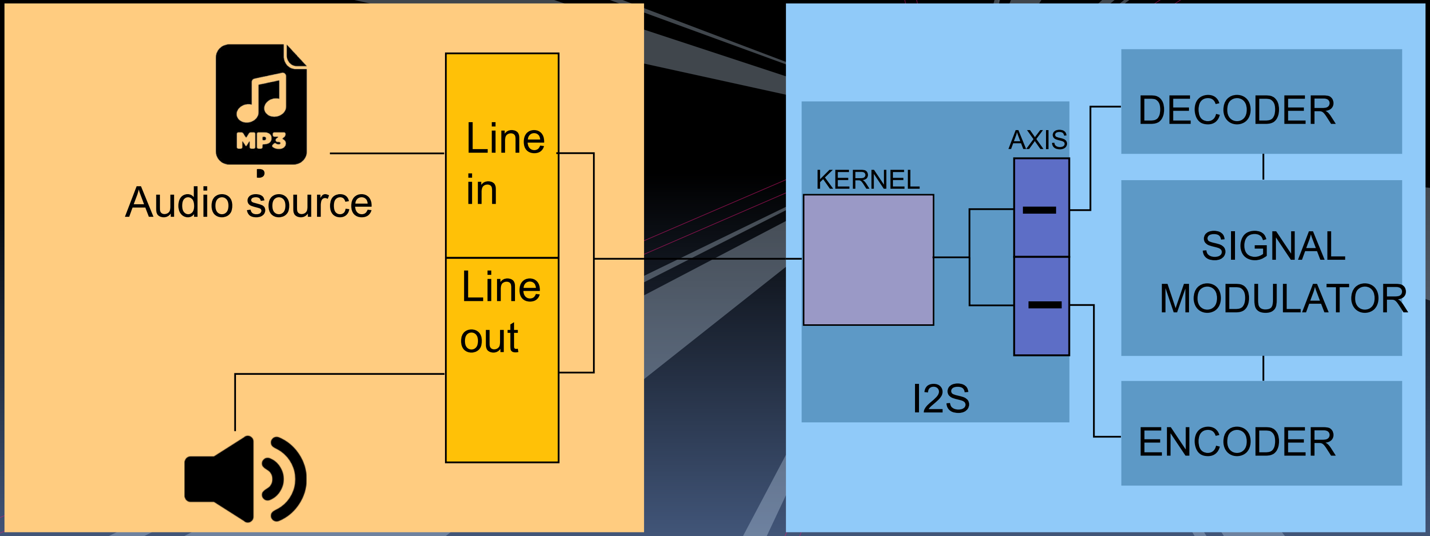
LAB3 OVERVIEW

Topics

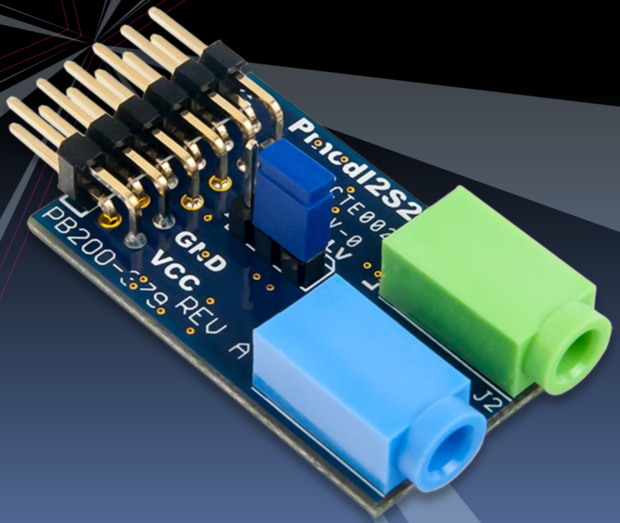
SPI module interfacing, audio source sampling, I2S "Implementation", Encoder/Decoder, Signal Manipulation



Home assignment

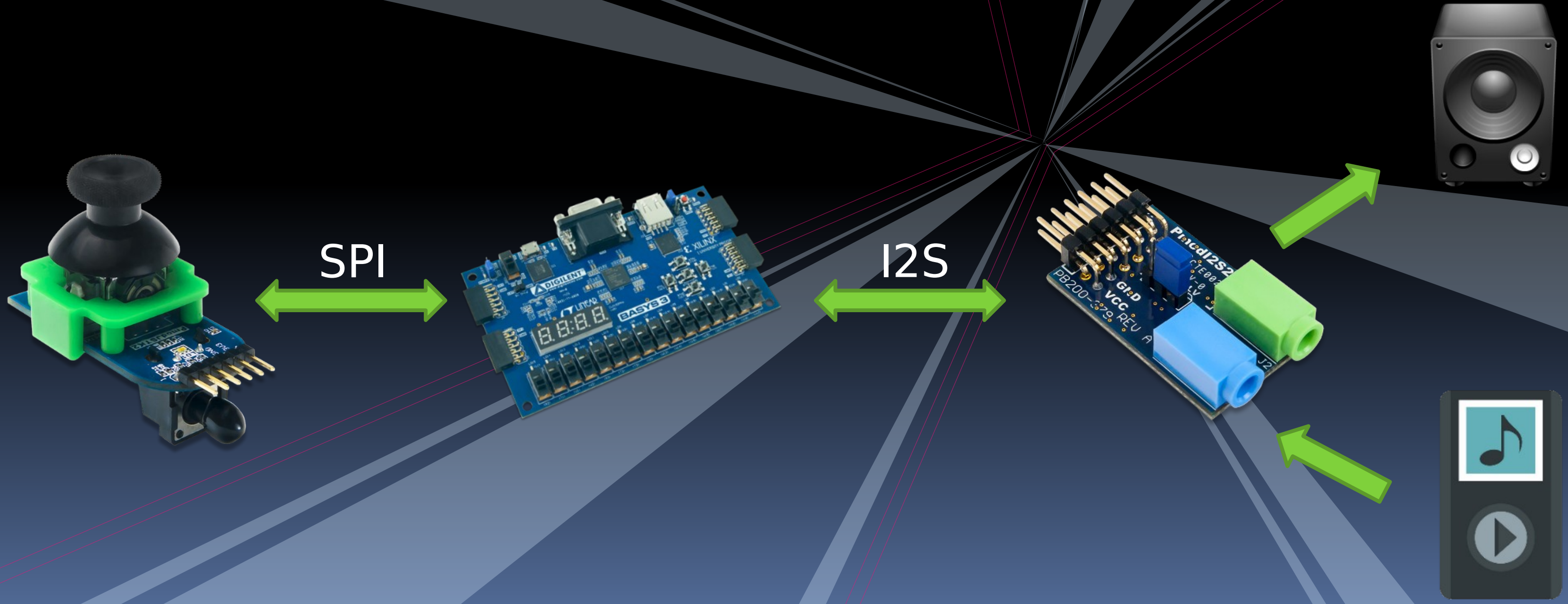
At the beginning of LAB3, we will give each one of you a Digilent Pmod Joystick SPI module and Digilent Pmod I2S2 module.

This can be connected to your Basys3 board through the Pmod connectors.



