I2S Controller (I2S)

28.1 Overview

ESP32-S3 has two built-in I2S interfaces (i.e., I2S*0* and I2S*1*), which provides a flexible communication interface for streaming digital data in multimedia applications, especially digital audio applications.

The I2S standard bus defines three signals: a bit clock signal (BCK), a channel/word select signal (WS), and a serial data signal (SD). A basic I2S data bus has one master and one slave. The roles remain unchanged throughout the communication. The I2S module on ESP32-S3 provides separate transmit (TX) and receive (RX) units for high performance.

> **Note:**
> The information provided in this chapter applies to I2S*0* and I2S*1*. Unless otherwise indicated, I2S*n* or I2S in this chapter refer to both I2S*0* and I2S*1*.

28.2 Terminology

To better illustrate the functionality of I2S*n*, the following terms are used in this chapter.

Master mode — As a master, I2S*n* outputs BCK/WS signals, and sends data to or receives data from a slave.

Slave mode — As a slave, I2S*n* inputs BCK/WS signals, and receives data from or sends data to a master.

Full-duplex — The sending line and receiving line between the master and the slave are independent. Sending data and receiving data happen at the same time.

Half-duplex — Only one side, the master or the slave, sends data first, and the other side receives data. Sending data and receiving data can not happen at the same time.

TDM RX mode — In this mode, pulse code modulated (PCM) data is received and stored into memory via DMA, in a way of time division multiplexing (TDM). The signal lines include: BCK, WS, and DATA. Data from 16 channels at most can be received. TDM Philips standard, TDM MSB alignment standard, TDM PCM standard are supported in this mode, depending on user configuration.

PDM RX mode In this mode, pulse density modulation (PDM) data is received and stored into memory via DMA. The signal lines include: WS and DATA. PDM standard is supported in this mode by user configuration.

TDM TX mode In this mode, pulse code modulated (PCM) data is sent from memory via DMA, in a way of time division multiplexing (TDM). The signal lines include: BCK, WS, and DATA. Data up to 16 channels can be sent. TDM Philips standard, TDM MSB alignment standard, TDM PCM standard are supported in this mode, depending on user configuration.

PDM TX mode In this mode, pulse density modulation (PDM) data is sent from memory via DMA. The signal lines include: WS and DATA. PDM standard is supported in this mode by user configuration.

PCM-to-PDM TX mode (for I2S*0* only) In this mode, I2S*0* as a **master**, converts the pulse code modulated (PCM) data from memory via DMA into pulse density modulation (PDM) data, and then sends the data out. The signal lines include: WS and DATA. PDM standard is supported in this mode by user configuration.

PDM-to-PCM RX mode (for I2S*0* only) In this mode, I2S*0* works as a master or a **slave**. Pulse density modulation (PDM) data is received, converted into pulse code modulated (PCM) data, and then stored into memory via DMA. The signal lines include: WS and DATA. PDM standard is supported in this mode by user configuration.

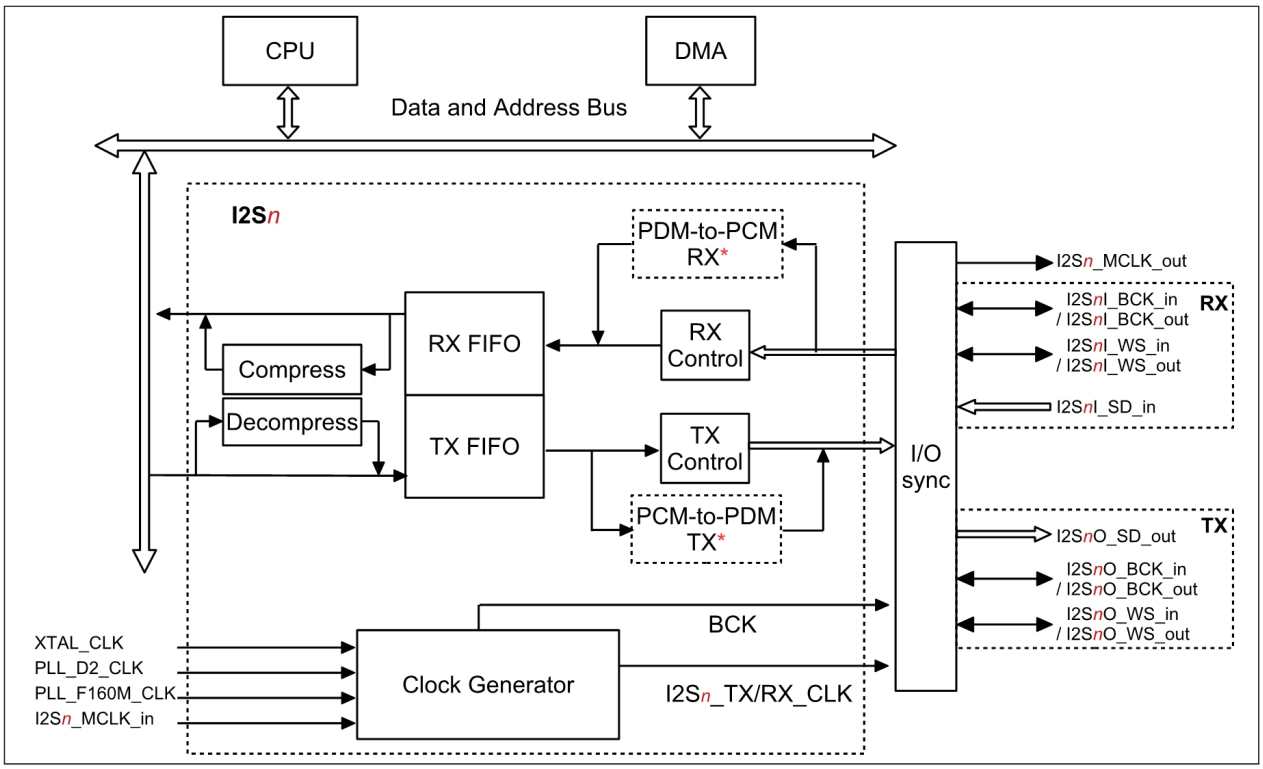
28.3 Features

I2S*n* has the following features:

- Master mode and slave mode
- Full-duplex and half-duplex communications
- Separate TX unit and RX unit, independent of each other
- TX unit and RX unit to work independently and simultaneously
- A variety of audio standards supported:
  - TDM Philips standard
  - TDM MSB alignment standard
  - TDM PCM standard
  - PDM standard
- Various TX/RX modes supported:
  - TDM TX mode
  - TDM RX mode

  - PDM TX mode
  - PDM RX mode
  - PCM-to-PDM TX mode (for I2S*0* only)
  - PDM-to-PCM RX mode (for I2S*0* only)
- Configurable high-precision sample clock
- Various frequencies supported: 8 kHz, 16 kHz, 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz, 128 kHz, and 192 kHz (192 kHz is not supported in 32-bit slave mode).
- 8-/16-/24-/32-bit data communication
- DMA access
- Standard I2S interface interrupts

28.4 System Architecture



Note: PDM-to-PCM RX and PCM-to-PDM TX are only supported by I2S*0*.

Figure 28.4-1. ESP32-S3 I2S System Diagram

Figure 28.4-1 shows the structure of ESP32-S3 I2S*n* module, consisting of:

- TX unit (TX control)
- RX unit (RX control)
- Input and Output Timing unit (I/O sync)
- Clock Divider (Clock Generator)
- 64 x 32-bit TX FIFO

- 64 x 32-bit RX FIFO
- Compress/Decompress units

ESP32-S3 I2S*n* module supports direct access (GDMA) to internal memory and external memory, see Chapter 3 *GDMA Controller (GDMA)*.

Both the TX unit and the RX unit have a three-line interface that includes a bit clock line (BCK), a word select line (WS), and a serial data line (SD). The SD line of the TX unit is fixed as output, and the SD line of the RX unit as input. BCK and WS signal lines for TX unit and RX unit can be configured as master output mode or slave input mode.

The signal bus of I2S*n* module is shown at the right part of Figure 28.4-1. The naming of these signals in RX and TX units follows the pattern: I2S*n*A_B_C, for example, I2S*n*I_BCK_in.

- "A": direction of data bus
  - "I": input, receiving
  - "O": output, transmitting
- "B": signal function
  - bit clock signal (BCK)
  - word select signal (WS)
  - serial data signal (SD)
- "C": signal direction
  - "in": input signal into I2S*n* module
  - "out": output signal from I2S*n* module

Table 28.4-1 provides a detailed description of I2S*n* signals.

Table 28.4-1. I2S*n* Signal Description

Signal*	Direction	Function
I2S*n*I_BCK_in	Input	In slave mode, inputs BCK signal for RX unit.
I2S*n*I_BCK_out	Output	In master mode, outputs BCK signal for RX unit.
I2S*n*I_WS_in	Input	In slave mode, inputs WS signal for RX unit.
I2S*n*I_WS_out	Output	In master mode, outputs WS signal for RX unit.
I2S*n*I_Data_in	Input	Works as the serial input data bus for RX unit.
I2S*n*O_Data_out	Output	Works as the serial output data bus for TX unit.
I2S*n*O_BCK_in	Input	In slave mode, inputs BCK signal for TX unit.
I2S*n*O_BCK_out	Output	In master mode, outputs BCK signal for TX unit.
I2S*n*O_WS_in	Input	In slave mode, inputs WS signal for TX unit.
I2S*n*O_WS_out	Output	In master mode, outputs WS signal for TX unit.
I2S*n*_MCLK_in	Input	In slave mode, works as a clock source from the external master.
I2S*n*_MCLK_out	Output	In master mode, works as a clock source for the external slave.
I2S*0*I_Data1_in	Input	In PDM-to-PCM RX mode, works as the serial input data line for RX unit.
I2S*0*I_Data2_in	Input	In PDM-to-PCM RX mode, works as the serial input data line for RX unit.
I2S*0*I_Data3_in	Input	In PDM-to-PCM RX mode, works as the serial input data line for RX unit.
I2S*0*O_Data1_out	Output	In PCM-to-PDM TX mode, works as the serial output data line for TX unit.

* Any required signals of I2S*n* must be mapped to the chip's pins via GPIO matrix, see Chapter 6 *IO MUX and GPIO Matrix (GPIO, IO MUX)*.

28.5 Supported Audio Standards

ESP32-S3 I2S*n* supports multiple audio standards, including TDM Philips standard, TDM MSB alignment standard, TDM PCM standard, and PDM standard.

Select the standard by configuring the following bits:

- I2S_TX/RX_TDM_EN
  - 0: disable TDM mode.
  - 1: enable TDM mode.
- I2S_TX/RX_PDM_EN
  - 0: disable PDM mode.
  - 1: enable PDM mode.
- I2S_TX/RX_MSB_SHIFT
  - 0: WS and SD signals change simultaneously, i.e., enable MSB alignment standard.
  - 1: WS signal changes one BCK clock cycle earlier than SD signal, i.e., enable Philips standard or select PCM standard.
- I2S_TX/RX_PCM_BYPASS
  - 0: enable PCM standard.
  - 1: disable PCM standard.

28.5.1 TDM Philips Standard

Philips specifications require that WS signal changes one BCK clock cycle earlier than SD signal on BCK falling edge, which means that WS signal is valid from one clock cycle before transmitting the first bit of channel data and changes one clock before the end of channel data transfer. SD signal line transmits the most significant bit of audio data first.

Compared with Philips standard, TDM Philips standard supports multiple channels, see Figure 28.5-1.

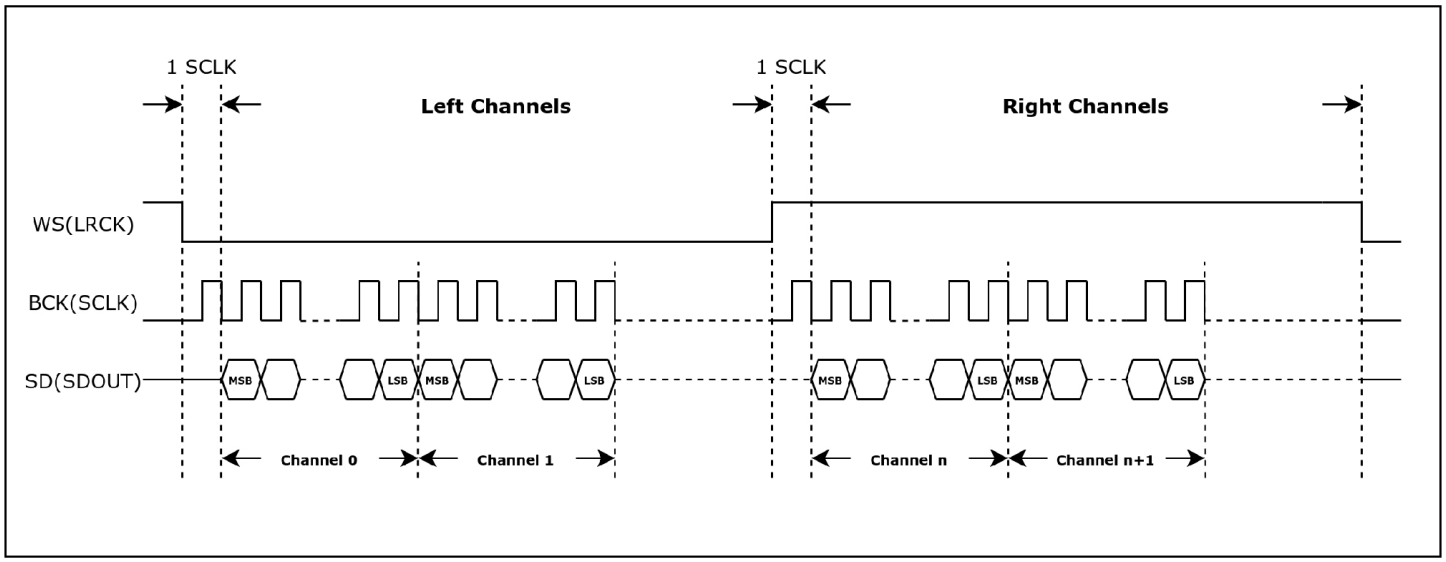


Figure 28.5-1. TDM Philips Standard Timing Diagram

28.5.2 TDM MSB Alignment Standard

MSB alignment specifications require WS and SD signals change simultaneously on the falling edge of BCK. The WS signal is valid until the end of channel data transfer. The SD signal line transmits the most significant bit of audio data first.

