MT7981B
Wi-Fi 6 Generation Router Platform: Datasheet

Open Version

Version: 1.0
Release Date: 2024-03-25

Version History

Version	Date	Description
1.0	2024-03-25	Initial release

Summary of Key Features

Feature	Description
Application Processor	
CPU	Arm Cortex-A53 (1.3 GHz, dual-core)
I-Cache, D-Cache	32KB, 32KB per core
L2 Cache	256KB
Security	Support 2 * 256-bit multi-key on OTP eFuse Support 64 version OTP eFuse for anti-rollback
Memory	
DRAM data	16-bit (external memory interface)
External DDR3/DDR4	2133 Mbps (8Gb support)
Connectivity	
Wi-Fi	2x2 (2ss) 11ax 2.4 GHz + 3x3 (2ss) 11ax 5 GHz Integrated PA, LNA and TR-SW 20/40/80/160 MHz bandwidth Support up to 1024QAM External LNA and PA support (optional)
Ethernet	HSGMII x 2; integrate 1G PHY for extender Only support SGMII-1 or internal-GBE due to MAC sharing
HNAT/HQoS	HQoS 64 queues, SFQ 1K queues HNAT (IPv4, IPv6 routing, DS-Lite, 6RD)
Peripheral	
USB	USB3.0 x 1
SPIM NAND Flash	Use on-die ECC
SPI Flash (NOR)	Max 50 MHz Data bit width x1/x2/x4 Support 4-byte address mode compatible with 3-byte address mode
eMMC/SD	eMMC v4.5 @52 MHz 3.3V
I2S	Support two-channel I2S x 1 Sample rate: 8 kHz to 192 kHz
I2C	I2C x 1 100 kHz; support 7/10-bit addressing
SPI	SPI x 1 Support DMA and FIFO mode
UART	UART-Lite (2-pin) x 1 UART (4-pin) x 2
Package	
Package	13.0 mm x 11.7 mm, TFBGA

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 General Description

MT7981B is a highly integrated wireless network router SoC (System-on-Chip) used for high wireless performance, home entertainment, and home automation and so on.

MT7981B is fabricated with advanced silicon process and integrates a dual-core Arm® Cortex-A53 MPCore™ operating up to 1.3 GHz and more DRAM bandwidth. This SoC also includes a variety of peripherals, including SGMII, and USB3.0 (host) ports. To support popular network applications, MT7981B also implements two 2.5Gbps HSGMII Ethernet interface. MT7981B combines with a RF chip, they can provide dual-band concurrent chipset solution for Wi-Fi 6 AX3000 wireless router platforms.

Besides the connectivity features, the hardware-based NAT engine with QoS embedded in MT7981B transporting the audio/video streams in higher priority than other non-timely services also enriches the home entertainment application. The SFQ separating P2P sessions from audio/video ones so that MT7981B guarantees the streaming service.

With the advanced technology and abundant features, MT7981B is well positioned to be the core of next-generation smart Wi-Fi AP router and home gateway systems.

1.1.1 Functional Block Diagram

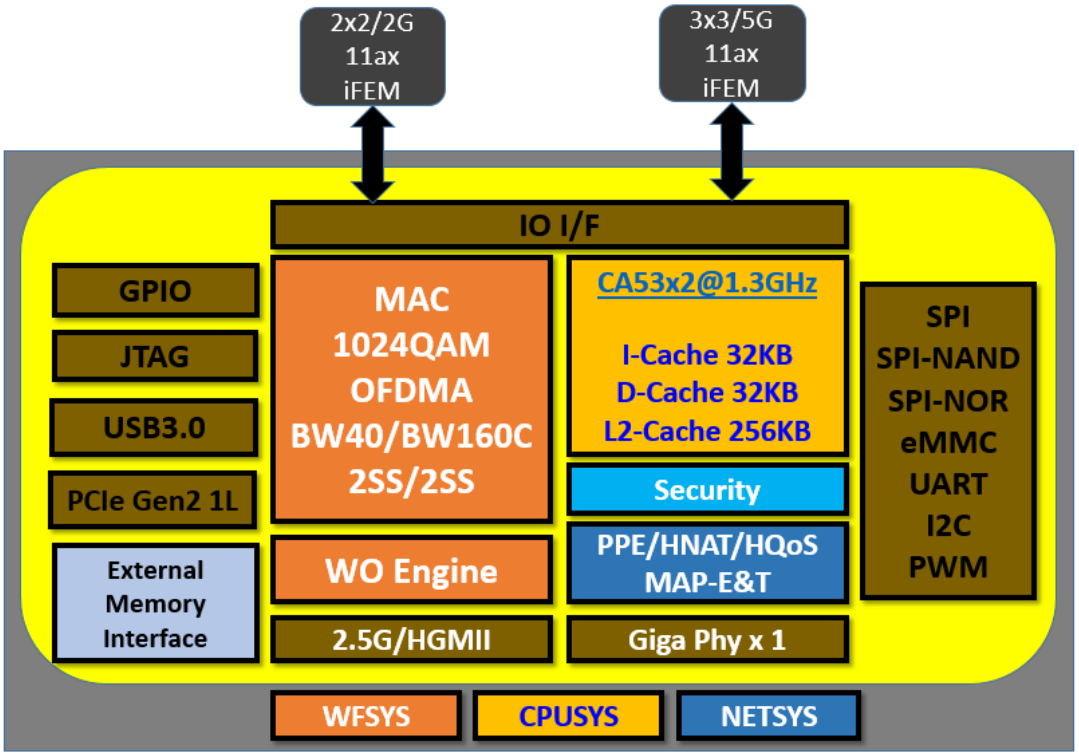


Figure 1-1. MT7981B functional block diagram

1.2 Features

1.2.1 Platform Features

- AP MCU Subsystem
  - Dual-core Arm® Cortex-A53 MPCore™ operating at 1.3 GHz
  - NEON processing engine with advanced SIMD and floating-point extension
  - 32KB L1 I-cache and 32KB L1 D-cache
  - 256KB unified L2 cache
  - Cryptography extension

- Memory Interface
  - 16-bit data bus width
  - Memory clock up to DDR3/DDR4-2133
  - Supports self-refresh/partial self-refresh mode
  - Programmable slew rate for memory controller's IO pads
  - Advanced bandwidth arbitration control

- Peripherals
  - 1 USB3.0/USB2.0 (host)
  - 1 PCIe Gen2 1-Lane interface
  - UART for external devices and debugging interfaces
  - SPI master for external devices
  - SPI NOR flash interface
  - SPI NAND flash interface
  - eMMC4.5/SD interface
  - I2C to control peripheral devices
  - General-Purpose Input/Output
  - PWM (Pulse Width Modulation)

- Operating conditions
  - Core voltage: 0.87V
  - I/O voltage: 1.8V/3.3V

- Package
  - TFBGA 13.0 mm x 11.7 mm
  - Ball pitch: 0.65 mm

1.2.2 Wireless Connectivity Features

1.2.2.1 Wi-Fi MAC Features

Wi-Fi MAC supports the following features:

- Support Dual-band Dual Concurrent
- Support all data rates of 802.11a/b/g/n/ac/ax
- Support short GI and all data rates of 802.11n including MCS0 to MCS7
- Support 802.11ac MCS0 to MCS11
- Support 802.11ax MCS0 to MCS11
- A-MPDU/A-MSDU RX (de-aggregation) and TX (aggregation) support
- TX beamformer and RX beamformee
- TX rate adaptation
- TX power control
- Security
  - 64-bit WEP (WEP-40) and 128-bit WEP (WEP-104) encryption with hardware TKIP processing
  - AES-CCMP hardware processing
  - GCMP hardware processing
- Management/control frame filtering

1.2.2.2 WLAN Baseband Features

Wi-Fi baseband supports the following features:

- Support Dual band Dual Concurrent
- 20/40/80/160 MHz channels
- HE MCS0-11 BW20/40/80/160 MHz with Nss = 1~2
- Short Guard Interval
- Space-Time Block Code (STBC)
- Low Density Parity Check (LDPC)
- Support digital pre-distortion to enhance PA performance
- Smoothing (channel estimation) extension to MIMO case
- Support radar detection
- Beamformer (explicit/implicit)
  - Encoded BW20/40/80/160 up to 2x2 BF matrix apply
- Beamformee
  - Decoded BW20/40/80/160 up to 4x2 MU matrix feedback
- UL OFDMA/MU-MIMO
- DL OFDMA/MU-MIMO
- Max RU number in 2G band is 8
- Max RU number in 5G band is 16
- Max user number is 512 (use DDR 512MB)

1.2.3 Wired Ethernet Features

- Frame Engine
  - Packet DMA (PDMA)
    - 4 Tx descriptor and 4 Rx descriptor rings
    - Scatter/Gather DMA
    - Configurable 4/8/16/32 32-bit burst length and delayed interrupt
    - Support TSO
  - QoS DMA (QDMA)
    - Supports 64 Tx physical queues and 4 sets of scheduler
    - 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 1024 virtual queues for 8 sets of SFQ
  - Packet Switch Engine (PSE)
    - Wire-speed NAT/NAPT routing
    - Egress rate limiting/shaping
    - IP/TCP/UDP checksum offload
    - IP/TCP/UDP checksum generation
    - VLAN & PPPoE header insertion
    - TCP segmentation offload
  - Packet Process Engine (PPE)
    - IPv4 NAP/NAPT, IPv6 Routing and Tunnel IP (DS-Lite, 6RD, 464XLAT, MAP-E/T)
    - 1/2/4/8/16/32K session/flow
    - Flow offloading technology for flexible/high performance packet L3/L4 packet processing
    - Support NAT/NAPT wire-speed within 128 flows for any packet size

- Wi-Fi WARP
  - Ethernet/Wi-Fi offlaod, forwarding packets directly
  - Dynamic buffer allocate and release

- Giga MAC (GMAC)
  - Support IEEE 802.3x full duplex flow control
  - Integrate 1G PHY for extender
  - Only support SGMII-1 or internal-GBE due to their MAC sharing
  - Support HSGMII interface
    - HSGMII supports 10/100/1000Mbps speed change through auto negotiation and configurable 2.5Gbps SerDes links

2 Terms and Abbreviations

2.1 Terms

Table 2-1 describes the terms in Chapter 3 Pin Information.

Table 2-1. Terms

Abbreviation	Description
I	Input
OH	Output high
OL	Output low
A	Analog
P	Power
G	Ground
NC	No connection
PD	Internal pull-down
PU	Internal pull-up
NP	No pull-down or pull-up

2.2 Abbreviations

Table 2-2. Abbreviations

Abbreviation	Description
6RD	IPv6 Rapid Deployment
AHB	Advanced Microcontroller Bus
AXI	Advanced eXtensible Interface
DDR	Double Data Rate
DMA	Direct Memory Access
ECC	Error Correction Code
EINTC	External Interrupt Controller
eTDM	Enhanced-Time-Division-Multiplexer
FE	Frame Engine
FSM	Finite State Machine
GIC	Generic Interrupt Controller
GMII	Gigabit Media-Independent Interface
GPHY	Gigabit Ethernet PHY
GPIO	General-Purpose Input/Output
GPT	General-Purpose Timer
HIF	Host Interface
HW	Hardware
I2C	Inter-Integrated Circuit
I2S	Inter-IC Sound
IRQ	Interrupt Request
LDPC	Low Density Parity Check
MII	Media-Independent Interface
NAPT	Network Address and Port Translation
NAT	Network Address Translation
NFI	NAND Flash Interface
OCC	On-Chip Clock
PCIe	Peripheral Component Interconnect Express
PCM	Pulse Code Modulation
PCS	Physical Coding Sublayer
PLL	Phase-Locked Loop
PWM	Pulse Width Modulator; Pulse Width Modulation
RR	Round-Robin
SerDes	Serializer/Deserializer
SGMII	Serial Gigabit Media-Independent Interface
SoC	System-on-Chip
SPI	Serial Peripheral Interface
STBC	Space-Time Block Code
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
WDT	Watchdog Timer

3 Pin Information

3.1 Pin Description

Table 3-1. Pin description

Pin	Name	Reset [1]		After Reset [1]				Pull [1] [3]	Voltage (V)	Driving (mA)	Description
		State [2]	Pull [3]	State [2]	Aux	Pull [3]	Driving				
GPIO											
F17	GPIO_WPS	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/GPIO_WPS
E18	GPIO_RESET	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/GPIO_RESET
L3	USB_VBUS	I	PD	OL	1	NP	4	PU/PD	3.3	2/4/8/12/16	General-purpose IO/USB_DRV_VBUS
N3	PCIE_PERESET_N	O	H-Z	OL	1	NP	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/PCIE_PERESET_N
M4	JTAG_JTRST_N	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/JTAG_JTRST_N
L4	JTAG_JTDI	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/JTAG_JTDI
L5	JTAG_JTMS	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/JTAG_JTMS
N4	JTAG_JTCLK	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/JTAG_JTCLK
M3	JTAG_JTDO	O	H-Z	OL	1	NP	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/JTAG_JTDO
R3	WO_JTAG_JTDO	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/WO_JTAG_JTDO
P3	WO_JTAG_JTCLK	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/WO_JTAG_JTCLK
P2	WO_JTAG_JTMS	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/WO_JTAG_JTMS
R4	WO_JTAG_JTDI	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/WO_JTAG_JTDI
R2	WO_JTAG_ JTRST_N	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	General-purpose IO/WO_JTAG_JTRST_N
UART											
G16	UART0_RXD	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	UART RX data
G17	UART0_TXD	O	H-Z	OH	1	NP	2	PU/PD	3.3	2/4/8/12/16	UART TX data
Pulse-Width Modulation (PWM)											
D20	PWM0	I	PD	OL	0	NP	4	PU/PD	3.3	2/4/8/12/16	PWM0
Serial Flash											
D19	SPI0_CLK	O	H-Z	OL	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash clock
B20	SPI0_CS	O	H-Z	OH	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash chip select
D18	SPI0_MOSI	O	H-Z	OL	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash master output, slave input
C18	SPI0_MISO	I	PD	I	1	PD	4	PU/PD	3.3	2/4/8/12/16	Serial flash master input, slave output
C19	SPI0_WP	I	PU	I	1	PD	4	PU/PD	3.3	2/4/8/12/16	Serial flash write protect
B19	SPI0_HOLD	I	PU	I	1	PU	4	PU/PD	3.3	2/4/8/12/16	Serial flash hold
A19	SPI1_CLK	O	H-Z	OL	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash clock
C17	SPI1_CS	O	H-Z	OH	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash chip select
B18	SPI1_MOSI	O	H-Z	OL	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash master output, slave input
A18	SPI1_MISO	I	PD	I	1	PD	4	PU/PD	3.3	2/4/8/12/16	Serial flash master input, slave output
T2	SPI2_CLK	I	PD	OL	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash clock
V2	SPI2_CS	O	H-Z	OH	1	NP	4	PU/PD	3.3	2/4/8/12/16	Serial flash chip select
T3	SPI2_MOSI	O	H-Z	OL	1	NP	2	PU/PD	3.3	2/4/8/12/16	Serial flash master output, slave input
T1	SPI2_MISO	I	PD	I	1	PD	2	PU/PD	3.3	2/4/8/12/16	Serial flash master input, slave output
U1	SPI2_WP	I	PU	I	1	PD	2	PU/PD	3.3	2/4/8/12/16	Serial flash write protect
U2	SPI2_HOLD	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	Serial flash hold
DRAM											
U19	EMI_EXTR	A	-	A	-	-	-	-	1.5	-	DRAM calibration resistor
R15	EMI_RESET_N	A	-	A	-	-	-	-	1.5	-	DRAM signal
T14	EMI0_A0	A	-	A	-	-	-	-	1.5	-	DRAM signal
U12	EMI0_A1	A	-	A	-	-	-	-	1.5	-	DRAM signal
R13	EMI0_A10	A	-	A	-	-	-	-	1.5	-	DRAM signal
T13	EMI0_A11	A	-	A	-	-	-	-	1.5	-	DRAM signal
V14	EMI0_A12	A	-	A	-	-	-	-	1.5	-	DRAM signal
U16	EMI0_A13	A	-	A	-	-	-	-	1.5	-	DRAM signal
U13	EMI0_A14	A	-	A	-	-	-	-	1.5	-	DRAM signal
V16	EMI0_A2	A	-	A	-	-	-	-	1.5	-	DRAM signal
T16	EMI0_A3	A	-	A	-	-	-	-	1.5	-	DRAM signal
T11	EMI0_A4	A	-	A	-	-	-	-	1.5	-	DRAM signal
U17	EMI0_A5	A	-	A	-	-	-	-	1.5	-	DRAM signal

Table 3-1. Pin description

Pin	Name	Reset [1]		After Reset [1]				Pull [1] [3]	Voltage (V)	Driving (mA)	Description
		State [2]	Pull [3]	State [2]	Aux	Pull [3]	Driving				
V13	EMI0_A6	A	-	A	-	-	-	-	1.5	-	DRAM signal
V17	EMI0_A7	A	-	A	-	-	-	-	1.5	-	DRAM signal
T12	EMI0_A8	A	-	A	-	-	-	-	1.5	-	DRAM signal
T15	EMI0_A9	A	-	A	-	-	-	-	1.5	-	DRAM signal
V18	EMI0_BA0	A	-	A	-	-	-	-	1.5	-	DRAM signal
R11	EMI0_BA1	A	-	A	-	-	-	-	1.5	-	DRAM signal
R16	EMI0_BA2	A	-	A	-	-	-	-	1.5	-	DRAM signal
R14	EMI0_CAS_N	A	-	A	-	-	-	-	1.5	-	DRAM signal
V11	EMI0_CK_C	A	-	A	-	-	-	-	1.5	-	DRAM signal
U10	EMI0_CK_T	A	-	A	-	-	-	-	1.5	-	DRAM signal
U11	EMI0_CKE0	A	-	A	-	-	-	-	1.5	-	DRAM signal
V19	EMI0_CS0_N	A	-	A	-	-	-	-	1.5	-	DRAM signal
U7	EMI0_DM0	A	-	A	-	-	-	-	1.5	-	DRAM signal
T5	EMI0_DM1	A	-	A	-	-	-	-	1.5	-	DRAM signal
U5	EMI0_DQ0	A	-	A	-	-	-	-	1.5	-	DRAM signal
V8	EMI0_DQ1	A	-	A	-	-	-	-	1.5	-	DRAM signal
R10	EMI0_DQ10	A	-	A	-	-	-	-	1.5	-	DRAM signal
R6	EMI0_DQ11	A	-	A	-	-	-	-	1.5	-	DRAM signal
T10	EMI0_DQ12	A	-	A	-	-	-	-	1.5	-	DRAM signal
T6	EMI0_DQ13	A	-	A	-	-	-	-	1.5	-	DRAM signal
R9	EMI0_DQ14	A	-	A	-	-	-	-	1.5	-	DRAM signal
R7	EMI0_DQ15	A	-	A	-	-	-	-	1.5	-	DRAM signal
V5	EMI0_DQ2	A	-	A	-	-	-	-	1.5	-	DRAM signal
U8	EMI0_DQ3	A	-	A	-	-	-	-	1.5	-	DRAM signal
V4	EMI0_DQ4	A	-	A	-	-	-	-	1.5	-	DRAM signal
V10	EMI0_DQ5	A	-	A	-	-	-	-	1.5	-	DRAM signal
U4	EMI0_DQ6	A	-	A	-	-	-	-	1.5	-	DRAM signal
U9	EMI0_DQ7	A	-	A	-	-	-	-	1.5	-	DRAM signal
T9	EMI0_DQ8	A	-	A	-	-	-	-	1.5	-	DRAM signal
T7	EMI0_DQ9	A	-	A	-	-	-	-	1.5	-	DRAM signal
V7	EMI0_DQS0_C	A	-	A	-	-	-	-	1.5	-	DRAM signal
U6	EMI0_DQS0_T	A	-	A	-	-	-	-	1.5	-	DRAM signal
T8	EMI0_DQS1_C	A	-	A	-	-	-	-	1.5	-	DRAM signal
R8	EMI0_DQS1_T	A	-	A	-	-	-	-	1.5	-	DRAM signal
U15	EMI0_ODT	A	-	A	-	-	-	-	1.5	-	DRAM signal
U14	EMI0_RAS_N	A	-	A	-	-	-	-	1.5	-	DRAM signal
U18	EMI0_WE_N	A	-	A	-	-	-	-	1.5	-	DRAM signal
(SSUSB, SGMII1 or PCIe)/USB											
H2	PCIE_CKP	A	-	A	-	-	-	-	0.9	-	PCIe CLK pin CK +
H3	PCIE_CKN	A	-	A	-	-	-	-	0.9	-	PCIe CLK pin CK -
G1	PCIE_LN0_RXP	A	-	A	-	-	-	-	0.9	-	SSUSB/SGMII1 data pin RX +
G2	PCIE_LN0_RXN	A	-	A	-	-	-	-	0.9	-	SSUSB/SGMII1 data pin RX -
F2	PCIE_LN0_TXP	A	-	A	-	-	-	-	0.9	-	SSUSB/SGMII1 data pin TX +
F1	PCIE_LN0_TXN	A	-	A	-	-	-	-	0.9	-	SSUSB/SGMII1 data pin TX -
K3	USB_DM	A	-	A	-	-	-	-	3.3	-	USB HS/FS/LS data pin data -
K2	USB_DP	A	-	A	-	-	-	-	3.3	-	USB HS/FS/LS data pin data +
Serial Management Interface (SMI)											
L17	SMI_MDC	O	H-Z	OL	1	PU	2	PU/PD	3.3	2/4/8/12/16	Serial management clock
K17	SMI_MDIO	I	PU	I	1	PU	2	PU/PD	3.3	2/4/8/12/16	Serial management data
GBE Interface											
M18	GBE_RESET	I	PD	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	GBE_RESET
M17	GBE_INT	I	PD	I	1	PD	2	PU/PD	3.3	2/4/8/12/16	GBE_INT
Thermal Sensor Interface (TSAUX)											
F19	AUXIN0	A	-	A	-	-	-	-	1.8	-	Aux ADC input 0
F20	AUXIN1	A	-	A	-	-	-	-	1.8	-	Aux ADC input 1
E20	AUXIN2	A	-	A	-	-	-	-	1.8	-	Aux ADC input 2
E19	TSAUX_MD	A	-	A	-	-	-	-	1.8	-	TSAUX_MD
G19	TSAUX_REFP	A	-	A	-	-	-	-	1.8	-	TSAUX_REFP

Table 3-1. Pin description

Pin	Name	Reset [1]		After Reset [1]				Pull [1] [3]	Voltage (V)	Driving (mA)	Description
		State [2]	Pull [3]	State [2]	Aux	Pull [3]	Driving				
SGMII											
J20	SGMII_LN0_RXN	A	-	A	-	-	-	-	0.9	-	SGMII 0 data pin RX -
J19	SGMII_LN0_RXP	A	-	A	-	-	-	-	0.9	-	SGMII 0 data pin RX +
K20	SGMII_LN0_TXN	A	-	A	-	-	-	-	0.9	-	SGMII 0 data pin TX -
K19	SGMII_LN0_TXP	A	-	A	-	-	-	-	0.9	-	SGMII 0 data pin TX +
GPHY/GBE interface											
T19	REXT	A	-	A	-	-	-	-	1.8	-	REXT
T20	GBE_TXVP_A_P0	A	-	A	-	-	-	-	3.3	-	GPHY A_Channel differential P node
R20	GBE_TXVN_A_P0	A	-	A	-	-	-	-	3.3	-	GPHY A_Channel differential N node
R19	GBE_TXVP_B_P0	A	-	A	-	-	-	-	3.3	-	GPHY B_Channel differential P node
P19	GBE_TXVN_B_P0	A	-	A	-	-	-	-	3.3	-	GPHY B_Channel differential N node
N19	GBE_TXVP_C_P0	A	-	A	-	-	-	-	3.3	-	GPHY C_Channel differential P node
N20	GBE_TXVN_C_P0	A	-	A	-	-	-	-	3.3	-	GPHY C_Channel differential N node
M20	GBE_TXVP_D_P0	A	-	A	-	-	-	-	3.3	-	GPHY D_Channel differential P node
M19	GBE_TXVN_D_P0	A	-	A	-	-	-	-	3.3	-	GPHY D_Channel differential N node
OSC Clock											
D16	MAIN_X40M_XIN	A	-	A	-	-	-	-	1.8	-	OSC clock input
Wi-Fi interface (2.4G/5G)											
E3	WF_TOP_CLK	I	PD	OH	1	NP	4	PU/PD	1.8	2/4/8/12/16	SPI clock
C1	WF_TOP_DATA	I	PD	OH	1	NP	4	PU/PD	1.8	2/4/8/12/16	SPI data
A2	WF_HB0	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI clock
D5	WF_HB0_B	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI clock
C3	WF_HB1	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf0 data[0]
B1	WF_HB2	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf0 data[1]
B2	WF_HB3	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf1 data[0]
B3	WF_HB4	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf1 data[1]
C5	WF_HB5	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf2 data[0]
D6	WF_HB6	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf2 data[1]
E6	WF_HB7	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf3 data[0]
D7	WF_HB8	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf3 data[1]
E7	WF_HB9	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf4 data[0]
E8	WF_HB10	I	PD	OL	1	NP	4	PU/PD	1.8	2/4/8/12/16	WRI wf4 data[1]
C2	WF_XO_REQ	I	PD	OH	1	NP	4	PU/PD	1.8	2/4/8/12/16	OSC clock request to RF chip
D3	WF_DIG_RESETB	I	PD	OH	1	NP	4	PU/PD	1.8	2/4/8/12/16	Reset RF chip digital
D2	WF_CBA_RESETB	I	PD	OH	1	NP	4	PU/PD	1.8	2/4/8/12/16	Reset RF chip analog
M1	WF2G_LED	O	H-Z	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	2G LED
M2	WF5G_LED	O	H-Z	I	0	PD	2	PU/PD	3.3	2/4/8/12/16	5G LED
A17	WF4_IP	A	-	A	-	-	-	-	1.8	-	WF4 I_Channel differential P node
B17	WF4_IN	A	-	A	-	-	-	-	1.8	-	WF4 I_Channel differential N node
B16	WF4_QP	A	-	A	-	-	-	-	1.8	-	WF4 Q_Channel differential P node
C15	WF4_QN	A	-	A	-	-	-	-	1.8	-	WF4 Q_Channel differential N node
C14	WF3_IP	A	-	A	-	-	-	-	1.8	-	WF3 I_Channel differential P node
B14	WF3_IN	A	-	A	-	-	-	-	1.8	-	WF3 I_Channel differential N node
C13	WF3_QP	A	-	A	-	-	-	-	1.8	-	WF3 Q_Channel differential P node
B13	WF3_QN	A	-	A	-	-	-	-	1.8	-	WF3 Q_Channel differential N node
C12	WF2_IP	A	-	A	-	-	-	-	1.8	-	WF2 I_Channel differential P node
C11	WF2_IN	A	-	A	-	-	-	-	1.8	-	WF2 I_Channel differential N node
B11	WF2_QP	A	-	A	-	-	-	-	1.8	-	WF2 Q_Channel differential P node
B10	WF2_QN	A	-	A	-	-	-	-	1.8	-	WF2 Q_Channel differential N node
C10	WF1_IP	A	-	A	-	-	-	-	1.8	-	WF1 I_Channel differential P node
C9	WF1_IN	A	-	A	-	-	-	-	1.8	-	WF1 I_Channel differential N node
B8	WF1_QP	A	-	A	-	-	-	-	1.8	-	WF1 Q_Channel differential P node
C8	WF1_QN	A	-	A	-	-	-	-	1.8	-	WF1 Q_Channel differential N node
B7	WF0_IP	A	-	A	-	-	-	-	1.8	-	WF0 I_Channel differential P node
C7	WF0_IN	A	-	A	-	-	-	-	1.8	-	WF0 I_Channel differential N node
B5	WF0_QP	A	-	A	-	-	-	-	1.8	-	WF0 Q_Channel differential P node
A5	WF0_QN	A	-	A	-	-	-	-	1.8	-	WF0 Q_Channel differential N node

