DDR4 Chiplet Technical Reference Manual

FLC Technology Group
v 1.2
10/10/24

Table of Contents

List of Figures

List of Tables

Preface

Note for Readers

This document is under development. More details are being added.

About the Manual

FLC Technology Group DDR4 Chiplet Technical Reference Manual. This manual contains documentation for the DDR4 Chiplet, the programmer's model, instruction set, registers, memory map, debug support and many other features.

Documentation Dependencies

The technical reference manual details about the FLC Technology Group DDR4 Chiplet alongside its dependencies on third party IPs from Cadence, Synopsys and RISC-V to align the functionality.

Audience

This manual is intended to help system designers, system integrators, verification engineers, and software programmers who are using DDR4 Chiplet.

Document Organization

This technical reference manual contains the following chapters:

Note for Readers

<TBD>

Web Resources

<TBD>

Customer Support

<TBD>

Revision History

Revision	Date	Description
1.2	10-10-2024	Updated the Preface and document format
1.1	09-09-2024	3.16 updated, 3 tables added. 6.1 Summary of Registers Cross-reference link added.
1.0	08-16-2024	Initial Release

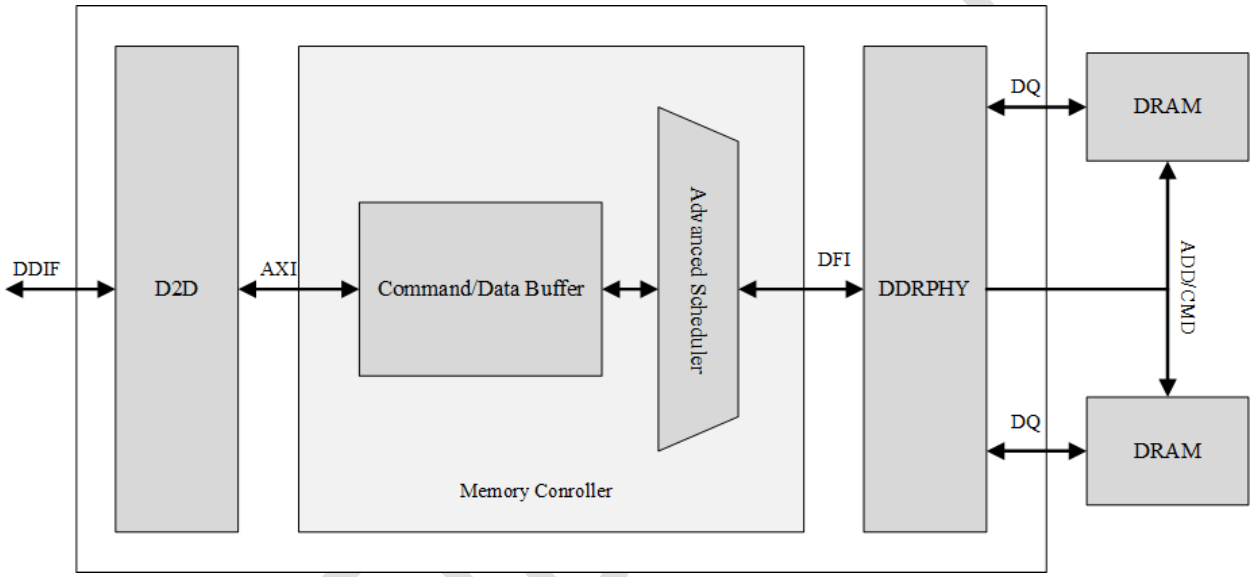
Chapter 1 Introduction

This chapter describes the processor, its application domain and its significant features. The chapter is divided into following sections:

1.1 Overview

The DDR4 chiplet is a quick solution on DDR4 memory subsystem. It can provide and optimize high bandwidth, high reliability, and low power DDR4 access solution. The chiplet involves D2D, DDR4 memory controller, and DDRPHY inside with DFI compliant interface as shown in the Figure 1.

Figure 1 DDR4 Chiplet System Level Block Diagram



1.2 Features

- Supports AMBA AXI4 protocol; bandwidth: 32 GB/s (256-bits data bus at 1 GHz).
- Supports DDR4 and DIMMs; bandwidth: 25.6 GB/s (64-bits DQ + 8-bits ECC at 3200 MT/s).
- Supports up to 4 DDR ranks with each rank's capacity up to 32 GB.
- Designed in TSMC 12 nm with balanced power and performance.
- End-to-end QoS arbitration for improved latency and bandwidth for critical reads.
- Hardware full training/period training/temperature and voltage drift compensation.
- Supports sideband SECDED ECC, x4 device Chipkill ECC, write CRC, CA parity, memory scrubbing, and retry buffer.
- Independent low frequency clock for refresh counter to avoid converting the refresh counter when performing frequency change.

1.2.1 D2D IP Features

- Supports AXI4 interface with 256-bit at 1 GHz over an industry standard Bunch-of-Wires (BoW) physical and electrical interface.

- Support AXI5-lite interface with 64-bits over an industry standard BoW physical and electrical interface.
- Supports proprietary virtual wire up to 1024 over an industry standard BoW physical and electrical interface.
- Supports PHY connecting to BoW slice pairs for 64 GB/s bandwidth in each direction.
- APB completer interface for local register access.

1.2.2 MC IP Features

- **AXI interface features:**
  - AXI4 protocol. Synchronous and asynchronous mode for AXI and MC clock domain.
  - One AXI port, 256-bits data width, up to 256 beats of burst length.
  - Supports narrow data width transfers.
  - 4 KB burst boundary limitation.
  - Aligned/unaligned start address transfers.
  - Incremental/wrapping burst type.
  - 4 KB outstanding write transactions, > 4 KB outstanding read transactions.
  - Up to 8 exclusive monitors.
  - 4 level read QoS priority (0 to 3) of which QoS = 3 is taken as critical read.
  - Up to 9 TrustZone with programable address range and access permission.
  - RAW guarantee coherency when read received after write response.
- **DDR4 features:**
  - Supports DDR4, DDR4-UDIMM, DDR4-SODIMM, DDR4-RDIMM, and DDR4-LRDIMM.
  - Up to 3200 MT/s, 64-bits data width, 1, 2, 3, or 4 ranks and each rank up to 32 GB.
  - Auto pre-charge.
  - CS to command latency.
  - CA parity.
  - Write CRC.
  - hPPR and sPPR.
  - Write data mask, write DBI, read DBI (write data mask and write DBI are not able to enable together).
  - PDA mode.
  - Control gear down.
  - Self-refresh, active power-down, pre-charged power-down, and maximum power saving mode.
  - Input frequency change and DLL on/off switching.
  - Up to 8 pull in/postponed refreshes.
  - MPR read and write.
  - Programmable ODT.
  - Programmable `1nCK/2nCK` Write Preamble; Programmable `1nCK/2nCK` Read Preamble.

    - Shorter `tCCD_s/tRRD_s/tWTR_s` timing requirement between different bank groups than same bank groups.
    - Burst length 8 and read burst CA can start from 0 or 4 aligned address.
    - 2-stacks 3DS DDR4.
    - Address mirroring DIMMs.
    - DQ swap.
- **DFI interface features:**
    - DFI 5.0 protocol.
    - 1:1 frequency ratio under 2666 MT/s; 1:2 frequency ratio for all speed bins up to 3200 MT/s.
    - Supports `dfi_init*`, `dfi_ctrlupd*`, `dfi_phyupd*`, and `dfi_phymstr*` interfaces.
    - Low-power interface.
- 64 reads + 64 writes lookup window for DDR utilization optimization.
- Command passes through latency as low as 5 clock cycles; Read data latency as low as 4 clock cycles.
- Flexible address mapping logic to SDRAM Rank, CID, Bank Group, Bank, Row, and Column address. DDR4 address segments can be mapped to adjacent or discrete master address bits.
- DDR4 Channel, CID, Bank Group, and Bank addresses support address hashing. The hashing bits are fully programmable by separated `hash_sel` CSRs.
- Delay writes until the programable threshold is reached for better performance.
- Out-of-order read execution for SDRAM bandwidth.
- Advanced read/write arbitration and transaction order optimization as per QoS and bank/page status.
- Separated request queue for critical reads (QoS = 3).
- Starvation prevention. Programable aging counter-based starvation prevention for low priority read requests to prevent deadlock.
- Supports write data coalescing and read data forwarding to reduce RMW and DDR read/write switch penalty.
- Programable page policy between leave page open after access or auto pre-charge when there are no further accesses available.
- Explicit SDRAM mode register updates under software control.
- Support auto power saving (self-refresh and power-down) and global/component clock gating.
- Command, address, and data pins hold last state after each command. Data pins go to tristate for power saving.
- SEDDEC ECC and x4 device Chipkill level ECC with extra 8 parallel check bits on SDRAM bus.
- Automatic logging of both correctable and uncorrectable errors. The recorded error information can be specified to Column address and nibble index.
- Programmable interrupt for ECC error.
- Memory scrubbing when master transaction is idle.
- Up to 16 slices of flop-based Shadow Memory Repair, which can be used to replace bad memory cells in DDR data width granularity.
- SRAM-based Memory Repair Capability:
    - Repair up to 32K x4 / 16K x8 random faults.

    - Repair up to 32 word-line (1KB) faults.
    - Scrubs entire memory space at boot time and in mission mode.
    - Creates fast, compact searchable fault database.
- Fault databases can be saved and restored for minimal boot time impact. Highly efficient read-modify-write when byte enables are used with ECC enabled or SDRAM DM disabled.
- Retry buffer for DDR4 CA parity and CRC error.

1.2.3 DDR PHY IP Features

- High-performance DDR PHY supports data rates up to 3200 Mbps.
- DFI 4.0-compatible controller interfaces to the memory controller.
- Multiple DDR4 standards are supported in one PHY (selection of DDR type is software configurable).
- PHY utility block supports in-system calibration, data training, and testability features.
- Supports up to 4 trained states/frequencies with < 5µs switching time.
- IO receiver decision feedback equalization and driver feed-forward equalization.
- VT compensated delay lines for DQS centering, read/write leveling, and per bit deskew.
- Optional DDR PHY hardening, Signal and Power Integrity (SIPI), and Sub-systems services are available.

1.2.4 Micro Controller Subsystem Features

- Processor: SiFive E21
- Processor speed: 200 MHz
- Operating mode: Machine mode (privilege mode) and user mode
- Physical memory protection: Four regions (configurable up to 16). PMP is equivalent to ARM's PMU
- Debug interface: JTAG
- Peripheral bus: AHB and APB for embedded peripherals/memory/CSR.
- Interrupt: CLIC (63 interrupt sources). No MSI/AIA/IMSIC and PLIC
- SRAM (TCM): 256 KB, 2 banks (2 × 128 KB)
- Mask ROM: 32 KB
- XIP NOR serial flash: x4 data bus with SRAM cache
- OTP: 1K x 32-bits
- SPI (x2): master and device modes
- I2C (x1)
- Timer: Local timer (x2) and processor embedded timer. Watchdog timer.
- UART (x1): Console terminal and firmware download
- GPIO: 20 pins
- DMA: 4 channels
- Security engine: AES and SHA
- I3C Master/Slave
- AXI4 Traffic generator

- Run-time temperature, voltage, and performance monitoring.

Chapter 2 Signal Description

This chapter explains the various Signals present in the processor and their usage.

This chapter is divided into the following sections:

2.1 Pin List

The DDR4 Chiplet (Goldfinch) signals are listed and described in Table 1. The power and ground pins are listed and described in Table 2.

Table 1 Goldfinch Pin List

Type	Pin Name	Category	Comment
Input	CB_nRST	Reset	Functional reset for chiplet, internal pull-up
input	CB_nPOR	Reset	Power on reset, internal pull-up
input	CORERSTn	Reset	CPU subsystem reset, internal pull-up
input	OSCCLK	Clock	25 MHz clock to main PLL
input	ref_100m_n	Clock	100Mhz Diff n clock to D2D PLL
input	ref_100m_p	Clock	100Mhz Diff p clock to D2D PLL
output	PLL_LOCKED	Status	1= SOC PLL locked; clock ready
inout	PAD_GPIO_[19:0]	GPIO	General purpose IO, Some GPIO may need 3.3V/5V tolerant (I2C) Refer to Table 8 GPIO Pin Multiplex for embedded peripheral functions muxes
output	sf_clk	SF-NOR	Serial flash clock
output	sf_cs	SF-NOR	Serial flash chip select
inout	sf_io [3:0]	SF-NOR	Serial flash data, internal pull-up controlled by sf_io_pullup_en in register PMU_CTRL [15:12]
input	boot_xip	Boot Option	0 = internal ROM 1 = serial flash
inout	i3c_master_scl	I3C	I3C master clock
inout	i3c_master_sda	I3C	I3C master data
input	i3c_slave_scl	I3C	I3C slave clock
inout	i3c_slave_sda	I3C	I3C slave data
inout	DDR4_RDQSB [8:0]	DDR4	Data Strobe
inout	DDR4_DQSB [8:0]	DDR4	Data Strobe
inout	DDR4_DQS [8:0]	DDR4	Data Strobe
inout	DDR4_DQM [8:0]	DDR4	Data Mask
inout	DDR4_DQ [71:0]	DDR4	Data Input/Output
output	DDR4_CKE [3:0]	DDR4	Clock Enable
output	DDR4_BG [1:0]	DDR4	Bank Group Inputs
output	DDR4_ACTN	DDR4	Activation Command Input
output	DDR4_ADDR [17:0]	DDR4	Address Inputs ADDR [16] = RAS_n ADDR [15] = CAS_n ADDR [14] = WE_n
output	DDR4_CLK [3:0]	DDR4	Clock
output	DDR4_CLKN [3:0]	DDR4	Clock

Type	Pin Name	Category	Comment
output	DDR4_BA [1:0]	DDR4	Bank Address Inputs DDR4_BA [2] removed, use DDR4_BG [0]
output	DDR4_PAR	DDR4	Command and Address Parity Input
output	DDR4_CID [2:0]	DDR4	Chip ID
output	DDR4_CSN [3:0]	DDR4	Chip Select
output	DDR4_ODT [3:0]	DDR4	On Die Termination
output	DDR4_RESET_N	DDR4	Active Low Asynchronous Reset
inout	DDR4_ALERT_N	DDR4	Alert
inout	DDR4_VREFD	DDR4	Voltage reference for receivers and analog test point for debug
inout	BP_M_TEST	DDR4	Test
input	BP_ZN_SENSE	DDR4	Sense bump for external resistor
output	BP_ZN	DDR4	Calibration external reference resistor
inout	d2d_ddr_tx0_d [15:0]	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_clk_tx0_p	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_clk_tx0_n	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_rx0_d [15:0]	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_clk_rx0_p	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_clk_rx0_n	D2D	D2D slave interface for other master chiplet
output	d2d_ddr_tx0_aux	D2D	D2D slave interface for other master chiplet
output	d2d_ddr_tx0_fec	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_rx0_aux	D2D	D2D slave interface for other master chiplet
inout	d2d_ddr_rx0_fec	D2D	D2D slave interface for other master chiplet
inout	ts_an_io [1:0]	TSensor	Temperature sensor Analogue Access[1]
inout	vs_an_io [1:0]	VSensor	Voltage sensor Analogue Access[1]
input	nTRST_E21	JTAG	E21 JTAG reset[2], internal pull-up
input	TCK_E21	JTAG	E21 JTAG clock[2], internal pull-down
input	TMS_E21	JTAG	E21 JTAG mode select[2], internal pull-up
input	TDI_E21	JTAG	E21 JTAG data input[2], internal pull-up
output	TDO_E21	JTAG	E21 JTAG data output[2]
input	nTRST	JTAG	DDR/D2D PHY/TDR2APB JTAG reset[2,3], internal pull-up
input	TCK	JTAG	DDR/D2D PHY/TDR2APB JTAG clock[2,3], internal pull-down
input	TMS	JTAG	DDR/D2D PHY/TDR2APB JTAG mode select[2,3], internal pull-up
input	TDI	JTAG	DDR/D2D PHY/TDR2APB JTAG data input[2,3], internal pull-up
output	TDO	JTAG	DDR/D2D PHY/TDR2APB JTAG data output[2,3]
input	BSCAN_TRST	BSCAN	BSCAN reset[2,4]
input	BSCAN_TCK	BSCAN	BSCAN clock[2,4]
input	BSCAN_TMS	BSCAN	BSCAN mode select[2,4]

Type	Pin Name	Category	Comment
input	BSCAN_TDI	BSCAN	BSCAN data input[2,4]
output	BSCAN_TDO	BSCAN	BSCAN data output[2,4]

1. If access to the analog test mode is not needed then *_an_io [1:0] pins should not be connected and should be left floating.
2. If access to the test mode is not needed then these pins can be left floating.
3. For JTAG selection:
   {GPIO_I [6], GPIO_I [7]} = 2'b00, select D2D PHY JTAG;
   {GPIO_I [6], GPIO_I [7]} = 2'b01, select DDR PHY JTAG;
   {GPIO_I [6], GPIO_I [7]} = 2'b1X, select TDR2APB JTAG.
4. For Scan mode operation:
   GPIO [0] = Edt_update;
   GPIO [1] = Scan_clock;
   GPIO [2] = Scan_enable;
   GPIO [3:10] = Edt_channel_in;
   GPIO [11:18] = Edt_channel_out.

Table 2 Goldfinch Power and Ground pins

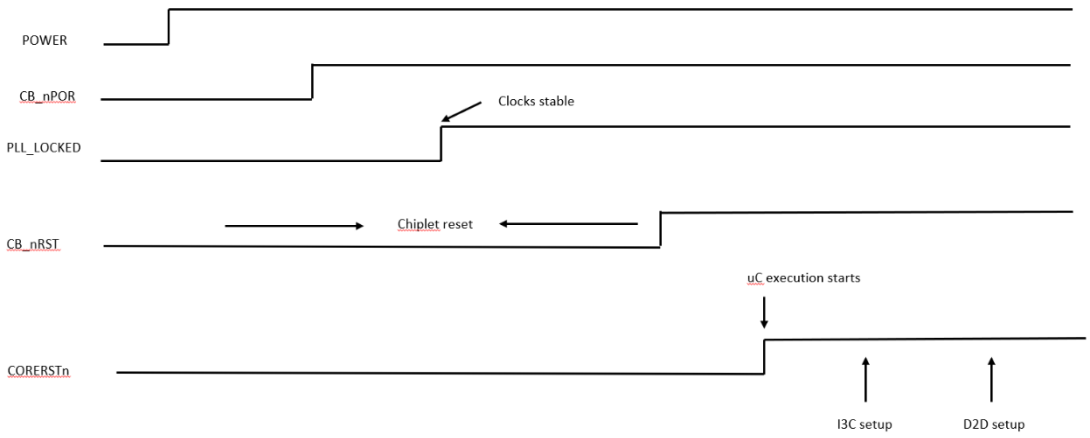
Type	Pin Name	Comment
Power	vdd_io	IO power, 1.8 V +/- 10%
Ground	vss_io	IO combined ground
Power	vdd_core	Core power, 0.8 V +/- 10%
Ground	vss_core	Core combined ground
Power	gfh_pll_vddhv	Main PLL Analog Supply Voltage, 1.8 V +/- 10%
Power	gfh_pll_vddpost	Main PLL Digital Supply Voltage, 0.8 V +/- 10%
Power	gfh_pll_vddref	Main PLL Digital Supply Voltage, 0.8 V +/- 10%
Power	d2d_pll_vddhv	D2D PLL Analog Supply Voltage, 1.8 V +/- 10%
Power	d2d_pll_vddpost	D2D PLL Digital Supply Voltage, 0.8 V +/- 10%
Power	d2d_pll_vddref	D2D PLL Digital Supply Voltage, 0.8 V +/- 10%
Power	DDR4_VDDQ	DDR4 PHY VDDQ Supply Voltage, 1.2 V +/- 10%
Power	DDR4_VAA	DDR4 PHY PLL Supply Voltage, 1.8 V +/- 10%
Power	d2d_vddq_s	D2D PHY VDDQ Supply Voltage, 0.3/0.5/0.825 V
Power	d2d_vddclk_s	D2D PHY VDDCLK Supply Voltage, 0.8 V +/- 10%
Power	d2d_vdda_s	D2D PHY VDDA Supply Voltage, 1.2 V +/- 10%
Power	otp_vdd2	OTP power, 1.8 V +/- 10%
Power	VDDA_TS	Temperature Sensor Analog Supply Voltage, 1.8V +/- 10%
Power	VDDA_VS	Voltage Monitor Analog Supply Voltage, 1.8V +/- 10%

2.2 Chip Reset and Clock Ports

- `CB_nPOR`: Active low, power on reset. Reset PLL and clock circuitry.
- `CB_nRST`: Active low, global chip reset.
- `CORERSTn`: Active low, reset embedded µC.
- External reset chip required to detect stable voltage level and reset timing.
- 25 MHz single ended reference clock for Goldfinch PLL
- 100 MHz reference (CML/differential) for D2D high speed PLL

Figure 2 illustrates the Reset and Clock timing diagram.

Figure 2 Reset and Clock



2.3 Initialization Sequence

There are two E21 FW memory boot options controlled by `boot_xip` pin. If `boot_xip=1'b0`, E21 will boot from internal ROM, if `boot_xip=1'b1`, E21 will boot from external serial flash.

1. Power applied and reference clocks ready.
2. µC fetches initialization code from ROM or SF flash.
3. µC set GPIO to select functions. Refer to 3.10.3 GPIO Pin.
4. µC programs I3C slave initialization and set `I3C_ready` (GPIO [18]), goldfinch I3C slave is ready after this step.
5. µC stayed in WFI (Wait for Interrupt) loop.
6. Master chiplet uses I3C to initial goldfinch slave D2D, goldfinch D2D is ready after this step.
7. Master chiplet use D2D to goldfinch slave D2D/AXI5-Lite for DDR4 MC/PHY initialization.

2.4 DDR4 Clock Setting

By changing the Goldfinch PLL setting (pll_ctl [95:0], 0x2600_0108~0x2600_0100) and DDR clock ratio setting (0x2600_0004 [31:30]), different speed of DDR4 can be supported. Table 3 lists the DDR4 clock setting.

Table 3 DDR4 Clock Setting

	Required input clocks		
		1:2 frequency ratio	1:1 frequency ratio
	PHY DfiClk	PHY DfiCtlClk = MC Clk	PHY DfiCtlClk = MC Clk
DDR4-3200	800 MHz	800 MHz	-
DDR4-2933	733.3 MHz	733.3 MHz	-
DDR4-2666	666.5 MHz	666.5 MHz	1333 MHz
DDR4-2400	600 MHz	600 MHz	1200 MHz
DDR4-2133	533.3 MHz	533.3 MHz	1066.5 MHz
DDR4-1866	466.5 MHz	466.5 MHz	933 MHz
DDR4-1600	400 MHz	400 MHz	800 MHz

- DDR4-3200, 1:2 frequency ratio: 0x2600_0004[31:30] = 2'b01, `pll_ctl` [95:0] = 96'h3000_0000_043c_2801_0000_0011. Default.
- DDR4-2666, 1:2 frequency ratio: 0x2600_0004[31:30] = 2'b01, `pll_ctl` [95:0] = 96'h3000_0000_053c_2801_0000_0011.
- DDR4-2666, 1:1 frequency ratio: 0x2600_0004[31:30] = 2'b00, `pll_ctl` [95:0] = 96'h3000_0000_023c_2801_0000_0011.
- DDR4-2400, 1:2 frequency ratio: 0x2600_0004[31:30] = 2'b01, `pll_ctl` [95:0] = 96'h3000_0000_075c_3c01_0000_0011.
- DDR4-2400, 1:1 frequency ratio: 0x2600_0004[31:30] = 2'b00, `pll_ctl` [95:0] = 96'h3000_0000_045c_3c01_0000_0011.
- DDR4-2133, 1:2 frequency ratio: 0x2600_0004[31:30] = 2'b01, `pll_ctl` [95:0] = 96'h3000_0000_063c_2801_0000_0011.
- DDR4-2133, 1:1 frequency ratio: 0x2600_0004[31:30] = 2'b00, `pll_ctl` [95:0] = 96'h3c.
- DDR4-1600, 1:2 frequency ratio: 0x2600_0004[31:30] = 2'b01, `pll_ctl` [95:0] = 96'h3000_0000_073c_2801_0000_0011.
- DDR4-1600, 1:1 frequency ratio: 0x2600_0004[31:30] = 2'b00, `pll_ctl` [95:0] = 96'h3000_0000_043c_2801_0000_0011.

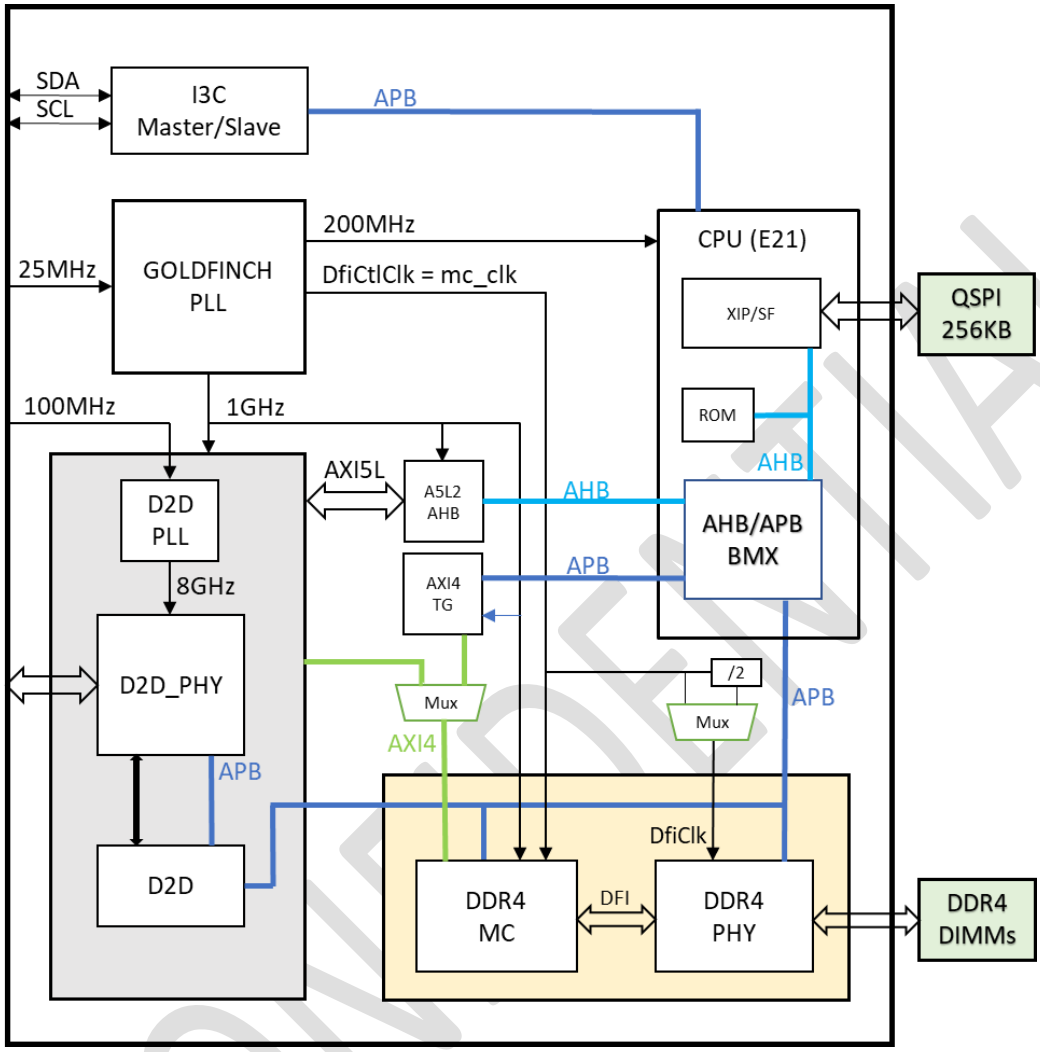
Chapter 3 Chiplet Modules

This chapter explains the various functional modules present within the chiplet.

This chapter is divided into following sections:

Figure 3 illustrates the chiplet modules present inside the Goldfinch chiplet.

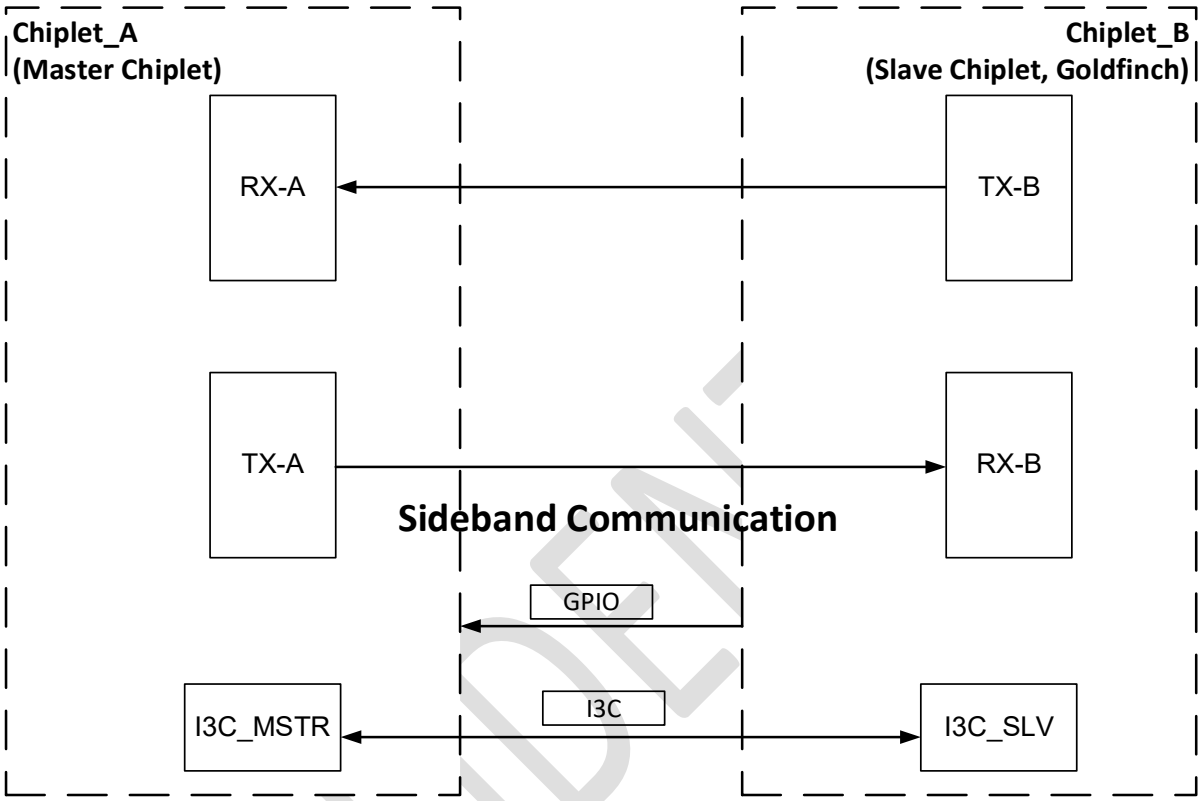
Figure 3 Internal Block Diagram



3.1 D2D Module

Figure 4 illustrates the D2D module.

Figure 4 D2D Module



3.1.1 Prerequisites

- APB reset is already de-asserted and APB clock is generated. Link layer requires a clock before PHY clock is up.
- Goldfinch GPIO[18] must be high indicating `Chiplet_B` (slave chiplet) is ready.
- I3C slave enablement (2 SCL + 1 stop).

3.1.2 Initialization

3.1.2.1 TX PHY Init

- TX Init must happen before RX Init.
    a. `Chiplet_A`: To initiate TX PHY Init for TX-A.
    b. `Chiplet_A` to `Chiplet_B`: Send `I3C_D2D_BU_MSG_0` (I3C private write).
    c. `Chiplet_B`: E21 to process `I3C_D2D_BU_MSG_0` (process opcode in RX buffer).
    d. `Chiplet_B`: E21 to load `I3C_D2D_BU_MSG_RESP_0` into TX buffer and toggle GPIO after completion.
    e. `Chiplet_A`: Receive/Process GPIO toggle.

f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C Private Read) to get response message.
g. Chiplet_A: Ensure TX PHY Init of TX-A has completed, and TX PHY Init of TX-B has successfully completed.
h. I3C_D2D_BU_MSG_0 = Request to initiate TX PHY Init for TX-B.
i. I3C_D2D_BU_MSG_RESP_0 = Response message for completion/failure – TX PHY Init.
j. I3C_12B_RD_MSG = Read request (get response message)

3.1.2.2 RX PHY Init

- Requires clock from TX.
  a. Chiplet_A: To initiate RX PHY Init for RX-A.
  b. Chiplet_A to Chiplet_B: Send I3C_D2D_BU_MSG_1 (I3C private write).
  c. Chiplet_B: E21 to process I3C_D2D_BU_MSG_1 (process opcode in RX buffer).
  d. Chiplet_B: E21 to load I3C_D2D_BU_MSG_RESP_1 into TX buffer and toggle GPIO after completion.
  e. Chiplet_A: Receive/Process GPIO toggle.
  f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C Private Read) to get response message.
  g. Chiplet_A: Ensure RX PHY Init of RX-A has completed, and RX PHY Init of RX-B has successfully completed.
  h. I3C_D2D_BU_MSG_1 = Request to initiate RX PHY Init for RX-B
  i. I3C_D2D_BU_MSG_RESP_1 = Response message for completion/failure – RX PHY Init
  j. I3C_12B_RD_MSG = Read request (get response message)

3.1.3 Link Training and Word Align

3.1.3.1 RX Train Mode

- RX must go into train mode first and should be ready to receive training data from TX.
  a. Chiplet_A: To initiate moving to RX train mode state for RX-A.
  b. Chiplet_A to Chiplet_B: Send I3C_D2D_BU_MSG_2 (I3C private write).
  c. Chiplet_B: E21 to process I3C_D2D_BU_MSG_2 (process opcode in RX buffer).
  d. Chiplet_B: E21 to load I3C_D2D_BU_MSG_RESP_2 into TX buffer and toggle GPIO after completion.
  e. Chiplet_A: Receive/Process GPIO toggle.
  f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C Private Read) to get response message.
  g. Chiplet_A: Ensure both RX-A and RX-B have successfully moved to train mode state.
  h. I3C_D2D_BU_MSG_2 = Request to move to RX train mode state for RX-B
  i. I3C_D2D_BU_MSG_RESP_2 = Response message for completion/failure – Move to RX train mode state
  j. I3C_12B_RD_MSG = Read request (get response message)

3.1.3.2 TX Train Mode

- TX needs to start sending training data to RX.
    a. `Chiplet_A`: To initiate moving to TX train mode state for TX-A.
    b. `Chiplet_A` to `Chiplet_B`: Send `I3C_D2D_BU_MSG_3` (I3C private write).
    c. `Chiplet_B`: E21 to process `I3C_D2D_BU_MSG_3` (process opcode in RX buffer).
    d. `Chiplet_B`: E21 to load `I3C_D2D_BU_MSG_RESP_3` into TX buffer and toggle GPIO after completion.
    e. `Chiplet_A`: Receive/Process GPIO toggle.
    f. `Chiplet_A` to `Chiplet_B`: Send `I3C_12B_RD_MSG` (I3C private read) to get response message.
    g. `Chiplet_A`: Ensure both TX-A and TX-B have successfully moved to train mode state.
    h. `I3C_D2D_BU_MSG_3` = Request to move to TX train mode state for TX-B
    i. `I3C_D2D_BU_MSG_RESP_3` = Response message for completion/failure – Move to TX train mode state
    j. `I3C_12B_RD_MSG` = Read request (get response message)

3.1.3.3 Fix PHY Alignment (both intra-slice and inter-slice alignment)

- Trigger sample for intra-slice phase alignment. Calculate and fix alignment in the PHY.
- Trigger sample for inter-slice alignment (fragment). Calculate and fix alignment in the PHY.
    a. `Chiplet_A`: To initiate alignment between TX-B and RX-A.
    b. `Chiplet_A` to `Chiplet_B`: Send `I3C_D2D_BU_MSG_4` (I3C private write).
    c. `Chiplet_B`: E21 to process `I3C_D2D_BU_MSG_4` (process opcode in RX buffer).
    d. `Chiplet_B`: E21 to load `I3C_D2D_BU_MSG_RESP_4` into TX buffer and toggle GPIO after completion.
    e. `Chiplet_A`: Receive/Process GPIO toggle.
    f. `Chiplet_A` to `Chiplet_B`: Send `I3C_12B_RD_MSG` (I3C private read) to get response message.
    g. `Chiplet_A`: Ensure TX-B/RX-A and TX-A/RX-B has successfully completed alignment.
    h. `I3C_D2D_BU_MSG_4` = Request to start alignment between TX-A and RX-B
    i. `I3C_D2D_BU_MSG_RESP_4` = Response message for completion/failure – slice alignment
    j. `I3C_12B_RD_MSG` = Read request (get response message)

3.1.4 TX IDLE and IDLE check

- Move to `TX_IDLE`, reset TX link, and ensure flit same registers are all zeros (IDLE check).
    a. `Chiplet_A`: To initiate moving to `TX_IDLE`, TX link reset and IDLE check for TX-A.

b. Chiplet_A to Chiplet_B: Send I3C_D2D_BU_MSG_5 (I3C private write).
c. Chiplet_B: E21 to process I3C_D2D_BU_MSG_5 (process opcode in RX buffer).
d. Chiplet_B: E21 to load I3C_D2D_BU_MSG_RESP_5 into TX buffer and toggle GPIO after completion.
e. Chiplet_A: Receive/Process GPIO toggle.
f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C private read) to get response message.
g. Chiplet_A: Ensure both TX-A and TX-B have successfully moved to TX_IDLE and passed the IDLE check.
h. I3C_D2D_BU_MSG_5 = Request to move to TX IDLE mode state, TX link reset, and IDLE check for TX-B.
i. I3C_D2D_BU_MSG_RESP_5 = Response message for completion/failure – TX IDLE and IDLE check.
j. I3C_12B_RD_MSG = Read request (get response message)

3.1.5 RX wait and RX link reset

a. Chiplet_A: To initiate moving to RX_WAIT RX-A.
b. Chiplet_A to Chiplet_B: Send I3C_D2D_BU_MSG_6 (I3C private write)
c. Chiplet_B: E21 to process I3C_D2D_BU_MSG_6 (process opcode in RX buffer)
d. Chiplet_B: E21 to load I3C_D2D_BU_MSG_RESP_6 into TX buffer and toggle GPIO after completion.
e. Chiplet_A: Receive/Process GPIO toggle.
f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C Private Read) to get response message.
g. Chiplet_A: Ensure both RX-A and RX-B have successfully moved to RX_WAIT and RX link reset.
h. I3C_D2D_BU_MSG_6 = Request to move to RX WAIT Mode state, RX link reset for RX-B
i. I3C_D2D_BU_MSG_RESP_6 = Response message for completion/failure – RX WAIT and RX link reset
j. I3C_12B_RD_MSG = Read request (get response message)

3.1.6 TX_RUN and poll for RX_RUN

- TX_RUN and poll for RX_RUN and set RxVirtualWireDisable00 to zero.

a. Chiplet_A: To initiate moving to TX_RUN for TX-A, poll for RX-A to be in RUN MODE and set RxVirtualWireDisable00 to zero for RX-A.
b. Chiplet_A to Chiplet_B: Send I3C_D2D_BU_MSG_7 (I3C private write).
c. Chiplet_B: E21 to process I3C_D2D_BU_MSG_7 (process opcode in RX buffer).
d. Chiplet_B: E21 to load I3C_D2D_BU_MSG_RESP_7 into TX buffer and toggle GPIO after completion.
e. Chiplet_A: Receive/Process GPIO toggle.

f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C Private Read) to get response message.
g. Chiplet_A: Ensure both TX-A and TX-B have successfully moved to TX_IDLE and passed the IDLE check.
h. I3C_D2D_BU_MSG_7 = Request to move to TX RUN mode state for TX-B, poll for RX-B to be in RUN mode and set RxVirtualWireDisable00 to zero for RX-B
i. I3C_D2D_BU_MSG_RESP_7 = Response message for completion/failure – TX RUN mode state and poll for RX RUN mode state
j. I3C_12B_RD_MSG = Read request (get response message)

3.1.7 Set TX Virtual Wire Disable00 to Zero

a. Chiplet_A: To set TxVirtualWireDisable00 to zero for TX-A.
b. Chiplet_A to Chiplet_B: Send I3C_D2D_BU_MSG_8 (I3C private write).
c. Chiplet_B: E21 to process I3C_D2D_BU_MSG_8 (process opcode in RX buffer).
d. Chiplet_B: E21 to load I3C_D2D_BU_MSG_RESP_8 into TX buffer and toggle GPIO after completion.
e. Chiplet_A: Receive/Process GPIO toggle.
f. Chiplet_A to Chiplet_B: Send I3C_12B_RD_MSG (I3C Private Read) to get response message.
g. Chiplet_A: Ensure both TX-A and TX-B have set TxVirtualWireDisable to zero.
h. I3C_D2D_BU_MSG_8 = Request to set TxVirtualWireDisable to zero for TX-B
i. I3C_D2D_BU_MSG_RESP_8 = Response message for completion/failure – set TxVirtualWireDisable to zero
j. I3C_12B_RD_MSG = Read request (get response message)

3.2 D2D PHY Module

For details on D2D PHY, See BoW PHY IP Integration Guide.

