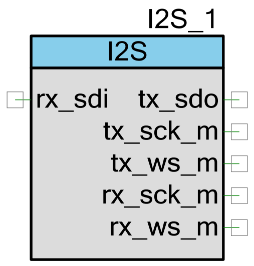
Inter-IC Sound Bus (I2S_PDL)

1.0

Features



- I2S Data Format
- Master only TX/RX operation (Full Duplex mode is not supported)
- Programmable Channel/Word Lengths
- Internal/external Clock Operation
- Hybrid Component (Peripheral Driver Library (PDL) and Component Application Programming Interface (API))

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 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
- 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) [v1.0]” 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:

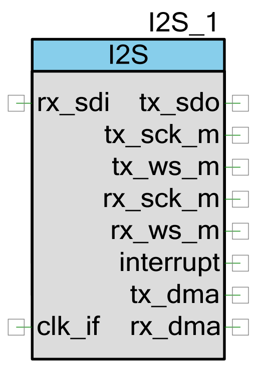
```
 (void)Cy_I2S_Init (I2S_1_HW, &I2S_1_config);
Cy_I2S_ClearRxFifo (I2S_1_HW);
Cy_I2S_ClearTxFifo (I2S_1_HW);
Cy_I2S_WriteTxData(I2S_1_HW, 0UL);
Cy_I2S_WriteTxData(I2S_1_HW, 0UL);
Cy_I2S_RxStart (I2S_1_HW);
Cy_I2S_TxStart (I2S_1_HW);
Cy_I2S_SetInterruptMask (I2S_1_HW, I2S_1_INT_MASK);
```

7. Build the project and program the device.

PRELIMINARY

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.

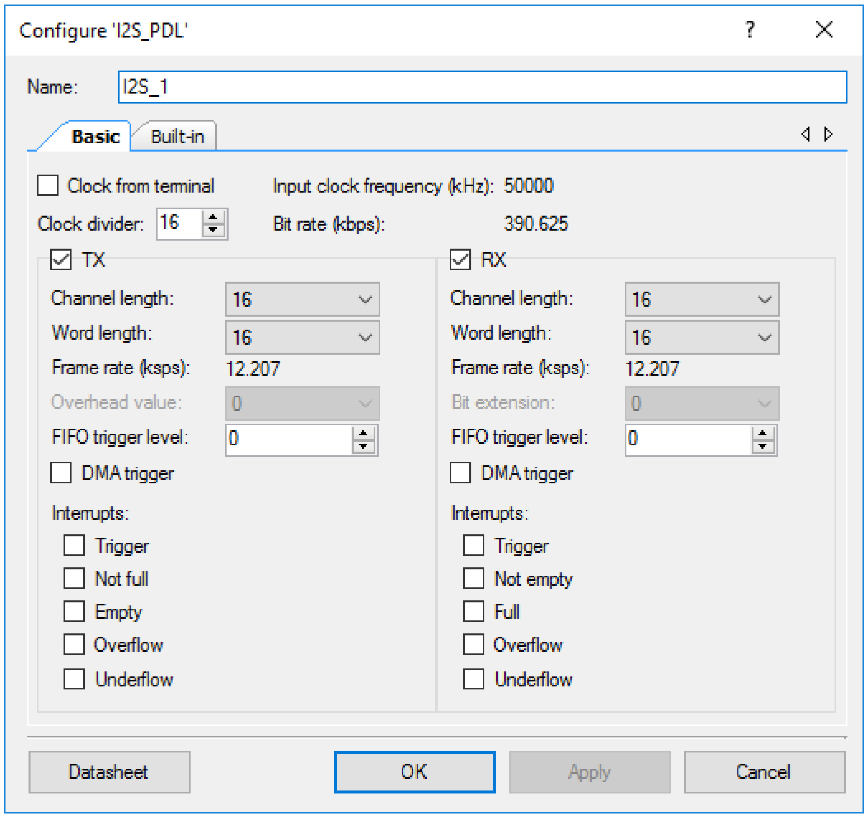


Terminal Name	I/O Type	Description
rx_sdi	Digital Input	Rx serial data input (visible when Rx is enabled)
clk_if	Digital Input	External interface clock (visible when “Clock from terminal” is enabled)
tx_sdo	Digital Output	Tx serial data output (visible when Tx is enabled)
tx_sck_m	Digital Output	Tx serial clock output (visible when Tx is enabled)
tx_ws_m	Digital Output	Tx word select output (visible when Tx is enabled)
tx_sck_m	Digital Output	Tx serial clock output (visible when Tx is enabled)
tx_ws_m	Digital Output	Tx word select output (visible when Tx 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


Configure 'I2S_PDL'

Name: I2S_1

Basic | Built-in

☐ Clock from terminal
Input clock frequency (kHz): 50000
Clock divider: 16
Bit rate (kbps): 390.625

