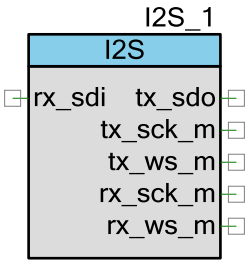
Inter-IC Sound Bus (I2S_PDL)

2.0

Features



- I2S, Left-Justified, TDM mode A, and TDM mode B Data Formats
- Master/Slave TX/RX Operation
- Programmable Channel/Word Lengths
- Internal/external Clock Operation
- Programmable Channel Number in TDM Data Formats
- Configurable Bit Clock Inversion

General Description

The I2S_PDL Component is used to send digital audio streaming data to external I2S devices, such as audio codecs or simple DACs. Furthermore, it is able to receive digital audio streaming data.

This Component is a hybrid graphical configuration entity with a set of Component-specific API built on top of the I2S driver available in the Peripheral Driver Library (PDL). It allows schematic-based connections and hardware configuration as defined by the Component Configure dialog.

When to Use an I2S_PDL Component

This Component can be used for voice/audio communication, automobile audio, portable audio, IoT systems, and home entertainment. Examples of I2S_PDL Component usage in an audio system include:

- USB audio OUT to Analog audio output
- Analog microphone to USB audio IN
- I2S digital microphone: CPU stores audio samples in PSoC SRAM location
- I2S digital microphone to Wireless Transmitter (A2DP over Bluetooth for example)
- Flash ROM audio data to play sounds or music

The I2S_PDL Component is typically part of a system to implement digital voice/audio recording/playing and processing such as:

- Wearable Device
- Headphone/Earphone
- IoT/IoE System
- Virtual Reality Gaming System
- Tablet
- Smart Automobile
- Smart Home System

Definitions

- I2S – Inter-IC Sound. See Philips I2S Bus Specification
- PDL - Peripheral Driver Library

Quick Start

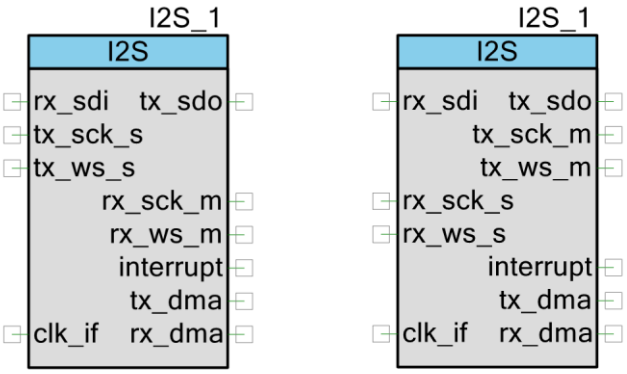
1. Drag an “I2S (fixed function) [v2.0]” Component or any of the provided I2S macros from the Component Catalog Cypress/Communications/I2S folder onto your schematic (the placed instance takes the name I2S_1).
2. Double-click to open the Configure dialog.
3. Set up the desired I2S settings (clock divider, channel/word lengths, interrupts, etc.).
4. Connect all the input/output digital pins (if using the base Component and not a macro).
5. Build the project in order to verify the correctness of your design. This will add the required PDL modules to the Workspace Explorer and generate the configuration data for the I2S_1 instance.
6. In *main.c*, initialize the peripheral and start the application:

```
(void)Cy_I2S_Init(I2S_1_HW, &I2S_1_config);
Cy_I2S_ClearRxFifo(I2S_1_HW);
Cy_I2S_ClearTxFifo(I2S_1_HW);
Cy_I2S_WriteTxData(I2S_1_HW, 0UL);
Cy_I2S_WriteTxData(I2S_1_HW, 0UL);
Cy_I2S_EnableRx(I2S_1_HW);
Cy_I2S_EnableTx(I2S_1_HW);
Cy_I2S_SetInterruptMask(I2S_1_HW, I2S_1_INT_MASK);
```

7. Build the project and program the device.

Input/Output Connections

This section describes the various input and output connections for the I2S_PDL Component. Each terminal may be shown on the Component symbol for the conditions listed in the description of that I/O.

