# System Design Document

## Audio digital signal processor

BeCreative Minor

Busse Lommers
Robin van den Dungen
Mahmud Gürler
Silas Kamphuis
Hein Verhallen
Youri Tils
Ahmed Abdelrahim
Fontys Hogescholen, De Rondom 1, 5612 AP Eindhoven
May 9, 2023

# Contents

# List of Figures

# List of Tables

# Abbreviation List

| **Abbreviation** | **Explanation** |
|---|---|
| - | - |
| - | - |
| - | - |

Table 1: List of commonly used Abbreviations

# Chapter 1: Background

When listening to music it is of great importance that the speakers are tuned to the environment and the position of the listener. This is necessary to achieve the best experience. If the speakers are not correctly tuned to the surrounding environment, a digital signal processor (DSP) is used to correct this. A DSP is a specialized processor which is used for digital signal processing.

In the audio world a DSP is used to optimize a sound system. For example some speakers have some imperfections and a DSP can be used to correct for these imperfections. It is also often used to add more dynamics to sound.

# Chapter 2: System context design

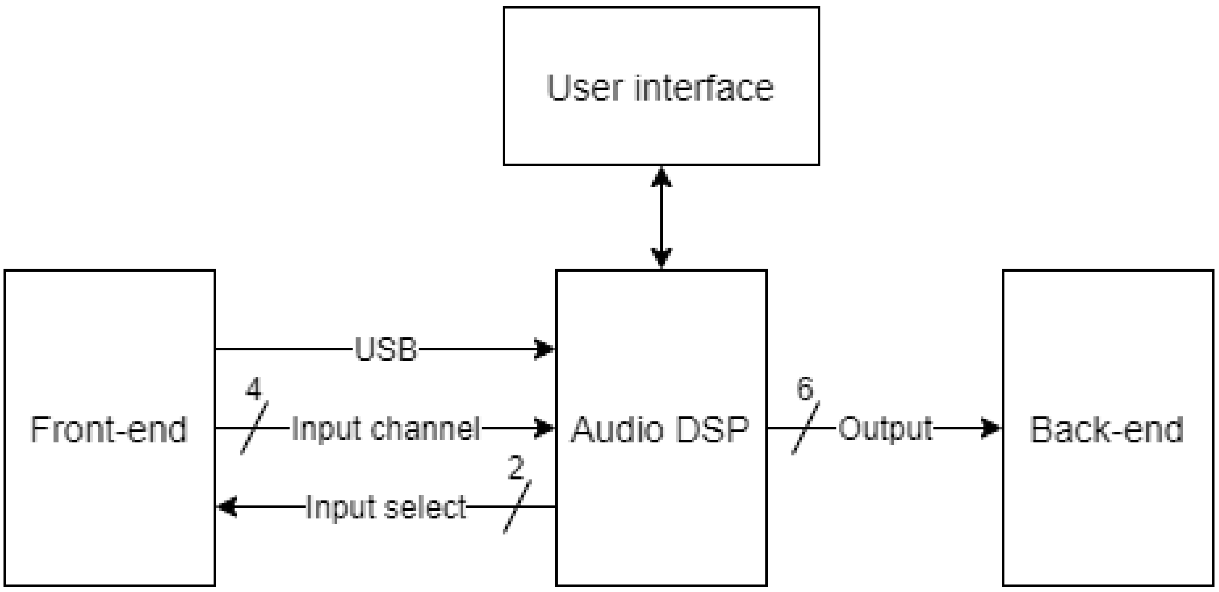

Figure 2.1: System context diagram of the top-level

After the system requirement document has been approved, the system context diagram could be made (see figure 2.1). The block called ”Audio DSP” is the heart of the system. This block represents the controller and thus the FPGA core. This system context design diagram fulfills all the requirements, including the should and could haves. Therefore the Audio-DSP has four analogue inputs, one USB input and six analogue outputs. The input select line is for selecting what line input you want on input channel 1 and 2. The user can either select a RCA or 6.35mm jack input on input channel 1 and 2.

With a user interface the user is able to configure the effect parameters, equalizer settings and volume of each channel. The user is also able to rearrange the position of effects in the effects loop per channel.