☑ TX
Channel length: 16
Word length: 16
Frame rate (ksps): 12.207
Overhead value: 0
FIFO trigger level: 0
☐ DMA trigger
Interrupts:
☐ Trigger
☐ Not full
☐ Empty
☐ Overflow
☐ Underflow

☑ RX
Channel length: 16
Word length: 16
Frame rate (ksps): 12.207
Bit extension: 0
FIFO trigger level: 0
☐ DMA trigger
Interrupts:
☐ Trigger
☐ Not empty
☐ Full
☐ Overflow
☐ Underflow

Parameter Name	Description
Clock divider	Sets the input Clock Divider
Clock from terminal	Selects input clock source: external interface clock (from the clk_if terminal) or internal clock (HFClk1)
TX DMA trigger	Enables TX DMA trigger
RX DMA trigger	Enables RX DMA trigger
TX Channel length	Set TX channel length (in bits)
TX Word length	Set TX word length (in bits)
RX Channel length	Set RX channel length (in bits)
RX Word length	Set RX word length (in bits)
TX Interrupts: Trigger	Trigger interrupt, specified by TX FIFO trigger level

Parameter Name	Description
TX Interrupts: Not full	Trigger interrupt, if TX FIFO is not full
TX Interrupts: Empty	Trigger interrupt, if TX FIFO is empty
TX Interrupts: Overflow	Trigger interrupt, if TX FIFO is overflowed
TX Interrupts: Underflow	Trigger interrupt, if TX FIFO is underflowed
RX Interrupts: Trigger	Trigger interrupt, specified by RX FIFO trigger level
RX Interrupts: Not empty	Trigger interrupt, if RX FIFO is not empty
RX Interrupts: Full	Trigger interrupt, if RX FIFO is full
RX Interrupts: Overflow	Trigger interrupt, if RX FIFO is overflowed
RX Interrupts: Underflow	Trigger interrupt, if RX FIFO is underflowed
TX Overhead value	Set the overhead bits level (active when word length is less than channel length)
TX FIFO trigger level	Set TX FIFO level for event trigger (interrupt or DMA request)
RX Bit extension	Set the extension bits level (active when word length is less than channel length)
RX FIFO trigger level	Set RX FIFO level for event trigger (interrupt or DMA request)

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the Component using software.

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**.

This Component uses 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.

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

Currently, there are no code examples developed for the I2S_PDL Component. TBD.

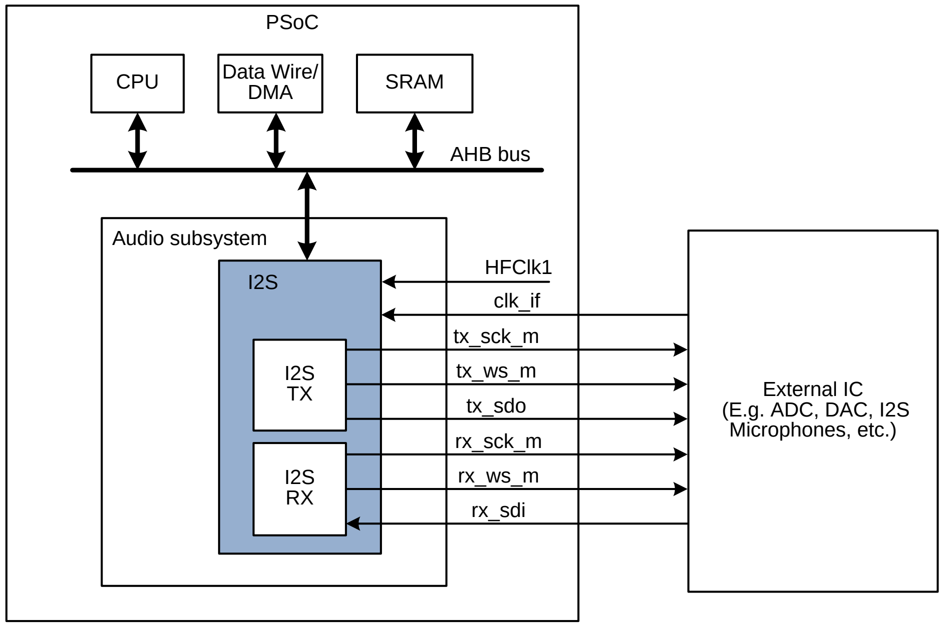
Functional Description

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.