3.3 DDR4 Micro Controller

DDR4 MC (Memory Controller) is interfaced between D2D and DDR PHY. There is an APB connection to µC subsystem for CSR programming.

µC access APB CSR (DDR4 MC Register) to adjust allocation of DDR region. Memory pooling among host clusters can also be accomplished.

DDR4 MC interface with D2D by AMBA AXI4 interface and DDR4 MC interface with DDR PHY by DFI interface. DDR4 MC translates AXI transactions from D2D to DDR PHY and DDR4 memory with optimized order for high bandwidth and maintains DDR in high performance and low-power state. It supports all kinds of RAS (Reliable and Security) features to make the data subsystem robust.

Refer to **Error! Reference source not found.** for the detailed MC supported features.

3.4 DDR4 PHY Module

DDR PHY interfaced to MC with DFI 5.0. DDR PHY supports 1:1 and 1:2 clocking modes. µC uses APB (CSR) to configure DDR PHY.

Each PHY includes an integer-N PLL that internally generates and distributes a full rate (that is, 2.133 GHz for 2133 MT/s, 6.4 GHz for 6400 MT/s, and so on.) clock. The PLL supports a wide range of output division ratios and the PLL's oscillator can be bypassed to simply forward 100 –133 MHz reference clock.

µC uses APB (CSR) to configure DDR PHY.

An embedded µC is put in DDR PHY. It executes firmware to do PHY initialization and training. Outside system, µC use APB bus to access internal µC IMEM and DMEM address range to fill instructions and data.

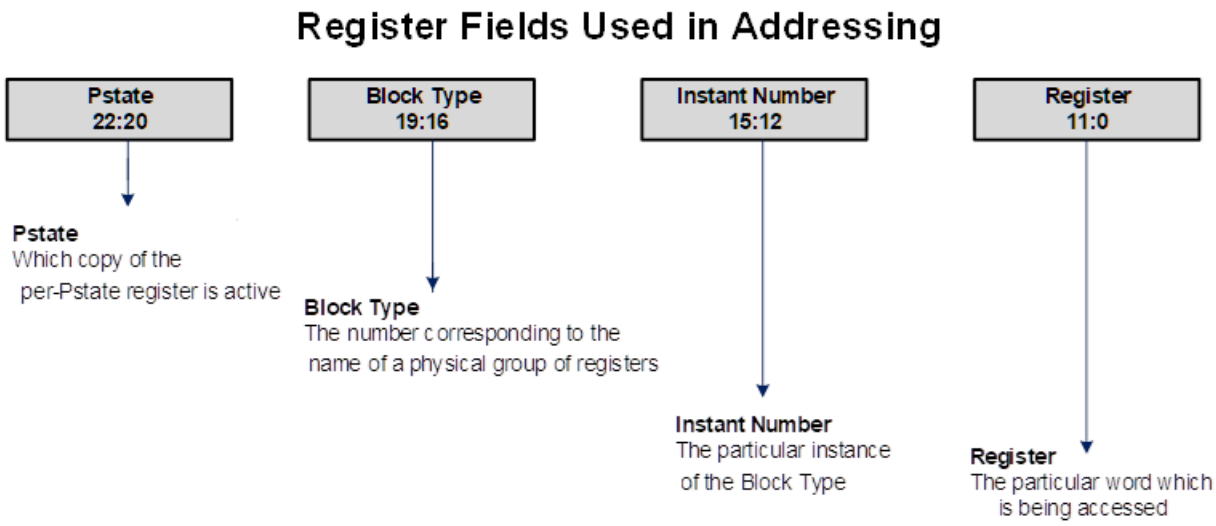
3.4.1 DDR4 PHY Register Map

DDR4 PHY register base address = 0x2200_0000

DDR4 PHY register APB address[31:0] = {9'b0, µC_address[24:2]}, this address is 32b aligned, data width = 16.

Figure 5 illustrates the DDR4 PHY register filed addresses.

Figure 5 DDR4 PHY Register field addresses



3.5 Micro Controller Subsystem

The micro controller subsystem design is based on SiFive E21 RISCV core and Buffalo lab peripheral IPs. Firmware is stored in on-chip mask ROM and off-chip XIP NOR flash. An AHB/APB bus matrix is used to connect CPU, memory, and IO.

After reset, the micro controller performs system initialization through AHB/APB bridge (refer to Table 20 µC Address Map). There are two local timers, a watchdog timer, and core

embedded timer for system timing control. Interrupt from E21's CLIC module with total of 63 maskable interrupt sources and 1 non-maskable NMI. (Refer to Table 21 µC IRQ Map)

Two on-core tightly couple memories (TIM), 128 KB each are included for fast execution.

3.5.1 Embedded IO Summary

Table 4 describes the embedded IO summary.

Table 4 Embedded IO Summary

Name	Mode	FIFO	DMA	Interrupt	Comment
SPI0	Master/Device	4	Yes	Intr[1]	Refer to 3.6 SPI Module
SPI1	Master/Device	4	Yes	Intr[2]	Refer to 3.6 SPI Module
I2C	Master	2	Yes	Intr[3]	Refer to 3.7 I2C Module
TIMER	Slave			Intr[62], Intr[6:4]	Refer to 3.8 TIMER Module 2 programmable timers, 1 watchdog timer
UART	Slave	32	Yes	Intr[0]	Refer to 3.9 UART Module
GPIO	Slave	N/A		Intr[7]	Refer to 3.10 GPIO Module 20 general purpose IO pins with configurable driving mode and pull-up/down. Refer to Table 8 GPIO Pin Multiplex and Section 7.3 GPIO Control Register.
SF_CTRL				Intr[20]	Refer to 3.11 SF Control Module
L1C				Intr[22]	Refer to 3.12 L1C Module
DMA				Intr[26:25], Intr[8]	Refer to 3.13 DMA Module
SEC	AHB Master			Intr[19:17]	Refer to 3.13.4 DMA Transmission Mode **3.13.4.1 Memory to memory** After this mode is started, the DMA will move the data from the source address to the destination address according to the set transfer size. After the transfer, the DMA controller will automatically return to the idle state and wait for the next transfer. The specific configuration process is as follows:

Name	Mode	FIFO	DMA	Interrupt	Comment
					1. Set the value of the register DMA_C[3:0]SRCADDR to the memory address of the source 2. Set the value of the register DMA_C[3:0]DSTADDR to the target memory address 3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 0, that is, select the memory-to-memory mode 4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: set the DI and SI bits to 1 to enable the automatic address accumulation mode, the DTW and STW bits set the transmission width of the source and destination, and the DBS and SBS bits set the burst type of the source and destination 5. Select the appropriate channel, enable DMA, and complete the data transfer **3.13.4.2 Memory to peripheral** In this working mode, the DMA will move data from the source to the internal cache according to the set transfer size (TransferSize). When the cache space is insufficient, the DMA will automatically suspend it. When there is sufficient cache space, continue to transfer until it reaches Set the moving quantity.

Name	Mode	FIFO	DMA	Interrupt	Comment
					On the other hand, when the target peripheral request triggers, it will burst the target configuration to the target address until it reaches the set number of moves and automatically returns to the idle state, waiting for the next startup. The specific configuration process is as follows: 1. Set the value of the register DMA_C[3:0]SRCADDR to the memory address of the source 2. Set the value of the register DMA_C[3:0]DSTADDR to the target peripheral address 3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 1, that is, select the memory-to-peripheral mode 4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: the SI bit is set to 1 to enable the address auto-accumulation mode, and the DI bit is set to 0 to disable the address auto-accumulation mode, the DTW and STW bits set the transmission width of the source and destination, and the DBS and SBS bits set the burst type of the source and destination 5. Select the appropriate channel, enable DMA, and complete the data transfer

3.13.4.3 Peripheral to memory

In this working mode, when the source peripheral request is triggered, the source configuration is burst to the buffer until the set number of moves reaches the stop. On the other hand, when the internal cache is enough for the target burst number once, the DMA will automatically move the cached content to the target address until it reaches the set number of moves and automatically returns to the idle state, waiting for the next startup

The specific configuration process is as follows:

1. Set the value of the register DMA_C[3:0]SRCADDR to the source peripheral address
2. Set the value of the register DMA_C[3:0]DSTADDR to the target memory address
3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 2, that is, select the peripheral-to-memory mode
4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: the DI bit is set to 1 to enable the address auto-accumulation mode, and the SI bit is set to 0 to disable the address auto-accumulation mode , the DTW and STW bits set the transmission width of the source and destination respectively, and the DBS and SBS bits set the burst type of

Name	Mode	FIFO	DMA	Interrupt	Comment
					the source and destination respectively 5. Select the appropriate channel, enable DMA, and complete the data transfer **3.13.4.4 Peripheral to Peripheral** In this working mode, when the source peripheral requests a trigger, the source configuration burst will be stored in the buffer, and it will stop until the set number of moves is reached. On the other hand, when the internal cache is enough for the target burst number, DMA will automatically move the cached content to the target address until the set number of transfers is reached and automatically return to the idle state, waiting for the next start The specific configuration process is as follows: 1. Set the value of the register DMA_C[3:0]SRCADDR to the peripheral address of the source 2. Set the value of the register DMA_C[3:0]DSTADDR to the target peripheral address 3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 3, that is, select the peripheral-to-peripheral mode 4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: DI and SI bits are set

Name	Mode	FIFO	DMA	Interrupt	Comment
					to 0, the address automatic accumulation mode is disabled, the STW and DTW bits respectively set the source and target transfer widths, and the SBS and DBS bits respectively set the source and target bursts type. 5. Select the appropriate channel, enable DMA, and complete the data transfer 3.14 SEC ENGINE Module Security engine supports hardware-based AES and SHA.
OTP	Slave	N/A		Intr[21]	Refer to 3.15 OTP Module
I3C	Master	5		Intr[9]	Refer to 3.16 I3C Module
I3C	Slave	5		Intr[10]	Refer to 3.16 I3C Module
AXI4TG				N/A	Refer to 3.17 AXI4 Traffic Generator Module

3.5.2 AHB/APB Bus Matrix

The µC bus matrix supports seven AHB masters and three AHB slaves with hardcoded address range. There are also subsection decoders for individual module's CSR.

Table 5 lists the matrix masters.

Table 5 Matrix Masters

	Name	Comment
M0	µC system bus	µC (E21) system port for code fetch
M1	D2D AXI5-Lite	AXI5-Lite to AHB
M2	DMA	DMA
M3	TDR2APB	TDR to APB
M4	CONFIG_SPI	(RTL simulation only) SPI BFM port
M5	µC peripheral bus 1	µC (E21) peripheral port 1
M6	Reserved	Reserved for µC peripheral bus 2
M7	AES	Security engine (AES) master port

	Name	Comment
M8	SHA	Security engine (SHA) master port
M9	Reserved	Reserved

Table 6 lists the matrix slaves.

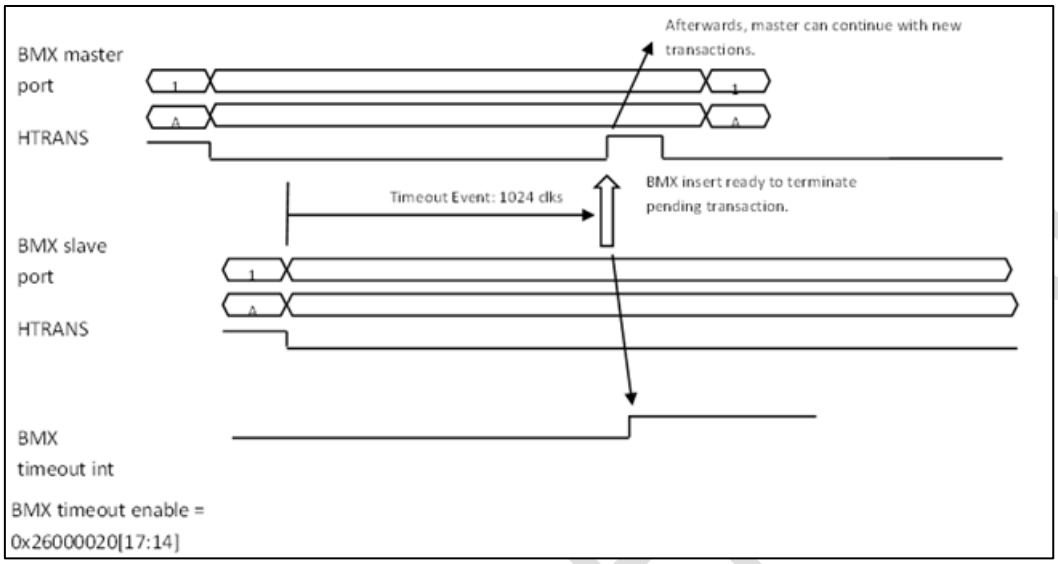
Table 6 Matrix Slaves

	Name	Comment
S0	Mem	Memory region, mask ROM, and XIP NOR
S1	Peripheral	Peripheral region includes µC embedded IO and CSRs
S2	Reserved	Reserved
S3	TIM	µC embedded memory (SRAM)

3.5.3 Bus Timeout

Figure 6 illustrates the bus matrix timeout

Figure 6 Bus matrix Timeout



3.6 SPI Module

3.6.1 SPI Overview

Serial Peripheral Interface (SPI) bus is a synchronous serial communication interface specification for short-range communication. Devices communicate in the full duplex mode, which is a master-slave mode where one master controls one or more slaves.

SPI completes full-duplex communication with four signal lines, namely CS (chip select), SCLK (clock), MOSI (master output slave input), and MISO (master input slave output).

3.6.2 SPI Features

- Can be used as SPI master or slave
- Both master and slave support four operating modes (CPOL and CPHA)
- Both master and slave support 1/2/3/4-byte transfer mode
- The sending and receiving channels each have a FIFO with a depth of 32 bytes
- The adaptive FIFO depth variation characteristic suits high-performance applications
    - When the frame is 32-bits, the depth of FIFO is 8
    - When the frame is 24-bits, the depth of FIFO is 8
    - When the frame is 16-bits, the depth of FIFO is 16
    - When the frame is 8-bits, the depth of FIFO is 32
- Adjustable byte transfer sequence
- Flexible clock configuration
- Configurable MSB/LSB transfer priority
- Receive ignore function: You can set to ignore the reception of data from the specified location
- Supports timeout mechanism in the slave mode
- Supports DMA transfer mode

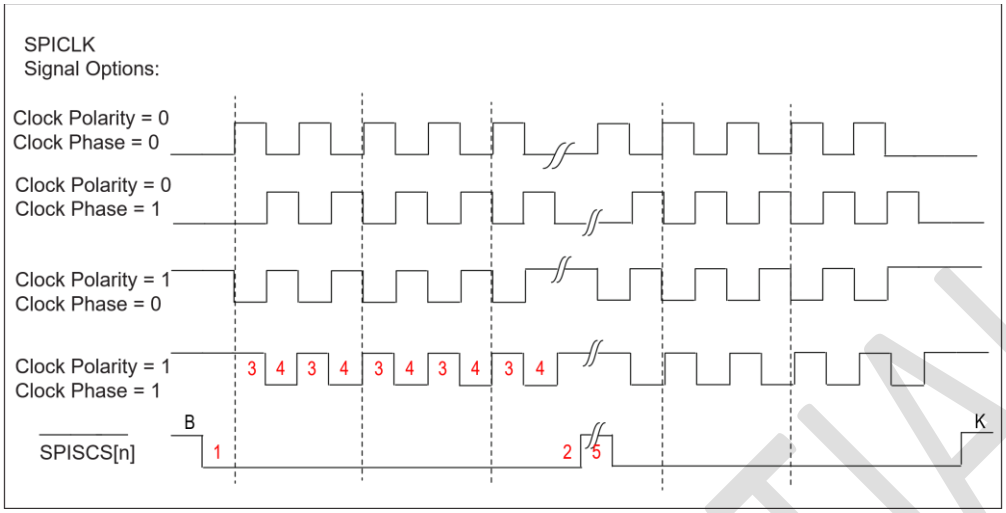
3.6.3 SPI Functional Description

3.6.3.1 SPI Clock Control

Due to different clock phases and polarity settings, the SPI clock has four modes, which can be set by cr_spi_sclk_pol (CPOL) and cr_spi_sclk_ph (CPHA) in the register SPI_CONFIG. The CPOL determines the level of SCK clock signal when it is idle. If CPOL = 0 (cr_spi_sclk_pol = 0), the idle level is low and if CPOL = 1 (cr_spi_sclk_pol = 1), the idle level is high. The CPHA determines the sampling time. If CPHA = 0 (cr_spi_sclk_ph = 1), sampling is made at the first lock edge of each cycle and if CPHA = 1 (cr_spi_sclk_ph = 0), that is made at the second clock edge of each cycle.

By setting the registers SPI_PRD_0 and SPI_PRD_1, you can also adjust the duration of the start and end levels of the clock, time of phase 0/1, and interval between frames of data. The settings of the four modes are shown in Figure 7.

Figure 7 SPI TIMING



The meanings of numbers are as follows:

- "1" denotes the length of the START condition, which is configured by cr_spi_prd_s in the register SPI_PRD_0.
- "2" denotes the length of the STOP condition, which is configured by cr_spi_prd_p in the register SPI_PRD_0.
- "3" denotes the length of phase 0, which is configured by cr_spi_prd_d_ph_0 in the register SPI_PRD_0.
- "4" denotes the length of phase 1, which is configured by cr_spi_prd_d_ph_1 in the register SPI_PRD_0.
- "5" denotes the interval between frames of data, which is configured by cr_spi_prd_i in the register SPI_PRD_1.

3.6.3.2 Master Continuous Transfer Mode

After this mode is enabled, the CS signal will not be released when the current data is sent and there are still available data in FIFO.

3.6.3.3 Master-Slave: Transfer and Receive Data

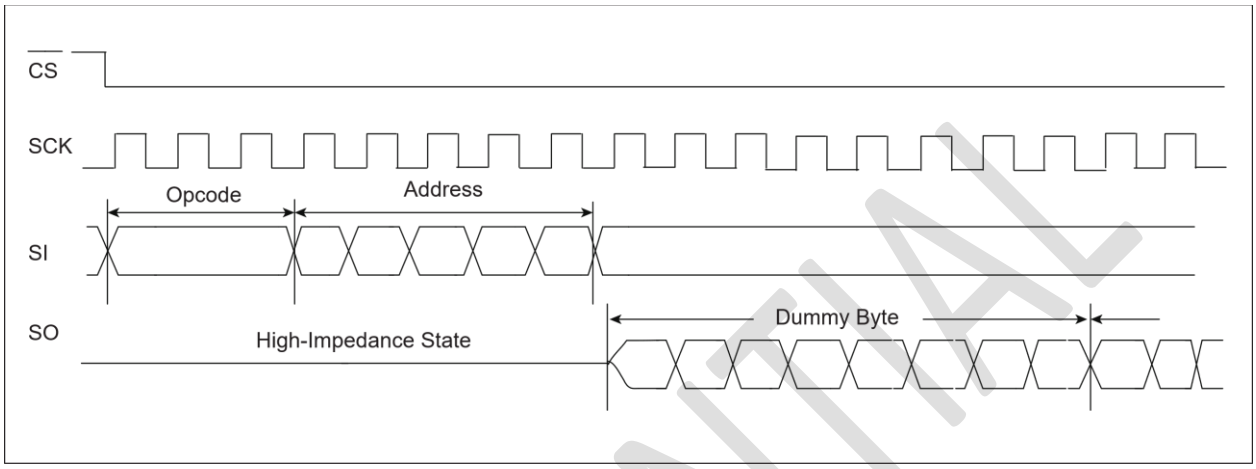
The same `frame_size` shall be set for the master and slave for transferring and receiving data by configuring the `cr_spi_frame_size` in the register SPI_CONFIG. When the master and slave agree to communicate at a 32-bits `frame_size`, if the `clk` of the master does not meet 32-bits due to an exception in a frame of data, the following symptoms occur.

- The data sent by the master cannot be transferred to the RX FIFO of the slave. The slave cannot receive data from the master.
- When the slave sends data, it will skip this frame of data and continue to send the next frame of data when the master's `clk` is normal again.

3.6.3.4 Receive Ignore Function

When the start and end bits to be filtered out are set, SPI will discard the corresponding data segments in the received data as shown in Figure 8.

Figure 8 SPI Ignore Waveform



You can enable this function by configuring the `cr_spi_rxd_ignr_en` in the register SPI_CONFIG. The start bit of this function is set by configuring the `cr_spi_rxd_ignr_s` in the register SPI_RXD_IGNR. The end bit of this function is set by configuring the `cr_spi_rxd_ignr_p` in the register SPI_RXD_IGNR.

In the above Figure 8, the start bit to be filtered is set to 0, if the end bit is set to 7, Dummy Byte will be received; if the end bit is set to 15, Dummy Byte will be discarded.

3.6.3.5 Filtering Function

When this function is enabled and a threshold is set, SPI will filter the data less than or equal to the width threshold. Assuming that the SPI top clock is 160 MHz and the threshold is set to 4, any data with a width below (4/160 MHz = 25 ns) will be filtered out.

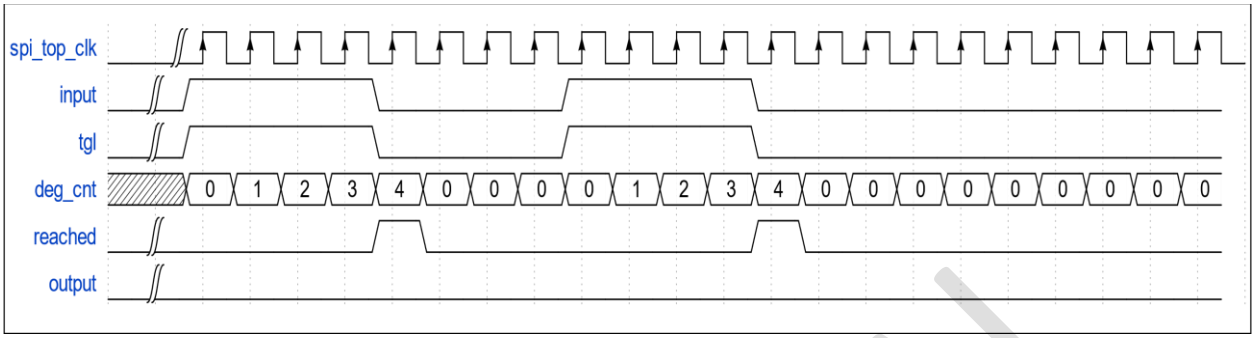
When this function is enabled by setting the `cr_spi_deg_en` in the register SPI_CONFIG and the threshold is set by configuring the `cr_spi_deg_cnt`, SPI will filter out the data that cannot reach the width threshold. As shown in the Figure 9 below, the data width is 4, `Input` is the initial data and `Output` is the filtered data.

Filtering logic process:

- `Tgl` is the exclusive XOR result of input and output.
- `Deg_cnt` counts from 0, and the counting condition is that `tgl` is at the high level and `reached` is at the low level.
- `Reached` means whether the current `deg_cnt` count reaches the set `cr_spi_deg_cnt` and it is at a high level once reached.
- When `reached` is at a high level, `input` is output to `output`.

Note: User-defined condition for `deg_cnt`: `tgl` is at a high level and `reached` is at a low level. In other cases, `deg_cnt` will be cleared to 0.

Figure 9 SPI Filter Waveform



3.6.3.6 Configurable MSB/LSB Transfer

The configurable MSB/LSB transfer mode is limited to the priority transfer sequence of 8-bits in one byte and the transfer sequence of bits in one byte is set by configuring the `cr_spi_bit_inv` bit in the register SPI_CONFIG. 0 indicates MSB and 1 indicates LSB.

For example, for data transfer where the frame size is 24-bits, the data format is Data[23:0] = 0 × 123456.

When MSB transfer is set, the transfer sequence is: 01010110 (binary, 1st byte: 0 × 56); 00110100 (binary, 2nd byte: 0 × 34); 00010010 (binary, 3rd byte: 0 × 12). When LSB transfer is set, the transfer sequence is: 01101010 (binary, 1st byte: 0 × 56); 00101100 (binary, 2nd byte: 0 × 34); 01001000 (binary, 3rd byte: 0 × 12).

3.6.3.7 Adjustable Byte Transfer Sequence

The adjustable byte transfer sequence is limited to the priority transfer sequence between different bytes in FIFO. The transfer sequence of bytes in FIFO is set by configuring the `cr_spi_byte_inv` bit in the register SPI_CONFIG. 0 means sending LSB first, and 1 means sending MSB first.

For example, for data transfer where the frame size is 24-bits, the data format is Data[23:0] = 0 × 123456.

When LSB transfer priority is set, the transfer sequence is 0 × 56 (1st byte: LSB); 0 × 34 (2nd byte: intermediate byte); 0 × 12 (3rd byte: MSB). When MSB transfer priority is set, the transfer sequence is 0 × 12 (3rd byte: MSB); 0×34 (2nd byte: intermediate byte); 0 × 56 (1st byte: LSB).

Adjustable byte transfer can be used in conjunction with configurable MSB/LSB transfer.

3.6.3.8 Slave Mode Timeout Mechanism

When a timeout threshold is set, an interrupt will be triggered when SPI in the slave mode receives no clock signal after the threshold exceeds.

3.6.3.9 IO Transfer Mode

The chip communication processor can perform FIFO padding and clearing operations in response to the interrupt from the FIFO. Each FIFO has a programmable FIFO trigger

threshold to trigger an interrupt. When `rx_fifo_cnt` in the register SPI_FIFO_CONFIG_1 is greater than the trigger threshold of `rx_fifo_th`, an interrupt will be generated to send a signal to the chip communication processor to clear the RX FIFO. When `rx_fifo_cnt` in the register SPI_FIFO_CONFIG_1 is greater than `rx_fifo_th`, an interrupt will be generated to send a signal to the chip communication processor to re-pad the TX FIFO.

You can query the SPI status register to determine the sampled value in the FIFO and the FIFO status. The software must provide correct trigger thresholds for RX FIFO and TX FIFO and prevent the overflow of RX FIFO and the underflow of TX FIFO.

3.6.3.10 DMA Transfer Mode

SPI supports the DMA transfer mode. To enable this mode, you must set the thresholds of TX FIFO and RX FIFO respectively.

Setting `spi_dma_tx_en` in the register SPI_FIFO_CONFIG_0 to 1 can enable the DMA sending mode. Setting `spi_dma_rx_en` in the register SPI_FIFO_CONFIG_0 to 1 can enable the DMA receiving mode.

When this mode is enabled, UART will check the TX/RX FIFO. Once the `tx_fifo_cnt/rx_fifo_cnt` in the register SPI_FIFO_CONFIG_1 is greater than `tx_fifo_th/rx_fifo_th`, a DMA request will be initiated, and DMA will transfer data into TX FIFO or remove data from RX FIFO as configured.

3.6.3.11 SPI Interrupt

SPI supports the following interrupt control modes.

- SPI end of transfer interrupt
  - In the master mode, the SPI end of transfer interrupt will be triggered when the transfer of each frame of data ends.
  - In the slave mode, the interrupt is triggered when the CS signal is released.
- TX FIFO request interrupt
  - The TX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold and the interrupt flag will be cleared automatically when the condition is unmet.
- RX FIFO request interrupt
  - The RX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold and the interrupt flag will be cleared automatically when the condition is unmet.
- Slave mode transfer timeout interrupt
  - The slave mode transfer timeout interrupt will be triggered when no clock signal is received in the slave mode after the threshold exceeds. If the TX/RX FIFO overflows or underflows, it will trigger the TX/RX FIFO overflow interrupt.
- Slave mode TX overload interrupt
  - Slave mode TX overload interrupt will trigger when the TX is not ready to transmit in slave mode.
- TX/RX FIFO overflow interrupt

- TX/RX FIFO overflow interrupt is triggered if an overflow or underflow occurs in the TX/RX FIFO. When the `tx_fifo_clr/rx_fifo_clr` bit in the FIFO clear register SPI_FIFO_CONFIG_0 is set to 1, the corresponding FIFO will be cleared, and the overflow interrupt flag will be cleared automatically.

You can query the interrupt status through the register SPI_INT_STS and write 1 to the corresponding bit to clear the interrupt.

3.7 I2C Module

3.7.1 I2C Overview

Inter-Integrated Circuit (I2C) is a serial communication bus, which uses a multi-slave and multi-master architecture and is connected to low-speed peripherals. Each device has a unique address identifier and can be used as a transmitter or receiver. The address of each device connected to the bus can be set by software through a unique address and the existing master or slave relation. The master can work as a master transmitter or a master receiver. If two or more masters are initialized at the same time, data can be prevented from being damaged through conflict detection and arbitration during transmission.

To facilitate communication with slaves with the FIFO of 2-word depth and interrupt function, it can be used with DMA to improve efficiency and supports flexible adjustment of clock frequency.

3.7.2 I2C Features

- Master mode
- Multi-master mode and arbitration function
- Flexible control of the level duration of the start, end, and data transmission phases in segments
- Supports 7-bit address mode and 10-bit address mode
- Supports DMA transfer mode
- Supports multiple interrupt mechanisms

3.7.3 I2C Functional Description

Table 7 lists the I2C pins.

Table 7 I2C Pin List

Name	Type	Description
I2Cx_SCL	Input/output	I²C serial clock signal
I2Cx_SDA	Input/output	I²C serial data signal

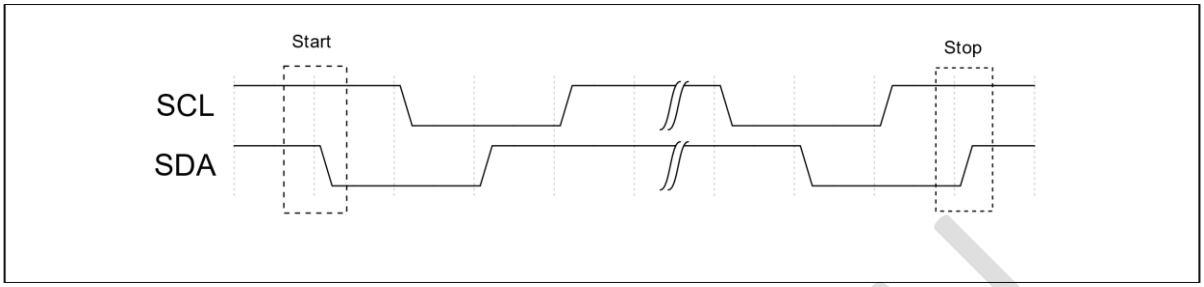
3.7.3.1 Start and Stop Conditions

All transmissions start with a START condition and end with a STOP condition. The START and STOP conditions are generally generated by the master. The bus is busy after the START condition and to be idle for a certain period after the STOP condition.

- START condition: SDA produces a high-to-low level transition when SCL is high
- STOP condition: SDA produces a low-to-high level transition when SCL is high

Figure 10 shows the timing diagram.

Figure 10 Waveform Diagram



3.7.3.2 Data Transfer Format

3.7.3.2.1 7-bit Address Mode

The first 8-bits transferred are addressing bytes, including a 7-bit slave address and a 1-bit direction bit. Sending or receiving data by the master is controlled by the 8th bit in the first byte sent by the master. If it is 0, it means that the data is sent by the master, while "1" indicates that data is received by the master. After the direction bit is the answer bit (ACK), which is sent by the slave to answer (pull the signal low) and the host starts transmitting the specified length of data after receiving the answer. Once the data transfer is complete, the master sends out a STOP signal, with waveform shown as in Figure 11 to

Figure 14.

Figure 11 Master Transmit and Slave Receive Data Formats

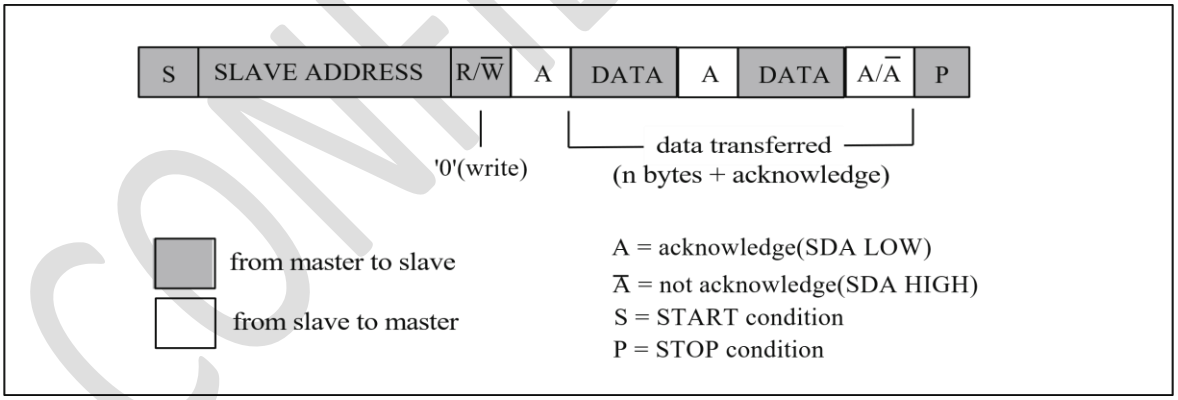


Figure 12 Master Receive and Slave Transmit Data Formats

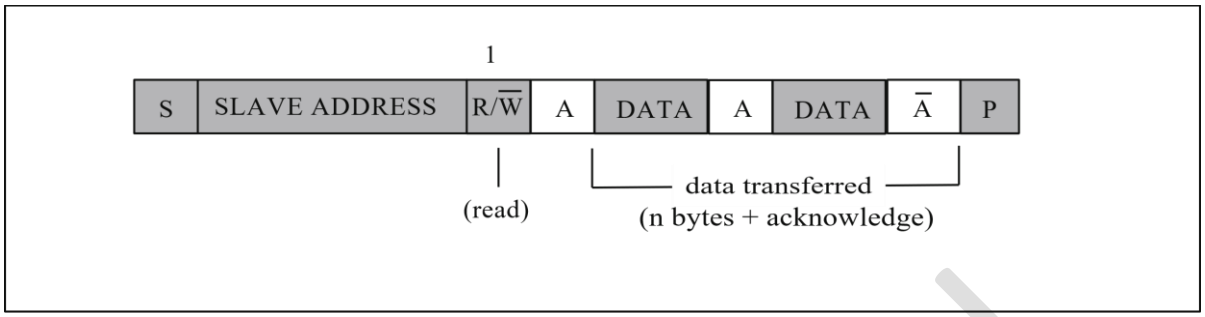


Figure 13 Timing of Master Transmitter and Slave Receiver

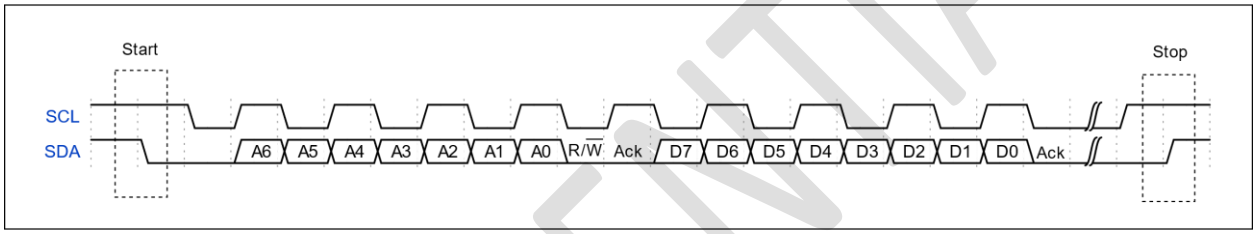
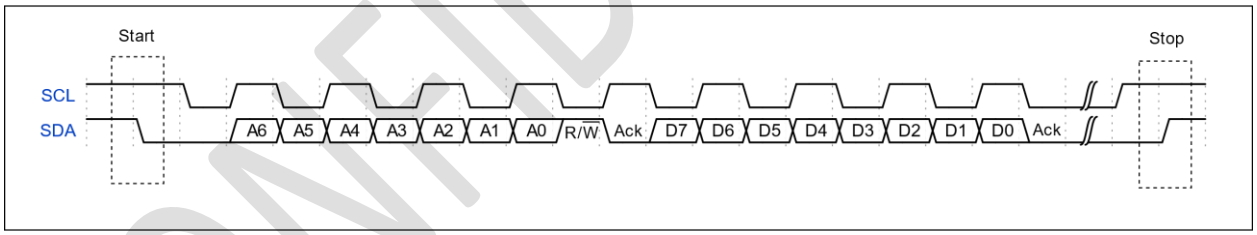


Figure 14 Timing of Master Receiver and Slave Transmitter



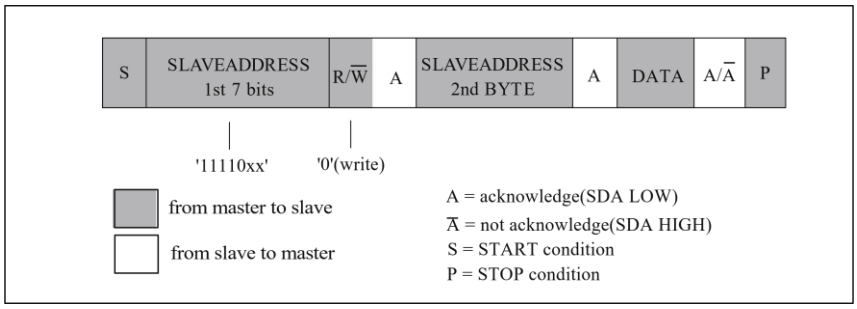
3.7.3.2.2 10-bit Address Mode

The `cr_i2c_10b_addr_en` in the register I2C_CONFIG must be set to 1 before use.

