TMS320VC5505/5504 DSP Inter-IC Sound (I2S) Bus

User's Guide

Literature Number: SPRUFP4
September 2009–Revised January 2010

List of Figures

List of Tables

Read This First

About This Manual

This document describes the features and operation of Inter-IC Sound (I2S) Bus for the TMS320VC5505/5504 Digital Signal Processor (DSP). This structure assumes that a support model where only specific use cases are supported. Although the general architecture is described and the register functions are included, only the functions supported by TI are documented and supported.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the TMS320VC5505/5504 Digital Signal Processor (DSP). Copies of these documents are available on the internet at http://www.ti.com.

SWPU073 — TMS320C55x 3.0 CPU Reference Guide. This manual describes the architecture, registers, and operation of the fixed-point TMS320C55x digital signal processor (DSP) CPU.

SPRU652 — TMS320C55x DSP CPU Programmer's Reference Supplement. This document describes functional exceptions to the CPU behavior.

SPRUFO0 — TMS320VC5505/5504 Digital Signal Processor (DSP) Universal Serial Bus 2.0 (USB) User's Guide. This document describes the universal serial bus 2.0 (USB) in the TMS320VC5505/5504 Digital Signal Processor (DSP). The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices.

SPRUFO1 — TMS320VC5505/5504 Digital Signal Processor (DSP) Inter-Integrated Circuit (I2C) Peripheral User's Guide. This document describes the inter-integrated circuit (I2C) peripheral in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The I2C peripheral provides an interface between the device and other devices compliant with Phillips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1 and connected by way of an I2C-bus. This document assumes the reader is familiar with the I2C-bus specification.

SPRUFO2 — TMS320VC5505/5504 Digital Signal Processor (DSP) Timer/Watchdog Timer User's Guide. This document provides an overview of the three 32-bit timers in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The 32-bit timers of the device are software programmable timers that can be configured as general-purpose (GP) timers. Timer 2 can be configured as a GP, a Watchdog (WD), or both simultaneously.

SPRUFO3 — TMS320VC5505/5504 Digital Signal Processor (DSP) Serial Peripheral Interface (SPI) User's Guide. This document describes the serial peripheral interface (SPI) in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 32 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI supports multi-chip operation of up to four SPI slave devices. The SPI can operate as a master device only.

SPRUFO4 — TMS320VC5505/5504 Digital Signal Processor (DSP) General-Purpose Input/Output (GPIO) User's Guide. This document describes the general-purpose input/output (GPIO) on the TMS320VC5505/5504 digital signal processor (DSP). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of an internal register. When configured as an output you can write to an internal register to control the state driven on the output pin.

SPRUFO5 — TMS320VC5505/5504 Digital Signal Processor (DSP) Universal Asynchronous Receiver/Transmitter (UART) User's Guide. This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The UART performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU.

SPRUF07 — TMS320VC5505/5504 Digital Signal Processor (DSP) Real-Time Clock (RTC) User's Guide. This document describes the operation of the Real-Time Clock (RTC) module in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The RTC also has the capability to wake-up the power management and apply power to the rest of the device through an alarm, periodic interrupt, or external WAKEUP signal.

SPRUFO8 — TMS320VC5505/5504 Digital Signal Processor (DSP) External Memory Interface (EMIF) User's Guide. This document describes the operation of the external memory interface (EMIF) in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The purpose of the EMIF is to provide a means to connect to a variety of external devices.

SPRUFO9 — TMS320VC5505/5504 Digital Signal Processor (DSP) Direct Memory Access (DMA) Controller User's Guide. This document describes the features and operation of the DMA controller that is available on the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The DMA controller is used to move data among internal memory, external memory, and peripherals without intervention from the CPU and in the background of CPU operation.

SPRUFP0 — TMS320VC5505 Digital Signal Processor (DSP) System User's Guide. This document describes various aspects of the TMS320VC5505/5504 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUGL6 — TMS320VC5504 Digital Signal Processor (DSP) System User's Guide. This document describes various aspects of the TMS320VC5505/5504 digital signal processor (DSP) including: system memory, device clocking options and operation of the DSP clock generator, power management features, interrupts, and system control.

SPRUFP1 — TMS320VC5505 Digital Signal Processor (DSP) Successive Approximation (SAR) Analog to Digital Converter (ADC) User's Guide. This document provides an overview of the Successive Approximation (SAR) Analog to Digital Converter (ADC) on the TMS320VC5505/5504 Digital Signal Processor (DSP). The SAR is a 10-bit ADC using a switched capacitor architecture which converts an analog input signal to a digital value.

SPRUFP3 — TMS320VC5505 Digital Signal Processor (DSP) Liquid Crystal Display Controller (LCDC) User's Guide. This document describes the liquid crystal display controller (LCDC) in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. The LCD controller includes a LCD Interface Display Driver (LIDD) controller.

SPRUFP4 — TMS320VC5505/5504 Digital Signal Processor (DSP) Inter-IC Sound (I2S) Bus User's Guide. This document describes the features and operation of Inter-IC Sound (I2S) Bus in the TMS320VC5505/5504 Digital Signal Processor (DSP) device. This peripheral allows serial transfer of full duplex streaming data, usually streaming audio, between DSP and an external I2S peripheral device such as an audio codec.

Inter-IC Sound (I2S) Bus

1 Introduction

This document describes the features and operation of Inter-IC Sound (I2S) Bus. This peripheral allows serial transfer of full duplex streaming data, usually streaming audio, between DSP and an external I2S peripheral such as an audio codec.

1.1 Purpose of the Peripheral

The I2S bus is used as an interface for full-duplex serial ports such as those found in audio or voice-band analog to digital converters (ADC) to acquire audio signals or digital-to analog converters (DAC) to drive speakers and headphones.

1.2 Features

The I2S bus supports the following features:

- Full-duplex (transmit and receive) communication.
- Double buffered data registers that allow for continuous data stream.
- Most significant bit (MSB) - first data transfers.
- I2S/Left-justified and DSP serial data communication formats with a data delay of 1 or 2 bits.
- Data word-lengths of 8, 10, 12, 14, 16, 18, 20, 24, or 32 bits.
- Ability to sign-extend received data samples for easy use in signal processing algorithms.
- Ability to pack multiple data words into CPU or DMA accessible data registers to reduce interrupts for more efficient operation.
- Programmable polarity for both frame synchronization and bit-serial clocks.
- Digital loopback of data from transmit to receive data register(s) for application code debug.
- Stereo (in I2S/Left-justified or DSP data formats) or mono (in DSP data format) mode.
- Programmable divider for serial data clock (bit-clock) generation when I2S bus is used as a master device.
- Programmable divider for frame sync clock generation when I2S bus is used as the master device.
- Detection of over-run, under-run, and frame-synchronization error conditions.

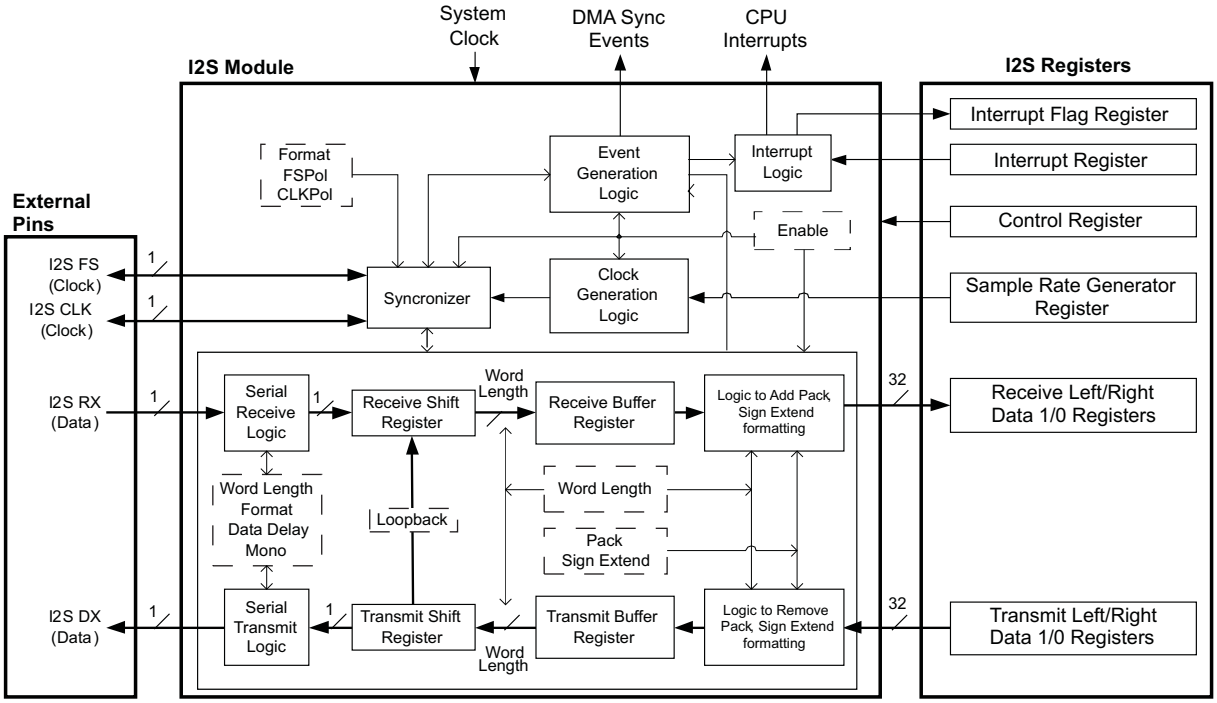
The DSP includes four independent I2S modules .

