Diplomarbeit

Software Defined Radio
with
Graphical User Interface

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter Leitung von

Dipl.-Ing. Michael Fischer,

Dipl.-Ing. Dr. Werner Keim,

und

Ao. Univ. Prof. Dr. Arpad L. Scholtz

E389

Institut für Nachrichtentechnik und Hochfrequenztechnik

eingereicht an der Technischen Universität Wien

Fakultät für Elektrotechnik und Informationstechnik

von

Thomas Reymund

9525013

Engelsdorfer Weg 5
A-3730 Eggenburg

Wien, im Oktober 2007

Abstract

Nowadays information is transmitted over the air in a multitude of standards (different modulation formats and various protocols). Each of these standards requires a different hardware and software setup. Software Defined Radio (SDR) brings together standardized hardware with the possibility to define its behaviour with software. Therefore, SDR greatly reduces the effort needed to apply a new standard.

The objective of this diploma thesis was to set up and implement a SDR which allows each telecommunication student of the Technical University of Vienna to experiment on actual radio transmission in the laboratory course *Labor Mobilfunk*. Students can easily alter the parameters of the radio link. By means of the results obtained, students can further their skills and knowledge in the area of mobile communications.

In this thesis a complete radio link was set up using dedicated SDR hardware and ordinary personal computers with audio soundcards. A transmitter and a receiver were developed in MATLAB with special focus on the user interface. The software operates in real-time using block processing. Furthermore, data packet detection and synchronization at the receiver side were implemented for pulse amplitude modulation.

Contents

Chapter 1

Introduction

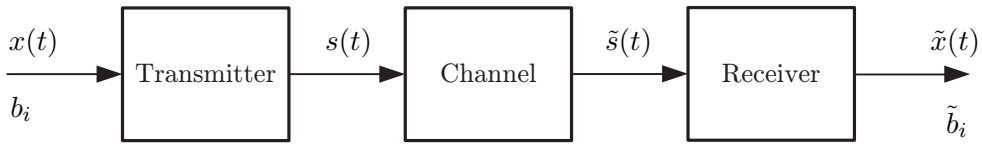


Figure 1.1: Components of a communication system.

Any communication system consists of the three basic components [1] (see Figure 1.1):

- The *Transmitter* transforms the analog information signal $x(t)$ or the digital data sequence b_i into an analog signal waveform $s(t)$, that is suited for transmission over the channel.
- The *Channel* is the physical medium that carries the transmit signal $s(t)$ from the transmitter to the receiver. The channel i.e. distorts the transmit signal in various ways.
- The *Receiver* processes the corrupted signal waveform $\tilde{s}(t)$ and transforms it into an analog signal $\tilde{x}(t)$ or a digital data sequence $\tilde{b}_i$, which will be hopefully not too different from the transmitted one.

In a digital communication system, parts of the transmitter (like source coding, channel coding, ...) and of the receiver (like detection, decoding, ...) have to be implemented in software. The realization of the whole transmitter and receiver in software is called *Software Radio (SR)*. Figure 1.2 shows a block diagram of a communication system implemented as a SR.

The SR-transmitter processes the digital information stream b_i to the digital representation $s[n]$ of the desired analog signal $s(t)$ via a Digital Signal Processor (DSP). The final conversion to the analog domain will be done by the Digital to Analog Converter (DAC).

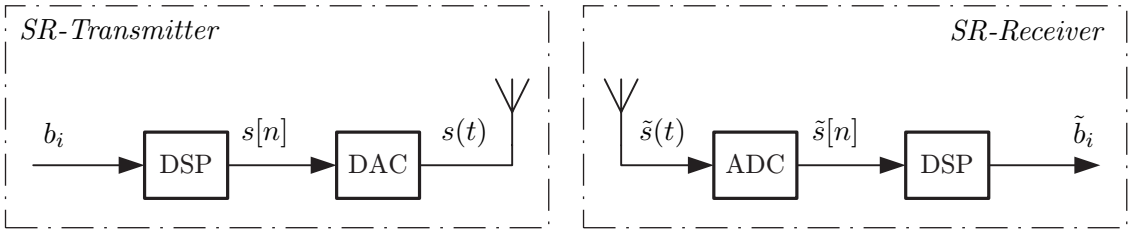


Figure 1.2: Concept of a Software Radio.

The SR-receiver converts the received analog signal $\tilde{s}(t)$ by an Analog to Digital Converter (ADC) to the digital representation $\tilde{s}[n]$. This signal will then be processed by an other DSP to a received digital information stream $\tilde{b}_i$.

The *advantages* [2] of the SR concept are

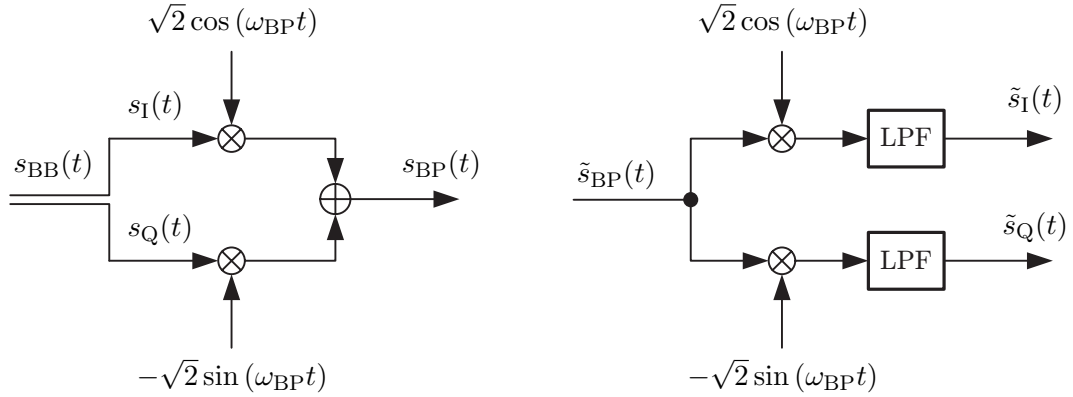
- any modification of the implementation can be very easily done (except hardware modifications).
- a software update can be done over the internet (or other networks). This will be cheaper than an update by a professional who has to come on-site.
- the introduction of standardized hardware platforms. This will increase the exchangeability between different manufacturers.

The main *disadvantage* [2] of this concept will be an extremely high power consumption compared to a conventional approach for a high frequency signal $s(t)$, as used in mobile communication systems (like in Universal Mobile Telecommunication System (UMTS) or Global System for Mobile Communications (GSM)). Processing of such a signal requires fast ADCs, DACs and DSPs. This yields to very high power consumption, which will be

1. unpractical for battery powered equipment, like a mobile phone, because of the shorter operation time, and
2. expensive, because of the increasing energy price

To take advantage of the SR-concept and to minimize the disadvantages, the Band Pass (BP)-signal of high frequency $s(t)$ will be downconverted by hardware to a much lower Intermediate Frequency (IF) f_{IF} and the software processing will be done on this signal. This is called *Software Defined Radio (SDR)*. A SDR reduces the requirements on the DACs, ADCs and DSPs and yields to a lower power consumption [3].

***BB→BP* transformation** Many practical channels do not support transmission of a Base Band (BB)-signal. Therefore a BB-signal has to be transformed



(a) ***BB→BP*** (b) ***BP→BB***

Figure 1.3: *BB→BP* and *BP→BB* transformations.

into a suitable BP-signal. This is shown in Figure 1.3a[1][2]. $s_{\mathrm{BB}}(t)$ will be a *complex*[3] BB-signal

$$s_{\mathrm{BB}}(t) = s_{\mathrm{I}}(t) + j s_{\mathrm{Q}}(t) \tag{1.1}$$

where $s_{\mathrm{I}}(t)$ is called the *Inphase (I)* component and $s_{\mathrm{Q}}(t)$ the *Quadrature (Q)* component of the BP-signal $s_{\mathrm{BP}}(t)$ [1].

$$\omega_{\mathrm{BP}} = 2\pi f_{\mathrm{BP}}$$

will be the center frequency of the BP-signal.

***BP→BB* transformation** The transformation from the BP-signal $\tilde{s}_{\mathrm{BP}}(t)$ to the BB-signal $\tilde{s}_{\mathrm{BB}}(t)$ is shown in Figure 1.3b. The Low Pass Filter (LPF) suppresses the undesired frequency band [1].

1.1 Communication System Overview

Figure 1.4 depicts the SDR communication system which has been developed in this diploma thesis. It further lists the desired frequency bands.

This diploma thesis will focus only on the Base Band Pulse Amplitude Modulation (PAM) computing and the *BB→IF* and *IF→BB* realization at the transmitter and receiver. The *IF→RF* and *RF→IF* realizations already exists.

1.1.1 The SDR-Transmitter

As shown in Figure 1.4 the SDR-transmitter implements a *heterodyne conversion* concept [3].

[1] $\otimes$ indicates a multiplication in figures
[2] $\oplus$ indicates a sum in figures
[3] indicates by double lines in figures

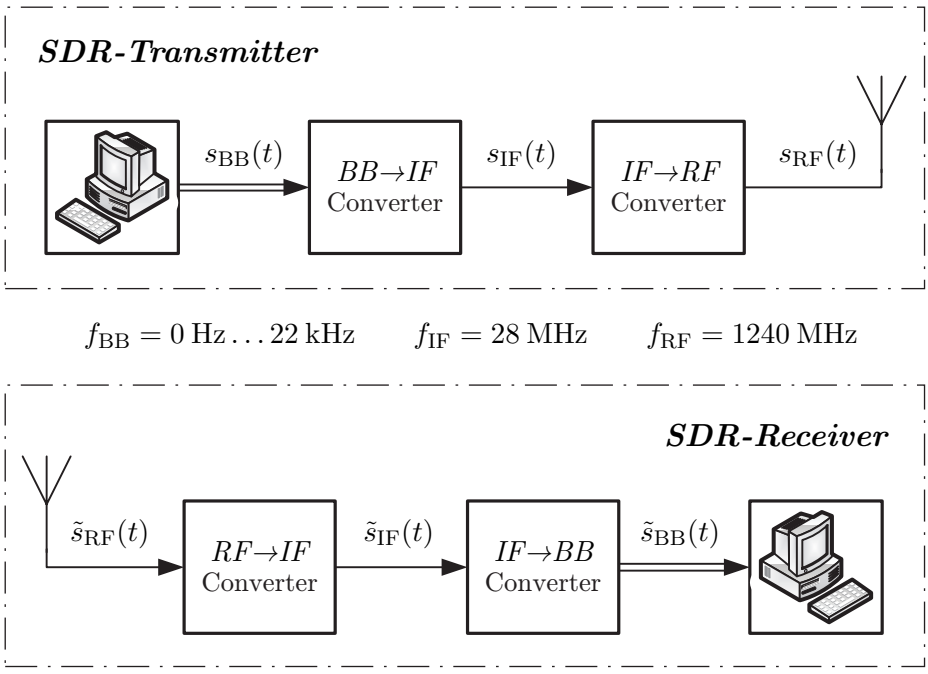


Figure 1.4: Overview of the communication system to be developed.

A computer generates a complex BB-signal $s_{\mathrm{BB}}(t)$ that will be fed into the first mixing stage. This stage converts the BB-signal up to a IF-signal $s_{\mathrm{IF}}(t)$ with the center frequency f_{IF}.

At the second mixing stage the IF-signal $s_{\mathrm{IF}}(t)$ will be finally upconverted to the desired Radio Frequency (RF)-signal $s_{\mathrm{RF}}(t)$ with the desired center frequency f_{RF}.

At the end of the transmitter, the RF-signal will be fed to an antenna to be transmitted over the given channel.

1.1.2 The SDR-Receiver

The SDR-receiver also implements a heterodyne concept [3].

An antenna receives a BP-signal $\tilde{s}_{\mathrm{RF}}(t)$. At the first mixing stage the RF-signal will be downconverted to the IF-signal $\tilde{s}_{\mathrm{IF}}(t)$ and at the second stage the IF-signal down to the complex BB-signal $\tilde{s}_{\mathrm{BB}}(t)$. Further processing of the complex BB-signal will be done on a computer.

This diploma thesis will focus only on PAM as BB computing and the *BB→IF* and *IF→BB* realization at the transmitter and receiver. The *IF→RF* and *RF→IF* realizations already exists.

1.2 Outline of the Thesis

The next chapter gives a theoretical overview of a base band pulse amplitude modulation transmission over a given channel.

Chapter 3 deals with the hardware used for the system implementation.

Chapter 4 discusses the software realization of the PAM transmitter and receiver shown at Chapter 2.

Chapter 2

Baseband Pulse Amplitude Modulation

The dependence of $s_{\mathrm{BB}}(t)$ to the information bits b_i to be transmitted is called digital modulation. There are various possible types of modulation (e.g. Pulse Amplitude Modulation (PAM), Frequency Shift Keying (FSK), Minimum Shift Keying (MSK), Gaussian Minimum Shift Keying (GMSK), Orthogonal Frequency Division Multiplexing (OFDM)), but this thesis deals only with the PAM.

Section 2.1 discusses the elementary structure of a PAM *transmitter*.

Section 2.2 shows the model of the transmission *channel*.

Section 2.3 discusses the elementary structure of a PAM *receiver*.

2.1 The PAM Transmitter

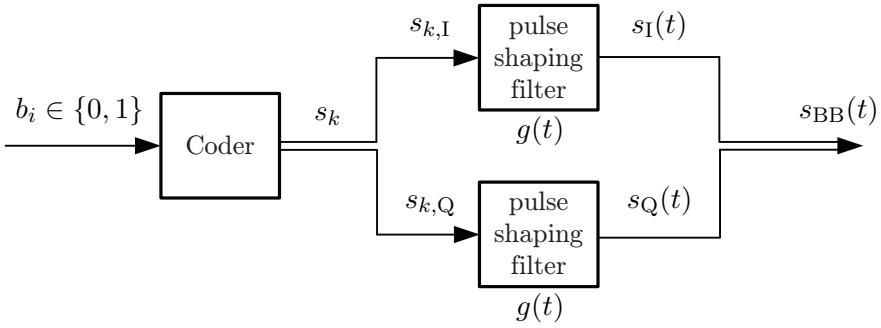


Figure 2.1: Elementary PAM *transmitter*.

An elementary transmitter for a PAM as shown in Figure 2.1 consists of the following components [1]:

Coder: The coder maps the incoming bit sequence b_i into a sequence of complex-valued symbols

$$s_k = s_{k,\mathrm{I}} + j s_{k,\mathrm{Q}}. \tag{2.1}$$

This is done by dividing the bit sequence into blocks of l-bits and mapping each block onto a symbol s_k taken from an complex-valued M_s-ary alphabet $\mathcal{A} = \left\{s^{(1)}, s^{(2)}, \cdots, s^{(M_\mathrm{s})}\right\}$, where

$$s^{(m)} = s_\mathrm{I}^{(m)} + j s_\mathrm{Q}^{(m)} \tag{2.2}$$

is the m-th symbol of the alphabet. The symbol rate becomes

$$R_\mathrm{sym} = \frac{R_\mathrm{bit}}{l}. \tag{2.3}$$

Pulse shaping filter: The pulse shaping filter defines the behavior of the transmission. The symbol components $s_{k,\mathrm{I}}$ and $s_{k,\mathrm{Q}}$ are passed through these *real-valued* Linear Time Invariant (LTI) filters with impulse response $g(t)$. Hence the real-valued I- and Q-components become

$$s_\mathrm{I}(t) = \sum_{k=1}^{K} s_{k,\mathrm{I}}\, g\left(t - kT_\mathrm{sym}\right) \tag{2.4}$$

$$s_\mathrm{Q}(t) = \sum_{k=1}^{K} s_{k,\mathrm{Q}}\, g\left(t - kT_\mathrm{sym}\right). \tag{2.5}$$

Equ. (2.4) and Equ. (2.5) inserted into Equ. (1.1) yield to

$$s_\mathrm{BB}(t) = \sum_{k=1}^{K} s_k\, g\left(t - kT_\mathrm{sym}\right). \tag{2.6}$$

To obtain a valid PAM signal $s_\mathrm{BB}(t)$, the pulse shaping filters have to be *equal* for both components and *real*.

2.2 The Channel

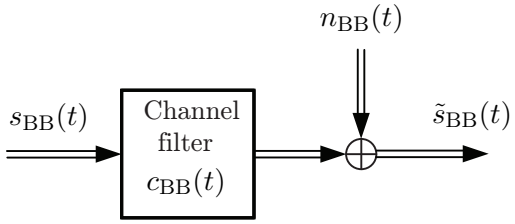


Figure 2.2: Representation of the equivalent BB-channel.

