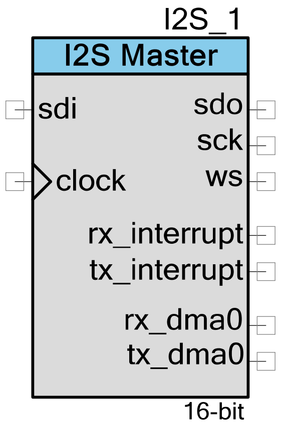
Inter-IC Sound Bus (I2S)

2.20

Features



- Master only
- 8 to 32 data bits per sample
- 16-, 32-, 48-, or 64-bit word select period
- Data rate up to 96 kHz with 64-bit word select period: 6.144 MHz
- Tx and Rx FIFO interrupts
- DMA support
- Independent left and right channel FIFOs or interleaved stereo FIFOs
- Independent enable of Rx and Tx

General Description

The Integrated Inter-IC Sound Bus (I2S) is a serial bus interface standard used for connecting digital audio devices together. The specification is from Philips® Semiconductor (I2S bus specification; February 1986, revised June 5, 1996).

The I2S component operates in master mode only. It also operates in two directions: as a transmitter (Tx) and a receiver (Rx). The data for Tx and Rx are independent byte streams. The byte streams are packed with the most significant byte first and the most significant bit in bit 7 of the first word. The number of bytes used for each sample (a sample for the left or right channel) is the minimum number of bytes to hold a sample.

When to Use an I2S

The component provides a serial bus interface for stereo audio data. This interface is most commonly used by audio ADC and DAC components.

Input/Output Connections

This section describes the various input and output connections for the I2S component. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

sdi – Input *

Serial data input. Displays if you select an Rx option for the **Direction** parameter.

If this signal is connected to an input pin, then the "Input Synchronized" selection for this pin should be disabled. This signal should already be synchronized to SCK, so delaying the signal with the input pin synchronizer could cause the signal to be shifted into the next clock cycle.

clock – Input

The clock rate provided must be two times the desired clock rate for the output serial clock (SCK). For example to produce 48-kHz audio with a 64-bit word select period, the clock frequency would be:

$$2 \times 48\ \text{kHz} \times 64 = 6.144\ \text{MHz}$$

sdo – Output *

Serial data output. Displays if you select a Tx option for the **Direction** parameter.

sck – Output

Output serial clock.

ws – Output

Word select output indicates the channel being transmitted.

rx_interrupt – Output *

Rx direction interrupt. Displays if you select an Rx option for the **Direction** parameter.

tx_interrupt – Output *

Tx direction interrupt. Displays if you select a Tx option for the **Direction** parameter.

rx_DMA0 – Output *

Rx direction DMA request for FIFO 0 (Left or Interleaved). Displays if you select **Rx DMA** under the **DMA Request** parameter.

rx_DMA1 – Output *

Rx direction DMA request for FIFO 1 (Right). Displays if you select **Rx DMA** under the **DMA Request** parameter and **Separated L/R** under the **Data Interleaving** parameter for Rx.

tx_DMA0– Output *

Tx direction DMA request for FIFO 0 (Left or Interleaved). Displays if you select **Tx DMA** under the **DMA Request** parameter.

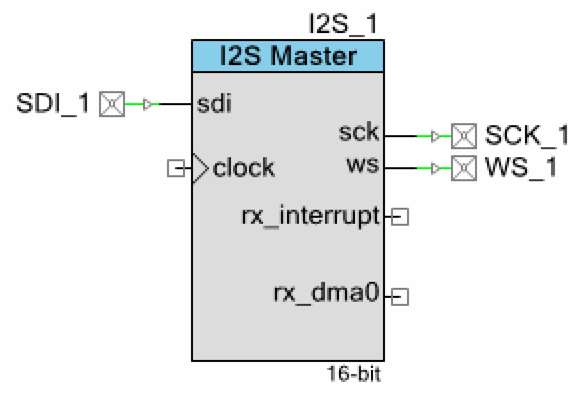
tx_DMA1 – Output *

Tx direction DMA request for FIFO 1 (Right). Displays if you select **Tx DMA** under the **DMA Request** parameter and **Separated L/R** under the **Data Interleaving** parameter for Tx.

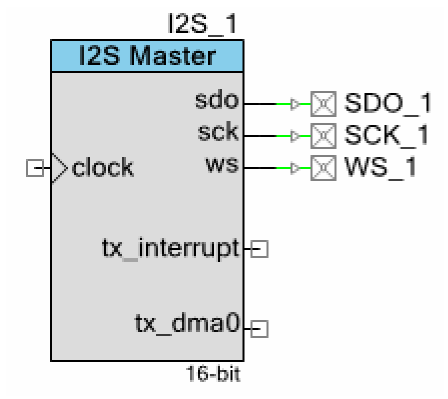
Schematic Macro Information

By default, the PSoC Creator Component Catalog contains three Schematic Macro implementations for the I2S component. These macros contain the I2S component already connected to digital pin components. The “Input Synchronized” option is unchecked on the SDI pin and the generation of APIs for all of the pins is turned off. The Schematic Macros use the I2S component, configured for Rx only, Tx only, and both Rx and Tx directions, as shown in the following diagrams.

