15 Inter-integrated circuit (I^2C) interface

15.1 Introduction

I^2C (Inter-Integrated Circuit) Bus Interface serves as an interface between the microcontroller and the serial I^2C bus. It provides multi-master capability, and controls all I^2C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes. It is also SMBus 2.0 compatible.

It may be used for a variety of purposes, including CRC generation and verification, SMBus (System Management Bus) and PMBus (Power Management Bus).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

15.2 Main features

- Parallel-bus/I^2C protocol converter
- Multi-master capability: the same interface can act as Master or Slave
- I^2C Master features:
  - Clock generation
  - Start and Stop generation
- I^2C Slave features:
  - Programmable I^2C Address detection
  - Dual Addressing Capability to acknowledge 2 slave addresses
  - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and General Call
- Supports different communication speeds:
  - Standard Speed (up to 100 kHz),
  - Fast Speed (up to 400 kHz)
- Status flags:
  - Transmitter/Receiver mode flag
  - End-of-Byte transmission flag
  - I^2C busy flag
- Error flags:
  - Arbitration lost condition for master mode
  - Acknowledgement failure after address/ data transmission
  - Detection of misplaced start or stop condition
  - Overrun/Underrun if clock stretching is disabled
- 2 Interrupt vectors:
  - 1 Interrupt for successful address/ data communication
  - 1 Interrupt for error condition
- Optional Clock Stretching
- 1-byte buffer with DMA capability

- Configurable PEC (Packet Error Checking) Generation or Verification:
  - PEC value can be transmitted as last byte in Tx mode
  - PEC error checking for last received byte
- SMBus 2.0 Compatibility:
  - 25 ms clock low timeout delay
  - 10 ms master cumulative clock low extend time
  - 25 ms slave cumulative clock low extend time
  - Hardware PEC generation/verification with ACK control
  - Address Resolution Protocol (ARP) supported
- PMBus Compatibility

Note: Some of the above features may not be available in certain products. The user should refer to the product data sheet, to identify the specific features supported by the I^2C interface implementation.

15.3 General description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or fast (up to 400 kHz) I^2C bus.

Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a STOP generation occurs, allowing Multi-Master capability.

Communication flow

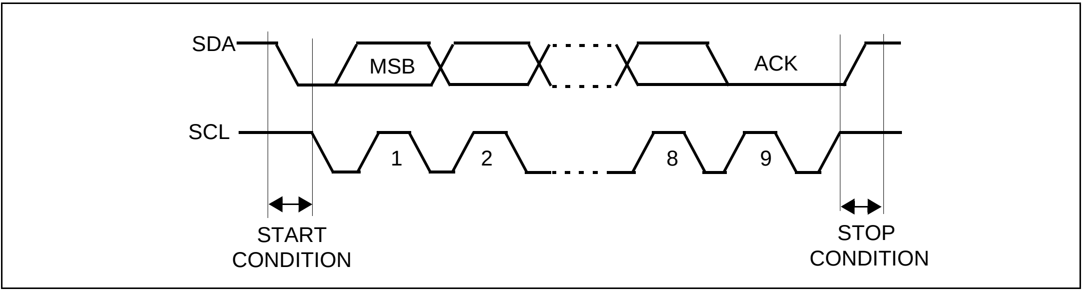
In Master mode, the I^2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

