASE(SD Card)	0xD4280000
SD2_BASE(SDIO)	0xD4280800
SD3_BASE(eMMC)	0xD4281000

10.5.4.1 SD_SYS_ADDR REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	DMA_ ADDR	R/W	0x0000	- DMA Address High: contains 16 MSb of DMA system buffer starting byte address.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
	_H			- This register is used with the Auto Cmd 23 to set a 32-bit block count value to the argument of cmd23. This register holds the upper 16bits of the cmd23 argument.
15:0	DMA_ADDR_L	R/W	0x0000	- DMA Address Low: contains 16 LSb of DMA system buffer starting byte address. - This register is used with the Auto Cmd 23 to set a 32-bit block count value to the argument of cmd23. This register holds the lower 16bits of the cmd23 argument.

10.5.4.2 SD_BLOCK_SIZE_CNT REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	BLOCK_COUNT	R/W	0x0000	Block Count The Host Controller decrements the block count after each block transfer. 0x1 = 1 block ... 0xFFFF = 65535 blocks The current value of block count is reflected in the Current Block Count Register.
15	RSVD	R	0	Reserved for future use
14:12	HOST_DMA_BDRY	R/W	0x0	Host DMA Buffer Boundary This field specifies the host memory buffer boundary. If this boundary is crossed, an interrupt (dma_int) is generated. This interrupt is reflected in <Tx Ready> field of the Normal Interrupt Status Register. 0x0: 4 KB 0x1: 8 KB 0x2: 16 KB 0x3: 32 KB 0x4: 64 KB 0x5: 128 KB 0x6: 256 KB 0x7: 512 KB
11:0	BLOCK_SIZE	R/W	0x000	Block Size

10.5.4.3 SD_ARG REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	ARG_H	R/W	0x0000	Argument High 16 MSb of Command Argument This value is inserted into 48 bits command token bits[39:24].
15:0	ARG_L	R/W	0x0000	Argument Low 16 LSb of Command Argument This value is inserted into 48 bits command token bits[23:8].

10.5.4.4 SD_TRANSFER_MODE_CMD REGISTER

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	R	0	Reserved for future use
29:24	CMD_INDEX	R/W	0x00	Command Index These bits will be inserted into Command token bits[45:40]
23:22	CMD_TYPE	R/W	0x0	Command Type 0x0: Normal command 0x1: Suspend command 0x2: Resume command 0x3: Abort command
21	DATA_PRESENT	R/W	0x0	Data Present 1: Indicates that data is present and will be transferred using the MMC1_DAT[3:0] line. 0: Commands using only MMC1_CMD lines or commands with no data transfer but using busy signal on MMC1_DAT[0] line (for example, CMD 38)
20	CMD_INDEX_CHK_EN	R/W	0x0	Command Index Check Enable 1: The Host Controller checks the index field in the response to ensure it matches the command index. - If there is a mismatch, it is reported as a Command Index Error.
19	CMD_CRC_CHK_EN	R/W	0x0	Command CRC Check Enable 1: Host controller checks the CRC field in response. - If an error is detected, it is reported as a command CRC error. The number of bits checked by the CRC field value changes according to the length of response.
18	RSVD	R	0	Reserved for future use
17:16	RESP_TYPE	R/W	0x0	Response Type Select for SD/SD in SPI Modes For SD mode: 0x0: No response 0x1: Response length is 136 bits

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x2: Response length is 48 bits 0x3: Response length is 48 bits and check busy after response CRC field for R3 and R4 is expected to be all 1 bits. CRC Check should be disabled for these response types. For SD in SPI mode: 0x0: Response length is 8 bits 0x1: Response length is 16 bits 0x2: Response length is 40 bits 0x3: Reserved
15:6	RSVD	R	0	Reserved for future use
5	MULTI_BLK_SEL	R/W	0x0	Multiple Block Select This bit should be set to 1 only when multiple blocks are to be transferred.
4	TO_HOST_DIR	R/W	0x0	Data Transfer Direction Select This bit defines the direction of the MMC1_DAT[3:0] line data transfer. 1: Transfer data from the SD card to the SD Host Controller, 0: Used for all other commands.
3:2	AUTO_CMD_EN	R/W	0x0	Auto CMD Enable This field determines use of auto command functions. 0x0: Auto Command disabled 0x1: Auto CMD12 Enable 0x2: Auto CMD23 Enable 0x3: Reserved
1	BLK_CNT_EN	R/W	0x0	Block Count Enable This bit validates the value in the Block Count Register.
0	DMA_EN	R/W	0x0	DMA Enable If Programmed Input/Output (PIO) mode is required, this bit should be reset to 0.

10.5.4.5 SD_RESP_0 REGISTER

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RESP1	R	0x0000	Response 1 This register contains bits[39:24] of the response token.
15:0	RESP0	R	0x0000	Response 0 This register contains bits[23:8] of the response token.

10.5.4.6 SD_RESP_1 REGISTER

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RESP3	R	0x0000	Response 3 - For 48-bit response tokens: Not used. - For 136-bit response tokens: Stores bits [71:56] of the response token.
15:0	RESP2	R	0x0000	Response 2 - For 48-bit response tokens: Not used. - For 136-bit response tokens: Stores bits [55:40] of the response token.

10.5.4.7 SD_RESP_2 REGISTER

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RESP5	R	0x0000	Response 5 - For 48-bit response tokens: Not used. - For 136-bit response tokens: Stores bits [103:88] of the response token.
15:0	RESP4	R	0x0000	Response 4 - For 48-bit response tokens: Not used. - For 136-bit response tokens: Stores bits [87:72] of the response token.

10.5.4.8 SD_RESP_3 REGISTER

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RESP7	R	0x0000	Response 7 - For 48-bit response tokens: Not used. - For 136-bit response tokens: Stores bits [127:120] of the response token. - For Auto CMD12 responses: Stores bits [39:24] of the response token.
15:0	RESP6	R	0x0000	Response 6 - For 48-bit response tokens: Not used. - For 136-bit response tokens: Stores bits [119:104] of the response token. - For Auto CMD12 responses: Stores bits [23:8] of the response token.

10.5.4.9 SD_BUFFER_DATA_PORT REGISTER

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	BUF_DATA	R/W	0x0	Buffer Data

10.5.4.10 SD_PRESENT_STATE_1 REGISTER

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:25	RSVD	R	0	Reserved for future use
24	CMD_LEVEL	R	0x1	MMC1_CMD Line Signal Level This status is used to check the MMC1_CMD line level to recover from errors and for debugging.
23:20	DAT_LEVEL	R	0xF	MMC1_DAT[3:0] Line Signal Level This status is used to check the MMC1_DAT[3:0] line level to recover from errors and for debugging. This is especially useful in detecting the busy signal level from MMC1_DAT[0].
19	WRITE_PROT	R	0x0	Write Protect This field reflects the position of the write_protect latch on the SD card. This field should be ignored if there is no such feature being provided by the card in use.
18	CARD_DET	R	0x0	Card Detect This field reflects the value of the MMC1_CD pin. 0: Card is not detected 1: Card is detected Note: This field is only used for testing.
17	CARD_STABLE	R	0x0	Card Stable It indicates the debounced value of the card present condition. 0: Card is unstable 1: Card is stable Note: This field is only used for testing.
16	CARD_INSERTED	R	0x0	Card Inserted This field indicates whether an SD card is present or not:. 0: Card is not inserted 1: Card is inserted
15:12	RSVD	R	0	Reserved for future use

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
11	BUFFER_RD_EN	R	0x0	Buffer Read Enable This field changes from 0x0 to 0x1 when block data is ready in the buffer, and from 0x1 to 0x0 when all the block data is read from the buffer.
10	BUFFER_WR_EN	R	0x1	Buffer Write Enable This field changes from 0x0 to 0x1 when block data can be written to the buffer. So if this bit is set to 0x1, the entire block can be written to the buffer. This field changes from 0x1 to 0x0 when all the block data is written to the buffer.
9	RX_ACTIVE	R	0x0	Rx Active This field indicates read transfer is active. 1: Active 0: Inactive
8	TX_ACTIVE	R	0x0	Tx Active Indicates write transfer is active. 0: No valid write data exists in the Host Controller 1: active
7:4	RSVD	R	0	Reserved for future use
3	RETUNING_REQ	R	0x0	Re-Tuning Request This field provides the status of the sampling clock. 0: Fixed or well tuned sampling clock 1: Sampling clock needs re-tuning
2	_DAT_ACTIVE	R	0x0	Data Line Active This field provides the status of the data line. 0: Data line is free 1: Data line is busy
1	CMD_INHIBIT_DAT	R	0x0	Command Inhibit Data This field provides the host driver status for issuing data commands. 0: Data command can be issued 1: Data command cannot be issued
0	CMD_INHIBIT_CMD	R	0x0	Command Inhibit Command 0: The MMC1_CMD line is available for issuing a new command. - The host controller can issue a command using MMC1_CMD line. - This bit is cleared (set to 0) when the command response is received from the device. - If the <Command Inhibit Data> field is set to 1, commands using only the MMC1_CMD line can

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
				still be issued if this bit is 0. 1: The MMC1_CMD line is currently in use or unavailable. - This bit is set after the command register is written, indicating that a command is in progress. - It remains set until the command response is received. When the bit changes from 1 to 0 (indicating that the command response has been received), a command complete interrupt is generated in the Normal Interrupt Status Register. If the host controller cannot issue a command due to a command conflict error (e.g., attempting to issue a new command while the previous command is still in progress), this bit remains set to 1.

10.5.4.11 SD_HOST_CTRL REGISTER

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:27	RSVD	R	0	Reserved for future use
26	W_REMOVAL	R/W	0x0	Wakeup on Card Removal 1: Enable wakeup event on card removal detection 0: No wakeup event
25	W_INSERTION	R/W	0x0	Wakeup on Card Insertion 1: Enable wakeup event on card insertion detection 0: No wakeup event
24	W_CARD_INT	R/W	0x0	Wakeup on Card Interrupt 1: Enable wakeup event on card interrupt detection 0: No wakeup event
23:20	RSVD	R	0	Reserved for future use
19	INT_BLK_GAP	R/W	0x0	Block Gap Interrupt This field is only valid for 4-bit mode. 1: Enables interrupt detection at block gap for multiple block transfers 0: Disables interrupt detection at block gap for multiple block transfers
18	RD_WAIT_CTL	R/W	0x0	Read Wait Control - If the card supports read wait, set this bit to enable use of

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
				the read wait protocol to stop read data using the MMC1_DAT[2] line by Host hardware. Otherwise, the Host Controller has to stop the SD clock from holding read data. - When the Host driver detects a card insertion, it will set this bit according to the CCCR of the SDIO card. - This field is only considered during the block gap phase. Within a block, hardware will stall the clock top stop read data if the host cannot accept any more data (e.g., due to FIFO being full, etc.) - When this field is cleared by software, operation continues as usual. - During read wait, software can issue a different commands for different operation as long as it does not require MMC1_DAT[3:0] lines. - To resume the read operation after waiting, the software must clear this bit (set it to 0)
17	CONT_REQ	R/WAC	0x0	Continue Request - This field is used to restart a transaction which was stopped using the <Stop At Block Gap Request>. - To cancel stop at the block gap, set the <Stop At Block Gap Request> field to 0 and set this field to 1 to restart the transfer. - The Host Controller automatically clears this field in either of the following cases: 1. For a read transaction, when the MMC1_DAT[3:0] Line Active changes from 0 to 1, indicating a read transaction restarts. 2. For a write transaction, when the Write Transfer Active changes from 0 to 1, indicating the write transaction restarts. As a result, it is not necessary for the Host driver to set this bit to 0. - If <Stop At Block Gap Request> is set to 1, any write to this bit is ignored.
16	STOP_AT_BLOCK_GAP_REQ	R/W	0x0	Stop at Block Gap Request - This field is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. - The Host driver will leave this bit set to 1 until transfer completion is indicated (when transfer complete is set to 1) - Clearing both this field and the <Continue Request> field will not cause the transaction to restart. - For read transactions, Read Wait can be used to stop the transaction at the block gap. - For write transactions, the host controller will stop the

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
				clock at the block gap request. - For read transactions, if Read Wait Control is set to 0, the Host controller will stop the clock; otherwise, the Host controller will issue a Read Wait command to stop read data.
15:12	RSVD	R	0	Reserved for future use
11:9	SD_BUS_VLT	R/W	0x0	SD Bus Voltage This field reflects the voltage at operating conditions 0x7: 3.3V 0x6: 3.0V 0x5: 1.8V 0x0 to 0x4: Reserved
8	SD_BUS_POWER	R/W	0x0	SD Bus Power This field controls the power going out to the SD card. The field will be cleared if: - The sd_bus_vlt does not match the voltage support in Capabilities Register 1, or - A card removal state is detected.
7	CARD_DET_S	R/W	0x0	Card Detect Signal Selection This field selects the source for card detection. 0: Card detect input pin 1: Card detect test level (for debugging purposes only) - When the source for card detection is switched, the interrupt should be disabled during the switching period by clearing the Normal Interrupt Status Enable Register in order to mask unexpected interrupts being caused by the glitch. - This signal should be disabled via the Normal Interrupt Status Enable Register during debounce period.
6	CARD_DET_L	R/W	0x0	Card Detect Test Level 1: Card inserted 0: No card inserted
5	EX_DATA_WIDTH	R/W	0x0	This bit controls the 8-bit mode. 0x0: Data width for bus mode is determined by <DATA_WIDTH> 0x1: 8-bit data width.
4:3	DMA_SEL	R/W	0x0	DMA Select One of the supported DMA modes is selected. The host driver checks support for DMA modes using the Capabilities Register 1. Use of the selected DMA is determined by the <DMA Enable> field in the Transfer Mode Register. 0x0: SDMA

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x1: ADMA 1 0x2: 32-bit address ADMA2 0x3: Reserved
2	HI_SPEED_EN	R/W	0x0	Extend Data Output Enable 0: Normal 1: MMC1_CMD and MMC1_DAT[3:0] are driven from rising edge of clock
1	DATA_WIDTH	R/W	0x0	Data Width 1: 4-bit data mode 0: 1-bit data mode, using only MMC1_DAT[0] Refer to Bit [5] EX_DATA_WIDTH for 8-bit mode support.
0	LED_CTRL	R/W	0x0	LED Control 1: LED on 0: LED off

10.5.4.12 SD_CLOCK_CTRL REGISTER

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:27	RSVD	R	0	Reserved for future use
26	SW_RST_DAT	R/WAC	0x0	Soft Reset for Data Port of Logic
25	SW_RST_CMD	R/WAC	0x0	Soft Reset for Command Part of Logic
24	SW_RST_ALL	R/WAC	0x0	Software Reset for All This reset affects the status, state machine, and FIFOs synchronously. This field also resets all private registers.
23:20	RSVD	R	0	Reserved for future use
19:16	TIMEOUT_VALUE	R/W	0x0	Timeout Value Determines the interval by which MMC1_DAT[3:0] line timeouts are detected. This timeout is initiated in the following cases: - For read transaction, this timeout is about waiting for data from cards. It refers to the timing value in the SD specification, which specifies the maximum timing from read command to read data (i.e., card data access time); - For write transaction, this timeout is about waiting for data from AXI slave, AXI Master, or processor, or waiting

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
				for CRC status of write block. - Timeout Calculation: 1. 0x0: SDCLK x 2^13 2. 0x1: SDCLK x 2^14 3. ... 4. 0xE: SDCLK x 2^27 For example, 1. If sd_clk frequency = 200 MHz (base value), then the timeout base = 50 MHz (period = 20 ns). - For 0xE setting: Timeout = 2^27 * 20 ns ≈ 2.684 seconds. 2. If sd_clk frequency = 50 MHz (base value divided by 4), then the timeout base = 12.5 MHz (period = 80 ns). - For 0xE setting: Timeout = 2^27 * 80 ns ≈ 10.73 seconds. 0xF is Reserved for future use. For other transactions, please refer to the SD specification for more information on these fixed values.
15:8	SD_FREQ_SEL_LO	R/W	0x00	SDCLK Frequency Select Lower bits This field, together with the <SD_FREQ_SEL_HI> field, defines the clock divider value to be used by the host controller. The final value for the SD clock frequency divider is determined by combining both fields as follows: SD_FREQ_SEL = {SD_FREQ_SEL_HI[1:0],SD_FREQ_SEL_LO[7:0]} The selected value is multiplied by 2 to calculate the actual divide value. The possible SD clock frequency settings are: - SD_FREQ_SEL = 0x00: Base clock - SD_FREQ_SEL = 0x01: Divide by 2 of base clock - SD_FREQ_SEL = 0x02: Divide by 4 of base clock - SD_FREQ_SEL = 0x3: Divide by 6 of base clock - ... - SD_FREQ_SEL = 0x3FF: Divide by 2046 of base clock
7:6	SD_FREQ_SEL_HI	R/W	0x0	SDCLK Frequency Select Upper bits This field, together with <SD_FREQ_SEL_LO> defines the clock divider value to be used by the host controller. The final value for the SD clock frequency divider is determined by combining both fields as follows: SD_FREQ_SEL = {SD_FREQ_SEL_HI[1:0],SD_FREQ_SEL_LO[7:0]}.
5	CLK_GEN_SEL	R/W	0x0	Clock Generator Select This field is used to select the clock generator mode. 0x1: Programmable Clock Mode 0x0: Divided Clock mode

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
4:3	RSVD	R	0	Reserved for future use
2	SD_CLK_EN	R/W	0x0	SDCLK Clock Enable This bit controls the SDCLK to the card. Before using the card, this bit should be set during the initialization phase.
1	INT_CLK_STABLE	R	0x0	Internal Clock Stable This field is set to 1 once the controller detects that the internal clock is stable after setting of the <Internal Clock Enable> field.
0	INT_CLK_EN	R/W	0x0	Internal Clock Enable This field controls the SDCLK to the internal logic. 1: Enable 0: Disable

10.5.4.13 SD_NORMAL_INT_STATUS REGISTER

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CRC_STATUS_ERR	R/W1C	0x0	CRC Status Error It is set to 1 if there is an error in any of the following: - CRC status start bit - CRC status end bit - Boot ACK status These errors are returned from the card in write transaction
30	CPL_TIMEOUT_ERR	R/W1C	0x0	Command Completion Signal Timeout Error This field is applicable for CE-ATA mode only. 1: A command completion signal timeout occurred
29	AXI_RESP_ERR	R/W1C	0x0	AXI Bus Response Error 1: A response other than OKAY was received on the AXI bus.
28	SPI_ERR	R/W1C	0x0	SPI Mode Error 1: Error occurred in SPI mode for which cause can be determined by reading the <SPI Error Token> field in the SPI Mode Register 0: No error
27:26	RSVD	R	0	Reserved for future use
25	ADMA_ERR	R/W1C	0x0	ADMA (Advanced Direct Memory Access) Error This bit is set when the host controller detects any errors

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
				during an ADMA-based data transfer. The state of the ADMA at the time of the error is recorded in the ADMA Error Status Register. This interrupt is also triggered when the host controller detects invalid descriptor data. The <ADMA Error State> field in the ADMA Error Status Register indicates the state in which an error occurred. The host driver may find that a Valid bit is not set at the error descriptor. 1 = Error 0 = No error
24	AUTO_CMD12_ERR	R/W1C	0x0	Auto CMD12 Error occurs when detecting that one of the bits in Auto CMD12 Error Status Register has changed from 0 to 1.
23	CUR_LIMIT_ERR	R/W1C	0x0	Current Limit Error **Note** This feature is not supported currently, and this bit will always be read as 0.
22	RD_DATA_END_BIT_ERR	R/W1C	0x0	ReadData End Bit Error This bit is set to 1 when a 0 is detected at the end bit position of read data that uses the MMC1_DAT[3:0] line, or at the end bit position of the CRC status.
21	RD_DATA_CRC_ERR	R/W1C	0x0	Read Data CRC Error This bit is set to 1 when there is a CRC error detected in the read data transferred via the MMC1_DAT[3:0] line, or when the Write CRC status has a value other than 010.
20	DATA_TIMEOUT_ERR	R/W1C	0x0	Data Timeout Error This bit is set to 1 when a timeout error occurs during data transfer, which could happen in the following scenarios: - Busy timeout after write CRC status - Write CRC status timeout - Read data timeout
19	CMD_INDEX_ERR	R/W1C	0x0	Command Index Error 0: No command index error has occurred in the command response 1: Command index error has occurred in the command response
18	CMD_END_BIT_ERR	R/W1C	0x0	Command End Bit Error 0: Detection of end bit of a command response in 1 1: Detection of end bit of a command response is 0
17	CMD_CR	R/W1C	0x0	Command CRC Error

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
	C_ERR			This bit is set to 1 in two cases: - A CRC error is detected in the command response. - The host controller detects a conflict on the MMC1_CMD line while issuing a command. In this case, the host controller will abort the command by stopping the MMC1_CMD line, and the <Command Timeout Error> field will also be set to 1 to distinguish the MMC1_CMD line conflict.
16	CMD_TIMEOUT_ERR	R/W1C	0x0	Command Timeout Error 1: No response is returned within 64 SDCLK cycles from the end bit of the command
15	ERR_INT	R	0x0	Error Interrupt If any of bits in the Error Interrupt Status Register are set, then this bit is set.
14	CQ_INT	RC	0x0	Command Queuing Interrupt This interrupt is asserted when at least one of the bits in the CQIS register is set. This interrupt is cleared only when the source interrupt in the CQIS register is cleared.
13	RSVD	R	0	Reserved for future use
12	RETUNING_INT	R/W1C	0x0	Re-tuning Event Interrupt This status is set when the Re-Tuning Request in the <Present State Register> changes from 0x0 to 0x1. This indicates that the host controller is requesting the host driver to perform re-tuning starting from the next data transfer. The current data transfer can be completed without requiring re-tuning.
11	INT_C	R/W1C	0x0	This status is set when INT_C is enabled and INT_C# pin is in low level. Writing this bit to 0x1 does not clear this bit. It is cleared by resetting the INT_C interrupt factor. Refer to shared bus control register (SHARED_BUS_CTRL).
10	INT_B	R/W1C	0x0	This status is set when INT_B is enabled and INT_B# pin is in low level. Writing this bit to 0x1 does not clear this bit. It is cleared by resetting the INT_B interrupt factor. Refer to shared bus control register (SHARED_BUS_CTRL).
9	INT_A	R/W1C	0x0	This status is set if INT_A is enabled and INT_A# pin is in low level. Writing this bit to 0x1 does not clear this bit. It is cleared by resetting the INT_A interrupt factor. Refer to shared bus control register (SHARED_BUS_CTRL).
8	CARD_INT	R	0x0	Card Interrupt 1: Host controller detects an interrupt from the card

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
7	CARD_REM_INT	R/W1C	0x0	Card Removal Interrupt 1: Card removal event detected
6	CARD_INS_INT	R/W1C	0x0	Card Insertion Interrupt 1: Card insertion event detected
5	RX_RDY	R/W1C	0x0	Rx Ready This status is set when the <Buffer Read Enable> field in the Present State Register 1 changes from 0x0 to 0x1.
4	TX_RDY	R/W1C	0x1	Tx Ready This status is set when the <Buffer Write Enable> field in the Present State Register 1 changes from 0x0 to 0x1.
3	DMA_INT	R/W1C	0x0	DMA Interrupt This status is set when the Host Controller detects DMA crossing over the <Host DMA Buffer Boundary> field in the Block Size Register.
2	BLOCK_GAP_EVT	R/W1C	0x0	Block Gap Event This field is set when a read or write transaction is stopped at a block gap, but only if the <Stop At Block Gap Request> field in the Block Gap Control Register is set. - If the <Stop At Block Gap Request> field is not set to 1, this bit will remain 0.
1	XFER_COMPLETE	R/W1C	0x0	Transfer Complete This bit is set when a read/write transaction is completed. - For read transactions: This bit is set at the falling edge of Read Transfer Active Status, and it can occur in two cases: 1. Data transfer is completed as specified by the data length. 2. The data transfer is stopped at the block gap and completed after setting the <Stop At Block Gap Request> field in the SD Block Gap Control Register. - For write transactions: This bit is set at the falling edge of the MMC1_DAT[3:0] Line Active status, and it can occur in two cases: 1. Data transfer is completed as specified by the data length, and the busy signal is released. 2. The data transfer is stopped at the block gap and completed after setting the <Stop At Block Gap Request> field.
0	CMD_COMPLETE	R/W1C	0x0	Command Complete This bit is set when the end bit of the command response (Except Auto CMD12) is received. **Note.** Command Timeout Error has higher priority than

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
				Command complete.

10.5.4.14 SD_NORMAL_INT_STATUS_EN REGISTER

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CRC_STATUS_ERR_EN	R/W	0x0	CRC Status Error Enable 0: Disabled 1: Enabled
30	CPL_TIMEOUT_ERR_EN	R/W	0x0	CPL Timeout Error Enable 0: Disabled 1: Enabled
29	AXI_RESP_ERR_EN	R/W	0x0	AXI Response Error 0: Disabled 1: Enabled
28	SPI_ERR_EN	R/W	0x0	SPI Error Enable 0: Disabled 1: Enabled
27	RSVD	R	0	Reserved for future use
26	TUNING_ERR_EN	R/W	0x0	Tuning Error Enable 0: Disabled 1: Enabled
25	ADMA_ERR_EN	R/W	0x0	ADMA Error Enable 0: Disabled 1: Enabled
24	AUTO_CMD12_ERR_EN	R/W	0x0	Auto CMD12 Error Enable 0: Disabled 1: Enabled
23	CUR_LIM_ERR_EN	R/W	0x0	Current Limit Error Enable 0: Disabled 1: Enabled
22	RD_DATA_END_BIT_ERR_EN	R/W	0x0	Data End Bit Error Enable 0: Disabled 1: Enabled
21	RD_DATA_CRC_ERR_EN	R/W	0x0	Data CRC Error Enable 0: Disabled 1: Enabled
20	DATA_TIMEOUT_ERR_EN	R/W	0x0	Data Timeout Error Enable

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Disabled 1: Enabled
19	CMD_INDEX_ERR_EN	R/W	0x0	Command Index Error Enable 0: Disabled 1: Enabled
18	CMD_END_BIT_ERR_EN	R/W	0x0	Command End Bit Error Enable 0: Disabled 1: Enabled
17	CMD_CRC_ERR_EN	R/W	0x0	Command CRC Error Enable 0: Disabled 1: Enabled
16	CMD_TIMEOUT_ERR_EN	R/W	0x0	Command Timeout Error Enable 0: Disabled 1: Enabled
15	RSVD	R	0	Reserved for future use
14	CQ_STATUS_EN	R/W	0x0	Command Queuing Status Enable 0: Disabled 1: Enabled
13	RSVD	R	0	Reserved for future use
12	RETUNE_INT_EN	R/W	0x0	Re_tuning Interrupt Enable 0: Disabled 1: Enabled
11	INT_C_INT_EN	R/W	0x0	INT_C Enable 0: Disabled 1: Enabled
10	INT_B_INT_EN	R/W	0x0	INT_B Enable 0: Disabled 1: Enabled
9	INT_A_INT_EN	R/W	0x0	INT_A Enable 0: Disabled 1: Enabled
8	CARD_INT_EN	R/W	0x0	Card Interrupt Enable 0: Disabled 1: Enabled
7	CARD_REM_EN	R/W	0x0	Card Removal Status Enable 0: Disabled 1: Enabled
6	CARD_INS_EN	R/W	0x0	Card Insertion Status Enable 0: Disabled

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
				1: Enabled
5	RD_RDY_EN	R/W	0x0	Buffer Read Ready Enable 0: Disabled 1: Enabled
4	TX_RDY_EN	R/W	0x0	Buffer Write Ready Enable 0: Disabled 1: Enabled
3	DMA_INT_EN	R/W	0x0	DMA Interrupt Enable 0: Disabled 1: Enabled
2	BLOCK_GAP_EVT_EN	R/W	0x0	Block Gap Event Enable 0: Disabled 1: Enabled
1	XFER_COMPLETE_EN	R/W	0x0	Transfer Complete Enable 0: Disabled 1: Enabled
0	CMD_COMPLETE_EN	R/W	0x0	Command Complete Enable 0: Disabled 1: Enabled

10.5.4.15 SD_NORMAL_INT_STATUS_INT_EN REGISTER

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CRC_STATUS_ERR_INT_EN	R/W	0x0	CRC Status Error Interrupt Enable 0: Disabled 1: Enabled
30	CPL_TIMEOUT_ERR_INT_EN	R/W	0x0	CPL Timeout Error Interrupt Enable 0: Disabled 1: Enabled
29	AXI_RESP_ERR_INT_EN	R/W	0x0	AXI Response Error Interrupt Enable 0: Disabled 1: Enabled
28	SPI_ERR_INT_EN	R/W	0x0	SPI Error Interrupt Enable 0: Disabled 1: Enabled
27	RSVD	R	0	Reserved for future use
26	TUNE_ERR_INT_EN	R/W	0x0	Tuning Error Interrupt Enable

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Disabled 1: Enabled
25	ADMA_ERR_INT_EN	R/W	0x0	ADMA Error Interrupt Enable 0: Disabled 1: Enabled
24	AUTO_CMD12_ERR_INT_EN	R/W	0x0	Auto CMD12 Error Interrupt Enable 0: Disabled 1: Enabled
23	CUR_LIM_ERR_INT_EN	R/W	0x0	Current Limit Error Interrupt Enable 0: Disabled 1: Enabled
22	RD_DATA_END_BIT_ERR_INT_EN	R/W	0x0	Data End Bit Error Interrupt Enable 0: Disabled 1: Enabled
21	RD_DATA_CRC_ERR_INT_EN	R/W	0x0	Data CRC Error Interrupt Enable 0: Disabled 1: Enabled
20	DATA_TIMEOUT_ERR_INT_EN	R/W	0x0	Data Timeout Error Interrupt Enable 0: Disabled 1: Enabled
19	CMD_INDEX_ERR_INT_EN	R/W	0x0	Command Index Error Interrupt Enable 0: Disabled 1: Enabled
18	CMD_END_BIT_ERR_INT_EN	R/W	0x0	Command End Bit Interrupt Error Enable 0: Disabled 1: Enabled
17	CMD_CRC_ERR_INT_EN	R/W	0x0	Command CRC Error Interrupt Enable 0: Disabled 1: Enabled
16	CMD_TIMEOUT_ERR_INT_EN	R/W	0x0	Command Timeout Error Interrupt Enable 0: Disabled 1: Enabled
15	RSVD	R	0	Reserved for future use
14	CQ_SIGNAL_ENABLE	R/W	0x0	Command Queuing Signal Enable 0: Disabled 1: Enabled
13	CARD_ASYNC_INT_INT_EN	R/W	0x0	SDIO Card Async INT without AXI/SD function clock running Interrupt Enable 0: Disabled 1: Enabled

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
12	RETUNE_INT_INT_EN	R/W	0x0	Re-Tuning Interrupt Interrupt Enable 0: Disabled 1: Enabled
11	INT_C_INT_INT_EN	R/W	0x0	INT_C Interrupt Interrupt Enable 0: Disabled 1: Enabled
10	INT_B_INT_INT_EN	R/W	0x0	INT_B Interrupt Interrupt Enable 0: Disabled 1: Enabled
9	INT_A_INT_INT_EN	R/W	0x0	INT_A Interrupt Interrupt Enable 0: Disabled 1: Enabled
8	CARD_INT_INT_EN	R/W	0x0	Card Interrupt Interrupt Enable 0: Disabled 1: Enabled
7	CARD_REM_INT_EN	R/W	0x0	Card Removal Interrupt Enable 0: Disabled 1: Enabled
6	CARD_INS_INT_EN	R/W	0x0	Card Insertion Interrupt Enable 0: Disabled 1: Enabled
5	RX_RDY_INT_EN	R/W	0x0	Buffer Read Ready Interrupt Enable 0: Disabled 1: Enabled
4	TX_RDY_INT_EN	R/W	0x0	Buffer Write Ready Interrupt Enable 0: Disabled 1: Enabled
3	DMA_INT_INT_EN	R/W	0x0	DMA Interrupt Interrupt Enable 0: Disabled 1: Enabled
2	BLOCK_GAP_EVT_INT_EN	R/W	0x0	Block Gap Event Interrupt Enable 0: Disabled 1: Enabled
1	XFER_COMPLETE_INT_EN	R/W	0x0	Transfer Complete Interrupt Enable 0: Disabled 1: Enabled
0	CMD_COMPLETE_INT_EN	R/W	0x0	Command Complete Interrupt Enable 0: Disabled 1: Enabled

10.5.4.16 SD_AUTO_CMD12_ERROR_STATUS REGISTER

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PRE_VAL_EN	R/W	0x0	Preset Value Enable 0x1: Automatic selection by Preset Value are Enabled 0x0: SDCLK and Driver Strength are controlled by Host Driver
30	ASYNC_INT_EN	R/W	0x1	Asynchronous Interrupt Enable This bit can be set to 0x1 if a card supports asynchronous interrupts and <async_int_support> is set to 0x1 in the capabilities register. Asynchronous interrupt is effective when DAT[1] interrupt is used in 4-bit SD mode (and <int_pin_sel> is set to 0 in Shared Bus Control register). When this bit is set to 0x1, the host driver can stop the SDCLK during asynchronous interrupt period to save power. During this period, the host controller continues to deliver the Card Interrupt to the host when it is asserted by the card. 0x1: Enabled 0x0: Disabled
29:24	RSVD	R	0	Reserved for future use
23	SAMPLING_CLK_SEL	R/W	0x0	Sampling Clock Select Host controller uses this bit to select sampling clock to receive CMD and DAT. This bit is set by tuning procedure and valid after the completion of tuning (when <exe_tuning> is cleared). 0x1: Tuned clock is used to sample data (indicating that tuning was successful). 0x0: Fixed clock is used to sample data (indicating that tuning failed). Writing 0x1 to this bit is meaningless and ignored if tuning is not yet complete. If the bit is written as 0x0, the tuning circuit is reset. However, resetting the tuning circuit requires time to complete the tuning sequence. The host driver should keep this bit set to 0x1 to perform re-tuning quickly when necessary. Changes to this bit are not allowed while the host controller is receiving responses or a read data block.
22	EXE_TUNING	R/WAC	0x0	Execute Tuning This bit is set to 0x1 to start tuning procedure and automatically cleared when tuning procedure is completed. The result of the tuning is reflected in the <sampling_clk_sel>. The tuning procedure is aborted by writing 0x0.

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x1: Execute Tuning 0x0: Not tuned or Tuning completed
21:20	DRV_STRENGTH_SEL	R/W	0x0	Driver Strength Select. Host Controller output driver in 1.8V signaling is selected by this field. With 3.3V signaling, this field is not effective. This field can be set depending on Driver Type A, C and D support bits in the Capabilities register. This bit depends on the setting of <pre_val_en>. - If <pre_val_en> = 0x0, this field is set by the host driver. - If <pre_val_en> = 0x1, this field is automatically set by a value specified in one of the Preset Value registers. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
19	SDH_V18_EN	R/W	0x0	1.8V Signaling Enable 0x1: 1.8V Signaling enable 0x0: 3.3V Signaling enable
18:16	UHS_MODE_SEL	R/W	0x0	UHS Mode Select This field is used to select one of UHS-I modes and effective when <sdh_v18_en> = 0x1. If <pre_val_en> in the Host Control2 register is set to 0x1, host controller sets SDCLK Frequency select, Clock generator select in the clock control register, and driver strength select according to the Preset Value registers. In this case, one of the preset value registers is selected by this field. Host driver should reset <sd_clk_en> before changing this field to avoid generating clock glitch 0x0: SDR12 0x1: SDR25 0x2: SDR50 0x3: SDR104 0x4: DDR50 For MMC mode, added two backdoor defined modes: 0x5: HS200 mode 0x6: HS400 mode All other values are Reserved
15:8	RSVD	R	0	Reserved for future use
7	CMD_NOT_ISSUED	RC	0x0	Command Not Issued This is due to auto_cmd12 Error
6:5	RSVD	R	0	Reserved for future use
4	AUTO_CMD_INDE	R/W1	0x0	Auto CMD12 or Auto CMD23 Error

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
	X_ERR	C		This error occurs when the command index error occurs in response to a command 0: Disabled 1: Enabled
3	AUTO_CMD_END_BIT_ERR	R/W1C	0x0	Auto CMD12 or Auto CMD23 End Bit Error This error occurs when detecting that the end bit of command response is 0. 0: Disabled 1: Enabled
2	AUTO_CMD_CRC_ERR	R/W1C	0x0	Auto CMD12 or Auto CMD23 CRC Error This error occurs when detecting CRC error in the command response. 0: Disabled 1: Enabled
1	AUTO_CMD_TIMEOUT_ERR	R/W1C	0x0	Auto CMD12 or Auto CMD23 Timeout Error This error occurs when no response is returned within 64 SDCLK cycles from the end bit of command. 0: Disabled 1: Enabled
0	AUTO_CMD12_NOT_EXE	R/W1C	0x0	Auto CMD12 Not Executed This error occurs when host controller cannot issue Auto cmd12 to stop multiple block data transfer due to some errors. 0: Disabled 1: Enabled

10.5.4.17 SD_CAPABILITIES_1 REGISTER

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	CFG_SLOT_TYPE	R/W	0x0	Slot Type This field indicates what type of slot the host controller is connected to. 0x0: Removable card slot 0x1: Embedded slot for one device 0x2: Shared bus slot 0x3: Reserved
29	ASYNC_INT_SUPPORT	R	0x1	Asynchronous Interrupt Support. 0x1: Asynchronous Interrupt Supported 0x0: Asynchronous Interrupt not supported

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
28	SYS_BUS_64_SUPPORT	R	0x0	64-bit System Bus Support This bit indicates whether the host controller is capable of 64-bit system bus. 0x1: 64-bit system bus supported 0x0: 64-bit system bus not supported
27	RSVD	R	0	Reserved for future use
26	VLG_18_SUPPORT	R	0x1	Voltage Support 1.8V This bit indicates whether the host controller is capable of 1.8V. 0x1: 1.8V Supported 0x0: 1.8V not supported
25	VLG_30_SUPPORT	R	0x0	Voltage Support 3.0V This bit indicates whether the host controller is capable of 3.0V. 0x1: 3.0V Supported 0x0: 3.0V not supported
24	VLG_33_SUPPORT	R	0x1	Voltage Support 3.3V This bit indicates whether the host controller is capable of 3.3V. 0x1: 3.3V Supported 0x0: 3.3V not supported
23	SUS_RES_SUPPORT	R	0x1	Suspend Resume Support This bit indicates whether the host controller is capable of suspend resume commands. 0x1: Suspend/Resume Supported 0x0: Suspend/Resume not supported.
22	SDMA_SUPPORT	R	0x1	SDMA Support This bit indicates whether the host controller is capable of SDMA. 0x1: SDMA Supported 0x0: SDMA not supported.
21	HI_SPEED_SUPPORT	R	0x1	High Speed Support This bit indicates whether the host controller is capable of high speed mode (25 - 50MHz). 0x1: High speed mode Supported 0x0: High speed mode not supported.
20	ADMA1_SUPPORT	R	0x1	ADMA1 Support This bit indicates whether the host controller is capable of ADMA1. 0x1: ADMA1 Supported 0x0: ADMA1 not supported
19	ADMA2_SUPPORT	R	0x1	ADMA2 Support

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
				This bit indicates whether the host controller is capable of ADMA2. 0x1: ADMA2 Supported 0x0: ADMA2 not supported
18	EX_DATA_WIDTH_SUPPORT	R	0x1	8-bit Support This bit indicates whether the host controller is capable of 8-bit bus operation. 0x1: 8-bit Supported 0x0: 8-bit not supported
17:16	MAX_BLK_LEN	R	0x0	Maximum Block Length The maximum block length in bytes. 0x0: 512 Bytes
15:8	BASE_FREQ	R	0x00	Base Frequency The base clock frequency for SDCLK 0xC8: 200 MHz (Actual frequency: 198.24 MHz) 0x0: Get frequency information via another method
7	TIMEOUT_UNIT	R	0x1	Timeout Unit The unit of base clock used to detect timeouts. 1: MHz 0: kHz
6	RSVD	R	0	Reserved for future use
5:0	TIMEOUT_FREQ	R	0x00	Timeout Frequency This value indicates the base clock frequency used to detect timeouts. 0x32: 50 MHz (Actual frequency: 49.56 MHz). 0x0: Get frequency information via another method

10.5.4.18 SD_CAPABILITIES_3 REGISTER

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	R	0	Reserved for future use
23:16	<CLK_MULTIPLIER>	R	0x0	Clock Multiplier This field indicates clock multiplier value of programmable clock generator. 0x0: Host Controller does not support a programmable clock generator.
15:1	RETUNE_MODES	R	0x0	Re_tuning modes.

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
4				This field selects the re-tuning method and limits the maximum data length. 0x0: Mode1, Timer 0x1: Mode2, Timer and Re-Tuning Request 0x2: Mode3, Auto Re-Tuning (for transfer) Timer and Re-Tuning Request 0x3: Reserved
13	SDR50_TUNE	R	0x1	Use Tuning for SDR50 mode. 0x1: SDR50 requires tuning 0x0: SDR50 does not require tuning
12	RSVD	R	0	Reserved for future use
11:8	TMR_RETUNE	R	0xf	Timer count for Re-Tuning This field indicates an initial value of the Re-Tuning Timer for Mode 1 to 3. 0xf: Get information from other sources
7	RSVD	R	0	Reserved for future use
6	DRV_TYPE_D	R	0x1	Driver Type D Support 0x1: Driver Type D is supported 0x0: Driver Type D is not supported
5	DRV_TYPE_C	R	0x1	Driver Type C Support. 0x1: Driver Type C is supported 0x0: Driver Type C is not supported
4	DRV_TYPE_A	R	0x1	Driver Type A Support 0x1: Driver Type A is supported 0x0: Driver Type A is not supported
3	RSVD	R	0	Reserved for future use
2	DDR50_SUPPORT	R	0x1	DDR50 Support 0x1: DDR50 is supported 0x0: DDR50 is not supported
1	SDR104_SUPPORT	R	0x1	SDR104 Support 0x1: SDR104 is supported 0x0: SDR104 is not supported
0	SDR50_SUPPORT	R	0x1	SDR50 Support 0x1: SDR50 is supported 0x0: SDR50 is not supported

10.5.4.19 SD_MAX_CURRENT_1 REGISTER

Offset: 0x48				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	R	0	Reserved for future use
23:16	MAX_CUR_18	R	0x0	Maximum Current for 1.8V 0x0: Get information via another method 0x1: 4 mA 0x2: 8 mA ... 0xFF: 1020 mA
15:8	MAX_CUR_30	R	0x0	Maximum Current for 3.0V 0x0: Get information via another method 0x1: 4 mA 0x2: 8 mA ... 0xF: 1020 mA
7:0	MAX_CUR_33	R	0x0	Maximum Current for 3.3V 0x0: Get information via another method 0x1: 4 mA 0x2: 8 mA ... 0xF: 1020 mA

10.5.4.20 SD_MAX_CURRENT_3 REGISTER

Offset: 0x4C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RSVD	R	0	Reserved for future use

10.5.4.21 SD_FORCE_EVENT_AUTO_CMD12_ERROR REGISTER

Offset: 0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
31	F_CRC_STATUS_ERR	W	0x0	Force Event for CRC Status Error When 1 is written at this location, it sets the <CRC Status Error> field in the Error Interrupt Status Register.
30	F_CPL_TIMEOUT_ERR	W	0x0	Force Event for CPL Timeout Error When 1 is written at this location, it sets the <Command Completion Signal Timeout Error> field in the Error Interrupt

Offset: 0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
				Status Register.
29	F_AXI_RESP_ERR	W	0x0	Force Event for AXI Response Bit Error When 1 is written at this location, it sets the <AXI Bus Response Error> field in the Error Interrupt Status Register.
28	F_SPI_ERR	W	0x0	Force Event for SPI Error When 1 is written at this location, it sets the <SPI Mode Error> field in the Error Interrupt Status Register.
27:26	RSVD	R	0	Reserved for future use
25	F_ADMA_ERR	W	0x0	Force Event for ADMA Error When 1 is written at this location, it sets the <ADMA Error> field in the Error Interrupt Status Register.
24	F_ACMD12_ERR	W	0x0	Force Event for Auto Cmd12 Error When 1 is written at this location, it sets the <Auto CMD12 Error> field in the Error Interrupt Status Register.
23	F_CURRENT_ERR	W	0x0	Force Event for Current Limit Error When 1 is written at this location, it sets the <Current Limit Error> field in the Error Interrupt Status Register.
22	F_DAT_END_BIT_ERR	W	0x0	Force Event for Data End Bit Error When 1 is written at this location, it sets the <ReadDataEnd Bit Error> field in the Error Interrupt Status Register.
21	F_DAT_CRC_ERR	W	0x0	Force Event for Data CRC Error When 1 is written at this location, it sets the <Read Data CRC Error> field in the Error Interrupt Status Register.
20	F_DAT_TO_ERR	W	0x0	Force Event for Data Timeout Error When 1 is written at this location, it sets the <Read Data CRC Error> field in the Error Interrupt Status Register.
19	F_CMD_INDEX_ERR	W	0x0	Force Event for Command Index Error When 1 is written at this location, it sets the <Data Timeout Error> field in the Error Interrupt Status Register.
18	F_CMD_END_BIT_ERR	W	0x0	Force Event for Command End Bit Error When 1 is written at this location, it sets the <Command Index Error> field in the Error Interrupt Status Register.
17	F_CMD_CRC_ERR	W	0x0	Force Event for Command CRC Error When 1 is written at this location, it sets the <Command CRC Error> field in the Error Interrupt Status Register.
16	F_CMD_TO_ERR	W	0x0	Force Event for Command Timeout Error When 1 is written at this location, it sets the <Command Timeout Error> field in the Error Interrupt Status Register.
15:8	RSVD	R	0	Reserved for future use

Offset: 0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
7	F_ACMD12_ISSUE_ERR	W	0x0	Force Event for Command not Issued by Auto Cmd12 Error When 1 is written at this location, it sets the <Command Not Issued Due to auto_cmd12 Error> field in the Auto CMD12 Error Status Register.
6:5	RSVD	R	0	Reserved for future use
4	F_ACMD_INDEX_ERR	W	0x0	Force Event for Auto Cmd Index Error When 1 is written at this location, it sets the <Auto CMD Error> field in the Auto CMD Error Status Register.
3	F__ACMD_EBIT_ERR	W	0x0	Force Event for Auto Cmd End Bit Error When 1 is written at this location, it sets the <Auto CMD End Bit Error> field in the Auto CMD Error Status Register.
2	F_ACMD_CRC_ERR	W	0x0	Force Event for Auto Cmd CRC Error When 1 is written at this location, it sets the <Auto CMD CRC Error> field in the Auto CMD Error Status Register.
1	F_ACMD_TO_ERR	W	0x0	Force Event for Auto Cmd Timeout Error When 1 is written at this location, it sets the <Auto CMD Timeout Error> field in the Auto CMD Error Status Register.
0	F_ACMD12_NEXE_ERR	W	0x0	Force Event for Auto Cmd12 Not Executed Error When 1 is written at this location, it sets the <Auto CMD12 Not Executed> field in the Auto CMD12 Error Status Register.

10.5.4.22 SD_ADMA_ERROR_STATUS REGISTER

Offset: 0x54				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	RSVD	R	0	Reserved for future use
2	ADMA_LEN_ERR	R/W	0x0	ADMA Length Mismatch Error This error occurs in the following two cases: - While Block Count Enable (in SD_TRANSFER_MODE_CMD Bit[1]) is set, the total data length is specified by the descriptor table, which is different from that specified by the block count and block Length. - Total data length cannot be divided by block length
1:0	ADMA_STATE	R/W	0x0	ADMA Error State This field indicates the state of ADMA when error occurred during ADMA transfer. This field never indicates 0x2 because ADMA never stops in this state.

10.5.4.23 ADMA SYSTEM ADDRESS REGISTER 1 REGISTER

Offset: 0x58				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	ADMA_SYS_ADDR	R/W	0x0	ADMA System Address This register holds the byte address of the executing command in the Descriptor table. - At the start of ADMA, this register should be programmed to point to the starting address of the Descriptor table. - This register is incremented as the system fetches each descriptor line. - In case of an ADMA Error Interrupt, this register will contain a valid Descriptor address based on the ADMA state at the time.
15:0	ADMA_SYS_ADDR	R/W	0x0	ADMA System Address This register holds the byte address of the executing command in the Descriptor table. - At the start of ADMA, this register should be programmed to point to the starting address of the Descriptor table. - This register is incremented as the system fetches each descriptor line. - In case of an ADMA Error Interrupt, this register will contain a valid Descriptor address based on the ADMA state at the time.

10.5.4.24 SD_ADMA_SYS_ADDR_3 REGISTER

Offset: 0x5C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	ADMA_SYS_ADDR	R/W	0x0	ADMA System Address This register holds the byte address of the executing command in the Descriptor table. - At the start of ADMA, this register should be programmed to point to the starting address of the Descriptor table. - This register is incremented as the system fetches each descriptor line. - In case of an ADMA Error Interrupt, this register will contain a valid Descriptor address based on the ADMA state at the time.
15:0	ADMA_SYS_ADDR	R/W	0x0	ADMA System Address This register holds the byte address of the executing command in the Descriptor table. - At the start of ADMA, this register should be programmed to point to the starting address of the Descriptor table.

Offset: 0x5C				
Bits	**Field**	**Type**	**Reset**	**Description**
				- This register is incremented as the system fetches each descriptor line. - In case of an ADMA Error Interrupt, this register will contain a valid Descriptor address based on the ADMA state at the time.

10.5.4.25 PRESET_VALUE_FOR_INIT REGISTER

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
29:27	RSVD	R	0	Reserved for future use
26	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value. This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
25:16	SDCLK_FREQ_SEL_VAL	R	0x004	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.
15:14	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
13:11	RSVD	R	0	Reserved for future use
10	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value. This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
9:0	SDCLK_FR	R	0x100	SDCLK Frequency Select Value

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
	EQ_SEL_VAL			10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.

10.5.4.26 PRESET_VALUE_FOR_HS REGISTER

Offset: 0x64				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
29:27	RSVD	R	0	Reserved for future use
26	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
25:16	SDCLK_FREQ_SEL_VAL	R	0x004	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.
15:14	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
13:11	RSVD	R	0	Reserved for future use
10	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
9:0	SDCLK_FREQ_SEL_VAL	R	0x002	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the

Offset: 0x64				
Bits	**Field**	**Type**	**Reset**	**Description**
				Clock Control register.

10.5.4.27 PRESET_VALUE_FOR_SDR25 REGISTER

Offset: 0x68				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
29:27	RSVD	R	0	Reserved for future use
26	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
25:16	SDCLK_FREQ_SEL_VAL	R	0x004	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.
15:14	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
13:11	RSVD	R	0	Reserved for future use
10	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
9:0	SDCLK_FREQ_SEL_VAL	R	0x002	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.

10.5.4.28 PRESET_VALUE_FOR_SDR104 REGISTER

Offset: 0x6C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
29:27	RSVD	R	0	Reserved for future use
26	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
25:16	SDCLK_FREQ_SEL_VAL	R	0x004	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.
15:14	DRV_STRENGTH_VAL	R	0x0	Driver Strength Select Value Driver strength is supported by 1.8V signaling bus speed modes. The field is not relevant for 3.3V signaling. 0x0: Driver Type B 0x1: Driver Type A 0x2: Driver Type C 0x3: Driver Type D
13:11	RSVD	R	0	Reserved for future use
10	CLKGEN_SEL_VAL	R	0x0	Clock Generator Select Value This bit is effective when Host Controller supports programmable clock generator. 0x1: Programmable clock generator 0x0: Divided clock
9:0	SDCLK_FREQ_SEL_VAL	R	0x002	SDCLK Frequency Select Value 10-bit preset value to set <sdclk_freq_sel> in the Clock Control register.

10.5.4.29 SHARED_BUS_CTRL REGISTER

Offset: 0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RSVD	R	0	Reserved for future use
30:24	BEND_PWR_CTRL	R/W	0x0	Back-End Power Control Each bit of this field controls back-end power supply for an embedded device. The maximum number of devices that can be controlled is 7, as specified by <num_clk_pins>. Each bit is mapped to a device specific, where - bit[24] corresponds to Device 1 - bit[30] corresponds to Device 7 Note: Host interface voltage is not controlled by this field. The function of each bit is: 0x0: Back-end power is off 0x1: Back-end power is supplied
23	RSVD	R	0	Reserved for future use
22:20	INT_PIN_SEL	R/W	0x0	Interrupt Pin Select Interrupt pin inputs are enabled by this field. Enabling an unsupported interrupt pin has no effect. 0x0: Interrupt is detected by interrupt cycle.
19	RSVD	R	0	Reserved for future use
18:16	CLK_PIN_SEL	R/W	0x0	Clock Pin Select One of the clock pin outputs is selected by this field. 0x0: Clock Pins are disabled
15	RSVD	R	0	Reserved for future use
14:8	BUS_WIDTH_PRESET	R	0x0	Bus Width Preset This field defines the bus width preset for devices on a shared bus. 0x0: Bus width defined by <data_transfer_width> 0x1: 8-bit mode
7:6	RSVD	R	0	Reserved for future use
5:4	NUM_INT_PINS	R	0x0	Number of interrupt input pins This field defines the number of interrupt input pins supported on shared bus system. 0x0: Interrupt input pin is not supported
3	RSVD	R	0	Reserved for future use
2:0	NUM_CLK_PINS	R	0x0	Number of clock pins

Offset: 0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field indicates whether clock pins are supported for selecting one of the devices on a shared bus. 0x0: Shared bus is not supported

10.5.4.30 SD_SLOT_INT_STATUS REGISTER

Offset: 0xFC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	VENDOR_VER	R	0x0	Version Number
23:16	SD_VER	R	0x2	SD Host Specification Number 0x0: Supports version 1.0 0x1: Supports version 2.0 0x2: Supports version 3.0 All other values are reserved.
15:2	RSVD	R	0	Reserved for future use
1	SLOT_INT1	R	0x0	Interrupt Line for Slot 1
0	SLOT_INT0	R	0x0	Interrupt Line for Slot 0

10.5.4.31 SDHC_VID_PID REGISTER

Offset: 0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	VERSION_ID	R	0x1	0x1: The IP version
27:20	PROJECT_ID	R	0x1	0x1: The project version
19:0	VENDOR_ID	R	0xa1312	Stands for vendor name

10.5.4.32 SDHC_OP_CTRL REGISTER

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	R	0	Reserved for future use
15	WR_OSTDG	R/W	0x0	This field controls whether outstanding write

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
				requests are allowed on AXI. 0x0: Allow outstanding requests on AXI. 0x1: DO not allow outstanding requests on AXI.
14	RD_OSTDG	R/W	0x0	This field controls whether outstanding read requests are allowed on AXI. 0x0: Allow outstanding requests on AXI. 0x1: DO not allow outstanding requests on AXI.
13:8	RSVD	R	0	Reserved for future use
7	WR_ENDIAN	R/W	0x1	This field determines the endian format for the data being written to the card. 1: Little Endian 0: Big Endian
6	RD_ENDIAN	R/W	0x1	This field determines the endian format for the data being read from the card. 1: Little Endian 0: Big Endian
5	AXI_NON_POST_WR	R/W	0x0	AXI Non-post Write 1: All AXI master write requests are non-post write 0: Only the last request is issued as non-post write
4	PRIORITY	R/W	0x0	This field controls the priority level for AXI requests, acting as the most significant bit of the ID when requests are made 0x0: low priority 0x1: high priority Note: This bit should only be modified before or after the completion of a data command, not during the transaction.
3:2	DMA_SIZE	R/W	0x3	FIFO Threshold This field sets the FIFO threshold for the internal FSM to generate a DMA request to the AXI master. 0x0: 64 bytes 0x1: 128 bytes 0x2: 192 bytes 0x3: 256 bytes
1:0	BRST_SIZE	R/W	0x2	DMA Burst Size on the AXI Fabric 0x0: 32 bytes burst 0x1: 64 bytes burst

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x2: 128 bytes 0x3: 256 bytes

10.5.4.33 SDHC_OP_EXT_REG REGISTER

Offset: 0x108				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CEN_DEASSERT	R/W	0x0	Backdoor control bit to deassert the DPSRAM CENA/CENB port. Setting this bit allows DPSRAM CLKPA/CLKPB to receive additional clock cycles. 0: CEN deasserted disable 1: CEN deasserted enable
30:28	EMA	R/W	0x7	DPSRAM extra time for memory read and write 000: Fastest 111: Slowest
27:26	EMAW	R/W	0x1	DPSRAM delay for write operation by extending the internal write pulse. 2'b00: Fastest 2'b11: Slowest
25	EMAS	R/W	0x0	This field controls the extension of the DPSRAM pulse width of the sense-amplifier enable signal. The default setting is low but when driving high, the pulse is extended.
24	RET1N	R/W	0x1	Retention mode 1 enable, active low.
23:20	RSVD	R	0	Reserved for future use
19:16	PRE_GATE_CLK_CNT	R/W	0x9	This field controls the amount of clock cycles that are provided before clock gating is enabled on the sd_clk io pad.
15	AUTOCMD12_XFER_ENABLE	R/W	0x0	This bit is backdoor enable bit when software executes CMD25 with AutoCMD12.There are two conditions that can trigger a transfer interrupt: - Write operation busy bit check - Normal data transfer AUTOCMD12 finished Normally, a transfer interrupt is triggered only when the write operation busy bit check is completed. However, enabling this bit allows

Offset: 0x108				
Bits	**Field**	**Type**	**Reset**	**Description**
				an additional interrupt trigger condition when the normal data transfer of AutoCMD12 is finished. By default, software can keep this bit set to 0.
14	PDLVMC	R/W	0x0	This field controls some power down function for the internal memory.
13	PDFVSSM	R/W	0x0	This field controls some power down function for the internal memory.
12	FORCE_CLK_ON	R/W	0x0	This bit forces the SD I/O pad clock to remain on. Setting this field to 0x1 overrides the SD Clock I/O pad clock gate, ensuring the clock is continuously active. It is typically used together with the <OVRRD_CLK_OEN> field Bit[12:11] Setting: 2'bx0: setting is ignored (not used) 2'b01: Clocks are forced off 2'b11: Clocks are forced on
11	OVRRD_CLK_OEN	R/W	0x0	Override Pad Clock Output Enable Setting this field to 0x1 overrides the SD Pad clock output enable. It is typically used together with the <FORCE_CLK_ON> field.
10	CLK_OE_USE_POS	R/W	0x1	This field controls the SD/eMMC bus CLK PAD output enable signal use clock rising edge output or use clock falling edge output 0: Use internal clock falling edge output (to eliminate clock signal Tri-state issue) 1: Use internal clock rising edge output
9	CLK_GATE_ON	R/W	0x0	Clock Gate On 0: Enable dynmical clock gate 1: Enable clock free running This field affects all clock gates in the SD design if the <Clock Gate Ctl> field is set to 1.
8	CLK_GATE_CTL	R/W	0x0	Clock Gate Control 0: Disable software clock gating override 1: Enable software clock gating override
7	USE_DAT3	R/W	0x0	This field allows the card detect functionality to be detected using the DAT[3] pin. This field controls whether the DAT[3] pin is used for card detection. 0x0: Use dedicated pin for card detection

Offset: 0x108				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x1: Use DAT[3] for card detection
6	PDWN	R/W	0x0	Power Down This bit controls the Power Down port on the internal DPFIFO.
5	FIFO_CS	R/W	0x0	FIFO CS This field should be written to 0x1 before any toggling of the PDWN bit.
4	FIFO_CLK	R/W	0x0	FIFO Clock This field should be set to 0x1 before any toggling of the PDWN bit.
3:2	WTC	R/W	0x0	WTC This field is used for FIFO speed setting.
1:0	RTC	R/W	0x0	RTC This field is used for FIFO speed setting.

10.5.4.34 SDHC_LEGACY_CTRL_REG REGISTER

Offset: 0x10C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	GEN_PAD_CLK_CNT	R/W	0x4a	Pad Clock Count This field should be used with <gen_pad_clk_on> This field configures the number of clock cycles generated on the IO pad. The default value of 0x4A generates 75 clock cycles.
23:14	RSVD	R	0	Reserved for future use
13:9	SPI_ERR_TOKEN	R/W	0x0	SPI Error Token This is the SPI Error token received in command response when SPI mode is enabled.
8	SPI_EN	R/W	0x0	Enable SPI Mode This field indicates whether SPI mode is enabled. When enabled, the host controller drives the signals according to the SPI protocol. 1 = SPI mode enabled 0 = SPI mode disabled
7	RSVD	R	0	Reserved for future use

Offset: 0x10C				
Bits	**Field**	**Type**	**Reset**	**Description**
6	GEN_PAD_CLK_ON	R/WAC	0x0	Generate Pad Clock This bit works together with the <gen_pad_clk_cnt> field. Setting this bit to 0x1 will generate the programmed number of clock cycles on the IO pad.
5	SQU_FULL_CHK	R/W	0x0	SQU Full Check This bit should be set to 0x1 only when using a specific piece of memory in the SQU in FIFO mode and performing a "read" transaction on the SD device.
4	SQU_EMPTY_CHK	R/W	0x0	SQU Empty Check This bit should be set to 0x1 only when using a specific piece of memory in the SQU in FIFO mode and performing a "write" transaction on the SD device.
3	BOOT_ACK	R/W	0x0	Boot Acknowledgment If the boot nowledgment mode is enabled in the MMC device, then this field should be written to 0x1 before issuing the alternate boot CMD0.
2	INAND_SEL	R/W	0x1	When the driver programs the highest byte of Rx0c, the host DAT/CMD line related registers will be reset. 0x1: Enable soft reset when trigger command. 0x0: Disable soft reset when trigger command.
1	ASYNC_IO_EN	R/W	0x0	Asynchronous Read Interface Enable This bit enables the asynchronous latching of input data. 0x1: Async interface is enabled. The clock used to latch the input data is asynchronous with the internal logic clock. 0x0: Async interface is disabled.
0	PIO_RDFC	R/W	0x1	PIO mode read operation FIFO check. 0x0: The system checks if all FIFO data has been read by the CPU. If not, the bus clock will stop, and the state will wait until the last block of FIFO data has been read by the CPU. Once the data is done being read, the next block will begin transferring. 0x1 = The system does not check if the FIFO

Offset: 0x10C				
Bits	**Field**	**Type**	**Reset**	**Description**
				data has been read by the CPU. It is recommended setting this bit to 1 before use PIO mode. This is especially important in high-speed modes like HS200/SDR104, where the software must set this bit to 1 to ensure proper operation.

10.5.4.35 SDHC_LEGACY_CEATA_REG REGISTER

Offset: 0x110				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	R	0	Reserved for future use
29:16	CPL_TIMEOUT	R/W	0x3FFF	Command Completion Signal Timeout Value
15:3	RSVD	R	0	Reserved for future use
2	CHK_CPL	R/W	0x0	Check Command Completion Signal When this field is set to 0x1 and the <CE-ATA Card> field is set to 1, indication is sent to the host controller to check for command completion signal from the CE-ATA card.
1	SND_CPL	R/W	0x0	Send Command Completion Disable Signal When this field is set to 1 and the <CE-ATA Card> field is set to 1, indication is sent to the host controller to send the command completion disable signal to the CE-ATA card.
0	CEATA_CARD	R/W	0x0	CE-ATA Card 1: CE-ATA Card mode 0: Non CE-ATA card mode

10.5.4.36 SDHC_MMC_CTRL_REG REGISTER

Offset: 0x114				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	DAT_LEVEL	R	0xFF	MMC1_DAT[7:0] Line Signal Level This status is used to check the MMC_DAT[7:0] line level. It is useful for error recovery and debugging. This helps in detecting issues, especially the busy

Offset: 0x114				
Bits	**Field**	**Type**	**Reset**	**Description**
				signal level from MMC_DAT[0].
23:13	RSVD	R	0	Reserved for future use
12	MMC_CARD	R/W	0x0	MMC Card 1: MMC Card mode 0: SD Card mode
11	MMC_RESETN	R/W	0x1	MMC Resetn This bit controls the value of the pin MMC_RESETN going to the eMMC device.
10	MMC_HS200	R/W	0x0	This bit is set when host read DEVICE_TYPE[196] field of Extended CSD register in MMC card, and the card support HS200 mode. It should be set before the host clock changes to 200Mhz. 1: MMC HS200 mode enable. 0: MMC HS200 mode disable
9	MMC_HS400	R/W	0x0	This bit is set when the host reads the DEVICE_TYPE[196] field of the Extended CSD register in the MMC card, and the card supports HS400 mode. It should be set before the host clock changes to 200 MHz, following the eMMC 5.0 specification. 1: MMC HS400 mode enable. 0: MMC HS400 mode disable
8	ENHANCE_STROBE_EN	R/W	0x0	Enhanced Strobe Enable This bit controls whether the host/PHY will use the delayed strobe signal to sample the CMD response. This feature is part of the eMMC 5.1 specification to enhance HS400 mode. Software (SW) should check the device-related EXT to decide whether the host/PHY should support this feature.
7	RSVD	R	0	Reserved for future use
6	CPL_COMPLETE	R/W1C	0x0	This bit is set to 1 when a command completion signal is detected and the <cpl_complete Enable> field has been set to 1. The field is cleared by writing 0x1. A write of 0x0 has no effect.
5	CPL_COMPLETE_EN	R/W	0x0	This bit controls whether the CPL_COMPLETE bit gets set when a command completion signal is detected. When set to 1, the CPL_COMPLETE bit will be set to 1 when a command completion signal is detected.
4	CPL_COMPLETE_INT	R/W	0x0	cpl_complete Interrupt Enable 1: An interrupt will be generated when the

Offset: 0x114				
Bits	**Field**	**Type**	**Reset**	**Description**
	_EN			<cpl_complete> field is set
3	RSVD	R	0	Reserved for future use
2	MISC_INT	R/W1C	0x0	Miscellaneous Interrupt This status bit is set to 1 when the programmed number of clocks in the <gen_pad_clk_cnt> field is completed and the <misc_int_en> field has been set to 0x1. This field is cleared by writing 0x1. A write of 0x0 has no effect.
1	MISC_INT_EN	R/W	0x0	Miscellaneous Interrupt Status Enable When this bit is set to 0x1 will enable the misc_int bit to be set to 0x1 when the programmed number of clocks have been generated on the Pad.
0	MISC_INT_INT_EN	R/W	0x0	misc_int Interrupt Enable 0x1: An interrupt will be generated when the <misc_int> field is set to 0x1.

10.5.4.37 SDHC_RX_CFG_REG REGISTER

Offset: 0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	RSVD	R	0	Reserved for future use
27:18	TUNING_DLY_INC	R/W	0x0	Tuning Delay Increment This field defines the increment value used for each step when the hardware performs auto-tuning. When the host controller takes control of the delay value during auto-tuning, this field specifies by how much the delay value will be incremented at each step.
17:8	SDCLK_DELAY	R/W	0x0	This field controls the delay value to the delay element.
7:4	RSVD	R	0	Reserved for future use
3:2	SDCLK_SEL1	R/W	0x0	This field is used for Rx data/CMD sample clock selections. This field controls the second mux selects. 0x0: Select the clock from GPIO pad feedback. 0x1: Select the clock output from DDL. If for software tuning, set this value to 0x1 0x2: Select internal clock 0x3: Select internal clock

Offset: 0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
1:0	SDCLK_SEL0	R/W	0x0	This field is used for the software tuning process. This field controls the first mux selects. 0x0: Select clock from pad 0x1: Select inverted clock from pad 0x2: Select internal clock 0x3: Select inverted internal clock.

10.5.4.38 SDHC_TX_CFG_REG REGISTER

Offset: 0x11C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	TX_MUX_SEL	RW	0x0	TX output clock selection 0x0: Select clock from inverter of base clock input 0x1: Select clock from DDLLoutput clock
30	TX_INT_CLK_SEL	RW	0x0	TX output clock selection. 0x0: Select the clock from the original inverter of base clock or DDLL output clock 0x1: Select the clock from the inverter of the internal work clock, ensuring the hold time in default speed mode or high-speed mode.
29	TX_DLINE_SRC_SEL	RW	0x0	TX delayline clock source selection. 0x0: Select the base clock as TX delayline input source clock 0x1: Select the internal work clock as the TX delayline input source clock. Normally, this bit only worked at HS200 mode, but in DDR mode, if TX tuning is required, this TX delayline input clock source bit should be force to 0
28:26	RSVD	R	0	Reserved for future use
25:16	TX_HOLD_DELAY1	RW	0x37	This field controls the delay value of the TX delay element for SDR104 mode.
15:10	RSVD	R	0	Reserved for future use
9:0	TX_HOLD_DELAY0	RW	0xc5	This field controls the delay value of the TX delay element for all modes other than the SDR104 mode.

10.5.4.39 SDHC_HWTUNE_CFG_REG REGISTER

Offset: 0x120				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	R	0	Reserved for future use
29:20	TUNING_CLK_DLY	R	0x0	This read-only field indicates the final DDLL delay counter value after the tuning process is completed. During the tuning process, this field increments step by step. Therefore, it is typically not meaningful to read this value during hardware tuning.
19:10	TUNING_WD_CNT	R/W	0x0a	This field controls the tuning success window width. If tuning success times ≥ TUNING_WD_CNT, the tuning is considered successful. Then, hardware will select the middle of the window as the final tuning DDLL delay counter value. The default value is 10, which aligns with the specifications, but the driver can adjust this count based on real-world conditions.
9:0	TUNING_TT_CNT	R/W	0x27	This field controls the total tuning times. The default value is 40, as specified in the specifications. The driver can adjust this count based on real-world conditions. The total number of tuning attempts is calculated as TUNING_TT_CNT + 1

10.5.4.40 SDHC_HWTUNE_CFG2_REG REGISTER

Offset: 0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
31:26	RSVD	R	0	Reserved for future use
25:16	TUNING_HW_START_CNT	R/W	0x0	This field allows software to configure the starting DLL counter value for hardware auto-tuning. The value set here determines where the hardware auto-tuning process will begin.
15:6	_TUNING_SUCCESS_CNT	R	0x0	This field provides software with a reference to determine if the tuning step (TUNING_DLY_INC) or the tuning window count (TUNING_WD_CNT) should be adjusted. It is only valid after the tuning process has completed and failed. Reading this field during the tuning process is meaningless.
5:4	RSVD	R	0	Reserved for future use
3:2	TUNING_HW	R	0x0	This field is used for hardware auto-tuning, and its

Offset: 0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
	_SDCLK_SEL1			function is similar to the SW tuning definition at 0x118<3:2>.
1:0	TUNING_HW_SDCLK_SEL0	R/W	0x0	This field allows software to configure the hardware auto-tuning DLL source clock selection. It provides flexibility in choosing the clock source for the tuning process.

10.5.4.41 SDHC_ROUNDTRIP_TIMING_REG REGISTER

Offset: 0x128				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	RSVD	R	0	Reserved for future use
27:24	DATA0BUSY_WAIT_CYCLES	R/W	0x2	This field controls how many clock cycles the host should wait before checking the DATA0 busy signal after detecting the end bit of the CRC status in a write operation. This setting is mainly applicable to high-speed modes such as HS200, SDR104, and HS400.
23:20	RSVD	R	0	Reserved for future use
19:16	WRDATA0_WAIT_CYCLES	R/W	0x5	This field is valid only when bit [2], bit [1], or bit [0] of this register is set in the corresponding speed mode. - If bit [1] or bit [0] is set, this field specifies how many cycles the host controller's internal logic should wait before changing the bus driving direction—from the end bit of DATA0 to the start bit of the CRC status—during a write operation.
15:12	RSVD	R	0	Reserved for future use
11:8	CMD2RESP_WAIT_CYCLES	R/W	0x5	This field only valid when this register bit[2] or bit[1] or bit[0] be set at corresponding speed mode, if bit[1] or bit[0] be set, this field indicate the host controller internal logic CMD FSM should wait how many clock cycles between the bus driving direction turn around from CMD end bit to response start bit. This field is valid only when bit [2], bit [1], or bit [0] of this register is set for the corresponding speed mode. - If bit [1] or bit [0] is set, this field specifies how many clock cycles the host controller's internal logic should wait before changing the bus driving direction—from the end bit of a CMD to the start bit of the response.
7:3	RSVD	R	0	Reserved for future use
2	TRS2RCV_P	R/W	0x0	This field controls whether, in HS400 mode, software can

Offset: 0x128				
Bits	**Field**	**Type**	**Reset**	**Description**
	ARAM_EN2			manually configure the number of wait cycles that should be inserted between CMD/DATA transmission and reception during direction turnaround. In HS400 mode, PHY output and input DLL latency should be enabled when setting this field.
1	TRS2RCV_PARAM_EN1	R/W	0x0	This field controls whether, in DDR50/SDR50 mode, software can manually configure the number of wait cycles that should be inserted between CMD/DATA transmission and reception during direction turnaround.
0	TRS2RCV_PARAM_EN0	R/W	0x0	This field controls whether, in HS200/SDR104 mode, software can manually configure the number of wait cycles that should be inserted between CMD/DATA transmission and reception during direction turnaround.

10.5.4.42 SDHC_GPIO_CFG_REG REGISTER

Offset: 0x12C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	SDHC_GPO	R	0x0	Value to be driven to GPO pins. Software programs these 16-bit fields, and the values will appear on the GPO output ports of the SDHC top module.
15:0	SDHC_GPI	R	0x0	Value on GPI ports. These 16-bit fields are read-only and reflect the current state of the GPI input ports.

10.5.4.43 SDHC_DLINE_CTRL_REG REGISTER

Offset: 0x130				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	TX_DLINE_CODE	R/W	0x0	Delayline DTC delay control signals for transmission. Software programs this field during the tuning process.
23:16	RX_DLINE_CODE	R/W	0x0	Delayline DTC delay control signals for reception. Software programs this field during the tuning process.
15:1	RSVD	R	0	Reserved for future use
0	DLINE_PU	R/W	0x0	Power-up signal for delayline: 0: Power down 1: Power up The time period from setting the power-up signal to the

Offset: 0x130				
Bits	**Field**	**Type**	**Reset**	**Description**
				internal regulator voltage stabilizing is approximately 100ns.

10.5.4.44 SDHC_DLINE_CFG_REG REGISTER

Offset: 0x134				
Bits	**Field**	**Type**	**Reset**	**Description**
31:25	RSVD	R	0	Reserved for future use
24	TX_DLINE_GAIN	R/W	0x0	This field is only effective when TX_DLINE_REG[6] is set. 1: Extended delayline delay range with decreased resolution. Typically used for a frequency of 100 MHz, where the DLL delayline is still required. 0: Default setting (for 200 MHz)
23:16	TX_DLINE_REG	R/W	0x0	Set the delay line parameters for the tuned clock in HS200 and DDR52 modes. - Bits 7:6: Reserved - Bits 5:4: Delay line tuning range (Default: 2'b01). - Bits 3:2: Delay line LDO output voltage (Default: 2'b11): 1. 2'b00: 0.7V 2. 2'b01: 0.8V 3. 2'b10: 0.85V 4. 2'b11: 0.9V - Bit 1: Enable delay line LDO resistor load (Default: 1'b1). - Bit 0: Enable delay line LDO soft start (Default: 1'b1).
15:9	RSVD	R	0	Reserved for future use
8	RX_DLINE_GAIN	R/W	0x0	This field is only effective when RX_DLINE_REG[6] is set. 1 = Extended delayline delay range with decreased resolution. Typically used for a frequency of 100 MHz, where the DLL delayline is still required. 0 = Default (for 200Mhz case)
7:0	RX_DLINE_REG	R/W	0x1F	set delay line parameter for tuned clock in HS200 and DDR52 mode. - Bits 7:6: Reserved - Bits 5:4: Delay line tuning range (Default: 2'b01). - Bits 3:2: Delay line LDO output voltage (Default: 2'b11): 1. 2'b00: 0.7V 2. 2'b01: 0.8V 3. 2'b10: 0.85V 4. 2'b11: 0.9V - Bit 1: Enable delay line LDO resistor load (Default: 1'b1). - Bit 0: Enable delay line LDO soft start (Default: 1'b1).

10.5.4.45 SDHC_PHY_CTRL_REG REGISTER

Offset: 0x160

Bits	Field	Type	Reset	Description
31	HOST_LEGACY_MODE	R/W	0x0	This field is backdoor register for SW to use old legacy topology host mode. By default, due to EMMC5 PHY/host topology changes, clock generation logic will moved into PHY inside, PHY will output working clock to host. if SW set this bit, below factors will be true: This field is a backdoor register for software to enable the old legacy topology host mode. By default, the EMMC5 PHY/host topology has changed so that the clock generation logic is moved into the PHY, which outputs the working clock to the host. When this bit is set, the following happens: - The host will use the internal clock divider to generate the clock. - SW/HW RX tuning will be affected. The host will use the 0x118/0x130 register settings for DLL slave delayline control, instead of the new 0x168 setting for DLL slave delayline control. - The host will treat the PHY as just GPIO and use the PHY test mode interface signals (TDI/TDO/TDOE) to output/input data, CMD, and CLK. - If the host legacy mode is set, it's recommended not to support HS400 mode. The base clock frequency should be ≤ 200MHz, with a maximum of HS200 mode. - TBD: Whether to add a backdoor delayline in the RX/TX tuning path for clock tuning (without using the PHY internal delayline). 0x0: Disable Host legacy mode (default, using the new external EMMC5.0/SD PHY topology). 0x1: Enable Host legacy mode.
30:24	RSVD	R	0	Reserved for future use
23:16	PHY_DCHNL_STATUS	R	0x01	Indicates the current status of the PHY 8 data channel enable settings. After a reset, the hardware logic defaults to enabling the 1-bit mode data channel.
15:8	PHY_DCHNL_SW	R/W	0x0	Software-programmed 8 data channel enable signals. Each bit represents one data channel: 0x1: This data channel is enabled. 0x0: This data channel is disabled. **Note.** The CMD channel is always enabled inside the PHY.
7:3	RSVD	R	0	Reserved for future use
2	PHY_DCHNL_SEL	R/W	0x0	Controls the selection method for enabling PHY 8 data channels: 0x1: PHY data channel function is controlled by software using bits [15:8]. If this bit is set, software must configure [15:8] before using PHY for data transfer. 0x0: PHY data channel function is automatically managed by

Offset: 0x160				
Bits	**Field**	**Type**	**Reset**	**Description**
				host hardware. - In this mode, software can read [23:16] for the current hardware data channel status. - Normally, for eMMC, the host hardware enables all 8 data channels during initialization. - For SD, the host hardware enables 4 data channels during initialization.
1	PHY_PLL_LOCK	R/W	0x0	When following the PHY programming sequence, after enabling the PHY input source clock from APMU offset 0xE0, software should set this bit to 1 to notify the PHY that the internal 400MHz clock source is stable. 0x1: PHY 400MHz clock source is stable. 0x0: PHY 400MHz clock source is unstable.
0	PHY_FUNC_EN	R/W	0x0	This bit controls the PHY function enable signal, which is used for internal circuit reset. If the host core logic enters a lower power mode with power gating (UDR latch), phy_en should remain 0. For normal operation, software should set this bit before using the PHY and configure other PHY-related settings beforehand. - 0x1: PHY function enabled / PHY powered up. - 0x0: PHY function disabled / PHY enters lower power mode.

10.5.4.46 SDHC_PHY_FUNC_REG REGISTER

Offset: 0x164				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	R	0	Reserved for future use
18	RX_USE_STROBE	R	0x0	Indicates whether the host is in HS400 DDR mode. The value is determined by the following condition: rx_use_strobe = (uhs_mode[2:0] == 110b) | hs400_mode | enhance_strobe_en; This means **RX strobe is used** when: - **UHS mode** is set to 110b - **HS400 mode** is enabled - **Enhanced strobe** is enabled
17	RX_USE_DLYLINE	R	0x0	Indicates that the host is in HS200 or SDR104 mode. The host also has a backdoor register (bit3) to set this bit in SDR50/DDR50(52) modes.
16	TX_USE_INVERT	R	0x0	This bit is set under either of the following conditions: - 0x3E<2:0> is set to all SDR modes. - 0x118<30> = 0 is set. However, when 0x118<30> = 1, it only

Offset: 0x164				
Bits	**Field**	**Type**	**Reset**	**Description**
				affects PHY test mode output paths (TDO/TDOE signals), while all functional output data, CMD, and output enable signals are still controlled by PHY logic.
15	HS200_USE_RFIFO	R/W	0x1	This bit control whether software enables CMD line RFIFO in HS200/HS400 mode: 0x0: Disables CMD line RFIFO in HS200/HS400 mode (default for HS200 mode). In this case, the data is sampled using the Delayline output clock and directly sampled again by the host internal core clock. 0x1: Enables CMD line RFIFO in HS200/HS400 mode. When RFIFO is enabled, data sampled from the Delayline output clock is sent to the Async FIFO, then popped from the FIFO and sent to the host.
14	RX_DIS_CK STOP	R/W	0x0	This bit is a backdoor register added by the host to prevent the clock from being stopped in the middle of a data block during FIFO overflow conditions. The clock can only be stopped during a block gap. This field is actually a backdoor register for only SDR25/HS50/DDR50 or lower mode, since for HS200/HS400 higher mode SPEC claimed could not stop the bus clock in middle of the data block.
13	PHY_TDI_S EL	R/W	0x1	This field control PHY test mode input signal TDI source selection: 0x0: TDI is sampled by PHY internal clock signal, which is the same as the host working clock (cclk_in) from PHY. 0x1: TDI is directly from PHY internal IO pad DI port.
12	TX_CKOUT _REVERSE	R/W	0x0	This field is backdoor register to control whether host controller reverse the output data phase: 0x0: Keep the original clock phase design. The odd phase is always 0, and the even phase is controlled by card_clk_en. 0x1: Reverse the odd phase data to the even output port, and reverse the even phase data to the odd output port.
11	TX_DDR_R EVERSE	R/W	0x0	This field is backdoor register to control whether host controller reverse the output data phase: 0x0: Keep the original design data for odd/even output phase 0x1: Reverse the odd phase data to the even output port and the even phase data to the odd output port.
10	RX_DDR_B KEN	R/W	0x0	This backdoor register controls the DDR mode RX direction in DDR50 mode. It forces PHY to use the delayline output clock to sample bus data on both edges, then outputs dq_in_o[7:0] and dq_in_e[7:0] to the host. The delayline's input clock is cki. 0x1: Backdoor mode is enabled. 0x0: Backdoor mode is disabled.

Offset: 0x164				
Bits	**Field**	**Type**	**Reset**	**Description**
9	RFIFO_BYPASS	R/W	0x0	When set, the host bypasses the PHY interface Read FIFO in HS400 mode. If not set, the host uses ck_rx_cmd to directly sample the CMD in certain modes. Normally, RFIFO is used in HS400 mode because the internal clock is asynchronous with the PHY output data/CMD Rx clock. **Note.** This bit only works in HS400 mode. In tuning mode with a free-running clock, the DATA/CMD async read path can choose to use the data async FIFO in the host RX interface (via async_io_en, 0x10C<1>).
8	CMD_USE_EVEN	R/W	0x1	This bit indicates whether the CMD uses a 3/4T DS sampled signal as the controller input in enhanced HS400 mode. By default, it uses a 1/4T DS sampled signal.
7	PHY_TEST_EN	R/W	0x0	This bit enables PHY test mode and controls the source of pad output and output enable signals. - **0x1**: Enables test mode (PHY bypass mode), where the pad output is controlled by TDO (output data) and TDOE (output enable) from the host. In this mode, the host treats PHY as GPIO and uses the test mode interface signals TDI/TDO/TDOE for data, CMD, and CLK input/output. - **0x0**: Uses the normal EMMC function for data, CMD, and CLK paths. This field works differently from **0x160<31> (host_legacy_mode)**. While **host_legacy_mode** has a broader effect, this field only impacts the data, CMD, and CLK input/output paths. If this bit is set, only **item 3 of 0x160<31>** will take effect, meaning: - **0x1**: The host treats PHY as GPIO and uses the test mode interface signals (TDI/TDO/TDOE) for I/O. - **0x0**: The host uses the normal EMMC data/CMD/CLK paths. **Note**: It is recommended to set this bit in the Bootrom for safety. The **APMU** also has a backdoor register to configure PHY bypass mode, but setting the APMU register will let TDO/TDOE come from other MFPI functions, not the EMMC controller. If this mode is used, it assumes the PHY's internal clock generation logic is functioning, and this bit is not suitable for DDR mode backup. For full backup mode, use **0x160<31>** to rely solely on PHY's IO pad.
6:4	PHY_MODE_STATUS	R	0x0	These 3 bits reflect current PHY working mode selection in host design, - If PHY_MODE_SWEN = 1, then the value equals PHY_MODE_SW[2:0]. - If PHY_MODE_SWEN = 0, then the value reflects the hardware-controlled signal PHY_MODE_HW[2:0].

Offset: 0x164				
Bits	**Field**	**Type**	**Reset**	**Description**
3:1	PHY_MODE_SW	R/W	0x0	These 3 bits are valid only when PHY_MODE_SWEN = 1. When enabled, the host controller uses this field instead of the internally generated PHY_MODE_HW signal. 0x000: MMC Default Speed Mode (≤26 MHz), SD DS/SDR12/SDR25 (≤50 MHz) 0x001: MMC HS Mode (≤50 MHz, SDR protocol), SD SDR50 (≤100 MHz) 0x010: DDR50 (or DDR52) 0x011: HS200 (or SDR104) 0x100: HS400 0x101: HS400 CMD Enhanced Mode Others: Reserved
0	PHY_MODE_SWEN	R/W	0x0	This bit allows software to control the PHY working mode selection. If set, software should ensure proper synchronization when setting this bit.

10.5.4.47 SDHC_PHY_DLLCFG_REG REGISTER

Offset: 0x168				
Bits	**Field**	**Type**	**Reset**	**Description**
31	DLL_ENABLE	R/W	0x0	Enable DLL. 0: DLL function disable (power down) 1: DLL function enable (power up)
30	DLL_DELAY_SRC	R/W	0x0	PHY internal dedicated delayline input clock source selection. 0: Uses PCLK (bus clock feedback) as the default source. 1: Switches to PHY internal cki (cclk_in), typically used for manual tuning in HS200 mode.
29	DLL_REFRESH_SW	R/W	0x0	Controls manual updates to the DLL_REFRESH signal: 1: Triggers a manual update. 0: Keeps the DLL_REFRESH signal invalid.
28	DLL_REFRESH_SWEN	R/W	0x0	1: Software manually controls DLL_REFRESH updates. 0: Host Hardware manually controls DLL_REFRESH updates.
27	DLL_REFRESH_ENABLE	R/W	0x1	1: Host controller generates DLL_REFRESH signal for manually adjusting the delay of strobe signal; 0: DLL_RERRESH remains inactive
26:16	RSVD	R	0	Reserved for future use
15:8	DLL_DELAY_CTRL	R/W	0x0	Controls the delay value for the PHY DLL slave delayline tuning in HS400 mode, used when the PHY does not use the internal DLL master counter to update the DLL slave. This field only takes effect when dll_reg1<1> (0x168<1>) = 1.

Offset: 0x168				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. In the new EMMC5.x topology, the DLL slave delay line is integrated into the PHY. For older topologies (such as those in host_legacy_mode with 0x160<31>=1) or for HS200/DDR50 tuning, the 0x114/0x130/0x134 registers will be used to control the tuning process.
7:6	DLL_VREG_CTRL	R/W	0x1	DLL regulator output voltage control
5:4	DLL_FULLDLY_RANGE	R/W	0x1	DLL delayline full delay range
3:2	DLL_PREDLY_NUM	R/W	0x1	DLL delayline Pre-delay numbers
1	DLL_BYPASS_ENABLE	R/W	0x0	DLL Master Bypass Enable for HS400 1: DLL master is bypassed, and DLL_DELAY_CTRL directly controls the slave DLL. 0: DLL is enabled. Use delay value is from the master DLL.
0	DLL_REFRESH_METHOD	R/W	0x0	DLL master code refresh method: After locked: refreshed by host generated dll_refresh rising edge. 1: Always refreshed by ck_refresh, synchronized to the filter clock. 0: Before lock: refreshed by ck_refresh. After lock: refreshed by the host-generated dll_refresh rising edge.

10.5.4.48 SDHC_PHY_DLLCFG1_REG REGISTER

Offset: 0x16C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	DLL_REG4_CTRL	R/W	0x00	Detailed bits refer to analog PHY pin list 'dll_reg4' descriptions
23:16	DLL_REG3_CTRL	R/W	0x0e	Detailed bits refer to analog PHY pin list 'dll_reg3' descriptions
15:8	DLL_REG2_CTRL	R/W	0x4a	Detailed bits refer to analog PHY pin list 'dll_reg2' descriptions
7:0	DLL_REG1_CTRL	R/W	0x00	Detailed bits refer to analog PHY pinlist 'dll_reg1' descriptions

10.5.4.49 SDHC_PHY_DLLSTS_REG REGISTER

Offset: 0x170				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	R	0	Reserved for future use
15:8	PHY_WORK_MODE	R	0x0	This field indicates the current PHY working mode flag, from PHY output signal rdo_reg2[7:0]: - [7]: SDR26 - [6]: SDR52 - [5]: DDR52 - [4]: HS200 - [3]: HS400 - [2]: cmd__extd_mode - [1]: start_dll - [0]: dll_error
7:2	RSVD	R	0	Reserved for future use
1	DLL_REFRESH_STATE	R	0x0	This bit indicates DLL manully refresh mode refresh signal state, this bit only valid when DLL_REFRESH_EN is set. 0: No refresh 1: Manully refresh state, is from DLL_REFRESH_HW (host HW control logic)
0	DLL_LOCK_STATE	R	0x0	This bit indicates whether Master DLL is at LOCK State 0: UNLOCK State 1: LOCK State. The RX path only works in **HS400** and **HS400_extend** mode when **dll_lk** is set to 1'b1

10.5.4.50 SDHC_PHY_DLLSTS1_REG REGISTER

Offset: 0x174				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	R	0	Reserved for future use
23:16	DLL_MASTER_DELAY	R	0x0	The newest delay value for the delay line in master DLL. Note: This field always reflects the most recent value regardless of whether the DLL is locked or not.
15:8	RSVD	R	0	Reserved for future use
7:0	DLL_SLAVE_DELAY	R	0x0	This field reflects the delay value currently used for Strobe Signal or the final delay value from SW/HW tuning process.

10.5.4.51 SDHC_PHY_PADCFG_REG REGISTER

Offset: 0x178

Bits	Field	Type	Reset	Description
31:22	RSVD	R	0	Reserved for future use
21:20	CLK_PU	R/W	0x0	CLK Pullup 2 bits resistor selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
19:18	DS_PU	R/W	0x0	DS Pullup 2 bits resistor selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
17:16	CMD_PU	R/W	0x2	CMD Pullup 2 bits resistor selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
15:14	RSVD	R	0	Reserved for future use
13:12	CLK_PD	R/W	0x2	CLK Pulldown 2 bits resistor selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
11:10	DS_PD	R/W	0x2	DS Pulldown 2 bits resistor selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
9:8	CMD_PD	R/W	0x0	CMD Pulldown 2 bits resistor selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
7	RSVD	R	0	Reserved for future use
6	IO_DRV_HZ	R/W	0x0	This backdoor bit enables the bus IO high-Z state after the Analog EMMC PHY fix for the AutoCMD23 timeout issue (when IO becomes floating or uncontrollable) during PHY_EN switching. - 1'b1: Enable High-Z state - 1'b0: Disable High-Z state

Offset: 0x178				
Bits	**Field**	**Type**	**Reset**	**Description**
5	RX_BIAS	R/W	0x1	- 1'b0: High current mode - 1'b1: Low current mode
4:3	SLEW_RATE	R/W	0x0	PAD slew rate control: - 2'b00: Low - 2'b01: Medium - 2'b10: High - 2'b11: Very high
2:0	DRIVE_SEL	R/W	0x4	For Drive Nominal Impedance selection: - 3'b000: high-Z - 3'b001: 200 - 3'b010: 100 - 3'b011: 66 - 3'b100: 50 - 3'b101: 40 - 3'b110: 33 - 3'b111: 33

10.5.4.52 SDHC_PHY_PADCFG1_REG REGISTER

Offset: 0x17C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	DQX_PU	R/W	0xaaaa	DQ Pullup Resistor value selection, every two bits stands for one IO selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm
15:0	DQX_PD	R/W	0x0	DQ Pulldown Resistor value selection, every two bits stands for one IO selection: - 2'b00: high-Z - 2'b01: 50K Ohm - 2'b10: 40K Ohm - 2'b11: 33K Ohm

10.5.4.53 SDHC_PHY_LBCTRL_REG REGISTER

Offset: 0x180				
Bits	**Field**	**Type**	**Reset**	**Description**

Offset: 0x180				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CLEAR_LB_ERR_STATUS	W	0x0	Write 1 to clear SDHC_PHY_LBSTS_REG register
30:3	RSVD	R	0	Reserved for future use
2	START_STUCK0_DET_CLK	R/WAC	0x0	After operation is finished, check EMMC_LB_Err_Status Register.
1	START_STUCK1_DET_CLK	R/WAC	0x0	After operation is finished, check EMMC_LB_Err_Status Register.
0	LB_TEST_TRIGGER	R/WAC	0x0	Triggers Loopback Testing immediately. Before setting this bit, the host driver should configure the SDHC_PHY_LBCNT_REG register (offset: 0x188). - The test will run until SDHC_PHY_LBCNT_REG reaches 0, at which point the test stops and this bit clears automatically. - This bit is only effective when Loopback Mode Enable is set.

10.5.4.54 SDHC_PHY_LBFUNC_REG REGISTER

Offset: 0x184				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	RSVD	R	0	Reserved for future use
27:24	LB_DS_CNT	R/W	0x0	Represents the total latency from TX to RX through the PHY loopback path. - Software should program this field to match PHY characteristics. - Recommended values: 1. HS400 or HS200 with RFIFO mode: Set to 5 2. Other modes: Set to 4 - Must be greater than 1. Note: this field should be > 1.
23:20	LB_FILTER_CNT	R/W	0x0	The first LB_FILTER_CNT bits and the last LB_FILTER_CNT bits are ignored in the comparison process.
19:17	RSVD	R	0	Reserved for future use
16	LB_CM	R/W	0x0	If set, the corresponding path will not perform lookback test.

Offset: 0x184				
Bits	**Field**	**Type**	**Reset**	**Description**
	D_MASK			
15:8	LB_DQ_MASK	R/W	0x0	If set, the corresponding path will not perform lookback test.
7:4	RSVD	R	0	Reserved for future use
3	LB_INVERT_CLK	R/W	0x0	Normally, when loopback test is triggered and data pattern starts to be driven, Ckout_e will become 1 and Ckout_o stays at 0. If this bit is set, Ckout_e will become 1 and Ckout_o stays at 0. Only valid when bit[28] is set.
2	CLK_PASSTH_DS	R/W	0x0	Controls whether the host needs to send clock control signals (wr_ck_o/wr_ck_e/wr_ck_oe) to the PHY during loopback mode. 0: No need to output clock control signals to PHY. Used for HS400 mode with DS test, as the DS in loopback mode comes from the PHY's internal free-running clock (cki). 1: Required for other working modes in loopback mode. In this case, wr_ck_oe will be 1T earlier than wr_ck_o/wr_ck_e.
1	LB_PATTERN_SEL	R/W	0x0	0: Use programmable 32-bit pattern; 1: PRBS7
0	LB_MODE_EN	R/W	0x0	Enter Loopback Mode. Normally, host driver shall only change this bit when no other operation is processing.

10.5.4.55 SDHC_PHY_LBCNT_REG REGISTER

Offset: 0x188				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	LB_COMP_CNT	R/W	0x0	This field specifies the number of bits to be compared during the current loopback testing. - Once loopback testing starts, the host controller decrements this counter until it reaches 0. - Please note that after SW writes this field, the host will dynamically control this counter value. - During the loopback test process, this counter may be changed cycle by cycle.

10.5.4.56 SDHC_PHY_LBSTS_REG REGISTER

Offset: 0x18C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	R	0	Reserved for future use
21	LB_CLK_STUCK0_ERR	RC	0x0	CLK Path is stuck at 0
20	LB_CLK_STUCK1_ERR	RC	0x0	CLK Path is stuck at 1
19:18	RSVD	R	0	Reserved for future use
17	LB_CMD_EVEN_ERR	RC	0x0	If set, the CMD line path failed the Loopback testing
16	LB_CMD_ODD_ERR	RC	0x0	If set, the CMD line path failed the Loopback testing
15:8	LB_DQ_EVEN_ERR	RC	0x0	If set, the corresponding data path failed Loopback testing
7:0	LB_DQ_ODD_ERR	RC	0x0	If set, the corresponding data path failed Loopback testing

10.5.4.57 CQE_CQBDCTRL_REG0 REGISTER

Offset: 0x1F0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CQE_FSM_RST	R/W	0x0	- This is a backdoor register allowing software to force the CQE state machine into a stable IDLE state if the hardware encounters issues. - 1: Forces CQE FSM into IDLE state. - 0: No effect.
30:4	RSVD	R	0	Reserved for future use
3:0	CQE_DEBUG_SEL	R/W	0x0	These 4 bits field register selects which internal 32-bit debug bus signals are presented in the 0x1f4 register: - 4'b0000: Slot index & FSM informations - 4'b0001: Slot index informations - 4'b0010: Internal task slot fetch signals - 4'b0011: Internal task ready signals **Note.** This field is mainly intended for hardware designers to debug.

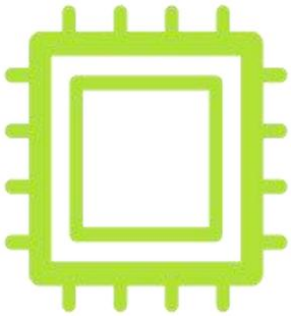
10.5.4.58 CQE_CQBDCTRL_REG1 REGISTER

Offset: 0x1F4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CQE_D	R	0x0	32-bit debug signal information output to software.

Offset: 0x1F4				
Bits	**Field**	**Type**	**Reset**	**Description**
	EBUG_INFO			**Note.** These register values may change on each cycle because the internal design logic of the CQE hardware may still be running when the software reads these values. To determine which 32-bit debug bus information is selected for output to this field, refer to 0x1F0<3:0>

Chapter 11

Video & Graphics



Key Stone® K1 User Manual

11.1 Video Subsystem

11.1.1 Video Processing Unit (VPU)

11.1.1.1 Introduction

The Video Processing Unit (VPU) is a video accelerator engine with two cores designed for decoding and encoding multiple video standards. It includes a host CPU to run firmware to control the hardware engine of functions, such as bit stream parsing, control of video hardware sub-blocks and error resilience.

Moreover, VPU is designed to optimally share most of the sub-blocks that are used in common for video processing, which contributes to the ultra-low power and low gate count.

VPU is connected with system AXI bus to access system DDR and connected with APB for configuration.

The VPU can work at up to 819MHz clock frequency, and supports a wide range of video standards, including H.265, H.264, VP8, VP9, MPEG4, MPEG2 and H263. It supports simultaneous

- Encoding and decoding at 1080P@60fps
- H264/H265 encoding at 1080P@30fps and H264/H265 decoding at 4K@30fps

11.1.1.2 VPU execution

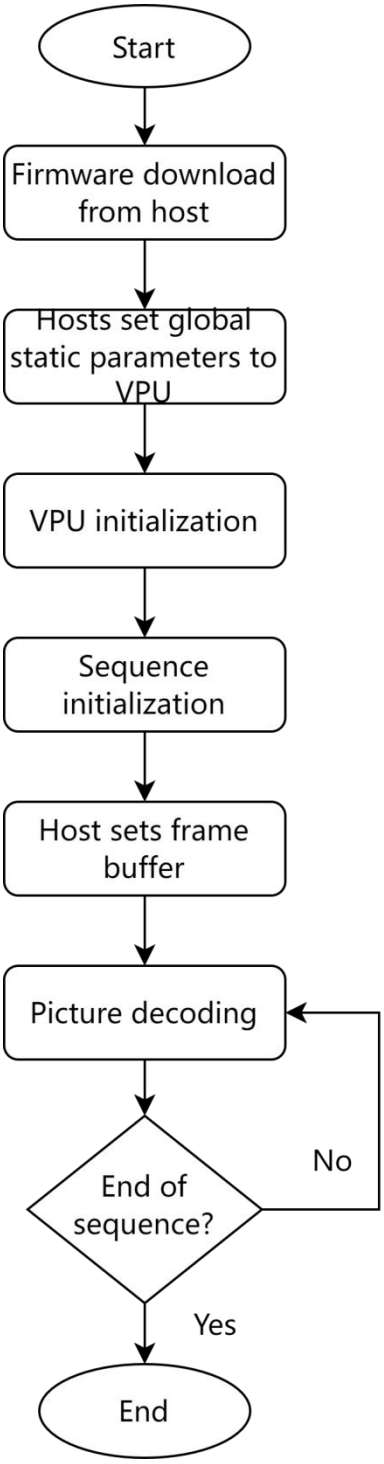
The video codec core block executes the actual decoding and encoding for each standard by using hardwired logic. Among them, Macroblock Sequencer is the main controller that schedules process flows of the sub-blocks, and aims to reduce loads on the processor and complexity of the firmware.