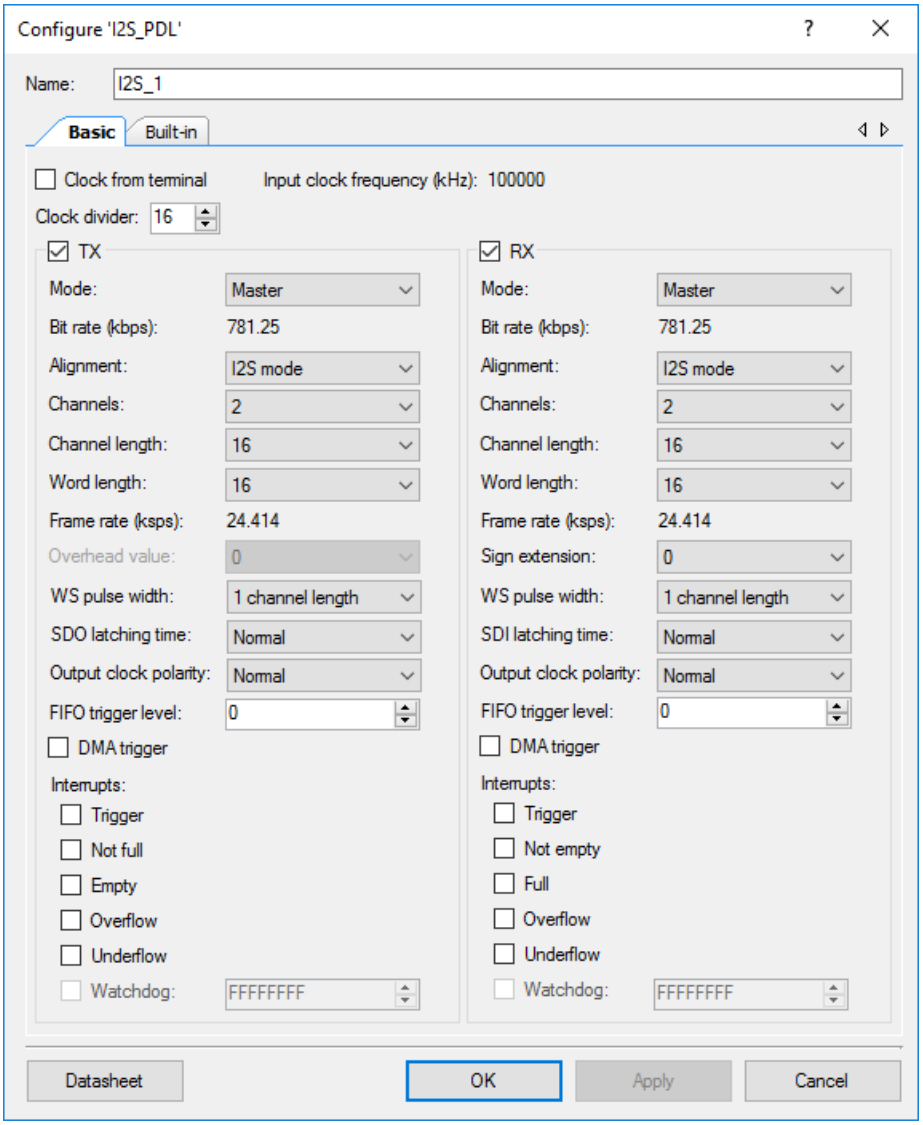


Terminal Name	I/O Type	Description
rx_sdi	Digital Input	Rx serial data input (visible when Rx is enabled)
clk_if	Digital Input	External interface clock (visible when “Clock from terminal” is enabled)
tx_sdo	Digital Output	Tx serial data output (visible when Tx is enabled)
tx_sck_m	Digital Output	Tx serial clock output (visible when Tx master is enabled)
tx_ws_m	Digital Output	Tx word select output (visible when Tx master is enabled)
rx_sck_m	Digital Output	Rx serial clock output (visible when Rx master is enabled)
rx_ws_m	Digital Output	Rx word select output (visible when Rx master is enabled)
tx_sck_s	Digital Input	Tx serial clock input (visible when Tx slave is enabled)
tx_ws_s	Digital Input	Tx word select input (visible when Tx slave is enabled)
rx_sck_s	Digital Input	Rx serial clock input (visible when Rx slave is enabled)
rx_ws_s	Digital Input	Rx word select input (visible when Rx slave is enabled)
interrupt	Digital Output	Interrupt signal output (visible when at least one interrupt is enabled)
tx_dma	Digital Output	Tx DMA transfer request signal (visible when Tx DMA trigger is enabled)
rx_dma	Digital Output	Rx DMA transfer request signal (visible when Rx DMA trigger is enabled)