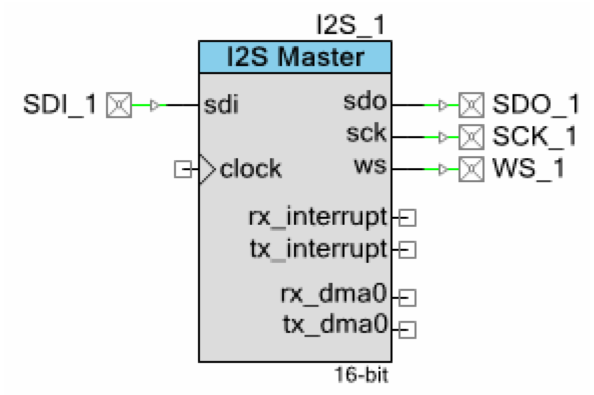
I2S Rx Only



I2S Tx Only



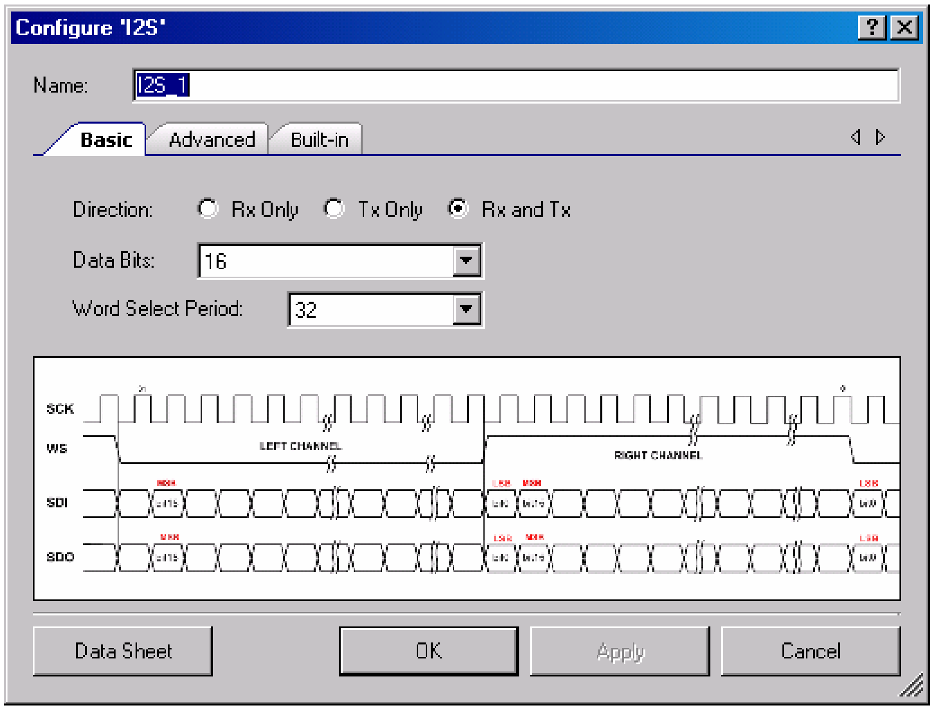
I2S Rx and Tx



Component Parameters

Drag an I2S component onto your design and double-click it to open the **Configure** dialog. This dialog has two tabs to guide you through the process of setting up the I2S component.

Basic Tab



Name: I2S_1

Direction: ( ) Rx Only ( ) Tx Only [✓ Rx and Tx]

Data Bits: 16

Word Select Period: 32

Direction

Determines which direction the component operates. This value can be set to **Rx Only**, **Tx Only**, or **Rx and Tx** (default).

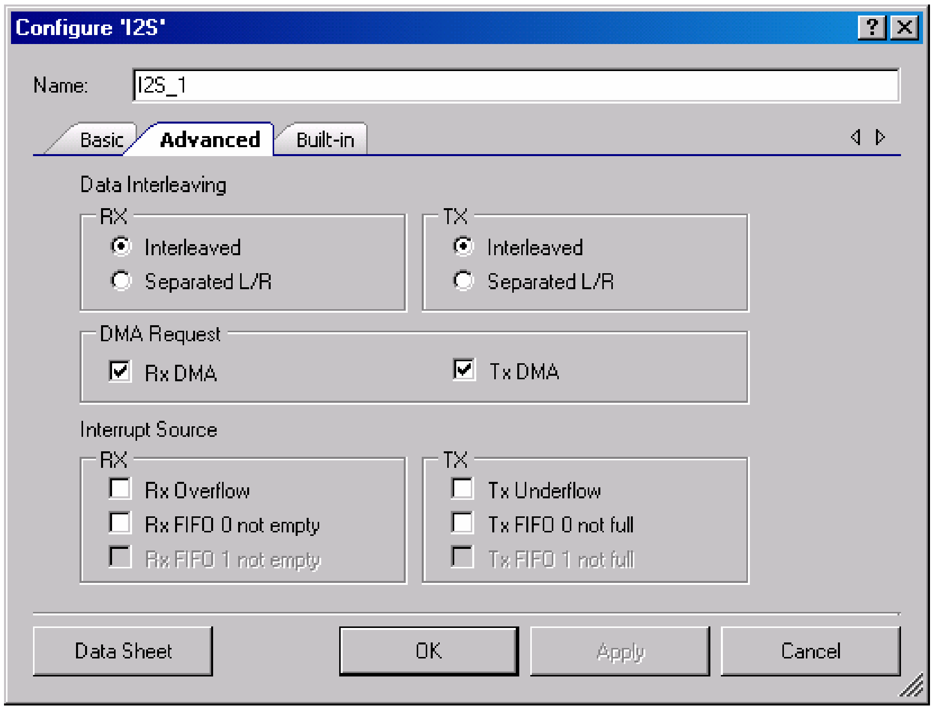
Data Bits

Determines the number of data bits configured for each sample (hardware compiled). This value can be set between 8 and 32. The default setting is **16**.

Word Select Period

Defines the period of a complete sample of both left and right channels. This value can be set to: **16**, **32** (default), **48**, or **64**.

Advanced Tab



Name: I2S_1

Data Interleaving

RX: [✓ Interleaved] Separated L/R

TX: [✓ Interleaved] Separated L/R

DMA Request

☑ Rx DMA ☑ Tx DMA

Interrupt Source

RX:
☐ Rx Overflow
☐ Rx FIFO 0 not empty
☐ Rx FIFO 1 not empty

TX:
☐ Tx Underflow
☐ Tx FIFO 0 not full
☐ Tx FIFO 1 not full

Data Interleaving

Allows you to select whether the data is **Interleaved** (default) or **Separate L/R**. You select Rx and Tx independently.

DMA Request

Allows you to enable and disable the DMA request signals for the component. You set Rx and Tx independently. These options are enabled by default.

Interrupt Source

Select the source of the I2S interrupts. Rx and Tx interrupts are separate. Multiple sources may be ORed together. Settings include:

- Rx:
  - **Rx Overflow**
  - **Rx FIFO 0 not empty** (Left or Interleaved)
  - **Rx FIFO 1 not empty** (Right) - Only an option if not Interleaved
- Tx:
  - **Tx Underflow**
  - **Tx FIFO 0 not full** (Left or Interleaved)
  - **Tx FIFO 1 not full** (Right) - Only an option if not Interleaved

Clock Selection

There is no internal clock in this component. You must attach a clock source. The clock rate provided must be two times the desired clock rate for the output serial clock (SCK).

Placement

The I2S component is placed throughout the UDB array. All placement information is provided to the API through the *cyfitter.h* file.

Resources

	Resource Type				API Memory (Bytes)		
Resources	**Datapath Cells**	**PLDs**[1]	**Status Cells**	**Control/ Count7 Cells**	**Flash**	**RAM**	**Pins (per External I/O)**
Rx Direction	1	2(3)	1	2	220	3	3
Tx Direction	1	2(3)	1	2	220	3	3
Rx and Tx	2	4(6)	2	2	326	3	4

[1] The numbers of PLDs indicate typical and maximum resource use for the specified configuration.

