8 Interfaces

8.1 CIR Receiver (CIR_RX)

8.1.1 Overview

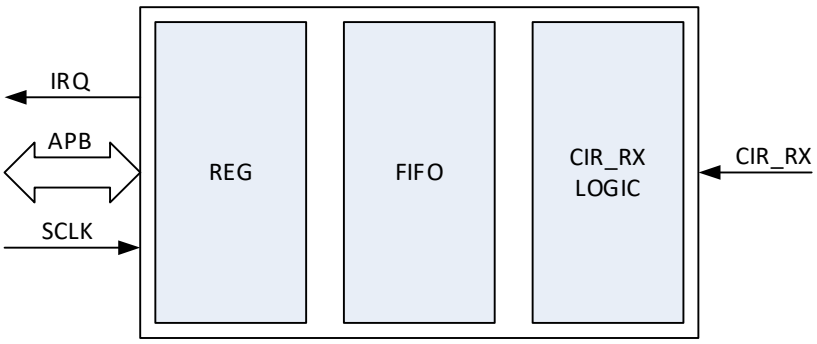
The Consumer Infrared (CIR) receiver captures pulse from the IR Receiver module and uses the Run-Length Code (RLC) to encode the pulse.

The CIR receiver has the following features:

- One CIR_RX interface in CPUX domain and one CIR_RX interface in CPUS domain
- Full physical layer implementation
- Supports NEC format infra data
- Supports CIR for remote control
- 64x8 bits FIFO for data buffer
- Sample clock up to 1 MHz
- Supports interrupt and DMA mode

8.1.2 Block Diagram

Figure 8-1 CIR Receiver Block Diagram



The CIR receiver samples the input signal on the programmable frequency and records these samples into RX FIFO when one CIR signal is found on the air. The CIR receiver uses Run-Length Code (RLC) to encode pulse width. The encoded data is buffered in 64 levels and 8-bit width RX FIFO; the MSB bit is used to record the polarity of the receiving CIR signal, the rest 7 bits are used for the length of RLC. The maximum length of the RLC is 128. If the duration of one level (high or low level) is more than 128, another byte is used.

8.1.3 Functional Description

8.1.3.1 External Signals

The following table describes the external signals of CIR Receiver.

Table 8-1 CIR Receiver External Signals

Signal Name	Description	Type
IR-RX	Consumer Infrared Receiver	I
S-IR-RX	Consumer infrared receiver	I

8.1.3.2 Clock Sources

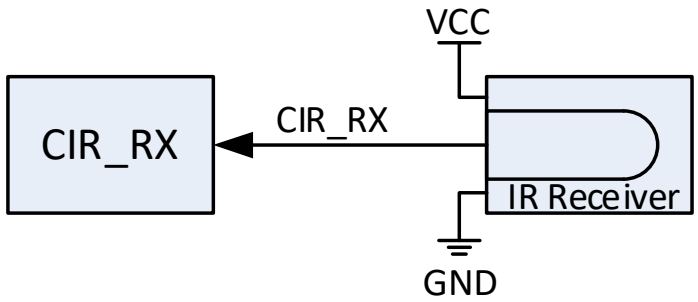
The following table describes the clock sources of CIR Receiver.

Table 8-2 CIR Receiver Clock Sources

Clock Sources	Description	Module
CIR_RX		
CLK32K	By default, CLK32K is 32.768 kHz.	CCU
HOSC	By default, HOSC is 24 MHz.	CCU
S_CIRRX		
CLK32K	By default, CLK32K is 32.768 kHz.	PRCM
CLK24M	By default, CLK24M is 24 MHz.	PRCM

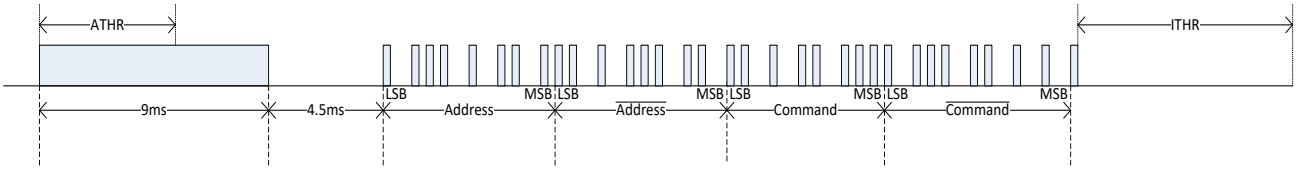
8.1.3.3 Typical Application

Figure 8-2 CIR Receiver Application Diagram



8.1.3.4 NEC Protocol Format

Figure 8-3 NEC Protocol



The CIR receiver module is a timer with a capture function.

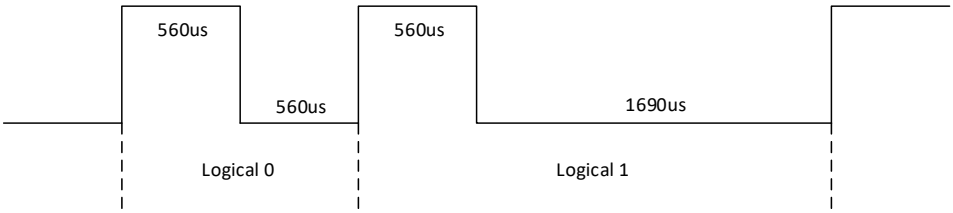
When CIR_RX signals satisfy the Active Threshold (ATHR), the CIR receiver can start to capture. In the process, the signal is ignored if the pulse width of the signal is less than NTHR. When CIR_RX signals satisfy ITHR (Idle Threshold), the capture process is stopped and the Receiver Packet End interrupt is generated, then the Receiver Packet End Flag is asserted.

In a capture process, every effective pulse is buffered to FIFO in bytes according to the form of the Run-Length Code. The MSB bit of a byte is the polarity of pulse, and the rest 7 bits is pulse width by taking Sample Clock as a basic unit. This is the code form of the RLC-Byte. When the level changes or the pulse width counter overflows, the RLC-Byte is buffered to FIFO. The CIR_RX module receives the infrared signals transmitted by the infrared remote control, the software decodes the signals.

8.1.3.5 Operating Mode

Sample Clock

Figure 8-4 Logical '0' and Logical '1' of NEC Protocol



For NEC protocol, a logical "1" takes 2.25 ms (560 us+1680 us) to transmit, while a logical "0" is only half of that, being 1.12 ms (560 us+560 us).

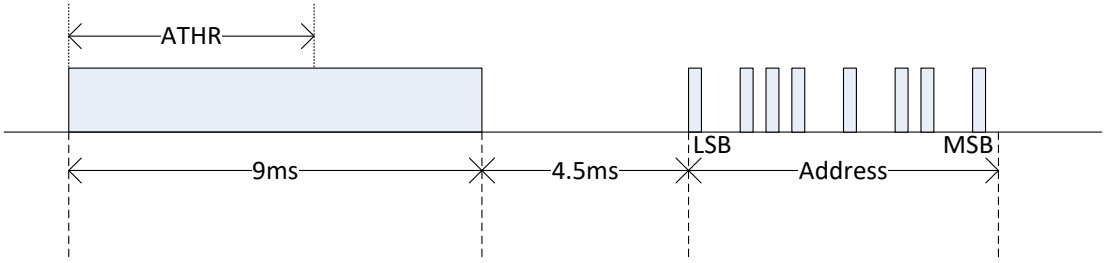
For example, if the sample clock is 31.25 kHz, a sample cycle is 32 us, then 18 sample cycles are 560 us. So the RLC of 560 us low level is 0x12 (b'00010010), the RLC of 560 us high level is 0x92 (b'10010010). Then a logical "1" takes code 0x12 (b'00010010) and code 0xb5 (b'10110101) to transmit, a logical "0" takes code 0x12 and code 0x92 to transmit.

Active Threshold (ATHR)

When the CIR receiver is in Idle state, if the electrical level of the IR-RX signal changes (positive jump or negative jump), and the duration reaches this threshold, then the CIR receiver takes the

starting of the signal as a lead code, and the CIR receiver turns into an active state and starts to capture IR-RX signals.

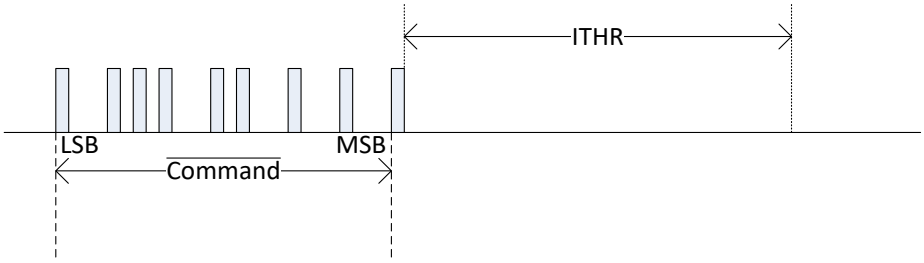
Figure 8-5 ATHR Definition



Idle Threshold (ITHR)

If the electrical level of IR-RX signals has no change, and the duration reaches this threshold, then the CIR receiver enters into Idle state and ends this capture.

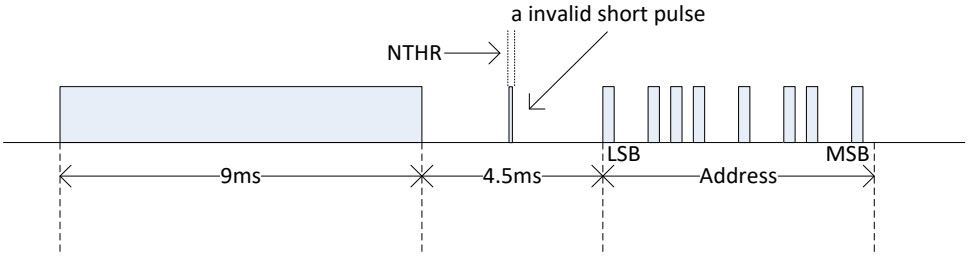
Figure 8-6 ITHR Definition



Noise Threshold (NTHR)

In the capture process, the pulse is ignored if the pulse width is less than the Noise Threshold.

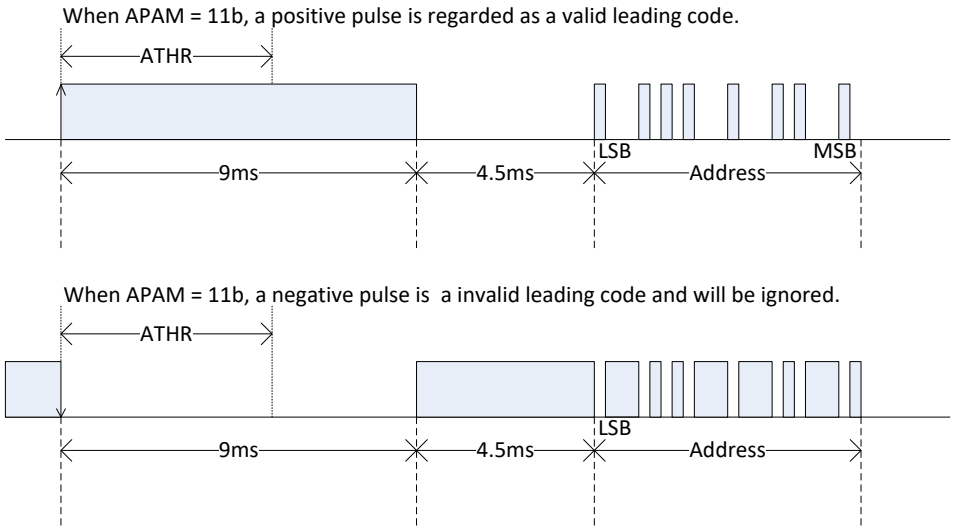
Figure 8-7 NTHR Definition



Active Pulse Accept Mode (APAM)

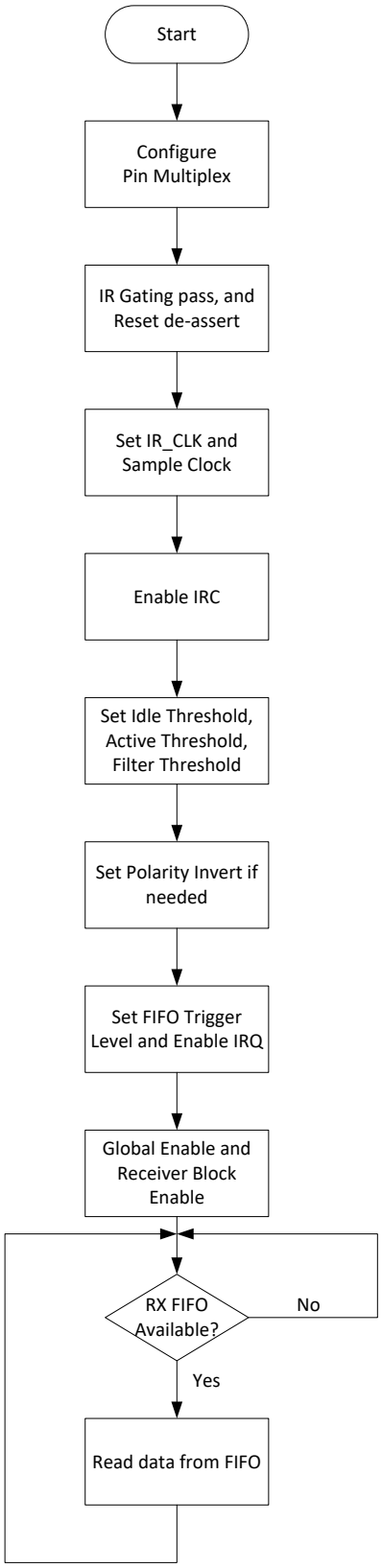
The APAM is used to fit the type of lead code. If a pulse does not fit the type of lead code, it is not regarded as a lead code even if the pulse width reaches ATHR.

Figure 8-8 APAM Definition



8.1.4 Programming Guidelines

Figure 8-9 CIR Receiver Process



8.2 CIR Transmitter (IR_TX)

8.2.1 Overview

The CIR transmitter (CIR_TX) can transfer arbitrary waves which can be modulated with configurable carrier waves such as 38 kHz. CIR_TX only uses lower 8 bits of the 32-bit registers. CIR_TX stores a 16-bit number in 2 registers, where one register contains the higher 8 bits while the other contains the lower 8 bits.

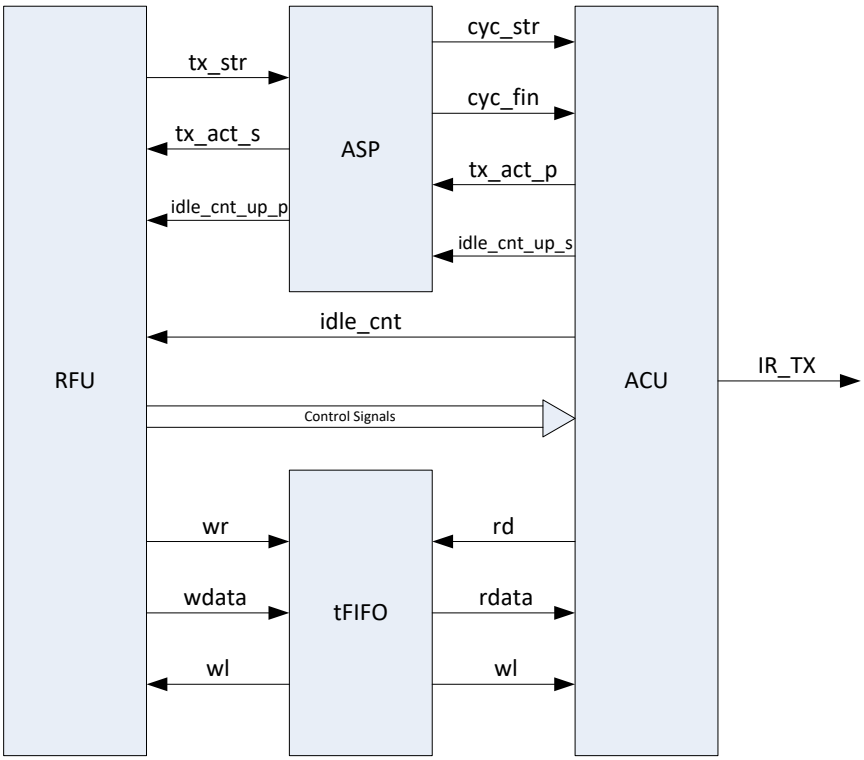
The CIR_TX has the following features:

- One CIR_TX interface in CPUX domain
- Supports industry-standard AMBA Peripheral Bus (APB) and it is fully compliant with the AMBA Specification, Revision 2.0
- Full physical layer implementation
- Arbitrary wave generator
- Configurable carrier frequency
- Handshake mode and waiting mode of DMA
- 128 bytes FIFO for data buffer
- Supports Interrupts and DMA

8.2.2 Block Diagram

The following figure shows a block diagram of the CIR_TX.

Figure 8-10 CIR_TX Block Diagram



8.2.3 Functional Description

8.2.3.1 External Signals

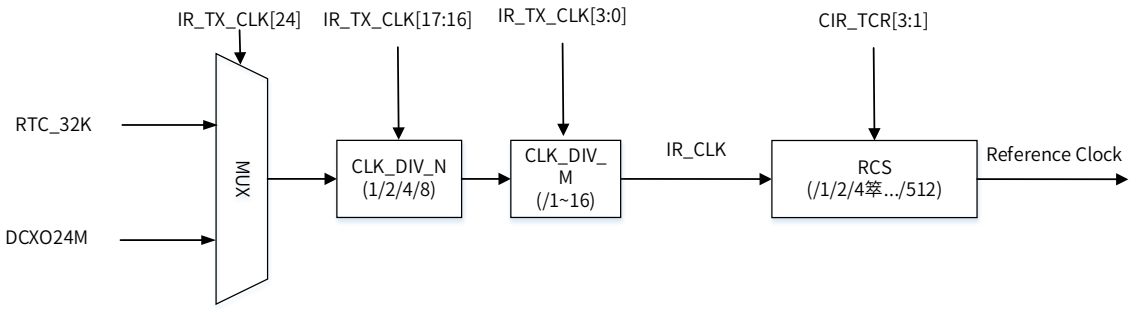
The following table describes the external signals of CIR_TX.

Table 8-3 CIR_TX External Signals

Signal Name	Description	Type
IR-TX	Consumer Infrared Receiver	I

8.2.3.2 Clock Sources

Figure 8-11 CIR_TX Clock Description



8.2.3.3 Function Implementation

The CIR_TX is used to generate a waveform of arbitrary length, arbitrary shape, and no high-speed requirement, and it can change the data into the high-/low-level sequence of a certain length. Every transmitting data is in bytes, the Bit[7] of a byte means whether the level of a transmitting wave is high or low, the Bit[6:0] is the length of this wave. If the current transmitting frequency-division is 1, 0x88 is a high level of 8 cycles, 0x08 is a low level of 8 cycles. If the current transmitting frequency-division is 4, 0x88 is a high level of 32 cycles, 0x08 is a low level of 32 cycles.

The CIR_TX has two transmission modes: non-cycle transmission, and cycle transmission.

The non-cycle transmission is to transmit all the data in TX_FIFO until the FIFO is empty.

The cycle transmission is to transmit all the data in TX_FIFO, after the transmission completion, wait for a certain time to recover the data in TX_FIFO and then send it until a stop signal is detected. The data recovery in FIFO is implemented by clearing the read pointer.

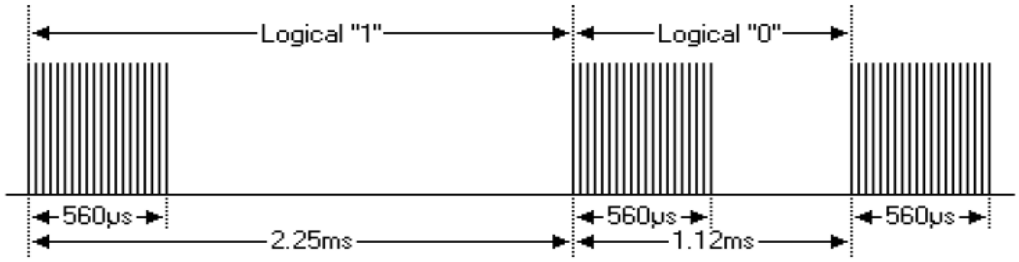
8.2.3.4 Timing Diagram

The IR remote control contains many protocols designed by different manufacturers. Here to NEC protocol as an example, the CIR_TX module uses a variable pulse-width modulation technique to encompass the various formats of infrared encoding for remote-control applications. A message

is started by a 9 ms AGC burst, which is used to set the gain of the earlier CIR receivers. This AGC burst is then followed by a 4.5 ms space, which is then followed by the address and command.

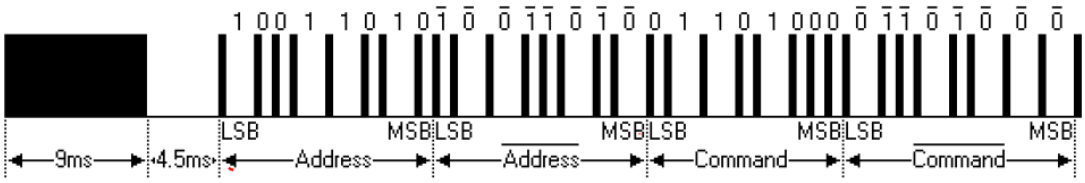
Bit definition: the logical “1” takes 2.25 ms to transmit, while a logical “0” is only 1.12 ms.

Figure 8-12 Definitions of Logical “1” and Logical “0”



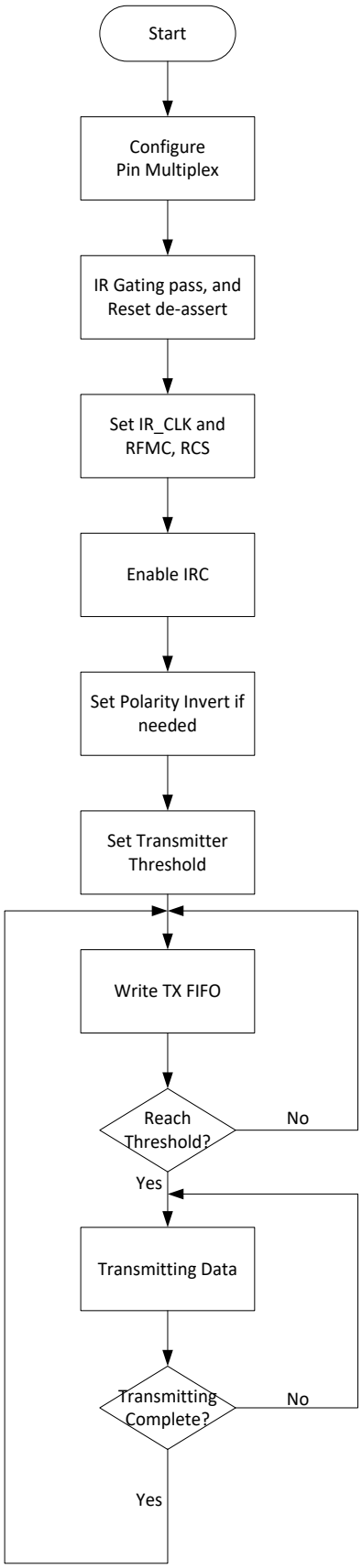
Timing for a message:

Figure 8-13 CIR Message Timing Diagram



8.2.4 Programming Guidelines

Figure 8-14 CIR Transmitter Process



8.3 GMAC

8.3.1 Overview

The Gigabit Medium Access Controller (GMAC) enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2002 standard. It supports 10/100/1000 Mbit/s external PHY with RMII/RGMII interface in full-duplex and half-duplex modes. The internal DMA is designed for packet-oriented data transfers based on a linked list of descriptors.

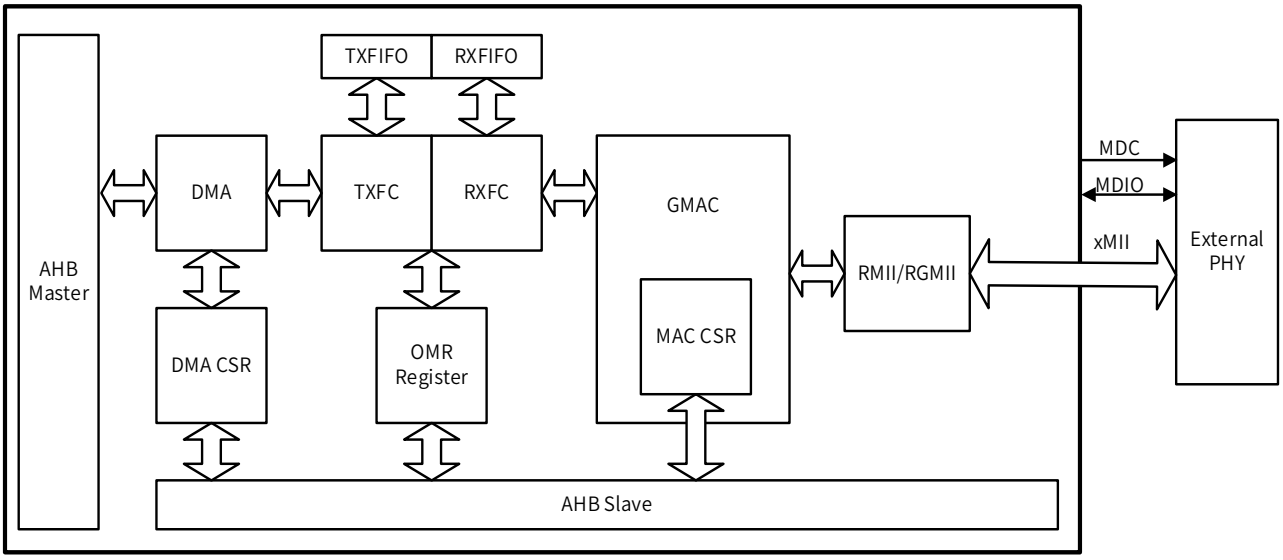
The GMAC has the following features:

- One GMAC interface (GMAC0) for connecting external Ethernet PHY
- 10/100/1000 Mbit/s Ethernet port with RGMII and RMII interfaces
- Compliant with IEEE 802.3-2002 standard
- Supports both full-duplex and half-duplex operations
- Programmable frame length to support Standard or Jumbo Ethernet frames with sizes up to 16 KB
- Supports a variety of flexible address filtering modes
- Separate 32-bit status returned for transmission and reception packets
- Optimization for packet-oriented DMA transfers with frame delimiters
  - Supports linked-list descriptor list structure
  - Descriptor architecture, allowing large blocks of data transfer with minimum CPU intervention; each descriptor can transfer up to 2 KB of data
  - Comprehensive status reporting for normal operation and transfers with errors
- 2 KB TXFIFO for transmission packets and 8 KB RXFIFO for reception packets
- Programmable interrupt options for different operational conditions
- Provides the management data input/output (MDIO) interface for PHY device configuration and management with configurable clock frequencies

8.3.2 Block Diagram

The following figure shows the block diagram of GMAC.

Figure 8-15 GMAC Block Diagram



8.3.3 Functional Description

8.3.3.1 External Signals

The following table describes the pin list of GMAC.

Table 8-4 GMAC External Siganls

Signal Name	Description	Type
RGMII0-RXD0/RMII0-RXD0	RGMII/RMII Receive Data0	I
RGMII0-RXD1/RMII0-RXD1	RGMII/RMII Receive Data1	I
RGMII0-RXD2/RMII0-NULL	RGMII Receive Data2	I
RGMII0-RXD3/RMII0-NULL	RGMII Receive Data3	I
RGMII0-RXCK/ RMII0-NULL	RGMII Receive Clock	I
RGMII0-RXCTRL/RMII0-CRS-DV	RGMII Receive Control/RMII Carrier Sense Receive Data Valid	I
RGMII0-CLKIN/RMII0-RXER	RGMII Transmit Clock from External/RMII Receive Error	I
RGMII0-TXD0/RMII0-TXD0	RGMII/RMII Transmit Data0	O
RGMII0-TXD1/RMII0-TXD1	RGMII/RMII Transmit Data1	O
RGMII0-TXD2/RMII0-NULL	RGMII Transmit Data2	O
RGMII0-TXD3/RMII0-NULL	RGMII Transmit Data3	O
RGMII0-TXCK/RMII0-TXCK	RGMII/RMII Transmit Clock For RGMII, IO type is output; For RMII, IO type is input	I/O
RGMII0-TXCTRL/RMII0-TXEN	RGMII Transmit Control/RMII Transmit Enable	O

Signal Name	Description	Type
RGMII0-MDC	RGMII Management Data Clock	O
RGMII0-MDIO	RGMII Management Data Input/ Output	I/O
RGMII0-EPHY-25M	25 MHz Output for GMAC PHY	O

8.3.3.2 Clock Sources

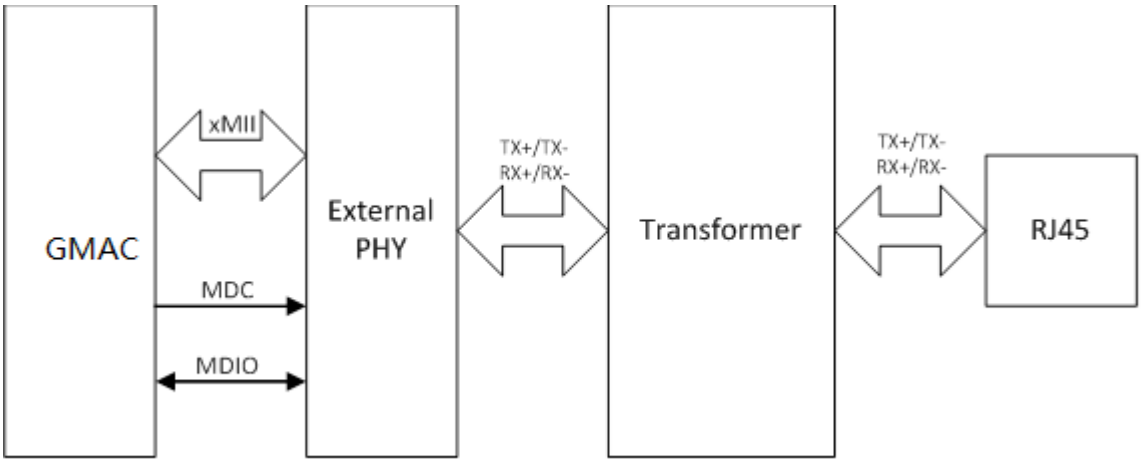
GMAC has two clock sources. The following table describes the clock sources of the GMAC.

Table 8-5 GMAC Clock Sources

Clock Sources	Description	Module
AHB	Bus clock. The bus frequency is 200 MHz.	CCU
GMAC_25M	GMAC 25M clock. The default value is 25 MHz.	

8.3.3.3 Typical Application

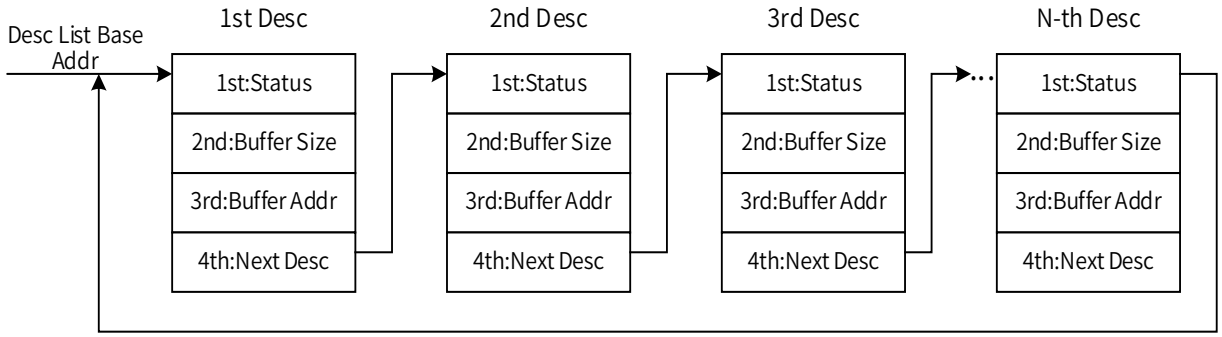
Figure 8-16 GMAC Typical Application



8.3.3.4 GMAC RX/TX Descriptor

The internal DMA of GMAC transfers data between host memory and internal RX/TX FIFO by a linked list of descriptors. Each descriptor consists of four words and contains some necessary information to transfer TX and RX frames. The following figure shows the descriptor list structure. The address of each descriptor must be 32-bit aligned.

Figure 8-17 GMAC RX/TX Descriptor List



8.3.3.5 TX Descriptor

1st Word of TX Descriptor

Bits	Description
31	TX_DESC_CTL When set, the current descriptor can be used by DMA. This bit is cleared by DMA when the whole frame is transmitted or all data in the buffer of the current descriptor are transmitted.
30:17	Reserved
16	TX_HEADER_ERR When set, the checksum of the header for the transmitted frame is wrong.
15	Reserved
14	TX_LENGHT_ERR When set, the length of the transmitted frame is wrong.
13	Reserved
12	TX_PAYLOAD_ERR When set, the checksum of the payload for the transmitted frame is wrong.
11	Reserved
10	TX_CRS_ERR When set, the carrier is lost during transmission.
9	TX_COL_ERR_0 When set, the frame is aborted because of a collision after the contention period.
8	TX_COL_ERR_1 When set, the frame is aborted because of too many collisions.
7	Reserved
6:3	TX_COL_CNT The number of collisions before transmission.
2	TX_DEFER_ERR When set, the frame is aborted because of too much deferral.
1	TX_UNDERFLOW_ERR When set, the frame is aborted because of the TX FIFO underflow error.

Bits	Description
0	TX_DEFER When set in Half-Duplex mode, the GMAC defers the frame transmission.

2nd Word of TX Descriptor

Bits	Description
31	TX_INT_CTL When it is set and the current frame has been transmitted, the TX_INT in Interrupt Status Register will be set.
30	LAST_DESC When it is set, the current descriptor is the last one of the current frame.
29	FIR_DESC When it is set, the current descriptor is the first one of the current frame.
28:27	CHECKSUM_CTL These bits control to insert checksum in the transmit frame.
26	CRC_CTL When it is set, the CRC field is not transmitted.
25:11	Reserved
10:0	BUF_SIZE The size of the buffer specified by the current descriptor.

3rd Word of TX Descriptor

Bits	Description
31:0	BUF_ADDR The address of the buffer specified by the current descriptor.

4th Word of TX Descriptor

Bits	Description
31:0	NEXT_DESC_ADDR The address of the next descriptor. It must be 32-bit aligned.

8.3.3.6 RX Descriptor

1st Word of RX Descriptor

Bits	Description
31	RX_DESC_CTL When it is set, the current descriptor can be used by DMA. This bit is cleared by DMA when the complete frame is received or the buffer of the current descriptor is full.
30	RX_DAF_FAIL When it is set, the current frame does not pass the DA filter.

Bits	Description
29:16	RX_FRM_LEN When LAST_DESC is not set and no error bit is set, this field is the length of received data for the current frame. When LAST_DESC is set, RX_OVERFLOW_ERR and RX_NO_ENOUGH_BUF_ERR are not set, this field is the length of the received frame.
15	Reserved
14	RX_NO_ENOUGH_BUF_ERR When it is set, the current frame is clipped because of no enough buffer.
13	RX_SAF_FAIL When it is set, the current fame does not pass the SA filter.
12	Reserved
11	RX_OVERFLOW_ERR When it is set, a buffer overflow error occurred and the current frame is wrong.
10	Reserved
9	FIR_DESC When it is set, the current descriptor is the first descriptor of the current frame.
8	LAST_DESC When it is set, the current descriptor is the last descriptor of the current frame.
7	RX_HEADER_ERR When it is set, the checksum of the frame header is wrong.
6	RX_COL_ERR When it is set, there is a late collision during the reception in half-duplex mode.
5	Reserved
4	RX_LENGTH_ERR When it is set, the length of the current frame is wrong.
3	RX_PHY_ERR When it is set, the receive error signal from PHY is asserted during the reception.
2	Reserved
1	RX_CRC_ERR When it is set, the CRC field of the received frame is wrong.
0	RX_PAYLOAD_ERR When it is set, the checksum or length of the payload for the received frame is wrong.

2nd Word of RX Descriptor

Bits	Description
31	RX_INT_CTL When it is set and a frame has been received, the RX_INT will not be set.
30:11	Reserved
10:0	BUF_SIZE The size of the buffer is specified by the current descriptor.

3rd Word of RX Descriptor

Bits	Description
31:0	BUF_ADDR The address of the buffer specified by the current descriptor.

4th Word of RX Descriptor

Bits	Description
31:0	NEXT_DESC_ADDR The address of the next descriptor. This field must be 32-bit aligned.

8.3.4 Programming Guidelines

8.3.4.1 GMAC System Configuration

Perform the following steps:

Step 4 Write 0 to GMAC0_BGR_REG[bit16] to assert the module reset.

Step 5 Write 1 to GMAC0_BGR_REG [bit16] to deassert the module reset.

Step 6 Write 1 to GMAC0_BGR_REG [bit0] to enable the bus clock of the module.

Step 7 Configure GMAC_EPHY_CLK_REG0 the pin interfaces of GMAC by setting GPIO module.

Step 8 Configure to set the transmission clock source of RGMII/RMII.

- For RGMII RXCLK/CLK125M:

In RGMII mode, in addition to the configuration of the transmission clock source, it is generally necessary to adjust the timing by configuring the transmission clock delay, reception clock delay, transmission clock reverse, reception clock reverse.

- Write 0 to the bit [13] and write 1 to the bit [2] to select the RGMII interface.
- If selecting RXCLK as the clock source of RGMII, write 2 to the bit [1:0]; if selecting CLK125M as the clock source of RGMII, write 1 to the bit [1:0].
- Write 0 to the bit [3], write 0 to the bit [4], write 31 to the bit [9:5], and write 7 to the bit [12:10] to transmit the reception sequence adjustment.

- For RMII TXCLK:

- Write 1 to the bit [13] and write 0 to the bit [2] to select the RMII interface.
- Write 0 to the bit [0] to select TXCLK as the clock source of RMII.

The configuration value of GMAC_EPHY_CLK_REG0 can refer to the following table.

Table 8-6 GMAC_EPHY_CLK_REG0 Configuration Value

GMAC_EPHY_CLK_REG0	PHY_SEL	RMII_EN	ETXDC	ERXDC	ERXIE	ETXIE	RMII/RGMII	ETCS
	Bit15	Bit13	Bit[12:10]	Bit[9:5]	Bit4	Bit3	Bit2	Bit[1:0]
RGMII	0	0	7	31	0	0	1	1/2
RMII	0	1	0	0	0	0	0	0

8.3.4.2 GMAC Initialization

Step 9 Write 1 to GMAC_BASIC_CTL1[bit0] to perform the software reset.

Step 10 Write 1 to GMAC_BASIC_CTL1 [bit1] to set the DMA priority of TX/RX.

Step 11 Configure GMAC_TX_CTL1 and GMAC_RX_CTL1 to set the configuration of DMA TX and DMA RX.

Step 12 Configure GMAC_INT_EN to set the corresponding interrupts and shield the needless interrupts.

Step 13 Configure GMAC_TX_DMA_LIST and GMAC_RX_DMA_LIST to set the first address of the TX descriptor and the RX descriptor, respectively.

Step 14 Configure GMAC_TX_CTL0 and GMAC_RX_CTL0 to set the TX and RX parameters. Configure GMAC_BASIC_CTL0 to set the speed, duplex mode, loopback configuration. (If enabled the auto-negotiation, the configuration is performed as a result of the negotiation)

Step 15 Configure GMAC_RX_FRM_FLT to set the RX frame filter.

Step 16 Configure GMAC_TX_FLOW_CTL and GMAC_RX_CTL0 to set the control mechanism of TX and RX.

Step 17 Clear all interrupt flags.

Step 18 Write 1 to GMAC_TX_CTL0 [bit31] and write 1 to GMAC_TX_CTL0 [bit31] to enable the TX and RX functions.

8.3.5 Register List

Module Name	Base Address
GMAC0	0x04500000

Register Name	Offset	Description
GMAC_BASIC_CTL0	0x0000	GMAC Basic Control Register0
GMAC_BASIC_CTL1	0x0004	GMAC Basic Control Register1

8.4 GMAC200

8.4.1 Overview

The GMAC (QOS) enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2015 standard.

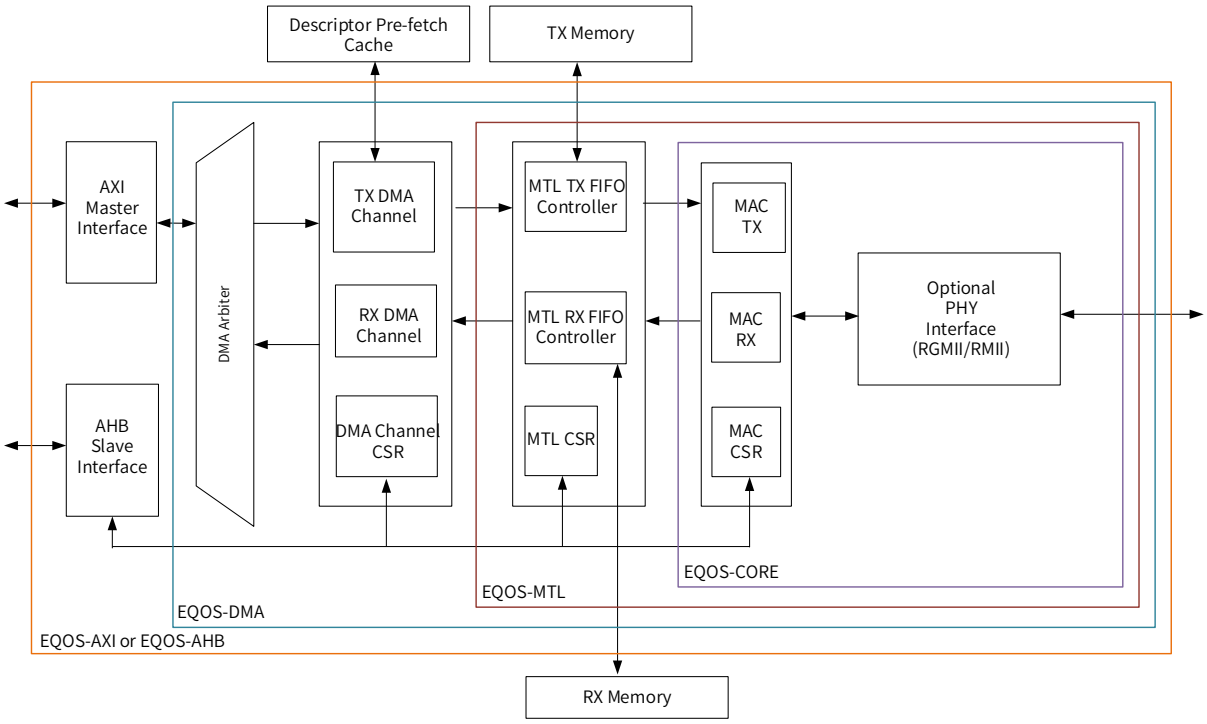
The GMAC has the following features:

- One GMAC interface (GMAC1) for connecting to external Ethernet PHY
- Compliant with the IEEE 802.3-2015 standard
- IEEE 1588-2008 for precision networked clock synchronization, support Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008
- Supports 10/100/1000 Mbit/s data transfer rates
- Supports RMII/RGMII PHY interface
- AMBA4 AXI master interface with 64-bit data transfer
- Supports both full-duplex and half-duplex operation
- Full-duplex flow control
- Programmable frame length to support Standard or Jumbo Ethernet frames with sizes up to 16 KB
- Source Address field and VLAN insertion or replacement, Double VLAN
- Transmit TCP/IP Checksum Offload
- Supports a variety of flexible address filtering modes (include Hash filter function)
- Optimization for packet-oriented DMA transfers with frame delimiters
  - Supports linked-list descriptor list structure
  - Descriptor architecture, allowing large blocks of data transfer with minimum CPU intervention: each descriptor can transfer up to 32 KB of data
  - Comprehensive status reporting for normal operation and transfers with errors
- Supports 4KB TXFIFO for transmission packets and 8KB RXFIFO for reception packets
- Supports 16 Descriptors Descriptor Pre-fetch cache for TX DMA and RX DMA
- Programmable interrupt options for different operational conditions
- Supports MDIO Interface for PHY device configuration and management
- Configurable big-endian and little-endian mode for Transmit and Receive paths
- Supports statistics on the received and transmitted packets
- Supports split header

8.4.2 Block Diagram

The following figure shows the block diagram of the GMAC.

Figure 8-18 GMAC Block Diagram



The following table describes the components of the GMAC.

Table 8-7 GMAC Components

Components	Description
Interfaces	• AHB slave interface The 32-bit AHB slave interface provides access to the DMA, MTL, and MAC CSR space. • AXI master interface The DMA controller interfaces with the application through the AMBA4 AXI interface. The AMBA4 AXI bus interface provides the characteristics to support highly-effective data traffic throughput. • PHY interface The PHY interface can be configured to GMII/MII RGMII and RMII.
DMA	One DMA Channel has independent Transmit (TX) and Receive (RX) engines, and a CSR space. The TX engine transfers data from the system memory to the device port (MTL), whereas the Rx engine transfers data from the device port to the system memory. The DMA engine uses descriptors to efficiently move data from source to destination with minimal application CPU intervention.

Components	Description
MTL	The MAC Transaction Layer (MTL) provides the FIFO memory Interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. One MTL queue. The MTL layer has two data paths: Transmit path and Receive Path.
MAC	The MAC supports many interfaces towards the PHY chip.

8.4.3 Functional Description

8.4.3.1 External Signals

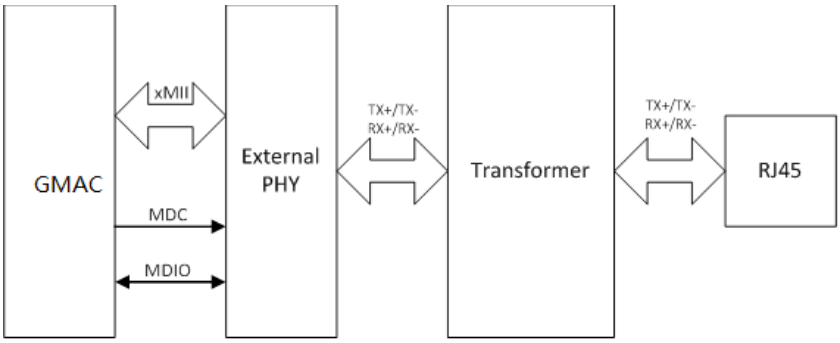
The following table describes the external signals of the GMAC.

Table 8-8 GMAC External Signals

Signal Name	Description	Type
RGMII1-RXD0/RMII1-RXD0	RGMII1/RMII1 Receive Data0	I
RGMII1-RXD1/RMII1-RXD1	RGMII1/RMII1 Receive Data1	I
RGMII1-RXD2/RMII1-NULL	RGMII1 Receive Data2	I
RGMII1-RXD3/RMII1-NULL	RGMII1 Receive Data3	I
RGMII1-RXCK/ RMII1-NULL	RGMII1 Receive Clock	I
RGMII1-RXCTL/RMII1-CRS-DV	RGMII1 Receive Control/RMII1 Carrier Sense Receive Data Valid	I
RGMII1-CLKIN/RMII1-RXER	RGMII1 Transmit Clock from External/RMII1 Receive Error	I
RGMII1-TXD0/RMII1-TXD0	RGMII1/RMII1 Transmit Data0	O
RGMII1-TXD1/ RMII1-TXD1	RGMII1 Transmit Data1	O
RGMII1-TXD2/RMII1-NULL	RGMII1 Transmit Data2	O
RGMII1-TXD3/RMII1-NULL	RGMII1 Transmit Data3	O
RGMII1-TXCK/RMII1-TXCK	RGMII1/RMII1 Transmit Clock For RGMII, IO type is output; For RMII, IO type is input	I/O
RGMII1-TXCTL/RMII1-TXEN	RGMII1 Transmit Control/RMII1 Transmit Enable	O
RGMII1-MDC	RGMII1 Management Data Clock	O
RGMII1-MDIO	RGMII1 Management Data Input/ Output	I/O
RGMII1-EPHY-25M	25 MHz Output for GMAC PHY	O

8.4.3.2 Typical Application

Figure 8-19 Typical Application



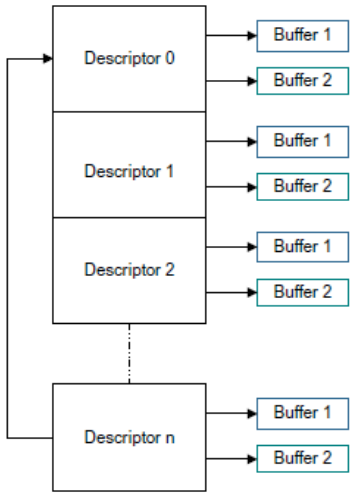
8.4.3.3 Descriptors

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The GMAC supports the following two types of descriptors:

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted.

The GMAC supports the ring structure for DMA descriptor. Each normal descriptor contains two buffers and two address pointers.

Figure 8-20 Descriptor Ring Structure



In Ring structure, the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

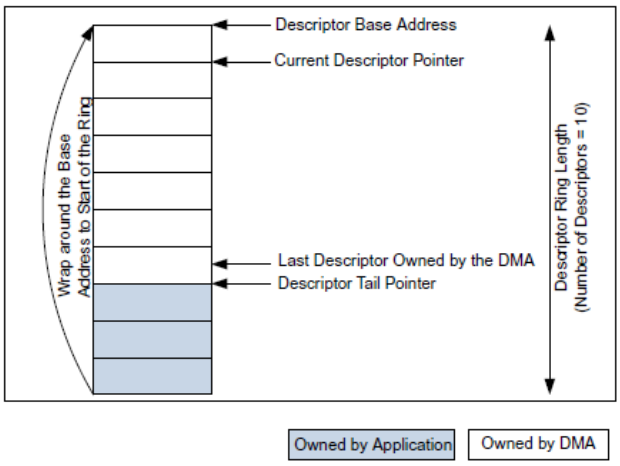
- Transmit Descriptor Ring Length Register

- Receive Descriptor Ring Length Register

The Descriptor Tail Pointer Register contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

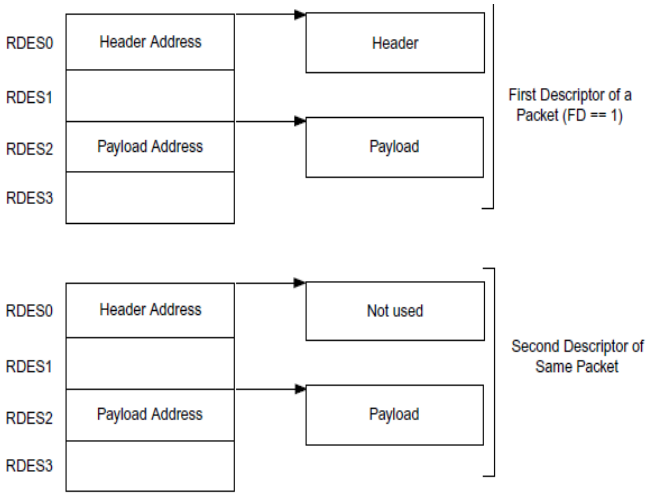
Current Descriptor Pointer == Descriptor Tail Pointer;

Figure 8-21 DMA Descriptor Ring



GMAC supports splitting the incoming receive packet header such that the header and the payload are stored in different buffers (buffer-1 and buffer-2) in the system memory.

Figure 8-22 Descriptors with Split Header Feature

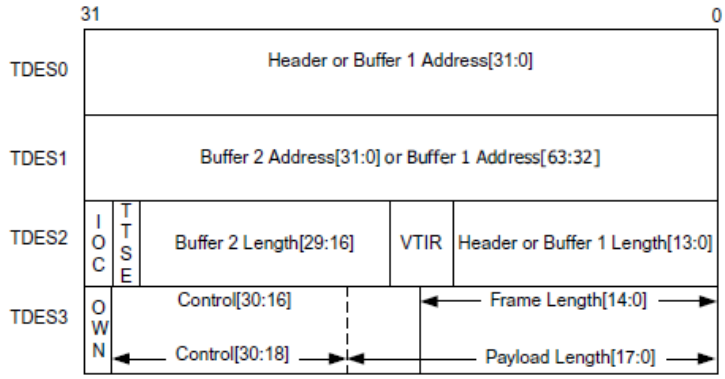


Transmit Descriptor

Transmit Normal descriptor has two formats: Read format and Write-Back format.

- Transmit Normal Descriptor (Read Format)

Figure 8-23 Transmit Descriptor Read Format



Bit	Name	Description
TDES0 Normal Descriptor (Read Format)		
31:0	BUF1AP	Buffer 1 Address Pointer or TSO Header Address Pointer These bits indicate the physical address of Buffer 1. These bits indicate the TmSO Header Address pointer when the following bits are set: • TSE bit of TDES3 • FD bit of TDES3
TDES1 Normal Descriptor (Read Format)		
31: 0	BUF2AP	Buffer 2 or Buffer 1 Address Pointer This bit indicates the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment. In 40- or 48-bit addressing mode, these bits indicate the most-significant 8- or 16- bit of the Buffer 1 Address Pointer.
TDES2 Normal Descriptor (Read Format)		
31	IOC	Interrupt on Completion This bit controls the setting of TI and ETI status bits in the DMA_CH#_Status register. When ETIC = 1 and TDES2[LD] = 0, this bit sets the ETI bit. When TDES3[LD] = 1, this bit sets the TI status bit.
30	TTSE/TMWD	Transmit Timestamp Enable or External TSO Memory Write Enable This bit enables the IEEE1588 time stamping for Transmit packet referenced by the descriptor, if TSE bit is not set. If TSE bit is set and external TSO memory is enabled, setting this bit disables external TSO memory writing for this packet.
29:16	B2L	Buffer 2 Length The driver sets this field. When set, this field indicates Buffer 2 length.

Bit	Name	Description
15:14	VTIR	VLAN Tag Insertion or Replacement These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN Tag Insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits: 2'b00: Do not add a VLAN tag. 2'b01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 2'b10: Insert a VLAN tag with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. 2'b11: Replace the VLAN tag in packets with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets. These bits are valid when the Enable SA and VLAN Insertion on TX option is selected while configuring the core.
13:0	HL or B1L	Header Length or Buffer 1 Length For Header length only bits [9:0] are taken. The size 13:0 is applicable only when interpreting buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length in bytes from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.
TDES3 Normal Descriptor (Read Format)		
31	OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	Context Type This bit should be set to 1'b0 for normal descriptor.
29	FD	First Descriptor When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.

Bit	Name	Description
27:26	CPC	CRC Pad Control This field controls the CRC and Pad Insertion for TX packet. This field is valid only when the first descriptor bit (TDES3[29]) is set. The following list describes the values of Bits [27:26]: 2'b00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 2'b01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit Buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 2'b10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 2'b11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. This field is valid only for the first descriptor. **Note: When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.**

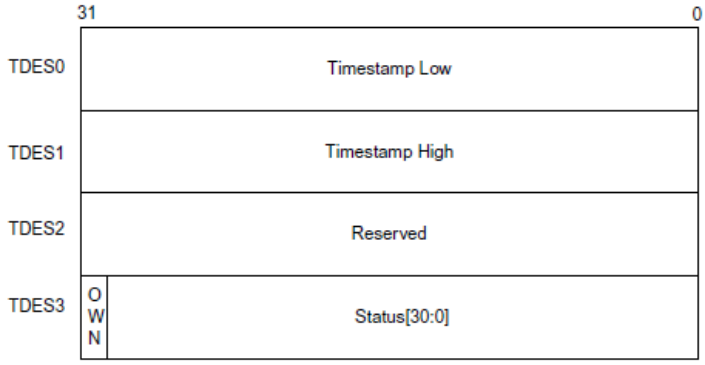
Bit	Name	Description
25:23	SAIC	SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must set the CRC Pad Control bits appropriately when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits [24:23]: 2'b00: Do not include the source address 2'b01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 2'b10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 2'b11: Reserved These bits are valid in the EQOS-DMA, EQOS-AXI, and EQOS-AHB configurations when the Enable SA and VLAN Insertion on TX option is selected while configuring the core and when the First Segment control bit (TDES3 [29]) is set. This field is valid only for the first descriptor.
22:19	SLOTNU Mor THL	SLOTNUM: Slot Number Control Bits in AV Mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field DMA_CH#_Slot_Function_Control_Status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels. THL: TCP/UDP Header Length If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. The value must be equal to 2 for UDP header. This field is valid only for the first descriptor.
18	TSE	TCP Segmentation Enable When this bit is set, the DMA performs the TCP/UDP segmentation or UDP fragmentation for a packet depending on the TSE_MODE [1:0] bit of the DMA_CH(#i) _Tx_Control register. This bit is valid only if the FD bit is set.

Bit	Name	Description
17:16	CIC/TPL	Checksum Insertion Control or TCP Payload Length These bits control the checksum calculation and insertion. The following list describes the bit encoding: 2'b00: Checksum Insertion Disabled. 2'b01: Only IP header checksum calculation and insertion are enabled. 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the Enable Transmit TCP/IP Checksum Offload option is selected and the TSE bit is reset. When the TSE bit is set, this field contains the upper bits [17:16] of the TCP Payload (or IP Payload for UDP fragmentation). This allows the TCP/UDP packet length field to be spanned across TDES3[17:0] to provide 256 KB packet length support. This field is valid only for the first descriptor.
15	TPL	Reserved or TCP Payload Length When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is Bit 15 of the TCP payload length [17:0]. This field is valid only when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected while configuring the core.
14:0	FL/TPL	Frame Length or TCP Payload Length This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length in case of segmentation and IP payload in case of UDP fragmentation. In case of segmentation, this length does not include Ethernet header or TCP/UDP/IP header length. In case of fragmentation, this length does not include Ethernet header and IP header. When DWRR/WFQ algorithm is NOT enabled, value written into this field is not used when TSE = 0.

- Transmit Normal Descriptor (Write-Back Format)

The write-back format of the Transmit Descriptor includes timestamp low, timestamp high, OWN, and Status bits. The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 8-24 Transmit Context Descriptor



Bit	Name	Description
TDES0 Normal Descriptor (Write Format)		
31:0	TTSL	Transmit Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.
TDES1Normal Descriptor (Write Format)		
31: 0	TTSH	Transmit Packet Timestamp High The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding transmit packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.
TDES2 Normal Descriptor (Write Format)		
31:0	/	Reserved
TDES3 Normal Descriptor (Write Format)		
31	OWN	Own Bit When this bit is set, it indicates that the GMAC DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 'b0.
30	CTXT	Context Type This bit should be set to 'b0 for Normal descriptor.
29	FD	First Descriptor This bit indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor This bit is set 'b1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.

Bit	Name	Description
27:24	/	Reserved
23	DE	Descriptor Error Descriptor Error When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor Errors can be: • Incorrect sequence from the context descriptor. For example, a location after the first descriptor for a packet. • All 1s • CTXT is set to 1 along with LD or FD bits set to 1. **Note:** • **When Descriptor Error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the DMA_CH#_Status register.** • **Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.**
22:18	/	Reserved
17	TTSS	TX Timestamp Status This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set. This bit is valid only when IEEE1588 timestamping feature is enabled; otherwise, it is reserved.
16	EUE	ECC Uncorrectable Error Status Indicates the ECC uncorrectable error in the TSO memory. **Note: Uncorrectable error in Transmit FIFO memory is reported with (Bit 13) FF = 1. This is because, all such packets are flushed by GMAC.**

Bit	Name	Description
15	ES	Error Summary This bit indicates the logical OR of the following bits: TDES3[0]: IP Header Error TDES3[14]: Jabber Timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error This bit is also set when EUE (bit 16) is set.
14	JT	Jabber Timeout Jabber Timeout This bit indicates that the MAC transmitter has experienced a jabber time-out. This bit is set only when the JD bit of the MAC_Configuration register is not set.
13	FF	Packet Flushed Packet Flushed This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	Payload Checksum Error This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the Payload Length field of the IP Header or the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when Bus Error is detected during packet transfer. When the Full Checksum Offload engine is not enabled, this bit is reserved.
11	LoC	Loss of Carrier This bit indicates that Loss of Carrier occurred during packet transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
10	NC	No Carrier This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.

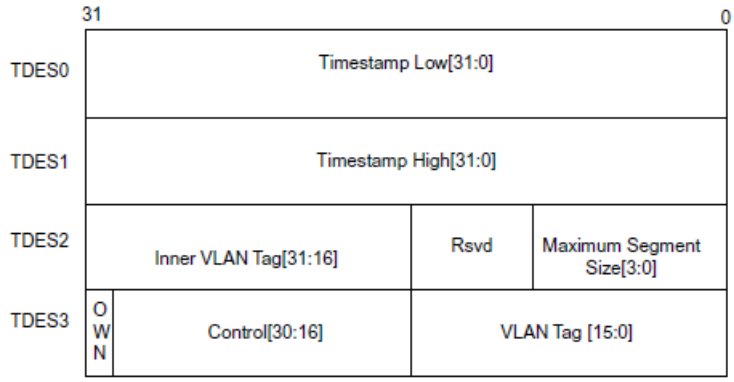
Bit	Name	Description
9	LC	Late Collision This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode and 512 byte times including Preamble and Carrier Extension in GMII mode). This bit is not valid if Underflow Error is set.
8	EC	Excessive Collision This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	Collision Count This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	Excessive Deferral This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits' times in 1000 Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the MAC_Configuration register. When TBS is enabled in full duplex mode and this bit is set, it indicates that the frame has been dropped after the expiry time has reached.
2	UF	Underflow Error This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: ● The DMA encountered an empty Transmit Buffer while transmitting the packet. ● The application filled the MTL TX FIFO slower than the MAC transmit rate. The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the MTL_Interrupt_Status register.
1	DB	Deferred Bit This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.

Bit	Name	Description
0	IHE	IP Header Error When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when TX Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload. In full duplex mode, when EST/Qbv is enabled and this bit is set, it indicates the frame drop status due to Frame Size error or Schedule Error.

- Transmit Context Descriptor

The context descriptor is used to provide the timestamps for one-step timestamp correction, VLAN Tag ID for VLAN insertion feature. The Transmit Context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction and VLAN Tag ID for VLAN insertion feature. Write back is done on a context descriptor only to reset the OWN bit.

