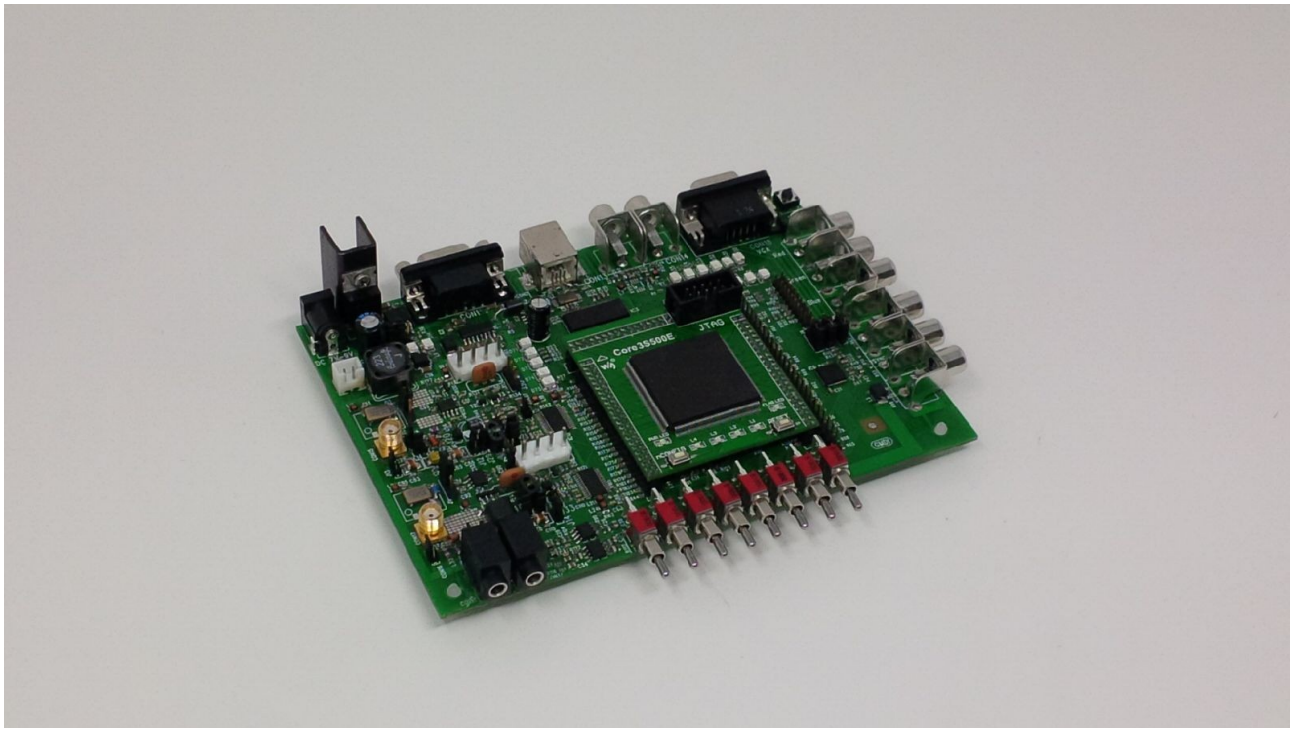
HUBER
SIGNAL PROCESSING

# Users Manual

## SDR-500 Rev. 3
## Software Defined Radio Evaluation System



# Table of Content

# 1 Introduction

The SDR-500 is a Software-Defined-Radio evaluation board with RF frontend. Possible areas of application are:

- Software-Defined-Radio
- Spectral Analysis
- MPX (de-)coder (stereo, video, ... )
- Video Pattern Generator
- Arbitrary Waveform Generator

The following subassemblies are included in the SDR-500:

- FPGA Spartan 3-500 (optional Spartan-6)
- USB controller
- Configurable HF RX/TX frontend with antialiasing filter
- 3-channel 8-bit DAC (e.g. Video RGB)
- Audio I/O - Analog Stereo Codec
- GPIO
- RS-232, USB 2.0 High speed
- Switchpad, LEDs, AUX Output
- RS-232

Figure 1 shows the structure with the partial components.