Component Parameters

Drag an I2S_PDL Component onto your design and double click it to open the Configure dialog. This dialog has the following tabs with different parameters.

Basic Tab



Parameter Name	Description
Clock from terminal	Selects input clock source: external interface clock (from the clk_if terminal) or internal clock (HFClk1).
Clock divider	Sets the input Clock Divider.

Parameter Name	Description
TX and RX (unless specified)	
Mode	Sets mode to master or slave.
Bit rate (kbps)	Read-only field that displays the bite rate. See also Clock Selection.
Alignment	Set interface transfer mode: I2S, Left Justified, TDM mode A, or TDM mode B.
Channels	Number of channels per frame (2 is the only valid value for Left Justified and I2S modes).
Channel length	Set channel length in bits (32 bit is the only valid value for TDM modes).
Word length	Set word length (in bits).
Frame rate (ksps)	Read-only field that displays the frame rate. See also Clock Selection.
TX Overhead value	Set the overhead bits level (available only when word length is less than channel length). It fills all the channel's LSBs (after the data LSB) on SDO wire by 1 or 0.
RX Sign extension	Set the sign extension bits level (available only when word length is less than 32 bit). It fills all the RX FIFO word MSBs by the data word's sign bit (MSB) or by 0.
WS pulse width	Pulse width of the WS signal: one SCK period or one channel length. The "one channel length" is the only valid value for I2S and Left Justified modes.
TX SDO latching time	Available only in Slave mode; should be "Normal" in master mode. • If "Normal" – the TX output data is set/changed on the SDO wire at the falling SCK edge (accordingly to the I2S Standard, if TX Output clock polarity is "Normal"). • If "Half-clock advanced" – the TX output data is set/changed on the SDO wire at the rising SCK edge (if TX Output clock polarity is "Normal") which goes before that "Normal" falling SCK edge, i.e. the SDO waveform is advanced by 0.5 SCK period. If TX Output clock polarity is "Inverted" – the rising/falling edges just swaps in above explanations. The "latching" in terms of TX means the real time when the output TX trigger sets the data on SDO wire; that is, the data changing time.
RX SDI latching time	Available only in Master mode; should be Normal in slave mode. • If "Normal" – the RX input data is captured on the SDI wire at the rising SCK edge (accordingly to the I2S Standard, if RX Output clock polarity is "Normal"). • If "Half-clock delayed" – the RX input data is captured on the SDI wire at the falling SCK edge (if RX Output clock polarity is "Normal") which goes after that "Normal" rising SCK edge, i.e. the SDI waveform is delayed by 0.5 SCK period. If RX Output clock polarity is "Inverted" – the rising/falling edges just swaps in above explanations. The "latching" in terms of RX means the real time when the input RX trigger captures the data on SDI wire, i.e. about at a middle between the data changing edges.
Input clock polarity	Polarity of the input SCK signal (available only in slave mode).

Parameter Name	Description
Output clock polarity	Polarity of the output SCK signal (available only in master mode).
FIFO trigger level	Set FIFO level to trigger an event (interrupt or DMA request). Should not be greater than [255 - (number of channels)].
DMA trigger	Enables DMA trigger.
Interrupts:	
Trigger	Trigger interrupt if FIFO entries number is less than the FIFO trigger level. **Note** This is a primary interrupt for usual data flow process. All other interrupts are used for corner cases/exception conditions handling.
TX Not full	Trigger interrupt if TX FIFO is not full.
RX Not empty	Trigger interrupt if RX FIFO is not empty.
TX Empty	Trigger interrupt if TX FIFO is empty.
RX Full	Trigger interrupt if RX FIFO is full (contains exactly 256 data words). **Note** For 3, 5, 6, and 7 channels in TDM modes, the RX_FULL interrupt won't occur because the data flow on the HW side goes frame-by-frame, and 256 is not a multiple of these frame sizes.
Overflow	Trigger interrupt if FIFO is overflowed.
Underflow	Trigger interrupt if FIFO is underflowed.
Watchdog	Trigger interrupt if Watchdog timer value reaches zero (available only in slave mode).
Watchdog Value	The reset value of the Watchdog timer (available only in slave mode).

