I2S Driver
User Guide
V1.00.01

I2S Driver 3
1.1. I2S Introduction 3
1.2. I2S Feature 3
1.3. Constant Definition 3
1.4. Type Definition 4
E_I2S_CHANNEL 4
E_I2S_CALLBACK_TYPE 4
1.5. Macro Functions 5
_DRVI2S_WRITE_TX_FIFO 5
_DRVI2S_READ_RX_FIFO 5
_DRVI2S_READ_TX_FIFO_LEVEL 6
_DRVI2S_READ_RX_FIFO_LEVEL 7
1.6. Functions 7
DrvI2S_Open 7
DrvI2S_Close 9
DrvI2S_EnableInt 10
DrvI2S_DisableInt 11
DrvI2S_GetBCLK 12
DrvI2S_SetBCLK 13
DrvI2S_GetMCLK 13
DrvI2S_SetMCLK 14
DrvI2S_EnableZeroCrossDetect 14
DrvI2S_EnableTxDMA 15
DrvI2S_EnableRxDMA 16
DrvI2S_EnableTx 16
DrvI2S_EnableRx 17
DrvI2S_EnableTxMute 18
DrvI2S_EnableMCLK 18
DrvI2S_ClearTxFIFO 19
DrvI2S_ClearRxFIFO 19
DrvI2S_SelectClockSource 20
DrvI2S_GetSourceClock 21
DrvI2S_GetVersion 21
I2S_IRQHandler 22
2. Revision History 23

I2S Driver

1.1. I2S Introduction

The I2S controller is a peripheral for serial transmission and reception of audio PCM (Pulse-Code Modulated) signals across a 4-wire bus. The bus consists of a bit clock (I2S_BCLK) a frame synchronization clock (I2S_FS) and serial data in (I2S_SDI) and out (I2S_SDO) lines. This peripheral allows communication with an external audio CODEC or DSP. The peripheral is capable of mono or stereo audio transmission with 8-32bit word sizes. Audio data is buffered in 8 word deep FIFO buffers and has DMA capability.

1.2. I2S Feature

- I2S can operate as either master or slave
- Master clock generation for slave device synchronization.
- Capable of handling 8, 16, 24 and 32 bit word sizes.
- Mono and stereo audio data supported.
- I2S and MSB justified data format supported.
- 8 word FIFO data buffers for transmit and receive.
- Generates interrupt requests when buffer levels crosses programmable boundary.
- Two DMA requests, one for transmit and one for receive.

1.3. Constant Definition

Constant Name	Value	Description
DRVI2S_DATABIT_8	0x00	Data size is 8 bit
DRVI2S_DATABIT_16	0x01	Data size is 16 bit
DRVI2S_DATABIT_24	0x02	Data size is 24 bit

DRVI2S_DATABIT_32	0x03	Data size is 32 bit
DRVI2S_STEREO	0x00	Data is stereo format
DRVI2S_MONO	0x01	Data is mono format
DRVI2S_FORMAT_I2S	0x00	I2S data format
DRVI2S_FORMAT_MSB	0x01	MSB justified data format
DRVI2S_MODE_MASTER	0x00	I2S operates as master mode
DRVI2S_MODE_SLAVE	0x01	I2S operates as slave mode
DRVI2S_FIFO_LEVEL_WORD_0	0x00	FIFO threshold level is 0 word
DRVI2S_FIFO_LEVEL_WORD_1	0x01	FIFO threshold level is 1 word
DRVI2S_FIFO_LEVEL_WORD_2	0x02	FIFO threshold level is 2 word
DRVI2S_FIFO_LEVEL_WORD_3	0x03	FIFO threshold level is 3 word
DRVI2S_FIFO_LEVEL_WORD_4	0x04	FIFO threshold level is 4 word
DRVI2S_FIFO_LEVEL_WORD_5	0x05	FIFO threshold level is 5 word
DRVI2S_FIFO_LEVEL_WORD_6	0x06	FIFO threshold level is 6 word
DRVI2S_FIFO_LEVEL_WORD_7	0x07	FIFO threshold level is 7 word
DRVI2S_FIFO_LEVEL_WORD_8	0x08	FIFO threshold level is 8 word
DRVI2S_INTERNAL_10K	0	I2S clock source is from internal 10KHz oscillator.
DRVI2S_EXT_32K	1	I2S clock source is from external 32KHz crystal clock
DRVI2S_HCLK	2	I2S clock source is from HCLK
DRVI2S_INTERNAL_48M	3	I2S clock source is from internal 48MHz oscillator.