Home assignment Goal

The objective of LAB3 home assignment is to build a “Digital Audio Console” by using the Pmod_I2S2, Pmod_JSTK2 modules and IP-Cores



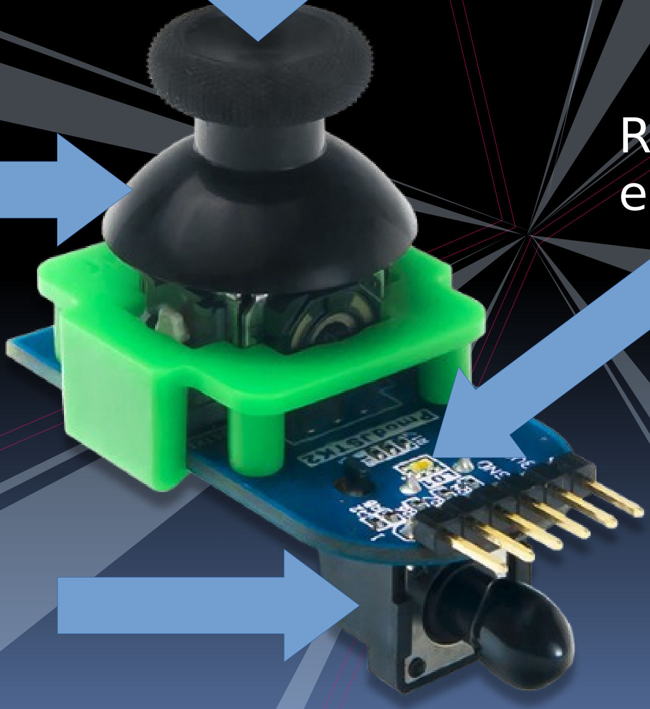
Digilent Pmod JSTK2 components

Joystick button (1 bit, pressed/not-pressed)

Joystick position (10 bits X-axis, 10 bits Y-axis)

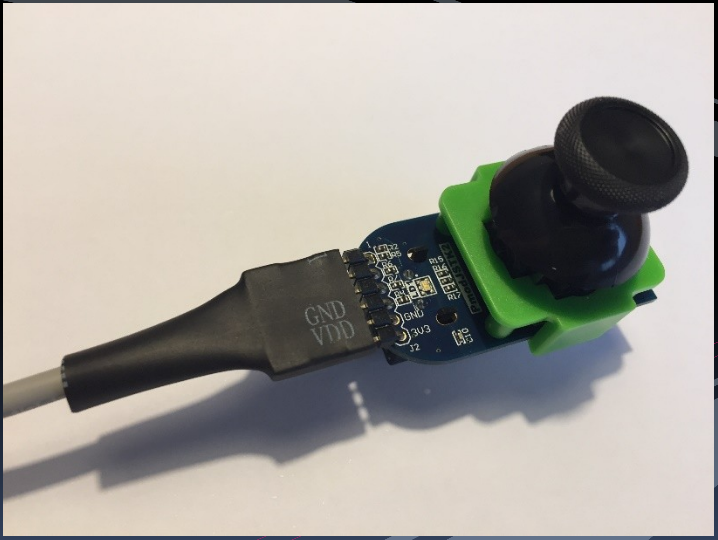
RGB LED (8 bits for each channel)

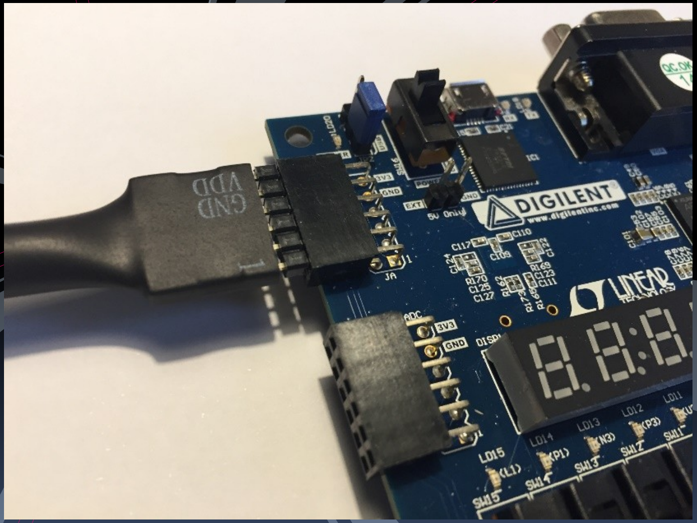
“Trigger” button (1 bit, pressed/not-pressed)



How to connect the Joystick

Connect the Joystick with the provided cable to “JA”, top row, paying attention to the VCC and GND position.





JSTK2 interfacing

The Digilent Pmod JSTK2 module protocol is fully described in its reference manual.

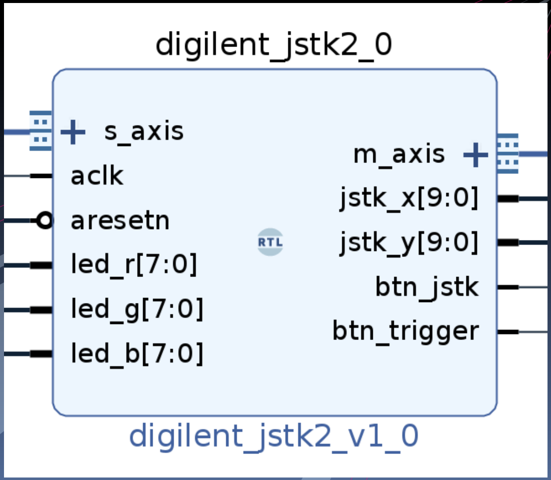
In short, it uses the SPI protocol to receive “commands” and send back the “readings” of the Joystick position and the buttons state.

See slides/recording from March 26th

Home assignment Goal #1

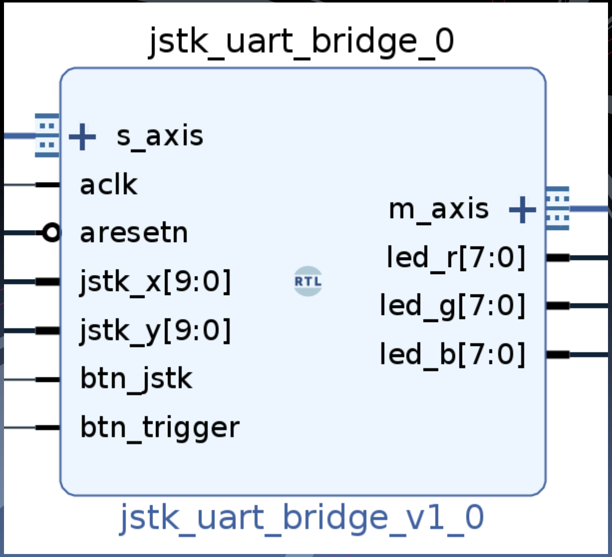
You will have to build 1 module:

"digilent_jstk2", to "talk" to the module and expose its "state" (jstk_x, jstk_y, jstk_btn, trigger_btn) and "controls" (led_r, led_g, led_b) as std_logic or std_logic_vector.



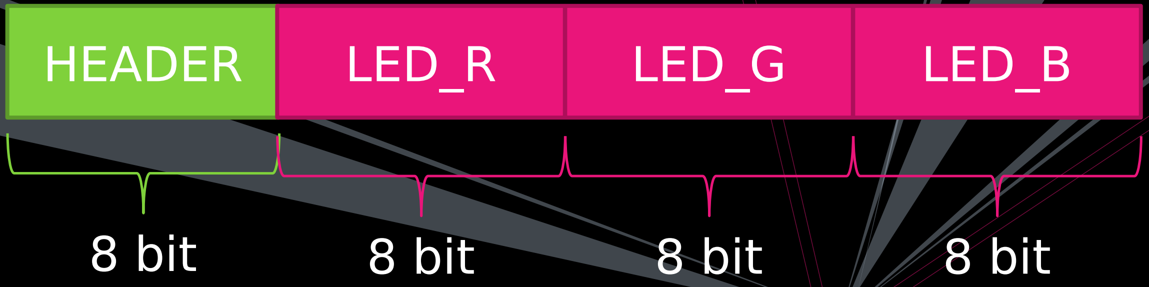
Check Goal #1

To test your digilent_jstk2 interface you can use this jstk_uart_bridge (provided), to send and receive those values to the UART IP-Core and see them via a serial terminal.