Compared with MSB alignment standard, TDM MSB alignment standard supports multiple channels, see Figure 28.5-2.

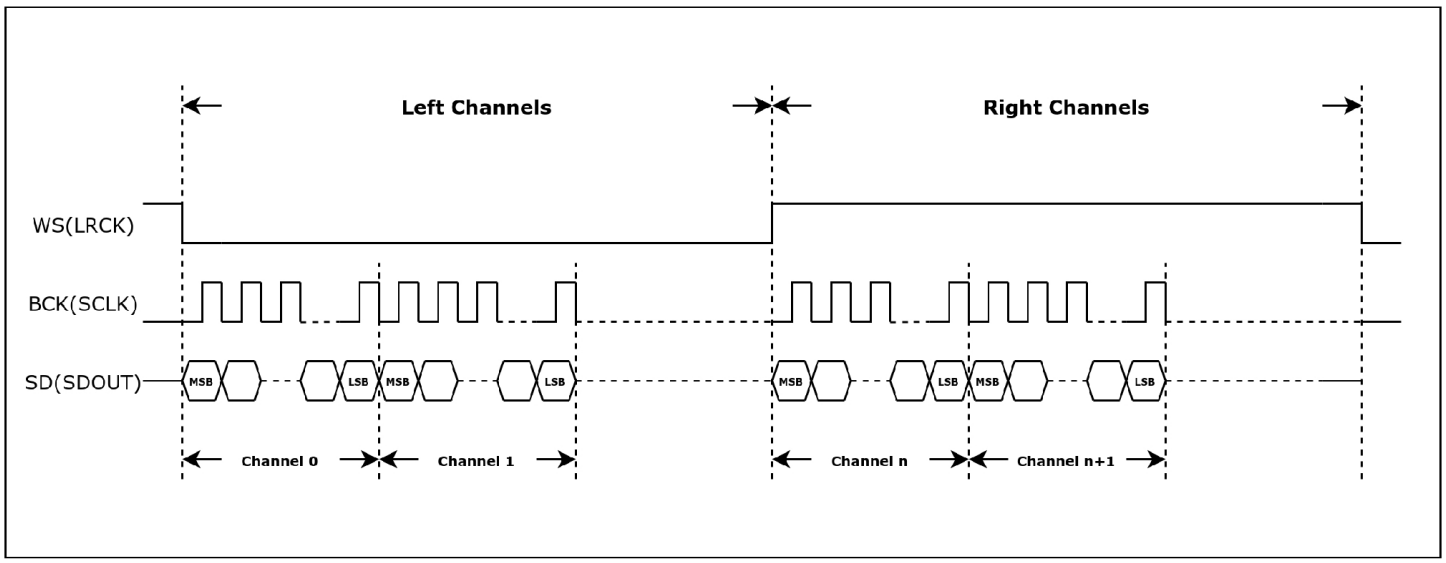


Figure 28.5-2. TDM MSB Alignment Standard Timing Diagram

28.5.3 TDM PCM Standard

Short frame synchronization under PCM standard requires WS signal changes one BCK clock cycle earlier than SD signal on the falling edge of BCK, which means that the WS signal becomes valid one clock cycle before transferring the first bit of channel data and remains unchanged in this BCK clock cycle. SD signal line transmits the most significant bit of audio data first.

Compared with PCM standard, TDM PCM standard supports multiple channels, see Figure 28.5-3.

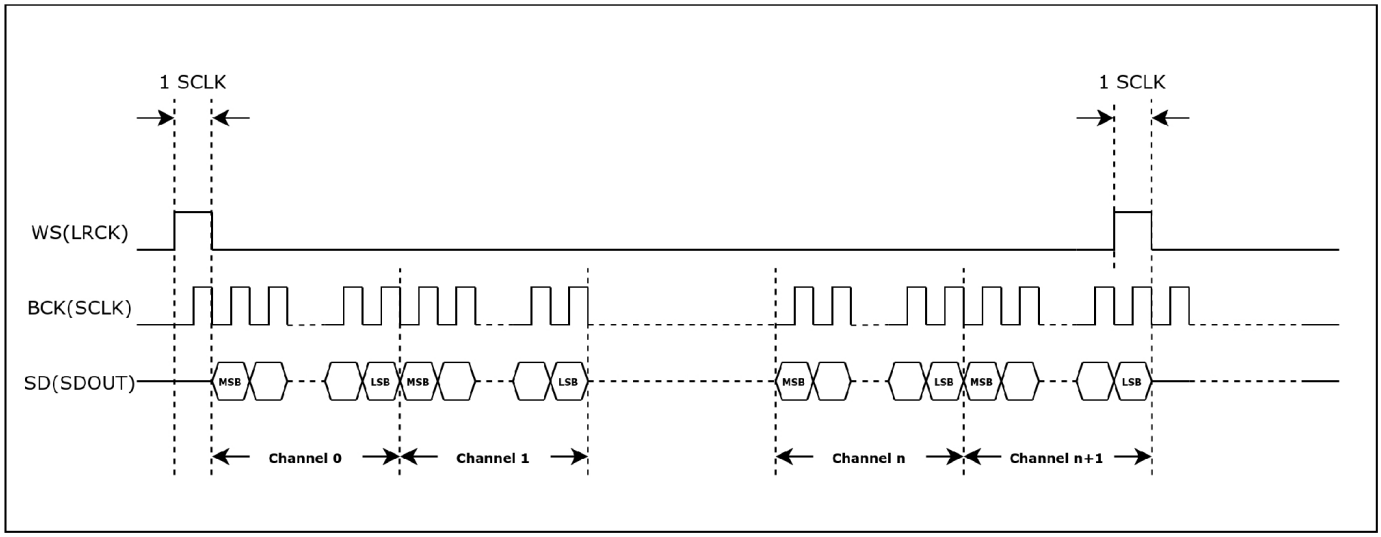


Figure 28.5-3. TDM PCM Standard Timing Diagram

28.5.4 PDM Standard

Under PDM standard, WS signal changes continuously during data transmission. The low-level and high-level of this signal indicates the left channel and right channel, respectively. WS and SD signals change simultaneously on the falling edge of BCK, see Figure 28.5-4.

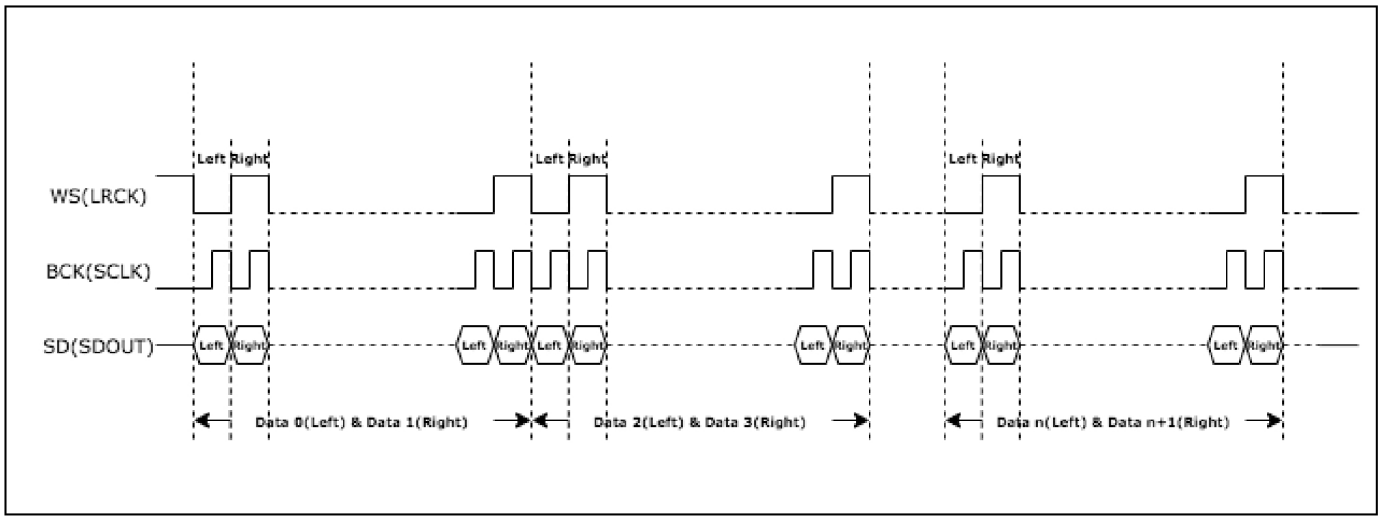


Figure 28.5-4. PDM Standard Timing Diagram

28.6 TX/RX Clock

I2S*n*_TX/RX_CLK as shown in Figure 28.6-1 is the master clock of I2S*n* TX/RX unit, divided from:

- 40 MHz XTAL_CLK
- 160 MHz PLL_F160M_CLK
- 240 MHz PLL_D2_CLK
- or external input clock: I2S*n*_MCLK_in

I2S_TX/RX_CLK_SEL is used to select clock source for TX/RX unit, and I2S_TX/RX_CLK_ACTIVE to enable or disable the clock source.

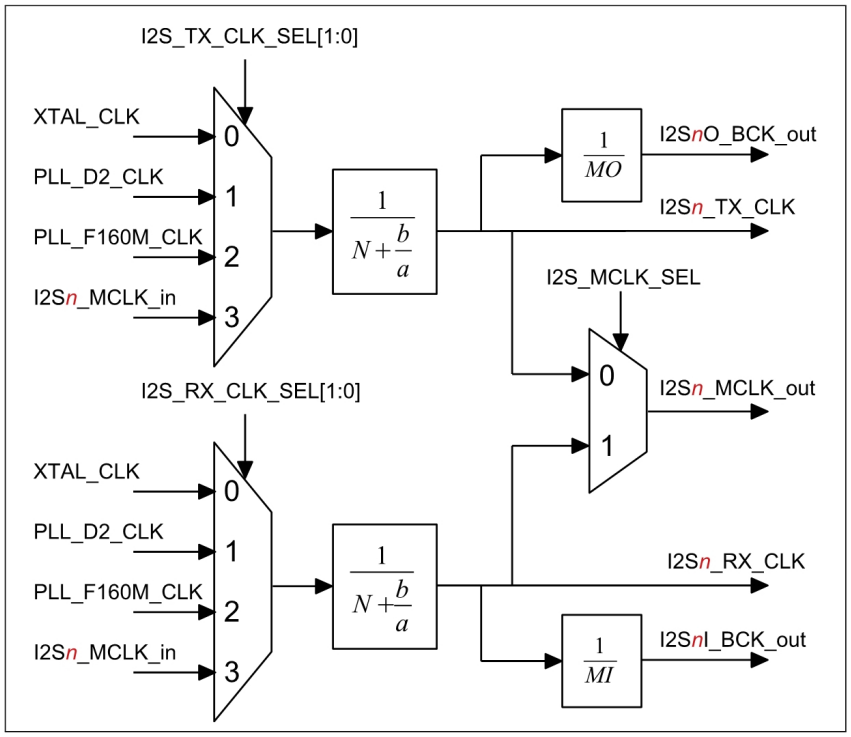


Figure 28.6-1. I2S*n* Clock

The following formula shows the relation between I2S*n*_TX/RX_CLK frequency (f_{I2Sn_TX/RX_CLK}) and the divider clock source frequency ($f_{I2Sn\_CLK\_S}$):

$$f_{I2Sn\_TX/RX\_CLK} = \frac{f_{I2Sn\_CLK\_S}}{N + \frac{b}{a}}$$

N is an integer value between 2 and 256. The value of N corresponds to the value of I2S_TX/RX_CLKM_DIV_NUM in register I2S_TX/RX_CLKM_CONF_REG as follows:

- When I2S_TX/RX_CLKM_DIV_NUM = 0, N = 256.
- When I2S_TX/RX_CLKM_DIV_NUM = 1, N = 2.
- When I2S_TX/RX_CLKM_DIV_NUM has any other value, N = I2S_TX/RX_CLKM_DIV_NUM.

The values of "a" and "b" in fractional divider depend only on x, y, z, and yn1. The corresponding formulas are as follows:

- When b $<= \frac{a}{2}$, yn1 $= 0$, x $= floor([\frac{a}{b}]) - 1$, y = a%b, z = b;
- When b $> \frac{a}{2}$, yn1 $= 1$, x $= floor([\frac{a}{a-b}]) - 1$, y = a%(a - b), z = a - b.

The values of x, y, z, and yn1 are configured in I2S_TX/RX_CLKM_DIV_X, I2S_TX/RX_CLKM_DIV_Y, I2S_TX/RX_CLK
M_DIV_Z, and I2S_TX/RXCLKM_DIV_YN1.

To configure the integer divider, clear I2S_TX/RX_CLKM_DIV_X and I2S_TX/RX_CLKM_DIV_Z, then set I2S_TX/RX_C
LKM_DIV_Y to 1.

> **Note:**
> Using fractional divider may introduce some clock jitter.

The serial clock (BCK) of the I2S*n* TX/RX unit is divided from I2S*n*_TX/RX_CLK, as shown in Figure 28.6-1.

In master TX mode, the serial clock BCK for I2S*n* TX unit is I2S*n*O_BCK_out, divided from I2S*n*_TX_CLK. That is:

$$f_{\text{I2S}n\text{O\_BCK\_out}} = \frac{f_{\text{I2S}n\text{\_TX\_CLK}}}{\text{MO}}$$

"MO" is an integer value:

$$\text{MO} = \text{I2S\_TX\_BCK\_DIV\_NUM} + 1$$

Note:

I2S_TX_BCK_DIV_NUM must not be configured as 1.

In master RX mode, the serial clock BCK for I2S*n* RX unit is I2S*n*I_BCK_out, divided from I2S*n*_RX_CLK. That is:

$$f_{\text{I2S}n\text{I\_BCK\_out}} = \frac{f_{\text{I2S}n\text{\_RX\_CLK}}}{\text{MI}}$$

"MI" is an integer value:

$$\text{MI} = \text{I2S\_RX\_BCK\_DIV\_NUM} + 1$$

Note:

- I2S_RX_BCK_DIV_NUM must not be configured as 1.
- In slave mode, make sure $f_{\text{I2S}n\text{\_TX/RX\_CLK}}$ >= 8 * f_{BCK}. I2S*n* module can output I2S*n*_MCLK_out as the master clock for peripherals.

