



I2S_W Audio Streaming Controller

Summary

This document provides detailed reference information with respect to the I2S_W peripheral device.

Core Reference
CR0160 (v2.0) March 10, 2008

Altium Designer's I2S_W Audio Streaming Controller is used to facilitate data transfers over the inter-IC sound (I^2S) bus. The I^2S bus – developed by Philips as a dedicated serial link for digital audio – allows a standardized communication medium for an ever-increasing array of digital audio devices.

The standard I^2S bus consists of three lines:

- a data line, consisting of two time-division-multiplexed channels (Left and Right)
- a word select line
- a clock line.

The I2S_W acts as a Master on the bus and, as such, generates the continuous serial bit clock and the word select signal for the bus.

For further information on the I^2S bus, refer to the [I2S bus specification](#).

The I2S_W Controller uses two data lines – one to transmit data and one to receive data.

Features

- I^2S -compatible interface
- Operates as I^2S Master
- Supports four data word widths:
 - 16, 20, 24 and 32 bit word widths
- 32 x 24-bit word Transmitter and Receiver FIFOs
- Independently selectable mono/stereo for Receiver and Transmitter
- Wishbone-compliant
- Wishbone data to Transmitter right-aligned in 16, 20 and 24 bit word-width modes (bit 0 is the LSB)
- Wishbone data to Transmitter left-aligned in 32 bit word-width mode (bit 31 is the MSB)

Available Devices

The I2S_W Controller device can be found in the FPGA Peripherals integrated library (FPGA_Peripherals.IntLib), located in the `\Library\Fpga` folder of the installation.

I2S_W Audio Streaming Controller

Functional Description

Symbol

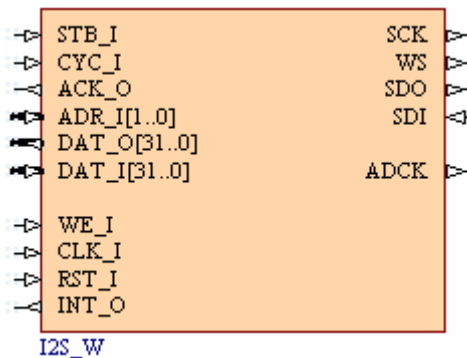


Figure 1. I2S_W Controller Symbol.

Pin Description

Table 1. I2S_W pin description

Name	Type	Polarity/ Bus size	Description
Control Signals			
CLK_I	I	Rise	External (system) clock signal
RST_I	I	High	External (system) reset
Host Processor Interface Signals			
STB_I	I	High	Strobe signal. When asserted, indicates the start of a valid Wishbone data transfer cycle
CYC_I	I	High	Cycle signal. When asserted, indicates the start of a valid Wishbone cycle
ACK_O	O	High	Standard Wishbone device acknowledgement signal. When this signal goes high, the Controller (Wishbone Slave) has finished execution of the requested action and the current bus cycle is terminated
ADR_I	I	2	Address bus, used to select an internal register of the device for writing to/reading from
DAT_O	O	32	Data to be sent to host processor
DAT_I	I	32	Data received from host processor
WE_I	I	Level	Write enable signal. Used to indicate whether the current local bus cycle is a Read or Write cycle: 0 = Read 1 = Write
INT_O	O	High	Interrupt output line. An interrupt can be generated based on the comparison of the level of data in the Receive or Transmit FIFOs, with the value defined for the watermark, loaded into the Control register. For more information, see Interrupt generation .

I²S Bus Interface Signals			
SCK	O	Rise	Continuous serial clock
WS	O	Level	Word Select. 0 = Channel 1 (Left) 1 = Channel 2 (Right) This signal not only determines the data channel, but also is used to effectively delimit transmitted data words. WS always changes state one clock cycle (SCK) before the MSB of the next data word is transmitted. The frequency of this signal is equal to the sample rate. In one cycle of WS, one sample (left + right channel) can be transmitted.
SDO	O	1	Serial Data Out
SDI	I	1	Serial Data In
ADCK	O	Rise	Additional Serial Clock. The frequency of this signal is calculated as $256 * f_{WS}$, where f_{WS} is the frequency of the Word Select line and is equal to the sample rate.