Application Programming Interface

The Application Programming Interface (API) is provided by the **I2S** driver module from the PDL. The driver is copied into the “pdl\drivers\peripheral\i2s\” directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the “**Open PDL Documentation…**” option in the drop-down menu.

The Component generates the configuration structures and base address described in the Global Variables and Preprocessor Macros sections. Pass the generated data structure and the base address to the associated I2S driver function in the application initialization code to configure the peripheral. Once the peripheral is initialized, the application code can perform run-time changes by referencing the provided base address in the driver API functions.

By default, PSoC Creator assigns the instance name **I2S_1** to the first instance of a Component in a given design. You can rename it 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 section is **I2S_1**.

Global Variables

The I2S_PDL Component populates the following peripheral initialization data structure. The generated code is placed in C source and header files that are named after the instance of the Component (e.g. I2S_1.c). Each variable is also prefixed with the instance name of the Component.

cy_stc_i2s_config_t I2S_1_config

The instance-specific configuration structure. This should be used in the associated Cy_I2S_Init() function.

Preprocessor Macros

The I2S_PDL Component generates the following preprocessor macros. Note that each macro is prefixed with the instance name of the Component (e.g. “I2S_1”).

#define I2S_1_HW (I2S_1_cy_mxs40_i2s__HW)

The pointer to the base address of the HW I2S instance.

#define I2S_1_TX_FIFO_WR_PTR ((uint32_t *) &I2S_1_HW->TX_FIFO_WR)

Tx FIFO Write register pointer for DMA initialization.

#define I2S_1_RX_FIFO_RD_PTR ((uint32_t *) &I2S_1_HW->RX_FIFO_RD)

Rx FIFO Read register pointer for DMA initialization.

#define I2S_1_INT_MASK (I2S_1_INT_MASK_TX | I2S_1_INT_MASK_RX)

The default (configured in GUI) interrupt mask.

Component Functions

This Component also includes a set of Component-specific wrapper functions that provide simplified access to the basic I2S operation. These functions are generated during the build process and are all prefixed with the name of the Component instance.

Data in RAM

The generated data may be placed in flash memory (const) or RAM. The former is the more memory-efficient choice if you do not wish to modify the configuration data at run-time. Under the

Built-In tab of the Configure dialog set the parameter CONST_CONFIG to make your selection. The default option is to place the data in flash.

Code Examples and Application Notes

Code Examples

PSoC Creator provides access to code examples in the Find Code Example dialog. For Component-specific examples, open the dialog from the Component Catalog or an instance of the Component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page.

Application Notes

Cypress provides a number of application notes describing how PSoC can be integrated into your design. You can access the Cypress Application Notes search web page at www.cypress.com/appnotes.

