AVR32788: AVR®32 How to use the SSC in I2S mode

Features

- **I²S protocol overview**
- **I²S on the AVR32**
- **I²S sample rate configurations**
- **Example of use with AT32UC3A on EVK1105 board**

1 Introduction

This application note describes how the I²S protocol is handled on AVR32 devices and gives important information about how to get the best configuration for different sample rates.

32-bit AVR® Microcontrollers

Application Note

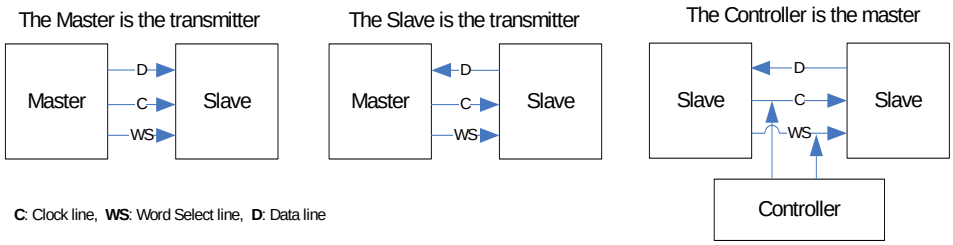
Rev. 32127A-AVR32-06/09

2 I²S protocol overview

Inter-IC Sound bus (I²S) is a serial bus interface that is used to transport digital sound samples between different devices like A/D converters or digital signal processors.

The protocol uses 3 lines to transport the data from a device to another (**clock** line, **data** line and **word select** line). As in most communication protocols, this protocol needs a master and a slave to operate. The role of the master is to provide the clock and word select lines.

Figure 2-1 I²S protocol signal routing

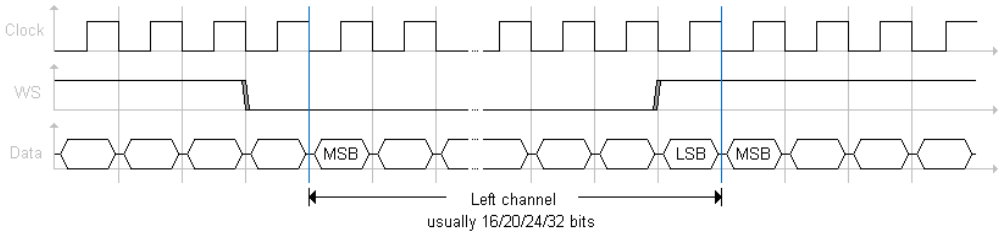


The **clock** is operating at a frequency which is a multiple of the sample rate of the audio stream. This frequency depends also on the number of bits per sample and the number of channels. *For example, a stereo audio stream with samples using 16 bits of precision and with a sampling frequency of 44.1KHz would have to use a clock rated at 1,4112 MHz to transport this audio stream. (2 (stereo) x 16 (precision) x 44100 (Fs) = 1411200 Hz).*

The **data** is fed on the data line at the rate of the clock line with one bit clock delay. It is transmitted in Little-Endian format (MSB first) and there is no limitation on the sample length but usually it uses a 16/20/24 or 32 bits format.

The **word select** line is used as a control signal. It tells the slave which channel the sample is designed for (left or right channel). During the transmission of one sample this line is either high or low with one clock ahead. This allows both the transmitting and receiving devices to have different sample length but also to not care about the audio precision of the remote device.

Figure 2-2 I²S protocol waveforms



Note that the word select line can have a different length (in term of bit clock), than the data resolution (sample precision). If the word select's length is smaller, then the

data will be truncated and only the most significant bits will be kept on the data line. In opposite, if the word select's length is greater, the rest of the data bit will be fed with zeros, which will automatically cast the sample precision to the appropriate resolution thanks to the endianness of the data. This feature makes it possible to mix and match components of varying precision without reconfiguration.

