# 28. I²C

This chapter explains the I²C™ block and its associated registers. The I2C communications block is a serial processor designed to implement a complete I2C slave or master. For a complete table of the I2C registers, refer to the "Summary Table of the System Resource Registers" on page 440. For a quick reference of all PSoC registers in address order, refer to the Register Details chapter on page 139.

## 28.1 Architectural Description

The I2C communications block is a serial to parallel processor, designed to interface the PSoC device to a two-wire I2C serial communications bus. To eliminate the need for excessive M8C microcontroller intervention and overhead, the block provides I2C specific support for status detection and generation of framing bits.

The I2C block directly controls the data (SDA) and clock (SCL) signals to the external I2C interface, through connections to two dedicated GPIO pins. The PSoC device firmware interacts with the block through IO (input/output) register reads and writes, and firmware synchronization will be implemented through polling and/or interrupts.

PSoC I2C features include:

- Master/Slave, Transmitter/Receiver operation
- Byte processing for low CPU overhead
- Interrupt or polling CPU interface
- Master clock rates: 50K, 100K, 400K
- Multi-master clock synchronization
- Multi-master mode arbitration support
- 7- or 10-bit addressing (through firmware support)
- SMBus operation (through firmware support)

Hardware functionality provides basic I2C control, data, and status primitives. A combination of hardware support and firmware command sequencing provides a high degree of flexibility for implementing the required I2C functionality.

Hardware limitations in regards to I2C are as follows:

1. There is no hardware support for automatic address comparison. When Slave mode is enabled, every slave address will cause the block to interrupt the PSoC device and possibly stall the bus.
2. Since receive and transmitted data are not buffered, there is no support for automatic receive acknowledge. The M8C microcontroller must intervene at the boundary of each byte and either send a byte or ACK received bytes.

The I2C block is designed to support a set of primitive operations and detect a set of status conditions specific to the I2C protocol. These primitive operations and conditions are manipulated and combined at the firmware level to support the required data transfer modes. The CPU will set up control options and issue commands to the unit through IO writes and obtain status through IO reads and interrupts.

The block operates as either a slave, a master, or both. When enabled in Slave mode, the unit is always listening for a Start condition, or sending or receiving data. Master mode can work in conjunction with Slave mode. The master supplies the ability to generate the START or STOP condition and determine if other masters are on the bus. For Mult-Master mode, clock synchronization is supported. If Master mode is enabled and Slave mode is not enabled, the block does not generate interrupts on externally generated Start conditions.

### 28.1.1 Basic I²C Data Transfer

Figure 28-1 shows the basic form of data transfers on the I2C bus with a 7-bit address format. (For a more detailed description, see the Phillips Semiconductors' I²C™ Specification, version 2.1.)

A Start condition (generated by the master) is followed by a data byte, consisting of a 7-bit slave address (there is also a 10-bit address mode) and a Read/Write (RW) bit. The RW bit sets the direction of data transfer. The addressed slave is required to acknowledge (ACK) the bus by pulling the data line low during the ninth bit time. If the ACK is received, the transfer may proceed and the master can transmit or receive an indeterminate number of bytes, depending on the RW direction. If the slave does not respond with an ACK for any reason, a Stop condition is generated by the master to terminate the transfer or a Restart condition may be generated for a retry attempt.

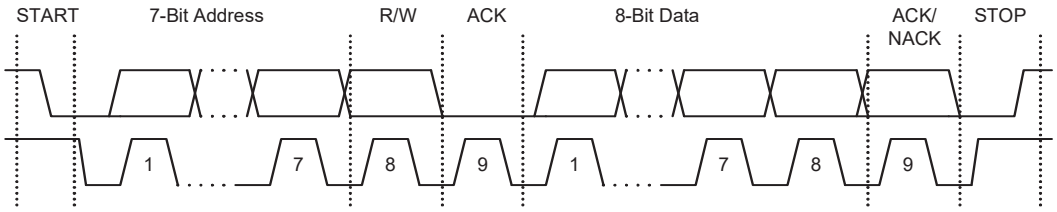


Figure 28-1. Basic I²C Data Transfer with 7-Bit Address Format

## 28.2 Application Description

### 28.2.1 Slave Operation

Assuming Slave mode is enabled, it is continually listening to or on the bus for a Start condition. When detected, the transmitted Address/RW byte is received and read from the I2C block by firmware. At the point where eight bits of the address/RW byte have been received, a byte complete interrupt is generated. On the following low of the clock, the bus is stalled by holding the SCL line low, until the PSoC device has had a chance to read the address byte and compare it to its own address. It will issue an ACK or NACK command based on that comparison.

If there is an address match, the RW bit determines how the PSoC device will sequence the data transfer in Slave mode, as shown in the two branches of Figure 28-2. I2C handshaking methodology (slave holds the SCL line low to "stall" the bus) will be used as necessary, to give the PSoC device time to respond to the events and conditions on the bus. Figure 28-2 is a graphical representation of a typical data transfer from the slave perspective.

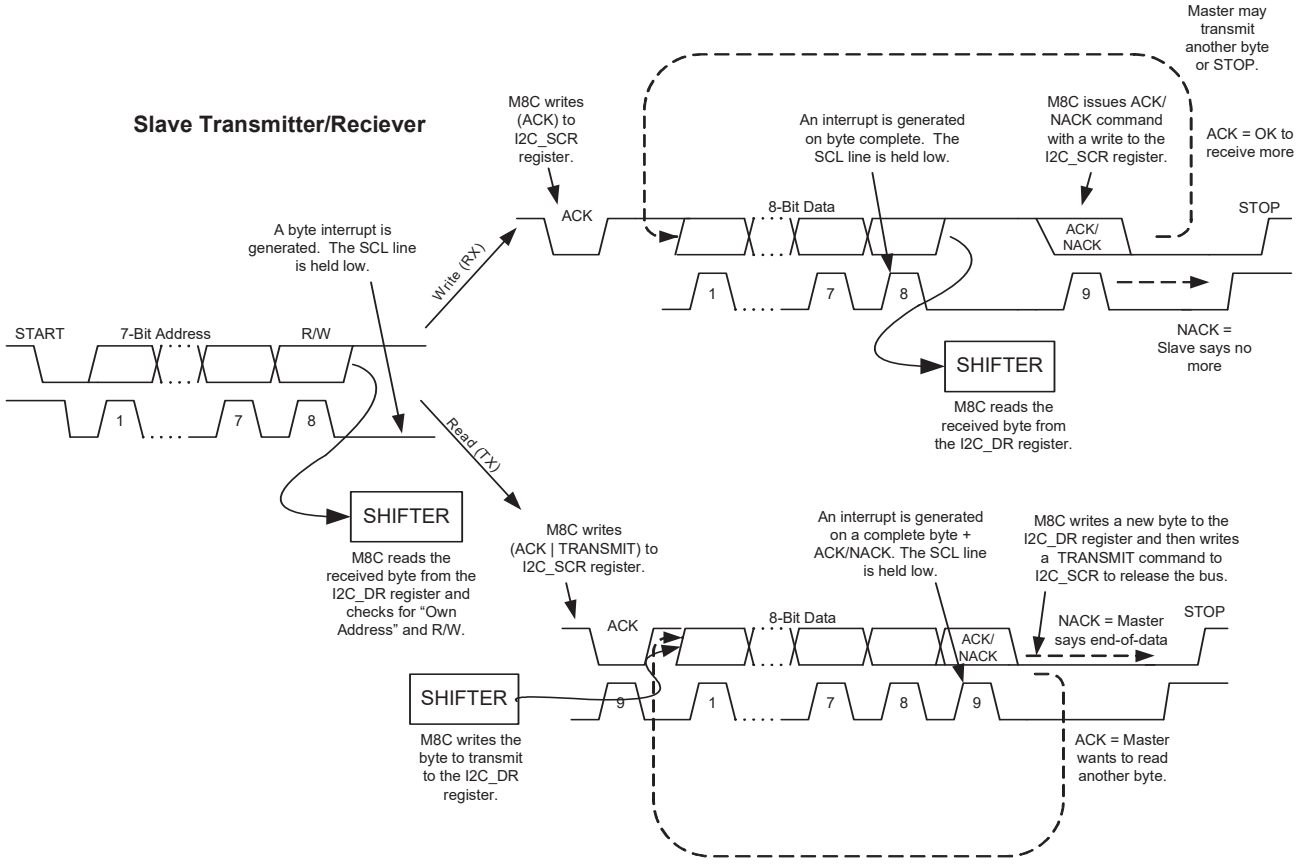


Figure 28-2. Slave Operation

## 28.2.2 Master Operation

To prepare for a Master mode transaction, the PSoC device must determine if the bus is free. This is done by polling the BusBusy status. If busy, interrupts can be enabled to detect a Stop condition. Once it is determined that the bus is available, firmware should write the address byte into the I2C_DR register and set the Start Gen bit in the I2C_MSCR register.

