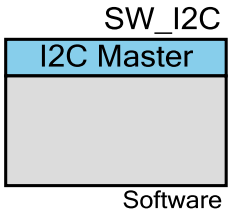
Software I2C Master Slave

0.1

Features



- Industry-standard NXP® I²C bus interface
- Requires only two pins (SDA and SCL) to interface to I²C bus
- Supports standard data rates of 50/100 kbps
- High-level APIs require minimal user programming

General Description

The Software I²C Master component supports I²C master designed without any hardware block. The I²C bus is an industry-standard, two-wire hardware interface developed by Philips. The master initiates all communication on the I²C bus and supplies the clock for all slave devices.

The Software I²C component supports two clock speeds – 50 KHz and 100 KHz. The component is compatible with other third-party slave devices.

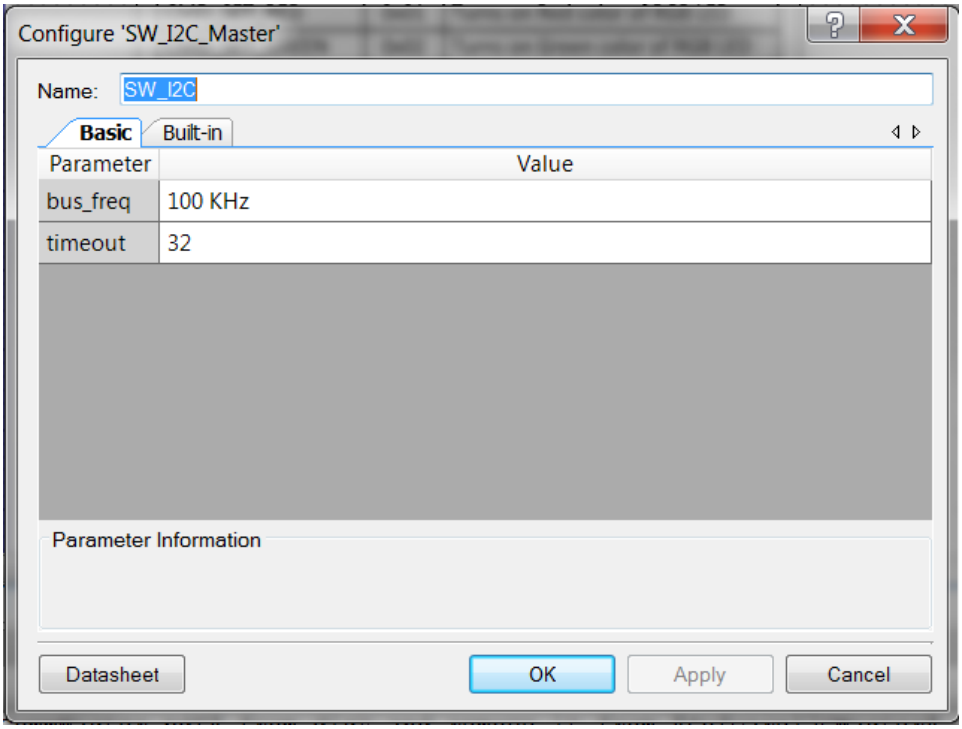
When to Use an Software I²C Master Component

The Software I²C Master component is used to send I2C frames as a master device. The component consists solely of firmware and a two pins, so it is useful on devices without digital resources, or in projects where all digital resources are consumed.

Component Parameters

Drag a Software I²C Master component onto your design and double-click it to open the **Configure** dialog.

Basic Tab



Bus Frequency

This parameter is used to set the I²C clock frequency value. It supports 50 KHz or 100 KHz. The actual frequency may differ based on the PSoC clock frequency and how long interrupt service routines are executed during an I2C transfer.