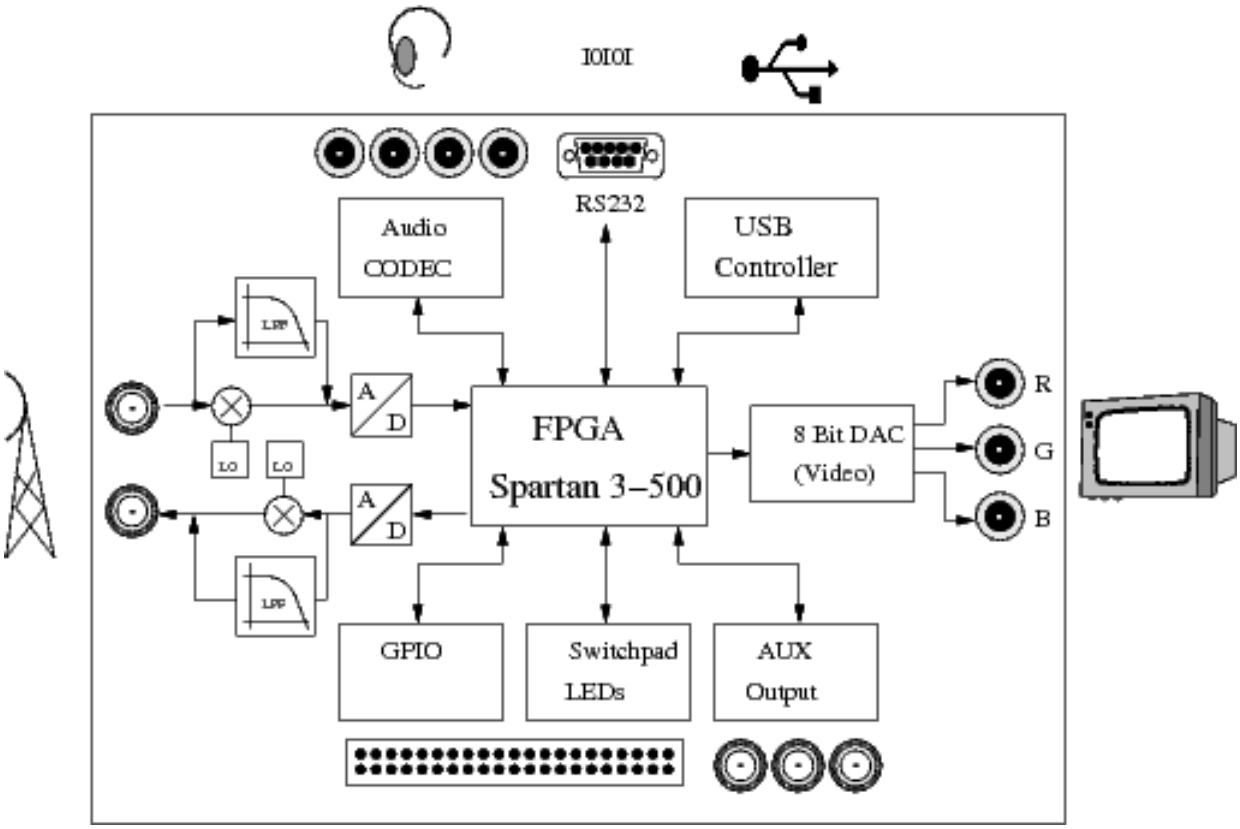



*Figure 1: Block Diagram SDR-500*

# 2 Explanation of the subassemblies

Figure 2 shows the arrangement of the sub-assemblies on the board, as well as the position of the sockets.