If the slave sub-unit is not enabled, the block is in Master Only mode. In this mode, the unit does not generate interrupts or stall the I2C bus on externally generated Start conditions.

In a multi-master environment there are two additional outcomes possible:

1. The PSoC device was too late to reserve the bus as a master, and another master may have generated a Start and sent an Address/RW byte. In this case, the unit as a master will fail to generate a Start and is forced into Slave mode. The Start will be pending and eventually occur at a later time when the bus becomes free. When the interrupt occurs in Slave mode, the PSoC device can determine that the Start command was unsuccessful by reading the I2C_MSCR register Start bit, which is reset on successful Start from this unit as master. If this bit is still a '1' on the Start/Address interrupt, it means that the unit is operating in Slave mode. In this case, the data register has the master's address data.
2. If another master starts a transmission at the same time as this unit, arbitration occurs. If this unit loses the arbitration, the LostArb status bit is set. In this case, the block releases the bus and switches to Slave operation. When the Start/Address interrupt occurs, the data register has the winning master's address data.

Figure 28-3 is a graphical representation of a typical data transfer from the master perspective.

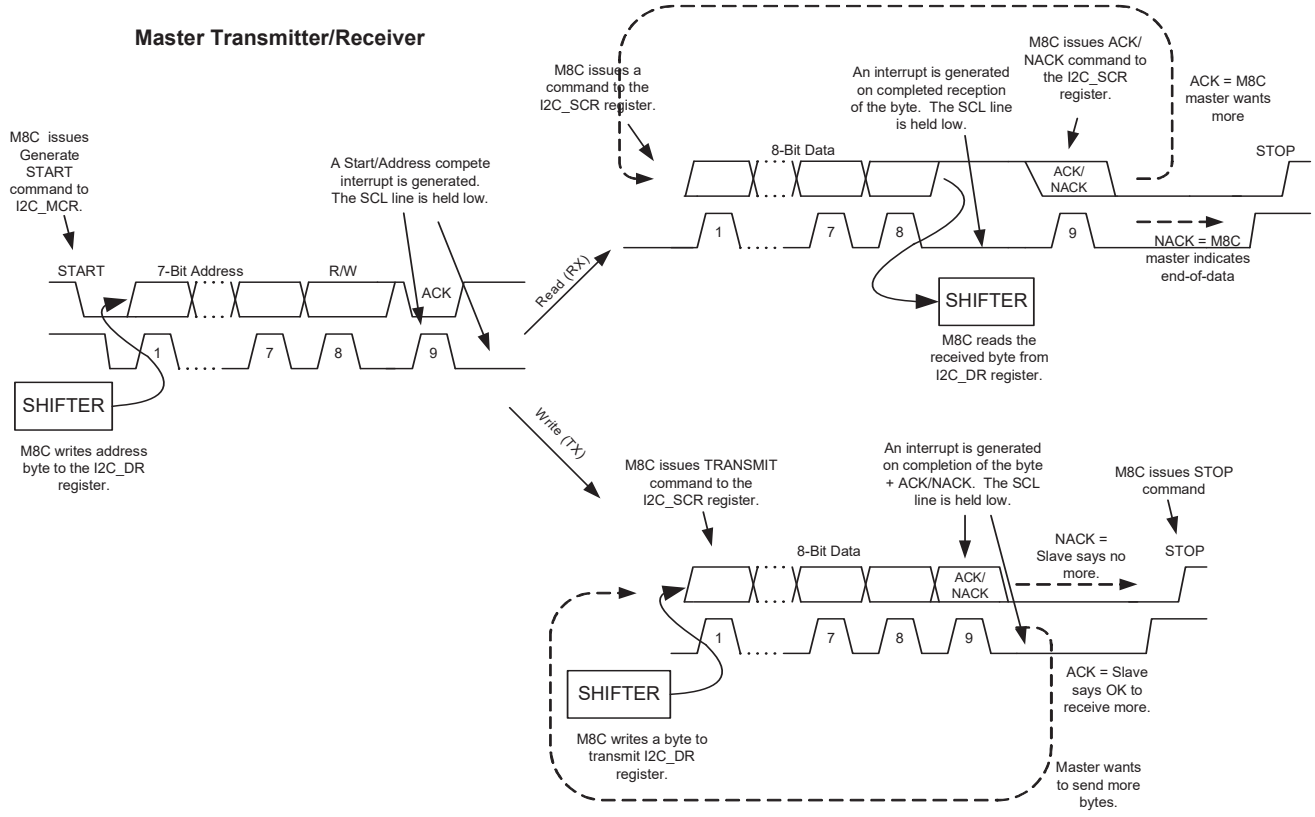


Figure 28-3. Master Operation

## 28.3 Register Definitions

The following registers are associated with I2C and are listed in address order. Each register description has an associated register table showing the bit structure for that register. The bits in the tables that are grayed out are reserved bits and are not detailed in the register descriptions that follow. Reserved bits should always be written with a value of '0'. For a complete table of I2C registers, refer to the "Summary Table of the System Resource Registers" on page 440.

### 28.3.1 I2C_CFG Register

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Access |
|---|---|---|---|---|---|---|---|---|---|---|
| 0,D6h | I2C_CFG | | PSelect | Bus Error IE | Stop IE | Clock Rate[1:0] | | Enable Master | Enable Slave | RW : 00 |

The I2C Configuration Register (I2C_CFG) is used to set the basic operating modes, baud rate, and selection of interrupts.

The bits in this register control baud rate selection and optional interrupts. The values are typically set once for a given configuration. The bits in this register are all RW.

Table 28-1. I2C_CFG Configuration Register

| Bit | Access | Description | Mode |
|---|---|---|---|
| 6 | RW | I2C Pin Select<br>0 = P1[7], P1[5]<br>1 = P1[1], P1[0] | Master/ Slave |
| 5 | RW | Bus Error IE<br>Bus error interrupt enable.<br>0 = Disabled.<br>1 = Enabled. An interrupt is generated on the detection of a Bus Error. | Master Only |
| 4 | RW | Stop IE<br>Stop interrupt enable.<br>0 = Disabled.<br>1 = Enabled. An interrupt is generated on the detection of a Stop Condition. | Master/ Slave |
| 3:2 | RW | Clock Rate<br>00 = 100K Standard Mode<br>01 = 400K Fast Mode<br>10 = 50K Standard Mode<br>11 = Reserved | Master/ Slave |
| 1 | RW | Enable Master<br>0 = Disabled<br>1 = Enabled | Master/ Slave |
| 0 | RW | Enable Slave<br>0 = Disabled<br>1 = Enabled | Master/ Slave |

**Bit 6: PSelect.** With the default value of zero, the I2C pins are P1[7] for clock and P1[5] for data. When this bit is set, the pins for I2C switch to P1[1] for clock and P1[0] for data. This bit may not be changed while either the Enable Master or Enable Slave bits are set. However, the PSelect bit may be set at the same time as the enable bits. The two sets of pins that may be used on I2C are not equivalent. The default set, P1[7] and P1[5], are the preferred set. The alternate set, P1[1] and P1[0], are provided so that I2C may be used with 8-pin PSoC parts.