28.7 I2S*n* Reset

The units and FIFOs in I2S*n* module are reset by the following bits.

- I2S*n* TX/RX units: reset by the bits I2S_TX_RESET and I2S_RX_RESET.
- I2S*n* TX/RX FIFO: reset by the bits I2S_TX_FIFO_RESET and I2S_RX_FIFO_RESET.

Note: I2S*n* module clock must be configured first before the module and FIFO are reset.

28.8 I2S*n* Master/Slave Mode

The ESP32-S3 I2S*n* module can operate as a master or a slave, depending on the configuration of I2S_TX_SLAVE_MOD and I2S_RX_SLAVE_MOD.

- I2S_TX_SLAVE_MOD
  - 0: master TX mode
  - 1: slave TX mode
- I2S_RX_SLAVE_MOD
  - 0: master RX mode
  - 1: slave RX mode

28.8.1 Master/Slave TX Mode

- I2S*n* works as a master transmitter:
  - Set the bit I2S_TX_START to start transmitting data.
  - TX unit keeps driving the clock signal and serial data.
  - If I2S_TX_STOP_EN is set and all the data in FIFO is transmitted, the master stops transmitting data.
  - If I2S_TX_STOP_EN is cleared and all the data in FIFO is transmitted, meanwhile no new data is filled into FIFO, then the TX unit keeps sending the last data frame.
  - Master stops sending data when the bit I2S_TX_START is cleared.
- I2S*n* works as a slave transmitter:
  - Set the bit I2S_TX_START.
  - Wait for the master BCK clock to enable a transmit operation.
  - If I2S_TX_STOP_EN is set and all the data in FIFO is transmitted, then the slave keeps sending zeros, till the master stops providing BCK signal.
  - If I2S_TX_STOP_EN is cleared and all the data in FIFO is transmitted, meanwhile no new data is filled into FIFO, then the TX unit keeps sending the last data frame.
  - If I2S_TX_START is cleared, slave keeps sending zeros till the master stops providing BCK clock signal.

28.8.2 Master/Slave RX Mode

- I2S*n* works as a master receiver:
  - Set the bit I2S_RX_START to start receiving data.
  - RX unit keeps outputting clock signal and sampling input data.
  - RX unit stops receiving data when the bit I2S_RX_START is cleared.
- I2S*n* works as a slave receiver:
  - Set the bit I2S_RX_START.
  - Wait for master BCK signal to start receiving data.
  - RX unit stops receiving data when the bit I2S_RX_START is cleared.

28.9 Transmitting Data

Note:
Updating the configuration described in this and subsequent sections requires to set I2S_TX_UPDATE accordingly, to synchronize I2S*n* TX registers from APB clock domain to TX clock domain. For more detailed configuration, see Section 28.11.1.

In TX mode, I2S*n* first reads data from DMA and sends these data out via output signals according to the configured data mode and channel mode.

28.9.1 Data Format Control

Data format is controlled in the following phases:

- Phase I: read data from memory and write it to TX FIFO.
- Phase II: read the data to send (TX data) from TX FIFO and convert the data according to output data mode.
- Phase III: clock out the TX data serially.

28.9.1.1 Bit Width Control of Channel Valid Data

The bit width of valid data in each channel is determined by I2S_TX_BITS_MOD and I2S_TX_24_FILL_EN, see the table below.

Table 28.9-1. Bit Width of Channel Valid Data

Channel Valid Data Width	I2S_TX_BITS_MOD	I2S_TX_24_FILL_EN
32	31	x[1]
	23	1
24	23	0
16	15	x
8	7	x

[1] x: This value is ignored.

28.9.1.2 Endian Control of Channel Valid Data

When I2S*n* reads data from DMA, the data endian under various data width is controlled by I2S_TX_BIG_ENDIAN, see the table below.

Table 28.9-2. Endian of Channel Valid Data

Channel Valid Data Width	Origin Data	Endian of Processed Data	I2S_TX_BIG_ENDIAN
32	{B3, B2, B1, B0}	{B3, B2, B1, B0}	0
		{B0, B1, B2, B3}	1
24	{B2, B1, B0}	{B2, B1, B0}	0
		{B0, B1, B2}	1
16	{B1, B0}	{B1, B0}	0
		{B0, B1}	1
8	{B0}	{B0}	x

28.9.1.3 A-law/μ-law Compression and Decompression

ESP32-S3 I2S*n* compresses/decompresses the valid data into 32-bit by A-law or by μ-law. If the bit width of valid data is smaller than 32, zeros are filled to the extra high bits of the data to be compressed/decompressed by default.

> **Note:**
> Extra high bits here mean the bits[31: channel valid data width] of the data to be compressed/decompressed.

Configure I2S_TX_PCM_BYPASS to:

- 0: Compress or decompress the data.
- 1: Do not compress or decompress the data.

Configure I2S_TX_PCM_CONF to:

- 0: Decompress the data using A-law.
- 1: Compress the data using A-law.
- 2: Decompress the data using μ-law.
- 3: Compress the data using μ-law.

At this point, the first phase of data format control is complete.

28.9.1.4 Bit Width Control of Channel TX Data

The TX data width in each channel is determined by I2S_TX_TDM_CHAN_BITS.

- If TX data width in each channel is larger than the valid data width, zeros will be filled to these extra bits. Configure I2S_TX_LEFT_ALIGN to:
  - 0: The valid data is at the lower bits of TX data.
  - 1: The valid data is at the higher bits of TX data.
- If the TX data width in each channel is smaller than the valid data width, only the lower bits of valid data are sent out, and the higher bits are discarded.

At this point, the second phase of data format control is complete.

28.9.1.5 Bit Order Control of Channel Data

The channel data will be stored as the data to be input in order from high to low. The data bit order in each channel is controlled by I2S_TX_BIT_ORDER:

- 0: Not reverse the valid data bit order;
- 1: Reverse the valid data bit order.

At this point, the data format control is complete. The data after format control will be sent sequentially from high to low. Figure 28.9-1 shows a complete process of TX data format control.

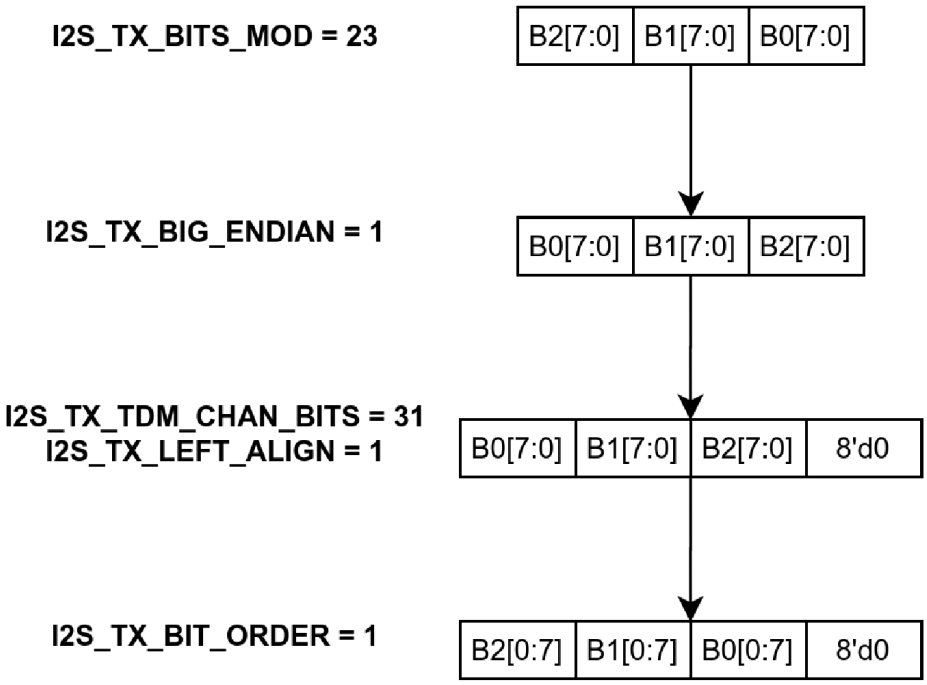


Figure 28.9-1. TX Data Format Control

28.9.2 Channel Mode Control

ESP32-S3 I2S*n* supports both TDM TX mode and PDM TX mode. Set I2S_TX_TDM_EN to enable TDM TX mode, or set I2S_TX_PDM_EN to enable PDM TX mode.

Note:

- I2S_TX_TDM_EN and I2S_TX_PDM_EN must not be cleared or set simultaneously.
- Most stereo I2S codecs can be controlled by setting the I2S*n* module into 2-channel mode under TDM standard.

28.9.2.1 I2S*n* Channel Control in TDM Mode

In TDM mode, I2S*n* supports up to 16 channels to output data. The total number of TX channels in use is controlled by I2S_TX_TDM_TOT_CHAN_NUM. For example, if I2S_TX_TDM_TOT_CHAN_NUM is set to 5, six channels in total (channel 0 ~ 5) will be used to transmit data, see Figure 28.9-2.

In these TX channels, if I2S_TX_TDM_CHANn_EN is set to:

- 1: This channel sends the channel data out.
- 0: The TX data to be sent by this channel is controlled by I2S_TX_CHAN_EQUAL:
  - 1: The data of previous channel is sent out.
  - 0: The data stored in I2S_SINGLE_DATA is sent out.

In TDM master mode, WS signal is controlled by I2S_TX_WS_IDLE_POL and I2S_TX_TDM_WS_WIDTH:

- I2S_TX_WS_IDLE_POL: the default level of WS signal
- I2S_TX_TDM_WS_WIDTH: the cycles the WS default level lasts for when transmitting all channel data

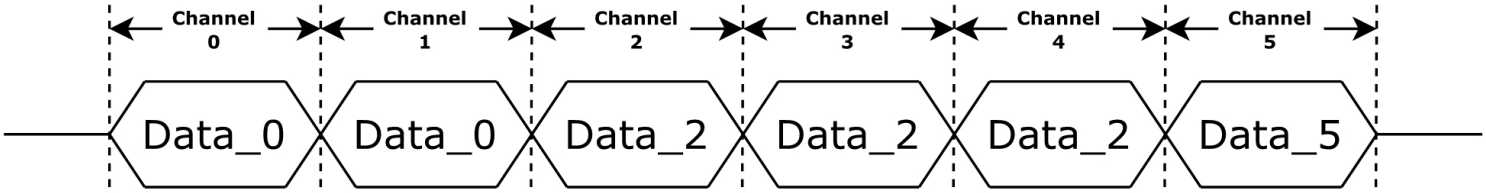
I2S_TX_HALF_SAMPLE_BITS x 2 is equal to the BCK cycles in one WS period.

TDM Channel Configuration Example

In this example, the register configuration is as follows.

- I2S_TX_TDM_TOT_CHAN_NUM = 5, i.e., channel 0 ~ 5 are used to transmit data.
- I2S_TX_CHAN_EQUAL = 1, i.e., that data of previous channel will be transmitted if the bit I2S_TX_TDM_CHANn_EN is cleared. *n* = 0 ~ 5.
- I2S_TX_TDM_CHAN0/2/5_EN = 1, i.e., these channels send their channel data out.
- I2S_TX_TDM_CHAN1/3/4_EN = 0, i.e., these channels send the previous channel data out.

Once the configuration is done, data is transmitted as follows.



I2S_TX_TDM_CHAN_NUM = 5; I2S_TX_CHAN_EQUAL = 1;
I2S_TX_TDM_CHAN0_EN = 1; I2S_TX_TDM_CHAN1_EN = 0; I2S_TX_TDM_CHAN2_EN = 1;
I2S_TX_TDM_CHAN3_EN = 0; I2S_TX_TDM_CHAN4_EN = 0; I2S_TX_TDM_CHAN5_EN = 1;

Figure 28.9-2. TDM Channel Control

28.9.2.2 I2S*n* Channel Control in PDM Mode

In PDM mode, fetching data from DMA is controlled by I2S_TX_MONO and I2S_TX_MONO_FST_VLD, see the table below. Please configure the two bits according to the data stored in memory, be it the single-channel or dual-channel data.

Table 28.9-3. Data-Fetching Control in PDM Mode

Data-Fetching Control Option	Mode	I2S_TX_MONO	I2S_TX_MONO_FST_VLD
Post data-fetching request to DMA at any edge of WS signal	Stereo mode	0	x
Post data-fetching request to DMA only at the second half period of WS signal	Mono mode	1	0
Post data-fetching request to DMA only at the first half period of WS signal	Mono mode	1	1

In PDM mode, I2S*n* channel mode is controlled by I2S_TX_CHAN_MOD and I2S_TX_WS_IDLE_POL, see the table below.

Table 28.9-4. I2S*n* Channel Control in PDM Mode

Channel Control Option	Left Channel	Right Channel	Mode Control Field[1]	Channel Select Bit[2]
Stereo mode	Transmit the left channel data	Transmit the right channel data	0	x
Mono mode	Transmit the left channel data	Transmit the left channel data	1	0
	Transmit the right channel data	Transmit the right channel data	1	1
	Transmit the right channel data	Transmit the right channel data	2	0
	Transmit the left channel data	Transmit the left channel data	2	1
	Transmit the value of I2S_SINGLE_DATA	Transmit the right channel data	3	0
	Transmit the left channel data	Transmit the value of I2S_SINGLE_DATA	3	1
	Transmit the left channel data	Transmit the value of I2S_SINGLE_DATA	4	0
	Transmit the value of I2S_SINGLE_DATA	Transmit the right channel data	4	1

[1] I2S_TX_CHAN_MOD
[2] I2S_TX_WS_IDLE_POL

In PDM master mode, the WS level of I2S*n* module is controlled by I2S_TX_WS_IDLE_POL. The frequency of WS signal is half of BCK frequency. The configuration of WS signal is similar to that of BCK signal, see Section 28.6 and Figure 28.9-3.

ESP32-S3 I2S*0* also supports PCM-to-PDM output mode, in which the PCM data from DMA is converted to PDM data and then output in PDM signal format. Configure I2S_PCM2PDM_CONV_EN to enable this mode.

