Contents

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

Figures

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

Tables

8 Interface Control

8.1 USB3OTG

8.1.1 Function Description

Features

The USB 3.0 On-The-Go (OTG) of the Hi3660 contains the USB 3.0 controller and USB 3.0 femtoPHY, and supports the host and device functions. As a device, the USB 3.0 OTG connects to the PC or USB host. As a host, the USB 3.0 OTG connects to the USB flash drive or USB device.

The USB module has the following features:

- Complies with the USB 3.0 protocol.
- Integrates the USB controller and USB 3.0 femtoPHY.
- Supports the Super-Speed, High-speed, Full-speed, and low speed in host mode.
- Supports the Super-Speed, High-speed, and Full-speed in device mode.
- Supports the Link Power Management (LPM) protocol.
- Supports a maximum of 16 IN endpoints and 16 OUT endpoints when functioning as a device.
- Uses endpoint 0 as the control endpoint. The type of endpoint 1 to endpoint 15 can be set to bulk, isochronous, or interrupt.
- Provides an independent TX FIFO for each endpoint. The FIFO size is dynamically configurable.
- Complies with the eXtensible Host Controller Interface 1.0 (xHCI 1.0) protocol when functioning as a host.
- Supports the built-in DMA in scatter or gather mode.
- Supports the PHY interface for the PCI Express (PIPE) between the controller and PHY when working in Super-Speed mode. The clock rate is 125 MHz.
- Supports the USB 2.0 transceiver macrocell interface (UTMI) between the controller and PHY when working in High-speed or Full-speed mode. The clock rate is 60 MHz.
- Complies with the BC 1.2 protocol (excluding ACA).

8.1.2 Register Description

See the Synopsys IP manual (https://www.synopsys.com/cn/IP/InterfaceIP/USB/Pages/default.aspx).

8.2 UART

8.2.1 Function Description

Features

The universal asynchronous receiver transmitter (UART) performs serial-to-parallel conversion on the receive (RX) data and parallel-to-serial conversion on the transmit (TX) data.

The Hi3660 integrates nine UARTs and provides nine UART interfaces to the outside. All the UARTs support flow control except UART7. The UARTs described here do not include the UART used for modem debugging.

UART6 supports up to 1.2 Mbaud rate, and the other UARTs support up to 9 Mbaud rate.

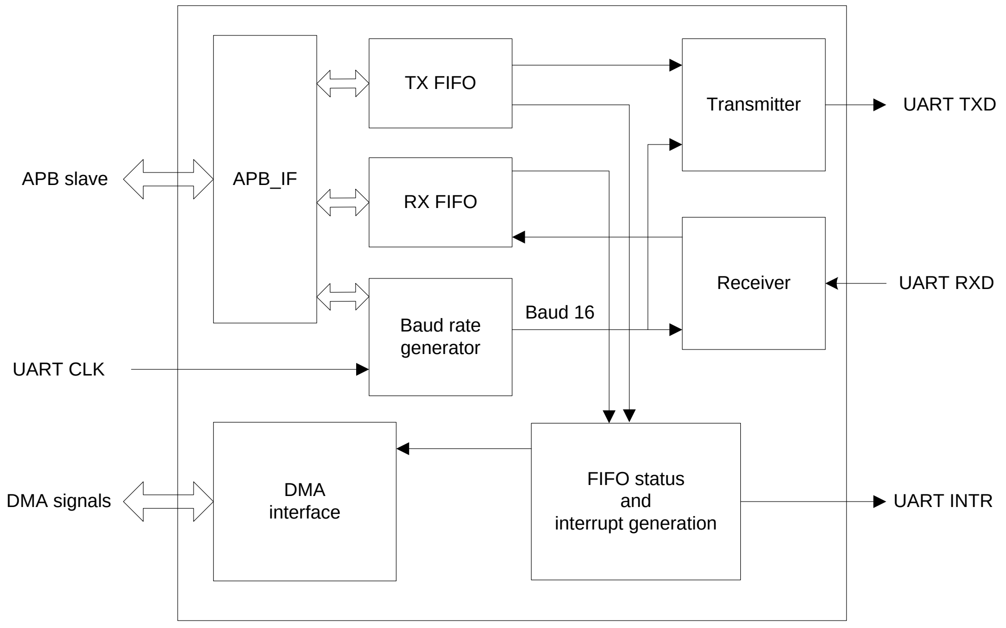
The UART module has the following features:

- Configurable data bit width and stop bit width. The data bit width can be 5, 6, 7, or 8 bits, and the stop bit width can be 1 bit or 2 bits
- Parity check or no check bit
- Programmable transfer rate, up to 9 Mbit/s
- Data transfer in direct memory access (DMA) mode (UART7 does not support DMA.)
- RX FIFO interrupt, TX FIFO interrupt, RX timeout interrupt, and error interrupt
- TX FIFO with 64-bit depth and 8-bit width and RX FIFO with 64-bit depth and 12-bit width (For UART7 and UART8, the depth of the TX FIFO and RX FIFO is 16 bits.)
- Infrared Data Association (IrDA) serial infrared mode

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

Logical Block Diagram

Figure 8-1 Logical block diagram of the UART module



The UART module consists of eight units. The function or operating principle of each unit is described as follows:

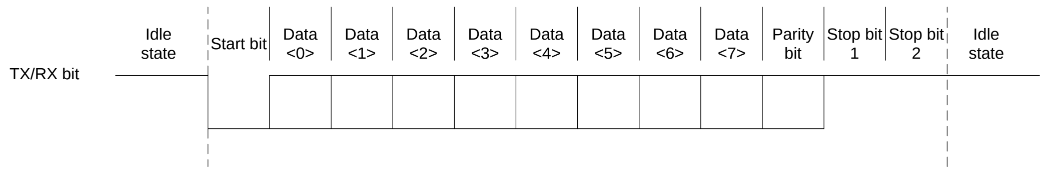
- Advanced peripheral bus (APB) interface: APB slave interface used to process the read/write data of the bus and configure registers
- TX FIFO: Stores the data to be transmitted.
- RX FIFO: Stores the received data.
- Baud rate generator: Generates the configured baud 16 clock.
- Transmitter: Performs parallel-to-serial conversion on the data and then outputs data, and decodes data into the infrared format for output.
- Receiver: Performs serial-to-parallel conversion on the received data and decodes data in the infrared format.
- DMA interface: Handles DMA interface processing.
- FIFO status and interrupt generation: Monitors the FIFO state and generates interrupts based on the configured interrupt FIFO level.

Frame Format

One frame transfer of the UART involves the start bit, data bits, parity bit, and stop bits, as shown in Figure 8-2.

The TX and RX data is the non-return-to-zero (NRZ) code. The data frame is output from the TXD end of one UART and then input to the RXD end.

Figure 8-2 UART frame format



The fields in the frame format are described as follows:

- Start bit

  It indicates the start of a data frame. According to the UART protocol, a low level in the TX signal indicates the start of a data frame. When the UART does not transmit data, the start bit must be fixed at 1.

- Data bit

  The data bit width can be set to 5 bits, 6 bits, 7 bits, or 8 bits based on the application requirements.

