APPLICATION NOTE

AT07908: SAM4L Inter-IC Sound Controller (IISC)

ASF PROGRAMMERS MANUAL

SAM4L Inter-IC Sound Controller (IISC)

This driver for SAM4L devices provides an interface for the configuration and management of the device's Inter-IC Sound Controller (IISC) module.

The IISC provides a 5-wire, bidirectional, synchronous, digital audio link with off-chip audio devices.

The outline of this documentation is as follows:

Table of Contents

Software License

1. Prerequisites

There are no prerequisites for this module.

2. Module Overview

The Inter-IC Sound Controller (IISC) provides a 5-wire, bidirectional, synchronous, digital audio link with external audio devices.

This module is compliant with the Inter-IC Sound (I^2S) bus specification.

The IISC consists of a Receiver, a Transmitter, and a common Clock Generator, that can be enabled separately, to provide Master, Slave, or Controller modes with Receiver, Transmitter, or both active.

The IISC can use either a single DMA channel for both audio channels or one DMA channel per audio channel. This allows high bitrate data transfer without processor intervention to the following external devices:

- Audio CODECs in Master, Slave, or Controller mode
- Stereo DAC or ADC through dedicated I^2S serial interface

The 8- and 16-bit compact stereo format allows the required DMA bandwidth to be reduced by transferring the left and right samples within the same data word.

3. Special Considerations

- Power Management

  If the CPU enters a low power sleep mode that disables clocks used by the IISC, the IISC will stop functioning and resume operation after the system wakes up from sleep mode.

  The clock for the IISC bus interface is generated by the Power Manager. It is recommended to disable the IISC before disabling the clock in order to avoid freezing the IISC in an undefined state.

- DMA

  The IISC DMA handshake interfaces are connected to the Peripheral DMA Controller. Using the IISC DMA functionality requires the Peripheral DMA Controller (PDCA) to be configured first.

- Interrupts

  The IISC interrupt line is connected to the device's Interrupt Controller. Using the IISC interrupt requires the Interrupt Controller to be configured first.

- Debug Operation

  When an external debugger forces the CPU into debug mode, the IISC continues normal operation. If the IISC module is configured such that it requires periodically servicing by the CPU through interrupt requests (or similar) then improper operation or data loss may result.

4. Extra Information

For extra information, see Extra Information for the Inter-IC Sound Controller. This includes:

- Acronyms
- Dependencies
- Errata
- Module History

5. Examples

For a list of examples related to this driver, see Examples for the Inter-IC Sound Controller.

6. API Overview

6.1 Variable and Type Definitions

6.1.1 Type iis_callback_t

```
typedef void(* iis_callback_t )(void)
```

Callback function prototype.

6.1.2 Type iis_interrupt_source_t

```
typedef enum iis_interrupt_source iis_interrupt_source_t
```

IISC interrupt source ID type.

6.2 Structure Definitions

6.2.1 Struct iis_config

Configuration setting structure.

Table 6-1. Members

Type	Name	Description
enum iis_data_format	data_format	Number of bits per sample.
enum iis_fs_rate	fs_ratio	Master Clock to Sample Frequency (fs) Ratio.
bool	loopback	1 for loopback, 0 for normal.
bool	master	1 for master, 0 for slave.
enum iis_number_of_channels	rx_channels	Number of channels for Rx.
enum iis_dma_channel	rx_dma	DMA channel usage for IIS reception.
enum iis_slot_length	slot_length	Number of bits of the slot.
enum iis_number_of_channels	tx_channels	Number of channels for Tx.
enum iis_dma_channel	tx_dma	DMA channel usage for IIS transmission.

6.2.2 Struct iis_dev_inst

Table 6-2. Members

Type	Name	Description
struct iis_config *	cfg	Pointer to IIS configuration structure.
Iisc *	hw_dev	Base Address of the IISC module.

6.3 Function Definitions

6.3.1 Function iis_clear_status()

Clear the IISC interrupt status.

```
void iis_clear_status(
  struct iis_dev_inst *const dev_inst,
  iis_interrupt_source_t source)
```

Table 6-3. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	source	Interrupt source ID

6.3.2 Function iis_disable()

Disable the IIS module.

```
void iis_disable(
  struct iis_dev_inst *const dev_inst)
```

Table 6-4. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.3 Function iis_disable_clocks()

Disable the IISC module clocks.

```
void iis_disable_clocks(
  struct iis_dev_inst * dev_inst)
```

Table 6-5. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.4 Function iis_disable_interrupt()

Disable the specified IISC interrupt source.

```
void iis_disable_interrupt(
  struct iis_dev_inst *const dev_inst,
  iis_interrupt_source_t source)
```