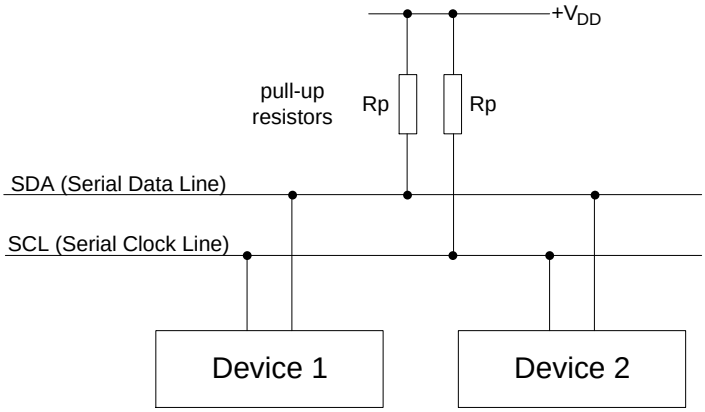
Timeout

This parameter sets a timeout in milliseconds when sending an I2C frame. Since the APIs are blocking and the slave might stretch the clock during the process, this timeout is mechanism to avoid a forever loop scenario.

External Electrical Connections

As shown in the following figure, the I^2C bus requires external pull-up resistors. The pull-up resistors (R_P) are primarily determined by the supply voltage, bus speed, and bus capacitance. For detailed information on how to calculate the optimum pull-up resistor value for your design we recommend using the UM10204 I^2C-bus specification and user manual Rev. 6, available from the NXP website at www.nxp.com.

Figure 1. Connection of Devices to the I^2C Bus



For most designs, the default values shown in the following table provide excellent performance without any calculations. The default values were chosen to use standard resistor values between the minimum and maximum limits.

Table 1. Recommended Default Pull-up Resistor Values

Standard Mode (0 – 100 kbps)	Fast Mode (0 – 400 kbps)	Fast Mode Plus (0 – 1000 kbps)	Units
4.7 k, 5%	1.74 k, 1%	620, 5%	Ω

These values work for designs with 1.8 V to 5.0V V_{DD}, less than 200 pF bus capacitance (C_B), up to 25 µA of total input leakage (I_{IL}), up to 0.4 V output voltage level (V_{OL}), and a max V_{IH} of 0.7 * V_{DD}.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component during run time. The following table lists and describes the interface to each function. The subsequent sections discuss each function in more detail.

By default, PSoC Creator assigns the instance name "SW_I2C_Master_1" to the first instance of a component in a given design. You can rename the instance to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "I2C."

All API functions assume that data direction is from the perspective of the I^2C master. A write event occurs when data is written from the master to the slave. A read event occurs when the master reads data from the slave.

Function	Description
I2C_WriteBuf()	Writes the referenced data buffer to a specified slave address.
I2C_ReadBuf()	Reads data from the specified slave address and places the data in the referenced buffer.
I2C_SendStart()	Sends only a Start to the specific address.
I2C_SendStop()	Generates a Stop condition.
I2C_WriteByte()	Writes a single byte. This is a manual command that should only be used with the I2C_MasterSendStart() or I2C_MasterSendRestart() functions.
I2C_ReadByte()	Reads a single byte. This is a manual command that should only be used with the I2C_MasterSendStart() or I2C_MasterSendRestart() functions.

uint32 I2C_WriteBuf(uint32 slaveAddress, uint8 * wrData, uint832 cnt, uint32 mode)

Description: This function automatically writes an entire buffer of data to a slave device. This function is blocking and does not return until the entire frame is completed or an error occurred.

Parameters: uint32 slaveAddress: Right-justified 7-bit slave address (valid range 0 to 127).

uint8 wrData: Pointer to the buffer of the data to be sent.

uint32 cnt: Number of bytes of the buffer to send.

uint32 mode: Transfer mode defines: (1) Whether a Start or Restart condition is generated at the beginning of the transfer, and (2) Whether the transfer is completed or halted before the Stop condition is generated on the bus.

