MT7988A
Wi-Fi 7 Generation Router Platform: Datasheet

Open Version

Version: 0.1
Release Date: 2023-10-18

Summary of Key Features

Feature	Description
Application Processor	
CPU	Quad-core Arm® 1.8 GHz Cortex-A73 processor 64KB L1 I-cache 64KB L1 D-cache 1MB L2 cache DVFS capability
External Memory	
via DDR	16-bit/32-bit Up to 8GB density PCDDR4 up to 3200 MHz PCDDR3 up to 2133 MHz
via SPI	NOR flash/NAND flash devices
via eMMC	eMMC v5.1 devices
Connectivity	
Ethernet	4 x Gigabit Ethernet (GbE) ports 1 x 2.5G Gigabit Ethernet (2.5GbE) port 2 x USXGMII/HSGMII ports
Networking Frame Engine	Packet Aggregated DMA QoS DMA
Crypto	Look-aside and inline encryption and decryption
TOPS	Tunnel Offload Processor System Five-core microprocessor
PCIe	2 x PCIe Gen 3.0 2-lane ports 2 x PCIe Gen 3.0 1-lane ports (1)
Peripheral	
USB	2 x USB 3.2 Gen 1 ports with built-in PHY (1)
UART	2 x UART (4-pin) ports 1 x UART-Lite (2-pin) port
I2C	2 x I2C interfaces, up to 400 kHz
SPI	3 x Generic SPIs (2 SPI pins are shared with NOR/NAND flash)
Audio	1 x I2S/PCM output interface
PWM	8 x PWM interfaces

Note:

(1) 1 PCIe Gen 3.0 1-lane port and 1 USB 3.2 Gen 1 port physically share the same interface. Only one of them may be enabled at a time.

1 Introduction

1.1 General Description

The MediaTek MT7988A is a world-leading network processing platform for high-performance and reliable networking experiences, both in wired and wireless applications. The MT7988A comprises a rich connection interface sets including 4 Gigabit Ethernet ports, 2 USXGMII interfaces, 4 PCIe 3.0 interfaces, and 2 USB 3.2 Gen1 ports.

The MT7988A further enables seamless Wi-Fi 7 tri-band, 2.4 GHz, 5 GHz and 6 GHz, connectivity, with its Wi-Fi 7 companion chip that features 320 MHz bandwidth, 4096-QAM, MLO, MRU, and AFC.

To answer the need of fast-evolving tunneling applications, the MT7988A comes with MediaTek Tunnel Offload Processor System (TOPS), which facilitates the processing of a wide range of tunneling protocols. The MT7988A optimizes networking performance with exquisitely tuned ISA of MediaTek TOPS, and is dedicated to lifting networking offloading performance to premium level.

1.1.1 Functional Block Diagram

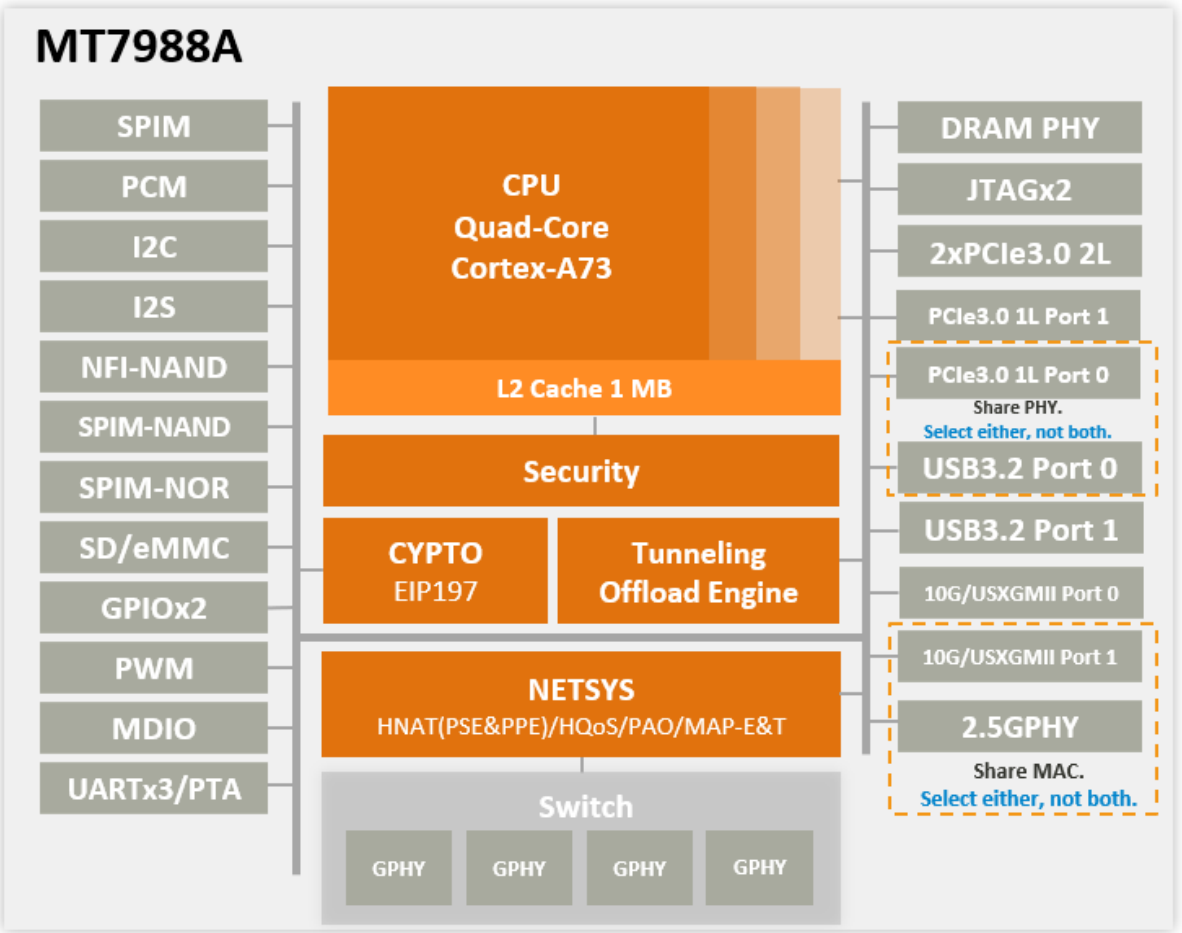


Figure 1-1. MT7988A Functional Block Diagram

1.2 Features

1.2.1 Platform Features

- AP MCU Subsystem
  - Quad-core Arm® 1.8 GHz Cortex-A73 processor
  - 64KB L1 I-cache
  - 64KB L1 D-cache
  - 1MB L2 Cache
  - DVFS capability

- Security
  - Fault-tolerance (Fault Injection Countermeasure)
    - Secure Boot-ROM
    - Power Glitch Detection (PGD)
  - TRNG 800-90C
  - Anti-rollback
  - Secure Key Storage, unique per-chip Key: HUK
  - Firmware encryption (ACPU firmware encryption supported from using BL2 on flash)
  - Cryptography function (AES)

- External Memory Interface
  - Memory clock up to PCDDR3-2133 / PCDDR4-3200
  - DDR memory space up to 8GB

- Peripherals
  - 2 x PCIe 3.0 2-lane ports
  - 2 x PCIe 3.0 1-lane ports
  - 2 x USB 3.2 Gen1 ports
  - 1 x eMMC 5.1 port
  - 2 x UART ports
  - 1 x UART-Lite port
  - 3 x SPIM interfaces
    - Support SPI-NAND with on-die ECC
    - Support SPI-NOR
  - 1 x Serial NAND Flash Interface with 24-bit hardware ECC engine
  - 2 x I2C interfaces
  - 8 x PWM channels
  - GPIOs

1.2.2 Connectivity Features

- Frame Engine
  - Packet aggregated DMA (A-DMA)
    - 4 TX descriptors and 4 RX descriptor rings
    - Scatter/Gather DMA
    - Configurable 4-, 8-, 16- or 32-bit burst length and delayed interrupt
    - Supports LRO, RSS
  - QoS DMA (QDMA)
    - Supports 128 TX physical queues and 4 sets of schedulers
    - Per TX queue forward/drop packet accounting
    - Per TX queue forward byte accounting
    - Supports TX queue min./max. rate control and SP/WFQ egress scheduler
    - Supports up to 1,024 virtual queues for 8 sets of SFQ
  - Packet Switch Engine (PSE)
    - Egress rate limiting and shaping
    - IP, TCP and UDP checksum offload
    - IP, TCP and UDP checksum generation
    - VLAN and PPPoE header insertion
    - TCP segmentation offload
  - Look-aside and inline encryption and decryption engine (EIP-197)
    - Optional MAC header parsing (Ethernet II and IEEE 802.2 LLC/SNAP), including VLAN and PBB
    - IPv4/IPv6 and IPsec-ESP/DTLS header parsing to look-up a flow or transform
  - Packet Processing Engine (PPE)
    - IPv4/IPv6 NAT/NAPT and routing
    - IPv4/IPv6 transition mechanism: Tunneling (DS-Lite, 6RD, MAP-E/T) and transition (464XLAT)
    - 1K, 2K, 4K, 8K, 16K or 32K sessions and flows
    - Flow offloading technology for flexible and high-performance packet L3/L4 packet processing

    Note: All PPE features mentioned above require software porting to function.

- Wi-Fi Warp
  - Wi-Fi <--> Ethernet, Wi-Fi <--> Wi-Fi packet forwarding offload
  - Packet aggregation offload
  - Dynamic buffer allocation and release

- 4-port Embedded L2 Switch
  - Supports full L2 bridge function among 4 ports embedded GPHY
  - Internal 384-KB packet SRAM for 64-byte wire speed
  - 2K MAC address shared with IGMP table
  - 4K VLAN table selectable for IVL and SVL
  - 256 ACL rule table mapped to 128 action entries

- Gigabit MAC (GMAC)
  - IEEE 802.3 MAC and 802.3x full duplex flow control
  - SGMII and HSGMII SerDes interface
    - 1 Gbps, 100 or 10 Mbps data rate with SGMII (8b/10b Auto-Negotiation)
    - Fixed 2.5 Gbps data rate with HSGMII

- 10 Gigabit MAC (XGMAC)
  - MAC layer functions of IEEE 802.3 and IEEE 802.3x flow control in full duplex mode
  - 10, 5, 2.5 or 1 Gbps, or 100 Mbps data rate with 64-bit XGMII interface
  - XGMII TX to RX loopback
  - 2.5 Gbps MACsec with 32 SA (Security Association)

- Tunnel Offload Processor System (TOPS)
  - Five-core microprocessor
  - High-performance bus fabric and peripherals
  - Tunnel encapsulation or decapsulation functions offloading
  - Boosts packet forwarding rate for different tunnel types

2 Abbreviations

2.1 Abbreviations

Table 2-1. Abbreviations

Abbreviation	Description
6RD	IPv6 Rapid Deployment
ACL	Access Control List
ACPU	Application CPU
AES	Advanced Encryption Standard
AFC	Automated Frequency Coordination
AP	Access Point
DDR	Double Data Rate
DMA	Direct Memory Access
DS-Lite	IPv6 Dual-Stack Lite
DVFS	Dynamic Voltage and Frequency Scaling
DTLS	Datagram Transport Layer Security
eMMC	Embedded MultiMediaCard
ECC	Error Correction Code
EEE	Energy-Efficient Ethernet
ESP	Encapsulating Security Payload
FIFO	First in, First Out
NFI	NAND Flash Interface
GbE	Gigabit Ethernet
GPHY	Gigabit Ethernet PHY
GPIO	General-Purpose Input/Output
HNAT	Host Network Address Translation
HQoS	Hierarchical Quality of Service
HSGMII	High Serial Gigabit Media-Independent Interface
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
IGMP	Internet Group Management Protocol
ISA	Instruction Set Architecture
IP	Internet Protocol
IPsec	Internet Protocol Security
IVL	Independent VLAN
JTAG	Joint Test Action Group
LLC	Logical Link Control
LRO	Large Receive Offload
NAPT	Network Address and Port Translation
NAT	Network Address Translation
MAP-E/T	Mapping Address Port Encapsulation/Translation
MCU	Microcontroller
MLO	Multi-Link Operation
MRU	Multiple Resource Unit
PAO	Partial A-MSDU (Aggregated MAC Service Data Unit) Operation

Abbreviation	Description
PBB	Provider Backbone Bridge
PCDDR3	Personal Computer Double Data Rate 3rd Generation
PCDDR4	Personal Computer Double Data Rate 4th Generation
PCIe	Peripheral Component Interconnect Express
PCM	Pulse Code Modulation
PGD	Power Glitch Detection
PHY	Physical layer
PMIC	Power Management Integrated Circuit
PPE	Packet Processing Engine
PPPoE	Point-to-Point Protocol over Ethernet
PSE	Packet Switch Engine
PWM	Pulse-Width Modulation
QAM	Quadrature Amplitude Modulation
QDMA	QoS DMA
QoS	Quality of Service
RoHS	Restriction of Hazardous Substances
ROM	Read-only Memory
RSS	Receive Side Scaling
SA	Security Association
SerDes	Serializer/Deserializer
SFQ	Stochastic Fairness Queueing
SGMII	Serial Gigabit Media-Independent Interface
SMI	Serial Management Interface
SNAP	Subnetwork Access Protocol
SP	Strict Priority
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SVL	Shared VLAN
TCP	Transmission Control Protocol
TOPS	Tunnel Offload Processor System
TRNG	True Random Number Generator
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
USXGMII	Universal Serial 10 Gigabit Media-Independent Interface
VLAN	Virtual Local Area Network
XGMAC	10 Gigabit MAC
XGMII	10 Gigabit Media-Independent Interface
WFQ	Weighted-Fair Queuing

3 Pin Information

3.1 Strapping Options

Table 3-1. Strapping Options

Strap Definition/ Strap Pin Name	Strap Mode/ Strap Value			
Reference Clock Source	**Crystal**	**Oscillator**		
LED_A	0	1		
DDR Data Width	**DDR3/4 16-bit x 2**	**DDR3/4 16-bit x 1**		
LED_B	0	1		
Boot Sequence	**SPI-NOR (SPI2)**	**SPI-NAND→SD (SPI0)**	**eMMC (eMMC)**	**SNFI → SD (SPI0)**
LED_D	0	0	1	1
LED_C	0	1	0	1

3.2 Pin Sharing Schemes

Some pins are shared with GPIO for maximum flexibility for system designers. You can configure related registers to specify pin function.

Table 3-2. Pin Share Table

Ball Name	Aux Func.0	Aux Func.1	Aux Func.2	Aux Func.3	Aux Func.4	Aux Func.5	Aux Func.6
UART2_RXD	GPIO0	UART2_RXD	-	-	-	-	-
UART2_TXD	GPIO1	UART2_TXD	-	-	-	-	-
UART2_CTS	GPIO2	UART2_CTS	-	-	-	-	-
UART2_RTS	GPIO3	UART2_RTS	-	-	-	-	-
GPIO_A	GPIO4	-	-	PWMD7	-	-	-
SMI_0_MDC	GPIO5	SMI_0_MDC	I2C_0_SCL	-	I2C_1_SCL	-	-
SMI_0_MDIO	GPIO6	SMI_0_MDIO	I2C_0_SDA	-	I2C_1_SDA	-	-
PCIE30_2L_0_WAKE_N	GPIO7	PCIE30_2L_0_WAKE_N	-	-	-	-	-
PCIE30_2L_0_CLKREQ_N	GPIO8	PCIE30_2L_0_CLKREQ_N	-	-	-	-	-
PCIE30_1L_1_WAKE_N	GPIO9	PCIE30_1L_1_WAKE_N	-	-	-	-	-
PCIE30_1L_1_CLKREQ_N	GPIO10	PCIE30_1L_1_CLKREQ_N	-	-	-	-	-
GPIO_P	GPIO11	-	-	-	-	-	-
WATCHDOG	GPIO12	WATCHDOG	-	-	-	-	-
GPIO_RESET	GPIO13	-	-	-	-	-	-
GPIO_WPS	GPIO14	-	-	-	-	-	-
PMIC_I2C_SCL	GPIO15	I2C_0_SCL	-	-	-	-	-
PMIC_I2C_SDA	GPIO16	I2C_0_SDA	-	-	-	-	-
I2C_1_SCL	GPIO17	I2C_1_SCL	-	-	-	-	-
I2C_1_SDA	GPIO18	I2C_1_SDA	-	-	-	-	-
PCIE30_2L_0_PRESET_N	GPIO19	PCIE30_2L_0_PRESET_N	-	-	-	-	-
PCIE30_1L_1_PRESET_N	GPIO20	PCIE30_1L_1_PRESET_N	-	-	-	-	-
PWMD1	GPIO21	PWMD1	-	-	-	EMMC_RSTB	-
SPI0_WP	GPIO22	SPI0_WP	SNFI_WP	-	-	-	-
SPI0_HOLD	GPIO23	SPI0_HOLD	SNFI_HOLD	-	-	-	-
SPI0_CSB	GPIO24	SPI0_CSB	SNFI_CS	-	-	-	-

Ball Name	Aux Func.0	Aux Func.1	Aux Func.2	Aux Func.3	Aux Func.4	Aux Func.5	Aux Func.6
SPI0_MISO	GPIO25	SPI0_MISO	SNFI_MISO	-	-	-	-
SPI0_MOSI	GPIO26	SPI0_MOSI	SNFI_MOSI	-	-	-	-
SPI0_CLK	GPIO27	SPI0_CLK	SNFI_CLK	-	-	-	-
SPI1_CSB	GPIO28	SPI1_CSB	UART2_RXD	-	-	EMMC_DATA_7	-
SPI1_MISO	GPIO29	SPI1_MISO	UART2_TXD	-	-	EMMC_DATA_6	-
SPI1_MOSI	GPIO30	SPI1_MOSI	UART2_CTS	-	-	EMMC_DATA_5	-
SPI1_CLK	GPIO31	SPI1_CLK	UART2_RTS	-	-	EMMC_DATA_4	-
SPI2_CLK	GPIO32	SPI2_CLK	UART1_RXD	UART2_RXD	-	EMMC_DATA_3	-
SPI2_MOSI	GPIO33	SPI2_MOSI	UART1_TXD	UART2_TXD	-	EMMC_DATA_2	-
SPI2_MISO	GPIO34	SPI2_MISO	UART1_CTS	UART2_CTS	-	EMMC_DATA_1	-
SPI2_CSB	GPIO35	SPI2_CSB	UART1_RTS	UART2_RTS	-	EMMC_DATA_0	-
SPI2_HOLD	GPIO36	SPI2_HOLD	-	-	-	EMMC_CMD	-
SPI2_WP	GPIO37	SPI2_WP	-	-	-	EMMC_CK	-
EMMC_RSTB	GPIO38	EMMC_RSTB	-	-	-	-	-
EMMC_DSL	GPIO39	EMMC_DSL	-	-	-	-	-
EMMC_CK	GPIO40	EMMC_CK	-	-	-	-	-
EMMC_CMD	GPIO41	EMMC_CMD	-	-	-	-	-
EMMC_DATA_7	GPIO42	EMMC_DATA_7	-	-	-	-	-
EMMC_DATA_6	GPIO43	EMMC_DATA_6	-	-	-	-	-
EMMC_DATA_5	GPIO44	EMMC_DATA_5	-	-	-	-	-
EMMC_DATA_4	GPIO45	EMMC_DATA_4	-	-	-	-	-
EMMC_DATA_3	GPIO46	EMMC_DATA_3	-	-	-	-	-
EMMC_DATA_2	GPIO47	EMMC_DATA_2	-	-	-	-	-
EMMC_DATA_1	GPIO48	EMMC_DATA_1	-	-	-	-	-
EMMC_DATA_0	GPIO49	EMMC_DATA_0	-	-	-	-	-
PCM_FS_I2S_LRCK	GPIO50	PCM_FS_I2S_LRCK	UART2_RXD	-	-	-	-
PCM_CLK_I2S_BCLK	GPIO51	PCM_CLK_I2S_BCLK	UART2_TXD	-	-	-	-
PCM_DRX_I2S_DIN	GPIO52	PCM_DRX_I2S_DIN	UART2_CTS	-	-	-	-
PCM_DTX_I2S_DOUT	GPIO53	PCM_DTX_I2S_DOUT	UART2_RTS	-	-	-	-
PCM_MCK_I2S_MCLK	GPIO54	PCM_MCK_I2S_MCLK	-	-	-	-	-
UART0_RXD	GPIO55	UART0_RXD	-	-	-	-	-
UART0_TXD	GPIO56	UART0_TXD	-	-	-	-	-
PWMD0	GPIO57	PWMD0	-	-	-	-	-
JTAG_JTDI	GPIO58	JTAG_JTDI	UART1_RXD	UART2_RXD	-	PWMD2	GBE_LED1_P0
JTAG_JTDO	GPIO59	JTAG_JTDO	UART1_TXD	UART2_TXD	-	PWMD3	GBE_LED1_P1
JTAG_JTMS	GPIO60	JTAG_JTMS	UART1_CTS	UART2_CTS	-	PWMD4	GBE_LED1_P2
JTAG_JTCLK	GPIO61	JTAG_JTCLK	UART1_RTS	UART2_RTS	-	PWMD5	GBE_LED1_P3
JTAG_JTRST_N	GPIO62	JTAG_JTRST_N	-	-	-	PWMD6	2P5G_LED1_P0
USB_DRV_VBUS_P1	GPIO63	USB_DRV_VBUS_P1	-	-	-	-	-
LED_A	GPIO64	GBE_LED0_P0	-	-	-	-	-
LED_B	GPIO65	GBE_LED0_P1	-	-	-	-	-
LED_C	GPIO66	GBE_LED0_P2	-	-	-	-	-
LED_D	GPIO67	GBE_LED0_P3	-	-	-	-	-
LED_E	GPIO68	2P5G_LED0_P0	-	-	-	-	-
GPIO_B	GPIO69	SMI_0_MDC	I2C_1_SCL	PWMD6	I2C_2_SCL	-	-
GPIO_C	GPIO70	SMI_0_MDIO	I2C_1_SDA	PWMD7	I2C_2_SDA	-	-
I2C_2_SCL	GPIO71	I2C_2_SCL	-	-	-	-	-
I2C_2_SDA	GPIO72	I2C_2_SDA	-	-	-	-	-
PCIE30_2L_1_PRESET_N	GPIO73	PCIE30_2L_1_PRESET_N	-	-	-	-	-
PCIE30_1L_0_PRESET_N	GPIO74	PCIE30_1L_0_PRESET_N	-	-	-	-	-
PCIE30_2L_1_WAKE_N	GPIO75	PCIE30_2L_1_WAKE_N	-	-	-	-	-
PCIE30_2L_1_CLKREQ_N	GPIO76	PCIE30_2L_1_CLKREQ_N	-	-	-	-	-
PCIE30_1L_0_WAKE_N	GPIO77	PCIE30_1L_0_WAKE_N	-	-	-	-	-
PCIE30_1L_0_CLKREQ_N	GPIO78	PCIE30_1L_0_CLKREQ_N	-	-	-	-	-

Ball Name	Aux Func.0	Aux Func.1	Aux Func.2	Aux Func.3	Aux Func.4	Aux Func.5	Aux Func.6
USB_DRV_VBUS_P0	GPIO79	USB_DRV_VBUS_P0	-	-	-	-	-
UART1_RXD	GPIO80	UART1_RXD	PWMD2	-	-	-	-
UART1_TXD	GPIO81	UART1_TXD	PWMD3	-	-	-	-
UART1_CTS	GPIO82	UART1_CTS	PWMD4	-	-	-	-
UART1_RTS	GPIO83	UART1_RTS	PWMD5	-	-	-	-

4 Electrical Characteristics

4.1 Absolute Maximum Ratings

Table 4-1. Absolute Maximum Ratings

Ball Name	Description	Min	Max	Unit
AVDD09_CKM_PE4 AVDD09_PCIE_1L_0 AVDD09_PCIE_1L_1 AVDD09_PCIE_2L_0 AVDD09_PCIE_2L_1 AVDD09_SSUSB AVDD09_USXGMII_0 AVDD09_USXGMII_1 AVDD09_USXGMII_PLL	Analog power	-0.3	1.08	V
AVDD12_CKSQ AVDD12_PCIE_1L_0 AVDD12_PCIE_1L_1 AVDD12_PCIE_2L_0 AVDD12_PCIE_2L_1 AVDD12_USXGMII_0 AVDD12_USXGMII_1 AVDD12_USXGMII_PLL	Analog power	-0.3	1.44	V
AVDD15_POR	Analog power	-0.3	1.8	V
AVDD18_2P5GBT AVDD18_CKM_PE4 AVDD18_CKSQ AVDD18_COM_P0 AVDD18_COM_P1 AVDD18_COM_P2 AVDD18_COM_P3 AVDD18_DDR AVDD18_DDR AVDD18_DDR AVDD18_LD_P0 AVDD18_LD_P1 AVDD18_LD_P2 AVDD18_LD_P3 AVDD18_MCU AVDD18_PCIE_1L_0 AVDD18_PCIE_1L_1 AVDD18_PCIE_2L_0 AVDD18_PCIE_2L_1 AVDD18_PLLGP AVDD18_POR AVDD18_SSUSB AVDD18_USB_P0 AVDD18_USB_P1 AVDD18_USXGMII_0 AVDD18_USXGMII_1 AVDD18_VQPS AVDD18_XTAL	Analog power	-0.3	2.16	V

Ball Name	Description	Min	Max	Unit
AVDD33_2P5GBT0 AVDD33_2P5GBT1 AVDD33_LD_P0 AVDD33_LD_P1 AVDD33_LD_P2 AVDD33_LD_P3 AVDD33_USB_P0 AVDD33_USB_P1	Analog power	-0.3	3.93	V
DVDD_CORE	Digital power	-0.3	1.02	V
DVDD_PROC	Digital power	-0.3	1.15	V
DVDD_PROC_SRAM	Digital power	-0.3	1.15	V
DVDD18G_GPIO_LB DVDD18G_GPIO_RB_3 DVDD18G_GPIO_TL DVDD18G_GPIO_TR	Digital power	-0.3	2.2	V
AVDDQ_CLK_EMI AVDDQ_EMI	Analog power	-0.3	1.8	V
DVDD33G_GPIO_LB DVDD33G_GPIO_RB_1 DVDD33G_GPIO_RB_2 DVDD33G_GPIO_RB_3 DVDD33G_GPIO_TL DVDD33G_GPIO_TR DVDD33G_GPIO_TR_3	IO power	-0.3	3.93	V

4.2 Recommended Operating Conditions

Table 4-2. Recommended Operating Conditions for Power Supply

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
L31	AVDD09_CKM_PE4	Analog power	0.855	0.9	0.945	V
Y22	AVDD09_PCIE_1L_0	Analog power	0.855	0.9	0.945	V
Y21	AVDD09_PCIE_1L_1	Analog power	0.855	0.9	0.945	V
T21	AVDD09_PCIE_2L_0	Analog power	0.855	0.9	0.945	V
T20	AVDD09_PCIE_2L_1	Analog power	0.855	0.9	0.945	V
P9	AVDD09_SSUSB	Analog power	0.855	0.9	0.945	V
N20	AVDD09_USXGMII_0	Analog power	0.855	0.9	0.945	V
M20	AVDD09_USXGMII_1	Analog power	0.855	0.9	0.945	V
B30	AVDD09_USXGMII_PLL	Analog power	0.855	0.9	0.945	V
G31	AVDD12_CKSQ	Analog power	1.14	1.2	1.26	V
AA22	AVDD12_PCIE_1L_0	Analog power	1.14	1.2	1.26	V
W22	AVDD12_PCIE_1L_1	Analog power	1.14	1.2	1.26	V
U20	AVDD12_PCIE_2L_0	Analog power	1.14	1.2	1.26	V
V21	AVDD12_PCIE_2L_1	Analog power	1.14	1.2	1.26	V
F26	AVDD12_USXGMII_0	Analog power	1.14	1.2	1.26	V
F24	AVDD12_USXGMII_1	Analog power	1.14	1.2	1.26	V
C29	AVDD12_USXGMII_PLL	Analog power	1.14	1.2	1.26	V
A19	AVDD15_POR	Analog power	1.08	1.5	1.65	V
K6	AVDD18_2P5GBT	Analog power	1.71	1.8	1.89	V
L30	AVDD18_CKM_PE4	Analog power	1.71	1.8	1.89	V
G30	AVDD18_CKSQ	Analog power	1.71	1.8	1.89	V

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
H16	AVDD18_COM_P0	Analog power	1.71	1.8	1.89	V
H14	AVDD18_COM_P1	Analog power	1.71	1.8	1.89	V
H13	AVDD18_COM_P2	Analog power	1.71	1.8	1.89	V
H11	AVDD18_COM_P3	Analog power	1.71	1.8	1.89	V
AA12	AVDD18_DDR	Analog power	1.71	1.8	1.89	V
AA16	AVDD18_DDR	Analog power	1.71	1.8	1.89	V
AA21	AVDD18_DDR	Analog power	1.71	1.8	1.89	V
G16	AVDD18_LD_P0	Analog power	1.71	1.8	1.89	V
G14	AVDD18_LD_P1	Analog power	1.71	1.8	1.89	V
G13	AVDD18_LD_P2	Analog power	1.71	1.8	1.89	V
G11	AVDD18_LD_P3	Analog power	1.71	1.8	1.89	V
V14	AVDD18_MCU	Analog power	1.71	1.8	1.89	V
W21	AVDD18_PCIE_1L_0	Analog power	1.71	1.8	1.89	V
V22	AVDD18_PCIE_1L_1	Analog power	1.71	1.8	1.89	V
U21	AVDD18_PCIE_2L_0	Analog power	1.71	1.8	1.89	V
U22	AVDD18_PCIE_2L_1	Analog power	1.71	1.8	1.89	V
W12	AVDD18_PLLGP	Analog power	1.71	1.8	1.89	V
B19	AVDD18_POR	Analog power	1.71	1.8	1.89	V
P8	AVDD18_SSUSB	Analog power	1.71	1.8	1.89	V
AC28	AVDD18_USB_P0	Analog power	1.71	1.8	1.89	V
P7	AVDD18_USB_P1	Analog power	1.71	1.8	1.89	V
G26	AVDD18_USXGMII_0	Analog power	1.71	1.8	1.89	V
G24	AVDD18_USXGMII_1	Analog power	1.71	1.8	1.89	V
D30	AVDD18_VQPS	Analog power	1.71	1.8	1.89	V
E30	AVDD18_XTAL	Analog power	1.71	1.8	1.89	V
J7	AVDD33_2P5GBT0	Analog power	3.135	3.3	3.465	V
J6	AVDD33_2P5GBT1	Analog power	3.135	3.3	3.465	V
F16	AVDD33_LD_P0	Analog power	3.135	3.3	3.465	V
F14	AVDD33_LD_P1	Analog power	3.135	3.3	3.465	V
F13	AVDD33_LD_P2	Analog power	3.135	3.3	3.465	V
F11	AVDD33_LD_P3	Analog power	3.135	3.3	3.465	V
AC29	AVDD33_USB_P0	Analog power	3.135	3.3	3.465	V
P6	AVDD33_USB_P1	Analog power	3.135	3.3	3.465	V
M11	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
M13	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
M15	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
M17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
M19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
N11	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
N13	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
N15	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
N17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
N19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
P11	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
P13	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
P15	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
P17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
P19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
R17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
R19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
T17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
T19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
U17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
V17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
V19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
W11	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
W13	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
W15	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
W17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
W19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
Y11	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
Y13	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
Y15	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
Y17	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
Y19	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
Y20	DVDD_CORE	Digital power	0.8075	0.85	0.8925	V
R11	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
R13	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
R15	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
T11	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
T13	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
U11	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
U13	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
U15	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
V11	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
V13	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
V15	DVDD_PROC	Digital power	0.8075	0.85	1.04	V
T15	DVDD_PROC_SRAM	Digital power	0.8075	0.85	1.04	V
AG31	DVDD18G_GPIO_LB	Digital power	1.71	1.8	1.89	V
AH2	DVDD18G_GPIO_RB_3	Digital power	1.71	1.8	1.89	V
B20	DVDD18G_GPIO_TL	Digital power	1.71	1.8	1.89	V
F1	DVDD18G_GPIO_TR	Digital power	1.71	1.8	1.89	V

Table 4-3. Recommended Operating Conditions – For DDR3

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
AB16	AVDDQ_CLK_EMI	Analog power	1.425	1.5	1.575	V
AB17	AVDDQ_CLK_EMI	Analog power	1.425	1.5	1.575	V
AA13	AVDDQ_EMI	Analog power	1.425	1.5	1.575	V
AA14	AVDDQ_EMI	Analog power	1.425	1.5	1.575	V
AA15	AVDDQ_EMI	Analog power	1.425	1.5	1.575	V
AA18	AVDDQ_EMI	Analog power	1.425	1.5	1.575	V
AA19	AVDDQ_EMI	Analog power	1.425	1.5	1.575	V
AA20	AVDDQ_EMI	Analog power	1.425	1.5	1.575	V

Table 4-4. Recommended Operating Conditions – For DDR4

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
AB16	AVDDQ_CLK_EMI	Analog power	1.14	1.2	1.26	V
AB17	AVDDQ_CLK_EMI	Analog power	1.14	1.2	1.26	V
AA13	AVDDQ_EMI	Analog power	1.14	1.2	1.26	V
AA14	AVDDQ_EMI	Analog power	1.14	1.2	1.26	V
AA15	AVDDQ_EMI	Analog power	1.14	1.2	1.26	V
AA18	AVDDQ_EMI	Analog power	1.14	1.2	1.26	V
AA19	AVDDQ_EMI	Analog power	1.14	1.2	1.26	V
AA20	AVDDQ_EMI	Analog power	1.14	1.2	1.26	V

Table 4-5. Recommended Operating Conditions – For IO Power Working at Typical 3.3V

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
AJ31	DVDD33G_GPIO_LB	IO power	3.135	3.3	3.465	V
AD2	DVDD33G_GPIO_RB_1	IO power	3.135	3.3	3.465	V
Y1	DVDD33G_GPIO_RB_2	IO power	3.135	3.3	3.465	V
AD1	DVDD33G_GPIO_RB_3	IO power	3.135	3.3	3.465	V
A21	DVDD33G_GPIO_TL	IO power	3.135	3.3	3.465	V
F2	DVDD33G_GPIO_TR	IO power	3.135	3.3	3.465	V
E2	DVDD33G_GPIO_TR_3	IO power	3.135	3.3	3.465	V

Table 4-6. Recommended Operating Conditions – For IO Power Working at Typical 1.8V

Ball Location	Ball Name	Description	Min	Typ	Max	Unit
AJ31	DVDD33G_GPIO_LB	IO power	1.71	1.8	1.89	V
AD2	DVDD33G_GPIO_RB_1	IO power	1.71	1.8	1.89	V
Y1	DVDD33G_GPIO_RB_2	IO power	1.71	1.8	1.89	V
AD1	DVDD33G_GPIO_RB_3	IO power	1.71	1.8	1.89	V
A21	DVDD33G_GPIO_TL	IO power	1.71	1.8	1.89	V
F2	DVDD33G_GPIO_TR	IO power	1.71	1.8	1.89	V
E2	DVDD33G_GPIO_TR_3	IO power	1.71	1.8	1.89	V

4.3 Thermal Characteristics

Thermal characteristics when stationary without an external heat sink in an air-conditioned environment.

Table 4-7. MT7988A Thermal Characteristics

Symbol	Description	Performance	
		Typ	Unit
T_J	Maximum junction temperature (plastic package)	125	°C
θ_{JA}	Junction to ambient temperature thermal resistance[1] for JEDEC 4L PCB	10.3	°C/W
θ_{JC}	Junction to case temperature thermal resistance	3.43	°C/W
θ_{JB}	Junction to case board thermal resistance	5.83	°C/W
ψ_{Jt}	Junction to the package thermal resistance for JEDEC 4L PCB	2.19	°C/W

Note: JEDEC 51-9 system FR4 PCB size: 101.5 x 114.5 mm (4” x 4.5”)

4.4 DC Electrical Specifications

4.4.1 SPI/PCM/I2S/GPIO DC Electrical Characteristics

Table 4-8. SPI/PCM/I2S Electrical Characteristics (VDDIO: Pin IO Power = 1.8V)

Parameter	Description	Min	Typ	Max	Unit
VIH	Input logic low voltage	1.27	-	VDDIO + 0.3	V
VIL	Input logic high voltage	-0.3	-	0.58	V
VOH	DC output logic low voltage	1.4	-	VDDIO + 0.3	V
VOL	DC output logic high voltage	-0.3	-	0.45	V

Table 4-9. SPI/PCM/I2S DC Electrical Characteristics (VDDIO: Pin IO Power = 3.3V)

Parameter	Description	Min	Typ	Max	Unit
VIH	Input logic low voltage	0.625*VDDIO	-	VDDIO + 0.3	V
VIL	Input logic high voltage	-0.3	-	0.25*VDDIO	V
VOH	DC output logic low voltage	0.75*VDDIO	-	VDDIO + 0.3	V
VOL	DC output logic high voltage	-0.3	-	0.125*DVDD18_IOBM	V

4.4.2 MSDC DC Electrical Characteristics

Table 4-10. MSDC DC Electrical Characteristics (VDDIO = 1.8V)

Parameter	Description	Min	Typ	Max	Unit
VIH	Input logic low voltage	0.7* VDDIO	-	VDDIO + 0.3	V
VIL	Input logic high voltage	-0.3	-	0.3* VDDIO	V
VOH	DC output logic low voltage	0.9* VDDIO	-	VDDIO + 0.3	V
VOL	DC output logic high voltage	-0.3	-	0.3* VDDIO	V

Table 4-11. MSDC DC Electrical Characteristics (VDDIO = 3.3V)

Parameter	Description	Min	Typ	Max	Unit
VIH	Input logic low voltage	0.625* VDDIO	-	VDDIO + 0.3	V
VIL	Input logic high voltage	-0.3	-	0.25* VDDIO	V
VOH	DC output logic low voltage	0.75* VDDIO	-	VDDIO + 0.3	V
VOL	DC output logic high voltage	-0.3	-	0.125* VDDIO	V

4.4.3 Open-drain I2C/I3C DC Electrical Characteristics

Table 4-12. I2C/I3C DC Electrical Characteristics (VDDIO: Pin IO Power)

Parameter	Description	Min	Typ	Max	Unit
VIH	Input logic low voltage	0.7*VDDIO	-	5.5	V
VIL	Input logic high voltage	-0.3	-	0.3*VDDIO	V
VOL	IO output I_{ol}= -4mA	-0.3	-	0.4	V

4.5 AC Electrical Specifications

4.5.1 UART Interface

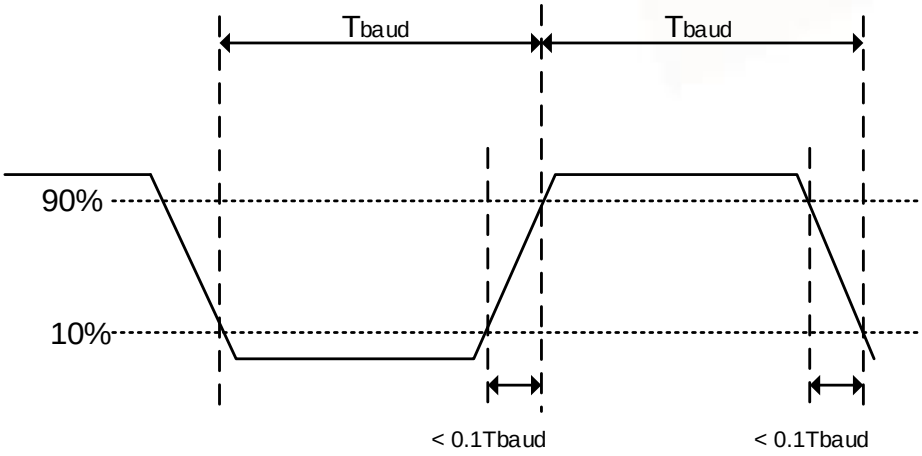


Figure 4-1. UART Timing

4.5.2 Generic SPI

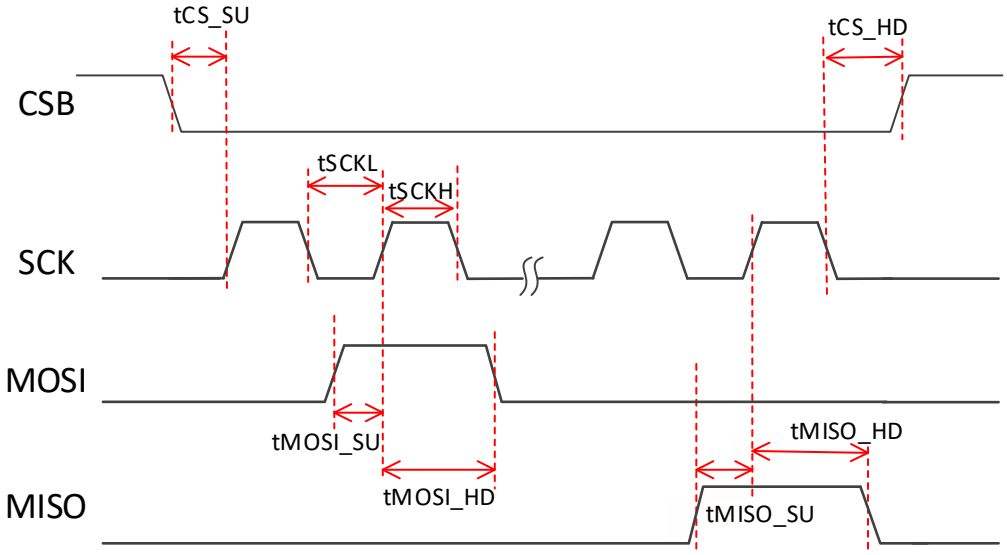


Figure 4-2. SPI Timing

Table 4-13. SPI Electrical Specifications

Symbol	Description	Performance			Unit
		Min	Typ	Max	
fSCK	SPI master SCK clock frequency	-	-	52	MHz
tMOSI_SU	MOSI to SCK rising setup time	6	-	-	ns
tMOSI_HD	SCK rising to MOSI hold time	6	-	-	ns
tSCKL	SCK low pulse	7.2	-	-	ns
tSCKH	SCK high pulse	7.2	-	-	ns
tCSB_SU (1)	CSB falling to SCK rising setup time	1.8	-	-	ns
tCSB_HD (1))	SCK falling to CSB rising hold time	1.8	-	-	ns
tMISO_SU (2)	MISO to SCK rising setup time requirement	0	-	-	ns
tMISO_HD (3)	SCK rising to MISO hold time requirement	0	-	-	ns

Note:

(1) In CS GPIO mode, SPI_CS is handled by software. Software should pull down SPI_CS pin before SPI starts transferring and pull up SPI_CS pin when SPI completes the transaction. Based on the sequence above, the minimum specification of tCSB_SU and tCSB_HD time cannot be satisfied.