Figure 2.2 shows the equivalent BB-model of the channel used for the PAM transmission. The output of the channel $\tilde{s}_\mathrm{BB}(t)$ can be derived as[1]

$$\tilde{s}_\mathrm{BB}(t) = \left(s_\mathrm{BB} * c_\mathrm{BB}\right)(t) + n_\mathrm{BB}(t). \tag{2.7}$$

[1] $(\cdot * \cdot)$ denotes the convolution

For a BP-transmission the *real* valued LTI channel filter $c_{\mathrm{BP}}(t)$ and the *real* valued noise signal $n_{\mathrm{BP}}(t)$ can be transformed [1] via the $BP \rightarrow BB$ transformation to the *complex* BB-channel filter

$$c_{\mathrm{BB}}(t) = c_{\mathrm{BP}}(t) \cdot 2\cos(\omega_{\mathrm{BP}} t) - c_{\mathrm{BP}}(t) \cdot 2\sin(\omega_{\mathrm{BP}} t) \tag{2.8}$$

and to the *complex* valued BB-noise

$$n_{\mathrm{BB}}(t) = c_{\mathrm{BP}}(t) \cdot \sqrt{2}\cos(\omega_{\mathrm{BP}} t) - c_{\mathrm{BP}}(t) \cdot \sqrt{2}\sin(\omega_{\mathrm{BP}} t)\,. \tag{2.9}$$

The BP-channel noise will be assumed as a random process $N_{\mathrm{BP}}(t)$ with Power Spectral Density (PSD) $S_{\mathrm{N_{BP}}}(j\omega)$. The PSD for the equivalent BB-noise process becomes

$$S_{\mathrm{N_{BB}}}(j\omega) = 2\; S_{\mathrm{N_{BP}}}\left(j(\omega + \omega_{\mathrm{BP}})\right). \tag{2.10}$$

2.3 The PAM Receiver

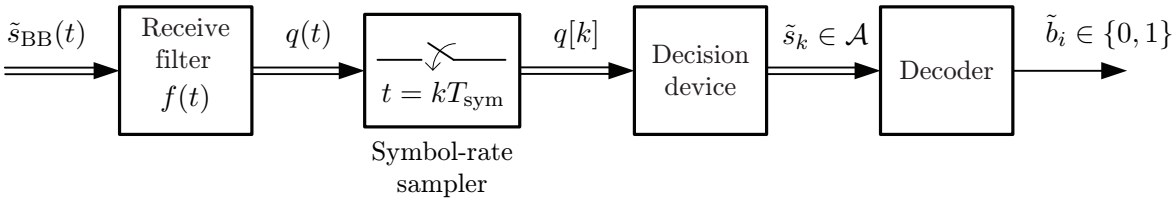


Figure 2.3: Elementary PAM *receiver*.

Figure 2.3 shows an elementary receiver for a PAM in baseband. It consists of the following components [1]:

Receive filter: The LTI receive filter filters the received PAM signal $\tilde{s}_{\mathrm{BB}}(t)$ with the in general *complex-valued* impulse response $f(t)$. It is called Matched Filter (MF), if the Signal to Noise Ratio (SNR) of the output signal $q(t)$ will be maximized. This is the case if $f(t)$ satisfies the constraint[2] [1]

$$f_{\mathrm{MF}}(t) = U h^*(-t) \tag{2.11}$$

where U is an arbitrary complex factor and

$$h(t) = (g * c_{\mathrm{BB}})(t) \tag{2.12}$$

will be the *complex-valued* so called *received pulse* comprising the *real-valued* pulse shaping filter $g(t)$ and the *complex* equivalent BB-channel response $c_{\mathrm{BB}}(t)$. The combination of pulse shaping, channel and receive filter will result in the *overall pulse*

$$p(t) = (g * c_{\mathrm{BB}} * f)(t) = (h * f)(t). \tag{2.13}$$

[2] $h^*(t)$ denotes the complex conjunction of $h(t)$

The output signal $q(t)$ can be derived to

$$q(t) = \sum_{k=1}^{K} s_k \, p\left(t - kT_{\text{sym}}\right) + z(t) \tag{2.14}$$

where

$$z(t) = (n * f)(t) \tag{2.15}$$

is the filtered channel noise.

Symbol-rate sampler: The symbol rate sampler samples the output of the receive filter $q(t)$ with a sampling rate equal to the symbol rate

$$q[k] = q\left(kT_{\text{sym}} + \tau_0\right). \tag{2.16}$$

τ_0 is a time offset accounting from the delay introduced by the pulse shaping filter $g(t)$, channel $c_{\text{BP}}(t)$ and receive filter $f(t)$. If the time offset will be neglected ($\tau_0 = 0$) and an infinite sequence of transmitted symbols s_l ($-\infty < l < \infty$) will be assumed for Equ. (2.14), Equ. (2.16) becomes

$$\begin{aligned} q[k] &= \sum_{l=-\infty}^{\infty} s_l \, p(kT_{\text{sym}} - lT_{\text{sym}}) + z(kT_{\text{sym}}) \\ &= \sum_{l=-\infty}^{\infty} s_l \, p[k-l] + z[k] \\ &= \sum_{v=-\infty}^{\infty} s_{k-v} \, p[v] + z[k] \qquad (2.17) \\ &= \underbrace{s_k \, p[0]}_{\text{desired symbol}} + \underbrace{s_{k-1} \, p[1] + s_{k-2} \, p[2] + \cdots}_{\text{past symbols}} + \\ &\quad + \underbrace{s_{k+1} \, p[-1] + s_{k+2} \, p[-2] + \cdots}_{\text{future symbols}} + z[k] \qquad (2.18) \end{aligned}$$

Equ. (2.18) shows, that $q[k]$ depends not only on the desired symbol s_k but also on the past symbols $s_{k-1}, s_{k-2}, \cdots$ and the future symbols $s_{k+1}, s_{k+2}, \cdots$. This is called Inter-Symbol Interference (ISI). ISI can be avoided if the *sampled* overall pulse $p[k]$ satisfies

$$p(kT_{\text{sym}}) = \delta[k] = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \tag{2.19}$$

Such pulses $p(t)$ are called *Nyquist pulse*.

Decision device: The decision device decides for each sampler output $q[k]$ on a "detected" symbol $\tilde{s}_k \in \mathcal{A}$. The decision was correct, if $\tilde{s}_k$ equals s_k, otherwise a symbol error has occurred. The number of wrong decisions will be minimized, if the SNR of $q[k]$ will be a maximum and hence a MF has to be used as receive filter.

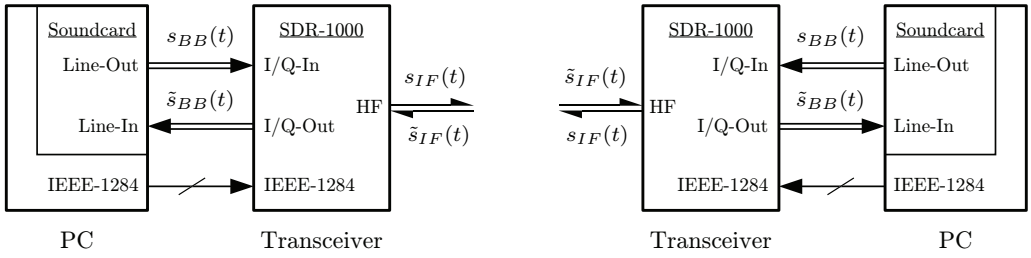
Decoder: The decoder inverts the encoding operation (bits → symbol mapping) done by the coder of the transmitter (see Section 2.1).

Chapter 4 discusses the real-time software implementation of the PAM transmitter and receiver.

Chapter 3

The SoftRadio Hardware System

This chapter describes the hardware system and the components used behind the implementation of the $BB \rightarrow IF$ and $IF \rightarrow BB$ transformation introduced in Section 1.1.



(a) SDR-Transmitter (b) SDR-Receiver

Figure 3.1: The SoftRadio hardware system.

Figure 3.1 illustrates the hardware system for the software implementation SoftRadio described in Chapter 4. There is no difference in the hardware configuration between the SDR-transmitter and -receiver, since the SDR-1000 is a transceiver. Therefore each PC/SDR-1000 combination can operate either as a transmitter or a receiver.

3.1 The Personal Computer

The exchange of the I- and Q-signals between the Personal Computer (PC) and the transceiver SDR-1000 are realized via a conventional audio sound card. The stereo signal (the left and the right channel) of the audio sound card will be used as I- and Q-signals.

The functions of the SDR-1000 are controlled through the IEEE-1284 compliant parallel port of the PC.

3.2 The Transceiver SDR-1000

The transceiver SDR-1000, designed by FlexRadio [4], will be used for the implementation of the $BB \rightarrow IF$ upconversion at the SDR-transmitter and the $IF \rightarrow BB$ downconversion at the SDR-receiver of the overall system depicted in Figure 1.4.

The transceiver SDR-1000 implements a 0 Hz to 65 MHz general coverage receiver with 1 W transmit capability on all licensed amateur radio bands.

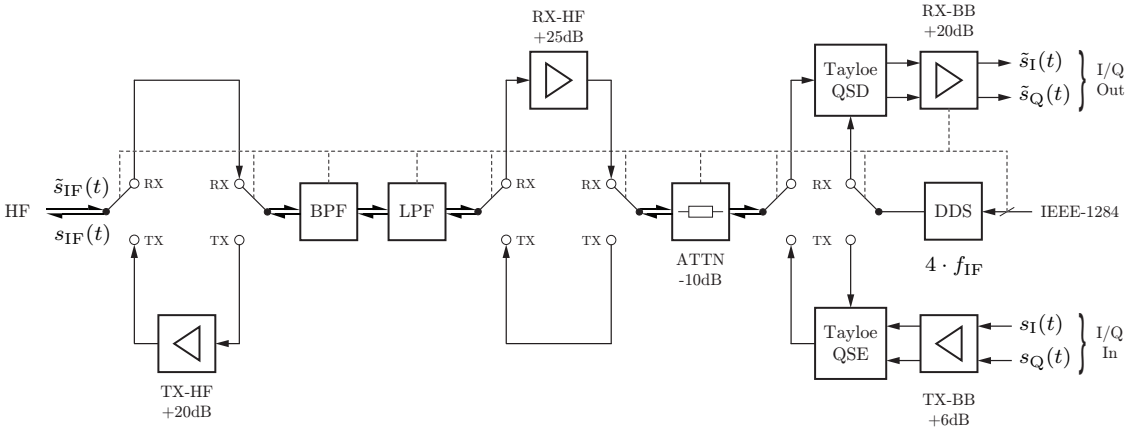


Figure 3.2: Block diagram of the transceiver "SDR-1000".

The block diagram shown in Figure 3.2 illustrates the $BB \rightarrow BP$ and $BP \rightarrow BB$ transformation realized at the SDR-1000. The *TX/RX switches* (controlled by the PC via the parallel port) select either the up- or the downconversion mode. In Figure 3.2 the downconversion mode is selected.

For the *downconversion* the received BP-signal $\tilde{s}_{IF}(t)$ at the HF-input will be first filtered by a *Band Pass Filter (BPF)* followed by a *LPF* to suppress the out of band noise. The bandlimited signal will then be amplified by the *RX-HF amplifier*. The amplified signal can be attenuated by -10 dB (controlled by the PC) before it gets to the *Tayloe Quadrature Sampling Detector (QSD)*. The Tayloe detector performs the actual downconversion of the BP-signal (for details see Section 3.2.1). Controlled by the PC, the *Direct Digital Synthesizer (DDS)* generates a carrier signal of $4 \cdot f_{IF}$ for this conversion (for details see Section 3.2.2). At the end the I- and Q-signals $\tilde{s}_I(t)$, $\tilde{s}_Q(t)$ can be amplified via the *RX-BB amplifier*.

At the *upconversion* the I- and Q-signals $s_I(t)$, $s_Q(t)$ will be amplified by the *TX-BB amplifier* before it gets to the *Tayloe Quadrature Sampling Exciter (QSE)*. The Tayloe exciter generates the BP-signal of the given I- and Q-signals (by means of the carrier of the DDS). The BP-signal can then be attenuated by the -10 dB *attenuator*. It will be filtered by a *LPF* followed by a *BPF* before it gets to the *TX-HF amplifier* and the High Frequency (HF)-output.

At the SDR-1000 the BPF can be chosen out of 6 different filter implementations and the LPF out of 9 different filters respectively via the PC.

Due to the design of the transceiver, it is however impossible to realize a full duplex connection.

3.2.1 The Tayloe Detector

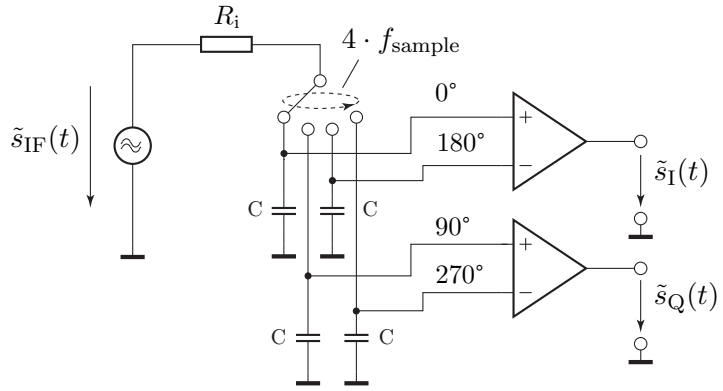


Figure 3.3: Schematic of a tayloe detector.

Figure 3.3 illustrates the single-balanced Tayloe detector [5, 6] used for the $BP \rightarrow BB$ transformation at the SDR-1000 (see Figure 3.2).

The Tayloe detector realizes four filtered *sample and hold* circuits [7] (resistor, switch, capacitor and amplifier). The switch rotates at four times the sampling frequency f_sample. If

$$f_\mathrm{sample} = f_\mathrm{IF}, \tag{3.1}$$

then the received, bandlimited BP-signal $\tilde{s}_\mathrm{IF}(t)$ will be sampled at the phases 0°, 90°, 180° and 270°. Sampling the signal $\tilde{s}_\mathrm{IF}(t)$ with the spectral representation $\tilde{S}_\mathrm{IF}(j\omega)$ at the frequency f_sample results to

$$S(j\omega) = \frac{1}{T_\mathrm{sample}} \sum_{m=-\infty}^{\infty} \tilde{S}_\mathrm{IF}(j(\omega - m \cdot \omega_\mathrm{sample})) \tag{3.2}$$

where $\tilde{S}_\mathrm{IF}(j(\omega - m \cdot \omega_\mathrm{sample}))$ are called aliases of the signal $\tilde{s}_\mathrm{IF}(t)$ (see Figure 3.4) [8]. If Equ. (3.1) is satisfied, the alias $m = -1$ appears at 0 Hz ($\hat{=}$ BB-signal). The resistor R_i and each capacitor C form a LPF that suppresses all the other aliases.

The I-component $\tilde{s}_\mathrm{I}(t)$ will be generated by differentially summing the samples of 0° and 180° and the Q-component by summing the samples of 90° and 270° respectively.

Such decoders are also called *Quadrature Sampling Detectors (QSDs).*

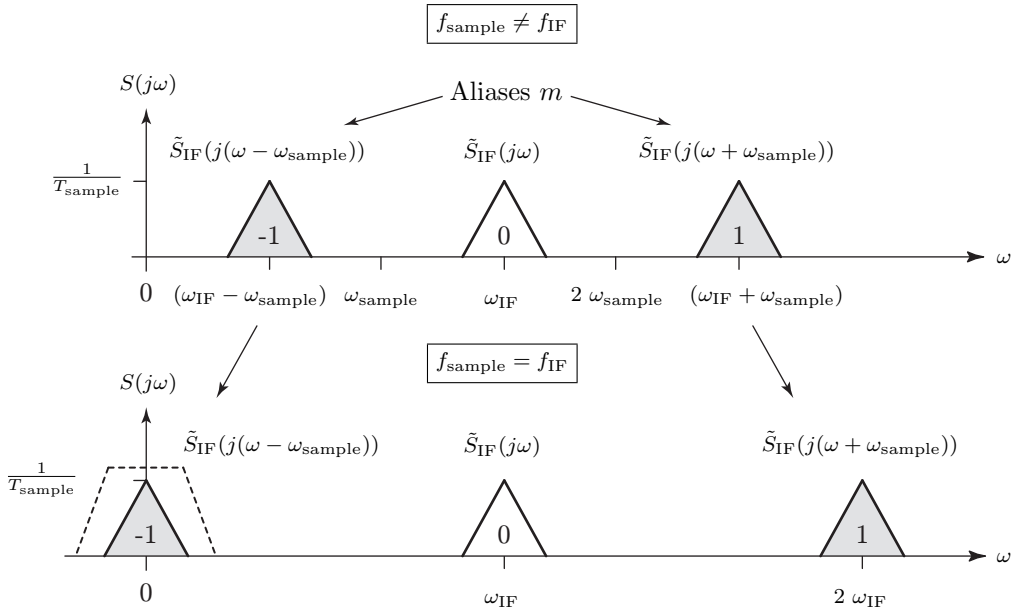


Figure 3.4: Aliases due to sampling a signal.

The main advantage of such a *homodyne*[1] receiver over one with a heterodyne concept is that an image frequency signal does not appear in the BB-signal [3].

The main disadvantage of the Tayloe detector is that the switches must be switched at four times the carrier frequency of the BP-signal.

With little modifications, the Tayloe detector can be also used for the upconversion (QSE) as depicted in Figure 3.5.

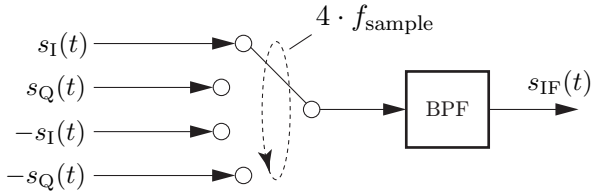


Figure 3.5: Schematic of a Quadrature Sampling Exciter.

3.2.2 The Direct Digital Synthesizer (DDS)

For the $BB \rightarrow BP$ and $BP \rightarrow BB$ transformation (see Figure 1.3 and Figure 1.4), the I-carrier

$$\sqrt{2}\cos(\omega_{\text{IF}} t)$$

[1] the BP-signal will be translated into the BB-signal without using an IF stage (see Section 1.1.2)

and the Q-carrier

$$-\sqrt{2}\sin(\omega_{\mathrm{IF}}t)$$

will be provided by the quadrature DDS AD9854 by Analog Devices [9]. To avoid crosstalk between the I- and Q-components $s_{\mathrm{I}}(t)$ and $s_{\mathrm{Q}}(t)$ at the BP-signal $s_{\mathrm{IF}}(t)$, a frequency- or a phase offset between the two carriers must be prevented [1]. This can be achieved very easily with a DDS.