The register configuration for PCM-to-PDM output mode is as follows:

- Configure 1-line PDM output format or 1-/2-line DAC output mode as the table below:

Table 28.9-5. PCM-to-PDM Output Mode

Channel Output Format	I2S0_TX_PDM_DAC_MODE_EN	I2S0_TX_PDM_DAC_2OUT_EN
1-line PDM output format[1]	0	x
1-line DAC output format[2]	1	0
2-line DAC output format	1	1

[1] In PDM output format, SD data of two channels is sent out in one WS period.
[2] In DAC output format, SD data of one channel is sent out in one WS period.

- Configure sampling frequency and upsampling rate
  In PCM-to-PDM mode, PDM clock frequency is equal to BCK frequency. The relation of sampling frequency ($f_{Sampling}$) and BCK frequency is as follows:

$$f_{Sampling} = \frac{f_{BCK}}{OSR}$$

  Upsampling rate (OSR) is related to I2S0_TX_PDM_SINC_OSR2 as follows:

$$OSR = I2S0\_TX\_PDM\_SINC\_OSR2 \times 64$$

  Sampling frequency $f_{Sampling}$ is related to I2S_TX_PDM_FS as follows:

$$f_{Sampling} = I2S0\_TX\_PDM\_FS \times 100$$

  Configure the registers according to needed sampling frequency, upsampling rate, and PDM clock frequency.

PDM Channel Configuration Example

In this example, the register configuration is as follows.

- I2S_TX_CHAN_MOD = 2, i.e., mono mode is selected.
- I2S_TX_WS_IDLE_POL = 1, i.e., both the left channel and right channel transmit the left channel data.

Once the configuration is done, the channel data is transmitted as follows.

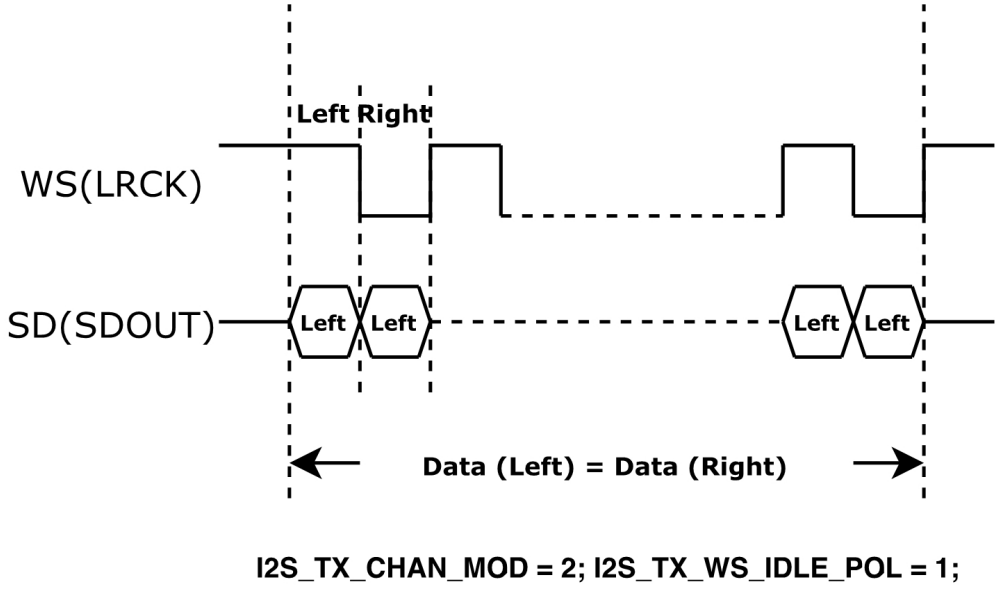


Figure 28.9-3. PDM Channel Control

28.10 Receiving Data

> **Note:**
> Updating the configuration described in this and subsequent sections requires to set I2S_RX_UPDATE accordingly, to synchronize I2S*n* RX registers from APB clock domain to RX clock domain. For more detailed configuration, see Section 28.11.2.

In RX mode, I2S*n* first reads data from peripheral interface, and then stores the data into memory via DMA, according to the configured channel mode and data mode.

28.10.1 Channel Mode Control

ESP32-S3 I2S*n* supports both TDM RX mode and PDM RX mode. Set I2S_RX_TDM_EN to enable TDM RX mode, or set I2S_RX_PDM_EN to enable PDM RX mode.

Note: I2S_RX_TDM_EN and I2S_RX_PDM_EN must not be cleared or set simultaneously.

28.10.1.1 I2S*n* Channel Control in TDM Mode

In TDM mode, I2S*n* supports up to 16 channels to input data. The total number of RX channels in use is controlled by I2S_RX_TDM_TOT_CHAN_NUM. For example, if I2S_RX_TDM_TOT_CHAN_NUM is set to 5, channel 0 ~ 5 will be used to receive data.

In these RX channels, if I2S_RX_TDM_CHANn_EN is set to:

- 1: This channel data is valid and will be stored into RX FIFO.
- 0: This channel data is invalid and will not be stored into RX FIFO.

In TDM master mode, WS signal is controlled by I2S_RX_WS_IDLE_POL and I2S_RX_TDM_WS_WIDTH:

- I2S_RX_WS_IDLE_POL: the default level of WS signal
- I2S_RX_TDM_WS_WIDTH: the cycles the WS default level lasts for when receiving all channel data

I2S_RX_HALF_SAMPLE_BITS x 2 is equal to the BCK cycles in one WS period.

28.10.1.2 I2S*n* Channel Control in PDM Mode

In PDM mode, I2S*n* converts the serial data from channels to the data to be entered into memory.

In PDM master mode, the WS level of I2S*n* module is controlled by I2S_RX_WS_IDLE_POL. WS frequency is half of BCK frequency. The configuration of BCK signal is similar to that of WS signal as described in Section 28.6. Note, in PDM RX mode, the value of I2S_RX_HALF_SAMPLE_BITS must be same as that of I2S_RX_BITS_MOD.

I2S*0* supports PDM-to-PCM input mode, in which the received PDM data is converted to PCM data and controlled according to the data mode. Configure I2S_RX_PDM2PCM_EN to enable this mode.

The register configuration for PDM-to-PCM input mode is as follows:

- Configure sampling frequency and downsampling rate.
  In PDM-to-PCM input mode, PDM clock frequency is:
  - in master mode: PDM clock frequency is equal to BCK frequency.
  - in slave mode: PDM clock is provided by external device.

  The sampling frequency ($f_{Sampling}$) is related to PDM clock frequency as follows:

$$f_{Sampling} = \frac{f_{PDM}}{DSR}$$

  Downsampling rate (DSR) is related to I2S*0*_RX_PDM_SINC_DSR_16_EN as follows:

$$DSR = I2S0\_RX\_PDM\_SINC\_DSR\_16\_EN \times 64$$

  Configure the registers according to needed master/slave mode, sampling frequency, and downsampling rate.
- Configure valid channels.
  In PDM-to-PCM mode, input signals from eight channels are supported at most. See Table 28.10-1 for the register configuration and related channels.

Table 28.10-1. PDM-to-PCM Input Mode

Input Data Signal	Channel	Enable Register
I2S*0*I_Data_in	Left channel	I2S*0*_RX_TDM_PDM_CHAN0_EN
	Right channel	I2S*0*_RX_TDM_PDM_CHAN1_EN
I2S*0*I1_Data_in	Left channel	I2S*0*_RX_TDM_PDM_CHAN2_EN
	Right channel	I2S*0*_RX_TDM_PDM_CHAN3_EN
I2S*0*I2_Data_in	Left channel	I2S*0*_RX_TDM_PDM_CHAN4_EN
	Right channel	I2S*0*_RX_TDM_PDM_CHAN5_EN
I2S*0*I3_Data_in	Left channel	I2S*0*_RX_TDM_PDM_CHAN6_EN
	Right channel	I2S*0*_RX_TDM_PDM_CHAN7_EN

28.10.2 Data Format Control

Data format is controlled in the following phases:

- Phase I: serial input data is converted into the data to be saved to RX FIFO.
- Phase II: the data is read from RX FIFO and converted according to input data mode.

28.10.2.1 Bit Order Control of Channel Data

The channel data will be stored as the data to be input in order from high to low. The data bit order in each channel is controlled by I2S_RX_BIT_ORDER:

- 0: The bit order of the data to be input is not reversed;
- 1: The bit order of the data to be input is reversed.

At this point, the first phase of data format control is complete. The data to be input after bit order control is stored in the RX FIFO.

28.10.2.2 Bit Width Control of Channel Storage (Valid) Data

The storage data width in each channel is controlled by I2S_RX_BITS_MOD and I2S_RX_24_FILL_EN, see the table below.

Table 28.10-2. Channel Storage Data Width

Channel Storage Data Width	I2S_RX_BITS_MOD	I2S_RX_24_FILL_EN
32	31	x
	23	1
24	23	0
16	15	x
8	7	x

28.10.2.3 Bit Width Control of Channel RX Data

The RX data width in each channel is determined by I2S_RX_TDM_CHAN_BITS.

- If the storage data width in each channel is smaller than the received (RX) data width, then only the bits within the storage data width is saved into memory. Configure I2S_RX_LEFT_ALIGN to:
  - 0: Only the lower bits of the received data within the storage data width is stored to memory.
  - 1: Only the higher bits of the received data within the storage data width is stored to memory.
- If the received data width is smaller than the storage data width in each channel, the higher bits of the received data will be filled with zeros and then the data is saved to memory.

28.10.2.4 Endian Control of Channel Storage Data

The received data is then converted into storage data (to be stored to memory) after some processing, such as dis- carding extra bits or filling zeros in missing bits. The endian of the storage data is controlled by I2S_RX_BIG_ENDIAN under various data width, see the table below.

Table 28.10-3. Channel Storage Data Endian

Channel Storage Data Width	Origin Data	Endian of Processed Data	I2S_RX_BIG_ENDIAN
32	{B3, B2, B1, B0}	{B3, B2, B1, B0}	0
		{B0, B1, B2, B3}	1
24	{B2, B1, B0}	{B2, B1, B0}	0
		{B0, B1, B2}	1
16	{B1, B0}	{B1, B0}	0
		{B0, B1}	1
8	{B0}	{B0}	x

28.10.2.5 A-law/μ-law Compression and Decompression

ESP32-S3 I2S*n* compresses/decompresses the data to be stored in 32-bit by A-law or by μ-law. By default, zeros are filled to high bits.

Configure I2S_RX_PCM_BYPASS to:

- 0: Compress or decompress the data.
- 1: Do not compress or decompress the data.

Configure I2S_RX_PCM_CONF to:

- 0: Decompress the data using A-law.
- 1: Compress the data using A-law.
- 2: Decompress the data using μ-law.
- 3: Compress the data using μ-law.

At this point, the data format control is complete. Data then is stored into memory via DMA.

28.11 Software Configuration Process

28.11.1 Configure I2S*n* as TX Mode

Follow the steps below to configure I2S*n* as TX mode via software:

1. Configure the clock as described in Section 28.6.
2. Configure signal pins according to Table 28.4-1.
3. Select the mode needed by configuring the bit I2S_TX_SLAVE_MOD.
    - 0: master TX mode
    - 1: slave TX mode
4. Set needed TX data mode and TX channel mode as described in Section 28.9, and then set the bit I2S_TX_UPDATE.
5. Reset TX unit and TX FIFO as described in Section 28.7.

6. Enable corresponding interrupts, see Section 28.12.
7. Configure DMA outlink.
8. Set I2S_TX_STOP_EN if needed. For more information, please refer to Section 28.8.1.
9. Start transmitting data:
    - In master mode, wait till I2S*n* slave gets ready, then set I2S_TX_START to start transmitting data.
    - In slave mode, set the bit I2S_TX_START. When the I2S*n* master supplies BCK and WS signals, I2S*n* slave starts transmitting data.
10. Wait for the interrupt signals set in Step 6, or check whether the transfer is completed by querying I2S_TX_IDLE:
    - 0: transmitter is working.
    - 1: transmitter is in idle.
11. Clear I2S_TX_START to stop data transfer.

28.11.2 Configure I2S*n* as RX Mode

Follow the steps below to configure I2S*n* as RX mode via software:

1. Configure the clock as described in Section 28.6.
2. Configure signal pins according to Table 28.4-1.
3. Select the mode needed by configuring the bit I2S_RX_SLAVE_MOD.
    - 0: master RX mode
    - 1: slave RX mode
4. Set needed RX data mode and RX channel mode as described in Section 28.10, and then set the bit I2S_RX_UPDATE.
5. Reset RX unit and its FIFO according to Section 28.7.
6. Enable corresponding interrupts, see Section 28.12.
7. Configure DMA inlink, and set the length of RX data in I2S_RXEOF_NUM_REG.
8. Start receiving data:
    - In master mode, when the slave is ready, set I2S_RX_START to start receiving data.
    - In slave mode, set I2S_RX_START to start receiving data when get BCK and WS signals from the master.
9. The received data is then stored to the specified address of ESP32-S3 memory according the configuration of DMA. Then the corresponding interrupt set in Step 6 is generated.

28.12 I2S*n* Interrupts

- I2S_TX_HUNG_INT: triggered when transmitting data is timed out. For example, if I2S*n* module is configured as TX slave mode, but the master does not provide BCK or WS signal for time specified in

I2S_LC_HUNG_CONF_REG, then this interrupt will be triggered.

- I2S_RX_HUNG_INT: triggered when receiving data is timed out. For example, if I2S*n* module is configured as RX slave mode, but the master does not send data for time specified in I2S_LC_HUNG_CONF_REG, then this interrupt will be triggered.
- I2S_TX_DONE_INT: triggered when transmitting data is completed.
- I2S_RX_DONE_INT: triggered when receiving data is completed.

28.13 Register Summary

The addresses in this section are relative to [I2S*n*] base address provided in Table 4.3-3 in Chapter 4 *System and Memory*.

