Chapter 13 Inter-integrated Circuit (I2C) interface

The Internal Integrated Circuit Bus (I2C) is widely used for communication between microcontrollers and sensors and other off-chip modules, it supports multi-master and multi-slave modes, and can communicate at 100KHz (standard) and 400KHz (fast) using only two lines (SDA and SCL). Timing and DMA, with CRC checksum function.

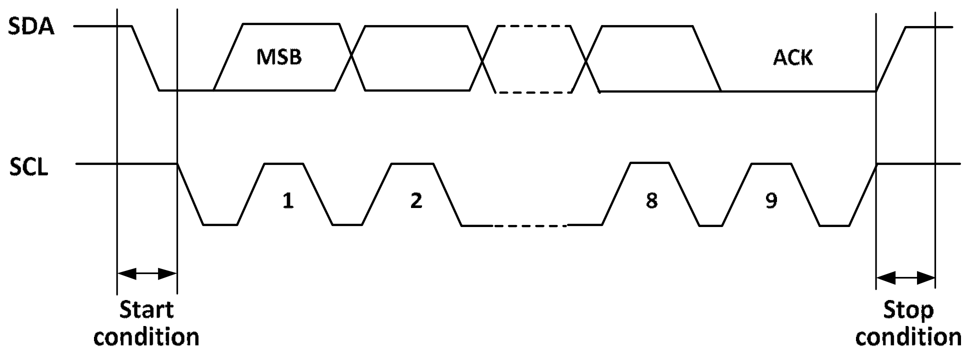
13.1 Main Features

- Support master and slave modes
- Support 7-bit or 10-bit addresses
- Slave devices support dual 7-bit addresses
- Support two speed modes: 100KHz and 400KHz
- Multiple status modes, multiple error flags
- Support extended clock function
- 2 interrupt vectors
- DMA support
- Support PEC
- SMBus compatible

13.2 Overview

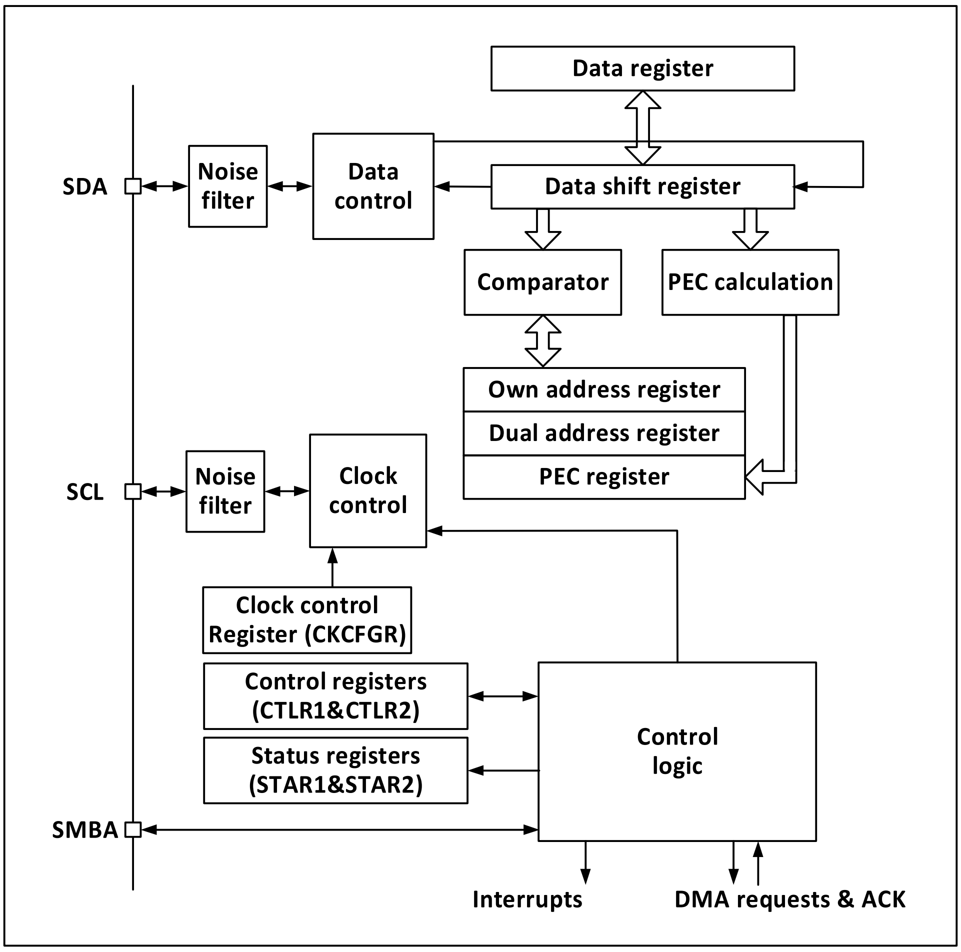
I2C is a half-duplex bus that can only operate in one of the following four modes at the same time: master device transmit mode, master device receive mode, slave device transmit mode and slave device receive mode. the I2C module works in slave mode by default and automatically switches to master mode when a start condition is generated and to slave mode when arbitration is lost or a stop signal is generated. the I2C module supports multi-master functionality. When working in master mode, the I2C module actively emits data and addresses. Both data and address are transmitted in 8-bit units, with the high bit before and the low bit after. After the start event is a one-byte (in 7-bit address mode) or two-byte (in 10-bit address mode) address, and for every 8-bit data or address sent by the host, the slave needs to reply with an answer ACK, which pulls the SDA bus low, as shown in Figure 13-1.