- Parity bit

  The parity bit is a 1-bit error correction signal. The UART parity bit can be an odd parity bit, even parity bit, or stick parity bit. The parity bit can be enabled or disabled.

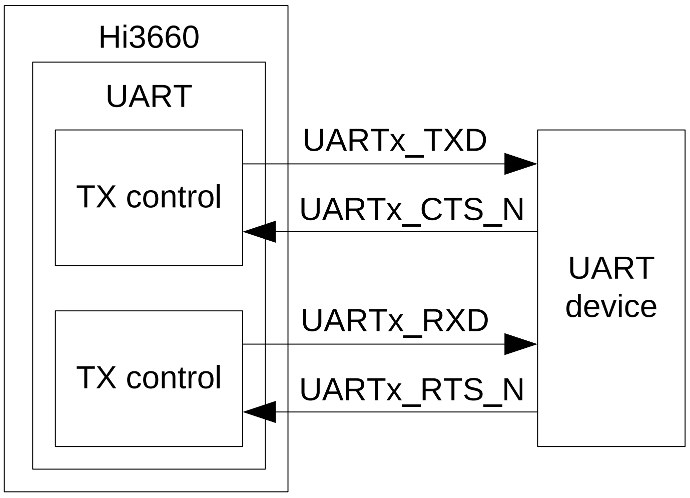
- Stop bit

  The stop signal is the stop bit of a data frame. The stop bit width is 1 bit or 2 bits. Setting the TX signal to 1 indicates the end of a data frame.

Typical Application Scenario

The UART receives data from or transmits data to the off-chip component or interface complying with the UART protocol, and implements serial-to-parallel conversion on the RX data and parallel-to-serial conversion on the TX data. The UART can also transmit the infrared data that complies with the IrDA protocol.

Figure 8-3 Typical application scenario of the UART module



The UART supports the following operating modes:

- UART mode: Supports flow control and reception and transmission of the data complying with the UART protocol.
- Infrared mode: Supports the reception and transmission of the serial infrared data complying with the IrDA protocol.

8.2.2 Signal Description

Figure 8-4 UARTx signals

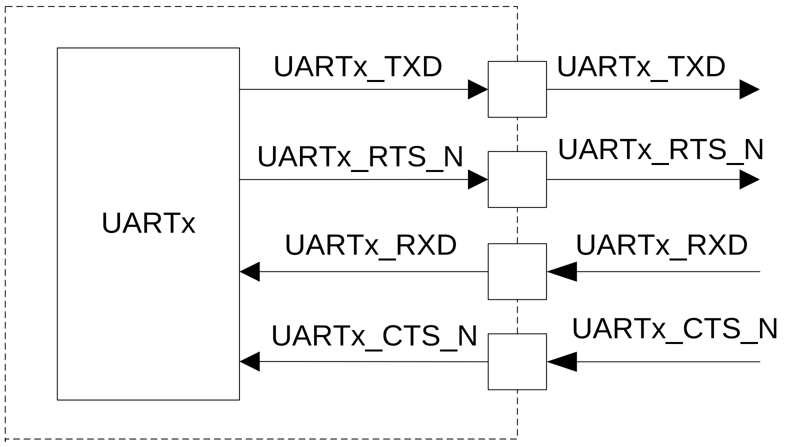


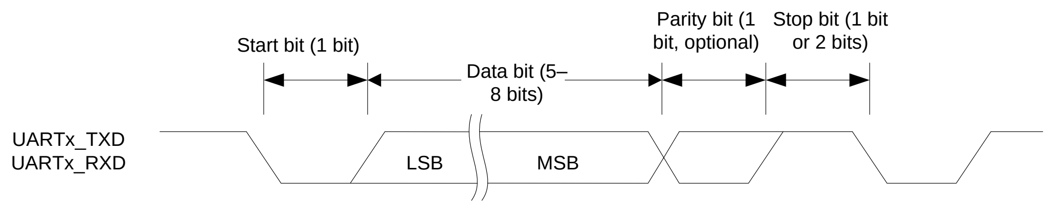
Table 8-1 UART interface signals

Signal	Direction	Description
UARTx_TXD	O	TX data signal of UART*x*
UARTx_RXD	I	RX data signal of UART*x*
UARTx_RTS_N	O	Request to send (RTS) signal of UART*x*
UARTx_CTS_N	I	Clear to send (CTS) signal of UART*x*

8.2.3 Timing

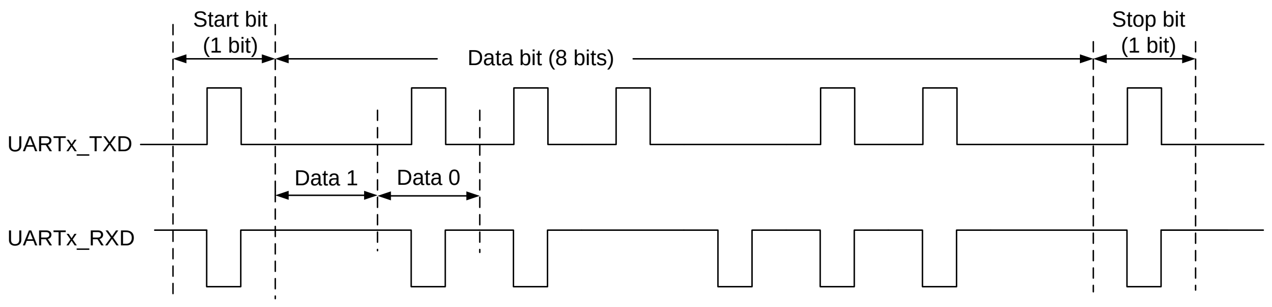
UART Timing

Figure 8-5 UART timing



Timing in Infrared Mode

Figure 8-6 UART timing in infrared mode



8.2.4 Register Description

See the ARM public-version PL011 IP manual (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0183g/index.html).

8.3 SPI

8.3.1 Function Description

Features

The serial peripheral interface (SPI) transfers data in serial or parallel mode. In the Hi3660, the SPI is used as the master to implement synchronous serial communication with external devices. The SPI does not serve as a slave.

The Hi3660 provides five SPIs: SPI0 to SPI4.

- SPI0 is reserved temporarily.
- SPI1 has one chip select (CS) for connecting the ESE.
- SPI2 has four CSs.

- SPI2_CS0 connects to the fingerprint sensor.
- SPI2_CS1 to SPI2_CS3 are reserved.