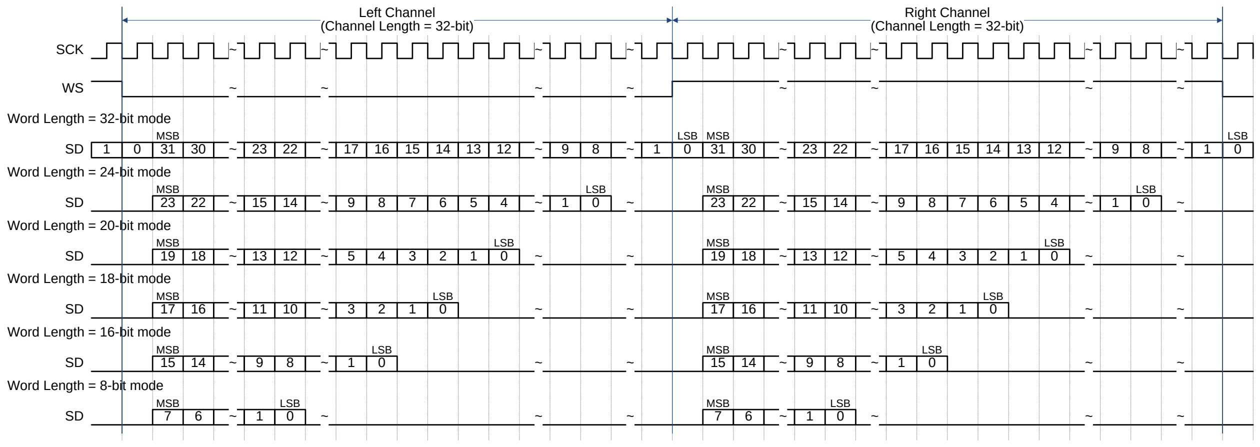
The transmitter and receiver each have their own I2S interface, and each operates as an I2S master (i.e. the I2S bus clock SCK and word select WS signals are driven by the master).

Block Diagram and Configuration

The following is a simplified diagram of the I2S hardware:



The I2S frame looks like the following:


Left Channel
(Channel Length = 32-bit)
Right Channel
(Channel Length = 32-bit)
SCK
WS
Word Length = 32-bit mode
Word Length = 24-bit mode
Word Length = 20-bit mode
Word Length = 18-bit mode
Word Length = 16-bit mode
Word Length = 8-bit mode
SD
MSB
LSB

This is an example for channel length = 32. It is similar to other channel lengths, except the word length could be less than or equal to the channel length. See 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.

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 2 channels). This value is indicated by the **Frame rate (ksps)** field on the Configure dialog.

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

If using an external interface clock, the **Bit rate** and **Frame rate** are unknown, because the external clock source frequency is unknown.

PRELIMINARY

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. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The I2S_PDL Component does not have any specific deviations.

I2S Bus

The Component implements the Philips I^{2}S Bus standard.

Registers

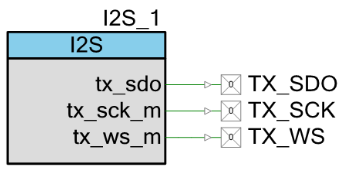
See the I2S Registers section in the chip Technical Reference Manual (TRM) for more information about the registers.

Schematic Macro Information

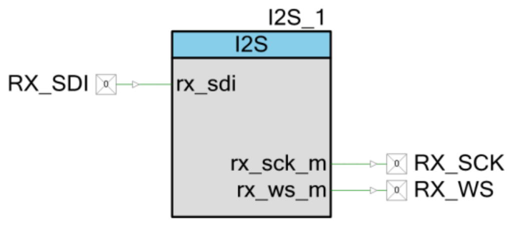
This section contains pertinent information for the schematic macros associated with the I2S_PDL.

Schematic Macro Name	Purpose	Description
I2S Tx Master Only	Use for a system requiring I2S data transmission only.	The I2S_PDL Component configured as a transmitter with all the needed pins connected.
I2S Rx Master Only	Use for a system requiring I2S data reception only.	The I2S_PDL Component configured as a receiver with all the needed pins connected.
I2S Tx Master and Rx Master	Use for a system requiring I2S data transmission and reception.	The I2S_PDL Component configured as a transceiver with all the needed pins connected.

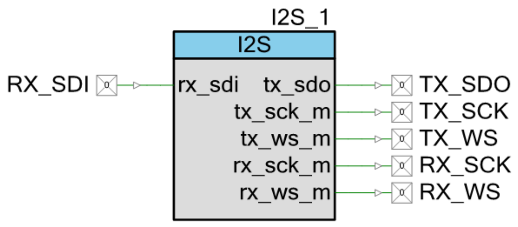
I2S Tx Master Only



I2S Rx Master Only



I2S Tx Master and Rx Master



Resources

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

PRELIMINARY

DC and AC Electrical Characteristics

Characterization data of the I2S_PDL Component. TBD.

References

List of references related to I2S_PDL Component:

- Philips I^{2}S Bus – "Inter-IC Sound Bus" industry standard specification

Component Errata

This section lists known problems with the I2S_PDL Component.

Cypress ID	Component Version	Problem	Workaround
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
1.0.b	The datasheet is enhanced.	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	