Application Programming Interface

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

By default, PSoC Creator assigns the instance name "I2S_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 "I2S."

Function	Description
I2S_Start()	Starts the I2S interface.
I2S_Stop()	Disables the I2S interface.
I2S_EnableTx()	Enables the Tx direction of the I2S interface.
I2S_DisableTx()	Disables the Tx direction of the I2S interface.
I2S_EnableRx()	Enables the Rx direction of the I2S interface.
I2S_DisableRx()	Disables the Rx direction of the I2S interface.
I2S_SetRxInterruptMode()	Sets the interrupt source for the I2S Rx direction interrupt.
I2S_SetTxInterruptMode()	Sets the interrupt source for the I2S Tx direction interrupt.
I2S_ReadRxStatus()	Returns state in the I2S Rx status register.
I2S_ReadTxStatus()	Returns state in the I2S Tx status register.
I2S_ReadByte()	Returns a single byte from the Rx FIFO.
I2S_WriteByte()	Writes a single byte into the Tx FIFO.
I2S_ClearRxFIFO()	Clears out the Rx FIFO.
I2S_ClearTxFIFO()	Clears out the Tx FIFO.
I2S_Sleep()	Saves configuration and disables the I2S interface
I2S_WakeUp()	Restores configuration and enables the I2S interface
I2S_Init()	Enables the I2S interface
I2S_Enable()	Initializes or restores default I2S configuration
I2S_SaveConfig()	Saves configuration of I2S interface
I2S_RestoreConfig()	Restores configuration of I2S interface

Global Variables

Variable	Description
I2S_initVar	Indicates whether the I2S has been initialized. The variable is initialized to 0 and set to 1 the first time I2S_Start() is called. This allows the component to restart without reinitialization in after the first call to the I2S_Start() routine. If reinitialization of the component is required call I2S_Init() before calling I2S_Start(). Alternatively, the I2S can be reinitialized by calling the I2S_Init() and I2S_Enable() functions.

void I2S_Start(void)

Description: Starts the I2S interface. Enables Active mode power template bits or clock gating as appropriate. Starts the generation of the sck and ws outputs. The Tx and Rx directions remain disabled.

Parameters: None

Return Value: None

Side Effects: None

void I2S_Stop(void)

Description: Disables the I2S interface. Disables Active mode power template bits or clock gating as appropriate. The sck and ws outputs are set to 0. The Tx and Rx directions are disabled and their FIFOs are cleared.

Parameters: None

Return Value: None

Side Effects: None

void I2S_EnableTx(void)

Description: Enables the Tx direction of the I2S interface. Transmission begins at the next word select falling edge.

Parameters: None

Return Value: None

Side Effects: None

void I2S_DisableTx(void)

Description: Disables the Tx direction of the I2S interface. Transmission of data stops and a constant 0 value is transmitted at the next word select falling edge.

Parameters: None

Return Value: None

Side Effects: None

void I2S_EnableRx(void)

Description: Enables the Rx direction of the I2S interface. Data reception begins at the next word select falling edge.

Parameters: None

Return Value: None

Side Effects: None

void I2S_DisableRx(void)

Description: Disables the Rx direction of the I2S interface. At the next word select falling edge, data reception is no longer sent to the receive FIFO.

Parameters: None

Return Value: None

Side Effects: None

void I2S_SetRxInterruptMode(uint8 interruptSource)

Description: Sets the interrupt source for the I2S Rx direction interrupt. Multiple sources may be ORed.

Parameters: uint8: Byte containing the constant for the selected interrupt sources

I2S Rx Interrupt Source	Value
RX_FIFO_OVERFLOW	0x01
RX_FIFO_0_NOT_EMPTY	0x02
RX_FIFO_1_NOT_EMPTY	0x04

Return Value: None

Side Effects: None

void I2S_SetTxInterruptMode(uint8 interruptSource)

Description: Sets the interrupt source for the I2S Tx direction interrupt. Multiple sources may be ORed.

Parameters: uint8: Byte containing the constant for the selected interrupt sources

I2S Tx Interrupt Source	Value
TX_FIFO_UNDERFLOW	0x01
TX_FIFO_0_NOT_FULL	0x02
TX_FIFO_1_NOT_FULL	0x04

Return Value: None

Side Effects: None

uint8 I2S_ReadRxStatus(void)

Description: Returns state of the I2S Rx status register.

Parameters: None

Return Value: uint8: State of the I2S Rx status register

I2S Rx Status Masks	Value	Type
RX_FIFO_OVERFLOW	0x01	Clear-on-Read
RX_FIFO_0_NOT_EMPTY	0x02	Transparent
RX_FIFO_1_NOT_EMPTY	0x04	Transparent

Side Effects: Clears the bits of the I2S Rx status register that are clear-on-read.

uint8 I2S_ReadTxStatus(void)

Description: Returns state of the I2S Tx status register.

Parameters: None

Return Value: uint8: State of the I2S Tx status register

I2S Tx Status Masks	Value	Type
TX_FIFO_UNDERFLOW	0x01	Clear-on-Read
TX_FIFO_0_NOT_FULL	0x02	Transparent
TX_FIFO_1_NOT_FULL	0x04	Transparent

Side Effects: Clears the bits of the I2S Rx status register that are clear on read.

uint8 I2S_ReadByte(uint8 wordSelect)

Description:	Returns a single byte from the Rx FIFO. Check the Rx status before this call to confirm that the Rx FIFO is not empty.
Parameters:	uint8: Indicates to read from the Left (0) or Right (1) channel. In interleaved mode, this parameter is ignored.
Return Value:	uint8: Byte containing the data received
Side Effects:	None

void I2S_WriteByte(uint8 wrData, uint8 wordSelect)

Description:	Writes a single byte into the Tx FIFO. Check the Tx status before this call to confirm that the Tx FIFO is not full.
Parameters:	uint8 wrData: Byte containing the data to transmit
	uint8 wordSelect: Indicates to write to the Left (0) or Right (1) channel. In interleaved mode, this parameter is ignored
Return Value:	None
Side Effects:	None

void I2S_ClearRxFIFO(void)

Description:	Clears out the Rx FIFO. Any data present in the FIFO is lost. Call this function only when the Rx direction is disabled.
Parameters:	None
Return Value:	None
Side Effects:	None

void I2S_ClearTxFIFO(void)

Description:	Clears out the Tx FIFO. Any data present in the FIFO is lost. Call this function only when the Tx direction is disabled.
Parameters:	None
Return Value:	None
Side Effects:	None

void I2S_Sleep(void)