1.3 Functional Block Diagram

Figure 1 is a functional block diagram of the I2S bus illustrating the different control, data transfer, clock generation and event management blocks and their interactions. The I2S peripheral has a set of control and data registers which the CPU can access through its I/O space. The DMA can also make 32-bit accesses to receive and transmit data registers for efficient data transfer.

The bus is configured by writing to the I2S*n* Serializer Control Register (I2SSCTRL) bit fields. The bit fields in this register determine the communication protocol over the I2S bus and the arrangement of data in the data registers.

Figure 1. Functional Block Diagram



Data on the I2S*n*_RX pin is shifted serially into the Receive Shift Register and then copied into the Receive Buffer Register. The data is then copied to I2S*n* Receive Left/Right Data *n* Registers. For each channel (left and right), these registers can be accessed as two 16-bit registers by the CPU or as a 32-bit register by the DMA. Similarly, the I2S*n* Transmit Left/Right Data *n* Registers store the data to be transmitted out of the I2S peripheral. The CPU or DMA writes the transmit data to the I2S*n* Transmit Left/Right Data *n* Registers which is then copied to the Transmit Shift Register via the Transmit Buffer Register and shifted serially out to I2S*n*_DX pin. This structure allows internal data movement and external data communications simultaneously. Data handling and movement is discussed in further detail in later sections.

The control block consists of internal clock generation, frame synchronization signal generation, and their control. The I2S*n* Sample Rate Generator Register (I2SSRATE) contains fields to configure the frame-synchronization and bit-clock dividers to drive the I2S*n*_FS and I2S*n*_CLK clocks when the I2S peripheral is configured as a master device. When configured as a slave device, the internal clock generation logic is disabled and frame synchronization is performed on the clocks generated by the external master I2S device (see Section 2.2). The polarities of the bit-clock and the frame-synchronization can be set by the CLKPOL bit and FSPOL bit respectively in the I2SSCTRL register. The I2S supports a data delay of 1 bit or 2 bits as configured by the DATADLY bit in the I2SSCTRL register.

The I2S peripheral can be configured to interrupt the CPU by writing to the I2Sn Interrupt Mask Register (I2SINTMASK). When interrupts are enabled, the event-generation block posts a transmit interrupt when transmit data registers are empty and a receive interrupt when receive data registers are full. The corresponding flag is set in the I2Sn Interrupt Flag Register (I2SINTFL). In addition to data transaction interrupts, error events are also flagged in this register. Error events are not connected to interrupts on the CPU. The I2S also sends synchronization events corresponding to the transmit and receive events to the DMA controller associated with the I2S module. The I2SINTMASK register has no effect on DMA sync signal generation events. (See Section 2.10, Section 2.11, Section 3.7 and Section 3.8)

1.4 Industry Standard(s) Compliance

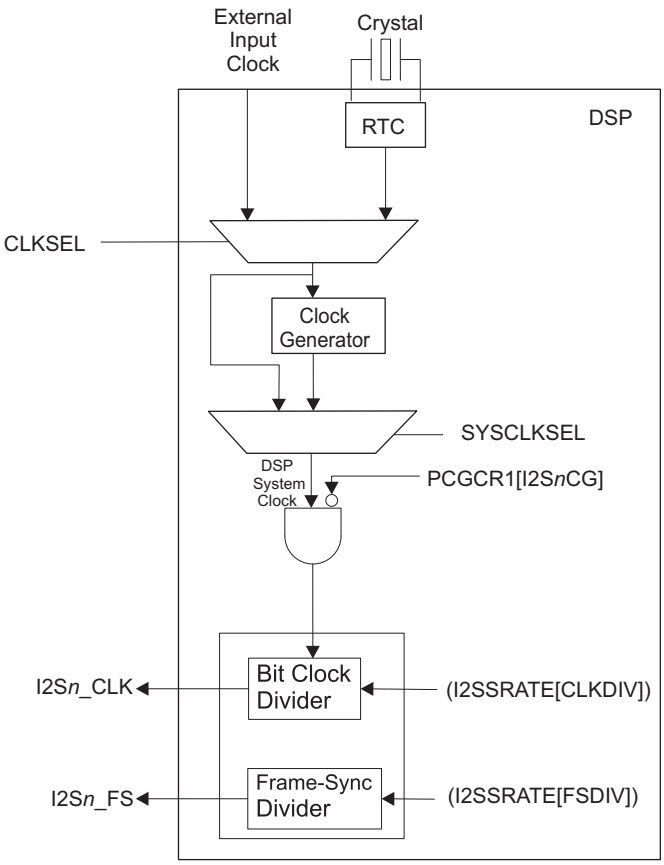
This Inter-IC Sound (I2S) Bus is compliant to industry I2S Bus Standard.

2 Architecture

2.1 Clock Control

As shown in Figure 2, the I2S bus is driven by the system clock. Unused I2S modules can be independently idled (clock-gated) via the peripheral clock gating configuration register 1 (PCGCR 1) for power dissipation savings. Each I2S bus should be brought out of idle before being programmed. For more details, see Section 2.12.

Figure 2. Inter-IC Sound Clock Control Diagram



If the I2S bus is configured as the master device, the DSP clock generator may need to be programmed to achieve an appropriate system clock so that the I2S clock dividers can generate the required clock rates. For more information on the DSP clock generation options, see the *TMS320VC5505/5504 DSP System Guide* (SPRUFP0).

2.2 I2S Clock Generator

The I2S*n* Sample Rate Generator Register (I2SSRATE) controls the clock generation logic in the I2S bus. In slave mode (MODE = 0 in I2SSCTRL - see Section 3.1), the required clock signals (I2Sn_CLK and I2Sn_FS) are generated by the external master I2S device and the internal I2S clock generator is not used. Configuring the I2SSRATE register has no effect in this mode. However, when configured as a master device (MODE = 1), the I2S module generates these clocks by dividing the system clock by a value calculated from the CLKDIV and FSDIV fields programmed in the I2SSRATE register (see Section 3.2). The clocks can be calculated as shown below:

I2S*n*_CLK = SystemClock / ($2^{CLKDIV+1}$)

I2S*n*_FS = I2S*n*_CLK / ($2^{FSDIV+3}$)

I2S*n*_CLK is the bit-clock that determines the rate at which data bits are transferred on the serial I2S bus. I2S*n*_FS is referred to as the frame-sync clock or word clock and is the rate at which a data word is transferred to and/or from the I2S module. This can also be seen as the frequency at which data (audio from a microphone for example) is sampled by an analog-to-digital converter (ADC) or an audio codec.

The clock divide-by value, $2^{FSDIV+3}$, that derives the frame-sync clock from the bit-clock (as shown above), gives the number of data bits (bit-clocks) in one cycle of the frame-sync clock. Since one cycle of the frame-sync clock should accommodate two data words (one left and one right channel data word) for stereo operation and one data word (one left channel only) for mono operation, the following restrictions apply while choosing an appropriate setting for FSDIV:

$2^{FSDIV+3} \geq 2 * WDLNGTH$ (for stereo mode)

$2^{FSDIV+3} \geq WDLNGTH$ (for mono mode)

For example, to achieve a particular sampling rate of I2S*n*_FS = 48000 Hz with stereo operation of data length of 16 bits, the value of the FSDIV bit in the I2SSRATE register should be first chosen such that:

$2^{FSDIV+3} \geq 2*16 = 32.$

If we choose

FSDIV = 2 (010 binary)

the resultant I2S*n*_CLK can be calculated as:

I2S*n*_CLK = I2S*n*_FS * ($2^{FSDIV+3}$) = 48000 * 32 = 1.536 MHz.

Based on application requirements, if the DSP needs to be run at a minimum system clock or DSP clock of 45 MHz, the CLKDIV bit in the I2SSRATE register can be chosen such that,

SystemClock ≥ I2S*n*_CLK * $2^{CLKDIV+1}$ ≥ 45 MHz

Hence we should choose:

CLKDIV = 4 (100 binary)

Which will give us,

SystemClock = 1.536 MHz * 2^{4+1} = 49.15 MHz

As a result, the DSP clock generator should be configured to generate the required clock of 49.15 MHz. Due to limitations/restrictions of the DSP clock generator, it may not be possible to generate the exact system clock required, in which case I2S*n*_FS will deviate from the expected value. While it is sufficient to choose a setting for FSDIV such that 2FSDIV+3 is equal to the required number of data bits per frame-sync clock (as shown in example above), it may sometimes be necessary to choose a higher setting so that a faster bit-clock can be achieved. For a given system clock, this expedites transfer of data bits of the programmed word length on the I2S bus. As a result, the interrupts/events occur earlier in the frame-sync cycle, providing more time for the CPU/DMA to service the interrupt/event before the next interrupt/event. This is a particularly useful technique when the I2S uses CPU to handle data transfers to/from its data registers.

NOTE: I2S peripheral clock generator should only be configured if the I2S is configured as a master device. When the I2S is configured as the slave, the external master device supplies the required clocks.