Direct Digital Synthesis is a technique using digital data processing to generate a frequency- and phase-tunable output signal referenced to a *fixed-frequency precision clock* source [10]. A DDS can be implemented as shown in Figure 3.6.

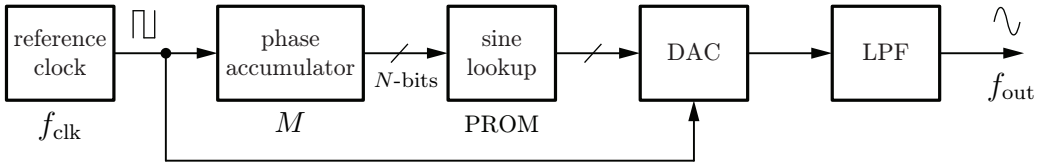


Figure 3.6: Block diagram of a DDS.

A Programmable Read Only Memory (PROM) is used as a *sine lookup table*, which stores 2^N samples of a complete cycle of a sinewave. The phase accumulator is a counter that increments its stored number by M each time it receives the clock pulse from the reference clock. Therefore every M-th sample will be addressed at the sine lookup table by the phase accumulator and presented to the DAC. This results to a analog sinewave of frequency

$$f_{\mathrm{out}} = \frac{M \cdot f_{\mathrm{clk}}}{2^N}.$$

To get a continuous sinewave, the phase accumulator overflows. A phaseshift can be realized by changing the start value of the phase accumulator. The precision of the generated sinewave depends on

1. the number of samples 2^N in the sine lookup table, and
2. the precision of the clock.

Changes to the value of M in the DDS architecture results in immediate and phase-continuous changes in the output frequency.

The next chapter discusses the software part of the transmitter and receiver realized by the program `SoftRadio.m` written in MATLAB.

Chapter 4

The Software Implementation `SoftRadio`

This chapter discusses the implementation of the software part of the transmitter and receiver of the communication system introduced in Section 1.1.

It will be realized by the program `SoftRadio.m` written in MATLAB. The following features are implemented:

- a ***real-time*** PAM *transmitter* ***and*** a ***real-time*** PAM *receiver*
- several ***real-time*** transmitting data symbol sources
- miscellaneous PAM symbol constellations (Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), Quadrature Amplitude Modulation with a symbol alphabet of n (nQAM))
- different pulse shaping and receive filters (rectangle and Root Raised Cosine (RRC))
- a Graphical User Interface (GUI)
- the controlling interface to the SDR-1000

4.1 Introduction

The *real-time software* implementation of the PAM transmitter and receiver in MATLAB, as described in Chapter 2, yields to the following realization concept:

- For an efficient generation of a *continuous*, discrete-time signal

  $$x[l] = x(lT_{\text{sample}}) \qquad -\infty \leq l \leq \infty$$

  a signal generation algorithm will be executed periodically. This algorithm produces a N-samples signal block

  $$x_{\text{N}}^{\{i\}}[n] = (x[i\cdot N+o],\ x[i\cdot N+o+1],\ x[i\cdot N+o+2],\ \cdots,\ x[i\cdot N+o+(N-1)])$$

of $x[l]$, where $i \geq 0$ indicates the signal block index, $0 \leq n < N$ the time index of x_{N} and o the offset between $x[l]$ and the starting index of the first block $x_{\mathrm{N}}^{\{0\}}[n]$ (see Figure 4.1).

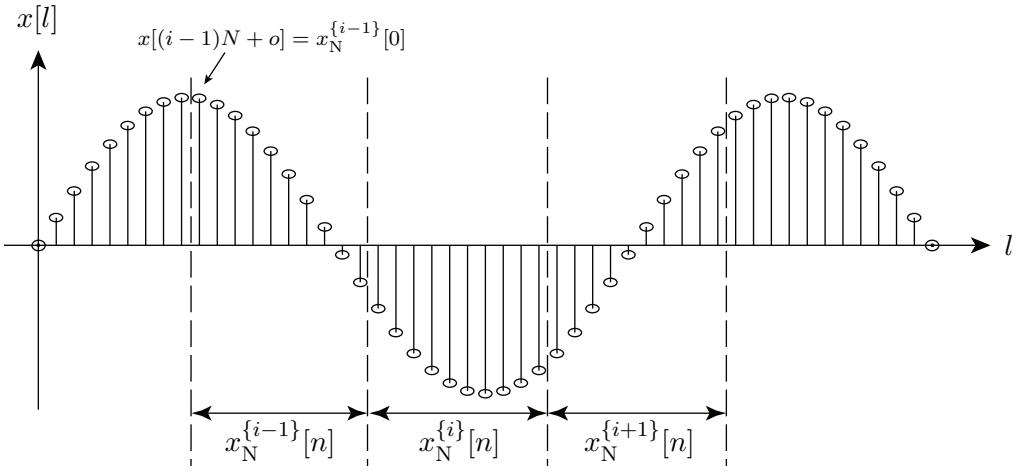


Figure 4.1: Decomposition of a continuous signal into signal blocks.

- The digital signal processing of a signal $x[l]$ has to be adapted to a periodic processing of the current signal block $x_{\mathrm{N}}^{\{i\}}[n]$ and the previous (stored) blocks $x_{\mathrm{N}}^{\{i-1\}}[n]$, $x_{\mathrm{N}}^{\{i-2\}}[n]$, $\cdots$, $x_{\mathrm{N}}^{\{i-u\}}[n]$. This is accomplished for example by the filtering via the overlap & add or overlap & save method[1].

4.1.1 The Overlap & Add Filtering Method

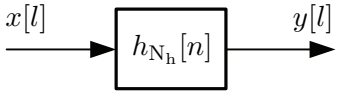


Figure 4.2: Filtering of a continuous signal.

This method implements the filtering of a *continuous* discrete-time signal $x[l]$ with a Finite Impulse Response (FIR) $h_{\mathrm{N_h}}[n]$ of length N_{h}

$$y[l] = (x * h_{\mathrm{N_h}})[l] \qquad -\infty \leq l \leq \infty \tag{4.1}$$

(depicted in Figure 4.2) as periodic processing of the signal blocks $x_{\mathrm{N}}^{\{i\}}[n]$ and $x_{\mathrm{N}}^{\{i-1\}}[n]$ [8] (see Figure 4.3).

Figure 4.4a shows $x[l]$ composed of $x_{\mathrm{N}}^{\{i-1\}}[n]$ and $x_{\mathrm{N}}^{\{i\}}[n]$ (for simplicity $x_{\mathrm{N}}^{\{i\}}[n]$ and $x_{\mathrm{N}}^{\{i-1\}}[n]$ have been chosen equally and constant). Figure 4.4b depicts the

[1] overlap & save will not be discussed

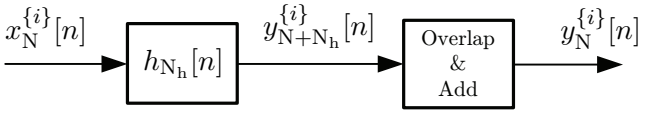


Figure 4.3: Overlap & Add filtering method.

filter impulse response $h_{\mathrm{N_h}}[n]$ of length N_{h} and Figure 4.4c the result $y[l]$ of Equ. (4.1).

At the processing of the i-th signal block, filtering of $x_{\mathrm{N}}^{\{i\}}[n]$ with the FIR $h_{\mathrm{N_h}}[n]$ results to

$$y_{\mathrm{N+N_h}}^{\{i\}}[n] = (x_{\mathrm{N}}^{\{i\}} * h_{\mathrm{N_h}})[n] \qquad 0 \le n \le N + N_{\mathrm{h}} - 1, \tag{4.2}$$

a signal of $(N + N_{\mathrm{h}})$ samples (see Figure 4.4d). The concatenation of $y_{\mathrm{N+N_h}}^{\{i-1\}}[n]$ and $y_{\mathrm{N+N_h}}^{\{i\}}[n]$

$$\hat{y}[l] = (y_{\mathrm{N+N_h}}^{\{i-1\}}[n],\ y_{\mathrm{N+N_h}}^{\{i\}}[n])$$

(see Figure 4.4e) is *not equal to* $y[l]$, due to the N_{h} samples of $y_{\mathrm{N+N_h}}^{\{i\}}[n]$ and $y_{\mathrm{N+N_h}}^{\{i-1\}}[n]$. This inconsistency can be eliminated by overlapping the last N_{h} samples of $y_{\mathrm{N+N_h}}^{\{i-1\}}[n]$ with the first samples of $y_{\mathrm{N+N_h}}^{\{i\}}[n]$ (depicted in Figure 4.4f).

To get a valid $y_{\mathrm{N}}^{\{i\}}[n]$, the first N_{h} signal samples of $y_{\mathrm{N+N_h}}^{\{i\}}[n]$ have to be corrected by adding the last N_{h} samples of $y_{\mathrm{N+N_h}}^{\{i-1\}}[n]$ and the last N_{h} samples have to be suppressed. They are stored for the processing of the $(i+1)$-th signal block (see Figure 4.4f,g).

$$y_{\mathrm{N}}^{\{i\}}[n] = \left(\begin{pmatrix} y_{\mathrm{N+N_h}}^{\{i\}}[0] \\ +y_{\mathrm{N+N_h}}^{\{i-1\}}[N] \end{pmatrix}, \cdots, \begin{pmatrix} y_{\mathrm{N+N_h}}^{\{i\}}[N_h - 1] \\ +y_{\mathrm{N+N_h}}^{\{i-1\}}[N + N_h - 1] \end{pmatrix}, y_{\mathrm{N+N_h}}^{\{i\}}[N_h], \cdots, y_{\mathrm{N+N_h}}^{\{i\}}[N-1] \right) \tag{4.3}$$

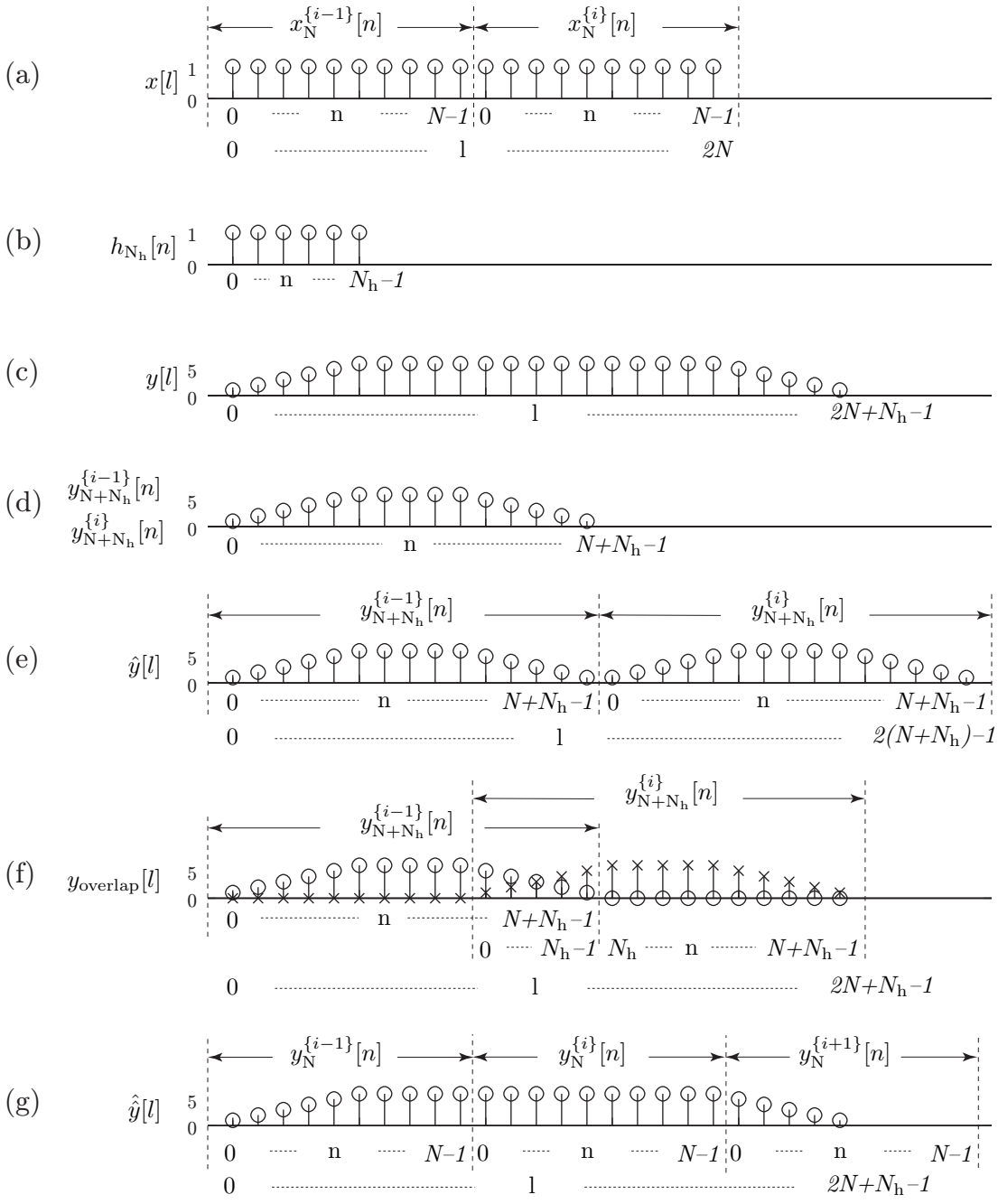


Figure 4.4: Signal Filtering of a continuous signal.

4.2 The SoftRadio PAM Transmitter

Section 2.1 explained the elementary structure of a BB-PAM transmitter. Figure 4.5 shows the modified *real-time software* structure of this transmitter, realized by SoftRadio and performed for every signal block.

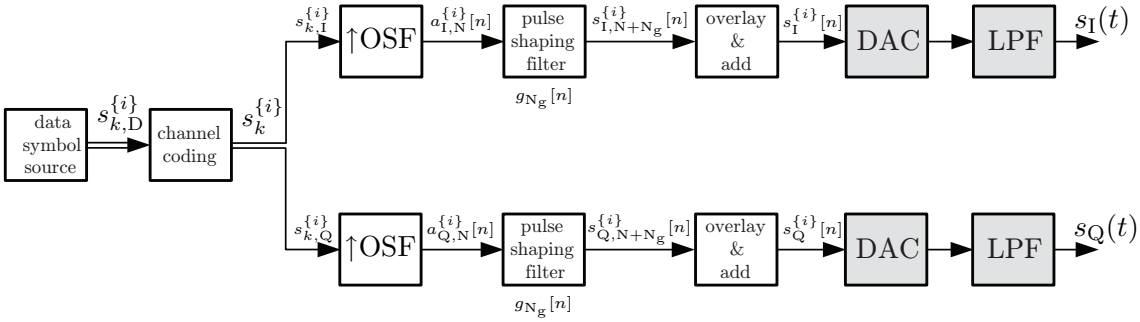


Figure 4.5: Blockdiagram of the BB-PAM *transmitter* realized by SoftRadio.

At the i-th signal block the following processing will be done:
The *data symbol source* generates a *complex* data symbol sequence

$$s_{k,\mathrm{D}}^{\{i\}} = (s_{1,\mathrm{D}}^{\{i\}},\ s_{2,\mathrm{D}}^{\{i\}},\ \cdots,\ s_{K_\mathrm{D},\mathrm{D}}^{\{i\}}) \quad 1 \le k \le K_\mathrm{D} \quad s_{k,\mathrm{D}}^{\{i\}} \in \mathcal{A} \tag{4.4}$$

of length K_D corresponding to the symbol rate R_sym. K_D will be equal for every signal block. The *channel coding* adds further status symbols for the receiver

$$s_{k,\mathrm{P}}^{\{i\}} = (s_{1,\mathrm{P}}^{\{i\}},\ s_{2,\mathrm{P}}^{\{i\}},\ \cdots,\ s_{K_\mathrm{P},\mathrm{P}}^{\{i\}}) \quad 1 \le k \le K_\mathrm{P} \quad s_{k,\mathrm{P}}^{\{i\}} \in \mathcal{A} \tag{4.5}$$

of length K_P to the data symbol sequence. K_P will again be equal for each block. For more information about the channel coder and the format of this *packet header* $s_{k,\mathrm{P}}^{\{i\}}$ see Section 4.2.2. The new symbol sequence

$$s_k^{\{i\}} = (\underbrace{s_{1,\mathrm{P}}^{\{i\}},\ s_{2,\mathrm{P}}^{\{i\}},\ \cdots,\ s_{K_\mathrm{P},\mathrm{P}}^{\{i\}}}_{\text{packet header } s_{k,\mathrm{P}}^{\{i\}}},\ \underbrace{s_{1,\mathrm{D}}^{\{i\}},\ s_{2,\mathrm{D}}^{\{i\}},\ \cdots,\ s_{K_\mathrm{D},\mathrm{D}}^{\{i\}}}_{\text{data symbols } s_{k,\mathrm{D}}^{\{i\}}}) \qquad 1 \le k \le K \tag{4.6}$$

of constant length

$$K = K_\mathrm{P} + K_\mathrm{D} \tag{4.7}$$

is called *symbol packet* (depicted in Figure 4.6) and will be separated into the I- and Q- symbol sequences $s_{k,\mathrm{I}}^{\{i\}},\ s_{k,\mathrm{Q}}^{\{i\}}$ (Equ. (2.1)). They will be processed at identical function trees (see Figure 4.5), hence the description of the successional processing steps will only be done for the I-symbol sequence $s_{k,\mathrm{I}}^{\{i\}}$.