## 2.1 Front-end

## 2.2 Audio-DSP

The Audio-DSP block is made in the digital domain of the system (see figure 2.3). Therefore this block will be made in the FPGA. Each analogue input signal needs to be sampled in order for the system to be able to process the data. Thus each analogue input signal has a sampler block. The USB input signal has an USB decoder block as a sampler.

After the sampler blocks each sampled signal goes to six channels with each a 5 to 1 MUX (multiplexer). With this MUX the user is able to select what input signal will be processed

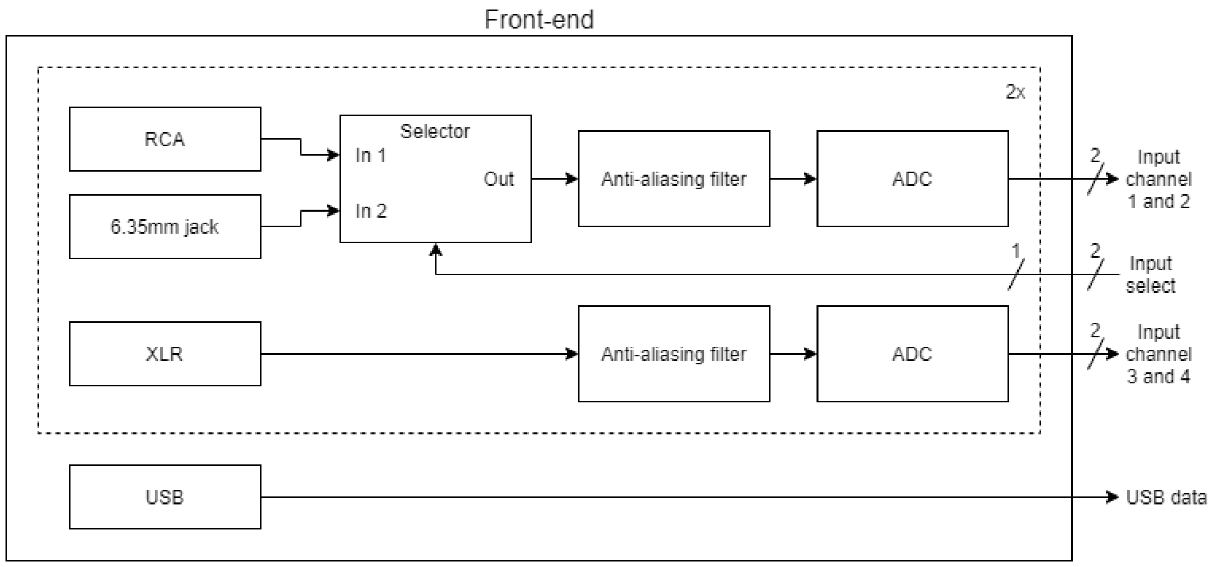

Figure 2.2: System context diagram of front-end design

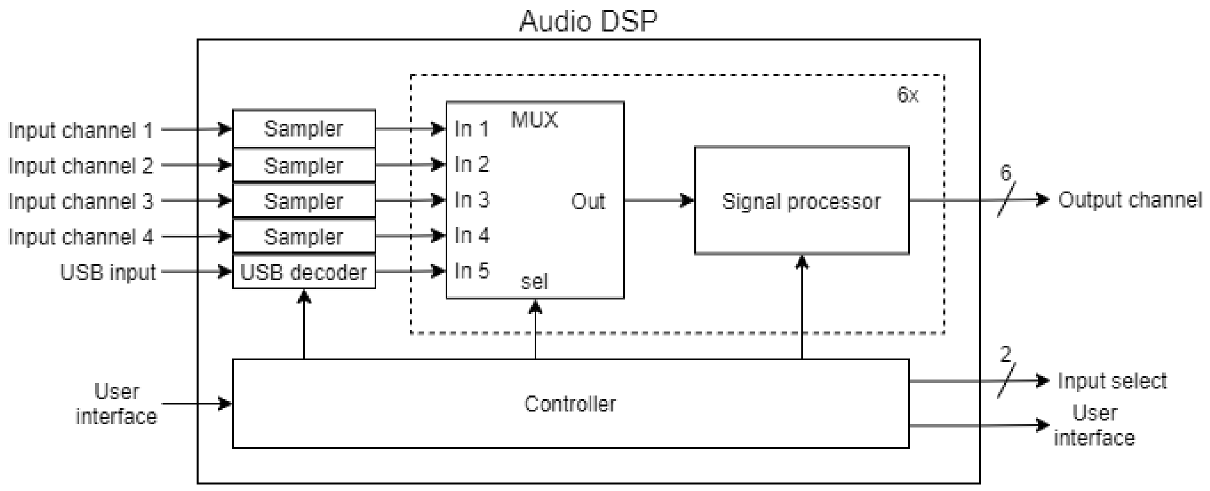

