Graz University of Technology

Bachelor Thesis

Sample Rate Conversion in Digital Signal Processors

conducted at the
Signal Processing and Speech Communications Laboratory
Graz University of Technology,
Austria

by
Marian Forster,
1031275

Supervisor:
DI Dr. Werner Magnes

Graz, May 26, 2014

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Abstract

The simultaneous usage of different sample rates within digital systems has become an important topic in digital signal processing. No matter if you deal with sensor signals, still images, audio or video data, in many cases they are needed in a different clock domain. This thesis discusses the challenges of multi-rate systems and the transition from one sample rate to a higher, a lower or an unsynchronized one. Initially a theoretical consideration of *Sample Rate Conversion* (SRC), to point out all the problems and possible solutions, has been done. By means of a state of the art *Digital Signal Processor* (DSP), the theory has been confronted with the practice, to show the implementation and achievable performance. One goal of this thesis is the development of some vivid examples for the *Digitale Audiotechnik* laboratory. The students should understand the idea of SRC by means of some code examples. In the final chapter two stand-alone SRC Chips from different manufacturers are compared, especially in terms of performance.

Zusammenfassung

Die simultane Verwendung von unterschiedlichen Abtastraten in digitalen Systemen stellt ein wichtiges Thema der Signalverarbeitung dar. Sowohl Sensordaten, Bilder oder Audiosignale werden sehr oft in einer anderen Domäne benötigt. Diese Arbeit beschäftigt sich mit den Herausforderungen von Multiraten-Systemen und dem Übergang von einer Abtastrate auf eine höhere, niedrigere oder asynchrone. Zunächst erfolgte eine theoretische Betrachtung von *Sample Rate Conversion* (SRC), die alle Herausforderungen, aber auch Lösungsvorschläge aufzeigen soll. Anhand eines modernen Signalprozessors wurde anschließend die Theorie der Praxis gegenüber gestellt, das die konkrete Umsetzung sowie die damit erreichbare Leistungsfähigkeit aufzeigen soll. Ein Ziel dieser Arbeit ist die Ausarbeitung von anschaulichen Übungsaufgaben für das *Digitale Audiotechnik Labor*. Im Zuge dessen wurden einige Beispielprogramme erstellt, mit denen die Teilnehmenden der Laborübung die Grundprinzipien eines SRC verstehen sollen. Den Abschluss dieser Arbeit bildet der Vergleich zweier Sample Rate Converter von unterschiedlichen Herstellern. Die beiden Chips wurden speziell auf ihre Leistungsfähigkeit verglichen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose and Importance of Sample Rate Conversion

Due to engineering, economic or historical reasons different systems use different sample rates. For example the US and the European television follow different standards and use different frame rates. To move films back and forth, *Sample Rate Conversion* (SRC) is needed because simply replaying the data at the new rate would cause change in pitch and movement. But also for rescaling or rotating still images and reducing computational complexity of certain narrow-band digital filters, SRC is mandatory today [6]. For audio signals (20 Hz to 20 kHz) a minimum sampling frequency of $f_s = 40\,\text{kHz}$ is required to completely reproduce the signal without spectral distortion[1]. For easier anti aliasing filter design, commonly used sampling frequencies are even higher (e.g. 44.1 kHz, 48 kHz or 96 kHz). Using a sampling frequency higher than $2 \cdot f_{max}$ is called *Oversampling*. Another benefit is an improvement of the *Signal to Noise Ratio* (SNR). The quantization noise is then distributed over a wider spectrum and decreases in the bandwidth of interest. By a rule of thumb, an oversampling ratio of 2 ($f_s = 4 \cdot f_{max}$) increases the SNR by 3 dB [7].

In a digital studio environment most of the equipment is synchronized over a *Word Clock* (WC) connection which is provided by a high precision clock source. This mechanism synchronizes all data from external devices. Stable synchronization is a very important requirement in digital signal processing because every deviation leads to audible distortion. However, some equipment uses asynchronous data formats (further discussed in Section 1.2) and to sync it with the clock of the *Digital Signal Processor* (DSP) (for example a mixing console) an SRC is needed.

1.2 Digital Audio Formats

The development of standards to interconnect audio hardware has become more and more important since the mid 1980s. As mentioned, most studio equipment is driven by a separate *Word Clock Generator* (WCG) to ensure that every sample comes in time. However, asynchronous formats (like AES/EBU or S/PDIF) use their own internal clocks. The above mentioned formats are very similar although S/PDIF is more often used in consumer devices and AES/EBU

[1] Shannons theorem: $f_{max} = f_s/2$

in professional studio applications.

In this section we will have a brief look at the two digital audio interface formats which are important for this thesis: the asynchronous S/PDIF format and the synchronous I^2S format, which is commonly used on a hardware level for chip-to-chip communication. In Chapter 4 both formats will be used to demonstrate SRC by means of a demo example.

1.2.1 AES/EBU and S/PDIF

The AES/EBU (or AES-3) format is a normed, self-clocking format often used to interconnect studio hardware. It is capable of transmitting two channels (stereo) at a maximum word length of 24 Bit. It supports sampling frequencies of 32 kHz, 44.1 kHz, 48 kHz, 88.2 kHz, 96 kHz and up to 192 kHz. The data is encoded with a *Biphase Mark Code* (BMC), where every logic low is a shift in level and a logic high is a double shift per Bit. Each sample time, a 64 Bit frame is transmitted. One frame consists of two subframes of 32 Bit, each containing one sample (left and right). However, only a maximum of 24 Bit is available for audio data. 192 frames are grouped into one audio block, where certain status information is only transmitted once per block. A sample rate of 48 kHz results in a BMC clock of 6.144 MHz

The first four time slots (Bits) of a subframe, frame or audio block are used to recognize the start. This is done by a double harm of the BMC. In case of a word length bigger than 20 Bit, the next four time slots carry the LSB of the sample. Otherwise they can be used to transmit auxiliary information, such as low quality talk-back audio from the producer. The last four time slots carry some flags to provide the device with additional information such as: validity, one user defined bit, a channel status bit and a parity bit to retain zero DC offset.

The channel status bits are collected from each frame (192 Bit = 24 Byte) to a status word, which are completely different in AES/EBU and S/PDIF. The very first bit distinguishes between the two standards [7][8].

1.2.2 I^2S Format

With greater popularity of digital audio in the late 1980s (CD, digital audio tape, digital TV-sound etc.), Philips introduced a standardized communication protocol for a number of *Integrated Circuits* (IC), such as: *Analog-to-Digital Converters* (ADC) and *Digital-to-Analog Converters* (DAC), DSPs, digital filters, error correction for compact disc and many more. To increase flexibility for both, the equipment and IC manufacturer, they developed the *Inter IC Sound* (I^2S) bus.

Since only audio data needs to be transfered (other signals, such as sub-coding and control are transmitted separately), a simple serial 3-wire bus is used. The bus consists of a data line for two time multiplexed data channels (SD), a word select line (WS)[2] and a clock line (SCK). The SCK and WS signals can be provided by the transmitter, the receiver or in more complex systems by a separate controller as shown in Figure 1.1. The SCK and WS generating device is called master.

Data is always transfered MSB first because the transmitter and receiver may have different word lengths. In case the receiver is sent more bits than its word length, all the bits after the LSB are ignored. If the receiver has to receive fewer bits than its word length, the missing bits are set to zero.

[2] The word select signal is often called *Frame Sync* (FS) as well

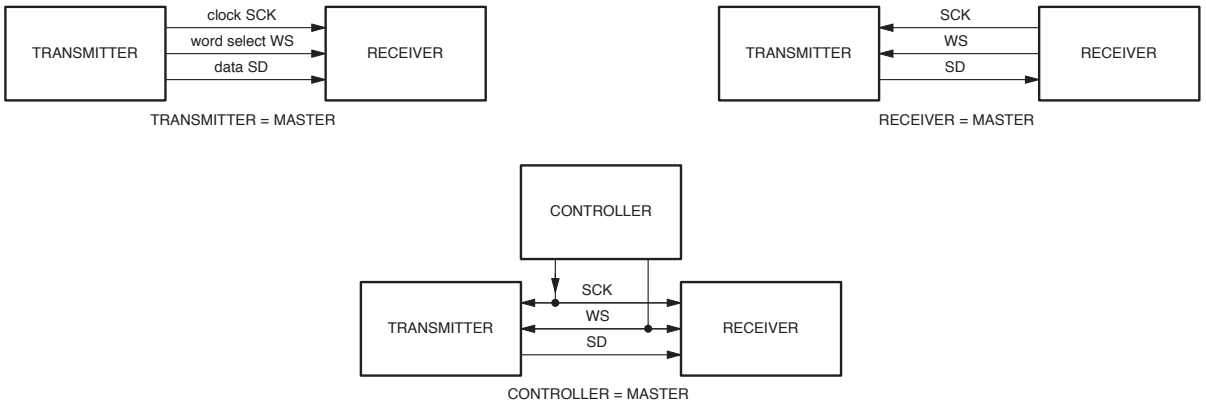


Figure 1.1: Different I²S Bus configurations [3, p. 1]

The word select line indicates the currently transmitted channel, whereby WS=0 means channel 1 (left) and WS=1 means channel 2 (right). As shown in Fig. 1.2, the WS signal is changing one clock period before the MSB of the data word. This allows the slave transmitter to sync its data for transmission. All timing requirements are specified relative to the minimum allowed clock periods of the devices in the system, which means that higher data rates are possible with faster hardware. This is why the standard is still very commonly used, even after 30 years [3].

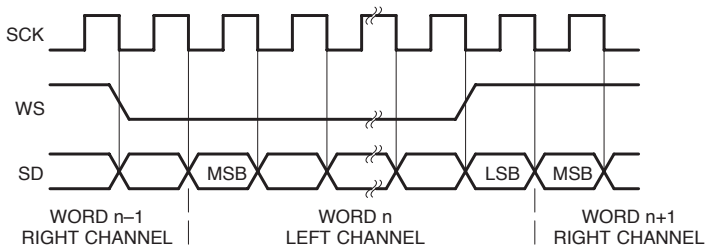


Figure 1.2: I²S Bus timing [3, p. 1]

1.3 Motivation and Objectives of this Thesis

Being highly interested in signal processing and programming hardware, the SHARC® DSP from *Analog Devices* draw my attention during the *Digital-Audio-Techniques Laboratory* (DAT-Lab[3]). To experiment with the development-kit of the DSP and therefore gaining basic knowledge about the procedures going along with digital signal processing was a great opportunity. The objective of this thesis is to get a deeper understanding of sample rate conversion, the theoretical background and their implementation on a state of the art signal processor. The behavior should be investigated and some vivid examples should be found, to be used in the DAT-Lab.

1.4 Structure of this Thesis

This thesis consists of three main parts: A theoretical consideration of up- and down-sampling, decimation and interpolation and finally the implementation of SRC in the DSP (Chapters 2 and 3). A demo example of a common problem (using analog inputs together with asynchronous formats) including a detailed description of the required steps to set it up, can be found in Chapter 4. Finally a comparison of two SRC chips shows how different manufacturers solved the challenges of SRC for a single-chip solution (Chapter 5).

[3] Course number 441.177, taught by DI Dr. Werner Magnes and DI David Fischer.

2 Theory of Sample Rate Conversion

The theoretical part of this thesis is mainly a summary of the chapter "Sample Rate Conversion" in the book *Understanding Digital Signal Processing* [4]. It addresses the challenges of decimation and interpolation and other important topics which are required to understand SRC.

2.1 The Idea of SRC

The most obvious way to perform SRC is to connect an ADC in series with a DAC to first convert the sequence $x[n]$ to a continuous signal $x(t)$ and then back into the digital domain at the new sample rate $f_{s,new}$. Due to spectral distortion induced by a DAC followed by an ADC conversion, this method is typically avoided in practice. The better alternative are all-digital solutions. There are three kinds of SRC: decreasing the sample rate (called *Decimation*, $f_{s,new} < f_{s,old}$), increasing it (called *Interpolation*, $f_{s,new} > f_{s,old}$) and synchronizing two signals with the same sampling frequency ($f_{s,new} = f_{s,old}$). In the latter case one might think, a *Zero Order Hold* (ZOH) element would do the job. Unfortunately two clock sources never have exactly the same frequency and a ZOH element would periodically drop a sample or hold it too long which leads to distortion. However, in practice the signal gets oversampled by a very high factor (2^{20}) and then a ZOH element is applied. This way the error can be reduced significantly. More about the practical point of view follows in Chapter 4.

2.1.1 Decimation

First let's have a look at the case $f_{s,new} < f_{s,old}$, which is called *Decimation* or *Downsampling*. To reduce the sample rate of a digital signal by a factor of M, the process is simple: retain every Mth sample and discard all the remaining ones. The new sampling frequency is now $f_{s,new} = f_{s,old}/M$. As we see in Figure 2.1 (here $M = 3$), the downsampling process is shifting the spectrum towards $f_{s,new}$ and since we have a periodic spectrum[4], as well as to every multiple of $f_{s,new}$. If the bandwidth was not limited, the spectra would overlap and we would lose information. Now we can see the limit of downsampling. The new sampling frequency needs to be $f_{s,new} > 2B$, where B is the bandwidth. If a decimation application requires $f_{s,new}$ to be less than $2B$, $x[n]$ needs to be low-pass filtered before discarding samples.

[4] By quantizing the analog signal (ADC), the original spectrum is convolved with the sampling frequency which results in a periodic spectrum [7].