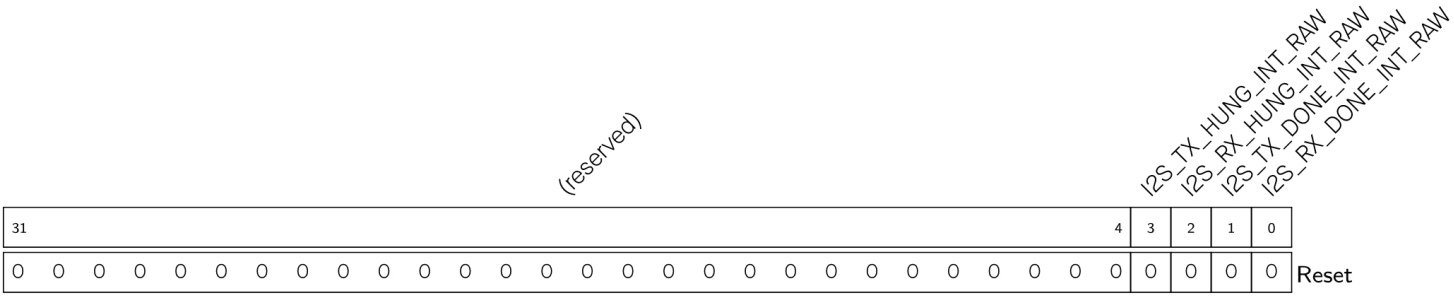
The abbreviations given in Column **Access** are explained in Section *Access Types for Registers*.

Name	Description	I2S0 Address	I2S1 Address	Access
Interrupt registers				
I2S_INT_RAW_REG	Interrupt raw register	0x000C	0x000C	RO/WTC/SS
I2S_INT_ST_REG	Interrupt status register	0x0010	0x0010	RO
I2S_INT_ENA_REG	Interrupt enable register	0x0014	0x0014	R/W
I2S_INT_CLR_REG	Interrupt clear register	0x0018	0x0018	WT
RX/TX control and configuration registers				
I2S_RX_CONF_REG	RX configuration register	0x0020	0x0020	varies
I2S_RX_CONF1_REG	RX configuration register 1	0x0028	0x0028	R/W
I2S_RX_CLKM_CONF_REG	RX clock configuration register	0x0030	0x0030	R/W
I2S_TX_PCM2PDM_CONF_REG	TX PCM-to-PDM configuration register	0x0040	—	R/W
I2S_TX_PCM2PDM_CONF1_REG	TX PCM-to-PDM configuration register 1	0x0044	—	R/W
I2S_RX_TDM_CTRL_REG	TX TDM mode control register	0x0050	0x0050	R/W
I2S_RXEOF_NUM_REG	RX data number control register	0x0064	0x0064	R/W
I2S_TX_CONF_REG	TX configuration register	0x0024	0x0024	varies
I2S_TX_CONF1_REG	TX configuration register 1	0x002C	0x002C	R/W
I2S_TX_CLKM_CONF_REG	TX clock configuration register	0x0034	0x0034	R/W
I2S_TX_TDM_CTRL_REG	TX TDM mode control register	0x0054	0x0054	R/W
RX clock and timing registers				
I2S_RX_CLKM_DIV_CONF_REG	RX unit clock divider configuration register	0x0038	0x0038	R/W
I2S_RX_TIMING_REG	RX timing control register	0x0058	0x0058	R/W
TX clock and timing registers				
I2S_TX_CLKM_DIV_CONF_REG	TX unit clock divider configuration register	0x003C	0x003C	R/W
I2S_TX_TIMING_REG	TX timing control register	0x005C	0x005C	R/W
Control and configuration registers				
I2S_LC_HUNG_CONF_REG	Timeout configuration register	0x0060	0x0060	R/W
I2S_CONF_SIGLE_DATA_REG	Single data register	0x0068	0x0068	R/W

Name	Description	I2S0 Address	I2S1 Address	Access
TX status registers				
I2S_STATE_REG	TX status register	0x006C	0x006C	RO
Version register				
I2S_DATE_REG	Version control register	0x0080	0x0080	R/W

28.14 Registers

Register 28.1. I2S_INT_RAW_REG (0x000C)



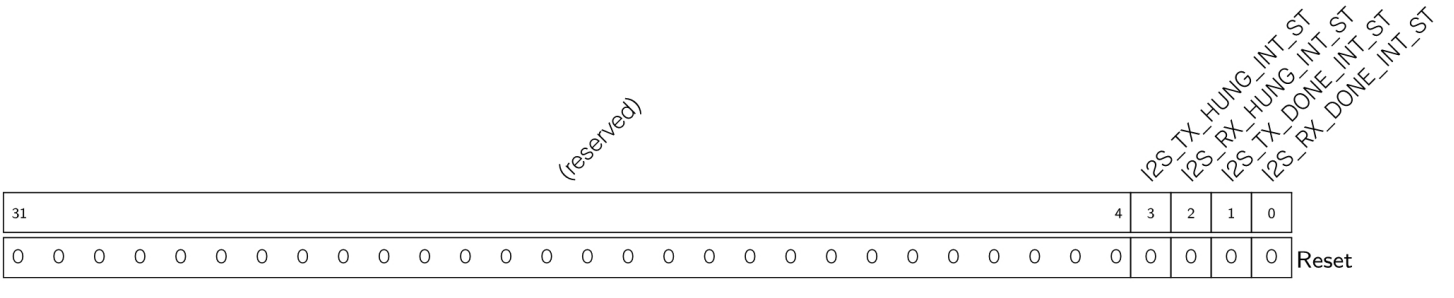
I2S_RX_DONE_INT_RAW The raw interrupt status bit for I2S_RX_DONE_INT interrupt. (RO/WTC/SS)

I2S_TX_DONE_INT_RAW The raw interrupt status bit for I2S_TX_DONE_INT interrupt. (RO/WTC/SS)

I2S_RX_HUNG_INT_RAW The raw interrupt status bit for I2S_RX_HUNG_INT interrupt. (RO/WTC/SS)

I2S_TX_HUNG_INT_RAW The raw interrupt status bit for I2S_TX_HUNG_INT interrupt. (RO/WTC/SS)

Register 28.2. I2S_INT_ST_REG (0x0010)



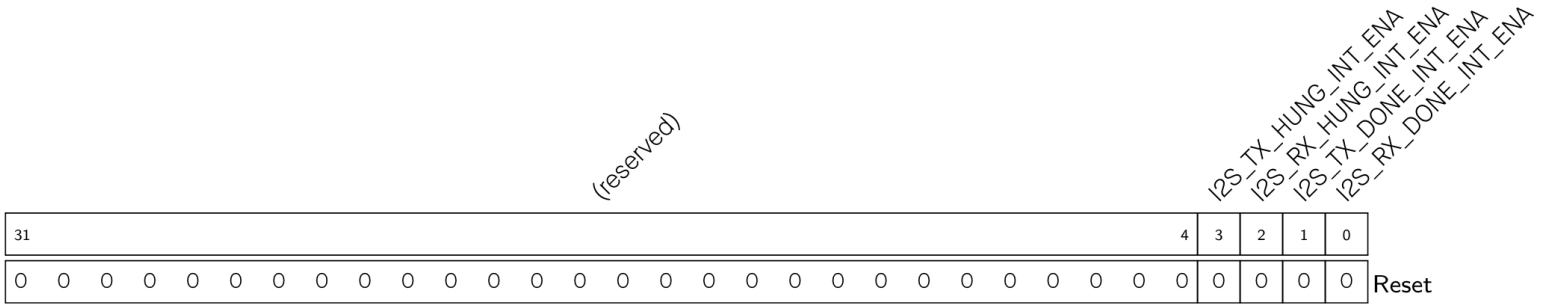
I2S_RX_DONE_INT_ST The masked interrupt status bit for I2S_RX_DONE_INT interrupt. (RO)

I2S_TX_DONE_INT_ST The masked interrupt status bit for I2S_TX_DONE_INT interrupt. (RO)

I2S_RX_HUNG_INT_ST The masked interrupt status bit for I2S_RX_HUNG_INT interrupt. (RO)

I2S_TX_HUNG_INT_ST The masked interrupt status bit for I2S_TX_HUNG_INT interrupt. (RO)

Register 28.3. I2S_INT_ENA_REG (0x0014)



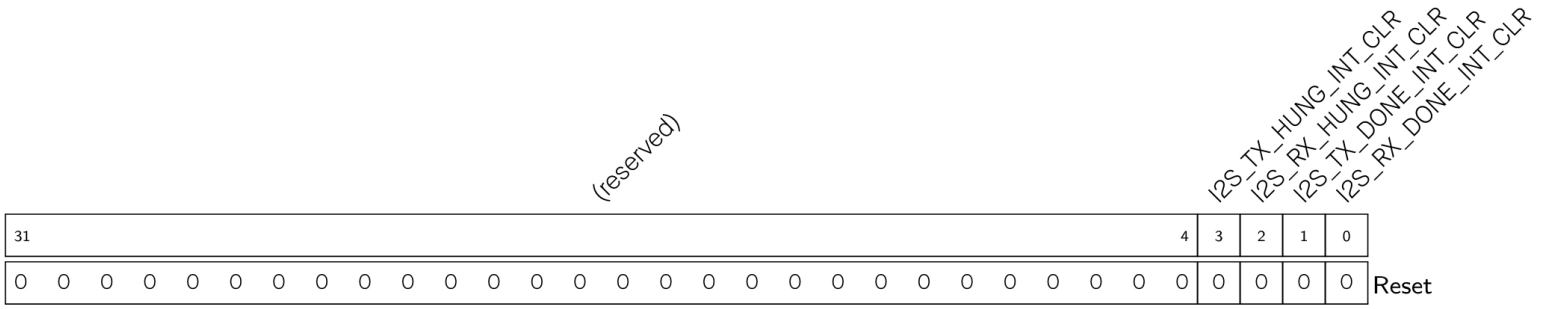
I2S_RX_DONE_INT_ENA The interrupt enable bit for I2S_RX_DONE_INT interrupt. (R/W)

I2S_TX_DONE_INT_ENA The interrupt enable bit for I2S_TX_DONE_INT interrupt. (R/W)

I2S_RX_HUNG_INT_ENA The interrupt enable bit for I2S_RX_HUNG_INT interrupt. (R/W)

I2S_TX_HUNG_INT_ENA The interrupt enable bit for I2S_TX_HUNG_INT interrupt. (R/W)

Register 28.4. I2S_INT_CLR_REG (0x0018)



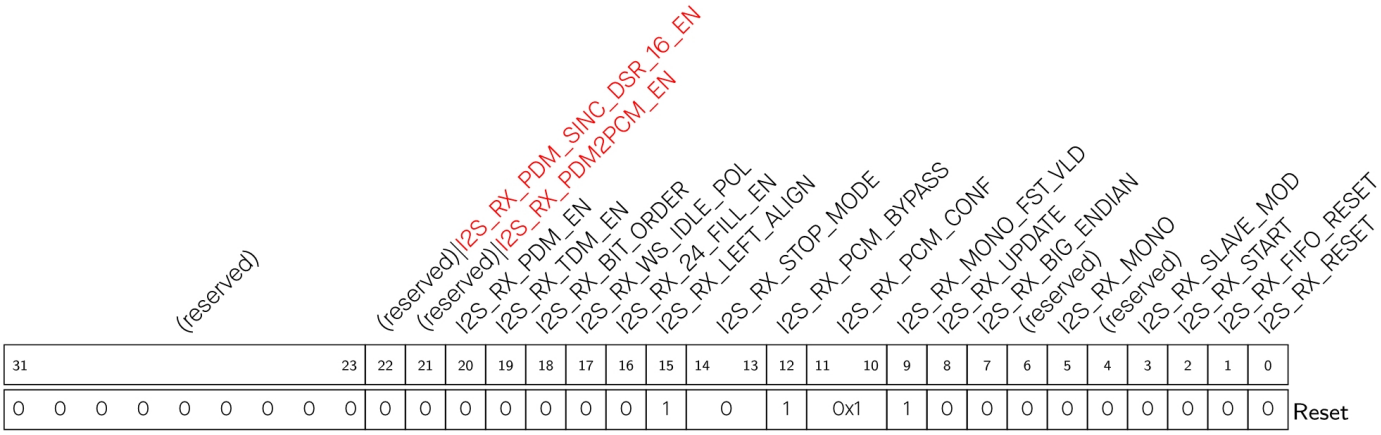
I2S_RX_DONE_INT_CLR Set this bit to clear I2S_RX_DONE_INT interrupt. (WT)

I2S_TX_DONE_INT_CLR Set this bit to clear I2S_TX_DONE_INT interrupt. (WT)

I2S_RX_HUNG_INT_CLR Set this bit to clear I2S_RX_HUNG_INT interrupt. (WT)

I2S_TX_HUNG_INT_CLR Set this bit to clear I2S_TX_HUNG_INT interrupt. (WT)

Register 28.5. I2S_RX_CONF_REG (0x0020)



I2S_RX_RESET Set this bit to reset RX unit. (WT)

I2S_RX_FIFO_RESET Set this bit to reset RX FIFO. (WT)

I2S_RX_START Set this bit to start receiving data. (R/W)

I2S_RX_SLAVE_MOD Set this bit to enable slave RX mode. (R/W)

I2S_RX_MONO Set this bit to enable RX unit in mono mode. (R/W)

I2S_RX_BIG_ENDIAN I2S RX byte endian. 1: low address data is saved to high address. 0: low address data is saved to low address. (R/W)

I2S_RX_UPDATE Set 1 to update I2S RX registers from APB clock domain to I2S RX clock domain. This bit will be cleared by hardware after register update is done. (R/W/SC)

I2S_RX_MONO_FST_VLD 1: The first channel data value is valid in I2S RX mono mode. 0: The second channel data value is valid in I2S RX mono mode. (R/W)

I2S_RX_PCM_CONF I2S RX compress/decompress configuration bit. 0 (atol): A-Law decompress, 1 (ltoa): A-Law compress, 2 (utol): μ-Law decompress, 3 (ltou): μ-Law compress. (R/W)

I2S_RX_PCM_BYPASS Set this bit to bypass Compress/Decompress module for received data. (R/W)

I2S_RX_STOP_MODE 0: I2S RX stops only when I2S_RX_START is cleared. 1: I2S RX stops when I2S_RX_START is 0 or in_suc_eof is 1. 2: I2S RX stops when I2S_RX_STAR is 0 or RX FIFO is full. (R/W)

I2S_RX_LEFT_ALIGN 1: I2S RX left alignment mode. 0: I2S RX right alignment mode. (R/W)

I2S_RX_24_FILL_EN 1: store 24-bit channel data to 32 bits (Extra bits are filled with zeros). 0: store 24-bit channel data to 24 bits. (R/W)

I2S_RX_WS_IDLE_POL 0: WS remains low when receiving left channel data, and remains high when receiving right channel data. 1: WS remains high when receiving left channel data, and remains low when receiving right channel data. (R/W)

I2S_RX_BIT_ORDER Configures whether to reverse the bit order of the I2S RX data to be received. 0: Not reverse. 1: Reverse. (R/W)

Register 28.5. I2S_RX_CONF_REG (0x0020)

Continued from the previous page...