Table 3-1. Pin description

Pin	Name	Reset [1]		After Reset [1]				Pull [1] [3]	Voltage (V)	Driving (mA)	Description
		State [2]	Pull [3]	State [2]	Aux	Pull [3]	Driving				
POR											
B4	POR_RSTB	A	-	A	-	-	-	-	1.8	-	PMU_RSTB
A4	POR_BG_OUT	A	-	A	-	-	-	-	1.8	-	BG_OUT
PLLGP											
L1	PLLGP_TP	A	-	A	-	-	-	-	1.8	-	PLLGP_TP test point
L2	PLLGP_TN	A	-	A	-	-	-	-	1.8	-	PLLGP_TN test point
Misc.											
R1	SYS_WATCHDOG	OL	-	OH	-	-	-	-	3.3	-	Watchdog reset
N2	SYSRSTB	I	PU	I	-	PU	-	PU	3.3	-	Power on reset
F16	TESTMODE	I	PD	I	-	PD	-	PD	3.3	-	Test mode
Power											
F8, F9, F10, F13, F14, G6, G7, G13, H6, H14, J6, J15, K14, L8, L9, L10, L14, M7, M14, M15	VCCK	P	-	P	-	-	-	-	0.85	-	Core power supply (DVDD_CORE)
F7 F6 K6 L6 L16 G15 F15	DVDD18IO_TL DVDD18IO_LT DVDD18IO_LB DVDD18IO_BL DVDD18IO_RB DVDD18IO_RTC0 DVDD18IO_RTC1	P	-	P	-	-	-	-	1.8	-	IO power supply
M6 L7 L15 H15 G14	DVDD33IO_LB DVDD33IO_BL DVDD33IO_RB DVDD33IO_RTC0 DVDD33IO_RTC1	P	-	P	-	-	-	-	3.3	-	IO power supply
G4	AVDD09_PCIE	P	-	P	-	-	-	-	0.9	-	SGMII1/PCIe power supply
J18	AVDD09_SGMII	P	-	P	-	-	-	-	0.9	-	SGMII power supply
F4	AVDD18_PCIE	P	-	P	-	-	-	-	1.8	-	SGMII1/PCIe power supply
H19	AVDD18_SGMII	P	-	P	-	-	-	-	1.8	-	SGMII power supply
A3	AVDD15_POR	P	-	P	-	-	-	-	1.5	-	POR power supply
C4	AVDD18_POR	P	-	P	-	-	-	-	1.8	-	POR power supply
E15	AVDD12_CKSQ	P	-	P	-	-	-	-	1.2	-	CKSQ power supply
E14	AVDD18_CKSQ	P	-	P	-	-	-	-	1.8	-	CKSQ power supply
G20	AVDD18_AUXADC	P	-	P	-	-	-	-	1.8	-	TSAUX power supply
J5	AVDD18_PLLGP	P	-	P	-	-	-	-	1.8	-	PLLGP power supply
R17	AVDD18_RDDR	P	-	P	-	-	-	-	1.8	-	DDR power supply
M11	VDDIO_DDR_R	P	-	P	-	-	-	-	1.5	-	DDR power supply
M12	VDDIO_DDR_CA	P	-	P	-	-	-	-	1.5	-	DDR power supply
N8	VDDIO_DDR_DQ	P	-	P	-	-	-	-	1.5	-	DDR power supply
N9	VDDIO_DDR_DQ	P	-	P	-	-	-	-	1.5	-	DDR power supply
N10	VDDIO_DDR_MCLK	P	-	P	-	-	-	-	1.5	-	DDR power supply
L11	DVDD_DDR_TX	P	-	P	-	-	-	-	0.87		DDR core power supply
L12	DVDD_DDR_RX	P	-	P	-	-	-	-	0.87		DDR core power supply
J4	AVDD18_USB	P	-	P	-	-	-	-	1.8	-	USB power supply
K4	AVDD33_USB	P	-	P	-	-	-	-	3.3	-	USB power supply
E11	AVDD12_WBG	P	-	P	-	-	-	-	1.2	-	AFE power supply
E12	AVDD18_WBG	P	-	P	-	-	-	-	1.8	-	AFE power supply
R18	AVDD18_COM	P	-	P	-	-	-	-	1.8	-	GPHY power supply
P18	AVDD33_LD_P0	P	-	P	-	-	-	-	3.3	-	GPHY power supply
E5	DVDD18_VQPS	P	-	P	-	-	-	-	1.8	-	eFuse blow power supply

Table 3-1. Pin description

Pin	Name	Reset [1]		After Reset [1]				Pull [1] [3]	Voltage (V)	Driving (mA)	Description
		State [2]	Pull [3]	State [2]	Aux	Pull [3]	Driving				
Ground											
D4, D17, E1, E2, E4, E16, E17, F3, F5, F11, F12, F18, G3, G5, G8, G9, G10, G11, G12, G18, H1, H4, H5, H7, H8, H9, H10, H11, H12, H13, H16, H17, H18, J2, J3, J7, J8, J9, J10, J11, J12, J13, J14, J16, J17, K5, K7, K8, K9, K10, K11, K12, K13, K15, K16, K18, L13, L18, L19, L20, M5,M13, M16, N5, N6, N7, N11, N12, N13, N14, N15, N16, N17, N18, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, R5, T4, T17, T18, U3, U20, V3, V6, V9, V12, V15	GNDK	G	-	G	-	-	-	-	-	-	Ground (DVSS)
A7, A11, A14, B6, B15, C6, C16, D8, D9, D10, D11, D12, D13, D14, D15, E9, E10, E13	AVSS18_WBG	G	-	G	-	-	-	-	-	-	Ground
A1 A20 V1 V20	NC_A1 NC_A20 NC_V1 NC_V20	-	-	-	-	-	-	-	-	-	NC ball

(1) See Table 2-1 for definitions of I, OH, OL, A, P, G, NC, PD, PU and NP.

(2) The internal pull resistance range: 10 kΩ to 75kΩ, depending on IO configurations.

(3) While IO is set as GPIO mode, the IO driving strength can be either one of 2/4/6/8/10/12/14/16 mA and the default is [ ] mA.

3.1.1 Constantly Tied Pin

Table 3-2. Constantly tied pin

Pin Name	Description
TESTMODE	Test mode (tie to GND)

3.2 Pin Sharing Schemes

Some pins are shared with GPIO to provide maximum flexibility for system designers. You can configure the register to specify pin function.

Table 3-3. Pin sharing scheme

Pin Name	Aux Func.0	Aux Func.1	Aux Func.2	Aux Func.3	Aux Func.4	Aux Func.5	Aux Func.6
GPIO_WPS	B:GPIO0	-	I0:WA_AICE_TCKC	I0:WM_AICE_TCKC	-	O:WM_UART_TXD	-
GPIO_RESET	B:GPIO1	-	B0:WA_AICE_TMSC	B0:WM_AICE_TMSC	-	O:WA_UART_TXD	-
SYS_WATCHDOG	B:GPIO2	O:SYS_WATCHDOG	-	-	-	-	-
PCIE_PERESET_N	B:GPIO3	O:PCIE_PERESET_N	-	-	-	-	-
JTAG_JTDO	B:GPIO4	O:JTAG_JTDO	O:WM_JTAG_JTDO	I1:UART2_RXD	I0:PTA_EXT_ACT	O:SPI1_CLK	-
JTAG_JTDI	B:GPIO5	I1:JTAG_JTDI	I1:WM_JTAG_JTDI	O:UART2_TXD	I0:PTA_EXT_PRI	O:SPI1_MOSI	-
JTAG_JTMS	B:GPIO6	B1:JTAG_JTMS	I1:WM_JTAG_JTMS	I1:UART2_CTS	O:PTA_EXT_WLAN_ACT	I0:SPI1_MISO	B1:I2C_SCL
JTAG_JTCLK	B:GPIO7	I1:JTAG_JTCLK	I1:WM_JTAG_JTCLK	O:UART2_RTS	O:PWM2	O:SPI1_CS	B1:I2C_SDA
JTAG_JTRST_N	B:GPIO8	I0:JTAG_JTRST_N	I0:WM_JTAG_JTRST_N	O:GBE_LED0	O:NET_WO0_UART_TXD	-	-
WO_JTAG_JTDO	B:GPIO9	O:WO0_JTAG_JTDO	I0:WM_AICE_TCKC	-	O:PCM_DTX (1)	-	-
WO_JTAG_JTDI	B:GPIO10	I1:WO0_JTAG_JTDI	B0:WM_AICE_TMSC	-	I0:PCM_DRX (1)	-	-
WO_JTAG_JTMS	B:GPIO11	B1:WO0_JTAG_JTMS	-	-	O:PCM_CLK (1)	-	-
WO_JTAG_JTCLK	B:GPIO12	I1:WO0_JTAG_JTCLK	-	-	O:PCM_FS (1)	-	-
WO_JTAG_JTRST_N	B:GPIO13	I0:WO0_JTAG_JTRST_N	O:PWM0	O:GBE_LED1	O:PCM_MCK (1)	O:SYS_WATCHDOG	-
USB_VBUS	B:GPIO14	O:DRV_VBUS	O:PWM1	O:NET_WO0_UART_TXD	-	-	-
PWM0	B:GPIO15	O:PWM0	O:EMMC_RSTB	O:PWM1	O:NET_WO0_UART_TXD	-	-
SPI0_CLK	B:GPIO16	O:SPI0_CLK	B1:EMMC_DAT0	O:SNFI_CLK	I1:UART1_RXD	-	-
SPI0_MOSI	B:GPIO17	B0:SPI0_MOSI	B1:EMMC_DAT1	B0:SNFI_MOSI	O:UART1_TXD	-	-
SPI0_MISO	B:GPIO18	B0:SPI0_MISO	B1:EMMC_DAT2	B0:SNFI_MISO	I1:UART1_CTS	-	-
SPI0_CS	B:GPIO19	O:SPI0_CS	B1:EMMC_DAT3	O:SNFI_CS	O:UART1_RTS	-	-
SPI0_HOLD	B:GPIO20	B0:SPI0_HOLD	B1:EMMC_DAT4	B0:SNFI_HOLD	O:WM_UART_TXD	-	-
SPI0_WP	B:GPIO21	B0:SPI0_WP	B1:EMMC_DAT5	B0:SNFI_WP	O:WA_UART_TXD	-	-
SPI1_CLK	B:GPIO22	O:SPI1_CLK	B1:EMMC_DAT6	I1:UART2_RXD	I0:PTA_EXT_ACT	-	-
SPI1_MOSI	B:GPIO23	O:SPI1_MOSI	B1:EMMC_DAT7	O:UART2_TXD	I0:PTA_EXT_PRI	-	-
SPI1_MISO	B:GPIO24	I0:SPI1_MISO	B1:EMMC_CMD	I1:UART2_CTS	O:PTA_EXT_WLAN_ACT	-	-
SPI1_CS	B:GPIO25	O:SPI1_CS	B1:EMMC_CLK	O:UART2_RTS	O:PCM_MCK (1)	-	-
SPI2_CLK	B:GPIO26	O:SPI2_CLK	I1:UART1_RXD	-	-	-	-
SPI2_MOSI	B:GPIO27	B0:SPI2_MOSI	O:UART1_TXD	-	-	-	-
SPI2_MISO	B:GPIO28	B0:SPI2_MISO	I1:UART1_CTS	I0:WA_AICE_TCKC	-	-	-
SPI2_CS	B:GPIO29	O:SPI2_CS	O:UART1_RTS	B0:WA_AICE_TMSC	-	-	-
SPI2_HOLD	B:GPIO30	B0:SPI2_HOLD	O:WF2G_LED	O:WM_UART_TXD	B1:I2C_SCL	-	-
SPI2_WP	B:GPIO31	B0:SPI2_WP	O:WF5G_LED	O:WA_UART_TXD	B1:I2C_SDA	-	-
UART0_RXD	B:GPIO32	I1:UART0_RXD	B1:SGMII1_PHY_I2C_SCL	B1:U3_PHY_I2C_SCL	-	-	-
UART0_TXD	B:GPIO33	O:UART0_TXD	B1:SGMII1_PHY_I2C_SDA	B1:U3_PHY_I2C_SDA	-	-	-
WF2G_LED	B:GPIO34	O:WF2G_LED	B1:PCIE_CLK_REQ	-	-	-	-
WF5G_LED	B:GPIO35	O:WF5G_LED	I1:PCIE_WAKE_N	-	-	-	-
SMI_MDC	B:GPIO36	O:SMI_MDC	B1:I2C_SCL	I1:GBE_EXT_MDC	-	-	-
SMI_MDIO	B:GPIO37	B0:SMI_MDIO	B1:I2C_SDA	B1:GBE_EXT_MDIO	-	-	-
GBE_INT	B:GPIO38	I0:MT7531_INT	-	-	-	-	-
GBE_RESET	B:GPIO39	-	-	-	-	-	-
WF_DIG_RESETB	B:GPIO40	O:WF0_DIG_RESETB	-	-	-	-	-
WF_CBA_RESETB	B:GPIO41	O:WF0_CBA_RESETB	-	-	-	-	-
WF_XO_REQ	B:GPIO42	O:WF0_XO_REQ	-	-	-	-	-
WF_TOP_CLK	B:GPIO43	O:WF0_TOP_CLK	-	-	-	-	-
WF_TOP_DATA	B:GPIO44	B0:WF0_TOP_DATA	-	-	-	-	-
WF_HB1	B:GPIO45	B0:WF_HB1	O:WF0_MODE_SEL_1	-	-	-	-
WF_HB2	B:GPIO46	B0:WF_HB2	O:WF0_MODE_SEL_2	-	-	-	-
WF_HB3	B:GPIO47	B0:WF_HB3	O:WF0_XTAL_SEL_0	-	-	-	-
WF_HB4	B:GPIO48	B0:WF_HB4	O:WF0_XTAL_SEL_1	-	-	-	-
WF_HB0	B:GPIO49	O:WF_O_HB0	O:WF0_MODE_SEL_0	-	-	-	-
WF_HB0_B	B:GPIO50	O:WF_O_HB0_B	-	-	-	-	-
WF_HB5	B:GPIO51	B0:WF_HB5	O:WF0_XTAL_SEL_2	-	-	-	-
WF_HB6	B:GPIO52	B0:WF_HB6	-	-	-	-	-
WF_HB7	B:GPIO53	B0:WF_HB7	-	-	-	-	-
WF_HB8	B:GPIO54	B0:WF_HB8	-	-	-	-	-
WF_HB9	B:GPIO55	B0:WF_HB9	-	-	-	-	-
WF_HB10	B:GPIO56	B0:WF_HB10	-	-	-	-	-

(1) Pin sharing:

PCM_DTX: I2S Data Out

PCM_DRX: I2S Data In

PCM_CLK: I2S BCLK/SCK

PCM_FS: I2S WS/LRCK

PCM_MCK: I2S MCLK

3.3 Strapping Options

Table 3-4. Strapping

Pin Name	Strapping Name	Description
USB_VBUS PWM0	Boot Mode	{PWM0, USB_VBUS} 00: SPI-NOR 01: SPI-NAND → SD 10: eMMC 11: SNAND (SNFI) → SD
SPI2_CLK	A-die Crystal	0: 80 MHz 1: 40 MHz (supported)

4 Electrical Characteristics

4.1 Absolute Maximum Ratings

Table 4-1. Absolute maximum ratings

Symbol or Pin Name	Description	Min	Max	Unit
VCCK (DVDD_CORE) DVDD_DDR_TX DVDD_DDR_RX	0.87V supply voltage	-0.3	0.95	V
DVDD18IO_TL DVDD18IO_LT DVDD18IO_LB DVDD18IO_BL DVDD18IO_RB DVDD18IO_RTC0 DVDD18IO_RTC1	1.8V supply voltage	-0.3	1.98	V
DVDD33IO_LB DVDD33IO_BL DVDD33IO_RB DVDD33IO_RTC0 DVDD33IO_RTC1	3.3V supply voltage	-0.3	3.6	V
AVDD33_USB AVDD33_LD_P0	3.3V supply voltage	-0.3	3.6	V
AVDD12_CKSQ AVDD12_WBG	1.2V supply voltage	-0.3	1.32	V
AVDD09_PCIE AVDD09_SGMII	0.9V supply voltage	-0.3	0.99	V
AVDD18_PCIE AVDD18_SGMII AVDD18_COM AVDD18_POR AVDD18_AUXADC AVDD18_PLLGP AVDD18_CKSQ AVDD18_USB AVDD18_WBG AVDD18_RADDR	1.8V supply voltage	-0.3	1.98	V
VDDIO_DDR_DQ VDDIO_DDR_MCLK VDDIO_DDR_R VDDIO_DDR_CA AVDD15_POR	1.5V supply voltage	-0.3	1.575	V
DVDD18_VQPS	1.8V supply voltage	-0.3	1.98	V

4.2 Recommended Operating Range

Table 4-2. Recommended operating range

Symbol or Pin Name	Description	Min	Typ	Max	Unit
VCCK (DVDD_CORE) DVDD_DDR_TX DVDD_DDR_RX	0.87V supply voltage	0.826	0.87	0.914	V
DVDD18IO_TL DVDD18IO_LT DVDD18IO_LB DVDD18IO_BL DVDD18IO_RB DVDD18IO_RTC0 DVDD18IO_RTC1	1.8V supply voltage	1.71	1.8	1.89	V
DVDD33IO_LB DVDD33IO_BL DVDD33IO_RB DVDD33IO_RTC0 DVDD33IO_RTC1	3.3V supply voltage	3.135	3.3	3.465	V
AVDD33_USB AVDD33_LD_P0	3.3V supply voltage	3.135	3.3	3.465	V
AVDD12_CKSQ AVDD12_WBG	1.2V supply voltage	1.14	1.2	1.26	V
AVDD09_PCIE AVDD09_SGMII	0.9V supply voltage	0.855	0.9	0.945	V
AVDD18_PCIE AVDD18_SGMII AVDD18_COM AVDD18_POR AVDD18_AUXADC AVDD18_PLLGP AVDD18_CKSQ AVDD18_USB AVDD18_WBG AVDD18_RADDR	1.8V supply voltage	1.71	1.8	1.89	V
VDDIO_DDR_DQ VDDIO_DDR_MCLK VDDIO_DDR_R VDDIO_DDR_CA AVDD15_POR	1.5V supply voltage	1.425	1.5	1.575	V
DVDD18_VQPS	1.8V supply voltage	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 4-3 Thermal characteristics

Symbol	Description	Performance	
		Typ	Unit
TJ	Maximum junction temperature (plastic package)	125	°C
θJA	Junction to ambient temperature thermal resistance (1) for JEDEC 2L system PCB	23.65	°C/W
θJA	Junction to ambient temperature thermal resistance (1) for JEDEC 4L system PCB	18.05	°C/W
θJC	Junction to case temperature thermal resistance for JEDEC system PCB	8.05	°C/W
ψJt	Junction to the package thermal resistance for JEDEC 2L PCB	1.75	°C/W
ψJt	Junction to the package thermal resistance for JEDEC 4L PCB	1.15	°C/W

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

4.4 AC Electrical Specifications

4.4.1 UART Interface

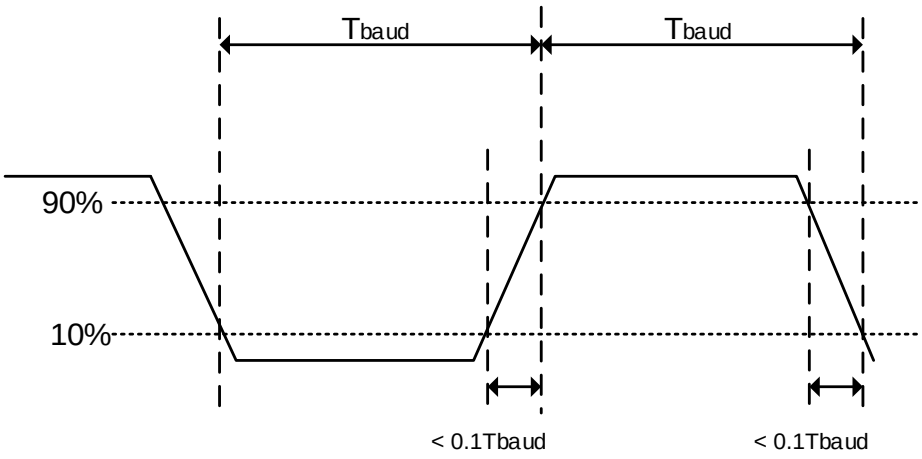


Figure 4-1. UART timing

4.4.2 SPI Interface

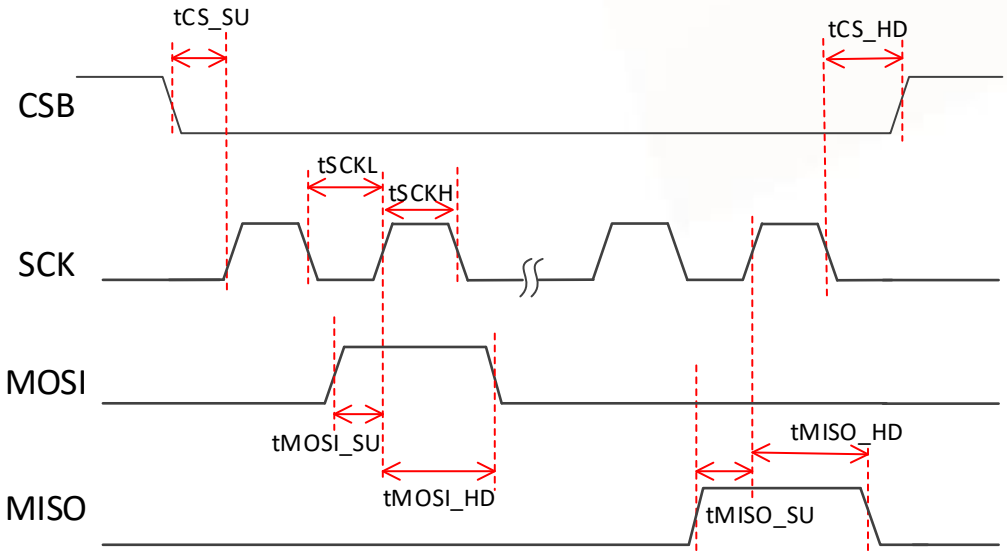


Figure 4-2. SPI master timing

Table 4-4. SPI master electrical specifications

Symbol	Description	Performance			Unit	Note
		Min	Typ	Max		
fSCK	SPI master SCK clock frequency	-	-	52	MHz	-
tMOSI_SU	MOSI to SCK rising setup time	6.6	-	-	ns	Tsck/2–Tskew-Tmargin
tMOSI_HD	SCK rising to MOSI hold time	6.6	-	-	ns	Tsck/2–Tskew-Tmargin
tSCKL	SCK low pulse	7.2	-	-	ns	Tsck/2*0.75
tSCKH	SCK high pulse	7.2	-	-	ns	Tsck/2*0.75
tCSB_SU (1)	CSB falling to SCK rising setup time	1.8	-	-	ns	Tbclk–Tskew–Tmargin
tCSB_HD (1)	SCK falling to CSB rising hold time	1.8	-	-	ns	Tbclk–Tskew–Tmargin
tMISO_SU (2)	MISO to SCK rising setup time requirement	0	-	-	ns	-
tMISO_HD (3)	SCK rising to MISO hold time requirement	0	-	-	ns	-

(1) In CS GPIO mode, SPI_CS is handled by SW. SW should pull down SPI_CS pin before SPI starts transferring and pulling up SPI_CS pin when SPI completes the transaction. Based on the sequence above, the minimum specification of tCSB_SU and tCSB_HD time can be satisfied.

(2) To achieve the min value of tMISO_SU, the internal sample clock delay of SPI master should be adjusted.