Figure 13-1 I2C Timing Diagram



In order to work properly the I2C must be fed with the correct clock, which is a minimum of 2MHz in standard mode and 4MHz in fast mode.

Figure 13-2 shows the functional block diagram of the I2C module.



13.3 Master Mode

In master mode, the I2C module dominates the data transfer and outputs the clock signal, and the data transfer starts with a start event and ends with an end event. The steps to use master mode communication are.

Setting the correct clock in control register 2 (R16_I2Cx_CTLR2) and clock control register (R16_I2Cx_CKCFGR).

Setting the appropriate rising edge in the rising edge register (R16_I2Cx_RTR).

Setting the PE bit in the control register (R16_I2Cx_CTLR1) to start the peripheral.

Set the START bit in the control register (R16_I2Cx_CTLR1) to generate the start event.

After setting the START bit, the I2C module will automatically switch to the main mode, the MSL bit will be set and the start event will be generated. After the start event is generated, the SB bit will be set and if the ITEVTEN bit (in R16_I2Cx_CTLR2) is set, an interrupt will be generated. The status register 1 (R16_I2Cx_STAR1) should be read at this time and the SB bit will be cleared automatically after writing from the address to the data register.

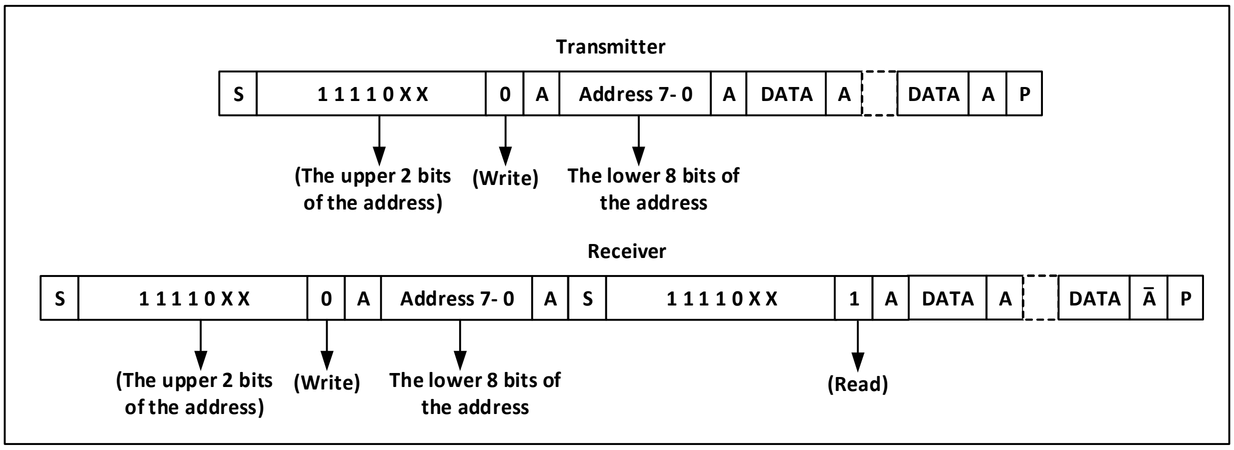
If the 10-bit address mode is used, then the write data register sends the header sequence (the header sequence is 11110xx0b, where the xx bits are the top two bits of the 10-bit address). After sending the header sequence, the ADD10 bit of the status register will be set, and if the ITEVTEN bit has been set, an interrupt will be generated, at this time the R16_I2Cx_STAR1 register should be read and the ADD10 bit cleared after writing the second address byte to the data register.

Then write the data register to send the second address byte, after sending the second address byte, the ADDR bit of the status register will be set, if the ITEVTEN bit is already set, an interrupt will be generated, at this time the R16_I2Cx_STAR1 register should be read and then read the R16_I2Cx_STAR2 register once to clear the ADDR bit;

If the 7-bit address mode, then write data register to send address byte, after sending address byte, ADDR bit