As mentioned, several standard-independent blocks share common logics while they are in operation in order to ensure efficiency and streamlined performance.

The APB3 and AXI buses serve as the primary interfaces between the VPU and the external system, in particular:

- APB3 handles the control interface
- AXIx is used for DDR data transfer.

The VPU has a 128-bit AXI bus interface. VPU basically works as depicted below.



To be highlighted:

- Host downloads firmware to the internal program memory of VPU
- Host sets global static parameters such as code, work, parameter buffer for initializing the VPU
- VPU is initialized
- Host commands the VPU to start sequence initialization process (SEQ INIT process)
- Host registers frame buffer information to VPU (FRAME BUF set process)

- Host gives VPU a picture decoding or encoding command sequentially (PIC RUN process)
- If there is no bitstream to decode, the VPU terminates decoding (PIC END process)
- in order to provide conveniently the pinout description of K1 in the following subsections.

11.1.1.3 Video Encoder

11.1.1.3.1 Encoding Features

- Configurable Arm Frame Buffer Compression (AFBC) 1.0 or 1.2 for input
- Support for YUV422 and YUV420 AFBC block split for 16 x 16
- Support for stride (not applicable to AFBC input formats)
- Horizontal and vertical mirroring (not applicable to AFBC input formats)
- Optional source frame rotation in 90-degree steps before encoding (not applicable to AFBC input format)

  Note. If YUV422 is rotated by 90 or 270 degrees and not converting to YUV420, the result will be converted to YUV440.
- Encoding support for the following source-frame input formats:
  - 1-plane YUV422, scan-line format, interleaved in YUYV or UYVY order

    Note. YUV422 input scan be converted to YUV420
  - 1-plane RGB (8-bit) in byte-address order: RGBA, BGRA, ARGB or ABGR
  - 2-plane YUV420, scan-line format, with chroma interleaved in UV or VU order
  - 3-plane YUV420, scan-line format

    Note. 3-plane format is supported for testing purposes only, and should not be used for optimal performance
  - AFBC YUV422
  - AFBC YUV420

11.1.1.3.2 Supported Encoding Formats

- HEVC (H.265) Main
- H.264 Baseline Profile (BP)
- H.264 Main Profile (MP)
- H.264 High Profile (HP)
- VP8
- VP9 Profile 0

11.1.1.3.2.1 HEVC (H.265) Encoding Features

- Encoded bit stream is compliant with the HEVC (H.265) Main Profile
- Encoding speed of 1080p@60fps (dual cores at approximately 300 MHz)
- Bitrates up to 50MBit/s using a single core operating at 300MHz
- Max frame width and height: 4096 pixels
- 8-bit encoding with I, P, and B frames
- Progressive encoding with 64×64 CTU size
- Support for tiled mode up to four tiles with horizontal splits only
- Wave front parallel encoding
- Motion Estimation (ME) search window dimensions: ±128 pixels horizontally, ±64 pixels vertically
- ME search precision: down to Quarter Picture Element (QPEL) resolution
- Luma intra-modes: 8×8, 16×16, and 32×32
- Chroma intra-modes: 4×4, 8×8, and 16×16
- Inter-modes: 8×8, 16×16, and 32×32
- Transform size for luma: 8×8, 16×16, and 32×32
- Transform size for chromas: 4×4, 8×8, and 16×16
- Skipped CUs and Merge modes
- Deblocking
- Sample Adaptive Offset (SAO)
- Constrained intra-prediction selectable
- Fixed Quantization Parameters (QP) or rate-controlled operation.
- Rate control uses a leaky bucket model based on bitrate and buffer size settings
- Long term reference frame support
- Selectable intra-frame refresh interval
- Slice insertion on a CTU row granularity
- Selectable limits for the search window and split options
- Encoders do not prevent the output from exceeding the maximum number of bits per CTU

11.1.1.3.2.2 H.264 Encoding Features

- Encoded bitstream is compliant with the Baseline, Main, High Profiles
- Encoding speed of 1080p@60fps (dual cores at approximately 300 MHz)
- Bitrates up to 50MBit/s using a single core operating at 300MHz
- Max frame width and height: 4096 pixels.
- Support for I, P, and B frames
- Support for progressive encoding
- Context Adaptive Binary Arithmetic Coding (CABAC) or Context Adaptive Variable Length Coding (CAVLC) entropy coding

  Note. B frames are not supported with CAVLC entropy coding

- Motion Estimation (ME) search window dimensions: ±128 pixels horizontally, ±64 pixels vertically
- ME search precision: down to Quarter Picture Element (QPEL) resolution
- Luma intra-modes: 4×4, 8×8, 16×16
- Chroma intra-modes: 8×8
- Inter-modes: 8×8, and 16×16
- Transform size: 4×4 and 8×8
- Support for skipped macroblocks
- Deblocking
- Constrained intra-prediction selectable
- Fixed QP operation or rate-controlled operation
- Rate control uses a leaky bucket model based on bitrate and buffer size settings
- Support for long term reference frame
- Selectable intra-frame refresh intervals
- Slice insertion granularity of 32-pixel high rows
- Possible to limit the search window and the macroblock split options
- Always enabled the escape option to prevent the emulation of a Network Abstraction Layer (NAL) unit start code regardless of the NAL packet format setting

Notes.

- For further details, please refer to ITU-T H.264 Annex B: VC-1 Compressed Video Bitstream Format and Decoding Process
- Encoders do not prevent the output from exceeding the maximum number of bits per macroblock

11.1.1.3.2.3 VP8 Encoding Features

- Encoding speed of 1080p@60fps (dual core at approximately 400 MHz)
- Bitrate up to 50MBit/s using a single core operating at 400MHz
- Max frame width and height: 2048 pixels
- Support for I and P frames
- Support for progressive encoding
- Motion Estimation (ME) search window dimensions: ±128 pixels horizontally, ±64 pixels vertically
- ME search precision: down to QPEL resolution
- Luma intra-modes: 4×4, 8×8, 16×16
- Chroma intra-modes: 8×8
- Inter-modes: 8x8, and 16×16
- Support for macroblocks skipping
- Deblocking
- Fixed QP operation or rate-controlled operation
- Rate control uses a leaky bucket model based on bitrate and buffer size settings

- Selectable intra-frame refresh intervals
- Possible to limit the search window and the macroblock split

11.1.1.3.2.4 VP9 Encoding Features

- Encoded bitstream is compliant with VP9 Profile 0 at 8-bit depth
- Encoding speed of 1080p@60fps (dual core at approximately 300 MHz)
- Bitrate up to 50MBit/s using a single core operating at 300MHz
- Max frame width and height: 4096 pixels
- Support for 8-bit sample depth
- Support for I and P frames
- Support for progressive encoding
- Tiled rows and columns
- Motion Estimation (ME) search window dimensions: ± 128 pixels horizontally, ± 64 pixels vertically
- ME search precision: down to Quarter Picture ELement (QPEL) resolution
- Luma intra-modes: 8×8, 16×16, and 32×32
- Chroma intra-modes: 4×4, 8×8, and 16×16
- Inter-modes: 8×8, 16×16, and 32×32
- Transform size for luma: 8×8, 16×16, and 32×32
- Transform size for chroma: 4×4, 8×8, and 16×16
- Support for superblocks skipping
- Deblocking
- Fixed QP operation or rate-controlled operation
- Rate control uses a leaky bucket model based on bitrate and buffer size settings
- Selectable intra-frame refresh intervals
- Support for implicit or explicit probability update using delayed contexts

11.1.1.4 Video Decoder

11.1.1.4.1 Decoding Features

- Support for the following source frame output formats:
  - 2-plane YUV420 scan line format: chroma interleaved in UV or VU order
  - 3-plane YUV420 scan line format

    Notes.

    - Support for 3-plane format is included for testing purposes only, do not use such max performance for normal applications
    - Ensure of correct alignment of YUV buffer and stride for optima performance
- YUV420 AFBC format, 8-bit color depth

- Configurable for AFBC 1.0 or AFBC 1.2 output
- Support for stride for scan-line formats only
- Decoded frame rotation is supported in 90-degree steps before output

  Note. Not applicable for AFBC output formats
- Support for output average luminance (brightness) and chrominance (color) values for each 32×32 pixel block in every displayed output frame

11.1.1.4.2 Supported Decoding Formats

- HEVC (H.265): Main Profile
- H.264: Baseline, Main, High Profile
- VP8
- VP9: Profile 0
- VC-1: SP/MP/AP
- MPEG4: SP/ASP
- MPEG2: MP
- H.263: Profile 0

11.1.1.4.2.1 HEVC (H.265) Decoding Features

- Fully compliance with the Main Profiles
- Support for 2160p@30fps using dual core operating at approximately 300MHz
- Capability of handling average bitrate up to 100MBit/s with a single core at 600MHz
- Max frame width and height: 4096 pixels
- Error concealment is performed for handling bit errors
- Output of relevant stream parameter information during decoding

11.1.1.4.2.2 H.264 Decoding Features

- Fully compliance with H.264 Baseline, Main, High and High 10 progressive Profiles
- For streams using Flexible Macroblock Ordering (FMO) or Arbitrary Slice Ordering (ASO) in Baseline Profile, it is used WVGA resolution with decoding speed of 30fps with a single core at 400MHz
- For streams without FMA and ASO, the decoding speeds are as follows:
  - 2160p@30fps using dual core at approximately 300MHz
  - 1080i@120fps using dual core at 400MHz
- For progressive streams:
  - Average bitrate up to 100MBit/s with a single core at 600MHz
  - Max frame width and height: 4096 pixels

- For interlaced streams:
  - Average bitrate up to 50MBit/s with a single core at 400MHz
  - Max frame width: 2048 pixels
  - Max frame height: 4096 pixels
- Error concealment is performed for managing bitstream errors
- Output of relevant stream parameter information during decoding
- Always enabled the escape option to prevent the emulation of a Network Abstraction Layer (NAL) unit start code, regardless of the NAL packet format setting

  Note. For further details, please refer to ITU-T H.264 Annex B: VC-1 Compressed Video Bitstream Format and Decoding Process

11.1.1.4.2.3 VP8 Decoding Features

- Fully compliance with the VP8 Specification
- Support for decoding speed of 1080p@60fps using dual core at approximately 400MHz
- Average bitrate up to 50MBit/s with single core at 400MHz
- Max frame width and height: 2048 pixels
- Error concealment is performed for managing bitstream errors

11.1.1.4.2.4 VP9 Decoding Features

- Fully compliance with Profile 0
- Support for decoding speed of 2160p@30fps using dual core at approximately 300MHz and assuming no non-visible and no Alt-Ref frames
- Support for decoding speed of 2160p@30fps using dual core at approximately 400MHz and assuming an Alt-Ref frame distance of 4
- Average bitrate up to 60MBit/s using single core at 600MHz
- Max frame width and height: 4096 pixels
- Error concealment is performed for managing bitstream errors
- Output of relevant stream parameter information during decoding

11.1.1.4.2.5 VC-1 Decoding Features

- Fully compliance with VC-1 Simple, Main, and Advanced Profiles
- Support for decoding speeds of 1080p@60fps and 1080i@120fps using dual core at approximately 400MHz
- Average bitrate up to 40MBit/s with single core at 400MHz
- Max frame width: 2048 pixels
- Max frame height: 4096 pixels

- Error concealment is performed for managing bitstream errors

Notes.

- Advanced Profile bitstream data must always include the Encapsulation Mechanism regardless of the NAL packet format setting.
- For further details, please refer to SMPTE-421M-2006 Annex E.
- The range mapping feature of the VC-1 Advanced Profile does not apply to AFBC output.

11.1.1.4.2.6 MPEG4 Decoding Features

- Compliance with MPEG4 Simple Profile and Advanced Simple Profile
- Support for Global Motion Compensation (GMC) with a limitation of a single warp point
- Support for decoding speed of 1080p@60fps or 1080i@120fps using dual core at 400MHz
- Handling of average bitrate up to 20MBit/s with a single core operating at 400MHz
- Max frame width and height: 2048 pixels
- Error concealment is performed for managing bitstream errors

11.1.1.4.2.7 MPEG2 Decoding Features

- Compliance with MPEG2 Main Profile
- Support for decoding speed of 1080p@60fps or 1080i@120fps using dual core at 400MHz
- Handling of average bitrate up to 20MBit/s with single core operating at 400MHz
- Max frame width: 4906 pixels (2,048 pixels for interlaced stream)
- Max frame height: 4096 pixels
- Error concealment is performed for managing bitstream errors

11.1.1.4.2.8 H.263 Decoding Features

- Compliance with H.263 Profile 0
- Support for decoding speed of 1080p@60fps using dual core at approximately 400MHz
- Handling of average bitrates up to 20MBit/s with single core operating at 400MHz
- Max frame width and height: 2048 pixels
- Error concealment is performed for managing bitstream errors

11.2 Image Subsystem

11.2.1 Graphics Processing Unit (GPU)

11.2.1.1 Introduction

- GPU is built around multi-threaded Unified Shading Clusters (USCs) that features an ALU architecture with high SIMD efficiency, and supports tile-based deferred rendering with concurrent processing of multiple tiles.
- The GPU engine handles a number of different workloads, including:
- 3D graphics workload: vertex and pixel data processing for rendering 3D scenes
- Compute workload (GP-GPU): general purpose data processing

Note. 3D graphics and compute (with barriers) workloads cannot be overlapped at the same time

The GPU core has an AXI 128bits bus for accessing SOC's DDR memory with a core frequency of up to 819MHz.

11.2.1.2 General Features

- Base architecture which is fully compliant with the following APIs:
  - OpenGL ES 1.1/3.2
  - EGL1.5
  - OpenCL 3.0
  - Vulkan 1.3
- Tile-based deferred rendering architecture (TBDR) for 3D graphics workloads, with concurrent processing of multiple tiles where data are processed in two phases as follows:
  - Geometry Processing Phase: involvement of vertex operations such as transformation and vertex lighting as well as dividing a 3D scene into tiles
  - Fragment Processing Phase: involvement of pixel operations such as rasterization, texturing and shading of pixels
- Programmable high quality image anti-aliasing
- Fine grain triangle culling
- Support for Digital Right Management (DRM) security
- Support for GPU virtualization as follows:
  - Up to 8 virtual GPUs
  - IMG hyperlane technology with 8 hyperlanes available
  - Separate IRQs per OSI
- Multi-threaded Unified Shading Cluster (USC) engine incorporating pixel shader, vertex shader and GP-GPU (compute shader) functionality
- USC incorporates an ALU architecture with high SIMD efficiency
- Fully virtualized memory addressing (up to 64 GB address space), supporting unified memory architecture
- Fine-grained task switching, workload balancing and power management

- Advanced DMA driven operation for minimum host CPU interaction
- Cache type as follows:
  - 32KB System Level Cache (SLC)
  - Specialized Texture Cache Unit (TCU)
- Compressed Texture Decoding
- Lossless and/or visually lossless low area image compression, using imagination frame buffer compression and decompression (TFBC) algorithm
- Dedicated processor for B-Series core firmware execution
- Single-threaded firmware processor with a 2KB instruction cache and a 2KB data cache
- Separated power island for the firmware processor
- On-chip performance, power and statistics registers

11.2.1.3 3D Graphics Features

- **Rasterization**
  - Deferred pixel shading
  - On-chip tile floating point depth buffer
  - 8-bit stencil with on-chip tile stencil buffer
  - Maximum 2 tiles in flight (per ISP)
  - 16 parallel depth/stencil tests per clock
  - 1 fixed-function rasterisation pipeline(s)
- **Texture Lookups**
  - Support for loading from source instruction
  - Texture write enabled through the Texture Processing Unit (TPU)
- **Filtering**
  - Point, bilinear and trilinear filtering
  - Anisotropic filtering
  - Corner filtering support for cube environment mapped textures and filtering across faces
- **Texture Formats**
  - ASTC LDR compressed texture format support
  - TFBC lossless and/or lossy compression format support for non-compressed textures and YUV textures
  - ETC
  - YUV planar support
- **Resolution Support**
  - Max frame buffer size: 8K×8K
  - Max texture max size: 8K×8K

- **Anti-Aliasing**
  - Max 4× multisampling
- **Primitive Assembly**
  - Early hidden object removal
  - Tile acceleration
- **Render to Buffers**
  - Twiddled format support
  - Multiple On-Chip Render Targets (MRT)
  - Lossless and/or lossy frame buffer compression/decompression
  - Programmable geometry shader support
  - Direct geometry stream out (transform feedback)
- **Compute**
  - 1, 2 and 3 dimensional compute primitives
  - Block DMA to/from USC Common Store (for local data)
  - Per task input data DMA (to USC Unified Store)
  - Conditional execution
  - Execution fences
  - Compute workload can be overlapped with any other workload
  - Round to nearest even

11.2.1.4 Unified Shading Cluster (USC) Features

- 2 ALU pipelines
- 8 parallel instances per clock
- Local data, texture and instruction caches
- Variable length instruction set encoding
- Full support for OpenCL™ atomic operations
- Scalar and vector SIMD execution model
- USC F16 Sum-of-Products Multiply-Add (SOPMAD) Arithmetic Logic Unit (ALU)

11.2.2 V2D

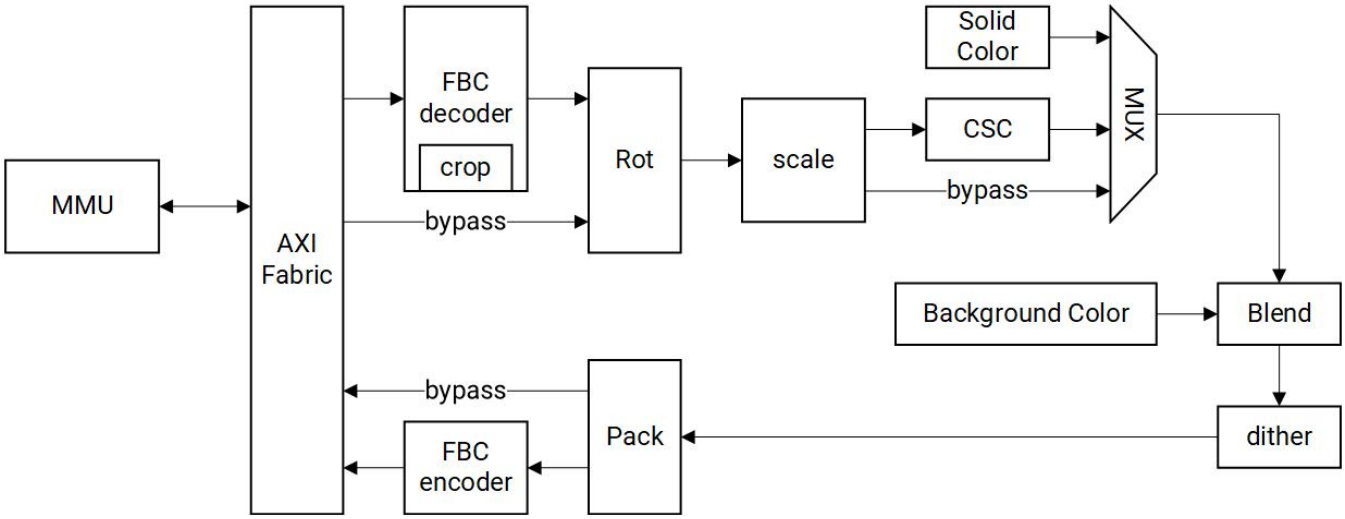
11.2.2.1 Features

- Support for upscaling (up to 8x) and downscaling (down to 1/8x)
- Support for 0°, 90°, 180°, 270° rotation as well as mirror and flip option
- Support for simple layer and background blending
- Support for image cropping

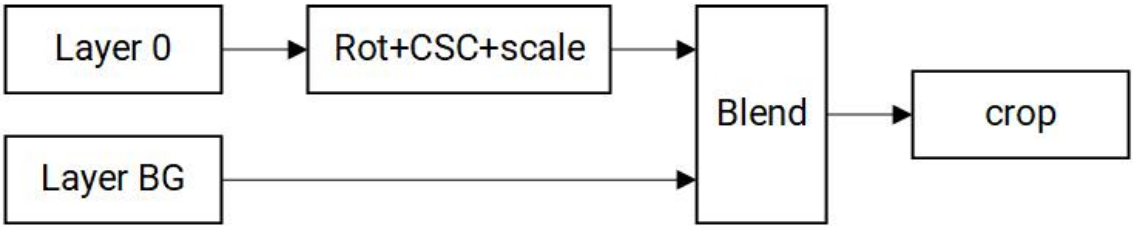
- Support for fetch solid color
- Support for color space conversion between RGB, BT601 and BT709 (both narrow and full range)
- 4656x3596 or 4672x3504 as max NV12 resolution
- Support for dithering for smoother color transitions
- Support for MMU
- Support for APB3 and AXI3 bus interfaces
- Support for the following **input formats**:
  - RGB888 (with optional RB swap)
  - RGBX888 (with optional RB swap)
  - RGBA8888 (with optional RB swap)
  - ARGB8888 (with optional RB swap)
  - RGB565 (with optional RB swap)
  - RGBA5658 (with optional RB swap)
  - ARGB8565 (with optional RB swap)
  - A8 (8-bit alpha image)
  - Y8 (8-bit gray image)
  - YUV420 semi-planar (UV can swap)
  - AFBC 16x16 RGBA8888 (layerout0 split and non-split)
  - AFBC 16x16 NV12 (layerout1 split and non-split)
- Support for the following **output formats**:
  - RGB888 (with optional RB swap)
  - RGBX888 (with optional RB swap)
  - RGBA8888 (with optional RB swap)
  - ARGB8888 (with optional RB swap)
  - RGB565 (with optional RB swap)
  - RGBA5658 (with optional RB swap)
  - ARGB8565 (with optional RB swap)
  - A8 (8-bit alpha image)
  - Y8 (8-bit gray image)
  - YUV420 semi planar (UV can swap)
  - AFBC 16x16 RGBA8888 (layerout0 split and non-split)
  - AFBC 16x16 NV12 (layerout1 split and non-split)

11.2.2.2 Block Diagram

The micro-architecture of the V2D subsystem is depicted below.



Instead, the typical V2D work scenario is depicted below.

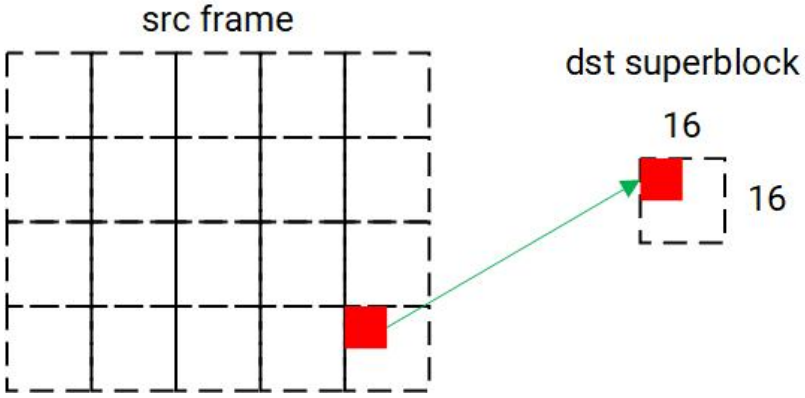


11.2.2.3 Functions

11.2.2.3.1 Fetch Data

The process of fetching a 16×16 block of data from a source frame (src frame) and related mapping to the destination superblock (dst superblock) is depicted below, where

- **AFBC**: fetch rect left, top, width, height 4 align
- **Non-AFBC**: fetch rect left, top, width, height 1 align



The code for fetching data for displaying is listed below, and the details of the specific variables and registers involved are tabled immediately after.

```
Input param: Rect_left, Rect_top, Rect_width, Rect_height
Rect_width = Rect_left%4 + Rect_width;
Rect_height = Rect_top%4 + Rect_height;
Rect_left = Rect_left/4 × 4;
Rect_top = Rect_top/4 × 4;
if LayerX_format == YUV420
{
   Rect_width  = ALIGN(Rect_left %2 + Rect_width, 2);
   Rect_height   = ALIGN(Rect_top%2 + Rect_height, 2);
   Rect_left = Rect_left/2 × 2;
   Rect_top = Rect_top/2 × 2;
}
Take the data in the Rect
Loop every pixel in Rect
{
   if LayerX_format == YUV420
   {
      upsample YUV420 to YUV444;
      c0 = channel 0; // Y
      c1 = channel 1; // U
      c2 = channel 2; // V
      c3 = 0xff;
   }
   if LayerX_format == RGB888
   {
      c0 = channel 0; // R
      c1 = channel 1; // G
      c2 = channel 2; // B
      c3 = 0xff; // A
   }
   if LayerX_format == RGBX8888
   {
```

```
    c0 = channel 0; // R
    c1 = channel 1; // G
    c2 = channel 2; // B
    c3 = 0xff; // A
}
if LayerX_format == RGBA8888
{
    c0 = channel 0; // R
    c1 = channel 1; // G
    c2 = channel 2; // B
    c3 = channel 3; // A
}
if LayerX_format == ARGB8888
{
    c0 = channel 1; // R
    c1 = channel 2; // G
    c2 = channel 3; // B
    c3 = channel 0; // A
}
if LayerX_format == RGB565
{
    c0 = byte_low &0x1f; // R5
    c1 = ((byte_high << 3) | (byte_low >> 5)) & 0x3f; // G6
    c2 = (byte_high >> 3) &0x1f; // B5
    c0 = (c0 << 3) | (c0 >> 2); // R8
    c1 = (c1 << 2) | (c1 >> 4); // G8
    c2 = (c2 << 3) | (c2 >> 2); // B8
    c3 = 0xff; // A8
}
if LayerX_format == YUV420 && LayerX_swap == 1
    Swap(c1, c2);
else if LayerX_swap == 1
    Swap(c0, c2);
Index = Rect_y%16 × 16 + Rect_x;
```

```
    data[0][index] = c0;
    data[1][index] = c1;
    data[2][index] = c2;
    data[3][index] = c3;
}
```

Variable	Bit	Comment
Rect_left Rect_top	16bit unsigned	Range [0, 65535]
Rect_width Rect_height	5bit unsigned	Range [1, 16]
Rect_x Rect_y	16bit unsigned	Range [0, 65535] Pixel global position
c0, c1, c2, c3	8bit unsigned	Range [0, 255]
byte_low byte_high	8bit unsigned	Range [0, 255] byte_low: lower byte in RGB565 byte_high: higher byte in RGB565
data[4][256]	8bit unsigned × 4 × 256	Range [0, 255]
index	8bit unsigned	Range [0, 255]

Register	Comment
LayerX_format	X is either 0 or 1, refer to module register
LayerX_swap	X is either 0 or 1, refer to module register

11.2.2.3.2 Solid Color

The code for applying the solid color within a specific rectangle is listed below, and the details of the specific variables and registers involved are tabled immediately after.

Notes.

- If the register LayerX_solid is enabled, the fetched data is set to solid R, G, B, A
- The coordinates of the fetch rect and solid rect are updated after rotation

```
Input param: Rect_left, Rect_top, Rect_width, Rect_height.
if LayerX_solid_enable = 1
{
    c0 = LayerX_solid_R;
```

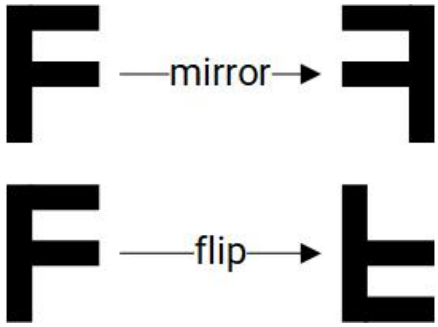
```
    c1 = LayerX_solid_G;
    c2 = LayerX_solid_B;
    c3 = LayerX_solid_A;
    Loop all pixels in Rect
    {
        Index = Rect_y%16 × 16 + Rect_x;
        data[0][index] = c0;
        data[1][index] = c1;
        data[2][index] = c2;
        data[3][index] = c3;
    }
    Skip fetch data from ddr
}
```

Variable	Bit	Comment
Rect_left, Rect_top	16bit unsigned	Range [0, 65535]
Rect_width, Rect_height	5bit unsigned	Range [1, 16]
Rect_x, Rect_y	16bit unsigned	Range [0, 65535] Pixel global position
c0, c1, c2, c3	8bit unsigned	Range [0, 255]
data[4][256]	8bit unsigned × 4 × 256	Range [0, 255]
index	8bit unsigned	Range [0, 255]

Register	Comment
LayerX_solid_enable	X is 0 or 1, refer to module register
LayerX_solid_R	X is 0 or 1, refer to module register
LayerX_solid_G	X is 0 or 1, refer to module register
LayerX_solid_B	X is 0 or 1, refer to module register
LayerX_solid_A	X is 0 or 1, refer to module register

11.2.2.3.3 Rotation

Support for 0°, 90°, 180°, 270° rotation (performed clockwise) as well as mirror and flip option, as depicted below (example).



The code for rotating, mirroring and flipping graphical content is listed below, and the details of the specific variables and registers involved are tabled immediately after).

```
Input param: Rect_left, Rect_top, Rect_width, Rect_height, data_in[4][256].
Output: Block_rect_left, Block_rect_top,  Block_rect_width,  Block_rect_height, data_out[4][256].
Block_rect_left = Rect_left;
Block_rect_top = Rect_top;
Block_rect_width = Rect_width;
Block_rect_height = Rect_height;
if LayerX_degree == ROT_0{
    Org_rect_left = Rect_left;
    Org_rect_top = Rect_top;
    Org_rect_width = Rect_width;
    Org_rect_height = Rect_height;
}
if LayerX_degree == ROT_90{
    Org_rect_left = Rect_top;
    Org_rect_top = ALIGN(LayerX_height,16) - Rect_left - Rect_width;
    Org_rect_width = Rect_height;
    Org_rect_height = Rect_width;
}
if LayerX_degree == ROT_180{
    Org_rect_left = ALIGN(LayerX_width,16) - Rect_left - Rect_width;
    Org_rect_top = ALIGN(LayerX_height,16) - Rect_top - Rect_height;
    Org_rect_width = Rect_width;
    Org_rect_height = Rect_height;
}
```

```
if LayerX_degree == ROT_270{
    Org_rect_left = ALIGN(LayerX_width,16)-Rect_top-Rect_height;
    Org_rect_top = Rect_left;
    Org_rect_width = Rect_height;
    Org_rect_height = Rect_width;
}
if LayerX_degree == ROT_MIRROR{
    Org_rect_left = ALIGN(LayerX_width,16) - Rect_left - Rect_width;
    Org_rect_top = Rect_top;
    Org_rect_width = Rect_width;
    Org_rect_height = Rect_height;
}
if LayerX_degree == ROT_FLIP{
    Org_rect_left = Rect_left;
    Org_rect_top = ALIGN(LayerX_height,16) - Rect_top - Rect_height;
    Org_rect_width = Rect_width;
    Org_rect_height = Rect_height;
}//fetch data in Org_rect
Fetch_data(Org_rect, &data_in[4][256]);
Loop all pixels in data_in{
    dst_index=jx16 + i;
    if LayerX_degree == ROT_0
        src_index=jx16 + i;
    if LayerX_degree == ROT_90
        src_index=(15-i)x16 + j;
    if LayerX_degree == ROT_180
        src_index=(15-j)x16 + (15-i);
    if LayerX_degree == ROT_270
        src_index= ix16+(15-j);
    if LayerX_degree == ROT_MIRROR
        src_index = jx16 + (15-i);
    if LayerX_degree == ROT_FLIP
        src_index = (15-j)x16 + i;
    data_out[0][dst_index]= data_in[0][src_index];
```

```
    data_out[1][dst_index]= data_in[1][src_index];
    data_out[2][dst_index]= data_in[2][src_index];
    data_out[3][dst_index]= data_in[3][src_index];
}
```

Variable	Bit	Comment
Rect_left, Rect_top	16bit unsigned	Range [0, 65535]
Rect_width, Rect_height	5bit unsigned	Range [1, 16]
Block_rect_left, Block_rect_top	16bit unsigned	Range [0, 65535]
Block_rect_width, Block_rect_height	5bit unsigned	Range [1, 16]
data_in[4][256], data_out[4][256]	8bit unsigned × 4 × 256	Range [0, 255]

Register	Bit	Comment
LayerX_degree	3bit unsigned	X is 0 or 1, refer to module register
LayerX_width, LayerX_height	16bit unsigned	X is 0 or 1, refer to module register

11.2.2.3.4 CSC

Support for Color Space Conversion (CSC) as per formats below:

- BT601 and BT709: conversion between narrow and full range
- RGB to YUV
- YUV to RGB

The conversion process transforms input channels into output channels by using a transformation matrix with clamping in order to ensure valid output values, i.e. within the range [0, 255].

For that purpose, the formulas below are implemented, and the details of the specific variables and registers involved are tabled immediately after.

[Firstly for computing the intermediate channel values]

- C0inter =

  (Layer_matrix[0][0] x C0in + Layer_matrix[0][1] x C1in + Layer_matrix[0][2] x C2in + 512)

  >>

  (10 + Layer_matrix[0][3])

- C1inter =

  (Layer_matrix[1][0] x C0in + Layer_matrix[1][1] x C1in + Layer_matrix[1][2] x C2in + 512)

  >>

  (10 + Layer_matrix[1][3])

- C2inter =

  (Layer_matrix[2][0] x C0in + Layer_matrix[2][1] x C1in + Layer_matrix[2][2] x C2in + 512)

  >>

  (10 + Layer_matrix[2][3])

[Then for clamping in order to ensure valid output values]

- C0out = clamp(C0inter, 0, 255)
- C1out = clamp(C1inter, 0, 255)
- C2out = clamp(C2inter, 0, 255)
- C3out = clamp(C3in, 0, 255)

Variable	Bit	Comment
C0in, C1in, C2in, C3in	8bit unsigned	Input channel
C0inter, C1inter, C2inter	10bit signed	Intermediate channel value
C0out, C1out, C2out, C3out	8bit unsigned	Output channel

Register	Index	Bit	Comment
LayerX_CSC_enable	-	1bit unsigned	0: disable 1: enable
Layer_matrix[#][#]	0-11	13bit signed	Range [-4096, 4095]

In the code, the conversion process is applied with the following condition:

```
if LayerX_CSC_enable == 0
   skip CSC function
```

11.2.2.3.5 Scaling

The scaling operation follows a systematic superblock-based approach, where

- The first four superblocks are outputted horizontally then vertically
- After the vertical output is completed, the process restarts from the first row of superblocks

11.2.2.3.6 Storing

A 16×16 image block can be stored in DDR memory, however only the portion that falls within the output crop region is stored which is converted to the specified output color format, such as YUV, RGB, etc.

The code for storing an image block is listed below, and the details of the specific variables and registers involved are tabled immediately after.

```
Input param: Rect_left, Rect_top, Rect_width, Rect_height, data_in[4][256]
if output_format == YUV420
{
   s0=0;
   s1=1;
   s2=2;
   if(output_swap){
      Swap(s1, s2);
   }
   Loop all pixels by 2x2{
      if(pixel in output_crop_rect){
         Y00=data_in[s0][pixel_index00];
         Y01=data_in[s0][pixel_index01];
         Y10=data_in[s0][pixel_index10];
         Y11=data_in[s0][pixel_index11];
         U00=data_in[s1][pixel_index00];
         U01=data_in[s1][pixel_index01];
         U10=data_in[s1][pixel_index10];
         U11=data_in[s1][pixel_index11];
         V00=data_in[s2][pixel_index00];
         V01=data_in[s2][pixel_index01];
         V10=data_in[s2][pixel_index10];
         V11=data_in[s2][pixel_index11];
         Downsample and store to output frame
```

```
            U=(U00+U01+U10+U11+2)>>2;
            V=(V00+V01+V10+V11+2)>>2;
        }
    }
}
if output_format == RGB888
{
    s0=0;
    s1=1;
    s2=2;
    if(output_swap){
        Swap(s0, s2);
    }
    Loop all pixels{
        if(pixel in output_crop_rect){
            R=data_in[s0][pixel_index];
            G=data_in[s1][pixel_index];
            B=data_in[s2][pixel_index];
            store to output frame.
        }
    }
}
if output_format == RGBX888 || output_format == RGBA888
{
    s0=0;
    s1=1;
    s2=2;
    s3=3;
    if(output_swap){
        Swap(s0, s2);
    }
    Loop all pixels{
        if(pixel in output_crop_rect){
            R=data_in[s0][pixel_index];
```

```
            G=data_in[s1][pixel_index];
            B=data_in[s2][pixel_index];
            A=data_in[s3][pixel_index];
            store to output frame.
        }
    }
}
if output_format == ARGB8888
{
    s0=3;
    s1=0;
    s2=1;
    s3=2;
    if(output_swap){
        Swap(s1, s3);
    }
    Loop all pixels{
        if(pixel in output_crop_rect){
            R=data_in[s0][pixel_index];
            G=data_in[s1][pixel_index];
            B=data_in[s2][pixel_index];
            A=data_in[s3][pixel_index];
            store to output frame.
        }
    }
}
```

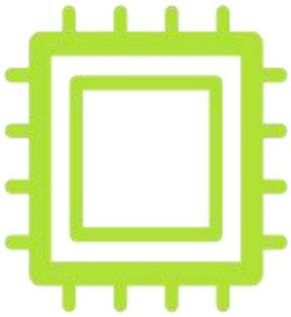
Variable	Bit	Comment
Rect_left Rect_top	16bit unsigned	Range [0, 65535]
Rect_width Rect_height	5bit unsigned	Range [1, 16]
pixel_index	8bit unsigned	Range [0, 65535]
s0, s1, s2, s3	8bit unsigned	Range [0, 255]

Variable	Bit	Comment
Y00, Y01, Y10, Y11, U00, U01, U10, U11, V00, V01, V10, V11, U, V, R, G, B, A	8bit unsigned	Range [0, 255]
data_in[4][256]	8bit unsigned × 4 × 256	Range [0, 255]

Register	Bit	Comment
Output_format	3bit unsigned	0: RGB888 (R at low address, B at high address) 1: RGBX8888 2: RGBA8888 3: ARGB8888 (A at low address, B at high address) 5: yuv420sp (U at low address, V at high address)
Output_swap	1bit unsigned	0: No swap 1: RGB swap RB, YUV swap UV
Output_layout	1bit unsigned	0: Linear 1: FBC compressed
Output_crop_left	16bit unsigned	Range [0, 65534] crop_left < output_left + output_width
Output_crop_top	16bit unsigned	Range [0, 65534] crop_top < output_top + output_height
Output_crop_width	16bit unsigned	Range [1, 65535] crop_left + crop_wdith ≤ output_left + output_width
Output_crop_height	16bit unsigned	Range [1, 65535] crop_top + crop_height ≤ output_top + output_height

Chapter 12

Display Subsystem



Key Stone® K1 User Manual

12.1 Display Controller

12.1.1 Introduction

The Display Controller is a hardware block that is used to transfer display data from the display's internal memory to the DSI controller. It supports one independent display device through MIPI DSI.

12.1.2 Features

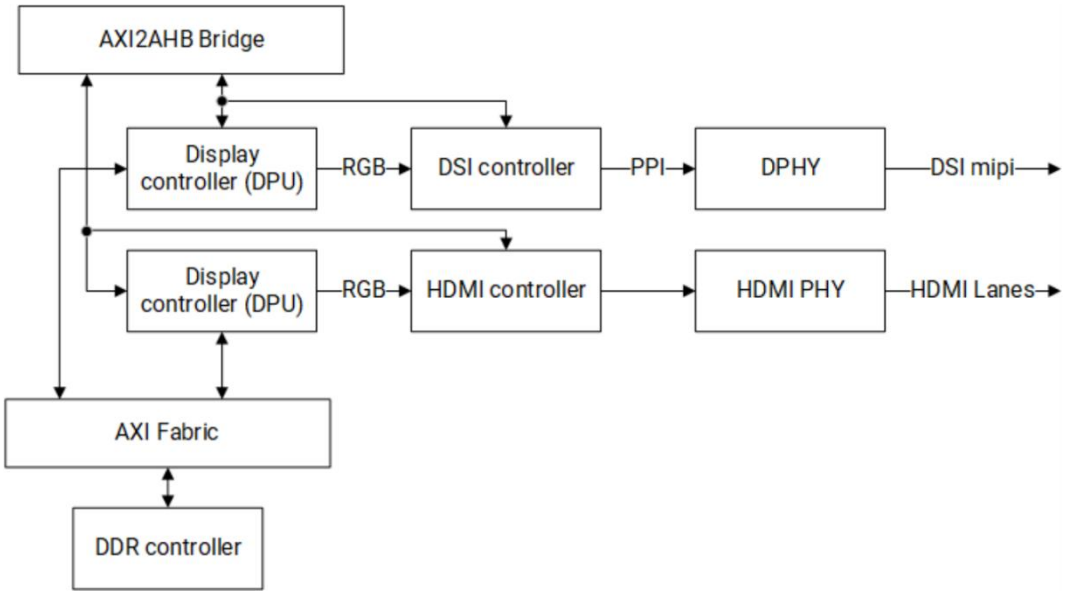
- Support for up to Full HD (1920x1080@60fps)
- Support for up to 4-full-size-layer composer and maximum 8 layer-composers by up-down layer reuse in the RDMA channel
- Support for **cmdlist** mechanism allowing hardware register parameters to be configured
- Support for concurrent write-back operations with both raw and AFBC format
- Support for dithering, cropping, rotation in write-back path
- Advanced MMU (virtual address) mechanism for nearly no page missing during 90° and 270° rotation
- Support for color keying and solid color generation
- Support for both advanced error diffusion and pattern-based dithering for the panel
- Support for both AFBC and raw format image sources
- Color saturation and contrast enhancement
- Support for both video mode and **cmd** mode (with frame buffer in LCM) for the panel
- Support for dynamic DDR frequency adjustment with an embedded DFC buffer
- Support for the following **input formats** (see also the map shown immediately after):
  - A2BGR101010, A2RGB101010, BGR101010A2, RGB101010A2
  - ABGR8888, ARGB8888, BGRA8888, RGBA8888
  - XBGR8888, XRGB8888, BGRX8888, RGBX8888
  - BGR888, RGB888, ABGR1555, RGBA5551, BGR565/RGB565
  - XYUV_444_P1_8, XYUV_444_P1_10, YVYU_422_P1_8, VYUY_422_P1_8
  - YUV_420_P2_8, YUV_420_P3_8

Format Name	Byte Address and Order															
	127 15 120	119 14 112	111 13 104	103 12 96	95 11 88	87 10 80	79 9 72	71 8 64	63 7 56	55 6 48	47 5 40	39 4 32	31 3 24	23 2 16	15 1 8	7 0 0
ABGR_2101010	A3 [1:0]	B3[9:0]	G3[9:0]	R3[9:0]	A2 [1:0]	B2[9:0]	G2[9:0]	R2[9:0]	A1 [1:0]	B1[9:0]	G1[9:0]	R1[9:0]	A0 [1:0]	B0[9:0]	G0[9:0]	R0[9:0]
ARGB_2101010	A3 [1:0]	R3[9:0]	G3[9:0]	B3[9:0]	A2 [1:0]	R2[9:0]	G2[9:0]	B2[9:0]	A1 [1:0]	R1[9:0]	G1[9:0]	B1[9:0]	A0 [1:0]	R0[9:0]	G0[9:0]	B0[9:0]
BGRA_2101010	B3[9:0]	G3[9:0]	R3[9:0]	A3 [1:0]	B2[9:0]	G2[9:0]	R2[9:0]	A2 [1:0]	B1[9:0]	G1[9:0]	R1[9:0]	A1 [1:0]	B0[9:0]	G0[9:0]	R0[9:0]	A0 [1:0]
RGBA_2101010	R3[9:0]	G3[9:0]	B3[9:0]	A3 [1:0]	R2[9:0]	G2[9:0]	B2[9:0]	A2 [1:0]	R1[9:0]	G1[9:0]	B1[9:0]	A1 [1:0]	R0[9:0]	G0[9:0]	B0[9:0]	A0 [1:0]
ABGR_8888	A3[7:0]	B3[7:0]	G3[7:0]	R3[7:0]	A2[7:0]	B2[7:0]	G2[7:0]	R2[7:0]	A1[7:0]	B1[7:0]	G1[7:0]	R1[7:0]	A0[7:0]	B0[7:0]	G0[7:0]	R0[7:0]
ARGB_8888	A3[7:0]	R3[7:0]	G3[7:0]	B3[7:0]	A2[7:0]	R2[7:0]	G2[7:0]	B2[7:0]	A1[7:0]	R1[7:0]	G1[7:0]	B1[7:0]	A0[7:0]	R0[7:0]	G0[7:0]	B0[7:0]
BGRA_8888	B3[7:0]	G3[7:0]	R3[7:0]	A3[7:0]	B2[7:0]	G2[7:0]	R2[7:0]	A2[7:0]	B1[7:0]	G1[7:0]	R1[7:0]	A1[7:0]	B0[7:0]	G0[7:0]	R0[7:0]	A0[7:0]
RGBA_8888	R3[7:0]	G3[7:0]	B3[7:0]	A3[7:0]	R2[7:0]	G2[7:0]	B2[7:0]	A2[7:0]	R1[7:0]	G1[7:0]	B1[7:0]	A1[7:0]	R0[7:0]	G0[7:0]	B0[7:0]	A0[7:0]
XBGR_8888	X[7:0]	B3[7:0]	G3[7:0]	R3[7:0]	X[7:0]	B2[7:0]	G2[7:0]	R2[7:0]	X[7:0]	B1[7:0]	G1[7:0]	R1[7:0]	X[7:0]	B0[7:0]	G0[7:0]	R0[7:0]
XRGB_8888	X[7:0]	R3[7:0]	G3[7:0]	B3[7:0]	X[7:0]	R2[7:0]	G2[7:0]	B2[7:0]	X[7:0]	R1[7:0]	G1[7:0]	B1[7:0]	X[7:0]	R0[7:0]	G0[7:0]	B0[7:0]
BGRX_8888	B3[7:0]	G3[7:0]	R3[7:0]	X[7:0]	B2[7:0]	G2[7:0]	R2[7:0]	X[7:0]	B1[7:0]	G1[7:0]	R1[7:0]	X[7:0]	B0[7:0]	G0[7:0]	R0[7:0]	X[7:0]
RGBX_8888	R3[7:0]	G3[7:0]	B3[7:0]	X[7:0]	R2[7:0]	G2[7:0]	B2[7:0]	X[7:0]	R1[7:0]	G1[7:0]	B1[7:0]	X[7:0]	R0[7:0]	G0[7:0]	B0[7:0]	X[7:0]
ABGR_1555	A7 [0] B7 [4:0]	G7 [4:0] R7 [4:0]	A6 [0] B6 [4:0]	G6 [4:0] R6 [4:0]	A5 [0] B5 [4:0]	G5 [4:0] R5 [4:0]	A4 [0] B4 [4:0]	G4 [4:0] R4 [4:0]	A3 [0] B3 [4:0]	G3 [4:0] R3 [4:0]	A2 [0] B2 [4:0]	G2 [4:0] R2 [4:0]	A1 [0] B1 [4:0]	G1 [4:0] R1 [4:0]	A0 [0] B0 [4:0]	G0 [4:0] R0 [4:0]
RGBA_1555	R7 [4:0]	G7 [4:0]	B7 [4:0]	A7 [0]	R6 [4:0]	G6 [4:0]	B6 [4:0]	A6 [0]	R5 [4:0]	G5 [4:0]	B5 [4:0]	A5 [0]	R4 [4:0]	G4 [4:0]	B4 [4:0]	A4 [0]
BGR_565	B7 [4:0]	G7 [5:0]	R7 [4:0]	B6 [4:0]	G6 [5:0]	R6 [4:0]	B5 [4:0]	G5 [5:0]	R5 [4:0]	B4 [4:0]	G4 [5:0]	R4 [4:0]	B3 [4:0]	G3 [5:0]	R3 [4:0]	B2 [4:0]
RGB_565	R7 [4:0]	G7 [5:0]	B7 [4:0]	R6 [4:0]	G6 [5:0]	B6 [4:0]	R5 [4:0]	G5 [5:0]	B5 [4:0]	R4 [4:0]	G4 [5:0]	B4 [4:0]	R3 [4:0]	G3 [5:0]	B3 [4:0]	R2 [4:0]
XYUV_444_P1_8	X[7:0]	Y03[7:0]	U03[7:0]	V03[7:0]	X[7:0]	Y02[7:0]	U02[7:0]	V02[7:0]	X[7:0]	Y01[7:0]	U01[7:0]	V01[7:0]	X[7:0]	Y00[7:0]	U00[7:0]	V00[7:0]
XYUV_444_P1_10	X [1:0]	Y03[9:0]	U03[9:0]	V03[9:0]	X [1:0]	Y02[9:0]	U02[9:0]	V02[9:0]	X [1:0]	Y01[9:0]	U01[9:0]	V01[9:0]	X [1:0]	Y00[9:0]	U00[9:0]	V00[9:0]
YVYU_422_P1_8	Y07[7:0]	V06[7:0]	Y06[7:0]	U06[7:0]	Y05[7:0]	V04[7:0]	Y04[7:0]	U04[7:0]	Y03[7:0]	V02[7:0]	Y02[7:0]	U02[7:0]	Y01[7:0]	V00[7:0]	Y00[7:0]	U00[7:0]
VYUY_422_P1_8	V06[7:0]	Y07[7:0]	U06[7:0]	Y06[7:0]	V04[7:0]	Y05[7:0]	U04[7:0]	Y04[7:0]	V02[7:0]	Y03[7:0]	U02[7:0]	Y02[7:0]	V00[7:0]	Y01[7:0]	U00[7:0]	Y00[7:0]
YUV_420_P2_8 Y PLANE	Y15[7:0]	Y14[7:0]	Y13[7:0]	Y12[7:0]	Y11[7:0]	Y10[7:0]	Y09[7:0]	Y08[7:0]	Y07[7:0]	Y06[7:0]	Y05[7:0]	Y04[7:0]	Y03[7:0]	Y02[7:0]	Y01[7:0]	Y00[7:0]
YUV_420_P2_8 UV PLANE	V14[7:0]	U14[7:0]	V12[7:0]	U12[7:0]	V10[7:0]	U10[7:0]	V08[7:0]	U08[7:0]	V06[7:0]	U06[7:0]	V04[7:0]	U04[7:0]	V02[7:0]	U02[7:0]	V00[7:0]	U00[7:0]
YUV_420_P3_8 Y PLANE	Y15[7:0]	Y14[7:0]	Y13[7:0]	Y12[7:0]	Y11[7:0]	Y10[7:0]	Y09[7:0]	Y08[7:0]	Y07[7:0]	Y06[7:0]	Y05[7:0]	Y04[7:0]	Y03[7:0]	Y02[7:0]	Y01[7:0]	Y00[7:0]
YUV_420_P3_8 U PLANE	U30[7:0]	U28[7:0]	U26[7:0]	U24[7:0]	U22[7:0]	U20[7:0]	U18[7:0]	U16[7:0]	U14[7:0]	U12[7:0]	U10[7:0]	U8[7:0]	U6[7:0]	U4[7:0]	U2[7:0]	U0[7:0]
YUV_420_P3_8 V PLANE	V30[7:0]	V28[7:0]	V26[7:0]	V24[7:0]	V22[7:0]	V20[7:0]	V18[7:0]	V16[7:0]	V14[7:0]	V12[7:0]	V10[7:0]	V8[7:0]	V6[7:0]	V4[7:0]	V2[7:0]	V0[7:0]

- Support for the following **output formats**:
  - RGB888, RGB565, RGB666

12.1.3 Block Diagram

The micro-architecture of the display subsystem is depicted below.



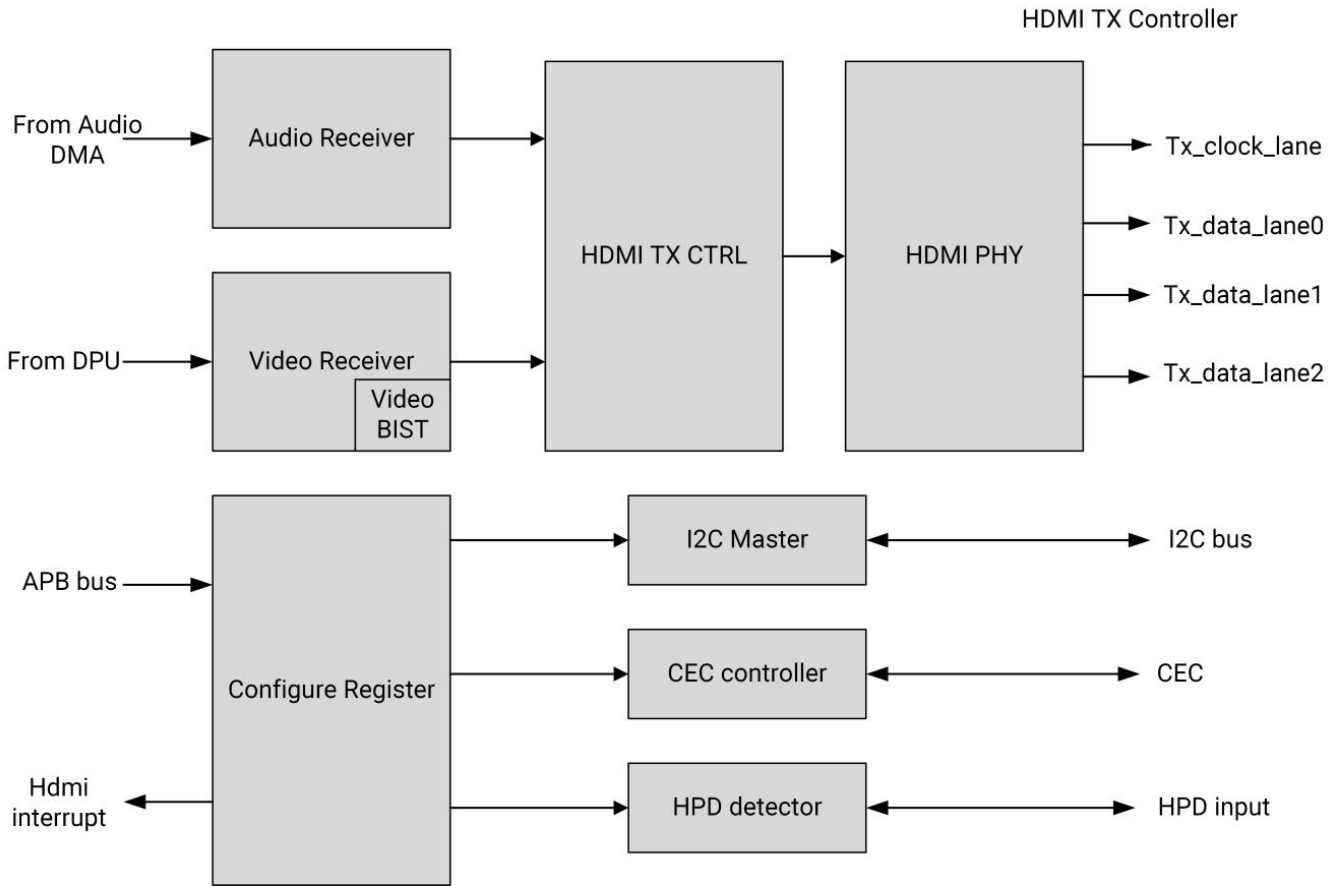
12.2 HDMI Interface

12.2.1 Features

- Compliance with HDMI Specification v1.4
- Dual-channel audio stream within the range 32~192KHz
- Physical lane speed up to 2.4Gbps/lane × 3lane
- Support for up to 1920x1440@60Hz
- Support for RGB and YcbCr 4:2:2 / 4:4:4 input video format
- Support for RGB and YcbCr 4:2:2 / 4:4:4 output video formats
- Support for 8bpc / 10bpc / 12bpc input and output color depths
- Support for EIA/CEA-861-F video timing and InfoFrame structure
- Support for L-PCM(IEC 60958), 32~192KHz dual channel audio data
- Support for Consumer Electronic Control (CEC) standard packets and user-defined packets
- Inclusion of an Internal I2C Master for remote ED access, supporting 100~400Kbps speed

12.2.2 Block Diagram

The architecture of the HDMI interface is depicted below.



12.3 MIPI-DSI

12.3.1 Introduction

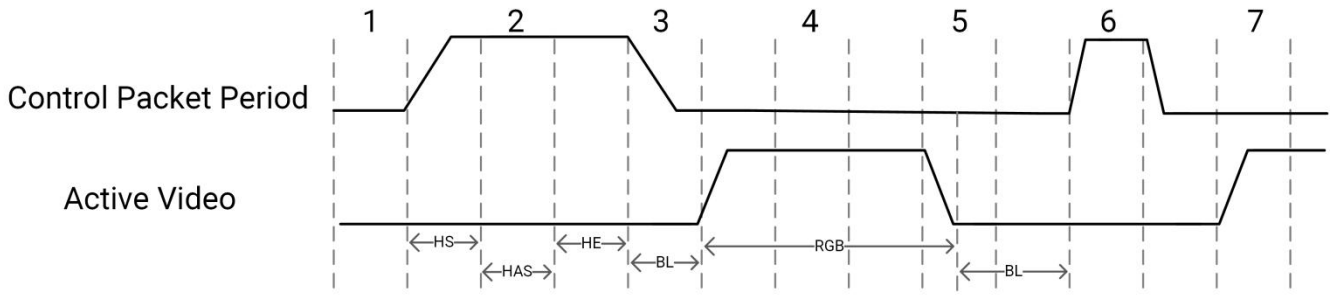
The MIPI Display Serial Interface (MIPI DSI) is a high-speed interface between a host processor and peripheral devices that adheres to MIPI Alliance specifications for mobile device interfaces.

12.3.2 Features

- Compliance with the MIPI DSI standard v1.0
- Compliance with the MIPI DPHY specification v1.1
- Support for MIPI DPHY up to 4 data lanes and speed up to 1200Mbps per lane
- Support for 1 active panel per DPHY link
- Compliance with the Display Command Set (DCS) standard
- Support for all pixel formats defined in DSI and DCS
- Support for video burst mode with DPHY up to 1.2Ghz per lane
- Support for virtual channels in the MIPI Link
- Support for up to 1080p resolution
- Support for command, video and burst modes
- Support for HS-TX, LP-TX, LP-RX and LP-CD signaling

12.3.3 Functional Description

The DSI packet arbitration is depicted below.



As can be seen, the DSI packet arbitration is based on the timing period of the DSI internal timing control, which divides the H line time into 4 periods, in particular:

- **Control packet period:** HSS, HSA, and HSE
- **Blanking period 1**: BL or smart panel packet + LP
- **Active video**: RGB
- **Blanking period 2**: BL or smart panel packet + LP

The BL packet word count is programmable and is defined as bytes in byte clock. If smart panel packet is inserted, DSI hardware switches to LP after the smart panel packet is sent.

The DSI Timing Relationship with LCD Timing is as follows:

- The DSI controller generates its timing based in LCD vsync
- The DSI syncs up every frame with the LCD controller
- Both DSI and LCD share the same H total value and run at the same pixel clock rate. As a result, the horizontal line time is the same for both the DSI and LCD controllers.
- Additionally, the shift between LCD controller line to DSI line is programmable

12.3.4 Register Description

Note. Base address = **0xD421_A800**

12.3.4.1 DSI_CTRL_0 REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CFG_SOFT_RST	RW	0x0	Software Reset DSI Module 1: Reset DSI module 0: De-assert software reset
30	CFG_SOFT_RST_REG	RW	0x0	Software Reset Configuration Registers 1: Reset DSI configuration registers to default values 0: De-assert reset
29	CFG_CLR_PHY_FIFO	RW	0x0	Configure Clear PHY Tx FIFO 1: Clear FIFO data to 0 0: De-assert clear It is NOT used currently and reserved for future use.
28	CFG_RST_TXLP	RW	0x0	Software Reset LP TX submodule 1: Reset LP TX module 0: De-assert software reset
27	CFG_RST_CPU	RW	0x0	Software Reset CPU TX submodule 1: Reset CPU TX module 0: De-assert software reset
26	CFG_RST_CPN	RW	0x0	Software Reset CPN TX submodule 1: Reset CPN TX module 0: De-assert software reset
25	RSVD	RO	0	Reserved for future use
24	CFG_RST_VPN	RW	0x0	Software Reset Video Panel submodule 1: Reset VPN module 0: De-assert software reset
23	CFG_DSI_PHY_RST	RW	0x0	Software Reset DPHY submodule 1: Reset DPHY

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: De-assert software reset
22:18	RSVD	RO	0	Reserved for future use
17	CFG_DSI_HCLK_DIS	RW	0x0	DSI AHB Clock Disable **Note.** DSI configuration registers can still be written or read even if the DSI AHB clock is disabled. 1: DSI AHB clock will be gated 0: DSI AHB clock is passed to DSI module
16	CFG_DSI_CLK_DIS	RW	0x0	DSI Clock disable 1: DSI clock will be gated 0: DSI clock is passed to DSI module
15:9	RSVD	RO	0	Reserved for future use
8	CFG_VPN_TX_EN	RW	0x1	Video Panel Interface TX Enable 1: Enable Video Panel TX packet to DPHY. DSI will send video packets to peripherals. 0: Disable Video Panel interface TX.
7:5	RSVD	RO	0	Reserved for future use
4	CFG_VPN_SLV	RW	0x1	Video Panel Interface in slave mode 1: Video Panel works in slave mode. It receives VSYNC from input LCD interface, and is used to control the internal timing. 0: Video Panel interface works in master mode. DSI sends VSYNC to LCD module, and controls the V timing and H timing. **Note.** This bit must set to 1, VPN only supports slave mode.
3	RSVD	RO	0	Reserved for future use
2	CFG_CPN_EN	RW	0x0	Command Panel Interface Enable 1: Command panel is running and can accept data from the Command Panel interface 0: Disable Command Panel interface
1	RSVD	RO	0	Reserved for future use
0	CFG_VPN_EN	RW	0x0	Video Panel Interface Enable 1: Video Panel is active and running. 0: Video Panel interface is disabled. **Note.** Set this field to 1 to start the Video Panel timing.

12.3.4.2 DSI_CTRL_1 REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	RSVD	RO	0	Reserved for future use
8	CFG_EOTP_EN	RW	0x0	EOTP Enable 1: Enable EOTP packet 0: Disable EOTP packet
7:6	CFG_CPN_VCH_NO	RW	0x3	Command Panel Virtual Channel Number
5:2	RSVD	RO	0	Reserved for future use
1:0	CFG_VPN_VCH_NO	RW	0x0	Video Panel Virtual Channel Number for Active Panel 1 This parameter defines the virtual channel number for VPN

12.3.4.3 DSI_IRQ_ST1 REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	RO	0	Reserved for future use
3	IRQ_DPHY_ERR_HS_RXP	RO	0x0	DPHY HSTX contention RXP Error
2	IRQ_DPHY_ERR_HS_RXN	RO	0x0	DPHY HSTX contention RXN Error
1	IRQ_DPHY_ERR_HS_CONTP	RO	0x0	DPHY HSTX contention contp Error
0	IRQ_DPHY_ERR_HS_CONTN	RO	0x0	DPHY HSTX contention contn Error

12.3.4.4 DSI_IRQ_MASK1 REGISTER

Offset: 0xc				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	RO	0	Reserved for future use
3:0	CFG_IRQ_MASK1	RW	0x0	DSI interrupt mask This field is used to mask interrupt requests. If one bit is set to 0x1, the corresponding interrupt status is masked.

12.3.4.5 DSI_IRQ_ST REGISTER

Offset: 0x10

Bits	Field	Type	Reset	Description
31	IRQ_LAST_LINE	RO	0x0	Last Line interrupt
30	IRQ_CPN_TE	RO	0x0	Command Panel Tearing Effect.
29	IRQ_TA_TIMEOUT	RO	0x0	Turnaround Acknowledgement Timeout for DPHY
28	IRQ_RX_TIMEOUT	RO	0x0	LP-RX Timeout for DPHY
27	IRQ_TX_TIMEOUT	RO	0x0	HS TX Timeout for DPHY
26	IRQ_RX_STATE_ERR	RO	0x0	Peripheral Status Error After DSI receives an acknowledgement with error report packet from slave, it will mark this bit if an error status is reported.
25	IRQ_RX_ERR	RO	0x0	DSI RX Packet Error DSI receives a packet (with error status, such as ecc error/crc error/unknown packet) from slave
24	IRQ_RX_FIFO_FULL_ERR	RO	0x0	RX FIFO Full Error
23	IRQ_PHY_FIFO_UNDERRUN	RO	0x0	PHY FIFO Underrun Error
22	IRQ_REQ_CNT_ERR	RO	0x0	TX Request Count Error This error occurs when the delays between an Active Panel TX request and the DPHY ready signal are inconsistent.
21	IRQ_RXPSR_FIFO_FULL_ERR	RO	0x0	RX Parser FIFO Full Error
20	IRQ_VPN_REQ_PHY_DLY_ERR	RO	0x0	VPN Request Delay Error at PHY Interface VPN packets are delayed at the PHY interface.
19	IRQ_VPN_BF_UNDERRUN_ERR	RO	0x0	VPN Buffer Underrun Error
18	IRQ_VPN_REQ_ARB_DLY_ERR	RO	0x0	VPN Request Delay Error at Arbiter Interface VPN packets are delayed at arbiter point.
17	IRQ_VPN_BF_OVERRUN_ERR	RO	0x0	VPN Buffer Overrun Error
16	IRQ_VPN_TIMING_ERR	RO	0x0	VPN Data Timing Error This error indicates that pixel data might be incorrect. It occurs when the Data FIFO for the VPN path is read too early or too late, leading to an empty FIFO when it is accessed.

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
15	IRQ_VPN_VACT_DONE	RO	0x0	VPN VACT Done
14	IRQ_VPN_BF_FULL	RO	0x0	VPN Buffer Full Error Pixel data may be incorrect.
13	IRQ_CPN_BF_FULL	RO	0x0	CPN Buffer Full Error Pixel data may be incorrect.
12	IRQ_DPHY_ERR_CONT_LP1	RO	0x0	DPHY LP1 Contention Detect PPI ErrContertionLP1
11	IRQ_DPHY_ERR_CONT_LP0	RO	0x0	DPHY LP0 Contention Detect PPI ErrContertionLP0
10	IRQ_DPHY_ERR_SYNC_ESC	RO	0x0	DPHY Sync Error PPI ErrSyncEsc Partial byte is detected
9	IRQ_DPHY_ERR_ESC	RO	0x0	DPHY Invalid Command Detect PPI ErrEsc Invalid ESC command is detected
8	IRQ_DPHY_RX_LINE_ERR	RO	0x0	DPHY Invalid Line State Detect PPI ErrControl
7	IRQ_RX_TRG3	RO	0x0	DPHY RX Trigger 3 Received By default, the value of trigger 3 is 0x05, Note: Its exact meaning is not defined by specification.
6	IRQ_RX_TRG2	RO	0x0	DPHY RX Trigger 2 Received By default, trigger 2 is for acknowledgement Trigger, and its value is 0x84.
5	IRQ_RX_TRG1	RO	0x0	DPHY RX Trigger 1 Received By default, trigger 1 is for TE Trigger, and its value is 0xBA.
4	IRQ_RX_TRG0	RO	0x0	DPHY RX Trigger 0 Received By default, trigger 0 is for Reset Trigger, and its value is 0x46.
3	IRQ_RX_ULPS	RO	0x0	DPHY RX ULPS Received
2	IRQ_RX_PKT	RO	0x0	DPHY RX Packet Received
1	IRQ_CPN_TX_DONE	RO	0x0	Command Panel Data Transmission Done
0	IRQ_CPU_TX_DONE	RO	0x0	CPU Packet Transmission Done

12.3.4.6 DSI_IRQ_MASK REGISTER

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_IRQ_MASK	RW	0x0	DSI interrupt mask This field is used to mask interrupt requests. If one bit is set to 0x1, the corresponding interrupt status is masked.

12.3.4.7 DSI_CPU_CMD_0 REGISTER

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CFG_CPU_CMD_REQ	RW	0x0	CPU Command Request 1: CPU packet request 0: No request or request done After software writes a command with this bit set to 1, the DSI module sends out a packet as requested. DSI de-asserts this field after packet is sent.
30	CFG_CPU_SP	RW	0x0	CPU Short Packet 1: CPU packet is a short packet 0: CPU packet is a long packet
29	CFG_CPU_TURN	RW	0x0	CPU Turn Around 1: After CPU packet, turn around the bus 0: Don't turn around bus after CPU packet
28	RSVD	RO	0	Reserved for future use
27	CFG_CPU_TXLP	RW	0x0	Low Power TX for CPU Packets 1 = Transfer CPU packets in low power mode 0 = Use high-speed mode to send CPU packets
26:16	RSVD	RO	0	Reserved for future use
15:0	CFG_CPU_WC	RW	0x0	CPU Packet Byte Count For high-speed transfer, this represents the payload byte count for long packets (excluding CRC bytes). For high-speed short packet transfer, this field is ignored. For low power transfer, this is the byte count for the entire packet, including CRC bytes, and CFG_CPU_SP is ignored.

12.3.4.8 DSI_CPU_CMD_1 REGISTER

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	RO	0	Reserved for future use
23:20	CFG_TXLP_LPDT	RW	0x0	LPDT TX Enable LPDT TX enables signals for low power TX
19:16	CFG_TXLP_ULPS	RW	0x0	ULPS TX Enable ULPS TX enables signals for Low power TX
15:0	CFG_TXLP_TRIGGER_CODE	RW	0x0	Low Power TX Trigger Code

12.3.4.9 DSI_CPU_CMD_3 REGISTER

Offset: 0x2c				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CFG_CPU_DAT_REQ	RW	0x0	CPU Packet Data Buffer Read/Write Request 1: CPU packet data request 0: No request or request done After software writes a command with this bit set to 1, the DSI module will write data to the packet data buffer or read data from the data buffer as requested. DSI will de-assert this bit after write/read operation is done. Read data will be valid after this bit is reset to 0.
30	CFG_CPU_DAT_RW	RW	0x0	CPU Packet Data Buffer Read/Write Operation 1: CPU packet data write operation 0: CPU packet data read operation
29:24	RSVD	RO	0	Reserved for future use
23:16	CFG_CPU_DAT_ADDR	RW	0x0	CPU Packet Data Address This is the byte address for the packet data. - For each read/write operation, 4 bytes of data are written or read. Software should increment the address by 4 after each operation. - The packet data starts with the packet header: 1. At address 0: - Bits [7:0] represent Type_id - Bits [23:8] represent Length - Bits [31:24] represent ECC 2. At address 4: Payload data if it's a long packet, and so on. - The maximum packet data buffer is 256 bytes.
15:0	RSVD	RO	0	Reserved for future use

12.3.4.10 DSI_CPU_WDAT REGISTER

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_CPU_WDAT	RW	0x0	CPU Data Write Register (wdata 0) This register defines the data for CPU packets. It holds the CPU packet data that will be written to the packet data buffer. - Software should first program the packet data into this register, then configure the DSI CPU Packet Command Register 3 to load the packet data into the Tx packet data buffer. - For each read/write operation, 4 bytes of data are written or read. 1. Bits [7:0]: LSB 2. Bits [31:24]: MSB - For packet data at address 0: 1. Bits [7:0]: Type_id 2. Bits [23:8]: Length 3. Bits [31:24]: ECC - For data at address 4: The payload data if it's a long packet, and so on. - High-speed transmission: Hardware automatically generates ECC and CRC codes, replacing them in the packet data buffer. - Low-power transmission: Hardware does not insert ECC/CRC, and instead sends out the ECC/CRC from the packet data buffer

12.3.4.11 DSI_CPU_STATUS_0 REGISTER

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	RO	0	Reserved for future use
15:0	CFG_CPU_PKT_CNT	RW	0x0	CPU Packet Counter This counter counts the number of CPU packets sent out through DSI. This register is write clear.

12.3.4.12 DSI_CPU_STATUS_1 REGISTER

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_CPU_CMD_TX_CNT	RO	0x0	CPU CMD TX Counter This counter counts the number of byte clock cycles required to transfer the current CPU command. It begins to count after CPU command is received, and stops to counter after DPHY gets ready for another TX request. This counter could help to decide the values of DSI_VPN_SLOT_CNT_0 and DSI_VPN_SLOT_CNT_1 register.

12.3.4.13 DSI_CPU_STATUS_2 REGIST

Offset: 0x3c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_CPU_CMD_CNT	RW	0x0	CPU CMD Execution Counter This counter counts the number of byte clock cycles required to execute the current CPU command. This counter only counts the cycles which CPU engine is busy. This counter could help to decide the values of DSI_VPN_SLOT_CNT_0 and DSI_VPN_SLOT_CNT_1 register.

12.3.4.14 DSI_CPU_STATUS_3 REGISTER

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_TXLP_CNT	RO	0x0	Low Power TX byte clock count This counter counts the number of byte clock cycles required to transfer a low power packet.

12.3.4.15 DSI_CPU_STATUS_4 REGISTER

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_BTA_CNT	RO	0x0	Bus Turn Around byte clock count This counter counts the number of byte clock cycles required to

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
				complete a bus turn around operation.

12.3.4.16 DSI_CPN_STATUS_1 REGISTER

Offset: 0x4c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_CPN_STATUS_1	RW	0x0	Command Panel Path Status 1 It includes the following fields: - smt_bf_cnt[5:0] - smt_fifo_bcnt[9:0] - smt_cs[4:0] - smt_wr_on - smt_dma_on - smt_fifo_empty - smt_bf_empty - smt_fifo_full_r - smt_bf_full_r

12.3.4.17 DSI_CPN_CMD REGISTER

Offset: 0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	CFG_CPN_TE_EN	RW	0x0	Command Panel Tearing Effect Signal Enable
27	RSVD	RO	0	Reserved for future use
26:24	CFG_CPN_RGB_TYPE	RW	0x0	Command Panel Data RGB Type 0x0: 888 mode 0x1: 666 unpacked mode 0x2: 565 mode 0x3: 444 mode 0x4: 332 mode 0x5: 111 mode
23:4	RSVD	RO	0	Reserved for future use
3	CFG_CPN_BURST_MODE	RW	0x1	Command Panel Interface Burst Mode Enable 0: Enable Previous Command Pandel interface. 1: Burst mode interface between LCD and

Offset: 0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
				DSI will take effect. This interface provides a more efficient connection than the previous mode
2	CFG_CPN_FIRSTP_SEL	RW	0x0	Command panel First packet select 0: FIFO empty 1: vsync from DP650
1	CFG_CPN_DMA_DIS	RW	0x0	Command Panel dma_on Disable 1: Disable smt_dma_on signal from LCD controller. DSI will not receive Command Panel interface data from LCD even smt_dma_on signal is active high. 0 = Receive LCD Command Panel interface data when smt_dma_on is high.
0	CFG_CPN_ADDR0_EN	RW	0x0	Command Panel Address Bit Indicator 0 : When smt_addr = 1, bus data is for pixel RGB data. When smt_addr = 0, bus data is ignored. 1 : When smt_addr = 0, bus data is for pixel RGB data. When smt_addr = 1, bus data is ignored.

12.3.4.18 DSI_CPN_CTRL_0 REGISTER

Offset: 0x54				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	RO	0	Reserved for future use
21:16	CFG_DCS_LONGWR_CODE	RW	0x39	DSI Command Code for Writing Command Panel Data The default data is 0x39 from DSI specification.
15:8	CFG_DCS_WR_CON_CODE	RW	0x3C	DCS Command for Continuous Write The default value is 0x3C in MIPI Alliance Standard for Display Command Set Specification.
7:0	CFG_DCS_WR_STR_CODE	RW	0x2C	DCS Command for First Write The default value is 0x2C in the MIPI Alliance Standard for Display Command Set Specification.

12.3.4.19 DSI_CPN_CTRL_1 REGISTER

Offset: 0x58				
Bits	**Field**	**Type**	**Reset**	**Description**
31:26	RSVD	RO	0	Reserved for future use
25:16	CFG_CPN_PKT_CNT	RW	0x100	Command Panel Packet Length This field defines the packet length for Command Panel packets.
15:10	RSVD	RO	0	Reserved for future use
9:0	CFG_CPN_FIFO_FULL_LEVEL	RW	0x200	Command Panel FIFO Full Level, in byte count

12.3.4.20 DSI_CPN_STATUS_0 REGISTER

Offset: 0x5c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_CPN_FRM_CNT	RW	0x0	Command Panel Frame Counter This counter counts the numbers of Command Panel frames sent through DSI. This register is write clear.

12.3.4.21 DSI_RX_PKT_ST_0 REGISTER

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RX_PKT0_ST_VLD	RWC	0x0	Rx Packet 0 Status Valid 1: Valid status 0: Invalid status
30:27	RSVD	RO	0	Reserved for future use
26	RX_PKT0_ST_EOTP	RWC	0x0	Rx Packet 0 is EOTP 1: Received packet is EOTP packet 0: Other packet. It is valid only when RX_PKT0_ST_VLD = 1.
25	RX_PKT0_ST_ACK	RWC	0x0	Rx Packet 0 is ACK Packet 1: Received packet is an ACK packet, with or without errors. 0 :Other packet. It is valid only when RX_PKT0_ST_VLD = 1.

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
24	RX_PKT0_ST_SP	RWC	0x0	Rx Packet 0 Short Packet 1: Received packet is a short packet. 0: Long packet, it is valid only when RX_PKT0_ST_VLD = 1.
23:22	RSVD	RO	0	Reserved for future use
21:16	RX_PKT0_PKT_PTR	RWC	0x0	Rx Packet 0 Data Pointer Packet header in FIFO is the raw data from DPHY and is before ECC correction. It is valid only when RX_PKT0_ST_VLD = 1.
15:14	RX_PKT0_VCH	RWC	0x0	Rx Packet 0 Virtual Channel Number It is valid only when RX_PKT0_ST_VLD = 1.
13:12	RSVD	RO	0	Reserved for future use
11:8	RX_PKT0_ECC_FLAGS	RWC	0x0	Rx Packet 0 ECC Error Flags Bit [11]: 1: No ECC error 0: ECC error Bit [10]: 1: Correctable error in data bits. Bit [9]: 1: Correctable error happens at parity bits. Bit [8]: 1: Incorrectable error It is valid only when RX_PKT0_ST_VLD = 1.
7:5	RSVD	RO	0	Reserved for future use
4	RX_PKT0_NO_CRC	RWC	0x0	Rx Packet 0 Without CRC Rx packet does not include CRC, and CRC part contains 0x0000. It is valid only when RX_PKT0_ST_VLD = 1.
3	RX_PKT0_UNKNOWN_ERR	RWC	0x0	Rx Packet 0 Type Unknown Error It is valid only when RX_PKT0_ST_VLD = 1.
2	RX_PKT0_ST_ERR	RWC	0x0	Rx Packet 0 ack Status Error Indicates an error in the acknowledge packet status. The DSI_RX_PKT_HDR_0 should be checked to identify the specific error. It is valid only when RX_PKT0_ST_VLD = 1.
1	RX_PKT0_ECC_ERR	RWC	0x0	Rx Packet 0 ECC Error It is valid only when RX_PKT0_ST_VLD = 1.
0	RX_PKT0_CRC_ERR	RWC	0x0	Rx Packet CRC Error It is valid only when RX_PKT0_ST_VLD = 1.

12.3.4.22 DSI_RX_PKT_HDR_0 REGISTER

Offset: 0x64				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_PKT0_HDR	RW	0x0	Rx Packet 0 Header Bits [7:0]: DataID Bits [23:8]: Length Bits [31:23]: ECC--Corrected if an error is detected

12.3.4.23 DSI_RX_PKT_ST_1 REGISTER

Offset: 0x68				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RX_PKT1_ST_VLD	RWC	0x0	Rx Packet 1 Status Valid 1: Valid status 0: Invalid status
30:27	RSVD	RO	0	Reserved for future use
26	RX_PKT1_ST_EOTP	RWC	0x0	Rx Packet 1 is EOTP 1: Received packet is EOTP packet. 0: Other packet. It is valid only when RX_PKT0_ST_VLD = 1.
25	RX_PKT1_ST_ACK	RWC	0x0	Rx Packet 1 is ACK Packet 1 = Received packet is an ACK packet, with or without error. 0 = Other packet. It is valid only when RX_PKT0_ST_VLD = 1.
24	RX_PKT1_ST_SP	RWC	0x0	Rx Packet 1 Short Packet 1: Received packet is a short packet. 0: Long packet valid only when RX_PKT0_ST_VLD = 1.
23:22	RSVD	RO	0	Reserved for future use
21:16	RX_PKT1_PKT_PTR	RWC	0x0	Rx Packet 1 Data Pointer Packet header in FIFO is the raw data from DPHY before ECC correction. Valid only when RX_PKT0_ST_VLD = 1.
15:14	RX_PKT1_VCH	RWC	0x0	Rx Packet 1 Virtual Channel Number Valid only when RX_PKT0_ST_VLD = 1.
13:12	RSVD	RO	0	Reserved for future use
11:8	RX_PKT1_ECC_FLAGS	RWC	0x0	Rx Packet 1 ECC Error Flags Bit [11]: 1: No ECC error

Offset: 0x68				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: ECC error Bit [10]: 1: Correctable error in data bits Bit [9]: 1: Correctable error happens at parity bits Bit [8]: 1: Incorrectable error. It is valid only when RX_PKT0_ST_VLD = 1.
7:5	RSVD	RO	0	Reserved for future use
4	RX_PKT1_NO_CRC	RWC	0x0	Rx Packet 1 Without CRC Rx packet does not include CRC, and CRC part contains 0x0000. It is valid only when RX_PKT0_ST_VLD = 1.
3	RX_PKT1_UNKNOWN_ERR	RWC	0x0	Rx Packet Type Unknown Error It is valid only when RX_PKT0_ST_VLD = 1.
2	RX_PKT1_ST_ERR	RWC	0x0	Rx Packet 1 ack Status Error DSI_RX_PKT_HDR_0 should be checked to identify the specific error. It is valid only when RX_PKT0_ST_VLD = 1.
1	RX_PKT1_ECC_ERR	RWC	0x0	Rx Packet 1 ECC Error It is valid only when RX_PKT0_ST_VLD = 1.
0	RX_PKT1_CRC_ERR	RWC	0x0	Rx Packet 1 CRC Error It is valid only when RX_PKT0_ST_VLD = 1.

12.3.4.24 DSI_RX_PKT_HDR_1 REGISTER

Offset: 0x6c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_PKT1_HDR	RW	0x0	Rx Packet 1 Header Bits [7:0]: DataID Bits [23:8]: Length Bits [31:23]: ECC--Corrected if an error is detected

12.3.4.25 DSI_RX_PKT_CTRL REGISTER

Offset: 0x70				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RX_PKT_RD_REQ	RW	0x0	Rx Packet FIFO Read Request 1 = Read request 0 = Invalid request This bit will be cleared to 0 after read operation is done, and Rx data is valid.
30:22	RSVD	RO	0	Reserved for future use
21:16	RX_PKT_RD_PTR	RW	0x0	Rx Packet Data FIFO Read Pointer For every read operation, the hardware returns data from the current pointer address. Software must increment this pointer for the next data after each byte is read.
15:8	RSVD	RO	0	Reserved for future use
7:0	RX_PKT_RD_DATA	RW	0x0	Rx FIFO Read Data Valid when RX_PKT_RD_REQ = 0. - First byte: DataID - Second byte: wc0 - Third byte: wc1 - Fourth byte: raw ECC received from DPHY, not corrected - Fifth byte and beyond: long packet data

12.3.4.26 DSI_RX_PKT_CTRL_1 REGISTER

Offset: 0x74				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	RO	0	Reserved for future use
11:8	RX_PKT_CNT	RWC	0x0	RX Packet Count in Rx FIFO All LP RX packets are stored in the FIFO and start from address 0.
7:0	RX_PKT_BCNT	RWC	0x0	RX Byte Count in Rx FIFO The whole LP RX data is stored in the FIFO and starts from address 0.

12.3.4.27 DSI_RX_PKT_ST_2 REGISTER

Offset: 0x78				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RX_PKT2_ST_VLD	RWC	0x0	Rx Packet 2 Status Valid

Offset: 0x78				
Bits	**Field**	**Type**	**Reset**	**Description**
				1: Valid status 0: Invalid status
30:27	RSVD	RO	0	Reserved for future use
26	RX_PKT2_ST_EOTP	RWC	0x0	Rx Packet 2 is EOTP 1: Received packet is EOTP packet 0: Other packet. It is valid only when RX_PKT0_ST_VLD = 1.
25	RX_PKT2_ST_ACK	RWC	0x0	Rx Packet 2 is an ACK Packet 1: Received packet is an ACK packet, with or without error 0: Other packet. It is valid only when RX_PKT0_ST_VLD = 1.
24	RX_PKT2_ST_SP	RWC	0x0	Rx Packet 2 Short Packet 1: Received packet is a short packet 0: Long packet Valid only when RX_PKT0_ST_VLD = 1
23:22	RSVD	RO	0	Reserved for future use
21:16	RX_PKT2_PKT_PTR	RWC	0x0	Rx Packet 2 Data Pointer The packet header in FIFO is the raw data from DPHY before ECC correction. Valid only when RX_PKT0_ST_VLD = 1.
15:14	RX_PKT2_VCH	RWC	0x0	Rx Packet 2 Virtual Channel Number Valid only when RX_PKT0_ST_VLD = 1
13:12	RSVD	RO	0	Reserved for future use
11:8	RX_PKT2_ECC_FLAGS	RWC	0x0	Rx Packet 2 ECC Error Flags bit [11]: 1: No ECC error 0: ECC error Bit [10]: 1: Correctable error in data bits Bit [9]: 1: Correctable error happens at parity bits Bit [8]: 1: Incorrectable error. It is valid only when RX_PKT0_ST_VLD = 1.
7:5	RSVD	RO	0	Reserved for future use
4	RX_PKT2_NO_CRC	RWC	0x0	Rx Packet 2 Without CRC Rx packet does not include CRC and CRC part contains 0x0000. It is valid only when RX_PKT0_ST_VLD = 1.
3	RX_PKT2_UNKNO	RWC	0x0	Rx Packet 2 Type Unknown Error

Offset: 0x78				
Bits	**Field**	**Type**	**Reset**	**Description**
	WN_ERR			It is valid only when RX_PKT0_ST_VLD = 1.
2	RX_PKT2_ST_ERR	RWC	0x0	Rx Packet 2 ack Status Error DSI_RX_PKT_HDR_0 should be checked to identify the specific error. It is valid only when RX_PKT0_ST_VLD = 1.
1	RX_PKT2_ECC_ERR	RWC	0x0	Rx Packet 2 ECC Error It is valid only when RX_PKT0_ST_VLD = 1.
0	RX_PKT2_CRC_ERR	RWC	0x0	Rx Packet 2 CRC Error It is valid only when RX_PKT0_ST_VLD = 1.

12.3.4.28 DSI_RX_PKT_HDR_2REGISTER

Offset: 0x7c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_PKT2_HDR	RW	0x0	Rx Packet 2 Header

12.3.4.29 DSI_LCD_BDG_CTRL0 REGISTER

Offset: 0x84				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	RSVD	RO	0	Reserved for future use
27:16	CFG_VPN_FIFO_AFULL_CNT	RW	0x0	DSI VPN FIFO Almost Full Count The difference between the FIFO read pointer and write pointer must be greater than this value.
15:10	RSVD	RO	0	Reserved for future use
9	CFG_HSYNC_MISSING_FIX	RW	0x0	Fix for the Hsync missing bug
8	CFG_TXLP_LANE_TURN_FIX	RW	0x0	Fix for the TXLP lane turn bug
7	RSVD	RO	0	Reserved for future use
6	CFG_VPN_FIFO_AFULL_BYPASS	RW	0x0	Bypass VPN FIFO almost full 0: Not bypass 1: Bypass (LCD outputs pixel data to VPN FIFO regardless of the almost full signal)

Offset: 0x84				
Bits	**Field**	**Type**	**Reset**	**Description**
5	CFG_CPN_VSYNC_EDGE	RW	0x0	CPN Vsync signal edge select 0: posedge (Positive edge) select 1: negedge (Negative edge) select
4	CFG_CPN_TE_EDGE	RW	0x0	CPN tearing effect signal edge select 0: posedge (Positive edge) select 1: negedge (Negative edge) select
3:2	CFG_CPN_TE_MODE	RW	0x0	CPN tearing effect mode select 0: No Tearing Effect 1: Mode A – Tearing effect signal consists of V-Blanking only 2: Mode B – Tearing effect signal consists of both V-Blanking and H-Blanking 3: Mode C – Tearing effect signal outputs the N H-Blanking
1	CFG_PIXEL_SWAP	RW	0x0	LCD output pixel swap 0: Do not swap LCD output pixel data 1: Swap LCD output pixel data
0	CFG_SPLIT_EN	RW	0x0	Split Mode enable This bit should be set as same as LCD split mode 0: Split mode disable, only DSIA is used for display 1: Split mode enable, both DSIA and DSIB are used for display

12.3.4.30 DSI_LCD_BDG_CTRL1 REGISTER

Offset: 0x88				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_CPN_TE_DLY_CNT	RW	0x10	CPN Tearing Effect Delay Count The LCD output pixel data will be delayed by the number of cycles specified in this field after the TE pulse.
15:0	CFG_CPN_TE_LINE_CNT	RW	0x0	CPN Tearing Effect line Count When TE_MODE = 2, this field takes effect. The LCD output pixel data will be delayed for a number of TE pulses specified by this field.

12.3.4.31 DSI_TX_TIMER REGISTER

Offset: 0xe4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_TX_TIMER_CNT	RW	0xffffffff	Tx Transmission Timer Value This timer monitors the Tx operation on the DSI output side. It could generate IRQ after timer timeout. By default setting, timeout will not occur because the reset value is set to the maximum value (0xffffffff).

12.3.4.32 DSI_RX_TIMER REGISTER

Offset: 0xe8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_RX_TIMER_CNT	RW	0xffffffff	Rx Timer Value This timer monitors the Rx operation on the DSI operation. It could generate IRQ after timer timeout. By default setting, timeout will not occur because the reset value is set to the maximum value (0xffffffff).

12.3.4.33 DSI_TURN_TIMER REGISTER

Offset: 0xec				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_TURN_TIMER_CNT	RW	0xffffffff	Bus Turn Around Timer Value This timer monitors the turn around operation on the DSI. It could generate IRQ after timer timeout. By default setting, timeout will not occur because the reset value is set to the maximum value (0xffffffff).

12.3.4.34 DSI_VPN_CTRL_0 REGISTER

Offset: 0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_DLY_CNT	RW	0x100	VPN Vsync Delay Count in slave mode. In slave mode, the DSI begins H/V timing based on the input Vsync from the LCD module. After receiving the Vsync from the LCD controller, the DSI will start the Vsync timing by delaying it for the number of clock

Offset: 0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
				cycles specified by this field.
15:8	RSVD	RO	0	Reserved for future use
7:0	CFG_VPN_TX_DLY_CNT	RW	0x10	VPN TX Delay Count After the DSI starts Hsync timing, this field defines the delay in DPHY byte clock cycles before initiating a VSS packet transfer. This internal delay ensures a fixed TX timing at the DPHY interface.

12.3.4.35 DSI_VPN_CTRL_1 REGISTER

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CFG_VPN_VSYNC_RST_EN	RW	0x0	LCD Vsync Reset Enable in slave mode 1: Reset DSI vertical state machine when LCD Vsync comes. This will only take effect when LCD is in slave mode. 0: Do not reset the DSI vertical state machine
30:28	RSVD	RO	0	Reserved for future use
27	CFG_VPN_AUTO_WC_DIS	RW	0x0	VPN Auto Word Count Disable This bit has lower priority than CFG_VPN_HACT_WC_EN 0x0: Enable auto word count calculation, and hardware automatically calculates the number of bytes that will be sent in each H line slot 0x1: Auto word count calculation will not be effective
26	CFG_VPN_HACT_WC_EN	RW	0x0	VPN Hact Word Count Enable This bit has higher priority than CFG_VPN_AUTO_WC_EN 0x0: CFG_HACT_WC will not be effective if CFG_VPN_AUTO_WC_DIS is set to 0 0x1: Enable Hact word count parameter, and CFG_HACT_WC will be used to decide the number of bytes that are sent
25	CFG_VPN_TIMING_CHECK_DIS	RW	0x0	VPN Hss/Hse/Hact TX Timing Check Disable 0x0: Check timing before requesting DPHY for TX 0x1: No check timing before requesting DPHY for TX
24	CFG_VPN_AUTO_DLY_DIS	RW	0x0	VPN Auto Vsync Delay Count Disable 0x0: Enable automatic Vsync delay count calculation, and hardware will automatically use half of CFG_HTOTAL_CNT to replace CFG_VPN_DLY_CNT for Vsync delay. 0x1: Disables the auto Vsync delay count. The hardware will use the value in CFG_VPN_DLY_CNT for Vsync delay.

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
23	RSVD	RO	0	Reserved for future use
22	CFG_VPN_HLP_PKT_EN	RW	0x0	Long Blanking Packet Enable 1: DSI sends out a long blanking packet during the HLP time slot. 0: Long blanking packet is disabled. DSI will enter low power during this time slot (In most cases, this field should be programmed to 0x0)
21	CFG_VPN_HEX_PKT_EN	RW	0x0	Extra Long Blanking Packet Enable 1: DSI sends out a long blanking packet after pixel data transmission and before the hfp. 0: Extra long blanking packet is disabled. DSI enter low power during this time slot (In most cases, this field should be programmed to 0x0)
20	CFG_VPN_HFP_PKT_EN	RW	0x0	Front Porch Packet Enable 1: DSI sends out a long blanking packet during the hfp time slot 0: hfp long blanking packet is disabled DSI will go to low power during this time slot If front porch period is not long enough for DPHY to go to low power state and come back to HS again timely for next Hss packet, this field should be programmed to 0x1.
19	RSVD	RO	0	Reserved for future use
18	CFG_VPN_HBP_PKT_EN	RW	0x0	Back Porch Packet Enable 1: DSI sends out a long blanking packet during the hbp time slot 0: hbp long blanking packet is disabled. DSI will enter low power during this time slot If the back porch period is not long enough for DPHY to go to low power state and return to HS mode in time for next pixel data packet, this field should be programmed to 0x1.
17	CFG_VPN_HSE_PKT_EN	RW	0x0	Hse Packet Enable 1: DSI will send out hse packet during hbp time slot 0: hse packet is disabled DSI will go to low power during this time slot > **Note.** Enable this bit when transmission mode is in Non-burst mode with sync pulse.
16	CFG_VPN_HSA_PKT_EN	RW	0x0	Hsa Packet Enable 1: DSI sends out hsa long blanking packet during the hbp time slot 0: hsa packet is disabled. DSI will go to low power during this time slot - If transmission mode is in non-burst mode (with sync event) or burst mode, this field should be disabled. - If transmission mode is non-burst mode (with sync pulse), this field can be programmed to 0x1.
15	RSVD	RO	0	Reserved for future use
14	CFG_VPN_HEX_SLOT_EN	RW	0x0	Extra Long Packet Enable after Pixel Data 1: Enable extra long packet after pixel data transfer, this will insert a long blanking packet before hfp 0: No extra long packet is inserted after pixel data transfer

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field takes effect only in burst mode. In most cases, this field should be programmed to 0x0.
13:11	RSVD	RO	0	Reserved for future use
10	CFG_VPN_LAST_LINE_TURN	RW	0x0	Turn Around Bus at Last h Line 1: DSI will turn around the bus at the last horizontal line of each frame. This will request the slave to return an acknowledgment or an acknowledgment with an error. 0: DSI will not turn around the bus during the last horizontal line of the frame. In most cases, this field should be set to 0x0.
9	CFG_VPN_LPM_FRAME_EN	RW	0x0	Go to Low Power Every Frame 1: DSI will go to low power mode during the last horizontal line of every frame 0: DSI will not go to low power mode during the last h line In most cases, this field should be programmed to 0x0.
8:4	RSVD	RO	0	Reserved for future use
3:2	CFG_VPN_BURST_MODE	RW	0x0	DSI Transmission Mode for LCD 1 0x0: Non-burst mode with sync pulse 0x1: Non-burst mode with sync event 0x2: Burst mode
1:0	CFG_VPN_RGB_TYPE	RW	0x0	LCD 1 Input Data RGB Mode for LCD 1 0x0: 565 RGB mode 0x1: 666 packet mode 0x2: 666 un-packet mode 0x3: 888 RGB mode

12.3.4.36 DSI_VPN_TIMING_0 REGISTER

Offset: 0x110				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_HACT_CNT	RW	0x0	VPN hact Clock Count in byte clock domain This parameter defines the byte clock cycle numbers for horizontal line pixel data period The data byte number for this period is HACT_BYTE_CNT= HACT_CNT*lane_num
15:0	CFG_VPN_HTOTAL_CNT	RW	0x0	VPN htotal Clock Count in byte clock domain. This parameter defines the byte clock cycle numbers for horizontal line period The data byte number for this period is

Offset: 0x110				
Bits	**Field**	**Type**	**Reset**	**Description**
				HTOTAL_BYTE_CNT = HTOTAL_CNT*lane_num

12.3.4.37 DSI_VPN_TIMING_1 REGISTER

Offset: 0x114				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_HSYNC_CNT	RW	0x0	VPN hsync Clock Count in byte clock domain. This parameter defines the byte clock cycle numbers for horizontal line hsync period The data byte number for this period is HSYNC__BYTE_CNT= HSYNC_CNT*lane_num
15:0	CFG_VPN_HBP_CNT	RW	0x0	VPN hbp Clock Count in byte clock domain. This parameter defines the byte clock cycle numbers for horizontal line back porch period - The data byte number for this period is HBP_BYTE_CNT= HBP_CNT*lane_num - Front porch clock count can be calculated by: HFP_CNT = HTOTAL_CNT - HSYNC_CNT - HACT_CNT - HBP_CNT - The data byte number for front porch period is HFP_BYTE_CNT= HFP_CNT*lane_num

12.3.4.38 DSI_VPN_TIMING_2 REGISTER

Offset: 0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_VACT_CNT	RW	0x0	VPN vact Line Count
15:0	CFG_VPN_VTOTAL_CNT	RW	0x0	VPN vtotal Line Count

12.3.4.39 DSI_VPN_TIMING_3 REGISTER

Offset: 0x11c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_VSYNC_CNT	RW	0x0	VPN vsync Line Count
15:0	CFG_VPN_VBP_CNT	RW	0x0	VPN vbp Line Count

12.3.4.40 DSI_VPN_WC_0 REGISTER

Offset: 0x120				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_HBP_WC	RW	0x0	VPN hbp packet payload data Byte Count This parameter must be programmed if HBP_PKT_EN is 0x1, otherwise it can be kept as 0x0 - If transmission mode is non-burst mode with sync pulse, the formula is: HBP_WC=HBP_BYTE_CNT−HSE_BYTE_CNT(4)−HBP_PKT_OVERHEAD(6) - If transmission mode is non-burst mode with sync event or burst mode, the formula is: HBP_WC=HSYNC_BYTE_CNT+HBP_BYTE_CNT−HSS_BYTE_CNT(4)−HBP_PKT_OVERHEAD(6)
15:0	CFG_VPN_HSA_WC	RW	0x0	VPN hsa packet payload data Byte Count This parameter must be programmed if HSA_PKT_EN is 0x1, otherwise it can be kept as 0x0 - If transmission mode is non-burst mode with sync pulse, the formula is: HSA_WC = HSYNC_BYTE_CNT - HSS_BYTE_CNT(4) - HSA_PKT_OVERHEAD(6) - Otherwise it is 0x0

12.3.4.41 DSI_VPN_WC_1 REGISTER

Offset: 0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_HFP_WC	RW	0x0	VPN hfp packet payload data Byte Count This parameter must be programmed if HFP_PKT_EN is 0x1, otherwise it can be kept as 0x0 - If transmission mode is non-burst mode with sync pulse, or non-burst mode with sync event, the formula is: HFP_WC = HFP_BYTE_CNT - HACT_PKT_OVERHEAD(6) - HFP_PKT_OVERHEAD(6) - If transmission mode is burst mode and HEX_PKT_EN = 1, the formula is: HFP_WC = HFP_BYTE_CNT - HACT_PKT_OVERHEAD(6) - HFP_PKT_OVERHEAD(6) - If transmission mode is burst mode and HEX_PKT_EN = 0, the formula is:

Offset: 0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
				HFP_WC = HFP_BYTE_CNT + (HACT_BYTE_CNT - HACT_WC) - HACT_PKT_OVERHEAD(6) - HFP_PKT_OVERHEAD(6)
15:0	CFG_VPN_HACT_WC	RW	0x0	VPN hact packet payload data Byte Count This parameter is equal to Active pixel RGB data total byte count

12.3.4.42 DSI_VPN_WC_2 REGISTER

Offset: 0x128				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_HEX_WC	RW	0x0	VPN hex packet payload data Byte Count This parameter must be programmed if HEX_PKT_EN is 0x1, otherwise it can be kept as 0x0 - If transmission mode is burst mode, the formula is: HEX_WC = HACT_BYTE_CNT - HACT_WC - HEX_PKT_OVERHEAD(6) - Otherwise HEX_WC = 0
15:0	CFG_VPN_HLP_WC	RW	0x0	VPN hlp packet payload data Byte Count This parameter must be programmed if HLP_PKT_EN is 0x1, otherwise it can be kept as 0x0 - If transmission mode is non-burst mode with sync pulse, the formula is: HLP_WC = HTOTAL_BYTE_CNT - HSYNC_BYTE_CNT - HSE_BYTE_CNT(4) - HLP_PKT_OVERHEAD(6) - If transmission mode is non-burst mode with sync event or burst mode, the formula is: HLP_WC = HTOTAL_BYTE_CNT - HSS_BYTE_CNT(4) - HLP_PKT_OVERHEAD(6)

12.3.4.43 DSI_VPN_SLOT_CNT_0 REGISTER

Offset: 0x130				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_SLOT_SP_CNT	RW	0x0	VPN Time Slot Count for Short Packet. This parameter defines a MIN slot period for short packet transmission, which should ensure DPHY can go to low power, send the short packet, and return to HS again in time for the next active panel packet which has a strict timing

Offset: 0x130				
Bits	**Field**	**Type**	**Reset**	**Description**
				requirement. If any DSI active panel data flow is working, and CPU or smart interface wants to send short packet between the active panel packets, the internal state machine will try to find a time slot between active panel packets which has a larger period than the defined value. DSI will only send CPU or Command Panel short packet during such slot to ensure DPHY has enough time to go to low power, send the packet, and return to HS again in time for next active panel packet which has a strict timing requirement. The programming of this parameter is necessary only when multiple panels or data paths are working simultaneously.
15:0	CFG_VPN_SLOT_LP_CNT	RW	0x0	VPN Time Slot Count for Long Packet. This parameter defines a minimum slot period for long packet transmission, which should ensure DPHY can enter low power mode, send the long packet, and return to HS again in time for the next active panel packet which has a strict timing requirement. The programming of this parameter is necessary only when multiple panels or data paths are working simultaneously.