I2S_W Audio Streaming Controller

Hardware Description

Block Diagram

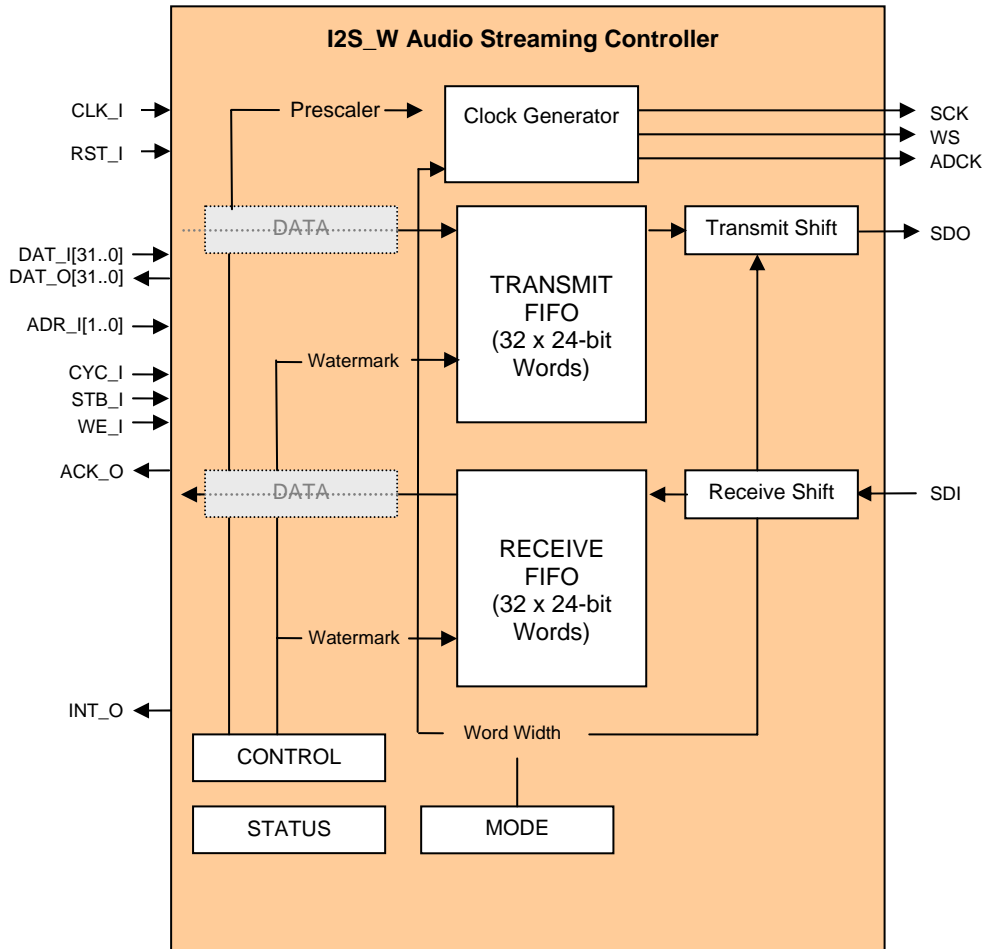


Figure 2. I2S_W Controller block diagram.

Internal Registers

The following sections detail the internal registers for the I2S_W Controller that can be accessed from the host processor.

Control Register (CONTROL)

Address: 0h

Access: Read and Write

Value after Reset: 0000_0000h

This register is used to store the 8-bit `prescaler` and 8-bit `watermark` values for the Controller.

Table 2. The CONTROL register

MSB				LSB			
31	16	15	8	7	0		
-			watermark		prescaler		

Table 3. The MODE register bit functions

Bit	Symbol	Function
CONTROL.31..CONTROL.16	-	Not Used.
CONTROL.15..CONTROL.8	watermark	8-bit watermark value, effectively used to set the interrupt generation point for both the Transmitter and Receiver.
CONTROL.7..CONTROL.0	prescaler	8-bit prescaler value, used to divide the CLK_I signal frequency to 256*sample frequency.

Mode Register (MODE)

Address: 1h

Access: Read and Write

Value after Reset: 0000_0000h

This register is used to set the operational mode of the Controller and to enable interrupts for the Transmitter and Receiver.