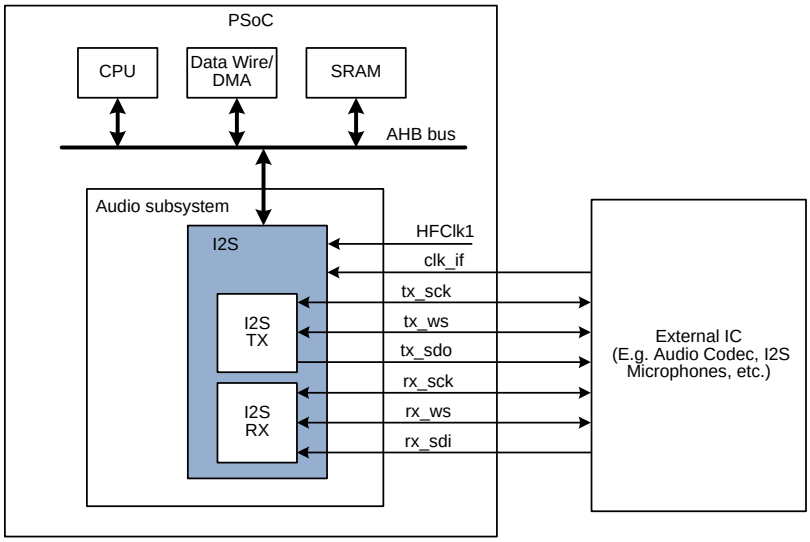
Functional Description

The fixed-function I2S block incorporates two I2S serial interface converters, a transmitter and a receiver. The I2S is based on a de-facto industry standard from Philips.

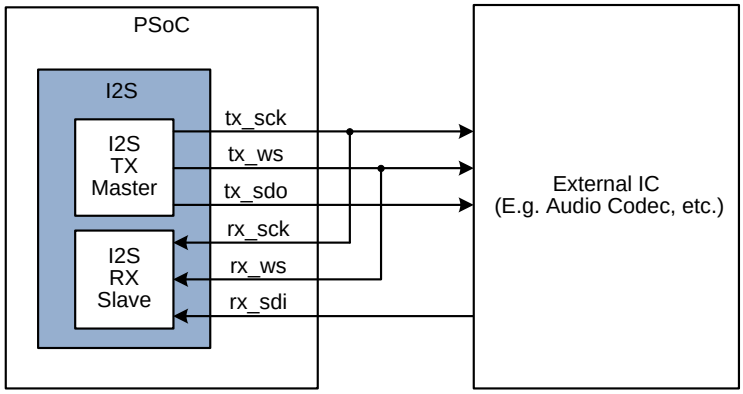
The transmitter and receiver each have their own I2S interface, and each operates as either an I2S master or an I2S slave.

Block Diagram and Configuration

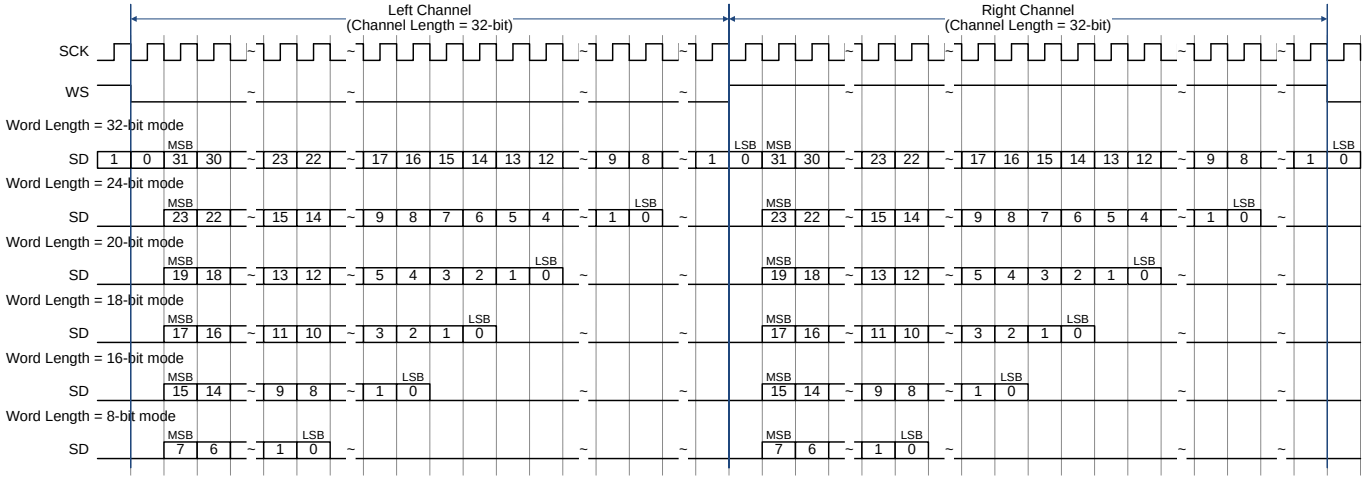
The following is a simplified diagram of the I2S hardware:



The following is a connection example for Full Duplex operation:



The I2S frame looks like the following:



This is an example for channel length = 32. It is similar for all other channel lengths, with only one limitation: the word length could be less or equal to the channel length.