12.3.4.44 DSI_VPN_SLOT_CNT_1 REGISTER

Offset: 0x134				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_SLOT_TX_LP_CNT	RW	0x0	VPN Time Slot Count for Low Power packet TX. This parameter defines a minimum slot period for Low Power packet transmission, which should ensure DPHY can enter low power mode, send the Low Power packet, and return to HS again in time for the next active panel packet which has a strict timing requirement. The programming of this parameter is necessary only when multiple panels or data paths are working simultaneously.
15:0	CFG_VPN_SLOT_TN_CNT	RW	0x0	VPN Time Slot Count for Bus Turn Around. This parameter defines a minimum slot period for short packet transmission, which should ensure DPHY can enter low power mode, turn around the bus, and return to HS again in time for the next active panel packet which has a strict timing requirement.

12.3.4.45 DSI_VPN_SYNC_CODE REGISTER

Offset: 0x138				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	RO	0	Reserved for future use
29:24	CFG_VPN_HSE_CODE	RW	0x31	MIPI DSI Hsync End Code
23:22	RSVD	RO	0	Reserved for future use
21:16	CFG_VPN_HSS_CODE	RW	0x21	MIPI DSI Hsync Start Code
15:14	RSVD	RO	0	Reserved for future use
13:8	CFG_VPN_VSE_CODE	RW	0x11	MIPI DSI Vsync End Code
7:6	RSVD	RO	0	Reserved for future use
5:0	CFG_VPN_VSS_CODE	RW	0x01	MIPI DSI Vsync Start Code

12.3.4.46 DSI_VPN_STATUS_0 REGISTER

Offset: 0x140				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CFG_VPN_RD_ERR	RO	0x0	VPN input buffer read error It includes - CFG_VPN_RD_2EARLY - CFG_VPN_LINE_MISS - CFG_VPN_RD_UNDERRUN
30	CFG_VPN_LINE_MISS	RO	0x0	VPN input buffer line miss This indicates a whole H line pixel data is missed.
29	CFG_VPN_RD_2EARLY	RO	0x0	VPN input buffer read too early
28	CFG_VPN_RD_UNDERRUN	RO	0x0	VPN input buffer underrun
27	CFG_VPN_BF_FULL	RO	0x0	VPN input buffer full
26	CFG_VPN_RD_DELAY_ERR	RO	0x0	VPN Request Delay Error at arbiter
25:21	RSVD	RO	0	Reserved for future use
20:0	CFG_VPN_STATUS_0	RO	0x811	DSI VPN Status Register for debug purpose It includes the following fields: - l1_lcd 4:0_cs - l1_vst 6:0 - l1_hst 8:0

12.3.4.47 DSI_VPN_STATUS_1 REGISTER

Offset: 0x144				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_WRDONE_RDDONE_CNT	RO	0x0	VPN Input Buffer Write Done to input buffer Read Done Clock Count This could help to tune the vsync delay count.
15:0	CFG_VPN_WR2RD_CNT	RO	0x0	VPN Input Buffer Write to input buffer Read Clock Count. This could help to tune the vsync delay count.

12.3.4.48 DSI_VPN_STATUS_2 REGISTER

Offset: 0x148				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_UNDERRUN_CNT	RO	0x0	VPN input buffer underrun count
15:0	CFG_VPN_RD_DATWR_CNT	RO	0x0	VPN input buffer read to data write count

12.3.4.49 DSI_VPN_STATUS_3 REGISTER

Offset: 0x14c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_VPN_REQ_ARB_DLY_CNT	RO	0x0	VPN tx request delay count at arbiter interface
15:0	CFG_VPN_REQ_PHY_DLY_CNT	RO	0x0	VPN tx request delay count at dphy interface

12.3.4.50 DSI_VPN_STATUS_4 REGISTER

Offset: 0x150				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFG_VPN_FRM_CNT	RO	0x0	DSI VPN TX frame count

12.3.4.51 DSI_PHY_CTRL_0 REGISTER

Offset: 0x180				
Bits	**Field**	**Type**	**Reset**	**Description**
31	CFG_RX_TRG_REG_DIS	RW	0x0	Disable Register for low power rx trigger signals. Note: Internal use only
30	CFG_TX_LANE_0	RW	0x0	New packet tx start from lane 0: If two packets are transferred continuously, all data is packed and distributed to all enabled lanes. The second packet could start from any lane. 1: Transmission of every new packet starts from lane 0. If two packets are transferred continuously, and the first packet doesn't occupy all lanes, then extra byte of 0 will be inserted in at the end of first packet to ensure the second packet start from lane 0 This is an debug option and should be set to 0
29:28	RSVD	RO	0	Reserved for future use
27	CFG_FCLK_NOT	RW	0x0	Reverse Input Byte Clock from DPHY to DSI Control Logic The output data to DPHY should be valid at the falling edge of the byte clock.
26:24	RSVD	RO	0	Reserved for future use
23:16	CFG_STOP_ST_CNT	RW	0x10	DPHY stops state count Defines the stop state count for TXLP and PHY control
15:8	CFG_RX_DLY_CNT	RW	0x30	DPHY rx_delay count Defines the delay state count for RX control
7:0	RSVD	RO	0	Reserved for future use

12.3.4.52 DSI_PHY_CTRL_1 REGISTER

Offset: 0x184				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	RSVD	RO	0	Reserved for future use
17	CFG_VDD_ANA_VALID	RW	0x1	DPHY Analog VDD Valid
16	CFG_VDD_DVM_VALID	RW	0x1	DPHY Digital VDD Valid

Offset: 0x184				
Bits	**Field**	**Type**	**Reset**	**Description**
15:3	RSVD	RO	0	Reserved for future use
2	CFG_ULPS_REQ_BYTE	RW	0x0	DPHY All Lane Force to ULPS
1	CFG_TX_ULPS_CLK_ESC	RW	0x0	DPHY clk Lane Force to ULPS
0	CFG_CONT_CLK_HS	RW	0x0	DPHY Clock Lane Continuous Clocking in HS

12.3.4.53 DSI_PHY_CTRL_2 REGISTER

Offset: 0x188				
Bits	**Field**	**Type**	**Reset**	**Description**
31:15	RSVD	RO	0	Reserved for future use
14	CFG_CSR_HSTX_RX_EN	RW	0x0	RX enable when DPHY HSTX 0x0: Disable 0x1: Enable
13:12	CFG_CSR_LANE_MAP	RW	0x0	DPHY Data map to lane order 0x0: Lane0, Lane1, Lane2, Lane3 0x1: Lane0, Lane3, Lane1, lane2 0x2: Lane0, Lane2, Lane3, Lane1 0x3: Reserved
11:8	CFG_CSR_LANE_RESC_EN	RW	0x0	DPHY LP Receiver Enable Enable the reverse escape LP receiver. Lane immediately transmits to receive mode.
7:4	CFG_CSR_LANE_EN	RW	0x0	DPHY Data Lane Enable
3:0	CFG_CSR_LANE_TURN	RW	0x0	DPHY Bus Turn Around This field indicates that the protocol desires to turn the lane around, allowing the other side to begin transmitting.

12.3.4.54 DSI_PHY_CTRL_3 REGISTER

Offset: 0x18c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	RSVD	RO	0	Reserved for future use
9	CFG_FORCECLK_HIZ_HS	RW	0x0	DPHY clk Lane Force to High-Z in HS Mode
8	CFG_FORCECLK_HIZ_LP	RW	0x0	DPHY clk Lane Force to High-Z in LP mode

Offset: 0x18c				
Bits	**Field**	**Type**	**Reset**	**Description**
7:4	CFG_FORCE_HIZ_HS	RW	0x0	DPHY Force Data Lane to High-Z in HS Mode
3:0	CFG_FORCE_HIZ_LP	RW	0x0	DPHY Data Lane Force to High-Z in LP Mode

12.3.4.55 DSI_PHY_STATUS_0 REGISTER

Offset: 0x190				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	DPHY_RDY_HS_BYTE	RWC	0x0	DPHY HS TX ready signals
27:24	TX_REQ_HS_BYTE	RWC	0x0	DPHY HS TX request signals
23:20	RSVD	RO	0	Reserved for future use
19:16	DPHY_LANE_RX_LINE_ERR	RWC	0x0	PPI ErrControl Illegal line state detected
15:12	DPHY_ERR_SYNC_ESC	RWC	0x0	PPI ErrSyncEsc Partial byte detected
11:8	DPHY_ERR_ESC	RWC	0x0	PPI ErrEsc Invalid esc command detected
7:4	DPHY_ERR_CONT_LP0	RWC	0x0	PPI ErrContentionLP0 Contention detect
3:0	DPHY_ERR_CONT_LP1	RWC	0x0	PPI ErrContentionLP0 Contention detect

12.3.4.56 DSI_PHY_STATUS_1 REGISTER

Offset: 0x194				
Bits	**Field**	**Type**	**Reset**	**Description**
31	DPHY_ULP_STATE_BYTE	RO	0x1	All lanes are in ULP state
30	DPHY_STOP_STATE_BYTE	RO	0x1	PPI Stopstate - All lanes in stop state
29	DPHY_CLK_ULPS_ACTIVE_N	RO	0x1	PPI clock UlpsActiveNot
28	DPHY_RX_CLK_ULPS_N	RO	0x1	PPI RxUlpsClkNot
27:24	DPHY_LANE_DIR	RO	0x0	PPI Direction
23:20	DPHY_ULPS_ACTIVE_N	RO	0xf	PPI UlpsActiveNot
19:16	DPHY_LANE_RX_LINE_ERR	RO	0x0	PPI ErrControl - Illegal line state detected
15:12	DPHY_ERR_ESC	RO	0x0	PPI ErrEsc - Invalid ESC command detected
11:8	DPHY_ERR_SYNC_ESC	RO	0x0	PPI ErrSyncEsc - Partial byte detected
7:4	DPHY_ERR_CONT_LP0	RO	0x0	PPI ErrContentionLP0 - Contention detect
3:0	DPHY_ERR_CONT_LP1	RO	0x0	PPI ErrContentionLP0 - Contention detect

12.3.4.57 DSI_PHY_LPRX_0 REGISTER

Offset: 0x198				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	DPHY_LANE_RX_TRG3	RO	0x0	dphy_lane_rx_trg3
27:24	DPHY_LANE_RX_TRG2	RO	0x0	dphy_lane_rx_trg2
23:20	DPHY_LANE_RX_TRG1	RO	0x0	dphy_lane_rx_trg1
19:16	DPHY_LANE_RX_TRG0	RO	0x0	dphy_lane_rx_trg0
15:12	DPHY_LANE_RX_ULPS	RO	0x0	dphy_lane_rx_ulps
11:8	DPHY_LANE_RX_LPDT	RO	0x0	dphy_lane_rx_lpdt
7:4	DPHY_LANE_RX_DVALID	RO	0x0	dphy_lane_rx_dvalid
3:0	DPHY_LANE_RX_CLK	RO	0x0	dphy_lane_rx_clk

12.3.4.58 DSI_PHY_LPRX_1 REGISTER

Offset: 0x19c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	DPHY_LANE_DOUT_RX	RO	0x0	dphy_lane_dout_rx

12.3.4.59 DSI_PHY_LPTX_0 REGISTER

Offset: 0x1a0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	DPHY_TX_TRIGGER_ESC_L	RO	0x0	tx_trigger_esc[11:0]
19:16	DPHY_TX_ULPS_ESC	RO	0x0	tx_ulps_esc
15:12	DPHY_TX_LPDT_ESC	RO	0x0	tx_lpdt_esc
11:8	DPHY_TX_VALID_ESC	RO	0x0	tx_valid_esc
7:4	DPHY_TX_REQ_ESC	RO	0x0	tx_req_esc
3:0	DPHY_LANE_RDY_ESC	RO	0x0	dphy_lane_rdy_esc

12.3.4.60 DSI_PHY_LPTX_1 REGISTER

Offset: 0x1a4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	RO	0	Reserved for future use
3:0	DPHY_TX_TRIGGER_ESC_H	RO	0x0	tx_trigger_esc[15:12]

12.3.4.61 DSI_PHY_LPTX_2 REGISTER

Offset: 0x1a8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	DPHY_TX_DATA_ESC	RO	0x0	tx_data_esc

12.3.4.62 DSI_PHY_STATUS_2 REGISTER

Offset: 0x1ac				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	CFG_TX_REQ_CNT_R	RO	0x0	TX previous request to ready delay count
15:0	CFG_TX_REQ_CNT	RO	0x0	TX request to ready delay count

12.3.4.63 DSI_PHY_TIME_0 REGISTER

Offset: 0x1c0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	CFG_CSR_TIME_HS_EXIT	RW	0x0	Length of HS Exit Period in tx_clk_esc Cycles This field is used for the time to drive LP-11 after a HS burst. The HS exit period is calculated as: HS Exit Period = (1 + CFG_CSR_HS_EXIT) / 66 MHz By default, the DPHY escape clock frequency is 66 MHz. According to the MIPI specification, the minimum value for this period is 100 ns.
23:16	CFG_CSR_TIME_HS_TRAIL	RW	0x0	DPHY HS Trail Period Length This field is used for the time to drive the flipped differential state after the last payload data bit of a HS transmission burst. The length of the HS trail period is in tx_clk_esc cycles.

Offset: 0x1c0				
Bits	**Field**	**Type**	**Reset**	**Description**
				HS Trail Time = (1 + CFG_CSR_HS_TRAIL) / 66 MHz According to the MIPI specification, the minimum value is defined by: max(8 * UI, 60 ns + 4 * UI)
15:8	CDG_CSR_TIME_HS_ZERO	RW	0x0	DPHY HS Zero State Length This field is used for the time to drive HS-0 before the sync sequence. The length of the HS zero state is in tx_clk_esc cycles. The HS zero state length should be: HS Zero State Length ≥ (CFG_CSR_TIME_ZERO/ 66 MHz + 3 * Tbyte_clk According to the MIPI specification, the minimum value for the sum of Time HS Prep and Time HS Zero is 145 ns + 10 * UI.
7:0	CFG_CSR_TIME_HS_PREP	RW	0x0	DPHY HS Prepare State Length This field is used for the time to drive LP-00 to prepare for HS transmission. It represents the length of the HS prepare state period in tx_clk_esc cycles. Time HS Prep = (1 + CFG_CSR_TIME_HS_PREP) / 66 MHz According to the MIPI specification for DPHY - the minimum value for this parameter is 40 ns + 4 * UI, and - the maximum value is 85 ns + 6 * UI.

12.3.4.64 DSI_PHY_TIME_1 REGISTER

Offset: 0x1c4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	CFG_CSR_TIME_TA_GET	RW	0x0	Time to Drive LP-00 by New Transmitter in tx_clk_esc cycles TA Get Time = (1 + CFG_CSR_TIME_TA_GET)/66 MHz According to the MIPI specification, the typical value is 5 * Tlpx, where Tlpx is the DPHY LP length: Tlpx = (1 + CFG_CSR_TIME_LPX) / 66 MHz.
23:16	CFG_CSR_TIME_TA_GO	RW	0x0	Time to Drive LP-00 after Turn Request in tx_clk_esc Cycles

Offset: 0x1c4				
Bits	**Field**	**Type**	**Reset**	**Description**
				TA Go Time = (1 + TA_GO)/66 MHz According to the MIPI specification, the typical value is 4*Tlpx.
15:0	CFG_CSR_TIME_WAKEUP	RW	0x0	DPHY HS Wakeup Period Length This field is the recovery time from Ultra-Low Power State (ULPS). Twakeup = (1 + CFG_CSR_TIME_WAKEUP)/66 MHz According to the MIPI specification, the minimum value is 1 ms.

12.3.4.65 DSI_PHY_TIME_2 REGISTER

Offset: 0x1c8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	CFG_CSR_TIME_CK_EXIT	RW	0x0	DPHY CLK Exit Period Length in tx_clk_esc cycles Tck_exit = (1 + CFG_CSR_TIME_CK_EXIT)/66 MHz This field should use the same value as CFG_CSR_TIME_HS_EXIT
23:16	CFG_CSR_TIME_CK_TRAIL	RW	0x0	DPHY CLK Trail Period Length in tx_clk_esc cycles This field is the time to drive HS differential state after the last payload clock bit of an HS transmission burst. CLK Trail Time = (1 + CFG_CSR_TIME_CK_TRAIL)/66 MHz According to the MIPI specification, the minimum value is 60 ns.
15:8	CFG_CSR_TIME_CK_ZERO	RW	0x0	DPHY CLK Zero State Length in tx_clk_esc cycles This field is the time for lead HS-0 drive period before starting the clock. Tck_zero = (1 + CFG_CSR_TIME_CK_ZERO)/66 MHz According to the MIPI specification, the minimum value for (Tck_prep+Tck_zero) is 300 ns, where in <var Product Number> Tck_prep is the same as Time HS Prep defined by CFG_CST_TIME_HS_PREP.
7:0	CFG_CSR_TIME_CK_LPX	RW	0x0	DPHY CLK LP Length This field is the length of CLK Low Power state period in tx_clk_esc cycles. CLK Lpx Time = Tck_lpx = (1 + CFG_CSR_TIME_CK_LPX) / 66 MHz This field should be set to the same value as CFG_CST_TIME_LPX.

12.3.4.66 DSI_PHY_TIME_3 REGISTER

Offset: 0x1cc				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	RSVD	RO	0	Reserved for future use
15:8	CFG_CSR_TIME_LPX	RW	0x0	DPHY LP Length This field is the length of any Low Power state period in tx_clk_esc cycles. Lpx Time = Tlpx = (1 + CFG_CSR_TIME_LPX) / 66 MHz According to the MIPI specification, the minimum value is 50 ns.
7:0	CFG_CSR_TIME_REQRDY	RW	0x0	DPHY HS Request to Ready Length This field is the minimum byte clock cycles of DSI HS TX request to DPHY ready. Sometimes it may be important for this length to be consistent to maintain precise Vertical and Horizontal timing. In most cases, this parameter should be kept at the default of 0x0. Total cycles between DSI HS TX request to DPHY ready are composed by DPHY: clock lane timing, gap, data lane timing, and some other items inside DPHY. - Clock lane timing = (2*(CFG_CSR_TIME_CK_LPX + 1) + (CFG_CSR_TIME_HS_PREP + 1) + (CFG_CSR_TIME_CK_ZERO + 1))/66 MHz - Gap = (16UI +2)/66 MHz - Data lane timing = ((2*CFG_CSR_TIME_LPX + 1) + (CFG_CSR_TIME_HS_PREP + 1) + (CFG_CSR_TIME_HS_ZERO + 1))/66 MHz The formula to calculate the total cycles is: (1 + CFG_CSR_TIME_REQRDY)/frequecy_byte_clk = (clock lane timing + gap + data lane timing + 10/66 MHz) Alternative method to get the value: - After the DSI active panel data flow is running, read the value from Bits [7:0] of the DSI_PHY_STATUS_2 register. This gives the current clock cycle delay between the DSI TX request and DPHY ready. - Add 2 to the value obtained from the DSI_PHY_STATUS_2 register The formula is: CFG_CSR_TIME_REQRDY=(Value read from DSI_PHY_STATUS_2)+2

12.3.4.67 DSI_PHY_CODE_0 REGISTER

Offset: 0x1d0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	CFG_TRIG3_CODE	RW	0x05	DPHY Trigger 3 Code
23:16	CFG_TRIG2_CODE	RW	0x84	DPHY Trigger 2 Code
15:8	CFG_TRIG1_CODE	RW	0xBA	DPHY Trigger 1 Code
7:0	CFG_TRIG0_CODE	RW	0x46	DPHY Trigger 0 Code

12.3.4.68 DSI_PHY_CODE_1 REGISTER

Offset: 0x1d4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	CFG_CSR_ULPS_CODE	RW	0x78	DPHY Ultra Low Power Code
23:16	CFG_CSR_LPDT_CODE	RW	0x87	DPHY Low Power Data Transfer Code
15:0	RSVD	RO	0	Reserved for future use

12.3.4.69 DSI_PHY_ANA_PWR_CTRL REGISTER

Offset: 0x1e0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	RSVD	RO	0	Reserved for future use
8	CFG_DPHY_ANA_RESETB	RW	0x1	DPHY Analog reset 0: Reset DPHY analog 1: De-reset DPHY analog
7:1	RSVD	RO	0	Reserved for future use
0	CFG_DPHY_ANA_PU	RW	0x1	DPHY Analog power up 0: Power down DPHY analog 1: Power up DPHY analog

12.3.4.70 DSI_PHY_ANA_CTRL0 REGISTER

Offset: 0x1e4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:29	CFG_DPHY_LPRX	RW	0x2	LPRX reference voltage high (20mV per

Offset: 0x1e4				
Bits	**Field**	**Type**	**Reset**	**Description**
	_VTTH			stage) 000: 760mV 001: 780mV 010: 800mV 111: 900 mV
28:26	CFG_DPHY_LPRX_VTHL	RW	0x3	LPRX reference voltage low (20mV per stage) 000: 540mV 001: 560mV 010: 580mV 011: 600mV 111: 680 mV
25:24	CFG_DPHY_LPTX_RES	RW	0x2	LPRX driver impedance control 10: 200 Ohm
23:21	CFG_DPHY_HSTX_RES	RW	0x4	HSTX driver impedance control. 000: 120 Ohm (differential @TT) 100: 100 Ohm 111: 89 ohm
20	CFG_DPHY_HSTX_LP	RW	0x0	Low power mode for hstx drivers. 1 = Lower power 0 = Normal
19:17	CFG_DPHY_ADJ_DLY_CK	RW	0x0	Adjust delay of CH_CK hstx driver output to manage skew between channels. The delay is 30ps per stage (@tt) . 000: 0 stage 111: 7 stages
16	CFG_DPHY_EN_CH_CK	RW	0x1	Enable for CH_CK. When disabled, CH_CK is power down mode, pad_ckp and pad_ckn are in high-z mode 0: Disable 1: Enable
15:13	CFG_DPHY_ADJ_DLY3	RW	0x0	Adjust delay of CH3 hstx driver output to manage skew between channels. The delay is 30ps per stage (@tt) . 000: 0 stage 111: 7 stages
12	CFG_DPHY_EN_CH3	RW	0x1	Enable for CH3. When disabled, CH3 is power down mode, pad_dn3 and pad_p3 are in high-z mode 0: Disable 1: Enable

Offset: 0x1e4				
Bits	**Field**	**Type**	**Reset**	**Description**
11:9	CFG_DPHY_ADJ_DLY2	RW	0x0	Adjust delay of CH2 hstx driver output to manage skew between channels. The delay is 30ps per stage (@tt) . 000: 0 stage 111: 7 stages
8	CFG_DPHY_EN_CH2	RW	0x1	Enable for CH2. When disabled, CH2 is power down mode, pad_dn2 and pad_p2 are in high-z mode 0: Disable 1: Enable
7:5	CFG_DPHY_ADJ_DLY1	RW	0x0	Adjust delay of CH1 hstx driver output to manage skew between channels. The delay is 30ps per stage (@tt) . 000: 0 stage 111: 7 stages
4	CFG_DPHY_EN_CH1	RW	0x1	Enable for CH1. When disabled, CH1 is power down mode, pad_dn1 and pad_p1 are in high-z mode 0: Disable 1: Enable
3:1	CFG_DPHY_ADJ_DLY0	RW	0x0	Adjust delay of CH0 hstx driver output to manage skew between channels. The delay is 30ps per stage (@tt) . 000: 0 stage 111: 7 stages
0	CFG_DPHY_EN_CH0	RW	0x1	Enable for CH0. When disabled, CH0 is power down mode, pad_dn0 and pad_p0 are in high-z mode 0: Disable 1: Enable

12.3.4.71 DSI_PHY_ANA_CTRL1 REGISTER

Offset: 0x1e8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	RO	0	Reserved for future use
23	CFG_CLK_SEL	RW	0x0	DPHY bit clk select 0: PLL_DIV output 1: mipi_bit_clk mux output
22:21	RSVD	RO	0	Reserved for future use

Offset: 0x1e8				
Bits	**Field**	**Type**	**Reset**	**Description**
20	CFG_SWAP_PN_CH3	RW	0x0	swap pn polarity for ch3
19	CFG_SWAP_PN_CH2	RW	0x0	swap pn polarity for ch2
18	CFG_SWAP_PN_CHCK	RW	0x0	swap pn polarity for ch ck
17	CFG_SWAP_PN_CH1	RW	0x0	swap pn polarity for ch1
16	CFG_SWAP_PN_CH0	RW	0x0	swap pn polarity for ch0
15	CFG_SET_TEST	RW	0x0	Select analog phy self-test 1: enable phy self-test. At this mode, phybypass all control signal 0: disable phy self-test.
14	DFG_SET_TEST_LP	RW	0x0	Select LP or HS self-test mode. It is only valid when sel_test is high. 1: LP mode self test 0: HS mode self test
13:12	DFG_TEST_PATTERN	RW	0x0	select self-test pattern generation 00: All 0 01: All 1 10: CK pattern (0101) 11: PRBS7
11	CFG_EN_CLK_DIV2	RW	0x0	enable half rate HSTX mode. When enabled, PHY will work at half the data rate of PLL input clock. 1: enable 0: disable
10:8	CFG_DPHY_HSTX_VREF	RW	0x3	hstx vreg control (20mV per stage) 000: 340mV 001: 360mV 011: 400mV 111: 480 mV
7:5	CFG_DPHY_LPCD_VTHH	RW	0x5	lpcd reference voltage high (20mV per stage) 000: 300mV 001: 320mV 101: 400mV 111: 440 mV
4:2	CFG_DPHY_LPCD_VTHL	RW	0x2	lpcd reference voltage low (20mV per stage) 000: 200mV 001: 220mV 010: 240mV

Offset: 0x1e8				
Bits	**Field**	**Type**	**Reset**	**Description**
				111: 340 mV
1	CFG_DPHY_PULL_DN	RW	0x1	Pull down enable: 1: Enable pull down PAD IO for ch1,ch2,ch3, check when both HSTX and LPTX are not enabled. 0: Disable pull down PAD IO for ch1,ch2,ch3, check when both HSTX and LPTX are not enabled. It's high z in this state.
0	CFG_DPHY_PULL_DN_CH0	RW	0x1	Pull down enable for ch0 1: Enable pull down PAD IO when all HSTX, LPTX and LPRX are not enabled. 0: Disable pull down PAD IO when both HSTX and LPTX are not enabled. It's high z in this state.

12.3.4.72 DSI_PHY_DEBUG REGISTER

Offset: 0x1ec				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	RSVD	RO	0	Reserved for future use
0	CFG_DDR_CLK_SEL	RW	0x1	DDR Clock Select 0: First bit is sent on DDR clock falling edge 1: First bit is sent on DDR clock rising edge

12.3.5 SPI LCD Display Interface

12.3.5.1 Introduction

The SPI LCD Display Interface is used to

- Send image data commands
- Read image data
- Transmit image data

It supports the operational modes

- Single data line mode
- Dual data line mode

where each of which support the work modes

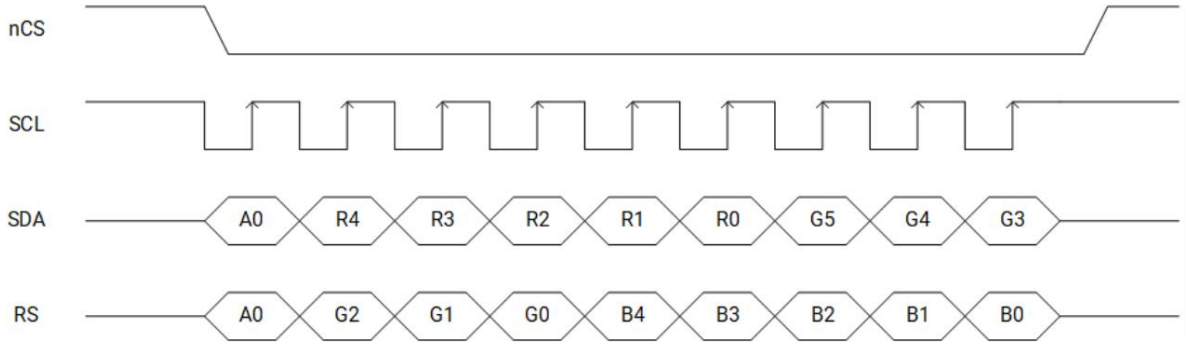
- 3-line/9bit mode
- 4-line/8bit mode

By software, it is possible to configure which line will be the first for transmitting data. Further, it is possible to configure the transfer mode choosing between

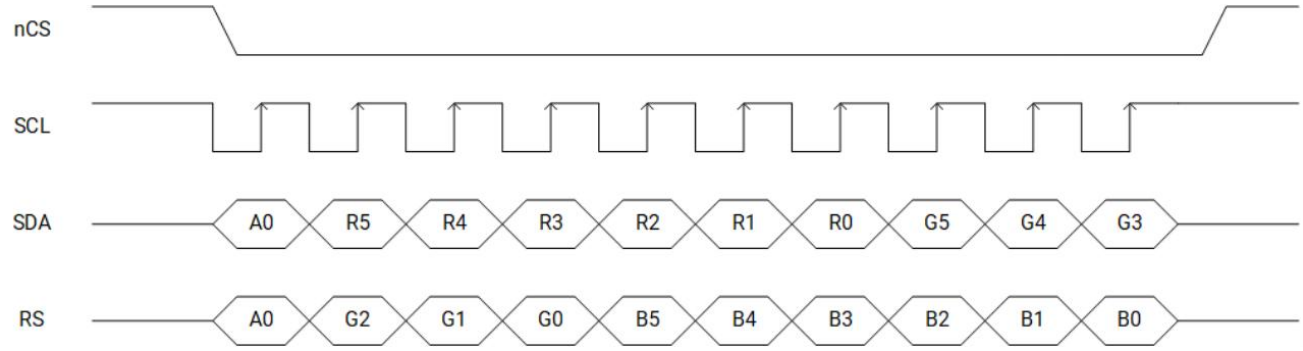
- Packet transfer mode
- Unpacked transfer mode

As example, below are depicted the transfers modes for some color formats, highlighting how data are organized and transmitted.

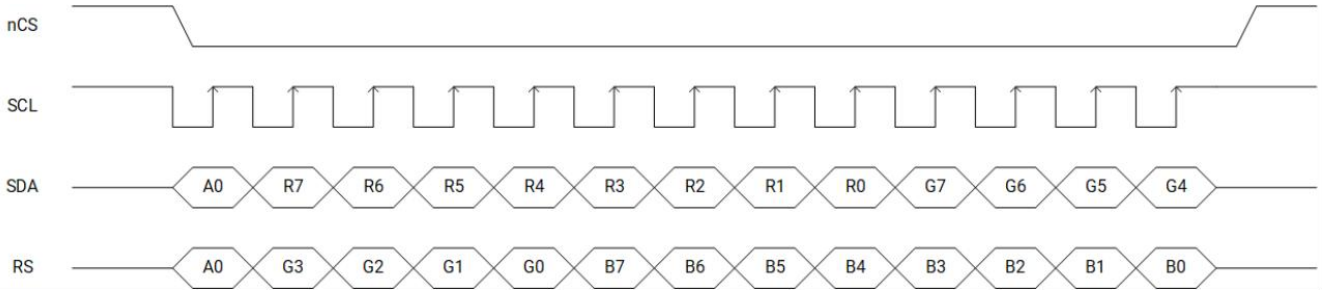
[Packet transfer mode for RGB565]



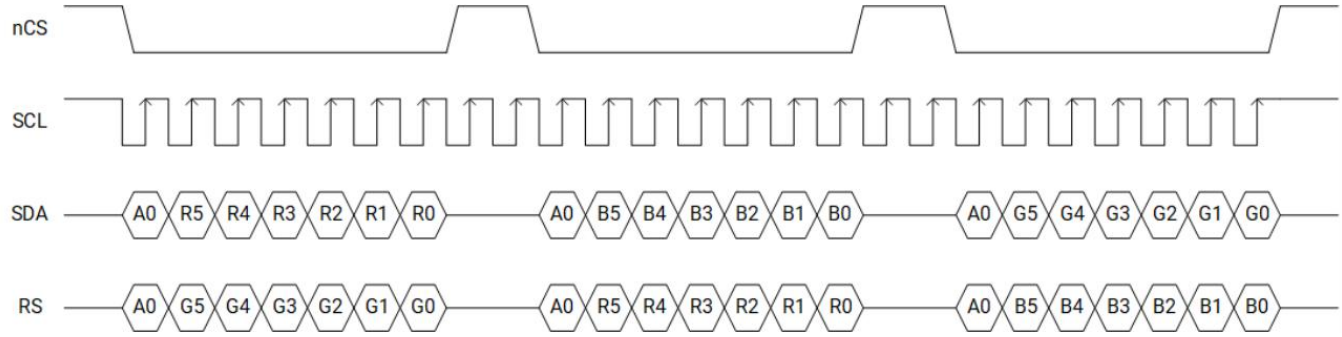
[Packet transfer mode for RGB666]



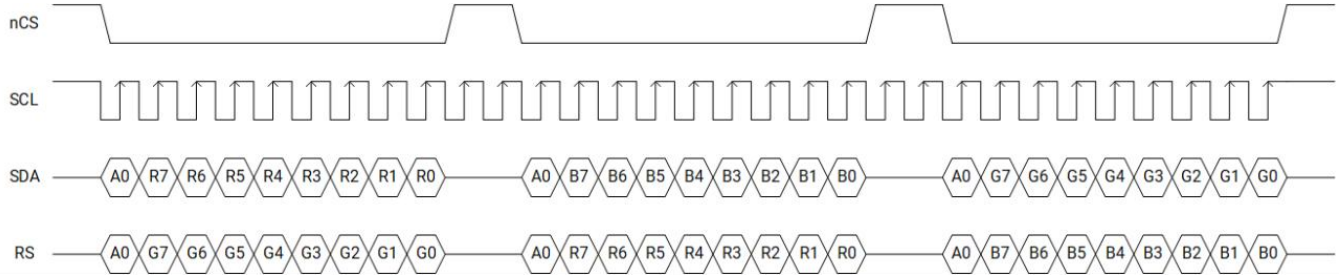
[Packet transfer mode for RGB888]



[Unpacked transfer mode for RGB666]



[Unpacked transfer mode for RGB888]



12.3.5.2 Features

- Support for SPI LCD module with resolution up to 320x240
- Support for 3-/4-line Serial Peripheral Interface (SPI) and 2-line SPI data transmission
- Support for up to 3 simultaneous overlays (2 for RGB, 1 for YUV & RGB)
- Support for dithering
- Support for gamma curve
- Alpha blending with configurable alpha values or per-pixel alpha blending
- Support for YUV to RGB color space conversion
- Support for image scaling
- Support for color keying
- Support for memory write-back
- Support for the following **input formats for image layer**:
  - YUV422 planar
  - YUV422 packet
  - YUV420 planar
  - RGB888
  - RGB565
  - RGB666

- BGR888
- BGR565
- BGR666

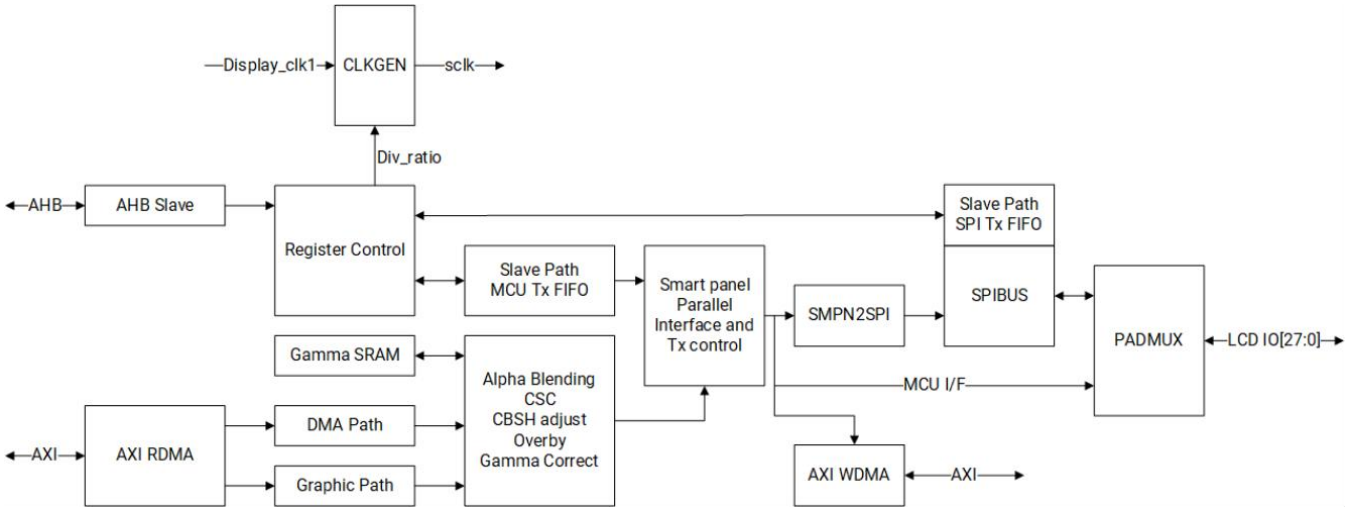
Note. As can be seen, it is supported **R-B swap option** for the sake of flexibility

- Support for the following **input formats for OSD layer**:
  - RGB888
  - RGB565
  - RGB666
  - BGR888
  - BGR565
  - BGR666

Note. As can be seen, it is supported **R-B swap option** for the sake of flexibility

12.3.5.3 Block Diagram

The architecture of the SPI LCD Display Interface is depicted below.



It is clearly understandable how the display data are efficiently processed, then converted into SPI-compatible signals, then transmitted to the connected LCD display.

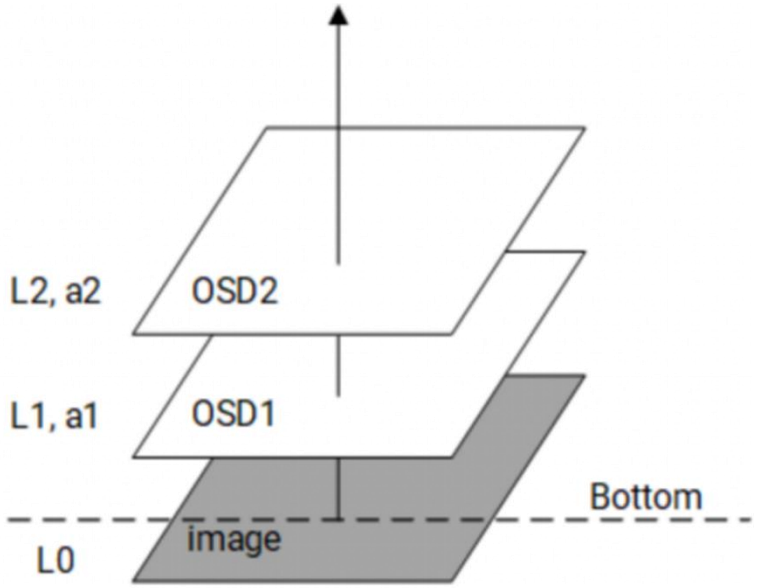
12.3.5.4 Functions

12.3.5.4.1 Blending Function

The blending function of the DSI controller is used to combine multiple layers of images or overlays with different levels of transparency (alpha values).

An example of layers and their respective alpha values is depicted below, where

- **L0**: Bottom layer, base image
- **L1**: Middle layer, alpha value **a1**
- **L2**: Top layer, alpha value **a2**



The following blending modes are supported:

- Normal Alpha Blending Mode
- Pre-Multiple Alpha Blending Mode
- Special Alpha Blending Mode

In the code, a different formula is implemented for each blending mode that uses the alpha value **a1** as per the following conditions:

```
if (L1 == color_key)
a1 = 8'h0;
else if (layer_alpha_sel == 1)
a1 = layer_alpha;
else
a1 = pixel_alpha;
```

Details for each blending mode are explained in the following subsections.

12.3.5.4.1.1 Normal Alpha Blending Mode

With reference to the example figure shown above,

- For **2 layers**, the formula implemented is

L'=L1×a1+L0×(1-a1)

- For **3 layers** (not recommended), the formula implemented is

  L'=L2×a2+L1×a1×(1-a2)+L0×(1-a1)×(1-a2)

  Note. Alpha value is not supported for write-back in this case

In the code, the pixel value **L'** depends on the alpha value **a1** as per the following conditions:

```
if (a1 == 8'hFF)
L' = L1;
else if (a1 == 8'h00)
L' = L0;
else
L' = (L1-L0) × a1/256 + L0
```

12.3.5.4.1.2 Pre-Multiple Alpha Blending Mode

With reference to the example figure shown above,

- For **2 layers**, the formula implemented is

  L'=L1+L0×(1−a1)

- For **3 layers** (not recommended), the formula implemented is

  L'=L2+L1×(1−a2)+L0×(1−a1)×(1−a2)

  Note. Alpha value is supported for write-back and its value is given by the formula a'=a1+a2−a1×a2

In the code, the pixel value **L'** depends on the alpha value **a1** as per the following conditions:

```
if (a1 == 8'hFF)
L' = L1;
else if (a1 == 8'h00)
L' = L0;
else
L' = L1-L0 × (1-a1)/256;
```

12.3.5.4.1.3 Special Alpha Blending Mode

With reference to the example figure shown above,

- For **2 layers**, the formula implemented is

L'=L1+L0×a1

- For **3 layers** (not recommended), the formula implemented is

  L'=L2+L1×a2+L0×a1×a2

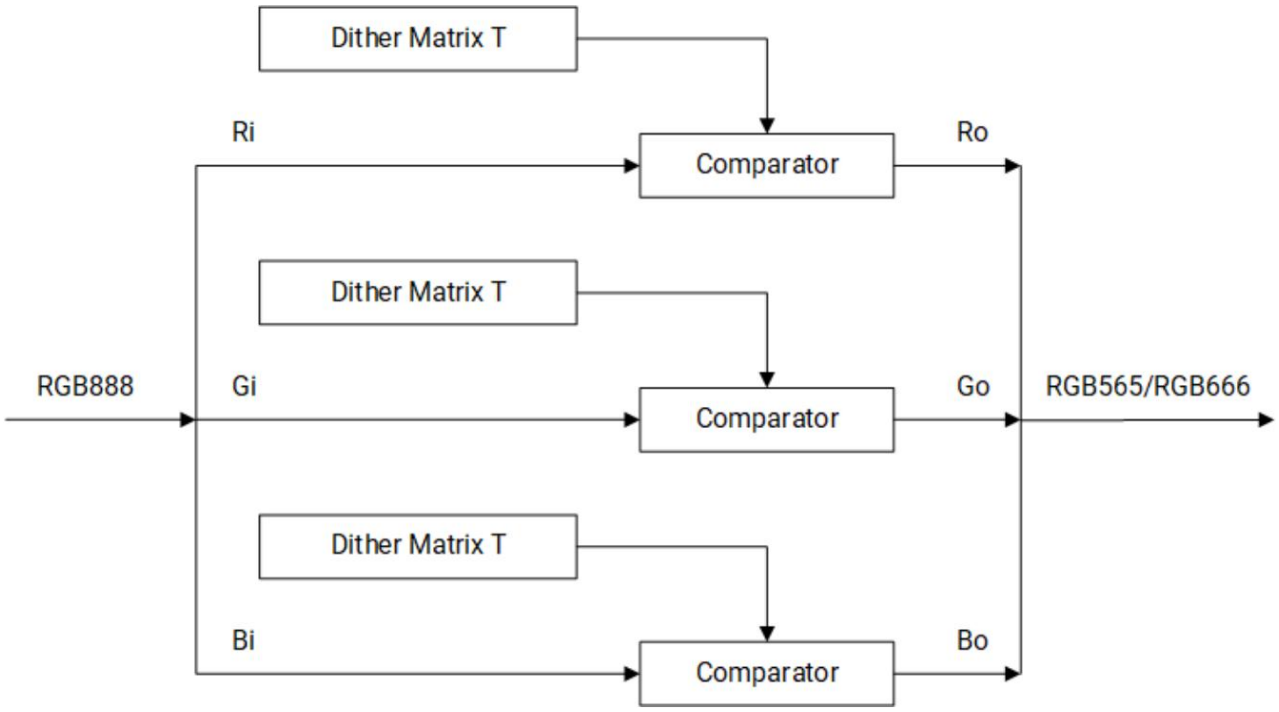
  Note. Alpha value is not supported for write-back in this case

In the code, the pixel value **L'** depends on the alpha value **a1** as per the following conditions:

```
if (a1 == 8'hFF)
L' = L0;
else
L' = L1 + L0 × a1/256;
```

12.3.5.4.2 Dither Function

The process of the Dither function is depicted below.



The Dither function can be enabled/disabled by software.

12.3.5.4.3 Fmark Function

The Fmark function controls the start of displaying output. In particular,

- If Fmark function is **enabled**, displaying output will wait until the Fmark signal is received
- If Fmark function is **disabled**, displaying output will start immediately after initiated by software

By software is possible to enable/disable Fmark function as well as control the polarity of the Fmark signal.

It is recommended to have a register to set how long displaying output is delayed after LCDC received the Fmark signal.

12.3.5.4.4 Background Color Display Function

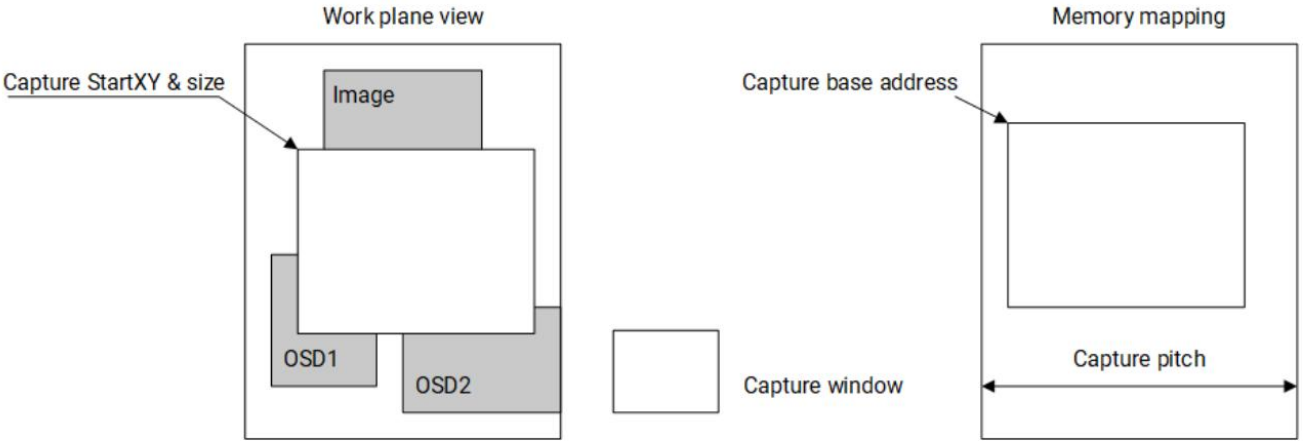
When no layer is enabled, a background color can be displayed without fetching data from DDR. The background color can be configured by software.

12.3.5.4.5 Image Capture Function

To apply the image capture function, the following parameters should be configured by software firstly:

- **startx** = X coordinate of the start point of the capture
- **starty** = Y coordinate of the start point of the capture
- **width** = Width (in pixels) of the capture from (X,Y) start point
- **height** = Height (in pixels) of the capture from (X, Y) start point
- **base_addr** = Memory start address for storing the capture
- **pitch** = Distance (in bytes) between the start of two consecutive rows of pixels stored in the memory, including any padding for alignment or hardware requirements

The process of the image capture function is depicted below.



12.3.6 Register Description

Note. Base address = **0xD420A000**

12.3.6.1 SHADOW_CTRL_REG REGISTER

Offset: 0x00C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	SHADOW_FLAG	RW	0x0	0x0: No change 0x1: Update parameters
30:1	Reserved	RO	0	Reserved for future use
0	DIS_SHADOW	RW	0x0	0x0: Shadow swap mode 0x1: Direct swap mode

12.3.6.2 LCD_TVG_START_ADDR0_REG REGISTER

Offset: 0x034				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TV Path Graphic Frame 0 Starting Address	RW	0x0	TV Path Graphic Frame 0 Starting Address in bytes

12.3.6.3 LCD_TVG_PITCH_REG REGISTER

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	TV Backlight Duty Cycle Control	RW	0x0	Controls the duty cycle percentage of the TV path backlight clock. - When the <TV Path Backlight Clock Divider> field (i.e., Bit[27:16]) is not 0x0, the backlight is controlled by the clock. - This field determines the duty cycle: 1. This Field/16 of the cycle is high 2. the rest is low
27:16	TV Path Backlight Clock Divider	RW	0x0	Configures the clock divider to generate the TV path backlight control clock. 0xFFF: Generates 32 kHz divided by 4096. 0x1: Generates 32 kHz divided by 2. If both this field and the <TV Backlight Duty Cycle Control> field (i.e., Bit[31:28]) are 0x0000 - The backlight clock function is disabled; - The backlight is controlled by the <Dumb LCD TV GPIO Control Pin> field in the Dumb LCD TV Control Register.
15:0	TV Path Graphic	RW	0x0	Specifies the TV Path Graphic Memory Pitch in bytes

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
	Memory Pitch			

12.3.6.4 LCD_TVG_OVSA_HPXL_VLN_REG REGISTER

Offset: 0x040				
Bits	**Field**	**Type**	**Reset**	**Description**
31	TV Path Graphic DMA Frame Selection Enable	RW	0x0	This field enables selection of a start frame.
30:28	Reserved	RO	0	Reserved for future use
27:16	TV Path Graphic Destination Starting Vertical Line on Screen	RW	0x0	Field Sum Constraint Formula: This Field Value+"Graphic Vertical Line Number after Zooming" ≤ "Screen Active Vertical Lines" Where: - "Graphic Vertical Line Number after Zooming" is from the Graphic Destination Size (after Zooming) Register. - "Screen Active Vertical Lines" is the value defined in the Total Screen Active Size Register
15:13	Reserved	RO	0	Reserved for future use
12	TV Path Graphic DMA Frame	RW	0x0	This field is only valid if the <TV Path Graphic DMA Frame Selection Enable> field (i.e. Bit[31])is enabled. 0x0: Select frame 0 0x1: Select frame 1
11:0	TV Path Graphic Destination Starting Horizontal Pixel on Screen	RW	0x0	Field Sum Constraint Formula: This Field Value+"Graphic Horizontal Pixel Number after Zooming" ≤ "Screen Horizontal Active Pixels" Where: - "Graphic Horizontal Pixel Number after Zooming" is from the Graphic Destination Size (after Zooming) Register.. - "Screen Horizontal Active Pixels" is the value defined in the Total Screen Active Size Register.

12.3.6.5 LCD_TVG_HPXL_VLN_REG REGISTER

Offset: 0x044				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	TV Path Graphic Source Vertical Line Number	RW	0x0	This field sets the source vertical size of the graphic object in memory before zooming.
15:12	Reserved	RO	0	Reserved for future use
11:0	TV Path Graphic Source Horizontal Pixel Number	RW	0x0	This field sets the source horizontal size of the graphic object in memory before zooming.

12.3.6.6 LCD_TVGZM_HPXL_VLN_REG REGISTER

Offset: 0x048				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	TV Path Graphic Destination Vertical Line Number after Zooming	RW	0x0	This field sets the vertical display size. - Zooming Down: If the <Graphic Vertical Line Number> field in the Graphic Source Size Register is greater than this field, the image is scaled down (made smaller). - Zooming Up: If the <Graphic Vertical Line Number> field in the Graphic Source Size Register is less than this field, the image is scaled up (made larger). - No Zooming: If the <Graphic Vertical Line Number> field in the Graphic Source Size Register is equal to this field, no zooming is performed.
15:12	Reserved	RO	0	Reserved for future use
11:0	TV Path Graphic Destination Horizontal Pixel Number after Zooming	RW	0x0	This field sets the horizontal display size. - Zooming Down: If the <Graphic Horizontal Pixel Number> in the Graphic Source Size Register is greater than this field, the image is scaled down (made smaller). - Zooming Up: If the <Graphic Horizontal Pixel Number> in the Graphic Source Size Register is less than this field, the image is scaled up (made larger). - No Zooming: If the <Graphic Horizontal Pixel Number> in the Graphic Source Size Register is equal to this field, no

Offset: 0x048				
Bits	**Field**	**Type**	**Reset**	**Description**
				zooming is performed.

12.3.6.7 LCD_TV_COLORKEY_Y_REG REGISTER

Note. Definition of symbols used:

- Y: Luminance value of a pixel in the YUV color model
- Y1 (<CFG_TV_CKEY_Y1>): Lower threshold for color keying
- Y2 (<CFG_TV_CKEY_Y2>): Upper threshold for color keying
- R: Red component in the RGB color model, used as a replacement if graphic color keying is enabled

Offset: 0x070				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	TV Path Color Key Y2	RW	0x0	Defines the maximum Y value (Y2) for color keying. If a pixel's Y component falls within Y1 to Y2, the system: - Applies an alpha value <CFG_TV_ALPHA_Y> - Replaces Y with <CFG_TV_CKEY_Y> - If graphic color keying is enabled, Y is replaced with R
23:16	TV Path Color Key Y1	RW	0x0	Defines the minimum Y value (Y1) for color keying. If a pixel's Y component falls within Y1 to Y2, the system: - Applies an alpha value <CFG_TV_ALPHA_Y> - Replaces Y with <CFG_TV_CKEY_Y> - If graphic color keying is enabled, Y is replaced with R
15:8	TV Path Color Key Y	RW	0x0	Specifies the replacement Y value (<CFG_TV_CKEY_Y>) used when the pixel's Y component is within Y1 to Y2. If graphic color keying is enabled, Y is replaced with R.
7:0	TV Path Color Alpha Y	RW	0x0	Defines the alpha transparency level (<CFG_TV_ALPHA_Y>) applied when a pixel's Y component is within Y1 to Y2. If graphic color keying is enabled, Y is replaced with R.

12.3.6.8 LCD_TV_COLORKEY_U_REG REGISTER

Note. Definition of symbols used:

- U: Chrominance blue component in the YUV color model
- U1 (<TV Path Color Key U1>): Lower threshold for color keying
- U2 (<TV Path Color Key U2>): Upper threshold for color keying
- G: Green component in the RGB color model, used as a replacement if graphic color keying is enabled

Offset: 0x074

Bits	Field	Type	Reset	Description
31:24	TV Path Color Key U2	RW	0x0	Defines the maximum U value (U2) for color keying. If a pixel's U component falls within U1 to U2, the system: - Applies an alpha value <Color Alpha U> - Replaces U with <TV Path Color Key U> - If graphic color keying is enabled, U is replaced with G
23:16	TV Path Color Key U1	RW	0x0	Defines the minimum U value (U1) for color keying. If a pixel's U component falls within U1 to U2, the system: - Applies an alpha value <Color Alpha U> - Replaces U with <TV Path Color Key U> - If graphic color keying is enabled, U is replaced with G
15:8	TV Path Color Key U	RW	0x0	Specifies the replacement U value (<TV Path Color Key U>) applied when the pixel's U component is within U1 to U2. If graphic color keying is enabled, U is replaced with G.
7:0	TV Path Color Alpha U	RW	0x0	Defines the alpha transparency level (<Color Alpha U>) used when a pixel's U component is within U1 to U2. If graphic color keying is enabled, U is replaced with G.

12.3.6.9 LCD_TV_COLORKEY_V_REG REGISTER

Note. Definition of symbols used:

- V: Chrominance red component in the YUV color model
- V1 (<TV Path Color Key V1>): Lower threshold for color keying
- V2 (<TV Path Color Key V2>): Upper threshold for color keying
- B: Blue component in the RGB color model, used as a replacement if graphic color keying is enabled

Offset: 0x078

Bits	Field	Type	Reset	Description
31:24	TV Path Color Key	RW	0x0	Defines the maximum V value (V2) for color keying. If a pixel's V component falls within V1 to V2, the system:

Offset: 0x078				
Bits	**Field**	**Type**	**Reset**	**Description**
	V2			- Applies an alpha value <Color Alpha V> - Replaces V with <TV Path Color Key V> - If graphic color keying is enabled, V is replaced with B
23:16	TV Path Color Key V1	RW	0x0	Defines the minimum V value (V1) for color keying. If a pixel's V component falls within V1 to V2, the system: - Applies an alpha value <Color Alpha V> - Replaces V with <TV Path Color Key V> - If graphic color keying is enabled, V is replaced with B.
15:8	TV Path Color Key V	RW	0x0	Specifies the replacement V value (<TV Path Color Key V>) applied when the pixel's V component is within V1 to V2. If graphic color keying is enabled, V is replaced with B.
7:0	TV Path Color Alpha V	RW	0x0	Defines the alpha transparency level (<Color Alpha V>) used when a pixel's V component is within V1 to V2. If graphic color keying is enabled, V is replaced with B.

12.3.6.10 LCD_TV_CTRL0_REG REGISTER

Offset: 0x80				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	Reserved	RO	0	Reserved for future use
29	TV Path Video Contrast/Saturation/Brightness/Hue Adjust Enable	RW	0x0	1: Enabled 0: Disabled
28	TV Path Palette Color Enable	RW	0x0	1: Enabled 0: Disabled This field enables the palette color SRAM table when the selected Memory Color Format is palette4bit or palette8bit. Palette mode can be selected via video or graphic DMA. **Note.** Only one palette table in the TV path.
27:20	Reserved	RO	0	Reserved for future use
19:16	TV Path Graphic DMA Memory Color	RW	0x0	Defines the graphic DMA memory color format. Options: 0x0: RGB565; 0x1: RGB1555; 0x2: RGB888; 0x3: RGB888 unpacked;

Offset: 0x80				
Bits	**Field**	**Type**	**Reset**	**Description**
	Format			0x4: RGBA888; 0x5: YUV422packed; 0x9: Palette color 4-bit; 0xA: Palette color 8-bit; 0xB: RGB888A; Others: Reserved.
15	Reserved	RO	0	Reserved for future use
14	TV Path Graphic Horizontal Smooth Enable	RW	0x0	1: Enabled 0: Disabled
13	TV Path Graphic DMA Test Mode Enable	RW	0x0	1: Enabled 0: Disabled
12	TV Path Graphic DMA Swap R and B	RW	0x0	1: Swap enabled 0: Swap disabled Swaps the red (R) and blue (B) channels in the RGB color model (e.g., RGB → BGR).
11	TV Path Graphic DMA Swap U and V	RW	0x0	1: Swap enabled 0: Disabled Swaps the U and V components in YUV color space (e.g., YUYV → YVYU).
10	TV Path Graphic DMA Swap Y and U/V	RW	0x0	1: Swap enabled 0: Disabled Swaps the Y component with U/V in YUV color space (e.g., UYVY → YUYV).
9	TV Path Graphic DMA YUV to RGB Color Space Conversion	RW	0x0	TV Path Graphic DMA YUV to RGB Color Space Conversion, 1 = Enabled; 0 = Disabled. There is only one CSC in the TV path, so either this field or the <TV Path Video DMA YUV to RGB Color Space Conversion> field can be enabled. Both cannot be enabled simultaneously. 1: Enabled 0: Disabled Converts YUV to RGB color space. **Note.** Either this field or the <TV Path Video DMA YUV to RGB Color Space Conversion> field can be enabled at a time.
8	TV Path Graphic	RW	0x0	1: Enable 0: Disabled

Offset: 0x80				
Bits	**Field**	**Type**	**Reset**	**Description**
	DMA Transfer Enable			
7:0	Reserved	RO	0	Reserved for future use.

12.3.6.11 LCD_TV_CTRL1_REG REGISTER

Offset: 0x084				
Bits	**Field**	**Type**	**Reset**	**Description**
31:27	Reserved	RO	0	Reserved for future use
26:24	TV Path Color Key Mode	RW	0x0	0x0: Disabled color key function 0x1: Video Y (or Graphic R) color key is enabled 0x2: Video U color key is enabled 0x3: Graphic RGB color key is enabled 0x4: Video V color key is enabled 0x5: Video YUV color key is enabled 0x6: Video Luma key is enabled 0x7: Graphic B color key is enabled
23	TV Path Configure Low Bits	RW	0x0	1: Low bits are the extension of the maximum bit when converting RGB565/1555/4-bit color into 24-bit RGB color. 0: Fill zeros into low bits when converting RGB565/1555/4-bit color into 24-bit RGB color.
22	Reserved	RO	0	Reserved for future use
21	TV Path Graphic DMA Color Key Enable	RW	0x0	1: Enabled 0: Disabled
20	TV Path Video DMA Color Key Enable	RW	0x0	1: Enabled 0: Disabled
19	Panel Path Graphic DMA Color Key Enable	RW	0x0	1: Enabled 0: Disabled
18	Panel Path Video DMA Color Key Enable	RW	0x0	1: Enabled 0: Disabled
17	Reserved	RO	0	Reserved for future use

Offset: 0x084				
Bits	**Field**	**Type**	**Reset**	**Description**
16	TV Path Configure Video/Graphic Path	RW	0x0	Selects the panel path graphic alpha for overlay. 0x0: Software configuration, 0x1: Pixel-based configuration.
15:8	TV Path Configure Alpha	RW	0x0	This field specifies the alpha value for blending graphics and video in the TV path when no color key alpha or alpha combined with pixel. - 0xFF: Selects all video, no graphics. - Other Values: Proportionally blends video and graphics.
7	Reserved	RO	0	Reserved for future use
6	Re-synchronize Panel Path and TV Path Timing	RW	0x0	This field resynchronizes both timing generation modules when Panel path and TV path are using the same clock. 1: Disable both together, no display 0: Enable both at the same time after setting to 1 and back to 0
5	LCD I/O Pads Panel Signals to TV	RW	0x0	Output on LCD I/O Signals [39:0]: Panel pad signals connect to TV output. - Bits [3:2]: The entire path is swapped, including MIPI DSI1, DSI2, and HDMI. - Bits [5:4]: The swap only occurs on 40 digital I/O signals.
4	LCD I/O Pads TV Signals to Panel	RW	0x0	On the LCD I/O signals [39:0], TV output signals connect to the Panel path. - Bits [3:2]: Swaps the entire signal path, including MIPI DSI1, DSI2, and HDMI. - Bits [5:4]: Swaps only 40 digital I/O signals.
3	Panel Path Occupies TV Interface	RW	0x0	By default, the TV interface is reserved for TV path data. However, this field allows Panel path data to use the TV interface when enabled. 1: Enable Panel path pass-through via TV interface. 0: Disabled (default). **Note.** This feature is only used for testing.
2	TV Path Occupies Digital Panel Interface	RW	0x0	By default, the digital panel interface is reserved for Panel path data. However, enabling this field allows TV path data to occupy the digital panel interface instead. 1: Enable TV path pass-through via the Panel interface. 0: Disabled (default). **Note.** This feature is only used for testing.

Offset: 0x084				
Bits	**Field**	**Type**	**Reset**	**Description**
1	Reserved	RO	0	Reserved for future use.
0	Enable Interlaced Mode on TV Interface	RW	0x0	Enables interlaced mode on the TV interface. 1: Interlaced mode enabled, 0: Disabled.

12.3.6.12 LCD_TV_CONTRAST_REG REGISTER

Offset: 0x088				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	TV Path Video Brightness Control	RW	0x0	Bits [15:8] are used for sign extension. Bits [7:0] are used for integer brightness control. All adjustments are performed before CSC, brightness change range is +/- 0 to 255. These bits are 2's complement code. - Bits [15:8]: Used for sign extension. - Bits [7:0]: Define the integer brightness level. Adjustments are applied before color space conversion (CSC). Brightness control range: ±0 to 255 (represented in two's complement format).
15:0	TV Path Video Contrast Control	RW	0x0	- Bit [15]: Sign bit. - Bit [14]: Integer part. - Bits [13:0]: Fractional contrast value. Contrast control is represented in two's complement format. Example contrast settings: - 0x4000 → Ratio 1.0 (no change) - 0x6000 → Ratio 1.5 (increase) - 0x2000 → Ratio 0.5 (decrease)

12.3.6.13 LCD_TV_SATURATION_REG REGISTER

Offset: 0x08C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	TV Path Configure Multiplier	RW	0x0	- Bit [15]: Sign bit. - Bits [14:13]: Integer part. - Bits [12:0]: Fractional multiplier value. Values are represented in two's complement format.
15:0	TV Path Configure Saturation	RW	0x0	- Bit [15]: Sign bit. - Bit [14]: Integer part. - Bits [13:0]: Fractional saturation value.

Offset: 0x08C				
Bits	**Field**	**Type**	**Reset**	**Description**
				Values are represented in two’s complement format.

12.3.6.14 LCD_TV_CBSH_HUE_REG REGISTER

Offset: 0x090				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	TV Path Video HUE Sine Correction	RW	0x0	Adjusts the HUE of the TV path video using Sine correction. - Bit [15]: Sign bit. - Bit [14]: Integer part. - Bits [13:0]: Fractional sine (delta phase). Values are represented in two’s complement format. Formula for Hue Correction: - Corrected U = U × cos + V × sin - Corrected V = V × cos - U × sin Example: - If CFG_SIN = 0 and CFG_COS = 0x4000, no correction is applied.
15:0	TV Path Video HUE Cosine Correction	RW	0x0	Adjusts the HUE of the TV path video using Cosine correction. - Bit [15]: Sign bit. - Bit [14]: Integer part. - Bits [13:0]: Fractional cosine (delta phase). Values are represented in two’s complement format. Formula for Hue Correction: - Corrected U = U × cos + V × sin - Corrected V = V × cos - U × sin Example: - If CFG_SIN = 0x2000 and CFG_COS = 0x376D, a 30-degree HUE correction is applied.

12.3.6.15 LCD_DMA_START_ADDR_Y0_REG REGISTER

Offset: 0x0C0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Panel Path Video Frame 0 Y Starting Address	RW	0x0	Panel Path Video Frame 0 Y Starting Address in bytes

12.3.6.16 LCD_DMA_START_ADDR_U0_REG REGISTER

Offset: 0x0C4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Panel Path Video Frame 0 U Starting Address in bytes	RW	0x0	Panel Path Video Frame 0 U Starting Address in bytes

12.3.6.17 LCD_DMA_START_ADDR_V0_REG REGISTER

Offset: 0x0C8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Panel Path Video Frame 0 V Starting Address	RW	0x0	Panel Path Video Frame 0 V Starting Address in bytes

12.3.6.18 LCD_DMA_PITCH_YC_REG REGISTER

Offset: 0x0E0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15:0	Panel Path Video Y Pitch	RW	0x0	Panel Path Video Y Pitch in bytes

12.3.6.19 LCD_DMA_PITCH_UV_REG REGISTER

Offset: 0x0E4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Panel Path Video V Pitch	RW	0x0	Panel Path Video V Pitch in bytes
15:0	Panel Path Video U Pitch	RW	0x0	Panel Path Video U Pitch in bytes

12.3.6.20 LCD_DMA_OVSA_HPXL_VLN_REG REGISTER

Offset: 0x0E8				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Panel Path Video DMA	RW	0x0	This field is used to enable Panel Path video DMA to select either frame0 or frame1.

Offset: 0x0E8				
Bits	**Field**	**Type**	**Reset**	**Description**
	Frame Selection Enable			The frame number is controlled by bits [15:13]. 1: Enable to select; 0: Disable the function
30:28	Reserved	RO	0	Reserved for future use
27:16	Panel Path Video Starting Vertical Line on Screen	RW	0x0	Defines the starting vertical line for Panel Path Video on the screen. Valid Range: 0 to 0xFFF. The sum of this field and <Video Vertical Line Number after Zooming> (from the **Video Destination Size (After Zooming) Register**) must not exceed the active display lines set in <Screen Active Vertical Lines> (from the **Total Screen Active Size Register**).
15	Panel Path Video Y DMA Frame	RW	0x0	0: frame0 1: frame1 **Note.** This field is only used if the <Panel Path Video DMA Frame Selection Enable> field is disabled.
14	Panel Path video U frame	RW	0x0	0: frame0 1: frame1 **Note.** This field is only used if the <Panel Path Video DMA Frame Selection Enable> field is enabled.
13	Panel Path Video V DMA Frame	RW	0x0	0: frame0 1: frame1 **Note**. This field is only used if the <Panel Path Video DMA Frame Selection Enable> field is enabled.
12	Panel Path Smart Panel Command DMA Frame	RW	0x0	0: frame0 1: frame1 **Note.** This field is only used if the <Panel Path Video DMA Frame Selection Enable> field is enabled.
11:0	Panel Path Video Starting Horizontal Pixel on Screen	RW	0x0	Specifies the starting horizontal pixel position for Panel Path Video on the screen. Valid Range: 0x0 to 0xFFF The sum of this field and the <Video Horizontal Pixel Number after Zooming> field (from the Video Destination Size [After Zooming] Register) must not exceed the active display horizontal width defined in the <Screen Horizontal Active Pixels> field (from the Total Screen Active Size Register).

12.3.6.21 LCD_DMA_HPXL_VLN_REG REGISTER

Offset: 0x0EC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	Panel Path Video Vertical Line Number	RW	0x0	This field is used for the source vertical size of the video object in memory before zooming.
15:12	Reserved	RO	0	Reserved for future use
11:0	Panel Path Video Horizontal Pixel Number	RW	0x0	This field is used for the source horizontal size of the video object in memory before zooming.

12.3.6.22 LCD_DMAZM_HPXL_VLN_REG REGISTER

Offset: 0x0F0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	Panel Path Video Vertical Line Destination Number after Zooming	RW	0x0	Sets the vertical display size after zooming. - Zoom Down (shrink image): If the source vertical lines > this field - Zoom Up (enlarge image): If the source vertical lines < this field - No Zoom: If the source vertical lines = this field Where the source vertical lines are from the <Video Vertical Line Number> field in the Video Source Size Register
15:12	Reserved	RO	0	Reserved for future use
11:0	Panel Path Video Horizontal Pixel Destination Number after Zooming	RW	0x0	Sets the horizontal display size after zooming. - Zoom Down (shrink image): If the source horizontal pixels > this field - Zoom Up (enlarge image): If the source horizontal pixels < this field - No Zoom: If the source horizontal pixels = this field Where the source horizontal pixels are from the <Video Horizontal Pixel Number> field in Video Source Size Register

12.3.6.23 LCD_GRA_START_ADDR0_REG REGISTER

Offset: 0x0F4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Panel Path Graphic Frame 0 Starting Address	RW	0x0	Panel Path Graphic Frame 0 Starting Address in bytes.

12.3.6.24 LCD_GRA_PITCH_REG REGISTER

Offset: 0x0FC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Panel Backlight Duty Cycle Control	RW	0x0	When the <Dumb Panel Backlight Clock Divider> field is not 0x0, the Dumb Panel backlight is controlled by clock, and this field controls the clock duty cycle percentage. This field/16 of the cycle is high, others is low. Controls the duty cycle percentage for the Dumb Panel backlight when the <Dumb Panel Backlight Clock Divider> field is not 0x0. - The duty cycle is Field/16 of the cycle (high), with the rest low.
27:16	Panel Backlight Clock Divider	RW	0x0	Configures the clock divider to generate the Dumb Panel backlight control clock. - 0xFFF: Generates 32 kHz divided by 4096. - 0x1: Generates 32 kHz divided by 2. If both this field and the <Duty Cycle Control> field> are 0x0000, the backlight clock function is disabled, and the backlight is controlled by the <Dumb LCD Panel GPIO Control Pin> field in the Dumb LCD Panel Control Register.
15:0	Panel Path Graphic Memory Pitch	RW	0x0	Panel Path Graphic Memory Pitch in bytes

12.3.6.25 LCD_GRA_OVSA_HPXL_VLN_REG REGISTER

Offset: 0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Panel Path Graphic DMA Select Frame 0 or 1	RW	0x0	1: Enabled 0: Disabled
30:28	Reserved	RO	0	Reserved for future use

Offset: 0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
27:16	Panel Path Graphic Destination Starting Vertical Line on Screen	RW	0x0	The sum of this field and the <Graphic Vertical Line Number after Zooming> field (from the Graphic Destination Size (after Zooming) Register) must not exceed the Screen Active Vertical Lines field (from the Total Screen Active Size Register).
15:13	Reserved	RO	0	Reserved for future use
12	Panel Path Graphic DMA Frame	RW	0x0	Specifies the frame used for DMA transfer. 1: Frame1 0: Frame0 Note: This field is only valid if the <Panel Path Graphic DMA Select Frame 0 or 1> field is enabled
11:0	Panel Path Graphic Destination Starting Horizontal Pixel on Screen	RW	0x0	The sum of this field and the <Graphic Horizontal Pixel Number after Zooming> field (from the Graphic Destination Size (after Zooming) Register) must not exceed the <Screen Horizontal Active Pixels> field (from the Total Screen Active Size Register).

12.3.6.26 LCD_GRA_HPXL_VLN_REG REGISTER

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	Panel Path Graphic Source Vertical Line Number	RW	0x0	This field sets the source vertical size of the graphic object in memory before zooming.
15:12	Reserved	RO	0	Reserved for future use
11:0	Panel Path Graphic Source Horizontal Pixel Number	RW	0x0	This field sets the source horizontal size of the graphic object in memory before zooming.

12.3.6.27 LCD_GRAZM_HPXL_VLN_REG REGISTER

Offset: 0x108				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use

Offset: 0x108				
Bits	**Field**	**Type**	**Reset**	**Description**
27:16	Panel Path Graphic Destination Vertical Line Number after Zooming	RW	0x0	Sets the vertical display size after zooming. - Zoom Out (smaller image): If "the graphic vertical lines" > this field - Zoom In (larger image): If "the graphic vertical lines" < this field - No Zoom: If "the graphic vertical lines" = this field Where "the graphic vertical lines" is the <Graphic Vertical Line Number> field from the Graphic Source Size Register
15:12	Reserved	RO	0	Reserved for future use
11:0	Panel Path Graphic Destination Horizontal Pixel Number after Zooming	RW	0x0	Sets the horizontal display size after zooming. - Zoom Out (smaller image): If "the pixel number" > this field - Zoom In (larger image): If "the pixel number" < this field - No Zoom: If "the pixel number" = this field Where "the pixel number" is the <Graphic Horizontal Pixel Number> field from the Graphic Source Size Register.

12.3.6.28 LCD_PN_V_H_ACTIVE_REG REGISTER

Offset: 0x118				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	Panel Path Screen Active Vertical Lines	RW	0x0	This field sets the active vertical screen display size for both Dumb Panel and Smart Panel.
15:12	Reserved	RO	0	Reserved for future use
11:0	Panel Path Screen Horizontal Active Pixels	RW	0x0	This field sets the active horizontal screen display width for both Dumb Panel and Smart Panel.

12.3.6.29 LCD_PN_BLANKCOLOR_REG REGISTER

Offset: 0x124				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	Panel Path Background Color	RW	0x0	Defines the background color displayed when no objects overlay it or when no valid pixels exist within the active area. - Bits [7:0]: Red component. - Bits [15:8]: Green component. - Bits [23:16]: Blue component.

12.3.6.30 LCD_PN_ALPHA_COLOR1_REG REGISTER

Offset: 0x128				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	Panel Path Hardware Cursor Color 1	RW	0x0	- Bits [7:0]: Red component. - Bits [15:8]: Green component. - Bits [23:16]: Blue component.

12.3.6.31 LCD_PN_ALPHA_COLOR2_REG REGISTER

Offset: 0x12C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	Panel Path Hardware Cursor Color 2	RW	0x0	- Bits [7:0]: Red component. - Bits [15:8]: Green component. - Bits [23:16]: Blue component.

12.3.6.32 LCD_PN_COLORKEY_Y_REG REGISTER

Note. Definition of symbols used:

- Y: Luminance value of a pixel in the YUV color model
- Y1 (<CFG_PN_CKEY_Y1>): Lower threshold for color keying
- Y2 (<CFG_PN_CKEY_Y2>): Upper threshold for color keying
- R: Red component in the RGB color model, used as a replacement if graphic color keying is enabled

Offset: 0x130				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Panel Path Color Key Y2	RW	0x0	Defines the maximum Y value (Y2) for color keying. If a pixel's Y component falls within Y1 to Y2, the system: - Applies an alpha value <CFG_PN_ALPHA_Y> - Replaces Y with <CFG_PN_CKEY_Y> - If graphic color keying is enabled, Y is replaced with R
23:16	Panel Path Color Key Y1	RW	0x0	Defines the minimum Y value (Y1) for color keying. If a pixel's Y component falls within Y1 to Y2, the system: - Applies an alpha value <CFG_PN_ALPHA_Y> - Replaces Y with <CFG_PN_CKEY_Y> - If graphic color keying is enabled, Y is replaced with R
15:8	Panel Path Color Key Y	RW	0x0	Specifies the replacement Y value (<CFG_PN_CKEY_Y>) used when the pixel's Y component is within Y1 to Y2. If graphic color keying is enabled, Y is replaced with R.
7:0	Panel Path Color Alpha Y	RW	0x0	Defines the alpha transparency level (<CFG_PN_ALPHA_Y>) applied when a pixel's Y component is within Y1 to Y2. If graphic color keying is enabled, Y is replaced with R.

12.3.6.33 LCD_PN_COLORKEY_U_REG REGISTER

Note. Definition of symbols used:

- U: Chrominance blue component in the YUV color model
- U1 (<Panel Path Color Key U1>): Lower threshold for color keying
- U2 (<Panel Path Color Key U2>): Upper threshold for color keying
- G: Green component in the RGB color model, used as a replacement if graphic color keying is enabled

Offset: 0x134				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Panel Path Color Key U2	RW	0x0	Defines the maximum U value (U2) for color keying. If a pixel's U component falls within U1 to U2, the system: - Applies an alpha value <Color Alpha U> - Replaces U with <Panel Path Color Key U> - If graphic color keying is enabled, U is replaced with G
23:16	Panel Path Color Key U1	RW	0x0	Defines the minimum U value (U1) for color keying. If a pixel's U component falls within U1 to U2, the system: - Applies an alpha value <Color Alpha U> - Replaces U with <Panel Path Color Key U> - If graphic color keying is enabled, U is replaced with G

Offset: 0x134				
Bits	**Field**	**Type**	**Reset**	**Description**
15:8	Panel Path Color Key U	RW	0x0	Specifies the replacement U value (<Panel Path Color Key U>) applied when the pixel's U component is within U1 to U2. If graphic color keying is enabled, U is replaced with G.
7:0	Panel Path Color Alpha U	RW	0x0	Defines the alpha transparency level (<Color Alpha U>) used when a pixel's U component is within U1 to U2. If graphic color keying is enabled, U is replaced with G.

12.3.6.34 LCD_PN_COLORKEY_V_REG REGISTER

Note. Definition of symbols used:

- V: Chrominance red component in the YUV color model
- V1 (<Panel Path Color Key V1>): Lower threshold for color keying
- V2 (<Panel Path Color Key V2>): Upper threshold for color keying
- B: Blue component in the RGB color model, used as a replacement if graphic color keying is enabled

Offset: 0x138				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Panel Path Color Key V2	RW	0x0	Defines the maximum V value (V2) for color keying. If a pixel's V component falls within V1 to V2, the system: - Applies an alpha value <Color Alpha V> - Replaces V with <Panel Path Color Key V> - If graphic color keying is enabled, V is replaced with B
23:16	Panel Path Color Key V1	RW	0x0	Defines the minimum V value (V1) for color keying. If a pixel's V component falls within V1 to V2, the system: - Applies an alpha value <Color Alpha V> - Replaces V with <Panel Path Color Key V> - If graphic color keying is enabled, V is replaced with B.
15:8	Panel Path Color Key V	RW	0x0	Specifies the replacement V value (<Panel Path Color Key V>) applied when the pixel's V component is within V1 to V2. If graphic color keying is enabled, V is replaced with B.
7:0	Panel Path Color Alpha V	RW	0x0	Defines the alpha transparency level (<Color Alpha V>) used when a pixel's V component is within V1 to V2. If graphic color keying is enabled, V is replaced with B.

12.3.6.35 LCD_PN_SEPXLCNT_REG REGISTER

Offset: 0x13C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Debug Read Index	RW	0x0	Specifies the read port index for the Panel Slave Path Status and Debug Register. 0: Normal function
27:16	Panel Path VSYNC Falling Edge Pixel Position of the Line	RW	0x0	This field is used for the Panel Path horizontal pixel count from the first valid pixel to the VSYNC pulse falling edge point. VSYNC pulse is configured by both line number and pixel number.
15:12	Reserved	RO	0	Reserved for future use
11:0	Panel Path VSYNC Rising Edge Pixel Position of the Line	RW	0x0	This field is used for the Panel Path horizontal pixel count from the first valid pixel to the VSYNC pulse rising edge point. VSYNC pulse is configured by both line number and pixel number.

12.3.6.36 LCD_SPI_RXDATA_REG REGISTER

Offset: 0x140				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SPI Read Data	RO	0x0	SPI Read Data

12.3.6.37 LCD_ISA_RXDATA_REG REGISTER

Offset: 0x144				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	16-bit or 8-bit Smart Panel Read Data	RO	0x0	If the <Configure Command Format> field in the Smart Panel 8-bit Bus Control Register is set to 16-bit format: - Bits [7:0] contain the last read data - Bits [15:8] contain the second last read data

12.3.6.38 LCD_READ_IOPAD_REG REGISTER

Offset: 0x148				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:0	I/O Pad Read Value	RO	0x0	This field contains the read value of the Digital Panel interface I/O pad [27:0]. This is useful for I/O pad checking and GPIO read data.

12.3.6.39 LCD_DMAVLD_YC_REG REGISTER

Offset: 0x14C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Panel Video Path Y Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
30	Panel Video Path U Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
29	Panel Video Path V Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
28	Panel Graphic Path Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
27	TV Video Path Y Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
26	TV Video Path U Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
25	TV Video Path V Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
24	TV Graphic Path Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
23	Panel Path Smart Panel Command Starting Address Update Flag	RO	0x0	0: Update enabled 1: Update disabled
22	tvd_sa_cflag	RW	0x0	
21:16	Reserved	RO	0	Reserved for future use
15:0	Panel Path Video Actual Y Line Length in Memory	RO	0x0	Panel video Path Y valid length in bytes, generated from color format and pixel number of each line

12.3.6.40 LCD_DMAVLD_UV_REG REGISTER

Offset: 0x150				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	Reserved	RO	0	Reserved for future use
19:10	Panel Path Video Actual V Line Length in Memory	RO	0x0	Panel video Path U valid length in bytes, generated from color format and pixel number of each line.
9:0	Panel Path Video Actual U Line Length in Memory	RO	0x0	Panel video Path V valid length in bytes, generated from color format and pixel number of each line.

12.3.6.41 LCD_TVGGRAVLD_HLEN_REG REGISTER

Offset: 0x154				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	TV Path Graphic Actual Line Length in Memory	RO	0x0	TV graphic Path Y valid length in bytes, generated from color format and pixel number of each line.
15:0	Panel Path Graphic Actual Line Length in Memory	RO	0x0	Panel graphic Path Y valid length in bytes, generated from color format and pixel number of each line.

12.3.6.42 LCD_PN_GAMMA_RDDAT_REG REGISTER

Offset: 0x15C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0	Reserved for future use
7:0	Panel Path Gamma Table SRAM Read Data	RO	0x0	Panel Path Gamma Table SRAM Read Data

12.3.6.43 LCD_PN_PALETTE_RDDAT_REG REGISTER

Offset: 0x160				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	Panel Path Palette Table SRAM Read Data	RO	0x0	Panel Path Palette Table SRAM Read Data

12.3.6.44 LCD_SLV_DBG_REG REGISTER

Offset: 0x164				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	Reserved	RO	0	Reserved for future use
19:12	AHB Slave Path FIFO Space Count	RO	0x20	Indicates the available space count (in bytes) in the AHB Slave Path FIFO for the Smart Panel. This value shows the total bytes of data can be written to the AHB Slave Path Data Port Register. It's recommended to check this value before writing to ensure space availability. The maximum value is 128 bytes.
11:4	AHB Slave Path FIFO Data Count	RO	0x0	Indicates the data count (in bytes) in the AHB Slave Path FIFO for the Smart Panel. The sum of this field and the <AHB Slave Path FIFO Space Count> should equal 0x80, which is the FIFO size.
3:0	Reserved	RO	0	Reserved for future use

12.3.6.45 LCD_TV_GAMMA_RDDAT_REG REGISTER

Offset: 0x174				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0	Reserved for future use
7:0	TV Path Gamma Table SRAM Read Data	RO	0x0	TV Path Gamma Table SRAM Read Data

12.3.6.46 LCD_TV_PALETTE_RDDAT_REG REGISTER

Offset: 0x178				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	TV Path Palette Table SRAM Read Data	RO	0x0	TV Path Palette Table SRAM Read Data

12.3.6.47 LCD_FRAME_CNT_REG REGISTER

Offset: 0x17C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15:14	Current IRE Frame Number for TV Path	RO	0x0	Current IRE Frame Number for TV Path
13:12	Current CCIC Frame Number for TV Path	RO	0x0	Current CCIC Frame Number for TV Path
11:10	Current IRE Frame Number for Panel Path	RO	0x0	Current IRE Frame Number for Panel Path
9:8	Current CCIC Frame Number for Panel Path	RO	0x0	Current CCIC Frame Number for Panel Path
7:6	TV Current Graphic Frame Number	RO	0x0	TV Path Current Graphic Frame Number
5:4	TV Current Video Frame Number	RO	0x0	TV Path Current Video Frame Number
3:2	Panel Current Graphic Frame Number	RO	0x0	Panel Path Current Graphic Frame Number
1:0	Panel Current Video Frame Number	RO	0x0	Panel Path Current Video Frame Number

12.3.6.48 LCD_SPI_CTRL_REG REGISTER

Offset: 0x180				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Configure SPI dcx	RW	0x0	Configures the dcx bit for SPI 4-wire mode, either in command or data mode.
30:24	Configure SPI Clock Divider	RW	0x0	Defines the SPI clock divider with the valid values ranging from 0xFF to 0x02, used to generate the SPI clock from the panel path pixel clock.
23:16	Configure SPI Receive	RW	0x0	0x1F: Read/receive 32 bits 0x01: Read/receive 2 bits 0x00: Receive disabled - If the <Configure SPI Transmit> field is set to 0x00, only receive operations are allowed. The maximum length per trigger (write 0x01 to the <Start SPI Transfer> field) is 32 bits, but unlimited read length can be achieved by triggering repeatedly when the <Configure Continuous Transfer> field is set to 0x1. - If the <Configure SPI Transmit> field is not

Offset: 0x180				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x00, it will first transmit serial bits, then receive serial bits.
15:8	Configure SPI Transmit	RW	0x0	0x1F = Write/transmit 32 bits 0x01 = Write/transmit 2 bits 0x00 = Do not transfer - If the <Configure SPI Receive> field is set to 0x00, only transmission occurs. The maximum write length is 32 bits per trigger, but unlimited write length is possible when triggering repeatedly when the <Configure Continuous Transfer> field is set to 0x1. - If the <Configure SPI Receive> field is not 0x00, it will first transmit serial bits, and then receive incoming bits.
7	Configure Clock Inverse	RW	0x0	0: SPI clock rising edge samples the data, falling edge sends out data. 1: SPI clock falling edge samples the data, rising edge sends out data.
6	Configure continuous Transfer	RW	0x0	When set to 1, the SPI chip select is kept low until this bit is cleared, allowing continuous shifting of bits (in or out).
5	Configure Receive order	RW	0x0	0: Receive from higher bit to bit 0. 1: Receive from bit 0 to higher bit.
4	Configure Transmit order	RW	0x0	0: Transfer from higher bit to bit 0. 1: Transfer from bit 0 to higher bit.
3	Enable SPI	RW	0x0	0: Disabled 1: Enabled
2	Configure SPI Port	RW	0x0	0: SPI port 0 1: SPI port 1
1	Configure 3-/4-Wire SPI	RW	0x0	1: 3-wire SPI (SPI_DIN used for both transmit and receive). 0: 4-wire SPI (SPI_DIN for data from the product to the SPI panel, SPI_DOUT for data from the SPI panel to the product).
0	Start SPI Transfer	RW	0x0	1: Start the transfer (requires the <Enable SPI> field is set to be 1). 0: Transfer not started.

12.3.6.49 LCD_SPI_TXDATA_REG REGISTER

Offset: 0x184				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SPI Transfer Data	RW	0x0	SPI Transfer Data Up to 32 bits are shifted out by each trigger. It is configurable from high to low or low to high and to shift any length. 0 = MSb to LSb; 1 = LSb to MSb. Defines the data to be shifted out during SPI transfer. The transfer length is configurable up to 32 bits per trigger. The shifting direction can be set to either high to low (MSb to LSb) or low to high (LSb to MSb). 0: Shifts from MSb to LSb. 1: Shifts from LSb to MSb.

12.3.6.50 LCD_SMPN_CTRL_REG REGISTER

Offset: 0x188				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Configure ISA Receive Low	RW	0x0	This field programs the bus read active low time. 0xF: 6 clock cycles 0x0: 1 clock cycle.
27:24	Configure ISA Receive High	RW	0x0	This field programs the bus read hold time. 0xF: 16 clock cycles 0x0: 1 clock cycle
23:20	Configure ISA Transmit Low	RW	0x0	This field programs the bus write active low time. 0xF: 16 clock cycles 0x0: 1 clock cycle
19:16	Configure ISA Transmit High	RW	0x0	This field programs the bus read hold time. 0xF: 16 clock cycles 0x0: 1 clock cycle
15:14	Select Smart Panel VSYNC Trigger	RW	0x0	This field is used when the <Panel Path VSYNC Input Trigger Modes> field (in the Panel Path DMA Control 1 Register) is set to 0x0 or 0x1. 0x0: Parallel bus I/O pad input as Smart panel DMA VSYNC trigger signal. 0x1: MIPI DSI1 as Smart panel DMA VSYNC trigger signal. 0x2: MIPI DSI2 as Smart panel DMA VSYNC trigger signal.

Offset: 0x188				
Bits	**Field**	**Type**	**Reset**	**Description**
				All other values: Reserved
13	Configure ISA iordy Mask	RW	0x0	0x0: The I/O ready signal is always active high on the Smart Panel parallel bus. 0x1: The Smart Panel parallel bus is held when the I/O ready signal is low. The I/O ready signal goes high when the bus is ready. - Note: Set this bit to 1 when using the SPI interface to connect to a display device.
12	Configure Slave Only Mode	RW	0x0	- This field allows the Smart Panel to operate without DMA mode. - When both the Panel Path (HDMI) and TV Path (MIPI DSI) are enabled, no extra FIFO is available for Smart Panel DMA mode. - In slave-only mode, up to three panels can be supported simultaneously. - When set high, the Smart Panel operates in slave mode only.
11:8	Configure Pixel Format	RW	0x0	When in DMA transfer, this field defines the pixel format. 0x0: RGB888, 3 cycles per pixel; 0x1: RGB666, 3 cycles per pixel; 0x2: RGB565, 2 cycles per pixel; 0x3: RGB888, 1 cycle per pixel; 0x4: RGB666, 1 cycle per pixel; 0x5: RGB565 1 cycle per pixel; 0x6: RGB666_GC, 3 cycles per pixel.
7	Configure Command Format	RW	0x0	1: 32-bit command format, used for up to 24-bit bus width (I/O pins limited to 18-bit). 0: 16-bit command format, used for 8-bit bus width.
6	Write Byte Order	RW	0x0	Configures the byte order for writing. 0: Write 8-bit bus from low byte to high byte. 1: Write 8-bit bus from high byte to low byte. This field is used when writing one pixel across 2 or 3 byte writes.
5	Smart Panel Parallel Bus Chip Select	RW	0x0	Selects the chip select line for the Smart Panel parallel bus. 0: CSB[0] 1: CSB[1]
4	AHB Slave Path Enable	RW	0x0	0: Disabled 1: Enabled
3	Smart Panel Reset	RW	0x0	0: Reset pin is high 1: Reset pin is low

Offset: 0x188				
Bits	**Field**	**Type**	**Reset**	**Description**
2	Configure 8086/6800	RW	0x0	0: 8-bit bus read/write conforming to 8086 series 1: 8-bit bus read/write conforming to 6800 series
1	Configure Transfer	RW	0x0	1: Force chip select to low until this bit is set to 0. This field is useful when a Smart Panel requires the chip select to remain low during read and write.
0	Smart Panel Enable	RW	0x0	1: Enabled 0: Disabled When both this field and the <AHB Slave Path Enable> field are high, writing a command word to the AHB Slave Path Data Port Register initiates Smart Panel parallel bus cycles. - If the <Configure Command Format> field = 1 (High): 1. Commands are in 16-bit format. 2. Two 16-bit commands are packed into a single 32-bit word (e.g., Command 0 in bits 0–15, Command 1 in bits 16–31). - If the <Configure Command Format> field = 0 (Low): 1. Commands are in 32-bit format. 2. Only one 32-bit command fits per 32-bit word.

12.3.6.51 LCD_SLV_PORT_REG REGISTER

Offset: 0x18C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Command 1 A0	WO	0x0	Command 1 A0
30:26	Reserved	RO	0	Reserved for future use
25	Command 1 Read	WO	0x0	Command 1 Read
24	Command 1 Write	WO	0x0	Command 1 Write
23:16	Command 1 Data	WO	0x0	- If the <Configure Command Format> field is set to 16-bit format, this field holds the 8-bit command 1 data. - If the <Configure Command Format> field is set to 32-bit format, bits [23:0] represent Smart Panel data[23:0].
15	Command 0 A0	WO	0x0	- If the <Configure Command Format> field is set to 16-bit format, this field holds the command 0 A0. - If the <Configure Command Format> field is set to 32-bit format, bits [23:0] represent Smart Panel data[23:0].
14:10	32-bit Command Format Data	WO	0x0	- If the <Configure Command Format> field is set to 16-bit format, this field is reserved. - If the <Configure Command Format> field is set to 32-bit

Offset: 0x18C				
Bits	**Field**	**Type**	**Reset**	**Description**
				format, bits [23:0] represent Smart Panel data[23:0].
9	Command 0 Read	WO	0x0	- If the <Configure Command Format> field is set to 16-bit format, this field holds the command 0 read. - If the <Configure Command Format> field is set to 32-bit format, bits [23:0] represent Smart Panel data[23:0].
8	Command 0 Write	WO	0x0	- If the <Configure Command Format> field is set to 16-bit format, this field holds the command 0 write. - If the <Configure Command Format> field is set to 32-bit format, bits [23:0] represent Smart Panel data[23:0].
7:0	Command 0 Data	WO	0x0	- If the <Configure Command Format> field is set to 16-bit format, this field holds the command 0 data. - If the <Configure Command Format> field is set to 32-bit format, bits [23:0] represent Smart Panel data[23:0].

12.3.6.52 LCD_PN_CTRL0_REG REGISTER

Offset: 0x190				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Reserved	RO	0	Reserved for future use
30	Panel Path Gamma Correction Enable	RW	0x0	1: Enabled 0: Disabled
29	Panel Path Video Contrast/Saturation/Brightness/ Hue Adjust Enable	RW	0x0	1: Enabled 0: Disabled
28	Panel Path Palette Color Enable	RW	0x0	This field enables the palette color SRAM table. 1: Enabled 0: Disabled When the color format selected in the <Video Memory Color Format> field is palette4bit or palette8bit, this field should be 1. Either video or graphic DMA can select palette mode. There is only one palette table in the Panel Path. - This field must be set to 1 when the color format selected in the <Video Memory Color Format> field is palette4bit or palette8bit. - Either video or graphic DMA can use palette mode.

