19 I 2C

19.1 Overview

The I2C module implements synchronous communication between the MCU and external I2C interface devices, and implements serial-to-parallel conversion by hardware. Supports I2C master and slave

The system only supports single-host mode and does not support multi-host mode.

Features:

ÿ 1 independent I2C interface

ÿ Supports master and slave modes, does not support multi-master mode

ÿ Support 7-bit or 10-bit slave address

ÿ Transmission speed supports standard mode (100Kbps), fast mode (400Kbps) and Fm+ (1Mbps)

ÿ Support DMA, independent DMA channels for master and slave

ÿ Low power slave design, can send and receive data without system clock

ÿ Support asynchronous slave address match wake-up, data frame reception completion wake-up or START detection wake-up

19.2 Block Diagram

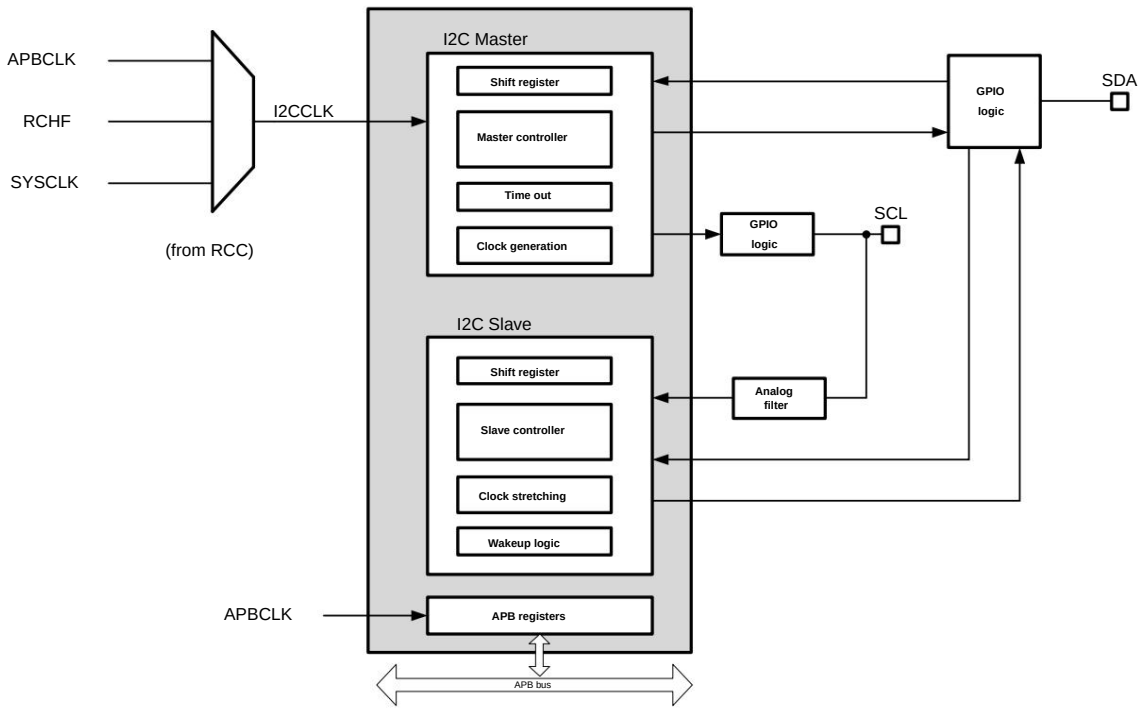

Figure **19-1 I2C** module block diagram

19.3 Pin Definition and Pull-up Resistor Range

The I2C module uses two true open-drain pins to communicate with external devices and requires external pull-up resistors for operation:

Pinout	**I2Cx**	symbol	Function
PA11	I2C	SCL	I2C Clock
PA12		SDA	I2C Data

Table **19-1 I2C** pin list

The I2C bus protocol specifies the maximum signal rise time for standard-mode, fast-mode, and fast-mode plus, as well as the minimum sink current that the IO can support. See the table below.

Symbol	**Parameter**	**Conditions**	**Standard-mode**		**Fast-mode**		**Fast-mode Plus Unit**		
			Min	**Max**	**Min**	**Max**	**Min**	**Max**	
WILL	LOW-level input voltage[1]		ÿ0.5	0.3VDD	ÿ0.5	0.3VDD	ÿ0.5	0.3VDD V	
HIV	HIGH-level input voltage[1]		0.7VDD	[2]	0.7VDD	[2]	0.7VDD[1]	[2]	In
Vhys	hysteresis of Schmitt trigger inputs		·	·	0.05VDD	·	0.05VDD	·	In
VOL1	LOW-level output voltage 1	(open-drain or open-collector) at 3 mA sink current; VDD>2V	0	0.4	0	0.4	0	0.4	In
VOL2	LOW-level output voltage 2	(open-drain or open-collector) at 2 mA sink current[3] ; VDDÿ2V	·	·	0	0.2VDD	0	0.2VDD V	
IOL	LOW-level output current	VOL = 0.4 V	3	·	3	·	20	·	m.a.
		VOL = 0.6 V[4]	·	·	6	·	·	·	m.a.
tof	output fall time from VIhminto VILmax		·	250[5]	ÿ 1 (VDD / 5.5 V)[6]	250[5]	ÿ 1 (VDD / 5.5 V)[6]	120[7]	ns
tSP	pulse width of spikes that must be suppressed by the input filter		·	·	0	50[8]	0	50[8]	ns
Ii	input current each I/O pin	0.1VDD< VI< 0.9VDdmax	ÿ10	+10	ÿ10[9]	+10[9]	ÿ10[9]	+10[9] ÿA	

There	capacitance for each I/O pin[10]		·	10	·	10	·	10	pF

Table **19-2 I2C** protocol electrical parameters

The following table defines the maximum allowed rise and fall times for bus signals.

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus Unit		
			Min	Max	Min	Max	Min	Max	
fSCL	SCL clock frequency		₀	100	₀	400	₀	1000 kHz	
tHD;STA	hold time (repeated) START condition After this period, the first clock pulse is generated.		4.0	·	0.6	·	0.26	·	ÿs
tLOW	LOW period of the SCL clock		4.7	·	1.3	·	0.5	·	ÿs
tHIGH	HIGH period of the SCL clock		4.0	·	0.6	·	0.26	·	ÿs
tSU;STA	set-up time for a repeated START condition		4.7	·	0.6	·	0.26	·	ÿs
tHD;DAT	data hold time[2]	CBUS compatible masters (see Remark in Section 4.1)	5.0	·	·	·	·	·	ÿs
		I2C-bus devices	0[3]	-[4]	0[3]	-[4]	₀	·	ÿs
tSU;DAT	data set-up time rise		250	·	100[5]	·	50	·	ns
tr	time of both SDA and SCL signals		·	1000	20	300	·	120	ns
tf	fall time of both SDA and SCL signals[3][6][7][8]		·	300	ÿ 1 (VDD / 5.5 V)	300	ÿ 1 (VDD / 5.5 V)[9]	120[8]	ns
tSU;STO	set-up time for STOP condition		4.0	·	0.6	·	0.26	·	ÿs
tBUF	bus free time between a STOP and START condition		4.7	·	1.3	·	0.5	·	ÿs
Cb	capacitive load for each bus line[10]		·	400	·	400	·	550	pF
tVD;DAT	data valid time[11]		·	3.45[4]	·	0.9[4]	·	0.45[4]	ÿs
tVD;ACK	data valid acknowledge time[12]		·	3.45[4]	·	0.9[4]	·	0.45[4]	ÿs
VnL	noise margin at the LOW level	for each connected device (including hysteresis)	0.1VDD	·	0.1VDD	·	0.1VDD	·	In
VnH	noise margin at the HIGH level	for each connected device (including hysteresis)	0.2VDD	·	0.2VDD	·	0.2VDD	·	In

Table **19-3 I2C** protocol timing parameters

Based on the above protocol specifications, we can calculate the reasonable range of external pull-up resistors.

Assuming the bus signal rises from VIL=0.3VDD to VIH=0.7VDD, the charging time can be calculated as:

V(t1) = 0.3 ÿ VDD = VDD (1 ÿ eÿt1 / RC); t1 = 0.3566749 ÿ RC

V(t2) = 0.7 ÿ VDD = VDD (1 ÿ eÿt2 / RC); t2 = 1.2039729 ÿ RC

T = t2 ÿ t1 = 0.8473 ÿ RC

Based on the bus capacitive load size and the protocol's requirements for the maximum signal rise time, we can calculate the maximum value of the pull-up resistor:

$$R_{(max)p} = \frac{t_r}{0.8473 \text{ ÿ } C_b}$$

The minimum value of the pull-up resistor is determined by the bus power supply voltage VDD and the IO current sink capability. The I2C pin sink capability of FM33LC0XX is 20mA, while the minimum sink capability required by the protocol is

It is 3mA in standard/fast mode and 20mA in Fm+ mode.

$$R_{p\ (min)} = \frac{VDD - V_{OL\ (max)}}{I_{OL}}$$

19.4 Clock

Both the I2C master and slave use a dual clock structure:

ÿ The bus register clock of the master and slave is represented by PCLK, which is derived from APBCLK.

When using the internal registers, PCLK must be enabled.

ÿ The host data receiving and sending clock is represented by I2CCLK, which can be derived from APBCLK, RCHF,

SYSCLK and RCMF can work independently of APBCLK. I2CCLK must be enabled to transmit and receive data.

ÿ The data transmission and reception clock of the slave uses the SCL bus clock input, so data can be transmitted and received without the system clock

The control of PCLK and I2CCLK is completed in the CMU module. Before I2C communication, the corresponding CMU control registers must be correctly configured.

Memory.

The dual clock structure can make the I2C work not limited by the APBCLK configuration. When some peripherals need to work at a very high

When the APBCLK frequency is high, I2C can still work at a reduced frequency; or vice versa, the CPU works at a lower frequency.

It does not affect I2C data communication at a higher baud rate.

Theoretically, there is no constraint on the relative relationship between PCLK and the baud rate clock, and the baud rate clock can be faster or slower than PCLK.

The application needs to pay attention to whether the CPU or DMA has enough time to transfer data when the frequency difference between the two is large.