The 10-bit slave address consists of the two bytes after the START condition (S) or the repeated START condition (Sr). The first 7-bits of the first byte are 1111 0XX, where XX are the first two bits of MSB of the 10-bit address. The 8th bit of the first byte is the read/write bit that determines the transfer direction. The second byte is the remaining low 8-bits of the 10-bit address. The data transfer format is as shown in

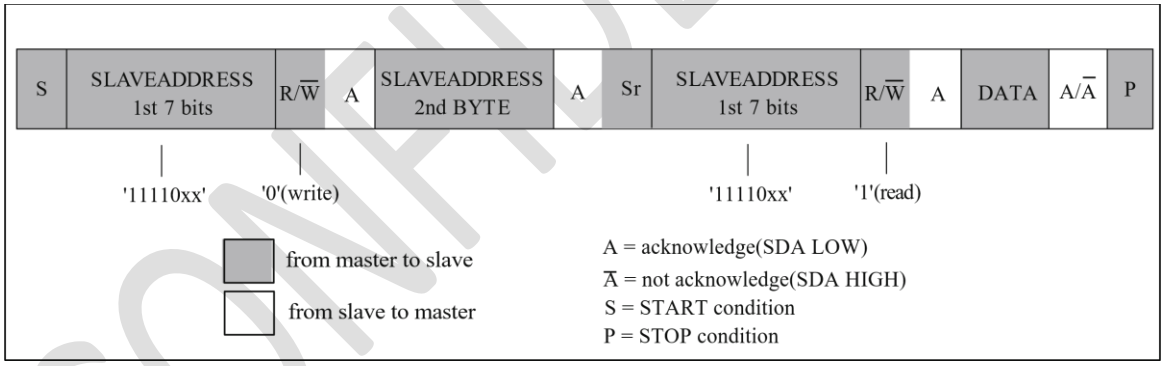
Figure 15.

Figure 15 Master Transmit and Slave Receive Data Format (10 bit slave address)



When receiving the 10-bit address following the START condition, the slave compares the first byte (1111 0XX) of the slave address with its own address and checks whether the eighth bit (read/write bit) is 0. If the value of XX in the first byte is the same as the top two bits of the slave's 10-bit address, the first byte match passes and the slave will give answer A. If there are multiple slave devices connected to the bus, more than one device may match and generate answer A. Next, all slaves start to match the second byte (XXXX XXXX), where only one slave will have the exact same lower eight bits of the 10-bit address as the second byte, and that slave will give answer A. The slave that is addressed by the master will remain addressed until it receives a termination condition, or a repeat start condition.

Figure 16 Master Receive and Slave Transmit Data Format (10 bit slave address)



Before the second acknowledgement A, the process is same as that of the master-transmitter addressing the slave-receiver. After the repeated START condition (Sr), the matched slave will remain in the addressed state. This slave will check whether the first 7-bits of the first byte after Sr are 1111 0XX, and then test whether the 8th bit is 1 (read). If this also matches, the slave considers that it is addressed as a transmitter and generates an acknowledgement (A). The slave-transmitter will remain in the addressed state until it receives the STOP condition (P) or the repeated START condition (Sr) followed by a different slave address. Then, under Sr, all the slaves will compare their addresses with 11110XX and test the eighth bit (read/write bit). However, they will not be addressed, because for 10-bit devices, the read/write bit is 1, or for 7-bit devices, the slave addresses of 1111 0XX do not match.

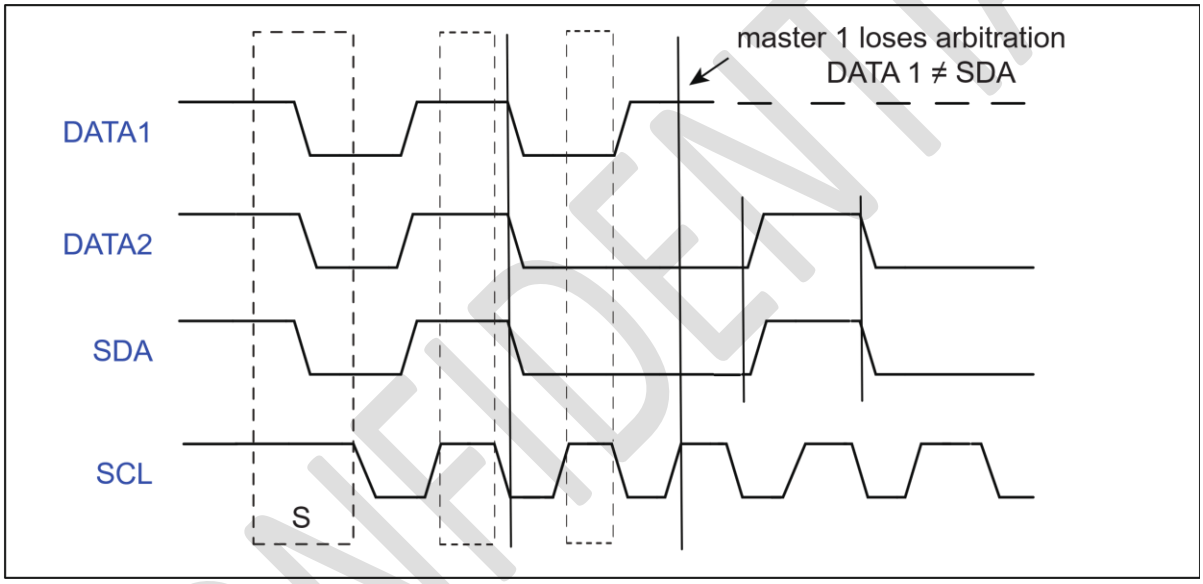
3.7.4 Arbitration

When there are multiple masters on I²C bus, it may happen that multiple masters start data transfer at the same time. At this time, the arbitration mechanism will decide which master has the right to transfer data, while other masters must give up control of the bus and wait until the bus is idle before transferring data again.

During data transfer, all masters must check whether the SDA is consistent with the data they want to send when SCL stays high. When the SDA level is different from the expected one, it means that other masters are transferring data at the same time. The masters with different SDA levels will lose the arbitration and other masters will complete the data transfer.

The waveform of two masters transferring data and initiating the arbitration mechanism at the same time is as shown in Figure 17.

Figure 17 Waveform of Simultaneous Data Transfer

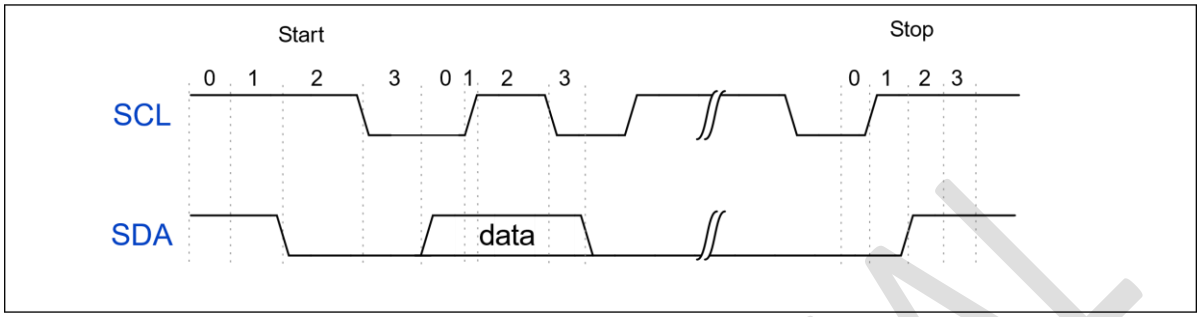


3.7.5 I2C Clock Setting

I2C clock can be derived from `bclk` (bus clock) and `xclk`, and frequency division can be done on this basis. The duration of the start condition, each bit of data and the end condition are set by registers I2C_PRD_START, I2C_PRD_DATA and I2C_PRD_STOP respectively. Each of these durations can be subdivided into four phases, and the number of samples in each phase is controlled by a separate byte in the register (the actual value is the register value plus 1). The four phase settings in the data section together determine the frequency division factor of the I2C clock. As shown in the figure below, suppose the I2C clock source is selected as 80M bclk and the register I2C_PRD_DATA is set to 0x09070b09, then the second 0 in the figure is 0x09 + 1 = 0x0a, the second 1 is 0x07 + 1 = 0x08, the second 2 is 0x0b + 1 = 0x0c, and the second 3 is 0x09 + 1 = 0x0a. Then the clock frequency of I2C is 80 MHz/(0x0a + 0x08 + 0x0c + 0x0a) = 2 MHz. Similarly, the first 0, 1, 2 and 3 are set by register I2C_PRD_START, which determines the duration of the start condition, and the third

0, 1, 2, and 3 are set by register I2C_PRD_STOP, which determines the duration of the end condition.

Figure 18 I2C Clock Setting



3.4.6 I2C Configuration Flow

Configuration Items:

- Read/write flag bit
- Slave address
- Slave register address
- Slave register address length
- Data (TX: configure the sent data; RX: store the received data)
- Data length
- Enable signal

3.7.6.1 Read/Write Flag Bit

I2C supports TX and RX working statuses. The `cr_i2c_pkt_dir` in the register I2C_CONFIG represents the TX/RX status, 0 for TX status and 1 for RX status.

3.7.6.2 Slave Address

Each slave connected to I2C will have a unique device address, which is usually 7-bits long. This address will be written into the `cr_i2c_slv_addr` in the register I2C_CONFIG. I2C will automatically shift to the left by 1 bit before sending the address, and the TX/RX direction bit will be added to the LSB.

3.7.6.3 Slave Register Address

The slave register address represents the register address where I2C must read and write a slave register. The slave register address is written to the register I2C_SUB_ADDR and the `cr_i2c_sub_addr_en` in the register I2C_CONFIG must be set to 1. If `cr_i2c_sub_addr_en` in the register I2C_CONFIG is set to 0, the I2C master will skip the slave register address field when sending.

3.7.6.4 Slave Register Address Length

The slave register address length is subtracted by 1 and then written to `cr_i2c_sub_addr_bc` in the register I2C_CONFIG.

3.7.6.5 Data

It refers to the data that must be sent to or received from the slave. When sending data, I2C must write the data (in word) into the register I2C_FIFO_WDATA. When receiving data, I2C must read out the data (in word) from the register I2C_FIFO_RDATA.

3.7.6.6 Data Length

The `cr_i2c_pkt_len` in the register I2C_CONFIG sets the send data length (the value written to the register + 1 is the send data length) and the maximum send length is 256-bytes.

3.7.6.7 Enable Signal

After the above items are configured, when `cr_i2c_m_en` in the enable signal register I2C_CONFIG is set to 1, the I2C sending process will be started automatically.

When the read/write flag bit is configured as 0, I2C sends data. For example, consider sending 2 bytes, the master's transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1-bit + 0) + ACK
3. Slave register address + ACK
4. 1-byte data + ACK
5. 1-byte data + ACK
6. Stop bit

When the read/write flag bit is configured as 1, I2C receives data. For example, consider receiving 2 bytes, the master's transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1-bit + 0) + ACK
3. Slave register address + ACK
4. Start bit
5. (The slave address shifts to the left by 1-bit + 1) + ACK
6. 1-byte data + ACK
7. 1-byte data + ACK
8. Stop bit

3.7.7 FIFO Management

I2C FIFO has a 2-word depth, and I2C includes RX FIFO and TX FIFO. The `rx_fifo_cnt` in the register I2C_FIFO_CONFIG_1 represents how much data (in word) in RX FIFO needs to be read. The `tx_fifo_cnt` in the register I2C_FIFO_CONFIG_1 represents how much free space (in word) in TX FIFO for writing.

I2C FIFO status:

- **RX FIFO underflow:** When the data in RX FIFO is completely read out or empty, if I2C continues to read data from RX FIFO, the `rx_fifo_underflow` in the register I2C_FIFO_CONFIG_0 will be set to 1
- **RX FIFO overflow:** When I2C receives data until the two words of RX FIFO are filled without reading RX FIFO, if I2C receives data again, the `rx_fifo_overflow` in the register I2C_FIFO_CONFIG_0 will be set to 1
- **TX FIFO underflow:** When the data size filled into TX FIFO does not meet the configured I2C data length `cr_i2c_pkt_len` in register I2C_CONFIG and no new data is filled into TX FIFO, the `tx_fifo_underflow` in the register I2C_FIFO_CONFIG_0 will be set to 1
- **TX FIFO overflow:** After the two words of TX FIFO are filled before the data in TX FIFO is sent out, if data is filled into TX FIFO again, the `tx_fifo_overflow` in the register I2C_FIFO_CONFIG_0 will be set to 1.

3.7.8 Use with DMA

I2C can send and receive data through DMA. Setting `i2c_dma_tx_en` in the register I2C_FIFO_CONFIG_0 to 1 will enable the DMA TX mode. After the channel for I2C is allocated, DMA will transfer data from memory to the register I2C_FIFO_WDATA. Setting `i2c_dma_rx_en` in the register I2C_FIFO_CONFIG_0 to 1 will enable the DMA RX mode. After the channel for I2C is allocated, DMA will transfer the data in the register I2C_FIFO_RDATA to memory. When I2C is used with DMA, DMA will automatically transfer data, so it is unnecessary for CPU to write data into I2C TX FIFO or read data from I2C RX FIFO.

3.7.8.1 DMA Sending Flow

1. Set read/write flag bit to 0
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data length
6. Set enable signal register to 1
7. Configure DMA transfer size
8. Configure the transfer width of DMA source address
9. Configure the transfer width of DMA destination address (when I2C is used with DMA, the transfer width of destination address must be set to 32-bits, which is word-aligned)
10. Configure the DMA source address as the memory address for storing sent data
11. Configure the DMA destination address to I2C TX FIFO address, `i2c_fifo_wdata`
12. Enable DMA

3.7.8.2 DMA Receiving Flow

1. Set read/write flag bit to 1
2. Set slave address
3. Set slave register address
4. Set slave register address length

5. Data length
6. Set enable signal register to 1
7. Configure DMA transfer size
8. Configure the transfer width of DMA source address (when I2C is used with DMA, the transfer width of source address must be set to 32-bits, which is word-aligned)
9. Configure the transfer width of DMA destination address
10. Configure the DMA source address to I2C RX FIFO address, `i2c_fifo_rdata`
11. Configure the DMA destination address as the memory address for storing received data
12. Enable DMA

3.7.9 I2C Interrupt

I2C includes the following interrupts:

- `I2C_TRANS_END_INT`: I2C transfer end interrupt, which is generated when I2C completes a transfer.
- `I2C_TX_FIFO_READY_INT`: When `tx_fifo_cnt` in I2C_FIFO_CONFIG_1 is greater than `tx_fifo_th`, a TX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied.
- `I2C_RX_FIFO_READY_INT`: When `rx_fifo_cnt` in I2C_FIFO_CONFIG_1 is greater than `rx_fifo_th`, an RX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied.
- `I2C_NACK_RECV_INT`: When the I2C module detects a NACK state, a NACK interrupt is generated.
- `I2C_ARB_LOST_INT`: `I2C` arbitration lost interrupt.
- `I2C_FIFO_ERR_INT`: `FIFO` ERROR interrupt is generated when TX/RX FIFO overflows or underflows.

3.8 TIMER Module

3.8.1 Timer Overview

The chip has two 32-bit counters, each of which can independently control and configure its parameters and clock frequency.

There is a watchdog counter in the chip. Unpredictable software or hardware behavior may cause the application to malfunction. A watchdog timer can help the system recover from it. If the current time exceeds the predetermined time, but the dog is not fed or closed Timer, which can trigger interrupt or system reset according to the setting.

Figure 19 BL TIMER

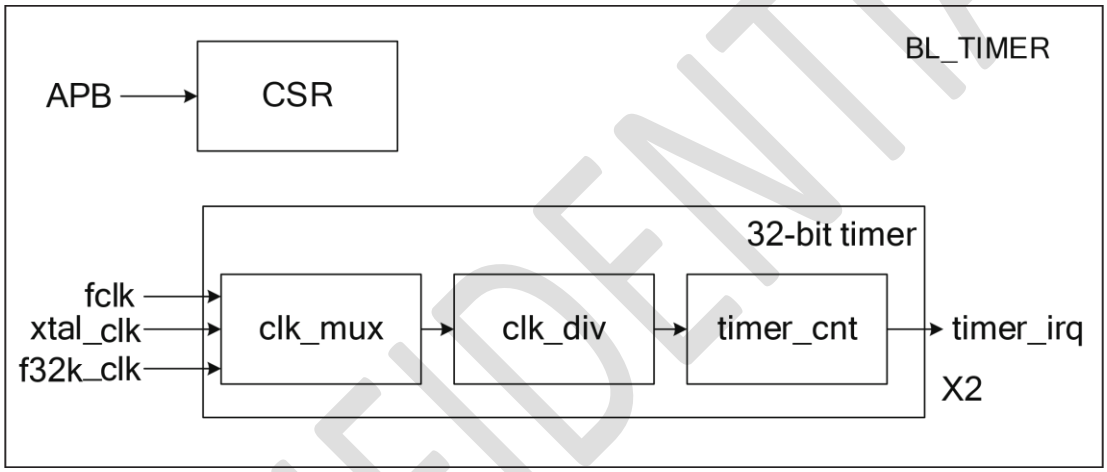
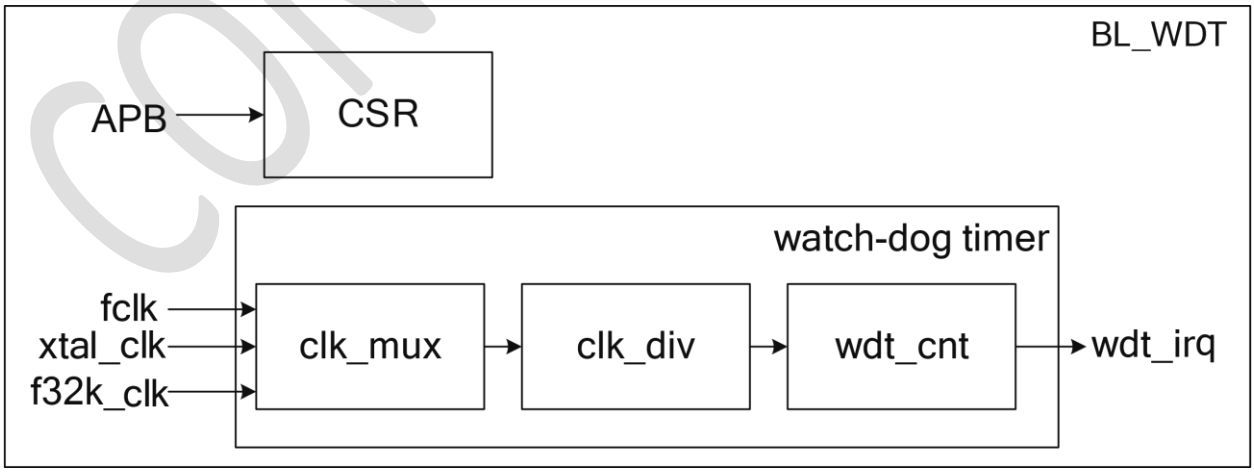


Figure 20 BL WDT



3.8.2 Timer Features

- Multiple clock source options
- 8-bit clock divider with a division factor of 1-256.
- Two 32-bit timers
- Each timer contains three alarm value settings, which can be set independently to alarm when each alarm value overflows
- Support FreeRun mode and PreLoad mode
- 16-bit watchdog timer
- Supports write password protection to prevent system abnormalities caused by incorrect settings
- Support two watchdog overflow methods: interrupt or reset

3.8.3 TIMER function description

8-bit Divider

There are three types of Watchdog timer clocks:

- Fclk--System master clock
- 32K--32K clock
- Xtal--External crystal

There are four timer clock sources:

- Fclk--System master clock
- 32K--32K clock
- 1K--1K clock（32K frequency division）
- Xtal--External crystal

Each counter has its own 8-bit frequency divider. The selected clock can be divided by 1-256 through APB. Specifically, when it is set to 0, it means no frequency division, and when it is set to 1, it divides it by 2. The maximum frequency division coefficient is 256, the counter will use the divided clock as the unit of the counting cycle, each time a counting cycle is increased by one.

3.8.4 General timer operating mode

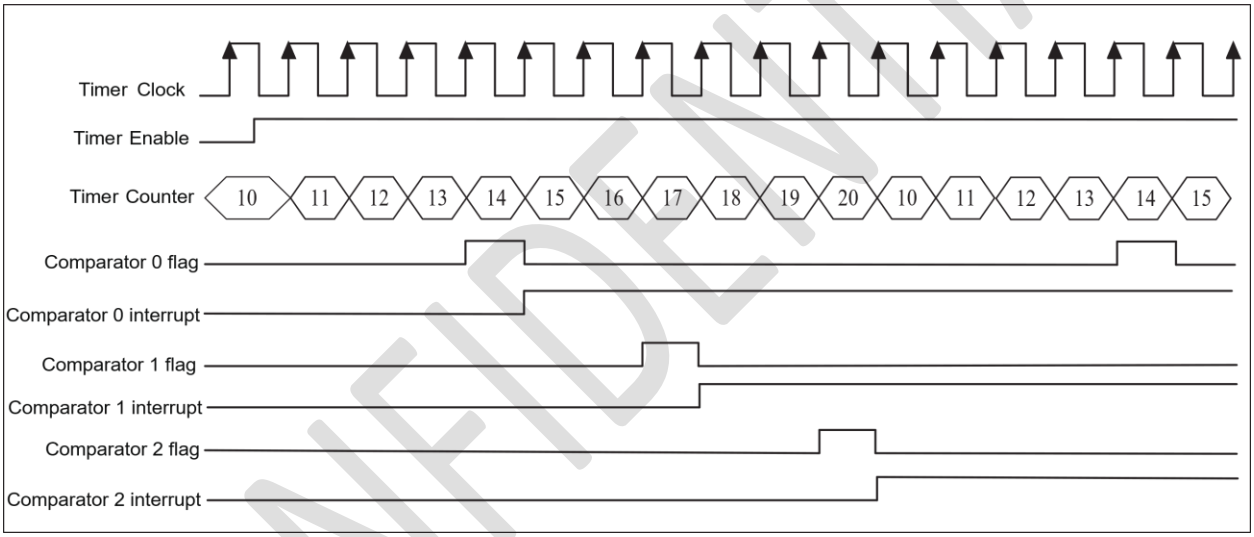
Each general-purpose timer includes three comparators, a counter and a preload register. When the clock source is set and the timer is started, the counter starts to count. When the counter value is equal to the comparator, the comparison is performed. When the flag is set, a compare interrupt is generated.

The initial value of the counter depends on the timing mode. In FreeRun mode, the initial value of the counter is 0, and then counts. When it reaches the maximum value, it starts counting from 0 again.

In PreLoad mode, the initial value of the counter is the value of the PreLoad register and then counts. When the PreLoad condition is met, the value of the counter is set to the value of the PreLoad register, and then the counter starts to count again. During the counting process, once the value of the counter matches one of the three comparators, the comparator's comparison flag will be set and a corresponding comparison interrupt can be generated.

If the value of the preload register is 10, the value of Comparator 0 is 13, the value of Comparator 1 is 16, and the value of Comparator 2 is 19, the working sequence of the timer in PreLoad mode is as follows:

Figure 21 PRELOAD



In FreeRun mode, the timer working sequence is basically the same as PreLoad, the difference is that the counter will start to accumulate from 0 to the maximum value. The mechanism of the generated compare flags and compare interrupts is the same as in PreLoad mode.

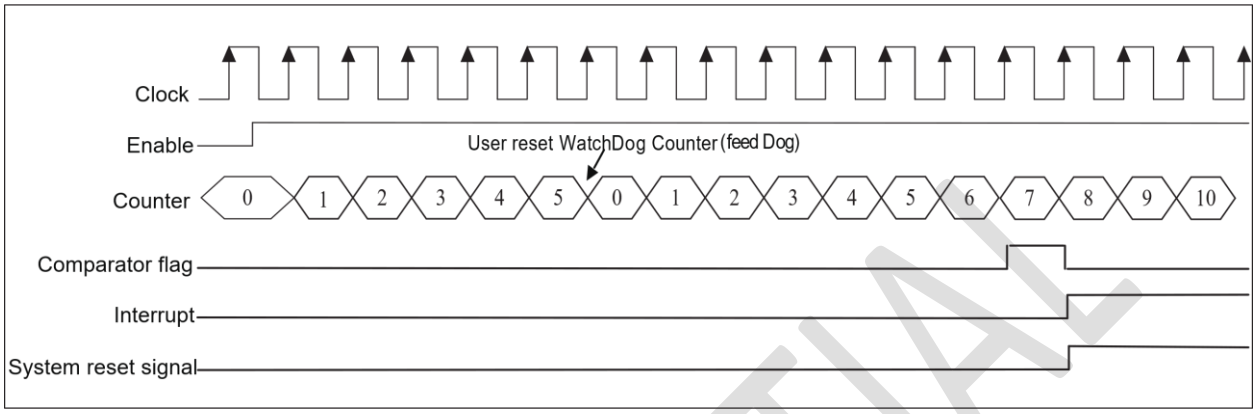
3.8.5 Watchdog timer operating mode

The watchdog timer includes a counter and a comparator. The counter counts from 0. If the counter is reset (feed the dog), it starts counting from 0 again. When the counter value is equal to the comparator, a comparison interrupt signal or a system reset signal will be generated, and the user can choose to use one of them as required.

The watchdog counter is incremented by one in each counting cycle unit. Software can reset the watchdog counter to zero at any point in time through the APB.

If the value of the comparator is 6, the working sequence of Watchdog is shown in the figure below:

Figure 22 WATCHDOG COUNTER



3.8.6 Alarm setting

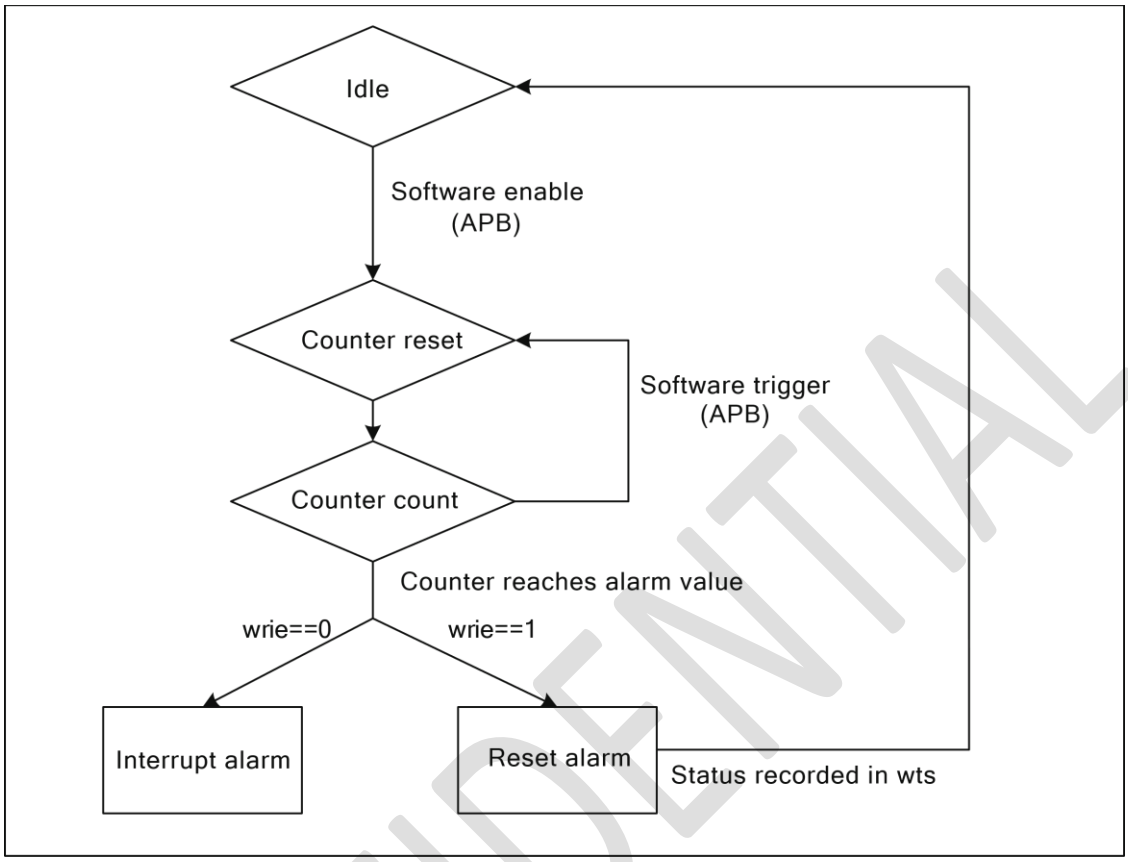
Each counter has three comparison values, and can set whether each comparison value triggers an alarm interrupt. When the counter matches the comparison value and the setting will alarm, the counter will notify the processor through the interrupt.

The software can read through the APB whether an alarm has occurred and which comparison value triggered the alarm interrupt. When the alarm interrupt is cleared, the alarm status is also cleared simultaneously.

3.8.7 Watchdog Alarm

A comparison value can be set for each counter. When the software fails to reset the watchdog counter to zero due to a system error, which causes the watchdog counter to exceed the comparison value, a watchdog alarm is triggered. There are two types of alarms. The first is to perform necessary actions through interrupt notification software. The second is to enter the system watchdog reset. When the watchdog reset is triggered, it will notify the system reset controller and prepare for system reset. When everything is ready, enter the system watchdog reset. It is worth noting that software can read the WSR register through APB to know if a watchdog system reset has occurred.

Figure 23 WATCHDOG ALARM



3.9 UART Module

3.9.1 UART Overview

The Universal Asynchronous Receiver/Transmitter (UART) provides a flexible way to exchange full-duplex data with external devices. µC subsystem is provided with 1 UARTs, which can be used together with DMA to achieve efficient data communication.

3.9.2 UART Features

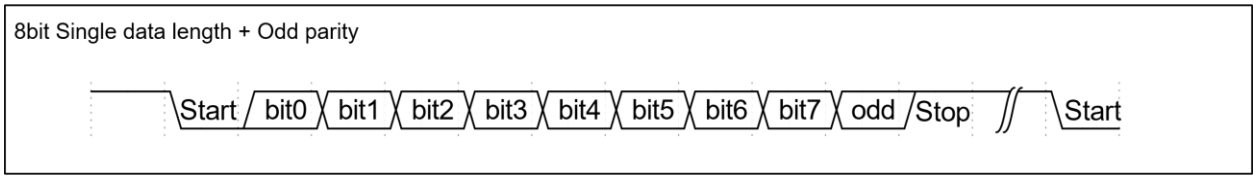
- Full-duplex asynchronous communication
- Optional data bit length: 5/6/7/8-bit
- Optional stop bit length: 0.5/1/1.5/2-bit
- Supports odd/even/none check bit
- Error-detectable start bit
- Abundant interrupt control modes
- Hardware flow control (RTS/CTS)
- Convenient baud rate programming
- Configurable MSB/LSB transfer priority
- Automatic baud rate detection of ordinary/fixed characters
- 32-byte TX/RX FIFO
- Supports DMA transfer mode
- Supports baud rate of 10 Mbps and below
- Supports the LIN bus protocol
- Supports the RS485 mode
- Optional clock sources: 160M/BCLK/XCLk
- Support filter function

3.9.3 UART Functional Description

3.9.3.1 Data Formats

The normal UART communication data consists of start bits, data bits, parity check bits, and stop bits. The UART supports configurable data bits, parity check bits, and stop bits, which are set in registers UTX_CONFIG and URX_CONFIG. The waveform of a frame of data is as shown below in Figure 24.

Figure 24 UART Data Format



The start bit of the data frame occupies 1-bit, and the stop bit can be 0.5/1/1.5/2-bit wide by configuring the `cr_utx_bit_cnt_p` bit in the register UTX_CONFIG. The start bit is at a low

level and the stop bit is at a high level. The data bit width can be set to 5/6/7/8-bit by the `cr_utx_bit_cnt_d` bit in the register UTX_CONFIG.

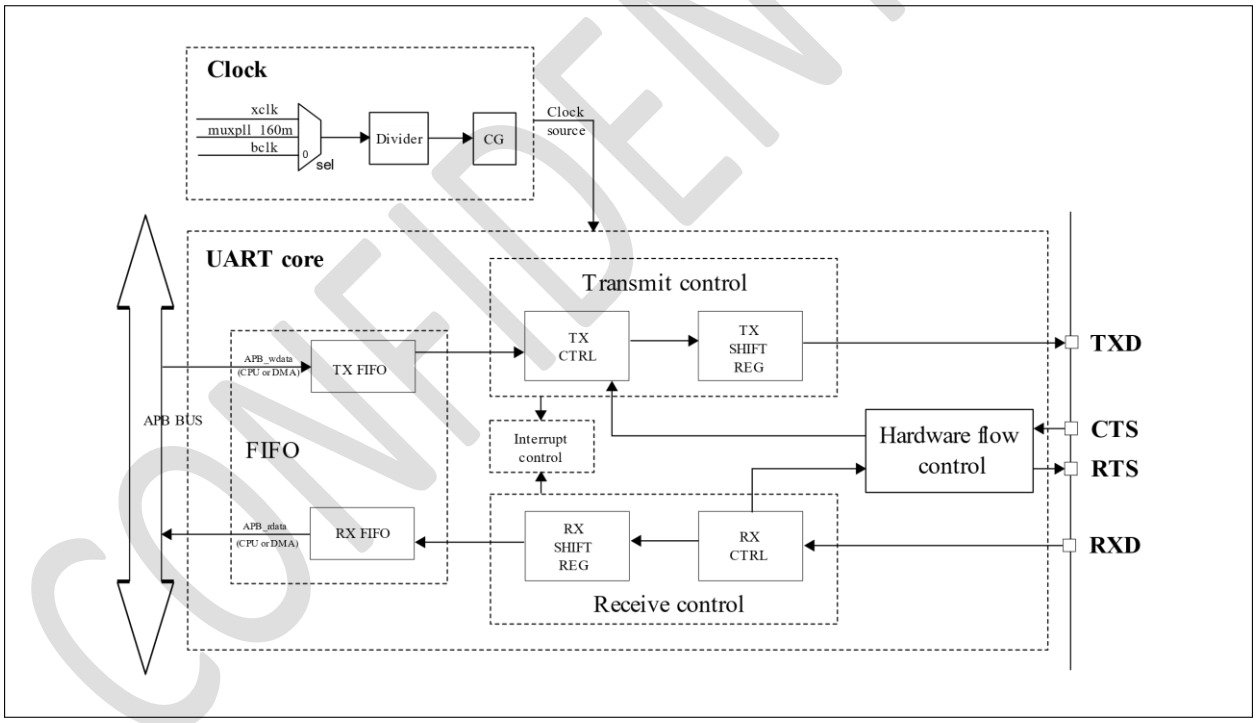
When the `cr_utx_prt_en` bit in the register UTX_CONFIG and the `cr_urx_prt_en` bit in the register URX_CONFIG is set, the data frame adds a parity check bit after the data. The `cr_utx_prt_sel` bit in the register UTX_CONFIG and the `cr_urx_prt_sel` bit in the register URX_CONFIG is used to select odd or even parity check. When the receiver detects the check bit error of the input data, it will generate the check error interrupt. However, the received data will still be stored into the FIFO.

Calculation method of odd parity check: If there is an odd number of "1" in the current data bit, the odd parity check bit is set to 0. Otherwise, it is set to 1.

Calculation method of even parity check: If there is an odd number of "1" in the current data bit, the even parity check bit is set to 1. Otherwise, it is set to 0.

3.9.3.2 Basic Architecture

Figure 25 UART Data Architecture



The UART has 3 clock sources: XCLK, 200 MHz CLK, and BCLK. The frequency divider in the clock is used to divide the frequency of the clock source and then generate the clock signal to drive the UART module.

The UART controller is divided into two functional blocks: transmitter and receiver.

3.9.3.3 Transmitter

The transmitter contains a 32-byte TX FIFO to store the data to be sent. When the transmission enable bit is set, the data stored in FIFO will be output from the TX pin. Software can transfer data into TX FIFO through DMA or APB bus. Software can check the status of transmitter by querying the remaining free space count value of TX FIFO through `tx_fifo_cnt` in the register UART_FIFO_CONFIG_1.

FreeRun mode of transmitter:

- If the FreeRun mode is disabled, transmission will be terminated, and an interrupt will be generated when the sent bytes reach the specified length. Before next transmission, you must re-disable and enable the `TxE` bit.
- If the FreeRun mode is enabled, the transmitter will send when there is data in the TX FIFO and will not stop working because the sent bytes reach the specified length.

3.9.3.4 Receiver

The receiver contains a 32-byte RX FIFO to store the received data. Software can check the status of receiver by querying the available data count value of RX FIFO through `rx_fifo_cnt` in the register UART_FIFO_CONFIG_1. The low 8-bits of the register URX_RTO_TIMER are used to set a receiving timeout threshold, which will trigger an interrupt when the receiver fails to receive data beyond the threshold. The `cr_urx_deg_en` and `cr_urx_deg_cnt` in the register URX_CONFIG is used to enable the deburring function and set the threshold, which controls the filtering part before sampling by UART. UART will filter out the glitches whose width is lower than the threshold in the waveform and then sends it to sampling.

3.9.3.5 Baud Rate Setting

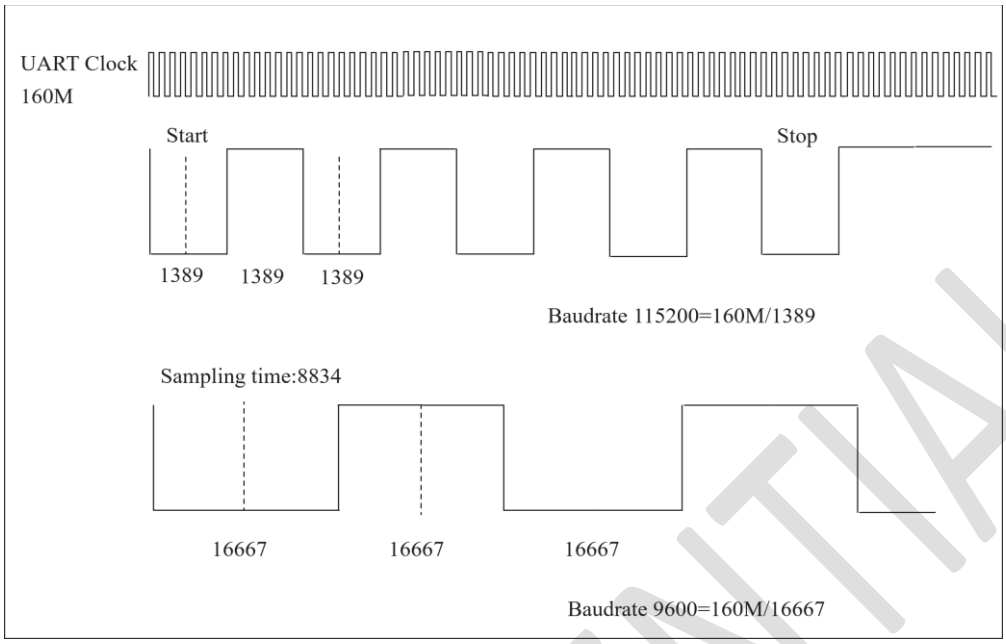
```
Baudrate = \frac{UART\_clk}{uart\_prd + 1}
```

The user can set the baud rate of RX and TX separately. For example, consider TX the value of `uart_prd` is the value of the lower 16-bits `cr_utx_bit_prd` of the register UART_BIT_PRD. Since the maximum value of the 16-bit bit width coefficient is 65535, the minimum baud rate supported by UART is `UART_clk`/65536.

Before sampling the data, UART will filter the data to remove the glitches in the waveform. Then, the data will be sampled at the intermediate value of the 16-bit width factor, so that the sampling time can be adjusted based on baud rates to ensure that the intermediate value is always sampled providing much higher flexibility and accuracy. The sampling process is as shown in

Figure 26.