Next, $s_{k,\mathrm{I}}^{\{i\}}$ will be upsampled to the sampling frequency f_sample of the shaping pulse $g_{\mathrm{N_g}}[n]$ at the upsampler with the *Over-Sampling Factor (OSF)* (see Section 4.2.1). The output of the upsampler

$$a_{\mathrm{I,N}}^{\{i\}}[n] = \sum_{k=1}^{K} s_{k,\mathrm{I}}^{\{i\}}\ \delta\left[n - (k-1)\cdot \mathrm{OSF}\right] \qquad 0 \le n \le N-1 \tag{4.8}$$

of length

$$N = K \cdot \mathrm{OSF} \tag{4.9}$$

will be filtered by the *pulse shaping filter* with the FIR $g_{\mathrm{N_g}}[n]$ of length N_g ($0 \leq n \leq N_\mathrm{g} - 1$). The output of the pulse shaping filter becomes

$$s^{\{i\}}_{\mathrm{I,N+N_g}}[n] = (a^{\{i\}}_{\mathrm{I,N}} * g_{\mathrm{N_g}})[n] = \sum_{k=1}^{K} s^{\{i\}}_{k,\mathrm{I}}\; g_{\mathrm{N_g}}\left[n - (k-1)\cdot \mathrm{OSF}\right] \qquad 0 \leq n \leq N+N_\mathrm{g}-1. \tag{4.10}$$

Due to the fact that $a^{\{i\}}_{\mathrm{I,N}}[n]$ is a N-samples cutout of the much longer signal $a_\mathrm{I}[n]$, $s^{\{i\}}_{\mathrm{I,N+N_g}}[n]$ has to be processed by an *Overlap & Add* stage (see Section 4.1.1). The output of the Overlap & Add stage $s^{\{i\}}_{\mathrm{I,N}}[n]$ will be fed to the DAC and LPF of the audio sound card to get the continuous-time PAM I-signal $s_\mathrm{I}(t)$ for the SDR-1000 (see Figure 1.4).

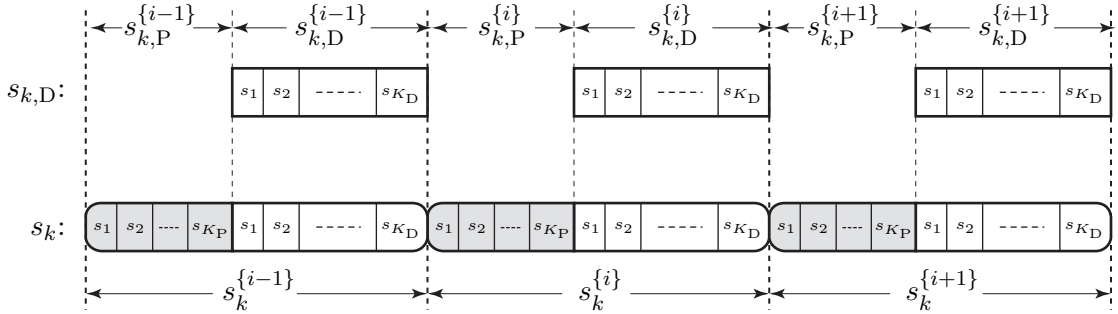


Figure 4.6: Symbol stream format transmitted by `SoftRadio`.

4.2.1 The Data Symbol Source

Figure 4.7 shows the blockdiagram of the data symbol source implemented in `SoftRadio`. The user can choose among the following sources for the i-th symbol packet:

Random Symbols generates a white random symbol sequence $s^{\{i\}}_{k,\mathrm{D,rand}}$ with user defined PSD

$$S_\mathrm{S}(e^{j\theta}) = S_0$$

of length K_D from the symbol alphabet $\mathcal{A}$.

Incremental Symbols generates a periodic linear symbol sequence

$$s^{\{i\}}_{k,\mathrm{D,inc}} = \left(s^{(1)}, s^{(2)}, \cdots, s^{(M_\mathrm{s})}, s^{(1)}, s^{(2)}, \cdots, s^{(M_\mathrm{s})}, s^{(1)}, s^{(2)}, \cdots, s^{(m)}\right)$$

of length K_D.

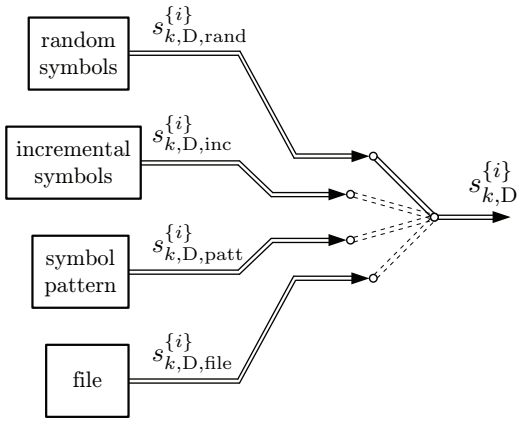


Figure 4.7: Blockdiagram of the data symbol source implemented by `SoftRadio`.

Symbol Pattern generates a periodic symbol sequence of length K_D

$$s^{\{i\}}_{k,\mathrm{D,patt}} = (s_{k,\mathrm{pattern}}, s_{k,\mathrm{pattern}}, \cdots, s_{k,\mathrm{pattern}})$$

of an user defined symbol pattern, e.g.

$$s_{k,\mathrm{pattern}} = (s^{(5)}, s^{(3)}, s^{(3)}, s^{(M_\mathrm{s}-3)}, \cdots, s^{(1)}).$$

File converts a file into I_packets data symbol sequences $s^{\{i\}}_{k,\mathrm{D,file}}$ of length K_D.

For a valid *multirate* signal processing (R_sym & f_sample), the symbol rate R_sym must satisfy the condition

$$R_\mathrm{sym} = \frac{f_\mathrm{sample}}{\mathrm{OSF}} \quad \text{where} \quad \mathrm{OSF} \in \mathbb{N}. \tag{4.11}$$

f_sample represents the sampling frequency of the DAC at the transmitter or ADC at the receiver (see Figure 4.5 and Figure 4.14 respectively) and OSF the *integer* factor between R_sym and f_sample. If the condition is *not* satisfied, OSF can be determined by[2]

$$\mathrm{OSF} = \left\lfloor \frac{f_\mathrm{sample}}{R_\mathrm{sym}} \right\rfloor \qquad \mathrm{OSF} \in \mathbb{N}. \tag{4.12}$$

This will be the case at `SoftRadio`, where the value of f_sample is constant and the user is allowed to enter an arbitrary value $R_\mathrm{sym,user}$ for R_sym. Hence the data symbol source generates a symbol sequence at the symbol rate

$$R_\mathrm{sym} = \frac{f_\mathrm{sample}}{\left\lfloor \frac{f_\mathrm{sample}}{R_\mathrm{sym,user}} \right\rfloor}, \tag{4.13}$$

which satisfies Equ. (4.11).

[2] $\lfloor x \rfloor$ floor function (largest integer less than or equal to x)

4.2.2 The Channel Coding

The major task of the channel coding of SoftRadio is to add status symbols to the data symbol stream $s_{k,\mathrm{D}}$ generated by the data symbol source (see Section 4.2). The status symbols will be required at the SoftRadio PAM receiver.

The channel coding generates a sequence of status symbols $s_{k,\mathrm{P}}^{\{i\}}$. Each status symbol sequence $s_{k,\mathrm{P}}^{\{i\}}$ depends on the corresponding $s_{k,\mathrm{D}}^{\{i\}}$. The format of the status symbol sequence (packet header) is depicted in Figure 4.8.

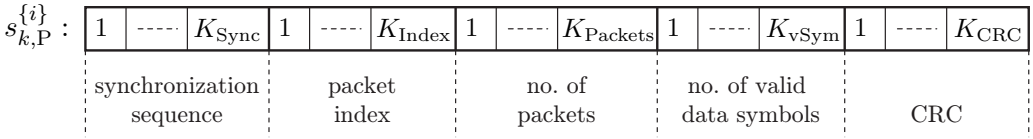


Figure 4.8: Format of the packet header used in SoftRadio.

Synchronization Sequence: Due to the time delay introduced by the channel, the filters and the different clock oscillators at the transmitter and receiver, the beginning of the symbol packet $\tilde{s}_k^{\{i\}}$ in the received signal samples $\tilde{s}_{\tilde{\mathrm{N}}}^{\{i\}}[n]$ is unknown and must be detected at the receiver side (see Section 4.3). This can be done by introducing a predefined symbol sequence

$$s_{k,\mathrm{Sync}}^{\{i\}} = (s_{1,\mathrm{Sync}}^{\{i\}},\ s_{2,\mathrm{Sync}}^{\{i\}},\ \cdots,\ s_{K_{\mathrm{Sync}},\mathrm{Sync}}^{\{i\}}) \qquad 1 \le k \le K_{\mathrm{Sync}} \tag{4.14}$$

of length K_{Sync}, which identifies the beginning of the symbol packet $s_k^{\{i\}}$. Therefore the synchronization sequence has been chosen *equal* for every symbol packet

$$s_{k,\mathrm{Sync}}^{\{i\}} = s_{k,\mathrm{Sync}}^{\{i-1\}} \qquad \forall i$$

and is also *known* by the receiver.

For an easier detection, a pseudo-orthogonal synchronization sequence [11] of only two symbols

$$s_{k,\mathrm{Sync}} \in \left\{s^{(1)}, s^{(2)}\right\} \quad \text{where} \quad s^{(2)} = -s^{(1)}$$

from the symbol alphabet $\mathcal{A}$ will be used (for detailed information see Section 4.3).

Packet Index: Every symbol packet gets an index $\hat{i}$, to indicate the transmitted packet order. This index will be increased for every new symbol packet, so that the receiver can detect a packet loss.

The *Packet Index* $\hat{i}$ will be coded into

$$s_{k,\mathrm{Index}}^{\{i\}} = (s_{1,\mathrm{Index}}^{\{i\}},\ s_{2,\mathrm{Index}}^{\{i\}},\ \cdots,\ s_{K_{\mathrm{Index}},\mathrm{Index}}^{\{i\}}) \qquad 1 \le k \le K_{\mathrm{Index}} \tag{4.15}$$

where

$$\hat{i} = s^{\{i\}}_{1,\mathrm{Index}}\ M_\mathrm{s}^0 + s^{\{i\}}_{2,\mathrm{Index}}\ M_\mathrm{s}^1 + s^{\{i\}}_{3,\mathrm{Index}}\ M_\mathrm{s}^2 + \cdots + s^{\{i\}}_{K_\mathrm{Index},\mathrm{Index}}\ M_\mathrm{s}^{(K_\mathrm{Index}-1)}. \tag{4.16}$$

Due to Equ. (4.16) the packet index can be

$$0 \le \hat{i} \le \left(M_\mathrm{s}^{K_\mathrm{Index}} - 1\right).$$

If more than $M_\mathrm{s}^{K_\mathrm{Index}}$ packets will be transmitted, $\hat{i}$ starts again at 0.

Number of Packets: The number of packets I_Packets tells the receiver, how many symbol packets have been transmitted for a *finite* data symbol stream

$$s_{k,\mathrm{D}} = (s^{\{0\}}_{k,\mathrm{D}},\ s^{\{1\}}_{k,\mathrm{D}},\ \cdots,\ s^{\{I_\mathrm{Packets}\}}_{k,\mathrm{D}}) \tag{4.17}$$

(like a file for example).

I_Packets will be coded into

$$s^{\{i\}}_{k,\mathrm{Packets}} = (s^{\{i\}}_{1,\mathrm{Packets}},\ s^{\{i\}}_{2,\mathrm{Packets}},\ \cdots,\ s^{\{i\}}_{K_\mathrm{Packets},\mathrm{Packets}}) \qquad 1 \le k \le K_\mathrm{Packets} \tag{4.18}$$

similar to the packet index $\hat{i}$ in Equ. (4.16).

$\hat{i}$ always starts at zero at the beginning of a transmission.

An *infinite* transmission of data symbols (for example for a speech transmission) is indicated by

$$I_\mathrm{Packets} = 0.$$

Number of Valid Data Symbols: The number of valid data symbols K_vData tells the receiver, how many data symbols of the received packet are valid. This will be required for the transmission of a *finite* data symbol stream $s_{k,\mathrm{D}}$ as in Equ. (4.17), where the last packet

$$s^{\{I_\mathrm{Packets}\}}_{k,\mathrm{D}} = (\underbrace{s^{\{I_\mathrm{Packets}\}}_{1,\mathrm{D}},\ \cdots,\ s^{\{I_\mathrm{Packets}\}}_{K_\mathrm{vData},\mathrm{D}}}_{\text{last symbols of } s_{k,\mathrm{D}}},\ \underbrace{s^{\{I_\mathrm{Packets}\}}_{K_\mathrm{vData}+1,\mathrm{D}}\ \cdots,\ s^{\{I_\mathrm{Packets}\}}_{K_\mathrm{D},\mathrm{D}}}_{\text{symbols to fill up packet}}).$$

carries only K_vData data symbols, generated by the data symbol source. The last $(K_\mathrm{D} - K_\mathrm{vData})$ packet symbols do not carry any information, they just fill up the packet. For the reason of simplification, `SoftRadio` *does not* produce symbol packets of variable length K.

The *Number of Valid Data Symbols* K_vData will be coded similar to the packet index $\hat{i}$ in Equ. (4.16).

$$K_\mathrm{vData} = 0$$

means, that all symbols of the packet are relevant.

CRC: For a symbol error detection at the receiver, $s_{k,\mathrm{D}}^{\{i\}}$, $s_{k,\mathrm{Index}}^{\{i\}}$, $s_{k,\mathrm{Packets}}^{\{i\}}$ and $s_{k,\mathrm{vSym}}^{\{i\}}$ of a packet will be secured by introducing Cyclic Redundancy Check (CRC) symbols.

The *CRC*-value will be coded similar to the packet index $\hat{i}$ in Equ. (4.16).

The second task of the channel coding is *interleaving* [12] of the data symbol stream $s_{k,\mathrm{D}}$. Interleaving is used to protect the transmission against burst errors[3] and to avoid long intervals of equal symbols. A symbol sequence of equal symbols would result in a PAM-signal with Direct Current (DC)-offset, which cannot be transmitted due to the DC-suppression of the SDR-1000 and the audio sound card. This has not been implemented yet.

4.2.3 The Graphical User Interface

Figure 4.9 shows the *standard*[4] Graphical User Interface of `SoftRadio` as *transmitter*. The multitude of functions are combined into several function blocks, which are described in the following sections.

4.2.3.1 Digital Modulation

The *Digital Modulation* block (see Figure 4.10) controls the digital modulation of the data symbol stream adjusted at the *Data Symbol Source* block. It defines the symbol alphabet $\mathcal{A}$ and the symbol rate R_{sym}.

Field 1 lets the user choose between the following possible symbol constellations (i.e. alphabet $\mathcal{A}$)

- BPSK,
- QPSK,
- 16-QAM,
- 64-QAM,
- 256-QAM.

Field 2 lets the user choose, how he/she wants to specify the timing of the data symbol stream. This can be done either by the

- Symbol Rate R_{sym} or the
- Symbol Period T_{sym} or the
- Bitrate R_{bit}.

[3] a corruption of successional symbols

[4] for information about the advanced mode see Appendix, Section B

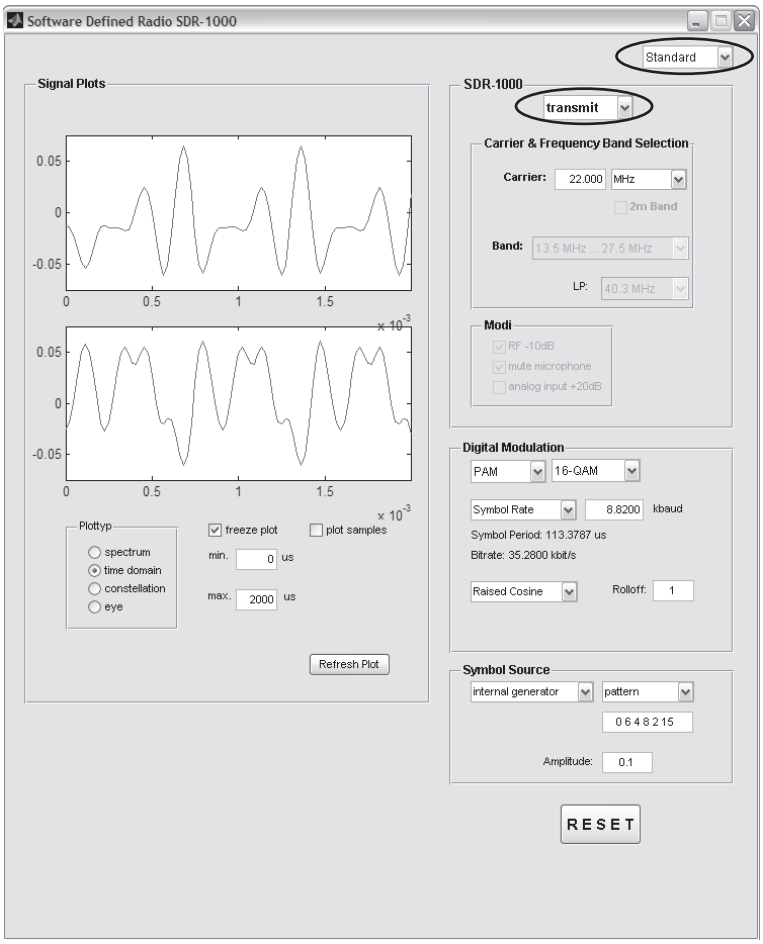


Figure 4.9: *Standard* GUI of the `SoftRadio` in **transmit** mode.

After entering an arbitrary value in **textfield 1**, `SoftRadio` determines the nearest valid value for the multirate signal processing (see the determination of the OSF in Section 4.2.1).

Field 3 lets the user choose between the following pulse shaping filters $f_{\mathrm{N_f}}[n]$:

- Rectangle and
- Root Raised Cosine (RRC).
  If RRC is chosen, **textfield 2** appears and lets the user specify the rolloff factor α ($0 \leq \alpha \leq 1$) [1] of the filter.

