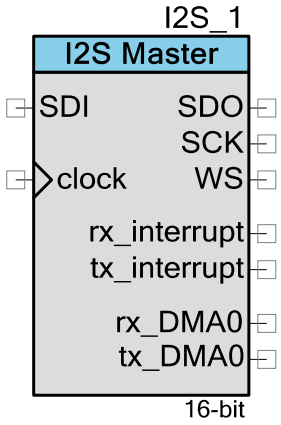
Inter-IC Sound Bus (I2S)
1.0

Features



- Master only
- 8 - 32 data bits per sample
- 16, 32, 48, or 64 bit word select period
- Data rate up to 192KHz with 64 bit word select period: 12.288MHz
- Interrupts:
  - Tx FIFO not full
  - Rx FIFO not empty
  - Tx FIFO underflow
  - Rx FIFO overflow
- 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) 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 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). This specification provides a serial bus interface for stereo audio data. This interface is most commonly used by audio ADC and DAC components.

PRELIMINARY

Input/Output Connections

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

clock

The clock rate provided must be the same clock rate as the desired clock rate for the output serial clock (SCK). For example to produce 48KHz audio with a 64 bit word select period, the clock frequency would be 48KHz * 64 = 3.072MHz.

SDI *

Serial data input. Display only if Rx direction is present.

SCK

Output serial clock.

WS

Word select output indicates the channel being transmitted.

SDO *

Serial data output. Display only if Tx direction is present.

rx_interrupt *

Rx direction interrupt. Display only if Rx direction is present.

tx_interrupt*

Tx direction interrupt. Display only if Tx direction is present.

rx_DMA0 *

Rx direction DMA request for FIFO 0 (Left or Interleaved). Display only if DMA is selected for the Rx direction.

rx_DMA1 *

Rx direction DMA request for FIFO 1 (Right). Display only if DMA is selected for the Rx direction and separate FIFOs are selected.

tx_DMA0 *

Tx direction DMA request for FIFO 0 (Left or Interleaved). Display only if DMA is selected for the Tx direction.