Offset: 0x190				
Bits	**Field**	**Type**	**Reset**	**Description**
				- There is only one palette table in the Panel Path, shared between video and graphic DMA.
27	AXI Bus Arbiter Fast Mode Enable	RW	0x0	1: DMA AXI bus arbiter allows multiple burst requests. 0: Arbiter switches after one request is done. For faster read, enable this bit in normal case.
26	Reserved	RO	0	Reserved for future use
25	Panel Path CS Low Delay Enable	RW	0x0	1: Enabled 0: Disabled
24	Panel Path Force Blank-Color Enable	RW	0x0	1: Enabled 0: Disabled
23:20	Panel Path Video Memory Color Format	RW	0x0	0x0: RGB565 0x1: RGB1555 0x2: RGB888 packed 0x3: RGB888 unpacked 0x4: RGBA888 0x5: YUV422 packed 0x6: YUV422 planar 0x7: YUV420 planar 0x8: Smart Panel command 0x9: Palette color 4-bit per pixel 0xA: Palette color 8-bit per pixel 0xB: RGB888A All other values: Reserved
19:16	Panel Path Graphic Memory Color Format	RW	0x0	0x0: RGB565 0x1: RGB1555 0x2: RGB888 packed 0x3: RGB888 unpacked 0x4: RGBA888 0x5: YUV422 packed 0x6 to 0x8 = Reserved 0x9: Palette color 4-bit per pixel 0xA: Palette color 8-bit per pixel 0xb: RGB888A All other values: Reserved
15	Reserved	RO	0	Reserved for future use
14	Panel Path Graphic Horizontal Smooth Enable	RW	0x0	1: Enabled 0: Disabled
13	Panel Path	RW	0x0	1: Enabled

Offset: 0x190				
Bits	**Field**	**Type**	**Reset**	**Description**
	Graphic DMA Test Mode Enable			0: Disabled
12	Panel Path graphic DMA Swap R and B	RW	0x0	Swap R and B (e.g., RGB to BGR). 1: Swap enabled 0: Swap disabled
11	Panel Path Graphic DMA Swap U and V	RW	0x0	Swap U and V (e.g., YUYV to YVYU). 1: Swap enabled 0: Swap disabled
10	Panel Path Graphic DMA Swap Y and U/V	RW	0x0	Swap Y and U/V (e.g., UYVY to YUYV). 1: Swap enabled 0: Swap disabled
9	Panel Path Graphic YUV to RGB Color Space Conversion	RW	0x0	1: Enabled 0: Disabled Either this or the <Panel Path Video YUV to RGB Color Space Conversion> field can be enabled, but both cannot be enabled simultaneously.
8	Panel Path Graphic Transfer Enable	RW	0x0	1: Enabled 0: Disabled
7	Reserved	RO	0	Reserved for future use
6	Panel Path Video Horizontal Smooth Enable	RW	0x0	1: Enabled 0: Disabled
5	Panel Path Video Path Test Mode Enable	RW	0x0	1: Enabled 0: Disabled
4	Panel Path Video DMA Swap R and B	RW	0x0	Swap R and B (e.g., RGB to BGR). 1: Swap enabled 0: Swap disabled
3	Panel Path Video DMA Swap U and V	RW	0x0	Swap U and V (e.g., YUYV to YVYU). 1: Swap enabled 0: Swap disabled
2	Panel Path Video DMA Swap Y and U/V	RW	0x0	Swap Y and U/V (e.g., UYVY to YUYV). 1: Swap enabled 0: Swap disabled
1	Panel Path	RW	0x0	1: Enabled

Offset: 0x190				
Bits	**Field**	**Type**	**Reset**	**Description**
	Video YUV to RGB Color Space Conversion			0: Disabled Either this or the <Panel Path Graphic YUV to RGB Color Space Conversion> field can be enabled, but both cannot be enabled simultaneously.
0	Panel Path Video and Command Transfer Enable	RW	0x0	1: Enabled 0: Disabled

12.3.6.53 LCD_PN_CTRL1_REG REGISTER

Offset: 0x194				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Panel Path DMA Transfer Trigger	RW	0x0	1: DMA transfer is triggered, this is equivalent to VSYNC pulse. This field is useful when Smart Panel VSYNC input is not available.
30:28	Panel Path VSYNC Input Trigger Modes	RW	0x0	0x0: Smart Panel VSYNC input triggers DMA start transfer. 0x1: Smart Panel VSYNC input triggers DMA transfer, and generates asynchronous interrupt to processor. 0x7: Write 1 to the <Panel Path DMA Transfer Trigger> field triggers DMA transfer All other values: Reserved
27	Panel Path Rising/Falling Edge Triggers DMA Transfer	RW	0x0	0: Rising edge triggers 1: Falling edge
26:24	Panel Path Color Key Mode	RW	0x0	0x0: Disable color key function 0x1: Video Y (or Graphic R) color key is enabled 0x2: Video U color key is enabled 0x3: Graphic RGB color key is enabled 0x4: Video V color key is enabled 0x5: Video YUV color key is enabled 0x6: Video Luma key is enabled 0x7: Graphic B color key is enabled
23	Panel Path Configure Low Bits	RW	0x0	1: Low bits are the extension of the maximum bit when converting RGB565/1555/4-bit color into 24-bit RGB color. 0: Fill zeros into low bits when converting RGB565/1555/4-bit color into 24-bit RGB color.

Offset: 0x194				
Bits	**Field**	**Type**	**Reset**	**Description**
22:20	Reserved	RO	0	Reserved for future use
19:18	Configure Scaling	RW	0x0	This field controls scaling down when Smart Panel DMA mode is enabled. - 0x0 – No scaling down - 0x1 – Scale down by 1/2 (in addition to any zooming applied) - 0x2 – Scale down by 1/4 (in addition to any zooming applied) Example: If the original image is 160 × 120 and the target size is 80 × 60, setting this field to 0x1 will further reduce the final output to 40 × 30.
17	Reserved	RO	0	Reserved for future use
16	Panel Path Alpha Selection	RW	0x0	This field determines how the graphic alpha is selected for the overlaid area in the Panel Path. - 0x0: Software-configured alpha - 0x1: Alpha is taken from the pixel data
15:8	Panel Path Configure Alpha	RW	0x0	This field configures the alpha blending in the Panel Path when neither color key alpha nor pixel-based alpha is used. - 0xFF – Full video display, no graphics - Other values – Blends video and graphics proportionally based on the set value
7:0	Pixel Command	RW	0x0	This field should be set to 0x81. All other values are reserved.

12.3.6.54 LCD_SRAM_CTRL_REG REGISTER

Offset: 0x198				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15:14	SRAM Init Write/Read	RW	0x0	0x0: Read SRAM 0x2: Write SRAM 0x3: Initialize SRAM to default value
13:12	Reserved	RO	0	Reserved for future use
11:8	SRAM Address LCD ID	RW	0x0	0x0: Panel ID_gamma_yr 0x1: Panel ID_gamma_ug 0x2: Panel ID_gamma_vb
7:0	SRAM Address[7:0]	RW	0x0	SRAM Address[7:0]

12.3.6.55 LCD_SRAM_WRDAT_REG REGISTER

Offset: 0x19C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SRAM Write Data	RW	0x0	SRAM Write Data. When generating an SRAM write command, this write data will be moved into SRAM.

12.3.6.56 LCD_SCLK_DIV_REG REGISTER

Offset: 0x1A8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	Panel Path Clock Source Select	RW	0x0	This field selects the pixel clock source 0x0: Select AXI bus clock 0x1: Select LCD Display clock 1 0x2: Select LCD Display clock 2 0x3: Select DSI PLL clock.
29	Reserved	RO	0	Reserved for future use
28	Panel Path Pixel Clock Disable	RW	0x0	0: Clock enabled 1: Clock disabled.
27:16	Panel Path Pixel Clock Fraction Divider	RW	0x0	This field fine-tunes the pixel clock when the desired clock rate cannot be achieved using only an integer divider. - Example: To obtain 80 MHz from 83 MHz, this register should be set to (83-80)/83*4096 = <0x094> Three clocks are suppressed and 80 clocks are used, and the duty cycle is not 50%. **Notes.** - This feature is useful for Smart Panels. - It is not recommended for Dumb Panels. - For basic clock adjustments, refer to the the <Clock Integer Divider> field.
15:12	Reserved	RO	0	Reserved for future use
11:8	Panel Path MIPI bit Clock Divider	RW	0x0	0x0: Clock disabled 0x1: Clock bypass (no division) 0x2 to 0x15: Panel path MIPI PLL clock is divided by 2 up to 15.
7:0	Panel Path Pixel Clock Integer Divider	RW	0x0	This field sets the integer divider to generate the required pixel clock for Smart Panel and Dumb Panel after selecting the pixel clock source.

Offset: 0x1A8				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x0: Clock disabled 0x1: Clock bypass (no division) 0x2 to 0xFF: Divides the pixel clock source by 2 to 255 **Note.** For the TV path, 0x2 to 0x15 divides the MIPI PLL clock by 2 to 15. For example, when the source AXI bus clock is 166 MHz, setting this register to 0x2 generates an 83 MHz (166 ÷ 2) clock.

12.3.6.57 LCD_PN_CONTRAST_REG REGISTER

Offset: 0x1AC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Panel Path Video Brightness Control	RW	0x0	Bits [15:8]: Used for sign extension (for positive or negative values). Bits [7:0]: Used for the integer value of brightness control. - Brightness adjustments are applied before the Color Space Conversion (CSC). - The brightness change range is +/- 0 to 255. - The value is represented in 2's complement: Example: 1. Set this field to 0x10 to make the video brighter. 2. Set this field to 0xFFF0 to make the video darker.
15:0	Panel Path Video Contrast Control	RW	0x0	Bit [15]: Used for the sign (for positive or negative values). Bit [14]: Used for the integer part. Bits [13:0]: Used for the fractional part of contrast control. - The value is represented in 2's complement: Example: 1. Set this field to 0x4000 for a ratio of 1.0 (No change) 2. Set this field to 0x6000 for a ratio of 1.5 (Increase contrast) 3. Set this field to 0x2000 for a ratio of 0.5 (Decrease contrast)

12.3.6.58 LCD_PN_SATURATION_REG REGISTER

Offset: 0x1B0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Panel Path Configure Multiplier	RW	0x0	Bit [15]: Used for the sign (for positive or negative values). Bits [14:13]: Used for integer. Bits [12:0] Used for fraction of multiplier of contrast and saturation. These bits are 2's complement code.
15:0	Panel Path Configure Saturation	RW	0x0	Bit [15]: Used for the sign (for positive or negative values). Bit [14]: Used for integer. Bits [13:0]: Used for fraction of saturation control. These bits are 2's complement code.

12.3.6.59 LCD_PN_CBSH_HUE_REG REGISTER

Offset: 0x1B4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Panel Path Video HUE Sine Correction	RW	0x0	Bit [15]: Used for the sign (positive or negative correction). Bit [14]: Used for integer. Bits [13:0]: Used for fraction of sine (delta phase). These bits are 2's complement code. Formula: Corrected U = U * cos + V * sin Corrected V = V * cos - U * sin - For example, CFG_SIN=0x0, CFG_COS=0x4000 makes no correction.
15:0	Panel Path Video HUE Cosine Correction	RW	0x0	Bit [15]: Used for the sign (positive or negative correction). Bit [14]: Used for integer. Bits [13:0]: Used for fraction of sine (delta phase). These bits are 2's complement code. Formula: Corrected U = U * cos + V * sin Corrected V = V * cos - U * sin - For example, CFG_SIN=0x2000, CFG_COS=0x376D makes a 30 degree correction.

12.3.6.60 LCD_DUMB_CTRL_REG REGISTER

Offset: 0x1B8

Bits	Field	Type	Reset	Description
31:28	Panel Path Configure Dumb Panel Color Mode	RW	0x0	This field is used for the Panel Path to convert internal RGB888 pixels into the destination color format. 0x0: LDD[15:0] is 16-bit RGB565; 0x1: LDD[23:8] is 16-bit RGB565; 0x2: LDD[17:0] is 18-bit RGB666; 0x3: LDD[23:6] is 18-bit RGB666; 0x4: LDD[11:0] is 12-bit RGB444; 0x5: LDD[23:12] is 12-bit RGB444; 0x6: LDD[23:0] is RGB888; Other values: Output blank color (as set by the Panel Screen Blank Color Register) to the I/O.
27:20	LCD GPIO Output Data	RW	0x0	LCD GPIO Output Data
19:12	LCD GPIO Output Data Enable	RW	0x0	LCD GPIO Output Data Enable
11	Panel Path Delay Graphic DMA	RW	0x0	- This field is used to reduce AXI bus activity by delaying graphic DMA until sufficient FIFO space is available for multiple bursts. - It is only applicable when: 1. The graphic path is enabled. 2. The AXI bus traffic is low. - If not used properly, FIFO underflow could occur, which would disrupt the data flow.
10	Reserved	RO	0	Reserved for future use
9	Panel Path I/O Pads Output Disable	RW	0x0	1: Disabled after next VSYNC 0x0: Enabled after next VSYNC
8	Panel Path Dumb LCD Panel GPIO Control Pin	RW	0x0	This field controls the backlight of a Dumb LCD Panel. Active only when both the <Dumb Panel Backlight Clock Divider> and <Duty Cycle Control> fields in the Graphic Line Length (Pitch) Register are 0x0000.
7	Panel Path Configure Reverse RGB Bit Order	RW	0x0	1: Reverse the RGB bit order 1. For example, R[7:0] G[7:0] B[7:0] are reversed to R[0:7] G[0:7] B[0:7]. 0: Do not reverse.
6	Panel Path Invert Composite Blank Signal	RW	0x0	1: Invert 0: Do not invert.
5	Panel Path Invert Composite Sync Signal	RW	0x0	1: Invert 0: Do not invert.
4	Panel Path Invert Pixel Valid Enable	RW	0x0	1: Invert 0: Do not invert.

Offset: 0x1B8				
Bits	**Field**	**Type**	**Reset**	**Description**
3	Panel Path Invert VSYNC	RW	0x0	1: Invert 0: Do not invert.
2	Panel Path Invert HSYNC	RW	0x0	1: Invert 0: Do not invert.
1	Panel Path Invert Pixel Clock	RW	0x0	1: Invert 0: Do not invert.
0	Panel Path Enable Dumb LCD Panel	RW	0x0	1: Enabled **Note.** Smart Panel should be disabled when this bit is 1 0: Disabled.

12.3.6.61 PN_IOPAD_CONTROL_REG REGISTER

Offset: 0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Mask Panel Path Video Y SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
30	Mask Panel Path Video U SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
29	Mask Panel Path Video V SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
28	Mask Panel Path Graphic Y SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
27	Mask TV Path Video Y SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1.

Offset: 0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
				0 = Updates SA on each frame.
26	Mask TV Path Video U SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
25	Mask TV Path Video V SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
24	Mask TV Path Graphic Y SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
23	Mask Panel Path Command SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
22	Mask TV Path Command SA Update	RW	0x0	1: Masks Y Starting Address (SA) update for Panel Path Video. Updates can occur once after writing both this bit and the <Enable SA Update> field to 1. 0 = Updates SA on each frame.
21	Reserved	RO	0	Reserved for future use
20	Enable SA Update	RW	0x0	- When set to 1, and the mask function is enabled, the Starting Address will be updated only once after VSYNC. - If the mask function is disabled, this field is not used.
19	TV Path Graphic Vertical Mirror Enable	RW	0x0	1: Enables vertical mirroring. DMA fetches from the last line to the first line. Note: The starting address for DMA must be set to the last line's starting address. 0: Disable
18	Reserved	RO	0	Reserved for future use
17	TV Path Video Image Vertical Mirror Enable	RW	0x0	1: Enables vertical mirroring. DMA fetches from the last line to the first line. Note: The starting address for DMA must be set to the last line's starting address.

Offset: 0x1BC				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Disable
16	Reserved	RO	0	Reserved for future use
15	Panel Path Graphic Vertical Mirror Enable	RW	0x0	1: Enables vertical mirroring. DMA fetches from the last line to the first line. Note: The starting address for DMA must be set to the last line's starting address. 0: Disable
14	Reserved	RO	0	Reserved for future use
13	Panel Path Video Image Vertical Mirror Enable	RW	0x0	1: Enables vertical mirroring. DMA fetches from the last line to the first line. Note: The starting address for DMA must be set to the last line's starting address. 0: Disable
12	Panel Path Command Vertical Mirror Enable	RW	0x0	1: Enables vertical mirroring. DMA fetches from the last line to the first line. Note: The starting address for DMA must be set to the last line's starting address. 0: Disable
11:10	TV Path Configure Color Space Conversion	RW	0x0	Configures color space conversion for TV Path: 0x0: CCIR601 YUV → Computer RGB 0x1: CCIR601 YUV → Studio RGB 0x2: CCIR709 YUV → Computer RGB 0x3: CCIR709 YUV → Studio RGB
9:8	Panel Path Configure Color Space Conversion	RW	0x0	Configures color space conversion for Panel Path: 0x0: CCIR601 YUV → Computer RGB 0x1: CCIR601 YUV → Studio RGB 0x2: CCIR709 YUV → Computer RGB 0x3: CCIR709 YUV → Studio RGB
7:6	Reserved	RO	0	Reserved for future use
5	Indicates Boundary	RW	0x0	0: No crossing 4 KB boundary 1: No crossing 1 KB boundary (usually for DDR memory).
4	Indicates Cycle Burst Length	RW	0x0	0: 8-cycle burst 1: 16-cycle burst (recommended for better performance).
3:0	Reserved	RO	0	Reserved for future use.

12.3.6.62 SPU_IRQ_ENA_REG REGISTER

Offset: 0x1C0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Panel Path DMA Frame 0 Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
30	smt tx done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
29	Panel Path DMA FIFO Underflow IRQ Enable	RW	0x0	1: Enabled 0: Disabled
28	AXI Bus Error IRQ Enable	RW	0x0	1: Enabled 0: Disabled
27	Panel Path Graphic Frame 0 IRQ Enable	RW	0x0	1: Enabled 0: Disabled
26	Panel Path Graphic Frame 1 IRQ Enable	RW	0x0	1: Enabled 0: Disabled
25	Panel Path Graphic FIFO Underflow IRQ Enable	RW	0x0	1: Enabled 0: Disabled
24	TV Path Hardware Cursor/OSD Frame Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
23	Panel Path VSYNC Input Rising Edge IRQ Enable	RW	0x0	1: Enabled 0: Disabled
22	Panel Path Dumb LCD Panel Frame Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
21	Panel Path Smart Panel Frame Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
20	SPI Transfer Frame Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
19	AHB Slave Path All Command Empty IRQ Enable	RW	0x0	1: Enabled 0: Disabled
18	SPI Bus Transfer Complete IRQ Enable	RW	0x0	1: Enabled 0: Disabled
17	Power Down Request IRQ Enable	RW	0x0	1: Enabled 0: Disabled
16	AXI Bus Latency Too Long IRQ Enable	RW	0x0	1: Enabled 0: Disabled
15	Write Back fifo overrun Enable	RW	0x0	1: Enabled 0: Disabled
14	Write Back Dma Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled

Offset: 0x1C0				
Bits	**Field**	**Type**	**Reset**	**Description**
13	Write Back fifo underrun IRQ Enable	RW	0x0	1: Enabled 0: Disabled
12	TV Path VSYNC Input Rising Edge IRQ Enable	RW	0x0	1: Enabled 0: Disabled
11	TV Path Graphic Frame 0 IRQ Enable	RW	0x0	1: Enabled 0: Disabled
10	TV Path Graphic Frame 1 IRQ Enable	RW	0x0	1: Enabled 0: Disabled
9	TV Path Graphic FIFO Underflow IRQ Enable	RW	0x0	1: Enabled 0: Disabled
8	TV Path Display Frame Done IRQ Enable	RW	0x0	1: Enabled 0: Disabled
7:0	Reserved	RO	0	Reserved for future use

12.3.6.63 SPU_IRQ_ISR_RAW_REG REGISTER

Offset: 0x1C4				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Panel Path Video Frame 0 Done Rising Edge IRQ	RW	0x0	Panel Path Video Frame 0 Done Rising Edge IRQ (before masking). 0: Clear interrupt 1: No effect
30	smt tx done IRQ	RW	0x0	SMT TX Done IRQ (before masking) 0: Clear interrupt 1: No effect
29	Reserved	RO	0	Reserved for future use
28	AXI Bus Error IRQ	RW	0x0	AXI Bus Error IRQ (before masking) 0: Clear interrupt 1: No effect
27	Panel Path Graphic Frame 0 Done Rising Edge IRQ	RW	0x0	Panel Path Graphic Frame 0 Done Rising Edge IRQ (before masking) 0: Clear interrupt 1: No effect
26	Panel Path Graphic Frame 1 Done Rising Edge IRQ	RW	0x0	Panel Path Graphic Frame 1 Done Rising Edge IRQ (before masking) 0: Clear interrupt 1: No effect

Offset: 0x1C4				
Bits	**Field**	**Type**	**Reset**	**Description**
25	Panel Path Graphic FIFO Underflow IRQ	RW	0x0	Panel Path Graphic FIFO Underflow IRQ (before masking) 0: Clear interrupt 1: No effect
24	Reserved	RO	0	Reserved for future use
23	Panel Path VSYNC Input Rising Edge IRQ	RW	0x0	Panel Path VSYNC Input Rising Edge IRQ (before masking) 0: Clear interrupt 1: No effect
22	Reserved	RO	0	Reserved for future use
21	Panel Path Smart Panel Display Area DMA Done IRQ	RW	0x0	Panel Path Smart Panel Display Area DMA Done IRQ (before masking) 0: Clear interrupt 1: No effect
20	SPI Transfer Frame Done IRQ	RW	0x0	SPI Frame data Transfer Done IRQ (before masking) 0: Clear interrupt 1: No effect
19	AHB Slave Path All Commands Output Done IRQ	RW	0x0	AHB Slave Path All Commands Output Done IRQ (before masking) 0: Clear interrupt 1: No effect
18	SPI Bus Transfer Done IRQ	RW	0x0	SPI Bus Transfer Done IRQ (before masking) 0: Clear interrupt 1: No effect
17	Reserved	RO	0	Reserved for future use
16	AXI Bus Latency Too Long IRQ	RW	0x0	AXI Bus Latency Too Long IRQ This interrupt occurs when the response time exceeds 512 bus clocks (before masking). 0: Clear interrupt 1: No effect
15	Write Back fifo overrun	RW	0x0	Write Back FIFO underrun
14	Write Back Dma Done	RW	0x0	Write Back DMA Done
13	Write Back fifo underrun	RW	0x0	Write Back FIFO underrun
12	TV Path VSYNC Input Rising Edge IRQ	RW	0x0	TV Path VSYNC Input Rising Edge IRQ (before masking). 0: Clear interrupt 1: No effect

Offset: 0x1C4				
Bits	**Field**	**Type**	**Reset**	**Description**
11	TV Path Graphic Frame 0 Done Rising Edge IRQ	RW	0x0	TV Path Graphic Frame 0 Done Rising Edge IRQ (before masking) 0: Clear interrupt 1: No effect
10	TV Path Graphic Frame 1 Done Rising Edge IRQ	RW	0x0	TV Path Graphic Frame 1 Done Rising Edge IRQ (before masking) 0: Clear interrupt 1: No effect
9	TV Path Graphic FIFO Underflow IRQ	RW	0x0	TV Path Graphic FIFO Underflow IRQ (before masking) 0: Clear interrupt 1: No effect
8	TV Path Display Screen Done IRQ	RW	0x0	TV Path Display Screen Done IRQ. 0: Clear interrupt 1: No effect
7:4	Reserved	RO	0	Reserved for future use
3	Level of DMA_FF_EMPY	RW	0x0	Level of DMA_FF_EMPY (before masking)
2	Level of GRA_FF_EMPTY	RW	0x0	Level of GRA_FF_EMPTY (before masking)
1	Reserved	RO	0	Reserved for future use
0	Level of TVG_FF_EMPTY	RW	0x0	Level of TVG_FF_EMPTY (before masking)

12.3.6.64 SPU_IRQ_RSR_REG REGISTER

Offset: 0x1C8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Read to Reset Status Register (Clean ISR[31:8])	RW	0x0	- When SPU_IRQ_RSR[i] = 1: 1. Reading SPU_IRQ_ISR[i] clears the status and masks the interrupt. 2. If a new event occurs, the status is updated, but no new interrupt is triggered unless 0 is written to SPU_IRQ_ISR[i]. - When SPU_IRQ_RSR[i] = 0: 1. Reading SPU_IRQ_ISR[i] does not clear the status; 2. To clear both the status and the interrupt mask, write 0 to SPU_IRQ_ISR[i].

Offset: 0x1C8				
Bits	**Field**	**Type**	**Reset**	**Description**
				This mechanism is useful for systems that use status polling.
7:0	Reserved	RO	0	Reserved for future use

12.3.6.65 LCD_GRA_CUTHPXL_REG REGISTER

Offset: 0x1CC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Graphic Color [7:4]	RW	0x0	- Defines the upper 4 bits of the Graphic Color used in the Partial Display Disable area. - In the disabled area, the graphic is covered by video, so there is no need to read graphic memory. - The disabled rectangle hollow area is defined by horizontal starting pixel, ending pixel and vertical starting line, ending line. - Within the Partial Display Disable area, the pseudo 64-bit read data is formed as: {2{2{GRA_CUTCOLOR[15:0]}}}. - This field is part of a spare area that defines the pseudo 64-bit read data color. - By default: All bits are 0.
27:16	Partial Display Disable Horizontal Ending Pixel Number	RW	0x0	- Specifies the horizontal ending pixel number of the Partial Display Disable area. - This value must be less than the source graphic horizontal pixel number defined in the the <Panel Path Graphic Source Horizontal Pixel Number> field (in the Panel Graphic Source Size Register). - The area between the starting pixel number and ending pixel number is the horizontal gap where graphic memory reads are disabled to reduce bandwidth.
15:12	Graphic Color [3:0]	RW	0x0	- Defines the lower 4 bits of the Graphic Color used in the Partial Display Disable area. - Similar to Bits [31:28], this field is part of the pseudo 64-bit read data color definition.
11:0	Partial Display Disable Horizontal Starting Pixel Number	RW	0x0	- Specifies the horizontal starting pixel number of the Partial Display Disable area. - This value must be less than the source graphic horizontal pixel number defined in the the <Panel Path Graphic Source Horizontal Pixel Number> field (in the Panel Graphic Source Size Register). - The area between the starting pixel number and

Offset: 0x1CC				
Bits	**Field**	**Type**	**Reset**	**Description**
				ending pixel number is the horizontal gap where graphic memory reads are disabled to reduce bandwidth.

12.3.6.66 LCD_GRA_CUTVLN_REG REGISTER

Offset: 0x1D0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Graphic Color [15:12]	RW	0x0	- Defines bits [15:12] of the Graphic Color (GRA_CUTCOLOR[15:0]) used in the Partial Display Disable area. - In the disabled area, the graphic is covered by video, so there is no need to read graphic memory. - The disabled rectangle hollow area is defined by horizontal starting pixel, ending pixel and vertical starting line, ending line. - The pseudo 64-bit read data in this area is composed of {2{2{GRA_CUTCOLOR[15:0]}}}. - This field is part of a spare area that defines the pseudo 64-bit read data color. - By default: All bits are 0.
27:16	Partial Display Disable Vertical Ending Pixel Number	RW	0x0	- Specifies the vertical ending line number of the Partial Display Disable area. - This value must be less than the source graphic vertical line number defined in the <Panel Path Graphic Source Vertical Line Number> field (in the Panel Graphic Source Size Register). - The area between the starting line number and ending line number is the vertical gap where graphic memory reads are disabled to reduce bandwidth.
15:12	Graphic Color [11:8]	RW	0x0	- Defines bits [11:8] of the Graphic Color (GRA_CUTCOLOR[15:0]) used in the Partial Display Disable area. - Similar to Bits 31:28, this field is part of the pseudo 64-bit read data color definition.
11:0	Partial Display Disable Vertical Starting Pixel Number	RW	0x0	- Specifies the vertical starting line number of the Partial Display Disable area. - This value must be less than the source graphic vertical line number defined in the <Panel Path Graphic Source Vertical Line Number> field (in the Panel Graphic Source Size Register). - The area between the starting line number and ending line number is the vertical gap where graphic memory

Offset: 0x1D0				
Bits	**Field**	**Type**	**Reset**	**Description**
				reads are disabled to reduce bandwidth.

12.3.6.67 LCD_TVG_CUTHPXL_REG REGISTER

Offset: 0x1D4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Graphic Color [7:4]	RW	0x0	- Defines bits [7:4] of the Graphic Color (TVG_CUTCOLOR[15:0]) used in the Partial Display Disable area. - In the disabled area, the graphic is covered by video, so there is no need to read graphic memory. - The disabled rectangle hollow area is defined by horizontal starting pixel, ending pixel and vertical starting line, ending line. - The pseudo 64-bit read data in this area is composed of {2{2{TVG_CUTCOLOR[15:0]}}}. - This field is part of a spare area that defines the pseudo 64-bit read data color. - By default: All bits are 0.
27:16	Partial Display Disable Horizontal Ending Pixel Number	RW	0x0	- Specifies the horizontal ending pixel number of the Partial Display Disable area. - This value must be less than the source graphic horizontal pixel number defined in the <TV Path Graphic Source Horizontal Pixel Number> field (in the TV Graphic Source Size Register). - The area between the starting pixel number and ending pixel number is the horizontal gap where graphic memory reads are disabled to reduce bandwidth.
15:12	Graphic Color [3:0]	RW	0x0	- Defines bits [3:0] of the Graphic Color (TVG_CUTCOLOR[15:0]) used in the Partial Display Disable area. - Similar to Bits 31:28, this field is part of the pseudo 64-bit read data color definition.
11:0	Partial Display Disable Horizontal Starting Pixel Number	RW	0x0	- Specifies the horizontal starting pixel number of the Partial Display Disable area. - This value must be less than the source graphic horizontal pixel number defined in the <TV Path Graphic Source Horizontal Pixel Number> field (in the TV Graphic Source Size Register). - The area between the starting pixel number and ending pixel number is the horizontal gap where graphic memory reads are disabled to reduce bandwidth.

12.3.6.68 LCD_TVG_CUTVLN_REG REGISTER

Offset: 0x1D8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Graphic Color [15:12]	RW	0x0	- Defines bits [15:12] of the Graphic Color (GRA_CUTCOLOR[15:0]) used in the Partial Display Disable area. - In the disabled area, the graphic is covered by video, so there is no need to read graphic memory. - The disabled rectangle hollow area is defined by horizontal starting pixel, ending pixel and vertical starting line, ending line. - The pseudo 64-bit read data in this area is composed of {2{2{GRA_CUTCOLOR[15:0]}}}. - This field is part of a spare area that defines the pseudo 64-bit read data color.
27:16	Partial Display Disable Vertical Ending Pixel Number	RW	0x0	- Specifies the vertical ending line number of the Partial Display Disable area. - This value must be less than the source graphic vertical line number defined in the <Panel Path Graphic Source Vertical Line Number> field (in the Panel Graphic Source Size Register). - The area between the starting line number and ending line number is the vertical gap where graphic memory reads are disabled to reduce bandwidth.
15:12	Graphic Color [11:8]	RW	0x0	- Defines bits [11:8] of the Graphic Color (GRA_CUTCOLOR[15:0]) used in the Partial Display Disable area. - Similar to Bits 31:28, this field is part of the pseudo 64-bit read data color definition.
11:0	Partial Display Disable Vertical Starting Pixel Number	RW	0x0	- Specifies the vertical starting line number of the Partial Display Disable area. - This value must be less than the source graphic vertical line number defined in the in the <Source Vertical Line Number> field (in the Panel Graphic Source Size Register). - The area between the starting line number and ending line number is the vertical gap where graphic memory reads are disabled to reduce bandwidth.

12.3.6.69 LCD_TOP_CTRL_REG REGISTER

Offset: 0x1DC				
Bits	**Field**	**Type**	**Reset**	**Description**

Offset: 0x1DC				
Bits	**Field**	**Type**	**Reset**	**Description**
31	Invert I/O Pad VSYNC	RW	0x0	1: Invert dumb panel I/O pad VSYNC signal 0: No change
30	Invert I/O Pad HSYNC	RW	0x0	1: Invert Dumb panel IO pad HSYNC signal 0: No change
29	Invert I/O Pad PCLK	RW	0x0	1: Invert Dumb panel IO pad PCLK signal 0: No change
28	Invert I/O Pad DENA	RW	0x0	1: Invert Dumb panel IO pad DENA signal 0: No change
27:24	Panel Path Configure MIPI DSI1 or CMU Input Color Mode	RW	0x0	This field is used for the Panel Path to convert internal RGB888 pixels into the MIPI DSI1 color format. 0x0: 24-bit RGB88 0x1: 24-bit RGB88, but swap R and B Others: Reserved
23:22	Select All Objects for Panel or TV Interface	RW	0x0	This field selects all four DMA objects to go to Panel or HDMI TV. 0x0: Auto-detect 0x1: All DMA objects go to Panel 0x2: All DMA objects go to HDMI TV 0x3: Panel path DMA to Panel, TV path DMA to TV
21	Select Clock when One Clock Domain is Enabled	RW	0x0	0: Select panel clock for TV when one clock domain is enabled. 1: Select TV clock for Panel when one clock domain is enabled.
20	Clock Domain Selection	RW	0x0	This field enables one clock domain for both TV and Panel. 1: One clock domain is selected 0: TV path and Panel path use different clock domains.
19	Swap TV Path Processing	RW	0x0	0: TV path video processing is for video DMA, palette table is for graphic DMA. 1: TV path video processing is for graphic DMA, palette table is for video DMA.
18	Swap TV Path Enable	RW	0x0	0: Auto detect TV path video processing path (<Swap TV Path Processing> field is ignored) 1: TV path video processing is selected by the <Swap TV Path Processing> field.
17	Swap Panel Path Processing	RW	0x0	0: Panel path video processing is for video DMA, palette table is for graphic DMA. 1: Panel path video processing is for graphic DMA, palette table is for video DMA.

Offset: 0x1DC				
Bits	**Field**	**Type**	**Reset**	**Description**
16	Swap Panel Path Enable	RW	0x0	0: Auto detect Panel path video processing path (<Swap Panel Path Processing> field is ignored). 1: Panel path video processing is selected by the <Swap Panel Path Processing> field.
15:14	Select TV Path Graphic DMA Burst Length	RW	0x0	0x0: One burst (64 bytes) 0x1: Two bursts (128 bytes) 0x2: Three bursts (192 bytes) 0x3: Four bursts (256 bytes)
13:12	Select TV Path Video DMA Burst Length	RW	0x0	0x0: One burst (64 bytes) 0x1: Two bursts(128 bytes) 0x2: Three bursts (192 bytes) 0x3: Four bursts (256 bytes)
11:10	Select Panel Path Graphic DMA Burst Length	RW	0x0	0x0: One burst (64 bytes) 0x1: Two bursts (128 bytes) 0x2: Three bursts(192 bytes) 0x3: Four bursts (256 bytes)
9:8	Select Panel Path Video DMA Burst Length	RW	0x0	0x0: One burst (64 bytes) 0x1: Two bursts (128 bytes) 0x2: Three bursts(192 bytes) 0x3: Four bursts (256 bytes)
7:6	AHB Slave Read Wait Cycles	RW	0x0	0x0: One AHB read wait cycle 0x1: Two AHB read wait cycles 0x2: Three AHB read wait cycles 0x3: Four AHB read wait cycles
5:4	AHB Slave Write Wait Cycles	RW	0x0	0x0: One AHB read wait cycle 0x1: Two AHB read wait cycles 0x2: Three AHB read wait cycles 0x3: Four AHB read wait cycles
3:0	Reserved	RO	0	Reserved for future use.

12.3.6.70 LCD_AFA_ALL2ONE_REG REGISTER

Offset: 0x1E8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23	Enable Two-Level Zoom Down in TV Path Graphic DMA	RW	0x0	Enables two-level zoom down for the TV Path Graphic DMA. The source horizontal pixel (defined in the <TV Path Graphic Source Horizontal Pixel Number> field (in

Offset: 0x1E8				
Bits	**Field**	**Type**	**Reset**	**Description**
				the TV Graphic Source Size Register)) must be an even number. The zoom down is performed in two stages: - First, the image is divided by 2. - Then, it is further shrunk to the exact destination pixel number. 0x0: Disabled (direct zoom down, limited by 64-bit/pixel clock). 0x1: Enabled (allows higher zoom down ratios, e.g., for RGB565 mode, zoom down is limited by 4x, while two-level zoom down can make it to 8x).
22	Enable Two-Level Zoom Down in TV Path Video DMA	RW	0x0	Enables two-level zoom down for the TV Path Video DMA. Similar to Bit [23], but applies to the Video DMA. 0x0: Disabled (direct zoom down, limited by 64-bit/pixel clock). 0x1: Enabled (allows higher zoom down ratios, e.g., for RGB565 mode, zoom down is limited by 4x, while two-level zoom down can make it to 8x).
21	Enable Two-Level Zoom Down in Panel Path Graphic DMA	RW	0x0	Enables two-level zoom down for the Panel Path Graphic DMA. The source horizontal pixel (defined in the <TV Path Graphic Source Horizontal Pixel Number> field (in the TV Graphic Source Size Register)) must be an even number. The zoom down is performed in two stages: - First, the image is divided by 2. - Then, it is further shrunk to the exact destination pixel number. 0x0: Disabled (direct zoom down, limited by 64-bit/pixel clock). 0x1: Enabled (allows higher zoom down ratios, e.g., for RGB565 mode, zoom down is limited by 4x, while two-level zoom down can make it to 8x).
20	Enable Two-Level Zoom Down in Panel Path Video DMA	RW	0x0	Enables two-level zoom down for the Panel Path Video DMA. Similar to Bit [21], but applies to the Video DMA. 0x0: Disabled (direct zoom down, limited by 64-bit/pixel clock). 0x1: Enabled (allows higher zoom down ratios, e.g., for RGB565 mode, zoom down is limited by 4x, while two-level zoom down can make it to 8x).
19:18	Enable Graphic DMA Vertical Smooth	RW	0x0	Enables vertical smoothing for graphic DMA. Two graphic DMA channels are used to read two

Offset: 0x1E8				
Bits	**Field**	**Type**	**Reset**	**Description**
				lines near the resampling line and apply smooth filtering. 0x0: Disabled (Graphic vertical smooth is not done). 0x1: Reserved. 0x2: Panel graphic vertical smooth (TV graphic DMA filters Panel graphic). 0x3: TV graphic vertical smooth (Panel graphic DMA filters TV graphic).
17:16	Enable Video DMA Vertical Smooth	RW	0x0	Enables vertical smoothing for video DMA. Two video DMA channels are used to read two lines near the resampling line and apply smooth filtering. 0x0: Disabled (Video vertical smooth is not done). 0x1: Reserved. 0x2: Panel video vertical smooth (TV video DMA filters Panel video). 0x3: TV video vertical smooth (Panel video DMA filters TV video).
15:14	Select Alpha when Panel and TV Path Graphics Overlaid	RW	0x0	Selects the alpha blending source when Panel and TV Path graphics are overlaid. 0x0: TV path graphic DMA alpha. 0x1: Panel path graphic DMA alpha. 0x2: Panel path configured alpha. 0x3: TV path configured alpha.
13:12	Select Alpha when Panel Path Graphic and TV Path Video Overlaid	RW	0x0	Selects the alpha blending source when Panel Path graphic and TV Path video are overlaid. 0x0: TV path video DMA alpha. 0x1: Panel path graphic DMA alpha. 0x2: Panel path configured alpha. 0x3: TV path configured alpha.
11:10	Select Alpha when Panel and TV Path Graphics Overlaid	RW	0x0	Selects the alpha blending source when Panel and TV Path graphics are overlaid. 0x0: Panel path video DMA alpha. 0x1: TV path graphic DMA alpha. 0x2: Panel path configured alpha. 0x3: TV path configured alpha.
9:8	Select Alpha when Panel and TV Path Videos Overlaid	RW	0x0	Selects the alpha blending source when Panel and TV Path videos are overlaid. 0x0: Panel path video DMA alpha. 0x1: TV path video DMA alpha. 0x2: Panel path configured alpha. 0x3: TV path configured alpha.
7:5	Reserved	RO	0	Reserved for future use
4	Select Hardware Cursor	RW	0x0	When one path is disabled, all four objects and two

Offset: 0x1E8				
Bits	**Field**	**Type**	**Reset**	**Description**
	when Cursor Overlaid			cursor objects can be overlaid together. Selects the hardware cursor when two cursors are overlaid. 0x0: Use Panel path cursor 0x1: Use TV path cursor
3:2	Reserved	RO	0	Reserved for future use
1:0	Alpha Blending Mode	RW	0x0	Alpha Blending Mode Selection. 0x0 = L0*(1-A1)+L1*A1, 0x1 = L0*(1-A1)+L1, 0x2 =L0*A1+L1, 0x3 = reserved. Selects the alpha blending mode. 0x0: L0*(1-A1) + L1*A1 0x1: L0*(1-A1) + L1 0x2: L0*A1 + L1 0x3: Reserved

12.3.6.71 LCD_DITHER_CTRL_REG REGISTER

Offset: 0x1EC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	Reserved	RO	0	Reserved for future use
17:16	Dither Table Index Selection	RW	0x0	Dither Table Index Selection. This field is used when software needs to read from or write to the dither table. Selects the dither table index for software access. 0x0: Access dither table index 0. 0x1: Access dither table index 1. 0x2: Access dither table index 2. 0x3: Access dither table index 3.
15	Reserved	RO	0	Reserved for future use
14:12	Dither Mode Selection For TV Path	RW	0x0	Selects the dither mode for the TV Path. 0x0: RGB 444 mode. 0x1: RGB 565 mode. 0x2: RGB 666 mode.
11:10	Reserved	RO	0	Reserved for future use
9	Dither Table 4x4 or 4x8 for TV Path	RW	0x0	Selects the dither table size for the TV Path. 0x0: 4x4 dither table. 0x1: 4x8 dither table.
8	Dither Enable for TV Path	RW	0x0	Enables or disables dithering for the TV Path. 0x0: Disable dithering.

Offset: 0x1EC				
Bits	**Field**	**Type**	**Reset**	**Description**
				0x1: Enable dithering.
7	Reserved	RO	0	Reserved for future use
6:4	Dither Mode Selection for Panel Path	RW	0x0	Selects the dither mode for the Panel Path. 0x0: RGB 444 mode. 0x1: RGB 565 mode. 0x2: RGB 666 mode.
3:2	Reserved	RO	0	Reserved for future use
1	Dither Table 4x4 or 4x8 for Panel Path	RW	0x0	Selects the dither table size for the Panel Path. 0x0: 4x4 dither table. 0x1: 4x8 dither table.
0	Dither Enable for Panel Path	RW	0x0	Enables or disables dithering for the Panel Path. 0x0: Disable dithering. 0x1: Enable dithering.

12.3.6.72 LCD_DITHER_TBL_DATA_REG REGISTER

Offset: 0x1F0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	LCD Dither Table Data Port	RW	0x0	There is a total of 128 bits for the dither table. To access the dither table, use the <Dither Table Select Index> in the Dither Control Register.

12.3.6.73 LCD_MISC_CTRL_REG REGISTER

Offset: 0x1F8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Configure SPI Transmit HW	RW	0x0	hw control tx ibits, only for spi fast mode image. Configure SPI Transmit. 0x1F = Write/transmit 32 bits, 0x01 = Write/transmit 2 bits, 0x00 = Do not transfer. If the <Configure SPI Receive> field is set to 0x00, there is no receiving, only transmit. The maximum write length is 32 bits per trigger , and unlimited write length can be reached if triggering one after another and if the <Configure Continuous Transfer> field is set to 0x01. If the <Configure SPI Receive> field is not 0x00, it will first transmit serial bits, and then receive incoming bits. Hardware-Controlled SPI Transmit (Fast Mode Only):

Offset: 0x1F8				
Bits	**Field**	**Type**	**Reset**	**Description**
				This field configures SPI data transmission in fast mode. 0x1F: Write/transmit 32 bits. 0x01: Write/transmit 2 bits. 0x00: Do not transfer. - If the <Configure SPI Receive> field is set to `0x00`, only data transmission occurs without receiving. 1. The maximum write length is 32 bits per trigger. 2. Continuous transmission is possible if triggers are sent consecutively and the <Configure Continuous Transfer> field is set to 0x01. - If the Configure SPI Receive field is not 0x00, the system will first transmit the serial bits and then receive incoming data.
23:21	cfg_ch_time	RW	0x0	cs hold time = (cfg_ch_time+1)*period_sclk
20:19	cfg_csu_time	RW	0x0	cs setup time = (cfg_csu_time+2)*period_sclk
18	Reserved	RO	0	Reserved for future use
17	slv_fast_mode	RW	0x0	indicate cmd in burst mode
16	slv_burst_trigger	RW	0x0	Set to 1 to trigger burst mode after command is settled (for slv_fast_mode).
15:12	burst_length_hw	RW	0x0	Hardware-controlled burst length (actual length is burst_length_hw + 1) for fast mode.
11:8	burst_length_sw	RW	0x0	Software-controlled burst length (actual length is burst_length_sw + 1).
7	spi cmd trigger	RW	0x0	Write 1 before setting cmd only for fast mode
6	spi_fast_mode	RW	0x0	SPI fast mode control: 0x1: Software does not wait for SPI command completion IRQ. 0x0: Software waits.
5:4	smpn2spi_mode	RW	0x0	0x0: 1 data lane mode 0x1: 2 data lane RGB888-3cycle mode 0x2: 2 data lane RGB666-3cycle mode
3	Configure 3-/4-line SPI	RW	0x0	1: 3-line SPI, 9bit serial data 0: 4-line SPI, 8bit serial data+D/CX pin.
2	spi_2ln_mode	RW	0x0	0x0: 1 data lane mode 0x1: 2 data lane mode.
1	Smart Panel RB swap	RW	0x0	0x0: No swap 0x1: Swap RB (Set 1 for GC9305).
0	SPI data path swap to hw	RW	0x0	0x0: CPU control 0x1: Hareware control

12.3.6.74 LCD_WDMA_CTRL1_REG REGISTER

Offset: 0x200				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	wdma_img_pitch	RW	0x0	wdma pitch by bytes
15:13	Reserved	RO	0	Reserved for future use
12:8	wdma_burst_len	RW	0x0	wdma burst length by bytes
7:6	Reserved	RO	0	Reserved for future use
5:4	wdma_pix_fmt	RW	0x0	0: 16bit RGB565 1: 24bit RGB888 2: 32bit ARGB8888 3: 32bit RGBA8888
3:1	Reserved	RO	0	Reserved for future use
0	wdma_ena	RW	0x0	1: Valid

12.3.6.75 LCD_WDMA_CTRL2_REG REGISTER

Offset: 0x204				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	wdma_img_height	RW	0x0	wdma_img_height
15:13	Reserved	RO	0	Reserved for future use
12:0	wdma_img_width	RW	0x0	wdma_img_width

12.3.6.76 LCD_WDMA_CTRL3_REG REGISTER

Offset: 0x208				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	wdma_base_addr	RW	0x0	wdma_base_addr

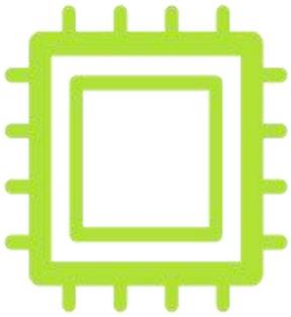
12.3.6.77 LCD_WDMA_CTRL4_REG REGISTER

Offset: 0x20C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:21	Reserved	RO	0	Reserved for future use

Offset: 0x20C				
Bits	**Field**	**Type**	**Reset**	**Description**
20	dmac_wr_err	RO	0x0	dmac_wr_err
19	dmac_rst_n_pwr	RW	0x0	dmac_rst_n_pwr
18	dmac_rst_req	RW	0x0	dmac_rst_req
17	dmac_wr_int_clr	RW	0x0	dmac_wr_int_clr
16	dmac_axi_sec	RW	0x0	dmac_axi_sec
15	dmac_wr_post_en	RW	0x0	dmac_wr_post_en
14:12	dmac_max_req_num	RW	0x0	dmac_max_req_num
11:8	dmac_arqos	RW	0x0	dmac_awqos
7:4	dmac_awqos	RW	0x0	dmac_awqos
3:0	dmac_user_id	RW	0x0	dmac_user_id

Chapter 13

Video Capture Subsystem

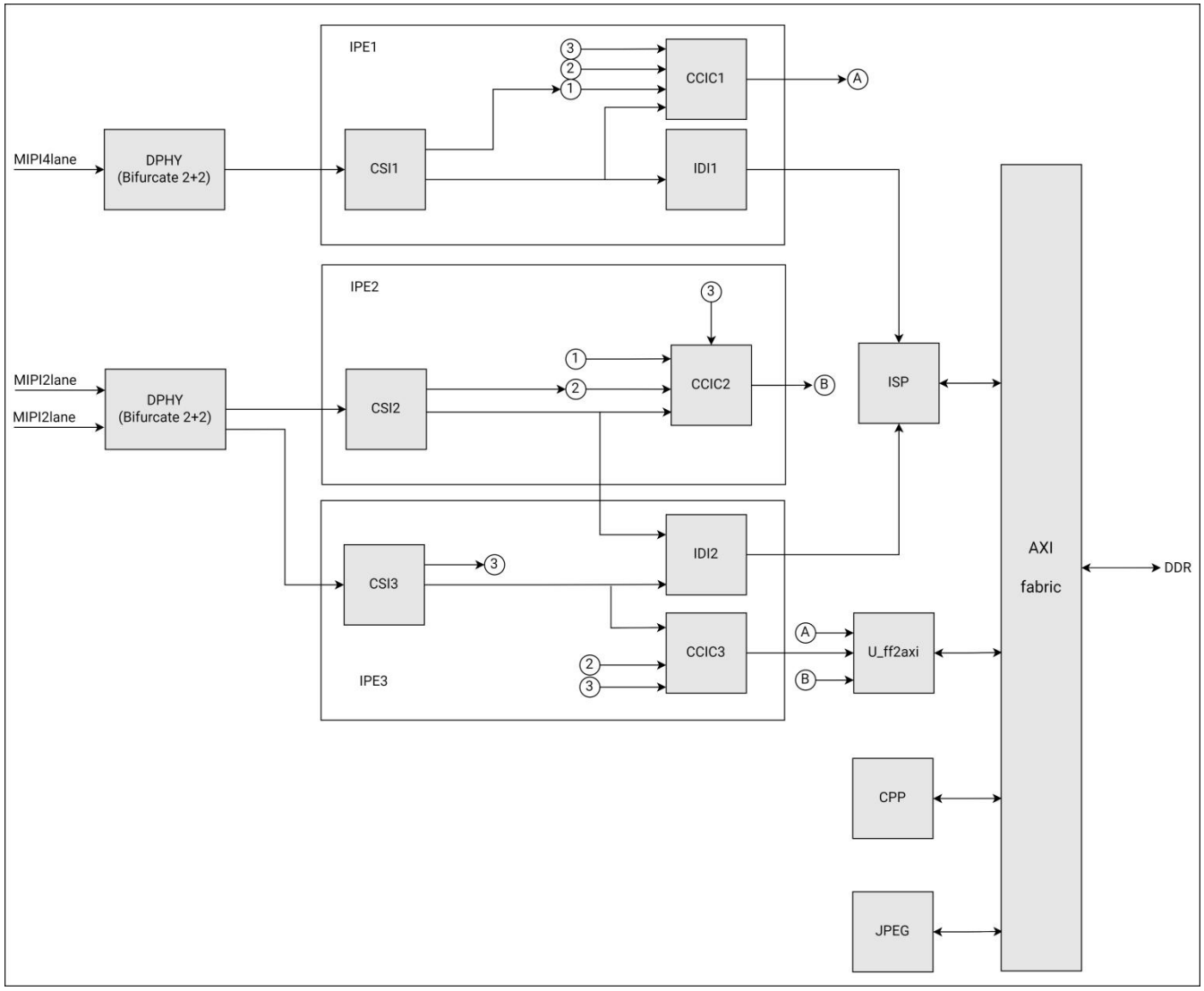


Key Stone® K1 User Manual

13.1 Overview

The video capture subsystem includes, among other components, MIPI Camera IN Interface and ISP, and is responsible for preview, video and capture functions, as well as image processing from the camera sensor. It supports up to 3 camera sensors simultaneously. The processed image data is then written back to DDR for further use.

The block diagram of the video capture subsystem is depicted below.



13.2 MIPI Camera IN Interface

13.2.1 Introduction

The MIPI Camera IN interface features two MIPI-CSI2 v1.1 controllers both equipped with 4 lanes each of which supports a maximum transfer rate of 1.5Gbps.

13.2.2 Features

- Support for the following modes to allocate lanes to sensors:
  - 4-Lane + 4-Lane mode (double sensor)
  - 4-Lane + 2-Lane mode (double sensor)
  - 4-Lane + 2-Lane + 2-Lane mode (triple sensor)

    Note. In “4-Lane + 2-Lane + 2-Lane mode (triple sensor)”, only 2 Bayer RAW and 1 YUV input format are supported.
- Support for the following input formats:
  - Legacy YUV420 8-bit
  - YUV420 8-bit
  - RAW8
  - RAW10
  - RAW12
  - RAW14
  - Embedded data type
- Support for the following types of data interleaving:
  - Data type interleaving
  - Virtual channel interleaving

13.2.3 Register Description

To be defined

13.3 ISP

13.3.1 Introduction

K1 includes a high-performance Image Signal Processor (ISP) which supports simultaneous processing of up to two raw video streams, with a total processing capacity of 21M@30fps.

13.3.2 Features

- Support for both video and picture mode
- Processing of RAW sensor data and output YUV data to DRAM
- Hardware JPEG encoder/decoder (support up to 23M)
- Support for YUV, EXIF, JFIF format
- Auto-focus (AF), Auto-exposure (AE) and Auto-white balance (AWB)
- Face detection

- Digital zoom and panorama view
- Phase Detection Auto-focus (PDAF)
- Picture-in-Picture (PiP)
- Continuous video AF
- Hardware 3D denoise
- Multi-layer 2D YUV denoise
- Post-processing of lens shading correction
- Edge enhancement

Notes. To be highlighted the following limitations:

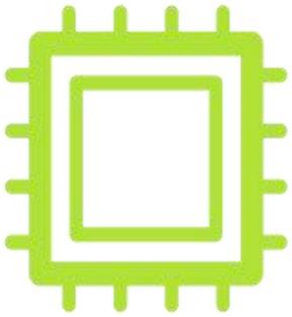
- The system supports dual-camera video stream processing (RAW). In the “4-Lane + 2-Lane + 2-Lane mode (triple sensor)” as per **Section 13.2.2**, one sensor must be a YUV input format source, and the write path should not use the MMU.
- When processing dual-camera video stream (RAW), the total input width of each channel should not exceed 4750 pixels. The combination of the instantaneous speed of the output pixel from both sensors must be less than “**ISP’s clock ÷ 6**”
- For video recording, the maximum width of the output video is 1920 pixels, regardless of the input resolution.
- For photo capture, the output image size can match the input resolution.

13.3.3 Register Description

To be defined

Chapter 14

RCPU Subsystem



Key Stone® K1 User Manual

14.1 Overview

The RCPU subsystem includes

- 256kB SRAM
- DMA
- Various APB peripherals (such as I2C, SSP, PWM, etc.) for the Sensor-Hub Subsystem

The power domain of the RCPU subsystem is independent from other modules, allowing the audio subsystem to function even when it enters low-power modes.

14.2 Features

14.2.1 General

The applications of the RCPU subsystem can access its memory and peripherals.

14.2.2 Main Peripherals

- **1 x I2C Controller:** Support for 100kHz (standard mode), 400kHz (fast mode), 3.4MHz (high-speed mode)
- **One Timer Group:** Inclusion of three counters, each with three match values
- **1x SPI Controller:** Support for data rate up to 24Mbps
- **2 x UART Controllers:** One UART supports auto-flow control
- **10 x PWM Controller**

14.2.3 Power & Clock Management

- Advanced clock gating, including software-controlled gating for peripherals and automatic clock gating for buses, fabrics and bridges
- Audio subsystem can operates in D0 (normal power on) mode and D1, D2 (low power) modes
- Four power mode for audio subsystem:
  - **ACTIVE**
  - **CLK_GATING**
  - **PLL_OFF**
  - **PWR_OFF**

14.3 Register Description

14.3.1 AUD_PMU Registers

Note. Base address = **0xC0A10000**

14.3.1.1 AUDIO_PMU_VOTE REGISTER

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0	Reserved for future use
3	Vote for audio PMU to enter PWR OFF mode	RW	0x1	1: Allow audio PMU to enter power-off mode. 0: Deny audio PMU from entering power-off mode.
2	Vote for audio PMU to enter PLL OFF mode	RW	0x1	1: Allow audio PMU to enter PLL-off mode. 0: Deny audio PMU from entering PLL-off mode.
1	Vote for audio PMU to enter low-power mode	RW	0x1	1: Allow audio PMU to enter low-power mode. 0: Deny audio PMU from entering low-power mode.
0	Reserved	RO	0	Reserved for future use

14.3.1.2 AUDIO_VOTE_FOR_MAIN_PMU REGISTER

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	Reserved	RO	0	Reserved for future use
6	Audio vote for AP AXI clock off	RW	0x1	1: Allow AP AXI clock off 0: Deny AP AXI clock off
5	Audio vote for DDR shutdown	RW	0x1	1: Allow DDR shutdown 0: Deny DDR shutdown
4	Reserved	RO	0	Reserved for future use
3	Audio vote for VCTCXO off	RW	0x1	1: Allow VCTCXO off. 0: Deny VCTCXO off.
2	Audio vote for main PMU sleep state	RW	0x1	1: Allow main PMU to enter sleep state. 0: Deny main PMU sleep state.
1	Audio vote for AP standby state	RW	0x1	1: Allow AP to go to standby state.

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Deny AP standby state.
0	Reserved	RO	0	Reserved for future use

14.3.1.3 AUDIO_WAKEUP_EN REGISTER

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0	Reserved for future use
7	Reserved	RO	0	Reserved for future use
6	Reserved	RO	0	Reserved for future use
5	Reserved	RO	0	Reserved for future use
4	timer_wkup_en	RW	0x0	Audio timer wake up enable 1: Enable 0: Disable
3	ap_wkup_en	RW	0x0	AP wake up enable 1: Enable 0: Disable
2	Reserved	RO	0	Reserved for future use
1	ipc_ap_wkup_en	RW	0x0	Audio to AP IPC wake up enable 1: Enable 0: Disable
0	shub_int_wkup_en	RW	0x0	Audio sensor hub wake up enable 1: Enable 0: Disable

14.3.1.4 AON_PER_CLK_RST_CTRL REGISTER

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0	Reserved for future use
1	ipc2ap clk enable	RW	0x0	1: Enable 0: Disable
0	aipc_ap_rstn	RW	0x0	1: Release reset 0: Reset

14.3.1.5 MCU_EXECUTION_CTRL REGISTER

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0	Reserved for future use
0	mcu_execution_ctrl	RW	0x0	1: Let MCU run 0: Halt MCU

14.3.1.6 AUDIO_BUS_CLK_DIV REGISTER

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	Reserved	RO	0	Reserved for future use
6:4	apb_clk_div	RW	0x3	0: rsvd 1: div2 2: div4 3: div8 4: div16 5: rsvd 6: rsvd 7: rsvd
3:2	Reserved	RO	0	Reserved for future use
1:0	axi_clk_div	RW	0x1	0: div1 1: div2 2: div4 3: div8

14.3.1.7 SHUB_GPO REGISTER

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0	Reserved for future use
7:0	shub_gpo	RW	0x0	Sensor hub GPIO output value

14.3.1.8 AUDIO_CTRL REGISTER

Offset: 0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	audio ADC test interrupt status	RW	0x0	Audio ADC test mode interrupt status 1: Test Done
30	Reserved	RO	0	Reserved for future use
29	hook key interrupt clear	RW	0x0	Clear the latched interrupt for hook key event 1: Clear Hook key interrupt status is stored in AUDIO_STATUS[21:14]
28:24	Hook key detection debuonce clock divider	RW	0xF	Specify the hook-key-detection debounce clock. The debounce clock is divided from 32KHz clock.
23	Class G right channel OCP interrupt clear	RW	0x0	Clear the latched interrupt for Class G right channel OCP. 1: Clear Interrupt status is stored in AUDIO_STATUS[13]
22	Class G left channel OCP interrupt clear	RW	0x0	Clear the latched interrupt for Class G left channel OCP. 1: Clear Interrupt status is stored in AUDIO_STATUS[12]
21	Class AB OCP interrupt clear	RW	0x0	Clear the latched interrupt for Class AB OCP. 1: Clear Interrupt status is stored in AUDIO_STATUS[11]
20:15	Reserved	RO	0	Reserved for future use
14	classge_shortpwr int enable	RW	0x0	0: Disable Class G short power interrupt. 1: Enable Class G short power interrupt.
13	classge_r_shortpwr_clr	RW	0x0	Clear the classg_r_short pwr interrupt. 1: Clear interrupt
12	classge_l_shortpwr_clr	RW	0x0	Clear the classg_l_short pwr interrupt. 1: Clear interrupt
11:4	Reserved	RO	0	Reserved for future use
3	plug wakeup interrupt clear	RW	0x0	Clear the latched plug in or plug out wakeup interrupt. 1: Clear interrupt Plug interrupt status is stored in AUDIO_STATUS[10:9]
2	auto OCP reset assertion	RW	0x1	Enable automatic generation of ocp_rst for class G and class AB. 1: Enable 0: Disable

Offset: 0xE0				
Bits	**Field**	**Type**	**Reset**	**Description**
1	analog test mode interrupt mask	RW	0x0	Enable interrupt when audio ADC test is done. 1: Enable interrupt
0	analog plugin polarity	RW	0x0	0: Plugin status high indicates a plugin event. 1: Plugin status low indicates a plugin event.

14.3.1.9 AUDIO_CTRL2 REGISTER

Offset: 0xE4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:19	OCP length for Class G	RW	0xff	Specifies the number of 32KHz cycles of Class G OCP assertion before ocp_rst is automatically generated.
18:14	OCP occurance times before interrupt for classg	RW	0x1f	Specifies the number of OCP assertions before an interrupt is generated for Class G.
13:5	OCP length for rcv	RW	0x7	Specifies the number of 32KHz cycles of RCV OCP assertion before ocp_rst is automatically generated.
4:0	OCP occurance times before interrupt for rcv	RW	0x2	Specifies the number of OCP assertions before an interrupt is generated for RCV.

14.3.1.10 AUD_DET_CLK_DIV REGISTER

Offset: 0xE8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:28	Reserved	RO	0	Reserved for future use
27:16	hok_deb_div	RW	0xFF	hok_deb_div
15:12	Reserved	RO	0	Reserved for future use
11:0	plg_deb_div	RW	0xFF	plg_deb_div

14.3.1.11 AUD_INT_MSK REGISTER

Offset: 0xF0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0	Reserved for future use
2	Interrupt mask for aud_ocpgr_int, aud_ocpgl_int, aud_ocpab_int	RW	0x0	0: Enable interrupt 1: Disable interrupt
1	Interrupt mask for plugin/plugout	RW	0x0	0: Enable interrupt 1: Disable interrupt
0	hook_key_int_msk	RW	0x0	0: Enable interrupt 1: Disable interrupt

14.3.2 AUD_MCUSYSCTRL Registers

Note. Base address = **0xC0880000**

14.3.2.1 SHUBSSP0_CLK_RES_CTRL REGISTER

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	Reserved	RO	0	Reserved for future use
18:8	shubssp0 clk div	RW	0x0	Clock divider for SHub SSP0: fclk = source_clk / (shubssp0_fclk_div + 1)
7:6	Reserved	RO	0	Reserved for future use
5:4	shubssp0 fclk sel	RW	0x0	Select the SHub SSP0 frequency clock: 00: clk_62m 01: clk_24p576m 10: clk_13m 11: clk_3p25m
3	Reserved	RO	0	Reserved for future use
2	shubssp0 pclk Enable/Disable	RW	0x0	Enable bit for SHub SSP0 peripheral clock (pclk) 0: Disable 1: Enable
1	shubssp0 fclk Enable/Disable	RW	0x0	Enable bit for SHub SSP0 frequency clock (fclk) 0: Disable 1: Enable
0	shubssp0 reset Enable/Disable	RW	0x0	Enable bit for SHub SSP0 reset 0: Disable 1: Enable

14.3.2.2 SHUBI2C0_CLK_RES_CTRL REGISTER

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	Reserved	RO	0	Reserved for future use
18:8	shubi2c0 clk div	RW	0x0	Clock divider for SHub I²C0: fclk= source_clk/(shubi2c0_fclk_div + 1)
7:6	Reserved	RO	0	Reserved for future use
5:4	shubi2c0 fclk sel	RW	0x0	Select the SHub I²C0 frequency clock: 00: clk_62m 01: clk_26m 10: clk_13m 11: clk_3p25m
3	Reserved	RO	0	Reserved for future use
2	shubi2c0 pclk Enable/Disable	RW	0x0	Enable bit for SHub I²C0 peripheral clock (pclk) 0: Disable 1: Enable
1	shubi2c0 fclk Enable/Disable	RW	0x0	Enable bit for SHub I²C0 frequency clock (fclk) 0: Disable 1: Enable
0	shubi2c0 reset Enable/Disable	RW	0x0	Enable bit for SHub I²C0 reset 0: Disable 1: Enable

14.3.2.3 UART1_CLK_RES_CTRL REGISTER

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	Reserved	RO	0	Reserved for future use
18:8	uart clk div	RW	0x0	UART clock divider: fclk = source_clk / (uart_fclk_div + 1)
7:6	Reserved	RO	0	Reserved for future use
5:4	uart fclk sel	RW	0x0	Select the UART frequency clock: 00: clk_62m 01: clk_26m 10: clk_13m 11: clk_3p25m
3	Reserved	RO	0	Reserved for future use
2	uart pclk Enable/Disable	RW	0x0	Enable bit for UART peripheral clock (pclk)

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Disable 1: Enable
1	uart fclk Enable/Disable	RW	0x0	Enable bit for UART frequency clock (fclk) 0: Disable 1: Enable
0	uart reset Enable/Disable	RW	0x0	Enable bit for UART reset 0: Disable 1: Enable

14.3.2.4 R_CAN_CLK_RES_CTRL REGISTER

Offset: 0x48				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	Reserved	RO	0	Reserved for future use
18:8	CAN clk div	RW	0x0	CAN clock divider: fclk= source_clk/(shubi2c1_fclk_div + 1)
7:6	Reserved	RO	0	Reserved for future use
5:4	CAN fclk sel	RW	0x0	Select the CAN frequency clock 00: clk_62m 01: clk_26m 10: clk_13m 11: clk_3p25m
3	Reserved	RO	0	Reserved for future use
2	CAN pclk Enable/Disable	RW	0x0	Enable bit for CAN peripheral clock (pclk) 0 = disable 1 = enable
1	CAN fclk Enable/Disable	RW	0x0	Enable bit for CAN frequency clock (fclk) 0: Disable 1: Enable
0	CAN reset Enable/Disable	RW	0x0	Enable bit for CAN reset 0: Disable 1: Enable

14.3.2.5 R_R_IR_CLK_RES_CTRL REGISTER

Offset: 0x4C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	Reserved	RO	0	Reserved for future use
18:8	Reserved	RO	0	Reserved for future use
7:6	Reserved	RO	0	Reserved for future use
5:4	Reserved	RO	0	Reserved for future use
3	Reserved	RO	0	Reserved for future use
2	R_IR pclk Enable/Disable	RW	0x0	Enable bit for R_IR peripheral clock (pclk) 0: Disable 1: Enable
1	Reserved	RO	0	Reserved for future use
0	R_IR reset Enable/Disable	RW	0x0	Enable bit for R_IR reset 0: Disable 1: Enable

14.3.2.6 DDR_REMAP_BASE REGISTER

Offset: 0xC0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	ddr_remap_base	RW	0x0	Base address for STAR core to access DDR. - Memory space: 0x30000000 - 0x3FFFFFFF - Remaps to DDR address range: ddr_remap_base + 0x0 to ddr_remap_base + 0x0FFFFFFF.

14.3.2.7 UART0_CLK_RES_CTRL REGISTER

Offset: 0xD8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	Reserved	RO	0	Reserved for future use
18:8	uart clk div	RW	0x0	UART Clock Divider: fclk= source_clk/(shubi2c1_fclk_div + 1)
7:6	Reserved	RO	0	Reserved for future use
5:4	uart fclk sel	RW	0x0	Select UART frequency clock (fclk) 00: clk_62m 01: clk_26m

Offset: 0xD8				
Bits	**Field**	**Type**	**Reset**	**Description**
				10: clk_13m 11: clk_3p25m
3	Reserved	RO	0	Reserved for future use
2	uart pclk Enable/Disable	RW	0x0	Enable bit for UART peripheral clock (pclk) 0: Disable 1: Enable
1	uart fclk Enable/Disable	RW	0x0	Enable bit for UART frequency clock (fclk) 0: Disable 1: Enable
0	uart reset Enable/Disable	RW	0x0	Enable bit for UART reset 0: Disable 1: Enable

14.3.3 AUD_AUDCLOCK Registers

Note. Base address = **0xC0882000**

14.3.3.1 AUDIO_CODEC_TX_RX_CLK_CTRL REGISTER

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	Reserved	RO	0	Reserved for future use
17:16	sspa_func_clk_source_sel	RW	0x0	Select SSPA function clock source 0: SSPA_FCLK_SRC = 245.76M 1: SSPA_FCLK_SRC = 24.576M
15	Reserved	RO	0	Reserved for future use
14:4	sspa_func_clk_div	RW	0x9f	SSPA function clock divider SSPA_FCLK = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	Reserved	RO	0	Reserved for future use
2	sspa_func_clk_en	RW	0x0	0: SSPA function clock gated 1: SSPA function clock open
1	tx_rx_bus_clk_en	RW	0x0	0: bus clock gated 1: bus clock open
0	tx_rx_sw_rstn	RW	0x0	Reset control for audio SSPA and ADMA 0: Reset 1: Release reset

14.3.3.2 AUDIO_DFE_CLK_CTRL REGISTER

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved	RO	0	Reserved for future use
5	dfe_sw_reset	RW	0x0	0: Reset 1: Release reset
4	dfe_func_clk_enable	RW	0x0	0: Disble 1: Enable
3	dac_sw_reset	RW	0x0	0: Reset 1: Release reset
2	dac_clk_inv_en	RW	0x0	DAC clock can use the original clock from analog or the invert clock 0: DAC uses the original clock 1: DAC uses the invert clock
1	adc_sw_reset	RW	0x0	0: Reset 1: Release reset
0	adc_clk_inv_en	RW	0x0	ADC clock can use the original clock from analog or the invert clock 0: ADC uses the original clock 1: ADC uses the invert clock

14.3.3.3 AUDIO_I2S1_TX_RX_CLK_CTRL REGISTER

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	Reserved	RO	0	Reserved for future use
17:16	sspa_func_clk_source_sel	RW	0x0	SSPA function clock source 0: SSPA_FCLK_SRC = 245.76M 1: SSPA_FCLK_SRC = 24.576M
15	Reserved	RO	0	Reserved for future use
14:4	sspa_func_clk_div	RW	0x9f	SSPA function clock divider SSPA_FCLK = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	Reserved	RO	0	Reserved for future use
2	sspa_func_clk_en	RW	0x0	0: SSPA function clock gated 1: SSPA function clock open
1	tx_rx_bus_clk_en	RW	0x0	0: Bus clock gated 1: Bus clock open

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
0	tx_rx_sw_rstn	RW	0x0	Reset control for audio SSPA and ADMA 0: Reset 1: Release reset

14.3.3.4 AUDIO_HDMI_CLK_CTRL REGISTER

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	Reserved	RO	0	Reserved for future use
17:16	sspa_func_clk_source_sel	RW	0x0	SSPA function clock source 0: SSPA_FCLK_SRC = 245.76M 1: SSPA_FCLK_SRC = 24.576M
15	Reserved	RO	0	Reserved for future use
14:4	sspa_func_clk_div	RW	0x9f	SSPA function clock divider SSPA_FCLK = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)
3	Reserved	RO	0	Reserved for future use
2	sspa_func_clk_en	RW	0x0	0: SSPA function clock gated 1: SSPA function clock open
1	tx_rx_bus_clk_en	RW	0x0	0: Bus clock gated 1: Bus clock open
0	tx_rx_sw_rstn	RW	0x0	Reset control for audio SSPA and ADMA 0: Reset 1: Release reset

14.3.3.5 AUDIO_I2S0_TX_RX_CLK_CTRL REGISTER

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31:18	Reserved	RO	0	Reserved for future use
17:16	sspa_func_clk_source_sel	RW	0x0	SSPA function clock source 0: SSPA_FCLK_SRC = 245.76M 1: SSPA_FCLK_SRC = 24.576M
15	Reserved	RO	0	Reserved for future use
14:4	sspa_func_clk_div	RW	0x9f	SSPA function clock divider SSPA_FCLK = SSPA_FCLK_SRC/(SSPA_FCLK_DIV+1)

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
3	Reserved	RO	0	Reserved for future use
2	sspa_func_clk_en	RW	0x0	0: SSPA function clock gated 1: SSPA function clock open
1	tx_rx_bus_clk_en	RW	0x0	0: Bus clock gated 1: Bus clock open
0	tx_rx_sw_rstn	RW	0x0	Reset control for audio SSPA and ADMA 0: Reset 1: Release reset

14.3.4 AUD_AHBDMA Registers

Note. Base address = **0xC0884000**

14.3.4.1 DMA_DCR REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:3	Reserved	RO	0	Reserved for future use
2	DMA Access Mode	RW	0x0	0: Privileged access 1: User access
1	DMA Soft Reset	W1C	0x0	0: No effect 1: Generates a 3-cycle reset pulse
0	DMA Enable	RW	0x0	0: DMA disable 1: DMA enable

14.3.4.2 DMA_SR REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15	CH15 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
14	CH14 Int pending	W1C	0x0	0: No interrupt

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
				1: Interrupt is pending Writing 1 clears the interrupt.
13	CH13 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
12	CH12 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
11	CH11 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
10	CH10 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
9	CH9 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
8	CH8 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
7	CH7 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
6	CH6 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
5	CH5 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
4	CH4 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
3	CH3 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
2	CH2 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.
1	CH1 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
				Writing 1 clears the interrupt.
0	CH0 Int pending	W1C	0x0	0: No interrupt 1: Interrupt is pending Writing 1 clears the interrupt.

14.3.4.3 DMA_DIMR REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15:0	Channel n Interrupt Mask	RW	0xFFFF	Channel n (15 ~ 0) 0: Enables interrupts 1: Disables interrupts

14.3.4.4 DMA_SARN REGISTER

Offset: 0x080+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Source Address	RW	0x0	This register holds the source address from where data is read during a DMA transfer. DMA does not perform **misaligned accesses**. The alignment behavior depends on the transfer size: - For **32-bit transfers**, the **lower two bits** of the address are ignored - For **8-bit transfers**, begin from the exact address specified **Note.** Software must ensure proper alignment, particularly in systems that do not support non-word-aligned accesses.

14.3.4.5 DMA_DARN REGISTER

Offset: 0x084+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**

Offset: 0x084+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Destination Address	RW	0x0	This register holds the destination address from where data is read during a DMA transfer. DMA does not perform **misaligned accesses**. The alignment behavior depends on the transfer size: - For **32-bit transfers**, the **lower two bits** of the address are ignored - For **8-bit transfers**, begin from the exact address specified **Note.** Software must ensure proper alignment, particularly in systems that do not support non-word-aligned accesses.

14.3.4.6 DMA_CNTRN REGISTER

Offset: 0x088+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	Count	RW	0x0	Contains the number of bytes of data to be transferred during a DMA cycle.

14.3.4.7 DMA_CCRN REGISTER

Offset: 0x08c+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:14	Reserved	RO	0	Reserved for future use
13:12	Destination Mode	RW	0x0	Selects the destination transfer mode 0x0: Memory 0x2: FIFO 0x1/0x3: Reserved
11:10	Source Mode	RW	0x0	Selects the destination transfer mode 0x0: Memory 0x2: FIFO 0x1/0x3: Reserved
9:8	Reserved	RO	0	Reserved for future use
7:6	Destination Size	RW	0x0	Selects the destination size of a data transfer. If the number of bytes to be written is less than the DSIZ setting, only that many bytes will be valid in the DMA

Offset: 0x08c+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
				write cycle to AHB. However, all DMA write cycles to the destination will be of DSIZ size. DMA always writes data as per DSIZ in all modes. 00: 32-bit destination port 01: 8-bit destination port 10: 16-bit destination port 11: Reserved
5:4	Source Size	RW	0x0	Selects the source size of a data transfer. If the number of bytes to be read is less than the SSIZ setting, only that many bytes will be used by the DMA. However, all DMA read cycles to the source will be of SSIZ size. 00: 32-bit source port 01: 8-bit source port 10: 16-bit source port 11: Reserved
3	Request Enable	RW	0x0	Enables or disables the DMA request signal. - When REN is set, DMA burst is initiated by the dma_req signal from the I/O FIFO. - When REN is cleared, DMA transfer is initiated by CEN (Channel Enable). 0: Disables the DMA request signal (when the peripheral asserts a DMA request, no DMA transfer is triggered). 1: Enables the DMA request signal (when the peripheral asserts a DMA request, a DMA transfer is triggered).
2:1	Reserved	RO	0	Reserved for future use
0	DMA Channel Enable	RW	0x0	**Note.** Disabling CEN during an ongoing burst on the AHB will stop the burst in the middle of the transfer. 0: Disables the DMA channel 1: Enables the DMA channel

14.3.4.8 DMA_RSSRN REGISTER

Offset: 0x090+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved	RO	0	Reserved for future use
5:0	Request Source Select	RW	0x0	Selects 1 of the 64 dma_req signals that initiates DMA transfer cycle for the channel. 000000: Select dma_req[0] 000001: Select dma_req [1] ...

Offset: 0x090+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
				011111: Select dma_req [31] .... 111111: Select dma_req [63]

14.3.4.9 DMA_BLRN REGISTER

Offset: 0x094+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved	RO	0	Reserved for future use
5:0	Burst Length	RW	0x0	Contains the number of data bytes that are transferred in a DMA burst. 000000: 64 bytes read follow 64 bytes write 000001: 1 byte read follow 1 byte write 000010: 2 bytes read follow 2 bytes write .... 111111: 63 bytes read follow 63 bytes write

14.3.4.10 DMA_TRSF_CNT REGISTER

Offset: 0x09c+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0	Reserved for future use
23:0	TRSF_CNT	RO	0x0	Indicates the number of bytes transferred for the channel

14.3.4.11 DMA_BTYPE REGISTER

Offset: 0x0a0+0x40*n				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0	Reserved for future use
1:0	Burst Type	RW	0x0	0x0: INCR 0x2: INCR4 0x3: INCR8 0x4: INCR16

14.3.5 PWM Registers

Note. Base address of PWMn (n = 1, 2, … 20) is **0xD401A000** with a stride of **0x400**

14.3.5.1 PWM_CRX REGISTER

PWM Control register. This register control the behavior of the PWM module, including:

- Shutdown response configuration
- Clock divisor settings by adjusting the input clock frequency to the PWM control unit which determines the frequency of the scaled counter clock.

Offset: 0x0+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	Reserved	RO	0x0	Reserved for future use.
8	PWM_OUTCNTen	RW	0x0	PWM Output Counter Register enable 0: Disable 1: Enable
7	Reserved	RO	0	Reserved for future use.
6	Pulse Width Modulator Shutdown Mode	RW	0x0	0: Graceful shutdown of PWM when the SoC stops the clock to the PWM. 1: Abrupt shutdown of PWM when the SoC stops the clocks to the PWM.
5:0	Prescale	RW	0x0	The scaled counter clock frequency is: Frequency = PSCLK_PWM / (PRESCALE + 1)

14.3.5.2 PWM_DCR REGISTER

PWM Duty Cycle register. This register configures the duty cycle of the corresponding PWM output signals (PWM_OUT).

Offset: 0x4+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:11	Reserved	RO	0x0	Reserved for future use.
10	Full Duty Cycle	RW	0x0	0: The PWM output signal (PWM_OUT) is determined by the <Duty Cycle of PWM_OUT> value. 1: The PWM output signal (PWM_OUT) is continuously asserted (i.e., it remains high).
9:0	Duty Cycle of PWM_OUT	RW	0x0	Defines the active high period of PWM_OUT: 0: The PWM output signal (PWM_OUT) is continuously de-asserted (i.e., it remains low). 1: The PWM output signal (PWM_OUT) is high for a specific duration with the calculation formula: High Time = (<PRESCALE> + 1) * (1 / 12.8 MHz), where

Offset: 0x4+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
				<PRESCALE> is a field in the PWM Control Registers **Note.** If <Full Duty Cycle> is set to **1**, this filed has no effect on the output of PWM.

14.3.5.3 PWM_PCR REGISTER

PWM Period Control register. This register is used to configure the cycle time of the corresponding PWM_OUT signals. When this register is set to zero (cleared), the PWM output signal (PWM_OUT) will stay in a high state.

Offset: 0x8+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0x0	Reserved for future use.
9:0	Period Value	RW	0x4	Defines the the cycle time of the PWM_OUT signal. - The value written to this field specifies the number of scaled clock cycles per PWM cycle, plus one. - Formula: PWM Cycle Time = (**Period Value+ 1**) × (1 / Scaled Clock Frequency) **Note.** Writing all zeros to this field, causes the PWM_OUT signal to remain high continuously.

14.3.5.4 PWM_OUTCNT REGISTER

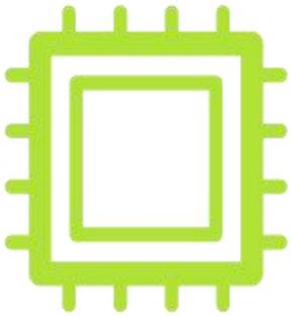
PWM Output Counter register.

Offset: 0x10+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	Counter Value	RW	0x0	Specifies the value of PWM out pulse number.

SPACEMIT

Chapter 15

High-Speed Interface System



Key Stone® K1 User Manual

15.1 USB

15.1.1 Introduction

K1 includes three USB ports as follows:

- A USB2.0 OTG Port
- A USB2.0 Host Only Port
- A USB3.0 Port with a USB2.0 DRD interface

15.1.2 Features

15.1.2.1 USB2.0 OTG Port

The USB2.0 OTG (On-The-Go) core supports both Host and Device functions in compliance with the USB 2.0 standard. Features are as follows:

- **Controller:**
  - Support for both USB2.0 Host and Device mode
  - Compliance with the USB2.0 standard
  - Support for USB2.0 High Speed (480Mb/s) and Full Speed (12Mb/s) for both Host and Device modes
  - Support for USB2.0 Low Speed (1.5Mb/s) for Host Only Mode
  - Host controller registers and data structures are compliant with the Intel EHCI specification
  - Device controller registers and data structures are implemented as extensions to the EHCI programming interface
  - Bus interface is compliant with AMBA-AHB specification
- **Communication Interface:**
  - Implementation of a UTMI+ interface to communicate with a USB2.0 PHY
- **Protocols:**
  - Support for the Session Request Protocol (SRP)
  - Support for the Host Negotiation Protocol (HNP)
- **Channel & Endpoint:**
  - Support for up to 16 host channels
  - In Device mode, support for 16 IN and 16 OUT endpoints, where
  - 16KB buffer is for transmitting data
  - 2KB buffer is for receiving data

15.1.2.2 USB2.0 Host Only Port

The USB2.0 Host core supports

- High-Speed (HS) host function

- Full-Speed (FS) host function
- Low-Speed (LS) host function

in compliance with the USB 2.0 standard. And it is Host-Only, i.e. cannot act as a device. Features are as follows:

- **Controller:**
  - Support for USB2.0 HS, USB2.0 FS, USB2.0 LS Host modes
  - Compliance with the USB2.0 standard
  - Support for High Speed (480Mb/s), Full Speed (12Mb/s), Low Speed (1.5Mb/s) for Host mode
  - Host controller registers and data structures are compliant with the Intel EHCI specification
  - Bus interface is compliant with AMBA-AHB specification
- **Communication Interface:**
  - Implementation of a UTMI+ interface to communicate with a USB2.0 PHY
- **Channel Support:**
  - Support for up to 16 host channels

15.1.2.3 USB3.0 Port with a USB2.0 DRD interface

The USB3.0 DRD (Dual-Role Device) core supports both Host and Device functions for USB 3.0 and USB 2.0 standards. Features are as follows:

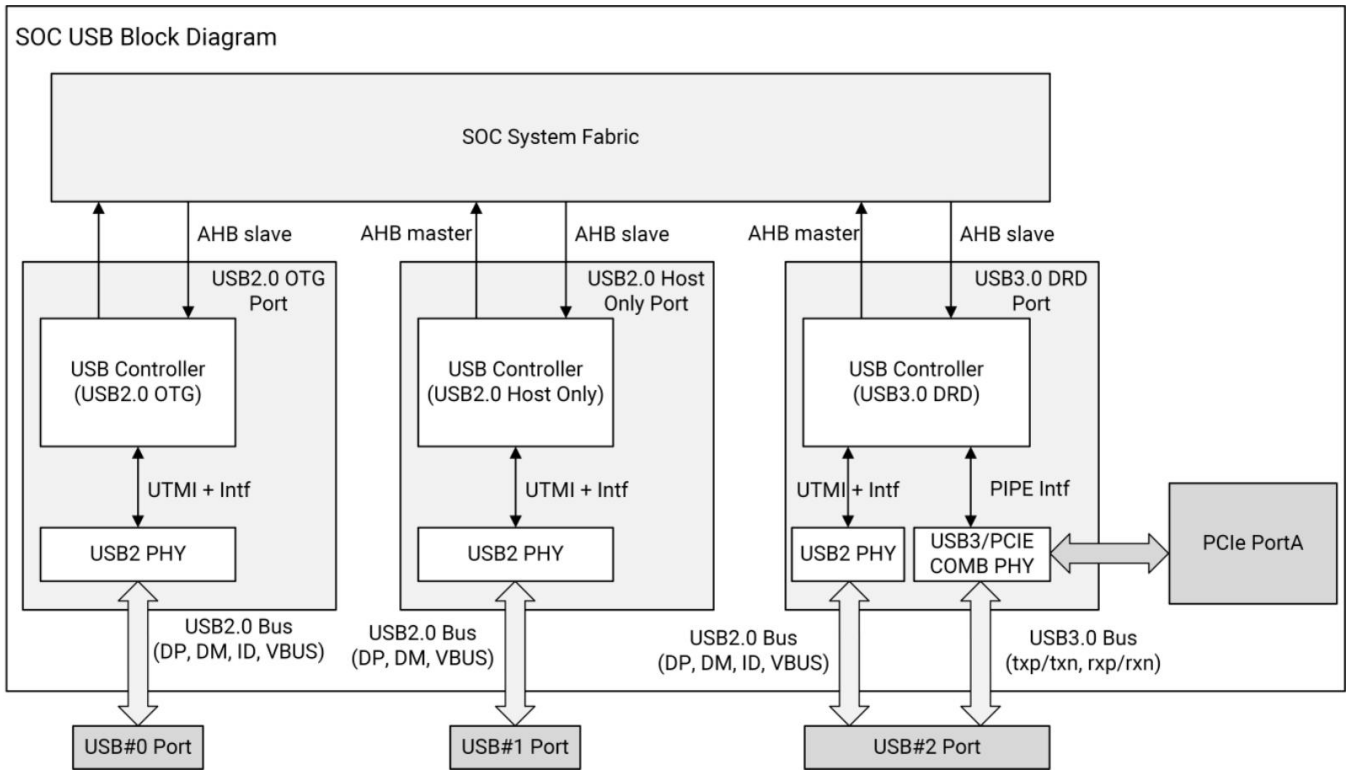
- **Controller:**
  - Support for both USB3.0 Host and Device modes
  - Support for both USB2.0 Host and Device modes
  - Compliance with both the USB3.0 and USB2.0 standards
  - Support for USB3.0 (Super Speed) and USB2.0 Host and Device mode
  - USB3.0 Host Controller registers and data structures are compliant with the Intel xHCI specification
  - USB3.0 Device controller registers and data structures are self-defined requiring software configuration
  - Support for one USB3.0 port and one USB2.0 port
  - Support for High Speed (480Mb/s) and Full Speed (12Mb/s) for Host and Device mode
  - Support for Low Speed (1.5Mb/s) for Host-Only mode
- **Communication Interface:**
  - Use of a PIPE3 (125MHz) interface for USB3.0 PHY
  - Use of a UTMI+ (30/60MHz) interface for USB2.0 PHY
- **Clock Domains:**
  - PIPE3 PHY (125MHz)
  - UTMI+ PHY (30/60MHz)
  - MAC (nominal 125MHz)
  - BUS clock domain
  - RAM clock domain

- **System & Power Management:**
  - Internal DMA controller
  - Support for USB2.0 suspend mode
  - Support for U1/U2/U3 low-power modes for USB3.0
- **Endpoint & Memory Support:**
  - Support for up to 32 endpoints in Device mode
  - Flexible endpoint FIFO sizes (not limited to powers of 2) allowing the use of contiguous memory locations
  - Descriptor caching and data pre-fetching for improving performance in high-latency systems
- **Additional Features:**
  - Software-controlled standard USB commands (USB SETUP commands forwarded to application for decoding)
  - Hardware-level error handling for USB bus and packet-level issues
  - Support for interrupts

15.1.3 Block Diagram

The architecture of the USB port set is depicted below, where

- **USB#0 Port** = USB2.0 OTG Port
- **USB#1 Port** = USB2.0 Host-Only Port
- **USB#2 Port** = USB3.0 Port with a USB2.0 DRD interface



15.1.4 Register Description

Note. Base Address = **0xC0A10000**

15.1.4.1 USB3_CTRL_CLK_CFG REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0	Reserved for future use
9	use_vbus_valid_ext	RW	0x0	1: Routes the VBUS detection signal from GPIO to the pipe_PowerPresent signal.
8	bigendian_gs	RW	0x0	Signals connect to USB3 Controller
7:6	Reserved	RO	0	Reserved for future use
5:0	fladj_30mhz_reg	RW	0x20	Signals connect to USB3 Controller

15.1.4.2 USB3_CTRL_MISC_CFG_0 REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	gp_in	RW	0x0	Signals connect to USB3 Controller
15:5	Reserved	RO	0	Reserved for future use
4	pme_en	RW	0x0	Signals connect to USB3 Controller
3:0	bus_filter_bypass	RW	0x0	Signals connect to USB3 Controller

15.1.4.3 USB3_CTRL_HOST_CFG REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	spare	RW	0x0	Reserved for future use

15.1.4.4 USB3_CTRL_CLK_CFG REGISTER

Offset: 0xc				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0	Reserved for future use
9	host_msi_enable	RW	0x0	
8	host_port_power_control_present	RW	0x0	
7:6	Reserved	RO	0	Reserved for future use
5	host_u3_port_disable	RW	0x0	
4	host_u2_port_disable	RW	0x0	Reserved for future use
3:2	hub_port_perm_attach	RW	0x0	Signals connect to USB3 Controller
1:0	hub_port_overcurrent	RW	0x0	Signals connect to USB3 Controller

15.1.4.5 USB3_CTRL_CLK_CFG REGISTER

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31	bus_err_int	RO	0x0	Bus Error Interrupt valid
30:22	Reserved	RO	0	Reserved for future use
21	gsts_buserraddvld_sync	RO	0x0	Bus access error
20	host_system_err_sync	RO	0x0	Host system error
19:16	usb3_buserr_sts	RO	0x0	Bit 0: host_system_error rise edge interrupt status Bit 1: host_system_error fall edge interrupt status Bit 2: gsts_buserraddvld rise edge interrupt status Bit 3: gsts_buserraddvld fall edge interrupt status
15:8	Reserved	RO	0	Reserved for future use
7:4	usb3_buserr_int_mask	RW	0x0	Bit 0: host_system_error rise edge interrupt enable Bit 1: host_system_error fall edge interrupt enable Bit 2: gsts_buserraddvld rise edge interrupt enable Bit 3: gsts_buserraddvld fall edge interrupt enable
3:1	Reserved	RO	0	Reserved for future use
0	usb3_buserr_int_clr	RW	0x0	Write 1 to clear bus_err_int It will be clear to '0' by hardware.

15.1.4.6 P_ADDR_PUMON0 REGISTER

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_trigger_mask[31:0]	RW	0	Mask for signal capture in Monitor Module

15.1.4.7 P_ADDR_PUMON1 REGISTER

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_trigger_mask[63:32]	RW	0	Mask for signal capture in Monitor Module

15.1.4.8 P_ADDR_PUMON2 REGISTER

Offset: 0x2c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_trigger_trigger_pattern[31:0]	RW	0	Trigger pattern

15.1.4.9 P_ADDR_PUMON3 REGISTER

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_trigger_trigger_pattern[63:32]	RW	0	Trigger pattern

15.1.4.10 P_ADDR_PUMON4 REGISTER

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0	Reserved for future use
9	pumon_trigger_done	RW	0	
8	pumon_trigger_ing	RW	0	
7	pumon_trigger_force_stop	RW	0	
6	pumon_trigger_start	RW	0	
5	pumon_trigger_mode	RW	0	

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
4:0	pumon_sample_sel	RW	0	

15.1.4.11 P_ADDR_RO0 REGISTER

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_trigger_signals_ro[31:0]	RO	0x0	Debug signal from usb3 controller

15.1.4.12 P_ADDR_RO1 REGISTER

Offset: 0x3c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_trigger_signals_ro[63:32]	RO	0x0	Debug signal from usb3 controller

15.1.4.13 P_ADDR_RO2 REGISTER

Offset: 0x40				
Bits	**Field**	**Type**	**Reset**	**Description**
31:00	pumon_monitor_ro[31:0]	RW	0x0	Debug signal from PUPHY

15.1.4.14 P_ADDR_RO3 REGISTER

Offset: 0x44				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_monitor_ro[63:32]	RO	0x0	Debug signal from PUPHY

15.1.4.15 P_ADDR_RO4 REGISTER

Offset: 0x48				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	pumon_monitor_ro[95:64]	RO	0x0	Pipe rx data

15.1.4.16 P_ADDR_RO5 REGISTER

Offset: 0x4c				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0	Reserved for future use
15:0	pumon_monitor_ro[112:96]	RO	0x0	Debug signal from USB2 PHY

15.1.4.17 USB3_BW_CALC_CTRL REGISTER

Offset: 0x50				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	bw_calc_ctrl	RW	0x8000FFFF	Dedicated > **Note.** Deprecated

15.1.4.18 USB3_CTRL_MISC_ST REGISTER

Offset: 0x80				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	gp_out	RO	0x0	Signals connect to USB3 Controller Output signals
15:8	Reserved	RO	0	Reserved for future use
7:6	operational_mode	RO	0x0	Signals connect to USB3 Controller Output signals
5	pme_generation	RO	0x0	Signals connect to USB3 Controller Output signals
4	host_system_err	RO	0x0	Signals connect to USB3 Controller Output signals
3	Reserved	RO	0	Reserved for future use
2:0	clk_gate_ctrl	RO	0x0	Signals connect to USB3 Controller Output signals

15.1.4.19 USB3_CTRL_HOST_ST REGISTER

Offset: 0x84				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	Reserved	RO	0	Reserved for future use
11:0	host_current_belt	RO	0x0	Signals connect to USB3 Controller Output signals

15.1.4.20 IP_REVISION REGISTER

Offset: 0x100				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	phy_ip_revision[15:0]	RO	0xb112	Read ip_revision[15:0]
15:0	ip_revision[15:0]	RO	0x0252	Read phy_ip_revision[15:0]

15.1.4.21 USB_CTL REGISTER

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
31:15	Reserved	RO	0	Reserved for future use
14	USB2 phy reg reset	RW	0x0	Soft reset for USB2 PHY registers. - Set **1** to reset the USB2 PHY registers - Clear to **0** to release the reset Note: This reset is for backup purposes only.
13	PHY suspendm enable	RW	0x1	Controls whether the **suspendm signal** from the USB2 controller affects the PHY suspension. **1**: PHY suspension depends on the controller's suspendm signal. - If the controller does not indicate PHY suspension, the PHY will not suspend. - If the controller indicates PHY suspension, the PHY may suspend based on its internal logic. **0**: The USB2 PHY is **not affected** by the controller's suspendm signal.
12	Reserved	RO	0	Reserved for future use
11:10	reg_opmode	RW	0x0	Provides **register control** for the PHY interface's **op_mode**.
9:8	reg_xcvr_select	RW	0x0	Provides **register control** for the PHY interface's **xcvr_select**.
7	reg_term_select	RW	0x0	Provides **register control** for the PHY interface's **term_select**.
6	reg_sel	RW	0x0	**1**: Bits [11:7] (reg_opmode, reg_xcvr_select, reg_term_select) take effect and control the PHY interface. **0**: The controller controls the PHY interface.
5:4	vbusvalid_ctl	RW	0x0	Controls the **VBUS_ON** signal for the PHY. - **Bit 4**: **1**: Use **Bit 5** to control PHY VBUS_ON. **0**: Use the **VBUSVALID** signal from the USB2

Offset: 0x104				
Bits	**Field**	**Type**	**Reset**	**Description**
				PHY to control VBUS_ON. - **Bit 5**: When **Bit 4** is **1**, this bit directly controls the VBUS_ON signal.
3	otg_sel	RW	0x1	Controls the **OTG PHY owner**.
2:0	Reserved	RO	0	Reserved for future use

15.1.4.22 USB_VBUS_REG REGISTER

Offset: 0x114				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb_vbus_reg	RW	0x0	**Note.** Deprecated

15.1.4.23 USB2_CTRL_STATUS0 REGISTER

Offset: 0x140				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status0	RO	0x0	Provides bits [31:0] of the USB2 controller debug signal.

15.1.4.24 USB2_CTRL_STATUS1 REGISTER

Offset: 0x144				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status1	RO	0x0	Provides bits [63:32] of the USB2 controller debug signal.

15.1.4.25 USB2_CTRL_STATUS2 REGISTER

Offset: 0x148				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status2	RO	0x0	Provides bits [66:64] of the USB2 controller debug signal.

15.1.4.26 USB2_CTRL_STATUS3 REGISTER

Offset: 0x14C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status3	RO	0x0	Provides bits [31:0] of the USB controller's logic_analyzer_trace signal.

15.1.4.27 USB2_CTRL_STATUS4 REGISTER

Offset: 0x150				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status4	RO	0x0	Provides bits [63:32] of the USB controller's logic_analyzer_trace signal.

15.1.4.28 USB2_CTRL_STATUS5 REGISTER

Offset: 0x154				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status5	RO	0x0	Provides USB2 PHY monitor signals.

15.1.4.29 USB2_CTRL_STATUS6 REGISTER

Offset: 0x158				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status6	RO	0x0	Reserved

15.1.4.30 USB2_CTRL_STATUS7 REGISTER

Offset: 0x15C				
Bits	Field	Type	Reset	Description
31:0	usb2_ctrl_status7	RO	0x0	Reserved

15.1.4.31 USB2_CTRL_STATUS8 REGISTER

Offset: 0x160				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	usb2_ctrl_status8	RO	0x0	Reserved

15.1.4.32 USB2_CTRL_STATUS9 REGISTER

Offset: 0x164				
Bits	**Field**	**Type**	**Reset**	**Description**
31:13	Reserved	RO	0	Reserved for future use
12	chep_last_trans	RO	0x0	chep_last_trans signal
11:8	chep_number	RO	0x0	chep_number signal
7:5	Reserved	RO	0	Reserved for future use
4	int_dma_done	RO	0x0	int_dma_done signal
3	int_dma_req	RO	0x0	int_dma_req signal
2	sof_sent_rcvd_tgl	RO	0x0	sof_sent_rcvd_tgl signal
1	sof_toggle_out	RO	0x0	sof_toggle_out signal
0	interrupt	RO	0x0	interrupt signal

15.2 PCIe

15.2.1 Introduction

K1 implements three PCIe Dual-Mode ports which can be configured as either Root Complex (RC) or Endpoint (EP) device.

All ports support Gen2 with a data transfer speed of 5GT/s per lane. However, one port supports one lane only and two ports support two lanes each.

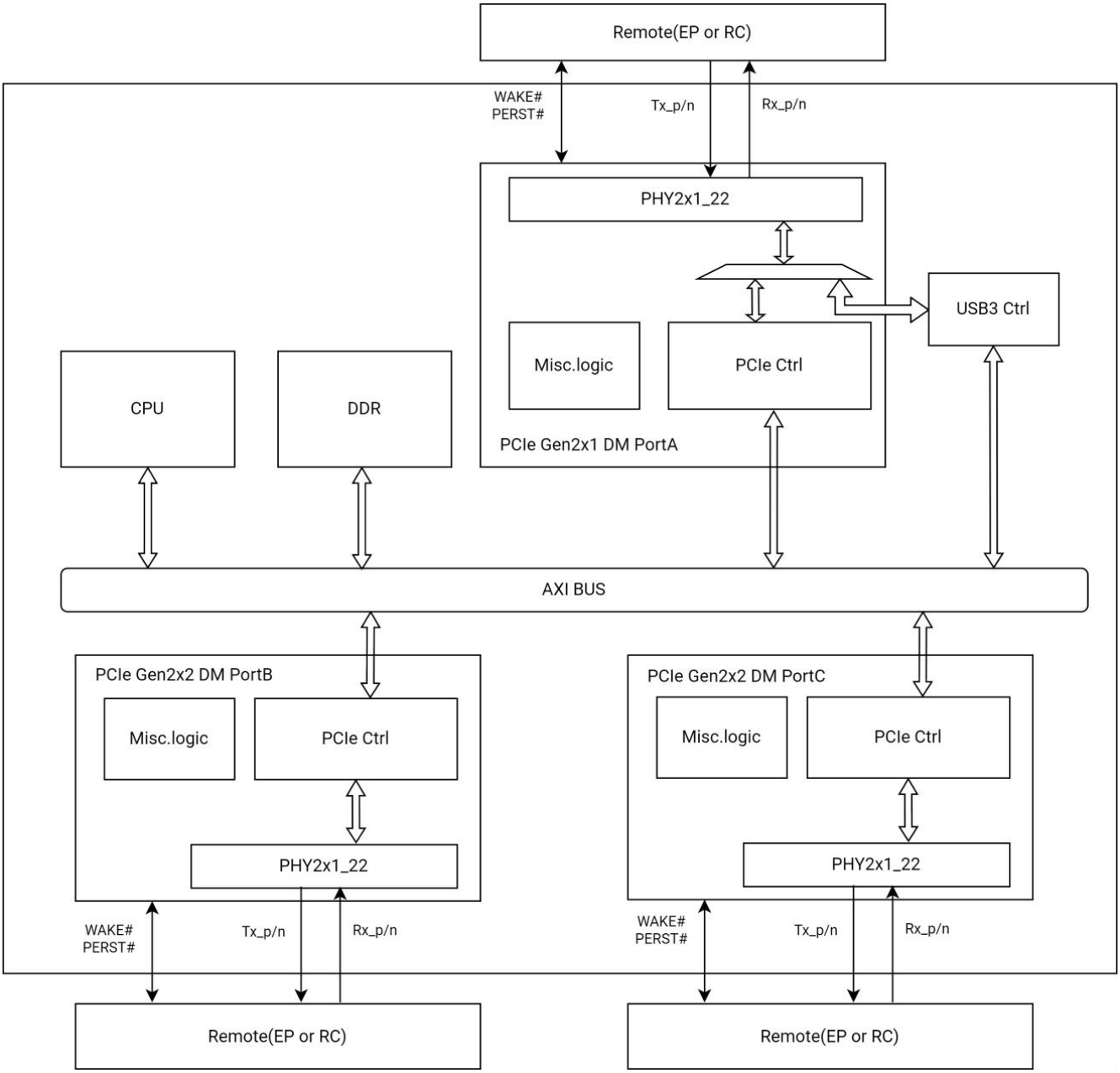
15.2.2 Features

- Support for Dual-Mode, programmable as either Complex (RC) or Endpoint (EP) device
- Support for all non-optional features of the PCI Express Base Specification - Revision 5.0 - Version 1.0 (limited to Gen2 speed scope)
- Support for Internal Address Translation Unit (iATU) with 8 entries for outbound and 8 entries for inbound traffic

- Support for Embedded DMA with Hardware Flow Control which includes 4 write channels and 4 read channels
- Support for ECRC generation and check
- Support for max payload size up to 256 bytes
- Support for Automatic Lane Flip and Reversal
- Support for L0 and L1 Power State of Active State Link PM
- Support for Latency Tolerance Reporting (LTR)
- Support for only Virtual Channel 0
- Support for ID Based Ordering (IDO)
- Support for Completion Timeout Ranges
- Support for Separate Reference Clock With Independent Spread (SRIS)
- Support for up to 64 outbound Non-Post Requests
- Support for up to 32 outstanding AXI slave Non-Post requests
- Support for only Function 0 with 6 size-programmable BARs in EP Mode
- Support for MSI Capability in EP Mode
- Support for Integrated MSI Reception Module in RC Mode

15.2.3 Functional Description

The architecture of the PCIe Dual-Mode port set is depicted below.



As can be seen, there are

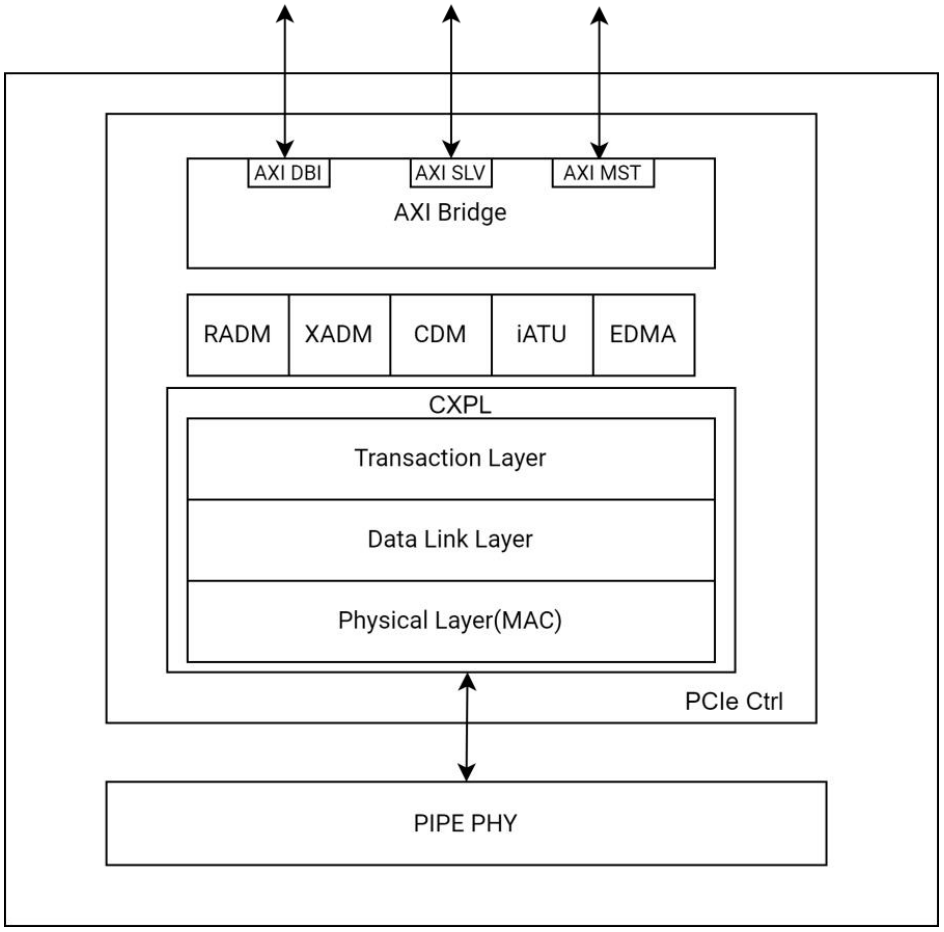
- One PCIe Gen2x1 Dual-Mode port (hereafter Port A)
- Two PCIe Gen2x2 Dual-Mode ports (hereafter Port B and Port C)

as said previously, and all them consists of

- A **controller** integrated into SoC via **3 AXI ports** which are designed as
  - **AXI Master Port** to manages inbound traffic (i.e. data coming into the system) either from a remote device or through the PCIe controller's internal DMA, allowing the access to DDR memory for transferring data both to and from the remote device
  - **AXI Data Slave Port** to allows the local CPU accessing itself for outbound traffic
  - **AXI DBI Slave Port** to be used for the PCIe controller's configuration interface

- A **PHY** complied with PIPE 3 specification and distinguished in
  - **Phy2x1_22** which
    - Supports Gen2 with one lane (x1)
    - Is built using a 22nm process
    - Is shared between Port A and USB3 controller but not simultaneously, i.e. both Port A and USB3 controller can operate but not at the same time
  - **Phy2x2_22** which
    - Supports Gen2 with two lanes (x2)
    - Is built using a 22nm process
    - Comes for Port B and Port C distinctly, i.e. Port B and Port C have their own dedicated PHY
- A **miscellaneous logic**, in particular **chip I/O with remote links partner** as follows:
  - **Differential Data Signals**: Rx_p/n, Tx_p/n (x2 lanes for Port B/C, x1 lane for Port A)
  - **Reference Clock Signals**: refclk_p/n (support for both input and output mode)
  - **Warm Reset Signal**: PERST# (input in EP mode, output in RC mode)
  - **Wake-Up signal**: WAKE# (output in EP mode, input in RC mode)

Each PCIe port includes a PHY and a controller. The controller is divided into key functional blocks as depicted below.



Details are provided in the following subsections.

15.2.3.1 AXI Bridge

The AXI bridge module acts as a bridge between the standard AXI interfaces and the PCIe controller native interfaces. The bridge supports up to 3 AXI interfaces as follows:

- **Master Interface:** Allows a remote PCIe device to read/write to an AXI slave connected to the bridge
- **Link Slave Interface:** Enables an AXI master to read/write to a remote PCIe device through the bridge
- **DBI Slave Interface:** Allows an AXI master to access the PCIe controller registers

The bridge facilitates communication between AXI-embedded systems and PCIe devices, acting as either a Root Port or an Endpoint.

15.2.3.2 CXPL

Common Express Port Logic (CXPL) Module implements the basic functionality for

- **PCIe Physical Layer**
- **PCIe Link Layer**
- **PCIe Transaction Layer**

The key features are as follows:

- Implementation of the Transaction Layer logic, Data Link Layer logic and the MAC portion of the Physical Layer
- Inclusion of the Link Training and Status State Machine (LTSSM) for link initialization and management
- Connection to the external PHY via the PIPE (PHY Interface for PCI Express)

CXPL provides the core logic for PCIe communication and link management.

15.2.3.3 XADM

Transmit Application-Dependent Module (XADM) handles application-specific functionality for packet transmission in the Transaction Layer. The key features are as follows:

- **TLP Arbitration**
- **TLP Formation**
- **Flow Control (FC) Credit checking**
- **Cut-Through Architecture:** Transmits packets without buffering (except for the retry buffer)
- **Target Completion Lookup Table:** Stores TLP header information for received requests

XADM manages the transmit path for PCIe packets.

15.2.3.4 RADM

Receive Application-Dependent Module (RADM) handles application-specific functionality for packet reception in the Transaction Layer. The key features are as follows:

- **TLP Sorting/Filtering:** Filters and routes received TLPs based on configurable rules
- **TLP Buffering/Queuing:** Buffers and queues incoming TLPs
- **TLP Routing:** Routes the received TLPs to the appropriate receive interfaces
- **Receive Completion Lookup Table (LUT):** Tracks completions and monitors completion timeouts for non-posted requests

RADM Manages the receive path for PCIe packets and ensures reliable completion tracking.

15.2.3.5 CDM

Configuration-Dependent Module (CDM) implements the configuration space for the PCIe controller. The key features are as follows:

- **Standard PCIe Configuration Space:** Compliant with PCIe specifications
- **Controller-Specific Register Space:** Includes Port Logic Registers for custom configurations

CDM Provides access to PCIe configuration and controller-specific settings.

15.2.3.6 iATU

The Internal Address Translation Unit (iATU) implements address translation for TLPs. The key features are as follows:

- **Address Translation:** Replaces TLP address and header fields with translated addresses
- **Type Translation:** Supports translation of address types
- **Outbound/Inbound Translation:** Maps application memory space to PCI memory space

iATU enables flexible address mapping between local and remote memory spaces. Without iATU, memory addresses pass through unchanged. Custom mappings can be configured by using software.

15.2.3.7 EDMA

The embedded DMA controller (EDMA) offloads data transfer tasks from the CPU and supports efficient transfers of large blocks of data with minimal CPU intervention, leaving the CPU free to perform other tasks. The key features are as follows:

- **Channels:** 4 read channels and 4 write channels, where
  - **DMA Write:** Transfers data from local (application) memory to remote (link partner) memory
  - **DMA Read:** Transfers data from remote (link partner) memory to local (application) memory
- **Full Duplex Operation:** Supports simultaneous read and write transfers, and in parallel with normal (non-DMA) traffic
- **Interrupts:** Notifies the CPU upon transfer completion or error. The DMA can
  - Interrupt the local CPU
  - Send an Interrupt Memory Write (IMWr) to the remote CPU

- **Configuration & Programming:** The DMA is highly configurable and can be programmed through
  - Local DBI (Designated Bus Interface)
  - PCIe wire (for remote configuration)
- **Linked List (LL) Mode:** Transfers multiple blocks of data using a pre-programmed list of descriptors. This mode enables efficient data movement with minimal CPU intervention and reduces the need to program the DMA repeatedly for multiple block transfers. To be highlighted:
  - **Linked List Structure**
    - Each transfer block is defined by a Linked List (LL) element, also called a descriptor, in local memory.
    - Each data element in the transfer list can transfer up to 4 GB of data.
  - **In LL mode**, the DMA:
    - Fetches the channel context (transfer control information)
    - Executes transfers based on the DMA elements programmed in local memory

15.3 EMAC

15.3.1 Introduction

K1 features a EMAC core which includes the essential protocol requirements for the operation of 10/100/1000 Mbps Ethernet/IEEE 802.3-2012 compliant node.

On the system side, the EMAC core features a 64-bit AXI Master Interface and a 32-bit AXI Target (Slave) Interface for seamless integration with the AXI Bus.

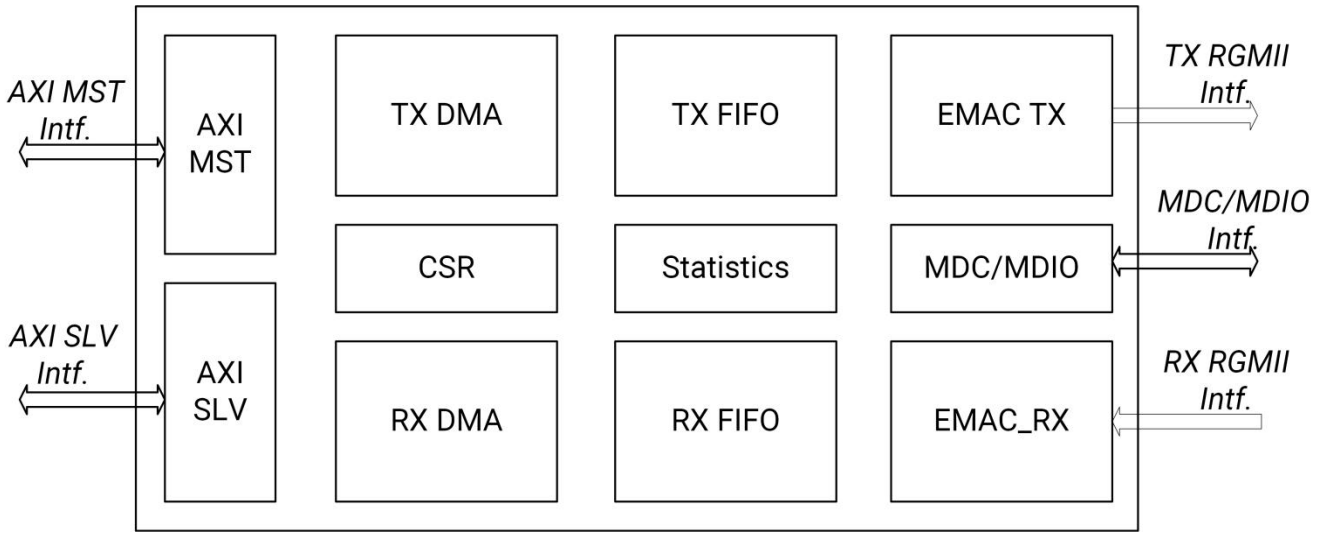
The EMAC core can operate at 10 Mbps, 100 Mbps (Fast Ethernet) or 1000 Mbps (Gigabit Ethernet). Additionally, it includes a powerful 64-bit Scatter-Gather DMA to transfer packets between HOST Memory and Internal FIFOs to achieve high performance.

15.3.2 Features

- Capability of handling transmit/receive data encapsulation functions, including Framing (frame boundary delimitation, frame synchronization) and Error Detection (physical medium transmission errors)
- Media access management with medium allocation (collision avoidance) and contention resolution (collision handling) in Half-Duplex Mode of operation at speeds of 10/100 Mbps
- Retransmission of frames that result in Collision in Half-Duplex mode
- Support for Flow Control functions in Full Duplex mode by decoding PAUSE control frames, disabling the transmitter and generating PAUSE control Frames
- Support for a 4-bit data path based RGMII Interface to connect with RGMII-based PHY
- Support for Management Interface by generating management frames on the MDC/MDIO pins to communicate with external PHY devices
- Bus mastering on the AXI interface to transfer packets between the HOST memory and the internal FIFOs using 64-bit transfer mode
- Automatic transfer of packets between the HOST memory and internal FIFOs (based on descriptors) to minimize CPU overhead

15.3.3 Block Diagram

The micro-architecture of EMAC unit is depicted below.



A brief description of each module is provided below.

- **EMAC_TX**: This module implements the Frame Transmit State Machine of the CSMA/CD protocol by transmitting frames onto RGMII Interface based on Speed.
- **EMAC_RX**: This module implements the Frame Receive State Machine of the CSMA/CD Protocol by receiving frames from RGMII Interface and performing field extraction and error checking.
- **MDC/MDIO Controller**: This module generates MDIO Frames to talk to external PHY Devices.
- **STATISTICS**: This module maintains various counters to perform Statistics for both Transmit and Receive operation.
- **EMAC_RXFIFO**: This module has a 4KB FIFO to store the received frames before they are transferred to the Host memory. This module also has additional control logic to flush error packets from the FIFO.
- **EMAC_TXFIFO**: This module has a 4KB FIFO to store the Frames that are received from the Host memory before the frames are transferred onto Ethernet Interface. This module also has additional FIFO control logic that helps in frame retransmission during collisions.
- **TXDMA**: This module implements the Transmit DMA to transfer the frames from the Host Memory to the Transmit FIFO using the Transmit Descriptors.
- **RXDMA**: This module implements the Receive DMA to transfer the frames from the Internal FIFO to the Host Memory using the Receive Descriptors
- **Registers**: This module implements the Control and Status Registers to control the operation of Receive and Transmit DMA and also the Receive and Transmit operations of the Ethernet Interface. This module also provides access to the Statistics Counters and provides interrupt/status information.
- **AXI Master Interface**: This module provides the AXI Master functionality to generate transactions on the AXI Bus. The transactions are generated based on the requests from the Transmit/Receive DMA's.
- **AXI Target Interface**: This module provides the AXI Target functionality to the AXI Host (CPU). This interface is used to access all the DMA/MAC registers in the Registers Module.

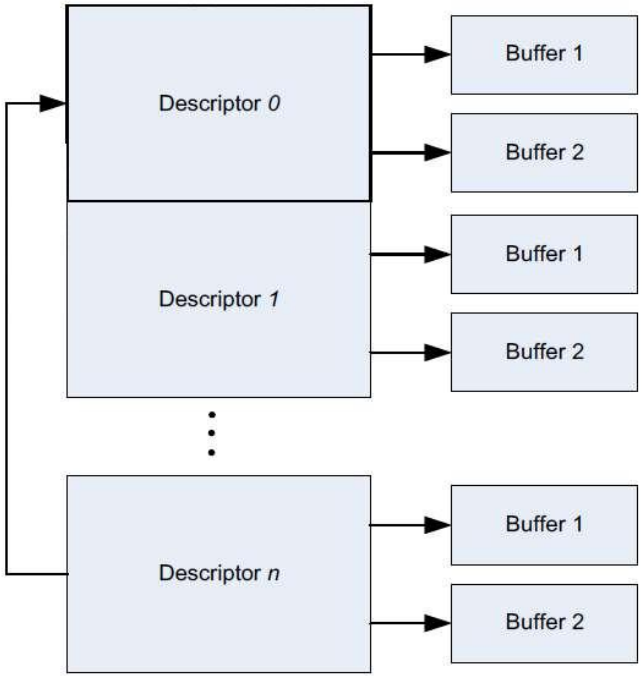
15.3.4 Programming Model

The EMAC Core transfers received data frames/packets from the RMII/RGMII interface to the receive buffers in the host memory. Similarly, it transmits data frames/packets from the transmit buffers in the host memory to the RMII/RGMII interface. Descriptors, which reside in the host memory, act as pointers to these buffers. The receive and transmit FIFOs inside the EMAC Core serve as temporary storage for frame transmission and reception.

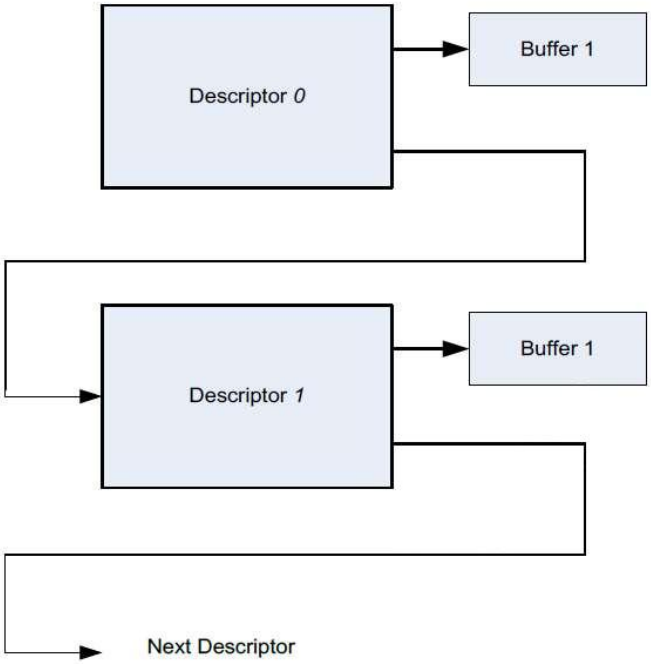
Two descriptor lists are maintained, one for receive and one for transmit. The base address of each list is written into the DMA Base Address Registers. The descriptor list is forward-linked, either implicitly or explicitly. The last descriptor can point back to the first entry to create a ring structure. Explicit chaining of descriptors is achieved by setting the Second Address Chained bits in both transmit and receive descriptors. The descriptor list resides in the host memory address space.

Each descriptor can point to a maximum of two buffers, allowing the use of two physically non-contiguous buffers in memory. A data buffer can contain either an entire frame or part of a frame but cannot exceed the size of a single frame. Buffers contain only data, while buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers and can be enabled or disabled as needed. All data buffers reside in the host physical memory space.

Below is depicted the descriptor ring and its structure. Each descriptor can support up to two buffers, and the next descriptor follows the current descriptor (optionally with a programmable skip length). The last descriptor can be chained back to the first one to form a ring.



Below is depicted the descriptor chain structure. In this configuration, each descriptor points to only one buffer. The address of the next descriptor is included as part of the current descriptor, enabling the creation of a descriptor chaining (linked list) structure.



15.3.5 Register Description

The EMAC registers are used to control the operation of the EMAC core and to read status/interrupt information from it. The EMAC core decodes all 16 bits of the AHB/AXI (Slave) target address to access the registers. These registers are 32-bit word-aligned and must be accessed using 32-bit aligned addresses only. Reserved fields should be written as zero, and the EMAC core returns a value of zero in those fields. All registers are set to their default values upon reset.

The EMAC core contains five sets of registers, depending on the IP configuration as follows:

- **DMA Controller Registers**: Control the operation of the DMA controller, starting at offset **0x0000**.
- **MAC Interface Registers**: Control the MAC interface, starting at offset **0x0100**.
- **PTP Registers**: Control the PTP operation, starting at offset **0x0300**.
- **AVB Registers**: Control the AVB operation, starting at offset **0x0400**.

15.3.5.1 DMA Configuration Register

The DMA Configuration Register is used to program the global parameters for the DMA Controller.

Offset: 0x0000				
Bits	**Field**	**Type**	**Reset**	**Description**
31:17	Reserved			Reserved for future use
20	BIG_LITTLE	RW	0	Big/Little Endian Bit 1

Offset: 0x0000				
Bits	**Field**	**Type**	**Reset**	**Description**
	_ENDIAN			This bit should be set only when the EMAC-AHB core is configured to operate in 64-bit mode (Bit [18] is set). Together with Bit [14], this bit determines the endianness for data buffers. The following combinations of {Bit [20], Bit [14]} define the endianness: - 2'b00: Little Endian Byte Order with in the 64-bit QWORD. - 2'b01: Big Endian Byte Order within 32-bit DWORD in a QWORD. - 2'b10: Reserved - 2'b11: Big Endian Byte Order with in the 64-bit QWORD. (Default: 1'b0)
19	DESCRIPTOR_BYTE_ORDERING	RW	0	Descriptor Byte Order Bit (Bit 1) This bit should be set only when the EMAC core is configured to operate in 64-bit mode (Bit [18] is set). Together with Bit [13], this bit determines the byte order for descriptors. The following combinations of {Bit [19], Bit [13]} define the descriptor byte ordering: 2'b00: Normal descriptor QWORD format. 2'b01: Big Endian byte ordering within the descriptor DWORD. 2'b10: Big Endian format for DWORDS within the QWORD. 2'b11: Big Endian format for both DWORDS and bytes within DWORDS. (Default: 1'b0)
18	DMA_64BIT_MODE	RW	0	64-bit Mode - When this bit is set, the receive and transmit DMAs perform 64-bit data transfers for both descriptors and buffer transfers. Note that the addressing remains 32-bitIn this mode, the AHB/AXI master interface performs 64-bit data transfers by setting the size on the AHB/AXI master interface to 3'b011. - This bit should be set only when the EMAC is interfaced to a 64-bit AHB/AXI bus and the EMAC's host data bus width is configured to be 64-bit. - When this bit is reset, the receive and transmit DMAs perform only 32-bit data transfers for both descriptors and buffer transfers. In this mode, the AHB/AXI master interface performs 32-bit data transfers by setting the size on the AHB/AXI master interface to 3'b010. (Default: 1'b0)
17	STRICT_BURST	RW	0	Strict Burst Mode - When this bit is set, the receive and transmit DMAs operate in strict burst mode. In this mode, the DMA restricts the burst size to either the value specified in the burst length field or a single DWORD/QWORD. This is particularly useful when the AHB/AXI bus supports only specific burst sizes, such as 1, 4, 8, or 16.

Offset: 0x0000				
Bits	**Field**	**Type**	**Reset**	**Description**
				- When this bit is reset, the receive and transmit DMAs can use any burst size from 1 up to the value programmed in the burst length field. This allows for optimized performance and is beneficial when the AHB bus supports bursts of any size. (Default: 1'b0)
16	WAIT_FOR_DONE	RW	0	Wait for Done - When this bit is set, the transmit DMA waits for the Done signal from the FIFO interface before fetching the next packet's descriptor. - When this bit is reset, the transmit DMA does not wait for the Done signal from the FIFO interface and continuously processes transmit descriptors, provided the FIFO is not full. (Default: 1'b0)
15	TX_RX_ARBITRATION	RW	0	TX/RX Arbitration This bit selects the internal bus arbitration scheme between the receive and transmit DMA state machines. - When this bit is set, a round-robin arbitration scheme is applied, ensuring equal sharing of the bus between the receive and transmit DMAs. - When this bit is reset, the receive DMA has priority over the transmit DMA, unless the transmit DMA controller is actively transmitting. (Default: 1'b0)
14	BIG_LITLE_ENDIAN	RW	0	Big/Little Endian Bit 0 - In 32-bit mode, setting this bit configures the DMA Controller to operate in Big Endian byte ordering mode for data buffers. When this bit is reset, the DMA Controller operates in Little Endian byte ordering mode for data buffers. - In 64-bit mode, both this bit and Bit [20] determine the Endianness. In 32-bit mode, the encoding of this bit is as follows when Bit [20] is reserved: 1'b0: Little Endian byte order within the DWORD. 1'b1: Big Endian byte order within the DWORD. **Note**. This is valid only for EMAC-AHB with AHB Bus Interface. The AXI Bus is inherently Little Endian, and this setting should not be enabled for the EMAC-AXI. (Default: 1'b0)
13	DESCRIPTOR_BYTE_ORDERING	RW	0	Descriptor Byte Ordering Bit 0 - In 64-bit mode, this bit, along with Bit [19], determines the descriptor byte ordering. - In 32-bit mode, setting this bit causes the receive and transmit DMAs to operate in Big-Endian mode within the 32-bit

Offset: 0x0000				
Bits	**Field**	**Type**	**Reset**	**Description**
				DWORD for descriptors. And this bit has the following encoding values (Bit [19] is reserved): 1’b0: Normal descriptor DWORD format 1’b1: Big-Endian byte ordering within the descriptor DWORD. (Default: 1'b0)
12:8	DESCRIPTOR_SKIP_LENGTH	RW	0	Descriptor Skip Length This field specifies the number of 32-bit DWORDS to skip between two unchained descriptors. It applies to both transmit and receive DMAs. Default: 5’b00000 (continuous descriptors)
7:1	BURST_LENGTH	RW	0x4	Burst Length This field indicates the maximum number of 32-bit DWORDs or 64-bit QWORDS to be transferred to/from the host interface in a single DMA transaction. The permissible values for the burst length field are as follows: - 7’b0000001: 1 DWORD/QWORD - 7’b0000010: 2 DWORDS/QWORDS - 7’b0000100: 4 DWORDS/QWORDS - 7’b0001000: 8 DWORDS/QWORDS - 7’b0010000: 16 DWORDS/QWORDS - 7’b0100000: 32 DWORDS/QWORDS - 7’b1000000: 64 DWORDS/QWORDS Default: 7’b0000100 (4 DWORDS/QWORDS) **Note.** For EMAC-AXI with an AXI interface, the maximum burst length should be restricted to 16 DWORDS/QWORDS to comply with the AXI bus protocol.
0	SOFTWARE_RESET	RW	0	Software Reset When this bit is set, the DMA controller is reset to its default state, clearing all internal state information. Both the receive and transmit DMAs will enter the stopped state. When this bit is reset, the DMA controller operates in normal mode. (Default : 1’b0)

15.3.5.2 DMA Control Register

The DMA Control Register is used to control the Start/Stop of the Transmit/Receive DMA.

Offset: 0x0004				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved			Reserved for future use
1	START_STOP_RECEIVE_DMA	RW	0	Start/Stop Receive DMA - When set: 1. The receive DMA enters the Running state, and the EMAC core attempts to acquire a descriptor from the receive list to process incoming frames. 2. Descriptor acquisition is attempted from the current position in the list (as specified in the Receive Base Address Register) or from the position retained when the receive DMA was previously stopped. 3. If no descriptor is owned by the EMAC core, the receive DMA enters the Suspended state, and the Receive Buffer Unavailable flag is set. 4. The Start Reception command is effective only when the receive DMA is in the Stopped state. 5. If this bit is set before programming the Receive Base Address Register, the EMAC core's behavior will be unpredictable. - When reset: 1. The receive DMA enters the Stopped state after completing the reception of the current frame. 2. The next descriptor position in the receive list is saved and becomes the current position when the receive DMA is restarted. 3. The Stop Receive DMA command is effective only when the receive DMA is in either the Running or Suspended state. (Default: 1'b0)
0	START_STOP_TRANSMIT_DMA	RW	0	Start/Stop Transmit DMA - When set: 1. The transmit DMA enters the Running state, and the EMAC core checks the transmit list at the current position for a frame to be transmitted. 2. Descriptor acquisition is attempted either from the current position in the list (as specified in the Transmit Base Address Register) or from the position retained when the transmit DMA was previously stopped. 3. If the current descriptor is not owned by the EMAC core, the transmit DMA enters the Suspended state, and the Transmit Buffer Unavailable flag is set.

Offset: 0x0004				
Bits	**Field**	**Type**	**Reset**	**Description**
				4. The Start Transmission command is effective only when the transmit DMA is in the Stopped state. If this bit is set before programming the Transmit Base Address Register, the EMAC core's behavior will be unpredictable. - When reset: 1. The transmit DMA enters the Stopped state after completing the transmission of the current frame. 2. The next descriptor position in the transmit list is saved and becomes the current position when the transmit DMA is restarted. 3. The Stop Transmit DMA command is effective only when the transmit DMA is in either the Running or Suspended state. (Default: 1'b0)

15.3.5.3 DMA Status and IRQ Register

The DMA Status and IRQ Register provide Status and IRQ information on various conditions that need to be monitored by the Host software. The IRQ bits are used to generate interrupts to the host.

Offset: 0x0008				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved			Reserved for future use
23:20	RECEIVE_DMA_STATE	RO	0x0	Receive DMA State This field represents the current state of the Receive DMA, with values that change dynamically. The following are the encodings for the Receive DMA states: - 4'b0000: STOPPED - 4'b0001: FETCH_DESCRIPTOR - 4'b0010: WAIT_FOR_END_OF_RECEIVE - 4'b0011: WAIT_FOR_RXFRAME - 4'b0100: SUSPENDED - 4'b0101: CLOSE_DESCRIPTOR - 4'b0110: FLUSH_BUFFER - 4'b0111: PUT_BUFFER - 4'b1000: WAIT_FOR_STATUS (Default: 4'b0000)
19	Reserved			Reserved for future use
18:16	TRANSMIT_DMA_STATE	RO	0x0	Transmit DMA State This field represents the current state of the Transmit DMA,

Offset: 0x0008				
Bits	**Field**	**Type**	**Reset**	**Description**
				with values that change dynamically. The following are the encodings for the Transmit DMA states: - 3'b000: STOPPED - 3'b001: FETCH_DESCRIPTOR - 3'b010: Reserved - 3'b011: FETCH_DATABUFFER - 3'b100: CLOSE_DESCRIPTOR - 3'b101: SUSPENDED - 3'b110: Reserved - 3'b111: Reserved (Default: 3'b000)
15:10	Reserved			Reserved for future use
9	PTP_IRQ	RO	0	1588 Interrupt When set, this bit indicates that an interrupt has occurred from the 1588 portion of the EMAC Core. Software should read the 1588 Interrupt Register to determine the exact source of the interrupt. This bit is cleared only when the corresponding bits in the 1588 Interrupt Register are cleared. This behavior is valid only when the 1588 operation is enabled and configured.
8	MAC_IRQ	RO	0	MAC Interrupt When set, this bit indicates that an interrupt has occurred from the MAC portion of the EMAC Core. Software should read the MAC Interrupt Register to determine the exact source of the MAC interrupt. This bit is cleared only when the corresponding bits in the MAC Interrupt Register are cleared.
7	RECEIVE_MISSED_FRAME_IRQ	RW	0	Receive Missed Frame IRQ When set, this bit indicates that a frame was dropped (missed) because no host receive descriptors were available. The frame is flushed from the Internal FIFO. This bit is set only when the Receive DMA is in the Suspended state and a new frame is received. At this point, the Receive DMA attempts to fetch the descriptor again, but if the descriptor is still owned by the host, the frame is considered missed. The IRQ is cleared by writing a 1 to this bit. (Default: 1'b0)
6	RECEIVE_DMA_STOPPED_IRQ	RW	0	Receive DMA Stopped IRQ This bit is set when the Receive DMA enters the Stopped state. The IRQ is cleared by writing 1 to this bit. (Default: 1’b0)

Offset: 0x0008				
Bits	**Field**	**Type**	**Reset**	**Description**
5	RECEIVE_DES_UNAVAILABLE_IRQ	RW	0	Receive Descriptor Unavailable IRQ When set, this bit indicates that the next descriptor in the receive list is owned by the host and cannot be acquired by the EMAC Core, causing the Receive DMA to enter the Suspended state. To resume processing receive descriptors, the host should change the ownership of the descriptor and write to the Receive Poll Demand Register. If no write to the Receive Poll Demand Register is issued, the Receive DMA will resume automatically when the next recognized incoming frame is received. The IRQ is cleared by writing a 1 to this bit. (Default: 1'b0)
4	RECEIVE_TRANSFER_DONE_IRQ	RW	0	Receive Transfer Done IRQ When set, this bit indicates the completion of a frame reception and the transfer of the frame contents to host memory. Specific frame status information is posted in the descriptor (RDES0) of the Last Descriptor. The Receive DMA enters the Running state and will fetch the next descriptor. In Receive Interrupt Mitigation Mode, the bit is set when the programmed number of frames has been transferred to host memory, or when the Receive Interrupt Timeout counter expires and at least one frame has been transferred to host memory without setting the Receive Transfer Done IRQ. The IRQ is cleared by writing a 1 to this bit. (Default: 1'b0)
3	Reserved			Reserved for future use
2	TRANSMIT_DMA_STOPPED_IRQ	RW	0	Transmit DMA Stopped IRQ This bit is set when the Transmit DMA enters the Stopped state. The IRQ is cleared by writing 1 to this bit. (default: 1’b0)
1	TRANSMIT_DES_UNAVAILABLE_IRQ	RW	0	Transmit Descriptor Unavailable IRQ When set, this bit indicates that the next descriptor on the transmit list is owned by the host and cannot be acquired by the EMAC Core. In this case, the Transmit DMA enters the Suspended state. To resume processing transmit descriptors, the host software should change the ownership bit of the descriptor and then write to the Transmit Poll Demand Register, unless Transmit Auto Polling is enabled.
0	TRANSMIT_TRANSFER_DONE_IRQ	RW	0	Transmit Transfer Done IRQ When set, this bit indicates that a frame transmission has been completed, and the Interrupt On Completion (TDES1[31]) is set in the first descriptor of the frame. - If the Wait for Done bit is not set, the IRQ is triggered when the transmit frame is enqueued into the Transmit FIFO.

Offset: 0x0008				
Bits	**Field**	**Type**	**Reset**	**Description**
				- If the Wait for Done bit is set, the IRQ is triggered only when the frame is completely transmitted onto the MII/GMII interface. The IRQ is cleared by writing a 1 to this bit. (Default: 1'b0)

15.3.5.4 DMA Interrupt Enable Register

The DMA Interrupt Enable Register is used to enable interrupt bits for various IRQ conditions, allowing the generation of interrupts onto the AHB/AXI Bus.

Offset: 0x000C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved			Reserved for future use
9	PTP_INTR_ENABLE	RW	0	1588 Interrupt Enable - When set, this bit enables the 1588 interrupt to generate an interrupt on the AHB/AXI Bus. - When reset, the 1588 interrupt is blocked from generating an interrupt on the AHB/AXI Bus. (Default: 1'b0)
8	MAC_INTR_ENABLE	RW	0	MAC Interrupt Enable - When set, this bit enables the MAC interrupt to trigger an interrupt on the AHB/AXI Bus. - When reset, the MAC interrupt is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
7	RECEIVE_MISSED_FRAME_INTR_ENABLE	RW	0	Receive Missed Frame Interrupt Enable - When set, this bit enables the Receive Missed Frame IRQ to trigger an interrupt on the AHB/AXI Bus. - When reset, the Receive Missed Frame IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
6	RECEIVE_DMA_STOPPED_INTR_ENABLE	RW	0	Receive DMA Stopped Interrupt Enable - When set, this bit enables the Receive DMA Stopped IRQ to trigger an interrupt on the AHB/AXI Bus. - When reset, the Receive DMA Stopped IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
5	RECEIVE_DES_UNAVAILABLE_INTR_ENABLE	RW	0	Receive Descriptor Unavailable Interrupt Enable - When set, this bit enables the Receive Descriptor Unavailable IRQ to trigger an interrupt on the AHB/AXI Bus.

Offset: 0x000C				
Bits	**Field**	**Type**	**Reset**	**Description**
				- When reset, the Receive Descriptor Unavailable IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
4	RECEIVE_TRANSFER_DONE_INTR_ENABLE	RW	0	Receive Transfer Done Interrupt Enable - When set, this bit enables the Receive Transfer Done IRQ to trigger an interrupt on the AHB/AXI Bus. - When reset, the Receive Transfer Done IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
3	Reserved			Reserved for future use
2	TRANSMIT_DMA_STOPPED_INTR_ENABLE	RW	0	Transmit DMA Stopped Interrupt Enable - When set, this bit enables the Transmit DMA Stopped IRQ to trigger an interrupt on the AHB/AXI Bus. - When reset, the Transmit DMA Stopped IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
1	TRANSMIT_DES_UNAVAILABLE_INTR_ENABLE	RW	0	Transmit Descriptor Unavailable Interrupt Enable - When set, this bit enables the Transmit Descriptor Unavailable IRQ to trigger an interrupt on the AHB/AXI Bus. - When reset, the Transmit Descriptor Unavailable IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)
0	TRANSMIT_TRANSFER_DONE_INTR_ENABLE	RW	0	Transmit Transfer Done Interrupt Enable - When set, this bit enables the Transmit Transfer Done IRQ to trigger an interrupt on the AHB/AXI Bus. - When reset, the Transmit Transfer Done IRQ is blocked from triggering an interrupt on the AHB/AXI Bus. (Default: 1'b0)

15.3.5.5 Transmit Auto Poll Counter Register

The DMA Transmit Auto Poll Counter Register determines the polling frequency for the Transmit DMA when it is in the Suspended state. In this state, descriptor fetching is reinitiated either when the Transmit Auto Poll Counter expires or when a write is made to the Transmit Poll Demand Register.

Offset: 0x0010				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use

Offset: 0x0010				
Bits	**Field**	**Type**	**Reset**	**Description**
15:0	TRANSMIT_AUTO_POLL	RW	0x0	Transmit Auto Poll Value It defines the number of AHB/AXI clocks the system waits in the Suspended state before the Transmit DMA attempts to re-fetch the descriptor. - If no descriptor is available, the Transmit DMA returns to the Suspended state, and the Transmit Descriptor Unavailable Interrupt is set. - If a descriptor is available, the Transmit DMA resumes operation. The internal Auto Poll Counter operates only when the Transmit DMA is in the Suspended state and is reset in other states. The Transmit Auto Polling feature is disabled when this value is written as all zeros. (Default: 16'h0000)

15.3.5.6 DMA Transmit Poll Demand Register

The DMA Transmit Poll Demand Register is used to instruct the Transmit DMA to begin fetching the descriptor again while the Transmit DMA is in the Suspended State.

Offset: 0x0014				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TRANSMIT_POLL_DEMAND	WO	0x0	Transmit Poll Demand When the Transmit DMA is in the Suspended State, writing any value to this register prompts the EMAC Core to check for frames to transmit by re-fetching the descriptor. - If no descriptor is available, the Transmit DMA returns to the Suspended State, and the Transmit Descriptor Unavailable Interrupt is set. - If a descriptor is available, the Transmit DMA resumes operation.

15.3.5.7 DMA Receive Poll Demand Register

The DMA Receive Poll Demand Register is used to signal the Receive DMA to begin fetching the descriptor again while the Receive DMA is in the Suspended State.

Offset: 0x0018				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RECEIVE_POLL_DEMAND	WO	0x0	Receive Poll Demand When the Transmit DMA is in the Suspended state, writing

Offset: 0x0018				
Bits	**Field**	**Type**	**Reset**	**Description**
				any value to this register triggers the EMAC Core to check for frames to be transmitted by re-fetching the descriptor. - If no descriptor is available, the Transmit DMA returns to the Suspended state, and the Transmit Descriptor Unavailable Interrupt is set. - If a descriptor is available, the Transmit DMA resumes operation.

15.3.5.8 DMA Transmit Base Address Register

This register specifies the starting address of the Transmit Descriptor list in the host's memory space. The value programmed into this register must be 32-bit aligned.

Offset: 0x001C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TRANSMIT_BASE_ADDRESS	RW	0x0	Transmit Base Address It holds the starting address of the Transmit Descriptor list in the host memory space. When the Transmit DMA starts, it uses the value in this register to fetch descriptors (only if the register has been updated). Otherwise, it continues from the last saved address before the DMA was stopped. - All descriptors are 32-bit aligned, so the programmed value must be 32-bit aligned. - In 64-bit Host Bus width mode, descriptors are 64-bit aligned, requiring a 64-bit aligned programmed value. - This register should be written only when the Transmit DMA is in the Stopped state. (Default: 32'h0000_0000)

15.3.5.9 DMA Receive Base Address Register

The DMA Receive Base Address Register is used to point to the Start of Receive Descriptor list in the Host's Memory Space. The value programmed in this register should be 32-bit aligned value.

Offset: 0x0020				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RECEIVE_BASE_ADDRESS	RW	0x0	Receive Base Address It holds the starting address of the Receive Descriptor list in the host memory space. When the Receive DMA starts, it uses the value in this register to fetch descriptors (only if the register has been updated). Otherwise, it continues

Offset: 0x0020				
Bits	**Field**	**Type**	**Reset**	**Description**
				from the last saved address before the DMA was stopped. - All descriptors are 32-bit aligned, so the programmed value must be 32-bit aligned. - In 64-bit mode, descriptors are 64-bit aligned, requiring a 64-bit aligned programmed value. - This register should be written only when the Receive DMA is in the Stopped state. (Default: 32'h0000_0000)

15.3.5.10 DMA Missed Frame Counter Register

The DMA Missed Frame Counter Register tracks the number of frames missed due to the unavailability of a receive descriptor. When the Receive DMA is suspended and a new receive frame is received, it attempts to fetch a descriptor. If the descriptor is unavailable (i.e. owned by the host), the frame is flushed from the receive FIFO. This counter represents the total number of frames missed since the counter was last read.

Offset: 0x0024				
Bits	**Field**	**Type**	**Reset**	**Description**
31	MISSED_FRAME_COUNTER_OVERFLOW_IRQ	RW	0	Missed Frame Counter Overflow IRQ This bit is set when the Missed Frame Counter overflows. **Note.** Setting this bit does not trigger an interrupt. The bit is cleared when this register is read by the host. (Default: 1'b0)
30:0	MISSED_FRAME_COUNTER	RW	0x0	Missed Frame Counter It tracks the number of frames missed due to the unavailability of a receive descriptor. When the Receive DMA is suspended and a new receive frame is received, it attempts to fetch a descriptor. If the descriptor is unavailable (i.e., owned by the host), the frame is flushed from the receive FIFO. This counter represents the total number of such frames missed since the counter was last read. The counter is cleared when it is read. (Default: 31'h0000_0000)

15.3.5.11 DMA Stop Flush Counter Register

The DMA Stop Flush Counter Register tracks the number of frames that were flushed because the Receive DMA is in Stopped State, since the counter was last read. An interrupt is generated when the counter overflows.

Offset: 0x0028				
Bits	**Field**	**Type**	**Reset**	**Description**
31	STOP_COUNTER_OVERFLOW_IRQ	RW	0	Stop Counter Overflow IRQ This bit is set when the Stop Flush Counter overflows. Setting this bit does not trigger an interrupt. The bit is cleared when this register is read by the Host. (Default: 1'b0)
30:0	STOP_FLUSH_COUNTER	RW	0x0	Stop Flush Counter It tracks the number of frames that were flushed from the Receive FIFO due to the Receive DMA being in the Stopped State. This counter represents the total number of frames flushed since the counter was last read. The counter is cleared upon reading. (Default: 31'h0000_0000)

15.3.5.12 DMA Receive Transfer Done Interrupt Mitigation Control

The DMA Receive Transfer Done Interrupt Control Register governs the behavior of interrupt generation when a packet is received and successfully transferred to Host Memory (Receive Transfer Done Interrupt).

Offset: 0x002C				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RECEIVE_IRQ_MITIGATION_ENABLE	RW	0	Receive Transfer Done Interrupt Mitigation Control Enable - When set: Enables Receive Transfer Done Interrupt Mitigation Control using the packet/timer counter. - When reset: Disables interrupt mitigation, and the Receive Transfer Done Interrupt is asserted for each packet transferred to host memory. The DMA Interrupt Register will reflect this interrupt. (Default Value: 1'b0)
30	RECEIVE_IRQ_FRAME_COUNTER_MODE	RW	0	Receive Interrupt Frame Counter Mode - When set: The Receive Interrupt Frame Counter is reset to 8'h01 after the Receive Transfer Done Interrupt is generated when operating in Receive Interrupt Mitigation Mode. - When reset: The Receive Interrupt Frame Counter is not modified

Offset: 0x002C				
Bits	**Field**	**Type**	**Reset**	**Description**
				internally and retains the value programmed by the software.
29:28	Reserved			Reserved for future use
27:8	RECEIVE_IRQ_TIMEOUT_COUNTER	RW	0xFFFF	Receive Interrupt Timeout Counter These bits define the maximum time (in AHB/AXI clock periods) between the last Receive Transfer Done Interrupt and the assertion of a new Receive Transfer Done Interrupt, provided that at least one frame has been transferred to Host Memory. When Interrupt Mitigation is enabled, the Receive Transfer Done Interrupt is asserted after every 'n' frames, where 'n' is determined by the Receive Interrupt Frame Counter. The Receive Transfer Done Interrupt is asserted when either: - 'n' frames have been transferred to Host Memory, or - The internal Timeout Counter exceeds the value programmed in these bits, and at least one frame has been transferred to Host Memory. (Default Value: 20'h0_FFFF)
7:0	RECEIVE_IRQ_FRAME_COUNTER	RW	0x1	Receive Interrupt Frame Counter These bits specify the number of frames to be counted before asserting the Receive Transfer Done Interrupt. The value in this register is only valid when Bit [31] is set. When Bit [31] is set, the Receive DMA will assert the Receive Transfer Done Interrupt after every 'n' frames, where 'n' is the value programmed in this register. Legal values: 1 - 255 Default Value: 8'b0000_0001

15.3.5.13 DMA Current Tx. Descriptor Pointer Register

This register holds the pointer to the current descriptor that the Transmit DMA is using.

Offset: 0x0030				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CURRENT_TRANSMIT_DES_POINTER	RO	0x0	Current Transmit Descriptor Pointer This field holds the current descriptor pointer being used by the Transmit DMA. The value is aligned according to the configured bus width of the DMA, either 32-bit or 64-bit.

Offset: 0x0030				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. This is a read-only value, typically used for debugging purposes. (Default: 32'h0000_0000)

15.3.5.14 DMA Current Tx. Buffer Pointer Register

This register contains the current buffer pointer being used by the Transmit DMA.

Offset: 0x0034				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CURRENT_TRANSMIT_BUFER_POINTER	RO	0x0	Current Transmit Buffer Pointer This field contains the current buffer pointer the Transmit DMA is using. **Note.** This is a Read only value used for debugging purposes. (Default: 32'h0000_0000)

15.3.5.15 DMA Current Rx. Descriptor Pointer Register

This register contains the current descriptor pointer being used by the Receive DMA.

Offset: 0x0038				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CURRENT_RECEIVE_DES_POINTER	RO	0x0	Current Receive Descriptor Pointer This field contains the current descriptor pointer the Receive DMA is using. This is a 32-bit aligned value or 64-bit aligned value based on the configured bus width of the DMA. **Note.** This is a read-only value used for debugging purposes. (Default: 32'h0000_0000)

15.3.5.16 DMA Current Rx. Buffer Pointer Register

This register contains the current buffer pointer being used by Receive DMA.

Offset: 0x003C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CURRENT_RECEIVE_BUFFER_POINTER	RO	0x0	Current Receive Buffer Pointer This field contains the current buffer pointer the Receive DMA is using. **Note.** This is a read-only value used for debugging purposes. (Default: 32'h0000_0000)

15.3.5.17 MAC Global Control Register

The MAC Global Control Register is used to program the global parameters for the MAC in the EMAC Core.

Offset: 0x0100				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved			Reserved for future use
9	MAGIC_PACKET_WAKEUP_MODE	RW	0	Magic Packet Wakeup Mode When set, the EMAC Core operates in Wakeup Mode, preventing received frames from being written into the RXFIFO of the MAC. A wakeup event is triggered when a Magic Packet, whose Destination Address (DA) passes address filtering, contains a valid Magic Packet Signature in the packet contents. (Default: 1'b0)
8	UNICAST_WAKEUP_MODE	RW	0	Unicast Wakeup Mode When set, the EMAC Core operates in Wakeup Mode, preventing received frames from being written into the RXFIFO of the MAC. A wakeup event is triggered when a unicast frame, whose destination address (DA) matches the value programmed in the MAC Address #0 Registers, is received. (Default: 1'b0) **Note.** This register bit is reserved if Wakeup Mode is not supported in the current configuration.

Offset: 0x0100				
Bits	**Field**	**Type**	**Reset**	**Description**
7:5	Reserved			Reserved for future use
4	RESET_TX_STAT_COUNTERS	RW	0	Reset Tx. Stat. Counters When set, this field initiates the resetting of all Statistic Counters related to the Transmit path in the EMAC Core. The actual reset process begins after this bit is cleared. (Default: 1'b0)
3	RESET_RX_STAT_COUNTERS	RW	0	Reset Rx. Stat. Counters When set, this field initiates the resetting of all Statistic Counters related to the Receive path in the EMAC Core. The actual reset process begins after this bit is cleared. (Default: 1'b0)
2	DUPLEX_MODE	RW	0	Duplex Mode - When set: The EMAC core operates in Full-Duplex mode, enabling simultaneous transmission and reception of data. - When reset: The EMAC core operates in Half-Duplex mode, implementing the CSMA/CD protocol to monitor collisions and apply back-off mechanisms in case of collisions. Default Value: 1'b0 1'b0: Half-Duplex Mode 1'b1: Full-Duplex Mode **Note.** - Changing the Full-Duplex bit is permitted only if the transmitter and receiver are disabled. - When the Speed variable is set to 2'b10 (1000 Mbps), the EMAC core must be configured for Full-Duplex mode, as it does not support Half-Duplex mode in Gigabit operation.
1:0	SPEED	RW	0x0	Speed This field determines the Ethernet interface speed of the EMAC Core. Based on the configured speed, the EMAC Core selects either the GMII or MII interface. The txclk and rxclk frequencies should match the selected speed.

Offset: 0x0100				
Bits	**Field**	**Type**	**Reset**	**Description**
				Permissible values: - 2'b00 (10 Mbps) – Default - 2'b01 (100 Mbps) - 2'b10 (1000 Mbps) - 2'b11 (Reserved)

15.3.5.18 MAC Transmit Control Register

The MAC Transmit Control Register is used to configure the parameters for the transmit portion of the MAC in the EMAC Core.

Offset: 0x0104				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	Reserved			Reserved for future use
9:7	PREAMBLE_LENGTH	RW	0x0	Preamble Length It determines the number of Preamble Bytes followed by SFD to be prepended to outgoing frames. According to the IEEE 802.3 specification, each frame must have 7 bytes of Preamble followed by 1 byte of SFD at the beginning. The permissible values for the Preamble Length field are as follows: - 3'b000: 7 Bytes of Preamble (Default) - 3'b001: 1 Byte of Preamble - 3'b010: 2 Bytes of Preamble - 3'b011: 3 Bytes of Preamble - 3'b100: 4 Bytes of Preamble - 3'b101: 5 Bytes of Preamble - 3'b110: 6 Bytes of Preamble - 3'b111: 7 Bytes of Preamble **Note.** Programming the Preamble Length to anything other than 3'b000 (7 Bytes of Preamble) violates the IEEE 802.3 specification and may cause interoperability issues with peer network devices. This field only controls the length of the Preamble; each outgoing frame will automatically have the SFD added after the Preamble and before the Destination Address field.
6:4	IFG_LEN	RW	0x0	IFG Length It determines the minimum Inter-Packet Gap (IPG) or Inter-Frame Gap (IFG) to be inserted between

Offset: 0x0104				
Bits	**Field**	**Type**	**Reset**	**Description**
				outgoing frames when the transmit FIFO has back-to-back data. The IEEE 802.3 specification requires a minimum IPG of 96 bit-times between frames. The permissible values for the IFG Length field are as follows: - 3'b000: 96 Bit Times of IFG - 3'b001: 64 Bit Times of IFG - 3'b010: 128 Bit Times of IFG - 3'b011: 256 Bit Times of IFG - 3'b100: 24 Bit Times of IFG - 3'b101: 32 Bit Times of IFG - 3'b110: 40 Bit Times of IFG - 3'b111: 48 Bit Times of IFG Default: 3'b000 (96 bit-times) **Note.** - Programming the IFG Length to less than 96 bit-times may cause interoperability issues with peer network devices. - Programming the IFG Length to 3'b010 or 3'b011 (128 or 256 bit-times) may result in performance degradation.
3	TRANSMIT_AUTO_RETRY	RW	0	Transmit Auto Retry - When set: The EMAC core automatically retries frame transmission on collision in Half-Duplex mode. The frame contents remain in the transmit FIFO until the collision window has passed. - When reset: The host is expected to retransmit the frame manually on collision. (Default: 1'b0) **Note.** This field must be set to 1'b1 for proper operation of the EMAC core in Half-Duplex mode.
2	DISABLE_FCS_INSERT	RW	0	Disable FCS Insertion - When set: The FCS calculation and insertion logic is disabled in the transmit path. FCS insertion is disabled for all frames. - When reset: The FCS is calculated for all frames and inserted at the end of the outgoing frame, provided the per-frame FCS insertion is disabled (TDES1[28]).

Offset: 0x0104				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. When Disable FCS Insertion is enabled, it is expected that frames from the host memory will: - Meet the 64-byte minimum frame size requirement, and - Include the FCS field to ensure compliance with the IEEE 802.3 specification. (Default: 1'b0)
1	INVERT_FCS	RW	0	Invert FCS - When set: The EMAC Core inverts the FCS field being inserted into the outgoing frame. - When reset: The EMAC Core performs normal FCS insertion without inversion. **Note.** - According to the IEEE 802.3 specification, the FCS is calculated from the first byte of the DA field to the last byte of the DATA/PAD field. The calculated FCS is then inverted and transmitted in MSB first. - In normal mode, the FCS is inverted before being inserted into the outgoing frame. - When the Invert FCS is set, the FCS field is double-inverted (effectively no inversion occurs). (Default: 1'b0)
0	TRANSMIT_ENABLE	RW	0	Transmit Enable - When set The EMAC Core’s transmitter is enabled and will transmit frames from the Transmit FIFO onto the MII/GMII Interface. - When reset The EMAC Core’s transmitter is disabled and will not transmit any frames. (Default: 1'b0)

15.3.5.19 MAC Receive Control Register

The MAC Receive Control Register is used to program the parameters for the Receive portion of the MAC in the EMAC Core.

Offset: 0x0108				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	Reserved			Reserved for future use
6	ACOOUNT_VLAN	RW	0	Account VLANs According to the IEEE 802.3 specification: - The minimum frame size (minFrameSize) for untagged frames is 64 bytes. - The maximum frame size (maxFrameSize) for untagged frames is 1518 bytes. For VLAN-tagged frames, the frame size values are adjusted to account for the 4-byte VLAN tag, increasing the maxFrameSize to 1522 bytes. The EMAC core supports up to 3 VLAN tags per frame. It can dynamically adjust the maxFrameSize field to accommodate up to 12 bytes (3 VLAN tags) based on the number of VLAN tags present in the incoming frame. This prevents false maxFrameSize violation reporting for VLAN-tagged frames that exceed 1518 bytes but are within the adjusted limit (1518 + tag bytes). - When set: The maxFrameSize field is dynamically adjusted to account for up to 3 VLAN tags (12 bytes) based on the number of VLAN tags in the incoming frame. - When reset: The maxFrameSize field is not adjusted and remains fixed at the value programmed in the Maximum Frame Size Register. (Default: 1’b0)
5	PASS_BAD_FRAMES	RW	0	Pass Bad Frames When the EMAC core operates in Store-and-Forward mode, frames with errors (e.g., CRC errors, minimum frame length errors, fragments, maximum frame length errors, receive errors, etc.) are typically flushed from the receive FIFO and not transferred to host memory. - When set: Frames with errors are transferred to host memory, and the appropriate error status is indicated in the Frame Status field (written back in the TDES0 of the last descriptor). - When reset: Frames with errors are not transferred to host memory and are flushed from the receive FIFO. (Default: 1’b0)
4	STATUS_FIRST	RW	0	Status First

Offset: 0x0108				
Bits	**Field**	**Type**	**Reset**	**Description**
				When the EMAC core operates in Store-and-Forward mode, a frame is sent from the internal MAC to the DMA only after the entire frame is stored in the receive FIFO. - When set: The status field is transferred before the start of the packet (SOP). - When reset: The status field is transferred after the end of the packet (EOP). For proper operation of the EMAC core, this bit should always be reset (1’b0).
3	STORE_FORWARD	RW	0	Store and Forward - When set: The EMAC core operates in store-and-forward mode in the receive direction. A frame is transferred to host memory only after the entire frame is stored in the receive FIFO. Frames with errors (e.g., CRC errors, minimum frame length errors, fragments, maximum frame length errors, receive errors, etc.) are flushed from the receive FIFO and not transferred to host memory. - When reset: The EMAC core operates in cut-through mode in the receive direction. Frames are transferred without waiting for the end of the frame. (Default: 1’b0) When operating in store-and-forward mode, the receive FIFO should be large enough to store at least one full-length frame. If jumbo frames are supported, the receive FIFO must accommodate one full jumbo frame.
2	STRIP_FCS	RW	0	Strip FCS - When set: The FCS field is stripped from the frame before it is transferred to host memory. The frame length field is updated to reflect the new length (excluding the FCS field). - When reset: The FCS field is not stripped and remains part of the frame transferred to host memory. Regardless of the Strip FCS setting, FCS checking is performed on every frame unless Disable FCS Check is set. If the FCS field is checked, CRC error status is reported for every frame.

Offset: 0x0108				
Bits	**Field**	**Type**	**Reset**	**Description**
				(Default: 1'b0)
1	DISABLE_FCS_CHECK	RW	0	Disable FCS Checking - When set: The EMAC core does not perform FCS checking on incoming frames, and the CRC error status is not set for any frames. - When reset: The EMAC core performs FCS checking on every incoming frame. Frames with CRC errors are reported and may be dropped. (Default: 1'b0)
0	RECEIVE_ENABLE	RW	0	Receive Enable - When set: The EMAC core's receiver is enabled, allowing it to receive frames from the MII/GMII interface and transfer them into the receive FIFO. - When reset: The EMAC core's receiver is disabled and does not receive any frames. (Default: 1'b0)

15.3.5.20 MAC Maximum Frame Size Register

The MAC Maximum Frame Size Register is used to set the value of the MaxFrameSize field to check for MaxFrameLength violations.

Offset: 0x010C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:14	Reserved			Reserved for future use
13:0	MAX_FRAME_SIZE	RW	0x5EE	Maximum Frame Size This field defines the maximum frame size for untagged frames used to check MaxFrameLength violations. - In store-and-forward mode: Any frame exceeding this value is flushed from the receive FIFO unless Pass Bad Frames is set. - When Pass Bad Frames is set: Frames larger than this value are transferred to host memory with the MaxFrameLength Error bit set in the Frame Status, which is written back in RDES0 of the last descriptor.

Offset: 0x010C				
Bits	**Field**	**Type**	**Reset**	**Description**
				(Default: 14'h05EE)

15.3.5.21 MAC Transmit Jabber Size Register

This register sets the jabber size limit for outgoing frames:

- If an outgoing frame exceeds this size, it is truncated and marked with EOP-ERROR.

Offset: 0x0110				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	TRANSMIT_JABBER_SIZE	RW	0x600	Transmit Jabber Size This field defines the jabber size for outgoing (transmit) frames. - If an outgoing frame exceeds this size, it is considered a Jabber Frame and is truncated at that point with EOP-ERROR. - The PHY will then force an error code onto the line, preventing excessive frame transmission due to programming errors. (Default: 16'h0600)

15.3.5.22 MAC Receive Jabber Size Register

This register sets the jabber size limit for incoming frames:

- If an incoming frame exceeds this size, it is truncated, and the Jabber Error is set in the Frame Status.
- The remaining portion of the jabbered frame is received but ignored.

Offset: 0x0114				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	RECEIVE_JABBER_SIZE	RW	0x600	Receive Jabber Size This field defines the jabber size for incoming (receive) frames. - If an incoming frame exceeds this size, it is considered a

Offset: 0x0114				
Bits	**Field**	**Type**	**Reset**	**Description**
				Jabber Frame and is truncated at that point. - The Frame Status is updated with a Jabber Error, and the remaining portion of the frame is ignored. - In store-and-forward mode: The frame is flushed from the receive FIFO unless Pass Bad Frames is set. (Default: 16'h0600)

15.3.5.23 MAC Address Control Register

This register controls MAC address checking for all incoming frames:

- Address Filtering:
  - The EMAC core filters incoming frames based on the Destination Address field.
  - Unicast frames are checked using the four MAC Address Registers.
  - Multicast frames are filtered using the Multicast Hash Table.
- Operation in Normal Mode:
  - Only frames that pass address filtering are transferred to host memory.
  - Frames that fail address filtering are flushed from the receive FIFO.
  - The filtering result for each frame is recorded in the TDES0 field of the last descriptor.
- Operation in Promiscuous Mode:
  - All frames, regardless of whether they pass or fail address filtering, are transferred to host memory.

Offset: 0x0118				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	Reserved			Reserved for future use
8	PROMISCUOUS_MODE	RW	0	Promiscuous Mode - When set, the EMAC transfers all frames to host memory, regardless of the Destination Address. - The Packet Status field will still reflect the result of the address filtering process. (Default: 1'b0)
7	INVERSE_MAC_ADDRESS4_ENABLE	RW	0	Inverse MAC Address #4 Enable - When set: Inverse filtering is performed on the Destination Address field using the value in MAC Address Register #4. Inverse filtering is only enabled when the MAC Address #4 Enable bit is set.

Offset: 0x0118				
Bits	**Field**	**Type**	**Reset**	**Description**
				- When reset: Normal address filtering is performed using MAC Address Register #4. (Default: 1'b0)
6	INVERSE_MAC_ADDRESS3_ENABLE	RW	0	Inverse MAC Address #3 Enable - When set: Inverse filtering is performed on the Destination Address field using the value in MAC Address Register #3. Inverse filtering is enabled only when the MAC Address #3 Enable bit is set. - When reset: Normal address filtering is performed using MAC Address Register #3. (Default: 1'b0)
5	INVERSE_MAC_ADDRESS2_ENABLE	RW	0	Inverse MAC Address #2 Enable - When set: Inverse filtering is performed on the Destination Address field using the value in MAC Address Register #2. Inverse filtering is enabled only when the MAC Address #2 Enable bit is set. - When reset: Normal address filtering is performed using MAC Address Register #2. (Default: 1'b0)
4	INVERSE_MAC_ADDRESS1_ENABLE	RW	0	Inverse MAC Address #1 Enable - When set: Inverse filtering is performed on the Destination Address field using the value in MAC Address Register #1. Inverse filtering is enabled only when the MAC Address #1 Enable bit is set. - When reset: Normal address filtering is performed using MAC Address Register #1. (Default: 1'b0)
3	MAC_ADDRESS4_ENABLE	RW	0	MAC Address #4 Enable - When set: The MAC Address Register #4 is used to perform address filtering on the Destination Address field of all incoming frames. - When reset: The MAC Address Register #4 is not used for address filtering. (Default: 1'b0)
2	MAC_ADDRES	RW	0	MAC Address #3 Enable

Offset: 0x0118				
Bits	**Field**	**Type**	**Reset**	**Description**
	S3_ENABLE			- When set: The MAC Address Register #3 is used to perform address filtering on the Destination Address field of all incoming frames. - When reset: The MAC Address Register #3 is not used for address filtering. (Default: 1'b0)
1	MAC_ADDRESS2_ENABLE	RW	0	MAC Address #2 Enable - When set: The MAC Address Register #2 is used to perform address filtering on the Destination Address field of all incoming frames. - When reset: The MAC Address Register #2 is not used for address filtering. (Default: 1'b0)
0	MAC_ADDRESS1_ENABLE	RW	0	MAC Address #1 Enable - When set: The MAC Address Register #1 is used to perform address filtering on the Destination Address field of all incoming frames. - When reset: The MAC Address Register #1 is not used for address filtering. (Default: 1'b0)

15.3.5.24 MAC MDIO Clock Division Control Register

This register adjusts the ratio between the MDC (Management Data Clock) and the host clock, which can either be the AHB clock or the AXI clock.

Offset: 0x011C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved			Reserved for future use
7:0	MDC_CLK_DIV_CONTROL	RW	0x40	MDC Clock Division Control This field defines the ratio between the MDC Clock and the host clock. - For the default value of 64, the MDC Clock is high for 32 host clock periods and low for 32 host clock periods. - Only even values are supported for this field, with a

Offset: 0x011C				
Bits	**Field**	**Type**	**Reset**	**Description**
				maximum value of 8'hFE. (Default: 8'h40)

15.3.5.25 MAC Address#1 High Register

The MAC Address #1 High Register contains the first two bytes of MAC Address #1 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address #1 are stored in the:

- MAC Address #1 Medium Register
- MAC Address #1 Low Register

Offset: 0x0120				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS1_02_BYTE	RW	0x0	MAC Address #1 Second Byte This field contains the second byte of MAC Address #1 in Canonical Format. It is the second byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS1_01_BYTE	RW	0x0	MAC Address #1 First Byte This field contains the first byte of MAC Address #1 in Canonical Format. It is the first byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.26 MAC Address#1 Med Register

The MAC Address#1 Med Register holds the middle two bytes of MAC Address#1 in Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address #1 are stored in the:

- MAC Address#1 High Register
- MAC Address#1 Low Register

Offset: 0x0124				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS1_04_BYTE	RW	0x0	MAC Address#1 Fourth Byte This byte contains the fourth byte of MAC Address #1 in Canonical Format. It is the fourth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS1_03_BYTE	RW	0x0	MAC Address#1 Third Byte This byte contains the third byte of MAC Address #1 in Canonical Format. It is the third byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.27 MAC Address#1 Low Register

The MAC Address #1 Low Register contains the last two bytes of MAC Address#1 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#1 are stored in the:

- MAC Address#1 High Register
- MAC Address#1 Medium Register

Offset: 0x0128				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS1_05_BYTE	RW	0x0	MAC Address#1 Fifth Byte This byte contains the fifth byte of MAC Address#1 in Canonical Format. It is the fifth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS_06_SIXTH_BYTE	RW	0x0	MAC Address#1 Sixth Byte This byte contains the sixth byte of MAC Address#1 in Canonical Format. It is the sixth (last) byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.28 MAC Address#2 High Register

The MAC Address#2 High Register contains the first two bytes of MAC Address#2 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#2 are stored in the:

- MAC Address#2 Medium Register
- MAC Address#2 Low Register

Offset: 0x012C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS2_02_BYTE	RW	0x0	MAC Address#2 Second Byte This field contains the second byte of MAC Address#2 in Canonical Format. It is the second byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS2_01_BYTE	RW	0x0	MAC Address#2 First Byte This field contains the first byte of MAC Address#2 in Canonical Format. It is the first byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.29 MAC Address#2 Med Register

The MAC Address#2 Med Register holds the middle two bytes of MAC Address#2 in Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#2 are stored in the:

- MAC Address#2 High Register
- MAC Address#2 Low Register

Offset: 0x0130				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS2_04_BYTE	RW	0x0	MAC Address#2 Fourth Byte This byte contains the fourth byte of MAC Address#2 in Canonical Format. It is the fourth byte that goes out on the line as part of

Offset: 0x0130				
Bits	**Field**	**Type**	**Reset**	**Description**
				the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS2_03_BYTE	RW	0x0	MAC Address#2 Third Byte This byte contains the third byte of MAC Address#2 in Canonical Format. It is the third byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.30 MAC Address#2 Low Register

The MAC Address#2 Low Register contains the last two bytes of MAC Address#2 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address #2 are stored in the:

- MAC Address#2 High Register
- MAC Address#2 Medium Register

Offset: 0x0134				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS2_05_BYTE	RW	0x0	MAC Address#2 Fifth Byte This byte contains the fifth byte of MAC Address#2 in Canonical Format. It is the fifth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS2_06_BYTE	RW	0x0	MAC Address#2 Sixth Byte This byte contains the sixth byte of MAC Address#2 in Canonical Format. It is the sixth (last) byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.31 MAC Address#3 High Register

The MAC Address#3 High Register contains the first two bytes of MAC Address#3 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#3 are stored in the:

- MAC Address#3 Medium Register
- MAC Address#3 Low Register

Offset: 0x0138				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS3_02_BYTE	RW	0x0	MAC Address#3 Second Byte This field contains the second byte of MAC Address#3 in Canonical Format. It is the second byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS3_01_BYTE	RW	0x0	MAC Address#3 First Byte This field contains the first byte of MAC Address#3 in Canonical Format. It is the first byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.32 MAC Address#3 Med Register

The MAC Address#3 Med Register holds the middle two bytes of MAC Address#3 in Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#3 are stored in the:

- MAC Address#3 High Register
- MAC Address#3 Low Register

Offset: 0x013C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS3_04_BYTE	RW	0x0	MAC Address#3 Fourth Byte This byte contains the fourth byte of MAC Address#3 in Canonical Format. It is the fourth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS3_03_BYTE	RW	0x0	MAC Address#3 Third Byte This byte contains the third byte of MAC Address#3 in

Offset: 0x013C				
Bits	**Field**	**Type**	**Reset**	**Description**
				Canonical Format. It is the third byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.33 MAC Address#3 Low Register

The MAC Address#3 Low Register contains the last two bytes of MAC Address#3 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#3 are stored in the:

- MAC Address#3 High Register
- MAC Address#3 Medium Register

Offset: 0x0140				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS3_05_BYTE	RW	0x0	MAC Address#3 Fifth Byte This byte contains the fifth byte of MAC Address#3 in Canonical Format. It is the fifth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS3_06_BYTE	RW	0x0	MAC Address#3 Sixth Byte This byte contains the sixth byte of MAC Address#3 in Canonical Format. It is the sixth (last) byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.34 MAC Address#4 High Register

The MAC Address#4 High Register contains the first two bytes of MAC Address#4 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address #4 are stored in the:

- MAC Address#4 Medium Register
- MAC Address#4 Low Register

Offset: 0x0144				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS4_02_BYTE	RW	0x0	MAC Address#4 Second Byte This field contains the second byte of MAC Address#4 in Canonical Format. It is the second byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS4_01_BYTE	RW	0x0	MAC Address #4 First Byte This field contains the first byte of MAC Address#4 in Canonical Format. It is the first byte transmitted on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.35 MAC Address#4 Med Register

The MAC Address#4 Med Register holds the middle two bytes of MAC Address#4 in Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#4 are stored in the:

- MAC Address#4 High Register
- MAC Address#4 Low Register

Offset: 0x0148				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS4_04_BYTE	RW	0x0	MAC Address#4 Fourth Byte This byte contains the fourth byte of MAC Address#4 in Canonical Format. It is the fourth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS4_03_BYTE	RW	0x0	MAC Address#4 Third Byte This byte contains the third byte of MAC Address#4 in Canonical Format. It is the third byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.36 MAC Address#4 Low Register

The MAC Address#4 Low Register contains the last two bytes of MAC Address#4 in the Canonical Format. In this format, bytes are transmitted serially over the Ethernet Interface.

The remaining bytes of the 48-bit MAC Address#4 are stored in the:

- MAC Address#4 High Register
- MAC Address#4 Medium Register

Offset: 0x014C

Bits	Field	Type	Reset	Description
31:16	Reserved			Reserved for future use
15:8	MAC_ADDRESS4_05_BYTE	RW	0x0	MAC Address#4 Fifth Byte This byte contains the fifth byte of MAC Address#4 in Canonical Format. It is the fifth byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)
7:0	MAC_ADDRESS4_06_BYTE	RW	0x0	MAC Address#4 Sixth Byte This byte contains the sixth byte of MAC Address#4 in Canonical Format. It is the sixth (last) byte that goes out on the line as part of the Destination Address (DA) field. (Default: 8'b0)

15.3.5.37 MAC Multicast Hash Table#1 Register

The MAC Multicast Hash Table Register (#1 to #4) forms a 64-bit multicast hash table used for group address filtering.

Hash Filtering Process:

- The destination address (DA) field of the incoming frame is passed through the CRC logic.
- The upper 6 bits of the CRC register are used to index the hash table:
  - A value of 6'b000000 selects bit [0] of the hash table.
  - A value of 6'b111111 selects bit [63] of the hash table.
- When a multicast frame is hashed into the table, if the corresponding bit is '1', the frame is accepted. If it is '0', the frame is rejected.

Offset: 0x0150

Bits	Field	Type	Reset	Description
31:16	Reserved			Reserved for future use
15:0	HASH_TABLE_15_TO_0	RW	0x0	Hash Table Bits [15:0] These bits represent the lower 16 bits of the Multi-Cast Hash Table. - When a bit is set to '1', the corresponding multicast frame hashed to that bit is accepted and forwarded to the host. - If the bit is set to '0', the multicast frame is dropped. (Default: 16'b0)

15.3.5.38 MAC Multicast Hash Table#2 Register

The MAC Multicast Hash Table Register (#1 to #4) forms a 64-bit multicast hash table used for group address filtering.

Hash Filtering Process:

- The destination address (DA) field of the incoming frame is passed through the CRC logic.
- The upper 6 bits of the CRC register are used to index the hash table:
  - A value of 6’b000000 selects bit [0] of the hash table.
  - A value of 6’b111111 selects bit [63] of the hash table.
- When a multicast frame is hashed into the table, if the corresponding bit is ‘1’, the frame is accepted. If it is ‘0’, the frame is rejected.

Offset: 0x0154				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	HASH_TABLE_31_TO_16	RW	0x0	Hash Table Bits [31:16] These bits represent the lower 16 bits of the Multi-Cast Hash Table. - When a bit is set to '1', the corresponding multicast frame hashed to that bit is accepted and forwarded to the host. - If the bit is set to '0', the multicast frame is dropped. (Default: 16'b0)

15.3.5.39 MAC Multicast Hash Table#3 Register

The MAC Multicast Hash Table Register (#1 to #4) forms a 64-bit multicast hash table used for group address filtering.

Hash Filtering Process:

- The destination address (DA) field of the incoming frame is passed through the CRC logic.
- The upper 6 bits of the CRC register are used to index the hash table:
  - A value of 6'b000000 selects bit [0] of the hash table.
  - A value of 6'b111111 selects bit [63] of the hash table.
- When a multicast frame is hashed into the table, if the corresponding bit is '1', the frame is accepted. If it is '0', the frame is rejected.

Offset: 0x0158				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	HASH_TABLE_47_TO_32	RW	0x0	Hash Table Bits [47:32] These bits represent the lower 16 bits of the Multi-Cast Hash Table. - When a bit is set to '1', the corresponding multicast frame hashed to that bit is accepted and forwarded to the host. - If the bit is set to '0', the multicast frame is dropped. (Default: 16'b0)

15.3.5.40 MAC Multicast Hash Table#4 Register

The MAC Multicast Hash Table Register (#1 to #4) forms a 64-bit multicast hash table used for group address filtering.

Hash Filtering Process:

- The destination address (DA) field of the incoming frame is passed through the CRC logic.
- The upper 6 bits of the CRC register are used to index the hash table:
  - A value of 6'b000000 selects bit [0] of the hash table.
  - A value of 6'b111111 selects bit [63] of the hash table.
- When a multicast frame is hashed into the table, if the corresponding bit is '1', the frame is accepted. If it is '0', the frame is rejected.

Offset: 0x015C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	HASH_TABLE_63_TO_48			Hash Table Bits [63:48] These bits represent the lower 16 bits of the Multi-Cast Hash Table. - When a bit is set to '1', the corresponding multicast frame hashed to that bit is accepted and forwarded to the host. - If the bit is set to '0', the multicast frame is dropped. (Default: 16'b0)

15.3.5.41 MAC Flow-Control Register

This register manages Flow-Control features, including PAUSE frame generation and reception, for the EMAC Core in Full-Duplex mode.

- In **Full-Duplex mode**, Flow-Control is enabled.
- In **Half-Duplex mode**, Flow-Control is disabled, and received PAUSE frames are treated as normal data frames without decoding.

Offset: 0x0160				
Bits	**Field**	**Type**	**Reset**	**Description**
15:8	ENABLE_PRIORITIES	RW	0x0	Enabled Priorities This field specifies which priorities are active for Priority PAUSE flow control generation and detection. - The XOFF/XON control signals from user logic are monitored only for the enabled priorities. - During reception, only the enabled priorities are considered for the Time-Vector field. (Default: 8'b0)
7	Reserved			Reserved for future use
6	PRIORITY_PAUSE_DECODE_ENABLE	RW	0	Priority PAUSE Decode Enable This field enables or disables Priority PAUSE Flow-Control in the EMAC Core. - When enabled (set to '1'): 1. The EMAC Core decodes incoming frames for Priority PAUSE Control Frames, as per IEEE 802.3. 2. If a valid Priority PAUSE Control Frame is received, the timers for enabled priorities are extracted. 3. A corresponding XOFF notification is sent to the user logic

Offset: 0x0160				
Bits	**Field**	**Type**	**Reset**	**Description**
				while the counter remains nonzero. - When disabled (reset to '0'): 1. The EMAC Core does not decode Priority PAUSE Control Frames. **Note.** This bit is valid only in Full-Duplex mode. (Default: 1’b0)
5	PRIORITY_FC_GENERATION_ENABLE	RW	0	Priority Flow-Control Generation Enable This field enables or disables the transmission of Priority PAUSE Control Frames by the EMAC Core. - When enabled (set to '1'): 1. The EMAC Core transmits Priority PAUSE Control Frames, which can be triggered either by software control or external Priority XON/XOFF controls. 2. The parameters for the generated Priority PAUSE Frame are controlled independently. - When disabled (reset to '0'): 1. The EMAC Core does not generate any Priority PAUSE Control Frames. **Note.** - This bit is valid only in Full-Duplex mode. - PAUSE and Priority PAUSE functionalities are mutually exclusive, so only one should be enabled at a time. (Default: 1’b0)
4	BLOCK_PAUSE_FRAMES	RW	0	Block Pause Frames This field controls whether received PAUSE Control Frames are forwarded to the Host Memory. - When enabled (set to '1'): 1. The EMAC Core decodes the received PAUSE Control Frames but does not forward them to Host Memory. 2. These frames are purged from the Receive FIFO. - When disabled (reset to '0'): 1. The received PAUSE Control Frames are decoded and transmitted to the Host Memory like normal data frames. (Default: 1’b0)
3	MULTICAST_MODE	RW	0	Flow-Control Multi-Cast Mode This field determines the Destination Address (DA) used in generated PAUSE Flow Control frames. - When set: 1. The DA field in the PAUSE Flow Control frames is set to the reserved multicast address: 01:80:C2:00:00:01

Offset: 0x0160				
Bits	**Field**	**Type**	**Reset**	**Description**
				- When reset: 1. The DA field uses the MAC Flow-Control Destination Address stored in the High/Med/Low Registers. 2. This address is typically the unicast address of the device at the other end of the cable. (Default: 1'b0)
2	AUTO_FC_GENERATION_ENABLE	RW	0	Auto Flow Control Generation Enable - When set: The EMAC core generates automatic PAUSE control frames when the receive FIFO fill level crosses the configured high or low threshold values (after debouncing). 1. If the receive FIFO fill level exceeds the high threshold, a PAUSE control frame is generated with the PauseTime value set to the MAC Auto High Pause Time Register. 2. If the receive FIFO fill level falls below the low threshold, a PAUSE control frame is generated with the PauseTime value set to the MAC Auto Low Pause Time Register. - When reset: Automatic generation of PAUSE control frames is disabled. (Default: 1'b0)
1	FC_GENERATION_ENABLE	RW	0	Flow-Control Generation Enable This field controls whether the EMAC Core can transmit PAUSE Control frames. - When set: 1. The EMAC Core transmits PAUSE Control frames in one of the following ways: - Software control - Auto-generation mode (when the Receive FIFO fill level crosses the High or Low Threshold) - External XON/XOFF controls 2. The parameters for PAUSE frame generation are managed separately. - When reset: 1. The EMAC Core does not generate PAUSE Control frames. **Note.** This feature is only valid in Full-Duplex mode. (Default: 1'b0)
0	FC_DECODE_ENABLE	RW	0	Flow-Control Decode Enable This field controls whether the EMAC Core processes incoming PAUSE Control frames. - When set: 1. The EMAC Core decodes PAUSE Control frames according

Offset: 0x0160				
Bits	**Field**	**Type**	**Reset**	**Description**
				to the IEEE 802.3 Specification. 2. If a valid PAUSE Control frame is received, the EMAC Core halts user data transmission for the duration specified in the PAUSE_TIME field. - When reset: 1. The Flow-Control Operation in the EMAC Core is disabled and the Core does not decode the frames for PAUSE Control Frames **Note.** This feature is only valid in Full-Duplex mode. (Default: 1'b0)

15.3.5.42 MAC Flow-Control PAUSE Frame Generate Register

This register allows to trigger the generation of **PAUSE Control Frames**.

- The **contents** of the generated PAUSE Frame are taken from **various Flow-Control registers**.
- It includes a **handshake mechanism** between **software and the EMAC Core** to:
  - **Initiates the creation of a PAUSE Control Frame**
  - **Indicate** when the PAUSE Frame generation is complete

Offset: 0x0164				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved			Reserved for future use
1	GENERATE_PRIORITY_PAUSE_FRAME	RW	0	Generate Priority PAUSE Frame This bit is set by the software to instruct the EMAC Core to generate a Priority PAUSE Control Frame. - Before setting this bit, software must ensure it is cleared. - Once set, the EMAC Core attempts to generate the frame immediately: 1. If the MAC Transmitter is idle, the frame is generated right away. 2. If the MAC Transmitter is busy transmitting a user data frame, the Priority PAUSE Control Frame is queued. It will be transmitted once the current frame completes and the IFG is met. - The software must continuously monitor this bit to check if the Priority PAUSE Frame transmission is complete. - Once the frame is successfully transmitted, the EMAC Core automatically clears this bit.

Offset: 0x0164				
Bits	**Field**	**Type**	**Reset**	**Description**
				Note. - The generated Priority PAUSE Frame is based on the Enabled Priorities. - This bit is valid only in Full-Duplex mode and when the Priority PAUSE Flow-Control Generation Enable bit in the MAC Flow-Control Register is set. (Default: 1'b0)
0	GENERATE_PAUSE_FRAME	RW	0	Generate PAUSE Frame This bit is set by the software to instruct the EMAC Core to generate a PAUSE Control Frame. - Before setting this bit, the software must ensure it is cleared. - Once set, the EMAC Core attempts to generate the frame immediately: 1. If the MAC Transmitter is idle, the frame is generated right away. 2. If the MAC Transmitter is busy transmitting a user data frame, the PAUSE Control Frame is queued. It will be transmitted once the current frame completes and the IFG is met. - The software must continuously monitor this bit to check if the PAUSE Frame transmission is complete. - Once the frame is successfully transmitted, the EMAC Core automatically clears this bit. **Note.** This bit is valid only in Full-Duplex mode and when the Flow-Control Generation Enable bit in the MAC Flow-Control Register is set. (Default: 1'b0)

15.3.5.43 MAC Flow-Control Source Address High Register

The MAC Flow-Control Source Address High Register stores the first two bytes of the Source Address (in Canonical Format) used in generated PAUSE and Priority PAUSE Control Frames.

- In Canonical Format, bytes are transmitted serially on the Ethernet interface.
- The remaining four bytes of the 48-bit Source Address are stored in:
  - MAC Flow-Control Source Address Med Register (middle two bytes).
  - MAC Flow-Control Source Address Low Register (last two bytes).

Offset: 0x0168				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_FC_SOURCE_ADDRESS_02_BYTE	RW	0x0	MAC Flow-Control Source Address Second Byte This register holds the second byte of the Source Address in Canonical Format, used in generated PAUSE and Priority PAUSE Control Frames. - It represents the second byte transmitted on the Ethernet line as part of the Source Address (SA) field in Flow-Control Frames. (Default : 8'b0)
7:0	MAC_FC_SOURCE_ADDRESS_01_BYTE	RW	0x0	MAC Flow-Control Source Address First Byte This register holds the first byte of the Source Address in Canonical Format, used in generated PAUSE and Priority PAUSE Control Frames. - It represents the first byte transmitted on the Ethernet line as part of the Source Address (SA) field in Flow-Control Frames. (Default : 8'b0)

15.3.5.44 MAC Flow-Control Source Address Med Register

The MAC Flow-Control Source Address Med Register stores the first two bytes of the Source Address (in Canonical Format) used in generated PAUSE and Priority PAUSE Control Frames.

- In Canonical Format, bytes are transmitted serially on the Ethernet interface.
- The remaining four bytes of the 48-bit Source Address are stored in:
  - MAC Flow-Control Source Address High Register (first two bytes).
  - MAC Flow-Control Source Address Low Register (last two bytes).

Offset: 0x016C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_FC_SOURCE_ADDRESS_04_BYTE	RW	0x0	MAC Flow-Control Source Address Fourth Byte This register holds the fourth byte of the Source Address in Canonical Format, used in generated PAUSE and Priority PAUSE Control Frames. - It represents the fourth byte transmitted on the Ethernet line as part of the Source Address (SA) field in Flow-Control Frames.

Offset: 0x016C				
Bits	**Field**	**Type**	**Reset**	**Description**
				(Default : 8'b0)
7:0	MAC_FC_SOURCE_ADDRESS_03_BYTE	RW	0x0	MAC Flow-Control Source Address Third Byte This register holds the third byte of the Source Address in Canonical Format, used in generated PAUSE and Priority PAUSE Control Frames. - It represents the third byte transmitted on the Ethernet line as part of the Source Address (SA) field in Flow-Control Frames. (Default : 8'b0)

15.3.5.45 MAC Flow-Control Source Address Low Register

The MAC Flow-Control Source Address High Register stores the last two bytes of the Source Address (in Canonical Format) used in generated PAUSE and Priority PAUSE Control Frames.

- In Canonical Format, bytes are transmitted serially on the Ethernet interface.
- The remaining four bytes of the 48-bit Source Address are stored in:
  - MAC Flow-Control Source Address High Register (first two bytes).
  - MAC Flow-Control Source Address Med Register (middle two bytes).

Offset: 0x0170				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_FC_SOURCE_ADDRESS_06_BYTE	RW	0x0	MAC Flow-Control Source Address Sixth Byte This register holds the sixth byte of the Source Address in Canonical Format, used in generated PAUSE and Priority PAUSE Control Frames. - It represents the sixth byte transmitted on the Ethernet line as part of the Source Address (SA) field in Flow-Control Frames. (Default : 8'b0)
7:0	MAC_FC_SOURCE_ADDRESS_05_BYTE	RW	0x0	MAC Flow-Control Source Address Fifth Byte This register holds the fifth byte of the Source Address in Canonical Format, used in generated PAUSE and Priority PAUSE Control Frames. - It represents the fifth byte transmitted on the Ethernet line as part of the Source Address (SA) field in Flow-Control Frames. (Default : 8'b0)

15.3.5.46 MAC Flow-Control Dst. Address High Register

The MAC Flow-Control Destination Address High Register stores the first two bytes of the Destination Address (in Canonical Format) used in generated PAUSE and Priority PAUSE Control Frames when a Unicast Address is applied in the DA field (i.e., when Multi-Cast mode is disabled).

- In Canonical Format, bytes are transmitted serially on the Ethernet interface.
- The remaining four bytes of the 48-bit Destination Address are stored in:
  - MAC Flow-Control Destination Address Med Register (middle two bytes).
  - MAC Flow-Control Destination Address Low Register (last two bytes).

Offset: 0x0174				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_FC_DESTINATION_ADDRESS_02_BYTE	RW	0x0	MAC Flow-Control Destination Address Second Byte The register stores the second byte of the Destination Address (in Canonical Format) used in generated PAUSE Control Frames. - This is the second byte transmitted on the Ethernet line as part of the DA field in PAUSE Control Frames. (Default : 8'b0)
7:0	MAC_FC_DESTINATION_ADDRESS_01_BYTE	RW	0x0	MAC Flow-Control Destination Address First Byte The register stores the first byte of the Destination Address (in Canonical Format) used in generated PAUSE Control Frames. - This is the first byte transmitted on the Ethernet line as part of the DA field in PAUSE Control Frames. (Default : 8'b0)

15.3.5.47 MAC Flow-Control Dst. Address Med Register

The MAC Flow-Control Destination Address Med Register stores the middle two bytes of the Destination Address (in Canonical Format) used in generated PAUSE and Priority PAUSE Control Frames when a Unicast Address is applied in the DA field (i.e., when Multi-Cast mode is disabled).

- In Canonical Format, bytes are transmitted serially on the Ethernet interface.
- The remaining four bytes of the 48-bit Destination Address are stored in:
  - MAC Flow-Control Destination Address High Register (first two bytes).
  - MAC Flow-Control Destination Address Low Register (last two bytes).

Offset: 0x0178				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_FC_DESTINATION_ADDRESS_04_BYTE	RW	0x0	MAC Flow-Control Destination Address Fourth Byte The register stores the fourth byte of the Destination Address (in Canonical Format) used in generated PAUSE Control Frames. - This is the fourth byte transmitted on the Ethernet line as part of the DA field in PAUSE Control Frames. (Default : 8'b0)
7:0	MAC_FC_DESTINATION_ADDRESS_03_BYTE	RW	0x0	MAC Flow-Control Destination Address Third Byte The register stores the third byte of the Destination Address (in Canonical Format) used in generated PAUSE Control Frames. - This is the third byte transmitted on the Ethernet line as part of the DA field in PAUSE Control Frames. (Default : 8'b0)

15.3.5.48 MAC Flow-Control Dst. Address Low Register

The MAC Flow-Control Destination Address High Register stores the last two bytes of the Destination Address (in Canonical Format) used in generated PAUSE and Priority PAUSE Control Frames when a Unicast Address is applied in the DA field (i.e., when Multi-Cast mode is disabled).

- In Canonical Format, bytes are transmitted serially on the Ethernet interface.
- The remaining four bytes of the 48-bit Destination Address are stored in:
  - MAC Flow-Control Destination Address High Register (first two bytes).
  - MAC Flow-Control Destination Address Med Register (middle two bytes).

Offset: 0x017C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	MAC_FC_DESTINATION_ADDRESS_06_BYTE	RW	0x0	MAC Flow-Control Destination Address Sixth Byte The register stores the sixth byte of the Destination Address (in Canonical Format) used in generated PAUSE Control Frames. - This is the sixth byte transmitted on the Ethernet line as part of the DA field in PAUSE Control Frames. (Default : 8'b0)
7:0	MAC_FC_DESTINA	RW	0x0	MAC Flow-Control Destination Address Fifth Byte

Offset: 0x017C				
Bits	**Field**	**Type**	**Reset**	**Description**
	TION_ADDRESS_05_BYTE			The register stores the fifth byte of the Destination Address (in Canonical Format) used in generated PAUSE Control Frames. - This is the fifth byte transmitted on the Ethernet line as part of the DA field in PAUSE Control Frames. (Default : 8'b0)

15.3.5.49 MAC Flow-Control Pause Time Value Register

This register contains the **Pause-Time** value used in the generated **PAUSE Control Frame**.

- The value is used when the **Generate (Priority) PAUSE Frame** bit in the **MAC Flow-Control (Priority) PAUSE Frame Generate Register** is set.

Offset: 0x0180				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_FC_PAUSE_TIME	RW	0x0	MAC Flow-Control Time This field stores the Pause-Time value used in the generated (Priority) PAUSE Control Frame. - The 16-bit value is transmitted as per the IEEE 802.3 specification, with the most significant byte first, followed by the least significant byte. (Default: 16'b0)

15.3.5.50 MAC Flow-Control Auto Gen Hi Pause Time Value Register

This register contains the Pause-Time value used in the generated PAUSE Control Frame when the Receive FIFO fill level exceeds the threshold set in the Receive FIFO Auto Gen Hi Threshold Register.

Offset: 0x0184				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_FC_HIGH_PAUSE_TIME	RW	0x0	MAC Flow-Control Auto Gen High Pause Time This field stores the Pause-Time value used in the generated PAUSE control frame when the Receive FIFO Fill Level exceeds the programmed high threshold value.

Offset: 0x0184				
Bits	**Field**	**Type**	**Reset**	**Description**
				- The 16-bit value is transmitted as per the IEEE 802.3 specification, with the most significant byte first, followed by the least significant byte. (Default: 16'b0)

15.3.5.51 MAC Flow-Control Auto Lo Pause Time Value Register

This register contains the **Pause-Time** value used in the generated **PAUSE Control Frame** when the **Receive FIFO** fill level falls below the programmed **Low Threshold** value, which had previously crossed the **High Threshold** value.

Offset: 0x0188				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_FC_LOW_PAUSE_TIME	RW	0x0	MAC Flow-Control Pause Time This field stores the Pause-Time value used in the generated PAUSE Control Frame when the Receive FIFO Fill Level drops below the programmed Low Threshold value, after previously crossing the programmed High Threshold value. - The 16-bit value is transmitted as per the IEEE 802.3 specification, with the most significant byte first, followed by the least significant byte. (Default: 16'b0)

15.3.5.52 MAC Flow-Control Auto Pause Frame Gen Hi Threshold Register

This register stores the high threshold value for the Receive FIFO Fill Level.

- When the Receive FIFO Fill Level reaches or exceeds this threshold value, a PAUSE control frame is generated.
- The Pause-Time value used in the generated PAUSE control frame is the value programmed in the MAC Flow-Control Auto Gen High Pause Time Register.

Offset: 0x018C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use

Offset: 0x018C				
Bits	**Field**	**Type**	**Reset**	**Description**
15:0	MAC_FC_PAUSE_HIGH_THRESHOLD	RW	0xFFFF	This field stores the High Threshold value for the Receive FIFO Fill Level. When the fill level reaches or exceeds this threshold, a PAUSE Control Frame is generated using the Pause-Time value programmed in the MAC Flow-Control Auto Gen Hi Pause Time Value Register. - The value represents the MAC Receive FIFO fill level. - Each location corresponds to either an 8-byte-wide FIFO or a 16-byte-wide FIFO. (Default: 16'hFFFF)

15.3.5.53 MAC Flow-Control Auto Pause Frame Gen Lo Threshold Register

This register stores the low threshold value for the Receive FIFO Fill Level.

- When the Receive FIFO Fill Level falls below this threshold value, a PAUSE control frame is generated.
- The Pause-Time value used in the generated PAUSE control frame is the value programmed in the MAC Flow-Control Auto Gen High Pause Time Register.

Offset: 0x0190				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_FC_PAUSE_LOW_THRESHOLD	RW	0x0	MAC Flow-Control Auto Pause Low Threshold This field contains the Receive FIFO Low Threshold value used to generate the PAUSE Control Frame when the Receive FIFO Fill Level falls below this threshold. When this happens, the PAUSE Control Frame is generated using the Pause Time value programmed in the MAC Flow-Control Auto Gen Lo Pause Time Value Register. - The threshold value is based on the MAC Receive FIFO fill level. - Each location corresponds to either an 8-byte wide FIFO or a 16-byte wide FIFO. (Default: 16'b0)

15.3.5.54 MAC MDIO Control Register

The MAC MDIO Control Register is used to control and generate MDIO frames to the external PHY (Physical Layer) Controller chip. This register contains various fields of the MDIO frame, including:

- PHY Address
- Register Address
- Etc.

Offset: 0x01A0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15	START_MDIO_TRANS	RW	0	Start MDIO Transaction This bit is set by the software to instruct the EMAC Core to generate an MDIO frame. - Before setting this bit, the software must ensure it is cleared. - Once set, the EMAC Core generates the MDIO frame based on the register fields: 1. For write transactions, the data from the MDIO Data Register is used. 2. For read transactions, the data read from the PHY is written into the MDIO Data Register. - The software must continuously poll this bit to check if the MDIO frame transmission is completed. - After the MDIO frame transmission is complete, the EMAC Core automatically clears this bit. **Note.** For read transactions, the software should read from the MDIO Data Register after the EMAC clears this bit. (Default: 1'b0)
14:11	Reserved			Reserved for future use
10	MDIO_READ_WRITE	RW	0	MDIO Read/Write This bit determines the type of MDIO transaction to be performed. - Encodings: 1. 1'b1: Read transaction 2. 1'b0: Write transaction (Default)
9:5	REGISTER_ADDRESS	RW	0x0	Register Address This field contains the Register Address field that is used in the MDIO Frame. It can address up to 32 registers in the addressed PHY device. (Default: 5'b0)
4:0	PHY_ADDRESS	RW	0x0	PHY Address This field contains the PHY Address field that is used in

Offset: 0x01A0				
Bits	**Field**	**Type**	**Reset**	**Description**
				the MDIO Frame. It can address up to 32 PHY devices. (Default: 5'b0)

15.3.5.55 MAC MDIO Data Register

The register stores the 16-bit data used in MDIO transfers:

- For **MDIO write transfers**, it contains the 16-bit data to be written to the addressed PHY register.
- For **MDIO read transfers**, it holds the 16-bit data read from the addressed PHY register.

Offset: 0x01A4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MDIO_DATA	RW	0x0	MDIO Data This field holds the 16-bit value: - In MDIO read transfers, it contains the data read from the PHY. - In MDIO write transfers, it contains the data to be written to the PHY. (Default: 16'b0)

15.3.5.56 MAC Receive StatCtr. Control Register

This register is used to initiate the reading of the Receive Statistic Counters in the EMAC Core.

- The specific counter to be read is programmed through this register.
- After the counter read operation is complete, the resulting data is provided in the MAC Receive StatCtr. Data High/Low Registers for further processing.

Offset: 0x01A8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15	START_RX_COUNTER_READ	RW	0x0	Start Receive Counter Read This bit is set by the software to instruct the EMAC Core to read the specified Receive Statistic Counter from the internal memory. - Before setting this bit, software must ensure that it is cleared.

Offset: 0x01A8				
Bits	**Field**	**Type**	**Reset**	**Description**
				- Once set, the EMAC Core immediately attempts to read the addressed counter. - The software should continuously poll this bit to determine when the counter read operation is complete. - Once the operation is finished, the EMAC Core automatically clears this bit. - After the bit is cleared, software should read the MAC Receive StatCtr. Data High/Low Registers to retrieve the counter value. (Default: 1'b0)
14:5	Reserved			Reserved for future use
4:0	RX_COUNTER_NUMBER	RO	0x0	Receive Counter Number This field contains the Receive Statistic Counter Number that is to be read. - The Counter is a 32-bit rollover counter. (Default: 5’b0)

15.3.5.57 MAC Receive StatCtr. Data High Register

This register contains the upper 16-bits of the 32-bit counter data that was read in the previous read operation.

Offset: 0x01AC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	RX_STATCTR_DATA_HIGH	RO	0x0	Receive StatCtr. Data High This field contains the upper 16-bits of the 32-bit Receive Counter that was read in the previous operation. (Default: 16’b0)

15.3.5.58 MAC Receive StatCtr. Data Low Register

This register contains the lower 16-bits of the 32-bit counter data that was read in the previous read operation.

Offset: 0x01B0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	RX_STATCTR_DATA_LOW	RO	0x0	Receive StatCtr. Data Low This field contains the lower 16-bits of the 32-bit Receive Counter that was read in the previous operation. (Default: 16'b0)

15.3.5.59 MAC Transmit StatCtr. Control Register

This register is used to initiate the reading of the Transmit Statistic Counters in the EMAC Core.

- The specific counter to be read is programmed through this register.
- After the counter read operation is complete, the resulting data is provided in the MAC Transmit StatCtr. Data High/Low Registers for further processing

Offset: 0x01B4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15	START_TX_COUNTER_READ	RW	0	Start Transmit Counter Read This bit is set by the software to instruct the EMAC Core to read the specified Transmit Statistic Counter from the internal memory. - Before setting this bit, software must ensure that it is cleared. - Once set, the EMAC Core immediately attempts to read the addressed counter. - The software should continuously poll this bit to determine when the counter read operation is complete. - Once the operation is finished, the EMAC Core automatically clears this bit. - After the bit is cleared, software should read the MAC Transmit StatCtr. Data High/Low Registers to retrieve the counter value. (Default: 1'b0)
14:5	Reserved			Reserved for future use
4:0	TX_COUNTER_NUM	RO	0x0	Transmit Counter Number

Offset: 0x01B4				
Bits	**Field**	**Type**	**Reset**	**Description**
	BER			This field contains the Transmit Statistic Counter Number that is to be read. - The Counter is a 32-bit rollover counter. (Default: 5'b0)

15.3.5.60 MAC Transmit StatCtr. Data High Register

This register contains the upper 16-bits of the 32-bit counter data that was read in the previous read operation.

Offset: 0x01B8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	TX_STATCTR_DATA_HIGH	RO	0x0	Transmit StatCtr. Data High This field contains the upper 16-bits of the 32-bit Receive Counter that was read in the previous operation. (Default: 16'b0)

15.3.5.61 MAC Transmit StatCtr. Data Low Register

This register contains the lower 16-bits of the 32-bit counter data that was read in the previous read operation.

Offset: 0x01B0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	TX_STATCTR_DATA_LOW	RO	0x0	Transmit StatCtr. Data Low This field contains the lower 16-bits of the 32-bit Receive Counter that was read in the previous operation. (Default: 16'b0)

15.3.5.62 MAC Transmit FIFO AlmostFull Threshold Register

This register contains the threshold value used by the Transmit FIFO in the EMAC Core to trigger the "Almost Full" condition to the DMA.

- When the Transmit FIFO reaches this threshold, it signals the DMA to stop fetching the next burst of data from the Host Memory, as the FIFO is nearing full capacity.
- The value programmed in this register represents the number of empty FIFO locations that must remain before the “Almost Full” condition is declared.

Offset: 0x01C0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:14	Reserved			Reserved for future use
13:0	TX_FIFO_AF	RW	0x0	Transmit FIFO AlmostFull Threshold This field specifies the number of occupied locations (each 32-bit wide) in the Transmit FIFO that triggers the "Almost Full" condition within the EMAC Core. - Once the threshold is reached, the Transmit FIFO asserts the Almost Full condition, signaling the Transmit DMA to stop fetching the next burst of data from Host Memory. - For proper operation of the EMAC Core, the value programmed in this field should be FIFO_SIZE - 14'h0008 where, FIFO_SIZE represents the depth of the Transmit FIFO in 32-bit words. (Default: 14'b0)

15.3.5.63 MAC Transmit Packet Start Threshold Register

This field specifies the threshold value for the number of bytes in the Transmit FIFO before a packet is transmitted onto the MII/GMII Interface.

- The EMAC Core waits for at least the specified number of bytes or the end of the packet in the Transmit FIFO before initiating transmission.
- Setting an appropriate value in this register helps reduce Transmit Underflow conditions, particularly when the Host Bus has high bandwidth demands.

Offset: 0x01C4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:14	Reserved			Reserved for future use
13:0	TX_PACKET_START_THRESHOLD	RW	0x0	Transmit Packet Start Threshold This field specifies the number of bytes that must be present in the Transmit FIFO before the packet is transmitted onto the MII/GMII Interface. - The EMAC Core waits until the Transmit FIFO contains at least the specified number of bytes of

Offset: 0x01C4				
Bits	**Field**	**Type**	**Reset**	**Description**
				data, or the End of Packet is in the FIFO, before initiating transmission. - Setting a higher value helps reduce Transmit Underflow situations, particularly when the DMA cannot fetch data from the Host Memory fast enough. Suggested values based on Ethernet Interface Speed (assuming the Host bandwidth is sufficiently available for DMA to fetch data from Host Memory): - 10 Mbps: 64 Bytes - 100 Mbps: 128 Bytes - 1000 Mbps: 1024 Bytes A value of 1518 sets the Transmit MAC to operate in Store and Forward mode for all packets, regardless of the Ethernet speed. (Default: 14'b0)

15.3.5.64 MAC Receive Packet Start Threshold Register

This field specifies the number of bytes that must be present in the Receive FIFO before the received packet is transferred to Host Memory.

- If the packet is smaller than the value programmed in this register, it will be dropped from the Receive FIFO and not transferred to Host Memory.
- This mechanism helps prevent the transfer of small packets, such as Runts and Fragments, optimizing CPU resources by ensuring only valid, sufficiently-sized packets are passed to the Host.

Offset: 0x01C8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:14	Reserved			Reserved for future use
13:0	RX_PACKET_START_THRESHOLD	RW	0xE	Receive Packet Start Threshold This field defines the minimum number of bytes required in a received packet before it is transferred from the MII/GMII Interface to Host Memory. - If the packet size is smaller than the programmed value, it is treated as a Runt/Fragment and dropped from the Receive FIFO without being transferred. - A value of 64 in this field ensures that all Runt frames are filtered out and not transferred to Host Memory. - A value of 0 configures the EMAC Core to transfer all frames, regardless of size or DA field, to Host

Offset: 0x01C8				
Bits	**Field**	**Type**	**Reset**	**Description**
				Memory. - A minimum value of 12 is necessary for the Address Filtering logic to function and filter out frames not addressed to this device. Setting a value less than 12 disables the Address Filtering logic. - If the "Store And Forward" bit in the MAC Receive Control Register is set, this register's value is ignored, and the EMAC operates in store-and-forward mode for packet reception. (Default: 14'h000E)

15.3.5.65 MAC Transmit FIFO AlmostEmpty Threshold Register

This register contains the threshold value used by the Transmit FIFO in the EMAC Core to generate the Almost Empty condition for the DMA.

- When the number of occupied locations in the Transmit FIFO is less than or equal to the programmed threshold value, the Almost Empty condition is triggered.

Offset: 0x01CC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved			Reserved for future use
7:0	TX_FIFO_AE	RW	0x10	Transmit FIFO AlmostEmpty Threshold This field contains the threshold value for the Transmit FIFO in the EMAC Core. - When the number of occupied locations (each 32-bit or 64-bit wide) in the Transmit FIFO equals or falls below this threshold, the Transmit FIFO Almost Empty signal is generated to the DMA. (Default: 8'h10)

15.3.5.66 MAC Transmit FIFO Space Available Hi Threshold Register

This register holds the threshold value used by the Transmit FIFO in the EMAC Core.

- When the FIFO fill level reaches or exceeds this value, the Space Available indication to the DMA is deasserted, signaling that there is insufficient space for additional data

Offset: 0x01D0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	TX_FIFO_SPACE_AVAILABLE_HIGH_THRESHOLD	RW	0x100	Transmit FIFO Space Available Hi Threshold This field contains the threshold value. When the number of locations (each 32-bit or 64-bit wide) occupied in the Transmit FIFO equals or exceeds this value, the Space Available signal is deasserted. (Default: 16'h0100)

15.3.5.67 MAC Transmit FIFO Space Available Lo Threshold Register

This register contains the threshold value that is used by the Transmit FIFO in the EMAC Core to assert the Space Available indication to DMA when the Fill level equals to or dips this value.

This register contains the threshold value used by the Transmit FIFO in the EMAC Core.

- When the FIFO fill level equals or dips below this value, the Space Available indication is asserted to the DMA, signaling that there is available space for more data.

Offset: 0x01D4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	TX_FIFO_SPACE_AVAILABLE_LOW_THRESHOLD	RW	0x10	Transmit FIFO Space Available Lo Threshold This field contains the threshold value. When the number of locations (each 32-bit or 64-bit wide) occupied in the Transmit FIFO equals or dips below this value, the Space Available signal is deasserted. (Default: 16'h0010)

15.3.5.68 MAC Receive FIFO Packet Available Threshold#1 Register

This register contains the threshold values used by the Receive FIFO in the EMAC Core to assert the Packet Available indication to DMA when the Fill level equals to or exceeds the selected threshold value.

Offset: 0x01D8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use

Offset: 0x01D8				
Bits	**Field**	**Type**	**Reset**	**Description**
15:8	RX_FIFO_PACKET_AVAILABLE_THRESHOLD2	RW	0x20	Receive FIFO Packet Available Threshold2 This field specifies the threshold value for the receive FIFO. When the number of occupied locations (each 32-bit or 64-bit wide) in the receive FIFO equals or exceeds this value, the Receive Packet Available signal is generated (provided that Threshold Selection is set to Threshold2). (Default: 8’20)
7:0	RX_FIFO_PACKET_AVAILABLE_THRESHOLD1	RW	0x10	Receive FIFO Packet Available Threshold1 This field specifies the threshold value for the receive FIFO. When the number of occupied locations (each 32-bit or 64-bit wide) in the receive FIFO equals or exceeds this value, the Receive Packet Available signal is generated (provided that Threshold Selection is set to Threshold1). (Default: 8’10)

15.3.5.69 MAC Receive FIFO Packet Available Threshold#2 Register

This register contains the threshold values used by the Receive FIFO in the EMAC Core to assert the Packet Available indication to DMA when the Fill level equals to or exceeds the selected threshold value

Offset: 0x01DC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	Reserved			Reserved for future use
4	RX_FIFO_PACKET_AVAILABLE_THRESHOLD_SELECTION	RW	0	Receive FIFO Packet Available Threshold Selection This bit selects one of the two Receive FIFO Packet Available Threshold values for generating the Receive Packet Available signal. (Default: 8'b0) - 1'b0: Receive FIFO Packet Available Threshold1 Selected - 1'b1: Receive FIFO Packet Available Threshold2 Selected Currently, this value is not used by the DMA and does not need to be programmed.
3:0	RX_FIFO_PACKET_COUNT_THRESHOLD	RW	0x1	Receive FIFO Packet Count Threshold This field contains a threshold value. When the number of packets in the Receive FIFO equals to or

Offset: 0x01DC				
Bits	**Field**	**Type**	**Reset**	**Description**
				Exceeds this value, the Receive packet available signal is generated. (Default: 4'b0001)

15.3.5.70 MAC Status and IRQ Register

This register provides status and interrupt request (IRQ) information on various conditions that need to be monitored by the host software, which results from the MAC module in the EMAC Core. The IRQ bits are used to generate interrupts for the host.

Offset: 0x01E0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved			Reserved for future use
1	MAC_JABBER_IRQ	RW	0	MAC Jabber IRQ - When set: Indicates that a jabber condition has been detected while transferring a frame to the MII/GMII interface. This occurs when the frame length exceeds the value programmed in the Transmit Jabber Count Register. 1. The MAC truncates the frame at this point and forces EOP/ERR onto the MII/GMII interface. 2. Any remaining data bytes received from the host are ignored. - The interrupt is cleared by writing 1 to this bit. (Default: 1’b0)
0	MAC_UNDERRUN_IRQ	RW	0	MAC Underrun IRQ - When set: Indicates that an underrun condition has occurred while transferring a frame to the MII/GMII interface. This happens when the DMA is unable to sustain the data transfer rate required by the MAC. 1. The MAC forces EOP/ERR onto the MII/GMII interface. 2. Any remaining data bytes received from the host are ignored. - The interrupt is cleared by writing 1 to this bit. (Default: 1’b0)

15.3.5.71 MAC Interrupt Enable Register

This register is used to enable interrupts for various conditions in the MAC module that need to be monitored by the host software in the EMAC Core.

Offset: 0x01E4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved			Reserved for future use
1	JABBER_INTERRUPT_ENABLE	RW	0	Jabber Interrupt Enable - When set: The MAC Jabber IRQ will trigger an interrupt on the AHB/AXI Bus. - When cleared: The MAC Jabber IRQ will be blocked and will not generate an interrupt. (Default: 1'b0)
0	MAC_UNDERRUN_INTERRUPT_ENABLE	RW	0	MAC Underrun Interrupt Enable - When set: The MAC Underrun IRQ will trigger an interrupt on the AHB/AXI Bus. - When cleared: The MAC Underrun IRQ will be blocked and will not generate an interrupt. (Default: 1'b0)

15.3.5.72 MAC VLAN TPID#1 Register

Offset: 0x01E8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_VLAN_TPID1	RW	0x8100	MAC VLAN TPID#1 This field contains the 1st of the three custom TPID values that are used to detect VLAN fields. (Default: 16’h8100)

15.3.5.73 MAC VLAN TPID#2 Register

Offset: 0x01EC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_VLAN_TPID2	RW	0x9100	MAC VLAN TPID#2 This field contains the 2nd of the three custom TPID values that are used to detect VLAN fields. (Default: 16'h9100)

15.3.5.74 MAC VLAN TPID#3 Register

Offset: 0x01F0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	MAC_VLAN_TPID3	RW	0x88A8	MAC VLAN TPID#3 This field contains the 3rd of the three custom TPID values that are used to detect VLAN fields. (Default: 16'h88A8)

15.3.5.75 1588 Control Register

Tis register is used to control the main features of the 1588 Module and the Timestamping of the packets.

Offset: 0x0300				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved			Reserved for future use
5:3	RX_PTP_PKT_TYPE	RW	0x0	RX PTP Packet Type This field controls the selection of packets for timestamping on the receive path. A packet is timestamped and reported to software only if it is received error-free and matches all the programmed criteria. The following PTP packet types are monitored: - 3'b000: Timestamp Version #2 PTP packets in L2

Offset: 0x0300				
Bits	**Field**	**Type**	**Reset**	**Description**
				encapsulation (PTP packets as L2 payload) only. This includes Sync or Delay_Req packet types based on the PTP MessageID. - 3'b001: Timestamp Version #1 PTP packets in L4 encapsulation (PTP packets as UDP payload) only. This includes Sync or Delay_Req packet types based on the PTP MessageID. - 3'b010: Timestamp Version #2 PTP packets in either L2 or L4 encapsulation (PTP packets as L2 payload or UDP payload). - 3'b011 – 3'b111: Reserved. (Default: 3'b000**)**
2	RX_TIMESTAMP_EN	RW	0	RX Timestamping Enable - When set: Enables timestamping for incoming PTP packets that match the programmed criteria (such as MessageID, PTP EtherType, UDP Port, Version, etc.). - When cleared: Disables the timestamping feature for receive packets. (Default: 1'b0)
1	TX_TIMESTAMP_EN	RW	0	TX Timestamping Enable - When set: Enables timestamping for outgoing PTP packets under software control via the descriptor bit. - When cleared: Disables the timestamping feature for transmit packets. (Default: 1'b0)
0	Reserved			Reserved for future use

15.3.5.76 Increment Attributes Register

This register is used to control incrementing of the System Timer.

Offset: 0x0304				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	INCR_PERIOD	RW	0x0	Increment Period - This field sets the period for performing the Increment operation on the System Timer. It defines the number of clk_1588 clock cycles between each Increment operation. - When the value is 0x00, the increment operation is disabled. (Default: 8'h00)
23:0	INRC_VAL	RW	0x0	Increment Value - This field specifies the value to be added to the System Timer during each increment operation. The value is determined by the granularity and accuracy of the implemented System Timer. (Default: 24'h000000)

15.3.5.77 PTP Ethertype Register

This register is used to determine the Ethertype of the packets when the PTP message is using L2 Encapsulation

Offset: 0x0308				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	PTP_ETH_TYPE	RW	0x0	PTP Ethertype - Specifies the Ethertype in incoming packets for PTP message detection. - Applies when V2 PTP Packets use Layer 2 (L2) Encapsulation. - The register value is programmed or read in network order. - Typical Ethertype for PTP messages: 16'h88F7 (in network order). (Default: 16'h0000)

15.3.5.78 PTP Message ID Register

This register is used to define the PTP Message ID of the incoming PTP Packets that will be selected for timestamping.

Offset: 0x030C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved			Reserved for future use
7:0	PTP_MSG_ID	RW	0x0	PTP Message ID - Defines the PTP Message ID (or PTP Control for V1) for timestamping. - Used to select SYNC or DELAY_REQ packets based on device role (Master/Slave). - For monitoring V1 PTP messages, this field corresponds to PTP Control. - For monitoring V2 PTP messages, this field corresponds to PTP Message ID. (Default: 8'h00)

15.3.5.79 PTP UDP Port Register

This register is used to determine the destination UDP Port number of the packets when the PTP message is using L4 Encapsulation.

Offset: 0x0310				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:0	PTP_UDP_PORT	RW	0x0	PTP UDP Port - Specifies the destination UDP port in incoming packets for monitoring PTP messages. - Applies when V1/V2 PTP Packets use L4 Encapsulation. - The register value is programmed/read in Network order. - Typical PTP UDP ports: 16'h0319 or 16'h0320 (in network order). - Default: 16'h0000.

15.3.5.80 System Time Value (Lower) Register

The System Time Value (Lower & Upper) Registers are used to read the current value of the System Timer from the logic.

- The first register contains the lower 32 bits of the System Timer.
- The second register contains the upper 32 bits of the System Timer.

Offset: 0x0320				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SYS_TIME_LOW	RO	0x0	System Time Value (Lower) - This field holds the lower 32 bits of the System Timer value. - Read this register first before reading the System Time Value (Upper) register. - Reading this register latches the upper 32 bits into the upper register. (Default: 32'h0000_0000)

15.3.5.81 System Time Value (Upper) Register

The System Time Value (Lower & Upper) Registers are used to read the current value of the System Timer from the logic.

- The first register contains the lower 32 bits of the System Timer.
- The second register contains the upper 32 bits of the System Timer.

Offset: 0x0324				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SYSTEM_TIME_HI	RO	0x0	System Time Value (Upper) - This field holds the upper 32 bits of the System Timer value. - Read this register after reading the System Time Value (Lower) register. (Default: 32'h0000_0000)

15.3.5.82 System Time Adjust Control (Lower) Register

The System Time Adjust Control (Lower & Upper) Registers are used to read and adjust the System Timer.

- Software should first program the System Timer Adjust Control Lower Register with the adjustment parameters, and then, program the Upper Register to apply the adjustment.

Offset: 0x0328				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SYS_TIME_ADJ_LOW	RW	0x0	System Time Adjust Value (Lower) - This field defines the 32-bit adjustment value for the System Timer. - The type (positive or negative) and the magnitude of the adjustment are controlled by the System Time Adjust Control

Offset: 0x0328				
Bits	**Field**	**Type**	**Reset**	**Description**
				Upper Register. (Default: 32'h0000_0000)

15.3.5.83 System Time Adjust Control (Upper) Register

The System Time Adjust Control (Lower & Upper) Registers are used to read and adjust the System Timer.

- Software should first program the System Timer Adjust Control Lower Register with the adjustment parameters, and then, program the Upper Register to apply the adjustment.

Offset: 0x032C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SYS_TIME_ADJ_HI	RW	0x0	System Time Adjust Control (Upper) - This Write-only register is used to adjust the System Timer. - Adjustment occurs when software writes to this register. - Bit[31] determines the adjustment type: 1. 0 (Positive Adjustment): Adds the System Time Adjust Value to the current System Timer. 2. 1 (Negative Adjustment): Subtracts the System Time Adjust Value from the current System Timer.

15.3.5.84 Transmit Timestamp (Lower) Register

The Transmit Timestamp (Lower & Upper) Registers are used to read the current timestamp for the indicated transmit packet.

- The logic will not update this value (i.e., it will not perform timestamping on another packet) until these registers are read.
- These registers are read-only.
  - The first register contains the lower 32 bits of the transmit timestamp value.
  - The second register contains the upper 32 bits of the transmit timestamp value.

Offset: 0x0330				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TX_TIMESTAMP_LOW	RO	0x0	Transmit Timestamp (Lower) - This field contains the lower 32-bits of the Transmit Timestamp value. - This register is read first before reading the Transmit Timestamp (Upper) Register (Default: 32'h0000_0000)

15.3.5.85 Transmit Timestamp (Upper) Registers

The Transmit Timestamp (Lower & Upper) Registers are used to read the current timestamp for the indicated transmit packet.

- The logic will not update this value (i.e., it will not perform timestamping on another packet) until these registers are read.
- These registers are read-only.
  - The first register contains the lower 32 bits of the transmit timestamp value.
  - The second register contains the upper 32 bits of the transmit timestamp value

Offset: 0x0334				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TX_TIMESTAMP_HI	RO	0x0	Transmit Timestamp (Upper) - This field contains the upper 32-bits of the Transmit Timestamp value. - This register is read after the Transmit Timestamp (Lower) Register has been read. (Default: 32'h0000_0000)

15.3.5.86 Receive Timestamp (Lower) Registers

The Receive Timestamp (Lower & Upper) Registers are used to read the current timestamp for the indicated receive packet.

- The logic will not update this value (i.e., it will not perform timestamping on another packet) until these registers are read.
- These registers are read-only.
  - The first register contains the lower 32 bits of the receive timestamp value.
  - The second register contains the upper 32 bits of the receive timestamp value.

Offset: 0x0340				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_TIMESTAMP_LOW	RO	0x0	Receive Timestamp (Lower) - This field contains the lower 32-bits of the Receive Timestamp value. Software - This register is read first before reading the Receive Timestamp (Upper) Register (Default: 32'h0000_0000)

15.3.5.87 Receive Timestamp (Upper) Registers

The Receive Timestamp (Lower & Upper) Registers are used to read the current timestamp for the indicated receive packet.

- The logic will not update this value (i.e., it will not perform timestamping on another packet) until these registers are read.
- These registers are read-only.
  - The first register contains the lower 32 bits of the receive timestamp value.
  - The second register contains the upper 32 bits of the receive timestamp value

Offset: 0x0344				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_TIMESTAMP_HI	RO	0x0	Transmit Timestamp (Upper) - This field contains the upper 32-bits of the Receive Timestamp value. - This register is read after the Receive Timestamp (Lower) Register has been read. (Default: 32'h0000_0000)

15.3.5.88 Receive PTP Packet Attribute (Lower) Registers

The Receive PTP Packet Attributes (Lower/Middle/Upper) Registers are used to read the Source ID (SrcID) and Sequence ID (SeqID) of the received PTP message on which the timestamp operation was performed.

- The SrcID corresponds to the byte offset 20 to 29 in the PTP message.

Offset: 0x0348				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	PTP_SRC_ID_LOW	RO	0x0	SrcID (Lower) - This field contains the lower 16-bits of the SrcID from the timestamped Receive PTP Packet. - The SrcID is extracted from Byte Offset 28-29 in the PTP message packet. (Default: 32'h0000_0000)
15:0	PTP_SEQ_ID	RO	0x0	SeqID This field contains the 16-bit SeqID extracted from the timestamped Receive PTP Packet. (Default: 16'h0000)

15.3.5.89 Receive PTP Packet Attribute (Middle) Registers

The Receive PTP Packet Attributes (Lower/Middle/Upper) Registers are used to read the Source ID (SrcID) and Sequence ID (SeqID) of the received PTP message on which the timestamp operation was performed.

- The SrcID corresponds to the byte offset 20 to 29 in the PTP message.

Offset: 0x034C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	PTP_SRC_ID_MID	RO	0x0	SrcID (Middle) - This field contains the middle 32-bits of the SrcID from the timestamped Receive PTP Packet. - The SrcID is extracted from Byte Offset 24-27 in the PTP message packet. (Default: 32'h0000_0000)

15.3.5.90 Receive PTP Packet Attribute (Upper) Registers

The Receive PTP Packet Attributes (Lower/Middle/Upper) Registers are used to read the Source ID (SrcID) and Sequence ID (SeqID) of the received PTP message on which the timestamp operation was performed.

- The SrcID corresponds to the byte offset 20 to 29 in the PTP message.

Offset: 0x0350				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	TP_SRC_ID_HI	RO	0x0	SrcID (Upper) - This field contains the upper 32-bits of the SrcID from the timestamped Receive PTP Packet. - The SrcID is extracted from Byte Offset 20-23 in the PTP message packet. (Default: 16'h0000_0000)

15.3.5.91 1588 IRQ Register

The 1588 IRQ Register provides interrupt (IRQ) information related to receive and transmit timestamping conditions that need to be monitored by the host software. These conditions are generated by the 1588 module in the EMAC core.

- The IRQ bits in this register are used to generate interrupts to the host.

Offset: 0x0360				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved			Reserved for future use
1	RX_TIMESTAMP_IRQ	RW	0	Receive Timestamp Valid IRQ - When set, it indicates that a valid PTP Packet that matches the programmed parameters has been received and timestamped. - The value is available to read in the Receive Timestamp (Lower/Upper) Registers - The IRQ is cleared by writing 1 to this bit.

Offset: 0x0360				
Bits	**Field**	**Type**	**Reset**	**Description**
				(Default: 1'b0)
0	TX_TIMESTAMP_IRQ	RW	0	Transmit Timestamp Valid IRQ - When set, it indicates that a Transmit Packet has been timestamped - The value is available to read in the Transmit Timestamp (Lower/Upper) Registers. - The IRQ is cleared by writing 1 to this bit. (Default: 1'b0)

15.3.5.92 1588 Interrupt Enable Register

The 1588 Interrupt Enable Register is used to enable the generation of 1588 interrupts for various conditions that need to be monitored by the Host software. These conditions arise from the 1588 module in the EMAC Core.

Offset: 0x0340				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved			Reserved for future use
1	RX_TIMESTAMP_IRQ_ENABLE	RW	0	Receive Timestamp Interrupt Enable - When set, enables the Receive Timestamp IRQ to trigger an interrupt on the AHB/AXI Bus. - When cleared, the Receive Timestamp Valid IRQ is blocked from generating an interrupt. (Default: 1'b0)
0	TX_TIMESTAMP_IRQ_ENABLE	RW	0	Transmit Timestamp Interrupt Enable - When set, enables the Transmit Timestamp IRQ to trigger an interrupt on the AHB/AXI Bus. - When cleared, the Transmit Timestamp Valid IRQ is blocked from generating an interrupt. (Default: 1'b0)

15.3.5.93 AVB Control Register

The AVB Control Register configures and controls the operation of the AVB Transmit and Receive interface.

Offset: 0x0400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved			Reserved for future use
15:8	AVB_TX_THRESHOLD	RW	0x0	AVB Transmit Threshold Register - This register defines the transmit threshold for the AVB Transmit interface. - It sets the txavb_afull signal based on empty locations in the transmit buffer. - The programmed value determines the number of txavb_vld strobes that can be asserted after txavb_afull is triggered. (Default: 8'h00)
7:5	Reserved			Reserved for future use
4	AVB_RX_INTF_ENABLE	RW	0	Enable AVB Receive Interface - When set, the AVB Receive Interface is activated to process AVB frames. - Received AVB frames are forwarded to the AVB receive interface. - When cleared, the AVB Receive Interface is disabled, and no frames are identified. - For proper operation, enable this bit before receiving traffic. (Default: 1'b0)
3:1	Reserved			Reserved for future use
0	AVB_TX_INTF_ENABLE	RW	0	Enable AVB Transmit Interface - When set: 1. The AVB Transmit Interface is activated, prioritizing frames for transmission via TX_FIFO. 2. Frames are transmitted using the Credit-Based Shaper Algorithm. - When cleared: 1. The AVB Transmit Interface is disabled, and no frames are dequeued. 2. The Transmit FIFO on the AVB interface is cleared. For proper operation, enable this bit before sending traffic. (Default: 1'b0)

15.3.5.94 AVB Transmit SendSlope Register

This register configures the **SendSlope** variable, as defined by IEEE 802.1Qav.

It is used by the Rate Shaper to control the credit decrease rate.

Offset: 0x0404				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	Reserved			Reserved for future use
19:0	SEND_SLOPE	RW	0x800	SendSlope - The SendSlope variable, defined in IEEE 802.1Qav, represents the credit decrease rate in bits per second. - It is used by the Traffic Shaper to decrement the bucket value when AVB traffic is transmitted. (Default: 20'h00800)

15.3.5.95 AVB Transmit IdleSlope Register

This register configures the **IdleSlope** variable, as defined by IEEE 802.1Qav.

It is used by the Rate Shaper to control the idle rate.

Offset: 0x0408				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	Reserved			Reserved for future use
19:0	IDLE_SLOPE	RW	0x1800	IdleSlope - The IdleSlope variable, defined in IEEE 802.1Qav, represents the credit increase rate in bits per second. - It is used by the Traffic Shaper to decrement the bucket value when legacy traffic is transmitted and AVB traffic is queued. (Default: 20'h01800)

15.3.5.96 AVB RxPacket Filter1 Register

This register is used to program the filter value that is used to identify AVB traffic on the receive path.

Offset: 0x0410				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_FILTER1	RW	0x0	Filter1 - The value programmed in this register is compared against the extracted VLAN_TAG and ID fields. It helps identify AVB traffic on the receive path. - When the corresponding bit in the Mask register is 1, that bit is used for comparison with the received value. - For example, 1. To identify packets with a PCP (Priority Code Point) of 2 and a VLAN_ID of 2, program the value 0x8100_4002 in this register. (Default: 0x0000_0000)

15.3.5.97 AVB RxPacket Mask1 Register

This register is used to program the mask value.

- This mask enables the Filter1 value for identifying AVB traffic on the receive path

Offset: 0x0414				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_MASK1	RW	0x0	Mask1 - The Mask1 register is a bit-to-bit mask for the Filter1 register. - When the Mask bit is 1, the corresponding bit in the Filter1 register is used for comparison against the received VLAN TAG/ID field. - When the Mask bit is 0, the corresponding bit in Filter1 is ignored. - To compare the full 32-bit VLAN_TAG/ID field, program this register to 0xFFFF_FFFF. (Default: 0x0000_0000)

15.3.5.98 AVB RxPacket Filter2 Register

This register is used to program the filter value that is used to identify AVB traffic on the receive path.

Offset: 0x0418				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_FILTER2	RW	0x0	Filter2 - The value programmed in this register is compared against the extracted VLAN_TAG and ID fields. It helps identify AVB traffic on the

Offset: 0x0418				
Bits	**Field**	**Type**	**Reset**	**Description**
				receive path. - When the corresponding bit in the Mask register is 1, that bit is used for comparison with the received value. - For example, 1. To identify packets with a PCP (Priority Code Point) of 3 and a VLAN_ID of 2, program the value 0x8100_6002 in this register. (Default: 0x0000_0000)

15.3.5.99 AVB RxPacket Mask2 Register

This register is used to program the mask value.

- This mask enables the Filter2 value for identifying AVB traffic on the receive path

Offset: 0x041C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	RX_MASK2	RW	0x0	Mask2 - The Mask2 register is a bit-to-bit mask for the Filter2 register. - When the Mask bit is 1, the corresponding bit in the Filter2 register is used for comparison against the received VLAN TAG/ID field. - When the Mask bit is 0, the corresponding bit in Filter12 is ignored. - To compare the full 32-bit VLAN_TAG/ID field, program this register to 0xFFFF_FFFF. (Default: 0x0000_0000)

15.3.5.100 AVB HiLimit Register

This register is used to program the High Limit value for the credits that are accumulated when the legacy traffic is transmitted. This value is determined based on the MaxFrameSize of the Legacy Traffic.

Offset: 0x0420				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	HI_LIMT	RW	0xF0000000	HiLimit - This field defines the maximum accumulated credits when legacy traffic is transmitted. - It limits AV frame bursts. - It is based on the MaxFrameSize of legacy frames. (Default: 0xF000_0000, based on 2000-byte Max Frame Size)

15.3.6 Descriptor Details

15.3.6.1 Receive Descriptor

The RX descriptor consists of four DWORDs (RDES0–RDES3), containing receive status and control information, buffer sizes, and buffer or next-descriptor addresses, as summarized below.

DWORD	Main Contents	Software / Hardware Usage
RDES0	Receive status, Own, First Descriptor, Last Descriptor, Application Status, Frame/Packet Length	Written back mainly by GEMAC. The host initializes Own=1 to hand the descriptor to DMA. DMA clears Own after a frame is received or when the associated buffer is full. Frame Length and Application Status are valid only in the Last Descriptor.
RDES1	Control information, PTP flags, End of Ring, Second Address Chained, Buffer 1/2 Size	Initialized mainly by the host. PTP Packet and PTP Timestamped are initialized to 0. When IEEE 1588 PTP is enabled, GEMAC writes them back according to the matching result.
RDES2	Buffer Address 1	The host writes the physical address of receive buffer 1.
RDES3	Buffer Address 2 or Next Descriptor Address	In normal mode, this field is the physical address of buffer 2. When Second Address Chained=1, it becomes the physical address of the next descriptor.

The following tables detail the bit-field definitions of RDES0–RDES3.

15.3.6.1.1 RDES0 Fields

Bit	Field	Description
31	Own Bit	1 means DMA owns the descriptor; 0 means the host owns it. DMA clears this bit after receiving a frame or when the descriptor-associated buffer is full.
30	First Descriptor	Indicates that this descriptor contains the first buffer of a frame. If the buffer size is 0, the start data may be placed in buffer 2 or in a subsequent descriptor.
29	Last Descriptor	Indicates that the descriptor points to the last buffer of the frame.
28:14	Application Status	Application-specific status. Valid only in the Last Descriptor.
13:0	Frame / Packet Length	Length of the frame written back to host memory, in bytes. Valid only in the Last Descriptor.

15.3.6.1.2 RDES1 Fields

Bit	Field	Description
31	PTP Packet	Set to 0 when the host creates the descriptor. When IEEE 1588 PTP is enabled, GEMAC may set this bit to 1 to indicate that the received packet matches the PTP parameters.
30	PTP Timestamped	Set to 0 when the host creates the descriptor. If the PTP packet further matches the MessageID/control field, GEMAC sets this bit to 1 and latches the timestamp, SeqID, and SrcID into registers for software to read.
29:27		
26	End of the Ring	Indicates that the list has reached the last descriptor. DMA returns to the list base address to form a ring.
25	Second Address Chained	RDES3 no longer represents buffer 2. It represents the next descriptor address for linked-list mode.
24		
23:12	Buffer 2 Size	Length of buffer 2, from 0 to 4095 bytes. Invalid when Second Address Chained=1. If set to 0, DMA ignores this buffer and fetches the next descriptor.
11:0	Buffer 1 Size	Length of buffer 1, from 0 to 4095 bytes. If set to 0, DMA ignores buffer 1 and uses buffer 2.

15.3.6.1.3 RDES2 Fields

Bit	Field	Description
31:0	Buffer Address 1	Physical address of Receive Buffer 1. The GEMAC Core supports any byte alignment for the buffer start address. The descriptor itself must still be aligned according to 32-bit or 64-bit mode.

15.3.6.1.4 RDES3 Fields

Bit	Field	Description
31:0	Buffer Address 2 / Next Descriptor Address	When Second Address Chained=0, this field points to the physical address of Receive Buffer 2. When Second Address Chained=1, it points to the physical address of the next descriptor. The next descriptor address must be 32-bit aligned in 32-bit DMA mode and 64-bit aligned in 64-bit DMA mode.

15.3.6.2 Transmit Descriptor

The TX descriptor consists of four DWORDs (TDES0–TDES3), containing transmit status and control information, buffer length, and buffer or next-descriptor addresses, as summarized below.

DWORD	Main Contents	Software / Hardware Usage
TDES0	Transmit status, Own, Timestamp Packet, Transmit Frame/Packet Status	The host sets Own=1 to hand the descriptor to DMA. GEMAC clears Own after transmission completes or when the associated buffer is empty, and writes back status on the Last Segment.
TDES1	Control information, frame segmentation flags, CRC/padding control, ring/chain control, buffer length	The host initializes the transmit control fields. First Segment and Last Segment mark the frame boundaries in the descriptor chain.
TDES2	Buffer Address 1	The host writes the physical address of transmit buffer 1.
TDES3	Buffer Address 2 or Next Descriptor Address	In normal mode, this field is the physical address of buffer 2. When Second Address Chained=1, it becomes the physical address of the next descriptor.

The following tables detail the bit-field definitions of RDES0–RDES3.

15.3.6.2.1 TDES0 Fields

Bit	Field	Description
31	Own Bit	1 means DMA owns the descriptor; 0 means the host owns it. DMA clears this bit after frame transmission completes or when the descriptor-associated buffer is empty.
30	Timestamp Packet	Indicates that the transmitted packet requires a timestamp. Valid only in the First Segment and should be set only when IEEE 1588 PTP is enabled; otherwise it is ignored.
29:0	Transmit Frame / Packet Status	Application-specific frame status written back to the host after transmission completes. Valid only in the Last Segment.

15.3.6.2.2 TDES1 Fields

Bit	Field	Description
31	Interrupt On Completion	When set to 1, DMA asserts the transmit interrupt after the current frame is sent to the Application. Valid only in the Last Segment.
30	Last Segment	Indicates that this buffer contains the last segment of the frame.
29	First Segment	Indicates that this buffer contains the first segment of the frame.
28	Add CRC Disable	Valid only in the First Segment. When set to 1, GEMAC does not automatically calculate or insert FCS. This applies when the host buffer already contains FCS.Valid only in the First Segment. When set to 1, GEMAC does not automatically calculate or insert FCS. This applies when the host buffer already contains FCS.

27	Disable Padding	Valid only in the First Segment. When set to 1, no padding is added even if the frame is shorter than 64 bytes. When cleared, the frame is automatically padded to the minimum frame length.
26	End of the Ring	Indicates that the transmit list has reached the last descriptor. DMA returns to the base address.
25	Second Address Chained	TDES3 no longer represents buffer 2. It represents the next descriptor address for linked-list mode.
24	Force EOP Error	Valid only in the Last Segment. Forces DMA to generate an EOP with ERROR on the FIFO interface, used to discard or terminate a packet under construction.
23:12	Buffer 2 Size	Length of buffer 2, from 0 to 4095 bytes. Invalid when Second Address Chained=1. If set to 0, DMA ignores this buffer and fetches the next descriptor.
11:0	Buffer 1 Size	Length of buffer 1, from 0 to 4095 bytes. If set to 0, DMA ignores buffer 1 and uses buffer 2.

15.3.6.2.3 TDES2 Fields

Bit	Field	Description
31:0	Buffer Address 1	Physical address of Transmit Buffer 1. GEMAC supports any byte alignment for the transmit buffer address.

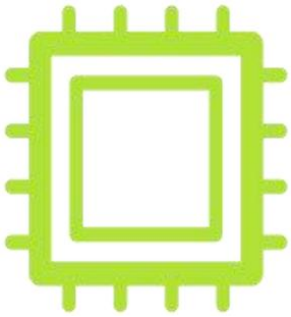
15.3.6.2.4 TDES3 Fields

Bit	Field	Description
31:0	Buffer Address 2 / Next Descriptor Address	When Second Address Chained=0, this field points to the physical address of Transmit Buffer 2. When Second Address Chained=1, it points to the physical address of the next descriptor. The next descriptor address must be 32-bit aligned in 32-bit DMA mode and 64-bit aligned in 64-bit DMA mode.

SPACEMIT

Chapter 16

Low-Speed Interface System



Key Stone® K1 User Manual

16.1 I2C Bus Interface

16.1.1 Introduction

The Inter-Integrated Circuit (I2C) bus is a true multi-master bus including collision detection and arbitration.

A dedicated I2C module, referred to as the power I2C module, is used to interface to the power management IC.

The I2C bus interface can function as both a master and a slave device on the I2C bus. This serial bus, developed by Philips Corporation, uses a 2-pin interface as follows:

- SDA: Data pin for input and output functions
- SCL: Clock pin for timing reference and control of the I2C bus

The I2C bus allows the I2C unit to interface with other I2C peripherals and microcontrollers. It requires minimal hardware, providing an economical solution for communicating status and control information between chips and external devices.

The I2C bus interface is a peripheral device residing on the peripheral bus that performs

- Data transfer, handled through a buffered interface for reliable communication
- Control and status management, accessed via memory-mapped registers

16.1.2 Features

- Compliance with I2C bus specification_ _with the exception of the support for the hardware general call, 10-bit slave addressing and CBUS compatibility
- Support for Multi-Master and Arbitration
- Operation modes and speeds as follows:
  - Standard Operation Mode: up to 100 Kbps
  - Fast Operation Mode: up to 400 Kbps
  - High-Speed Slave Operation Mode: up to 3.4 Mbps (High-Speed I2C only)
  - High-Speed Master Operation Mode: up to 3.3 Mbps (High-Speed I2C only)

  Note. In High-Speed Master Operation Mode, I2C operational frequencies decrease due to the pull-up resistors on the bus. The SCL frequency is inversely proportional to the pull-up resistor value (1/R).

16.1.3 Functional Description

I2C is a serial protocol used for communication between devices (agents) on the bus. It operates through a 2-pin interface as follows:

- **SDA (Serial Data and Address)**: Carries data and address information.
- **SCL (Serial Clock Line)**: Provides the clock signal to synchronize communication.

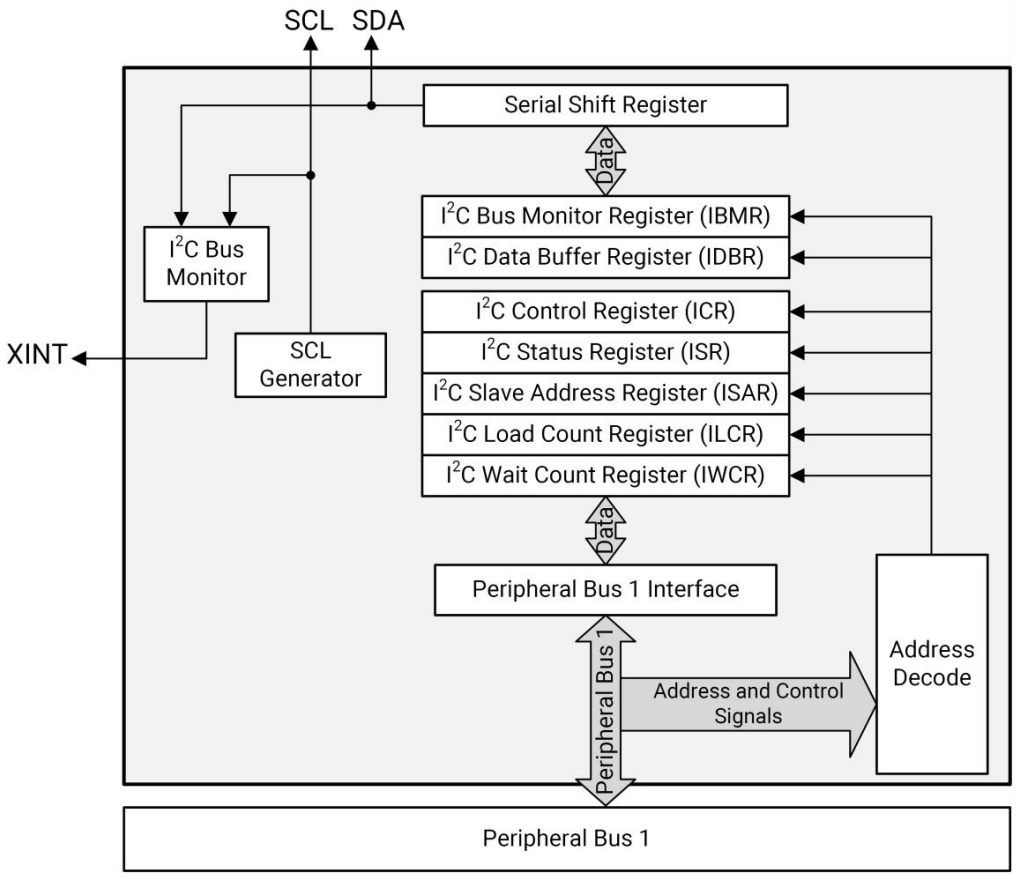
Each device on the I2C bus is identified by a unique 7-bit address and can function as either a transmitter or receiver in Master or Slave mode.

The key I2C terminology is tabled below.

I2C terminology	Definition
Transmitter	Sends data over the I2C bus
Receiver	Receives data over the I2C bus
Master	Initiates transfers, generates clock signals, and terminates transactions
Slave	Responds to the master's requests by transmitting or receiving data
Multi-master	Allows multiple masters to control the bus without corrupting messages
Arbitration	Ensures that only one master controls the bus when multiple masters compete. This technique avoids message corruption
Acknowledge	A response by the receiver to the master's acknowledge clock pulse on SCL The acknowledge can be either positive (ACK) or negative (NAK)
ACK	Positive Acknowledge: The receiver pulls SDA low during the clock pulse
NAK	Negative Acknowledge: The receiver keeps SDA high during the clock pulse

16.1.3.1 Block Diagram

The architecture of I2C bus interface is depicted below.



The I2C unit is a peripheral device that resides on the peripheral bus, and it interacts with the CPU through an interrupt mechanism or software polling:

- Interrupts notify the CPU about specific events on the I2C bus
- Software polling involves checking the I2C Status Register to track I2C activity without using interrupts

The I2C consists of:

- Two-wire interface to the I2C bus
- An 8-bit buffer for passing data
- A set of Control and Status registers
- A Shift register for parallel/serial conversions
- FIFO Mode:
  - TX FIFO: 8-entry buffer for outgoing data
  - RX FIFO: 16-entry buffer for incoming data
  - FIFO Pointers: Read/Write registers that can be cleared by software to flush the FIFOs after a critical interrupt

For the interrupt mechanism, the I2C unit can generate interrupts to notify the CPU of specific events, such as:

- Buffer full/empty.
- Stop condition detected (as a slave).
- I2C slave address detected.
- Arbitration lost.
- Bus error condition.

Note. All interrupt conditions must be explicitly cleared by software.

- About the Memory-Mapped Registers:
  - The I2C unit's Control, Status, and Data registers are located in the I2C memory-mapped address space
  - I2C Data Buffer Register (IDBR): 8-bit register used for transmitting and receiving data via the internal Shift register

16.1.3.2 I2C Master-Slave Operation

[Example]

The I2C can act as a master on the bus to communicate with an EEPROM as the slave:

- **Master Transmitter**: When the I2C sends data to the EEPROM, it operates as a master transmitter, and the EEPROM functions as a slave receiver.
- **Master Receiver:** When the I2C reads data from the EEPROM, it operates as a master receiver, and the EEPROM functions as a slave transmitter.

Regardless of the direction (transmitter or receiver), the master is responsible for:

- Generating the clock (SCL)
- Initiating the transaction
- Terminating the transaction

16.1.3.3 I2C Bus Structure & Clock Control

The I2C bus uses an open-drain wired-AND structure, allowing multiple devices to share and control the bus. This structure supports communication about key events, including:

- Arbitration
- Wait States
- Error Conditions

During data transfers, the master drives the SCL line:

- Data is transmitted when the clock is high
- If a slave cannot keep up with the master's clock rate, it can hold SCL low to insert wait intervals (clock stretching)

The SCL line can be altered only in two cases:

- Another master during arbitration
- A slow slave holding the clock low to delay communication

16.1.3.4 Multi-Master & Arbitration

The I2C bus supports multi-master operation, allowing more than one device to initiate data transfers simultaneously. When two or more masters try to control the bus, arbitration resolves the conflict:

- If two masters send the same data, both remain in control.
- A master loses arbitration if it attempts to drive SDA high while another master is driving it low.

The SCL line is a synchronized signal formed by combining the clocks generated by all active masters through the wired-AND connection.

16.1.3.5 I2C Transaction Types

I2C transactions can be initiated by the I2C as a master or received by the I2C as a slave. Both roles can involve:

- Read operations
- Write operations
- Combined Read/Write operations

16.1.3.6 I2C Bus Interface Mode

The I2C unit can accomplish a transfer in different operational modes as summarized below.

Mode	Description
Master-Transmit	- I2C acts as a master - Used to transmit operations - I2C sends the data - I2C generates the clock - Slave device is in Slave-Receive mode
Master-Receive	- I2C acts as a master - Used to receive operations - I2C receives the data - I2C generates the clock - Slave device is in Slave-Transmit mode
Slave-Transmit	- I2C acts as a slave - Responds to a master Read operation - I2C sends the data - Master device is in Master-Receive mode
Slave-Receive (default)	- I2C acts as a slave - Responds to a master Write operation - I2C receives the data - Master device is in Master-Transmit mode

16.1.3.6.1 Default Mode: Slave-Receive

- When the I2C unit is idle, it defaults to Slave-Receive mode. This allows the I2C interface to:
  - Monitor the bus for activity.
  - Receive any matching slave addresses.

16.1.3.6.2 High-Speed Mode Entry

- The I2C unit enters **high-speed mode** under the following conditions:
  - It detects a HS-mode master code (slave address 00001XX, where X is either 0 or 1).
  - Bit [9] of the I2C Control Register is set to 0x1 (enabling HS-mode).
    While in **HS-mode**, the I2C remains in **Slave-Receive mode** until further action.

16.1.3.6.3 Slave Address Detection/Matching

- When the I2C unit detects a matching 7-bit address in the I2C Slave Address Register (ISAR), it either:
  - Stays in Slave-Receive mode (for Write operations).
  - Switches to Slave-Transmit mode (for Read operations).
    The transition depends on the Read/Write (R/nW) bit:
  - R/nW is clear (0): Master intends to write → I2C remains in Slave-Receive mode.
  - R/nW is set (1): Master intends to read → I2C switches to Slave-Transmit mode.
  - Note: The R/nW bit is the least significant bit (LSb) in the byte containing the slave address.

16.1.3.6.4 Master Mode Transitions

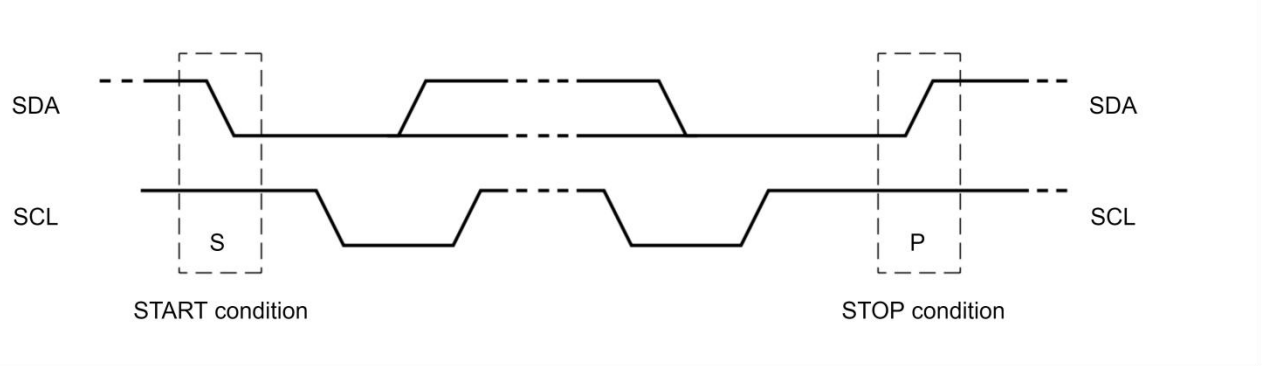
- When a peripheral initiates a transaction (Read or Write) on the I2C bus:
  - The I2C unit switches from the default Slave-Receive mode to Master-Transmit mode.
  - For Write Transactions:
    - The I2C unit remains in Master-Transmit mode after the address transfer is completed.
  - For Read Transactions:
    - The I2C unit transmits the slave address and then switches to Master-Receive mode.

16.1.3.7 Start & Stop Bus States

The I2C bus specification defines a Start transaction used at the beginning of a transfer and a Stop transaction used at the end of a transfer as follows:

- A Start condition occurs if a high-to-low transition occurs on the SDA line when SCL is high
- A Stop condition occurs if a low-to-high transition occurs on the SDA line when SCL is high

The relationship between the SDA and SCL lines for Start and Stop is depicted below.



The I2C unit uses the ICR[START] and ICR[STOP] bits to:

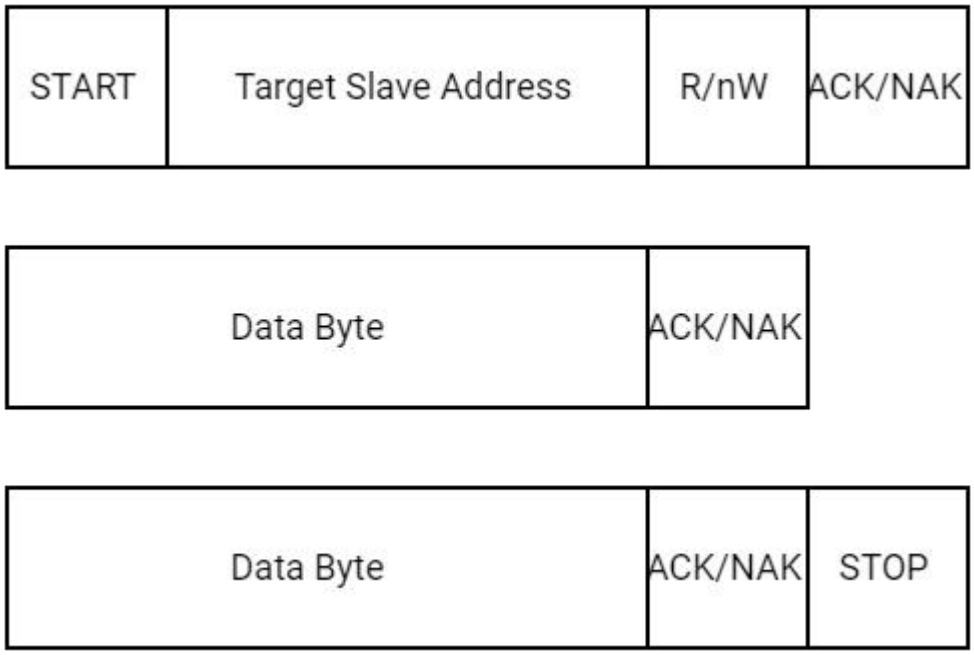
- Initiate an additional byte transfer
- Initiate a Start condition on the I2C bus
- Enable data chaining (repeated Start)
- Initiate a Stop condition on the I2C bus

The definitions of the START and STOP bits in the ICR are tabled below.

STOP Bit	START Bit	Condition	Notes
0	0	No Start or Stop	- Used for continuous data transfer without sending a Start or Stop condition.
0	1	Start condition and repeated Start	Starting condition: - The I2C sends a Start condition and transmits the IDBR 8-bit contents. - IDBR must contain the 7-bit slave address and the R/nW bit

STOP Bit	START Bit	Condition	Notes
			before a Start is initiated. For a repeated start: - The IDBR contains the target slave address and the R/nW bit. - Allows the master to perform multiple transfers to different slaves without releasing the bus. - The interface stays in Master-Transmit mode for Writes and switches to Master-Receive mode for Reads.
1	X	Stop condition	In Master-Transmit mode: - the I2C transmits the IDBR 8-bit contents and sends a Stop condition on the I2C bus. In Master-Receive mode: - ICR[ACKNAK] must be set to define a Negative-Acknowledge (NAK) pulse. - The I2C transmits the NAK pulse, places the received data byte into the IDBR, and sends a Stop condition on the I2C bus.

The Start and Stop Conditions of I2C are depicted below.



16.1.3.7.1 Start Condition

- **Control Bits:**
  - ICR[START] = 1
  - ICR[STOP] = 0

- **Purpose:**
  - Initiates a master transaction or a repeated Start.
- **Procedure:**
  - **Software Preparation**:
    - Load the target slave address and the R/nW bit into the IDBR.
  - **Transmission:**
    - Set ICR[TB] to transmit the Start condition and the IDBR contents on the I2C bus.
  - **Mode Transition:**
    - For Write requests: The I2C bus remains in Master-Transmit mode.
    - For Read requests: The I2C bus switches to Master-Receive mode.
  - **Repeated Start:**
    - Used to change the R/nW bit or the target slave address without releasing the bus.
    - The IDBR must contain the updated slave address and R/nW bit.
- **Arbitration Loss Handling**:
  - If the I2C loses arbitration while initiating a Start, it may re-attempt the Start when the bus is free.
- **Clearing:**
  - The Start condition is not cleared automatically by the I2C.

16.1.3.7.2 No Start or Stop Condition

- **Control Bits:**
  - ICR[START] = 0
  - ICR[STOP] = 0
- **Purpose:**
  - Used in Master-Transmit mode for continuous data transfer (multiple bytes).
- **Procedure:**
  - **Data Transfer:**
    - Software writes a data byte to the IDBR.
    - The I2C sets ISR[ITE] and clears ICR[TB].
  - **Next Byte:**
    - Software writes the next byte to the IDBR and sets ICR[TB] to initiate the next byte transmission.
  - **Continuation:**
    - This process repeats until software sets ICR[START] or ICR[STOP].
  - **Acknowledge Pulse:**
    - The I2C issues an ACK/NAK as defined by ICR[ACKNAK].
  - **Wait States:**
    - After each byte transfer (including the acknowledge pulse), the I2C holds the SCL line low to insert wait states until ICR[TB] is set.
    - This allows software to prepare the next byte for transmission.

- **Clearing:**
  - ICR[START] and ICR[STOP] are not cleared automatically after transmission.

16.1.3.7.3 Stop Condition

The Stop condition (ICR[START]=X, ICR[STOP]=1) terminates a data transfer. In Master-Transmit mode, ICR[STOP] and ICR[TB] must be set to initiate the last byte transfer. In Master-Receive mode, the I2C must set ICR[ACKNAK], ICR[STOP], and ICR[TB] to initiate the last transfer. Software must clear ICR[STOP] after the Stop condition is transmitted.

- **Control Bits:**
  - ICR[START] = X (don't care)
  - ICR[STOP] = 1
- **Purpose:**
  - Terminates a data transfer.
- **Procedure:**
  - **Master-Transmit Mode:**
    - Set ICR[STOP] and ICR[TB] to initiate the last byte transfer.
    - The I2C transmits the IDBR contents and sends a Stop condition.
  - **Master-Receive Mode:**
    - Set ICR[ACKNAK] to define a Negative-Acknowledge (NAK) pulse.
    - Set ICR[STOP] and ICR[TB] to initiate the last transfer.
- **Clearing:**
  - Software must clear ICR[STOP] after the Stop condition is transmitted.

16.1.3.8 Data Transfer Sequence

The I2C unit transfers data in 1-byte increments and always follows this sequence:

- Start
- 7-bit slave address
- R/nW bit
- Acknowledge pulse
- 8 bits of data
- Acknowledge pulse
- Repeat of steps 5 and 6 for the required number of bytes
- Repeated Start (repeat step 1) or Stop

16.1.3.9 Data & Addressing Management

The I2C Data Buffer register (IDBR) and the I2C Slave Address register (ISAR) are used for managing data transfer and slave addressing. Each register plays a specific role in handling I2C communication:

- IDBR: Stores 1 byte of data or a 7-bit slave address plus the Read/Not Write (R/nW) bit
- ISAR: Holds the programmable slave address for the I2C device

16.1.3.9.1 Data Handling Overview

- Receiving Data: The I2C controller places incoming data into the IDBR after receiving and acknowledging a full byte
- Transmitting Data: The CPU writes data to the IDBR, and the I2C controller sends it to the serial bus when ICR[TB] is set

16.1.3.9.2 Master or Slave Transmit Mode

- **Data Transmission:**
  - **Software Writes Data:**
    - Software writes data to the IDBR over the internal bus
    - This initiates a master transaction or sends the next data byte after ISR[ITE] is set
  - **I2C Transmits Data:**
    - The I2C transmits data from the IDBR when ICR[TB] is set
  - **Interrupts:**
    - If ICR[ITEIE] is set, an IDBR transmit-empty interrupt is signaled when a byte is transferred and the acknowledge cycle is complete
  - **Wait States:**
    - If the I2C is ready to transfer the next byte but the CPU has not written to the IDBR, the I2C inserts wait states until the CPU writes a new value to the IDBR and sets ICR[TB]

16.1.3.9.3 FIFO Mode

In FIFO mode, software writes control + data information to the TX FIFO instead of the IDBR.

16.1.3.9.4 Master or Slave Receive Mode

- **Data Reception:**
  - **Software Reads Data**: Software reads data from the IDBR over the internal bus after the IDBR receive-full interrupt is signaled (if ICR[DRFIE] is set)
  - **I2C Transfers Data:** The I2C transfers data from the Shift register to the IDBR after the acknowledge cycle completes
  - **Wait States:** The I2C inserts wait states until the IDBR is read by the CPU

- **Next Byte Transfer:** After the CPU reads the IDBR, the I2C unit updates the ICR[ACKNAK] and ICR[TB] bits, allowing the next byte transfer to proceed

- **FIFO Mode:**
  - In FIFO mode, software reads from the RX FIFO instead of the IDBR

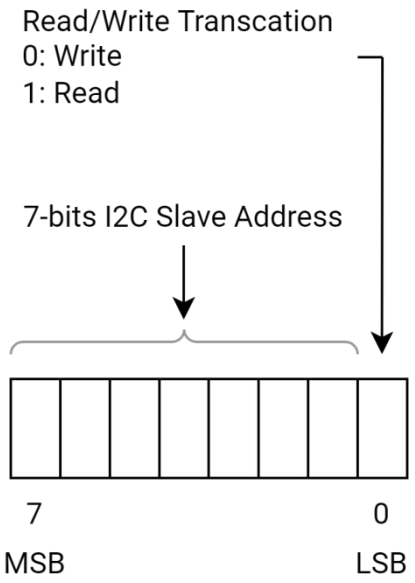
16.1.3.9.5 Addressing a Slave Device

Addressing a slave device is an important step in I2C communication.

A detailed explanation of how the **I2C master** addresses a slave device, different transaction types and the behavior of the I2C unit in different modes is provided in the following subsections.

16.1.3.9.5.1 Master Device Addressing a Slave

- **First Byte of Transaction (as depicted below)**:
  - The master composes and sends the **first byte** of the transaction
  - This byte consists of:
    - **7-bit slave address**: Identifies the target slave device
    - **R/nW bit**: Defines the transaction type (Read or Write)
  - The first byte can also be a **master code** indicating the start of a high-speed transaction
  - **Procedure**:
    - **Write to IDBR**: The master writes the **slave address** and **R/nW bit** to the **IDBR**
    - **Transmit First Byte**: The I2C transmits the first byte on the bus
    - **Acknowledge (ACK)**: The addressed slave responds with a **positive-acknowledge (ACK)** pulse



16.1.3.9.5.2 Transaction Types

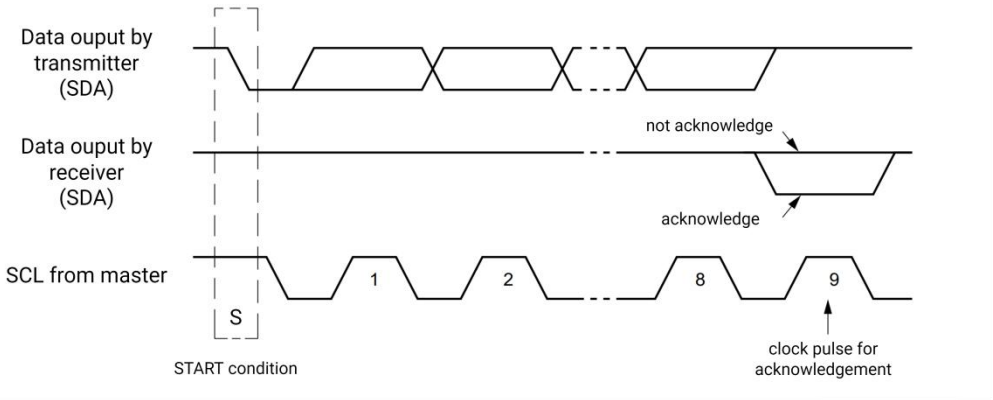
- **When the transaction is a Write (R/nW = 0)**
  - The I2C remains in **Master-Transmit mode**
  - The slave remains in **Slave-Receive mode**
  - The master sends data to the slave
- **When the transaction is a Read (R/nW = 1)**
  - The I2C switches to **Master-Receive mode** after the ACK
  - The slave switches to **Slave-Transmit mode**
  - The master reads data from the slave
- If the slave responds with a **NAK**, the I2C:
  - Aborts the transaction
  - Automatically sends a **Stop condition**
  - Sets the **ISR[BED]** (Bus Error Detected) bit

16.1.3.9.5.3 Slave Device Behavior

- **Idle State**:
  - When the I2C is enabled and idle, it remains in **Slave-Receive mode**
  - It monitors the I2C bus for a **Start condition**
- **Start Condition Detection**:
  - When a Start condition is detected, the I2C:
    - Reads the **first 7 bits** (slave address) and compares them to the value in the **I2C Slave Address Register (ISAR)**
    - Reads the **8th bit** (R/nW bit)
    - Transmits an **ACK pulse** if the address matches
- **Mode Transition**:
  - If the address matches:
    - **R/nW = 0**: The I2C remains in **Slave-Receive mode**
    - **R/nW = 1**: The I2C switches to **Slave-Transmit mode**

16.1.3.10 I2C Acknowledge

Every I2C byte transfer must be accompanied by an acknowledge (ACK) pulse that the receiver (master or slave) generates. The transmitter must release the SDA line for the receiver to transmit the acknowledge pulse. The acknowledge pulse on the I2C bus is depicted below.



16.1.3.10.1 In Master-Transmit Mode

- If the target slave receiver cannot generate the Acknowledge (ACK) pulse, the SDA line remains high, which indicates a Not-Acknowledge (NAK)
- The lack of ACK causes
  - The I2C sets ISR[BED]
  - The associated interrupt is generated (if enabled)
  - The I2C automatically generates a Stop condition and aborts the transaction

16.1.3.10.2 In Master-Receive Mode

- The I2C sends a negative-acknowledge (NAK) pulse to signal the slave transmitter to stop sending data
- The ICR[ACKNAK] bit controls the ACK/NAK pulse value driven onto the I2C bus

Procedure:

- The master receives a byte from the slave
- The I2C automatically transmits the ACK pulse after receiving each byte from the serial bus, unless it is the last bytes
- Before receiving the last byte, software must set ICR[ACKNAK] to generate a NAK
- The NAK pulse is sent after the last byte has been sent, to signals the end of data reception (and to stop sending data)
- **Note.** As required by the I2C bus protocol, ISR[BED] is not set for a Master-Receive mode NAK

16.1.3.10.3 In Slave mode

- The I2C automatically acknowledges its own slave address, regardless of the **ICR[ACKNAK]** setting
- In **Slave-Receive mode**,
  - After receiving a data byte, the slave sends an **ACK** automatically, regardless of the **ICR[ACKNAK]** setting
  - The I2C unit sends the **ACK** pulse after receiving the 8th data bit of the byte

- In **Slave-Transmit mode**,
  - If the master sends a **NAK**, it indicates the last byte is transferred
  - After sending a **NAK**, the master may issue either a Stop or repeated Start
  - The **ISR[UB]** remains set until a Stop or repeated Start is detected

16.1.3.11 Arbitration

I2C bus arbitration is required due to the multi-master capabilities of the I2C bus. Arbitration is used when 2 or more masters simultaneously generate a Start condition within the minimum I2C hold time of the Start condition.

16.1.3.11.1 Arbitration Process

- Arbitration can continue for an extended period if the **address field** and **R/nW bit** are the same.
- If the address, R/nW bit, or data differ, the master whose data does not match the **SDA line** (i.e., the master is driving a high state while SDA is low) loses arbitration.

16.1.3.11.2 Wired-AND Nature of I2C

- The I2C bus uses a **wired-AND** configuration, meaning:
  - If multiple masters output the same data, no data is lost.
  - If a master outputs a high state while the bus is low, it loses arbitration.

16.1.3.11.3 Behavior on Arbitration Loss

When a master loses arbitration:

- The I2C unit disables the SDA and SCL drivers for the remainder of the byte transfer.
- The arbitration loss detected bit ISR[ALD] is set.
- The I2C returns to idle (Slave-Receive) mode.

16.1.3.11.4 Handling Arbitration Loss in FIFO mode

Software must flush the FIFOs after arbitration loss. That can be done by clearing the Read and Write pointer registers for both the Transmit and Receive FIFOs.

- WFIFO_RPTR
- WFIFO_WPTR
- RFIFO_RPTR
- RFIFO_WPTR

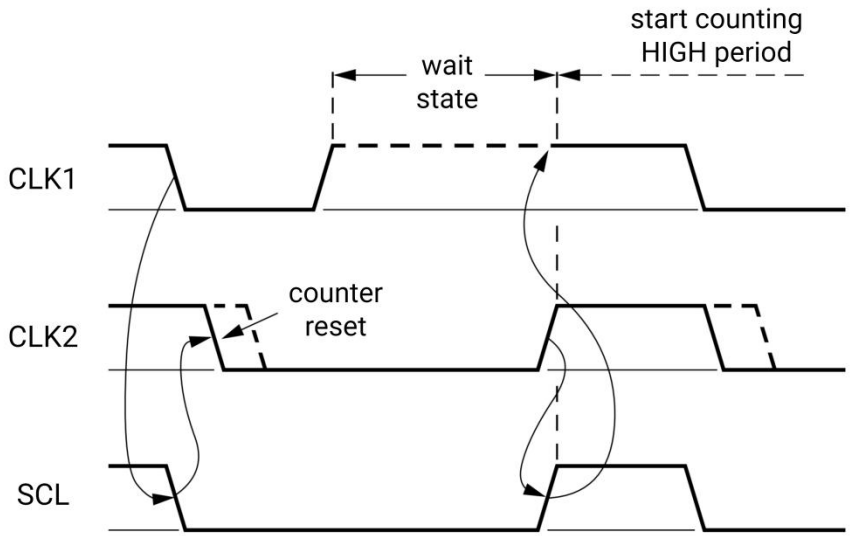
16.1.3.11.5 SCL (Serial Clock Line) Arbitration

Each master on the I2C bus generates its own clock on the SCL line for data transfers. This means that different masters may have different clock frequencies.

Since data is valid during the high period of the clock, a defined clock synchronization procedure is necessary to ensure proper communication. This is achieved through bit-by-bit arbitration.

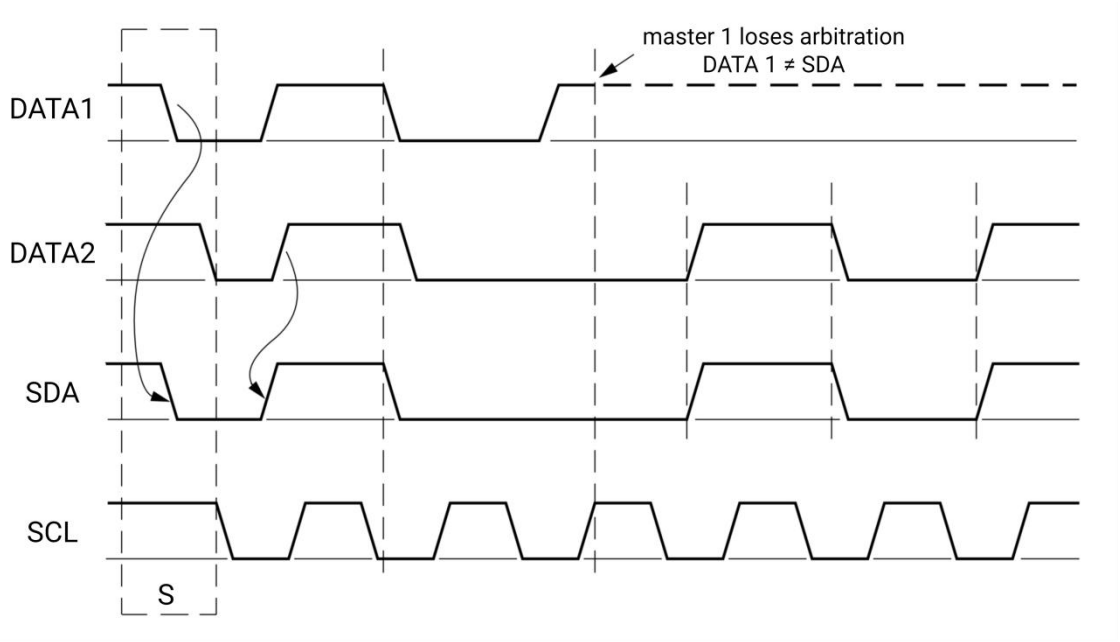
Clock Synchronization Mechanism (as depicted below)**:**

- Clock synchronization is achieved through the **wired-AND connection** of the I2C devices to the SCL line.
- **High to Low Transition**: When a master's clock transitions from **high to low**, it holds the SCL line for its own clock period.
- **Low to High Transition**: A clock cannot switch from **low to high** until all masters complete their low periods.
- The master with the **longest** low period holds the SCL line low. Masters with shorter low periods enter a **high wait-state** until the master with the longest low period completes.
- Once the master with the longest low period completes, the **SCL line** transitions to high, and other masters with shorter periods can continue their data cycles.
  The master with the longest clock period controls the SCL line, ensuring synchronized data transfer.



16.1.3.11.6 SDA Arbitration

Arbitration on the **SDA (Serial Data Line)** can extend over a significant duration, as it begins with the transmission of the **address** and **R/nW bits** and continues through the **data bits**. Below is depicted the arbitration procedure for two masters, although more than two masters may participate if connected to the bus.



16.1.3.11.6.1 Address & R/nW Checking

- **Condition 1:** If the **address** and **R/nW bit** transmitted by multiple masters are identical, arbitration proceeds to the **data bits**.
  - Due to the **wired-AND nature** of the I2C bus, no data is lost if multiple masters signal the same bus states.
- **Condition 2:** If the **address**, **R/nW bit**, or **data** differ, the master that transmits the first **high data bit** loses arbitration.
  - If the I2C loses arbitration:
    - Stop sending by shutting off its **SDA** and **SCL drivers** for the remainder of the byte transfer.
    - Sets the **ISR[ALD]** (Arbitration Loss Detected) bit.
    - Returns to **Slave-Receive mode**.

16.1.3.11.6.2 Arbitration Loss Case 1: Re-send

- If arbitration is lost during the transfer of **address bits** and the I2C unit is not addressed, it re-sends the address when the bus becomes free.
- This is possible because the **IDBR** (I2C Data Buffer Register) and **ICR** (I2C Control Register) are not overwritten during arbitration loss.

16.1.3.11.6.3 Arbitration Loss Case 2: Addressed as a Slave

If the I2C loses arbitration because another bus master addresses the I2C unit as a **slave device**, the I2C

- Switches to Slave-Receive mode
- Overwrites the original data in register IDBR.

Software must clear the Start and re-initiate the master transaction.

Note. Software must ensure that the I2C unit does not write to its own slave address, as this cause the I2C bus to enter an **indeterminate** state.

16.1.3.11.6.4 Boundary conditions

Boundary conditions exist for arbitration when an arbitration process is in progress and a repeated Start or Stop condition is transmitted on the I2C bus.

To prevent errors, the I2C unit, acting as a master, no arbitration occurs in these cases:

- Between a **repeated Start condition** and a **data bit**
- Between a **data bit** and a **Stop condition**
- Between a **repeated Start condition** and a **Stop condition**

These situations arise only when different masters write the **same data to the same target slave** simultaneously and arbitration is not resolved after the first data-byte transfer.

Note. The software must ensure that arbitration is resolved promptly. For example:

- The software can ensure that masters send unique data by requiring each master to transmit its **I2C address** as the first data byte of any transaction.
- When arbitration is resolved, the winning master sends a **restart** and begins a valid data transfer.
- The slave discards the master's address and processes the remaining data.

16.1.3.12 High Speed Mode

16.1.3.12.1 Introduction

The I2C unit supports the HS-mode of operation with

- The slave data transfer rates up to 3.4 Mbps
- The master data transfer rates up to 3.3 Mbps

HS-mode devices maintain backward compatibility with Fast and Standard mode (F/S-mode) devices.

When operating in HS-mode, the bus protocol and data format remain the same as in F/S mode, except for the following:

- **No Clock synchronization and arbitration** are performed in HS-mode. These processes are completed before HS-mode is entered.

HS-mode is entered when a master running in F/S-mode sends a master code and wins arbitration. At this point,

- The master switches to HS-mode and generates I2C transactions.

HS-mode ends when a **Stop condition** is generated by the master.

The master codes are a set of reserved slave addresses that are used to indicate the start of a HS-mode transfer. The master codes always win arbitration against other slave addresses. In the case of a multi-master, HS-mode system,

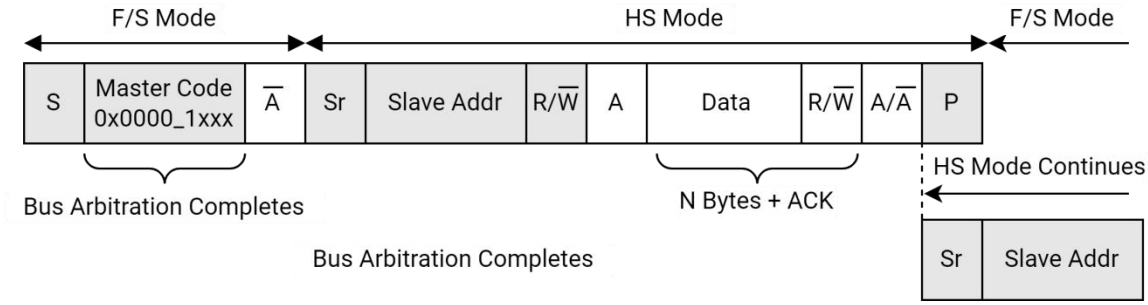
- Each master is assigned a unique master code. This ensures that the clock synchronization and bus arbitration finishes in F/S-mode.
- There are 8 possible master codes of the form: 8'b0000_1xxx (where x is either 0 or 1). Thus, a maximum of 8 masters are allowed in a multi-master HS-mode system.

16.1.3.12.2 Data Transfer in HS-mode

HS-mode is entered when a master running in F/S-mode sends and detects the following:

- send Start condition
- send Master Code (0x0000_1xxx)
- Detect not-acknowledge bit (A)

During this sequence, clock sync and arbitration have completed in Fast-mode and only one winning master remains. The master then switches to HS-mode and begins a bus transaction by issuing a repeated Start condition. Additional high speed data transfers can be linked by separating them with repeated start conditions. HS-mode ends when a Stop condition is sent. This sequence is depicted below.



To use the I2C unit in HS-mode as either a master or a slave, set the **ICR[MODE]** bits as follows:

- When MODE = 2'b10, all non-high speed transmits occur in Standard-mode.
- When MODE = 0b11, all non-high speed transmits occur in Fast-mode.

16.1.3.12.3 HS-mode Data Rate

According to I2C Bus Specification, the maximum data rate supported in HS-mode is dependent on the capacitive load of each bus line.

Capacitive Load:

- **<100 pF**: Max data rate speed (**3.4 Mbps** slave / **3.3 Mbps** master).
- **400 pF**: Data rate speed halves (**1.7 Mbps** slave / **1.65 Mbps** master).

Note. For capacitive loads between 100 pF and 400 pF, the maximum data rate is **linearly interpolated**. Capacitive loads must not exceed **400 pF**.

16.1.3.13 Master Operations

When software initiates a Read or Write on the I2C bus, the I2C unit switches from the default Slave-Receive mode to Master-Transmit mode. The 7-bit slave address and the R/nW bit follow the Start pulsen.

After the master receives an ACK, the I2C enters 1 of 2 Master modes:

- Master-transmit - I2C writes data
- Master-receive - I2C reads data

When transmitting the master code, the master should receive a NAK and then enter HS-mode. The 7-bit slave address and the R/nW bit follow a repeated Start condition. The master receives an ACK and the I2C unit enters 1 of 2 Master modes listed above.

The CPU writes to the ICR register to initiate a master transaction. Data is read and written from the I2C unit through the memory-mapped registers. The I2C unit responsibilities as a master device are tabled below.

I2C Master Action	Mode of Operation	Definition
Generate clock output	Master-transmit Master-receive	- The master drives the SCL line. - ICR[SCLE] and ICR[IUE] must be set.
Write target slave address to IDBR	Master-transmit Master-receive	- The CPU writes to IDBR bits [7:1] before enabling a Start condition. - The first 7 bits are sent on the I2C bus after the Start. See **Section 16.1.3.7**, Start and Stop Bus States.
Write R/nW bit to IDBR	Master-transmit Master-receive	- CPU writes to least significant IDBR bit with R/nW control bit - If the R/nW bit is low, the master remains a Master-Transmitter. If high, the master switches to a master receiver. See **Section 16.1.3.9**, Data and Addressing Management.
Signal Start condition	Master-transmit Master-receive	See Generate clock output action in this table. After the target slave address and R/nW bit are in the IDBR, - Software sets ICR[START]. - Software sets ICR[TB] to initiate the Start condition. See **Section 16.1.3.7**, Start and Stop Bus States.
Initiate first data byte transfer	Master-transmit Master-receive	- The CPU writes a data byte to the IDBR - The I2C transmits the byte when ICR[TB] is set. - The I2C clears ICR[TB] and sets ISR[ITE] when the transfer is complete.
Arbitrate for I2C bus	Master-transmit Master-receive	If 2 or more masters signal a Start within the same clock period, arbitration must occur. - The I2C arbitrates for as long as needed. Arbitration takes place during slave address and R/nW bit or data transmission and continues until all but one master loses the bus. No data is lost. - If the I2C loses arbitration, it sets ISR[ALD] after the byte transfer is completed and switches to Slave-Receive mode. - If the I2C loses arbitration as it attempts to send the target address byte, it attempts to resend the byte when the bus becomes free. Software must ensure that the boundary conditions described in **Section 16.1.3.11.6.4**, Operation do not occur.
Write one data byte to the IDBR	Master-transmit only	- Occurs when ISR[ITE] is set and ICR[TB] is clear. If the IDBR transmit-empty interrupt is enabled, the interrupt is generated.

I2C Master Action	Mode of Operation	Definition
		- The CPU writes 1 data byte to the IDBR, sets the appropriate START/STOP bit combination, and sets ICR[TB] to send the data. Eight bits are taken from the Shift register and written to the serial bus. The eight bits are followed by a Stop, if requested by ICR[STOP] being set.
Wait for Acknowledge from slave receiver	Master-transmit only	As a master transmitter, the JINDIE I2C generates the clock for the acknowledge pulse. The JINDIE I2C releases the SDA line to allow Slave-Receiver acknowledge transmission. See **Section 16.1.3.10,** I2C Acknowledge.
Read one byte of I2C data from the IDBR	Master-receive only	Eight bits are read from the serial bus, collected in the Shift register, then transferred to the IDBR after the ICR[ACKNAK] bit is read. - The CPU reads the IDBR when ISR[IRF] is set and ICR[TB] is clear. If the IDBR receive-full interrupt is enabled, it is signalled to the CPU. - When the IDBR is read, if ISR[ACKNAK] is clear (indicating ACK), the software must clear the ICR[ACKNAK] bit and set ICR[TB] to initiate the next byte Read. - If ISR[ACKNAK] is set (indicating NAK), ICR[TB] is clear, ICR[STOP] is set, and ISR[UB] is set, then the last data byte has been read into the IDBR, and the I2C is sending the Stop. - If ISR[ACKNAK] is set (indicating NAK) and ICR[TB] is clear, but ICR[STOP] is clear, then the software has 2 options: 1. Set ICR[START], write a new target address to the IDBR, and set ICR[TB], which sends a repeated Start. 2. Set ICR[MA] and leave ICR[TB] clear, which sends a Stop only.
Transmit acknowledge to slave transmitter	Master-receive only	- As a master receiver, the JINDIE I2C generates the clock for the acknowledge pulse and drives the SDA line during the acknowledge cycle. - If the next data byte is to be the last transaction, the user software sets ICR[ACKNAK] for NAK generation. See **Section 16.1.3.10,** I2C Acknowledge.
Generate a repeated Start to chain I2C transactions	Master-transmit Master-receive	Data chaining takes place by using a repeated Start condition instead of a Stop condition. - The repeated Start is generated after the last data byte of a transaction has been transmitted on the I2C bus, as described in **Section 16.1.3.8** Data Transfer Sequence. - The software must write the next target slave address and the R/nW bit to the IDBR, sets ICR[START], and sets ICR[TB]. See **Section 16.1.3.7,** Start and Stop Bus States.
Generate a Stop	Master-transmit Master-receive	- A Stop is generated after the last data byte of a transaction has been transmitted on the I2C bus, as described in **Section 16.1.3.8**, Data Transfer Sequence. - ICR[STOP] must be set in order to generate the Stop condition. See **Section 16.1.3.7,** Start and Stop Bus States.

16.1.3.14 FIFO mode

FIFO mode is an extension to the I2C module. The main features of FIFO mode are:

- FIFOs are added on both transmit and receive sides.

This helps reducing the number of IDBR empty/full interrupts. Instead of the core writing or reading to/from the IDBR 1 byte at a time, the FIFOs allow reading and writing multiple bytes without interrupting the core after each Byte.

- DMA mode is added.

DMA mode allows improvement in long I2C transactions (typically more than 8 bytes) where complete transaction can be programmed in DMA and allows reducing number of FIFO interrupts.

Note. FIFO mode is completely backward compatible. It is allowed to disable FIFO mode and work in I2C legacy mode by writing ICR[FIFO_EN] = 0.

16.1.3.14.1 Transmit FIFO (Tx FIFO)

- **Structure:**
  - Width: 12 bits (4-bit control word + 8-bit data)
  - Number of entries: 8
  - Each entry consists of a data byte concatenated with a 4-bit control word

- **Control Word:**
  - The control word corresponds to the **ICR[3:0]** bits, which required for each transmitted data byte.
- **Data Transmission Process:**
  - After a byte is transferred, the next byte is copied from the Tx FIFO into the shift register.
  - Simultaneously, the associated control word is copied into the **ICR** register.
  - This process continues until the **STOP** bit is set, signaling the end of transmission

.

16.1.3.14.2 Receive FIFO (Rx FIFO)

- **Structure:**
  - Width: 8 bits (only data)
  - Number of entries: 16
  - Each entry stores one received byte
- **Data Handling Process:**
  - When the Rx FIFO is **half full**, it triggers either an **interrupt** or a **DMA request**.
  - At this point, the stored data must be read from the FIFO.
  - Any additional incoming data is stored in the remaining free entries until the FIFO is full.

16.1.3.14.3 FIFO Mode Support

In order to support the FIFO mode and fully utilize its capabilities, the following status and control bits were added:

- **ICR[FIFO_EN]:** enables FIFO mode.
- **ICR[TXBEGIN]:** starts a transaction.
- New status bits in ISR have been added for FIFO mode interrupts and also a bit to flag Transaction done. ICR has the enable bits for all these interrupts.
- TXDONE interrupt generated at the end of each transaction (when STOP bit is send).
- **ICR[DMA_EN]:** Enables or disables **DMA mode** and switches between **DMA** and **PIO mode**

In **DMA mode**, all the FIFO related interrupts have to be disabled in ICR and ICR[DMA_EN] bit has to be set. And only DMA requests are sent to the DMA and not any interrupts to the core.

Similarly in **PIO mode**, interrupts have to enabled and ICR[DMA_EN] bit cleared. So only interrupts to the core are generated when the Transmit FIFO is full or when the Receive FIFO is half full, full or overrun. ICR[TXDONE_IE] is for enabling Transaction Done Interrupt and needs to be set in both PIO and DMA modes. The core needs to get an interrupt after each transaction is done.

Note: This FIFO mode should only be used when the I2C is in Master mode. This FIFO mode is not for Slave mode.

16.1.3.15 I2C Serial Clock Programming Guidelines

Before each of the I2C modules has been initialized, set the clock:

1. Open and select the I2C clock by setting the Clock/Reset Control Register for I2C.
2. Set the I2C Load Count Register (ILCR) to the desired frequency.

16.1.3.16 Master Mode Programming Examples

16.1.3.16.1 Initialize Unit

1. Set the slave address in the ISAR
2. Enable the preferred interrupts in the ICR. Do not enable the arbitration-loss-detected interrupt
3. Set ICR[MODE] to the desired bus rate. For HS-mode bus rate, the master-code must be transmitted before HS-mode bus timing is enabled. Software should always set the HS mode bit (ICR[16]) as that bit makes the I2C HS capable. This bit does not affect the non-HS transfers
4. Set the ICR[IUE] and ICR[SCLE] bits to enable the I2C and SCL

16.1.3.16.2 Write 1 Byte as a Master

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 0 for a Write
2. Initiate the Write
3. Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB]
4. When an IDBR transmit-empty interrupt occurs

5. Read ISR register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 0
6. Write a 1 to the ISR[ITE] bit to clear interrupt
7. Write a 1 to the ISR[ALD] bit if set
8. If the master loses arbitration, it performs an address retry when the bus becomes free. The arbitration-loss-detected interrupt is disabled to allow the address retry.
9. Load data byte to be transferred in the IDBR
10. Initiate the Write
11. Clear ICR[START], set ICR[STOP], set ICR[ALDIE], set ICR[TB]
12. When an IDBR transmit-empty interrupt occurs (unit is sending Stop). Read ISR register: ISR[ITE] = 1, ISR[UB] = x, ISR[RWM] = 0
13. Write a 1 to the ISR[ITE] bit to clear the interrupt
14. Clear ICR[STOP] bit

16.1.3.16.3 Read 1 Byte as a Master

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 1 for a Read
2. Initiate the Write
3. Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB]
4. When an IDBR transmit-empty interrupt occurs
5. Read ISR register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 1
6. Write a 1 to the ISR[ITE] bit to clear the interrupt
7. Initiate the Read
8. Clear ICR[START], set ICR[STOP], set ICR[ALDIE], set ICR[ACKNAK], set ICR[TB]
9. When an IDBR receive-full interrupt occurs (unit is sending Stop)
10. Read ISR register: IDBR receive full (1), ISR[UB] = x, ISR[RWM] = 1, ACK/NAK bit (1)
11. Write a 1 to the ISR[IRF] bit to clear the interrupt
12. Read IDBR data
13. Clear ICR[STOP] and ICR[ACKNAK] bits

16.1.3.16.4 Write 2 Bytes and Repeated Start Read 1 Byte as a Master

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 0 for a Write
2. Initiate the Write
3. Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB]
4. When an IDBR transmit-empty interrupt occurs
5. Read ISR register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 0
6. Write a 1 to the ISR[ITE] bit to clear interrupt
7. Load data byte to be transferred in the IDBR
8. Initiate the Write
9. Clear ICR[START], clear ICR[STOP], set ICR[ALDIE], set ICR[TB]

10. When an IDBR transmit-empty interrupt occurs
11. Read ISR register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 0
12. Write a 1 to the ISR[ITE] bit to clear interrupt
13. Repeat step 5 to 8 one time
14. Load target slave address and R/nW bit in the IDBR. R/nW must be 1 for a Read
15. Send repeated Start as a master
16. Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB]
17. When an IDBR transmit-empty interrupt occurs
18. Read ISR register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 1
19. Write a 1 to the ISR[ITE] bit to clear interrupt
20. Initiate the Read
21. Clear ICR[START], set ICR[STOP], set ICR[ALDIE], set ICR[ACKNAK], set ICR[TB]
22. When an IDBR receive-full interrupt occurs (unit is sending Stop
23. Read ISR register: ISR[IRF] = 1, ISR[UB] = x, ISR[RWM] = 1, ISR[ACKNAK] = 1
24. Write a 1 to the ISR[IRF] bit to clear the interrupt
25. Read IDBR data
26. Clear ICR[STOP] and ICR[ACKNAK] bits

16.1.3.16.5 Read 2 Bytes as a Master - Send Stop Using the Abort

1. Load target slave address and R/nW bit in the IDBR. R/nW must be 1 for a Read
2. Initiate the Write
3. Set ICR[START], clear ICR[STOP], clear ICR[ALDIE], set ICR[TB]
4. When an IDBR transmit-empty interrupt occurs
5. Read ISR register: ISR[ITE] = 1, ISR[UB] = 1, ISR[RWM] = 1
6. Write a 1 to the ISR[ITE] bit to clear interrupt
7. Initiate the Read
8. Clear ICR[START], clear ICR[STOP], set ICR[ALDIE], clear ICR[ACKNAK], set ICR[TB]
9. When an IDBR receive-full interrupt occurs
10. Read ISR register: ISR[IRF] = 1, ISR[UB] = 1, ISR[RWM] = 1, ACK/NAK bit (0)
11. Write a 1 to the ISR[IRF] bit to clear the interrupt
12. Read IDBR data
13. Clear ICR[STOP] and ICR[ACKNAK] bits
14. Initiate the Read
15. Clear ICR[START], clear ICR[STOP], set ICR[ALDIE], set ICR[ACKNAK], set ICR[TB]
16. ICR[STOP] is not set because Stop or repeated Start is determined on the byte Read
17. When an IDBR receive-full interrupt occurs
18. Read ISR register: ISR[IRF] = 1, ISR[UB] = 1, ISR[RWM] = 1, ISR[ACKNAK] = 1
19. Write a 1 to the ISR[IRF] bit to clear the interrupt
20. Read IDBR data
21. Initiate Stop abort condition (Stop with no data transfer). Set ICR[MA]

16.1.3.16.6 High-speed Mode: Write 1 Byte as a Master

1. Load master code in the IDBR
2. Enable interrupts. Set ICR[ITEIE]
3. Enable high speed mode
4. Set ICR[MODE] = 0b10 or 0b11
5. Initiate the Write
6. Set ICR[MODE], set ICR[START], clear ICR[STOP], set ICR[TB]
7. When an IDBR transmit-empty interrupt occurs
8. Read ISR: IDBR transmit empty (1), unit busy (1), ACKNAK (1)
9. Write a 1 to the ISR[ITE] bit to clear interrupt
10. Send a repeated start and slave address
11. Load the slave address and the R/W bit in the IDBR
12. Initiate the Write
13. Set ICR[START], clear ICR[STOP], set ICR[TB]
14. Wait for the IDBR transmit empty interrupt
15. Write a 1 to the ISR[ITE] bit to clear interrupt
16. Load data byte to be transferred in the IDBR
17. Initiate the Write
18. Clear ICR[START], set ICR[STOP], set ICR[TB]
19. When an IDBR transmit-empty interrupt occurs (unit is sending Stop). Read ISR: IDBR transmit empty (1), unit busy (x), R/nW bit (0)
20. Write a 1 to the ISR[ITE] bit to clear the interrupt

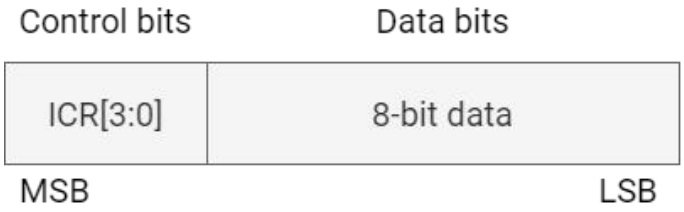
16.1.3.16.7 High-speed Mode: Read 1 Byte as a Master

1. Load master code in the IDBR
2. Enable interrupts. Set ICR[ITEIE]
3. Initiate the Write
4. Set ICR[MODE], set ICR[START], clear ICR[STOP], set ICR[TB]
5. When an IDBR transmit-empty interrupt occurs
6. Read ISR: IDBR transmit empty (1), unit busy (1), ACKNAK (1)
7. Write a 1 to the ISR[ITE] bit to clear interrupt
8. Send a repeated start and slave address
9. Load the slave address and the R/W bit (1) in the IDBR
10. Initiate the Write
11. Set ICR[START], clear ICR[STOP], set ICR[TB]
12. Wait for the IDBR transmit empty interrupt
13. Write a 1 to the ISR[ITE] bit to clear interrupt
14. Initiate the Read
15. Clear ICR[START], set ICR[STOP], set ICR[ACKNAK], set ICR[TB]

16. When an IDBR receive-full interrupt occurs (unit is sending Stop)
17. Read ISR: IDBR receive full (1), unit busy (x), R/nW bit (1), ACKNAK (1)
18. Write a 1 to the ISR[IRF] bit to clear the interrupt
19. Read IDBR data
20. Clear ICR[STOP] and ICR[ACKNAK] bits

16.1.3.16.8 FIFO mode: Write/Read n Bytes as a Master in PIO mode

1. Program I2C slave address
2. Write ICR[FIFOEN] to enable FIFO mode and enable the FIFO interrupts. An interrupt is generated here if the TX FIFO Empty interrupt is enabled.
3. Fill up the TX FIFO with control + data in proper format as depicted below
4. Write a 1 to the ISR[TXE] bit to clear the TX FIFO empty interrupt
5. While the data transfer is happening, wait for the TX FIFO empty interrupt
6. When TX FIFO empty interrupt is seen, refill the FIFO with more control+data and then write 1 to the ISR[TXE] bit to clear the TX FIFO empty interrupt.
7. Wait for TX DONE or TX empty again (depending on the transaction)
8. If TX DONE interrupt is received, program ICR for the next transaction and once the FIFO has the control+data, set the ICR[TXBEGIN] bit. This bit starts the next transaction.
9. If the RX FIFO becomes half full during this sequence, RX FIFO Half Full Interrupt is generated (If enabled)
10. Read the RX FIFO data and then write 1 to the ISR[RXF] bit to clear the RX FIFO Half Full interrupt



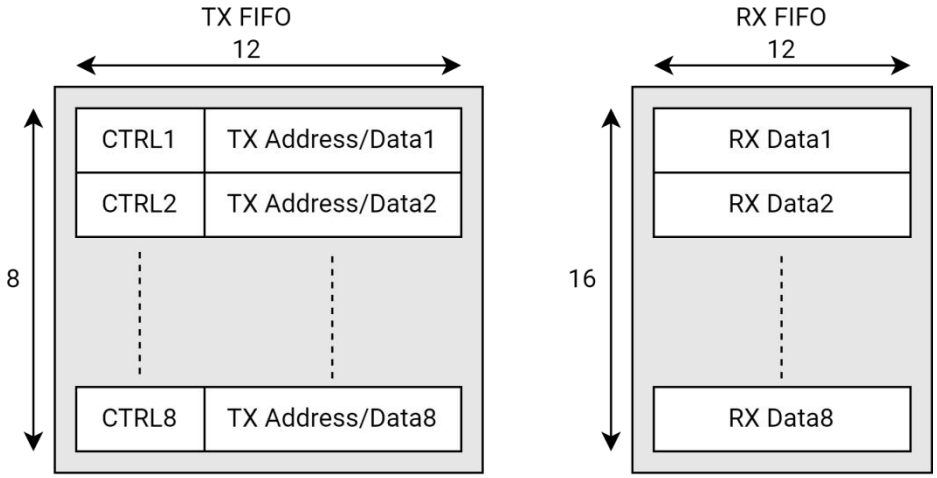
16.1.3.16.9 FIFO mode: Write/Read n Bytes as a Master in DMA mode

1. Program DMA descriptors and put control + data in memory in the proper format as depicted above
2. Program I2C slave address
3. Write 1 to the ICR[FIFOEN] bit to enable FIFO mode and the ICR[DMAEN] bit to enable DMA mode (disable TX FIFO empty & RX full interrupts, but enable Transmit Done interrupt)
4. TX FIFO empty DMA request is generated immediately since the TX FIFO is empty when I2C comes out of reset
5. DMA services the request and fills up the TX FIFO with control+data
6. The I2C starts the transaction on the bus and when the TX FIFO becomes empty, a TX FIFO empty DMA request is generated
7. Repeat #5 and #6 until a TX DONE interrupt is received

8. When TX DONE is received (this means a STOP occurred), CPU must write to ISR and ICR registers to reconfigure them for the next transaction and also set the ICR[TXBEGIN] bit for the next transaction to start.
9. Transaction continues and when TX FIFO is empty, the TX FIFO empty DMA request is generated once again
10. If the RX FIFO ever becomes half full during this sequence it sends a RX FIFO Half Full DMA request to the DMA
11. DMA reads the RX FIFO contents
12. After receiving a TXDONE interrupt, if there are any trailing bytes (Only for Read Transactions) it is up to the software to handle them. The hardware does not handle trailing bytes.
13. Each entry in the Tx FIFO has the format as shown in Figure-9 - . The LSB 8 bits are for data and the MSB 4 bits are for control bits. For a Write transaction, data consists of the Slave address followed by actual Write data.The control bits are nothing but ICR[3:0] bits.
14. For a Read transaction, Data consists of the Slave address followed by dummy data (actual Read data from the slave goes into the Rx FIFO). Again, the control bits are ICR[3:0] bits.