(2) To achieve the max frequency of SCK, the internal sample clock delay of SPI master should be adjusted.

(3) MISO data valid time should be one cycle of fSCK.

(4) For dual mode or quad mode, all the output data pins can refer to the MOSI timing parameters, and all the input data pins can refer to the MISO timing parameters.

4.5.3 SPI NOR Flash Interface

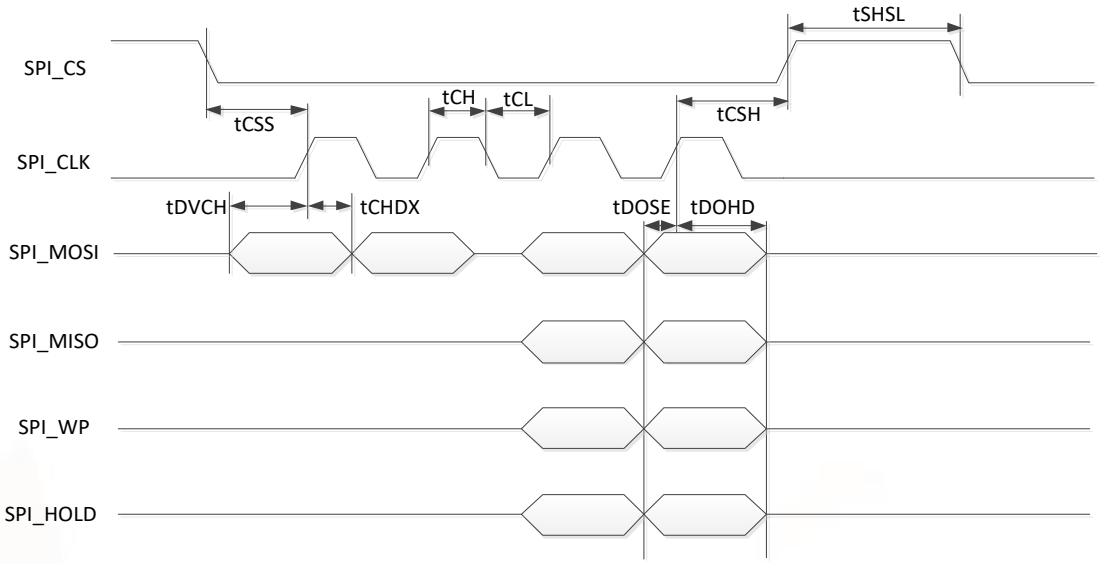


Figure 4-3. SPI NOR Interface Timing

Table 4-14. SPI NOR Interface Diagram Key

Symbol	Description	Min	Max	Unit
fC	Clock frequency	-	50	MHz
tCH	Clock high time (relative to CK)	6.1	6.8	ns
tCL	Clock low time (relative to CK)	6.3	6.9	ns
tSHSL	CS deselect time	88	-	ns
tCSS	CS active setup time	42	-	ns
tCSH	CS active hold time	67	-	ns
tDVCH	DI setup time	5.7	-	ns
tCHDX	DI hold time	6.3	-	ns
tDOSE	DO setup time	0	-	ns
tDOHD	DO hold time	3	-	ns

4.5.4 SPI NAND Flash Interface

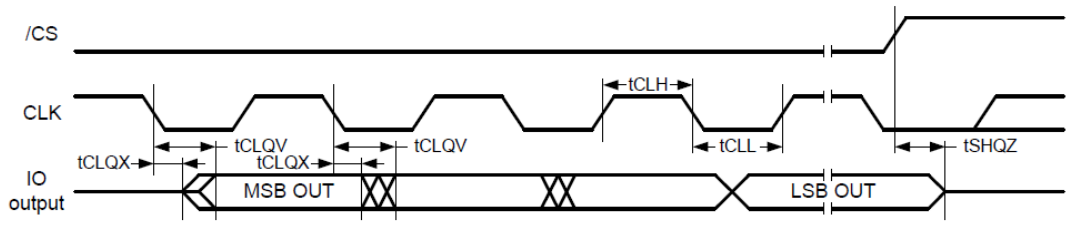


Figure 4-4. SPI NAND Serial Output Timing

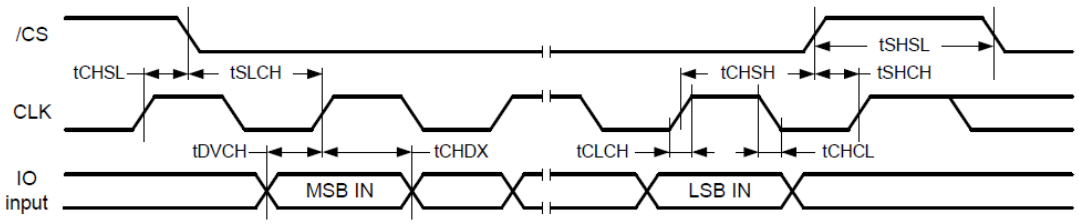


Figure 4-5. SPI NAND Serial Input Timing

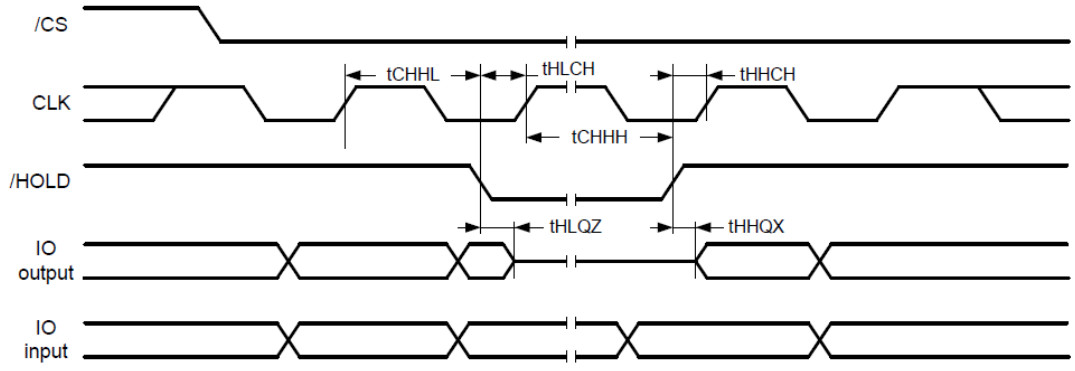


Figure 4-6. SPI NAND /HOLD Timing

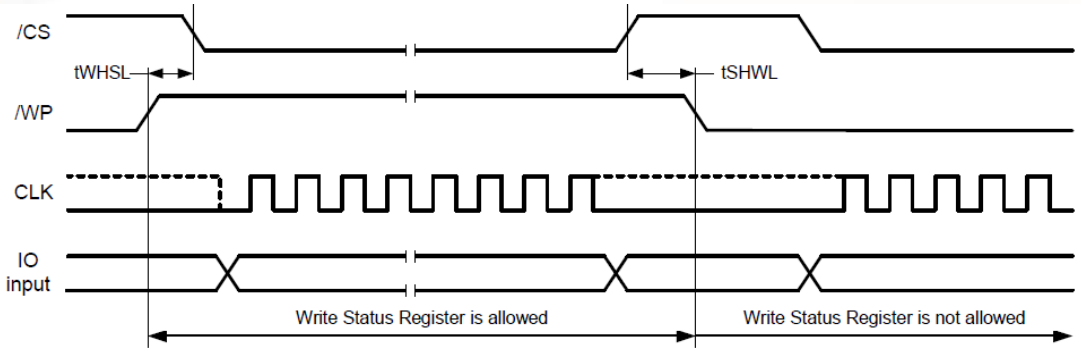


Figure 4-7. SPI NAND/WP Timing

Table 4-15. SPI NAND Interface Diagram Key

Symbol	Description	Min	Max	Unit
tCLH, tCLL,	Clock high, low time for all instructions	4	-	ns
tCLCH	Clock rise time peak to peak	0.1	-	V/ns
tCHCL	Clock fall time peak to peak	0.1	-	V/ns
tSLCH	/CS active setup time relative to CLK	5	-	ns
tCLCH	/CS not active hold time relative to CLK	5	-	ns
tDVCH	Data in setup time	2	-	ns
tCHDX	Data in hold time	3	-	ns
tCHSH	/CS active hold time relative to CLK	3	-	ns
tSHCH	/CS not active setup time relative to CLK	3	-	ns
tSHSL1	/CS deselect time (for array read → array read)	10	-	ns
tSHSL2	/CS deselect time (for erase, program or read status registers → read status registers)	50	-	ns
tSHQZ	Output disable time	-	7	ns
tCLQV	Clock low to output valid	-	7	ns
tCLQX	Output hold time	2	-	ns
tHLCH	/HOLD active setup time relative to CLK	5	-	ns
tCHHH	/HOLD active hold time relative to CLK	5	-	ns
tHHCH	/HOLD not active setup time relative to CLK	5	-	ns
tCHHL	/HOLD not active hold time relative to CLK	5	-	ns
tHHQX	/HOLD to output low-Z	-	7	ns
tHLQZ	/HOLD to output high-Z	-	12	ns
tWHSL	Write protect setup time before /CS cow	20	-	ns
tSHWL	Write protect hold time after /CS high	100	-	ns
tW	Status register write time	-	50	ns
tRST	/CS high to next instruction after reset during page data read/program execute/block erase	-	5/10/500	ns
tRD1	Read page data time (ECC disabled)	-	25	us
tRD2	Read page data time (ECC enabled)	-	60	us

4.6 Power-On Sequence

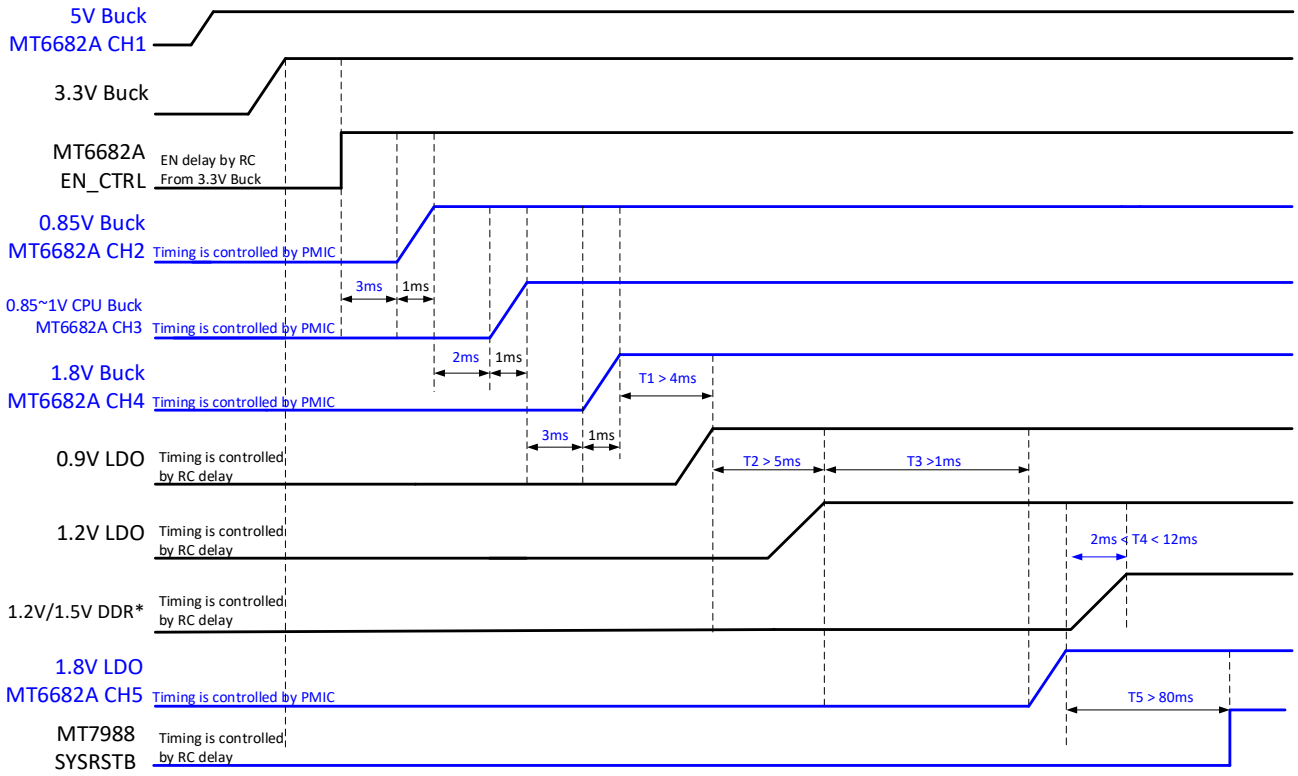


Figure 4-8. Power-On Sequence

Note:

(1) DDR3: 1.5V, DDR4: 1.2V

(2) DDR4: VPP (2.5V) must ramp up at the same time as or earlier than VDD (1.2V). VPP (2.5V) must be equal to or higher than VDD (1.2V) at all times.

(3) MT6682A CH2, 3, 4 and 5 timing values are only for reference. There is no need to measure them.

Table 4-16. Power-On Sequence Symbol Definition

Symbol	Description	Min	Max	Unit
T1	MT6682A buck 1.8V to ext. LDO 0.9V	4	-	ms
T2	ext. LDO 0.9V to ext. LDO 1.2V	5	-	ms
T3	ext. LDO 1.2V to MT6682A LDO 1.8V	1	-	ms
T4	MT6682A LDO 1.8V to ext. LDO 1.2V (DDR4) or LDO 1.5V (DDR3)	2	12	ms
T5	MT6682A LDO 1.8V to SYSRSTB	80	-	ms

5 MCU and Bus Fabric

5.1 Memory Map

Table 5-1. Memory Map

Start Address	End Address	Size	Device
0x0000_0000	0x001F_FFFF	2MB	ROM & SYSRAM
0x0F00_0000	0x0FFF_FFFF	16MB	ETH2P5G_S
0x1000_8000	0x1000_8FFF	4KB	APXGPT
0x1000_A000	0x1000_AFFF	4KB	SEJ_APB_S
0x1000_B000	0x1000_BFFF	4KB	AP_CIRQ_ENIT_APB
0x1001_7000	0x1001_7FFF	4KB	SYS_TIMER
0x1001_A000	0x1001_AFFF	4KB	SECURITY_AO_APB_S
0x1001_B000	0x1001_BFFF	4KB	CKSYS
0x1001_C000	0x1001_CFFF	4KB	CKSYS
0x1001_D000	0x1001_DFFF	4KB	CKSYS
0x1001_E000	0x1001_EFFF	4KB	TOP_CENTER_APB0
0x1001_F000	0x1001_FFFF	4KB	TOP_CENTER_APB1
0x1002_0000	0x1002_0FFF	4KB	TOP_CENTER_APB2
0x1002_1000	0x1002_1FFF	4KB	TOP_CENTER_APB3
0x1002_2000	0x1002_2FFF	4KB	TOP_CENTER_APB4
0x1004_8000	0x1004_8FFF	4KB	PWM
0x1006_0000	0x1006_7FFF	32KB	SGMII_SBUS0
0x1007_0000	0x1007_7FFF	32KB	SGMII_SBUS1
0x1008_0000	0x1008_0FFF	4KB	USXGMII_SBUS0
0x1008_1000	0x1008_1FFF	4KB	USXGMII_SBUS1
0x100E_0000	0x100E_FFFF	64KB	MCUSYS_CFGREG_APB
0x1020_4000	0x1020_4FFF	4KB	SYS_CIRQ
0x1020_F000	0x1020_FFFF	4KB	TRNG
0x1021_2000	0x1021_2FFF	4KB	CQ_DMA
0x1021_4000	0x1021_4FFF	4KB	SRAMROM APB
0x1021_7000	0x1021_7FFF	4KB	AP_DMA
0x1021_9000	0x1021_9FFF	4KB	EMI_APB
0x1021_C000	0x1021_CFFF	4KB	INFRACFG_MEM_APB
0x1022_6000	0x1022_6FFF	4KB	EMI_MPU_APB
0x1023_0000	0x1023_1FFF	8KB	DRAMC_CH0_TOP0
0x1023_2000	0x1023_3FFF	8KB	DRAMC_CH0_TOP1
0x1023_4000	0x1023_5FFF	8KB	DRAMC_CH0_TOP2
0x1023_6000	0x1023_7FFF	8KB	DRAMC_CH0_TOP3
0x1023_8000	0x1023_9FFF	8KB	DRAMC_CH0_TOP4
0x1023_A000	0x1023_BFFF	8KB	DRAMC_CH0_TOP5
0x1023_C000	0x1023_DFFF	8KB	DRAMC_CH0_TOP6
0x1024_0000	0x1024_1FFF	8KB	DRAMC_CH1_TOP0
0x1024_2000	0x1024_3FFF	8KB	DRAMC_CH1_TOP1
0x1024_4000	0x1024_5FFF	8KB	DRAMC_CH1_TOP2
0x1024_6000	0x1024_7FFF	8KB	DRAMC_CH1_TOP3
0x1024_8000	0x1024_9FFF	8KB	DRAMC_CH1_TOP4
0x1024_A000	0x1024_BFFF	8KB	DRAMC_CH1_TOP5
0x1024_C000	0x1024_DFFF	8KB	DRAMC_CH1_TOP6

Start Address	End Address	Size	Device
0x1054_0000	0x1054_7FFF	32KB	DRAMC_MD32
0x1100_0000	0x1100_00FF	256B	UART0
0x1100_0100	0x1100_01FF	256B	UART1
0x1100_0200	0x1100_02FF	256B	UART2
0x1100_1000	0x1100_1FFF	4KB	NFI
0x1100_2000	0x1100_2FFF	4KB	NFI_ECC
0x1100_3000	0x1100_3FFF	4KB	I2C0
0x1100_4000	0x1100_4FFF	4KB	I2C1
0x1100_5000	0x1100_5FFF	4KB	I2C2
0x1100_6000	0x1100_6FFF	4KB	I2C3
0x1100_7000	0x1100_7FFF	4KB	SPI0
0x1100_8000	0x1100_8FFF	4KB	SPI
0x1100_9000	0x1100_9FFF	4KB	SPI2
0x1100_A000	0x1100_AFFF	4KB	PTP_THERM_CTRL
0x1119_0000	0x1119_19FF	6.5KB	SSUSB_S
0x1119_1A00	0x1119_1BFF	512B	SSUSB_S-1
0x1119_1C00	0x1119_FFFF	57KB	SSUSB_S-2
0x1120_0000	0x1120_19FF	6.5KB	SSUSB_P1_S
0x1120_1A00	0x1120_1BFF	512B	SSUSB_P1_S-1
0x1120_1C00	0x1120_FFFF	57KB	SSUSB_P1_S-2
0x1121_0000	0x1121_8FFF	36KB	AUDIO_S
0x1121_9000	0x1121_93FF	1KB	AUDIO_S-1
0x1123_0000	0x1123_FFFF	64KB	MDSC0
0x1128_0000	0x1128_7FFF	32KB	PCIE core configuration Port2
0x1129_0000	0x1129_7FFF	32KB	PCIE core configuration Port3
0x1130_0000	0x1130_7FFF	32KB	PCIE core configuration Port0
0x1131_0000	0x1131_7FFF	32KB	PCIE core configuration Port1
0x11C0_0000	0x11C0_FFFF	64KB	IOCFG_RT (4KB)
0x11C1_0000	0x11C1_FFFF	64KB	IOCFG_TR (4KB)
0x11C2_0000	0x11C2_FFFF	64KB	RSVD
0x11C3_0000	0x11C3_FFFF	64KB	ETH2P5G_APB (64KB)
0x11C4_0000	0x11C4_FFFF	64KB	GDU (4KB)
0x11C5_0000	0x11C5_FFFF	64KB	USB_PHYD_1 (64KB)
0x11C6_0000	0x11C6_FFFF	64KB	RSVD
0x11C7_0000	0x11C7_FFFF	64KB	RSVD
0x11D0_0000	0x11D0_FFFF	64KB	IOCFG_BR (4KB)
0x11D1_0000	0x11D1_FFFF	64KB	TOP_MISC_REG (4KB)
0x11D2_0000	0x11D2_FFFF	64KB	IOCFG_RB (4KB)
0x11D3_0000	0x11D3_FFFF	64KB	RSVD
0x11D4_0000	0x11D4_FFFF	64KB	PGD(4KB)
0x11D5_0000	0x11D5_FFFF	64KB	RSVD
0x11D6_0000	0x11D6_FFFF	64KB	msdc_pad_macro (4KB)
0x11D7_0000	0x11D7_FFFF	64KB	RSVD
0x11E0_0000	0x11E0_FFFF	64KB	IOCFG_LB (4KB)
0x11E1_0000	0x11E1_FFFF	64KB	USB_PHYD_0 (64KB)
0x11E2_0000	0x11E2_FFFF	64KB	RSVD
0x11E3_0000	0x11E3_FFFF	64KB	RSVD
0x11E4_0000	0x11E4_FFFF	64KB	PCIE_PHYD_0 (64KB)
0x11E5_0000	0x11E5_FFFF	64KB	PCIE_PHYD_1 (64KB)
0x11E6_0000	0x11E6_FFFF	64KB	PCIE_PHYD_2 (64KB)

Start Address	End Address	Size	Device
0x11E7_0000	0x11E7_FFFF	64KB	PCIE_PHYD_3 (64KB)
0x11F0_0000	0x11F0_FFFF	64KB	IOCFG_TL (4KB)
0x11F1_0000	0x11F1_FFFF	64KB	CKM (4KB)
0x11F2_0000	0x11F2_FFFF	64KB	USXGMII_PHY0 (64KB)
0x11F3_0000	0x11F3_FFFF	64KB	USXGMII_PHY1 (64KB)
0x11F4_0000	0x11F4_FFFF	64KB	USXGMII_PLL (4Kb)
0x11F5_0000	0x11F5_FFFF	64KB	EFUSE_CTRL (64KB)
0x11F6_0000	0x11F6_FFFF	64KB	RSVD
0x11F7_0000	0x11F7_FFFF	64KB	RSVD
0x1500_0000	0x151D_FFFF	1.875MB	NETSYS
0x151E_0000	0x151F_FFFF	128KB	NETSYS-1
0x2000_0000	0x27FF_FFFF	256MB	PCIE2 data block (64-bit address)
0x2800_0000	0x2FFF_FFFF	256MB	PCIE3 data block (64-bit address)
0x3000_0000	0x37FF_FFFF	256MB	PCIE0 data block (64-bit address)
0x3800_0000	0x3FFF_FFFF	256MB	PCIE1 data block (64-bit address)

5.1.1 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

5.2 External Interrupt Controller

The External Interrupt Controller (EINTC) processes all off-chip interrupt sources and forwards interrupt request signals to AP MCU.

5.2.1 Features

EINTC supports up to 84 external interrupt signals and performs the following processes to the interrupt signals coming from external sources:

- Polarity inversion
- Edge or level trigger selection
- De-bounce with a configurable 32 kHz clock (optional)

Depending on register configurations, the external interrupt source is forwarded to the GIC (Generic Interrupt Controller) with different IRQ signals, eint_irq or eint_direct_irq. Then, EINTC generates wake-up events to AP MCU.

5.2.2 Block Diagram

Figure 5-1 shows the block diagram of the external interrupt controller in the MT7988A. Every functional block is controlled by the corresponding control registers defined in Section 5.2.3.

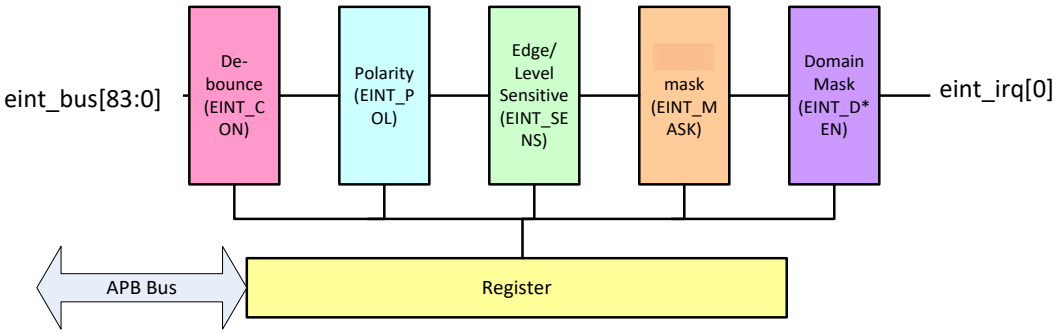


Figure 5-1. Block Diagram of Clock Management Unit and Sleep Controller.

Normally the external interrupt source goes through the de-bounce unit, which is driven by a 32 kHz clock and triggers the corresponding CPU with eint_irq. Therefore, the minimum latency from eint_bus to eint_irq is 30.52 µs. Since the latency introduced by the de-bounce module may be too long for some applications, EINTC provides an alternative path which bypasses the de-bounce module and directly triggers the interrupt signals, eint_direct_irq[3:0], to AP MCU.

5.2.3 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

5.3 System Interrupt Controller

For processors like Cortex-A53, which have embedded interrupt controllers (GIC), the MCUSYS needs to keep feeding clock and power to make the interrupt functional. However, due to power/leakage overhead introduced by higher clock ratio and deep submicron processes, reserving an always on (or frequently turned on) domain in MCUSYS has become power ineffective.

The system interrupt controller (SYS_CIRQ) is a low power interrupt controller designed to work outside MCUSYS as a second level interrupt controller. With SYS_CIRQ, MCUSYS can be completely turned off to improve system power performance without losing interrupts.

5.3.1 Features

SYS_CIRQ supports up to 240 interrupts which can configure the following attributes individually.

- Polarity inversion
- Edge or level trigger selection

The 240 interrupts feed through SYS_CIRQ and connect to GIC in MCUSYS. When SYS_CIRQ is enabled, the controller records the edge-sensitive interrupts and generates a pulse signal to CPU GIC when the flush command is executed.

5.3.2 Block Diagram

Figure 5-2 shows the system level block diagram of SYS_CIRQ.

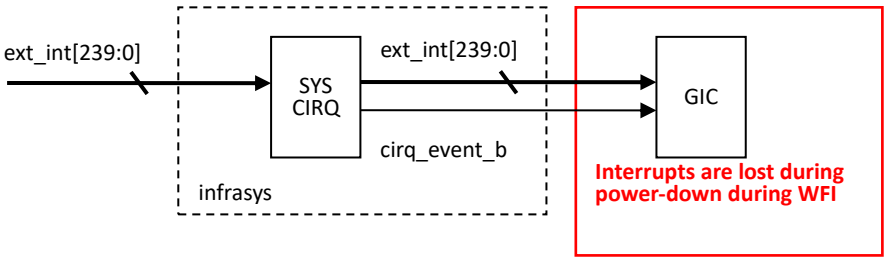


Figure 5-2. System Level Block Diagram of System Interrupt Controller

The SYS_CIRQ controller is integrated between MCUSYS and other interrupt sources as the second level interrupt controller. All interrupts are fed through SYS_CIRQ controller and then bypassed to MCUSYS. In normal mode (when MCUSYS GIC is active), SYS_CIRQ is disabled, and interrupts are directly issued to MCUSYS. The MT7988A does not have sleep mode, so MCUSYS is always active.

For products that support sleep mode, when MCUSYS enters the sleep mode, where GIC is powered down, SYS_CIRQ controller is enabled and monitors all edge-triggered interrupts. Note that only edge-triggered interrupts are lost in this scenario. When an edge-triggered interrupt is triggered, the interrupt is recorded in the SYS_CIRQ_STA register and can be restored to GIC by software context restore or the SYS_CIRQ flush function.

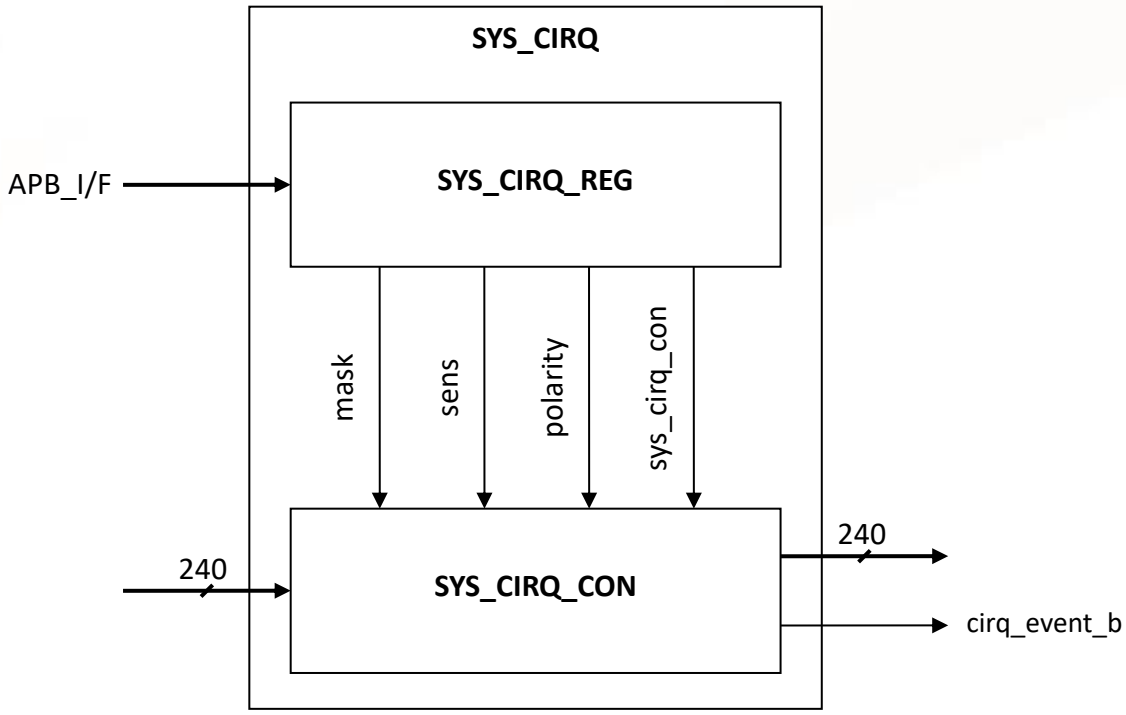


Figure 5-3. Block Diagram of System Interrupt Controller

Figure 5-3 is the architecture of SYS_CIRQ. SYS_CIRQ_REG stores the mask, sensitivity and polarity attributes of each interrupt signal and SYS_CIRQ_CON is used to mask and detect edge-triggered interrupts.

5.3.3 Programming Guide

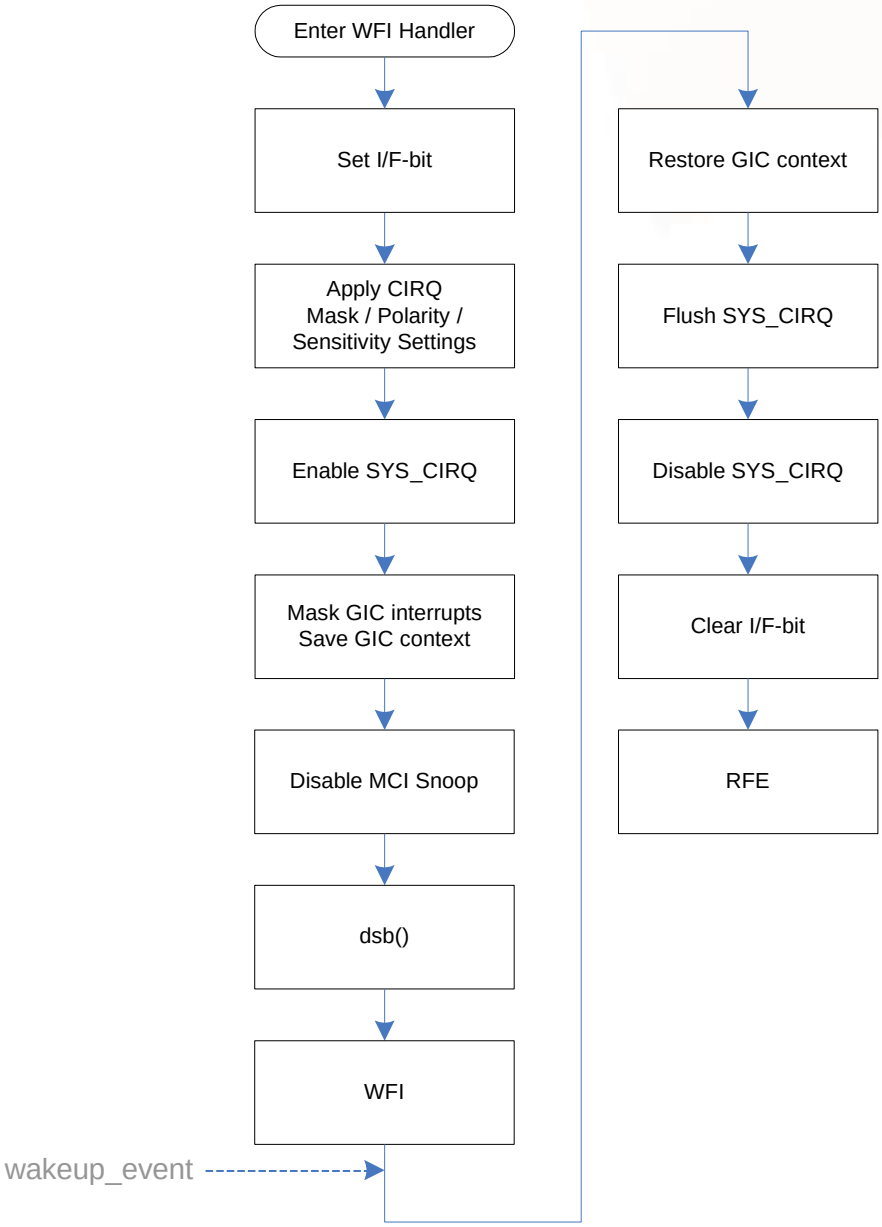


Figure 5-4. Software Programming Flowchart

Note:

(1) DSB: Data Synchronization Barrier (Arm instruction)

(2) WFI: Wait for Interrupt (Arm instruction)

(3) RFE: Return from an Exception (Arm instruction)

5.3.4 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

5.4 AP_DMA (Application Processor Direct Memory Access)

5.4.1 Introduction

The purpose of Application Processor Direct Memory Access (AP_DMA) is to perform data transfer between memory and peripherals.

5.4.2 Features

The features of AP_DMA are as follows:

- Up to 9 channels of simultaneous data transfers
- System bus (AXI) compliance
- Embedded 128-bit data FIFO in DMA channel
- Peripherals and channels:
  - I2C x 2
  - UART x 3 (TX/RX channels are separated, 6 channels in total)
  - DMA transmission realized by using signal and hand-shaking signal from a peripheral
- Source and destination configuration
  - Only one side (either source or destination) can be programmed; the other side can be selected as the specified peripheral
- Burst size and burst length
  - 8 bytes/1 beat
- TrustZone
  - The corresponding channel is treated as secure channel, if SEC_EN (AP_DMA_I2Cx_SEC_EN, AP_DMA_UARTx_TX_SEC_EN or AP_DMA_UARTx_RX_SEC_EN) [0] = 1.
  - If the channel is set as secure, it can issue secure requests, and its configuration registers can only be accessed via secure masters.
- Interrupt notification
  - FIFO data are over or under a certain threshold.
- Scheduling scheme
  - Round-Robin (RR). If many channels are triggered simultaneously, the triggering priority depends on channel number. The channel with a smaller number has a higher priority. For example, channel 0 > channel 1 > channel 2
- Cache coherency is not supported.

5.4.3 Block Diagram

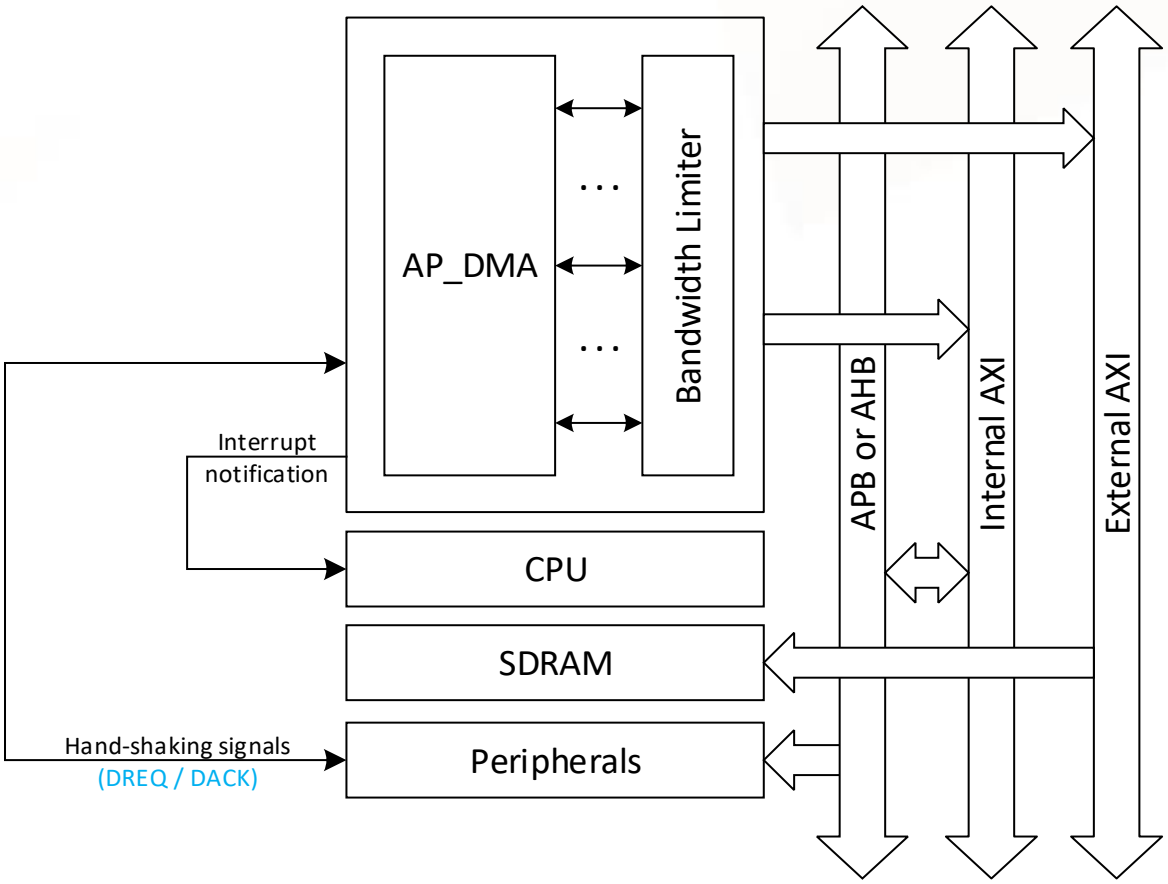


Figure 5-5. Block Diagram of AP_DMA

Table 5-2. AP_DMA Access Matrix

AP_DMA		Signal Description			
		SDRAM	SYSRAM	I2C/I3C	UART
Destination	SRAM	X	X	V	V
	SYSRAM	X	X	V	V
	I2C/I3C	V	V	X	X
	UART	V	V	X	X

Figure 5-5 is the basic block diagram of AP_DMA. There are a total of 9 channels in DMA. The external AXI is connected to the main AXI bus fabric to provide external memory access ability. The internal AXI is also connected to the peripheral AXI bus fabric to provide connectivity for related peripherals such as UART. The configuration space of DMA can be accessed via APB interface. All the control registers are divided into two groups: global registers and local registers. Common status and configurations are allocated in global registers. The local registers are based on channel-dependent configurations and exist in every DMA channel. For better throughput performance, a memory block is used as a buffer for every DMA and thus the data size is 8 bytes per request in an external AXI. But for an internal AXI, the data size is only 1 byte per request.

- UART virtual FIFO (VFF) DMA:

VFF is like a ring buffer and uses two address pointers, VFF_WPT and VFF_RPT, to control VFF condition.
Two symbols, VFF_VALID_SIZE and VFF_LEFT_SIZE, are defined to represent valid data and available space in VFF.