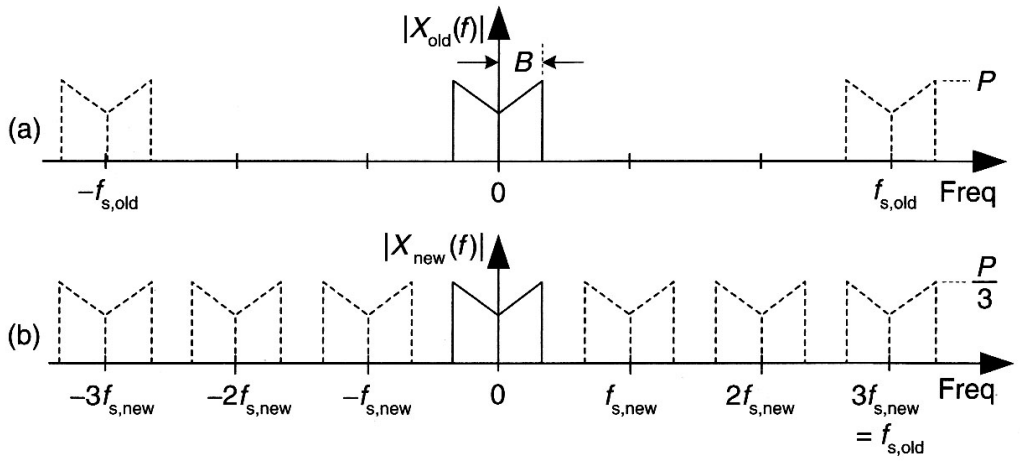


Figure 2.1: Spectrum before and after downsampling [4, p.509]

If the original signal has the bandwidth B and we are interested in retaining the bandwidth B', we see the response of the required low-pass filter for the given decimation in Figure 2.2. Notice that the full attenuation frequency f_{stop} can be as high as $f_{stop} = f_{s,new} - B'$ and no spectral aliasing will occur in the band of interest B'.

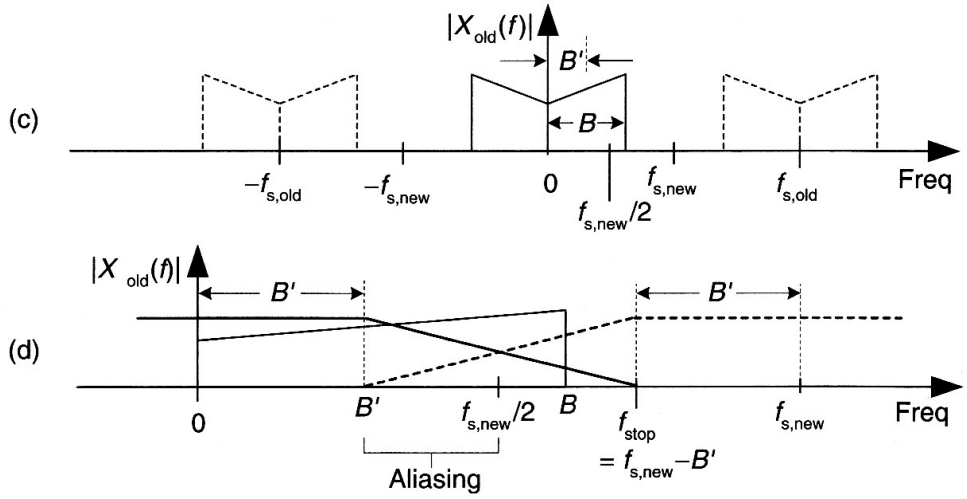


Figure 2.2: Low-pass filters frequency response relative to desired bandwidth B' [4, p.509]

In practice, the band limitation is performed by a non-recursive tapped-delay line *Finite Impulse Response* (FIR) structure, due to its linear phase and therefore constant group-delay of $N/2$ samples (N being the filter length).

2.1.2 Two-Stage Decimation

For large decimation factors M, a single stage filtering/decimation can get complex in computation. In practice, then decimation is performed in two stages. The first stage reduces the sample rate by a factor of M_1, the second stage by M_2. The desired decimation factor is the product of M_1 and M_2. A single stage decimation by a factor $M = 100$[5] would require an FIR filter with 2727 taps, which is simply not practical. The two stage concept can reduce this number

[5] For typical audio application the maximum decimation factor needed is much lower ($M < 20$).

significantly. The optimum value for M_1 is given by the following equation:

$$M_{1,opt} = 2M\frac{1-\sqrt{MF/(2-F)}}{2-F(M+1)} \tag{2.1}$$

where $F = (f_{stop} - B')/f_{stop}$ and M is the desired decimation factor.

For the same decimation as above with $M_1 = 25$ and $M_2 = 4$, the computational complexity can be reduced to a total of $N = 197$ tabs. Reducing from the largest to the smallest factor is always favorable. It is also advantageous to set the decimation factors equal to integer powers of 2 because then we can use efficient half-band filters.

2.1.3 Interpolation

If we want to increase the sampling frequency ($f_{s,new} > f_{s,old}$) we have to perform interpolation where new sample values need to be calculated. To increase a given sampling frequency $f_{s,old}$ by an integer factor L, we have to insert $L-1$ zero valued samples between each given sample (*Upsampling* or *Zero-Stuffing*) and then apply the sequence to a low-pass filter to get the interpolated output. In Figure 2.3 the concept of interpolation is shown. In 2.3 (a) we see the original sequence $x_{old}[n]$ and its spectrum $|X_{old}(f)|$. The dashed lines are the spectral replications. In the next step we insert three zeros between every sample and also append three zeros after the last one. This creates the new intermediate sequence $x_{in}[m]$ with its spectrum in 2.3 (b) and the new sampling frequency $f_{s,new} = 4f_{s,old}$. The solid curves in the spectrum are called images. The last step is to suppress the images with a low-pass filter, which is also called an *Interpolation Filter*. The remains of the images are called *Residual Images*. The quality of our interpolated sequence only depends on the low-pass filter. The better the images are attenuated, the better is the result. The interpolation process also has an inherit amplitude loss by the factor of L due to the zero stuffing. To achieve unity gain between $x_{old}[n]$ and $x_{new}[m]$ the filter must have a DC gain of L.

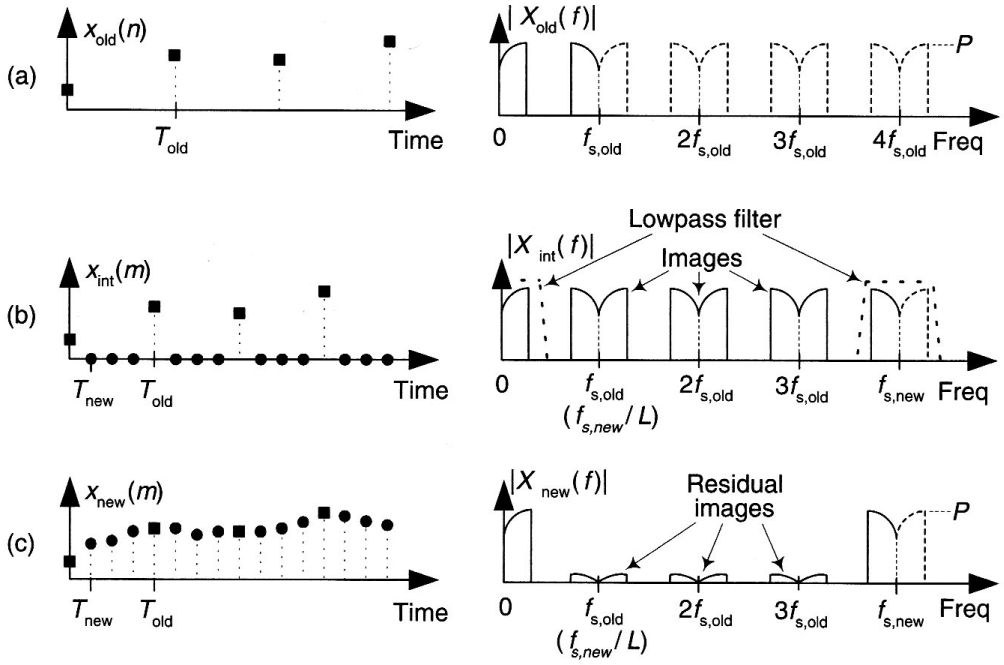


Figure 2.3: Example of an interpolation process [4, p.518]

2.1.4 Sample Rate Converter

We know now how to increase the sample rate by a factor L or decrease it by M (L and M are integer values). By combining decimation and interpolation we are able to change the sample rate by any rational factor L/M. This combination is often called *Sample Rate Converter* (SRC). Since both stages require a low-pass filter (decimation and interpolation filter), we can save resources by implementing only one filter which suppresses all spectral components above $f_{s,old}/2$ or $f_{s,old}/2 \cdot (L/M)$ whichever is smaller. This is mainly a theoretical model because it is not very efficient. If we want to change the sample rate by a factor of 4/3 for example, we fill in 3 zeros, apply them to the filter and then discard two-thirds of the values. In addition to that, three-fourth of the multiplications in the FIR filter (convolution with the filter coefficients) are by zero. The better solution is to use *Polyphase* filters, discussed in the following section.

2.1.5 Polyphase Filter Concept

We now introduce digital polyphase filters, by looking at an example. We want to perform an interpolation on a given sequence by $L = 4$. The upsampled sequence and the impulse response of the used interpolation filter are plotted in Figure 2.4. Note that the index k of the coefficients is mirrored towards the sequence index m. This is for better visualization of the convolution process. The filtering requires 12 multiplications per output sample, 9 of which are by zero. We reduce multiplications by just performing the 3 needed ones with the corresponding coefficients. For example $x_{new}[11] = x_{old}[2]h[3] + x_{old}[1]h[7] + x_{old}[0]h[11]$. To calculate the next output sample ($x_{new}[12]$), we simply slide the coefficients to the right and perform the convolution again. Now we need a different subset of coefficients. In particular $h[0]$, $h[4]$, $h[8]$ are used. If we repeat this process four times we see that we always need the same four subsets of coefficients and we can also save the effort of inserting zero valued samples, because only the samples of $x_{old}[n]$ are needed in the process. Removing the redundant calculations from an interpolation filter is called *Polyphase Filtering*.

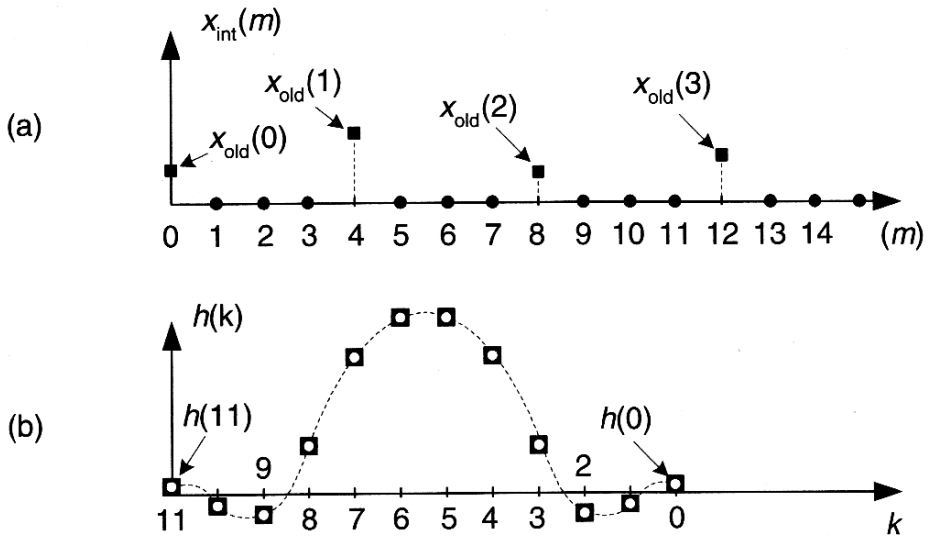


Figure 2.4: Interpolation by $L = 4$, upsampled sequence (a), filter coefficients (b) [4, p.523]

For easier implementation the filter prototype should have an integer multiple of L number of stages. There is a gain loss by the factor of L which can be compensated by scaling the coefficients with L or multiplying the output sequence by L.