If In-circuit System Serial Programming (ISSP™) is to be used and the alternate I2C pin set is also used, it is necessary to take into account the interaction between the PSoC Test Controller and the I2C bus. The interface requirements for ISSP should be reviewed to ensure that they are not violated.

Even if ISSP is not used, pins P1[1] and P1[0] will respond differently to a POR or XRES event than other IO pins. After an XRES event, both pins are pulled down to ground by going into the resistive zero Drive mode, before reaching the High-Z Drive mode. After a POR event, P1[0] will drive out a one, then go to the resistive zero state for some time, and finally reach the High-Z drive mode state. After POR, P1[1] will go into a resistive zero state for a while, before going to the High-Z Drive mode.

Another issue with selecting the alternate I2C pins set is that these pins are also the crystal pins. Therefore, a crystal may not be used when the alternate I2C pin set is selected.

**Bit 5: Bus Error IE (Interrupt Enable).** This bit controls whether the detection of a bus error will generate an interrupt. A bus error is typically a misplaced Start or Stop.

This is an important interrupt with regards to Master operation. When there is a misplaced Start or Stop on the I2C bus, all slave devices (including this device, if Slave mode is enabled) will reset the bus interface and synchronize to this signal. However, when the hardware detects a bus error in Master Mode operation, the device will release the bus and transition to an idle state. In this case, a Master operation in progress will never have any further status or interrupts associated with it. Therefore, the master may not be able to determine the status of that transaction. An immediate bus error interrupt will inform the master that this transfer did not succeed.

**Bit 4: Stop IE (Interrupt Enable).** When this bit is set, a master or slave can interrupt on Stop detection. The status bit associated with this interrupt is the Stop Status bit in the Slave Status and Control register. When the Stop Status bit transitions from '0' to '1', the interrupt is generated. It is important to note that the Stop Status bit is not automatically cleared. Therefore, if it is already set, no new interrupts are generated until it is cleared by firmware.

**Bits 3 and 2: Clock Rate[1:0].** These bits offer a selection of three sampling and bit rates. All block clocking is based on the SYSCLK input, which is nominally 24 MHz (unless the PSoC device is in external clocking mode). The sampling rate and the baud rate are determined as follows:

- Sample Rate = SYSCLK/Pre-scale Factor
- Baud Rate = 1/(Sample Rate x Samples per Bit)

The nominal values, when using the internal 24 MHz oscillator, are shown in Table 28-2.

Table 28-2. I²C Clock Rates

| Clock Rate [1:0] | I2C Mode | SYSCLK Pre-scale Factor | Samples per Bit | Internal Sampling Freq./Period (24 MHz) | Master Baud Rate (nominal) | Start/Stop Hold Time (8 clocks) |
|---|---|---|---|---|---|---|
| 00b | Standard | /16 | 16 | 1.5 MHz/667 ns | 93.75 kHz | 5.3 µs |
| 01b | Fast | /4 | 16 | 6 MHz/167 ns | 375 kHz | 1.33 µs |
| 10b | Standard | /16 | 32 | 1.5 MHz/667 ns | 46.8 kHz | 10.7 µs |
| 11b | Reserved | | | | | |

When clocking the input with a frequency other than 24 MHz (for example, clocking the PSOC device with an external clock), the baud rates and sampling rates will scale accordingly. Whether the block will work in a Standard Mode or Fast Mode system depends on the sample rate. The sample rate must be sufficient to resolve bus events, such as Start and Stop conditions. (See the Phillips Semiconductors' I²C™ Specification, version 2.1, for minimum Start and Stop hold times.)

**Bit 1: Enable Master.** When this bit is set, the Master Status and Control register is enabled (otherwise it is held in reset) and I2C transfers can be initiated in Master mode. When the master is enabled and operating, the block will clock the I2C bus at one of three baud rates, defined in the Clock Rate register. When operating in Master mode, the hardware is multi-master capable, implementing both clock synchronization and arbitration. If the Slave Enable bit is not set, the block will operate in Master Only mode. All external Start conditions are ignored (although the Bus Busy status bit will still keep track of bus activity). Block enable will be synchronized to the SYSCLK clock input (see "Timing Diagrams" on page 474).

**Bit 0: Enable Slave.** When the slave is enabled, the block generates an interrupt on any Start condition and an address byte that it receives, indicating the beginning of an I2C transfer. When operating as a slave, the block is clocked from an external master. Therefore, the block will work at any frequency up to the maximum defined by the currently selected clock rate. The internal clock is only used in Slave mode, to ensure that there is adequate setup time from data output to the next clock on the release of a slave stall. When the Enable Slave and Enable Master bits are both '0', the block is held in reset and all status is cleared. See Figure 28-3 for a description of the interaction between the Master/Slave Enable bits. Block enable will be synchronized to the SYSCLK clock input (see "Timing Diagrams" on page 474).

Table 28-3. Enable Master/Slave Block Operation

| Enable Master | Enable Slave | Block Operation |
|---|---|---|
| No | No | Disabled:<br>The block is disconnected from the GPIO pins, P1[5] and P1[7]. (The pins may be used as general purpose IO.) When either the master or slave is enabled, the GPIO pins are under control of the I2C hardware and are unavailable.<br>All internal registers (except I2C_CFG) are held in reset. |
| No | Yes | Slave Only Mode:<br>Any external Start condition will cause the block to start receiving an address byte. Regardless of the current state, any Start resets the interface and initiates a Receive operation. Any Stop will cause the block to revert to an idle state<br>The I2C_MSCR register is held in reset. |
| Yes | No | Master Only Mode:<br>External Start conditions are ignored in this mode. No Byte Complete interrupts on external traffic are generated, but the Bus Busy status bit continues to capture Start and Stop status, and thus may be polled by the master to determine if the bus is available.<br>Full multi-master capability is enabled, including clock synchronization and arbitration.<br>The block will generate a clock based on the setting in the Clock Rate register |
| Yes | Yes | Master/Slave Mode:<br>Both master and slave may be operational in this mode. The block may be addressed as a slave, but firmware may also initiate Master mode transfers.<br>In this configuration, when a master loses arbitration during an address byte, the hardware will revert to Slave mode and the received byte will generate a slave address interrupt. |

For additional information, refer to the I2C_CFG register on page 216.

## 28.3.2 I2C_SCR Register

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Access |
|---|---|---|---|---|---|---|---|---|---|---|
| 0,D7h | I2C_SCR | Bus Error | Lost Arb | Stop Status | ACK | Address | Transmit | LRB | Byte Complete | # : 00 |

**LEGEND**

# Access is bit specific. Refer to Table 28-4 for detailed bit descriptions.

The I2C Status and Control Register (I2C_SCR) is used by both master and slave to control the flow of data bytes and to keep track of the bus state during a transfer.

This register contains status bits, for determining the state of the current I2C transfer, and control bits, for determining the actions for the next byte transfer. At the end of each byte transfer, the I2C hardware interrupts the M8C microcontroller and stalls the I2C bus on the subsequent low of the clock, until the PSoC device intervenes with the next command. This register may be read as many times as necessary; but on a subsequent write to this register, the bus stall is released and the current transfer will continue.

There are six status bits: Byte Complete, LRB, Address, Stop Status, Lost Arb, and Bus Error. These bits have Read/Clear (R/C) access, which means that they are set by hardware but may be cleared by a write of '0' to the bit position. Under certain conditions, status is cleared automatically by the hardware. These cases are noted in Table 28-4.

There are two control bits: Transmit and ACK. These bits have RW access and may be cleared by hardware.

