UM10360

Chapter 19: LPC176x/5x I2C0/1/2

Rev. 4. 1 — 19 December 2016 **User manual**

19.1 Basic configuration

The $I^2C0/1/2$ interfaces are configured using the following registers:

1. Power: In the PCONP register (Table 46), set bit PCI2C0/1/2.

   Remark: On reset, all I^2C interfaces are enabled (PCI2C0/1/2 = 1).

2. Clock: In PCLKSEL0 select PCLK_I2C0; in PCLKSEL1 select PCLK_I2C1 or PCLK_I2C2 (see Section 4.7.3).

3. Pins: Select I^2C0, I^2C1, or I^2C2 pins through the PINSEL registers. Select the pin modes for the port pins with I^2C1 or I^2C2 functions through the PINMODE registers (no pull-up, no pull-down resistors) and the PINMODE_OD registers (open drain) (See Section 8.5).

   Remark: I^2C0 pins SDA0 and SCL0 are open-drain outputs and fully I^2C-bus compliant (see Table 73). I^2C0 can be further configured through the I2CPADCFG register to support Fast Mode Plus (See Table 100).

   Remark: I^2C0 is not available in the 80-pin package.

   Remark: I^2C pins that do not use specialized I^2C pads (as identified in Table 73) can be configured to an open-drain mode via the relevant IOCON registers, and can be used with fast mode (400 kHz) and standard mode (100 kHz) I^2C. These pins do not include an analog filter to suppress line glitches, but a similar function is performed by the digital filter in the I^2C block itself. These pins should be configured as: no pull-up, no pull-down, open drain mode.

4. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

5. Initialization: see Section 19.9.8.1 and Section 19.10.1.

19.2 Features

- Standard I^2C compliant bus interfaces may be configured as Master, Slave, or Master/Slave.
- Arbitration is handled between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock allows adjustment of I^2C transfer rates.
- Data transfer is bidirectional between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization is used as a handshake mechanism to suspend and resume serial transfer.
- Supports Fast Mode Plus (I^2C0 only).
- Optional recognition of up to 4 distinct slave addresses.
- Monitor mode allows observing all I^2C-bus traffic, regardless of slave address, without affecting the actual I^2C-bus traffic.
- The I^2C-bus can be used for test and diagnostic purposes.

- I^2C0 is a standard I^2C compliant bus interface with open-drain pins. This interface supports functions described in the I^2C specification for speeds up to 1 MHz (Fast Mode Plus). This includes multi-master operation and allows powering off this device in a working system while leaving the I^2C-bus functional.

19.3 Applications

Interfaces to external I^2C standard parts, such as serial RAMs, LCDs, tone generators, other microcontrollers, etc.

19.4 Description

A typical I^2C-bus configuration is shown in Figure 85. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^2C-bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte, unless the slave device is unable to accept more data.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I^2C-bus will not be released.

The LPC176x/5x I^2C interfaces are byte oriented and have four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

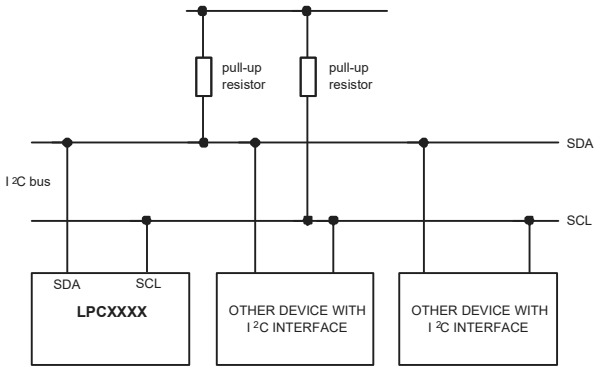


Fig 85. I²C-bus configuration

19.4.1 I²C FAST Mode Plus

Fast Mode Plus is a 1 Mbit/sec transfer rate to communicate with the I²C products which the NXP Semiconductors is now providing.

In order to use Fast Mode Plus, the I²C0 pins must be configured, then rates above 400 kHz and up to 1 Mhz may be selected, see Table 395. To configure the pins for Fast Mode Plus, the SDADRV0 and SCLDRV0 bits in the I2CPADCFG register must be set, see Section 8.5.21.

19.5 Pin description

Table 381. I²C Pin Description

Pin	Type	Description
SDA0[1]	Input/Output	I²C0 Serial Data
SCL0[1]	Input/Output	I²C0 Serial Clock
SDA1	Input/Output	I²C1 Serial Data
SCL1	Input/Output	I²C1 Serial Clock
SDA2	Input/Output	I²C2 Serial Data
SCL2	Input/Output	I²C2 Serial Clock

[1] I²C0 is only available in 100-pin LPC176x/5x devices. The SDA0 and SCL0 pins are open-drain pins to comply with I²C specifications. The pins must be configured in the I2CPADCFG register for Fast Mode Plus.

The internal logic of the 3 I²C interfaces is identical. These interfaces can be brought out to device pins in several ways, some of which have different pin I/O characteristics. I2C0 on pins P0[27] and P0[28] use specialized I²C pads that support fully spec compliant fast mode, standard mode, and fast mode plus I²C.

Any of the I²C interfaces brought out to pins other than those just mentioned use standard I/O pins. These pins also support I²C operation in fast mode and standard mode. The primary difference is that these pins do not include an analog spike suppression filter that exists on the specialized I²C pads. The I²C interfaces all include a digital filter that can serve the same purpose.

19.6 I²C operating modes

In a given application, the I²C block may operate as a master, a slave, or both. In the slave mode, the I²C hardware looks for any one of its four slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I²C block switches to the slave mode immediately and can detect any of its own configured slave addresses in the same serial transfer.

19.6.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the I2CONSET register must be initialized as shown in Table 382. I2EN must be set to 1 to enable the I²C function. If the AA bit is 0, the I²C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI bit is cleared by writing 1 to the SIC bit in the I2CONCLR register. THe STA bit should be cleared after writing the slave address.

Table 382. I2C0CONSET, I2C1CONSET and I2C2CONSET used to configure Master mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	0	-	-

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I²C interface will enter master transmitter mode when software sets the STA bit. The I²C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the I2STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the I2DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 0x18, 0x20, or 0x38 for the master mode, or 0x68, 0x78, or 0xB0 if the slave mode was enabled (by setting AA to 1). The appropriate actions to be taken for each of these status codes are shown in Table 399 to Table 402.

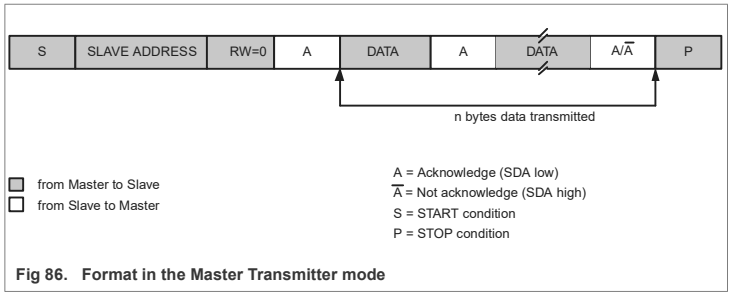


Fig 86. Format in the Master Transmitter mode

19.6.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^2C Data register (I2DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 400.