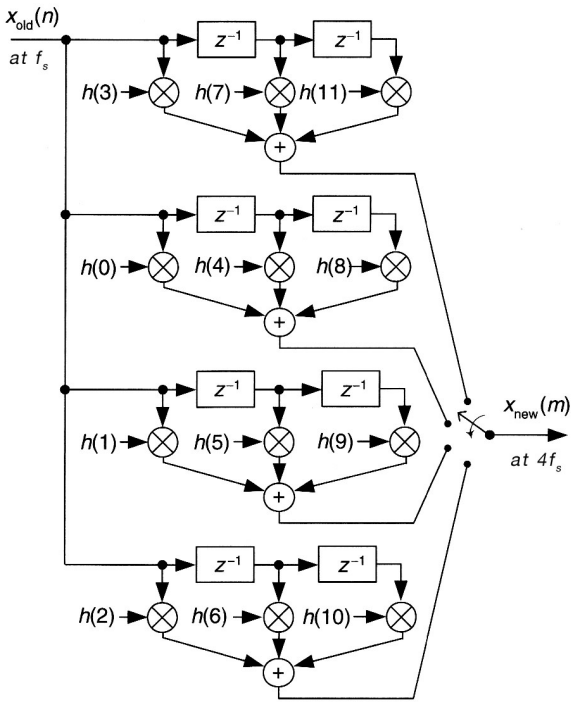


Figure 2.5: Polyphase interpolation [4, p.525]

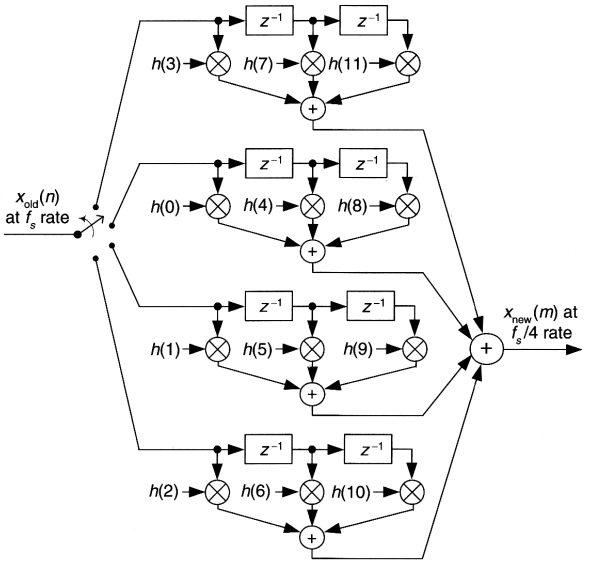


Figure 2.6: Polyphase decimation [4, p.527]

Figure 2.5 shows a possible implementation of a polyphase interpolation filter. There are four ($L = 4$) subfilters, each provided with the same input samples $x_{old}[n]$ but with a different set of coefficients. During each input period the switch on the output is rotating one cycle counterclockwise, producing 4 new samples. By slightly altering the setup a decimation filter can be created out of the interpolation structure as shown in Figure 2.6. Providing each subfilter with different input samples and accumulating the filter outputs, we are left with one sample per four input values (switch rotation). Note that the coefficients remain the same, no matter if the sequence gets interpolated or decimated. In practice large changes of sample rate are performed in multiple stages.

3 SRC on the SHARC® ADSP-21369

In this chapter we will have a closer look at the SRC module implemented in a state of the art SHARC® DSP by *Analog Devices*. Keeping the theory of Chapter 2 in mind, the differences to the practical approach will be shown. This chapter is based on the Hardware Reference of the DSP [1].

3.1 Conceptual Overview of the SRC-Module

The DSP is equipped with four stereo SRC modules (`SRC0-3`). Each module provides a *Signal-to-Noise Ratio* (SNR) of up to 128 dB. They can be used to perform synchronous and asynchronous sample rate conversion across independent stereo channels. Synchronous conversion is by a fixed rational factor, the output sample rate is only dependent on the input rate and the conversion factor. Asynchronous sample rate conversion is mapping any input rate to a given output rate. The modules also can be configured to work together to convert multichannel audio without phase mismatch. In this case, one SRC is configured to be the master and the slaves are using the sample rate ratio of the master module. Another application is to clean up jittery audio sources, like the S/PDIF receiver.

The SRC has a three wire interface (clock, frame sync, data) which supports I^2S, left- and right justified 16-, 18-, 20- and 24-Bit modes. If 20-, 18- or 16-Bit output is selected, the data gets dithered down to the desired word length.

Conceptually, the SRC interpolates the data by a factor of 2^{20} and picks the nearest value to the output sampling frequency. The interpolation is performed by a 64-tap *Finite Impulse Response* (FIR) filter with 2^{20} poly phases, a FIFO[6], a digital servo loop and a digital circuit. The last two components measure the difference between input and output sample rate within 5 ps. This is necessary in order to select the correct filter coefficients and keep the distortion at a minimum. The jitter rejection of the digital servo loop starts at less than 1 Hz, so a long settling time[7] is required when the SRC is enabled or the sample rate changes. To reduce the setteling time, whenever this happens, the servo loop enters the fast settling mode. During this period the

[6] A FIFO (First-In-First-Out) is a circular buffer.

[7] The settling time is the time until the difference measurement between input and output sample rate is within 5 ps.

MUTE_OUT signal is asserted high. Normally the MUTE_OUT signal is connected to the MUTE_IN input, which is used to automatically mute the SRC.

The determined sample rate ratio is used to scale the length of the decimation filter. To avoid oscillation in the scaling, hysteresis in measuring the sample rate ratio is used. In multichannel applications, when multiple SRC blocks are driven by the same input and output clock, this hysteresis can cause different group delays between the channels. In this case, the phase-matching mode should be used. Then one master-SRC transmits its sample rate ratio to the slaves. Thus, the group delay remains the same.

3.2 Hardware Model of the SRC

Interpolating a signal sampled with 192 kHz by a factor of 2^{20} results in a sample rate of 201.3 GHz which is clearly unpractical. Since interpolating involves zero-stuffing, the poly phase concept, discussed in Section 2.1.5, is a clever way to reduce the workload. A 64-tap FIR filter with the right coefficients is sufficient to suppress the images caused by the interpolation. The difficulty is to select the right convolution from 2^{20} possibilities. The arrival of the $f_{s,out}$-clock must be measured with an accuracy of 1/201.3 GHz=4.96 ps. To achieve this, several coarse measurements are made and averaged over time.

Another problem with implementing the polyphase concept is the required amount of filter coefficients. Since there are 2^{20} possible convolutions with a 64-tap FIR filter, there have to be $2^{26} = 67108864$ coefficients stored in a *Read Only Memory* (ROM). To reduce ROM size, the SRC only stores a small subset of coefficients and performs a high order interpolation on them.

So far the approach described above only works for $f_{s,out} > f_{s,in}$. To reduce the sample rate, the ROM starting adress, input data and length of the convolution must be scaled. As the input sample rate rises over the output sample rate, the coefficients are dynamically altered and the length of the FIR filter is increased by the factor $f_{s,in}/f_{s,out}$ to move the cutoff frequency of the anti-aliasing filter downwards.

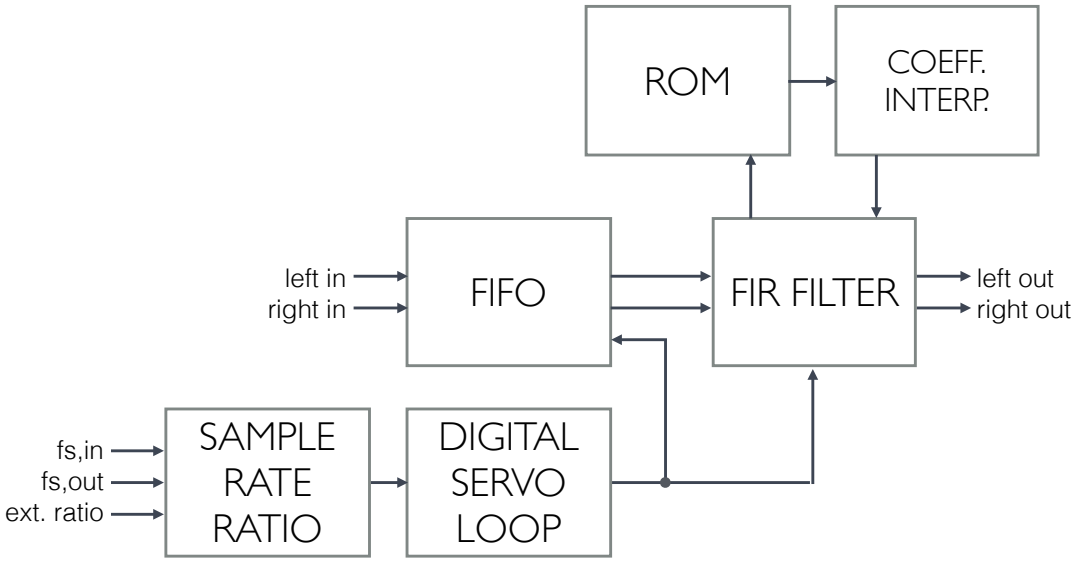


Figure 3.1: Architecture of SRC block

Figure 3.1 shows the architecture block diagram of the SRC module. The FIFO block adjusts the amplitude of the left and right sample respectively for both soft muting and to compensate the amplitude loss caused by the interpolation, as discussed in Section 2.1.3. The RAM in the FIFO is 512 words large for both channels. The $f_{s,in}$-counter provides the FIFO with the write address and the servo loop with a ramp input. An offset to the write address is added to prevent the read pointer from overlapping with the write address. The group delay select signal (`GRPDLYS`) selects the offset. When the `GRPDLYS` signal is high, a small offset (16) and when it is low a large offset (64) is added. A higher offset is useful when small changes in the sample rate are expected. The maximum decimation factor for each offset can be calculated from the RAM size (512) and `GRPDLYS`: $(512 - 16)/64 = 7.75$ for short group delay and $(512 - 64)/64 = 7$ for long group delay.

If $f_{s,out} > f_{s,in}$ (case 1) the FIR filter has 64 taps. In the case of $f_{s,out} < f_{s,in}$ (case 2) it is $64 \cdot (f_{s,out}/f_{s,in})$ taps. Before the convolution the RAM and ROM are provided with the start address pointers by the digital servo loop. It is triggered at the start of the $f_{s,out}$ period. The filter then steps through the RAM and decrements its pointer by 1 for every tap. The ROM pointer increments by 2^{20} for case 1 and $(f_{s,out}/f_{s,in}) \cdot 2^{20}$ for case 2. The convolution is completed when the ROM pointer runs over.

4 Demo Example

The purpose of this demo is to show how the Evaluation Kit of the Sharc DSP used in the DAT laboratory needs to be set up, how the signals need to be routed and how the SRC needs to be configured to connect asynchronous data formats to the DSP. It also will be shown what happens if the configuration is wrong and why the SRC is mandatory at least when we want to use asynchronous formats together with the ADC inputs and outputs. This chapter is based on an example that comes with the Visual DSP++ Software from Analog Devices. It has been modified and extended to be as vivid as possible.

4.1 The ADSP-21369 EZ-KIT LITE®

The main tool used for investigating sample rate conversion in practice is the ADSP-21369 EZ-KIT LITE® from *Analog Devices*. This is a development environment for rapid prototyping and debugging audio applications. It consists of analog audio inputs and outputs via an external Codec chip (AD1835), several multi purpose LEDs and puch-buttons, a number of interfaces for communication and debugging and of course the ADSP-21369 (DSP) [5].

The DSP is a high performance floating-point signal processor especially designed for audio applications[8]. As can be seen in Figure 4.1, the core processor mainly consists of two *Processing Elements* (PEX and PEY). Each element contains a 32-Bit computation unit, two *Data Address Generators* (DAG) for indirect addressing and circular data buffers in hardware and four data buses for communication to the different memory blocks[9]. The I/O processor is of major importance because it provides all the external signals and disburdens the core processor. It has an integrated *Direct Memory Access* (DMA) controller, *Serial Peripheral Interfaces* (SPI) and 8 *Serial Ports* (SPORT) for communication purposes. It is also capable of routing any data stream to the desired destination via the *Signal Routing Unit* (SRU). The data from the Codec to one of the SRC modules for example. The SRU also allows us to connect a set of physical pins of the DSP package to any module or other pin. It is possible to route the S/PDIF signal provided by an external device to the S/PDIF receiver module for example [1].

A detailed description of all the features and modules implemented in the DSP would break the frame of this thesis. We will focus on the setup of the modules needed, which are the Signal

[8] Due to its capability of computing two sets of data (stereo) within one cycle (SIMD)
[9] Up to 3 Mbit on-chip SRAM

Routing Unit (SRU), the Serial Ports (SPORT), the Codec Chip (AD1835), the S/PDIF receiver and of course the Sample Rate Converter (SRC). But first we discuss the different possibilities of working with S/PDIF (or AES/EBU) data steams.

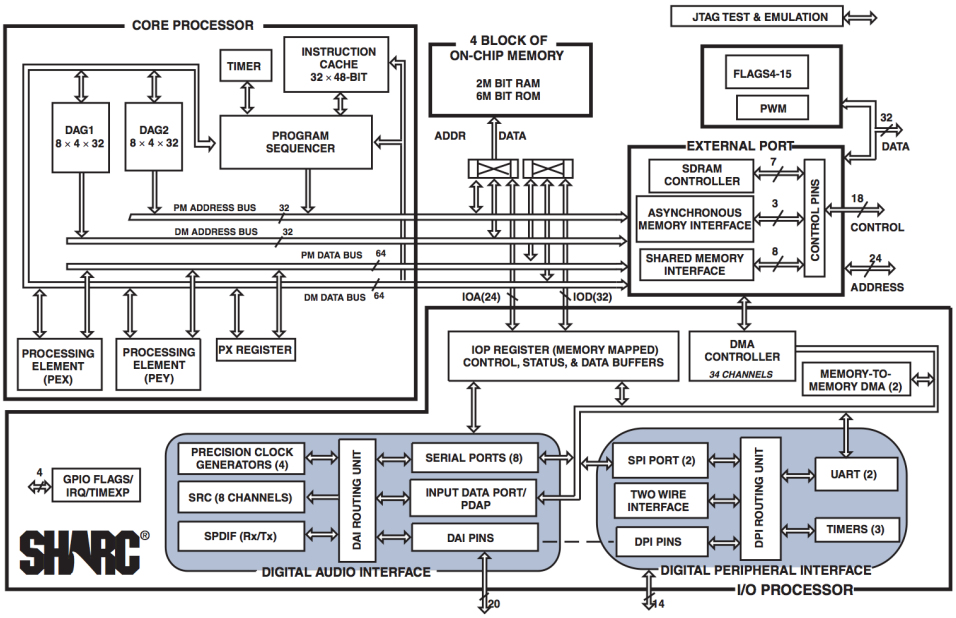


Figure 4.1: Functional block diagram [1]

4.2 Connection With S/PDIF Hardware

As discussed in Section 1.2.1, S/PDIF is a selfclocking format. It means that the clock can be regenerated at the receiver side and no separate clock wire is needed. An S/PDIF receiver derives the three needed signals from the data stream to convert it into an I^2S stream: Bit Clock (CLK), Frame Sync (FS) and Data (DAT). There are several scenarios how to handle the S/PDIF signal and the synchronization inside the DSP:

1. **S/PDIF input - S/PDIF output:**
   The regenerated CLK and FS is used to drive the interrupts, signal processing[10] and the S/PDIF transmitter. Only one stereo input and output is possible because the DSP only features one S/PDIF receiver and transmitter.