Mode Constants	Description
I2C_MODE_COMPLETE_XFER	Perform complete transfer from Start to Stop.
I2C_MODE_REPEAT_START	Send Repeat Start instead of Start.
I2C_MODE_NO_STOP	Execute transfer without a Stop

Return Value: uint32: Error Status. See the I2C_MasterSendStart() function for constants.

Side Effects: None

uint32 I2C_ReadBuf(uint32 slaveAddress, uint8 * rdData, uint32 cnt, uint32 mode)

Description: This function automatically reads an entire buffer of data from a slave device. This function is blocking and does not return until the entire frame is completed or an error occurred.

Parameters: uint32 slaveAddress: Right-justified 7-bit slave address (valid range 0 to 127).

uint8 rdData: Pointer to the buffer in which to put the data from the slave.

uint32 cnt: Number of bytes of the buffer to read.

uint32 mode: Transfer mode defines: (1) Whether a Start or Restart condition is generated at the beginning of the transfer and (2) Whether the transfer is completed or halted before the Stop condition is generated on the bus.

Mode Constants	Description
I2C_MODE_COMPLETE_XFER	Perform complete transfer for Start to Stop.
I2C_MODE_REPEAT_START	Send Repeat Start instead of Start.
I2C_MODE_NO_STOP	Execute transfer without a Stop

Return Value: uint32: Error Status. See the I2C_MasterSendStart() function for constants.

Side Effects: None

uint32 I2C_SendStart(uint32 slaveAddress)

Description: This function generates a Start condition and sends the slave address with the read/write bit. Disables the I²C interrupt.

Parameters: uint32 slaveAddress: Right-justified 8-bit slave address with RW LSB bit

Return Value: uint32: Error Status.

Mode Constants	Description
I2C_ NO_ERROR	Function completed without error.
I2C_ BUS_BUSY	Bus is busy, Start condition was not generated.
I2C_ NOT_READY	The master is not a valid master on the bus, or a slave operation is in progress.
I2C_ERR_LB_NAK	The last byte was NAKed.
I2C_ ERR_ARB_LOST	The master lost arbitration while the Start was generated. (This status is only valid if multi-master is enabled.)
I2C_ERR_ABORT_START_GEN	Start condition generation was aborted because of the start of slave operation. (This status is only valid in multi-master-slave mode.)

Side Effects: This function is blocking and does not return until the address byte is transferred.

uint32 I2C_SendStop(void)

Description: Generates Stop condition on the bus. The NAK is generated before Stop in case of a read transaction. At least one byte has to be read if a Start or ReStart condition with read direction was generated before.

Parameters: None

Return Value: uint32: Error Status. See the I2C_MasterSendStart() command for constants.

Side Effects: This function is blocking and does not return until a stop condition is generated or error occurred.

uint32 I2C_WriteByte(uint32 theByte)

Description: This function sends one byte to a slave.
This function is blocking and does not return until byte is transmitted or error occurred.

Parameters: uint32 theByte: Data byte to send to the slave.

Return Value: uint32: Error Status.

Mode Constants	Description
I2C_MSTR_NO_ERROR	Function complete without error.
I2C_MSTR_NOT_READY	The master is not a valid master on the bus or slave operation is in progress.
I2C_MSTR_ERR_LB_NAK	The last byte was NAKed.
I2C_MSTR_ERR_ARB_LOST	The master lost arbitration. (This status is valid only if multi-master is enabled.)

Side Effects: None

uint32 I2C_ReadByte(uint32 acknNak)

Description: Reads one byte from a slave and generates ACK or prepares to generate NAK.
This function is blocking. It does not return until a byte is received or an error occurs.

Parameters: uint32 acknNack: Response to received byte.

Response constants	Description
I2C_ACK_DATA	Generates ACK. The master notifies slave that transfer continues.
I2C_NAK_DATA	Prepares to generate NAK. The master will notify slave that transfer is completed.

Return Value: uint32: Byte read from the slave.

Side Effects: None

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
0.10	Initial version	