Figure 2.3: System context diagram of Audio-DSP

in each channel. The chosen signal will then go to the signal processor block. In the signal processor block the input signal will be modified by the various configurable effects, equalizer and volume settings. These effects, equalizer and volume configurations can be configured by the user via the user interface.

After the signal has been modified by the signal processor block it will be fed out of the FPGA to the back-end of the system.

## 2.3 Back-end

## 2.4 User interface

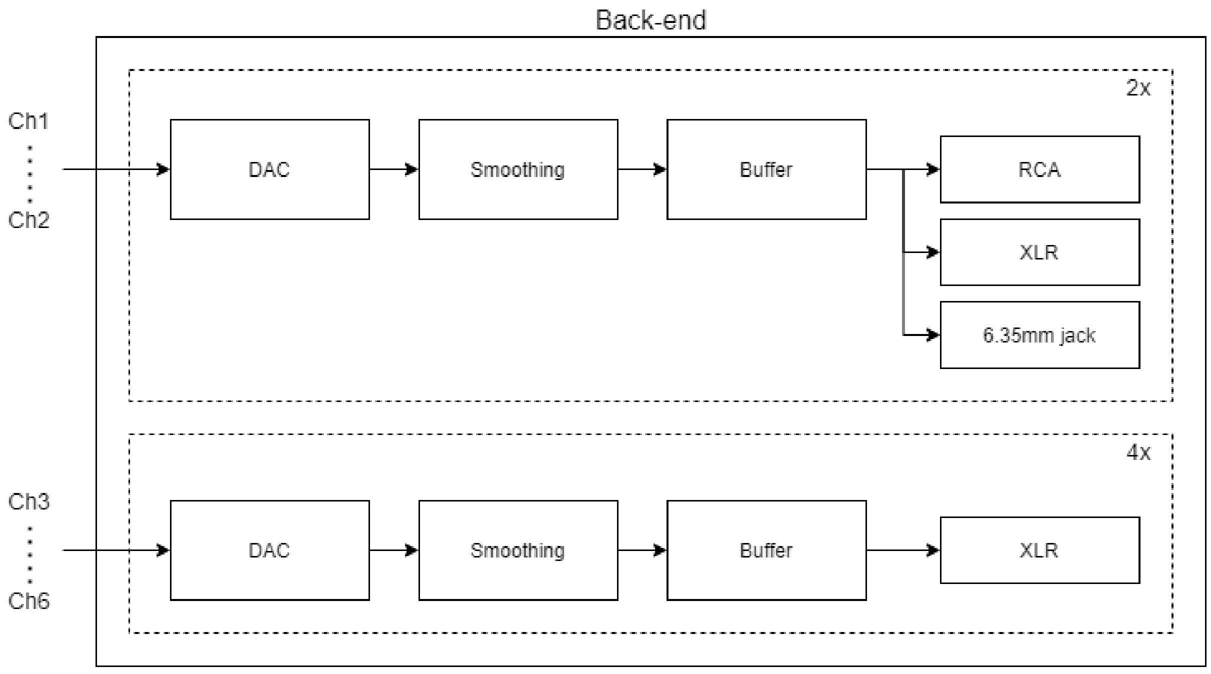

Figure 2.4: System context diagram of back-end

# Chapter 3: System Architecture

The lower level design of the system are explained in the architecture design of the system.