4.2.3.2 Data Symbol Source

The *Data Symbol Source* block (see Figure 4.11) controls the data symbol source discussed in Section 4.2.1.

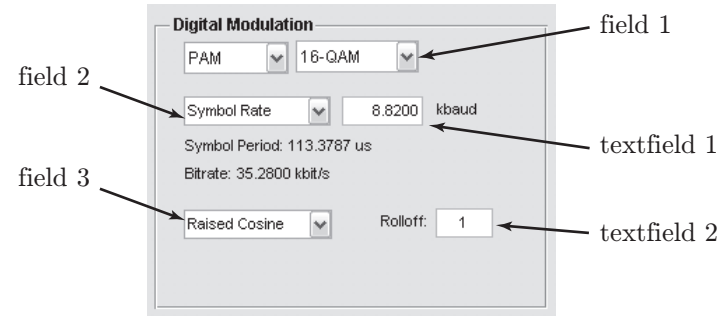


Figure 4.10: The Digital Modulation GUI.

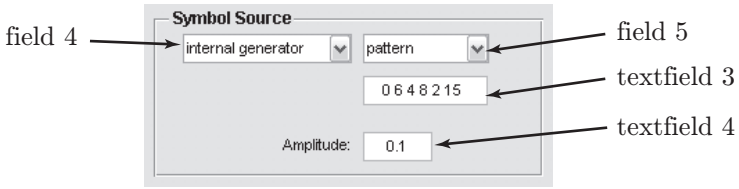


Figure 4.11: The Symbol Source GUI.

Field 4 lets the user choose between an *internal symbol generator* or a *file* as data source.

- For the *internal generator* **field 5** appears and lets the user select between
  - random symbols $s^{\{i\}}_{k,\mathrm{D,rand}}$ or
  - incremental symbols $s^{\{i\}}_{k,\mathrm{D,inc}}$ or a
  - symbol pattern $s^{\{i\}}_{k,\mathrm{D,patt}}$.

  If the *symbol pattern* has been chosen, **textfield 3** appears and lets the user specify the actual symbol pattern $s_{k,\mathrm{pattern}}$.
- For *file* **textfield 5** (not depicted in Figure 4.11) appears and lets the user specify the file location and name.

Textfield 4 lets the user enter the amplitude of the generated BB-signal (≤ 1).

4.2.3.3 SDR-1000

The *SDR-1000* block controls the functions of the transceiver SDR-1000 (see Figure 4.12).

Field 6 lets the user choose between *transmit* and *receive* mode of `SoftRadio` and SDR-1000.

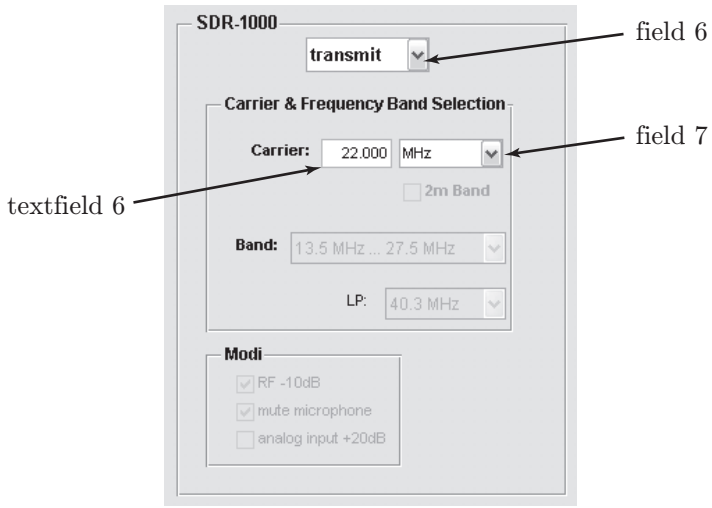


Figure 4.12: The SDR-1000 GUI.

Carrier & Frequency Band Selection: This function block controls the carrier frequency and the filter selection of the SDR-1000. The value of the IF-carrier will be defined by entering the base in **textfield 6** and choosing the exponential unit from **field 7**.

4.2.3.4 Signal Plots

The *Signal Plots* block controls the view of the implemented plots of the generated transmit symbol/signal stream (see Figure 4.13). The following plot types are implemented:

- symbol constellation plot of the transmitted data symbols $s_{k,\mathrm{D}}^{\{i\}}$,
- time domain plot of the generated I- and Q-components $s_{\mathrm{I,N}}^{\{i\}}[n]$, $s_{\mathrm{Q,N}}^{\{i\}}[n]$ (without the packet header),
- eye diagram[5] of I- and Q-components $s_{\mathrm{I,N}}^{\{i\}}[n]$, $s_{\mathrm{Q,N}}^{\{i\}}[n]$ (without the packet header),
- PSD of the generated complex BB-signal

$$s_{\mathrm{N}}^{\{i\}}[n] = s_{\mathrm{I,N}}^{\{i\}}[n] + j s_{\mathrm{Q,N}}^{\{i\}}[n].$$

The user can

- zoom in/out of the plots by means of the textfields,

[5] An eye diagram consists of many overlaid traces of small sections (in this case one symbol period) of the PAM signal [1]

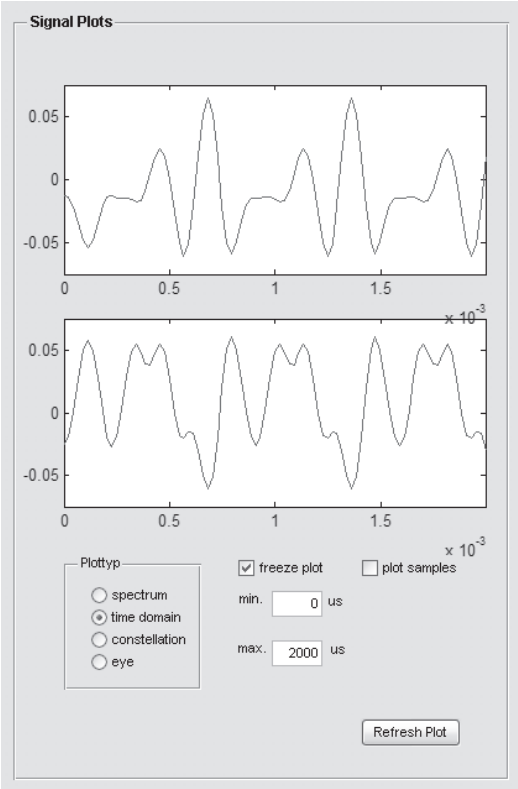


Figure 4.13: The Signal Plots GUI.

- freeze the plot or plot the samples of the signal via the checkboxes,
- request the update of the plots (if they are frozen) by using the *Refresh Plot* button.

4.3 The SoftRadio PAM Receiver

Section 2.3 demonstrated the elementary structure of a BB-PAM receiver. Figure 4.14 shows the modified *real-time software* structure of this receiver implemented in SoftRadio and performed for every signal block.

The continuous-time I- and Q-signals $\tilde{s}_{\mathrm{I}}(t)$, $\tilde{s}_{\mathrm{Q}}(t)$ from the SDR-1000 will be filtered by the audio sound card antialiasing LPF $\tilde{s}_{\mathrm{I,LP}}(t)$, $\tilde{s}_{\mathrm{Q,LP}}(t)$ and digitized by the ADC at the sampling frequency f_{sample}.

At the i-th signal block the following processing will be done:
The received $\tilde{N}$-samples cutout $\tilde{s}^{\{i\}}_{\mathrm{I},\tilde{\mathrm{N}}}[n]$ and $\tilde{s}^{\{i\}}_{\mathrm{Q},\tilde{\mathrm{N}}}[n]$ of the ADC output $\tilde{s}_{\mathrm{I}}[n]$ and $\tilde{s}_{\mathrm{Q}}[n]$ will be combined to the complex BB-signal samples $\tilde{s}^{\{i\}}_{\tilde{\mathrm{N}}}[n]$ and processed by the *packet detection*. The packet detection detects the transmitted packets $s^{\{i\}}_{\mathrm{N}}[n]$ in the received samples $\tilde{s}^{\{i\}}_{\tilde{\mathrm{N}}}[n]$ (for further information about the functionality

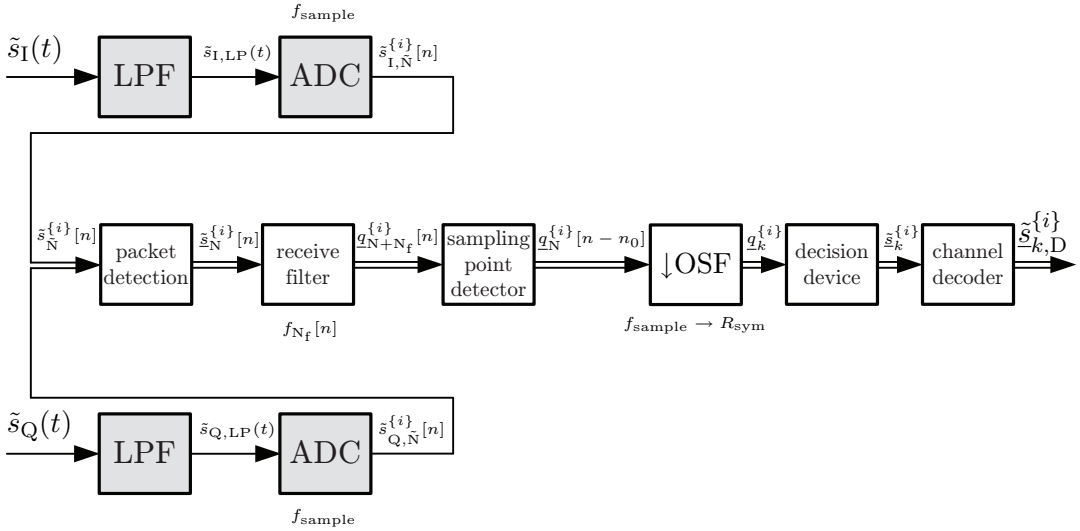


Figure 4.14: Blockdiagram of the BB-PAM receiver realized by `SoftRadio`.

of packet detection see Section 4.3.1). Due to the fact, that $\tilde{N} > N$ is chosen, at least one symbol packet can always be detected. The output of the packet detection results to

$$\underline{\tilde{s}}^{\{i\}}_{\text{N}}[n] = (\tilde{s}^{\{i-I_\text{N}\}}_{\text{N}_\text{lp}}[n], \tilde{s}^{\{i-I_\text{N}+1\}}_{\text{N}}[n], \cdots, \tilde{s}^{\{i-2\}}_{\text{N}}[n], \tilde{s}^{\{i-1\}}_{\text{N}}[n], \tilde{s}^{\{i\}}_{\text{N}_\text{fp}}[n]) \tag{4.19}$$

where

$$I_\text{N} = \left\lfloor \frac{\tilde{N}}{N} \right\rfloor$$

is the *number of detected packets*. $\tilde{s}^{\{i-I_\text{N}\}}_{\text{N}_\text{lp}}[n]$ are the last N_{lp} samples of the $\tilde{s}^{\{i\}}_{\text{N}}[n]$ packet received and processed at the *previous* signal block $(i-1)$ (see Figure 4.15). Equally $\tilde{s}^{\{i\}}_{\text{N}_\text{fp}}[n]$ are the first N_{fp} samples of $\tilde{s}^{\{i\}}_{\text{N}}[n]$.

After the packet detection, $\underline{\tilde{s}}^{\{i\}}_{\text{N}}[n]$ will be filtered by the *receive filter* with the FIR $f_{\text{N}_\text{f}}[n]$ of length N_f $(0 \leq n \leq N_\text{f} - 1)$. This filter reduces the noise to get a higher SNR at the input of the decision device which yields to a lower decision error rate. It will be a matched filter, if the channel can be assumed perfect, which means

$$c_{\text{BB}}(t) = c_0 \cdot \delta(t - \tau_0),$$

and the impulse response equals to the pulse shaping filter

$$f_{\text{N}_\text{f}}[n] = U \cdot g_{\text{N}_\text{g}}[n]$$

(see Section 2.3). The output of the receive filter

$$\begin{aligned} \underline{q}^{\{i\}}_{\text{N+N}_\text{f}}[n] &= \left((\tilde{s}^{\{i-I_\text{N}\}}_{\text{N}_\text{lp}} * f_{\text{N}_\text{f}})[n], (\tilde{s}^{\{i-I_\text{N}+1\}}_{\text{N}} * f_{\text{N}_\text{f}})[n], \cdots, (\tilde{s}^{\{i-1\}}_{\text{N}} * f_{\text{N}_\text{f}})[n], (\tilde{s}^{\{i\}}_{\text{N}_\text{fp}} * f_{\text{N}_\text{f}})[n]\right) \\ &= (q^{\{i-I_\text{N}\}}_{\text{N}_\text{lp}+\text{N}_\text{f}}[n],\ q^{\{i-I_\text{N}+1\}}_{\text{N+N}_\text{f}}[n],\ \cdots,\ q^{\{i-1\}}_{\text{N+N}_\text{f}}[n],\ q^{\{i\}}_{\text{N}_\text{fp}+\text{N}_\text{f}}[n]) \end{aligned} \tag{4.20}$$

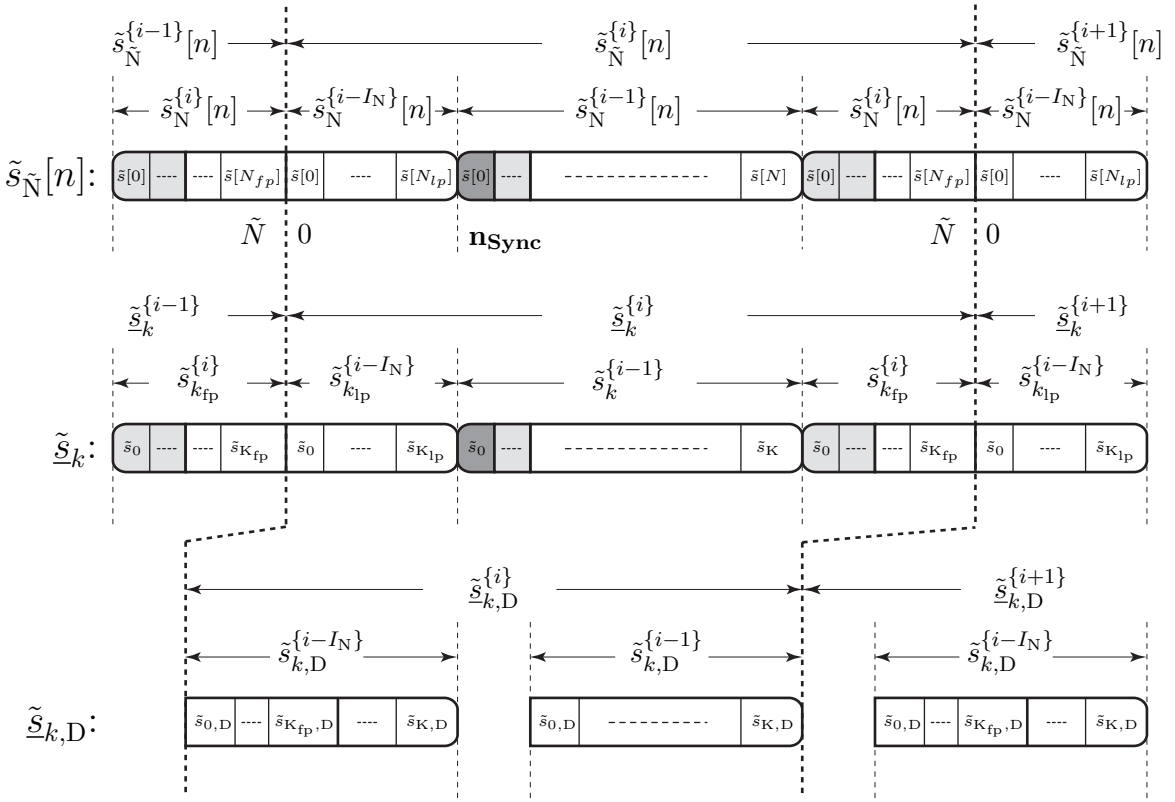


Figure 4.15: Data symbol detection.

will be processed at the *sampling point detector*. This detector detects the optimum sampling index n_0 on an interval of one symbol period T_{sym} to minimize τ_0 in Equ. (2.16)[6]. It also suppresses the N_{f} samples of $\underline{q}^{\{i\}}_{\mathrm{N+N_f}}[n]$ to avoid new invalid symbols (see Section 4.3.2). The output of the sampling point detector

$$\underline{q}^{\{i\}}_{\mathrm{N}}[n-n_0] = (q^{\{i-I_{\mathrm{N}}\}}_{\mathrm{N_{lp}}}[n-n_0],\ q^{\{i-I_{\mathrm{N}}+1\}}_{\mathrm{N}}[n-n_0],\ \cdots,\ q^{\{i-1\}}_{\mathrm{N}}[n-n_0],\ q^{\{i\}}_{\mathrm{N_{fp}}}[n-n_0]) \tag{4.21}$$

will be downsampled at the *OSF downsampler*. This yields the sequence

$$\begin{aligned} \underline{q}^{\{i\}}_{k} &= \underline{q}^{\{i\}}_{\mathrm{N}}[\mathrm{OSF}\cdot(k-1)] && (4.22)\\ &= \left(q^{\{i-I_{\mathrm{N}}\}}_{\mathrm{N_{lp}}}[\mathrm{OSF}\cdot(k-1)],\ \cdots,\ q^{\{i-1\}}_{\mathrm{N}}[\mathrm{OSF}\cdot(k-1)],\ q^{\{i\}}_{\mathrm{N_{fp}}}[\mathrm{OSF}\cdot(k-1)]\right) \\ &= (q^{\{i-I_{\mathrm{N}}\}}_{k_{lp}},\ q^{\{i-I_{\mathrm{N}}+1\}}_{k},\ \cdots,\ q^{\{i-1\}}_{k},\ q^{\{i\}}_{k_{fp}}), && (4.23) \end{aligned}$$