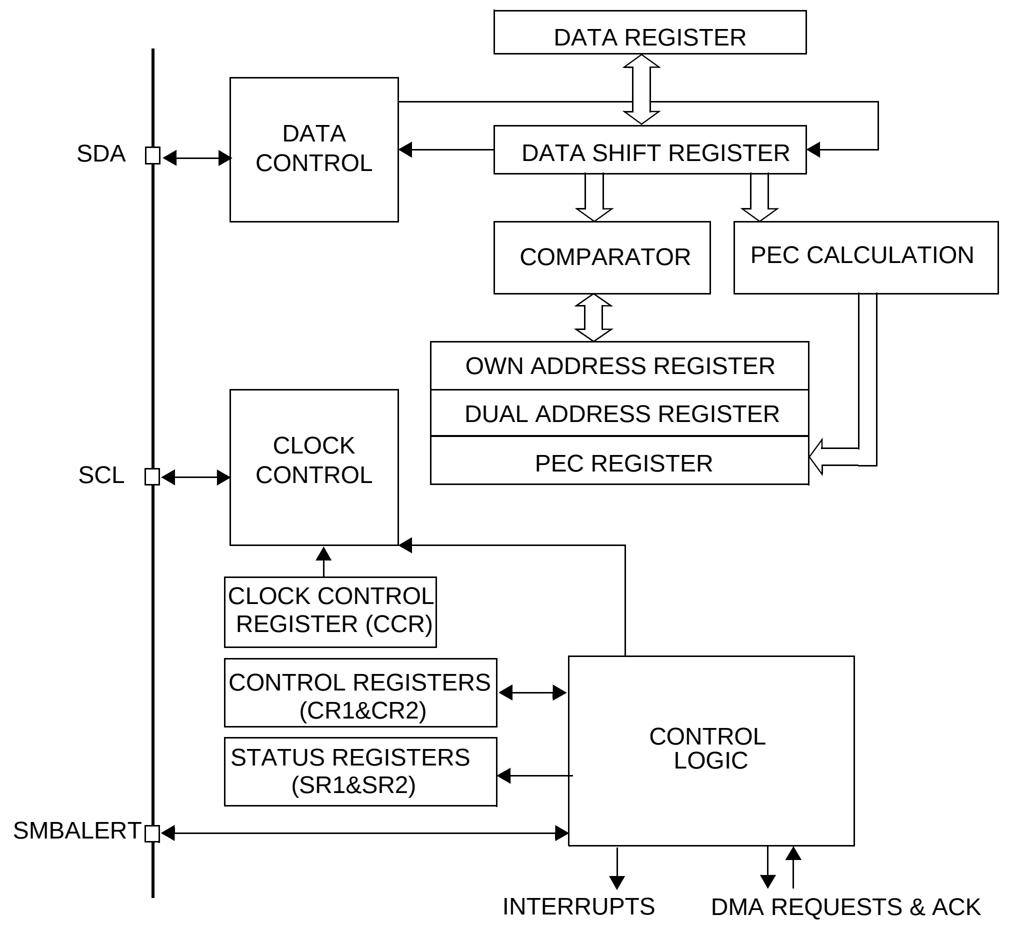
Figure 133. I^{2}C bus protocol



Acknowledge may be enabled or disabled by software. The I^{2}C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.

The Block Diagram of the I^{2}C interface is shown in *Figure 134*.

Figure 134. I^{2}C block diagram



Note: *SMBALERT is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.*

15.4 Functional description

By default the I^{2}C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.

15.4.1 I^{2}C slave mode

The peripheral input clock must be programmed in the I2C_CR2 register in order to generate correct timings. The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (OAR1) and with OAR2 (if ENDUAL=1) or the General Call address (if ENGC = 1).

Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.

Header or address not matched: the interface ignores it and waits for another Start condition.

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.

Address matched: the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
- If ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.

Slave transmitter

Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.

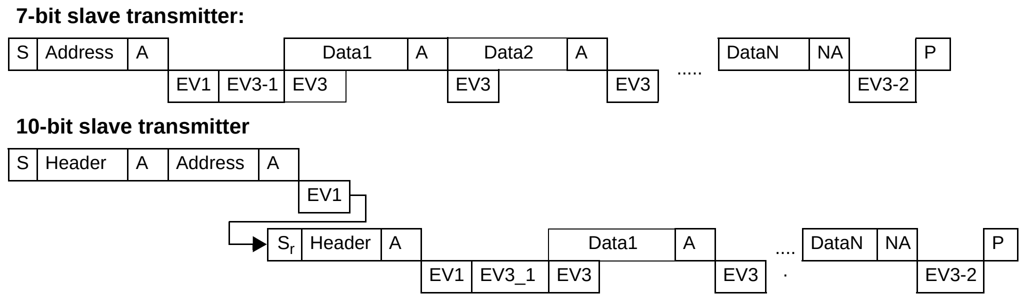
The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see *Figure 135* Transfer sequencing EV1 EV3).

When the acknowledge pulse is received:

- The TxE bit is set by hardware with an interrupt if the ITEVFEN and the ITBUFEN bits are set.

If TxE is set and a data was not written in the DR register before the end of the last data transmission, the BTF bit is set and the interface waits for a write in the DR register, stretching SCL low.

Figure 135. Transfer sequence diagram for slave transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge,
EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV3-1: TxE=1, shift register empty.
EV3: TxE=1, cleared by writing DR; shift register not empty
EV3-2: AF=1; AF is cleared by writing ‘0’ in AF bit of SR1 register.

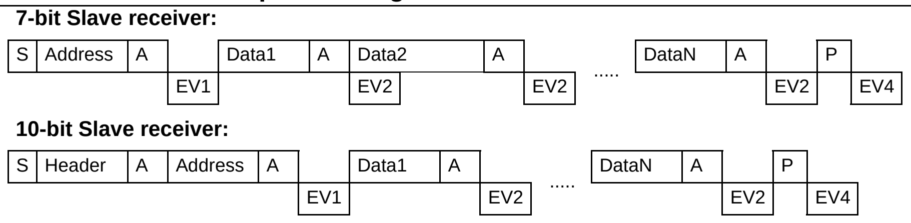
Slave receiver

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RxNE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bit is set.

If RxNE is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set and the interface waits for a read to the DR register, stretching SCL low (see *Figure 136* Transfer sequencing).

Figure 136. Transfer sequence diagram for slave receiver



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV2: RxNE=1 cleared by reading DR register.
EV4: STOPF=1, cleared by reading SR1 register followed by writing to the CR1 register

Closing slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets,

- The STOPF bit and generates an interrupt if the ITEVFEN bit is set.

Then the interface waits for a read of the SR1 register followed by a write to the CR1 register (see *Figure 136* Transfer sequencing EV4).

15.4.2 I^{2}C master mode

In Master mode, the I^{2}C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Master mode is selected as soon as the Start condition is generated on the bus with a START bit.

The following is the required sequence in master mode.

- Program the peripheral input clock in I2C_CR2 Register in order to generate correct timings
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CR1 register to enable the peripheral
- Set the START bit in the I2C_CR2 register to generate a Start condition

The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

Start condition

Setting the START bit while the BUSY bit is cleared causes the interface to generate a Start condition and switch to Master mode (M/SL bit set).

Once the Start condition is sent:

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address (see *Figure 137* & *Figure 138* Transfer sequencing EV5).

Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 10-bit addressing mode, sending the header sequence causes the following event:
  - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see *Figure 137* & *Figure 138* Transfer sequencing).
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 137* & *Figure 138* Transfer sequencing).
- In 7-bit addressing mode, one address byte is sent.

  As soon as the address byte is sent,
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 137* & *Figure 138* Transfer sequencing).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent.