## 3.1 Audio-DSP

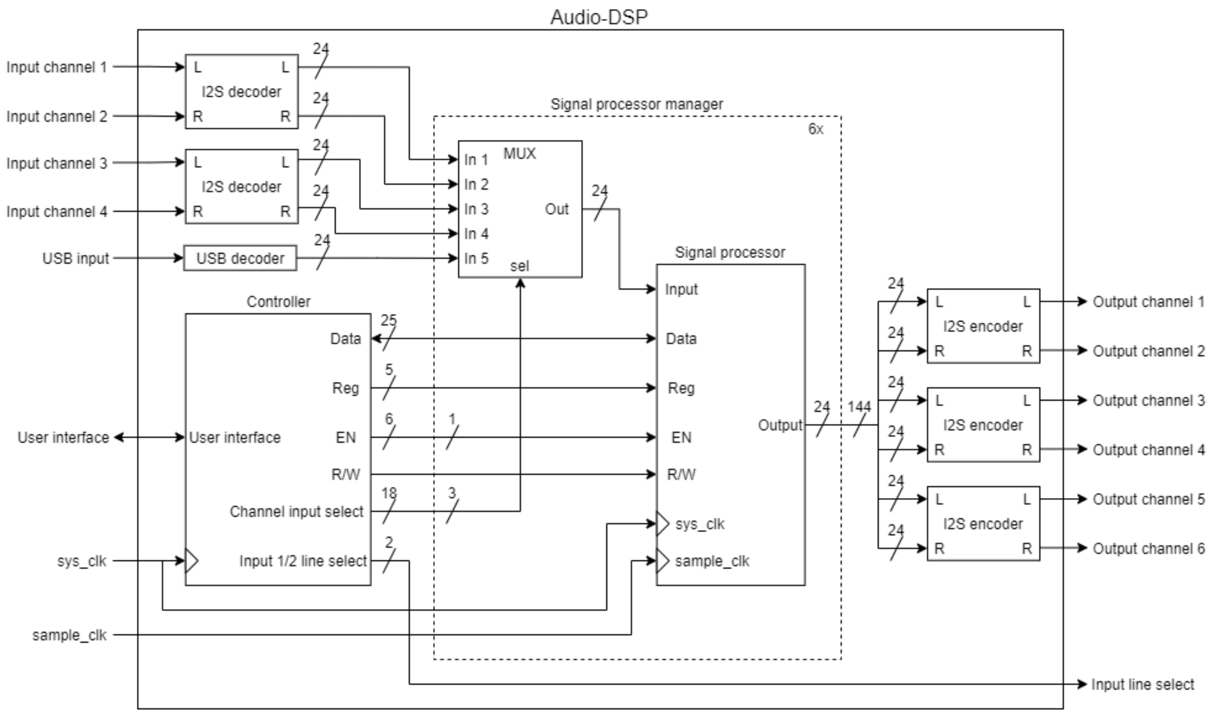

Figure 3.1: Top-level architecture of audio-DSP

In figure 3.1 you see the architecture design of the Audio-DSP. Compared to the system context diagram of the Audio-DSP the block is now much more detailed. For instance the input samplers are replaced by $I^2S$ decoders and the output signals are now encoded to $I^2S$. Because most ADCs and DACs use the $I^2S$ protocol to transfer audio data it has been chosen to use the $I^2S$ protocol. The $I^2S$ encoders and decoders each have a left and right input and output. This is because the $I^2S$ protocol transfers a stereo audio signal. But this system uses mono signals. Therefore the system inputs the mono signals on the left and right inputs. This gives the system the ability to transfer two audio channels via one $I^2S$ bus.

The MUX in each of the signal processor managers is controlled via a 3-bit select line. This select line comes from the controller. For the controller to be able to handle all the multiplexers in each signal processor manager, 3-bits $\cdot$ 6 channels = 18 bits are needed. To select the RCA or 6.35mm jack input a 2-bit select line is used.

In order for the signal processor to modify the signal with various digital effects, memory is needed. This memory is stored inside the signal processor block itself. To access and modify the memory some kind of communication protocol had to be chosen in order for the controller to configure the effect parameters. For this a register based memory has been chosen.