[6]The timeoffset due to the channel and transmit filter has been already eliminated by the packet detection.

where

$$1 \leq k \leq K$$
$$1 \leq k_{lp} \leq K_{lp}$$
$$1 \leq k_{fp} \leq K_{fp}$$

(for the determination of the OSF see Section 4.2.1). The sampling point detector and the OSF-downconverter realize the symbol-rate sampler in Figure 2.3. Next, the *decision device* decides for each sample $q_k^{\{i-u\}}$ of $\underline{q}_k^{\{i\}}$ a "detected" symbol $\tilde{s}_k^{\{i-u\}} \in \mathcal{A}$. The decision was correct, if $\tilde{s}_k^{\{i-u\}}$ equals the transmitted $s_k^{\{i-u\}}$, otherwise a symbol error has occurred. In `SoftRadio` for the decision of $\tilde{s}_k^{\{i-u\}} = s^{(m)}$, the symbol $s^{(m)}$ with the *minimum* distance $|q_k^{\{i-u\}} - s^{(m)}|$ to the sample $q_k^{\{i-u\}}$ will be used[7]. The output of the decision device

$$\underline{\tilde{s}}_k^{\{i\}} = (\tilde{s}_{k_{lp}}^{\{i-I_\mathrm{N}\}},\ \tilde{s}_k^{\{i-I_\mathrm{N}+1\}},\ \cdots,\ \tilde{s}_k^{\{i-1\}},\ \tilde{s}_{k_{fp}}^{\{i\}}) \tag{4.24}$$

will be the detected symbol packets. The *channel decoder* merges the first part $\tilde{s}_{k_{fp}}^{\{i-I_\mathrm{N}\}}$ (stored from the previous signal block processing) with the last part $\tilde{s}_{k_{lp}}^{\{i-I_\mathrm{N}\}}$ and stores $\tilde{s}_{k_{fp}}^{\{i\}}$ for the next signal block processing. Furthermore the packet header $\tilde{s}_{k,\mathrm{P}}^{\{i-u\}}$ will be processed and removed. For more information about the channel decoder see Section 4.3.3. The output of the channel decoder results to the final detected data symbol sequences

$$\underline{\tilde{s}}_{k,\mathrm{D}}^{\{i\}} = (\tilde{s}_{k,\mathrm{D}}^{\{i-I_\mathrm{N}\}},\ \cdots,\ \tilde{s}_{k,\mathrm{D}}^{\{i-2\}},\ \tilde{s}_{k,\mathrm{D}}^{\{i-1\}}). \tag{4.25}$$

Figure 4.15 sketches the detection of the data symbol sequences $\underline{\tilde{s}}_{k,\mathrm{D}}^{\{i\}}$ from the received sample stream $\tilde{s}_{\tilde{\mathrm{N}}}^{\{i\}}[n]$.

The `SoftRadio` receiver does not implement an *Overlap & Add* stage after the receive filter since the detected packets $\tilde{s}_\mathrm{N}^{\{i-u\}}[n]$ of $\underline{\tilde{s}}_\mathrm{N}^{\{i\}}[n]$ are processed independently. Therefore the N_f samples of the filter output $q_{\mathrm{N}+\mathrm{N_f}}^{\{i-u\}}[n]$ do not influence the processing of the next packet $q_{\mathrm{N}+\mathrm{N_f}}^{\{i-u+1\}}[n]$.

The `SoftRadio` receiver does not implement a channel equalization. This could be added in the future.

[7]This detector is called *Maximum Likelihood* detector [1]

4.3.1 The Packet Detection

The packet detection detects the position n_{Sync} of the first sample of the first packet header $\tilde{s}_{\tilde{\mathrm{N}}}^{\{i-I_{\mathrm{N}}+1\}}[0]$ in the received samples $\tilde{s}_{\tilde{\mathrm{N}}}^{\{i\}}[n]$ via correlation with the synchronization sequence $s_{k,\mathrm{Sync}}^{\{i\}}$ introduced by the transmitter (see Figure 4.15 and Figure 4.8). Therefore the known *complex* synchronization sequence $s_{k,\mathrm{Sync}}^{\{i\}}$

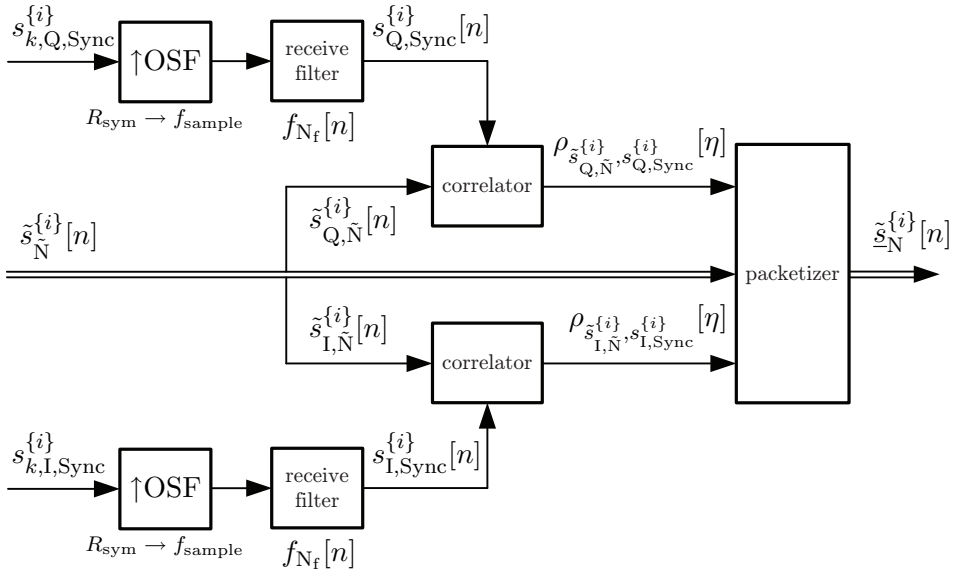


Figure 4.16: Blockdiagram of the packet detector realized by `SoftRadio`.

has to be upsampled by the OSF and filtered by the receive filter $f_{\mathrm{N_f}}[n]$ to get the transmitted[8] PAM synchronization sample sequence for the I- and Q-parts

$$s_{\mathrm{I,Sync}}^{\{i\}}[n] = \sum_{k=1}^{K_{\mathrm{Sync}}} s_{k,\mathrm{I,Sync}}^{\{i\}} \; f_{\mathrm{N_f}}\left[n - (k-1)\cdot \mathrm{OSF}\right] \qquad 0 \le n \le K_{\mathrm{Sync}} \cdot \mathrm{OSF} + N_{\mathrm{f}} - 1 \tag{4.26}$$

$$s_{\mathrm{Q,Sync}}^{\{i\}}[n] = \sum_{k=1}^{K_{\mathrm{Sync}}} s_{k,\mathrm{Q,Sync}}^{\{i\}} \; f_{\mathrm{N_f}}\left[n - (k-1)\cdot \mathrm{OSF}\right] \qquad 0 \le n \le K_{\mathrm{Sync}} \cdot \mathrm{OSF} + N_{\mathrm{f}} - 1 \tag{4.27}$$

(see Figure 4.16). Next, for the I-path, the I-components $\tilde{s}_{\mathrm{I,\tilde{N}}}^{\{i\}}[n]$ and $s_{\mathrm{I,Sync}}^{\{i\}}[n]$ will be correlated. The correlation function

$$\rho_{\tilde{s}_{\mathrm{I,\tilde{N}}}^{\{i\}}, s_{\mathrm{I,Sync}}^{\{i\}}}[\eta] = \sum_{n=-\infty}^{\infty} \tilde{s}_{\mathrm{I,\tilde{N}}}^{\{i\}}[n] \cdot s_{\mathrm{I,Sync}}^{\{i\}}[n+\eta] \tag{4.28}$$

[8] if $f_{\mathrm{N_f}}[n] \hat{=} g_{\mathrm{N_g}}[n]$

measures the similarity of $\tilde{s}^{\{i\}}_{\mathrm{I},\tilde{\mathrm{N}}}[n]$ and $s^{\{i\}}_{\mathrm{I,Sync}}[n]$ as a function of the time shift η. For $\eta = n_{\mathrm{Sync}}$ a peak of maximum strength occurs due to

$$\tilde{s}^{\{i\}}_{\tilde{\mathrm{N}}}[n_{\mathrm{Sync}}] = \tilde{s}^{\{i-I_{\mathrm{N}}+1\}}_{\mathrm{N}}[0] = \tilde{s}^{\{i-I_{\mathrm{N}}+1\}}_{\mathrm{Sync}}[0]$$

(for n_{Sync} see Figure 4.15). For a precise and reliable detection of this peak, the following requirements have to be satisfied:

1. only one peak of maximum strength should appear per packet $\tilde{s}^{\{i-u\}}_{\mathrm{N}}[n]$,
2. all other peaks of the correlation function should be under a certain threshold ρ_{tr},
3. the synchronization sequence should be equal for all symbol packets $s^{\{i\}}_{\mathrm{N}}[n]$.

ad 1.) If more maximum peaks appear, then the data symbol sequence $s^{\{i\}}_{k,\mathrm{D}}$ (generated by the data symbol source at the `SoftRadio` transmitter) contains the synchronization sequence $s^{\{i\}}_{k,\mathrm{Sync}}$. The number of synchronization symbols K_{Sync} determines the probability for the appearence of the synchronization sequence in the rest of the symbol packet. For a appearance probability under a certain threshold, K_{Sync} has to be larger than a certain value. K_{Sync} depends only on the size of the symbol alphabet M_s and the number of symbols K in a packet. If the alphabet size M_s grows, the number of synchronization symbols K_{sync} can be reduced if the number of symbols K in a packet remains unchanged, and vice versa.

ad 2.) This can be achieved by a *pseudo-orthogonal* synchronization sequence $s^{\{i\}}_{k,\mathrm{I,Sync}}$ of sufficient length. The autocorrelation sequence

$$\rho_{\kappa,\tilde{s}^{\{i\}}_{\mathrm{I},\tilde{\mathrm{N}}}} = \sum_{k=-\infty}^{\infty} \tilde{s}^{\{i\}}_{k,\mathrm{I,Sync}} \cdot \tilde{s}^{\{i\}}_{k+\kappa,\mathrm{I,Sync}} \tag{4.29}$$

of a pseudo-orthogonal symbol sequence (scaled by the maximum value) results to

$$\frac{1}{\rho_{0,s^{\{i\}}_{\mathrm{I,Sync}}}} \rho_{\kappa,s^{\{i\}}_{\mathrm{I,Sync}}} = \begin{cases} 1 & \kappa = 0 \\ < \rho_{\mathrm{tr}} & \kappa \neq 0 \end{cases}$$

(see Figure 4.17). For the reason of simplicity, measuring tasks and scaling tasks, in `SoftRadio` only two symbols

$$s_{k,\mathrm{Sync}} \in \left\{ s^{(1)}, s^{(2)} \right\} \quad \text{where} \quad s^{(2)} = -s^{(1)}$$

from the symbol alphabet $\mathcal{A}$ will be used for the synchronization sequence.

ad 3.)

$$s^{\{i\}}_{k,\mathrm{Sync}} = s^{\{i-1\}}_{k,\mathrm{Sync}} \qquad \forall i$$

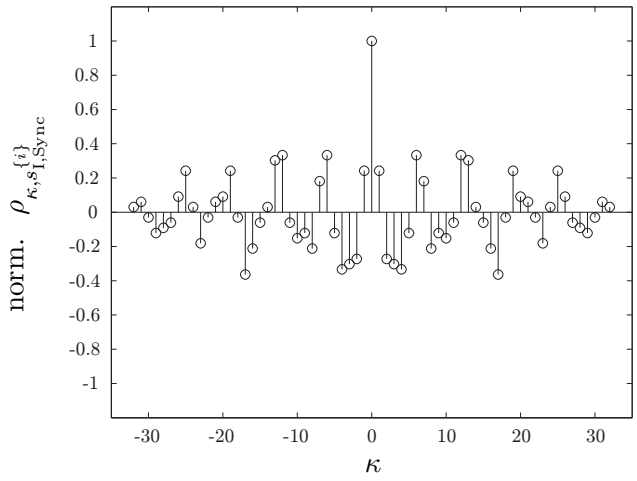


Figure 4.17: normalized correlation sequence of the pseudo-orthogonal synchronization sequence.

simplifies the detection of $\tilde{s}_{k,\mathrm{Sync}}^{\{i\}}$.

The correlation streams of the I- and Q-components will be analyzed by the *packetizer*. The packetizer decides the starting index n_{Sync} and creates the detected packets

$$\underline{\tilde{s}}_{\mathrm{N}}^{\{i\}}[n] = (\tilde{s}_{\mathrm{N_{lp}}}^{\{i-I_{\mathrm{N}}\}}[n], \tilde{s}_{\mathrm{N}}^{\{i-I_{\mathrm{N}}+1\}}[n], \cdots, \tilde{s}_{\mathrm{N}}^{\{i-1\}}[n], \tilde{s}_{\mathrm{N_{fp}}}^{\{i\}}[n])$$

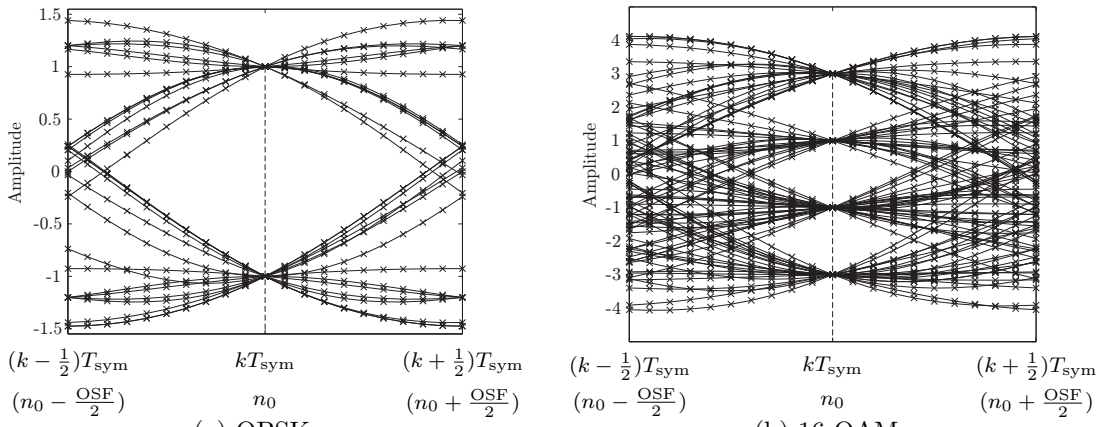
from the received sample stream $\tilde{s}_{\tilde{N}}^{\{i\}}[n]$.

4.3.2 The Sampling Point Detector

In Section 2.3, Equ. (2.16) shows that $q(t)$ has to be sampled at the optimum sampling point kT_{sym} to obtain Equ. (2.18) and $q[k] = s_k\ p[0]$ in the case of an ISI-free transmission.

The sampling point detector detects the optimum sampling point n_0 of $\underline{q}_{\mathrm{N}}^{\{i\}}[n]$ on an interval of one symbol period T_{sym}. The optimum sampling point for PAM will be the time index, where the eye diagram of $q_{\mathrm{N}}^{\{i-u\}}[n]$ will be vertically widest open [1]. Figure 4.18 shows the eye diagram of the I- or Q-component of a QPSK and a 16-QAM raised cosine signal and the optimum sampling point n_0 ($\times\ldots$ indicates the samples of $q_{\mathrm{N}}^{\{i-u\}}[n]$).

The second task is to suppress the N_{f} samples of $\underline{q}_{\mathrm{N+N_f}}^{\{i\}}[n]$. The filtering of $\tilde{s}_{\mathrm{N}}^{\{i-u\}}[n]$ with $f_{\mathrm{N_f}}[n]$ of length N_{f} results to $q_{\mathrm{N+N_f}}^{\{i-u\}}[n]$ of length $(N + N_{\mathrm{f}})$. Due to the adjacent downsampling by the OSF, the N_{f} samples would introduce new samples into $q_k^{\{i-u\}}$ and therefore new symbols to $\tilde{s}_k^{\{i-u\}}$, if $N_{\mathrm{f}} \geq \mathrm{OSF}$.



(a) QPSK (b) 16-QAM

Figure 4.18: Eye diagrams of raised cosine PAM signals

4.3.3 The Channel Decoding

The major task of the channel decoding at the `SoftRadio` receiver is to process the detected status symbols $\tilde{s}_{k,\text{P}}^{\{i-u\}}$ added by the channel coding at the `SoftRadio` transmitter. This header is depicted again in Figure 4.19.

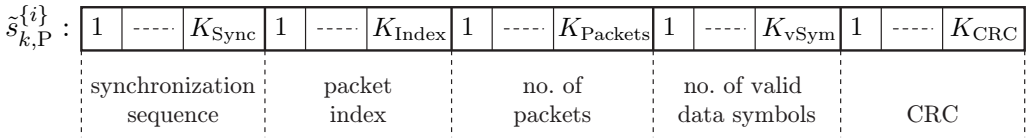


Figure 4.19: Format of the detected packet header at `SoftRadio`.