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

Note:

- **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.**

4.4.3 SPI NAND Flash Interface

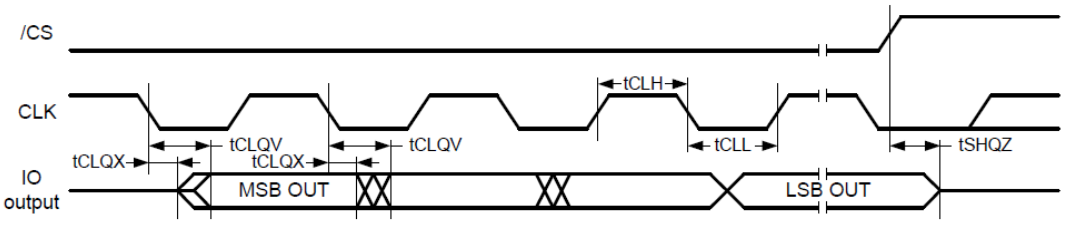


Figure 4-3. SPI NAND serial output timing

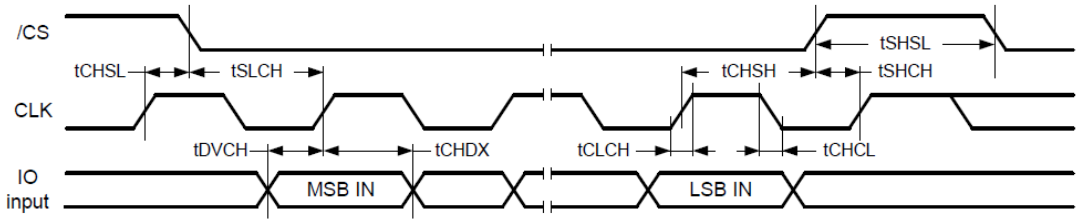


Figure 4-4. SPI NAND serial input timing

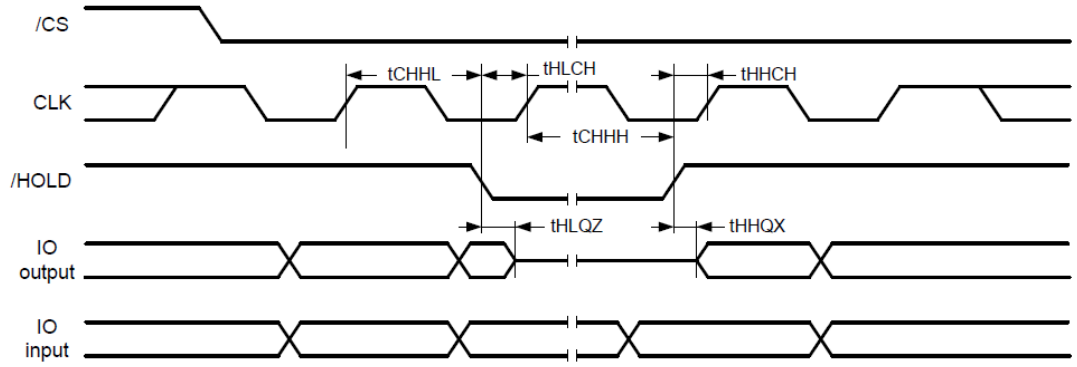


Figure 4-5. SPI NAND/HOLD timing

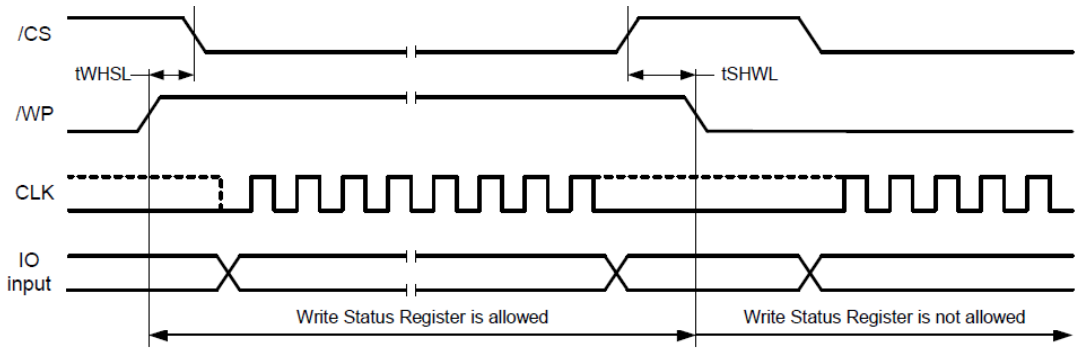


Figure 4-6. SPI NAND/WP timing

Table 4-5. 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 low	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

4.4.4 I2S/PCM Interface

Table 4-6. I2S/PCM AC timing characteristics

Parameter	Description	Min	Typ	Max	Unit
f_S	Sampling frequency	8	-	192	kHz
t_{WS}	Word select period	5.2	-	125	us
f_{MCK}	Master clock frequency	0.768	-	49.152	MHz
f_{BCK}	Serial clock frequency	0.256	-	12.288	MHz
t_{BCK_H}	BCK high-level time	-	$0.5*(1/f_{BCK})$	-	ns
t_{BCK_L}	BCK low-level time	-	$0.5*(1/f_{BCK})$	-	ns
t_{V_WS}	WS valid time	0	-	10	ns
t_{H_WS}	WS hold time	0	-	-	ns
t_{V_DO}	DO valid time	0	-	10	ns
t_{H_DO}	DO hold time	0	-	-	ns
t_{S_DI}	DI setup time	10	-	-	ns
t_{H_DI}	DI hold time	10	-	-	ns

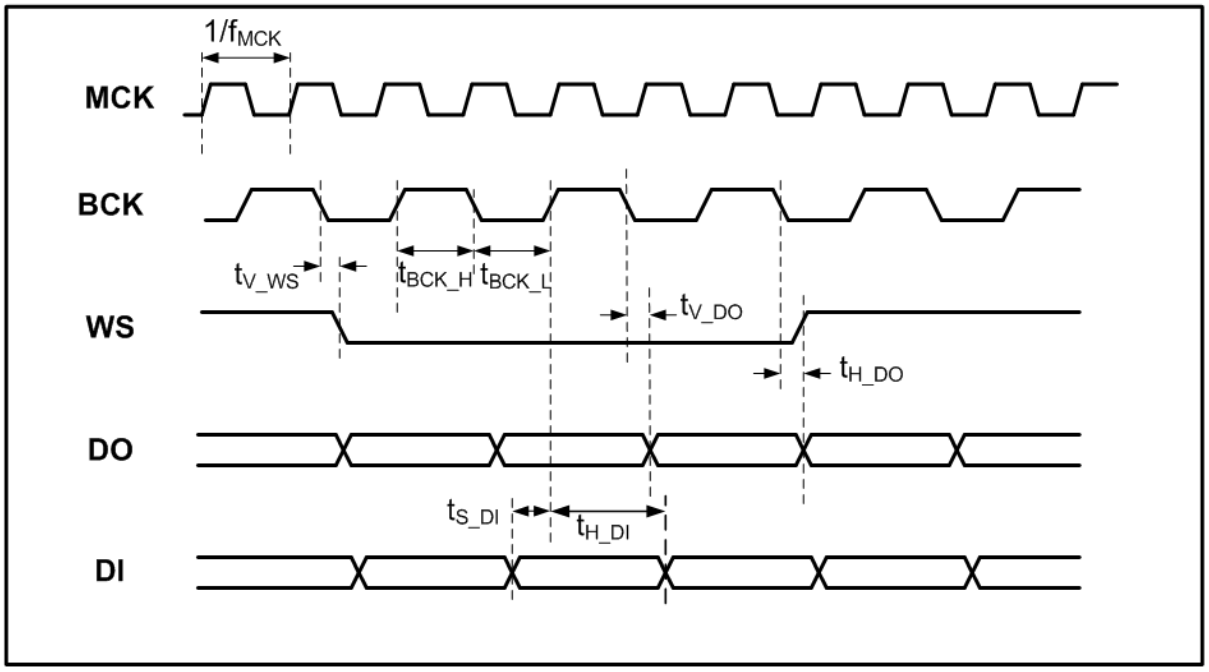


Figure 4-7 I2S/PCM master mode timing diagram

4.5 DC Electrical Characteristics

4.5.1 3.3V IO

Table 4-7. 3.3V IO electrical characteristics

Symbol	Parameter	Min	Typ	Max	Unit
VIL	Input low voltage	-0.30	-	0.83	V
VIH	Input high voltage	2.06	-	3.63	V
VOL	Output low voltage	-0.30	-	0.41	V
VOH	Output high voltage	2.48	-	3.63	V
RPU	Input pull-up resistance	10	50	100	KΩ
RPD	Input pull-down resistance	5	7.5	10	KΩ

4.5.2 1.8V IO

Table 4-8. 1.8V IO electrical characteristics

Symbol	Parameter	Min	Typ	Max	Unit
VIL	Input low voltage	-0.30	-	0.63	V
VIH	Input high voltage	1.17	-	2.10	V
VOL	Output low voltage	-	-	0.45	V
VOH	Output high voltage	1.35	-	-	V
RPU	Input pull-up resistance	40	75	190	KΩ
RPD	Input pull-down resistance	40	75	190	KΩ

4.6 Power-on Sequence

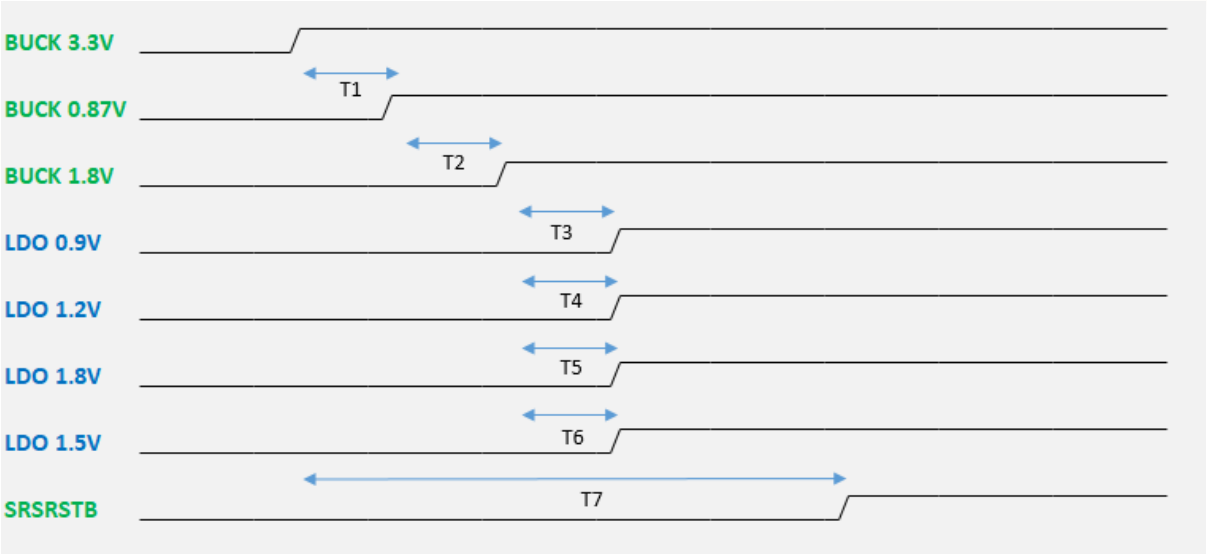


Figure 4-8. Power-on sequence

Table 4-9. Power-on sequence parameters

Symbol	Description	Min	Max	Unit
T1	BUCK_3.3V to BUCK_0.87V	0.5	5	ms
T2	BUCK_0.87V to BUCK_1.8V	0.5	5	ms
T3	BUCK_1.8V to LDO_0.9V	2	6	ms
T4	BUCK_1.8V to LDO_1.2V	2	6	ms
T5	BUCK_1.8V to LDO_1.8V	2	6	ms
T6	BUCK_1.8V to LDO_1.5V	2	6	ms
T7	BUCK_3.3V to SRSRSTB release	35	-	ms

5 MCU and Bus Fabric

5.1 External Interrupt Controller

5.1.1 Introduction

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

5.1.2 Features

EINTC supports up to 99 external interrupt signals and performs the following processes to the interrupt signals coming from external sources:

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

According to the register configuration, the external interrupt source will be forwarded to the Cortex-A53 built-in interrupt controller with different IRQ (Interrupt Request) signals.

5.1.3 Block Diagram

Figure 5-1 shows the block diagram of the external interrupt controller in the MT7981B. Every functional block is controlled by the corresponding control registers defined in Section 5.1.4.

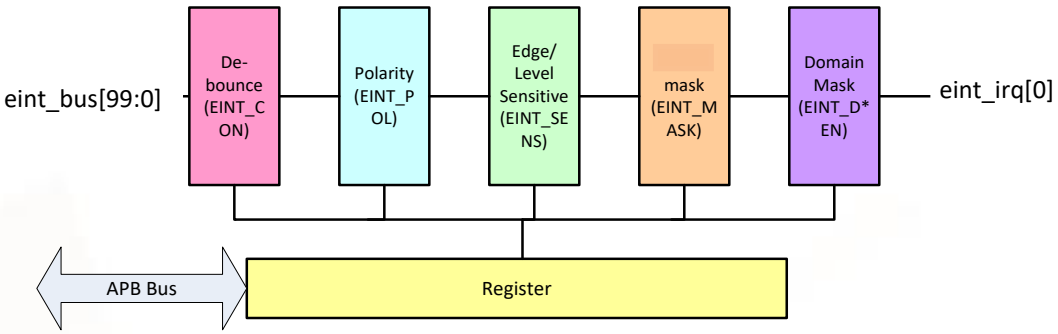


Figure 5-1. Block diagram of clock management unit and sleep controller

Normally, the external interrupt source goes through the de-bounce unit, which is driven by the 32 kHz clock and triggers the corresponding CPU with eint_irq. Therefore, the minimum latency from eint_bus to eint_irq will be 30.52 µs. Since the latency introduced by the de-bounce module may be too long for some applications, EINTC provides an

alternative path that bypasses the de-bounce module and directly triggers the interrupt signals, eint_direct_irq[7:0], to AP MCU.

5.1.4 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

5.2 System Interrupt Controller

5.2.1 Introduction

For processors like CA53, which has embedded Generic Interrupt Controllers (GICs), the part of the MCUSYS will need to keep feeding clock and Fpower 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 the system power consumption without losing interrupts.

5.2.2 Features

SYS_CIRQ supports up to 219 interrupts, which can configure the following attributes individually.

- Polarity inversion
- Edge/level trigger selection

The 219 interrupts will feed through SYS_CIRQ and connect to GIC in MCUSYS. When SYS_CIRQ is enabled, it will record the edge-sensitive interrupts and generate a pulse signal to CPU GIC when the flush command is executed.

5.2.3 Block Diagram

The following figure shows the system level block diagram of the system interrupt controller in MT7981B.

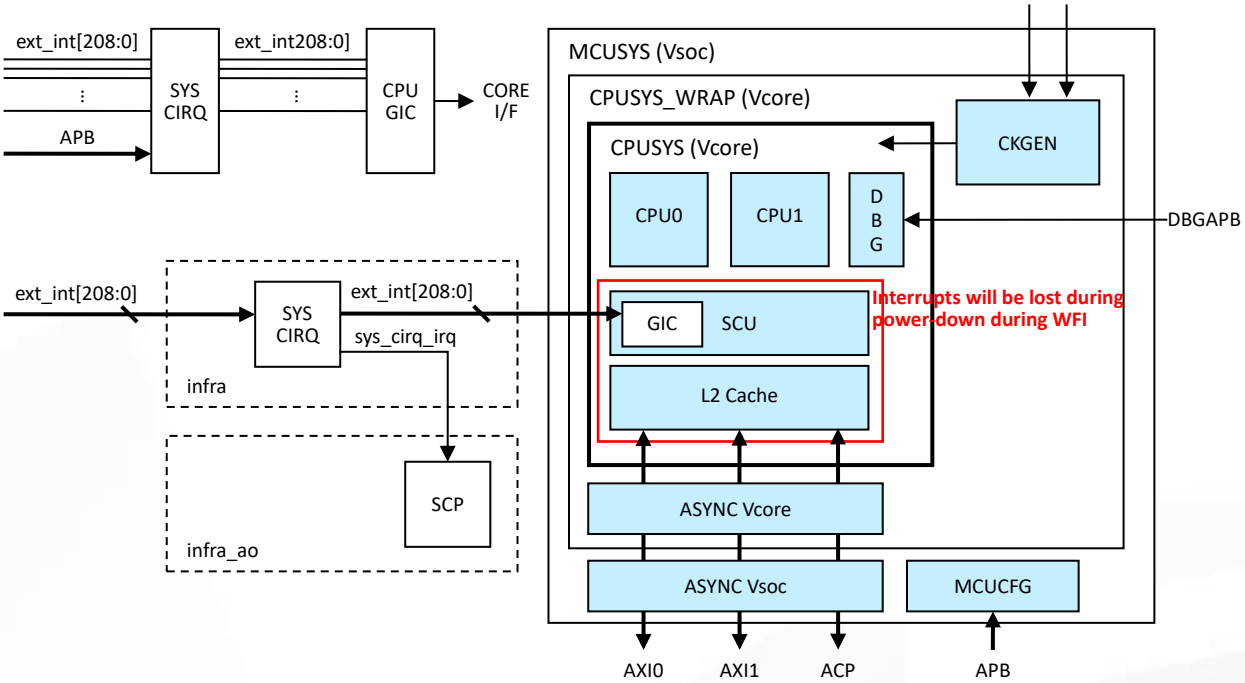


Figure 5-2. System level block diagram of system interrupt controller

The SYS_CIRQ controller is integrated in 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 (where MCUSYS GIC is active), SYS_CIRQ is disabled, and interrupts will directly issue to MCUSYS. When MCUSYS enters the sleep mode, where GIC is power downed. SYS_CIRQ controller will be enabled and monitor all edge-trigger interrupts (only edge-triggered interrupt will be lost in this scenario). When an edge-trigger interrupt is triggered, it will be recorded in the SYS_CIRQ_STA register and can be restored to GIC by SW context restore or the SYS_CIRQ flush function.

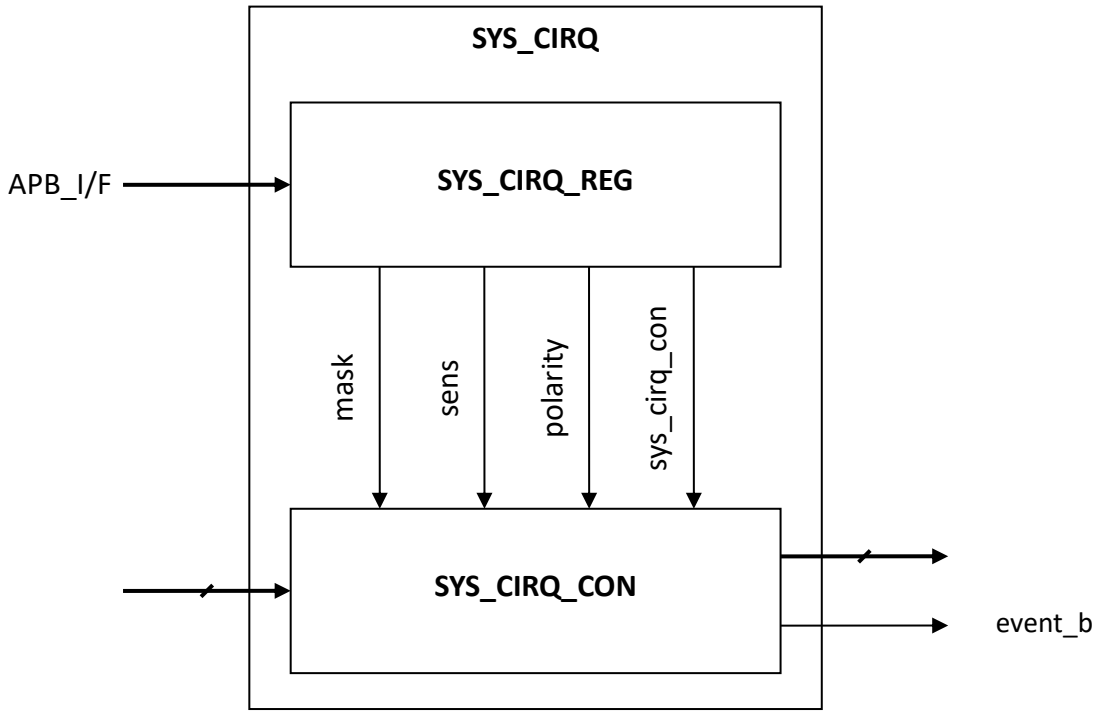


Figure 5-3. Block diagram of system interrupt controller

The above figure is the architecture of SYS_CIRQ. SYS_CIRQ_REG stores the mask/sensitivity/polarity attributes of each interrupt signal and SYS_CIRQ_CON is used to mask and detect edge-triggered interrupts.

5.2.4 Programming Guide

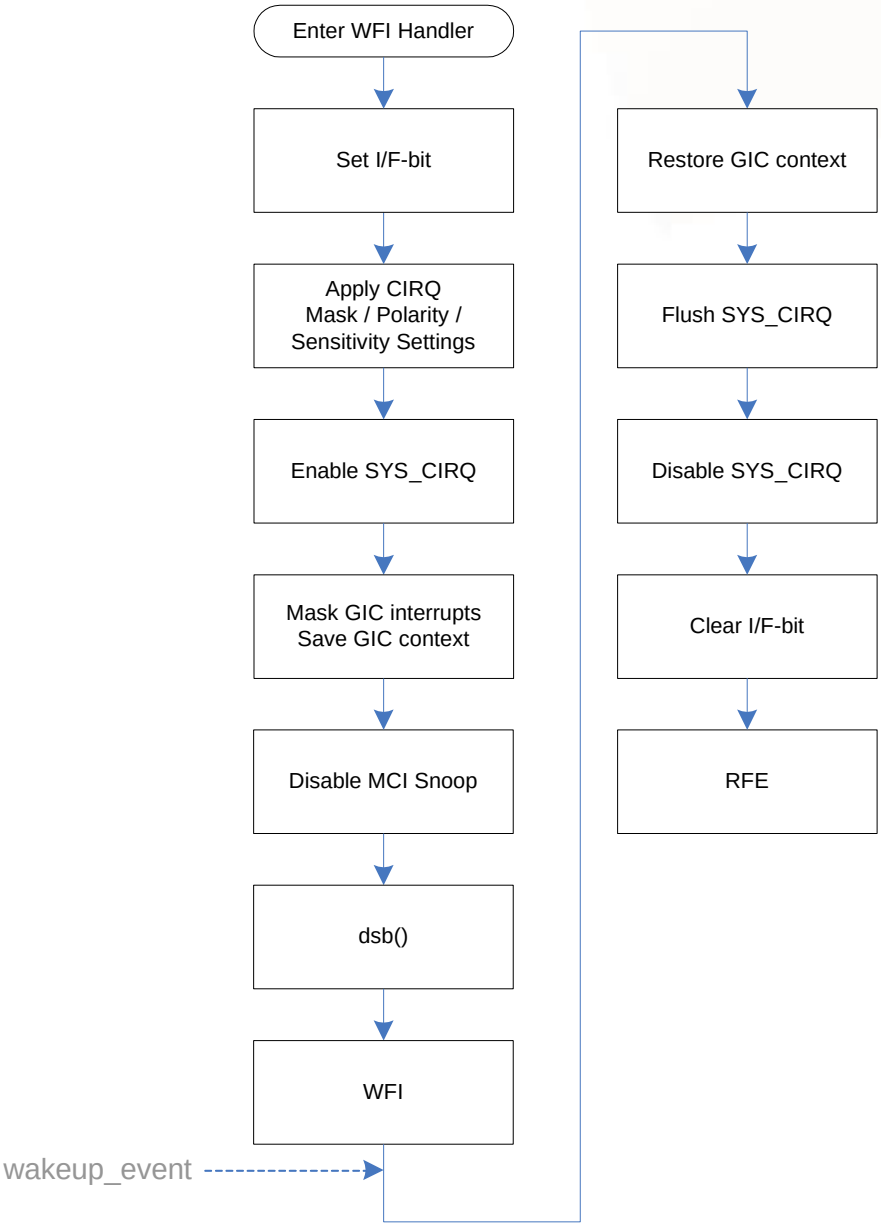


Figure 5-4. Programming guide of system interrupt controller

5.2.5 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

5.3 AP_DMA (Application Processor Direct Memory Access)

5.3.1 Introduction

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

5.3.2 Features

- Supports up to 7 channels of simultaneous data transfers
- Compliant with system bus (AXI)
- A data FIFO of 128 bits is embedded in DMA channel.
- Peripherals and channels:
  - UART x 3 (Tx/Rx channels are separated, 6 channels in total)
  - I2C x 1
  - Method to trigger DMA transmission by using the hand-shaking signal from a peripheral
- Source/Destination configuration
  - Only one side (either source or destination) is programmable; the other side can be selected as the specified peripheral.
- Burst size/burst length
  - 8 bytes/1 beat
- TrustZone
  - The corresponding channel is set as the 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 to a secure one, it can issue secure requests, and its configuration registers can only be accessed via secure masters.
- Interrupt notification
  - FIFO data are over/under a certain threshold.
- Scheduling scheme
  - Round-Robin (RR). If many channels are triggered simultaneously, the triggering priority will depend on the 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.

5.3.3 Block Diagram

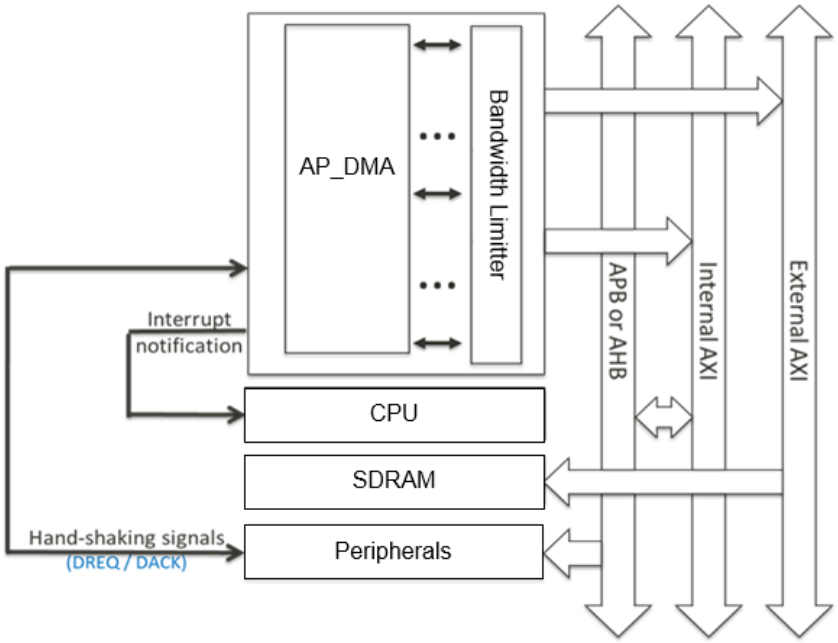


Figure 5-5. Block diagram of AP_DMA

Table 5-1. AP_DMA access matrix

AP_DMA		Source			
		SDRAM	SYSRAM	I2C	UART
Destination	SDRAM	X	X	V	V
	SYSRAM	X	X	V	V
	I2C	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 12 channels in DMA. The external AXI (Advanced eXtensible Interface) is connected to the main AXI bus fabric to provide external memory access ability. The internal AXI is connected to the peripheral AXI bus fabric to provide the connectivity for the related peripherals, e.g. I2C and UART. The configuration space of DMA can be accessed via the 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 for every DMA channel. For better throughput performance, a memory block is used as a buffer for every DMA. Thus, the data size is 8 bytes per request in an external AXI. However, for an internal AXI, the data size is only one byte per request.

5.3.3.1 UART Virtual FIFO (VFF) DMA

VFF, like the ring buffer, uses two address pointers (VFF_WPT/VFF_RPT) to control VFF condition. According to the two address pointers, two symbols (VFF_VALID_SIZE/VFF_LEFT_SIZE) are defined to represent valid data and available space in VFF.

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

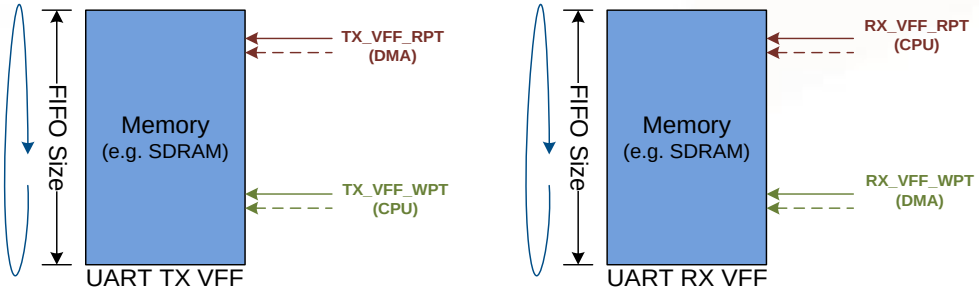


Figure 5-6. UART VFF pointers relation

5.3.3.1.1 UART Tx VFF DMA

When CPU writes n bytes of data to VFF, VFF_WPT address pointer is updated. DMA then pops data to UART FIFO from VFF with handshaking signals and updates the VFF_RPT address pointer. If UART FIFO is available and VFF_VALID_SIZE is not 0, VFF_VALID_SIZE (VFF_WPT - VFF_RPT) is the data byte stored in VFF and not yet sent to UART FIFO. VFF_LEFT_SIZE (VFF_LEN - VFF_VALID_SIZE) is the data byte, the available space, in VFF.

There is a threshold (TX_VFF_THRE) to trigger the interrupt. If VFF_LEFT_SIZE ≥ “X_VFF_THRE, the CPU can push data after receiving IRQ. In other words, if there are a lot of available space (VFF_LEFT_SIZE) in VFF, DMA will trigger IRQ, which informs the CPU to 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 will not issue read request to T VFF, even though DMA FIFO is empty. There might be an issue that if the total data size pushed to VFF by the CPU is not 8-byte aligned, the last remaining data (< 8 bytes) might be kept in VFF forever. To deal with this issue, software should set up the “TX FLUSH” register, and DMA will issue a read request to get the last remaining data in VFF.