- In 7-bit addressing mode,
  - To enter Transmitter mode, a master sends the slave address with LSB reset.
  - To enter Receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode,
  - To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address with LSB reset, (where xx denotes the two most significant bits of the address).
  - To enter Receiver mode, a master sends the header (11110xx0) and then the slave address with LSB reset. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).

  The TRA bit indicates whether the master is in Receiver or Transmitter mode.

Master transmitter

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits until TxE is cleared, (see *Figure 137* Transfer sequencing EV8).

When the acknowledge pulse is received:

- The TxE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set.

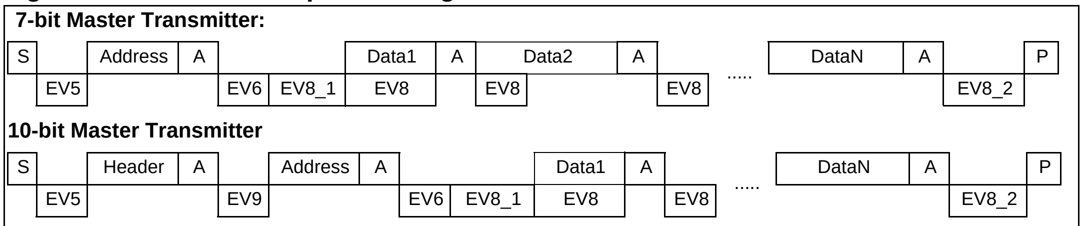
If TxE is set and a data byte was not written in the DR register before the end of the last data transmission, BTF is set and the interface waits until BTF is cleared.

Closing the communication

After writing the last byte to the DR register, the STOP bit is set by software to generate a Stop condition (see *Figure 137* Transfer sequencing EV8_2). The interface goes automatically back to slave mode (M/SL bit cleared).

Note: Stop condition should be programmed during EV8_2 event, when either TxE or BTF is set.

Figure 137. Transfer Sequence Diagram for Master Transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register with Address.
EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2.
EV8_1: TxE=1 shift register empty
EV8: TxE=1 cleared by writing DR register.
EV8_2: TxE=1, BTF = 1 cleared by HW by stop condition
EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

Master receiver

Following the address transmission and after clearing ADDR, the I^{2}C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RxNE bit is set and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set (see *Figure 138* Transfer sequencing EV7).

If the RxNE bit is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set by hardware and the interface waits for a read in the DR register.

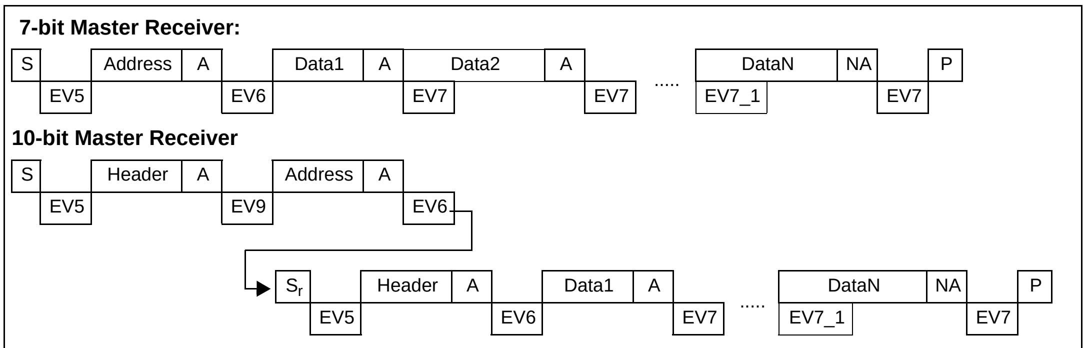
Closing the communication

The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Re-Start condition.

- In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RxNE event).
- In order to generate the Stop/Re-Start condition, software must set the STOP/START bit just after reading the second last data byte (after the second last RxNE event).

After the Stop condition generation, the interface goes automatically back to slave mode (M/SL bit cleared).

Figure 138. Transfer sequence diagram for master receiver



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register.

EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.

EV7: RxNE=1 cleared by reading DR register.

EV7_1: RxNE=1 cleared by reading DR register, program ACK=0 and STOP request

EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

15.4.3 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

This error occurs when the I^2C interface detects a Stop or a Start condition during a byte transfer. In this case,

- The BERR bit is set and an interrupt is generated if the ITERREN bit is set
- In case of Slave: data is discarded and the lines are released by hardware:
  - in case of misplaced start, the slave considers it is a restart and waits for address, or stop condition.
  - in case of misplaced stop, the slave reacts like for a stop condition and the lines are released by hardware.

Acknowledge failure (AF)

This error occurs when the interface detects a non-acknowledge bit. In this case,

- The AF bit is set and an interrupt is generated if the ITERREN bit is set
- A transmitter which receives a NACK must reset the communication:
  - If Slave: lines are released by hardware
  - If Master: a Stop condition must be generated by software

Arbitration lost (ARLO)

This error occurs when the I^2C interface detects an arbitration lost condition. In this case,

- The ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set)
- The I^2C Interface goes automatically back to slave mode (the M/SL bit is cleared)
- Lines are released by hardware

Overrun/underrun error (OVR)

An Overrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is receiving data. The interface has received a byte (RxNE=1) and the data in DR has not been read, before the next byte is received by the interface. In this case,

- The last received byte is lost.
- In case of Overrun error, software should clear the RxNE bit and the transmitter should re-transmit the last received byte.

Underrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is transmitting data. The interface has not updated the DR with the next byte (TxE=1), before the clock comes for the next byte. In this case,

- The same byte in the DR register will be sent again
- The user should make sure that data received on the receiver side during an underrun error is discarded and that the next bytes are written within the clock low time specified in the I^2C bus standard.

15.4.4 SDA/SCL line control

- If clock stretching is enabled:
  - Transmitter mode: If TxE=1 and BTF=1: the interface holds the clock line low before transmission to wait for the microcontroller to read SR1 and then write the byte in the Data Register (both buffer and shift register are empty).
  - Receiver mode: If RxNE=1 and BTF=1: the interface holds the clock line low after reception to wait for the microcontroller to read SR1 and then read the byte in the Data Register (both buffer and shift register are full).
- If clock stretching is disabled in Slave mode:
  - Overrun Error in case of RxNE=1 and no read of DR has been done before the next byte is received. The last received byte is lost.
  - Underrun Error in case TxE=1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
  - Write Collision not managed.

15.4.5 SMBus

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^{2}C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.

The System Management Bus Specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

Similarities between SMBus and I^{2}C

- 2 wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional
- Master-slave communication, Master provides clock
- Multi master capability
- SMBus data format similar to I^{2}C 7-bit addressing format (*Figure 133*).

Differences between SMBus and I^{2}C

The following table describes the differences between SMBus and I^{2}C.

Table 45. SMBus vs I^{2}C

SMBus	I^{2}C
Max. speed 100 kHz	Max. speed 400 kHz
Min. clock speed 10 kHz	No minimum clock speed
35 ms clock low time-out	No time-out
Logic levels are fixed	Logic levels are VDD dependent

Table 45. SMBus vs I^2C

SMBus	I^2C
Different address types (reserved, dynamic etc.)	7-bit, 10-bit and general call slave address types
Different bus protocols (quick command, process call etc.)	No bus protocols

SMBus application usage

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

Device identification

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

Bus protocols

The SMBus specification supports up to 9 bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*). These protocols should be implemented by the user software.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus

Unique device identifier (UDID)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).

For the details on 128 bit UDID and more information on ARP, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

SMBus alert mode

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBALERT is a wired-AND signal just as the SCL and SDA signals are. SMBALERT is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are 2 bytes long.

A slave-only device can signal the host through SMBALERT that it wants to talk by setting ALERT bit in I2C_CR1 register. The host processes the interrupt and simultaneously accesses all SMBALERT devices through the *Alert Response Address* (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBALERT low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C_SR1 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBALERT low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBALERT pull-down. If the host still sees SMBALERT low when the message transfer is complete, it knows to read the ARA again.

A host which does not implement the SMBALERT signal may periodically access the ARA.

For more details on SMBus Alert mode, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

Time-out error

There are differences in the timing specifications between I^{2}C and SMBus.

SMBus defines a clock low time-out, TIMEOUT of 35 ms. Also SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these time-outs, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

The status flag Timeout or Tlow Error in I2C_SR1 shows the status of this feature.

How to use the interface in SMBus mode

To switch from I^{2}C mode to SMBus mode, the following sequence should be performed.

- Set the SMBus bit in the I2C_CR1 register
- Configure the SMBTYPE and ENARP bits in the I2C_CR1 register as required for the application

If you want to configure the device as a master, follow the Start condition generation procedure in *Section 15.4.2: I2C master mode*. Otherwise, follow the sequence in *Section 15.4.1: I2C slave mode*.

The application has to control the various SMBus protocols by software.

- SMB Device Default Address acknowledged if ENARP=1 and SMBTYPE=0
- SMB Host Header acknowledged if ENARP=1 and SMBTYPE=1
- SMB Alert Response Address acknowledged if SMBALERT=1

15.4.6 DMA requests

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. When the number of data transfers which has been programmed for the corresponding DMA channel is reached, the DMA controller sends an End of Transfer EOT signal to the I^{2}C interface and generates a Transfer Complete interrupt if enabled:

- Master Transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the STOP condition.
- Master Receiver: The DMA controller sends a hardware signal EOT_1 corresponding to the (number of bytes -1). If, in the I2C_CR2 register, the LAST bit is set, I^{2}C automatically sends a NACK after the next byte following EOT_1. The user can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.

Note: Please refer to the product specs for availability DMA controller. If DMA is not available in the product, the user should use I^{2}C as explained in section 1.4. In the I^{2}C ISR, the user can clear TxE/ RxNE flags to achieve continuous communication.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAEN bit in the I2C_CR2 register. The DMAEN bit must be set only after receiving the address sequence, when ADDR is cleared. Data will be loaded from a Memory area configured using the DMA peripheral (refer to the DMA specification) to the I2C_DR register whenever the TxE bit is set. To map a DMA channel for I^{2}C transmission, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in the DMA_CPARx register. The data will be moved to this address from the memory after each TxE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded into I2C_DR from this memory after each TxE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each TxE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Set the DIR bit and, in the DMA_CCRx register, configure interrupts after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^{2}C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: Do not enable the ITEVTEN bit in the I2C_CR2 register if DMA is used for transmission.

Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CR2 register. The DMAEN bit must be set only after receiving the address sequence, when ADDR is cleared. Data will be loaded from the I2C_DR register to a Memory area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for I^{2}C reception, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in DMA_CPARx register. The data will be moved from this address to the memory after each RxNE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded from the I2C_DR register to this memory area after each RxNE event.

3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each RxNE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Reset the DIR bit and configure interrupts in the DMA_CCRx register after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^{2}C interface and DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: Do not enable the ITEVTEN bit in the I2C_CR2 register if DMA is used for reception.