Figure 8-25 Transmit Context Descriptor



Bit	Name	Description
TDES0 Context Descriptor		
31:0	TTSL	Transmit Packet Timestamp Low For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Bit	Name	Description
TDES1 Context Descriptor		
31: 0	TTSH	Transmit Packet Timestamp High For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.
TDES2 Context Descriptor		
31:16	IVT	Inner VLAN Tag When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	/	Reserved
13:0	MSS	Maximum Segment Size When the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected, the driver can provide maximum segment size in this field. This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.
TDES3 Context Descriptor		
31	OWN	Own Bit When this bit is set, it indicates that the GMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit immediately after the read.
30	CTXT	Context Type This bit should be set to 1'b1 for Context descriptor.
29:28	/	Reserved
27	OSTC	One-Step Timestamp Correction Enable When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
26	TCMS SV	One-Step Timestamp Correction Input or MSS Valid When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	/	Reserved

Bit	Name	Description
23	CDE	Context Descriptor Error When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the context descriptor. Context Descriptor Errors can be: ● Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. ● All 1s. ● CD, LD, and FD bits set to 1. Note: ● **When the Context Descriptor Error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the DMA_CH#_Status register.** ● **Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.** ● **This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported).**
22:20	Rsvd	Reserved
19:18	IVTIR	Inner VLAN Tag Insert or Replace When this bit is set, these bits request the MAC to perform Inner VLAN tagging or un-tagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: ● 2'b00: Do not add the inner VLAN tag. ● 2'b01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. ● 2'b10: Insert an inner VLAN tag with the tag value programmed in the MAC_In-ner_VLAN_Incl register or context descriptor. ● 2'b11: Replace the inner VLAN tag in packets with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. This option should be used only with the VLAN frames. These bits are valid when the Enable SA and VLAN Insertion on Tx and Enable Double VLAN Processing options are selected.

Bit	Name	Description
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the MAC_VLAN_Incl register is reset.

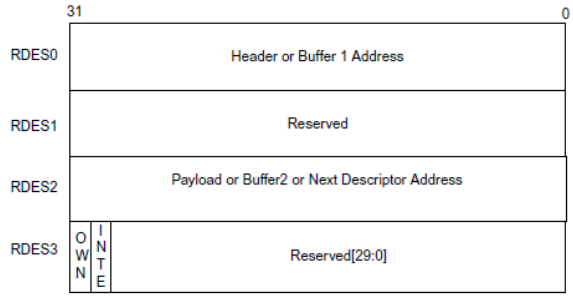
Receive Descriptor

All RX descriptors are prepared by the software and given to the DMA as “Normal” Descriptors with the content as shown in Receive Normal Descriptor (Read Format). The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The format of this status is given in the “Receive Normal Descriptor (Write-Back Format)”.

- Receive Normal Descriptor (Read Format)

The read format for a Receive Normal descriptor is made up of a header or Buffer 1 address, reserved field, payload or Buffer 2 or Next Descriptor address, a 30-bit reserved filed, OWN bit, and an interrupt bit.

Figure 8-26 Receive Normal Descriptor (Read Format)



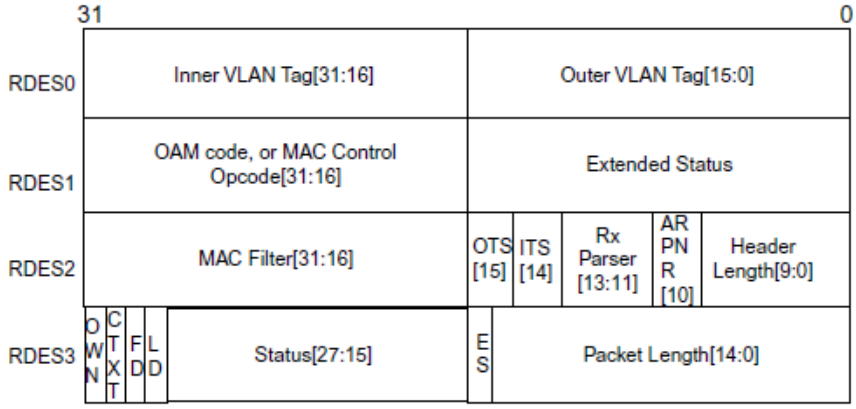
Bit	Name	Description
RDES0 Normal Descriptor (Read Format)		
31:0	BUF1AP	Header or Buffer 1 Address Pointer When the SPH bit of Control register of a channel is reset, these bits indicate the physical address of Buffer 1. When the SPH bit is set, these bits indicate the physical address of Header Buffer where the Rx DMA writes the L2/L3/L4 header bytes of the received packet.

Bit	Name	Description
RDES1 Normal Descriptor (Read Format)		
31: 0	Reserved or BUF1AP	In 64-bit addressing mode, this field contains the most-significant 32 bits of the Buffer 1 Address Pointer. Otherwise, this field is reserved.
RDES2 Normal Descriptor (Read Format)		
31: 0	BUF2A P	Buffer 2 Address Pointer These bits indicate the physical address of Buffer 2. When the SPH bit of the DMA_CH#_Control register is set, the buffer address pointer must be bus width-aligned, that is, RDES2[3:0, 2:0, or 1:0] = 0 corresponding to 128, 64, or 32 bus width. LSBs are ignored internally.
RDES3 Normal Descriptor (Read Format)		
31	OWN	Own Bit When this bit is set, it indicates that the GMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: ● The DMA completes the packet reception. ● The buffers associated with the descriptor are full.
30	IOC	Interrupt Enabled on Completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Rsvd	Reserved
25	BUF2V	Buffer 2 Address Valid When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address, to which the Buffer 2 address in RDES2 is pointing, to write received packet data.
24	BUF1V	Buffer 1 Address Valid When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid. The application must set this value if the address pointed to by Buffer 1 address in RDES1 can be used by the DMA to write received packet data.
23:16	Rsvd	Reserved
30	IOC	Interrupt Enabled on Completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Rsvd	Reserved
15:0	BUF2A P	Buffer2 Address Pointer In the 64 bit addressing mode, this field contains the most-significant 16 bits of the Buffer2 address pointer. Otherwise, this field is reserved.

Bit	Name	Description
24	BUF1V	Buffer 1 Address Valid
23:1 6	Rsvd	Reserved
15:0	BUF2A P	Buffer2 Address Pointer In the 64 bit addressing mode, this field contains the most-significant 16 bits of the Buffer2 address pointer. Otherwise, this field is reserved.

- Receive Normal Descriptor (Write-Back Format)

Figure 8-27 Receive Normal Descriptor (Write-Back Format)



Bit	Name	Description
RDES0 Normal Descriptor (Write-Back Format)		
31:1 6	IVT	Inner VLAN Tag This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	Outer VLAN Tag This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.
RDES1 Normal Descriptor (Write-Back Format)		
31:16	OPC	OAM Sub-Type Code, or MAC Control Packet opcode OAM Sub-Type Code If Bits[18:16] of RDES3 are set to 3'b111, this field contains the OAM sub-type and code fields. MAC Control Packet opcode The bits[15:8] of RDES3 contains the subtype and bits [7:0] contains the code.
15	TD	Timestamp Dropped This bit indicates that the timestamp was captured for this packet but it got dropped in the MTL Rx FIFO because of overflow. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.

Bit	Name	Description
14	TSA	Timestamp Available When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	PTP Version This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
12	PFT	PTP Packet Type This bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
11:8	PMT	PTP Message Type These bits are encoded to give the type of the message received: 0000: No PTP message received 0001: SYNC (all clock types) 0010: Follow_Up (all clock types) 0011: Delay_Req (all clock types) 0100: Delay_Resp (all clock types) 0101: Pdelay_Req (in peer-to-peer transparent clock) 0110: Pdelay_Resp (in peer-to-peer transparent clock) 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000: Announce 1001: Management 1010: Signaling 1011–1110: Reserved 1111: PTP packet with Reserved message type These bits are available only when you select the Timestamp feature.
7	IPCE	IP Payload Error When this bit is set, it indicates either of the following: ● The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. ● The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. ● The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.

Bit	Name	Description
6	IPCB	IP Checksum Bypassed This bit indicates that the checksum offload engine is bypassed. This bit is available when you select the Enable Receive TCP/IP Checksum Check feature.
5	IPV6	IPv6 header Present This bit indicates that an IPV6 header is detected. When the Enable Split Header Feature option is selected and the SPH bit of Control Register of a channel is set, the IPV6 header is available in the header buffer area to which RDES0 is pointing.
4	IPV4	IPV4 Header Present This bit indicates that an IPV4 header is detected. When the SPH bit of RDES3 is set, the IPV4 header is available in the header buffer area to which RDES0 is pointing.
3	IPHE	IP Header Error When this bit is set, it indicates either of the following: ● The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. ● The IP datagram version is not consistent with the Ethernet Type value. ● Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either Bit 5 or Bit 4 is set. This bit is available when you select the Enable Receive TCP/IP Checksum Check feature.
2:0	PT	Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): 3'b000: Unknown type or IP/AV payload not processed 3'b001: UDP 3'b010: TCP 3'b011: ICMP 3'b110: AV Tagged Data Packet 3'b111: AV Tagged Control Packet 3'b101: AV Untagged Control Packet 3'b100: IGMP if IPV4 Header Present bit is set else DCB (LLDP) Control Packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

Bit	Name	Description
RDES2 Normal Descriptor (Write-Back Format)		
31:29	L3L4FM	Layer 3 and Layer 4 Filter Number Matched These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: 000: Filter 0 001: Filter 1 010: Filter 2 011: Filter 3 100: Filter 4 101: Filter 5 110: Filter 6 111: Filter 7 This field is valid only when Bit 28 or Bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter. **Note: This status is not available when Flexible RX Parser is enabled.**
28	L4FM	Layer 4 Filter Match When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: ● Layer 3 fields are not enabled and all enabled Layer 4 fields match. ● All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by Bits[31:29]. **Note: This status is not available when Flexible RX Parser is enabled.**
27	L3FM	Layer 3 Filter Match When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: ● All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed. ● All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by Bits[31:29]. **Note: This status is not available when Flexible RX Parser is enabled.**

Bit	Name	Description
26:19	MADRM	MAC Address Match or Hash Value When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the hash value computed by the MAC. A packet passes the hash filter when the bit corresponding to the hash value is set in the hash filter register. **Note: This status is not available when Flexible RX Parser is enabled.**
18	HF	Hash Filter Status When this bit is set, it indicates that the packet passed the MAC address hash filter. Bits[26:19] indicate the hash value. **Note: This status is not available when Flexible RX Parser is enabled.**
17	DAF/RXPI	Destination Address Filter Fail When Flexible RX Parser is disabled, and this bit is set, it indicates that the packet failed the DA Filter in the MAC. When Flexible RX Parser is enabled, this bit is set to indicate that the packet parsing is incomplete (RXPI) due to ECC error. **Note: When this bit is set, ES bit of RDES3 is also set.**
16	SAF/RXP D	SA Address Filter Fail When Flexible RX Parser is disabled, and this bit is set, it indicates that the packet failed the SA Filter in the MAC. When Flexible RX Parser is enabled, this bit is set to indicate that the packet is dropped (RXPD) by the parser. **Note: When this bit is set, ES bit of RDES3 is also set.**
15	OTS	VLAN Filter Status When set, this bit indicates that the VLAN Tag of the received packed passed the VLAN filter. This bit is valid only when DWC_EQOS_ERVFE is not enabled. If DWC_EQOS_ERVFE is enabled, the bit is redefined as Outer VLAN Tag Filter Status (OTS). This bit is valid for both Single and Double VLAN Tagged frames
14	ITS	Inner VLAN Tag Filter Status (ITS) This bit is valid only when DWC_EQOS_ERVFE is enabled. This bit is valid only for Double VLAN Tagged frames, when Double VLAN Processing is enabled. For more information, see the Filter Status topic.

Bit	Name	Description
13:11	RxParser	3'b000: RX Parser filter passed 3'b001: Packet rejected in Rx Parser 3'b010: Rx Parser overflow errors 3'b011: Incomplete Rx parsing of the packet 3'b100: AF = 1, RF = 1, Bypass 3'b101: RA = 1 and L2 filter fails (Rx parser bypass) 3'b110: Reserved 3'b111: Incomplete parsing due to ECC error
10	ARPNR	ARP Reply Not Generated When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time). This bit is reserved when the Enable IPv4 ARP Offload option is not selected.
9:0	HL	L3/L4 Header Length This field contains the length of the header of the packet split by the MAC at L3 or L4 header boundary as identified by the MAC receiver. This field is valid only when the first descriptor bit is set (FD = 1). The header data is written to the Buffer 1 address of corresponding descriptor. If header length is zero, this field is not valid. It implies that the MAC did not identify and split the header. This field is valid when the Enable Split Header Feature option is selected.
RDES3 Normal Descriptor (Write-Back Format)		
31	OWN	Own Bit When this bit is set, it indicates that the GMAC DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: ● The DMA completes the packet reception ● The buffers associated with the descriptor are full

Bit	Name	Description
30	CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} 2'b00: Intermediate Descriptor 2'b01: First Descriptor 2'b10: Reserved 2'b11: Descriptor Error (due to all 1s) **Note: When Descriptor Error occurs, the Receive DMA closes the receive descriptor indicating Descriptor Error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. Receive DMA sets the CDE field of the DMA_CH#_Status register but does not the RI field even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.**
29	FD	First Descriptor When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. See the CTXT bit description for details of using the CTXT bit and FD bit together.
28	LD	Last Descriptor When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	Receive Status RDES2 Valid When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	Receive Status RDES1 Valid When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	Receive Status RDES0 Valid When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
24	CE	CRC Error When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.

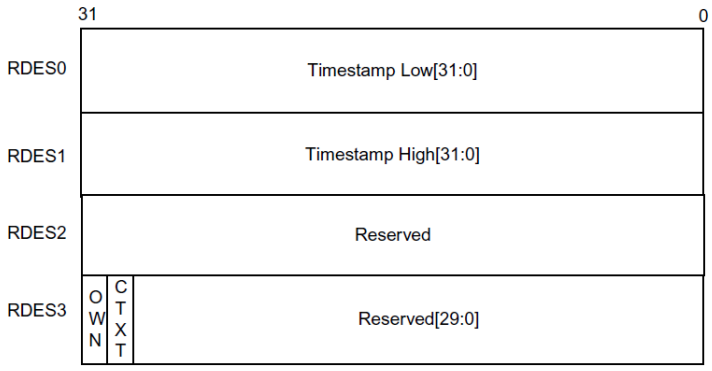
Bit	Name	Description
23	GP	Giant Packet When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). **Note: Giant packet indicates only the packet length. It does not cause any packet truncation.**
22	RWT	Receive Watchdog Timeout When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	Overflow Error When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. **Note: This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.**
20	RE	Receive Error When this bit is set, it indicates that the gmii_rxer_i signal is asserted while the gmii_rxdv_i signal is asserted during packet reception. This error also includes carrier extension error in the GMII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	Dribble Bit Error When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	Length/Type Field This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 3'b000: The packet is a length packet 3'b001: The packet is a type packet. 3'b011: The packet is a ARP Request packet type 3'b100: The packet is a type packet with VLAN Tag 3'b101: The packet is a type packet with Double VLAN Tag 3'b110: The packet is a MAC Control packet type 3'b111: The packet is a OAM packet type 3'b010: Reserved

Bit	Name	Description
15	ES	Error Summary When this bit is set, it indicates the logical OR of the following bits: RDES3[24]: CRC Error RDES3[19]: Dribble Error RDES3[20]: Receive Error RDES3[22]: Watchdog Timeout RDES3[21]: Overflow Error RDES3[23]: Giant Packet RDES2[17]: Destination Address Filter Fail, when Flexible RX Parser is enabled RDES2[16]: SA Address Filter Fail, when Flexible RX Parser is enabled. This field is valid only when the LD bit of RDES3 is set.
14:0	PL	Packet Length These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

- Receive Context Descriptor

This descriptor is read-only for the application. Only the DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. The Bit 30 of RDES3 indicates the context type descriptor.

Figure 8-28 Receive Context Descriptor



Bit	Name	Description
RDES0 Context Descriptor		
31: 0	RTSL	Receive Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.
RDES1 Context Descriptor		
31: 0	RTSH	Receive Packet Timestamp High The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.
RDES2 Context Descriptor		
31:0	/	Reserved
RDES3 Context Descriptor		
31	OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: ● The DMA completes the packet reception. ● The buffers associated with the descriptor are full.
30	CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor. DMA writes 2'b11 to indicate a descriptor error due to all 1s. When CTXT and DE bits are used together, {CTXT, DE} 2'b00: Reserved 2'b01: Reserved 2'b10: Context Descriptor 2'b11: Descriptor Error **Note: When Descriptor Error occurs, the Receive DMA closes the receive descriptor indicating Descriptor Error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data Receive DMA sets the CDE bit in DMA_CH#_Status register but does not set the RI field even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.**
29	DE	Descriptor Error See the CTXT bit description for details of using theDE bit along with CTXT bit.

Bit	Name	Description
28:0	/	Reserved

8.5 General Purpose ADC (GPADC)

8.5.1 Overview

The General Purpose ADC (GPADC) can convert the external signal into a certain proportion of digital value, to realize the measurement of analog signal, which can be applied to power detection and key detection. This ADC is a type of successive approximation register (SAR) A/D converter.

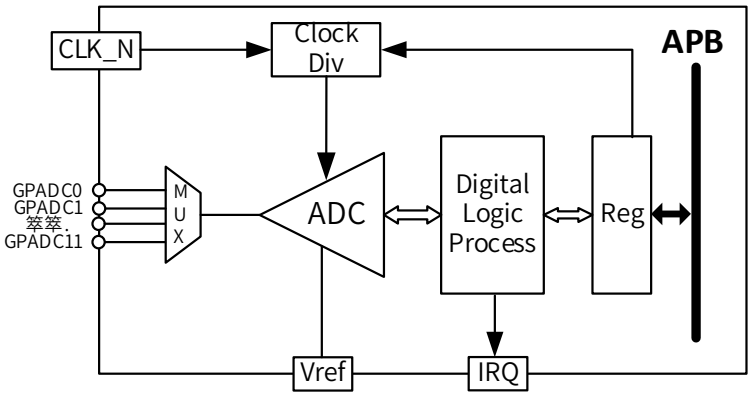
The GPADC has the following features:

- 24-ch successive approximation register (SAR) analog-to-digital converter (ADC)
  - GPADC_CTRL0: 12 channels
  - GPADC_CTRL1: 12 channels
- 64 FIFO depth of data register
- 12-bit sampling resolution and 10-bit precision
- Power reference voltage: VCC-ADC, analog input voltage range: 0 to VCC-ADC
- Maximum sampling frequency up to 1 MHz
- Supports three operation modes: single conversion mode, continuous conversion mode, burst conversion mode
- Input voltage: 0 V to 1.8 V

8.5.2 Block Diagram

The following table shows the block diagram of the GPADC_CTRL0 and the block diagram of GPADC_CTR1 is the same.

Figure 8-29 GPADC_CTRL0 Block Diagram



8.5.3 Functional Description

8.5.3.1 External Signals

The following table describes the external signals of the GPADC.

Table 8-9 GPADC External Signals

Signal Name	Description	Type
GND-ADC	Analog Ground	G
VCM-ADC	External Capacitor Connection	A I/O
VREFP-ADC	GPADC Reference Voltage (Positive)	P
VREFN-ADC	GPADC Reference Voltage (Negative)	P
VCC-ADC	Power Supply for GPADC	P
GPADC_CTRL0		
GPADC0	General Purpose ADC Input Channel 0/ BROM Boot Select in GPADC_CTRL0	AI
GPADC1	General Purpose ADC Input Channel 1 in GPADC_CTRL0	AI
GPADC2	General Purpose ADC Input Channel 2 in GPADC_CTRL0	AI
GPADC3	General Purpose ADC Input Channel 3 in GPADC_CTRL0	AI
GPADC4	General Purpose ADC Input Channel 4 in GPADC_CTRL0	AI
GPADC5	General Purpose ADC Input Channel 5 in GPADC_CTRL0	AI
GPADC6	General Purpose ADC Input Channel 6 in GPADC_CTRL0	AI
GPADC7	General Purpose ADC Input Channel 7 in GPADC_CTRL0	AI
GPADC8	General Purpose ADC Input Channel 8 in GPADC_CTRL0	AI
GPADC9	General Purpose ADC Input Channel 9 in GPADC_CTRL0	AI
GPADC10	General Purpose ADC Input Channel 10 in GPADC_CTRL0	AI
GPADC11	General Purpose ADC Input Channel 11 in GPADC_CTRL0	AI
GPADC_CTRL1		
GPADC12	General Purpose ADC Input Channel 0 in GPADC_CTRL1	AI
GPADC13	General Purpose ADC Input Channel 1 in GPADC_CTRL1	AI
GPADC14	General Purpose ADC Input Channel 2 in GPADC_CTRL1	AI
GPADC15	General Purpose ADC Input Channel 3 in GPADC_CTRL1	AI
GPADC16	General Purpose ADC Input Channel 4 in GPADC_CTRL1	AI
GPADC17	General Purpose ADC Input Channel 5 in GPADC_CTRL1	AI
GPADC18	General Purpose ADC Input Channel 6 in GPADC_CTRL1	AI
GPADC19	General Purpose ADC Input Channel 7 in GPADC_CTRL1	AI
GPADC20	General Purpose ADC Input Channel 8 in GPADC_CTRL1	AI
GPADC21	General Purpose ADC Input Channel 9 in GPADC_CTRL1	AI
GPADC22	General Purpose ADC Input Channel 10 in GPADC_CTRL1	AI
GPADC23	General Purpose ADC Input Channel 11 in GPADC_CTRL1	AI

8.5.3.2 Clock Sources

The GPADC has one clock source. The following table describes the clock source for GPADC. Users can see section 2.6 Clock Controller Unit (CCU) for clock setting, configuration, and gating information.

Table 8-10 GPADC Clock Sources

Clock Sources	Description	module
HOSC	The default frequency is 24 MHz	CCU

8.5.3.3 GPADC Work Mode

- Single conversion mode

The GPADC completes one conversion in a specified channel, the converted data is updated at the data register of the corresponding channel.

- Continuous conversion mode

The GPADC has continuous conversion in a specified channel until the software stops, the converted data is updated at the data register of the corresponding channel.

- Burst conversion mode

The GPADC samples and converts in a specified channel, and sequentially stores the results in FIFO.

8.5.3.4 Clock and Timing Requirements

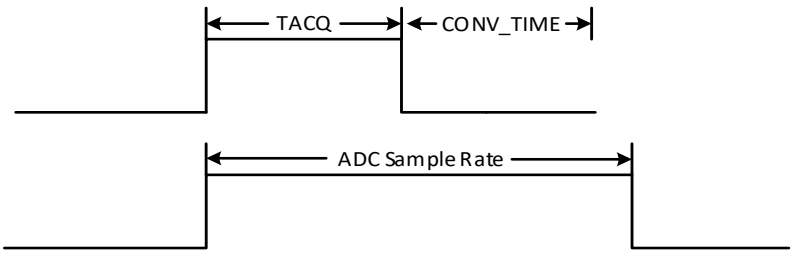
CLK_IN = 24 MHz

CONV_TIME (Conversion Time) = 1/(24MHz/13Cycles) =0.542 (us)

TACQ (ADC acquiring time) > 10RC (R is output impedance of ADC sample circuit, C = 6.4 pF)

ADC Sample Frequency > TACQ+CONV_TIME

Figure 8-30 GPADC Clock and Timing Requirement



8.5.3.5 GPADC Calculate Formula

GPADC calculate formula: GPADC_DATA = Vin/VREF*4095

Where:

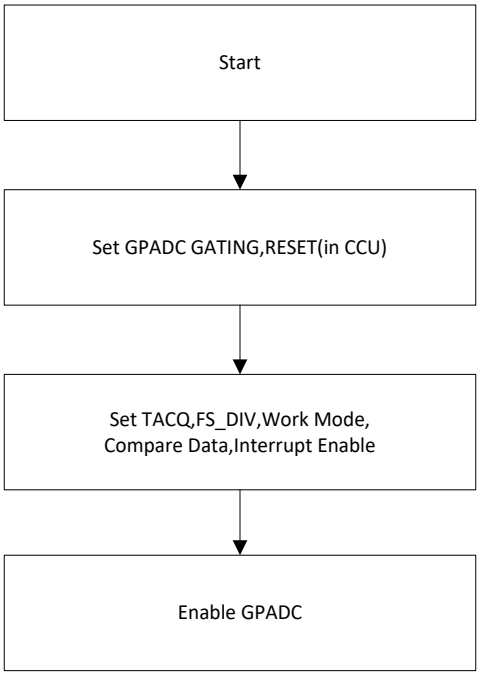
VREF = 1.8 V

8.5.4 Programming Guidelines

8.5.4.1 Initializing GPADC

The GPADC initial process is as follows.

Figure 8-31 GPADC Initial Process



Query Mode

Take channel0 for example:

Step 1 Write 0x1 to the bit [16] of GPADC_BGR_REG to dessert reset.

Step 2 Write 0x1 to the bit [0] of GPADC_BGR_REG to enable the GPADC clock.

Step 3 Write 0x2F to the bit [15:0] of GP_SR_CON to set the acquiring time of ADC.

Step 4 Write 0x1DF to the bit [31:16] of GP_SR_CON to set the ADC sample frequency divider.

Step 5 Write 0x2 to the bit [19:18] of GP_CTRL to set the continuous conversion mode.

Step 6 (Optional) If you need to configure the sampling frequency for each channel, follow these steps:

a) Configure GP_SMP_TMS (offset: 0x00C0) register to set GP_SMP_TMS value. The initial sampling frequency set in step4 is multiplied by this value to get the actual sampling frequency. For detailed configuration information, please refer to section 8.5.4.2 Configuring Sampling Frequency.

b) Configure the GP_CH_SMP_BYP bit (bit {0}) of GP_SMP_BYP (offset: 0x00D0) register as 0.

Step 7 Write 0x1 to the bit [0] of GP_CS_EN to enable the analog input channel.

Step 8 Write 0x1 to the bit [16] of GP_CTRL to enable the ADC function.

Step 9 Read the bit [0] of GP_DATA_INTS, if the bit is 1, then data conversion is complete.

Step 10 Read the bit [11:0] of GP_CH0_DATA, and calculate voltage value based on GPADC formula.

Interrupt Mode

Take channel0 for example:

Step 1 Write 0x1 to the bit [16] of GPADC_BGR_REG to dessert reset.

Step 2 Write 0x1 to the bit [0] of GPADC_BGR_REG to enable the GPADC clock.

Step 3 Write 0x2F to the bit [15:0] of GP_SR_CON to set the acquiring time of ADC.

Step 4 Write 0x1DF to the bit [31:16] of GP_SR_CON to set the ADC sample frequency divider.

Step 5 Write 0x2 to the bit [19:18] of GP_CTRL to set the continuous conversion mode.

Step 6 (Optional) If you need to configure the sampling frequency for each channel, follow these steps:

a) Configure GP_SMP_TMS (offset: 0x00C0) register to set GP_SMP_TMS value. The initial sampling frequency set in step4 is multiplied by this value to get the actual sampling frequency. For detailed configuration information, please refer to section 8.5.4.2 Configuring Sampling Frequency.

b) Configure the GP_CH_SMP_BYP bit (bit {0}) of GP_SMP_BYP (offset: 0x00D0) register as 0. Write 0x1 to the bit [0] of GP_CS_EN to enable the analog input channel.

Step 7 Write 0x1 to the bit [0] of GP_DATA_INTC to enable the GPADC data interrupt.

Step 8 Set the GIC module based on the IRQ 93.

Step 9 Put interrupt handler address into interrupt vector table based on the IRQ 93.

Step 10 Write 0x1 to the bit16 of GP_CTRL to enable the ADC function.

Step 11 Read the bit [11:0] of GP_CH0_DATA from the interrupt handler, calculate voltage value based on GPADC formula.

8.5.4.2 Configuring Sampling Frequency

The sampling frequency is only configurable in continue conversion mode. Note that the sampling frequency for each channel is only able to be configured as a multiple of 32 kHz and the total sampling frequency (sum of the sampling frequency of each channel) should be less than or equal to 1 MHz.

The sampling frequency configuration steps are as follows.

Step 1 If GPADC function is enabled, Configure the ADC_EN bit (bit [16]) of GP_CTRL (offset: 0x0004) register to disable ADC function.

Step 2 Configure the FS_DIV bit (bit [31:16]) of GP_SR_CON (offset: 0x0000) register to set the initial sampling frequency of each channel.

Step 3 Configure GP_SMP_TMS (offset: 0x00C0) register to set GP_SMP_TMS value. The initial sampling frequency is multiplied by this value to get the actual sampling frequency.

NOTE

All GPADC channels should be configured simultaneously.

The following are the all available sampling frequencies for each channel and corresponding GP_SMP_TMS values.

Table 8-11 GP_SMP_TMS Value Corresponding to Each Sampling Frequency

Sampling frequency	GP_SMP_TMS Value	Sampling frequency	GP_SMP_TMS Value
32 kHz	1	544 kHz	17
64 kHz	2	576 kHz	18
96 kHz	3	608 kHz	19
128 kHz	4	640 kHz	20
160 kHz	5	672 kHz	21
192 kHz	6	704 kHz	22
224 kHz	7	736 kHz	23
256 kHz	8	768 kHz	24
288 kHz	9	800 kHz	25
320 kHz	10	832 kHz	26
352 kHz	11	864 kHz	27
384 kHz	12	896 kHz	28
416 kHz	13	928 kHz	29
448 kHz	14	960 kHz	30

Sampling frequency	GP_SMP_TMS Value	Sampling frequency	GP_SMP_TMS Value
480 kHz	15	992 kHz	31
512 kHz	16	/	/

Step 4 Configure the GP_CH_SMP_BYP bit (bit {0}) of GP_SMP_BYP (offset: 0x00D0) register as 0.

Step 5 Configure the ADC_EN bit (bit [16]) of GP_CTRL (offset: 0x0004) register to enable ADC function.

8.5.5 Register List

Module Name	Base Address
GPADC_CTRL0	0x02009000
GPADC_CTRL1	0x02009C00

Register Name	Offset	Description
GP_SR_CON	0x0000	GPADC Sample Rate Configure Register
GP_CTRL	0x0004	GPADC Control Register
GP_CS_EN	0x0008	GPADC Compare and Select Enable Register
GP_FIFO_INTC	0x000C	GPADC FIFO Interrupt Control Register
GP_FIFO_INTS	0x0010	GPADC FIFO Interrupt Status Register
GP_FIFO_DATA	0x0014	GPADC FIFO Data Register
GP_CDATA	0x0018	GPADC Calibration Data Register
GP_DATAL_INTC	0x0020	GPADC Data Low Interrupt Configure Register
GP_DATAH_INTC	0x0024	GPADC Data High Interrupt Configure Register
GP_DATA_INTC	0x0028	GPADC Data Interrupt Configure Register
GP_DATAL_INTS	0x0030	GPADC Data Low Interrupt Status Register
GP_DATAH_INTS	0x0034	GPADC Data High Interrupt Register
GP_DATA_INTS	0x0038	GPADC Data Interrupt Status Register
GP_CH0_CMP_DATA	0x0040	GPADC CH0 Compare Data Register
GP_CH1_CMP_DATA	0x0044	GPADC CH1 Compare Data Register
GP_CH2_CMP_DATA	0x0048	GPADC CH2 Compare Data Register
GP_CH3_CMP_DATA	0x004C	GPADC CH3 Compare Data Register
GP_CH4_CMP_DATA	0x0050	GPADC CH4 Compare Data Register
GP_CH5_CMP_DATA	0x0054	GPADC CH5 Compare Data Register
GP_CH6_CMP_DATA	0x0058	GPADC CH6 Compare Data Register
GP_CH7_CMP_DATA	0x005C	GPADC CH7 Compare Data Register
GP_CH8_CMP_DATA	0x0060	GPADC CH8 Compare Data Register
GP_CH9_CMP_DATA	0x0064	GPADC CH9 Compare Data Register
GP_CH10_CMP_DATA	0x0068	GPADC CH10 Compare Data Register
GP_CH11_CMP_DATA	0x006C	GPADC CH11 Compare Data Register
GP_CH0_DATA	0x0080	GPADC CH0 Data Register
GP_CH1_DATA	0x0084	GPADC CH1 Data Register

8.6 GPIO

8.6.1 Overview

The general purpose input/output (GPIO) is one of the blocks controlling the chip multiplexing pins. The A527 supports 12 groups of GPIO pins. Each pin can be configured as input or output and these pins are used to generate input signals or output signals for special purposes.

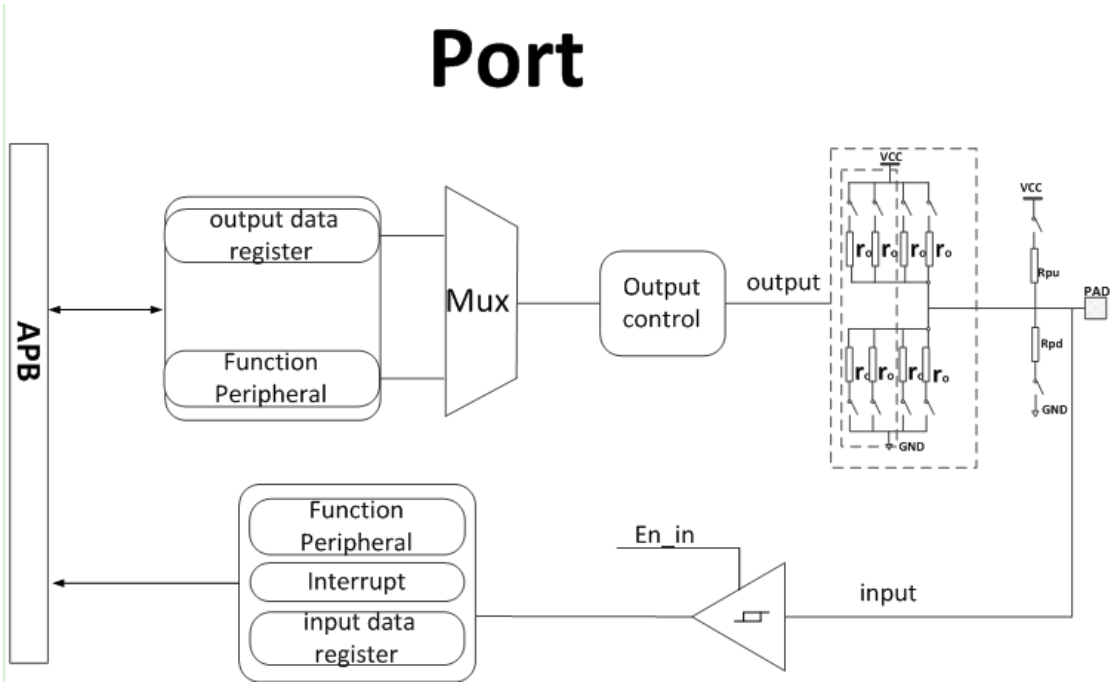
The GPIO has the following features:

- 12 groups of ports (PB, PC, PD, PE, PF, PG, PH, PI, PJ, PK, PL, PM)
- Software control for each signal pin
- Data input (capture)/output (drive)
- Each GPIO peripheral can produce an interrupt
- Pull-up/Pull-down/no-Pull register control
- Control the direction of every signal
- 4 drive strengths in each operating mode
- Up to 203 interrupts
- Configurable interrupt edges

8.6.2 Block Diagram

The following figure shows the block diagram of the GPIO.

Figure 8-32 GPIO Block Diagram



The GPIO consists of the digital part (GPIO, external interface) and IO analog part (output buffer, dual pull down, pad). The digital part can select the output interface by the MUX switch; the analog part can configure pull up/down and buffer strength.

When executing GPIO read state, the GPIO reads the current level of the pin into the internal register bus. When not executing GPIO read state, the external pin and the internal register bus are off-status, which is high-impedance.

8.6.3 Functional Description

8.6.3.1 Multi-function Port

The A527 includes 203 multi-functional input/output port pins. There are 12 ports as listed below.

Table 8-12 Multi-function Port

Port Name	Number of Pins	Input Driver	Output Driver	Multiplex Pins	Typical Power Supply
PB	15	Schmitt	CMOS	UART/TWI/ OWA/I2S/ SPI/JTAG/PWM/LCD/HDMI	3.3 V
PC	17	Schmitt	CMOS	NAND/SDC/SPI/SPIFC	3.3 V/1.8 V
PD	24	Schmitt	CMOS	LCD/PWM/LVDS/SPI/DSI/UART/PWM	3.3 V/1.8 V
PE	16	Schmitt	CMOS	MCSI/TWI/UART/PWM/I2S/ SPI/LEDC/ LCD/NCSI	3.3 V/1.8 V
PF	7	Schmitt	CMOS	SDC /JTAG/UART/I2S	3.3 V/1.8 V
PG	15	Schmitt	CMOS	SDC/UART/PCIE/I2S	3.3 V/1.8 V
PH	20	Schmitt	CMOS	TWI/UART/DMIC/CIR/ SPI/I2S/LEDC/OWA/ RGMII0/RMII0/PCIE	3.3 V
PI	17	Schmitt	CMOS	TWI/UART/SPI/IRRX/OWA/DMIC/PWM/I2S	3.3 V/1.8 V
PJ	28	Schmitt	CMOS	UART/TWI/SPI/RGMII1/RMII1	3.3 V/1.8 V
PK	24	Schmitt	CMOS	MCSI /TWI/UART/PWM/NCSI	3.3V/1.8 V
PL	14	Schmitt	CMOS	CIR/JTAG/TWI/UART/PWM /JTAG/I2S/DMIC/SPI	3.3 V/1.8 V
PM	6	Schmitt	CMOS	UARY/TWI/PWM/CIR/JTAG	3.3 V/1.8 V

8.6.3.2 GPIO Multiplex Function

Table 8-13 to Table 8-24 show the multiplex function pins of the A527.

NOTE

For each GPIO, Function0 is input function; Function1 is output function; Function7 to Function13 are reserved.

Table 8-13 PB Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PB0	I/O	UART2-TX	SPI2-CS0	JTAG-MS	LCD0-D0	PWM0-6	PB-EINT0
PB1	I/O	UART2-RX	SPI2-CLK	JTAG-CK	LCD0-D1	PWM0-7	PB-EINT1
PB2	I/O	UART2-RTS	SPI2-MOSI	JTAG-DO	LCD0-D8		PB-EINT2
PB3	I/O	UART2-CTS	SPI2-MISO	JTAG-DI	LCD0-D9		PB-EINT3
PB4	I/O	TWI1-SCK	I2S0-MCLK		PWM0-8	HDMI-SCL	PB-EINT4
PB5	I/O	TWI1-SDA	I2S0-BCLK		PWM0-9	HDMI-SDA	PB-EINT5
PB6	I/O		I2S0-LRCK		PWM0-10	HDMI-CEC	PB-EINT6
PB7	I/O	OWA-IN	I2S0-DOUT0	I2S0-DIN1	LCD0-D16	PWM0-11	PB-EINT7
PB8	I/O	OWA-OUT	I2S0-DIN0	I2S0-DOUT1	LCD0-D17	PWM0-0	PB-EINT8
PB9	I/O	UART0-TX	TWI0-SCK		I2S0-DIN2	I2S0-DOUT2	PB-EINT9
PB10	I/O	UART0-RX	TWI0-SDA	PWM0-1	I2S0-DIN3	I2S0-DOUT3	PB-EINT10
PB11	I/O	TWI5-SCK	UART7-RTS	SPI1-CS0	PWM0-2		PB-EINT11
PB12	I/O	TWI5-SDA	UART7-CTS	SPI1-CLK	PWM0-3		PB-EINT12
PB13	I/O	TWI4-SCK	UART7-TX	SPI1-MOSI	PWM0-4		PB-EINT13
PB14	I/O	TWI4-SDA	UART7-RX	SPI1-MISO	PWM0-5		PB-EINT14

(1) I2S0 signals and S-I2S0 signals cannot be connected simultaneously.

Table 8-14 PC Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PC0	I/O	NAND-WE	SDC2-DS				PC-EINT0
PC1	I/O	NAND-ALE	SDC2-RST				PC-EINT1
PC2	I/O	NAND-CLE		SPI0-MOSI	SPIF-MOSI		PC-EINT2
PC3	I/O	NAND-CE1		SPI0-CS0	SPIF-CS0		PC-EINT3
PC4	I/O	NAND-CE0		SPI0-MISO	SPIF-MISO		PC-EINT4
PC5	I/O	NAND-RE	SDC2-CLK				PC-EINT5
PC6	I/O	NAND-RB0	SDC2-CMD				PC-EINT6
PC7	I/O	NAND-RB1		SPI0-CS1	SPIF-DQS		PC-EINT7
PC8	I/O	NAND-DQ7	SDC2-D3		SPIF-D7		PC-EINT8
PC9	I/O	NAND-DQ6	SDC2-D4		SPIF-D6		PC-EINT9
PC10	I/O	NAND-DQ5	SDC2-D0		SPIF-D5		PC-EINT10
PC11	I/O	NAND-DQ4	SDC2-D5		SPIF-D4		PC-EINT11
PC12	I/O	NAND-DQS		SPI0-CLK	SPIF-CLK		PC-EINT12
PC13	I/O	NAND-DQ3	SDC2-D1				PC-EINT13
PC14	I/O	NAND-DQ2	SDC2-D6				PC-EINT14
PC15	I/O	NAND-DQ1	SDC2-D2	SPI0-WP	SPIF-WP		PC-EINT15
PC16	I/O	NAND-DQ0	SDC2-D7	SPI0-HOLD	SPIF-HOLD		PC-EINT16

Table 8-15 PD Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PD0	I/O	LCD0-D2	LVDS0-D0P	DSI0-D0P	PWM0-0		PD-EINT0
PD1	I/O	LCD0-D3	LVDS0-D0N	DSI0-D0N	PWM0-1		PD-EINT1
PD2	I/O	LCD0-D4	LVDS0-D1P	DSI0-D1P	PWM0-2		PD-EINT2
PD3	I/O	LCD0-D5	LVDS0-D1N	DSI0-D1N	PWM0-3		PD-EINT3
PD4	I/O	LCD0-D6	LVDS0-D2P	DSI0-CKP	PWM0-4		PD-EINT4
PD5	I/O	LCD0-D7	LVDS0-D2N	DSI0-CKN	PWM0-5		PD-EINT5
PD6	I/O	LCD0-D10	LVDS0-CKP	DSI0-D2P	PWM0-6		PD-EINT6
PD7	I/O	LCD0-D11	LVDS0-CKN	DSI0-D2N	PWM0-7		PD-EINT7
PD8	I/O	LCD0-D12	LVDS0-D3P	DSI0-D3P	PWM0-8		PD-EINT8
PD9	I/O	LCD0-D13	LVDS0-D3N	DSI0-D3N	PWM0-9		PD-EINT9
PD10	I/O	LCD0-D14	LVDS1-D0P	DSI1-D0P	PWM0-10	SPI1-CS0/DBI-CSX	PD-EINT10
PD11	I/O	LCD0-D15	LVDS1-D0N	DSI1-D0N	PWM0-11	SPI1-CLK/DBI-SCLK	PD-EINT11
PD12	I/O	LCD0-D18	LVDS1-D1P	DSI1-D1P	PWM0-12	SPI1-MOSI/DBI-SDO	PD-EINT12

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PD13	I/O	LCD0-D19	LVDS1-D1N	DSI1-D1N	PWM0-13	SPI1-MISO/DBI-SDI/DBI-TE/DBI-DCX	PD-EINT13
PD14	I/O	LCD0-D20	LVDS1-D2P	DSI1-CKP	PWM0-14	UART3-TX	PD-EINT14
PD15	I/O	LCD0-D21	LVDS1-D2N	DSI1-CKN	PWM0-15	UART3-RX	PD-EINT15
PD16	I/O	LCD0-D22	LVDS1-CKP	DSI1-D2P	PWM1-0	UART3-RTS	PD-EINT16
PD17	I/O	LCD0-D23	LVDS1-CKN	DSI1-D2N	PWM1-1	UART3-CTS	PD-EINT17
PD18	I/O	LCD0-CLK	LVDS1-D3P	DSI1-D3P	PWM1-2	UART4-TX	PD-EINT18
PD19	I/O	LCD0-DE	LVDS1-D3N	DSI1-D3N	PWM1-3	UART4-RX	PD-EINT19
PD20	I/O	LCD0-HSYNC	PWM0-2	UART2-TX	UART7-RTS	UART4-RTS	PD-EINT20
PD21	I/O	LCD0-VSYNC	PWM0-3	UART2-RX	UART7-CTS	UART4-CTS	PD-EINT21
PD22	I/O	PWM0-1	SPI1-HOLD/DBI-DCX/DBI-WRX	UART2-RTS	UART7-TX	TWI0-SCK	PD-EINT22
PD23	I/O	PWM0-0	SPI1-WP/DBI-TE	UART2-CTS	UART7-RX	TWI0-SDA	PD-EINT23

Table 8-16 PE Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PE0	I/O	MCSI0-MCLK					PE-EINT0
PE1	I/O	TWI2-SCK	UART4-TX				PE-EINT1
PE2	I/O	TWI2-SDA	UART4-RX				PE-EINT2
PE3	I/O	TWI3-SCK	UART4-RTS				PE-EINT3
PE4	I/O	TWI3-SDA	UART4-CTS				PE-EINT4
PE5	I/O	MCSI1-MCLK		I2S2-MCLK(1)	LEDC		PE-EINT5
PE6	I/O			I2S2-BCLK(1)	LCD0-TRIG	NCSI-D8	PE-EINT6
PE7	I/O			I2S2-LRCK(1)	LCD1-TRIG	NCSI-D9	PE-EINT7
PE8	I/O			I2S2-DOUT0(1)	LCD2-TRIG	NCSI-D10	PE-EINT8
PE9	I/O			I2S2-DIN0(1)		NCSI-D11	PE-EINT9
PE10	I/O	MCSI3-MCLK	PWM0-3			NCSI-D12	PE-EINT10
PE11	I/O	TWI1-SCK	UART5-RTS	SPI2-CS0	UART6-TX	NCSI-D13	PE-EINT11
PE12	I/O	TWI1-SDA	UART5-CTS	SPI2-CLK	UART6-RX	NCSI-D14	PE-EINT12
PE13	I/O	TWI4-SCK	UART5-TX	SPI2-MOSI	UART6-RTS	CSI0-XVS-FSYNC	PE-EINT13
PE14	I/O	TWI4-SDA	UART5-RX	SPI2-MISO	UART6-CTS	CSI1-XVS-FSYNC	PE-EINT14
PE15	I/O	MCSI2-MCLK	PWM0-2			NCSI-D15	PE-EINT15

(1) If I2S2 needs to be used, please ensure that the peripheral device connected to I2S2 signals will not be used with HDMI/eDP interface simultaneously. If you have more questions, please contact Allwinner FAE.

Table 8-17 PF Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PF0	I/O	SDC0-D1	JTAG-MS		I2S3-DIN0	I2S3-DOUT1	PF-EINT0
PF1	I/O	SDC0-D0	JTAG-DI		I2S3-DOUT0	I2S3-DIN1	PF-EINT1
PF2	I/O	SDC0-CLK	UART0-TX		I2S3-DIN2	I2S3-DOUT2	PF-EINT2
PF3	I/O	SDC0-CMD	JTAG-DO		I2S3-LRCK		PF-EINT3
PF4	I/O	SDC0-D3	UART0-RX		I2S3-DIN3	I2S3-DOUT3	PF-EINT4
PF5	I/O	SDC0-D2	JTAG-CK		I2S3-BCLK		PF-EINT5
PF6	I/O				I2S3-MCLK		PF-EINT6

Table 8-18 PG Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PG0	I/O	SDC1-CLK					PG-EINT0
PG1	I/O	SDC1-CMD					PG-EINT1
PG2	I/O	SDC1-D0	PCIE0-PERSTN				PG-EINT2
PG3	I/O	SDC1-D1	PCIE0-WAKEN				PG-EINT3
PG4	I/O	SDC1-D2	PCIE0-CLKREQN				PG-EINT4
PG5	I/O	SDC1-D3					PG-EINT5
PG6	I/O	UART1-TX					PG-EINT6
PG7	I/O	UART1-RX					PG-EINT7
PG8	I/O	UART1-RTS					PG-EINT8
PG9	I/O	UART1-CTS					PG-EINT9
PG10	I/O		I2S1-MCLK				PG-EINT10
PG11	I/O		I2S1-BCLK				PG-EINT11
PG12	I/O		I2S1-LRCK				PG-EINT12
PG13	I/O		I2S1-DOUT0	I2S1-DIN1			PG-EINT13
PG14	I/O		I2S1-DIN0	I2S1-DOUT1			PG-EINT14

Table 8-19 PH Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PH0	I/O	TWI0-SCK			RGMII0-RXD1/RMII0-RXD1		PH-EINT0
PH1	I/O	TWI0-SDA			RGMII0-RXD0/RMII0-RXD0		PH-EINT1
PH2	I/O	TWI1-SCK		I2S2-DIN3(1)	RGMII0-RXCTL/RMII0-CRS-DV	I2S2-DOUT3(1)	PH-EINT2
PH3	I/O	TWI1-SDA	IR-TX	I2S2-DIN2(1)	RGMII0-CLKIN/RMII0-RXER	I2S2-DOUT2(1)	PH-EINT3
PH4	I/O	UART3-TX	SPI1-CS0		RGMII0-TXD1/RMII0-TXD1		PH-EINT4
PH5	I/O	UART3-RX	SPI1-CLK	LEDC	RGMII0-TXD0/RMII0-TXD0		PH-EINT5
PH6	I/O	UART3-RTS	SPI1-MOSI	OWA-IN	RGMII0-TXCK/RMII0-TXCK		PH-EINT6
PH7	I/O	UART3-CTS	SPI1-MISO	OWA-OUT	RGMII0-TXCTL/RMII0-TXEN		PH-EINT7
PH8	I/O	DMIC-CLK(2)	SPI2-CS0	I2S2-MCLK(1)	I2S2-DIN2(1)		PH-EINT8
PH9	I/O	DMIC-DATA0(2)	SPI2-CLK	I2S2-BCLK(1)	RGMII0-MDC		PH-EINT9
PH10	I/O	DMIC-DATA1(2)	SPI2-MOSI	I2S2-LRCK(1)	RGMII0-MDIO		PH-EINT10
PH11	I/O	DMIC-DATA2(2)	SPI2-MISO	I2S2-DOUT0(1)	I2S2-DIN1(1)	PCIE0-PERSTN	PH-EINT11
PH12	I/O	DMIC-DATA3(2)	TWI3-SCK	I2S2-DIN0(1)	I2S2-DOUT1(1)	PCIE0-WAKEN	PH-EINT12
PH13	I/O		TWI3-SDA	I2S3-MCLK	RGMII0-EPHY-25M		PH-EINT13
PH14	I/O			I2S3-BCLK	RGMII0-RXD3/RMII0-NULL		PH-EINT14
PH15	I/O			I2S3-LRCK	RGMII0-RXD2/RMII0-NULL		PH-EINT15
PH16	I/O		I2S3-DOUT0	I2S3-DIN1	RGMII0-RXCK/RMII0-NULL	CLK-FANOUT0	PH-EINT16
PH17	I/O		I2S3-DOUT1	I2S3-DIN0	RGMII0-TXD3/RMII0-NULL		PH-EINT17
PH18	I/O	IR-TX	I2S3-DOUT2	I2S3-DIN2	RGMII0-TXD2/RMII0-NULL		PH-EINT18
PH19	I/O	IR-RX	I2S3-DOUT3	I2S3-DIN3	LEDC	PCIE0-CLKREQN	PH-EINT19

(1) If I2S2 needs to be used, please ensure that the peripheral device connected to I2S2 signals will not be used with HDMI/eDP interface simultaneously. If you have more questions, please contact Allwinner FAE.

(2) DMIC signals and S-DMIC signals cannot be connected simultaneously.

Table 8-20 PI Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PI0	I/O	TWI4-SCK	UART4-TX	PWM0-1	I2S2-DIN3(1)	I2S2-DOUT3(1)	PI-EINT0
PI1	I/O	TWI4-SDA	UART4-RX	PWM0-2	I2S2-DIN2(1)	I2S2-DOUT2(1)	PI-EINT1
PI2	I/O	UART5-TX	SPI1-CS0	PWM0-3	I2S2-BCLK(1)		PI-EINT2
PI3	I/O	UART5-RX	SPI1-CLK	PWM0-4	I2S2-LRCK(1)		PI-EINT3
PI4	I/O	UART5-RTS	SPI1-MOSI	PWM0-5	I2S2-DOUT0(1)	I2S2-DIN1(1)	PI-EINT4
PI5	I/O	UART5-CTS	SPI1-MISO	PWM0-6	I2S2-DIN0(1)	I2S2-DOUT1(1)	PI-EINT5
PI6	I/O	UART6-TX	UART4-RTS	PWM0-7	SPI2-CS0		PI-EINT6
PI7	I/O	UART6-RX	UART4-CTS	PWM0-8	SPI2-CLK		PI-EINT7
PI8	I/O	TWI5-SCK	IR-RX	PWM0-9	SPI2-MOSI		PI-EINT8
PI9	I/O	TWI5-SDA	DMIC-DATA2(2)	PWM0-10			PI-EINT9
PI10	I/O	OWA-OUT	DMIC-DATA1(2)	PWM0-11	I2S2-MCLK(1)		PI-EINT10
PI11	I/O	UART3-TX	DMIC-DATA0(2)	PWM0-12			PI-EINT11
PI12	I/O	UART3-RX		PWM0-13	SPI2-MISO		PI-EINT12
PI13	I/O	UART6-CTS	DMIC-DATA3(2)	PWM0-14	I2S2-MCLK(1)		PI-EINT13
PI14	I/O	UART6-RTS	DMIC-CLK(2)	PWM0-15			PI-EINT14
PI15	I/O	UART3-RTS	TWI2-SCK	PWM1-0			PI-EINT15
PI16	I/O	UART3-CTS	TWI2-SDA	PWM1-1			PI-EINT16

(1) If I2S2 needs to be used, please ensure that the peripheral device connected to I2S2 signals will not be used with HDMI/eDP interface simultaneously. If you have more questions, please contact Allwinner FAE.

(2) DMIC signals and S-DMIC signals cannot be connected simultaneously.

Table 8-21 PJ Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PJ0	I/O	LCD1-D0	LVDS2-D0P		RGMII1-RXD1/RMII1-RXD1		PJ-EINT0
PJ1	I/O	LCD1-D1	LVDS2-D0N		RGMII1-RXD0/RMII1-RXD0		PJ-EINT1
PJ2	I/O	LCD1-D2	LVDS2-D1P		RGMII1-RXCTL/RMII1-CRS-DV		PJ-EINT2
PJ3	I/O	LCD1-D3	LVDS2-D1N		RGMII1-CLKIN/RMII1-RXER		PJ-EINT3
PJ4	I/O	LCD1-D4	LVDS2-D2P		RGMII1-TXD1/RMII1-TXD1		PJ-EINT4
PJ5	I/O	LCD1-D5	LVDS2-D2N		RGMII1-TXD0/RMII1-TXD0		PJ-EINT5
PJ6	I/O	LCD1-D6	LVDS2-CKP		RGMII1-TXCK/RMII1-TXCK		PJ-EINT6
PJ7	I/O	LCD1-D7	LVDS2-CKN		RGMII1-TXCTL/RMII1-TXEN		PJ-EINT7
PJ8	I/O	LCD1-D8	LVDS2-D3P		RGMII1-MDC		PJ-EINT8
PJ9	I/O	LCD1-D9	LVDS2-D3N		RGMII1-MDIO		PJ-EINT9
PJ10	I/O	LCD1-D10	LVDS3-D0P		RGMII1-EPHY-25M		PJ-EINT10

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PJ11	I/O	LCD1-D11	LVDS3-D0N		RGMII1-RXD3/RMII1-NULL		PJ-EINT11
PJ12	I/O	LCD1-D12	LVDS3-D1P		RGMII1-RXD2/RMII1-NULL		PJ-EINT12
PJ13	I/O	LCD1-D13	LVDS3-D1N		RGMII1-RXCK/RMII1-NULL		PJ-EINT13
PJ14	I/O	LCD1-D14	LVDS3-D2P		RGMII1-TXD3/RMII1-NULL		PJ-EINT14
PJ15	I/O	LCD1-D15	LVDS3-D2N		RGMII1-TXD2/RMII1-NULL		PJ-EINT15
PJ16	I/O	LCD1-D16	LVDS3-CKP				PJ-EINT16
PJ17	I/O	LCD1-D17	LVDS3-CKN				PJ-EINT17
PJ18	I/O	LCD1-D18	LVDS3-D3P				PJ-EINT18
PJ19	I/O	LCD1-D19	LVDS3-D3N				PJ-EINT19
PJ20	I/O	LCD1-D20	UART2-TX	UART3-RTS	SPI0-CS0		PJ-EINT20
PJ21	I/O	LCD1-D21	UART2-RX	UART3-CTS	SPI0-CLK		PJ-EINT21
PJ22	I/O	LCD1-D22	UART2-RTS	UART3-TX	SPI0-MOSI		PJ-EINT22
PJ23	I/O	LCD1-D23	UART2-CTS	UART3-RX	SPI0-MISO		PJ-EINT23
PJ24	I/O	LCD1-CLK	TWI4-SCK	UART4-TX	SPI0-CS1		PJ-EINT24
PJ25	I/O	LCD1-DE	TWI4-SDA	UART4-RX	SPI0-WP		PJ-EINT25
PJ26	I/O	LCD1-HSYNC	TWI5-SCK	UART4-RTS	SPI0-HOLD		PJ-EINT26
PJ27	I/O	LCD1-VSYNC	TWI5-SDA	UART4-CTS			PJ-EINT27

Table 8-22 PK Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PK0	I/O	MCSIA-D0N					PK-EINT0
PK1	I/O	MCSIA-D0P					PK-EINT1
PK2	I/O	MCSIA-D1N					PK-EINT2
PK3	I/O	MCSIA-D1P					PK-EINT3
PK4	I/O	MCSIA-CKN	TWI2-SCK				PK-EINT4
PK5	I/O	MCSIA-CKP	TWI2-SDA				PK-EINT5
PK6	I/O	MCSIB-D0N					PK-EINT6
PK7	I/O	MCSIB-D0P					PK-EINT7
PK8	I/O	MCSIB-D1N					PK-EINT8
PK9	I/O	MCSIB-D1P					PK-EINT9
PK10	I/O	MCSIB-CKN	TWI3-SCK				PK-EINT10
PK11	I/O	MCSIB-CKP	TWI3-SDA				PK-EINT11
PK12	I/O	MCSIC-D0N	UART7-TX	TWI4-SCK	NCSI-PCLK		PK-EINT12
PK13	I/O	MCSIC-D0P	UART7-RX	TWI4-SDA	NCSI-MCLK		PK-EINT13
PK14	I/O	MCSIC-D1N	UART7-RTS	UART5-RTS	NCSI-HSYNC		PK-EINT14
PK15	I/O	MCSIC-D1P	UART7-CTS	UART5-CTS	NCSI-VSYNC		PK-EINT15
PK16	I/O	MCSIC-CKN	TWI5-SCK	UART5-TX	NCSI-D0		PK-EINT16
PK17	I/O	MCSIC-CKP	TWI5-SDA	UART5-RX	NCSI-D1		PK-EINT17
PK18	I/O	MCSID-D0N	MCSI0-MCLK	UART6-TX	NCSI-D2		PK-EINT18
PK19	I/O	MCSID-D0P	TWI2-SCK	UART6-RX	NCSI-D3		PK-EINT19
PK20	I/O	MCSID-D1N	TWI2-SDA	UART6-RTS	NCSI-D4		PK-EINT20
PK21	I/O	MCSID-D1P	MCSI1-MCLK	UART6-CTS	NCSI-D5		PK-EINT21
PK22	I/O	MCSID-CKN	TWI3-SCK	PWM0-6	NCSI-D6		PK-EINT22
PK23	I/O	MCSID-CKP	TWI3-SDA	PWM0-7	NCSI-D7		PK-EINT23

Table 8-23 PL Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PL0	I/O	S-TWI0-SCK					PL-EINT0
PL1	I/O	S-TWI0-SDA					PL-EINT1
PL2	I/O	S-UART0-TX	S-UART1-TX	MCU-PWM0-0			PL-EINT2
PL3	I/O	S-UART0-RX	S-UART1-RX	MCU-PWM0-1			PL-EINT3
PL4	I/O	S-JTAG-MS		MCU-PWM0-2	S-I2S0-BCLK(1)		PL-EINT4
PL5	I/O	S-JTAG-CK		MCU-PWM0-3	S-I2S0-LRCK(1)	S-DMIC-DATA3(2)	PL-EINT5
PL6	I/O	S-JTAG-DO	MCU-PWM0-4	S-I2S0-DIN1(1)	S-I2S0-DOUT0(1)	S-DMIC-DATA2(2)	PL-EINT6
PL7	I/O	S-JTAG-DI	MCU-PWM0-5	S-I2S0-DOUT1(1)	S-I2S0-DIN0(1)	S-DMIC-DATA1(2)	PL-EINT7
PL8	I/O	S-TWI1-SCK		S-RJTAG-MS	S-I2S0-MCLK(1)	S-DMIC-DATA0(2)	PL-EINT8
PL9	I/O	S-TWI1-SDA		S-RJTAG-CK	S-PWM0-1	S-DMIC-CLK	PL-EINT9
PL10	I/O	S-PWM0-0		S-RJTAG-DO	S-DMIC-DATA0(2)	S-SPI0-CS0	PL-EINT10
PL11	I/O	S-IR-RX		S-RJTAG-DI	S-DMIC-DATA1(2)	S-SPI0-CLK	PL-EINT11
PL12	I/O	S-TWI2-SCK	MCU-PWM0-6	S-UART0-TX	S-DMIC-DATA2(2)	S-SPI0-MOSI	PL-EINT12
PL13	I/O	S-TWI2-SDA	MCU-PWM0-7	S-UART0-RX	S-DMIC-DATA3(2)	S-SPI0-MISO	PL-EINT13

(1) I2S0 signals and S-I2S0 signals cannot be connected simultaneously.

(2) DMIC signals and S-DMIC signals cannot be connected simultaneously.

Table 8-24 PM Multiplex Function

Pin Name	IO Type	Function2	Function3	Function4	Function5	Function6	Function14
PM0	I/O	S-UART0-TX	S-UART1-TX	MCU-PWM0-0			PM-EINT0
PM1	I/O	S-UART0-RX	S-UART1-RX	MCU-PWM0-1			PM-EINT1
PM2	I/O	S-TWI1-SCK	S-RJTAG-MS	MCU-PWM0-4			PM-EINT2
PM3	I/O	S-TWI1-SDA	S-RJTAG-CK	MCU-PWM0-5			PM-EINT3
PM4	I/O	MCU-PWM0-6	S-RJTAG-DO	S-TWI2-SCK			PM-EINT4
PM5	I/O	S-IR-RX	S-RJTAG-DI	S-TWI2-SDA	MCU-PWM0-7		PM-EINT5

8.6.3.3 Port Function

The Port Controller supports 120 GPIOs, every GPIO can configure as Input, Output, Function Peripheral, IO disable or Interrupt function. The configuration instruction of every function is as follows.