2. **S/PDIF input - DAC output (no SRC):**
   The external DAC (AD1835) is using the regenerated CLK and FS from the S/PDIF interface. It is possible to distribute the input to up to four stereo outputs. Keep in mind that jitter in the S/PDIF signal will affect the quality of all outputs as well. Nevertheless, this configuration works just fine for most applications although the possibilities are limited, since the hole system has to run at the same sample rate as the S/PDIF device.

3. **S/PDIF input - DAC output (with SRC)**:
   The Codec is clocked by the high *Precision Clock Generator* (PCG) of the DSP and the S/PDIF input is linked in by one of the on-chip SRC modules. We are no longer forced to use the same sample rate as the S/PDIF device. This provides jitter rejection and maximum flexibility.

[10] Any user defined manipulation of the signal

4. **More than one S/PDIF inputs - DAC output (with SRC)**[11]:
   In this configuration we clearly see the benefits of using an SRC. Different devices can have different sample rates (e.g. 44.1 kHz and 48 kHz) of course, but at least they have unsynchronized clocks. By connecting the digital inputs to an SRC, we can now work seamlessly with all the data.

4.3 Step-by-Step Setup of the Hardware

In this section the software of configuration 3 will be discussed. This is a very representative example which shows the benefits of SRC quite well. The hardware and software will be discussed as detailed as necessary to get a deep understanding of what is happening. The code also provides an option to use a sample-hold method instead of the SRC. By disabling the SRC, the dropped samples, caused by the inaccuracies of two clocks sources are clearly audible during playback.

4.3.1 Hardware

To set up a test environment, some additional equipment is needed. The list of required hardware is shown in Table 4.1. The S/PDIF device ideally has a modifiable sample rate, but for testing purposes a simple CD player also does the job. During software developing the Audio Precision SYS 2700 was used, due to its convenient features (like jitter generation, wide sampling frequency range,...).

Hardware	Comment
ADSP-21369 EZ-KIT Lite®	
Power supply + USB Cable	
S/PDIF device	Audio Precision, CD-Player,..
Analog audio source	Mobile phone, PC,..
Speaker	Or headphones to listen to the results
S/PDIF Cable	1-2 m
Chinch Cable	

Table 4.1: Required hardware

4.3.2 Software Overview

The code is segmented into a number of files to enhance readability. All the initialization functions are found in the respective file (for example `initSRC.asm`). A list of all files is provided in Table 4.2. Note that the signal processing takes place in `SPORTisr.asm` and not in `main.asm` as one could expect. All the routines are written in assembly language.

File	Comment
`21369_IVT.asm`	Interrupt Vector Table
`ad1835.h`	Definitions for Codec
`init1835viaSPI.asm`	Init Functions for Codec
`initPLL_SDRAM.asm`	Init Functions for PLL and SDRAM
`initSPDIF.asm`	Init Functions for S/PDIF Module
`initSPORT.asm`	Init Functions for SPORTs

[11] This configuration is not possible with the Evaluation Kit due to the lack of a second S/PDIF input

`initSRC.asm`	Init Functions for SRC
`initSRU.asm`	Init Functions for Signal Routing Unit
`main.asm`	Containing Main Loop
`SPORTisr.asm`	Interrupt Service Routine

Table 4.2: Software architecture

4.3.3 Clock Domain

As mentioned earlier, the goal of this configuration is to decouple input and output sample rate as well as to reject jitter from the input. To achieve this, we want the Codec chip to be clocked by a high precision source. In the standard configuration, the Codec expects a 12.288 MHz clock (MCLK), which is divided internally to the desired CLK and FS for the I^2S communication (see Section 1.2.2). This is depending on the settings initially transmitted via SPI. For example, a sample rate of $f_s = 48\,\text{kHz}$ requires an FS signal of $f_{FS} = 48\,\text{kHz}$ (left and right sample per period) but a CLK edge for each bit, for example $f_{CLK} = 24\,\text{Bit} \cdot 48\,\text{kHz} = 768\,\text{kHz}$. The S/PDIF receiver module expects the digital signal from the external device and provides all the important signals like Data Out (`DIR_DAT_O`), Frame Sync Out (`DIR_FS_O`) and the regenerated Bit-Clock (`DIR_CLK_O`). Figure 4.2 shows the required routing of the clock signals. `SPORT0` and `SPORT1`, as well as the SRC draw the CLK and FS from the S/PDIF receiver. `SPORT2`, `SPORT3` and the second clock input of the SRC are in sync with the Codec. The block diagram shows a multirate system where the SRC forms the bridge between two clock domains.

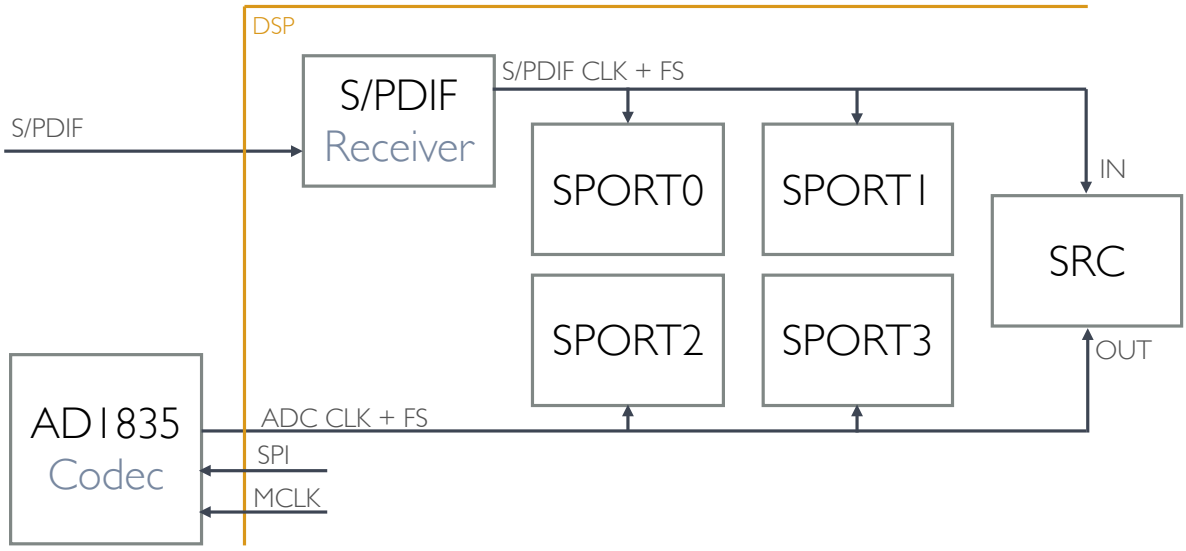


Figure 4.2: Clock domains and distribution

4.3.4 Configuration of the Signal Routing Unit (SRU)

The DSP is equipped with two peripheral interfaces, the *Digital Audio Interface* (DAI) and the *Digital Peripheral Interface* (DPI) which are comprised of two groups of peripherals and their *Signal Routing Units* (SRU1 and SRU2). The inputs and outputs of the peripherals (the assignments are listed below) are not directly wired to a physical pin, rather the SRUs connect all the signals to a set of pins and an internal patchbay. This way the user can interconnect them in any order and experiences great flexibility.

DAI SRU1	DPI SRU2
Serial Ports (up to 8)	Timers (3)
S/PDIF Interface	Serial Peripheral Interface (2)
Interrupts	Two Wire Interface
Precision Clock Generators (4)	Universal Asynchronous Interface (2)
Asynchronous Sample Rate Converters (8)	General Purpose I/O (9)
Input Data Port	Flags (12)
General Purpose I/O (20)	

Table 4.3: DAI/DPI assignments

All needed modules for this example are served by the SRU1. Figure 4.3 shows the DAI Pin-buffers, the SRU1 and the peripherals of its group. Each physical pin is operated by an active device called *Pin Buffer* (PB). The signal flow (input or output) is defined by the Pin Buffer Enable (PBEN) wire. If the signal is high, the PB is forwarding the signal connected to its input (PBxxI) to the designated pin and vice versa. There is also a pull up resistor for each pin, which can be activated if needed. In the VisualDSP++ environment a macro is provided to ease the signal routing process: `SRU(Output Signal, Input Signal)`. Keep in mind that you can only connect outputs of certain modules with inputs of other modules or PBs.

There are some more mandatory connection. All required routings, which are defined in `initSRU.asm`, are listed in Table 4.4.

Output	Input	Comment
DAI_PB18_O	DIR_I	S/PDIF signal to S/PDIF Receiver
DIR_CLK_O	SPORT0_CLK_I	S/PDIF Clock to SPORT0
	SPORT1_CLK_I	S/PDIF Clock to SPORT1
	SRC1_CLK_IP_I	S/PDIF Clock to SRC1 Clock Input
DIR_FS_O	SPORT0_FS_I	S/PDIF FS to SPORT0 FS
	SPORT1_FS_I	S/PDIF FS to SPORT1 FS
	SRC1_FS_IP_I	S/PDIF FS to SRC1 Input FS
DIR_DAT_O	SPORT0_DA_I	S/PDIF Data to SPORT0A
SPORT1_DA_O	SRC1_DAT_IP_I	SPORT1 Data A to SRC1
DAI_PB05_O	SPORT2_DB_I	ADC Data to SPORT2B
DAI_PB17_O	DAI_PB06_I	MCLK (12.288 MHz) to Codec
DAI_PB07_O	DAI_PB13_I	Codec Clock to PB13
	SRC1_CLK_OP_I	Codec Clock to SRC1
	SPORT2_CLK_I	Codec Clock to SPORT2
	SPORT3_CLK_I	Codec Clock to SPORT3
	SPORT4_CLK_I	Codec Clock to SPORT4
DAI_PB08_O	DAI_PB14_I	Codec FS to PB14
	SRC1_FS_OP_I	Codec FS to SRC1 Output FS
	SPORT2_FS_I	Codec FS to SPORT2 FS
	SPORT3_FS_I	Codec FS to SPORT3 FS
	SPORT4_FS_I	Codec FS to SPORT4 FS
SRC1_DAT_OP_O	SPORT2_DA_I	SRC1 Data Output to SPORT2A
SPORT3_DA_O	DAI_PB09_I	Data to DAC
SPORT3_DB_O	DAI_PB10_I	Data to DAC

SPORT4_DA_O	DAI_PB11_I	Data to DAC
SPORT4_DB_O	DAI_PB12_I	Data to DAC

Table 4.4: Routed signals

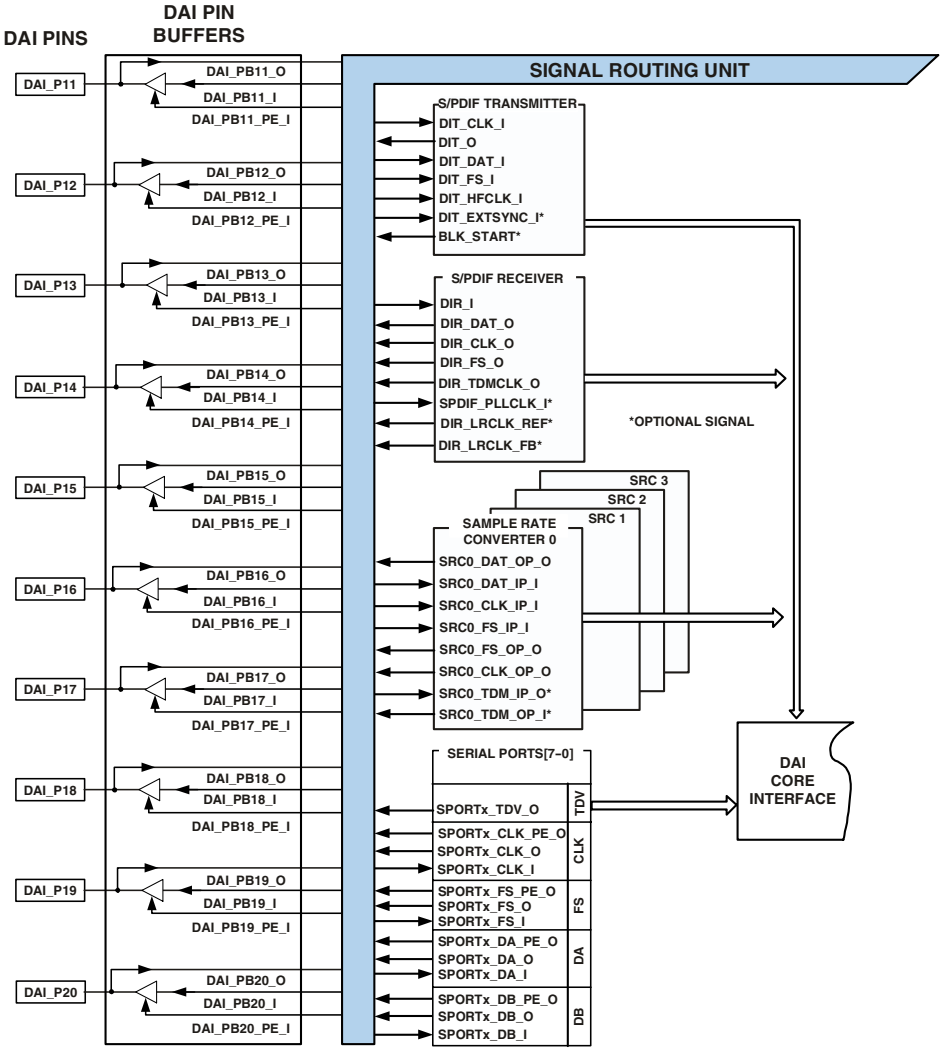


Figure 4.3: Signal Routing Unit (SRU1) and Digital Audio Interface (DAI) [1, p. 206]

4.3.5 Initialize Transmit and Receive SPORTs

The 8 Serial Ports can be used to interface a variety of digital peripheral devices. Each SPORT consists of two data lines, a clock and frame sync. They can be programmed either to transmit or receive data. To send a sample via an SPORT, the data has to be written into the respective transmit register, for example `TXSP0A`. Then the sample gets clocked out to the connected destination (for example a module like the S/PDIF transmitter). The SPORTs can be routed to 16 simultaneous transmit or receive pins that support up to 32 channels (two channels per data line). They can be operated in a number of modes. In this example only the I^2S-Mode is used. If data has arrived, an interrupt can be triggered by each SPORT [2].