Refer to Figure 5-6 for the relation between UART VFF read and write pointers. When TX_VFF_WPT or RX_VFF_RPT is wrapped to a ring head again, invert TX_VFF_WPT_WRAP (AP_DMA_UART_x_TX_VFF_WPT) [16] for UART TX VFF DMA, or RX_VFF_RPT_WRAP (AP_DMA_UART_x_RX_VFF_RPT) [16] for UART RX VFF DMA.

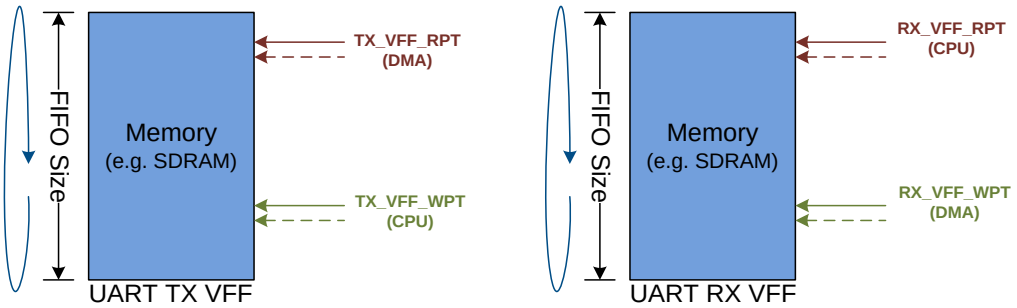


Figure 5-6. UART VFF Pointers Relation

- UART TX VFF DMA

When CPU writes n-byte data to VFF and updates “VFF_WPT” address pointer, DMA pops data to UART FIFO from VFF with handshaking signals and updates the “VFF_RPT” address pointer by hardware. If UART FIFO is available and VFF_VALID_SIZE is not zero, “VFF_VALID_SIZE” (VFF_WPT – VFF_RPT) is the size of data which is stored in VFF and has not yet been sent to UART FIFO. “VFF_LEFT_SIZE” (VFF_LEN – VFF_VALID_SIZE) is the size of available space in VFF. The unit is bytes.

There is a threshold (TX_VFF_THRE) to trigger the interrupt. If “VFF_LEFT_SIZE” ≥ “TX_VFF_THRE”, CPU can push data after receiving IRQ. In other words, if there are a lot of available space (VFF_LEFT_SIZE) in VFF, DMA triggers IRQ to inform CPU that it can push data.

For better bandwidth efficiency, every read request from TX DMA to TX VFF is 8 bytes. If “VFF_VALID_SIZE” is smaller than 8 bytes, DMA does not issue read request to TX VFF, even though DMA FIFO is empty. There might be an issue. If the total data size pushed to VFF by CPU is not 8-byte aligned, the last remaining data (< 8 bytes) may be kept in VFF forever. To deal with this case, software needs to set “TX FLUSH” register and then DMA issues read request to get the last remaining data in VFF.

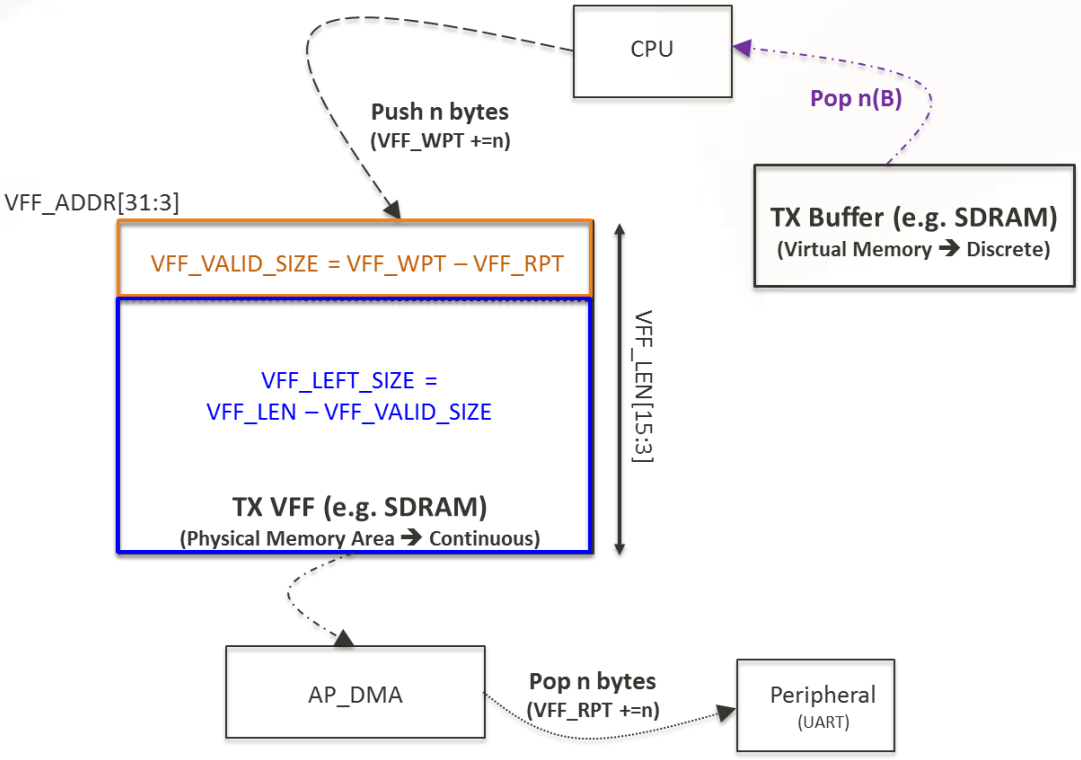


Figure 5-7. UART TX VFF Data Flow

- UART RX VFF DMA

In RX VFF DMA mode, data transfer direction is opposite to that in TX VFF DMA mode. VFF_WPT is updated by hardware after UART pushes data to VFF with handshaking signals. VFF_RPT is updated by CPU after CPU pops data from VFF if “VFF_VALID_SIZE” > 0.

There is also a threshold (RX_VFF_THRE) to trigger the interrupt. If “VFF_VALID_SIZE” ≥ “RX_VFF_THRE”, CPU can pop data after receiving IRQ. In other words, if there are many valid data stored in VFF, DMA triggers IRQ, which informs CPU that it can pop data. If CPU cannot pop data from VFF immediately, the data from UART may be lost. To avoid this phenomenon, DMA can inform UART by side-band signal, and CPU receives IRQ if “VFF_LEFT_SIZE” < “RX_FLOW_CTRL_THRE”. Software can respond, for example, by asserting RTS, if receiving IRQ.

Every write request from RX DMA to VFF is also 8 bytes. If the total data from UART is not 8-byte aligned, the last remaining data (< 8 bytes) is kept in DMA FIFO forever. There are two methods to solve this problem. The first method is set “RX FLUSH” register manually and DMA issues write request to pop the remaining data from DMA FIFO to VFF. The second method is UART issues a flush request to DMA by side-band signal when UART FIFO is empty and time-out happens. Note that data are not pushed towards UART FIFO in 4-byte baud-rate period. After flush request is done, the interrupt flag (FLAG1: AP_DMA_UART_x_RX_INT_FLAG[1]) is raised.

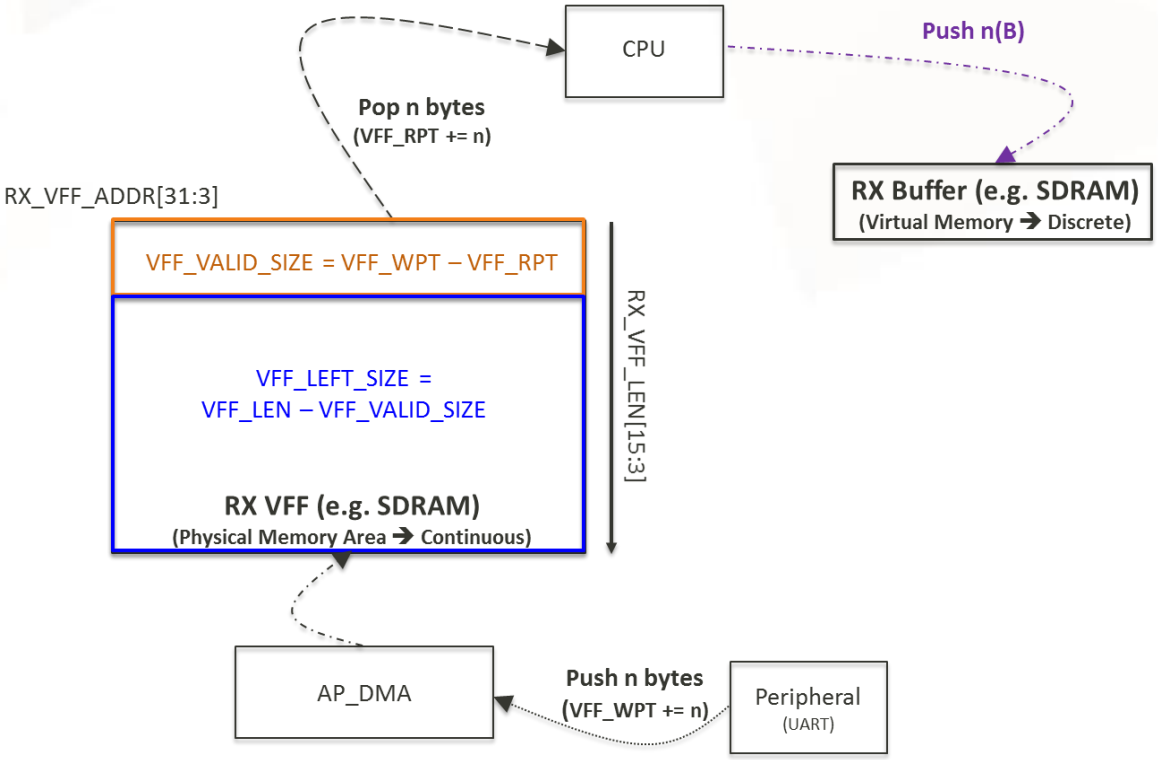


Figure 5-8. UART RX VFF Data Flow

- Peripheral (I2C) DMA

The behavior of I2C DMA is similar to that of CQ-DMA. The major difference is that the source or destination is I2C FIFO, not SDRAM. And there are handshaking signals, DREQ and DACK, between DMA and I2C. Because of handshaking signals, the DMA channels with corresponding I2C channels are fixed.
The DMA direction (DIR: AP_DMA_I2C_x_CON[0]) is controlled by I2C register, SLAVE_ADDR[0]. Every DMA channel can support both TX (I2C: write bit) and RX (I2C: read bit) directions. The transfer length and memory address register settings are also divided to TX and RX portions. Complying with I2C transfer format (refer to the corresponding I2C section), DMA supports all kinds of transfers, except the ‘‘direction change (write, and then read)’’ condition. To support “direction change” transfer, another hardware method needs to be applied. If I2C direction changes, the side-band signal (TX2RX) is sent to DMA when DMA_EN: CONTROL (I2Cx_Base + 0x10)[2] = 1, and DIR_CHANGE: CONTROL(I2Cx_Base + 0x10)[4] = 1.

Take Figure 5-9 as an example. Table 5-3 shows the settings of DMA and I2C, and the settings should be completed before I2C transaction (START: START(I2Cx_Base + 0x24)[0]).

Table 5-3. AP_DMA and I2C Sequential Random Read Settings

AP_DMA		I2C	
TX Transfer Length (AP_DMA_I2C_x_TX_LEN[15:0])	1	1	TX Transfer Length (TRANSFER_LEN[15:0])
RX Transfer Length (AP_DMA_I2C_x_RX_LEN[15:0])	N	N	RX Transfer Length (TRANSFER_LEN_AUX[15:0])
Direction (DIR: AP_DMA_I2C_x_CON[0])	TX (Controlled by I2C)	Write (0)	Direction (SLAVE_ADDR[0])
Enable DMA (EN: AP_DMA_I2C_x_EN[0])	1	1	Enable DMA (DMA_EN: CONTROL[2])
Direction Change	See Note.	1	Direction Change (DMA_EN: CONTROL[4])

Note:
DMA direction is controlled by I2C. When I2C changes DMA direction as Figure 5-9 shows, it sends side-band signal to DMA. When DMA direction (TX or RX) is changed by I2C, enable DMA again until "RX Transfer length" is received.

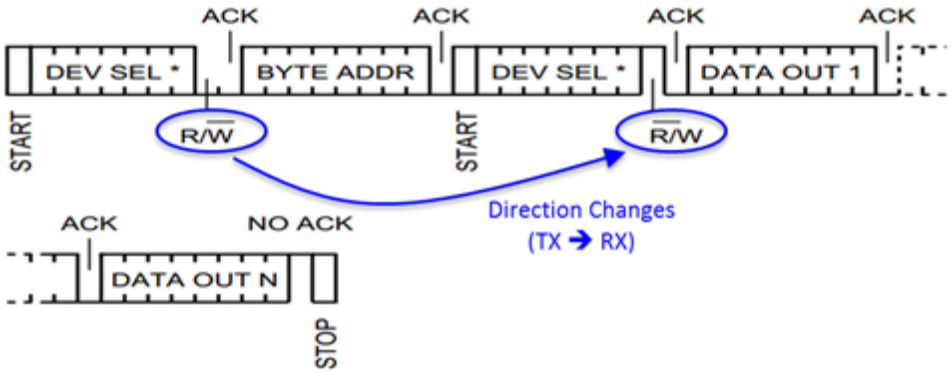


Figure 5-9. I2C Sequential Random Read Protocol

5.4.4 Warm Reset and Hard Reset

Warm reset and hard reset exist in DMA global control and each DMA engine. When warm reset is set, the engine is reset after the current transaction is finished. Therefore, the warm reset does not cause any bus hang. Conversely, when hard reset is set, the engine is reset immediately. Therefore, the bus might go down due to unfinished transaction.

- Mechanism of global warm reset

When the software needs to re-start all engines or re-clear all engines in DMA, set the global WARM_RST to 1 and wait for (poll) all global running status to be 0. Next, set WARM_RST back to 0 to finish the global warm reset.

- Mechanism of global hard reset

When the software needs to re-start all engines or re-clear all engines in DMA without waiting, set the global HARD_RST to 1 then back to 0 to finish the global hard reset. Note that this might break the bus protocol and cause system hang.

5.4.4.1 Pause and Resume

Pause and resume functions are available for all DMA channels. Table 5-4 shows the mechanism of I2C channel.

Table 5-4. Example of Pause and Resume

Step	Address	Register Name	Local Address	R/W	Value	Description
1	AP_DMA+0x08	AP_DMA_I2C_x_EN	AP_DMA_I2C_x_EN[0]	W	1'b1	Start DMA. (Necessary settings for the program; set EN (AP_DMA+0x08)[0] to 1.)
2	AP_DMA+0x10	AP_DMA_I2C_x_STOP	AP_DMA_I2C_x_STOP[1]	W	1'b1	Pause DMA. (Set PAUSE (AP_DMA+0x10)[1] to 1.)
3	AP_DMA+0x10	AP_DMA_I2C_x_STOP	AP_DMA_I2C_x_STOP[1]	W	1'b0	Resume DMA. (Set PAUSE (AP_DMA+0x10)[1] to 0.)
4	AP_DMA+0x08	AP_DMA_I2C_x_EN	AP_DMA_I2C_x_EN[0]	R	-	Wait for DMA to finish. (EN(AP_DMA+0x08)[0] becomes 0, and flag is set to 1.)

Note:
Start DMA (Necessary settings for the program. Set EN(AP_DMA+0x08)[0] to 1.)
Pause DMA (Set PAUSE (AP_DMA+0x10)[1] to 1.)
Resume DMA (Set PAUSE (AP_DMA+0x10)[1] to 0.)
The software can repeat steps 2 and 3 many times when DMA is running and monitor the idle bit to see if DMA is stopped.
DMA does not pause immediately. It waits for the last transaction to be finished.

5.4.4.2 Programming Sequence for Different Types of Channels

5.4.4.2.1 Peripheral DMA (Example: I2C)

Table 5-5 shows the reference setting for peripheral DMA.

Table 5-5. Reference Setting for Peripheral DMA (I2C)

Step	Address	Register Name	Local Address	R/W	Value	Description
1	AP_DMA+0x18	AP_DMA_I2C_x_CON	AP_DMA_I2C_x_CON[0]	W	1'b0 1'b1	Half duplex peripheral DMA direction 0: TX 1: RX
2	AP_DMA+0x24 (TX) AP_DMA+0x28 (RX)	AP_DMA_I2C_x_TX_LEN AP_DMA_I2C_x_RX_LEN	AP_DMA_I2C_x_TX_LEN[15:0] AP_DMA_I2C_x_RX_LEN[15:0]	W	Length	Peripheral DMA transfer length It can be any byte alignment.
3	AP_DMA+0x1C (TX) AP_DMA+0x54 (TX) AP_DMA+0x20 (RX) AP_DMA+0x58 (RX)	AP_DMA_I2C_x_TX_MEM_ADDR AP_DMA_I2C_x_TX_MEM_ADDR2 AP_DMA_I2C_x_RX_MEM_ADDR AP_DMA_I2C_x_RX_MEM_ADDR2	AP_DMA_I2C_x_TX_MEM_ADDR[31:0] AP_DMA_I2C_x_TX_MEM_ADDR2[3:0] AP_DMA_I2C_x_RX_MEM_ADDR[31:0] AP_DMA_I2C_x_RX_MEM_ADDR2[3:0]	W	Address	Peripheral DMA memory address It can be any byte alignment.
4	AP_DMA+0x04	AP_DMA_I2C_x_INT_EN	AP_DMA_I2C_x_INT_EN[0] AP_DMA_I2C_x_INT_EN[1] AP_DMA_I2C_x_INT_EN[2]	W	1'b1	Enables interrupt [0]: Enable interrupt for TX_FLAG [1]: Enable interrupt for RX_FLAG [2]: Enable interrupt for TX-to-RX

Step	Address	Register Name	Local Address	R/W	Value	Description
5	AP_DMA+0x08	AP_DMA_I2C_x_EN	P_DMA_I2C_x_EN[0]	W	1'b1	Enables peripheral DMA 0: Disable 1: Enable
6	Waiting for interrupt					
7	AP_DMA+0x00	AP_DMA_I2C_x_INT_FLGA	AP_DMA_I2C_x_INT_FLGA[0] AP_DMA_I2C_x_INT_FLGA[1] AP_DMA_I2C_x_INT_FLGA[2]	W	1'b1	Clears interrupt flag [0]: Clear interrupt flag for TX_FLAG [1]: Clear interrupt flag for RX_FLAG [2]: Clear interrupt flag for TX-to-RX

5.4.4.2.2 Virtual FIFO DMA TX (Example: UART_TX)

Table 5-6 shows the reference setting for virtual FIFO DMA TX.

Table 5-6. Reference Setting for Virtual FIFO DMA TX (UART_TX)

Step	Address	Register Name	Local Address	R/W	Value	Description
1	AP_DMA+0x0C	AP_DMA_UART_x_TX_RST	AP_DMA_UART_x_TX_RST[0]	W	1'b1	Sets channel warm reset
2	AP_DMA+0x0C	AP_DMA_UART_x_TX_RST	AP_DMA_UART_x_TX_RST[0]	R	-	Waits for AP_DMA_UART_x_TX_RST[0] = 0
3	AP_DMA+0x24	AP_DMA_UART_x_TX_VFF_LEN	AP_DMA_UART_x_TX_VFF_LEN[15:3]	W	Length	Sets virtual size
4	AP_DMA+0x1C	AP_DMA_UART_x_TX_VFF_ADDR	AP_DMA_UART_x_TX_VFF_ADDR[31:3]	W	Address	Sets virtual FIFO address
5	AP_DMA+0x28	AP_DMA_UART_x_TX_VFF_THRE	AP_DMA_UART_x_TX_VFF_THRE[15:0]	W	Threshold	VFF threshold in byte alignment
6	AP_DMA+0x04	AP_DMA_UART_x_TX_INT_EN	AP_DMA_UART_x_TX_INT_EN[0]	W	1'b1	Controls interrupt enable 0: Disable 1: Enable
7	AP_DMA+0x08	AP_DMA_UART_x_TX_EN	AP_DMA_UART_x_TX_EN[0]	W	1'b1	Enables UART TX virtual FIFO DMA 0: Disable 1: Enable
8	Waiting for interrupt					
9	AP_DMA+0x00	AP_DMA_UART_x_TX_INT_FLAG	AP_DMA_UART_x_TX_INT_FLAG[0]	W	1'b0	Clears interrupt flag
10	Push data to virtual FIFO					
11	AP_DMA+0x2C	AP_DMA_UART_x_TX_VFF_WPT	AP_DMA_UART_x_TX_VFF_WPT[15:0]	W	-	Byte alignment FIFO write pointer
12	Repeat step 8 to step 11					
13	AP_DMA+0x10	AP_DMA_UART_x_TX_STOP	AP_DMA_UART_x_TX_STOP[0]	W	1'b1	Stops UART TX virtual FIFO

5.4.4.2.3 Virtual FIFO DMA RX (Example: UART_RX)

Table 5-7 shows the reference setting for virtual FIFO DMA RX.

Table 5-7. Reference Setting for Virtual FIFO DMA RX (UART_RX)

Step	Address	Register Name	Local Address	R/W	Value	Description
1	AP_DMA+0x0C	AP_DMA_UART_x_RX_RST	AP_DMA_UART_x_RX_RST[0]	W	1'b1	Sets channel warm reset
2	AP_DMA+0x0C	AP_DMA_UART_x_RX_RST	AP_DMA_UART_x_RX_RST[0]	R	-	Waits for AP_DMA_UART_x_RX_RST [0] = 0
3	AP_DMA+0x24	AP_DMA_UART_x_RX_VFF_LEN	AP_DMA_UART_x_RX_VFF_LEN[15:3]	W	Length	Sets virtual size
4	AP_DMA+0x1C	AP_DMA_UART_x_RX_VFF_ADDR	AP_DMA_UART_x_RX_VFF_ADDR [31:3]	W	Address	Sets virtual FIFO address
5	AP_DMA+0x28	AP_DMA_UART_x_RX_VFF_THRE	AP_DMA_UART_x_RX_VFF_THRE [15:0]	W	Threshold	VFF threshold in byte alignment
6	AP_DMA+0x04	AP_DMA_UART_x_RX_INT_EN	AP_DMA_UART_x_RX_INT_EN[0]	W	1'b1	Controls interrupt enable 0: Disable 1: Enable
7	AP_DMA+0x08	AP_DMA_UART_x_RX_EN	AP_DMA_UART_x_RX_EN[0]	W	1'b1	Enables UART RX virtual FIFO DMA 0: Disable 1: Enable
8	Waiting for interrupt					
9	AP_DMA+0x00	AP_DMA_UART_x_RX_INT_FLAG	AP_DMA_UART_x_RX_INT_FLAG[0]	W	1'b1	Clears interrupt flag
10	Pop data to virtual FIFO					
11	AP_DMA+0x30	AP_DMA_UART_x_RX_VFF_RPT	AP_DMA_UART_x_RX_VFF_RPT[15:0]	W	-	Byte alignment FIFO read pointer
12	AP_DMA+0x04	AP_DMA_UART_x_RX_INT_EN	AP_DMA_UART_x_RX_INT_EN[0]	W	1'b1	Controls interrupt enable 0: Disable 1: Enable
13	Repeat step 8 to step 12					
14	AP_DMA+0x10	AP_DMA_UART_x_RX_STOP	AP_DMA_UART_x_RX_STOP[0]	W	1'b1	Stops UART RX virtual FIFO

5.4.5 Register Definition

Refer to "MT7988A Wi-Fi 7 Platform Registers" for detailed register descriptions.

6 Clock and Power Control

6.1 Top Clock Generator

6.1.1 Introduction

This section introduces the Top Clock Generator (TOPCKGEN), which is also referred to as “CKSYS”. Clock architecture and a simple programming guide are included.

6.1.2 Features

Features of TOPCKGEN include:

- Use divider to generate clock from PLL to other designs
- Switch clock frequency for different designs
- Measure clocks from TOPCKGEN or other designs

TOPCKGEN provides a series of clocks of every IP. Each clock has several clock sources and can be turned off as well. When certain clock is switched from frequency A to frequency B, make sure frequencies A and B are available or else a system hang may occur. TOPCKGEN also comprises glitch-free clock MUX and digital clock divider to generate various clock frequencies.

To make sure that PLL or clock generated from TOPCKGEN is precise, a frequency meter design is available in TOPCKGEN. The frequency meter can measure clock not only from TOPCKGEN but also from other designs, such as DRAM controller or MCUSYS.

Additionally, debug output signals from TOPCKGEN to GPIO are available. These debug signals include XTAL clock, bandgap clock, measured clock and different reset signals. You can utilize these signals to check the condition of the chip.

6.1.3 Block Diagram

Clock generators exist not only at the top-level but also in every partition or system. Figure 6-1 shows the location of the top-level clock generator.

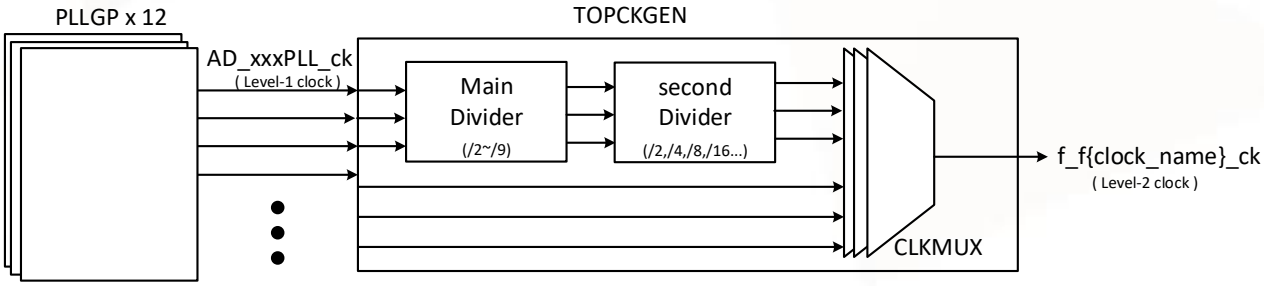


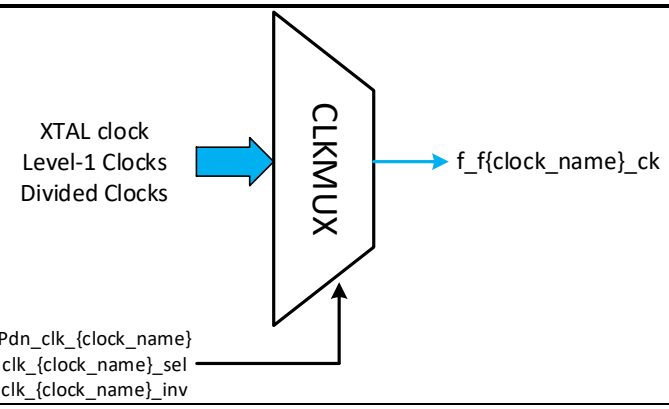
Figure 6-1. Block Diagram of Clock Architecture

6.1.3.1 Clock Multiplier

The clock dividers divide level-1 clock sources into several clocks. And the division ratio of each divider depends on level-2 clock's requirements.

A group of regular MUXs or buffers is available. You can choose the suitable sources from divided clocks or level-1 clocks directly to become level-2 clocks. All the clock MUXs have three kinds of control functions as Table 6-1 shows. XTAL is always one of the clock MUXs' sources and is also selected as default value to avoid system hang if PLL does not work.

Table 6-1. Example of Clock Multiplier

Related Registers: **CLK_CFG_0 to N** **CLK_CFG_0 to N_SET** **CLK_CFG_0 to N_CLR**	
Control Function 0 - pdn_*	Turn off CKMUX output
Control Function 1 - clk_*_inv	Inverse CKMUX output phase
Control Function 2 - clk_*_sel	Select CKMUX source

6.1.3.2 PLL

Figure 6-2 shows the PLL dedicated to MCUSYS.

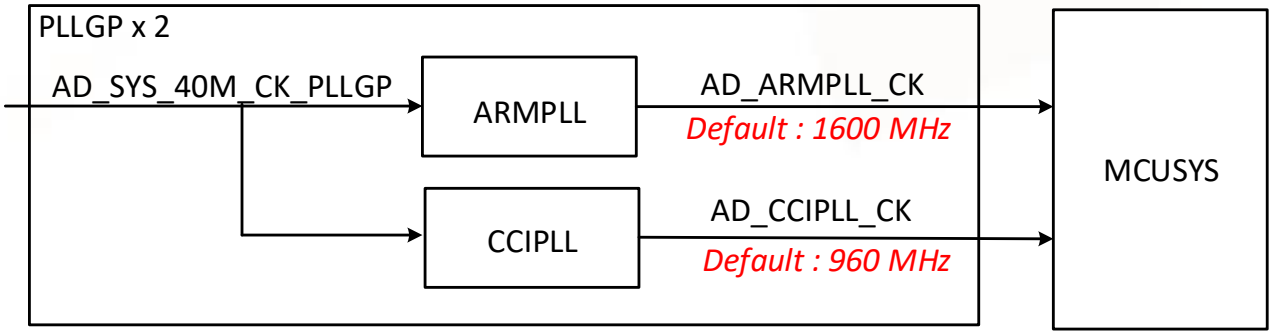


Figure 6-2. PLLx2 Block Diagram for MCUSYS

Figure 6-3 shows the remaining PLL structure. PLL can be dedicated to Audio, SGMII, DRAMC, MSDC, and USXGMII. Other types of PLL, such as MPLL, MMPLL, are shared with the top module; they can make the count of PLL be minimum and provide the clock frequency required by the top module.

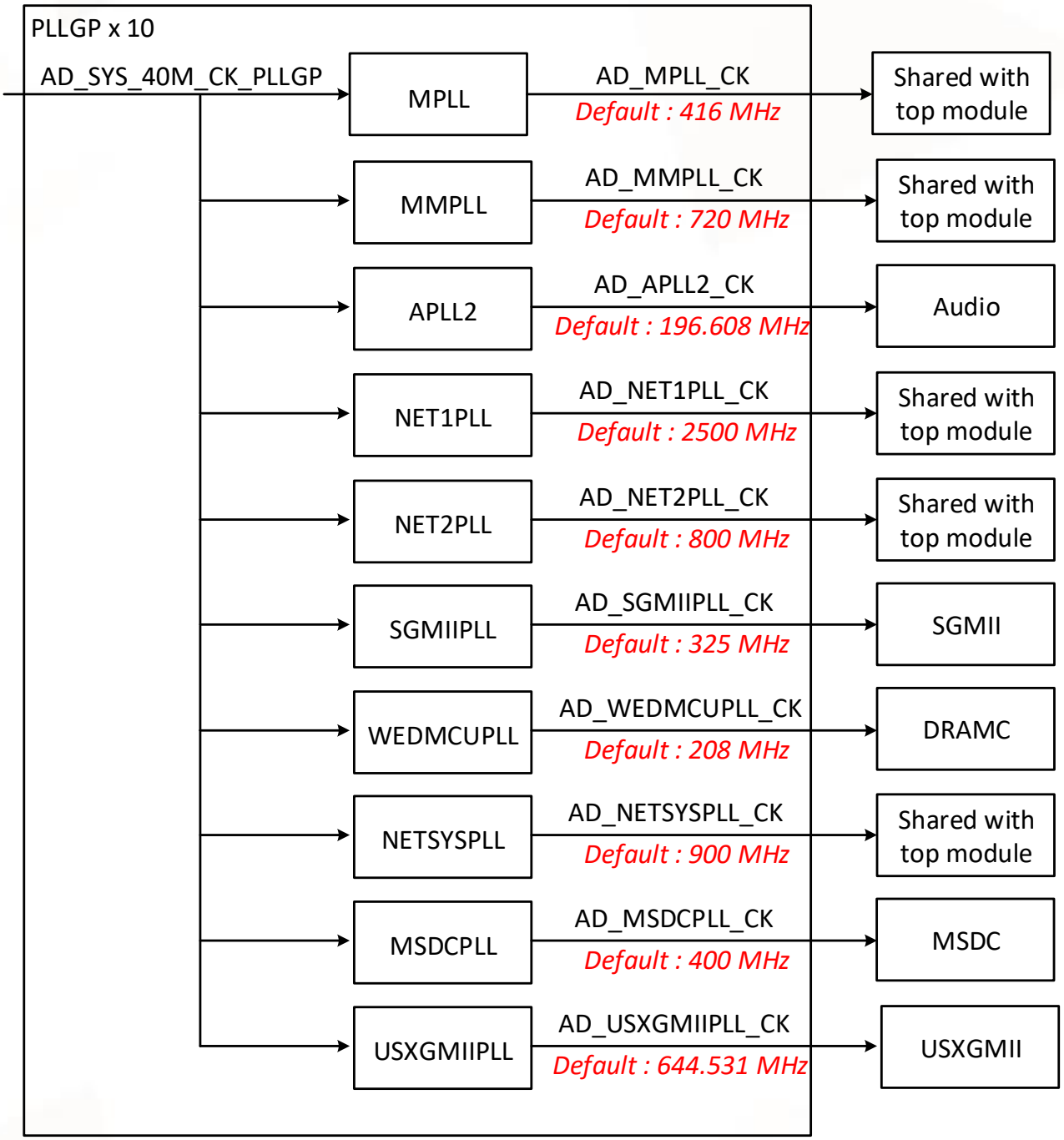


Figure 6-3. PLLx10 Block Diagram (For Reference Only)

6.1.3.3 Frequency Meter

There is one frequency meter inside TOPCKGEN and three types of input clock sources for the frequency meter. The first is for Phase-Locked Loops (PLLs) and TEST clock, called abist2_clock. The second one is for clocks generated from TOPCKGEN.

The frequency meter supports PAD output that allows detecting frequency directly instead of reading results from the frequency meter through IO monitor bus. See Figure 6-4 for the architecture and peripheral design.

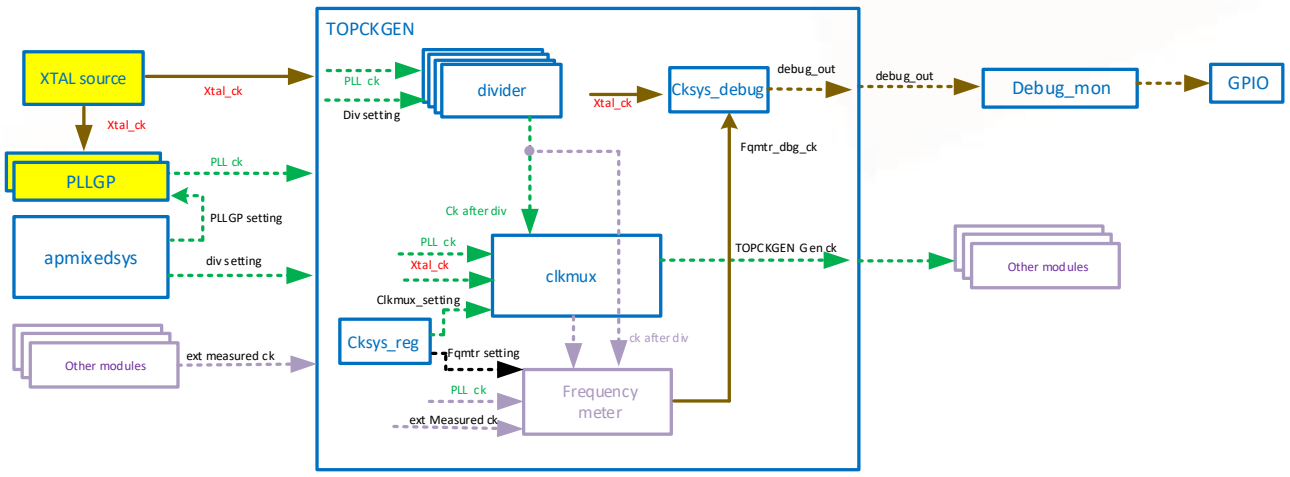


Figure 6-4. TOPCKGEN Clock Architecture

- CKMUX clock off
  - The clock can be turned on or off through setting the value of pdn_*.
  - This control bit cannot be switched along with clk_*_sel and clk_*_inv.
    - Same control address
  - Changing pdn_* with SET and CLEAR is recommended.

- Frequency meter
  - Firmware setting
  - For ABIST
    - CLK_DBG_CFG[1:0]: fqmtr_ck_sel: Select 0x0
    - CLK_DBG_CFG[21:16]: abist_ck_sel[5:0]: Select abist_clk0~63
  - For TOPCKGEN
    - CLK_DBG_CFG[1:0]: fqmtr_ck_sel: Select 0x1
    - CLK_DBG_CFG[14:8]: ckgen_ck_dbg_sel[5:0]: Select ckgen_clk0~128
  - CLK_MISC_CFG_0[31:24]: ckgen_k1[7:0]: Set to 0x00 for non-divider mode
  - CLK26CALI_0[8]: fmeter_en: Set 0x1 to enable frequency meter
  - CLK26CALI_0[0]: ckgen_tri_cal: Set 0x1 to start frequency measure
  - Wait CLK26CALI_0[0]: ckgen_tri_cal: Wait for 0x0; calibration is completed
  - Read CLK26CALI_0[31:16]: cal_cnt[15:0]
  - CLK26CALI_1[29:20]: ckgen_load_cnt[9:0]: Set to 0x27, frequency measure cycle
  - CLK26CALI_1[19:16]: ckgen_cal_target_offset[3:0]: Set measure offset
  - CLK26CALI_1[15:0]: ckgen_cal_target_value [15:0]: Set to measure clock target count
  - CLK26CALI_0[12]: ckgen_cal_cnt_valid

    * ckgen_cal_cnt_valid is 1 if

    (cal_cnt > ckgen_cal_target_value - ckgen_cal_target_offset)

    & (cal_cnt < ckgen_cal_target_value + ckgen_cal_target_offset)

- For example:

Write	CLK26CALI_1	0x1001B324	0x02700000	[29:20]: load_cnt
Write	CLK_MISC_CFG_0	0x1001B200	0x00000000	[31:24]: Set meter divider
Write	CLK_DBG_CFG	0x1001B240	0x00250000	[21:16]: Select xxx_ck [1:0] = 0: abist_clk
Write	CLK26CALI_0	0x1001B320	0x00000101	[8]: Enable frequency meter [0]: Start to measure

- Measure Freq = Read value (CLK26CALI_1[31:16]) * 40/(CLK26CALI_1[29:20] + 1).