16.1.3.16.10 FIFO Programming Examples

Transmit and Receive FIFOs are depicted below.



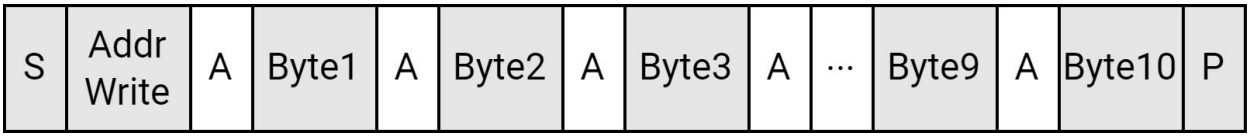
Each control word (CTRL) is 4 bit length:

- [TB]
- [ACKNACK]
- [STOP]
- [START]

Each entry in the Tx FIFO has a control word concatenated with Address/Data byte. The control word has control information to send/receive that particular byte. Note that Transmit FIFO has 8 entries and RX FIFO has 16 entries. Interrupt/DMA requests are made when the Tx FIFO is empty or when Rx FIFO is half full.

16.1.3.16.10.1 Programming model for Case 1

1. Core initially programs the ICR for FIFO mode. It then receives a Tx FIFO empty interrupt if in PIO mode or a DMA request if in DMA mode.
2. The core/DMA then writes 1 address byte + 7 data bytes (Byte1-Byte7) to the Tx FIFO (with each byte having a corresponding control word).
3. I2C starts sending out the bytes and when the Tx FIFO is empty, it generates a Tx FIFO empty interrupt/DMA request.
4. The core/DMA then writes the last 3 bytes (Byte8-Byte10) to the Tx FIFO.
5. I2C sends out the last 3 bytes, and when it sees that STOP has been sent out, it sets the ISR[TX_DONE] bit and generates an interrupt.
6. The core then cleans up the control and status registers (example: Clear ICR[STOP] bit, clear ISR[TX_DONE] bit) and starts the next transaction (set ICR[TX_BEGIN] bit).



16.1.3.16.10.2 Programming Model for Case 2

1. Core initially programs the ICR for FIFO mode. It then receives a Tx FIFO empty interrupt if in PIO mode or a DMA request if in DMA mode.
2. The core/DMA then writes 1 Addr + 7 Data Bytes to the TX FIFO (control word + dummy data since it is a Read transaction).
3. After the address is sent out on the bus, for each control word, a Read byte is received and saved off in the Rx FIFO. Once the Tx FIFO is empty (mote that there are 7 bytes in the Rx FIFO by now), an interrupt/DMA request is made and the remaining bytes (1 data byte from Read transaction and 3 bytes from the next Write transaction) are loaded into the Tx FIFO.
4. After the 8 byte is received into the Rx FIFO, the Rx FIFO Half full interrupt/DMA request is set. This Read data now needs to be read out of the FIFO.
5. By now, the Read transaction is also done. But since there is NO stop bit after the Read and instead Repeated Start is used for the Write Transaction, the ISR[TX_DONE] status bit is NOT set as it would have normally been set at the end of a transaction.
6. I2C now starts the Write transaction by sending out the Address followed by the 2 Write bytes.
7. Once the Write transaction is done, ICR[TX_BEGIN] is automatically cleared and ISR[TX_DONE] bit is set, which generates an interrupt to the core.



16.1.3.17 Slave Operations

How I2C unit operates as a slave device is tabled below.

I2C Slave Action	Mode of Operation	Definition
Slave-receive (default mode)	Slave-receive only	- The I2C monitors all slave address transactions. - ICR[IUE] must be set. - The I2C monitors bus for Start conditions. When a Start is detected, the interface reads the first 8 bits and compares the most significant 7 bits with the 7-bit ISAR. If there is a match, the I2C sends an ACK. - If the eighth bit of the first byte (R/nW bit) is low, the I2C stays in Slave-Receive mode, and ISR[SAD] is cleared. If R/nW bit is high, the I2C unit switches to Slave-Transmit mode, and ISR[SAD] is set.
Set the slave address - detected bit	Slave-receive Slave-transmit	- Indicates the interface has detected an I2C operation that addresses current I2C. - An interrupt is generated, if enabled, after the matching slave address is received and acknowledged.
Read one byte of I2C data from the IDBR	Slave-receive only	- This operation occurs when ISR[IRF] is set and ICR[TB] is clear. If enabled, the IDBR receive-full interrupt is generated. - Eight bits are read from the serial bus into the shift register. When a full byte has been received and the ACK/NAK bit is completed, the byte is transferred from the Shift register to the IDBR. - Occurs when the IDBR receive full bit in the ISR is set and the transfer byte bit is clear. If enabled, the IDBR receive-full interrupt is signalled to the CPU. - Software reads one data byte from the IDBR. When the IDBR is read, the software must write the preferred ICR[ACKNAK] bit and sets ICR[TB]. This causes the I2C to stop inserting wait states and let the master transmitter transmit the next chunk of information.
Transmit Acknowledge to master transmitter	Slave-receive only	- As a slave receiver, the I2C pulls the SDA line low to generate the ACK pulse during the high SCL period. - ICR[ACKNAK] controls the acknowledge pulse that the I2C drives. See **Section 16.1.3.10,** I2C Acknowledgee.
Write one byte of I2C data to the IDBR	Slave-transmit only	- This operation occurs when ISR[ITE] is set and ICR[TB] is clear. If enabled, the IDBR transmit-empty interrupt is generated. - The software must write a data byte to IDBR and sets ICR[TB] to start the transfer.
Wait for Acknowledge from master receiver	Slave-transmit only	- As a slave transmitter, the I2C releases the SDA line to allow the master receiver to pull the line low for the ACK. See **Section 16.1.3.10,** I2C Acknowledge.

16.1.3.18 Slave Mode Programming Examples

16.1.3.18.1 Initialize Unit

1. Set the slave address in the ISAR

2. Enable preferred interrupts in the ICR
3. If the I2C unit is a HS-mode slave, set ICR[MODE] = 0b10 or 0b11. Software should always set the HS mode bit (ICR[16]) as that bit makes the I2C HS capable. This bit does not affect the non-HS transfers.
4. Set the ICR[IUE] bit to enable the I2C

16.1.3.18.2 Transmit n Bytes as a Slave

1. When a slave-address-detected interrupt occurs. Read ISR register: ISR[SAD] = 1, ISR[UB] = 1, ISR[RWM] = 1, ISR[ACKNAK] = 0
2. Write a 1 to the ISR[SAD] bit to clear the interrupt
3. Return from interrupt
4. Load data byte to transfer in the IDBR
5. Set ICR[TB] bit
6. When an IDBR transmit-empty interrupt occurs. Read ISR register: ISR[ITE] = 1, ISR[ACKNAK] = 0, ISR[RWM] = 0
7. Load data byte to transfer in the IDBR
8. Set the ICR[TB] bit
9. Write a 1 to the ISR[ITE] bit to clear interrupt
10. Return from interrupt
11. Repeat steps 6 to 10 for **n**-1 times. If, at any time, the slave does not have data, the I2C keeps SCL low until data is available
12. When a IDBR transmit-empty interrupt occurs
13. Read ISR register: ISR[ITE] = 1, ISR[ACKNAK] = 1, ISR[RWM] = 0
14. Write a 1 to the ISR[ITE] bit to clear interrupt
15. Return from interrupt
16. When a slave-Stop-detected interrupt occurs. Read ISR register: ISR[UB] = 0, ISR[SSD] = 1
17. Write a 1 to the ISR[SSD] bit to clear interrupt

16.1.3.18.3 Receive n Bytes as a Slave

1. When a slave-address-detected interrupt occurs. Read ISR register: ISR[SAD] = 1, ISR[UB] = 1, ISR[RWM] = 0
2. Write a 1 to the ISR[SAD] bit to clear the interrupt
3. Return from interrupt
4. Set ICR[TB] bit to initiate the transfer
5. When an IDBR receive-full interrupt occurs. Read ISR register: ISR[IRF] = 1, ISR[ACKNAK] = 0, ISR[RWM] = 0
6. Read IDBR to get the received byte
7. Write a 1 to the ISR[IRF] bit to clear interrupt
8. Return from interrupt
9. Repeat steps 5 to 8 for **n**-1 times. Once the IDBR is full, the I2C keeps SCL low until the data is read.

10. Set ICR[TB] bit to release I2C bus and allow next transfer
11. When a slave-stop-detected interrupt occurs. Read ISR register: ISR[UB] = 0, ISR[SSD] = 1. Write a 1 to the ISR[SSD] bit to clear interrupt

16.1.3.18.4 High-speed Mode: Transmit 1 Byte as a Slave

1. Enable slave address detect interrupts. Set ICR[SADIE]
2. Enable high speed mode
3. Set ICR[MODE] = 0b10 or 0b11
4. A master sends a master code at the standard-mode or fast-mode data rate. Then, a slave address is sent at the high speed mode data rate.
5. Wait for the slave address detect interrupt
6. Read the ISR: slave address detect (1), Unit Busy (1), R/nW bit (1), ACK/NAK (0)
7. Write a 1 to the ISR[SAD] bit to clear the interrupt
8. Load the IDBR with the data to be written (read by the master)
9. Set the ICR[TB] bit
10. When an IDBR transmit-empty interrupt occurs
11. Read ISR: IDBR transmit empty (1), unit busy (1), ACKNAK (1), R/nW bit (0)
12. Write a 1 to the ISR[ITE] bit to clear interrupt
13. Return from interrupt
14. Wait for a slave stop detect interrupt
15. Read ISR: slave stop detect (1), Unit Busy (0)
16. Write a 1 to the ISR[SSD] bit to clear interrupt

16.1.3.18.5 High-speed Mode: Receive 1 Byte as a Slave

1. Enable slave address detect interrupts. Set ICR[SADIE]
2. Enable high speed mode
3. Set ICR[MODE] = 0b10 or 0b11
4. A master sends a master code at the standard-mode or fast-mode data rate. Then a slave address is sent at the high speed mode data rate.
5. Wait for the slave address detect interrupt
6. Read the ISR: slave address detect (1), Unit Busy (1), R/nW bit (0), ACK/NAK (0)
7. Write a 1 to the ISR[SAD] bit to clear the interrupt
8. Set the ICR[TB] bit to initiate transfer
9. When an IDBR receive-full interrupt occurs
10. Read ISR: IDBR receive full (1), unit busy (1), ACKNAK (0), R/nW bit (0)
11. Read IDBR to get the received byte
12. Write a 1 to the ISR[IRF] bit to clear interrupt
13. Return from interrupt
14. Set ICR[TB] bit to release I2C bus and allow the next transfer
15. Wait for a slave stop detect interrupt
16. Read ISR: slave stop detect (1), Unit Busy (0)

17. Write a 1 to the ISR[SSD] bit to clear interrupt

16.1.3.19 Glitch Suppression Logic

The I2C unit has built-in glitch-suppression logic. The glitch suppression logic is implemented differently depending on whether the I2C unit is in **Fast/Standard (F/S) mode** or **High-speed (HS) mode**.

For **F/S mode**, the glitch suppression specification is **50ns**.

- Glitches are suppressed for a length of time given by the formula:
  - Suppression time = 4 * (1 / I2C input clock frequency)
- For example, with a 31.5 MHz input clock frequency, glitches of 127 ns or shorter are suppressed.

For **HS mode**, the glitch suppression specification is **10 ns**.

- Glitches are suppressed for a length of time given by the formula:
  - Suppression time = 1* (1 / I2C input clock frequency)
- For example, with a 61.44 MHz internal input clock frequency, glitches of 16.3 ns or shorter are suppressed.

16.1.3.20 Reset Conditions

When performing a reset on the I2C unit, the following conditions must be met to ensure proper operation:

- **Bus and Unit Status Checks:**
  - Ensure the I2C unit is **not busy** by verifying ISR[UB] = 0 before asserting a reset.
  - After the reset, confirm that the **I2C bus is idle** by checking ISR[IBB] = 0 before enabling the unit.
- **Reset Behavior:**
  - When a reset is triggered, **all registers** return to their **default reset values** except for the **ISAR** register, which remains unchanged.
  - Setting the ICR[UR] bit initiates a reset while preserving the **I2C Memory-Mapped Registers (MMRs)**.

To reset the I2C unit using the **ICR** register, follow these steps:

- Set the reset bit in the ICR register and clear the remainder of the register
- Clear the ISR register
- Clear reset in the ICR

16.1.4 Register Description

Note. The base address of I2C registers are tabled below.

Name	Address
SHUB_I2C_BASE	0xC0887000
I2C0_BASE	0xD4010800

Name	Address
I2C1_BASE	0xD4011000
I2C2_BASE	0xD4012000
I2C3_BASE(SEC_I2C)	0xF0614000
I2C4_BASE	0xD4012800
I2C5_BASE	0xD4013800
I2C6_BASE	0xD4018800
I2C7_BASE	0xD401d000
I2C8_BASE (PWR_I2C)	0xD401d800

16.1.4.1 ICR REGISTER

The bits in the I2C Control register (ICR) are used to control the I2C unit. These are read/write registers. Ignore reads from reserved bits.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RXOV_IE	R/W	0x0	Receive FIFO overrun Interrupt Enable. 0: Receive FIFO overrun (ISR[RXOV]) interrupt is not enabled; 1: Receive FIFO overrun (ISR[RXOV]) interrupt is enabled.
30	RXF_IE	R/W	0x0	Receive FIFO full Interrupt Enable. 0: Receive FIFO full (ISR[RXF]) interrupt is not enabled; 1: Receive FIFO full (ISR[RXF]) interrupt is enabled.
29	RXHF_IE	R/W	0x0	Receive FIFO half full Interrupt Enable. 0: Receive FIFO half full (ISR[RXHF]) interrupt is not enabled; 1: Receive FIFO half full (ISR[RXHF]) interrupt is enabled.
28	TXE_IE	R/W	0x0	Transmit FIFO empty Interrupt Enable. 0: Transmit FIFO empty (ISR[TXE]) interrupt is not enabled; 1: Transmit FIFO empty (ISR[TXE]) interrupt is enabled.
27	TXDONE_IE	R/W	0x0	Transaction Done Interrupt Enable. 0: Transaction done (ISR[TXD]) interrupt is not enabled.; 1: Transaction done (ISR[TXD]) interrupt is enabled.
26	MSDE	R/W	0x0	Master Stop Detected Enable: 0: Master Stop Detect (ISR[MSD]) status is not enabled. 1: Master Stop Detect (ISR[MSD]) status is enabled.
25	MSDIE	R/W	0x0	Master Stop Detected Interrupt Enable: 0: Disable interrupt. 1: Enables the I2C unit to interrupt the upon detecting a

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				Master Stop sent by the I2C unit.
24	SSDIE	R/W	0x0	Slave Stop Detected Interrupt Enable: 0: Disable interrupt. 1: Enables the I2C to interrupt the when it detects a Stop condition while in slave mode.
23	SADIE	R/W	0x0	Slave Address Detected Interrupt Enable: 0: Disable interrupt. 1: Enables the I2C to interrupt the upon detecting a slave address match or a general call address.
22	BEIE	R/W	0x0	Bus Error Interrupt Enable: 0: Disable interrupt. 1: Enables the I2C to interrupt the for the following I2C bus errors:
21	GCD	R/W	0x0	General Call Disable: 0: Enable the I2C to respond to general call messages. 1: Disable I2C response to general call messages as a slave. This bit must be set when sending a master mode general call message from the I2C.
20	DRFIE	R/W	0x0	DBR Receive Full Interrupt Enable: 0: Disable interrupt. 1: Enables the I2C to interrupt the when the IDBR has received a data byte from the I2C bus.
19	ITEIE	R/W	0x0	IDBR Transmit Empty Interrupt Enable: 0: Disable interrupt 1: Enables the I2C to interrupt the after transmitting a byte onto the I2C bus
18	ALDIE	R/W	0x0	Arbitration Loss Detected Interrupt Enable: 0: Disable interrupt. 1: Enables the I2C to interrupt the upon losing arbitration while in master mode.
17	CURSRC_FIX_BYPASS	R/W	0x0	Bypass the cursrc fix : 0: cursrc fix effective 1: bypass the cursrc fix
16	HS_STRETCH_FIX_BYPASS	R/W	0x0	Bypass the hs stretch fix : 0: hs stretch fix effective 1: bypass the hs stretch fix
15	RSVD	R	0	Reserved for future use
14	IUE	R/W	0x0	I2C Unit Enable: 0: Disables the unit and does not master any transactions or respond to any slave transactions.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				1: Enables the I2C (defaults to slave-receive mode). Software must ensure the I2C bus is idle before setting this bit. Software must ensure that the internal clock to the I2C unit is enabled (D0CKEN_B[4] must be set) before setting or clearing this bit.
13	SCLE	R/W	0x0	SCL Enable: 0: Disables the I2C from driving the SCL line. 1: Enables the I2C clock output for master-mode operation.
12	MA	R/W	0x0	Master Abort: Used by the I2C in master mode to generate a Stop without transmitting another data byte: 0: The I2C transmits Stop on if ICR[STOP] is set. 1: The I2C sends Stop without data transmission. In Master-Transmit Mode: - After transmitting a data byte: 1. The ICR[TB] bit is cleared. 2. The IDBR[ITE] bit is set. - When no more data needs to be sent: 1. Set the Master Abort (MA) bit to send a Stop condition. 2. Ensure the ICR[TB] bit remains clear during this operation. In Master-Receive Mode: - If a NAK is sent without a Stop (because ICR[STOP] was not set) and the unit does not send a repeated Start: 1. Setting the MA bit forces a Stop. 2. Again, the ICR[TB] bit must remain clear.
11	I2C_BUS_RESET_REQ_	R/W	0x0	The I2C will do bus reset upon this bit set.this bit is self-cleared
10	UR	R/W	0x0	Unit Reset 0: No reset 1: Reset the I2C only
9:8	MODE	R/W	0x2	Bus Mode (Master operation): 00: Standard-mode: Supports up to 100 Kbps 01: Fast-mode: Supports up to 400 Kbps 10: High-speed (HS) mode: Supports up to 3.3 Mbps in master mode and 3.4 Mbps in slave mode; operates in Standard mode when not performing a high-speed transfer. 11: High-speed (HS) mode: Supports up to 3.3 Mbps in master mode and 3.4 Mbps in slave mode; operates in Fast mode when not performing a high-speed transfer. Bus Mode (Slave operation): 0X: HS-mode Disabled: I2C unit uses Standard/Fast mode

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				timing on the SDA pin. 1X: HS-mode Enabled: I2C unit uses HS-mode timing on the SDA pin when a master code is received.
7	DMA_EN	R/W	0x0	DMA Enable for both TX and RX FIFOs: 0: DMA mode is NOT enabled; 1: DMA mode enabled;
6	GPIOEN	R/W	0x0	GPIO mode Enable for SCL during HS mode. 0: GPIO mode disabled: SCL operates as an open-collector output. 1: GPIO mode enabled: SCL is directly driven by the I2C unit.
5	FIFOEN	R/W	0x0	FIFO mode: 0: FIFO mode disabled; Data is read from or written to the IDBR directly. 1: FIFO mode enabled; Data is managed through the Transmit and Receive FIFOs.
4	TXBEGIN	R/W	0x0	Transaction Begin Set this for a new Transaction only after ISR[TXDONE] is set: 0: No transaction starting; 1: A new transaction begins. This is cleared by the hardware at the end of each transaction after a STOP bit is sent out. The software has to set it again to start a new transaction.
3	TB	R/W	0x0	Transfer Byte: Used to send or receive a byte on the I2C bus: 0: Cleared by I2C when the byte is sent/received. 1: Send/receive a byte. Monitoring this bit can determine when the byte transfer has completed. In master or slave mode, after each byte transfer including acknowledge pulse, the I2C holds the SCL line low (inserting wait states) until TB is set.
2	ACKNAK	R/W	0x0	The positive/negative acknowledge control bit, ACK/NAK, defines the type of acknowledge pulse sent by the I2C when in master receive mode: 0: Send a positive acknowledge (ACK) pulse after receiving a data byte; 1: Send a negative acknowledge (NAK) pulse after receiving a data byte. The I2C automatically sends an ACK pulse when responding to its slave address or when responding in slave-receive mode, regardless of the ACKNAK control-bit setting.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
1	STOP	R/W	0x0	Stop Used to initiate a Stop condition after transferring the next data byte on the I2C bus when in master mode. In master-receive mode, the ACKNAK control bit must be set in conjunction with the STOP bit. 0: Do not send a Stop. 1: Send a Stop.
0	START	R/W	0x0	Start Used to initiate a Start condition to the I2C unit when in master mode. 0: Do not send a Start pulse. 1: Send a Start pulse.

16.1.4.2 ISR REGISTER

I2C interrupts are signaled to the interrupt controller by the I2C Interrupt Status register. Software uses the ISR bits to check the status of the I2C unit and bus. ISR bits (bits 9-5) are updated after the ACK/NAK bit has completed on the I2C bus.

The I2C has transmitted a STOP signal when configured as a master when :

- IDBR receive full
- IDBR transmit empty
- Slave address detected
- Bus error detected
- Stop condition detect
- Arbitration lost

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31	RXOV	R/W1C	0x0	Receive FIFO Overrun (Used in FIFO mode) . 0: Transmit FIFO NOT overrun; 1: Transmit FIFO overrun happened.
30	RXF	R/W1C	0x0	Receive FIFO Full (Used in FIFO mode). 0: Receive FIFO in NOT full; 1: Receive FIFO is full;
29	RXHF	R/W1C	0x0	Receive FIFO Half Full (Used in FIFO mode): 0: Receive FIFO in NOT half full; 1: Receive FIFO is half full.
28	TXE	R/W1C	0x0	Transmit FIFO Empty (Used in FIFO mode). 0: Transmit FIFO is NOT empty; 1: Transmit FIFO is empty.

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
27	TXDONE	R/W1C	0x0	Transaction Done (Used in FIFO mode): 0: Transaction is NOT done. 1: Transaction is done.
26	MSD	R/W1C	0x0	Master Stop Detected: 0: No Master Stop Detected. 1: Set when the I2C detects a Stop while in master-receive or master-transmit mode.
25	RSVD	R	0	Reserved for future use
24	SSD	R/W1C	0x0	Slave Stop Detected: 0: No Stop detected. 1: Set when the I2C detects a Stop while in slave-receive or slave-transmit mode.
23	SAD	R/W1C	0x0	Slave Address Detected: 0: No slave address was detected. 1: The I2C detected a seven-bit address that matches the general call address or ISAR. An interrupt is signaled when enabled in the ICR.
22	BED	R/W1C	0x0	Bus Error Detected: 0: No error detected. 1: The I2C sets this bit when it detects one of the following error conditions: - As a master transmitter, no ACK is detected on the interface after a byte is sent. - As a slave receiver, the I2C generates a NAK pulse.
21	GCAD	R/W1C	0x0	General Call Address Detected: 0: No general call address received. 1: I2C received a general call address.
20	IRF	R/W1C	0x0	IDBR Receive Full: 0: The IDBR has not received a new data byte or the I2C is idle. 1: The IDBR register received a new data byte from the I2C bus. An interrupt is signaled when enabled in the ICR.
19	ITE	R/W1C	0x0	IDBR Transmit Empty: 0: The data byte is still being transmitted. 1: The I2C has finished transmitting a data byte on the I2C bus. An interrupt is signaled when enabled in the ICR.
18	ALD	R/W1C	0x0	Arbitration Loss Detected: Used during multi-master operation: 0: Cleared when arbitration is won or never took place. 1: Set when the I2C loses arbitration.
17	EBB	R	0x0	Early Bus Busy:

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
				0: Bus is idle or the I2C unit is actively using the bus (unit busy). 1: Early Bus Busy: SCL or SDA is low without detecting a START condition. Bit will remain set until the I2C unit detects the bus is idle by detecting a STOP condition. Bit will also be set whenever the IBB bit is set.
16	IBB	R	0x0	I2C Bus Busy: 0: I2C bus is idle or the I2C unit is actively using the bus (unit busy). 1: Bus is busy due to external activity (another master using the bus).
15	UB	R	0x0	Unit Busy 0: I2C not busy. 1: I2C is busy. This is defined as the time between the first Start and Stop.
14	ACKNAK	R	0x0	ACK/NACK Status: 0: The I2C received or sent an ACK on the bus. 1: The I2C received or sent a NAK. On the bus, this bit is used in slave-transmit mode to determine when the byte transferred is the last one. This bit is updated after each byte and ACK/NAK information is received.
13	RWM	R	0x0	Read/write Mode: 0: The I2C is in master-transmit or slave-receive mode. 1: The I2C is in master-receive or slave-transmit mode. This is the R/nW bit of the slave address. It is cleared automatically by hardware after a Stop state.
12:0	RSVD	R	0	Reserved for future use

16.1.4.3 ISAR REGISTER

- The ISAR defines the I2C' s seven-bit slave address.
- In slave-receive mode, the device responds only when the seven-bit address matches the value stored in the ISAR.
- The system writes to the ISAR before enabling I2C operations.
- The ISAR is fully programmable (no address is assigned to the I2C) so it can be set to a value other than those of hard-wired I2C slave peripherals in the system.

These are read/write registers. Ignore reads from reserved bits.

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	R	0	Reserved for future use
6:0	SLAVE_ADDRESS	R/W	0x0	The seven-bit address to which the I2C responds when in slave-receive mode

16.1.4.4 IDBR REGISTER

The I2C Data Buffer Register (IDBR) is used to transmit and receive data from the I2C bus. It is accessed by programmed I/O on one side and the I2C Shift register on the other side.

- **Receiving Data:**
  - The IDBR receives data coming into the I2C unit after a full byte is received and acknowledged.
- **Transmitting Data:**
  - The core writes data going out of the I2C to the IDBR, which then sends it to the serial bus.

16.1.4.4.1 IDBR Operation in Transmit Mode (Master or Slave)

- **Data Writing to IDBR:**
  - When the I2C is in transmit mode (master or slave), the core writes data to the IDBR over the internal bus.
  - Data is written to the IDBR either when a master transaction is initiated or when the IDBR transmit-empty interrupt is signaled.
- **Data Transfer from IDBR to Shift Register:**
  - Data moves from the IDBR to the Shift register when the transfer byte bit is set.
- **Transmit-Empty Interrupt:**
  - The IDBR transmit-empty interrupt (if enabled) is signaled when a byte is transferred on the I2C bus and the acknowledge cycle is complete.
- **Wait States:**
  - If the IDBR is not written by the core and a Stop condition is not in place before the I2C bus is ready to transfer the next byte packet, the I2C unit inserts wait states until the core writes the IDBR and sets the transfer byte bit.

16.1.4.4.2 IDBR Operation in Receive Mode (Master or Slave)

- **Data Reading from IDBR:**
  - When the I2C is in receive mode (master or slave), the core reads data from the IDBR over the internal bus.
- **Receive-Full Interrupt:**
  - The core reads data from the IDBR when the IDBR receive-full interrupt is signaled.
- **Data Movement from Shift Register to IDBR:**
  - Data moves from the Shift register to the IDBR when the acknowledge cycle is complete.
- **Wait States:**
  - The I2C unit inserts wait states until the IDBR is read.
- **Acknowledge Pulse:**
  - For more information on the acknowledge pulse in receive mode, check **Section 16.1.3.10**.
- **Next Byte Transfer:**
  - After the software reads the IDBR, the ICR[ACKNAK] register is written by the software to allow the next byte transfer to proceed on the I2C bus.

These are read/write registers. Ignore reads from reserved bits.

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	R	0	Reserved for future use
7:0	DATA_BUFFER	R/W	0x0	Buffer for I2C bus send/receive data.

16.1.4.5 ILCR REGISTER

The I2C must generate the SCL in master mode and the Load Count Monitor register (ILCR) allows minor adjustments to this clock.

The reset value of this register are determined based on a 31.5 MHz I2C input clock which allows the maximum frequency to be derived for

- Fast mode (up to 400 Kbps)
- Normal mode (up to 100 Kbps)

Due to input clock frequency limitations, the default values of ILCR only generate an SCL than can support up to **1.8 Mbps** for **HS mode**.

For alternate clock setting, an alternate **61.44 MHz I2C input clock** is selected (via writing CCU ACCR1[8] to 1 while the I2C is disabled), the recommended value of ILCR to achieve the proper Fast, Normal and High-Speed SCL frequencies is

- 32’5 h 082CBB56

This register must also be written while the I2C is disabled.

- **The SCL Frequency adjustment:**
- **Increasing** the load value **decreases** the SCL frequency.
- **Decreasing** the load value **increases** the SCL frequency.
- Each increment or decrement corresponds to **one I2C clock period**.

Reset values are designed to allow the highest possible SCL frequency while meeting the **I2C Bus Specification Version 2.1** minimum requirements. This register should be written before enabling the I2C and should not be modified during bus activity.

Writing all zeros to any of the four individual Load Values prevents the I2C from generating an SCL for that specific mode when operating as a master.

Extreme caution is required when modifying this register to avoid disrupting I2C operations.

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:27	HLVL	R/W	0x1	- Load value for High-Speed Mode SCL (master mode) – Low phase. - With the reset value, the I2C in master mode generates an SCL supporting data rates up to 1.8 Mbps.
26:18	HLVH	R/W	0xb	- Load value for High-Speed Mode SCL (master mode) – High phase. - With the reset value, the I2C in master mode generates an SCL supporting data rates up to 1.8 Mbps.
17:9	FLV	R/W	0x5d	- Load value for Fast Mode SCL (master mode) – Both high and low phases. - With the reset value, the I2C in master mode generates an SCL supporting data rates up to 400 Kbps.
8:0	SLV	R/W	0x156	- Load value for Standard Mode SCL (master mode) – Both high and low phases. - With the reset value, the I2C in master mode generates an SCL supporting data rates up to 100 Kbps.

16.1.4.6 IWCR REGISTER

The I2C Wait Count register controls the setup and hold times during slave mode (standard, fast, or high speed).

This register works together with the ILCR register control the setup and hold times for all modes.

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:15	RSVD	R	0	Reserved for future use
14:10	HS_COUNT2	R/W	0x5	Count value for defining high speed mode STOP bit setup and hold times.

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
				Default: Decimal 5
9:5	HS_COUNT1	R/W	0x01	Count value for defining high speed mode START bit setup and hold times. Default: Decimal 1
4:0	COUNT	R/W	0x1A	Controls the counter values defining the setup and hold times in standard and fast mode. Recommended values: 01010 => 33 MHz I2C functional clock; 10100 => 66 MHz I2C functional clock; Default: Decimal 26.

16.1.4.7 IRCR REGISTER

The I2C bus reset cycle counter defines the cycles of SCL during bus reset

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	RSVD	R	0	Reserved for future use
7	I2C_SDA_GLITCH_FIX_BYPASS	R/W	0x0	Bypass the SDA glitch fix: 0: The SDA glitch fix effective; 1: Bypass the SDA glitch fix.
6	I2C_READ_HANG_FIX_BYPASS	R/W	0x0	Bypass the read hang fix. 0: The read hang fix effective; 1: Bypass the read hang fix.
5	SCL_SW_CTRL	R/W	0x0	1: The SCL output is controlled by SW_SCL
4	SW_SCL	R/W	0x0	0: SCL output set to 0; 1: SCL output set to 1;
3:0	RST_CYC	R/W	0x9	The cycles of SCL during bus reset.

16.1.4.8 IBMR REGISTER

The I2C Bus Monitor register (IBMR) tracks the status of the SCL and SDA pins. The values of these pins are recorded in this read-only IBMR, so software can determine when the I2C bus is hung and the I2C unit must be reset.

This a read-only register. Ignore reads from reserved bits.

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	RSVD	R	0	Reserved for future use
1	SCL	R	0x1	IBMR[SCL] continuously reflects the value of the SCL pin.
0	SDA	R	0x1	IBMR[SDA] continuously reflects the value of the SDA pin.

16.1.4.9 WFIFO REGISTER

The I2C Write FIFO has 8 entries and each entry is 12-bit wide (4-bit control + 8-bit data).

This FIFO can be filled up in PIO or DMA mode.

If this FIFO is empty, an interrupt or a DMA request is generated.

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	R	0	Reserved for future use
11:8	CONTROL	R	0x0	I2C Bus send/receive data control bits. These control bits are essential for ICR[3:0] bits.
7:0	DATA	R	0x00	I2C Bus send data for Write Transactions and dummy data for Read Transactions

16.1.4.10 WFIFO_WPTR REGISTER

The I2C Write FIFO Pointer has the TX FIFO write entry location information.

This is a read/write register.

Software can write ‘0’ to this register to flush the FIFO after handling interrupts like Bus error, Arbitration loss, etc.

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	R	0	Reserved for future use
3:0	DATA	R/W	0x0	This is the location in the TX FIFO where the next entry will be written to by the software.

16.1.4.11 WFIFO_RPTR REGISTER

The I2C Write FIFO Read Pointer has the TX FIFO read entry location information.

This is a read/write register.

Software can write ‘0’ to this register to flush the FIFO after an interrupt.

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	R	0	Reserved for future use
3:0	DATA	R/W	0x0	This is the location in the TX FIFO where the next entry will be read from by the hardware.

16.1.4.12 RFIFO REGISTER

The I2C Read FIFO has 16 entries and each entry is 8-bit wide (8-bit data).

This FIFO can be emptied in PIO or DMA mode.

If this FIFO is half full, an interrupt or a DMA request is generated.

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:12	RSVD	R	0	Reserved for future use
11:8	CONTROL	R	0x0	I2C Bus send/receive data control bits. These control bits are essential for ICR[3:0] bits.
7:0	DATA	R	0x00	I2C Bus send data for Write Transactions and dummy data for Read Transactions

16.1.4.13 RFIFO_WPTR REGISTER

The I2C Read FIFO Write Pointer has the RX FIFO write entry location information.

This is a read/write register.

Software can write ‘0’ to this register to flush the FIFO after handling interrupts like Bus error, Arbitration loss, etc.

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	R	0	Reserved for future use
3:0	DATA	R/W	0x0	This is the location in the TX FIFO where the next entry will be written to by the software.

16.1.4.14 RFIFO_RPTR REGISTER

The I2C Read FIFO Read Pointer has the RX FIFO read entry location information.

This is a read/write register.

Software can write ‘0’ to this register to flush the FIFO after an interrupt.

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	R	0	Reserved for future use
3:0	DATA	R/W	0x0	This is the location in the TX FIFO where the next entry will be read from by the hardware.

16.2 SPI/I2S

16.2.1 Introduction

The SPI/I2S is a synchronous serial controller that can be connected to a variety of external Analog-to-Digital converters (ADC), audio and telecommunication codecs and many other devices that use serial protocols for data transfer. The SPI/I2S Controllers directly support the following protocols:

- Motorola* Serial Peripheral Interface (SPI)
- Inter-IC Sound protocol (I2S).

The SPI/I2S operates as full-duplex devices for the SPI and I2S protocols

The SPI/I2S can be configurate to operate in Master mode (the attached peripheral functions as a slave) or Slave mode (the attached peripheral functions as a master). And it support serial bit rates from 6.3 Kbps (minimum recommended speed) up to 52 Mbps. Serial data sample size can be set to 8, 16, 18, or 32 bits in length. A FIFO is provided for transmit data, and a second independent FIFO is provided for Receive data. The two FIFOs are both 32 samples deep x 32 bits wide or 64 samples deep x 16 bits wide.

The FIFOs can be loaded or emptied by CPU using programmed I/O (PIO) or DMA burst transfers.

16.2.2 Features

- Directly support for Motorola* Serial Peripheral Interface (SPI)
- The I2S is supported by programming, and data sample sizes can be set to 8, 16, 18 or 32 bits
- One FIFO for Transmit data (TXFIFO), and a second independent FIFO for Receive data (RXFIFO). For Non-Packed Data mode, the two FIFOs are each 32 rows deep x 32 bits wide for a total of 32 samples
- FIFO Packed mode allows double depth FIFOs if the samples are 8 bits or 16 bits wide. For Packed Data mode, both FIFOs are 64 locations deep x 16 bits wide for a total of 64 samples
- 52 Mbps maximum serial bit-rate
- Master mode and Slave mode operation are supported
- Receive-without-Transmit operation
- Audio clock control to provide a 4x or 8x output clock to support most standard audio frequencies

16.2.3 Functional Description

Data transfers between an SPI/I2S and memory are initiated by the CPU using programmed I/O (PIO) or DMA bursts. Separate Transmit and Receive FIFOs and serial data paths permit simultaneous transfers in both directions to and from the external peripheral, depending on the protocols chosen.

PIO can transfer data between:

- The CPU and the FIFO Data Register for the TXFIFO
- The CPU and the FIFO Data Register for the RXFIFO
- The CPU and the SPI/I2S configuration and status registers

DMA bursts can transfer data between:

- The memory and the FIFO Data Register for the TXFIFO
- The memory and the FIFO Data Register for the RXFIFO

Data written to the FIFO Data Register by either the CPU or DMA is automatically transferred to the Transmit FIFO. When reading the FIFO Data Register by either the CPU or DMA, the “oldest” data in the Receive FIFO is automatically transferred to the FIFO Data Register.

16.2.3.1 SPI/I2S FIFO Access

The data is accessed through the TXFIFO and RXFIFO. A CPU access, that is normally triggered by an interrupt caused by an SPI/I2S Status Register event, takes the form of PIO, transferring one FIFO entry per access and must always be 32-bits wide. The CPU writes to the TXFIFO are 32-bits wide, but the serializing logic ignores all bits beyond the programmed FIFO data size (see SSCR[DSS] fields in the Registers List). The CPU reads from the RXFIFO are also 32-bits wide, but the data that is received by the RXD interface signal is written, with zeroes inserted in the MSBs down to the programmed data size, into the RXFIFO.

The TXFIFO and RXFIFO can also be accessed by DMA bursts, which must be 8, 16, or 32 bytes in length, and must transfer one FIFO entry per access.

The TXFIFO and RXFIFO are each seen as one 32-bit location by the CPU. For data transmission, the SPI/I2S takes the data from the TXFIFO, serializes it, and transmits it via the output serial interface signal to the external peripheral. Data received from the external peripheral via the input interface signal is converted to parallel words and written into the RXFIFO.

A programmable FIFO trigger threshold, when exceeded, generates an interrupt or DMA service request that, if enabled, signals the CPU or DMA, respectively, to empty the RXFIFO or to refill the TXFIFO.

The TXFIFO and RXFIFO are differentiated by whether the access is a Read or a Write transfer. Reads from the FIFO Data Register automatically target the RXFIFO. Writes to the FIFO Data Register automatically target the TXFIFO. From a memory-map perspective, the TXFIFO and the RXFIFO are at the same address. Each FIFO is 32 rows deep x 32 bits wide for a total of 32 data samples. Each sample can be 8, 16, 18, or 32 bits in length.

16.2.3.1.1 FIFO Operation in Packed Mode

When the TXFIFO and RXFIFO are operating in packed mode, each FIFO is 64 rows deep x 16-bits wide for a total of 64 data samples. For packed mode, each sample can be 8 or 16 bits in length.

When the data is serialized and transmitted, Bits 15 to 0 are transmitted first, followed by Bits 31 to 16.

When the TXFIFO and RXFIFO are operating in packed mode, they may best be thought of as a single entry of 32 bits holding two 8- or 16-bit samples. Thus, the CPU or the DMA should write and read 32 bits of data at a time where each Write or Read transfers two samples. The entire FIFO width (32 bits) must be read/written in this mode. The SPI/I2S does not support writing two separate 16-bit samples in this mode. Calculate the thresholds based on the number of 32-bit Writes or Reads, not the number of 16-bit or less values.

Note: At serial bit rates approaching 13 MHz for continuous data transfers, the DMA might not be able to access the RXFIFO or TXFIFO fast enough to avoid overflow or underflow, respectively. Using packed mode improves performance.

16.2.3.2 Trailing Bytes in RXFIFO

When the number of samples in the RXFIFO is less than its trigger threshold level and no additional data is received, the remaining bytes are called RXFIFO trailing bytes. RXFIFO trailing bytes can be handled by either the CPU or by DMA, as indicated by the Trailing Byte field in the SSCR[TRAIL]. RXFIFO trailing bytes are identified by means of a time-out mechanism and the existence of data within the RXFIFO after timeout.

Note: When FIFO packed mode is used, the DMA can not be used to handle the RXFIFO trailing bytes. The RXFIFO trailing bytes must be handled by the CPU.

16.2.3.2.1 Timeout

A timeout condition exists when the RXFIFO has been idle for a period of time defined by the value programmed within the SSTO[TIMEOUT] field. When a timeout occurs, the Receiver Time-out Interrupt bit, SSSR[TINT], is set to 1, and if the Receiver Time-out Interrupt Enable bit, SSINTEN[TINTE], is set, a timeout interrupt signals the CPU that a timeout condition has occurred. The timeout timer is reset after a new data sample is received into the RXFIFO. Once the SSSR[TINT] bit is set, it must be cleared by writing 0x1 to the SSSR[TINT] bit. Clearing it also causes the timeout interrupt, if enabled, to be de-asserted.

16.2.3.2.2 Peripheral Trailing Byte Interrupt

It is possible for the DMA to reach the end of its Descriptor chain while removing RXFIFO data. When this happens, the CPU must take over because the DMA can no longer service the SPI/I2S until a new chain is linked. When the DMA has reached the end of its Descriptor chain with data in the RXFIFO, the SPI/I2S:

- Sets the Peripheral Trailing Byte Interrupt bit, SSSR[PINT].
- Asserts the SPI/I2S interrupt to signal to the CPU that a peripheral trailing-byte interrupt condition has occurred (if the interrupt is enabled by setting the Peripheral Trailing Byte Interrupt Enable bit, SSINTEN[PINTE]).
- Sets the End Of Chain, SSSR[EOC]. If more data is received after the SSSR[EOC] field was set (and it remains set), then the SSSR[PINT] field is set. The SSSR[EOC] field must be cleared by writing 0x1 to it.
- Once the SSSR[PINT] field is set, the CPU must clear the bit by writing 0x1 to it. Clearing it also de-asserts the SPI/I2S interrupt if it has been enabled (SSINTEN[PINTE] = 1).

The remaining bytes must then be removed with the PIO method by the CPU as described in Removing FIFO Trailing Bytes or by reprogramming a new Descriptor chain and restarting the DMA. Programmers need to be aware of this possibility. For details, refer to **Section 9.4.3.5**, How DMA Handles Trailing Bytes.

16.2.3.2.3 Removing FIFO Trailing Bytes

When the Trailing Byte, SSCR[TRAIL], bit is cleared, trailing bytes left in the RXFIFO are handled by the CPU programmed I/O method. This is the default method.

If a timeout occurs, the CPU is only interrupted by a timeout interrupt if it has been enabled by setting the Receiver Time-out Interrupt Enable SSINTEN[TINTE] field. To read out the trailing bytes from the RXFIFO, software should wait for the timeout interrupt and then read all trailing bytes as indicated by the Odd Sample Status, SSSR[OSS], Receive FIFO Level, SSSR[RFL], and Receive FIFO Not Empty, SSSR[RNE] fields. To remove trailing bytes using PIO, enable the timeout interrupt by setting the SSINTEN[TINTE] field.

Note. If FIFO Packed mode is enabled (SSFCR[FPCKE]=1), trailing bytes must be removed using programmed I/O. If the SSSR[OSS] field is set, then the last FIFO line only contains one sample.

When the Trailing Byte SSCR[TRAIL] bit is set, trailing bytes left in the RXFIFO are handled by the DMA controller.

A DMA service request is issued automatically after the SSCR[TRAIL] field is set and a timeout occurs, SSSR[TINT]=1. The DMA empties the RXFIFO unless the DMA reaches the end of its Descriptor chain. When handling trailing bytes using the DMA, if a timeout occurs and the RXFIFO is empty (SSSR[RNE]=0), an end-of-receive (EOR) is sent to the DMA.

If a DMA EOC occurs (SSSR[EOC]=1) at the time that the last sample is read out of the RXFIFO (the DMA Descriptor chain was just exactly long enough) and the timeout counter is still running (that is, a timeout has not occurred and SSTO[TIMEOUT] is non-zero), then, when the timeout does occur, the SPI/I2S generates a DMA request. When this occurs, re-initialize the DMA registers and re-enable the channel for the SPI/I2S to send its EOR to the DMA controller.

Notes.

- When the SPI/I2S is running in Network mode and the FIFO Packing Enable SSFCR[FPCKE] bit is set, use the CPU to handle trailing bytes in the RXFIFO with the SSCR[TRAIL] bit cleared and the SSINTEN[PINTE] bit set. After the Peripheral Trailing Byte Interrupt SSSR[PINT]=1 occurs, the interrupt service routine must clear the SSNWC[MOD] bit and wait for SSNWS[NMBSY] bit to go low before removing any extra or trailing samples from the RXFIFO, which can be discarded.
- When the SSNWC[MOD] bit is set, the SPI/I2S continues transceiving data even after the TXFIFO is empty and until the SSNWC[MOD] bit is cleared. Since the DMA does not have a way to clear the SSNWC[MOD] bit, it is possible that extra samples are received in the RXFIFO. Since software must clear the SSNWC[MOD] bit, the CPU must also handle the trailing bytes.

16.2.3.3 Data Formats

The types of formats used to transfer serial data between the CPU and external peripherals are described in the following subsections.

16.2.3.3.1 Serial Data Formats for Transfer to/from Peripherals

Two interface signals for each SPI/I2S transfer data between the CPU and external peripherals. Although serial-data formats exist, each has the same basic structure, and in all cases, the interface signals used are:

- **SS_SCLK** - Defines the bit rate at which serial data is driven onto and sampled from the port
- **SS_FRM** - Defines the boundaries of a basic data “unit” which is comprised of multiple serial bits

- **SS_TX** - The serial datapath for transmitted data from the SPI/I2S to the peripheral
- **SS_RX** - The serial datapath for received data from peripheral to the SPI/I2S

A data frame can contain 8, 16, 18, or 32 bits (SSCR[DSS] fields. Serial data is transmitted with the MSb first. The formats directly supported are the Motorola* SPI, and the I2S protocol is supported by programming the PSP format.

The SS_FRM function and use varies between each format. SS_FRM is programmable in direction, delay, polarity, and width. Both Master and Slave modes are supported.

- SPI format: SS_FRM functions as a chip select to enable the external device (target of the transfer) and is held active-low during the data transfer. During continuous transfers, the SS_FRM signal can be either held low or pulsed depending upon the value of the Motorola* SPI SS_SCLK phase setting in SSCR[SPH], Master and Slave modes are supported. SPI is a full-duplex format.
- PSP format (I2S): SS_FRM is programmable in direction, delay, polarity, and width. Master and Slave modes are supported. PSP can be programmed to be either full- or half-duplex format. I2S is supported by programming PSP format.

The SS_SCLK function and use varies between each format:

- SPI format: Programmers choose which edge of SS_SCLK to use for switching Transmit data and for sampling Receive data. In addition, moving the phase of SS_SCLK can be user-initiated, shifting its active state one-half cycle earlier or later at the start and end of a frame. Master and Slave modes are supported, and in both, the SS_SCLK only toggles during active transfers (does not run continuously).
- PSP format (I2S): Programmers choose which edge of SS_SCLK to use for switching Transmit data and for sampling Receive data. In addition, programmers can control the Idle state for SS_SCLK and the number of active clocks that precede and follow the data transmission. Master and Slave modes are supported. When driven by the SPI/I2S port, the SS_SCLK toggles only during active transfers, not continuously. When the SS_SCLK is driven by another device, it is allowed to be either continuous or driven only during transfers, but certain restrictions on PSP parameters apply.

Normally, if the serial clock (SS_SCLK) is driven by the SPI/I2S port, it toggles only while an active data transfer is underway. However, there are several conditions that may cause the clock to run continuously. If the Receive-without-Transmit mode is enabled by setting the Receive Without Transmit, SSRWT[RWOT], the SS_SCLKtoggles regardless of whether Transmit data exists within the Transmit FIFO. The SS_SCLK also toggles continuously if the SPI/I2S port is in Network mode. At other times, SS_SCLK is held in an inactive or idle state, as defined by the specified protocol under which it operates.

16.2.3.3.2 Motorola* SPI Format

The SPI format has four possible sub-modes depending on the SS_SCLK edges selected for driving data and sampling received data and on the selection of the phase mode of SS_SCLK for a complete description of each sub-mode).

Note. The following description applies only when SPH = 0 and SPO = 0. Other combinations of SPH and SPO result in different polarities and timings.

When the SPI/I2S is disabled or in idle mode, SS_SCLK and SS_TX are low and SS_FRM is high. When Transmit data is ready to be sent, SS_FRM goes low (one clock period before the first rising edge of SS_SCLK) and stays low for the remainder of the frame. The most significant bit of the serial data is driven onto SS_TX one half-cycle later. Halfway into the first bit period, SS_SCLK asserts high and continues toggling for the remaining

data bits. Data transitions on the falling edge of SS_SCLK and is sampled on the rising edge of SS_SCLK. 8, 16, 18, or 32 bits can be transferred per frame.

With the assertion of SS_FRM, Receive data is driven simultaneously from the peripheral on SS_RX , MSb first. Data transitions on SS_SCLK falling edges and is sampled by the controller on SS_SCLK rising edges. At the end of the frame, SS_FRM is de-asserted high one clock period (one half clock cycle after the last falling edge of SS_SCLK) after the last bit has been latched at its destination and the completed incoming word is shifted into the “incoming” FIFO. The peripheral can drive SS_RX to a high-impedance state after sending the last bit of the frame.

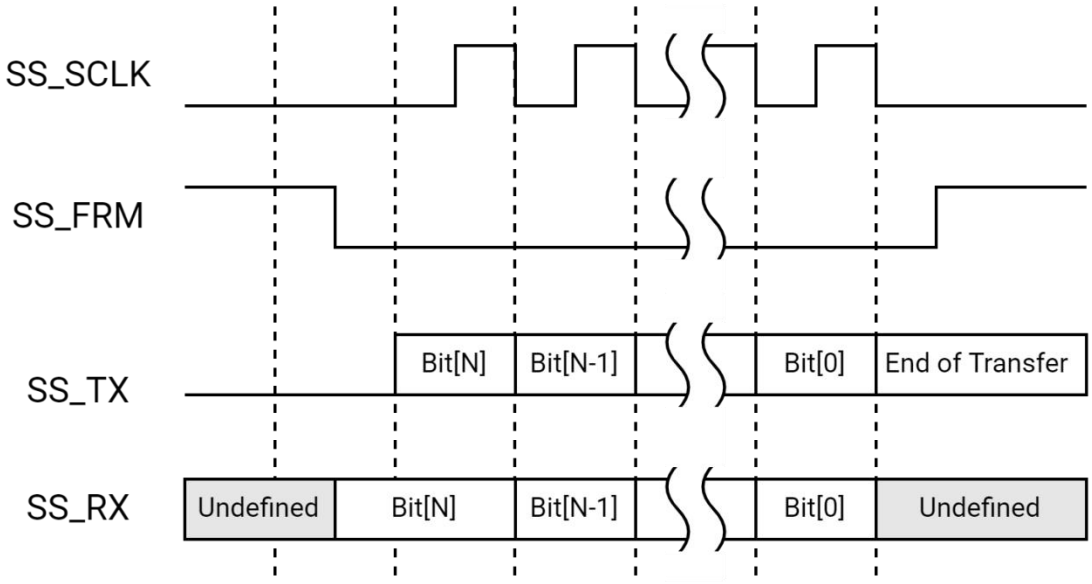
SS_TX retains the last value transmitted when the controller goes into Idle mode, unless the SPI/I2S is disabled or reset (which forces SS_TX to zero).

For back-to-back transfers, start and completion are like those of a single transfer, but SS_FRM does not de-assert between words. Both transmitter and receiver are configured for the word length and internally track the start and end of frames. There are no “dead” bits; the LSb of one frame is followed immediately by the MSb of the next.

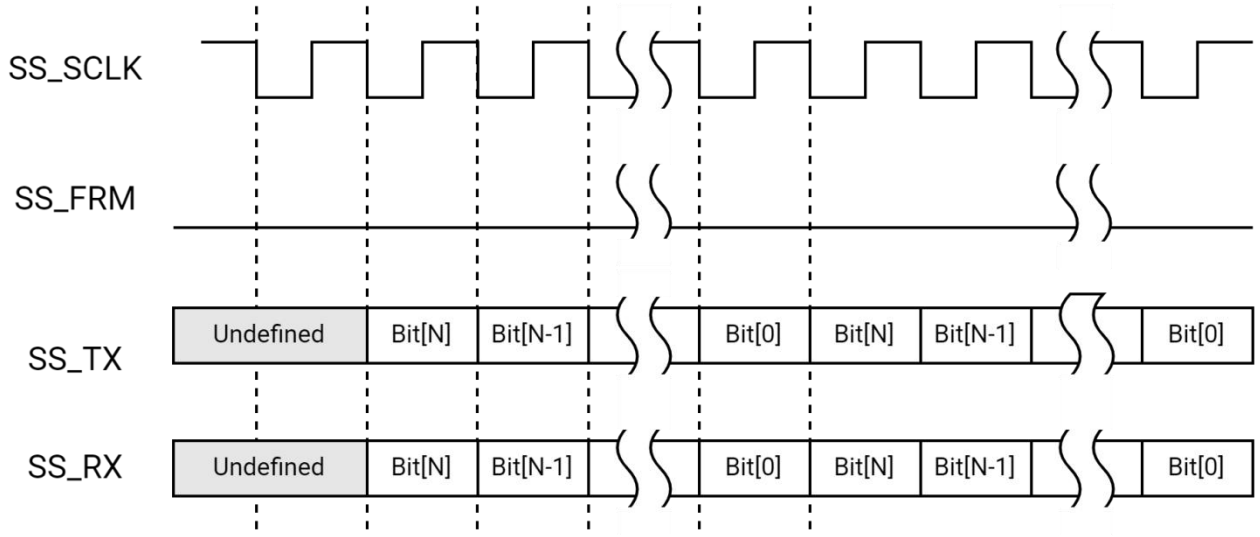
When in Motorola* SPI format, the SPI/I2S can be either a master or a slave device, but the clock and frame direction must be the same. For example, the Serial Bit Rate Clock Direction, SSCR[SCLKDIR], and the Frame Direction, SSCR[SFRMDIR], fields must either both be set or cleared.

When in Motorola* SPI format, if the SPI/I2S is the master and SSPSP[ETDS] is cleared, the end-of-transfer data state for SS_TX is low. If the SPI/I2S is the master and SSPSP[ETDS] is set, the end-of-transfer data state for SS_TX remains at the last bit transmitted (LSB). If the SPI/I2S is the slave, then the SSPSP[ETDS] is undefined. SS_RX is undefined before the frame is active and after the LSB is received. SS_RX must not float. When the SPI/I2S is configured as a master and SSCR[TTE] is set, SSPSP[ETDS] is ignored and SS_TX becomes high impedance between active transfers.

Below are depicted the four possible configurations for the Motorola* SPI frame protocol for a single transmitted frame.



Instead, below is depicted how back-to-back frames are transmitted for the Motorola* SPI frame protocol.



Note. The phase and polarity of SS_SCLK can be configured for four different modes. This example shows just one of those modes (SSCR[SPO] and SSCR[SPH] cleared). Other settings for SPO and SPH result in different polarities and timing.

16.2.3.3.3 Programmable Serial Protocol (I2S) Format

The PSP format defines programmable parameters that determine the transfer timings between data samples and used for I2S protocol.

Four serial clock modes are defined in the Serial Bit-rate Clock Mode, SSPSP[SCMODE]. These modes select the SS_SCLK rising and falling edges for driving data, sampling received data, and the SS_SCLK idle state.

The Idle and Disabled modes of the SS_TX, SS_SCLK, and SS_FRM interface signals are programmable using the following fields in the SSPSP Register: End Of Transfer Data State (SSPSP[ETDS]), Serial Frame Polarity (SSPSP[SFRMP]), and Serial Bit-rate Clock Mode (SSPSP[SCMODE]). When transmit data is ready, SS_SCLK remains in its idle state for the number of serial clocks (SS_SCLK) periods programmed into the Start Delay (SSPSP[STRTDLY]) field in the SSP Programmable Serial Protocol Register.

SS_SCLK then starts toggling. SS_TX remains in the idle state for the number of serial clock periods programmed into the Dummy Start (SSPSP[DMYSTRT]) field. SS_FRM is asserted after the number of half serial clock periods programmed into the Serial Frame Delay (SSPSP[SFRMDLY]) field. SS_FRM remains asserted for the number of serial clock periods programmed into the Serial Frame Width (SSPSP[SFRMWDTH]) field, then SS_FRM de-asserts.

Serial data of 8, 16, 18, or 32 bits can be transferred per frame by setting the SSCR[DSS] fields to the preferred data size select. Once the last bit (LSb) is transferred, SS_SCLK continues toggling for the number of serial clock periods programmed into the Dummy Stop (SSPSP[DMYSTOP]) field. Depending on the value programmed into the End Of Transfer Data State (SSPSP[EDTS]) field when the SPI/I2S port goes into idle mode, SS_TX either retains the last bit-value transmitted or is forced to 0 unless the SPI/I2S port is disabled or reset, which forces SS_TX to 0.

With the assertion of SS_FRM, Receive data is driven simultaneously from the peripheral onto SS_RX, MSb first. Data transitions on the SS_SCLK edge based on the serial-clock mode that is selected (SSPSP[SCMODE]) and is sampled by the SPI/I2S port on the opposite clock edge. When the SPI/I2S port is a master to SS_FRM and a slave to SS_SCLK, at least three extra SS_SCLKs are needed at the beginning and end of each block of transfers to synchronize control signals from the APB clock domain into the SPI/I2S clock domain (a block of transfers is a group of back-to-back continuous transfers).

In general, because of the programmable nature of the PSP protocol, this protocol can be used to achieve a variety of serial protocols, including I2S.

The programmable protocol parameters of SPI/I2S are tabled below.

Symbol	Definition	Range	Units
-	Serial clock mode (SSPSP[SCMODE])	(Drive, Sample, SS_ SCLK Idle) 0 = Fall, rise, low 1 = Rise, fall, low 2 = Rise, fall, high 3 = Fall, rise, high	-
-	Serial frame polarity (SSPSP[SFRMP])	High or low	-
T1	Start delay (SSPSP[STRTDLY])	0 to 7	Clock period
T2	Dummy start (SSPSP[EDMYSTRT] + SSPSP[DMYSTRT])	0 to 15	Clock period
T3	Data size (SSCR[DSS])	4 to 32	Clock period
T4	Dummy stop(SSPSP[EDMYSTOP] + SSPSP[DMYSTOP])	0 to 31	Clock period
T5	SS_FRM delay (SSPSP[SFRMDLY])	0 to 127	Half-clock period
T6	SS_FRM width (SSPSP[SFRMWDTH])	1 to 63	Clock period
-	End of transfer data state (SSPSP[ETDS])	Low or bit 0	-

The SS_FRM delay (T5) must not extend beyond the end of T4. The SS_FRM width (T6) must be asserted for at least one SS_SCLK period and should be de-asserted before the end of T4 (for example, in terms of time, not bit values

- (T5 + T6) <= (T1 + T2 + T3 + T4)
- 1<= T6 < (T2 + T3 + T4)
- (T5 + T6) >= (T1 + 1)

to ensure that SS_FRM is asserted for at least two edges of SS_SCLK). Program T1 to 0 when SS_SCLK is enabled by the SSCR[SCFR] fields.

While the SPI/I2S can be programmed to generate the assertion of SS_FRM during the middle of the data transfer (for example, after the MSB has been sent), the SPI/I2S port is unable to Receive data in frame-Slave mode. Transmit data transitions from the end-of-transfer-data state (SSPSP[ETDS]) to the next MSB data value upon assertion of the internal version of SS_FRM. Program the SSPSP[STRTDLY] field to 0x00 whenever SS_SCLK or SS_FRM is configured as an input (for example, SSCR[SCLKDIR] and SSCR1[SFRMDIR] are cleared.

Note. When the SPI/I2S port is slave to the frame, the sum of T1+T2+T3+T4 can be less than the actual time from the beginning of the current frame to the beginning of the next frame. For example, when the rate of

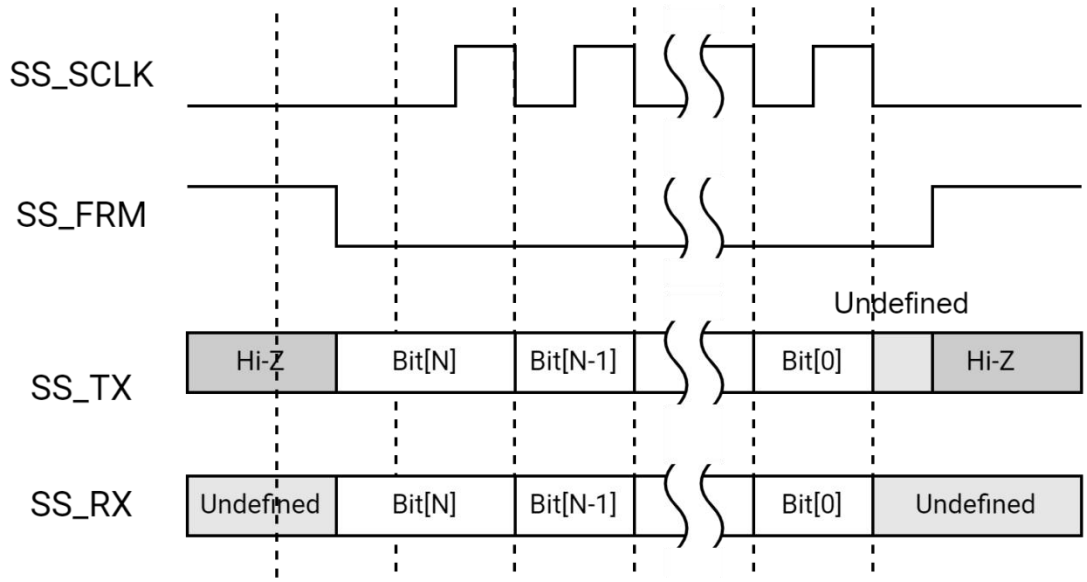
SS_SCLK is 12.8 MHz and the data sample size is 16-bits, the beginning of the frame can occur at a rate of 8 kHz.

16.2.3.4 High Impedance on SS_TX

The SPI/I2S supports placing the SS_TX into high impedance during idle time instead of driving SS_TX as controlled by the TXD Three-State Enable (SSCR[TTE]) and TXD Three-state Enable On Last Phase (SSCR[TTELP]) field. The SSCR[TTE] enables a high-impedance state on SS_TX. The SSCR[TTELP] determines on which SS_SCLK phase SS_TX becomes high impedance.

16.2.3.4.1 Motorola* SPI Format

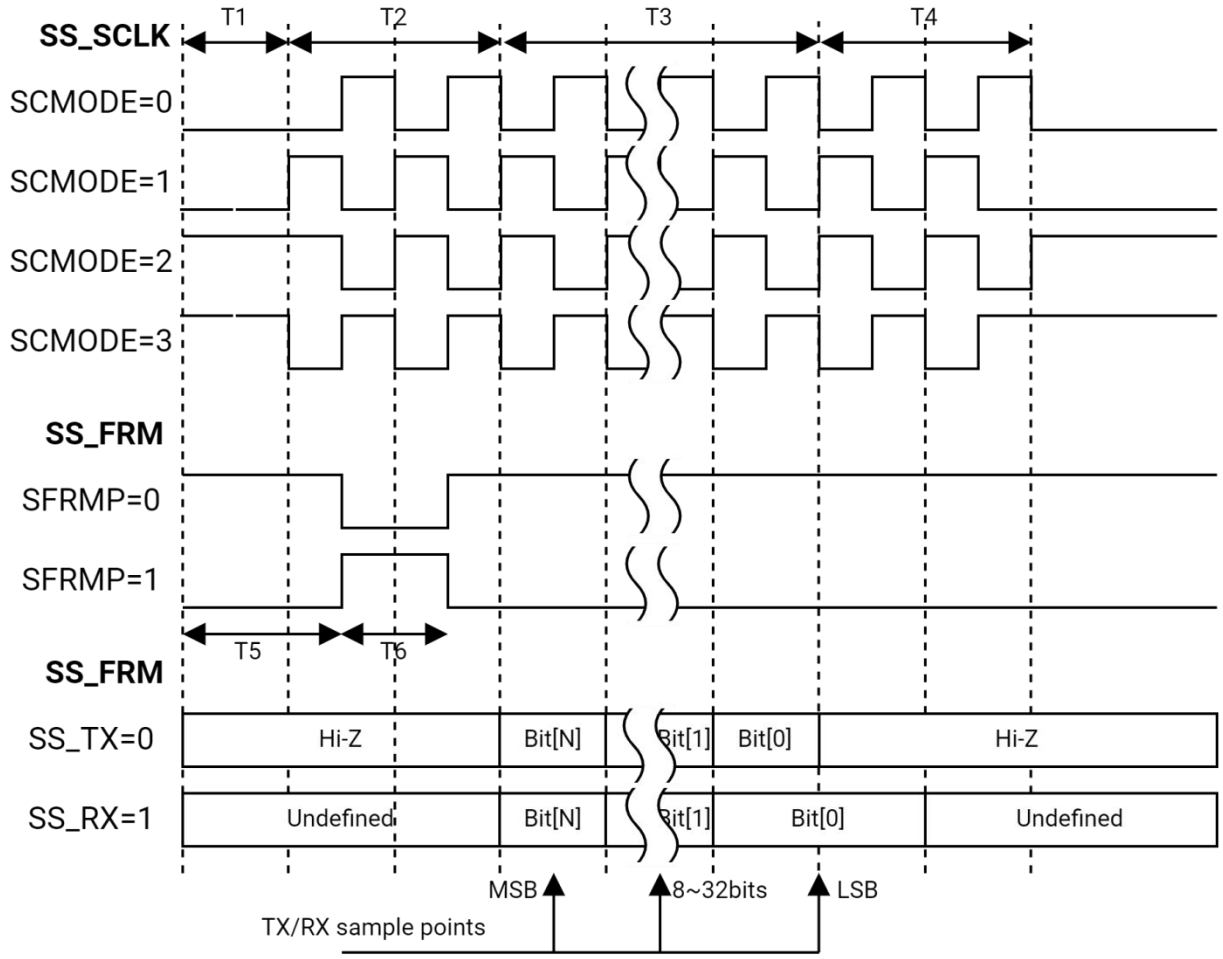
For Motorola* SPI format, SSCR[TTELP] must be cleared. If SSCR[TTE] = 1. SS_TX functionality is depicted below.



For Motorola* SPI format, SS_TX goes to a high-impedance state whenever SS_FRM is not asserted.

16.2.3.4.2 PSP Format

SS_TX functionality when SSCR[TTE] = 1, SS_SSCR[TTELP] = 0 and SSCR[SFRMDIR] = 1, are depicted below.



16.2.3.5 Network Mode Operation

The SSNWC[MOD] bit selects between Normal and Network modes. Normal mode (MOD = 0x0) is used when using the Motorola* Serial Peripheral Interface (SPI). Network mode (MOD = 0x1) is used for the I2S protocol.

Software should set MOD only when using the PSP format. If the SPI/I2S port is a master of the clock and SSCR[SCLKDIR] is cleared, then setting MOD causes the SS_SCLK to run continuously.

When in Network mode, only one SS_FRM is sent (Master mode) or received (Slave mode) for the number of time slots programmed into the SSNWC[FRDC] field. When beginning in Network mode, while the SPI/I2S port is a master to the SS_FRM interface signal, the first SS_FRM signal does not occur until after data is in the TXFIFO. After assertion of the first SS_FRM signal, if the SPI/I2S port is a master to SS_FRM, subsequent SS_FRM signals continue to assert regardless of whether data resides in the TXFIFO. Therefore, the transmit underrun bit, SSSR[TUR], is set to 1 if there is no data in the TXFIFO and the SPI/I2S port is programmed to drive SS_TX data in the current time slot, even if the SPI/I2S port is master to SS_FRM. When using PSP format in Network mode, the parameters SFRMDLY, STRTDLY, DMYSTOP, DMYSTRT must all be 0. The other parameters SFRMP, SCMODE, FSRT, SFRMWDTH are programmable.

When the SPI/I2S port is a master to the SS_FRM signal and a need arises to exit from Network mode, software should:

- Clear the SSNWC[MOD] bit. SSCR[SSE] does not need to change
- Wait until SSNWS[NMBSY] is cleared

- Disable the SPI/I2S port by clearing SSCR[SSE]
- Before exiting Network mode, verify the TXFIFO is empty (SSSR[TFL]=0 and SSSR[TNF]=1)
- If data remains in the TXFIFO after the Network mode is exited, a non-Network mode frame is sent

Due to synchronization delay between the internal bus and the SPI/I2S port clock domain, one extra frame may be transmitted after software clears the SSNWC[MOD] bit. The SPI/I2S port continues to drive SS_SCLK (if SSCR[SCLKDIR] is cleared) and SS_FRM (if SSCR[SFRMDIR] is cleared) until the end of the last valid time slot.

If the SPI/I2S port is a slave to both SS_SCLK (SSCR[SCLKDIR] set) and SS_FRM (SSCR[SFMRDIR] set), the SSNWS[NMBSY] bit remains asserted until the SSNWC[MOD] bit is cleared or until one SS_SCLK after the end of the last valid time slot.

16.2.3.6 Parallel Data Formats for FIFO Storage

All CPU and DMA accesses transfer one FIFO entry per access. Data in the FIFOs is either stored with one 32-bit value per data sample (in non-packed or sample > 16 bits) or in a 16-bit value in packed mode when the data is 4 or 16 bits.

Within each 32- or 16-bit field, the stored data sample is right-justified, with the LSb of the word in bit 0. In the Receive FIFO, unused bits are packed as zeroes above the MSb. In the Transmit FIFO, unused “don’t-care” bits are above the MSb.

For example, DMA and CPU accesses do not have to write to the unused bit locations. Logic in the SPI/I2S automatically formats data in the Transmit FIFO so that the sample is properly transmitted on SS_TX in the selected frame format.

16.2.3.7 FIFO Operation

This section describes the operation of Transmit and Receive FIFOs.

Two separate and independent FIFOs are present for transmitting (TXFIFO to peripheral) and receiving (RXFIFO from peripheral) serial data. The FIFOs are filled or emptied by programmed I/O or DMA bursts.

16.2.3.7.1 Using Programmed I/O Data Transfers

FIFO filling and emptying can be performed by the CPU in response to an interrupt from the FIFO logic. Each FIFO has a programmable FIFO trigger threshold that triggers an interrupt.

When the number of entries in the RXFIFO exceeds the RXFIFO Trigger Threshold (SSFCR[RFT]) field, an interrupt is generated (if enabled) that signals the CPU to empty the RXFIFO. When the number of entries in the TXFIFO is less than or equal to the TXFIFO Trigger Threshold (SSFCR[TFT]) field plus 1, an interrupt is generated (if enabled) that signals the CPU to refill the TXFIFO.

The SSSR can be polled to determine how many samples are in a FIFO and whether the FIFO is full or empty. Software is responsible for ensuring that the proper RXFIFO Trigger Threshold and TXFIFO Trigger Threshold values are chosen to prevent Receive FIFO Overrun and Transmit FIFO Underrun error conditions.

16.2.3.7.2 Using DMA Data Transfers

The DMA controller can also be programmed to transfer data to and from the FIFOs. To prevent overruns of the TXFIFO or underruns of the RXFIFO when using the DMA, be careful when setting the FIFO trigger threshold levels by setting the correct DMA burst sizes. TXFIFO overruns and RXFIFO underruns are silent errors: There is no indication of the overrun or underrun condition other than missing data at the receiving end of the link. The DMA burst size must be smaller than the trigger threshold.

The programming model for using DMA is:

- Program the total number of transmit/receive byte lengths, burst sizes, and peripheral width.
- When not using the FIFO packed mode, program the Width field in the DMA Command Registers to 0x1 for FIFO data sizes of 8 bits, 0x2 for FIFO data sizes of 16 bits, and 0x3 for FIFO data sizes of more than 16 bits. When not using packed mode, the SPI/I2S stores one data sample per FIFO location where each FIFO has 16 locations. For example, the DMA burst size must not exceed 16 bytes when Width field in the DMA Command Registers is set to 0x1 (byte wide).
- When using FIFO packed mode, program the Width field in the DMA Command Registers to 0x3. When using packed mode, the SPI/I2S stores two data samples per FIFO location where each FIFO has 16 locations. Therefore, the DMA burst size must not exceed 16 bytes when Width field in the DMA Command Registers is set to 0x3 (more than 16 bits wide).
- Because the SPI/I2S is not flow-controlled and has only 16 location FIFOs, software must program the TXFIFO threshold (SSFCR[TFT]) field, RXFIFO threshold (SSFCR[RFT]) field, and the DMA burst size to ensure that a TXFIFO overrun or RXFIFO underrun does not occur. Software must also ensure that the SPI/I2S DMA requests are properly prioritized in the system to prevent overruns and underruns.
- Program the preferred values into the SSCR
- Enable the SPI/I2S by setting the Synchronous Serial Port Enable (SSCR[SSE]) field
- Set the run bits in the DMA Command Register
- The DMA waits for either the TXFIFO or RXFIFO service request
- If the receive byte length is not an even multiple of the transfer burst size, a trailing-byte condition may occur

In full-duplex formats where the SPI/I2S always receives the same number of data samples that it transmits, the DMA channel should be set up to transmit and receive the same number of bytes.

Note. When the FIFO Packing Enable (SSFCR[FPCKE]) field is set to 0x1, the SSFCR[TFT] and SSFCR[RFT] fields represent twice the number of FIFO entries as when the packing enable bit is 0x0. So, when in packed mode, the maximum allowed DMA burst size (8 or 16) could be doubled.

16.2.3.8 Baud-Rate Generation

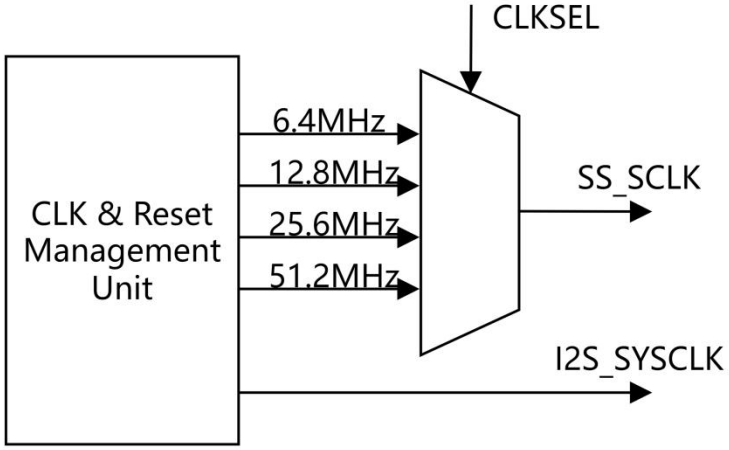
When the SPI/I2S is configured as the master (output) of SS_SCLK as determined by clearing the SSCR[SCLKDIR] field, the baud rate (or serial bit-rate clock SS_SCLK) is obtained by selecting from one of the four fixed clock sources (6.4 MHz, 12.8 MHz, 25.6 MHz, or 51.2 MHz). A variable clock source is available from the output of two dithering dividers external to the SPI/I2S unit. Clock selection is achieved by writing to the Functional Clock Select field of the SPI/I2S Clock/Reset Control Register.

When the source clock is to be changed, software must:

- Disable the SPI/I2S port by writing 0x0 to SSCR[SSE]
- Disable the SPI/I2S port internal clock by clearing the appropriate bit in the Clock Enable Register
- Write clock and reset related registers to enable functional clocks to the SPI/I2S units

- If applicable, set appropriate values for the frequency dividers
- Enable APB clock to the SPI/I2S unit
- Set the SSCR[SSE] bit to re-enable the SPI/I2S port. Whenever the baud rate is to be changed, software must:
  - Disable the SPI/I2S port by clearing SSCR[SSE] bit
  - Set the SSCR0[SSE])bit to re-enable the SPI/I2S port

Wait two SS_SCLK cycles before writing new data to the TXFIFO. The SPI/I2S Baud Rate Generation is depicted below.



16.2.4 Register Description

Note. The base address of SPI/I2S registers are tabled below.

Name	Address
SHUB_SSP0_BASE	0xC0885000
SSP3_BASE	0xD401C000
SSPA0_BASE	0xD4026000
SSPA1_BASE	0xD4026800
SSP2_BASE	0xF0613000

16.2.4.1 SSCR REGISTER

SPI/I2S top control register.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:19	RSVD	R	0	Reserved for future use

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
18	TTELP	R/W	0x0	TXD Three-state Enable On Last Phase 0 = SS_TX is three-stated 1/2 clock cycle after the beginning of the LSB 1 = SS_TX output signal is three-stated on the clock edge that ends the LSB
17	TTE	R/W	0x0	TXD Three-State Enable 0 = SS_TX output signal is not three-stated 1 = SS_TX is three-stated when not transmitting data
16	SCFR	R/W	0x0	Slave Clock Free Running 0 = Clock input to SS_CLK is continuously running 1 = Clock input to SS_CLK is only active during data transfers.
15	IFS	R/W	0x0	Invert Frame Signal 0 = SS_FRM polarity is determined by the PSP polarity bits 1 = SS_FRM will be inverted from normal-SS_FRM (as defined by the PSP polarity bits). (Works in all frame formats: SPI and PSP)
14	HOLD_FRAME_LOW	R/W	0x0	Hold Frame Low Control 1=After this field is set to 1 and the SPI/I2S is operating in master mode. Used for SPI Format Rx FIFO Auto Full Control, which makes the frame clock is still low during there's no bit clock, or the data transfers before the stop clock will be discarded.
13	TRAIL	R/W	0x0	Trailing Byte 0 = Trailing bytes are handled by the CPU 1 = Trailing bytes are handled by DMA bursts
12	LBM	R/W	0x0	Loopback Mode (Test Mode Bit) 0 = Normal serial port operation is enabled 1 = Output of TX serial shifter is internally connected to input of RX serial shifter
11	SPH	R/W	0x0	Motorola SPI SS_SCLK phase setting 0 = SS_SCLK is inactive until one cycle after the start of a frame and active until 1/2 cycle before the end of a frame 1 = SS_SCLK is inactive until 1/2 cycle after the start of a frame and active until one cycle before the end of a frame
10	SPO	R/W	0x0	Motorola SPI SS_SCLK Polarity Setting 0 = The inactive or idle state of SS_SCLK is low

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				1 = The inactive or idle state of SS_SCLK is high
9:5	DSS	R/W	0x0	SPI/I2S Work data size, register bits value 0~31 indicated data size 1~32 bits, usually use data size 8bits, 16bits, 24bits, 32bits
4	SFRMDIR	R/W	0x0	SS_FRM Direction 0 = Master mode, SPI/I2S port drives SS_FRM 1 = Slave mode, SPI/I2S port receives SS_FRM
3	SCLKDIR	R/W	0x0	SS_SCLK Direction 0 = Master mode, SPI/I2S port drives SS_SCLK 1 = Slave mode, SPI/I2S port receives SS_SCLK
2:1	FRF	R/W	0x0	Frame Format 0x0 = Motorola* Serial Peripheral Interface (SPI) 0x3 = Programmable Serial Protocol (PSP) Others = reserved
0	SSE	R/W	0x0	SPI/I2S Enable 0 = SPI/I2S port is disabled 1 = SPI/I2S port is enabled

16.2.4.2 SSFCR REGISTER

SPI/I2S FIFO control register.

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:22	RSVD	R	0	Reserved for future use
21	RXENDIAN	R/W	0x0	select the big endian or little endian for RX transfer 0 = big endian 1 = little endian
20	TXENDIAN	R/W	0x0	select the big endian or little endian for TX transfer 0 = big endian 1 = little endian
19	STRF	R/W	0x0	Select FIFO For Test Mode Bit 0 = TXFIFO is selected for both writes and reads through the SPI/I2S Data Register 1 = RXFIFO is selected for both writes and reads through the SPI/I2S Data Register
18	EFWR	R/W	0x0	Enable FIFO Write/read (Test Mode Bit) 0 = FIFO write/read special function is disabled 1 = FIFO write/read special function is enabled

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
17	RXFIFO_AUTO_FULL_CTRL	R/W	0x0	Rx FIFO Auto Full Control After this field is set to 1 and the SPI/I2S is operating in master mode, the SS_FSM returns to IDLE state and stops the SS_SCLK. When Rx FIFO is full, the SS_FSM continues transferring data after the Rx FIFO is not full. This field is used to avoid an Rx FIFO overrun issue. 1= Enable Rx FIFO auto full control 0= Disable Rx FIFO auto full control
16	FPCKE	R/W	0x0	FIFO Packing Enable 0 = FIFO packing mode disabled 1 = FIFO packing mode enabled
15:14	TXFIFO_WR_ENDIAN	R/W	0x0	apb_pwdata Write to Tx FIFO Endian 0x0 = txfifo_wdata[31:0] = apb_pwdata[31:0] 0x1 = fifo_wdata[31:0] = {apb_pwdata[15:0], apb_pwdata[31:16]} 0x2 = txfifo_wdata[31:0] = {apb_pwdata[7:0], apb_pwdata[15:8], apb_pwdata[23:16], apb_pwdata[31:24]} 0x3 = txfifo_wdata[31:0] = {apb_pwdata[23:16], apb_pwdata[31:24], apb_pwdata[7:0], apb_pwdata[15:8]}
13:12	RXFIFO_RD_ENDIAN	R/W	0x0	apb_prdata Read from Rx FIFO Endian 0x0 = apb_prdata[31:0] = rxfifo_wdata[31:0] 0x1 = apb_prdata[31:0] = {rxfifo_wdata[15:0], rxfifo_wdata[31:16]} 0x2 = apb_prdata[31:0]= {rxfifo_wdata[7:0], rxfifo_wdata[15:8], rxfifo_wdata[23:16], rxfifo_wdata[31:24]} 0x3 = apb_prdata[31:0]= {rxfifo_wdata[23:16], rxfifo_wdata[31:24], rxfifo_wdata[7:0], rxfifo_wdata[15:8]}
11	RSRE	R/W	0x0	Receive Service Request Enable 0 = DMA service request is disabled 1 = DMA service request is enabled
10	TSRE	R/W	0x0	Transmit Service Request Enable 0 = DMA service request is disabled 1 = DMA service request is enabled
9:5	RFT	R/W	0x0	RXFIFO Trigger Threshold This field sets the threshold level at which RXFIFO asserts interrupt. The level should be set to the preferred threshold value minus 1.
4:0	TFT	R/W	0x0	TXFIFO Trigger Threshold This field sets the threshold level at which TXFIFO asserts interrupt.

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
				The level should be set to the preferred threshold value minus 1.

16.2.4.3 SSINTEN REGISTER

SPI/I2S interrupt enable register.

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:7	RSVD	R	0	Reserved for future use
6	EBCEI	R/W	0x0	Enable Bit Count Error Interrupt 0 = Interrupt due to a bit count error is disabled 1 = Interrupt due to a bit count error is enabled
5	TIM	R/W	0x1	Transmit FIFO Underrun Interrupt Mask 0 = TUR events generate an interrupt 1 = TUR events do NOT generate an interrupt
4	RIM	R/W	0x1	Receive FIFO Overrun Interrupt Mask 0 = ROR events generate an interrupt 1 = ROR events do NOT generate an interrupt
3	TIE	R/W	0x0	Transmit FIFO Interrupt Enable 0 = TXFIFO threshold-level-reached interrupt is disabled 1 = TXFIFO threshold-level-reached interrupt is enabled
2	RIE	R/W	0x0	Receive FIFO Interrupt Enable 0 = RXFIFO threshold-level-reached interrupt is disabled 1 = RXFIFO threshold-level-reached interrupt is enabled
1	TINTE	R/W	0x0	Receiver Time-out Interrupt Enable 0 = Receiver time-out interrupt is disabled 1 = Receiver time-out interrupt is enabled
0	PINTE	R/W	0x0	Peripheral Trailing Byte Interrupt Enable 0 = Peripheral trailing byte interrupt is disabled 1 = Peripheral trailing byte interrupt is enabled

16.2.4.4 SSTO REGISTER

SPI/I2S time out register. These registers specify the timeout (TIMEOUT) value used to signal a period of inactivity within the RXFIFO. When a timeout occurs, the SSSR[TINT] field is set. When the TIMEOUT value is set to 0x000000, no timeout occurs and the SSSR[TINT] field is not set. The TIMEOUT interval is given by the calculation in the TIMEOUT Interval Equation.

TimeOut Interval = SSTO [TIMEOUT] / APB Clock Frequency, APB Clock Frequency = 25.6 MHz OR 51.2MHz.

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	R	0	Reserved for future use
23:0	TIMEOUT	R/W	0x0	Timeout Value TIMEOUT value is the value that defines the time-out interval. The time-out interval is given by the equation shown in the TIMEOUT Interval Equation.

16.2.4.5 SSDATR REGISTER

SPI/I2S data register.

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	DATA	R/W	0x0	DATA This field is used for data to be written to the TXFIFO read from the RXFIFO.

16.2.4.6 SSSR REGISTER

SPI/I2S status register.

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	RSVD	R	0	Reserved for future use
23	OSS	R	0x0	Odd Sample Status 0 = RxFIFO entry has two samples 1 = RxFIFO entry has one sample **Note.** This bit needs to be looked at only when FIFO Packing is enabled (SSFCR[FPCKE] field is set). Otherwise, this bit is zero. When SPI/I2S port is in Packed mode and the CPU is used instead of DMA to read the RxFIFO, the CPU should make sure that SSSR[RNE] = 1 AND this field = 0 before it attempts to read the RxFIFO.
22	TX_OSS	R	0x0	TX FIFO Odd Sample Status 0 = TxFIFO entry has an even number of samples 1 = TxFIFO entry has an odd number of samples **Note.** This bit needs to be read only when FIFO Packing is enabled (SSFCR[FPCKE] field is set). Otherwise, this bit is zero.
21	BCE	R/W1C	0x0	Bit Count Error

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
				0 = The SPI/I2S port has not experienced a bit count error 1 = The SS_FRM signal was asserted when the bit counter was not zero
20	ROR	R/W1C	0x0	Receive FIFO Overrun 0 = RXFIFO has not experienced an overrun 1 = Attempted data write to full RXFIFO, causes an interrupt request
19:15	RFL	R	0x1F	Receive FIFO Level This field is the number of entries minus one in RXFIFO. When the value 0x1F is read, the RXFIFO is either empty or full, and software should read the SSSR[RNE] field.
14	RNE	R	0x0	Receive FIFO Not Empty 0 = RXFIFO is empty 1 = RXFIFO is not empty
13	RFS	R	0x0	Receive FIFO Service Request 0 = RXFIFO level is at or below RFT threshold (RFT) or SPI/I2S port is disabled 1 = RXFIFO level exceeds RFT threshold (RFT), causes an interrupt request
12	TUR	R/W1C	0x0	Transmit FIFO Underrun 0 = The TXFIFO has not experienced an underrun 1 = A read from the TXFIFO was attempted when the TXFIFO was empty, causing an interrupt if it is enabled
11:7	TFL	R	0x0	Transmit FIFO Level This field is the number of entries in TXFIFO. When the value 0x0 is read, the TXFIFO is either empty or full, and software should read the SSSR[TNF] field.
6	TNF	R	0x1	Transmit FIFO Not Full 0 = TXFIFO is full 1 = TXFIFO is not full
5	TFS	R	0x0	Transmit FIFO Service Request 0 = TX FIFO level exceeds the TFT threshold (TFT + 1) or SPI/I2S port disabled 1 = TXFIFO level is at or below TFT threshold (TFT + 1), causes an interrupt request
4	EOC	R/W1C	0x0	End Of Chain 0 = DMA has not signaled an end of chain condition 1 = DMA has signaled an end of chain condition
3	TINT	R/W1C	0x0	Receiver Time-out Interrupt 0 = No receiver time-out is pending 1 = Receiver time-out pending, causes an interrupt request
2	PINT	R/W1C	0x0	Peripheral Trailing Byte Interrupt

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
				0 = No peripheral trailing byte interrupt is pending 1 = Peripheral trailing byte interrupt is pending
1	CSS	R	0x0	Clock Synchronization Status 0 = The SPI/I2S port is ready for slave clock operations 1 = The SPI/I2S port is currently busy synchronizing slave mode signals
0	BSY	R	0x0	SPI/I2S Busy 0 = SPI/I2S port is idle or disabled 1 = SPI/I2S port is currently transmitting or receiving framed data

16.2.4.7 SSPSP REGISTER

SPI/I2S programmable serial protocol control register.

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:30	RSVD	R	0	Reserved for future use
29:27	EDMYSTOP	R/W	0x0	Extended Dummy Stop Most-significant bits of the dummy stop delay.Do not used in PSP Network mode.
26:25	DMYSTOP	R/W	0x0	Dummy Stop Least-significant bits of the dummy stop delay Programmed value of SSPSP[EDMYSTOP] + this field specifies the number (0-31) of active clocks (SS_SCLK) that follow the end of the transmitted data.Do not used in PSP Network mode.
24:23	EDMYSTRT	R/W	0x0	Extended Dummy Start Most-significant bits of the dummy start delay.Do not used in PSP Network mode.
22:21	DMYSTRT	R/W	0x0	Dummy Start Least-significant bits of the dummy start delay Programmed value of this field specifies the number (0-15) of active clocks (SS_SCLK) between the end of start delay and when the most-significant bit of transmit/receive data is driven. Do not used in PSP Network mode.
20:18	STRTDLY	R/W	0x0	Start Delay Programmed value specifies the number (0-7) of non-active clocks (SS_SCLK) that define the duration of idle time. Do not used in PSP Network mode.
17:12	SFRMWDTH	R/W	0x0	Serial Frame Width Least-significant bits of the serial frame width Programmed value of this field specifies the frame width from 0x00 (one SS_SCLK cycle) to 0x3F (63 SS_SCLK cycles).

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
11:5	SFRMDLY	R/W	0x0	Serial Frame Delay Programmed value specifies the number (0 -127) of active one-half clocks (SS_SCLK) asserted from the most-significant bit of TX (output) or RX (input) being driven to SS_FRM.Do not used in PSP Network mode.
4	SFRMP	R/W	0x0	Serial Frame Polarity 0 = SS_FRM is active low (0x0) 1 = SS_FRM is active high (0x1)
3	FSRT	R/W	0x0	Frame Sync Relative Timing Bit 0 = Next frame is asserted after the end of the DMTSTOP timing 1 = Next frame is asserted with the LSB of the previous frame
2	ETDS	R/W	0x0	End Of Transfer Data State 0 = Low 1 = Last Value <Bit 0>
1:0	SCMODE	R/W	0x0	Serial Bit-rate Clock Mode 0x0 = Data Driven (Falling), Data Sampled (Rising), Idle State (Low) 0x1 = Data Driven (Rising), Data Sampled (Falling), Idle State (Low) 0x2 = Data Driven (Rising), Data Sampled (Falling), Idle State (High) 0x3 = Data Driven (Falling), Data Sampled (Rising), Idle State (High)

16.2.4.8 SSNWCR REGISTER

SPI/I2S network control register.

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:20	RSVD	R	0	Reserved for future use
19:12	RTSA	R/W	0x0	RX Time Slot Active, only used in network mode 0 = SPI/I2S port does not receive data in this time slot 1 = SPI/I2S port receives data in this time slot.
11:4	TTSA	R/W	0x0	TX Time Slot Active, only used in network mode 0 = SPI/I2S port does NOT transmit data in this time slot 1 = SPI/I2S port does transmit data in this time slot
3:1	FRDC	R/W	0x0	Frame Rate Divider Control Value of 0x0-0x7 specifies the number of time slots per frame when in network mode (the actual number of time slots is this field +1, so 1 to 8 time slots

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
				can be specified).
0	MOD	R/W	0x0	Mode 0 = Normal mode 1 = Network mode. When set this bit to 1, must make sure at same time SSCR[FRF]=0x3

16.2.4.9 SSNWS REGISTER

SPI/I2S network status register.

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	RSVD	R	0	Reserved for future use
3	NMBSY	R	0x0	Network Mode Busy 0 = SPI/I2S port is in network mode and no frame is currently active 1 = SPI/I2S port is in network mode and a frame is currently active
2:0	TSS	R	0x0	Time Slot Status Value indicates which time slot is currently active. Because of synchronization between the SPI/I2S port's SS_SCLK domain and an internal bus clock domain, the value in this field becomes stable between the beginning and end of the currently active time slot.

16.2.4.10 SSRWT REGISTER

SPI/I2S root control register.

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	RSVD	R	0	Reserved for future use
4	MASK_RWOT_LAST_SAMPLE	R/W	0x0	Mask last_sample_flag in RWOT Mode 1 = Mask 0 = Unmask
3	CLR_RWOT_CYCLE	R/W	0x0	Clear Internal rwot_counter This field clears the rwot_counter to 0. This field is self-cleared after SSCR[SSE] = 1. 1 = Clear rwot_counter

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
2	SET_RWOT_CYCLE	R/W	0x0	Set RWOT Cycle This field is used to set the value of the RWTC register to the internal rwot_counter. This field is self-cleared by after SSCR[SSE] = 1. 1 = Set rwot_counter
1	CYCLE_RWOT_EN	R/W	0x0	Enable RWOT Cycle Counter Mode 1 = Enable 0 = Disable
0	RWOT	R/W	0x0	Receive Without Transmit 0 = Transmit/receive mode 1 = Receive without transmit mode

16.2.4.11 SSRWTCC REGISTER

SPI/I2S root counter cycles match register.

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SSRWOTCCM	R/W	0x0	It's just total SS_SCLK Cycles The value of this register defines the total number of SS_SCLK cycles when SSP works in master and RWOT mode. When the rwot_counter matches this value, SSP returns to IDLE state and does not output SS_SCLK anymore.

16.2.4.12 SSRWTCV REGISTER

SPI/I2S root counter value write for read request register.

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SSRWOTCVWR	R/W	0x0	This register prevents the risk of instability on rwot_counter value reading, it's only valid after SPI/I2S has been enabled Write 0 = No effect Write 1 = Capture value of rwot_counter Read: Returns the captured value of rwot_counter

16.3 UART

16.3.1 Introduction

The K1 has 10 UARTs (UART 0-9). The UARTs use the same programming model.

Each port contains an UART, a slow serial infrared transmit encoder and a Receive decoder conforming to the IrDA serial infrared specification.

Each UART performs serial-to-parallel conversion on data characters received from a peripheral device or a modem and parallel-to-serial conversion on data characters received from K1.

Software can read a complete UART status for the Line Status Register. Status information includes the type and condition of transfer operations and error conditions (parity, overrun, framing, or break interrupt) associated with the UART.

Each serial port operates in either FIFO or non-FIFO mode. In FIFO mode, a 64-byte Transmit FIFO holds data from K1 until it is transmitted on the serial link; a 64-byte Receive FIFO buffers data from the serial link until it is read by K1. In non-FIFO mode, the Transmit and Receive FIFOs are bypassed, and the Transmit Holding Register and Receive Buffer Register are used instead.

Each UART includes a programmable baud-rate generator that can divide the input clock by any value from 1 to (2^16 – 1), which produces a 16X clock that can be used to drive the internal Transmit and Receive logic. The software can program interrupts to meet its requirements, which minimizes the number of computations required to handle the communications link. Each UART operates in an environment that is either controlled by software and can be polled or is interrupt driven.

All 10 UARTs support the 16550A and 167502 functions, but support slightly different features as described in the following sections.

The supported baud rates of each UART are tabled below.

UART	Support Baud Rates												
	9600	19.2	38.4	57.6	115.2	230	460	921	1	1.5	1.8	3	3.6
	Hz	kHz							MHz				
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

16.3.2 Features

The serial ports are controlled via direct-memory access (DMA) or programmed I/O. The UARTs share the following features:

- Support for up to 10 UART interfaces
- Compatible with the 16550A and 16750 UART standards
- Support for adding and deleting standard asynchronous communication bits (start, stop and parity) in the serial data stream
- Independent control of transmission, reception, line status, data-set interrupts

- Modem control functions (CTSn and RTSn for both UART2 and UART3)
- Auto-flow capability for data I/O management without generating interrupts, where
  - RTSn (output) is controlled by the UART receive FIFO
  - CTSn (input) is from UART modem transmission controls
- Programmable serial interface with configurable options as follow:
  - 7-bit or 8-bit character length
  - Even, odd or no parity detection
  - 1 stop-bit generation
  - Baud rate generation up to 3.6Mbps for the 4 Fast UARTs
  - False start-bit detection
- 64-byte transmit FIFO
- 64-byte receive FIFO
- Support for complete status reporting
- Support for generating and detecting line breaks
- Support for internal diagnostics including:
  - Loopback control for fault isolation in communications link
  - Break, parity and framing error simulation
- Fully prioritized interrupt system
- Support for separated DMA requests for both transmit and receive data services
- Serial infrared asynchronous interface compliant with the Infrared Data Association (IrDA) specification

The UARTs are functionally compatible with the 16550A and 16750 industry standards. Each UART supports most of the 16550A and 16750 functions as well as the following features:

- DMA requests for Transmit and Receive data services
- Serial infrared asynchronous interface
- Non-Return to Zero (NRZ) encoding/decoding function
- 64 byte Transmit/Receive FIFO buffers
- Programmable Receive FIFO trigger threshold
- Auto baud-rate detection
- Auto flow

16.3.3 Functional Description

16.3.3.1 Signal Description

Each external signal that is connected to a UART module and how these pins function as modem control lines are tabled below.

Name	Type	Description
RXD	Input	**Serial Input** Serial data input to the Receive Shift register. In Infrared mode, it is connected to the infrared receiver input.
TXD	Output	**Serial Output** Serial data output to the communications-link peripheral, modem, or data set. The TXD signal is set to the logic 1 state upon a reset operation. It is connected to the output of the infrared transmitter in Infrared modeAuto-flow mode.
CTSn	Input	**Clear to Send** When asserted, indicates that the modem or data set is ready to exchange data. The CTSn signal is a modem status input, and its condition can be tested by reading the <CTS> field in the Modem Status Register. The <CTS> field is the complement of the CTSn signal. The <Delta Clear to Send> field in the Modem Status Register indicates whether the CTSn input has changed state since the last time the Modem Status Register was read. CTSn has no effect on the transmitter. When the <CTS> field changes state and the modem-status interrupt is enabled, an interrupt is generated. **Non-Auto-flow mode**: When not in Auto-flow mode, the <CTS> field indicates the state of CTSn. The <Delta Clear to Send> field indicates whether the CTSn input has changed state since the previous reading of MSR. CTSn has no effect on the transmitter. The user can program the UART to interrupt the K1 when DCTS changes state. Software can then stall the outgoing data stream by starving the Transmit FIFO or disabling the UART with the Interrupt Enable Register. **Note.** If UART transmission is stalled by disabling the UART, no Modem Status Register interrupt is received when CTSn re-asserts because disabling the UART also disables interrupts. To get around this issue, use either auto-CTS in Auto-flow mode or program the CTSn GPIO pin to interrupt. **Auto-flow mode**: In this mode, the UART Transmit circuit checks the state of CTSn before transmitting each byte. No data is transmitted when CTSn is high.
DSRn	Input	**Data Set Ready** When asserted, it indicates that the modem or data set is ready to establish a communications link with a UART. The DSRn signal is a modem-status input and its condition can be tested by reading the <Data Set Ready> field in the Modem Status Register, which is the complement of DSRn. The <Delta Data Set Ready> field in the Modem Status Register indicates whether the DSRn input has changed state since the Modem Status Register was last read. When the <Data Set Ready> changes state, an interrupt is generated if the modem-status interrupt is enabled.
DCDn	Input	**Data Carrier Detect** When asserted, indicates that the data carrier has been detected by the modem or data set. The DCDn signal is a modem-status input and its condition can be tested by reading the <Data Carrier Detect> field in the Modem Status Register, which is the complement of the DCDn signal. The <Delta Data Carrier Detect> field in the Modem Status Register indicates

Name	Type	Description
		whether the DCDn input has changed state since the previous reading of the Modem Status Register. DCDn has no effect on the receiver. An interrupt is generated when the <Data Carrier Detect> field changes state and the modem-status interrupt is enabled.
Rin	Input	**Ring Indicator** When asserted, indicates that the modem or data set has received a telephone ringing signal. The RIn signal is a modem-status input and its condition can be tested by reading the <Ring Indicator> field in the Modem Status Register, which is the complement of the RIn signal. The <Trailing Edge Ring Indicator> field in the Modem Status Register indicates whether the RIn input has changed from low to high since the Modem Status Register was last read. An interrupt is generated when the RI bit of the Modem Status Register changes from a high to low state and the modem-status interrupt is enabled.
DTRn	Output	**Data Terminal Ready** When asserted, signals the modem or the data set that the UART is ready to establish a communications link. To assert the DTRn output (active low), set the <Data Terminal Ready> field in the Modem Control Register, which is the complement of the output signal. A reset operation de-asserts this signal (high). Loop-mode operation holds DTRn de-asserted.
RTSn	Output	**Request To Send** When asserted, signals the modem or the data set that the UART is ready to exchange data. To assert the RTSn output (active low), set the <Request to Send> field in the Modem Control Register, which is the complement of the output signal. A reset operation de-asserts this signal (high). Loop-mode operation holds RTSn de-asserted. **Non-Auto-flow mode**: To assert the RTSn output (active low), set <Request to Send>. **Auto-flow mode**: RTSn is asserted automatically by the auto-flow circuitry when the Receive buffer exceeds its programmed trigger threshold. It is de-asserted when enough bytes are removed from the buffer to lower the data level back to the trigger threshold.