15.4.7 Packet error checking

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using a programmable polynomial serially on each bit.

- PEC calculation is enabled by setting the ENPEC bit in the I2C_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
  – In transmission: in the last TxE event: set the PEC transfer bit in the I2C_CR1 register. The PEC will be transferred after the current byte.
  – In reception: in the last RxNE event: set the PEC bit in the I2C_CR1 register so that receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. In case of Master-Receiver, a NACK must follow the PEC whatever the check result.
- A PECERR error flag/interrupt is also available in the I2C_SR1 register.
- If DMA and PEC calculation are both enabled:-
  – In transmission: when the I^{2}C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
  – In reception: when the I^{2}C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
- To allow intermediate PEC transfers, a control bit is available in the I2C_CR2 register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted by an arbitration loss.

15.5 Interrupt requests

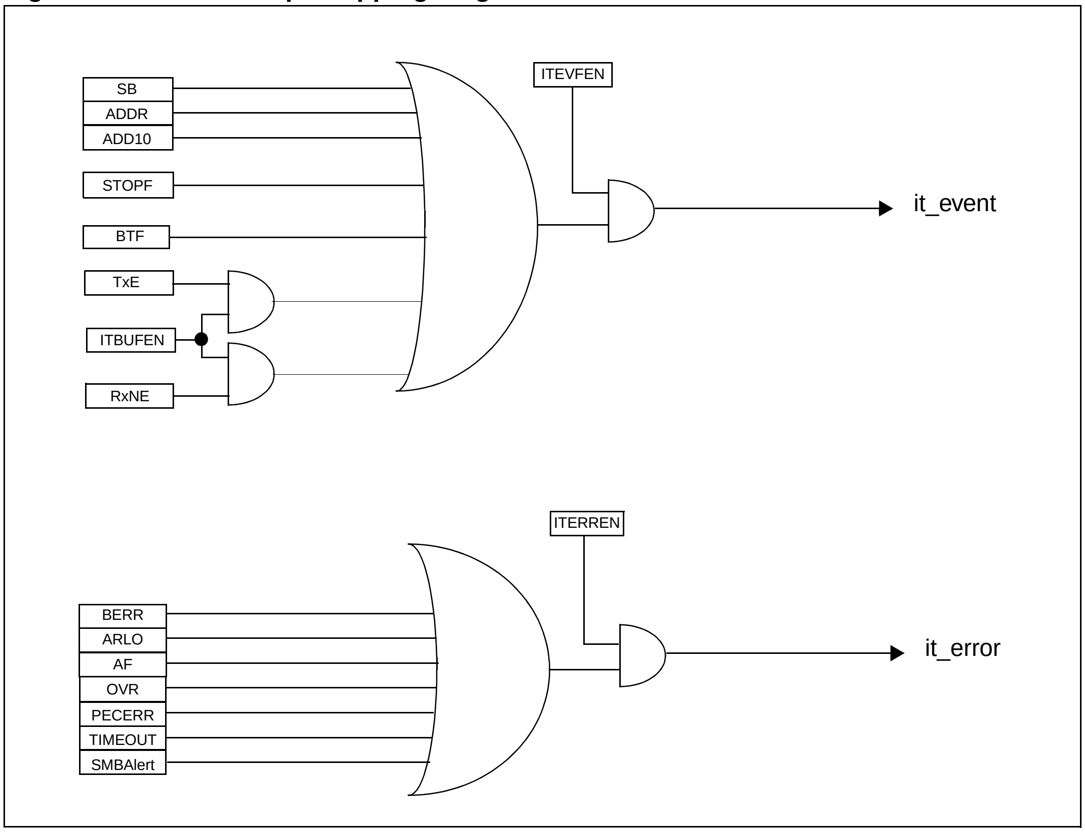
Table 46. I^{2}C Interrupt requests

Interrupt Event	Event Flag	Enable Control Bit
Start bit sent (Master)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit header sent (Master)	ADD10	
Stop received (Slave)	STOPF	
Data Byte Transfer Finished	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Transmit buffer empty	TxE	
Bus error	BERR	ITERREN
Arbitration loss (Master)	ARLO	
Acknowledge failure	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow error	TIMEOUT	
SMBus Alert	SMBALERT	

Note: 1 *SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are logically ORed on the same interrupt channel.*

2 *BERR, ARLO, AF, OVR, PECERR, TIMEOUT and SMBALERT are logically ORed on the same interrupt channel.*

Figure 139. I^{2}C interrupt mapping diagram



15.6 I2C register description

Refer to *Section 1.1 on page 23* for a list of abbreviations used in register descriptions.