Table 6-6. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Data direction	Parameter name	Description
[in]	source	Interrupt source ID

6.3.5 Function iis_disable_reception()

Disable the IISC module receiver.

```
void iis_disable_reception(
  struct iis_dev_inst * dev_inst)
```

Table 6-7. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.6 Function iis_disable_transmission()

Disable the IISC module transmitter.

```
void iis_disable_transmission(
  struct iis_dev_inst * dev_inst)
```

Table 6-8. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.7 Function iis_enable()

Enable the IISC module.

```
void iis_enable(
  struct iis_dev_inst *const dev_inst)
```

Table 6-9. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.8 Function iis_enable_clocks()

Enable the IIS module clocks.

```
void iis_enable_clocks(
  struct iis_dev_inst * dev_inst)
```

Table 6-10. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.9 Function iis_enable_interrupt()

Enable the specified IISC interrupt source.

```
void iis_enable_interrupt(
  struct iis_dev_inst *const dev_inst,
  iis_interrupt_source_t source)
```

Table 6-11. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	source	Interrupt source ID

6.3.10 Function iis_enable_reception()

Enable the IISC module receiver.

```
void iis_enable_reception(
  struct iis_dev_inst * dev_inst)
```

Table 6-12. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.11 Function iis_enable_transmission()

Enable the IISC module transmitter.

```
void iis_enable_transmission(
  struct iis_dev_inst * dev_inst)
```

Table 6-13. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.12 Function iis_get_config_defaults()

Get the default IIS module configuration.

```
void iis_get_config_defaults(
  struct iis_config *const cfg)
```

Data Type	Format
Data format	32 bits
Slot length	32 bits
Sample frequency ratio	1024

Data Type	Format
Tx channel	Stereo
Rx channel	Stereo
DMA for Tx	One DMA channel for One IIS Tx channel
DMA for Rx	One DMA channel for One IIS Rx channel
Loopback	No
Master	Yes

Table 6-14. Parameters

Data direction	Parameter name	Description
[out]	cfg	Pointer to an IISC configuration structure

6.3.13 Function iis_get_interrupt_mask()

Get the IISC interrupt mask value.

```
uint32_t iis_get_interrupt_mask(
  struct iis_dev_inst *const dev_inst)
```

Table 6-15. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Returns Interrupt mask value.

6.3.14 Function iis_get_status()

Get the IISC status value.

```
uint32_t iis_get_status(
  struct iis_dev_inst * dev_inst)
```

Table 6-16. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

Returns IISC status value.

6.3.15 Function iis_init()

Initialize and configure the IISC module.

```
enum status_code iis_init(
```

```
  struct iis_dev_inst *const dev_inst,
  Iisc * iisc,
  struct iis_config *const cfg)
```

Table 6-17. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	iisc	Device structure pointer
[in]	cfg	Pointer to an IISC configuration structure

Table 6-18. Return Values

Return value	Description
STATUS_OK	Success

6.3.16 Function iis_read()

Read a single element of data.

```
enum status_code iis_read(
  struct iis_dev_inst * dev_inst,
  uint32_t * data)
```

Table 6-19. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[out]	*data	Pointer for received data

Table 6-20. Return Values

Return value	Description
STATUS_OK	Success
STATUS_ERR_TIMEOUT	Operation timed out

6.3.17 Function iis_reset()

Reset the IISC module.

```
void iis_reset(
  struct iis_dev_inst * dev_inst)
```

Table 6-21. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer

6.3.18 Function iis_set_callback()

Set an IISC callback.

```
void iis_set_callback(
  struct iis_dev_inst *const dev_inst,
  iis_interrupt_source_t source,
  iis_callback_t callback,
  uint8_t irq_level)
```

Table 6-22. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	source	Interrupt source ID
[in]	callback	IISC callback function pointer
[in]	irq_level	Interrupt level

6.3.19 Function iis_write()

Write a single element of data.

```
enum status_code iis_write(
  struct iis_dev_inst * dev_inst,
  uint32_t data)
```

Table 6-23. Parameters

Data direction	Parameter name	Description
[in]	dev_inst	Driver structure pointer
[in]	data	The data element to write

Table 6-24. Return Values

Return value	Description
STATUS_OK	Success
STATUS_ERR_TIMEOUT	Operation timed out

6.4 Enumeration Definitions

6.4.1 Enum iis_data_format

IISC data format.