When the LPC176x/5x needs to acknowledge a received byte, the AA bit needs to be set accordingly prior to clearing the SI bit and initiating the byte read. When the LPC176x/5x needs to not acknowledge a received byte, the AA bit needs to be cleared prior to clearing the SI bit and initiating the byte read.

Note that the last received byte is always followed by a "Not Acknowledge" from the LPC176x/5x so that the master can signal the slave that the reading sequence is finished and that it needs to issue a STOP or repeated START Command. Once the "Not Acknowledge has been sent and the SI bit is set, the LPC176x/5x can send either a STOP (STO bit is set) or a repeated START (STA bit is set). Then the SI bit is cleared to initiate the requested operation.

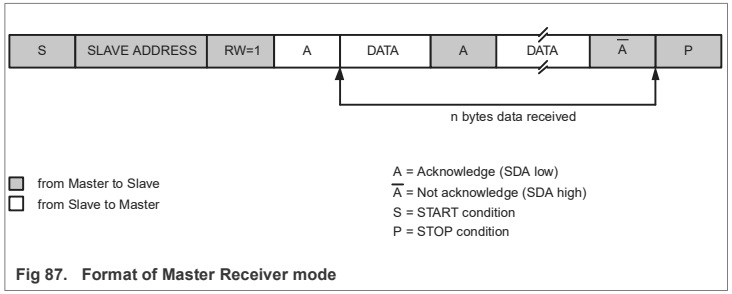


Fig 87. Format of Master Receiver mode

After a repeated START condition, I^2C may switch to the master transmitter mode.

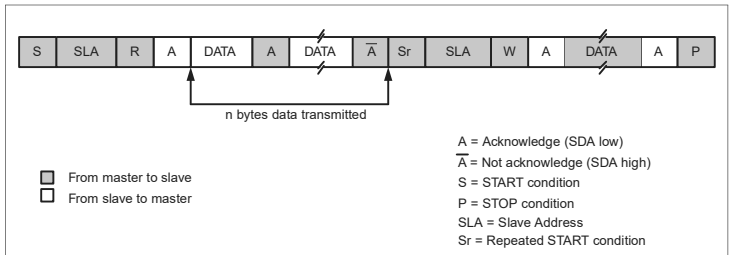


Fig 88. A Master Receiver switches to Master Transmitter after sending repeated START

19.6.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, write any of the Slave Address registers (I2ADR0-3) and Slave Mask registers (I2MASK0-3) and write the I^2C Control Set register (I2CONSET) as shown in Table 383.

Table 383. I2C0CONSET, I2C1CONSET and I2C2CONSET used to configure Slave mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

I2EN must be set to 1 to enable the I^2C function. AA bit must be set to 1 to acknowledge any of its own slave addresses or the General Call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I^2C interface waits until it is addressed by its any of its own slave addresses or General Call address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status register (I2STAT). Refer to Table 401 for the status codes and actions.

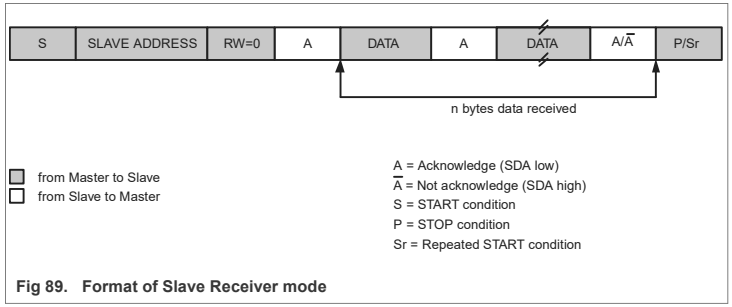


Fig 89. Format of Slave Receiver mode

19.6.4 Slave Transmitter mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for any of its own slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^2C interface switches to the slave mode immediately and can detect any of its own slave addresses in the same serial transfer.

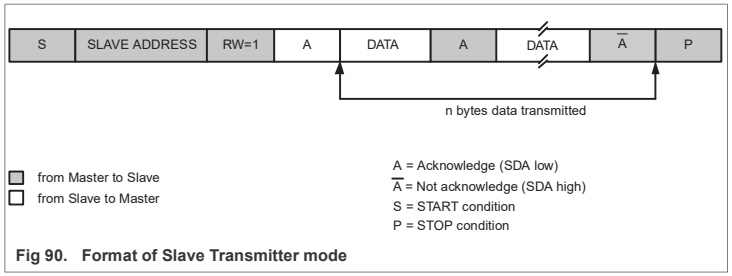


Fig 90. Format of Slave Transmitter mode

19.7 I^2C implementation and operation

Figure 91 shows how the on-chip I^2C-bus interface is implemented, and the following text describes the individual blocks.

19.7.1 Input filters and output stages

Input signals are synchronized with the internal clock, and spikes shorter than three clocks are filtered out.

The output for I^2C is a special pad designed to conform to the I^2C specification.

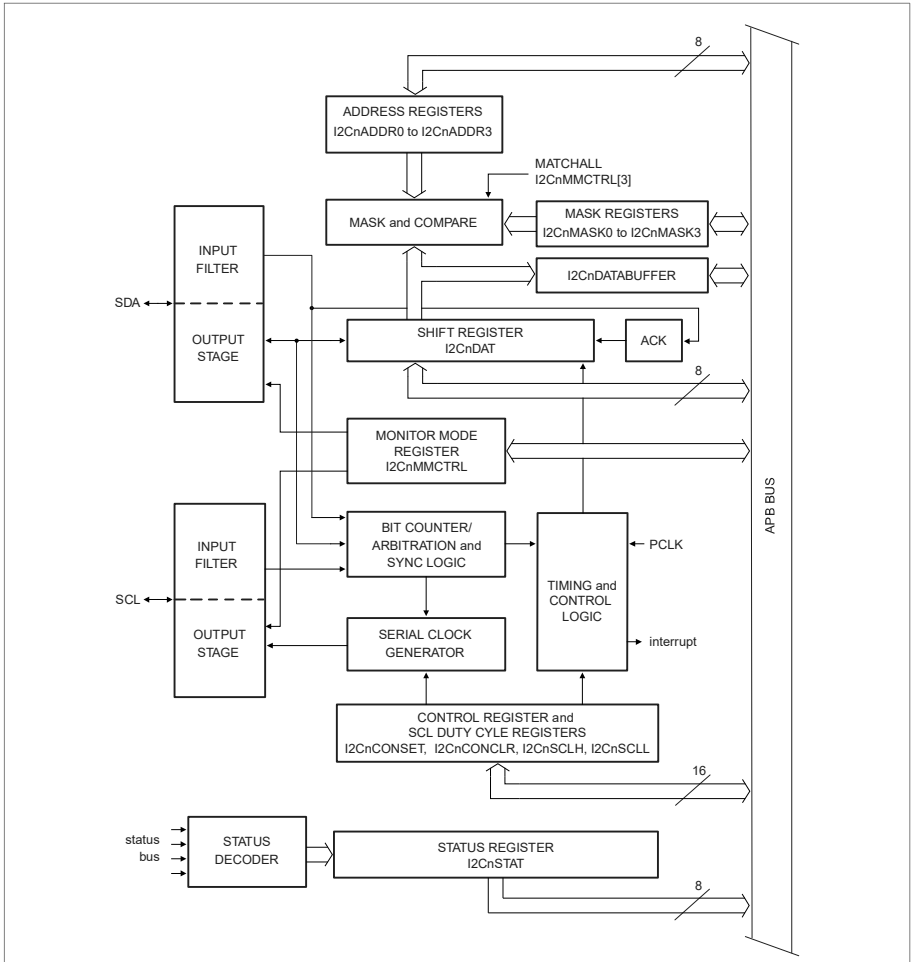


Fig 91. I^2C serial interface block diagram

19.7.2 Address Registers, I2ADR0 to I2ADR3

These registers may be loaded with the 7-bit slave address (7 most significant bits) to which the I^2C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable General Call address (0x00) recognition. When multiple slave addresses are enabled, the actual address received may be read from the I2DAT register at the state where the “own slave address” has just been received.

Remark: in the remainder of this chapter, when the phrase “own slave address” is used, it refers to any of the four configured slave addresses after address masking.

19.7.3 Address mask registers, I2MASK0 to I2MASK3

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADRn register associated with that mask register. In other words, bits in an I2ADRn register which are masked are not taken into account in determining an address match.

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine which received address actually caused the match.

19.7.4 Comparator

The comparator compares the received 7-bit slave address with any of the four configured slave addresses in I2ADR0 through I2ADR3 after masking. It also compares the first received 8-bit byte with the General Call address (0x00). If an a match is found, the appropriate status bits are set and an interrupt is requested.

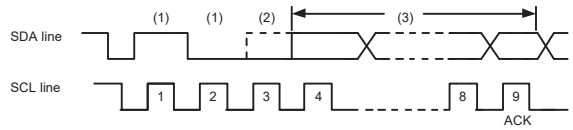
19.7.5 Shift register, I2DAT

This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in I2DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in I2DAT.

19.7.6 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^2C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^2C block immediately changes from master transmitter to slave receiver. The I^2C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^2C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal low. Since this can occur only at the end of a serial byte, the I^2C block generates no further clock pulses. Figure 92 shows the arbitration procedure.



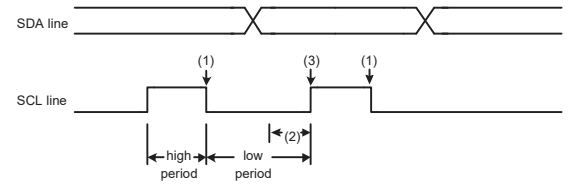
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I^2C master by pulling the SDA line low. Arbitration is lost, and this I^2C enters Slave Receiver mode.

(3) This I^2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I^2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 92. Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 93 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I^2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I^2C has timed a complete low time and released SCL. The I^2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 93. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I^2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

19.7.7 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I^2C block is in the master transmitter or master receiver mode. It is switched off when the I^2C block is in a slave mode. The I^2C output clock frequency and duty cycle is programmable

via the I²C Clock Control Registers. See the description of the I2CSCLL and I2CSCLH registers for details. The output clock pulses have a duty cycle as programmed unless the bus is synchronizing with other SCL clock sources as described above.

19.7.8 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for I2DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I²C-bus status.

19.7.9 Control register, I2CONSET and I2CONCLR

The I²C control register contains bits used to control the following I²C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I²C control register may be read as I2CONSET. Writing to I2CONSET will set bits in the I²C control register that correspond to ones in the value written. Conversely, writing to I2CONCLR will clear bits in the I²C control register that correspond to ones in the value written.

19.7.10 Status decoder and status register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I²C-bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I²C block are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

19.8 Register description

Each I²C interface contains 16 registers as shown in Table 384 below.

Remark: In the LPC176x/5x, the following registers have been added to support response to multiple addresses in Slave mode and a new Monitor mode: I2ADR1 to 3, I2MASK0 To 3, MMCTRL, and I2DATA_BUFFER.

Table 384. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Name & Address
I2CONSET	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is set. Writing a zero has no effect on the corresponding bit in the I²C control register.	R/W	0x00	I2C0CONSET - 0x4001 C000 I2C1CONSET - 0x4005 C000 I2C2CONSET - 0x400A 0000
I2STAT	**I2C Status Register.** During I²C operation, this register provides detailed status codes that allow software to determine the next action needed.	RO	0xF8	I2C0STAT - 0x4001 C004 I2C1STAT - 0x4005 C004 I2C2STAT - 0x400A 0004
I2DAT	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	R/W	0x00	I2C0DAT - 0x4001 C008 I2C1DAT - 0x4005 C008 I2C2DAT - 0x400A 0008
I2ADR0	**I2C Slave Address Register 0.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR0 - 0x4001 C00C I2C1ADR0 - 0x4005 C00C I2C2ADR0 - 0x400A 000C
I2SCLH	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I²C clock.	R/W	0x04	I2C0SCLH - 0x4001 C010 I2C1SCLH - 0x4005 C010 I2C2SCLH - 0x400A 0010
I2SCLL	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I²C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I²C master and certain times used in slave mode.	R/W	0x04	I2C0SCLL - 0x4001 C014 I2C1SCLL - 0x4005 C014 I2C2SCLL - 0x400A 0014
I2CONCLR	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is cleared. Writing a zero has no effect on the corresponding bit in the I²C control register.	WO	NA	I2C0CONCLR - 0x4001 C018 I2C1CONCLR - 0x4005 C018 I2C2CONCLR - 0x400A 0018
MMCTRL	**Monitor mode control register.**	R/W	0x00	I2C0MMCTRL - 0x4001 C01C I2C1MMCTRL - 0x4005 C01C I2C2MMCTRL - 0x400A 001C
I2ADR1	**I2C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR1 - 0x4001 C020 I2C1ADR1 - 0x4005 C020 I2C2ADR1 - 0x400A 0020

Table 384. I²C register map

Generic Name	Description	Access	Reset value[1]	I²Cn Name & Address
I2ADR2	**I2C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR2 - 0x4001 C024 I2C1ADR2 - 0x4005 C024 I2C2ADR2 - 0x400A 0024
I2ADR3	**I2C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	R/W	0x00	I2C0ADR3 - 0x4001 C028 I2C1ADR3 - 0x4005 C028 I2C2ADR3 - 0x400A 0028
I2DATA_ BUFFER	**Data buffer register.** The contents of the 8 MSBs of the I2DAT shift register will be transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus.	RO	0x00	I2C0DATA_ BUFFER - 0x4001 C02C I2C1DATA_ BUFFER - 0x4005 C02C I2C2DATA_ BUFFER - 0x400A 002C
I2MASK0	**I2C Slave address mask register 0**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK0 - 0x4001 C030 I2C1MASK0 - 0x4005 C030 I2C2MASK0 - 0x400A 0030
I2MASK1	**I2C Slave address mask register 1**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK1 - 0x4001 C034 I2C1MASK1 - 0x4005 C034 I2C2MASK1 - 0x400A 0034
I2MASK2	**I2C Slave address mask register 2**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK2 - 0x4001 C038 I2C1MASK2 - 0x4005 C038 I2C2MASK2 - 0x400A 0038
I2MASK3	**I2C Slave address mask register 3**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	R/W	0x00	I2C0MASK3 - 0x4001 C03C I2C1MASK3 - 0x4005 C03C I2C2MASK3 - 0x400A 003C

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

19.8.1 I²C Control Set register (I2CONSET: I²C0, I2C0CONSET - 0x4001 C000; I²C1, I2C1CONSET - 0x4005 C000; I²C2, I2C2CONSET - 0x400A 0000)

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I²C interface. Writing a one to a bit of this register causes the corresponding bit in the I²C control register to be set. Writing a zero has no effect. Reading this register provides the current values of the control and flag bits.

Table 385. I²C Control Set register (I2CONSET: I²C0, I2C0CONSET - address 0x4001 C000, I²C1, I2C1CONSET - address 0x4005 C000, I²C2, I2C2CONSET - address 0x400A 0000) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag.	0
3	SI	I²C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I²C interface enable.	0
31:7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I2EN I²C Interface Enable. When I2EN is 1, the I²C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I²C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I²C block is in the “not addressed” slave state, and the STO bit is forced to “0”.

I2EN should not be used to temporarily release the I²C-bus since, when I2EN is reset, the I²C-bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I²C interface to enter master mode and transmit a START condition or transmit a repeated START condition if it is already in master mode.

When STA is 1 and the I²C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I²C interface is already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I²C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the I2CONCLR register. When STA is 0, no START condition or repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I²C-bus if the interface is in master mode, and transmits a START condition thereafter. If the I²C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

STO is the STOP flag. Setting this bit causes the I²C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I²C-bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I^2C Interrupt Flag. This bit is set when the I^2C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is HIGH, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in I2CONCLR register. The SI bit should be cleared only after the required bit(s) has (have) been set and the value in I2DAT has been loaded or read.

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A matching address defined by registers I2ADR0 through I2ADR3, masked by I2MASK0 though I2MASK3, has been received.
2. The General Call address has been received while the General Call bit (GC) in I2ADR is set.
3. A data byte has been received while the I^2C is in the master receiver mode.
4. A data byte has been received while the I^2C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (HIGH level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I^2C is in the master receiver mode.
2. A data byte has been received while the I^2C is in the addressed slave receiver mode.

19.8.2 I^2C Control Clear register (I2CONCLR: I^2C0, I2C0CONCLR - 0x4001 C018; I^2C1, I2C1CONCLR - 0x4005 C018; I^2C2, I2C2CONCLR - 0x400A 0018)

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I^2C interface. Writing a one to a bit of this register causes the corresponding bit in the I^2C control register to be cleared. Writing a zero has no effect. I2CONCLR is a write-only register. The value of the related bits can be read from the I2CONSET register.

Table 386. I^2C Control Clear register (I2CONCLR: I^2C0, I2C0CONCLR - 0x4001 C018; I^2C1, I2C1CONCLR - 0x4005 C018; I^2C2, I2C2CONCLR - 0x400A 0018) bit description

Bit	Symbol	Description
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.
2	AAC	Assert acknowledge Clear bit.
3	SIC	I^2C interrupt Clear bit.

Table 386. I^2C Control Clear register (I2CONCLR: I^2C0, I2C0CONCLR - 0x4001 C018; I^2C1, I2C1CONCLR - 0x4005 C018; I^2C2, I2C2CONCLR - 0x400A 0018) bit description *…continued*

Bit	Symbol	Description
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.
5	STAC	START flag Clear bit.
6	I2ENC	I^2C interface Disable bit.
31:7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.

AAC is the Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.

SIC is the I^2C Interrupt Clear bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.

STAC is the START flag Clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.

I2ENC is the I^2C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.

19.8.3 I^2C Status register (I2STAT: I^2C0, I2C0STAT - 0x4001 C004; I^2C1, I2C1STAT - 0x4005 C004; I^2C2, I2C2STAT - 0x400A 0004)

Each I^2C Status register reflects the condition of the corresponding I^2C interface. The I^2C Status register is read-only.

Table 387. I^2C Status register (I2STAT: I^2C0, I2C0STAT - 0x4001 C004; I^2C1, I2C1STAT - 0x4005 C004; I^2C2, I2C2STAT - 0x400A 0004) bit description

Bit	Symbol	Description	Reset value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I^2C interface.	0x1F
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I^2C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 399 to Table 402.

19.8.4 I^2C Data register (I2DAT: I^2C0, I2C0DAT - 0x4001 C008; I^2C1, I2C1DAT - 0x4005 C008; I^2C2, I2C2DAT - 0x400A 0008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

Table 388. I²C Data register (I2DAT: I²C0, I2C0DAT - 0x4001 C008; I²C1, I2C1DAT - 0x4005 C008; I²C2, I2C2DAT - 0x400A 0008) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds data values that have been received or are to be transmitted.	0
31:8	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.8.5 I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C)

This register controls the Monitor mode which allows the I²C module to monitor traffic on the I²C-bus without actually participating in traffic or interfering with the I²C-bus.

Table 389. I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C) bit description

Bit	Symbol	Value	Description	Reset value
0	MM_ENA		Monitor mode enable.	0
		0	Monitor mode disabled.	
		1	The I²C module will enter monitor mode. In this mode the SDA output will be put in high impedance mode. This prevents the I²C module from outputting data of any kind (including ACK) onto the I²C data bus. Depending on the state of the ENA_SCL bit, the output may be also forced high, preventing the module from having control over the I²C clock line.	
1	ENA_SCL		SCL output enable.	0
		0	When this bit is cleared to '0', the SCL output will be forced high when the module is in monitor mode. As described above, this will prevent the module from having any control over the I²C clock line.	
		1	When this bit is set, the I²C module may exercise the same control over the clock line that it would in normal operation. This means that, acting as a slave peripheral, the I²C module can "stretch" the clock line (hold it low) until it has had time to respond to an I²C interrupt.[1]	

Table 389. I²C Monitor mode control register (I2MMCTRL: I²C0, I2C0MMCTRL - 0x4001 C01C; I²C1, I2C1MMCTRL- 0x4005 C01C; I²C2, I2C2MMCTRL- 0x400A 001C) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
2	MATCH_ALL		Select interrupt register match.	0
		0	When this bit is cleared, an interrupt will only be generated when a match occurs to one of the (up-to) four address registers, I2ADR0 through I2ADR3. That is, the module will respond as a normal slave as far as address-recognition is concerned.	
		1	When this bit is set to '1' and the I²C is in monitor mode, an interrupt will be generated on ANY address received. This will enable the part to monitor all traffic on the bus.	
31:3	-		Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

[1] When the ENA_SCL bit is cleared and the I²C no longer has the ability to stretch the clock, interrupt response time becomes important. To give the part more time to respond to an I²C interrupt under these conditions, an I2DATA_BUFFER register is used (Section 19.8.6) to hold received data for a full 9-bit word transmission time.

Remark: The ENA_SCL and MATCH_ALL bits have no effect if the MM_ENA is '0' (i.e. if the module is NOT in monitor mode).

19.8.5.1 Interrupt in Monitor mode

All interrupts will occur as normal when the module is in monitor mode. This means that the first interrupt will occur when an address-match is detected (any address received if the MATCH_ALL bit is set, otherwise an address matching one of the four address registers).

Subsequent to an address-match detection, interrupts will be generated after each data byte is received for a slave-write transfer, or after each byte that the module believes it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

19.8.5.2 Loss of arbitration in Monitor mode

In monitor mode, the I²C module will not be able to respond to a request for information by the bus master or issue an ACK. Some other slave on the bus will respond instead.

Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected.

19.8.6 I²C Data buffer register (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER - 0x4001 C02C; I²C1, I2C1DATA_BUFFER- 0x4005 C02C; I²C2, I2C2DATA_BUFFER- 0x400A 002C)

In monitor mode, the I²C module may lose the ability to stretch the clock if the ENA_SCL bit is not set. This means that the processor will have a limited amount of time to read the contents of the data received on the bus. If the processor reads the I2DAT shift register, as it ordinarily would, it could have only one bit-time to respond to the interrupt before the received data is overwritten by new data.

To give the processor more time to respond, a new 8-bit, read-only I2DATA_BUFFER register will be added. The contents of the 8 MSBs of the I2DAT shift register will be transferred to the I2DATA_BUFFER automatically after every 9 bits (8 bits of data plus ACK or NACK) has been received on the bus. This means that the processor will have 9 bit transmission times to respond to the interrupt and read the data before it is overwritten.

The processor will still have the ability to read I2DAT directly, as usual, and the behavior of I2DAT will not be altered in any way.

Although the I2DATA_BUFFER register is primarily intended for use in monitor mode with the ENA_SCL bit = '0', it will be available for reading at any time under any mode of operation.

Table 390. I²C Data buffer register (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER - 0x4001 C02C; I²C1, I2C1DATA_BUFFER- 0x4005 C02C; I²C2, I2C2DATA_BUFFER- 0x400A 002C) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds contents of the 8 MSBs of the I2DAT shift register.	0
31:8	-	Reserved. The value read from a reserved bit is not defined.	NA

19.8.7 I²C Slave Address registers (I2ADR0 to 3: I²C0, I2C0ADR[0, 1, 2, 3]- 0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3] - address 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3] - address 0x400A 00[0C, 20, 24, 28])

These registers are readable and writable and are only used when an I²C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

If these registers contain 0x00, the I²C will not acknowledge any address on the bus. All four registers will be cleared to this disabled state on reset.

Table 391. I²C Slave Address registers (I2ADR0 to 3: I²C0, I2C0ADR[0, 1, 2, 3]- 0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3] - address 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3] - address 0x400A 00[0C, 20, 24, 28]) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I²C device address for slave mode.	0x00
31:8	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

19.8.8 I²C Mask registers (I2MASK0 to 3: I²C0, I2C0MASK[0, 1, 2, 3] - 0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3] - address 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3] - address 0x400A 00[30, 34, 38, 3C])

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to '1' will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADRn register associated with that mask register. In other words, bits in an I2ADRn register which are masked are not taken into account in determining an address match.

The mask register has no effect on comparison to the General Call address ("0000000").

When an address-match interrupt occurs, the processor will have to read the data register (I2DAT) to determine which received address actually caused the match.

Table 392. I²C Mask registers (I2MASK0 to 3: I²C0, I2C0MASK[0, 1, 2, 3] - 0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3] - address 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3] - address 0x400A 00[30, 34, 38, 3C]) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. User software should not write ones to reserved bits. This bit reads always back as 0.	0
7:1	MASK	Mask bits.	0x00
31:8	-	Reserved. User software should not write ones to reserved bits. These bits read always back as zeroes.	0

19.8.9 I²C SCL HIGH duty cycle register (I2SCLH: I²C0, I2C0SCLH - 0x4001 C010; I²C1, I2C1SCLH - 0x4005 C010; I²C2, I2C2SCLH - 0x400A 0010)

Table 393. I²C SCL HIGH Duty Cycle register (I2SCLH: I²C0, I2C0SCLH - address 0x4001 C010; I²C1, I2C1SCLH - address 0x4005 C010; I²C2, I2C2SCLH - 0x400A 0010) bit description

Bit	Symbol	Description	Reset value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	NA

19.8.10 I²C SCL Low duty cycle register (I2SCLL: I²C0 - I2C0SCLL: 0x4001 C014; I²C1 - I2C1SCLL: 0x4005 C014; I²C2 - I2C2SCLL: 0x400A 0014)

Table 394. I²C SCL Low duty cycle register (I2SCLL: I²C0 - I2C0SCLL: 0x4001 C014; I²C1 - I2C1SCLL: 0x4005 C014; I²C2 - I2C2SCLL: 0x400A 0014) bit description

Bit	Symbol	Description	Reset value
15:0	SCLL	Count for SCL low time period selection.	0x0004
31:16	-	Reserved. The value read from a reserved bit is not defined.	NA

19.8.11 Selecting the appropriate I²C data rate and duty cycle

Software must set values for the registers I2SCLH and I2SCLL to select the appropriate data rate and duty cycle. I2SCLH defines the number of PCLK_I2C cycles for the SCL HIGH time, I2SCLL defines the number of PCLK_I2C cycles for the SCL low time. The frequency is determined by the following formula (PCLK_I2C is the frequency of the peripheral bus APB):

$$I^2C_{bitfrequency} = \frac{PCLKI2C}{I2CSCLH + I2CSCLL} \quad (13)$$

The values for I2SCLL and I2SCLH must ensure that the data rate is in the appropriate I²C data rate range. Each register value must be greater than or equal to 4. Table 395 gives some examples of I²C-bus rates based on PCLK_I2C frequency and I2SCLL and I2SCLH values.

Table 395. Example I²C clock rates

I²C Rate	I2SCLL + I2SCLH values at PCLK_I2C (MHz)													
	6	8	10	12	16	20	30	40	50	60	70	80	90	100
100 kHz (Standard)	60	80	100	120	160	200	300	400	500	600	700	800	900	1000
400 kHz (Fast Mode)	15	20	25	30	40	50	75	100	125	150	175	200	225	250
1 MHz (Fast Mode Plus)	-	8	10	12	16	20	30	40	50	60	70	80	90	100

I2SCLL and I2SCLH values should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I²C-bus specification defines the SCL low time and high time at different values for a Fast Mode and Fast Mode Plus I²C.

19.9 Details of I²C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figure 94, Figure 95, Figure 96, Figure 97, and Figure 98. Table 396 lists abbreviations used in these figures when describing the I²C operating modes.

Table 396. Abbreviations used to describe an I²C operation

Abbreviation	Explanation
S	START condition
SLA	7-bit slave address
R	Read bit (HIGH level at SDA)
W	Write bit (LOW level at SDA)
A	Acknowledge bit (LOW level at SDA)
$\overline{A}$	Not acknowledge bit (HIGH level at SDA)
Data	8-bit data byte
P	STOP condition
Sr	Repeated START condition

In Figure 94 to Figure 98, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the I2STAT register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in I2STAT is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in tables from Table 399 to Table 403.

19.9.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 94). Before the master transmitter mode can be entered, I2CON must be initialized as follows:

Table 397. I2CONSET used to initialize Master Transmitter mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	x	-	-

The I^2C rate must also be configured in the I2SCLL and I2SCLH registers. I2EN must be set to logic 1 to enable the I^2C block. If the AA bit is reset, the I^2C block will not acknowledge its own slave address or the General Call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I^2C interface cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I^2C logic will now test the I^2C-bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (I2STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads I2DAT with the slave address and the data direction bit (SLA+W). The SI bit in I2CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 399. After a repeated START condition (state 0x10). The I^2C block may switch to the master receiver mode by loading I2DAT with SLA+R).

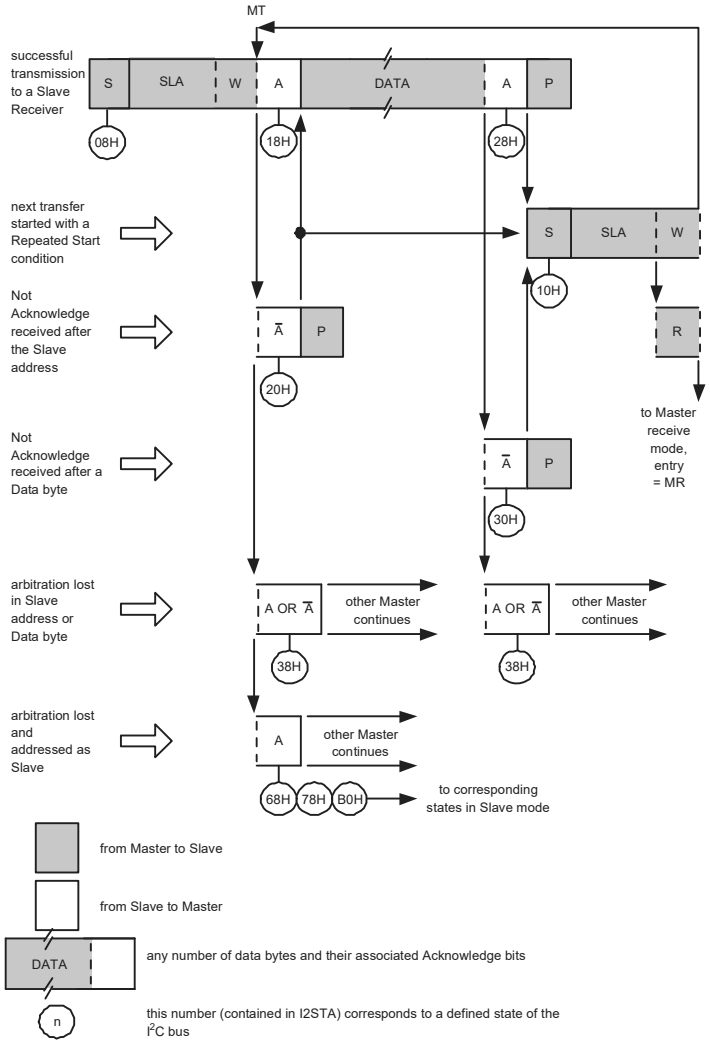


Fig 94. Format and states in the Master Transmitter mode

19.9.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 95). The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load I2DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in I2CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in I2STAT are possible. These are 0x40, 0x48, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 400. After a repeated START condition (state 0x10), the I^2C block may switch to the master transmitter mode by loading I2DAT with SLA+W.

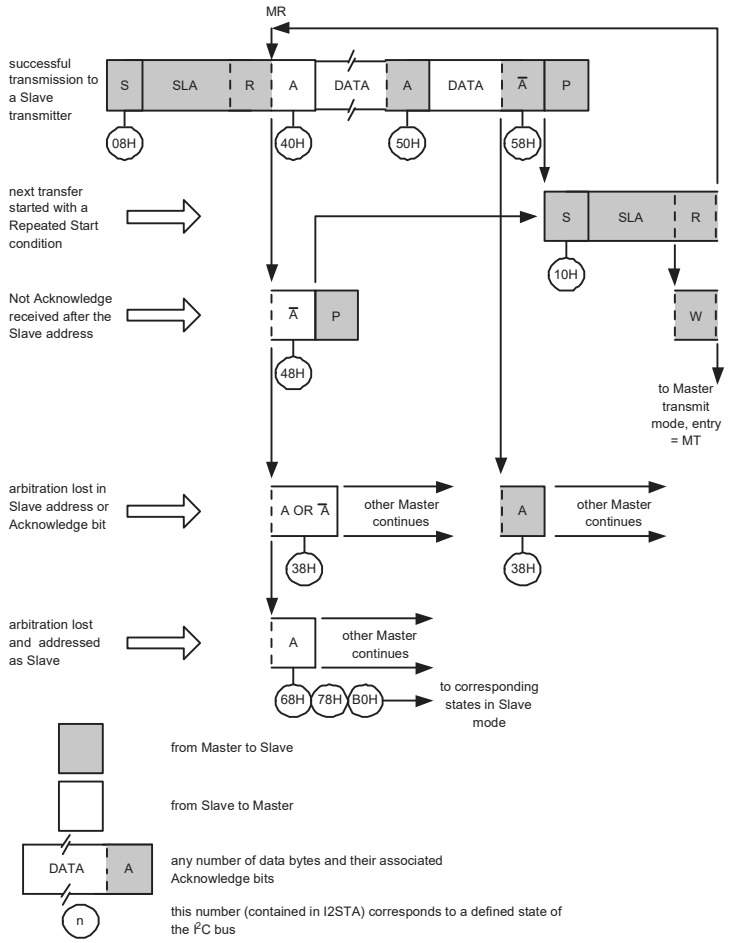


Fig 95. Format and states in the Master Receiver mode

19.9.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 96). To initiate the slave receiver mode, I2CON register, the I2ADR registers, and the I2MASK registers must be configured.

The values on the four I2ADR registers combined with the values on the four I2MASK registers determines which address(es) the I^2C block will respond to when slave functions are enabled. See sections 19.7.2, 19.7.3, 19.8.7, and 19.8.8 for details.

Table 398. I2CONSET used to initialize Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

The I^2C-bus rate settings do not affect the I^2C block in the slave mode. I2EN must be set to logic 1 to enable the I^2C block. The AA bit must be set to enable the I^2C block to acknowledge its own slave address or the General Call address. STA, STO, and SI must be reset.

When the I2ADR, I2MASK, and I2CON registers have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^2C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 401. The slave receiver mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^2C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

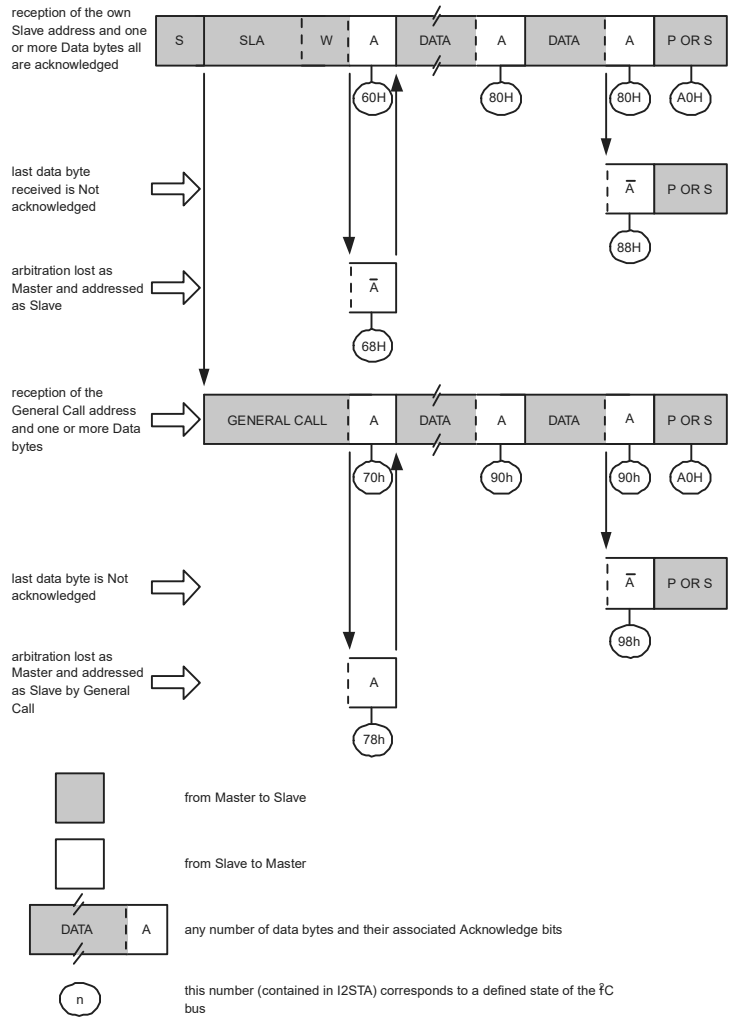


Fig 96. Format and states in the Slave Receiver mode

19.9.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 97). Data transfer is initialized as in the slave receiver mode. When I2ADR and I2CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “1” (R) for the I^2C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from I2STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 402. The slave transmitter mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I^2C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^2C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

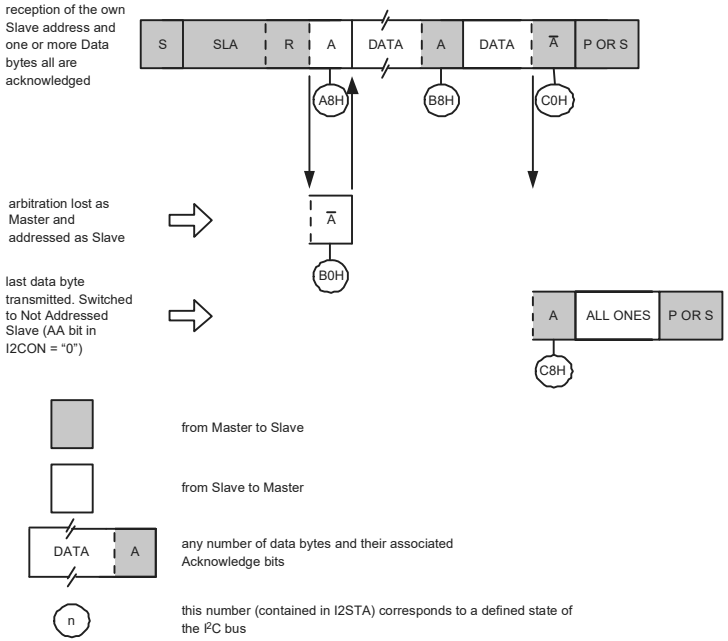


Fig 97. Format and states in the Slave Transmitter mode

19.9.5 Detailed state tables

The following tables show detailed state information for the four I^2C operating modes.

Table 399. Master Transmitter mode

I2CSTAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W; clear STA	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R; Clear STA	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in I2DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No I2DAT action or	1	0	0	X	Repeated START will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No I2DAT action or	0	0	0	X	I^2C-bus will be released; not addressed slave will be entered.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

Table 400. Master Receiver mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I²C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No I2DAT action or	0	0	0	X	I²C-bus will be released; the I²C block will enter a slave mode.
		No I2DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No I2DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No I2DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No I2DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No I2DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No I2DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 401. Slave Receiver mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General Call address (0x00) has been received; ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General Call address has been received, ACK has been returned.	No I2DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No I2DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLA address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Table 401. Slave Receiver mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave Receiver or Slave Transmitter.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

Table 402. Slave Transmitter mode

I2CSTAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received, ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in I2DAT has been transmitted; ACK has been received.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xC0	Data byte in I2DAT has been transmitted; NOT ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No I2DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in I2DAT has been transmitted (AA = 0); ACK has been received.	No I2DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No I2DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR[0] = logic 1.
		No I2DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No I2DAT action	1	0	0	01	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if I2ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

19.9.6 Miscellaneous states

There are two I2STAT codes that do not correspond to a defined I²C hardware state (see Table 403). These are discussed below.

19.9.6.1 I2STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I²C block is not involved in a serial transfer.

19.9.6.2 I2STAT = 0x00

This status code indicates that a bus error has occurred during an I²C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I²C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I²C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in I2CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 403. Miscellaneous States

Status Code (I2CSTAT)	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From I2DAT	To I2CON				
			STA	STO	SI	AA	
0xF8	No relevant state information available; SI = 0.	No I2DAT action	No I2CON action				Wait or proceed current transfer.
0x00	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 0x00 can also occur when interference causes the I²C block to enter an undefined state.	No I2DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the I²C block is switched to the not addressed SLV mode. STO is reset.

19.9.7 Some special cases

The I²C hardware has facilities to handle the following special cases that may occur during a serial transfer:

19.9.7.1 Simultaneous repeated START conditions from two masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 98). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I²C hardware detects a repeated START condition on the I²C-bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, the I²C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

19.9.7.2 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 92). Loss of arbitration is indicated by the following states in I2STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 94 and Figure 95).

If the STA flag in I2CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 0x08) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

19.9.7.3 Forced access to the I²C-bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I²C-bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I²C-bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I²C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware Figure 99.

19.9.7.4 I²C-bus obstructed by a LOW level on SCL or SDA

An I²C-bus hang-up can occur if either the SDA or SCL line is held LOW by any device on the bus. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the problem must be resolved by the device that is pulling the SCL bus line LOW.