19.5 Interface Timing

19.5.1 Interface Timing Diagram

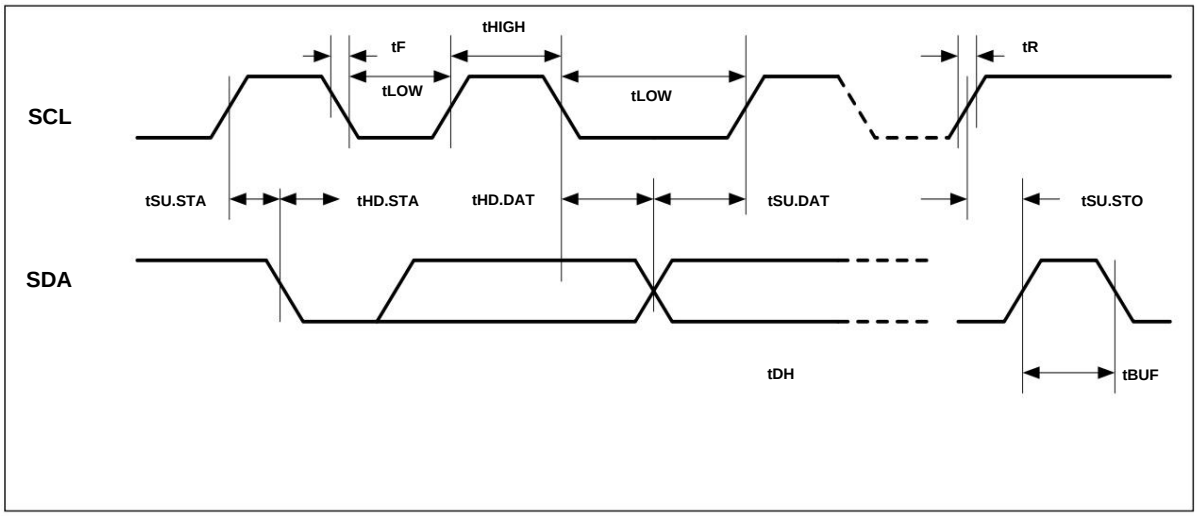

Figure **19-2 I² C-** bus Timing

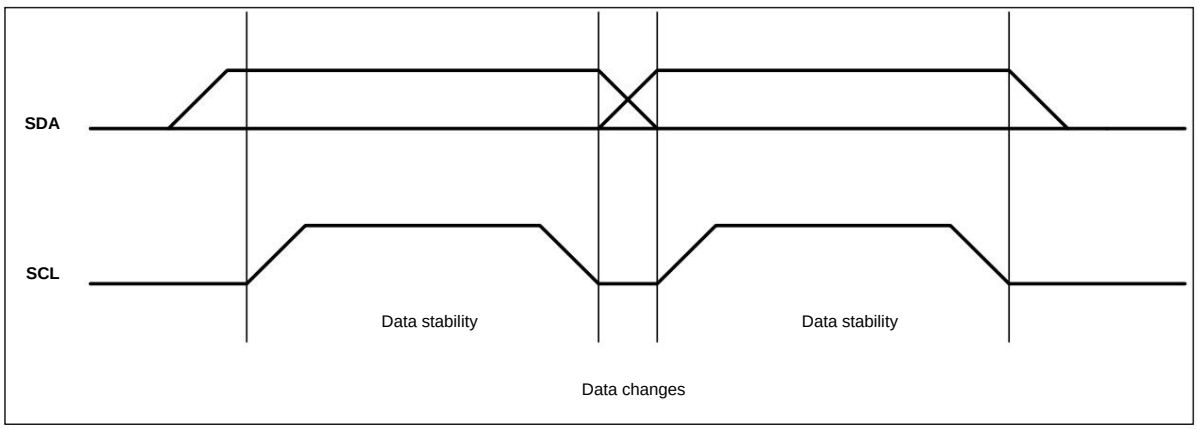

Figure **19-3** Data Validity Timing

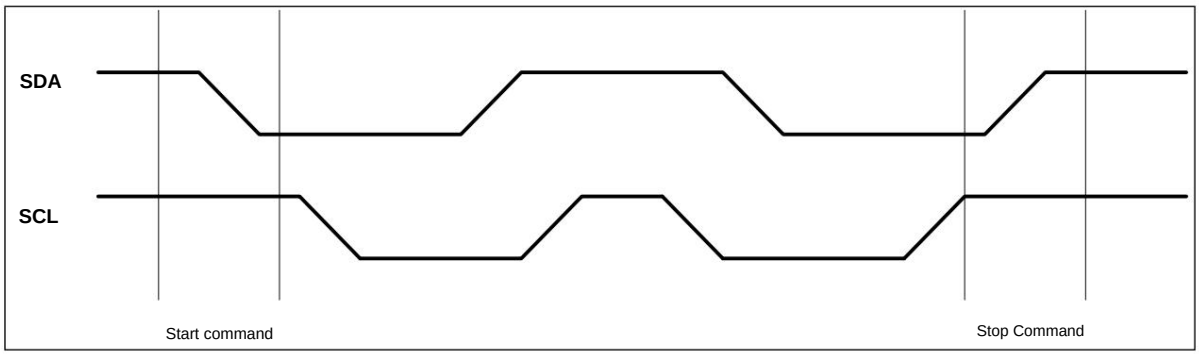

Figure **19-4** Start and Stop command definitions

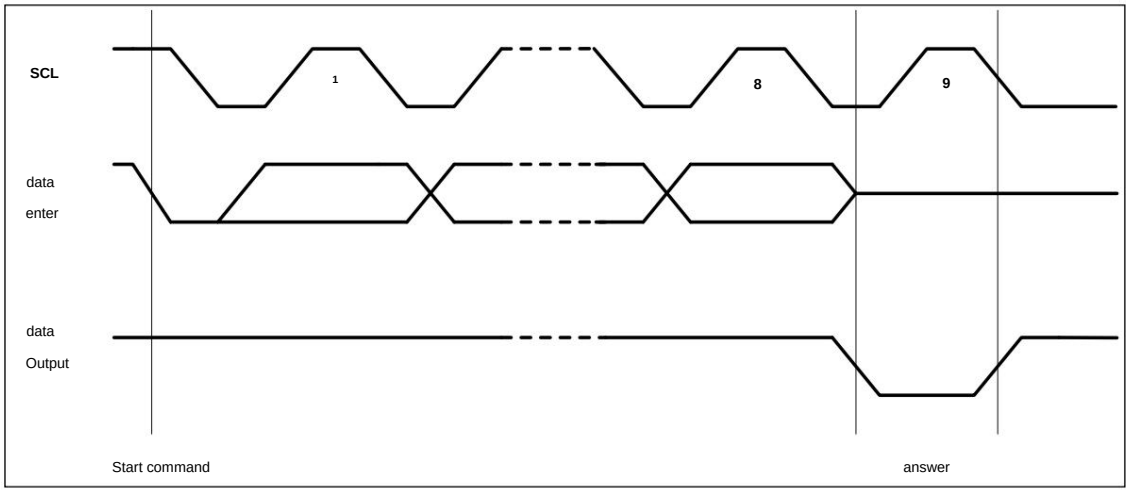

Figure **19-5 Output Acknowledgement (ACK)**

19.5.2 Interface Timing Description

Clock valid timing: The SDA pin is usually pulled high by the peripheral device. The data of the SDA pin should change when SCL is low (see Figure 19-3);

When data changes while SCL is high, it is interpreted as a start or stop command as described below.

Start command: When SCL is high, the change of SDA from high to low is regarded as the start command. The start command must be used as the start command for any read/

Start of a write operation command (see Figure 19-4).

Stop command: When SCL is high, the change of SDA from low to high is regarded as a stop command. After a read operation, the stop command will

The EEPROM enters a wait state low power mode (see Figure 19-4).

Output response: The data on SDA is serially input and output in groups of 8 bits, with the MSB sent first. The receiver responds after receiving each word.

After the holiday, an acknowledge bit (hereinafter referred to as ack) should be sent back in the 9th cycle. The clock of ack is provided by the host.

The sender suspends SDA during the ack period, and the receiver must pull SDA low to ensure that SDA is low during the high level of the ack clock, forming a

A valid ack signal is sent (see Figure 19-5).

parameter	symbol	**Standard mode (100K)**		**Fast mode (400K)** Minimum		unit
		Maximum	Minimum	Maximum		
SCL clock	FSCL	0	100	0	400	kHz
frequency Start condition setup time		4.7	-	0.6	— us	
TSU:STA Start condition hold time		4.0	-	0.6	— us	
THD:STA clock low level time TLOW		4.7	-	1.3	— us	
clock high level time THIGH 4.0 Data input setup			-	0.6	— us	
time TSU:DAT 250			-	100(4)	— ns	
Data input hold time THD:DAT		5.0 0(2)	- 3.45(3)	- 0(2)	- 0.9(3)	us us
SDA and SCL rise time TR		-	1000	20+0.1Cb(5)	300	ns
SDA and SCL fall time TF stop		-	300	20+0.1Cb(5)	300	ns
condition setup time TSU: STO bus idle		4.0	-	0.6	— us	
time	TBUF	4.7	-	1.3	— us	

parameter	symbol	Standard mode (100K)		Fast mode (400K)		unit
		Minimum	Maximum	Minimum	Maximum	
Capacitive load of the bus	Cb —		400 —		400	Pf
Noise tolerance low	VnL	0.1V DD —	0.2V	0.1V DD	— In	
value Noise tolerance high value	VnH	DD —		0.2V DD	— In	

Table **19-4 I** ^{2} **C** interface timing requirements

19.6 I²C working mode

The I2C module supports the following operating modes:

ÿ Host receives

ÿ Host sends

ÿ Slave receiving

ÿ Slave sends

After the chip is powered on, the I2C module is disabled by default, and neither the master nor the slave works. The software needs to select the module working mode according to the application.

Set MSPEN to enable master communication, or set SSPEN to enable slave communication.

The master and slave cannot work at the same time because they reuse the same IO pins as SCL and SDA. In principle, software is prohibited from using MSPEN and SSPEN are set to 1.

19.7 I 2 C slave address format

The I2C bus protocol defines the following reserved addresses. For most of these reserved addresses, the I2C slave hardware does not make a legality check, but the software can Customized processing is performed based on the address received.

However, for 10-bit slave address applications, that is, when SSPCON.A10EN=1, the first byte must start with 11110.

The ADDR_ERROR error flag will be triggered. In the case of SSPCON.A10EN=0, if the slave receives a signal starting with 11110

The address byte will also set the ADDR_ERROR error flag.

Slave	**R/W_ bit** 0 1	Description
address 0000		General Call address
000 0000		START byte
0000 0000	X	CBUS address
001 0000 010	X	Reserved for different bus format
0000 011	X	Reserved for future purpose
0000 1XX	X	HS-mode master code
1111 1XX	X	Reserved for future purpose
1111 0XX	X	10bit slave addressing

Table **19-5 I2C** slave reserved address definition

19.8 I²C Initialization

The I2C module must be initialized correctly before I2C communication. It is recommended that the software be initialized according to the following steps:

ÿ Clear the I2CRST register of the RCC module to ensure that the I2C module is not in reset state

ÿ Set the I2C_APBEN register of the RCC module to enable the I2C module register bus interface clock

ÿ Configure the I2C_CKSEL and I2C_CKEN registers of the RCC module to select and enable the I2C working clock (if it is in slave mode,
This step is not required)

ÿ Configure analog filter enable as needed (SCL and SDA input analog filter, >50ns)

19.8.1 IO Configuration

FM33LC0XX has at most two groups of pins for data transmission. Before starting I2C communication, the FCR register of the corresponding pins needs to be set to OF:

SDAÿPA12/PD12

SCLÿPA11/PB15

Note that if PA11 and PB15 are configured as SDA at the same time, PA11 is connected to the I2C module and PB15 is invalid.

PA12 and PD12 are configured as SCL functions at the same time. In master mode, both pins will output SCL signals. In slave mode, only PA12 is connected to the SCL input of the I2C slave.

PA11 and PA12 are strong drive true OD pins and must be used with external bus pull-up resistors and have a 20mA sink current.

Ability to support Fm+ mode.

19.8.2 Host baud rate configuration

The I2C master needs to configure the communication baud rate before enabling, but the slave does not need to configure.

MSPBRG[8:0] baud rate configuration register is used to generate the communication baud rate. MSPBRG is a 9-bit baud rate division factor.

The calculation formula is as follows:

TSCLÿ2TBRG

TBRGÿ2×TI2CCLK×(MSPBRG[8:0]+1); TI2CCLK is the I2C working clock cycle, that is:

MSPBRG = FI2CCLK/(4*FSCL) – 1

For example, for a 100k baud rate, if the I2C working clock is 8M, then MSPBRG=19.

19.8.3 Input Analog Filtering and Output Delay of Slave Devices

The analog filter function is only for the SCL pin and can only be enabled on the SCLi input signal of the slave.

At the same time, the slave's SDA output delay is increased by adding an analog delay of more than 300ns to SDAo to ensure that SDA is relative to Output hold time for the falling edge of SCL.

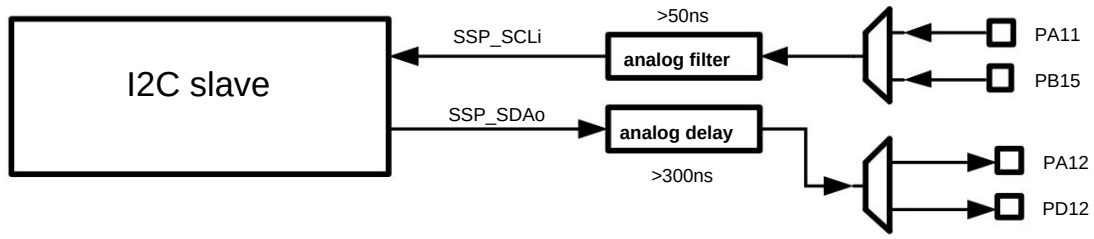

Figure **19-6** Slave signal filtering

19.9 I²C Host Function

The I2C master mode of FM33LC0XX does not support multi-master bus, so other devices on the bus are slaves.

The host provides the synchronous clock SCL, and the data flow direction of SDA can be the master sending and the slave receiving, or the slave sending and the master receiving.

I2C bus communication is always initiated by the host, and the host mode supports 7-bit or 10-bit addressing.

19.9.1 7-bit addressing

In 7-bit addressing, the first byte sent by the host contains the slave address and the transmission direction bit (R $\overline{W}$ /) .

A transfer is when the master writes data to the slave (R $\overline{W}$/ =0) or the master reads data from the slave (R $\overline{W}$/ =1).

name	Slave Address Byte							
Bit 7		6	5	4	3	2	1	0
Position Name	address							R/W

Description:

Position No.	Mnemonics	Functional Description
7-1	address	Slave device address
0	R/W	0: Write means sending data (sent by the master) 1: Read means requesting data (slave sends back)

The host writes data to the slave

The typical 7-bit addressing frame structure of the host writing data to the slave is shown in the figure below.