2.3 Signal and Pin Descriptions

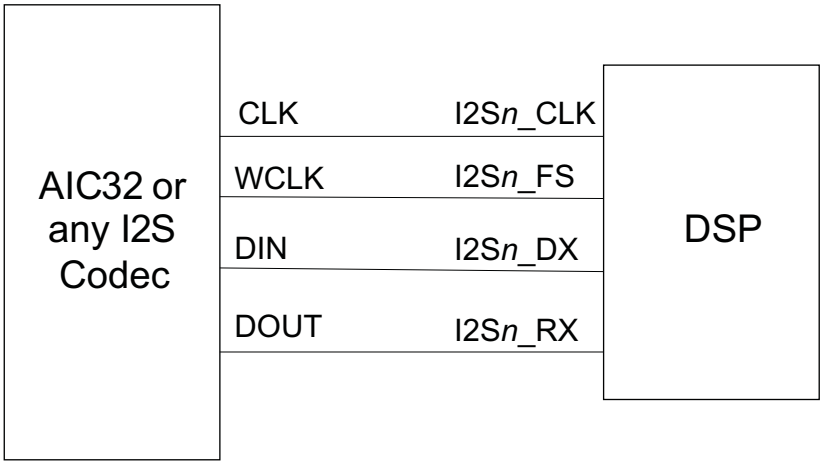
The I2S bus is a four-wire interface with two clock pins, bit-serial clock (I2S*n*_CLK) and frame-synchronization or word clock (I2S*n*_FS), and two data pins, serial data transmit (I2S*n*_DX) and serial data receive (I2S*n*_RX), for data communication as shown in Figure 1. The I2S*n*_CLK and I2S*n*_FS pins are bi-directional based on whether the I2S peripheral is configured as a master or slave device.

Table 1. I2S Signal Descriptions

Name	Signal	Description
I2S*n*_CLK	INPUT /OUTPUT	I2S Clock
I2S*n*_FS	INPUT /OUTPUT	I2S Frame Sync Clock
I2S*n*_DX	OUTPUT	I2S Data Transmit
I2S*n*_RX	INPUT	I2S Data Receive

The diagram below is a typical connection between I2S interface to an audio or voice-band Codec.

Figure 3. Block Diagram of I2S Interface to Audio/Voice Band Codec



2.3.1 Pin Multiplexing

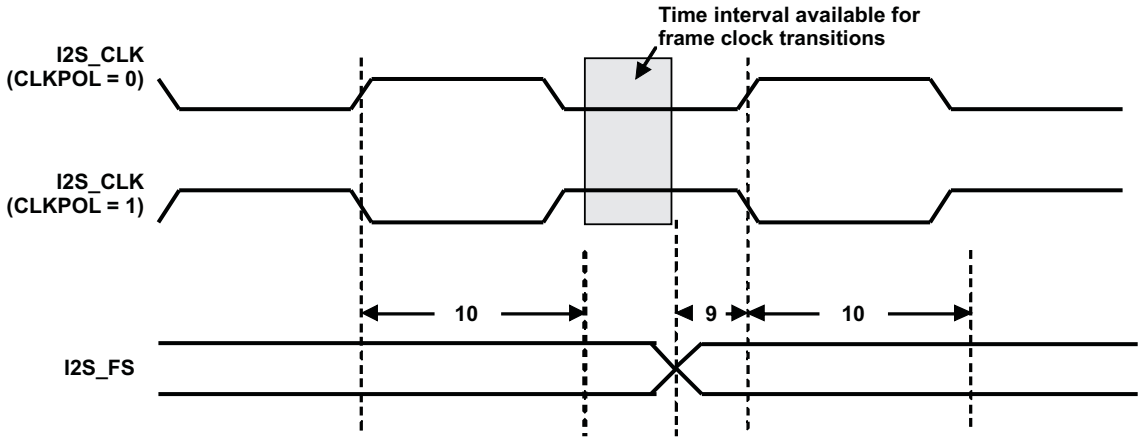
Depending on the I2S bus being used, the DSP should be configured to route those I2S signals to the multiplexed Serial Port 0, Serial Port 1, or Parallel Port pins by writing to the External Bus Selection Register (EBSR). For more information on pin multiplexing, see the *TMS320VC5505/5504 DSP System Guide* (SPRUFP0).

NOTE: Configuring the EBSR to route I2S0 or I2S1 signals to Serial Port0 or Serial Port1 respectively also routes those I2S interrupts to the CPU (see Section 2.10).

2.4 Frame Clock Timing Requirement in Slave Mode

When configured as the slave, frame clock (I2S_FS) is required to be latched on both edges of the bit clock (I2S_CLK), which are generated by the external master device. This imposes an additional constraint on the timing of I2S_FS as illustrated in Figure 4. The generated frame clock should meet the specified setup and hold requirements with respect to the sampling edge of the generated bit clock. For actual timing requirements, see the I2S section of the TMS320C55xx data sheet. These constraints imply that the frame clock transitions should occur in the time window as indicated by the shaded region in the figure.

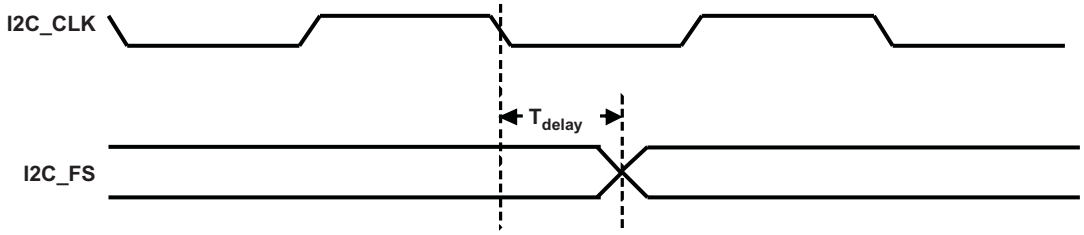
Figure 4. I2S Frame Clock Timing Constraint in Slave Mode



No.	Parameter	Description
9	$t_{su(FSV-CLKH)}$	Minimum setup time, I2S_FS valid before I2S_CLK high (CLKPOL = 0)
	$t_{su(FSV-CLKL)}$	Minimum setup time, I2S_FS valid before I2S_CLK low (CLKPOL = 1)
10	$t_{h(CLKH-FSV)}$	Minimum hold time, I2S_CLK high to I2S_FS (CLKPOL = 0)
	$t_{h(CLKL-FSV)}$	Minimum hold time, I2S_CLK low to I2S_FS (CLKPOL = 1)

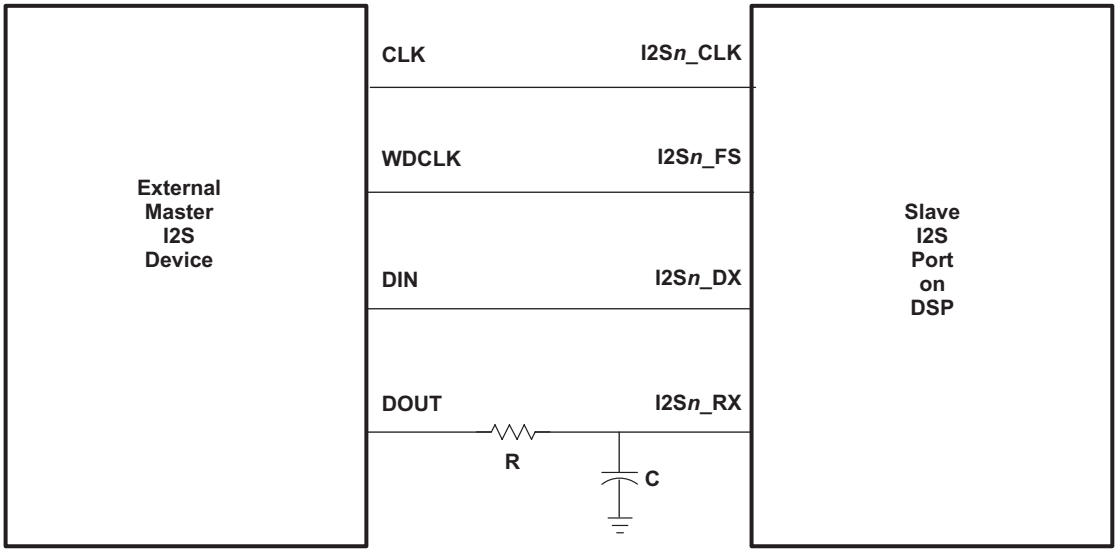
Devices (ADCs, DACS and audio/voice-band codecs) that interface to the I2S module usually only specify a maximum delay for the frame clock transition from the falling edge of the bit clock in master mode as indicated by parameter T_{delay} in Figure 5.

Figure 5. Typical Frame Clock Timing Specification



Synchronization issues may occur if the frame clock transitions close to the falling edge of the bit clock violating the previously described hold requirement resulting in incorrect data transfer. In these circumstances, the frame clock should be delayed with respect to the bit clock by introducing a time delay in its signal path as shown in Figure 6. The RC circuit delays the frame clock by a value given by the relation Trc = RC.

Figure 6. Delaying I2S Frame Clock to Overcome Synchronization Problems



NOTE: Signal should be probed as close to the TMS320C55xx device pins as possible for better results.

2.5 Protocol Description

The I2S bus communicates with a corresponding external I2S peripheral in a series of 1's and 0's. This series has a hierarchical organization that can be described in terms of bits, words, and frames.

A bit is the smallest entity in the serial stream. A "1" is represented by logic high on the data pin for the entire duration of a single bit clock. A "0" is represented by a logic low for the entire duration of a single bit clock.

A word is a group of bits that make up the data being transmitted or received. The length of the word is programmed by the user in the WDLNGTH field in the I2S*n* Serializer Control Register (I2SSCTRL).

A frame is a group of words (usually one – mono or two – stereo) that make up the data being transmitted or received. The number of bit clocks per frame and the frame rate (sampling frequency) is programmed by the user in the I2S*n* Sample Rate Generator Register (I2SSRATE).

I2S supports two serial data communication formats with external I2S devices: I2S/Left-justified format and DSP format. The I2S format is a specialized case of the more general left-justified data format. In DSP mode, the frame is marked between two consecutive pulses of the frame sync signal. On I2S, the frame is marked by a whole clock cycle of the frame sync signal with 50% duty cycle.

2.5.1 I2S/Left-Justified Format

In the left-justified format, the frame-synchronization or word clock has a 50% duty cycle indicating dual channel data fields with left channel data transferred during one half of the cycle and right channel data transferred during the other half. The MSB-first data is transferred serially, left justified in its own field with appropriate bit delays.