The differences between I2S and Left Justified modes are as follows:

- The MSB of data word is available on the first rising edge of SCK right after frame sync (WS) transition (not the second, as is in the I2S format), i.e. the data is one SCK advanced (shifted left relative to WS) in comparison to the I2S format.
- The WS = 1 for left channel and WS = 0 for right channel, i.e. in opposite order to the I2S format.

See the “Inter-IC Sound Bus (I2S)” section in the device *Technical Reference Manual (TRM)* for more details.

Clock Selection

The internal clock source for I2S is the HFClk1 (which is configurable in the PSoC Design Wide Resources Clock Editor). The Component can accept an external interface clock through the clk_if terminal, which can be connected to the dedicated GPIO pin only.

In slave mode, the I2S SCK signal must be synchronous to either HFClk1 or external interface clock. So, either the HFClk1 should be routed out as a master clock for I2S master device, or external interface clock (generated either by the I2S master device or independent master clock generator for I2S bus) should be routed into the PSoC I2S block (via the clk_if terminal).

For master mode, either an external or internal clock is divided by a common configurable “Clock divider” and then by a constant 8x divider. The resulting clock at the output of these dividers is the actual bit clock of the I2S bus stream. This value is indicated by the **Bit rate (kbps)** field on the Configure dialog.

Bit rate (kbps) = HFClk1 (kHz) / Clock divider / 8

The actual frame rate can be calculated from the bit rate divided by the frame length (channel length x number of channels). For I2S/Left Justified modes the number of channels is always 2:

Frame rate (ksps) = Bit rate (kbps) / Channel length / 2

For TDM modes, the Frame length is always 256 bit clocks, independent of the “Channels” setting. So the Frame rate expression is simplified:

Frame rate (ksps) = Bit rate (kbps) / 256

This value is shown in the **Frame rate (ksps)** field on the Configure dialog. If using an external interface clock, the **Bit rate** and **Frame rate** are unknown, because the external clock source frequency is unknown.

DMA Support

The I2S_PDL Component supports Direct Memory Access (DMA) transfers. The Component may transfer to/from the following sources.

Name of DMA Source / Destination	Length	Direction	DMA Req Signal	DMA Req Type	Description
`I2S_1_TX_FIFO_WR_PTR`	Defined by the TX Trigger Level parameter (max is 256)	Destination	tx_dma	Level	I2S TX data input
`I2S_1_RX_FIFO_RD_PTR`	Defined by the RX Trigger Level parameter (max is 256)	Source	rx_dma	Level	I2S RX data output

Industry Standards

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component-specific deviations. Refer to PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6) for information on MISRA compliance and deviations for files generated by PSoC Creator.

The I2S_PDL Component does not have any specific deviations.

I2S Bus

The Component implements the Philips I²S Bus standard.

Registers

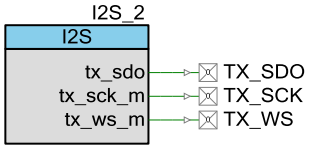
See the I2S Registers section in the chip Technical Reference Manual (TRM) for more information about the registers.

Schematic Macro Information

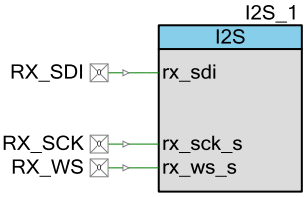
This section contains pertinent information for the schematic macros associated with the I2S_PDL Component.

Schematic Macro Name	Purpose	Description
I2S Tx Master Only	I2S data transmission only.	Transmitter with all the needed pins connected.
I2S Rx Master Only	I2S data reception only.	Receiver with all the needed pins connected.
I2S Tx Master and Rx Master	I2S data transmission and reception.	Transceiver with all the needed pins connected.
I2S Tx Slave Only	I2S data transmission only.	Transmitter with all the needed pins connected.
I2S Rx Slave Only	I2S data reception only.	Receiver with all the needed pins connected.
I2S Tx Slave and Rx Master	I2S data transmission and reception.	Transceiver with all the needed pins connected.
I2S Tx Master and Rx Slave	I2S data transmission and reception.	Transceiver with all the needed pins connected.
I2S Tx Slave and Rx Slave	I2S data transmission and reception.	Transceiver with all the needed pins connected.