Description:	This is the preferred routine to prepare the component for sleep. The I2S_Sleep() routine saves the current component state. Then it calls the I2S_Stop() function and calls I2S_SaveConfig() to save the hardware configuration. Disables Active mode power template bits or clock gating as appropriate. The sck and ws outputs are set to 0. The Tx and Rx directions are disabled. Call the I2S_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the PSoC Creator *System Reference Guide* for more information about power management functions.
Parameters:	None
Return Value:	None
Side Effects:	None

void I2S_WakeUp(void)

Description:	Restores I2S configuration and nonretention register values. Enables Active mode power template bits or clock gating, as appropriate. Starts the generation of the sck and ws outputs. Enables Rx and/or Tx direction according to their states before sleep.
Parameters:	None
Return Value:	None
Side Effects:	Calling the I2S_Wakeup() function without first calling the I2S_Sleep() or I2S_SaveConfig() function may produce unexpected behavior.

void I2S_Init(void)

Description:	Initializes or restores default I2S configuration provided with customizer that defines interrupt sources for the component.
Parameters:	None
Return Value:	None
Side Effects:	Restores only mask registers for interrupt generation. It does not clear data from the FIFOs and does not reset component hardware state machines.

void I2S_Enable(void)

Description:	Activates the hardware and begins component operation. It is not necessary to call I2S_Enable() because the I2S_Start() routine calls this function, which is the preferred method to begin component operation.
Parameters:	None
Return Value:	None
Side Effects:	None

void I2S_SaveConfig(void)

Description:	This function saves the component configuration and nonretention registers. It also saves the current component parameter values, as defined in the Configure dialog or as modified by appropriate APIs. This function is called by the I2S_Sleep() function.
Parameters:	None
Return Value:	None
Side Effects:	None

void I2S_RestoreConfig(void)

Description:	This function restores the component configuration and nonretention registers. It also restores the component parameter values to what they were prior to calling the I2S_Sleep() function.This routines is called by I2S_Wakeup() to restore the component when it exits sleep.
Parameters:	None
Return Value:	None
Side Effects:	Must be called only after I2S_SaveConfig() routine. Otherwise, the component configuration will be overwritten with its initial setting.

Sample Firmware Source Code

PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project 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 Example Project” topic in the PSoC Creator Help for more information.

Functional Description

Left/Right and Rx/Tx Configuration

The configuration for the Left and Right channels—the Rx and Tx direction, number of bits, and word-select period—are identical. If the application must have different configurations for the Rx and Tx, then you should use two unidirectional component instances.

Data Stream Format

The data for Tx and Rx is independent byte streams. The byte streams are packed with the most significant byte first and the most significant bit in bit 7 of the first word. The number of bytes used for each sample (for the left or right channel) is the minimum number of bytes to hold a sample. Any unused bits will be ignored on Tx and will be 0 on Rx.

The data stream for one direction can be a single byte stream, or it can be two byte streams. In the case of a single byte stream, the left and right channels are interleaved with a sample for the left channel first followed by the right channel. In the two-stream case, the left and right channel byte streams use separate FIFOs.

DMA

The I2S interface is a continuous interface that requires an uninterrupted stream of data. For most applications, this requires the use of DMA transfers to prevent the underflow of the Tx direction or the overflow of the Rx direction.

The I2S can drive up to two DMA components for each direction. You can use the DMA Wizard to configure DMA operation as follows:

Name of DMA Source/ Destination in the DMA Wizard	Direction	DMA Request Signal	DMA Request Type	Description
I2S_RX_FIFO_0_PTR	Source	rx_dma0	Level	Receive FIFO for Left or Interleaved channel
I2S_RX_FIFO_1_PTR	Source	rx_dma1	Level	Receive FIFO for Right channel
I2S_TX_FIFO_0_PTR	Destination	tx_dma0	Level	Transmit FIFO for Left or Interleaved channel
I2S_TX_FIFO_1_PTR	Destination	tx_dma1	Level	Transmit FIFO for Right channel

In all cases, a high signal on the DMA request signal indicates that an additional single byte may be transferred.

Enabling

The Rx and Tx directions have separate enables. When not enabled, the Tx direction transmits all 0 values, and the Rx direction ignores all received data. The transition into and out of the enabled state occurs at a word select boundary such that a left/right sample pair is always transmitted or received.

Error Handling

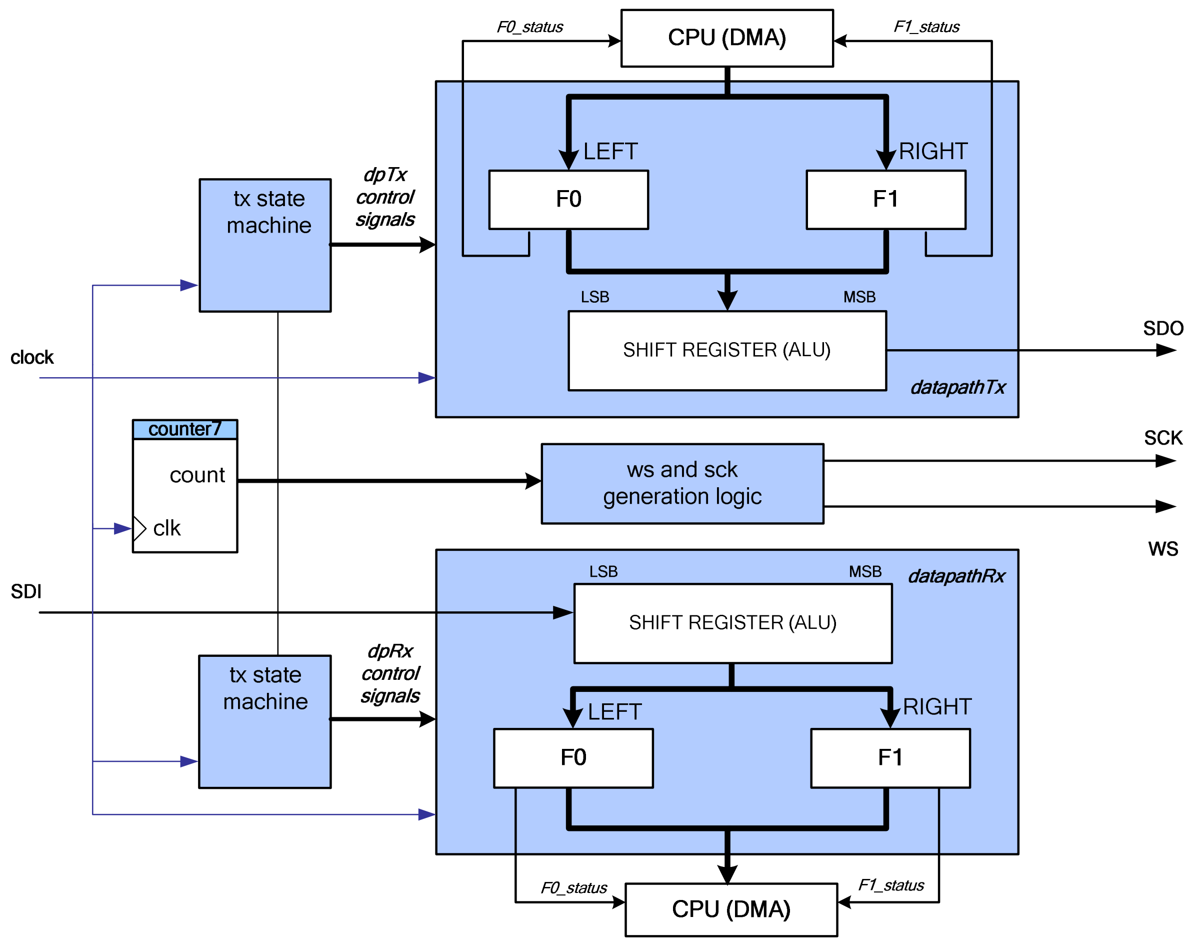
Two error conditions for the component can occur if the transmit FIFO is empty and a subsequent read occurs (transmit underflow), or the receive FIFO is full and a subsequent write occurs (receive overflow).