As shown in Figure 7, the typical I2S format utilizes left-justified format with a data delay of one bit and low frame synchronization pulse for left channel data and high pulse for right channel data. Serial data sent by the transmitter may be synchronized with either the trailing or the leading edge of serial clock I2S*n*_CLK. However, the serial data must be latched by the receiver on the leading edge of I2S*n*_CLK. In this format, the MSB of the left channel is valid on the second leading edge of the bit-clock, I2S*n*_CLK after the trailing edge of the frame-synchronization clock, I2S*n*_FS. Similarly the MSB of the right channel is valid on the second leading edge of I2S*n*_CLK after the leading edge of I2S*n*_FS.

Figure 7. Timing Diagram for Left-Justified Mode with Inverse Frame-Sync Polarity and One-Bit Delay

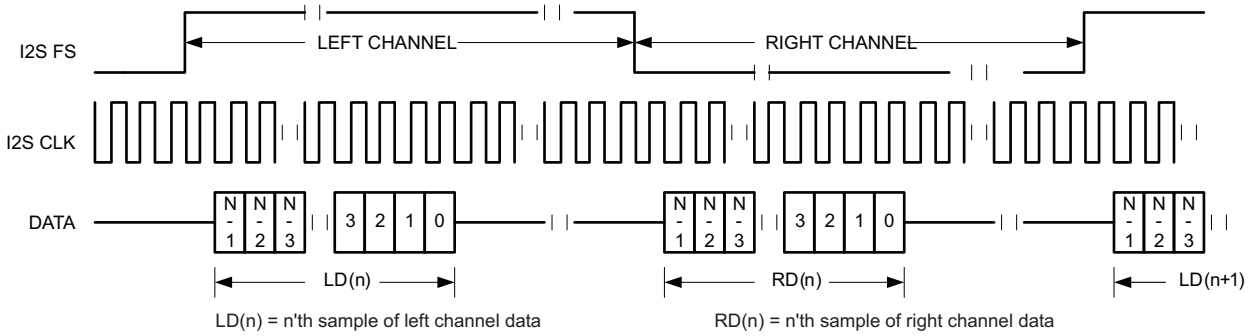


Figure 8. Timing Diagram for I2S Mode

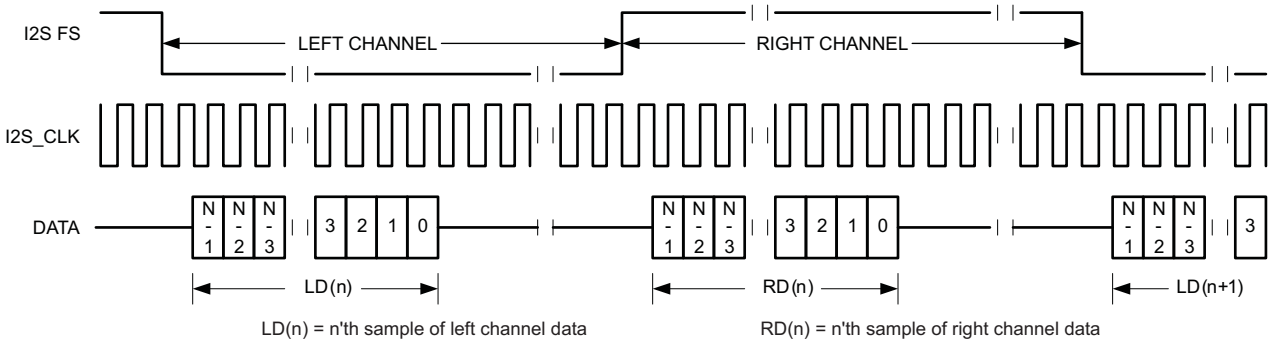
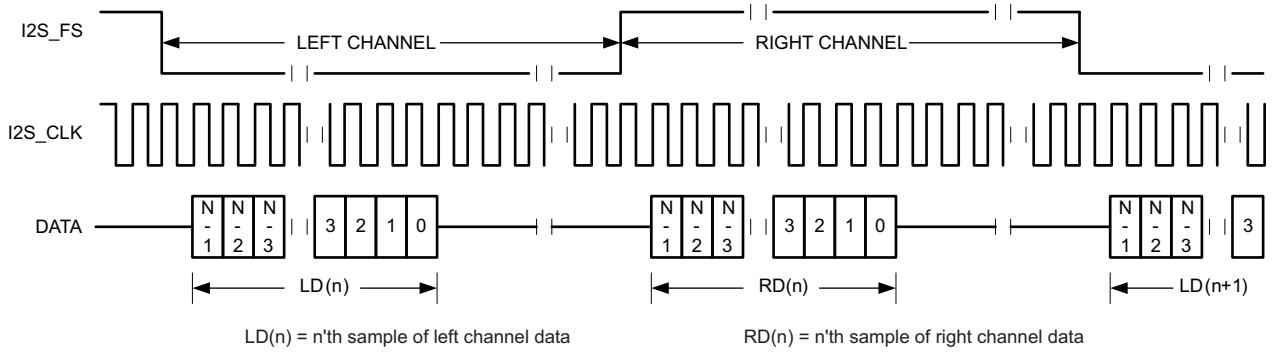


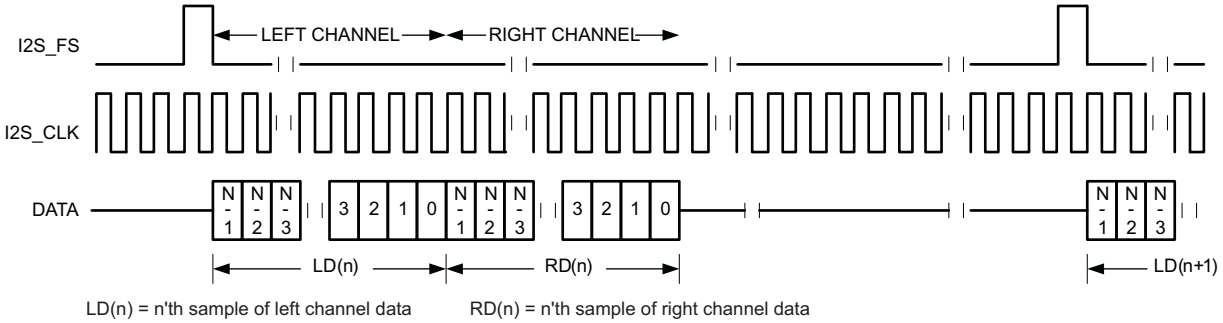
Figure 9. Timing Diagram for I2S Mode with Inverse Bit-Clock Polarity



2.5.2 DSP Format

In DSP format, the trailing edge of the frame-synchronization pulse, I2S*n*_FS, starts the data transfer with the left channel data first and immediately followed by the right channel data. Each data bit is valid on the trailing edge of the bit-clock, I2S*n*_CLK. The first data sample can be delayed by 1 bit or 2 bits after the trailing edge of I2S*n*_FS. With one bit delay, the MSB coincides with the trailing edge of I2S*n*_FS. With two bit delay, the MSB follows the trailing edge of I2S*n*_FS after one I2S*n*_CLK. Figure 10 illustrates DSP format operation with a one-bit data delay.

Figure 10. Timing Diagram for DSP Mode With One-Bit Delay



NOTE:

- The I2S*n*_DX and I2S*n*_RX pins are tri-stated during unused bit clocks in a frame.
- In I2S/Left-justified format:
  - Mono operation is not supported due to the 50% duty cycle restriction on the frame-synchronization clock.
  - The number of I2S*n*_CLKs should be greater than or equal to twice the configured data word-length.
- In DSP format:
  - The number of I2S*n*_CLKs should be greater than or equal to twice the configured data word-length for stereo operation.
  - The number of I2S*n*_CLKs should be greater than or equal to the configured data word-length for mono operation.

2.6 I2S Data Transfer and Control Behavior

When the I2S module is enabled, MSB-first data transfer starts when the appropriate level is detected on the frame-sync clock. Data in the Transmit Shift Register is shifted out serially to the I2Sn_DX while data bits are shifted in serially from the I2Sn_RX pin to the Receive Shift Register on the falling or leading edge of the bit-clock as programmed in the CLKPOL field of the I2SSCTRL. Data for the left channel is transferred first followed by the right channel data.

The module generates the transmit interrupt/event (the transmit and receive interrupts should be enabled in the I2SINTMASK register if CPU transfers are desired; if DMA is used to transfer data these interrupts should be disabled – see Sections 2.8 and 2.9) to indicate that the Transmit Left/Right Data1/0 registers should be filled with valid data for the next set of transfers. The CPU or DMA servicing this interrupt/event writes the next valid data to the above mentioned registers. Failure to do so before the next frame-sync cycle will result in the OUERROR being flagged in the I2SINTFL register (assuming that error detection has been enabled in the I2SINTMASK register). At the next frame-sync cycle, data from the Transmit Left/Right Data1/0 registers into the Transmit Buffer register and then to the Transmit Shift register.

NOTE: Data should be written into the Transmit Left/Right Data1/0 registers only on a transmit interrupt. Data can not be preloaded into these registers before enabling the I2S module or before receiving the first transmit interrupt.

When the required number of data words is received in the Receive Shift register (one for mono or two for stereo), the data is moved into the Receive Buffer register and ultimately into the Receive Left/Right Data1/0 registers. A receive interrupt/event is generated at this point. The CPU or DMA servicing this interrupt reads the register into memory. Failure to do so before the next frame-sync cycle will result in the OUERROR being flagged in the I2SINTFL register (assuming that error detection has been enabled in the I2SINTMASK register).