Looking at the system requirements it is known that the user is able to adjust the position of each effect in the effects loop. Therefore the signal processor needs registers to store the position of each effect. The system must support at least five effects and should support at least twenty effects. Thus in order to fulfill all the requirements the signal processor should be able to position twenty effects. For this we would need at least 5 bits at each position. The various effects available to the user have configurable parameters. Therefore each effect also needs a register in the signal processor block. Also the equalizer settings are adjustable by the user. This means the equalizer also has a register with data.

The size of the registers can be very large. But to make the communication to the signal processor intuitive the registers will be limited to a maximum amount of bits. In order to choose the most efficient register size the registers should use most of its bits. The size of the register is of no importance for the effect and equalizer parameters as the size only affects the resolution of the parameters. But for the position register 5-bits per position are needed. Thus the register should be dividable by 5 in order to use the most of the registers bits. As there are 20 positions it is chosen to have four position registers with 5 positions (see figure 3.2). This means the register size will be 25 bits.

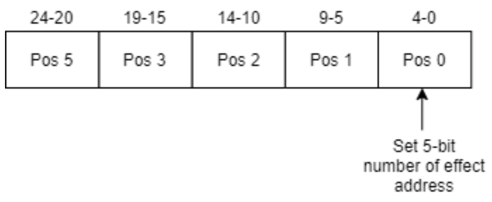

Figure 3.2: Effect loop position 0 to 4 register

Now that the register size has been defined we need to define the amount of registers needed. The system has 20 effects, therefore 20 effect parameter registers. Each effect needs to be positioned in the position registers which there are four of. Then the equalizer and volume registers are left. when using 5 bits for selecting the registers we would be able to access up to 32 registers. This means that there are $32 - (20 + 4) = 8$ registers left for the equalizer and volume parameters. That is more than enough for these registers.

Thus the data bus will be 25 bits and the register selector line will be 5 bits. With these two lines every register can be accessed by the controller. Now the memory needs to know if the data needs to be read or written. This is indicated by the R/W signal. When this signal is low the memory will be written and when the signal is high the memory will be read.

It is undesired that every signal processor memory will be read or written constantly. Therefore each signal processor block has an enable. When this enable signal is high the memory can be read or written. This gives the controller the ability to read or write to only one signal processor memory at a time. And the ability to write multiple signal processor memories at once.

## 3.1.1 Signal processor

The signal processor has to load a new sample into the effect loop and the previous modified sample to the output on the sample frequency. The sample frequency of the Audio-DSP is 192kHz. This means that the signal processor has $\frac{1}{192 \cdot 10^3} \approx 5\mu s$ to process the sample. With a