For configuration purposes a 32 Bit control register for each SPORT (`SPCTLx`) is provided. For this example we need to set `OPMODE` (Serial- or I^2S-Mode), `SLEN` (Number of bits in each word),

SPEN_A (Enable data line A) and SPTRANS (Transmit or receive data).

The SPORTs are configured to handle 32 Bit data for correct alignment of the data word in the 40 Bit registers. For detailed information on all the possible configurations, please refer to [1, p. A-36].

4.3.6 Initialize the Codec (AD1835) via SPI

The AD1835 is a single-chip Codec, equipped with four stereo DACs and one stereo ADC. It can input and output data with sample rates up to 96 kHz on all channels and one DAC at 192 kHz. The master input clock (MCLK) can be generated by the on-board 12.288 MHz oscillator or be supplied by the processor. This allows synchronization with other devices in the system. By default the MCLK gets multiplied by 2 (12.288 MHz · 2 = 24.576 MHz) and then decimated by 512. This is done by a *Phase Locked Loop* (PLL) and results in a sample rate of 48 kHz [10]. The Codec can be configured as the master or a slave. In master mode, it drives the CLK and FS signals to the processor. In slave mode, the processor generates and drives all the synchronization signals. The internal configuration registers are configured via the SPI port of the DSP [5].

In the init1835viaSPI.asm file, the configuration data is transmitted to the Codec. This includes ADC sample rate (ADCFS48), DAC sample rate (DACFS48), word length (DAC24BIT), DAC volume (DACVOL_MAX) and data format (ADCI2S). Keep in mind that the ADC only operates at f_s = 48 kHz or 96 kHz. The Codec provides a 32 Bit word independent of the configured resolution.

4.3.7 Initialize the S/PDIF interface

The S/PDIF receiver module converts the biphase encoded signal into a serial data stream, in our case a I^2S signal. It also can be formatted as left-justified or right-justified with word length of 16, 18, 20 or 24 bits [2].

DIRCTL is the control register of the S/PDIF receiver/transmitter. This 32-bit read/write register is used to set up preferences like clock source, error handling, mute and single-channel double-frequency mode. In this example we use the standard configuration by setting the DIRCTL to all zeros.

4.3.8 Initialize SRC Parameters

The operating mode, communication formats and emphasis-filters of an SRC module are defined in the SRCCTLx registers. Also input/output format, output word length, matched phase mode, dither and several muting options can be selected [1].

For this example we need to set input/output format to I^2S and output word length to 24 bits. Enabling the SRC (setting the SRC_ENABLE bit) should be done at least one cycle after setting the other configurations. There is also a bypass mode which can be used if the input and output sample rate are synchronous or to pass through non audio data. Setting the bypass bit is passing the data through a sample-hold element.

The MUTE_OUT bit is asserted high whenever the an SRC module is enabled or the sample rate ratio changes. In the standard configuration this signal (flag) is connected to MUTE_IN, which performs an automatic soft mute by fading out the signal whenever it is asserted high. The

MUTE_OUT bit is cleared when the sample rate has settled. By setting the SRCx_MUTE_EN bit, automatic muting can be disabled, which is not recommended because this can lead to distortions at the output. To hard or soft mute (turn off or fade out the signal) the SRC during runtime SRCx_HARD_MUTE or SRCx_SOFTMUTE[12] can be asserted high. When the SRCx_AUTO_MUTE bit is set, the NOAUDIO flag from the S/PDIF receiver is used to immediately mute the SRC. This is a useful method to detect non-audio data from the S/PDIF device.

There are also two read-only registers (SRCRAT0 and SRCRAT1) which report the mute and I/O sample rate ratio for all four SRC modules. The ratio ($f_{s,out}/f_{s,in}$) is a 15 bit fixed point number with four digits on the left side of the comma. Table 4.5 shows the ratio for the case $f_{s,out}/f_{s,in}$ = 48 kHz/88.2 kHz = 0.54394.

RAT	**0**	**0**	**0**	**0.**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**
Value	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}
	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$	$\frac{1}{512}$	$\frac{1}{1024}$	$\frac{1}{2048}$

Table 4.5: SRC sample rate ratio

4.3.9 Interrupt Service Routines

The processing takes place in two *Interrupt Service Routines* (ISR) which are triggered by the SPORTs. Every time a sample arrives at SPORT0, which is connected to the S/PDIF receiver, the DSP jumps out of the main loop into an ISR (Table 4.6, line 1). Since an SPORT can only be configured to either transmit or receive, this ISR writes the new data into the transmit register (TXSP1A) of SPORT1, which is connected to the data input of SRC1. The second ISR is triggered by SPORT2. As shown in Fig. 4.4, the data channel A is served by the Codec (ADC input) and channel B by the SRC1 output. Since the SRC has changed the clock domain of the S/PDIF signal to be in sync with the Codec (Fig. 4.2), we are now left with two sets of data in SPORT2. By using the command `r10 = dm(RXSP0A)` the data can be accessed and an algorithm can be performed (Table 4.6, line 18). In this example, the digital S/PDIF input (r10) gets mixed with the analog input (r11) and the data is sent to to all four DAC outputs (via SPORT3). Note that the processed data is in stereo and the described program is executed for the left and right sample individually.

This demo example can be seen as a simple 4 channel mixer with 2 analog inputs, 2 digital inputs and 8 analog outputs where the sample rate of the digital device does not affect the system. In line 14 and 15 the SRC input and output are written into two ring buffers for visualization of the SRC performance. This will be used to discuss some measurements regarding group-delay and distortion, in Section 4.4.

```
1  _ISR_SP0: // Interrupt called by incoming SPDIF sample (via SPORT0A)
2
3      r7=dm(RXSP0A);        // Read new sample from SPDIF (via SPORT0A)
4      dm(TXSP1A)=r7;        // write it to SRC1 (via SPORT1A)
5
6  _ISR_SP0.end:
7      rti;
8
9  _ISR_SP2: // Interrupt called by incoming ADC sample (via SPORT2B)
10
11     r10=dm(RXSP2A);        // read new sample from SRC1 Output (via SPORT2A)
12     r11=dm(RXSP2B);        // read new sample from ADC (via SPORT2B)
13
14     dm(I0,M0)=r7;          // write digital input into ringbuffer 0
15     dm(I1,M1)=r10;         // write SRC output input into ringbuffer 1
```

[12] Note that there is no underscore between SOFT and MUTE.

```
16
17  //-----------algorithm----------------------
18      r9 = r10 + r11;         // S/PDIF + ADC
19  //------------------------------------------
20
21      dm(TXSP3A)=r9;          // write to DAC4 and Headphone Jack
22      dm(TXSP3B)=r9;          // write to DAC3
23      dm(TXSP4A)=r9;          // write to DAC2
24      dm(TXSP4B)=r9;          // write to DAC1
25
26  _ISR_SP2.end:
27      rti;
```

Table 4.6: Interrupt Service Routines of SPORT0 and SPORT2

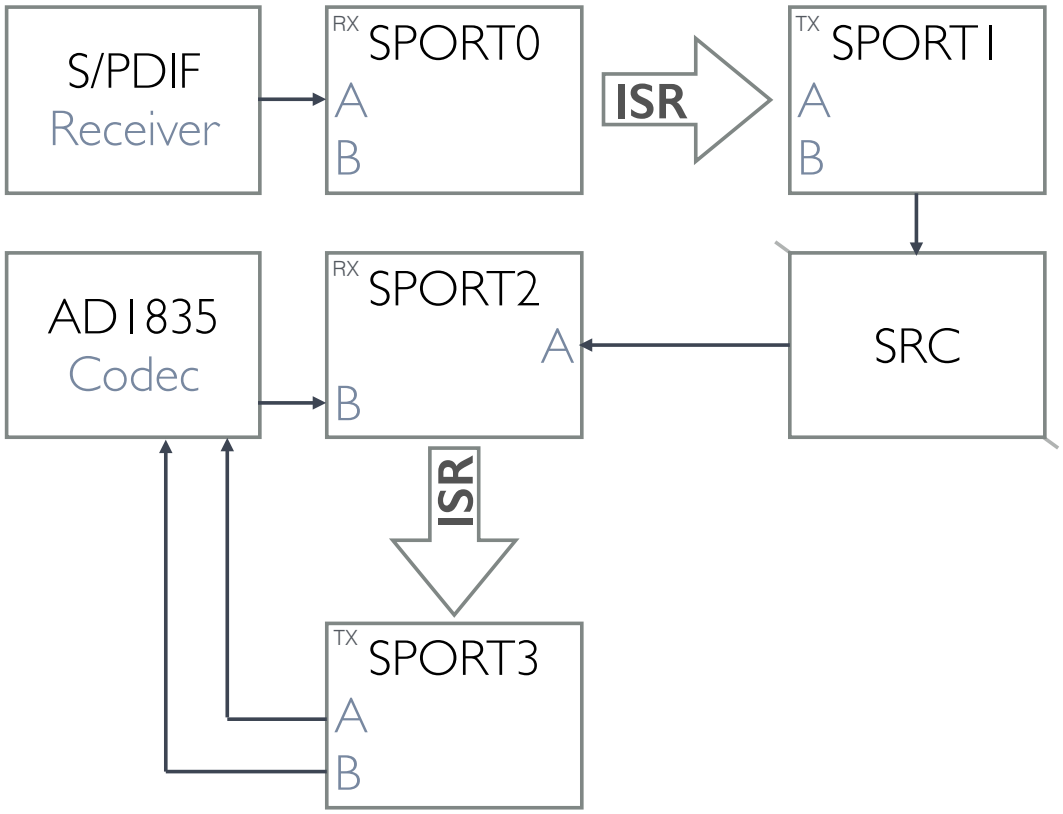


Figure 4.4: Data flow

4.4 Measurements

In the ISR the input and SRC output are written into two ring-buffers (Table 4.6, line 14 and 15) to compare the two signals in time domain. In Figure 4.5 two effects of SRC can be observed very vividly. The green signal is the sample-hold output (SRC in bypass mode) and the red one is the SRC output for a 1 kHz sine input. In this example the input rate is $f_{s,in} = 44.100\,\text{kHz}$ and the output rate $f_{s,out} = 48.000\,\text{kHz}$. Since the input signal offers not enough samples per second, the sample-hold element uses samples twice periodically. These steps in the signal lead to non-harmonic distortions which are clearly audible. The spectral measurements below will show this once again.

The second effect SRC has on a signal is a time delay. This is due to the fact that interpolation requires FIR filtering and this leads to a groupdelay of half the filter order. The actual delay is in fact more than one period, it is about 1 ms.

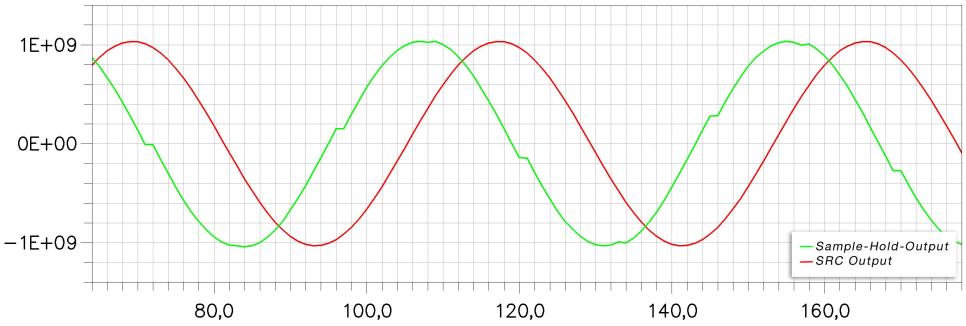


Figure 4.5: 1 kHz, $f_{s,in} = 44.100\,kHz$ and $f_{s,out} = 48.000\,kHz$

To further investigate the behavior of a sample rate converter a digital impulse was passed through, using a predefined ring buffer. The input (green) and output (red) of an SRC module can be seen in Figure 4.6. Now the exact groupdelay can be determined. For a sample rate ratio of 48 kHz/44.1 kHz the delay is 58 output samples (1.2 ms). The response of an FIR filter to a digital impulse is also the set of implemented filter coefficients. If the input sample rate rises over the output rate, the filter length is reduced and the group delay decreases as well. In this example (48 kHz/96 kHz in Figure 4.7) the delay is only 45 output samples (0.9 ms).