Table 4. The MODE register

MSB											LSB		
31	15	14	13	12	11	10	9	8	7	6	5	0	
-			rxie	txie	rxen	txen	slr	mi	mo	-	-	wwidth	

Table 5. The MODE register bit functions

Bit	Symbol	Function
MODE.31..MODE.15	-	Not Used.
MODE.14	rxie	Receiver Interrupt Enable. 0 = Disabled 1 = Enabled
MODE.13	txie	Transmitter Interrupt Enable. 0 = Disabled 1 = Enabled
MODE.12	rxen	Receiver Enable. 0 = Disabled 1 = Enabled
MODE.11	txen	Transmitter Enable. 0 = Disabled 1 = Enabled
MODE.10	slr	Select Left/Right for Mono In. 0 = Left 1 = Right This bit is ignored if the mi bit (MODE.9) = 0.
MODE.9	mi	Mono In. 0 = Stereo 1 = Mono
MODE.8	mo	Mono Out. 0 = Stereo 1 = Mono

I2S_W Audio Streaming Controller

Bit	Symbol	Function
MODE.7..MODE.6	-	Not Used
MODE.5..MODE.0	wwidth	Word Width. Valid values are: 16 (010000) 20 (010100) 24 (011000) 32 (100000)

Status Register (STATUS)

Address: 2h

Access: Read only

Value after Reset: 0000_0032h

This register is used to determine the current state of the Controller.

Table 6. The STATUS register

MSB								LSB	
31	8	7	6	5	4	3	2	1	0
-		rxw	rxf	rxl	rxr	txw	txf	txe	txlr

Table 7. The STATUS register bit functions

Bit	Symbol	Function
STATUS.31..STATUS.8	-	Not Used.
STATUS.7	rxw	Receive FIFO Watermark state. 0 = Watermark not reached 1 = Watermark reached
STATUS.6	rxf	Receive FIFO Full state 0 = Not Full 1 = Full
STATUS.5	rxl	Receive FIFO Empty state. 0 = Not Empty 1 = Empty
STATUS.4	rxr	Receive FIFO Left/Right state. 0 = Next data is from Left channel 1 = Next data is from Right channel
STATUS.3	txw	Transmit FIFO Watermark state. 0 = Watermark not reached 1 = Watermark reached
STATUS.2	txf	Transmit FIFO Full state 0 = Not Full 1 = Full
STATUS.1	txe	Transmit FIFO Empty state. 0 = Not Empty 1 = Empty

Bit	Symbol	Function
STATUS.0	txlr	Transmit FIFO Left/Right state. 0 = Next data is for Left channel 1 = Next data is for Right channel


Data Register (DATA)

Address: 3h

Access: Read and Write

This is not actually a register in the true sense of the word, but rather is a single address that is used to access the Transmit and Receive Buffers. Performing a Wishbone Write to the DATA address loads data directly into the Transmit Buffer.

Performing a Wishbone Read from the DATA address retrieves data directly from the Receive Buffer. If no data words are available in the Receive Buffer, the returned data is invalid. Otherwise, the retrieved data word is removed from the buffer, effectively freeing up space.

 Provided you are performing a Write (WE_I input High), you will access the Transmit Buffer. When performing a Read (WE_I input Low) you will access the Receive Buffer.

Interrupt Generation

The I2S_W Controller has a single interrupt line (INT_O) that is shared by both the Transmitter and Receiver sections. Interrupt generation is essentially determined by the value defined for *watermark*, in the Control register (CONTROL15..8).

The INT_O signal is made active (taken High) under the following two circumstances:

- When there are less data words in the Transmit Buffer than the *watermark* value, the *txw* status flag (STATUS.3) will be set. An interrupt will only be generated provided the corresponding interrupt enable bit – *txie* – is also set in the Mode register (MODE.13).
- If the number of data words in the Receive Buffer exceeds the *watermark* value, the *rxw* status flag (STATUS.7) will be set. An interrupt will only be generated provided the corresponding interrupt enable bit – *rxie* – is also set in the Mode register (MODE.14).

Transmitter and Receiver Reset

The Transmitter section of the Controller can be reset in the following ways:

- Upon a global reset (RST_I = '1')
- If the Transmitter Enable bit – *txen* – in the Mode register (MODE.11) is cleared.

After a reset, the valid data in the Transmit FIFO will be cleared, the Transmit Shift register will be cleared (000000h) and the output on the SDO line will be '0'.

The Receiver section of the Controller can be reset in the following ways:

- Upon a global reset (RST_I = '1')
- If the Receiver Enable bit – *rxen* – in the Mode register (MODE.12) is cleared.