**Bit 7: Bus Error.** The Bus Error status detects misplaced Start or Stop conditions on the bus. These may be due to noise, rogue devices, or other devices that are not yet synchronized with the I2C bus traffic. According to the I2C specification, all compatible devices must reset their interface on a received Start or Stop. This is a natural thing to do in Slave mode, because a Start will initiate an address reception and a Stop will idle the slave. In the case of a master, this event will force the master to release the bus and idle. However, since a master does not respond to external Start or Stop conditions, an immediate interrupt on this event allows the master to continue to keep track of the bus state.

A bus error is defined as follows. A Start is only valid if the block is idle (master or slave) or a Slave receiver is ready to receive the first bit of a new byte after an ACK. Any other timing for a Start condition causes the Bus Error bit to be set. A Stop is only valid if the block is idle or a Slave receiver is ready to receive the first bit of a new byte after an ACK. Any other timing for a Stop condition causes the Bus Error bit to be set.

Table 28-4. I2C_SCR Status and Control Register

| Bit | Access | Description | Mode |
|---|---|---|---|
| 7 | RC | Bus Error<br>1 = A misplaced Start or Stop condition was detected.<br>This status bit must be cleared by firmware with a write of '0' to the bit position. It is never cleared by the hardware. | Master Only |
| 6 | RC | Lost Arb<br>1 = Lost Arbitration.<br>This bit is set immediately on lost arbitration; however, it does not cause an interrupt. This status may be checked after the following Byte Complete interrupt.<br>Any Start detect will automatically clear this bit. | Master Only |
| 5 | RC | Stop Status<br>1 = A Stop condition was detected.<br>This status bit must be cleared by firmware with a write of '0' to the bit position. It is never cleared by the hardware. | Master/ Slave |
| 4 | RW | ACK: Acknowledge Out<br>0 = NACK the last received byte.<br>1 = ACK the last received byte.<br>This bit is automatically cleared by hardware on the following Byte Complete event. | Master/ Slave |
| 3 | RC | Address<br>1 = The transmitted or received byte is an address.<br>This status bit must be cleared by firmware with a write of '0' to the bit position. | Master/ Slave |
| 2 | RW | Transmit<br>0 = Receive Mode.<br>1 = Transmit Mode.<br>This bit is set by firmware to define the direction of the byte transfer.<br>Any Start detect will automatically clear this bit. | Master/ Slave |
| 1 | RC | LRB: Last Received Bit<br>The value of the ninth bit in a Transmit sequence, which is the acknowledge bit from the receiver.<br>0 = Last transmitted byte was ACK'ed by the receiver.<br>1 = Last transmitted byte was NACK'ed by the receiver.<br>Any Start detect will automatically clear this bit. | Master/ Slave |
| 0 | RC | Byte Complete<br>Transmit Mode:<br>1 = 8 bits of data have been transmitted and an ACK or NACK has been received.<br>Receive Mode:<br>1 = 8 bits of data have been received.<br>Any Start detect will automatically clear this bit. | Master/ Slave |

**Bit 6: Lost Arb.** This bit is set when I2C bus contention is detected, during a Master mode transfer. Contention will occur when a master is writing a '1' to the SDA output line and reading back a '0' on the SDA input line at the given sampling point. When this occurs, the block immediately releases the SDA, but continues clocking to the end of the current byte. On the resulting byte interrupt, firmware can determine that arbitration was lost to another master by reading this bit.

The sequence occurs differently between Master transmitter and Master receiver. As a transmitter, the contention will occur on a data bit. On the subsequent Byte Complete interrupt, the Lost Arbitration status is set. In Receiver mode, the contention will occur on the ACK bit. The master that NACK'ed the last reception will lose the arbitration. However, the hardware will shift in the next byte in response to the winning master's ACK, so that a subsequent Byte Complete interrupt occurs. At this point, the losing master can read the Lost Arbitration status. Contention is checked only at the eight data bit sampling points and one ACK bit sampling point.

**Bit 5: Stop Status.** Stop status is set on detection of an I2C Stop condition. This bit is sticky, which means that it will remain set until a '0' is written back to it by the firmware. This bit may only be cleared if the Byte Complete status is set. If the Stop Interrupt Enable bit is set, an interrupt is also generated on Stop detection. It is never automatically cleared.

Using this bit, a slave can distinguish between a previous Stop or Restart on a given address byte interrupt. In Master mode, this bit may be used in conjunction with the Stop IE bit, to generate an interrupt when the bus is free. However, in this case, the bit must have previously been cleared prior to the reception of the Stop in order to cause an interrupt.

**Bit 4: ACK.** This control bit defines the acknowledge data bit that is transmitted out in response to a received byte. When receiving, a Byte Complete interrupt is generated after the eighth data bit is received. On the subsequent write to this register to continue (or terminate) the transfer, the state of this bit will determine the next bit of data that is transmitted. It is ***active high***. A '1' will send an ACK and a '0' will send a NACK.

A Master receiver normally terminates a transfer, by writing a '0' (NACK) to this bit. This releases the bus and automatically generates a Stop condition. A Slave receiver may also send a NACK, to inform the master that it cannot receive any more bytes.

**Bit 3: Address.** This bit is set when an address has been received. This consists of a Start or Restart, and an address byte. This bit applies to both master and slave.

In Slave mode, when this status is set, firmware will read the received address from the data register and compare it with its own address. If the address does not match, the firmware will write a NACK indication to this register. No further interrupts will occur, until the next address is received. If the address does match, firmware must ACK the received byte, then Byte Complete interrupts are generated on subsequent bytes of the transfer.

This bit will also be set when address transmission is complete in Master mode. If a lost arbitration occurs during the transmission of a master address (indicated by the Lost Arb bit), the block will revert to Slave mode if enabled. This bit then signifies that the block is being addressed as a slave.

If Slave mode is not enabled, the Byte Complete interrupt will still occur to inform the master of lost arbitration.

**Bit 2: Transmit.** This bit sets the direction of the shifter for a subsequent byte transfer. The shifter is always shifting in data from the I2C bus, but a write of '1' enables the output of the shifter to drive the SDA output line. Since a write to this register initiates the next transfer, data must be written to the data register prior to writing this bit. In Receive mode, the previously received data must have been read from the data register before this write. In Slave mode, firmware derives this direction from the RW bit in the received slave address. In Master mode, the firmware decides on the direction and sets it accordingly.

This direction control is only valid for data transfers. The direction of address bytes is determined by the hardware, depending on the Master or Slave mode.

The Master transmitter terminates a transfer by writing a zero to the transmit bit. This releases the bus and automatically sends a Stop condition, or a Stop/Start or Restart, depending on the I2C_MSCR control bits.

**Bit 1: LRB (Last Received Bit).** This is the last received bit in response to a previously transmitted byte. In Transmit mode, the hardware will send a byte from the data register and clock in an acknowledge bit from the receiver. On the subsequent byte complete interrupt, firmware will check the value of this bit. A '0' is the ACK value and a '1' is a NACK value. The meaning of the LRB depends on the current operating mode.

***Master Transmitter:***

**'0': ACK.** The slave has accepted the previous byte. The master may send another byte by first writing the byte to the I2C_DR register and then setting the Transmit bit in the I2C_SCR register. Optionally, the master may clear the Transmit bit in the I2C_SCR register. This will automatically send a Stop. If the Start or Restart bits are set in the I2C_MSCR register, the Stop may be followed by a Start or Restart.

**'1': NACK.** The slave cannot accept any more bytes. A Stop is automatically generated by the hardware on the subsequent write to the I2C_SCR register (regardless of the value written). However, a Stop/Start or Restart condition may also be generated, depending on whether

firmware has set the Start or Restart bits in the I2C_MSCR register.

***Slave Transmitter:***

**'0': ACK.** The master wants to read another byte. The slave should load the next byte into the I2C_DR register and set the transmit bit in the I2C_SCR register to continue the transfer.