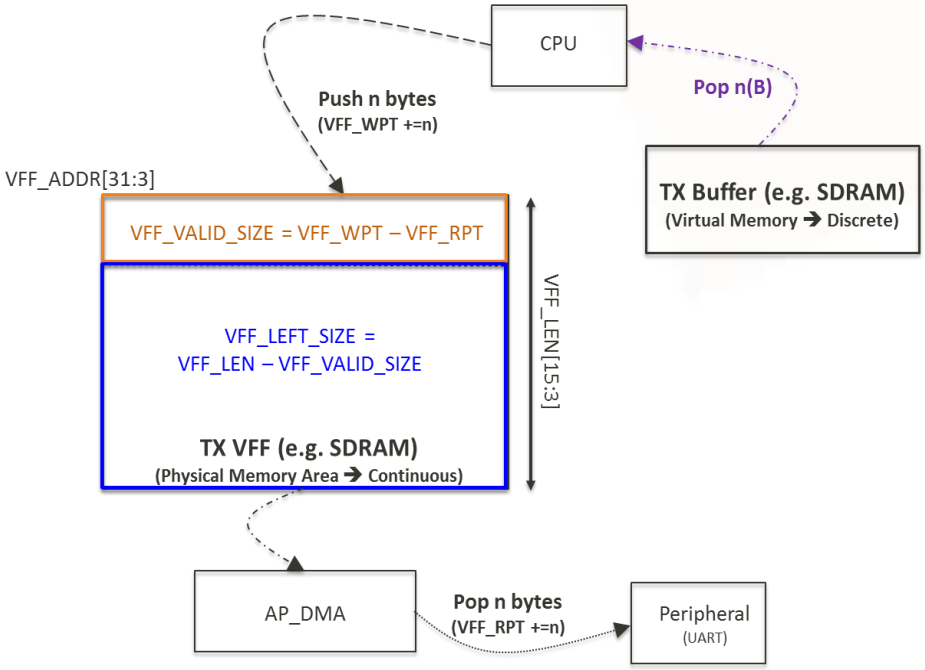


Figure 5-7. UART Tx VFF data flow diagram

5.3.3.1.2 UART Rx VFF DMA

In Rx VFF DMA mode, the 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 hand-shaking signals. VFF_RPT is updated by the CPU after CPU pops data from VFF if VFF_VALID_SIZE > 0.

There is also a threshold (RX_VFF_THRE) to trigger interrupts. If VFF_VALID_SIZE ≥ RX_VFF_THRE, the CPU can pop data after receiving IRQ. In other words, if there are a lot of valid data stored in VFF, DMA will trigger IRQ, which informs the CPU to pop data. If the CPU cannot pop data from VFF immediately, the data from UART might be lost. To avoid this, DMA can inform UART by side-band signals, and the CPU will receive IRQ if VFF_LEFT_SIZE < RX_FLOW_CTRL_THRE. Software can assert RTS if IRQ is received.

Every write request from Rx DMA to VFF is 8 bytes. If the total data from UART is not 8-byte aligned, the last remaining data (< 8 bytes) will be kept in DMA FIFO forever. There are two methods to solve this issue:

- Set up the “RX FLUSH” register manually, and DMA will issue a write request to pop the remaining data from DMA FIFO to VFF.
- Issue a flush request to DMA by UART with side-band signals when UART FIFO is empty and a timeout event happens (data is not pushed towards UART FIFO in the 4-byte baud-rate period). After the flush request is issued, the interrupt flag (FLAG1::AP_DMA_UART_x_RX_INT_FLAG[1]) will be raised.

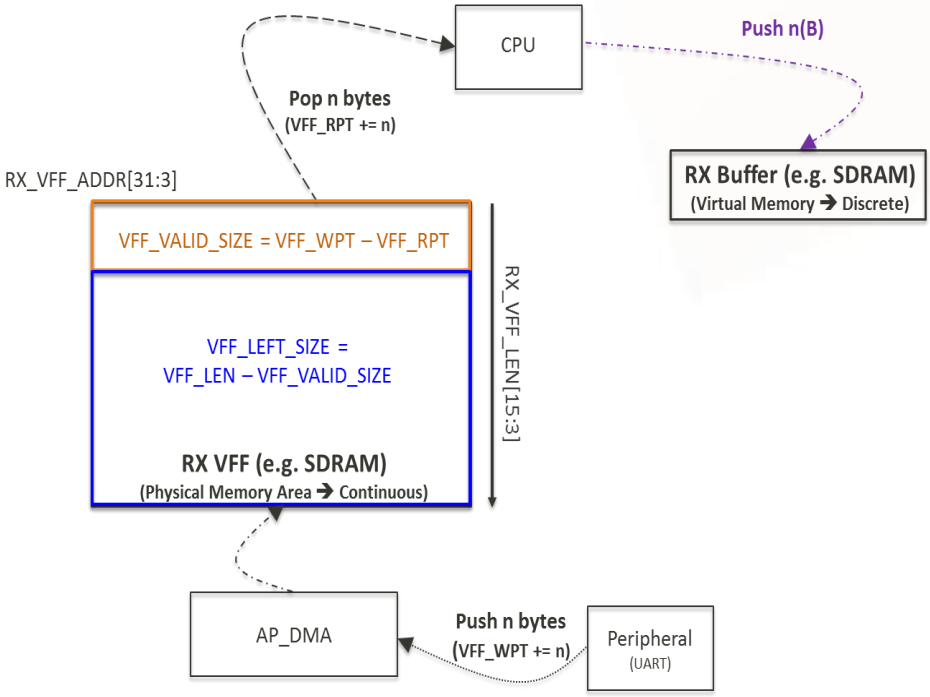


Figure 5-8. UART Rx VFF data flow diagram

5.3.3.2 Peripheral (I2C) DMA

The behavior of I2C DMA is similar to that of the command queue DMA (CQ_DMA). The major difference is that the source or destination is I2C FIFO, not SDRAM. There are hand-shaking signals (DREQ/DACK) between DMA and I2C. Because of the hand-shaking signals, the DMA channels and the corresponding I2C channels are fixed.

The DMA direction (DIR::AP_DMA_I2C_x_CON[0]) is controlled by the I2C register (SLAVE_ADDR[0]). Every DMA channel supports Tx (I2C: write bit) and Rx (I2C: read bit) direction. The transfer length and memory address register settings are also divided into Tx and Rx parts. In terms of the transfer format supported by the I2C (see 8.3 Inter-Integrated Circuit (I2C)), DMA supports all kinds of transfers, except for the “direction change (write and then read)” condition. In order to support “direction change” transfers, hardware method is required. If the I2C transfer direction changes, the side-band signal (TX2RX) will be sent to DMA when DMA_EN::CONTROL (I2Cx_Base + 0x10)[2] = 1 and DIR_CHANGE:: CONTROL(I2Cx_Base + 0x10)[4] = 1.

For example in Figure 5-9 (settings of DMA and I2C are listed in Table 5-2), the settings should be set before I2C transaction (START::START(I2Cx_Base + 0x24)[0]).

Table 5-2. AP_DMA and I2C sequential random read settings

AP DMA		I2C	
Tx transfer length (AP_DMA_I2C_x_TX_LEN[0:15)	1	1	Tx transfer length (TRANSFER_LEN[0:15]
Rx transfer length (AP_DMA_I2C_x_RX_LEN[0:15])	N	N	Rx transfer length (TRANSFER_LEN_AUX[0:15])
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:

- **The DMA direction is controlled by I2C. When I2C changes the DMA direction (as shown in Figure 5-9), it will send the side-band signal to DMA. When the DMA direction (Tx -> Rx) is changed by I2C, enable DMA again until "Rx transfer length" is received.**

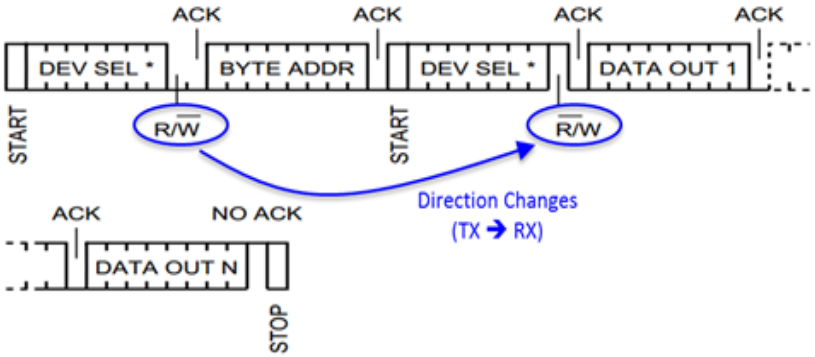


Figure 5-9. I2C sequential random read protocol

5.3.3.3 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.

5.3.3.4 Pause and Resume

Pause and resume functions are available for all DMA channels. The following table shows the mechanism of I2C channel.

Table 5-3. Example of pause and resume

Step	Address	Register Name	Local Address	R/W	Value	Description
1	AP_DMA+0x08	AP_DMA_I2C_0_EN	AP_DMA_I2C_0_EN[0]	W	1'b1	Starts DMA Program necessary settings then set EN (AP_DMA+0x08)[0] to 1.
2	AP_DMA+0x10	AP_DMA_I2C_0_STOP	AP_DMA_I2C_0_STOP[1]	W	1'b1	Pauses DMA Set PAUSE (AP_DMA+0x10)[1] to 1.
3	AP_DMA+0x10	AP_DMA_I2C_0_STOP	AP_DMA_I2C_0_STOP[1]	W	1'b0	Resumes DMA Set PAUSE (AP_DMA+0x10)[1] to 0.
4	AP_DMA+0x08	AP_DMA_I2C_0_EN	AP_DMA_I2C_0_EN[0]	R	-	Wait for DMA to finish. EN (AP_DMA+0x08)[0] will become 0, and flag will be set to 1.

Note:

- **Software can repeat steps 2 and 3 many times when DMA is running. You can monitor the idle bit to see if DMA pauses. DMA will not pause immediately and will wait for the last transaction to be finished.**

5.3.4 Programming Sequence for Different Types of Channels

5.3.4.1 Peripheral DMA (Example: I2C)

See Table 5-4 for the reference settings for the peripheral DMA.

Table 5-4. Reference settings 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 The register 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 The register 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
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	-	-	-	-	-	Wait for interrupt

Step	Address	Register Name	Local Address	R/W	Value	Description
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

5.3.4.2 Virtual FIFO DMA TX (Example: UART_TX)

See Table 5-5 for the reference settings for virtual FIFO DMA Tx.

Table 5-5. Reference settings 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 up channel warm reset
2	AP_DMA+0x0C	AP_DMA_UART_x_TX_RST	AP_DMA_UART_x_TX_RST[0]	R	-	Wait 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 up virtual size
4	AP_DMA+0x1C AP_DMA+0x54	AP_DMA_UART_x_TX_VFF_ADDR AP_DMA_UART_x_TX_VFF_ADDR2	AP_DMA_UART_x_TX_VFF_ADDR[31:3] AP_DMA_UART_x_TX_VFF_ADDR2[3:0]	W	Address	Sets up 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 enabling or not 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	-	Write pointer of byte alignment FIFO
12	Repeat steps 8 to 11.					
13	AP_DMA+0x10	AP_DMA_UART_x_TX_STOP	AP_DMA_UART_x_TX_STOP[0]	W	1'b0	Stops UART Tx virtual FIFO

5.3.4.3 Virtual FIFO DMA RX (Example: UART_RX)

See Table 5-6 for the reference settings for virtual FIFO DMA Rx.

Table 5-6. Reference settings 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 up channel warm reset
2	AP_DMA+0x0C	AP_DMA_UART_x_RX_RST	AP_DMA_UART_x_RX_RST[0]	R	-	Wait 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 up virtual size
4	AP_DMA+0x1C AP_DMA+0x54	AP_DMA_UART_x_RX_VFF_ADDR AP_DMA_UART_x_RX_VFF_ADDR2	AP_DMA_UART_x_RX_VFF_ADDR[31:3] AP_DMA_UART_x_RX_VFF_ADDR2[3:0]	W	Address	Sets up 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 enabling or not 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'b0	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	-	Read pointer of byte alignment FIFO
12	Repeat step 8 to step 11					
13	AP_DMA+0x10	AP_DMA_UART_x_RX_STOP	AP_DMA_UART_x_RX_STOP[0]	W	1'b0	Stops UART Rx virtual FIFO

5.3.5 Register Definition

Refer to "MT7981B Wi-Fi 6 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

CKSYS is responsible for generating the following clock signals:

- Free clock generation for the whole chip
- MCUSYS CPU clock
- Infrastructure and peripheral system clock, including the top level AXI fabric clock
- Ethernet DMA system clock
- DRAM reference clock

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

6.1.3 Block Diagram

There are two modules inside CKSYS. The first one is the strap controller, which is used to map lots of strap modes from chip central TAP controller. It also controls the reset to A-die and the A-die’s XTAL clock generation. The second is TOPCKCTL, which is the major module discussed in this section.

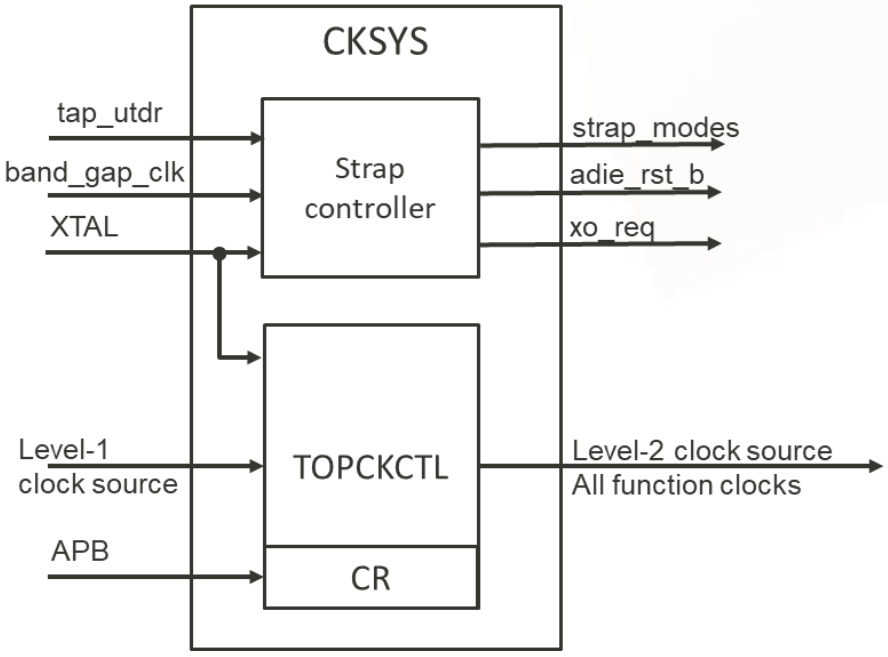


Figure 6-1. CKSYS block diagram

6.1.3.1 TOPCKCTL

TOPCKCTL is constructed by clock dividers, multiplexers (MUXs), and meter modules.

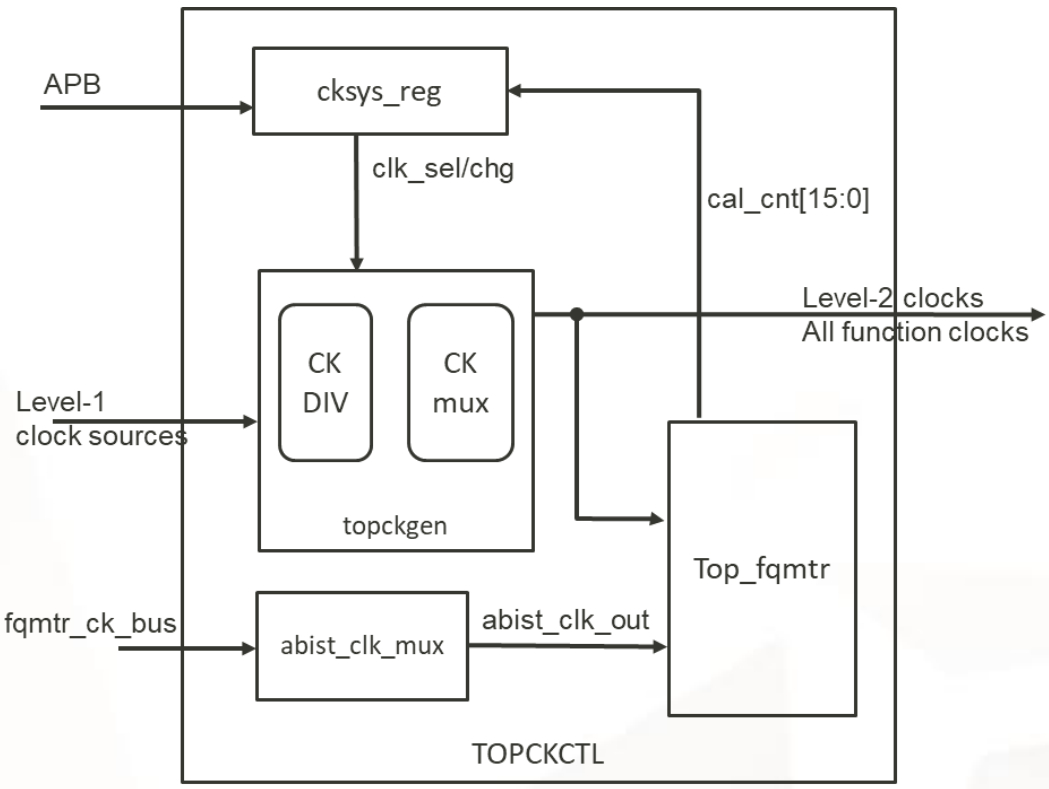
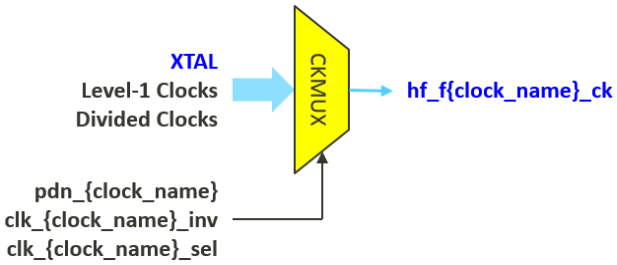


Figure 6-2. TOPCKCTL block diagram

The clock dividers divide level-1 clock sources into several divided clocks. And the number of divisions is different based on level-2 clock requirements.

There is a group of regular MUXs or buffers to choose the suitable sources from the divided clocks or level-1 clocks directly to become level-2. All the clock MUXs have three kinds of control functions (as shown in the following table). XTAL is always one of the clock MUXs' sources and also selected as the default value to avoid system hang when PLL is broken.

Table 6-1. TOPCKCTL clock multiplexer

Related Registers: **CLK_CFG_0~N** **CLK_CFG_0~N_SET** **CLK_CFG_0~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

Except for clock dividers and clock MUXs, there is one clock monitor module used to measure the generated clocks of TOPCKCTL and some chip internal clocks.

6.1.3.2 Frequency Meter

There is one frequency meter structure inside TOPCKCTL to measure the following clock groups:

- Most of level-1 clocks come from PLLs or other analog macros.
- The clock output of MUXs and buffers in TOPCKGEN is called ckgen_fmeter.

Both clock groups have one 63-to-1 clock MUX, one 1~256 clock divider, and a meter. The frequency meter related control registers are listed in the following table.

Table 6-2. TOPCKGEN meter setting

Register Name	Description
CLK_DBG_CFG	Measured clock selection
CLK_MISC_CFG_0	Clock divider selection
CLK26CALI_0~1	Frequency meter related controls

6.1.4 Register Definition

Refer to "MT7981B Wi-Fi 6 Platform Registers" for detailed register descriptions.

6.2 TOP Reset Generator Unit

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 Register Definition

Refer to “MT7981B Wi-Fi 6 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

A total of 8 PLLs are in the analog PLLGP macro: ARMPLL, NET1PLL, NET2PLL, MMPLL, MPLL, SGMIIPLL, WEDMCUPLL, APLL2. These reference clocks all come from the 40 MHz XTAL. These PLLs provide clock sources for CPU, BUS and eth.

6.3.2.2 Block Diagram

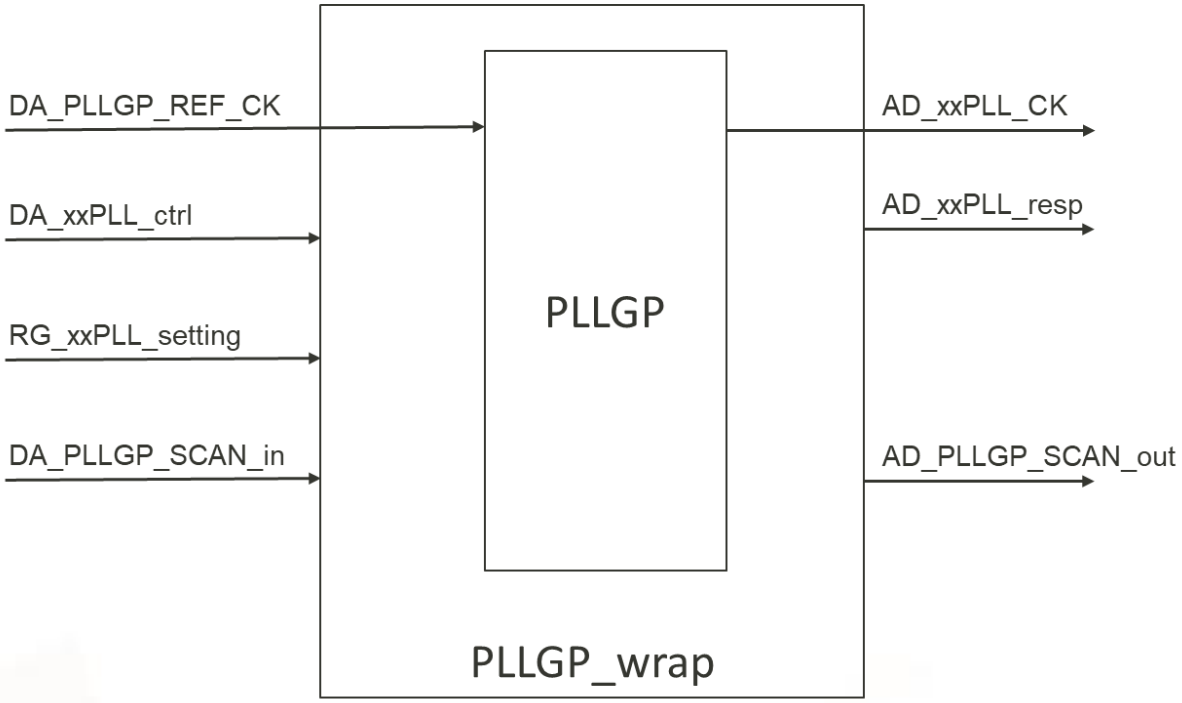


Figure 6-3. Block diagram of clock sources

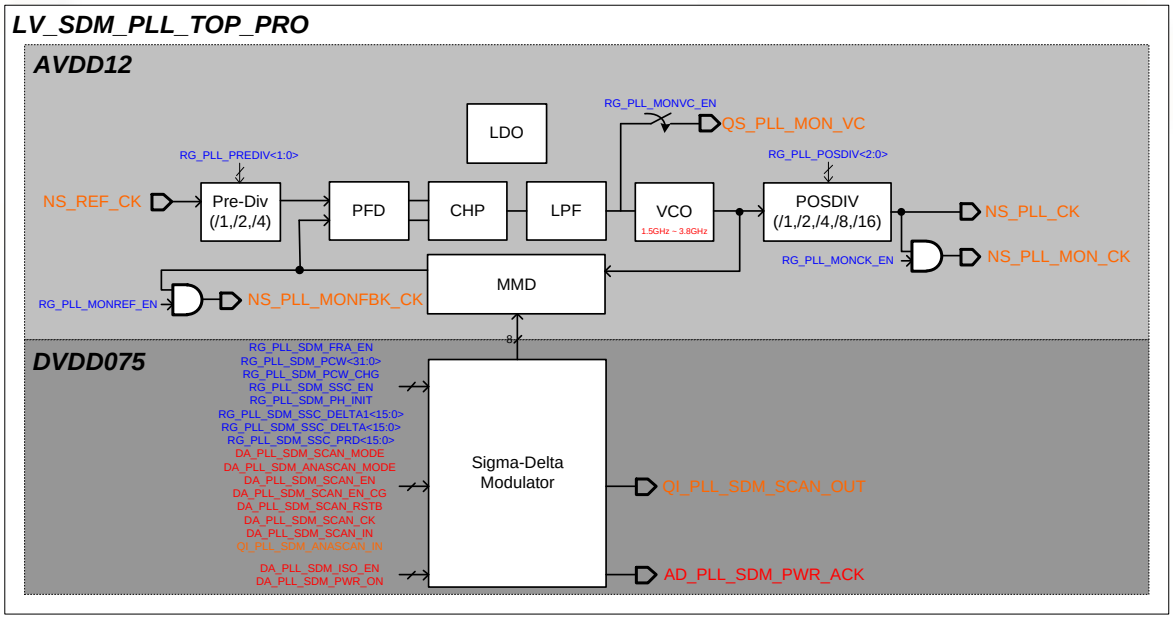


Figure 6-4. Block diagram of PLL core

6.3.2.3 Functional Specifications

See the table below for the functional specifications of PLL.

Table 6-3. PLL functional specifications

LV_SDM_PLL_TOP_V12			
Parameter	**Support**	**Unit**	**Note**
Input clock frequency **(Fin)**	20 ~ 120	MHz	Before Pre-divider
VCO clock frequency **(Fvco)**	1500 ~ 3800	MHz	Post-divider support /1, /2, /4, /8, or /16
Output clock frequency **(Fout)**	125 ~ 3800	MHz	-
Reference clock frequency **(Fref)**	20 ~ 30	MHz	(**Fin** /1, /2, or /4)
Modulus ratio	6 ~ 255	-	-
Output phase	1	-	-
Output clock duty cycle	45 ~ 55 (POSDIV: /1) 47 ~ 53 (POSDIV: others)	%	For typical voltage only
Output clock phase noise jitter	< 60ps	ps (rms)	-
Output clock period jitter	Freq < 1 GHz, < 60 Freq < 1-1.5 GHz, < 30 Freq < 1.5-2.0 GHz, < 25 Freq < 2.0-3.8 GHz, < 20	ps (pk-pk)	-
PLL bandwidth	Fref/15 or Fref/30	MHz	-
Static phase error	< ± 100ps	ps	-
Operating temperature	-40 ~ 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	-

LV_SDM_PLL_TOP_V12			
Parameter	**Support**	**Unit**	**Note**
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.