Typically, the SDA line may be obstructed by another device on the bus that has become out of synchronization with the current bus master by either missing a clock, or by sensing a noise pulse as a clock. In this case, the problem can be solved by transmitting additional clock pulses on the SCL line Figure 100. The I²C interface does not include a dedicated time-out timer to detect an obstructed bus, but this can be implemented using another timer in the system. When detected, software can force clocks (up to 9 may be required) on SCL until SDA is released by the offending device. At that point, the slave may still be out of synchronization, so a START should be generated to insure that all I²C peripherals are synchronized.

19.9.7.5 Bus error

A bus error occurs when a START or STOP condition is detected at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I^2C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I^2C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 403.

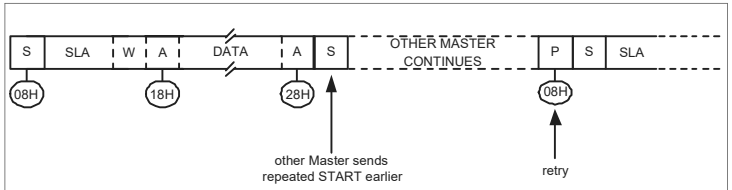


Fig 98. Simultaneous repeated START conditions from two masters

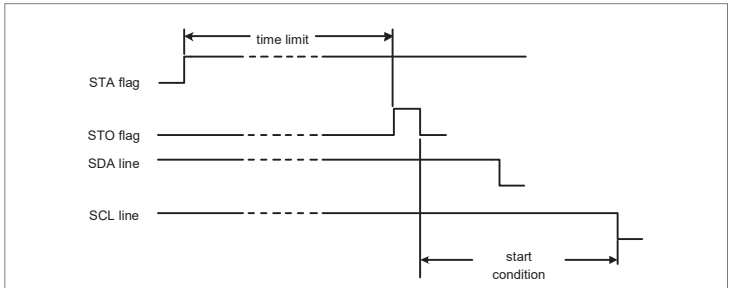
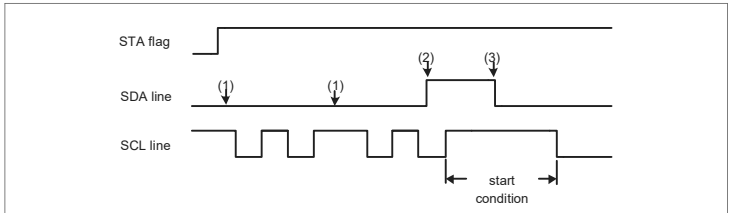


Fig 99. Forced access to a busy I^2C-bus



(1) Unsuccessful attempt to send a START condition.
(2) SDA line is released.
(3) Successful attempt to send a START condition. State 08H is entered.

Fig 100. Recovering from a bus obstruction caused by a LOW level on SDA

19.9.8 I^2C state service routines

This section provides examples of operations that must be performed by various I^2C state service routines. This includes:

- Initialization of the I^2C block after a Reset.
- I^2C Interrupt Service
- The 26 state service routines providing support for all four I^2C operating modes.

19.9.8.1 Initialization

In the initialization example, the I^2C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- The I2ADR registers and I2MASK registers are loaded with values to configure the part's own slave address(es) and the General Call bit (GC)
- The I^2C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the I2EN and AA bits in I2CON and the serial clock frequency (for master modes) is defined by loading the I2SCLH and I2SCLL registers. The master routines must be started in the main program.

The I^2C hardware now begins checking the I^2C-bus for its own slave address and General Call. If the General Call or the own slave address is detected, an interrupt is requested and I2STAT is loaded with the appropriate state information.

19.9.8.2 I^2C interrupt service

When the I^2C interrupt is entered, I2STAT contains a status code which identifies one of the 26 state services to be executed.

19.9.8.3 The state service routines

Each state routine is part of the I^2C interrupt routine and handles one of the 26 states.

19.9.8.4 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I^2C state codes. If one or more of the four I^2C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of time-out during I^2C operations, in order to trap an inoperative bus or a lost service routine.

19.10 Software example

19.10.1 Initialization routine

Example to initialize I^2C Interface as a Slave and/or Master.

1. Load the I2ADR registers and I2MASK registers with values to configure the own Slave Address, enable General Call recognition if needed.
2. Enable I^2C interrupt.
3. Write 0x44 to I2CONSET to set the I2EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to I2CONSET.

19.10.2 Start Master Transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

19.10.3 Start Master Receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to I2CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

19.10.4 I^2C interrupt routine

Determine the I^2C state and which state routine will be used to handle it.

1. Read the I^2C status from I2STA.
2. Use the status value to branch to one of 26 possible state routines.

19.10.5 Non mode specific states

19.10.5.1 State: 0x00

Bus Error. Enter not addressed Slave mode and release bus.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.5.2 Master States

State 0x08 and State 0x10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

19.10.5.3 State: 0x08

A START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

19.10.5.4 State: 0x10

A repeated START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.

1. Write Slave Address with R/W bit to I2DAT.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

19.10.6 Master Transmitter states

19.10.6.1 State: 0x18

Previous state was State 0x08 or State 0x10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted.

1. Load I2DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

19.10.6.2 State: 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.6.3 State: 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a STOP condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit
5. Load I2DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to I2CONSET to set the AA bit.
7. Write 0x08 to I2CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

19.10.6.4 State: 0x30

Data has been transmitted, NOT ACK received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.6.5 State: 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new START condition will be transmitted when the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.7 Master Receive states

19.10.7.1 State: 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.

3. Exit

19.10.7.2 State: 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to I2CONSET to set the STO and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.7.3 State: 0x50

Data has been received, ACK has been returned. Data will be read from I2DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from I2DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

19.10.7.4 State: 0x58

Data has been received, NOT ACK has been returned. Data will be read from I2DAT. A STOP condition will be transmitted.

1. Read data byte from I2DAT into Master Receive buffer.
2. Write 0x14 to I2CONSET to set the STO and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Exit

19.10.8 Slave Receiver states

19.10.8.1 State: 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

19.10.8.2 State: 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

19.10.8.3 State: 0x70

General Call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

19.10.8.4 State: 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General Call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to I2CONSET to set the STA and AA bits.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

19.10.8.5 State: 0x80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to I2CONSET to set the AA bit.
6. Write 0x08 to I2CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

19.10.8.6 State: 0x88

Previously addressed with own Slave Address. Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.8.7 State: 0x90

Previously addressed with General Call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from I2DAT into the Slave Receive buffer.
2. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.
3. Exit

19.10.8.8 State: 0x98

Previously addressed with General Call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.8.9 State: 0xA0

A STOP condition or repeated START has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit

19.10.9 Slave Transmitter states

19.10.9.1 State: 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

19.10.9.2 State: 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load I2DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to I2CONSET to set the STA and AA bits.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

19.10.9.3 State: 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load I2DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to I2CONSET to set the AA bit.
3. Write 0x08 to I2CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

19.10.9.4 State: 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit.

19.10.9.5 State: 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to I2CONSET to set the AA bit.
2. Write 0x08 to I2CONCLR to clear the SI flag.
3. Exit