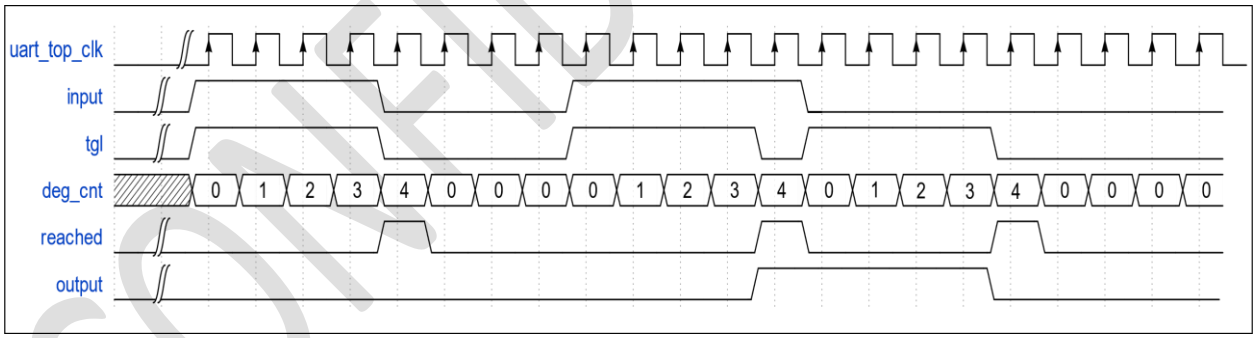
Figure 26 UART Sampling Waveform



3.9.3.6 Filtering

Figure 27 illustrates the UART filter waveform.

Figure 27 UART Filter Waveform



When this function is enabled by configuring `cr_urx_deg_en` and the threshold is set by configuring `cr_urx_deg_cnt` in the register URX_CONFIG, UART will filter out the data that cannot meet the width threshold. As shown in the above figure, when the data width is 4, setting `cr_urx_deg_cnt` to 4 can meet this condition. `Input` is the initial data and `output` is the filtered data.

Filtering logic process:

- `Tgl` is the exclusive XOR result of input and output.
- `Deg_cnt` counts from 0 and the counting condition is that `tgl` is at a high level and `reached` is at a low level.
- If the count value of `deg_cnt` reaches the value set by `cr_urx_deg_cnt`, `reached` is at a high level.

- When `reached` is at a high level, `input` is output to `output`.

Note: User-defined condition for `deg_cnt`: `tgl` is at a high level and `reached` is at a low level. In other cases, `deg_cnt` will be cleared to 0.

3.9.3.7 Automatic Baud Rate Detection

The UART module supports automatic baud rate detection, which is divided into two modes, a generic mode, and a fixed character (square wave) mode. The `cr_urx_abr_en` in the register URX_CONFIG enables auto baud rate detection, and when it is turned on, both detection modes are enabled.

3.9.3.7.1 Generic mode

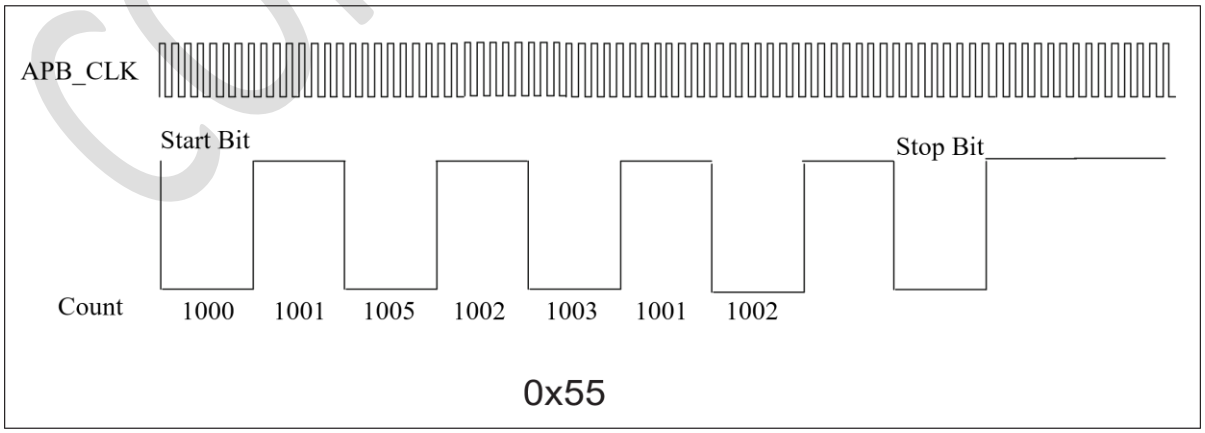
For any character data received, UART will count the number of clocks in the start bit width, which will then be written into the low 16-bits `sts_urx_abr_prd_start` in the register STS_URX_ABR_PRD and used to calculate the baud rate. So, the correct baud rate can be obtained when the first received data bit is 1, such as '0x01' under `LSBFIRST`.

3.9.3.7.2 Fixed character (square wave) mode

In this mode, after the UART module counts the number of clocks in the bit width, it will continue to count the number of clocks in the subsequent data bits and compare it with the start bit. The check is passed, otherwise the count value is discarded. The allowable error can be set by setting the `cr_urx_abr_pw_tol` bit in the register URX_ABR_PW_TOL, and the unit is the clock source of the UART.

Therefore, only when the fixed character '0x55'/'0xD5' under LSB-FIRST or '0xAA'/'0xAB' under MSB-FIRST is received, the UART module will write the clock count value in the starting bit width into the high 16-bit `sts_urx_abr_prd_0x55` of the register STS_URX_ABR_PRD as shown in Figure 28.

Figure 28 Waveform of UART in Fixed Character Mode



As shown above, assuming the maximum allowable error set is 4, for a received data with unknown baud rate, the UART uses `UART_CLK` to count the bit width of the starting bit as 1000, the bit width of the second bit as 1001, which is not more than 4 `UART_CLK` up or down from the previous bit width, then the UART will continue to count the third bit. The third bit is 1005, the difference with the starting bit is more than 4, the detection is not passed and the data is discarded. UART compares the first 6-bit widths of the data bits with the starting bit in turn.

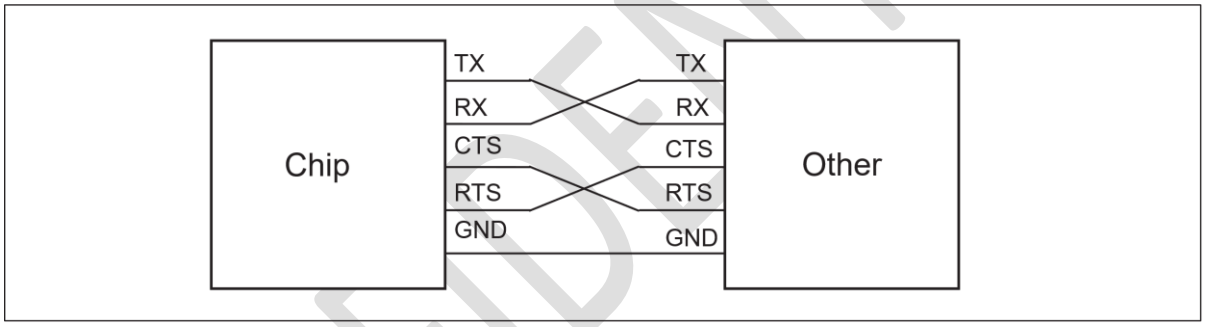
Formula for calculating the detected baud rate:

```
Baudrate = \frac{UART\_clk}{Count + 1}
```

3.9.3.8 Hardware Flow Control

UART supports hardware flow control in CTS/RTS mode to prevent data in FIFO from being lost due to too late processing. Hardware flow control connection is as shown in Figure 29.

Figure 29 UART Hardware Flow Control



Require To Send (RTS) is an output signal, which indicates whether the chip is ready to receive data from the other side. This is valid at a low level denoting that the chip can receive data.

Clear To Send (CTS) is an input signal, which determines whether the chip can send data to the other side. This is valid at a low level denoting that the chip can send data to the other side.

When the hardware flow control function is enabled, the low level of chip's RTS indicates requesting the other side to send data and the high level of that indicates informing the other side to stop sending data. When the chip detects that CTS goes high, TX will stop sending data and continue sending until CTS goes low. If CTS goes high or low at any time during communication, it does not affect the continuity of data sent by TX and the other side also can receive continuous data.

Following are the two ways for hardware flow control of the transmitter.

- Hardware control (the `cr_urx_rts_sw_mode` in the register UART_SW_MODE is 0): RTS goes high when `cr_urx_en` in the register URX_CONFIG is not turned on or the RX FIFO is almost full (one byte left).
- Software control (the `cr_urx_rts_sw_mode` in the register UART_SW_MODE is 1): The level of RTS can be changed by configuring `cr_urx_rts_sw_val` in the register UART_SW_MODE.

3.9.3.9 DMA Transfer

UART supports DMA transfer. Using DMA transfer, the TX and RX FIFO thresholds must be set respectively by `tx_fifo_th` and `rx_fifo_th` in register UART_FIFO_CONFIG_1. When this mode is enabled, if `tx_fifo_cnt` in `uart_fifo_config_1` is greater than `tx_fifo_th`, a DMA TX request will be triggered. After the DMA is configured, when the DMA receives the request, it will move the data from the memory to the TX FIFO according to the settings. If the `rx_fifo_cnt` in `uart_fifo_config_1` is greater than `rx_fifo_th`, the DMA RX request will be triggered. After the DMA is configured, when the DMA receives the request, it will transfer the data of the RX FIFO to the memory according to the settings.

To ensure the correctness of the data transferred by the chip DMA TX channel, the following conditions need to be met in the channel configuration.

```
(transferWidth * burstSize) ≤ (tx_fifo_th + 1)
```

To ensure the integrity of the data transferred by the chip DMA RX channel, the following conditions need to be met in the channel configuration.

```
(transferWidth * burstSize) = (rx_fifo_th + 1)
```

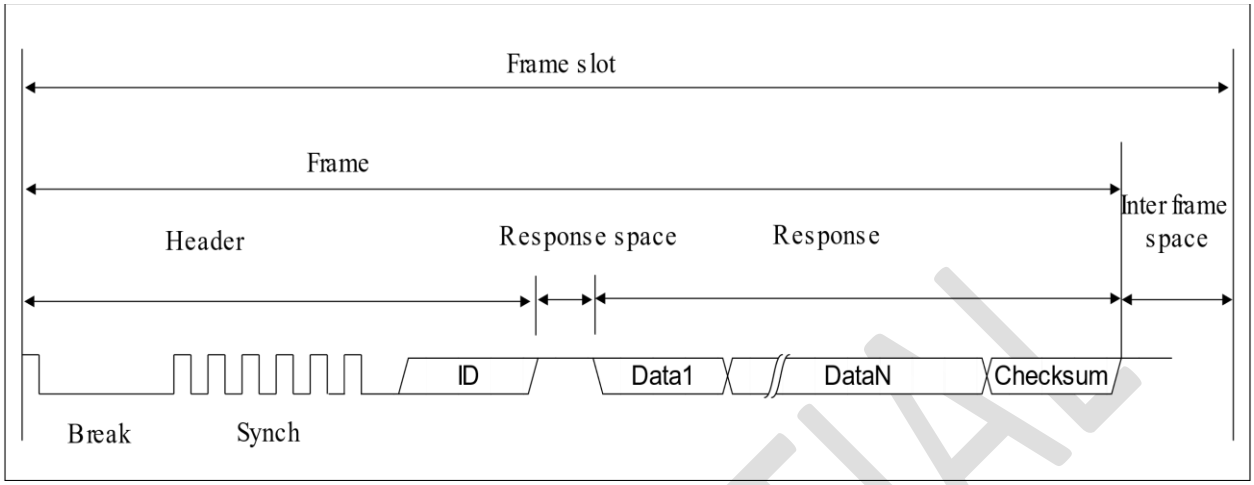
3.9.3.10 Support for LIN Bus

The protocol for the Local Interconnect Network (LIN) is based on the Volcano-Lite technology developed by the Volvo spin-out company -Volcano Communications Technology (VCT). LIN is a complementary protocol to CAN and SAE J1850, suitable for applications that have low requirement for time or require no precise fault tolerance (as LIN is not as reliable as CAN). LIN aims to be easy to use as a low-cost alternative to CAN. The vehicle parts where LIN can be used include window regulator, rearview mirror, wiper, and rain sensor.

UART supports the LIN bus mode. The LIN bus is under the master-slave mode, and data is always initiated by the master node. The frame (header) sent by the master node contains synchronization interval field, synchronization byte field, and identifier field. A typical LIN data transfer is as shown in

Figure 30.

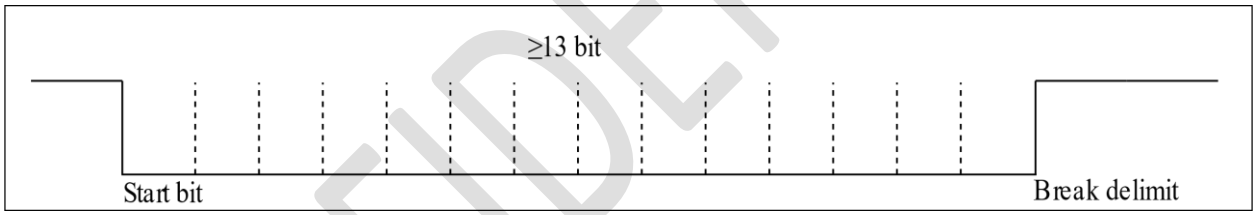
Figure 30 Typical LIN Frame



3.9.3.10.1 LIN break field

Figure 31 illustrates timing diagram for break field of LIN.

Figure 31 Break Field of LIN



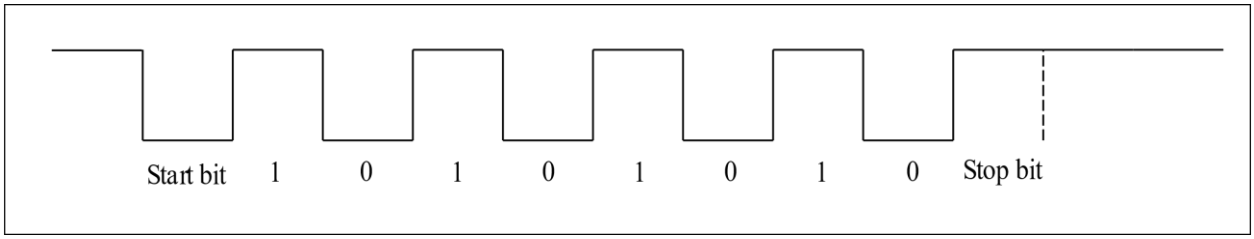
The synchronization interval field indicates the start of the message with at least 13 dominant bits (including the start bit). The synchronization interval ends with an "interval separator", which contains at least one recessive bit.

The length of the break in the LIN frame can be set by `cr_utx_bit_cnt_b` in UTX_CONFIG.

3.9.3.10.2 LIN Sync field

A synchronization byte field is sent to determine the time between two falling edges, to determine the transmission rate used by the master node. The bit pattern is 0x55 (01010101, maximum number of falling edges).

Figure 32 Sync Field of LIN

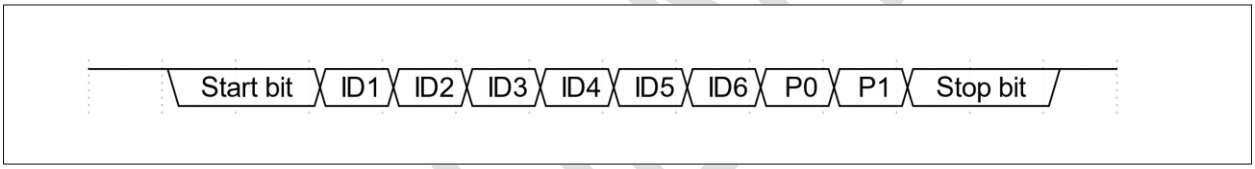


3.9.3.10.3 LIN ID field

The identifier field contains a 6-bit identifier and two parity check bits. The 6-bit identifier contains information about the sender and receiver, and the number of bytes required in the response. The parity check bit is calculated as follows:

The check bit P0 is the result of logical OR operation among ID0, ID1, ID2, and ID4. The check bit P1 is the result of inversion after logical OR operation among ID1, ID3, ID4, and ID5.

Figure 33 LIN ID Field



The slave node waits for the synchronization interval field, and then starts to synchronize between master and slave nodes through the synchronization byte field. Depending on the identifier sent by the master node, the slave node will receive, send, or do not respond. The slave node that should send data sends the number of bytes requested by the master node, and then ends the transmission with a checksum field.

UART supports the LIN transfer mode. To enable this mode, you must configure the `cr_utx_bit_cnt_b` by setting `cr_utx_lin_en` in the register UTX_CONFIG so that the synchronization interval field consists of at least 13-bit dominant level.

3.9.3.11 RS485 mode

UART supports the RS485 mode. After the `cr_utx_rs485_en` in the register UTX_RS485_CFG is set, UART can work in the RS485 mode. Then, UART can be connected to the RS485 bus through an external RS485 transceiver. In this mode, the RTS pin in the module performs the Dir function of transceiver. When UART has data to send, it will automatically control the RTS pin at a high level so that the transceiver can send data to the bus. Contrarily, when UART has no data to send, it will automatically control the RTS at a low level to keep the transceiver in the RX state.

UART supports the RS485 transfer mode. To enable this mode, you must set `cr_utx_rs485_pol` and `cr_utx_rs485_en` in the register UTX_RS485_CFG.

3.9.3.12 UART Interrupt

UART supports the following interrupt control modes:

- TX end of transfer interrupt
- RX end interrupt
- TX FIFO request interrupt
- RX FIFO request interrupt
- RX timeout interrupt
- RX parity check error interrupt
- TX FIFO overflow interrupt
- RX FIFO overflow interrupt
- RX BCR interrupt
- LIN synchronization error interrupt
- Auto baud rate detection (universal mode) interrupt
- Auto baud rate detection (fixed characters mode) interrupt

3.9.3.12.1 TX/RX end of transfer interrupt

You can set a transfer length for TX and RX respectively by configuring the high 16-bits of the registers UTX_CONFIG and URX_CONFIG. When the number of transferred bytes reaches this value, the corresponding TX/RX end of transfer interrupt will be triggered. While this interrupt is generated, TX stops working. To continue to use TX, you must re-initialize this module. Then, RX resumes to work. If the preset transfer length of TX is less than the data volume sent by TX, the other side can only receive the data equal to the transfer length and the remaining data will be stored in TX FIFO. After this module is re-initialized, the data in TX FIFO can be sent out.

For TX, if the data is continuously filled into the TX FIFO is greater than the set transmission length value, only the data of the set length value will be transmitted on the TX pin and the excess data will be kept in the TX FIFO, and then the TX will be re-enabled. After the function, the remaining data in the TX FIFO will be sent out.

For example, set the TX transmission length value to 64, enable the TX function, first fill 63 bytes into the TX FIFO, these 63 bytes will be transmitted on the pins, but no TX transmission completion interrupt is generated, and then the TX FIFO is sent to the TX FIFO. Fill in 1 byte, at this time, the transmission length reaches the transmission length value set by TX, a transmission completion interrupt will be generated, and the TX function will stop. Continue to fill 1 byte into the TX FIFO, you will find that there is no data transmission on the pin, the byte is still retained in the TX FIFO and the TX function is turned off and re-enabled, and the byte is sent out on the pin.

For RX, if the data length sent by the other party exceeds the set transmission length, RX can continue to receive data after the RX transmission completion interrupt is generated.

For example, set the RX transmission length value to 16, the other party sends 32 bytes of data, RX will generate RX transmission completion interrupt when it receives 16 bytes of data and continue to receive the remaining 16 bytes of data, all saved in the RX FIFO.

3.9.3.12.2 TX/RX FIFO request interrupt

An TX FIFO request interrupt will be generated when `tx_fifo_cnt` in `uart_fifo_config_1` is greater than `tx_fifo_th`. When the condition is not met, the interrupt flag will be cleared automatically.

An RX FIFO request interrupt will be generated when rx_fifo_cnt in `uart_fifo_config_1` is greater than `rx_fifo_th`. When the condition is not met, the interrupt flag will be cleared automatically.

TX/RX supports multiple rounds of transmission/receiving instead of reaching the value set by `tx_fifo_th/rx_fifo_th` at a time.

For example,

1. Set `tx_fifo_th/rx_fifo_th` in register UART_FIFO_CONFIG_1 to 16.
2. Set `cr_utx_frm_en` in register UTX_CONFIG to enable free run mode.
3. Set `cr_utx_frdy_mask/cr_urx_frdy_mask` in register UART_INT_MASK to 0 and enable FIFO interrupt of TX/RX.
4. Set `cr_utx_en/cr_urx_en` in register UTX_CONFIG/URX_CONFIG to enable TX/RX.
5. TX FIFO interrupt: TX will always enter the FIFO interrupt, when the chip sends 128-bytes, set `cr_utx_frdy_mask` to 1 to shield the interrupt. If you want to enter the TX FIFO interrupt again, set `cr_utx_frdy_mask` to 0.
6. RX FIFO interrupt: The other party first sends 15-bytes, no interrupt is generated. At this time, the value of `rx_fifo_cnt` is 15 and an interrupt is generated when 1 byte is sent again to reach the value set by `rx_fifo_th`. After the transmission is interrupted, the other party sends the data again and the chip can receive the data.

3.9.3.12.3 RX timeout interrupt

The RX timeout interrupt generation condition: After receiving data last time, the receiver will start timing and the interrupt will be triggered when the timing value exceeds the timeout threshold, and the next data has not been received. The time-out threshold value is in the unit of communication bit.

When the other party sends data to the chip, a timeout interrupt will be generated after the set timeout period is reached.

3.9.3.12.4 RX Parity Check Error Interrupt

The RX parity check error interrupt will be generated when a parity check error occurs. But it does not affect the RX, which still can correctly receive and analyze the data sent by the other side. When receiving data, RX takes the first 8-bits as data bits and ignores parity check bits ensuring data consistency.

For example, you can enable parity check by setting `cr_utx_prt_en/cr_urx_prt_en` in the register UTX_CONFIG/URX_CONFIG and select the parity check type by setting `cr_utx_prt_sel/cr_urx_prt_sel` in the register UTX_CONFIG/URX_CONFIG. When

the other side sends data to the chip through odd/even parity check, the parity check of RX is disabled but RX can receive correct data.

3.9.3.12.5 TX/RX FIFO Overflow Interrupt

If the TX/RX FIFO overflows or underflows, it will trigger the corresponding overflow interrupt. When the `tx_fifo_clr` and `rx_fifo_clr` bits in the FIFO clear bit register UART_FIFO_CONFIG_0 is set to 1, the corresponding FIFO will be cleared, and the overflow interrupt flag will be cleared automatically. You can query the interrupt status through the register UART_INT_STS and clear the interrupt by writing 1 to the corresponding bit in the register UART_INT_CLR.

3.9.3.12.6 RX BCR Interrupt

A BCR interrupt will be generated when the data received by RX reaches the value set by `cr_urx_bcr_value` in the register URX_BCR_INT_CFG.

The difference from RX END interrupt is that END interrupt is suitable for receiving data of known length, while BCR interrupt can be used to receive interrupts of unknown length. The trigger position of the END interrupt is controlled by `cr_urx_len` and the counter will be cleared to 0 when an interrupt is triggered. The trigger position of BCR interrupt is controlled by `cr_urx_bcr_value`. When the interrupt is triggered, the counter will accumulate instead of being cleared to 0, but it can be cleared by software `(cr_urx_bcr_clr)`. When the BCR interrupt is used together with the chained DMA, check the "count" to find out how many data have been transferred by the DMA.

3.9.3.12.7 LIN Synchronization Error Interrupt

When `cr_utx_lin_en` in UTX_CONFIG is enabled, the LIN mode is enabled. Then, if the synchronization field of the LIN bus is not detected when data is received in this mode, the LIN synchronization error interrupt will be generated.

3.9.3.12.8 Auto Baud Rate Detection (universal/fixed characters mode) Interrupt

In the auto baud rate detection mode, when a baud rate is detected, the auto baud rate detection (universal/fixed characters mode) interrupt will be generated as configured.

3.10 GPIO Module

3.10.1 GPIO Overview

Users can connect General Purpose I/O Ports (GPIO) with external hardware devices to control these devices.

3.10.2 GPIO Features

- 20 GPIO pins
- Each I/O pin can be configured in pull-up, pull-down, or floating mode
- Each I/O pin can be configured as input, output or Hi-Z state mode
- The output mode of each I/O pin has 4 optional drive capabilities
- The input mode of each I/O pin can be set to enable/disable the Schmitt trigger

3.10.3 GPIO Pin Multiplex

The µC's GPIO pins are multiplexed with embedded peripheral functions. Table 8 lists the pins.

Table 8 GPIO Pin Multiplex

GPIO#	Function Select = 0			Function Select = 1, 2, 3		Scan Mode
	POS/BOOT	config_spi_en = 1 (For simulation)	spi1_en = 1			
gpio[19]	FW_MODE (i)[1]					
gpio[18]	I3C_ready (o)					Edt_channel_out
gpio[17]	ext_rx_phy_rstn (i)[2]					Edt_channel_out
gpio[16]	ext_tx_phy_rstn (i)[2]	config_spi_do				Edt_channel_out
gpio[15]	ext_phy_rst_en (i)[2]	config_spi_di				Edt_channel_out
gpio[14]	BOOT_LED[2] (o)[3]	config_spi_clk				Edt_channel_out
gpio[13]	BOOT_LED[1] (o)[3]		spi1_sclk			Edt_channel_out
gpio[12]	BOOT_LED[0] (o)[3]		spi1_ss_o			Edt_channel_out
gpio[11]	CHIP_ID[1] (i)[4]		spi1_miso			Edt_channel_out
gpio[10]	CHIP_ID[0] (i)[4]		spi1_mosi			Edt_channel_in
gpio[9]				i2c_sda (pull-up)	3	Edt_channel_in
gpio[8]				i2c_scl (pull-up)	3	Edt_channel_in
gpio[7]	Jtag_sel[0] (i)[5]			spi_clk	2	Edt_channel_in
gpio[6]	Jtag_sel[1] (i)[5]			spi_ss	2	Edt_channel_in

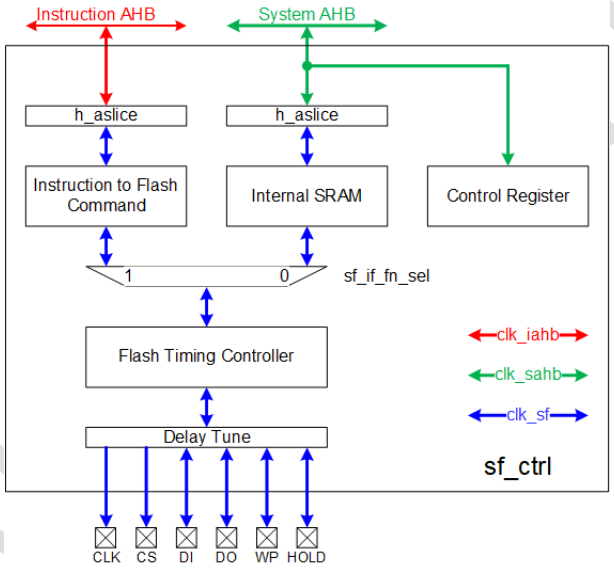
GPIO#	Function Select = 0			Function Select = 1, 2, 3		Scan Mode
	POS/BOOT	config_spi_en = 1 (For simulation)	spi1_en = 1			
gpio[5]				spi_miso	2	Edt_channel_in
gpio[4]				spi_mosi	2	Edt_channel_in
gpio[3]				uart_rxd	1	Edt_channel_in
gpio[2]				uart_txd	1	Scan_enable
gpio[1]				uart_cts_n	1	Scan_clock
gpio[0]				uart_rts_n	1	Edt_update

1. FW_MODE = 1'b1, use SF (SPI/QSPI) to download FSBL;
   FW_MODE = 1'b0, use I3C to download FSBL. External pull down for GFH default.
2. For D2D loop back test.
3. For F/W debug.
4. A total of 4 GFHs can be connected to one master chiplet. External pull up/down.
   CHIP_ID[1:0] = 2'b00: GFH#0; 2'b01: GFH#1; 2'b10: GFH#2; 2'b11: GFH#3.
5. JTAG select. External pull up/down,
   Jtag_sel[1:0] = 2'b00, select D2D PHY JTAG;
   Jtag_sel[1:0] = 2'b01, select DDR PHY JTAG;
   Jtag_sel[1:0] = 2'b1X, select TDR2APB JTAG.

3.11 SF Control Module

- Execute-In-Place (XIP) for MCU application.
- Configurable flash controller supports all brands of QSPI Flash
- Support Max 256MB Flash capacity.
- Support Max 133MHz Flash speed (or faster according to timing constraint)
- The Flash Controller is a QSPI timing controller. It converts data to QSPI protocol in the following two modes:
    - Indirect mode: read/erase/program flash from system bus. The Flash controller execute the sram data to/from QSPI interface.
    - XIP mode: read flash data from instruction bus. Once the Flash controller receives the target address from instruction bus, it execute read command immediately and read back the data from QSPI flash.

Figure 34 SF Control Architecture

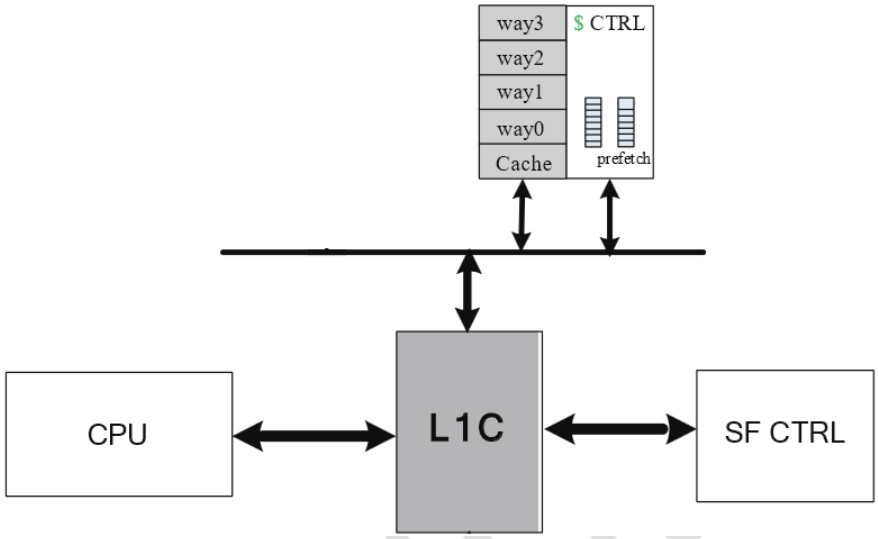


3.12 L1C Module

3.12.1 L1C Overview

L1 Cache Controller is a unit module located outside the processor, used to manage the code or data buffer on Flash and improve the speed of CPU access to Flash. The architecture is illustrated in the fllowing figure.

Figure 35 L1C Architecture



L1C is a high-speed unit integrated between the processor and Flash. Because the speed of the processor is very fast, when the processor needs to wait for a long time to access Flash, the less time wasted, the higher the efficiency. The L1C cache can be used as a lubricating role between the processor and the Flash to improve the efficiency of the processor.

3.12.2 Features

- 4-way Set-Associative mapping
- Support cache performance statistics

3.12.3 L1C function description

3.12.3.1 Mutual conversion between TCM and Cache RAM resources

To increase memory usage efficiency, it is supported to adjust all or part of the Cache's 16K RAM, so that users can adjust the memory usage and efficiency according to the actual situation. The maximum Cache can be set to 16K, divided into 4 ways, each way is 4K, and the unit of adjustment is 1 way, which is 4K. The default size is 16K. Set by WayDisable. The actual space size of Cache can be flexibly adjusted.

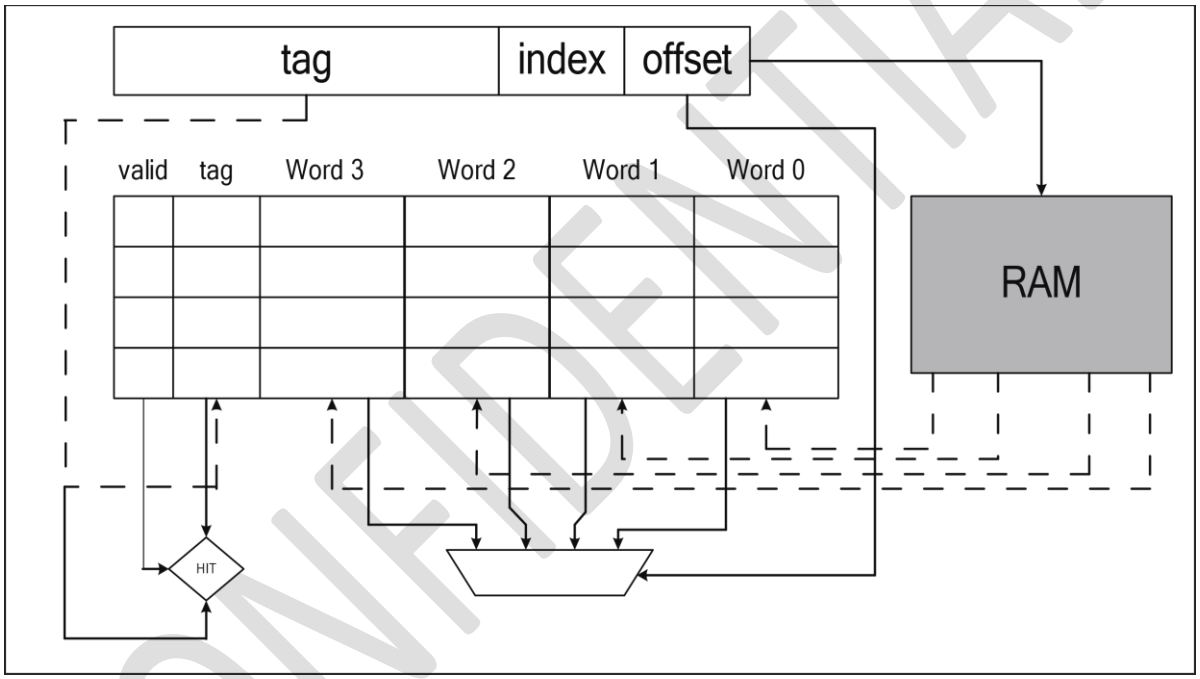
Table 9 Space size of Cache and ITCM

WayDisable	Cache	ITCM
none	16K	0K
one way	12K	4K
two way	8K	8K
three way	4K	12K
four way	0K	16K

3.12.3.2 Cache

The unit of each line buffer is 32 bytes, and the 4-way associative mapping cache is used. The application architecture is as follows:

Figure 36 Cache Architecture



Each set of associative mapping caches contains two parts, the first is a tag, which contains the valid value and the address mapping relationship. The second part is data storage. When the processor accesses the cache, the cache processor compares the relationship between the address and the tag. When the address comparison is successful, the representative can directly get data from the cache. Conversely, the cache processor will capture related data through the AHB master and put the data into the cache and respond to the processor's data.

When most of the data can be successfully compared in the tag, the waiting time of the processor can be greatly reduced, and the use efficiency can be increased.

3.13 DMA Module

3.13.1 DMA Overview

DMA (Direct Memory Access) is a memory access technology that can independently read and write system memory directly without processor intervention. Under the same degree of processor load, DMA is a fast data transfer method. The DMA controller has 4 channels, which manage the data transfer between peripheral devices and memory to improve bus efficiency.

There are four main types of transfers: memory to memory, memory to peripheral, peripheral to peripheral and peripheral to memory. And support LLI link list function. Use the software to configure the transmission data size, data source address, and destination address.

3.13.2 DMA main Features

- 4 independently configurable channels (requests) on DMA
- Independent control of source and destination access width (single-byte, double-byte, four-byte)
- Each channel acts as a read-write cache independently
- Each channel can be triggered by independent peripheral hardware or software
- Support peripherals including UART, I2C and SPI
- 4 kinds of process control
    - DMA flow control, source memory, target memory
    - DMA flow control, source memory, target peripheral
    - DMA flow control, source peripheral, target memory
    - DMA flow control, source peripheral, target peripheral
- Support LLI linked list function to improve DMA efficiency

3.13.3 DMA Functional Description

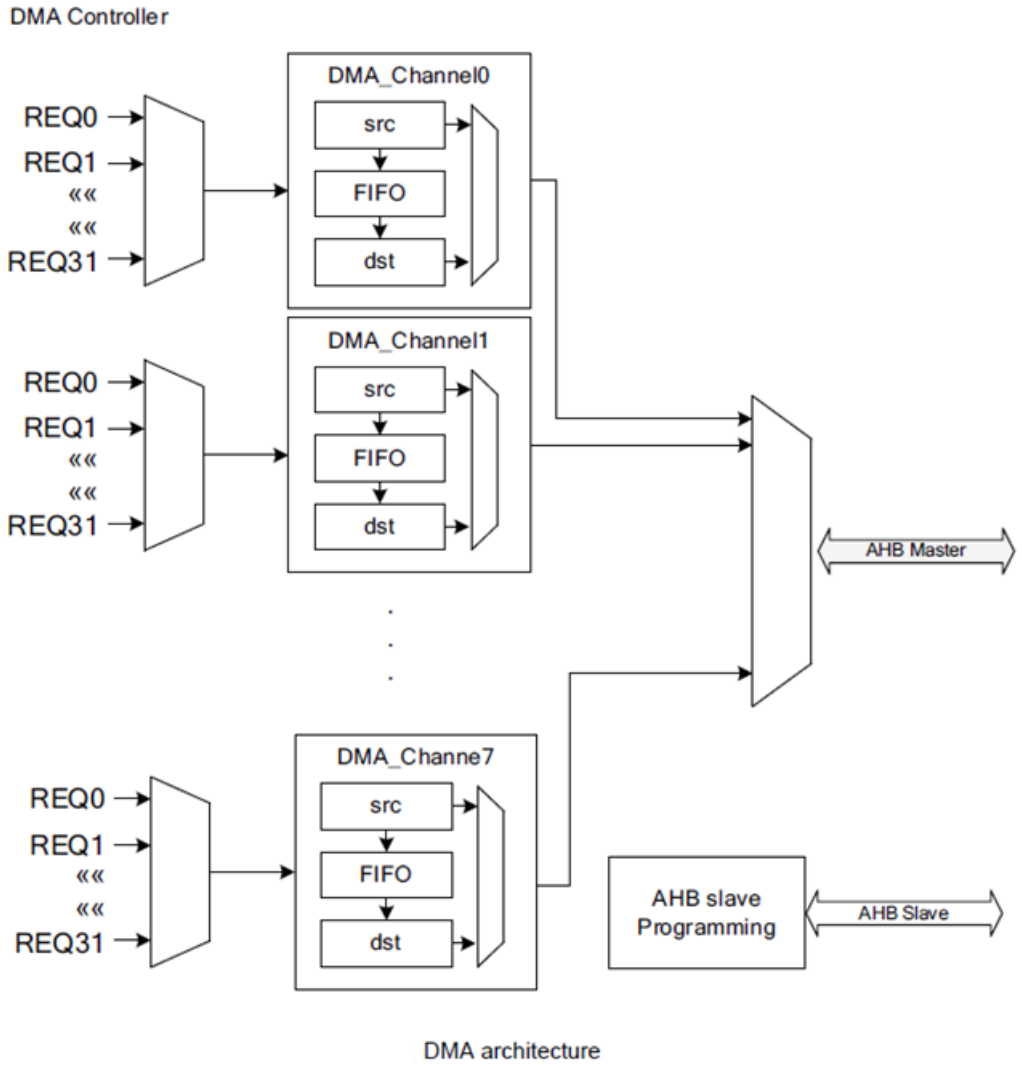
3.13.3.1 Operating principle

When a device attempts to transfer data (usually a large amount of data) directly to another device via the bus, it will first send a DMA request signal to the CPU. The peripheral device makes a bus request to the CPU to take over the bus control right through the DMA. After the CPU receives the signal, after the current bus cycle ends, it will respond to the DMA signal according to the priority of the DMA signal and the order of the DMA request.