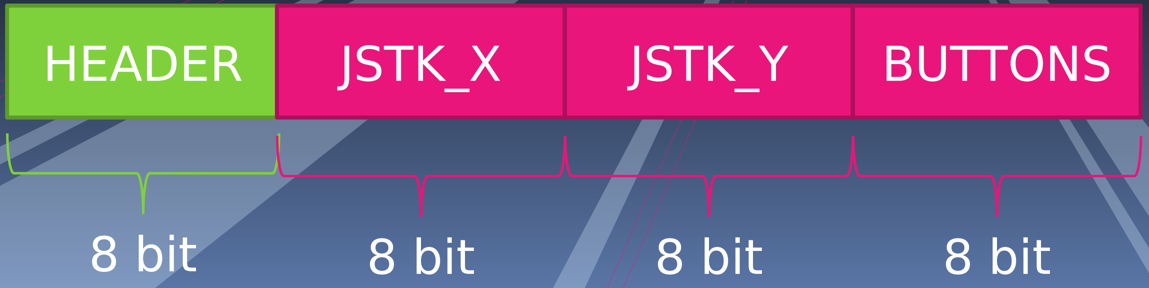


Check Goal #1

You need to send a 4-byte Packet from the serial terminal composed as follows:

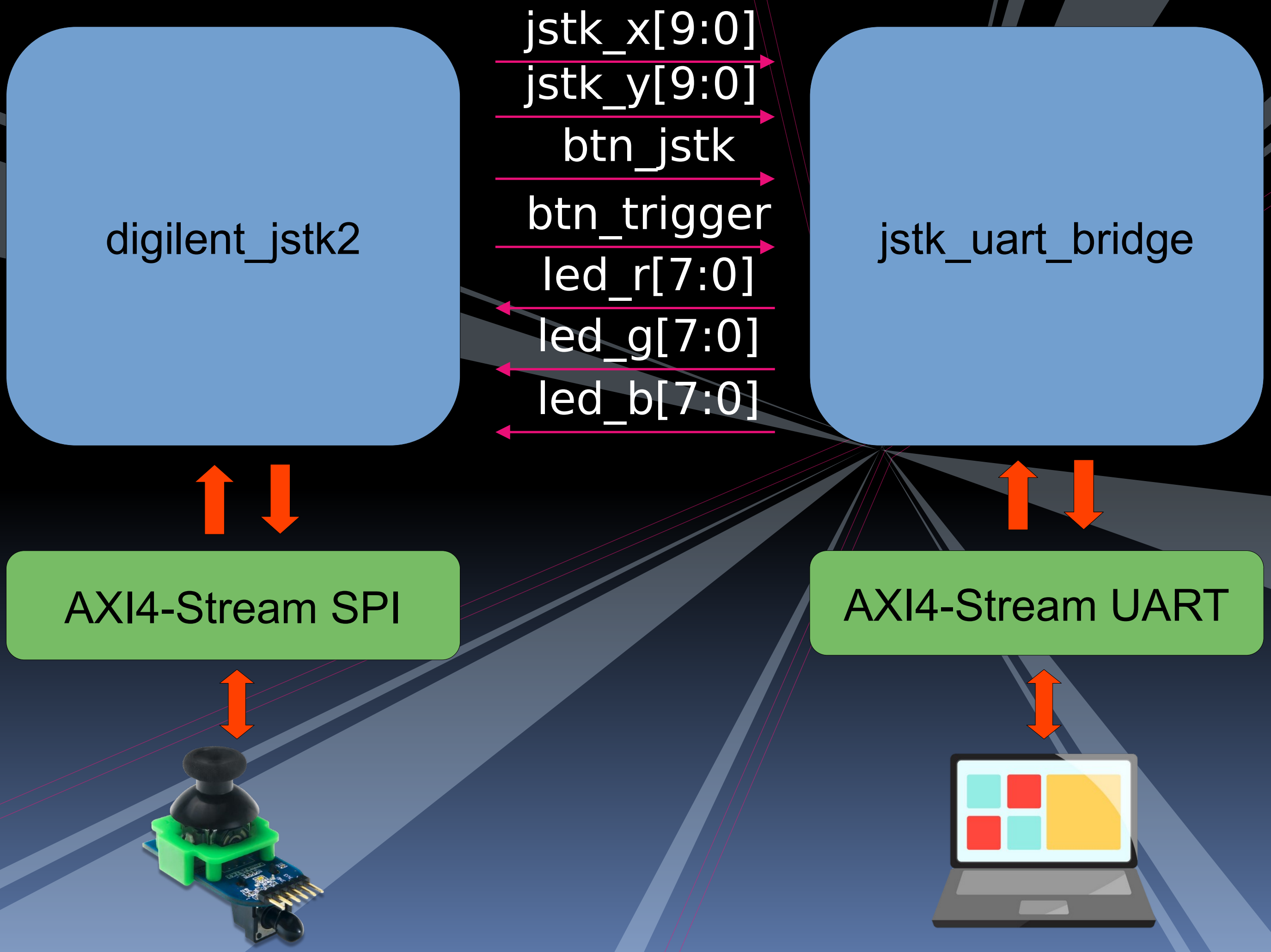


You'll receive back (if everything works fine) packets like this:



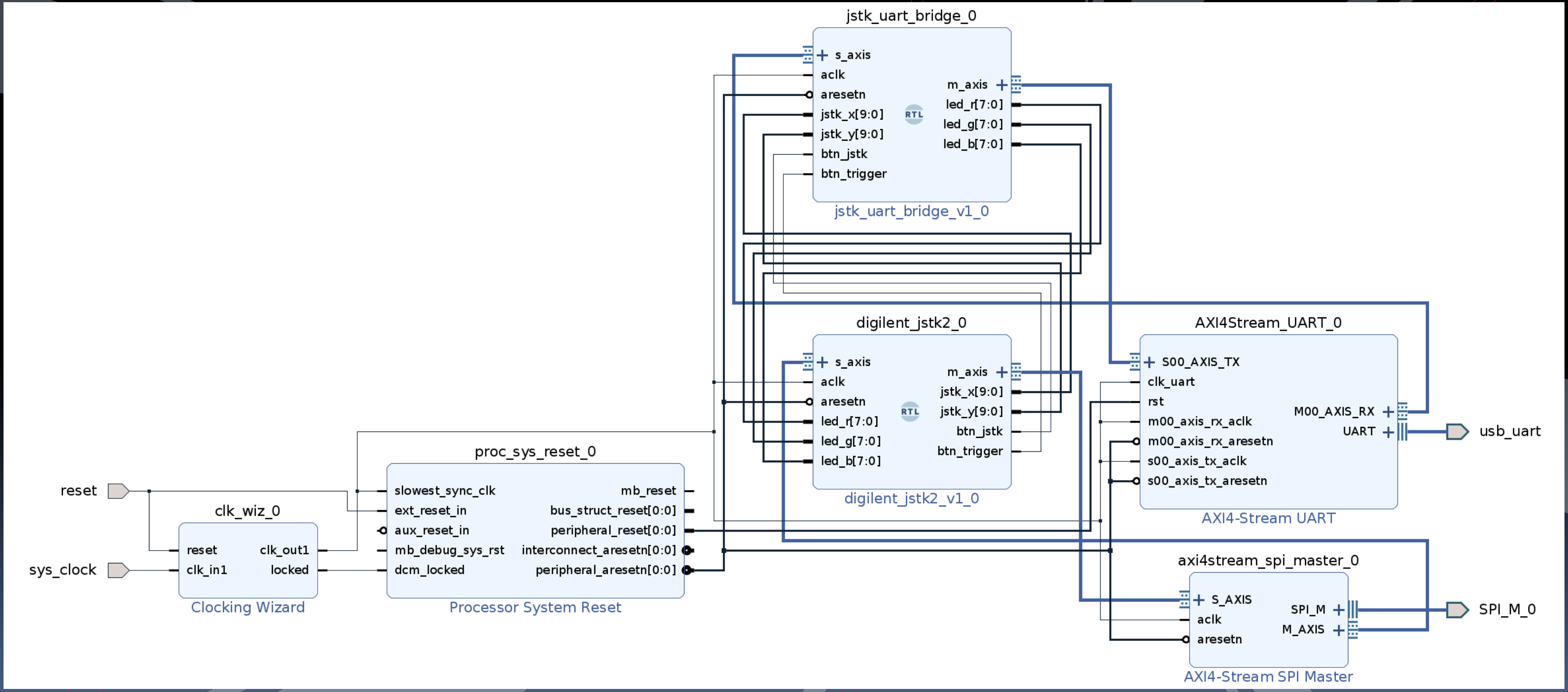
Check Goal #1

At the end, your test project should have this structure:



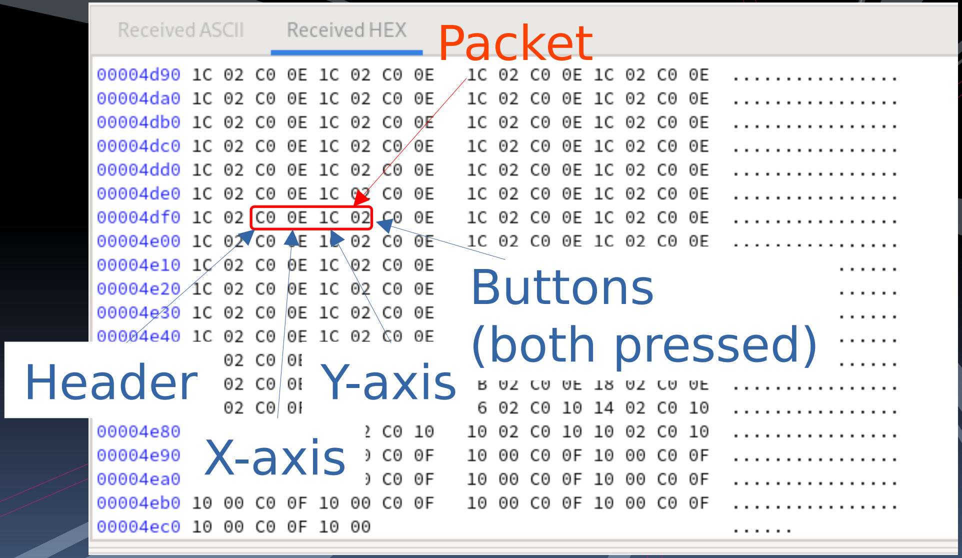
Check Goal #1

At the end, your test project should have this structure:



Testing: FPGA → PC

Open your serial terminal and see if you receive the expected data when you move the joystick and press the buttons:



```
00004d90 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004da0 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004db0 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004dc0 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004dd0 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004de0 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004df0 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004e00 1C 02 C0 0E 1C 02 C0 0E   1C 02 C0 0E 1C 02 C0 0E  ................
00004e10 1C 02 C0 0E 1C 02 C0 0E                                       ......
00004e20 1C 02 C0 0E 1C 02 C0 0E                                       ......
00004e30 1C 02 C0 0E 1C 02 C0 0E                                       ......
00004e40 1C 02 C0 0E 1C 02 C0 0E                                       ......
            02 C0 0                                                    ......
            02 C0 0                 B 02 C0 0E 18 02 C0 0E  ................
            02 C0 0                 6 02 C0 10 14 02 C0 10  ................
00004e80                  C0 10   10 02 C0 10 10 02 C0 10  ................
00004e90                  C0 0F   10 00 C0 0F 10 00 C0 0F  ................
00004ea0                  C0 0F   10 00 C0 0F 10 00 C0 0F  ................
00004eb0 10 00 C0 0F 10 00 C0 0F   10 00 C0 0F 10 00 C0 0F  ................
00004ec0 10 00 C0 0F 10 00                                  ......
```

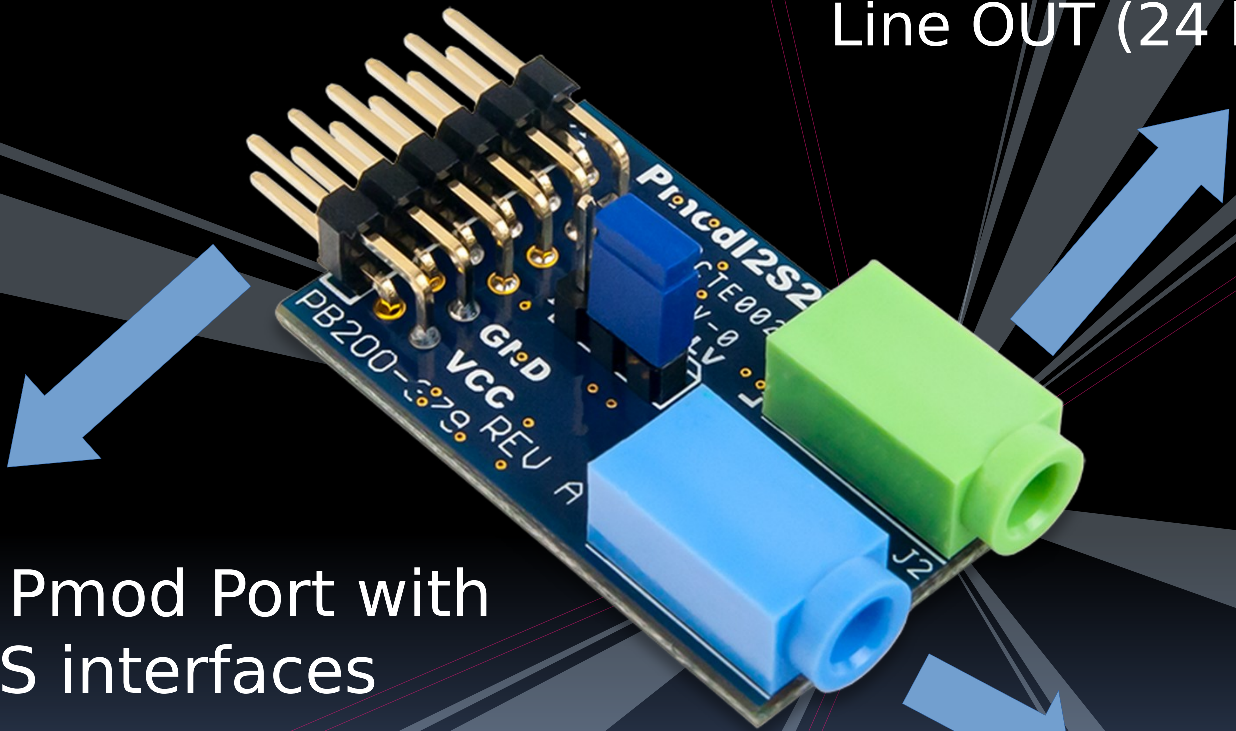
Testing: PC → FPGA

Send some data in the proper format and see if the LED changes color.

For example:

- `C0 ff ff ff` → white
- `C0 00 00 00` → OFF
- `C0 ff 00 00` → red
- `C0 ff ff 00` → yellow
- `C0 10 00 10` → light purple

Digilent Pmod I2S2 components



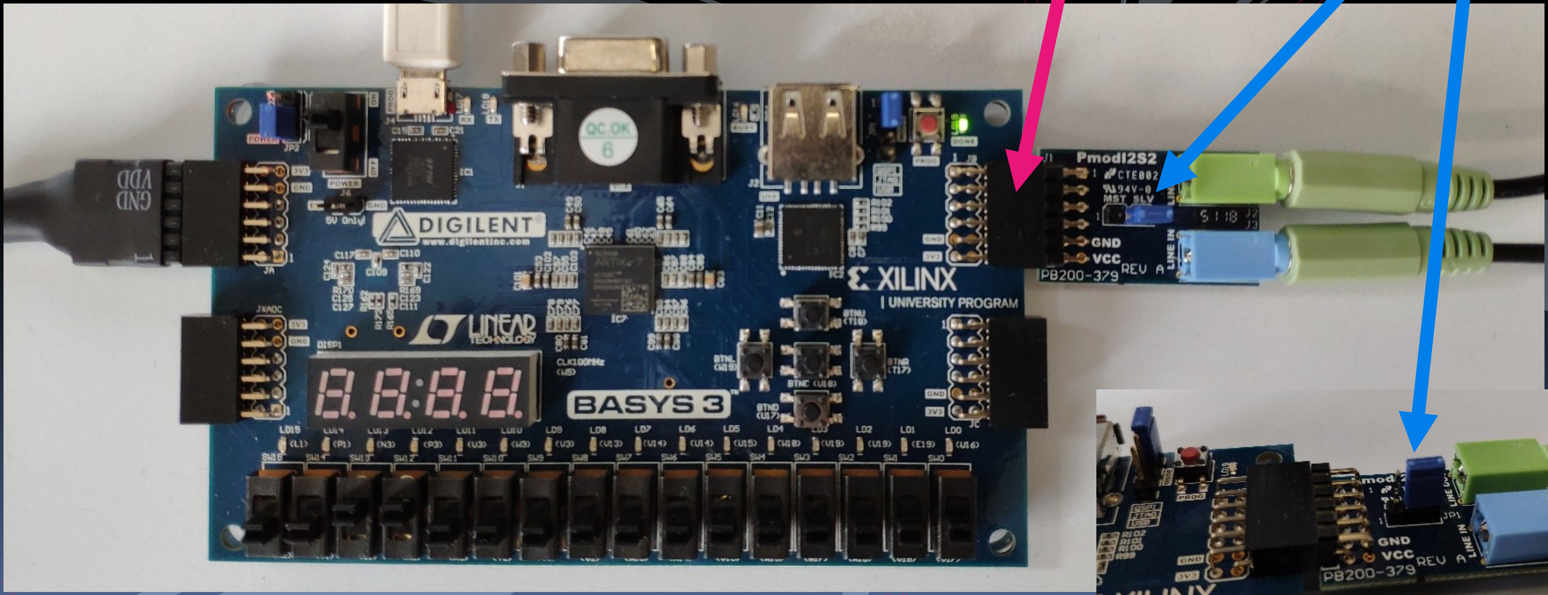
Line OUT (24 bit D/A)

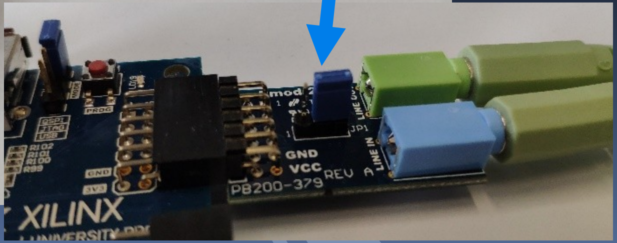
12-pin Pmod Port with **two** I2S interfaces

Line IN (24 bit A/D)

How to connect the I2S2 module

The provided constraints are for the JB connector (top right of the board). Also make sure that the jumper on the PmodI2S2 module is on the **SLV** position (right position).