Transmit and receive interrupts/events are continuously generated after every transfer from the transmit data registers to the transmit buffer register and from the receive buffer register to the receive data registers respectively. If packed mode is enabled (PACK = 1 in I2SSCTRL), interrupts/events are generated after all the required number of data words have been transmitted/received (see Section 2.8).

2.7 I2S Data Transfer Latency

Due to the buffered nature of the I2S module, there exists some latency in the transmit and receive paths (from the Transmit Data registers to the I2S_DX pin and I2S_RX pin to Receive Data registers) which is dependent on the desired configuration of the module. The latency may not be of consequence when the I2S module in its intended scope of a streaming audio peripheral. However, it is documented here in the interests of completeness. It will also help explain observed loopback data (LOOPBACK=1 in I2SSCTRL) as the latency is also present in loopback mode.

2.7.1 Transmit Path Latency

After the I2S is enabled, the first valid data sample on the I2Sn_DX pin will appear after:

- Five frame-sync clocks if PACK mode is used (PACK=1 in I2SSCTRL) or,
- Three frame-sync clocks if PACK mode is not used (PACK=0 in I2SSCTRL)

Hence there is a latency of three or five samples (for each channel) before the first data sample that is written to the Transmit Left/Right Data 1/0 registers after the first transmit interrupt/event. During this time, the I2S transmits random data bits which should be discarded or ignored.

2.7.2 Receive Path Latency

After the I2S is enabled, the receive path starts receiving data after:

- 1 or 2 frame-sync clocks for 8-, 18-, 20-, 24-, or 32-bit data depending on other configuration
- 1, 2 or 3 frame-sync clocks for 10-, 12-, 14-, or 16-bit data depending on other configuration

2.7.3 Loopback Path Latency

The internal loopback mode (LOOPBACK=1 in I2SSCTRL) can be used as a debug tool to verify the user's program to service I2S interrupts/events. This is different from an external loopback which would require the I2Sn_DX pin to be connected to the I2Sn_RX pin. In the internal loopback mode, data from the Transmit Shift register is directly routed to the Receive Shift register, changing the data latency as given below:

- If pack mode is used (PACK=1 in I2SSCTRL)
  - Ignore the first 6 samples received for 8-bit data and FSDIV=000 in I2SSRATE
  - Ignore the first 5 samples received for 8-bit data and FSDIV > 000 in I2SSRATE
  - Ignore the first 6 samples received for 10-, 12-, 14- or 16-bit data
- If pack mode is not used (PACK=0 in I2SSCTRL), ignore the first 2 samples received.

2.8 Data Packing and Sign Extension Options

The I2S bus supports the use of packed (multiple) and/or sign extended data words in its data registers to reduce software overheads in servicing interrupts for every data sample and for ease of data handling in software algorithms.

NOTE: Using the Pack or Sign Extend options does not affect transmission of data samples over the serial I2S bus; it only affects how data words are arranged in the Receive and Transmit Data Registers.

2.8.1 Data Pack Mode

Setting the PACK bit field in the I2SSCTRL register enables data packing in the 32-bit I2S data registers for word-lengths of 8, 10, 12, 14 and 16 bits. This mode can be used in the following scenario:

- Using DMA to transfer data samples: to make better use of data buffers.
- Using CPU to transfer data samples: to reduce interrupt overheads.

During the receive operation, the I2S bus puts successive data samples into the I2S*n* Receive Left/Right Data *n* Registers (Data 1 register first, Data 0 register next) before generating the interrupt/event. The transmit data is expected in a similar format in the I2S*n* Transmit Left/Right Data *n* Registers. Four 8-bit data samples or two 10, 12, 14 or 16-bit data samples can be packed in the data registers, as shown in Section 2.8.3.

The advantages of using the PACK mode can be seen as given below:

- Reduces the number of I2S interrupts/events, which results in reducing interrupt overheads and the better use of bus bandwidth.
- Efficient use of internal/on-chip memory if DMA is used for transferring data between I2S and main memory. Since the DMA transfers a double-word (all 32-bits of the I2S*n* Transfer Left/Right Data *n* Registers) during each transaction, using packed I2S*n* Receive/Transmit Left/Right Data *n* Registers results in efficient transfer of data samples. Hence, for a given number of samples, size of data buffers is reduced by a factor of two for 10-, 12-, 14- or 16-bit word length or by a factor of four for 8-bit word length.

Figure 11 and Figure 12 illustrate packed and unpacked data receive behavior for mono transmission of four 12-bit data samples (sign extension is not enabled). When pack mode is not used, the I2S module stores 12-bit data left-justified in MSW (with trailing zeros) and generates DMA event resulting in zeros stored in alternate memory locations. When pack mode is used, the module packs 12-bit data left-justified in MSW first and then LSW (with trailing zeros) and generates DMA event once every two transfers resulting in better utilization of memory.

Figure 11. Example of Unpacked 12-Bit Data Receive

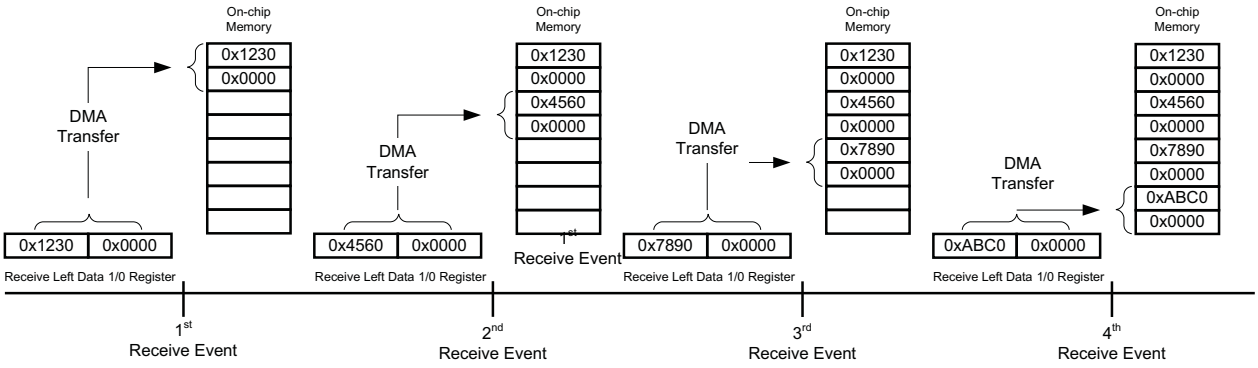
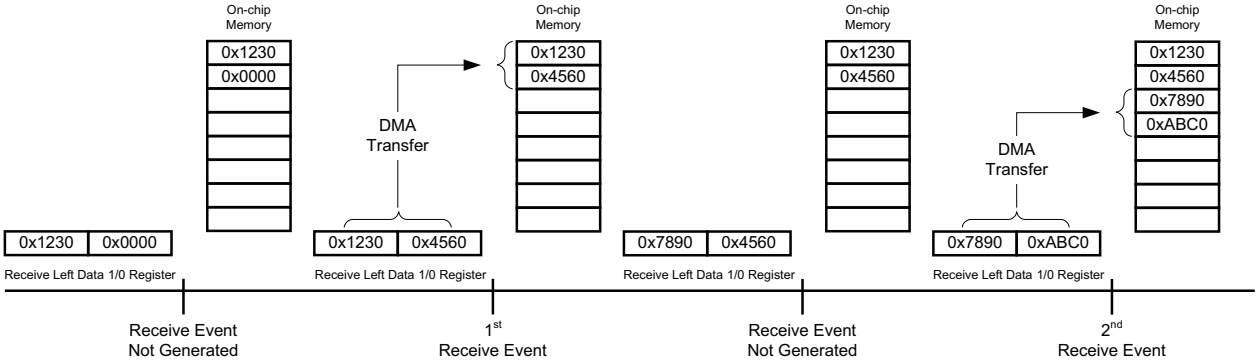


Figure 12. Example of Packed 12-Bit Data Receive



2.8.2 Data Sign Extend Mode

The I2S peripherals can be configured to work with sign extended data samples for ease of use in commonly used S16 or S32 data representations in signal processing algorithms. Data words of lengths 8-, 10-, 12- or 14-bits are sign-extended (right-justified) to 16 bits (8-bit data cannot be sign extended in packed mode as four samples are packed into the 32-bit data registers). Data of word lengths 18-, 20- or 24-bits are sign extended (right-justified) to 32 bits.

The choice of PACK, SIGN_EXT and WDLNGTH bit options in the I2SSCTRL register affects the arrangement of received data in the I2S*n* Receive Left/Right Data *n* Registers. Similarly, the I2S peripheral expects data to be arranged in a similar fashion in the I2S*n* Transmit Left/Right Data *n* Registers for the correct data value to be shifted out to the I2S*n*_DX pin.

Table 2 illustrates sign extension behavior with an example.

Table 2. Example of Sign Extension Behavior

		Data Register With SIGN_EXT=0		Data Register With SIGN_EXT=1	
Word Length	Data on I2S Bus	Data 1 Register	Data 0 Register	Data 1 Register	Data 0 Register
8	0xA1	0xA100	0x0000	0xFFA1	0x0000
10	0x2B7	0xAD80	0x0000	0xFEB7	0x0000
12	0x6AA	0x6AA0	0x0000	0x06AA	0x0000
14	0x3999	0xE666	0x0000	0xF999	0x0000
16	0x29CC	0x29CC	0x0000	0x29CC	0x0000
18	0x19EFA	0x67BE	0x8000	0x0001	0x9EFA
20	0x7ADE1	0x7ADE	0x1000	0x0007	0xADE1
24	0x98A311	0x98A3	0x1100	0xFF98	0xA311
32	0xD16AEE09	0xD16A	0xEE09	0xD16A	0xEE09

2.8.3 PACK and Sign Extend Data Arrangement for Various Word Lengths

The tables in this section describe sign extended and packed data arrangement for each configurable word length. Data samples are indicated by x[0], x[1], x[2] and x[3] in increasing temporal order (x[3] is the most recent sample).