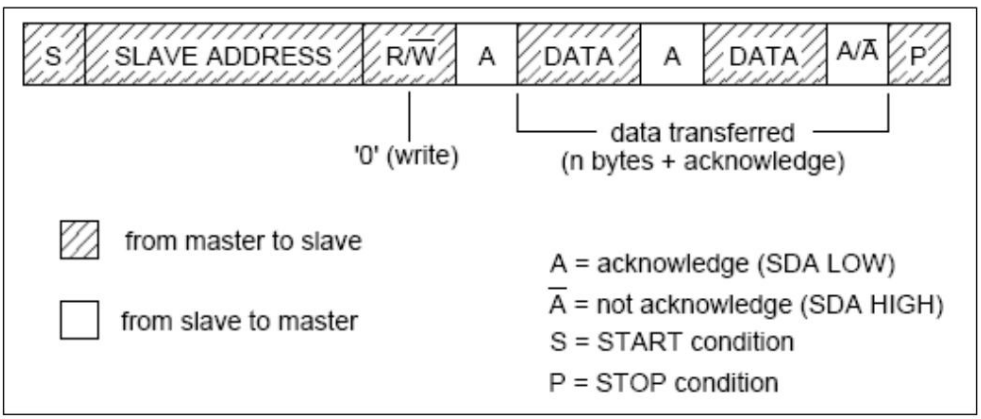

Figure **19-7** Frame format when the host writes data to the **7-** bit address slave

1. Host initiates START timing

2. The host sends the slave address. The slave address contains 7 bits of slave address and 1 bit of R/W flag. When sending data, the R/W bit is 0.

4. The host sends the first frame of 8-bit data

5. After sending 8 bits of data each time, the host will determine whether a valid ACK is detected at the 9th SCL. If the host detects

After ACK is successful, the next byte of data will be output.

6. If the slave fails to respond to ACK, the host should send a STOP sequence to terminate the transmission after detecting NACK.

7. After the host completes sending all data, it sends a STOP sequence

The operation flow of software starting I2C host sending is as follows:

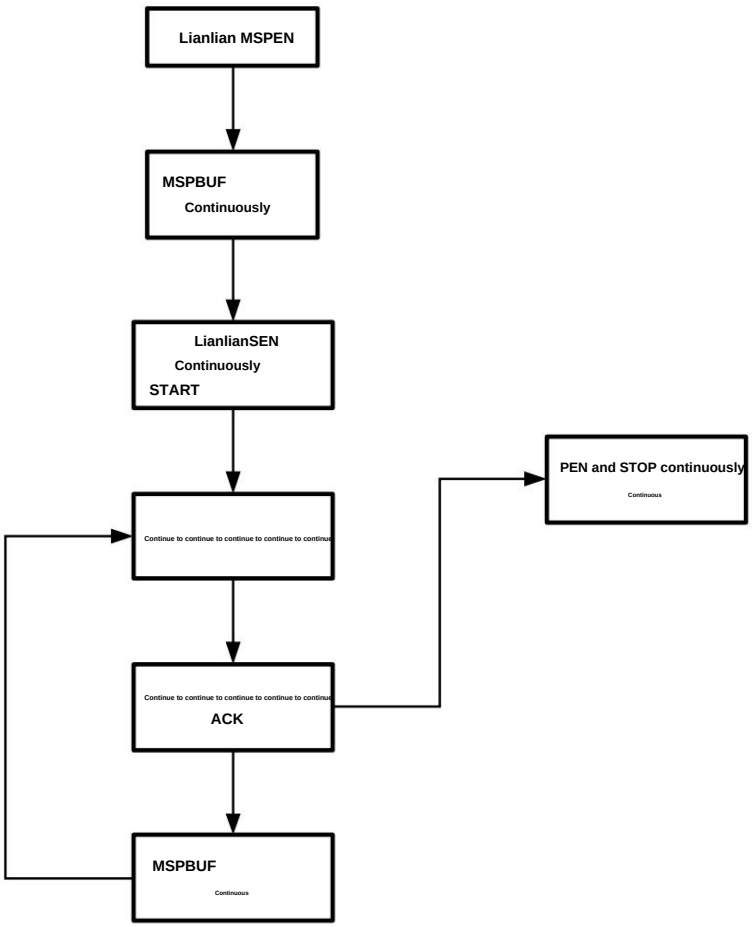

Figure **19-8 I2C** software sending data flow chart

The waveform diagram of the I2C host writing data to the 7-bit address slave is as follows:

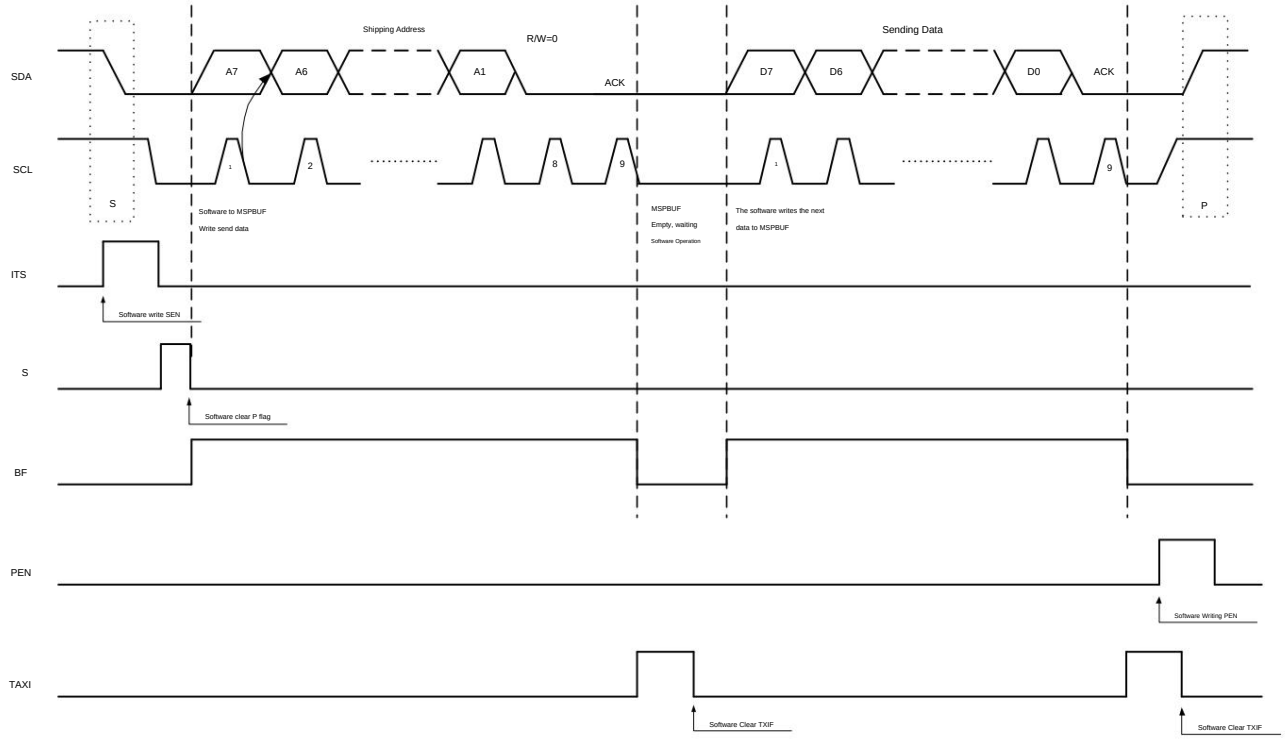

Figure **19-9 I2C** master sends data flow to **7-** bit address slave

The master reads data from the slave

The typical 7-bit addressing frame format for the host to read data from the slave is shown in the figure below.

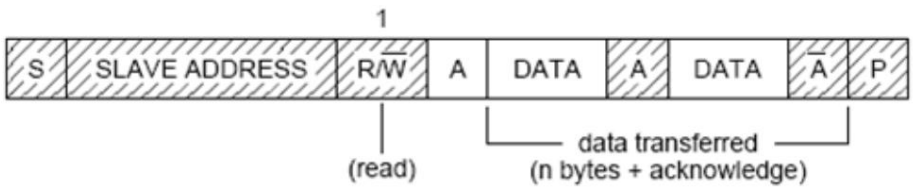

Figure **19-10** Frame format when the host reads data from a **7-** bit address slave

1. Host initiates START timing

2. The host sends the slave address. The slave address contains 7 bits of slave address and 1 bit of R/W flag. When reading data, the R/W bit is 1.

3. At this time, set MSPCON.RCEN to 1, and the host automatically switches to the receiving state.

4. The host starts to receive the first 8-bit data byte and sends a valid ACK to the slave at the 9th SCL, thus continuing to read the next word.

8-bit data

5. After the master reads the last byte, it sends a NACK to the slave at the 9th SCL.

6. The host sends a STOP sequence to terminate the reading

The operation flow of software starting I2C receiving is as follows:

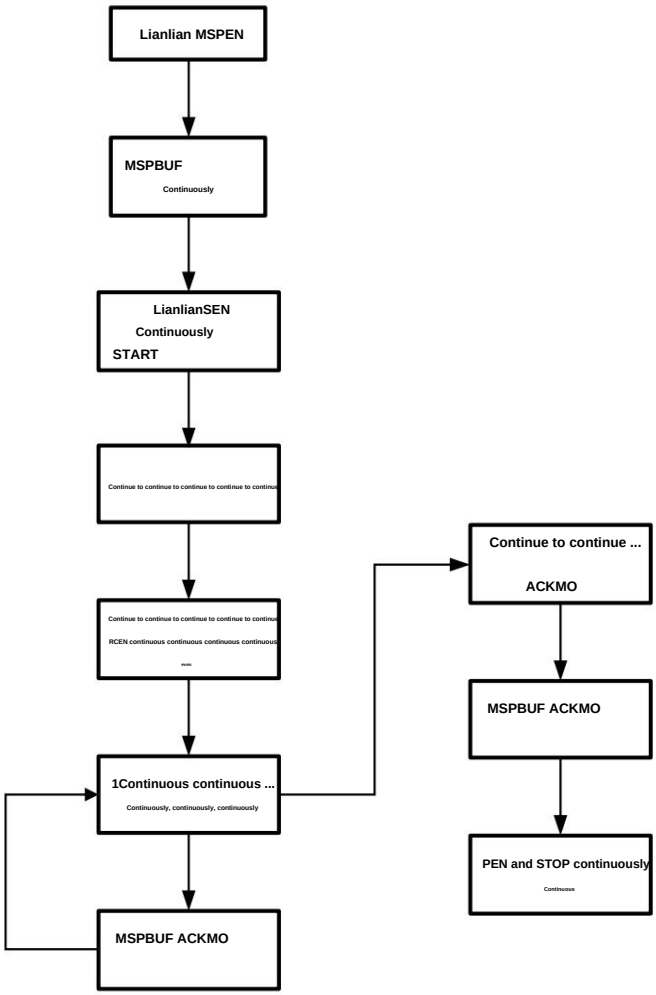

Figure **19-11I2C** software sending data flow chart

After the host receives the data sent by the slave, it sends a response according to the ACKMO register. The reset value of ACKMO is 0, which is the default state.

If the software wants the host to send NACK after receiving the data, it needs to send NACK after the previous byte is received.

The ACKMO register is rewritten to 1 during the interrupt. When ACKMO is 1, the host will automatically clear ACKMO after sending the response.

The waveform diagram of the I2C host reading data from the 7-bit address slave is as follows:

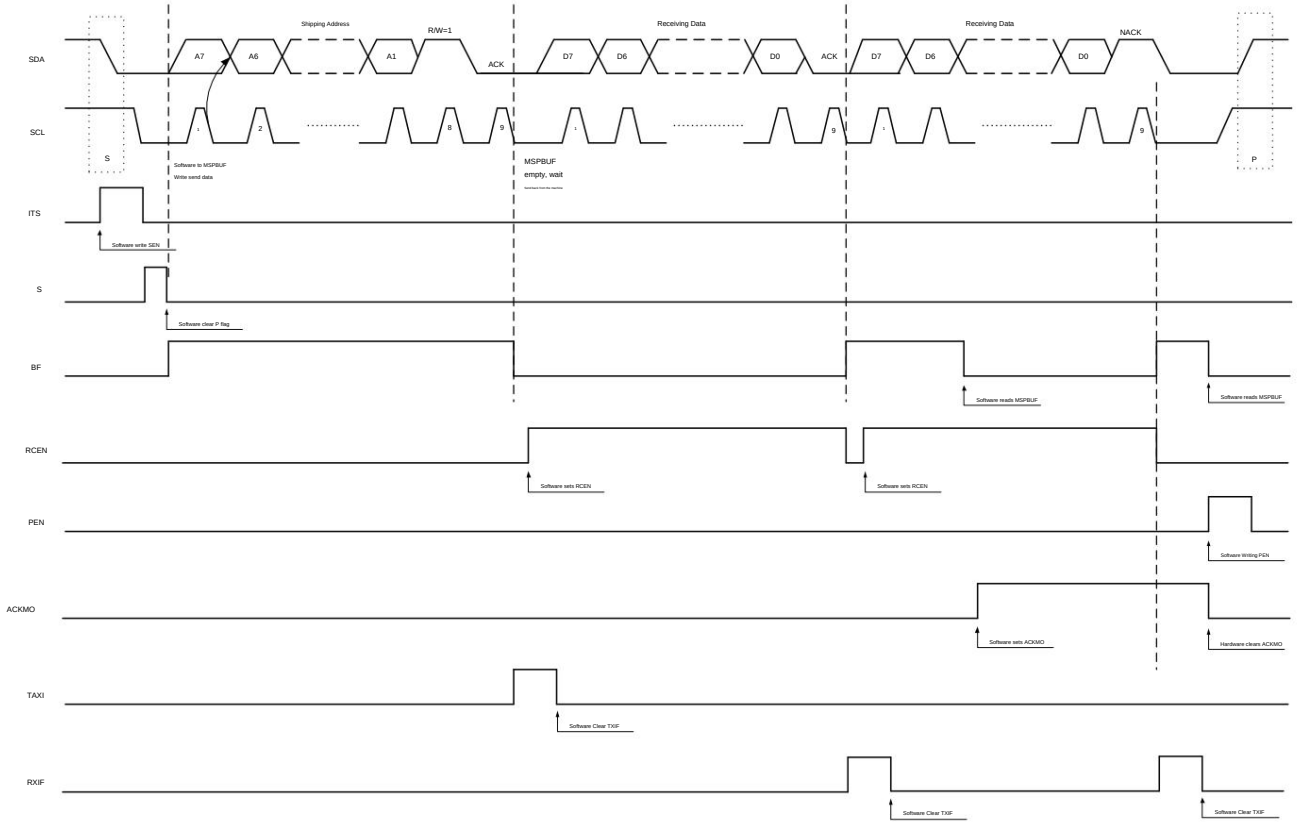

Figure **19-12I2C** reads data flow from a **7-** bit address slave

Bidirectional data transmission (combined mode)

The typical bidirectional data read and write flow chart is shown in the figure below. When the host sends or reads data, the host can send a Repeated Start sequence to restart a new send or read communication, so the host can send data or

There is data to read.