Table 6-2. Abist2 Clock Selection

Name	Type	Sel.
1'b0	ABIST2	0
hf_faud_i2s_m_ck	ABIST2	1
AD_MPLL_416M_CK	ABIST2	2
AD_MPLL_416M_D2	ABIST2	3
AD_MPLL_416M_D3	ABIST2	4
AD_MPLL_416M_D4	ABIST2	5
AD_MPLL_416M_D8	ABIST2	6
AD_MMPLL_720M_CK	ABIST2	7
AD_MMPLL_720M_D2	ABIST2	8
AD_MMPLL_720M_D3	ABIST2	9
AD_MMPLL_720M_D3_D5	ABIST2	10
AD_MMPLL_720M_D4	ABIST2	11
AD_MMPLL_720M_D6	ABIST2	12
AD_MMPLL_720M_D8	ABIST2	13
AD_APLL2_196M_CK	ABIST2	14
AD_NET1PLL_2500M_D4	ABIST2	15
AD_NET1PLL_2500M_D5	ABIST2	16
AD_NET1PLL_2500M_D8	ABIST2	17
AD_NET2PLL_800M_CK	ABIST2	18
AD_NET2PLL_800M_D2	ABIST2	19
AD_NET2PLL_800M_D4	ABIST2	20
AD_NET2PLL_800M_D6	ABIST2	21
AD_NET2PLL_800M_D8	ABIST2	22
AD_SGMIIPLL_325M_CK	ABIST2	23
AD_WEDMCUPLL_208M_CK	ABIST2	24
AD_NETSYSPLL_850M_CK	ABIST2	25
AD_MSDCPLL_400M_CK	ABIST2	26
AD_XFIPLL_644M_OCC_CK	ABIST2	27

Table 6-3. TOPCKGEN Clock Selection

Name	Type	Sel.	Name	Type	Sel.
1'b0	CKGEN	0	f_faud_intbus_ck	CKGEN	32
f_fnetsys_ck	CKGEN	1	f_fapll_tuner_ck	CKGEN	33
f_fnetsys_500m_ck	CKGEN	2	f_fsspxtp_ref_ck	CKGEN	34
f_fnetsys_2x_ck	CKGEN	3	f_fssusb_dphy_ref_ck	CKGEN	35
f_fnetsys_gsw_ck	CKGEN	4	f_fusxgmii_sbus_0_ck	CKGEN	36
f_feth_gmii_ck	CKGEN	5	f_fusxgmii_sbus_1_ck	CKGEN	37
f_fnetsys_wed_mcu_ck	CKGEN	6	f_fsgmii_0_ck	CKGEN	38

Name	Type	Sel.	Name	Type	Sel.
f_fnetsys_pao_2x_ck	CKGEN	7	f_fsgmii_sbus_0_ck	CKGEN	39
f_feip197_ck	CKGEN	8	f_fsgmii_1_ck	CKGEN	40
f_faxi_infra_ck	CKGEN	9	f_fsgmii_sbus_1_ck	CKGEN	41
f_fuart_bclk_ck	CKGEN	10	f_fxfi_phy_0_ref_xtal_ck	CKGEN	42
f_femmc_250m_ck	CKGEN	11	f_fxfi_phy_1_ref_xtal_ck	CKGEN	43
f_femmc_400m_ck	CKGEN	12	f_fsysaxi_ck	CKGEN	44
f_fspi_ck	CKGEN	13	f_fsysapb_ck	CKGEN	45
f_fspim_mst_ck	CKGEN	14	f_feth_refclk_50m_ck	CKGEN	46
f_fnfi1x_ck	CKGEN	15	f_feth_syspll_200m_ck	CKGEN	47
f_fspinfi_bclk_ck	CKGEN	16	f_feth_syspll_ck	CKGEN	48
f_fpwm_bclk_ck	CKGEN	17	f_feth_xgmii_ck	CKGEN	49
f_fi2c_bclk_ck	CKGEN	18	f_fbus_tops_ck	CKGEN	50
f_fpcie_mbist_250m_ck	CKGEN	19	f_fnpu_tops_ck	CKGEN	51
f_fpextp_tl_clk_ck	CKGEN	20	dramc_ref_ck	CKGEN	52
f_fpextp_tl_clk_p1_ck	CKGEN	21	pwr_dramc_md32_ck	CKGEN	53
f_fpextp_tl_clk_p2_ck	CKGEN	22	csw_infra_f26m_ck	CKGEN	54
f_fpextp_tl_clk_p3_ck	CKGEN	23	f_fpextp_p0_ref_ck	CKGEN	55
f_fssusb_top_sys_ck	CKGEN	24	f_fpextp_p1_ref_ck	CKGEN	56
f_fssusb_top_sys_p1_ck	CKGEN	25	f_fpextp_p2_ref_ck	CKGEN	57
f_fssusb_top_xhci_ck	CKGEN	26	f_fpextp_p3_ref_ck	CKGEN	58
f_fssusb_top_xhci_p1_ck	CKGEN	27	f_fda_xtp_glb_p0_ck	CKGEN	59
f_fssusb_top_frmcnt_ck	CKGEN	28	f_fda_xtp_glb_p1_ck	CKGEN	60
f_fssusb_top_frmcnt_p1_ck	CKGEN	29	f_fda_xtp_glb_p2_ck	CKGEN	61
f_faud_ck	CKGEN	30	f_fda_xtp_glb_p3_ck	CKGEN	62
f_fa1sys_ck	CKGEN	31	f_fckm_ref_ck	CKGEN	63
f_fda_ckm_xtal_ck	CKGEN	64	f_fnetsys_tops_400m_ck	CKGEN	70
f_fpextp_ref_ck	CKGEN	65	f_fnetsys_ppefb_250m_ck	CKGEN	71
f_ftops_p2_26m_ck	CKGEN	66	f_fnetsys_warp_ck	CKGEN	72
mcusys_backup_625m_ck	CKGEN	67	f_feth_mii_ck	CKGEN	73
f_fnetsys_sync_250m_ck	CKGEN	68	f_fclk_npu_ck_cm_tops_ck	CKGEN	74
f_fmacsec_ck	CKGEN	69	1'b0	CKGEN	75

Table 6-4. Clock MUX

Clock Name	Clock Source	Clock Frequency (MHz)
f_fnetsys_ck	cb_cksq_40m	40.000
	cb_net2pll_d2	400.000
	cb_mmpll_d2	360.000
f_fnetsys_500m_ck	cb_cksq_40m	40.000
	cb_net1pll_d5	500.000
	net1pll_d5_d2	250.000
f_fnetsys_2x_ck	cb_cksq_40m	40.000
	cb_net2pll_800m	800.000
	cb_mmpll_720m	720.000
f_fnetsys_gsw_ck	cb_cksq_40m	40.000
	cb_net1pll_d4	625.000
	cb_net1pll_d5	500.000
f_feth_gmii_ck	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000

Clock Name	Clock Source	Clock Frequency (MHz)
f_fnetsys_wed_mcu_ck	cb_cksq_40m	40.000
	cb_net2pll_800m	800.000
	cb_mmpll_720m	720.000
	cb_net1pll_d4	625.000
	cb_net1pll_d5	500.000
	cb_mpll_416m	416.000
f_fnetsys_pao_2x_ck	cb_cksq_40m	40.000
	cb_net2pll_800m	800.000
	cb_mmpll_720m	720.000
	cb_net1pll_d4	625.000
	cb_net1pll_d5	500.000
	cb_mpll_416m	416.000
f_feip197_ck	cb_cksq_40m	40.000
	cb_netsyspll_850m	850.000
	cb_net2pll_800m	800.000
	cb_mmpll_720m	720.000
	cb_net1pll_d4	625.000
	cb_net1pll_d5	500.000
f_faxi_infra_ck	cb_cksq_40m	40.000
	net1pll_d8_d2	156.250
f_fuart_bclk_ck	cb_cksq_40m	40.000
	cb_mpll_d8	52.000
	mpll_d8_d2	26.000
f_femmc_250m_ck	cb_cksq_40m	40.000
	net1pll_d5_d2	250.000
	cb_mmpll_d4	180.000
f_femmc_400m_ck	cb_cksq_40m	40.000
	cb_msdcpll_400m	400.000
	cb_mmpll_d2	360.000
	cb_mpll_d2	208.000
	cb_mmpll_d4	180.000
	net1pll_d8_d2	156.250
f_fspi_ck	cb_cksq_40m	40.000
	cb_mpll_d2	208.000
	cb_mmpll_d4	180.000
	net1pll_d8_d2	156.250
	cb_net2pll_d6	133.333
	net1pll_d5_d4	125.000
	cb_mpll_d4	104.000
	net1pll_d8_d4	78.125
f_fspim_mst_ck	cb_cksq_40m	40.000
	cb_mpll_d2	208.000
	cb_mmpll_d4	180.000
	net1pll_d8_d2	156.250
	cb_net2pll_d6	133.333
	net1pll_d5_d4	125.000
	cb_mpll_d4	104.000
	net1pll_d8_d4	78.125

Clock Name	Clock Source	Clock Frequency (MHz)
f_fnfi1x_ck	cb_cksq_40m	40.000
	cb_mmpll_d4	180.000
	net1pll_d8_d2	156.250
	cb_net2pll_d6	133.333
	cb_mpll_d4	104.000
	cb_mmpll_d8	90.000
	net1pll_d8_d4	78.125
	cb_mpll_d8	52.000
f_fspinfi_bclk_ck	cksq_40m_d2	20.000
	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000
	cb_mpll_d4	104.000
	cb_mmpll_d8	90.000
	net1pll_d8_d4	78.125
	mmpll_d6_d2	60.000
	cb_mpll_d8	52.000
f_fpwm_bclk_ck	cb_cksq_40m	40.000
	net1pll_d8_d2	156.250
	net1pll_d5_d4	125.000
	cb_mpll_d4	104.000
	mpll_d8_d2	26.000
	cb_rtc_32k	0.032
f_fi2c_bclk_ck	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000
	cb_mpll_d4	104.000
	net1pll_d8_d4	78.125
f_fpcie_mbist_250m_ck	cb_cksq_40m	40.000
	net1pll_d5_d2	250.000
f_fpextp_tl_clk_ck	cb_cksq_40m	40.000
	cb_net2pll_d6	133.333
	cb_mmpll_d8	90.000
	mpll_d8_d2	26.000
	cb_rtc_32k	0.032
f_fpextp_tl_clk_p1_ck	cb_cksq_40m	40.000
	cb_net2pll_d6	133.333
	cb_mmpll_d8	90.000
	mpll_d8_d2	26.000
	cb_rtc_32k	0.032
f_fpextp_tl_clk_p2_ck	cb_cksq_40m	40.000
	cb_net2pll_d6	133.333
	cb_mmpll_d8	90.000
	mpll_d8_d2	26.000
	cb_rtc_32k	0.032
f_fpextp_tl_clk_p3_ck	cb_cksq_40m	40.000
	cb_net2pll_d6	133.333
	cb_mmpll_d8	90.000
	mpll_d8_d2	26.000
	cb_rtc_32k	0.032
f_fssusb_top_sys_ck	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000

Clock Name	Clock Source	Clock Frequency (MHz)
f_fssusb_top_sys_p1_ck	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000
f_fssusb_top_xhci_ck	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000
f_fssusb_top_xhci_p1_ck	cb_cksq_40m	40.000
	net1pll_d5_d4	125.000
f_fssusb_top_frmcnt_ck	cb_cksq_40m	40.000
	cb_mmpll_d3_d5	48.000
f_fssusb_top_frmcnt_p1_ck	cb_cksq_40m	40.000
	cb_mmpll_d3_d5	48.000
f_faud_ck	cb_cksq_40m	40.000
	cb_apll2_196m	196.608
f_fa1sys_ck	cb_cksq_40m	40.000
	cb_apll2_d4	49.152
f_faud_intbus_ck	cb_cksq_40m	40.000
	cb_apll2_196m	196.608
	mpll_d8_d2	26.000
f_fapll_tuner_ck	cb_cksq_40m	40.000
	cb_apll2_d4	49.152
f_fsspxtp_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fssusb_dphy_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fusxgmii_sbus_0_ck	cb_cksq_40m	40.000
	net1pll_d8_d4	78.125
f_fusxgmii_sbus_1_ck	cb_cksq_40m	40.000
	net1pll_d8_d4	78.125
f_fsgmii_0_ck	cb_cksq_40m	40.000
	cb_sgmiipll_325m	325.000
f_fsgmii_sbus_0_ck	cb_cksq_40m	40.000
	net1pll_d8_d4	78.125
f_fsgmii_1_ck	cb_cksq_40m	40.000
	cb_sgmiipll_325m	325.000
f_fsgmii_sbus_1_ck	cb_cksq_40m	40.000
	net1pll_d8_d4	78.125
f_fxfi_phy_0_ref_xtal_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fxfi_phy_1_ref_xtal_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fsysaxi_ck	cb_cksq_40m	40.000
	net1pll_d8_d2	156.250
f_fsysapb_ck	cb_cksq_40m	40.000
	mpll_d3_d2	69.333
f_feth_refclk_50m_ck	cb_cksq_40m	40.000
	net2pll_d4_d4	50.000
f_feth_syspll_200m_ck	cb_cksq_40m	40.000
	cb_net2pll_d4	200.000
f_feth_syspll_ck	cb_cksq_40m	40.000
	net1pll_d5_d2	250.000

Clock Name	Clock Source	Clock Frequency (MHz)
f_feth_xgmii_ck	cksq_40m_d2	20.000
	net1pll_d8_d8	39.063
	net1pll_d8_d16	19.531
f_fbus_tops_ck	cb_cksq_40m	40.000
	cb_net1pll_d5	500.000
	cb_net2pll_d2	400.000
f_fnpu_tops_ck	cb_cksq_40m	40.000
	cb_net2pll_800m	800.000
dramc_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
pwr_dramc_md32_ck	cb_cksq_40m	40.000
	cb_mpll_d2	208.000
	cb_wedmcupll_208m	208.000
csw_infra_f26m_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fpextp_p0_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fpextp_p1_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fpextp_p2_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fpextp_p3_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fda_xtp_glb_p0_ck	cb_cksq_40m	40.000
	cb_net2pll_d8	100.000
f_fda_xtp_glb_p1_ck	cb_cksq_40m	40.000
	cb_net2pll_d8	100.000
f_fda_xtp_glb_p2_ck	cb_cksq_40m	40.000
	cb_net2pll_d8	100.000
f_fda_xtp_glb_p3_ck	cb_cksq_40m	40.000
	cb_net2pll_d8	100.000
f_fckm_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fda_ckm_xtal_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_fpextp_ref_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
f_ftops_p2_26m_ck	cksq_40m_d2	20.000
	mpll_d8_d2	26.000
mcusys_backup_625m_ck	cb_cksq_40m	40.000
	cb_net1pll_d4	625.000
f_fnetsys_sync_250m_ck	cb_cksq_40m	40.000
	net1pll_d5_d2	250.000
f_fmacsec_ck	cb_cksq_40m	40.000
	cb_sgmiipll_325m	325.000
	cb_net1pll_d8	312.500
f_fnetsys_tops_400m_ck	cb_cksq_40m	40.000
	cb_net2pll_d2	400.000
f_fnetsys_ppefb_250m_ck	cb_cksq_40m	40.000
	net1pll_d5_d2	250.000

Clock Name	Clock Source	Clock Frequency (MHz)
f_fnetsys_warp_ck	cb_cksq_40m	40.000
	cb_net2pll_d2	400.000
	cb_mmpll_d2	360.000
f_feth_mii_ck	cksq_40m_d2	20.000
	net2pll_d4_d8	25.000
f_fclk_npu_ck_cm_tops_ck	cb_cksq_40m	40.000
	cb_net2pll_800m	800.000
	cb_mmpll_720m	720.000

6.1.4 Programming Guide

6.1.4.1 Clock Switching

Make sure clock A and clock B are available before changing the setting of clk_*_sel. If switched to a non-existing clock, the clock switch is stuck until non-existing clock is turned on to free the clock switch.

Multi-clock switching is supported at the same time (without changing pdn_*).

6.1.4.2 Switching from Clock A to Clock B

Make sure clock B is ready.

1. Change clk_*_sel.
2. Write *_ck_update.
3. Wait until chg_sta = 1'b0 (optional).

6.1.5 Register Definition

Refer to "MT7988A Wi-Fi 7 Platform Registers" for detailed register descriptions.

6.2 TOP Reset Generator Unit (TOPRGU)

6.2.1 Introduction

The top reset generator unit (TOPRGU) generates reset signals and distributes the signals to each system. A watchdog timer (WDT) is also included in this module.

6.2.2 Features

- Hardware reset signals for the whole chip
- Software controllable reset for each system (except for infrastructure and apmixedsys systems)
- Watchdog timer
- Reset output signals for companion chips

6.2.3 Block Diagram

In the MT7988A, TOPRGU only supports SEJ and WDT events.

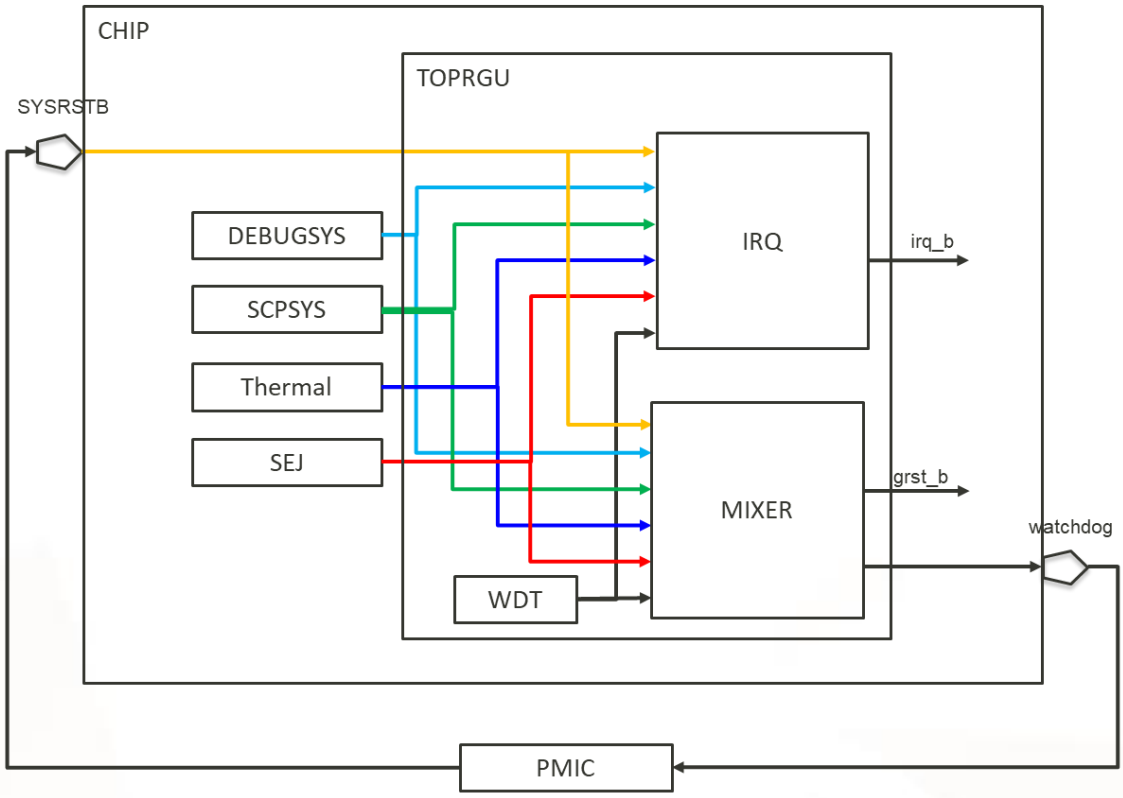


Figure 6-5. Block Diagram of Top Reset Generator Unit

6.2.4 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

6.3 AP Mixedsys

6.3.1 Introduction

AP (Application Processor) mixedsys provides the control signals of Phase-Locked Loops (PLLs) and some other analog macros. It also provides a debug output for On-Chip Clock (OCC).

6.3.2 Phase-Locked Loop

6.3.2.1 Clock Introduction

There are in total 12 PLLs in the analog PLLGP macro, ARMPLL_B, CCIPLL2_B, MSDCPLL, ARMPLL_RSV, NETSYSPLL, NET1PLL, NET2PLL, MMPLL, MPLL, SGMIIPLL, WEDMCUPLL and APLL2. These reference clocks all come from 40 MHz XTAL. The PLLs provide clock sources for CPU, BUS and Ethernet.

6.3.2.2 Block Diagram

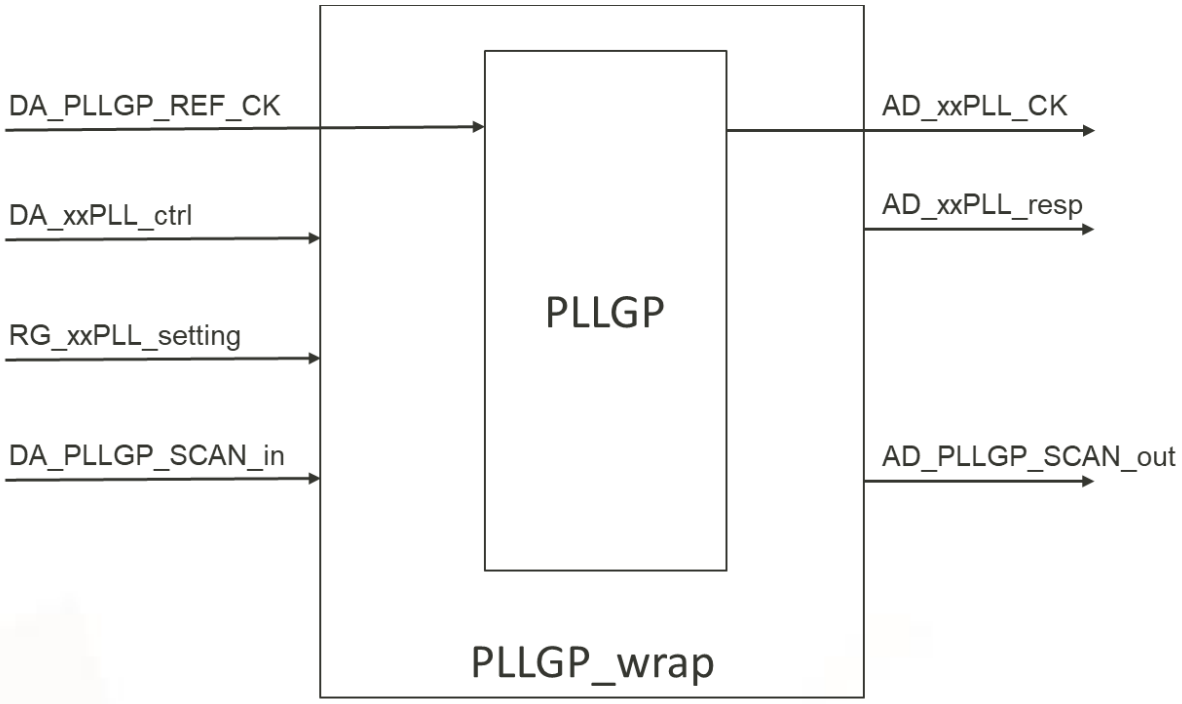


Figure 6-6. Block Diagram of Clock Sources

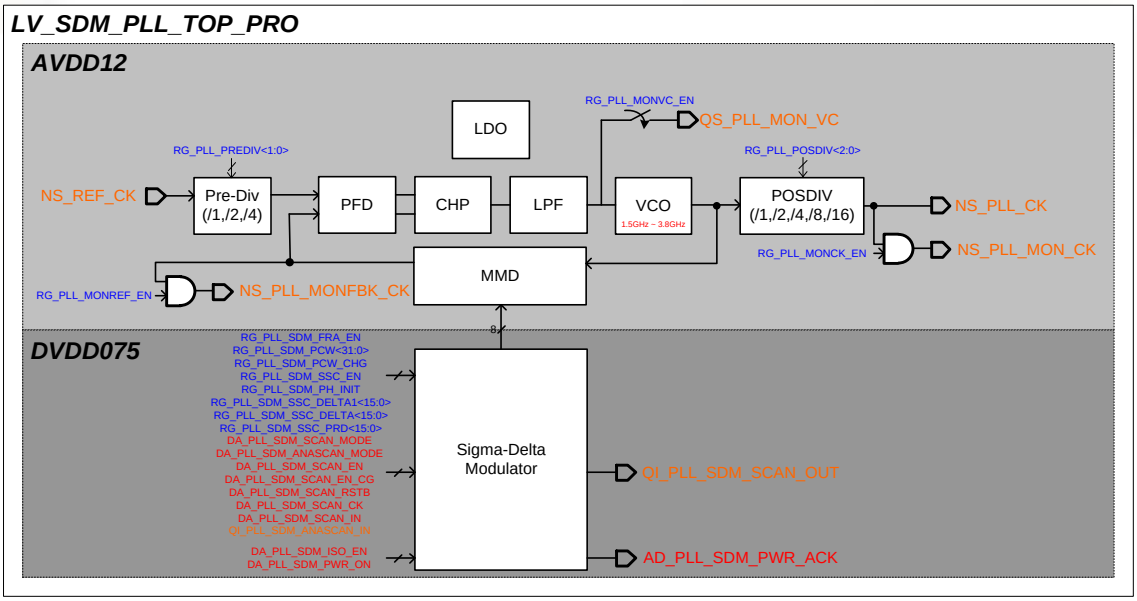


Figure 6-7. Block Diagram of PLL Core

6.3.2.3 Functional Specifications

See the table below for the functional specifications of PLL.

LV_SDM_PLL_TOP_V12			
	Support	**Unit**	**Note**
Input clock frequency (Fin)	20 to 120	MHz	Before Pre-divider
VCO clock frequency (Fvco)	1500 to 3800	MHz	Post-divider supports /1, /2, /8, or /16
Output clock frequency (Fout)	125 to 3800	MHz	-
Reference clock frequency (Fref)	20 to 30	MHz	(Fin /1, /2, or /4)
Modulus ratio	6 to 255	-	-
Output phase	1	-	-
Output clock duty cycle	45 to 55 (POSDIV: /1) 47 to 53 (POSDIV: others)	%	For typical voltage only
Output clock phase noise jitter	< 60 ps	ps (rms)	-
Output clock period jitter	Freq < 1 GHz < 60 Freq < 1 to 1.5 GHz < 30 Freq < 1.5 to 2.0 GHz < 25 Freq < 2.0 to 3.8 GHz < 20	ps (pk-pk)	-
PLL bandwidth	Fref/15 or Fref/30	MHz	-
Static phase error	< ± 100 ps	ps	-
Operating temperature	-40 to 125	°C	-
Low voltage power supply	DVDD075 = 0.75 ± 10%	V	DVDD075
High voltage power supply	AVDD18 = 1.8 ± 10% AVDD12 = 1.2 ± 10%	V	AVDD18 AVDD12
Current consumption	AVDD12 < 1 AVDD18 < 0.1 DVDD075 < 0.1	mA	-
Power-down current	AVDD12 < 1 AVDD18 < 1 DVDD075 < 1	uA	MTCMOS for decreasing leakage in DVDD075
Area (W x H)	100 x 100	um^2	-
Power-on setting time	< 20	us	-

6.3.2.4 PLL Power-on Sequence

As the following figure shows, LV_SDM_PLL requires an appropriately configured power-on sequence. The power-on or power-off setting sequence is implemented by software as long as the timing constraints in the following figure are followed. Software has to turn on each PLL signal step by step.

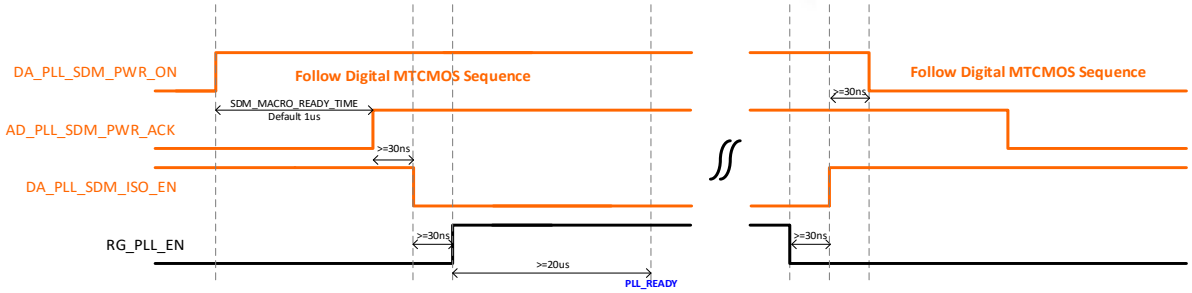


Figure 6-8. PLL Power-on Sequence

6.3.3 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7 Peripherals

7.1 MSDC (MMC and SD Card Controller)

7.1.1 Introduction

The MMC (MultiMediaCard) and SD (Secure Digital) card controller, also referred to as MSDC, fully supports the functions listed below.

- SD memory card specification version 3.0
- SDIO card specification version 3.0
- eMMC 5.1

7.1.2 Features

There is only one host controller of MSDC, MSDC0, which outputs two pinmux ports. Detailed features of the MSDC are as follows:

- MSDC0 is used as eMMC 5.1.
- MSDC0a is used as SD/eMMC 4.5.

The features of each MSDC are as follows:

MSDC0

- 64-bit data access on Advanced eXtensible Interface (AXI) bus
- 32-bit access on Advanced High-Performance Bus (AHB) for control registers
- Support basic Direct Memory Access (DMA) mode and linked-list based DMA mode
- Support eMMC speed modes:
  - Backwards compatibility with legacy MMC
  - High-speed Single Data Rate (SDR) mode
  - High-speed Dual Data Rate (DDR) mode
  - HS200 mode
  - HS400 mode
- Support for eMMC boot-up mode
- eMMC bus widths: 1 bit, 4 bits or 8 bits
- Support command queuing

MSDC0a

- 64-bit data access on Advanced eXtensible Interface (AXI) bus
- 32-bit access on Advanced High-Performance Bus (AHB) for control registers
- Support for basic DMA mode and linked-list based DMA mode
- eMMC speed mode:

- Backwards compatibility with legacy MMC
- High-speed Single Data Rate (SDR) mode
- High-speed Dual Data Rate (DDR) mode

- SD bus speed mode:
  - Default speed mode
  - High-speed mode
  - SDR12 mode
  - SDR25 mode
  - SDR50 mode
  - DDR50 mode
- eMMC4.5 bus widths: 1 bit, 4 bits or 8 bits
- SD bus widths: 1 bit or 4 bits

7.1.3 Block Diagram

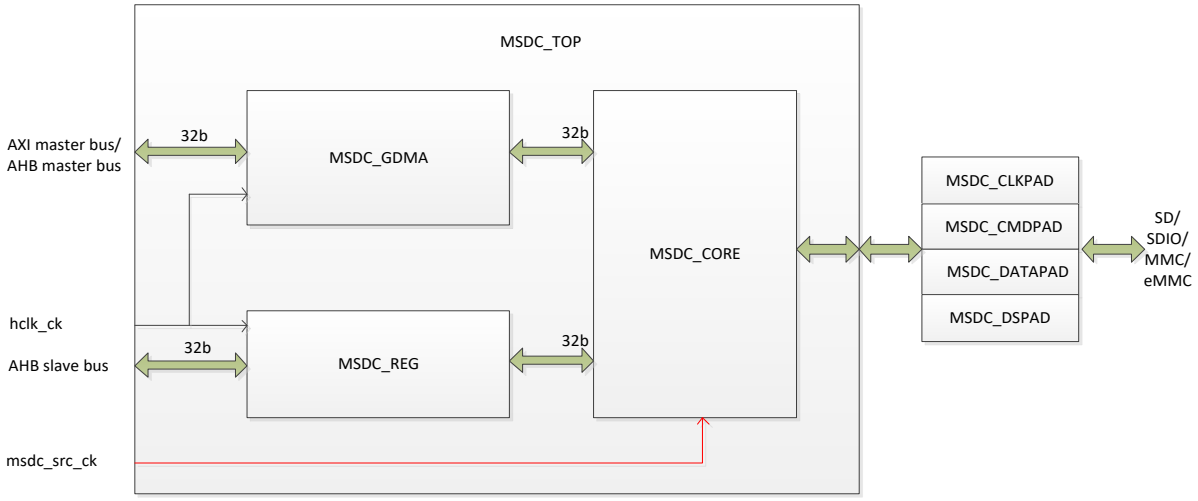


Figure 7-1. Block Diagram of MSDC

Table 7-1. MSDC Functions and Address

MSDC	Base Address	Feature
MSDC0	0x11230000	eMMC5.1
MSDC1	0x11240000	SD3.0/SDIO3.0

The MSDC is composed of three parts: MSDC_GDMA, MSDC_CORE and MSDC_REG.

- MSDC_REG: Register to configure MSDC
- MSDC_CORE: Main controller of MSDC, implementing the transfer between host and device
- MSDC_GDMA: DMA engine, transferring data between MSDC and memory

7.1.4 MSDC Recommended Command Sequence

For SD command without response

- Check whether SDC_STA.SDCBUSY is 0 before starting to issue this command.
- When this command is done, check the MSDC_INT. SD_CMDRDY, MSDC_INT.SD_CMDTO, MSDC_INT.SD_RESP_CRCERR bits for status.

For SD command with response

- Check whether SDC_STA.SDCBUSY is 0 before starting to issue this command
- When this command is done, check the MSDC_INT. SD_CMDRDY, MSDC_INT.SD_CMDTO, MSDC_INT.SD_RESP_CRCERR bits for status.
- Find the response in SDC_RESP0 ~ SDC_RESP3.

For SD command with data read/write transfer

- Check whether SDC_STA.SDCBUSY is 0 before starting to issue this command.
- When this command is done, check the MSDC_INT. SD_CMDRDY/SD_CMDTO/SD_RESP_CRCERR bits for command phase status.
- Find the response in SDC_RESP0 ~ SDC_RESP3.
- Enable DMA if needed (Program DMA_CTRL register).
- Program the DMA_CTRL register after the command register is programmed.
- Use the PIO mode to move data (MSDC_FIFOCS, MSDC_RXDAT, MSDC_TXDAT registers)
- Do not switch between PIO mode and DMA mode in the same transfer; otherwise, the result will be unexpected.
- Check MSDC_INT.SD_XFER_COMPLETE/DMA_DONE/SD_DATTO/SD_DATA_CRCERR for data phase status. See detailed timing in the timing diagrams in the following sections.

For SD, software can always check SDC_STA.SDCBUSY before issuing a new command.

- SDC_STA_CMDBUSY is left to be referenced.

7.1.4.1 Single Block Read/Write in PIO Mode Flow Chart

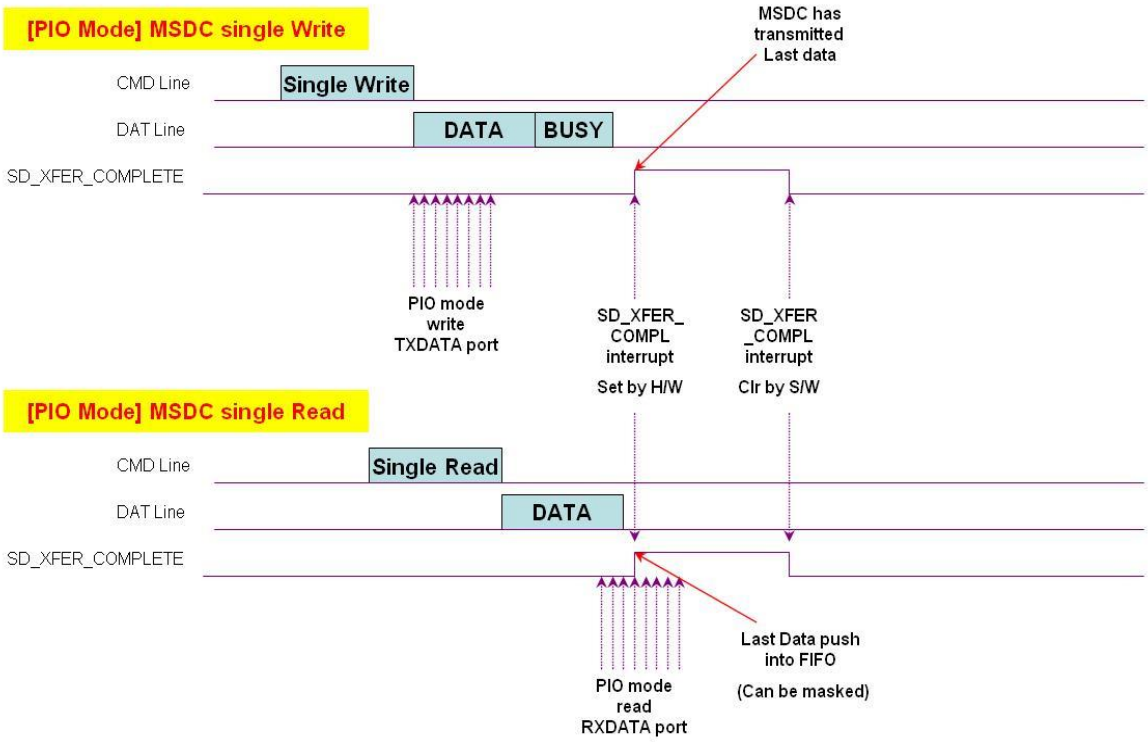


Figure 7-2. PIO Single Block Timing

7.1.4.2 Multiple Block Read/Write in PIO Mode Flow Chart

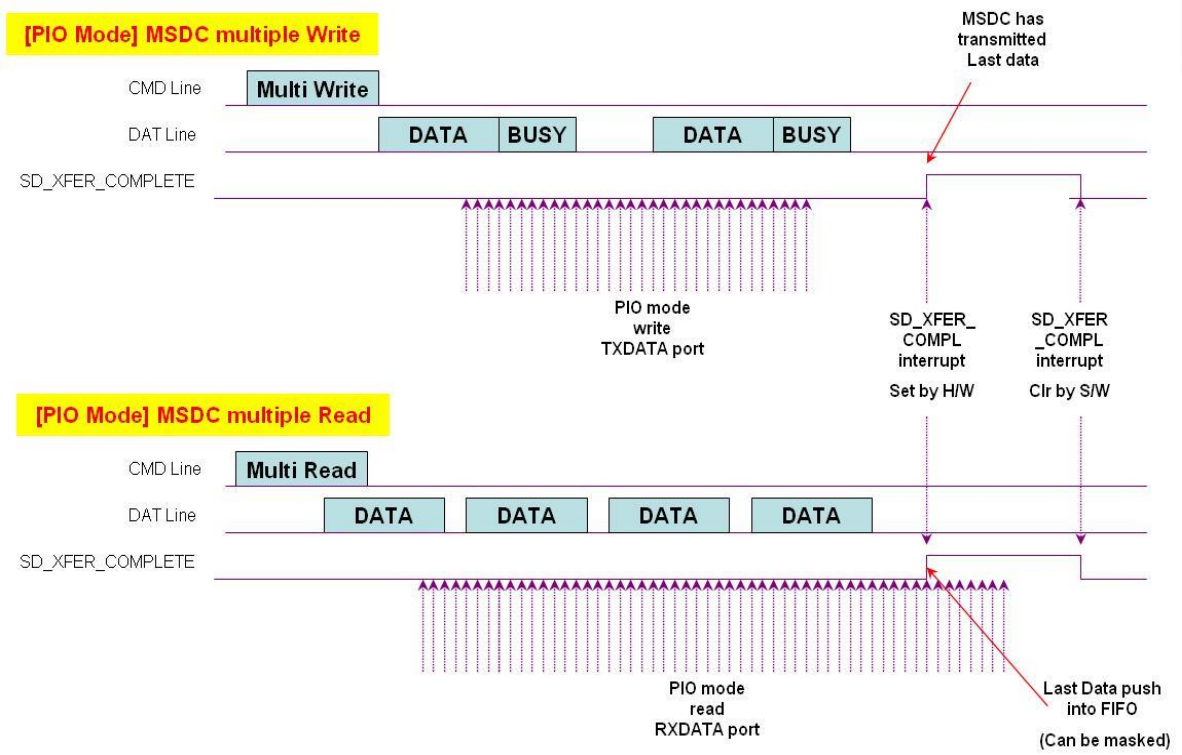


Figure 7-3. PIO Multiple Block Timing

7.1.4.3 MSDC_INT.SD_XFER_COMPLETE/DMA_DONE Timing Diagram

DMA_Read

When MSDC_CFG.PIO_MODE (MSDC0 base_address+0x0000) [3] is not set, the MSDC works in the DMA mode. The MSDC receives the data from device and writes them to the target SDRAM address through MSDC_GDMA control. Software configures the DMA_SA (MSDC0 base_address+0x0090) [31:0] register as the start address in SDRAM. After the transfer is finished, two interrupts are generated. Software clears the interrupt bit after receiving it.

DMA_Write

When MSDC_CFG.PIO_MODE (MSDC0 base_address+0x0000) [3] is not set, the MSDC works in the DMA mode. The MSDC gets the data from SDRAM through MSDC GDMA control and writes them to device. Software configures the DMA_SA register as the start address in SDRAM. After the transfer is finished, two interrupts are generated, and software clears the interrupt bit after receiving it.

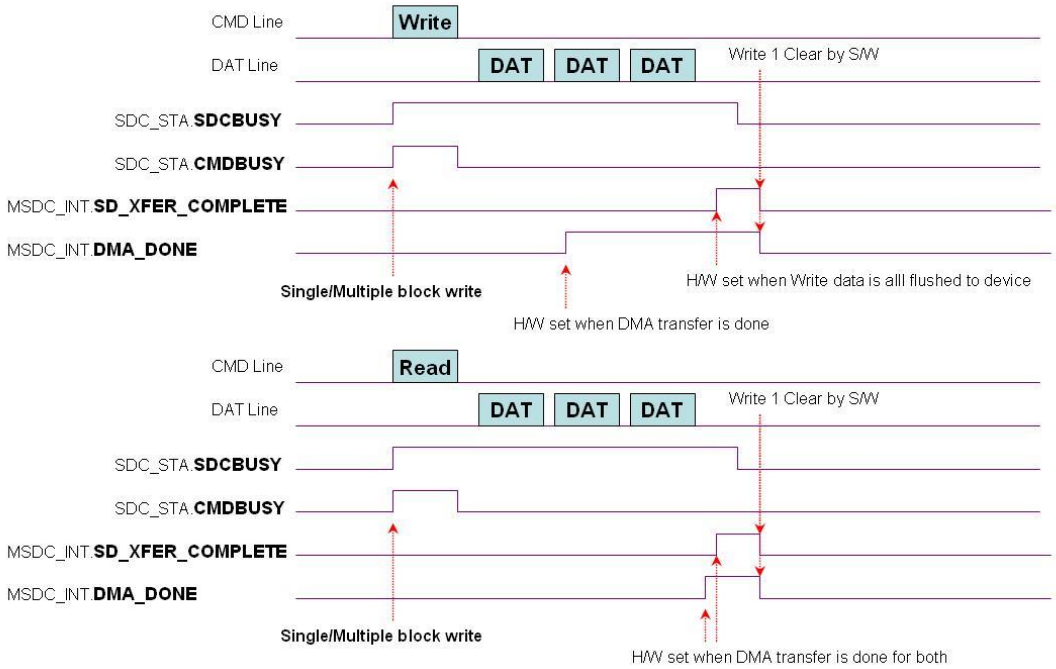


Figure 7-4. DMA Single/Multiple Block Timing

7.1.4.4 MSDC Tuning Support

The host controller provides a tuning algorithm to guarantee that the host can sample the data, command response and CRC status stably from the device. Both command and data have pad_delay. The tuning flow is as follows:

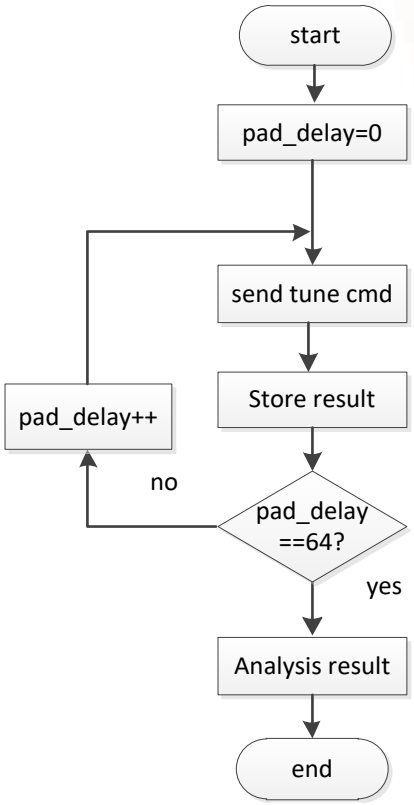


Figure 7-5. Tuning Flow

7.1.5 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.2 Serial Peripheral Interface (SPI) Master

7.2.1 Introduction

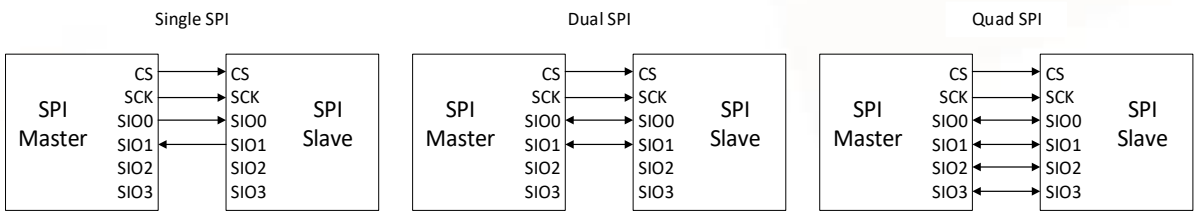


Figure 7-6. Pin Connection Between SPI Master and SPI Slave

The serial peripheral interface (SPI) is a bit-serial transmission protocol supporting single mode (4-pin), dual mode (4-pin) and quad mode (6-pin) for increased data throughput. The maximum serial clock (SCK) frequency is 52 MHz. Note that single mode and dual mode support both full and half duplex modes, and the quad mode only supports half duplex mode. Figure 7-6 is an example of the connection between the SPI master and the SPI slave. The SPI is a master responsible for data transmission with the slave.