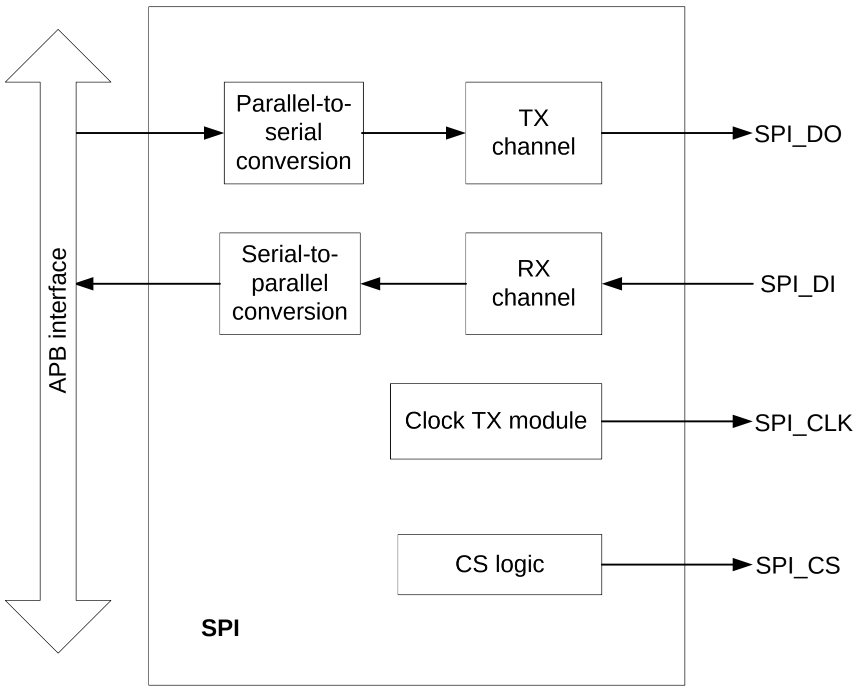
- SPI3 has four CSs.
  - SPI3_CS0 connects to the MINI_ISP.
  - SPI3_CS1 connects to the ink screen.
  - SPI3_CS2 and SPI3_CS3 are reserved.
- SPI4 has four CSs.
  - SPI4_CS0 connects to the fingerprint sensor.
  - SPI4_CS1 to SPI4_CS3 are reserved.

The SPI module supports the following functions:

- Supports the programmable interface clock frequency.
- Provides two separate 256 x 16-bit FIFOs: one TX FIFO and one RX FIFO.
- Supports SPI frame formats.
- Supports the programmable length of the serial data frame: 4 bits to 16 bits.
- Supports the programmable threshold of the TX FIFO and RX FIFO to request interrupts.
- Supports the programmable threshold for the TX FIFO and RX FIFO to request the DMA to implement burst transfer.
- Independently masks TX FIFO interrupts, RX FIFO interrupts, RX timeout interrupts, and RX FIFO overflow interrupts.
- Provides the internal loopback test.
- Supports the DMA operation.

Logical Block Diagram

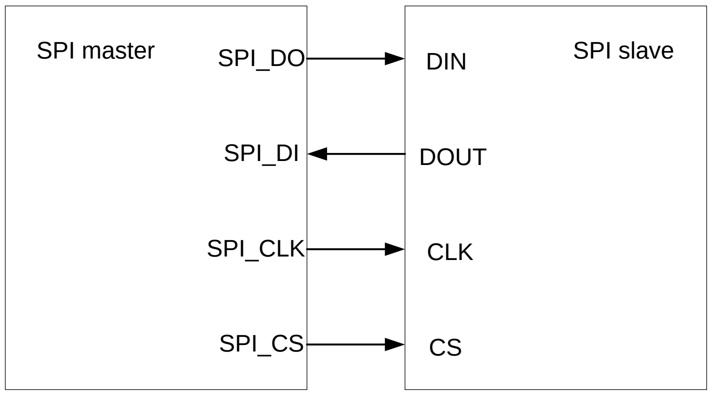
Figure 8-7 Functional block diagram of the SPI module



HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

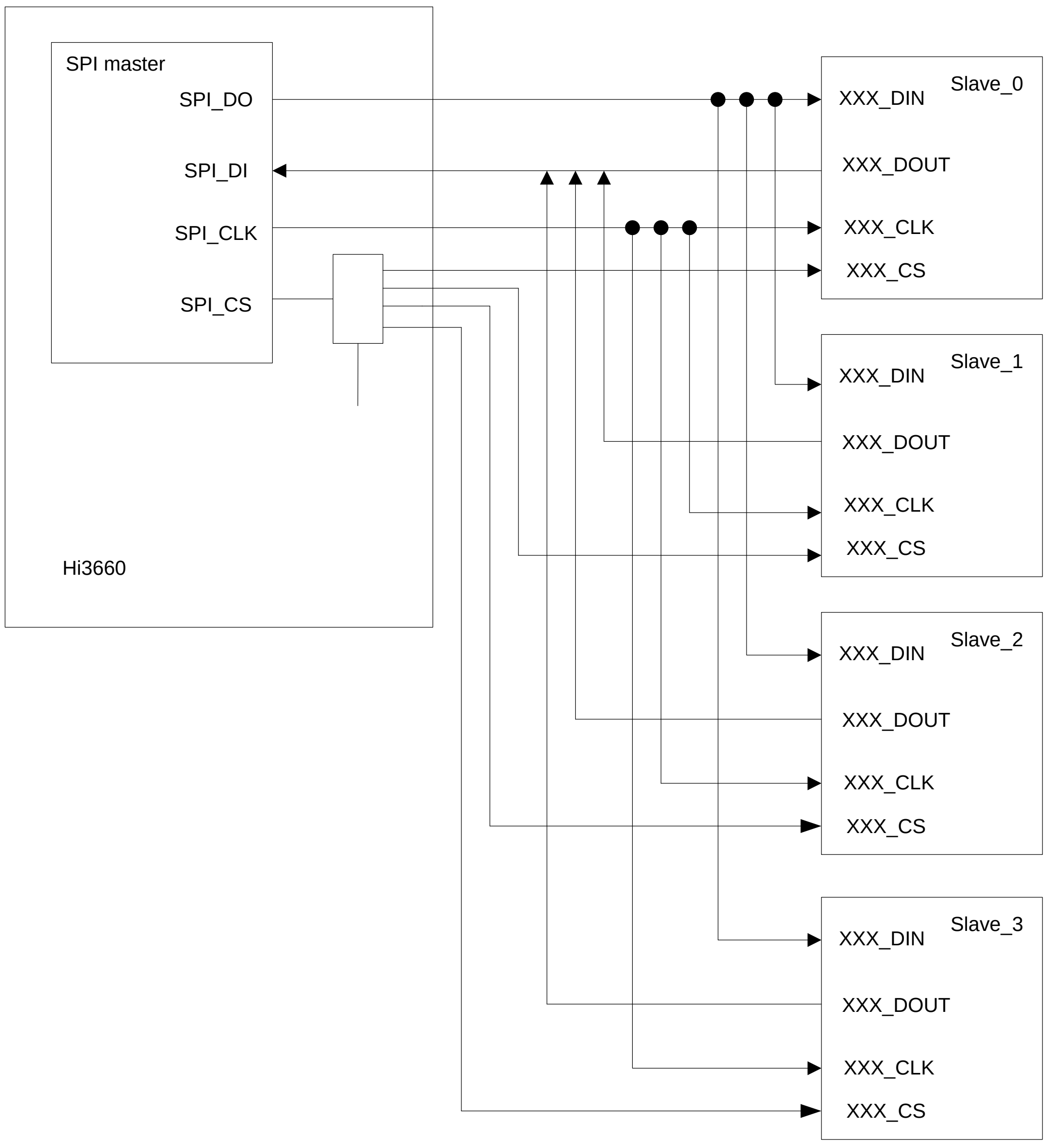
SPI0 to SPI4 can connect to a single slave device, as shown in Figure 8-8.