When the CPU responds to a DMA request to a device interface, it will give up bus control.

Therefore, under the management of the DMA controller, the peripherals and the memory directly exchange data without CPU intervention. After the data transfer is complete, the device sends a DMA end signal to the CPU, returning the bus control.

Figure 37 DMA Architecture



DMA architecture

The DMA controller includes a set of AHB master interfaces and a set of AHB slave interfaces. The AHB master interface actively accesses memory or peripheral through the system bus according to current configuration requirements as a port of data movement. The AHB slave interface is used to configure DMA interface and only supports 32-bit access.

3.13.3.2 DMA Channel Configuration

DMA supports 4 channels in total, each channel does not interfere with each other and can run at same time. The following lists the configuration process of DMA channel x.

1. Set 32-bit source address in register DMA_C[3:0]SRCADDR.
2. Set 32-bit target address in register DMA_C[3:0]DSTADDR.
3. Configure SI (source) and DI (destination) in DMA_C[3:0]CONTROL to set whether to enable automatic address accumulation.

4. Set the transfer data width STW (source) and DTW (destination) in register DMA_C[3:0]CONTROL . The options are single, double, and four byte.
5. Set burst type, SBS (source) and DBS (destination). The options are single, INCR4, INCR8, and INCR16.
6. A single burst cannot exceed 16-bytes.
7. Set data transmission length range: 0-4095

3.13.3.3 Peripheral support

Peripheral ports are controlled by setting the value of register DMA_C[3:0]CONFIG (0x2600_4[4:1]10), bit[10:6]=DSTPH (Destination peripherals) and bit[5:1]=SRCPH (Source peripherals).

Table 10 Peripheral Port

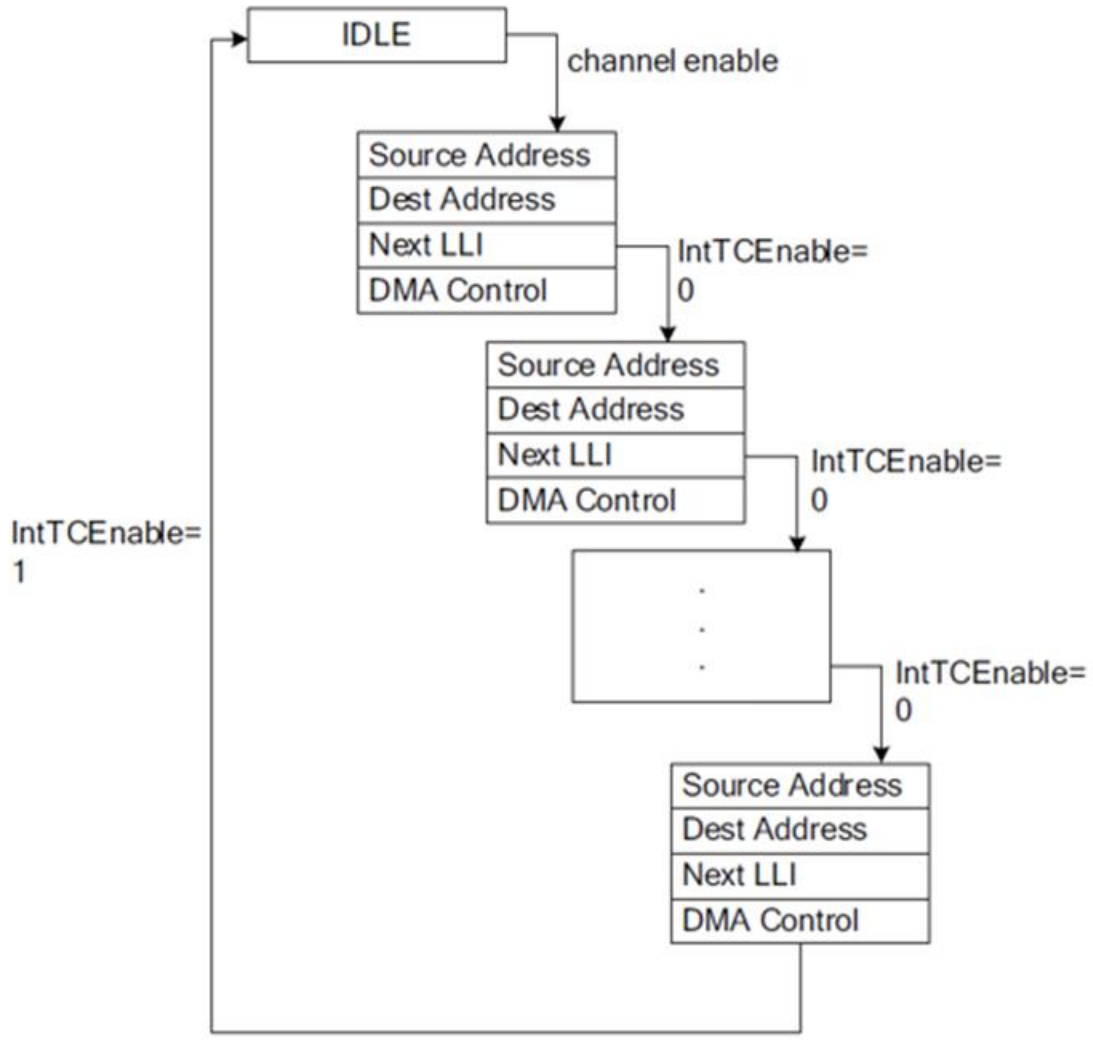
0	1	2	3	4	5	6	7
UART TX	UART RX	SPI0 TX	SPI0 RX	SPI1 TX	SPI1 RX	I2C TX	I2C RX

3.13.3.4 Linked List Mode

DMA supports linked list operation mode: When performing DMA read or write operation, you can fill data in next linked list. After completing the data transfer of current list, read the register DMA_C[3:0]LLI to obtain the start address of the next linked list, and directly transfer the data in the next linked list.

Ensure continuous and uninterrupted work during DMA transfer and improve the efficiency of CPU and DMA.

FIGURE 38 LL1 ARCHITECTURE



3.13.3.5 DMA interrupt

- DMA_INT_TCOMPLETED
    - Data transfer complete interrupt: This interrupt will be generated when a data transfer is completed
- DMA_INT_ERR
    - Data transfer error interrupt: This interrupt will be generated when an error occurs during data transfer

3.13.4 DMA Transmission Mode

3.13.4.1 Memory to memory

After this mode is started, the DMA will move the data from the source address to the destination address according to the set transfer size. After the transfer, the DMA controller will automatically return to the idle state and wait for the next transfer.

The specific configuration process is as follows:

1. Set the value of the register DMA_C[3:0]SRCADDR to the memory address of the source

2. Set the value of the register DMA_C[3:0]DSTADDR to the target memory address

3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 0, that is, select the memory-to-memory mode

4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: set the DI and SI bits to 1 to enable the automatic address accumulation mode, the DTW and STW bits set the transmission width of the source and destination, and the DBS and SBS bits set the burst type of the source and destination

5. Select the appropriate channel, enable DMA, and complete the data transfer

3.13.4.2 Memory to peripheral

In this working mode, the DMA will move data from the source to the internal cache according to the set transfer size (TransferSize). When the cache space is insufficient, the DMA will automatically suspend it. When there is sufficient cache space, continue to transfer until it reaches Set the moving quantity.

On the other hand, when the target peripheral request triggers, it will burst the target configuration to the target address until it reaches the set number of moves and automatically returns to the idle state, waiting for the next startup.

The specific configuration process is as follows:

1. Set the value of the register DMA_C[3:0]SRCADDR to the memory address of the source

2. Set the value of the register DMA_C[3:0]DSTADDR to the target peripheral address

3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 1, that is, select the memory-to-peripheral mode

4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: the SI bit is set to 1 to enable the address auto-accumulation mode, and the DI bit is set to 0 to disable the address auto-accumulation mode, the DTW and STW bits set the transmission width of the source and destination, and the DBS and SBS bits set the burst type of the source and destination

5. Select the appropriate channel, enable DMA, and complete the data transfer

3.13.4.3 Peripheral to memory

In this working mode, when the source peripheral request is triggered, the source configuration is burst to the buffer until the set number of moves reaches the stop. On the other hand, when the internal cache is enough for the target burst number once, the DMA will automatically move the cached content to the target address until it reaches the set number of moves and automatically returns to the idle state, waiting for the next startup

The specific configuration process is as follows:

1. Set the value of the register DMA_C[3:0]SRCADDR to the source peripheral address

2. Set the value of the register DMA_C[3:0]DSTADDR to the target memory address

3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 2, that is, select the peripheral-to-memory mode

4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: the DI bit is set to 1 to enable the address auto-accumulation mode, and the SI bit is set to 0 to disable the address auto-accumulation mode , the DTW and STW bits set the transmission width of the source and destination respectively, and the DBS and SBS bits set the burst type of the source and destination respectively

5. Select the appropriate channel, enable DMA, and complete the data transfer

3.13.4.4 Peripheral to Peripheral

In this working mode, when the source peripheral requests a trigger, the source configuration burst will be stored in the buffer, and it will stop until the set number of moves is reached. On

the other hand, when the internal cache is enough for the target burst number, DMA will automatically move the cached content to the target address until the set number of transfers is reached and automatically return to the idle state, waiting for the next start

The specific configuration process is as follows:

1. Set the value of the register DMA_C[3:0]SRCADDR to the peripheral address of the source

2. Set the value of the register DMA_C[3:0]DSTADDR to the target peripheral address

3. Select the transmission mode and set the value of the FLOWCTRL bit in register DMA_C[3:0]CONFIG to 3, that is, select the peripheral-to-peripheral mode

4. Set the value of the corresponding bit in the DMA_C[3:0]CONTROL register: DI and SI bits are set to 0, the address automatic accumulation mode is disabled, the STW and DTW bits respectively set the source and target transfer widths, and the SBS and DBS bits respectively set the source and target bursts type.

5. Select the appropriate channel, enable DMA, and complete the data transfer

3.14 SEC ENGINE Module

3.14.1 SEC Engine Overview

SEC ENG has a variety of built-in computing modules, including AES, SHA.

3.14.2 SEC Engine Features

- AES
  - Supports 128-bit, 192-bit, and 256-bit key lengths.
  - Supports encryption and decryption of multiple link modes (ECB/CBC/CTR/XTS)
  - Exclusive AES LINK function
- SHA
  - Supports SHA1/SHA224/SHA256/SHA384/SHA512
  - Exclusive SHA LINK function

3.14.3 SEC Engine Functional Description

3.14.3.1 AES Accelerator

3.14.3.1.1 Key

The key length required for encryption mode and decryption mode can be selected by configuring `se_aes_0_mode` and `se_aes_0_dec_en` in the register SE_AES_0_CTRL.

Table 11 AES operation mode diagram

amode	adec en	Operation mode
0	0	AES–128 encryption
1	0	AES–256 encryption
2	0	AES–192 encryption
0	1	AES-128 decryption
1	1	AES-256 decryption
2	1	AES-192 decryption

Select whether to enable the hardware key through `aes_0_hw_key_en` in the register SE_AES_0_CTRL. If you use a software key, you also need to configure the registers SE_AES_0_KEY_0~SE_AES_0_KEY_7 to store the key and each register stores a 4-byte key.

3.14.3.1.2 Link mode

Different link modes can be selected through `se_aes_0_block_mode` in the register SE_AES_0_CTRL. Currently, ECB, CBC, CTR, and XTS modes are supported.

3.14.3.1.3 Plaintext or ciphertext

- Plaintext or ciphertext must be a multiple of 16
- The register SE_AES_0_MSA stores the plaintext address entered during encryption or the ciphertext address entered during decryption
- The register SE_AES_0_MDA stores the ciphertext address output during encryption or the plaintext address output during decryption
- `se_aes_0_msg_len` in the register SE_AES_0_CTRL is used to set the length of ciphertext or plaintext (in units of 16-bytes)

3.14.3.1.4 Initialization vector

The registers SE_AES_0_IV_0~SE_AES_0_IV_3 store the initialization vector (IV). You can choose whether to use a new IV by configuring `aes_0_iv_sel in` register SE_AES_0_CTRL. You must clear 0 when configuring iv for the first time and must set 1 if you continue to use this iv or automatically update iv.

3.14.3.1.5 Encryption and decryption configuration process

- Enable AES with `se_aes_0_en` in the configuration register SE_AES_0_CTRL.
- Configuration register SE_AES_0_ENDIAN, including `se_aes_0_dout_endian`, `se_aes_0_din_endian`, `se_aes_0_key_endian`, `se_aes_0_iv_endian`, and `se_aes_0_twk_endian`. If the value is 0, it means little-endian and if the value is 1, it means big-endian.
- The `se_aes_0_block_mode` in the configuration register SE_AES_0_CTRL selects the link mode.
- The `se_aes_0_mode` in the configuration register SE_AES_0_CTRL selects the key length
- To use software key, configure registers SE_AES_0_KEY_0~SE_AES_0_KEY_7 to store the key. To use a hardware key, set `aes_0_hw_key_en` in register SE_AES_0_CTRL.
- Configure registers SE_AES_0_IV_0~SE_AES_0_IV_3 to set IV, the filling order for MSB is `se_aes_0_iv_0~se_aes_0_iv_3` and the filling order for LSB is `se_aes_0_iv_3~se_aes_0_iv_0`.
- The `se_aes_0_dec_en` in the configuration register SE_AES_0_CTRL selects the encryption or decryption mode.
- The configuration register SE_AES_0_MSA sets the source address of the data to be processed.
- The configuration register SE_AES_0_MDA sets the destination address where the processing result is stored.
- The `se_aes_0_msg_len` in the configuration register SE_AES_0_CTRL sets the length of the data to be processed in units of 16-bytes

- The se_aes_0_trig_1t in the configuration register SE_AES_0_CTRL triggers AES to run.
- The result is output to the destination address specified by register SE_AES_0_MDA.

3.14.3.2 SHA Accelerator

3.14.3.2.1 SHA mode

The se_sha_0_mode in the register SE_SHA_0_CTRL:

0:SHA-256; 1:SHA-224; 2:SHA-1; 3:SHA-1; 4:SHA-512; 5:SHA-384; 6:SHA-512/224; 7:SHA-512/256;

3.14.3.2.2 Plaintext and Ciphertext

- The register SE_SHA_0_MSA stores the plaintext address.
- The registers SE_SHA_0_HASH_L_0~SE_SHA_0_HASH_L_7 stores the ciphertext.

3.14.3.2.3 Operation Flow

- Configure se_sha_0_mode in the register SE_SHA_0_CTRL to set the specific mode of SHA.
- Enable SHA by configuring se_sha_0_en in the register SE_SHA_0_CTRL.
- Configure se_sha_0_hash_sel in the register SE_SHA_0_CTRL. 0 means starting a new HASH calculation, and 1 means using the last result for HASH calculation.
- Configure the register SE_SHA_0_MSA to set the source address of the data to be processed.
- Configure se_sha_0_msg_len in the register SE_SHA_0_CTRL to set the length of the data to be processed (512-bits for SHA1, SHA224, and SHA256, while 1024-bits for SHA512, SHA384, SHA512/224, and SHA512/256)
- Configure se_sha_0_trig_1t in the register SE_SHA_0_CTRL to trigger SHA.
- The output result is stored in registers SE_SHA_0_HASH_L_0~SE_SHA_0_HASH_L_7,
    - MSB: se_sha_0_hash_l_0~se_sha_0_hash_l_7
    - LSB: se_sha_0_hash_l_7~se_sha_0_hash_l_0

3.15 OTP Module

1K x 32-bits One Time Programmable (OTP) device is integrated into Goldfinch's µC subsystem (base address: 0x2a080000, IRQ21).

The OTP system bus APB is connected to µC. `PINTRPT` is an interrupt that is connected to E21 IRQ21. The size of the OTP is 4 KB (1024 by 32-bits) and the system APB bus runs at µC APB speed at 200 MHz.

Table 12 OTP Controller Address

Offset	Size	Usage	Comment
0x8000 ~ 0x8ffc	32K	OTP	eFuse
0x0800 ~ 0x081c		PTR	Test row address pa[0]~pa[7]
0x3400 ~ 0x3508		CFG	CSR in register section

3.16 I3C Module

3.16.1 I3C Overview

An I3C controller is implemented in each chiplet for inter-chiplet synchronization and message passing. µC in each chiplet access register bus of application layer to transmit and receive packets between chiplets.

The controller in I3C system supports Dynamic Address Assignment (DAA). The controller is responsible for generating the clock, issuing commands, and controlling the data transfer. Each I3C device has a unique address assigned by DAA during power up or hot join.

The APB interface is connected to µC's (E21) bus matrix (BMX). I3C transferred through Command/Tx/Rx FIFO. Interrupt is serviced by µC interrupt controller. DMA interface is disabled.

- Application clock (`Pclk`) = µC bus clock (200 MHz)
- Core clock (`core_clk`) = freerun µC bus clock (200 MHz)

Table 13 shows an I3C example.

Table 13 I3C Example

Clock Domain	Minimum Frequency	Typical Frequency	Maximum Frequency
core_clk (core clock)	125 MHz	125 MHz	700 MHz
pclk (application clock)	30 MHz	50 MHz	700 MHz
dma_clk (DMA clock)	30 MHz	50 MHz	700 MHz

I3C controller supports only subset of I2C features. Following are the limitations.

- 10-bit addressing is not supported
- High speed mode is not supported
- SCL is always driven in push-pull
- I2C controller in system is not supported
- Bus clear feature is not supported

3.16.2 I3C Slave provisional ID register

These registers are used in slave mode of operation.

Register: SLV_MIPI_ID_VALUE

Address: 0x2A010070

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	NA	NA	Reserved

Bits	Bit Name	Default	Type	Comment
15:1	SLV_MIPI_MFG_ID	15'h0	R/W	Specifies the MIPI manufacture ID. PID [47:33]. Reset value of this register field is considered from input port signal slv_pid[47:33]. Note: FLC's MIPI Manufacture ID = 15'h0528
0	SLV_PROV_ID_SEL	1'b0	R/W	Specifies the Provisional ID type selector PID [32]. Reset value of this register field is considered from input port signal `slv_pid`[32] 1'b1: Random Value 1'b0: Vendor Fixed Value

Register: SLV_PID_VALUE

Address: 0x2A010074

Bits	Bit Name	Default	Type	Comment
31:16	SLV_PART_ID	16'h0	R/W	Specifies the part ID of `DWC_mipi_i3c` device PID [31:16]. Reset value of this register field is considered from input port signal `slv_pid` [31:12]. Note: FLC's part ID = 16'h0001
15:12	SLV_INST_ID	4'h0	R/W	This field is used to program the instance ID of the slave. Reset value of this register field is considered from input port signal `slv_pid` [15:12]. Note: FLC's instance ID = 4'h0
11:0	SLV_PID_DCR	12'h0	R/W	Specifies the additional 12-bit ID of `DWC_mipi_i3c` device PID [11:0]. Reset value of this register field is considered from input port signal `slv_pid` [11:0]. Note: FLC's additional ID = 12'h0C6

3.16.3 I3C Private read/write opcodes

Below is the format that has been followed for Goldfinch/Mockingbird to define the I3C private read/write commands.

Table 14 I3C private read/write commands

	OCTET 1	OCTET 2	OCTET 3	OCTET 4 (LSB)
WORD 1	ERROR CODE	DATA LENGTH		OPCODE
WORD 2	ADDRESS			
WORD 3	DATA1			
WORD 4	DATA2			

Based on the Data size, the word size can be 4 or 64 words.

The below OPCODES are predefined by FLC and are used as part of the Firmware to define the I3C communication between MKB and GFH.

0x01 READ_REGISTER_OPCODE

0x02 WRITE_REGISTER_OPCODE
0x03 READ_REGISTER_RESPONSE_OPCODE
0x04 WRITE_REGISTER_RESPONSE_OPCODE
0x05 D2D_OPCODE
0x06 D2D_RESP_OPCODE
0x80 FW_DOWNLOAD_COMPLETE_OPCODE
0x81 FW_DOWNLOAD_COMPLETE_RESP_OPCODE

Also note that, based On Opcode, Data format might change. Below are the examples for the READ_REGISTER_OPCODE and D2D_OPCODE.

Table 15 I3C Read Register OPCODE

READ_REGISTER_OPCODE	**OCTET 1**	**OCTET 2**	**OCTET 3**	**OCTET 4 (LSB)**
WORD 1	ERROR CODE	DATA LENGTH		READ_REGISTER_OPCODE
WORD 2	ADDRESS			
WORD 3	DATA1			
WORD 4	DATA2			

Table 16 I3C D2D OPCODE

D2D_OPCODE	**OCTET 1**	**OCTET 2**	**OCTET 3**	**OCTET 4 (LSB)**
WORD 1	ERROR CODE	DATA LENGTH		D2D_OPCODE
WORD 2	D2D SYNC STATUS			
WORD 3	DATA1			
WORD 4	DATA2			

3.17 AXI4 Traffic Generator Module

3.17.1 AXI4 Traffic Generator Overview

The AXI4 traffic generator is a fully synthesizable AXI4-compliant core with the following features.

- Configurable option to generate and accept data according to different traffic profiles.
- Supports dependent/independent transaction between read/write masgter port with configurable delays.
- Programmable repeat count for each transaction with constant/increment/random address.
- External start/stop to generate traffic without processor intervention.
- Generates IP specific traffic on AXI interface for pre-defined protocols.

3.17.2 AXI4 Traffic Generator Sequence

The AXI4 traffic generator sequence:

1. Set `axi4tg_en (0x2600_0004, reg_pmu_ctrl[27]) = 1'b1.`
2. Set `axi4tg_resetn (0x2600_0418, axi4tg_ctl[0]) = 1'b1.`
3. Load `CMDRAM (R/W)`:
   `CMDRAM (RD) 0x2a028_000~0x2a028_fff` (256 commands 128-bit wide),
   `CMDRAM (WR) 0x2a029_000~0x2a029_fff` (256 commands 128-bit wide).
   Note: There should be at least one command with `valid_cmd` bit set to zero (i.e., one invalid command) for both reads and write.
4. Load `PARAM` (Write only):
   `PARAM (RD) 0x2a021_000~0x2a021_3ff` (256 commands 32-bit wide),
   `PARAM (WR) 0x2a021_400~0x2a021_7ff` (256 commands 32-bit wide).
5. Load `MSTRAM` (R/W):
   `0x2a02c_000~0x2a02c_fff.` (128 data 256-bit wide),
   AXI data width = 256-bit (`mstram_index` valid values = 0x0, 0x20, 0x40,…,fe0).
6. Enable the desired interrupt/status bits:
   Set `MSTIRQEN` (0x2a02_000c[31]) and other error enable bits,
   Set `MINTREN` (0x2a02_0010[15]).
7. Start `AXI4TG`:
   Set `MSTEN` (0x2a02_0000[20]) or
   set `axi4tg_start` (0x2600_0418[1]).
8. Check completion:
   Wait for interrupt `axi4tg_irq` (0x2600_0418[16]) or
   Poll for `MSTDONE` (0x2a02_0008[31], write 1'b1 to clear) and check other error status bits or
   Check `MSTEN` (0x2a02_0000[20]), this bit is automatically cleared to indicate to SW that the `AXI4TG` is done.
9. Compare R/W data in `MSTRAM`.

3.18 JTAG Module

GFH has two sets of JTAG ports, one for E21 (nTRST_E21, TCK_E21, TMS_E21, TDI_E21, TDO_E21) and another one for D2D PHY/DDR PHY/TDR2APB (nTRST, TCK, TMS, TDI, TDO).

D2D PHY, DDR PHY and TDR2APB shared one JTAG chain and selected by GPIO[6:7].

Table 17 JTAG Options

GPIO[6:7]	Select JTAG
2'b00	D2D PHY
2'b01	DDR PHY
2'b1x	TDR2APB

3.18.1 D2D PHY JTAG

For `D2D PHY JTAG` loopback test, Apply `GPIO`[15:17] to enable `D2D LL` and wrapper during JTAG test sequence.

- `GPIO[15] -> EXT_PHY_RST_EN`
- `GPIO[16] -> EXT_TX_PHY_RSTN`
- `GPIO[17] -> EXT_RX_PHY_RSTN`

3.18.2 TDR2APB Module

- `JTAG APB TDR`
    - IR: Data reg =1 (write), Status reg (read) =2
    - DR: Data and status reg format:

Table 18 JTAG APB TDR Register Field

Bits	Description
[71]	reset_n Reset jtag bridge.
[70]	start Set high to trigger apb write/read command, set low before start new apb transaction
[69:67]	pprot Apb used
[66]	pwrite Apb used
[65:34]	paddr Apb used
[33:02]	pwdata/prdata Apb used
[01]	pslverr Apb used
[00]	busy Status of apb write/read command

Steps:

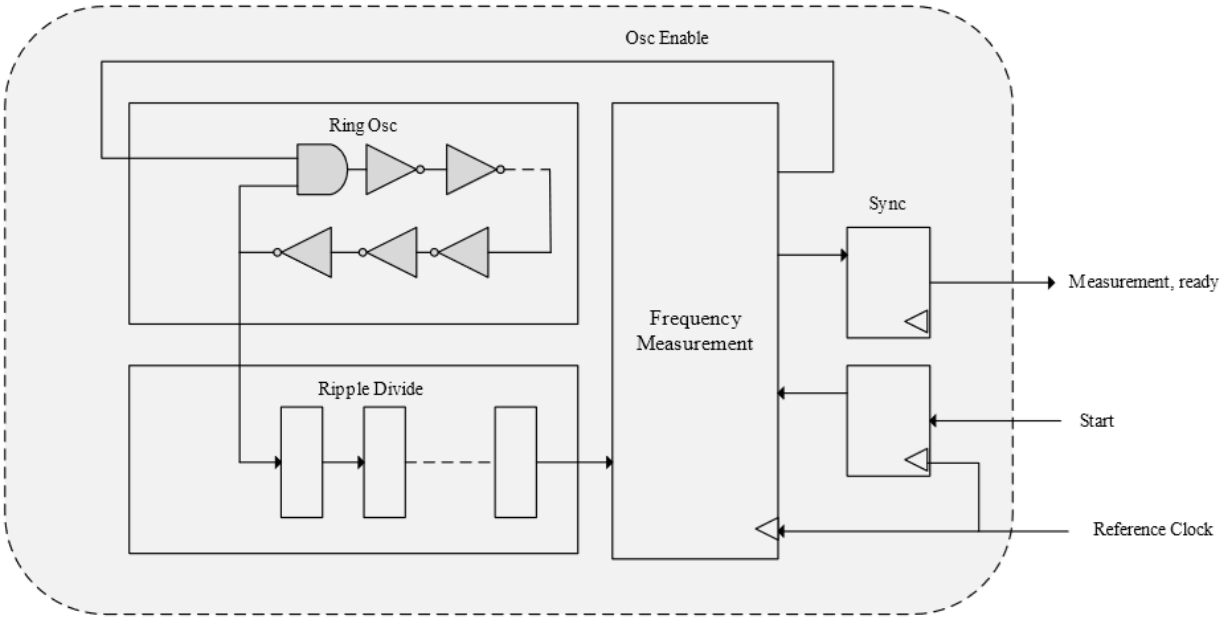
1. Toggle `ntrst`. `TMS=0`. `start` in TDR reset to `0`.
2. Write TDR: {address, pprot, pwrite..}, set `start=1` to start APB cycle.
3. Read TDR, polling for `busy=0`

4. `busy=0` indicates APB cycle completed, {prdata and pslverr} valid.
5. Write TDR with `start=0`.
6. Back to step 2 for more APB transactions.

3.19 DRO Module

Six DRO modules are included for process monitoring.

FIGURE 39 DRO FOR PROCESS MONITOR



Reference clock: 200MHz (`uc_clk`)

rosc chain: 83 stages, typical case, period= ~938ps

- Multiple Vt and/or combined cells. Harden at PAR stage.
- Ring stopped automatically after measurement complete (ready=1).

Multiple instances at different locations.

Ripple: 8 stage, period = 938ps*256 = 240128ps

Period count by 200Mhz clock: measurement = 240128/5000 = ~ 48

CSR bit

- Start (R/W):
  0x2600_0800[0] = dro0_start; 0x2600_0800[16] = dro1_start;
  0x2600_0804[0] = dro2_start; 0x2600_0804[16] = dro3_start;
  0x2600_0808[0] = dro4_start; 0x2600_0808[16] = dro5_start;
- Measurement Ready (RO):
  0x2600_0800[1] = dro0_ready; 0x2600_0800[17] = dro1_ready;
  0x2600_0804[1] = dro2_ready; 0x2600_0804[17] = dro3_ready;
  0x2600_0808[1] = dro4_ready; 0x2600_0808[17] = dro5_ready;

- Measurement result (RO):
  0x2600_0800[15:8] = dro0_mea; 0x2600_0800[31:24] = dro1_mea;
  0x2600_0804[15:8] = dro2_mea; 0x2600_0804[31:24] = dro3_mea;
  0x2600_0808[15:8] = dro4_mea; 0x2600_0808[31:24] = dro5_mea;

3.20 External Clock Source Examples

The following table shows the external clock source examples

Table 19 External Clock Source examples

Frequency	Part Number	Comment
25MHz	ASD3-25.000MHZ-LR-T	±25ppm
100MHz	Si52204-A01BGMR	0.085ps rms, PCIe Gen 6

Chapter 4 Address Space

This chapter lists the address Map.

The following table shows the Address map within the microcontroller.

Table 20 μC Address Map

Start Address	End Address	Size	Description	Comment
0x0000_0000	0x0000_0fff	4K	Debug controller control	
0x0000_3000	0x0000_3fff	4K	Error-device	
0x0000_4000	0x0000_4fff	4K	Test status	
0x0170_0000	0x0170_0fff	4K	Bus error unit	
0x0200_0000	0x02ff_ffff	16M	Interrupt controller	
0x2000_0000	0x2000_0fff	4K	SPI0	
0x2000_1000	0x2000_1fff	4K	SPI1	
0x2000_2000	0x2000_2fff	4K	I2C	I2C host
0x2000_3000	0x2000_3fff	4K	LTMR	Local timer, WDT
0x2000_4000	0x2000_4fff	4K	UART	
0x2000_5000	0x2000_5fff	4K	GPIO	
0x2000_6000	0x2000_ffff	40K	Reserved	
0x2100_0000	0x210f_ffff	1M	DDR4 MC	DDR4 MC CSR
0x2200_0000	0x23ff_ffff	32M	DDR4 PHY	DDR4 PHY CSR
0x2600_0000	0x2600_0fff	4K	System Control register	
0x2600_1000	0x2600_1fff	4K	SF_CTRL	SPI serial flash controller, SRAM buffer offset: 0x600
0x2600_2000	0x2600_2fff	4K	L1C	L1C cache controller for XIP flash
0x2600_4000	0x2600_4fff	4K	DMA	DMA controller
0x2600_5000	0x2600_5fff	4K	SEC	Security module
0x2608_0000	0x2608_ffff	64K	OTP	1Kx32 OTP
0x2700_0000	0x270f_ffff	1M	D2D_PHY	D2D PHY
0x2710_0000	0x27ff_ffff	15M	D2D_IO	D2D_IO
0x2a00_0000	0x2a00_ffff	64K	I3C Master	I3C master
0x2a01_0000	0x2a01_ffff	64K	I3C Slave	I3C slave
0x2a02_0000	0x2a02_ffff	64K	AXI4TG	AXI4 traffic generator
0x3000_0000	0x300f_ffff	1M	Boot ROM	Boot ROM up to 1 MB
0x3010_0000	0x3fff_ffff	255M	XIP NOR flash	XIP access
0x8000_0000	0x8001_ffff	128K	TIM0	

Start Address	End Address	Size	Description	Comment
0x8002_0000	0x8003_ffff	128K	TIM1	

Chapter 5 µC IRQ Map

This chapter lists interrupts present in the Microcontroller.

The following table shows the IRQ map within the microcontroller.

Table 21 µC IRQ Map

IRQ	Device	Comment
0	UART	Refer to 3.9.3.12 UART Interrupt and register UART_INT_STS
1	SPI0	Refer to 3.6.3.11 SPI Interrupt and register SPI_INT_STS
2	SPI1	Refer to 3.6.3.11 SPI Interrupt and register SPI_INT_STS
3	I2C	Refer to 3.7.9 I2C Interrupt and register I2C_INT_STS
4	Reserved	Reserved for local timer 1
5	TMR2	Local timer 2
6	TMR3	Local timer 3
7	GPIO	
8	DMA	Refer to 3.13.3.5 DMA interrupt
9	i3c_master_ic_intr	I3C Master
10	i3c_slave_ic_intr	I3C Slave
16-11	Reserved	
17	SEC-SHA	SEC-SHA
18	SEC-AES	SEC-AES
19	SEC-TRNG	SEC-TRNG
20	SF_CTRL	XIP/SF controller
21	OTP	Refer to register OFS_INTRPT
22	L1C	
23	BMX_Timeout	Bus Matrix timeout
24	BMX_Error	Bus Matrix error
25	DMA_interr	DMA_interr
26	DMA_inttc	DMA_inttc
35-27	Reserved	
36	DDR4 MC	DDR4 MC
37	DDR4 PHY	DDR4 PHY
39-38	Reserved	
40	D2D_DDR	D2D_DDR
56-41	Reserved	
57	ts_irq	Temperature sensor

IRQ	Device	Comment
58	vs_irq	Voltage monitor
61-59	Reserved	Reserved
62	Watchdog Timer	Local timer WDT
127-63	Reserved	
NMI	Watchdog Timer	Local timer WDT

Chapter 6 Registers

6.1 Summary of Registers

CSR registers are µC controlled and selected by µC address map. Refer to **Table 20 µC Address Map** for complete address space assignment.

The following table shows the summary of registers.