While transmission is enabled if the transmit FIFO becomes empty and data is not available for transmission (Transmit underflow), the component will force the constant transmission of 0s. Before transmission begins again, transmission must be disabled, the FIFOs should be cleared, data for transmit must be buffered, and then transmission re-enabled. The CPU can monitor this underflow condition using the transmit status bit I2S_TX_FIFO_UNDERFLOW. An interrupt can also be configured for this error condition.

While reception is enabled if the receive FIFO becomes full and additional data is received (Receive overflow), the component stops capturing data. Before reception begins again, reception must be disabled, the FIFOs should be cleared, and then reception re-enabled. The CPU can monitor this overflow condition using the receive status bit I2S_RX_FIFO_OVERFLOW. An interrupt can also be configured for this error condition.

Block Diagram and Configuration

The I2S is implemented as a set of configured UDBs. The implementation is shown in the following block diagram.



Registers

I2S_CONTROL_REG

Bits	7	6	5	4	3	2	1	0
Value	reserved					enable	rxenable	txenable

- enable: Enable/Disable I2S component
- rxenable, txenable: Enable/Disable Rx and Tx directions respectively

I2S_TX_STATUS_REG

Bits	7	6	5	4	3	2	1	0
Value	reserved					F1_not_full	F0_not_full	underflow

- F1_not_full: If set, Tx FIFO 1 is not full
- F0_not_full: If set, Tx FIFO 0 is not full
- underflow: If set, Tx FIFOs underflow event has occurred

The register value can be read with the I2S_ReadTxStatus() API function.

I2S_RX_STATUS_REG

Bits	7	6	5	4	3	2	1	0
Value	reserved					F1_not_empty	F0_not_empty	overflow

- F1_not_empty: If set, Rx FIFO 1 is not empty
- F0_not_empty: If set, Rx FIFO 0 is not empty
- overflow: If set, Rx FIFOs overflow event has occurred

The register value can be read with the I2S_ReadRxStatus() API function.

DC and AC Electrical Characteristics

The following values indicate expected performance and are based on initial characterization data.

Timing Characteristics "Maximum with Nominal Routing"

Parameter	Description	Min	Typ	Max [1]	Unit
f_S	Sampling frequency[2]	–	–	96	kHz
t_{WS}	Word select period	16	–	64	bits
f_{SCK}	Serial clock (output) frequency	–	$f_S \times t_{WS}$	6.144	MHz
f_{CLOCK}	Component clock frequency	–	–	$2 \times f_{SCK}$	MHz

[1] The maximum component clock frequency is derived from t_{CLK_SCK} in combination with the routing path delays of the SCK output and the SDI input (described later in this document). These "Nominal" numbers provide a maximum safe operating frequency of the component under nominal routing conditions. The component can run at higher clock frequencies, at which point you will need to validate the timing requirements with STA results.

[2] Sampling frequency is given for the maximum word select period.

Parameter	Description	Min	Typ	Max [1]	Unit
t_{SCKH}	SCK high-level time	–	0.5	–	$1/f_{SCLK}$
t_{SCKL}	SCK low-level time	–	0.5	–	$1/f_{SCLK}$
t_{SCK_WS}	Delay time, SCK falling edge to WS valid	–20	–	20	ns
t_{SCK_SDO}	Delay time, SCK falling edge to SDO valid	–20	–	20	ns
t_{WS_SDO}	Delay time, WS edge to SDO valid	–20	–	20	ns
t_{S_SDI}	SDI setup time	25	–	–	ns

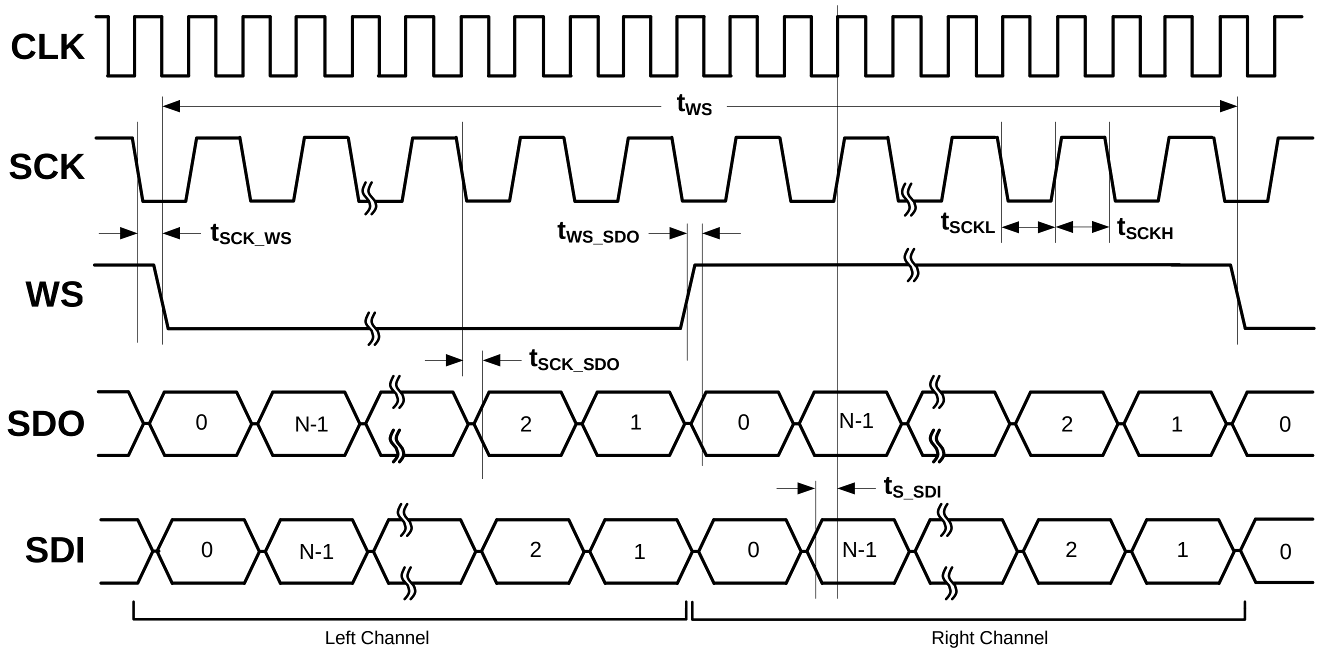
Timing Characteristics “Maximum with All Routing”