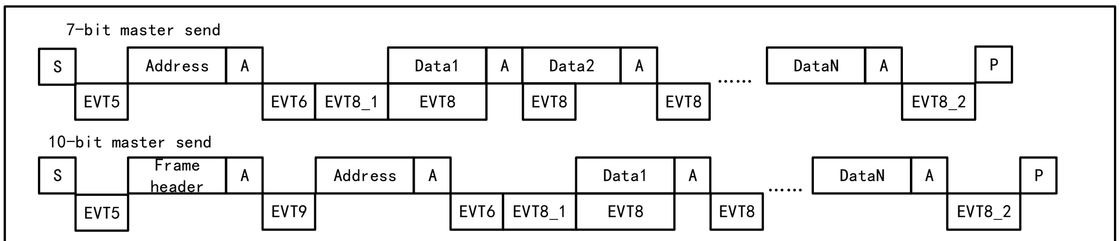
of status register will be set, if ITEVTEN bit has been set, then interrupt will be generated, at this time, R16_I2Cx_STAR1 register should be read and then R16_I2Cx_STAR2 register should be read once to clear ADDR bit;
In 7-bit address mode, the first byte sent is the address byte, the first 7 bits represent the address of the target slave device, the 8th bit determines the direction of the subsequent message, 0 means the master device writes data to the slave device, 1 means the master device reads information to the slave device.
In 10-bit address mode, as shown in Figure 13-3, in the send address phase, the first byte is 11110xx0, xx is the highest 2 bits of the 10-bit address, and the second byte is the lower 8 bits of the 10-bit address. If subsequently enter the master device transmit mode, continue to send data; if subsequently ready to enter the master device receive mode, you need to re-send a start condition, follow to send a byte as 11110xx1, and then enter the master device receive mode.

Figure 13-3 Schematic diagram of master sending and receiving data at 10-bit address



Master transmit mode:
The master device's internal shift register sends data from the data register to the SDA line. When the master device receives an ACK, TxE in status register 1 (R16_I2Cx_STAR1) is set, and an interrupt is also generated if ITEVTEN and ITBUFEN are set. Writing data to the data register will clear the TxE bit.
If the TxE bit is set and no new data was written to the data register before the last data was sent, then the BTF bit will be set and SCL will remain low until it is cleared, and writing data to the data register after reading R16_I2Cx_STAR1 will clear the BTF bit.

Figure 13-4 Master transmitter transmission sequence diagram



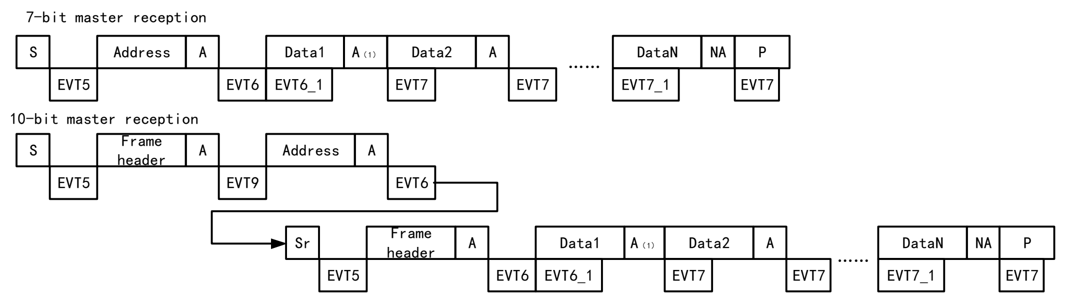
Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=non-response, EVTx=event (interrupt generated when ITEVFEN=1)
EVT5: SB=1, reading SR1 and then writing the address to the DR register will clear the event.
EVT6;ADDR=1,reading SR1 then reading SR2 will clear the event.
EVT8_1: TxE=1, shift register empty, data register empty, write DR register.
EVT8: TxE=1,shift register is not empty,data register is empty,writing DR register will clear the event.
EVT8_2: TxE=1, BTF=1, request to set the stop bit. the TxE and BTF bits are cleared by hardware when the stop condition is generated.
EVT9: ADDR10=1, reading SR1 and then writing to DR register will clear the event.

Note: 1: EVT5, EVT6, EVT9, EVT8_1 and EVT8_2 events elongate the SCL low until the end of the corresponding software sequence.
2: The software sequence of EVT8 must be completed before the end of the current byte transfer.

Master receive mode:

The I2C module will receive data from the SDA line and write it into the data register via a shift register. After each byte, if the ACK bit is set, then the I2C module will send an answer low, and the RxNE bit will be set, and an interrupt will be generated if ITEVTEN and ITBUFEN are set. If RxNE is set and the original data is not read before the new data is received, the BTF bit will be set and SCL will remain low until the BTF is cleared, and reading R16_I2Cx_STAR1 and then reading the data register will clear the BTF bit.

Figure 13-5 Receiver transmission sequence diagram



Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=non-response, EVTx=event (interrupt generated when ITEVFEN=1)

EVT5: SB=1, reading SR1 and then writing the address to DR register will clear the event.
EVT6: ADDR=1,reading SR1 and then reading SR2 will erase this event. In 10-bit master receive mode, START=1 of CR2 should be set after this event.
EVT6_1: There is no corresponding event flag and it is only suitable for receiving 1 byte. Exactly after EVT6 (i.e. after ADDR is cleared), the response and stop condition generation bits should be cleared.
EVT7: RxNE=1, read DR register to clear the event.
EVT7_1: RxNE=1, read the DR register to clear this event. Set ACK=0 and STOP request.
EVT9: ADDR10=1, reading SR1 and then writing to DR register will clear this event.

When the master device ends sending data, it will actively send an end event, i.e. set the STOP bit, and the I2C will switch to slave mode. In receive mode, the master device needs to NAK at the answer position of the last data bit, and after receiving NACK, the slave device releases control of the SCL and SDA lines; the master device can then send a stop/restart condition. Note that the I2C module will automatically switch to slave mode after the stop condition is generated.

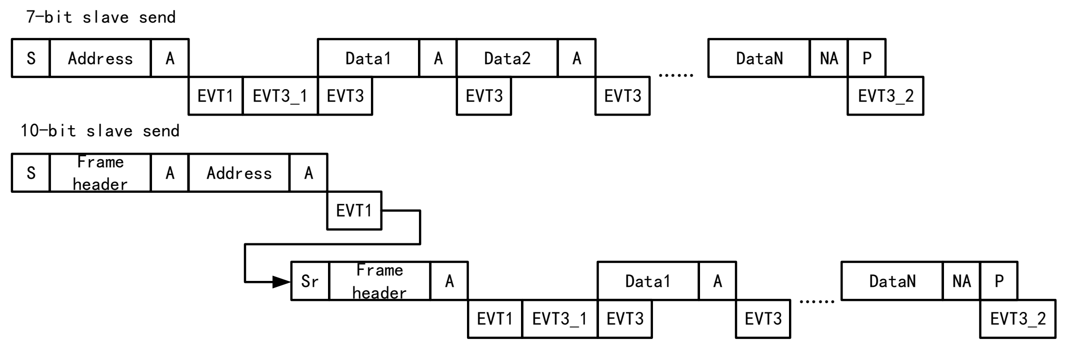
13.4 Slave Mode

When in slave mode, the I2C module recognizes its own address and the broadcast call address. The software can control whether the recognition of the broadcast call address is enabled or disabled. Once a start event is detected, the I2C module compares the SDA data through the shift register with its own address (number of bits depends on ENDUAL and ADDMODE) or the broadcast address (when ENGC is set), if there is a mismatch it will be ignored until a new start event is generated. If it matches the header sequence, an ACK signal is generated and the address of the second byte is waited for; if the address of the second byte also matches or the full segment address matches in the case of a 7- bit address, then:
first an ACK answer is generated;
the ADDR bit is set, and if the ITEVTEN bit is already set, then a corresponding interrupt is also generated;
if the dual address mode is used (ENDUAL bit is set), the DUALF bit also needs to be read to determine which address the host is evoking.
The slave mode is receive mode by default. In case the last bit of the received header sequence is 1, or the last bit of the 7-bit address is 1 (depending on whether the header sequence is received for the first time or a normal 7-bit address), the I2C module will go to transmitter mode and the TRA bit will indicate whether it is currently receiver or transmitter mode.

Slave transmit mode:
After clearing the ADDR bit, the I2C module sends bytes from the data register to the SDA line via a shift register. After an answer ACK is received, the TxE bit is set and an interrupt is generated if ITEVTEN and ITBUFEN are set. If TxE is set but no new data is written to the data register before the end of the next data send, the BTF bit will be set. SCL will remain low until the BTF is cleared. Reading status register 1

(R16_I2Cx_STAR1) and then writing data to the data register will clear the BTF bit.

Figure 13-6 Slave transmitter transmission sequence diagram



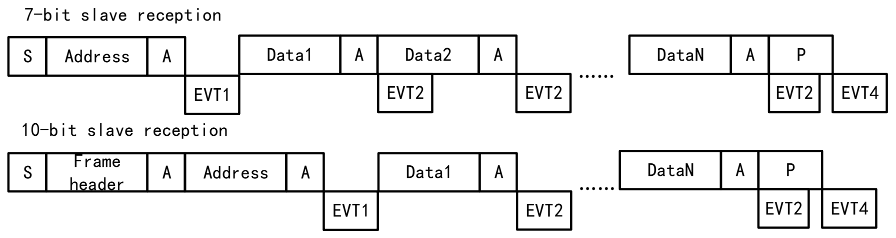
Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=non-response, EVTx=event (interrupt is generated when ITEVFEN=1)

EVT1;ADDR=1,read SR1 then read SR2 will eliminate the event.
EVT3_1: TxE=1, shift register empty, data register empty, write DR.
EVT3: TxE=1,shift register is not empty,data register is empty,writing DR will clear the event.
EVT3_2: AF=1, write '0' in AF bit of SR1 register to clear AF bit.

Note: 1: EVT1 and EVT3_1 events elongate SCL low until the end of the corresponding software sequence.
2: The software sequence of EVT3 must be completed before the end of the current byte transfer.

Slave receive mode:
After ADDR is cleared, the I2C module stores the data on SDA into the data register via the shift register. After each byte is received, the I2C module sets an ACK bit and sets the RxNE bit, and generates an interrupt if ITEVTEN and ITBUFEN are set. If RxNE is set and the old data is not read before the new data is received, then BTF is set. SCL will remain low until the BTF bit is cleared. Reading status register 1 (R16_I2Cx_STAR1) and reading the data in the data register will clear the BTF bit.

Figure 13-7 Receiver transmission sequence diagram



Description: S=Start (start condition), Sr=repeated start condition, P=Stop (stop condition), A=response, NA=non-response, EVTx=event (interrupt generated when ITEVFEN=1)
EVT1;ADDR=1,reading SR1 then SR2 will erase the event.
EVT2: RxNE=1, reading DR will clear the event.
EVT4: STOPF=1,reading SR1 and then writing CR1 register will clear the event.

Note: 1: EVT1 event elongates SCL low until the end of the corresponding software sequence.
2: The software sequence of EVT2 must be completed before the end of the current byte transfer.

The master device will generate a stop condition after the last data byte is transferred. When the I2C module detects a stop event, it will set the STOPF bit, and if the ITEVFEN bit is set, it will also generate an interrupt. The user needs to read the status register (R16_I2Cx_STAR1) and then write the control register (e.g. reset control word SWRST) to clear it. (See EVT4 in the figure above).

13.5 Error Conditions

13.5.1 Bus Error (BERR)

A bus error will be generated when the I2C module detects an external start or stop event during address or data transfer. When a bus error is generated, the BERR bit is set and an interrupt is generated if ITERREN is set. In slave mode, the data is discarded and the hardware releases the bus. If it is a start signal, the hardware assumes it is a restart signal and starts waiting for an address or stop signal; if it is a stop signal, it operates ahead of normal stop conditions. In master mode, the hardware does not release the bus while not affecting the current transfer, and it is up to the user code to decide whether to abort the transfer.

13.5.2 Acknowledge Failure (AF)

An answer error will be generated when the I2C module detects a byte and then no answer. When an answer error is generated: AF will be set and an interrupt will be generated if ITERREN is set; when an AF error is encountered, the hardware must release the bus if the I2C module is working in slave mode and the software must generate a stop event if it is in master mode.

13.5.3 Arbitration Lost (ARLO)

An arbitration lost error is generated when the I2C module detects an arbitration lost. When an arbitration loss error is generated: the ARLO bit is set and an interrupt is generated if ITERREN is set; the I2C module switches to slave mode and no longer responds to transfers initiated against its slave address unless a new start event is initiated by the host; the hardware releases the bus.

13.5.4 Overrun/underrun Error (OVR)

- **Overrun error**

In Slave mode, if the clock extension is disabled and the I2C module is receiving data, an overrun error will occur if a byte of data has been received but the last received data has not been read out. When an overrun error occurs, the last received byte will be discarded and the sender should retransmit the last sent byte.

- **Underrun error**

In Slave mode, if the clock is forbidden to extend and the I2C module is sending data, an underrun error will occur if new data has not been written to the data register before the next byte of the clock comes. In case of an underrun error, the data in the previous data register will be sent twice, and if an underrun error occurs, then the receiver should discard the data received repeatedly. In order not to generate an underrun error, the I2C module should write the data to the data register before the first rising edge of the next byte.

13.6 Clock Extension

If clock extension is disabled, then there is a possibility of overrun/underrun errors. However, if clock extension is enabled:

- In transmit mode, if TxE is set and BTF is set, SCL will always be low, always waiting for the user to read the status register and write the data to be sent to the data register.
- In receive mode, if RxNE is set and BTF is set, SCL will remain low after data is received until the user reads the status register and reads the data register.

It can be seen that enabling clock extension can avoid overrun/underrun errors.

13.7 SMBus

SMBus is also a 2-wire interface, which is generally used between system and power management. SMBus and I2C have many similarities, for example, SMBus uses the same 7-bit address mode as I2C, and the following are common to SMBus and I2C.

1) Master-slave communication mode, where the host provides the clock and supports multiple masters and slaves.
2) 2-wire communication architecture, with an optional warning line for SMBus.
3) Both support 7-bit address format.

There are also differences between SMBus and I2C.

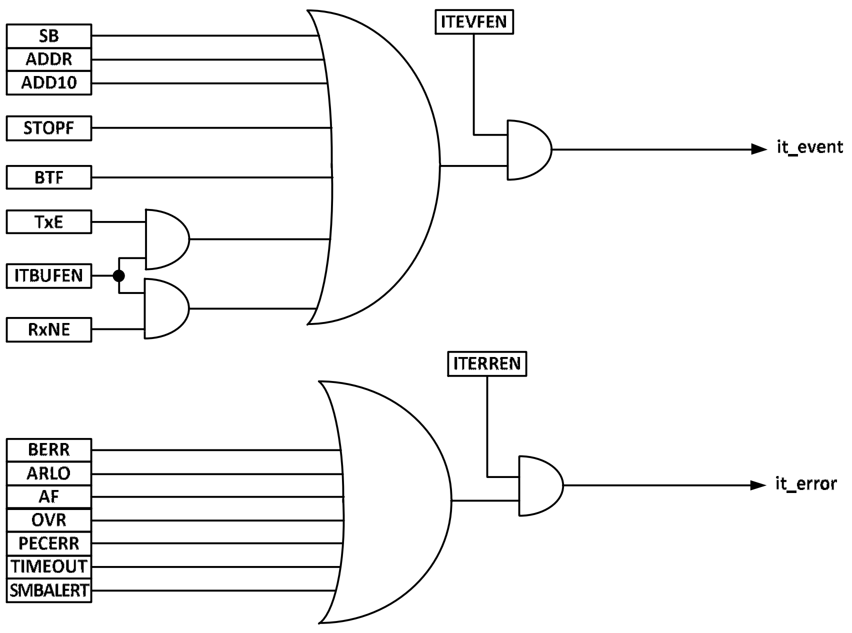
1) I2C supports speeds up to 400 KHz, while SMBus supports up to 100 KHz, and SMBus has a minimum speed limit of 10 KHz.
2) A timeout will be reported when the SMBus clock is low for more than 35mS, but there is no such limit for I2C.
3) SMBus has a fixed logic level, while I2C does not, depending on VDD.
4) SMBus has a bus protocol, while I2C does not.

SMBus also includes device identification, address resolution protocols, unique device identifiers, SMBus reminders and various bus protocols as described in the SMBus specification version 2.0. When using SMBus, only the SMBus bit of the control register needs to be set, and the SMBTYPE bit and ENAARP bit need to be configured as needed.

13.8 Interruptions

Each I2C module has two interrupt vectors, event interrupts and error interrupts. Both interrupts support the interrupt sources in Figure 13-4.

Figure 13-4 I2C Interrupt Request



13.9 DMA

DMA can be used to send and receive bulk data. The ITBUFEN bit of the control register cannot be set when using DMA.

- Transmission using DMA

DMA mode can be activated by setting the DMAEN bit of the CTLR2 register. As long as the TxE bit is set, data will be loaded by DMA from the set memory into the data register of the I2C. The following settings are required to allocate channels for I2C.

1) Set the I2Cx_DATAR register address to the DMA_PADDRx register and the memory address in the DMA_MADDRx register so that after each TxE event, data will be sent from memory to the

I2Cx_DATAR register.

2) Set the required number of bytes to be transferred in the DMA_CNTRx register. This value will be decremented after each TxE event.
3) Configure the channel priority using the PL[0:1] bits in the DMA_CFGRx register.
4) Set the DIR bit in the DMA_CFGRx register and depending on the application requirements can be configured to issue an interrupt request when the entire transfer is half or fully completed.
5) Activate the channel by setting the EN bit on the DMA_CFGRx register.

When the number of data transfer bytes set in the DMA controller has been completed, the DMA controller sends an end of transfer EOT/ EOT_1 signal to the I2C interface. A DMA interrupt will be generated if the interrupt is allowed.

- Reception using DMA

DMA receive mode can be performed after setting DMAEN in the CTLR2 register. When using DMA receive, DMA transfers the data in the data register to the preset memory area. The following steps are required to allocate channels for I2C.

1) Set the I2Cx_DATAR register address to the DMA_PADDRx register and the memory address in the DMA_MADDRx register so that after each RxNE event, data will be written to memory from the I2Cx_DATAR register.
2) Set the required number of bytes to be transferred in the DMA_CNTRx register. This value will be decremented after each RxNE event.
3) Configure the channel priority with PL[0:1] in the DMA_CFGRx register.
4) The DIR bit in the DMA_CFGRx register is cleared, and depending on the application requirements, an interrupt request can be set to be issued when the data transfer is half or fully completed.
5) Set the EN bit in the DMA_CFGRx register to activate the channel.

When the number of data transfers set in the DMA controller has been completed, the DMA controller sends an end of transfer EOT/EOT_1 signal to the I2C interface. A DMA interrupt will be generated if the interrupt is allowed.

13.10 Packet Error Checking

Packet Error Checksum (PEC) is an additional CRC8 checksum step to provide transmission reliability, calculated for each bit of serial data using the following polynomial.

$$C=X^8 +X^2 +X+1$$

The PEC calculation is activated by the ENPEC bit in the control register and is performed on all information bytes, including address and read/write bits. In transmitting, enabling PEC adds a byte of CRC8 calculation result after the last byte of data; while in receiving mode, in the last byte is considered as CRC8 check result, and if it does not match with the internal calculation result, it will reply a NAK, and in case of the main receiver, regardless of the correct check result.

13.11 Register Description

Table 13-1 I2C-related registers list

Name	Offset address	Description	Reset value
R16_I2C_CTLR1	0x40005400	I2C control register 1	0x0000
R16_I2C_CTLR2	0x40005404	I2C control register 2	0x0000
R16_I2C_OADDR1	0x40005408	I2C address register 1	0x0000
R16_I2C_OADDR2	0x4000540C	I2C address register 2	0x0000
R16_I2C_DATAR	0x40005410	I2C data register	0x0000
R16_I2C_STAR1	0x40005414	I2C status register 1	0x0000
R16_I2C_STAR2	0x40005418	I2C status register 2	0x0000
R16_I2C_CKCFGR	0x4000541C	I2C clock register	0x0000

13.11.1 I2C Control Register 1(I2C1_CTLR1)

Offset address: 0x00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Reserved		PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	Reserved				PE

Bit	Name	Access	Description	Reset value
15	SWRST	RW	Software reset, setting this bit by user code will reset the I2C peripheral. Make sure the pins of the I2C bus are released and the bus is idle before the reset. *Note: This bit resets the I2C module when no stop condition is detected on the bus but the busy bit is 1.*	0
[14:13]	Reserved	RO	Reserved	0
12	PEC	RW	Packet error checking bit, set this bit to enable packet error detection. The user code can set or clear this bit; the hardware clears this bit when the PEC is transmitted, when a start or end signal is generated, or when the PE bit is cleared to 0. 1: With PEC. 0: Without PEC. *Note: The PEC is invalidated when arbitration is lost.*	0
11	POS	RW	ACK and PEC position setting bits, which can be set or cleared by user code and can be cleared by hardware after the PE has been cleared. 1: ACK bit controls the ACK or NAK of the next byte received in the shift register. The next byte received in the PEC shift register is the PEC. 0: The ACK bit controls the ACK or NAK of the byte currently being accepted in the shift register. the PEC bit indicates that the byte in the shift register before the current bit is PEC. *Note: The POS bit is used in 2-byte data reception as follows: it must be configured before reception. In order to NACK the 2nd byte, the ACK bit must be cleared immediately after clearing the ADDR bit; in order to detect the PEC of the second byte, the PEC bit must be set after the ADDR event and after configuring the POS bit.*	0
10	ACK	RW	Acknowledge enable, This bit is set and cleared by software and cleared by hardware when PE=0. 1：Acknowledge returned after a byte is received. 0：No acknowledge returned.	0
9	STOP	RW	Stop generation bit. This bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected. In Master mode: 1：Stop generation after the current byte transfer or after the current Start condition is sent. 0：No Stop generation. In Slave mode: 1：Release the SCL and SDA lines after the current byte transfer.	0

			0：No Stop generation.	
8	START	RW	Start generation. This bit is set and cleared by software and cleared by hardware when start is sent or PE=0. In Master mode: 1：Repeated start generation 0：No Start generation In Slave mode: 1：Start generation when the bus is free 0：No Start generation	0
7	NOSTRETCH	RW	Clock stretching disable bit. This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software. 1：Clock stretching disabled. 0：Clock stretching enabled.	0
6	ENGC	RW	General call enable bit. Set this bit to enable broadcast call and answer broadcast address 00h.	0
5	ENPEC	RW	PEC enable bit, set this bit to enable PEC calculation.	0
[4:1]	Reserved	RO	Reserved	0
0	PE	RW	I2C peripheral enable bit. 1: Enable the I2C module. 0: Disable the I2C module.	0

13.11.2 I2C Control Register 2(I2C1_CTLR2)

Offset address: 0x04

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved		FREQ[5:0]					

Bit	Name	Access	Description	Reset value
[15:13]	Reserved	RO	Reserved	0
12	LAST	RW	DMA last transfer bit. 1：Next DMA EOT is the last transfer. 0：Next DMA EOT is not the last transfer. *Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.*	0
11	DMAEN	RW	DMA requests enable bit. Set this bit to allow DMA request when TxE or RxEN is set.	0
10	ITBUFEN	RW	Buffer interrupt enable bit. 1：When TxE or RxEN is set, event interrupt is generated. 0：When TxE or RxEN is set, no interrupt is generated.	0

9	ITEVTEN	RW	Event interrupt enable bit. Set this bit to enable event interrupt. This interrupt will be generated under the following conditions.
			SB=1 (Master mode).
			ADDR=1 (Master-slave mode).
			ADDR10 = 1 (Master mode).
			STOPF=1 (Slave mode).
			BTF = 1, but no TxE or RxEN events.
			TxE event to 1 if ITBUFEN = 1.
			RxNE event to 1if ITBUFEN = 1.
				0

8	ITERREN	RW	Error interrupt enable bit. Set to allow error interrupts. The interrupt will be generated under the following conditions. BERR=1; ARLO=1; AF=1; OVR=1; PECERR=1. TIMEOUT=1; SMBAlert=1.	0
[7:6]	Reserved	RO	Reserved	0
[5:0]	FREQ[5:0]	RW	The I2C module clock frequency field, which must be entered at the correct clock frequency to produce the correct timing, allows a range between 8-48MHz. It must be set between 001000b and 110000b in MHz.	0

13.11.3 I2C Own Address Register 1(I2C1_OAR1)

Offset address: 0x08

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0

Bit	Name	Access	Description	Reset value
15	ADDMODE	RW	Address mode. 1: 10-bit slave address (does not respond to 7-bit addresses). 0: 7-bit slave address (does not respond to 10-bit address)	0
[14:10]	Reserved	RO	Reserved	0
[9:8]	ADD[9:8]	RW	Interface address, bits 9-8 when using a 10-bit address, ignored when using a 7-bit address.	0
[7:1]	ADD[7:1]	RW	Interface address, bits 7-1.	0
0	ADD0	RW	Interface address, bit 0 when using a 10-bit address, ignored when using a 7-bit address.	0

13.11.4 I2C Own Address Register 2(I2C1_OAR2)

Offset address: 0x0C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL

Bit	Name	Access	Description	Reset value
[15:8]	Reserved	RO	Reserved	0
[7:1]	ADD2[7:1]	RW	Interface address, bits 7-1 of the address in dual address mode.	0
0	ENDUAL	RW	Dual address mode enable bit, set this bit to allow ADD2 to be recognized as well.	0

13.11.5 I2C Data Register (I2C_DATAR)

Offset address: 0x10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DR[7:0]							

Bit	Name	Access	Description	Reset value
15:8	Reserved	RO	Reserved	0
7:0	DR[7:0]	RW	Data register, this field is used to store the received data or to store the data used to send to the bus.	0

13.11.6 I2C Status Register 1(I2C_STAR1)

Offset address: 0x14

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB

Bit	Name	Access	Description	Reset value
[15:13]	Reserved	RO	Reserved	0
12	PECERR	RW0	The PEC error flag bit occurs on reception, and this bit can be reset by a user write of 0 or by hardware when PE goes low. 1: There is a PEC error and the PEC is received and NAK is returned. 0: No PEC error.	0
11	OVR	RW0	Overrun and underrun flag bits. 1: There are overrun and underrun events occurring: when NOSTRETCH=1, when a new byte is received in receive mode, the content in the data register has not been read out, then the newly received byte will be lost; when in send mode, no new data is written to the data register, and the same byte will be sent twice. 0: No overrun or underrun events.	0
10	AF	RW0	Acknowledge failure bit. Cleared by software writing 0, or by hardware when PE=0. 1：Acknowledge failure. 0：No acknowledge failure.	0
9	ARLO	RW0	Arbitration lost bit. Cleared by software writing 0, or by hardware when PE=0. 1：Arbitration Lost detected. 0：No Arbitration Lost detected.	0
8	BERR	RW0	The bus error flag bit, which can be reset by a user write of 0, or by hardware when PE goes low. 1: Error in start or stop condition; 0: Normal.	0
7	TxE	RO	Data register empty bit. Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0. 1：Data register empty. 0：Data register not empty.	0
6	RxNE	RO	Data register not empty bit. Cleared by software reading or writing the DR register or by hardware when PE=0. 1：Data register not empty. 0：Data register empty.	0
5	Reserved	RO	Reserved	0
4	STOPF	RO	Stop detection bit. Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0 1：Set by hardware when a Stop condition is detected on the bus by the slave after an	0

			acknowledge (if ACK=1). 0：No Stop condition detected.	
3	ADD10	RO	10-bit header sent bit. Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0. 1：Master has sent first address byte. 0：No ADD10 event occurred.	0
2	BTF	RO	Byte transfer finished bit. Cleared by software reading SR1 followed by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0. 1：Data byte transfer succeeded. When NOSTRETCH=0: when sending, when a new data is sent and the data register has not yet been written with new data; when receiving, when a new byte is received but the data register has not yet been read. 0：Data byte transfer not done.	0
1	ADDR	RW0	Address sent /matched bit. This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0. In Master mode: 1：End of address transmission. For 10-bit addressing, the bit is set after the ACK of the 2nd byte. For 7-bit addressing, the bit is set after the ACK of the byte. 0：No end of address transmission. In Slave mode: 1：Received address matched. 0：Address mismatched or not received.	0
0	SB	RO	Start bit. Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0 1：Start condition generated. 0：No Start condition.	0

13.11.7 I2C Status Register 2(I2C_STAR2)

Offset address: 0x18

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	Reserved		GENCALL	Reserved	TRA	BUSY	MSL

Bit	Name	Access	Description	Reset value
[15:8]	PEC[7:0]	RO	Packet error checking bit. When PEC is enabled (ENPEC is set), this field holds the value of PEC.	0
7	DUALF	RO	Dual flag. Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0. 1：Received address matched with OAR2. 0：Received address matched with OAR1.	0
[6:5]	Reserved	RO	Reserved	0
4	GENCALL	RO	General call address bit. Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.	0

			1：General Call Address received when ENGC=1. 0：No General Call.	
3	Reserved	RO	Reserved	0
2	TRA	RO	Transmitter/receiver bit. It is cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0. 1：Data bytes transmitted. 0：Data bytes received. This bit is set depending on the R/W bit of the address byte.	0
1	BUSY	RO	Bus busy bit. Cleared by hardware on detection of a Stop condition. This information is still updated when the interface is disabled (PE=0). 1：Communication ongoing on the bus: low level present in SDA or SCL. 0：No communication on the bus.	0
0	MSL	RO	Master/slave bit. Set by hardware as soon as the interface is in Master mode (SB=1). Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.	0

13.11.8 I2C Clock Register (I2C1_CKCFGR)

Offset address: 0x1C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											

Bit	Name	Access	Description	Reset value
15	F/S	RW	Master mode selection bit. 1：Fm mode I2C. 0：Sm mode I2C	0
14	DUTY	RW	Duty cycle in the fast mode: 1: $T_{Low\ level}/T_{High\ level}$=16/9; 0: $T_{Low\ level}/T_{High\ level}$=2.	0
[13:12]	Reserved	RO	Reserved	0
[11:0]	CCR[11:0]	RW	Clock control register in Fm/Sm mode	0