**'1': NACK.** The master is done reading bytes. The slave will revert to IDLE state on the subsequent I2C_SCR write (regardless of the value written).

**Bit 0: Byte Complete.** The I2C hardware operates on a byte basis. In Transmit mode, this bit is set and an interrupt is generated at the end of nine bits (the transmitted byte + the received ACK). In Receive mode, the bit is set after the eight bits of data are received. When this bit is set, an interrupt is generated at these data sampling points, which are associated with the SCL input clock rising (see details in the Timing section). If the PSoC device responds with a write back to this register before the subsequent falling edge of SCL (which is approximately one-half bit time), the transfer will continue without interruption. However, if the PSoC device is unable to respond within that time, the hardware will hold the SCL line low, stalling the I2C bus. In both Master and Slave mode, a subsequent write to the I2C_SCR register will release the stall.

For additional information, refer to the I2C_SCR register on page 217.

## 28.3.3 I2C_DR Register

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Access |
|---|---|---|---|---|---|---|---|---|---|---|
| 0,D8h | I2C_DR | Data[7:0] | | | | | | | | RW : 00 |

The I2C Data Register (I2C_DR) provides read/write access to the Shift register.

**Bits 7 to 0: Data[7:0].** This register is not buffered; and therefore, writes and valid data reads may only occur at specific points in the transfer. These cases are outlined as follows.

- **Master or Slave Receiver** – Data in the I2C_DR register is only valid for reading, when the Byte Complete status bit is set. Data bytes must be read from the register before writing to the I2C_SCR register, which continues the transfer.
- **Master Start or Restart** – Address bytes must be written in I2C_DR before the Start or Restart bit is set in the I2C_MSCR register, which causes the Start or Restart to generate and the address to shift out.
- **Master or Slave Transmitter** – Data bytes must be written to the I2C_DR register before the transmit bit is set in the I2C_SCR register, which causes the transfer to continue.

For additional information, refer to the I2C_DR register on page 219.

## 28.3.4 I2C_MSCR Register

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Access |
|---|---|---|---|---|---|---|---|---|---|---|
| 0,D9h | I2C_MSCR | | | | | Bus Busy | Master Mode | Restart Gen | Start Gen | R : 00 |

The I2C Master Status and Control Register (I2C_MSCR) implements I2C framing controls and provides Bus Busy status.

**Bit 3: Bus Busy.** This read only bit is set to '1' by any Start condition and reset to '0' by a Stop condition. It may be polled by firmware to determine when a bus transfer may be initiated.

**Bit 2: Master Mode.** This bit indicates that the device is operating as a master. It is set in the detection of this block's Start condition and reset in the detection of the subsequent Stop condition.

**Bit 1: Restart Gen.** This bit is only used at the end of a master transfer (as noted in Other Cases 1 and 2 of the Start Gen bit). If an address is loaded into the data register and this bit is set prior to NACKing (Master receiver) or resetting the transmit bit (Master transmitter), or after a Master transmitter is NACK'ed by the slave, a Restart condition is generated followed by the transmission of the address byte.

**Bit 0: Start Gen.** Before setting this bit, firmware must write the address byte to send into the I2C_DR register. When this bit is set, the Start condition is generated followed immediately by the transmission of the address byte. (No control in the I2C_SCR register is needed for the master to initiate a transmission; the direction is inherently "transmit.")

The bit is automatically reset to ‘0’ after the Start has been generated.

There are three possible outcomes as a result of setting the Start Gen bit.

1. The bus is free and the Start condition is generated successfully. A Byte Complete interrupt is generated after the Start and the address byte are transmitted. If the address was ACK’ed by the receiver, the firmware may then proceed to send data bytes.
2. The Start command is too late. Another master in a multi-master environment has generated a valid Start and the bus is busy. The resulting behavior depends upon whether Slave mode is enabled.

   Slave mode is enabled: A Start and address byte interrupt is generated. When reading the I2C_MSCR register, the master will see that the Start Gen bit is still set and that the I2C_SCR register has the Address bit set, indicating that the block is addressed as a slave.

   Slave mode is not enabled: The Start Gen bit will remain set and the Start is queued, until the bus becomes free and the Start condition is subsequently generated. An interrupt is generated at a later time, when the Start and address byte has been transmitted.
3. The Start is generated, but the master looses arbitration to another master in a multi-master environment. The resulting behavior depends upon whether Slave mode is enabled.

   Slave mode is enabled: A Start and address byte interrupt is generated. When reading the I2C_MSCR, the master will see that the Start Gen bit cleared, indicating that the Start was generated. However, the Lost Arb bit is set in the I2C_SCR register. The Address status is also set, indicating that the block has been addressed as a slave. The firmware may then ACK or NACK the address to continue the transfer.

   Slave mode is not enabled: A Start and address byte interrupt is generated. The Start Gen bit is cleared and the Lost Arb bit is set. The hardware will wait for command input, stalling the bus if necessary. In this case, the master will clear the I2C_SCR register, to release the bus and allow the transfer to continue, and the block will idle.

Other cases where the Start bit may be used to generate a Start condition are as follows.

1. When a master is finished with a transfer, a NACK is written to the I2C_SCR register (in the case of the Master receiver) or the transmit bit is cleared (in case of a Master transmitter). Normally, the action will free the stall and generate a Stop condition. However, if the Start bit is set and an address is written into the data register prior to the I2C_SCR write, a Stop, followed immediately by a Start (minimum bus free time), is generated. In this way, messages may be chained.
2. When a Master transmitter is NACK’ed, an automatic Stop condition is generated on the subsequent I2C_SCR write. However, if the Start Gen bit has previously been set, the Stop is immediately followed by a Start condition.

Table 28-5. I2C_MSCR Master Status/Control Register

| Bit | Access | Description | Mode |
|---|---|---|---|
| 3 | R | Bus Busy<br>This bit is set to ‘1’ when any Start condition is detected and reset to ‘0’ when a Stop condition is detected. | Master Only |
| 2 | R | Master Mode<br>This bit is set to ‘1’ when a start condition, generated by this block, is detected and reset to ‘0’ when a stop condition is detected. | Master Only |
| 1 | RW | Restart Gen<br>1 = Generate a Restart condition.<br>This bit is cleared by hardware when the Start generation is complete. | Master Only |
| 0 | RW | Start Gen<br>1 = Generate a Start condition and send a byte (address) to the I2C bus.<br>This bit is cleared by hardware when the Start generation is complete. | Master Only |

For additional information, refer to the I2C_MSCR register on page 220.

## 28.4 Timing Diagrams

### 28.4.1 Clock Generation

Figure 28-4 illustrates the I2C input clocking scheme. The SYSCLK pin is an input into a four-stage ripple divider that provides the baud rate selections. When the block is disabled, all internal state is held in a reset state. When either the Master or Slave Enable bits in the I2C_CFG register are set, the reset is synchronously released and the clock generation is enabled. Two taps from the ***ripple divider*** are selectable (/4, /16) from the clock rate bits in the I2C_CFG register. If any of the two divider taps is selected, that clock is resynchronized to SYSCLK. The resulting clock is routed to all of the synchronous elements in the design.