Table 7-2. SPI Master Interface

Signal Name	Type	Description
CS	O	Low active chip select signal
SCK	O	The (bit) serial clock (max. SCK clock rate = 52 MHz)
SIO0/MOSI	I/O	Data signal 0
SIO1/MISO	I/O	Data signal 1
SIO2/WP	I/O	Data signal 2
SIO3/HLOD	I/O	Data signal 3

The abbreviations used in this chapter are listed in Table 7-3.

Table 7-3. Abbreviations for SPI Master

Abbreviation	Definition
MOSI	Master out slave in
MISO	Master in slave out
CPOL	Clock polarity
CPHA	Clock phase

7.2.2 Features

The features of the SPI master are:

- Supports single mode (4-pin), dual mode (4-pin) and quad mode (6-pin). Can automatically set port direction for data input/output if registers SPI_PIN_MODE, SPI_HALF_DUPLEX_EN and SPI_HALF_DUPLEX_DIR are already set.
- Configurable CS_N setup time, hold time and idle time (see Figure 7-7)

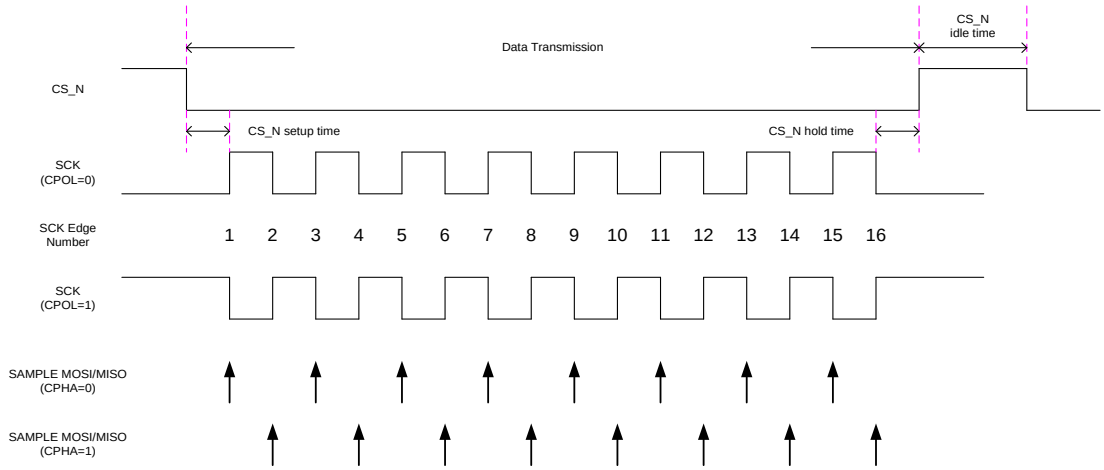


Figure 7-7. SPI Transmission Formats

- CS_N setup time, hold time and idle time can be adjusted by setting the corresponding registers.
- Programmable SCK high time and low time: SCK high time and low time can be set separately. Thus, for a given baud rate, SCK with a wide range of duty cycles can be generated.
- Configurable transmitting and receiving bit order: 2 options for bit order, MSB or LSB first
- 2 configurable modes for the source of the data to be transmitted
  - In TX DMA mode, the SPI master automatically fetches the data to be put on the MOSI line from memory.
  - In TX PIO mode, the data to be transmitted on the MOSI line are written to FIFO by software before the start of the transaction.

    Note: The value of SPI_TX_SRC (SPIn Base address+0x0008) [31:0] must be 4-byte aligned. TX data should be prepared before the transaction. In DMA mode, set TX_DMA_EN (SPIn Base address+0x0018) [11] to 1'b1. In PIO mode, software must put data into TX FIFO through the SPI_TX_DATA (SPIn Base address+0x0010) [31:0] register.
- The depth of TX FIFO is 32 bytes.
  - In TX DMA mode, the data to be put on the MOSI line are prepared in advance in memory; the SPI master automatically reads the data from memory.
  - In TX PIO mode, writing to the SPI_TX_DATA (SPIn Base address+0x0010) [31:0] register writes 4 bytes to TX FIFO. The TX FIFO pointer automatically moves towards the next 4 bytes.
- 2 configurable modes for the destination of data to be received.
  - In RX DMA mode, the SPI master automatically stores the received data (from MISO line) to memory.
  - In RX PIO mode, the received data are stored in RX FIFO of the SPI master. The processor must read back the data by itself.

    Note: The value of SPI_RX_DST (SPIn Base address+0x000C) [31:0] must be 4-byte aligned.
- The depth of RX FIFO is 32 bytes.

  - In RX DMA mode, the received data from the MISO line are moved to memory automatically by the SPI master.
  - In RX PIO mode, reading from SPI_RX_DATA (SPIn Base address+0x0014) [31:0] register reads 4 bytes from RX FIFO. The RX FIFO pointer automatically moves towards the next 4 bytes.
- Programmable byte length for transmission
  - PACKET_LENGTH (SPIn Base address+0x0004) [31:16] defines the number of bytes in one packet.
  - PACKET_LOOP_CNT (SPIn Base address+0x0004) [15:8] defines the number of packets within one transaction
  - Number of bytes in one packet = PACKET_LENGTH + 1
  - Number of packets in one transaction = PACKET_LOOP_CNT + 1
  - Total bytes of one transaction = (PACKET_LENGTH + 1)*(PACKET_LOOP_CNT + 1)
- Unlimited length for transmission
  - Achieved by the operation in PAUSE mode
  - In PAUSE mode, the CS_N signal stays active (low) after the transmission. At this time, the SPI master is in PAUSE_IDLE state, ready to receive the resume command. See Figure 7-8 for the state transition.

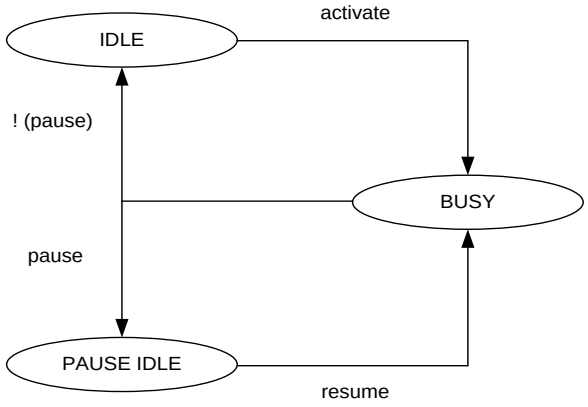


Figure 7-8. Operation Flow With or Without PAUSE Mode

- Configurable option to control CS_N deassert between byte transfers. The controller supports a special transmission format called CS_N deassert mode. Figure 7-9 illustrates the waveform in this transmission format.

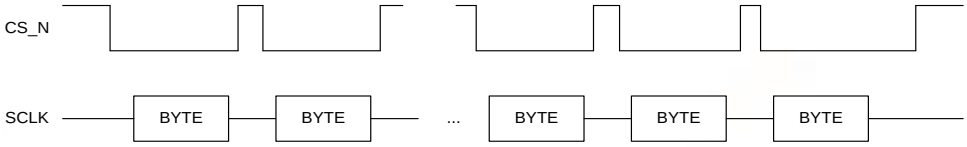


Figure 7-9. CS_N Deassert Mode

- When the SPI master operates in dual or quad mode with half duplex, the transmission package includes three phases: command phase, address phase and data phase.
  - The command phase always operates in single mode.
  - The address phase cannot transmit or receive data.

- The data phase operation depends on the settings of SPI_PIN_MODE and SPI_HALF_DUPLEX_DIR. The command phase and address phase are useful for special applications, such as reading or writing serial flash data.

- 4 communication modes available (Modes 0, 1, 2 and 3)
  - The four modes define the SCLK edge where the MOSI line toggles and the master samples the MISO line. They also define the SCLK steady level - whether the clock level is high or low, when the clock is not active. Each mode is formally defined with a pair of parameters, clock polarity (CPOL) and clock phase (CPHA).

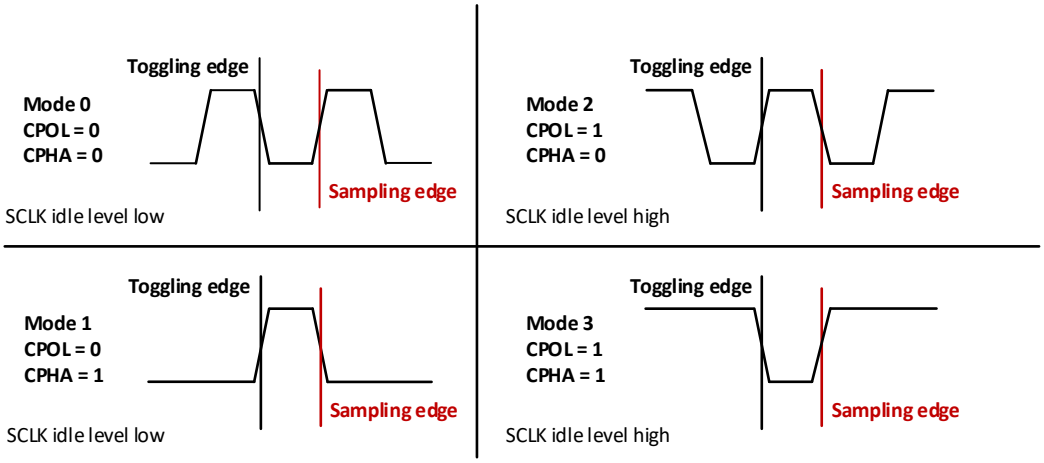


Figure 7-10. Waveforms of Four Communication Modes

- CS GPIO mode (SPIn Base address+0x0018) [18]: The SPI_CS pin can be replaced by any GPIO pin. Software controls the GPIO pin as the SPI_CS pin, so a single SPI master can be selected and communicate with multiple SPI slaves. Figure 7-11 is the block diagram of the multi-slave topology for single mode. The CS pin for devices 2, 3, 4 and so on is replaced by the GPIOn pin.

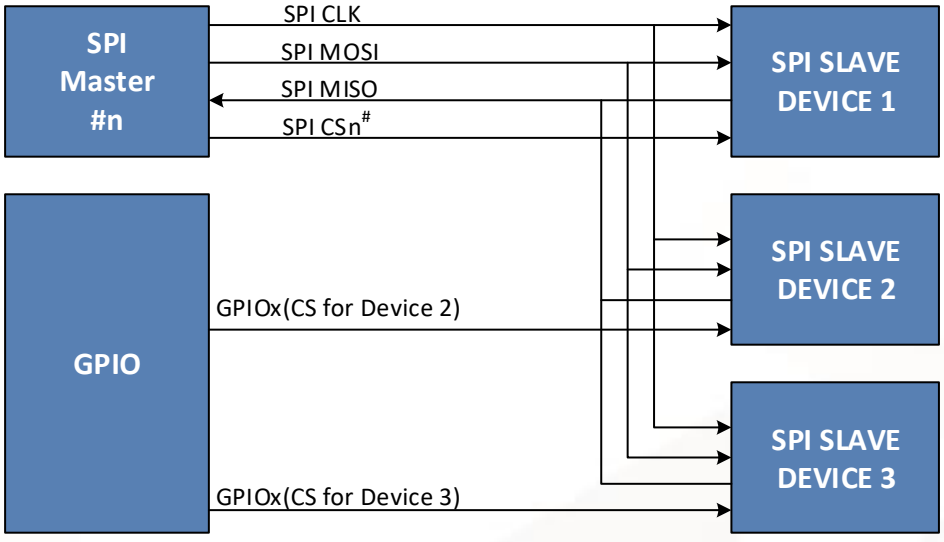


Figure 7-11. Pin Connection Between One Master and Multiple Slaves in Single Mode

7.2.3 Block Diagram

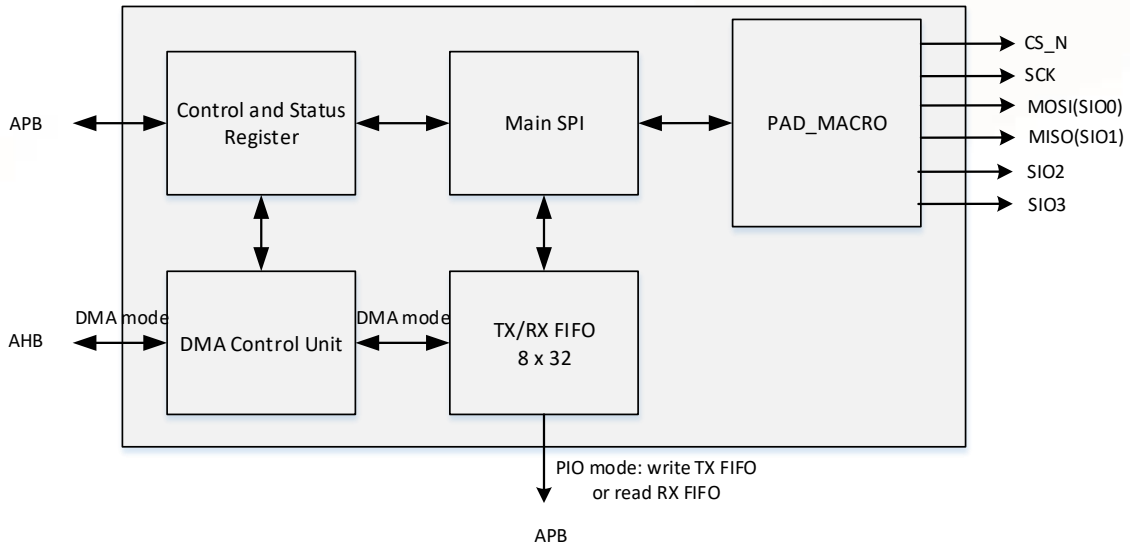


Figure 7-12. Block Diagram of SPI Master

See Figure 7-12. The SPI master consists of control and status registers, DMA control unit, main SPI, FIFO and PAD_MACRO. PAD_MACRO controls the SPI data capture and data transmission to and from SPI. The control and status registers receive commands from the system. The DMA control unit communicates with SYSRAM when the SPI master is set to the DMA mode. Both TX and RX have an 8 x 32-bit FIFO for storing data. The main SPI is the functional unit.

In PIO mode, software can write data into TX FIFO via SPI_TX_DATA (SPIn Base address+0x0010) [31:0] or read data from RX FIFO via SPI_RX_DATA (SPIn Base address+0x0014) [31:0]. In DMA mode, the SPI master can automatically get data from or send data to SYSRAM via the AHB after software configures DMA parameters.

See Table 7-2 for pin descriptions. The SPI master I/O operates at 1.8V.

7.2.4 Theory of Operation

7.2.4.1 SPI Transaction Format

The SPI master supports single mode, dual mode and quad mode. The single mode and dual mode support both full and half duplex modes. The quad mode only supports half duplex mode.

7.2.4.1.1 SPI Single Mode and Full Duplex Mode Transaction

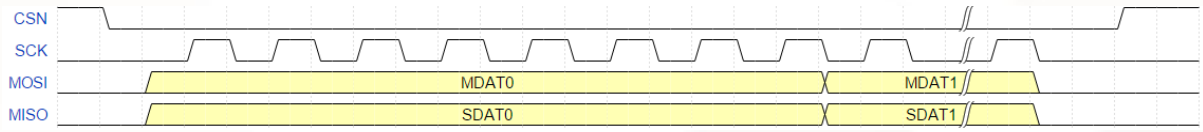


Figure 7-13. SPI Single Mode, Full Duplex Mode Transaction Formats

In DMA mode, the data to be transferred should be prepared in memory in advance. In PIO mode, the system should push the data that are to be transferred into SPI TX FIFO first. After receiving the START (SPIn Base address+0x0018) [0] command, the SPI sends data to slave continuously and receives data from slave at the same time. If register SPI_CFG3[1:0] is set to 2'b00, SPI_CFG3[3] is set to 1'b0, and the SPI master works in single mode and full duplex mode. See Figure 7-13 for the waveform on the SPI pins. MDATA is the data transferred from SPI master to SPI slave. SDATA is the data transferred from SPI slave to SPI master.

7.2.4.1.2 SPI Single Mode and Half Duplex Mode Transaction

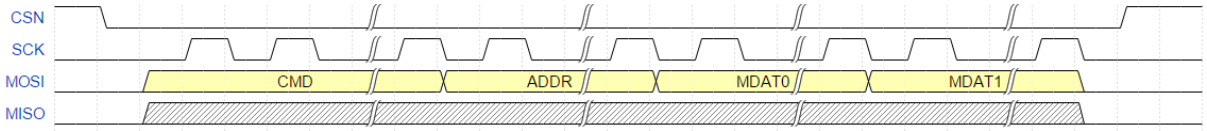


Figure 7-14. SPI Single Mode, Half Duplex Mode, TX Transaction Formats

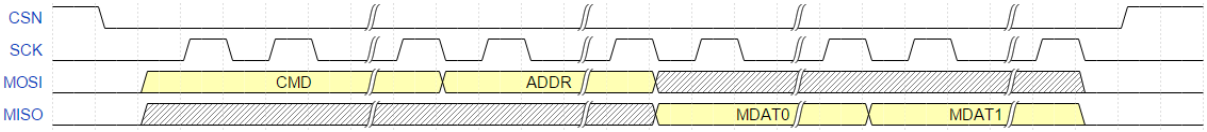


Figure 7-15. SPI Single Mode, Half Duplex Mode, RX Transaction Formats

If register SPI_CFG3[1:0] is set to 2'b00 and register SPI_CFG3[3] is set to 1'b1, the SPI master works in single mode and half duplex mode. If register SPI_CFG3[2] is set to 1'b0, the SPI master transfers data to SPI slave without any data being transferred from SPI slave to SPI master. See Figure 7-14 for the waveform on the SPI pins. If register SPI_CFG3[2] is set to 1'b1, the SPI master receives data from SPI slave without any data being transferred from SPI master to SPI slave. See Figure 7-15 for the waveform on the SPI pins. Configure the CMD length and ADDR length in register SPI_CFG3. If the CMD length is 0, the ADDR length is also forced to 0. In this case, only data are transferred. The CMD length and ADDR length are not counted in packet_length in register SPI_CFG1.

7.2.4.1.3 SPI Dual Mode and Half Duplex Mode Transaction

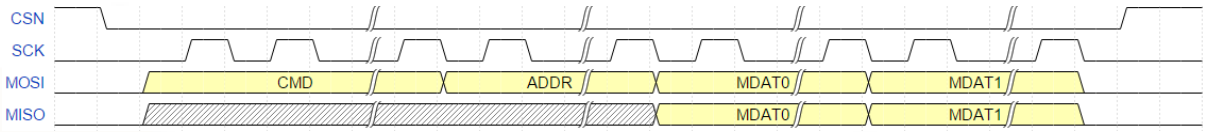


Figure 7-16. SPI Dual Mode, Half Duplex Mode, TX Transaction Formats

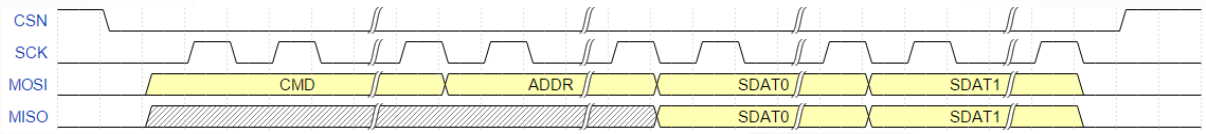


Figure 7-17. SPI Dual Mode, Half Duplex Mode, RX Transaction Formats

If register SPI_CFG3[1:0] is set to 2'b01 and register SPI_CFG3[3] is set to 1'b1, the SPI master works in dual mode and half duplex mode. If register SPI_CFG3[2] is set to 1'b0, the SPI master transfers data to SPI slave without any data being transferred from SPI slave to SPI master. See Figure 7-16 for the waveform on the SPI pins. If register SPI_CFG3[2] is set to 1'b1, the SPI master receives data from SPI slave without any data being transferred from SPI master to SPI slave. See Figure 7-17 for the waveform on the SPI pins.

7.2.4.1.4 SPI Quad Mode and Half Duplex Mode Transaction

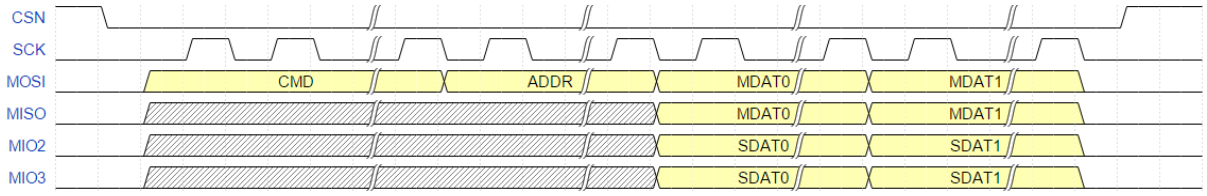


Figure 7-18. SPI Dual Mode, Full Duplex Mode Transaction Format

If register SPI_CFG3[1:0] is set to 2'b01 and register SPI_CFG3[3] is set to 1'b0, the SPI master works in dual mode and full duplex mode. See Figure 7-16 for the waveform on the SPI pins. The SPI master needs two more pins if it works in dual mode and full duplex mode.

7.2.4.1.5 SPI Quad Mode and Half Duplex Mode Transaction

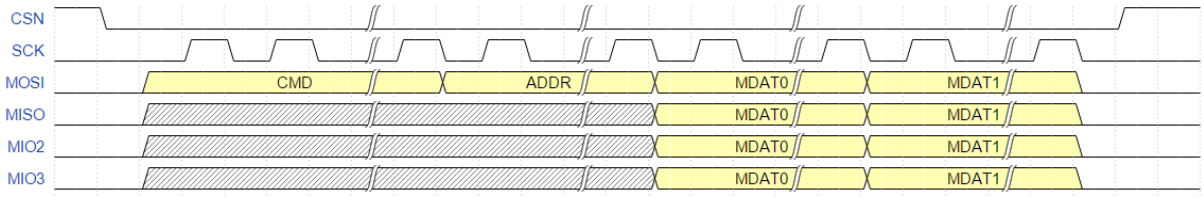


Figure 7-19. SPI Quad Mode, Half Duplex Mode, TX Transaction Formats

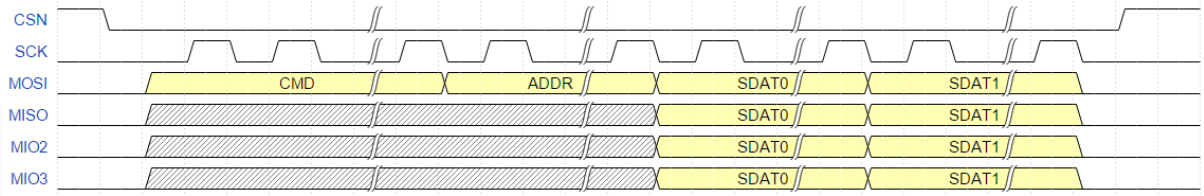


Figure 7-20. SPI Quad Mode, Half Duplex Mode, RX Transaction Formats

If register SPI_CFG3[1:0] is set to 2'b10 and register SPI_CFG3[3] is set to 1'b1, the SPI master works in quad mode and half duplex mode. If register SPI_CFG3[2] is set to 1'b0, the SPI master transfers data to SPI slave without any data being transferred from SPI slave to SPI master. See Figure 7-19 for the waveform on the SPI pins. If register SPI_CFG3[2] is set to 1'b1, the SPI master receives data from SPI slave without any data being transferred from SPI master to SPI slave. See Figure 7-20 for the waveform on the SPI pins.

7.2.4.2 SPI Master Bus Busy Status and Transaction Complete Interrupt

The SPI master provides busy status of the bus, BUSY (SPIn Base address+0x0020) [0]. When BUSY (SPIn Base address+0x0020) [0] = 0, the SPI transaction is presently under way, and software should not try to start a new SPI transaction. On the other hand, when BUSY (SPIn Base address+0x0020) [0] = 1, no SPI transaction is ongoing, and software might start a new SPI transaction. Software can confirm the SPI bus status by polling the spi_busy bit before a new transaction.

If the transaction is completed and interrupt FINISH_IE (SPIn Base address+0x0018) [16] is enabled, the SPI master sets FINISH (SPIn Base address+0x001C) [0] and asserts an SPI interrupt to notify software that the SPIM transaction is completed.

If both PAUSE mode PAUSE_EN (SPIn Base address+0x0018) [4] and PAUSE interrupt PAUSE_IE (SPIn Base address+0x0018) [17] are enabled, the SPI master sets PAUSE (SPIn Base address+0x001C) [1] and asserts an SPI interrupt to notify software that the SPIM transaction is completed. At this moment, the SPI master is in PAUSE_IDLE state and ready to receive the resume command by RESUME (SPIn Base address+0x0018) [1].

7.2.5 Programming Guide

7.2.5.1 SPI DMA Mode

Table 7-4. SPI Master DMA Mode Guide

Step	Sequence	REG_Name	REG_Value	Address
1	Basic configuration	SPI_CFG0 SPI_CFG1 SPI_CFG3	USER_DEFINE	SPIn Base Addr+0x00[31:0] SPIn Base Addr+0x04[31:0] SPIn Base Addr+0x40[31:0]
2	Configure DMA address (max. 36-bit address supported)	SPI_TX_SRC SPI_TX_EXT_ADDR SPI_RX_DST SPI_RX_EXT_ADDR	USER_DEFINE	SPIn Base Addr+0x08[31:0] SPIn Base Addr+0x2C[3:0] SPIn Base Addr+0x0C[31:0] SPIn Base Addr+0x30[3:0]
3	Enable DMA mode	SPI_CMD	2'b11	SPIn Base Addr+0x18[11:10]
4	Set interrupt enable	SPI_CMD	USER_DEFINE	SPIn Base Addr+0x18[17:16]
5	Configure SPI transaction format	SPI_CMD	USER_DEFINE	SPIn Base Addr+0x18[9:3] SPIn Base Addr+0x18[15:12] SPIn Base Addr+0x18[24:18]
6	Trigger DMA	SPI_CMD	USER_DEFINE	SPIn Base Addr+0x18[1:0]
7	CPU waits for interrupt	-	-	-
8	Clear interrupt flag	Read SPI_IRQ[0] or when in pause mode, read SPI_IRQ[1] to clear interrupt	USER_DEFINE	SPIn Base Addr+0x1C[1:0]
9	Get data received from buffer	Read data starting from "destination address"	-	-

7.2.5.2 SPI FIFO Mode

Table 7-5. SPI Master FIFO Mode Guide

Step	Sequence	REG_Name	REG_Value	Address
1	Basic configuration	SPI_CFG0 SPI_CFG1 SPI_CFG3	USER_DEFINE	SPIn Base Addr+0x00[31:0] SPIn Base Addr+0x04[31:0] SPIn Base Addr+0x40[31:0]
2	Write data into TX FIFO	Write to SPI_TX_DATA	USER_DEFINE	SPIn Base Addr+0x10[31:0]
3	Set interrupt enable	SPI_CMD	USER_DEFINE	SPIn Base Addr+0x18[17:16]
3	Configure SPI transaction format	SPI_CMD	USER_DEFINE	SPIn Base Addr+0x18[9:3] SPIn Base Addr+0x18[15:12] SPIn Base Addr+0x18[24:18]
4	Trigger FIFO	SPI_CMD	USER_DEFINE	SPIn Base Addr+0x18[1:0]
5	CPU waits for interrupt	-	-	-
6	Clear interrupt flag	Read SPI_IRQ[0] or when in pause mode, read SPI_IRQ[1] to clear interrupt	USER_DEFINE	SPIn Base Addr+0x1C[1:0]
7	Get data received from RX FIFO	Read from SPI_RX_DATA	-	SPIn Base Addr+0x14[31:0]

7.2.6 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.3 NAND Flash Interface (NFI)

7.3.1 Introduction

The NFI and ECC (Error Correction Code) engine (in NFI mode) can automatically generate ECC syndrome bits when programming or reading the device. If you approve of the way NFI stores syndrome bits in the spare area for each page, the HW_ECC mode can be used. Or else, you can prepare the data, which may contain operating system information or ECC syndrome bits, for the spare area with another arrangement. In former cases, the NFI and ECC engine (in NFI mode) check the syndrome bits when reading from the device. The ECC module features BCH code, which is capable of correcting up to 16-bit errors within one sector.

7.3.2 Features

The SPI NAND flash interface supports the following features:

- ECC (BCH code) acceleration capable of 24-bit error correction (with ECC engine)
- Programmable page size and spare size
- Programmable FDM data size and protected FDM data size
- Word or byte access through APB
- DMA for massive data transfer
- Latch sensitive interrupt to indicate the ready state for read, program and erase operations
- Programmable wait states, command/address setup and hold time, read enable hold time and write enable recovery time
- Support for 1-chip selection for SPI NAND flash parts
- Support for X4/X2/Quad/Dual mode
- Support for device clock, sample clock, data skew adjustment

7.3.3 Block Diagram

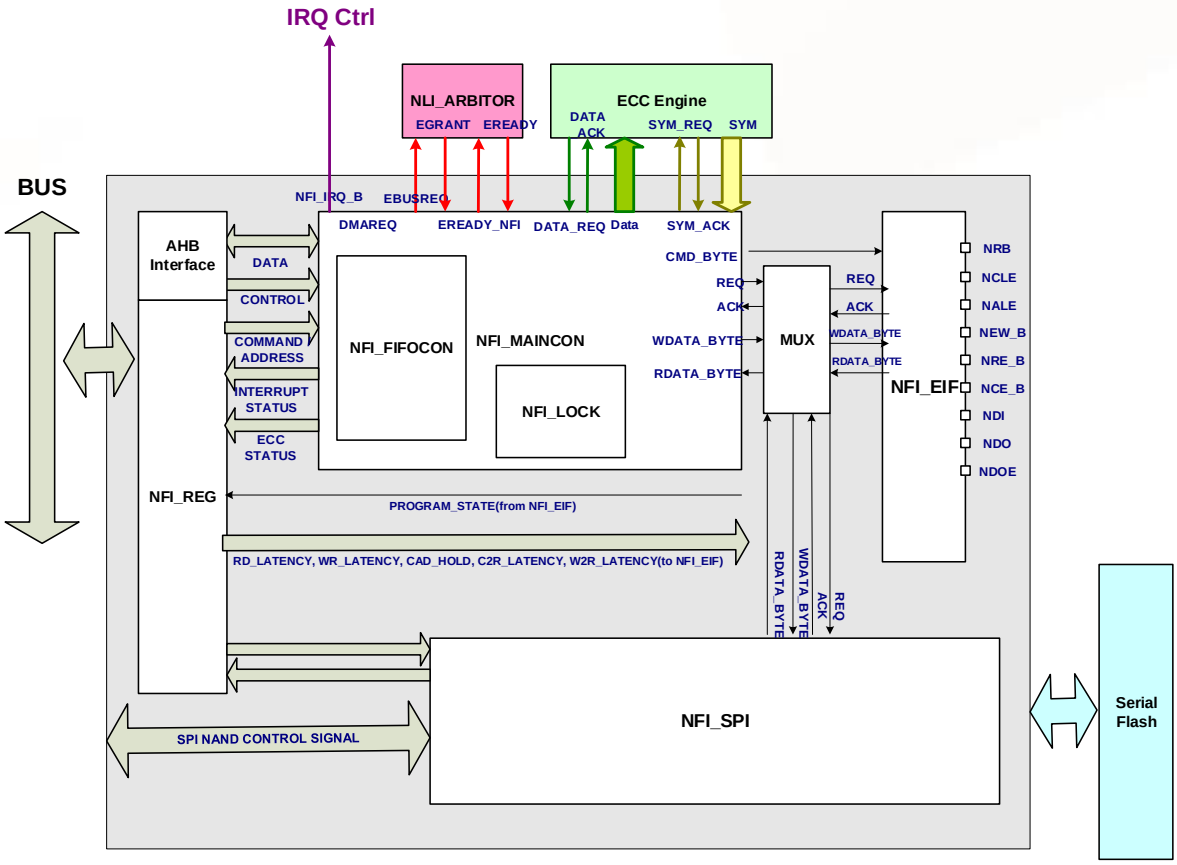


Figure 7-21. NFI Block Diagram

NAND flash controller uses the APB slave bus for accessing register configuration and data read/write, and uses the AHB master bus for faster data read/write. It also supports interrupts that are level active for the interrupt process. The ECC engine is used for encoding and decoding user data when needed. The NAND flash controller uses the standard protocol for communication with NAND devices.

7.3.4 Programming Guide

This section lists the programming sequences of the SPI NAND flash operations.

7.3.4.1 Read ID (MAC Mode)

Configuration	Memo
*CMD_SNF_MISC_CTL = 0x0400010a	// reg_latch_ltc=1 for 104 MHz
*CMD_SNF_MAC_CTL = 0x8	// MAC_mode=1
*SPI_GPRAM_ADDR = 0x9f	-
*CMD_SNF_MAC_OUTL = 0x2	-
*CMD_SNF_MAC_INL = 0x2	-
*CMD_SNF_MAC_CTL = 0xc	// MAC_mode trigger
Polling *CMD_SNF_MAC_CTL	// Poll WIP to 0
*CMD_SNF_MAC_CTL = 0x0	// MAC_mode=0; tigger = 0
int gpram_data = *SPI_GPRAM_ADDR	// Read gpram data (ID)

7.3.4.2 Set Features (MAC Mode)

Configuration	Memo
*CMD_SNF_MISC_CTL = 0x0400010a	// reg_latch_ltc=1 for 104 MHz
*CMD_SNF_MAC_CTL = 0x8	// MAC_mode=1
*SPI_GPRAM_ADDR = 0xa01f	// Clear BP0~ BP3 for unlocking all addresses // Based on Micro spec
*CMD_SNF_MAC_OUTL = 0x3	-
*CMD_SNF_MAC_INL = 0x0	-
*CMD_SNF_MAC_CTL = 0xc	// MAC_mode trigger
Polling *CMD_SNF_MAC_CTL	// Poll WIP to 0
*CMD_SNF_MAC_CTL = 0x0	// MAC_mode=0; tigger = 0

7.3.4.3 Write Enable (MAC Mode)

Configuration	Memo
*CMD_SNF_MAC_CTL = 0x8	// MAC_mode=1
*SPI_GPRAM_ADDR = 0x6	-
*CMD_SNF_MAC_OUTL = 0x1	// Clear BP0~ BP3 for unlocking all addresses // Based on Micro spec
*CMD_SNF_MAC_INL = 0x0	-
*CMD_SNF_MAC_CTL = 0xc	// MAC_mode trigger
Polling *CMD_SNF_MAC_CTL	// Poll WIP to 0
*CMD_SNF_MAC_CTL = 0x0	// MAC_mode=0; tigger = 0

7.3.4.4 Auto Block Erase (Auto Mode)

Configuration	Memo
*CMD_SNF_ER_CTL2 = 0x542310	-
*CMD_SNF_GF_CTL3 = 0xf0020	-
*CMD_SNF_MISC_CTL = 0x10a	// reg_latch_ltc=1 for 104 MHz
*CMD_SNF_ER_CTL = 0xd801	// Auto erase trigger
Polling *CMD_SNF_MAC_CTL1[24]	// Poll status or wait for interrupt
*CMD_SNF_ER_CTL = 0xd800	// Close auto erase trigger

7.3.4.5 Auto Program Load (Auto Mode)

Configuration	Memo
*CMD_SNF_PG_CTL1 = 0x00100206	// Program flow command
*CMD_SNF_PG_CTL2 = 0x0	// Program load address
*CMD_SNF_PG_CTL3 = 0x0	// Program execute address
*CMD_SNF_MISC_CTL = 0x10a	-
*CMD_SNF_MISC_CTL2 = 0x8400840	-
*CMD_SNF_GF_CTL3 = 0xf000a	-
// Setting NFI part	-
*NFI_CON = 0x3	// NFI Reset
*NFI_CNFG = 0x3005	// 0x3305 if autofmt, and ecc is enabled
*NFI_STRADDR = pointer to buffer array	-
*NFI_CMD = 0x80	// Set dummy command
*NFI_STRDATA = 0x1	// Set to 1 when using op_mode = custom mode
*NFIECC_ENCCNFG = 0x1000_0010	// ECC related setting; can be ignored if ECC is disabled
*NFIECC_ENCCON = 0x0	-
*NFIECC_DECCON = 0x0	-
*NFIECC_ENCCON = 0x1	-
	-
*NFI_FDMXX = fdm_data	// Set FDM data. This can be removed if autofmt is disabled
	-
*NFI_CON = 0x4200	// Trigger burst write and trigger SPI
*NFI_INTR_EN = 0x40	// Interrupt enable
Waiting for ahb done interrupt	-
Polling *CMD_SNF_MAC_CTL1[26]	// Wait for auto program done

7.3.4.6 Auto Read Mode (Auto Mode)

Configuration	Memo
*CMD_SNF_MISC_CTL = 0x10a	-
*CMD_SNF_GF_CTL3 = 0xf000a	-
// Setting NFI part	-
*NFI_CON = 0x3	// Reset NFI register status
*NFI_PAGEFMT = 0x2	// Set pagefmt
*NFI_CNFG = 0x631f	// Custom mode is a must for SPI_NAND (read_mode is prohibited)
*NFI_CMD = 0	// Dummy command to trigger the state to custom mode
*NFI_STRDATA = 0x1	// Set to 1 when using op_mode = custom mode
*NFIECC_DECCNFG = 0x90343110	// ECC related setting; can be ignored if ECC is disabled
*NFIECC_DECCON = 0x0	-
*NFIECC_ENCCON = 0x0	-
*NFIECC_DECCON = 0x1	-
*NFIECC_ENCCNFG = 0x1000_0010	-
*NFIECC_ENCCON = 0x0	-
*NFIECC_DECCON = 0x0	-
*NFIECC_ENCCON = 0x1	-
Handling *NFI_FDMXX data	// Check FDM data. The data does not exist here if autofmt is disabled
*NFI_STRADDR = pointer to buffer array	-
*NFI_CON = 0x4100	// Trigger to start transferring data
*NFI_INTR_EN = 0x40	// Interrupt enable
Waiting for ahb done interrupt	-
Polling *CMD_SNF_MAC_CTL1[25]	// Wait for auto read done interrupt

7.3.5 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.4 Inter-Integrated Circuit (I2C)

7.4.1 Introduction

Inter-Integrated Circuit (I2C) controller is a two-wire serial interface. The two signals are SCL and SDA. Both SCL and SDA are bi-directional data signals that can be driven by either the master or the slave. This generic controller supports the master role and conforms to the I2C specifications.

7.4.2 Features

The features of I2C are as follows:

- I2C compliant master mode operation
- Adjustable clock speed for Standard mode operation
- Slave clock extension
- Manual transfer mode
- Multi-write per transfer
- Multi-read per transfer
- Multi-transfer per transaction
- Combined format transfer with length change capability
- Active drive and wired AND I/O configuration
- Repeated start multiple transfers

7.4.2.1 Manual Transfer Mode

The controller offers the manual mode. When the manual mode is selected, in addition to the slave address register, the controller has a built-in 16-byte deep FIFO, which allows MCU to prepare bytes of data for a write transfer, or to read bytes of data for a read transfer.

7.4.2.2 Transfer Format Support

This controller is designed to be as generic as possible in order to support a wide range of devices that may utilize different combinations of transfer formats. Figure 7-22 to Figure 7-24 illustrate the transfer format types supported through different software configurations.