1.4. Type Definition

E_I2S_CHANNEL

Enumeration identifier	Value	Description
I2S_LEFT_CHANNEL	0	I2S for left channel
I2S_RIGHT_CHANNEL	1	I2S for right channel

E_I2S_CALLBACK_TYPE

Enumeration identifier	Value	Description
I2S_RX_UNDERFLOW	0	For RX FIFO underflow interrupt

I2S_RX_OVERFLOW	1	For RX FIFO overflow interrupt
I2S_RX_FIFO_THRESHOLD	2	For RX FIFO threshold level interrupt
I2S_TX_UNDERFLOW	8	For TX FIFO underflow interrupt
I2S_TX_OVERFLOW	9	For TX FIFO overflow interrupt
I2S_TX_FIFO_THRESHOLD	10	For TX FIFO threshold level interrupt
I2S_TX_RIGHT_ZERO_CROSS	11	For TX right channel zero cross interrupt
I2S_TX_LEFT_ZERO_CROSS	12	For TX left channel zero cross interrupt

1.5. Macro Functions

_DRVI2S_WRITE_TX_FIFO

Prototype

static __inline void _DRVI2S_WRITE_TX_FIFO (uint32_t u32Data);

Description

Write word data to Tx FIFO.

Parameter

u32Data [in]

Write data to Tx FIFO.

Include

Driver/DrvI2S.h

Return Value

None

Example

/* Write word data 0x12345678 into I2S Tx FIFO */

_DRVI2 S_WRITE_TX_FIFO (0x12345678);

_DRVI2S_READ_RX_FIFO

Prototype

static __inline uint32_t _DRVI2 S_READ_RX_FIFO (void);

Description

Read out word data from Rx FIFO.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

Word data from Rx FIFO.

Example

```
uint32_t u32data;
/* Read wo rd data from I2S Rx FIFO */
u32data = _DRVI2S_READ_ RX_FIFO ();
```

_DRVI2S_READ_TX_FIFO_LEVEL

Prototype

static __inline uint32_t _DRVI2 S_READ_TX_FIFO_LEVEL (void);

Description

Get word data number in Tx FIFO.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

0~8: word data in Tx FIFO

Example

```
uint32_t u32len;
```

```
/* Get word data number in Tx FIFO */
u32len = _DRVI2S_READ_TX_ FIFO_LEVEL ();
```

_DRVI2S_READ_RX_FIFO_LEVEL

Prototype

static __inline uint32_t _DRVI2 S_READ_RX_FIFO_LEVEL (void);

Description

Get word data number in Rx FIFO.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

0~8: word data in Rx FIFO

Example

```
uint32_t u32len;
/* Get word data number in Rx FIFO */
u32len = _DRVI2S_READ_RX_FIFO_LEVEL ( );
```

1.6. Functions

DrvI2S_Open

Prototype

int32_t DrvI2S_Open (S_DRVI2S_ DATA_T *sParam);

Description

This function is used to enable I2S clock and function, and configure the data length/data format/FIFO threshold level/BCLK (Bit Clock). The data

and audio formats are shown in I2S Operation and FIFO Operation of I2S Section in TRM. For master mode, I2S_BCLK and I2S_ LRCLK pins are output mode; for slave mode, I2S_BCLK and I2S _LRCLK pins are input mode. Also, the I2S signals (I2S_BCLK and I2S_LRCLK) are shown in I2S Block Diagram of I2S Section in TRM.