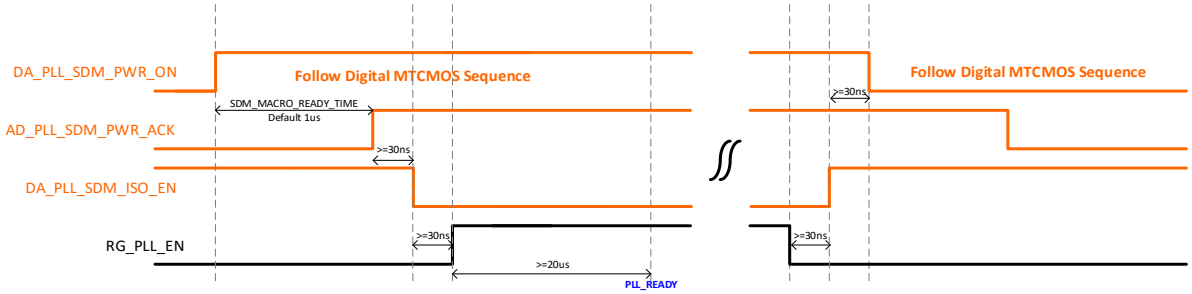


Figure 6-5. PLL power-on sequence

6.3.3 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

6.4 Thermal Controller

6.4.1 Introduction

The thermal management controls the platform computing performance to achieve the requirement and maintain the Raven within the temperature constraints. Operation under over-high temperature for a long time will have a risk of damage for Raven reliability.

In this product, it embeds several temperature sensors in possible hot spots on the die. The thermal controller module executes a periodic measurement for each hot spot. The temperature readings are readable by software. In order to minimize the software effort of temperature monitoring, the thermal controller generates interrupts to the microprocessors to notify them of the abnormal condition.

6.4.2 Features

- Supports up to 6 thermal sensors
- Periodic temperature measurement
- Temperature monitoring
- Different types of low pass filters for thermal sensor reading

6.4.3 Block Diagram

Figure 6-6 shows a basic sketch of the connection between thermal controller and the thermal sensors (on-die PNP). There are two thermal sensors. One is inside MCUSYS, and another is placed nearby base band PHY on TOP.

You can specify some pre-defined parameters to thermal controller through APB by the software. The pre-defined parameters include the information like the interval to periodically measure thermal sensors, the MUX address of thermal sensors, etc. After the temperature monitor enable command is specified, the thermal controller will be triggered and start to ask AUXADC to sample thermal sensor readings. The thermal controller sends the polling temperature requests and receives the measurement data to/from AUXADC through AHB (Advanced Microcontroller Bus). Since the thermal sensor is an analog design, the measurement results are transferred from thermal sensor to AUXADC by SADC_SIF.

The thermal controller is also designed with a thermal protection mechanism. Several thresholds can be pre-defined as shown in Figure 6-7. Once the measurement temperature exceeds certain threshold, thermal controller will issue interrupt to inform the system, so that the system temperature can be monitored and thus help prevent the system from abnormality.

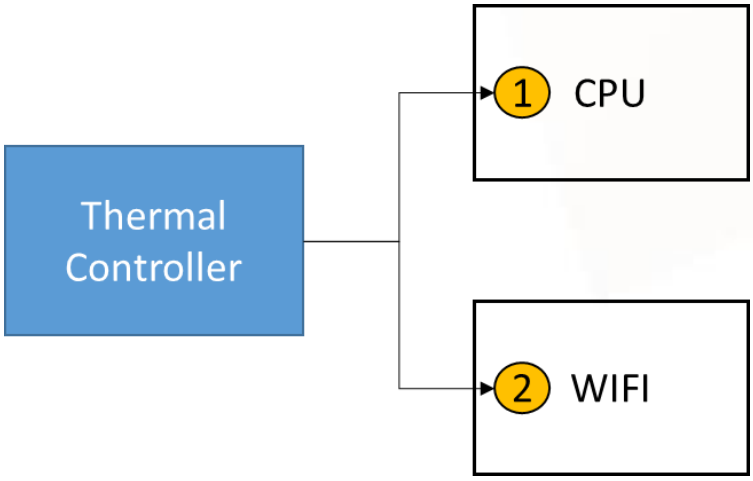


Figure 6-6. Block diagram of system temperature measurement

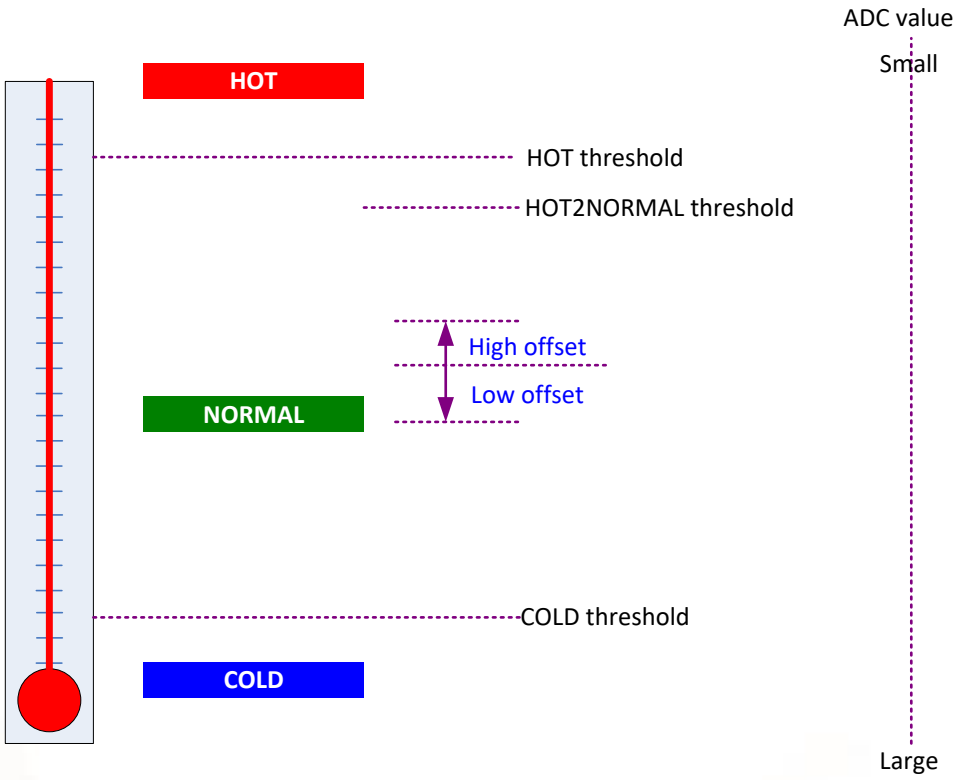


Figure 6-7. Block diagram of system temperature measurement

6.4.4 Programming Guide

:

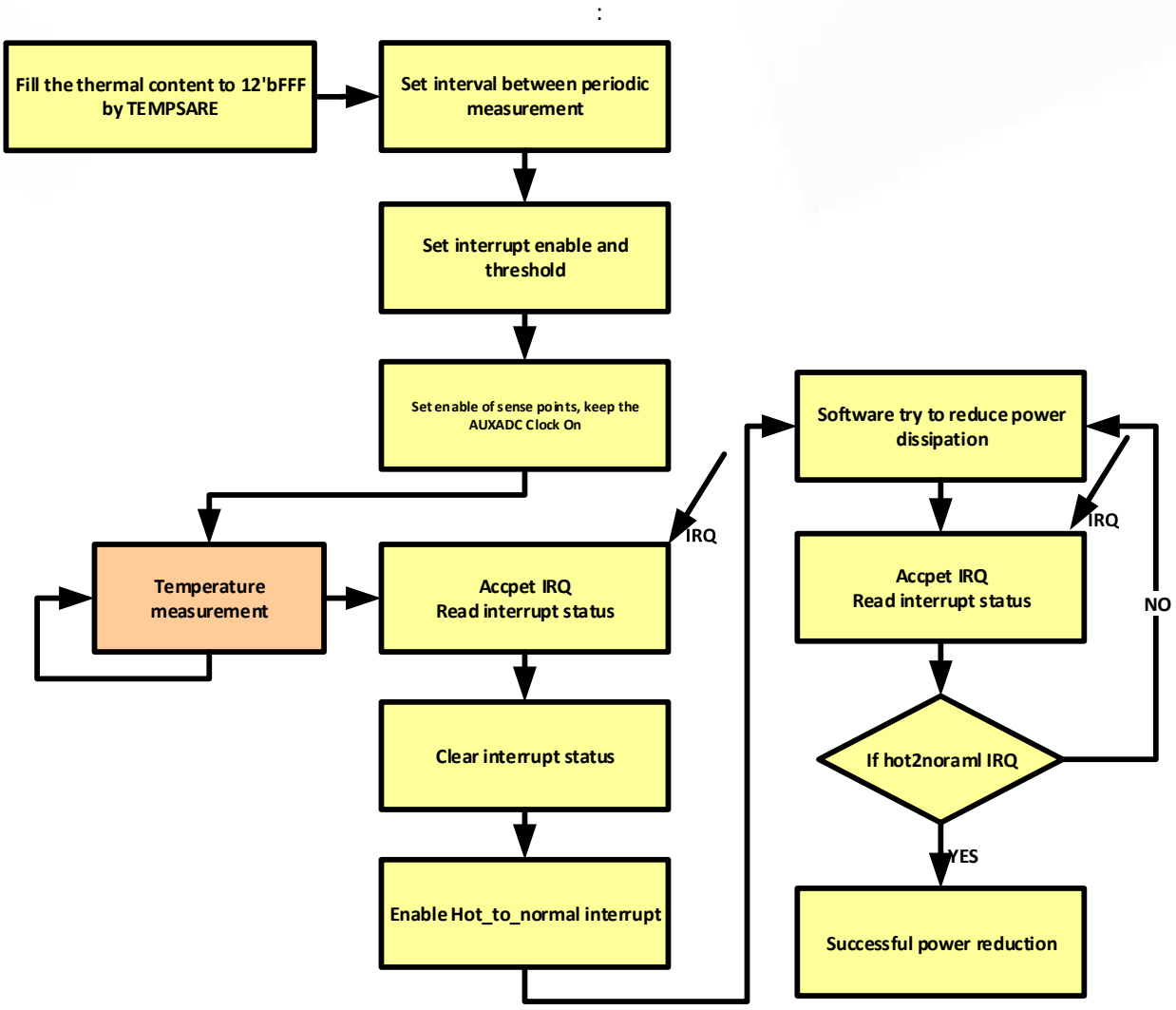


Figure 6-8. Programming flow

1. Fill the thermal content to 12'bFFF by accessing TEMPSPARE.

```
WriteREG(PTPCORESEL, 'h003F0000 + BANK_NUM);
WriteREG(TEMPMONCTL1, 'h0);
WriteREG(TEMPMONCTL2, 'h0);
WriteREG(TEMPAHBPOLL, 'h0); // Polling interval to check if temp sense is ready
WriteREG(TEMPAHBTO, 'hFF); // Exceed this polling time, IRQ would be inserted
WriteREG(TEMPSPARE0, 'h1FFF); // Set the TEMPSPARE0 register as 'h1FFF
WriteREG(TEMPPNPMUXADDR, 32'hTS_CON1); // The adxadc mux address to select to Thermal channel
WriteREG(TEMPADCENADDR, 32'TEMPSPARE1); // The adxadc enable address to trigger Thermal senser
WriteREG(TEMPADCVALIDADDR, 32'hTEMPSPARE1); // The adxadc status address to check if Thermal senser reading is valid
WriteREG(TEMPADCVOLTADDR, 32'hTEMPSPARE0); // The adxadc temperature address for the value read back from temp senser
```

WriteREG(TEMPRDCTRL, 'h0); // Use TEMPSPARE0 as valid address
WriteREG(TEMPADCVALIDMASK, 'h2c); // Set adxadc valid polarity to 0
WriteREG(TEMPMONCTL0, 'h0F); // Enable all sense points including the debug one
Wait until the content of TEMPIMMD are filled by 'hFFF

2. Set up interval between periodic temperature measurement if the MODULE clock is 66 MHz.
WriteREG(PTPCORESEL, 'h003F0000 + BANK_NUM);
WriteREG(TEMPMONCTL1, 'h3FF); // Counting unit is 1024*15.15ns=15.5 us
WriteREG(TEMPMONCTL2, 'h3FF); // Sensing interval is 1024*15.5us=15.87 ms
WriteREG(TEMPAHBPOLL, 'h0F); // Polling interval to check if temp sense is ready
WriteREG(TEMPAHBTO, 'hFF); // Exceed this polling time, IRQ would be inserted
WriteREG(TEMPPNPMUXADDR, 32'hTS_CON1); // The adxadc mux address to select to Thermal channel
WriteREG(TEMPADCENADDR, 32'hAUXADC_CON1); // The adxadc enable address to trigger Thermal senser
WriteREG(TEMPADCVALIDADDR, 32'hAUXADC_CON3); // The adxadc status address to check if Thermal senser reading is valid
WriteREG(TEMPADCVOLTADDR, 32'hAUXADC_DAT11); // The adxadc temperature address for the value read back from temp senser
WriteREG(TEMPRDCTRL, 'h0); // Use AUXADC_DAT11 as valid address
WriteREG(TEMPADCVALIDMASK, 'h2c); // Set adxadc valid polarity to 0

3. Set up monitoring threshold and SPM wakeup event.
WriteREG(TEMPHTHRE, 'hxxx); // Set hot threshold
WriteREG(TEMPCTHRE, 'hxxx); // Set cold threshold
WriteREG(TEMPCTHRE, 'hxxx); // Set hot to normal threshold
WriteREG(TEMPPROTCTL, 'h20xxx); // Set hot to wakeup event control
WriteREG(TEMPPROTTC, 'hxxx); // Set hot to HOT wakeup event
WriteREG(TEMPMONINT, 'h8000001F); // Enable interrupt

4. Enable sensing points.
WriteREG(TEMPMONCTL0, 'h07); // Enable all three sense points

5. Accept IRQ.
ReadREG(TEMPMONINTSTS); // Read interrupt and clear interrupt status

6. Read temperature readings. (optional)
ReadREG(TEMPMSR0); // Read temperature reading of sense point 0
ReadREG(TEMPMSR1); // Read temperature reading of sense point 1
ReadREG(TEMPMSR2); // Read temperature reading of sense point 2

7. Release pause of periodic temperature measurement
WriteREG(TEMPMSRCTL1, vReadREG(TEMPMSRCTL1) & 0xFFFE);

Immediate temperature measurement:

After each immediate is done, the software should disable the immediate mode.

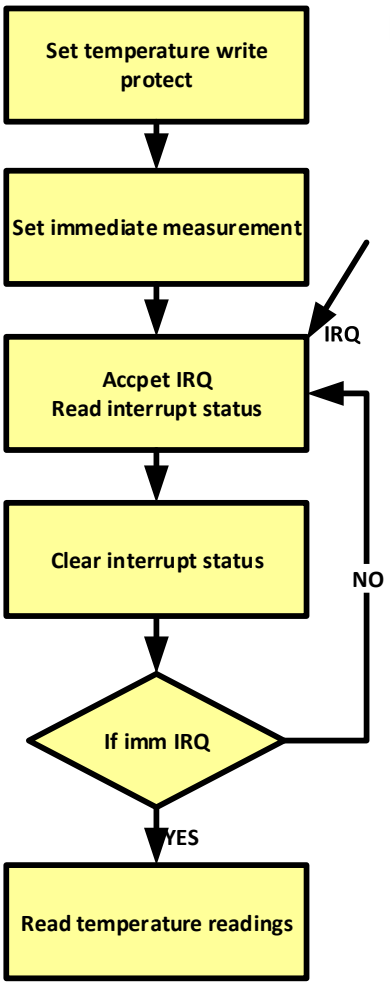


Figure 6-9. Immediate measurement programming flow

6.4.4.1 Interrupt Control Flow

The interrupt condition of high and low temperature monitoring is shown in Figure 6-12.

The software will accept interrupts when the following three conditions occur. The software determines which temperature sensor is to be monitored. Once the condition in any one of the three temperature sensors occurs, the interrupt will be issued.

In Figure 6-11:

- Cold interrupt: When the temperature decreases to lower than the cold threshold from the normal temperature range, it means when the state of NORMAL is transferred into the state of COLD.
- Hot interrupt: When the temperature increases to higher than the hot threshold from the temperature below the hot threshold.

The state of VERY_HOT indicates that temperature is higher than the hot threshold. The state of HOT1 indicates that the temperature is higher than the hot to normal threshold. NORAML state cannot be transferred into VERY_HOT directly.

- Hot to normal temperature interrupt: When HOT2 state is transferred into the NORMAL state.

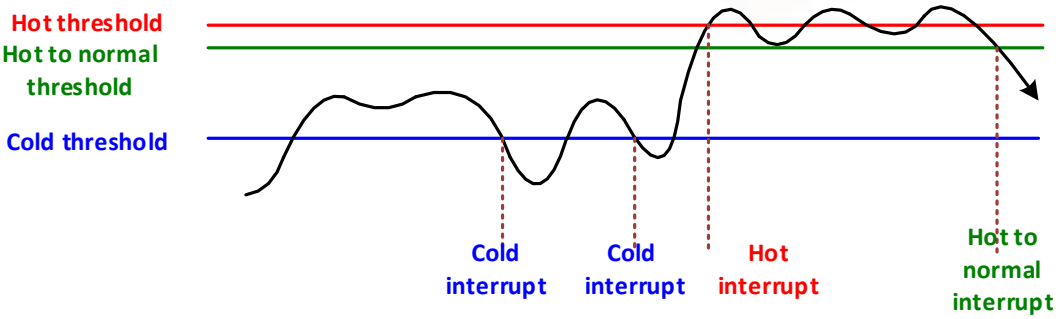


Figure 6-10. Interrupt condition of high/low temperature monitoring

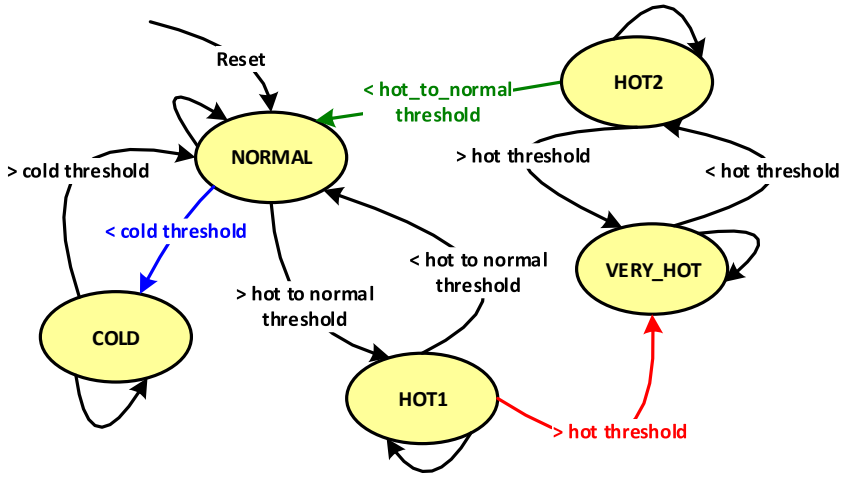


Figure 6-11. Finite state machine of high/low temperature monitoring

In Figure 6-12, when the software immediate measurement is enabled, the state will maintain the current state until the software disables the immediate mode. It is shown as the "*" mark.

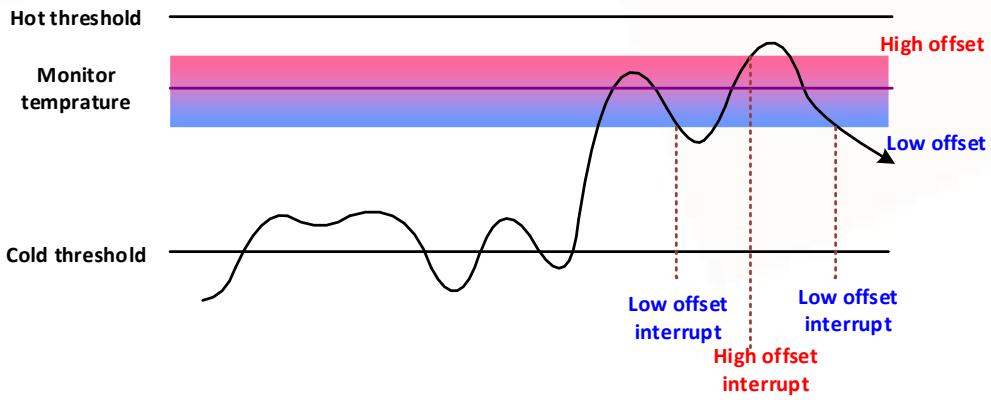


Figure 6-12. Interrupt condition of high/low offset monitoring

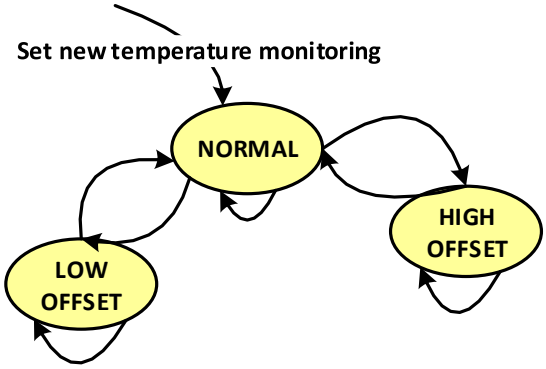


Figure 6-13. Finite state machine of high/low offset monitoring

6.4.5 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

7 General System

7.1 General-Purpose Timer (GPT)

7.1.1 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).

7.1.2 Features

The four operation modes of GPT are ONE-SHOT, REPEAT, KEEP-GO and FREERUN. For the details, please refer to Table 7-1.

Table 7-1. 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…
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.**

7.1.3 Block Diagram

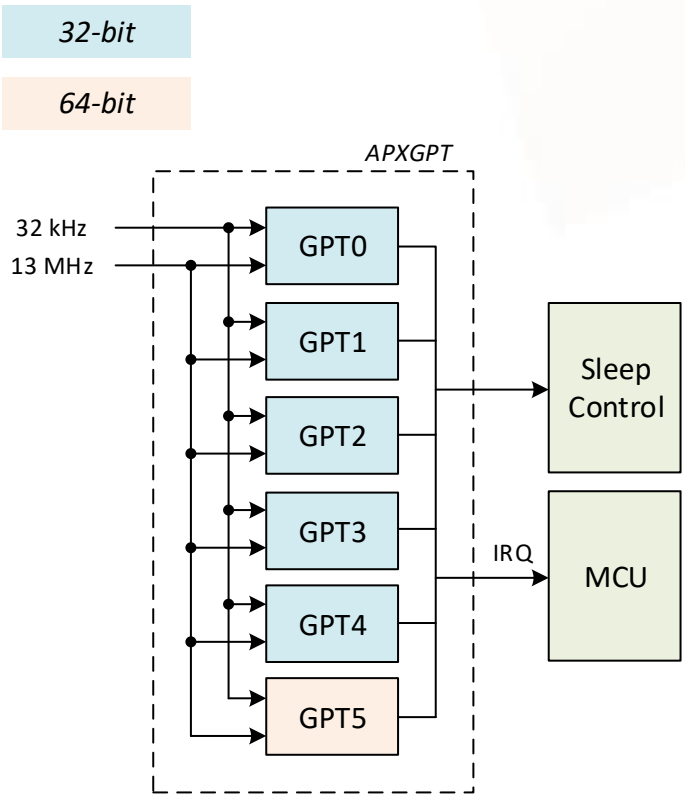


Figure 7-1. Block diagram of APXGPT

7.1.4 Theory of Operations

For the GPT6 64-bit timer, the read operation of the 64-bit timer value will be separated into two APB reads since an APB read is of 32-bit width. To perform the read of 64-bit timer value, the lower word should be read first and then the higher word. The read operation of lower word will freeze the “read value” of the higher word but will not freeze the timer counting. This ensures that the separated read operation acquires the correct timer value.

To program and use GPT, note that:

- The counter value can be read at any time when the clock source is system clock.
- The counter value can be read at any time even when the clock source is RTC clock.
- The comparative value can be programmed at any time. When the comparative value was rewritten during count operation, counter would be reset to 0 and restart count.

7.2 System Timer (sys_timer)

7.2.1 Introduction

Sys_timer is a 64-bit, non-stop, always-on up-counter that 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.).

7.2.2 Features

The sys_timer supports the following features:

- A 64-bit, always-on up-counter (this counter is 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
- HW counter incremented compensation when switching to 32 kHz clock source
- 8 x 32-bit counter timeout value (read as 32-bit down counter)
- Security access permission control for each control register (with one-time lock bit)

7.2.3 Block Diagram

The block diagram of sys_timer is shown in Figure 7-2.

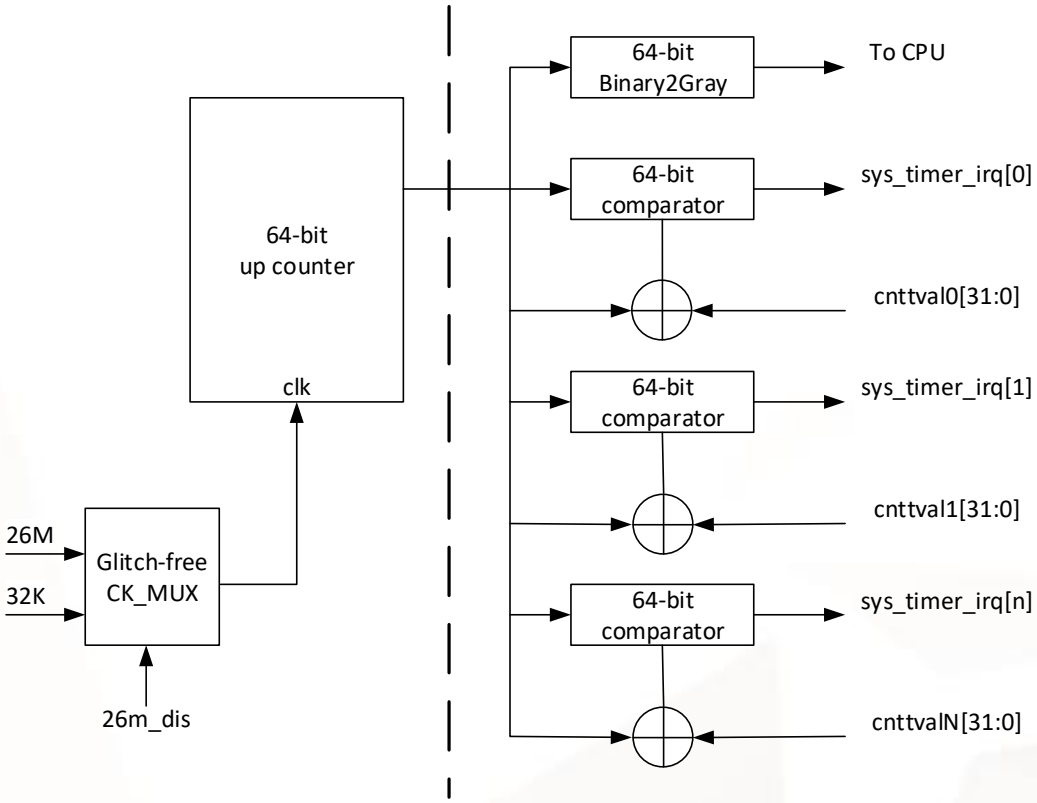


Figure 7-2. sys_timer block diagram

7.2.4 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 will start ticking with 13 MHz clock period after reset is released. It can also be programmed to tick with 26 MHz, 13 MHz, or 6.5 MHz clock period and switched to 32kHz 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 the programmer to set the 32-bit counter's timeout values, which can trigger the interrupts after timeout. When the programmer writes 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 the following figure.