Pmod I2S2 interfacing

The Digilent Pmod I2S2 module protocol is fully described in its reference manual.

In short, it uses the I2S protocol for both receiving an input audio signal from a source (through an ADC) and sending back an audio signal by means of a DAC to any kind of "speaker".

Audio Signal Format

- Typical sound frequency passband of the ears is between 20 and 20KHz.
  (Shannon theorem => 44.1 Ksamples)

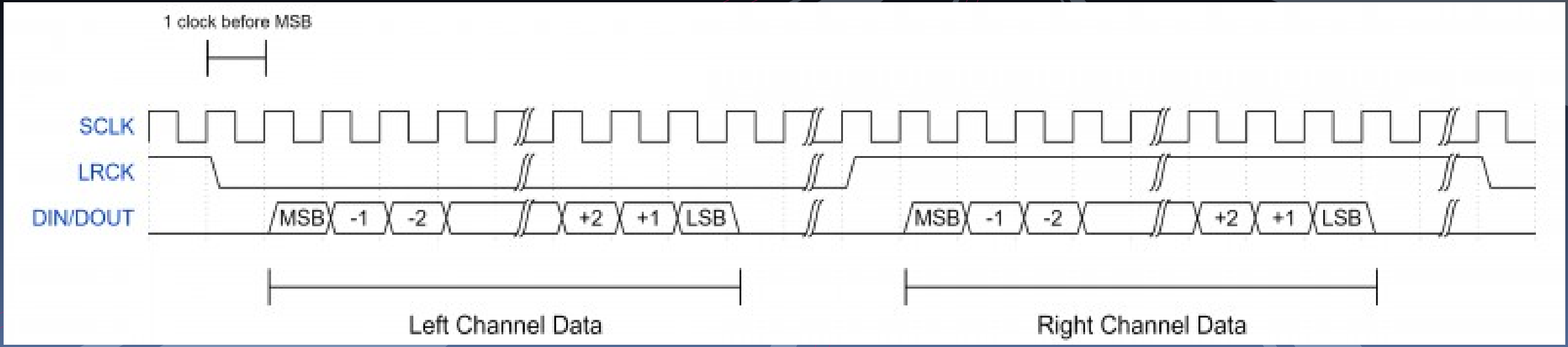
- For a good quality audio we choose a 24bits/channel depth.
  (we consider stereo audio: L/R channels)

I2S protocol

The Inter-IC Sound (I2S) is a popular serial protocol for digital audio devices connections.

In its basic form it is composed by 4 signals:

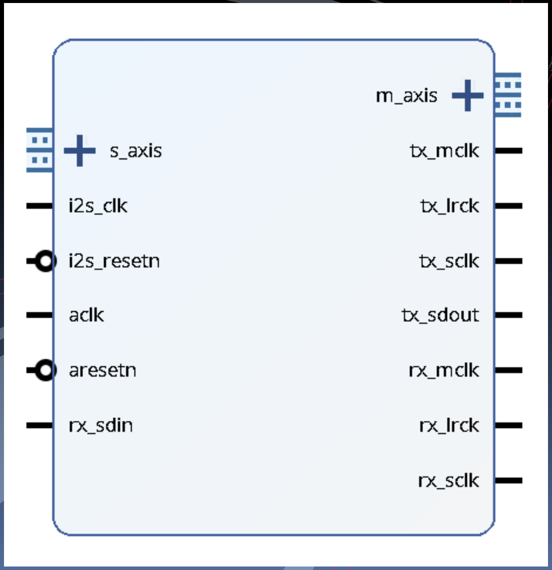
- Master CLocK (MCLK)
- Left/Right CLocK (LRCLK, aka Word Select)
- Serial CLocK (SCLK, aka Bit Clock)
- Din/Dout (for Line In/Line Out channels)



I2S IP-Core

While I2S is a simple protocol, describing it in VHDL is not immediate.

To ease your work, we will give you a “AXI4-Stream I2S2” IP-Core, similar to the SPI and UART ones.



I2S IP-Core

The provided Pmod-I2S2 IP-Core has:

- Two I2S interfaces, to be connected to the external pins of the FPGA.
- Two AXI4-Stream interfaces, to read the data from the ADC or to send the data to the DAC.
- Two clocks:
  - one for the I2S and (22.579 MHz)
  - one for the AXI4-Stream
- Two input active-low reset signals (one for each clock domain).

I2S IP-Core: 44.1 kHz Audio

The IP core needs a 22.579 MHz clock (to be connected to i2s_clk) and a 100 MHz clock (to be connected to every other clock input).

In details, for a 44.1 kHz audio sampling rate:

MCLK = 22.579 MHz (I2S Clk of the IP-Core)

SCLK = MCLK/8

LRCLK = SCLK/64 = 44.1 kHz

Pmod-I2S2 AXI4-Stream format

The AXI4-Stream interface has an additional line called **TLAST**, which is used to determine the end of a packet.

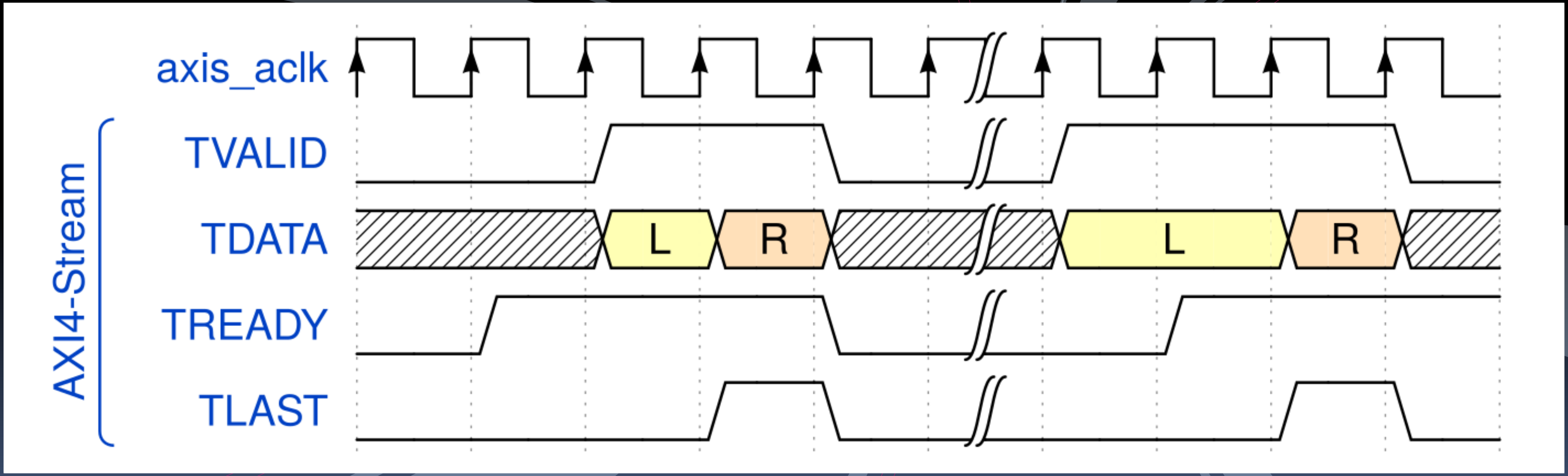
Each packet is composed by two 24-bits words: the first one is the audio data of the left channel, the second one the audio data of the right one.