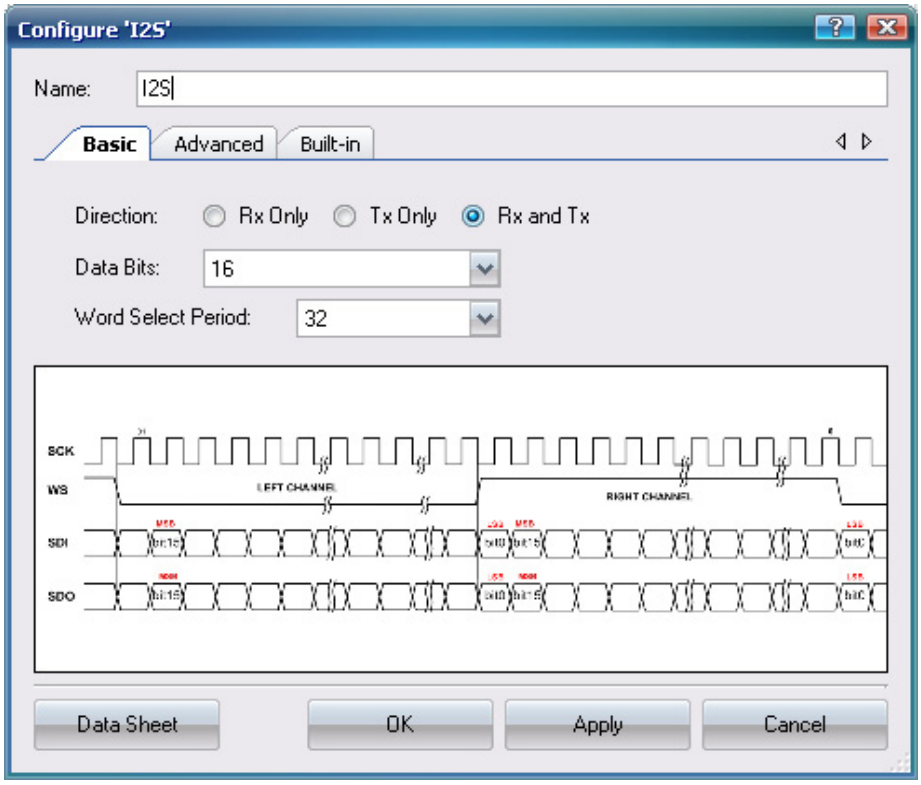
tx_DMA1 *

Tx direction DMA request for FIFO 1 (Right). Display only if DMA is selected for the Tx direction and separate FIFOs are selected.

Parameters and Setup

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

Basic Tab



The I2S dialog contains the following settings:

Direction

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

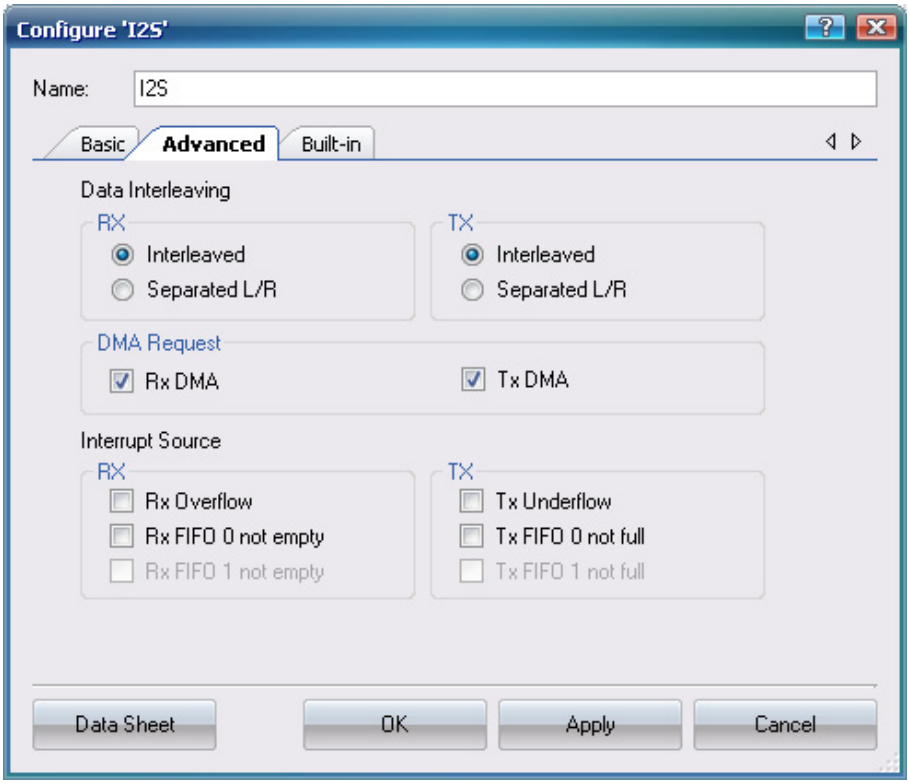
Data Bits

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 (hardware compiled). This value can be set to: 16, 32 (default), 48, or 64.

Advanced Tab



Data Interleaving

Determines whether the data is interleaved. Selectable independently for Rx and Tx (hardware compiled). This value can be set to: Interleaved (default) or Separate L/R.

DMA Source

Sets whether DMA request signals are present for the component. Separately selectable for Rx and Tx directions (hardware compiled). This value can be set to: DMA (default) or No DMA.

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.