Table 8-25 Port Function

	Function	Buffer Strength	Pull Up	Pull Down
Input	GPIO/Multiplexing Input	/	X	X
Output	GPIO/Multiplexing Output	Y	X	X
Disable	Pull Up	/	Y	N
	Pull Down	/	N	Y
Interrupt	Trigger	/	X	X

/: non-configure, configuration is invalid

Y: configure

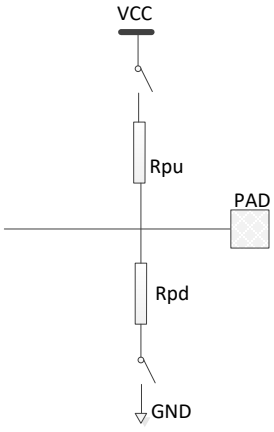
X: Select configuration according to the actual situation

N: Forbid to configure

8.6.3.4 Pull Up/Down and High-Impedance Logic

Each IO pin can configure the internal pull-up/down function or high-impedance.

Figure 8-33 Pull up/down Logic



High-impedance, the output is float state, all buffer is off, the level is decided by external high/low level. When high-impedance, the software configures the switch on Rpu and Rpd as off, and the multiplexing function of IO is set as IO disable or input by software.

Pull-up, an uncertain signal is pulled high by resistance, the resistance has a current-limiting function. When pulling up, the switch on Rpu is conducted by software configuration, the IO is pulled up to VCC by Rpu.

Pull-down, an uncertain signal is pulled low by a resistance. When pulling down, the switch on Rpd is conducted by software configuration, the IO is pulled down to GND by Rpd.

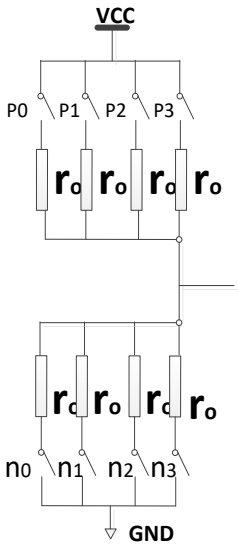
The pull-up/down of each IO is weak pull-up/down.

The setting of pull-down, pull-up, high-impedance is decided by the external circuit.

8.6.3.5 Buffer Strength

Each IO can be set as different buffer strength. The IO buffer diagram is as follows.

Figure 8-34 IO Buffer Strength Diagram



When output high level, the n0, n1, n2, n3 of NMOS is off, the p0, p1, p2, p3 of PMOS is on. When the buffer strength is set to 0 (buffer strength is weakest), only the p0 is on, the output impedance is maximum, the impedance value is r0. When the buffer strength is set to 1, only the p0 and p1 is on, the output impedance is equivalent to two r0 in parallel, the impedance value is r0/2. When the buffer strength is 2, only the p0, p1, and p2 is on, the output impedance is equivalent to three r0 in parallel, the impedance value is r0/3. When buffer strength is 3, the p0, p1, p2, and p3 is on, the output impedance is equivalent to four r0 in parallel, the impedance value is r0/4.

When output low level, the p0, p1, p2, p3 of PMOS is off, the n0, n1, n2, n3 of NMOS is on. When the buffer strength is set to 0 (buffer strength is weakest), only the n0 is on, the output impedance is maximum, the impedance value is r0. When the buffer strength is set to 1, only the n0 and n1 is on, the output impedance is equivalent to two r0 in parallel, the impedance value is r0/2. When the buffer strength is 2, only the n0, n1, and n2 is on, the output impedance is equivalent to three r0 in parallel, the impedance value is r0/3. When the buffer strength is 3, the n0, n1, n2, and n3 is on, the output impedance is equivalent to four r0 in parallel, the impedance value is r0/4.

When GPIO is set to input or interrupt function, between the output driver circuit and the port is unconnected, the driver configuration is invalid.

8.6.3.6 Interrupt

Each group IO has an independent interrupt number. The IO within-group uses one interrupt number when one IO generates interrupt, the GPIO pins sent interrupt request to GIC. External Interrupt Status Register is used to query which IO generates interrupt.

The interrupt trigger of GPIO supports the following trigger types.

- Positive Edge: When a low level changes to a high level, the interrupt will generate. No matter how long a high level keeps, the interrupt generates only once.
- Negative Edge: When a high level changes to a low level, the interrupt will generate. No matter how long a low level keeps, the interrupt generates only once.
- High Level: Just keep a high level and the interrupt will always generate.
- Low Level: Just keep a low level and the interrupt will always generate.
- Double Edge: Positive and negative edge.

External Interrupt Configure Register is used to configure the trigger type.

The GPIO interrupt supports hardware debounce function by setting External Interrupt Debounce Register. Sample trigger signal using a lower sample clock, to reach the debounce effect because the dither frequency of the signal is higher than the sample frequency.

Set the sample clock source by PIO_INT_CLK_SELECT and the prescale factor by DEB_CLK_PRE_SCALE.

8.6.4 Programming Guidelines

8.6.4.1 Disable

The steps to disable I/O pins are as follows:

Step 1 Write FFFF to the Px_SELECT bit of the Px_CFG register to disable the I/O pins.

Step 2 If it is needed to control whether the I/O pins are pulled up or pulled down, configure Px_PULL register.

- Write 2'b01 to the Px_PULL bit of Px_PULL register to pull up the I/O pins. In this case, the default status of I/O pins is logic-high.
- Write 2'b10 to the Px_PULL bit of Px_PULL register to pull down the I/O pins. In this case, the default status of I/O pins is logic-low.

8.6.4.2 Input

The steps to configure I/O pins as inputs are as follows:

Step 1 Write 0000 to the Px_SELECT bit of the Px_CFG register to enable input function.

Step 2 If it is needed to control whether the I/O pins are pulled up or pulled down, configure Px_PULL register.

- Write 2'b01 to the Px_PULL bit of Px_PULL register to pull up the I/O pins.
- Write 2'b10 to the Px_PULL bit of Px_PULL register to pull down the I/O pins.

Step 3 Configure the Px_DAT bit of the Px_DAT register to read the pin status.

- If the external input is driven, the value of the Px_DAT bit is the external input value.
- If the external input is not driven, the value of the Px_DAT bit is 1 when the I/O pins are pulled up and is 0 when pins are pulled down.

8.6.4.3 Output

The steps to configure I/O pins as outputs are as follows:

Step 1 Write 0001 to the Px_SELECT bit of the Px_CFG register to enable output function.

Step 2 If it is needed to set the buffer strength of the I/O pins, configure the Px_DRV bit of the Px_DRV register.

- When the Px_DRV bit is configured as 00, the buffer strength is the weakest.
- When the Px_DRV bit is configured as 11, the buffer strength is the strongest.
- The default value of the Px_DRV register is 01.

Step 3 If it is needed to control whether the I/O pins are pulled up or pulled down, configure Px_PULL register.

- Write 2'b01 to the Px_PULL bit of Px_PULL register to pull up the I/O pins.
- Write 2'b10 to the Px_PULL bit of Px_PULL register to pull down the I/O pins.

Step 4 Configure the Px_DAT bit of the Px_DAT register to output 1 or 0 to the I/O pins.

- If output function is enabled, the pin status is the same as the corresponding bit.
- If output function is disabled, the value of the Px_DAT bit is 1 when the I/O pins are pulled up and is 0 when pins are pulled down.

8.6.4.4 Interrupt

The steps to configure I/O pins as interrupt pins are as follows:

Step 1 Write 1110 to the Px_SELECT bit of the Px_CFG register to enable interrupt function.

Step 2 If it is needed to control whether the I/O pins are pulled up or pulled down, configure Px_PULL register.

- Write 2'b01 to the Px_PULL bit of Px_PULL register to pull up the I/O pins. in this case, if external pins are not driven, the input level is high by default.

- Write 2'b10 to the Px_PULL bit of Px_PULL register to pull down the I/O pins. In this case, if external pins are not driven, the input level is low by default.

Step 3 Configure the EINTx_CFG bit of Px_INT_CFG register to set the interrupt generation mode.

Step 4 Configure Px_INT_DEB register to set debounce parameters including sample clock source and prescale factor.

Step 5 Write 1 to the EINTx_STATUS bit of the Px_INT_STA register to clear IRQ pending.

Step 6 Write 1 to the EINTx_CTL bit of the Px_INT_CTL register to enable interrupt.

Step 7 After an interrupt is processed, repeat Step5 to clear IRQ pending and wait for next interrupt operation.

Step 8 Write 0 to the EINTx_CTL bit of the Px_INT_CTL register to disable interrupt function.

8.6.4.5 Multi Function

The steps to configure I/O pins as for function multiplexing are as follows:

Step 1 Configure the Px_SELECT bit of the Px_CFG register to select the function needed.

Step 2 Configure the Px_DRV bit of the Px_DRV register to set buffer strength depending on the characteristic of the selected function.

Step 3 If it is needed to control whether the I/O pins are pulled up or pulled down, configure Px_PULL register.

- Write 2'b01 to the Px_PULL bit of Px_PULL register to pull up the I/O pins.
- Write 2'b10 to the Px_PULL bit of Px_PULL register to pull down the I/O pins.
- If external pins are driven, the pin status is the same as the corresponding bit.
- If external pins are not driven, the value of the Px_DAT bit is 1 when the I/O pins are pulled up and 0 when pins are pulled down.

8.6.4.6 Power Configuration

Configuring Group Power Voltage

A527 only supports to set group power voltage in PF ports, which is able to be set as 3.3 V or 1.8V by configuring GPIO_POW_VAL_SET_CTL register. The group power voltage in other ports is un configurable and depended on the peripheral circuit.

Configuring Group Withstand Voltage

Configuring group withstand voltage intends to ensure that the internal withstand circuit voltage is consistent with group voltage. There are two modes, adaptive mode and manual mode, for users. In adaptive mode, the internal withstand circuit will adjust group withstand mode

automatically depending on the detected GPIO voltage. The default mode is manual mode and it is recommended to choose manual mode in A527.

The following steps describe how to configure group withstand voltage.

- **Adaptive Mode**

The following table shows the configuration for registers in adaptive mode.

Bit	Register
Px_PWR_MOD_SEL bit = 0	GPIO_POW_MOD_SEL
VCC_Px_WS_VOL_MOD_SEL bit = 0	GPIO_POW_MS_CTL

- **Manual Mode (Recommended)**

Step 1 Before using GPIO, read GPIO_POW_VAL register to obtain current group voltage.

- If the current power supply is 1.8V, continue from Step2.
- If the current power supply is 3.3V, continue from Step4.

Step 2 Read the PB_PWR_VAL bit (bit [1]) of GPIO_POW_VAL register to obtain current group voltage.

Step 3 Configure withstand voltage.

The following table shows the corresponding withstand voltage for each group voltage and corresponding configuration register.

Group Voltage	Withstand Voltage	Bit	Register
1.8V	1.8V	Px_PWR_MOD_SEL bit = 0	GPIO_POW_MOD_SEL
		VCC_Px_WS_VOL_MOD_SEL bit = 1	GPIO_POW_MS_CTL
2.5V	3.3V	Px_PWR_MOD_SEL bit = 1	GPIO_POW_MOD_SEL
3.3V	3.3V	Px_PWR_MOD_SEL bit = 1	GPIO_POW_MOD_SEL

Step 4 After the I/O pins are power-on, repeat Step2 and Step3 to configure withstand voltage according to the actual group voltage.

Step 5 When adjusting group voltage during usage, follow the steps blow to configure withstand voltage.

- If the group voltage is to be switched from 1.8 V to 3.3 V, configure withstand voltage to 3.3 V before switching.
- If the group voltage is to be switched from 3.3 V to 1.8 V, configure withstand voltage to 1.8 V after switching.

8.6.5 Register List

There are two groups of registers for GPIO.

8.7 LEDC

8.7.1 Overview

The LEDC is used to control the external LED lamp.

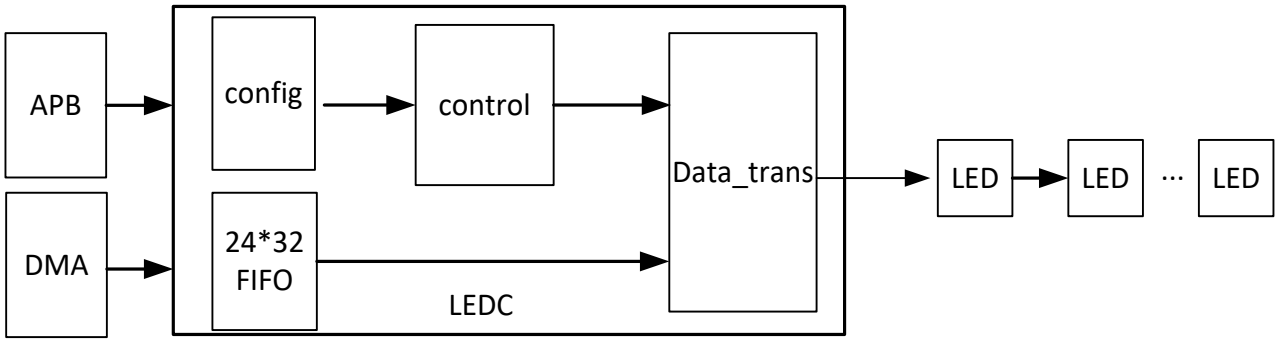
The LEDC has the following features:

- Configurable LED output high-/low-level width
- Configurable LED reset time
- Configurable interval time for packets and frame data
- LEDC data supports DMA configuration mode and CPU configuration mode
- Maximum 1024 LEDs serial connect
- LED data transfer rate up to 800 kbit/s
- Configurable RGB display mode

8.7.2 Block Diagram

The following figure shows a block diagram of the LEDC.

Figure 8-35 LEDC Block Diagram



LEDC contains the following sub-blocks:

Table 8-26 LEDC Sub-blocks

Sub-block	Description
config	register configuration
control	LEDC timing control and status control
FIFO	24-bit width x 32 depth
Data_trans	Convert input data to the 0 and 1 characters of LED

8.7.3 Functional Description

8.7.3.1 External Signals

The following table describes the external signals of the LEDC.

Table 8-27 LEDC External Signals

Signal Name	Description	Type
LEDC	Intelligent Control LED Signal Output	O

8.7.3.2 Clock Sources

The following table describes the clock sources of the LEDC.

Table 8-28 LEDC Clock Sources

Clock Sources	Description	module
HOSC	24 MHz	CCU
PERI0_600M	Peripheral Clock. The default value is 600 MHz	

8.7.3.3 Reset

The following table describes the reset of the LEDC.

Table 8-29 LEDC Reset

Reset signal	Source
LEDC_RST	CCU

8.7.3.4 LEDC Timing

Figure 8-36 LEDC Package Output Timing Diagram

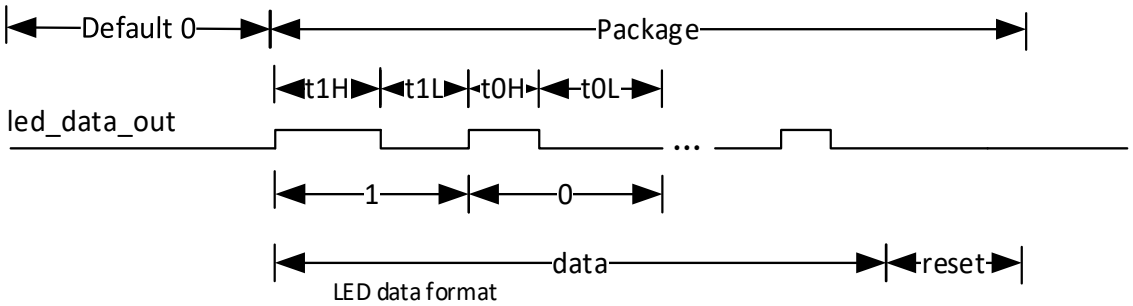
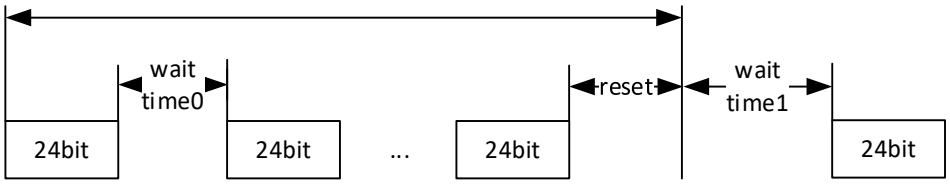


Figure 8-37 LEDC 1-frame Output Timing Diagram



8.7.3.5 LEDC Input Data Structure

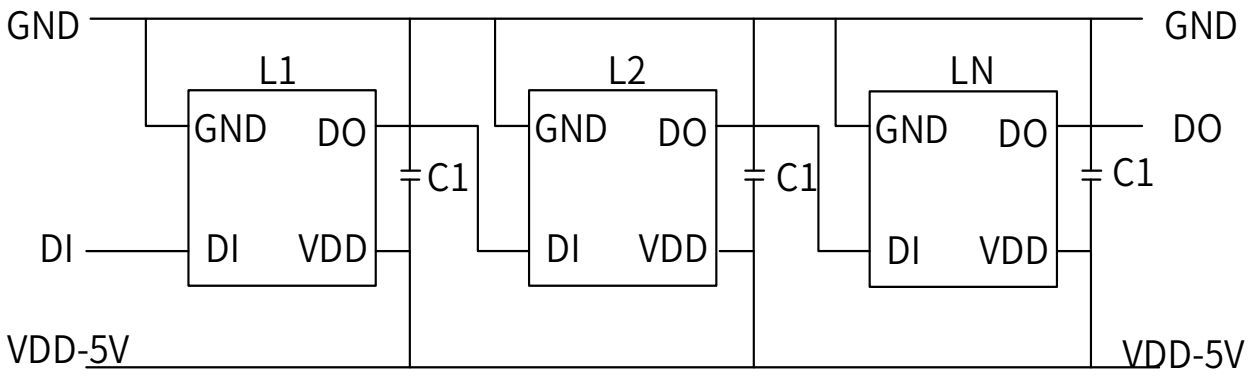
The RGB mode of LEDC data is configurable. By default, the data is sent in GRB order, and the higher bit is transmitted first.

Figure 8-38 LEDC Input Data Structure

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

8.7.3.6 LEDC Typical Circuit

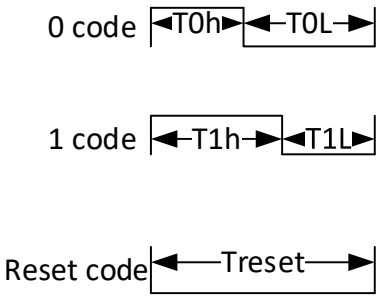
Figure 8-39 LEDC Typical Circuit



C1 is the filter capacitor of LED light, and its value is usually 100 Nf.

8.7.3.7 LEDC Data Input Code

Figure 8-40 LEDC Data Input Code



8.7.3.8 LEDC Data Transfer Time

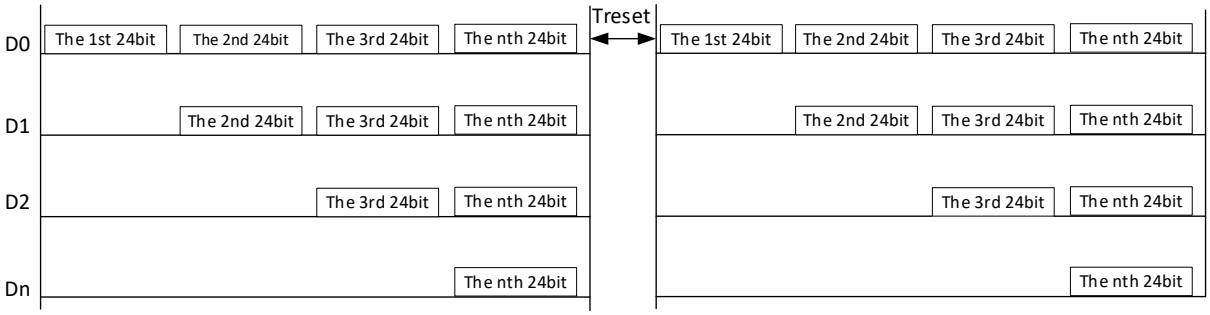
The time parameter of the typical LED specification shows as follows.

Table 8-30 Time Parameters of Typical LED Specification

T0H	0 code, high-level time	220 ns to 380 ns
T0L	0 code, low-level time	580 ns to 1.6 us
T1H	1 code, high-level time	580 ns to 1.6 us
T1L	1 code, low-level time	220 ns to 420 ns
RESET	Frame unit, low-level time	> 280 us

8.7.3.9 LEDC Data Transfer Mode

Figure 8-41 LEDC Data Transfer Mode



8.7.3.10 LEDC Parameter

- PAD rate > 800 kbit/s
- LED number supported: $T_{0\text{-}code}$: 800 ns to 1980 ns, $T_{1\text{-}code}$: 800 ns to 2020 ns

When the LED refresh rate is 30 frame/s, LED number supported is (1 s/30-280 us)/ ((800 ns to 2020 ns) *24) =1023 to 681.

When the LED refresh rate is 60 frame/s, LED number supported is (1 s/60-280 us)/ ((800 ns to 2020 ns) *24) =853 to 337.

8.7.3.11 LEDC Data Transfer

The LEDC supports DMA data transfer mode or CPU data transfer mode. The DMA data transfer mode is set by LEDC_DMA_EN

- Data transfer in DMA mode

When the valid space of internal FIFO is greater than the setting FIFO free space threshold, the LEDC sends DMA_REQ to require DMA to transfer data from DRAM to LEDC. The maximum data transfer size in DMA mode is 16 words. (The internal FIFO level is 32.)

- Data transfer in CPU mode

When the valid space of internal FIFO is greater than the setting FIFO free space threshold, the LEDC sends LEDC_CPUREQ_INT to require CPU to transfer data to LEDC. The transfer data size in CPU mode is controlled by software. The internal FIFO destination address is 0x06700014. The data width is 32-bit. (The lower 24-bit is valid.)

8.7.3.12 LEDC Interrupt

Module Name	Description
FIFO_OVERFLOW_INT	FIFO overflow interrupt. The data written by external is more than the maximum storage space of LED FIFO, the LEDC will be in data loss state. At this time, software needs to deal with the abnormal situation. The processing mode is as follows. The software can query LED_FIFO_DATA_REG to determine which data has been stored in the internal FIFO of LEDC. The LEDC performs soft_reset operation to refresh all data.
FIFO_CPUREQ_INT	FIFO request CPU data interrupt When FIFO data is less than a threshold, the interrupt will be reported to the CPU.
LEDC_TRANS_FINISH_INT	Data transfer complete interrupt The value indicates that the data configured as total_data_length has been transferred completely.

LEDC interrupt usage scenario:

- CPU mode

The software can enable GLOBAL_INT_EN, FIFO_CPUREQ_INT_EN, WAITDATA_TIMEOUT_INT_EN, FIFO_OVERFLOW_INT_EN, LEDC_TRANS_FINISH_INT_EN, and cooperate with LEDC_FIFO_TRIG_LEVEL to use. When FIFO_CPUREQ_INT is set to 1, the software can configure data of LEDC_FIFO_TRIG_LEVEL to LEDC.

- DMA mode

The software can enable GLOBAL_INT_EN, WAITDATA_TIMEOUT_INT_EN, FIFO_OVERFLOW_INT_EN, LEDC_TRANS_FINISH_INT_EN, and cooperate with

LEDC_FIFO_TRIG_LEVEL to use. When DMA receives LEDC DMA_REQ, DMA can transfer data of LEDC_FIFO_TRIG_LEVEL to LEDC.

8.7.4 Programming Guidelines

8.7.4.1 LEDC Normal Configuration Process

Step 1 Configure LEDC_CLK and bus pclk.

Step 2 Configure the written LEDC data.

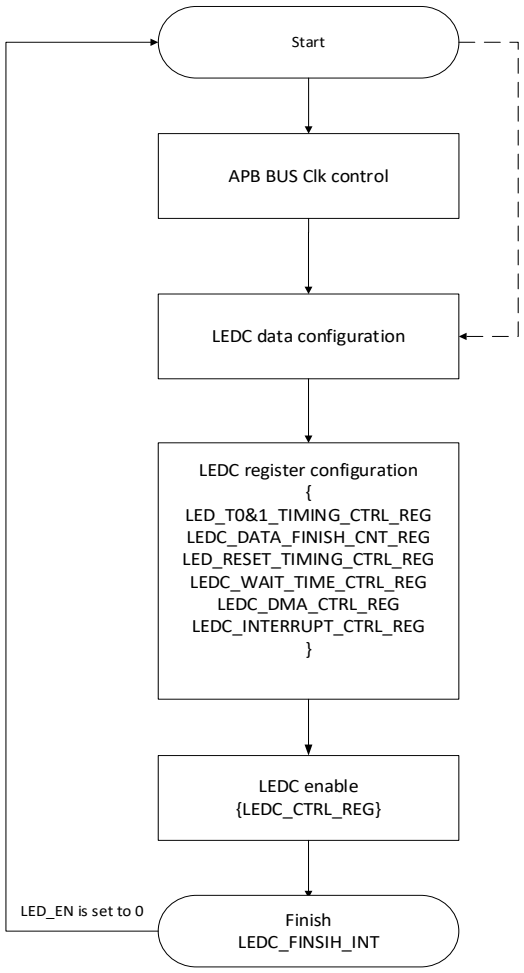
Step 3 Configure LED_T01_TIMING_CTRL_REG, LEDC_DATA_FINISH_CNT_REG, LED_RESET_TIMING_CTRL_REG, LEDC_WAIT_TIME0_CTRL_REG, LEDC_DMA_CTRL_REG, LEDC_INTERRUPT_CTRL_REG. Configure 0-code, 1-code, reset time, LEDC waiting time, and the number of external connected LEDC and the threshold of DMA transfer data.

Step 4 Configure LEDC_CTRL_REG to enable LEDC_EN, the LEDC will start to output data.

Step 5 When the LEDC interrupt is pulled up, it indicates the configured data has transferred complete, at this time LED_EN will be set to 0, and the read/write point of LEDC FIFO is cleared to 0.

Step 6 Repeat step1, 2, 3, 4 to re-execute a new round of configuration, enable LEDC_EN, the LEDC will start new data transfer.

Figure 8-42 LEDC Normal Configuration Process



8.7.4.2 LEDC Abnormal Scene Processing Flow

WAITDATA_TIMEOUT Abnormal Status

Step 1 When WAITDATA_TIMEOUT_INT appears, it indicates the internal FIFO data request of LEDC cannot obtain a response, at this time if the default output level is low, then the external LED may think there was a reset operation and cause LED data to be flushed incorrectly.

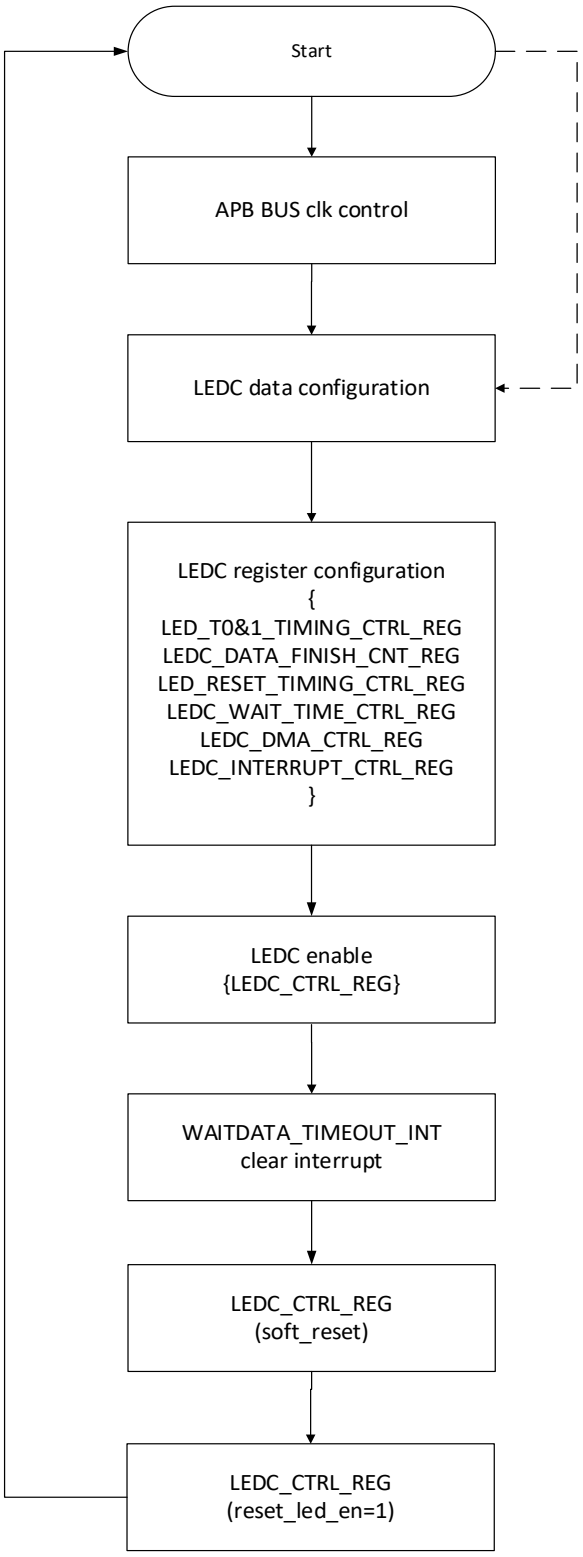
Step 2 The LEDC needs to be performed soft_reset operation (LEDC_SOFT_RESET=1); after soft_reset, the LEDC_EN will be pulled-down automatically, all internal status register and control state machine will return to the idle state, the LEDC FIFO read & write point is cleared to 0, the LEDC interrupt is cleared.

Step 3 Setting reset_led_en to 1 indicates LEDC can actively send a reset operation to ensure the external LED lamp in the right state.

Step 4 The software reads the status of reset_led_en, when the status value is 1, it indicates LEDC does not perform the transmission of LED reset operation; when the status value is 0, the LEDC completes the transmission of LED reset operation.

Step 5 When LEDC reset operation finishes, the LEDC data and register configuration need to be re-operated to start re-transmission data operation.

Figure 8-43 LEDC Timeout Abnormal Processing Flow



FIFO Overflow Abnormal Status

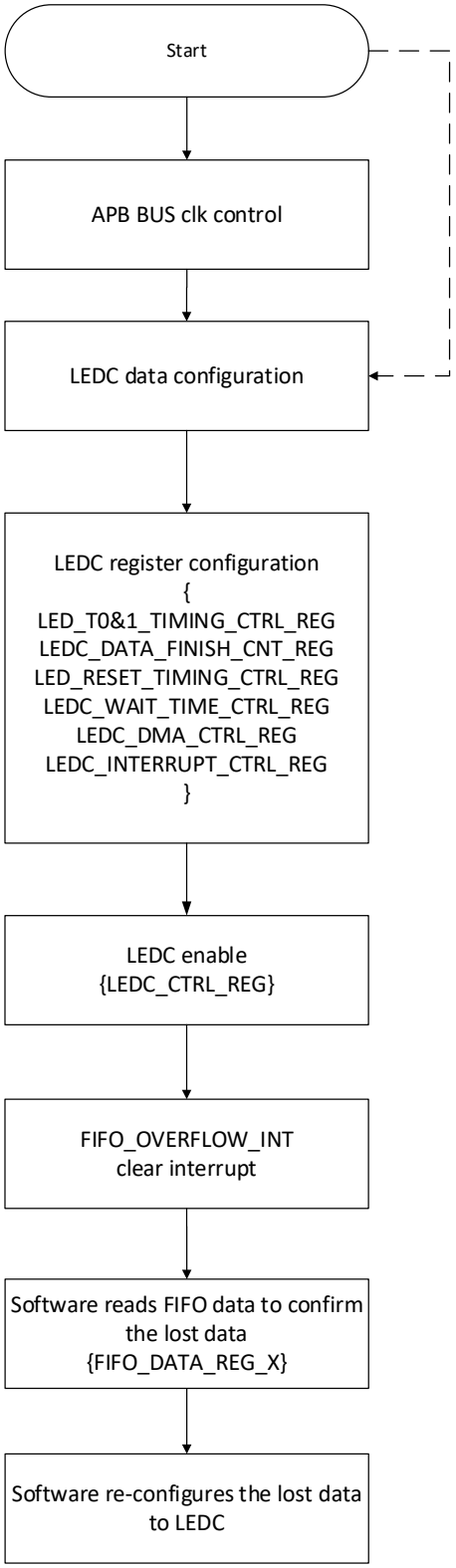
Step 1 When FIFO_OVERFLOW_INT appears, it indicates the data configured by software exceeds the LEDC FIFO space, at this time the redundant data will be lost.

Step 2 The software needs to read data in LEDC_FIFO_DATA_X to confirm the lost data.

Step 3 The software re-configures the lost data to the LEDC.

Step 4 If the software uses the soft_reset operation, the operation is the same with the timeout abnormal processing flow.

Figure 8-44 FIFO Overflow Abnormal Processing Flow



8.7.5 Register List

Module Name	Base Address
LEDC	0x02008000

8.8 Low rate ADC (LRADC)

8.8.1 Overview

The low rate analog-to-digital converter (LRADC) can convert the external signal into a certain proportion of digital value, to realize the measurement of analog signal, which can be applied to power detection and key detection.

The LRADC has the following features:

- 2-ch LRADC input
- 6-bit resolution
- Sampling rate up to 2 KHz
- Supports hold key and general key
- Support normal, continue and single work mode
- Power supply voltage:1.8V, power reference voltage:1.35V

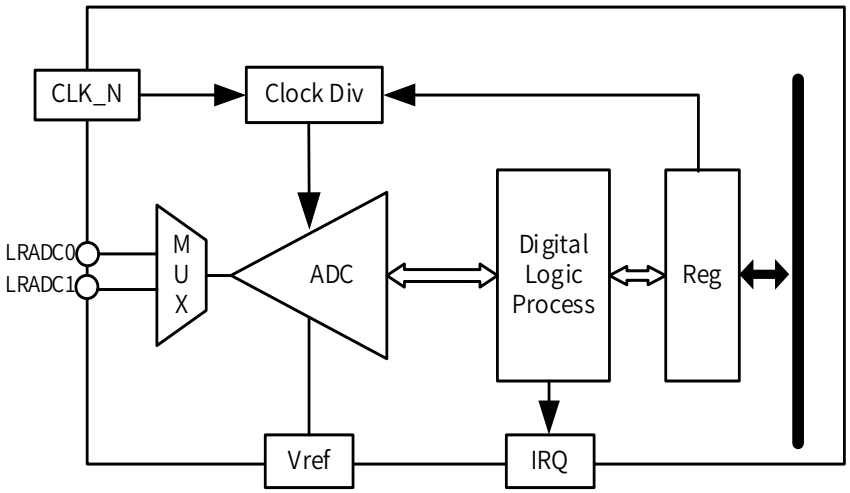
NOTE

The LRADC has a 6-bit resolution, 1-bit offset error, and 1-bit precision error. After the LRADC calibrates 1-bit offset error, the LRADC has 5-bit precision.

8.8.2 Block Diagram

The following figure shows the block diagram of the LRADC.

Figure 8-45 LRADC Block Diagram



8.8.3 Functional Description

8.8.3.1 External Signals

The following table describes the external signals of the LRADC. The LRADC pin is the analog input signal.

Table 8-31 LRADC External Signals

Signal Name	Description	Type
LRADC0	Low Rate ADC	AI
LRADC1	Low Rate ADC	AI

8.8.3.2 Clock Source

The LRADC has one clock source. The following table describes the clock source for LRADC.

Table 8-32 LRADC Clock Sources

Clock Source	Description
LOSC	32.768 kHz LOSC

8.8.3.3 LRADC Working Mode

- Normal Mode

The LRADC gathers 8 samples, the average value of these 8 samples is updated in the data register, and the data interrupt sign is enabled. It is sampled repeatedly according to this mode until the LRADC is disabled.

- Continuous Mode

The LRADC gathers 8 samples every other 8*(N+1) sample cycle. The average value of every 8 samples is updated in the data register, and the data interrupt sign is enabled. (N is defined in the bit [19:16] of LRADC_CTRL).

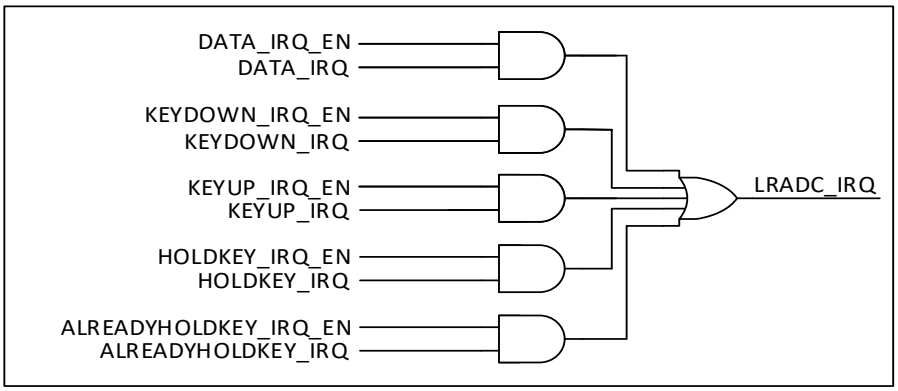
- Single Mode

The LRADC gathers 8 samples, and the average value of these 8 samples is updated in the data register, and the data interrupt sign is enabled at the same time, then the LRADC stops sample.

8.8.3.4 Interrupt

Each LRADC channel has five interrupt sources and five interrupt enable controls.

Figure 8-46 LRADC Interrupt



When the input voltage is between LEVEL A (1.35 V) and LEVEL B (control by the bit [5:4] of LRADC_CTRL), the IRQ1 can be generated. When the input voltage is lower than LEVEL B, the IRQ2 can be generated.

If the controller receives IRQ1 and does not receive IRQ2 at the same time, then the controller will generate Hold Key Pending, otherwise Data IRQ Pending.

The Hold KEY usually is used for the self-locking key. When the self-locking key holds a locking status, the controller receives the IRQ2, then the controller will generate Already Hold Pending.

8.8.3.5 Calculation Formula

Calculation formula: LRADC_DATA = Vin/VREF*63, VREF=1.35 V

8.8.4 Programming Guidelines

8.8.4.1 Normal Detecting

Perform the following steps for normal detecting mode:

Step 1 Configure LRADC_BGR_REG[LRADC_GATING] to 0 to disable the clock of LRADC.

Step 2 Configure LRADC_BGR_REG[LRADC_RST] to 1 to deassert the reset of LRADC.

Step 3 Configure LRADC_BGR_REG[LRADC_GATING] to 1 to enable the clock of LRADC.

Step 4 Configure LRADC_CTRL[LRADC_SAMPLE_RATE] to set the appropriate sampling frequency.

Step 5 Configure LRADC_CTRL[LEVELB_VOL] to set the appropriate voltage threshold.

Step 6 Configure LRADC_CTRL[FIRST_CONVER_DLY] and LRADC_CTRL[LEVELA_B_CNT] to set the appropriate debounce value.

Step 7 Configure LRADC_CTRL[LRADC_HOLD_KEY_EN] to 1.

Step 8 Configure LRADC_CTRL[KEY_MODE_SELECT] to 0 to set the normal mode.

Step 9 Configure LRADC_INTC to enable the corresponding interrupt.

Step 10 Configure LRADC_CTRL[LRADC_HOLD_KEY_EN] to 1.

Step 11 Read the corresponding key voltage value from LRADC_DATA when the CPU receives the LRADC interrupt.

8.8.4.2 Single Detecting

Perform the following steps for the single detecting mode:

Step 1 Configure LRADC_BGR_REG[LRADC_GATING] to 0 to disable the clock of LRADC.

Step 2 Configure LRADC_BGR_REG[LRADC_RST] to 1 to deassert the reset of LRADC.

Step 3 Configure LRADC_BGR_REG[LRADC_GATING] to 1 to enable the clock of LRADC.

Step 4 Configure LRADC_CTRL[LRADC_SAMPLE_RATE] to set the appropriate sampling frequency.

Step 5 Configure LRADC_CTRL[LEVELB_VOL] to set the appropriate voltage threshold.

Step 6 Configure LRADC_CTRL[FIRST_CONVER_DLY] and LRADC_CTRL[LEVELA_B_CNT] to set the appropriate debounce value.

Step 7 Configure LRADC_CTRL[LRADC_HOLD_KEY_EN] to 1.

Step 8 Configure LRADC_CTRL[KEY_MODE_SELECT] to 1 to set the single mode.

Step 9 Configure LRADC_INTC to enable the corresponding interrupt.

Step 10 Configure LRADC_CTRL[LRADC_HOLD_KEY_EN] to 1.

Step 11 Read the corresponding key voltage value from LRADC_DATA when the CPU receives the LRADC interrupt.

8.8.4.3 Continuous Detecting

Perform the following steps for continuous detecting mode:

Step 1 Configure LRADC_BGR_REG[LRADC_GATING] to 0 to disable the clock of LRADC.

Step 2 Configure LRADC_BGR_REG[LRADC_RST] to 1 to deassert the reset of LRADC.

Step 3 Configure LRADC_BGR_REG[LRADC_GATING] to 1 to enable the clock of LRADC.

Step 4 Configure LRADC_CTRL[LRADC_SAMPLE_RATE] to set the appropriate sampling frequency.

Step 5 Configure LRADC_CTRL[LEVELB_VOL] to set the appropriate voltage threshold.

Step 6 Configure LRADC_CTRL[FIRST_CONVER_DLY] and LRADC_CTRL[LEVELA_B_CNT] to set the appropriate debounce value.

Step 7 Configure LRADC_CTRL[LRADC_HOLD_KEY_EN] to 1.

Step 8 Configure LRADC_CTRL[KEY_MODE_SELECT] to 2 to set the continuous mode, and configure LRADC_CTRL[CONTINUE_TIME_SELECT] to set a sampling interval.

Step 9 Configure LRADC_INTC to enable the corresponding interrupt.

Step 10 Configure LRADC_CTRL[LRADC_HOLD_KEY_EN] to 1.

Step 11 Read the corresponding key voltage value from LRADC_DATA when the CPU receives the LRADC interrupt.

8.8.5 Register List

Module Name	Base Address
LRADC	0x02009800

Register Name	Offset	Description
LRADC_CTRL	0x0000	LRADC Control Register
LRADC_INTC	0x0004	LRADC Interrupt Control Register
LRADC_INTS	0x0008	LRADC Interrupt Status Register
LRADC_DATA0	0x000C	LRADC Data Register0
LRADC_DATA1	0x0010	LRADC Data Register1

8.8.6 Register Description

8.8.6.1 0x0000 LRADC Control Register (Default Value: 0x0100_0168)

Offset: 0x0000			Register Name: LRADC_CTRL
Bit	Read/Write	Default/Hex	Description
31:24	R/W	0x1	FIRST_CONVERT_DLY ADC First Convert Delay Setting ADC conversion is delayed by n samples.
23:22	R/W	0x0	CHANNEL_SEL Select the channel that be enabled 00: channel 0 only 01: channel 1 only 10: channel 1 and channel 0
21:20	/	/	/
19:16	R/W	0x0	CONTINUE_TIME_SELECT Continuous Mode Time Select One of 8*(N+1) sample as a valuable sample data.
15:14	/	/	/

8.9 USB2.0 DRD

8.9.1 Overview

The USB2.0 dual-role device (USB2.0 DRD) supports both device and host functions which can also be configured as a Host-only or Device-only controller. It complies with the USB2.0 Specification.

For saving CPU bandwidth, the DMA interface of the DRD module can also support the external DMA controller to do the data transfer between the memory and the DRD FIFO. The DRD core also supports USB power saving functions.

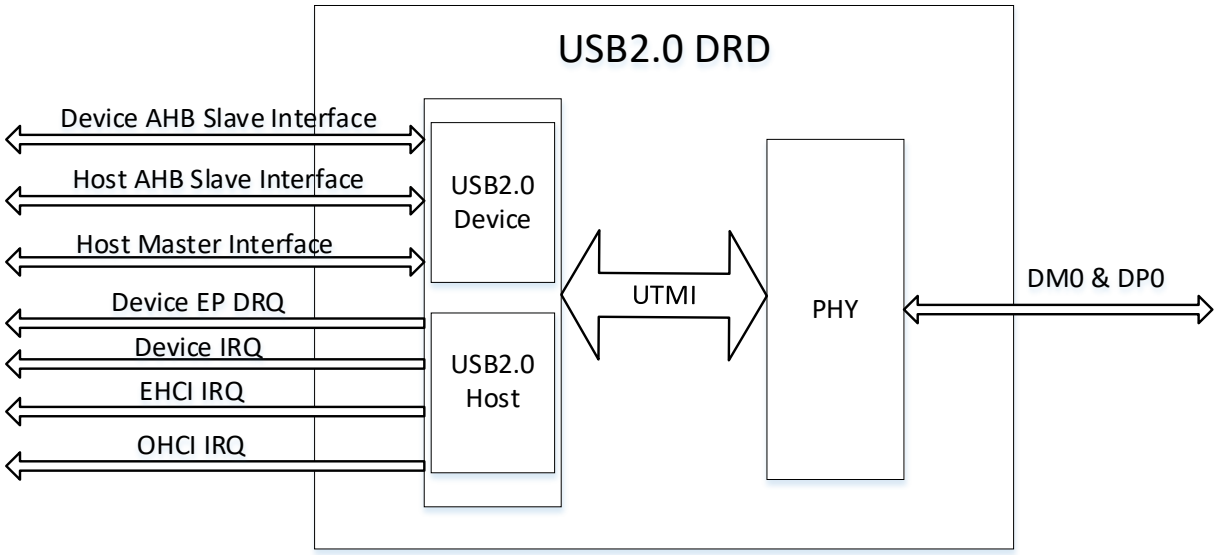
The USB2.0 DRD has the following features:

- One USB2.0 DRD (USB0), with integrated USB 2.0 analog PHY
- Complies with USB2.0 Specification
- Supports static host operation:
  - Compatible with Enhanced Host Controller Interface (EHCI) Specification, Version 1.0
  - Compatible with Open Host Controller Interface (OHCI) Specification, Version 1.0a
  - Supports High-Speed (HS, 480 Mbit/s), Full-Speed (FS, 12 Mbit/s), and Low-Speed (LS, 1.5 Mbit/s)
  - Supports only 1 USB Root port shared between EHCI and OHCI
- Supports static device operation:
  - Supports High-Speed (HS, 480 Mbit/s), Full-Speed (FS, 12 Mbit/s)
  - Supports bi-directional endpoint0 (EP0) for Control transfer
  - Up to 10 user-configurable endpoints (EP1 IN/OUT, EP2 IN/OUT, EP3 IN/OUT, EP4 IN/OUT, EP5 IN/OUT) for Bulk transfer, Isochronous transfer and Interrupt transfer
  - Up to (8 KB + 64 Bytes) FIFO for all EPs (including EP0)
  - Supports interface to an external Normal DMA controller for every EP (EP1, EP2, EP3, EP4, and EP5)
- Supports an internal DMA controller for data transfer with memory
- Supports High-Bandwidth Isochronous & Interrupt transfers
- Automated splitting/combining of packets for Bulk transfers
- Includes automatic PING capabilities
- Soft connect/disconnect function
- Hardware handles all data transfer
- Power optimization and power management capabilities
- Device and host controller share an 8K SRAM and a physical PHY

8.9.2 Block Diagram

The following figure shows the block diagram of USB2.0 DRD Controller.

Figure 8-47 USB2.0 DRD Controller Block Diagram



8.9.3 Functional Description

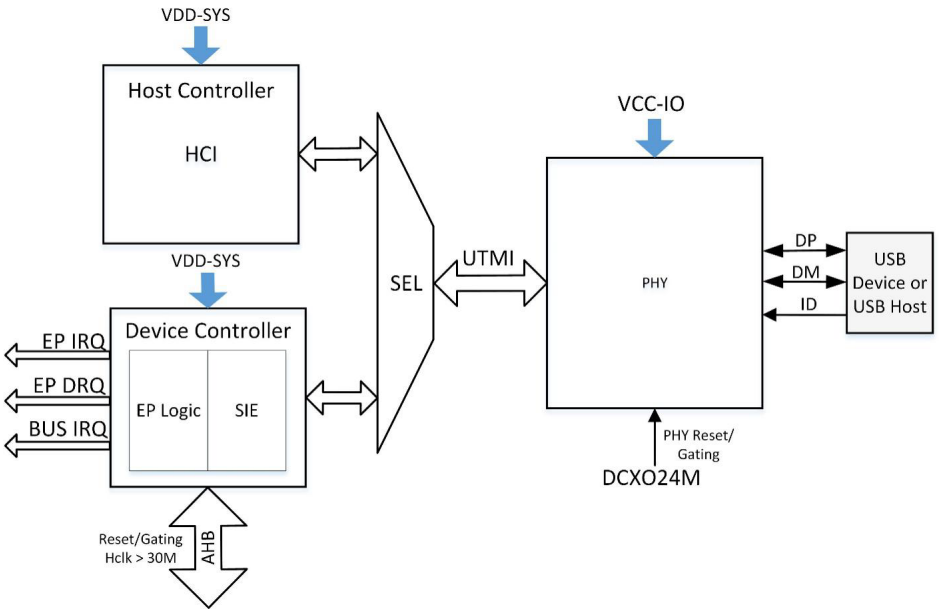
8.9.3.1 External Signals

Table 8-33 USB2.0 DRD External Signals

Signal Name	Description	Type
USB0-DM	USB2.0 Data Signal DM	A I/O
USB0-DP	USB2.0 Data Signal DP	A I/O
USB0-REXT	USB2.0 External Reference Resistor	AO
VCC33-USB	3.3 V Analog Power Supply for USB2.0 DRD and USB2.0 Host	P
VCC33-18-USB	3.3 V/1.8V Analog Power Supply for USB2.0 DRD and USB2.0 Host	P
VDD09-USB	0.9 V USB Digital Power Supply	p

8.9.3.2 Controller and PHY Connection Diagram

Figure 8-48 USB2.0 DRD Controller and PHY Connection Diagram



8.10 USB2.0 HOST

8.10.1 Overview

The USB Host Controller is fully compliant with USB 2.0 Specification, Enhanced Host Controller Interface (EHCI) Specification Revision 1.0 and Open Host Controller Interface (OHCI) Specification Release 1.0a.

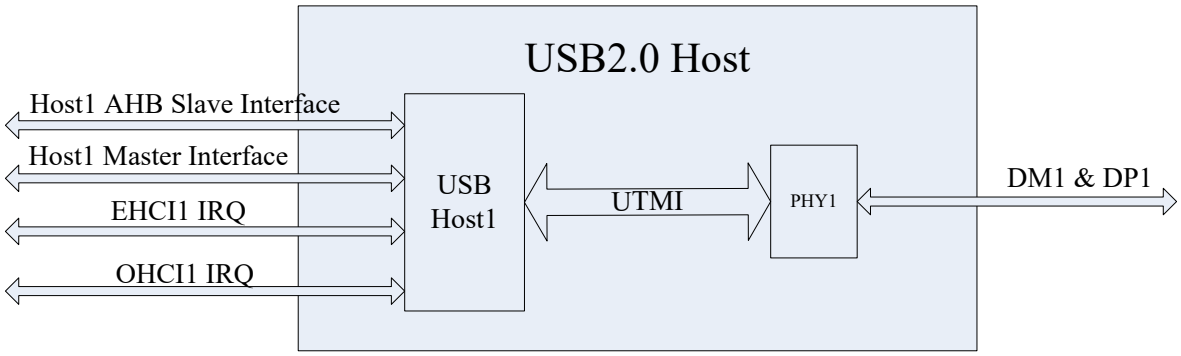
The USB2.0 host controller includes the following features:

- One USB 2.0 HOST (USB1), with integrated USB 2.0 analog PHY
- Supports High-Speed (HS, 480 Mbit/s), Full-Speed (FS, 12 Mbit/s) and Low-Speed (LS, 1.5 Mbit/s) Device
- Compatible with Enhanced Host Controller Interface (EHCI) Specification, Version 1.0
- Compatible with Open Host Controller Interface (OHCI) Specification, Version 1.0a
- Supports only 1 USB Root port shared between EHCI and OHCI
- An internal DMA Controller for data transfer with memory
- Supports the UTMI+ Level 3 interface and 8-bit bidirectional data buses
- Industry-standard AMBA High-Performance Bus (AHB), fully compliant with the AMBA Specification, Revision 2.0.
- 32-bit Little Endian AMBA AHB Slave Bus for Register Access
- 32-bit Little Endian AMBA AHB Master Bus for Memory Access

8.10.2 Block Diagram

The following figure shows the block diagram of USB2.0 Host Controller.

Figure 8-49 USB2.0 Host Controller Block Diagram



8.10.3 Functional Description

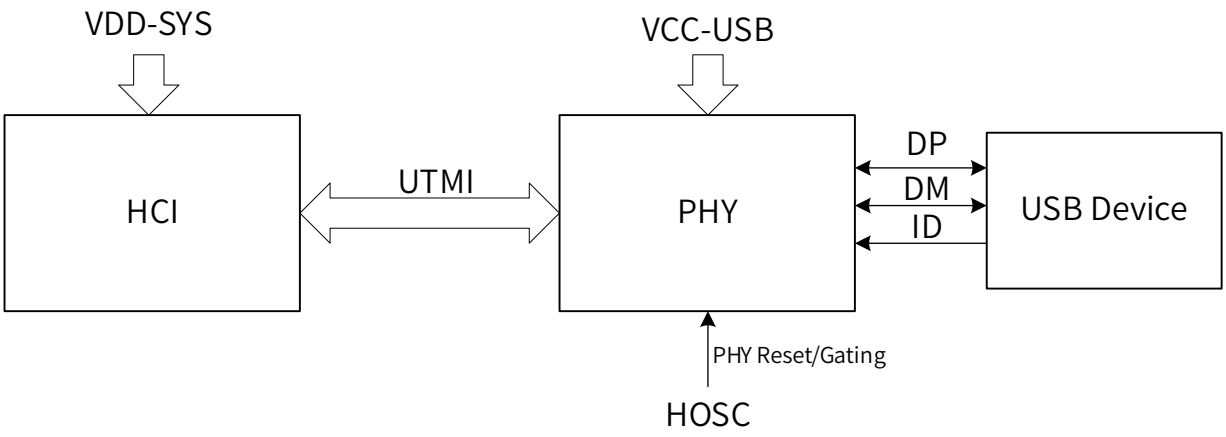
8.10.3.1 External Signals

Table 8-34 USB2.0 Host External Signals

Signal Name	Description	Type
USB1-DM	USB2.0 Data Signal DM	A I/O
USB1-DP	USB2.0 Data Signal DP	A I/O
USB1-REXT	USB2.0 External Reference Resistor AO	AO

8.10.3.2 Controller and PHY Connection Diagram

Figure 8-50 USB2.0 Host Controller and PHY Connection Diagram



8.11 PCIe2.1&USB3.1 Top System

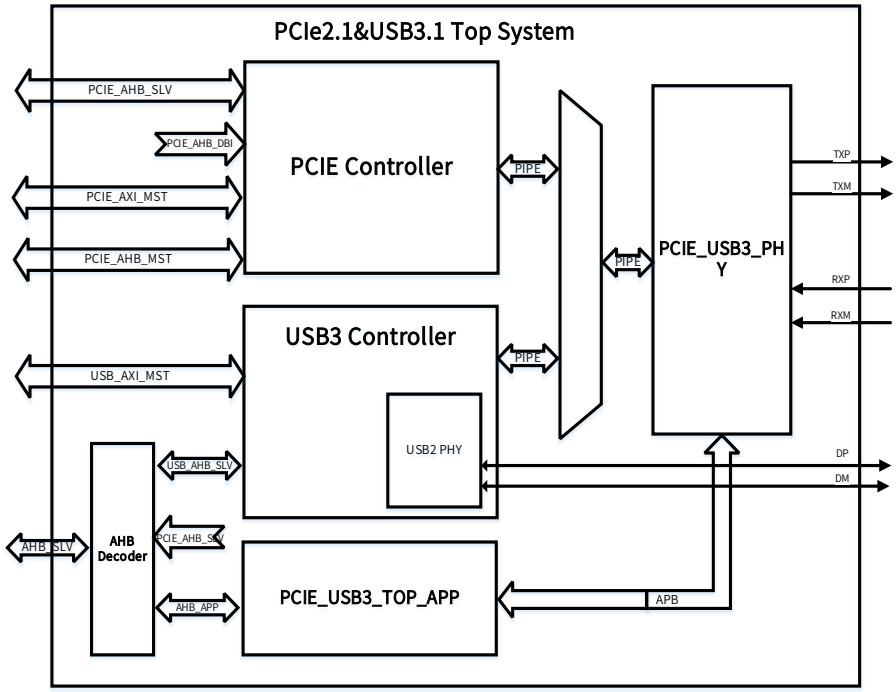
8.11.1 Overview

The PCIe2.1&USB3.1 top system integrates a PCIe2.1 RC controller and a USB3.1 DRD controller with a Combo PHY which supports PCIe GEN1 GEN2 and USB3.1 GEN1 speed and is shared through PIPE interface.

8.11.2 Block Diagram

The following figure shows the block diagram of PCIe2.1&USB3.1 top system.

Figure 8-51 PCIe2.1&USB3.1 Top System Block Diagram



NOTE

This chapter foucuses on the 1 PCIe2.1&USB3.1 combo PHY. For the detailed description of PCIe2.1 and USB3.1 DRD, please refer to section 8.13 PCIe2.1 and section 8.12 USB3.1 DRD.

8.11.3 Register List

Module Name	Base Address
PCIE_USB3_TOP_APP	0x04F00000

Register Name	Offset	Description
PCIEBGR	0x0004	PCIE Bus Gating Reset Register
USB3BGR	0x0008	USB3 Bus Gating Reset Register

8.12 USB3.1 DRD

8.12.1 Overview

The USB3.1 DRD is a Dual-Role Device (DRD) controller, which supports both device and host functions which can also be configured as a host-only or device-only controller, fully compliant with the USB 3.1 Specification. It can support super-speed (SS, 5-Gbit/s), high-speed (HS, 480-Mbit/s), full-speed (FS, 12-Mbit/s), and low-speed (LS, 1.5-Mbit/s) transfers in Host mode. It can support super-speed (SS, 5-Gbit/s), high-speed (HS, 480-Mbit/s), and full-speed (FS, 12-Mbit/s) in Device mode. Standard USB transceiver can be used through its UTMI+ interface and PIPE interface.

The USB3 DRD controller includes the following features:

- Compliant with USB3.1 GEN1 Specification
- One USB 2.0 UTMI+ PHY (USB2)
- One USB3.1 PIPE PHY (USB3)
- Support Device or Host operation at a time
- USB3.1 DRD Device mode supports the following:
  - Super-Speed (SS, 5 Gbit/s) for USB3.1 PHY
  - High-Speed (HS, 480 Mbit/s) and Full-Speed (FS, 12-Mbit/s) for USB2.0 PHY
- USB3.1 DRD HOST mode supports the following:
  - Super-Speed (SS, 5 Gbit/s) for USB3.1 PHY
  - High-Speed (HS, 480 Mbit/s), Full-Speed (FS, 12 Mbit/s) and Low-Speed (LS, 1.5 Mbit/s) for USB2.0 PHY
- AXI interface for DMA operation
- Reading and writing access to Control and Status Registers (CSRs) through AHB Slave interface
- Up to 10 Endpoints, including bi-directional control Endpoint 0 in Device mode:
  - 5 IN Endpoints: User EP1 IN, EP2 IN, EP3 IN, EP4 IN, Control EP0 IN
  - 5 OUT Endpoints: User EP1 OUT, EP2 OUT, EP3 OUT, EP4 OUT, Control EP0 OUT
- Simultaneous IN and OUT transfer in Super-Speed mode
- Dual-port interfaces for TX data buffering, RX data prefetching, descriptor caching, and register caching
- Three RAMs: RX data FIFO RAM, TX data FIFO RAM, and descriptor/register Cache RAM
- Hardware handles all data transfer
- Implements both static and dynamic power reduction techniques at multiple levels

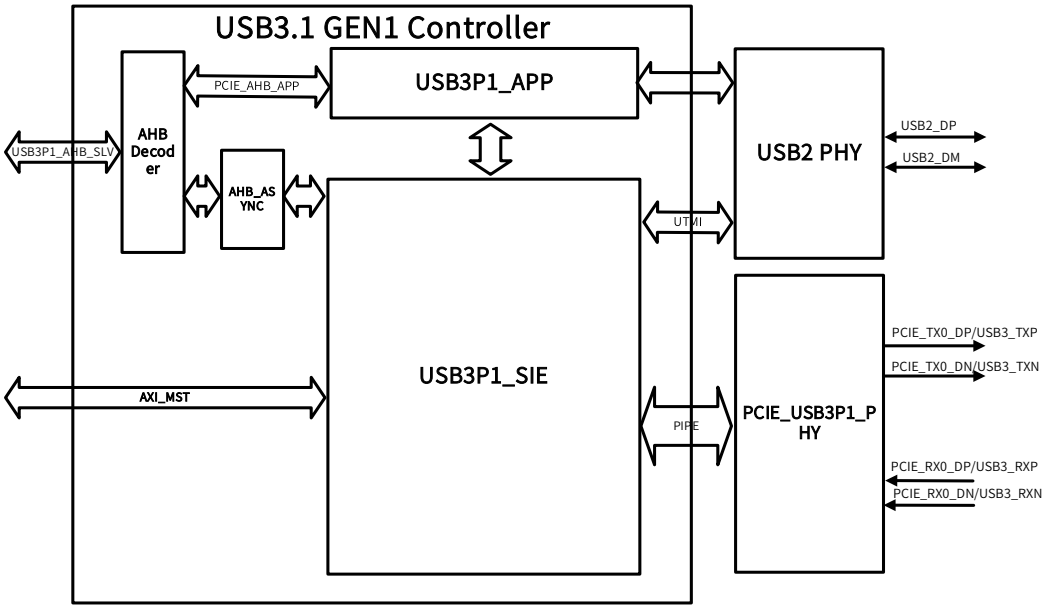
NOTE

USB2.0 PHY and USB3.1 PHY share the same controller. They cannot be used simultaneously.

8.12.2 Block Diagram

The following figure shows the block diagram of USB3.1 DRD Controller

Figure 8-52 USB3.1 DRD Controller Block Diagram



8.12.3 Functional Description

8.12.3.1 External Signals

Table 8-35 USB3.1 DRD External Signals

Signal Name	Description	Type
USB2-DM	USB2.0 Data Signal DM	A I/O
USB2-DP	USB2.0 Data Signal DP	A I/O
USB2-REXT	USB2.0 External Reference Resistor	AO
USB3-RXP	USB3.1 SuperSpeed Differential Signal of RX (Positive)	AI
USB3-RXN	USB3.1 SuperSpeed Differential Signal of RX (Negative)	AI
USB3-TXP	USB3.1 SuperSpeed Differential Signal of TX (Positive)	AO
USB3-TXN	SuperSpeed Differential Signal of TX (Negative)	AO
VCC33-USB-2	3.3 V Power Supply for USB2.0 PHY	P
VCC33-18-USB-2	3.3 V/1.8 V Power Supply for USB2.0 PHY	P

NOTE

For detailed functional information, please refer to *USB3.1 Specification, USB2.0 Specification,* and *eXtensible Host Controller Interface (XHCI) Specification, Version 1.1.*

8.12.4 Register List

There are four groups of registers in USB3.1 DRD.