Figure 8-8 Application of the SPI connected to a single slave device



SPI0, SPI2, SPI3, and SPI4 can connect to multiple slave devices, as shown in Figure 8-9.

Figure 8-9 Application of the SPI connected to multiple slave devices



Typical Application Scenario

The SPI supports two data transfer modes:

- Data transfer in interrupt or query mode
- Data transfer in DMA mode

8.3.2 Interrupt Handling

The SPI has five interrupts. The first four interrupts have independent sources and are maskable and active high.

- SPIRXINTR

  RX FIFO interrupt. When there are four or more valid data segments in the RX FIFO, this interrupt is enabled.

- SPITXINTR

  TX FIFO interrupt. When there are four or fewer valid data segments in the TX FIFO, this interrupt is enabled.
- SPIRORINTR

  RX overflow interrupt. When the FIFO is full and new data needs to be written to the FIFO, FIFO overflow occurs and this interrupt is enabled. In this case, data is written to the RX shift register rather than the FIFO.
- SPIRTINTR

  RX timeout interrupt. When the RX FIFO is not empty and the SPI is idle for more than one fixed 32-bit cycle, this interrupt is enabled.

  In this case, data in the RX FIFO needs to be transmitted. When the RX FIFO is read empty or new data is received into SPIRXD, this interrupt is disabled. This interrupt can be cleared by writing the SPIICR[RTIC] register.
- SPIINTR

  Combined interrupt, which is obtained after the preceding four interrupts are ORed. If any of the preceding four interrupts is enabled, this combined interrupt is enabled.

The IDs of the SPI0–4 interrupts are 145, 112, 148, 344, and 345, respectively.

8.3.3 Signal Description

Table 8-2 Signals of the SPI0 interface

Signal	Direction	Description
SPI0_CLK	Output	SPI clock, output in master mode
SPI0_CS0_N	Output	SPI CS0 in master mode
SPI0_CS1_N	Output	SPI CS1 in master mode
SPI0_CS2_N	Output	SPI CS2 in master mode
SPI0_CS3_N	Output	SPI CS3 in master mode
SPI0_DI	Input	RX data input of the SPI
SPI0_DO	Output	TX data output of the SPI

Table 8-3 Signals of the SPI1 interface

Signal	Direction	Description
SPI1_CLK	Output	SPI clock, output in master mode
SPI1_CS_N	Output	SPI CS0 in master mode
SPI1_DI	Input	RX data input of the SPI
SPI1_DO	Output	TX data output of the SPI

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

Table 8-4 Signals of the SPI2 interface

Signal	Direction	Description
SPI2_CLK	Output	SPI clock, output in master mode
SPI2_CS0_N	Output	SPI CS0 in master mode
SPI2_CS1_N	Output	SPI CS1 in master mode
SPI2_CS2_N	Output	SPI CS2 in master mode
SPI2_CS3_N	Output	SPI CS3 in master mode
SPI2_DI	Input	RX data input of the SPI
SPI2_DO	Output	TX data output of the SPI

Table 8-5 Signals of the SPI3 interface

Signal	Direction	Description
SPI3_CLK	Output	SPI clock, output in master mode
SPI3_CS0_N	Output	SPI CS0 in master mode
SPI3_CS1_N	Output	SPI CS1 in master mode
SPI3_CS2_N	Output	SPI CS2 in master mode
SPI3_CS3_N	Output	SPI CS3 in master mode
SPI3_DI	Input	RX data input of the SPI
SPI3_DO	Output	TX data output of the SPI

Table 8-6 Signals of the SPI4 interface

Signal	Direction	Description
SPI4_CLK	Output	SPI clock, output in master mode
SPI4_CS0_N	Output	SPI CS0 in master mode
SPI4_CS1_N	Output	SPI CS1 in master mode
SPI4_CS2_N	Output	SPI CS2 in master mode
SPI4_CS3_N	Output	SPI CS3 in master mode
SPI4_DI	Input	RX data input of the SPI
SPI4_DO	Output	TX data output of the SPI

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

8.3.4 Timings and Parameters

The acronyms in Figure 8-10 to Figure 8-17 are described as follows:

- MSB: most significant bit
- LSB: least significant bit
- Q: undefined signal

SPI Frame Formats

NOTE
SPO indicates the polarity of SPICLKOUT and SPH indicates the phase of SPICLKOUT. The corresponding register bits are SPICR0 bit[7] and SPICR0 bit[6], respectively.

(1) SPO = 0 and SPH = 0

Figure 8-10 Format of a single SPI frame (SPO = 0, SPH = 0)

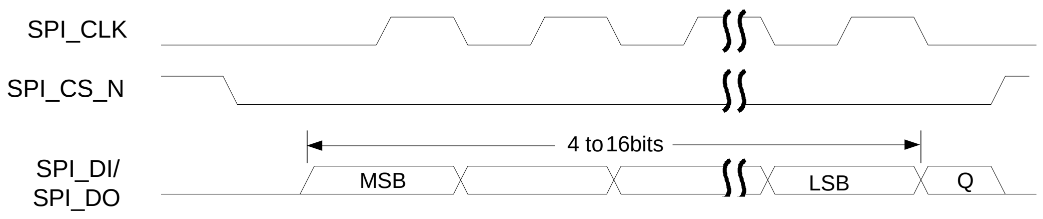
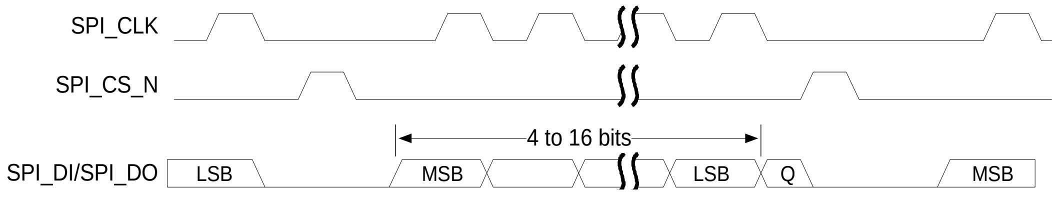


Figure 8-11 Format of consecutive SPI frames (SPO = 0, SPH = 0)



When the SPI is idle in this mode:

- The SPI_CLK signal is set to low.
- The SPI_CS_N signal is set to high.
- The TX data line SPI_DO is forced to low.

When the SPI is enabled and valid data is ready in the TX FIFO, setting the SPI_CS_N signal to low starts data transfer, and data of the enabled slave is placed on the master RX data line SPI_DI. Half an SPI_CLK cycle later, the valid master data is transmitted to SPI_DO. At this time, both the master data and slave data are valid. The SPI_CLK pin changes to high level half an SPI_CLK cycle later. Data is captured on the rising edge of the SPI_CLK clock and transmitted on the falling edge.