After a reset, the valid data in the Receive FIFO will be cleared and the Receive Shift register will also be cleared (000000h).

Word Width

The word width can be configured, using bits 5..0 of the MODE register, to change the data transfer for the Controller between the following modes:

16 Bits (Right-Aligned)

For transmission of data, the host processor writes a 32-bit value appearing on the DAT_I bus. The upper 8-bits are ignored, with the remaining 24-bit data word (DAT_I(23..0)) loaded into the head of the Transmit Buffer. The value sent out on the I²S bus will be:

TX_FIFO(Tail)(15..0).

For reception of data, a 24-bit value will be loaded into the Receive Buffer. Bits 15..0 of this word will be the value received from the I²S bus. The 32-bit value sent back to the host processor on the DAT_O bus will be:

"SSSSSSSSSSSSSSSS" & RX_FIFO(Tail)(15..0),

where S represents the sign bit – RX_FIFO(Tail)(15).

I2S_W Audio Streaming Controller

20 Bits (Right-Aligned)

For transmission of data, the host processor writes a 32-bit value appearing on the DAT_I bus. The upper 8-bits are ignored, with the remaining 24-bit data word (DAT_I(23..0)) loaded into the head of the Transmit Buffer. The value sent out on the I²S bus will be:

TX_FIFO(Tail)(19..0).

For reception of data, a 24-bit value will be loaded into the Receive Buffer. Bits 19..0 of this word will be the value received from the I²S bus. The 32-bit value sent back to the host processor on the DAT_O bus will be:

"SSSSSSSSSSSS" & RX_FIFO(Tail)(19..0),

where S represents the sign bit – RX_FIFO(Tail)(19).

When using the Controller in 16, 20 or 24 bit modes, the value read by the host processor is sign-extended, allowing negative values to remain negative when read as 32-bit values.

24 Bits (Right-Aligned)

For transmission of data, the host processor writes a 32-bit value appearing on the DAT_I bus. The upper 8-bits are ignored, with the remaining 24-bit data word (DAT_I(23..0)) loaded into the head of the Transmit Buffer. The value sent out on the I²S bus will be:

TX_FIFO(Tail).

For reception of data, the 24-bit value received from the I²S bus will be loaded into the Receive Buffer. The 32-bit value sent back to the host processor on the DAT_O bus will be:

"SSSSSSSS" & RX_FIFO(Tail),

where S represents the sign bit – RX_FIFO(Tail)(23).

24 Bits (Left-Aligned).

The last mode is actually termed 32 bit mode. The lower 8 bits are all set to '0'.

For transmission of data, the host processor writes a 32-bit value appearing on the DAT_I bus. The lower 8-bits are ignored, with the remaining 24-bit data word (DAT_I(31..8)) loaded into the head of the Transmit Buffer. The value sent out on the I²S bus will be:

TX_FIFO(Tail)

For reception of data, the 24-bit value received from the I²S bus will be loaded into the Receive Buffer. The value sent back to the host processor on the DAT_O bus will be:

RX_FIFO(Tail) & "00000000".

Clock Prescaler

The `prescaler` value loaded into the Control register (CONTROL7..0) is used to set the Sample Rate as follows:

$$\text{Sample Rate} = \frac{\text{CLK_I}}{\text{prescaler} * 256}$$

The frequency of the WS signal is equal to the Sample Rate. In one cycle of WS, one sample (left + right channel) can be transmitted.

The serial bit clock (SCK) is always at least 16, 20, or 24 times higher in frequency than the WS signal – depending on whether the configured Word Width is 16, 20, or 24/32 bits respectively. In reality, SCK is 16, 21, or 25 times the frequency of WS, due to the possible divider values in the core. These divider values also mean that SCK does not have a regular 50% duty-cycle.

The additional serial clock (ADCK) is always 256 times the frequency of WS, irrespective of the Word Width configured for the Controller. ADCK is simply an additional 'general purpose' clock signal. Some audio devices require such a signal – for example to clock digital filters in the ADC/DAC stages. Note that if the resulting frequency of ADCK is not the same as the frequency of clock required by the audio device, you will need to source your own clock signal from another part of your circuit.

Interfacing to a 32-bit Processor

Figure 3 shows an example of how an I2S_W device can be wired into a design that uses a 32-bit processor – in this case a TSK3000A. A configurable Wishbone Interconnect device (WB_INTERCON) is used to simplify connection and also handle the word addressing – taking the 24-bit address line from the processor and mapping it to the 2-bit address line used to drive the I2S_W.