TLAST is asserted on the second word; in other words:

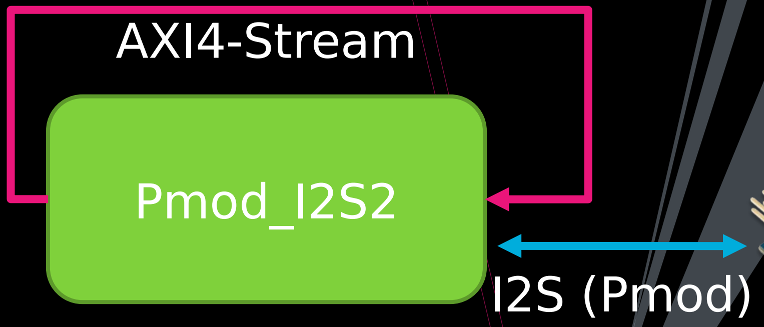
- TLAST = 0: left channel
- TLAST = 1: right channel

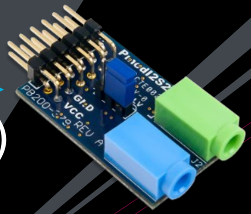
Pmod-I2S2 AXI4-Stream format

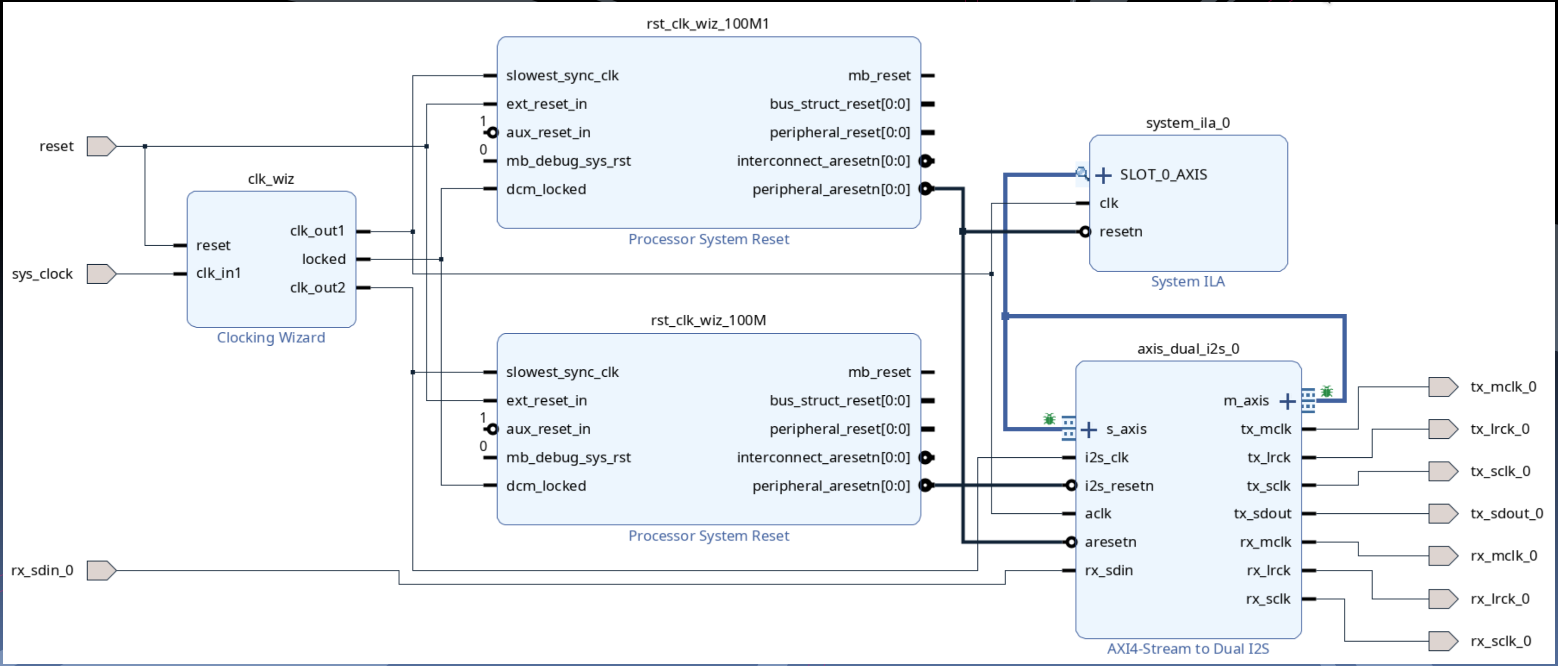
In this example, two «packets» have been transferred, first the left channel and then the right one for each one.



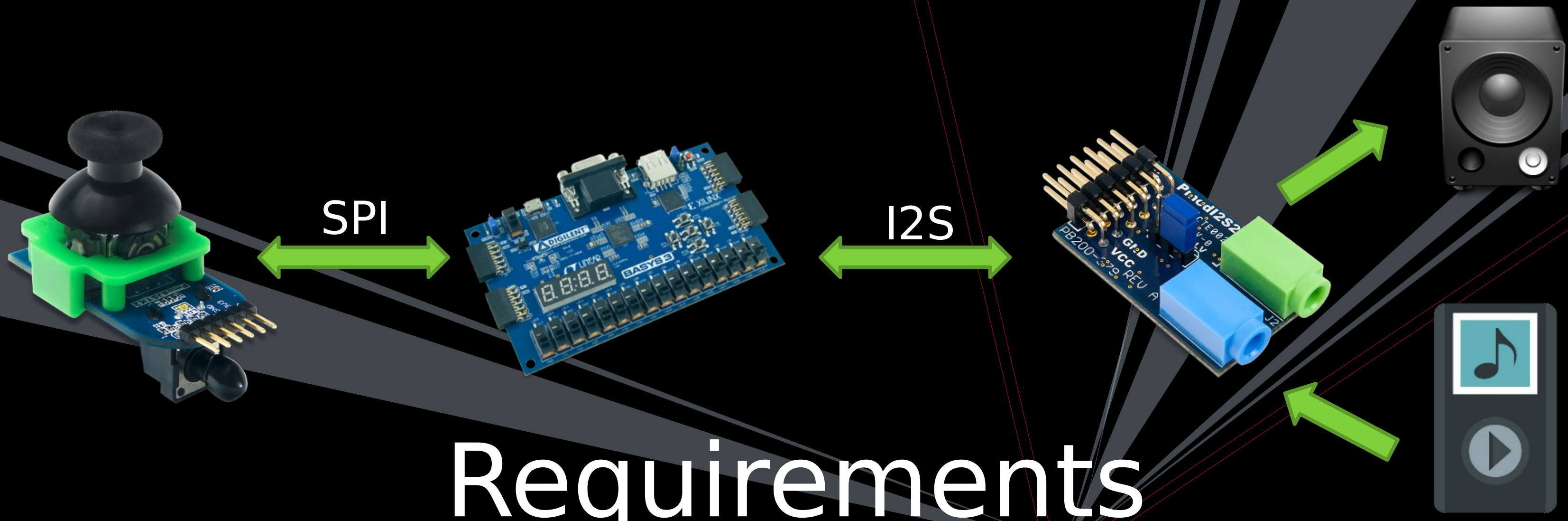
First step: loopback test







Back to Home assignment (1/4)



Requirements

The output audio should reproduce the input one with some “effects” applied:

- If no button is pressed, the vertical axis of the Joystick should control the volume of the output audio.
- If no button is pressed, the horizontal axis should control the audio balance.