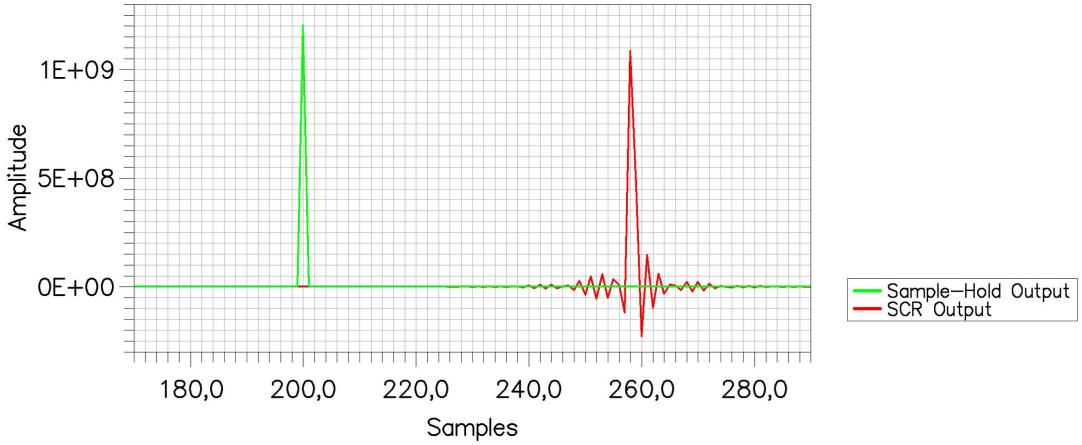


Figure 4.6: Impulse response, $f_{s,in} = 44.100\,kHz$ and $f_{s,out} = 48.000\,kHz$

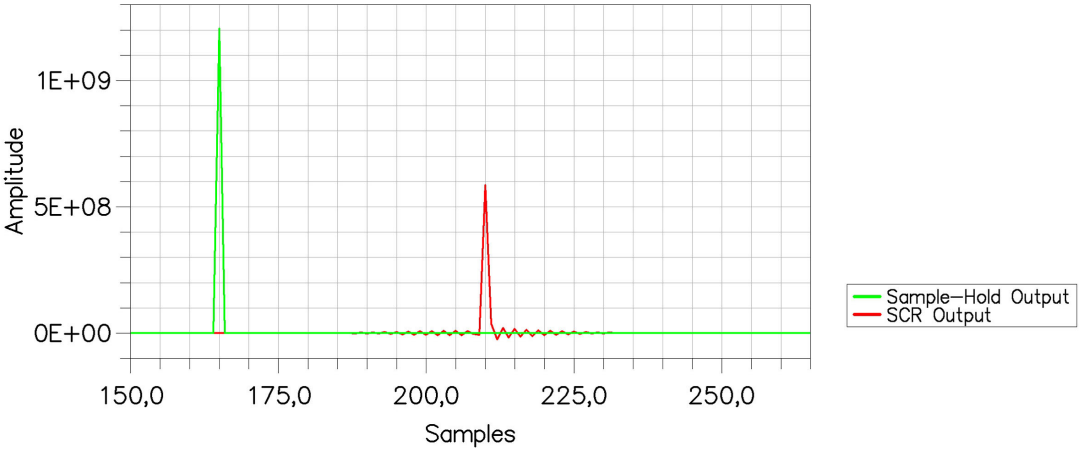


Figure 4.7: Impulse response, $f_{s,in} = 96.000\,kHz$ and $f_{s,out} = 48.000\,kHz$

After the observations in the time domain, some spectral measurements have been performed by using the *Audio Precision 2722* (AP2722). The test signal is again a 1 kHz sine with an amplitude of -3 dB_{FS}, which is provided by the digital generator of the AP2722. The DSP functions as an S/PDIF to analog converter. It's output gets analyzed by the AP2722 again, which performs a high order FFT. The first four plots show different input sample rates with active (blue) and bypassed[13] (red) sample rate converter. The Codec is running at $f_s = 48\,\text{kHz}$ for all measurements.

If the sample rates are unsynchronized but the same (48 kHz, Figure 4.9), there is no difference between the sample-hold and the SRC approach visible. Multiples of the Codec sample rate (96 kHz, Figure 4.11) are not producing any additional artifacts as well. Critical are none rational sample rate ratios (48 kHz/44.1 kHz and 48 kHz/88.2 kHz). Here not only the noise floor rises significantly, there are also none linear distortions with peaks between 35-55 dB under the signal level. These distortions are the result of twice used samples, shown in Figure 4.5. In every case the SRC removes all artifacts. The presence of the first and second harmonic (at 2 and 3 kHz) comes from non-linearities of the Codec.

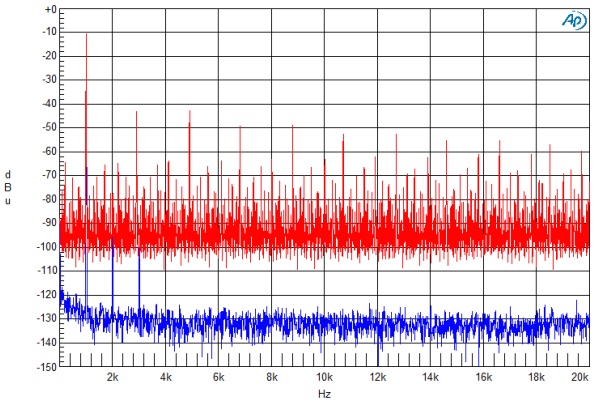


Figure 4.8: 1 kHz sine, f_s =44.1 kHz

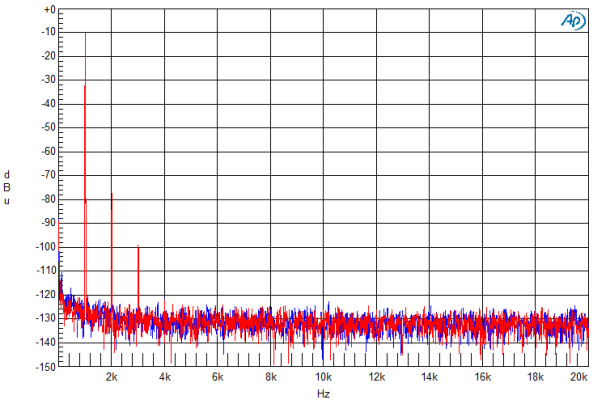


Figure 4.9: 1 kHz sine, f_s =48 kHz

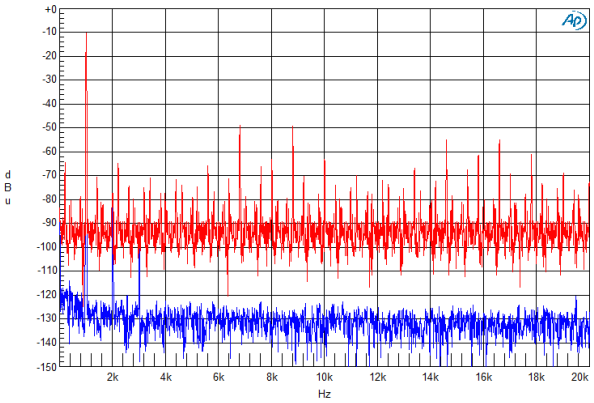


Figure 4.10: 1 kHz sine, f_s =88.2 kHz

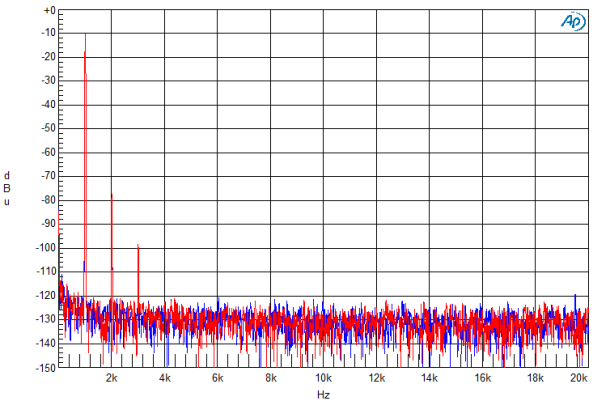


Figure 4.11: 1 kHz sine, f_s =96 kHz

The importance of using a sample rate converter gets clearly visible when the difference between input and output sample rate is very small. Due to tolerances in the components, this can become a practical issue. Figure 4.12 shows an over exaggerated example for sample rate mismatch. The blue spectrum shows the unconverted output of an $f_s = 48010\,\text{Hz}$ and the black spectrum shows the output of $f_s = 48500\,\text{Hz}$.

[13] If the SRC is inactive a sample-hold algorithm is performed.

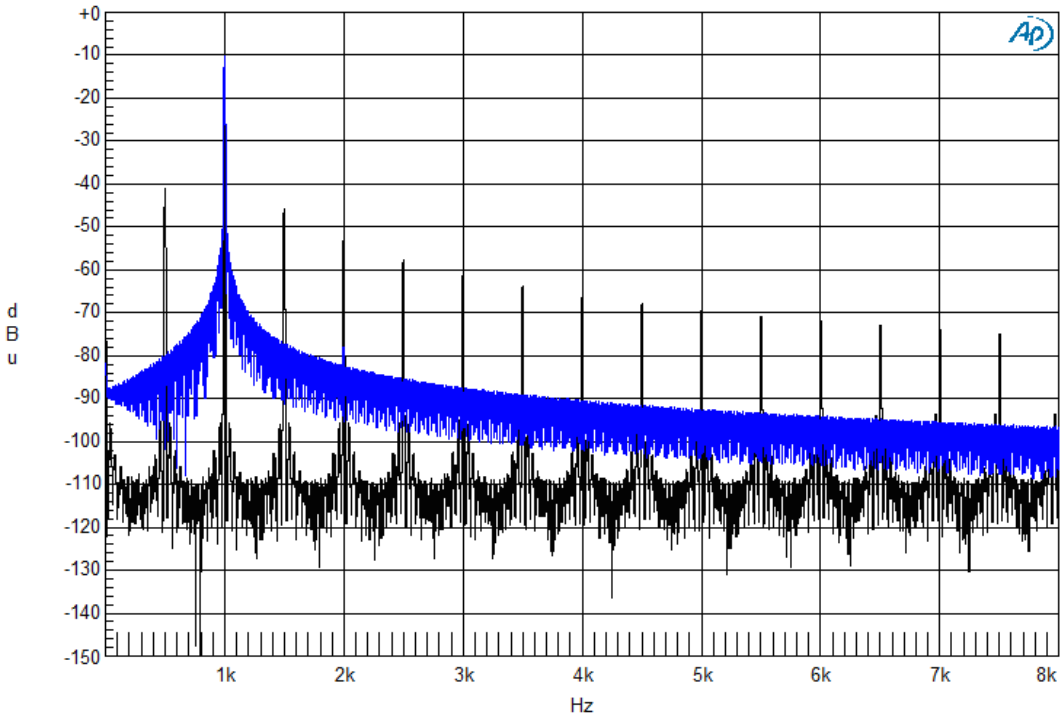


Figure 4.12: No SRC, f_s =47.990 kHz (blue), f_s =47.500 kHz (black)

This effect is due to the missing and twice used samples. The frequency of the twice used samples is the difference between the sample rates, here 10 Hz (blue) and 500 Hz (black). These steps in the signal introduce the peaks in the spectra at every multiple of the sample rate mismatch (for example 500 Hz, 1 kHz, 1.5 kHz,... in the black spectrum). The peaks in the blue spectrum are not well pictured because the FFT resolution was too low, it would look like the black one if the resolution was high enough.

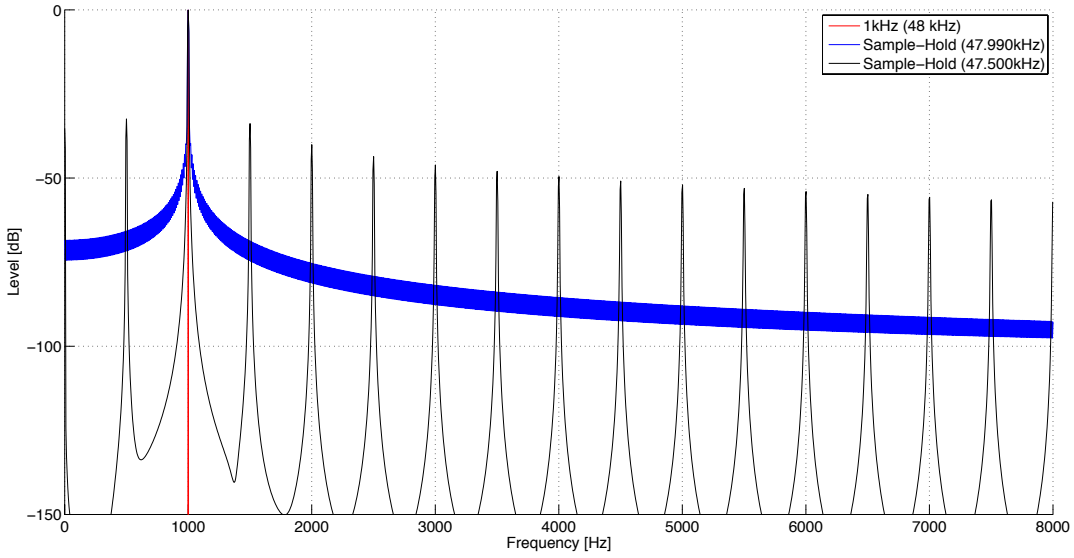


Figure 4.13: Sample-hold MATLAB simulation, f_s =47.990 kHz (blue), f_s =47.500 kHz (black)

Figure 4.13 shows a MATLAB simulation of the observed effects. Except for the noise the real system adds to the signals, the results are very similar. In the MATLAB simulation the effects of different sample rate mismatches can be studied easily without the need of expensive test-equipment.