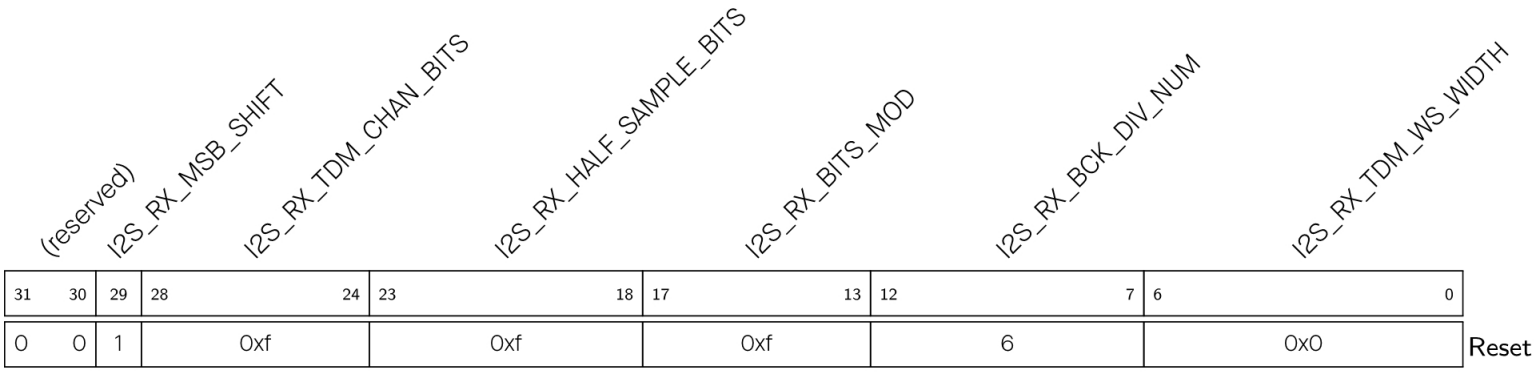
I2S_RX_TDM_EN 1: Enable I2S TDM RX mode. 0: Disable I2S TDM RX mode. (R/W)

I2S_RX_PDM_EN 1: Enable I2S PDM RX mode. 0: Disable I2S PDM RX mode. (R/W)

I2S_RX_PDM2PCM_EN (for I2S0 only) 1: Enable PDM-to-PCM RX mode. 0: Disable PDM-to-PCM RX mode. (R/W)

I2S_RX_PDM_SINC_DSR_16_EN (for I2S0 only) Configure the down sampling rate of PDM RX filter group1 module. 1: The down sampling rate is 128. 0: down sampling rate is 64. (R/W)

Register 28.6. I2S_RX_CONF1_REG (0x0028)



I2S_RX_TDM_WS_WIDTH The width of rx_ws_out (WS default level) in TDM mode is (I2S_RX_TDM_WS_WIDTH + 1) * T_BCK. (R/W)

I2S_RX_BCK_DIV_NUM Configure the divider of BCK in RX mode. Note this divider must not be configured to 1. (R/W)

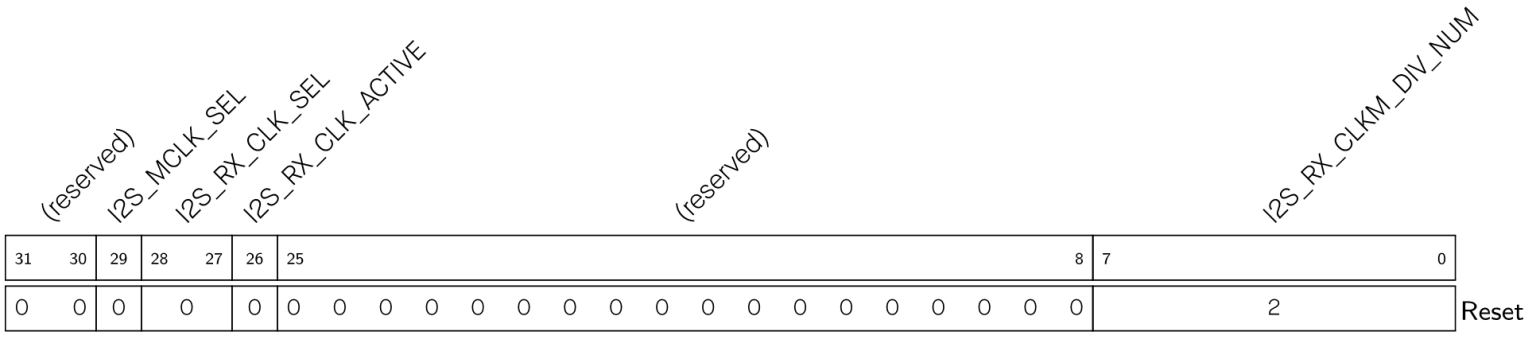
I2S_RX_BITS_MOD Configure the valid data bit length of I2S RX channel. 7: all the valid channel data is in 8-bit mode. 15: all the valid channel data is in 16-bit mode. 23: all the valid channel data is in 24-bit mode. 31: all the valid channel data is in 32-bit mode. (R/W)

I2S_RX_HALF_SAMPLE_BITS I2S RX half sample bits. This value x 2 is equal to the BCK cycles in one WS period. (R/W)

I2S_RX_TDM_CHAN_BITS Configure RX bit number for each channel in TDM mode. Bit number expected = this value + 1. (R/W)

I2S_RX_MSB_SHIFT Control the timing between WS signal and the MSB of data. 1: WS signal changes one BCK clock earlier. 0: Align at rising edge. (R/W)

Register 28.7. I2S_RX_CLKM_CONF_REG (0x0030)



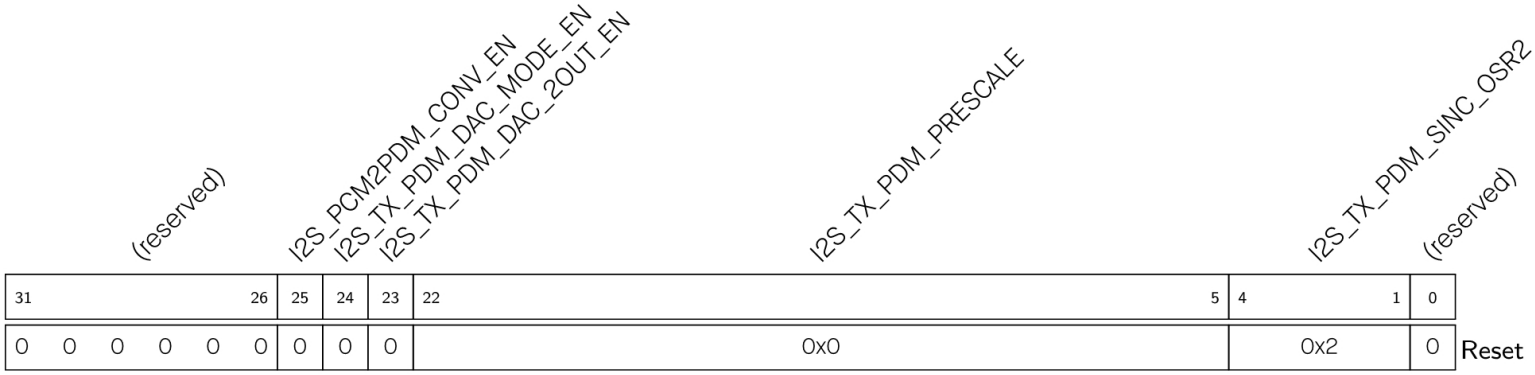
I2S_RX_CLKM_DIV_NUM Integral I2S RX clock divider value. (R/W)

I2S_RX_CLK_ACTIVE Clock enable signal of I2S RX unit. (R/W)

I2S_RX_CLK_SEL Select clock source for I2S RX unit. 0: XTAL_CLK. 1: PLL_D2_CLK. 2: PLL_F160M_CLK. 3: I2S_MCLK_in. (R/W)

I2S_MCLK_SEL 0: Use I2S TX unit clock as I2S_MCLK_OUT. 1: Use I2S RX unit clock as I2S_MCLK_OUT. (R/W)

Register 28.8. I2S_TX_PCM2PDM_CONF_REG (For I2S0 Only) (0x0040)



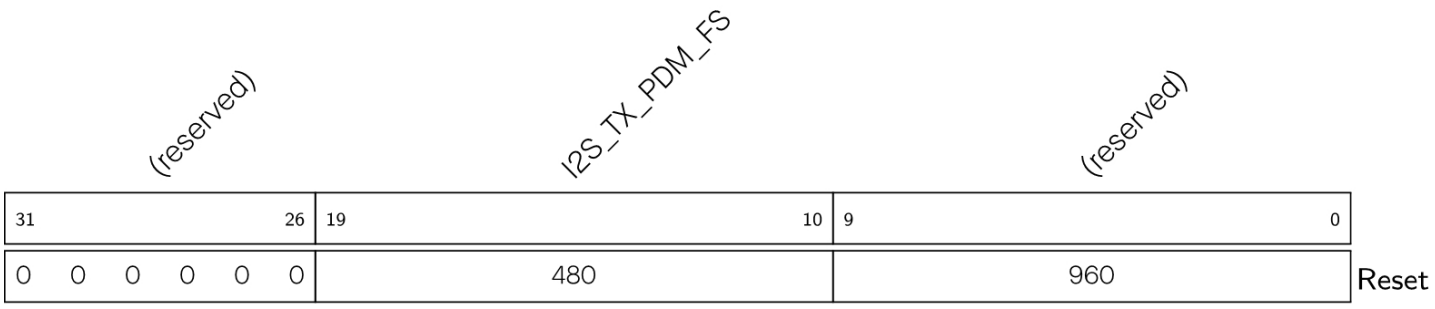
I2S_TX_PDM_SINC_OSR2 I2S TX PDM OSR value. (R/W)

I2S_TX_PDM_DAC_2OUT_EN 0: 1-line DAC output mode. 1: 2-line DAC output mode. Only valid when I2S_TX_PDM_DAC_MODE_EN is set. (R/W)

I2S_TX_PDM_DAC_MODE_EN 0: 1-line PDM output mode. 1: DAC output mode. (R/W)

I2S_PCM2PDM_CONV_EN Enable bit for I2S TX PCM-to-PDM conversion. (R/W)

Register 28.9. I2S_TX_PCM2PDM_CONF1_REG (For I2S0 Only) (0x0044)



I2S_TX_PDM_FS I2S PDM TX upsampling parameter. (R/W)

Register 28.10. I2S_RX_TDM_CTRL_REG (0x0050)

I2S_RX_TDM_PDM_CHAN0_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 0. 0: Disable. Channel 0 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN1_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 1. 0: Disable. Channel 1 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN2_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 2. 0: Disable. Channel 2 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN3_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 3. 0: Disable. Channel 3 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN4_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 4. 0: Disable. Channel 4 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN5_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 5. 0: Disable. Channel 5 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN6_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 6. 0: Disable. Channel 6 only inputs 0. (R/W)

I2S_RX_TDM_PDM_CHAN7_EN 1: Enable the valid data input of I2S RX TDM or PDM channel 7. 0: Disable. Channel 0 only inputs 7. (R/W)

I2S_RX_TDM_CHAN8_EN 1: Enable the valid data input of I2S RX TDM channel 8. 0: Disable. Channel 8 only inputs 0. (R/W)

I2S_RX_TDM_CHAN9_EN 1: Enable the valid data input of I2S RX TDM channel 9. 0: Disable. Channel 9 only inputs 0. (R/W)

I2S_RX_TDM_CHAN10_EN 1: Enable the valid data input of I2S RX TDM channel 10. 0: Disable. Channel 10 only inputs 0. (R/W)

I2S_RX_TDM_CHAN11_EN 1: Enable the valid data input of I2S RX TDM channel 11. 0: Disable. Channel 11 only inputs 0. (R/W)

I2S_RX_TDM_CHAN12_EN 1: Enable the valid data input of I2S RX TDM channel 12. 0: Disable. Channel 12 only inputs 0. (R/W)

I2S_RX_TDM_CHAN13_EN 1: Enable the valid data input of I2S RX TDM channel 13. 0: Disable. Channel 13 only inputs 0. (R/W)

Register 28.10. I2S_RX_TDM_CTRL_REG (0x0050)

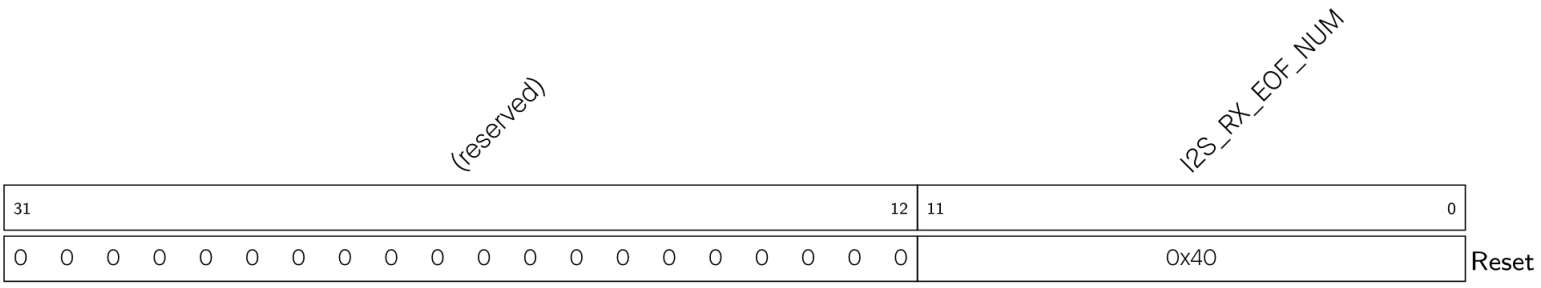
Continued from the previous page...