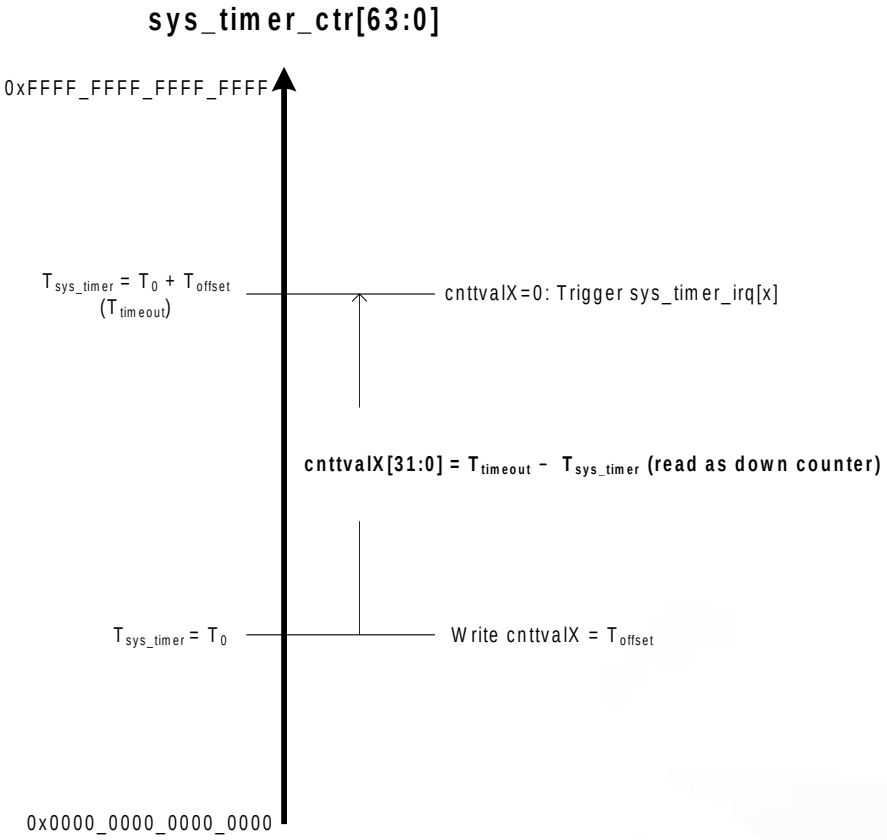


Figure 7-3. Behavior of sys_timer counter timeout value

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.

7.2.5 Programming Guide

This section describes the following operating sequence:

Take sys_timer0 as an example to show the programming sequence.

Table 7-2. 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

7.2.6 Register Definition

Refer to "MT7981B Wi-Fi 6 Platform Registers" for detailed register descriptions.

7.3 Audio

7.3.1 Introduction

The audio system provides the ability to exchange audio data. In MT7981B, the audio system includes an Enhanced-Time-Division-Multiplexer (eTDM) interface capable of providing the Pulse-Code Modulation (PCM) and Inter-IC Sound (I2S) interfaces. The eTDM interface transmits DRAM data into the PCM/I2S data format.

7.3.2 Features

- eTDM interface
  - Support Master input/output mode
  - Support sample rates: 8 kHz/12 kHz/16 kHz/24 kHz/32 kHz/48 kHz/96 kHz/192 kHz
  - Support I2S/PCM format
  - Support 16-bit/24-bit/32-bit precision
  - Support 16-bit/32-bit channel width
  - Support 2/4/6/8/12/16/24/32 channel number, but only the first 2CH contain data
  - eTDM interface signal description is shown in the following table.
  - PCM_CLK: bit clock. And the BCLK clock rate is listed in the following table.
  - 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 7-3. 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 7-3), max. frequency <= 12.288 MHz
PCM_DO	Output	TX data

Table 7-4. 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

7.3.3 Block Diagram

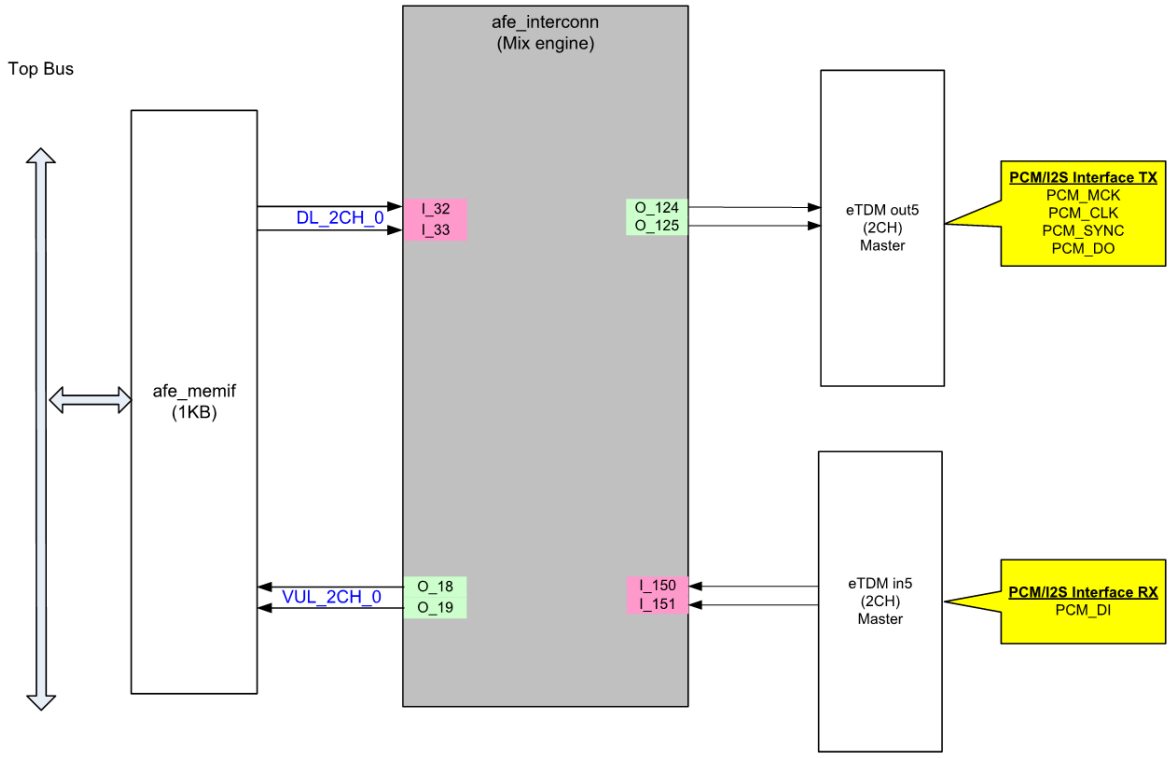


Figure 7-4. Audio block diagram

7.3.4 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

8 Peripherals

8.1 Serial Peripheral Interface (SPI) Master

8.1.1 Introduction

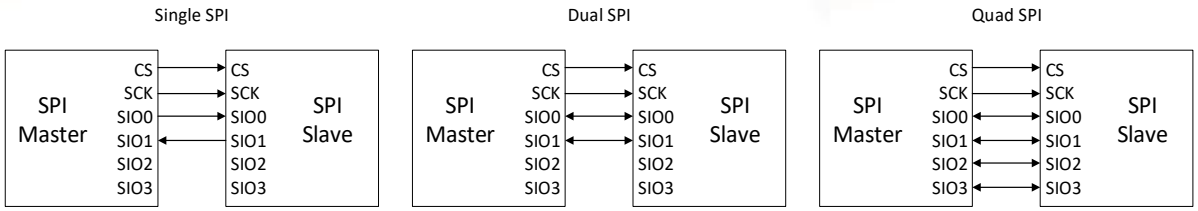


Figure 8-1. Pin connection between SPI master and SPI slave

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 8-1 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 8-1. 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 section are listed in Table 8-2.

Table 8-2. Abbreviations for SPI master

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

8.1.2 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 8-2)

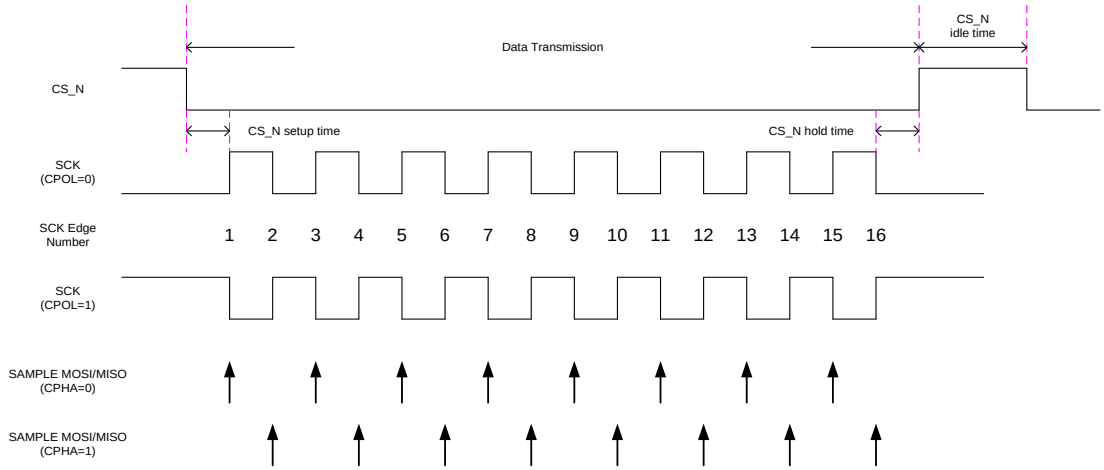


Figure 8-2. SPI transmission formats

- 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 8-3 for the state transition.

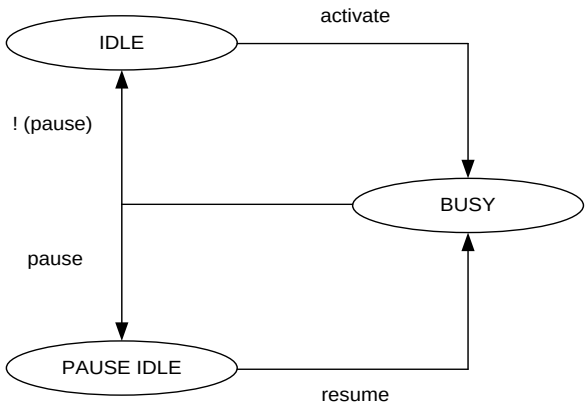


Figure 8-3. Operation flow with or without PAUSE mode

- Configurable option to control CS_N deassert between byte transfers. The controller supports a special transmission format called CS_N deassert mode. Figure 8-4 illustrates the waveform in this transmission format.

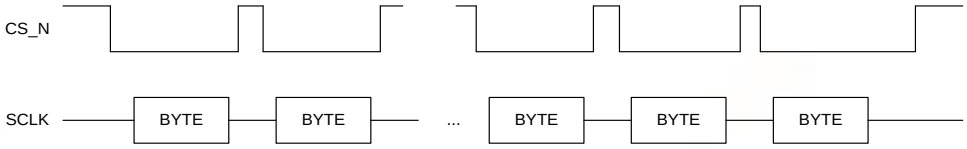


Figure 8-4. CS_N deassert mode

- 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).

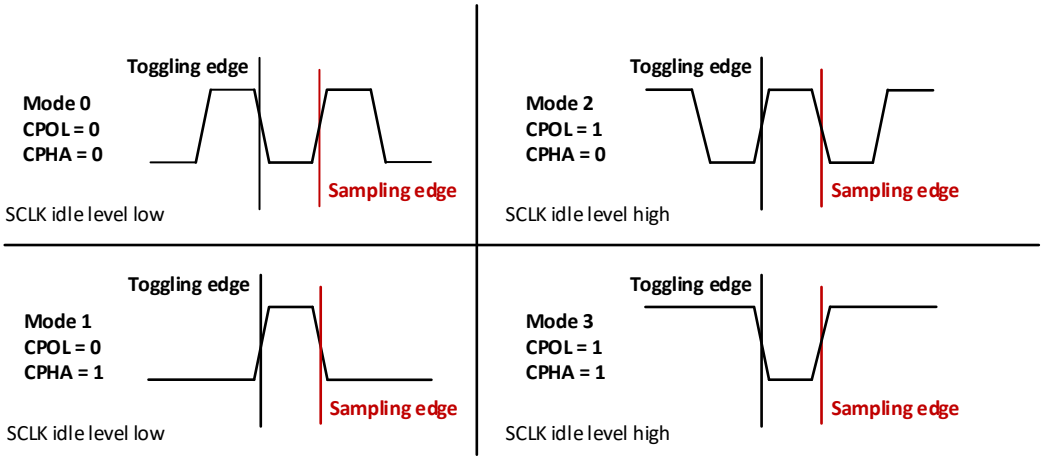


Figure 8-5. Waveforms of four communication modes

- 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 8-6 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.

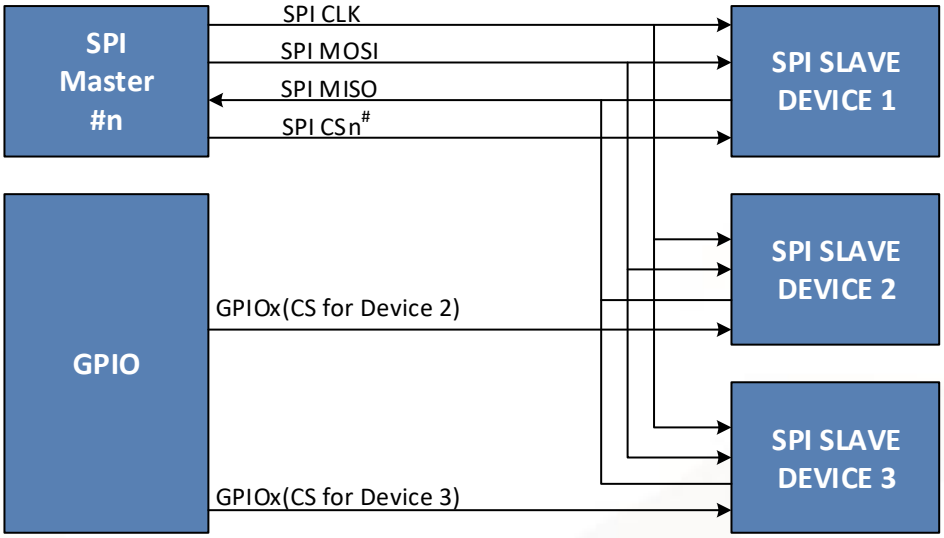


Figure 8-6. Pin connection between one master and multiple slaves in single mode

8.1.3 Block Diagram

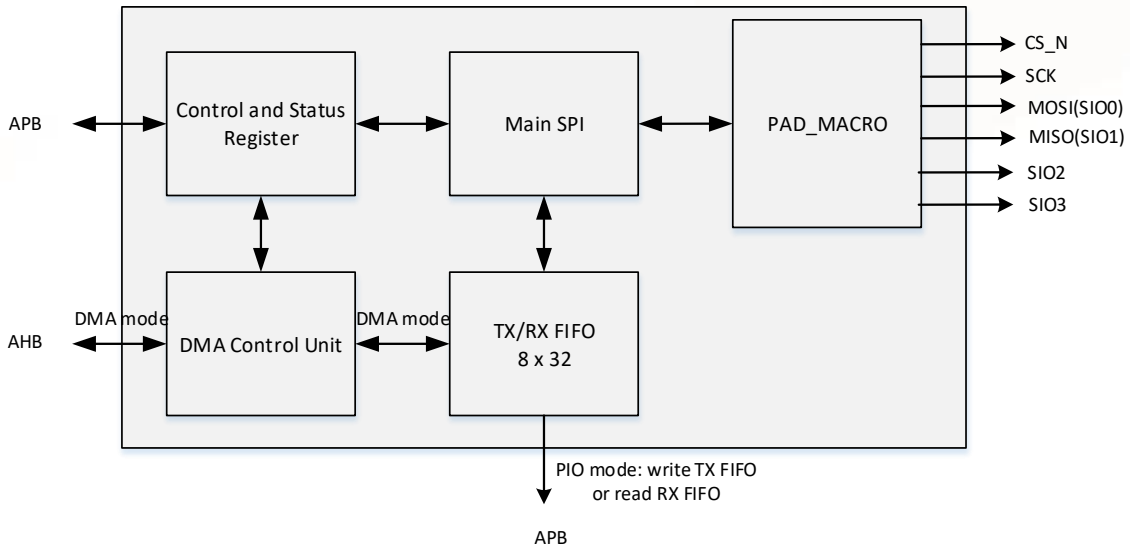


Figure 8-7. Block diagram of SPI master

See Figure 8-7. 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 8-1 for pin descriptions. The SPI master I/O operates at 1.8V.

8.1.4 Theory of Operation

8.1.4.1 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.

8.1.4.1.1 SPI Single Mode and Full Duplex Mode Transaction

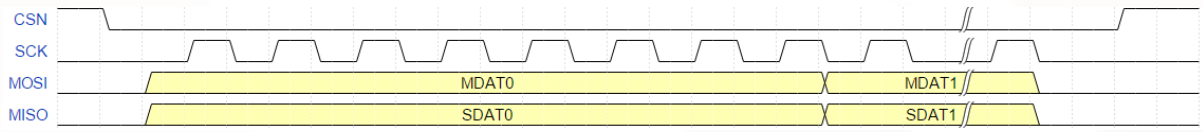


Figure 8-8. SPI single mode, full duplex mode transaction formats

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 8-8 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.

8.1.4.1.2 SPI Single Mode and Half Duplex Mode Transaction

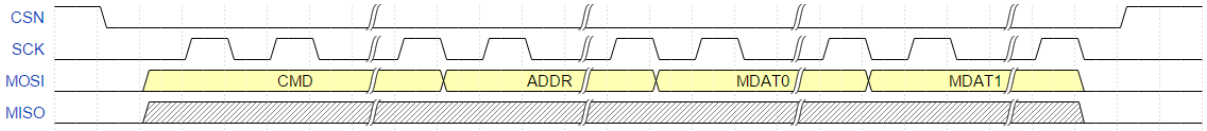


Figure 8-9. SPI single mode, half duplex mode, TX transaction formats

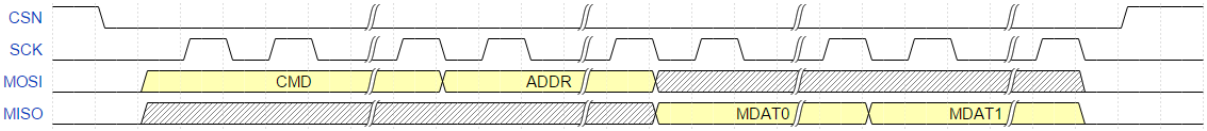


Figure 8-10. SPI single mode, half duplex mode, RX transaction formats

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 8-9 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 8-10 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.

8.1.4.1.3 SPI Dual Mode and Half Duplex Mode Transaction

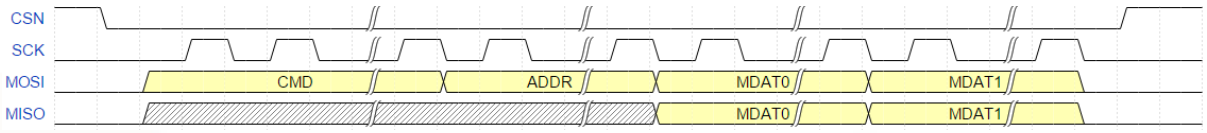


Figure 8-11. SPI dual mode, half duplex mode, TX transaction formats

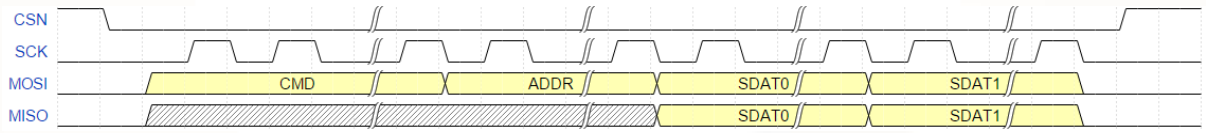


Figure 8-12. SPI dual mode, half duplex mode, RX transaction formats

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 8-11 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 8-12 for the waveform on the SPI pins.

8.1.4.1.4 SPI Quad Mode and Half Duplex Mode Transaction

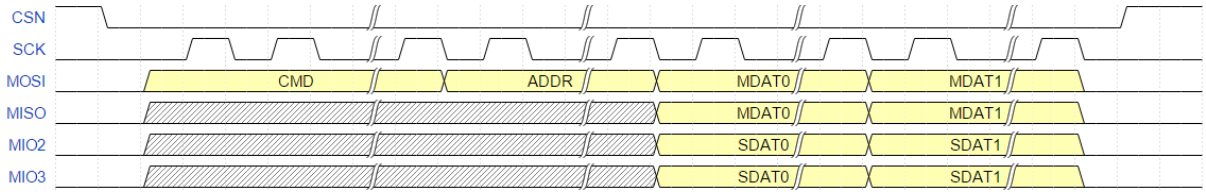


Figure 8-13. SPI dual mode, full duplex mode transaction format

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 8-11 for the waveform on the SPI pins. The SPI master needs two more pins if it works in dual mode and full duplex mode.

8.1.4.1.5 SPI Quad Mode and Half Duplex Mode Transaction

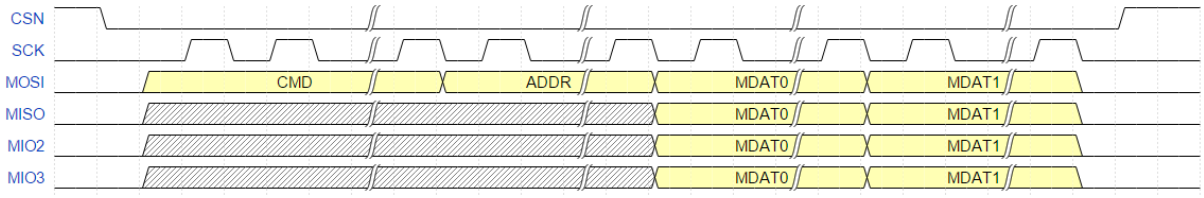


Figure 8-14. SPI quad mode, half duplex mode, TX transaction formats

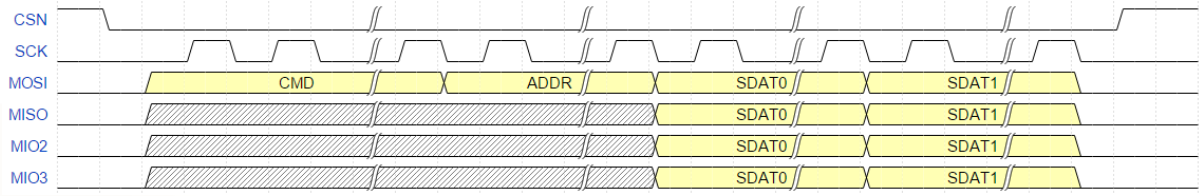


Figure 8-15. SPI quad mode, half duplex mode, RX transaction formats

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 8-14 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 8-15 for the waveform on the SPI pins.

8.1.4.2 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 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 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].

8.1.5 Programming Guide

8.1.5.1 SPI DMA Mode

Table 8-3. 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”	-	-

8.1.5.2 SPI FIFO Mode

Table 8-4. 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]

8.1.5.3 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

8.2 NAND Flash Interface (NFI)

8.2.1 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 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.

8.2.2 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

8.2.3 Block Diagram

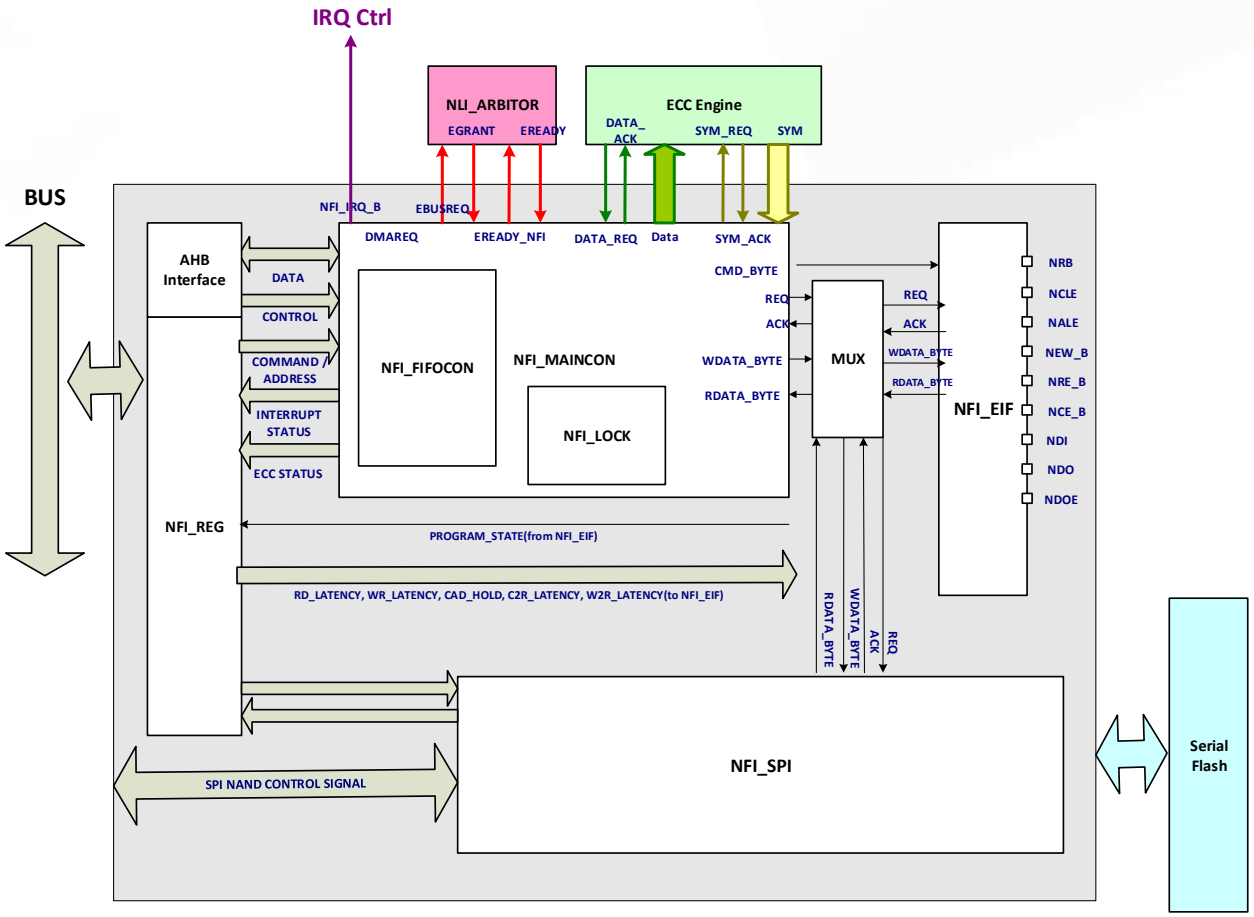


Figure 8-16. NFI block diagram

NAND flash controller uses APB slave bus for accessing register configuration and data read/write, and uses 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 standard protocol for communication with NAND device.

8.2.4 Programming Guide

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

8.2.4.1 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; trigger = 0
int gpram_data = *SPI_GPRAM_ADDR	// Read gpram data (ID)

8.2.4.2 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; trigger = 0

8.2.4.3 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; trigger = 0

8.2.4.4 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

8.2.4.5 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

8.2.4.6 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

8.2.5 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

8.3 Inter-Integrated Circuit (I2C)

8.3.1 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.