Register Class	Offset	Descriptions
USB3 (0x04D0 0000---0x04EF FFFF)		
xHCI Registers	0x00000000 - 0x00007FFF	For register definition and description, please see eXtensible Host Controller Interface for Universal Serial Bus (xHCI), Revision 1.1.
Global Registers	0x0000C100 - 0x0000C6FF	For register definition and description, please see DesignWare® Cores Enhanced SuperSpeed USB3.1 Controller Program Guide, Version 1.90a.
Device Registers	0x0000C700 - 0x0000CBFF	
Application Registers	0x0010 0000 - 0x001007FF	For register definition and description, please see section 8.12.4.1 and section 8.12.5.

8.12.4.1 Application Register List

Register Class	Base Address
Application Registers	0x04E0 0000 - 0x04E007FF

Register Name	Offset	Description
U2_ISCR	0x100000	USB2.0 Interface Status and Control Register
U2_PHYCTL	0x100010	USB2.0 PHY Control Register
U2_PHYTST	0x100014	USB2.0 PHY TEST Register
U2_PHYTUNE	0x100018	USB2.0 PHY Tune Register
U2_PHYSTS	0x100024	USB2.0 PHY Status Register

8.13 PCIe2.1

8.13.1 Overview

The PCI Express Controller (PCIe) is a general purpose I/O interconnect, which provides low pin count, high reliability, and high-speed data transfer at rates of up to 5.0 Gbit/s per lane per direction.

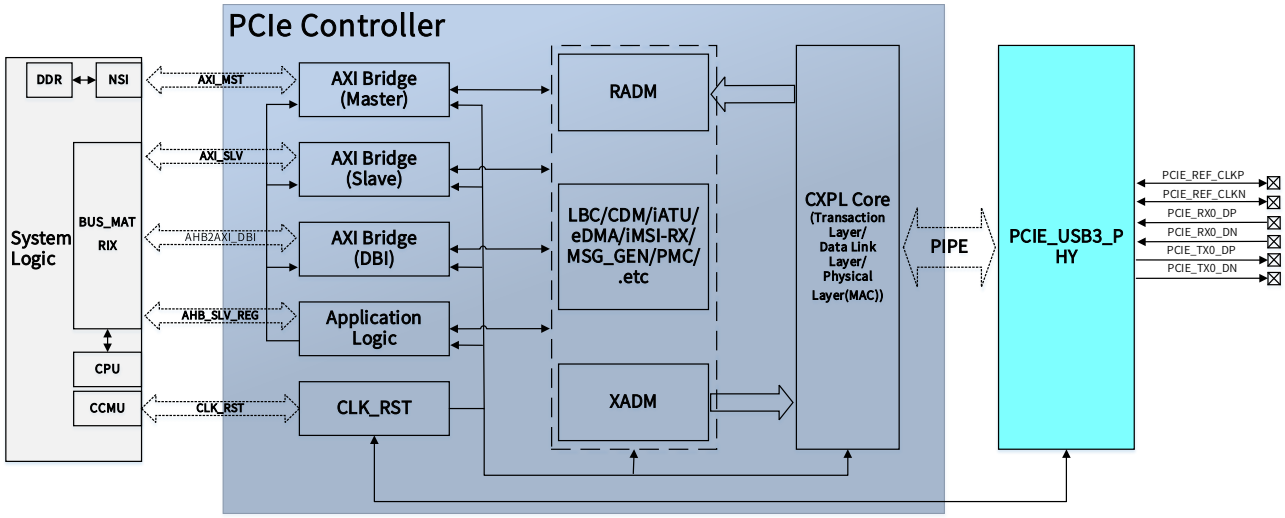
The PCIe controller includes the following features:

- Compliance to PCI Express Base Specification, Revision 2.1
- Supports Gen1(2.5 Gbit/s), Gen2 (5.0 Gbit/s) speed
- Supports 62.5MHz/125 MHz operation on PIPE interface for Gen1/Gen2, respectively
- Constant 32-bit PIPE width for Gen1/Gen2 modes
- Supports Root Complex (RC) mode
- Supports 1 lane link width
- Eight Traffic Classes (TC)
- Maximum payload size of 1K bytes
- 8 inbound and 8 outbound address translation regions
- 4 write/read channels for embedded DMA
- Maximum number of non-posted outstanding transactions: 32
- Supports Active State Power Management (ASPM)
- Support Advanced Error Reporting (AER)
- Support MSI interrupt

8.13.2 Block Diagram

The following figure shows the functional block diagram of the PCIe.

Figure 8-53 PCIe Block Diagram



The PCIe controller contains the following modules:

Table 8-36 PCIe Module

Module	Description
Common Express Port Logic (CXPL)	This module implements the basic functionality for the PCI Express physical, link, and transaction layers. The CXPL implements a large part of the transaction layer logic, all of the data link layer logic, and the MAC portion of the physical layer, including the link training and status state machine (LTSSM). The CXPL connects to the external PHY through the PIPE.
Transmit Application-Dependent Module (XADM)	This module implements the application-specific functionality of the PCI Express transaction layer for packet transmission. Its functions include: • TLP Arbitration • TLP Formation • Flow Control (FC) Credit checking The transmit path uses a cut-through architecture. It does not implement transmit buffering/queues (other than the retry buffer). The controller maintains an internal Target Completion Lookup Table to store certain TLP header information from the Rx request. Your application can use this information for transmitting completions.

Module	Description
Receive Application-Dependent Module (RADM)	This module implements application-specific functionality of the PCI Express transaction layer for packet reception. Its functions include: ● Sorting/filtering of received TLPs. The filtering rules and routing are configurable. ● Buffering and queuing of the received TLPs. ● Routing of received TLP to the controller's receive interfaces. The RADM maintains a Receive Completion Lookup Table (LUT) for completion tracking and completion-timeout monitoring of Tx non-posted requests. It indicates a timeout when an expected Rx completion does not arrive within the timeout period.
Configuration-Dependent Module (CDM)	This module implements: ● Standard PCI Express configuration space ● Controller-specific register space (Port Logic Registers)
Power Management Controller (PMC)	This module implements the power management features of the PCIe controller.
Local Bus Controller (LBC) and Data Bus Interface (DBI)	The LBC module provides a mechanism for a link partner (in EP mode only) or a local CPU (through the DBI) to access: ● Internal registers (in the CDM) ● External application registers connected externally to the ELBI
Message Generation Module (MSG_GEN)	This module transmits messages generated by the controller.
Integrated MSI Receiver (iMSI-RX)	The AXI bridge provides an integrated MSI reception module to detect and terminate inbound MSI requests(received on the RX wire).
Embedded DMA (eDMA)	The RC system CPU, or the EP application CPU, can offload the transferring of large blocks of data to the embedded DMA controller1, leaving the CPU free to perform other tasks. You can configure the DMA to have one to eight read channels and one to eight write channels.
Internal Address Translation Unit (iATU)	The PCIe controller uses the iATU to implement a local address translation scheme that replaces the TLP address and TLP header fields in the current TLP request header.

8.13.3 Functional Description

8.13.3.1 External Signals

The following table describes the external signals of the PCIe.

Table 8-37 PCIe External Signals

Signal Name	Description	Type
PCIE-REF-CLKP	PCIe2.1 Differential Signal REFCLK (Positive)	A I/O
PCIE-REF-CLKN	PCIe2.1 Differential Signal REFCLK (Negative)	A I/O

Signal Name	Description	Type
PCIE-REXT	PCIe2.1 External Reference Resistor	AO
PCIE-RX0-DP	PCIe2.1 Differential Signal of RX (Positive)	A I
PCIE-RX0-DN	PCIe2.1 Differential Signal of RX (Negative)	A I
PCIE-TX0-DP	PCIe2.1 Differential Signal of TX (Positive)	A O
PCIE-TX0-DN	PCIe2.1 Differential Signal of TX (Negative)	A O
PCIE0-PERSTN	PCIe2.1 Warm Reset	O
PCIE0-WAKEN	PCIe2.1 Wake Up	I
PCIE0-CLKREQN	PCIe2.1 Clock Request from PCIe Peripheral	I
VCC18-PCIE	1.8 V Power Supply for PCIe2.1	P
VDD09-PCIE	0.9 V Power Supply for PCIe2.1	P

8.13.3.2 Clock Sources

The following table describes the clock sources of the PCIe.

Table 8-38 PCIe Clock Sources

Clock Sources	Description	Module
USB3_PCIE_REF_CLK	24 MHz, USB3.1 DRD&Pcie2.1 PHY reference clock.	CCU
AUX_CLK	24 MHz, working clock in low power mode of the PCIe.	
MBUS_CLK	600 MHz, PCIe AXI master/slave bus clock.	
AHB_CLK	200 MHz, PCIe bus clock.	
PIPE_CLK	62.5 MHz/ 125MHz, normal working clock for the PCIe controller operating in Gen1/Gen2 mode.	

8.13.3.3 PCIe Reference Clock

PCIe reference clock is a 100 MHz differential clock supplied for the PCIe PHY. It has two clock sources.

- From Internal SoC

Configure PHY to use internal clock and enable the clock output via software. The REFCLKP/REFCLKN signal of PCIe controller serves as output signal and provides 100M differential clock for external EP.

- From External Clock Generator

Configure PHY to use external clock and disable the clock output via software. The REFCLKP/REFCLKN signal of PCIe controller serves as input signal and the external clock generator provides 100M differential clock for the PCIe PHY.

8.13.3.4 PCIe Memory Mapping

The address space of PCIe controller can be partitioned into three spaces.

- Core Configuration Space (CCS): The configuration register space of the PCIe controller itself. It is also the standard configuration register space defined by the PCIe specification.
- User Defined Space (UDS): The custom register space for the PCIe controller itself.
- Slave Command Space (SCS): Address space for configuration transaction and memory transaction. The read and write operations in this space will be transformed into configuration read and write transactions or memory read and write transactions in the PCIe domain by the PCIe controller. The PCIe command space segmentation is 0x20000000-0x2FFFFFFF.

8.13.3.5 ATU Operation

ATU is an address translation unit in PCIe controller and is used for transaction transformation and address translation. There are two types of ATU: Outbound ATU and Inbound ATU.

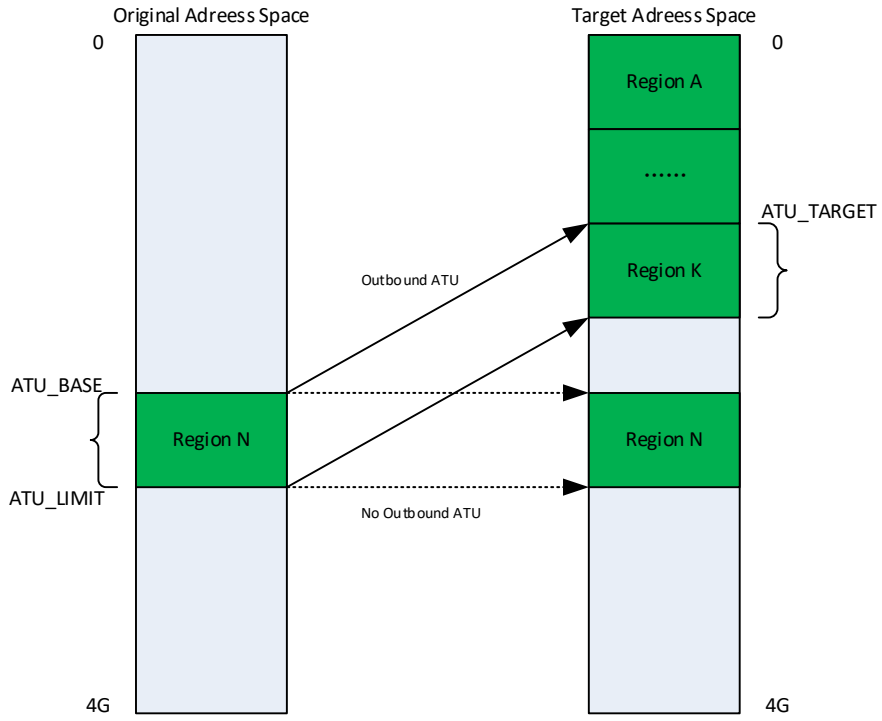
Outbound ATU

Outbound ATU is used to transform the read/write operation in the CPU domain of the original address space into the read/write transaction in the PCIe domain of the target address space. If the Outbound ATU is used, the original address in the CPU domain will be translated into another different address in the PCIe domain. If the Outbound ATU is not used, the PCIe controller will not perform address translation. The read/write operation initiated in the CPU domain will be transformed into the read/write transaction at the same address in the PCIe domain.

The following shows an example:

- When Outbound ATU is used, set address Region N in the CPU domain to be translated into Region K in PCIe domain. The read/write operation initiated at address Region N (CPU domain) will be transformed into the read/write transaction at address Region K (PCIe domain).
- When Outbound ATU is not used, read/write operation initiated at address Region N (CPU domain) will be transformed into the read/write transaction at address Region N (PCIe domain).

Figure 8-54 Outbound ATU



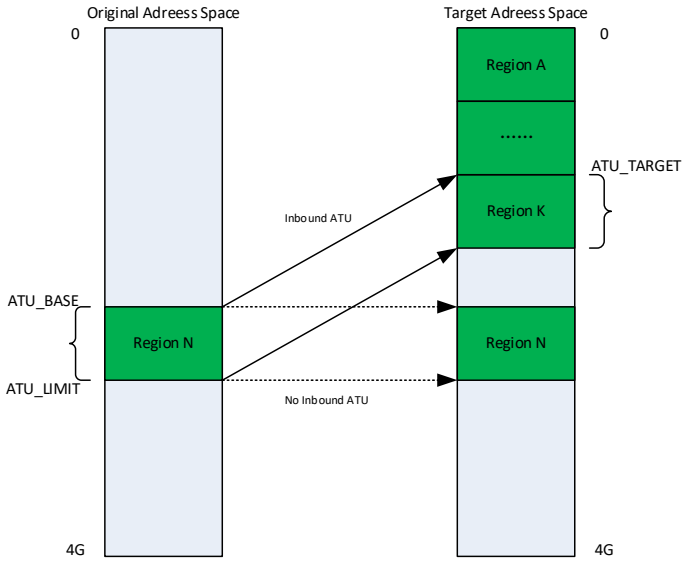
Inbound ATU

Inbound ATU is used to transform the read/write transaction in the PCIe domain of the original address space into the read/write operation in the CPU domain of the target address space. If Inbound ATU is used, the original address in the PCIe domain will be translated into another different address in the CPU domain. If the Inbound ATU is not used, the PCIe controller will not perform address translation. The read/write transaction initiated in the PCIe domain will be transformed into the read/write operation at the same address in the CPU domain.

The following shows an example:

- When Inbound ATU is used, set address Region N in the PCIe domain to be translated into Region K in the CPU domain. The read/write transaction initiated at address Region N (PCIe domain) will be transformed into the read/write operation at address Region K (CPU domain).
- When Inbound ATU is not used, read/write transaction initiated at address Region N (PCIe domain) will be transformed into the read/write operation at address Region N (CPU domain).

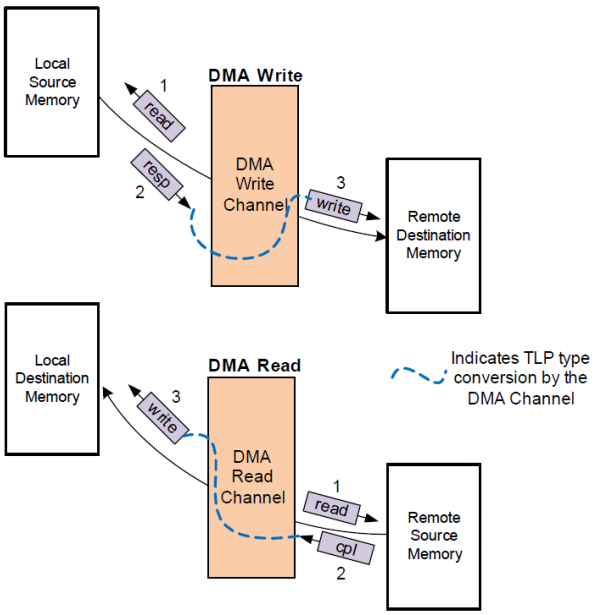
Figure 8-55 Inbound ATU



8.13.3.6 DMA Operation

There is a private DMA with read channels and write channels in the PCIe controller. The following figure describes the DMA write and read process.

Figure 8-56 DMA Operation



- DMA Write Channel

  the DMA write channel is used for data transfer from local bus address space to the address space in the PCIe domain. First, DMA controller reads data from the local bus address space and writes them to the address space in the CPU domain. Then, the PCIe controller transforms the write operation initiated by DMA into a write transaction in the PCIe domain and writes data to the destination address space in the PCIe domain.

- DMA Read Channel

  The DMA read channel is used for data transfer from the address space in the PCIe domain to the local bus address space such as DRAM address space. If a read operation is initiated by DMA in the original address space of the CPU domain, the PCIe controller will transform the read operation into a read transaction in the PCIe domain and write data to the local bus address space.

8.13.3.7 MSI Operation

Message Signaled Interrupt (MSI) is a kind of mechanism that Endpoints send interrupt requests to the CPU connected with the Root Complex, which uses memory write transaction. The transaction message for transmitting interrupt information is MSI message.

Assume an Endpoint needs to send a MSI interrupt request to CPU. First, the Endpoint needs to initiate a memory write transaction in MSI address of PCIe domain, which will be transformed into a MSI interrupt signal to CPU by the PCIe controller after the transaction is detected. Software obtains the Endpoint sending interrupt request and its interrupt vector according to the MSI message.

8.13.4 Register List

Module Name	Base Address
PCIE	0x04800000

Register Class	Offset
User Defined Registers	0x00400000 - 0x0047FFFF

Register Name	Offset	Description
MSTR_AWMISC_INFO_0	0x0100	PCIE AXI Master Write Misc Information Register0
MSTR_AWMISC_INFO_1	0x0104	PCIE AXI Master Write Misc Information Register1
MSTR_AWMISC_INFO_HDR_34DW_0	0x0108	PCIE AXI Master Write Misc Information Header Register0
MSTR_AWMISC_INFO_HDR_34DW_1	0x010C	PCIE AXI Master Write Misc Information Header Register1
MSTR_AWMISC_INFO_OTHER	0x0110	PCIE AXI Master Write Misc Information Other Register
MSTR_ARMISC_INFO_0	0x0120	PCIE AXI Master Read Misc Information Register0
MSTR_ARMISC_INFO_1	0x0124	PCIE AXI Master Read Misc Information Register1
MSTR_ARMISC_INFO_OTHER	0x0130	PCIE AXI Master Read Misc Information Other Register
MSTR_BMISC_INFO	0x0150	PCIE AXI Master Write Response Misc Information Register

8.14 Two Wire Interface (TWI)

8.14.1 Overview

The Two Wire Interface (TWI) provides an interface between a CPU and any TWI-bus-compatible device that connects via the TWI bus. The TWI is designed to be compatible with the standard I2C bus protocol. The communication of the TWI is carried out by a byte-wise mode based on interrupt polled handshaking. Each device on the TWI bus is recognized by a unique address and can operate as either transmitter or receiver, a device connected to the TWI bus can be considered as master or slave when performing data transfers. Note that a master device is a device that initiates a data transfer on the bus and generates the clock signals to permit the transfer. During this transfer, any device addressed by this master is considered a slave.

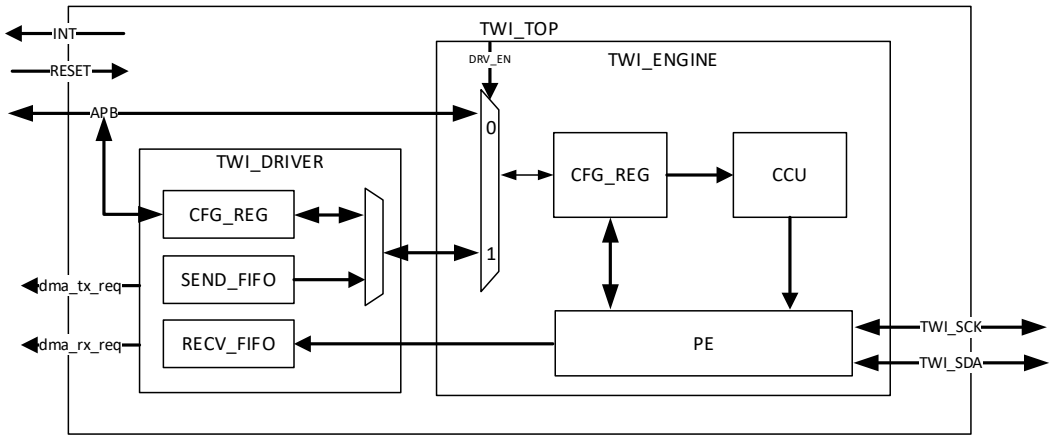
The TWI has the following features:

- Up to 9 TWI controllers
  - 6 TWI controllers in CPUX domain: TWI0, TWI1, TWI2, TWI3, TWI4, and TWI5
  - 3 TWI controllers in CPUS domain: S_TWI0, S_TWI1, and S_TWI2
- Compliant with I2C bus standard
- 7-bit and 10-bit device addressing modes
- Standard mode (up to 100 Kbit/s) and fast mode (up to 400 Kbit/s)
- Supports general call and start byte
- Master mode supports the following:
  - Bus arbitration in the case of multiple master devices
  - Clock synchronization and bit and byte waiting
  - Packet transmission and DMA
- Slave mode supports Interrupt on address detection

8.14.2 Block Diagram

the following figure shows the block diagram of TWI.

Figure 8-57 TWI Block Diagram



TWI contains the following sub-blocks:

Table 8-39 TWI Sub-blocks

Sub-block	Description
RESET	Module reset signal
INT	Module output interrupt signal
CFG_REG	Module configuration register in TWI
PE	Packet encoding/decoding
CCU	Module clock controller unit
SEND_FIFO	The register address bytes and the written data bytes are buffered in SEND_FIFO
RECV_FIFO	The read data bytes are buffered in RECV_FIFO

The controller includes TWI engine and TWI driver. Each time the TWI engine sends a START signal, a STOP signal, or a BYTE data, or a corresponding ACK, the TWI engine will generate an interrupt, and wait for the CPU to process and clear the interrupt before the next START, STOP, or BYTE, ACK transmission can be performed. Therefore, when a device communication is completed, many interrupts will be generated, and the CPU needs to wait for the previous interrupt before it can configure the next one. The TWI driver defines each communication with the device as a packet transmission. The CPU can directly configure the slave address, register address and data transmission for one or more package transmissions without waiting for interruption, then start the TWI driver, and the TWI driver can control the TWI engine to complete a pre-configured communication, and report an interrupt to the CPU after completion.

8.14.3 Functional Description

8.14.3.1 External Signals

The following table describes the external signals of the TWI. The TWIn-SCK and TWIn-SDA are bidirectional I/O, when the TWI is configured as a master device, the TWIn-SCK is an output pin; when the TWI is configurable as a slave device, the TWIn-SCK is an input pin. When using TWI, the corresponding PADs are selected as TWI function via section 8.6 GPIO.

Table 8-40 TWI External Signals

Signal Name	Description	Type
TWI0-SCK	TWI0 Serial Clock Signal	I/O
TWI0-SDA	TWI0 Serial Data Signal	I/O
TWI1-SCK	TWI1 Serial Clock Signal	I/O
TWI1-SDA	TWI1 Serial Data Signal	I/O
TWI2-SCK	TWI2 Serial Clock Signal	I/O
TWI2-SDA	TWI2 Serial Data Signal	I/O
TWI3-SCK	TWI3 Serial Clock Signal	I/O
TWI3-SDA	TWI3 Serial Data Signal	I/O
TWI4-SCK	TWI4 Serial Clock Signal	I/O
TWI4-SDA	TWI4 Serial Data Signal	I/O
TWI5-SCK	TWI5 Serial Clock Signal	I/O
TWI5-SDA	TWI5 Serial Data Signal	I/O
S-TWI0-SCK	S-TWI0 Serial Clock Signal	I/O
S-TWI0-SDA	S-TWI0 Serial Data Signal	I/O
S-TWI1-SCK	S-TWI1 Serial Clock Signal	I/O
S-TWI1-SDA	S-TWI1 Serial Data Signal	I/O
S-TWI2-SCK	S-TWI2 Serial Clock Signal	I/O
S-TWI2-SDA	S-TWI2 Serial Data Signal	I/O

8.14.3.2 Clock Sources

Each TWI controller has an input clock source. The following table describes the clock sources for TWI. After selecting a proper clock, users must open the gating of TWI and release the corresponding reset bit.

For more details on the clock setting, configuration, and gating information, see section 2.6 Clock Controller Unit (CCU) and section 2.12 Power Reset Clock Management (PRCM).

Table 8-41 TWI Clock Sources

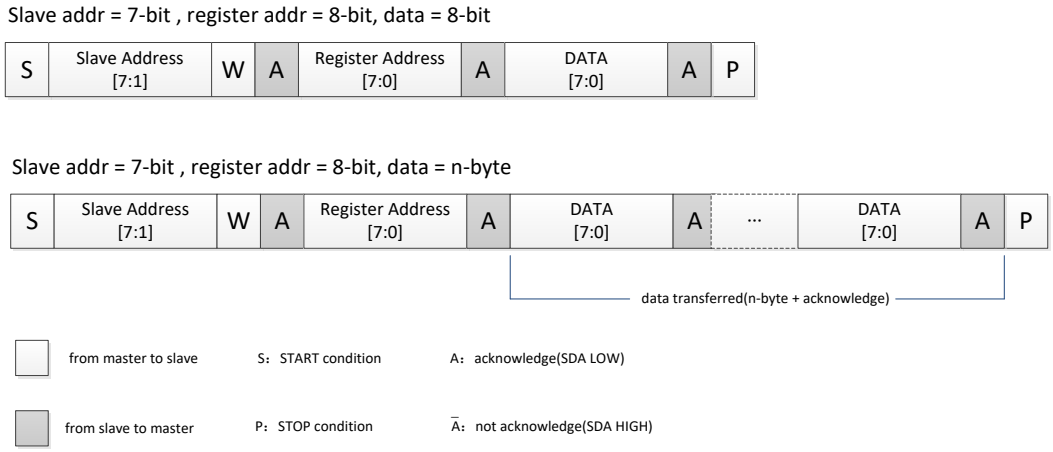
Clock Sources	Descsription	Module
APB1 Bus	TWI clock source. Refer to CCU for details on APB1.	CCU
APBS1 Bus	S_TWI clock source. Refer to PRCM for details on APBS1.	PRCM

8.14.3.3 Write/Read Timing in Standard and Extended Addressing Mode

This section is the 7-bit/10-bit addressing mode of the entire TWI protocol to read and write device registers. It can be achieved by directly using the TWI engine or using the TWI driver to control the TWI engine.

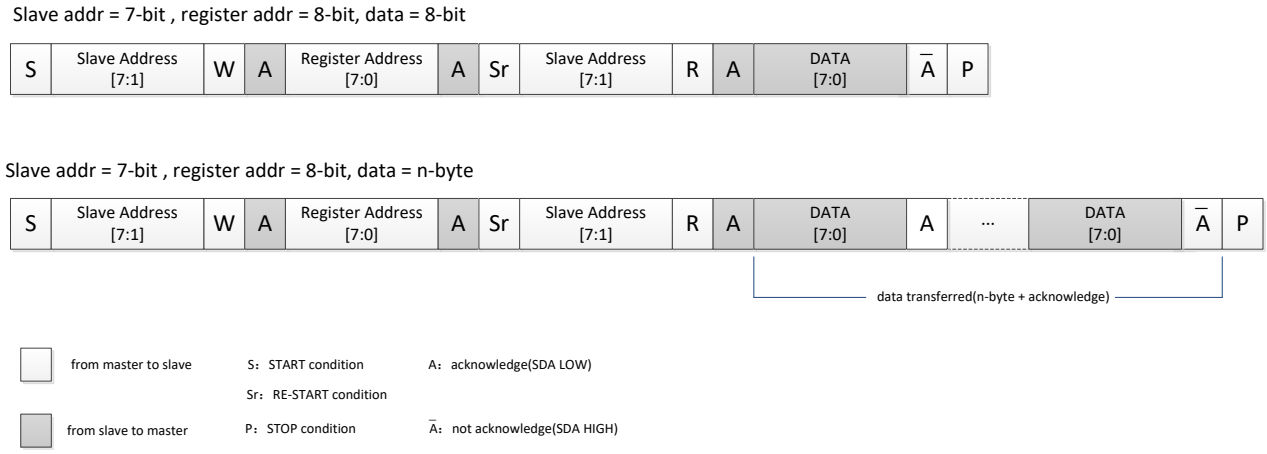
The following figure describes the write timing in 7-bit standard addressing mode.

Figure 8-58 Write Timing in 7-bit Standard Addressing Mode



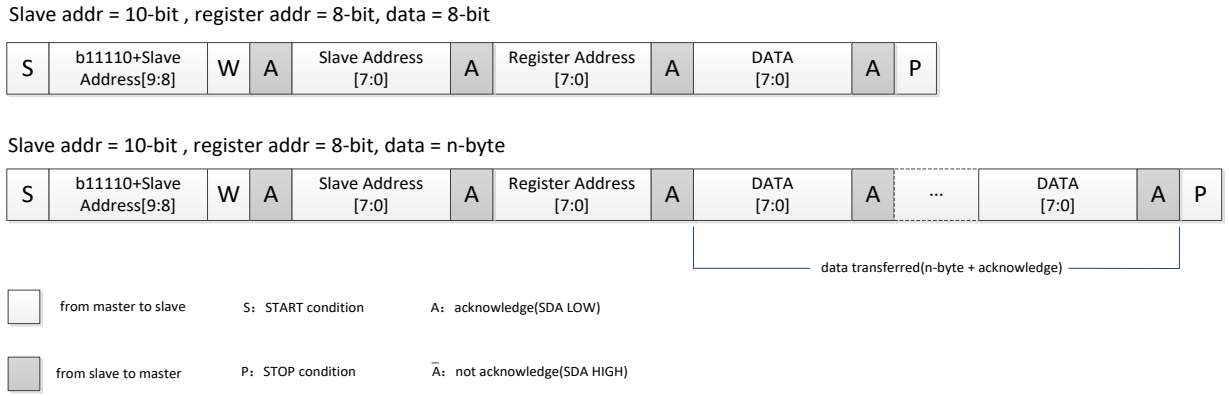
The following figure describes the read timing in 7-bit standard address mode.

Figure 8-59 Read Timing in 7-bit Standard Addressing Mode



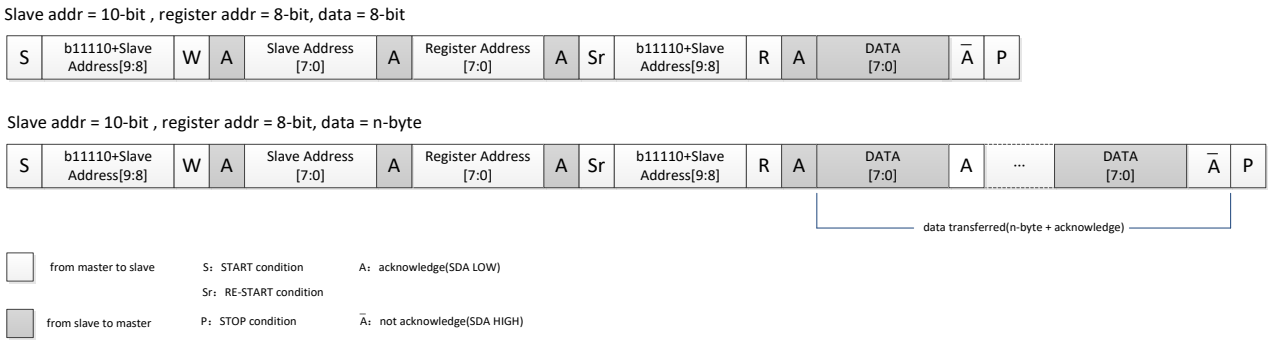
The following figure describes the write timing in 10-bit extended address mode.

Figure 8-60 Write Timing in 10-bit Extended Addressing Mode



The following figure describes the read timing in 10-bit extended address mode.

Figure 8-61 Read Timing in 10-bit Extended Addressing Mode



8.14.3.4 Write/Read Packet Transmission of TWI Driver

The TWI driver is only supported for master mode. When the TWI works in master mode, the TWI driver drives the TWI engine for one or more packet transmission instead of the CPU host. Packet transmission is defined in the following figures. The register address bytes and the written data bytes are buffered in SEND_FIFO, the read data bytes are buffered in RECV_FIFO.

Figure 8-62 TWI Driver Write Packet Transmission

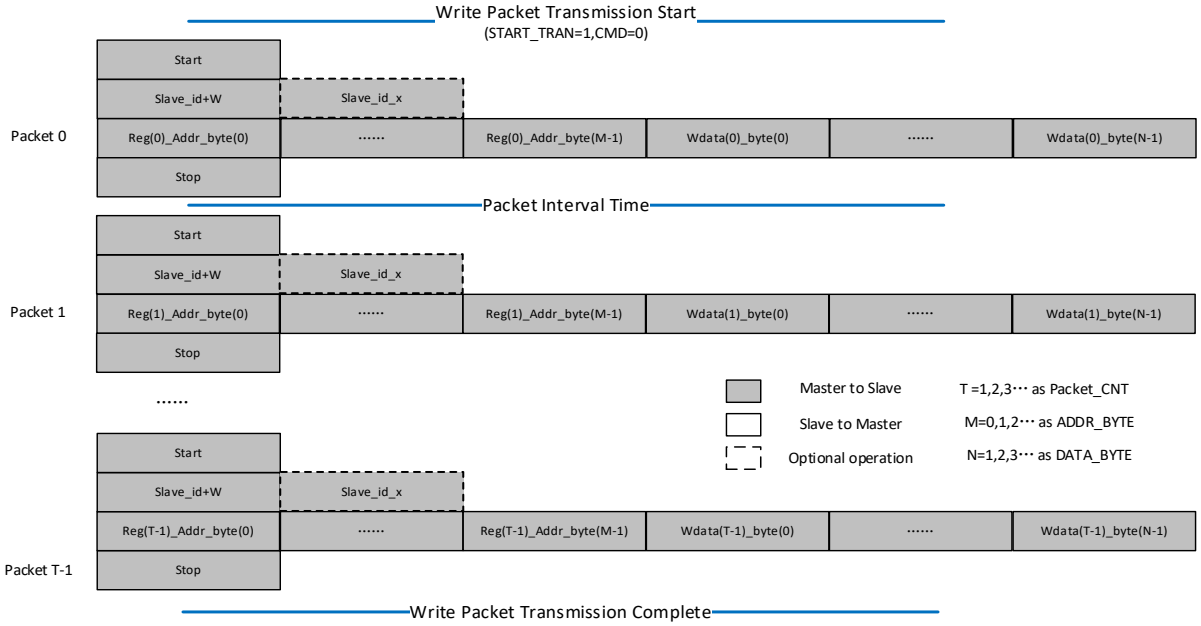
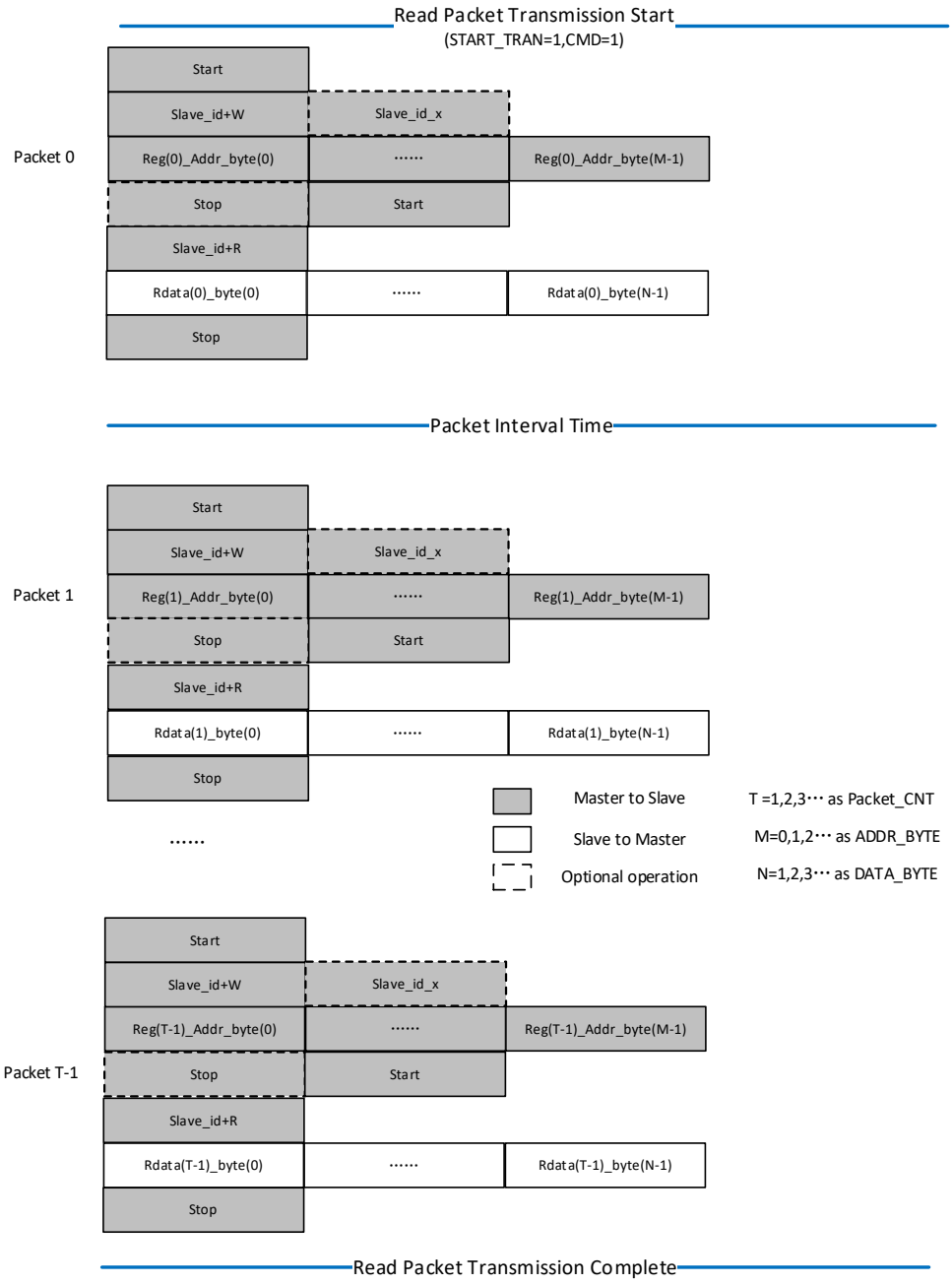


Figure 8-63 TWI Driver Read Packet Transmission



8.14.3.5 Master and Slave Mode of TWI Engine

In Master mode, the CPU host controls the TWI engine by writing command and data to its registers. The TWI engine transmits an interrupt to CPU when each time a byte transfer is done or a START/STOP command is detected. The CPU host can poll the status register if the interrupt mechanism is not disabled by the CPU host.

When the CPU host wants to start a bus transfer, it initiates a bus START to enter the master mode by setting TWI_CNTR[M_STA] to high. The TWI engine will assert the INT line and TWI_CNTR[INT_FLAG] to indicate a completion for the START command and each consequent

byte transfer. At each interrupt, the CPU host needs to check the current state by the TWI_STAT register. A transfer must conclude with the STOP command by setting TWI_CNTR[M_STP] to high.

In Slave mode, the TWI engine also constantly samples the bus and look for its own slave address during addressing cycles. Once a match is found, it is addressed, and the TWI engine interrupts the CPU host with the corresponding status. Upon request, the CPU host should read the status, read/write the TWI_DATA register, and set the TWI_CNTR register. After each byte transfer, a slave device always stops the operation of the remote master by holding the next low pulse on the SCL line until the CPU host responds to the status of the previous byte transfer or START command.

8.14.3.6 Generation of Repeated Start

After the data transfer, if the master still requires the bus, it can signal another Start followed by another slave address without signaling a Stop.

8.14.3.7 Programming State Diagram

Figure 8-64 shows the TWI programming state diagram. For the value between two states, see the TWI_STAT register in section 8.14.6.5.

M_SEND_S: master sends START signal;

M_SEND_ADDR: master sends slave address;

M_SEND_XADD: master sends slave extended address;

M_SEND_SR: master repeated start;

M_SEND_DATA: master sends data;

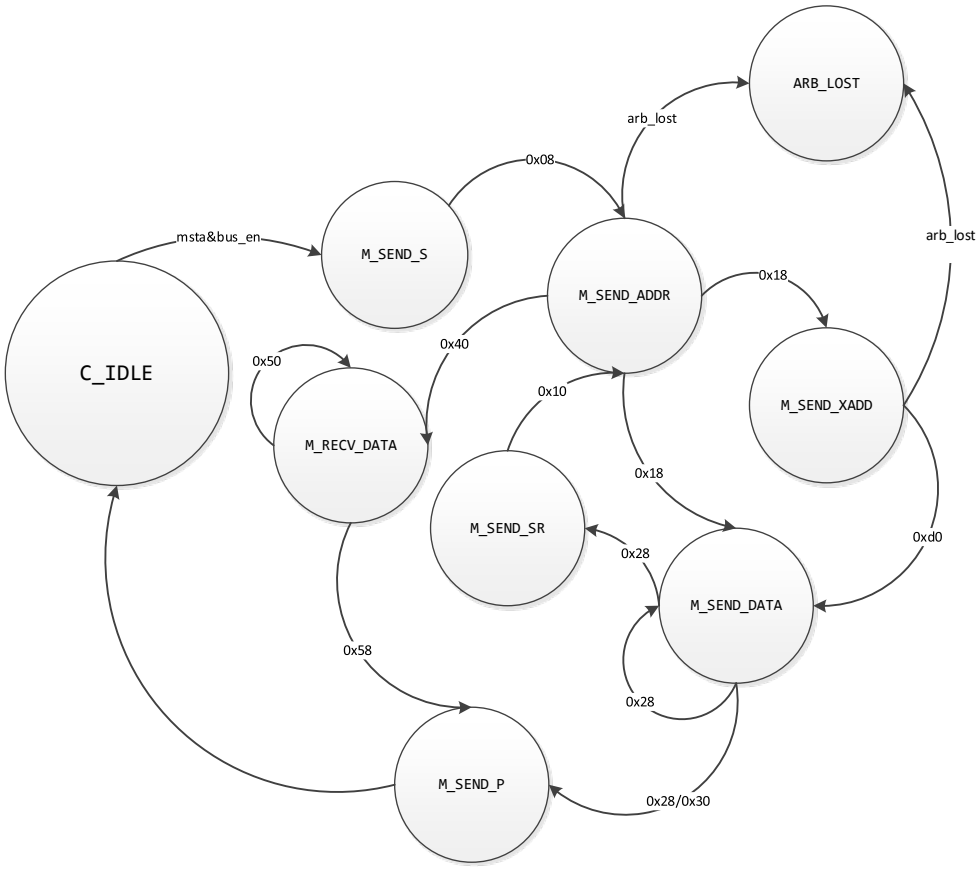
M_SEND_P: master sends STOP signal;

M_RECV_DATA: master receives data;

ARB_LOST: Arbitration lost;

C_IDLE: Idle.

Figure 8-64 TWI Programming State Diagram



8.14.4 Programming Guidelines

The TWI controller operates in an 8-bit data format. The data on the TWI_SDA line is always 8 bits long. At first, the TWI controller sends a start condition. When in the addressing formats of 7-bit, the TWI sends out an 8-bit message which includes 7 MSB slave address and 1 LSB read/write flag. The least significant of the salve address indicates the direction of transmission. When the TWI works in 10-bit slave address mode, the operation will be divided into two steps, for details on the operation, refer to register description in Section 8.14.6.1 and 8.14.6.2.

The following takes the TWI module in the CPUX domain as an example.

8.14.4.1 Initialization for TWI Engine

To initialize the TWI engine, perform the following steps:

Step 1 Configure corresponding GPIO multiplex function as TWI mode.

Step 2 For TWIn, set TWI_BGR_REG[TWIn_GATING] in CCU module to 0 to close TWIn clock.

Step 3 For TWIn, set TWI_BGR_REG[TWIn_RST] in CCU module to 0, then set to 1 to reset TWIn.

Step 4 For TWIn, set TWI_BGR_REG[TWIn_GATING] in CCU module to 1 to open TWIn clock.

Step 5 Configure TWI_CCR[CLK_M] and TWI_CCR[CLK_N] to get the needed rate (The clock source of TWI is from APB1).

Step 6 Configure TWI_CNTR[BUS_EN] and TWI_CNTR[A_ACK], when using interrupt mode, set TWI_CNTR[INT_EN] to 1, and register the system interrupt through GIC module. In slave mode, configure TWI_ADDR and TWI_XADDR registers to finish TWI initialization configuration.

8.14.4.2 Writing Data Operation for TWI Engine

To write data to the device, perform the following steps:

Step 1 Clear TWI_EFR register, and configure TWI_CNTR[M_STA] to 1 to transmit the START signal.

Step 2 After the START signal is transmitted, the first interrupt is triggered, then write device ID to TWI_DATA (For a 10-bit device ID, firstly write the first byte ID, secondly write the second byte ID in the next interrupt).

Step 3 The Interrupt is triggered again after device ID transmission completes, write device data address to be read to TWI_DATA (For a 16-bit address, firstly write the first-byte address, secondly write the second-byte address).

Step 4 Interrupt is triggered after data address transmission completes, write data to be transmitted to TWI_DATA (For consecutive write data operation, every byte transmission completion triggers interrupt, during interrupt write the next byte data to TWI_DATA).

Step 5 After transmission completes, write TWI_CNTR[M_STP] to 1 to transmit the STOP signal and end this write-operation.

8.14.4.3 Reading Data Operation for TWI Engine

To read data from the device, perform the following steps:

Step 1 Clear TWI_EFR register, and set TWI_CNTR[A_ACK] to 1, and configure TWI_CNTR[M_STA] to 1 to transmit the START signal.

Step 2 After the START signal is transmitted, the first interrupt is triggered, then write device ID to TWI_DATA (For a 10-bit device ID, firstly write the first-byte ID, secondly write the second-byte ID in the next interrupt).

Step 3 The Interrupt is triggered again after device ID transmission completes, write device data address to be read to TWI_DATA (For a 16-bit address, firstly write the first-byte address, secondly write the second-byte address).

Step 4 The Interrupt is triggered after data address transmission completes, write TWI_CNTR[M_STA] to 1 to transmit new START signal, and after interrupt triggers, write device ID to TWI_DATA to start read-operation.

Step 5 After device address transmission completes, each receive completion will trigger an interrupt, in turn, read TWI_DATA to get data, when receiving the previous interrupt of the last byte data, clear [A_ACK] to stop acknowledge signal of the last byte.

Step 6 Write TWI_CNTR[M_STP] to 1 to transmit the STOP signal and end this read-operation.

8.14.4.4 Initialization for TWI Driver

To initialize the TWI driver, perform the following steps:

Step 1 Configure corresponding GPIO multiplex function as TWI mode.

Step 2 For TWIn, set TWI_BGR_REG[TWIn_GATING] in CCU module to 0 to close TWIn clock.

Step 3 For TWIn, set TWI_BGR_REG[TWIn_RST] in CCU module to 0, then set to 1 to reset TWIn.

Step 4 For TWIn, set TWI_BGR_REG[TWIn_GATING] in CCU module to 1 to open TWIn clock.

Step 5 Set TWI_DRV_CTRL[TWI_DRV_EN] to 1 to enable the TWI driver.

Step 6 Configure TWI_DRV_BUS_CTRL[CLK_M] and TWI_DRV_BUS_CTRL[CLK_N] to get the needed rate (The clock source of TWI is from APB1).

Step 7 Set TWI_DRV_CTRL[RESTART_MODE] to 0 and [READ_TRAN_MODE] to 1, set TWI_DRV_INT_CTRL[TRAN_COM_INT_EN] to 1.

Step 8 When using DMA for data transmission, set TWI_DRV_DMA_CFG[DMA_RX_EN] and TWI_DRV_DMA_CFG[DMA_TX_EN] to 1, and configure TWI_DRV_DMA_CFG[RX_TRIG] and TWI_DRV_DMA_CFG[TX_TRIG] to set the thresholds of RXFIFO and TXFIFO.

8.14.4.5 Writing Packet Transmission for TWI Driver

To write package to the device, perform the following steps:

Step 1 Configure TWI_DRV_SLV[SLV_ID] to set the device ID, and configure TWI_DRV_SLV[CMD] to 0 to set the write operation.

Step 2 Configure TWI_DRV_FMT[ADDR_BYTE] according to the address width of the device register, and TWI_DRV_FMT[DATA_BYTE] according to the written data count in a packet.

Step 3 Configure TWI_DRV_CFG[PACKET_CNT] to set the written packet number.

Step 4 Configure DMA channel, including TWI TXFIFO, device register address, and the written data.

Step 5 Set [START_TRAN] to 1 to start TWI Driver transmission.

Step 6 When TWI driver transmission completes, the interrupt is triggered, it indicates that the write packet transmission ends.

8.14.4.6 Reading Packet Transmission for TWI Driver

Step 1 To read package from the device, perform the following steps:

Step 2 Configure TWI_DRV_SLV[SLV_ID] to set the device ID, and configure TWI_DRV_SLV[CMD] to 1 to set the read operation.

Step 3 Configure TWI_DRV_FMT[ADDR_BYTE] according to the address width of the device register, and TWI_DRV_FMT[DATA_BYTE] according to the read data count in a packet.

Step 4 Configure TWI_DRV_CFG[PACKET_CNT] to set the read packet number.

Step 5 Configure DMA channel, including TWI TXFIFO, TWI RXFIFO, device register address and the read data.

Step 6 Set [START_TRAN] to 1 to start TWI Driver transmission.

Step 7 When TWI driver transmission completes, the interrupt is triggered, it indicates that the read packet transmission ends.

8.14.5 Register List

Module Name	Base Address	Comments
TWI0	0x0250 2000	
TWI1	0x0250 2400	TWI1 register is the same with TWI0.
TWI2	0x0250 2800	TWI2 register is the same with TWI0.
TWI3	0x0250 2C00	TWI3 register is the same with TWI0.
TWI4	0x0250 3000	TWI4 register is the same with TWI0.
TWI5	0x0250 3400	TWI5 register is the same with TWI0.
S_TWI0	0x0708 1400	R-TWI0 register is the same with TWI0.
S_TWI1	0x0708 1800	R-TWI1 register is the same with TWI0.
S_TWI2	0x0708 1C00	R-TWI2 register is the same with TWI0.

Register Name	Offset	Description
TWI_ADDR	0x0000	TWI Slave Address Register
TWI_XADDR	0x0004	TWI Extended Slave Address Register
TWI_DATA	0x0008	TWI Data Byte Register
TWI_CNTR	0x000C	TWI Control Register
TWI_STAT	0x0010	TWI Status Register
TWI_CCR	0x0014	TWI Clock Control Register
TWI_SRST	0x0018	TWI Software Reset Register
TWI_EFR	0x001C	TWI Enhance Feature Register
TWI_LCR	0x0020	TWI Line Control Register

8.15 PWM

8.15.1 Overview

The Pulse Width Modulation (PWM) module can output the configurable PWM waveforms and measure the external input waveforms.

The PWM has the following features:

- Up to 30 PWM channels and 4 PWM controllers
  - PWM0-[15:0] for PWM0 controller
  - PWM1- [3:0] for PWM1 controller
  - S-PWM0-[1:0] for S_PWM0 controller
  - MCU-PWM0-[7:0] for MCU_PWM0 controller
- Maximum 16 independent PWM channels for PWM controller
  - Supports PWM continuous mode output
  - Supports PWM pulse mode output, and the pulse number is configurable
  - Output frequency range:
    - 0 to 24 MHz (when the clock source is DCXO24M)
    - 0 to 100 MHz (when the clock source is APB1 clock)
  - Various duty-cycle: 0% to 100%
  - Minimum resolution: 1/65536
- Maximum 8 complementary pairs output
  - The pairing methods for each controller are as follows. The components are internal PWM channels:
    - Maximum 8 pairs for PWM0:

      PWM0-0 + PWM0-1, PWM0-2 + PWM0-3, PWM0-4 + PWM0-5, PWM0-6 + PWM0-7, PWM0-8 + PWM0-9, PWM0-10 + PWM0-11, PWM0-12 + PWM0-13, PWM0-14 + PWM0-15
    - Maximum 2 pairs for PWM1:

      PWM1-0+PWM1-1, PWM1-2+PWM1-3
    - Maximum 1 pair for S_PWM0:

      S-PWM0-0+ S-PWM0-1
    - Maximum 4 pairs for MCU_PWM0:

      MCU-PWM0-0+ MCU-PWM0-1, MCU-PWM0-2+ MCU-PWM0-3, MCU-PWM0-4+ MCU-PWM0-5, MCU-PWM0-6+ MCU-PWM0-7
  - Supports dead-zone generator, and the dead-zone time is configurable

- Maximum 4 group of PWM channel output for controlling stepping motors
  - Supports any plural channels to form a group, and output the same duty-cycle pulse
  - In group mode, the relative phase of the output waveform for each channel is configurable
- Maximum 16 channels capture input
  - Supports rising edge detection and falling edge detection for input waveform pulse
  - Supports pulse-width measurement for input waveform pulse

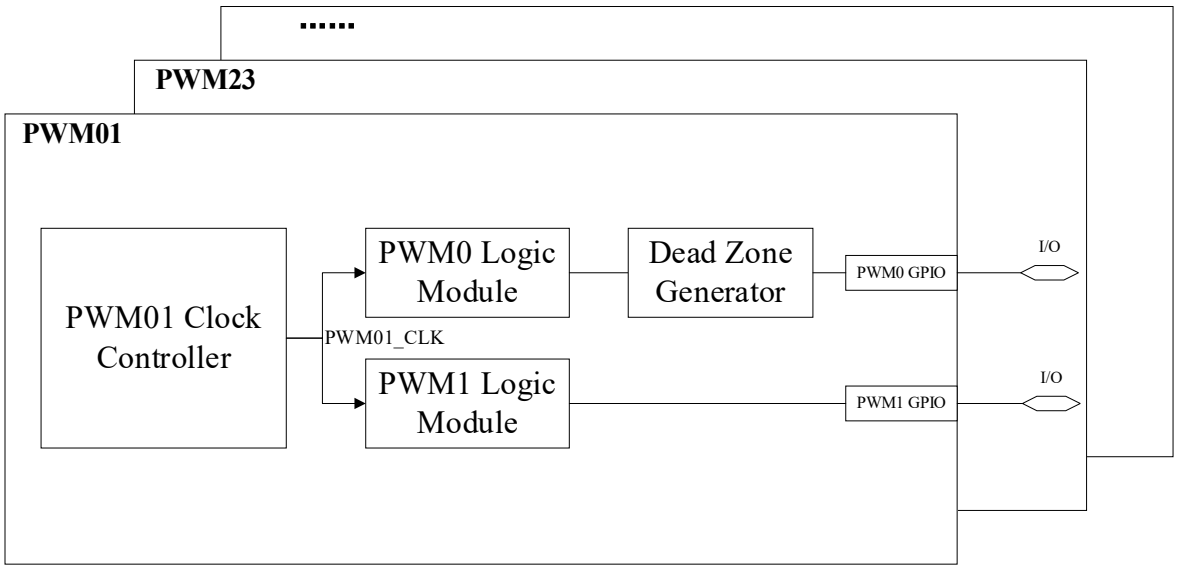
NOTE

For the corresponding relationship between the channels of each PWM controller and the external signals, please refer to section 8.15.3.1 External Signals.

8.15.2 Block Diagram

The PWM includes multi PWM channels. Each channel can generate different PWM waveform by the independent counter and duty-ratio configuration register. Each PWM pair shares one group of clock and dead-zone generator to generate PWM waveform.

Figure 8-65 PWM Block Diagram



Each PWM pair consists of 1 clock module, 2 timer logic module, and 1 programmable dead-zone generator.

8.15.3 Functional Description

8.15.3.1 External Signals

The following table describes the external signals of the PWM.

Table 8-42 PWM External Signals

Signal	Description	Type
PWM0		
PWM0-0	PWM channel0 in PWM0	I/O
PWM0-1	PWM channel1 in PWM0	I/O
PWM0-2	PWM channel2 in PWM0	I/O
PWM0-3	PWM channel3 in PWM0	I/O
PWM0-4	PWM channel4 in PWM0	I/O
PWM0-5	PWM channel5 in PWM0	I/O
PWM0-6	PWM channel6 in PWM0	I/O
PWM0-7	PWM channel7 in PWM0	I/O
PWM0-8	PWM channel8 in PWM0	I/O
PWM0-9	PWM channel9 in PWM0	I/O
PWM0-10	PWM channel10 in PWM0	I/O
PWM0-11	PWM channel11 in PWM0	I/O
PWM0-12	PWM channel12 in PWM0	I/O
PWM0-13	PWM channel13 in PWM0	I/O
PWM0-14	PWM channel14 in PWM0	I/O
PWM0-15	PWM channel15 in PWM0	I/O
PWM1		
PWM1-0	PWM channel0 in PWM1	I/O
PWM1-1	PWM channel1 in PWM1	I/O
PWM1-2	PWM channel2 in PWM1	I/O
PWM1-3	PWM channel3 in PWM1	I/O
S_PWM0		
S-PWM0-0	PWM channel0 in S_PWM0	I/O
S-PWM0-1	PWM channel1 in S_PWM0	I/O
MCU_PWM0		
MCU-PWM0-0	PWM channel0 in MCU_PWM0	I/O
MCU-PWM0-1	PWM channel1 in MCU_PWM0	I/O
MCU-PWM0-2	PWM channel2 in MCU_PWM0	I/O
MCU-PWM0-3	PWM channel3 in MCU_PWM0	I/O
MCU-PWM0-4	PWM channel4 in MCU_PWM0	I/O
MCU-PWM0-5	PWM channel5 in MCU_PWM0	I/O
MCU-PWM0-6	PWM channel6 in MCU_PWM0	I/O
MCU-PWM0-7	PWM channel7 in MCU_PWM0	I/O

8.15.3.2 Clock Sources

The following table describes the clock sources of the PWM controllers.

Table 8-43 PWM clock sources

PWM	Clock Sources	Description	Module
PWM0 PWM1	HOSC	24 MHz, external clock.	CCU
	APB1_CLK	24 MHz, PWM bus clock.	
S_PWM0 MCU_PWM0	CLK24M	By default, CLK24M is 24 MHz.	PRCM
	CLK32K	By default, CLK32K is 32 kHz.	
	CLK_RC	By default, CLK_RC is 16 MHz.	

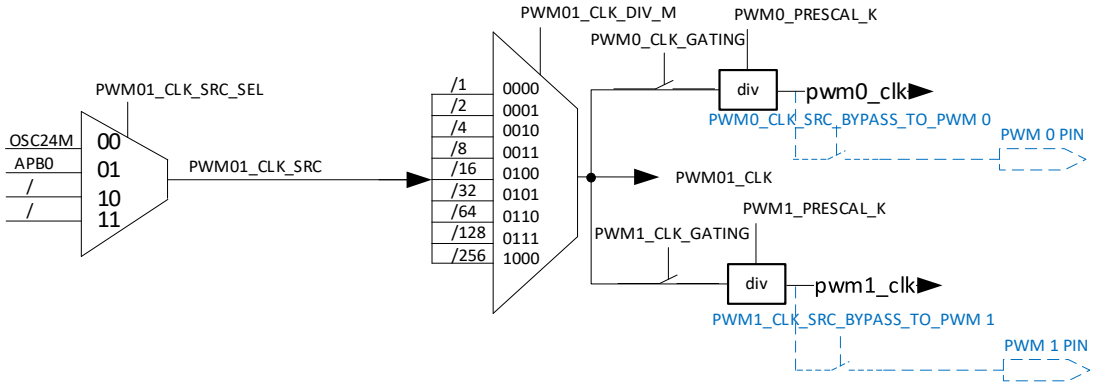
8.15.3.3 Typical Application

- Suitable for display device, such as LCD
- Suitable for electric motor control

8.15.3.4 Clock Controller

Using PWM01 as an example. The other PWM pairs are the same as PWM01.

Figure 8-66 PWM01 Clock Controller Diagram



The clock controller of each PWM pair includes clock source select (PWM01_CLK_SRC), 1-256 scaler (PWM01_CLK_DIV_M). Each PWM channel has the secondary frequency division (PWM_PRESCAL_K), clock source bypass (PWMx_CLK_BYPASS) and clock switch (PWMx_CLK_GATING).

The clock sources have HOSC and APB0. The HOSC comes from the external high-frequency oscillator; the APB0 is APB0 bus clock.

The bypass function of the clock source is that the clock source directly accesses PWM output, the PWM output waveform is the waveform of the clock controller output. The BYPASS gridlines in the

above figure indicate the bypass function of the clock source, see Figure 8-67 for the details about implement. At last, the output clock of the clock controller is sent to the PWM logic module.

8.15.3.5 PWM Output

Taking PWM01 as an example, Figure 8-67 indicates the PWM01 output logic diagram. The logic diagrams of other PWM pairs are the same as PWM01.

The timer logic module of PWM consists of one 16-bit up-counter (PCNTR) and three 16-bit parameters (PWM_ENTIRE_CYCLE, PWM_ACT_CYCLE, PWM_COUNTER_START). The PWM_ENTIRE_CYCLE is used to control the PWM cycle, the PWM_ACT_CYCLE is used to control the duty-cycle, the PWM_COUNTER_START is used to control the output phase (multi-channel synchronization work requirements).

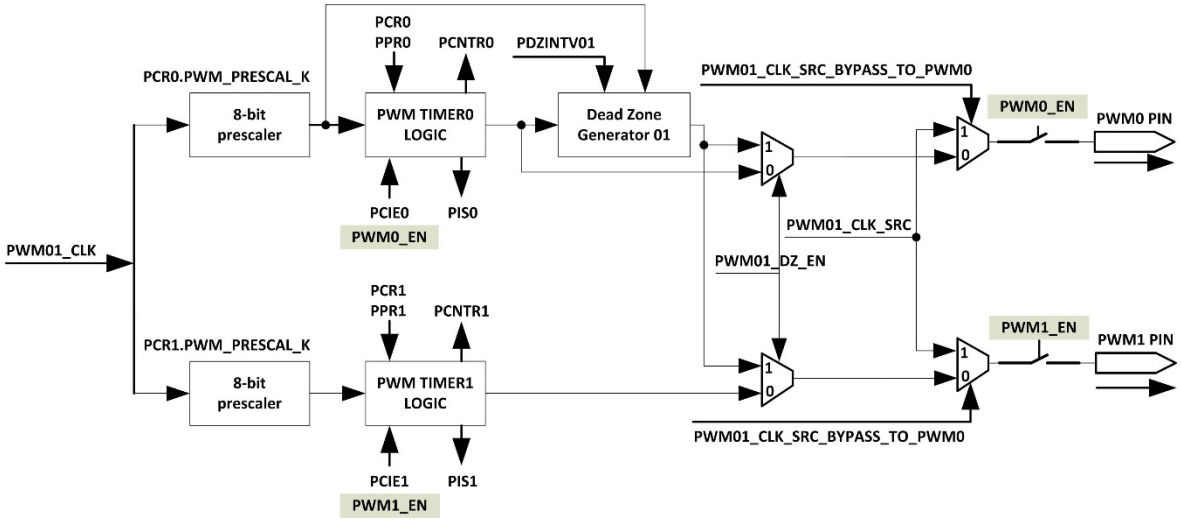
The PWM_ENTIRE_CYCLE and the PWM_ACT_CYCLE support the cache load, after PWM output is enabled, the register values of the PWM_ENTIRE_CYCLE and the PWM_ACT_CYCLE can be changed anytime, the changed value caches into the cache register. When the PCNTR counter outputs a period of PWM waveform, the value of the cache register can be updated for the PCNTR control. The purpose of the cache load is to avoid the unstable PWM output waveform with glitches when updating the values of the PWM_ENTIRE_CYCLE and PWM_ACT_CYCLE.