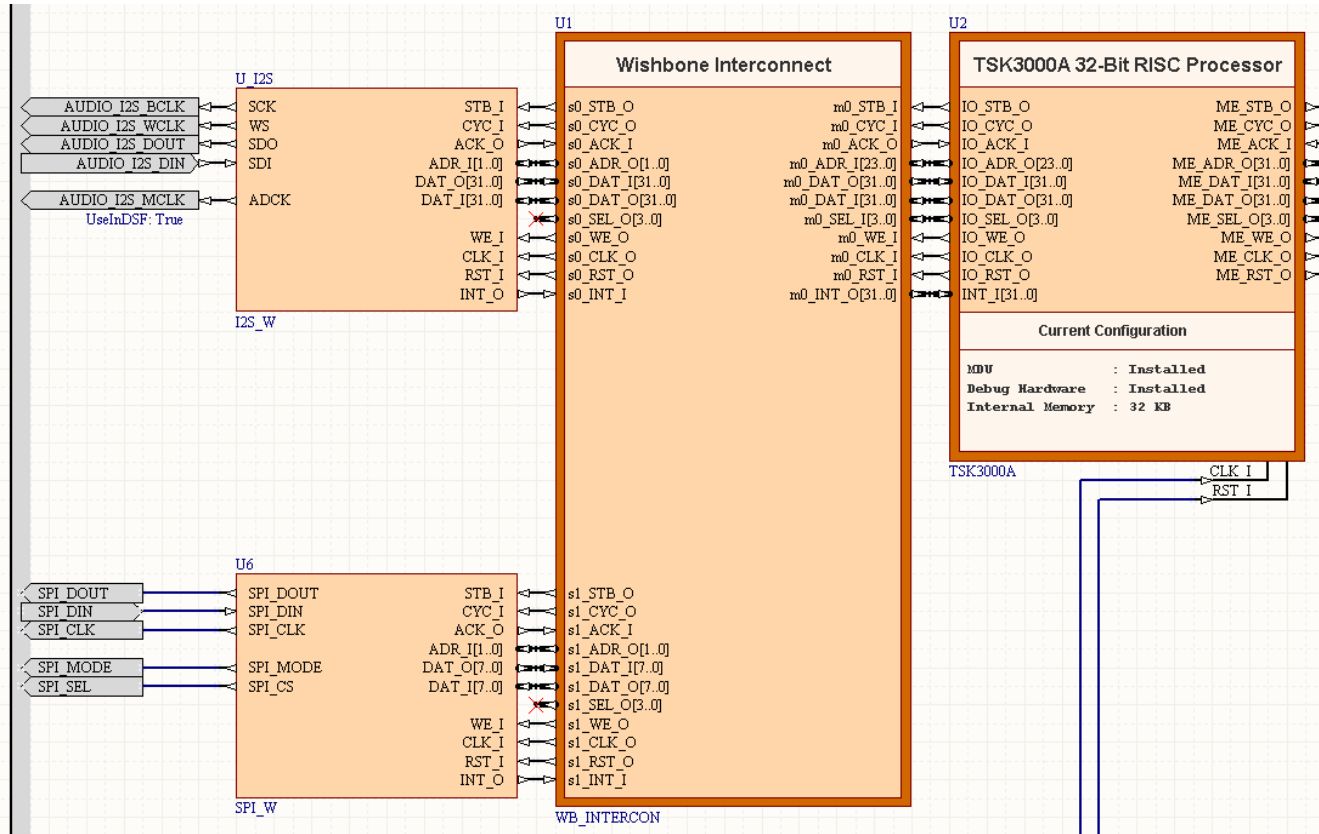


Figure 3. Example interfacing between a 32-bit processor (TSK3000A) and an I2S_W Controller.

When configuring the WB_INTERCON device – in particular the I2S_W slave interface – ensure that the Address Bus Mode is set to Word Addressing – $ADR_O(0) \leq ADR_I(1 \text{ or } 2)$. As the I2S_W's data bus width is 32-bit, the two lowest address bits are not connected to the slave device. $ADR_I(2)$ of the master is mapped to $ADR_O(0)$ of the slave, providing sequential word addresses (or addresses at every 4 bytes). Bits 3..2 of the output address line from the host processor (IO_ADR_O) are therefore mapped, through the WB_INTERCON, to bits 1..0 of the I2S_W's input address line (ADR_I).