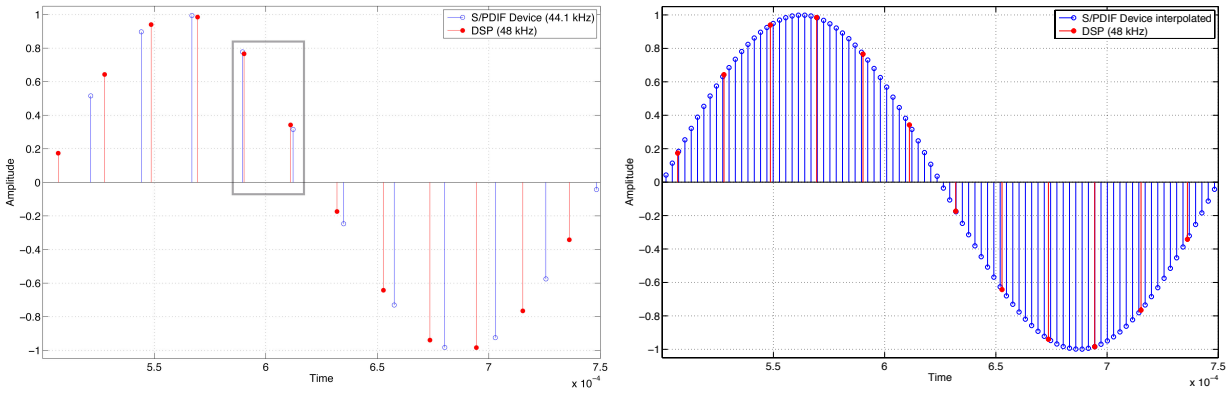
4.5 Exercise for the DAT-Laboratory

This section provides a short summary of SRC for the DAT-Laboratory and tasks to be performed during the course. The goal of this section is that the students understand the problem of working with various clock domains and see the operating principle and benefits of an SRC on the basis of a demo example. They will also learn how to set up a multirate system on the EZ Kit Lite and understand the signal flow inside the DSP.

4.5.1 Summary

Sample Rate Conversion (SRC) is a very important topic in digital signal processing. Due to technical and historical reasons, different applications use different sample rates (e.g. 44.1 kHz, 48 kHz,...). Not only can the clocks of two devices be different, they are most certainly unsynchronized. This means the samples arrive at different points in time but at the same frequency f_s. Because of component inaccuracies the sample rate is never perfectly stable, so eventually a sample gets lost or is processed twice. By using self-clocking formats (like S/PDIF, AES/EBU,...) jitter can become a problem as well, because the exact clock cannot be derived any more.

To avoid lost samples and add maximum flexibility to a multi-rate project, sample rate converters are used nowadays. An SRC is a device, which calculates intermediate samples to sync two clock domains. In Figure 4.14 one of three possible scenarios of sample rate conversion is shown: The input sample rate is smaller than the output rate ($f_{s,old} < f_{s,new}$). The digitalized signals in this example are even the same (4 kHz sine), to be more vivid. A zero order hold element, by means of saving one sample until the desired clock arrives, would periodically use one sample twice because there are 4100 samples missing every second. In practice one signal, in this example the blue one, runs through a high order interpolation before a sample hold is performed. Here an interpolation factor of 8 is used but in practice this factor can be as high as 2^{20} (1048576). Only this can guarantee an SNR of up to 125 dB.



(a) Unsynchronized signals

(b) Interpolated signal (L=8)

Figure 4.14: Synchronizing two signals by interpolation

The other two scenarios of sample rate conversion are: $f_{s,old} = f_{s,new}$ (but not synchronized) and $f_{s,old} > f_{s,new}$ (decimation). The first one works exactly the same, but decimation requires filtering as an additional processing. By decreasing the sampling frequency, the bandwidth (Shannon's frequency $f_s/2$) also gets smaller. All the signal components above $f_{s,new}/2$ need to be filtered by a low-pass to avoid aliasing.

4.5.2 Sample Hold vs. SRC Approach

1. Connect the 44.1 kHz S/PDIF device (NAD M5 CD/SACD Player) to the DSP board and load the project `SRC_Demo` onto the board. The LEDs should show an on-off pattern. Play some music on the S/PDIF device and listen to the results on any of the four stereo outputs of the DSP board. What do you hear? Is the signal clean?

2. Take a closer look at the software and find out what it does. The processing takes place in the Interrupt Service Routine (`ISR.asm`). What is the sample rate of the Codec and the S/PDIF device?

3. Comment out the bypass command in the `initSRC.asm` file to activate the SRC. Rebuild the project and listen to the result again. Do you still hear distortion? Which signals does an SRC module require (`initSRU.asm`) and what task does it perform?

4. Use a 1 kHz sine for the next task. Press Shift+F5 to stop the program and open a debug plot window (View → Debug Window → Plots → Restore: PlotSrcAndSampleHold.vps). In this window you see a visualization of the sample rate converter output and a sample-hold approach. What is the difference between the two signals? How large is the delay? Save the plot for your protocol.

5. Remove the comment in front of the impulse-train generator (`ISR.asm`) and build the program again. Stop it and view the results in the debug window. In this configuration the clock of the S/PDIF device is used but instead of the data, a digital impulse-train from a predefined ring-buffer gets passed through the SRC. What is the response of an FIR filter to a digital impulse? Count the delay in samples again and save the plot for you protocol.

6. Open the *SRC Mute and Ratio* debug window (Register → IOP → Sample Rate Converters → SRC Other). Look at the binary value of `SRC1_RATIO0`. This 15 bit register holds the $f_{s,out}/f_{s,in}$ ratio of SRC1 in a fixed-point format, shown in Table 4.7. Decode the value and write it into your protocol. What did you expect?

7. The SRC module is configured to automatically and softly mute the output whenever the sample rate changes to avoid clicks. Slowly disconnect the S/PDIF cable and listen what happens. Do you hear a fadeout? Disable auto-mute by uncommenting the respective lines in the `initSRC.asm` file and try it again.

8. Open the `SampleHoldSimulation.m` in MATLAB and run the first code-block by pressing Ctrl+Return. The plot window shows two signals with different sample rates (44.1 kHz and 48 kHz) and the solid line shows a sample-hold algorithm to sync both signals. One can clearly see the twice used samples which lead to non-harmonic distortions at the output. Save the plot and compare it with the plot of task 4.

9. Set the cursor into the next code-block and run it (Ctrl+Return) to see the distortions in the frequency domain. The red line indicates the input signal. Due to tolerances in the components, the sample rate mismatch between two devices can be only a few Hz in practice. By running the last code-block, the effects of slightly different sample rates can be observed. Where are the peaks? Hint: Look at the difference between input and output rate.

ratio	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
value	2^3	2^2	2^1	2^0.	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}

Table 4.7: SRC sample rate ratio

5 Comparison of two SRC Chips

There is a rather big selection of single chip SRC solutions on the market, which can cover even bigger sample rate ranges, SNR and special features than the integrated module of the ADSP-21369. This chapter will provide a comparison of two of these chips. The AD1895 from Analog Devices, which is very similar to the approach implemented in the DSP, and the Texas Instruments SRC4190. This chapter is based on their data sheets [12] [11].

5.1 Features

Beforehand it should be mentioned, that these chips have been chosen because of their very similar features and therefore good premises to be compared. The chips are even pin-compatible, so we will be able to see two different approaches to the same problem.

Both chips are headed for high-end audio applications with sample rates up to 192 kHz at a maximum resolution of 24 Bit. Also 16, 18 and 20 Bit data is accepted over the I^2S interface. The following table shows the features in a one by one comparison, the differences will be discussed below.

Feature	**AD1895**	**SRC4190**
Core Supply Voltage	3.3 V	3.3 V
Input/Output Sample Ratios	7.75:1 and 1:8	16:1 and 1:16
Minimum Input/Output Sample Rate	6 kHz	4 kHz
Maximum Input/Output Sample Rate	215 kHz	216 kHz
Dynamic range (SNR)	128 dB	128 dB
THD+N	up to -122 dB	up to -125 dB
Left-/Right Justified and I^2S	✓	✓
Master/Slave Mode	✓	✓
TDM Mode	✓	✓
Hardware controllable soft mute	✓	✓
On-chip MCLK generator	✓	✗
Power down mode	✗	✓

Table 5.1: Feature comparison

5.2 Range and Group Delay

As shown in Table 5.1, the Texas Instruments (TI) SRC has about twice the range at nearly the same SNR and THD+N. The other features are very similar. To achieve a greater frequency range, the chip needs more filter coefficients and therefore a bigger *Random Access Memory* (RAM) for the convolution. We now have a closer look at the filter characteristics. Although the data sheet of the TI SRC gives very little information about the implemented filters, we can compare the group delay for both decimation and interpolation.

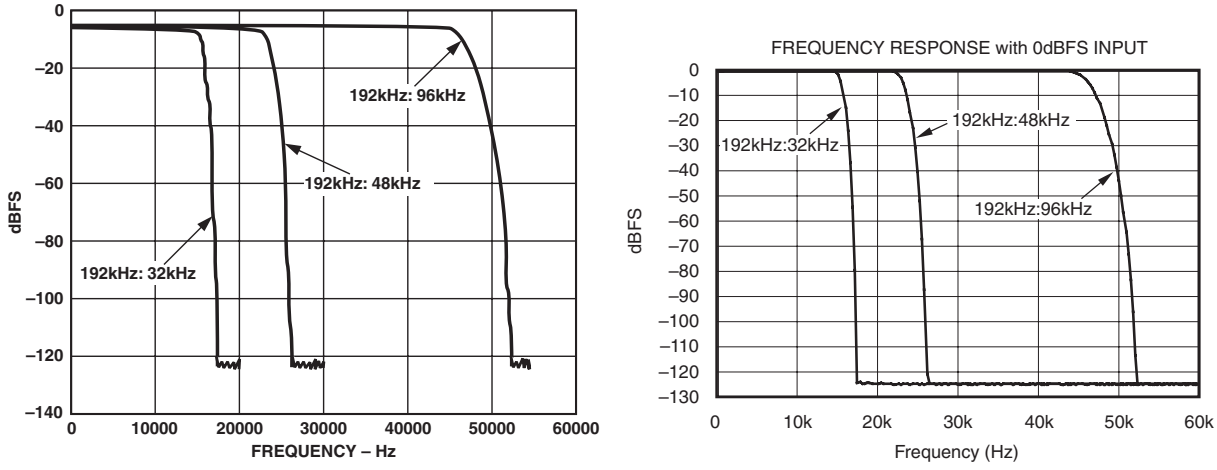
The normal interpolation group delay, given in the data sheet of the TI SRC, is $102.53125/f_{s,in}$ (in seconds), whereby the decimation is much faster with $36.47/f_{s,in}$ seconds. These delay times are still not too short. For example, at an input/output sample rate of 48 kHz/192 kHz, the interpolation delay is 2.14 ms and for 192 kHz/48 kHz the decimation group delay is 0.76 ms. This has to be kept in mind when designing a real time system. In some applications, a few milliseconds can make a difference.

The AD1895 uses a 64 tap FIR filter, which gets scaled up when the input sample rate rises over the output sample rate (as discussed in Chapter 3). The following equation, given in the data sheet of the AD1895, calculates the group delay for interpolation (5.1) and decimation (5.2):

$$GD = \frac{16}{f_{s,in}} + \frac{32}{f_{s,in}} \tag{5.1}$$

$$GD = \frac{16}{f_{s,in}} + \frac{32}{f_{s,out}} \tag{5.2}$$

For example, at an input/output sample rate of 48 kHz/192 kHz, the interpolation delay is 1 ms and for 192 kHz/48 kHz the decimation group delay is 0.75 ms. The group delays of both chips have the same order of magnitude and since these equations are often approximations, it is hard to say which chip performs better. Interesting though, is the first term of the equation. A 64 tap FIR filter would cause a group delay of 32 samples ($32/f_s$), like stated in the second term. The first term suggests a two stage decimation/interpolation approach, to minimize computational complexity, as discussed in Section 2.1.2.



(a) AD1895 (b) SRC4190

Figure 5.1: Frequency responses of the digital filters [12, p.12] [11, p.15]

Based on the filter characteristics shown in Figure 5.1, the cut-off frequencies for the respective decimation ratio and the noise floor can be compared. Both chips deliver great SNR well beyond the threshold of hearing. The following table compares the filters in a little more detail:

	AD1895		**SRC4190**		**Unit**
	Interpolation	Decimation	Interpolation	Decimation	
Passband	-	$0.4535 \cdot f_{s,out}$	$0.4535 \cdot f_{s,in}$	$0.4535 \cdot f_{s,out}$	Hz
Stop band	-	$0.5465 \cdot f_{s,out}$	$0.5465 \cdot f_{s,in}$	$0.5465 \cdot f_{s,out}$	Hz
Passband ripple	-	± 0.016	± 0.007	± 0.008	dB
Stop band attenuation	-	-125	-125	-125	dB

Table 5.2: Filter comparison

The filters are tuned to the exact same cut-off frequencies and they even have the same stop band attenuation. The AD1895 however, has a bit more pass band ripple of 0.016 dB. Anyway, 16 hundredth of a dB is far beyond the JND[14] as well. The data sheet of the AD1895 contains no separate information about the interpolation filter, but it can be assumed that it has the same specs as the decimation filter.

5.3 Operating Modes

Both Chips feature a soft-mute option. When the `MUTE_IN` pin is asserted high, the input signal gets linearly attenuated to -127 dB and when it's deasserted low, the attenuation is linearly decreased to 0 dB. This operation is controlled by a 12-bit counter which uses the `LRCLK_I` (Frame Sync) and takes 4096/`LRCLK_I` seconds. While the AD SRC has a dedicated `MUTE_OUT` pin which is automatically asserted high whenever the sample rate changes, the TI SRC features a `RDY` (ready) pin. This signal indicates whether the sample-rate-ratio has been detected or not. Both pins can be used to implement an auto-mute operation.

It is also possible to daisy-chain multiple SRC-Chips together to create a *Time Devision Multiplex* (TDM) signal. This is useful for multichannel audio like in home theater systems. The data-stream then is divided into frames and subframes as shown in Figure 5.2. Each subframe contains two samples (left and right = 64 bits) coming from one device. They are grouped together to a frame and sent to a DSP for example. The frame rate is equal to the output sample rate f_s and the `BCKO` is $N \cdot 64 \cdot f_s$, N beeing the number of chained devices.

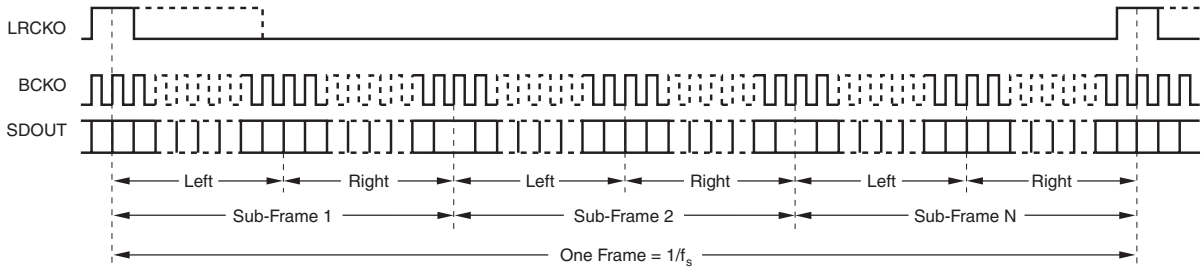


Figure 5.2: Functional block diagram [11, p.25]

For the TI SRC the maximum `BCKO` frequency is 27.136 MHz, so the number of daisy-chained devices is $(f_{BCKO}/f_s)/64 = 8.83$. The AD SRC also allows up to 8 chained devices. The

[14] Just Noticeable Difference

clocks can be generated by one of the SRC Chips (Master-Mode) or by the DSP, so all devices are operating in Slave-Mode. Both Chips feature a Bypass-Mode where the data gets passed through without being processed. This mode is useful if input and output sample rate are equal and synchronized or to pass through nonaudio data.

5.4 Conclusion

Keeping in mind that the AD SRC has a few more years on its back, the TI SRC was designed to be as compatible as possible. The differences in performance are more or less negligible unless an application needs a sample rate range which goes beyond the common values (44.1 kHz to 192 kHz). In this case the *Texas Instruments* SRC is the chip to be used. Still, both devices are very easy to use and need almost no setup since all configuration is done without any user interaction. The prices are between 3-4 $ a piece and therefore quit affordable for a high quality device.

A Bibliography

[1] Analog Devices Inc. *ADSP-21368 SHARC Processor Hardware Reference, Rev 1.0*. Norwood, MA. 2006.

[2] Analog Devices Inc. *ADSP-2136x Datasheet, Rev. E*. Norwood, MA. 2009.

[3] Philips Semiconductors. *I^2S Bus Specification*. 1996.

[4] Richard G. Lyons. *Understanding Digital Signal Processing*. Cloth, NY: Prentice Hall, 2004, pp. 507-567

[5] Analog Devices Inc. *EZ Kit Lite Evaluation System Manual, Rev. 2.1*. Norwood, MA. 2006.

[6] Wikipedia Inc. "Sample rate conversion". Internet: `http://en.wikipedia.org/wiki/Sample_rate_conversion` [13.09.2013]

[7] Dr. Gerhard Graber. *Digitale Audiotechnik 1, VO Skriptum*. Course no. 441.056, University of Technology, Graz, 2009.

[8] Wikipedia Inc. "AES3". Internet: `http://en.wikipedia.org/wiki/AES3` [23.09.2013]

[9] Wolfgang Nemitz, Christoph Frank, Christian Nachbar. *Digital Audio Engineering Laboratory, Handout*. Course no. 441.055, University of Technology, Graz, Apr. 2011.

[10] Analog Devices Inc. *AD1835 Datasheet, Rev. A*. Norwood, MA, 2003.

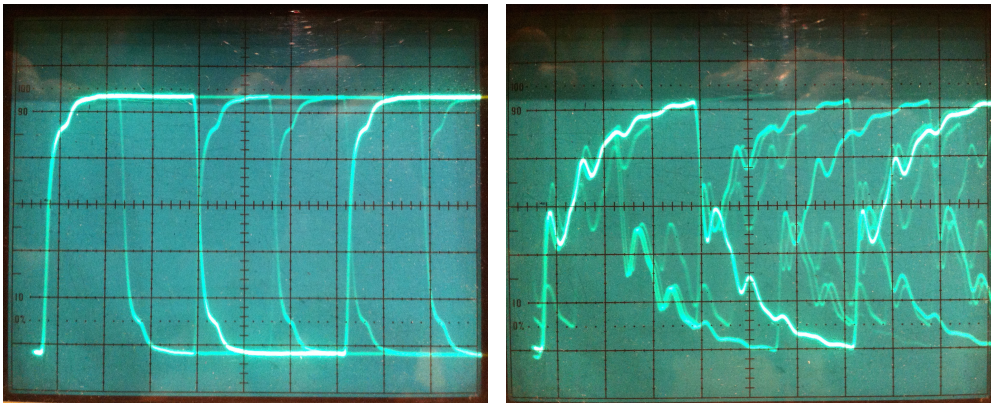
[11] Texas Instruments. *SRC4190 Datasheet*. Dallas, TX, 2009.

[12] Analog Devices Inc. *AD1895 Datasheet, Rev. B*, Norwood, MA, 2002.

B Findings

During the software development process, some problems and solutions with the Visual DSP++ Software and the EZ-KIT-LITE have been found. This is a list of findings which may be useful for future projects and thesis in this environment.

- Using the regenerated clock from the S/PDIF signal for the DAC (no SRC) works fine until a sample rate of $f_s = 70\,\text{kHz}$, then spikes can be heard. With SRC there are no audible effects until $f_s = 200\,\text{kHz}$.
- The S/PDIF signal over the long audio cable (6m) has at 48 kHz pretty noisy and shallow edges. The designated S/PDIF cable performs well (Figure B.1).



(a) S/PDIF cable, 3m (b) Audio Cable, 6m

Figure B.1: Waveform of S/PDIF signal

- Setting up the Audio Precision: Start PC > Start Audio Precision > Start *Audio Precision 2700* software > maximize the *Digital I/O* window > maximize the *Digital Generator* window > connect female XLR to Chinch Adapter to the digital output.
- Renaming of a project should be done inside Visual DSP++, simply changing the name of the file has no effect.

- The Codec chip only supports the following sample rates: DACFS48, DACFS96, DACFS192, ADCFS48, ADCFS96.
- The sample rates for ADC and DAC of the Codec can be changed independently. The SPORTs however use CLK and FS of the ADC.
- The sample-hold software (sample held in one of the core registers) produces about two spikes per second. This could be caused by an interrupt conflict, where both routines try to read/write into the same register.
- An SPORT can only receive or transmit. A transmit SPORT clocks out the data written to its TX-register. A receive SPORT expects synchronous data and writes it into its RX-register.
- When enabled, both A and B transmit registers of an SPORT need to be filled with data, otherwise there is no output.
- By configuring the SPORT word length as SLEN32, no shift before and after the algorithm is needed. The sign-bit is then on the correct position because the Codec always sends 32 Bit words.
- SRC Ratio, Mute,.. are shown in one of the debug windows: Register > IOP > Sample Rate Converter > SRC Other. MUTE_EN is HIGH active. SRCx_RATIO0 is a 15 Bit fixed point number with 4 digits before the comma (a factor of 1 is equal to 2048 in integer format).
- In some cases the Codec settings transmitted via SPI are not correct. In the SPI sending routine the SPIF bit is checked. By checking the TXS bit, the problem vanished.
- All the required registers and definitions for the DSP are declared in the `def21369.h` file.
- The SRC bypass mode works as expected. It simply performs a sample-hold until the output clock requests a new sample. In the case $f_{s,in} < f_{s,out}$ non-linear distortions are audible.
- If auto-mute is enabled the fade out is audible during disconnecting the S/PDIF cable but there are still distortions. The soft- and hard-mute bits turn the signal off but can not be easily tested in debug mode.
- the group delay select signal is not user accessible as described in the hardware reference of the DSP.
- The groupdelay for different input rates and 48 kHz output rate is: 58 samples for $f_{s,in} = 44.1$ kHz, 54 samples for $f_{s,in} = 48$ kHz, 45 samples for $f_{s,in} = 88.2$ kHz and 45 samples for $f_{s,in} = 96$ kHz. This was determined by passing a digital impulse-train from a ringbuffer through the SRC. In bypass-mode the delay is 3 samples, independent of the input rate.
- If the digital generator of the Audio Precision 2722 is used in stereo, very high harmonic distortion on the first channel occurs (shown in Figure B.2). This problem can be solved by using just one output of the digital generator. The remaining two harmonics (Figure B.3) are caused by the Codec chip.

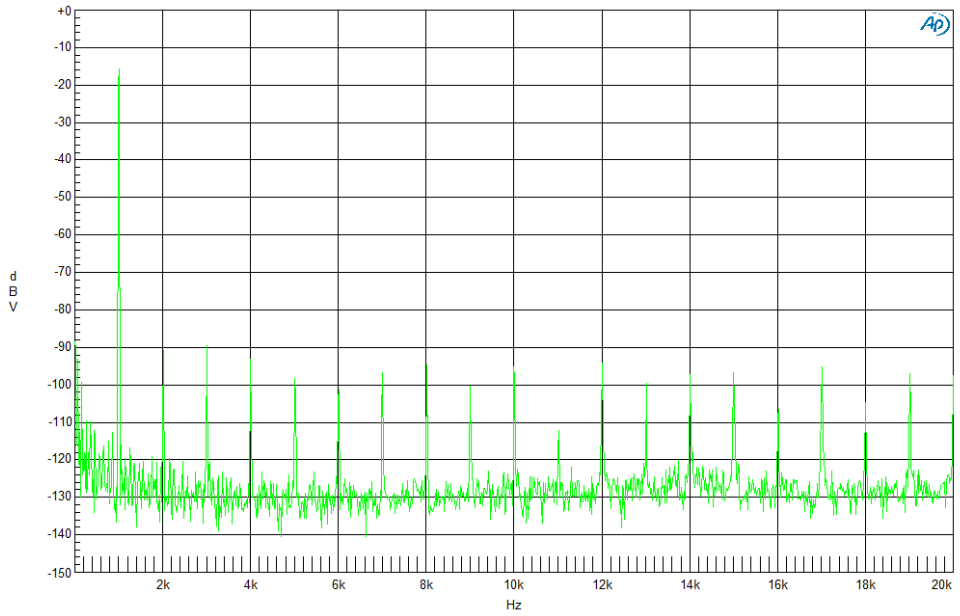


Figure B.2: Corrupt measurement

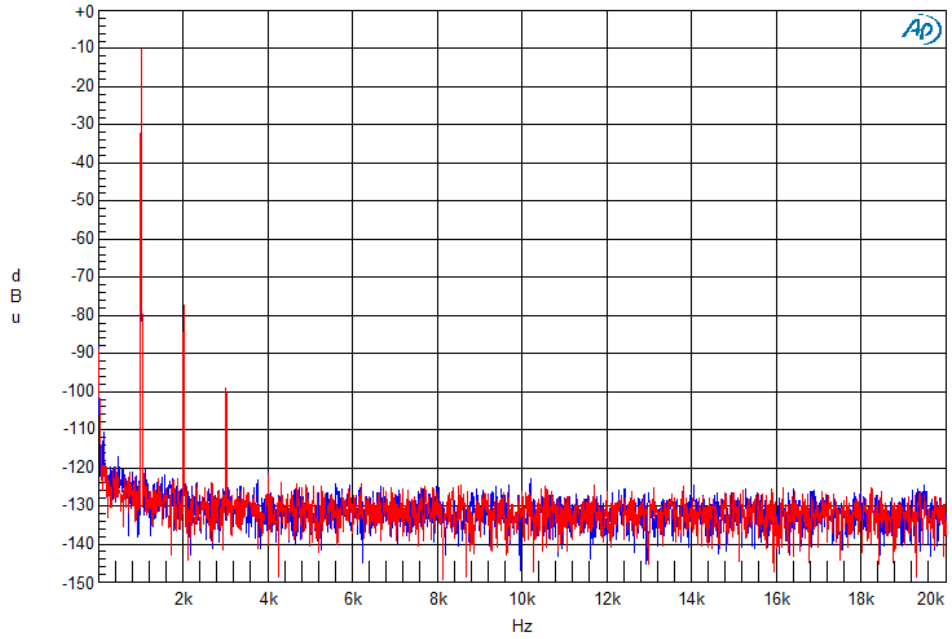


Figure B.3: Harmonic distortion of the Codec