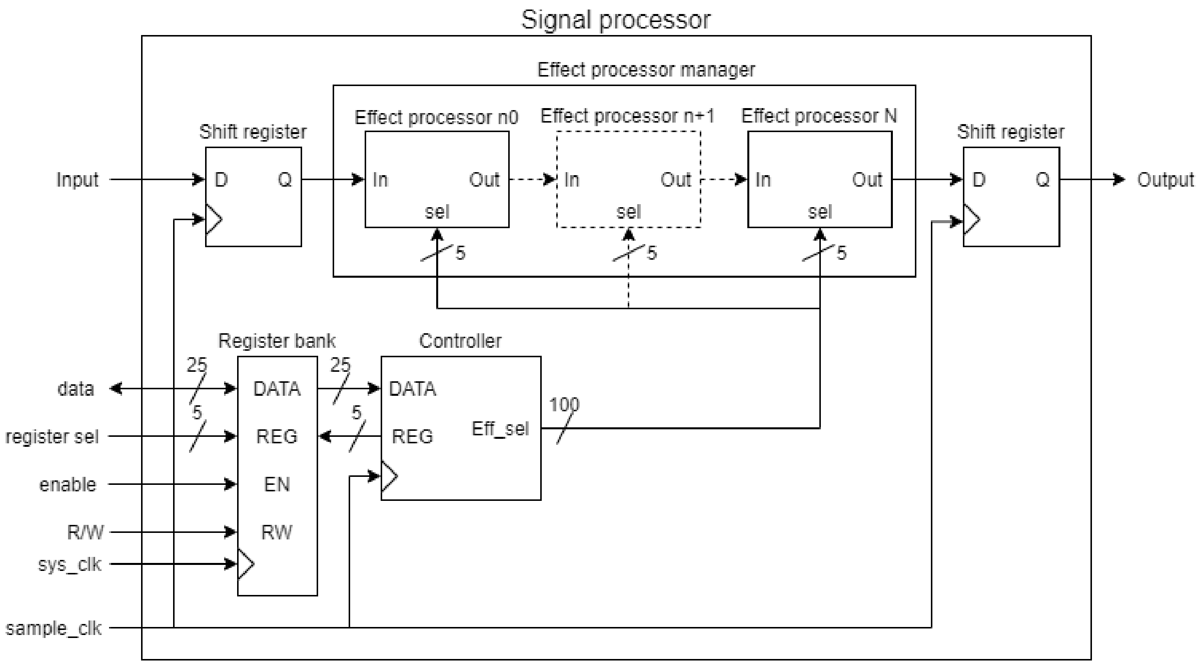

Figure 3.3: Signal processor architecture

clock of 50MHz the sample must be modified and loaded into the output within $\frac{50 \cdot 10^6}{192 \cdot 10^3} = 260$ clock ticks.

When a sample has been loaded into the signal processor, it is fed into the effect processor manager (see 3.3). The effect processor manager houses multiple effects processor in series. The sample will go through these effect processors. Each effect processor can be configured to be a certain effect (see 3.4). Because the system has at least twenty effects, the amount of bits the selector needs is 5 bits.

Each effect modifies the signal by applying a transfer function to the sample. The variables in that transfer function are configured by the user via the user interface. Therefore the effect processors needs to get the variables from the controller block, which gets the variables from the register bank. After the sample has gone through all the effect processors the modified sample waits before it is shifted out of the signal processor.

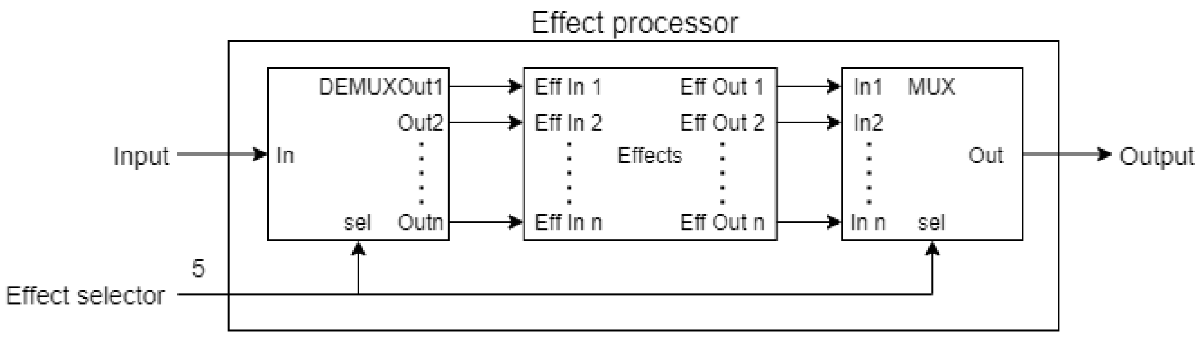

Figure 3.4: Effect processor architecture

# Chapter 4: Detailed Design

## 4.1 Hardware Design

## 4.2 Software Design

# Chapter 5: System Interfaces

# Chapter 6: Human Machine Interface

## 6.1 User interface

A well-designed user interface is a critical component of any audio DSP project, particularly when utilizing a Nextion or similar screen. The user interface should be designed with simplicity in mind to ensure it is accessible to everyone who uses it. One of the primary goals of a good UI is to ensure that it is intuitive and easy to remember, so that users can begin using the product without feeling frustrated or overwhelmed.

#### Consistent

The UI should maintain a consistent style throughout, so that each new menu or dial looks and feels the same as every other menu. This ensures that using the menus is easy and recognizable, even for new users.

#### Readability

Text shown in menus should not be cut off, as this can be frustrating for users trying to read it. Short sentences are preferred to keep the menus clean and easy to read. When a sentence is cut off by the edge of the screen, users may struggle to figure out what it says, leading to confusion and frustration.