The actual 24-bit address sent out from the processor on its IO_ADR_O line is therefore constructed as follows:

$$I2S_W \text{ Base Address} + (\text{Internal Register Address} \& \text{"00"})$$

The Base Address for the I2S_W Controller is specified as part of the peripheral's definition when adding it as a slave to the Wishbone Interconnect. For example, if the base address entered for the device is 100000h (mapping it to address FF10_0000h in the processor's address space), and you want to write to the Mode register (MODE) with address 1h, the value entered on the processor's IO_ADR_O line would be:

$$100000h + 4h = 100004h$$

For further information on the Wishbone Interconnect, refer to the [WB_INTERCON Configurable Wishbone Interconnect](#) core reference.

For further information on the TSK3000A processor, refer to the [TSK3000A 32-bit RISC Processor](#) core reference. Similar references can be found for other 32-bit processors supported by Altium Designer, by using the lower section of the **Knowledge Center** panel and navigating to the *Documentation Library » Embedded Processors and Software Development » FPGA Based and Discrete Processors* section.

The following example project includes an I2S_W Controller device: `\Examples\NB2DSK1 Examples\DSF I2S Audio\DSF_I2S_Audio.PrjFpg`.

I2S_W Audio Streaming Controller


Host to Controller Communications

Communications between a 32-bit host processor and the I2S_W Controller are carried out over a standard Wishbone bus interface. The following sections detail the communication cycles involved between Host and Controller for writing to/reading from the internal registers.

Writing to an Internal Register

Data is written from the host processor to an internal register in the I2S_W Controller, in accordance with the standard Wishbone data transfer handshaking protocol. The write operation occurs on the rising edge of the CLK_I signal and can be summarized as follows:

- The host presents the required 24-bit address based on the register to be written on its IO_ADR_O output and valid data on its IO_DAT_O output. It then asserts its IO_WE_O signal, to specify a write cycle
- The I2S_W receives the 2-bit address on its ADR_I input and, identifying the addressed register, prepares to receive data into that register
- The host asserts its IO_STB_O and IO_CYC_O outputs, indicating that the transfer is to begin. The I2S_W, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by latching the data appearing at its DAT_I input into the target register and asserting its ACK_O signal – to indicate to the host that the data has been received
- The host, which monitors its IO_ACK_I input on each rising edge of the CLK_I signal, responds by negating the IO_STB_O and IO_CYC_O signals. At the same time, the I2S_W negates the ACK_O signal and the data transfer cycle is naturally terminated.

 Remember that when writing to the Data register, you are actually loading a value directly into the Transmit Buffer. The value loaded will depend on the configured Word Width for the Controller, as specified in the Mode register (MODE5..0):

- 16, 20, 24 bit – DAT_I(23..0)
- 32 bit – DAT_I(31..8).

Reading from an Internal Register


Data is read from an internal register in accordance with the standard Wishbone data transfer handshaking protocol. The read operation, which occurs on the rising edge of the CLK_I signal, can be summarized as follows:

- The host presents the required 24-bit address based on the register to be read on its IO_ADR_O output. It then negates its IO_WE_O signal, to specify a read cycle
- The I2S_W receives the 2-bit address on its ADR_I input and, identifying the addressed register, prepares to transmit data from the selected register
- The host asserts its IO_STB_O and IO_CYC_O outputs, indicating that the transfer is to begin. The I2S_W, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by presenting the valid data on its DAT_O output and asserting its ACK_O signal – to indicate to the host that valid data is present
- The host, which monitors its IO_ACK_I input on each rising edge of the CLK_I signal, responds by latching the data appearing at its IO_DAT_I input and negating the IO_STB_O and IO_CYC_O signals. At the same time, the I2S_W negates the ACK_O signal and the data transfer cycle is naturally terminated.

Table 8 summarizes the 'make-up' of the 32-bit data word that is read back from each register.

Table 8. Values read from internal registers during a read.

Internal Register	Value presented to host processor
CONTROL	"0000000000000000" & CONTROL(15..0)
MODE	"0000000000000000" & MODE(14..8) & "00" & MODE(5..0)
STATUS	"000000000000000000000000" & STATUS(7..0)

 Remember that when reading from the Data register, you are actually retrieving a value directly from the Receive Buffer. The value presented to the host processor depends on the configured Word Width – see the section [Word Width](#).