Parameter

***sParam [in]**

It includes the following parameter

u32SampleRate: Sampling rate. The setting takes effect when I2S operates as master mode.

u8WordWidth: 8, 16, 24, or 32 bit data size - DRVI2S_DATABIT_8/DRVI2S_DATABIT_16/DRVI2S_DATABIT_24/DRVI2S_DATABIT_32

u8AudioFormat: Support mono or stereo audio data - DRVI2S_MONO/ DRVI2S_STEREO

u8DataFormat: Support I2S and MSB justified data format - DRVI2S_FORMAT_I2S/DRVI2S_FORMAT_MSB

u8Mode: Operate as master or slave mode - DRVI2S_MODE_MASTER / DRVI2S_MODE_SLAVE

u8TxFIFOThreshold: Tx FIFO threshold level - DRVI2S_FIFO_LEVEL_WORD_0 / DRVI2S_FIFO_LEVEL_WORD_ 1 / DRVI2S_FIFO_LEVEL_WORD_ 2 / DRVI2S_FIFO_LEVEL_WORD_ 3 / DRVI2S_FIFO_LEVEL_WORD_ 4 / DRVI2S_FIFO_LEVEL_WORD_ 5 / DRVI2S_FIFO_LEVEL_WORD_ 6 / DRVI2S_FIFO_LEVEL_WORD_ 7

u8RxFIFOThreshold: Rx FIFO threshold level - DRVI2S_FIFO_LEVEL_WORD_1 / DRVI2S_FIFO_LEVEL_WORD_2 / DRVI2S_FIFO_LEVEL_WORD_3 / DRVI2S_FIFO_LEVEL_WORD_4 /

DRVI2S_FIFO_LEVEL_WORD_5 /
DRVI2S_FIFO_LEVEL_WORD_6 /
DRVI2S_FIFO_LEVEL_WORD_7 /
DRVI2S_FIFO_LEVEL_WORD_8

Include

Driver/DrvI2S.h

Return Value

0 Success

Example

```
S_DRVI2S_DATA_T st;

st.u32SampleRate = 16000; /* Sampling rate is 16ksps */
st.u8WordWidth = DRVI2S_DATABIT_16; /* Data length is 16-bit */
st.u8AudioFormat = DRVI2S_STEREO; /* Stereo format */
st.u8DataFormat = DRVI2S_FORMAT_I2S; /* I2S data format */
st.u8Mode = DRVI2S_MODE_MASTER; /* Operate as master mode */
/* Tx FIFO threshold level is 0 word data */
st.u8TxFIFOThreshold = DRVI2S_FIFO_LEVEL_WORD_0;
/* Rx FIFO threshold level is 8 word data */
st.u8RxFIFOThreshold = DRVI2S_FIFO_LEVEL_WORD_ 8;
/* Enable I2S and configure its settings */
DrvI2S_Open (&st);
```

DrvI2S_Close

Prototype

void DrvI2S_Close (void);

Description

Close I2S controller and disable I2S clock.

Include

Driver/DrvI2S.h

Return Value

None

Example

DrvI2S_ Close ( ); /* Disable I2S */

DrvI2S_EnableInt

Prototype

int32_t DrvI2S_EnableInt (E_I2 S_CALLBACK_TYPE Type, I2S_CALLBACK callbackfn);

Description

To enable I2S interrupt function and install relative call back function in I2S interrupt handler.

Parameter

Type [in]

There are eight types for call back function.

I2S_RX_UNDERFLOW: Rx FIFO underflow

I2S_RX_OVERFLOW: Rx FIFO overflow.

I2S_RX_FIFO_THRESHOLD: Data word in Rx FIFO is higher than Rx thresho ld level.

I2S_TX_UNDERFLOW: Tx FIFO underflow.

I2S_TX_OVERFLOW: Tx FIFO overflow

I2S_TX_FIFO_THRESHOLD: Data word in Tx FIFO is less than Tx threshold level.

I2S_TX_RIGHT_ZERO_CROSS: Tx right channel zero cross.

I2S_TX_LEFT_ZERO_CROSS: Tx left chan nel zero cross.

callbackfn [in]

Call back function name for specified interrupt event.

Include

Driver/DrvI2S.h

Return Value

0: Succeed

others: Failed

Example

```
/* Enable Rx threshold level interrupt and install its callback function */
DrvI2S_EnableInt(I2S_RX_FIFO_THRESHOLD,
Rx_thresholdCallbackfn);
/* Enable Tx threshold level interrupt and install its callback function */
DrvI2S_EnableInt(I2S_TX_FIFO_THRESHOLD,Tx_
thresholdCallbackfn);
```

DrvI2S_DisableInt

Prototype

int32_t DrvI2S_DisableInt (E_I2S_CALLBACK_TYPE Type);

Description

To disable I2S interrup t function and uninstall relative call back function in I2S interrupt handler.

Parameter

Type [in]

There are eight types for call back function.