The PWM supports cycle and pulse waveform output.

Cycle mode: The PWM outputs the setting PWM waveform continually, that is, the output waveform is a continuous PWM square wave.

Pulse mode: After setting the PWM_PUL_NUM parameter, the PWM outputs (PWM_PULNUM+1) periods of PWM waveform, that is, the waveform with several pulses are output.

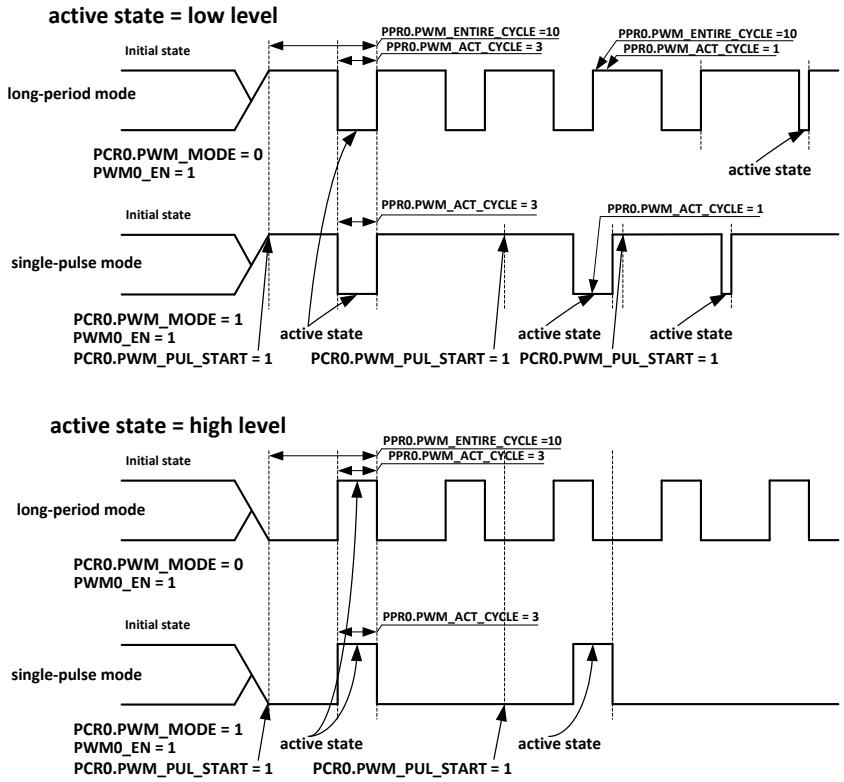
Figure 8-67 PWM01 Output Logic Module Diagram



8.15.3.6 Pulse Mode and Cycle Mode

The PWM output supports pulse mode and cycle mode. PWM in pulse mode outputs PCR[PWM_PUL_NUM] +1 cycles waveform, but PWM in cycle mode outputs continuous waveform. The following figure shows the PWM output waveform in pulse mode and cycle mode.

Figure 8-68 PWM0 Output Waveform in Pulse Mode and Cycle Mode



Each channel of the PWM module supports the PWM output of pulse mode and cycle mode, the active state of the PWM output waveform can be programmed to control.

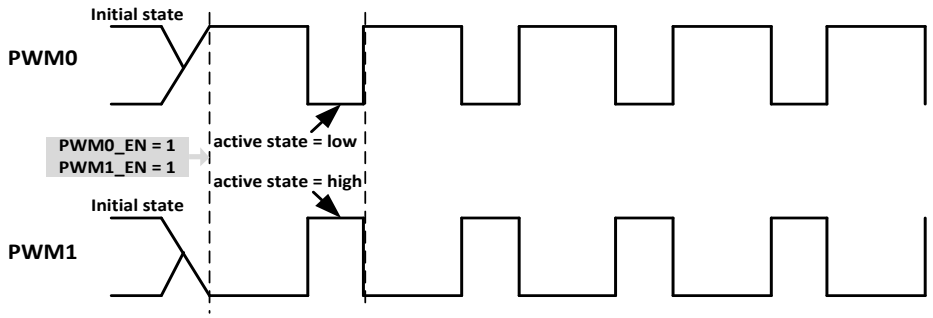
When PCR[PWM_MODE] is 0, the PWM0 outputs in cycle mode. When PCR[PWM_MODE] is 1, the PWM0 outputs in pulse mode.

Specifically, in pulse mode, after the PWM0 channel enabled, PCR[PWM_PUL_START] needs to be set to 1 when the PWM0 needs to output pulse waveform, after completed the output, PCR[PWM_PUL_START] can be cleared to 0 by hardware. The next setting 1 can be operated after PCR[PWM_PUL_START] is cleared.

8.15.3.7 Complementary Pair Output

Every PWM pair supports complementary pair output and PWM pair with dead-time. the following figure shows the complementary pair output of PWM01.

Figure 8-69 PWM01 Complementary Pair Output



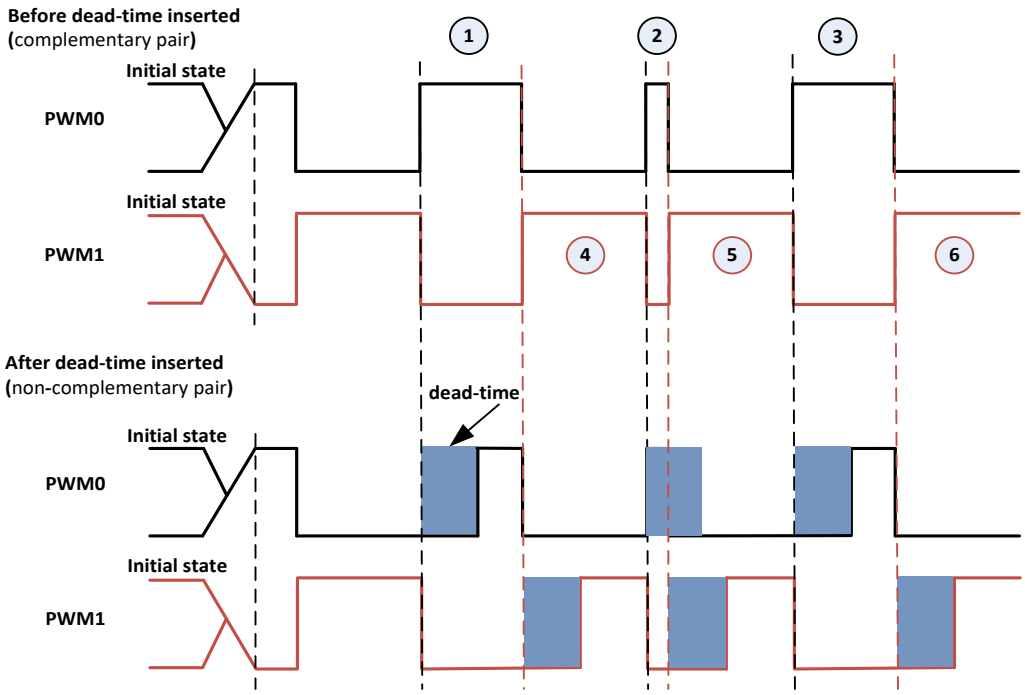
The complementary pair output needs to satisfy the following conditions:

- PWM0 and PWM1 have the same clock divider, frequency, duty-cycle, and phase
- PWM0 and PWM1 have an opposite active state
- Enable the clock gating of PWM0 and PWM1 at the same time
- Enable the waveform output of PWM0 and PWM1 at the same time

8.15.3.8 Dead-time Generator

Every PWM pair has a programmable dead-time generator. When the dead-time function of the PWM pair enabled, the PWM01 output waveform is decided by PWM timer logic and DeadZone Generator. the following figure shows the output waveform.

Figure 8-70 Dead-time Output Waveform



The PWM waveform before the insertion of dead-time indicates a complementary waveform pair of non-inserted dead-time in Dead Zone Generator 01.

The PWM waveform after the insertion of dead-time indicates a non-complementary PWM waveform pair inserted dead-time in a complementary waveform pair of Dead Zone Generator 01. The PWM waveform pair at last outputs to PWM0 pin and PWM1 pin.

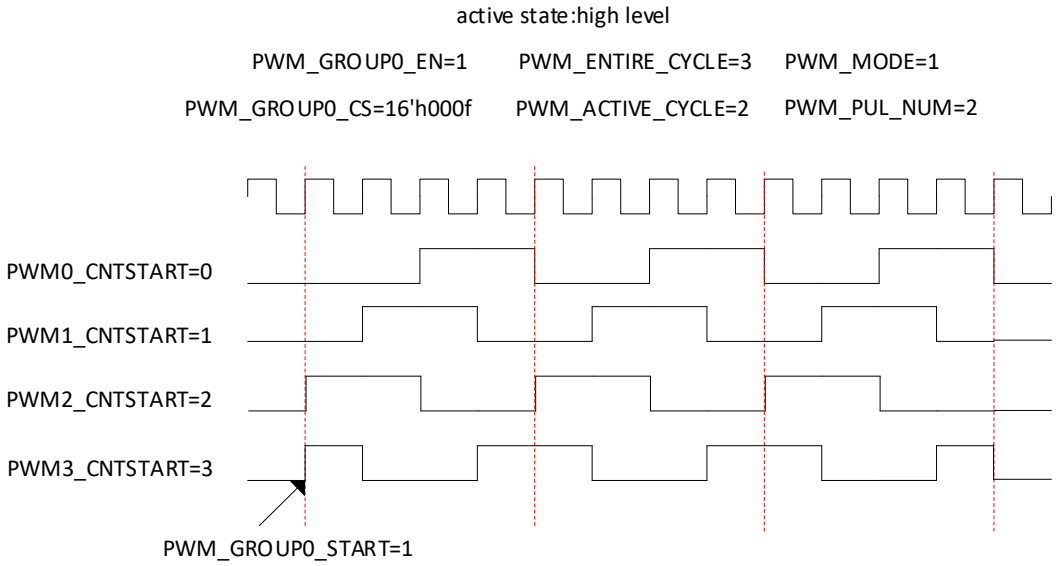
For the complementary pair of Dead Zone Generator 01, the principle of inserting dead-time is that to insert dead-time as soon as the rising edge came. If the high level time for mark② in the above figure is less than dead-time, then dead-time will override the high level. The setting of dead-time needs to consider the period and the duty-cycle of the output waveform. The dead-time formula is defined as follows:

Dead-time = (PWM01_CLK / PWM0_PRESCALE_K)$^{-1}$ * PDZINTV01

8.15.3.9 PWM Group Mode

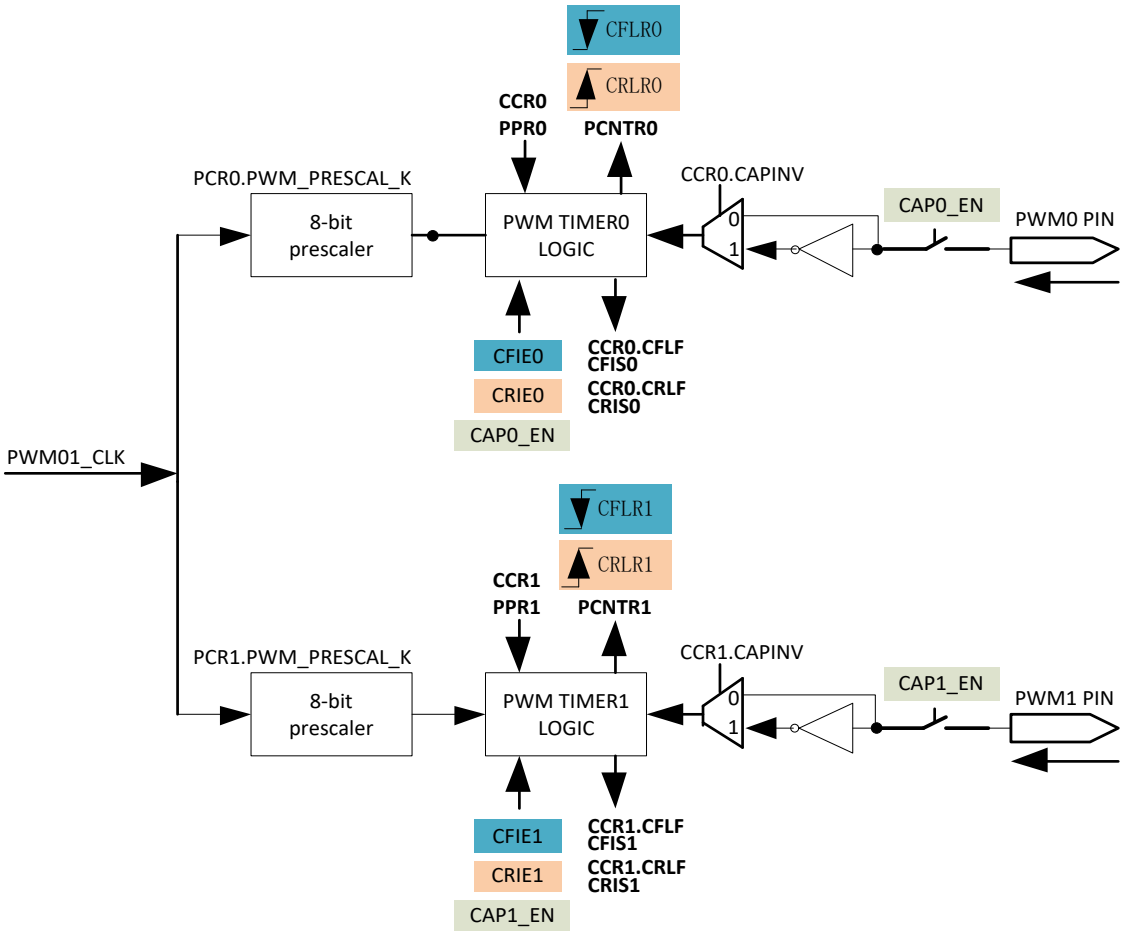
Taking PWM Group0 as an example. The same group of PWM channel is selected to work by PGR0.CS; the same PWM_ENTIRE_CYCLE, PWM_ACT_CYCLE are set by the same clock configuration; the different PWM_COUNTER_START can output PWM group signals with the same duty-cycle and the different phase.

Figure 8-71 Group 0-3 PWM Signal Output



8.15.3.10 Capture Input

Figure 8-72 PWM01 Capture Logic Module Diagram



Besides the timer logic module of every PWM channel generates PWM output, it can be used to capture the rising edge and the falling edge of the external clock. Using the PWM0 channel as an

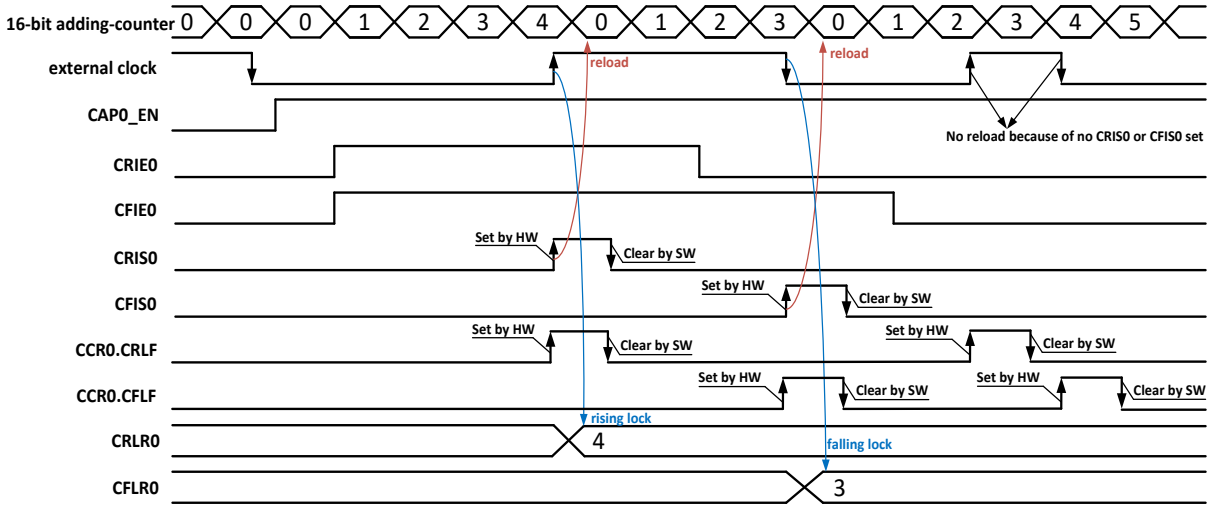
example, the PWM0 channel has one CFLR0 and one CRLR0 for capturing up-counter value on the falling edge and rising edge, respectively. You can calculate the period of the external clock by CFLR0 and CRLR0.

Thigh-level = (PWM01_CLK / PWM0_PRESCALE_K)$^{-1}$ * CRLR0

Tlow-level = (PWM01_CLK / PWM0_PRESCALE_K)$^{-1}$ * CFLR0

Tperiod = Thigh-level + Tlow-level

Figure 8-73 PWM0 Channel Capture Timing



When the capture input function of the PWM0 channel is enabled, the PCNTR of the PWM0 channel starts to work.

When the timer logic module of PWM0 captures a rising edge, the current value of the up-counter is locked to CRLR0 and CCR0[CRLF] is set to 1. If CRIE0 is 1, then CRIS0 is set to 1, the PWM0 channel sends interrupt requests, and the up-counter is loaded to 0 and continues to count. If CRIE0 is 0, the timer logic module of PWM0 captures a rising edge, CRIS0 cannot be set to 1, the up-counter is not loaded to 0.

When the timer logic module of PWM0 captures one falling edge, the current value of PCNTR is locked to CFLR0 and CCR0[CFLF] is set to 1. If CFIE0 is 1, then CFIS0 is set to 1, the PWM0 channel sends interrupt requests, and the up-counter is loaded to 0 and continues to count. If CFIE0 is 0, the timer logic module of PWM0 captures a falling edge, CFIS0 cannot be set to 1, the up-counter is not loaded to 0.

8.15.4 Programming Guidelines

The following working mode takes PWM01 as an example, other PWM pairs and PWM01 are consistent.

8.15.4.1 Configuring Clock

Step 1 PWM gating: When using PWM, write 1 to PCGR[PWMx_CLK_GATING].

Step 2 PWM clock source select: Set PCCR01[PWM01_CLK_SRC] to select HOSC or APB0 clock.

Step 3 PWM clock divider: Set PCCR01[PWM01_CLK_DIV_M] to select different frequency division coefficient (1/2/4/8/16/32/64/128/256).

Step 4 PWM clock bypass: Set PCGR[PWM_CLK_SRC_BYPASS_TO_PWM] to 1, output the PWM clock after the secondary frequency division to the corresponding PWM output pin.

Step 5 PWM internal clock configuration: Set PCR[PWM_PRESCAL_K] to select any frequency division coefficient from 1 to 256.

NOTE

For the channel of complementary output and group mode, firstly, set the same clock configurations (clock source selects APB0, clock division configures the same division factor); secondly, open clock gating at the same time; thirdly, configure PWM parameters; finally, enable PWM output at the same time to ensure each channel sync.

We suggest that the two channels of the same PWM pair cannot subject to two groups because of they have the same first level clock division and gating. If must allocate based on this way, the first level of clock division of the channel used by all groups needs to set to the same coefficient and open gating at the same time. And the total module needs to be reset when the group mode regroups.

8.15.4.2 Configuring PWM

Step 1 PWM mode: Set PCR[PWM_MODE] to select cycle mode or pulse mode, if pulse mode, PCR[PWM_PUL_NUM] needs to be configured.

Step 2 PWM active level: Set PCR[PWM_ACT_STA] to select a low level or high level.

Step 3 PWM duty-cycle: Configure PPR[PWM_ENTIRE_CYCLE] and PPR[PWM_ACT_CYCLE] after clock gating is opened.

Step 4 PWM starting/stoping phase: Configure PCNTR[PWM_COUNTER_START] after the clock gating is enabled and before the PWM is enabled. You can verify whether the configuration was successful by reading back PCNTR[PWM_COUNTER_STATUS].

Step 5 Enable PWM: Configure PER to select the corresponding PWM enable bit; when selecting pulse mode, PCR[PWM_PUL_START] needs to be enabled.

8.15.4.3 Configuring Deadzone

Step 1 Set initial value: set [PDZINTV01].

Step 2 Enable Deadzone: set [PWM01_DZ_CN].

8.15.4.4 Configuring Capture Input

Step 1 Enable capture: Configure CER to enable the corresponding channel.

Step 2 Capture mode: Configure CCR[CRLF] and CCR[CFLF] to select rising edge capture or falling edge capture, configure CCR[CAPINV] to select whether the input signal does reverse processing.

8.15.5 Register List

Module Name	Base Address
PWM0	0x0200 0C00
PMW1	0x0205 1000
S_PWM0	0x0702 0C00
MCU_PWM0	0x0710 3000

Register Name	Offset	Description
PIER	0x0000	PWM IRQ Enable Register
PISR	0x0004	PWM IRQ Status Register
CIER	0x0010	Capture IRQ Enable Register
CISR	0x0014	Capture IRQ Status Register
PCCR01	0x0020	PWM01 Clock Configuration Register
PCCR23	0x0024	PWM23 Clock Configuration Register
PCCR45	0x0028	PWM45 Clock Configuration Register
PCCR67	0x002C	PWM67 Clock Configuration Register
PCCR89	0x0030	PWM89 Clock Configuration Register
PCCRab	0x0034	PWMab Clock Configuration Register
PCCRcd	0x0038	PWMcd Clock Configuration Register
PCCRef	0x003C	PWMef Clock Configuration Register
PCGR	0x0040	PWM Clock Gating Register
PDZCR01	0x0060	PWM01 Dead Zone Control Register
PDZCR23	0x0064	PWM23 Dead Zone Control Register
PDZCR45	0x0068	PWM45 Dead Zone Control Register
PDZCR67	0x006C	PWM67 Dead Zone Control Register
PDZCR89	0x0070	PWM89 Dead Zone Control Register
PDZCRab	0x0074	PWMab Dead Zone Control Register
PDZCRcd	0x0078	PWMcd Dead Zone Control Register
PDZCRef	0x007C	PWMef Dead Zone Control Register
PER	0x0080	PWM Enable Register
PGR0	0x0090	PWM Group0 Register
PGR1	0x0094	PWM Group1 Register
PGR2	0x0098	PWM Group2 Register

8.16 SPI

8.16.1 Overview

The Serial Peripheral Interface (SPI) is a full-duplex, synchronous, four-wire serial communication interface between a CPU and SPI-compliant external devices. The SPI controller contains a 64 x 8 bits receiver buffer (RXFIFO) and a 64 x 8 bits transmit buffer (TXFIFO). It can work in master mode and slave mode.

The SPI has the following features:

- Three SPI interfaces:
  - SPI0 and SPI2 in CPUX Domain
  - S_SPI0 in CPUS Domain
- Multiple SPI modes:
  - Master mode and slave mode for standard SPI
  - Master mode for Dual-Output/Dual-Input SPI and Dual I/O SPI
  - Master mode for Quad-Output/Quad-Input SPI
  - Master mode for 3-wire SPI, with programmmable serial data frame length of 1 bit to 32 bits
- Maximum clock frequency: 100MHz
- TX/RX DMA slave interface
- 8-bit wide and 64-entry FIFO for both transmitting and receiving data
- 8-bit wide and 4-entry buffer for transmitting
- 8-bit wide and 128-entry buffer for receiving data
- Supports mode0, mode1, mode2, and mode3
- Polarity and phase of the Chip Select (SPI_SS) and SPI Clock (SPI_SCLK) are configurable

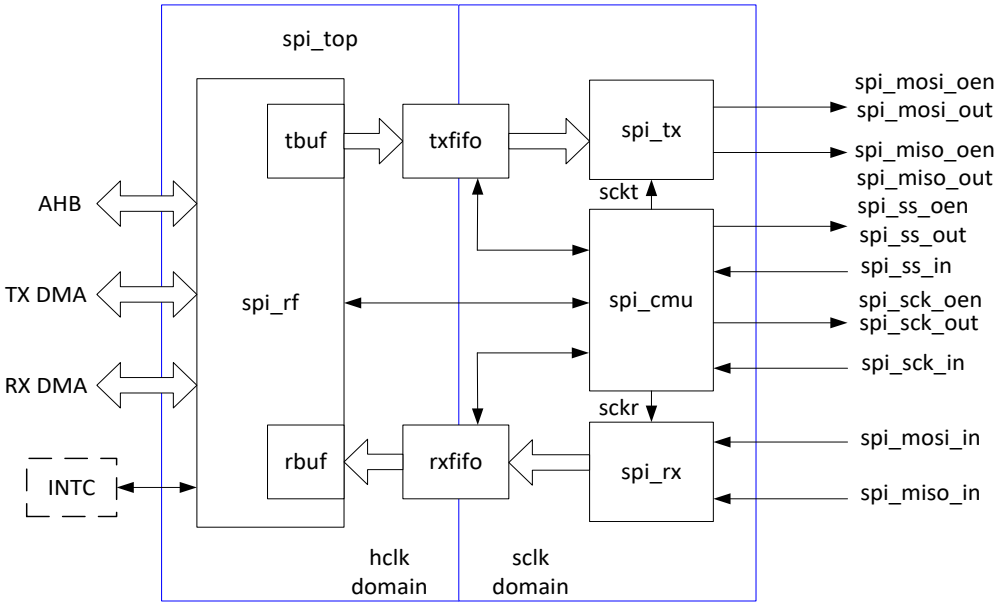
NOTE

This chapter only describes SPI0, SPI2, and S_SPI0. For detailed information of SPI1 (supports SPI mode and DBI mode), please refer to section 8.17 SPI_DBI.

8.16.2 Block Diagram

The following figure shows a block diagram of the SPI.

Figure 8-74 SPI Block Diagram



SPI contains the following sub-blocks:

Table 8-44 SPI Sub-blocks

Sub-block	Description
spi_rf	Responsible for implementing the internal register, interrupt, and DMA Request.
spi_tbuf	The data length transmitted from AHB to TXFIFO is converted into 8 bits, then the data is written into the RXFIFO.
spi_rbuf	The block is used to convert the RXFIFO data into the reading data length of AHB.
txfifo, rxfifo	The data transmitted from the SPI to the external serial device is written into the TXFIFO; the data received from the external serial device into SPI is pushed into the RXFIFO.
spi_cmu	Responsible for implementing SPI bus clock, chip select, internal sample, and the generation of transfer clock.
spi_tx	Responsible for implementing SPI data transfer, the interface of the internal TXFIFO, and status register.
spi_rx	Responsible for implementing SPI data receive, the interface of the internal RXFIFO, and status register.

8.16.3 Functional Description

8.16.3.1 External Signals

The following table describes the external signals of SPI. The MOSI and MISO are bidirectional I/O, when SPI is as a master device, the CLK and CS are the output pin; when SPI is as a slave device, the CLK and CS are the input pin. When using SPI, the corresponding PADs are selected as SPI function via section 8.6 GPIO.

Table 8-45 SPI External Signals

Signal Name	Description	Type
SPI0-CS[1:0]	SPI0 Chip Select Signal, Low Active	I/O
SPI0-CLK	SPI0 Clock Signal Provides serial interface timing.	I/O
SPI0-MOSI	SPI0 Master Data Out, Slave Data In	I/O
SPI0-MISO	SPI0 Master Data In, Slave Data Out	I/O
SPI0-WP	SPI0 Write Protect, Low Active Protects the memory area against all program or erase instructions. It also can be used for serial data input and output for SPI Quad Input or Quad Output mode.	I/O
SPI0-HOLD	SPI0 Hold Signal Pauses any serial communication with the device without deselecting or resetting it. It also can be used for serial data input and output for SPI Quad Input or Quad Output mode.	I/O
SPI2-CS0	SPI2 Chip Select Signal, Low Active	I/O
SPI2-CLK	SPI2 Clock Signal Provides serial interface timing.	I/O
SPI2-MOSI	SPI2 Master Data Out, Slave Data In	I/O
SPI2-MISO	SPI2 Master Data In, Slave Data Out	I/O
S-SPI0-CS0	S-SPI Chip Select Signal, Low Active	I/O
S-SPI0-CLK	S-SPI Clock Signal Provides serial interface timing.	I/O
S-SPI0-MOSI	S-SPI Master Data Out, Slave Data In	I/O
S-SPI0-MISO	S-SPI Master Data In, Slave Data Out	I/O

8.16.3.2 Clock Sources

Every SPI controller gets 5 different clock sources, users can select one of them to make SPI clock source. The following table describes the clock sources for SPI. For more details on the clock setting, configuration, and gating information, see section 2.6 Clock Controller Unit (CCU) and section 2.12 Power Reset Clock Management (PRCM).

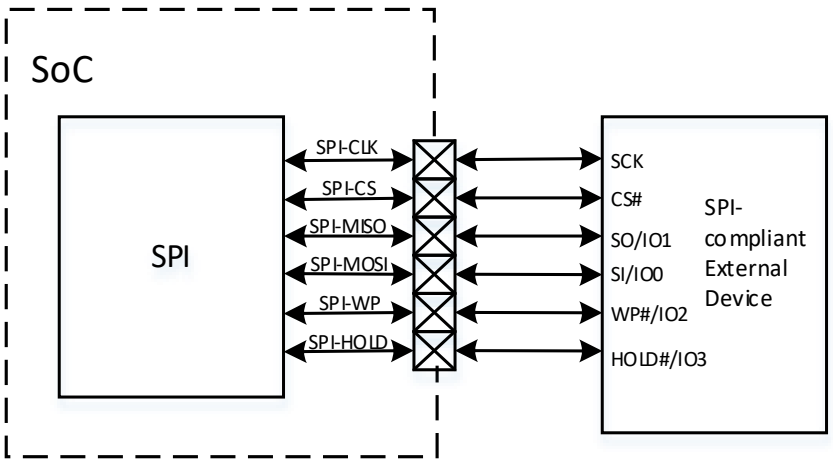
Table 8-46 SPI Clock Sources

SPI	Clock Sources	Description	Clock Module
SPI0, SPI2	HOSC	24 MHz Crystal	CCU
	PERI0_200M	Peripheral Clock, default value is 200 MHz.	
	PERI0_300M	Peripheral Clock, default value is 300 MHz.	
	PERI1_200M	Peripheral Clock, default value is 200 MHz.	
	PERI1_300M	Peripheral Clock, default value is 300 MHz.	
S_SPI0	DCXO24M	24 MHz Crystal	PRCM
	PERIPLL_DIV	Peripheral Clock, default value is 200 MHz.	
	PERI0_300M	Peripheral Clock, default value is 300 MHz.	
	PERI1_300M	Peripheral Clock, default value is 300 MHz.	
	AUDIO1PLL4X	Audio system clock, the defult value is 768 MHz.	

8.16.3.3 Typical Application

The following figure shows the application block diagram when the SPI master device is connected to a slave device.

Figure 8-75 SPI Application Block Diagram



8.16.3.4 SPI Transmit Format

The SPI supports 4 different formats for data transfer. The software can select one of the four modes in which the SPI works by setting the bit1 (Polarity) and bit0 (Phase) of SPI_TCR (Offset: 0x0008). The SPI controller master uses the SPI_SCLK signal to transfer data in and out of the shift register. Data is clocked using any one of four programmable clock phase and polarity combinations.

The CPOL (SPI_TCR [1]) defines the polarity of the clock signal (SPI_SCLK). The SPI_SCLK is a high level when CPOL is '1' and it is a low level when CPOL is '0'. The CPHA (SPI_TCR [0]) decides whether the leading edge of SPI_SCLK is used to setup or sample data. The leading edge is used to setup data when CPHA is '1', and sample data when CPHA is '0'. The following table lists the four modes.

Table 8-47 SPI Transmit Format

Mode	Polarity (CPOL)	Phase (CPHA)	Leading Edge	Trailing Edge
Mode0	0	0	Sample on the rising edge	Setup on the falling edge
Mode1	0	1	Setup on the rising edge	Sample on the falling edge
Mode2	1	0	Sample on the falling edge	Setup on the rising edge
Mode3	1	1	Setup on the falling edge	Sample on the rising edge

The following figures describe four waveform for SPI_SCLK.

Figure 8-76 SPI Phase 0 Timing Diagram

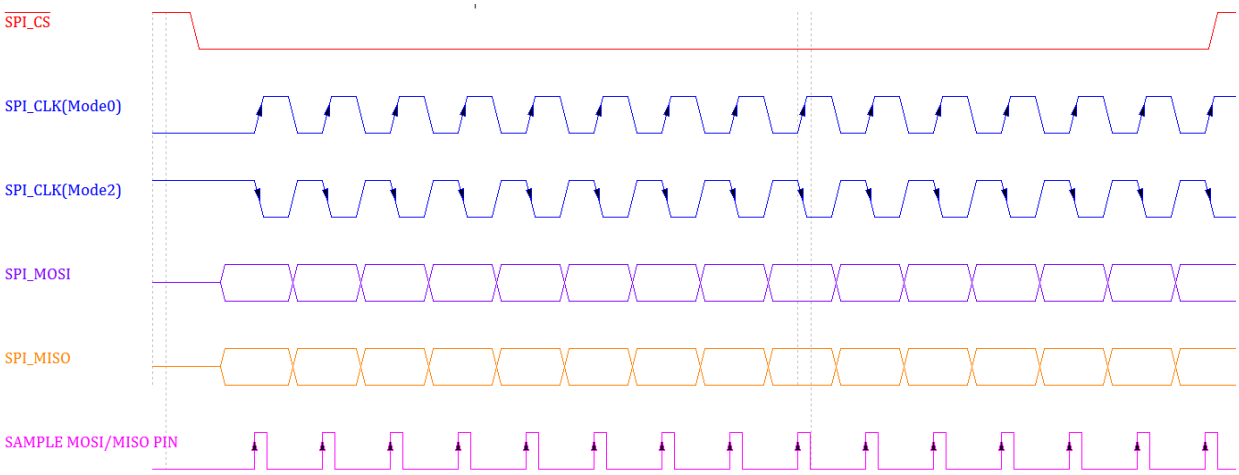
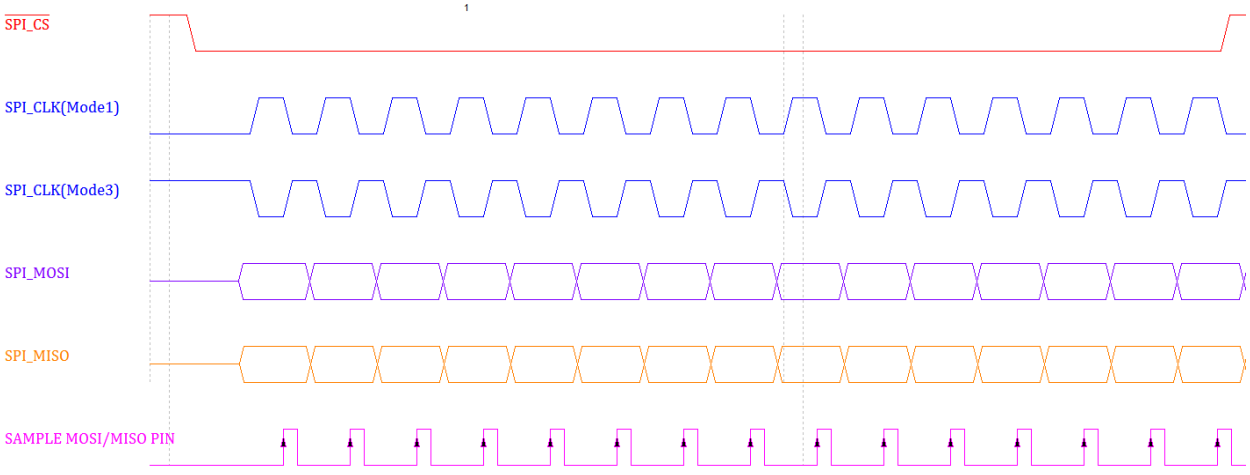


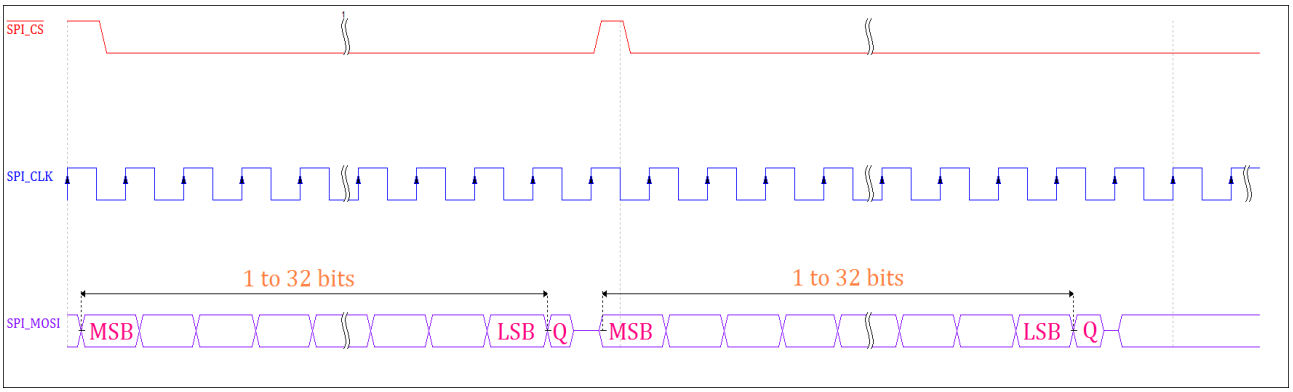
Figure 8-77 SPI Phase 1 Timing Diagram



8.16.3.5 SPI 3-Wire Mode

The SPI 3-wire mode is only valid when the SPI controller work in master mode, and is selected when the Work Mode Select bit (SPI_BATCR [1:0]) is equal to 0x2. And in the 3-wire mode, the input data and the output data use the same single data line. The following figure describes the 3-wire mode.

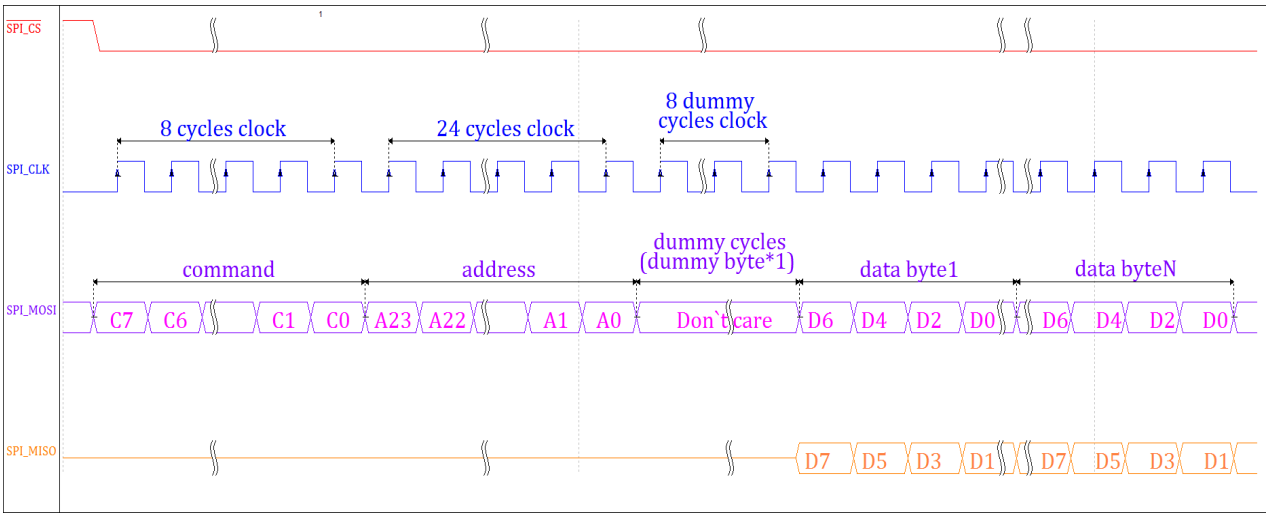
Figure 8-78 SPI 3-Wire Mode



8.16.3.6 SPI Dual-Input/Dual-Output and Dual I/O Mode

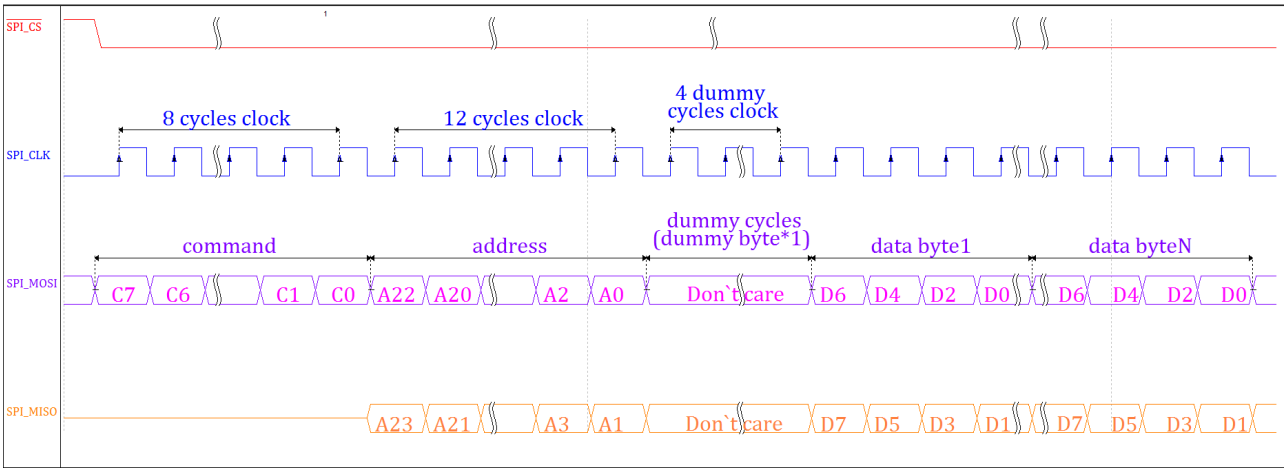
The dual read mode (SPI x2) is selected when the DRM is set in SPI_BCC (Offset: 0x0038) [28]. Using the dual mode allows data to be transferred to or from the device at double the rate of standard single mode, the data can be read at fast speed using two data bits (MOSI and MISO) at a time. The following figure describes the dual-input/dual-output SPI and the dual I/O SPI.

Figure 8-79 SPI Dual-Input/Dual-Output Mode



In the dual-input/dual-output SPI mode, the command, address, and the dummy bytes output in a unit of a single bit in serial mode through the SPI_MOSI line, only the data bytes are output (write) and input (read) in a unit of dual bits through the SPI_MOSI and SPI_MISO.

Figure 8-80 SPI Dual I/O Mode

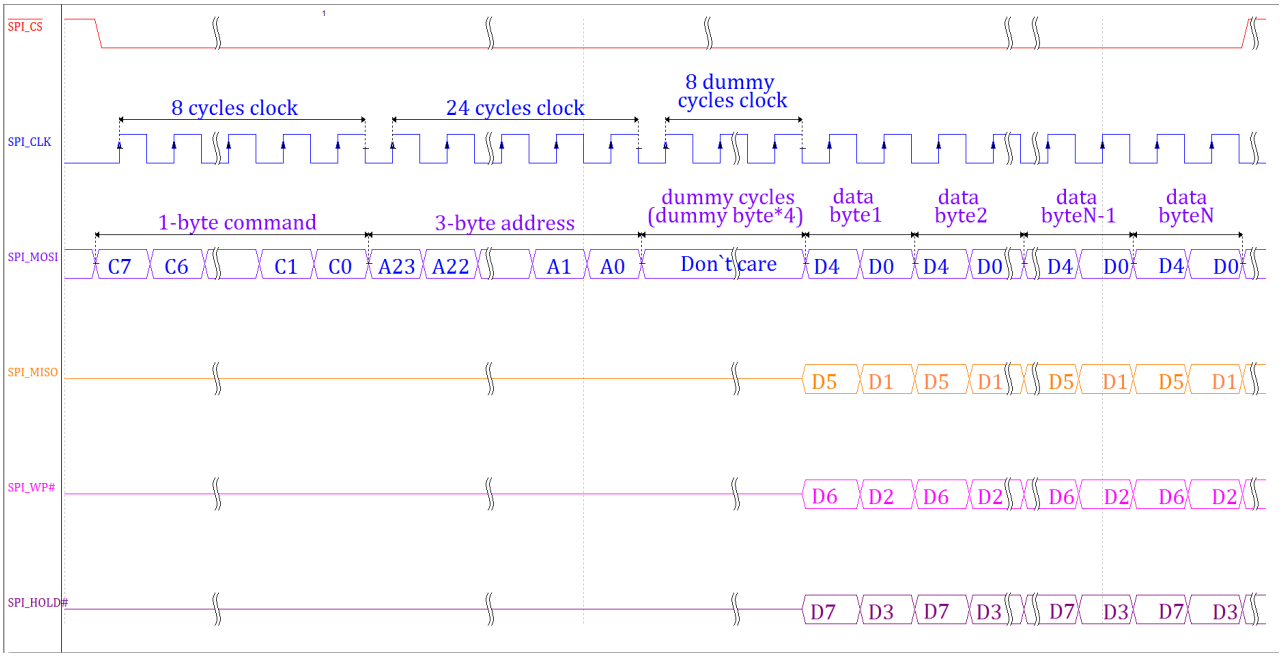


In the dual I/O SPI mode, only the command bytes output in a unit of a single bit in serial mode through the SPI_MOSI line. The address bytes and the dummy bytes output in a unit of dual bits through the SPI_MOSI and SPI_MISO. And the data bytes output (write) and input (read) in a unit of dual bits through the SPI_MOSI and SPI_MISO.

8.16.3.7 SPI Quad-Input/Quad-Output Mode

The quad read mode (SPI x4) is selected when the Quad_EN is set in SPI_BCC (Offset: 0x0038) [29]. Using the quad mode allows data to be transferred to or from the device at 4 times the rate of standard single mode, the data can be read at fast speed using four data bits (MOSI, MISO, IO2 (WP#) and IO3 (HOLD#)) at the same time. The following figure describes the quad-input/quad-output SPI.

Figure 8-81 SPI Quad-Input/Quad-Output Mode



In the quad-input/quad-output SPI mode, the command, address, and the dummy bytes output in a unit of a single bit in serial mode through SPI_MOSI line. Only the data bytes output (write) and input (read) in a unit of quad bits through the SPI_MOSI, SPI_MISO, SPI_WP#, and SPI_HOLD#.

8.16.3.8 Transmission/Reception Bursts in Master Mode

In SPI master mode, the transmission and reception bursts (byte in unit) are configured before the SPI transfers serial data between the processor and external device. The transmission bursts are written in MWTC (bit [23:0]) of the SPI Master Transmit Counter Register. The transmission bursts in single mode before automatically sending dummy bursts are written in STC (bit [23:0]) of the SPI Master Burst Control Counter Register. For dummy data, the SPI controller can automatically send before receiving by writing DBC (bit [27:24]) in the SPI Master Burst Control Counter Register. If users do not use the SPI controller to send dummy data automatically, then the dummy bursts are used as the transmission counters to write together in MWTC (bit [23:0]) of the SPI Master Transmit Counter Register. In master mode, the total burst numbers are written in MBC (bit [23:0]) of the SPI Master Burst Counter Register. When all transmission and reception bursts are transferred, the SPI controller will send a completed interrupt, at the same time, the SPI controller will clear DBC, MWTC, and MBC.

8.16.3.9 SPI Sample Mode and Run Clock Configuration

The SPI controller runs at 3 kHz–100 MHz at its interface to external SPI devices. The internal SPI clock should run at the same frequency as the outgoing clock in the master mode. The SPI clock is selected from different clock sources, the SPI must configure different work mode. There are three work modes: normal sample mode, delay half-cycle sample mode, delay one-cycle sample mode. Delay half-cycle sample mode is the default mode of the SPI controller. When the SPI runs at 40 MHz or below 40 MHz, the SPI can work at normal sample mode or delay half-cycle sample mode. When the SPI runs over 80 MHz, setting the SDC bit in the SPI Transfer Control Register to '1' makes the internal read sample point with a half-cycle delay of SPI_CLK, which is used in high speed read operation to reduce the error caused by the time delay of SPI_CLK between master and slave. The following tables show the different configurations of the SPI sample mode.

Table 8-48 SPI Old Sample Mode and Run Clock

SPI Sample Mode	SDM(bit13)	SDC(bit11)	Run Clock
normal sample	1	0	<=24 MHz
delay half cycle sample	0	0	<=40 MHz
delay one cycle sample	0	1	>=80 MHz

CAUTION

The remaining spectrum is not recommended. Because when the output delay of SPI (refer to the datasheet of the manufactures for the specific delay time) is the same with the half-cycle time of SPI working clock, the variable edge of the output data for the device bumps into the clock sampling edge of the controller, so setting 1 cycle of sampling delay would cause stability problem.

Table 8-49 SPI New Sample Mode

SPI Sample Mode	SDM (bit13)	SDC (bit11)	SDC1 (bit15)
normal sample	1	0	0
delay half cycle sample	0	0	0
delay one cycle sample	0	1	0
delay 1.5 cycle sample	1	1	0
delay 2 cycle sample	1	0	1
delay 2.5 cycle sample	0	0	1
delay 3 cycle sample	0	1	1

8.16.3.10 SPI Error Conditions

If any error conditions occur, the hardware will set the corresponding status bits in the SPI Interrupt Status Register (Offset: 0x0014) and stop the transfer. For the SPI controller, the following error scenarios can happen.

- TX_FIFO Underrun

The TX_FIFO underrun happens when the CPU/DMA reads data from TX FIFO when it is empty. In the case, the SPI controller will end the transaction and flag the error bit along with the TF_UDF bit in the SPI Interrupt Status Register (Offset: 0x0014). The SPI controller will generate an interrupt if interrupts are enabled. The software has to clear the error bit and the TF_UDF bit. To start a new transaction, the software has to reset the FIFO by writing to the SRST (soft reset) bit in the SPI Global Control Register (Offset: 0x0004).

- TX_FIFO Overflow

The TX_FIFO overflow happens when the CPU/DMA writes data into the TX FIFO when it is full. In the case, the SPI controller will end the transaction and flag the error bit along with the TF_OVF bit in the SPI Interrupt Status Register (Offset: 0x0014). The SPI controller will generate an interrupt if interrupts are enabled. The software has to clear the error bit and the TF_OVF bit. To start a new transaction, the software has to reset the FIFO by writing to the SRST (soft reset) bit in the SPI Global Control Register (Offset: 0x0004).

- RX_FIFO Underrun

The RX_FIFO underrun happens when the CPU/DMA reads data from RX FIFO when it is empty. In the case, the SPI controller will end the transaction and flag the error bit along with the RF_UDF bit in the SPI Interrupt Status Register (Offset: 0x0014). The SPI controller will generate an interrupt

if interrupts are enabled. The software has to clear the error bit and the RF_UDF bit. To start a new transaction, the software has to reset the FIFO by writing to the SRST (soft reset) bit in the SPI Global Control Register (Offset: 0x0004).

- RX_FIFO Overflow

The RX_FIFO overflow happens when the CPU/DMA writes data into the RX FIFO when it is full. In the case, the SPI controller will end the transaction and flag the error bit along with the RF_OVF bit in the SPI Interrupt Status Register (Offset: 0x0014). The SPI controller will generate an interrupt if interrupts are enabled. The software has to clear the error bit and the RF_OVF bit. To start a new transaction, the software has to reset the FIFO by writing to the SRST (soft reset) bit in the SPI Global Control Register (Offset: 0x0004).

8.16.4 Programming Guidelines

8.16.4.1 Writing/Reading Data Process

The SPI transfers serial data between the processor and the external device. The CPU mode and DMA mode are the two main operational modes for SPI. For each SPI, the data is simultaneously transmitted (shifted out serially) and received (shifted in serially). The SPI has 2 channels, including the TX channel and RX channel. The TX channel has the path from TX FIFO to the external device. The RX channel has the path from the external device to RX FIFO.

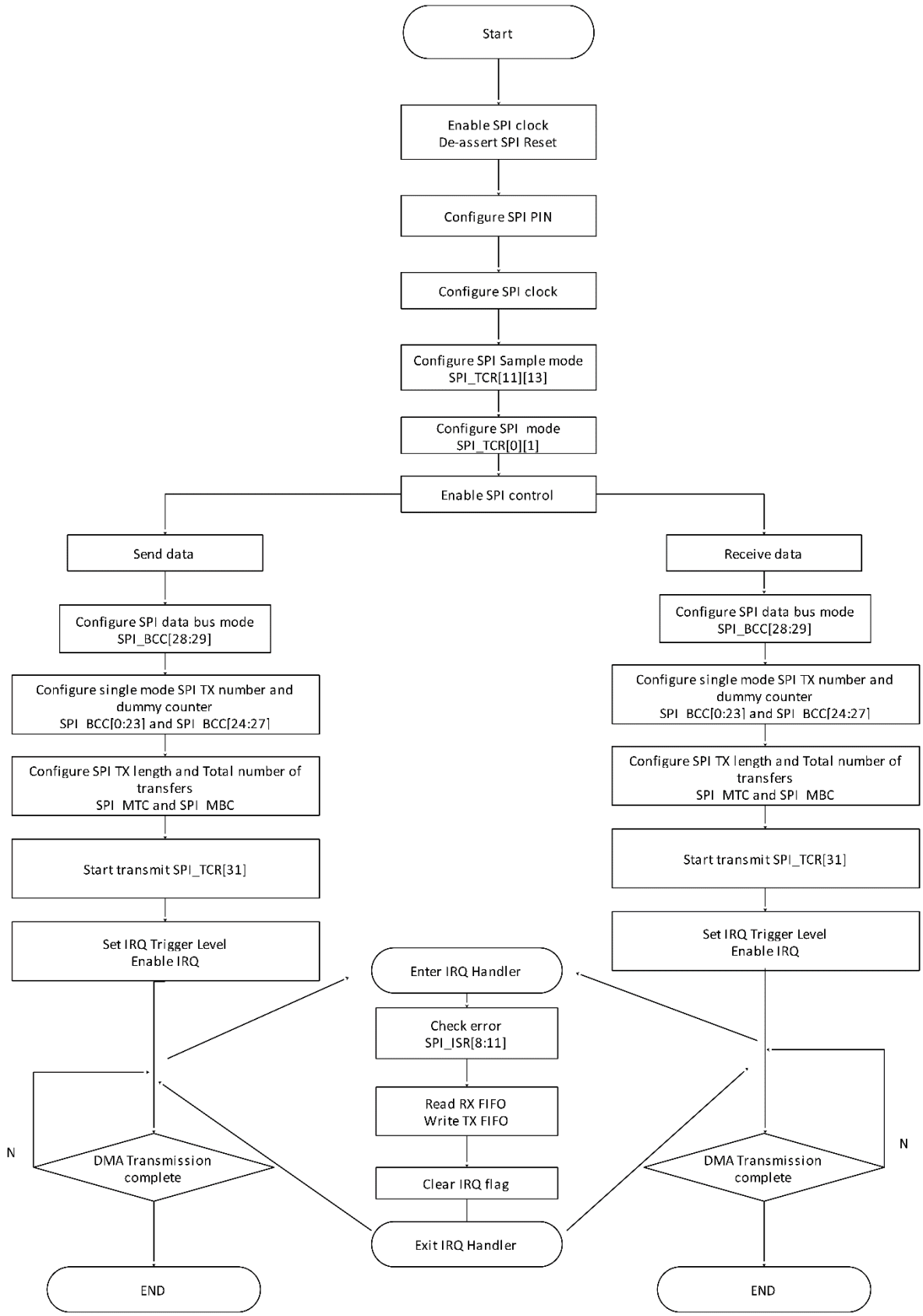
Write Data: The CPU or DMA must write data on the SPI_TXD (Offset: 0x0200), the data on the register are automatically moved to TX FIFO.

Read Data: To read data from RX FIFO, the CPU or DMA must access the SPI_RXD (Offset: 0x0300) and the data are automatically sent to the SPI_RXD (Offset: 0x0300).

In CPU or DMA mode, the SPI sends a completed interrupt (SPI_ISR[TC]) to the processor after each transmission is complete.

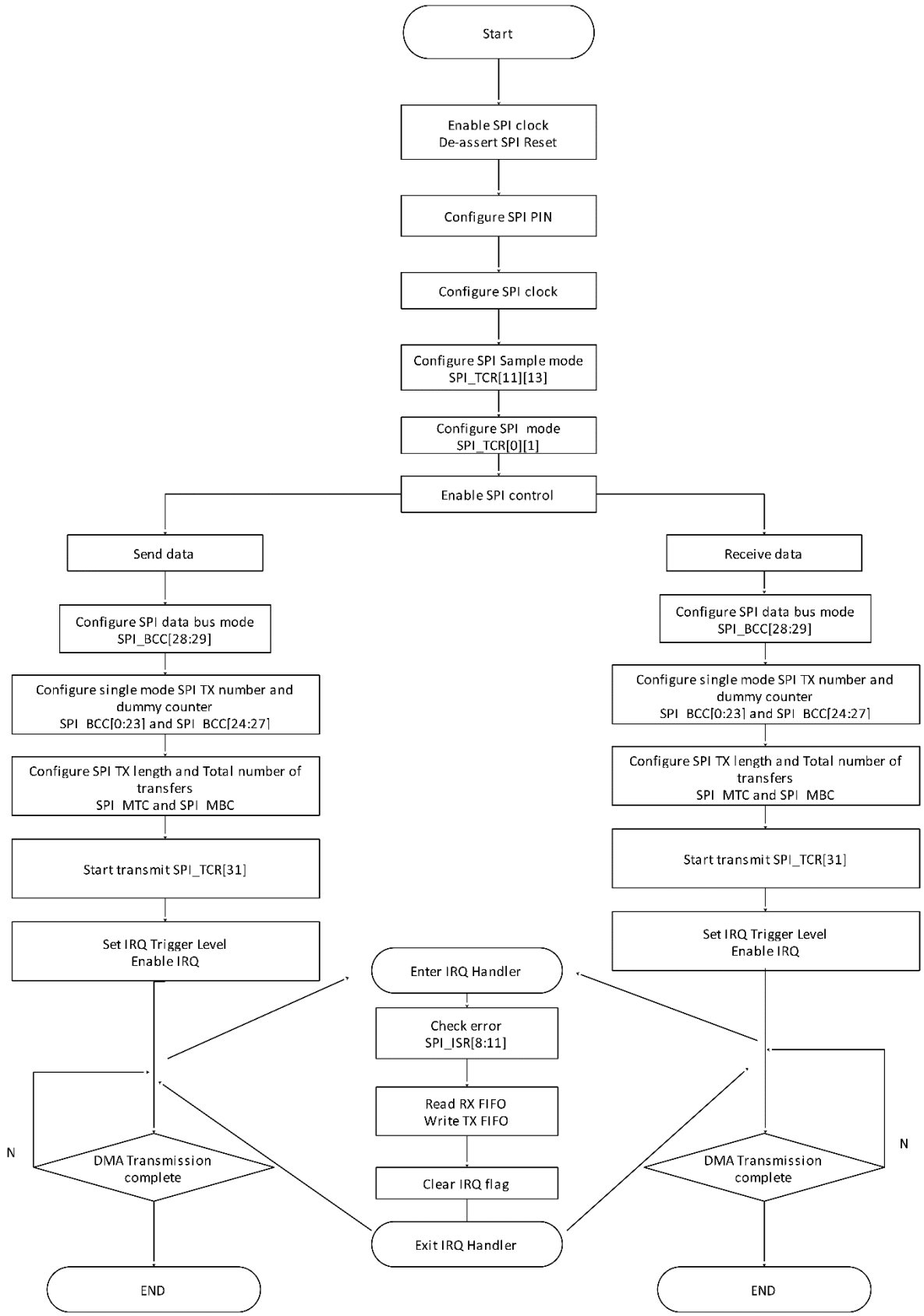
CPU Mode

Figure 8-82 SPI Write/Read Data in CPU Mode



DMA Mode

Figure 8-83SPI Write/Read Data in DMA Mode



8.17 SPI_DBI

8.17.1 Overview

The A527 provides a 3/4 line SPI display bus interface (SPI_DBI) for video data transmission. It supports DBI mode or SPI mode. The DBI mode is compatible with multiple video data formats at the same time. The SPI mode is used for low-cost display schemes.