Table 22 Summary of registers

System Control Register 0x2600_0000 ~ 0x2600_0FFF		
Register Name	**Offset**	**Reset Value**
GPIO_POS_REG	0x0	N/A
PMU_CTRL	0x4	0x40000000
SYS_CTRL	0x8	0x00000000
RESET_INFO	0x10	0x00000000
BMX_CONFIG	0x20	0x00000000
BMX_ERR_ADDR	0x24	0x00000000
D2D_PHY_CTRL	0x38	0x00000100
SE_AES_KEY	0x40~0x5C	0x00000000
D2D_IRQ	0x60	0x00000000
GFH_PLL_CTRL1	0x100	0x00000010
GFH_PLL_CTRL2	0x104	0x043C2801
GFH_PLL_CTRL3	0x108	0x30000000
AXI4TG_CTRL	0x418	0x00000000
I3C_DEBUG_PORT0	0x41C	0x8000555A
I3C_DEBUG_PORT1	0x420	0x40000000
I3CS_CTRL1	0x424	0x00000000
I3CS_CTRL2	0x428	0x00000000
I3CS_CTRL3	0x42C	0x00000000
I3CS_DEBUG_PORT0	0x68C	0x00005140
I3CS_DEBUG_PORT1	0x690	0x00000000
D2D_PLL_CTRL1	0x700	0x10000011
D2D_PLL_CTRL2	0x704	0x00001401
D2D_PLL_CTRL3	0x708	0x00000000
D2D_PLL_CTRL4	0x70C	0x00000000
D2D_PLL_STATUS	0x710	0x00000000
TS_CTRL	0x760	0x00004000
TS_OUT	0x764	0x00000000
VS_CTRL	0x780	0x00400000

VS_OUT	0x784	0x00000000
LVDS_CTRL	0x790	0x00000089
DRO_0_1	0x800	0x00000000
DRO_2_3	0x804	0x00000000
DRO_4_5	0x808	0x00000000
UC_SYSCTRL_PID0	0xFE0	0x0C890002
UC_SYSCTRL_PID1	0xFE4	0x00000000
UC_SYSCTRL_CID0	0xFF0	0x494E4348
UC_SYSCTRL_CID1	0xFF4	0x4F4C4446
UC_SYSCTRL_CID2	0xFF8	0x20202047
UC_SYSCTRL_CID3	0xFFC	0x464C4354
Serial Flash Control Register 0x2600_1000 ~ 0x2600_1FFF		
Register Name	**Offset**	**Reset Value**
SF_CTRL_0	0x0	0x1AF2001C
SF_CTRL_1	0x4	0xF3600000
SF_IF_SAHB_0	0x8	0x0D0403FC
SF_IF_SAHB_1	0xC	0x03000000
SF_IF_SAHB_2	0x10	0x00000000
SF_IF_IAHB_0	0x14	0x0D040000
SF_IF_IAHB_1	0x18	0x03000000
SF_IF_IAHB_2	0x1C	0x00000000
SF_IF_STATUS_0	0x20	0x00000000
SF_IF_STATUS_1	0x24	0x20000000
SF_AHB2SIF_STATUS	0x2C	0x01010003
SF_IF_IO_DLY_0	0x30	0x00000000
SF_IF_IO_DLY_1	0x34	0x00000000
SF_IF_IO_DLY_2	0x38	0x00000000
SF_IF_IO_DLY_3	0x3C	0x00000000
SF_IF_IO_DLY_4	0x40	0x00000000
SF_RESERVED	0x44	0x0000FFFF
SF_CTRL_2	0x70	0x00000000
SF_CTRL_3	0x74	0x2010C000
SF_ID0_OFFSET	0xA0	0x00000000
SF_DBG	0xB0	0x00000001
SF_CTRL_PROT_EN_RD	0x100	0x00000000
SF_SRAM	0x600 ~ 0x67F	N/A
L1C Control Register 0x2600_2000 ~ 0x2600_2FFF		
Register Name	**Offset**	**Reset Value**

L1C_CONFIG	0x0	0x00000000
L1C_HIT_CNT_LSB	0x4	0x00000000
L1C_HIT_CNT_MSB	0x8	0x00000000
L1C_MISS_CNT	0xC	0x00000000
L1C_AUX_CONFIG	0x10	0x00000000
DMA Register 0x2600_4000 ~ 0x2600_4FFF		
Register Name	**Offset**	**Reset Value**
DMA_INTSTATUS	0x0	0x00000000
DMA_INTTCSTATUS	0x4	0x00000000
DMA_INTTCCLEAR	0x8	0x00000000
DMA_INTERRORSTATUS	0xC	0x00000000
DMA_INTERRCLR	0x10	0x00000000
DMA_RAWINTTCSTATUS	0x14	0x00000000
DMA_RAWINTERRORSTATUS	0x18	0x00000000
DMA_ENBLDCHNS	0x1C	0x00000000
DMA_SOFTBREQ	0x20	0x00000000
DMA_SOFTSREQ	0x24	0x00000000
DMA_SOFTLBREQ	0x28	0x00000000
DMA_SOFTLSREQ	0x2C	0x00000000
DMA_CONFIG	0x30	0x00000000
DMA_SYNC	0x34	0x00000000
DMA_C[3:0]SRCADDR	0x[4:1]00	0x00000000
DMA_C[3:0]DSTADDR	0x[4:1]04	0x00000000
DMA_C[3:0]LLI	0x[4:1]08	0x00000000
DMA_C[3:0]CONTROL	0x[4:1]0C	0x0C489000
DMA_C[3:0]CONFIG	0x[4:1]10	0x00000000
Security Engine Control Register 0x2600_5000 ~ 0x2600_5FFF		
Register Name	**Offset**	**Reset Value**
SE_SHA_0_CTRL	0x0	0x00000000
SE_SHA_0_MSA	0x4	0x00000000
SE_SHA_0_STATUS	0x8	0x00000041
SE_SHA_0_ENDIAN	0xC	0x00000001
SE_SHA_0_HASH_L_0	0x10	0x00000000
SE_SHA_0_HASH_L_1	0x14	0x00000000
SE_SHA_0_HASH_L_2	0x18	0x00000000
SE_SHA_0_HASH_L_3	0x1C	0x00000000
SE_SHA_0_HASH_L_4	0x20	0x00000000
SE_SHA_0_HASH_L_5	0x24	0x00000000
SE_SHA_0_HASH_L_6	0x28	0x00000000

SE_SHA_0_HASH_L_7	0x2C	0x00000000
SE_SHA_0_CTRL_PROT	0xFC	0x00000006
SE_AES_0_CTRL	0x100	0x00000000
SE_AES_0_MSA	0x104	0x00000000
SE_AES_0_MDA	0x108	0x00000000
SE_AES_0_STATUS	0x10C	0x00004100
SE_AES_0_IV_0	0x110	0x00000000
SE_AES_0_IV_1	0x114	0x00000000
SE_AES_0_IV_2	0x118	0x00000000
SE_AES_0_IV_3	0x11C	0x00000000
SE_AES_0_KEY_0	0x120	0x00000000
SE_AES_0_KEY_1	0x124	0x00000000
SE_AES_0_KEY_2	0x128	0x00000000
SE_AES_0_KEY_3	0x12C	0x00000000
SE_AES_0_KEY_4	0x130	0x00000000
SE_AES_0_KEY_5	0x134	0x00000000
SE_AES_0_KEY_6	0x138	0x00000000
SE_AES_0_KEY_7	0x13C	0x00000000
SE_AES_0_ENDIAN	0x148	0x0000000F
SE_TRNG_0_CTRL_0	0x200	0x00000000
SE_TRNG_0_STATUS	0x204	0x00040020
SE_TRNG_0_DOUT_0	0x208	0x00000000
SE_TRNG_0_DOUT_1	0x20C	0x00000000
SE_TRNG_0_DOUT_2	0x210	0x00000000
SE_TRNG_0_DOUT_3	0x214	0x00000000
SE_TRNG_0_DOUT_4	0x218	0x00000000
SE_TRNG_0_DOUT_5	0x21C	0x00000000
SE_TRNG_0_DOUT_6	0x220	0x00000000
SE_TRNG_0_DOUT_7	0x224	0x00000000
SE_TRNG_0_TEST	0x228	0x00000000
SE_TRNG_0_CTRL_1	0x22C	0x0000FFFF
SE_TRNG_0_CTRL_2	0x230	0x000000FF
SE_TRNG_0_CTRL_3	0x234	0x80000000
SE_TRNG_0_TEST_OUT_0	0x240	0x00000000
SE_TRNG_0_TEST_OUT_1	0x244	0x00000000
SE_TRNG_0_TEST_OUT_2	0x248	0x00000000
SE_TRNG_0_TEST_OUT_3	0x24C	0x00000000
SE_CTRL_RESERVED_0	0xF04	0x00000000
SE_CTRL_RESERVED_1	0xF08	0xFFFFFFFF
SE_CTRL_RESERVED_2	0xF0C	0x00000000
OTP Register 0x2608_0000 ~ 0x2608_FFFF		

Register Name	Offset	Reset Value
OFS_VERSION	0x3400	0x20230613
OFS_PART_NUM	0x3404	0x4547512D
OFS_INTRPT	0x3408	0x00000000
OFS_STATUS	0x3500	0x00000000
OFS_DEEP	0x3504	0x40000001
OFS_CONFIG	0x3508	0x00000000
SPI0 Control Register 0x2000_0000 ~ 0x2000_0FFF **SPI1 Control Register 0x2000_1000 ~ 0x2000_1FFF**		
Register Name	**Offset**	**Reset Value**
SPI_CONFIG	0x0	0x00000000
SPI_INT_STS	0x4	0x3F003F02
SPI_BUS_BUSY	0x8	0x00000000
SPI_PRD_0	0x10	0xF0F0F0F
SPI_PRD_1	0x14	0x0000000F
SPI_RXD_IGNR	0x18	0x00000000
SPI_STO_VALUE	0x1C	0x00000FFF
SPI_FIFO_CONFIG_0	0x80	0x00000000
SPI_FIFO_CONFIG_1	0x84	0x00000004
SPI_FIFO_WDATA	0x88	0x00000000
SPI_FIFO_RDATA	0x8C	0x00000000
BACKUP_IO_EN	0xFC	0x00000000
I2C Control Register 0x2000_2000 ~ 0x2000_2FFF		
Register Name	**Offset**	**Reset Value**
I2C_CONFIG	0x0	0x0000000A
I2C_INT_STS	0x4	0x3F003F02
I2C_SUB_ADDR	0x8	0x00000000
I2C_BUS_BUSY	0xC	0x00000000
I2C_PRD_START	0x10	0xF0F0F0F
I2C_PRD_STOP	0x14	0xF0F0F0F
I2C_PRD_DATA	0x18	0xF0F0F0F
I2C_FIFO_CONFIG_0	0x80	0x00000000
I2C_FIFO_CONFIG_1	0x84	0x00000002
I2C_FIFO_WDATA	0x88	0x00000000
I2C_FIFO_RDATA	0x8C	0x00000000
Local Timer Register 0x2000_3000 ~ 0x2000_3FFF		

Register Name	Offset	Reset Value
TCCR	0x0	0x00000000
TMR2_0	0x10	0xFFFFFFFF
TMR2_1	0x14	0xFFFFFFFF
TMR2_2	0x18	0xFFFFFFFF
TMR3_0	0x1C	0xFFFFFFFF
TMR3_1	0x20	0xFFFFFFFF
TMR3_2	0x24	0xFFFFFFFF
TCR2	0x2C	0x00000000
TCR3	0x30	0x00000000
TMSR2	0x38	0x00000000
TMSR3	0x3C	0x00000000
TIER2	0x44	0x00000000
TIER3	0x48	0x00000000
TPLVR2	0x50	0x00000000
TPLVR3	0x54	0x00000000
TPLCR2	0x5C	0x00000000
TPLCR3	0x60	0x00000000
WMER	0x64	0x00000000
WMR	0x68	0x0000FFFF
WVR	0x6C	0x00000000
WSR	0x70	0x00000000
TICR2	0x78	0x00000000
TICR3	0x7C	0x00000000
WICR	0x80	0x00000000
TCER	0x84	0x00000000
TCMR	0x88	0x00000000
TILR2	0x90	0x00000000
TILR3	0x94	0x00000000
TCVWR2	0xA8	0x00000000
TCVWR3	0xAC	0x00000000
TCVSTN2	0xB4	0x00000000
TCVSTN3	0xB8	0x00000000
TCDR	0xBC	0x00000000
UART Control Register 0x2000_4000 ~ 0x2000_4FFF		
Register Name	**Offset**	**Reset Value**
UTX_CONFIG	0x0	0x00008F00
URX_CONFIG	0x4	0x00000700
UART_BIT_PRD	0x8	0x00FF00FF
DATA_CONFIG	0xC	0x00000000

UTX_IR_POSITION	0x10	0x009F0070
URX_IR_POSITION	0x14	0x0000006F
URX_RTO_TIMER	0x18	0x0000000F
UART_SW_MODE	0x1C	0x00000000
UART_INT_STS	0x20	0x00000004
UART_INT_MASK	0x24	0x00000FFF
UART_INT_CLR	0x28	0x00000000
UART_INT_EN	0x2C	0x00000FFF
UART_STATUS	0x30	0x00000000
STS_URX_ABR_PRD	0x34	0x00000000
URX_ABR_PRD_B01	0x38	0x00000000
URX_ABR_PRD_B23	0x3C	0x00000000
URX_ABR_PRD_B45	0x40	0x00000000
URX_ABR_PRD_B67	0x44	0x00000000
URX_ABR_PW_TOL	0x48	0x00000003
URX_BCR_INT_CFG	0x50	0x0000FFFF
UTX_RS485_CFG	0x54	0x00000002
UART_FIFO_CONFIG_0	0x80	0x00000000
UART_FIFO_CONFIG_1	0x84	0x00000080
UART_FIFO_WDATA	0x88	0x00000000
UART_FIFO_RDATA	0x8C	0x00000000
GPIO Control Register 0x2000_5000 ~ 0x2000_5FFF		
Register Name	**Offset**	**Reset Value**
GPIO_MISC	0x80	0x00000000
GPIO_CFGCTL0	0x100	0x00030003
GPIO_CFGCTL1	0x104	0x00030003
GPIO_CFGCTL2	0x108	0x00030003
GPIO_CFGCTL3	0x10C	0x00030003
GPIO_CFGCTL4	0x110	0x00030003
GPIO_CFGCTL5	0x114	0x00030003
GPIO_CFGCTL6	0x118	0x00030003
GPIO_CFGCTL7	0x11C	0x00030003
GPIO_CFGCTL8	0x120	0x00030003
GPIO_CFGCTL9	0x124	0x00030003
GPIO_CFGCTL30	0x180	0x00000000
GPIO_CFGCTL32	0x188	0x00000000
GPIO_CFGCTL34	0x190	0x00000000
GPIO_INT_MASK1	0x1A0	0x000FFFFF
GPIO_INT_STAT1	0x1A8	0x00000000
GPIO_INT_CLR1	0x1B0	0x00000000
GPIO_INT_MODE_SET1	0x1C0	0x00000000

GPIO_INT_MODE_SET2	0x1C4	0x00000000

DDR4 MC Register 0x2100_0000 ~ 0x210F_FFFF	here
DDR4 PHY Register 0x2200_0000 ~ 0x23FF_FFFF	here
D2D PHY Register 0x2700_0000 ~ 0x270F_FFFF	here
D2D Register 0x2710_0000 ~ 0x27FF_FFFF	here
I3C Master Register 0x2A000000 ~ 0x2A00_FFFF	here
I3C Slave Register 0x2A010000 ~0x2A01_FFFF	here
AXI4TG Register 0x2A020000 ~0x2A02_FFFF	here

6.2 System Control Register

System Control CSR address = 0x2600_0000 ~ 0x2600_0FFF

Register: GPIO_POS_REG

Address: 0x26000000

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19:0	gpio_pos_reg	N/A	R	Sample value of GPIO [19:0] at chiplet reset.

Register: PMU_CTRL

Address: 0x26000004

Bits	Bit Name	Default	Type	Comment
31:30	ddr_clock_ratio	2'b01	R/W	DDR clock ratio 00 = 1:1 01 = 1:2 1X = Reserved
29:28	Reserved	2'b0	R/W	Reserved
27	axi4tg_en	1'b0	R/W	0 = AXI4TG in reset state 1 = AXI4TG enabled
26	Reserved	1'b0	R/W	Reserved for ics map to gpio [18:17]
25	config_spi_en	1'b0	R/W	config_spi map to gpio [16:14]
24	spi1_en	1'b0	R/W	SPI1 map to gpio [13:10] 0 = SPI1 in reset state 1 = SPI1 enabled
23	spi0_en	1'b0	R/W	SPI0 map to gpio [9:8] 0 = SPI0 in reset state 1 = SPI0 enabled
22	i2c_en	1'b0	R/W	I2C map to gpio [7:4] 0 = I2C in reset state

Bits	Bit Name	Default	Type	Comment
				1 = I2C enabled
21:16	Reserved	6'b0	R/W	Reserved
15:12	sf_io_pullup_en	4'b0	R/W	[15]: sf_io [3] pull up_en [14]: sf_io [2] pull up_en [13]: sf_io [1] pull up_en [12]: sf_io [0] pull up_en
11:9	Reserved	3'b0	R/W	Reserved
8	d2d_ddr_rstn	1'b0	R/W	0 = D2D in reset state 1 = D2D enabled
7:1	Reserved	7'b0	R/W	Reserved
0	uc_hclk_halt_en	1'b0	R/W	1 = Stop μC hclk when WFI

Register: SYS_CTRL

Address: 0x26000008

Bits	Bit Name	Default	Type	Comment
31:15	Reserved	17'b0	R/W	Reserved
14	spi1_master_mode	1'b0	R/W	1 = Set SPI1 IO pads as master mode
13:4	AHB_NS_ACCESS_DIS	10'b0	R/W	1 = Disable AHB device access in non-secure mode [13]: XIP [12]: Boot ROM [11]: SEC [10]: DMA [9]: SYSCTRL [8]: AHB1 (D2D, DDR4, AXI4TG, I3C) [7]: OTP [6]: L1C [5]: SFCTRL [4]: AHB0 (SPI, I2C, LTMR, UART, GPIO)
3	force_dma_active	1'b0	R/W	1 = Force core debug module active
2	simulation_speedup	1'b0	R/W	1 = Speed up to 32 KHz clock for simulation.
1	wdogirqnmi	1'b0	R/W	WDOG Interrupt CPU NMI Enable
0	lockupreset	1'b0	R/W	CPU Lockup Reset

Register: RESET_INFO

Address: 0x26000010

Bits	Bit Name	Default	Type	Comment
31:4	Reserved	28'b0	R	Reserved, always read 'b0
3	boot_xip	N/A	R	boot_xip pin

Bits	Bit Name	Default	Type	Comment
2	LOCKUPRESET	1'b0	R/W1c	Write 1 to clear
1	WDOGRESETREQ	1'b0	R/W1c	Write 1 to clear
0	SYSRESETREQ	1'b0	R/W1c	Write 1 to clear

Register: BMX_CONFIG

Address: 0x26000020

Bits	Bit Name	Default	Type	Comment
31:27	Reserved	5'b0	R	Reserved, always read 'b0
26	bmx_timeout	1'b0	R/Wc	BMX time out latch/interrupt, write to clear
25	bmx_err_dec	1'b0	R	BMX error
24	bmx_err_addr_dis	1'b0	R/W	Disable BMX address error
23:20	bmx_dbg_sel	4'b0	R/W	BMX debug mux select
19	bmx_busy_option_dis	1'b0	R/W	Ignore HTRANS busy
18	bmx_err_dis	1'b0	R/W	Disable error response
17:14	bmx_ slave _timeout_en	4'b0	R/W	Bus timeout enable, 1-bit per slave 1XXX = S3 X1XX = S2 (Not Used in GFH) XX1X = S1 XXX1 = S0
13:10	bmx_arb_mode	4'b0	R/W	Arbitration priority
9:0	bmx_master_hsel	10'b0	R/W	Ignore waiting for slave's HREADYOUT for better timing, 1-bit per master 1XXXXXXXXX = M9 (Not Used in GFH) X1XXXXXXXX = M8 XX1XXXXXXX = M7 XXX1XXXXXX = M6 (Not Used in GFH) XXXX1XXXXX = M5 XXXXX1XXXX = M4 XXXXXX1XXX = M3 XXXXXXX1XX = M2 XXXXXXXX1X = M1 XXXXXXXXX1 = M0

Register: BMX_ERR_ADDR

Address: 0x26000024

Bits	Bit Name	Default	Type	Comment
31:0	bmx_err_addr	32'h0	R	Access error address

Register: D2D_PHY_CTRL

Address: 0x26000038

Bits	Bit Name	Default	Type	Comment
31:9	Reserved	23'b0	R/W	Reserved
8	d2d_phy_sel	1'b1	R/W	D2D PHY select
7:0	Reserved	8'b0	R/W	Reserved

Register: SE_AES_KEY

Address: 0x26000040~0x2600005C

Bits	Bit Name	Default	Type	Comment
255:0	se_aes_key	256'h0	R/W	Security engine key

Register: D2D_IRQ

Address: 0x26000060

Bits	Bit Name	Default	Type	Comment
31:9	Reserved	23'b0	R	Reserved, always read 'b0
8	d2d_irq_i	1'b0	R	D2D NMI out
7:1	Reserved	7'b0	R	Reserved, always read 'b0
0	d2d_irq_o	1'b0	R/W	D2D NMI in

Register: GFH_PLL_CTRL1

Address: 0x26000100

Bits	Bit Name	Default	Type	Comment
31	DESKEW_PLL_LOCKED	1'b0	R	Deskew calibration settled
30	PLL_LOCKED	1'b0	R	PLL locked
29	frefcmlen	1'b0	R/W	Enable free CML input
28	foutvcoen	1'b0	R/W	VCO rate output clock enable
27:23	foutvcobyp	5'b0	R/W	Bypass undivided vco clock to specific output,
22	foutdiffen	1'b0	R/W	Enable [FOUTDIFFP/N] (frequency is FVCO/POSTDIV4)
21	foutcmlen	1'b0	R/W	Enable FOUTCML[P/N]
20	dmsen	1'b0	R/W	Enable delta-sigma modulator. 0 = DSM is powered down (integer mode). 1 = DSM is powered up (fractional mode)
19	dskewfastcal	1'b0	R/W	Deskew fast calibration enable.
18:7	dskewcalin	12'b0	R/W	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration

Bits	Bit Name	Default	Type	Comment
6	dskewcalen	1'b0	R/W	Deskew calibration enable
5:3	dskewcalcnt	3'b010	R/W	Counter for deskew calibration loop
2	dskewcalbyp	1'b0	R/W	Deskew calibration bypass
1	dac_en	1'b0	R/W	Enabling fractional noise canceling DAC
0	pll_en	1'b0	R/W	PLL enable

Register: GFH_PLL_CTRL2

Address: 0x26000104

Bits	Bit Name	Default	Type	Comment
31:28	posdiv2	4'b0	R/W	PLL post divider 2 (not used)
27:24	posdiv1	4b0100	R/W	PLL post divider 1 4GHz/5 = 800MHz
23:20	posdiv0	4'b0011	R/W	PLL post divider 0 4GHz/4 = 1GHz
19:18	posdiv4	2'b11	R/W	PLL post divider 4 4GHz/12 (not used)
17:6	fbdiv	12'hA0	R/W	Feedback clock divider 25 MHz*160 = 4 GHz VCO
5:0	refdiv	6'd1	R/W	Reference clock divider

Register: GFH_PLL_CTRL3

Address: 0x26000108

Bits	Bit Name	Default	Type	Comment
31:28	fouten	4'b0011	R/W	Bit-wise post divide enable
27:24	posdiv3	4'b0	R/W	PLL post divider 3 (not used)
23:0	frac	24'h0	R/W	Fractional value of feed- back divider

Register: AXI4TG_CTRL

Address: 0x26000418

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	14'b0	R/W	Reserved
17	axi4tg_err	1'b0	R	1 = Error detected
16	axi4tg_irq	1'b0	R	1 = Traffic generation completion
15:2	Reserved	14'b0	R/W	Reserved
1	axi4tg_start	1'b0	R/W	Start generating or accepting the traffic
0	axi4tg_resetn	1'b0	R/W	Active-Low reset

Register: I3C_DEBUG_PORT0

Address: 0x2600041C

Bits	Bit Name	Default	Type	Comment
31:0	i3cm_debug_port [31:0]	32'h8000555A	R	I3C master debug ports[31:0]

Register: I3C_DEBUG_PORT1

Address: 0x26000420

Bits	Bit Name	Default	Type	Comment
31	wakeup	1'b0	R	I3C slave rxc wakeup
30	i2c_glitch_filter_en	1'b1	R	I3C slave I2C glitch filter enable
29:23	Reserved	7'b0	R	Reserved, always read 'b0
22:0	i3cm_debug_port [54:32]	23'h0	R	I3C master debug ports[54:32]

Register: I3CS_CTRL1

Address: 0x26000424

Bits	Bit Name	Default	Type	Comment
31	Reserved	1'b0	R/W	Reserved
30:23	i3cs_slv_dcr[7:0]	8'b0	R/W	Device Characteristic Register value
22:20	i3cs_slv_max_wr_speed[2:0]	3'b0	R/W	Slave maximum write data rate
19:17	i3cs_slv_max_rd_speed[2:0]	3'b0	R/W	Slave maximum read data rate
16:10	i3cs_static_addr[6:0]	7'b0	R/W	Slave static address
9	i3cs_static_addr_en	1'b0	R/W	Slave static address valid
8:5	i3cs_pending_int[3:0]	4'b0	R/W	Pending interrupt information
4:3	i3cs_act_mode[1:0]	2'b0	R/W	Slave activity mode
2	i3cs_slv_test_mode	1'b0	R/W	Slave test mode
1	i3cs_mode_i2c	1'b0	R/W	I2C or I3C mode select signal
0	legacy_i2c_xfer	1'b0	R	Legacy I2C transfer enable

Register: I3CS_CTRL2

Address: 0x26000428

Bits	Bit Name	Default	Type	Comment
31:0	i3c_slave_slv_pid [31:0]	32'h0	R/W	I3C slave PID[31:0]

Register: I3CS_CTRL3

Address: 0x2600042C

Bits	Bit Name	Default	Type	Comment
31:23	Reserved	9'b0	R/W	Reserved
22:20	i3cs_slv_clk_data_turn_time	3'b0	R/W	Slave maximum clock data turnaround time
19:16	Reserved	4'b0	R/W	Reserved
15:0	i3c_slave_slv_pid [47:32]	16'h0	R/W	I3C slave PID[47:32]

Register: I3CS_DEBUG_PORT0

Address: 0x2600068C

Bits	Bit Name	Default	Type	Comment
31:0	i3cs_debug_port [31:0]	32'h5140	R	I3C slave debug ports[31:0]

Register: I3CS_DEBUG_PORT1

Address: 0x26000690

Bits	Bit Name	Default	Type	Comment
31:23	Reserved	9'b0	R	Reserved, always read 'b0
22:0	i3cs_debug_port [54:32]	23'b0	R	I3C slave debug ports[54:32]

Register: D2D_PLL_CTRL1

Address: 0x26000700

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	2'b0	R/W	Reserved
29	frefcmlen	1'b0	R/W	Enable free CML input
28	foutvcoen	1'b1	R/W	VCO rate output clock enable, 8GHz drives BOW PHY
27:23	foutvcobyp	5'b0	R/W	Bypass undivided vco clock to specific output,
22	foutdiffen	1'b0	R/W	Enable [FOUTDIFFP/N] (frequency is FVCO/POSTDIV4)
21	foutcmlen	1'b0	R/W	Enable FOUTCML[P/N]
20	dmsen	1'b0	R/W	Enable delta-sigma modulator. 0 = DSM is powered down (integer mode). 1 = DSM is powered up (fractional mode)
19	dskewfastcal	1'b0	R/W	Deskew fast calibration enable.
18:7	dskewcalin	12'b0	R/W	Dskewcalbyp=0: initial value for deskew calibration. Dskewcalbyp=1: override value of deskew calibration
6	dskewcalen	1'b0	R/W	Deskew calibration enable
5:3	dskewcalcnt	3'b010	R/W	Counter for deskew calibration loop
2	dskewcalbyp	1'b0	R/W	Deskew calibration bypass
1	dac_en	1'b0	R/W	Enabling fractional noise canceling DAC
0	pll_en	1'b1	R/W	PLL enable

Register: D2D_PLL_CTRL2

Address: 0x26000704

Bits	Bit Name	Default	Type	Comment
31:28	posdiv2	4'b0	R/W	PLL post divider 2 (not used)
27:24	posdiv1	4'b0	R/W	PLL post divider 1 (not used)
23:20	posdiv0	4'b0	R/W	PLL post divider 0 (not used)
19:18	posdiv4	2'b0	R/W	PLL post divider 4 8GHz/4 (not used)
17:6	fbdiv	12'h50	R/W	Feedback clock divider 100MHz*80 = 8GHz VCO
5:0	refdiv	6'h1	R/W	Reference clock divider

Register: D2D_PLL_CTRL3

Address: 0x26000708

Bits	Bit Name	Default	Type	Comment
31:27	Reserved	5'b0	R/W	Reserved
26	LL_EXT_LPBK	0'b0	R/W	For loopback test
25	PHY_EXT_LPBK	0'b0	R/W	For loopback test
24	BC_PHY_SYSRS	0'b0	R/W	For loopback test
23:0	frac	24'h0	R/W	Fractional value of feed- back divider

Register: D2D_PLL_CTRL4

Address: 0x2600070C

Bits	Bit Name	Default	Type	Comment
31:0	Reserved	32'b0	R/W	Reserved

Register: D2D_PLL_STATUS

Address: 0x26000710

Bits	Bit Name	Default	Type	Comment
31:14	Reserved	18'b0	R	Reserved, always read 'b0
13:2	d2d_dskewcallout	12'h0	R	D2D dskewcallout
1	d2d_deskew_pll_locked	1'b0	R	D2D Deskew calibration settled
0	d2d_pll_locked	1'b0	R	D2D PLL locked

Register: TS_CTRL

Address: 0x26000760

Bits	Bit Name	Default	Type	Comment
31:17	Reserved	15'b0	R/W	Reserved
16	an_en	1'b0	R/W	Enables Analog Access
15	cload	1'b0	R/W	0 = normal operation 1 = load new values from cfg [7:0]
14	pd	1'b1	R/W	Active high power-down for the analog core
13:10	an_sel [3:0]	4'b0	R/W	Analog Select Control Set as 4'b0000 for normal operation
9:2	cfg [7:0]	8'h0	R/W	Configuration Selection cfg[7:5] = 3'b000 on dout [11:0] = 3'b001 ±0.4 C on dout [11:2] = 3'b010 ±1.5 C on dout [11:4] = Others Not allowed cfg[4] = 1'b0 Parallel Output = 1'b1 Serial Output cfg[3:0] = 4'b0000 Mode 1 Operation = 4'b0001 Mode 2 Operation
1	run	1'b0	R/W	Active high conversion enable
0	rstn	1'b0	R/W	Asynchronous active low reset

Register: TS_OUT

Address: 0x26000764

Bits	Bit Name	Default	Type	Comment
31:15	Reserved	17'b0	R	Reserved, always read 'b0
14	faultn	1'b0	R	Fault flag Indicates an internal fault when 1'b0. Normally a '1'
13	dout_type	1'b0	R	Indicates that the data output is temperature data when 1'b0. Is set to 1'b1 in analog access, signature select and fault debug
12	rdy	1'b0	R	Active high end of conversion pulse
11:0	dout [11:0]	12'h0	R	Conversion data

Register: VS_CTRL

Address: 0x26000780

Bits	Bit Name	Default	Type	Comment
31:24	Reserved	8'b0	R/W	Reserved
23	cload	1'b0	R/W	Configuration Load Set to 1'b1 to load new values from cfgn inputs
22	pd	1'b1	R/W	Active high power-down for the analog core

Bits	Bit Name	Default	Type	Comment
21:18	an_sel [3:0]	4'b0	R/W	Analog Select Control Set as 4'b0000 for normal operation
17:10	cfg2 [7:0]	8'b0	R/W	Sets the input source cfg2[7]: ext_ref cfg2 cfg2 [6:5]: Reserved cfg2 cfg2 [4:0]: input_sel
9:2	cfg1 [7:0]	8'b0	R/W	Sets output modes of voltage monitor cfg1[7]: Reserved cfg1[6:5]: resolution cfg1[4]: ser_mode cfg1[3:2]: pri_mode cfg1[1:0]: sec_mode
1	run	1'b0	R/W	Active high conversion enable
0	rstn	8'b0	R/W	Asynchronous active low reset

Register: VS_OUT

Address: 0x26000784

Bits	Bit Name	Default	Type	Comment
31:17	Reserved	15'b0	R	Reserved, always read 'b0
16	dout_type	1'b0	R	1'b0 Indicates that the data is valid voltage monitor data. 1'b1 indicates analog access, signature select and status output modes
15	faultn	1'b0	R	Fault flag Indicates an internal fault when 1'b0. Normally a '1'
14	rdy_latch	1'b0	R	Active high end of conversion latched
13:0	dout [13:0]	14'h0	R	Conversion data

Register: LVDS_CTRL

Address: 0x26000790

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R/W	Reserved
7:5	RTERM_VAL [2:0]	3'b100	R/W	Termination value set bits. This should always be set to the correct value for 100 Ohms. typical case termination value: 3'b000 –> 79 ohm 3'b001 –> 83 ohm 3'b010 –> 87 ohm 3'b011 –> 92 ohm 3'b100 –> 98 ohm 3'b101 –> 104 ohm 3'b110 –> 111 ohm

Bits	Bit Name	Default	Type	Comment
				3'b111 –> 120 ohm
4	VBIAS_SEL	1'b0	R/W	Select bias voltage for I/O 1'b1 –> From external bandgap reference gen on vbias in 1'b0 –> From internal resistor divide
3	BIAS_EN	1'b1	R/W	Bias circuit enable 0 –> Bias circuit disabled 1 –> Bias circuit enabled
2	RXCM_EN	1'b0	R/W	Enable receiver common mode generation internally. 1'b0 –> Internal common mode generation disabled 1'b1 –> Internal common mode generation enabled
1	RTERM_EN	1'b0	R/W	Termination Resistor enable/disable 0 –> Resistor Termination disabled 1 –> Resistor Termination enabled
0	RXEN	1'b1	R/W	LVDS receiver enable 0 –> Receiver disabled 1 –> Receiver enabled

Register: DRO_0_1

Address: 0x26000800

Bits	Bit Name	Default	Type	Comment
31:24	dro1_mea [7:0]	8'h0	R	DRO_1 measurement Result
23:18	Reserved	6'b0	R	Reserved, always read 'b0
17	dro1_ready	1'b0	R	DRO_1 ready
16	dro1_start	1'b0	R/W	DRO_1 start
15:8	dro0_mea [7:0]	8'h0	R	DRO_0 measurement Result
7:2	Reserved	6'b0	R	Reserved, always read 'b0
1	dro0_ready	1'b0	R	DRO_0 ready
0	dro0_start	1'b0	R/W	DRO_0 start

Register: DRO_2_3

Address: 0x26000804

Bits	Bit Name	Default	Type	Comment
31:24	dro3_mea [7:0]	8'h0	R	DRO_3 measurement Result
23:18	Reserved	6'b0	R	Reserved, always read 'b0
17	dro3_ready	1'b0	R	DRO_3 ready
16	dro3_start	1'b0	R/W	DRO_3 start
15:8	dro2_mea [7:0]	8'h0	R	DRO_2 measurement Result
7:2	Reserved	6'b0	R	Reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
1	dro2_ready	1'b0	R	DRO_2 ready
0	dro2_start	1'b0	R/W	DRO_2 start

Register: DRO_4_5

Address: 0x26000808

Bits	Bit Name	Default	Type	Comment
31:24	dro5_mea [7:0]	8'h0	R	DRO_5 measurement Result
23:18	Reserved	6'b0	R	Reserved, always read 'b0
17	dro5_ready	1'b0	R	DRO_5 ready
16	dro5_start	1'b0	R/W	DRO_5 start
15:8	dro4_mea [7:0]	8'h0	R	DRO_4 measurement Result
7:2	Reserved	6'b0	R	Reserved, always read 'b0
1	dro4_ready	1'b0	R	DRO_4 ready
0	dro4_start	1'b0	R/W	DRO_4 start

Register: UC_SYSCTRL_PID0

Address: 0x26000FE0

Bits	Bit Name	Default	Type	Comment
31	Reserved	1'b0	R	Reserved, always read 'b0
30:27	REVISION [3:0]	4'h1	R	Revision
26:16	JEPID [10:0]	11'h489	R	JTAG ID
15:0	PID [15:0]	16'h2	R	Part ID: 02-Bouffalo Lab IP

Register: UC_SYSCTRL_PID1

Address: 0x26000FE4

Bits	Bit Name	Default	Type	Comment
31:4	Reserved	28'b0	R	Reserved, always read 'b0
3:0	ECOREVNUM	4'h0	R	ECO Revision

Register: UC_SYSCTRL_CID0

Address: 0x26000FF0

Bits	Bit Name	Default	Type	Comment
31:0	CID0	32'h494E4348	R	Chip ID0 ("INCH")

Register: UC_SYSCTRL_CID1

Address: 0x26000FF4

Bits	Bit Name	Default	Type	Comment
31:0	CID1	32'h4F4C4446	R	Chip ID1 ("OLDF")

Register: UC_SYSCTRL_CID2

Address: 0x26000FF8

Bits	Bit Name	Default	Type	Comment
31:0	CID2	32'h20202047	R	Chip ID2 (" G")

Register: UC_SYSCTRL_CID3

Address: 0x26000FFC

Bits	Bit Name	Default	Type	Comment
31:0	CID3	32'h464C4354	R	Chip ID3 ("FLCT")

6.3 Serial Flash Control Register

Serial flash CSR address = 0x2600_1000 ~ 0x2600_1FFF

Register: SF_CTRL_0

Address: 0x26001000

Bits	Bit Name	Default	Type	Comment
31:24	sf_id	8'h1A	R/W	ID
23	sf_aes_iv_endian	1'b1	R/W	0: Little-endian 1: Big-endian
22	sf_aes_key_endian	1'b1	R/W	0: Little-endian 1: Big-endian
21	sf_aes_din_endian	1'b1	R/W	0: Little-endian 1: Big-endian
20	sf_aes_dout_endian	1'b1	R/W	0: Little-endian 1: Big-endian
19	sf_if_32b_adr_en	1'b0	R/W	1: Enable 32-bit address
18	sf_if_int_set	1'b0	R/W	1: Set interrupt
17	sf_if_int_clr	1'b1	R/W	1: Clear interrupt
16	sf_if_int	1'b0	R	Interrupt value
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11	sf_if_read_dly_en	1'b0	R/W	1: Enable flash controller read delay mode
10:8	sf_if_read_dly_n	3'b0	R/W	Flash controller read delay cycle = n + 1
7:5	Reserved	3'b0	R	Reserved, always read 'b0
4	sf_clk_out_inv_sel	1'b1	R/W	1: Select inverted clock out
3	sf_clk_out_gate_en	1'b1	R/W	1: Hardware auto gated clock out
2	sf_clk_sf_rx_inv_sel	1'b1	R/W	1: Select inverted flash Rx clock
1:0	Reserved	2'b0	R	Reserved, always read 'b0

Register: SF_CTRL_1

Address: 0x26001004

Bits	Bit Name	Default	Type	Comment
31	sf_ahb2sram_en	1'b1	R/W	1: Enable SRAM
30	sf_ahb2sif_en	1'b1	R/W	1: Enable IAHB to flash interface
29	sf_if_en	1'b1	R/W	0: Disable sf_if 1: enable sf_if
28	sf_if_fn_sel	1'b1	R/W	0: System AHB 1: icache AHB
27	sf_ahb2sif_stop	1'b0	R/W	1: Stop CPU access flash
26	sf_ahb2sif_stopped	1'b0	R	1: CPU stopped
25	sf_if_reg_wp	1'b1	R/W	Write protect

Bits	Bit Name	Default	Type	Comment
24	sf_if_reg_hold	1'b1	R/W	Hold
23	sf_ahb2sif_diswrap	1'b0	R/W	1: Disable IAHB to flash wrap access for XTS mode
22:20	sf_if_0_ack_lat	3'b110	R/W	ACK latency cycles
19	Reserved	1'b0	R	Reserved, always read 'b0
18	sf_if_sr_int_set	1'b0	R/W	1: Set interrupt
17	sf_if_sr_int_en	1'b0	R/W	1: Status read interrupt enable
16	sf_if_sr_int	1'b0	R	Interrupt value
15:8	sf_if_sr_pat	8'b0	R/W	Interrupt pattern
7:0	sf_if_sr_pat_mask	1'b0	R/W	Interrupt mask

Register: SF_IF_SAHB_0

Address: 0x26001008

Bits	Bit Name	Default	Type	Comment
31	sf_if_0_qpi_mode_en	1'b0	R/W	0: Normal SPI 1: QPI mode enable
30:28	sf_if_0_spi_mode	3'b000	R/W	000: Normal SPI 001: Dual output 010: Quad output 011: Dual IO 100: Quad IO
27	sf_if_0_cmd_en	1'b1	R/W	1: Command enable
26	sf_if_0_adr_en	1'b1	R/W	1: Address enable
25	sf_if_0_dmy_en	1'b0	R/W	1: Dummy enable
24	sf_if_0_dat_en	1'b1	R/W	1: Data enable
23	sf_if_0_dat_rw	1'b0	R/W	0: Read 1: write
22:20	sf_if_0_cmd_byte	3'b000	R/W	Number of command bytes minus 1
19:17	sf_if_0_adr_byte	3'b010	R/W	Number of address bytes minus 1
16:12	sf_if_0_dmy_byte	5'b0	R/W	Number of dummy bytes minus 1
11:2	sf_if_0_dat_byte	10'hFF	R/W	Number of command byte minus 1
1	sf_if_0_trig	1'b0	R/W	1: Trigger sf_if FSM
0	sf_if_busy	1'b0	R	1: Busy

Register: SF_IF_SAHB_1

Address: 0x2600100C

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_0_cmd_buf_0	32'h03000000	R/W	Command buffer 0

Register: SF_IF_SAHB_2

Address: 0x26001010

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_0_cmd_buf_1	32'h0	R/W	Command buffer 1

Register: SF_IF_IAHB_0

Address: 0x26001014

Bits	Bit Name	Default	Type	Comment
31	sf_if_1_qpi_mode_en	1'b0	R/W	0: Normal SPI 1: QPI mode enable
30:28	sf_if_1_spi_mode	3'b000	R/W	000: Normal SPI 001: Dual output 010: Quad output 011: Dual IO 100: Quad IO
27	sf_if_1_cmd_en	1'b1	R/W	1: Command enable
26	sf_if_1_adr_en	1'b1	R/W	1: Address enable
25	sf_if_1_dmy_en	1'b0	R/W	1: Dummy enable
24	sf_if_1_dat_en	1'b1	R/W	1: Data enable
23	sf_if_1_dat_rw	1'b0	R/W	0: Read 1: write
22:20	sf_if_1_cmd_byte	3'b000	R/W	Number of command bytes minus 1
19:17	sf_if_1_adr_byte	3'b010	R/W	Number of address bytes minus 1
16:12	sf_if_1_dmy_byte	5'b0	R/W	Number of dummy bytes minus 1
11:0	Reserved	12'b0	R	Reserved, always read 'b0

Register: SF_IF_IAHB_1

Address: 0x26001018

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_1_cmd_buf_0	32'h03000000	R/W	Command buffer 0

Register: SF_IF_IAHB_2

Address: 0x2600101C

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_1_cmd_buf_1	32'h0	R/W	Command buffer 1

Register: SF_IF_STATUS_0

Address: 0x26001020

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_status_0	32'h0	R	SF interface Status 0

Register: SF_IF_STATUS_1

Address: 0x26001024

Bits	Bit Name	Default	Type	Comment
31:0	sf_if_status_1	32'h20000000	R	SF interface Status 1

Register: SF_AHB2SIF_STATUS

Address: 0x2600102C

Bits	Bit Name	Default	Type	Comment
31:0	sf_ahb2sif_status	32'h1010003	R	AHB2SIF Status

Register: SF_IF_IO_DLY_0

Address: 0x26001030

Bits	Bit Name	Default	Type	Comment
31:30	sf_dqs_do_dly_sel	2'b0	R/W	IO delay selection
29:28	sf_dqs_di_dly_sel	2'b0	R/W	IO delay selection
27:26	sf_dqs_oe_dly_sel	2'b0	R/W	IO delay selection
25:10	Reserved	16'b0	R	Reserved, always read 'b0
9:8	sf_clk_out_dly_sel	2'b0	R/W	IO delay selection
7:4	Reserved	4'b0	R	Reserved, always read 'b0
3:2	sf_cs2_dly_sel	2'b0	R/W	IO delay selection
1:0	sf_cs_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_1