I2S_RX_TDM_CHAN14_EN 1: Enable the valid data input of I2S RX TDM channel 14. 0: Disable. Channel 14 only inputs 0. (R/W)

I2S_RX_TDM_CHAN15_EN 1: Enable the valid data input of I2S RX TDM channel 15. 0: Disable. Channel 15 only inputs 0. (R/W)

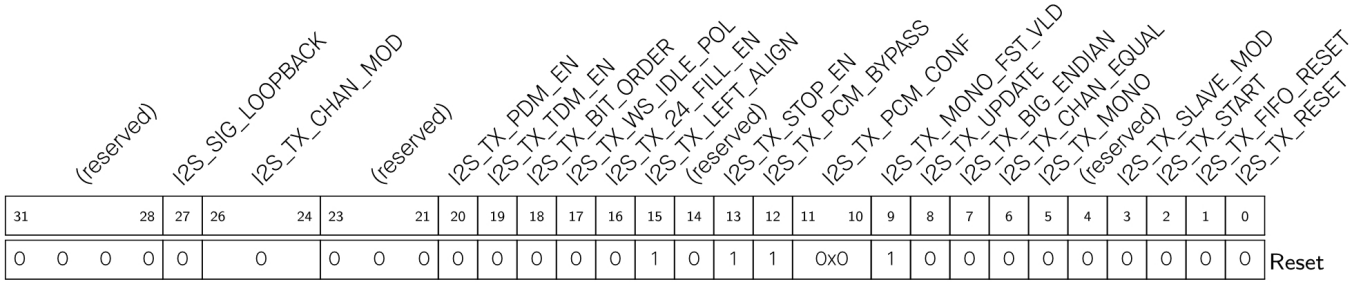
I2S_RX_TDM_TOT_CHAN_NUM The total number of channels in use in I2S RX TDM mode. Total channel number in use = this value + 1. (R/W)

Register 28.11. I2S_RXEOF_NUM_REG (0x0064)



I2S_RX_EOF_NUM The bit length of RX data is (I2S_RX_BITS_MOD + 1) * (I2S_RX_EOF_NUM + 1). Once the length of received data reaches such bit length, an in_suc_eof interrupt is triggered in the configured DMA RX channel. (R/W)

Register 28.12. I2S_TX_CONF_REG (0x0024)



I2S_TX_RESET Set this bit to reset TX unit. (WT)

I2S_TX_FIFO_RESET Set this bit to reset TX FIFO. (WT)

I2S_TX_START Set this bit to start transmitting data. (R/W)

I2S_TX_SLAVE_MOD Set this bit to enable slave TX mode. (R/W)

I2S_TX_MONO Set this bit to enable TX unit in mono mode. (R/W)

I2S_TX_CHAN_EQUAL 1: The left channel data is equal to right channel data in I2S TX mono mode or TDM mode. 0: The invalid channel data is I2S_SINGLE_DATA in I2S TX mono mode or TDM mode. (R/W)

I2S_TX_BIG_ENDIAN I2S TX byte endian. 1: low address data is saved to high address. 0: low address data is saved to low address. (R/W)

I2S_TX_UPDATE Set 1 to update I2S TX registers from APB clock domain to I2S TX clock domain. This bit will be cleared by hardware after register update is done. (R/W/SC)

I2S_TX_MONO_FST_VLD 1: The first channel data is valid in I2S TX mono mode. 0: The second channel data is valid in I2S TX mono mode. (R/W)

I2S_TX_PCM_CONF I2S TX compress/decompress configuration bits. 0 (atol): A-Law decompress, 1 (ltoa): A-Law compress, 2 (utol): μ-Law decompress, 3 (ltou): μ-Law compress. (R/W)

I2S_TX_PCM_BYPASS Set this bit to bypass Compress/Decompress module for transmitted data. (R/W)

I2S_TX_STOP_EN Set this bit to stop outputting BCK signal and WS signal when TX FIFO is empty. (R/W)

I2S_TX_LEFT_ALIGN 1: I2S TX left alignment mode. 0: I2S TX right alignment mode. (R/W)

I2S_TX_24_FILL_EN 1: Send 32 bits in 24-bit channel data mode. Extra bits are filled with zeros. 0: Send 24 bits in 24-bit channel data mode. (R/W)

I2S_TX_WS_IDLE_POL 0: WS remains low when sending left channel data, and remains high when sending right channel data. 1: WS remains high when sending left channel data, and remains low when sending right channel data. (R/W)

I2S_TX_BIT_ORDER Configures whether to reverse the bit order of valid data to be sent by the I2S TX. 0: Not reverse. 1: Reverse. (R/W)

I2S_TX_TDM_EN 1: Enable I2S TDM TX mode. 0: Disable I2S TDM TX mode. (R/W)

Register 28.12. I2S_TX_CONF_REG (0x0024)

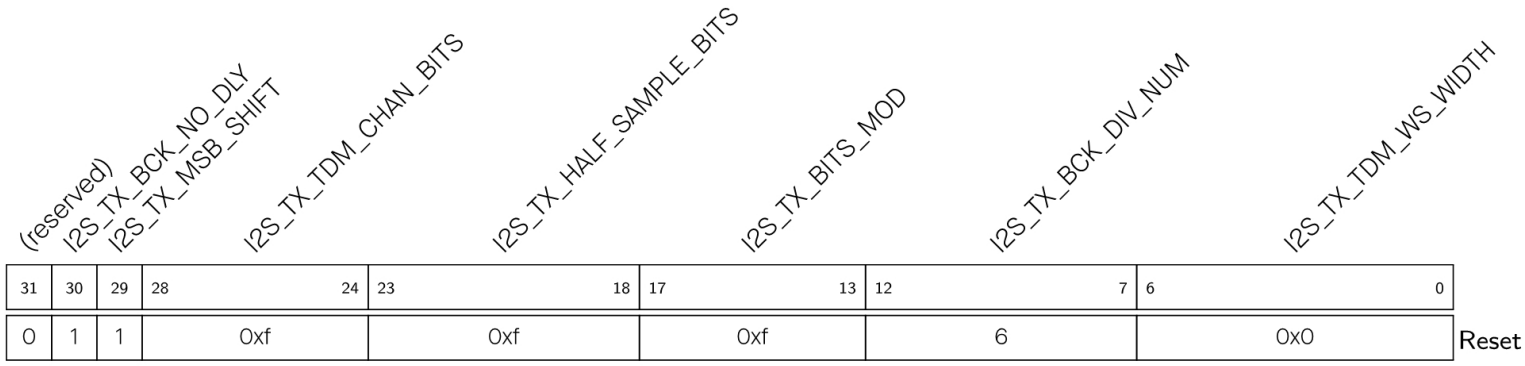
Continued from the previous page...

I2S_TX_PDM_EN 1: Enable I2S PDM TX mode. 0: Disable I2S PDM TX mode. (R/W)

I2S_TX_CHAN_MOD I2S TX channel configuration bits. For more information, see Table 28.9-4. (R/W)

I2S_SIG_LOOPBACK Enable signal loop back mode with TX unit and RX unit sharing the same WS and BCK signals. (R/W)

Register 28.13. I2S_TX_CONF1_REG (0x002C)



I2S_TX_TDM_WS_WIDTH The width of tx_ws_out (WS default level) in TDM mode is (I2S_TX_TDM_WS_WIDTH + 1) * T_BCK. (R/W)

I2S_TX_BCK_DIV_NUM Configure the divider of BCK in TX mode. Note this divider must not be configured to 1. (R/W)

I2S_TX_BITS_MOD Set the bits to configure the valid data bit length of I2S TX channel. 7: all the valid channel data is in 8-bit mode. 15: all the valid channel data is in 16-bit mode. 23: all the valid channel data is in 24-bit mode. 31: all the valid channel data is in 32-bit mode. (R/W)

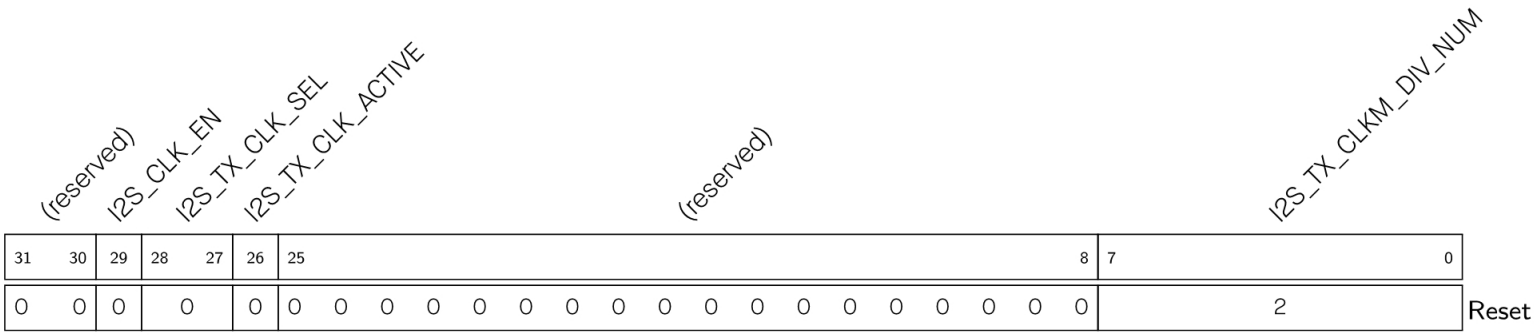
I2S_TX_HALF_SAMPLE_BITS I2S TX half sample bits. This value x 2 is equal to the BCK cycles in one WS period. (R/W)

I2S_TX_TDM_CHAN_BITS Configure TX bit number for each channel in TDM mode. Bit number expected = this value + 1. (R/W)

I2S_TX_MSB_SHIFT Control the timing between WS signal and the MSB of data. 1: WS signal changes one BCK clock earlier. 0: Align at rising edge. (R/W)

I2S_TX_BCK_NO_DLY 1: BCK is not delayed to generate rising/falling edge in master mode. 0: BCK is delayed to generate rising/falling edge in master mode. (R/W)

Register 28.14. I2S_TX_CLKM_CONF_REG (0x0034)



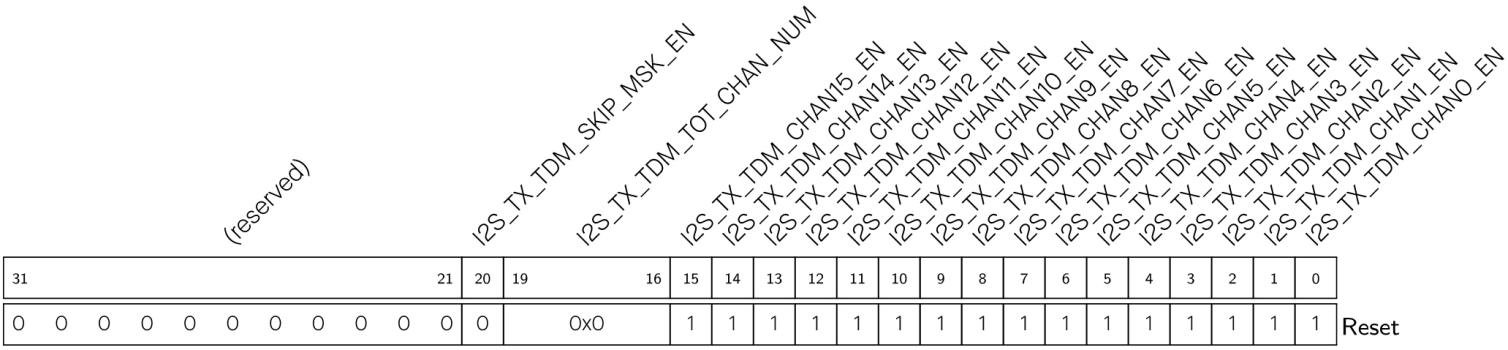
I2S_TX_CLKM_DIV_NUM Integral I2S TX clock divider value. (R/W)

I2S_TX_CLK_ACTIVE I2S TX unit clock enable signal. (R/W)

I2S_TX_CLK_SEL Select clock clock for I2S TX unit. 0: XTAL_CLK. 1: PLL_D2_CLK. 2: PLL_F160M_CLK. 3: I2S_MCLK_in. (R/W)

I2S_CLK_EN Set this bit to enable clock gate. (R/W)

Register 28.15. I2S_TX_TDM_CTRL_REG (0x0054)



I2S_TX_TDM_CHAN0_EN 1: Enable the valid data output of I2S TX TDM channel 0. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN1_EN 1: Enable the valid data output of I2S TX TDM channel 1. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN2_EN 1: Enable the valid data output of I2S TX TDM channel 2. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN3_EN 1: Enable the valid data output of I2S TX TDM channel 3. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN4_EN 1: Enable the valid data output of I2S TX TDM channel 4. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN5_EN 1: Enable the valid data output of I2S TX TDM channel 5. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN6_EN 1: Enable the valid data output of I2S TX TDM channel 6. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN7_EN 1: Enable the valid data output of I2S TX TDM channel 7. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN8_EN 1: Enable the valid data output of I2S TX TDM channel 8. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN9_EN 1: Enable the valid data output of I2S TX TDM channel 9. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

Register 28.15. I2S_TX_TDM_CTRL_REG (0x0054)

Continued from the previous page...