For the $(i{-}u)$-th detected symbol sequence $\tilde{s}_k^{\{i-u\}}$ the following processing will be done:

- The *synchronization sequence* will be dropped, since it has been already processed at the packet detection.
- For the detection of a symbol error in $\tilde{s}_k^{\{i-u\}}$, the channel decoding determines first the detected $\widetilde{\text{CRC}}$

$$\widetilde{\text{CRC}} = \tilde{s}_{1,\text{CRC}}^{\{i-u\}}\, M_{\text{s}}^0 + \tilde{s}_{2,\text{CRC}}^{\{i-u\}}\, M_{\text{s}}^1 + \cdots + \tilde{s}_{K_{\text{CRC}},\text{CRC}}^{\{i-u\}}\, M_{\text{s}}^{(K_{\text{CRC}}-1)}. \qquad (4.30)$$

  and recalculates the CRC of the detected packet symbols[9]. If

$$\widetilde{\text{CRC}} \;\neq\; \text{CRC}_{\text{recalc}},$$

  an error has been detected and the packet $\tilde{s}_k^{\{i-u\}}$ will be discarded. This error can be caused either by an symbol error or by incorrect packet detection, because the latter results in erroneously use of symbols for the detected $\widetilde{\text{CRC}}$.

[9] this excludes both the synchronization sequence *and* the CRC symbols $\tilde{s}_{k,\text{CRC}}^{\{i-u\}}$

- If no symbol error has been observed, the channel decoding determines the *packet index*

$$\tilde{\hat{i}}^{\{i-u\}} = \tilde{s}_{1,\mathrm{Index}}^{\{i-u\}}\ M_{\mathrm{s}}^{0} + \tilde{s}_{2,\mathrm{Index}}^{\{i-u\}}\ M_{\mathrm{s}}^{1} + \cdots + \tilde{s}_{K_{\mathrm{Index}},\mathrm{Index}}^{\{i-u\}}\ M_{\mathrm{s}}^{(K_{\mathrm{Index}}-1)} \quad (4.31)$$

of $\tilde{s}_k^{\{i-u\}}$ and compares it to $\tilde{\hat{i}}^{\{i-u-1\}}$ of the *previous* detected symbol packet $\tilde{s}_k^{\{i-u-1\}}$. If[10]

$$\tilde{\hat{i}}^{\{i-u\}} \neq (\tilde{\hat{i}}^{\{i-u-1\}} + 1) \bmod M_{\mathrm{s}}^{K_{\mathrm{Index}}},$$

a packet loss has occurred. This will be displayed by `SoftRadio`.

- If no packet loss has been detected, the channel decoding determines the *number of packets*

$$\tilde{I}_{\mathrm{Packets}} = \tilde{s}_{1,\mathrm{Packets}}^{\{i-u\}}\ M_{\mathrm{s}}^{0} + \tilde{s}_{2,\mathrm{Packets}}^{\{i-u\}}\ M_{\mathrm{s}}^{1} + \cdots + \tilde{s}_{K_{\mathrm{Packets}},\mathrm{Packets}}^{\{i-u\}}\ M_{\mathrm{s}}^{(K_{\mathrm{Packets}}-1)}. \quad (4.32)$$

If

$$\tilde{I}_{\mathrm{Packets}} = 0,$$

an *infinite* data symbol transmission is assumed and therefore every symbol packet $\tilde{s}_k^{\{i-u\}}$ will be used for generating the data symbol stream

$$\tilde{s}_{k,\mathrm{D}} = (\tilde{s}_{k,\mathrm{D}}^{\{0\}},\ \tilde{s}_{k,\mathrm{D}}^{\{1\}},\ \cdots).$$

$$\tilde{I}_{\mathrm{Packets}} > 0$$

indicates a *finite* data symbol transmission. Therefore only $\tilde{I}_{\mathrm{Packets}}$ will be used for generating the detected data symbol stream

$$\tilde{s}_{k,\mathrm{D}} = (\tilde{s}_{k,\mathrm{D}}^{\{0\}},\ \tilde{s}_{k,\mathrm{D}}^{\{1\}},\ \cdots,\ \tilde{s}_{k,\mathrm{D}}^{\{I_{\mathrm{Packets}}\}}).$$

- If $\tilde{s}_k^{\{i-u\}}$ will be used for the generation of the data symbol stream $\tilde{s}_{k,\mathrm{D}}$, then the channel decoding determines the *number of valid data symbols*

$$\tilde{K}_{\mathrm{vData}} = \tilde{s}_{1,\mathrm{vSym}}^{\{i-u\}}\ M_{\mathrm{s}}^{0} + \tilde{s}_{2,\mathrm{vSym}}^{\{i-u\}}\ M_{\mathrm{s}}^{1} + \cdots + \tilde{s}_{K_{\mathrm{vSym}},\mathrm{vSym}}^{\{i-u\}}\ M_{\mathrm{s}}^{\left(K_{\mathrm{vSym}}-1\right)} \quad (4.33)$$

of this symbol packet. If

$$\tilde{K}_{\mathrm{vData}} = 0,$$

then every data symbol of $\tilde{s}_k^{\{i-u\}}$ will be used for $\tilde{s}_{k,\mathrm{D}}^{\{i-u\}}$ and

$$K_{\mathrm{D}} = K - K_{\mathrm{P}}.$$

Otherwise only the first $\tilde{K}_{\mathrm{vData}}$ data symbols will be used for $\tilde{s}_{k,\mathrm{D}}^{\{i-u\}}$

$$K_{\mathrm{D}} = \tilde{K}_{\mathrm{vData}}.$$

[10] $(x \mod y)$... x modulo y

The *packet index, number of packets* and *number of valid data symbols* makes it possible to receive a file of arbitrary length as data symbol stream $\tilde{s}_{k,\mathrm{D}}$.

Before processing the packet header of the individual symbol sequences $\tilde{s}_k^{\{i-u\}}$, the channel coder merges the first part $\tilde{s}_{k_{fp}}^{\{i-I_\mathrm{N}\}}$ (stored from the previous signal block processing) with the last part $\tilde{s}_{k_{lp}}^{\{i-I_\mathrm{N}\}}$ and stores $\tilde{s}_{k_{fp}}^{\{i\}}$ for the next signal block processing.

The second task of the channel decoding is *deinterleaving* [12] of the data symbol stream $\tilde{s}_{k,\mathrm{D}}$ (see also Section 4.2.2), which is subject of future work.

4.3.4 The Graphical User Interface

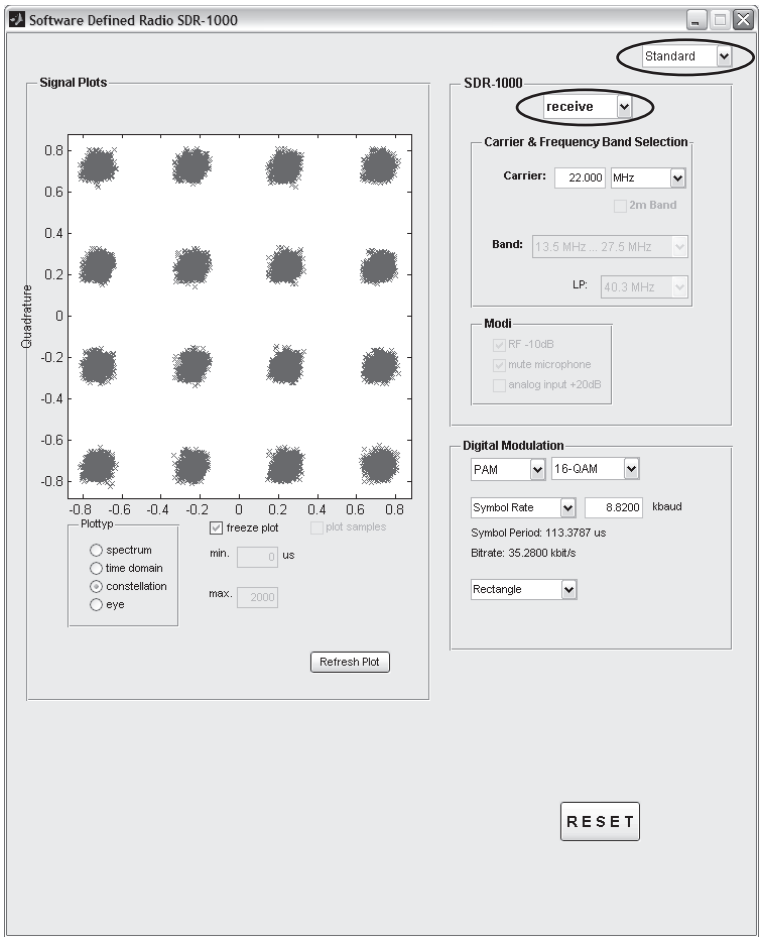


Figure 4.20: Standard GUI of the `SoftRadio` in **receive** mode.

Figure 4.20 shows the standard[11] Graphical User Interface of `SoftRadio` as it

[11] for information about the advanced mode see Appendix, Section B

looks like in *receiver* mode. In this section, only the differences to the transmit mode explained in Section 4.2.3 will be mentioned.

4.3.4.1 Digital Modulation

Field 3 in Figure 4.10 determines the receive filter $g_{\mathrm{N_g}}[n]$ instead of the pulse shaping filter $f_{\mathrm{N_f}}[n]$

4.3.4.2 Signal Plots

The following plot types are implemented:

- symbol constellation plot of the decision device input $\underline{q}_k^{\{i\}}$ (visible in Figure 4.20)
- time domain plot of the filtered I- and Q-components $\underline{q}_{\mathrm{I,N}}^{\{i\}}[n-n_0]$, $\underline{q}_{\mathrm{Q,N}}^{\{i\}}[n-n_0]$ (without the packet header samples)
- eye diagram of the filtered I- and Q-components $\underline{q}_{\mathrm{I,N}}^{\{i\}}[n-n_0]$, $\underline{q}_{\mathrm{Q,N}}^{\{i\}}[n-n_0]$ (without the packet header samples)
- PSD of the filtered $\underline{q}_{\mathrm{N}}^{\{i\}}[n-n_0]$ (without the packet header samples)

4.4 SoftRadio and the SDR-1000

SoftRadio controls the following hardware functions of the transceiver SDR-1000 (see Figure 3.2 in Section 3.2):

- the creation of the carrier at the given frequency f_{IF} by the DDS
- the switching between transmit (TX) and receive (RX) mode
- the attenuation of the transmit signal $s_{\mathrm{IF}}(t)$ or receive signal $\tilde{s}_{\mathrm{IF}}(t)$ by 10 dB
- the amplification of the received BB-signal $\tilde{s}_{\mathrm{BB}}(t)$ by 20 dB
- the selection of a BPF (individually or corresponding to the carrier frequency) from the filterbank
- the selection of a LPF (individually or corresponding to the BPF) from the filterbank
- the muting or unmuting of the microphone input

Chapter 5

Summary

This diploma thesis describes first the concept of software defined radio, its advantages and disadvantages, and gives a theoretical overview of base band pulse amplitude modulation transmission. Further the thesis deals with the hardware used for the software defined radio transmitter and receiver. In particular, it treats the transceiver SDR-1000, the embedded quadrature sampling decoder/exciter as a mixing stage and the direct digital synthesizer as a precise digital oscillator.

The major part of the thesis discusses the software realization of the transmitter and of the receiver. Especially, it deals with the problem of real time processing in software. It describes the software parts of the PAM transmitter, in particular the structure of the symbol source generator, the channel coding and the overlap & add method. It also shows the format of the transmitted symbol stream and the graphical user interface. At the receiver side, the thesis deals with the software structure, the packet detection and the sampling point detector for the synchronization, the decision device and the channel decoder.

Appendix A

Further Matlab Implementations

A.1 The *SDR*-Class

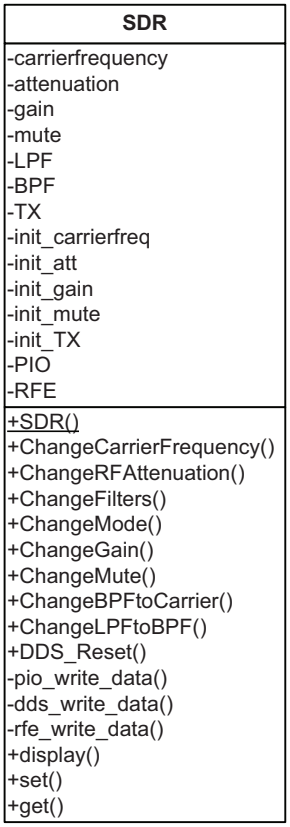


Figure A.1: The *SDR* class.

The following features of the transceiver SDR-1000 are implemented by this class depicted in Figure A.1 (see Figure 3.2 in Section 3.2):

- the creation of the carrier at the given frequency f_{IF} of the DDS

- the switching between transmit (TX) and receive (RX) mode
- the attenuation of the transmit signal $s_{\mathrm{IF}}(t)$ or receive signal $\tilde{s}_{\mathrm{IF}}(t)$ by 10 dB
- the amplification of the received BB-signal $\tilde{s}_{\mathrm{BB}}(t)$ by 20 dB
- the selection of a BPF (individual or corresponding to the carrier frequency) from the filterbank
- the selection of a LPF (individual or corresponding to the BPF) from the filterbank
- the muting or unmuting of the microphone input

Attributes

carrierfrequency: current carrier frequency f_{IF} in [Hz]

attenuation: current state of the 10 dB attenuator (on or off)

gain: current state of the 20 dB RX-BB amplifier (on or off)

mute: current state of the microphone muting (on or off)

LPF: current active LPF index

BPF: current active BPF index

TX: current state of the SDR-1000 operation mode (transmit or receive)

init_xxx: initial value of attribute xxx (for reset)

PIO: a structure, that stores all the information of the Parallel Input and Output (PIO) board

RFE: a structure, that stores all the information of the Radio Frequency Expansion (RFE) board

Methods

SDR(): *Constructor* of the class; sets the SDR-1000 up to the initial working state

ChangeCarrierFrequency(): changes the oscillator of the SDR-1000 to the current attribute value f_{IF}

ChangeRFAttenuation(): switches the 10 dB attenuator to the current attribute state (on or off)

ChangeFilters(): changes the SDR-1000 BPF and LPF to the current attribute indices

ChangeMode(): switches the SDR-1000 operation mode to the current attribute state (transmit or receive)

ChangeGain(): switches the 20 dB RX-BB amplifier to the current attribute state (on or off)

ChangeMute(): switches the microphone muting to the current attribute state (on or off)

ChangeBPFtoCarrier(): changes the SDR-1000 BPF and LPF corresponding to the current carrier frequency f_{IF}

ChangeLPFtoBPF(): changes the SDR-1000 LPF corresponding to the current BPF

DDS_Reset(): resets the DDS

pio_write_data(): writes a data-byte to a specified PIO board D-Latch Integrated Circuit (IC)

dds_write_data(): writes a data-byte into an specified address-register of the DDS

rfe_write_data(): writes a data-byte into a specified RFE board IC

display(): displays the current status information of the SDR-object

set(): to change the value of an attribute

get(): to get the value of an attribute

A.2 The *SignalOut*-Class

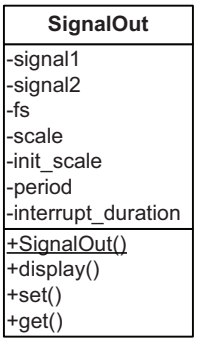
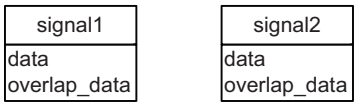


Figure A.2: The *SignalOut* class.



(a) (b)

Figure A.3: The *signal* structures of *SignalOut*.

This class (shown in Figure A.2) implements the software representation of the complex BB transmit signal

$$s_{\mathrm{BB}}[n] = s_{\mathrm{I}}[n] + j s_{\mathrm{Q}}[n].$$

Attributes

signal1: a structure (shown in Figure A.3a), that corresponds to the *Inphase* component $s_{\mathrm{I}}[n]$.

data: the *valid* I signal samples

overlap_add: the *invalid* I signal samples (results from filtering; for correction)

signal2: a structure (shown in Figure A.3b), that corresponds to the *Quadrature* component $s_{\mathrm{Q}}[n]$.

data: the *valid* Quadrature signal samples

overlap_add: the *invalid* Quadrature signal samples (results from filtering; for correction)

fs: the sampling frequency f_{sample} of $s_{\mathrm{BB}}(t)$ (fixed to 44.1 kHz)

scale: an amplitude scaling factor of signal1 and signal2 ($0 \leq \mathrm{scale} \leq 1$). It represents the rejection of the sound card by both signals (0... no rejection, 1... full rejection)

init_scale: initial scaling value (for reset)

period: interrupt period in number of samples

interrupt_duration: duration of MATLABs *Data Acquisition Toolbox (DAT)* analog output engines interrupt processing in [seconds]

Methods

SignalOut(): *Constructor* of the class; sets the analog output engine of MATLABs DAT to the initial working state

display(): displays the current status information of the SignalOut-object

set(): to change the value of an attribute

get(): to get the value of an attribute

A.3 The *SignalIn*-Class

This class (shown in Figure A.4) implements the software representation of the received complex BB-signal

$$\tilde{s}_{\mathrm{BB}}[n] = \tilde{s}_{\mathrm{I}}[n] + j \tilde{s}_{\mathrm{Q}}[n].$$

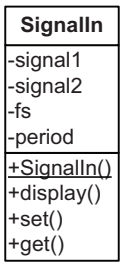
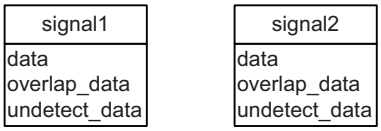


Figure A.4: The *SignalIn* class.



(a) (b)

Figure A.5: The *signal* structures of *SignalIn*.