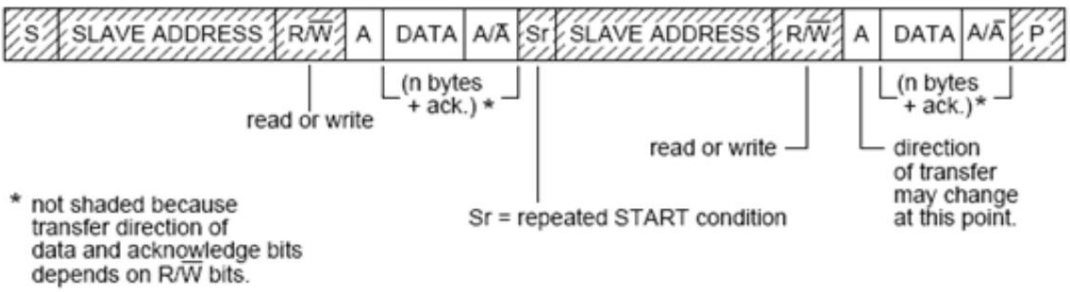

Figure **19-13** Two-way data communication frame format

The software operation flow of combined transmission is similar to that of unidirectional transmission, except that after a byte is sent and received, the ReSTART sequence and

The slave address byte modifies the transfer direction.

19.9.2 10-bit addressing

In 10-bit addressing, the first byte sent by the master contains part of the slave address (11110_A9_A8) and the transfer direction bit (R W/).

The second byte contains the remaining slave address (A7~A0). After the two bytes of address are sent, data transmission can be performed.

The host writes data to the slave

The typical 10-bit addressing data flow diagram of the host writing data to the slave is shown in the figure below.

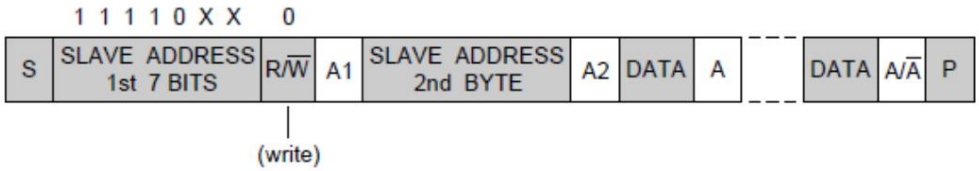

Figure **19-14 10-bit** addressing, the master writes data to the slave

1. Host initiates START timing

2. The host sends the first slave address byte, starting with 11110, followed by the 2-bit highest bit of the slave address and the R/W flag.

When sending data, the R/W bit is 0

3. The host checks the ACK sent back by the slave

4. The host sends the second slave address byte, which contains the lower 8 bits of the slave address.

5. The host checks the ACK sent back by the slave

6. The host continues to write data to the slave

7. After the host completes sending all data, it sends a STOP sequence

The operation flow of software starting I2C host sending is as follows:

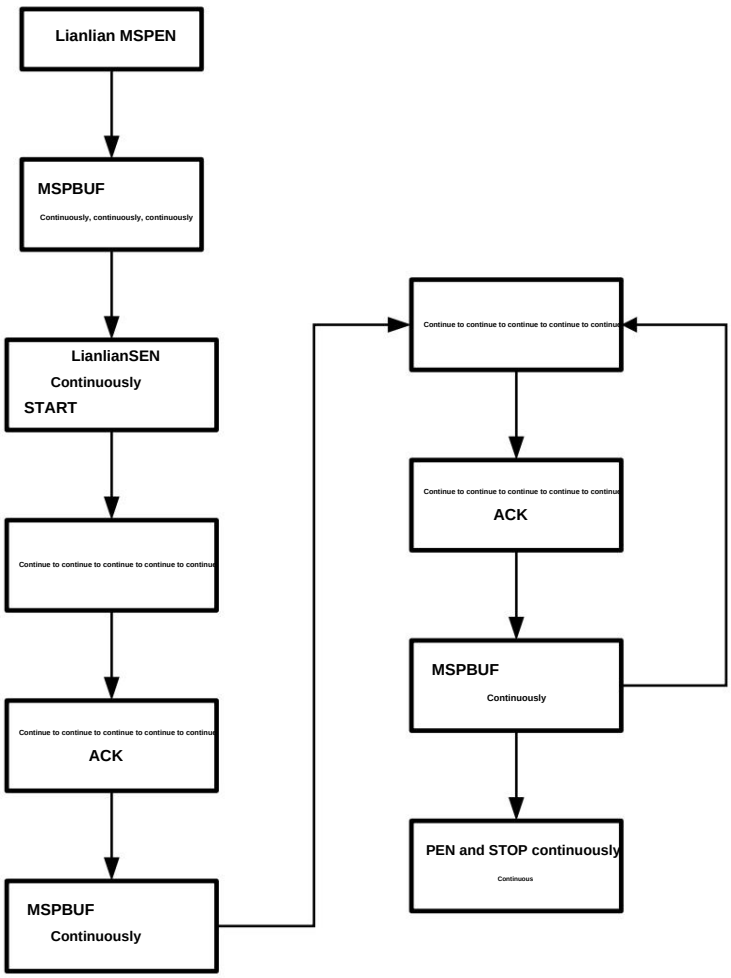

Figure **19-15 I2C** software sending data flow chart

The master reads data from the slave

The typical 10-bit addressing, the data flow diagram of the host reading data from the slave is shown in the figure below.

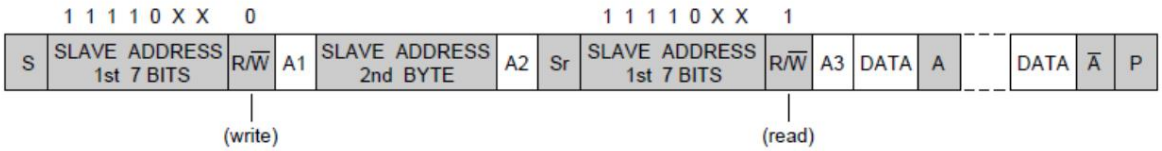

Figure **19-16 10-bit** addressing, the master reads data from the slave

1. Host initiates START timing

2. The host sends the first byte of the slave address, which includes a 5-bit leading code 11110, a 2-bit slave address highest bit, and a 1-bit R/W flag.

When reading data, the R/W bit is 1

3. The host sends the second byte of the slave address, including the lower 8 bits of the address

4. Host sends ReSTART timing

5. The host sends the first byte of the slave address again and changes R/W to 0

6. At this time, set MSPCON.RCEN to 1, and the host switches to the receiving state.

7. The host starts to receive the first 8-bit data byte and sends a valid ACK to the slave at the 9th SCL, thus continuing to read the next byte.

Byte 8 bits of data

8. After the master reads the last byte, it sends a NACK to the slave at the 9th SCL.

9. The host sends a STOP sequence to terminate the reading

The operation flow of software starting I2C receiving is as follows:

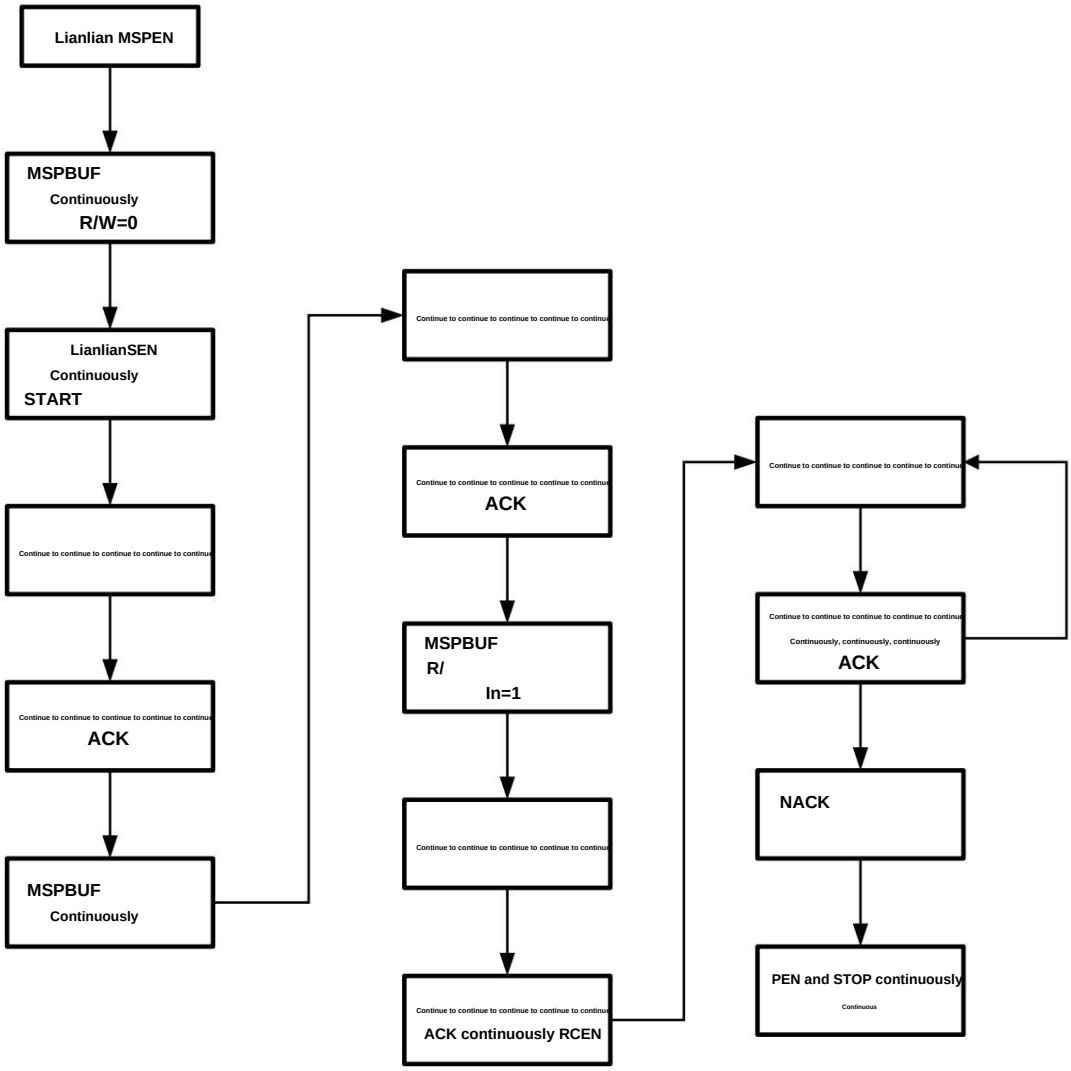

Figure **19-17 I2C** software sending data flow chart

Bidirectional data transmission (combined mode)

The typical bidirectional data read and write flow chart is shown in the figure below. When the host sends or reads data, the host can send a Repeated Start sequence to restart a new send or read communication, so the host can send data or

There is data to read.