If a single word is transferred, SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

If consecutive words are transferred, the SPI_CS_N signal must be pulled up by one SPI_CLK cycle at each word transfer interval. This is because when SPH is 0, the slave selection pin retains the data in the internal serial device register. Therefore, the master device must pull the SPI_CS_N signal up at each word transfer interval during consecutive transfer. When the transfer of consecutive words is complete, SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured.

(2) SPO = 0 and SPH = 1

Figure 8-12 Format of a single SPI frame (SPO = 0, SPH = 1)

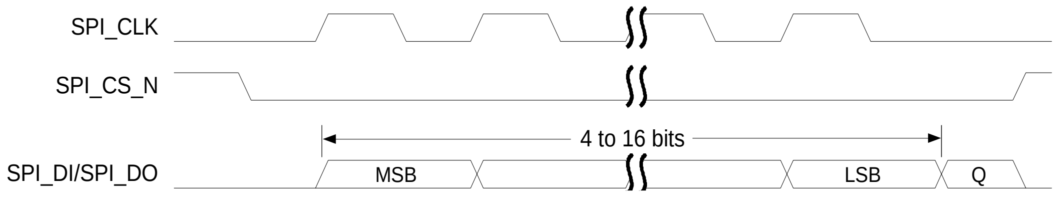
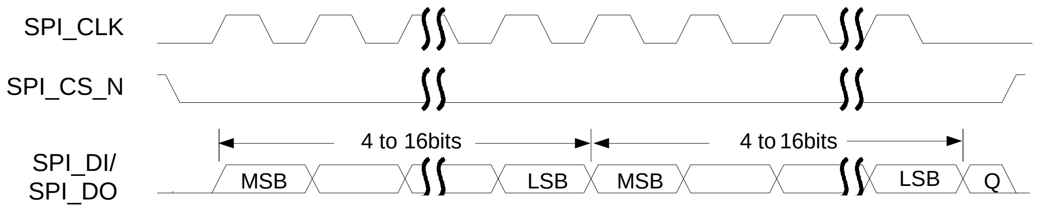


Figure 8-13 Format of consecutive SPI frames (SPO = 0, SPH = 1)



When the SPI is idle in this mode:

- The SPI_CLK signal is set to low.
- The SPI_CS_N signal is set to high.
- The TX data line SPI_DO is forced to low.

When the SPI is enabled and there is valid data in the TX FIFO, setting the SPI_CS_N signal to low starts a data transfer. If data transfer starts, the master data and slave data are valid on their respective transmission lines half an SPI_CLK cycle later. In addition, SPI_CLK becomes valid on the first rising edge. Data is captured on the falling edge of the SPI_CLK clock and transmitted on the rising edge.

If a single word is transferred, SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured.

If consecutive words are transferred, SPI_CS_N retains low at the word transfer interval. When the consecutive transfer ends, SPI_CS_N is restored to high level one SPI_CLK cycle later after the last 1-bit data is captured.

(3) SPO = 1 and SPH = 0

Figure 8-14 Format of a single SPI frame (SPO = 1, SPH = 0)

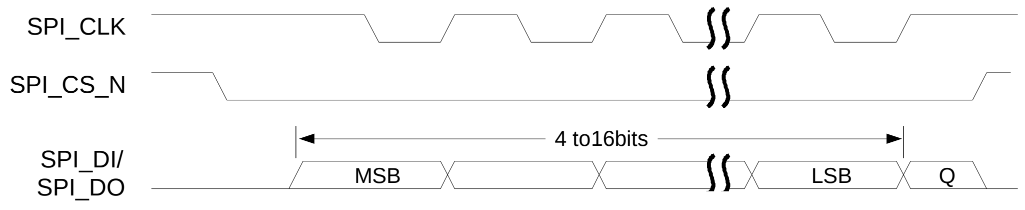
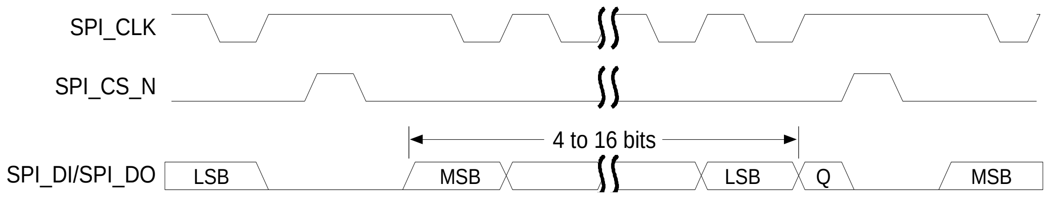


Figure 8-15 Format of consecutive SPI frames (SPO = 1, SPH = 0)



When the SPI is idle in this mode:

- The SPI_CLK signal is set to high.
- The SPI_CS_N signal is set to high.
- The TX data line SPI_DO is forced to low.

When the SPI is enabled and there is valid data in the TX FIFO, setting the SPI_CS_N signal to low starts a data transfer. The slave data is immediately transmitted to the master RX data line SPI_DI. Half an SPI_CLK cycle later, the valid master data is transmitted to SPI_DO. Another half SPI_CLK cycle later, the SPI_CLK master pin is set to low, indicating that data is captured on the falling edge of the SPI_CLK clock and transmitted on the rising edge.

If a single word is transferred, SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured.

If consecutive words are transferred, the SPI_CS_N signal must be pulled up at each word transfer interval. This is because when SPH is 0, the slave selection pin retains the data in the internal serial device register. SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured.

(4) SPO = 1 and SPH = 1

Figure 8-16 Format of a single SPI frame (SPO = 1, SPH = 1)

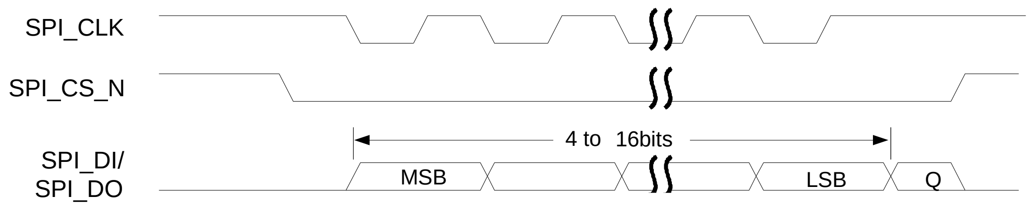
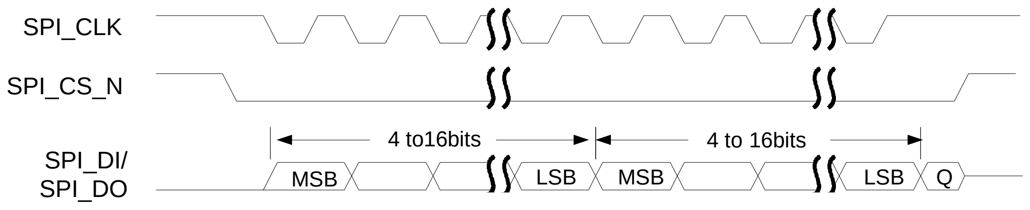


Figure 8-17 Format of consecutive SPI frames (SPO = 1, SPH = 1)



When the SPI is idle in this mode:

- The SPI_CLK signal is set to high.
- The SPI_CS_N signal is set to high.
- The TX data line SPI_DO is forced to low.