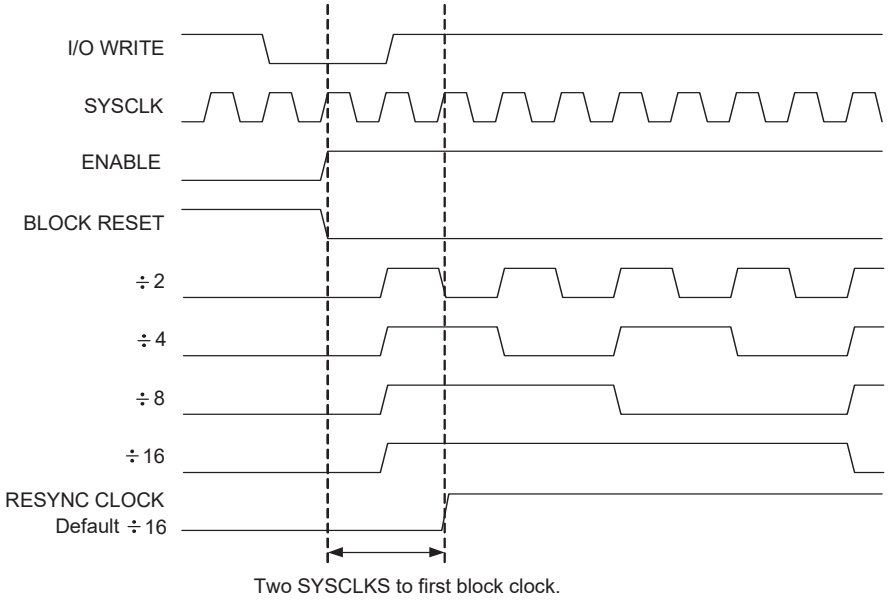


Figure 28-4. I²C Input Clocking

### 28.4.2 Basic Input/Output Timing

Figure 28-5 illustrates basic input output timing that is valid for both 16 times sampling and 32 times sampling. For 16 times sampling, N=4; and for 32 times sampling, N=12. N is derived from the half-bit rate sampling of eight and 16 clocks, respectively, minus the input latency of three (count of 4 and 12 correspond to 5 and 13 clocks).

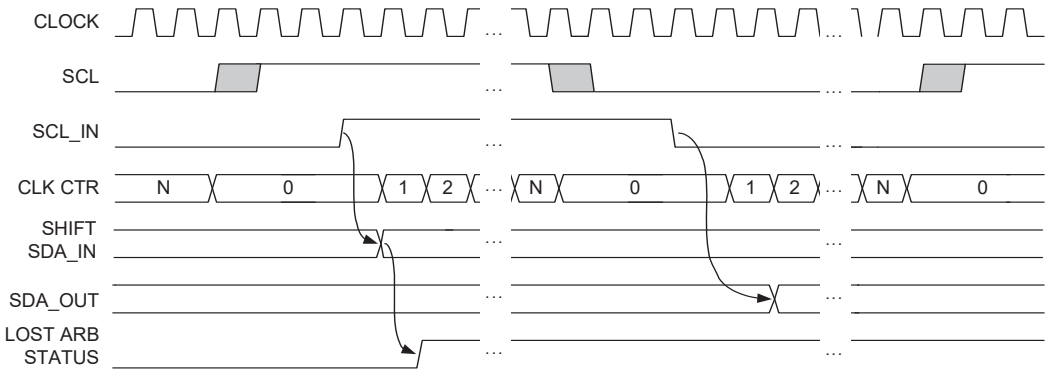


Figure 28-5. Basic Input/Output Timing

### 28.4.3 Status Timing

Figure 28-6 illustrates the interrupt timing for Byte Complete, which occurs on the positive edge of the ninth clock (byte + ACK/NACK) in Transmit mode and on the positive edge of the eighth clock in Receive mode. There is a maximum of three cycles of latency, due to the input synchronizer/filter circuit. As shown, the interrupt occurs on the clock following a valid SCL positive edge input transition (after the synchronizers). The Address bit is set with the same timing, but only after a slave address has been received. The LRB (Last Received Bit) status is also set with the same timing, but only on the ninth bit after a transmitted byte.

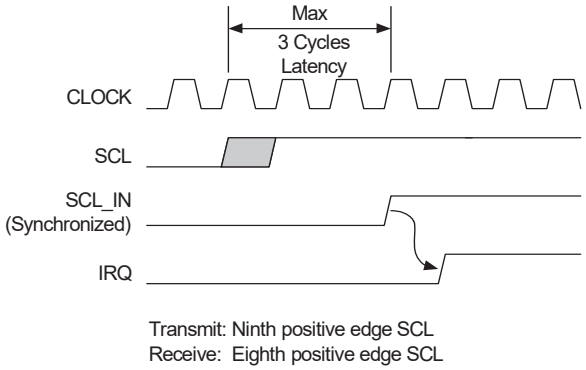


Figure 28-6. Byte Complete, Address, LRB Timing

Figure 28-7 shows the timing for Stop Status. This bit is set (and the interrupt occurs) two clocks after the synchronized and filtered SDA line transitions to a '1', when the SCL line is high.

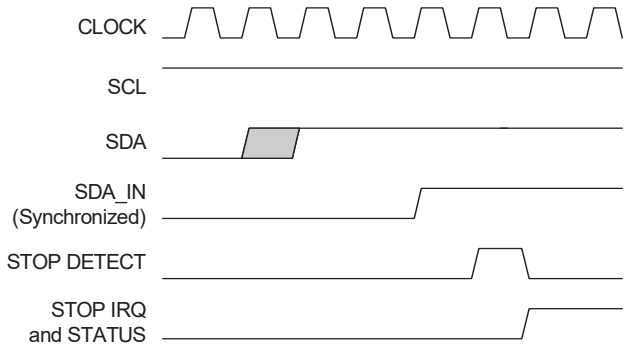


Figure 28-7. Stop Status and Interrupt Timing

Figure 28-8 illustrates the timing for bus error interrupts. Bus Error status (and Interrupt) occurs one cycle after the internal Start or Stop Detect (two cycles after the filtered and synced SDA input transition).

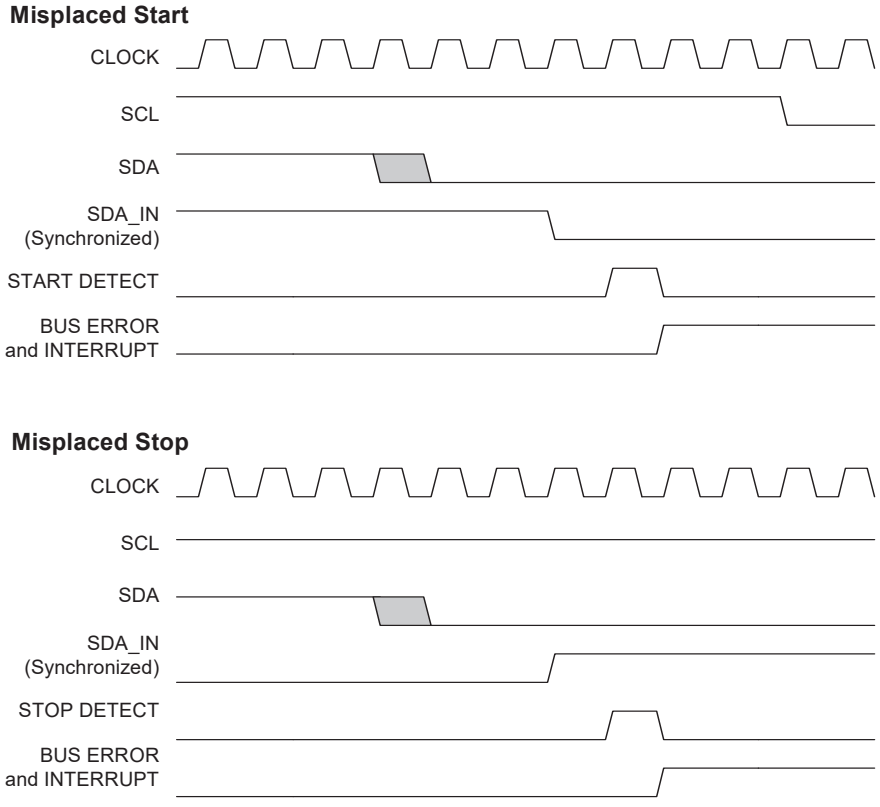


Figure 28-8. Bus Error Interrupt Timing

### 28.4.4 Master Start Timing

When firmware writes the Start Gen command, hardware resynchronizes this bit to SYSCLK, to ensure a minimum of a full SYSCLK of setup time to the next clock edge. When the Start is initiated, the SCL line is left high for 6/14 clocks (corresponding to 16/32 times sampling rates). During this initial SCL high period, if an external Start is detected, the Start sequence is aborted and the block returns to an IDLE state. However, on the next Stop detection, the block will automatically initiate a new Start sequence.