8.3.2 Features

The features of I2C are as follows:

- I2C compliant master mode operation
- Adjustable clock speed for Standard mode operations
- 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

8.3.2.1 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.

8.3.2.2 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 8-17 to Figure 8-19 illustrate the transfer format types supported through different software configurations.

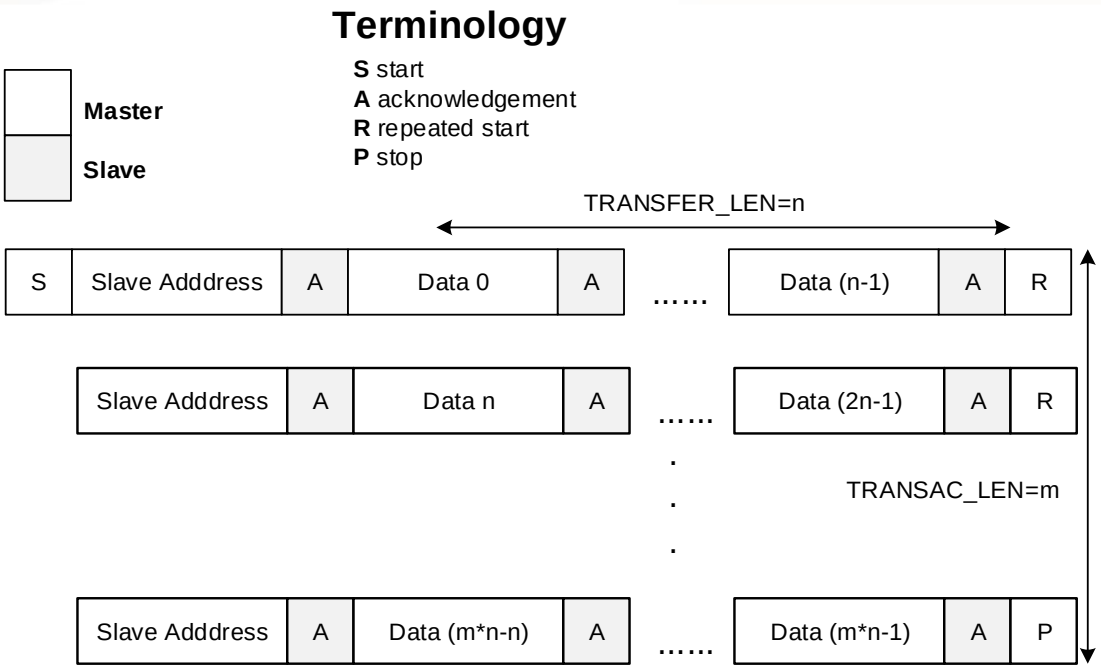


Figure 8-17. I2C transaction (write)

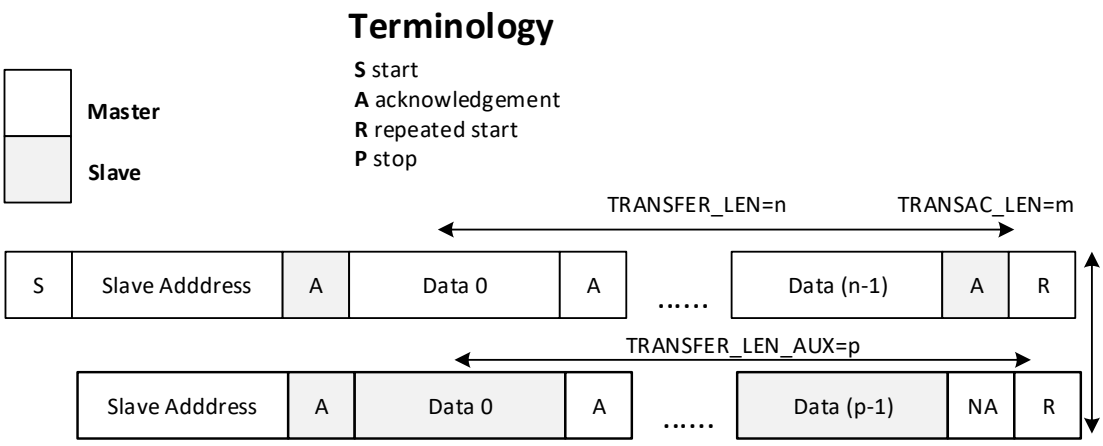


Figure 8-18. I2C transaction (read after write, m = 2)

Terminology

Master

Slave

S start
NA non-acknowledgement
R repeated start
P stop
I²C Transaction mentioned before

Figure 8-19. I2C high speed mode

8.3.3 Block Diagram

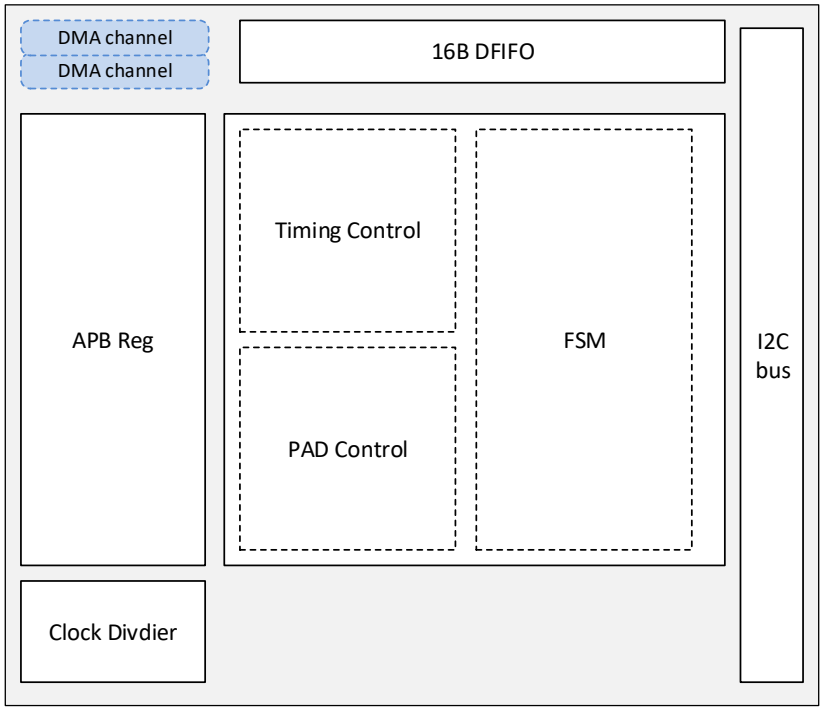


Figure 8-20. Block diagram of I2C

8.3.4 AC Timing Characteristics

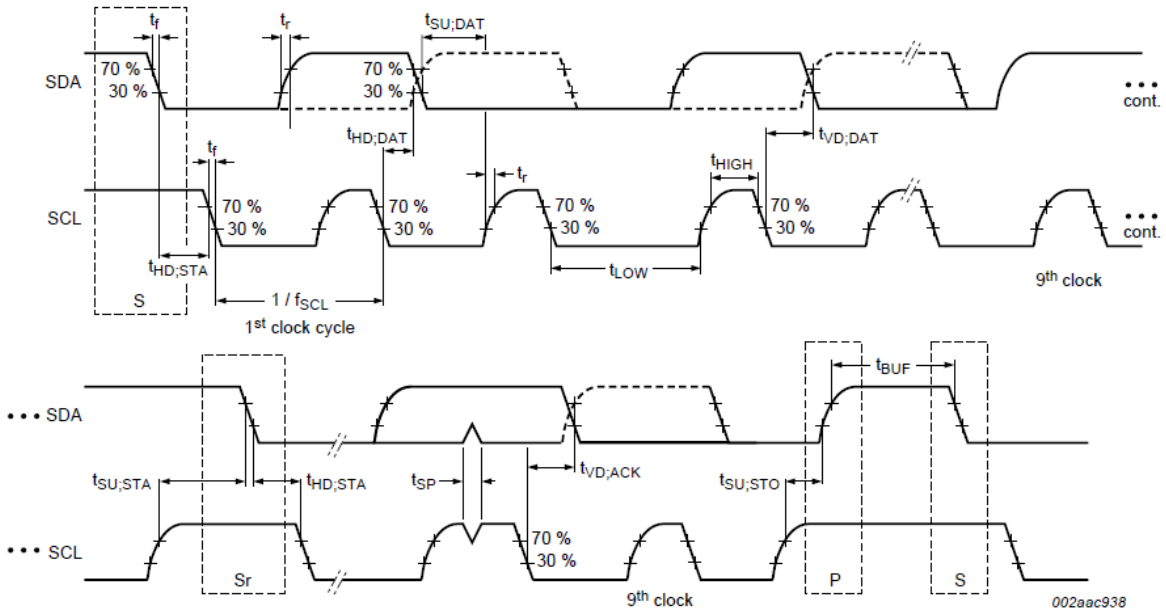


Figure 8-21. I2C AC timing diagram of Standard mode

Table 8-5. I2C AC timing parameters for Standard mode

Symbol	Parameter	Standard-mode		Unit	Note
		Min	Max		
fSCL	SCL clock frequency	0	100	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	-

8.3.5 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

8.4 Universal Asynchronous Receiver/Transmitter (UART)

8.4.1 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 that 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.

8.4.2 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.

8.4.3 Block Diagram

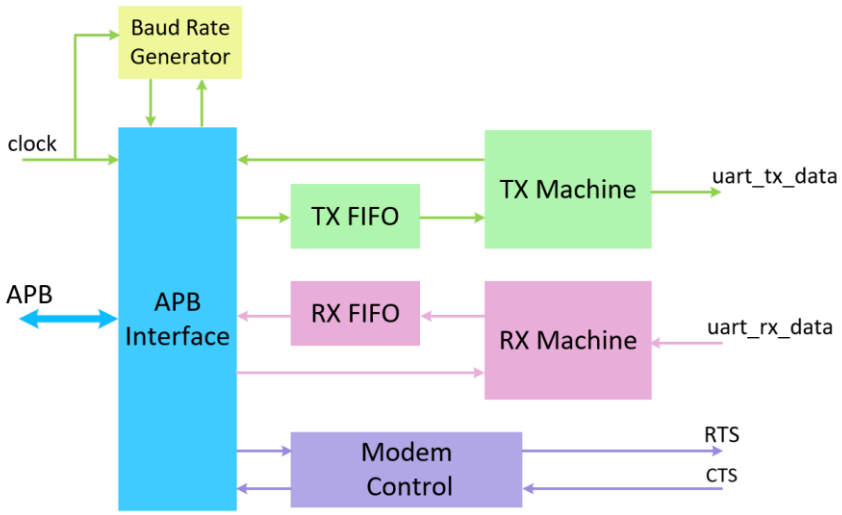


Figure 8-22. Block diagram of UART

Figure 8-22 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 8-6.

Table 8-6. 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

8.4.4 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.

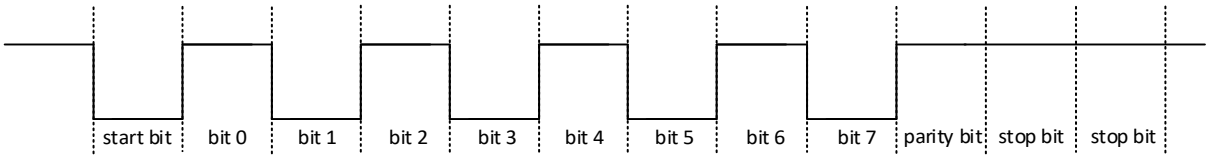


Figure 8-23. UART communication protocol

8.4.5 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 error could happen and the received data could get corrupted.**

UART Interrupt

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

Table 8-7. 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 8-8. 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 8-9. 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.

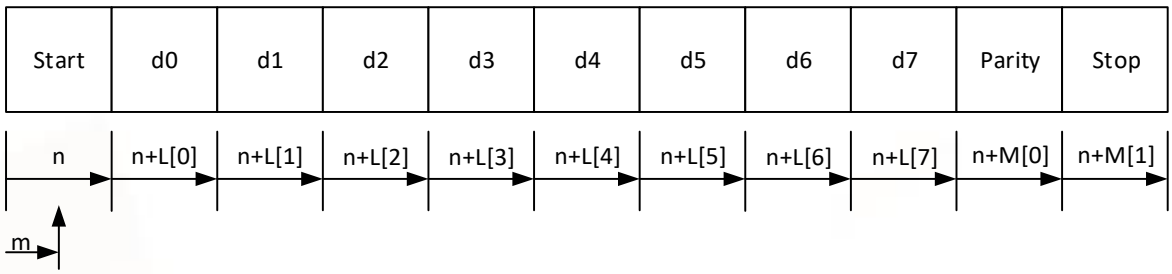
UARTn+0058h Fractional Divider MSB Address UARTn_FRACDIV_M

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 8-10 for details.

Table 8-10. 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

8.4.6 Programming Guide

8.4.6.1 Baud Rate Setting and UART Initialization

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

Table 8-11. 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 8-12 is an example of generating baud rate 115200 bps by clock inputs of 30 MHz:

Table 8-12. 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 8-13. UART hardware initialization

Step	Description	Related Register Setting
1	Baud rate setting: please refer to Table 8-12	-
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

8.4.6.2 Transmission

Data Transmission

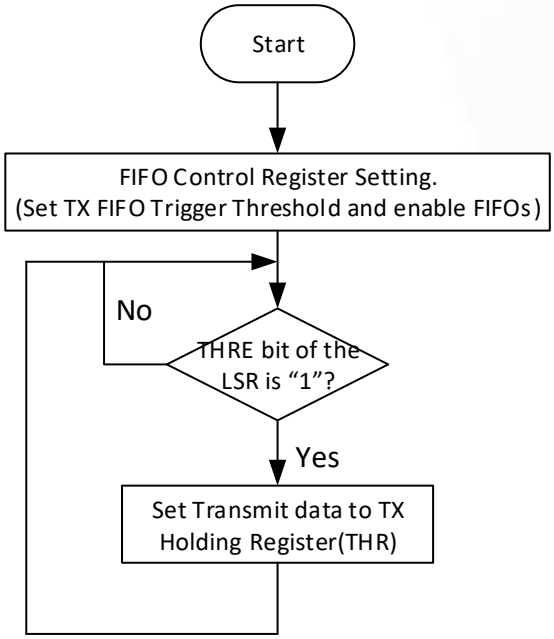


Figure 8-24. UART data transmission with THRE bit status polling

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

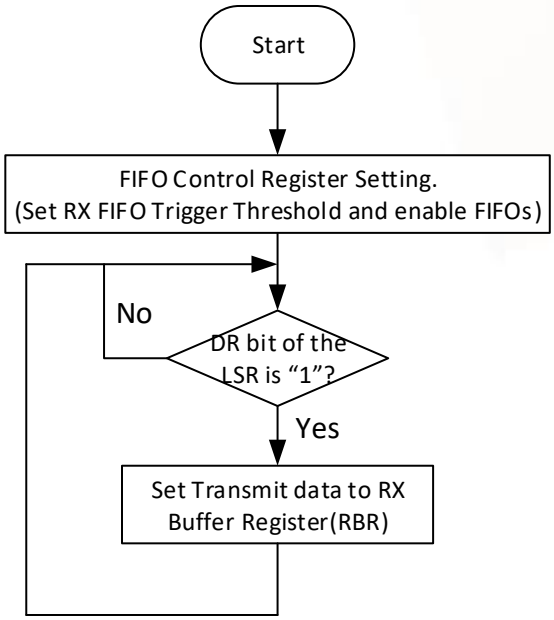


Figure 8-25. UART data reception with DR bit status polling

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)

8.4.7 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

8.5 Pulse Width Modulators (PWMs)

8.5.1 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 8-26).

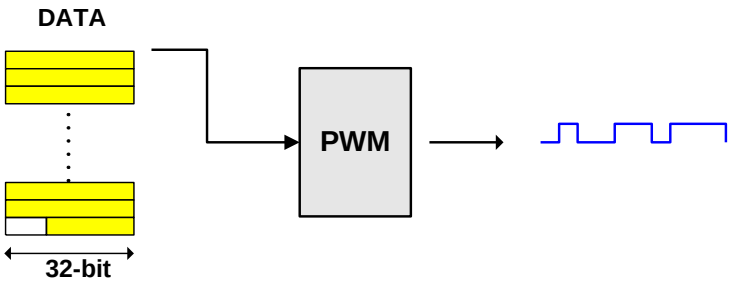


Figure 8-26. Generation procedure of PWM

8.5.2 Features

The features of PWM are as follows:

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

8.5.3 Block Diagram

Figure 8-27 shows the block diagram of the PWM.

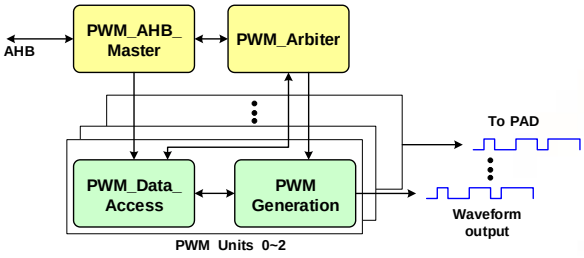


Figure 8-27. PWM block diagram

8.5.4 Theory of Operations

8.5.4.1 Old Mode

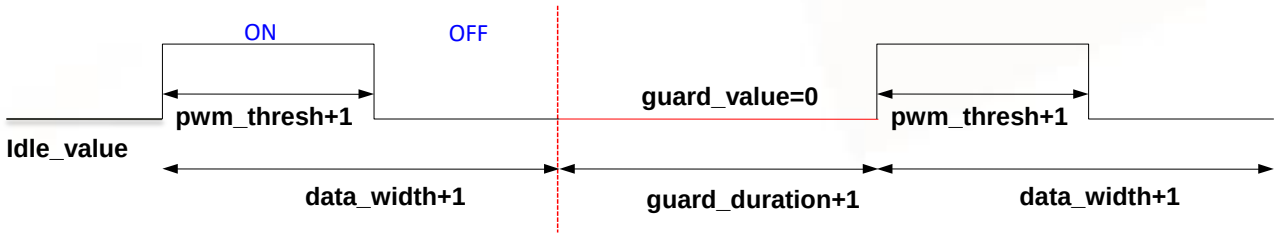


Figure 8-28. Old mode

In the old mode, the waveform generated by PWM is shown in Figure 8-28. 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.

8.5.4.2 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.

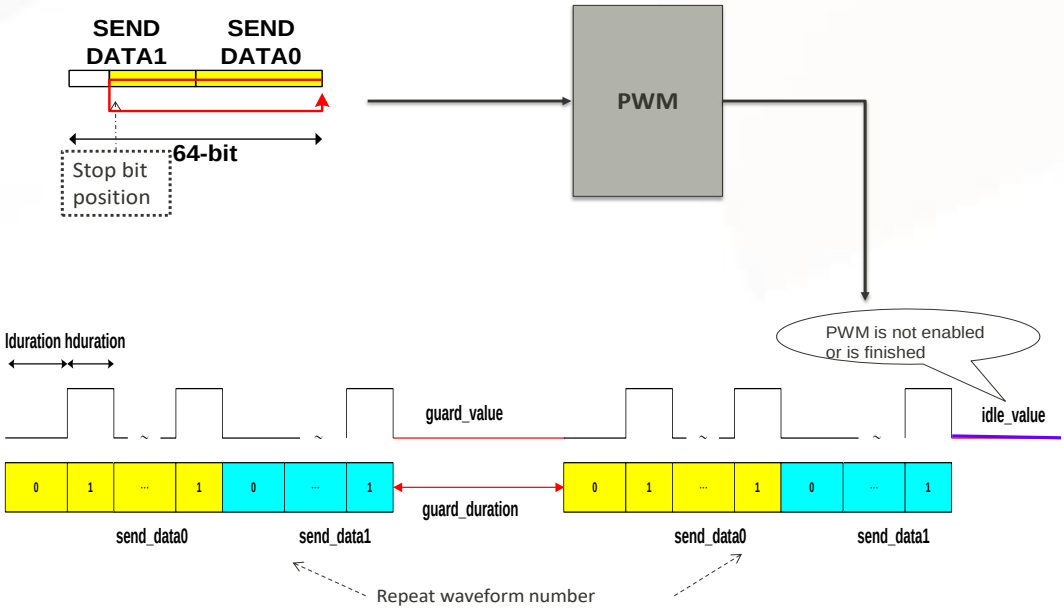


Figure 8-29. FIFO mode

8.5.4.3 Memory Mode

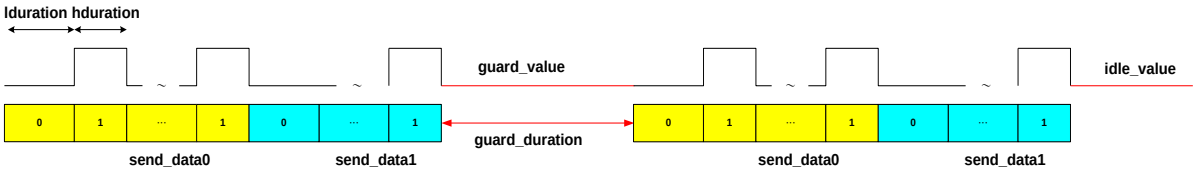


Figure 8-30. Memory mode

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.

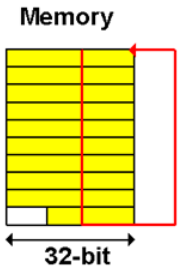


Figure 8-31. Memory mode and stop bit position

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.**

8.5.4.4 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 8-32. 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 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 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.

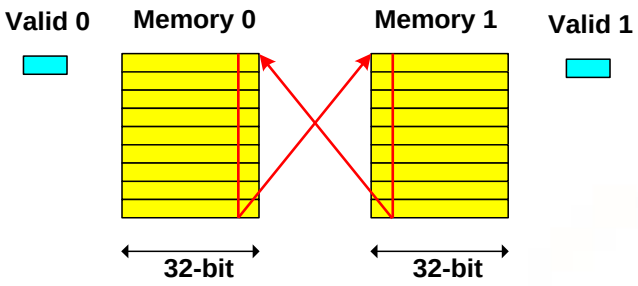


Figure 8-32. Random mode

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.**

8.5.5 Programming Guide

8.5.5.1 Old Mode

Table 8-14. 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]

8.5.5.2 FIFO Mode

Table 8-15. 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.
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]

8.5.5.3 Memory Mode

Table 8-16. 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]

8.5.5.4 Random Mode

Table 8-17. 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]

8.5.6 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

8.6 GPIO (General-Purpose Input/Output)

8.6.1 Introduction

The I/O pins can be programmed as multiple-purpose pins such as GPIO, SPI-NOR, etc. pins. By setting the GPIO_MODE register, I/O can be selected for specific functions.

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.

8.6.2 Features

Below is the block diagram of GPIO. Each GPIO controls the auxiliary mode by programming the GPIO_MODE_SELx command register. Besides, the dedicated register bits can be set to 1 or 0 by writing the bits of GPIO_MODEx_SET or GPIO_MODEx_CLR to 1.

GPIO_DIR, GPIO_DOUT, GPIO_DIN are also programmable by the same method of GPIO_MODE.

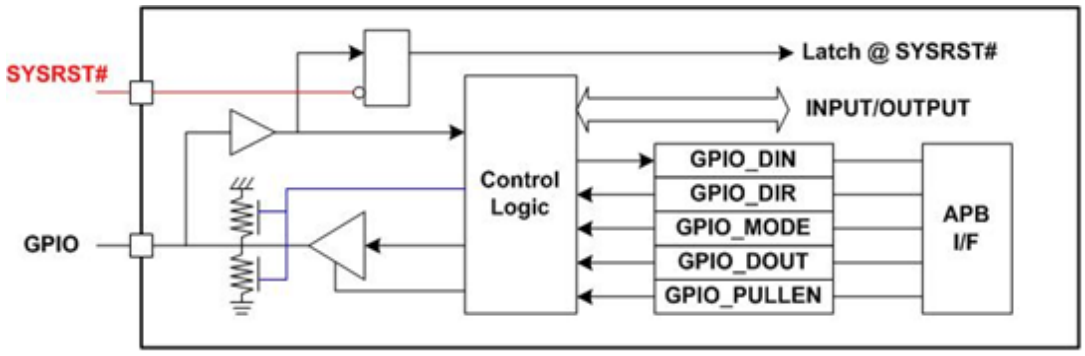


Figure 8-33. Block diagram of GPIO

8.6.3 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

9 Connectivity

9.1 NETSYS

9.1.1 Frame Engine (FE)

9.1.1.1 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/from Ethernet hardware offload, which is composed of WED/WDMA.

9.1.1.2 Features

The main features of these functions include.

- Frame engine with PPE, ADMA and QDMA that connects to DRAM via AXI bus (64-bit)
- IPv4 routing, NAT, NAPT
- IPv6 routing, DS-lite, 6RD, 6to4, 464
- QOS
- IP/TCP/UDP checksum offload
- TSO (TCP segmentation offload)
- LRO (Larger receive offload)

9.1.1.2.1 PSE Features

- Wire-speed (2.5Gbps) Ethernet LAN/WAN NAT/NAPT routing
- Egress rate limiting/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 & PPPOE header insertion
- TCP segmentation offload
- Auto-padding for sub-64 B packets

9.1.1.2.2 PPE Features

- IPv4 NAT/NAPT and IPv6 Routing
- IPv4/IPv6 transition mechanism:
  - Tunneling (DS-Lite, 6RD, MAPE/MAPT)
  - Transition (464XLAT)
- Support 1/2/4/8/16/32 K session/flows

- Virtual server, port-triggering and port forwarding
- All types of IPv4 NAT (NAPT, Twice NAT)
- All types of MAC/VLAN/PPPOE/IP/TCP/UDP binding
- 4 VLAN tagging (Q-in-Q)
- VID Swapping
- Support for 65536 PPPOE sessions
- PPPOE pass-through
- Cone-NAT, port-restricted NAT & symmetric NAT
- Per flow accounting or rate limiting
- Flow offloading technology for flexible/high performance packet L3/L4 packet processing
- Multi-WAN load balancing with hardware and software cooperation
- QoS for multimedia traffic
- Support NAT/NAPT wire-speed within 128 flows for any packet size

9.1.1.2.3 ADMA Features

- Supports 4 Tx descriptor rings and 4 Rx descriptor rings
- Scatter/Gather DMA
- Delayed interrupt
- Configurable 4/8/16/32 32-bit word burst length
- Support LRO (Large Receive Offload) with 4 normal RX ring and 4 LRO RX ring
- Support RSS (Receive Side Scaling)
- Support TSO (TCP Segmentation Offload)

9.1.1.2.4 QDMA Features

- Supports 1 Tx descriptor link list from software
- Supports direct Tx packet hardware forwarding from PPE
- Supports 1 Rx descriptor rings
- Configurable 4/8/16/32 32-bit word burst length
- Delayed interrupt
- Supports 128 Tx queues
- Per Tx queue forward/drop packet accounting
- Per Tx queue forward byte accounting
- Per Tx queue minimum/maximum rate control
- Strict priority arbitration between 16 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
- Supports Random Early Drop with configurable dropping probability
- 8 Tx queues support SFQ and DRR scheduling, with virtual queue capability
- Supports up to 1024 virtual queues, those are shared by 8 Tx queues
- Supports SFQ/DRR hash perturb

- Per virtual queue forward packet/byte and drop packet accounting

9.1.1.3 Block Diagram

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

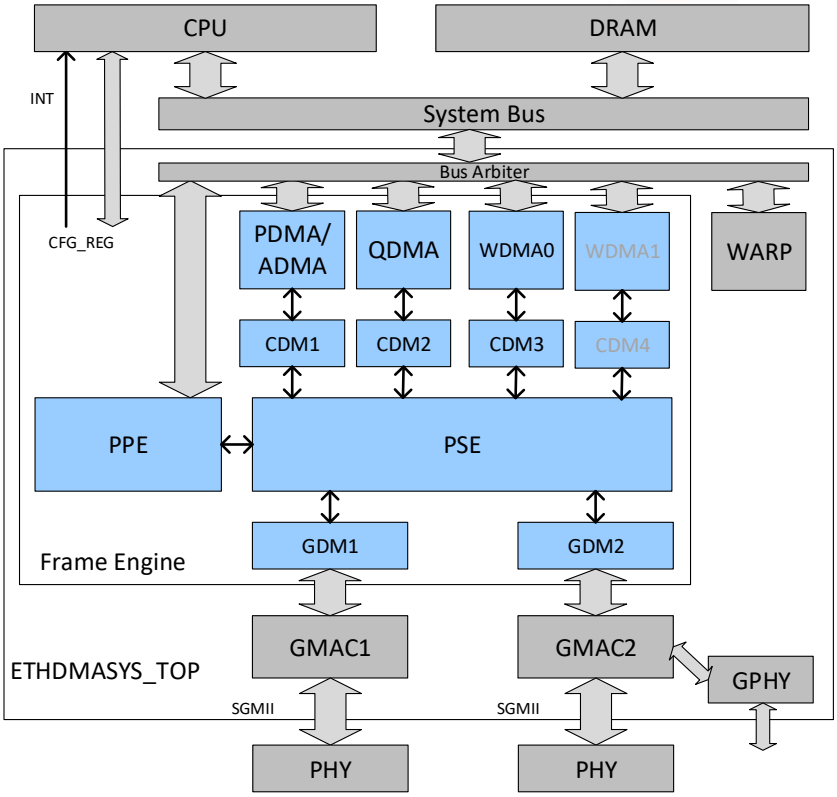


Figure 9-1. Block diagram of frame engine in NETSYS

9.1.2 GMAC (Gigabit-Media Access Controller)

9.1.2.1 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 other MAC clients).

9.1.2.2 Features

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

9.1.2.3 Block Diagram

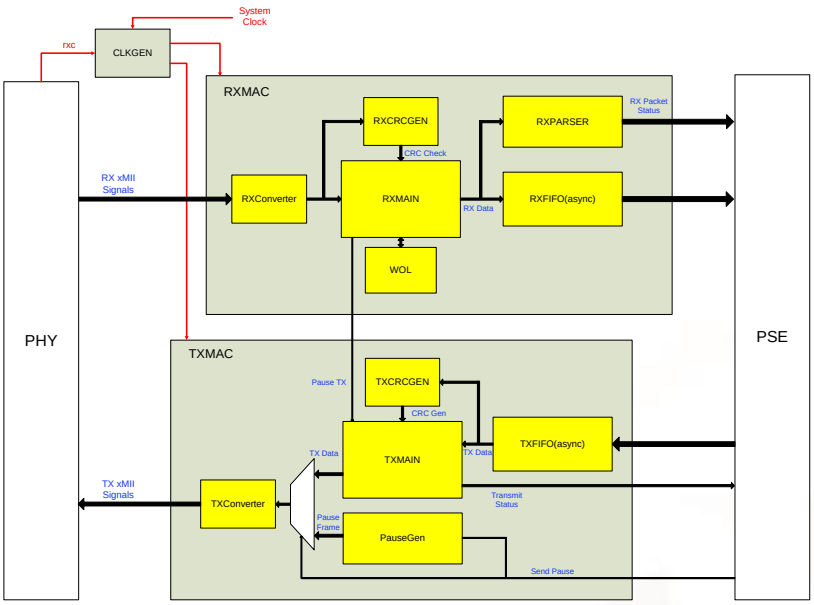


Figure 9-2. Block diagram of GMAC (per port)

9.2 GPHY (Gigabit Ethernet PHY)

9.2.1 Introduction

Gigabit Ethernet PHY supports the following interface:

- 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).

9.2.2 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

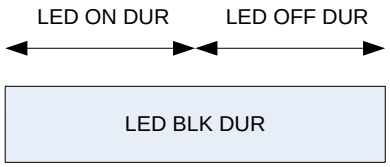
9.2.3 Programming Guide

9.2.3.1 LED Setting

51F00210	dev1Fh_reg021h	16	LED Basic Control Register
51F00220	dev1Fh_reg022h	16	LED On Duration Register
51F00230	dev1Fh_reg023h	16	LED Blinking Duration Register
51F00240	dev1Fh_reg024h	16	LED0_ON_CTL, LED0 On Control Register
51F00250	dev1Fh_reg025h	16	LED0_BLK_CTL, LED0 Blinking Control Register
51F00260	dev1Fh_reg026h	16	LED0_BLK_CTL, LED1 On Control Register
51F00270	dev1Fh_reg027h	16	LED1_BLK_CTL, LED1 Blinking Control Register
51F00280	dev1Fh_reg028h	16	LED0_BLK_CTL, LED2 On Control Register
51F00290	dev1Fh_reg029h	16	LED2_BLK_CTL, LED2 Blinking Control Register
51F002A0	dev1Fh_reg02Ah	16	LED0_BLK_CTL, LED3 On Control Register
51F002B0	dev1Fh_reg02Bh	16	LED3_BLK_CTL, LED3 Blinking Control Register

How to adjust the LED light:

- Set LED On Duration Register and LED Blinking Duration Register



How to use custom LED mode:

- led_enhance_mode = 1
- Set LED0_ON_CTL~LED3_ON_CTL and LED0_BLK_CTL~LED3_BLK_CTL.
  - 0xC000, LED0_BLK_CTL = 0x3C00, LED1_ON_CTL = 0xC006, LED0_BLK_CTL=0x0000

51F00210 GPHY_DEV1FH_REG021H LED Basic Control Register 000A

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	led_enhance_mode											led_event_all	Led_Clock_Enab	Led_Timing_Test	led_mode	
Type	RW											RW	RW	RW	RW	
Reset	0											0	1	0	1	0

Bit(s)	Name	Description
15	led_enhance_mode	**Extended Control** 0: LED functions are controlled by the 2-bit LED_MODE field of this register 1: LED functions of each pin are independently controlled by LED0_ON_CTL~LED3_ON_CTL and LED0_BLK_CTL~LED3_BLK_CTL
4	led_event_all	**Use all port LED events**
3	Led_Clock_Enab	**LED Clock Enable** LED Timing Test 0: Disable LED clock 1: Enable LED clock
2	Led_Timing_Test	**LED Timing Test** 0: Normal LED clock frequency 1: Increase LED clock frequency

Bit(s)	Name	Description
1:0	led_mode	**LED Mode Configuration.** The default register value refers to hardware pin-strapping and the detected LED Pin Polarity. This is decided by power-on strapping status. (If circuits cannot decide its polarity, the polarity should be assumed as active low) 00: All LED outputs are disabled. 01: 2 LED pins are enabled. LED0: Link. LED1: Activity. Each LED pin polarity is automatically configured. Note: the above settings can also be achieved by setting the following register fields when EXT_CTL = 1 LED0_ON_CTL' Status = 7'b0000111 LED0_BLK_CTL' Event = 10'b0000000000 LED1_ON_CTL' Status = 7'b0000000 LED1_BLK_CTL' Event = 10'b0000111111 10: 3 LED pins are enabled, active low. LED0: Link1000 Activity. LED1: Link100 Activity. LED2: Link10 Activity Each LED pin polarity is automatically configured. Note: the above settings can also be achieved by setting the following register fields when EXT_CTL = 1LED0_ON_CTL' Status = 7'b0000001LED0_BLK_CTL' Event = 10'b0000000011LED1_ON_CTL' Status = 7'b0000010LED1_BLK_CTL' Event = 10'b0000001100LED2_ON_CTL' Status = 7'b0000100LED2_BLK_CTL' Event = 10'b0000110000 11: 3 LED pins are enabled, active low. LED0: Link1000 Activity. LED1: Link10/100 Activity. LED2: Duplex/Collision Each LED pin polarity is automatically configured. Note: the above settings can also be achieved by setting the following register fields when EXT_CTL = 1LED0_ON_CTL' Status = 7'b0000001LED0_BLK_CTL' Event = 10'b0000000011LED1_ON_CTL' Status = 7'b0000110LED1_BLK_CTL' Event = 10'b0000111100LED2_ON_CTL' Status = 7'b0010000LED2_BLK_CTL' Event = 10'b0001000000

51F00220 GPHY_DEV1FH_REG022H LED On Duration Register 0C00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	led_on_dur															
Type	RW															
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
15:0	led_on_dur	**LED on Duration** LED ON duration in terms of number of 32.768 us. 32.768 us ~ 2.147 sec period. Default value = 0x0C00 * 32.768 us = 100.66 ms

51F00230 GPHY_DEV1FH_REG023H LED Blinking Duration Register 1400

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	led_blk_dur															
Type	RW															
Reset	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
15:0	led_blk_dur	**LED Blinking Duration** Blinking Periodic duration in terms of number of 32.768 us. 32.768 us ~ 2.147 sec period Default value = 0x1400 * 32.768 us = 167.77 ms

51F00240 **GPHY_DEV1FH_REG024H** **LED0 On Control Register** **8000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rg_led0_en	rg_led0_pol								led0_on_mask						
Type	RW	RW								RW						
Reset	1	0								0	0	0	0	0	0	0

Bit(s)	Name	Description
15	rg_led0_en	**Enable Ethernet LED Function.** 0: Disable (Hi-Z) 1: Enable
14	rg_led0_pol	**Select LED polarity. This field only takes effect when LED_EN is 1b1.** Enable Ethernet LED Function. 0: Active low (That is, LED On means Output 0) 1: Active high (That is, LED On means Output 1)
6:0	led0_on_mask	**LED is on if any of the following state holds. This field only takes effect when LED_EN is 1b1.** Select LED polarity. This field only takes effect when LED_EN is 1b1. [0]: Link 1000 [1]: Link 100 [2]: Link 10 [3]: Link Down [4]: Full Duplex [5]: Half Duplex [6]: Force On (Logic 1)

51F00250 **GPHY_DEV1FH_REG025H** **LED0 Blinking Control Register** **0000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							led0_blk_mask									
Type							RW									
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
9:0	led0_blk_mask	**LED blinks if any of the following event occurs. This field only takes effect when LED_EN is 1b1.** (Note: The LED-Blinking Priority Precedes The LED-On Priority, That is, when there is an event that triggers LED-Blinking, it will take control of LED output no matter what LED-On status is) [0]: 1000Mbps TX Activity [1]: 1000Mbps RX Activity [2]: 100Mbps TX Activity [3]: 100Mbps RX Activity [4]: 10Mbps TX Activity [5]: 10Mbps RX Activity [6]: Collision [7]: RX CRC Error [8]: RX Idle Error [9]: Force Blinks (Logic 1)

51F00260 **GPHY_DEV1FH_REG026H** **LED1 On Control Register** **8000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rg_led1_en	rg_led1_pol								led1_on_mask						
Type	RW	RW								RW						
Reset	1	0								0	0	0	0	0	0	0

Bit(s)	Name	Description
15	rg_led1_en	**Enable Ethernet LED Function.** 0: Disable (Hi-Z) 1: Enable
14	rg_led1_pol	**Select LED polarity. This field only takes effect when LED_EN is 1b1.** Enable Ethernet LED Function. 0: Active low (That is, LED On means Output 0) 1: Active high (That is, LED On means Output 1)
6:0	led1_on_mask	**LED is on if any of the following state holds. This field only takes effect when LED_EN is 1b1.** Select LED polarity. This field only takes effect when LED_EN is 1b1. [0]: Link 1000 [1]: Link 100 [2]: Link 10 [3]: Link Down [4]: Full Duplex [5]: Half Duplex [6]: Force On (Logic 1)

51F00270 **GPHY_DEV1FH_REG027H** **LED1 Blinking Control Register** **0000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							led1_blk_mask									
Type							RW									
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
9:0	led1_blk_mask	**LED blinks if any of the following event occurs. This field only takes effect when LED_EN is 1b1.** (Note: The LED-Blinking Priority Precedes The LED-On Priority, That is, when there is an event that triggers LED-Blinking, it will take control of LED output no matter LED-On status is) [0]: 1000Mbps TX Activity [1]: 1000Mbps RX Activity [2]: 100Mbps TX Activity [3]: 100Mbps RX Activity [4]: 10Mbps TX Activity [5]: 10Mbps RX Activity [6]: Collision [7]: RX CRC Error [8]: RX Idle Error [9]: Force Blinks (Logic 1)

51F00280 **GPHY_DEV1FH_REG028H** **LED2 On Control Register** **8000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rg_led2_en	rg_led2_pol								led2_on_mask						
Type	RW	RW								RW						
Reset	1	0								0	0	0	0	0	0	0

Bit(s)	Name	Description
15	rg_led2_en	**Enable Ethernet LED Function.** 0: Disable (Hi-Z) 1: Enable
14	rg_led2_pol	**Select LED polarity. This field only takes effect when LED_EN is 1b1.** Enable Ethernet LED Function. 0: Active low (That is, LED On means Output 0) 1: Active high (That is, LED On means Output 1)
6:0	led2_on_mask	**LED is on if any of the following state holds. This field only takes effect when LED_EN is 1b1.** Select LED polarity. This field only takes effect when LED_EN is 1b1. [0]: Link 1000 [1]: Link 100 [2]: Link 10 [3]: Link Down [4]: Full Duplex [5]: Half Duplex [6]: Force On (Logic 1)

51F00290 **GPHY_DEV1FH_REG029H** **LED2 Blinking Control Register** **0000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							led2_blk_mask									
Type							RW									
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
9:0	led2_blk_mask	**LED blinks if any of the following event occurs. This field only takes effect when LED_EN is 1b1.** (Note: The LED-Blinking Priority Precedes The LED-On Priority, That is, when there is an event that triggers LED-Blinking, it will take control of LED output no matter what LED-On status is) [0]: 1000Mbps TX Activity [1]: 1000Mbps RX Activity [2]: 100Mbps TX Activity [3]: 100Mbps RX Activity [4]: 10Mbps TX Activity [5]: 10Mbps RX Activity [6]: Collision [7]: RX CRC Error [8]: RX Idle Error [9]: Force Blinks (Logic 1)

51F002A0 **GPHY_DEV1FH_REG02AH** **LED3 On Control Register** **8000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rg_led3_en	rg_led3_pol								led3_on_mask						
Type	RW	RW								RW						
Reset	1	0								0	0	0	0	0	0	0

Bit(s)	Name	Description
15	rg_led3_en	**Enable Ethernet LED Function.** 0: Disable (Hi-Z) 1: Enable
14	rg_led3_pol	**Select LED polarity. This field only takes effect when LED_EN is 1b1.** Enable Ethernet LED Function. 0: Active low (That is, LED On means Output 0) 1: Active high (That is, LED On means Output 1)
6:0	led3_on_mask	**LED is on if any of the following state holds. This field only takes effect when LED_EN is 1b1.** Select LED polarity. This field only takes effect when LED_EN is 1b1. [0]: Link 1000 [1]: Link 100 [2]: Link 10 [3]: Link Down [4]: Full Duplex [5]: Half Duplex [6]: Force On (Logic 1)

51F002B0 **GPHY_DEV1FH_REG02BH** **LED3 Blinking Control Register** **0000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							led3_blk_mask									
Type							RW									
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
9:0	led3_blk_mask	**LED blinks if any of the following event occurs. This field only takes effect when LED_EN is 1b1.** (Note: The LED-Blinking Priority Precedes The LED-On Priority, That is, when there is an event that triggers LED-Blinking, it will take control of LED output no matter what LED-On status is) [0]: 1000Mbps TX Activity [1]: 1000Mbps RX Activity [2]: 100Mbps TX Activity [3]: 100Mbps RX Activity [4]: 10Mbps TX Activity [5]: 10Mbps RX Activity [6]: Collision [7]: RX CRC Error [8]: RX Idle Error [9]: Force Blinks (Logic 1)

9.2.4 Register Definition

Refer to “MT7981B Wi-Fi 6 Platform Registers” for detailed register descriptions.

10 High Speed Interface

10.1 SGMII (Serial Gigabit Media Independent Interface)

10.1.1 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 2.5Gbps (proprietary 2500Base-X) data rate of MAC by overclocking.

10.1.2 Block Diagram

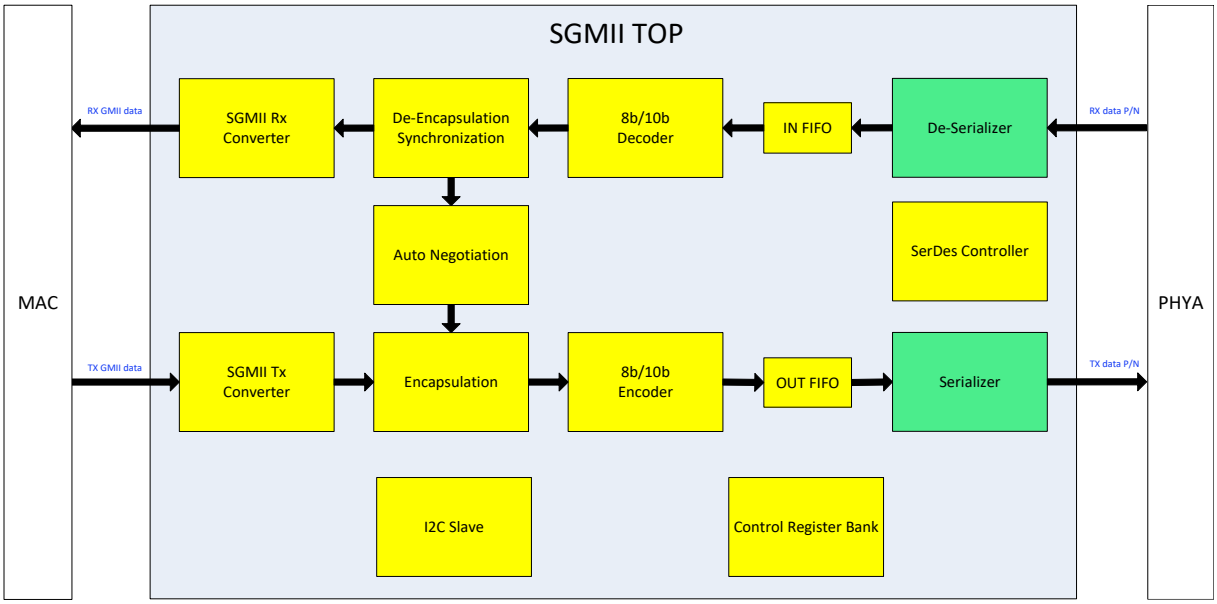


Figure 10-1. Block diagram of SGMII

10.1.3 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)

10.2 HIF (Host Interface)

10.2.1 Introduction

The HIF has one PCIe and one SSUSB port. Both are in host mode.

10.2.2 PCIe Controller

10.2.2.1 Overview

PCI Express (Peripheral Component Interconnect Express) controller is compliant with the Intel® PIPE (PHY Interface for the PCI Express) interface, allowing integration with PIPE-compliant PHY and also compliant with the AMBA® AXI4 specifications. It supports PCI Express Gen1 (2.5Gbps), PCI Express Gen2 (5.0Gbps).

10.2.2.2 Features

PCIe Interface:

- Supports Root Complex (RC) mode
- Supports x1 link
- Supports link rate of 2.5 GT/s, 5.0 GT/s per lane
- PCI Express Base Specification Revision 3.0 compliant
- PHY Interface for PCI Express (PIPE) 4.0 compliant
- Supports memory, I/O, Configuration, and Message transactions specified in the following section
- Support maximum payload size of 256 bytes
- Support maximum read request size of 256 bytes
- Support 1 VC (Virtual Channel).
- Support 1 Physical Function.
- AER (Advanced Error Reporting) support
- ECRC generation and check support
- Lane Reversal support
- Polarity Inverse support
- Legacy PCI Power Management support
- ASPM (Native Active State Power Management) L0s and L1 state support
- L1PMSS (L1 Power Management Substates) with CLKREQ# support
- LTR (Latency Tolerance Reporting) support
- MSI (Message Signaled Interrupt, per function up to 32) and INT x (Legacy Interrupts) support

APB/AXI Interface:

- 1 AHB slave interface for Bridge configuration
- Up to 1 AXI master interface, each supporting up to 16/16 outstanding write/read requests
- Up to 1 AXI slave interface, each supporting up to 4/4 outstanding write/read requests
- 128-bit data support for AXI master, 64-bit data support for slave interface
- Bypassable CDC for AXI master, slave interface

10.2.2.3 Block Diagram

The PCIe (PCI Express) consists of 4 different layers, PCI Express layer, Bridge layer, AXI layer and Physical layer. See Figure 10-2.

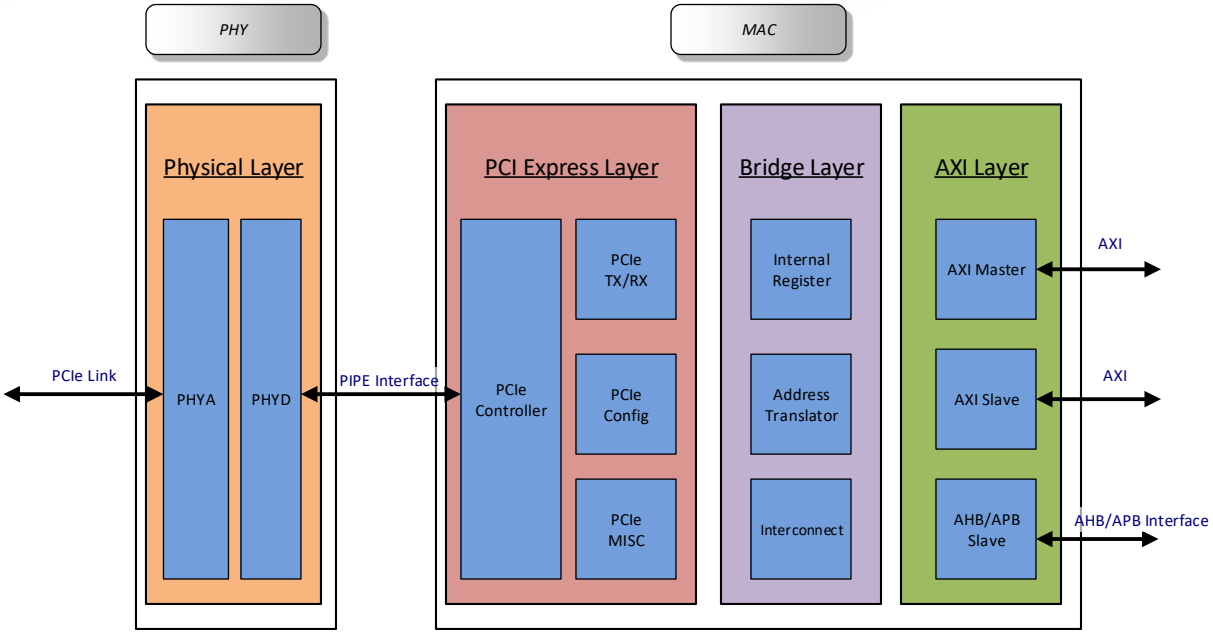


Figure 10-2. Block diagram of PCIe controller

The PCI Express layer consists of:

- A PCIe Controller, which is a PCI Express Core IP
- A PCIe Tx/Rx Interface between the Bridge and the PCIe Controller
- A PCIe Configuration Interface to give the Bridge Layer access to the PCIe Configuration Space
- A PCIe Misc Interface to allow the Bridge to manage Low-Power, Interrupts, etc.

The Bridge layer consists of:

- The Internal Registers of the PCIe controller
- Address Translator Modules to convert between the AXI and PCIe interfaces
- When transferring PCIe receive requests to the AXI Master, the Address Translator adds the corresponding AXI base address and forwards the request to the desired AXI Master interface
- The Address Translation method is similar when transferring AXI receive requests to the PCIe interface
- An Interconnect Module to interconnect and arbitrate between input and output flow

The AXI layer consists of:

- An AXI Master Interface, which manages requests and completions from bridge layer to AXI
- An AXI Slave Interface, which manages requests and completions from AXI to bridge layer
- An AHB Slave Interface for bridge configuration

10.2.3 SSUSB

10.2.3.1 Overview

MT7981 provides a USB3.1 Gen1 port that supports Gen1 host mode and USB2.0 host.

10.2.3.2 Features

- Support USB3.1 Gen1 host and USB2.0 host
- Host support EP number: 64
- Host support slot number: 15
- Host USB2.0 port number: 1
- Host USB3.1 port number: 1

10.2.3.3 Block Diagram

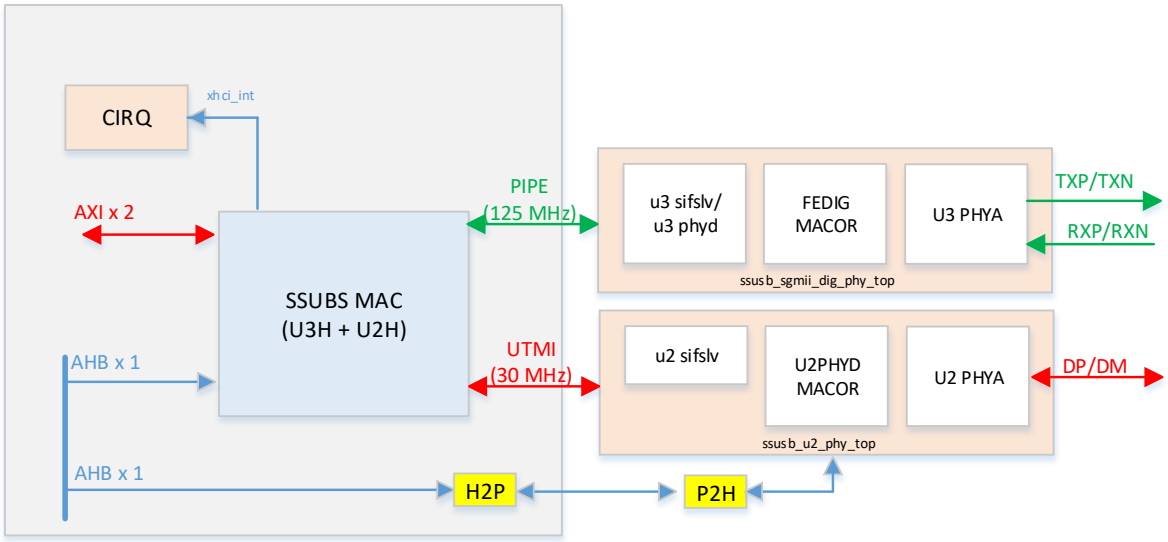


Figure 10-3. USB block diagram

10.3 PHYD Register Control

1. Default mode is USB PHY
2. PHY-Mode setting 0x11D10218[1:0]:
   - 00: PCIe PHY
   - 01: USB PHY
   - 10: SGMII PHY

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.