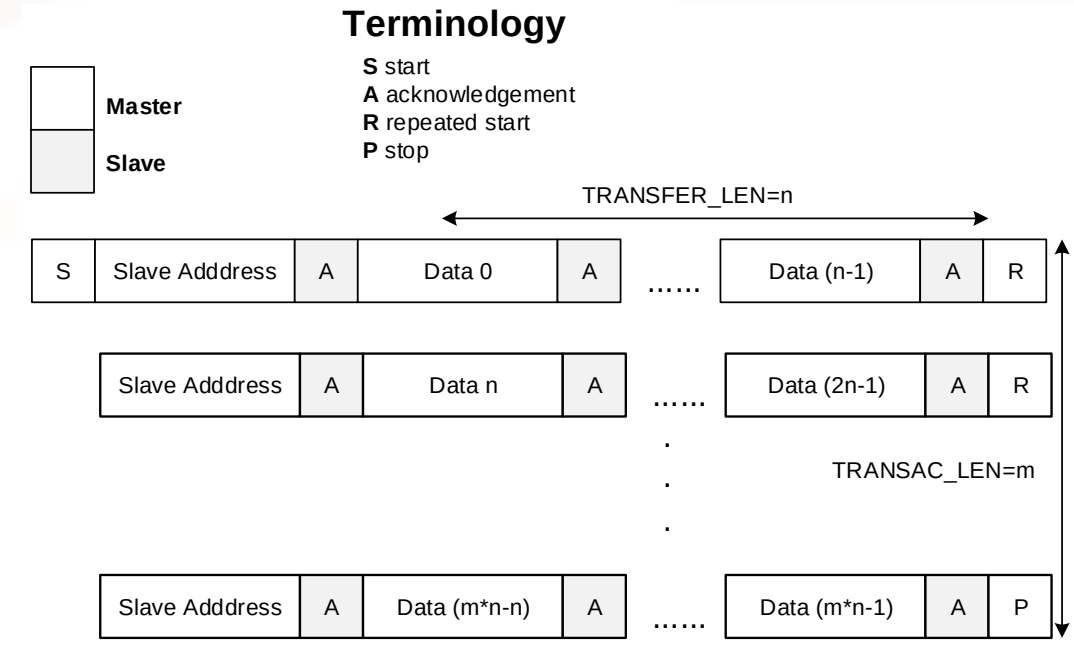


Figure 7-22. I2C Transaction (Write)

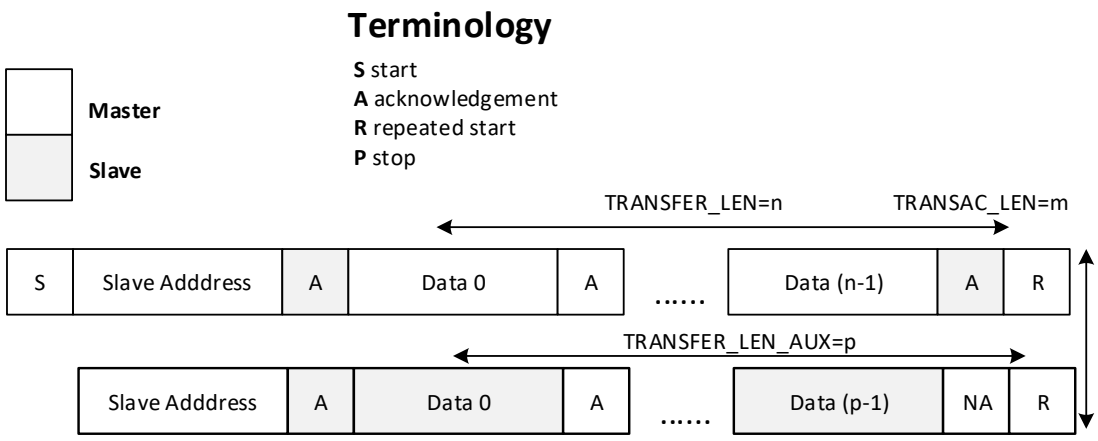


Figure 7-23. I2C Transaction (Read After Write, m = 2)

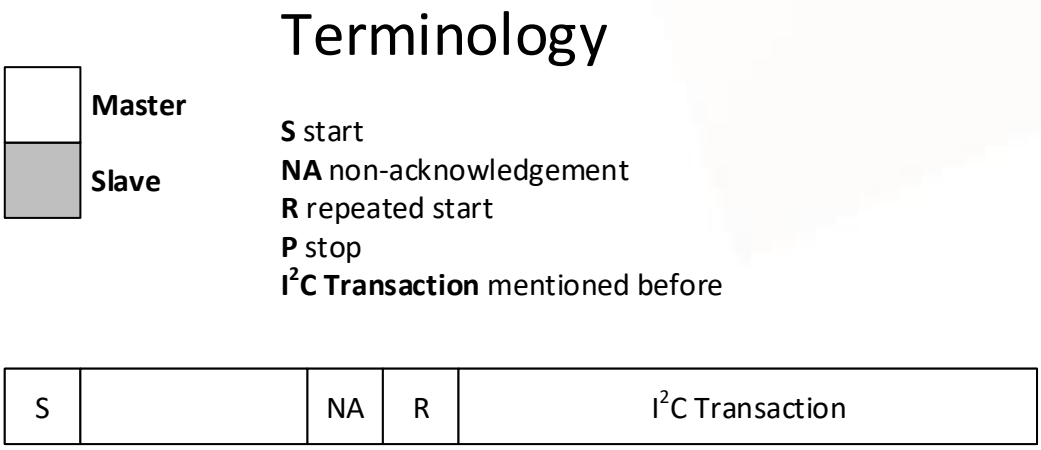


Figure 7-24. I2C High Speed Mode

7.4.3 Block Diagram

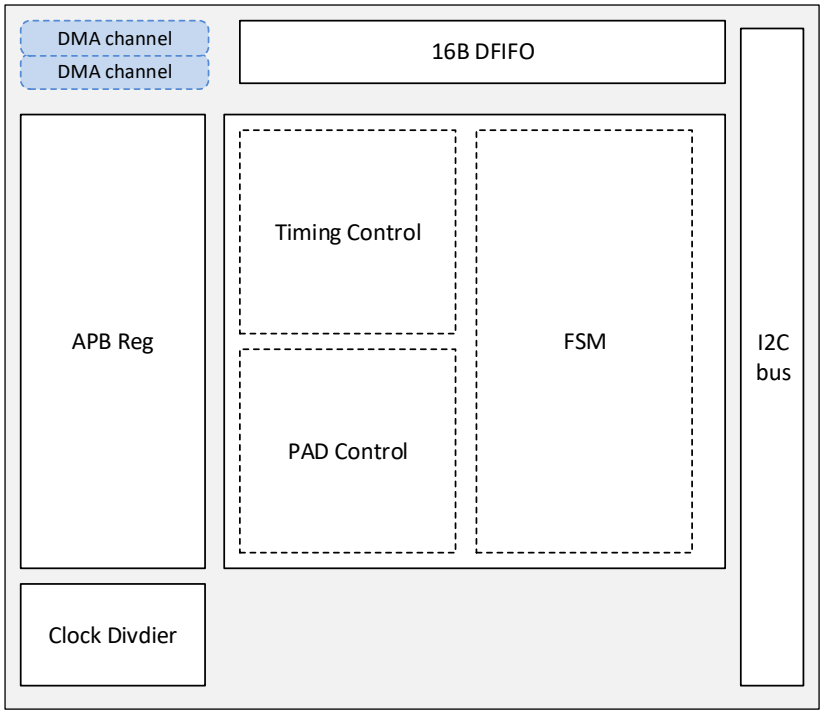


Figure 7-25. Block Diagram of I2C

7.4.4 AC Timing Characteristics

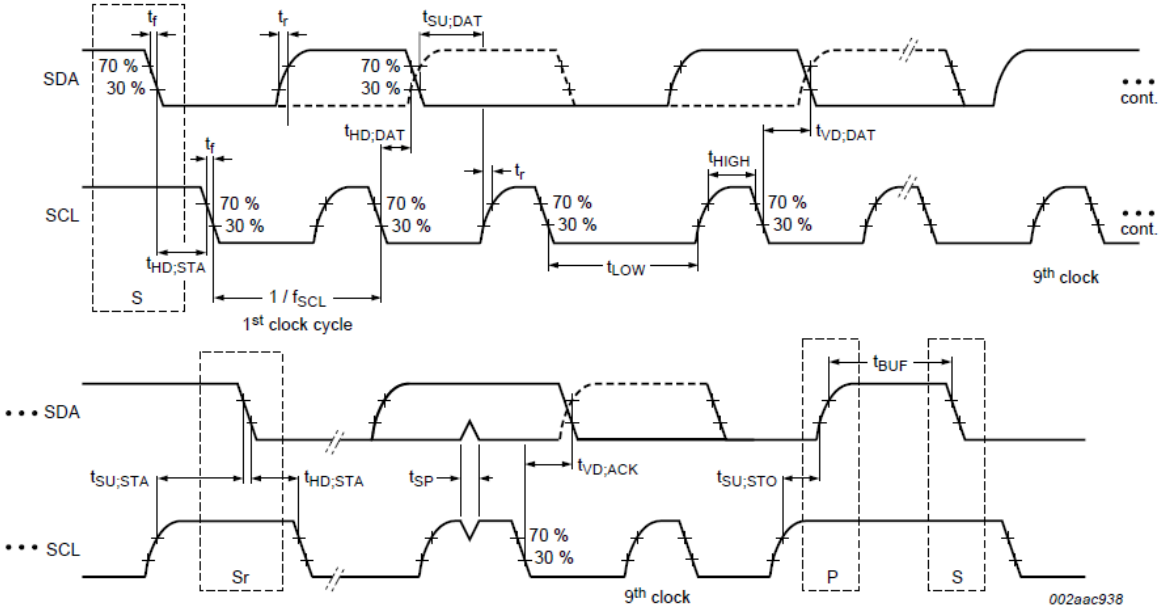


Figure 7-26. I2C AC Timing Diagram of Standard-mode

Table 7-6. I2C AC Timing Parameters for Standard-mode

Symbol	Parameter	Standard-mode		Unit	Note
		Min	Max		
fSCL	SCL clock frequency	0	400	kHz	-
tHD;STA	Hold time (repeated) START condition	4.0	-	µs	-
tLOW	LOW period of the SCL clock	4.7	-	µs	-
tHIGH	HIGH period of the SCL clock	4.0	-	µs	-
tSU;STA	Set-up time for a repeated START condition	4.7	-	µs	-
tHD;DAT	Data hold time	5.0	-	µs	I2C-bus devices
tSU;DAT	Data set-up time	250	-	ns	-
tr	Rise time of both SDA and SCL signals	-	1000	ns	-
tf	Fall time of both SDA and SCL signals	-	300	ns	VDD is I2C IO voltage.
tSU;STO	Set-up time for STOP condition	4.0	-	ns	-
tVD;DAT	Data valid time	-	3.45	µs	-
tVD;ACK	Data valid acknowledge time	-	3.45	µs	-

7.4.5 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.5 Universal Asynchronous Receiver/Transmitter (UART)

7.5.1 Introduction

The Universal Asynchronous Receiver/Transmitter (UART) provides full-duplex serial communication channels between the chip and external devices. The UART supports M16C450 and M16550A operation modes, which are compatible with a range of standard software drivers. The extensions are designed to be broadly software compatible with 16550A variants, but certain areas offer no consensus.

In common with M16550A, the UART supports word lengths from 5 to 8 bits, an optional parity bit and one or two stop bits, and this word length is fully programmable by a CPU interface. A 16-bit programmable baud rate generator and an 8-bit scratch register are included, along with separate transmission and received FIFOs. Eight modem control lines and a diagnostic loopback mode are provided. The UART also includes two Direct Memory Access (DMA) handshake lines which are used to indicate when the FIFOs are ready to transfer data to CPU. Interrupts can be generated from any of the several sources.

After hardware reset, the UART is in M16C450 mode. Its FIFOs can be enabled and the UART can enter M16550A mode. The UART adds further functionality beyond M16550A mode. Each of the extended functions can be selected individually via software control.

7.5.2 Features

The features of UART are as follows:

- 3 channels of UARTs
- UART0 are 2-pin (TX, RX), UART1 to UART2 are 4-pin (TX, RX, CTS, RTS) UART channels.
- M16C450 and M16550A operation modes
- Compatible with standard software drivers
- Transfer system: Asynchronous
- Data length: 5 to 8 bits
- Hardware flow control: CTS/RTS-based automatic transmission and reception of control
- Software flow control: Use special characters, XON and XOFF, to do software flow control
- Baud rate is programmable from 300 bps to 3 Mbps.
- Baud rate error: Less than 0.25 %
- Interrupt request: Receive interrupts and transmit interrupts
- Data transfer: DMA (Transmit/Receive) transfer is supported.

7.5.3 Block Diagram

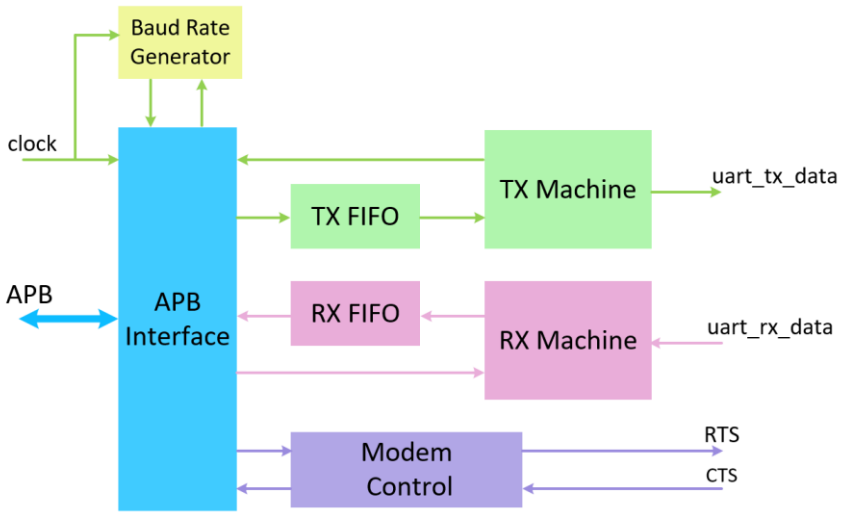


Figure 7-27. Block Diagram of UART

Figure 7-27 shows the block diagram of UART. It consists of the First In First Out (FIFO), the Finite State Machine (FSM), APB interface and Modem Control. To support full-duplex serial communication, UART has TX and RX channels, and each channel contains an FSM (TX FSM and RX FSM) and a 32-byte FIFO (TX FIFO and RX FIFO). The FSM is used to indicate the current transfer stage of TX and RX channel. The TX FIFO and RX FIFO store the data to be sent or the data received from outside. Through APB interface, the system can access the configuration and status registers of UART, read the received data or write the data to be sent. UART can support the function of hardware flow control through Modem Control. The UART signal descriptions are included in Table 7-7.

Table 7-7. Signal Descriptions

Signal Name	Signal Type	Description
uart_tx_data	Output	Serial data transmit
uart_rx_data	Input	Serial data receive
RTS	Output	Request to Send handshaking signal
CTS	Input	Clear to Send handshaking signal

7.5.4 Communication Protocol

UART communication protocol is as follows:

- 1-bit start bit must be low.
- The data bit length is 5 to 8 bits.
- The parity check can be odd parity or even parity.
- The stop bit length is 1 to 2 bits. It must be high.

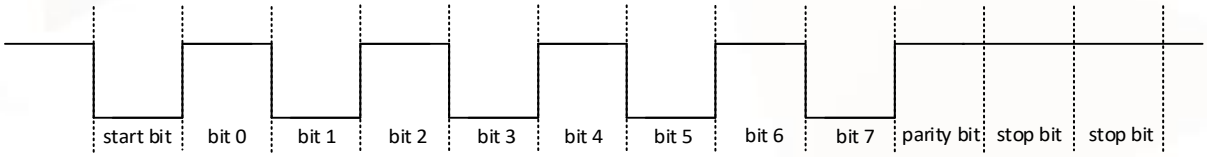


Figure 7-28. UART Communication Protocol

7.5.5 Theory of Operations

UART Enhancement Features

The UART provides more powerful enhancements than the industry-standard 16550:

Hardware flow control: Use two dedicated signals, Clear to Send (CTS) and Request to Send (RTS) signals, to indicate UART is ready to get data or send data. When CTS is low, UART can start to transmit data. As long as CTS is active, UART is not allowed to send data. RTS going low means UART FIFO in received circuit is sufficient to receive data. UART is not allowed to receive data when RTS is high. This feature is very useful when the Interrupt Service Routine (ISR) latency is hard to predict and control in the embedded applications. The MCU is relieved of having to fetch the received data within a fixed amount of time.

Software flow control: Use special characters XON and XOFF to do software flow control. Special characters XON and XOFF are software programmable. When XOFF is received, UART transmission is halted. The transmission will not be resumed until XON is received.

Note:
In order to enable any of the enhancements, the enhanced mode bit, EFR[4], must be set. If EFR[4] is not set, IER[7:5], FCR[5:4], and MCR[7:6] cannot be written. The Enhanced Mode bit ensures that the UART is backward compatible with software that has been written for 16C450 and 16550A devices.
When the oversampling ratio between UART clock and baud rate is less than 8, it is necessary to enable guard time function in customer's UART TX device to make MediaTek UART RX work properly. Otherwise, frame errors can happen and the received data can get corrupted.

UART Interrupt

UART generates several interrupts. The interrupt types are shown in Table 7-8.

Table 7-8. UART Interrupt Control Bits and Interrupt Factors

UARTn+0004h Interrupt Enable Register UARTn_IER

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CTSI	RTSI	XOFFI		EDSSI	ELSI	ETBEI	ERBFI
Type									R/W							
Reset									0							

IER[3:0] are modified when LCR[7] = 0.
IER[7:4] are modified when LCR[7] = 0&EFR[4] = 1.

UARTn+0008h Interrupt Identification Register UARTn_IIR

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									FIFOE		ID4	ID3	ID2	ID1	ID0	NINT
Type									RO							
Reset									0	0	0	0	0	0	0	1

ID4 and ID3 are presented only when EFR[4] = 1.

Table 7-9. UART Interrupt Types

Interrupt Type	Interrupt Request Bit (UARTn_IER)	Interrupt Identification (UARTn_IIR)	Interrupt Factor	Note
Received	ERBFI	IIR[5:0] = 000100b	RX Buffer contains data.	-
	-	IIR[5:0] = 001100b	Timeout on character in RX FIFO	-
Transmit	ETBEI	IIR[5:0] = 000010b	TX Holding Register is empty or the contents of the TX FIFO have been reduced to its trigger level.	-
Communication Error	ELSI	IIR[5:0] = 000110b	When frame error, parity error, break interrupt or FIFO overrun happens, the interrupt will be generated.	For the detailed interrupt status, please refer to LSR[4:1].
Modem	EDSSI	IIR[5:0] = 000000b	Modem status change	For the detailed interrupt status, please refer to MSR[4:1].
Enhancement Feature	CTSI	IIR[5:0] = 100000b	When a rising edge is detected on the CTS, the interrupt will be generated.	Available when enhanced feature is enabled (EFR[4] = 1)
	RTSI	IIR[5:0] = 100000b	When a rising edge is detected on the RTS, the interrupt will be generated.	
	XOFF1	IIR[5:0] = 010000b	When an XOFF character is received, the interrupt will be generated.	

Data Transmission

If the TX Holding Register (THR) is empty (FIFOs are disabled) or TX FIFO is reduced to its trigger level (FIFOs are enabled), TX Holding Register Empty (THRE) bit of the LSR is “1” and the transmitted data can be written in the THR.

When THR is not empty (FIFOs are disabled) or TX FIFO is increased to its trigger level (FIFOs are enabled), TX starts transmission automatically.

Software can write transmitted data into THR by asserting transmit interrupt (IIR[5:0] = 000010b) or polling the THRE bit status as “1” directly.
If FIFOs are enabled, the transmitted data can be written into the THR. The data will be transferred to TX FIFO directly.

Data Reception

If the RX Buffer is becoming full or a byte is being transferred into RX FIFO, the DR bit of the LSR is “1” and the received data can be read by RX Buffer Register (RBR).

Software can read the received data when receive interrupt (IIR[5:0] = 000100b) is asserted or UART is polling the DR bit status directly.

If FIFOs are enabled, the received data in the RX FIFO can be read by reading RBR.

Register Description

UART_BASE:

UART0 register base address is 0x1100_2000.

UART1 register base address is 0x1100_3000.

UART2 register base address is 0x1100_4000.

Table 7-10. UART Register Map

Address	Name	Description
UART_BASE+0x0C	LCR	Line Control Register (LCR)
UART_BASE+0x24	HIGHSPEED	HIGH SPEED UART
UART_BASE+0x28	SAMPLE_COUNT	SAMPLE_COUNT
UART_BASE+0x2C	SAMPLE_POINT	SAMPLE_POINT
UART_BASE+0x34	RATEFIX_AD	Rate Fix Address
UART_BASE+0x3C	GUARD	Guard Time Added Register
UART_BASE+0x40	ESCAPE_DAT	Escape Character register
UART_BASE+0x44	ESCAPE_EN	Escape Enable Register
UART_BASE+0x48	SLEEP_EN	Sleep Enable Register
UART_BASE+0x4C	VFIFO_EN	Virtual FIFO Enable Register
UART_BASE+0x50	RXTRI_AD	RX Trigger Address
UART_BASE+0x54	FRACDIV_L	Fractional Divider LSB Address
UART_BASE+0x58	FRACDIV_M	Fractional Divider MSB Address
UART_BASE+0x5C	FCR_RD	FIFO Control Register
UART_BASE+0x60	TX_ACTIVE_EN	TX Active Enable Address

	Condition: LCR[7] == 0		**Condition: LCR[7] == 1**	
Address	**Name**	**Description**	**Name**	**Description**
UART_BASE+0x00	THR RBR	TX Holding Register/ RX Buffer Register	DLL	Divisor Latch (LS)
UART_BASE+0x04	IER	Interrupt Enable Register	DLM	Divisor Latch (MS)
	Condition: LCR != 0xBF		**Condition: LCR == 0xBF**	
Address	**Name**	**Description**	**Name**	**Description**
UART_BASE+0x08	FCR IIR	FIFO Control Register/ Interrupt Identification Register	EFR	Enhanced Feature Register
UART_BASE+0x10	MCR	Modem Control Register	XON1	XON1
UART_BASE+0x14	LSR	Line Status Register	XON2	XON2
UART_BASE+0x18	MSR	Modem Status Register	XOFF1	XOFF1
UART_BASE+0x1C	SCR	Scratch Register	XOFF2	XOFF2

UARTn+0000h Divisor Latch (LS) UARTn_DLL

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DLL[7:0]							
Type									R/W							
Reset									1							

UARTn+0004h Divisor Latch (MS) UARTn_DLM

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DLM[7:0]							
Type									R/W							
Reset									0							

Note:
DLL and DLM can only be updated if LCR[7] is set ("1"). Note that division by 1 generates a BAUD signal that is constantly high. The table below shows the divisor that needs to generate a given baud rate from CLK inputs of 26 MHz.
e.g. clock source: 26 MHz, baud rate: 4800 bps
HIGHSPEED (0x24) = 0 case: 26 MHz/4800/16 ~= 339 = 0x153 → DLM: 0x01, DLL: 0x53
HIGHSPEED (0x24) = 1 case: 26 MHz/4800/8 ~= 677 = 0x2A3 → DLM: 0x02, DLL: 0xA3
HIGHSPEED (0x24) = 2 case: 26 MHz/4800/4 ~= 1354 = 0x54A → DLM: 0x05, DLL: 0x4A
HIGHSPEED (0x24) = 3 case: 26 MHz/4800/256 + 1 ~= 22 = 0x16 → DLM: 0x00, DLL: 0x16

UARTn+0024h HIGH SPEED UART UARTn_HIGHSPEED

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															SPEED [1:0]	
Type															R/W	
Reset															0	

SPEED UART sample counter base
0: Based on 16*baud_pulse, baud_rate = system clock frequency/16/{DLM, DLL}
1: Based on 8*baud_pulse, baud_rate = system clock frequency/8/{DLM, DLL}
2: Based on 4*baud_pulse, baud_rate = system clock frequency/4/{DLM, DLL}
3: Based on sample_count * baud_pulse, baud_rate = system clock frequency/(sample_count+1)/{DLM, DLL}

UARTn+0028h SAMPLE_COUNT UARTn_SAMPLE_COUNT

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SAMPLECOUNT [7:0]							
Type									R/W							
Reset									0							

When HIGHSPEED = 3, the sample_count is the threshold value for UART sample counter (sample_num).
Sample Count = clock source/baud rate/{DLM. DLL} - 1
e.g. clock source: 26 MHz, baud rate: 4800 bps; DLM: 0x00, DLL: 0x16
High Speed (0x24) = 0&1&2 case: No need to set SAMPLE_COUNT
High Speed (0x24) = 3 case: 26 MHz/4800/0x16 - 1 ~= 245 → SAMPLE_COUNT = 245

UARTn+002Ch SAMPLE_POINT UARTn_SAMPLE_POINT

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SAMPLEPOINT [7:0]							
Type									R/W							
Reset									Ffh							

When HIGHSPEED = 3, UART will get the input data when sample_count = sample_num.

The SAMPLE_POINT is usually ROUNDDOWN ((SAMPLE_COUNT+1)/2 - 1).

e.g. clock source: 26 MHz, baud rate: 4800 bps; DLM: 0x00, DLL: 0x16&SAMPLE_COUNT = 245

SAMPLE_POINT = ROUNDDOWN ((245+1)/2 – 1) = 122 (sample the central point to decrease inaccuracy)

UARTn+0054h Fractional Divider LSB Address UARTn_FRACDIV_L

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									FRACDIV_L							
Type									R/W							
Reset									0	0	0	0	0	0	0	0

FRACDIV_L: Add sampling count (+1) from state data7 to state data0 in order to improve fractional divisor.

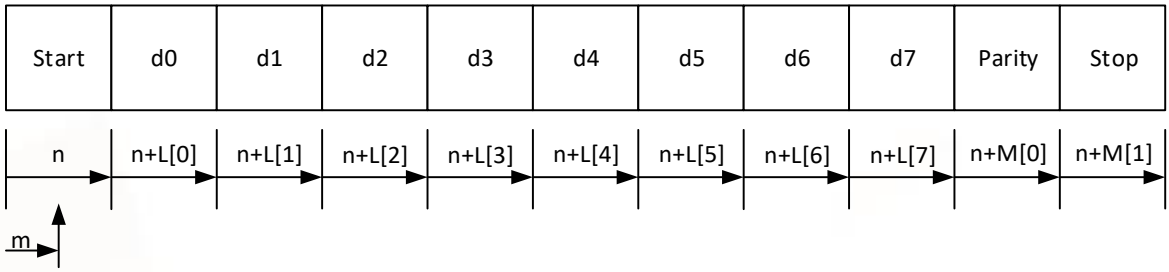
UARTn+0058h Fractional Divider MSB Address UARTn_FRACDIV_M

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															FRACDIV_M	
Type															R/W	
Reset															0	0

FRACDIV_M: Add sampling count in state stop and state parity in order to improve fractional divisor.

FRACDIV_L/FRACDIV_M: Add one sampling period to each symbol in order to increase baud rate accuracy.

bit_extend register = FRACDIV_L[7:0]
FRACDIV_M[1:0]



Bit extend number = ROUND ((clock source/baud rate/{DLM. DLL} – (SAMPLE_COUNT + 1))*10)

e.g. clock source: 26 MHz, baud rate: 4800 bps; DLM: 0x00, DLL: 0x16&SAMPLE_COUNT = 245

Bit extend number = ROUND ((26 MHz/4800/0x16 – (245+1))*10) = 2 → Need to compensate for 2 bits of one frame (e.g. FRACDIV_L = 0x44, FRACDIV_M = 0x00)

Refer to Table 7-11 for details.

Table 7-11. Bit Extend Number Reference

Bit Extend Number	FRACDIV_M	FRACDIV_L
0	0x00	0x00
1	0x00	0x10
2	0x00	0x44
3	0x00	0x92
4	0x01	0x29
5	0x01	0xaa
6	0x01	0xb6
7	0x01	0xdb
8	0x01	0xef
9	0x01	0xff
10	0x03	0xff

7.5.6 Programming Guide

7.5.6.1 Baud Rate Setting and UART Initialization

For suggested UART baud rate setting from clock inputs of 30 MHz, refer to the following table.

Table 7-12. Suggestion for UART Baud Rate Setting

Baud Rate	HIGHSPEED	{DLM, DLL}	SAMPLE_COUNT	SAMPLE_POINT	FRACDIV_M	FRACDIV_L
3,000,000	3	0x00, 0x01	0x09	0x03	0x0	0x0
2,000,000	3	0x00, 0x01	0x0E	0x06	0x0	0x0
1,000,000	3	0x00, 0x01	0x1D	0x0D	0x0	0x0
500,000	3	0x00, 0x01	0x3B	0x1C	0x0	0x0
250,000	3	0x00, 0x01	0x77	0x3A	0x0	0x0
153,600	3	0x00, 0x01	0xC2	0x60	0x0	0x92
115,200	3	0x00, 0x02	0x81	0x3F	0x0	0x44
76,800	3	0x00, 0x02	0xC2	0x60	0x0	0x92
57,600	3	0x00, 0x03	0xAC	0x55	0x1	0xB6
38,400	3	0x00, 0x04	0xC2	0x60	0x0	0x92
28,800	3	0x00, 0x05	0xCF	0x66	0x0	0x92
19,200	3	0x00, 0x07	0xDE	0x6E	0x0	0x44
9,600	3	0x00, 0x0D	0xEF	0x76	0x1	0x92
7,200	3	0x00, 0x11	0xF4	0x79	0x1	0x01
4,800	3	0x00, 0x19	0xF9	0x7B	0x0	0x0

Use Register DLL, DLM, HIGHSPEED, SAMPLE_COUNT, SAMPLE_POINT, FRACDIV_M and FRACDIV_L to set baud rate. After setting the baud rate, UART can start transmission by filling the TX FIFO and receiving data from RX FIFO.

Table 7-13 is an example of generating baud rate 115200 bps by clock inputs of 30 MHz:

Table 7-13. UART Baud Rate Setting Example

Step	Description	Related Register Setting
1	Select UART sample counter base to SPEED 3	HIGHSPEED = 0x3
2	Set sample counter	SAMPLE_COUNT = 0x81 SAMPLE_POINT = 0x3F FRACDIV_L = 0x44 FRACDIV_M = 0x1
3	Switch register to divisor mode (Register MAP condition 2) to do divisor latch setting. UARTn_LCR[7] = 1	LCR = 0x80
4	Set divisor latch	DLL = 0x2 DLM = 0x0
5	Set guard time	GUARD
6	Switch register to normal mode (Register MAP condition 1). UARTn_LCR[7] = 0	LCR = 0x00

Table 7-14. UART Hardware Initialization

Step	Description	Related Register Setting
1	Baud rate setting: please refer to Table 7-13	-
2	Enable enhanced feature (Register is accessible only when LCR = BF'h)	LCR = 0xBF EFR = 0x10 LCR = 0x00
3	Enable FIFO control	FCR
4	Word length (LCR[1:0]), parity (LCR[5:4]), STOP (LCR[2]) bit settings	LCR
5	Enable Interrupt	IER

The suggested ED software programming sequence is shown below.

- DRV_WriteReg32(UART_BASE+0x24, 0x00000003); //high-speed UART
- DRV_WriteReg32(UART_BASE+0x28, 0x00000081); //sample_count
- DRV_WriteReg32(UART_BASE+0x2C, 0x0000003F); //sample_point
- DRV_WriteReg32(UART_BASE+0x4C, 0x00000001); //Enable RX DMA
- DRV_WriteReg32(UART_BASE+0x54, 0x00000044); //FRACDIV_L
- DRV_WriteReg32(UART_BASE+0x58, 0x00000001); //FRACDIV_M
- DRV_WriteReg32(UART_BASE+0x0C, 0x000000BF); //LCR==0xBF, change to Condition 2
- DRV_WriteReg32(UART_BASE+0x00, 0x00000002); //DLL, LS
- DRV_WriteReg32(UART_BASE+0x04, 0x00000000); //DLM, MS
- DRV_WriteReg32(UART_BASE+0x08, 0x00000010); // Enable enhancement features
- DRV_WriteReg32(UART_BASE+0x0C, 0x00000000); //LCR !=0xBF, change to Condition 1
- DRV_WriteReg32(UART_BASE+0x08, 0x00000031); //FIFO trigger threshold and enable FIFO
- DRV_WriteReg32(UART_BASE+0x0C, 0x00000003); //8-bit word length
- DRV_WriteReg32(UART_BASE+0x04, 0x00000001); //Enable RX interrupt

7.5.6.2 Transmission

Data Transmission

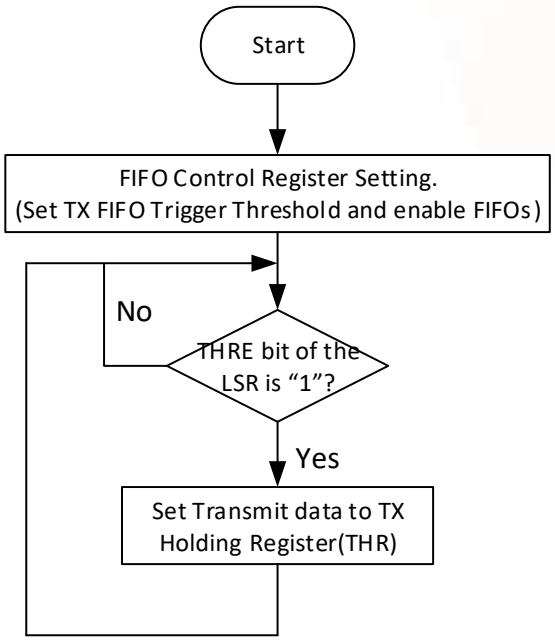


Figure 7-29. UART Data Transmission with THRE Bit Status Polling

FIFO Register Setting:

- Enable FIFO: Set x’08[0] to 1’b1
- TX FIFO Trigger Threshold x’08[5:4]
- 2’b00: 1 byte (down to trigger)
- 2’b01: 4 bytes (down to trigger)
- 2’b10: 8 bytes (down to trigger)
- 2’b11: 14 bytes (down to trigger)

Data Reception

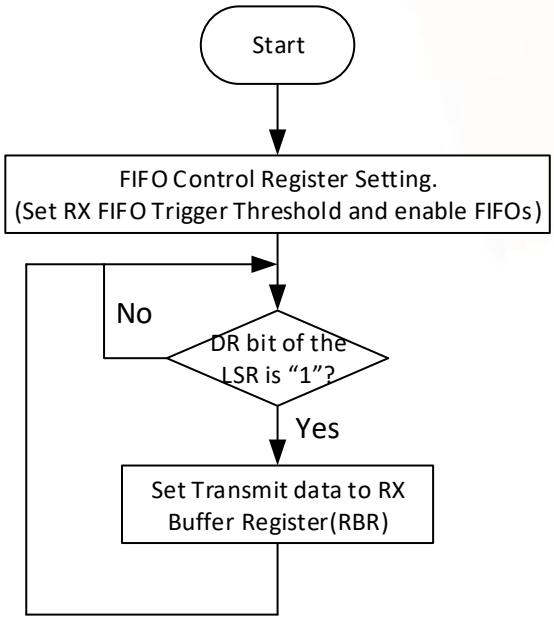


Figure 7-30. UART Data Reception with DR Bit Status Polling

FIFO Register Setting:

- Enable FIFO: Set x'08[0] to 1'b1
- RX FIFO Trigger Threshold x'08[7:6]
- 2'b00: 1 byte (up to trigger)
- 2'b01: 6 bytes (up to trigger)
- 2'b10: 12 bytes (up to trigger)
- 2'b11: register x'50[4:0] (up to trigger)

7.5.7 Register Definition

Refer to "MT7988A Wi-Fi 7 Platform Registers" for detailed register descriptions.

7.6 Pulse Width Modulators (PWMs)

7.6.1 Introduction

Eight generic Pulse Width Modulators (PWMs) are implemented to generate pulse sequences with programmable frequency and duration for LCD backlight, charging or other purposes. Before software enables PWM, the pulse sequences must be prepared in the memory or registers. Then, PWM reads the pulse sequences to generate random waveforms for all kinds of applications (see Figure 7-31).

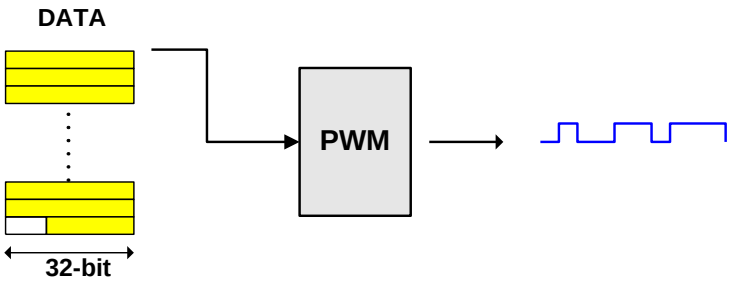


Figure 7-31. Generation Procedure of PWM

7.6.2 Features

The features of PWM are as follows:

- Old mode
- FIFO mode
- Periodical memory and random mode

7.6.3 Block Diagram

Figure 7-32 shows the block diagram of the PWM.

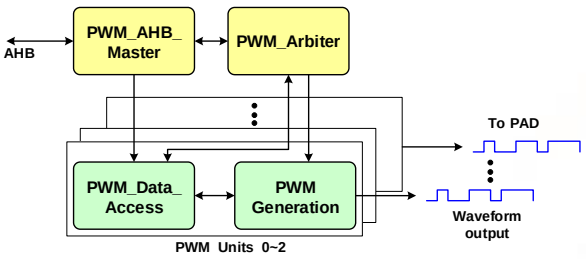


Figure 7-32. PWM Block Diagram

7.6.4 Theory of Operations

7.6.4.1 Old Mode

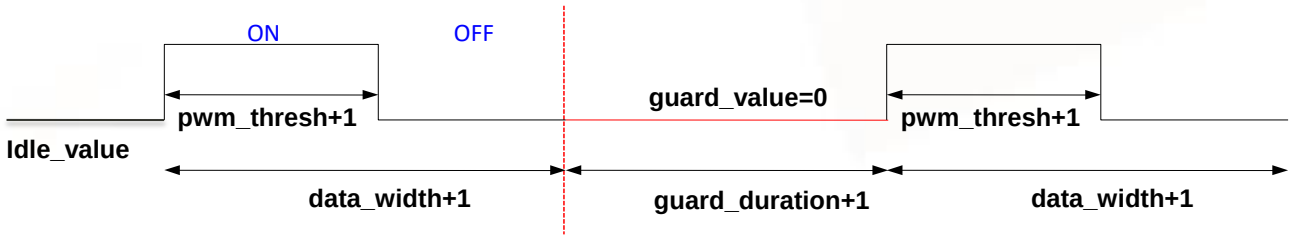


Figure 7-33. Old Mode

In the old mode, the waveform generated by PWM is shown in Figure 7-33. The frequency is determined by PWM_DATA_WIDTH (pwm Base address+0x003C)[12:0] and the duty cycle is determined by PWM_THRESH (pwm Base address+0x0040)[12:0].

$$\text{PWM frequency} = \frac{CLKSRC}{CLKDIV*(DATA\_WIDTH+1)}$$

Without considering guard_duration (pwm Base address+0x001C)[15:0]

$$\text{Duty cycle} = \frac{PWM\_THRESH + 1}{DATA\_WIDTH + 1}$$

guard_ duration (pwm Base address+0x001C)[15:0] is the time interval between two complete waveforms. When PWM_WAVE_NUM (pwm Base address+0x0038)[15:0] = 0, it means that hardware continuously outputs the waveform, and the waveform can only be terminated by disabling PWM.

7.6.4.2 FIFO Mode

If the pulse sequence data are less than or equal to 64 bits, they can be directly set in PWM_SEND_DATA0 (pwm Base address+0x0030)[31:0], PWM_SEND_DATA1 (pwm Base address+0x0034)[31:0] and SRCSEL (pwm Base address+0x0010)[5]=0 to reduce memory bandwidth.

STOP_BITPOS (pwm Base address+0x0010)[14:9] is used to indicate the stop bit position in the total 64-bit data. For example, if STOP_BITPOS (pwm Base address+0x0010)[14:9] is 0, only PWM_SEND_DATA0 (pwm Base address+0x0030)[31:0] is generated.