2.8.3.1 8 - Bit Word Length

Table 3. PACK and Sign Extend Data Arrangement for 8 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register					I2S Receive/Transmit Left or Right Data 0 Register				
PACK = 0 SIGN_EXT = 0	15	8	7	0		15				0
	x(0)		0			0				
PACK = 0 SIGN_EXT = 1	15	8	7	0		15				0
	Sign Bits (Bit #7)		x(0)			0				
PACK = 1 SIGN_EXT = 0	15	8	7	0		15	8	7		0
	x(0)		x(1)			x(2)		x(3)		
PACK = 1 SIGN_EXT = 1	15	8	7	0		15	8	7		
	x(0)		x(1)			x(2)		x(3)		

2.8.3.2 10 - Bit Word Length

Table 4. PACK and Sign Extend Data Arrangement for 10 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register					I2S Receive/Transmit Left or Right Data 0 Register				
PACK = 0 SIGN_EXT = 0	15	6	5	0		15				0
	x(0)		0			0				
PACK = 0 SIGN_EXT = 1	15	10	9	0		15				0
	Sign Bits (Bit #9)		x(0)			0				
PACK = 1 SIGN_EXT = 0	15	6	5	0		15	6	5		0
	x(0)		0			x(1)		0		
PACK = 1 SIGN_EXT = 1	15	10	9	0		15	10	9		
	Sign Bits (Bit #9)		x(0)			Sign Bits (Bit #9)		x(1)		

2.8.3.3 12 - Bit Word Length

Table 5. PACK and Sign Extend Data Arrangement for 12 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register					I2S Receive/Transmit Left or Right Data 0 Register				
PACK = 0 SIGN_EXT = 0	15	4	3	0		15				0
	x(0)		0			0				
PACK = 0 SIGN_EXT = 1	15	12	11	0		15				0
	Sign Bits (Bit #11)		x(0)			0				
PACK = 1 SIGN_EXT = 0	15	4	3	0		15	4	3		0
	x(0)		0			x(1)		0		
PACK = 1 SIGN_EXT = 1	15	12	11	0		15	12	11		
	Sign Bits (Bit #11)		x(0)			Sign Bits (Bit #11)		x(1)		

2.8.3.4 14 - Bit Word Length

Table 6. PACK and Sign Extend Data Arrangement for 14 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register					I2S Receive/Transmit Left or Right Data 0 Register				
PACK = 0 SIGN_EXT = 0	15	2	1	0		15				0
	x(0)		0			0				
PACK = 0 SIGN_EXT = 1	15	14	13	0		15				0
	Sign Bits (Bit #13)		x(0)			0				
PACK = 1 SIGN_EXT = 0	15	2	1	0		15	2	1		0
	x(0)		0			x(1)		0		
PACK = 1 SIGN_EXT = 1	15	14	13	0		15	14	13		
	Sign Bits (Bit #13)		x(0)			Sign Bits (Bit #13)		x(1)		

2.8.3.5 16 - Bit Word Length

Table 7. PACK and Sign Extend Data Arrangement for 16 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register		I2S Receive/Transmit Left or Right Data 0 Register	
PACK = 0 SIGN_EXT = 0	15	0	15	0
	x(0)		0	
PACK = 0 SIGN_EXT = 1	15	0	15	0
	x(0)		0	
PACK = 1 SIGN_EXT = 0	15	0	15	0
	x(0)		x(1)	
PACK = 1 SIGN_EXT = 1	15	0	15	
	x(0)		x(1)	

2.8.3.6 18 - Bit Word Length

Table 8. PACK and Sign Extend Data Arrangement for 18 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15		0	15	14 13	0
	x(0) Bits[17-2]			x(0) Bits[1-0]	0	
PACK = 0 SIGN_EXT = 1	15	2 1	0	15		0
	Sign Bits(Bit #17)	x(0) Bits[17-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported			Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported			Not Supported		

2.8.3.7 20 - Bit Word Length

Table 9. PACK and Sign Extend Data Arrangement for 20 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15		0	15	12 11	0
	x(0) Bits[19-4]			x(0) Bits[3-0]	0	
PACK = 0 SIGN_EXT = 1	15	4 3	0	15		0
	Sign Bits(Bit #19)	x(0) Bits[19-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported			Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported			Not Supported		

2.8.3.8 24 - Bit Word Length

Table 10. PACK and Sign Extend Data Arrangement for 24 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register			I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15		0	15	8 7	0
	x(0) Bits[23-8]			x(0) Bits[7-0]		0
PACK = 0 SIGN_EXT = 1	15	8 7	0	15		0
	Sign Bits(Bit #23)	x(0) Bits[23-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported			Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported			Not Supported		

2.8.3.9 32 - Bit Word Length

Table 11. PACK and Sign Extend Data Arrangement for 32 - Bit Word Length

	I2S Receive/Transmit Left or Right Data 1 Register		I2S Receive/Transmit Left or Right Data 0 Register		
PACK = 0 SIGN_EXT = 0	15	0	15		0
	x(0) Bits[32-16]		x(0) Bits[15-0]		
PACK = 0 SIGN_EXT = 1	15	0	15		0
	x(0) Bits[32-16]		x(0) Bits[15-0]		
PACK = 1 SIGN_EXT = 0	Not Supported		Not Supported		
PACK = 1 SIGN_EXT = 1	Not Supported		Not Supported		

2.9 Reset Considerations

The I2S bus has two reset sources: software reset and hardware reset.

2.9.1 Software Reset Considerations

The I2S bus can be reset by software through the Peripheral Reset Control Register (PRCR). The software reset is very similar to hardware reset with the only exception being that the data in shift registers are not reset. For more details on PRCR, see the *TMS320VC5505 DSP System Guide* (SPRUFP0).

2.9.2 Hardware Reset Considerations

A hardware reset is always initiated during a full chip reset. When a hardware reset occurs, all the registers of the I2S bus are set to their default values.

2.10 Interrupt Support

Every I2S bus supports transmit and receive data-transfer interrupts/events to CPU or DMA and flags two error conditions. These are enabled or disabled by writing to the I2SINTMASK register. There are separate data-transfer interrupt mask bits for stereo or mono operating modes. The interrupts are also flagged in the I2SINTFL.

For Stereo operating mode (MONO=0 in I2SSCTRL) and with the corresponding STEREO interrupts enabled, transmit/receive interrupts are generated after the right channel data has been transferred (since right channel data is always transmitted/received last). In Mono mode (MONO=1 in I2SSCTRL) , only left channel data is transferred, hence the interrupts are generated after the transmit/receive of left channel data. Mono mode can only be used with DSP format (FRMT = 1 in I2SSCRTL).

NOTE: I2S bus behavior is not defined if stereo and mono interrupts are simultaneously enabled. The I2S module flags an OUERROR (if enabled in I2SINTMASK) incorrectly when the module is enabled for the first time. The user program should ignore this error.

2.10.1 Interrupt Multiplexing

Interrupts to the CPU of two I2S peripherals, I2S0 and I2S1, are multiplexed with MMC/SD0 and MMC/SD1 interrupts. Configuring Serial Port 0 or Serial Port 1 fields in the External Bus Selection register to route I2S signals to the external pins, also routes the corresponding I2S interrupts to the CPU. For details, see the *TMS320VC5505 DSP System Guide* (SPRUFP0).

2.11 DMA Event Support

Each I2S bus has access to one of the four DMA peripherals on the DSP as shown in Table 12. To enable seamless transfers using the DMA, the I2S bus generates data-transfer events to DMA irrespective of the value configured in the I2SINTMASK register. Hence, it is recommended to disable CPU data-transfer interrupts in the I2SINTMASK register when using the DMA for data transfers. For Stereo operating mode (MONO=0 in I2SSCTRL), transmit/receive events are generated after the right channel data has been transferred (since right channel data is always transmitted/received last). In Mono mode (MONO=1 in I2SSCTRL) , only left channel data is transferred, hence the events are generated after the transmit/receive of left channel data. Mono mode can only be used with DSP format (FRMT = 1 in I2SSCRTL). For details on DMA configurations options, see the *TMS320VC5505/5504 DSP System Guide* (SPRUFP0).

Table 12. DMA Access to I2S

DMA Engine	I2S Bus
DMA0	I2S0
DMA1	I2S2
DMA2	I2S3
DMA3	I2S1

NOTE: The DMA can be used to transfer data samples in single double-word bursts from the I2S to:

- On-chip memory (DARAM/SARAM)
- NAND/NOR memory via EMIF

2.12 Power Management

The I2S peripherals can be put into idle condition to save power consumption. This is achieved by gating the system clock to the I2S peripherals via individual fields in the Peripheral Clock Gating Configuration registers described in the *TMS320VC5505 DSP System Guide* (SPRUFP0).

2.13 Emulation Considerations

An emulation halt will not stop I2S operation and an over-run/under-run error condition will be flagged (if enabled) in the I2SINTFL register if the CPU or DMA is unable to service I2S interrupts/events as a result of the emulation halt.

Refreshing CPU registers or memory contents in Code Composer Studio when the I2S is running could result in the DSP missing real-time operation due to JTAG communication overheads over the internal data buses. This would result in over-run/under-run errors being flagged (if enabled) in the I2SINTFL register.