Address: 0x26001034

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	14'b0	R	Reserved, always read 'b0
17:16	sf_io_0_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	6'b0	R	Reserved, always read 'b0
9:8	sf_io_0_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	6'b0	R	Reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
1:0	sf_io_0_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_2

Address: 0x26001038

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	14'b0	R	Reserved, always read 'b0
17:16	sf_io_1_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	6'b0	R	Reserved, always read 'b0
9:8	sf_io_1_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	6'b0	R	Reserved, always read 'b0
1:0	sf_io_1_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_3

Address: 0x2600103C

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	14'b0	R	Reserved, always read 'b0
17:16	sf_io_2_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	6'b0	R	Reserved, always read 'b0
9:8	sf_io_2_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	6'b0	R	Reserved, always read 'b0
1:0	sf_io_2_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_IF_IO_DLY_4

Address: 0x26001040

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	14'b0	R	Reserved, always read 'b0
17:16	sf_io_3_do_dly_sel	2'b0	R/W	IO delay selection
15:10	Reserved	6'b0	R	Reserved, always read 'b0
9:8	sf_io_3_di_dly_sel	2'b0	R/W	IO delay selection
7:2	Reserved	6'b0	R	Reserved, always read 'b0
1:0	sf_io_3_oe_dly_sel	2'b0	R/W	IO delay selection

Register: SF_RESERVED

Address: 0x26001044

Bits	Bit Name	Default	Type	Comment
31:0	sf_reserved	32'hFFFF	R/W	Reserved

Register: SF_CTRL_2

Address: 0x26001070

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7	sf_id_offset_lock	1'b0	R/W	SF ID offset lock
6:0	Reserved	7'b0	R	Reserved, always read 'b0

Register: SF_CTRL_3

Address: 0x26001074

Bits	Bit Name	Default	Type	Comment
31:29	sf_if_1_ack_lat	3'b001	R/W	ACK latency cycles
28:21	Reserved	8'b0	R	Reserved, always read 'b0
20	sf_cmds_core_en	1'b1	R/W	1: Enable command splitter core
19:18	sf_cmds_1_wrap_mode [1:0]	2'b0	R/W	[0] 0: cmds bypass wrap commands to macro 1: cmds handle wrap commands. [1] 0: original mode 1: cmds force wrap16x4 is split into two wrap 8x4
17	sf_cmds_1_en	1'b0	R/W	1: Enable command splitter
16:13	sf_cmds_1_wrap_len	4'd6	R/W	Wrap length 0:8, 1:16, 2:32, 3:64, 4:128, 5:256, 6:512, 7:1024, 8:2048, 9:4096
12:0	Reserved	13'b0	R	Reserved, always read 'b0

Register: SF_ID0_OFFSET

Address: 0x260010A0

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:0	sf_id0_offset	28'b0	R/W	SF ID offset, will be locked if sf_id_offset_lock(0x26001070[7]) = 1'b1

Register: SF_DBG

Address: 0x260010B0

Note: This register is reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
31:6	Reserved	26'b0	R	Reserved, always read 'b0
5	sf_autoload_st_done	1'b0	R	sf_autoload_st_done, hardwired 1'b0
4:0	sf_autoload_st	5'd1	R	sf_autoload_st, hardwired 5'd1

Register: SF_CTRL_PROT_EN_RD

Address: 0x26001100

Note: This register is reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
31	sf_dbg_dis	1'b0	R	sf_dbg_dis, hardwired 1'b0
30	sf_if_0_trig_wr_lock	1'b0	R	sf_if_0_trig_wr_lock, hardwired 1'b0
29	Reserved	1'b0	R	Reserved, always read 'b0
28	sf_sec_tzsid_lock	1'b0	R	sf_sec_tzsid_lock, hardwired 1'b0
27:0	Reserved	28'b0	R	Reserved, always read 'b0

Register: SF_SRAM

Address: 0x26001600~0x2600167F

Bits	Bit Name	Default	Type	Comment
31:0	SF SRAM Data	32'bX	R/W	SF SRAM (128x32 bits) Address[6:0] = SF SRAM address (00~7F)

6.4 L1C Control Register

L1C CSR address = 0x2600_2000 ~ 0x2600_2FFF

Register: L1C_CONFIG

Address: 0x26002000

Bits	Bit Name	Default	Type	Comment
31:14	Reserved	18'b0	R/W	Reserved
13:12	sf_clk_sel [1:0]	2'b00	R/W	00 = 25MHz 01 = 50MHz 1X = 100MHz
11:8	l1c_way_dis	4'b0	R/W	Disable part of cache ways & used as AHB SRAM
7:2	Reserved	6'b0	R/W	Reserved
1	l1c_cnt_en	1'b0	R/W	Cache performance counter enable
0	l1c_cacheable	1'b0	R/W	Cacheable regions enable

Register: L1C_HIT_CNT_LSB

Address: 0x26002004

Bits	Bit Name	Default	Type	Comment
31:0	l1c_hit_cnt_lsb	32'h0	R	Low 32-bit hit counter

Register: L1C_HIT_CNT_MSB

Address: 0x26002008

Bits	Bit Name	Default	Type	Comment
31:0	l1c_hit_cnt_msb	32'h0	R	High 32-bit hit counter

Register: L1C_MISS_CNT

Address: 0x2600200C

Bits	Bit Name	Default	Type	Comment
31:0	l1c_miss_cnt	32'h0	R	Miss counter

Register: L1C_AUX_CONFIG

Address: 0x26002010

Bits	Bit Name	Default	Type	Comment
31	l1c_invalid_done	1'b0	R	l1c invalid done
30:4	Reserved	27'b0	R/W	Reserved

Bits	Bit Name	Default	Type	Comment
3	early_resp_dis	1'b0	R/W	early_resp_dis
2	xip_2t_access	1'b0	R/W	Set 1 for ROM 2T access if CPU freq >72MHz
1	l1c_invalid	1'b0	R/W	l1c invalid
0	l1c_en	1'b0	R/W	1: burst=3'b100 0: burst=3'b001

6.5 DMA Control Register

DMA CSR address = 0x2600_4000 ~ 0x2600_4FFF

Register: DMA_INTSTATUS

Address: 0x26004000

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	INTSTA	8'b0	R	Status of DMA interrupts after masking

Register: DMA_INTTCSTATUS

Address: 0x26004004

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	INTTCSTA	8'b0	R	Interrupt terminal count request status

Register: DMA_INTTCCLEAR

Address: 0x26004008

Note: This register always read 32'b0

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	TCRC	8'b0	W	Terminal count request clear

Register: DMA_INTERRORSTATUS

Address: 0x2600400C

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	IES	8'b0	R	Interrupt error status

Register: DMA_INTERRCLR

Address: 0x26004010

Note: This register always read 32'b0

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
7:0	IEC	8'b0	W	Interrupt error clear

Register: DMA_RAWINTTCSTATUS

Address: 0x26004014

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	SOTCIPTM	8'b0	R	Status of terminal counter interrupt prior to masking

Register: DMA_RAWINTERRORSTATUS

Address: 0x26004018

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	SOTEIPTM	8'b0	R	Status of error interrupts prior to masking

Register: DMA_ENBLDCHNS

Address: 0x2600401C

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	CES	8'b0	R	Channel enable status

Register: DMA_SOFTBREQ

Address: 0x26004020

Bits	Bit Name	Default	Type	Comment
31:0	SBR	32'h0	R/W	Software burst request

Register: DMA_SOFTSREQ

Address: 0x26004024

Bits	Bit Name	Default	Type	Comment
31:0	SSR	32'h0	R/W	Software single request

Register: DMA_SOFTLBREQ

Address: 0x26004028

Bits	Bit Name	Default	Type	Comment
31:0	SLBR	32'h0	R/W	Software last burst request

Register: DMA_SOFTLSREQ

Address: 0x2600402C

Bits	Bit Name	Default	Type	Comment
31:0	SLSR	32'h0	R/W	Software last single request

Register: DMA_CONFIG

Address: 0x26004030

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1	AHBMEC	1'b0	R/W	AHB master endianness configuration: 0=little-endian, 1=big-endian.
0	SMDMAEN	1'b0	R/W	SMDMA enable

Register: DMA_SYNC

Address: 0x26004034

Bits	Bit Name	Default	Type	Comment
31:0	DSLFDRS	32'h0	R/W	DMA synchronization logic for DMA request signals. 0=enable, 1=disable.

Register: DMA_C[3:0]SRCADDR

Address: 0x26004[4:1]00

Bits	Bit Name	Default	Type	Comment
31:0	DMASA	32'h0	R/W	DMA source address

Register: DMA_C[3:0]DSTADDR

Address: 0x26004[4:1]04

Bits	Bit Name	Default	Type	Comment
31:0	DMADA	32'h0	R/W	DMA destination address

Register: DMA_C[3:0]LLI

Address: 0x26004[4:1]08

Bits	Bit Name	Default	Type	Comment
31:0	FLLI	32'h0	R/W	First linked list item. Bits [1:0] must be 0

Register: DMA_C[3:0]CONTROL

Address: 0x26004[4:1]0C

Bits	Bit Name	Default	Type	Comment
31	TCIEN	1'b0	R/W	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	PROTECT	3'b000	R/W	Protection
27	DI	1'b1	R/W	Destination increment. When set the destination address is incremented after each transfer
26	SI	1'b1	R/W	Source Increment. When set the source address is incremented after each transfer
25	Reserved	N/A	N/A	Reserved
24:23	FIXCNT	2'b00	R/W	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_- cnt«DWidth))«DWidth
22:21	DTW	2'b10	R/W	Destination transfer width: 8/16/32
20	Reserved	N/A	N/A	Reserved
19:18	STW	2'b10	R/W	Source transfer width: 8/16/32
17	ADDMODE	1'b0	R/W	Add mode: issue remain destination traffic
16:15	DBS	2'b01	R/W	Destination burst size: 1/4/8/16
14	MINMODE	1'b0	R/W	Minus mode. Not issue all destination traffic.
13:12	SBS	2'b01	R/W	Source burst size: 1/4/8/16. Note CH FIFO Size is 16Bytes and SBSize*Swidth should <= 16B
11:0	TS	12'h0	R/W	Transfer size: 0:4095: Number of data transfer left to complete when SMDMA is the flow controller.

Register: DMA_C[3:0]CONFIG

Address: 0x26004[4:1]10

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	N/A	N/A	Reserved
29:20	LLICOUNT	10'b0	R	LLI counter: Increased 1 each LLI run. Cleared 0 when config control.
19	Reserved	N/A	N/A	Reserved

Bits	Bit Name	Default	Type	Comment
18	HALT	1'b0	R/W	Halt. 0=enable DMA requests, 1=ignore subsequent source DMA requests.
17	ACTIVE	1'b0	R	Active. 0=no data in FIFO of the channel, 1=FIFO of the channel has data.
16	LOCK	1'b0	R/W	Lock
15	TCIM	1'b0	R/W	Terminal count interrupt mask
14	IEM	1'b0	R/W	Interrupt error mask
13:11	FLOWCTRL	3'b000	R/W	000: Memory to memory DMA 001: Memory to peripheral DMA 010: Peripheral to memory DMA 011: Source peripheral to destination peripheral DMA.
10:6	DSTPH	5'h0	R/W	Destination peripherals: 0: UART TX 1: UART RX 2: SPI0 TX 3: SPI0 RX 4: SPI1 TX 5: SPI1 RX 6: I2C TX 7: I2C RX Others: Reserved
5:1	SRCPH	5'h0	R/W	Source peripherals: 0: UART TX 1: UART RX 2: SPI0 TX 3: SPI0 RX 4: SPI1 TX 5: SPI1 RX 6: I2C TX 7: I2C RX Others: Reserved
0	CHEN	1'b0	R/W	Channel enable.

6.6 Security Engine Control Register

Security Engine CSR address = 0x2600_5000 ~ 0x2600_5FFF

Register: SE_SHA_0_CTRL

Address: 0x26005000

Bits	Bit Name	Default	Type	Comment
31:16	se_sha_0_msg_len	16'h0	R/W	Number of 512-bit block
15:12	Reserved	NA	NA	Reserved
11	se_sha_0_int_mask	1'b0	R/W	se_sha_0 interrupt mask
10	se_sha_0_int_set_1t	1'b0	W1p	1: Set Interrupt
9	se_sha_0_int_clr_1t	1'b0	W1p	1: Clear Interrupt
8	se_sha_0_int	1'b0	R	Interrupt value
7	Reserved	NA	NA	Reserved
6	se_sha_0_hash_sel	1'b0	R/W	0: New Hash 1: Accumulate Last Hash
5	se_sha_0_en	1'b0	R/W	1: Enable sha Engine
4:2	se_sha_0_mode	3'b000	R/W	000: SHA-256 001: SHA-224 010: SHA-1 011: SHA-1 100: SHA-512 101: SHA-384 110: SHA-512/224 111: SHA-512/256
1	se_sha_0_trig_1t	1'b0	W1p	1: Trigger sha Engine
0	se_sha_0_busy	1'b0	R	1: sha Engine Busy

Register: SE_SHA_0_MSA

Address: 0x26005004

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_msa	32'h0	R/W	Message Source Address

Register: SE_SHA_0_STATUS

Address: 0x26005008

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_status	32'h41	R	

Register: SE_SHA_0_ENDIAN

Address: 0x2600500C

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	N/A	N/A	Reserved
0	se_sha_0_dout_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian

Register: SE_SHA_0_HASH_L_0

Address: 0x26005010

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_0	32'h0	R	Big-Endian Hash 0 (MSB)

Register: SE_SHA_0_HASH_L_1

Address: 0x26005014

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_1	32'h0	R	Big-Endian Hash 1

Register: SE_SHA_0_HASH_L_2

Address: 0x26005018

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_2	32'h0	R	Big-Endian Hash 2

Register: SE_SHA_0_HASH_L_3

Address: 0x2600501C

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_3	32'h0	R	Big-Endian Hash 3

Register: SE_SHA_0_HASH_L_4

Address: 0x26005020

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_4	32'h0	R	Big-Endian Hash 4

Register: SE_SHA_0_HASH_L_5

Address: 0x26005024

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_5	32'h0	R	Big-Endian Hash 5

Register: SE_SHA_0_HASH_L_6

Address: 0x26005028

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_6	32'h0	R	Big-Endian Hash 6

Register: SE_SHA_0_HASH_L_7

Address: 0x2600502C

Bits	Bit Name	Default	Type	Comment
31:0	se_sha_0_hash_l_7	32'h0	R	Big-Endian Hash 7 (LSB)

Register: SE_SHA_0_CTRL_PROT

Address: 0x260050FC

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	NA	NA	Reserved
2	se_sha_id1_en	1'b1	R/W	id1 Access Right
1	se_sha_id0_en	1'b1	R/W	id0 Access Right
0	Reserved	NA	NA	Reserved

Register: SE_AES_0_CTRL

Address: 0x26005100

Bits	Bit Name	Default	Type	Comment
31:16	se_aes_0_msg_len	16'h0	R/W	Number of 128-bit Block
15	Reserved	NA	NA	Reserved
14	se_aes_0_iv_sel	1'b0	R/W	0: New iv 1: Same iv as Last One
13:12	se_aes_0_block_mode	2'b00	R/W	00: ECB mode 01: CTR mode 10: CBC mode 11: XTS mode
11	se_aes_0_int_mask	1'b0	R/W	se_aes_0 interrupt mask
10	se_aes_0_int_set_1t	1'b0	W1p	1: Set Interrupt
9	se_aes_0_int_clr_1t	1'b0	W1p	1: Clear Interrupt

Bits	Bit Name	Default	Type	Comment
8	se_aes_0_int	1'b0	R	Interrupt Value
7	se_aes_0_hw_key_en	1'b0	R/W	0: sw Key 1: hw Key
6	se_aes_0_dec_key_sel	1'b0	R/W	0: New Key 1: Same Key as Last One
5	se_aes_0_dec_en	1'b0	R/W	0: Encode 1: Decode
4:3	se_aes_0_mode	2'b00	R/W	00: 128-bit Mode 01: 256-bit Mode 10: 192-bit Mode 11: 128-bit Double Key Mode
2	se_aes_0_en	1'b0	R/W	1: Enable
1	se_aes_0_trig_1t	1'b0	W1p	1: Trigger aes Engine
0	se_aes_0_busy	1'b0	R	1: aes Engine Busy

Register: SE_AES_0_MSA

Address: 0x26005104

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_msa	32'h0	R/W	Message Source Address

Register: SE_AES_0_MDA

Address: 0x26005108

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_mda	32'h0	R/W	Message Destination Address

Register: SE_AES_0_STATUS

Address: 0x2600510C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_status	32'h4100	R	

Register: SE_AES_0_IV_0

Address: 0x26005110

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_0	32'h0	R/W	Big Endian Initial Vector (MSB)

Register: SE_AES_0_IV_1

Address: 0x26005114

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_1	32'h0	R/W	Big Endian Initial Vector

Register: SE_AES_0_IV_2

Address: 0x26005118

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_2	32'h0	R/W	Big Endian Initial Vector

Register: SE_AES_0_IV_3

Address: 0x2600511C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_iv_3	32'h0	R/W	Big Endian Initial Vector (LSB) (CTR Mode: 32-bit Counter Initial Value)

Register: SE_AES_0_KEY_0

Address: 0x26005120

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_0	32'h0	R/W	Big Endian aes Key (aes-128/256 Key MSB)

Register: SE_AES_0_KEY_1

Address: 0x26005124

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_1	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_2

Address: 0x26005128

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_2	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_3

Address: 0x2600512C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_3	32'h0	R/W	Big Endian aes Key (aes-128 key LSB)

Register: SE_AES_0_KEY_4

Address: 0x26005130

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_4	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_5

Address: 0x26005134

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_5	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_6

Address: 0x26005138

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_6	32'h0	R/W	Big Endian aes Key

Register: SE_AES_0_KEY_7

Address: 0x2600513C

Bits	Bit Name	Default	Type	Comment
31:0	se_aes_0_key_7	32'h0	R/W	Big Endian aes Key (aes-256 key LSB)

Register: SE_AES_0_ENDIAN

Address: 0x26005148

Bits	Bit Name	Default	Type	Comment
31:30	se_aes_0_ctr_len	2'b00	R/W	00:4-byte Counter 01:1-byte Counter 10:2-byte Counter 11:3-byte Counter
29:5	Reserved	N/A	N/A	Reserved
4	se_aes_0_twk_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian, default 1 for XTS
3	se_aes_0_iv_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian
2	se_aes_0_key_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian
1	se_aes_0_din_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian

Bits	Bit Name	Default	Type	Comment
0	se_aes_0_dout_endian	1'b1	R/W	0: Little-Endian 1: Big-Endian

Register: SE_TRNG_0_CTRL_0

Address: 0x26005200

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	N/A	N/A	Reserved
11	se_trng_0_int_mask	1'b0	R/W	se_trng_0 interrupt mask
10	se_trng_0_int_set_1t	1'b0	W1p	1: Set Interrupt
9	se_trng_0_int_clr_1t	1'b0	W1p	1: Clear Interrupt
8	se_trng_0_int	1'b0	R	Interrupt Value
7:5	Reserved	N/A	N/A	Reserved
4	se_trng_0_ht_error	1'b0	R	1: Health Test Error
3	se_trng_0_dout_clr_1t	1'b0	W1p	1: Clear trng_dout to Zero
2	se_trng_0_en	1'b0	R/W	1: Enable
1	se_trng_0_trig_1t	1'b0	W1p	1: Trigger trng Engine
0	se_trng_0_busy	1'b0	R	1: trng Engine Busy

Register: SE_TRNG_0_STATUS

Address: 0x26005204

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_status	32'h100020	R	TRNG status

Register: SE_TRNG_0_DOUT_0

Address: 0x26005208

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_0	32'h0	R	random value

Register: SE_TRNG_0_DOUT_1

Address: 0x2600520C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_1	32'h0	R	random value

Register: SE_TRNG_0_DOUT_2

Address: 0x26005210

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_2	32'h0	R	random value

Register: SE_TRNG_0_DOUT_3

Address: 0x26005214

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_3	32'h0	R	random value

Register: SE_TRNG_0_DOUT_4

Address: 0x26005218

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_4	32'h0	R	random value

Register: SE_TRNG_0_DOUT_5

Address: 0x2600521C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_5	32'h0	R	random value

Register: SE_TRNG_0_DOUT_6

Address: 0x26005220

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_6	32'h0	R	random value

Register: SE_TRNG_0_DOUT_7

Address: 0x26005224

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_dout_7	32'h0	R	random value

Register: SE_TRNG_0_TEST

Address: 0x26005228

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	N/A	N/A	Reserved

Bits	Bit Name	Default	Type	Comment
1	se_trng_0_cp_test_en	1'b0	R/W	1: Enable trng Conditional Component Test Mode
0	se_trng_0_test_en	1'b0	R/W	1: Enable trng Test Mode

Register: SE_TRNG_0_CTRL_1

Address: 0x2600522C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_reseed_n_lsb	32'hFFFF	R/W	Reload Seed When Number of Used Random Value is Larger than reseed_n

Register: SE_TRNG_0_CTRL_2

Address: 0x26005230

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	N/A	N/A	Reserved
15:0	se_trng_0_reseed_n_msb	16'hFF	R/W	Reload Seed When Number of Used Random Value is Larger than reseed_n

Register: SE_TRNG_0_CTRL_3

Address: 0x26005234

Bits	Bit Name	Default	Type	Comment
31	se_trng_0_rosc_en	1'b1	R/W	trng rosc Enable
30:0	Reserved	N/A	N/A	Reserved

Register: SE_TRNG_0_TEST_OUT_0

Address: 0x26005240

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_0	32'h0	R	

Register: SE_TRNG_0_TEST_OUT_1

Address: 0x26005244

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_1	32'h0	R	

Register: SE_TRNG_0_TEST_OUT_2

Address: 0x26005248

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_2	32'h0	R	

Register: SE_TRNG_0_TEST_OUT_3

Address: 0x2600524C

Bits	Bit Name	Default	Type	Comment
31:0	se_trng_0_test_out_3	32'h0	R	

Register: SE_CTRL_RESERVED_0

Address: 0x26005F04

Bits	Bit Name	Default	Type	Comment
31:0	se_ctrl_reserved_0	32'h0	R/W	Reserved

Register: SE_CTRL_RESERVED_1

Address: 0x26005F08

Bits	Bit Name	Default	Type	Comment
31:0	se_ctrl_reserved_1	32'hFFFFFFFF	R/W	Reserved

Register: SE_CTRL_RESERVED_2

Address: 0x26005F0C

Bits	Bit Name	Default	Type	Comment
31:0	se_ctrl_reserved_2	32'h0	R	Reserved Read SE_CTRL_RESERVED_0

6.7 OTP Register

OTP CSR address = 0x2608_0000 ~ 0x2608_FFFF

Register: OFS_VERSION

Address: 0x26083400

Bits	Bit Name	Default	Type	Comment
31:0	WRAPPER_VER	32'h20230613	R	Wrapper version

Register: OFS_PART_NUM

Address: 0x26083404

Bits	Bit Name	Default	Type	Comment
31:0	NEO_PART_NUM	32'h4547512D	R	Neo part number

Register: OFS_INTRPT

Address: 0x26083408

Bits	Bit Name	Default	Type	Comment
31:18	Reserved	14'b0	R	Reserved, always read 'b0
17:16	intrpt_en[1:0]	2'b00	R/W	Interrupt enable
15:2	Reserved	14'b0	R	Reserved, always read 'b0
1:0	intrpt_st[1:0] (R) intrpt_clr[1:0] (W)	2'b00	R/W1c	intrpt_st[1]: OTP read interrupt intrpt_st[0]: read outside memory interrupt Write 1'b1 = intrpt_clr

Register: OFS_STATUS

Address: 0x26083500

Bits	Bit Name	Default	Type	Comment
31:14	Reserved	18'b0	R	Reserved, always read 'b0
13	watch_dog_active	1'b0	R	Watch dog active
12	bist_mode	1'b0	R	BIST mode, write 0x2608_3508[31:24] = 8'hA5
11:8	bist_fail_flag[3:0]	4'b0	R	cp_flow_cnt in BIST mode
7:1	Reserved	7'b0	R	Reserved, always read 'b0
0	ctl_busy	1'b0	R	Control busy flag

Register: OFS_DEEP

Address: 0x26083504

Bits	Bit Name	Default	Type	Comment
31	bist_blank_check	1'b0	R/W	BIST blank check
30	bist_wtd_enable	1'b1	R/W	BIST watch dog enable
29:1	Reserved	29'b0	R	Reserved, always read 'b0
0	ctl_pdstb	1'b1	R/W	Power down strobe

Register: OFS_CONFIG

Address: 0x26083508

Bits	Bit Name	Default	Type	Comment
31:24	set_bist_mode	8'h0	W	Write 8'hA5 to set bist_mode(0x3500[12]) Always read 8'b0
23:11	Reserved	13'b0	R	Reserved, always read 'b0
10:8	ctl_rdmd	3'b000	R/W	Read modes: 3'b001 = CTL_INIT_MG_RD 3'b010 = CTL_PGM_MG_RD 3'b011 = CTL_HT_INIT_MG_RD 3'b100 = CTL_HT_PGM_MG_RD 3'b101 = CTL_LT_PGM_MG_RD Others: = CTL_RD
7:3	Reserved	5'b0	R	Reserved, always read 'b0
2	ctl_pgm_prt	1'b0	R/W	State machine control ctl_pgm_prt
1	ctl_pgm_ign	1'b0	R/W	State machine control ctl_pgm_ign
0	Reserved	1'b0	R	Reserved, always read 'b0

6.8 SPI Control Register

SPI0 CSR address = 0x2000_0000 ~ 0x2000_0FFF
SPI1 CSR address = 0x2000_1000 ~ 0x2000_1FFF

Register: SPI_CONFIG
Address: 0x2000[1:0]000

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	16'b0	R	Reserved, always read 'b0
15:12	cr_spi_deg_cnt	4'b0	R/W	De-glitch function cycle count.
11	cr_spi_deg_en	1'b0	R/W	Enable signal of de-glitch function.
10	cr_spi_s_3pin_mode	1'b0	R/W	SPI slave 3-pin mode 0: 4-pin mode (SS_n is enabled) 1: 3-pin mode (SS_n is disabled / don't care)
9	cr_spi_m_cont_en	1'b0	R/W	Enable signal of master continuous transfer mode 0: Disabled. SS_n will de-assert between each data frame. 1: Enabled. SS_n will stay asserted between each consecutive data frame if next data is valid in FIFO.
8	cr_spi_rxd_ignr_en	1'b0	R/W	Enable signal of RX data ignore function.
7	cr_spi_byte_inv	1'b0	R/W	Byte inverse signal for each FIFO entry. 0: Byte [0] is sent first. 1: Byte [3] is sent first.
6	cr_spi_bit_inv	1'b0	R/W	Bit inverse signal for each FIFO entry. 0: Each byte is sent MSB first. 1: Each byte [3] is sent LSB first.
5	cr_spi_sclk_ph	1'b0	R/W	SCLK clock phase inverse signal
4	cr_spi_sclk_pol	1'b0	R/W	SCLK polarity 0: SCLK output LOW at IDLE state 1: SCLK output HIGH at IDLE state
3:2	cr_spi_frame_size	2'b00	R/W	SPI frame size (also the valid width for each FIFO entry) 00: 8-bit 01: 16-bit 10: 24-bit 11: 32-bit
1	cr_spi_s_en	1'b0	R/W	Enable signal of SPI slave function. Master and slave should not be both enabled at same time. This bit becomes don't-care if cr_spi_m_en is enabled
0	cr_spi_m_en	1'b0	R/W	Enable signal of SPI Maser function. Asserting this bit will trigger the transaction and should be de-asserted after finish.

Register: SPI_INT_STS

Address: 0x2000[1:0]004

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	2'b0	R	Reserved, always read 'b0
29	cr_spi_fer_en	1'b1	R/W	Interrupt enable of spi_fer_int
28	cr_spi_txu_en	1'b1	R/W	Interrupt enable of spi_txu_int
27	cr_spi_sto_en	1'b1	R/W	Interrupt enable of spi_sto_int
26	cr_spi_rxf_en	1'b1	R/W	Interrupt enable of spi_rxv_int
25	cr_spi_txf_en	1'b1	R/W	Interrupt enable of spi_txe_int
24	cr_spi_end_en	1'b1	R/W	Interrupt enable of spi_end_int
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	Reserved	1'b0	R/W	Reserved
20	cr_spi_txu_clr	1'b0	W1c	Interrupt clear of spi_txu_int
19	cr_spi_sto_clr	1'b0	W1c	Interrupt clear of spi_sto_int
18:17	Reserved	2'b0	R/W	Reserved
16	cr_spi_end_clr	1'b0	W1c	Interrupt clear of spi_end_int
15:14	Reserved	2'b0	R	Reserved, always read 'b0
13	cr_spi_fer_mask	1'b1	R/W	Interrupt mask of spi_fer_int
12	cr_spi_txu_mask	1'b1	R/W	Interrupt mask of spi_txu_int
11	cr_spi_sto_mask	1'b1	R/W	Interrupt mask of spi_sto_int
10	cr_spi_rxf_mask	1'b1	R/W	Interrupt mask of spi_rxv_int
9	cr_spi_txf_mask	1'b1	R/W	Interrupt mask of spi_txe_int
8	cr_spi_end_mask	1'b1	R/W	Interrupt mask of spi_end_int
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	spi_fer_int	1'b0	R	SPI TX/RX FIFO error interrupt. Auto cleared when FIFO overflow/underflow error flag is cleared.
4	spi_txu_int	1'b0	R	SPI slave mode TX underrun error flag. Triggered when TXD is not ready during transfer in slave mode.
3	spi_sto_int	1'b0	R	SPI slave mode transfer time-out interrupt. Triggered when SPI bus is idle for a given value
2	spi_rxf_int	1'b0	R	SPI RX FIFO ready (rx_fifo_cnt>rx_fifo_th) interrupt, auto cleared when data is popped.
1	spi_txf_int	1'b1	R	SPI TX FIFO ready (tx_fifo_cnt>tx_fifo_th) interrupt, auto cleared when data is pushed.
0	spi_end_int	1'b0	R	SPI transfer end interrupt. Shared by both master and slave mode.

Register: SPI_BUS_BUSY

Address: 0x2000[1:0]008

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	31'b0	R	Reserved, always read 'b0
0	sts_spi_bus_busy	1'b0	R	Indicator of SPI bus busy

Register: SPI_PRD_0

Address: 0x2000[1:0]010

Bits	Bit Name	Default	Type	Comment
31:24	cr_spi_prd_d_ph_1	8'h0F	R/W	Length of DATA phase 1 (unit: SPI source clock period)
23:16	cr_spi_prd_d_ph_0	8'h0F	R/W	Length of DATA phase 0 (unit: SPI source clock period))
15:8	cr_spi_prd_p	8'h0F	R/W	Length of STOP condition (unit: SPI source clock period)
7:0	cr_spi_prd_s	8'h0F	R/W	Length of START condition (unit: SPI source clock period)

Register: SPI_PRD_1

Address: 0x2000[1:0]014

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	cr_spi_prd_i	8'h0F	R/W	Length of INTERVAL between frame (unit: SPI source clock period)

Register: SPI_RXD_IGNR

Address: 0x2000[1:0]018

Bits	Bit Name	Default	Type	Comment
31:21	Reserved	11'b0	R	Reserved, always read 'b0
20:16	cr_spi_rxd_ignr_s	5'b0	R/W	Starting point of RX data ignore function (unit: bit)
15:5	Reserved	11'b0	R	Reserved, always read 'b0
4:0	cr_spi_rxd_ignr_p	5'b0	R/W	Stopping point of RX data ignore function (unit: bit)

Register: SPI_STO_VALUE

Address: 0x2000[1:0]01C

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	20'b0	R	Reserved, always read 'b0
11:0	cr_spi_sto_value	12'hFFF	R/W	Time-out value for spi_sto_int triggering

Register: SPI_FIFO_CONFIG_0

Address: 0x2000[1:0]080

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7	rx_fifo_underflow	1'b0	R	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	1'b0	R	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	1'b0	R	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	1'b0	R	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	1'b0	W1c	Clear signal of RX FIFO
2	tx_fifo_clr	1'b0	W1c	Clear signal of TX FIFO
1	spi_dma_rx_en	1'b0	R/W	Enable signal of dma_rx_req/ack interface
0	spi_dma_tx_en	1'b0	R/W	Enable signal of dma_tx_req/ack interface

Register: SPI_FIFO_CONFIG_1

Address: 0x2000[1:0]084

Bits	Bit Name	Default	Type	Comment
31:26	Reserved	6'b0	R	Reserved, always read 'b0
25:24	rx_fifo_th	2'b00	R/W	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23:18	Reserved	6'b0	R	Reserved, always read 'b0
17:16	tx_fifo_th	2'b00	R/W	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:11	Reserved	5'b0	R	Reserved, always read 'b0
10:8	rx_fifo_cnt	3'b000	R	RX FIFO available count, means byte count of data received in RX FIFO (unit: byte)
7:3	Reserved	5'b0	R	Reserved, always read 'b0
2:0	tx_fifo_cnt	3'b100	R	TX FIFO available count, means empty space remained in TX FIFO (unit: byte)

Register: SPI_FIFO_WDATA

Address: 0x2000[1:0]088

Bits	Bit Name	Default	Type	Comment
31:0	spi_fifo_wdata	32'h0	W	SPI FIFO write data port

Register: SPI_FIFO_RDATA

Address: 0x2000[1:0]08C

Bits	Bit Name	Default	Type	Comment
31:0	spi_fifo_rdata	32'h0	R	SPI FIFO read data port

Register: BACKUP_IO_EN

Address: 0x2000[1:0]0FC

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	31'b0	R	Reserved, always read 'b0
0	backup_io_en	1'b0	R/W	Enable IO backup function

6.9 I2C Control Register

I2C CSR address = 0x2000_2000 ~ 0x2000_2FFF

Register I2C_CONFIG

Address: 0x20002000

Bits	Bit Name	Default	Type	Comment
31:28	cr_i2c_deg_cnt	4'b0	R/W	De-glitch function cycle count.
27:20	cr_i2c_pkt_len	8'h0	R/W	Packet length (unit: byte)
19:18	Reserved	2'b0	R	Reserved, always read 'b0
17:8	cr_i2c_slv_addr	10'h0	R/W	Slave address for I2C transaction (target address)
7	cr_i2c_10b_addr_en	1'b0	R/W	10-bit address enable
6:5	cr_i2c_sub_addr_bc	2'b00	R/W	Slave address field byte count 00: 1 byte 01: 2 byte 10: 3 byte 11: 4 byte
4	cr_i2c_sub_addr_en	1'b0	R/W	Enable signal of I2C sub-address field
3	cr_i2c_scl_sync_en	1'b1	R/W	Enable signal of I2C SCL synchronization. Should be enabled to support Multi-Master and Clock-Stretching (normally should be turned off)
2	cr_i2c_deg_en	1'b0	R/W	Enable signal of I2C de-glitch (for all input pins)
1	cr_i2c_pkt_dir	1'b1	R/W	Transfer direction of the packet. 0: write 1: read.
0	cr_i2c_m_en	1'b0	R/W	Enable signal of I2C Maser function. Asserting this bit will trigger the transaction and should be de-asserted after finish.

Register: I2C_INT_STS

Address: 0x20002004

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	2'b0	R	Reserved, always read 'b0
29	cr_i2c_fer_en	1'b1	R/W	Interrupt enable of i2c_fer_int
28	cr_i2c_arb_en	1'b1	R/W	Interrupt enable of i2c_arb_int
27	cr_i2c_nak_en	1'b1	R/W	Interrupt enable of i2c_nak_int
26	cr_i2c_rxf_en	1'b1	R/W	Interrupt enable of i2c_rxf_int
25	cr_i2c_txf_en	1'b1	R/W	Interrupt enable of i2c_txf_int
24	cr_i2c_end_en	1'b1	R/W	Interrupt enable of i2c_end_int
23:22	Reserved	2'b0	R	Reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
21	Reserved	1'b0	R/W	Reserved
20	cr_i2c_arb_clr	1'b0	W1c	Interrupt clear of i2c_arb_int
19	cr_i2c_nak_clr	1'b0	W1c	Interrupt clear of i2c_nak_int
18:17	Reserved	2'b0	R/W	Reserved
16	cr_i2c_end_clr	1'b0	W1c	Interrupt clear of i2c_end_int
15:14	Reserved	2'b0	R	Reserved, always read 'b0
13	cr_i2c_fer_mask	1'b1	R/W	Interrupt mask of i2c_fer_int
12	cr_i2c_arb_mask	1'b1	R/W	Interrupt mask of i2c_arb_int
11	cr_i2c_nak_mask	1'b1	R/W	Interrupt mask of i2c_nak_int
10	cr_i2c_rxf_mask	1'b1	R/W	Interrupt mask of i2c_rxf_int
9	cr_i2c_txf_mask	1'b1	R/W	Interrupt mask of i2c_txf_int
8	cr_i2c_end_mask	1'b1	R/W	Interrupt mask of i2c_end_int
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	i2c_fer_int	1'b0	R	I2C TX/RX FIFO error interrupt. Auto-cleared when FIFO overflow/underflow error flag is cleared.
4	i2c_arb_int	1'b0	R	I2C arbitration lost interrupt
3	i2c_nak_int	1'b0	R	I2C NAK received interrupt
2	i2c_rxf_int	1'b0	R	I2C RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto cleared when data is popped
1	i2c_txf_int	1'b1	R	I2C TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto cleared when data is pushed
0	i2c_end_int	1'b0	R	I2C transfer end interrupt.

Register: I2C_SUB_ADDR

Address: 0x20002008

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_sub_addr_b3	8'h0	R/W	Sub address field byte 3
23:16	cr_i2c_sub_addr_b2	8'h0	R/W	Sub address field byte 2
15:8	cr_i2c_sub_addr_b1	8'h0	R/W	Sub address field byte 1
7:0	cr_i2c_sub_addr_b0	8'h0	R/W	Sub address field byte 0

Register: I2C_BUS_BUSY

Address: 0x2000200C

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1	cr_i2c_bus_busy_clr	1'b0	W1c	Clear signal of bus_busy status, not for normal usage (in case I2C bus hangs.)