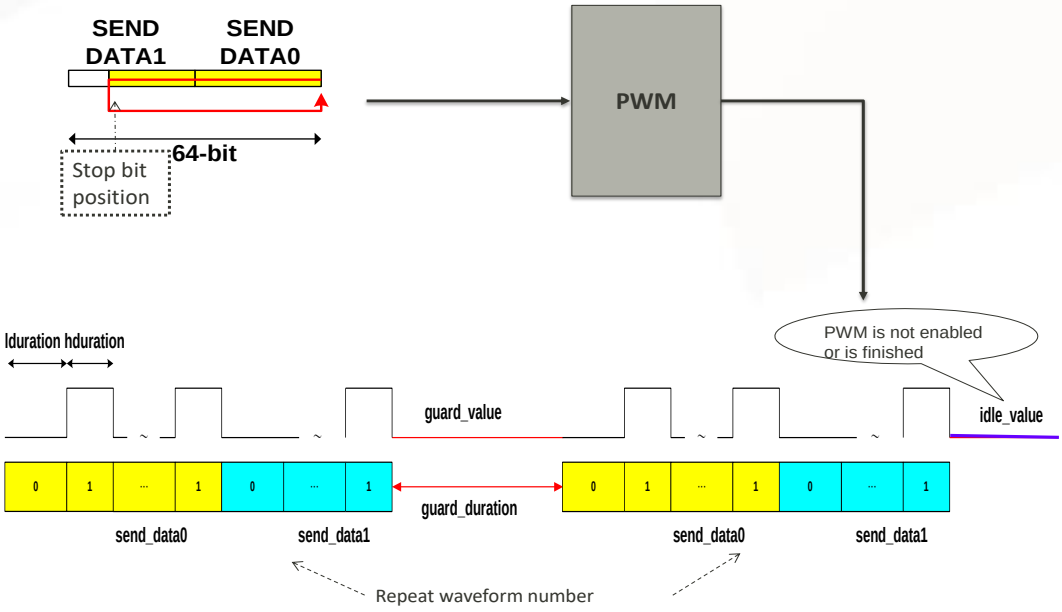


Figure 7-34. FIFO Mode

7.6.4.3 Memory Mode

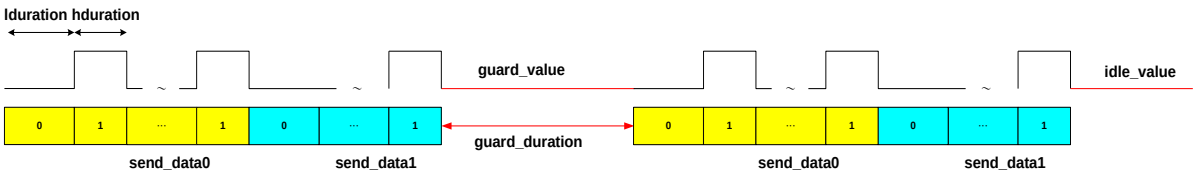


Figure 7-35. Memory Mode

In the periodical mode, all pulse sequences are repeatedly generated by the number of PWM_WAVE_NUM (pwm Base address+0x0038)[15:0]. If PWM_WAVE_NUM (pwm Base address+0x0038)[15:0]=0, hardware continuously outputs waveform, and the waveform generation can be stopped by PWM_ENABLE (PWM Base address + 0x0000)[31:0].

If SRCSEL (pwm Base address+0x0010)[5]=1, PWM is in the memory mode. The pulse sequence data are put in memory with address set by PWM_BUF0_BASE_ADDR (pwm Base address+0x0020)[31:0] and PWM_BUF0_BASE_ADDR2 (pwm Base address+0x004C)[3:0], and the length is PWM_BUF0_SIZE (pwm Base address+0x0024)[15:0]. STOP_BITPOS (pwm Base address+0x0010)[14:9] is to indicate the stop bit position in the last 32-bit data.

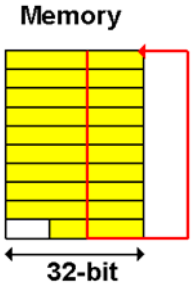


Figure 7-36. Memory Mode and Stop Bit Position

Note:
Any kind of memory can be used by PWM (usually SYSRAM) so long as software can read and write the address. Software issues a request, and PWM can locate an un-used memory and get the address, so you need to write the address and length at PWM_BUF0_BASE_ADDR.

7.6.4.4 Random Mode

On the other hand, the pulse sequence is stored in dual memory buffers in the random mode. The format of pulse sequences stored in the memory is shown in Figure 7-37. The valid bit is used to indicate that the data are ready in the respective memory buffer. The PWM generation clears this bit after all data in that buffer are fetched. The memory buffers are set by address PWM_BUF0_BASE_ADDR (pwm Base address+0x0020)[31:0] and PWM_BUF0_SIZE (pwm Base address+0x0024)[15:0] for memory 0, and are set by address PWM_BUF1_BASE_ADDR (pwm Base address +0x0028)[31:0] and PWM_BUF1_SIZE (pwm Base address+0x002C)[15:0] for memory 1.

The program should prepare for the pulse sequence and set the valid bit to 1 in time before all data in other memory buffers are fetched. Otherwise, the hardware issues the UNDERFLOW interrupt to inform that the pulse generation is stopped since there are no valid data. If the UNDERFLOW interrupt is received, software needs to disable PWM, set valid bit again and enable PWM to restart the pulse generation.

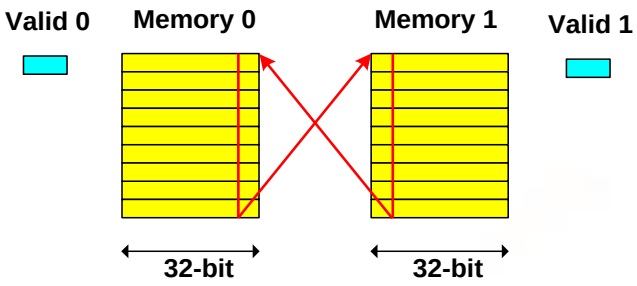


Figure 7-37. Random Mode

Note:
Any kind of memory can be used by PWM (usually SYSRAM) so long as software can read and write the address. Software issues a request, and PWM can locate an un-used memory and get the address, so you need to assign the address and length at PWM_BUF0_BASE_ADDR/PWM_BUF0_SIZE and PWM_BUF1_BASE_ADDR/PWM_BUF1_SIZE.

7.6.5 Programming Guide

7.6.5.1 Old Mode

Table 7-15. Old Mode Setting Procedure

PWM Setting Sequence for Old Mode							
Step	Description	R/W	Address	Bit	MACRO	Value	Note
1	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b0	Disable PWM[N]
2	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[15]	OLD_PWM_MODE	1'b1	-
3	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[8]	GUARD_VALUE	USER_DEFINED	-
4	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[7]	IDLE_VALUE	USER_DEFINED	-
5	Set PWM_WAVE_NUM	W	PWM_BASE + 0x038+ PWM_NUM*0x40	[15:0]	PWM_WAVE_NUM	USER_DEFINED	If WAVE_NUM=0, the waveform generation does not stop until PWM is disabled.
6	Set PWM_GDURATION	W	PWM_BASE + 0x01C+ PWM_NUM*0x40	[15:0]	PWM_GDURATION	USER_DEFINED	-
7	Set PWM_DATA_WIDTH	W	PWM_BASE + 0x3C + PWM_NUM*0x40	[12:0]	PWM_DATA_WIDTH	USER_DEFINED	-
8	Set PWM_THRESH	W	PWM_BASE + 0x40 + PWM_NUM*0x40	[12:0]	PWM_THRESH	USER_DEFINED	-
9	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b1	Enable PWM[N]

7.6.5.2 FIFO Mode

Table 7-16. FIFO Mode Setting Procedure

PWM Setting Sequence for FIFO Mode							
Step	Description	R/W	Address	Bit	MACRO	Value	Note
1	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b0	Disable PWM[N]
2	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[15]	OLD_PWM_MODE	1'b0	-
3	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[6]	MODE	1'b0	-
4	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[5]	SRCSEL	1'b0	-
5	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[8]	GUARD_VALUE	USER_DEFINED	-
6	Set IDLE_VALUE	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[7]	IDLE_VALUE	USER_DEFINED	-
7	Set PWM_WAVE_NUM	W	PWM_BASE + 0x038+ PWM_NUM*0x40	[15:0]	PWM_WAVE_NUM	USER_DEFINED	If WAVE_NUM=0, the waveform generation does not stop until PWM is disabled.
8	Set PWM_GDURATION	W	PWM_BASE + 0x01C+ PWM_NUM*0x40	[15:0]	PWM_GDURATION	USER_DEFINED	-
9	Set PWM_HDURATION	W	PWM_BASE + 0x014+ PWM_NUM*0x40	[15:0]	PWM_HDURATION	USER_DEFINED	-
10	Set PWM_LDURATION	W	PWM_BASE + 0x018+ PWM_NUM*0x40	[15:0]	PWM_LDURATION	USER_DEFINED	-
11	Set PWM_SEND_DATA0	W	PWM_BASE + 0x030+ PWM_NUM*0x40	[31:0]	PWM_SEND_DATA0	USER_DEFINED	This value should be written only in the periodical FIFO mode. In other modes, this buffer is for internal memory access.
12	Set PWM_SEND_DATA1	W	PWM_BASE + 0x034+ PWM_NUM*0x40	[31:0]	PWM_SEND_DATA1	USER_DEFINED	This value should be written only in the periodical FIFO mode. In other modes, this buffer is for internal memory access.

PWM Setting Sequence for FIFO Mode							
Step	Description	R/W	Address	Bit	MACRO	Value	Note
13	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[14:9]	STOP_BITPOS	USER_DEFINED	Stop bit position of source data in the periodical mode. In the FIFO mode, it is used to indicate the stop bit position in the total 64 bits. In the memory mode, it is for the stop bit position of the last 32 bits.
14	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b1	Enable PWM[N]

7.6.5.3 Memory Mode

Table 7-17. Memory Mode Setting Procedure

PWM Setting Sequence for Memory Mode							
Step	Description	R/W	Address	Bit	MACRO	Value	Note
1	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b0	Disable PWM[N]
2	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[15]	OLD_PWM_MODE	1'b0	-
3	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[6]	MODE	1'b0	-
4	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[5]	SRCSEL	1'b1	-
5	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[8]	GUARD_VALUE	USER_DEFINED	-
6	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[7]	IDLE_VALUE	USER_DEFINED	-
7	Set PWM_WAVE_NUM	W	PWM_BASE + 0x038+ PWM_NUM*0x40	[15:0]	PWM_WAVE_NUM	USER_DEFINED	If WAVE_NUM=0, the waveform generation does not stop until PWM is disabled.
8	Set PWM_GDURATION	W	PWM_BASE + 0x01C+ PWM_NUM*0x40	[15:0]	PWM_GDURATION	USER_DEFINED	-
9	Set PWM_HDURATION	W	PWM_BASE + 0x014+ PWM_NUM*0x40	[15:0]	PWM_HDURATION	USER_DEFINED	-
10	Set PWM_LDURATION	W	PWM_BASE + 0x018+ PWM_NUM*0x40	[15:0]	PWM_LDURATION	USER_DEFINED	-
11	Set PWM_BUF0_BASE_ ADDR	W	PWM_BASE + 0x020+ PWM_NUM*0x40	[31:0]	PWM_BUF0_BASE_ ADDR	USER_DEFINED	Base address of memory buffer0 for PWM's waveform data
12	Set PWM_BUF0_BASE_ ADDR2	W	PWM_BASE + 0x04C+ PWM_NUM*0x40	[31:0]	PWM_BUF0_BASE_ ADDR2	USER_DEFINED	Extend base address of memory buffer0 for PWM's waveform data
13	Set PWM_BUF0_SIZE	W	PWM_BASE + 0x024+ PWM_NUM*0x40	[2:0]	PWM_BUF0_BASE_ ADDR_EXTEND	USER_DEFINED	Length of waveform data in memory buffer0 that PWM should generate
14	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[14:9]	STOP_BITPOS	USER_DEFINED	Stop bit position of source data in the periodical mode. In the FIFO mode, STOP_BITPOS is used to indicate the stop bit position in the total 64 bits. In the memory mode, it is for the stop bit position of the last 32 bits.
15	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b1	Enable PWM[N]

7.6.5.4 Random Mode

Table 7-18. Random Mode Setting Procedure

PWM Setting Sequence for Random Mode							
Step	Description	R/W	Address	Bit	MACRO	Value	Note
1	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b0	Disable PWM[N]
2	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[15]	OLD_PWM_MODE	1'b0	-
3	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[6]	MODE	1'b0	-
4	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[5]	SRCSEL	1'b1	-
5	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[8]	GUARD_VALUE	USER_DEFINED	-
6	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[7]	IDLE_VALUE	USER_DEFINED	-
7	Set PWM_GDURATION	W	PWM_BASE + 0x01C+ PWM_NUM*0x40	[15:0]	PWM_GDURATION	USER_DEFINED	-
8	Set PWM_HDURATION	W	PWM_BASE + 0x014+ PWM_NUM*0x40	[15:0]	PWM_HDURATION	USER_DEFINED	-
9	Set PWM_LDURATION	W	PWM_BASE + 0x018+ PWM_NUM*0x40	[15:0]	PWM_LDURATION	USER_DEFINED	-
10	Set PWM_BUF0_BASE_ ADDR	W	PWM_BASE + 0x020+ PWM_NUM*0x40	[31:0]	PWM_BUF0_BASE _ADDR	USER_DEFINED	Base address of memory buffer0 for PWM's waveform data
11	Set PWM_BUF0_SIZE	W	PWM_BASE + 0x024+ PWM_NUM*0x40	[2:0]	PWM_BUF0_BASE _ADDR_EXTEND	USER_DEFINED	Length of waveform data in memory buffer0 that PWM should generate
12	Set PWM_BUF1_BASE_ ADDR	W	PWM_BASE + 0x028+ PWM_NUM*0x40	[31:0]	PWM_BUF1_BASE _ADDR	USER_DEFINED	Base address of memory buffer1 for PWM's waveform data
13	Set PWM_BUF1_SIZE	W	PWM_BASE + 0x02C+ PWM_NUM*0x40	[2:0]	PWM_BUF1_BASE _ADDR_EXTEND	USER_DEFINED	Length of waveform data in memory buffer1 that PWM should generate
14	Set PWM_VALID	W	PWM_BASE + 0x048+ PWM_NUM*0x40	[1:0]	BUF1_VALID/ BUF0_VALID	2’b11	1: Memory1/0 is not empty. When writing data to memory1 is finished, write 1 to inform PWM that the data in memory1 are ready.
15	Set PWM_CON	W	PWM_BASE + 0x010+ PWM_NUM*0x40	[14:9]	STOP_BITPOS	USER_DEFINED	Stop bit position of source data in the periodical mode. In the FIFO mode, STOP_BITPOS is used to indicate the stop bit position in the total 64 bits. In the memory mode, it is for the stop bit position of the last 32 bits.
16	Set PWM_ENABLE	W	PWM_BASE + 0x0000	[N]	PWM_ENABLE	1'b1	Enable PWM[N]

7.6.6 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.7 General-Purpose Timer

7.7.1 Introduction

The Application Processor X General-Purpose Timer (APXGPT) includes five 32-bit GPTs and one 64-bit GPT. Each GPT supports four operation modes and can operate on one of the two clock sources, RTC (32.768 kHz) or system clock (13 MHz).

7.7.2 Features

The four operation modes of GPT are ONE-SHOT, REPEAT, KEEP-GO and FREERUN. For the details, please refer to Table 7-19.

Table 7-19. Operation Mode of GPT

Mode	Auto Stop	Interrupt Supported	Count Behavior	When GPTn_COUNT Equals GPTn_COMPARE	Example: Compare Is Set to 2 (Underlining Means Interrupt Asserted)
ONE-SHOT	Yes	Yes	Count stops when GPTn_COUNT equals GPTn_COMPARE.	EN is reset to 0.	0,1,**2**,2,2,2,2,2,2,2,2,2, …
REPEAT	No	Yes	Count is reset to 0 when GPTn_COUNT equals GPTn_COMPARE.	Count is reset to 0.	0,1,**2**,0,1,**2**,0,1,**2**,0,1,**2**, …
KEEP-GO	No	Yes	Count is reset to 0 when the count is overflowed.		0,1,**2**,3,4,5,6,7,8,9,10, …
FREERUN	No	No	Count is reset to 0 when the count overflowed.		0,1,2,3,4,5,6,7,8,9,10, …

Note:
GPTn_COUNT (APXGPT Base address+ (0x0008+0x20*(n-1))), GPTn_COMPARE (APXGPT Base address + (0x000C+0x20*(n-1))) (n=1, 2, 3, 4, 5, 6).
Each timer's operation is independent and can be programmed to select the clock source of RTC (32.768 kHz) or system clock (13 MHz). After the clock source is determined, the division ratio of the selected clock can be programmed. The division ratio can be fine-granulated as 1 to 13 and coarse-granulated as 16, 32 and 64.

7.7.3 Block Diagram

APXGPT consists of five sets of 32-bit GPTs and one set of 64-bit GPT.

When the GPT triggers IRQ, it also issues a wakeup signal to "Sleep Control", and then “Sleep Control” can wake up MCU if MCU is in sleep mode.

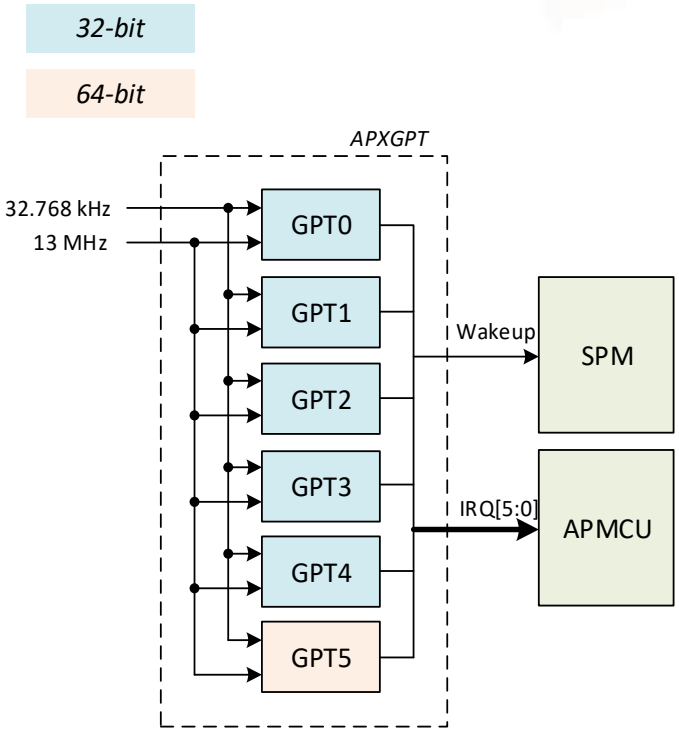


Figure 7-38. Block Diagram of APXGPT

7.7.4 Theory of Operations

For the GPT5 64-bit timer, the read operation of the 64-bit timer value is separated into two APB reads since an APB read is of 32-bit width. To perform the read operation of 64-bit timer value, the lower word should be read first and then the higher word. The read operation of lower word freezes the “read value” of the higher word but does not freeze the timer counting. This ensures that the separated read operation acquires the correct timer value.

To program and use GPT, please note:

- Counter value can be read at any time when the clock source is system clock.
- Counter value can be read at any time even when the clock source is RTC.
- Comparison value can be programmed at any time. When the comparison value is rewritten during count operation, counter is reset to 0 and restarts counting.

7.8 System Timer (sys_timer)

7.8.1 Introduction

sys_timer is a 64-bit, non-stop, always-on up-counter which is used as a universal timer in system. The counter value of sys_timer is passed to APMCU, SCP, GPU, and other micro-processors to provide uniform system timestamp for OSs (Android, Linux, RTOS etc.).

7.8.2 Features

The sys_timer supports the following features:

- A 64-bit, always-on up-counter enabled by default to tick with 13 MHz clock period
- Clock divider to allow timer to tick with 26/13/6.5 MHz clock period
- Hardware counter incremented compensation when clock source switches to 32 kHz
- 8 x 32-bit counter timeout values (read as 32-bit down counter)
- Security access permission control for each control register (with one-time lock bit)

7.8.3 Block Diagram

The block diagram of sys_timer is shown in Figure 7-39.

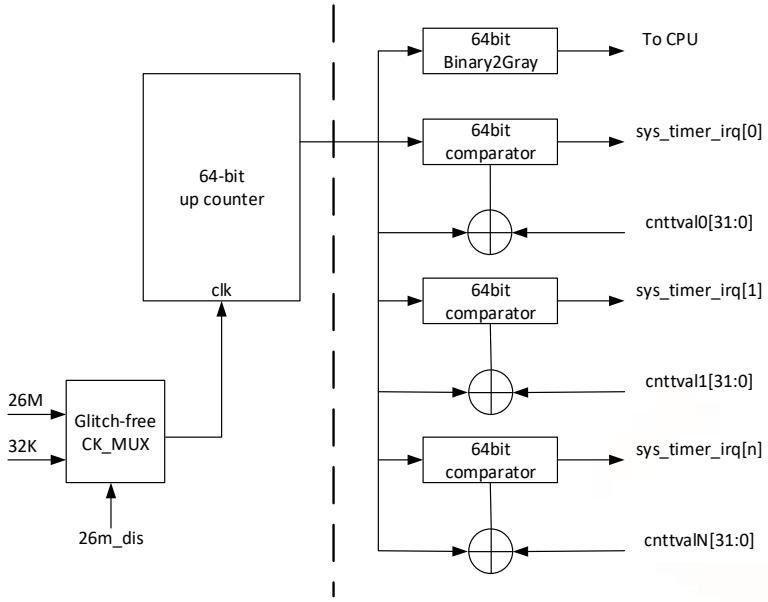


Figure 7-39. sys_timer Block Diagram

7.8.4 Theory of Operations

The sys_timer consists of one 64-bit up-counter, one glitch-free clock mux, one clock divider, and multiple 64-bit comparators. The 64-bit up-counter is enabled by default and starts ticking with 13 MHz clock period after reset is released. The counter can also be programmed to tick with 26 MHz, 13 MHz, or 6.5 MHz clock period and switched to 32 kHz clock period by power manager when 26 MHz clock source is unavailable. In the 32 kHz mode, the counter increment offset values change with the clock divider settings to compensate for the difference in the clock rate. The 64-bit counter value is exported to other sub-systems like CPU, GPU, SCP, and so on.

To avoid the problem of multi-bit clock domain crossing, the counter value is converted into gray-code before output, and a gray-to-binary converter is required to convert the counter value back at the receiving side. Aside from exporting the 64-bit counter value to the different sub-systems, the sys_timer also provides multiple comparators that allow you to set the 32-bit counter's timeout values, which can trigger the interrupts after timeout. When you write a 32-bit offset value into the CNTTVAL[n] register, the 32-bit offset value is added to the current 64-bit counter as the expected timeout value. The behavior of the timer is described in Figure 7-40.

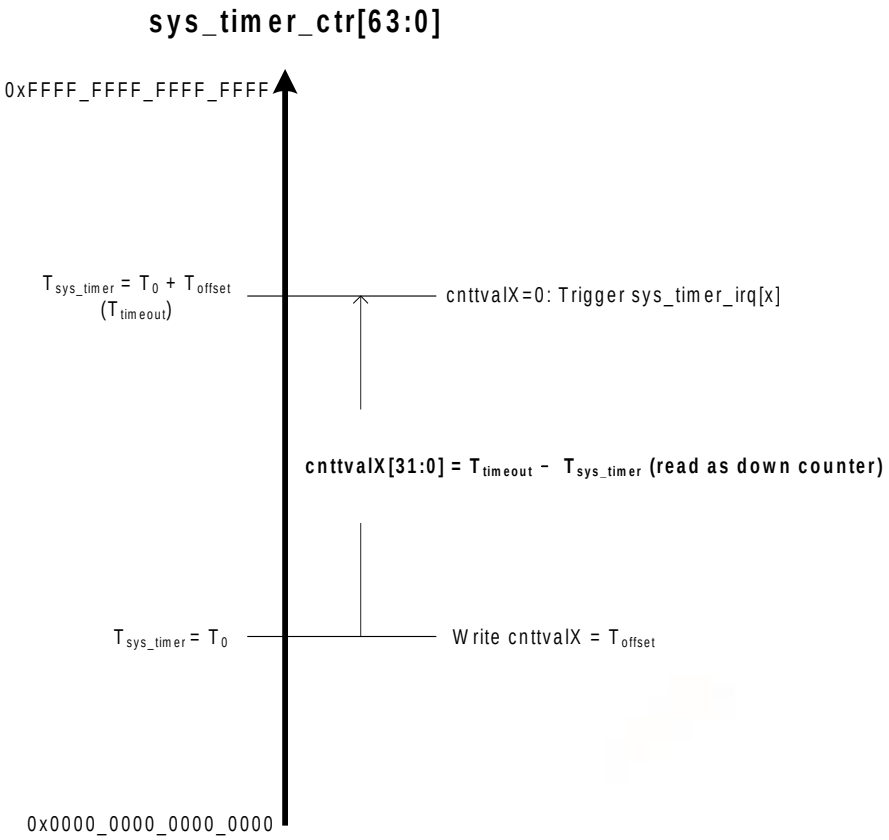


Figure 7-40. Behavior of sys_timer With Timeout Value

Reading CNTTVAL[n] represents the difference between the expected timeout value and the current 64-bit counter value. Therefore, the CNTTVAL[n] can be seen as a 32-bit down-counter (with 64-bit up-counter counting) which triggers sys_timer_irq[n] when CNTTVAL reaches zero.

7.8.5 Programming Guide

This section describes the following operating sequence:

Take sys_timer0 as an example to show the programming sequence.

Table 7-20. sys_timer Setting Flow

Step	Address	Register Name	Local Address	R/W	Value	Description
Set up timer						
1	Sys_timer base address + 0x040	CNTTVAL0_CON	CNTTVAL0_EN	RW	0'h1	Enable timer0
2	Sys_timer base address + 0x044	CNTTVAL0	CNTTVAL0	RW		Set timeout value
3	Sys_timer base address + 0x040	CNTTVAL0_CON		RW	0'h3	Enable interrupt
Wait for sys_timer0 issuing interrupt						
Update timer and clear interrupt						
4	Sys_timer base address + 0x044	CNTTVAL0	CNTTVAL0	RW		Set timeout value
5	Sys_timer base address + 0x040	CNTTVAL0_CON		RW	0'h13	Clear interrupt
Disable timer						
6	Sys_timer base address + 0x040	CNTTVAL0_CON		RW	0'h11	Clear interrupt and disable interrupt
7	Sys_timer base address + 0x044	CNTTVAL0_CON		RW	0'h0	Disable timer

7.8.6 Register Definition

Refer to "MT7988A Wi-Fi 7 Platform Registers" for detailed register descriptions.

7.9 LVTS (Low Voltage Thermal Sensing) Thermal Controller

7.9.1 Introduction

On an SoC platform, thermal management is very fundamental. Through thermal management, an SoC can operate within specific temperature constraints when achieving computing performance requirements.

Operation under over-temperature condition for a long period of time would cause SoC reliability issue. The thermal management system on an SoC platform includes several thermal sensors embedded in possible hotspots on the die and a thermal controller module for periodic measurement for each hotspot. The measurement results are readable by software. However, in order to minimize software efforts for monitoring temperature, the thermal controller generates interrupts to a system handler for abnormal conditions.

7.9.2 Features

The thermal management system includes the following features:

- Support for up to four thermal sensors per bank.
- Programmable periodic temperature measurement.
- Two independent Finite State Machines (FSMs) for temperature monitoring.
- Different types of low pass filters for thermal sensor readings.

7.9.3 Block Diagram

The thermal control subsystem consists of three major building blocks, Sensing device (TSMCU: Thermal Sensing Micro Circuit Unit), Convertor (LVTS convertor) and Controller (LVTS_CTRL) as Figure 7-41 shows.

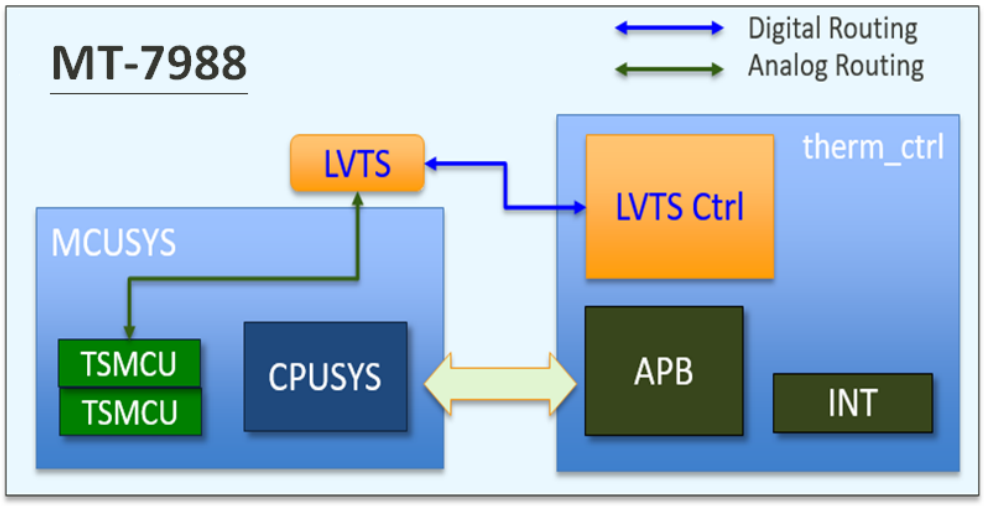


Figure 7-41. Thermal Control Sub-system

7.9.4 Temperature Measurement Scheme

Thermal controller periodically polls all sensors to make sure the SoC operates within a pre-defined temperature range to avoid function failure and reliability issues. According to the temperature measurement, the system performance can be adjusted for a system design with power dissipation being monitored. Figure 7-42 depicts the temperature measurement scheme.

The hottest location on a SoC may vary in different applications. When the thermal controller informs the software of an abnormal condition, the consecutive power reduction methodology should be efficient and with low latency.

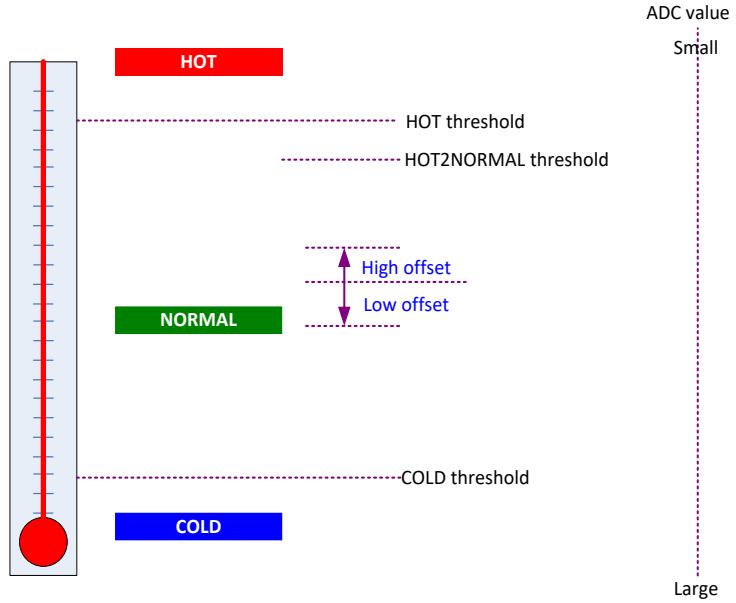


Figure 7-42. Block Diagram of System Temperature Measurement Scheme

7.9.5 Theory of Operations

7.9.5.1 Interrupt Control Flow

Figure 7-42 and Figure 7-43 show the interrupt conditions of high and low temperature monitor. Software can determine which temperature sensors to be monitored. Once any of the following three interrupt conditions occurs in any of the monitored temperature sensors, an interrupt is generated.

- Cold interrupt: Asserted when the temperature crosses and falls below the cold threshold.
- Hot interrupt: Asserted when the temperature crosses and rises above the hot threshold.
- Hot-to-normal interrupt: Asserted when the temperature crosses and falls below the hot-to-normal threshold.

Figure 7-44 shows the Finite State Machine (FSM) diagram. The states are described as below:

- COLD_ST: The temperature is lower than the cold threshold.
- NORMAL_ST: The temperature is within hot-to-normal threshold and cold threshold.
- HOT1_ST: The temperature is within hot-to-normal threshold and hot threshold, and the previous state is NORMAL_ST.

- VERY_HOT_ST: The temperature is higher than hot threshold.
- HOT2_ST: The temperature is within hot-to-normal threshold and hot threshold, and the previous state is VERY_HOT_ST.

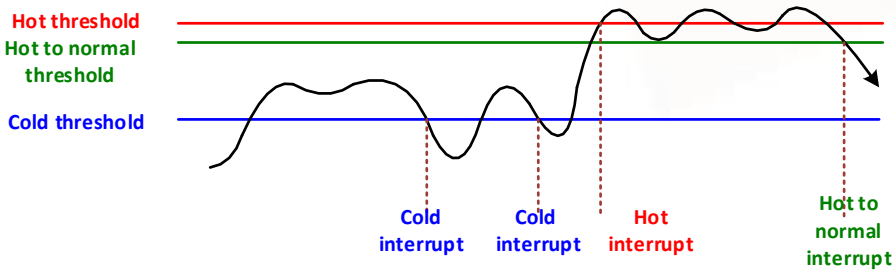


Figure 7-43. Interrupt Conditions for High/Low Temperature Monitoring

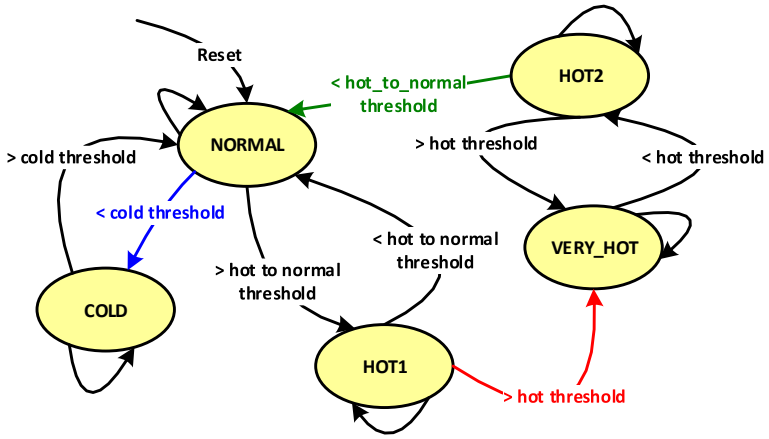


Figure 7-44. Finite State Machine for High/Low Temperature Monitoring

The system also provides a feature to control the junction temperature to be within the pre-defined HIGH-OFFSET and LOW-OFFSET. Figure 7-45 depicts the concept of the feature and Figure 7-46 shows the FSM. There are two interrupts and two alarms:

- LOW-OFFSET alarm to RGU: Assert alarm when the temperature crosses and drops below the low offset.
- HIGH-OFFSET alarm to RGU: Assert alarm when the temperature crosses and rises above the high offset.
- LOW-OFFSET interrupt: Asserted when the temperature crosses and drops below the low offset. The FSM is transferred from the NORMAL state into the LOW OFFSET state.
- HIGH-OFFSET interrupt: Asserted when the temperature crosses and rises above the high offset. The FSM is transferred from the NORMAL state into the HIGH OFFSET state.

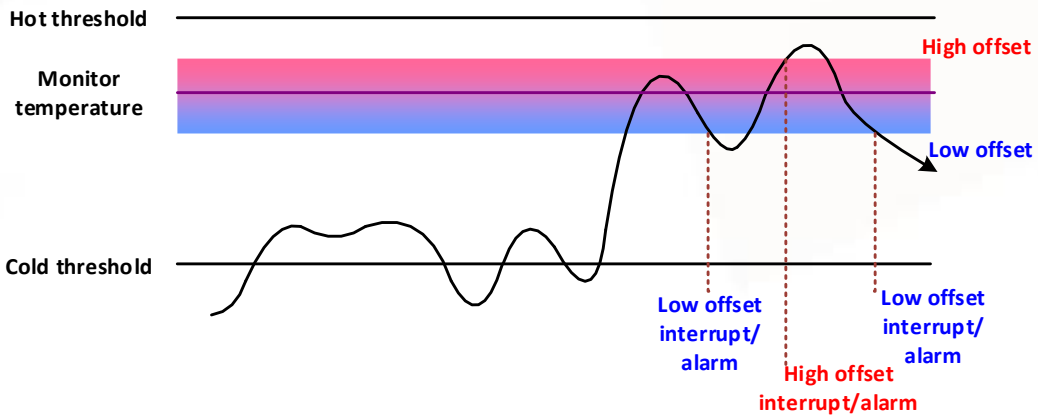


Figure 7-45. Interrupt Conditions of High/Low OFFSET Monitoring

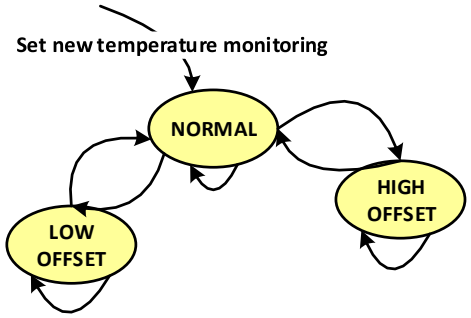


Figure 7-46. Finite State Machine for High/Low OFFSET Monitoring

7.9.6 Thermal Sensor Locations

Thermal sensors are placed at different locations within a die for monitoring junction temperature of different die locations. Table 7-21 lists all thermal sensor locations.

Table 7-21. Locations Monitored by Temperature Sensors

LVTS Bank	Module	Sensor
#1	CPU	0, 1
	ETH2P5G	2, 3
#2	TOPS	0, 1
	ETHWARP	2, 3

7.9.7 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.10 Audio

7.10.1 Introduction

The audio system provides the ability to exchange audio data. In the MT7988A, the audio system includes a Enhanced-Time-Division-Multiplexer (eTDM) interface which provides Pulse-Code Modulation (PCM) and Inter-IC Sound (I2S) interfaces. The eTDM interface transmits DRAM data into PCM or I2S data format.

7.10.2 Features

The audio system supports the following features:

- eTDM interface
  - Master input/output mode
  - Sample rates: 8 kHz/12 kHz/16 kHz/24 kHz/32 kHz/48 kHz/96 kHz/192 kHz
  - Format: I2S or PCM
  - 16-bit/24-bit/32-bit precision
  - Channel width: 16-bit or 32-bit
  - Channel number: 2/4/6/8/12/16/24/32. But only the first 2 channels contain data
  - eTDM interface signal description is shown in Table 7-23.
  - PCM_CLK: Bit clock. BCLK clock rate is listed in Table 7-22.
  - PCM_SYNC: Frame sync or LRCK
  - PCM_DO: eTDM serial output data signal
  - PCM_DI: eTDM serial input data signal
  - PCM_MCK: MCLK
  - Sample rate, bit precision and channel width of eTDM IN should be the same with those of eTDM OUT.

Table 7-22. eTDM BCLK Clock Rate

		eTDM Scenarios BCLK Clock Rate Table									
	Channel Number per FS	**32**	**32**	**16**	**16**	**8**	**8**	**4**	**4**	**2**	**2**
	Channel Width (bits)	**16**	**32**	**16**	**32**	**16**	**32**	**16**	**32**	**16**	**32**
Sample Rate (kHz)	**8.000**	4096	8192	2048	4096	1024	2048	512	1024	256	512
	12.000	6144	12288	3072	6144	1536	3072	768	1536	384	768
	16.000	8192	-	4096	8192	2048	4096	1024	2048	512	1024
	24.000	12288	-	6144	12288	3072	6144	1536	3072	768	1536
	32.000	-	-	8192	-	4096	8192	2048	4096	1024	2048
	48.000	-	-	12288	-	6144	12288	3072	6144	1536	3072
	96.000	-	-	-	-	12288	-	6144	12288	3072	6144
	192.000	-	-	-	-	-	-	12288	-	6144	12288

7.10.3 Block Diagram

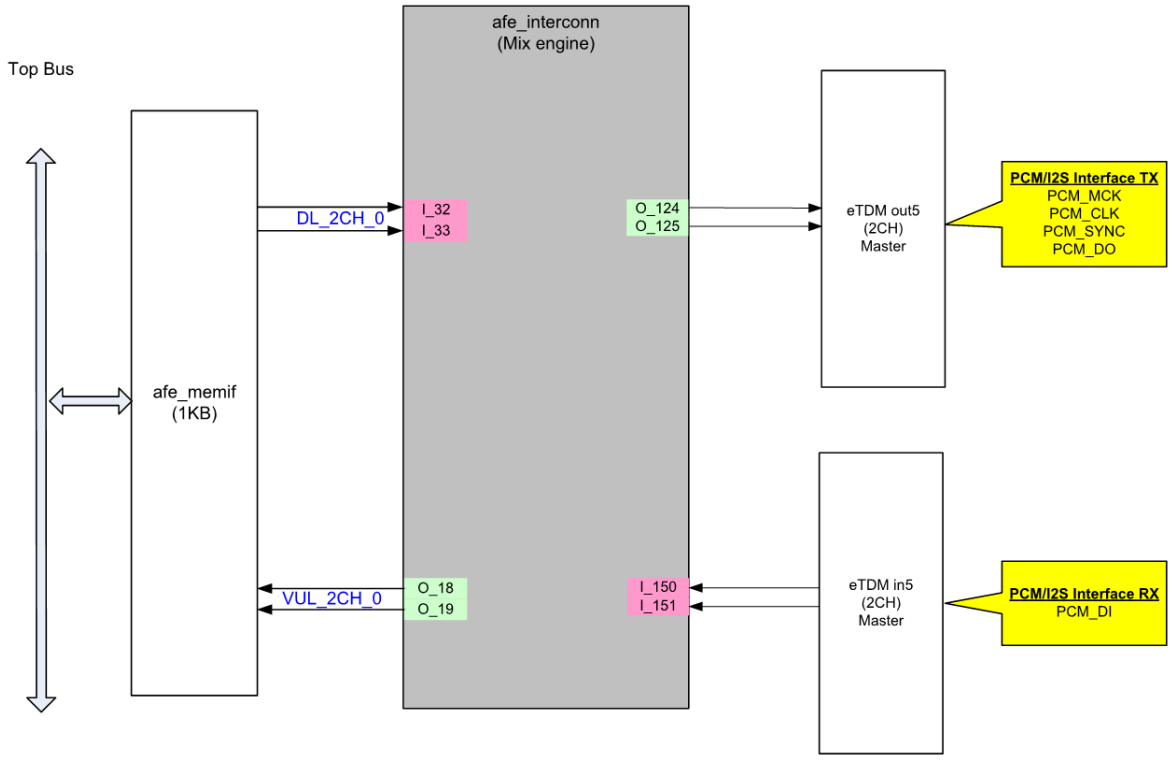


Figure 7-47. Audio Block Diagram

7.10.4 Signal Description

Table 7-23. eTDM Interface Signal Description

Signal Name	Direction	Description
PCM_MCK	Output	Max. frequency <= 49.152 MHz
PCM_SYNC	Output	8 kHz/12 kHz/16 kHz/24 kHz/32 kHz/48 kHz/96 kHz/192 kHz
PCM_CLK	Output	CHNUM*16*FS or CHNUM*32*FS (see Table 7-22). Max. frequency <= 12.288 MHz
PCM_DO	Output	TX data
PCM_DI	Input	RX data

7.10.5 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

7.11 GPIO (General-Purpose Input/Output)

7.11.1 Introduction

The I/O pins can be programmed as multiple-purpose pins such as GPIO, NAND or SPI pins. By setting the GPIO_MODE register, I/O can be selected for specific function.