2.14 Steps for I2S Configuration and I2S Interrupt Service Routine (ISR)

A sequence of steps for configuring an I2S bus and servicing interrupts in an interrupt service routine (ISR) are given below:

2.14.1 Initialization and Configuration Steps

- Bring I2S out of idle.
- If using DMA, reset DMA and MPORT idle bit-fields and run idle instruction to force MPORT out of idle.
- Disable DSP global interrupts.
- Clear DSP interrupt flag registers and enable appropriate I2S/DMA interrupts.
- If using DMA, configure DMA and sync DMA channel(s) to I2S sync event(s).
- Configure external I2S-compatible device.
- Enable DSP global interrupts.
- I2S Configuration:
  - Route I2S signals to external pins.
  - If I2S is the master device, configure the I2SSRATE register.
  - If using CPU interrupts, configure I2SINTMASK register to enable stereo receive/transmit interrupts for stereo mode operation or mono receive/transmit interrupts for mono mode operation (Peripheral behavior is not defined if both stereo and mono interrupts are enabled).
    - If the software/application is designed to respond to the detection of the error condition, enable error interrupts (disregard first OUERROR that is generated).
    - Do not enable stereo/mono TX or RX interrupts if DMA is used for data transfers.
  - Write desired configuration value to I2S*n* Serializer Control Register (I2SSCTRL) (If writing to individual bit fields, enable the I2S by setting the ENABLE bit in I2SSCTRL last.
    - The I2S bus now starts data conversion.
    - If configured as master device, I2S bus will generate interrupts/events even if external I2S-compatible device is not configured correctly and hence not executing data transfers.
    - If configured as a slave device, interrupt/event generation depends on proper operation of the external master device.
- If CPU is not performing other operations, the CPU can now be idled for power savings (if DMA is used for data transfers and software/application requires detection of error conditions, CPU may to read the I2S*n* Interrupt Flag Register (I2SINTFL) at regular intervals to check the error flag). For more information on the CPU idle mode, see the *TMS320VC5505 DSP System Guide* (SPRUFP0).
- An I2S (or DMA) interrupt will indicate completion of data transfer(s) and CPU is automatically brought of idle and the interrupt is taken.

2.14.2 ISR Steps (for CPU transfers)

Transmit and receive interrupts should have distinct interrupt service routines. A common framework is given below:

- Read the I2SINTFL register to reset the flags and if error detection is required, check for error flags .
- Read or write the I2S*n* Receive/Transmit Left/Right Data *n* Registers for receive and transmit interrupts respectively based on the I2S configuration (Mono, PACK, Sign Extend, Word Length options).
- Return from interrupt.

3 Registers

Table 13 through Table 16 lists the registers of the Inter-IC Sound (I2S) Bus. Refer to the sections listed for detailed information on each register.

Table 13. I2S0 Register Mapping Summary

CPU Word Address	Acronym	Description	
2800h	I2SSCTRL	I2S Serializer Control Register	Section 3.1
2804h	I2SSRATE	I2S Sample Rate Generator Register	Section 3.2
2808h	I2STXLT0	I2S Transmit Left Data 0 Register	Section 3.3
2809h	I2STXLT1	I2S Transmit Left Data 1 Register	Section 3.4
280Ch	I2STXRT0	I2S Transmit Right Data 0 Register	Section 3.5
280Dh	I2STXRT1	I2S Transmit Right Data 1 Register	Section 3.6
2810h	I2SINTFL	I2S Interrupt Flag Register	Section 3.7
2814h	I2SINTMASK	I2S Interrupt Mask Register	Section 3.8
2828h	I2SRXLT0	I2S Receive Left Data 0 Register	Section 3.9
2829h	I2SRXLT1	I2S Receive Left Data 1 Register	Section 3.10
282Ch	I2SRXRT0	I2S Receive Right Data 0 Register	Section 3.11
282Dh	I2SRXRT1	I2S Receive Right Data 1 Register	Section 3.12

Table 14. I2S1 Register Mapping Summary

CPU Word Address	Acronym	Description	
2900h	I2SSCTRL	I2S Serializer Control Register	Section 3.1
2904h	I2SSRATE	I2S Sample Rate Generator Register	Section 3.2
2908h	I2STXLT0	I2S Transmit Left Data 0 Register	Section 3.3
2909h	I2STXLT1	I2S Transmit Left Data 1 Register	Section 3.4
290Ch	I2STXRT0	I2S Transmit Right Data 0 Register	Section 3.5
290Dh	I2STXRT1	I2S Transmit Right Data 1 Register	Section 3.6
2910h	I2SINTFL	I2S Interrupt Flag Register	Section 3.7
2914h	I2SINTMASK	I2S Interrupt Mask Register	Section 3.8
2928h	I2SRXLT0	I2S Receive Left Data 0 Register	Section 3.9
2929h	I2SRXLT1	I2S Receive Left Data 1 Register	Section 3.10
292Ch	I2SRXRT0	I2S Receive Right Data 0 Register	Section 3.11
292Dh	I2SRXRT1	I2S Receive Right Data 1 Register	Section 3.12

Table 15. I2S2 Register Mapping Summary

CPU Word Address	Acronym	Description	
2A00h	I2SSCTRL	I2S Serializer Control Register	Section 3.1
2A04h	I2SSRATE	I2S Sample Rate Generator Register	Section 3.2
2A08h	I2STXLT0	I2S Transmit Left Data 0 Register	Section 3.3
2A09h	I2STXLT1	I2S Transmit Left Data 1 Register	Section 3.4
2A0Ch	I2STXRT0	I2S Transmit Right Data 0 Register	Section 3.5
2A0Dh	I2STXRT1	I2S Transmit Right Data 1 Register	Section 3.6
2A10h	I2SINTFL	I2S Interrupt Flag Register	Section 3.7
2A14h	I2SINTMASK	I2S Interrupt Mask Register	Section 3.8
2A28h	I2SRXLT0	I2S Receive Left Data 0 Register	Section 3.9
2A29h	I2SRXLT1	I2S Receive Left Data 1 Register	Section 3.10
2A2Ch	I2SRXRT0	I2S Receive Right Data 0 Register	Section 3.11

Table 15. I2S2 Register Mapping Summary (continued)

CPU Word Address	Acronym	Description	
2A2Dh	I2SRXRT1	I2S Receive Right Data 1 Register	Section 3.12

Table 16. I2S3 Register Mapping Summary

CPU Word Address	Acronym	Description	
2B00h	I2SSCTRL	I2S Serializer Control Register	Section 3.1
2B04h	I2SSRATE	I2S Sample Rate Generator Register	Section 3.2
2B08h	I2STXLT0	I2S Transmit Left Data 0 Register	Section 3.3
2B09h	I2STXLT1	I2S Transmit Left Data 1 Register	Section 3.4
2B0Ch	I2STXRT0	I2S Transmit Right Data 0 Register	Section 3.5
2B0Dh	I2STXRT1	I2S Transmit Right Data 1 Register	Section 3.6
2B10h	I2SINTFL	I2S Interrupt Flag Register	Section 3.7
2B14h	I2SINTMASK	I2S Interrupt Mask Register	Section 3.8
2B28h	I2SRXLT0	I2S Receive Left Data 0 Register	Section 3.9
2B29h	I2SRXLT1	I2S Receive Left Data 1 Register	Section 3.10
2B2Ch	I2SRXRT0	I2S Receive Right Data 0 Register	Section 3.11
2B2Dh	I2SRXRT1	I2S Receive Right Data 1 Register	Section 3.12

3.1 I2S*n* Serializer Control Register (I2SSCTRL)

The I2S*n* serializer control register (I2SSCTRL) is shown in Figure 13 and described in Table 17.

Figure 13. I2S*n* Serializer Control Register (I2SSCTRL)

15	14	13	12	11	10	9	8
ENABLE	Reserved		MONO	LOOPBACK	FSPOL	CLKPOL	DATADLY
R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	2	1	0
PACK	SIGN_EXT	WDLNGTH		MODE	FRMT
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 17. I2S*n* Serializer Control Register (I2SSCTRL) Field Descriptions

Bit	Field	Value	Description
15	ENABLE		Resets or enables the serializer transmission or reception.
		0	The I2S Bus (Data XFR, clock generation, and event gen logic) is disabled and held in reset state.
		1	I2S enabled.
14-13	Reserved	0	Reserved.
12	MONO		Sets I2S into mono or Stereo mode.
		0	Stereo mode.
		1	Mono mode. Valid only when bit 0, FRMT=1 (DSP Format).
11	LOOPBACK		Routes data from transmit shift register back to receive shift register for internal digital loopback.
		0	Normal operation, no loopback.
		1	Digital Loopback mode enabled.
10	FSPOL		Inverts I2S frame-synchronization polarity.
		0	The following: FRMT (bit 0): Function 0: (I2S/LJ) Frame-synchronization pulse is low for left word and high for right word 1: (DSP) Frame-synchronization is pulsed high
		1	The following: FRMT (bit 0): Function 0: (I2S/LJ) Frame-synchronization pulse is high for left word and low for right word 1: (DSP)Frame-synchronization is pulsed low
9	CLKPOL		Controls I2S clock polarity.
		0	The following: FRMT (bit 0): Function 0: (I2S/LJ) Receive data is sampled on the rising edge and transmit data shifted on the falling edge of the bit clock. 1: (DSP) Receive data is sampled on the falling edge and transmit data shifted on the rising edge of the bit clock.
		1	The following: FRMT (bit 0): Function 0: (I2S/LJ) Receive data is sampled on the falling edge and transmit data shifted on the rising edge of the bit clock. 1: (DSP) Receive data is sampled on the rising edge and transmit data falling on the rising edge of the bit clock.
8	DATADLY		Sets the I2S receive/transmit data delay.
		0	1-bit data delay.
		1	2-bit data delay.