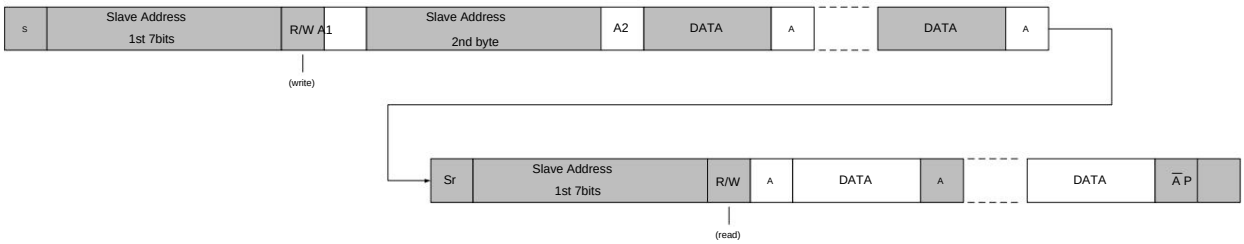

Figure **19-18 I2C** software sending data flow chart

The software operation flow of combined transmission is similar to that of unidirectional transmission, except that after a byte is sent and received, the ReSTART sequence and

The 1st slave address byte modifies the transfer direction.

19.9.3 DMA

The I2C host supports DMA. It should be noted that this can only be done when the bus clock (APBCLK) of the I2C module is enabled.

Use the DMA function.

The host uses DMA to write data to the slave

When the host uses DMA to send data, all data including the slave address byte and the send data need to be written into RAM in advance.

And send it out through DMA request. The software should configure the target DMA channel as I2C_TX in advance.

When DMAEN=1, MSPEN is set, and if the data buffer register MSPBUF is empty, the I2C module will generate DMA

Request, DMA module responds to the request and writes the waiting data in RAM into MSPBUF. At the same time, I2C module automatically sets SEN to generate

Generate START timing, start data transmission (the first byte is the slave address). In DMA transmission mode, I2C does not check the transmission

Data legitimacy, the software must ensure that the data in RAM is correct.

After each byte is sent, I2C checks the slave ACK. If the ACK is correct, a new DMA request is generated. If a NACK is received,

A NACK interrupt is generated and no DMA request is generated.

When DMA completes sending data of the specified length, a DMA transfer completion interrupt is generated. At this time, the software can set PEN to generate a STOP

The I2C hardware can also automatically set PEN to generate STOP timing according to the DMA transfer completion signal.

The desired strategy can be selected by using the AUTOEND register.

The process of the host using DMA to send is as follows:

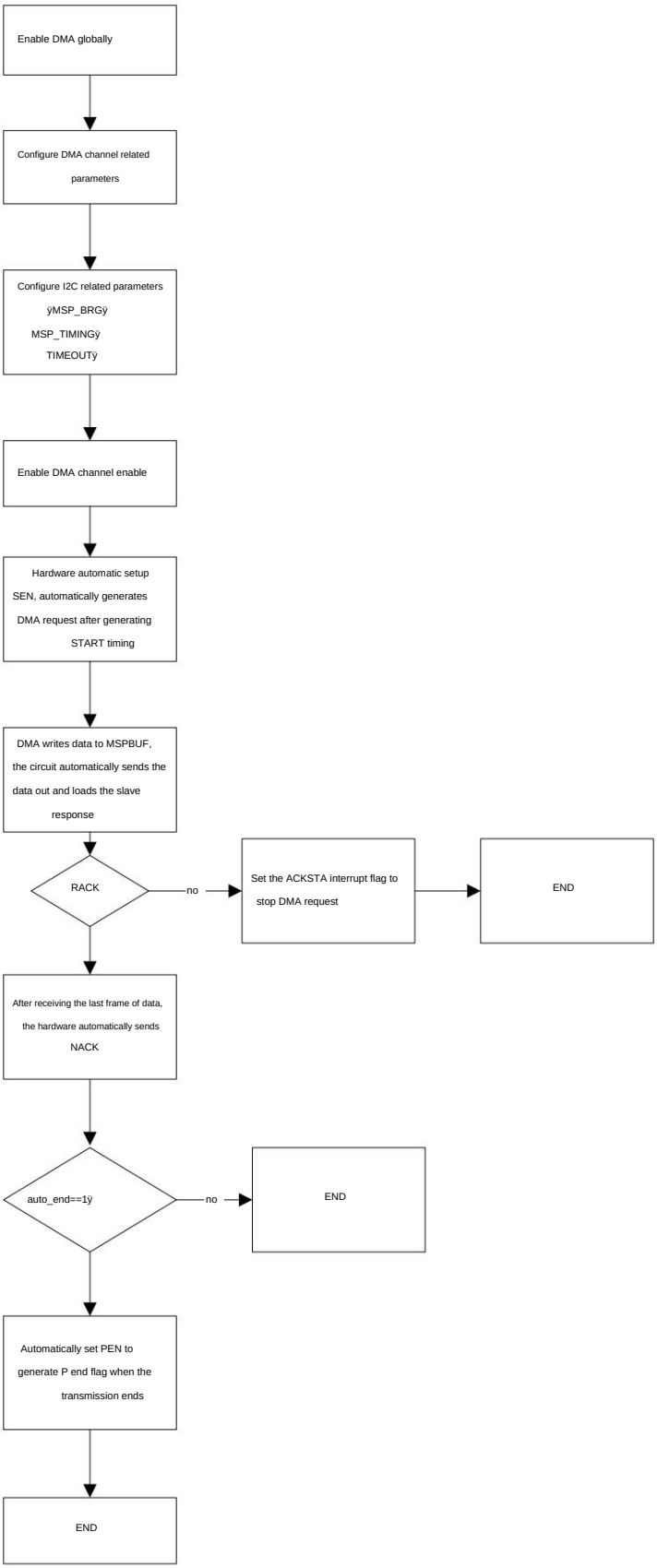

Figure **19-19 I2C** host **DMA** sending flow chart

The host uses DMA to read data from the slave

In this scenario, the slave address byte must be sent by software. The software should configure the target DMA channel as I2C_RX in advance.

After the software sends the slave address, it sets MSP_DMAEN=1, and then enables the corresponding DMA channel. The I2C automatically enters the receiving state.

After each byte is received, it generates a DMA request to notify the DMA to read the MSPBUF content and send a DMA request to the slave.

Send back ACK.

When the DMA transfer reaches the specified length, the DMA transfer completion flag will notify I2C to send back NACK.

Register configuration, can be set by software or hardware PEN to generate STOP timing.

Note: When *I2C* Host through *DMA* When receiving data, *AUTO END* Configuration and same*DMA* Transfer length ÿ *CHxTSIZE*) configuration, *DMA* The number of bytes received will vary.*AUTOEND=0* When the number of bytes received is*CHxTSIZE+1* ÿ when*AUTOEND=1* When the number of bytes received is*CHxTSIZE* ÿ

The process of the host using DMA to receive is as follows:

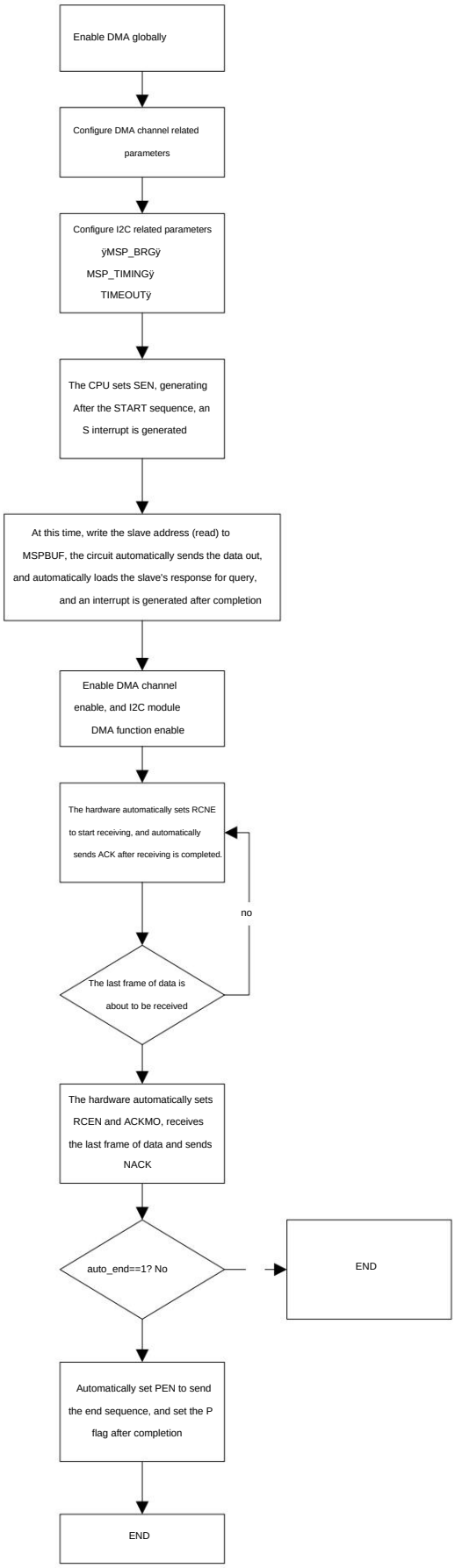

Figure **19-20 I2C** host **DMA** receiving flow chart

19.9.4 Slave Clock Stretching

The I2C bus runs at a low speed, and the slave notifies the master to suspend data communication by pulling down SCL. The I2C master must support this feature.

Therefore, at the start of each byte transmission and reception, the host needs to automatically check the actual SCL on the bus after trying to send SCL high level.

If it is not a high level, it means that the slave is extending SCL. The host will continue to monitor the SCL level until SCL

To high, start the subsequent operation.

Note: The host only receives the first byte of each byte. ***SCL*** On the rising edge ***SCL*** Extended inspection.

19.9.5 Timeout Mechanism

The I2C master also implements a timeout mechanism, that is, if the slave pulls down SCL for a long time and the bus cannot communicate, a timeout alarm is generated.

Interrupt and return to IDLE state.

When the host detects an SCL extension, its internal timer starts timing. The maximum SCL extension timeout period set by the host is 4096 SCL

Period, assuming the baud rate is 100K, the timeout period is approximately 40ms, if the baud rate is 400K, the timeout period is approximately 10ms.

The software can set the timeout period through the 12-bit TIMEOUT register. The software must set it when MSPEN is 0.

TIMEOUT register, the reset value of this register is 0xFFF, which means the longest timeout period is 4096*TSCL.

After that, the TIMEOUT register starts to decrement downward. When the count reaches 0, the count stops and the TIMEOUT register is reset to 0xFFF.

At the same time, the timeout interrupt is triggered. Therefore, by modifying the initial value of TIMEOUT, the timeout period can be set.

TSCL_STRETCHING_TIMEOUT=TIMEOUT[11:0] * TSCL

When a TIMEOUT interrupt occurs, it is recommended that software reset the I2C module.

This function can be turned off. If the hardware timeout is turned off, the software can also implement it by combining the timer with the SCL pin status judgment.

Timeouts of arbitrary length.

19.9.6 Programmable Timing

The master mode of the I2C module provides flexible timing programming features, allowing users to define the low level width, high level width, and

Degree, SDA data setup and hold time.

The low and high widths of SCL can be set through the MSPBRG register, and the SDA can be configured through the SDAHD register.

The length of the data hold and setup/hold time relative to the SCL clock pulse.

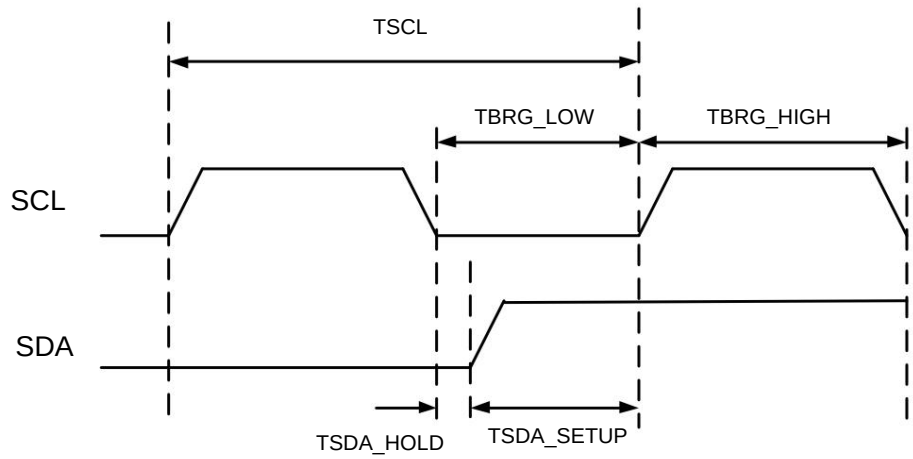

Figure **19-21** Host timing control

In the figure above, TSCL is the communication baud rate, and the various parameters can be expressed by the following formula:

TSCL = TBRG_LOW+ TBRG_HIGH

TSDA_SETUP = TBRG_LOW– TSDA_HOLD

Note that in the application *MSPBGRH* 、 *MSPBRGL* and *SDAHD* The register configuration must meet the following requirements. If these are violated

Requesting bus timing that will cause an exception:

MSPBRGH>= 2

MSPBRGL >=2

MSPBRGL-1>= SDAHD >=1

TIMEOUT >= 1

19.10 I2C Slave Function

The I2C slave does not require a system clock to operate, so data can be sent, received, and woken up when the chip is in sleep mode.

After receiving 1 byte of data, the slave generates an interrupt to notify the CPU to process the data. Before the CPU takes the data, the hardware can pull the SCL Low (software control enabled), notifying the sender that it is busy and that the sender should suspend sending until SCL is released. If the receiver cannot respond to ACK, After the sender fails to detect ACK, it should send P to terminate the communication or send Sr to start a new communication.

After the slave sends 1 byte of data, it generates an interrupt to notify the CPU. The hardware pulls down SCL to make the host wait. The CPU responds to the interrupt and prepares After receiving the next byte of data, the master releases SCL, and the slave continues to send SCL to allow the slave to continue sending data.

19.10.1 Slave Addressing

Depending on the state of the SSPCON.A10EN register, the slave can support 7-bit or 10-bit addressing. The slave address is determined by SLAVE_ADDR register definition.

For 10-bit slave address applications, that is, when SSPCON.A10EN=1, the first byte must start with 11110. The ADDR_ERROR error flag will be triggered. In the case of SSPCON.A10EN=0, if the slave receives 11110 The first address byte will also set the ADDR_ERROR error flag.

19.10.2 Slave sends data

Recommended operating procedures:

ÿ The slave receives the address byte (R/W=1), sends back ACK, and generates an address match interrupt

ÿ Since R/W=1, the hardware automatically extends SCL and the slave enters the sending state

ÿ The software responds to the interrupt, queries the R/W flag, and confirms that it is sent by the slave

ÿ The software writes the data to be sent into SSPBUF

ÿ Hardware automatically releases SCL

ÿ When a new SCL arrives, SSPBUF is shifted and output to the SDA bus

ÿ Receive ACK and generate a send completion interrupt

ÿ Repeat the data transmission process until the STOP timing is received or the host NACK is received

The following figure is a typical waveform diagram of slave data transmission:

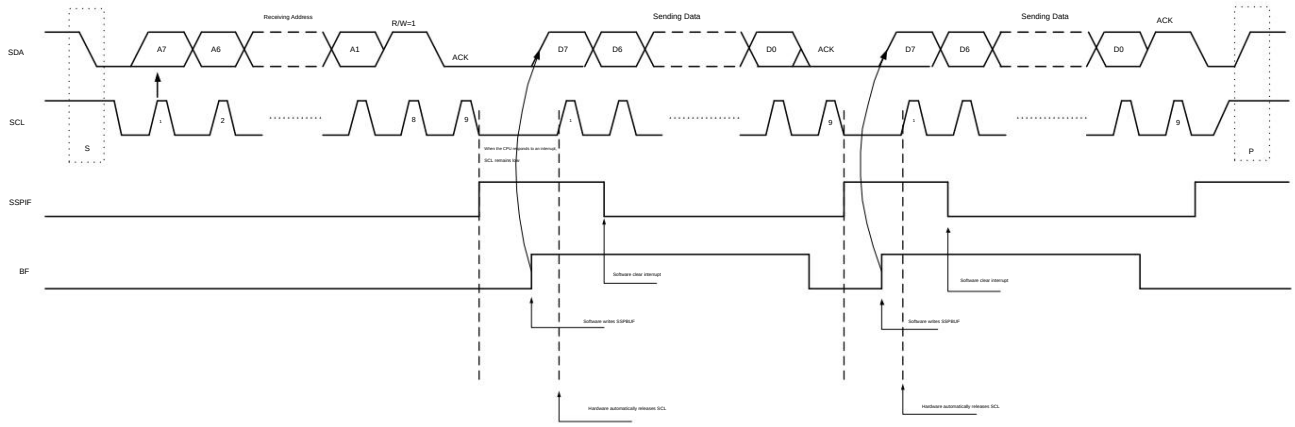

Figure **19-22** Slave data transmission waveform

In the slave transmission process, when the slave receives the correct address, the ADM flag is set and the address byte will not be written to SSPBUF.

The BF flag will not be set. The hardware automatically pulls down the SCL signal and waits for the software to write to SSPBUF. When the software writes to SSPBUF, BF is set.

When the hardware releases SCL.

19.10.3 Slave Receiving Data

Recommended operating procedures:

ÿ The slave receives the address byte (R/W=0), sends back ACK, and generates an address match interrupt

ÿ Since R/W=0, the hardware automatically extends SCL and the slave remains in the receiving state

ÿ The software responds to the interrupt, queries the R/W flag, and confirms that it is receiving from the slave

ÿ Software reads SSPBUF and hardware automatically releases SCL to start receiving data

ÿ The host data byte arrives, and the hardware sets the BF flag after the byte is received.

ÿ The slave sends back ACK and generates a receive completion interrupt

ÿ Hardware automatically extends SCL (SCLSEN=1)

ÿ The software responds to the interrupt, reads SSPBUF, and the hardware automatically clears the BF flag

ÿ Hardware automatically releases SCL

ÿ Repeat the data receiving process until the STOP timing is received, or the software sets ACKEN to 0

The following figure is a typical waveform diagram of slave data reception (SCLSEN=1):

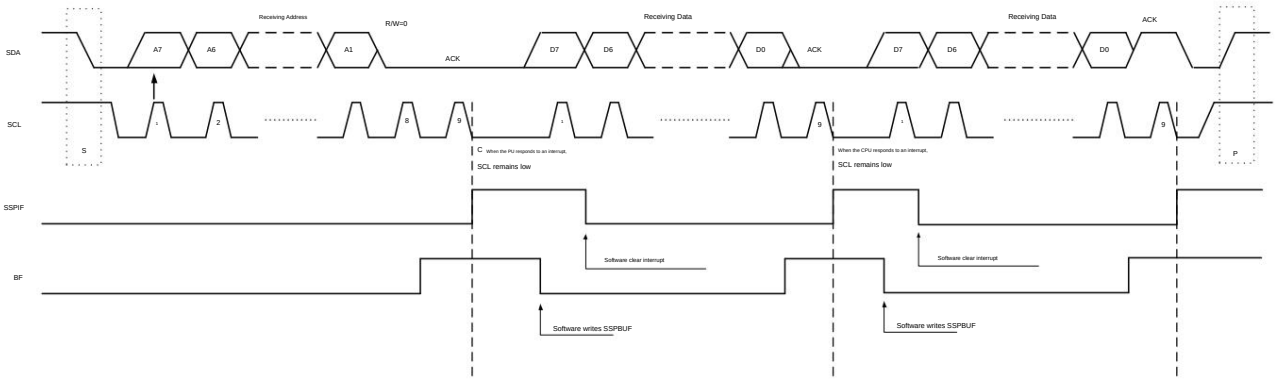

Figure 19-23 Slave data receiving waveform

During the slave receiving process, the slave first receives the address byte. If the address matches, the ADM flag is set and the address byte will be written SSPBUF and set the BF flag, then the hardware pulls SCL low. When the software reads SSPBUF, the BF flag is automatically cleared and the hardware releases SCL, subsequent data reception can be performed.

Note: The address byte will be written into the slave receiving process*SSPBUF* and leads to*BF* Set, software needs to read *SSPBUF* To clear *BF* and release*SCL* The address byte will not be written into the slave sending process*SSPBUF* Therefore, it will not be set*BF* Logo.

The slave device can passively or actively end the communication by receiving data.

If the host actively sends STOP, the slave passively ends the communication. Alternatively, the software registers ACKEN in the interrupt handler.

If the slave device is cleared, it will send back NACK after receiving the next byte. After receiving NACK, the host device will send STOP to end the communication. letter.

Slave SCL extension

I2C slaves enable SCL stretching by default, but software can disable this feature (SCLSEN register

to accommodate hosts that do not support slave SCL stretching.

When SCL extension is enabled, after data reception is completed, the software can only clear the receive buffer when reading the receive buffer during the SCL extension period.

If data overflow occurs during reception, the SSPOV flag is set, the hardware sends back a NACK, and SCL is no longer used.

When SSPOV is set, it is recommended that the software wait for the STOP flag to be set before reading the receive buffer.

The BF flag of the buffer is cleared.

Receive data overflow

When the slave receive buffer is full (BF=1), if new data is received, a receive overflow occurs and the SSPOV flag is set.

The old data in the receive buffer will be overwritten by the new data. This can only happen if the slave has disabled the SCL extension function.

Receive data overflow.

The following figure is a schematic diagram of data reception overflow when SCLSEN=0:

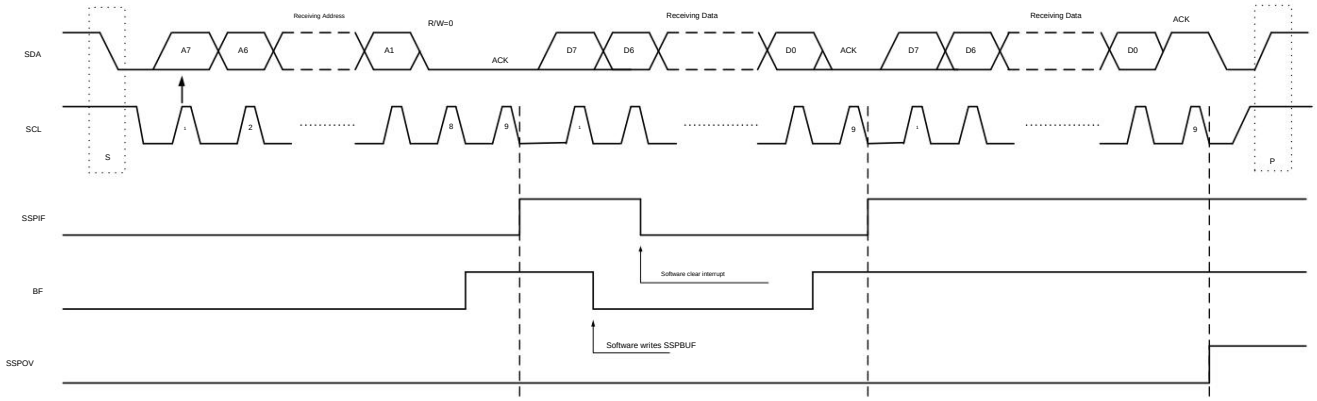

Figure 19-24 Slave data receiving waveform (SCLSEN=0, receiving overflow)

19.10.4 Slave Low Power Receive Wake-up

Since the I2C slave does not require a system clock to operate, it can receive data and wake up the MCU in sleep mode.

The I2C slave supports START timing wake-up, address match wake-up, and data reception completion wake-up.

Software setup process:

ÿ Turn off I2C master

ÿ Set slave address

ÿ Depending on the desired wake-up event, set the SE, ADME or BFE interrupt enable

ÿ Set the corresponding GPIO to I2C function

ÿ Set SSPEN to start the I2C slave

ÿ Enter sleep mode and wait for data reception

ÿ When a wake-up event occurs, the software queries the wake-up source and processes I2C data transmission

19.10.5 DMA

I2C slave supports DMA operation. It should be noted that this can only be done when the I2C bus clock (APBCLK) is enabled.

The bus clock is used to generate DMA requests and receive DMA responses.

The slave receives data using DMA

When the I2C slave receives the correct address, it generates an ADM interrupt flag. After the software responds to the interrupt, it queries the received R/W bit.

If it is 0, it means the host is ready to write data to the slave. At this time, the software can configure a specific DMA channel as I2C_RX and enable I2C

DMAEN of the slave; subsequently, each time the slave completes receiving a byte, it will generate a DMA request to notify DMA to read SSPBUF.

There are two possibilities to end DMA slave reception:

1) The data transfer length has not reached the DMA length configuration, the host sends a STOP timing, the software should respond to the STOP interrupt and Be proactive in handling the situation;

2) The data transfer length reaches the DMA length configuration, but since the DMA request is generated after the slave sends back an ACK, the software The slave should respond to the DMA transfer completion interrupt and clear ACKEN, so that the slave will send back NACK after receiving the next byte. End this communication.