Figure 2-3 I²S protocol waveforms – WS length smaller than Data length

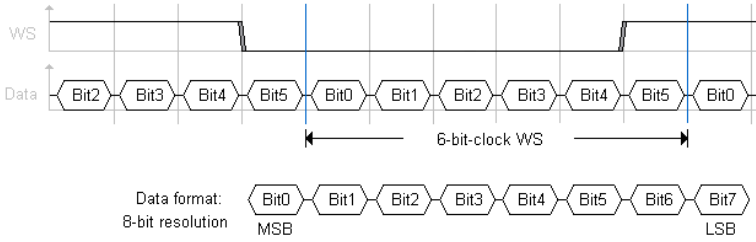
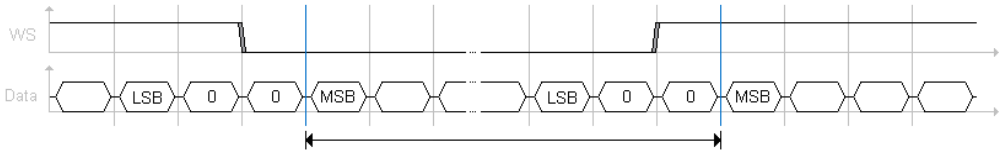


Figure 2-4 I²S protocol waveforms – WS length greater than Data length



3 I²S interface

3.1 Basic interface

The basic interface consists of three signals:

- Bit-clock
- Data line:
- Word select line: This line has the frequency of the sample rate since it toggles between the left and the right channel data.

This interface can be provided by the SSC module on AVR32 devices.

3.2 Systems with master clock

In addition to the basic I²S interface very often a master clock is required that must run at a multiple of the sample rate. Common ratios are e.g. 256 and 384.

The master clock must be synchronous to the other signals of the I²S interface but there is usually no phase relationship. This means that the clock can be generated by other hardware modules that have no relation to the I²S protocol handling but they their clock must be synchronous.

4 I²S on AVR32

4.1 Basic I²S interface

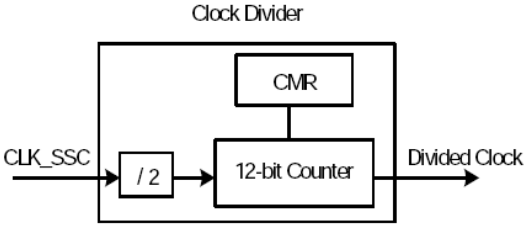
The basic I²S interface can be emulated by the SSC module on the AVR32. This module is capable to handle the I²S protocol in hardware.

4.1.1 Bit-clock generation

The bit-clock for the I²S protocol is generated in the SSC module and is based on a counter. The input to the counter can be an internal or external source. The internal source is provided by the bus the SSC module is connected to which is e.g. PBA on the UC3A.

If the clock divider is used it divides the clock first by ½ and this is fed into the counter. The pre-scaling is done to ensure a 50% duty cycle.

Figure 4-1 SSC clock generation



Following relative error rates are expected if a clock of 62.0928 MHz (5.5 * 11.2896MHz) is used as the input to the SSC module and the stereo samples are 16-bits.

Table 4-1 Relative error rates for the bit-clock

Sample rate (Hz)	Bit-clock (Hz)	Actual generated bit-clock	Relative error %	Musical interval (semitones
8000	256000	256582	0.23	0.039
11025	352800	352800	0	0
16000	512000	508957	-0.59	-0.130
22050	705600	705600	0	0
32000	1024000	1034880	1.06	0.138
44100	1411200	1411200	0	0
48000	1536000	1552320	1.06	0.138
88200	2822400	2822400	0	0
96000	3072000	3104640	1.06	0.138

4.2 Generating a master clock for I²S

The SSC module that is used to emulate the I²S protocol does not provide a master clock. Because of that this clock must be generated elsewhere. Some possible clock sources are a generic clock, a timer/counter or a PWM module.

The generic clock is the recommended way to generate the master clock since it offers more options to choose from source clocks like oscillators and PLLs.

Because the master clock must be synchronous to the sample rate both clock sources to the generic clock and to the SSC module must have the same source. This inflexibility leaves to a higher relative error rate as shown in. The table assumes that the external DAC needs a master clock that is 256 times the sample rate (implies 16-bit samples) and the source clock of the SSC and the generic clock is 45.1584MHz.