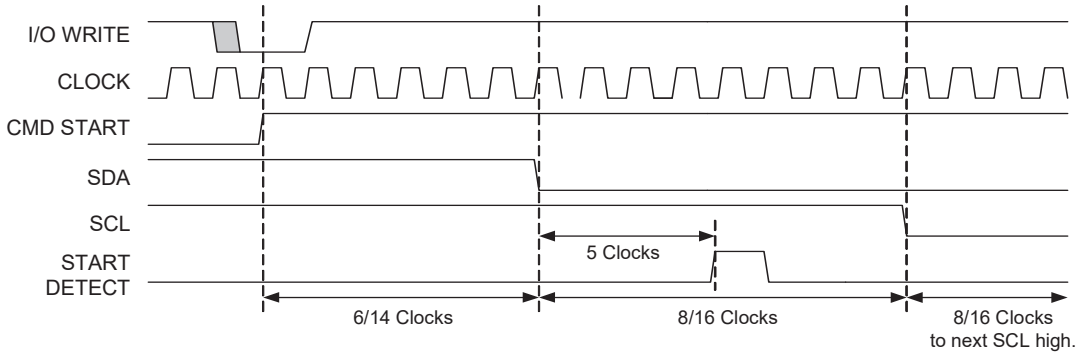


Figure 28-9. Basic Master Start Timing

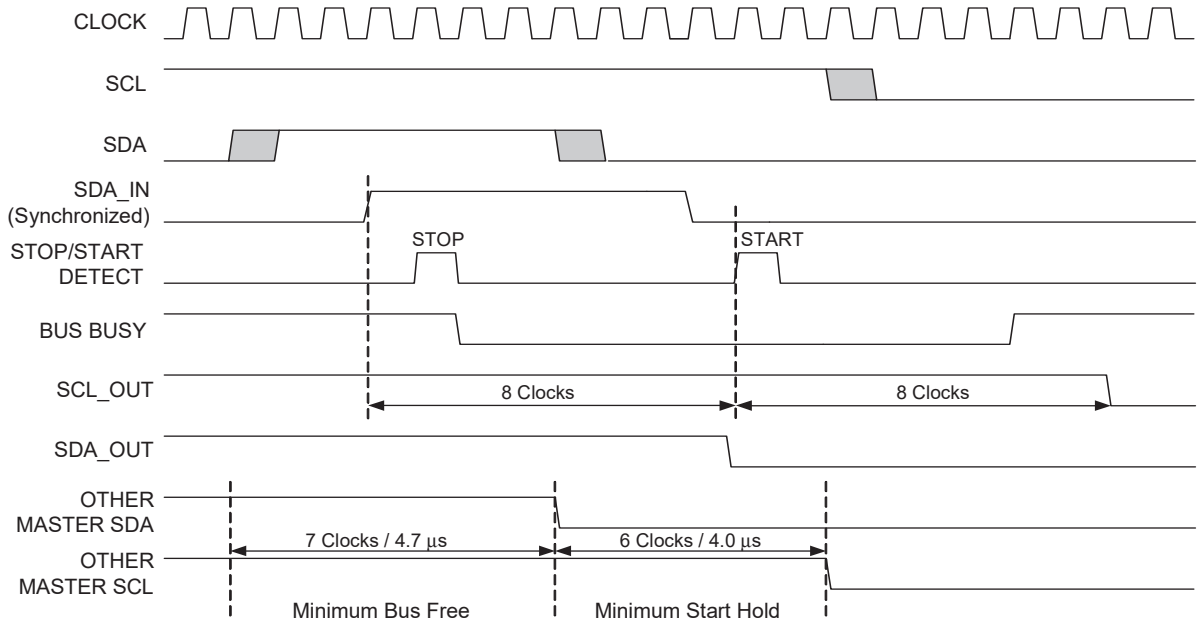


Figure 28-10. Start Timing with a Pending Start

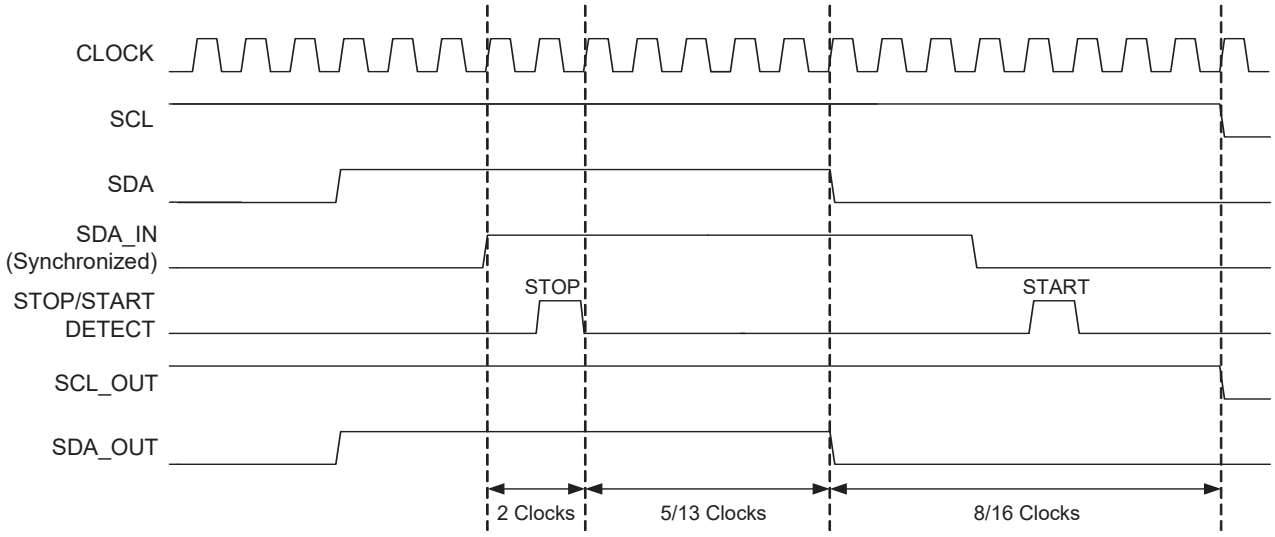


Figure 28-11. Master Stop/Start Chaining

## 28.4.5 Master Restart Timing

Figure 28-12 shows the Master Restart timing. After the ACK/NACK bit, the clock is held low for a half bit time (8/16 clocks corresponding to the 16 or 32 times sampling rates), during which time the data is allowed to go high, then a valid start is generated in the following 3 half bit times as shown.

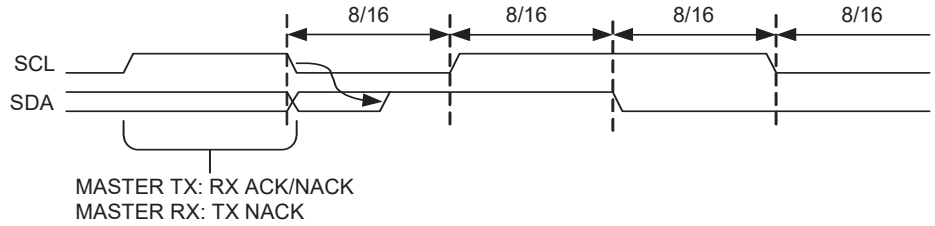


Figure 28-12. Master Restart Timing

## 28.4.6 Master Stop Timing

Figure 28-13 shows basic Master Stop timing. In order to generate a Stop, the SDA line is first pulled low, in accordance with the basic SDA output timing. Then, after the full low of SCL is completed and the SCL line is pulled high, the SDA line remains low for a full one-half bit time before it is pulled high to signal the Stop.