#### Feedback

Finally, it is important to provide feedback to the user when the system needs time to load in certain elements or execute specific settings. Without feedback, users may become frustrated and begin clicking buttons multiple times, which can lead to software errors. By providing clear feedback during loading processes, users are more likely to remain patient and avoid potential issues.

## 6.2 Designing the UI

The design was started with a diagram with all the necessary screens and information so that is was clear what menus should be in the DSP.

This UI is easy to understand when first using the DSP. Options and settings are easy to find from the first menu screen. While a channel based UI is unclear because every option is branched off from the channel select.

## 6.3 Nextion screen

For this project a Nextion screen is used. This screen can easily be programmed. It comes with software which is image based instead of code based. Images can be inserted and ”invisible”

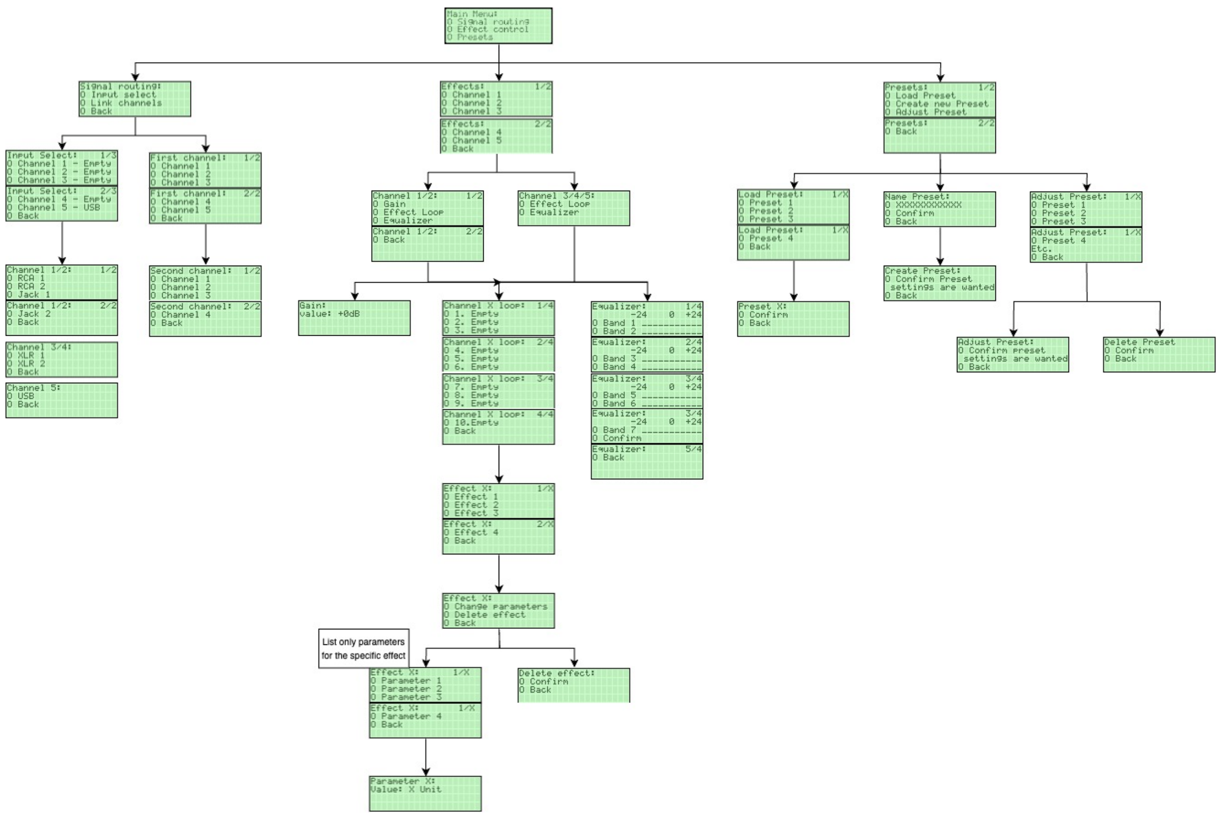

Figure 6.1: Function Based UI Design for a liquidCrystal screen

buttons can be used to make menus out of the inserted images.

This is a neat way to work because now it is possible to create images in Photoshop or a similar application to use for the whole menu design.