Table 6-25. Members

Enum value	Description
IIS_DATE_32BIT	32 bit mono or stereo format.
IIS_DATE_24BIT	24 bit mono or stereo format.
IIS_DATE_20BIT	20 bit mono or stereo format.
IIS_DATE_18BIT	18 bit mono or stereo format.
IIS_DATE_16BIT	16 bit mono or stereo format.
IIS_DATE_16BIT_COMPACT	16 bit compact stereo format, with left and right samples packed in the same word to reduce data transfers.

Enum value	Description
IIS_DATE_8BIT	8 bit mono or stereo format.
IIS_DATE_8BIT_COMPACT	8 bit compact stereo format, with left and right samples packed in the same word to reduce data transfers.

6.4.2 Enum iis_dma_channel

DMA channel usage for an IISC data transfer type.

Table 6-26. Members

Enum value	Description
IIS_ONE_DMA_CHANNEL_FOR_BOTH_CHANNELS	One DMA channel for both channels.
IIS_ONE_DMA_CHANNEL_FOR_ONE_CHANNEL	One DMA channel for one channel.

6.4.3 Enum iis_fs_rate

Supported Master Clock to Sample Frequency (fs) Ratio.

Table 6-27. Members

Enum value	Description
IIS_FS_RATE_16	Master Clock to Sample Frequency Ratio 16.
IIS_FS_RATE_32	Master Clock to Sample Frequency Ratio 32.
IIS_FS_RATE_48	Master Clock to Sample Frequency Ratio 48.
IIS_FS_RATE_64	Master Clock to Sample Frequency Ratio 64.
IIS_FS_RATE_96	Master Clock to Sample Frequency Ratio 96.
IIS_FS_RATE_128	Master Clock to Sample Frequency Ratio 128.
IIS_FS_RATE_192	Master Clock to Sample Frequency Ratio 192.
IIS_FS_RATE_256	Master Clock to Sample Frequency Ratio 256.
IIS_FS_RATE_384	Master Clock to Sample Frequency Ratio 384.
IIS_FS_RATE_512	Master Clock to Sample Frequency Ratio 512.
IIS_FS_RATE_768	Master Clock to Sample Frequency Ratio 768.
IIS_FS_RATE_1024	Master Clock to Sample Frequency Ratio 1024.

6.4.4 Enum iis_interrupt_source

IISC interrupt source ID type.

Table 6-28. Members

Enum value	Description
IIS_INTERRUPT_RXRDY	Receive Ready.
IIS_INTERRUPT_RXOR	Receive Overrun.
IIS_INTERRUPT_TXRDY	Transmit Ready.
IIS_INTERRUPT_TXUR	Transmit Underrun.
_IIS_INTERRUPT_SOURCE_NUM	Number of interrupt sources.

6.4.5 Enum iis_number_of_channels

Mono or stereo channel setting type.

Table 6-29. Members

Enum value	Description
IIS_CHANNEL_MONO	Mono.
IIS_CHANNEL_STEREO	Stereo.

6.4.6 Enum iis_slot_length

IISC channel data slot length type.

Table 6-30. Members

Enum value	Description
IIS_SLOT_LENGTH_8BIT	8 bit IISC channel data slot length.
IIS_SLOT_LENGTH_16BIT	16 bit IISC channel data slot length.
IIS_SLOT_LENGTH_24BIT	24 bit IISC channel data slot length.
IIS_SLOT_LENGTH_32BIT	32 bit IISC channel data slot length.

7. Extra Information for the Inter-IC Sound Controller

7.1 Acronyms

Below is a table listing the acronyms used in this module, along with their intended meanings.

Acronym	Definition
ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
PDCA	Peripheral DMA Controller
QSG	Quick Start Guide

7.2 Dependencies

This driver has the following dependencies:

- I/O Lines

  The IISC pins may be multiplexed with I/O Controller lines. The user must first program the I/O Controller to assign the desired IISC pins to their peripheral function. If the IISC I/O lines are not used by the application, they can be used for other purposes by the I/O Controller. It is required to enable only the IISC inputs and outputs actually in use.

- Power Management

  If the CPU enters a sleep mode that disables clocks used by the IISC, the IISC will stop functioning and resume operation after the system wakes up from sleep mode.

- Clocks

  The clock for the IISC bus interface (CLK_IISC) is generated by the Power Manager. It is recommended to disable the IISC before disabling the clock, to avoid freezing the IISC in an undefined state. One of the generic clocks is connected to the IISC. The generic clock (GCLK_IISC) can be set to a wide range of frequencies and clock sources. The GCLK_IISC must be enabled and configured before use.

- DMA

  The IISC DMA handshake interfaces are connected to the Peripheral DMA Controller. Using the IISC DMA functionality requires the Peripheral DMA Controller to be programmed first.