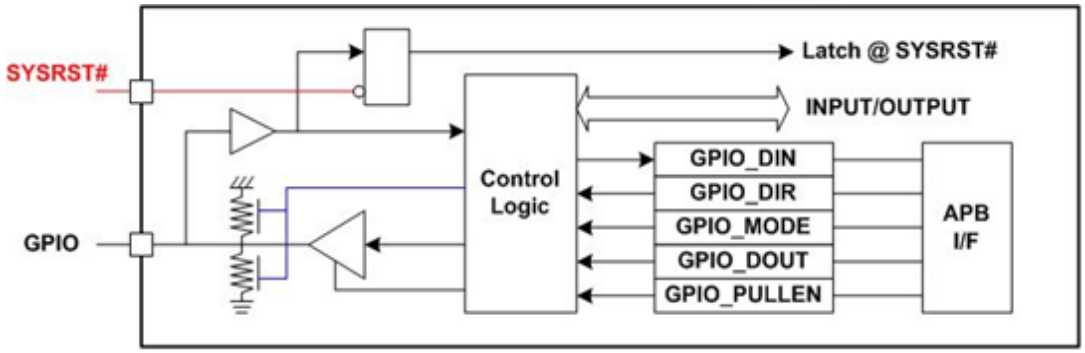


Figure 7-48. Block Diagram of GPIO

All functions should comply with the priority rule. If more than one I/Os are set for the same output function, all of the selected I/Os are able to output specific signals. However, if more than one I/Os are set for the same input (or bi-directional) function, only the I/O with the largest GPIO index works functionally.

Table 7-24. Summary of Configuration Registers of GPIOs

Address Base Name	Address
IOCFG_BR_BASE	0x11D0_0000
IOCFG_LB_BASE	0x11E0_0000
IOCFG_RB_BASE	0x11D2_0000
IOCFG_TL_BASE	0x11F0_0000
IOCFG_TR_BASE	0x11C1_0000
GPIO_BASE	0x1001_F000

7.11.2 GPIO Control Table

Table 7-25. GPIO Control Table

Name	IES	SMT	PU	PD	PUPD
PAD_EMMC_CK	IOCFG_RB_BASE+0x0050[0]	IOCFG_RB_BASE+0x0140[0]	-	-	IOCFG_RB_BASE+0x0070[0]
PAD_EMMC_CMD	IOCFG_RB_BASE+0x0050[1]	IOCFG_RB_BASE+0x0140[1]	-	-	IOCFG_RB_BASE+0x0070[1]
PAD_EMMC_DATA_0	IOCFG_RB_BASE+0x0050[2]	IOCFG_RB_BASE+0x0140[2]	-	-	IOCFG_RB_BASE+0x0070[2]
PAD_EMMC_DATA_1	IOCFG_RB_BASE+0x0050[3]	IOCFG_RB_BASE+0x0140[3]	-	-	IOCFG_RB_BASE+0x0070[3]
PAD_EMMC_DATA_2	IOCFG_RB_BASE+0x0050[4]	IOCFG_RB_BASE+0x0140[4]	-	-	IOCFG_RB_BASE+0x0070[4]
PAD_EMMC_DATA_3	IOCFG_RB_BASE+0x0050[5]	IOCFG_RB_BASE+0x0140[5]	-	-	IOCFG_RB_BASE+0x0070[5]
PAD_EMMC_DATA_4	IOCFG_RB_BASE+0x0050[6]	IOCFG_RB_BASE+0x0140[6]	-	-	IOCFG_RB_BASE+0x0070[6]
PAD_EMMC_DATA_5	IOCFG_RB_BASE+0x0050[7]	IOCFG_RB_BASE+0x0140[7]	-	-	IOCFG_RB_BASE+0x0070[7]
PAD_EMMC_DATA_6	IOCFG_RB_BASE+0x0050[8]	IOCFG_RB_BASE+0x0140[8]	-	-	IOCFG_RB_BASE+0x0070[8]
PAD_EMMC_DATA_7	IOCFG_RB_BASE+0x0050[9]	IOCFG_RB_BASE+0x0140[9]	-	-	IOCFG_RB_BASE+0x0070[9]
PAD_EMMC_DSL	IOCFG_RB_BASE+0x0050[10]	IOCFG_RB_BASE+0x0140[10]	-	-	IOCFG_RB_BASE+0x0070[10]
PAD_EMMC_RSTB	IOCFG_RB_BASE+0x0050[11]	IOCFG_RB_BASE+0x0140[11]	-	-	IOCFG_RB_BASE+0x0070[11]
PAD_GPIO_A	IOCFG_TL_BASE+0x0030[0]	IOCFG_TL_BASE+0x00c0[0]	-	-	IOCFG_TL_BASE+0x0050[0]
PAD_GPIO_B	IOCFG_TL_BASE+0x0030[1]	IOCFG_TL_BASE+0x00c0[1]	-	-	IOCFG_TL_BASE+0x0050[1]
PAD_GPIO_C	IOCFG_TL_BASE+0x0030[2]	IOCFG_TL_BASE+0x00c0[2]	-	-	IOCFG_TL_BASE+0x0050[2]
PAD_GPIO_P	IOCFG_TR_BASE+0x0040[0]	IOCFG_TR_BASE+0x00e0[0]	-	-	IOCFG_TR_BASE+0x0060[0]
PAD_GPIO_RESET	IOCFG_TR_BASE+0x0040[1]	IOCFG_TR_BASE+0x00e0[1]	IOCFG_TR_BASE+0x0070[0]	IOCFG_TR_BASE+0x0050[0]	-
PAD_GPIO_WPS	IOCFG_TR_BASE+0x0040[2]	IOCFG_TR_BASE+0x00e0[2]	IOCFG_TR_BASE+0x0070[1]	IOCFG_TR_BASE+0x0050[1]	-
PAD_I2C_1_SCL	IOCFG_TL_BASE+0x0030[3]	IOCFG_TL_BASE+0x00c0[3]	-	IOCFG_TL_BASE+0x0040[0]	-
PAD_I2C_1_SDA	IOCFG_TL_BASE+0x0030[4]	IOCFG_TL_BASE+0x00c0[4]	-	IOCFG_TL_BASE+0x0040[1]	-
PAD_I2C_2_SCL	IOCFG_TL_BASE+0x0030[5]	IOCFG_TL_BASE+0x00c0[5]	-	IOCFG_TL_BASE+0x0040[2]	-
PAD_I2C_2_SDA	IOCFG_TL_BASE+0x0030[6]	IOCFG_TL_BASE+0x00c0[6]	-	IOCFG_TL_BASE+0x0040[3]	-
PAD_JTAG_JTCLK	IOCFG_TR_BASE+0x0040[3]	IOCFG_TR_BASE+0x00e0[3]	-	-	IOCFG_TR_BASE+0x0060[1]
PAD_JTAG_JTDI	IOCFG_TR_BASE+0x0040[4]	IOCFG_TR_BASE+0x00e0[4]	-	-	IOCFG_TR_BASE+0x0060[2]
PAD_JTAG_JTDO	IOCFG_TR_BASE+0x0040[5]	IOCFG_TR_BASE+0x00e0[5]	-	-	IOCFG_TR_BASE+0x0060[3]
PAD_JTAG_JTMS	IOCFG_TR_BASE+0x0040[6]	IOCFG_TR_BASE+0x00e0[6]	-	-	IOCFG_TR_BASE+0x0060[4]
PAD_JTAG_JTRST_N	IOCFG_TR_BASE+0x0040[7]	IOCFG_TR_BASE+0x00e0[7]	-	-	IOCFG_TR_BASE+0x0060[5]

Name	IES	SMT	PU	PD	PUPD
PAD_LED_A	IOCFG_TR_BASE+0x0040[8]	IOCFG_TR_BASE+0x00e0[8]	-	-	IOCFG_TR_BASE+0x0060[6]
PAD_LED_B	IOCFG_TR_BASE+0x0040[9]	IOCFG_TR_BASE+0x00e0[9]	-	-	IOCFG_TR_BASE+0x0060[7]
PAD_LED_C	IOCFG_TR_BASE+0x0040[10]	IOCFG_TR_BASE+0x00e0[10]	-	-	IOCFG_TR_BASE+0x0060[8]
PAD_LED_D	IOCFG_TR_BASE+0x0040[11]	IOCFG_TR_BASE+0x00e0[11]	-	-	IOCFG_TR_BASE+0x0060[9]
PAD_LED_E	IOCFG_TR_BASE+0x0040[12]	IOCFG_TR_BASE+0x00e0[12]	-	-	IOCFG_TR_BASE+0x0060[10]
PAD_PCIE30_1L_0_CLKREQ_N	IOCFG_LB_BASE+0x0030[0]	IOCFG_LB_BASE+0x00b0[0]	IOCFG_LB_BASE+0x0060[0]	IOCFG_LB_BASE+0x0040[0]	-
PAD_PCIE30_1L_0_PRESET_N	IOCFG_LB_BASE+0x0030[1]	IOCFG_LB_BASE+0x00b0[1]	-	-	IOCFG_LB_BASE+0x0050[0]
PAD_PCIE30_1L_0_WAKE_N	IOCFG_LB_BASE+0x0030[2]	IOCFG_LB_BASE+0x00b0[2]	IOCFG_LB_BASE+0x0060[1]	IOCFG_LB_BASE+0x0040[1]	-
PAD_PCIE30_1L_1_CLKREQ_N	IOCFG_LB_BASE+0x0030[3]	IOCFG_LB_BASE+0x00b0[3]	IOCFG_LB_BASE+0x0060[2]	IOCFG_LB_BASE+0x0040[2]	-
PAD_PCIE30_1L_1_PRESET_N	IOCFG_LB_BASE+0x0030[4]	IOCFG_LB_BASE+0x00b0[4]	-	-	IOCFG_LB_BASE+0x0050[1]
PAD_PCIE30_1L_1_WAKE_N	IOCFG_LB_BASE+0x0030[5]	IOCFG_LB_BASE+0x00b0[5]	IOCFG_LB_BASE+0x0060[3]	IOCFG_LB_BASE+0x0040[3]	-
PAD_PCIE30_2L_0_CLKREQ_N	IOCFG_LB_BASE+0x0030[6]	IOCFG_LB_BASE+0x00b0[6]	IOCFG_LB_BASE+0x0060[4]	IOCFG_LB_BASE+0x0040[4]	-
PAD_PCIE30_2L_0_PRESET_N	IOCFG_LB_BASE+0x0030[7]	IOCFG_LB_BASE+0x00b0[7]	-	-	IOCFG_LB_BASE+0x0050[2]
PAD_PCIE30_2L_0_WAKE_N	IOCFG_LB_BASE+0x0030[8]	IOCFG_LB_BASE+0x00b0[8]	IOCFG_LB_BASE+0x0060[5]	IOCFG_LB_BASE+0x0040[5]	-
PAD_PCIE30_2L_1_CLKREQ_N	IOCFG_LB_BASE+0x0030[9]	IOCFG_LB_BASE+0x00b0[9]	IOCFG_LB_BASE+0x0060[6]	IOCFG_LB_BASE+0x0040[6]	-
PAD_PCIE30_2L_1_PRESET_N	IOCFG_LB_BASE+0x0030[10]	IOCFG_LB_BASE+0x00b0[10]	-	-	IOCFG_LB_BASE+0x0050[3]
PAD_PCIE30_2L_1_WAKE_N	IOCFG_LB_BASE+0x0030[11]	IOCFG_LB_BASE+0x00b0[11]	IOCFG_LB_BASE+0x0060[7]	IOCFG_LB_BASE+0x0040[7]	-
PAD_PCM_CLK_I2S_BCLK	IOCFG_RB_BASE+0x0050[12]	IOCFG_RB_BASE+0x0140[12]	-	-	IOCFG_RB_BASE+0x0070[12]
PAD_PCM_DRX_I2S_DIN	IOCFG_RB_BASE+0x0050[13]	IOCFG_RB_BASE+0x0140[13]	-	-	IOCFG_RB_BASE+0x0070[13]
PAD_PCM_DTX_I2S_DOUT	IOCFG_RB_BASE+0x0050[14]	IOCFG_RB_BASE+0x0140[14]	-	-	IOCFG_RB_BASE+0x0070[14]
PAD_PCM_FS_I2S_LRCK	IOCFG_RB_BASE+0x0050[15]	IOCFG_RB_BASE+0x0140[15]	-	-	IOCFG_RB_BASE+0x0070[15]
PAD_PCM_MCK_I2S_MCLK	IOCFG_RB_BASE+0x0050[16]	IOCFG_RB_BASE+0x0140[16]	-	-	IOCFG_RB_BASE+0x0070[16]
PAD_PMIC_I2C_SCL	IOCFG_TL_BASE+0x0030[7]	IOCFG_TL_BASE+0x00c0[7]	-	IOCFG_TL_BASE+0x0040[4]	-
PAD_PMIC_I2C_SDA	IOCFG_TL_BASE+0x0030[8]	IOCFG_TL_BASE+0x00c0[8]	-	IOCFG_TL_BASE+0x0040[5]	-
PAD_PWMD0	IOCFG_TR_BASE+0x0040[13]	IOCFG_TR_BASE+0x00e0[13]	-	-	IOCFG_TR_BASE+0x0060[11]
PAD_PWMD1	IOCFG_RB_BASE+0x0050[17]	IOCFG_RB_BASE+0x0140[17]	-	-	IOCFG_RB_BASE+0x0070[17]
PAD_SMI_0_MDC	IOCFG_TL_BASE+0x0030[9]	IOCFG_TL_BASE+0x00c0[9]	-	-	IOCFG_TL_BASE+0x0050[3]
PAD_SMI_0_MDIO	IOCFG_TL_BASE+0x0030[10]	IOCFG_TL_BASE+0x00c0[10]	-	-	IOCFG_TL_BASE+0x0050[4]

Name	IES	SMT	PU	PD	PUPD
PAD_SPI0_CLK	IOCFG_RB_BASE+0x0050[18]	IOCFG_RB_BASE+0x0140[18]	-	-	IOCFG_RB_BASE+0x0070[18]
PAD_SPI0_CSB	IOCFG_RB_BASE+0x0050[19]	IOCFG_RB_BASE+0x0140[19]	-	-	IOCFG_RB_BASE+0x0070[19]
PAD_SPI0_HOLD	IOCFG_RB_BASE+0x0050[20]	IOCFG_RB_BASE+0x0140[20]	-	-	IOCFG_RB_BASE+0x0070[20]
PAD_SPI0_MISO	IOCFG_RB_BASE+0x0050[21]	IOCFG_RB_BASE+0x0140[21]	-	-	IOCFG_RB_BASE+0x0070[21]
PAD_SPI0_MOSI	IOCFG_RB_BASE+0x0050[22]	IOCFG_RB_BASE+0x0140[22]	-	-	IOCFG_RB_BASE+0x0070[22]
PAD_SPI0_WP	IOCFG_RB_BASE+0x0050[23]	IOCFG_RB_BASE+0x0140[23]	-	-	IOCFG_RB_BASE+0x0070[23]
PAD_SPI1_CLK	IOCFG_RB_BASE+0x0050[24]	IOCFG_RB_BASE+0x0140[24]	-	-	IOCFG_RB_BASE+0x0070[24]
PAD_SPI1_CSB	IOCFG_RB_BASE+0x0050[25]	IOCFG_RB_BASE+0x0140[25]	-	-	IOCFG_RB_BASE+0x0070[25]
PAD_SPI1_MISO	IOCFG_RB_BASE+0x0050[26]	IOCFG_RB_BASE+0x0140[26]	-	-	IOCFG_RB_BASE+0x0070[26]
PAD_SPI1_MOSI	IOCFG_RB_BASE+0x0050[27]	IOCFG_RB_BASE+0x0140[27]	-	-	IOCFG_RB_BASE+0x0070[27]
PAD_SPI2_CLK	IOCFG_RB_BASE+0x0050[28]	IOCFG_RB_BASE+0x0140[28]	-	-	IOCFG_RB_BASE+0x0070[28]
PAD_SPI2_CSB	IOCFG_RB_BASE+0x0050[29]	IOCFG_RB_BASE+0x0140[29]	-	-	IOCFG_RB_BASE+0x0070[29]
PAD_SPI2_HOLD	IOCFG_RB_BASE+0x0050[30]	IOCFG_RB_BASE+0x0140[30]	-	-	IOCFG_RB_BASE+0x0070[30]
PAD_SPI2_MISO	IOCFG_RB_BASE+0x0050[31]	IOCFG_RB_BASE+0x0140[31]	-	-	IOCFG_RB_BASE+0x0070[31]
PAD_SPI2_MOSI	IOCFG_RB_BASE+0x0060[0]	IOCFG_RB_BASE+0x0150[0]	-	-	IOCFG_RB_BASE+0x0080[0]
PAD_SPI2_WP	IOCFG_RB_BASE+0x0060[1]	IOCFG_RB_BASE+0x0150[1]	-	-	IOCFG_RB_BASE+0x0080[1]
PAD_UART0_RXD	IOCFG_TR_BASE+0x0040[14]	IOCFG_TR_BASE+0x00e0[14]	-	-	IOCFG_TR_BASE+0x0060[12]
PAD_UART0_TXD	IOCFG_TR_BASE+0x0040[15]	IOCFG_TR_BASE+0x00e0[15]	-	-	IOCFG_TR_BASE+0x0060[13]
PAD_UART1_CTS	IOCFG_TR_BASE+0x0040[16]	IOCFG_TR_BASE+0x00e0[16]	-	-	IOCFG_TR_BASE+0x0060[14]
PAD_UART1_RTS	IOCFG_TR_BASE+0x0040[17]	IOCFG_TR_BASE+0x00e0[17]	-	-	IOCFG_TR_BASE+0x0060[15]
PAD_UART1_RXD	IOCFG_TR_BASE+0x0040[18]	IOCFG_TR_BASE+0x00e0[18]	-	-	IOCFG_TR_BASE+0x0060[16]
PAD_UART1_TXD	IOCFG_TR_BASE+0x0040[19]	IOCFG_TR_BASE+0x00e0[19]	-	-	IOCFG_TR_BASE+0x0060[17]
PAD_UART2_CTS	IOCFG_TL_BASE+0x0030[11]	IOCFG_TL_BASE+0x00c0[11]	-	-	IOCFG_TL_BASE+0x0050[5]
PAD_UART2_RTS	IOCFG_TL_BASE+0x0030[12]	IOCFG_TL_BASE+0x00c0[12]	-	-	IOCFG_TL_BASE+0x0050[6]
PAD_UART2_RXD	IOCFG_TL_BASE+0x0030[13]	IOCFG_TL_BASE+0x00c0[13]	-	-	IOCFG_TL_BASE+0x0050[7]
PAD_UART2_TXD	IOCFG_TL_BASE+0x0030[14]	IOCFG_TL_BASE+0x00c0[14]	-	-	IOCFG_TL_BASE+0x0050[8]
PAD_USB_DRV_VBUS_P0	IOCFG_LB_BASE+0x0030[12]	IOCFG_LB_BASE+0x00b0[12]	IOCFG_LB_BASE+0x0060[8]	IOCFG_LB_BASE+0x0040[8]	-
PAD_USB_DRV_VBUS_P1	IOCFG_TR_BASE+0x0040[20]	IOCFG_TR_BASE+0x00e0[20]	IOCFG_TR_BASE+0x0070[2]	IOCFG_TR_BASE+0x0050[2]	-

Name	IES	SMT	PU	PD	PUPD
PAD_WATCHDOG	IOCFG_TR_BASE+0x0040[21]	IOCFG_TR_BASE+0x00e0[21]	-	-	IOCFG_TR_BASE+0x0060[18]

7.11.3 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

8 Connectivity

8.1 NETSYS

8.1.1 Frame Engine (FE)

8.1.1.1 Introduction

Frame Engine is a high-performance network processing engine for network protocol layer 2 to layer 4, providing a DMA interface to transfer Ethernet packets between CPU and GMACs. Frame Engine includes ADMA, QDMA, PSE and PPE. It also supports Wi-Fi to and from Ethernet hardware offload, which is composed of WED and WDMA. To fulfill high-throughput encryption and description requirements, in-line EIP197 is added into Frame Engine dataflow. Besides, there is one individual channel to communicate with TOPS subsys for GRE tunnel process.

8.1.1.2 Features

The main features of these functions include.

- Frame Engine, composed of PPE, ADMA and QDMA, connects to DRAM via AXI bus (64-bit)
- IPv4 routing, NAT, NAPT
- IPv6 routing, NAT, NAPT, DS-lite, 6RD, MAPE/T, 6to4, 464
- QoS
- IP/TCP/UDP checksum offload
- TSO (TCP Segmentation Offload)
- LRO (Larger Receive Offload)

8.1.1.2.1 PSE Features

- Egress rate limiting and shaping
- Flow control for no-packet-loss guarantee
- Emulated multicast support for keep-alive (can mirror a TX packet to CPU)
- IP/TCP/UDP checksum offload
- IP/TCP/UDP checksum Generation
- VLAN and PPPoE header insertion
- TCP Segmentation Offload
- Auto-Padding for sub-64 B packets

8.1.1.2.2 ADMA Features

- Supports 4 TX descriptor rings and 4 RX descriptor rings
- Scatter/Gather DMA
- Support Delayed interrupt
- Configurable 8/16/32 32-bit word burst length

- TSO (TCP Segmentation Offload)
- LRO (Large Receive Offload) with 4 normal RX ring and 4 LRO RX ring
- RSS (Receive Side Scaling)

8.1.1.2.3 QDMA Features

- 1 TX link-list descriptor and 1 RX ring descriptor on AP side
- Supports Direct TX packet hardware forwarding from PPE
- Configurable 16/32/64/128 64-bit burst length
- Configurable up to 15 read outstanding transactions to improve bandwidth
- Delayed interrupt
- Supports 128 TX queues
- Per TX queue forward/drop packet accounting
- Per TX queue forward byte accounting
- Per TX queue minimum and maximum rate control
- Strict priority arbitration between 128 TX queues, for under MIN rate traffic
- Either Strict priority or Weighted Fair Queuing arbitration between 128 TX queues, for over MIN rate traffic
- 128 TX queues can be separated into 4 scheduling groups
- Random Early Drop with configurable dropping probability
- 8 TX queues support SFQ and DRR scheduling, with virtual queue capability
- Up to 1024 virtual queues shared by 8 TX queues
- SFQ/DRR hash perturb
- Per virtual queue forward packet/byte and drop packet accounting
- TSO (TCP Segmentation Offload)

8.1.1.2.4 EIP197 Features

- Look-aside and In-Line encryption and decryption Engine
- Optional MAC header parsing (Ethernet II and IEEE 802.2 LLC/SNAP), including VLAN and PBB
- IPv4/IPv6 and IPsec-ESP/DTLS header parsing to look up a flow or transform

8.1.1.3 Block Diagram

Frame Engine comprises DMA masters, configuration register bus slave, packet processor and switch element.

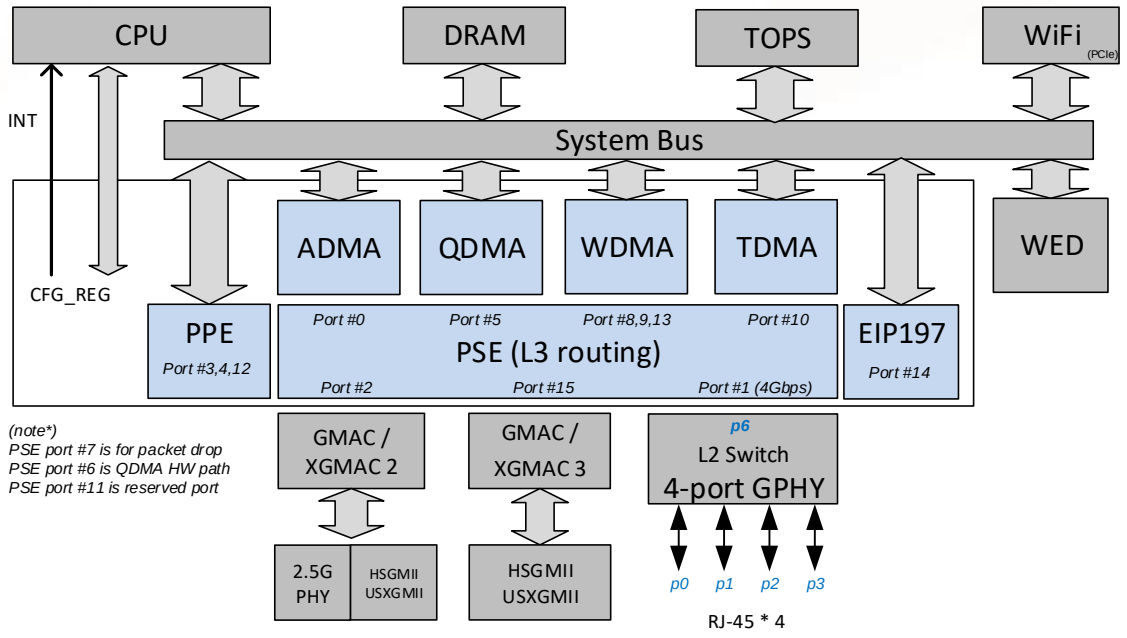


Figure 8-1. Block Diagram of Frame Engine in NETSYS

8.1.2 Gigabit-Media Access Controller (GMAC)

8.1.2.1 Introduction

The MAC sub-layer defines a medium-independent facility, built on the medium-dependent physical facility provided by the physical layer, and under the access-layer-independent LAN LLC sub-layer (or another MAC client).

8.1.2.2 Features

- MAC layer functions of IEEE 802.3 and Ethernet
- 10/100/1000/2500 Mbps bit rates
- Support for per port SGMII
- Automatic 32-bit CRC generation and checking
- Inter-Frame Gap Shrink (96 bits-->64 bits)
- Report packet status (good, CRC error, alignment, oversize, undersize, and other MIBs information)
- Flow control and automatic generation of control frames in full-duplex mode (IEEE 802.3x)
- EEE (Energy Efficient Ethernet) capability for full-duplex mode (IEEE 802.3az)
- Packet length: Up to 15K for jumbo frames application
- 2-port GMAC (port 1 for LAN, port 2 for WAN)

- PHY indirect-access by MDIO

8.1.2.3 Block Diagram

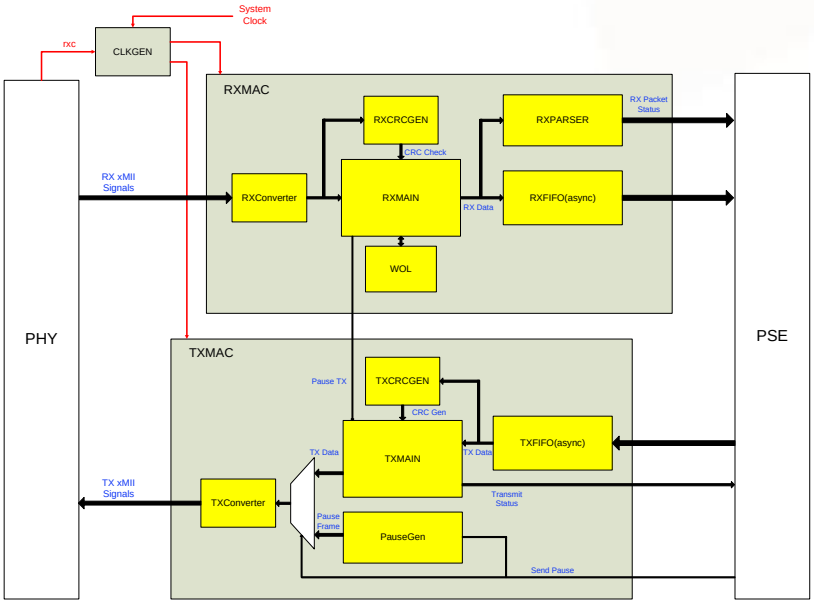


Figure 8-2. Block Diagram of GMAC (Per Port)

8.1.3 10 Gigabit-Media Access Controller (XGMAC)

8.1.3.1 Introduction

XGMAC is responsible for transferring data between FE (Frame Engine) and PCS (Physical Coding Sublayer). Ethernet TX packet from FE is sent to XGMAC through MBI and is delivered to PCS through XGMII. Similarly, Ethernet RX packet from PCS is sent to XGMAC through XGMII and is delivered to FE through MBI.

8.1.3.2 Features

- MAC layer functions of IEEE 802.3 and Ethernet
- 10G/5G/2.5G/1G/100 Mbps data rate with 64-bit XGMII
- 802.3x flow control and automatic generation of control frames in full-duplex mode
- Fault process: local fault and remote fault
- Configurable minimum TX Inter Packet Gap (IPG): 8 to 64 bytes
- Minimum RX Inter Packet Gap (IPG): 5 bytes
- Configurable maximum RX packet length: up to 16383 bytes
- Automatic 32-bit CRC generation and checking
- Statistics counters: packet_num, byte_num, error_packet_num etc.
- XGMII TX to RX loopback

8.1.3.3 Block Diagram

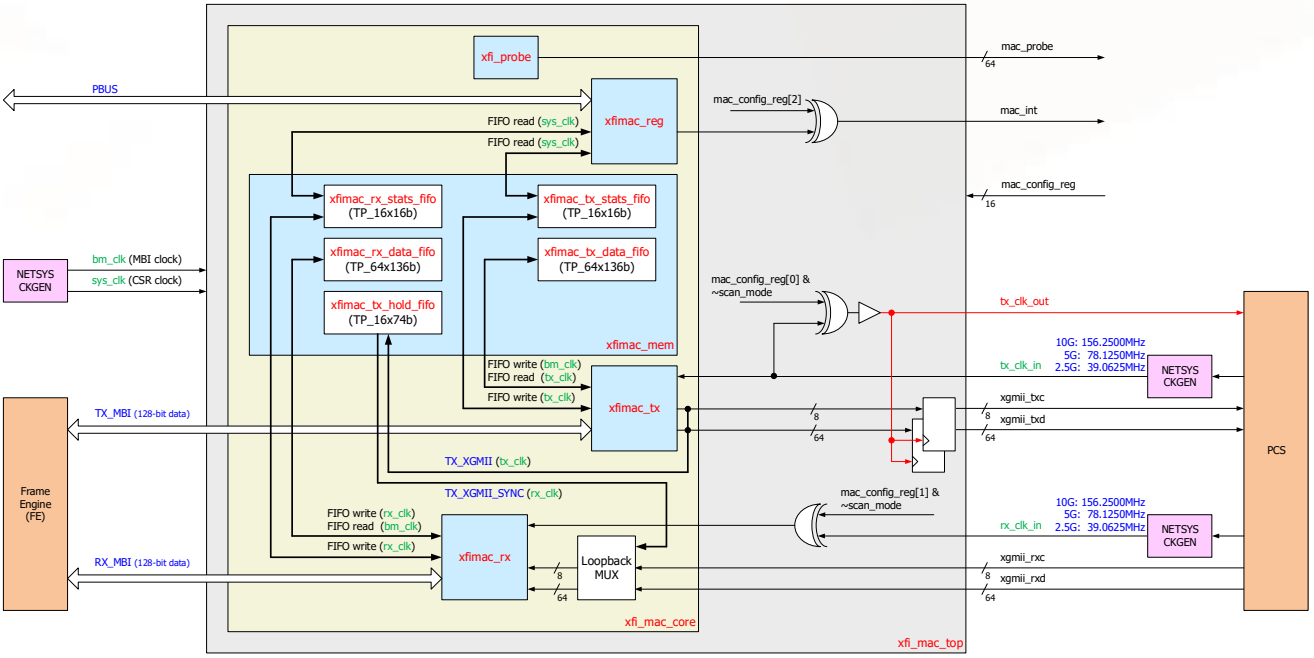


Figure 8-3. Block Diagram of XGMAC (Per Port)

8.1.4 4-port Gigabit L2 Switch (GSW)

8.1.4.1 Introduction

GSW is a highly integrated embedded wired Ethernet switch with high-performance and non-blocking forwarding. It includes a 4-port Gigabit Ethernet MAC and a 4-port Gigabit Ethernet PHY for home entertainment, home automation and so on. GSW provides 4 Gbps upstream to and downstream from Frame Engine.

8.1.4.2 Features

- 10Base-T, 10Base-Te, 100Base-TX, and 1000Base-T compliant transceivers
- Compliant with IEEE 802.3 Auto-Negotiation
- 1 LED per GEPHY port and integrated MDI resistors
- IEEE 802.3az Energy Efficient Ethernet
- Embedded 4-port 10/100/1000 Mbps MDI transceivers
- Built-in address table with 2K MAC addresses which can support up to 8 filtering databases
  - Accessible by managing interface to keep static addresses
  - IVL/SVL based on FID from VLAN table
  - Programmable aging timer: no aging out, 10 to 1,000,000 seconds; default is 300 seconds
  - Configurable address look-up algorithm. Address look-up based on the proprietary hashing algorithm, CRC16 or CRC32.
  - Collision Pool with 64 entries
- Per-port MAC address learning control to protect the system from being attacked by hackers

  - Disable learning or aging for Per-port
  - Limit SA learning number for Per-port
- IEEE 802.1x access control protocol and advanced security features
  - Access policy based on port-based, MAC-based and guest VLAN
  - Access control based on ACL rules
  - Drop unknown Source MAC or Destination MAC address for Per-port
- Up to 4K full VLAN entries with flexible support for IEEE 802.1Q and port-based VLAN
  - Port-based, tag-based, and up to 4 port-and-protocol based VLAN
  - Per-port VLAN tag addition, removal, or leave unchanged
  - Special tag for CPU port
  - Per Egress port stack VLAN (Q in Q)
  - Per Egress port 1:1 and N:1 VLAN Translation
- Embedded 384-KB packet buffer
  - 256 bytes per page to store packet data
  - In total 1,536 pages to provide effective flow control mechanism
- 8 priority queues per egress port and the advanced Priority Flow Control (PFC) mechanism
- Per queue MAX-MIN shapers with different schedulers – Strict Priority (SP), Weight Fair Queue (WFQ), Round-Robin (RR) and mixed ones
  - The minimum shapers can use either SP or RR
  - The maximum shapers can use either SP or WRQ
  - Per shaper is enabled/disabled
  - Per shaper threshold setting
  - Per shaper with both the leaky bucket and token bucket algorithms
- Ingress and egress rate control
  - Per port egress rate control with both the leaky bucket and token bucket approaches
- ACL Table includes 256 entries rule table and 128 entries rule control table
  - ACL Rules support layer 1 to layer 4. Rules include Port No., DA/SA, Ethernet Type, VLAN ID, IP Protocol, SIP/DIP, TCP/UDP, SP/DP and user-defined content
  - Actions: mirror, redirect, dropping, priority adjustment, and traffic rate policing
  - Optional per-port Enable/Disable of ACL function
  - Optional setting of per-port action when ACL is mismatched
- IGMP/MLD snooping
  - IPv4 IGMP v1/v2/v3 snooping and IPv6 MLD v1/v2 snooping
  - Trap all IGMP and MLD packets to the CPU port. CPU writes the correct multicast entry to the lookup table via management interface
  - Hardware IGMP(v1/v2) join and fast leave
  - Partial hardware IGMPv3 and MLDv2 - IS_EX(), TO_EX(), TO_IN(). User-defined SIP Table for IGMPv3/MLDv2. SIP Table hardware auto learning is not supported
- Broadcast, multicast and unknown DA storm control and alert depending on the number of frames received during a period of time to protect the system from being attacked by hackers
- Support for 40 MIB counters per port

8.1.5 Register Definition

Refer to “MT7988A Wi-Fi 7 Platform Registers” for detailed register descriptions.

8.2 GPHY (Gigabit Ethernet PHY)

8.2.1 Introduction

Gigabit Ethernet PHY supports the following interfaces:

- 10/100 Mbps and full-/half-duplex
- 1000 Mbps full-duplex

The GPHY is a fully featured physical layer transceiver with integrated PMD sublayers to support 10BASE-T, 100BASE-TX and 1000BASE-T Ethernet protocols. The GPHY is designed for easy implementation of 10/100/1000 Mbps Ethernet LANs. It interfaces directly with Twisted Pair media via an external transformer. This device interfaces directly with the MAC layer through the IEEE 802.3u Standard Media Independent Interface (MII), and the IEEE 802.3z Gigabit Media Independent Interface (GMII).

8.2.2 Features

The main features of these functions are listed below.

- 10/100/1000 Mbps in full-duplex mode and 10/100 Mbps in half-duplex mode
- One 10/100/1000 MAC port exposed with GMII and MII
- Compliant with IEEE 802.3 Specifications (10BASE-T/100BASE-TX/1000BASE-T)
- Compliant with IEEE802.3az (100BASE-TX EEE/1000BASE-T EEE)
- Auto-negotiation
- Auto-crossover
- Multi-loopback test modes
  - MII loopback
  - Digital loopback
  - Remote loopback
  - External loopback with external Jig

8.3 2.5GPHY (2.5 Gigabit Ethernet PHY)

8.3.1 Introduction

The 2.5 Gigabit Ethernet PHY (2.5GPHY) is a PHY transceiver that integrates 2.5GBASE-T, 1000BASE-T, 100BASE-T, and 10BASE-Te. The EEE (Energy-Efficient Ethernet) feature is also supported for these modes. Fast retrain is a new feature defined in 2.5GBASE-T and is also supported in this PHY. It provides all the necessary PHY functions to transmit and receive Ethernet packets over the CAT5e UTP (Unshielded Twisted Pair) cable.

8.3.2 Features

The key features of these functions are listed below.

- 10/100/1000/2500 Mbps in full-duplex mode. 10/100/1000 mode is the same as GHPY in 8.2
- Compliant with IEEE 802.3 Specifications (10BASE-Te/100BASE-TX/1000BASE-T/2500BASE-T)
- Complies with IEEE 802.3az and supports 100BASE-TX EEE, 1000BASE-T EEE, and 2500BASE-T EEE.
- Compliant with the fast retrain mechanism in 2500BASE-T
- The auto-negotiation feature is extended to the 2.5GPHY.
- The auto-crossover feature is extended to the 2.5GPHY.
- Multi-loopback test modes, which can be found in Section8.2, are also available for the 2.5GPHY.
- Advanced cable diagnostic tool

8.4 SGMII (Serial Gigabit Media Independent Interface)

8.4.1 Introduction

The SGMII is the interface between 10/100/1000/2500 Mbps PHY and Ethernet MAC. The specification was raised by Cisco in 1999, with the aim of reducing the number of pins required compared to the GMII. It uses 2 differential data pairs for TX and RX with clock embedded bit stream to convey frame data and port ability information. The core leverages the 1000Base-X PCS (Physical Coding Sublayer) and Auto-Negotiation from IEEE 802.3 specification (clause 36/37). This IP can support up to 3.125G baud for 2.5 Gbps (proprietary 2500Base-X) data rate of MAC by overclocking.

8.4.2 Features

The main features of these functions are listed below.

- Support 10/100/1000/2500 Mbps in full-duplex mode
- Programmable Link timer
- I2C interface for accessing
- Internal pattern generator with PRBS-7/clock/user-defined patterns for testing
- PCS/SerDes level loopback path in transmit/receive direction for system debugging
- Auto initialization for dumb-switch (auto force to 2500 Mbps mode)

8.5 USXGMII (Universal Serial 10 Gigabit Media Independent Interface)

8.5.1 Introduction

The USXGMII is the interface between 100M/1G/2.5G/5G/10G bps PHY and Ethernet MAC; the specification was raised by Cisco in 2015 for pin reduction compared with the XGMII. It uses 2 differential data pairs for TX and RX with clock embedded bit stream to convey frame data and port ability information. The core leverages the 10GBase-R 64B/66B PCS and Auto-Negotiation in IEEE 802.3 specification (clause 49/37).

The PCS is unchanged with additional functionality being added via the “ordered set” mechanism defined by IEEE. This IP can support up to 10.3125G/5.15625G baud for 10G/5G bps data rate of MAC by overclocking.

MAC-PHY IF	Network Port Types	Replications	PCS/AN	SerDes Speed (Gbps)
10G-USXGMII	100M/1G/2.5G/5G/10G	100/10/4/2/1	Clause 49/37	10.3125
5G-USXGMII	100M/1G/2.5G/5G	50/5/2/1	Clause 49/37	5.15625

8.5.2 Features

- System interface operates in full-duplex mode only
- Hardware assisted auto-negotiation for all supported speeds
- I2C interface for accessing
- Support PRBS-7/PRBS-9/PRBS-31 pattern for testing
- Support RATE-ADAPT/PCS level loopback path in transmit/receive direction for system debugging

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.