The process of slave receiving using DMA is as follows:

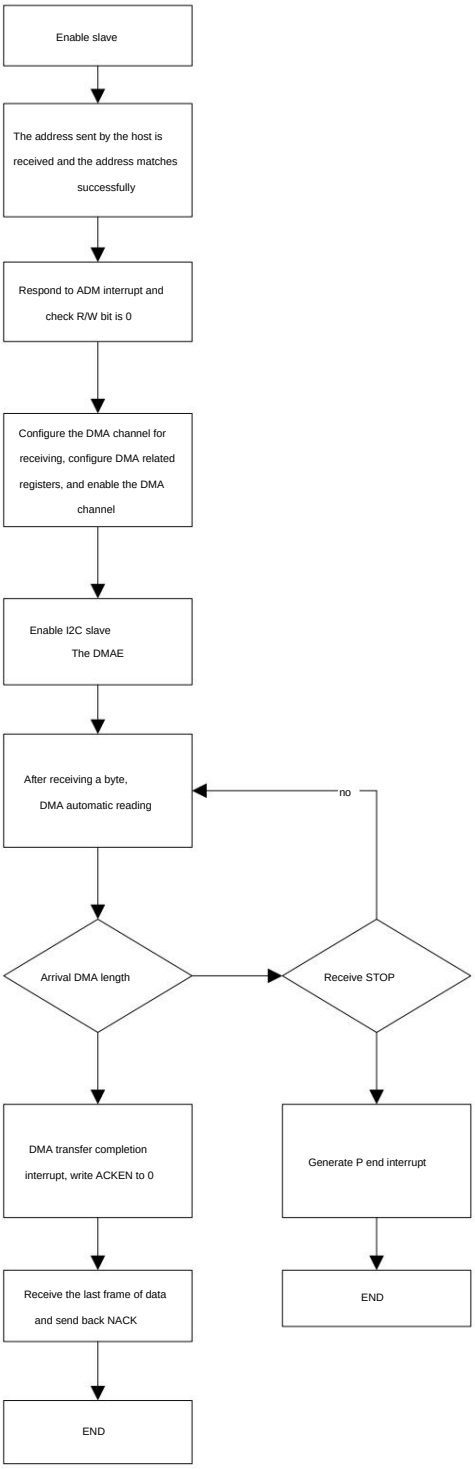

Figure **19-25 I2C** slave **DMA** receiving flow chart

The slave uses DMA to send data

When the I2C slave receives the correct address, it generates an ADM interrupt flag. After the software responds to the interrupt, it queries the received R/W bit.

If it is 1, it means the host is ready to read data from the slave. At this time, the software needs to read SSPBUF first to clear the BF flag, and then configure the special

Set the DMA channel to I2C_TX and enable the DMAEN of the I2C slave; then when the slave data buffer SSPBUF is empty,

Generates a DMA request, telling the DMA to write to the SSPBUF.

The read operation can only be terminated if the host sends back a NACK.

No longer responding to I2C requests, the slave will keep pulling SCL low until the software turns off the I2C slave module.

The process of the slave using DMA to send is as follows:

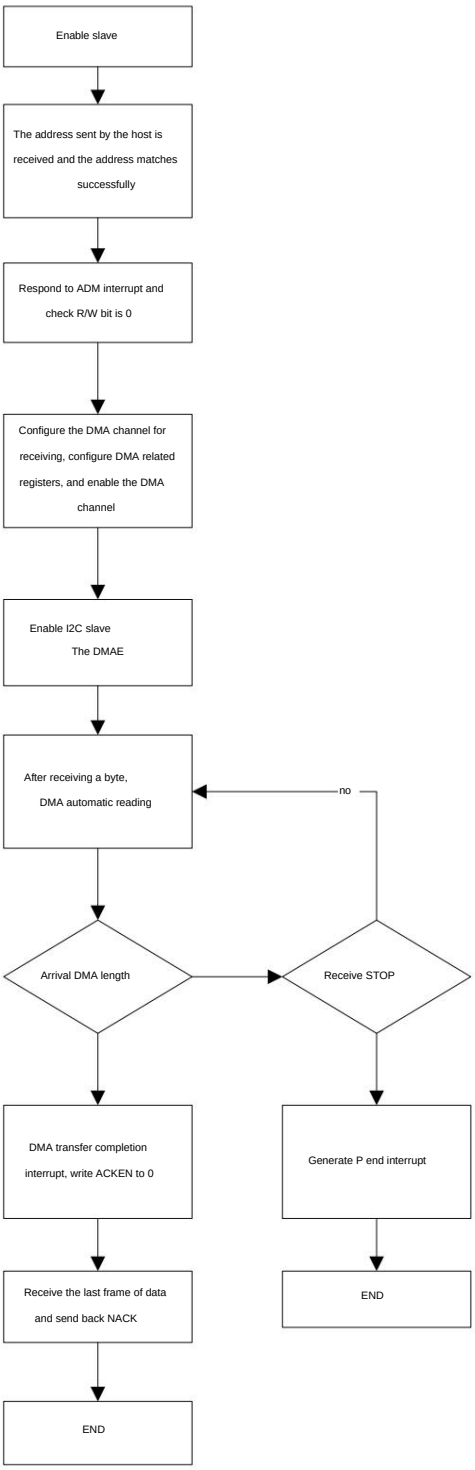

Figure **19-26 I2C** slave **DMA** sending flow chart

19.10.6 Slave Timing

Since the slave only uses SCL for data transmission and reception, some analog delay is required to implement the data setup and hold time of SDA.

The timing of SCL is completely controlled by the host.

The slave timing control is shown in the figure below. According to the I2C protocol requirements, the minimum data hold time of SDA relative to the falling edge of SCL is 0ns.

That is, the slave can meet the requirement by sending data using the falling edge of SCL. However, considering the actual falling time of the SCL waveform on the bus,

To better cover the hold time requirement, an additional RC delay of more than 300ns is added to the SDA output. This delay only needs to be applied

On the SDA output of the I2C slave (SSP_SDAO)

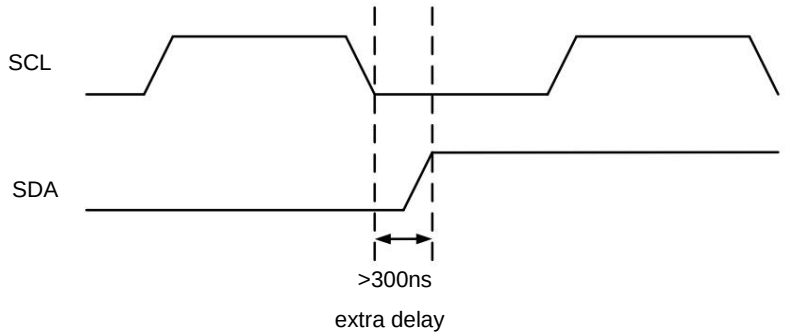

Figure **19-27 SDA** output delay waveform

19.11 Registers

offset address	name	symbol
I2C (module start address: 0x40012400)		
0x00000000	I2C Master Configuration Registers ÿI2C Master Config RegisterÿÿI2C Master Config Registerÿ	I2C_MSPCFGR
0x00000004	I2C Master Control Registers ÿI2C Master Control RegisterÿÿI2C Master Control Registerÿ	I2C_MSPCR
0x00000008	I2C Master Interrupt Enable Register ÿI2C Master Intterupt Enable RegisterÿÿI2C Master Intterupt Enable Registerÿ	I2C_MSPIER
0x0000000C	I2C Master Interrupt Flag Register ÿI2C Master Interrupt Status RegisterÿÿI2C Master Interrupt Status Registerÿ	I2C_MSPISR
0x00000010	I2C Master Status Register ÿI2C Master Status RegisterÿÿI2C Master Status Registerÿ	I2C_MSPSR
0x00000014	I2C Master Baud Rate Register ÿI2C Master Baud rate Generator RegisterÿÿI2C Master Baud rate Generator Registerÿ	I2C_MSPBGR
0x00000018	I2C Master Transceiver Buffer Register ÿI2C Master transfer BufferÿÿI2C Master transfer Bufferÿ	I2C_MSPBUF
0x0000001C	I2C Master Timing Control Register ÿI2C Master Timing Control RegisterÿÿI2C Master Timing Control Registerÿ	I2C_MSPTCR
0x00000020	I2C Master Timeout Register ÿI2C Master Time-Out RegisterÿÿI2C Master Time-Out Registerÿ	I2C_MSPTOR
0x00000024	I2C Slave Control Register ÿI2C Slave Control RegisterÿÿI2C Slave Control Registerÿ	I2C_SSPCR
0x00000028	I2C Slave Interrupt Enable Register ÿI2C Slave Interrupt Enable RegisterÿÿI2C Slave Interrupt Enable Registerÿ	I2C_SSPIER
0x0000002C	I2C Slave Interrupt Flag Register ÿI2C Slave Interrupt Status RegisterÿÿI2C Slave Interrupt Status Registerÿ	I2C_SSPISR
0x00000030	I2C Slave Status Register ÿI2C Slave Status RegisterÿÿI2C Slave Status Registerÿ	I2C_SSPSR
0x00000034	I2C Slave Transceiver Buffer Register ÿI2C Slave transfer BufferÿÿI2C Slave transfer Bufferÿ	I2C_SSPBUF
0x00000038	I2C Slave Address Register ÿI2C Slave Address RegisterÿÿI2C Slave Address Registerÿ	I2C_SSPADR

19.11.1 I2C Master Configuration Register (I2C_MSPCFGR)

name	I2C_MSPCFGR						
Offset	0x00000000						
Bit31 Bit Name Bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
	U-0						
Permission Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Position Name						CARS D	MSP_D STONE
Bit	U-0					R/W-0	R/W-0
permission bit Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Bit							
name Bit	U-0						
permission bit Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

Bit name		THEN MSPEN	
and permission	U-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:18		RFU: Unimplemented, read as 0
17	AUTO END	Host DMA automatic transfer end 1: After DMA completes the specified length transfer, it automatically sends a STOP sequence 0: After DMA completes the specified length transfer, wait for software to take over
16	MSP_DMAEN	Master DMA enable 0: Disable DMA function 1: Enable DMA function
15:2		RFU: Unimplemented, read as 0
1	WHEN	SCL pull low timeout enable (Time Out enable) 1: Enable the timeout function. The timeout period is defined by the MSPTO register. 0: Disable the timeout function
0	MSPEN	I2C master module enable control bit (Master enable) 1: I2C master enable 0: I2C master disabled

19.11.2 I2C Master Control Register (I2C_MSPCR)

name	I2C_MSPCR							
Offset	0x00000004							
bit Bit31	Bit name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
bit								
	U-0							
permission	Bit23 Bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
name								
bit	U-0							
permission	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Bit								
name	U-0							
bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
					RCEN PEN		RSEN	ITS
permission	Bit name bit permission	U-0			R/W-0	R/W-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:4		RFU: Unimplemented, read as 0
3	RCEN	In the host receiving mode, the receive enable bit (Receive enable) 1: Host receive enable 0: Receiving prohibited In host communication, the software sets RCEN to transmit after sending the address byte. The transmission direction is switched to master receiving, and then the data from the slave can be received. RCNE remains 1 during the reception process until software sets PEN to send a STOP signal. Timing.
2	PEN	STOP timing generation enable control bit, software writes 1 to send STOP timing, and sends After completion, the hardware will automatically clear to zero (Stop Enable)
1	RSEN	Repeated START sequence generation enable control bit, software writes 1 to send Repeated START sequence, and hardware automatically clears it after sending.

Position No.	Mnemonics	Functional Description
		Enable)
0	ITS	START sequence generation enable control bit, software writes 1 to send START sequence, send After completion, the hardware will automatically clear to zero (Start Enable)

19.11.3 I2C Master Interrupt Enable Register (I2C_MSPIER)

name	I2C_MSPIER							
Offset	0x00000008							
Bit31 Bit name bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	
	U-0							
permission Bit23 Bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	
name								
bit	U-0							
permission Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	
Bit								
name	U-0							
bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		WCOLE OVTE		SE	PE NECK TXIE			RXIE
permission Bit name bit permission	R/W-0 U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:7		RFU: Unimplemented, read as 0
6	WCOLE	WCOL interrupt enable register (Write collision interrupt enable) 1: Enable write conflict interrupt 0: Disable write conflict interrupt
5	OVTE	SCL timeout interrupt enable register (SCL overtime enable) 1: Enable timeout interrupt 0: Disable timeout interrupt
4	SE	START timing interrupt enable register (Start interrupt enable) 1: Enable START timing interrupt 0: Disable START timing interrupt
3	ON	STOP timing interrupt enable register (Stop interrupt enable) 1: Enable STOP timing interrupt 0: Disable STOP timing interrupt
2	NECK	NACK interrupt enable register in master transmission mode (Non-ACK interrupt enable) 1: Allow interrupt when NACK is received 0: Disable NACK interrupt
1	LET'S GO	I2C master transmit completion interrupt enable (Trasnmit done interrupt enable) 1: Enable the send completion interrupt 0: Disable the sending completion interrupt
0	RXIE	I2C master receive done interrupt enable (Receive done interrupt enable) 1: Enable receive completion interrupt 0: Disable receive completion interrupt

19.11.4 I2C Master Interrupt Flag Register (I2C_MSPISR)

name	I2C_MSPISR							
Offset	0x0000000C							
bit Bit31	Bit name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
bit								
	U-0							
permission	Bit23 Bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
name								
bit	U-0							
permission	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Bit								
name	U-0							
bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		WCOL OVT		S	P	ACKSTA TXIF		RXIF
permission	Bit name bit permission	R/W-0 U-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:7		RFU: Unimplemented, read as 0
6	WCOL	Write collision detection bit, MCU can only complete the START timing or send a frame to complete the read MSPBUF can only be written after writing, otherwise a write conflict occurs; hardware set, software write 1 ÿÿ(Write Collision Interrupt Flag,write 1 to clear) 1: Send write conflict 0: No conflict
5	OVT	SCL OverTime Interrupt Flag, only works when TOEN is 1 (SCL OverTime Interrupt Flag) 1: SCL timeout occurs 0: No SCL timeout occurred
4	S	START timing transmission completion interrupt flag, set by hardware and cleared after software reads (Start Interrupt flag)
3	P	STOP timing transmission completion interrupt flag, set by hardware and cleared after software reads (Stop interrupt flag)
2	ACKSTA	In master transmission mode, the response signal from the slave is received after the master sends NACK, this flag can generate an interrupt; it is set by hardware and cleared by software writing 1. (Acknowledge Status Flag ,write 1 to clear) 1: The slave responds with NACK 0: The slave responds with ACK
1	TAXI	I2C master transmit completed interrupt flag, hardware set, software write 1 to clear (Trasnmit done interrupt flag, write 1 to clear) This flag register is set after the master receives the ACK or NACK sent back by the slave.
0	RXIF	I2C host receive completed interrupt flag, hardware set, software write 1 to clear (Receive done interrupt flag, write 1 to clear) This flag register is set after the host sends back ACK or NACK.