Table 4-2 Relative error rates

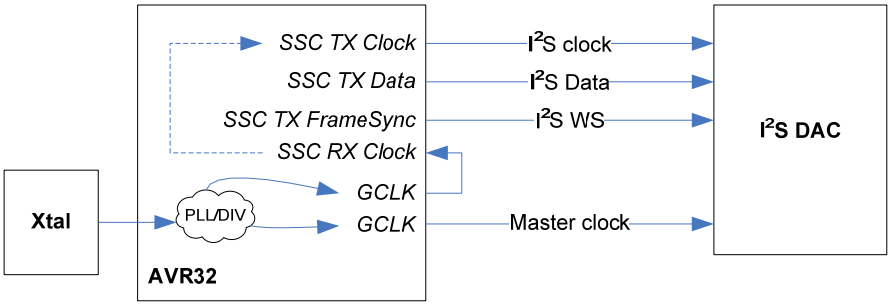
Sample rate (Hz)	Actual generated sample rate (Hz)	Relative error %	Musical interval (semitones
8000	8018	0.23	0.04
11025	11025	0	0
16000	14700	-8.13	-1.47
22050	22050	0	0
32000	29400	-8.13	-1.47
44100	44100	0	0
48000	44100	-8.13	-1.47
88200	88200	0	0
96000	88200	-8.13	-1.47

Since the SSC module derives its clock internally only from the peripheral bus, a change to other clock sources is not the best approach to provide better error rates for other sampling rates, since the whole system clock needs to be changed. This will affect most of the modules that are connected to the different system clocks and these would need a re-configuration.

4.3 Generating a master clock and an I²S clock with the same oscillator

A better solution is to output the generic clock on a pin and route the clock to the “RX-Clock” pin or even the “TX-Clock” pin (bit-clock output) of the of the SSC module. The TX-Clock pin can only be used if the external DAC device does not need the bit-clock. This approach makes the SSC module independent of the peripheral clock and now it is possible to generate sample rates from different system clocks. Note that on the AVR32, this will involve the use of 2 internal generic clocks fed by the same oscillator: one to generate the master clock and one to feed the “RX-Clock” pin.

Figure 4-2 Optimal I²S generation with the AVR32



For example a system has the following constellation:

- Oscillator 0 (OSC0) with 12MHz

- Oscillator 1 (OSC1) with 11.289600MHz
- PLL0 fed from OSC1 with 62.092MHz
- PLL1 fed from OSC0 with 48MHz

These settings are not optimum but they are available on the EVK1105 and shall serve as an example.

Table 4-2 Relative error rates for the bit-clock using 12MHz (OSC0) and 11.2896MHz (OSC1) crystals

Sample rate (Hz)	Actual generated sample rate (Hz)	Used clock	Relative error %	Musical interval (semitones
8000	8085	PLL0	1.06	0.04
11025	11025	OSC1	0	0
16000	15625	PLL1	-2.34	-0.14
22050	22050	OSC1	0	0
24000	24255	PLL0	1.06	0.04
32000	31250	PLL1	-2.34	-0.14
44100	44100	OSC1	0	0
48000	46875	PLL1	-2.34	-0.14

By replacing the 12MHz crystal by a 14.7456MHz crystal, the settings are optimum for common sampling rates. Then the oscillators' configuration becomes:

- Oscillator 0 (OSC0) with 14.7456MHz
- Oscillator 1 (OSC1) with 11.289600MHz
- PLL0 fed from OSC0 with 110.592 MHz
- PLL1 fed from OSC0 with 47.9232MHz

Table 4-3 Relative error rates for the bit-clock using 14.7456MHz (OSC0) and 11.2896MHz (OSC1) crystals

Sample rate (Hz)	Actual generated sample rate (Hz)	Used clock	Relative error %	Musical interval (semitones
8000	8000	PLL0	0	0
11025	11025	OSC1	0	0
16000	16000	PLL0	0	0
22050	22050	OSC1	0	0
24000	24000	PLL0	0	0
32000	32000	PLL0	0	0
44100	44100	OSC1	0	0
48000	48000	PLL0	0	0

Note that with this configuration, the USB can be fed with a 47.9232MHz clock signal using PLL1. The closet achievable CPU frequency which can be generated is 55.296MHz.