Operational Overview

Initialization

After an external reset, you will need to initialize the I2S_W Controller. This should be carried out in accordance with design requirements and can include:

- Loading the required values for the `watermark` and `prescaler` to the Control register.
- Setting the desired word width in the Mode register.
- Setting the Controller to operate in Mono using bits 10..8 in the Mode register. By default, the Controller will be reset to operate in Stereo mode.
- Enabling Transmitter and/or Receiver interrupts in the Mode register (MODE.13 and MODE.14 respectively).
- Enabling the Transmitter and Receiver by setting bits 11 and 12 in the Mode register respectively. This step should be carried out last in order that the Controller be fully configured to operate as required before data is transmitted and/or received.

Data Transmission

Provided the Transmitter is enabled, the I2S_W will start sending data as soon as it is available in the Transmit Buffer and the clock signals (SCK, WS) are ready.

The next data word in the Transmit Buffer is loaded into the Transmit Shift register. Depending on the configured word width, the 16, 20 or 24 bits of data are then shifted out onto the SDO line on each rising edge of SCK – MSB first.

Until there is data in the Transmit Buffer, or if the Transmitter is disabled, the SDO line will remain '0'.

If, when transmitting in Mono (no bit in Mode register is High), the Transmit Buffer becomes empty, the last sample is repeated until new data is available. This ensures that the target device avoids receiving a "click".

If transmitting in Stereo (no bit Low) and the Transmit Buffer becomes empty, the "click" is avoided by repeating transmission of the last two samples – Left and Right channels – until new data is available.

Why MSB First?

Serial data is transmitted in two's complement, with the most significant bit sent first. The reason for sending the data MSB first is because the target Receiver may be configured with a different Word Width to the I2S_W's Transmitter. Sending the MSB first solves the following two scenarios:

- When the target Receiver's Word Width is greater than that of the I2S_W, the missing bits are simply set to zero, internally, by the Receiver.
- When the target Receiver's Word Width is less than that of the I2S_W, the bits after the LSB are simply ignored.

Data Reception

Provided the Receiver is enabled, the I2S_W will start receiving data as soon as the connected remote device sends it.

The 16, 20 or 24 bits of data sent from the remote device's transmitter are read into the Receive Shift register on each rising edge of SCK. Once all bits in a data word are fully received, it is loaded into the Receive Buffer.

If the Receiver is disabled, incoming data is ignored.

Importance of WS

The WS line not only indicates the channel (Left or Right) being transmitted but is also used to provide effective delimiting between data words. The Transmitter always sends the MSB of the next data word one clock period (of SCK) after WS changes, allowing time for the remote device to:

- Derive synchronize timing for transmission of serial data from its Transmitter section
- Allow the remote device's Receiver to distinguish between the end of one data word and the start of the next.

As the WS signal is generated by the I2S_W (as Bus Master), the required synchronized timing for the Controller's Receiver section is also derived using this signal, so that the data from the remote device's Transmitter can be received correctly.

One other point to note is that the data channels – Left and Right – are multiplexed, with WS being the selection control. Data is therefore sent in the following fashion:

Right Channel (WS=1) > Left Channel (WS=0) > Right Channel (WS=1) > ...


I2S_W Audio Streaming Controller

Using the I2S_W with a TSC2301 Device

The following notes relate to using the I2S_W Controller to facilitate communications between a 32-bit processor and the Texas Instruments TSC2301 device – a programmable touch screen controller with stereo audio CODEC:

- Transmission is not possible when the I2S_W is configured to operate with a Word Width of 16 bits.
- The ADCK signal can be used to provide the clock signal required by the TSC2301's MCLK input.

More information on the TSC2301 can be obtained from the [Texas Instruments](#) website.

 As a general note, there are some audio devices in the market that do not fully implement the I²S Bus specification. Care should be taken to configure the I2S_W to only use modes supported by such a device.

Revision History

Date	Version No.	Revision
06-Oct-2006	1.0	Initial release
10-Mar-2008	2.0	Updated for Altium Designer Summer 08

Software, hardware, documentation and related materials:

Copyright © 2008 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, Altium Designer, Board Insight, Design Explorer, DXP, LiveDesign, NanoBoard, NanoTalk, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.