I2S_RX_UNDERFLOW: Rx FIFO underflow

I2S_RX_OVERFLOW: Rx FIFO overflow

I2S_RX_FIFO_THRESHOLD: Data word in Rx FIFO is higher than Rx threshold level.

I2S_TX_UNDERFLOW: Tx FIFO underflow.

I2S_TX_OVERFLOW: Tx FIFO overflow

I2S_TX_FIFO_THRESHOLD: Data word in Tx FIFO is less than Tx threshold level.

I2S_TX_RIGHT_ZERO_CROSS: Tx right channel zero cross.

I2S_TX_LEFT_ZERO_CROSS: Tx left chan nel zero cross.

Include

Driver/DrvI2S.h

Return Value

0: Succeed

others: Failed

Example

```
/* Disable Rx threshold level interrupt and uninstall its callback function
*/
DrvI2S_DisableInt (I2S_RX_FIFO_THRESHOLD);
/* Disable Tx threshold level interrupt and uninstall its callback function
*/
DrvI2S_DisableInt (I2S_TX_FIFO_THRESHOLD);
```

DrvI2S_GetBCLK

Prototype

```
uint32_t DrvI2S_GetBCLK (void);
```

Description

To get the I2S BCLK (Bit Clock) frequency.

BCLK = I2S source clock / (2 x (BCLK divider + 1))

Parameter

None

Include

Driver/DrvI2S.h

Return Value

I2S BCLK frequency. The unit is Hz.

Example

```
uint32_t u32clock;
/* Get I2S BCLK clock frequency */
u32clock = DrvI2S_GetBCLK ( );
```

DrvI2S_SetBCLK

Prototype

void DrvI2S_SetBCLK (uint32_t u32Bclk);

Description

To configure BCLK (Bit Clock) clock. The BCLK will work when I2S operates in master mode. BCLK = I2S source clock / (2 x BCLK divider + 1))

Parameter

u32Bclk [in]

I2S BCLK frequency. The unit is Hz.

Include

Driver/DrvI2S.h

Return Value

None

Example

/* Set I2S BCLK clock frequency 512 KHz */

DrvI2S_SetBCLK (512000);

DrvI2S_GetMCLK

Prototype

uint32_t DrvI2S_GetMCLK (void);

Description

To get the I2S MCLK (Master Clock) frequency.

MCLK = I2S source c lock / (2 x MCLK divider))

Parameter

None

Include

Driver/DrvI2S.h

Return Value

I2S MCLK frequency. The unit is Hz.

Example

```
uint32_t u32clock;
/* Get I2S MCLK clock frequency */
u32clock = DrvI2S_GetMCLK( );
```

DrvI2S_SetMCLK

Prototype

void DrvI2S_SetMCLK (uint32_t u32Mclk);

Description

To configure MCLK (Master Clock) clock.

MCLK = I2S source c lock / (2 x (MCLK divider))

Parameter

u32Mclk [in]

I2S MCLK frequency. The unit is Hz.

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* Set I2S MCLK clock frequency 12MHz */
DrvI2S_SetMCLK (12000000);
```

DrvI2S_EnableZeroCrossDetect

Prototype

int32_t DrvI2S_EnableZeroCrossDetect (E_I2S_CHANNEL channel,

int32_t i32flag);

Description

To enable or disable right/left channel zero cross detect function.

Parameter

channel [in]

I2S_LEFT_CHANNEL / I2S_RIGHT_CHANNEL

i32flag [in]

To enable or disable zero cross detect function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

0: Succeed

others: Failed

Example

```
/* Enable left channel zero cross detect */
DrvI2S_ EnableZeroCrossDetect (I2S_LEFT_CHANNEL, 1);
/* Disable right channel zero cross detect */
DrvI2S_ EnableZeroCrossDetect (I2S_RIGHT_CHANNEL, 0);
```

DrvI2S_EnableTxDMA

Prototype

void DrvI2S_EnableTxDMA (int32_t i32flag);

Description

To enable/disable I2S Tx DMA function. I2S requests DMA to transfer data to Tx FIFO.

Parameter

i32flag [in]

To enable or disable Tx DMA function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* Enable I2S Tx DMA function */
DrvI2S_EnableTxDMA (1);
```

DrvI2S_EnableRxDMA

Prototype

```
void        DrvI2S_EnableRxDMA (int32_t i32flag);
```

Description

To enable/disable I2S Rx DMA function. I2S requests DMA to transfer data from Rx FIFO.