I2S_TX_TDM_CHAN10_EN 1: Enable the valid data output of I2S TX TDM channel 10. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN11_EN 1: Enable the valid data output of I2S TX TDM channel 11. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN12_EN 1: Enable the valid data output of I2S TX TDM channel 12. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN13_EN 1: Enable the valid data output of I2S TX TDM channel 13. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

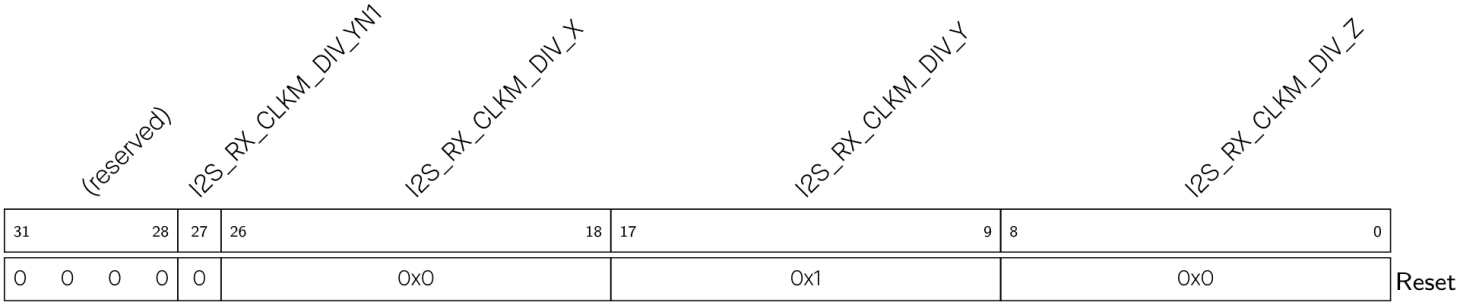
I2S_TX_TDM_CHAN14_EN 1: Enable the valid data output of I2S TX TDM channel 14. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_CHAN15_EN 1: Enable the valid data output of I2S TX TDM channel 15. 0: Channel TX data is controlled by I2S_TX_CHAN_EQUAL and I2S_SINGLE_DATA. See Section 28.9.2.1. (R/W)

I2S_TX_TDM_TOT_CHAN_NUM Set the total number of channels in use in I2S TX TDM mode. Total channel number in use = this value + 1. (R/W)

I2S_TX_TDM_SKIP_MSK_EN When DMA TX buffer stores the data of (I2S_TX_TDM_TOT_CHAN_NUM + 1) channels, and only the data of the enabled channels is sent, then this bit should be set. Clear it when all the data stored in DMA TX buffer is for enabled channels. (R/W)

Register 28.16. I2S_RX_CLKM_DIV_CONF_REG (0x0038)



I2S_RX_CLKM_DIV_Z For b <= a/2, the value of I2S_RX_CLKM_DIV_Z is b. For b > a/2, the value of I2S_RX_CLKM_DIV_Z is (a - b). (R/W)

I2S_RX_CLKM_DIV_Y For b <= a/2, the value of I2S_RX_CLKM_DIV_Y is (a%b). For b > a/2, the value of I2S_RX_CLKM_DIV_Y is (a%(a - b)). (R/W)

I2S_RX_CLKM_DIV_X For b <= a/2, the value of I2S_RX_CLKM_DIV_X is floor(a/b) - 1. For b > a/2, the value of I2S_RX_CLKM_DIV_X is floor(a/(a - b)) - 1. (R/W)

I2S_RX_CLKM_DIV_YN1 For b <= a/2, the value of I2S_RX_CLKM_DIV_YN1 is 0. For b > a/2, the value of I2S_RX_CLKM_DIV_YN1 is 1. (R/W)

Note:

“a” and “b” represent the denominator and the numerator of fractional divider, respectively. For more information, see Section 28.6.

Register 28.17. I2S_RX_TIMING_REG (0x0058)

I2S_RX_SD_IN_DM The delay mode of I2S RX SD input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_RX_SD1_IN_DM (for I2S0 only) The delay mode of I2S RX SD1 input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_RX_SD2_IN_DM (for I2S0 only) The delay mode of I2S RX SD2 input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_RX_SD3_IN_DM (for I2S0 only) The delay mode of I2S RX SD3 input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

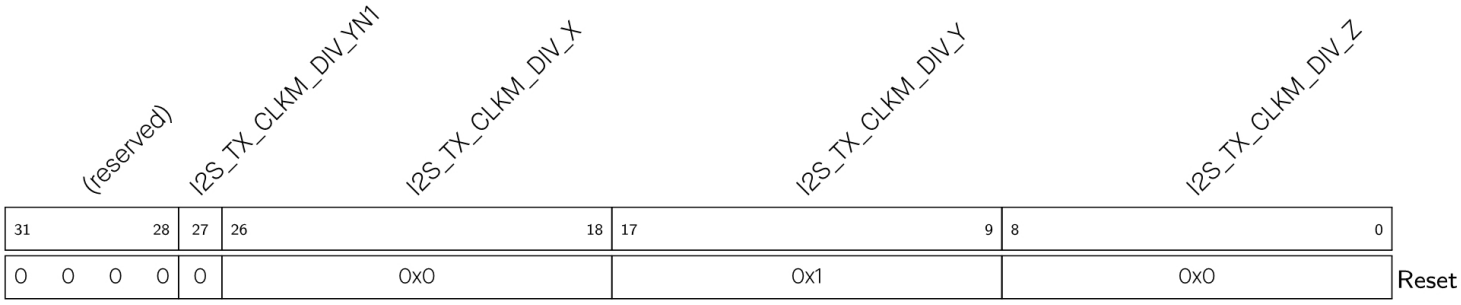
I2S_RX_WS_OUT_DM The delay mode of I2S RX WS output signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_RX_BCK_OUT_DM The delay mode of I2S RX BCK output signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_RX_WS_IN_DM The delay mode of I2S RX WS input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_RX_BCK_IN_DM The delay mode of I2S RX BCK input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

Register 28.18. I2S_TX_CLKM_DIV_CONF_REG (0x003C)



I2S_TX_CLKM_DIV_Z For b <= a/2, the value of I2S_TX_CLKM_DIV_Z is b. For b > a/2, the value of I2S_TX_CLKM_DIV_Z is (a - b). (R/W)

I2S_TX_CLKM_DIV_Y For b <= a/2, the value of I2S_TX_CLKM_DIV_Y is (a%b). For b > a/2, the value of I2S_TX_CLKM_DIV_Y is (a%(a - b)). (R/W)

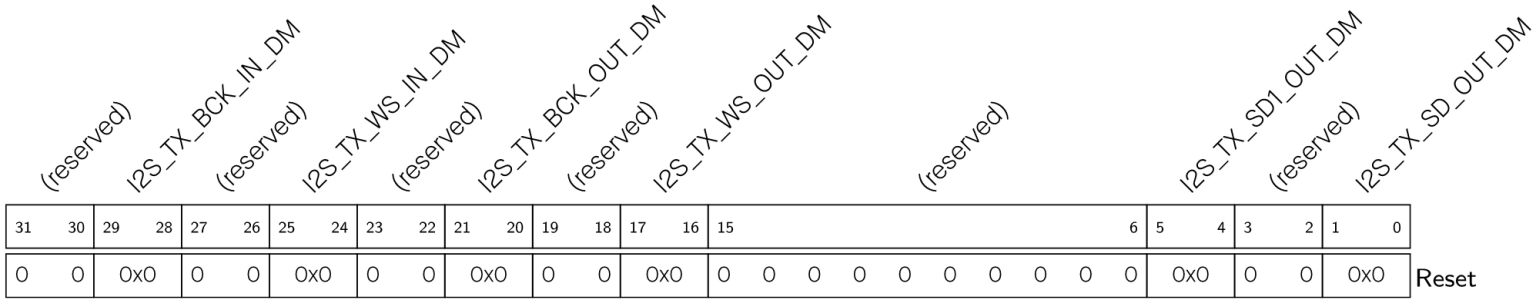
I2S_TX_CLKM_DIV_X For b <= a/2, the value of I2S_TX_CLKM_DIV_X is floor(a/b) - 1. For b > a/2, the value of I2S_TX_CLKM_DIV_X is floor(a/(a - b)) - 1. (R/W)

I2S_TX_CLKM_DIV_YN1 For b <= a/2, the value of I2S_TX_CLKM_DIV_YN1 is 0. For b > a/2, the value of I2S_TX_CLKM_DIV_YN1 is 1. (R/W)

Note:

“a” and “b” represent the denominator and the numerator of fractional divider, respectively. For more information, see Section 28.6.

Register 28.19. I2S_TX_TIMING_REG (0x005C)



I2S_TX_SD_OUT_DM The delay mode of I2S TX SD output signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_TX_SD1_OUT_DM The delay mode of I2S TX SD1 output signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

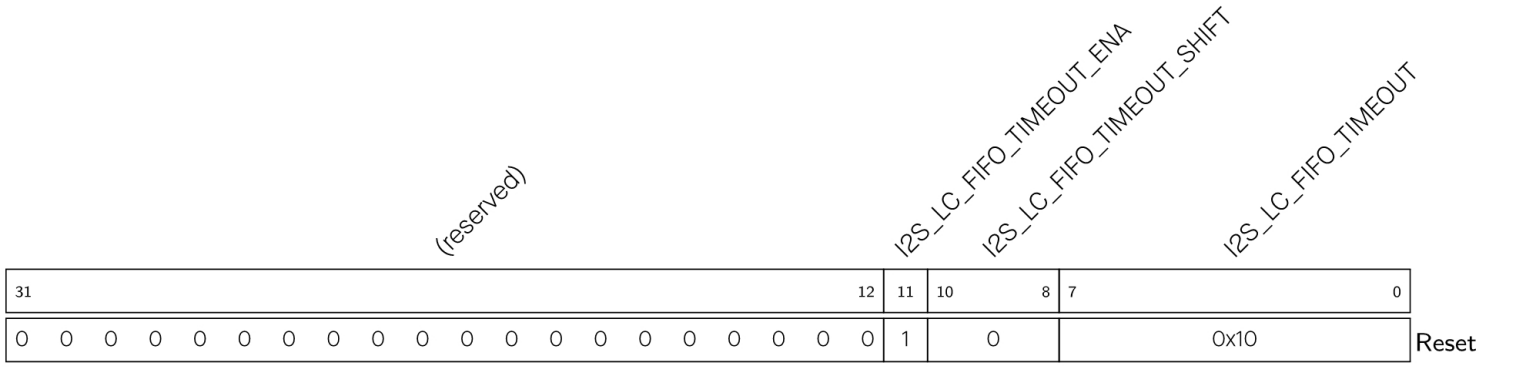
I2S_TX_WS_OUT_DM The delay mode of I2S TX WS output signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_TX_BCK_OUT_DM The delay mode of I2S TX BCK output signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_TX_WS_IN_DM The delay mode of I2S TX WS input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

I2S_TX_BCK_IN_DM The delay mode of I2S TX BCK input signal. 0: bypass. 1: delay by rising edge. 2: delay by falling edge. 3: not used. (R/W)

Register 28.20. I2S_LC_HUNG_CONF_REG (0x0060)

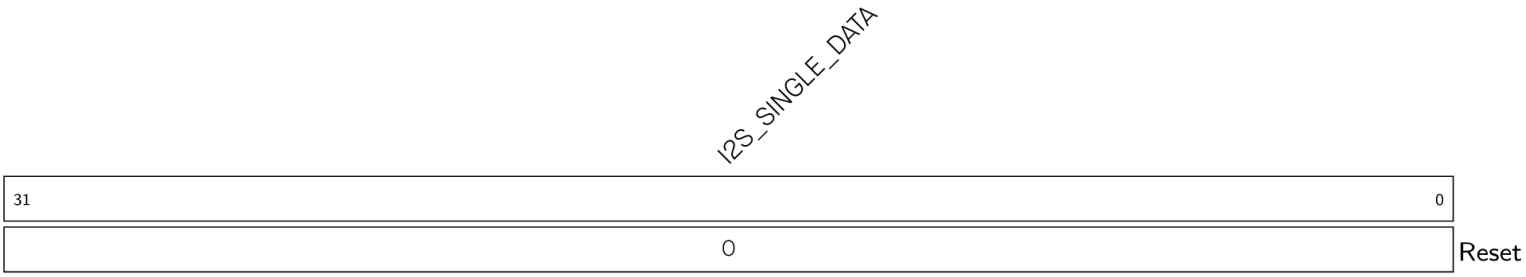


I2S_LC_FIFO_TIMEOUT I2S_TX_HUNG_INT or I2S_RX_HUNG_INT interrupt will be triggered when FIFO hung counter is equal to this value. (R/W)

I2S_LC_FIFO_TIMEOUT_SHIFT The bits are used to scale tick counter threshold. The tick counter is reset when counter value >= $88000/2^{I2S\_LC\_FIFO\_TIMEOUT\_SHIFT}$. (R/W)

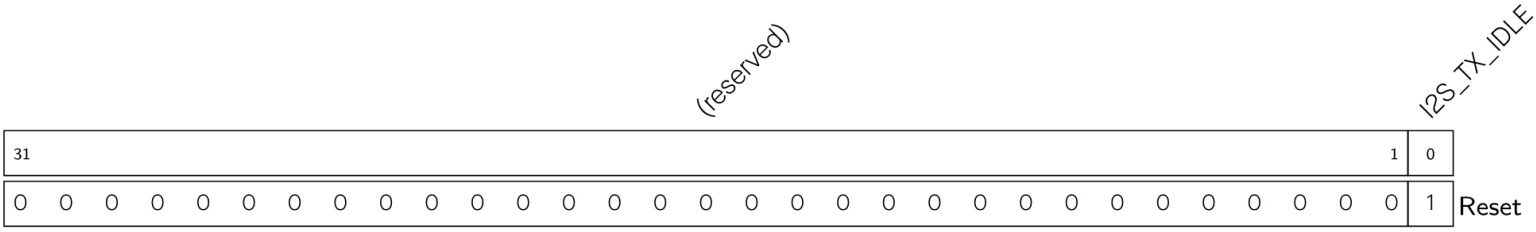
I2S_LC_FIFO_TIMEOUT_ENA The enable bit for FIFO timeout. (R/W)

Register 28.21. I2S_CONF_SIGLE_DATA_REG (0x0068)



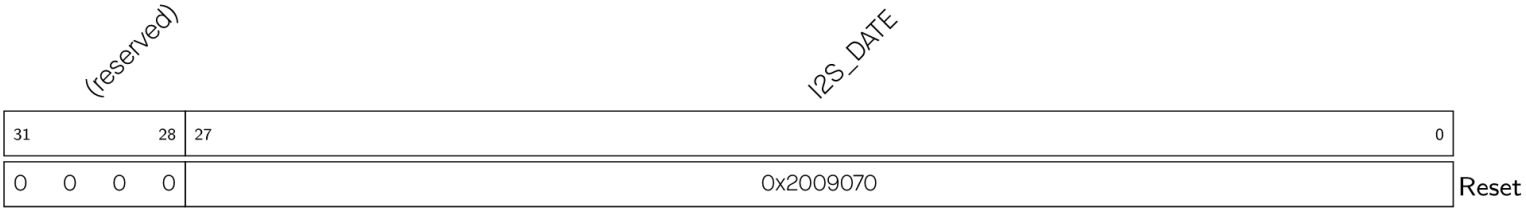
I2S_SINGLE_DATA The configured constant channel data to be sent out. (R/W)

Register 28.22. I2S_STATE_REG (0x006C)



I2S_TX_IDLE 1: I2S TX unit is in idle state. 0: I2S TX unit is working. (RO)

Register 28.23. I2S_DATE_REG (0x0080)



I2S_DATE Version control register. (R/W)