When the SPI is enabled and there is valid data in the TX FIFO, setting the SPI_CS_N master signal to low starts the data transfer. Half an SPI_CLK cycle later, the master data and slave data are valid on their respective transmission lines. In addition, SPI_CLK becomes valid on the first falling edge. Data is captured on the rising edge of the SPI_CLK clock and transmitted on the falling edge.

If a single word is transferred, SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured.

If consecutive words are transferred, the SPI_CS_N signal retains low. SPI_CS_N is restored to high level one SPI_CLK cycle after the last bit of data is captured. For the transfer of consecutive words, SPI_CS_N retains low during data transfer, and the end mode is the same as that during single word transfer.

Timing

Figure 8-18 SPI timing

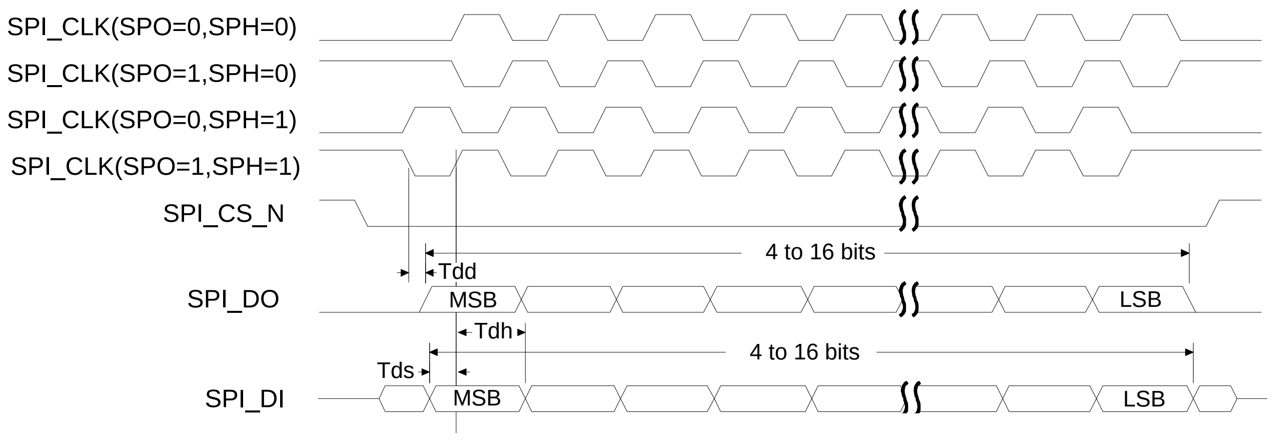


Table 8-7 Input timing parameters of the SPI

Parameter	Symbol	Min	Max	Unit
DI setup time	Tds	10	-	ns
DI hold time	Tdh	10	-	ns

8.3.5 Register Description

See the ARM public-version PL022 IP manual (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0183g/index.html).

8.4 I^{2}C

8.4.1 Function Description

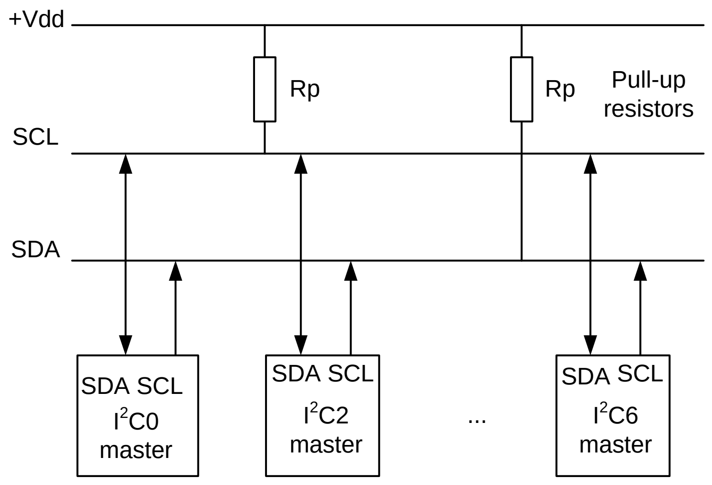
Functional Block Diagram

The I^{2}C controller uses the SCL and SDA signal lines to communicate with off-chip devices that provide I^{2}C interfaces. The I^{2}C interface can transmit data to and receive data from the slave device on the I^{2}C bus in compliance with I^{2}C specifications V2.1. In the Hi3660, the I^{2}C controller can only be used as the master device.

The Hi3660 integrates eight I^{2}C modules.

- I^{2}C0 is used to connect sensors, such as the acceleration sensor, gyro sensor, and barometer.
- I^{2}C1 is the backup of I^{2}C0.
- I^{2}C2 is used to connect the capacitive touch pad (TP).
- I^{2}C3 can be multiplexed as the common GPIO function.
- I^{2}C4 is used for charging, Near Field Communication (NFC), external flash light driver, and external speaker PA.
- I^{2}C5 is used to connect the external buck power chip.
- I^{2}C6 is reserved for the backup solution.
- I^{2}C7 is reserved.

Figure 8-19 Typical I^{2}C application circuit



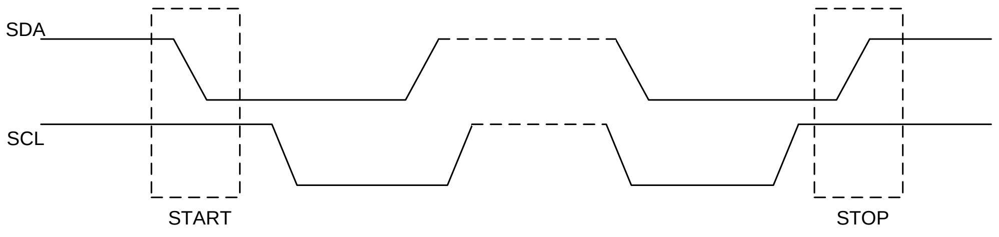
The I^{2}C controller has the following features:

- Supports the I^{2}C bus protocol V2.1.
- Serves as only the master device on the I^{2}C bus.
- Acts as the transmitter (master device) on the I^{2}C bus and sends data to the slave device.
- Supports the 7-bit standard slave address or 10-bit extended slave address when serving as a master.
- Supports the standard mode (100 kbit/s), fast mode (400 kbit/s), and high-speed mode (3.4 Mbit/s).
- Provides the TX FIFO and RX FIFO and supports DMA data transfer.
- Reports interrupts and queries the states of raw and masked interrupts.
- Supports clock stretching. During data transmission, when the TX FIFO is empty, pull down the SCL and wait for the FIFO to be filled with data again. During data reception, when the RX FIFO is empty, pull down the SCL and wait for the FIFO to be filled with data again.
- Supports the restart transmission of the SDA data. The data bus is not released and data continues to be transferred in the same channel.
- Configures the SDA setup time and hold time in registers.