Table 17. I2S*n* Serializer Control Register (I2SSCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
7	PACK		Enable data packing. Divides down the generation of interrupts so that data is packed into the 32-bit receive/transmit word registers for each channel (left/right).
		0	Data packing mode disabled.
		1	Data packing mode enabled.
			For more information about data packing, see Section 2.8.
6	SIGN_EXT		Enable sign extension of words.
		0	No sign extension.
		1	Received data is sign extended. Transmit data is expected to be sign extended.
			For more information about sign extension, see Section 2.8.
5-2	WDLNGTH		Choose serializer word length.
		0	8-bit data word.
		1h	10-bit data word.
		2h	12-bit data word.
		3h	14-bit data word.
		4h	16-bit data word.
		5h	18-bit data word.
		6h	20-bit data word.
		7h	24-bit data word.
		8h	32-bit data word.
		9h-Fh	Reserved.
1	MODE		Sets the serializer in master or slave mode.
		0	Serializer is configured as a slave. I2S*n*_CLK and I2S*n*_FS pins are configured as inputs. The bit-clock and frame-synchronization signals are derived from an external source and are provided directly to the I2S synchronizer without being further divided.
		1	Serializer is configured as a master. I2S*n*_CLK and I2S*n*_FS pins are configured as outputs and driven by the clock generators. The bit-clock and frame-synchronization signals are derived from the internal CPU clock.
0	FRMT		Sets the serializer data format.
		0	I2S/left-justified format.
		1	DSP format.

3.2 I2S*n* Sample Rate Generator Register (I2SSRATE)

The I2S*n* sample rate generator register (I2SSRATE) shown below controls the operation and various features of the sample rate generator. In master mode, the serializer generates the required clock signals by dividing the input clock by CLKDIV and additionally by FSDIV bit values programmed in the I2SSRATE register. In slave mode, the clocks are externally derived and fed directly to the serializer without division. Hence, this register is ignored in slave mode.

The I2S*n* sample rate generator register (I2SSRATE) is shown in Figure 14 and described in Table 18.

Figure 14. I2S*n* Sample Rate Generator Register (I2SSRATE)

15 6	5 3	2 0
Reserved	FSDIV	CLKDIV
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18. I2S*n* Sample Rate Generator Register (I2SSRATE) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5-3	FSDIV		Divider to generate I2S*n*_FS (frame-synchronization clock). The I2S*n*_CLK is divided down by the configured value to generate the frame-synchronization clock. (Has no effect when I2S is configured as slave device).
		0	Divide by 8
		1h	Divide by 16
		2h	Divide by 32
		3h	Divide by 64
		4h	Divide by 128
		5h	Divide by 256
		6h	Reserved
		7h	Reserved
2-0	CLKDIV		Divider to generate I2S*n*_CLK (bit-clock). The system clock (or DSP clock) to the I2S is divided down by the configured value to generate the bit clock. (Has no effect when I2S is configured as slave device).
		0	Divide by 2.
		1h	Divide by 4.
		2h	Divide by 8.
		3h	Divide by 16.
		4h	Divide by 32.
		5h	Divide by 64.
		6h	Divide by 128.
		7h	Divide by 256.

3.3 I2S*n* Transmit Left Data 0 Register (I2STXLT0)

The I2S*n* transmit left data 0 register (I2STXLT0) is shown in Figure 15 and described in Table 19.

Each I2S module has two double-word (32-bit) transmit data registers to hold left and right channel data respectively. Each double-word register is accessible as two 16-bit registers by the CPU or as a single double-word register by the DMA for efficient data transfer.

Figure 15. I2S*n* Transmit Left Data 0 Register (I2STXLT0)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 19. I2S*n* Transmit Left Data 0 Register (I2STXLT0) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Transmit left data lower 16 bits.

3.4 I2S*n* Transmit Left Data 1 Register (I2STXLT1)

The I2S*n* transmit left data 1 register (I2STXLT1) is shown in Figure 16 and described in Table 20.

Figure 16. I2S*n* Transmit Left Data 1 Register (I2STXLT1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 20. I2S*n* Transmit Left Data 1 Register (I2STXLT1) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Transmit left data upper 16 bits.

3.5 I2S*n* Transmit Right Data 0 Register (I2STXRT0)

TheI2S*n* transmit right data 0 register (I2STXRT0) is shown in Figure 17 and described in Table 21.

Figure 17. I2S*n* Transmit Right Data 0 Register (I2STXRT0)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 21. I2S*n* Transmit Right Data 0 Register (I2STXRT0) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Transmit right data lower 16 bits.

3.6 I2S*n* Transmit Right Data 1 Register (I2STXRT1)

The I2S*n* transmit right data 1 register (I2STXRT1) is shown in Figure 18 and described in Table 22.

Figure 18. I2S*n* Transmit Right Data 1 Register (I2STXRT1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 22. I2S*n* Transmit Right Data 1 Register (I2STXRT1) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Transmit right data upper 16 bits.

3.7 I2S*n* Interrupt Flag Register (I2SINTFL)

The I2S*n* interrupt flag register (I2SINTFL) is shown in Figure 19 and described in Table 23.

Figure 19. I2S*n* Interrupt Flag Register (I2SINTFL)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		XMITSTFL	XMITMONFL	RCVSTFL	RCVMONFL	FERRFL	OUERR
R-0		R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23. I2S*n* Interrupt Flag Register (I2SINTFL) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5	XMITSTFL		Stereo data transmit flag. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read.
		0	No pending stereo transmit interrupt.
		1	Stereo transmit interrupt pending. Write new data value to I2S Transmit Left and Right data 0 and 1 registers.
4	XMITMONFL		Mono data transmit flag. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read.
		0	No pending mono transmit interrupt.
		1	Mono transmit interrupt pending. Write new data value to Transmit Left Data 0 and 1 registers.
3	RCVSTFL		Stereo data receive flag. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read.
		0	No pending stereo receive interrupt.
		1	Stereo receive interrupt pending. Read Receive Left and Right data 0 and 1 registers.
2	RCVMONFL		Mono data receive flag. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read.
		0	No pending mono receive interrupt.
		1	Mono receive interrupt pending. Read Receive Left data 0 and 1 registers.
1	FERRFL		Frame-synchronization error flag. This bit is cleared on read.
		0	No frame-synchronization errors.
		1	Frame-synchronization error(s) occurred.
0	OUERRFL		Overrun or Underrun condition. This bit is cleared on read.
		0	No overrun/under-run errors.
		1	The data registers were not read from or written to before the receive/transmit buffer was overwritten.

3.8 I2S*n* Interrupt Mask Register (I2SINTMASK)

The I2S*n* interrupt mask register (I2SINTMASK) is shown in Figure 20 and described in Table 24.

Figure 20. I2S*n* Interrupt Mask Register (I2SINTMASK)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved		XMITST	XMITMON	RCVST	RCVMON	FERR	OUERR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 24. I2S*n* Interrupt Mask Register (I2SINTMASK) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5	XMITST		Enable stereo left/right transmit data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read.
		0	Disable stereo TX data interrupt.
		1	Enable stereo TX data interrupt.
4	XMITMON		Enable mono left transmit data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read.
		0	Disable mono TX data interrupt.
		1	Enable mono TX data interrupt.
3	RCVST		Enable stereo left/right receive data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read.
		0	Disable stereo RX data interrupt.
		1	Enable stereo RX data interrupt.
2	RCVMON		Enable mono left receive data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read.
		0	Disable mono RX data interrupt.
		1	Enable mono RX data interrupt.
1	FERR		Enable frame sync error.
		0	Disable frame-synchronization error interrupt.
		1	Enable frame-synchronization error interrupt.
0	OUERR		Enable overrun or underrun condition.
		0	Disable overrun/underrun error interrupt.
		1	Enable overrun/underrun error interrupt.

3.9 I2S*n* Receive Left Data 0 Register (I2SRXLT0)

The I2S*n* receive left data 0 register (I2SRXLT0) is shown in Figure 21 and described in Table 25.

Each I2S module has two double-word (32-bit) receive data registers to hold left and right channel data respectively. Each double-word register is accessible as two 16-bit registers by the CPU or as a single double-word register by the DMA for efficient data transfer.

Figure 21. I2S*n* Receive Left Data 0 Register (I2SRXLT0)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 25. I2S*n* Receive Left Data 0 Register (I2SRXLT0) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Receive left data lower 16 bits.

3.10 I2S*n* Receive Left Data 1 Register (I2SRXLT1)

The I2S*n* receive left data 1 register (I2SRXLT1) is shown in Figure 22 and described in Table 26.

Figure 22. I2S*n* Receive Left Data 1 Register (I2SRXLT1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 26. I2S*n* Receive Left Data 1 Register (I2SRXLT1) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Receive left data upper 16 bits.

3.11 I2S*n* Receive Right Data 0 Register (I2SRXRT0)

The I2S*n* receive right data 0 register (I2SRXRT0) is shown in Figure 23 and described in Table 27.

Figure 23. I2S*n* Receive Right Data 0 Register (I2SRXRT0)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 27. I2S*n* Receive Right Data 0 Register (I2SRXRT0) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Receive right data lower 16 bits.

3.12 I2Sn Receive Right Data 1 Register (I2SRXRT1)

The I2S*n* receive right data 1 register (I2SRXRT1) is shown in Figure 24 and described in Table 28.

Figure 24. I2S*n* Receive Right Data 1 Register (I2SRXRT1)

15	0
DATA	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after reset

Table 28. I2S*n* Receive Right Data 1 Register (I2SRXRT1) Field Descriptions

Bit	Field	Value	Description
15-0	DATA	0-FFFFh	Receive right data upper 16 bits.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265