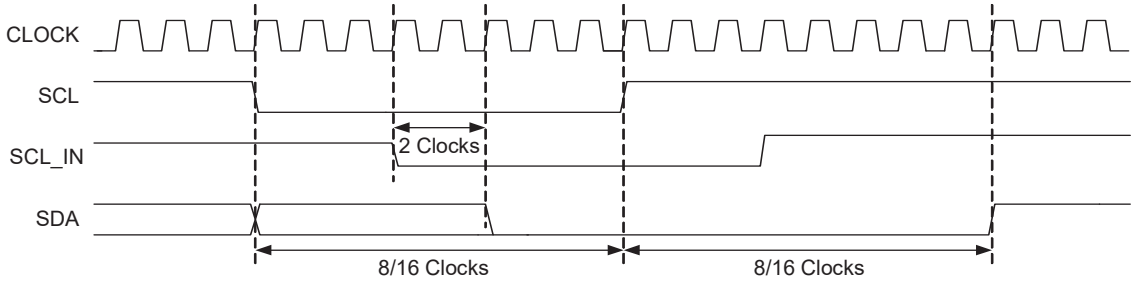


Figure 28-13. Master Stop Timing

## 28.4.7 Master/Slave Stall Timing

When a Byte Complete interrupt occurs, the PSoC device firmware must respond with a write to the I2C_SCR register to continue the transfer (or terminate the transfer). The interrupt occurs two clocks after the rising edge of SCL_IN (see "Status Timing" on page 475). As illustrated in Figure 28-14, firmware has until one clock after the falling edge of SCL_IN to write to the I2C_SCR register; otherwise, a stall occurs. Once stalled, the IO write releases the stall. The setup time between data output and the next rising edge of SCL will always be N-1 clocks.

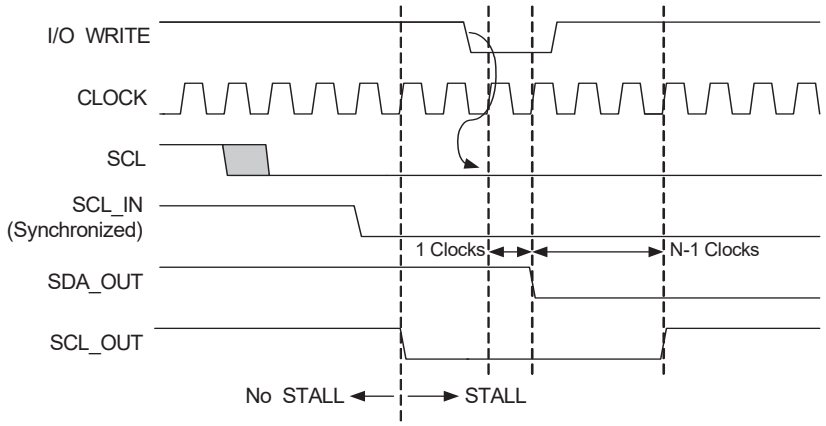


Figure 28-14. Master/Slave Stall Timing

## 28.4.8 Master Lost Arbitration Timing

Figure 28-15 shows a Lost Arbitration sequence. When contention is detected at the input (SDA_IN) sampling point, the SDA output is immediately released to an IDLE state. However, the master continues clocking until the Byte Complete interrupt, which is processed in the usual way. Any write to the I2C_SCR register results in the master reverting to an IDLE state, one clock after the next positive edge of the SCL_IN clock.

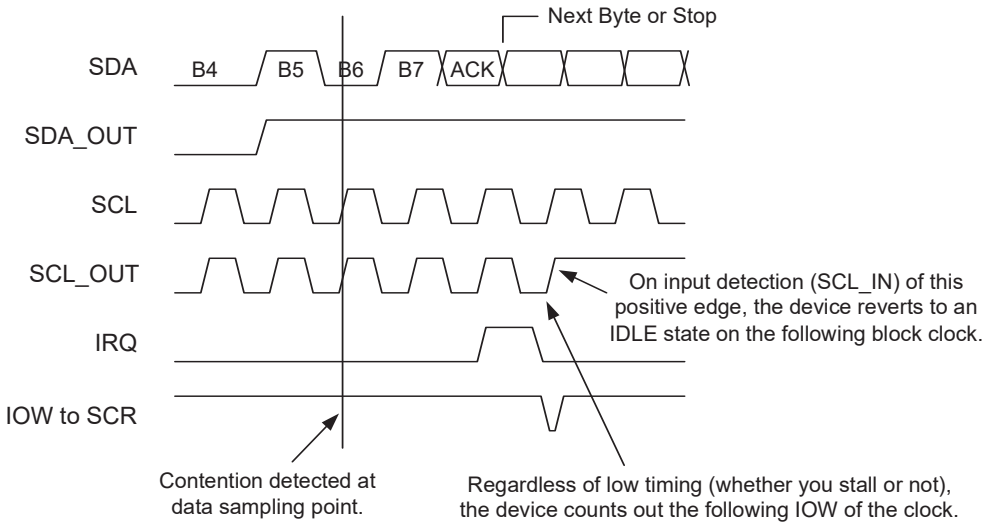


Figure 28-15. Lost Arbitration Timing (Transmitting Address or Data)

## 28.4.9 Master Clock Synchronization

Figure 28-16 shows the timing associated with Master Clock Synchronization. Clock synchronization is always operational, even if it is the only master on the bus. In which case, it is synchronizing to its own clock. In the wired AND bus, an SCL output of ‘0’ is seen by all masters. When the hardware asserts a ‘0’ to the output, it is immediately fed back from the PSoC device pin to the input synchronizer for the SCL input. The counter value (depending on the sampling rate) takes into account the worst case latency for input synchronization of three clocks, giving a net period of 8/16 clocks for both high and low time. This results in an overall clocking rate of 16/32 clocks per bit.

In multi-master environments when the hardware outputs a ‘1’ on the SCL output, if any other master is still asserting a ‘0’, the clock counter will hold until the SCL input line matches the ‘1’ on the SCL output line. When matched, the remainder of the high time is counted down. In this way, the master with the fastest frequency determines the high time of the clock and the master with the lowest frequency determines the low time of the clock.

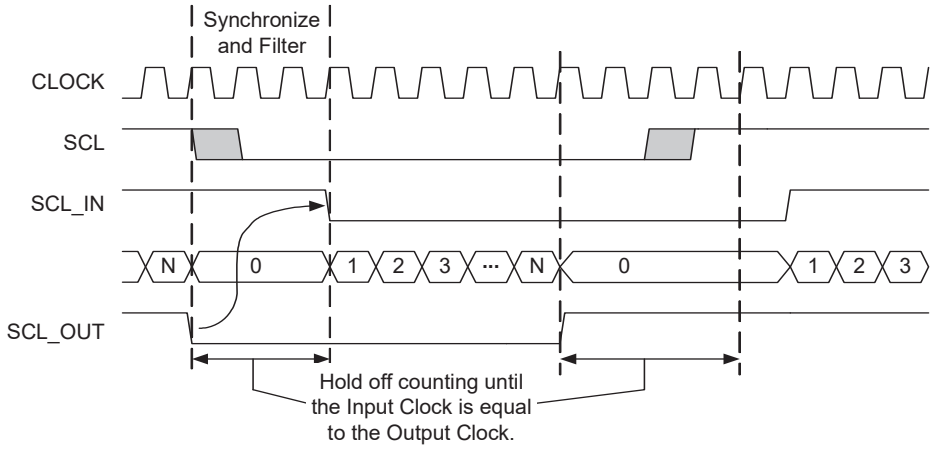


Figure 28-16. Master Clock Synchronization