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

START/STOP Conditions

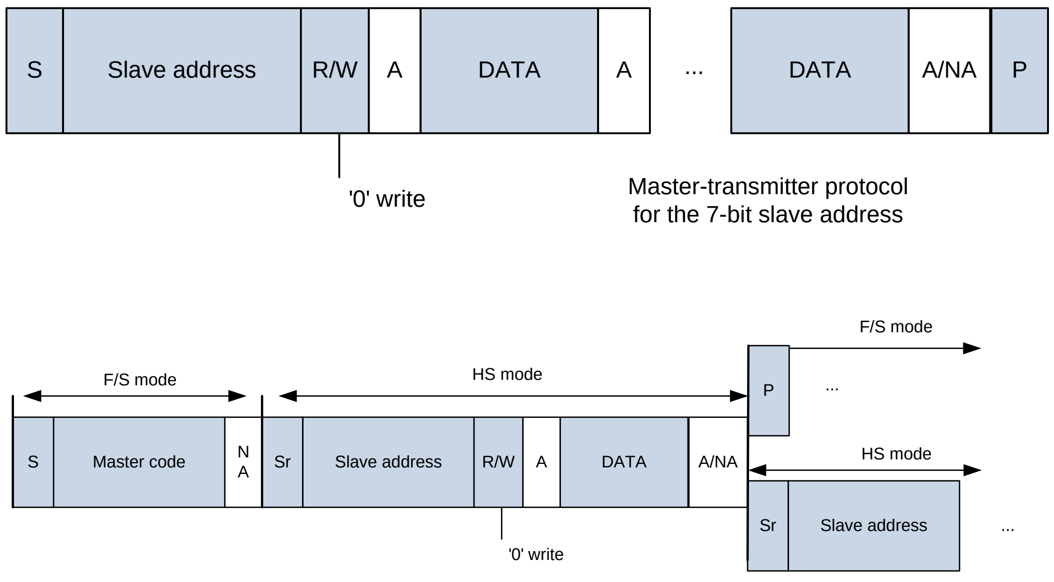
Figure 8-20 I^{2}C START/STOP timing



- START: The SDA level is switched from high to low when the SCL level is high.
- STOP: The SDA level is switched from low to high when the SCL level is high.

I^{2}C Frame format

Figure 8-21 I^{2}C frame formats in different transfer modes



NOTE
- S: start condition
- A: acknowledge (SDA low)
- P: stop condition
- NA: no acknowledge (SDA high)

The start signal (Start) specified in the I^{2}C standard protocol is transmitted from the master device on the bus to wake up all slave devices. The Start is a special signal indicating the start of data transfer.

When the master device works in F/S mode, the first data packet transmitted after the Start signal is sent is the address of the slave device (Slave address + W/R). This data packet consists of the 7-bit slave address (two data packets are required if the slave address is 10 bits) and 1-bit read/write data.

When working in HS mode, the master device needs to transmit the master code first. Then the frame format in HS mode is the same as that in F/S mode. The HS mode and F/S mode differ only in the speed. Each slave address determines whether to transmit or receive data based on the slave address information. After a slave address is successfully received, the corresponding slave device pulls the SDA down at the ninth clock cycle (SCL) and returns an ACK signal to the master device. Then, subsequent data transfer starts.

DATA refers to data reception or transmission based on the read or write command after the master device successfully receives the slave address ACK signal. During the data transfer, the SDA changes when the SCL level is low and retains when the SCL level is high. Each time the receiver (slave device) receives one byte, it must send an ACK signal to the transmitter (master device). If the transmitter fails to receive an ACK signal, it stops data transfer or starts re-transmission.

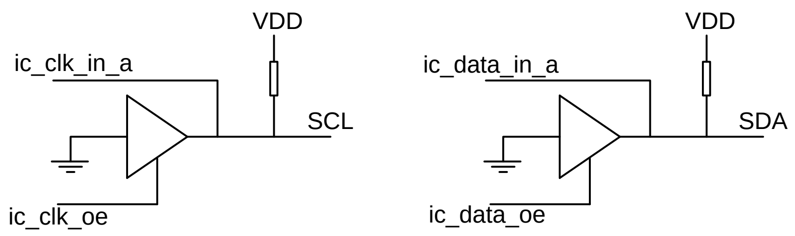
The stop signal (Stop) is sent when the master device completes the current data transfer and there is no new data to be transferred. The stop signal is a special signal indicating the end of data transfer, and complies with the standard I^{2}C protocol. After the master device sends a Stop signal, the slave device must release the bus.

8.4.2 Signal Description

8.4.2.1 OD Gate of the Interface

The I^{2}C interface uses the open drain (OD) gate, as shown in Figure 8-22.

Figure 8-22 Inputs and outputs of the OD gates for the SCL and SDA



8.4.2.2 Interface Signals

Table 8-8 describes the signals of the I^{2}C interface. For details about the multiplexing relationship of the signals in Table 8-8, see the IOC-related documents.

Table 8-8 Signals of the I^{2}C interface

Signal	Direction	Description
I2C0_SDA	Input/Output	I^{2}C0 data signal

HiSilicon Proprietary and Confidential
Copyright © HiSilicon Technologies Co., Ltd.

Signal	Direction	Description
I2C0_SCL	Output	I^{2}C0 clock signal
I2C2_SDA	Input/Output	I^{2}C2 data signal
I2C2_SCL	Output	I^{2}C2 clock signal
I2C3_SDA	Input/Output	I^{2}C3 data signal
I2C3_SCL	Output	I^{2}C3 clock signal
I2C4_SDA	Input/Output	I^{2}C4 data signal
I2C4_SCL	Output	I^{2}C4 clock signal
I2C5_SDA	Input/Output	I^{2}C5 data signal
I2C5_SCL	Output	I^{2}C5 clock signal
I2C6_SDA	Input/Output	I^{2}C6 data signal
I2C6_SCL	Output	I^{2}C6 clock signal
I2C7_SDA	Input/Output	I^{2}C7 data signal
I2C7_SCL	Output	I^{2}C7 clock signal

8.4.3 Timings and Parameters

Figure 8-23 I^{2}C timing

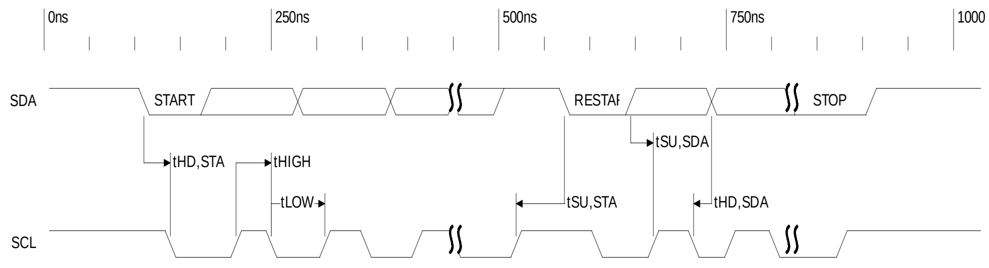


Table 8-9 I^{2}C timing parameters (F/S mode)