4.4 Software configuration

4.4.1 Using the I²S driver from the Software Framework

On the AVR32, the SSC peripheral is used to emulate the I²S protocol. The SSC to I²S driver provides a basic interface to initialize, control and send a sample through the serial bus line. This driver is located in the Software Framework under **/DRIVERS/SSC/I2S/**.

At first the module has to be initialized with the function ***ssc_i2s_init***. This function takes in parameters the *sample frequency* of the audio stream (usually 32000/44100 or 48000Hz), the *data bit resolution*, also called the sample precision (usually 16/20/24 or 32 bits), the *frame bit resolution* which is the word select bit clock count (usually the same length as the data bit resolution) and also the PBA frequency.

An I²S mode must also be specified to configure the SSC module and it can be one of the following modes:

- *SSC_I2S_MODE_STEREO_OUT*: Two output channels.
- *SSC_I2S_MODE_STEREO_OUT_EXT_CLK*: Two output channels sampled with an external clock received from the “RX Clock” line. For this particular mode, the PBA frequency passed in parameter to the *ssc_i2s_init* function will be ignored.
- *SSC_I2S_MODE_SLAVE_STEREO_OUT*: Two output channels controlled by the DAC.
- *SSC_I2S_MODE_STEREO_OUT_MONO_IN*: Two output, one input channel. This mode uses two I²S buses.
- *SSC_I2S_MODE_RIGHT_IN*: Right channel in. Used because one SSC only can manage one input channel at a time.

A simple API can be used to send a sample on the I²S bus, **ssc_i2s_transfer**, but it is recommended to use a PDCA channel in order to free CPU resources. This can be done using the following routine:

```
// Disable interrupts
if ((global_interrupt_enabled = Is_global_interrupt_enabled()))
     Disable_global_interrupt();
// Wait for the next frame synchronization event
// to avoid channel inversion if the previous PDCA transfer
// is already completed
if (pdca_get_transfer_status(SSC_TX_PDCA_CHANNEL) &
    PDCA_TRANSFER_COMPLETE)
{
     while (gpio_get_pin_value(SSC_TX_FRAME_SYNC_PIN));
     while (!gpio_get_pin_value(SSC_TX_FRAME_SYNC_PIN));
}
// Load the data to be transferred onto the PDCA channel
pdca_reload_channel(SSC_TX_PDCA_CHANNEL, data, sizeof(data));
// Re-enable the interrupts
if (global_interrupt_enabled)
     Enable_global_interrupt();
```

For a complete list of the functions provided by this driver, please refer directly to the **ssc_i2s.h** file.

4.4.2 Configuring the I²S on the EVK1105 board

The EVK1105 board uses a TLV320AIC23B I²S audio codec to output the audio stream.

By default, the audio player application (please refer to the application note AVR32709 for more information) is configured to use the PBA frequency in order to generate the I²S clock signal while a generic clock is used to generate the master clock signal to feed the I²S audio codec.

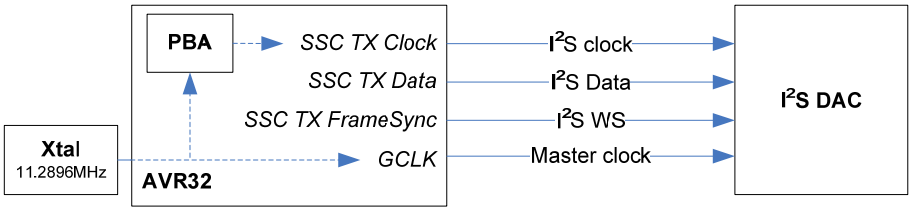
4.4.2.1 Using the TLV320AIC23B codec with the 11.2896MHz crystal

This is the default configuration. It provides optimal settings to the codec to handle 44.1 KHz sampling rates. All the other main sampling rates can be generated with a low relative error rate (see "*Table 4-3 Relative error rates for the bit-clock*" for more details).

This mode is set by using the following configuration (in /APPLICATIONS/EVK1105-AUDIO-PLAYER-MASS-STORAGE/):

- **./CONF/conf_tlv320aic23b.h**
  - **AIC23B_MCLK_HZ** **11289600**
  - **AIC23B_DAC_USE_RX_CLOCK** **DISABLED**
- **Use ./CLOCKS/clocks_fosc0_12000000_fosc1_11289600.c**

Figure 4-3 EVK1105 default I²S clock configuration



Note that on the audio player application, this mode still requires the 12MHz crystal to support the USB protocol.