Back to Home assignment (2/4)

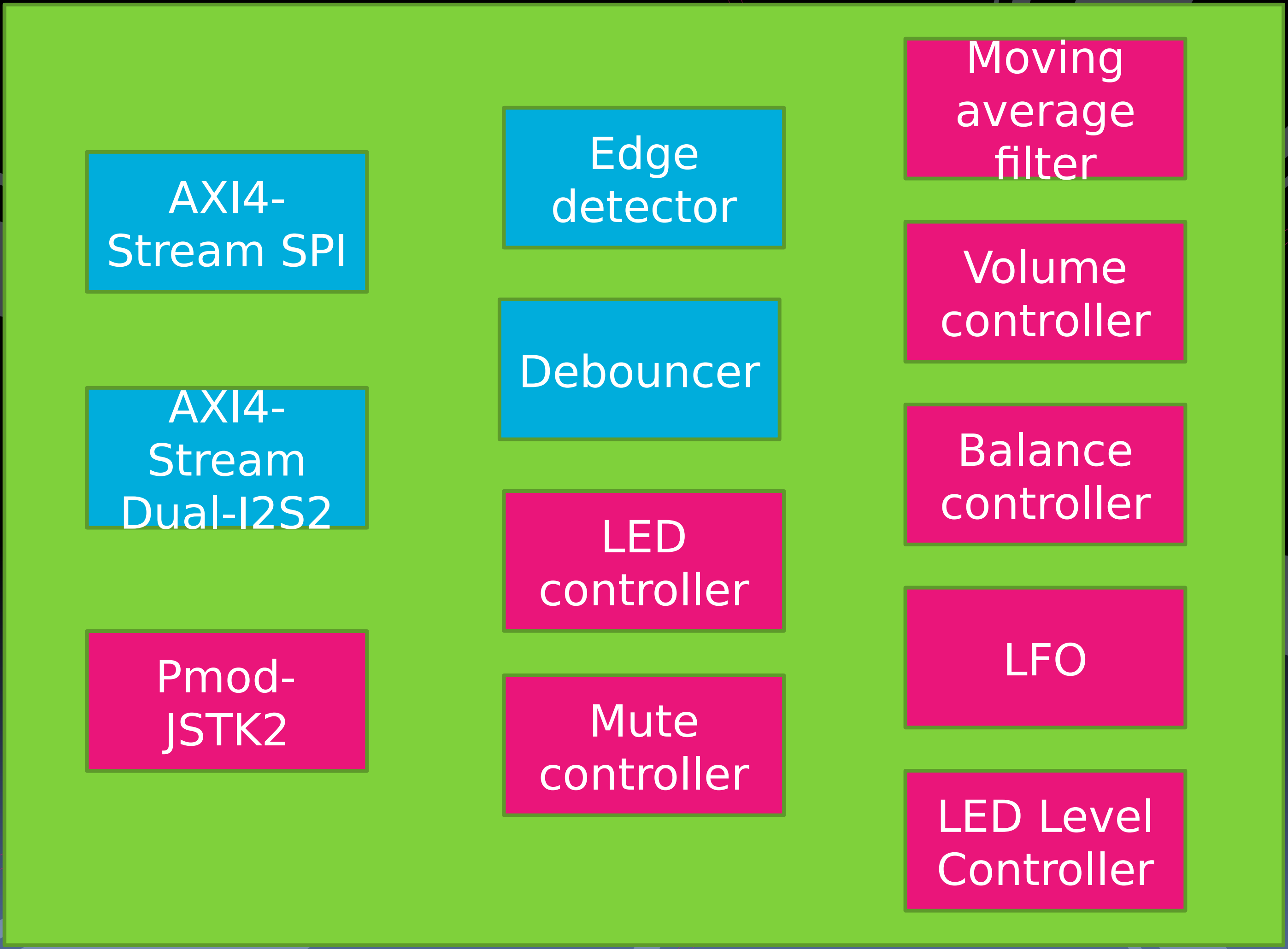
- If btnU is pressed the joystick vertical axis controls the LFO period.
- If btnU is pressed, moving the joystick horizontal axis has no effect.
- Turning SW0 on enables the LFO effect.
- The 16 LEDS of the Basys3 should turn on in a number proportional to the mean of the left and right audio sample

Back to Home assignment (3/4)

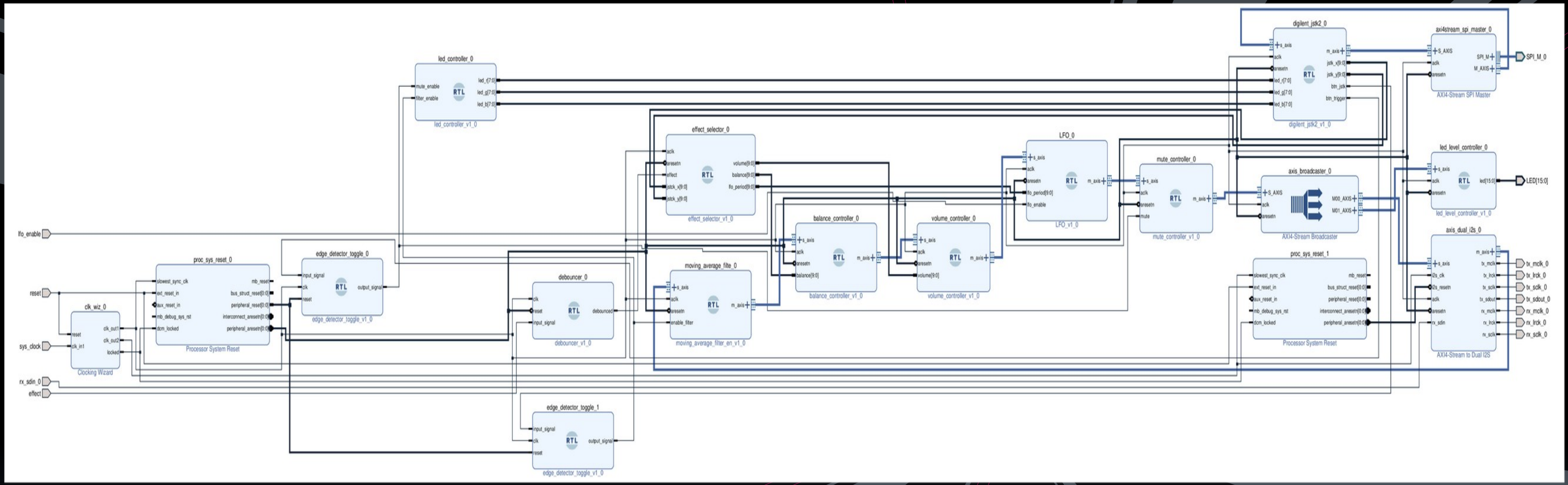
- Pushing* the “trigger button” mutes or unmutes the output channel.
- Pushing* the “joystick button” enables or disables a moving average filter (depth=32).
- The LED on the PmodJSTK2 module should show the status:
  muted (red), filter active (blue),
  no effects (green).

* toggles the status, not just “active when pressed”

Back to Home assignment (4/4)

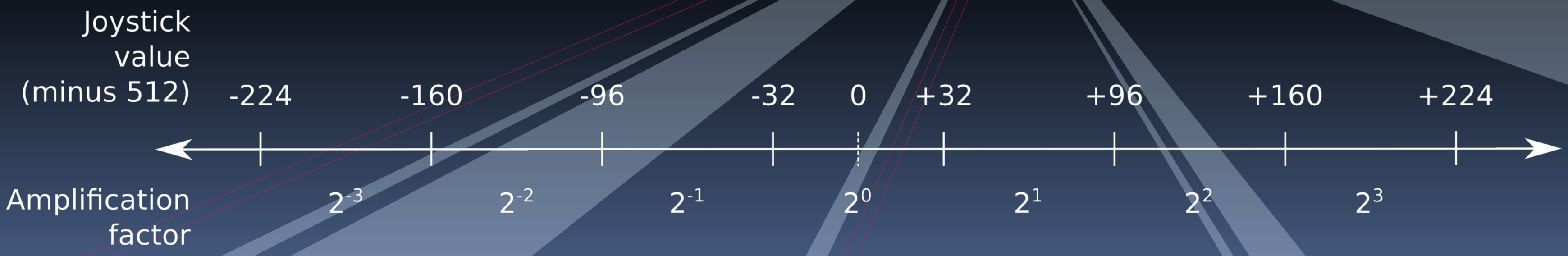


Block Design



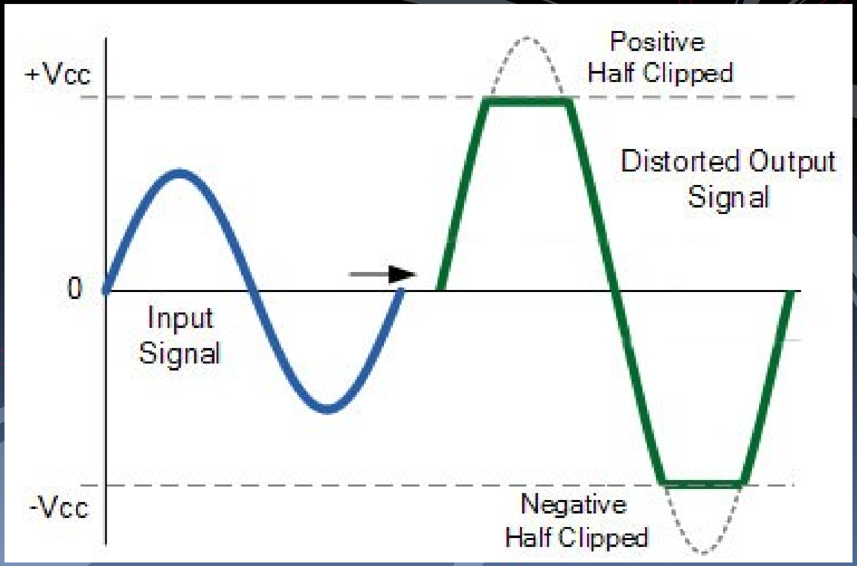
Details: volume (1/2)