The pins transmit digital CMOS-level signals are connected to K1 through GPIOs (refer to **Section 3.7**).

16.3.3.2 Operation

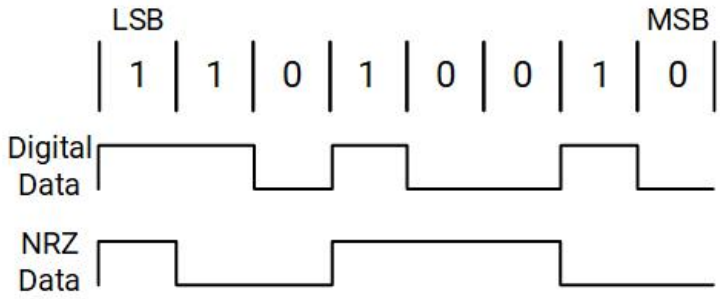
The Receive-data sample-counter frequency is 16 times the value of the bit frequency. The 16X clock is created by the baud-rate generator. Each bit is sampled three times in the middle. Other bits are optional and can be programmed by software.

Each data frame is between 9 and 11 bits long, depending on the size ofthe data programmed, whether parity is enabled A data frame begins by transmitting a start bit that is represented by a high-to-low transition. The start bit is followed by 8 bits of data that begin with the Least Significant bit (LSb). The data bits are followed by an optional parity bit. The parity bit is set if: even parity is enabled and the data byte has an odd number of ones or if odd parity is enabled and the data byte has an even number of ones. The data frame ends with 1 stop bit. The stop bit is represented by 1 successive bit period of logic one.

Each UART has 2 FIFOs: 1 Transmit and 1 Receive. The Transmit FIFO is 64 bytes deep and 8 bits wide. The Receive FIFO is 64 bytes deep and 11 bits wide. Three bits are used for tracking errors.

The UART can use NRZ coding to represent individual bitvalues. To enable NRZ coding, set the <NRZ Coding Enable> field in the Interrupt Enable Register. A bit value of 1 is represented by a line transition, and 0 is represented by no line transition.

The data byte 8'b0100_1011 in NRZ coding is depicted below (the LSB in the byte is transmitted first).



16.3.3.3 Reset

The UARTs are disabled on reset. To enable a UART, software must program the Multi-function Pin registers, then set the <UART Unit Enable> field in the Interrupt Enable Register. When the UART is enabled, the receiver waits for a frame start bit, and the transmitter sends data if it is available in the Transmit Holding Register. Transmit data can be written to the Transmit Holding Register before the UARTunit is enabled. In FIFO mode, data is transmitted from the FIFO to the Transmit Holding Register before it goes to the pin.

When the UART unit is disabled, the transmitter or receiver finishes the current byte and stops transmitting or receiving more data. Data in the FIFO is not cleared and transmission resumes when the UART is enabled.

16.3.3.4 FIFO Operation

Each UART has a Transmit FIFO and a Receive FIFO, with each FIFO holding 64 characters of data.There are 2 methods for moving data into or out of the FIFOs: DMA, Program I/O.

In DMA mode, interrupts are used to control the data flow; whereas, in Program I/O mode, polling is used.

In DMA mode, software must set the DMA stop interrupt on the last Descriptor in the chain to avoid errors.

16.3.3.4.1 FIFO Interrupt Mode: Receive Interrupt

For a Receive interrupt to occur, the Receive FIFO and Receive interrupts mustbe enabled. The <Interrupt Source Encoded> field in the Interrupt Identification Register changes to show that Receive data is available when the FIFO reaches its trigger threshold. The <Interrupt Source Encoded> field changes to show the next waiting interrupt when the FIFO drops below the trigger threshold. A change in the <Interrupt Source Encoded> field triggers an interrupt to the core.

Software reads the <Interrupt Source Encoded> field to determine the cause of the interrupt.

The Receive-line-status interrupt (Interrupt Identification Register = 0xC6) has the highest priority; the received-data-available interrupt (Interrupt Identification Register = 0xC4) is lower. The line-status interrupt occurs only when the character at the front of the FIFO has errors.

The <Data Ready> field in the Line Status Register is set when a character is transferred from the Shift register to the Receive FIFO. <Data Ready> is cleared when the FIFO is empty.

16.3.3.4.2 FIFO Interrupt Mode: Character Timeout Interrupt

A character(receiver) timeout interrupt occurs when the Receive FIFO and Receive timeout interrupt are enabled and all of the following conditions exist:

- At least 1 character is in the FIFO.
- The most recently received character was received more than 4 continuous character times ago.
- The most recent FIFO read was performed more than 4 continuous character times ago.

After the k1 reads 1 character from the Receive FIFO or a new start bit is received, the timeout interrupt is cleared, and the timeout is reset. If a timeout interrupt has not occurred, the timeout is reset when a new character is received or the k1 reads the Receive FIFO.

16.3.3.4.3 FIFO Interrupt Mode: Transmit interrupt

Transmit interrupts can occur only when the Transmit FIFO and Transmit interrupt are enabled. The Transmit data-request interrupt occurs when the Transmit FIFO is at least half empty. The interrupt is cleared when the Transmit Holding Register is written or the Interrupt Identification Register is read.

16.3.3.4.4 FIFO Interrupt Mode: Removing Trailing Bytes

The k1 must remove trailing bytes when not in DMA mode or when the DMA mode bit (<Trailing Bytes> field in the FIFO Control Register) is not set. The presence of trailing bytes is signaled by the assertion of a character timeout interrupt. When servicing a character timeout interrupt, the k1 uses the following procedure:

- Read the Line Status Register and check for errors.
- Disable the receiver timeout interrupt via <Receiver Time-out Interrupt Enable> field in the Interrupt Enable Register.
- Read data from the UART FIFO.
- Read the Line Status Register, check for errors, and LOOP back to the previous step. If the <Data Ready> field is SET, go to the next step.
- No more data in FIFO: Re-enable RTO interrupt via the <Modem Interrupt Enable> field in the Interrupt Enable Register.
- Done

16.3.3.4.5 FIFO Polled Mode Operation

When the FIFOs are enabled, clearing both the <DMA Requests Enable> field and bits [4:0] in the Interrupt Enable Register places the port in FIFO polled operating mode. The receiver and the transmitter are controlled separately. Either one or both can be in polled mode. In polled mode, software checks receiver and transmitter status via the Line Status Register. K1 polls the following bits for the Receive and Transmit data service:

- Receive Data Service – K1 checks the <Data Ready> field, which is set when 1 or more bytes remain in the Receive FIFO or Receive Buffer Register.
- Transmit Data Service – K1 checks the <Transmit Data Request> field in the Line Status Register, which is set when the transmitter needs data.

K1 can also check the <Transmitter Empty> field in the Line Status Register, which is set when the Transmit FIFO.

16.3.3.4.6 FIFO DMA Mode Operation

The UART has 2 DMA requests: 1 for Transmit data service and 1 for Receive data service. DMA requests are generated in FIFO mode only. The requests are activated by setting the <DMA Requests Enable>field in the Interrupt Enable Register.

- Data Transmitter Data Service – when <DMA Requests Enable> is set, if the Transmit FIFO is absolutely less than half full, the Transmit-DMA request is generated. The DMA Controller (DMAC) then writes data to the FIFO. For each DMA request, the DMAC can send 8, 16, or 32 bytes of data to the FIFO. The UART FIFO accepts partial-word or full-word transfers of 1, 2, 3, or 4 consecutive bytes from the DMAC or Program I/O. The actual number of bytes to be transmitted is programmed in the DMAC.
- Data Receiver Data Servic – when <DMA Requests Enable> is set, the Receive-DMA request is generated when the Receive FIFO reaches its trigger threshold with no errors in its entries. The DMAC then reads data from the FIFO. For each DMA request, the DMAC can read 8, 16, or 32 bytes of data from the FIFO. When in 32-bit peripheral bus mode, the DMAC always attempts to read 4 bytes of data per transfer. Where less than 4 bytes are being transferred, the valid bytes are indicated by a data-valid bus shared between the UART and the DMAC. The UART can send 1, 2, 3, or 4 bytes of data per bus transaction. The actual number of bytes to be read is programmed in the DMAC along with the bus width.

16.3.3.4.7 DMA Receive Programming Errors

If the DMA channel stops prematurely due to the end of a Descriptor chain or other error, the K1 must be notified since the DMAC can no longer service the UARTs FIFOs.If this occurs, the K1 must correct the situation by programming another Descriptor or by servicing the FIFOs via interrupt or polling mode, as described above. There are 2 methods for notifying the K1 of a stopped DMA channel:

- Program the DMAC to interrupt on the event of a stopped channel by setting DCSR[StopIrqEn].
- For the Receive channel, the UART interrupts with an end-of-Descriptor chain (EOC) interrupt if <Trailing Bytes> is set, such that the UART makes a DMA request to remove trailing bytes (see Removing Trailing Bytes In DMA Mode). Using the UART interrupt for the Receive channel is preferable to the DMA DCSR interrupt because extra logic exists to ensure that the UART EOC interrupt asserts only when necessary. For example, a UART EOC interrupt does not assert if the UART has completed the reception of its message (indicated by the character timeout timer) and the Receive FIFO is empty. The <DMA End of Descriptor Chain> field in the Interrupt Identification Register interrupt does not assert if <Trailing Bytes> is cleared.

16.3.3.4.8 DMA Error Handling

If an error occurs while in DMA mode, the Receive-DMA requests are disabled and the error interrupt,<Interrupt Source Encoded>, is generated.

The K1 must now read out the error bytes through Programmed Input/Output (PIO). After all errors have been removed from the FIFO, the Receive DMA requests are once again enabled by the UART.

If an error occurs when the Receive FIFO trigger threshold has been reached such that a Receive DMA request is set, software must wait for the DMA to finish the transfer before reading out the error bytes through PIO. Otherwise, FIFO underflow could occur.

16.3.3.4.9 Removing Trailing Bytes In DMA Mode

When the number of entries in the Receive FIFO is less than its trigger threshold and no additional data is received, the remaining bytes are called trailing bytes. Set <Trailing Bytes> to program the UART to make a DMA request to remove the trailing bytes. Setting <Trailing Bytes> also enables the <DMA End of Descriptor Chain> interrupt.

A request is issued automatically for the remaining number of bytes left in the Receive buffer when the DMAC is removing trailing bytes. The DMAC then empties the contents of the Receive buffer unless the DMA reaches the end of its Descriptor chain. If the DMA reaches the end of the Descriptor chain while removing trailing bytes, the K1 is forced to take over because the DMAC can no longer service the UART request until a new chain is linked. In this situation, the UART sets<DMA End of Descriptor Chain> ifdata exists in the Receive FIFO, and if <Receiver Time-out Interrupt Enable> is set, it also sets the <Time Out Detected> field in the Interrupt Identification Register. The remaining bytes must then be removed using Processor I/O mode as described in FIFO Interrupt Mode Operation.

16.3.3.4.10 False EOR Due to Character Time-out Expiration

It is possible for a false EOR to be asserted by the UART in the middle of receiving a message if a pause in the remote data transmissions is long enough to cause the timeout counter to expire. This situation causes an EOR to be sent to the DMAC if in DMA mode. If this situation occurs, the EOR is applied to the last byte of data in the FIFO when the DMA responds to the EOR request. The EOR is not applied to the last byte in the FIFO at the time of the character timeout. Therefore, if remote transmission resumes before the DMA responds to the EOR request, the EOR flag is applied to the new data that entered the FIFO and not to the last byte in the FIFO at the time of the character timeout.

16.3.3.4.11 EOR Must be Serviced Prior to Transmission of New Message

A caveat to this behavior could be encountered under legitimate EOR situations: for example, if Message A ends with 3 bytes in the FIFO, an EOR request is made to the DMAC to remove these bytes. If transmission of a new Message B resumes before the DMAC responds to the EOR request of Message A, the EOR could be applied to the first byte of Message B if this byte is written into the FIFO before the DMAC responds to message A's EOR request. Although this situation could occur, it would be considered a programming error because the higher communication protocol must prevent Message B transmission until the local receiver acknowledges the receipt of Message A. The exception to this scenario would be if enough new bytes enter the FIFO to push the FIFO level to its programmed data threshold. If this situation occurs, the request is treated as a normal service request and no EOR flag is asserted to the DMAC

16.3.3.5 Auto-Flow Control

Auto-flow control uses the Clear to Send (CTSn) and Request to Send (RTSn) signals to automatically control the flow of data between the UART and external modem. When auto-flow is enabled, the remote device is not

allowed to send data unless the UART asserts (that is, sets to 0) RTSn. If the UART de-asserts (that is, sets to 1) RTSn while the remote device is sending data, the remote device is allowed to send 1 additional byte after RTSn is de-asserted. An overflow could occur if the remote device violates this rule. Likewise, the UART is not allowed to transmit data unless the remote device asserts CTSn (that is, sets to 0). ASR recommends using this feature because it increases system efficiency and eliminates the possibility of a Receive-FIFO-overflow errordue to long interrupt latency.

Auto-flow mode can be used in 2 ways:

- full auto-flow, automating both CTSn and RTSn
- half auto-flow, automating only CTSn

Set the <Request to Send> and <Auto-flow Control Enable> fields in the Modem Control Register to enable full auto-flow. Set <Auto-flow Control Enable> and clear <Request to Send> to enable auto-CTSn-only mode.

16.3.3.5.1 RTSn (UART Output)

When in full Auto-flow mode, RTSn is asserted (0) when the UART FIFO is ready to receive data from the remote transmitter. This scenario occurs when the amount of data in the Receive FIFO is below the programmable trigger threshold value. RTSn is de-asserted (set to 1) when the amount of data in the Receive FIFO reaches the programmable trigger threshold. It is asserted again when enough bytes are removed from the FIFO to lower the data level below the trigger threshold.

16.3.3.5.2 CTSn (UART Input)

When in full- or half-Auto-flow mode, CTSn is asserted (set to 0) by the remote receiver when the receiver isready to receive data from the UART. The UART checks CTSn before sending the next byte of data and does not transmit the byte until CTSn is low. The transmitter completes this byte if CTSn goes high while the transfer of a byte is in progress.

If UART transmission is stalled by disabling the UART, none of the interrupts in the Modem Status Register indicate an interrupt when CTSn re-asserts because disabling the UART also disables interrupts. ASR recommends using auto-CTS in Auto-flow mode.

16.3.3.6 Auto-Baud-Rate Detection

Each UART supports auto-baud-rate detection. When enabled, the UART counts the number of clock cycles within the start-bit pulse. This number is then written into the Auto-Baud Count Register (as described in **K1 Registers**) and is used to calculate the baud rate. When the Auto-Baud Count Register is written, an auto-baud-lock interrupt is generated (if enabled), and the UART automatically programs the Divisor Latch Registers with the appropriate baud rate. If preferred, K1 can read the Auto-Baud Count Register and use this information to program the Divisor Latch Low Byte Register and Divisor Latch High Byte Register with a baud rate calculated by K1. After the baud rate has been programmed, the K1 verifies that the predetermined characters (usually AT or at) are being received correctly.

If the UART is to program the Divisor Latch Registers, software can use either of 2 methods for auto-baud calculation:

- Table-based method
- Formula-based

The method is selected via the <ABT> field in the Auto-Baud Control Register. The baud rates that are seen in most commercial electronics, which are referred to as “common,” include:

- **Formula-based method**
  Any baud rate can be programmed by the UART. This method works well for higher baud rates, but it could fail below 28.8 kbps if the remote transmitter’s actual baud rate differs by more than 1 percent of its target.

- **Table-based method**
  It is more immune to such errors, because the table rejects uncommon baud rates and rounds to the common ones. The table method allows any baud rate defined by the formula in **Section 16.3.3.8**. Programmable Baud-Rate Generator above 28.8 kbps. Below 28.8 kbps, the only baud rates that can be programmed by the UART are 19200, 14400, 9600, 4800, 1200, and 300 baud.

When the baud rate is detected, the auto-baud circuitry disables itself by clearing the <ABE> field in the Auto-Baud Count Register. To re-enable auto-baud detection, set the <ABE> field again.

Note. Changing the baud rate is not permitted when actively transmitting or receiving data. Auto-baud-rate detection is not supported in IrDA (serial infrared) mode

16.3.3.7 32-Bit Peripheral Bus

Each UART supports an 8- (default) or 32-bit peripheral bus. If a 32-bit bus is preferred, set the <32-Bit Peripheral Bus> field in the FIFO Control Register. The bytes are written in Little Endian format (7:0) with byte 3 (the most recent byte) starting at bit [31], byte 2 starting at bit [23], and so on.

8-bit mode—only the least significant byte contains valid data on the peripheral bus. The upper 24 bits are ignored.

32-bit mode—the UART can read or write partial words of 1, 2, 3, or 4 continuous bytes from the peripheral bus. The method in which the valid bytes of data are determined differs depending on whether the transaction is being handled by the DMAC or PIO.

DMA—the DMAC can read or write 1,2, 3, or 4 continuous bytes per word. The number of valid bytes available per word is determined internally between the DMAC and the UART.

PIO—the K1 is restricted to reading or writing 1, 2, or 4 bytes per word. When reading, the K1 must read the Receive FIFO Occupancy Register to retrieve the number of bytes available in the Receive buffer. If the number ofbytes available is 4 or greater, the K1 can request any number of bytes per word (except 3). If the number is less than 4, software must request the proper number of bytes. When 3 bytes are remaining, software must requesteither 2 bytes followed by 1 byte or 1 byte followed by 2 bytes. The UART retrieves unusable data for the non-valid bytes if the K1 reads more than the number of bytes available in the Receive buffer. The Receive FIFO counters do not increase.

Note. The Receive and Transmit FIFOs must be enabled when in 32-bit mode.

16.3.3.8 Programmable Baud-Rate Generator

Each UART contains a programmable baud-rate generator that can take a fixed-input clock and divide it down to generate the preferred baud rate. The baud rate is calculated by taking the 14.7456 MHz fixed-input clock or the 57.60 MHz clock (in high speed mode and dividing it by the Divisor Latch Low Register. For high speed mode, a divisor of 1 or 2 is required.

The baud-rate generator output frequency is 16 times the baud rate. Two 8-bit Divisor Latch Registers (Divisor Latch Low Register and Divisor Latch High Register as described in the **K1**_ Registers_) store the divisor in a 16-

bit binary format. Load these divisor latches during initialization to ensure that the baud-rate generator operates properly. The 16X clock stops if each Divisor Latch register is loaded with 0x0.

The recommended baud rates based on divisor values (Divisor Latch High Byte Register / Divisor Latch Low Byte Register) is tabled below.

Required Baud Rate	Divisor	14.7456 MHz Actual Baud Rate	48 MHz Actual Baud Rate	57.60 MHz Actual Baud Rate
9600	96	9600	—	—
19200	48	19200	—	—
38400	24	38400	—	—
57600	16	57600	—	—
115200	8	115200	—	—
230400	4	230400	—	—
460800	2	460800	—	—
921600	1	921600	—	—
1000000	3	—	1000000	—
1500000	2	—	1500000	—
1842000	2	—	—	1954398
3000000	1	—	3000000	—
3686400	1	—	—	3908796

The divisor reset value is 0x0002. Changing the baud rate (writing to registers Divisor Latch Low Byte Register and Divisor Latch High Byte Register) is not permitted while actively transmitting or receiving data.

16.3.4 Register Description

Note.

- The UART_0 Register Base Address is 0xF0612000
- The UART_2~9 Register Base Address is 0xD4017000 ~ 0xD4017800, each address space of 256-Byte

16.3.4.1 Receive Buffer Register

In non-FIFO mode, this register holds the character(s) received by the UART Receive Shift Register. If this register is configured to use fewer than 8 bits, the bits are right-justified and the most significant bits (MSbs) are zeroed. Reading the register empties the register and clears the <Data Ready> field in the Line Status Register. This register latches the value of the data byte at the front of the FIFO in FIFO mode.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	BYTE_3	RO	0x0	Byte 3. This field is only valid in 32-bit peripheral bus mode.
23:16	BYTE_2	RO	0x0	Byte 2. This field is only valid in 32-bit peripheral bus mode.
15:8	BYTE_1	RO	0x0	Byte 1. This field is only valid in 32-bit peripheral bus mode.
7:0	BYTE_0	RO	0x0	Byte 0. This field is only valid in 32-bit peripheral bus mode.

16.3.4.2 Transmit Holding Register

This register holds the data byte(s) to be transmitted next in non-FIFO mode. When the Transmit Shift Register is emptied, the contents of this register are loaded into the Transmit Shift Register and the <Transmit Data Request> field in the Line Status Register is set. A write to Transmit Holding Register puts data at the top of the FIFO in FIFO mode. The data at the front of the FIFO is loaded into the Transmit Shift Register when the Transmit Shift Register is empty.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	BYTE_3	WO	0x0	Byte 3. This field is only valid in 32-bit peripheral bus mode.
23:16	BYTE_2	WO	0x0	Byte 2. This field is only valid in 32-bit peripheral bus mode.
15:8	BYTE_1	WO	0x0	Byte 1. This field is only valid in 32-bit peripheral bus mode.
7:0	BYTE_0	WO	0x0	Byte 1. This field is only valid in 32-bit peripheral bus mode.

16.3.4.3 Divisor Latch Low Byte Register

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7:0	DLL	RW	0x2	Divisor Latch Low. Low-byte compare value to generate baud rate.

16.3.4.4 Divisor Latch High Byte Register

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0x0	Reserved for future use.
7:0	DLH	RW	0xx	Divisor Latch High. High-byte compare value to generate baud rate.

16.3.4.5 Interrupt Enable Register

This register enables the 5 types of interrupts that set a value in the Interrupt Identification Register. Software must clear the appropriate bit in this register to disable an interrupt. Software can enable some interrupts by setting the appropriate bit.

The character timeout-indication interrupt is separated from the received data-available interrupt to ensure that the K1 and the DMA controller do not service the receive FIFO at the same time. When a character-timeout-indication interrupt occurs, the K1 must handle the data in the receive FIFO through programmed I/O.

An error interrupt is used when DMA requests are enabled. The interrupt is generated when the <FIFO Error Status> field in the Line Status Register is set because a receive DMA request is not generated when the receive FIFO has an error. The error interrupt tells the K1 to handle the data in the receive FIFO through programmed I/O. The error interrupt is enabled when DMA requests are enabled, and it can not be masked. Receiver line-status interrupts occur when the error is at the front of the FIFO.

When DMA requests are enabled and an interrupt occurs, software must first read the Line Status Register to see if an error interrupt exists, then checks the Interrupt Identification Register for the source of the interrupt. Software must read the Infrared Selected Register to determine the error condition if an interrupt occurs and the <FIFO Error Status> field in the Line Status Register is clear. DMA requests are automatically enabled when the last error byte is read from the FIFO. Software is not required to check for the error interrupt if DMA requests are disabled because an error interrupt occurs only when DMA requests are enabled.

The <FIFO Error Status> field is used to enable DMA requests. This register also contains the unit enable and NRZ coding enables control bits. Bits [7:4] are used differently from the standard 16550A register definition.

Software must not set the <DMA Requests Enable> field while the <Transmit Data Request Interrupt Enable> or <Receiver Data Available Interrupt Enable> fields are set to ensure that the DMA controller and programmed I/O do not access the same FIFO.

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7	DMAE	RW	0x0	DMA Requests Enable. 0 = DMA requests are disabled. 1 = DMA requests are enabled.
6	UUE	RW	0x0	UART Unit Enable. UART transmit and receive enable. Transmit data can be written to the Transmit Holding Register before the UART unit is enabled. When the UART unit is disabled, the transmitter or receiver finishes the current byte and

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
				stops transmitting or receiving more data. Data in the FIFO is not cleared and transmission resumes when the UART is enabled 0 = Unit is disabled. 1 = Unit is enabled.
5	NRZE	RW	0x0	NRZ Coding Enable. NRZ encoding/decoding is only used in UART mode, not in infrared mode. If the serial infrared receiver or transmitter is enabled, NRZ coding is disabled. 0 = NRZ coding disabled. 1 = NRZ coding enabled.
4	RTOIE	RW	0x0	Receiver Time-out Interrupt Enable. The source for this field is the <Time Out Detected> field in the Interrupt Identification Register. 0 = Receiver data time-out interrupt disabled. 1 = Receiver data time-out interrupt enabled.
3	MIE	RW	0x0	Modem Interrupt Enable. The source for this field is the <Interrupt Source Encoded> field in the Interrupt Identification Register. 0 = Modem status interrupt disabled. 1 = Modem status interrupt enabled.
2	RLSE	RW	0x0	Receiver Line Status Interrupt Enable. The source for this field is the <Interrupt Source Encoded> field in the Interrupt Identification Register. 0 = Receiver line status interrupt disabled. 1 = Receiver line status interrupt enabled.
1	TIE	RW	0x0	Transmit Data Request Interrupt Enable. The source for this field is the <Interrupt Source Encoded> field in the Interrupt Identification Register. 0 = Transmit FIFO data request interrupt disabled. 1 = Transmit FIFO data request interrupt enabled.
0	RAVIE	RW	0x0	Receiver Data Available Interrupt Enable. The source for this field is the <Interrupt Source Encoded> field in the Interrupt Identification Register. 0 = Receiver data available (trigger threshold reached) interrupt disabled. 1 = Receiver data available (trigger threshold reached) interrupt enabled.

16.3.4.6 Interrupt Identification Register

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	Reserved	RO	0x0	Reserved for future use.
8	EOR	RO	0x0	UART End of receive Status. 0x0= uart rx not end. 0x1= uart rx end.
7:6	FIFOES10	RO	0x0	FIFO Mode Enable Status. 0x0 = Non-FIFO mode is selected. 0x1 = Reserved. 0x2 = Reserved. 0x3 = FIFO mode is selected (<Transmit and Receive FIFO Enable> field in the FIFO Control Register = 1)
5	EOC	RO	0x0	DMA End of Descriptor Chain (see **Section 16.3.3.7**, 32-Bit Peripheral Bus). 0 = DMA has not signaled the end of its programmed descriptor chain. 1 = DMA has signaled the end of its programmed descriptor chain.
4	ABL	RO	0x0	Auto-baud Lock (see **Section 16.3.3.6**, Auto-Baud-Rate Detection). 0 = Auto-baud circuitry has not programmed Divisor Latch registers. 1 = Divisor Latch registers programmed by auto-baud circuitry.
3	TOD	RO	0x0	Time Out Detected (see **Section 16.3.3.4.2**, Character Timeout Interrupt). 0 = No time out interrupt is pending. 1 = Time out interrupt is pending (FIFO mode only).
2:1	IID10	RO	0x0	Interrupt Source Encoded. 0x0 = Modem Status (CTS, DSR, RI, DCD modem signals changed state). 0x1 = Transmit FIFO requests data. 0x2 = Received data available. 0x3 = Receive error (Overrun, parity, framing, break, FIFO error. See Modem Status Register at **Section 16.3.4.11**).
0	NIP	RO	0x1	Interrupt Pending. 0 = Interrupt is pending (active low). 1 = No interrupt is pending.

16.3.4.7 FIFO Control Register

This is a write-only register that is located at the same address as the Interrupt Identification Register, which is a read-only register. This register enables/disables the transmit/receive FIFOs, clears the transmit/receive FIFOs, and sets the receive FIFO trigger threshold.

The trigger level must be equal to the DMA burst length programmed in the DMA registers.

When the number of bytes in the receive FIFO equals the interrupt trigger level programmed into this field and the received-data-available interrupt is enabled (via the Interrupt Enable Register), an interrupt is generated and the appropriate bits are set in the Interrupt Identification Register. The receive DMA request is generated as well when trigger level is reached. The trigger level must be greater than or equal to the DMA burst size programmed in the DMA registers.

Offset: 0x8

Bits	Field	Type	Reset	Description
31:8	Reserved	RO	0x0	Reserved for future use.
7:6	ITL	WO	0x0	Interrupt Trigger Level (threshold) When the number of bytes in the receive FIFO equals the interrupt trigger threshold programmed into this field and the received-data-available interrupt is enabled via the Interrupt Enable Register, an interrupt is generated and appropriate bits are set in the Interrupt Identification Register. The receive DMA request is also generated when the trigger threshold is reached. 0x0 = 1 byte or more in FIFO causes interrupt (not valid in DMA mode). 0x1 = 8 bytes or more in FIFO causes interrupt and DMA request. 0x2 = 16 bytes or more in FIFO causes interrupt and DMA request. 0x3 = 32 bytes or more in FIFO causes interrupt and DMA request.
5	BUS	WO	0x0	32-Bit Peripheral Bus. 0 = 8-bit peripheral bus. 1 = 32-bit peripheral bus.
4	TRAIL	WO	0x0	Trailing Bytes. 0 = Trailing bytes are removed by the K1. 1 = Trailing bytes are removed by the DMAC.
3	TIL	WO	0x0	Transmitter Interrupt Level 0 = Interrupt/DMA request when FIFO is half empty 1 = Interrupt/DMA request when FIFO is empty
2	RESETTF	WO	0x0	Reset Transmit FIFO. When this field is set, all the bytes in the transmit FIFO are cleared. The <Transmit Data Request> field in the Line Status Register is set and the Interrupt Identification Register shows a transmitter requests data interrupt, if the <Transmit Data Request Interrupt

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
				Enable> field in the Interrupt Enable Register is set. The Transmit Shift Register is not cleared, and it completes the current transmission. 0 = Writing 0 has no effect. 1 = The transmit FIFO is cleared.
1	RESETRF	WO	0x0	Reset Receive FIFO. When this field is set, all the bytes in the receive FIFO are cleared. The <Data Ready> field in the Line Status Register is reset to 0. All the error bits in the FIFO and the <FIFO Error Status> field in the Line Status Register are cleared. Any error bits, OE, PE, FE or BI, that had been set in the Line Status Register are still set. The Receive Shift Register is not cleared. If the Interrupt Identification Register had been set to receive data available, it is cleared. 0 = No effect. 1 = The receive FIFO is cleared.
0	TRFIFOE	WO	0x0	Transmit and Receive FIFO Enable. This field enables/disables the transmit and receive FIFOs. When set, both FIFOs are enabled (FIFO mode). When clear, the FIFOs are both disabled (non-FIFO mode). Writing 0x0 to this field clears all bytes in both FIFOs. When changing from FIFO mode to non-FIFO mode and vice versa, data is cleared automatically from the FIFOs. This field must be set when other fields in this register are written or the other bits are not programmed. 0 = FIFOs are disabled. 1 = FIFOs are enabled.

16.3.4.8 Line Control Register

This register specifies the format for the asynchronous data-communications exchange. The serial-data format consists of a start bit, 8 data bits, an optional parity bit, and 1 stop bit. This register has bits that allow access to the Divisor Latch registers and bits that can cause a break condition.

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7	DLAB	RW	0x0	Divisor Latch Access Bit. Must be set to access the Divisor Latch registers of the baud-rate generator during a read or write operation.

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
				Must be clear to access the receive buffer, the Transmit Holding Register or the Interrupt Enable Register. 0 = access Transmit Holding Register, Receive Buffer Register, and Interrupt Enable Register. 1 = access Divisor Latch registers (DLL and DLH)
6	SB	RW	0x0	Set Break. Causes a break condition to be transmitted to the receiving UART. Acts only on the TXD pin and has no effect on the transmit logic. In FIFO mode, wait until the transmitter is idle (<Transmitter Empty> field in the Line Status Register = 1] to set and clear SB. 0 = No effect on TXD output. 1 = Forces TXD output to 0 (space).
5	STKYP	RW	0x0	Sticky Parity. Forces the bit value at the parity bit location to be the opposite of the <Even Parity Select> field rather than the parity value. This stops parity generation. If <Parity Enable> = 0, this field is ignored. 0 = No effect on parity bit. 1 = Forces parity bit to be opposite of <Even Parity Select> field value.
4	EPS	RW	0x0	Even Parity Select. If <Parity Enable> = 0, this field is ignored. 0 = Sends or checks for odd parity. 1 = Sends or checks for even parity
3	PEN	RW	0x0	Parity Enable.This field enables a parity bit to be generated on transmission or checked on reception. 0 = No parity. 1 = Parity
2	STB	RW	0x0	Stop Bits. Specifies the number of stop bits transmitted and received in each character. When receiving, the receiver checks only the first stop bit. This field must be clear. 0 = 1 stop bit.
1:0	WLS10	RW	0x0	Word Length Select. Specifies the number of data bits in each transmitted or received character. 0x0, 0x1, 0x2 = 7-bit character. 0x3 = 8-bit character.

16.3.4.9 Modem Control Register

This register uses the modem control pins RTSn and DTRn to control the interface with a modem or data set. This register also controls the loopback mode. Loopback mode must be enabled before the UART is enabled.

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7	EPT_RXREQ_EN	RW	0x0	this bit set to enable uart send dma_rx_req when time out not care the fifo empty state. 0=do not send dma_rxreq when fifo empty. 1= send dma_rxreq when time out no matter if there is data in the rx fifo.
6	EOR_INT_MASK	RW	0x0	mask bit for eor interrupt. 0 = eor interrupt detection logic will work. 1 = eor interrupt detection logic will not work.
5	AFE	RW	0x0	Auto-flow Control Enable. 0 = Auto-RTS and auto-CTS are disabled. 1 = Auto-CTS is enabled. If <Request to Send> is also set, both auto-CTS and auto-RTS are enabled.
4	LOOP	RW	0x0	Loopback Mode. This field provides a local loopback feature for diagnostic testing of the UART. When set, the following occurs: The transmitter serial output is set to a logic 1 state. The receiver serial input is disconnected from the pin. The output of the Transmit Shift Register is <q>looped back</q> into the Receive Shift Register input. The four modem control inputs (CTSn, DSRn, DCDn, and RIn) are disconnected from the pins and the modem control output pins (RTSn and DTRn) are forced to their inactive state. Coming out of the loopback mode may result in unpredictable activation of the delta bits (bits 30) in the Modem Status Register. CHIP recommends that the Modem Status Register be read once to clear its delta bits. Loopback mode must be configured before the UART is enabled. The lower four bits of this register are connected to the upper four Modem Status Register bits.<Data Terminal Ready> = 1 forces <Data Set Ready> in the Modem Status Register to a 1.<Request to Send> = 1 forces <Clear to Send> in the Modem Status Register to a 1.<Test Bit> = 1 forces <Ring Indicator> in the Modem Status Register to a 1.<OUT2 Signal Control> = 1 forces <Data Carrier Detect> in the Modem Status Register to a 1.In loopback mode, data that is transmitted is received immediately. This feature allows the <var Product Number> to verify the transmit and receive

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
				data paths of the UART. The transmit, receive, and modem-control interrupts are operational, except that the modem control interrupts are activated by Modem Control Register bits, not by the modem-control pins. A break signal can also be transferred from the transmitter section to the receiver section in loopback mode. 0 = normal UART operation. 1 = loopback-mode UART operation.
3	OUT2	RW	0x0	OUT2 Signal Control. OUT2 connects the UART interrupt output to the interrupt controller unit. When <Loopback Mode> is clear. 0 = UART interrupt is disabled.1 = UART interrupt is enabled.When <Loopback Mode> is set, interrupts always go to the <var Product Number>. 0 = <Data Carrier Detect> field in the Modem Status Register forced to 0. 1 = <Data Carrier Detect> field forced to 1.
2	Reserved	RO	0x0	Reserved for future use.
1	RTS	RW	0x0	Request to Send. 0 = Non-auto-flow mode. RTSn pin is 1. Auto-RTS disabled. Auto-flow works only with auto-CTS. 1 = Auto-flow mode. RTSn pin is 0. Auto-RTS enabled. Auto-flow works with both auto-CTS and auto-RTS.
0	DTR	RW	0x0	Data Terminal Ready. 0 = DTRn pin is 1. 1 = DTRn pin is 0.

16.3.4.10 Line Status Register

This register provides data-transfer status information to the <var Product Number>. In non-FIFO mode, bits [4:2] show the error status of the character that has just been received. In FIFO mode, bits [4:2] show the status bits of the character that is currently at the front of the FIFO.

Bits [4:1] produce a receiver-line-status interrupt when the corresponding conditions are detected and the interrupt is enabled. In FIFO mode, the receiver-line-status interrupt occurs only when the erroneous character reaches the front of the FIFO. If the erroneous character is not at the front of the FIFO, a line-status interrupt is generated after the other characters are read, and the erroneous character becomes the character at the front of the FIFO.

This register must be read before the erroneous character is read. Bits [4:1] remain set until software reads this register.

See FIFO DMA Mode Operation in **Section 16.3.3.4.6** for details on using the DMAC to receive data.

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7	FIFOE	RO	0x0	FIFO Error Status. In non-FIFO mode, this bit is clear. In FIFO mode, this field is set when there is at least one parity error, framing error, or break indication for any of the characters in the FIFO. A <var Product Number> read of the this register does not reset this field. This field is reset when all erroneous characters have been read from the FIFO. If DMA requests are enabled (<DMA Requests Enable> field in the Interrupt Enable Register set) and this field is set, the error interrupt is generated, and no receive DMA request is generated even when the receive FIFO reaches the trigger threshold. Once the errors have been cleared by reading the FIFO, DMA requests are re-enabled automatically. If DMA requests are not enabled (<DMA Requests Enable> field clear), this field set does not generate an error interrupt. 0 = No FIFO or no errors in receive FIFO. 1 = At least one character in receive FIFO has errors.
6	TEMT	RO	0x1	Transmitter Empty. Set when the Transmit Holding Register and the Transmit Shift Register are both empty. It is cleared when either the Transmit Holding Register or the Transmit Shift Register contains a data character. In FIFO mode, this field is set when the transmit FIFO and the Transmit Shift Register are both empty. 0 = There is data in the Transmit Shift Register, the Transmit Holding Register, or the FIFO. 1 = All the data in the transmitter has been shifted out.
5	TDRQ	RO	0x1	Transmit Data Request. This field indicates that the UART is ready to accept a new character for transmission. In addition, this field causes the UART to issue an interrupt to the <var Product Number> when the transmit data request interrupt enable is set and generates the DMA request to the DMA controller if DMA requests and FIFO mode are enabled. This field is set when a character is transferred from the Transmit Holding Register into the Transmit Shift Register. This field is cleared with the loading of the Transmit Holding Register. In FIFO mode, this field is set when half of the characters in the FIFO have been loaded into the Transmit Shift Register or the <Reset Transmit FIFO> field in the FIFO Control Register has been set. It is cleared when

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
				the FIFO has more than half data. If more than 64 characters are loaded into the FIFO, the excess characters are lost. 0 = There is data in the holding register or FIFO waiting to be shifted out. 1 = The transmit FIFO has half or less than half data.
4	BI	RO	0x0	Break Interrupt. This field 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 bit). It is cleared when the <var Product Number> reads the LSR. 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.
3	FE	RO	0x0	Framing Error. This field indicates that the received character did not have a valid stop bit. It is set when the bit following the last data bit or parity bit is detected to be 0. It is cleared when the <var Product Number> reads this register. The UART will resynchronize after a framing error. To do this, it assumes that the framing error was due to the next start bit, so it samples this start bit twice and then reads in the data. In FIFO mode, this field shows a framing error for the character at the front of the FIFO, not for the most recently received character. 0 = No Framing error. 1 = Invalid stop bit has been detected.
2	PE	RO	0x0	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. This field is set upon detection of a parity error and is cleared when the <var Product Number> reads this register. In FIFO mode, this field 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.
1	OE	RO	0x0	Overrun Error. In non-FIFO mode, indicates that data in the Receive

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
				Buffer register was not read by the <var Product Number> before the next character was received. The new character is lost. In FIFO mode, this field indicates that all 64 bytes of the FIFO are full and the most recently received byte has been discarded. This field is set upon detection of an overrun condition and cleared when the <var Product Number> reads this register. 0 = No data has been lost. 1 = Receive data has been lost.
0	DR	RO	0x0	Data Ready. Set when a complete incoming character has been received and transferred into the Receive Buffer Register or the FIFO. In non-FIFO mode, this field is cleared when the receive buffer is read. In FIFO mode, this field is cleared if the FIFO is empty (last character has been read from Receive Buffer Register) or the FIFO is reset with the <Reset Receive FIFO> field in the FIFO Control Register. 0 = No data has been received. 1 = Data is available in Receive Buffer Register or the FIFO.

16.3.4.11 Modem Status Register

This register provides the current state of the control lines from the modem or data set (or a peripheral device emulating a modem) to the <var Product Number>. In addition to this current state information, four bits provide change information. Bits [3:0] are set when a control input from the modem changes state. They are cleared when the <var Product Number> reads this register.

The status of the modem control lines does not affect the FIFOs. The <Modem Interrupt Enable> field in the Interrupt Enable Register must be set to use these lines for flow control. The interrupt service routine must disable the UART when an interrupt occurs on one of the flow-control pins. The UART continues transmission/reception of the current character and then stops. The contents of the FIFOs are preserved. If the UART is re-enabled, transmission continues where it stopped.

When bit 0, 1, 2, or 3 is set, a modem-status interrupt is generated if the <Modem Interrupt Enable> field is set.

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7	DCD	RO	0x0	Data Carrier Detect. Complement of the data-carrier-detect (DCDn) input. Equivalent to <OUT2 Signal Control> field in the Modem Control Register if <Loopback Mode> is set in the Modem Control Register.

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
				0 = DCDn pin is 1. 1 = DCDn pin is 0.
6	RI	RO	0x0	Ring Indicator. Complement of the ring-indicator (RIn) input. Equivalent to the <Test Bit> field in the Modem Control Register if <Loopback Mode> is set. 0 = RIn pin is 1. 1 = RIn pin is 0.
5	DSR	RO	0x0	Data Set Ready. Complement of the data-set-ready (DSRn) input. Equivalent to <Data Terminal Ready> field in the Modem Control Register if <Loopback Mode> is set. 0 = DSRn pin is 1. 1 = DSRn pin is 0
4	CTS	RO	0x0	Clear to Send. Complement of the clear-to-send (CTSn) input. Equivalent to <Request to Send> field in the Modem Control Register if <Loopback Mode> is set. 0 = CTSn pin is 1. 1 = CTSn pin is 0.
3	DDCD	RO	0x0	Delta Data Carrier Detect. 0 = No change in DCDn pin since the last read of this register. 1 = DCDn pin has changed state.
2	TERI	RO	0x0	Trailing Edge Ring Indicator. 0 = RIn pin has not changed from 0 to 1 since the last read of this register. 1 = RIn pin has changed state.
1	DDSR	RO	0x0	Delta Data Set Ready. 0 = No change in DSRn pin since the last read of this register. 1 = DSRn pin has changed state.
0	DCTS	RO	0x0	Delta Clear to Send. 0 = No change in CTSn pin since the last read of this register. 1 = CTSn pin has changed state.

16.3.4.12 Scratchpad Register

This register has no effect on the UART. It is intended as a scratchpad register for use by programmers and is included for 16550A compatibility.

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7:0	SCRATCHPAD	RW	0x0	SCRATCHPAD. This field has no effect on UART functions.

16.3.4.13 Infrared Selection Register

Each UART can manage an IrDA module associated with it. This register controls the IrDA functions (see Serial Infrared Asynchronous Interface in the Datasheet).

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	Reserved	RO	0x0	Reserved for future use.
4	RXPL	RW	0x0	Receive Data Polarity. 0 = SIR decoder takes positive pulses as zeros. 1 = SIR decoder takes negative pulses as zeros.
3	TXPL	RW	0x0	Transmit Data Polarity. 0 = SIR encoder generates a positive pulse for a data bit of 0. 1 = SIR encoder generates a negative pulse for a data bit of 0.
2	XMODE	RW	0x0	Transmit Pulse Width Select. When this field is clear, the UART 16x clock is used to clock the IrDA transmit and receive logic. When this field is set, the receive decoder operation does not change, and the transmit encoder generates 1.6 ms pulses (that are 3/16 of a bit time at 115.2 kbps) instead of pulses 3/16 of a bit time wide. CHIP recommends setting this field. 0 = Transmit pulse width is 3/16 of a bit time wide. 1 = Transmit pulse width is 1.6 ms.
1	RCVEIR	RW	0x0	Receiver SIR Enable. When this field is set, the signal from the RXD pin is processed by the IrDA decoder before it is fed to the UART. If this field is clear, then all clocking to the IrDA decoder is blocked and the RXD pin is fed directly to the UART. 0 = Receiver is in UART mode. 1 = Receiver is in infrared mode.
0	XMITIR	RW	0x0	Transmitter SIR Enable. When this field is set, the normal TXD output from the UART is processed by the IrDA encoder before it is fed

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
				to the device pin. If this field is clear, all clocking to the IrDA encoder is blocked and the UART's TXD signal is connected directly to the device pin. When transmitter SIR enable is set, the TXD output pin, which is in a normally high default state, switches to a normally low default state. This can cause a false start bit unless the infrared LED is disabled before this field is set. 0 = Transmitter is in UART mode. 1 = Transmitter is in infrared mode

16.3.4.14 Receive FIFO Occupancy Register

This register shows the number of bytes currently remaining the receive FIFO.

This register can be used to determine the number of trailing bytes to remove in the case when the DMA reaches the end of its descriptor chain or when the <Trailing Bytes> field in the FIFO Control Register is clear (for details, see **Section 16.3.3.4.4**, FIFO Interrupt Mode: Removing Trailing Bytes).

This register is incremented once for each byte of data written to the receive FIFO and decremented once for each byte read.

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved	RO	0x0	Reserved for future use.
5:0	BYTE_COUNT	RO	0x0	BYTE COUNT. This field is used for the number of bytes (0-63) remaining in the receive FIFO.

16.3.4.15 Auto-Baud Control Register

This register controls the functionality and options for auto-baud-rate detection within the UART. Through this register, software can enable/disable the auto-baud-lock interrupt, direct either the <var Product Number> or the UART to program the final baud rate in the Divisor Latch registers, and choose between two methods used to calculate the final baud rate.

The auto-baud circuitry counts the number of clocks in the start bit and writes this count into the Auto-Baud Count register (ACR). It then interrupts the <var Product Number> if the <Auto-baud Lock> field in the Interrupt Identification Register is set. It also programs automatically the Divisor Latch registers (DLL and DLH) if the <ABUP> field is set.

Auto-baud-rate detection is not supported in IrDA serial-infrared mode.

Offset: 0x28				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0x0	Reserved for future use.
3	ABT	RW	0x0	ABT. 0 = Formula used to calculate baud rates, allowing all possible baud rates to be chosen by UART. 1 = Table used to calculate baud rates, which limits UART to choosing common baud rates
2	ABUP	RW	0x0	ABUP. 0 = <var Product Number> Programs Divisor Latch registers. 1 = UART Programs Divisor Latch registers.
1	ABLIE	RW	0x0	ABLIE. 0 = Auto-baud-lock interrupt disabled (Source <Auto-baud Lock> field). 1 = Auto-baud-lock interrupt enabled (Source <Auto-baud Lock> field).
0	ABE	RW	0x0	ABE. 0 = Auto-baud disabled. 1 = Auto-baud enabled.

16.3.4.16 Auto-Baud Count Register

This register stores the number of 14.7456-MHz clock cycles within a start-bit pulse. This value is then used by the <var Product Number> or the UART to calculate the baud rate. If auto-baud mode (<ABE> field in Auto-Baud Control Register) and auto-baud interrupts (<ABLIE> field in Auto-Baud Control Register) are enabled, the UART interrupts the <var Product Number> with the auto-baud-lock interrupt (IIR[ABL]) after it has written the count value into ACR. The value is written regardless of the state of the auto-baud UART program bit, (ABR[ABUP]).

Offset: 0x2C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	COUNT_VALUE	RO	0x0	COUNT VALUE. This field is used for the number of 14.7456-MHz clock cycles within a start-bit pulse.

16.3.4.17 Full Baud Divisor Register

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:8	DLH	RW	0x0	Divisor Latch High. High-byte compare value to generate baud rate
7:0	DLL	RW	0x2	Divisor Latch Low. Low-byte compare value to generate baud rate

16.3.4.18 FIFO Control Register

Another address for FCR. Please refer to Offset = 0x8 for its detailed description.

It is a write and read register when baud_newreg_en is asserted.

Offset: 0x34				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7:6	ITL	RW	0x0	Interrupt Trigger Level (threshold).
5	Bus	RW	0x0	32-Bit Peripheral Bus
4	TRAIL	RW	0x0	Trailing Bytes
3	TIL	RW	0x0	Transmitter Interrupt Level
2	RESETTF	RW	0x0	Reset Transmit FIFO
1	RESETRF	RW	0x0	Reset Receive FIFO
0	TRFIFOE	RW	0x0	Transmit and Receive FIFO Enable

16.3.4.19 Baud Newreg Enable Register

config the baud_newreg_en to use the new address for DLH, DLL, FCR

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
31:2	Reserved	RO	0x0	Reserved for future use.
1	BAUD_SYNC_DONE	RWC	0x0	baud_sync_done. 1 = the completion of {DLH, DLL } sync to clk_uart domain from clk_apb domain when <baud_newreg_en> is set previously, can be cleared by writing this resiger(0x38) or full baud divisor register(0x34). 0 = default status.

Offset: 0x38				
Bits	**Field**	**Type**	**Reset**	**Description**
0	BAUD_NEWREG_EN	RW	0x0	baud_newreg_en. 0= no influence with the previous config, except the new read access for FCR in offset=0x34. 1= enable another new address access for {DLH, DLL} in offset= 0x30 and FCR in offset=0x34. The previous access for DLH, DLL, FCR are all blocked.

16.4 GPIO

16.4.1 Introduction

GPIO is used to capture and generate application-specific input and output. All ports of GPIO are brought out via the alternate function muxing. GPIO unit is in charge of GPIO ports control and status check. When programmed as an input, a GPIO port can also serve as an interrupt source. At the assertion of all resets, all GPIO ports are configured as inputs and remain inputs until they are configured either by the boot process or by user software.

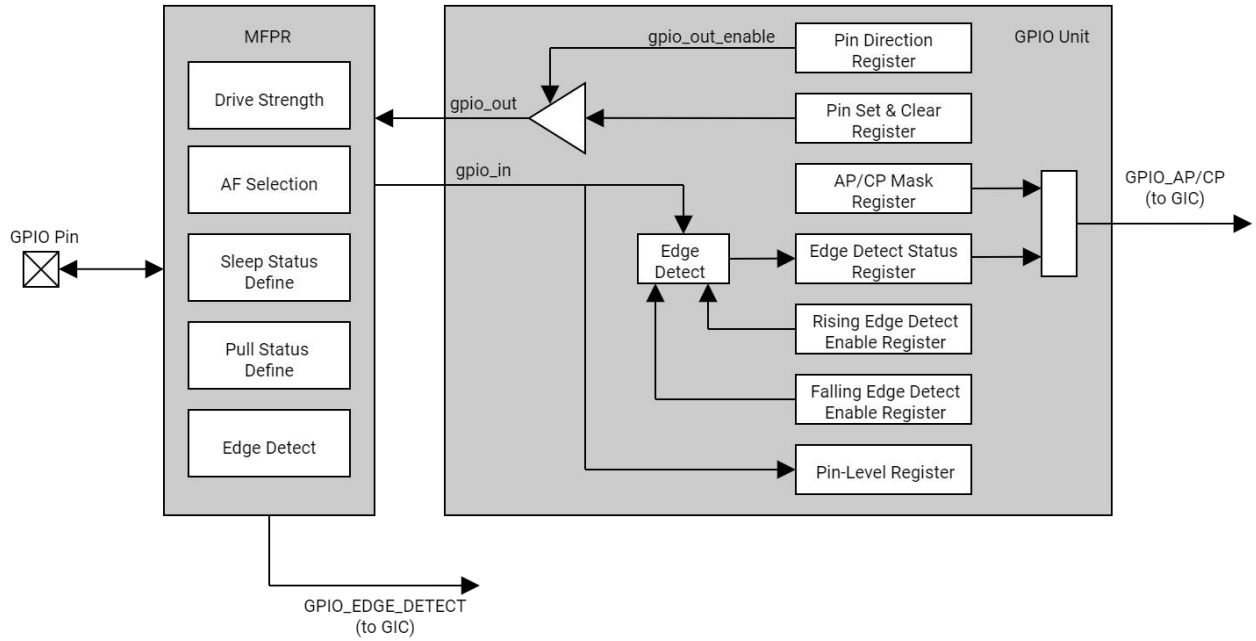
16.4.1.1 Features

- As inputs, they can be programmed to generate an interrupt at a rising edge, a falling edge or both
- As outputs, they can be cleared or set individually
- As inputs, the values can be read individually

16.4.2 Functional Description

16.4.2.1 Block Diagram

The GPIO ports and GPIO unit connections are depicted below.



AS can be seen, the GPIO unit output passes through MFPR to the GPIO pin, and the input passes through MFPR to the GPIO unit from the GPIO pin.

By virtue of the GPIO unit, software is able to set and check the status of a GPIO port. And there are 3 interrupts relevant to GPIO as tabled below

ICU Inputs	Signal Name	Notes
Int Req [58]	GPIO_AP	Generated in the GPIO unit
Int Req [59]	GPIO_AP_SEC	Generated in the GPIO secure unit
Int Req [60]	GPIO_edge_detect	Generated in MFPR, wake-up source

All of the interrupts may be generated when GPIO input edge is detected. However, they are generated in different modules. GPIO_EDGE_DETECT is generated in MFPR which is in always on domain and can be used as wake-up source. The other 2 interrupts (GPIO_AP/CP) are generated in GPIO unit which is shut down when chip is in sleep mode. As a result, GPIO_AP/CP can not be used as wake-up sources. Furthermore, GPIO_AP/CP has its corresponding bit masks which allows software to mask a bit or not, while GPIO_EDGE_DETECT does not have its bit mask.

16.4.2.2 Software Guide

The GPIO logic gives software the ability to configure and examine the value of a port, but does not control the selection of which logical function is connected to the actual physical pins.

For a GPIO port multiplexed on a particular multi-function I/O pin to have effect, 2 conditions must be met:

- The GPIO alternate function must be selected by using the MFPRx registers in the pad configuration registers
- The GPIO unit registers must be configured correctly

The direction of the GPIO ports is controlled by writing to the GPIO Pin Direction Register. When programmed as an output, the port can be set by writing to the GPIO Pin Output Set Register and cleared by writing to the GPIO

Pin Output Clear Register. The set and clear registers can be written regardless of whether the port is configured as an input or an output. If a port is configured as an input, the programmed output state takes effect when the port is reconfigured as an output.

The value of each GPIO port can be read through the GPIO Pin-Level Register. This register can be read at any time and can confirm the port state for both input and output configurations. In addition, each GPIO port can be programmed to detect a rising and/or falling edge through the GPIO

Rising-Edge Detect Enable Register and GPIO Falling-Edge Detect Enable Register. The state of the edge-detect can be read through the GPIO Edge Detect Status Register. These edge-detects can be programmed to generate interrupts.

The GPIO information described in this section applies only to the GPIO alternate function. Still, it is possible for a system to use the GPIO functions internally and not actually connect to a physical pin.

16.4.3 Register Description

Note. The base address of GPIO registers are tabled below.

Name	Address
GPIO0_BASE	0xD4019000
GPIO1_BASE	0xD4019004
GPIO2_BASE	0xD4019008
GPIO3_BASE	0xD4019100
SEC_GPIO0_BASE	0xF0619000
SEC_GPIO1_BASE	0xF0619004
SEC_GPIO2_BASE	0xF0619008
SEC_GPIO3_BASE	0xF0619100

16.4.3.1 GPIO_PLR REGISTER

The state of each of the GPIO ports is visible through this register. Each bit corresponds to the port number. These read-only registers determine the current value of a particular port (regardless of the programmed port direction).

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	PLn	R	0x0	GPIO port level n (where n = 0 ~ 31) 0: Port state is low 1: Port state is high

16.4.3.2 GPIO_PDR REGISTER

Users control port direction by programming the GPIO Pin Direction registers . They contain one direction control bit for each of the 32 ports.

If a direction bit is programmed to a 1, the GPIO function is an output. If it is programmed to a 0, it is an input. At reset, all bits in this register are cleared , which means all GPIO ports are configured to input.

A pair of set/clear registers (GPIO_SDRx and GPIO_CDRx) is also provided to enable the setting and clearing of individual bits of this register.

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	PDn	R/W	0x0	GPIO port direction n (where n = 0 ~ 31) 0: Port configured as an input 1: Port configured as an output

16.4.3.3 GPIO_PSR REGISTER

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31	PSn	W	0x0	GPIO output port set n (where n = 0 ~ 31) 0: Port level unaffected 1: If port configured as an output, set port level logic high

16.4.3.4 GPIO_PCR REGISTER

Offset: 0x24				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	PCn	W	0x0	GPIO output port clear n (where n = 0 ~ 31) 0: Port level unaffected 1: If port configured as an output, clear port level logic low

16.4.3.5 GPIO_RER REGISTER

The GPIO Edge Detect functionality described in this section is independent of and in addition to the Pin Control Unit edge-detect logic. In addition, the GPIO Edge Detect is used as an interrupt while the Pin Control Unit edge-detect logic is mainly used as a wakeup event.

Each GPIO can be programmed to detect a rising edge, falling edge, or either transition on a port. When an edge is detected that matches the type of edge programmed for the port, a status bit is set.

The GPIO Rising-Edge Detect Enable and GPIO Falling-Edge Detect Enable Registers (GPIO_RER and GPIO_FER, respectively) select the type of transition on a GPIO port, which causes a bit within the GPIO Edge-Detect Status Register (GPIO_EDR) to be set.

For a given GPIO port, its corresponding GPIO Rising-Edge Detect Enable Register bit is set to cause a GPIO Edge-Detect Status Register status bit to be set when the port transitions from logic level low to logic level high. Likewise, the GPIO Falling-Edge Detect Enable Register is used to set the corresponding GPIO Edge-Detect Status Register status bit when a transition from logic level high to logic level low occurs. When the corresponding bits are set in both registers, either a falling- or a rising-edge transition causes the corresponding GPIO Edge-Detect Status Register status bit to be set.

Offset: 0x30				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	PEn	R/W	0x0	GPIO port n rising-edge detect enable (where n = 0 ~ 31) 0: Disable rising-edge detect enable 1: Set corresponding GPIO Edge Detect Status Register status bit when a rising edge is detected on the GPIO port

16.4.3.6 GPIO_FER REGISTER

Refer to description of **GPIO_RER REGISTER**

Offset: 0x3C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	FEn	R/W	0x0	GPIO port falling-edge detect enable n (where n = 0 ~ 31) 0: Disable falling-edge detect enable 1: Set corresponding GPIO Edge Detect Status Register status bit when a falling edge is detected on the GPIO port

16.4.3.7 GPIO_EDR REGISTER

The GPIO Edge Detect Status Registers contain a total of 32 status bits that correspond to the 32 GPIO ports.

When an edge-detect occurs on a port that matches the type of edge programmed in the GPIO Rising-Edge Detect Enable and/or GPIO Falling-Edge Detect Enable Registers, the corresponding status bit is set in this register. Once a bit is set in this register, the CPU must clear it. Status bits in this register are cleared by writing a 1 to them. Writing a 0 has no effect.

Offset: 0x48				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	EDn	R/W1C	0x0	GPIO edge detect status n (where n = 0 ~ 31) 0: No edge detect has occurred on the port as specified in GPIO Rising-Edge Detect Enable and/or GPIO Falling-Edge Detect Enable Registers 1: Edge detect has occurred on the port as specified in the GPIO Rising-Edge Detect Enable and/or GPIO Falling-Edge Detect Enable Registers

16.4.3.8 GPIO_SDR REGISTER

Users control port direction by programming the GPIO pin Bit-wise Set of the GPIO Direction Registers. These registers contain one direction control bit for each of the 32 ports.

If a direction bit is programmed to 1, the corresponding bit in the GPIO Pin Direction Register is set and the GPIO function is configured as an output. If it is programmed to a 0, no change in the GPIO functionality or the GPIO Pin Direction Register occurs.

Offset: 0x54				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SDn	W	0x0	Set GPIO port direction n (where n = 0 ~31) 0: GPIO Pin Direction Register bit not affected 1: GPIO Pin Direction Register bit is set and GPIOx function is set to OUTPUT

16.4.3.9 GPIO_CDR REGISTER

Users control pin direction by programming the GPIO pin Bit-wise Clear of the GPIO Direction Registers. These registers contain one direction control bit for each of the 32 pins.

If a direction bit is programmed to a 1, the corresponding bit in the GPIO Pin Direction Register is cleared and the GPIO function is configured as an input. If it is programmed to a 0, no change in the GPIO functionality or the GPIO Pin Direction Register occurs.

Offset: 0x60				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CDn	W	0x0	Set GPIO port direction n (where n = 0 ~ 31) 0: GPIO Pin Direction Register bit not affected 1: GPIO Pin Direction Register bit is cleared and the GPIO n function is set to INPUT

16.4.3.10 GPIO_SRERX REGISTER

Bit-wise set of GPIO rising edge detect enable register.

Offset: 0x6C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SRERn	W	0x0	Set GPIO Rising Edge detect enable n (where n = 0~ 31) 0: GPIO Rising-Edge Detect Enable Register bit is not affected 1: GPIO Rising-Edge Detect Enable Register bit is set

16.4.3.11 GPIO_CRERX REGISTER

Bit-wise clear of GPIO rising edge detect enable register.

Offset: 0x78				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CRERn	W	0x0	Clear GPIO Rising Edge detect enable n (where n = 0 ~ 31) 0: GPIO Rising-Edge Detect Enable Register bit is not affected 1: GPIO Rising-Edge Detect Enable Register bit is cleared

16.4.3.12 GPIO_SFERX REGISTER

Offset: 0x84				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	SFERn	W	0x0	Set GPIO Falling Edge detect enable n (where n = 0 ~ 31) 0: GPIO Falling-Edge Detect Enable Register bit not affected 1: GPIO Falling-Edge Detect Enable Register bit is set

Bit-wise set of GPIO rising edge detect enable register.

16.4.3.13 GPIO_CFERX REGISTER

Bit-wise clear of GPIO rising edge detect enable register.

Offset: 0x90				
Bits	**Field**	**Type**	**Reset**	**Description**
31:0	CFERn	W	0x0	Clear GPIO Falling Edge detect enable n (where n = 0 ~ 31) 0: GPIO Falling-Edge Detect Enable Register bit not affected 1: GPIO Falling-Edge Detect Enable Register bit is cleared

16.5 One-Wire Bus Master Interface

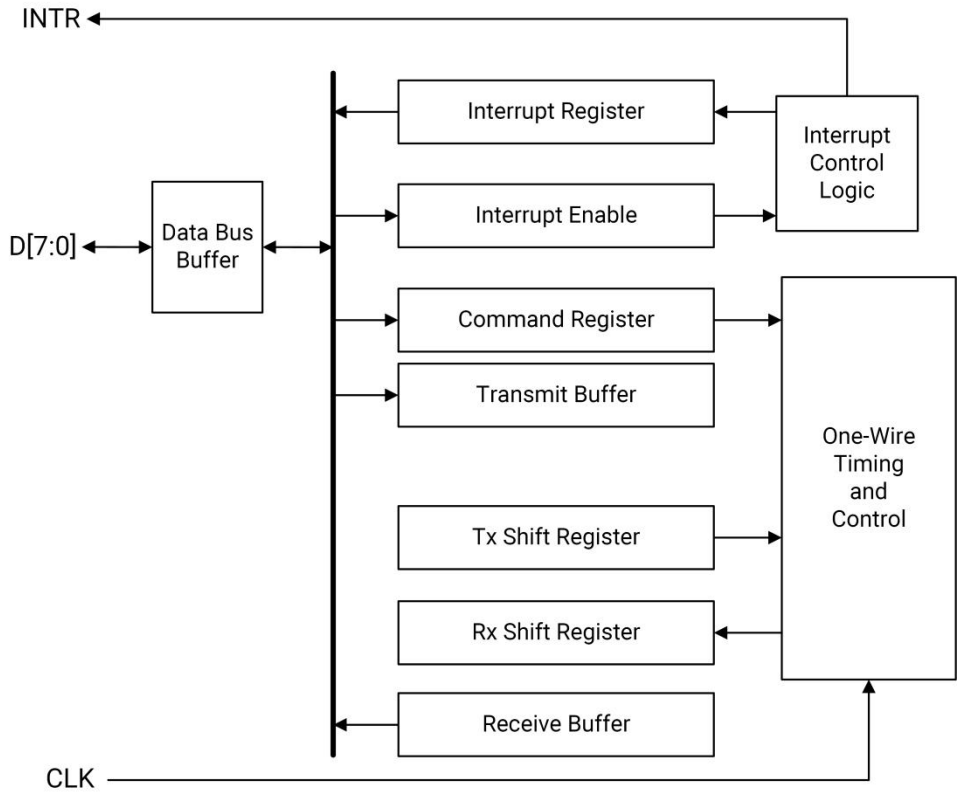
16.5.1 Introduction

The One-Wire Bus Master Interface Controller is responsible for receiving and transmitting data on the One-Wire bus. It fully controls the One-Wire bus using 8-bit commands. The processor interacts with the controller by loading commands, reading and writing data, and configuring interrupt controls through 5 specific registers.

All One-Wire bus timing and control are generated within the One-Wire Bus Master Interface Controller once a command or data is loaded by the host. When there is activity on the bus that requires the CPU to respond, the One-Wire Bus Master Interface Controller sets a status bit and, if enabled, sends an interrupt to the CPU.

For detailed information about specific slave implementations, please refer to the **Book of iButton® Standards** which describes the operation of the One-Wire bus master interface.

The architecture of the One-Wire Bus Master Interface is depicted below.



16.5.2 Functional Description

16.5.2.1 Signal

The functionality of the 1-Wire signal is tabled below.

Name	Direction	Description
ONE_WIRE	In/Out	1-Wire Data Line This open-drain line is the 1-Wire bidirectional data bus signal. 1-Wire slave devices are connected to this pin. This pin must be pulled high by an external resistor, nominally 5 KΩ.

16.5.2.2 Operations

This section describes the procedure for reading and writing on the 1-Wire interface. The 1-Wire Bus Master Interface Controller is located on the APB. The 1-Wire protocol requires a reset before any bus communication. Generating a 1-Wire reset on the bus is discussed in Initialization Sequence, in the following section.

The 1-Wire Bus Master Interface Controller generates Read, Write, or Reset commands on the 1-Wire bus. The 1-Wire reset is a special command that must precede the command given on the bus.
All waveform illustrations in this section correspond to the 1-Wire interface module signals.

16.5.2.2.1 Writing a Byte

To send a byte on the 1-Wire bus, write the byte to be transferred to the Transmit buffer. The data is then moved to the Tx Shift Register (Tx shift and Rx shift registers are not accessible by software) where it is shifted serially onto the bus, LSB first. A new byte of data can then be written to the transmit buffer. As soon as the Tx Shift Register is empty, the new data is transferred from the Transmit buffer and the process repeats.

Each register has a flag for use as an interrupt source. The Transmit buffer empty flag, W1INTR[TBE], is set when the Transmit buffer is empty and ready to accept a new byte. W1INTR[TBE] is cleared as soon as a byte is written into the Transmit buffer. The Tx Shift Register empty flag, W1INTR[TEMT], is set when the Tx Shift Register has no data in it and is ready to accept a new byte. As soon as a byte of data is transferred from the Transmit buffer, W1INTR[TEMT] is cleared and W1INTR[TBE] is set.

16.5.2.2.2 Reading a Byte

Before data can be read from a slave device, the device must be prepared to transmit data. The slave device is prepared through commands previously received from the CPU. Data is retrieved from the bus in a similar way as a Write operation. The host initiates a Read by writing to the Transmit buffer. The data that is then shifted into the Rx Shift Register (see note in Figure-20 -) is the wired-AND of the written data and the data from the slave device. Therefore, the host must write 0xFF to read a byte from a slave device. When the Rx Shift Register is full, the data is transferred to the Receive buffer where the host can access it.

Additional bytes can now be read by sending 0xFF again. If the slave device is not ready to transmit, the data received is identical to that transmitted. The 1-Wire Transmit/Receive Buffer (W1TRR) register can also generate interrupts. The Receive buffer flag, W1INTR[RBF], is set when data is transferred from the Rx Shift Register and cleared when the host reads the Rx Shift Register. If RBF is set, there must be no additional transmissions on the 1-Wire bus because the byte in the Receive buffer is overwritten by the next received byte, resulting in lost data. Interrupt flags are explained in detail in **Section 16.5.3.3**, Interrupt Register.

16.5.2.3 I/O signaling

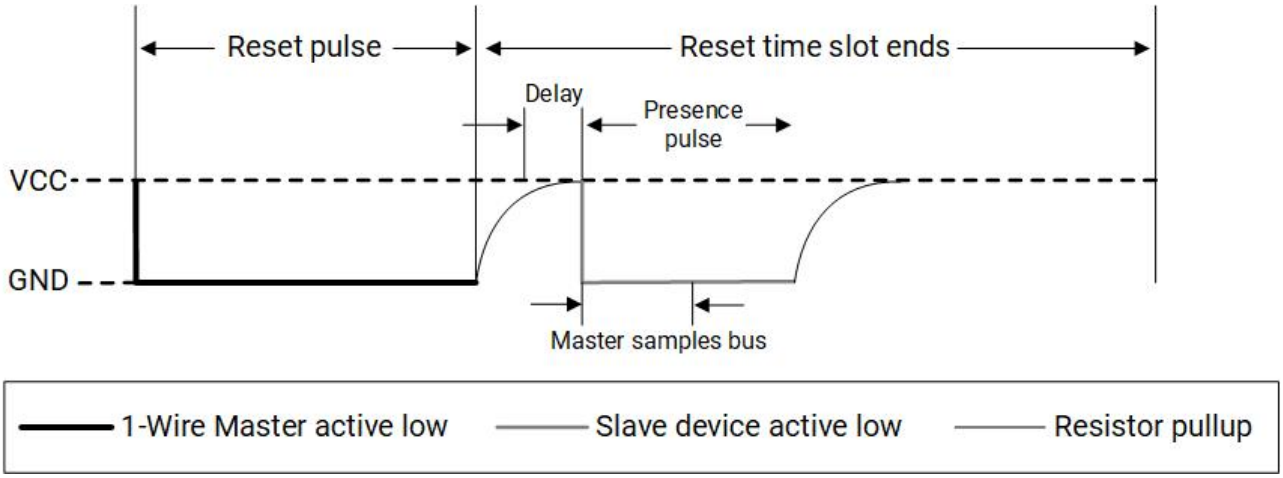
The 1-Wire bus requires strict signaling protocols to ensure data integrity. The 4 protocols used by the 1-Wire Bus Master Interface Controller are as follows:

- Initialization sequence (reset pulse followed by **presence pulse**)
- Write 0
- Write 1
- Read data

The master initiates all of these types of signaling except the presence pulse.

16.5.2.3.1 Initialization Sequence

The initialization sequence required to begin any communication with the bus slave is depicted below.

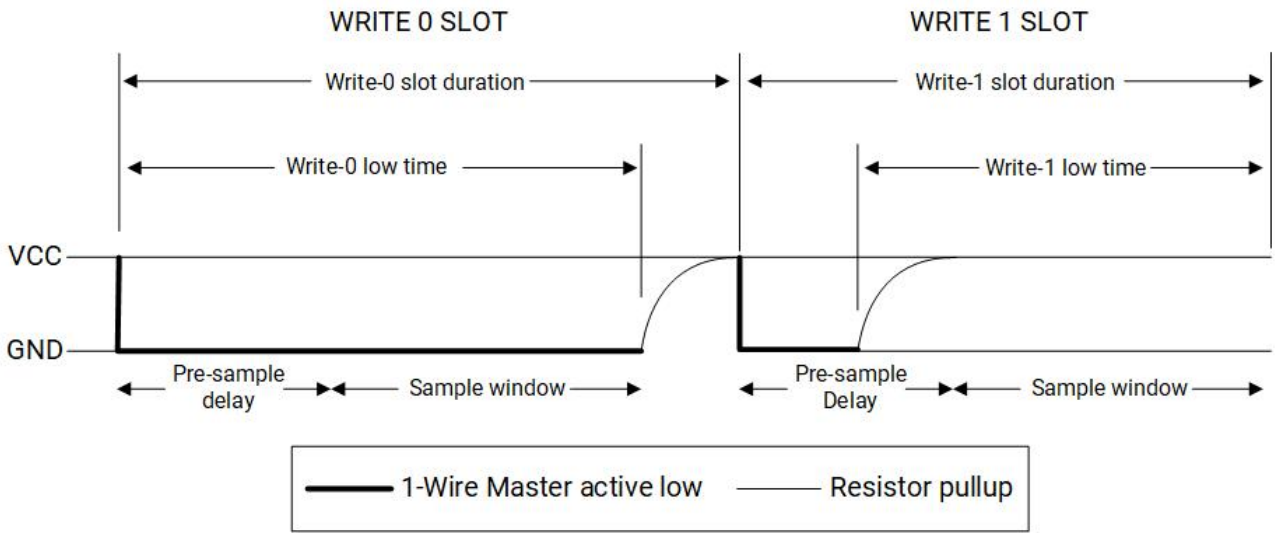


As can be seen:

- The 1-Wire Bus Master Controller transmits a reset pulse.
- The 1-Wire bus line is then pulled high by the external pullup resistor. Marvell recommends this pullup resistor have less than 5 kΩfor this initialization process to work correctly otherwise it may result in a small presence pulse width that can cause functional failure.
- After detecting the rising edge on the ONE_WIRE pin, the slave waits for a required amount of delay time and then transmits the presence pulse.
- The master samples the bus during the presence pulse after the slave responds to a test for a valid-presence pulse. The result of this sample is stored in the PDR bit of the 1-Wire Interrupt Register, W1INTR[PDR].
- The reset time slot ends a specified amount of time after the master releases the bus.

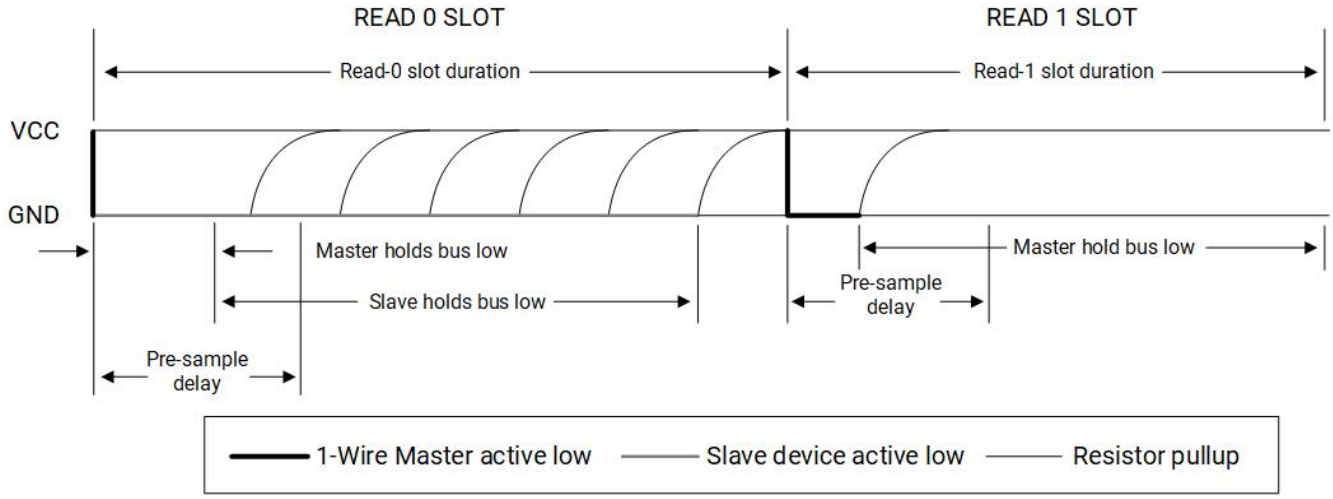
16.5.2.3.2 Write Time Slots

A Write time slot is initiated when the 1-Wire bus master drives the 1-Wire bus line from a logic-high (inactive) level to a logic-low level. The 1-Wire bus master generates a Write-1 time slot by releasing the bus line after the required amount of logic-low time for a Write-1 allowing the bus line to pull up to a logic-high level. The 1-Wire bus master generates a Write-0 time slot by holding the bus line low for the required amount of logic-low time. A slave device then samples the bus line a specified amount of time after the bus line transitions from a logic-high to a logic-low state. A Write-1 occurs if the bus line is logic-high when sampled, a Write-0 occurs if the bus line is logic-low when sampled, as depicted below.



16.5.2.3.3 Read Time Slots

A Read time slot is initiated when the 1-Wire bus master pulls the bus line low for the minimum required time and then releases it. If the slave device responds with a 0, the slave continues to hold the bus line low for up to a specified amount of time before the slave releases the bus line; the bus line is then pulled high by the external pullup resistor. If the slave device responds with a 1, the slave does not hold the bus line low and the bus line is pulled to a logic-high immediately after the master releases the line. The master samples the data after a specified amount of time after the start of the Read time slot. The master ends the Read slot after the required amount of time, as depicted below.



16.5.2.4 Operations

The 1-Wire Bus master can transmit a programmed set of bits (maximum 128 bits allowed) at the required clock frequency (in SDI mode). This interface is useful in programming the switch regulators on the board. For example, data of 0xCF is to be written to the switch regulator with the timing requirements as shown.

Since for a (“1” or “0”) pulse, the minimum period required is 0.25 µs which is about 4 MHz:

- Write a value of 0x5 to the OneWire Clock Divisor Register, which provides a clock frequency of approximately 4 MHz and is within the timing requirements
- The example in data 0xCF requires 18 pulses, so a total of 72 minimum periods need to be driven on the SDI bus. Program the 72 bits in the SDI Buffer Registers starting from offset 0x40. The “msb” is sent out on the interface first. In this example, the data programmed would be:
- SDI Buffer 0 = 0x1771_1111, SDI Buffer 1 = 0x1717_7171, SDI Buffer 2 = 0xXXXX_XX17,SDI Buffer 3 = 0xXXXX_XXXX, SDI Bit Count = 0x47
- If required, set the SDI Control Register interrupt enable. Set the <sdi_en> field in the SDI Control Register
- Once all the bits are sent out, the <sdi_en> field is cleared, and the interrupt is generated if it is enabled

16.5.2.5 Interrupt Handling

Three types of interrupts are programmed as follows:

- Receive buffer flag (RBF) interrupt
- Transmit Shift Register empty flag (TEMPT) interrupt and
- Transmit buffer empty (TBE) interrupt.

This section describes how to program these types of interrupts.

The typical programming flow for an RBF interrupt proceeds as follows:

- Enable the required interrupts and set W1IER[ERBF] and W1IER[EPD]
- Assert the INTR signal active high. Set W1IER[IAS]
- Reset the bus. Set W1CMDR[1WR]
- Unmask interrupts in the ICU
- When the PD interrupt occurs, mask interrupts in the ICU in the service routine, clear the W1IER[EPD] bit, and read the W1INTR[PDR] and W1INTR[PD] bits. Ensure that these values are correct (that is, PDR = 0 if there is a slave and PD = 1). Do not clear or write over the W1IER[IAS] bit. Exit the interrupt service routine.
- Send a command. Write to the xmit buffer: W1TRR[DATA]. Unmask interrupts. The Receive buffer is filled with the contents written to the xmit buffer.
- Wait for the RBF interrupt. In a service routine, mask interrupts, and clear the W1IER[ERBF] bit.
- Read the W1INTR[RBF] bit and ensure that it is set
- Read the Receive buffer W1TRR[DATA] to verify that the data is sent

The typical programming flow for a TEMPT interrupt proceeds as follows:

- Enable the required interrupts. Set W1IER[ETMT] and W1IER[EPD]
- Set the INTR signal active high. Set W1IER[IAS]
- Reset the bus. Set W1CMDR[1WR]
- Unmask interrupts in the ICU
- When the PD interrupt occurs, mask interrupts in the ICU in the service routine, clear the W1IER[EPD] bit, and read the W1INTR[PDR] and W1INTR[PD] bits

Ensure that these values are correct (that is, PDR = 0 if there is a slave and PD = 1). Do not clear or write over the W1IER[IAS] bit. Leave interrupt service routine.

- Send data. Write to the xmit buffer: W1TRR[DATA]. Unmask interrupts
- Wait for TEMT interrupt. In the service routine, mask interrupts, and clear the W1IER[ETMT] bit
- Read the W1INTR[TEMT] bit and ensure that it is set

The typical programming flow for the TBE interrupt proceeds as follows:

- Enable the required interrupts. Set W1IER[ETBE] and W1IER[EPD]
- Assert the INTR signal active high. Set W1IER[IAS]
- Reset the bus. Set W1CMDR[1WR]
- Unmask interrupts in the ICU
- When the PD interrupt occurs, mask interrupts in the ICU in the service routine, clear the W1IER[EPD] bit, and read the W1INTR[PDR] and W1INTR[PD] bits

Ensure that these values are correct (that is, PDR = 0 if there is a slave and PD = 1). Do not clear or write over the W1IER[IAS] bit. Unmask the interrupts. Leave the interrupt service routine.

- Send data. Write to the xmit buffer: W1TRR[DATA]
- Wait for TBE interrupt. In the service routine, mask interrupts, clear the W1IER[ETBE] bit. Read the W1INTR[TBE] bit and ensure that it is set
- New data, if needed, can be written into the buffer and interrupts should be unmasked

16.5.3 Register Description

Note. The base address of 1-Wire Bus Master Registers is 0xD4011800.

16.5.3.1 Command Register

This control register contains four valid bit fields that control the One-Wire bus master controller functionality. In addition, this register contains two bits to bypass the One-Wire bus master interface controller features and control the One-Wire bus directly.

This control register contains four valid bit fields that control the One-Wire bus master controller functionality. The One-Wire bus master interface controller can generate one special command on the bus, in addition to reading and writing, which is a One-Wire reset that must precede any command given on the bus. In addition, this register contains two bits to bypass the One-Wire bus master interface controller features and control the One-Wire bus directly.

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0x0	Reserved for future use.
3	DQI	RO	0x0	ONE_WIRE input. This bit reflects the present state of the One-Wire bus. Use it together with the <ONE_WIRE output> field when controlling the bus directly. The state of this bit does not affect any other functions of the One-Wire bus master interface controller. Operation of this bit is unaffected by the state of the

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
				<ONE_WIRE output enable> field in the One-Wire Interrupt Enable Register.
2	DQO	WO	0x0	ONE_WIRE output. This bit is used to bypass One-Wire bus master interface controller operations and drive the bus directly if needed. 0 = This bit is cleared on power-up or reset. Clearing this bit drives the bus high. One-Wire bus master interface controller operations only function while the One-Wire bus is held high. 1 = Setting this bit drives the bus low until it is cleared or the One-Wire bus master interface controller reset. While the One-Wire bus is held low, no other One-Wire bus master interface controller operations function. By controlling the length of time this bit is set and the point when the line is sampled (see <ONE_WIRE input> field), any One-Wire communication can be generated by the host controller. To prevent accidental writes to the bus, <ONE_WIRE output enable> field in the One-Wire Interrupt Enable Register before the functions in this field.
1	SRA	RW	0x0	Search ROM accelerator. 0 = SRA turned off. 1 = One-Wire bus master interface controller switches to SRA mode Refer to the Book of iButton Standards for more information on this feature.
0	1WR	RW	0x0	One-Wire reset. This field generates a reset on the One-Wire bus. 0 = Bus is not in reset mode. 1 = Setting this bit automatically clears the <Search ROM accelerator> field. This field is cleared automatically as soon as the One-Wire reset completes. The One-Wire bus master interface controller sets the <Presence detect> interrupt flag in the One-Wire Interrupt Register when the reset is complete and sufficient time for a presence detect to occur has passed. The result of the presence detect is placed in the <Presence detect result> field in the One-Wire Interrupt Register. If a presence detect pulse was received, the <Presence detect result> field is cleared, otherwise it is set.

16.5.3.2 Transmit/Receive Buffer Register

Data sent and received from the One-Wire bus master interface controller passes through the transmit/receive buffer location. The One-Wire bus master interface controller is double-buffered with separate transmit and receive buffers. Writing to this location connects the transmit buffer to the data bus, while reading connects the receive buffer to the data bus.

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7:0	Data	RW	0x0	Transmitted/received data.

16.5.3.3 Interrupt Register

This read-only register contains flags from transmit, receive, and One-Wire reset operations. Only the presence-detect flag (<Presence detect>) is cleared when the this register is read. The other flags are cleared automatically when the transmit and receive buffers are written to or read, respectively. These flags can generate an interrupt if the corresponding enable bit is set in the One-Wire Interrupt Enable Register.

This read-only register contains flags from transmit, receive, and One-Wire reset operations. Only the presence-detect flag (<Presence detect>) is cleared when this register is read. The other flags are cleared automatically when the transmit and receive buffers are written to or read, respectively. These flags can generate an interrupt on the INTR signal if the corresponding enable bit is set in the One-Wire Interrupt Enable Register. Reading the One-Wire Interrupt register always sets the INTR signal inactive, even if all flags are not cleared.

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved	RO	0x0	Reserved for future use.
5	RSRF	RO	0x0	Receive Shift Register Full. 0=This flag is cleared when the byte is read from the Receive shift register. 1=This flag is set when there's a byte written to shift register.
4	RBF	RO	0x0	Receive buffer full. 0 = This flag is cleared when the byte is read from the Receive Buffer register. 1 = This flag is set when there is a byte waiting to be read in the receive buffer.
3	TEMT	RO	0x0	Tx Shift register empty 0 = This flag is cleared when data is shifted into the Trx Shift register from the transmit buffer. 1 = This flag is set after the last bit has been transmitted on the One-Wire bus. TEMT status bit is valid upon completion of the first data transfer.
2	TBE	RO	0x1	Transmit buffer empty. 0 = This flag is cleared when data is written to the transmit buffer. 1 = This flag is set when the last bit is transferred to the Tx Shift register. TBE status bit is valid after the first data transfer.
1	PDR	RO	0x1	Presence detect result.When a Presence Detect interrupt occurs, this field reflects the result of the presence detect read. 0 = A slave device was found.

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
				1 = No slave device was found.
0	PD	RO	0x0	Presence detect. 0 = The required time after a One-Wire reset has not elapsed or this register has been read since the last One-Wire reset. 1 = After a One-Wire reset has been issued, this flag is set after the appropriate amount of time for a presence detect pulse to have occurred. This bit is cleared when the Interrupt register is read.

16.5.3.4 Interrupt Enable Register

This register allows system programmers to specify which of the interrupt sources causes an interrupt the INTR signal to be active and to define the active state for the INTR signal. When a master reset is received, all non-reserved bits in this register except for the <INTR active state> field are cleared to 0, disabling all interrupt sources. The <INTR active state> field is reset to 1 by the One-Wire controller.

This register allows system programmers to specify which of the interrupt sources causes an interrupt the INTR signal to be active and to define the active state for the INTR signal. When a master reset is received, all non-reserved bits in this register except for the <INTR active state> field are cleared to 0, disabling all interrupt sources. The <INTR active state> field is reset to 1 by the One-Wire controller, setting the active state of the INTR signal to high. This means the INTR signal is pulled low since all interrupts are disabled. The INTR signal is also reset to an inactive state by reading the Interrupt register.

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved for future use.
7	DQOE	RW	0x0	ONE_WIRE output enable. This bit acts as a control select for the ONE_WIRE bus. When set to 0, the bus is controlled by the One-Wire bus master interface controller as normal. 0 = This bit defaults to 0 on power-up or reset and should be left 0 unless users want to control the bus manually through the <ONE_WIRE output> field in the One-Wire Command Register. 1 = The <ONE_WIRE output> field controls the state of the bus directly.
6	Reserved	RO	0x0	Reserved for future use.
5	ERSF	RW	0x0	Enable Receive Shift Register Full Interrupt. 0 = Enable receive shift register full interrupt disabled. 1 = If the receive shift register full flag is set, then an interrupt is generated.
4	ERBF	RO	0x0	Enable Receive Buffer Full Interrupt. 0 = Enable receive buffer full interrupt disabled. 1 = If the receive buffer full flag is set, then an interrupt is

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
				generated.
3	ETMT	RO	0x0	Enable Tx Shift register empty interrupt. 0 = Enable Tx Shift register empty interrupt disabled. 1 = If the Tx Shift register empty flag is set, then an interrupt is generated.
2	ETBE	RO	0x0	Enable transmit buffer empty interrupt. 0 = Enable transmit buffer empty interrupt disabled. 1 = If the transmit buffer empty flag is set, then an interrupt is generated.
1	Reserved	RO	0x0	Reserved for future use
0	EPD	RW	0x0	Enable presence detect interrupt. 0 = Enable presence detect interrupt disabled. 1 = If the enable presence detect flag is set, an interrupt is generated whenever a One-Wire reset is sent and the required amount of time has passed for a presence detect pulse to have occurred.

16.5.3.5 Clock Divisor Register

This register divides the internal reference clock to generate the One-Wire clock timing patterns using a base clock of 24 MHz. This register must be programmed before using the One-Wire bus master interface.

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:5	Reserved	RO	0x0	Reserved for future use.
4:2	DIV	RW	0x0	Divider. The One-Wire bus master interface controller uses the output of the prescaler and divides by the DIV value to produce the One-Wire clocks. This clock must be approximately 1 MHz for correct operation. This value must be set to 0x2.
1:0	PRE	RW	0x0	Prescaler value. The One-Wire bus master interface controller uses the input 24-MHz clock and initially divides by this value before outputting to the divider section. This value must be set to 0x3, selecting a prescale of 7.

16.6 IR-RX

16.6.1 Overview

The IR-RX module is capable of receiving infrared signals and transforming the received signals into digital format. Received data can be accessed through FIFO by checking status or configuring interrupt.

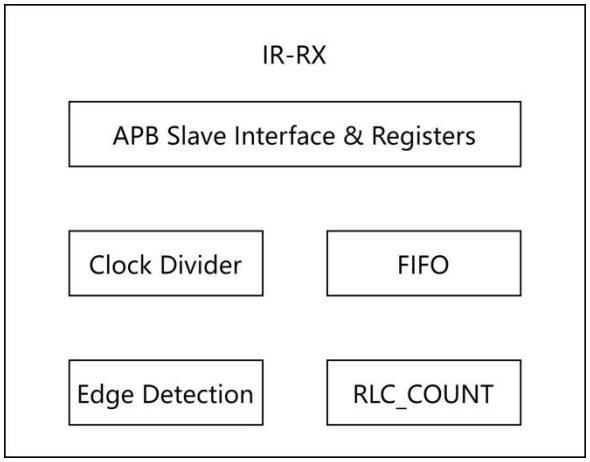
16.6.2 Features

- Infrared input signals are transformed into the Run-Length-Code (RLC) format
- Configurable signal width threshold for noise detecting
- 32 Bytes FIFO for received data storage

16.6.3 Functional Description

The IR-RX module receives infrared signals and transforms the received information into digital format. The input infrared signals are filtered depending on a configurable noise detecting threshold. The transformed data is written into FIFO as the Run-Length-Code (RLC). Software may read the data from FIFO by checking status or configuring interrupt.

The block diagram of IR-RX is depicted below.



The input clock signal (CLK) is divided by a configurable parameter to generate the internal working clock (WCLK) which is used to measure the duration or transtion time of the input infrared signal.

When IR_EN=1, the 7-bit width RLC_COUNT counts under the working clock. The RLC_COUNT is cleared to be 0 either if IR_RX changes or if it counts to 127.

When IR_RX changes or RLC_COUNT counts to 127, 1 byte data is written into FIFO.

If RLC_COUNT > NOISETHR at the data sampling moment, the current IR_RX value and the current RLC_COUNT value are stored, which is {IR_RX, RLC_COUNT[6:0]}.

If RLC_COUNT <= NOISETHR at the data sampling moment, the previously stored IR_RX value and the current RLC_COUNT value are stored, which is {IR_RX_old, RLC_COUNT[6:0]}.

For example, the following data in RLC format implies that:

- 0x98 –> Logic ‘1’ has sustained for 0x18 working clock cycles.
- 0x7F –> Logic ‘0’ has sustained for 0x7F working clock cycles.
- 0x06 –> Logic ‘0’ has sustained for 0x06 working clock cycles.
- 0xFF –> Logic ‘1’ has sustained for 0x7F working clock cycles.

16.6.4 Register Description

Note.

- The base address of IR-RX registers in the X60™ field is 0xD401_7F00
- The base address of IR-RX registers in the N308 field is 0xC088_E000

16.6.4.1 IRC_EN REGISTER

Offset: 0x0				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0x0	Reserved
0	IRC_EN	RW	0x0	This is the global enable bit for the IR-RX. 0x0: Disabled 0x1: Enabled

16.6.4.2 CLKDIV REGISTER

Offset: 0x4				
Bits	**Field**	**Type**	**Reset**	**Description**
31:24	Reserved	RO	0x0	Reserved
23:0	CLKDIV	RW	0x0	Frequency dividing parameter for generating the internal working clock (WCLK). The generated WCLK frequency is: Freq_of_WCLK = Freq_of_CLK / (CLKDIV +1)

16.6.4.3 18.6.4.3 NOISEHTR REGISTER

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved

Offset: 0x8				
Bits	**Field**	**Type**	**Reset**	**Description**
7:0	NOISETHR	RW	0x0	Noise detection threshold.

16.6.4.4 IDLE_STATE REGISTER

Offset: 0xC				
Bits	**Field**	**Type**	**Reset**	**Description**
31:1	Reserved	RO	0x0	Reserved
0	IDLE_STATE	RW	0x1	This is the IDLE status bit. 0x0: Not IDLE 0x1: IDLE It is cleared by hardware at the change of input infrared signal. Software could set this bit to 0x1.

16.6.4.5 FIFO_OUT REGISTER

Offset: 0x10				
Bits	**Field**	**Type**	**Reset**	**Description**
31:8	Reserved	RO	0x0	Reserved
7:0	FIFO_OUT	RO	0x0	This is the data output of FIFO.

16.6.4.6 FIFO_STS REGISTER

Offset: 0x14				
Bits	**Field**	**Type**	**Reset**	**Description**
31	FIFO_full	RO	0x0	Flag bit of FIFO full.
30	FIFO_empty	RO	0x1	Flag bit of FIFO empty.
29:6	Reserved	RO	0x0	Reserved
5:0	FIFO_CNT	RO	0x0	This is the number of unread data in FIFO.

16.6.4.7 FIFO_CMP REGISTER

Offset: 0x18				
Bits	**Field**	**Type**	**Reset**	**Description**
31:6	Reserved	RO	0x0	Reserved
5:0	FIFO_CMP	RW	0x0	Comparison value for the number of unread data in FIFO. It is used to generate interruption.

16.6.4.8 INT_EN REGISTER

Offset: 0x1C				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0x0	Reserved
3	CMP_INT_EN	RW	0x0	Interrupt enable bit for comparison between FIFO_CMP and the number of unread data in FIFO.
2	CNT_INT_EN	RW	0x0	Interrup enable bit for RLC_COUNT counts to 127.
1	PEDGE_INT_EN	RW	0x0	Interrupt enable bit for the positive edge of the input infrared signal.
0	NEDGE_INT_EN	RW	0x0	Interrupt enable bit for the negtive edge of the input infrared signal.

16.6.4.9 INT_FLAG REGISTER

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
31:4	Reserved	RO	0x0	Reserved
3	CMP_INT_FLAG	RW1C	0x0	This bit is set to 1 if the number of unread data in FIFO equals to FIFO_CMP. Interrupt is generated if CMP_INT_EN=1. It can be cleared by writing 0x1 to this bit.
2	CNT_INT_FLAG	RW1C	0x0	This bit is set to 1 if RLC_COUNT=127. Interrupt is generated if CNT_INT_EN=1. It can be cleared by writing 0x1 to this bit.
1	PEDGE_INT_FLAG	RW1C	0x0	This bit is set to 1 if positive edge of the input infrared signal is detected. Interrupt is generated if PEDGE_INT_EN=1. It can be cleared by writing 0x1 to this bit.
0	NEDGE_INT	RW1C	0x0	This bit is set to 1 if negtive edge of the input infrared signal

Offset: 0x20				
Bits	**Field**	**Type**	**Reset**	**Description**
	_FLAG			is detected. Interrupt is generated if NEDGE_INT_EN=1. It can be cleared by writing 0x1 to this bit

16.7 PWM

16.7.1 Introduction

K1 contains 20 Pulse-Width Modulation (PWM) channels labeled as PWMx where x=[0,19].

Each PWM channel operates independently with its own configuration registers and generates an output PWM signal on a multi-function pin.

Each PWM channel allows controlling over both the leading-edge timing and the trailing-edge timing of its output signal.

The timing of each PWM channel can be set to run continuously or be adjusted dynamically to meet the change of requirements.

The power-saving mode allows stopping the internal clock of a PWM channel (PSCLK_PWM), resulting to a constant high or low state of the output signal of that PWM channel (PWM_OUT), thus saving power when the output signal of that PWM channel is not needed.

16.7.2 Features

- Support for 50% duty-cycle ranging from 198.4Hz to 6.5MHz (additional duty-cycle options depend on the choice of the preferred frequency)
- Enhanced period time controlled through 6-bit clock divider and 10-bit period time counter
- 15-bit pulse counter control

16.7.3 Register Description

Note. The base address of PWMn (n=1, 2, … , 20) registers is 0xD401A000 with a stride of 0x400.

16.7.3.1 PWM_CRX REGISTER

PWM Control registers.

These registers configure the behavior characteristics of the PWM shutdown response and the divisor for the input clocks to the PWM control unit that configures the frequency of the scaled counter clock.

Offset: 0x0+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:9	Reserved	RO	0x0	Reserved for future use.
8	PWM_OUTCNTen	RW	0x0	PWM Out Counter Register enable. 0=disable, 1=enable.
7	Reserved	RO	0	Reserved for future use.
6	Pulse Width Modulator Shutdown Mode	RW	0x0	0=Graceful shutdown of PWM when the SoC stops the clock to the PWM. 1=Abrupt shutdown of PWM when the SoC stops the clocks to the PWM.
5:0	Prescale	RW	0x0	The scaled counter clock frequency is: PSCLK_PWM/(PRESCALE+1)

16.7.3.2 PWM_DCR REGISTER

PWM Duty Cycle registers.

These registers configure the duty cycle of the corresponding PWM_OUT signals.

Offset: 0x4+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:11	Reserved	RO	0x0	Reserved for future use.
10	Full Duty Cycle	RW	0x0	0=PWM_OUT is determined by the <Duty Cycle of PWM_OUT> value. 1=PWM_OUT is continuously asserted
9:0	Duty Cycle of PWM_OUT	RW	0x0	0=PWM_OUT is continuously de-asserted. 1=PWM_OUT is high for the number of 12 MHz clock periods equal to this field (<PRESCALE> in PWM Control Registers + 1). If <Full Duty Cycle> is set, this filed has no effect on the ouptut of PWM.

16.7.3.3 PWM_PCR REGISTER

PWM Period Control registers.

These registers configure the cycle time of the corresponding PWM_OUT signals.

If this register is cleared, the PWM_OUT signal maintains in a high state.

Offset: 0x8+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:10	Reserved	RO	0x0	Reserved for future use.
9:0	Perios Value	RW	0x4	The value of scaled clock cycles per cycle of PWM_OUT plus one. If all zeros are written to this register, the signal remains high.

16.7.3.4 PWM_OUTCNT REGISTER

PWM Output Counter registers.

Offset: 0x10+(n-1)*0x400				
Bits	**Field**	**Type**	**Reset**	**Description**
31:16	Reserved	RO	0x0	Reserved for future use.
15:0	Counter Value	RW	0x0	The value of pwm out pulse number.

SPACEMIT

Web

WeChat

Room 701, Building B, Future Sky Center, Wuchang Street,
Yuhang District, Hangzhou City, Zhejiang Province, China