Parameter	Description	Min	Typ	Max [1]	Unit
f_S	Sampling frequency [2]	–	–	96	kHz
t_{WS}	Word select period	16	–	64	bits
f_{SCK}	Serial clock (output) frequency	–	$f_S \times t_{WS}$	6.144	MHz
f_{CLOCK}	Component clock frequency	–	–	$2 \times f_{SCK}$	MHz
t_{SCKH}	SCK high-level time	–	0.5	–	$1/f_{SCLK}$
t_{SCKL}	SCK low-level time	–	0.5	–	$1/f_{SCLK}$
t_{SCK_WS}	Delay time, SCK falling edge to WS valid	–20	–	20	ns
t_{SCK_SDO}	Delay time, SCK falling edge to SDO valid	–20	–	20	ns
t_{WS_SDO}	Delay time, WS edge to SDO valid	–20	–	20	ns
t_{S_SDI}	SDI setup time	25	–	–	ns

[1] Maximum for “All Routing” is equal to “Nominal” because the clock frequency is not a limiting factor for this component. The main limiting factor is the round trip path delay from the falling edge of SCK at the pin of the master, to the slave and the path delay of the SDO output of the slave, back to the SDI input of the master (described later in this document).

[2] Sampling frequency is given for the maximum word select period.

Figure 1. Data Transition Timing Diagram



Note All of the component internal logic operates entirely from the input 2X clock. This causes the need to reference some timing constraints, such as t_{S_SDI} and t_{CLK_SCK}, with respect to this internal clock (described later in this document).

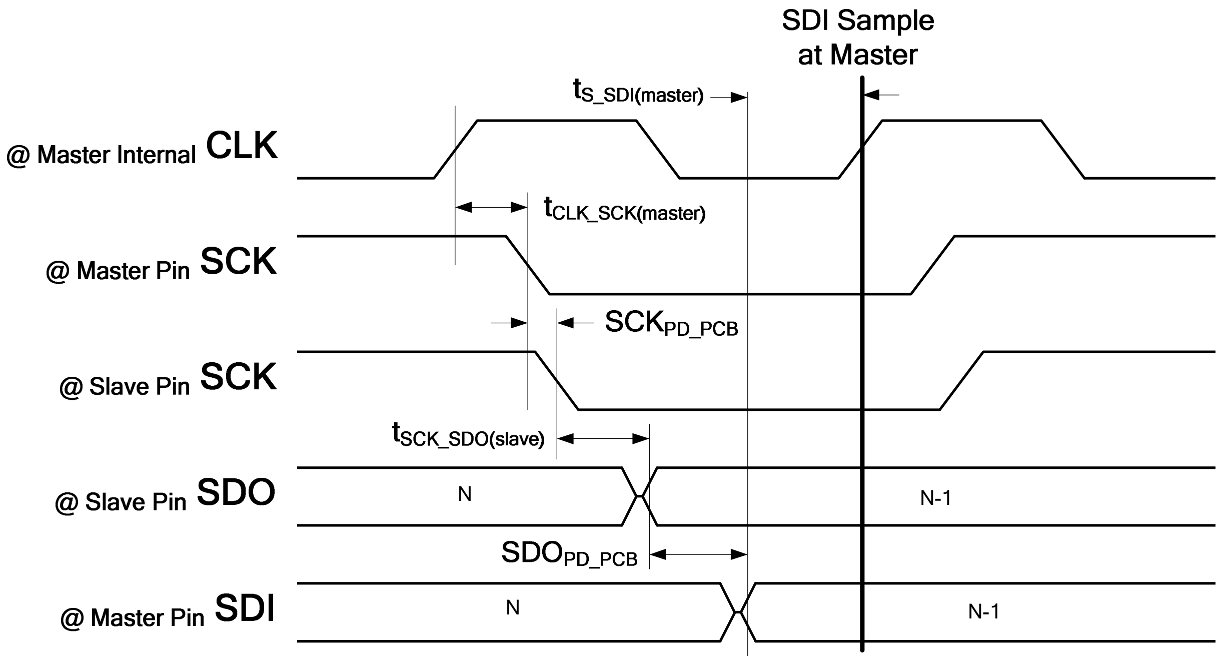
Component Clock Rates for Common Audio Sampling Frequencies

Sampling Frequency	Component Clock Frequency (f_{CLK}) MHz			
	t_{WS} = 16 bits	t_{WS} = 32 bits	t_{WS} = 48 bits	t_{WS} = 64 bits
8 kHz	0.2560	0.5120	0.7680	1.0240
16 kHz	0.5120	1.0240	1.5360	2.0480
32 kHz	1.0240	2.0480	3.0720	4.0960
44.1 kHz	1.4112	2.8224	4.2336	5.6448
48 kHz	1.5360	3.0720	4.6080	6.1440
88.2 kHz	2.8224	5.6448	8.4672	11.2896
96 kHz	3.0720	6.1440	9.2160	12.2880
192 kHz	6.1440	12.2880	N/A	N/A

How to Use STA results for Characteristics Data

f_{SCK} The maximum frequency of SCK (or the maximum bit rate) is not provided directly in the STA. However, the data provided in the STA results indicates some of the internal logic timing constraints. To calculate the maximum bit rate, you must consider several factors. You need board layout and slave communication device specs to fully understand the maximum. The main limiting factor in this parameter is the round trip path delay from the falling edge of SCK at the pin of the master, to the slave and the path delay of the SDO output of the slave, back to the SDI input of the master. In this case, the component must meet the setup time of SDI at the Master with the following equation.