The SPI mode has the following features:

- Multiple SPI modes:
  - Master mode and slave mode for standard SPI
  - Master mode for Dual-Output/Dual-Input SPI and Dual I/O SPI
  - Master mode for Quad-Output/Quad-Input SPI
  - Master mode for 3-wire SPI, with programmable serial data frame length of 1 bit to 32 bits
- Maximum clock frequency: 100MHz
- TX/RX DMA slave interface
- 8-bit wide by 64-entry FIFO for both transmitting and receiving data
- Supports mode0, mode1, mode2, and mode3
- Polarity and phase of the Chip Select (SPI_SS) and SPI Clock (SPI_SCLK) are configurable

The DBI mode has the following features:

- DBI Type C 3 Line/4 Line Interface Mode
- 2 Data Lane Interface Mode
- RGB111/444/565/666/888 video format
- Maximum resolution of RGB666 240 x 320@30Hz with single data lane
- Maximum resolution of RGB888 240 x 320@60Hz or 320 x 480@30Hz with dual data lane
- Tearing effect
- Software flexible control video frame rate

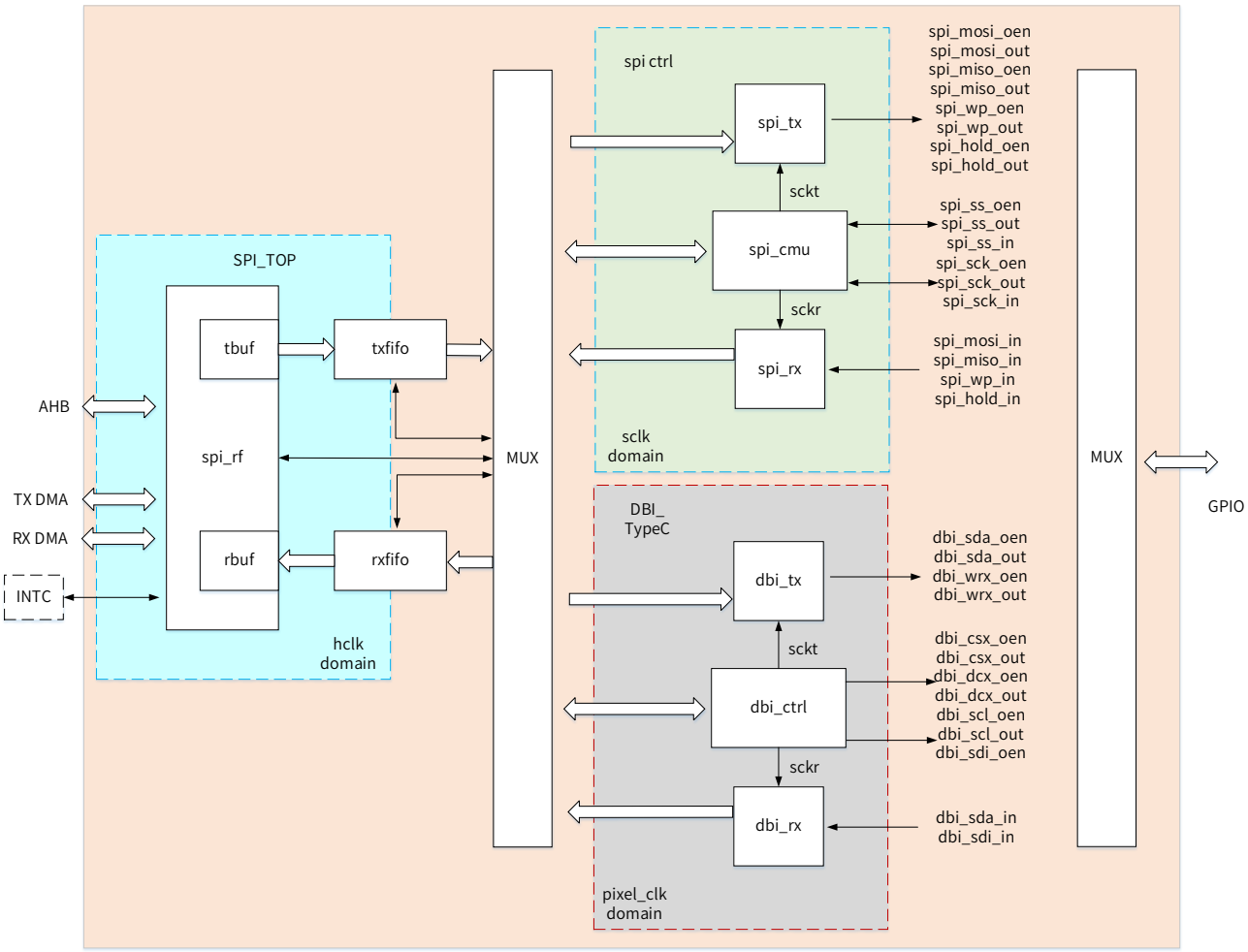
NOTE

This chapter only describes SPI1 (SPI mode and DBI mode). For detailed information of SPI0, SPI2, and S_SPI0, please refer to section 8.16 SPI.

8.17.2 Block Diagram

The following figure shows a block diagram of the SPI_DBI.

Figure 8-84 SPI_DBI Block Diagram



SPI_DBI contains the following sub-blocks:

Table 8-50 SPI_DBI Sub-blocks

Sub-block	Description
spi_rf	Responsible for implementing the internal register, interrupt, and DMA Request.
spi_tbuf	The data length transmitted from AHB to TXFIFO is converted into 8 bits, then the data is written into the RXFIFO.
spi_rbuf	The block is used to convert the RXFIFO data into the reading data length of AHB.
txfifo, rxfifo	The data transmitted from the SPI to the external serial device is written into the TXFIFO; the data received from the external serial device into SPI is pushed into the RXFIFO.
spi_cmu	Responsible for implementing SPI bus clock, chip select, internal sample, and the generation of transfer clock.

Sub-block	Description
spi_tx	Responsible for implementing SPI data transfer, the interface of the internal TXFIFO, and status register.
spi_rx	Responsible for implementing SPI data receive, the interface of the internal RXFIFO, and status register.
dbi_ctrl	Responsible for implementing DBI bus clock, chip select, data command select, RGB format reshape.
dbi_tx	Responsible for implementing DBI data transfer, the interface of the internal TXFIFO, and status register.
dbi_rx	Responsible for implementing DBI data receive, the interface of the internal RXFIFO, and status register.

8.17.3 Functional Description

8.17.3.1 External Signals

The following table describes the external signals of SPI_DBI. When using SPI_DBI, the corresponding PADs are selected as SPI_DBI function via section 8.6 GPIO.

Table 8-51 GPIO multiplexing of SPI1 and DBI

DBI	SPI1
DBI-CSX	SPI1-CS0
DBI-SCLK	SPI1-CLK
DBI-SDO	SPI1-MOSI
DBI-SDI/ DBI-TE/ DBI-DCX	SPI1-MISO
DBI-DCX/DBI-WRX	SPI1-HOLD
DBI-TE	SPI1-WP

Table 8-52 SPI_DBI External Signals

Signal Name	Description	Type
SPI Mode		
SPI1-CS0	SPI1 Chip Select Signal, Low Active	I/O
SPI1-CLK	SPI1 Clock Signal Provides serial interface timing.	I/O
SPI1-MOSI	SPI1 Master Data Out, Slave Data In	I/O
SPI1-MISO	SPI1 Master Data In, Slave Data Out	I/O
SPI1-WP	SPI1 Write Protect, Low Active Protects the memory area against all program or erase instructions. It also can be used for serial data input and output for SPI Quad Input or Quad Output mode.	I/O

Signal Name	Description	Type
SPI1-HOLD	SPI1 Hold Signal Pauses any serial communication with the device without deselecting or resetting it. It also can be used for serial data input and output for SPI Quad Input or Quad Output mode.	I/O
DBI Mode		
DBI-CSX	Chip Select Signal, Low Active	I/O
DBI-SCLK	Serial Clock Signal	I/O
DBI-SDO	Data Output Signal	I/O
DBI-SDI	Data Input Signal The data is sampled on the rising edge and the falling edge	I/O
DBI-TE	Tearing Effect Input It is used to capture the external TE signal edge. The rising and falling edge is configurable.	I/O
DBI-DCX	DCX pin is the select output signal of data and command. DCX = 0: register command; DCX = 1: data or parameter.	I/O
DBI-WRX	When DBI operates in dual data lane format, the RGB666 format 2 can use WRX to transfer data	I/O

8.17.3.2 Clock Sources

The SPI_DBI controller gets 5 different clock sources, users can select one of them to make SPI_DBI clock source. The following table describes the clock sources for SPI_DBI. For more details on the clock setting, configuration, and gating information, see section 2.6 Clock Controller Unit (CCU).

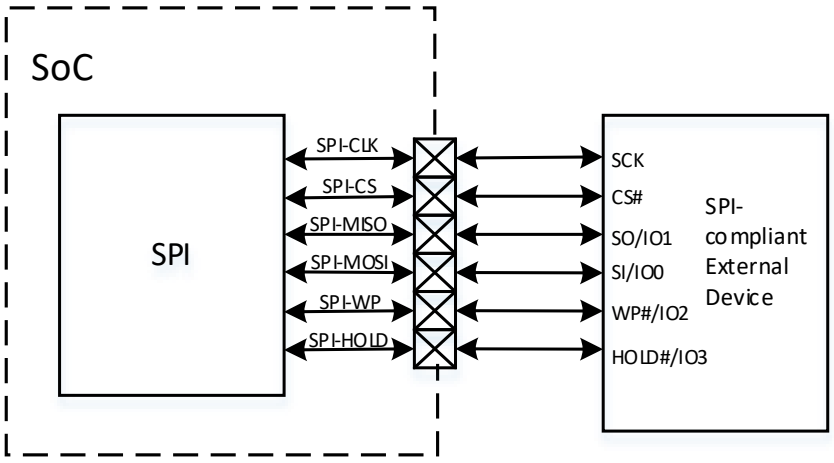
Table 8-53 SPI_DBI Clock Sources

Clock Sources	Description	Clock Module
HOSC	24 MHz Crystal	CCU
PERI0_200M	Peripheral Clock, default value is 200 MHz.	
PERI0_300M	Peripheral Clock, default value is 300 MHz.	
PERI1_200M	Peripheral Clock, default value is 200 MHz.	
PERI1_300M	Peripheral Clock, default value is 300 MHz.	

8.17.3.3 Typical Application

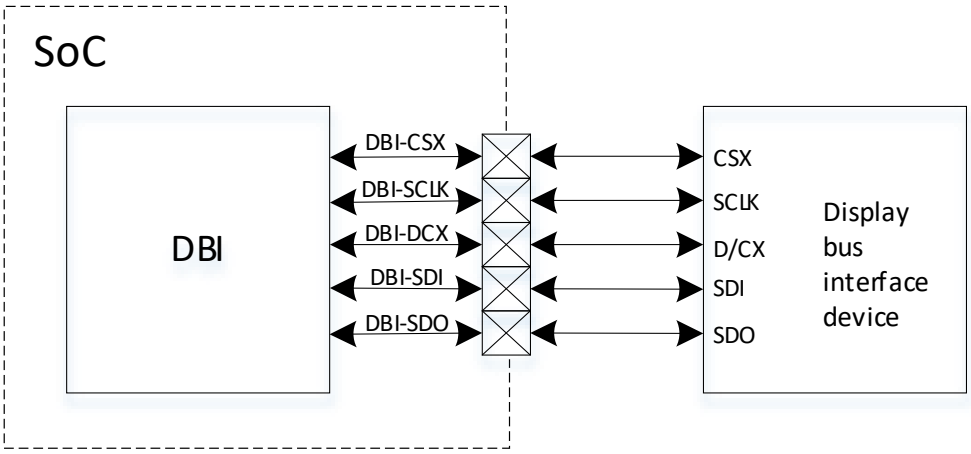
The following figure shows the application block diagram when the SPI master device is connected to a slave device.

Figure 8-85 SPI Application Block Diagram



The following figure shows the application block diagram when the DBI master device is connected to a display bus interface device.

Figure 8-86 DBI Application Block Diagram



8.17.3.4 SPI Transmission Format

The SPI supports 4 different formats for data transmission. The software can select one of the four modes in which the SPI works by setting the bit1 (Polarity) and bit0 (Phase) of SPI_TCR. The SPI controller master uses the SPI_SCLK signal to transfer data in and out of the shift register. Data is clocked using any one of four programmable clock phase and polarity combinations.

The CPOL (SPI_TCR[1]) defines the polarity of the clock signal (SPI_SCLK). The SPI_SCLK is a high level when CPOL is '1' and it is a low level when CPOL is '0'. The CPHA (SPI_TCR[0]) decides whether the leading edge of SPI_SCLK is used to setup or sample data. The leading edge is used to setup data when CPHA is '1', and sample data when CPHA is '0'. The following table lists the four modes.

Table 8-54 SPI Transmit Format

SPI Mode	Polarity (CPOL)	Phase (CPHA)	Leading Edge	Trailing Edge
mode0	0	0	Sample on the rising edge	Setup on the falling edge
mode1	0	1	Setup on the rising edge	Sample on the falling edge
mode2	1	0	Sample on the falling edge	Setup on the rising edge
mode3	1	1	Setup on the falling edge	Sample on the rising edge

The following figures describe four waveforms for SPI_SCLK.

Figure 8-87 SPI Phase 0 Timing Diagram

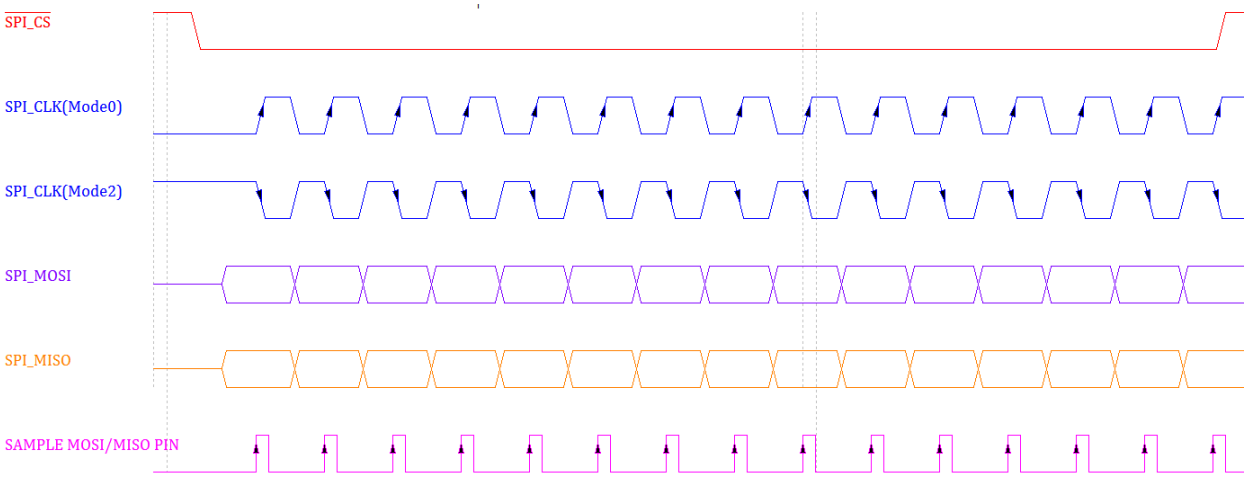
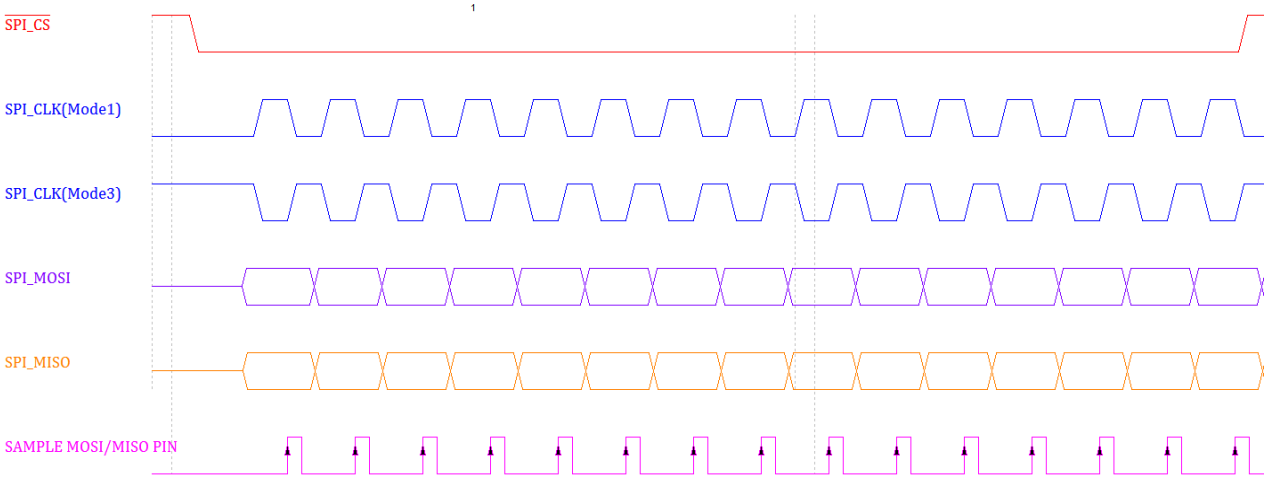


Figure 8-88 SPI Phase 1 Timing Diagram



8.17.3.5 SPI Master and Slave Mode

The SPI controller can be configured to a master or slave device. The master mode is selected by setting the MODE bit (SPI_GCR[1]); the slave mode is selected by clearing the MODE bit.

In master mode, the SPI_CLK is generated and transmitted to the external device, and the data from the TX FIFO is transmitted on the MOSI pin, the data from the slave is received on the MISO pin and sent to RX FIFO. The Chip Select (SPI_SS) is an active low signal, and it must be set low

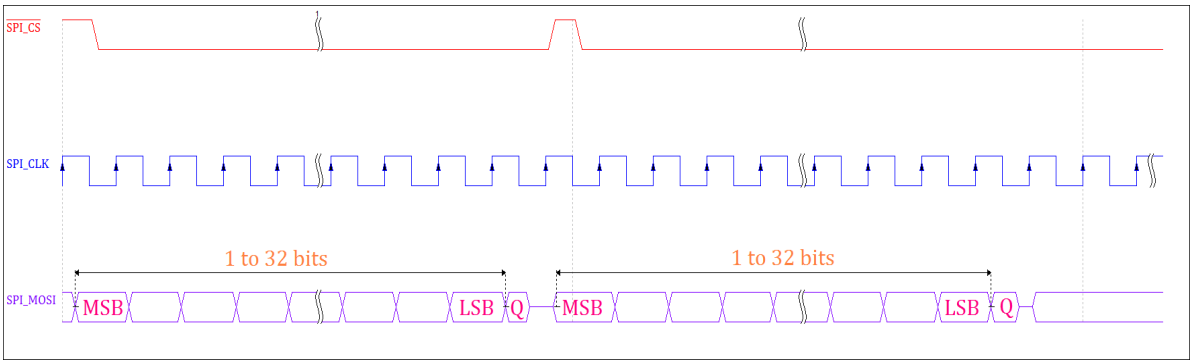
before the data are transmitted or received. The SPI_SS can be selected the auto control mode or the software manual control mode. When using auto control, the SS_OWNER (SPI_TCR[6]) must be cleared (default value is 0); when using manual control, the SS_OWNER must be set. And the level of SPI_SS is controlled by SS_LEVEL (SPI_TCR[7]).

In slave mode, after the software selects the MODE bit (SPI_GCR [1]) to '0', it waits for master initiate a transaction. When the master asserts SPI_SS, and SPI_CLK is transmitted to the slave, the slave data is transmitted from TX FIFO on the MISO pin, and the data from the MOSI pin is received in RX FIFO.

8.17.3.6 SPI 3-Wire Mode

The SPI 3-wire mode is only valid when the SPI controller work in master mode, and is selected when the Work Mode Select bit (SPI_BATC [1:0]) is equal to 0x2. And in the 3-wire mode, the input data and the output data use the same single data line. The following figure describes the 3-wire mode.

Figure 8-89 SPI 3-Wire Mode

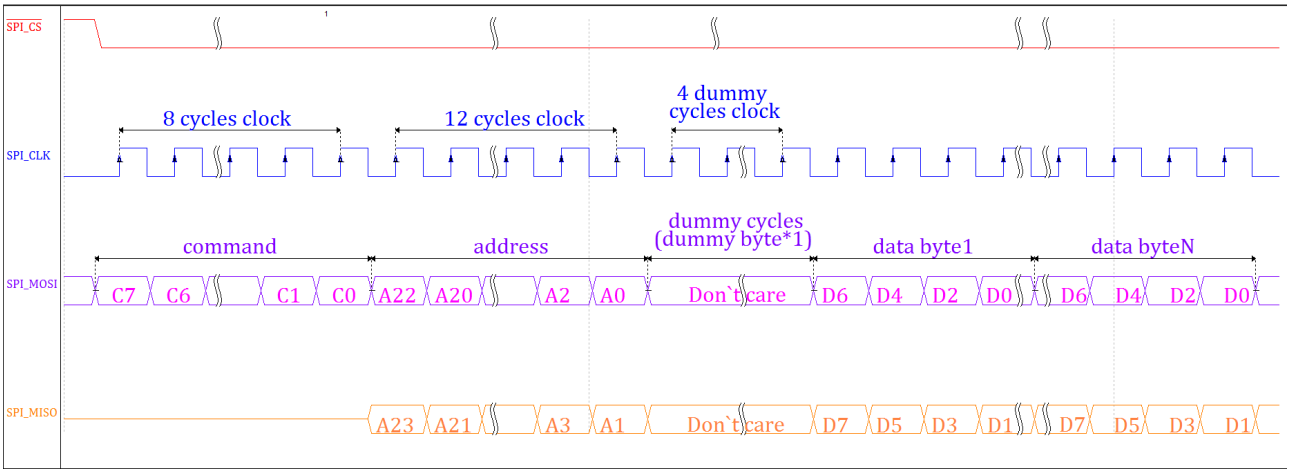


8.17.3.7 SPI Dual-Input/Dual-Output and Dual I/O Mode

The dual read mode (SPI x2) is selected when the DRM is set in SPI_BCC [28]. Using the dual mode allows data to be transferred to or from the device at double the rate of standard single mode SPI

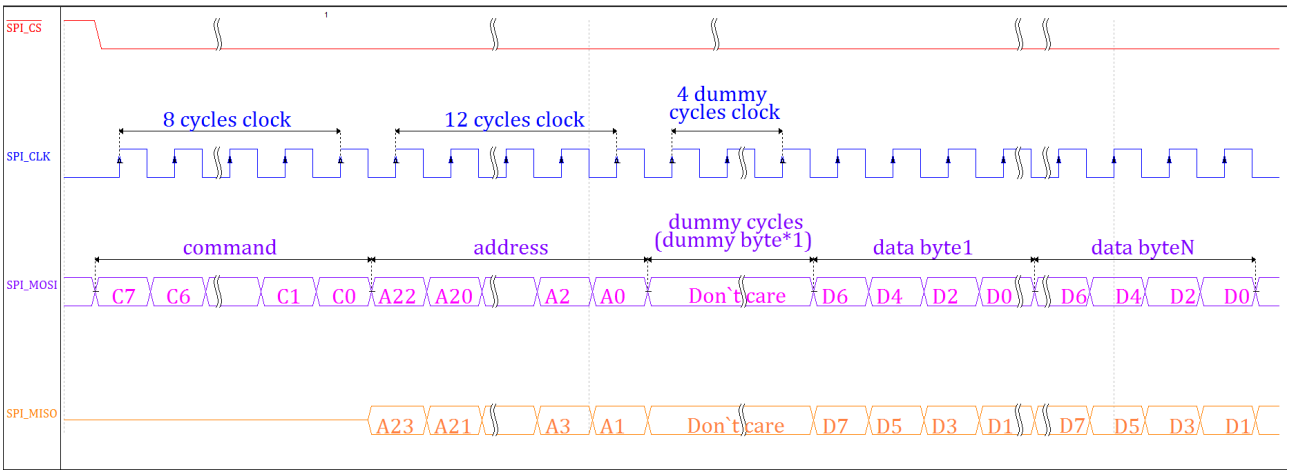
devices, the data can be read at fast speed using two data bits (MOSI and MISO) at a time. The following figure describes the dual-input/dual-output SPI and the dual I/O SPI

Figure 8-90 SPI Dual-Input/Dual-Output Mode



In the dual-input/dual-output SPI mode, the command, address, and the dummy bytes output in a unit of a single bit in serial mode through the SPI_MOSI line, only the data bytes are output (write) and input (read) in a unit of dual bits through the SPI_MOSI and SPI_MISO.

Figure 8-91 SPI Dual I/O Mode



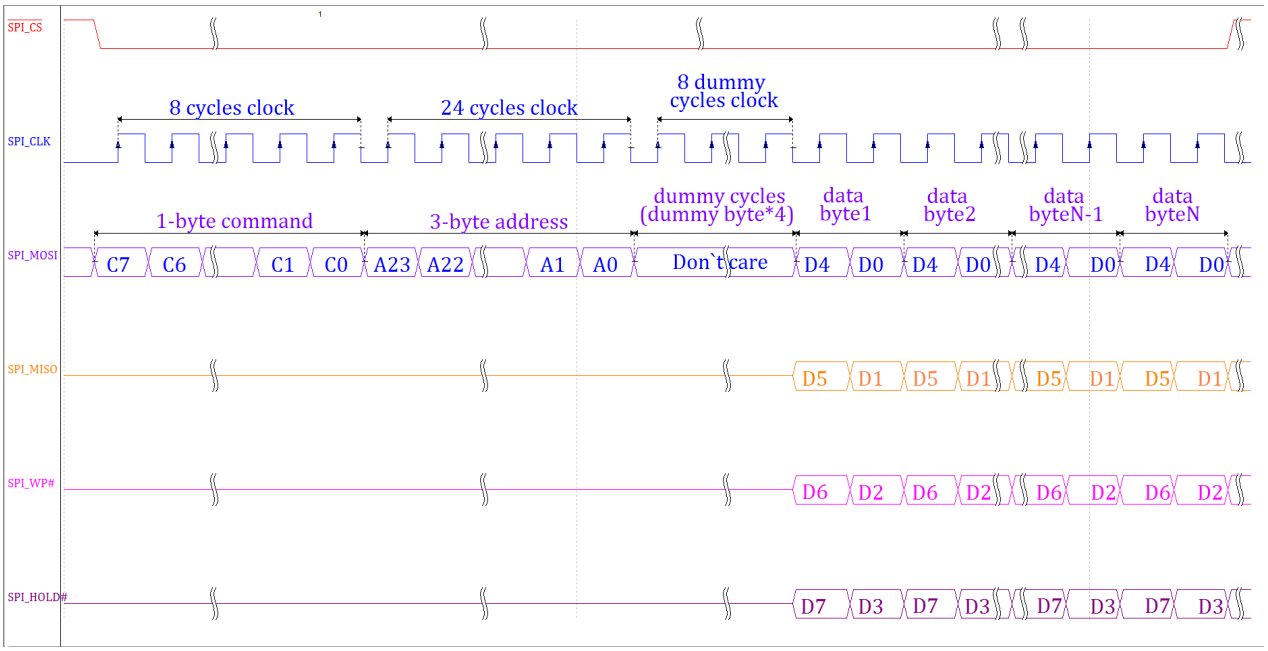
In the dual I/O SPI mode, only the command bytes output in a unit of a single bit in serial mode through the SPI_MOSI line. The address bytes and the dummy bytes output in a unit of dual bits through the SPI_MOSI and SPI_MISO. And the data bytes output (write) and input (read) in a unit of dual bits through the SPI_MOSI and SPI_MISO.

8.17.3.8 SPI Quad-Input/Quad-Output Mode

The quad read mode (SPI x4) is selected when the Quad_EN is set in SPI_BCC [29]. Using the quad mode allows data to be transferred to or from the device at 4 times the rate of standard single mode SPI devices, the data can be read at fast speed using four data bits (MOSI, MISO, IO2 (WP#)

and IO3 (HOLD#)) at the same time. The following figure describes the quad-input/quad-output SPI.

Figure 8-92 SPI Quad-Input/Quad-Output Mode



In the quad-input/quad-output SPI mode, the command, address, and the dummy bytes output in a unit of a single bit in serial mode through the SPI_MOSI line. Only the data bytes output (write) and input (read) in a unit of quad bits through the SPI_MOSI, SPI_MISO, SPI_WP#, and SPI_HOLD#.

8.17.3.9 Transmission/Reception Bursts in Master Mode

In SPI master mode, the transmission and reception bursts (byte in unit) are configured before the SPI transfers serial data between the processor and external device. The transmission bursts are written in MWTC (bit [23:0]) of the SPI Master Transmit Counter Register. The transmission bursts in single mode before automatically sending dummy bursts are written in STC (bit [23:0]) of the SPI Master Burst Control Counter Register. For dummy data, the SPI controller can automatically send before receiving by writing DBC (bit [27:24]) in the SPI Master Burst Control Counter Register. If users do not use the SPI controller to send dummy data automatically, then the dummy bursts are used as the transmission counters to write together in MWTC (bit [23:0]) of the SPI Master Transmit Counter Register. In master mode, the total burst numbers are written in MBC (bit [23:0]) of the SPI Master Burst Counter Register. When all transmission and reception bursts are transferred, the SPI controller will send a completed interrupt, at the same time, the SPI controller will clear DBC, MWTC, and MBC.

8.17.3.10 SPI Sample Mode and Run Clock Configuration

The SPI controller runs at 3 kHz–100 MHz at its interface to external SPI devices. The internal SPI clock should run at the same frequency as the outgoing clock in the master mode. The SPI clock is selected from different clock sources, the SPI must configure different work mode. There are three

work modes: normal sample mode, delay half-cycle sample mode, delay one-cycle sample mode. Delay half-cycle sample mode is the default mode of the SPI controller. When the SPI runs at 40 MHz or below 40 MHz, the SPI can work at normal sample mode or delay half-cycle sample mode. When the SPI runs over 80 MHz, setting the SDC bit in the SPI Transfer Control Register to '1' makes the internal read sample point with a half-cycle delay of SPI_CLK, which is used in high speed read operation to reduce the error caused by the time delay of SPI_CLK between master and slave. The following tables show the different configurations of the SPI sample mode.

Table 8-55 SPI Old Sample Mode and Run Clock

SPI Sample Mode	SDM(bit13)	SDC(bit11)	Run Clock
normal sample	1	0	<=24 MHz
delay half cycle sample	0	0	<=40 MHz
delay one cycle sample	0	1	>=80 MHz

CAUTION

The remaining spectrum is not recommended. Because when the output delay of SPI DBI (refer to the datasheet of the manufactures for the specific delay time) is the same with the half-cycle time of SPI working clock, the variable edge of the output data for the device bumps into the clock sampling edge of the controller, so setting 1 cycle of sampling delay would cause stability problem.

Table 8-56 SPI New Sample Mode

SPI Sample Mode	SDM (bit13)	SDC (bit11)	SDC1 (bit15)
normal sample	1	0	0
delay half cycle sample	0	0	0
delay one cycle sample	0	1	0
delay 1.5 cycle sample	1	1	0
delay 2 cycle sample	1	0	1
delay 2.5 cycle sample	0	0	1
delay 3 cycle sample	0	1	1

8.17.3.11 DBI 3-Line Interface Writing and Reading Timing

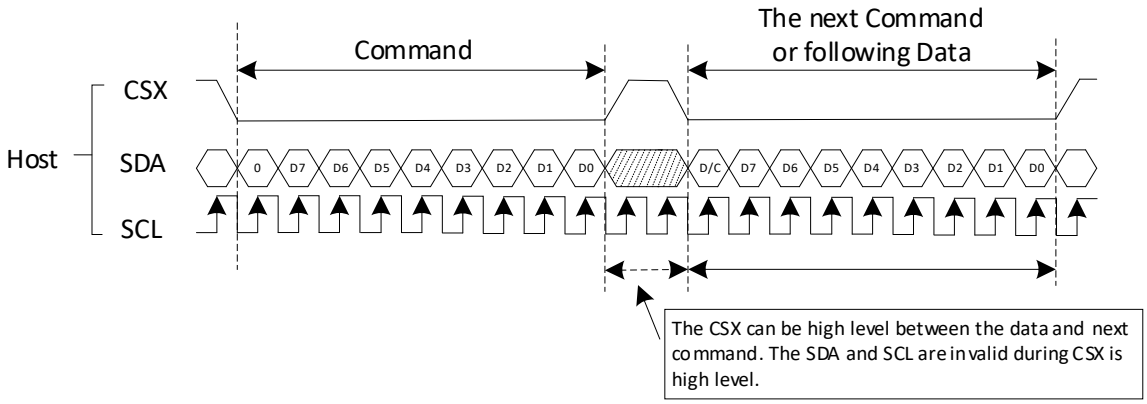
The 3-line DBI Interface I contains CSX, SDA, and SCL, where SDA shares this port for bidirectional port data input and output.

The 3-line DBI Interface II contains CSX, SDA, SCL, and SDI; Data input and output ports are independent of each other.

Since the 3-line display bus mode has no Data/Command data line indicating whether Data or Command is currently being transmitted, an extra bit is added to the data-stream before MSB to indicate whether Data or Command is currently being transmitted. (0: Command, 1: Data)

The following figure shows the writing operation format of 3-line DBI Interface I and Interface II.

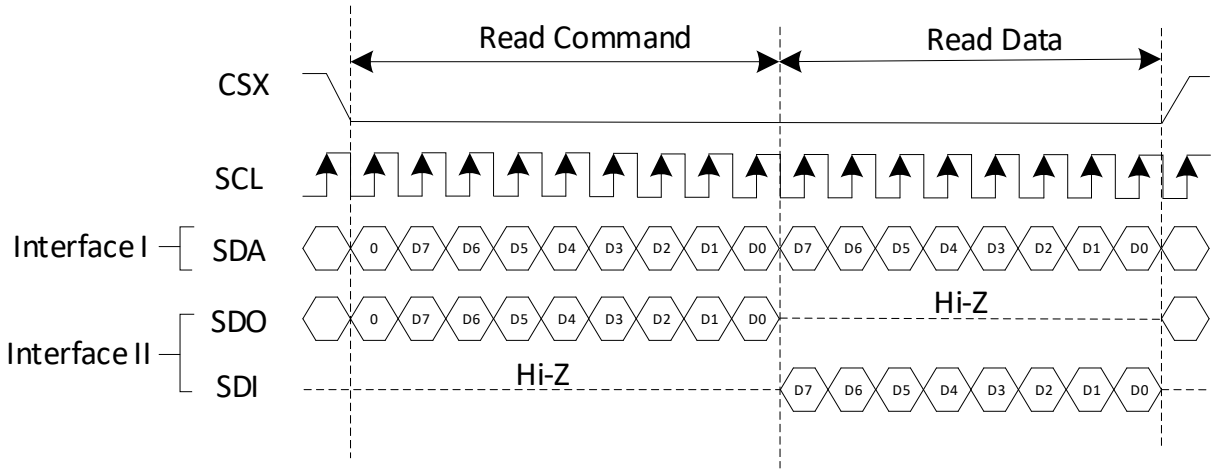
Figure 8-93 DBI 3-Line Display Bus Serial Interface Writing Operation Format



The 3-line DBI Interface I uses the SDA port as bidirectional data input and output port. There are only three cases of data reading volume, 8bits/24bits/32bits, and the first data sampled is high.

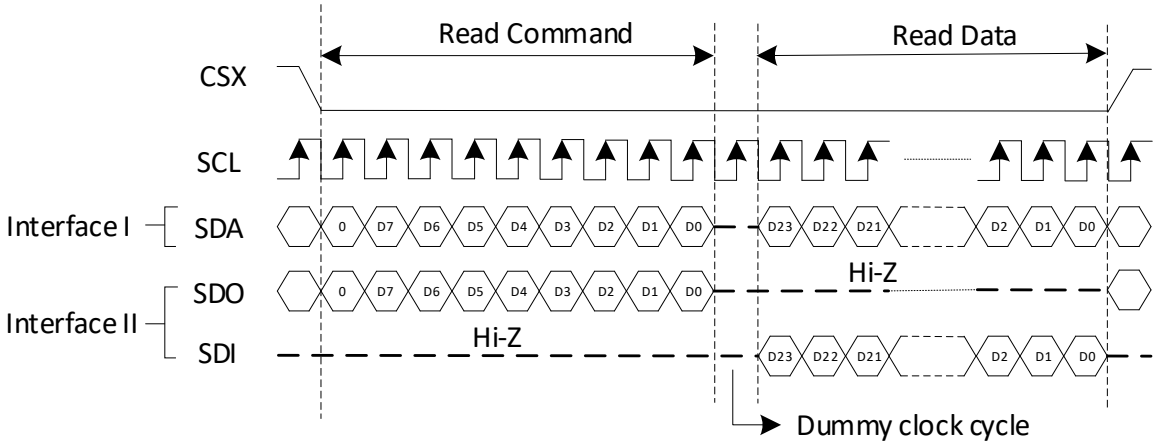
The following figure shows the 8 bits reading operation format of 3-line DBI Interface I and Interface II. After the read command is transmitted, the data is read immediately with on dummy period.

Figure 8-94 DBI 3-Line Display Bus Serial Interface 8-bit Reading Operation Format



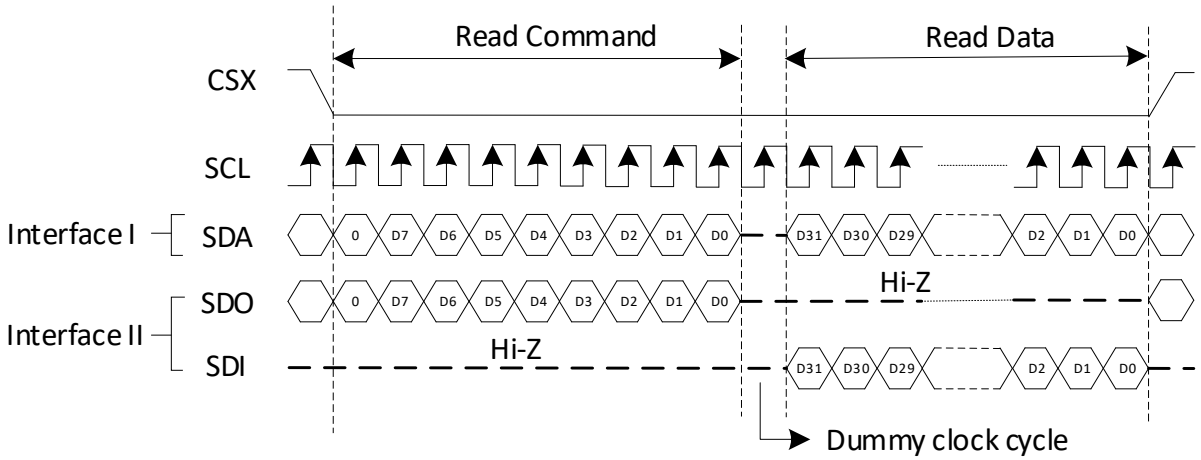
The following figure shows the 24 bits reading operation format of 3-line DBI Interface I and Interface II. After the read command is transmitted, the data is read after waiting for the dummy clock cycle.

Figure 8-95 DBI 3-Line Display Bus Serial Interface 24-bit Reading Operation Format



The following figure shows the 32 bits reading operation format of 3-line DBI Interface I and Interface II. After the read command is transmitted, the data is read after waiting for the dummy clock cycle.

Figure 8-96 DBI 3-Line Display Bus Serial Interface 32-bit Reading Operation Format



8.17.3.12 DBI 4-Line Interface Writing and Reading Timing

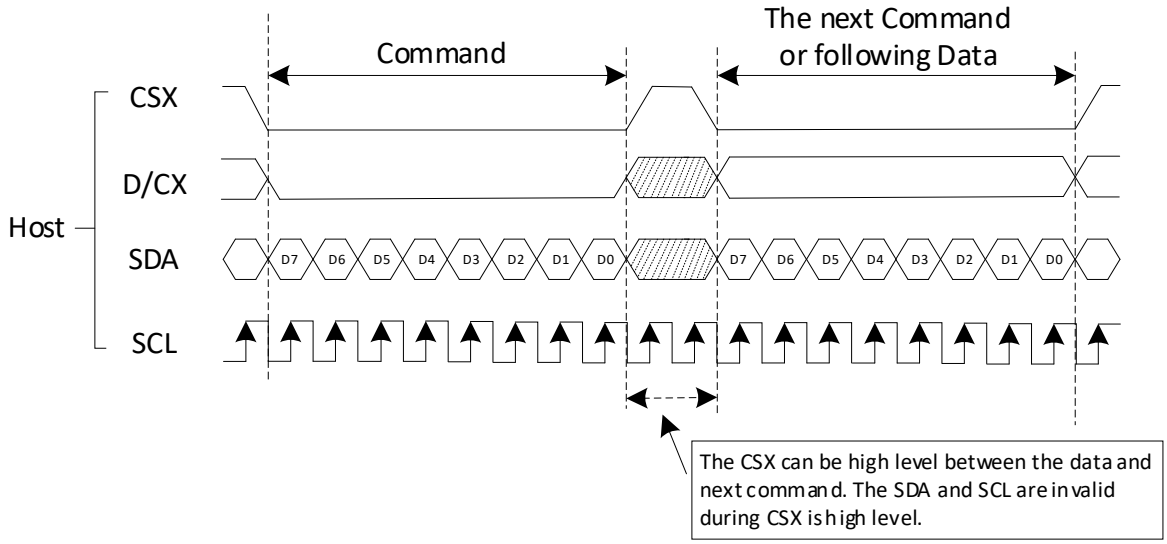
The 4-line DBI Interface I contains CSX, D/CX, SDA, and SCL, where SDA shares this port for bidirectional port data input and output.

The 4-line DBI Interface II contains CSX, D/CX, SDA, SCL, and SDI; Data input and output ports are independent of each other.

Since the 4-line display bus mode has a Data/Command data line indicating whether Data or Command is currently being transmitted (0: Command, 1: Data). So there is no need to add an extra bit to data-stream before MSB like the 3-line DBI.

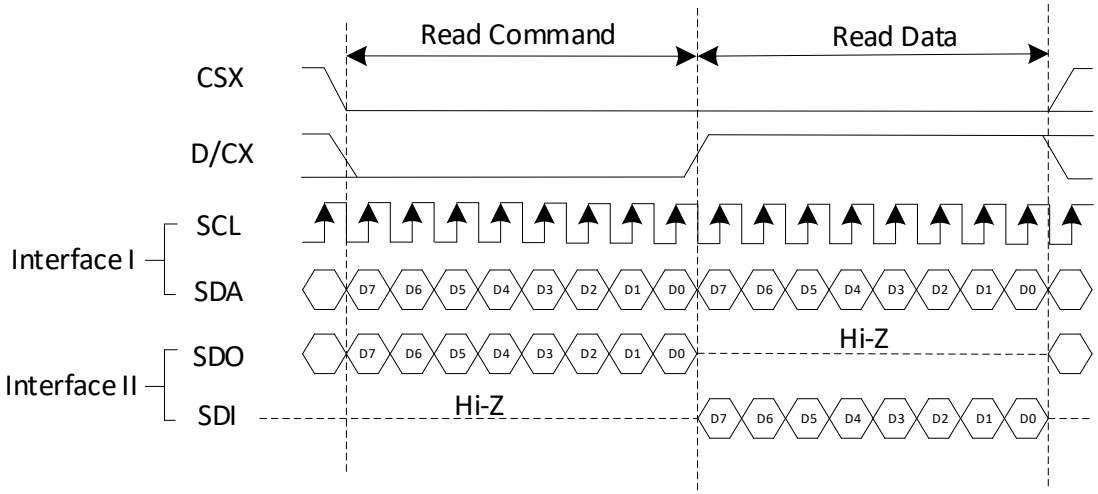
The following figure shows the writing operation format of 4-line DBI Interface I and Interface II.

Figure 8-97 DBI 4-Line Display Bus Serial Interface Writing Operation Format



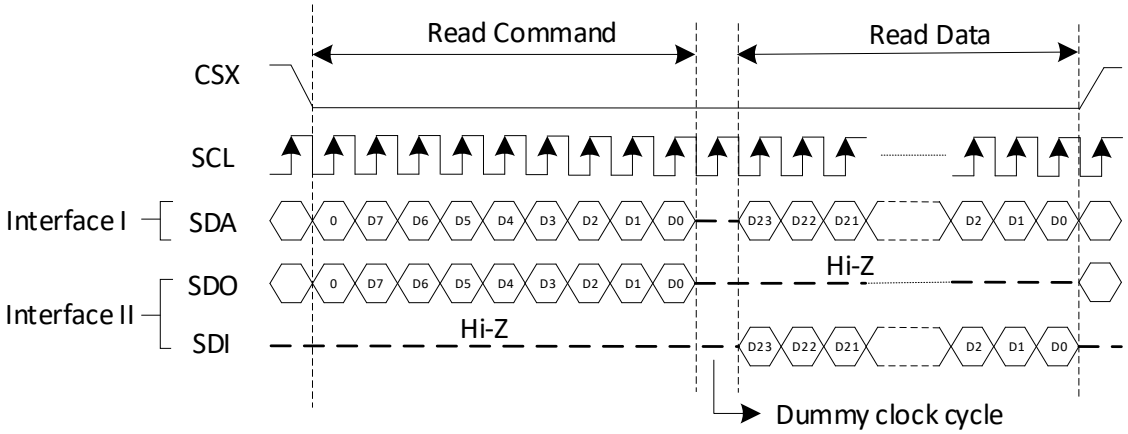
The following figure shows the 8 bits reading operation format of 4-line DBI Interface I and Interface II.

Figure 8-98 DBI 4-Line Display Bus Serial Interface 8-bit Reading Operation Format



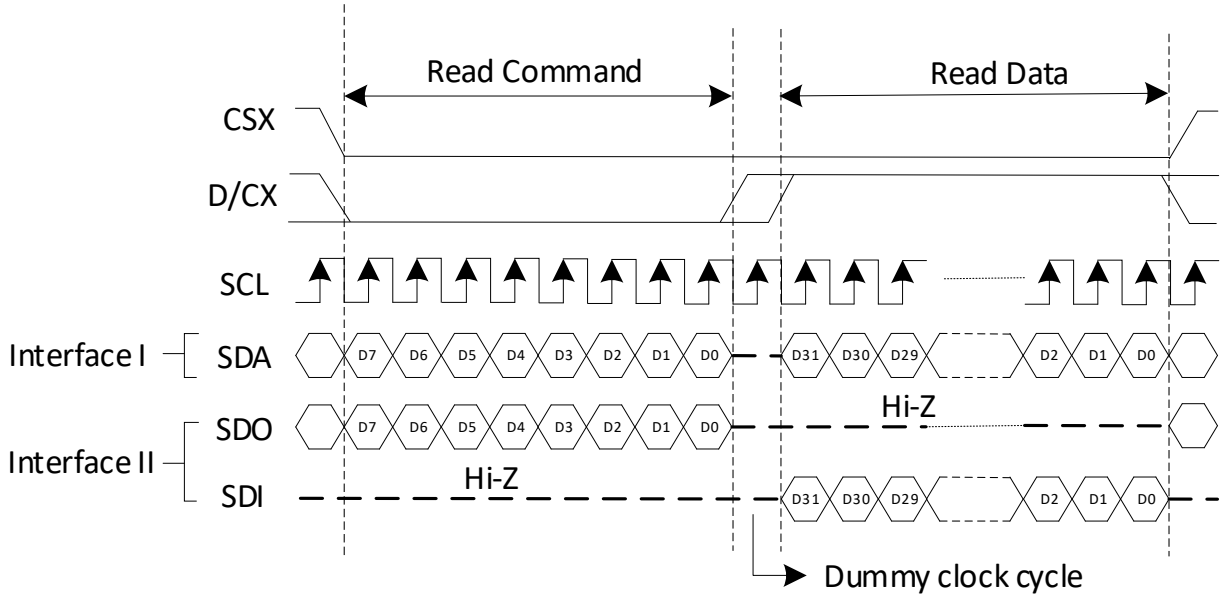
The following figure shows the 24 bits reading operation format of 4-line DBI Interface I and Interface II.

Figure 8-99 DBI 4-Line Display Bus Serial Interface 24-bit Reading Operation Format



The following figure shows the 32 bits reading operation format of 4-line DBI Interface I and Interface II.

Figure 8-100 DBI 4-Line Display Bus Serial Interface 32-bit Reading Operation Format



8.17.3.13 DBI 3-Line Interface Transmit Video Format

Figure 8-101 RGB111 3-Line Interface Transmit Video Format

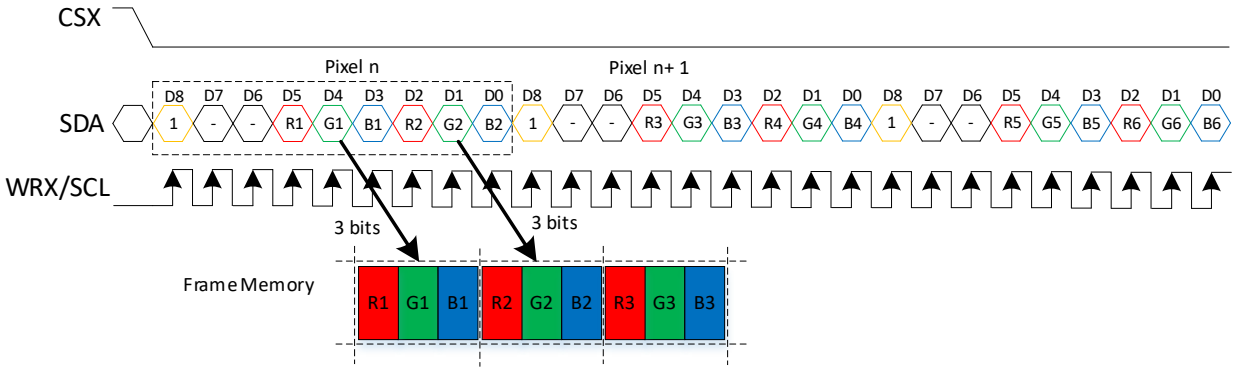
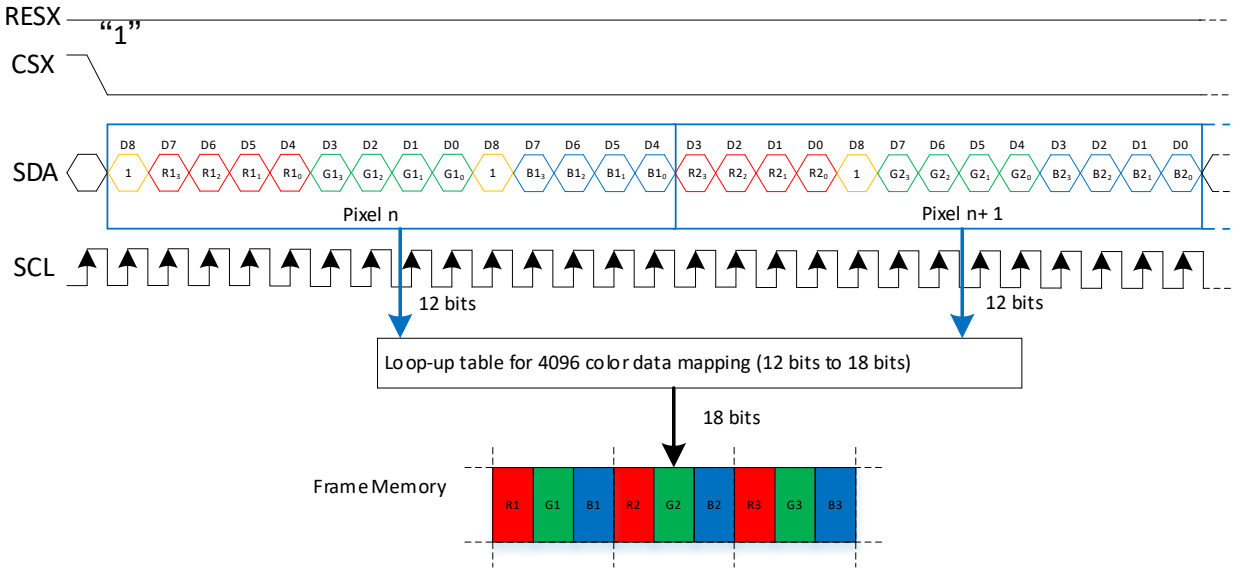
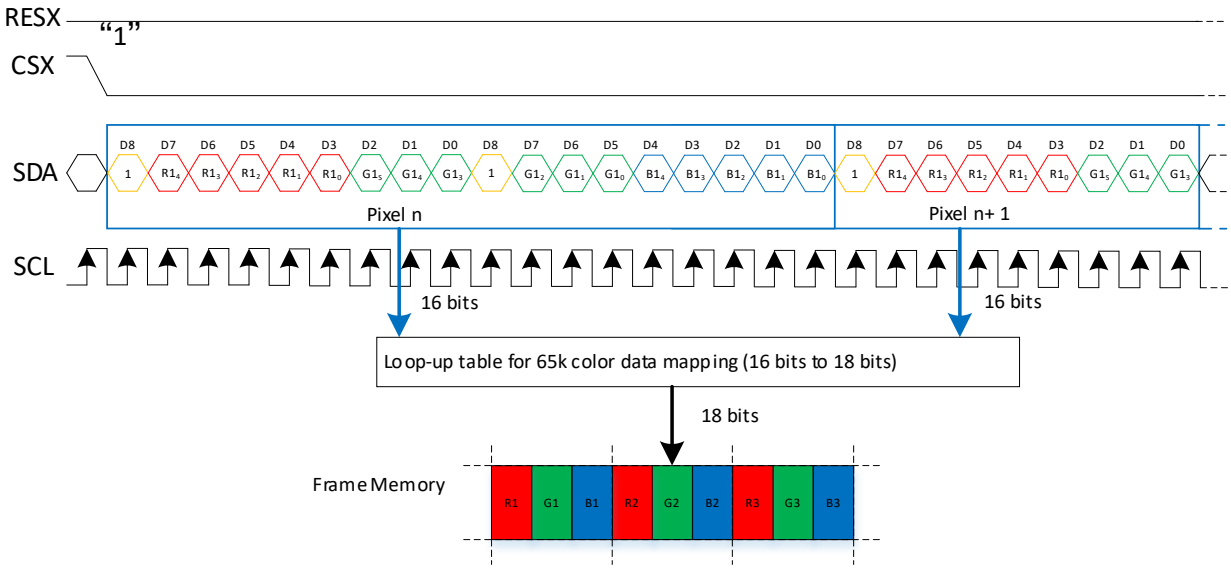


Figure 8-102 RGB444 3-Line Interface Transmit Video Format



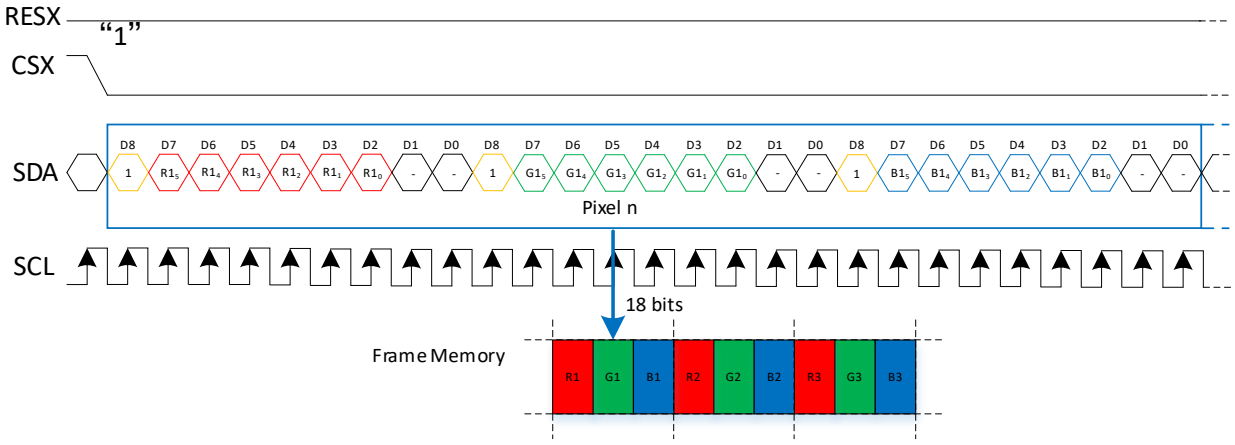
Note 1. Pixel data with 12-bit color depth information
Note 2. The most significant bits are: Rx3, Gx3 and Bx3
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-103 RGB565 3-Line Interface Transmit Video Format



Note 1. Pixel data with 16-bit color depth information
Note 2. The most significant bits are: Rx4, Gx5 and Bx4
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-104 RGB666 3-Line Interface Transmit Video Format



Note 1. Pixel data with 18-bit color depth information
Note 2. The most significant bits are: Rx5, Gx5 and Bx5
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

8.17.3.14 DBI 4-Line Interface Transmit Video Format

Figure 8-105 RGB111 4-Line Interface Transmit Video Format

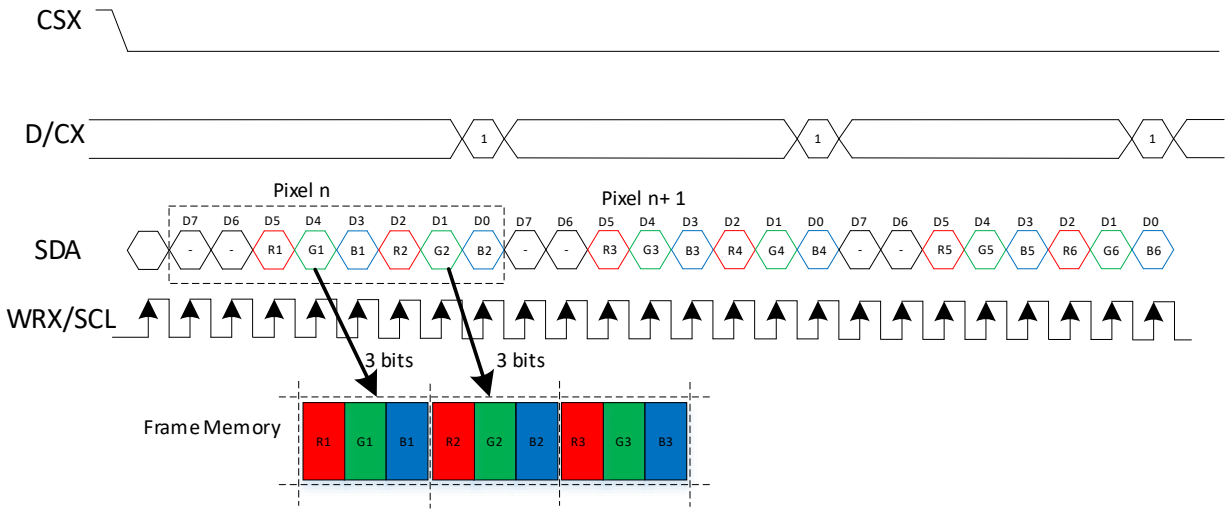
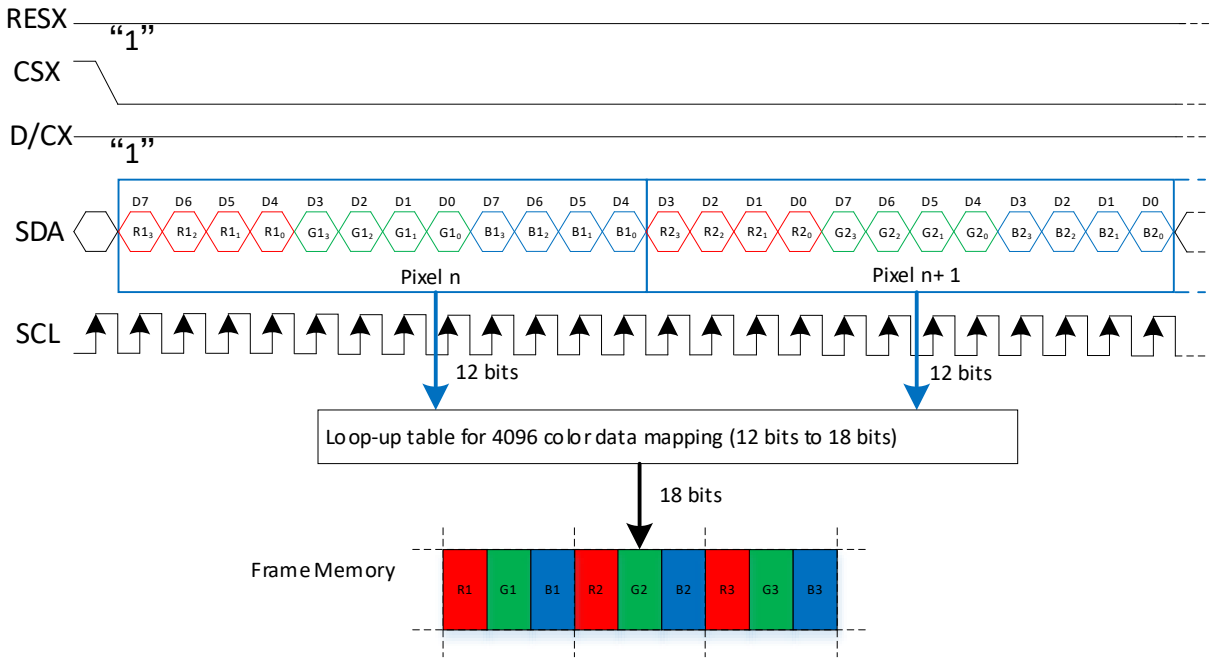
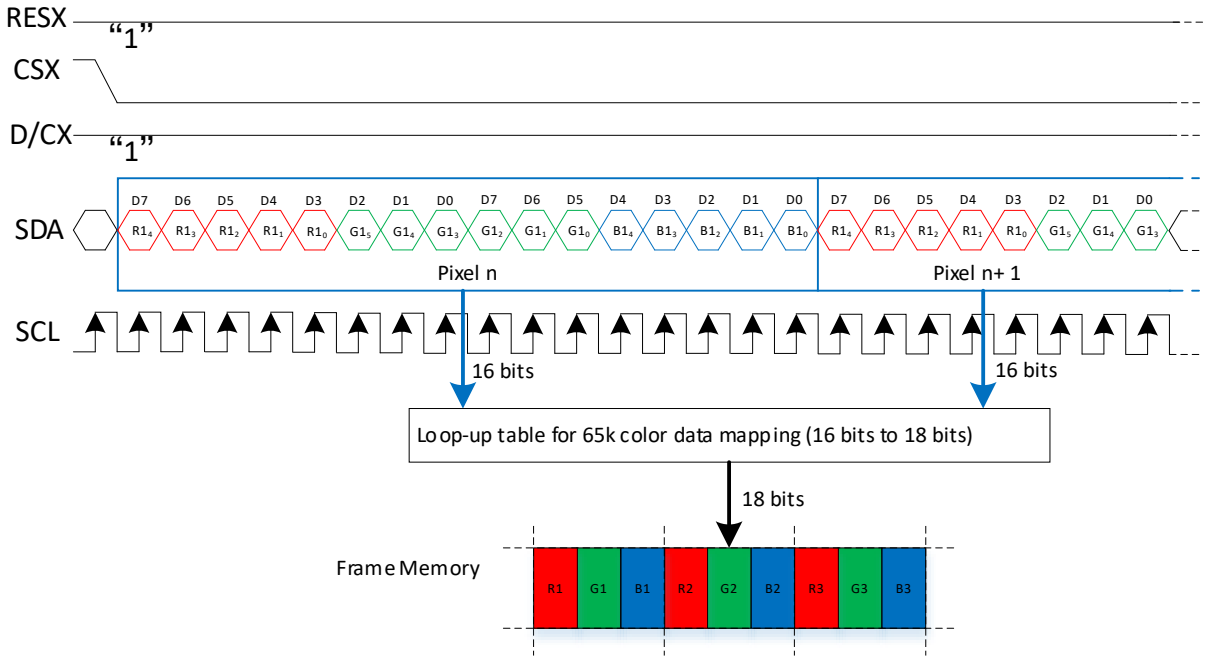