Parameter	Description	Standard Mode		Fast Mode		Unit
		Min	Max	Min	Max	
fSCL	SCL clock frequency	0	100	0	400	kHz
tHD;STA	Hold time of the START condition	4.0	-	0.6	-	µs
tLOW	Width of the SCL low level	4.7	-	1.3	-	µs

Parameter	Description	Standard Mode		Fast Mode		Unit
		Min	Max	Min	Max	
tHIGH	Width of the SCL high level	4.0	-	0.6	-	µs
tSU;STA	Setup time of the START condition	4.7	-	0.6	-	µs
tHD;DAT	Data hold time[a]	0	3.45	0	0.9	µs
tSU;DAT	Data setup time[a]	250	-	100	-	ns
tR	Rising time of the SDA and SCL signals	-	1000	20+0.1Cb	300	ns
tF	Falling time of the SDA and SCL signals	-	300	20+0.1Cb	300	ns
tSU;STO	Setup time of the STOP condition	4.0	-	0.6	-	µs
tBUF	Bus idle duration between the STOP and START states	4.7	-	1.3	-	µs
Cb	Capacitor load of each bus line	-	400	-	400	pF

a: For the Hi3660, the registers related to the data hold time and setup time are configurable.

Table 8-10 I^{2}C timing parameters (HS mode)

Parameter	Description	Cb (Max) = 100 pF		Cb = 400 pF		Unit
		Min	Max	Min	Max	
fSCL	SCL clock frequency	0	3.4	0	1.7	MHz
tSU;STA	Setup time of the START/RESTART condition	160	-	160	-	ns
tHD;STA	Hold time of the START/RESTART condition	160	-	160	-	ns
tLOW	Width of the SCL low level	160	-	320	-	ns
tHIGH	Width of the SCL high level	60	-	120	-	ns
tSU;DAT	Data setup time[a]	10	-	10	-	ns
tHD;DAT	Data hold time[a]	0	70	0	150	ns

Parameter	Description	Cb (Max) = 100 pF		Cb = 400 pF		Unit
		Min	Max	Min	Max	
trCL	Rising time of the SCL signal	10	40	20	80	ns
trCL1	Rising time of the SCL signal after the RESTART and response bit	10	80	20	160	ns
tF	Falling time of the SCL signal	10	40	20	80	ns
trDA	Rising time of the SDA signal	10	80	20	160	ns
tfDA	Falling time of the SD signal	10	80	20	160	ns
tSU;STO	Setup time of the STOP condition	160	-	160	-	ns
Cb	Capacitor load between the SCL and SDA	-	100	-	400	pF

a: For the Hi3660, the registers related to the data hold time and setup time are configurable.

CAUTION

The conditions for the I^{2}C data transfer rate to reach 3.4 Mbit/s are as follows:

- The maximum value of **tHD:DAT** must be no greater than 70 ns.
- The minimum data window duration must be 230 ns.
- The maximum hold time must be 70 ns.

8.4.4 Register Description

See the Synopsys I^{2}C IP manual (https://www.synopsys.com/dw/ipdir.php?c=DW_apb_i2c).

8.5 GPIO

8.5.1 Function Description

Features

The Hi3660 has 27 common GPIO modules: GPIO0–17, GPIO20–21, and GPIO22–28. The peripheral area has 20 GPIO modules (GPIO0–17 and GPIO20–21), and the always-on area (AON_SUBSYS) has seven GPIO modules (GPIO22–28).

The Hi3660 provides two secure GPIO modules: one (GPIO0_SE) in the peripheral area and one (GPIO1_SE) in the always-on area (AON_SUBSYS).

The Hi3660 also has the following GPIO modules in the independent subsystems:

- Four GPIO modules in the sensor hub: GPIO0_SH to GPIO3_SH
- Two GPIO modules in the UFS_PERI_SUBSYS area: GPIO18 and GPIO 19
- Two GPIO modules in the MMC1_PERI_SUBSYS area: GPIO0_EMMC and GPIO1_EMMC

Each GPIO module corresponds to a group of eight GPIO interfaces. Each GPIO group generates three interrupts, which are sent to the generic interrupt controller (GIC), LPMCU, and CCPU, respectively. (The interrupts of GPIO0SH to GPIO3SH are sent to the GIC and IOMCU, respectively.) The source interrupts of the three index interrupts share the eight GPIO interfaces in the same group but have independent mask bits.

Table 8-11 GPIO group information

Instance	Valid GPIO Pins	Interrupt Registration Support
GPIO22 to GPIO27	GPIO_176 to GPIO_190 GPIO_192 to GPIO_222	GIC
GPIO28	GPIO28 is not used.	GIC
GPIO1_SE	GPIO_008_SE to GPIO_015_SE	GIC
GPIO0 to GPIO17	GPIO_001 to GPIO_098 GPIO_103 to GPIO_127	GIC
GPIO20	GPIO_160 to GPIO_165	GIC
GPIO21	GPIO_168 to GPIO_173	GIC
GPIO0_SE	GPIO_000_SE to GPIO_007_SE	GIC
GPIO0_SH to GPIO3_SH	GPIO_000_SH to GPIO_027_SH	GIC
	GPIO_028_SH to GPIO_031_SH	GIC
GPIO18 to GPIO19	GPIO_144 to GPIO_155	GIC
GPIO0_EMMC to GPIO1_EMMC	GPIO_00_EMMC to GPIO_09_EMMC	GIC

Each GPIO group provides eight programmable I/O pins to generate output signals or collect input signals for specific applications.

You can obtain the group ID of a GPIO pin by using the following formula:

Group ID = int(GPIO pin number/8)

The remainder is the sequence number of this pin in the group, ranging from 0 to 7. (0 is the LSB and 7 is the MSB.)

Sequence number within the group = mod(GPIO pin number/8)

Take GPIO_017 as an example. The pin number is 17. Its group ID is 2 (int(17/8)), and the remainder is 1. Therefore, GPIO_017 belongs to GPIO group 2 (GPIO2), and its sequence number within this group is 1. Similarly, the sequence number of GPIO_016 in GPIO2 is 0, and that of GPIO_023 in GPIO2 is 7.

The GPIO has the following features:

- Configures each GPIO pin as the input, output, or OD output.
  - When acting as an input pin, a GPIO pin can be used as an interrupt source. Each GPIO pin supports independent interrupt control.
  - When acting as an output pin, a GPIO pin can be independently set to 0 or 1.
  - When a GPIO pin acts as the OD output, pull-up control is required on the board. The output is enabled by using the GPIODIR register to implement the "wired AND" function on the board.
- Queries the states of raw and masked interrupts.
- Supports the interrupt wakeup system. Configure the GPIO enable interrupt before the system enters the sleep state. After the system enters the sleep state, the GPIO will generate an interrupt to wake up the system when peripheral inputs change.
- Does not support separate soft reset.

8.5.2 Signal Description

NOTE
GPIO_000 is an internal signal and is invisible to users.

Table 8-12 GPIO interface signals

Signal	Direction	Description
GPIO_001	Input/Output	Standard I/O port
GPIO_002	Input/Output	
…		
GPIO_174	Input/Output	
GPIO_222	Input/Output	

8.5.3 Register Description

See the ARM public-version PL061 IP manual (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0183g/index.html).