- Interrupts

  The IISC interrupt line is connected to the Interrupt Controller. Using the IISC interrupt requires the Interrupt Controller to be programmed first.

- Debug Operation

  When an external debugger forces the CPU into debug mode, the IISC continues normal operation. If this module is configured in a way that requires it to be periodically serviced by the CPU through interrupt requests or similar, improper operation or data loss may result during debugging.

7.3 Errata

There are no errata related to this driver.

7.4 Module History

An overview of the module history is presented in the table below, with details on the enhancements and fixes made to the module since its first release. The current version of this corresponds to the newest version in the table.

Changelog
Initial document release

8. Examples for the Inter-IC Sound Controller

This is a list of the available Quick Start Guides (QSGs) and example applications for SAM4L Inter-IC Sound Controller (IISC). QSGs are simple examples with step-by-step instructions to configure and use this driver in a selection of use cases. Note that QSGs can be compiled as a standalone application or be added to the user application.

- Quick Start Guide for the SAM4L IIS driver

8.1 Quick Start Guide for the SAM4L IIS driver

This is the quickstart guide for the SAM4L Inter-IC Sound Controller driver, with step-by-step instructions on how to configure and use the driver in a selection of use cases.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

8.1.1 Basic Use Case

This use case will demonstrate how to initialize the IIS module in Master mode with loopback enabled.

8.1.1.1 Prerequisites

- System Clock Management (Sysclock)

8.1.2 Clock Setup Steps

8.1.2.1 Example Code

Enable the following macros in the header file conf_clock.h:

```
#define CONFIG_SYSCLK_SOURCE       SYSCLK_SRC_DFLL
#define CONFIG_DFLL0_SOURCE        GENCLK_SRC_OSC32K
```

Add the following code to your application C-file:

```
sysclk_init();
```

8.1.2.2 Workflow

1. Set the system clock source to DFLL:

   ```
   #define CONFIG_SYSCLK_SOURCE       SYSCLK_SRC_DFLL
   ```

2. Set Set the DFLL clock source to OSC32K:

   ```
   #define CONFIG_DFLL0_SOURCE        GENCLK_SRC_OSC32K
   ```

3. Initialize the system clock:

   ```
   sysclk_init();
   ```

8.1.3 Usage Steps

8.1.3.1 Example Code

Add to, e.g., main loop in your application C-file:

```
uint32_t data = 0;
```

```
struct iis_config config;
struct iis_dev_inst dev_inst;
config.data_format = IIS_DATE_16BIT_COMPACT;
config.slot_length = IIS_SLOT_LENGTH_16BIT;
config.fs_ratio = IIS_FS_RATE_256;
config.tx_channels = IIS_CHANNEL_STEREO;
config.rx_channels = IIS_CHANNEL_STEREO;
config.loopback = true;
config.master = true;

if (iis_init(&dev_inst, IISC, &config) != STATUS_OK)
{
    return -1;
}
iis_enable(&dev_inst);
iis_enable_transmission(&dev_inst);
iis_enable_clocks(&dev_inst);
iis_enable_reception(&dev_inst);
if (iis_write(&dev_inst, data) != STATUS_OK)
{
    return -1;
}
if (iis_read(&dev_inst, &data) != STATUS_OK)
{
    return -1;
}
```

8.1.3.2 Workflow

1. Initialize the IISC module with a given configuration:

```
uint32_t data = 0;
struct iis_config config;
struct iis_dev_inst dev_inst;
config.data_format = IIS_DATE_16BIT_COMPACT;
config.slot_length = IIS_SLOT_LENGTH_16BIT;
config.fs_ratio = IIS_FS_RATE_256;
config.tx_channels = IIS_CHANNEL_STEREO;
config.rx_channels = IIS_CHANNEL_STEREO;
config.loopback = true;
config.master = true;

if (iis_init(&dev_inst, IISC, &config) != STATUS_OK)
{
    return -1;
}
```

2. Enable the IISC module:

```
iis_enable(&dev_inst);
```

3. Enable transmitter, receiver, and clocks:

```
iis_enable_transmission(&dev_inst);
iis_enable_clocks(&dev_inst);
iis_enable_reception(&dev_inst);
```

4. Use write/read functions to transfer data:

```
if (iis_write(&dev_inst, data) != STATUS_OK)
{
    return -1;
}
if (iis_read(&dev_inst, &data) != STATUS_OK)
{
    return -1;
}
```

Index

Document Revision History

Doc. Rev.	Date	Comments
42315A	05/2014	Initial document release

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | www.atmel.com