Parameter

i32flag [in]

To enable or disable Rx DMA function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* Enable I2S Rx DMA function */
DrvI2S_EnableRxDMA (1);
```

DrvI2S_EnableTx

Prototype

```
void DrvI2S_EnableTx(int32_t i32flag);
```

Description

To enable/disable I2S Tx function.

Parameter

i32flag [in]

To enable or disable Rx DMA function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* Enable I2S Tx function */
DrvI2S_EnableTx (1);
```

DrvI2S_EnableRx

Prototype

```
void   DrvI2S_EnableRx(int32_t i32flag);
```

Description

To enable/disable I2S Rx function.

Parameter

i32flag [in]

To enable or disable Rx DMA function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* Enable I2S Rx function */
```

DrvI2S_EnableRx (1);

DrvI2S_EnableTxMute

Prototype

void DrvI2S_EnableTxMute(int32_t i32flag);

Description

To enable/disable I2S Tx Mute function.

Parameter

i32flag [in]

To enable or disable Rx DMA function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

None

Example

/* Enable I2S Tx Mute function */

DrvI2S_EnableTxMute (1);

DrvI2S_EnableMCLK

Prototype

void DrvI2S_EnableMCLK(int32_t i32flag);

Description

To enable/disable I2S MCLK output from GPIOA Pin15.

Parameter

i32flag [in]

To enable or disable Rx DMA function. (1: enable 0: disable)

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* Enable MCLK output */
DrvI2S_EnableMCLK (1);
```

DrvI2S_ClearTxFIFO

Prototype

```
void      DrvI2S_ClearTxFIFO (void);
```

Description

To clear Tx FIFO. The internal pointer of Tx FIFO is reset to start point.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

None

Example

```
DrvI2S_ClearTxFIFO ( );     /* Clear Tx FIFO */
```

DrvI2S_ClearRxFIFO

Prototype

```
void      DrvI2S_ClearRxFIFO (void);
```

Description

To clear Rx FIFO. The internal pointer of Rx FIFO is reset to start point.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

None

Example

```
DrvI2S_ClearRxFIFO ( );     /* Clear Rx FIFO */
```

DrvI2S_SelectClockSource

Prototype

```
void      DrvI2S_SelectClockSource (uint8_t u8ClkSrcSel);
```

Description

To select I2S clock source, including external 12M, PLL clock, HCLK and internal 22M.

Parameter

u8ClkSrcSel [in]

To select I2S clo ck source. There are four sources for I2S:

DRVI2S_EXT_ 12M: external 12MHz crystal clock

DRVI2S_PLL: PLL clock

DRVI2S_HCLK: HCLK.

DRVI2S_INTERNAL_22M: internal 22MHz oscillator clock

Include

Driver/DrvI2S.h

Return Value

None

Example

```
/* I2S clock source from external 12M */
DrvI2S_SelClockSource (DRVI2S_EXT_12M);
/* I2S clock source from PLL clock */
DrvI2S_SelClockSource (DRVI2S_PLL);
/* I2S clock source from HCLK */
DrvI2S_SelClockSource (DRVI2S_HCLK);
```

DrvI2S_GetSourceClock

Prototype

uint32_t DrvI2S_GetSourceClockFreq (void);

Description

To get I2S source clock frequency.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

I2S clock source frequency. The unit is Hz.

Example

```
uint32_t u32clock;
/* Get I2S source clock frequency */
u32clock = DrvI2S_GetSourceClock ( );
```

DrvI2S_GetVersion

Prototype

uint32_t DrvI2S_GetVersion (void);

Description

Get this module’s version.

Parameter

None

Include

Driver/DrvI2S.h

Return Value

Version number:

31:24	23:16	15:8	7:0
00000000	MAJOR_NUM	MINOR_NUM	BUILD_NUM

I2S_IRQHandler

Prototype

void I2S_IRQHandler(void)

Description

Install ISR to handle interrupt event.

Parameter

None

Include

Driver/ DrvI2S.h

Return Value

None.

2. REVISION HISTORY

Version	Date	Description
1.00.01	Mar. 2011	Preliminary I2S Driver User Guide of ISD9160