Figure 2. Calculating Maximum f_{SCK} Frequency



In this case, the f_{SCK} frequency must be calculated using the equation below:

$$f_{SCK} < 1 \div [2 \times [t_{RT\_PD} + t_{CLK\_SCK(master)} + t_{S\_SDI(master)}]]$$

Where

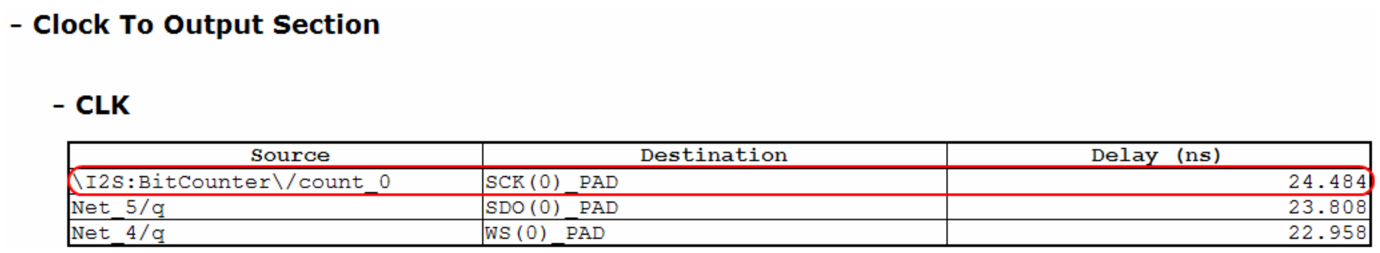
$$t_{RT\_PD} = [SCK_{PD\_PCB} + t_{SCK\_SDO(slave)} + SDO_{PD\_PCB}]$$

and:

SCK_{PD_PCB} is the PCB path delay of SCK from the pin of the master component to the pin of the slave device.

$t_{SCK\_SDO(slave)}$ must come from the Slave Device datasheet

$t_{CLK\_SCK(master)}$ is the internal CLK to SCK pin path delay of the master component. This is provided in the STA results clock to output section, as shown below:

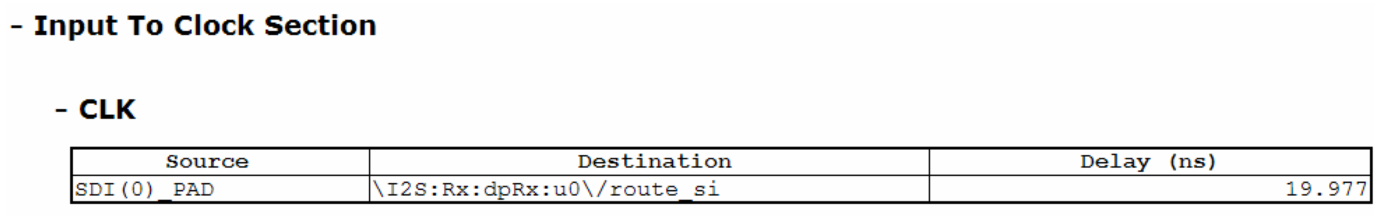


- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

$t_{S\_SDI(master)}$ is the SDI pin to the internal logic path delay of the master component. This is provided in the STA results input to clock section as shown below:



- Input To Clock Section

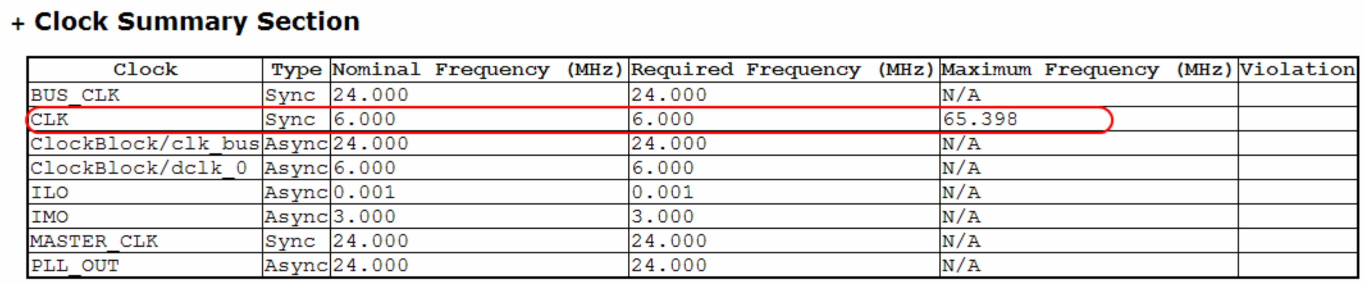
- CLK

Source	Destination	Delay (ns)
SDI(0)_PAD	\I2S:Rx:dpRx:u0\/route_si	19.977

The final equation that provides the maximum frequency of SCK, and therefore the maximum bit rate, is:

$$f_{SCK}\,(\text{Max.}) = 1 \div [2 \times [t_{CLK\_SCK(master)} + SCK_{PD\_PCB} + t_{SCK\_SDO(slave)} + SDO_{PD\_PCB} + t_{S\_SDI(master)}]]$$

f_{CLOCK} Maximum component clock frequency is provided in Timing results in the clock summary as the named external clock (CLK in this case). An example of the internal clock limitations from the STA report is below:



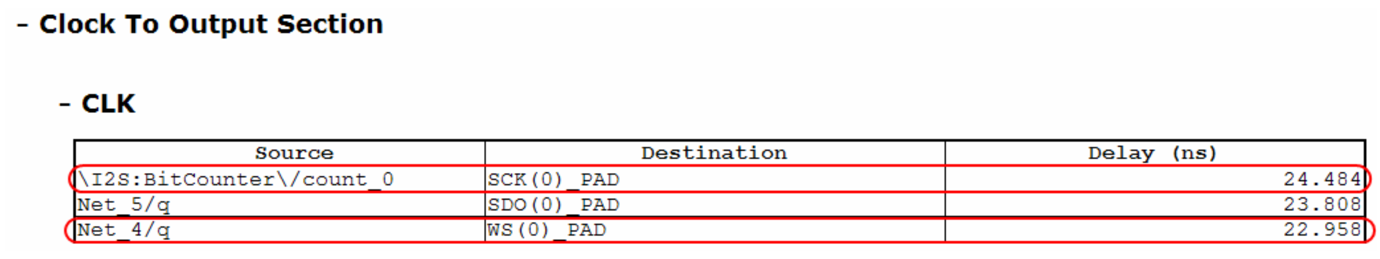
+ Clock Summary Section

Clock	Type	Nominal Frequency (MHz)	Required Frequency (MHz)	Maximum Frequency (MHz)	Violation
BUS_CLK	Sync	24.000	24.000	N/A	
CLK	Sync	6.000	6.000	65.398	
ClockBlock/clk_bus	Async	24.000	24.000	N/A	
ClockBlock/dclk_0	Async	6.000	6.000	N/A	
ILO	Async	0.001	0.001	N/A	
IMO	Async	3.000	3.000	N/A	
MASTER_CLK	Sync	24.000	24.000	N/A	
PLL_OUT	Async	24.000	24.000	N/A	

t_{SCKH} The I2S component generates a 50-percent duty cycle SCK

t_{SCKL} The I2S component generates a 50-percent duty cycle SCLK

t_{SCK_WS} The delay between SCK falling edge and WS valid. This value can be calculated as the difference between clock to output times for WS and SCK pins. The data can be extracted from the STA report, as shown below:

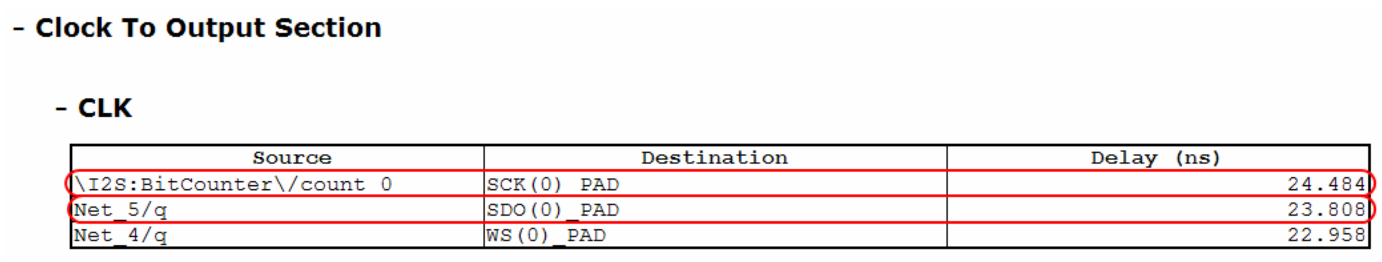


- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{SCK_SDO} The delay between SCK falling edge and WS valid. This value can be calculated as the difference between clock to output times for SDO and SCK pins. The values are provided in the STA results clock to output times, as shown below:

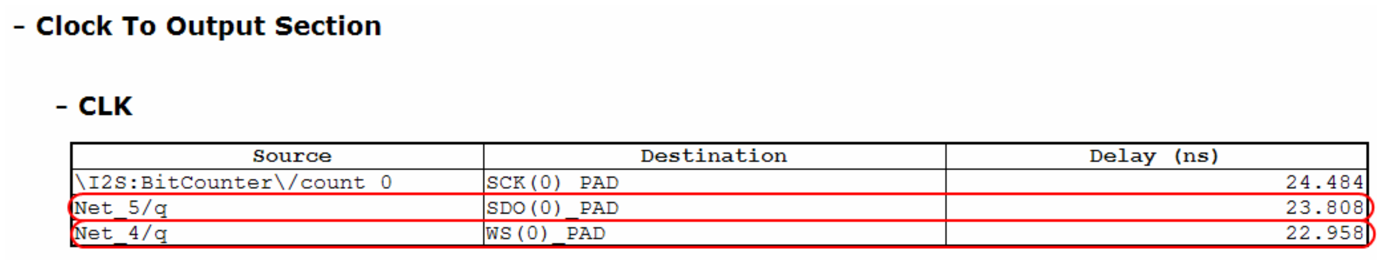


- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count 0	SCK(0) PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{WS_SDO} The delay between WS edge and SDO valid. The value can be determined by opening clock to output times section of STA report, as shown below:

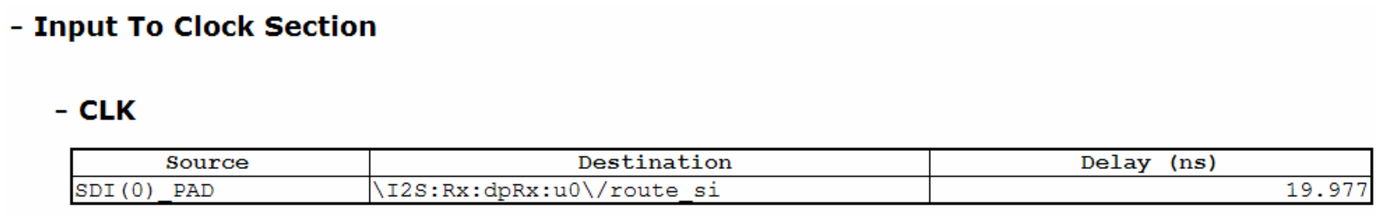


- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count 0	SCK(0) PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{S_SDI} SDI setup time is the SDI pin to internal logic path delay of the master component. This is provided in the STA results Input to Clock Section as shown below:



- Input To Clock Section

- CLK

Source	Destination	Delay (ns)
SDI(0)_PAD	\I2S:Rx:dpRx:u0\/route_si	19.977

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.20	Added internal FIFO error detection by the component.	Improved error handling by the component in case of an overflow/underflow error occurs.
2.10	Resampled FIFO block status signals to DP clock.	Allows component to function with the same timing results for all PSoC 3 and PSoC 5 silicons.
	Added characterization data to datasheet	
	Minor datasheet edits and updates	

Version	Description of Changes	Reason for Changes / Impact
2.0	Hardware implementation of this component was changed to require a 2X frequency signal on the clock input. The SCK output signal will be generated by dividing the incoming clock by 2.	Improves the control of the timing relationship between the SCK, WS, SDO and SDI signals.
	I2S_Start() function updated to match changes in the implementation. The functionality is unchanged.	Simplified implementation required changes to the initialization.
	The sleep mode APIs were added.	To support low-power modes.
	The status bits tx_not_full and rx_not_emty were changed from clear-on-read to transparent mode.	The status bits for any Full or Empty status from a FIFO need to be transparent to represent just the current live status of the FIFO.
	Added DMA Capabilities file to the component.	This file allows I2S to be supported by the DMA Wizard tool in PSoC Creator.
	I2S_Stop() API was changed to clear rx and tx FIFOs after component is disabled.	Resets the Tx and Rx FIFOs status to the initial values. Prevents unexpected operations after component is re-enabled.
	Added Keil function reentrancy support.	Add the capability for customers to specify individual generated functions as reentrant.