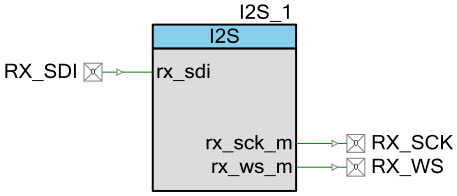
I2S Tx Master Only



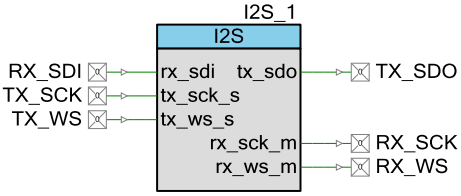
I2S Rx Slave Only



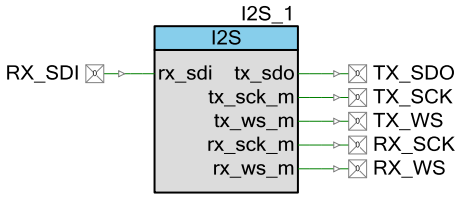
I2S Rx Master Only



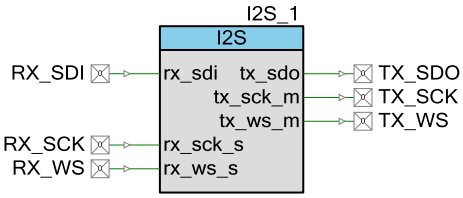
I2S Tx Slave and Rx Master



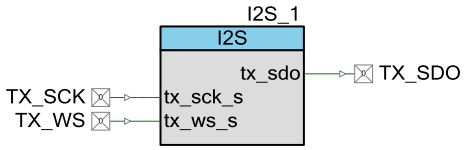
I2S Tx Master and Rx Master



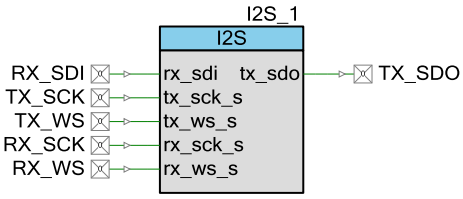
I2S Tx Master and Rx Slave



I2S Tx Slave Only



I2S Tx Slave and Rx Slave



Resources

The I2S_PDL Component uses the I2S fixed-function peripheral block.

DC and AC Electrical Characteristics

Note Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site

References

List of references related to I2S_PDL Component:

- Philips I²S Bus – “Inter-IC Sound Bus” industry standard specification

Component Errata

This section lists known problems with the I2S_PDL Component.

ID	Version	Problem	Workaround
N/A	2.0	This version of the Component doesn't support internally-wired full duplex mode (simultaneous TX and RX with common SCK and WS signals).	Full duplex mode configuration is achievable with external connections of the SCK and WS terminals for TX master and RX slave (or vice-versa, or both slaves).
N/A	1.0	This version of the Component doesn't support I2S slave mode, full duplex mode (simultaneous TX and RX with common SCK and WS signals), or other data formats like Left Justified and TDM.	The slave mode and the mentioned data formats are planned to be supported in the next Component version.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.0.c	Minor datasheet edits.	
2.0.b	Minor datasheet edits.	
2.0.a	Edited the datasheet.	Clarify technical details. Updated MISRA section.
2.0	Slave mode has been added for both TX and RX; Left Justified, TDM-A, and TDM-B data formats are added. The datasheet has been updated.	Major Component version update.
1.0.b	Updated the datasheet.	Updated the block diagram to be more understandable. Added general example of the I2S frame structure. Noted that the Full Duplex mode is not supported in Features section. Extended the code snippet in the Quick Start section.
1.0.a	Edited the datasheet.	Changed numerous sections throughout to reflect more accurate information about the Component.
1.0	New Component	

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical Components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical Component is any Component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