4.4.2.2 Using the TLV320AIC23B codec in USB mode with only one crystal

The TLV320AIC23B audio codec also provides a USB mode interface, which means it can be used directly with a 12MHz crystal. The main sampling frequency are then supported by this mode but gives some small error on non-12MHz-multiple frequencies such as the 44.1KHz. Therefore this mode is not optimal to play such sampling rates but it has the advantage to use only one 12MHz crystal which can also be used to feed other modules such as the USB IP.

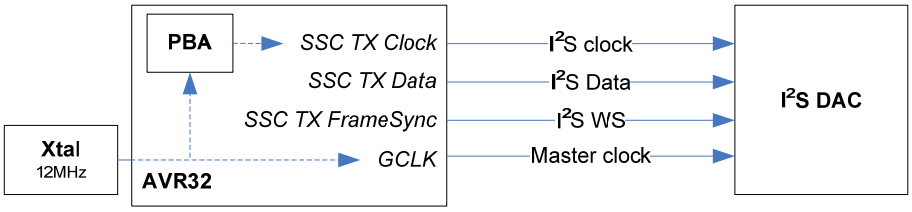
This mode is set by using the following configuration (in /APPLICATIONS/EVK1105-AUDIO-PLAYER-MASS-STORAGE/):

- **./CONF/conf_tlv320aic23b.h**

- **AIC23B_MCLK_HZ** **12000000**
- **AIC23B_DAC_USE_RX_CLOCK** **DISABLED**

- **Use ./CLOCKS/clocks_fosc0_12000000.c**

Figure 4-4 The TLV320AIC23B audio codec used in USB mode with 1 crystal



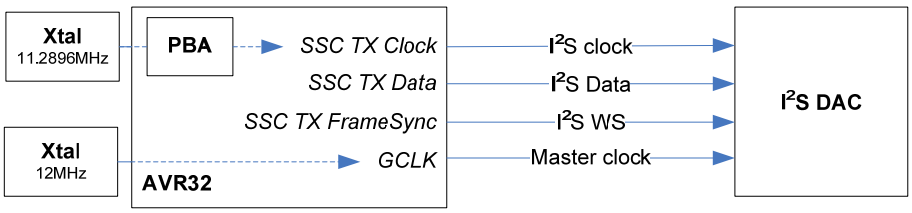
4.4.2.3 Using the TLV320AIC23B codec in USB mode with two crystals

This mode combines the advantage of the USB mode by supporting every main sample frequency and of the 11.2896MHz crystal to generate accurate sampling rates.

This mode is set by using the following configuration (in /APPLICATIONS/EVK1105-AUDIO-PLAYER-MASS-STORAGE/):

- **./CONF/conf_tlv320aic23b.h**
  - **AIC23B_MCLK_HZ** **12000000**
  - **AIC23B_DAC_USE_RX_CLOCK** **DISABLED**
- **Use ./CLOCKS/clocks_fosc0_12000000_fosc1_11289600.c**

Figure 4-5 The TLV320AIC23B audio codec used in USB mode with 2 crystals



4.4.2.4 Switching clock source with the TLV320AIC23B codec

Some DAC don't have this USB mode feature to handle multiple frequency rates with the same oscillator. Therefore the application must be able to switch clocks while it is running.

The audio player application on the EVK1105 implements this feature by using 2 generic clocks to generate both the master clock and the "TX Clock". Please read the "Generating a master clock and an I²S clock with the same oscillator" section for more information.

This mode is set by using the following configuration (in /APPLICATIONS/EVK1105-AUDIO-PLAYER-MASS-STORAGE/):

- **./CONF/conf_tlv320aic23b.h**
  - **AIC23B_DAC_USE_RX_CLOCK** **ENABLED**
- **Use one of the following files (depending on the hardware configuration):**

- **./CLOCKS/clocks_fosc0_12000000_fosc1_11289600.c**
- **./CLOCKS/clocks_fosc0_14745600_fosc1_11289600.c**

- **PB30 (pin #21) must be linked with PA18 (pin #62).**

5 Table of Contents

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
Avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature