Connecting the Atmel ARM-based Serial Synchronous Controller (SSC) to an \( \text{I}^2\text{S} \)-compatible Serial Bus

Introduction

This Application Note describes the configuration required to connect the Atmel ARM-based Synchronous Serial Controller (SSC) to a device with an \( \text{I}^2\text{S} \)-compatible serial bus, such as a stereo audio digital-to-analog converter (DAC) or a stereo audio Codec.

The digital interface of these audio devices is generally compliant with the \( \text{I}^2\text{S} \) standard. An \( \text{I}^2\text{S} \)-standard compliant device has a word length of 16 bits, as does the SSC peripheral embedded in the AT91RM9200 series microcontrollers.

\( \text{I}^2\text{S} \) (Inter-IC Sound) is a serial bus designed for digital audio devices and technologies, such as compact disc (CD) players, digital sound processors, and digital TV (DTV) sound. One of the characteristics of the \( \text{I}^2\text{S} \) protocol is the separate handling of audio data and clock signals. Separating the data and clock signals eliminates the need for anti-jitter devices by removing time-related errors.

The Application Note includes a dedicated software package for the AT91RM9200, but is applicable to all Atmel ARM-based products that embed the Synchronous Serial Controller (SSC).

This application note takes into account the SSC warnings as described in the AT91RM9200 Errata Sheet, literature number 6015.

Warranty

All delivered sources are free of charge and can be copied or modified without authorization.

The software is delivered “AS IS” without warranty or condition of any kind, either express, implied or statutory. This includes without limitation any warranty or condition with respect to merchantability or fitness for any particular purpose, or against the infringements of intellectual property rights of others.
I2S Audio Bus

The I2S (Inter-IC Sound) standard is based on a three-wire bus architecture. This standard defines a serial link dedicated to data transfer between integrated circuits in digital audio systems. This three-wire link provides additional information to audio data, such as subcoding and control, transferred separately. The three lines defined by the I2S protocol are:

- a Serial Data (SD) line containing two time-division multiplexed channels
- a left/right channel Word Select (WS)
- a continuous Serial Clock (SCK)

The I2S link is used primarily to send audio data from a processor (master) to an Audio DAC (slave). The three lines driven by an I2S master transmitter are:

- a Serial Data Out (SD Out) for two time-division multiplexed channels (from master to slave)
- a left/right channel Word Select (WS)
- a continuous Serial Clock (SCK); the I2S master (transmitter) and slave (receiver) share the same clock signal for data transmission

An additional line can be used to connect an I2S slave input signal (such as an audio Codec):

- a Serial Data In (SD In) for two time-division multiplexed channels (from slave to master)

Figure 1. I2S Block Diagram Example

```
TRANSMITTER = MASTER

I2S TRANSMITTER

Clock SCK
Word Select WS
Data SD Out
Data SD In

I2S RECEIVER

Clock SCK
Word Select WS
Data SD Out
Data SD In

Left Channel
Right Channel
```
Connecting the Atmel SSC to an I2S Bus

I2S Word Length Considerations

The I2S standard defines several possible data (word) lengths from 16 to 32 bits. For most audio applications, the data length is 16 bits, corresponding to a dynamic range of 96 dB.

Note: The dynamic range is given by the following formula:

\[
\text{Dynamic Range} = 20 \log_{10}(\text{Bit Range}) \text{ where bit range } = 2^{\text{bit}}
\]

Table 1. Bit Range versus Dynamic Range

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Dynamic Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2^{16})</td>
<td>96 dB</td>
</tr>
<tr>
<td>(2^{24})</td>
<td>144 dB</td>
</tr>
<tr>
<td>(2^{32})</td>
<td>196 dB</td>
</tr>
</tbody>
</table>

Only 16-bit words are managed by the SSC due to the maximum value of the field FSLEN of the Frame Mode Register.

I2S Clock Considerations

The sampling frequency of audio devices can vary from 8 to 48 kHz for 16-bit data.

To generate data with the correct bit rate, the SSC peripheral divides its internal peripheral clock (MCK) by an integer factor. Table 2 indicates the error on audio clock signal frequency (related to the peripheral main clock) compared to the ideal value. Due to this uncertainty, the peripheral clock value and thus the crystal frequency must be chosen carefully with respect to the limitations of the components.

The I2S bit rate determines data flow on the I2S bus and I2S clock signal frequency:

\[
\text{I2S Bit Rate} = \text{Number Bits per Channel} \times \text{Number of Channels} \times \text{Sampling Frequency}
\]

For 16-bit audio, left/right, the I2S bit rate is calculated as follows:

\[
\text{I2S Bit Rate} = 16 \times 2 \times \text{Sampling Frequency}
\]

The MCK divider factor value is set in the SSC_CMR register and equals half of the peripheral clock frequency (MCK) divided by the required bit rate. If this factor is not an integer, the real bit rate (generated by the SSC) is different from the theoretical one.

Table 2 gives the error between theoretical and real values of audio sample frequency. It is important to note that the difference between two frequency levels may result in distortion on the audio output signal.

Table 2. Using a 60 MHz MCK Clock

<table>
<thead>
<tr>
<th>Theoretical Audio Sample Frequency (Hz)</th>
<th>Theoretical Bit Rate</th>
<th>Theoretical MCK Divider Factor</th>
<th>SSC_CMR (Real MCK Divider Factor)</th>
<th>Audio Sample Frequency (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>48 000</td>
<td>1 536 000</td>
<td>39.06250</td>
<td>20 (40)</td>
<td>46 875</td>
</tr>
<tr>
<td>44 100</td>
<td>1 411 200</td>
<td>42.51701</td>
<td>21 (42)</td>
<td>44 643</td>
</tr>
<tr>
<td>22 050</td>
<td>705 600</td>
<td>85.03401</td>
<td>43 (86)</td>
<td>21 802</td>
</tr>
<tr>
<td>16 000</td>
<td>512 000</td>
<td>117.18750</td>
<td>59 (118)</td>
<td>15 890</td>
</tr>
<tr>
<td>8 000</td>
<td>256 000</td>
<td>234.37500</td>
<td>117 (234)</td>
<td>8 013</td>
</tr>
</tbody>
</table>
The following implementation using the I2S bus illustrates the DAC connection as audio output of an ARM-based microcontroller embedding the SSC and TWI peripherals.

The SSC is used and connected as an I2S processor, sending 16-bit words, a word-select signal and the serial clock. The standard audio DAC, only used as audio output, is connected as an I2S slave.

**Table 3. Hardware Connections**

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>DAC 3550A</th>
<th>Bus Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSC (TFx)</td>
<td>WSI</td>
<td>WS (I2S)</td>
</tr>
<tr>
<td>SSC (TKx)</td>
<td>CLI</td>
<td>SCK (I2S)</td>
</tr>
<tr>
<td>SSC (TDx)</td>
<td>DAI</td>
<td>SD (I2S)</td>
</tr>
<tr>
<td>TWI (TWCK)</td>
<td>SCL</td>
<td>SCL</td>
</tr>
<tr>
<td>TWI (TWD)</td>
<td>SDA</td>
<td>SDA</td>
</tr>
</tbody>
</table>
Connecting the Atmel SSC to an I2S Bus

Configuring the SSC

The following example illustrates the use of one of the SSC peripherals embedded in the AT91RM92000, the SSC1.

The related SSC I/O lines of the Parallel Input/Output Controller (PIO) are configured in peripheral mode. When using a DAC, only the three lines of the SSC emitter are used because the SSC is used as an I²S master only for emission.

The software is configured according to the following general characteristics:

- Sample audio frequency (FILE_SAMPLING_FREQ): 44.1 kHz
- Number of slots by frame (SLOT_BY_FRAME): 2 (left and right channels)
- Number of bits by slot (BITS_BY_SLOT): 16 (data length is 16 bits)
- SSC peripheral clock (MCK): 60 MHz

Note: All parameters and functions are compliant with the ARM-based Software Packages Application Note, lit. no. 6016.

Standard Initialization

The following configuration steps are common to all serial peripherals:

1. Configure the corresponding Parallel Input/Output Controller to work in peripheral mode, i.e., the three PIO lines TF, TK and TD related to the SSC1 and multiplexed with PIOB must not be in PIO management mode.

```
*AT91C_PIOB_PDR= ((unsigned int) AT91C_PB7_TK1 ) |
((unsigned int) AT91C_PB8_TD1 ) | ((unsigned int) AT91C_PB6_TF1 );
```

2. Configure the Power Management Controller to enable the current peripheral and set the PMC by enabling the SSC1 clock.

```
AT91F_SSC1_CfgPMC();
```

3. Reset the SSC1:

```
pSSC->SSC_CR = AT91C_SSC_SWRST ;
```

4. Clear Transmit and Receive PDC Counters:

```
AT91F_PDC_Close((AT91PS_PDC) &(pSSC->SSC_RPR));
```

Configuring the Clock Mode Register (SSC_CMR)

The definition of the Clock Mode Register is done for the characteristics described above (sample audio frequency, number of slots by frame, data length and peripheral clock frequency).

This setting is managed by the standard software package.

```
function AT91F_SSC_SetBaudrate:
  Bit rate= SLOT_BY_FRAME*BITS_BY_SLOT*FILE_SAMPLE_FREQ
           = 2*16*44100
           = 1.4112 MHz
  AT91F_SSC_SetBaudrate(pSSC,MCK, FILE_SAMPLING_FREQ*(BITS_BY SLOT*SLOT_BY_FRAME));
```
Configuring the Transmit Frame Mode Register (SSC_TFMR)

The Transmit Frame Mode register is used to manage both TF (corresponding to the Word Select line) and TD (corresponding to the data line) signals.

### Table 4. Transmit Frame Mode Register (SSC_TFMR) Settings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATLEN</td>
<td>BITS_BY_SLOT-1</td>
<td>Programs Data Length</td>
</tr>
<tr>
<td>DATDEF</td>
<td>0</td>
<td>Default value for data. For compatibility with the I^2^S protocol, all bits are set to zero when not sending data.</td>
</tr>
<tr>
<td>MSBF</td>
<td>1</td>
<td>Most Significant Bit First. MSB is sent first.</td>
</tr>
<tr>
<td>DATNB</td>
<td>SLOT_BY_FRAME - 1</td>
<td>Data Number per frame. Programs the number of data contained in one frame.</td>
</tr>
<tr>
<td>FSLEN</td>
<td>BITS_BY SLOT - 1</td>
<td>Frame Sync Length. Programs the duration of the active level on TF in number of serial clock cycles. The Word Select signal must be as long as one data word.</td>
</tr>
<tr>
<td>FSOS</td>
<td>Negative Pulse</td>
<td>Frame Sync Output Selection</td>
</tr>
<tr>
<td>FSDEN</td>
<td>Disabled</td>
<td>Frame Sync Data Enable. No Sync Data (SSC_TSHR) to emit.</td>
</tr>
<tr>
<td>FSEDGE</td>
<td>Positive Edge Detection</td>
<td>Frame Sync Edge Detection. Not used in this case.</td>
</tr>
</tbody>
</table>

\[
\text{*AT91C\_SSC\_TFMR = (AT91C\_SSC\_FSOS\_NEGATIVE |} \\
\text{((BITS\_BY\_SLOT-1)<<16) & AT91C\_SSC\_FSLEN) |} \\
\text{((SLOT\_BY\_FRAME-1)<<8) & AT91C\_SSC\_DATNB) |} \\
\text{AT91C\_SSC\_MSBF | (BITS\_BY\_SLOT-1) ;} \\
\]

Configuring Interrupt Mode

In this application, the Peripheral Data Controller (PDC) interrupt sources are handled to loop the sending of the wave file to the DAC.

1. Configure the Advanced Interrupt Controller (AIC) to handle SSC interrupts:

   ```c
   AT91F_AIC_ConfigureIt (AT91C_BASE_AIC,                        // AIC base address
                           AT91C_ID_SSC1,                         // System peripheral ID
                           IRQ_LEVEL_I2S,                         // Max priority
                           AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE, // Level sensitive
                           AT91F_ASM_I2S_Handler );              // AIC interrupt service routine
   ```

2. Enable the SSC interrupt in AIC:

   ```c
   AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SSC1);          // AIC interrupt enable
   ```

3. Enable SSC End Of Transmit interrupt in the SSC Interrupt Enable Register:

   ```c
   AT91F_SSC_EnableIt (pSSC, AT91C_SSC_ENDTX);                  // SSC interrupt enable
   ```
Connecting the Atmel SSC to an I2S Bus

Configuring the PDC

1. Configure the PDC:
   
   ```c
   AT91F_PDC_SetTx ((AT91PS_PDC) & (pSSC->SSC_RPR),
   (char *)wav_file, AT91C_WAV_FILE_SIZE/2);
   AT91F_PDC_SetNextTx ((AT91PS_PDC) & (pSSC->SSC_RPR),
   (char *)wav_file, AT91C_WAV_FILE_SIZE/2);
   ```

2. Enable the PDC feature:
   
   ```c
   AT91F_PDC_EnableTx ((AT91PS_PDC) & (pSSC->SSC_RPR));
   ```

Enabling the Peripheral and Starting Transmission

The Transmit Clock Mode Register must be configured before enabling and starting transmission. Then, when transmission is enabled, it starts automatically.

**Table 5. Transmit Clock Mode Register (SSC_TFMR) Settings**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKS</td>
<td>Divided Clock</td>
<td>Transmit Clock Selection. Internally generated divided clock (from peripheral clock MCK) is selected.</td>
</tr>
<tr>
<td>CKO</td>
<td>Continuous Transmit Clock</td>
<td>Clock Output Mode</td>
</tr>
<tr>
<td>CKI</td>
<td>Shifting out on serial clock falling edge</td>
<td>Transmit Clock Inversion. The data and the frame sync signals are shifted out on transmit clock falling edge.</td>
</tr>
<tr>
<td>START</td>
<td>Falling edge on TF signal</td>
<td>Transmit Start Selection. The first word to be transferred is left channel.</td>
</tr>
<tr>
<td>STTDLY</td>
<td>1</td>
<td>Transmit Start Delay. First bit after falling edge of TF signal is the last bit of right channel.</td>
</tr>
<tr>
<td>PERIOD</td>
<td>(((SLOT_BY_FRAME *BITS_BY_SLOT)/2) - 1)</td>
<td>Transmit Period Divider Selection. Frame Length, including left and right channel</td>
</tr>
</tbody>
</table>

*AT91C_SSC1_TCMR = (\(((BITS\_BY\_SLOT*SLOT\_BY\_FRAME)/2) -1\) <<24) | ((1<<16) & AT91C_SSC_STTDLY) | AT91C_SSC_START_FALL_RF | AT91C_SSC_CKO_CONTINOUS | AT91C_SSC_CKS_DIV;;

1. Enable the TX transmitter:
   
   ```c
   AT91F_SSC_EnableTx (pSSC);
   ```
Connection to a Stereo Audio Codec

The following examples illustrate the use of one or two SSC peripherals in the AT91RM9200 connecting to a stereo audio Codec. The SSC peripheral in the AT91RM9200 is able to manage two channels in output mode and one channel in input mode. This is sufficient for most applications using a Codec as generally only one channel is required in input mode. Hardware and software configurations related to this type of application are described in “Configuration for Stereo Output and Mono Input” on page 8.

When the two channels are managed in input mode (i.e., all received frame bits), the AT91RM9200 requires two SSC channels. Hardware and software implementation related to this type of application is described in “Configuration for Stereo Input and Output” on page 10.

Configuration for Stereo Output and Mono Input

To connect the Codec as stereo output and mono channel input (only left channel), the following implementation using I2S with a 16-bit data format can be used. The standard audio Codec is used in both output and input modes. The SSC, used in master mode, manages all I2S transmit signals.

Figure 3. Stereo Audio Codec with One Input Channel

Configuring the SSC

The following configuration is an example using the AT91RM9200 and one of its SSC peripherals (SSC1).

The Parallel Input/Output Controller (PIO) is configured in peripheral mode for the corresponding SSC I/O lines.

Configuring the Receive Frame Mode Register

The Receive Frame Mode Register (SSC_RFMR) is configured to manage data reception.

Table 6. Receive Frame Mode Register (SSC_RFMR) Settings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATLEN</td>
<td>BITS_BY_SLOT - 1</td>
<td>Programs Data Length</td>
</tr>
<tr>
<td>LOOP</td>
<td>NO LOOP</td>
<td>Loop Mode. For compatibility with I2S protocol, all bits are set to zero when not sending data.</td>
</tr>
<tr>
<td>MSBF</td>
<td>1</td>
<td>Most Significant Bit First. MSB is received first.</td>
</tr>
</tbody>
</table>
Connecting the Atmel SSC to an I2S Bus

**Table 6. Receive Frame Mode Register (SSC_RFMR) Settings (Continued)**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATNB</td>
<td>0 (1 slot per frame)</td>
<td>Data Number per frame. Programs the number of data contained in one frame.</td>
</tr>
<tr>
<td>FSLEN</td>
<td>0 (Default value because not used)</td>
<td>Frame Sync LENgth.</td>
</tr>
<tr>
<td>FSOS</td>
<td>Not used</td>
<td>Frame Sync Output Selection</td>
</tr>
<tr>
<td>FSDEN</td>
<td>Disabled</td>
<td>Frame Sync Data ENable. No Sync Data (SSC_RSHR) to receive.</td>
</tr>
<tr>
<td>FSEDGE</td>
<td>Positive Edge Detection (Default value)</td>
<td>Frame Sync Edge Detection. Not used in this case.</td>
</tr>
</tbody>
</table>

```
*AT91C_SSC1_RFMR = AT91C_SSC_MSBF | (BITS_BY_SLOT-1) ;
```

**Configuring the Receive Clock Mode Register**

**Table 7. SSC1 Receive Clock Mode Register (SSC1_RCMR) Settings**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKS</td>
<td>TK Clock Signal</td>
<td>Receive Clock Selection</td>
</tr>
<tr>
<td>CKO</td>
<td>0 (Default value)</td>
<td>Clock Output Mode Selection. Not Used.</td>
</tr>
<tr>
<td>CKI</td>
<td>Sampling on rising edge of serial clock</td>
<td>Receive Clock Inversion</td>
</tr>
<tr>
<td>START</td>
<td>Transmit Start</td>
<td>Receive Start Selection</td>
</tr>
<tr>
<td>STTDLY</td>
<td>1</td>
<td>Receive Start Delay. First bit after the falling edge is the last bit of right channel.</td>
</tr>
<tr>
<td>PERIOD</td>
<td>0</td>
<td>Receive Period Divider Selection. Not used</td>
</tr>
</tbody>
</table>

```
*AT91C_SSC1_RCMR = AT91C_SSC_CKS_TK | AT91C_SSC_START_TX | ((1<<16) & AT91C_SSC_STTDLY) | 0x1 << 8 | AT91C_SSC_CKI ;
```
Configuration for Stereo Input and Output

To connect the Codec as stereo input and output (both channels received), the following implementation using I²S with a 16-bit data format can be used. The standard audio Codec is used in input and output modes. The first SSC peripheral, used in master mode, manages all I²S transmit signals and left-channel input. The second SSC peripheral manages right-channel input only and is synchronized with the first SSC peripheral.

Figure 4. Stereo Audio Codec with Two Input Channels

Configuring the SSC1

The configuration and settings for SSC1 are the same as those described in “Configuration for Stereo Output and Mono Input” on page 8.
Connecting the Atmel SSC to an I2S Bus

Configuring the SSC0

The SSC0 must be configured to receive the second stereo input channel.

1. Configure the PMC by enabling the SSC0 clock:

   ```c
   *AT91C_PMC_PCER |= 1<< AT91C_ID_SSC0;    /* enable the SSC0 peripheral clock*/
   ```

2. Reset the SSC0 Peripheral:

   ```c
   *AT91C_SSC0_CR = AT91C_SSC_SWRST ;
   ```

3. Configure the SSC0 Receive Frame Mode Register. The Receive Frame Mode Register (RFMR) is configured to manage data reception.

   ```c
   *AT91C_SSC0_RFMR = AT91C_SSC_FSOS_NONE  | AT91C_SSC_MSBF | (16-1);
   ```

Table 8. SSC0 Receive Frame Mode Register (SSC0_RFMR) Settings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATLEN</td>
<td>BITS_BY_SLOT - 1</td>
<td>Programs Data Length</td>
</tr>
<tr>
<td>LOOP</td>
<td>NO LOOP</td>
<td>Loop Mode. For compatibility with I2S protocol, all bits are set to zero when not sending data.</td>
</tr>
<tr>
<td>MSBF</td>
<td>1</td>
<td>Most Significant Bit First. MSB is received first.</td>
</tr>
<tr>
<td>DATNB</td>
<td>0 (1 slot per frame)</td>
<td>Data Number per frame. Programs the number of data contained in one frame.</td>
</tr>
<tr>
<td>FSLEN</td>
<td>0 (Default value because not used)</td>
<td>Frame Sync LENgth.</td>
</tr>
<tr>
<td>FSOS</td>
<td>Not used</td>
<td>Frame Sync Output Selection</td>
</tr>
<tr>
<td>FSDEN</td>
<td>Disabled</td>
<td>Frame Sync Data ENable. No Sync Data (SSC_RSHR) to receive.</td>
</tr>
<tr>
<td>FSEDGE</td>
<td>Positive Edge Detection (Default value )</td>
<td>Frame Sync Edge Detection Not used in this case.</td>
</tr>
</tbody>
</table>

1. Settings are identical to those of SSC1.

   ```c
   *AT91C_SSC0_RFMR = AT91C_SSC_FSOS_NONE  | AT91C_SSC_MSBF | | (16-1));
   ```
Table 9. SSC0 Receive Clock Mode Register (SSC0_RCMR) Settings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKS</td>
<td>RK Clock Signal</td>
<td>Receive Clock Selection</td>
</tr>
<tr>
<td>CKO</td>
<td>0 (Default value)</td>
<td>Clock Output Mode Selection. Not Used.</td>
</tr>
<tr>
<td>CKI</td>
<td>Sampling on rising edge of serial clock</td>
<td>Receive Clock Inversion</td>
</tr>
<tr>
<td>START</td>
<td>On rising edge of RF</td>
<td>Receive Start Selection. Receiver of SSC0 must get the right channel while SSC1 is receiving the left channel. Start edge condition is inverted with respect to SSC1.</td>
</tr>
<tr>
<td>STTDLY</td>
<td>1</td>
<td>Receive Start Delay. First bit after the falling edge is the last bit of right channel.</td>
</tr>
<tr>
<td>PERIOD</td>
<td>0</td>
<td>Receive Period Divider Selection. Not used</td>
</tr>
</tbody>
</table>

*AT91C_SSC0_RCMR = AT91C_SSC_CKS_RK | AT91C_SSC_CKO_NONE | AT91C_SSC_CKI | AT91C_SSC_START_RISE_RF | ((1<<16) & AT91C_SSC_STTDLY) ;

4. Enable SSC0 receiver RX:

*AT91C_SSC0_CR = AT91C_SSC_RXEN; /* Enable Tx */