Figure 8-106 RGB444 4-Line Interface Transmit Video Format



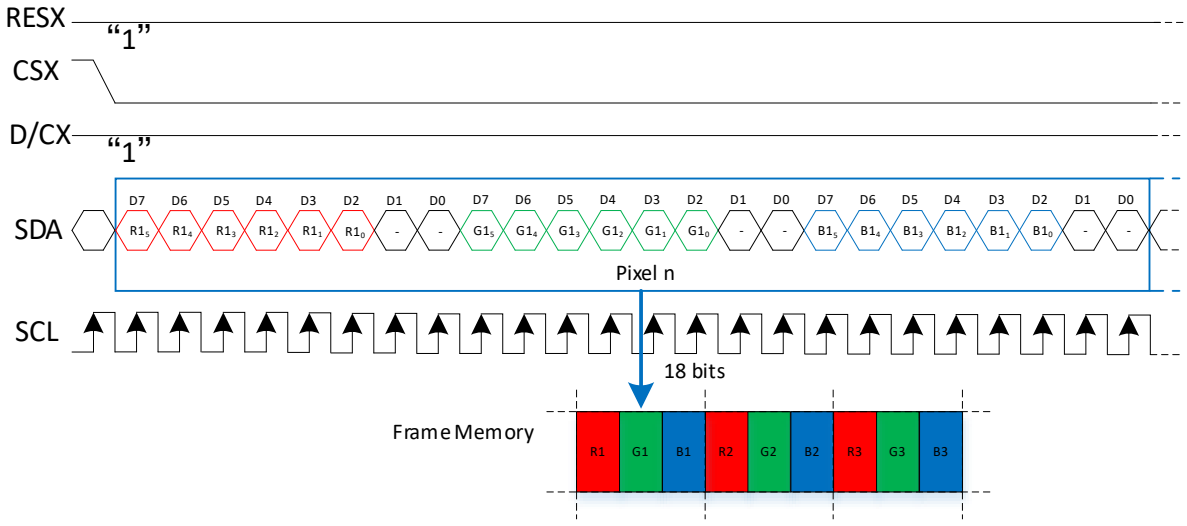
Note 1. Pixel data with 12-bit color depth information
Note 2. The most significant bits are: Rx3, Gx3 and Bx3
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-107 RGB565 4-Line Interface Transmit Video Format



Note 1. Pixel data with 16-bit color depth information
Note 2. The most significant bits are: Rx4, Gx5 and Bx4
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-108 RGB666 4-Line Interface Transmit Video Format

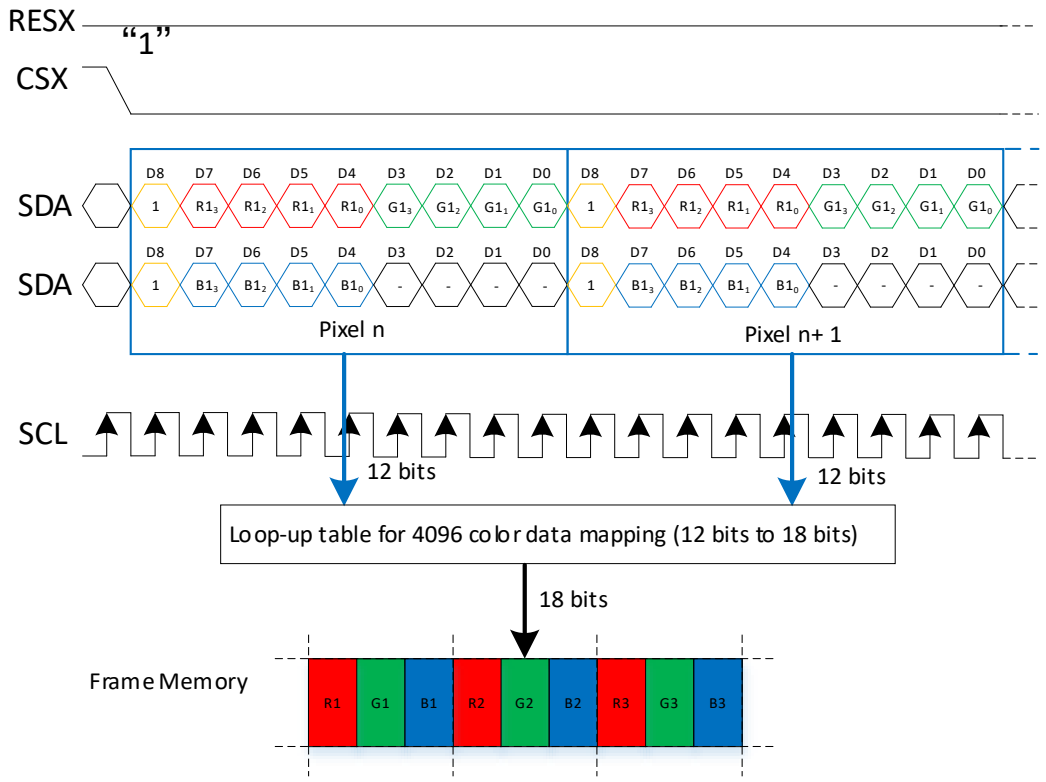


Note 1. Pixel data with 18-bit color depth information
Note 2. The most significant bits are: Rx5, Gx5 and Bx5
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

8.17.3.15 DBI 2 Data Lane Interface Transmit Video Format

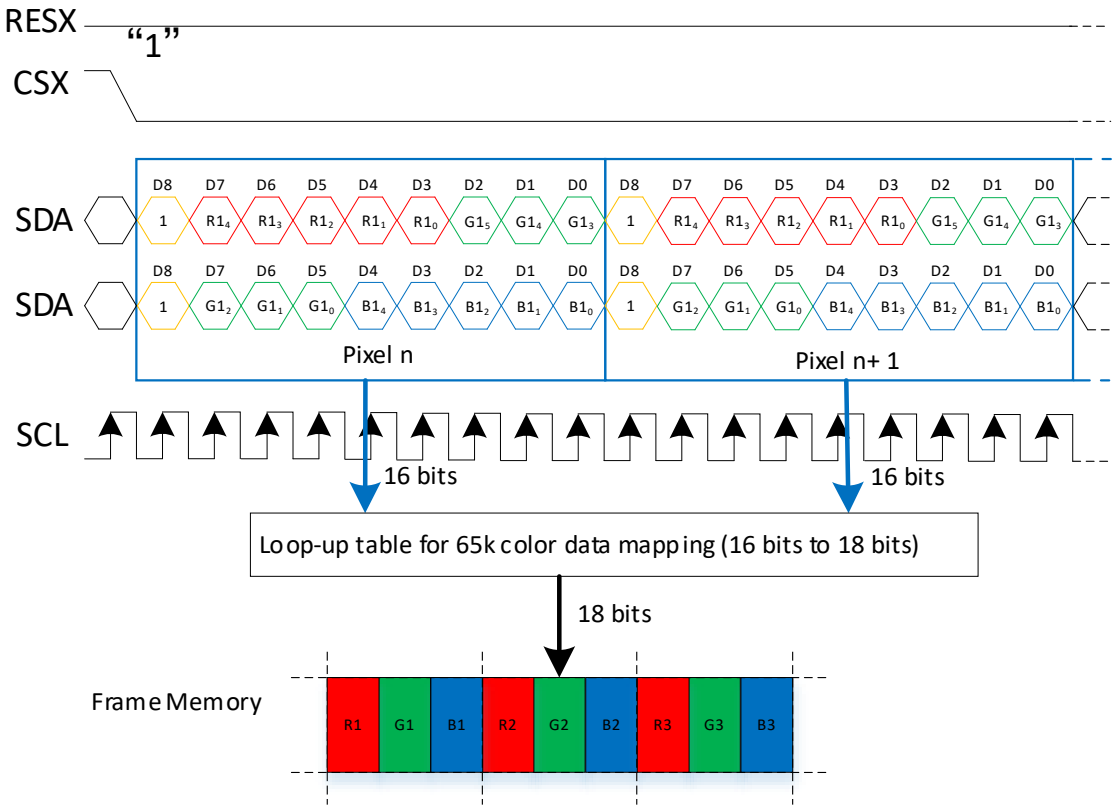
For RGB444:

Figure 8-109 RGB444 2 Data Lane Interface Transmit Video Format



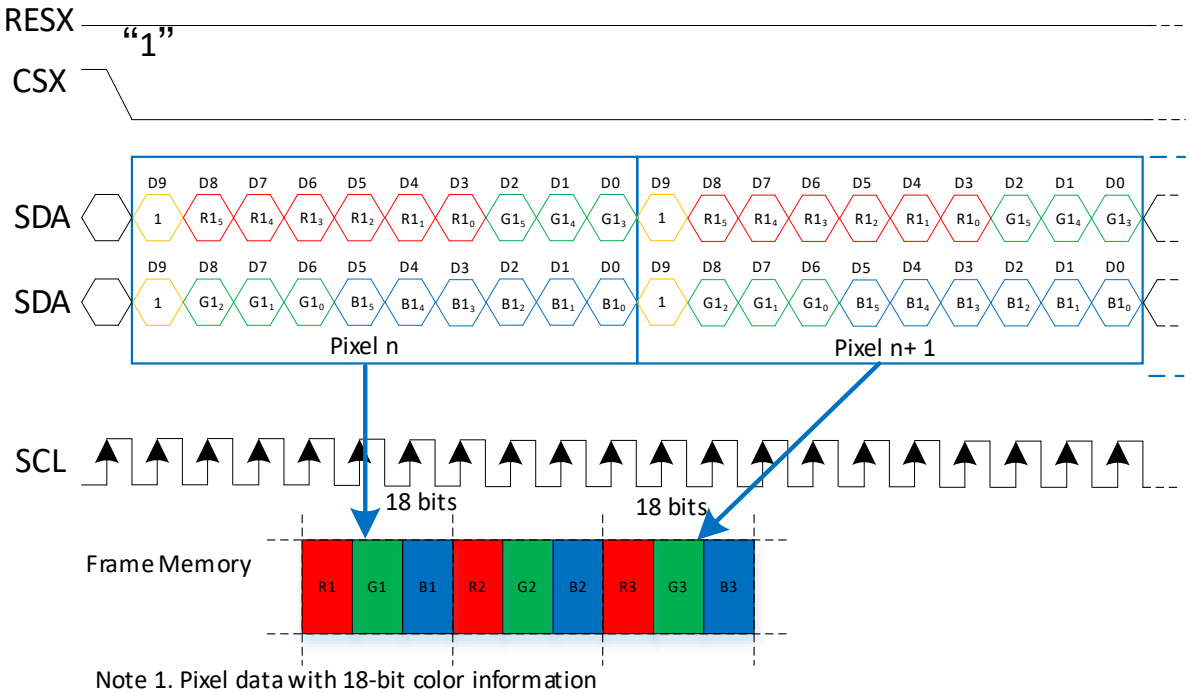
Note 1. Pixel data with 12-bit color information
Note 2. The most significant bits are: Rx3, Gx3 and Bx3
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-110 RGB565 2 Data Lane Interface Transmit Video Format



Note 1. Pixel data with 16-bit color information
Note 2. The most significant bits are: Rx4, Gx5 and Bx4
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-111 RGB666 2 Data Lane Interface Transmit Video Format 0



Note 1. Pixel data with 18-bit color information
Note 2. The most significant bits are: Rx5, Gx5 and Bx5
Note 3. The least significant bits are: Rx0, Gx0 and Bx0

Figure 8-112 RGB666 2 Data Lane Interface Transmit Video Format 1 (ilitek)

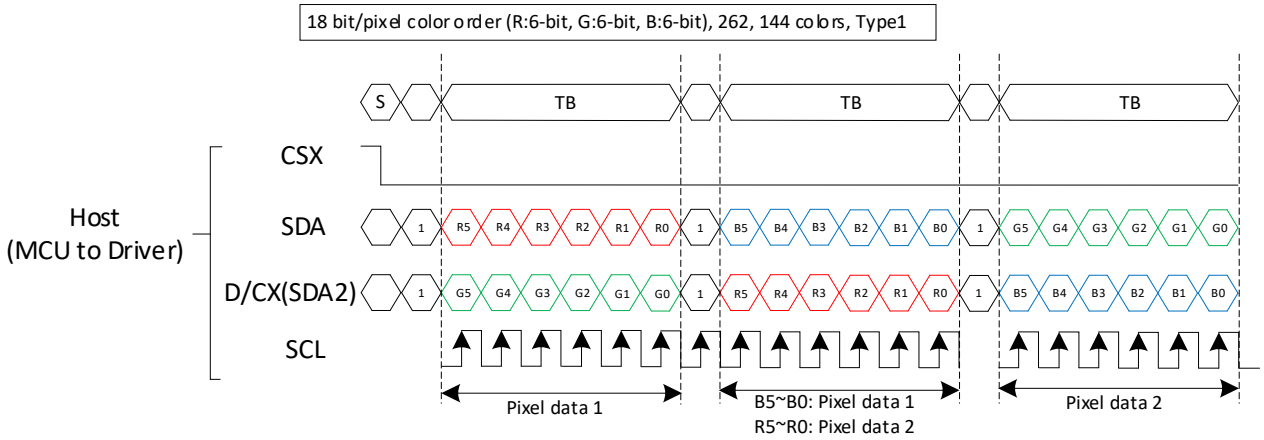


Figure 8-113 RGB666 2 Data Lane Interface Transmit Video Format 2 (New vision)

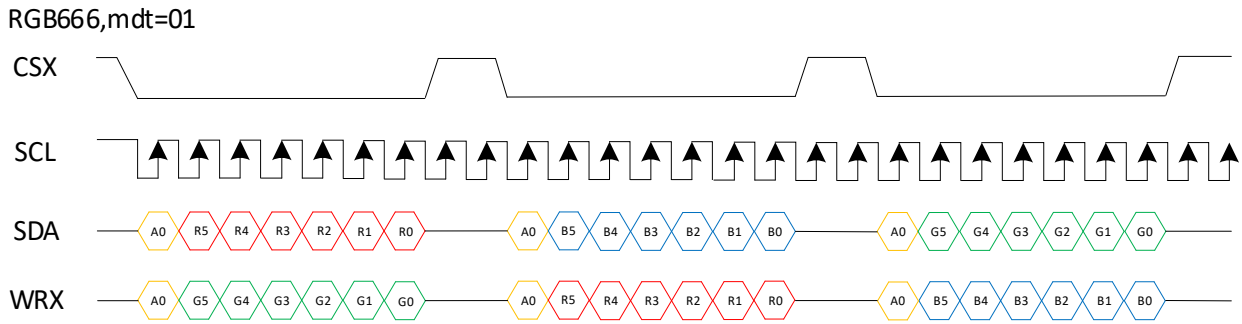
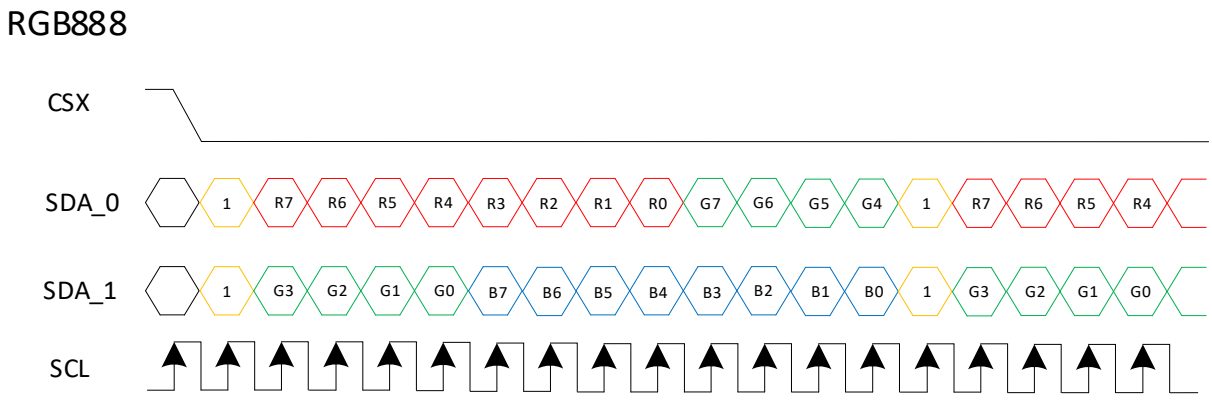


Figure 8-114 RGB888 2 Data Lane Interface Transmit Video Format



Note 1. Pixel data with 24-bit color information
Note 2. The most significant bits are: R7, G7 and B7
Note 3. The least significant bits are: R0, G0 and B0

8.17.4 Programming Guidelines

8.17.4.1 Writing/Reading Data Process Using SPI Mode

The SPI transfers serial data between the processor and the external device. CPU and DMA are the two main operational modes for SPI. For each SPI, the data is simultaneously transmitted (shifted out serially) and received (shifted in serially). The SPI has 2 channels, including the TX channel and

RX channel. The TX channel has the path from TX FIFO to the external device. The RX channel has the path from the external device to RX FIFO.

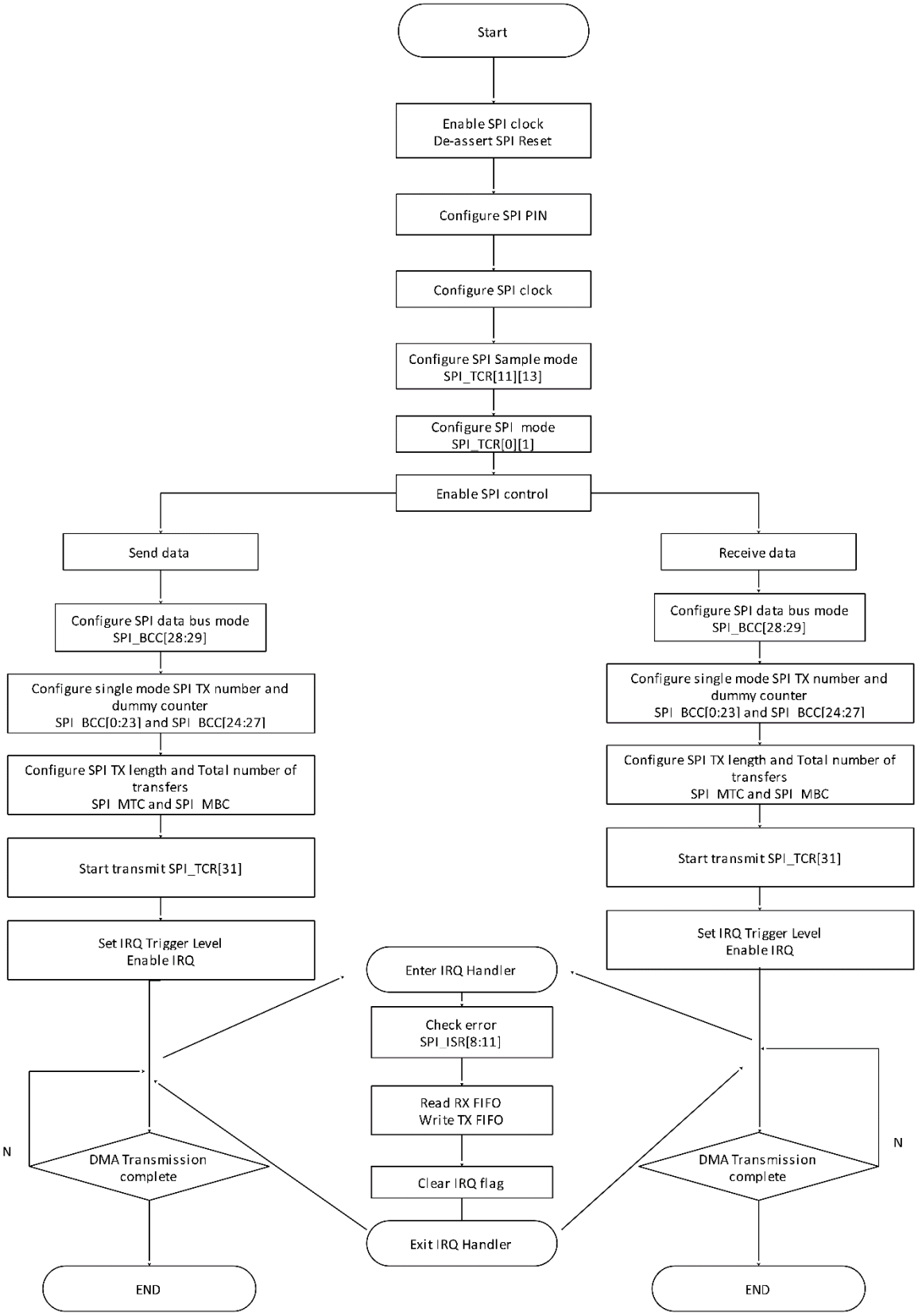
Write Data: CPU or DMA must write data on the SPI_TXD register, the data on the register are automatically moved to TX FIFO.

Read Data: To read data from RX FIFO, CPU or DMA must access the register SPI_RXD and data are automatically sent to the register SPI_RXD.

In CPU or DMA mode, the SPI sends a completed interrupt (SPI_ISR[TC]) to the processor at the end of each transfer.

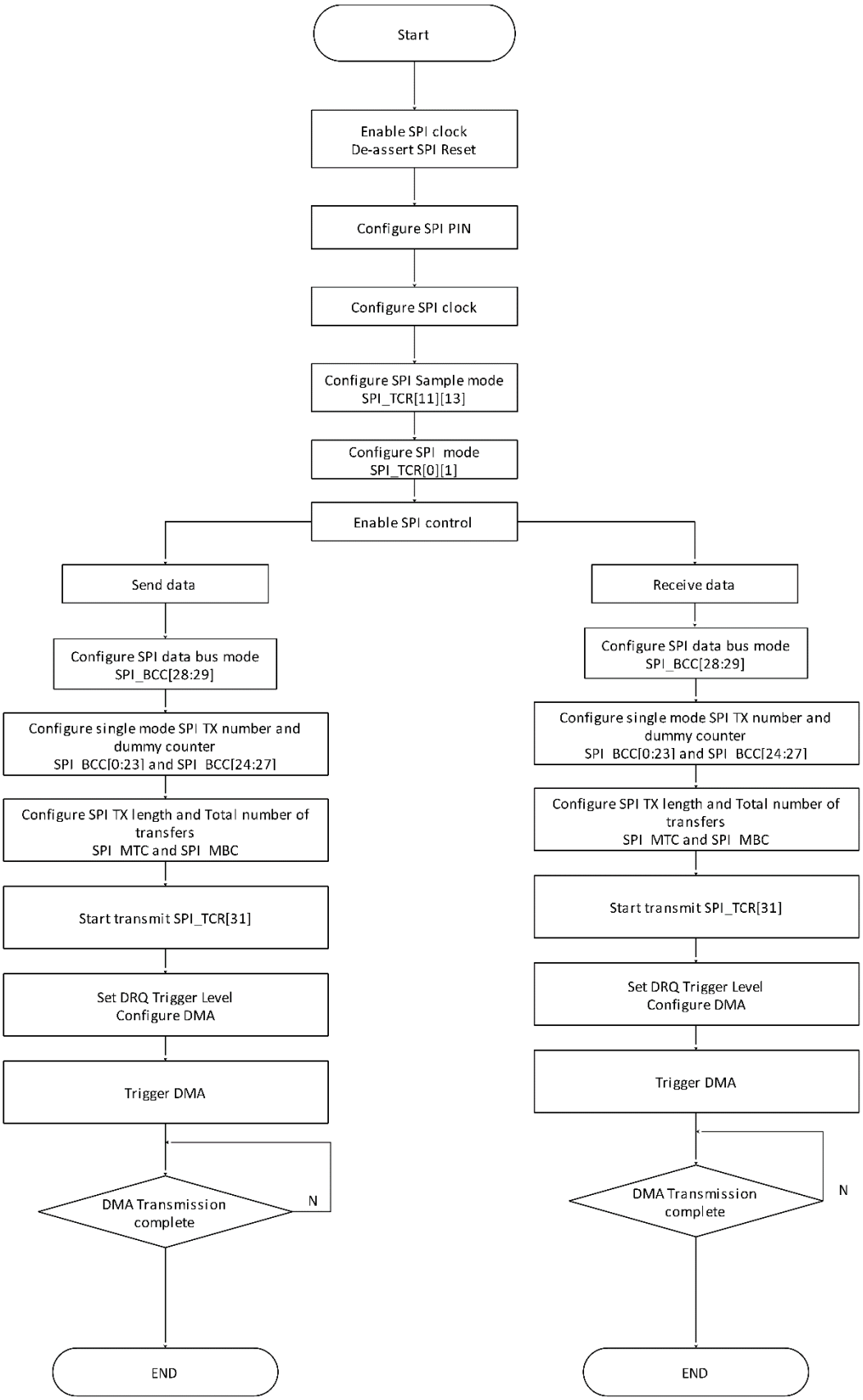
CPU Mode

Figure 8-115 SPI Write/Read Data in CPU Mode



DMA Mode

Figure 8-116 SPI Write/Read Data in DMA Mode



8.17.4.2 Transmitting Write Command Using DBI Mode

Step 1 Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

Step 2 Set the DBI EN MODE SEL (bit[30:29]) of DBI_CTL_1 (0x0104) to 0 to select the trigger mode of DBI.

Step 3 Configure the DBI_CTL_0 (0x0100).

a) Set DBI_CTL_0[Command Type] (bit31) to 0 to configure the writing command.

b) Set DBI_CTL_0[Write Command Dummy Cycles] (bit[30:20]) to configure the number of dummy cycles between commands.

c) Set DBI_CTL_0[Output Data Sequence] (bit19) to select the MSB or LSB.

d) Set DBI_CTL_0[Transmit Mode] (bit15) to 0 to select the command path.

e) Set DBI_CTL_0[Output Data Format] (bit[14:12]) to 0 to transmit the command.

f) Set DBI_CTL_0[DBI interface Select] (bit[10:8]) to select the DBI interface type.

g) The remaining values of the DBI_CTL_0 register remain the default value.

Step 4 Set DBI_CTL_1[DCX_DATA] (bit22) to 0 to send the command.

Step 5 DMA Path: Configure the SPI_FCR register (0x0018).

a) Set SPI_FCR[TF_DRQ_EN] (bit24) to 1 to enable TXFIFO DMA.

b) Set SPI_FCR[TX_TRIG_LEVEL] (bit[23:16]) to 255. It indicates the controller requests data from DMA if the remaining space of TX FIFO is greater than 255.

CPU Path: Write the command to be sent to the 0x200 address.

Step 6 Set SPI_GCR[DBI_EN] (bit4) to 1 to start transmitting the command.

Step 7 Wait until the TX FIFO underrun interrupt (SPI_ISR[TF_UDF]) is 1. It indicates that the command written to the TX FIFO is transmitted completely.

8.17.4.3 Transmitting Parameter Using DBI Mode

Step 1 Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

Step 2 Set the DBI EN MODE SEL (bit[30:29]) of DBI_CTL_1 (0x0104) to 0 to select the trigger mode of DBI.

Step 3 Configure the DBI_CTL_0 register (0x0100).

a) Set DBI_CTL_0[Command Type] (bit31) to 0 to configure the writing command.

b) Set DBI_CTL_0[Write Command Dummy Cycles] (bit[30:20]) to configure the number of dummy cycles between commands.

c) Set DBI_CTL_0[Output Data Sequence] (bit19) to select the MSB or LSB.

d) Set DBI_CTL_0[Transmit Mode] (bit15) to 0 to select the command path.

e) Set DBI_CTL_0[Output Data Format] (bit[14:12]) to 0 to transmit the command.

f) Set DBI_CTL_0[DBI interface Select] (bit[10:8]) to select the DBI interface type.

g) The remaining values of the DBI_CTL_0 register remain the default value.

Step 4 Set DBI_CTL_1[DCX_DATA] (bit22) to 1 to send the parameter.

Step 5 DMA Path: Configure the SPI_FCR register (0x0018).

a) Set SPI_FCR[TF_DRQ_EN] (bit24) to 1 to enable TXFIFO DMA.

b) Set SPI_FCR[TX_TRIG_LEVEL] (bit[23:16]) to 255. It indicates the controller requests data from DMA if the remaining space of TX FIFO is greater than 255.

CPU Path: Write the command to be sent to the 0x200 address.

Step 6 Set SPI_GCR[DBI_EN] (bit4) to 1 to start transmitting the command.

Step 7 Wait until the TX FIFO underrun interrupt (SPI_ISR[TF_UDF]) is 1. It indicates that the command written to the TX FIFO is transmitted completely.

8.17.4.4 Transmitting Video Using DBI Mode

Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

If the data is from the CPU path, the controller writes the command to be sent to the 0x0200 address by the AHB bus.

If the data is from the DMA path, configure DBI_CTL_1[DBI_FIFO_DRQ_EN] (bit15) to 1 and DBI_CTL_1[TX_TRIG_LEVEL] (bit[14:8]) to 64, which indicates the controller requests data from DMA if the remaining space of TX FIFO is greater than 64.

Software Trigger Mode

The software enables DBI_en_trigger when the edge interrupt of TE is detected.

After transmitting each frame image, the controller clears automatically the line_cnt, pixel_cnt and stops transmitting data.

Wait for the edge interrupt of TE, the software needs to enable DBI_en_trigger, in circulation.

The operation process is as follows.

Step 1 Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

Step 2 Set the DBI EN MODE SEL (bit[30:29]) of DBI_CTL_1 (0x0104) to 1 to select the software trigger mode.

Step 3 Configure the DBI_CTL_0 register (0x0100).

a) Set DBI_CTL_0[Command Type] (bit31) to 0 to set the writing command.

b) Set DBI_CTL_0[Write Command Dummy Cycles] (bit[30:20]) to configure the number of dummy cycles between commands.

c) Set DBI_CTL_0[Output Data Sequence] (bit19) to select the MSB or LSB.

d) Set DBI_CTL_0[Transmit Mode] (bit15) to 1 to select the image path.

e) Set DBI_CTL_0[Output Data Format] (bit[14:12]) to select RGB111//444/565/666/888.

f) Set DBI_CTL_0[DBI interface Select] (bit[10:8]) to select the DBI interface type.

g) The remaining values of the DBI_CTL_0 register remain the default value.

Step 4 Set DBI_CTL_1[DCX_DATA] (bit22) to 0 to send the image data.

Step 5 Configure DBI_Video_Size (0x110) according to the sent image size.

Step 6 Configure DBI_CTL_2 (0x0108) to set the TE-related parameter.

Step 7 Detect the TE interrupt of the DBI_INT (0x0120) register.

Step 8 Configure DBI_CTL_1[DBI_soft_trigger] to 1.

Timer Trigger Mode

The software configures timer_en to enable timer counting, and when the counter reaches the specified value, the DBI_EN automatically can be enabled to start transmitting data.

After transmitting each frame image, the controller clears automatically the line_cnt, pixel_cnt, and stops transmitting data.

The timer starts counting again. When the counter reaches the specified value, the controller automatically enables DBI_EN, and in circulation until the software turns off the timer_en.

The operation process is as follows.

Step 1 Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

Step 2 Set the DBI EN MODE SEL (bit30:29) of DBI_CTL_1 (0x0104) to 2 to select the timer trigger mode.

Step 3 Configure the DBI_CTL_0 register (0x0100).

a) Set DBI_CTL_0[Command Type] (bit31) to 0 to set the writing command.

b) Set DBI_CTL_0[Write Command Dummy Cycles] (bit[30:20]) to configure the number of dummy cycles between commands.

c) Set DBI_CTL_0[Output Data Sequence] (bit19) to select the MSB or LSB.

d) Set DBI_CTL_0[Transmit Mode] (bit15) to 1 to select the image path.

e) Set DBI_CTL_0[Output Data Format] (bit[14:12]) to select RGB111/444/565/666/888.

f) Set DBI_CTL_0[DBI interface Select] (bit[10:8]) to select the DBI interface type.

g) The remaining values of the DBI_CTL_0 register remain the default value.

Step 4 Set DBI_CTL_1[DCX_DATA] (bit22) to 0 to send the image data.

Step 5 Configure DBI_Video_Size (0x110) to transmit the image size.

Step 6 Configure the related parameter of DBI_Timer (0x10C).

TE Trigger Mode

When the edge changes of the TE are detected (The rising and falling edges are optional), the DBI_EN automatically can be enabled to start transmitting data.

After transmitting each frame image, the controller clears automatically the line_cnt, pixel_cnt, and stops transmitting data.

When the edge changes of the TE are detected (The rising and falling edges are optional), the DBI_EN automatically can be enabled to start transmitting data until the software shuts down TE_EN or the screen no longer sends TE signals.

The operation process is as follows.

Step 1 Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

Step 2 Set the DBI EN MODE SEL (bit30:29) of DBI_CTL_1 (0x0104) to 3 to select the TE Configure the DBI_CTL_0 register (0x0100).

Step 3 Set DBI_CTL_0[Command Type] (bit31) to 0 to set the writing command.

a) Set DBI_CTL_0[Write Command Dummy Cycles] (bit[30:20]) to configure the number of dummy cycles between commands.

b) Set DBI_CTL_0[Output Data Sequence] (bit19) to select the MSB or LSB.

c) Set DBI_CTL_0[Transmit Mode] (bit15) to 1 to select the image path.

d) Set DBI_CTL_0[Output Data Format] (bit[14:12]) to select RGB111/444/565/666/888.

e) Set DBI_CTL_0[DBI interface Select] (bit[10:8]) to select the DBI interface type.

f) The remaining values of the DBI_CTL_0 register remain the default value.

Step 4 Configure DBI_CTL_1[DCX_DATA] (bit22) to 0 to send the image data.

Step 5 Configure DBI_Video_Size (0x0110) to transmit the image size.

Step 6 Configure DBI_CTL_2 (0x0108) to set the TE-related parameter.

8.17.4.5 Transmitting Read Command and Read Data Using DBI Mode

Step 1 Set the SPI_DBI_MODE_SEL (bit3) of SPI_GCR (0x0004) to 1 to select DBI mode.

Step 2 Set the DBI EN MODE SEL (bit[30:29]) of DBI_CTL_1 (0x0104) to 0.

Step 3 Configure the DBI_CTL_0 register (0x0100).

a) Set DBI_CTL_0[Command Type] (bit31) to 0 to set the reading command.

b) Set DBI_CTL_0[Output Data Sequence] (bit19) to select the MSB or LSB.

c) Set DBI_CTL_0[Transmit Mode] (bit15) to 0 to select the command path.

d) Set DBI_CTL_0[Output Data Format] (bit[14:12]) to 0.

e) Set DBI_CTL_0[DBI interface Select] (bit[10:8]) to select the DBI interface type.

f) The remaining values of the DBI_CTL_0 register remain the default value.

Step 4 Configure the DBI_CTL_1 register (0x0104).

a) Configure DBI_CTL_1[DCX_DATA] (bit22) to 0 to send the command.

b) Configure DBI_CTL_1[Read_MSB_First] (bit20) to select whether the first bit of the read data is the highest or lowest bit of data.

c) Configure DBI_CTL_1[Read Data Number of Bytes] to set the byte number to be read.

d) Configure DBI_CTL_1[Read Command Dummy Cycles] to set the dummy cycle between the read command and the read data, when the dummy cycle is complete, the data starts to be sampled.

Step 5 DMA Path: Configure the SPI_FCR register (0x0018).

a) Set SPI_FCR[RF_DRQ_EN] (bit8) to 1 to enable RXFIFO DMA.

b) Set SPI_FCR[RX_TRIG_LEVEL] (bit[7:0]) to 32, which indicates the controller requests receiving data from DMA if the data of the RX FIFO is greater than 64.

CPU Path: Read data in RX FIFO from the 0x0300 address.

Step 6 Set SPI_GCR[DBI_EN] (bit4) to 1 to start transmitting command.

Step 7 Wait until DBI_INT[RD_DONE_INT] is 1. It indicates that the data is read completely.

8.17.5 Register List

Module Name	Base Address
SPI1	0x0402 6000

Register Name	Offset	Description
SPI_GCR	0x0004	SPI Global Control Register
SPI_TCR	0x0008	SPI Transfer Control register
SPI_IER	0x0010	SPI Interrupt Control register
SPI_ISR	0x0014	SPI Interrupt Status register
SPI_FCR	0x0018	SPI FIFO Control register
SPI_FSR	0x001C	SPI FIFO Status register
SPI_WCR	0x0020	SPI Wait Clock Counter register
SPI_SAMP_DL	0x0028	SPI Sample Delay Control Register
SPI_MBC	0x0030	SPI Master Burst Counter register

8.18 SPI Flash controller (SPIFC)

8.18.1 Overview

The SPI Flash Controller (SPIFC) is a synchronous, serial communication interface which allows rapid data communication with fewer software interrupts. Different from SPI, this IP is typically designed for higher speed Flash devices and it only works at Master mode.

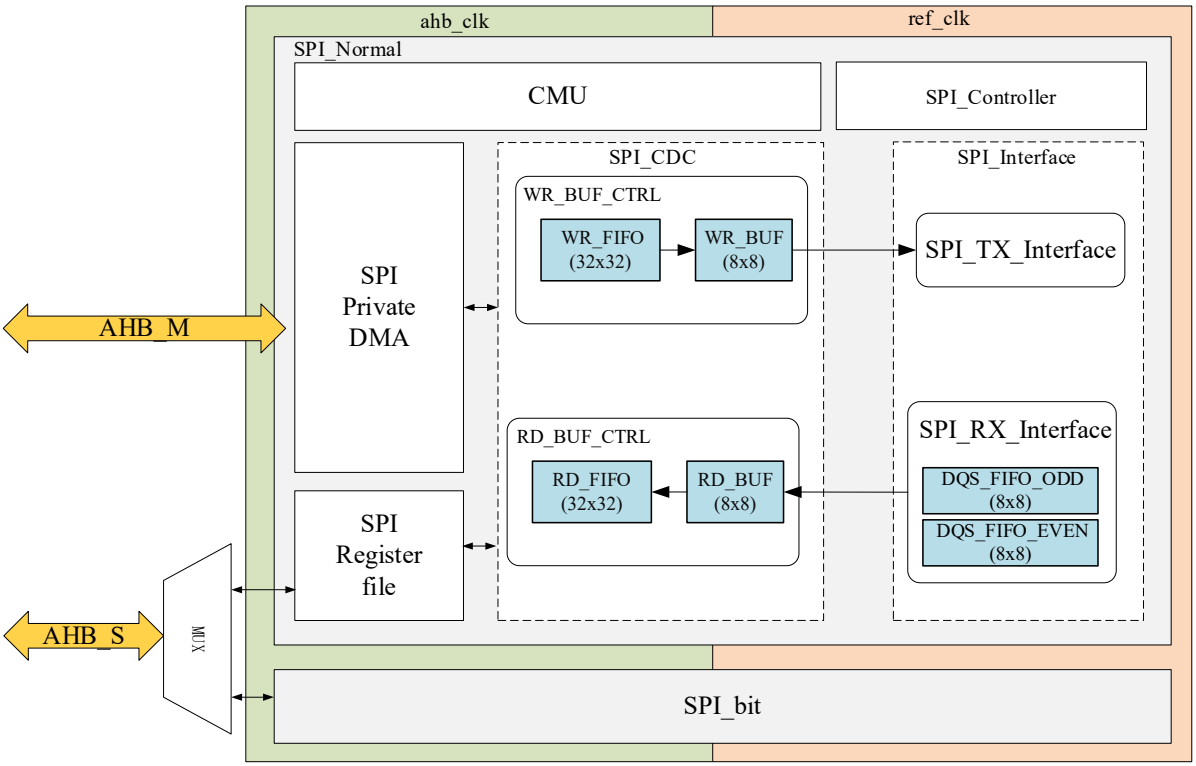
The SPI Flash Controller has the following features:

- Supports multiple SPI modes
  - Standard SPI
  - Dual-Input/Dual-Output SPI and Dual-I/O SPI
  - Quad-Input/Quad-Output SPI, Quad-I/O SPI, and QPI
  - Octal-Input/Octal-Output SPI, Octal-I/O SPI, and OPI
  - 3-wire SPI with programmable serial data frame length of 1 bit to 32 bits
- Supports STR mode and DTR mode, and DTR mode supports DQS signal
- High Speed Clock Frequency
  - 150MHz for STR Mode
  - 100MHz for DTR Mode
- Software Write Protection
  - Write protection for all/portion of memory via software
  - Top/Bottom Block protection
- Programmable delay between transactions
- Support Mode0, Mode1, Mode2 and Mode3
- Support control signal configuration
  - Up to four chip selects to support multiple peripherals
  - Polarity and phase of the Chip Select (SPI_SS) and SPI Clock (SPI_SCLK) are configurable

8.18.2 Block Diagram

The following figure shows a block diagram of the SPI Flash Controller.

Figure 8-117 SPI_Flash Block Diagram



SPI Flash Controller contains the following sub-blocks:

Table 8-57 SPI Flash Controller Sub-blocks

Sub-block	Description
SPI_Register_File	Responsible for implementing registers configuration through AHB1 bus.
SPI_Private_DMA	Private DMA for SPI, which contains AHB master and supports TXFIFO(WR_FIFO) and RXFIFO(RD_FIFO).
CMU	Responsible for generating internal clock; Implementing sckr delay, clock sourcess selection, clock gating, and scan; supporting synchronous release of asynchronous reset and scan.
SPI_CDC	SPI cross-clock module, in which bit width transition finishes. (TX: 32bit->8bit; RX: 8bit->32bit) WR_BUF: Cache the write SPI data of AHB, and write them to WR_FIFO. RD_BUF: Cache the read SPI data of RD_FIFO to AHB. WR_FIFO: SPI write data FIFO (SRAM 32x32) RD_FIFO: SPI read data FIFO (SRAM 32x32) in the normal mode.
SPI_Controller	SPI controlling center. It generates TX/RX controlling signal in the course of SPI communication.
SPI_Interface	Standard SPI interface. It is responsible for receiving and transmitting data with Devices.

Sub-block	Description
SPI_Bit	the processing module in SPI 3-wire mode.

8.18.3 Functional Description

8.18.3.1 External Signals

The following table describes the external signals of SPI Flash Controller. When using SPI Flash Controller, the corresponding PADs are selected as SPI Flash Controller function via section 8.6 GPIO.

Table 8-58 SPI Flash Controller External Signals

Signal Name	Description	Type
SPIF-CS0	SPI Peripheral Chip Select Signal, Low Active	O
SPIF-CLK	SPI Master Mode Clock Output	O
SPIF-MOSI	SPI Master Data Out, Slave Data In	I/O
SPIF-MISO	SPI Master Data In, Slave Data Out	I/O
SPIF-DQS	Data Strobe Signal	I
SPIF-D[7:4]	SPI Master Mode Data in Octal Mode	I/O
SPIF-WP	SPI Write Protect, Low Active	I/O
SPIF-HOLD	SPI Hold Signal	I/O

8.18.3.2 Clock Sources

The SPI_Flash controller gets 5 different clock sources and users can select one of them to make SPI Flash Controller clock source. The following table describes the clock sources for SPI Flash Controller. For more details on the clock setting, configuration, and gating information, see section 2.6 Clock Controller Unit (CCU).

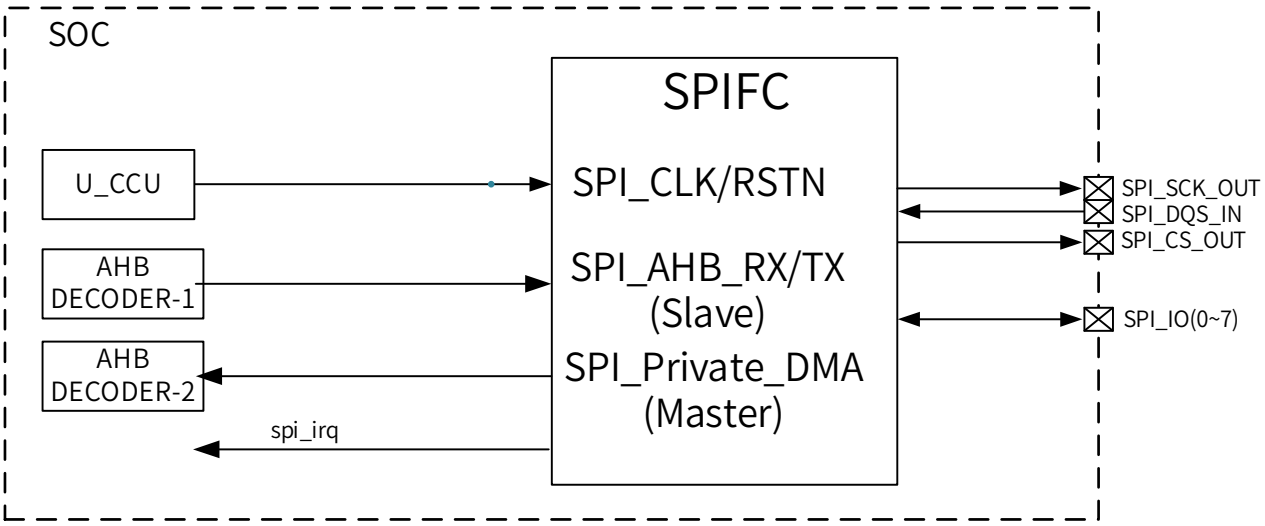
Table 8-59 SPI Flash Controller Clock Sources

Clock sources	Description	Clock module
HOSC	24 MHz Crystal	CCU
PERI0_400M	Peripheral Clock, default value is 400 MHz.	
PERI0_300M	Peripheral Clock, default value is 300 MHz.	
PERI1_400M	Peripheral Clock, default value is 400 MHz.	
PERI1_300M	Peripheral Clock, default value is 300 MHz.	

8.18.3.3 Typical Application

The following figure shows the application block diagram when the SPI master device is connected to a slave device.

Figure 8-118 Typical Application



The SPI Flash Controller is running in Master device. SPI_SCK is generated and transmitted to external device. The data from the TX FIFO is routed to the MOSI pin. The data from slave is received on the MISO pin and sent to RX FIFO. Chip Select(SPI_CS) signal is active in low level. SPI_CS must be set to low before data are transmitted or received.

8.18.3.4 SPI Flash Controller feature list

Table 8-60 SPI Flash Controller Feature List

SPI mode		Feature List					
		STR	DTR	DTR-RX-DQS	SCK_MODE	Address Size	Description
Bit Mode	3-wire 4-wire	√	×	×	mode0	×	/
Standard SPI	1-1-1-1	√	√	√	• mode 0/1/2/3 (STR) • mode0 (DTR)	24bit/32 bit	• 1-1-1-1: cmd-addr-mode-data. • mode is optional and can be turned off.
Dual SPI	1-1-1-2	√	√	√			

SPI mode		Feature List					
		STR	DTR	DTR-RX-DQS	SCK_MODE	Address Size	Description
	1-1-2-2						
	1-2-2-2						
	2-2-2-2						
Quad SPI	1-1-1-4	√	√	√			
	1-1-4-4						
	1-4-4-4						
	4-4-4-4						
Octal SPI	1-1-1-8	√	√	√			8-wire DTR only supports the following: ADDR-24+MODE ADDR-32
	1-1-8-8						
	1-8-8-8						
	8-8-8-8						

8.18.3.5 SPI Flash Controller Clock

SPI includes two clock domains: ahb_clk and spi_clk:

- The functions in ahb_clk: parameter analysis (ahb_register) and DMA.
- The functions in spi_clk: cross domain clock, main control unit, the communication between SPI-TX/RX and external devices.

The clock management unit (CMU) divides the external SPI reference clock and gets the o_spi_clk as the internal clock. Based on the clock properties configured by the CPU, the sckt/sckr is gotten to be used as communication clocks for the SPI_TX_INTERFACE/SPI_RX_INTERFACE.

The clock properties include：

- SPI Clock Mode
- DQS EN
- STR or DTR

When the SPI runs at a higher clock frequency, sckr may sample the wrong data because of lane delay. Thus, before sampling, the sckr should be processed by the receive clock latency.

SPI Flash Controller Clock Mode

The SPI Flash controller supports 4 different modes for data transfer. Software can select one of the four modes in which the SPI works by setting the bit5(SPI_CPOL) and bit4(SPI_CPHA) of SPI Global Control Register[0x0004].

The SPI_CPOL defines the signal polarity when SPI_SCLK is in the idle state. The SPI_SCLK is high level when POL is '1 'and it is low level when POL is '0'. The SPI_CPHA decides whether the leading

edge of SPI_SCLK is used for setup or sample data. The leading edge is used for setup data when PHA is '1' and for sample data when PHA is '0'. The four kind of modes are listed Table:

Table 8-61 SPIFC Modes with Clock Polarity and Phase

SPI Mode	POL	PHA	Leading Edge	Trailing Edge
0	0	0	Rising, Sample	Falling, Setup
1	0	1	Rising, Setup	Falling, Sample
2	1	0	Falling, Sample	Rising, Setup
3	1	1	Failing, Setup	Rising, Sample

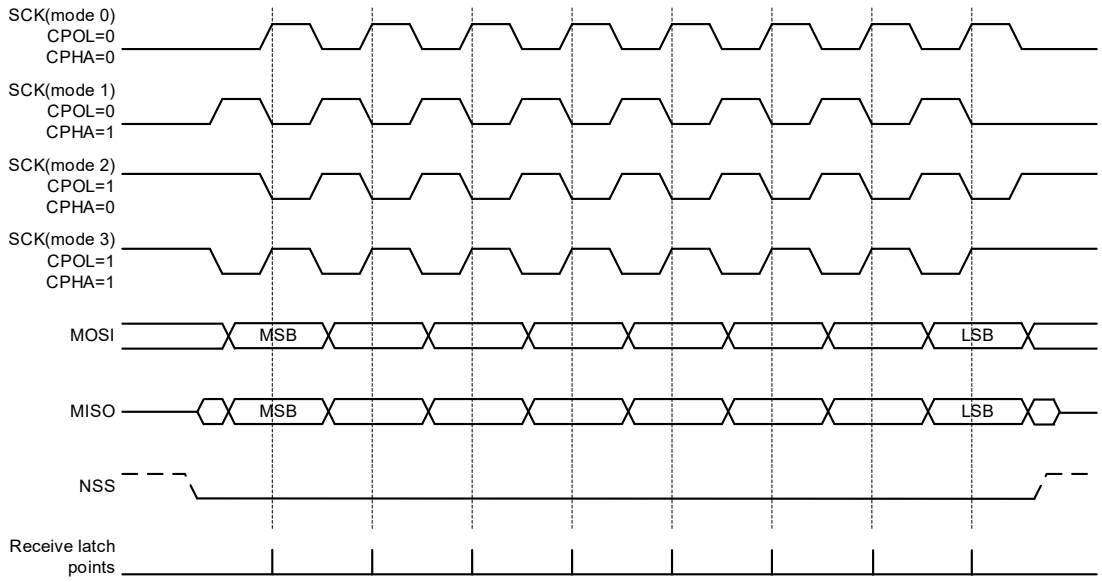
During Phase 0, Polarity 0 and Phase 1, Polarity 1 operations, output data changes on the falling edge and input data is shifted in on the rising edge.

During Phase 1, Polarity 0 and Phase 0, Polarity 1 operations, output data changes on the rising edges and is shifted in on falling edges.

SPI Bit Mode only supports Phase 0, Polarity 0 operation; the most significant bit (MSB) of data is transmitted first which is not configurable.

The following figure describe four waveforms for SPI_SCLK.

Figure 8-119 SPI Transfer Mode

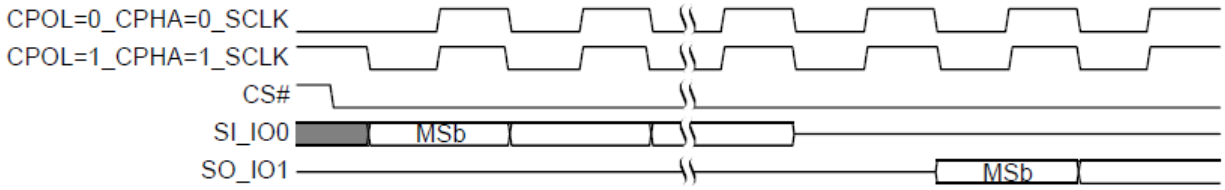


For MSB first

Single Transfer Rate (STR)

In mode 0 and mode 3, the input data of devices will be latched at the rising edge of SCK, and the output data will be used at the falling edge of SCK.

Figure 8-120 SPI STR Transfer



Double Transfer Rate (DTR)

As with the STR command, the instruction bit is latched at the rising edge of the clock in the DTR command, but the address and input data are latched at the dual edge. After the instruction bit is latched at the falling edge of SCK, the first address bit will be latched at the next rising edge of SCK. The first output data bit will be sent at the falling edge of the last access latency period.

As with the STR command, the SCK period is the cycle between two adjacent SCK falling edges. In mode 0, the SCK is already at a low level when some command starts to be executed, thus the first SCK period during the command execution indicates the cycle from the falling edge of CS# to the first falling edge of SCK.

Figure 8-121 SPI DTR Transfer Example, 1-4-4

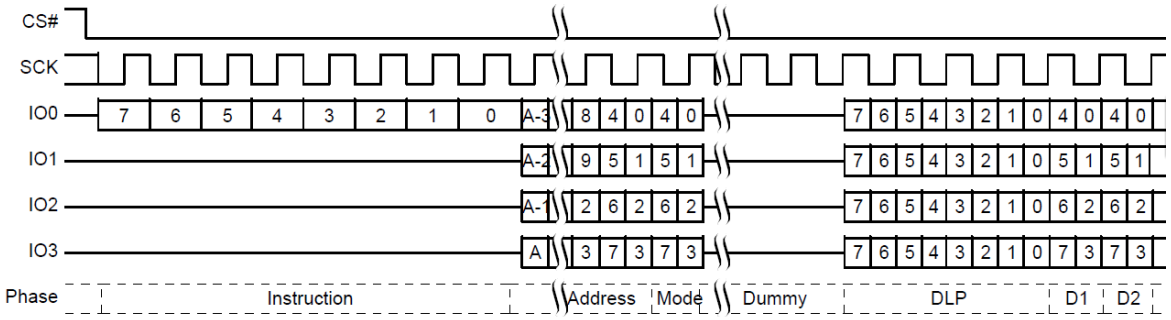
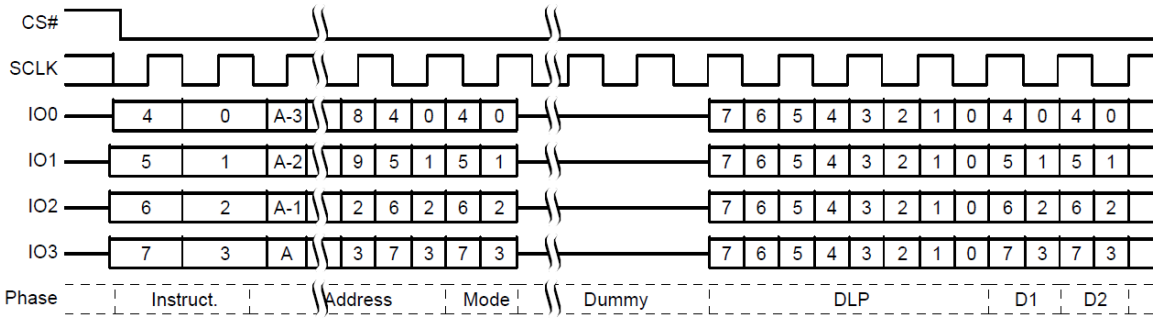


Figure 8-122 SPI DTR Transfer Example, 4-4-4

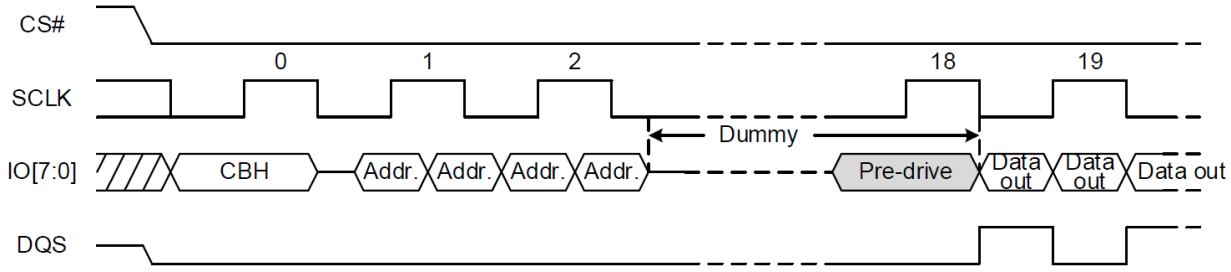


DQS

DQS（DATA Strobe Signal）signal indicates input/output data valid for DTR modes and is required to support high-speed data. When data strobe function is enabled, DQS signal is driven to ground

once CS# goes LOW till the device is driving output data, in which case DQS toggles to synchronize data output. When data strobe function is not enabled, DQS signal will not be driven.

Figure 8-123 SPI DTR Transfer with DQS Input Clock Signal



8.18.3.6 SPI Run Clock and Sample Mode

SCKR Delay through Digital Adjustment

To realize the SCKR delay, connect the clk_gate and clk_xor in series to control the opening time of the EN terminal of clk_gate and the polarity of the EN terminal of clk_xor. The specific steps are as follows:

Step 1 Enable the EN terminal delay of the first-level clk_gate module to realize a delay of 1 sclk. (the effective range is 0-3 sclk)

Step 2 Make the EN terminal of the second-level clk_xor module differ in polarity to realize a delay of 0.5 sclk. (the effective values are 0 sclk and 0.5 sclk)

SCKR Delay through Analog Adjustment

There are delay chains in SPI, used to generate delay to make proper timing between internal SPI clock signal and data signals. Delay chain is made up with 64 delay cells. The delay time of one delay cell can be estimated through delay chain calibration.

Take RX delay chain as an example: the steps to calibrate delay chain are as follows:

Step 1 Configure a proper clock for the SPI Flash Controller. Calibration delay chain is based on the clock for SPI FLASH CONTROLLER from Clock Controller Unit (CCU)

Step 2 Set proper initial delay value to (SPI Timing Configure Register, 0x000C). Write 0x60 to this register to set initial delay value 0x20 to delay chain. Then write 0x0 to delay control register to clear this value.

Step 3 Write 0x80 to SPI Timing Configure Register to start calibrate delay chain.

Step 4 Wait until the flag (Bit7 in SPI Timing Delay State Register 0x0010) of calibration done is set. The number of delay cells is shown at Bit5-Bit0 in SPI Timing Delay State Register. The delay time generated by these delay cells is equal to the cycle of SPI FLASH CONTROLLER's clock nearly. This value is the result of calibration.

Step 5 Calculate the delay time of one delay cell according to the cycle of SPI FLASH CONTROLLER's clock and the result of calibration.

8.18.3.7 SPI Transfer Mode

SPI supports multiple transfer modes such as SPI Bit Mode, SPI Standard Mode, SPI Dual Mode, SPI Quad Mode, and SPI Octal Mode. Their main differences are the number of wires. Even in the same transfer mode, the detail modes will be derived based on the number of wires used to data transfer. The number of data lines used by Command-Address-Data is indicated on the subdivision pattern heading, expressed as (x-x-x). For example, Command uses one Data line, Address and data both use two data lines, identified by (1-2-2).

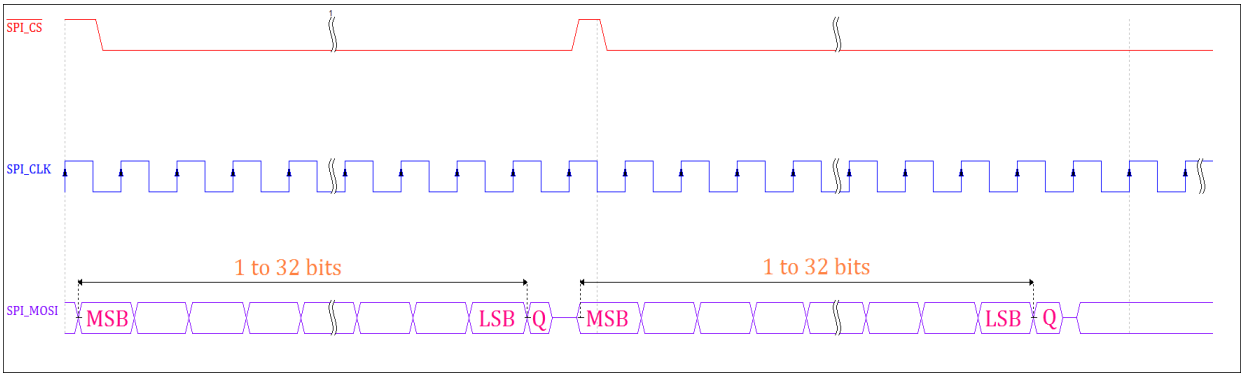
SPI Bit Mode (3-Wire/4-Wire)

For some specific scenarios, some devices such as the sensor use the SPI interfaces as the communication protocol. Generally, their data size is relatively small, and some devices support three wires, so the bit mode is added and 3-wire mode and 4-wire mode are subdivided, which includes SPI_CS, SPI_SCK, and 1/2 wires. The transmission length ranges from 1-32bit.

The 4-Wire Mode is selected when the Work Mode Select(bit[1:0]) is equal to 0x3 in the SPI Bit-Aligned Transfer Configure Register. In SPI 4-Wire Mode, the input data and output data use the independent two data line. The MISO is used for input data, and the MOSI is used for output data.

The SPI 3-Wire Mode is only valid when the SPI controller work as Master Device, and selected when the Work Mode Select(bit[1:0]) is equal to 0x2 in the SPI Bit-Aligned Transfer Configure Register. and in the 3-Wire mode, the input data and the output data use the same single data line. The following figure describe this mode.

Figure 8-124 SPI 3-Wire Mode



SPI Standard Mode (4-Wire)

Signal Wire: CS#, SCK, IO0, and IO1.

Figure 8-125 SPI CMD with Single IO

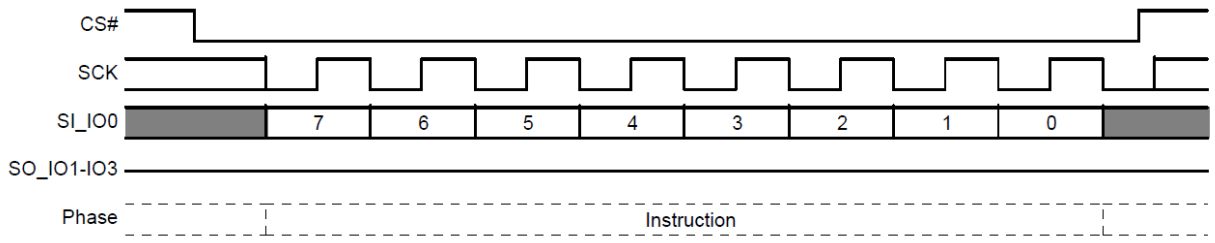


Figure 8-126 SPI Write CMD and DATA with One Wire

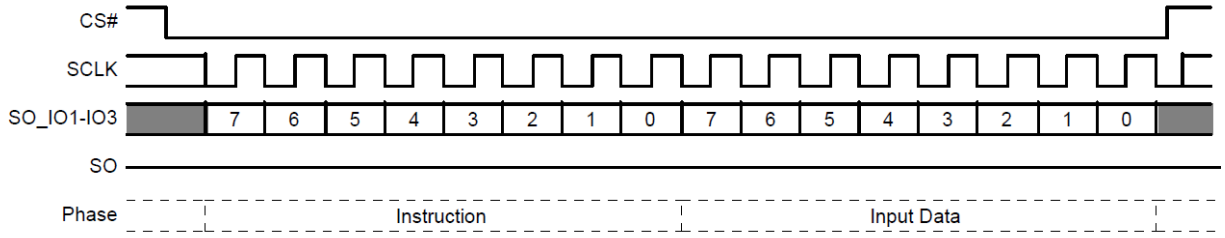


Figure 8-127 SPI Write CMD and Read DATA with One Wire (No Dummy)

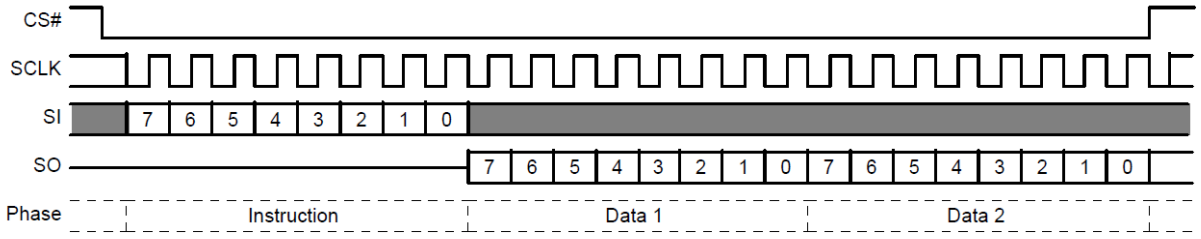
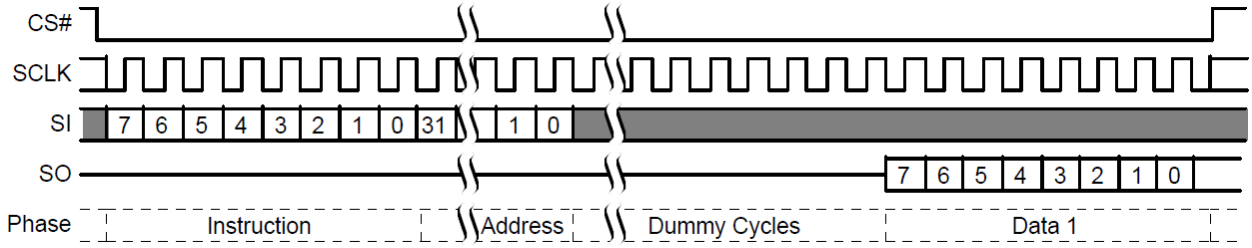


Figure 8-128 SPI Write CMD and Read DATA with One Wire (with Dummy)



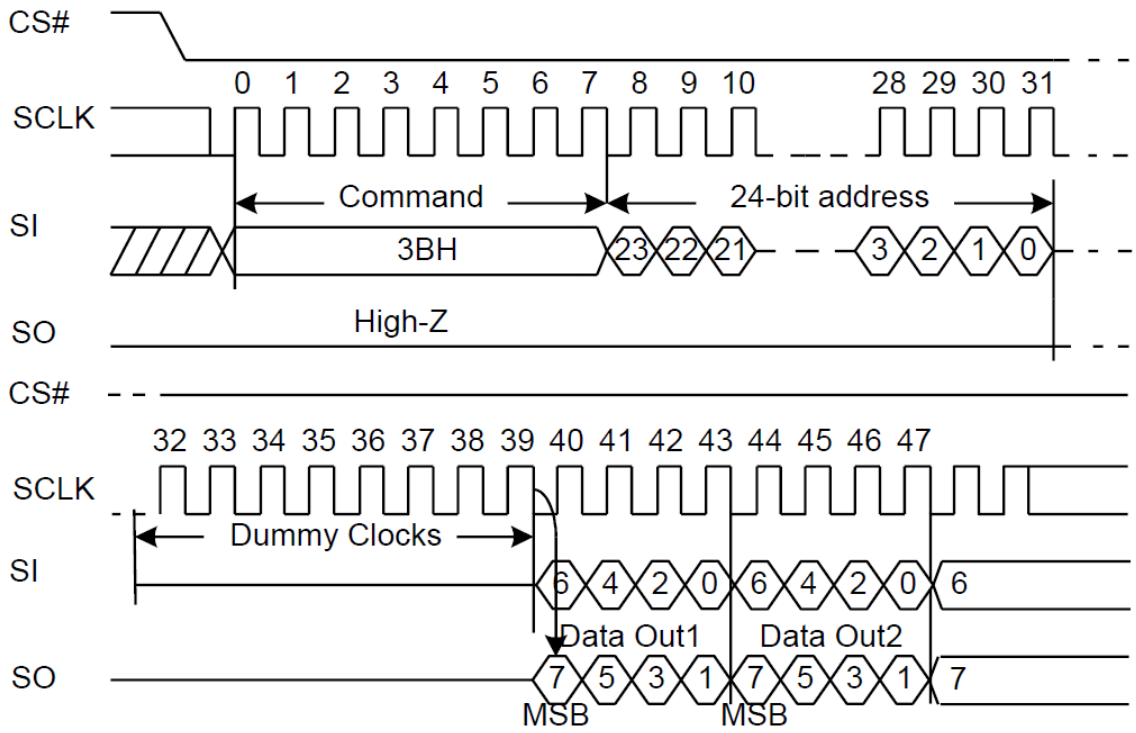
SPI Dual Mode

Using the dual mode allows data to be transferred to or from the device at two times the rate of standard single mode SPI devices. Data can be read at a faster speed using two data bits (MOSI and MISO) at a time. The following describe the Dual Input/Dual Output SPI (1-1-2), the Dual IO SPI (1-2-2), and the (2-2-2) SPI Mode.

- SPI Dual Input/dual Output Mode (1-1-2)

In the dual Input/dual Output SPI, the command, address, and the dummy bytes are output in the unit of a single bit in serial mode through SPI_MOSI line. Only the data bytes are output (write) and input (read) in unit of dual bits through the SPI_MOSI and SPI_MISO.

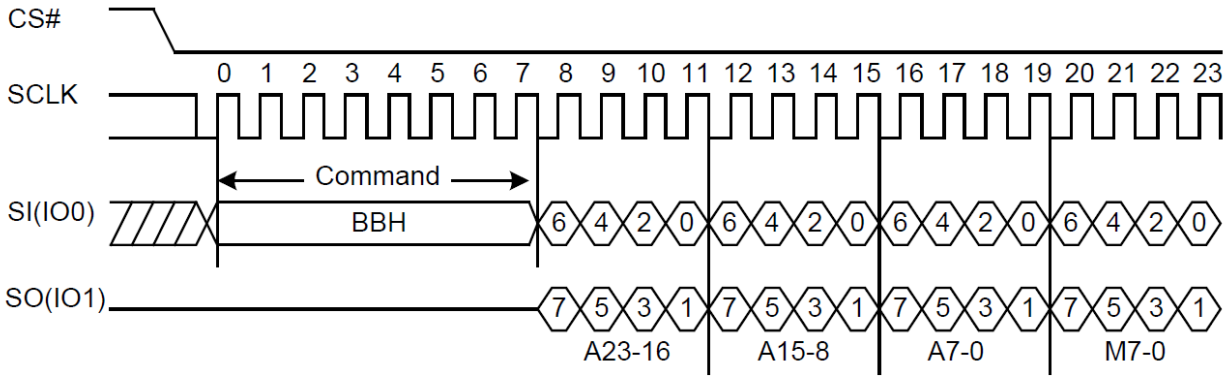
Figure 8-129 SPI Dual Input/Dual Output Mode（1-1-2）



- SPI Dual IO Mode (1-2-2)

In the Dual IO SPI, only the command bytes are output in the unit of a single bit in serial mode through SPI_MOSI line. The address bytes and the dummy bytes are output in the unit of dual bits through the SPI_MOSI and SPI_MISO. And the data bytes are output (write) and input (read) in the unit of dual bits through the SPI_MOSI and SPI_MISO.

Figure 8-130 SPI Dual Input/Dual Output Mode (1-2-2)



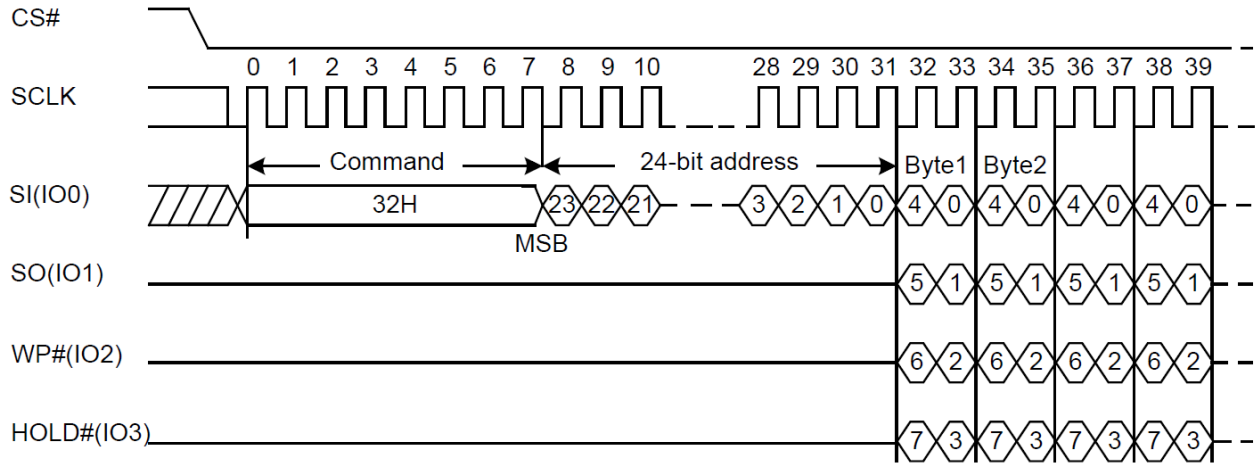
SPI Quad Mode

Using the quad mode allows data to be transferred to or from the device at 4 times the rate of standard single mode SPI devices, data can be read at fast speed using four data bits (MOSI, MISO, IO2(WP#) and IO3(HOLD#)) at the same time. The following describe the Quad Input/Quad Output SPI (1-1-4), 1-4-4 mode, and 4-4-4 mode.

- Quad Input/Quad Output SPI (1-1-4)

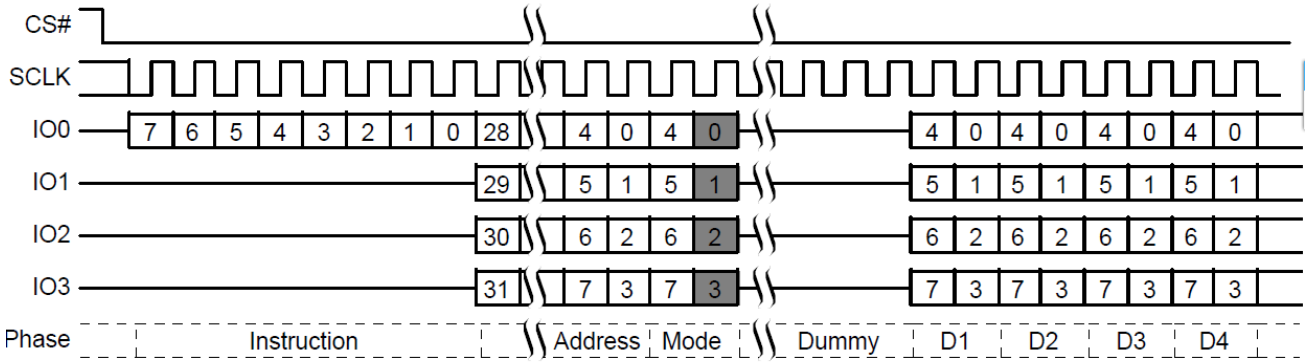
In the Quad Input/Quad Output SPI, the command, address, and the dummy bytes are output in unit of a single bit in serial mode through SPI_MOSI line. Only the data bytes are output (write) and input(read) in unit of quad bits through the SPI_MOSI, SPI_MISO, SPI_WP# and SPI_HOLD#.

Figure 8-131 SPI Quad Input/Dual Output Mode (1-1-4)



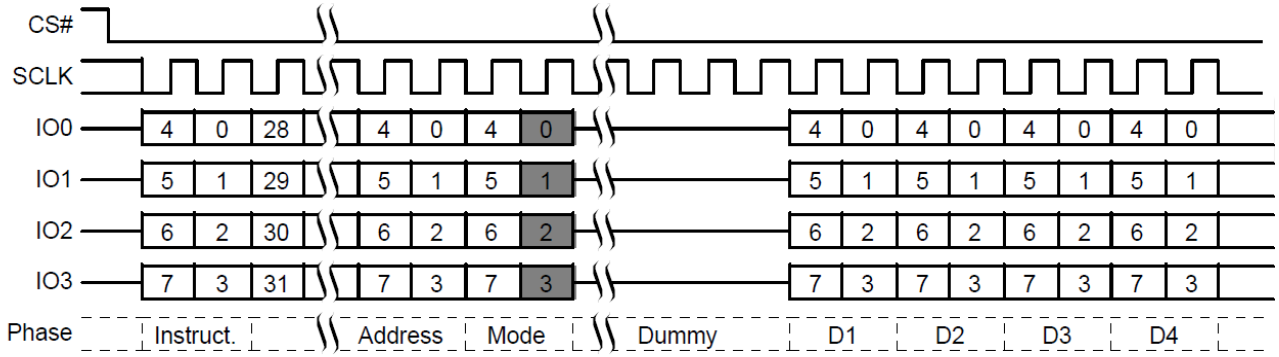
- 1-4-4 Mode

Figure 8-132 SPI Quad Input/Dual Output Mode (1-4-4)



- 4-4-4 Mode

Figure 8-133 SPI Quad Input/Dual Output Mode (4-4-4)

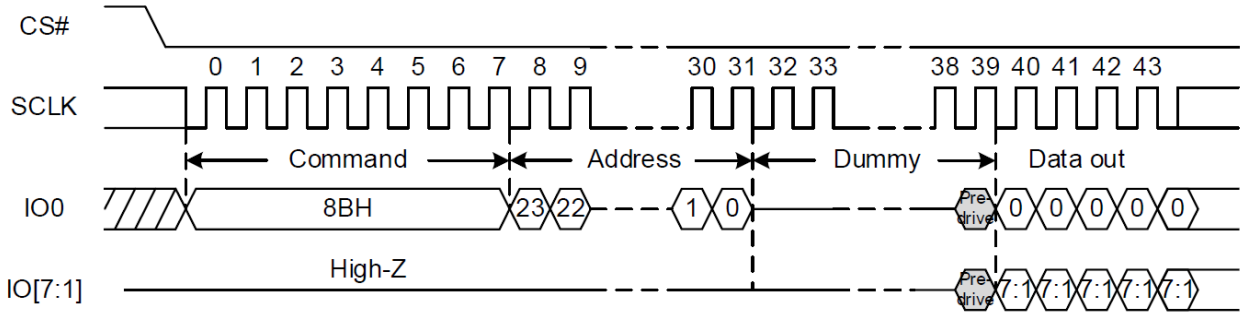


SPI Octal Mode

The Octal SPI Mode allow data to be transferred to or from the device at eight times the rate of the standard SPI. When using the Octal SPI, there are 8 data wires.

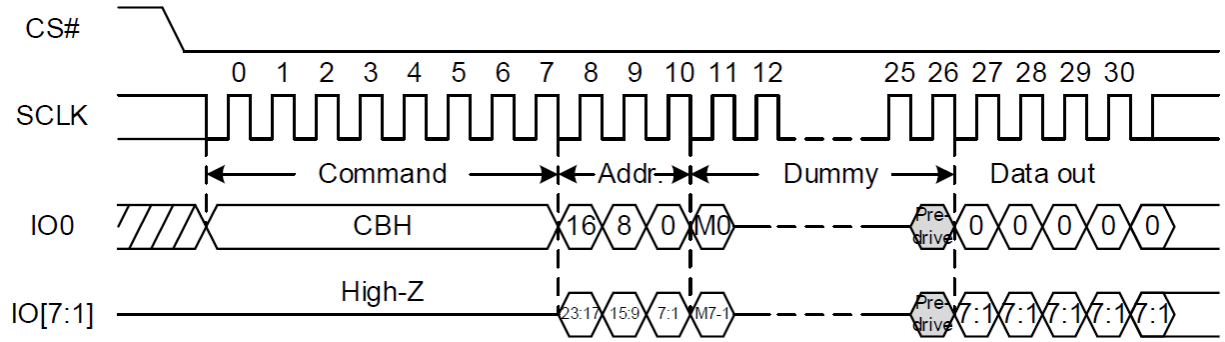
- SPI Octal 1-1-8

Figure 8-134 SPI Octal Input/Dual Output Mode (1-1-8)



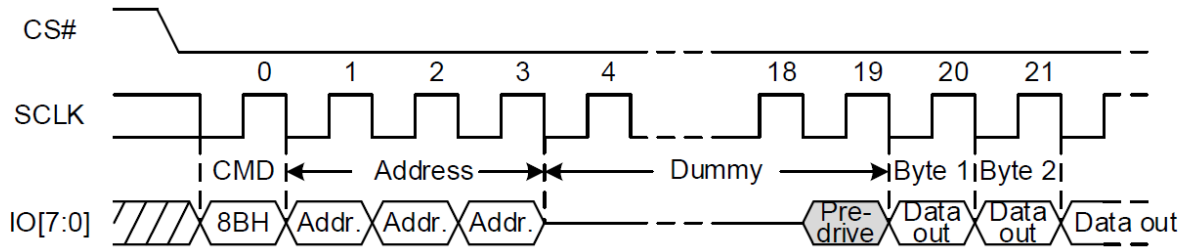
- SPI Octal 1-8-8

Figure 8-135 SPI Octal Input/Dual Output Mode (1-8-8)



- SPI Octal 8-8-8

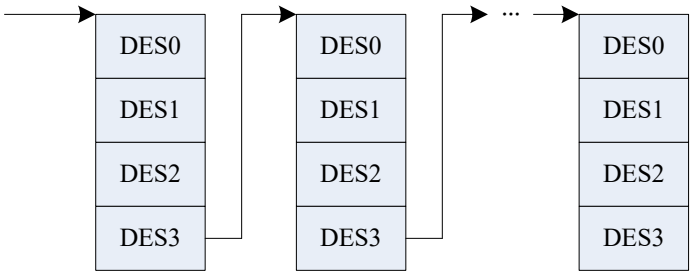
Figure 8-136 SPI Octal Input/dual Output Mode (8-8-8)



8.18.3.8 SPIFC Private DMA

The private DMA adopts the chained descriptor structure. The address and size of the first descriptor are configured by registers. After getting the first descriptor, the hardware will obtain the data from the address recorded in the descriptors. When the current transfer finishes, the hardware will continue to obtain descriptors based on the address of the next descriptor given by the previous descriptor until the terminal character is found. Then, a DMA transfer ends.

Figure 8-137 Descriptor Structure Diagram



The private DMA also supports SPI parameter configuration. The SPI transfer can be configured by the parameters of the descriptors 4-7.

Descriptor0 Definition

Bits	Descriptor
31:7	Reserved
6:4	HBURST_TYPE indicate hburst len 000: SINGLE, hburst_len = 1 011: INCR4, hburst_len = 4 101: INCR8, hburst_len = 8 111: INCR16, hburst_len = 16
3:2	/

Bits	Descriptor
1	DMA_DIR DMA Write Process or Read Process 0: Read 1:Write
0	DMA_FINISH_FLAG DMA Finish Flag

Descriptor1 Definition

Bits	Descriptor
31:24	DMA_BLK_LEN DMA Block Len Mode 0: 8Byte 1: 16Byte 2: 32Byte 3: 64Byte Recommended Configuration： The data volume of DMA_BLK_LEN is greater than or equal to that of HBURST_TYPE.
23:17	/
16:0	DMA_DATA_LEN Indicate the data byte number of current DMA operation.

Descriptor2 Definition

Bits	Descriptor
31:0	DMA_Buffer_ SADDR The real address is as below The word address is needed, namely, the byte address abandons the low 2 bits.

Descriptor3 Definition

Bits	Descriptor
31:0	NEXT_DESCRIPTOR_ADDR These bits indicate the pointer to the physical memory where the next descriptor is present, which are word (4byte) address (The lower two bits are deleted from the byte address).

Descriptor4 Definition

Bit	Description
31:30	/

Bit	Description
29	CMD_DTR CMD DTR Control 1: CMD DTR 0: CMD not DTR CAUTION: When CMD DTR, DTR must be on and CMD2 is required.
28	COMMAND_TRANS_EN Set to '1' if command data need trans to device SPI Controller FSM Phase Enable 1: Enable 0:Disable
27:25	/
24	ADDRESS_TRANS_EN Set to '1' if address need trans to device SPI Controller FSM Phase Enable 1: Enable 0:Disable
23:21	/
20	MODE_BIT_TRANS_EN Set to '1' if Mode bit need trans after address SPI Controller FSM Phase Enable 1: Enable 0:Disable
19:17	/
16	DUMMY_BIT_TRANS_EN Dummy Bit State Enable SPI Controller FSM Phase Enable 1: Enable 0:Disable
15:13	/
12	TX_DATA_EN Set to '1' if Data Need Trans SPI Controller FSM Phase Enable 1: Enable 0:Disable
11:9	/
8	RX_DATA_EN Set to '1' if Data Need Receive SPI Controller FSM Phase Enable 1: Enable 0:Disable
7:0	/

Descriptor5 Definition

Bit	Description
31:0	ADDR_OPCODE Address Content Trans Through SPI

Descriptor6 Definition

This register should be setup while the controller is idle.

Bit	Description
31:24	CMD_OPCODE Command Content Trans Through SPI
23:16	MODE_OPCODE Mode Content Trans Through SPI
15:8	CMD_OPCODE2 Command2 Content Trans Through SPI
7:6	CMD_TRANS_TYPE Command Transfer Type 00: Command can be Shifted to the device on DQ0 01: Command can be Shifted to the device on DQ0 and DQ1 10: Command can be Shifted to the device on DQ0- DQ3 11 : Command can be Shifted to the device on DQ0-DQ7
5:4	ADDR_TRANS_TYPE Address Transfer Type 00: Address can be Shifted to the device on DQ0 01: Address can be Shifted to the device on DQ0 and DQ1 10: Address can be Shifted to the device on DQ0- DQ3 11: Address can be Shifted to the device on DQ0-DQ7
3:2	MODE_BIT_TRANS_TYPE Mode Bit Transfer Type 00: Mode Bit can be Shifted to the device on DQ0 01: Mode Bit can be Shifted to the device on DQ0 and DQ1 10: Mode Bit can be Shifted to the device on DQ0- DQ3 11: Mode Bit can be Shifted to the device on DQ0-DQ7
1:0	DATA_TRANS_TYPE Data Transfer Type 00: Opcode can be Shifted to the device on DQ0 only 01: Opcode can be Shifted to the device on DQ0 and DQ1 only 10: Opcode can be Shifted to the device on DQ0- DQ3 11: Opcode can be Shifted to the device on DQ0-DQ7

Descriptor7 Definition

Bit	Description
31	DATA_TRANS_NUM[16]
30:29	
28	SPI_NORMAL_EN if dma config spi, this bit start SPI FSM.
27:25	/
24	ADDR_SIZE_MODE Address Size Mode 0: Address Size 24bit. 1: Address Size 32bit.
23:16	DUMMY_TRANS_NUM Number of Dummy Cycles A value of 0 = 1 Cycle A value of 1 = 1 Cycle ... A value of N = N Cycle
15:0	DATA_TRANS_NUM Num of Data Trans Through SPI(Byte) 0: Non-Write. 1: Write 1 Byte. 2: Write 2 Bytes. 3: Write 3 Bytes. ..... 65535: Write 65535 Bytes. **Note: These Bits Indicate number of data bytes in a CHIP SELECT period.** **Notice the difference between DATA_TRANS_NUM here and DMA_DATA_LEN in Descriptor1.**

8.18.4 Programming Guidelines

8.18.4.1 DMA Transfer

The software operation of the SPI DMA transfer is divided into 5 steps. 5 steps are described in detail in the following sections.

Step 1 System Setup

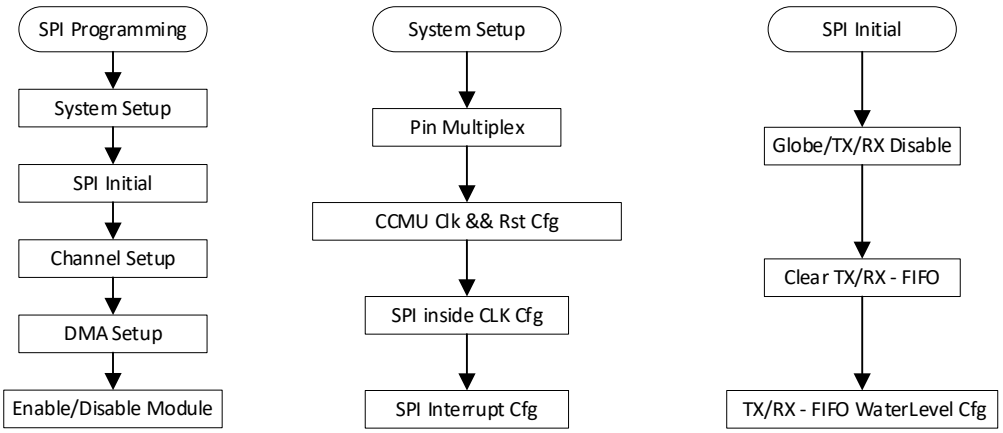
Step 2 SPI Initialization

Step 3 Channel Setup

Step 4 DMA Setup

Step 5 Enable SPI

Figure 8-138 SPI Programming flow



8.18.4.2 System Setup

Step 1 Configure SPI Pin

Programming the GPIO.

Step 2 Configure SPI Clock and Reset in CCU

Configure SPI ref clock, AHB Clock, De-assert SPI ref Reset, and AHB reset in Clock Controller Unit (CCU).

Step 3 Configure SPI Internal Working Clock

a) Configure clock source (SPI_TIMING_CFG[0x000C])

- STR Mode

  Write 0 to the CLK_SPI_SRC_SEL bit (bit [24]), the CLK_SCK_SRC_SEL bit (bit [25]) bit, and the CLK_SCKOUT_SRC_SEL bit (bit [26]) bit.

- DTR Mode

  Write 0 to the CLK_SPI_SRC_SEL bit (bit [24]) and the CLK_SCK_SRC_SEL bit (bit [25]) bit.

  Write 1 to the CLK_SCKOUT_SRC_SEL bit (bit [26]) bit.

NOTE

When the value of the CLK_SCKOUT_SRC_SEL bit (bit [26]) bit is 1, the real output clock frequency of SPIF-CLK signal is the SPIFC clock frequency divided by 2.

b) Generate the sckr of SPI data receiving

- Configure SCKR delay mode (SPI_TIMING_CFG[0x000C]). Refer to the section SPI Run Clock and Sample Mode for more details.

SCKR_DLY_MODE_SEL (bit [20]): Select delay mode. Writing 0 is the digital delay, and writing 1 is the digital and analog delay.

- Configure the digital delay of SCKR

  SPI_TIMING_CFG[0x000C], DIGITAL_SCKR_DELAY_CFG (bit[18:16]): digital delay volume. (step length: 0.5 clock)

- Configure the analog delay of SCKR

  SPI_TIMING_CFG[0x000C], ANALOG_SAMP_DL_SW_VALUE (bit[5:0])

c) Generate SPI output clock

- Select SPI working mode (SPI_MODE). Refer to the section SPI Flash Controller Clock Mode for more details.
- Configure SPI_GLOBAL_CTRL[0x0004]:

  SPI_CPOL (bit [5]): Clock Polarity

  SPI_CPHA (bit [4]): Clock Phase

- DTR switch (SPI_GLOBAL_CTRL[0x0004])

  DTR_EN (bit [16]): It is closed by default.

Step 4 Configure SPI Interrupt

Configure the SPI_INT_EN (0x0014[31:0]) as 0.

8.18.4.3 SPI Initialization

After the system setup, the registers of SPI can be setup. At first, the SPI needs to be initialized.

Step 1 Disable the SPI_GLOBAL_CTRL.

SPI_NMODE_EN (bit [2]): Write 0.

Step 2 Reset TX/RX FIFO (SPI_GLOBAL_CTRL_ADD[0x0008])

Reset the CDC-BUF/FIFO of TX channel: Write 1 to CDC_WF_SRST to reset the WR_BUFF and WR_FIFO in SPI_WR_BUF_CTRL and the SWF in SPI_TX_INTERFACE.

Reset the CDC-BUF/FIFO of RX channel: Write 1 to CDC_RF_SRST.

Step 3 Water Lever

Configure the water level of FIFO.

SPI_CDC_FIFO_TRIG_LEVEL[0x004C]

8.18.4.4 Channel Setup

Select SPI Channel Parameter Resource

- SPI channel parameter resources:

- CPU is configured by AHB.
- The private DMA fetches descriptors and parses the descriptors 4-7.

- Configure SPI channel parameter sources: SPI_GLOBAL_CTRL[0x0004] and SPI_CFG_MODE(bit[0])
  - 0: Source from CPU
  - 1: Source from DMA descriptor

NOTE

If the parameters source from CPU, configure in reference to all the guidelines in the section Channel Setup. If the parameters source from DMA, configure in reference to the first two guidelines in the section 8.18.4.4 Channel Setup.

SPI Interface Configuration—Public Configuration

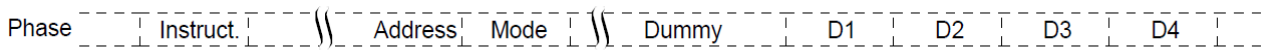
Step 1 CS Delay Configuration：SPI_CS_DELAY[0x001C]

- CSDA (bit [23:16]): The intervals of two adjacent CS enable. The minimum interval is 1sclk. The actual value is CSDA+1.
- CSEOT (bit [15:8]): After the SCLK_OUT is invalid, the CS signal will be de-asserted after CSEOT SCLK. The minimum interval is 1sclk. The actual value is CSEOT+1.
- CSSOT (bit [7:0]): After the SCLK_OUT is valid, the CS signal will be asserted after CSSOT SCLK. The minimum interval is 1sclk. The actual value is CSSOT+1.

Step 2 Activate SPI Trans Phase (SPI_TRANS_PHA_CFG[0x0020])

In a transmission of the SPI interface, the transferring content of I/O wire is as follows. Enable the corresponding Trans Phase based on the content to be transferred.

Figure 8-139 SPI Transfer Phase Flow Diagram



- COMMAND_TRANS_EN (bit [28]): Command (Instruct) Transfer Enable
- ADDRESS_TRANS_EN (bit [24]): Address Transfer Enable
- MODE_BIT_TRANS_EN (bit [20]): Mode Transfer Enable
- DUMMY_BIT_TRANS_EN (bit [16]): Dummy Transfer Enable
- TX_DATA_EN (bit [12]): Enable TX data transfer.
- RX_DATA_EN (bit [8]): Enable RX data transfer.

NOTE

RX_DATA_EN and TX_DATA_EN cannot be activated at the same time.

Step 3 Configure the number of I/O used by TransPhase. (SPI_TRANS_CFG2[0x0028])

- CMD_TRANS_TYPE (bit [13:12])
- ADDR_TRANS_TYPE (bit [9:8])
- MODE_BIT_TRANS_TYPE (bit [5:4])
- DATA_BIT_TRANS_TYPE (bit [1:0])

Step 4 Configure the transferring number of TransPhase (SPI_TRANS_NUM[0x002C])

SPI Interface Configuration—Normal Mode

Configure the contents of all TransPhase

- ADDR: SPI_TRANS_CFG1[0x0024], ADDR_OPCODE (bit [31:0])
- CMD: SPI_TRANS_CFG2[0x0028], CMD_OPCODE (bit[31:24])
- MODE: SPI_TRANS_CFG2[0x0028], CMD_OPCODE (bit[23:16])

SPI Interface Configuration—BIT Mode

Refer to the functions of SPI Bit Mode (3-Wire/4-Wire).

8.18.4.5 DMA Setup

DMA Configuration Registers

Step 1 Configure the descriptor size and starting address of the first descriptor.

- SPI_DMA_CTRL[0x0040]: DMA_DESCRIPTOR_LEN（bit[11:4]）
- SPI_DESCRIPTOR_SADDR[0x0044]

Step 2 Initiate DMA

SPI_DMA_CTRL[0x0040]，CFG_DMA_START（bit[0]）

DMA Configuration

Step 1 In the DMA register, configure the size and the starting address of storage location for the first descriptor.

Step 2 Configure the first DMA descriptor and store the corresponding address in the step 1.

- Descriptor0
  - AHB Master Burst Configuration

HBURST_TYPE (bit [6:4]): Support SINGLE/INCR4/INCR8/INCR16 mode.

- DMA Handling Direction Configuration

  DMA_DIR (bit [1]): Read by DMA, or Write by DMA.

- DMA Finish Flag

  DMA_FINISH_FLAG (bit [0]): Currently the memory space allocated by the CPU to DMA may be a part of the total data volume to be transferred. For instance, the 256 Byte memory space is allocated to transfer 1 MB data. In this situation, a DMA descriptor points to the allocated 256 Byte data location and the next descriptor to fetch data. When the data of the last descriptor is fetched by DMA, the stop bit should be pulled up to terminate a DMA handling.

➢ Descriptor1

- DMA_BLK_LEN ([31:16])

  DMA BLK length. It indicates the size of each DMA packet, with multiples of 8 Byte aligned to achieve the optimized rate when accessing storage.

- DMA_DATA_LEN (bit [15:0])

  It indicates the data volume indicated by the current descriptor of DMA

➢ Descriptor2

It indicates the data storage location of current descriptor.

➢ Descriptor3

It indicates the storage location of the next descriptor.

➢ Descriptor4-7

The SPI configuration parameters. When setting parameters with DMA for SPI, the content of descriptors is configured to SPI.

8.18.4.6 Enable SPI

Normal Mode

SPI_GLOBAL_CTRL[0x0004]，SPI_NMODE_EN (bit[2])

Write 1 to the SPI_NMODE_EN, then it will be auto pulled down.

8.18.4.7 CPU Transfer

The software operation of CPU transfer is almost the same as DMA transfer. There are still two main differences. One difference is that when CPU transfer, channel parameter cannot come from DMA descriptors. SPI_GLOBAL_CTRL[0x0004], SPI_CFG_MODE(bit[0]) must be set 0. All parameters should come from REGISTER FILE.

SPI_GLOBAL_CTRL[0x0004], SPI_CPU_MODE_EN(bit[20]) is used to start CPU transfer (Different from descriptor and SPI_GLOBAL_CTRL[0x0004], SPI_NMODE_EN(bit[2])).

Another difference is that when reading data from Register [0x0210], it is recommended to read SPI_CDC_FIFO_STA[0x0050], RF_CNT([bit5-0]), especially when total read number is not integer multiple of trigger level. When writing data to Register [0x0220], it is always ready as long as trigger level is not reached.

NOTE

Register [0x0040] and [0x0044] are not used in CPU transfer.

8.18.4.8 Status Reading

The software operation of STATUS READING is almost the same as CPU transfer. CMD_TRANS_EN and RX_DATA_EN must be set high because RX data path is used for status reading. MODE_BIT_TRANS_EN and DUMMY_BIT_TRANS_EN should be set accordingly. But STATUS OPCODE will not go through READ_FIFO and READ_BUFFER. Meanwhile, I/O Pins used should be set accordingly. SPI_STATUS_READ[0x0068] and SPI_STATUS_READ_2[0x006C] are used for STATUS READING. And to avoid endless reading and dead lock, CPU should tell the maximum reading times. Then, to start STATUS READING, SPI_READ_STATUS_MODE_EN should be set. In this mode, DATA_TRANS_NUM (002C, [15:0]) is suggested to be 1 and DTR_EN [0x0004, bit 16] is suggested to be 0.

8.18.5 Register List

Module Name	Base Address
SPIFC	0x047F 0000

Register Name	Offset	Description
SPI_GLOBAL_CTRL	0x0004	SPI Global Control Register
SPI_GLOBAL_CTRL_ADD	0x0008	SPI Global Control Additional Register
SPI_TIMING_CFG	0x000C	SPI Timing Configure Register
SPI_TIMING_DLY_STA	0x0010	SPI Timing Delay State Register
SPI_INT_EN	0x0014	SPI Interrupt Enable Register
SPI_INT_STA	0x0018	SPI Interrupt Status Register
SPI_CS_DELAY	0x001C	SPI Chipselect Delay Register
SPI_TRANS_PHA_CFG	0x0020	SPI Trans Phase Configure Register
SPI_TRANS_CFG1	0x0024	SPI Trans Configure1 Register
SPI_TRANS_CFG2	0x0028	SPI Trans Configure2 Register
SPI_TRANS_NUM	0x002C	SPI Trans Number Register
SPI_DMA_CTRL	0x0040	SPI DMA Control Register
SPI_DESCRIPTOR_SADDR	0x0044	SPI DMA Descriptor Start Address Register
SPI_CDC_FIFO_TRIG_LEVEL	0x004C	SPI CDC FIFO Trigger Level Register
SPI_CDC_FIFO_STA	0x0050	SPI CDC FIFO Status Register

8.19 UART

8.19.1 Overview

The universal asynchronous receiver transmitter (UART) provides an asynchronous serial communication with external devices, modem (data carrier equipment, DCE). It performs serial-to-parallel conversion on the data received from peripherals and transmits the converted data to the internal bus. It also performs parallel-to-serial conversion on the data that is transmitted to peripherals.

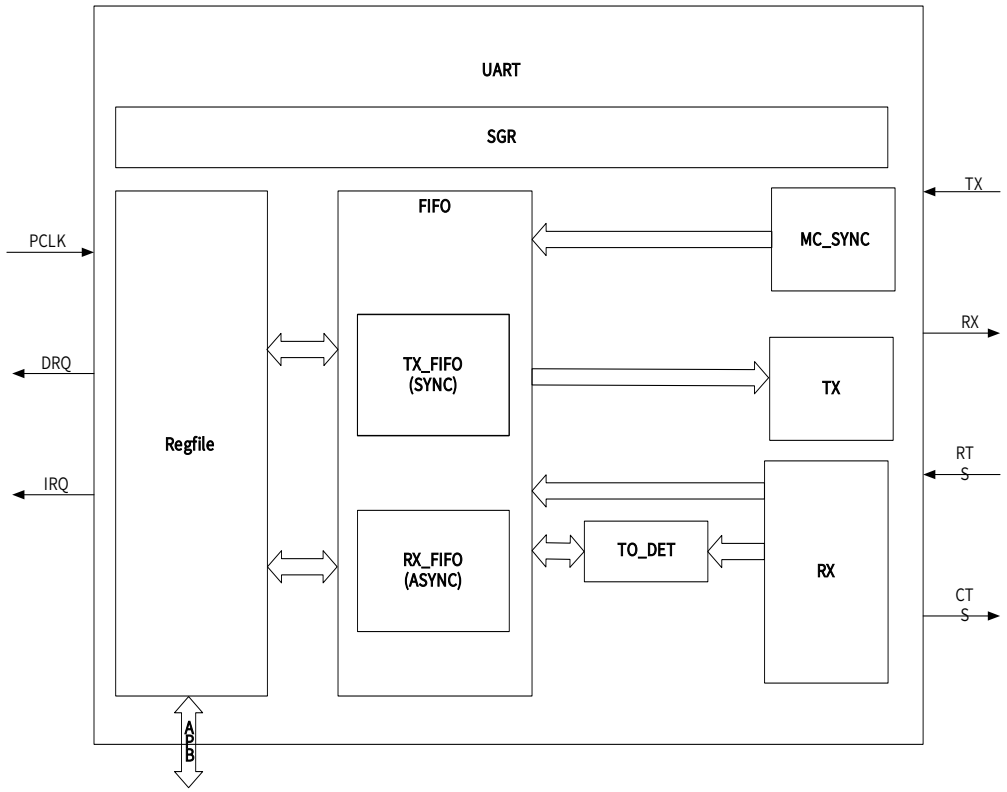
The UART has the following features:

- Up to 10 UART controllers
  - 8 UART controllers in CPUX domain: UART0, UART1, UART2, UART3, UART4, UART5, UART6, and UART7
  - 2 UART controllers in CPUS domain: S_UART0 and S_UART1
- Compatible with industry-standard 16450/16550 UARTs
- Two separate FIFOs: one is RX FIFO, and the other is TX FIFO
  - Each of them is 64 bytes for UART0, S_UART0, and S_UART1
  - Each of them is 128 bytes for UART1, UART2, UART3, UART4, UART5, UART6, and UART7
- The working reference clock is from the APB bus clock
  - Speed up to 10 Mbit/s with 160 MHz APB clock (excluding S_UART0 and S_UART1)
  - Speed up to 5 Mbit/s with 80 MHz APB clock (excluding S_UART0 and S_UART1)
  - Speed up to 3.75 Mbit/s with 60 MHz APB clock (excluding S_UART0 and S_UART1)
  - Speed up to 1.5 Mbit/s with 24 MHz APB clock
- 5 to 8 data bits for RS-232 format, or 9 bits RS-485 format
- 1, 1.5 or 2 stop bit(s)
- Programmable parity (even, odd, or no parity)
- Supports TX/RX DMA slave controller interface
- Supports software/hardware flow control
- Supports IrDA-compatible slow infrared (SIR) format
- Supports auto-flow by using CTS & RTS (excluding UART0, S_UART0, and S_UART1)

8.19.2 Block Diagram

The following figure shows a block diagram of the UART.

Figure 8-140 UART Block Diagram



8.19.3 Functional Description

8.19.3.1 External Signals

The following table describes the external signals of UART.

Table 8-62 UART External Signals

Signal Name	Description	Type
UART0-TX	UART0 Data Transmitter	O
UART0-RX	UART0 Data Receiver	I
UART1-TX	UART1 Data Transmitter	O
UART1-RX	UART1 Data Receiver	I
UART1-CTS	UART1 Data Clear to Send	I
UART1-RTS	UART1 Data Request to Send	O
UART2-TX	UART2 Data Transmitter	O
UART2-RX	UART2 Data Receiver	I
UART2-CTS	UART2 Data Clear to Send	I
UART2-RTS	UART2 Data Request to Send	O
UART3-TX	UART3 Data Transmitter	O
UART3-RX	UART3 Data Receiver	I

Signal Name	Description	Type
UART3-CTS	UART3 Data Clear to Send	I
UART3-RTS	UART3 Data Request to Send	O
UART4-TX	UART4 Data Transmitter	O
UART4-RX	UART4 Data Receiver	I
UART4-CTS	UART4 Data Clear to Send	I
UART4-RTS	UART4 Data Request to Send	O
UART5-TX	UART5 Data Transmitter	O
UART5-RX	UART5 Data Receiver	I
UART5-CTS	UART5 Data Clear to Send	I
UART5-RTS	UART5 Data Request to Send	O
UART6-TX	UART6 Data Transmitter	O
UART6-RX	UART6 Data Receiver	I
UART6-CTS	UART6 Data Clear to Send	I
UART6-RTS	UART6 Data Request to Send	O
UART7-TX	UART7 Data Transmitter	O
UART7-RX	UART7 Data Receiver	I
UART7-CTS	UART7 Data Clear to Send	I
UART7-RTS	UART7 Data Request to Send	O
S-UART0-TX	S-UART0 Data Transmitter	O
S-UART0-RX	S-UART0 Data Receiver	I
S-UART1-TX	S-UART1 Data Transmitter	O
S-UART1-RX	S-UART1 Data Receiver	I

8.19.3.2 Clock Sources

The following table describes the clock sources of UART.

Table 8-63 UART Clock Sources

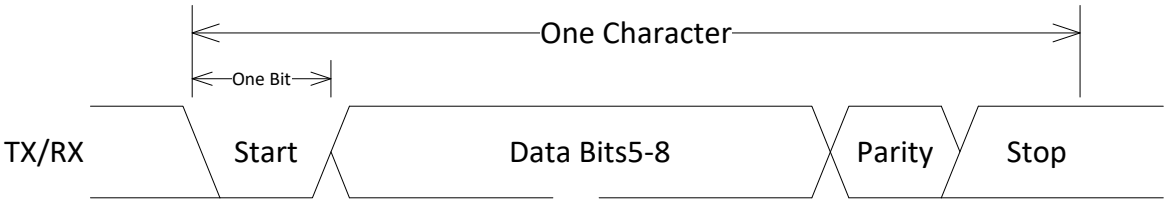
UART Interfaces	Clock Source	Description	Clock Module
UART0 to UART7	APB1 Bus	UART clock source. Refer to CCU for details on APB1.	CCU
S_UART0, S_UART1	APBS1 Bus	S_UART clock source. Refer to PRCM for details on APB1.	PRCM

8.19.3.3 Typical Applications and Timing Diagram

UART Serial Data Format

The following figure shows the UART serial data format. The start bit, data bit, parity bit, and stop bit can be configured.

Figure 8-141 UART Serial Data Format



Using UART for RTS/CTS Autoflow Control

Figure 8-142 shows the typical application diagram for RTS/CTS autoflow control. Figure 8-143 shows the data format of the RTS/CTS autoflow control.

Figure 8-142 Application Diagram for RTS/CTS Autoflow Control

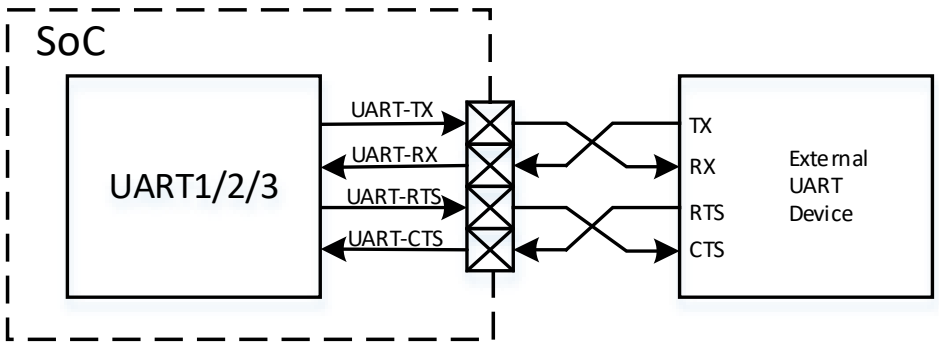
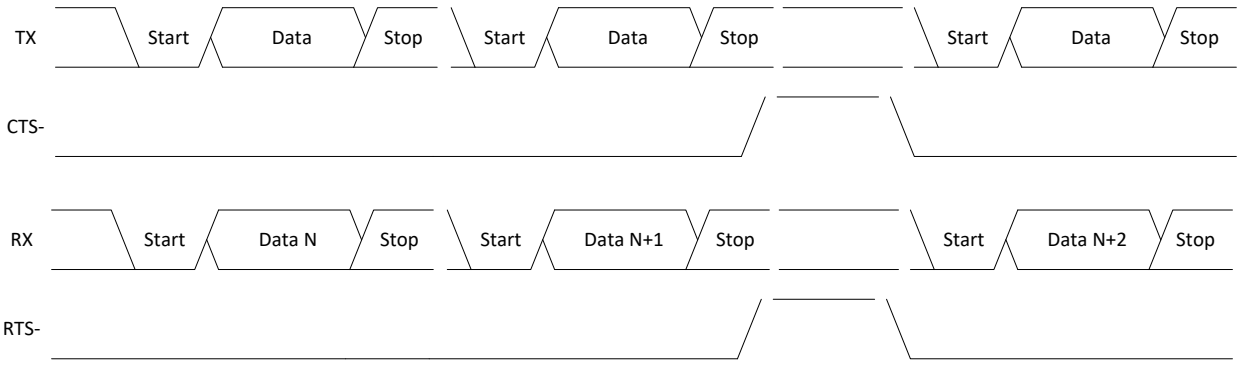


Figure 8-143 RTS/CTS Autoflow Control Data Format



Using UART for Serial IrDA

Figure 8-144 shows the application diagram for the IrDA transceiver. Figure 8-145 shows the data format of the serial IrDA.

Figure 8-144 Application Diagram for IrDA Transceiver

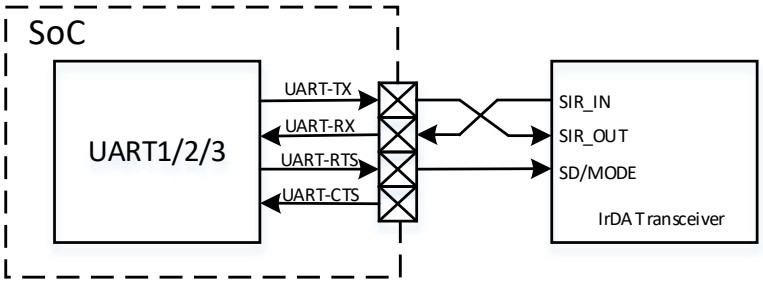
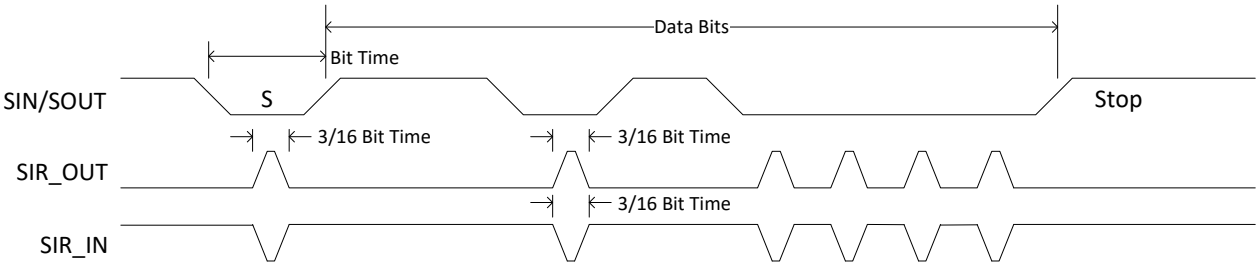


Figure 8-145 Serial IrDA Data Format



Using UART for RS-485

Figure 8-146 shows the application diagram for the RS-485 transceiver. Figure 8-147 shows the data format of the RS-485.

Figure 8-146 Application Diagram for RS-485 Transceiver

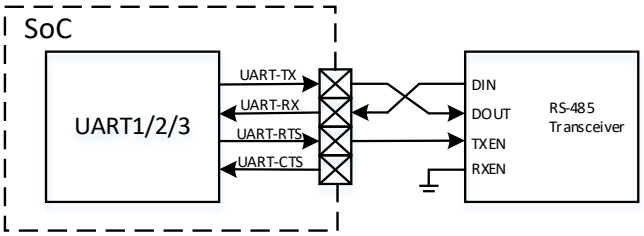
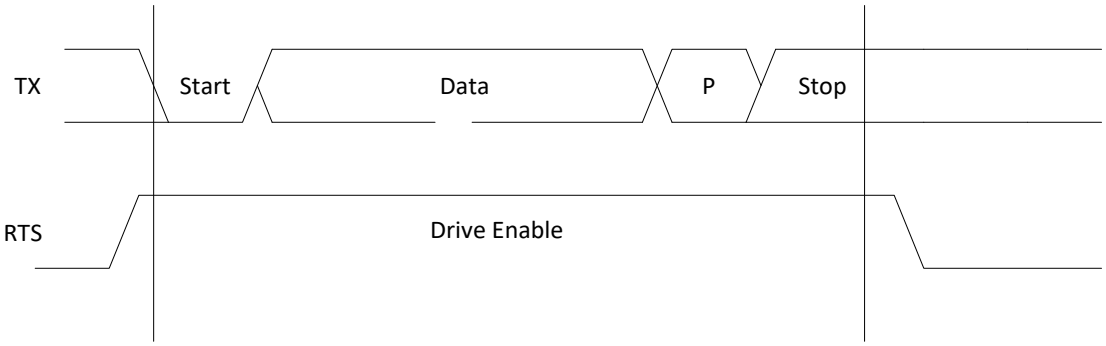


Figure 8-147 RS-485 Data Format



8.19.3.4 UART Operating Mode

Data Frame Format

The UART_LCR register can set the basic parameter of a data frame: data width (5 to 8 bits), stop bit number (1/1.5/2), parity type.

A frame transfer of the UART includes the start signal, data signal, parity bit, and stop signal. The LSB is transmitted first.

- Start signal (start bit): It is the start flag of a data frame. According to the UART protocol, the low level of the TXD signal indicates the start of a data frame. When the UART transmits data, the level needs to hold high.
- Data signal (data bit): The data bit width can be configured as 5-bit, 6-bit, 7-bit, and 8-bit through different applications. If RS-485 mode is enabled, th data bit width is 8-bit.
- Parity bit: It is a 1-bit error correction signal. Parity bit includes odd parity, even parity. The UART can enable and disable the parity bit by setting the UART_LCR register. If RS-485 mode is enabled, the parity bit must be keep enabled.
- Stop Signal (stop bit): It is the stop bit of a data frame. The stop bit can be set to 1-bit, 1.5-bit, and 2-bit by the UART_LCR register. The high level of the TXD signal indicates the end of a data frame.

Baud and Error Rates

The baud rate is calculated as follows: Baud rate = SCLK/(16 * divisor).

The SCLK is usually APB1 and can be set in section 2.6 Clock Controller Unit (CCU).

The divisor is frequency divider of UART. The frequency divider has 16-bit, the low 8-bit is in the UART_DLL register, the high 8-bit is in the UART_DLH register.

The relationship between the different UART mode and the error rate is as follows.

Table 8-64 UART Mode Baud and Error Rates

Clock Source	Divisor	Baud Rate	Over Sampling	Error(%)
24000000	5000	300	16	0
24000000	2500	600	16	0
24000000	1250	1200	16	0
24000000	625	2400	16	0
24000000	313	4800	16	-0.16
24000000	156	9600	16	0.16
24000000	78	19200	16	0.16
24000000	39	38400	16	0.16
24000000	26	57600	16	0.16
24000000	13	115200	16	0.16
48000000	13	230400	16	0.16
60000000	1	3750000	16	0

Clock Source	Divisor	Baud Rate	Over Sampling	Error(%)
75000000	5	921600	16	1.725
48000000	3	1000000	16	0
24000000	1	1500000	16	0
48000000	1	3000000	16	0
80000000	1	5000000	16	0
160000000	1	10000000	16	0

Table 8-65 IrDA Mode Baud and Error Rates

Clock source	Divisor	Baud rate	Encoding	Error(%)
24000000	5000	300	3/16	0
24000000	2500	600	3/16	0
24000000	1250	1200	3/16	0
24000000	625	2400	3/16	0
24000000	313	4800	3/16	-0.16
24000000	156	9600	3/16	0.16
24000000	78	19200	3/16	0.16
24000000	39	38400	3/16	0.16
24000000	26	57600	3/16	0.16
24000000	13	115200	3/16	0.16

Table 8-66 RS485 Mode Baud and Error Rates

Clock source	Divisor	Baud rate	Encoding	Error(%)
24000000	5000	300	16	0
24000000	2500	600	16	0
24000000	1250	1200	16	0
24000000	625	2400	16	0
24000000	313	4800	16	-0.16
24000000	156	9600	16	0.16
24000000	78	19200	16	0.16
24000000	39	38400	16	0.16
24000000	26	57600	16	0.16
24000000	13	115200	16	0.16
48000000	13	230400	16	0.16
60000000	1	3750000	16	0
75000000	5	921600	16	1.725
48000000	3	1000000	16	0
24000000	1	1500000	16	0
48000000	1	3000000	16	0
80000000	1	5000000	16	0
160000000	1	10000000	16	0

DLAB Definition

The DLAB control bit (UART_LCR[7]) is the access control bit of the divisor Latch register.

If DLAB is 0, then the 0x00 offset address is the UART_RBR/UART_THR (RX/TX FIFO) register, and the 0x04 offset address is the UART_IER register.

If DLAB is 1, then the 0x00 offset address is the UART_DLL register, and the 0x04 offset address is the UART_DLH register.

When the UART initials, the divisor needs to be set. That is, writing 1 to DLAB can access the UART_DLL and UART_DLH register, after finished the configuration, writing 0 to DLAB can access the UART_RBR/UART_THR register.

CHCFG_AT_BUSY Definition

The function of the CHCFG_AT_BUSY (UART_HALT [1]) and CHANGE_UPDATE (UART_HALT[2]) are as follows.

CHCFG_AT_BUSY: Enable the bit, the software can also set the UART controller when UART is busy, such as the UART_LCR, UART_DLH, UART_DLL register.

CHANGE_UPDATE: If CHCFG_AT_BUSY is enabled, and CHANGE_UPDATE is written to 1, the configuration of the UART controller can be updated. After completed the update, the bit is cleared to 0 automatically.

Setting divisor performs the following steps:

Write 1 to CHCFG_AT_BUSY to enable “configure at busy”.

Write 1 to DLAB (UART_LCR[7]) and set the UART_DLH and UART_DLL registers.

Write 1 to CHANGE_UPDATE to update the configuration. The bit is cleared to 0 automatically after completing the update.

UART Busy Flag

The UART_USR [0] is a busy flag of the UART controller.

When the TX transmits data, or the RX receives data, or the TX FIFO is not empty, or the RX FIFO is not empty, then the busy flag bit can be set to 1 by hardware, which indicates the UART controller is busy.

8.19.4 Programming Guidelines

The following takes the UART module in the CPUX domain as an example.

8.19.4.1 Initialization

Step 1 System Initialization

- Configure APB1_CLK_REG in the CCU module to set the APB1 bus clock (The clock is 24MHz by default).

- Set UART_BGR_REG[UARTx_GATING] to 1 to enable the module clock, and set UART_BGR_REG[UARTx_RST] to 1 to de-assert the module.

Step 2 UART Controller Initialization

- IO configuration: Configure GPIO multiplex as UART function, and set UART pins to internal pull-up mode (For detail, see the description in section 8.6 GPIO).
- Baud-rate configuration:
  - Set UART baud-rate (refer to section 8.19.3.4);
  - Write UART_FCR[FIFOE] to 1 to enable TX/RX FIFO;
  - Write UART_HALT[HALT_TX] to 1 to disable TX transfer;
  - Set UART_LCR[DLAB] to 1, remain default configuration for other bits; set 0x00 offset address to the UART_DLL register, set 0x04 offset address to the UART_DLH register;
  - Write the high 8-bit of divisor to the UART_DLH register, and write the low 8-bit of divisor to the UART_DLL register;
  - Set UART_LCR[DLAB] to 0, remain default configuration for other bits; set 0x00 offset address to the UART_RBR/UART_THR register, set 0x04 offset address to the UART_IER register;
  - Set UART_HALT[HALT_TX] to 0 to enable TX transfer.

Step 3 Controller Parameter Configuration

- Set data width, stop bits, and even/odd parity type by writing the UART_LCR register.
- Reset, enable FIFO and set FIFO trigger condition by writing the UART_FCR register.
- Set the flow control parameter by writing the UART_MCR register.

Step 4 Interrupt Configuration

- Configure UART interrupt vector number to request UART interrupt (Refer to section 2.8 Generic Interrupt Controller (GIC) for interrupt vector number).
- In DMA mode, write UART_IER to 0 to disable interrupt; write UART_HSK[Handshake configuration] to 0xE5 to set DMA handshake mode; write UART_FCR[DMAM] to 1 to set DMA transmission/reception mode; set DMA parameter and request DMA interrupt according to DMA configuration process.
- In Interrupt mode, configure UART_IER to enable the corresponding interrupt according to requirements: such as transmit (TX) interrupt, receive (RX) interrupt, receive line status interrupt, RS48 interrupt, etc. (Here TX/RX interrupt is usually used).

8.19.4.2 Transferring/Receiving Data in DMA Mode

Step 1 Initialize UART model. Refer to section 8.19.4.1 Initialization for initialization steps.

Step 2 Configure UART_TFL and UART_RFL to set DRQ trigger level for DMA.

Step 3 Configure UART_HALT to set PTE and DMA_PTE_RX.

Step 4 DMA data channel, including the transfer source address, the transfer destination address, the number of data to be transferred, and the transfer type, and so on. For details, see section 2.7 DMA Controller (DMAC).

Step 5 Enable the DMA transfer or receive function of the UART by setting the register of the DMA module.

Step 6 Determine whether UART data is transferred or received completely based on the DMA status. If all data is transferred or received completely, disable the DMA transfer or receive function of the UART.

8.19.4.3 Transferring/Receiving Data in Interrupt Mode

- Data transfer

Step 1 Initialize UART model. Refer to section 8.19.4.1 for initialization steps.

Step 2 Configure UART_TFL and UART_RFL to set DRQ trigger level for DMA.

Step 3 Configure UART_HALT to set PTE and DMA_PTE_RX.

Step 4 Set UART_IER[ETBEI] to 1 to enable the UART transmission interrupt.

Step 5 Write the data to be transmitted to UART_THR.

Step 6 When the data of TX_FIFO meets trigger condition (such as FIFO/2, FIFO/4), the UART transfer interrupt is generated.

Step 7 Check UART_USR[TFE] and determine whether TX_FIFO is empty. If UART_USR[TFE] is 1, it indicates that the data in TX_FIFO is transmitted completely.

Step 8 Clear UART_IER[ETBEI] to 0 to disable transfer interrupt.

- Data receive

Step 1 Initialize UART model. Refer to section 8.19.4.1 for initialization steps.

Step 2 Configure UART_TFL and UART_RFL to set DRQ trigger level for DMA.

Step 3 Configure UART_HALT to set PTE and DMA_PTE_RX.

Step 4 Set UART_IER[ERBFI] to 1 to enable the UART reception interrupt.

Step 5 When the received data from RX_FIFO meets trigger condition (such as FIFO/2, FIFO/4), the UART receive interrupt is generated.

Step 6 Read data from UART_RBR.

Step 7 Check RX_FIFO status by reading UART_USR[RFNE] and determine whether to read data. If the bit is 1, continue to read data from UART_RBR until UART_USR[RFNE] is cleared to 0, which indicates data is received completely.

8.19.4.4 Transferring/Receiving Data in RS485 Mode

Step 1 Initialize UART model. Refer to section 8.19.4.1 for initialization steps.

Step 2 Configure UART_485_CTL [1:0] to select UART RS485 receive data format.

Step 3 If AAD receive data mode is choosed, configure UART_RS485_ADDR_MATCH register to set receive address in AAD mode.

Step 4 If DMA mode is selected, perform Step2 to Step6 in section 8.19.4.2. Otherwise, perform Step2 to Step7 in section 8.19.4.3.

8.19.5 Register List

Module Name	Base Address
UART0	0x02500000
UART1	0x02500400
UART2	0x02500800
UART3	0x02500C00
UART4	0x02501000
UART5	0x02501400
UART6	0x02501800
UART7	0x02501C00
S_UART0	0x07080000
S_UART1	0x07080400

Register Name	Offset	Description
UART_RBR	0x0000	UART Receive Buffer Register
UART_THR	0x0000	UART Transmit Holding Register
UART_DLL	0x0000	UART Divisor Latch Low Register
UART_DLH	0x0004	UART Divisor Latch High Register
UART_IER	0x0004	UART Interrupt Enable Register
UART_IIR	0x0008	UART Interrupt Identity Register
UART_FCR	0x0008	UART FIFO Control Register
UART_LCR	0x000C	UART Line Control Register
UART_MCR	0x0010	UART Modem Control Register
UART_LSR	0x0014	UART Line Status Register
UART_MSR	0x0018	UART Modem Status Register
UART_SCH	0x001C	UART Scratch Register