19.11.5 I2C Master Status Register (I2C_MSPSR)

name	I2C_MSPSR							
Offset	0x00000010							
Bit31 Bit	Name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24

bit	U-0						
permission bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
bit							
name	U-0						
permission bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
bit							
name	U-0						
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
			BUSY RW			BF		ACKMO
permission bit name permission	U-0		R-0	R-0	U-0	R-0	U-0	R/W-0

Position No.	Mnemonics	Functional Description
31:6		RFU: Unimplemented, read as 0
5	BUSY	I2C communication status bit (Busy) 1: The interface is in read/write state and data transmission is in progress. 0: Data transfer completed
4	RW	I2C transfer direction status bit (Read/Write) 1: The host reads data from the slave 0: The host writes data to the slave
3		RFU: Unimplemented, read as 0
2	BF	Buffer Full Status Bit (Buffer Full) take over: 1: Receive completed, MSPBUF is full 0: Reception is not complete, MSPBUF is empty send: 1: Sending, MSPBUF is full 0: Sending completed, MSPBUF empty
1		RFU: Unimplemented, read as 0
0	ACKMO	In master receiving mode, the status of the host response signal (Ack Master output) 1: The host sends back NACK 0: The host sends back ACK Note: You must ***P*** The flag register is cleared to 0 before the software can set it. ***ACKMO***

19.11.6 I2C Master Baud Rate Register (I2C_MSPBGR)

name	I2C_MSPBGR							
Offset	0x00000014							
Bit31		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
Position Name								MSPBRGH[8]
Bit	U-0							R/W-0
permission bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	
Bit23	MSPBRGH[7:0]							
Bit name	R/W-0001 0011							
Bit permission bit Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	
Position Name								MSPBRGL[8]
Bit permissions	U-0							R/W-0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	MSPBRGL[7:0]							
Permissions	R/W-0001 0011							

Position No.	Mnemonics	Functional Description
31:25	·	RFU: Unimplemented, read as 0
24:16	MSPBRGH	The high level width of the SCL clock sent by the host is counted in the I2C working clock. (Master SCL High level length)
15:9	·	RFU: Unimplemented, read as 0
8:0	MSPBRGL	The low level width of the SCL clock sent by the host is counted in the I2C working clock. (Master SCL Low level length)

19.11.7 I2C Master Transmitter/Receiver Buffer Register (I2C_MSPBUF)

name	I2C_MSPBUF							
Offset	0x00000018							
bit Bit31	Bit name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
bit	·							
	U-0							
permission	Bit23 Bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
name	·							
bit	U-0							
permission	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Bit	·							
name	U-0							
bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	MSPBUF							
permission	Bit name bit permission	R/W-0000 0000						

Position No.	Mnemonics	Functional Description
31:8	·	RFU: Unimplemented, read as 0
7:0	MSPBUF	MSPBUF[7:0]: Data reading and writing are completed through the operation of MSPBUF. When the MSPBUF is written, the data is also loaded into the data transmission and reception shift register. (MSPSR); when receiving, MSPBUF and MSPSR form a double buffer structure, read The output data is the data of MSPBUF. After receiving one byte of data, MSPSR Load data into MSPBUF and set I2CIF at the same time. MSPSR is not a direct register Device, no physical address. (Master data Buffer)

19.11.8 I2C Master Timing Control Register (I2C_MSPTCR)

name	I2C_MSPTCR							
Offset	0x0000001C							
Bit31 Bit	Name Bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
	·							
	U-0							
Permission	Bit23 Bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Name	·							

	U-0							
Permission bit	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Position Name								SDAHD[8]
Bit permissions	U-0							R/W-0
Bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
name	SDAHD[7:0]							
Bit permissions	R/W-0000 1010							

Position No.	Mnemonics	Functional Description
31:9		RFU: Unimplemented, read as 0
8:0	SDAHD	Defines the hold time parameters of SDA relative to the falling edge of SCL, based on the I2C working clock count (SDA hold delay) Note: The minimum valid value is 1 and the maximum valid value is MSPBRGL

19.11.9 I2C Master Timeout Register (I2C_MSPTOR)

name	I2C_MSPTOR							
Offset	0x00000020							
Bit31 Bit name bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	
	U-0							
permission Bit23 Bit	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	
name								
bit	U-0							
permission Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	
Bit				TIMEOUT[11:8]				
name	U-0			R/W-1111				
bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	TIMEOUT[7:0]							
permission Bit name bit permission	R/W-1111 1111							

Position No.	Mnemonics	Functional Description
31:12		RFU: Unimplemented, read as 0
11:0	TIMEOUT	Define the slave SCL low level extension timeout period. Software can SCL stretching time out TSCL_STRETCHING_TIMEOUT=TIMEOUT[11:0] * TSCL

19.11.10 I2C Slave Control Register (I2C_SSPCR)

name	I2C_SSPCR							
Offset	0x00000024							
bit Bit31	Bit name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
bit								
	U-00000000							
permission	bit Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Bit								
name	U-00000000							
bit permission	bit Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Position Name							SCLSEN	SSP_DSTONE
Permission	U-00000000						R/W-1	R/W-0
bits	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Position Name				OMG	SDAO_DLYEN	SCLI_ANFEN	A10EN	SSPEN
Bit permissions	U-0			R/W-1	R/W-0	R/W-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:10		RFU: Unimplemented, read as 0
9	SCLSEN	I2C Slave Clock Stretching Enable (SCL Stretching Enable) 0: Disable slave clock stretching 1: Enable slave clock stretching Note: When the slave uses *DMA* When communicating, you must*SCLCEN* Place*1*
8	SSP_DMAEN	I2C Slave DMA enable 1: Enable DMA function 0: Disable DMA function
7:5		RFU: Unimplemented, read as 0
4	OMG	ACK enable bit: (Slave Ack Enable) 1 = slave will send back ACK after receiving. 0 = slave does not send back ACK
3	SDAO_DLYEN	SDA slave output delay enable (SDA output delay enable) 0: bypass slave SDA output delay 1: Enable slave SDA output delay
2	SCLI_END	SCL slave input analog filter enable 0: bypass analog filtering 1: Enable analog filtering
1	A10EN	10-bit address enable: (10bit Slave address enable) 1 = slave uses 10-bit address 0 = slave uses 7-bit address
0	SSPEN	I2C slave enable bit (Slave enable) 1: Enable I2C slave 0: Disable I2C slave

19.11.11 I2C Slave Interrupt Enable Register (I2C_SSPIER)

name	I2C_SSPIER
Offset	0x00000028

Name	I2C_SSPIER						
bit Bit31 Name bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
	U-0						
permission bit Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Name							
bit	U-0						
permission bit	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Bit15							
Name	U-0						
bit permissionBit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
NameADEE	SE	ON THE	WCHOOL	SSPOV AND	ADME	TXIE	RXIE
Bit Permission R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:8		RFU: Unimplemented, read as 0
7	ADEE	Slave address error interrupt enable, 1 is valid (Address Error Enable)
6	SE	Start interrupt enable, 1 is valid (Start interrupt enable)
5	ON	Stop interrupt enable, 1 is valid (Stop interrupt enable)
4	WCOLE	WCOL interrupt enable, 1 is valid (Write Collision Enable)
3	SSPOVE	SSPOV interrupt enable, 1 is valid (Slave Buffer Overflow Enable)
2	ADME	Slave address match interrupt enable, 1 is valid (Address Match Enable)
1	LET'S GO	Transmit interrupt enable, 1 is valid (Transmit interrupt enable)
0	RXIE	Receive completion interrupt enable, 1 is valid (Receive interrupt enable)

19.11.12 I2C Slave Interrupt Flag Register (I2C_SSPISR)

name	I2C_SSPISR						
Offset	0x0000002C						
bit Bit31 Bit name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
bit							
	U-0						
permission bit Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Bit							
name	U-0						
bit permission bit	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Bit15							
Bit name	U-0						
bit permissionBit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Bit Name ADE Bit	S	P	WCOL	SSPOV	ADM	TAXI	RXIF
Permission R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Position No.	Mnemonics	Functional Description
31:8		RFU: Unimplemented, read as 0
7	ADE	Slave address format error, hardware set, software write 1 to clear (Address Error flag,write 1 to clear) In the case of a 7-bit address, an address byte beginning with 11110 is received, or in the case of a 10-bit

Position No.	Mnemonics	Functional Description
		When the first byte of the address does not start with 11110, ADEE is triggered
6	S	When the start timing is detected, the hardware is set and the software automatically clears it after reading (Start flag)
5	P	When the stop timing is detected, the hardware is set and the software is automatically cleared after reading (Stop flag)
4	WCOL	Write Collision flag, hardware set, software write 1 to clear 1: When BF=1, software writes new data to SSPBUF 0: No write conflict When WCOL occurs, new data will be discarded
3	SSPOV	SSPBUF overflow flag, hardware set, software write 1 clear (Slave buffer overflow flag,write 1 to clear) 1: When BF=1, the slave receives new data. 0: No receive overflow If the slave device enables SCL stretching, the received data will not overflow; This SSPOV can only be set when SCLSEN=0.
2	ADM	Slave address match flag, hardware set, software write 1 clear (Address Matched flag ,write 1 to clear) 1: The received 7-bit or 10-bit address is the same as the content of the SLAVE_ADDR register. To 0: The received address is inconsistent with SLAVE_ADDR
1	TAXI	I2C slave sends the completed flag, hardware sets, software writes 1 to clear (Transmit interrupt flag,write 1 to clear)
0	RXIF	I2C slave receiving completion flag, hardware set, software write 1 to clear (Receive interrupt flagÿwrite 1 to clear)

19.11.13 I2C Slave Status Register (I2C_SSPSR)

name	I2C_SSPSR							
Offset	0x00000030							
bit Bit31	Name bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
	U-0							
permission	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Name								
bit	U-0							
permission	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Name								
bit	U-0							
permission	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Name					BUSY RW		AND	BF
bit permission	U-0				R-0	R-0	R-0	R-0

Position No.	Mnemonics	Functional Description
31:4		RFU: Unimplemented, read as 0
3	BUSY	Slave communication flag (Busy) 1: The slave is sending or receiving data. 0: Slave is idle

Position No.	Mnemonics	Functional Description
2	RW	Read/Write direction status register 1: slave receives R/W=1, slave needs to send data to master 0: slave is in the state of receiving data
1	AND	Data/address frame indication 1: The last byte received is data 0: The last byte received is the address
0	BF	Slave data buffer full flag (Buffer Full flag) 1: SSPBUF is full 0: SSPBUF empty

19.11.14 I2C Slave Transmitter Receiver Buffer Register (I2C_SSPBUF)

name	I2C_SSPBUF							
Offset	0x00000034							
bit Bit31	Name bit	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
	U-0							
permission	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Name								
bit	U-0							
permission	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Name								
bit	U-0							
permission	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Name	SSPBUF							
bit permission	R/W-0000 0000							

Position No.	Mnemonics	Functional Description
31:8		RFU: Unimplemented, read as 0
7:0	SSPBUF	SSPBUF[7:0]: Data reading and writing are completed through the operation of SSPBUF. Write to SSPBUF and also load the data transmit/receive shift register (SSPSR); when receiving, SSPBUF and SSPSR form a double buffer structure, read After receiving a byte of data, SSPSR Load data into SSPBUF and set I2CIF at the same time. SSPSR is not a direct Memory, no physical address (Slave Buffer)

19.11.15 I2C Slave Address Register (I2C_SSPADR)

name	I2C_SSPADR							
Offset	0x00000038							
bit Bit31	Bit name	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
bit								
	U-0							
permission	bit Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Bit								
name	U-0							
bit permission	bit Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

Name	I2C_SSPADR							
bit							SSPADDR[9:8]	
Name bit	U-0						R/W-00	
Permission bit Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
Name	SSPADDR[7:0]							
bit Permission bit	R/W-0000 0000							

Position No.	Mnemonics	Functional Description
31:10		RFU: Unimplemented, read as 0
9:0	SSPADDR	Slave Address Register A10EN = 1 All 10-bit addresses are valid A10EN = 0 Only the lower 7 bits are valid