Placement

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

Resources

Resolution	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
Rx Direction	1	5	1	1	1	?	?	3
Tx Direction	1	4	1	1	1	?	?	3
Rx and Tx	2	9	2	1	1	?	?	4

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
void I2S_Start(void)	Starts the I2S interface.
void I2S_Stop(void)	Disables the I2S interface.
void I2S_EnableTx(void)	Enables the Tx direction of the I2S interface.
void I2S_DisableTx(void)	Disables the Tx direction of the I2S interface.
void I2S_EnableRx(void)	Enables the Rx direction of the I2S interface.
void I2S_DisableRx(void)	Disables the Rx direction of the I2S interface.
void I2S_SetRxInterruptMode(uint8 interruptSource)	Sets the interrupt source for the I2S Rx direction interrupt.
void I2S_SetTxInterruptMode(uint8 interruptSource)	Sets the interrupt source for the I2S Tx direction interrupt.
uint8 I2S_ReadRxStatus(void)	Returns state in the I2S Rx status register.
uint8 I2S_ReadTxStatus(void)	Returns state in the I2S Tx status register.
uint8 I2S_ReadByte(uint8 wordSelect)	Returns a single byte from the Rx FIFO.
void I2S_WriteByte(uint8 wrData, uint8 wordSelect)	Writes a single byte into the Tx FIFO.
void I2S_ClearRxFIFO(void)	Clears out the Rx FIFO.
void I2S_ClearTxFIFO(void)	Clears out the Tx FIFO.

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. At the next word select falling edge transmission will begin.

Parameters: None

Return Value: None

Side Effects: None

void I2S_DisableTx(void)

Description: Disables the Tx direction of the I2S interface. At the next word select falling edge transmission of data will stop and a constant 0 value will be transmitted.

Parameters: None

Return Value: None

Side Effects: None

void I2S_EnableRx(void)

Description: Enables the Rx direction of the I2S interface. At the next word select falling edge reception of data will begin.

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 reception of data will no longer be 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.
- Rx FIFO 0 not empty (Left or Interleaved)
- Rx FIFO 1 not empty (Right)
- Rx FIFO overflow

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.
- Tx FIFO 0 not full (Left or Interleaved)
- Tx FIFO 1 not full (Right)
- Tx FIFO underflow

Return Value: None

Side Effects: None

uint8 I2S_ReadRxStatus(void)

Description: Returns state in the I2S Rx status register.

Parameters: None

Return Value: (uint8) state of the I2S Rx status register.

I2S Rx Status Masks	Value
RX_FIFO_OVERFLOW	0x01
RX_FIFO_0_NOT_EMPTY	0x02
RX_FIFO_1_NOT_EMPTY	0x04

Side Effects: Clears the I2S Rx status register.

uint8 I2S_ReadTxStatus(void)

Description: Returns state in the I2S Tx status register.

Parameters: None

Return Value: (uint8) state of the I2S Tx status register.

I2S Tx Status Masks	Value
TX_FIFO_UNDERFLOW	0x01
TX_FIFO_0_NOT_FULL	0x02
TX_FIFO_1_NOT_FULL	0x04

Side Effects: Clears the I2S Tx status register.

uint8 I2S_ReadByte(uint8 wordSelect)

Description: Returns a single byte from the Rx FIFO. The Rx status should be checked 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 the 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. The Tx status should be checked 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 the 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 will be lost. This call should be made 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 will be lost. This call should be made only when the Tx direction is disabled.

Parameters: None

Return Value: None

Side Effects: None

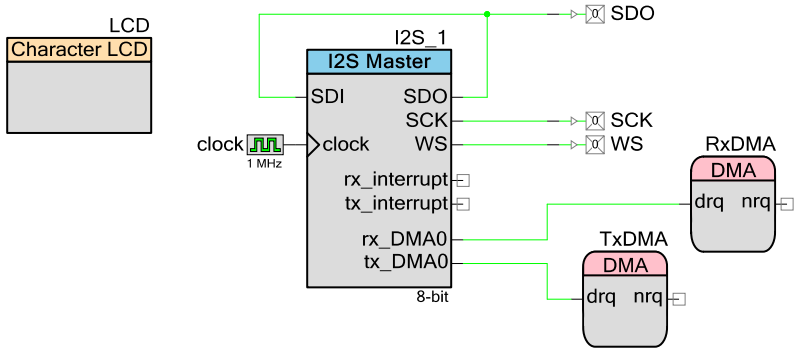
Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the I2S component. This example assumes the component has been placed in a design with the default name "I2S_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

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

As shown in the following schematic, two DMA components named "TxDMA" and "RxDMA" have been placed and connected to the tx_DMA0 and rx_DMA0 terminals of the I2S component. The I2S component has been configured with the DMA Request enabled for both the Tx and Rx directions. Note that the DMA request signals from the I2S component are level (instead of edge) signals, so the example code configures the DMA appropriately. Finally, there is a Character LCD component named "LCD" that is used to display the results.




```
#include <device.h>

void Dma_Rx_Configuration(void);
void Dma_Tx_Configuration(void);

#define TD_SIZE   0x08U

/* DMA channels and Transaction Descriptors */
uint8 RxChannel;
uint8 TxChannel;
uint8 RxTD;
uint8 TxTD;

uint8 RxBuff[8];    /* IN buffer */

/* Data to be transmitted by Tx */
uint8 test_data[8] = {0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77};

void main()
{
    uint8 i;
```

```
    /* Clear IN buffer */
    for(i=0; i<8; i++)
    {
        RxBuff[i] = 0;
    }

    LCD_Start();
    LCD_PrintString("WS=16 DATA=8");

    /* Enable I2S component */
    I2S_1_Start();

    /* Configure DMAs for each direction */
    Dma_Rx_Configuration();
    Dma_Tx_Configuration();

    LCD_Position(1,0);

    /* Through the APIs the Tx and Rx directions will be started independently. */
    /* Depending on exactly when the Rx and Tx get enabled relative to each     */
    /* other the bytes in RxBuff will all be present and in order, but they may */
    /* start with any of the values (cyclic shifted testData).                  */
    I2S_1_EnableRx();   /* Enable Rx direction */
    I2S_1_EnableTx();   /* Enable Tx direction */

    CyDelay(200);
    for(i=0; i<8; i++)
    {
        LCD_PrintHexUint8(RxBuff[i]);
    }

    for(;;);
}

void Dma_Rx_Configuration()
{
    uint32 SrcAddr, DstAddr;
    uint8 val;

    /* Set the Rx DMA request to be level sensitive          */
    /* By default the DMA requests are edge sensitive        */
    /* but these are being driven by the FIFO level signals  */
    val = CY_GET_XTND_REG8(RxDMA__DRQ_CTL);
    val &= ~(3 << ((RxDMA__DRQ_NUMBER & 0x3) * 2));
    val |= (1 << ((RxDMA__DRQ_NUMBER & 0x3) * 2));
    CY_SET_XTND_REG8(RxDMA__DRQ_CTL, val);

    SrcAddr = I2S_1_Rx_dpRx_u0__F0_REG;
    DstAddr = ((uint32) RxBuff);
#if (defined(__C51__))
    /* In the case of the PSoC 3 8051 processor the upper bytes of an SRAM  */
    /* pointer is a type field that does not apply for DMA operations.      */
    DstAddr &= 0xFFFF;
#endif
```

```
    /* Rx DMA Config */
    /* Init DMA, 1 byte bursts, each burst requires a request */
    RxChannel = RxDMA_DmaInitialize(1, 1, HI16(SrcAddr), HI16(DstAddr));
    RxTD = CyDmaTdAllocate();

    /* Configure this Td as follows: */
    /*  - The TD is looping on itself */
    /*  - Increment the destination address, but not the source address */
    CyDmaTdSetConfiguration(RxTD,
                            TD_SIZE,
                            RxTD,
                            TD_INC_DST_ADR);

    /* From the FIFO to the memory */
    CyDmaTdSetAddress(RxTD, LO16(SrcAddr), LO16(DstAddr));

    /* Associate the TD with the channel */
    CyDmaChSetInitialTd(RxChannel, RxTD);

    /* Enable the channel */
    CyDmaChEnable(RxChannel, 1);
}

void Dma_Tx_Configuration()
{
    uint32 SrcAddr, DstAddr;
    uint8 val;

    /* Set the Tx DMA request to be level sensitive         */
    /* By default the DMA requests are edge sensitive       */
    /* but these are being driven by the FIFO level signals */
    val = CY_GET_XTND_REG8(TxDMA__DRQ_CTL);
    val &= ~(3 << ((TxDMA__DRQ_NUMBER & 0x3) * 2));
    val |= (1 << ((TxDMA__DRQ_NUMBER & 0x3) * 2));
    CY_SET_XTND_REG8(TxDMA__DRQ_CTL, val);

    SrcAddr = (uint32) test_data;
    DstAddr = I2S_1_Tx_dpTx_u0__F0_REG;
#if (defined(__C51__))
    /* In the case of the PSoC 3 8051 processor the upper bytes of an SRAM  */
    /* pointer is a type field that does not apply for DMA operations.      */
    SrcAddr &= 0xFFFF;
#endif

    /* Tx DMA Config */
    /* Init DMA, 1 byte bursts, each burst requires a request */
    TxChannel = TxDMA_DmaInitialize(1, 1, HI16(SrcAddr), HI16(DstAddr));
    TxTD = CyDmaTdAllocate();

    /* Configure this Td chain as follows: */
    /*  - The TD is looping on itself */
    /*  - Increment the source address, but not the destination address */
    CyDmaTdSetConfiguration(TxTD,
                            TD_SIZE,
                            TxTD,
```

```
                                TD_INC_SRC_ADR);

    /* From the memory to the FIFO */
    CyDmaTdSetAddress(TxTD, LO16(SrcAddr), LO16(DstAddr));

    /* Associate the TD with the channel */
    CyDmaChSetInitialTd(TxChannel, TxTD);

    /* Enable the channel */
    CyDmaChEnable(TxChannel, 1);
}
```

Functional Description

Left/Right and Rx/Tx configuration

The configuration for the Left and Right channels and for the Rx and Tx direction number of bits and word select period are identical. If it is necessary for the application to have different configurations for the Rx and Tx, then two unidirectional component instances should be used.

Data stream format

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. Any unused bits will be ignored on Tx and will be 0 on Rx.

The data stream for one direction can either 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.

Enabling

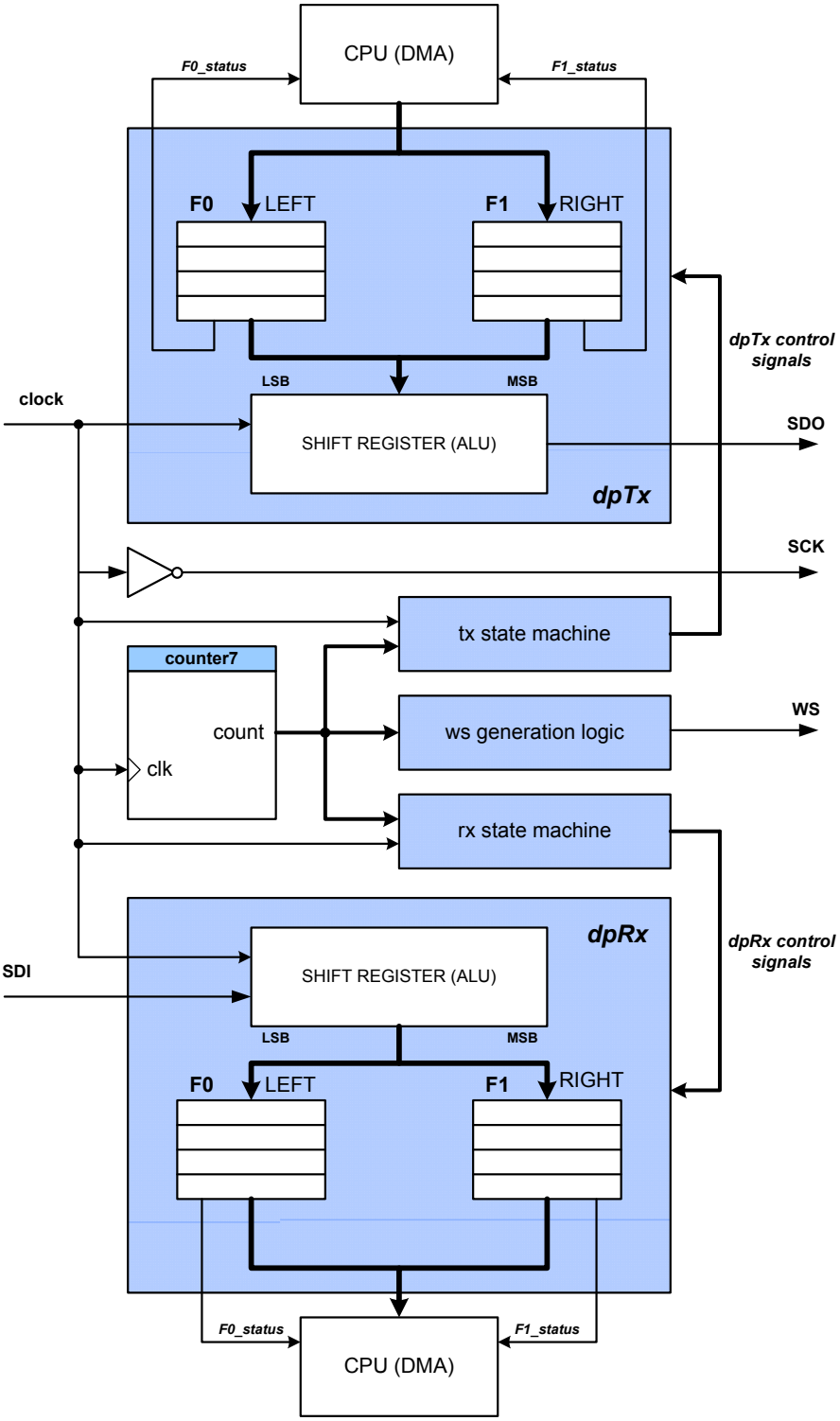
The Rx and Tx directions have separate enables. When not enabled, the Tx direction will transmit all 0 values, and the Rx direction will ignore all received data. The transition into and out of the enabled state will occur at a word select boundary such that a left / right sample pair is always transmitted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

DMA

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

Block Diagram and Configuration

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



PRELIMINARY

Registers

I2S_ControlReg

Bits	7	6	5	4	3	2	1	0
Value	reserved			CntEn	reset	CntLd	rxenable	txenable

- CntEn: enable / disable I2S bit counter
- reset: initial reset of all internal logic
- CntLd: initial load internal bit counter period
- rxenable, txenable: enable / disable rx and tx directions respectively

I2S_tx_sts_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 not full event has occurred
- F0_not_full: if set tx FIFO 0 not full event has occurred
- underflow: if set tx FIFOs underflow event has occurred

The register value may be read by the user with the I2S_ReadTxStatus() API function.

I2S_rx_sts_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 not empty event has occurred
- F0_not_empty: if set rx FIFO 0 not empty event has occurred
- overflow: if set rx FIFOs overflow event has occurred

The register value may be read by the user with the I2S_ReadRxStatus() API function.

References

Not applicable

DC and AC Electrical Characteristics

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Input Capacitance	---		---	pF	
Input Impedance	---		---	Ω	
Maximum Clock Rate	---		67	MHz	

Component Changes

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

Version	Description of Changes
1.0.a	Updated internal visibility expression; no affect on component behavior.
1.0	Initial release.