Attributes

signal1: a structure (shown in Figure A.5a), that corresponds to the *Inphase* component $\tilde{s}_{\mathrm{I}}[n]$.

data: the *valid* Inphase signal samples

overlap_add: the *invalid* Inphase signal samples (results from filtering; for correction)

undetect_data: the unprocessed Inphase signal samples from the previous interrupt

signal2: a structure (shown in Figure A.5b), that corresponds to the *Quadrature* component $\tilde{s}_{\mathrm{Q}}[n]$.

data: the *valid* Quadrature signal samples

overlap_add: the *invalid* Quadrature signal samples (results from filtering; for correction)

undetect_data: the unprocessed Quadrature signal samples from the previous interrupt

fs: the sampling frequency f_{sample} of $\tilde{s}_{\mathrm{BB}}(t)$ (fixed to 44.1 kHz)

period: interrupt period in numbers of samples

Methods

SignalIn(): *Constructor* of the class; sets the analog input engine of MATLABs DAT to the initial working state

display(): displays the current status information of the SignalIn-object

set(): to change the value of an attribute

get(): to get the value of an attribute

A.4 The *PAM*-Class

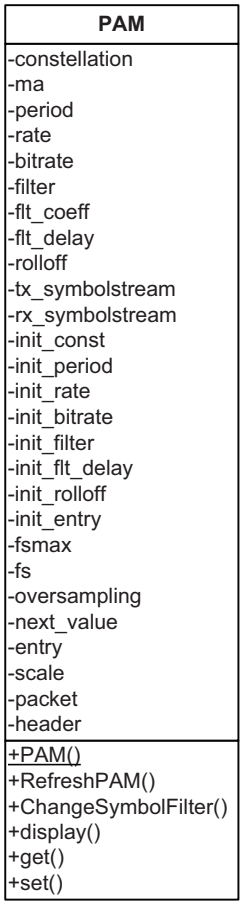


Figure A.6: The *PAM* class.

The following features of a PAM transmission are implemented by this class depicted in Figure A.6:

- the selection of various symbol constellations (BPSK, QPSK, nQAM)
- the specification of the timing property either by symbol period T_{sym}, symbol rate R_{sym} or by bit rate R_{bit}
- the selection and specification of the transmit/receive filter (rectangle or RRC of different length and rolloff factor)

Attributes

constellation: current active PAM symbol constellation (BPSK, QPSK, nQAM, ...)

ma: numbers of symbols in the symbol alphabet $\mathcal{A}$

period: current symbol period T_{sym} in [seconds]

rate: current symbol rate R_{sym} in [baud]

bitrate: current bit rate R_{bit} in [bit/s]

filter: current active symbol filter (rectangle or RRC)

flt_coeff: filter impulse response $g[n]$ of the symbol filter

flt_delay: current delay of the filter impulse response in numbers of symbols

$$\begin{cases} = 0 & \text{rectangle} \\ > 0 & \text{RRC} \end{cases}$$

rolloff: current rolloff factor α of the RRC filter

tx_symbolstream: transmitted symbol stream a_k (at Transmit mode of SDR-1000)

rx_symbolstream: detected symbol stream $\tilde{a}_k$ (at Receive mode of SDR-1000)

init_xxx: initial value of attribute xxx (for reset)

fsmax: maximum sampling frequency $f_{\text{sample,max}}$ of the SignalIn- or SignalOut-object (fixed to 44.1 kHz)

fs: current sampling frequency f_{sample} of the SignalIn- or SignalOut-object

oversampling: oversampling factor ($= \lfloor f_{\text{sample}}/R_{\text{sym}} \rfloor$)

next_value: next valid value of T_{sym} or R_{sym} or R_{bit}

entry: current active entry mode (symbol period, symbol rate or bitrate)

scale: an amplitude normalization factor for the PAM transmit signal stream

packet: a structure, that stores all the information of the whole packet

header: a structure, that stores all the information of the packet header

Methods

PAM(): *Constructor* of the class; sets the PAM transmission to the initial working state

RefreshPAM(): refreshes all the depending attributes and variables by a modification of a PAM attribute

ChangeSymbolFilter(): calculates the new filter impulse response $g[n]$ corresponding to the current symbol filter attribute

display(): displays the current status information of the PAM-object

set(): to change the value of an attribute

get(): to get the value of an attribute

A.5 The Subfunctions of `SoftRadio.m`

SignalOutCallbackFcn(): the callback (interrupt) function of the analog output engine of MATLABs DAT. This function implements the complete SDR transmitter.

SignalInCallbackFcn(): the callback function of the analog input engine of MATLABs DAT. This function implements the complete SDR receiver.

RefreshSignalPlots(): the callback function of the signal plot timer; calculates the different signal plots (time, PSD, eye, scatter)

sdr_eyediagram(): plots an eye diagram; modified version of MATLABs eyediagram.m function for the GUI

sdr_scatterplot(): plots a scatter plot; modified version of the MATLAB scatterplot.m function for the GUI

GetCarrierFrequency(): calculates the current f_{IF} from the GUI

GetSymbolPeriodRate(): calculates the current T_{sym}, R_{sym} or R_{bit} from the GUI

ChangeCarrierNum(): calculates the GUI presentation for the current f_{IF}

ChangeSymbolTimeNum(): calculates the GUI presentation for the current T_{sym}, R_{sym} or R_{bit}

ResetWindow(): resets the GUI

ChangeAppMode(): changes the GUI according to the current SoftRadio application mode

Appendix B

The Advanced GUI of `SoftRadio`

The GUI of `SoftRadio` distinguishes between the *Standard-* and the *Advanced-*mode. The advanced mode enables enhanced features of `SoftRadio` and displays detailed status information for use during software development. These will be equal in transmit and receive mode. Therefore only the extensions to the standard transmit mode are discussed here. The standard GUI of the transmit and receive mode has already been described in Section 4.2.3 and Section 4.3.4 respectively.

Figure B.1 shows the advanced Graphical User Interface of `SoftRadio` as *transmitter*. The GUI mode can be changed with the pulldown list at the top right corner. The following status information will be displayed:

ai-samples: number of samples stored in the receive buffer of the `analoginput` engine [13]

ao-samples: number of samples stored in the transmit buffer of the `analogoutput` engine [13]

packet index: the transmitted/detected *Packet Index* of the current symbol packet

CRC: the transmitted/detected *CRC* of the current symbol packet

no. of packets: the transmitted/detected *Number of Packets* of the current symbol packet

no. of datasym: the number of transmitted/detected datasymbols $K_\mathrm{D}/\tilde{K}_\mathrm{D}$ of the current symbol packet

B.1 Digital Modulation

The enhanced *Digital Modulation* block lets the user specify the delay ($\hat{=}$ length in number of symbols, see Figure B.2) of the RRC filter with **textfield 7**.

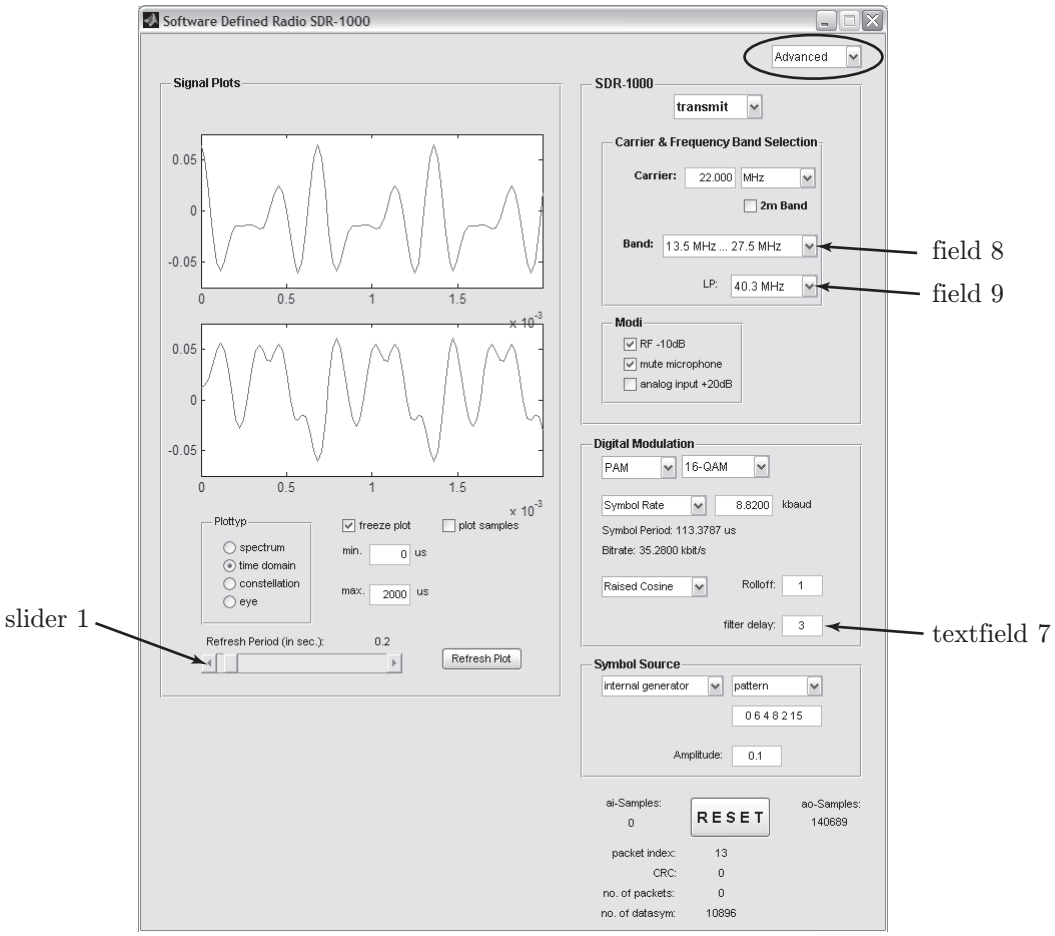


Figure B.1: The **advanced** Graphical User Interface of the `SoftRadio` in *transmit* mode.

B.2 SDR-1000

The enhanced *SDR-1000* block controls the following features of the transceiver SDR-1000.

Carrier & Frequency Band Selection: The BPF and LPF can be modified individually by changing **field 8** and **field 9**. In standard mode, changing the carrier frequency adjusts the BPF and LPF automatically.

Modi: This function block controls the following SDR-1000 functions by selecting/deselecting the associated checkbox:

- 10 dB attenuation of the transmitted or received IF-signal
- 20 dB amplification of the received BB-signal $\tilde{s}_{\mathrm{BB}}(t)$
- muting the microphone.

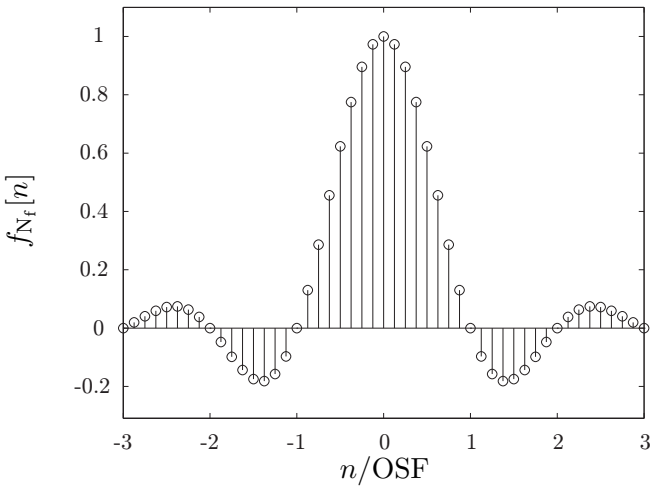


Figure B.2: Raised Cosine with a filter delay of 3 symbols.

B.3 Signal Plots

The enhanced *Signal Plots* block lets the user specify the refresh period of the plot view with **slider 1**.

Bibliography

[1] Franz Hlawatsch. *Analog and Digital Communication Techniques, Skriptum zur Vorlesung Übertragungsverfahren 1+2*. Institute of Communications and Radio Frequency Engineering, Vienna university of technology, Austria, October 1999.

[2] Jeffrey H. Reed. *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall PTR, 1st edition, May 2002.

[3] Hans Heinrich Meinke and Klaus Lange. *Taschenbuch der Hochfrequenztechnik*. Springer, Berlin, 5th edition, 1992.

[4] *FlexRadio Systems*. http://flex-radio.com/.

[5] Gerald Youngblood. *A Software-Defined Radio for the Masses, Part 1*. QEX, July/August 2002. http://www.arrl.org/tis/info/pdf/020708qex013.pdf.

[6] Gerald Youngblood. *A Software-Defined Radio for the Masses, Part 4*. QEX, March/April 2003. http://www.arrl.org/tis/info/pdf/030304qex020.pdf.

[7] Rupert Patzelt and Herbert Schweinzer. *Elektrische Meßtechnik*. Springer, Vienna, 2nd edition, 1996.

[8] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-time signal processing*. LinkPrentice Hall Internat., London, 2nd international edition, 1999.

[9] Analog Devices. *AD9854, CMOS 300 MSPS Quadrature Complete DDS data sheet*. http://www.analog.com/UploadedFiles/Data_Sheets/AD9854.pdf.

[10] Analog Devices. *A Technical Tutorial on Digital Signal Synthesis*, 1999. http://www.ieee.li/pdf/essay_dds.pdf.

[11] Hans Weinrichter and Franz Hlawatsch. *Stochastische Grundlagen nachrichtentechnischer Signale*. Springer, Vienna, 1991.

[12] John G. Proakis. *Digital communications*. McGraw-Hill, Inc., New York, 3. edition, 1995.

[13] MathWorks. MATLAB, *The Language of Technical Computing*, R2006a edition, January 2006.

List of Acronyms

AC	Alternate Current
ADC	Analog to Digital Converter
BB	Base Band
BP	Band Pass
BPF	Band Pass Filter
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DAT	Data Acquisition Toolbox
DC	Direct Current
DDS	Direct Digital Synthesizer
DSP	Digital Signal Processor
FIR	Finite Impulse Response
FSK	Frequency Shift Keying
GMSK	Gaussian Minimum Shift Keying
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HF	High Frequency
I	Inphase
IC	Integrated Circuit
IF	Intermediate Frequency
IQ	Inphase and Quadrature
ISI	Inter-Symbol Interference
LO	Local Oscillator
LPF	Low Pass Filter
LTI	Linear Time Invariant

MF	Matched Filter
ML	Maximum Likelihood
MSK	Minimum Shift Keying
nQAM	Quadrature Amplitude Modulation with a symbol alphabet of n
OFDM	Orthogonal Frequency Division Multiplexing
OSF	Over-Sampling Factor
PAM	Pulse Amplitude Modulation
PC	Personal Computer
PIO	Parallel Input and Output
PROM	Programmable Read Only Memory
PSD	Power Spectral Density
Q	Quadrature
QPSK	Quadrature Phase Shift Keying
QSD	Quadrature Sampling Detector
QSE	Quadrature Sampling Exciter
RF	Radio Frequency
RFE	Radio Frequency Expansion
RMS	Root Mean Square
RRC	Root Raised Cosine
SDR	Software Defined Radio
SNR	Signal to Noise Ratio
SR	Software Radio
UMTS	Universal Mobile Telecommunication System

List of Notations

$\mathbb{N}$	set of all natural numbers (except zero)
f	frequency
ω	angular frequency $2\pi f$
T	time period $1/f$
τ_0	time offset
$\{x_1, x_2, \cdots, x_n\}$	set of the n elements
$(x_1, x_2, \cdots, x_n)$	row vector of n elements
$\mathrm{Re}\{x\}$	real part of x
$\mathrm{Im}\{x\}$	imaginary part of x
$x \mod y$	x modulo y
$\lfloor x \rfloor$	floor function (largest integer less than or equal to x)
b_i	i-th bit of a bit stream
R_{bit}	bitrate [bits/s]
R_{sym}	symbol rate [symbols/s] = [baud]
K	number of symbols
$\mathcal{A}$	symbol alphabet
M_s	number of symbols in the symbol alphabet (= size of the alphabet)
$s^{(m)}$	m-th symbol of the symbol alphabet $\mathcal{A}$
s_k	k-th symbol of a symbol stream
$s_k^{\{i\}}$	k-th symbol of the i-th symbol block (sequence, packet)
$s_{k,\mathrm{D}}^{\{i\}}$	k-th data symbol of the i-th symbol block
$s_{k,\mathrm{P}}^{\{i\}}$	k-th status symbol of the i-th symbol block
$\underline{s}_k^{\{i\}}$	concatenation of I symbol blocks at the i-th processing block
$\rho_{\kappa,s}$	autocorrelation sequence of the symbol stream s_k as function of κ

$x(t)$	continuous-time signal
$x^*(t)$	complex conjugate of $x(t)$
$(x * y)(t)$	continuous-time convolution of $x(t)$ and $y(t)$
$(x * y)[n]$	discrete-time convolution of $x[n]$ and $y[n]$
$\rho_{x,y}[\eta]$	correlation function between the discrete-time signals $x[n]$ and $y[n]$ as function of η
$\delta[n]$	unit impulse
$x[n]$	discrete-time signal $-\infty < n < \infty$
$x_{\mathrm{N}}[n]$	discrete-time signal block of length N $(0 \leq n < N)$
$x_{\mathrm{N}}^{\{i\}}[n]$	the i-th discrete-time signal block of length N of $x[n]$ $(0 \leq n < N)$
$\underline{x}_{\mathrm{N}}^{\{i\}}[n]$	concatenation of I discrete-time signal blocks of length N at the i-th processing block
$X(j\omega)$	spectral function of the continuous-time signal $x(t)$
$X(t)$	continuous-time random process
$X[n]$	discrete-time random process
$S_{\mathrm{X}}(j\omega)$	power spectral density (PSD) of the continuous-time random process $X(t)$
$S_{\mathrm{X}}(e^{j\theta})$	power spectral density (PSD) of the discrete-time random process $X[n]$

List of Figures