15.6.1 Control register 1(I2C_CR1)

Address offset: 00h
Reset Value: 0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW RST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	EN PEC	EN ARP	SMB TYPE	Res.	SM BUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 15	**SWRST**: *Software Reset* When set, the I2C is under reset state. Before resetting this bit, make sure the I2C lines are released and the bus is free. 0: I^{2}C Peripheral not under reset 1: I^{2}C Peripheral under reset state **Note:** This bit can be used in case the BUSY bit is set to ‘1’ when no stop condition has been detected on the bus.
Bit 14	Reserved, forced by hardware to 0.
Bit 13	**ALERT**: *SMBus Alert* This bit is set and cleared by software, and cleared by hardware when PE=0. 0: Releases SMBAlert pin high. Alert Response Address Header followed by NACK. 1: Drives SMBAlert pin low. Alert Response Address Header followed by ACK.
Bit 12	**PEC**: *Packet Error Checking.* This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or STOP condition or when PE=0. 0: No PEC transfer 1: PEC transfer (in Tx or Rx mode) **Note:** PEC calculation is corrupted by an arbitration loss.
Bit 11	**POS**: *Acknowledge/PEC Position (for data reception).* This bit is set and cleared by software and cleared by hardware when PE=0. 0: ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates that current byte in shift register is a PEC. 1: ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates that the next byte in the shift register is a PEC **Note:** This bit must be configured before data reception starts. This configuration must be used only in ADDR stretch event in case there are only 2 data bytes
Bit 10	**ACK**: *Acknowledge Enable* This bit is set and cleared by software and cleared by hardware when PE=0. 0: No acknowledge returned 1: Acknowledge returned after a byte is received (matched address or data)
Bit 9	**STOP**: *Stop Generation* The 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: 0: No Stop generation. 1: Stop generation after the current byte transfer or after the current Start condition is sent. In Slave mode: 0: No Stop generation. 1: Release the SCL and SDA lines after the current byte transfer. **Note**: In Master mode, the BTF bit of the I2C_SR1 register must be cleared when STOP is requested.

Bit 8	**START**: *Start Generation* This bit is set and cleared by software and cleared by hardware when start is sent or PE=0. In Master Mode: 0: No Start generation 1: Repeated start generation In Slave mode: 0: No Start generation 1: Start generation when the bus is free
Bit 7	**NOSTRETCH**: *Clock Stretching Disable (Slave mode)* This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software. 0: Clock stretching enabled 1: Clock stretching disabled
Bit 6	**ENGC**: *General Call Enable* 0: General call disabled. Address 00h is NACKed. 1: General call enabled. Address 00h is ACKed.
Bit 5	**ENPEC**: *PEC Enable* 0: PEC calculation disabled 1: PEC calculation enabled
Bit 4	**ENARP**: *ARP Enable* 0: ARP disable 1: ARP enable SMBus Device default address recognized if SMBTYPE=0 SMBus Host address recognized if SMBTYPE=1
Bit 3	**SMBTYPE**: *SMBus Type* 0: SMBus Device 1: SMBus Host
Bit 2	Reserved, forced by hardware to 0.
Bit 1	**SMBUS**: *SMBus Mode* 0: I^2C mode 1: SMBus mode
Bit 0	**PE**: *Peripheral Enable* 0: Peripheral disable 1: Peripheral enable: the corresponding I/Os are selected as alternate functions depending on SMBus bit. **Note:** If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state. All bit resets due to PE=0 occur at the end of the communication. In master mode, this bit must not be reset before the end of the communication.

15.6.2 Control register 2 (I2C_CR2)

Address offset: 04h

Reset Value: 0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMA EN	ITBUF EN	ITEVT EN	ITER REN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 15:13	Reserved, forced by hardware to 0.
Bit 12	**LAST**: *DMA Last Transfer* 0: Next DMA EOT is not the last transfer 1: Next DMA EOT is the last transfer **Note:** This bit is used in master receiver mode to permit the generation of a NACK on the last received data.
Bit 11	**DMAEN**: *DMA Requests Enable* 0: DMA requests disabled 1: DMA request enabled when TxE=1 or RxNE =1 **Note:** The DMAEN bit must be set only after receiving the address sequence, when ADDR is cleared.
Bit 10	**ITBUFEN**: *Buffer Interrupt Enable* 0: TxE = 1 or RxNE = 1 does not generate any interrupt. 1:TxE = 1 or RxNE = 1 generates Event Interrupt (whatever the state of DMAEN)
Bit 9	**ITEVTEN**: *Event Interrupt Enable* 0: Event interrupt disabled 1: Event interrupt enabled This interrupt is generated when: – SB = 1 (Master) – ADDR = 1 (Master/Slave) – ADD10= 1 (Master) – STOPF = 1 (Slave) – BTF = 1 with no TxE or RxNE event – TxE event to 1 if ITBUFEN = 1 – RxNE event to 1if ITBUFEN = 1
Bit 8	**ITERREN**: *Error Interrupt Enable* 0: Error interrupt disabled 1: Error interrupt enabled This interrupt is generated when: – BERR = 1 – ARLO = 1 – AF = 1 – OVR = 1 – PECERR = 1 – TIMEOUT = 1 – SMBAlert = 1

Bits 7:6	Reserved, forced by hardware to 0.
Bits 5:0	**FREQ[5:0]**: *Peripheral Clock Frequency* Input clock frequency must be programmed to generate correct timings The allowed range is between 2 MHz and 50 MHz 000000: Not allowed 000001: Not allowed 000010: 2 MHz ... 110010: 50 MHz Higher than 110010: Not allowed

15.6.3 Own address register 1 (I2C_OAR1)

Reset Address offset: 08h
Value: 0000h

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

Bit 15	**ADDMODE** *Addressing Mode (Slave mode)* 0: 7-bit slave address (10-bit address not acknowledged) 1: 10-bit slave address (7-bit address not acknowledged)
Bit 14	Must be configured and kept at 1.
Bits 13:10	Reserved, forced by hardware to 0.
Bits 9:8	**ADD[9:8]**: *Interface Address* 7-bit addressing mode: don’t care 10-bit addressing mode: bits9:8 of address
Bits 7:1	**ADD[7:1]**: *Interface Address* bits 7:1 of address
Bit 0	**ADD0**: *Interface Address* 7-bit addressing mode: don’t care 10-bit addressing mode: bit 0 of address

15.6.4 Own address register 2 (I2C_OAR2)

Address offset: 0Ch
Reset Value: 0000h

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

Bits 15:8	Reserved, forced by hardware to 0.

Bits 7:1	**ADD2[7:1]**: *Interface address* bits 7:1 of address in dual addressing mode
Bit 0	**ENDUAL**: *Dual addressing mode enable* 0: Only OAR1 is recognized in 7-bit addressing mode 1: Both OAR1 and OAR2 are recognized in 7-bit addressing mode

15.6.5 Data register (I2C_DR)

Address offset: 10h
Reset Value: 0000h

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

Bits 15:8	Reserved, forced by hardware to 0.
Bits 7:0	**DR[7:0]** 8-bit *Data Register* (1)(2)(3) Byte received or to be transmitted to the bus. – Transmitter mode: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TxE=1) – Receiver mode: Received byte is copied into DR (RxNE=1). The received data in the DR register must be read before the next data reception, otherwise an overrun occurs and the last byte will be lost.

1. In slave mode, the address is not copied into DR.
2. Write collision is not managed (DR can be written if TxE=0).
3. If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.

15.6.6 Status register 1 (I2C_SR1)

Address offset: 14h
Reset Value: 0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOP F	ADD10	BTF	ADDR	SB
rc	rc		rc	rc	rc	rc	rc	r	r		r	r	r	r	r

Bit 15	**SMBALERT**: *SMBus Alert* In SMBus host mode: 0: no SMBAlert 1: SMBAlert event occurred on pin In SMBus slave mode: 0: no SMBAlert response address header 1: SMBAlert response address header to SMBAlert LOW received – Cleared by software writing 0, or by hardware when PE=0.
Bit 14	**TIMEOUT**: *Timeout or Tlow Error* 0: No time-out error 1: SCL remained LOW for 25 ms (Timeout) or Master cumulative clock low extend time more than 10 ms (Tlow:mext) or Slave cumulative clock low extend time more than 25 ms (Tlow:sext) – When set in slave mode: slave resets the communication and lines are released by hardware – When set in master mode: Stop condition sent by hardware – Cleared by software writing 0, or by hardware when PE=0.
Bit 13	Reserved, forced by hardware to 0.
Bit 12	**PECERR**: *PEC Error in reception* 0: no PEC error: receiver returns ACK after PEC reception (if ACK=1) 1: PEC error: receiver returns NACK after PEC reception (whatever ACK) – Cleared by software writing 0, or by hardware when PE=0.
Bit 11	**OVR**: *Overrun/Underrun* 0: No overrun/underrun 1: Overrun or underrun – Set by hardware in slave mode when NOSTRETCH=1 and: – In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet. New received byte is lost. – In transmission when a new byte should be sent and the DR register has not been written yet. The same byte is sent twice. – Cleared by software writing 0, or by hardware when PE=0. **Note:** If the DR write occurs very close to SCL rising edge, the sent data is unspecified and a hold timing error occurs

Bit 10	**AF**: *Acknowledge Failure.* 0: No acknowledge failure 1: Acknowledge failure – Set by hardware when no acknowledge is returned. – Cleared by software writing 0, or by hardware when PE=0.
Bit 9	**ARLO**: *Arbitration Lost (master mode)* 0: No Arbitration Lost detected 1: Arbitration Lost detected Set by hardware when the interface loses the arbitration of the bus to another master – Cleared by software writing 0, or by hardware when PE=0. After an ARLO event the interface switches back automatically to Slave mode (M/SL=0). **Note:** In SMBUS, the arbitration on the data in slave mode occurs only during the data phase, or the acknowledge transmission (not on the address acknowledge).
Bit 8	**BERR**: *Bus Error* 0: No misplaced Start or Stop condition 1: Misplaced Start or Stop condition – Set by hardware when the interface detects a misplaced Start or Stop condition – Cleared by software writing 0, or by hardware when PE=0.
Bit 7	**TxE**: *Data Register Empty (transmitters)* 0: Data register not empty 1: Data register empty – Set when DR is empty in transmission. TxE is not set during address phase. – Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0. TxE is not set if either a NACK is received, or if next byte to be transmitted is PEC (PEC=1)
Bit 6	**RxNE**: *Data Register not Empty (receivers).* 0: Data register empty 1: Data register not empty – Set when data register is not empty in receiver mode. RxNE is not set during address phase. – Cleared by software reading or writing the DR register or by hardware when PE=0. RxNE is not set in case of ARLO event.
Bit 5	Reserved, forced by hardware to 0.
Bit 4	**STOPF**: *Stop detection (Slave mode)* 0: No Stop condition detected 1: Stop condition detected – Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1). – Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0 **Note:** The STOPF bit is not set after a NACK reception

Bit 3	**ADD10**: *10-bit header sent (Master mode)* 0: No ADD10 event occurred. 1: Master has sent first address byte (header). – Set by hardware when the master has sent the first byte in 10-bit address mode. – 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. **Note:** ADD10 bit is not set after a NACK reception
Bit 2	**BTF**: *Byte Transfer Finished.* 0: Data Byte transfer not done 1: Data Byte transfer succeeded – Set by hardware when NOSTRETCH=0 and: – In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RxNE=1). – In transmission when a new byte should be sent and DR has not been written yet (TxE=1). – 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. **Note:** The BTF bit is not set after a NACK reception The BTF bit is not set if next byte to be transmitted is the PEC (TRA=1 in I2C_SR2 register and PEC=1 in I2C_CR1 register)
Bit 1	**ADDR**: *Address sent (master mode)/matched (slave mode)* This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0. **Address Matched (Slave)** 0: Address mismatched or not received. 1: Received address matched. – Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus Device Default Address or SMBus Host or SMBus Alert is recognized. (when enabled depending on configuration). **Address Sent (Master)** 0: No end of address transmission 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. **Note:** ADDR is not set after a NACK reception
Bit 0	**SB**: *Start Bit (Master mode).* 0: No Start condition 1: Start condition generated. – Set when a Start condition generated. – Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0

15.6.7 Status register 2 (I2C_SR2)

Address offset: 18h
Reset Value:0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMB DEF AULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

Bits 15:8	**PEC[7:0]** *Packet Error Checking Register* This register contains the internal PEC when ENPEC=1.
Bit 7	**DUALF**: *Dual Flag (Slave mode)* 0: Received address matched with OAR1 1: Received address matched with OAR2 – Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
Bit 6	**SMBHOST**: *SMBus Host Header (Slave mode)* 0: No SMBus Host address 1: SMBus Host address received when SMBTYPE=1 and ENARP=1. – Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
Bit 5	**SMBDEFAULT**: *SMBus Device Default Address (Slave mode)* 0: No SMBus Device Default address 1: SMBus Device Default address received when ENARP=1 – Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
Bit 4	**GENCALL**: *General Call Address(Slave mode)* 0: No General Call 1: General Call Address received when ENGC=1 – Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.
Bit 3	Reserved, forced by hardware to 0.
Bit 2	**TRA**: *Transmitter/Receiver* 0: Data bytes received 1: Data bytes transmitted This bit is set depending on R/W bit of address byte, at the end of total address phase. It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

Bit 1	**BUSY**: *Bus Busy* 0: No communication on the bus 1: Communication ongoing on the bus – Set by hardware on detection of SDA or SCL low – cleared by hardware on detection of a Stop condition. It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).
Bit 0	**MSL**: *Master/Slave* 0: Slave Mode 1: Master Mode – 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.

15.6.8 Clock control register (I2C_CCR)

Address offset: 1Ch
Reset Value: 0000h

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

Bit 15	**F/S** *I^2C Master Mode Selection* 0: Standard Mode I2C 1: Fast Mode I2C
Bit 14	**DUTY** *Fast Mode Duty Cycle* 0: Fast Mode t_{low}/t_{high} = 2 1: Fast Mode t_{low}/t_{high} = 16/9 (see CCR)
Bits 13:12	Reserved, forced by hardware to 0.
Bits 11:0	**CCR[11:0]** *Clock Control Register in Fast/Standard mode (Master mode)* Controls the SCL clock in master mode. <u>Standard Mode or SMBus</u>: T_{high} = CCR * T_{CK} T_{ow} = CCR * T_{CK} <u>Fast Mode</u>: If DUTY = 0: T_{high} = CCR * T_{CK} T_{ow} = 2 * CCR * T_{CK} If DUTY = 1: (to reach 400 kHz) T_{high} = 9 * CCR * T_{CK} T_{ow} = 16 * CCR * T_{CK} For instance: in standard mode, to generate a 100kHz SCL frequency: If FREQR = 08, T_{ck} = 125ns so CCR must be programmed with 28h (28h <=> 40d x 125ns = 5000 ns.) **Notes:** 1. The minimum allowed value is 04h, except in FAST DUTY mode where the minimum allowed value is 01h 2. t_{high} includes the SCLH rising edge 3. t_{low} includes the SCLH falling edge 4. These timings are without filters.

15.6.9 TRISE Register (I2C_TRISE)

Address offset: 20h
Reset Value: 0002h

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

Bits 15:6	Reserved, forced by hardware to 0.
Bits 5:0	**TRISE[5:0]**: *Maximum Rise Time in Fast/Standard mode (Master mode)* These bits must be programmed with the maximum SCL rise time given in the I^2C bus specification, incremented by 1. For instance: in standard mode, the maximum allowed SCL rise time is 1000 ns. If, in the I2C_CR2 register, the value of FREQ[5:0] bits is equal to 08h and t_{CK} = 125 ns therefore the TRISE[5:0] bits must be programmed with 09h. (1000 ns / 125 ns = 8 + 1) The filter value can also be added to TRISE[5:0]. If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the t_{HIGH} parameter. **Note:** TRISE[5:0] must be configured only when the I2C is disabled (PE = 0).

15.7 I^{2}C register map

Table 47. I^{2}C register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	I2C_CR1	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset Value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
04h	I2C_CR2	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved		FREQ[5:0]					
	Reset Value																				0	0	0	0	0			0	0	0	0	0	0
08h	I2C_OAR1	Reserved																ADDMODE	Reserved	Reserved				ADD[9:8]		ADD[7:1]							ADD0
	Reset Value																	0	1					0	0	0	0	0	0	0	0	0	0
0Ch	I2C_OAR2	Reserved																								ADD2[7:1]							ENDUAL
	Reset Value																									0	0	0	0	0	0	0	0
10h	I2C_DR	Reserved																								DR[7:0]							
	Reset Value																									0	0	0	0	0	0	0	0
14h	I2C_SR1	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset Value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
18h	I2C_SR2	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
1Ch	I2C_CCR	Reserved																F/S	DUTY	Reserved		CCR[11:0]											
	Reset Value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0
20h	I2C_TRISE	Reserved																										TRISE[5:0]					
	Reset Value																											0	0	0	0	1	0

Refer to *Table 1 on page 27* for the register boundary addresses.