Bits	Bit Name	Default	Type	Comment
0	sts_i2c_bus_busy	1'b0	R	I2C bus busy

Register: I2C_PRD_START

Address: 0x20002010

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_prd_s_ph_3	8'h0F	R/W	Length of START condition phase 3
23:16	cr_i2c_prd_s_ph_2	8'h0F	R/W	Length of START condition phase 2
15:8	cr_i2c_prd_s_ph_1	8'h0F	R/W	Length of START condition phase 1
7:0	cr_i2c_prd_s_ph_0	8'h0F	R/W	Length of START condition phase 0

Register: I2C_PRD_STOP

Address: 0x20002014

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_prd_p_ph_3	8'h0F	R/W	Length of STOP condition phase 3
23:16	cr_i2c_prd_p_ph_2	8'h0F	R/W	Length of STOP condition phase 2
15:8	cr_i2c_prd_p_ph_1	8'h0F	R/W	Length of STOP condition phase 1
7:0	cr_i2c_prd_p_ph_0	8'h0F	R/W	Length of STOP condition phase 0

Register: I2C_PRD_DATA

Address: 0x20002018

Bits	Bit Name	Default	Type	Comment
31:24	cr_i2c_prd_d_ph_3	8'h0F	R/W	Length of DATA phase 3
23:16	cr_i2c_prd_d_ph_2	8'h0F	R/W	Length of DATA phase 2
15:8	cr_i2c_prd_d_ph_1	8'h0F	R/W	Length of DATA phase 1
7:0	cr_i2c_prd_d_ph_0	8'h0F	R/W	Length of DATA phase 0

Register: I2C_FIFO_CONFIG_0

Address: 0x20002080

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7	rx_fifo_underflow	1'b0	R	Underflow flag of RX FIFO, cleared by rx_fifo_clr
6	rx_fifo_overflow	1'b0	R	Overflow flag of RX FIFO, cleared by rx_fifo_clr
5	tx_fifo_underflow	1'b0	R	Underflow flag of TX FIFO, cleared by tx_fifo_clr

Bits	Bit Name	Default	Type	Comment
4	tx_fifo_overflow	1'b0	R	Overflow flag of TX FIFO, cleared by tx_fifo_clr
3	rx_fifo_clr	1'b0	W1c	Clear signal of RX FIFO
2	tx_fifo_clr	1'b0	W1c	Clear signal of TX FIFO
1	i2c_dma_rx_en	1'b0	R/W	Enable signal of dma_rx_req/ack interface
0	i2c_dma_tx_en	1'b0	R/W	Enable signal of dma_tx_req/ack interface

Register: I2C_FIFO_CONFIG_1

Address: 0x20002084

Bits	Bit Name	Default	Type	Comment
31:25	Reserved	7'b0	R	Reserved, always read 'b0
24	rx_fifo_th	1'b0	R/W	RX FIFO threshold, dma_rx_req will not assert if rx_fifo_cnt is less than this value
23:17	Reserved	7'b0	R	Reserved, always read 'b0
16	tx_fifo_th	1'b0	R/W	TX FIFO threshold, dma_tx_req will not assert if tx_fifo_cnt is less than this value
15:10	Reserved	6'b0	R	Reserved, always read 'b0
9:8	rx_fifo_cnt	2'b00	R	RX FIFO available count
7:2	Reserved	6'b0	R	Reserved, always read 'b0
1:0	tx_fifo_cnt	2'b10	R	TX FIFO available count

Register: I2C_FIFO_WDATA

Address: 0x20002088

Bits	Bit Name	Default	Type	Comment
31:0	i2c_fifo_wdata	32'h0	W	I2C FIFO write data

Register: I2C_FIFO_RDATA

Address: 0x2000208C

Bits	Bit Name	Default	Type	Comment
31:0	i2c_fifo_rdata	32'h0	R	I2C FIFO read data

6.10 Local Timer Register

Local timer CSR address = 0x2000_3000 ~0x2000_3FFF

Register: TCCR

Address: 0x20003000

Bits	Bit Name	Default	Type	Comment
31:10	Reserved	22'b0	R	Reserved, always read 'b0
9:8	cs_wdt	2'b00	R/W	Clock source for timer WDT 00: fclk 01: f32_clk 10: 1 KHz 11: PLL 32 MHz
7	Reserved	1'b0	R	Reserved, always read 'b0
6:5	cs_3	2'b00	R/W	Clock source for timer #3 00: fclk 01: f32_clk 10: 1 KHz 11: PLL 32 MHz
4	Reserved	1'b0	R	Reserved, always read 'b0
3:2	cs_2	2'b00	R/W	Clock source for timer #2 00: fclk 01: f32_clk 10: 1 KHz 11: PLL 32 MHz
1:0	Reserved	2'b0	R	Reserved, always read 'b0

Register: TMR2_0

Address: 0x20003010

Bits	Bit Name	Default	Type	Comment
31:0	tmr_2_0	32'hFFFFFFFF	R/W	Timer2 match register 0

Register: TMR2_1

Address: 0x20003014

Bits	Bit Name	Default	Type	Comment
31:0	tmr_2_1	32'hFFFFFFFF	R/W	Timer2 match register 1

Register: TMR2_2

Address: 0x20003018

Bits	Bit Name	Default	Type	Comment
31:0	tmr_2_2	32'hFFFFFFFF	R/W	Timer2 match register 2

Register: TMR3_0

Address: 0x2000301C

Bits	Bit Name	Default	Type	Comment
31:0	tmr_3_0	32'hFFFFFFFF	R/W	Timer3 match register 0

Register: TMR3_1

Address: 0x20003020

Bits	Bit Name	Default	Type	Comment
31:0	tmr_3_1	32'hFFFFFFFF	R/W	Timer3 match register 1

Register: TMR3_2

Address: 0x20003024

Bits	Bit Name	Default	Type	Comment
31:0	tmr_3_2	32'hFFFFFFFF	R/W	Timer3 match register 2

Register: TCR2

Address: 0x2000302C

Bits	Bit Name	Default	Type	Comment
31:0	tcr2_counter	32'h0	R	Timer2 counter register

Register: TCR3

Address: 0x20003030

Bits	Bit Name	Default	Type	Comment
31:0	tcr3_counter	32'h0	R	Timer3 counter register

Register: TMSR2

Address: 0x20003038

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
2	tmsr2_2	1'b0	R	Timer2 match register 2 status. Clear interrupt would also clear this bit.
1	tmsr2_1	1'b0	R	Timer2 match register 1 status. Clear interrupt would also clear this bit.
0	tmsr2_0	1'b0	R	Timer2 match register 0 status. Clear interrupt would also clear this bit.

Register: TMSR3

Address: 0x2000303C

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tmsr3_2	1'b0	R	Timer3 match register 2 status. Clear interrupt would also clear this bit.
1	tmsr3_2	1'b0	R	Timer3 match register 1 status. Clear interrupt would also clear this bit.
0	tmsr3_2	1'b0	R	Timer3 match register 0 status. Clear interrupt would also clear this bit.

Register: TIER2

Address: 0x20003044

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tier2_2	1'b0	R/W	Timer2 match value 2 interrupt enable.
1	tier2_1	1'b0	R/W	Timer2 match value 1 interrupt enable.
0	tier2_0	1'b0	R/W	Timer2 match value 0 interrupt enable.

Register: TIER3

Address: 0x20003048

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tier3_2	1'b0	R/W	Timer3 match value 2 interrupt enable.
1	tier3_1	1'b0	R/W	Timer3 match value 1 interrupt enable.
0	tier3_0	1'b0	R/W	Timer3 match value 0 interrupt enable.

Register: TPLVR2

Address: 0x20003050

Bits	Bit Name	Default	Type	Comment
31:0	tplvr2	32'b0	R/W	Timer2 Pre-Load Value

Register: TPLVR3

Address: 0x20003054

Bits	Bit Name	Default	Type	Comment
31:0	tplvr3	32'b0	R/W	Timer3 Pre-Load Value

Register: TPLCR2

Address: 0x2000305C

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1:0	tplcr2	2'b0	R/W	Timer2 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

Register: TPLCR3

Address: 0x20003060

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1:0	tplcr3	2'b0	R/W	Timer3 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

Register: WMER

Address: 0x20003064

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1	wrie	1'b0	R/W	WDT reset/interrupt mode 1'b0 - WDT expiration to generate interrupt 1'b1 - WDT expiration to generate reset source
0	we	1'b0	R/W	WDT enable register

Register: WMR

Address: 0x20003068

Bits	Bit Name	Default	Type	Comment
31:17	Reserved	15'b0	R	Reserved, always read 'b0
16	wdt_align	1'b0	R/W	WDT compare value update align interrupt
15:0	wmr	16'hFFFF	R/W	WDT counter match value

Register: WVR

Address: 0x2000306C

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	16'b0	R	Reserved, always read 'b0
15:0	wdt_cnt	16'b0	R	WDT counter value

Register: WSR

Address: 0x20003070

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	31'b0	R	Reserved, always read 'b0
0	wts	1'b0	R/W	WDT reset status Write 0 to clear the WDT reset status Read 1 indicates reset was caused by the WDT

Register: TICR2

Address: 0x20003078

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tclr2_2	1'b0	R/W	Timer2 Interrupt clear for match comparator 2
1	tclr2_1	1'b0	R/W	Timer2 Interrupt clear for match comparator 1
0	tclr2_0	1'b0	R/W	Timer2 Interrupt clear for match comparator 0

Register: TICR3

Address: 0x2000307C

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tclr3_2	1'b0	R/W	Timer3 Interrupt clear for match comparator 2
1	tclr3_1	1'b0	R/W	Timer3 Interrupt clear for match comparator 1
0	tclr3_0	1'b0	R/W	Timer3 Interrupt clear for match comparator 0

Register: WICR

Address: 0x20003080

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	31'b0	R	Reserved, always read 'b0
0	wiclr	1'b0	R/W	WDT Interrupt Clear

Register: TCER

Address: 0x20003084

Bits	Bit Name	Default	Type	Comment
31:7	Reserved	25'b0	R	Reserved, always read 'b0
6	tcr3_cnt_clr	1'b0	R/W	Timer3 count clear
5	tcr2_cnt_clr	1'b0	R/W	Timer2 count clear
4:3	Reserved	2'b0	R	Reserved, always read 'b0
2	timer3_en	1'b0	R/W	Timer3 count enable
1	timer2_en	1'b0	R/W	Timer2 count enable
0	Reserved	1'b0	R	Reserved, always read 'b0

Register: TCMR

Address: 0x20003088

Bits	Bit Name	Default	Type	Comment
31:7	Reserved	25'b0	R	Reserved, always read 'b0
6	timer3_align	1'b0	R/W	Timer3 compare value update align interrupt
5	timer2_align	1'b0	R/W	Timer2 compare value update align interrupt
4:3	Reserved	2'b0	R	Reserved, always read 'b0
2	timer3_mode	1'b0	R/W	0:pre-load mode; 1:free run mode
1	timer2_mode	1'b0	R/W	0:pre-load mode; 1:free run mode
0	Reserved	1'b0	R	Reserved, always read 'b0

Register: TILR2

Address: 0x20003090

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tilr2_2	1'b0	R/W	0:level; 1:edge
1	tilr2_1	1'b0	R/W	0:level; 1:edge
0	tilr2_0	1'b0	R/W	0:level; 1:edge

Register: TILR3

Address: 0x20003094

Bits	Bit Name	Default	Type	Comment
31:3	Reserved	29'b0	R	Reserved, always read 'b0
2	tilr3_2	1'b0	R/W	0:level; 1:edge
1	tilr3_1	1'b0	R/W	0:level; 1:edge
0	tilr3_0	1'b0	R/W	0:level; 1:edge

Register: WCR

Address: 0x20003098

Note: Write only register, always read 'b0

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	31'b0	R	Reserved, always read 'b0
0	wcr	1'b0	WO	WDT Counter Reset

Register: WFAR

Address: 0x2000309C

Note: Write only register, always read 'b0

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	16'b0	R	Reserved, always read 'b0
15:0	wcr	16'b0	WO	WDT access key1 - 16'hBABA

Register: WSAR

Address: 0x200030A0

Note: Write only register, always read 'b0

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	16'b0	R	Reserved, always read 'b0
15:0	wsar	16'b0	WO	WDT access key2 - 16'hEB10

Register: TCVWR2

Address: 0x200030A8

Bits	Bit Name	Default	Type	Comment
31:0	tcr2_cnt_lat	32'b0	R	Timer2 Counter Latch Value

Register: TCVWR3

Address: 0x200030AC

Bits	Bit Name	Default	Type	Comment
31:0	tcr3_cnt_lat	32'b0	R	Timer3 Counter Latch Value

Register: TCVSTN2

Address: 0x200030B4

Bits	Bit Name	Default	Type	Comment
31:0	tcr2_cnt_sync	32'b0	R	Timer2 Counter Sync Value (continue readable)

Register: TCVSTN3

Address: 0x200030B8

Bits	Bit Name	Default	Type	Comment
31:0	tcr3_cnt_sync	32'b0	R	Timer3 Counter Sync Value (continue readable)

Register: TCDR

Address: 0x200030BC

Bits	Bit Name	Default	Type	Comment
31:24	wcdr	8'b0	R/W	WDT clock division value register
23:16	tcdr3	8'b0	R/W	Timer3 clock division value register
15:8	tcdr2	8'b0	R/W	Timer2 clock division value register
7:0	Reserved	8'b0	R	Reserved, always read 'b0

6.11 UART Control Register

UART CSR address = 0x2000_4000 ~ 0x2000_4FFF

Register: UTX_CONFIG

Address: 0x20004000

Bits	Bit Name	Default	Type	Comment
31:16	cr_utx_len	16'b0	R/W	Length of UART TX data transfer (Unit: character/byte) (Don't-care if cr_utx_frm_en is enabled)
15:13	cr_utx_bit_cnt_b	3'b100	R/W	UART TX BREAK bit count (for LIN protocol) Note: Additional 8 bit times will be added since LIN Break field requires at least 13 bit times
12:11	cr_utx_bit_cnt_p	2'b01	R/W	UART TX STOP bit count (unit: 0.5 bit)
10:8	cr_utx_bit_cnt_d	3'b111	R/W	TX data bit count for each character
7	cr_utx_ir_inv	1'b0	R/W	Inverse signal of TX output in IR mode
6	cr_utx_ir_en	1'b0	R/W	Enable signal of TX IR mode
5	cr_utx_prt_sel	1'b0	R/W	Select signal of TX parity bit 1: Odd parity 0: Even parity
4	cr_utx_prt_en	1'b0	R/W	Enable signal of TX parity bit
3	cr_utx_lin_en	1'b0	R/W	Enable signal of TX LIN mode (LIN header will be sent before sending data)
2	cr_utx_frm_en	1'b0	R/W	Enable signal of TX free run mode (utx_end_int will be disabled)
1	cr_utx_cts_en	1'b0	R/W	Enable signal of TX CTS flow control function.
0	cr_utx_en	1'b0	R/W	Enable signal of TX function

Register: URX_CONFIG

Address: 0x20004004

Bits	Bit Name	Default	Type	Comment
31:16	cr_urx_len	16'b0	R/W	Length of RX data (unit: character/byte). Urx_end_int will assert when this length is reached
15:12	cr_urx_deg_cnt	4'b0	R/W	De-glitch function cycle count
11	cr_urx_deg_en	1'b0	R/W	Enable signal of RXD input de-glitch function
10:8	cr_urx_bit_cnt_d	3'b111	R/W	RX data bit count for each character
7	cr_urx_ir_inv	1'b0	R/W	Inverse signal of RX output in IR mode
6	cr_urx_ir_en	1'b0	R/W	Enable signal of RX IR mode
5	cr_urx_prt_sel	1'b0	R/W	Select signal of RX parity bit 1: Odd parity 0: Even parity
4	cr_urx_prt_en	1'b0	R/W	Enable signal of RX parity bit
3	cr_urx_lin_en	1'b0	R/W	Enable signal of RX LIN mode (LIN header will be required and checked before receiving data)
2	Reserved	1'b0	R	Reserved, always read 'b0

Bits	Bit Name	Default	Type	Comment
1	cr_urx_abr_en	1'b0	R/W	Enable signal of RX auto baud rate detection function.
0	cr_urx_en	1'b0	R/W	Enable signal of RX function

Register: UART_BIT_PRD

Address: 0x20004008

Bits	Bit Name	Default	Type	Comment
31:16	cr_urx_bit_prd	16'h00FF	R/W	Period of each UART RX bit, related to baud rate.
15:0	cr_utx_bit_prd	16'h00FF	R/W	Period of each UART TX bit, related to baud rate.

Register: DATA_CONFIG

Address: 0x2000400C

Bits	Bit Name	Default	Type	Comment
31:1	Reserved	31'b0	R	Reserved, always read 'b0
0	cr_uart_bit_inv	1'b0	R/W	Bit inverse signal for each data byte 0: Each byte is sent LSB-first 1: Each byte is sent MSB-first

Register: UTX_IR_POSITION

Address: 0x20004010

Bits	Bit Name	Default	Type	Comment
31:16	cr_utx_ir_pos_p	16'h009F	R/W	Stop position of UART TX IR pulse.
15:0	cr_utx_ir_pos_s	16'h0070	R/W	Start position of UART TX IR pulse.

Register: URX_IR_POSITION

Address: 0x20004014

Bits	Bit Name	Default	Type	Comment
31:16	Reserved	16'b0	R	Reserved, always read 'b0
15:0	cr_urx_ir_pos_s	16'h6F	R/W	Start position of UART RXD pulse recovered from IR signal.

Register: URX_RTO_TIMER

Address: 0x20004018

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	cr_urx_rto_value	8'hF	R/W	Time-out value for triggering RTO interrupt (unit: bit time)

Register: UART_SW_MODE

Address: 0x2000401C

Bits	Bit Name	Default	Type	Comment
31:4	Reserved	28'b0	R	Reserved, always read 'b0
3	cr_urx_rts_sw_val	1'b0	R/W	UART RX RTS output SW control value
2	cr_urx_rts_sw_mode	1'b0	R/W	UART RX RTS output SW control mode
1	cr_utx_txd_sw_val	1'b0	R/W	UART TX TXD output SW control value
0	cr_utx_txd_sw_mode	1'b0	R/W	UART TX TXD output SW control mode

Register: UART_INT_STS

Address: 0x20004020

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	20'b0	R	Reserved, always read 'b0
11	urx_ad5_int	1'b0	R	UART RX ABR Detection finish interrupt using codeword 0x55
10	urx_ads_int	1'b0	R	UART RX ABR Detection finish interrupt using START bit
9	urx_bcr_int	1'b0	R	UART RX byte count reached interrupt
8	urx_lse_int	1'b0	R	UART RX LIN mode sync field error interrupt
7	urx_fer_int	1'b0	R	UART RX FIFO error interrupt. Auto cleared when FIFO overflow/underflow error flag is cleared.
6	utx_fer_int	1'b0	R	UART TX FIFO error interrupt. Auto cleared when FIFO overflow/underflow error flag is cleared.
5	urx_pce_int	1'b0	R	UART RX parity check error interrupt.
4	urx_rto_int	1'b0	R	UART RX timeout interrupt
3	urx_frdy_int	1'b0	R	UART RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto cleared when data is popped.
2	utx_frdy_int	1'b1	R	UART TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto cleared when data is pushed.
1	urx_end_int	1'b0	R	UART RX transfer end interrupt (set according to cr_urx_len)
0	utx_end_int	1'b0	R	UART TX transfer end interrupt (set according to cr_utx_len)

Register: UART_INT_MASK

Address: 0x20004024

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	20'b0	R	Reserved, always read 'b0
11	cr_urx_ad5_mask	1'b1	R/W	Interrupt mask of urx_ad5_int
10	cr_urx_ads_mask	1'b1	R/W	Interrupt mask of urx_ads_int
9	cr_urx_bcr_mask	1'b1	R/W	Interrupt mask of urx_bcr_int
8	cr_urx_lse_mask	1'b1	R/W	Interrupt mask of urx_lse_int
7	cr_urx_fer_mask	1'b1	R/W	Interrupt mask of urx_fer_int

Bits	Bit Name	Default	Type	Comment
6	cr_utx_fer_mask	1'b1	R/W	Interrupt mask of utx_fer_int
5	cr_urx_pce_mask	1'b1	R/W	Interrupt mask of urx_pce_int
4	cr_urx_rto_mask	1'b1	R/W	Interrupt mask of urx_rto_int
3	cr_urx_frdy_mask	1'b1	R/W	Interrupt mask of urx_fifo_int
2	cr_utx_frdy_mask	1'b1	R/W	Interrupt mask of utx_fifo_int
1	cr_urx_end_mask	1'b1	R/W	Interrupt mask of urx_end_int
0	cr_utx_end_mask	1'b1	R/W	Interrupt mask of utx_end_int

Register: UART_INT_CLR

Address: 0x20004028

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	20'b0	R	Reserved, always read 'b0
11	cr_urx_ad5_clr	1'b0	R/W1c	Interrupt clear of urx_ad5_int
10	cr_urx_ads_clr	1'b0	R/W1c	Interrupt clear of urx_ads_int
9	cr_urx_bcr_clr	1'b0	R/W1c	Interrupt clear of urx_bcr_int
8	cr_urx_lse_clr	1'b0	R/W1c	Interrupt clear of urx_lse_int
7:6	Reserved	2'b0	R/W	Reserved
5	cr_urx_pce_clr	1'b0	R/W1c	Interrupt clear of urx_pce_int
4	cr_urx_rto_clr	1'b0	R/W1c	Interrupt clear of urx_rto_int
3:2	Reserved	2'b0	R/W	Reserved
1	cr_urx_end_clr	1'b0	R/W1c	Interrupt clear of urx_end_int
0	cr_utx_end_clr	1'b0	R/W1c	Interrupt clear of utx_end_int

Register: UART_INT_EN

Address: 0x2000402C

Bits	Bit Name	Default	Type	Comment
31:12	Reserved	20'b0	R	Reserved, always read 'b0
11	cr_urx_ad5_en	1'b1	R/W	Interrupt enable of urx_ad5_int
10	cr_urx_ads_en	1'b1	R/W	Interrupt enable of urx_ads_int
9	cr_urx_bcr_en	1'b1	R/W	Interrupt enable of urx_bcr_int
8	cr_urx_lse_en	1'b1	R/W	Interrupt enable of urx_lse_int
7	cr_urx_fer_en	1'b1	R/W	Interrupt enable of urx_fer_int
6	cr_utx_fer_en	1'b1	R/W	Interrupt enable of utx_fer_int
5	cr_urx_pce_en	1'b1	R/W	Interrupt enable of urx_pce_int
4	cr_urx_rto_en	1'b1	R/W	Interrupt enable of urx_rto_int
3	cr_urx_frdy_en	1'b1	R/W	Interrupt enable of urx_fifo_int
2	cr_utx_frdy_en	1'b1	R/W	Interrupt enable of utx_fifo_int
1	cr_urx_end_en	1'b1	R/W	Interrupt enable of urx_end_int
0	cr_utx_end_en	1'b1	R/W	Interrupt enable of utx_end_int

Register: UART_STATUS

Address: 0x20004030

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1	sts_urx_bus_busy	1'b0	R	Indicator of UART RX bus busy
0	sts_utx_bus_busy	1'b0	R	Indicator of UART TX bus busy

Register: STS_URX_ABR_PRD

Address: 0x20004034

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_0x55	16'b0	R	Bit period of Auto Baud Rate detection using codeword 0x55
15:0	sts_urx_abr_prd_start	16'b0	R	Bit period of Auto Baud Rate detection using START bit

Register: URX_ABR_PRD_B01

Address: 0x20004038

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit1	16'b0	R	Bit period of Auto Baud Rate detection - bit[1]
15:0	sts_urx_abr_prd_bit0	16'b0	R	Bit period of Auto Baud Rate detection - bit[0]

Register: URX_ABR_PRD_B23

Address: 0x2000403C

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit3	16'b0	R	Bit period of Auto Baud Rate detection - bit[3]
15:0	sts_urx_abr_prd_bit2	16'b0	R	Bit period of Auto Baud Rate detection - bit[2]

Register: URX_ABR_PRD_B45

Address: 0x20004040

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit5	16'b0	R	Bit period of Auto Baud Rate detection - bit[5]
15:0	sts_urx_abr_prd_bit4	16'b0	R	Bit period of Auto Baud Rate detection - bit[4]

Register: URX_ABR_PRD_B67

Address: 0x20004044

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_abr_prd_bit7	16'b0	R	Bit period of Auto Baud Rate detection - bit[7]
15:0	sts_urx_abr_prd_bit6	16'b0	R	Bit period of Auto Baud Rate detection - bit[6]

Register: URX_ABR_PW_TOL

Address: 0x20004048

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	cr_urx_abr_pw_tol	8'h3	R/W	Auto Baud Rate detection pulse-width tolerance for using codeword 0x55

Register: URX_BCR_INT_CFG

Address: 0x20004050

Bits	Bit Name	Default	Type	Comment
31:16	sts_urx_bcr_count	16'b0	R	Current byte count of urx_bcr_int counter, auto-cleared bt cr_urx_bcr_clr
15:0	cr_urx_bcr_value	16'hFFFF	R/W	Byte count setting for urx_bcr_int counter

Register: UTX_RS485_CFG

Address: 0x20004054

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1	cr_utx_rs485_pol	1'b1	R/W	cr_utx_rs485_pol
0	cr_utx_rs485_en	1'b0	R/W	cr_utx_rs485_pol

Register: UART_FIFO_CONFIG_0

Address: 0x20004080

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7	rx_fifo_underflow	1'b0	R	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	1'b0	R	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	1'b0	R	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	1'b0	R	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	1'b0	R/W1c	Clear signal of RX FIFO
2	tx_fifo_clr	1'b0	R/W1c	Clear signal of TX FIFO
1	uart_dma_rx_en	1'b0	R/W	Enable signal of dma_rx_req/ack interface

Bits	Bit Name	Default	Type	Comment
0	uart_dma_tx_en	1'b0	R/W	Enable signal of dma_tx_req/ack interface

Register: UART_FIFO_CONFIG_1

Address: 0x20004084

Bits	Bit Name	Default	Type	Comment
31	Reserved	1'b0	R	Reserved, always read 'b0
30:24	rx_fifo_th	7'b0	R/W	RX FIFO threshold, dma_rx_req will not be asserted if rx_fifo_cnt is less than this value
23	Reserved	1'b0	R	Reserved, always read 'b0
22:16	tx_fifo_th	7'b0	R/W	TX FIFO threshold, dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15:8	rx_fifo_cnt	8'b0	R	RX FIFO available count
7:0	tx_fifo_cnt	8'h80	R	TX FIFO available count

Register: UART_FIFO_WDATA

Address: 0x20004088

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	uart_fifo_wdata	8'b0	W	UART FIFO write data

Register: UART_FIFO_RDATA

Address: 0x2000408C

Bits	Bit Name	Default	Type	Comment
31:8	Reserved	24'b0	R	Reserved, always read 'b0
7:0	uart_fifo_rdata	8'b0	R	UART FIFO read data

6.12 GPIO Control Register

GPIO CSR address = 0x2000_5000 ~ 0x2000_5FFF

Register: GPIO_MISC

Address: 0x20005080

Bits	Bit Name	Default	Type	Comment
31:2	Reserved	30'b0	R	Reserved, always read 'b0
1	reg_spi_swap	1'b0	R/W	SPI config: swap miso/mosi
0	reg_spi_master_mode	1'b0	R/W	SPI config: master/slave mode

Register: GPIO_CFGCTL0

Address: 0x20005100

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_1_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_1_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_1_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_1_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_1_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_1_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_0_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_0_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_0_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_0_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_0_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_0_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL1

Address: 0x20005104

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_3_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_3_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_3_pu	1'b0	R/W	GPIO pull up control

Bits	Bit Name	Default	Type	Comment
19:18	reg_gpio_3_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_3_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_3_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_2_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_2_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_2_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_2_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_2_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_2_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL2

Address: 0x20005108

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_5_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_5_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_5_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_5_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_5_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_5_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_4_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_4_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_4_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_4_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_4_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_4_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL3

Address: 0x2000510C

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_7_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_7_pd	1'b0	R/W	GPIO pull down control

Bits	Bit Name	Default	Type	Comment
20	reg_gpio_7_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_7_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_7_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_7_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_6_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_6_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_6_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_6_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_6_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_6_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL4

Address: 0x20005110

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_9_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_9_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_9_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_9_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_9_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_9_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_8_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_8_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_8_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_8_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_8_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_8_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL5

Address: 0x20005114

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_11_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_11_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_11_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_11_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_11_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_11_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_10_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_10_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_10_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_10_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_10_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_10_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL6

Address: 0x20005118

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_13_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_13_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_13_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_13_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_13_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_13_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_12_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_12_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_12_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_12_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_12_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_12_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL7

Address: 0x2000511C

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_15_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_15_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_15_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_15_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_15_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_15_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_14_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_14_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_14_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_14_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_14_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_14_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL8

Address: 0x20005120

Bits	REGISTER NAME	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_17_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_17_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_17_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_17_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_17_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_17_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_16_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_16_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_16_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_16_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_16_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_16_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL9

Address: 0x20005124

Bits	Bit Name	Default	Type	Comment
31:28	Reserved	4'b0	R	Reserved, always read 'b0
27:24	reg_gpio_19_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
23:22	Reserved	2'b0	R	Reserved, always read 'b0
21	reg_gpio_19_pd	1'b0	R/W	GPIO pull down control
20	reg_gpio_19_pu	1'b0	R/W	GPIO pull up control
19:18	reg_gpio_19_drv	2'b0	R/W	GPIO driving control
17	reg_gpio_19_smt	1'b1	R/W	GPIO SMT control
16	reg_gpio_19_ie	1'b1	R/W	GPIO input enable
15:12	Reserved	4'b0	R	Reserved, always read 'b0
11:8	reg_gpio_18_func_sel	4'b0	R/W	GPIO function select (Default: SWGPIO)
7:6	Reserved	2'b0	R	Reserved, always read 'b0
5	reg_gpio_18_pd	1'b0	R/W	GPIO pull down control
4	reg_gpio_18_pu	1'b0	R/W	GPIO pull up control
3:2	reg_gpio_18_drv	2'b0	R/W	GPIO driving control
1	reg_gpio_18_smt	1'b1	R/W	GPIO SMT control
0	reg_gpio_18_ie	1'b1	R/W	GPIO input enable

Register: GPIO_CFGCTL30

Address: 0x20005180

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19	reg_gpio_19_i	1'b0	R	Register controlled GPIO input value
18	reg_gpio_18_i	1'b0	R	Register controlled GPIO input value
17	reg_gpio_17_i	1'b0	R	Register controlled GPIO input value
16	reg_gpio_16_i	1'b0	R	Register controlled GPIO input value
15	reg_gpio_15_i	1'b0	R	Register controlled GPIO input value
14	reg_gpio_14_i	1'b0	R	Register controlled GPIO input value
13	reg_gpio_13_i	1'b0	R	Register controlled GPIO input value
12	reg_gpio_12_i	1'b0	R	Register controlled GPIO input value
11	reg_gpio_11_i	1'b0	R	Register controlled GPIO input value
10	reg_gpio_10_i	1'b0	R	Register controlled GPIO input value
9	reg_gpio_9_i	1'b0	R	Register controlled GPIO input value
8	reg_gpio_8_i	1'b0	R	Register controlled GPIO input value
7	reg_gpio_7_i	1'b0	R	Register controlled GPIO input value
6	reg_gpio_6_i	1'b0	R	Register controlled GPIO input value
5	reg_gpio_5_i	1'b0	R	Register controlled GPIO input value
4	reg_gpio_4_i	1'b0	R	Register controlled GPIO input value

Bits	Bit Name	Default	Type	Comment
3	reg_gpio_3_i	1'b0	R	Register controlled GPIO input value
2	reg_gpio_2_i	1'b0	R	Register controlled GPIO input value
1	reg_gpio_1_i	1'b0	R	Register controlled GPIO input value
0	reg_gpio_0_i	1'b0	R	Register controlled GPIO input value

Register: GPIO_CFGCTL32

Address: 0x20005188

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19	reg_gpio_19_o	1'b0	R/W	Register controlled GPIO output Value
18	reg_gpio_18_o	1'b0	R/W	Register controlled GPIO output Value
17	reg_gpio_17_o	1'b0	R/W	Register controlled GPIO output Value
16	reg_gpio_16_o	1'b0	R/W	Register controlled GPIO output Value
15	reg_gpio_15_o	1'b0	R/W	Register controlled GPIO output Value
14	reg_gpio_14_o	1'b0	R/W	Register controlled GPIO output Value
13	reg_gpio_13_o	1'b0	R/W	Register controlled GPIO output Value
12	reg_gpio_12_o	1'b0	R/W	Register controlled GPIO output Value
11	reg_gpio_11_o	1'b0	R/W	Register controlled GPIO output Value
10	reg_gpio_10_o	1'b0	R/W	Register controlled GPIO output Value
9	reg_gpio_9_o	1'b0	R/W	Register controlled GPIO output Value
8	reg_gpio_8_o	1'b0	R/W	Register controlled GPIO output Value
7	reg_gpio_7_o	1'b0	R/W	Register controlled GPIO output Value
6	reg_gpio_6_o	1'b0	R/W	Register controlled GPIO output Value
5	reg_gpio_5_o	1'b0	R/W	Register controlled GPIO output Value
4	reg_gpio_4_o	1'b0	R/W	Register controlled GPIO output Value
3	reg_gpio_3_o	1'b0	R/W	Register controlled GPIO output Value
2	reg_gpio_2_o	1'b0	R/W	Register controlled GPIO output Value
1	reg_gpio_1_o	1'b0	R/W	Register controlled GPIO output Value
0	reg_gpio_0_o	1'b0	R/W	Register controlled GPIO output Value

Register: GPIO_CFGCTL34

Address: 0x20005190

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19	reg_gpio_19_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)

Bits	Bit Name	Default	Type	Comment
18	reg_gpio_18_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
17	reg_gpio_17_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
16	reg_gpio_16_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
15	reg_gpio_15_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
14	reg_gpio_14_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
13	reg_gpio_13_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
12	reg_gpio_12_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
11	reg_gpio_11_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
10	reg_gpio_10_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
9	reg_gpio_9_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
8	reg_gpio_8_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
7	reg_gpio_7_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
6	reg_gpio_6_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
5	reg_gpio_5_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
4	reg_gpio_4_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
3	reg_gpio_3_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)

Bits	Bit Name	Default	Type	Comment
2	reg_gpio_2_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
1	reg_gpio_1_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)
0	reg_gpio_0_oe	1'b0	R/W	Register controlled GPIO output enable (used when GPIO function select to register control GPIO)

Register: GPIO_INT_MASK1

Address: 0x200051A0

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19:0	reg_gpio_int_mask1	20'hFFFFF	R/W	reg_gpio_int_mask [19:0]

Register: GPIO_INT_STAT1

Address: 0x200051A8

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19:0	gpio_int_stat1	20'b0	R	gpio_int_stat [19:0]

Register: GPIO_INT_CLR1

Address: 0x200051B0

Bits	Bit Name	Default	Type	Comment
31:20	Reserved	12'b0	R	Reserved, always read 'b0
19:0	reg_gpio_int_clr1	20'b0	R/W	reg_gpio_int_clr [19:0]

Register: GPIO_INT_MODE_SET1

Address: 0x200051C0

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	12'b0	R	Reserved, always read 'b0
29:0	reg_gpio_int_mode_set1	20'b0	R/W	{reg_gpio9_int_mode [2:0], ……. reg_gpio0_int_mode [2:0]}

Register: GPIO_INT_MODE_SET2

Address: 0x200051C4

Bits	Bit Name	Default	Type	Comment
31:30	Reserved	12'b0	R	Reserved, always read 'b0
29:0	reg_gpio_int_mode_set2	20'b0	R/W	{reg_gpio19_int_mode [2:0], ……. reg_gpio10_int_mode [2:0]}

6.13 I3C Register

I3C Master Register Base address = 0x2A000000

I3C Slave Register Base address = 0x2A010000

For more information, see Chapter 5 of DWC_mipi_i3c_databook.pdf here

6.14 AXI4TG Register

AXI4TG Register Base address = 0x2A020000

For more information, see axi-traffic-gen.pdf. here

Notes: Address width is configured to 32, the Address RAM is not present and cannot be accessed.

6.15 DDR4 MC Register

DDR4 MC Register Base address = 0x21000000

For more information, see mc_reg_r4710.html. here

6.16 DDR4 PHY Register

DDR4 PHY Register Base address = 0x22000000

For more information, see Chapter 12 of dwc_ddr43_phy_pub_databook.pdf. here

6.17 D2D PHY Register

D2D PHY Register Base address = 0x27000000

For more information, see `blynx_phy_csr.pdf`. here

6.18 D2D Register

D2D Register Base address = 0x27100000

For more information, see `d2d_reg.html`. here

Chapter 7 Acronyms

Table provides definitions for the acronyms, abbreviations, and terms used in this document.

Table 23 Acronyms

Acronym or term	Definition
ACK	Acknowledge
AES	Auger Electron Spectroscopy
AHB	Advanced High-Performance Bus
AIA	Advanced Interrupt Architecture
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
AXI	Advanced Extensible Interface
AXI4	Advanced Extensible Interface (Version 4)
AXI4TG	Advanced Extensible Interface (Version 4) Traffic Generator
AXI5	Advanced Extensible Interface (Version 5)
BFM	Bus Functional Model
BMX	Bus Matrix
BoW	Bunch-Of-Wires
CA	Chip Address
CLIC	Core Local Interrupt Controller
CRC	Cyclic Redundancy Check
CS	Chip Select
CSR	Control And Status Register
DAA	Dynamic Address Assignment
DAC	Digital-To-Analog Converter
DBI	Direct Bond Interconnect
DDR	Double Data Rate
DDR4	Double Data Rate 4
DDRPHY	Dual Data Rate Physical interface
DFI	DDR PHY interface
DIMM	Dual In-Line Memory Module
DLL	Delay Locked Loops
DMA	Direct Memory Access
DRAM	Dynamic Random-Access Memory
DSM	Dynamic Single-Mode
DQ	DDR Data line
D2D	Die-To-Die
ECC	Error Correction Codes
ECO	Engineering Change Order
ELA	Embedded Logic Analyzer
FLC	Final Level Cache

Acronym or term	Definition
FSBL	First Stage Bootloader
GFH	Goldfinch
GPIO	General-Purpose Input/Output
hPPR	Hard Post-Package Repair
HTRANS	Transfer type, this can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY
IMSIC	Incoming MSI Controller
IP	Intellectual Property
IPM	In-Package Memory
IR	Instruction Register
IRQ	Interrupt Request
I2C	Inter Integrated Circuit
I3C	Improved Inter Integrated Circuit
JTAG	Joint Test Action Group
LIN	Local Interconnect Network
LRDIMM	Load Reduced Dual In-Line Memory Module
LSB	Least Significant Bit
LVCMOS	Low Voltage Complementary Metal Oxide Semiconductor
MAC	Memory Access Controller
MC	Memory Controller
µC	Micro Controller
MKB	Mockingbird
MPR	Multi-Purpose Register
MSB	Most Significant Bit
MSI	Medium Scale Integration Circuits
NMI	Non-Maskable Interrupt
OTP	One-Time Programmable
PCIe	Peripheral Component Interconnect Express
PDA	Personal Digital Assistant mode
PHY	Physical Layer
PLIC	Platform-Level Interrupt Controller
PLL	Phase-Locked Loop
PMP	Physical Memory Protection
PMU	Parametric Measurement Unit
POS	Power On Strapping
QoS	Quality Of Service
QPI	Quasiparticle Interference
QSPI	Quad Serial Peripheral Interface
RDIMM	Registered Dual In-Line Memory Module
RMW	Read Write Memory
ROM	Read Only Memory
RSVD	Reserved
RTO	Rapid Thermal Oxidation
RXD	Receive Data
SCL	Source-Coupled Logic
SCLK	Serial Clock

Acronym or term	Definition
SDA	Serial Data
SDI	Serial Data Input
SDO	Serial Data Output
SRAM	Static Random-Access Memory
SDRAM	Synchronous Dynamic Random-Access Memory
SEC	Security Engine
SECDED	Single Error Correct Double Error Detect
SF	Serial Flash
SHA	Secure Hash Algorithm
SIPI	Signal Integrity Power Integrity
SMT	Surface Mount Technology
SODIMM	Signal And Power Integrity
SPI	Serial Peripheral Interface
sPPR	Soft Post-Package Repair
TCK	Test Clock (JTAG)
TCM	Transmission Control Module
TDI	Test Data Input
TDO	Test Data Output
TG	Glass Transition Temperature
TIM	Thermal Interface Material
TMR	Test Mode Reset
TMS	Test Mode Select
TSMC	Taiwan Semiconductor Manufacturing Company
UART	Universal Asynchronous Receiver-Transmitter
UDIMM	Unbuffered Dual Inline Memory Module
VDDC	The core supply voltage
VDDQ	The supply voltage to the output buffers of a memory chip
VSS	Source Supply Voltage
WDT	Watchdog Timer
WFI	Wait For Interrupt
XIP	Execute In Place mode