We perceive “loudness” in a logarithmic way, so the volume control should be exponential. The amplification factor should double every 2^N “joystick units” (with the center in the half of the joystick dynamic, with N as generic).
N=6 returns good results.



Details: volume (2/2)

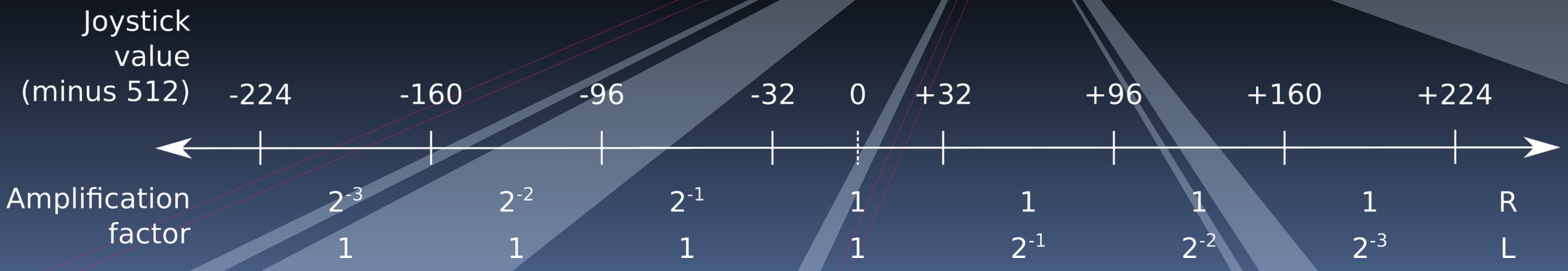
Be careful when multiplying: the signal must saturate at the maximum possible value (“clipping”), you must handle this manually to avoid unexpected results.



Details: balance control

Use an exponential amplification factor also for the balance control.

- Moving the joystick to the right decreases the left channel volume.
- Moving the joystick to the left decreases the right channel volume.



Details: mute

It is quite self explanatory

Details: moving average filter

The moving average module must be able to selectively apply a moving average filtering (of a fixed order of 32, set by generic) on the samples.

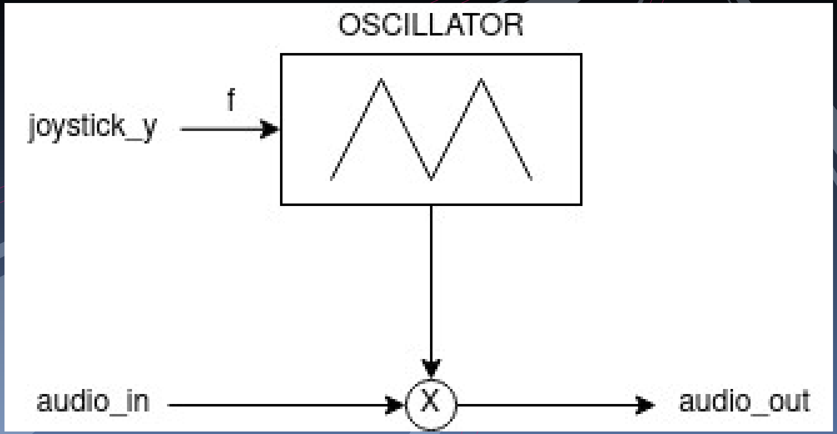
The module should filter the samples when “enable_filter” is high, and should simply pass the samples unmodified when “enable_filter” is low.

Details: LED controller

LED controller: sets the Pmod-JSTK2 LEDs to the correct colors, depending on its inputs (“mute_on” and “filter_on”)

Details: LFO

The LFO (Low Frequency Oscillator) is a triangle-shape modulation on the volume of the incoming data, with its peak at 1 (meaning we don't have a boost on the volume) and minimum at 0.
The frequency of the LFO can be speed up or slowed down by moving the **y-axis** of the Pmod-JSTK2, **if btnU** is pressed. The LFO effect is enabled with the switch **SW0** of the Basys3 board.



Details: LFO

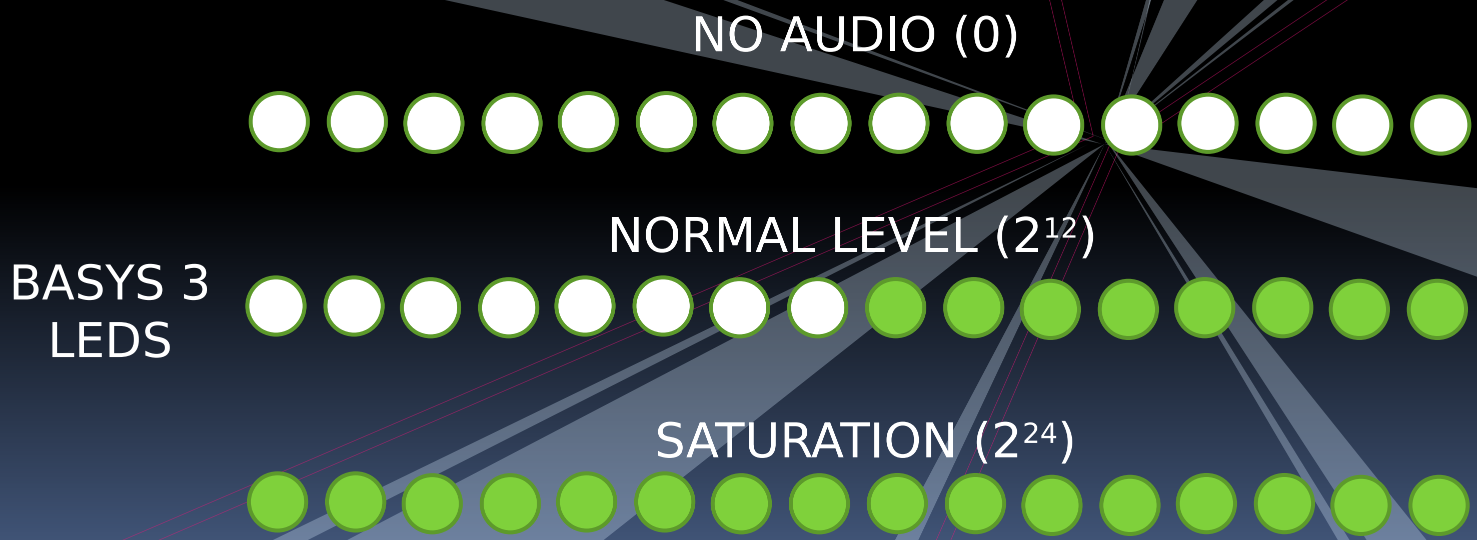
The LFO triangular (up and down) counter, which number of bit depends on the generic TRIANGULAR_COUNTER_LENGTH, has a base step dependent on the input according to the following formula:

LFO_Period := LFO_COUNTER_BASE_PERIOD - ADJUSTMENT_FACTOR*joystick_y

Where LFO_COUNTER_BASE_PERIOD and ADJUSTMENT_FACTOR are two constants.

Details: LED level controller

LED level controller: it turns on the 16 LEDs on the board, depending on the level of the audio at the output (right and left channels are averaged).


NO AUDIO (0)
NORMAL LEVEL (2^{12})
SATURATION (2^{24})
BASYS 3 LEDS