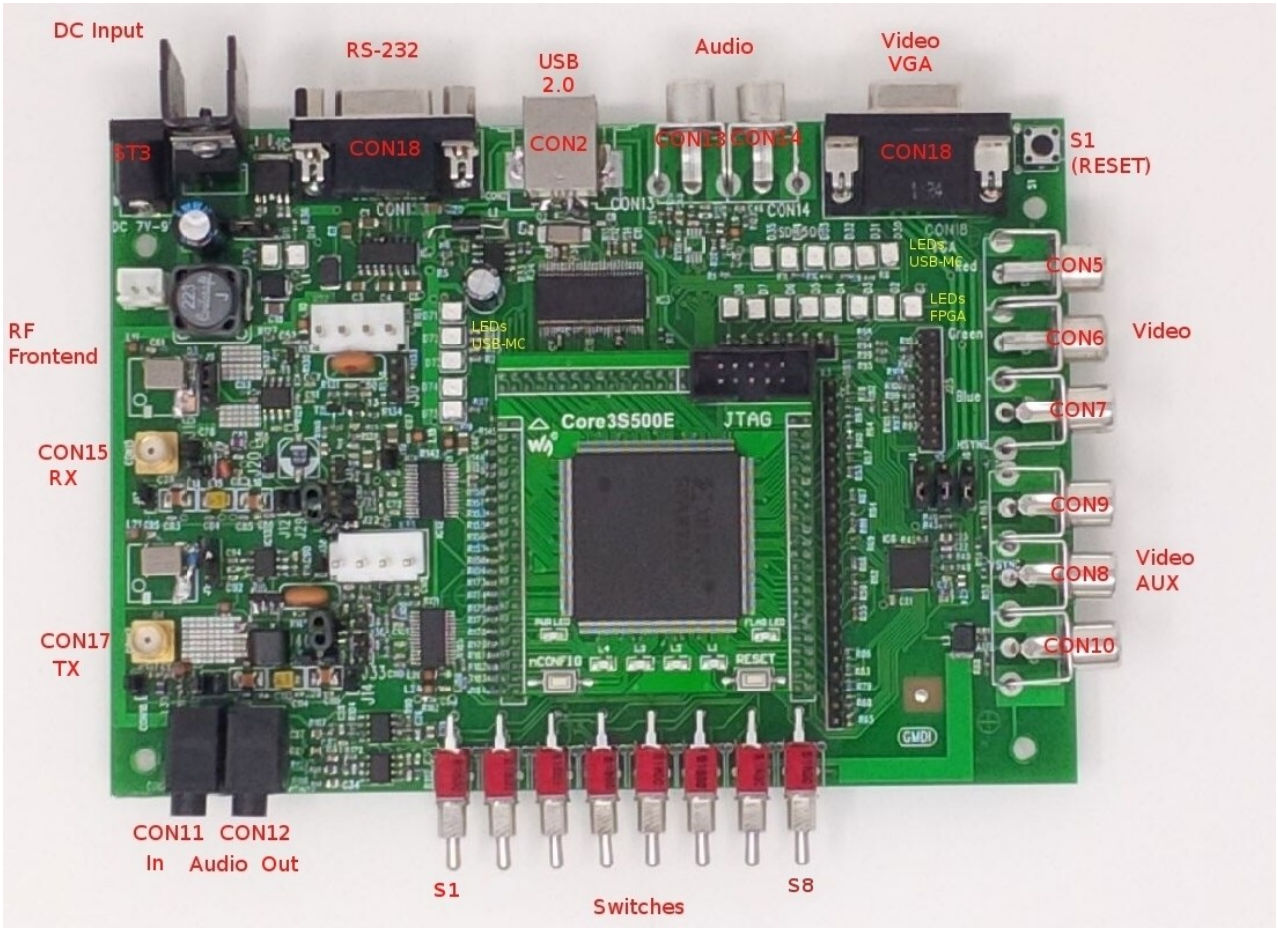



*Figure 2: Arrangement of subassemblies*

The position of the jumpers can be found in the assembly diagram in the appendix.

## *2.1 FPGA*

As FPGA a XILINX Spartan 3S500e is used, which is mounted on a clip-on board. This makes it possible to use future plug-on boards with more powerful FPGAs (e.g. Spartan-6).

The I/O pins of the FPGA are led out to the base board via the pin headers X1, X2 and X4.

The operating voltages 5 V and 3.3 V are provided via the base board.

In addition to the 50 MHz clock (at I/O184) on the plug-in board, a clock signal with 54 MHz frequency can alternatively be used at I/O177 of the FPGA - or fed in externally (through J34, J35).

In addition, there are 4 LEDs, a power LED, the FLAG LED, the RESET button

and the nCONFIG button on the plug-in board.

## 2.2 GPIO

As GPIO certain pins of the FPGA are available at X3 and X5 (outputs only).

Double assignment X5:

| X5 Pin | Signal Video | FPGA Pin |
|---|---|---|
| 1 | undocumented | undocumented |
| 2 | ROT0 | I/O160 |
| 3 | ROT1 | I/O152 |
| 4 | ROT2 | I/O150 |
| 5 | ROT3 | I/O146 |
| 6 | ROT4 | I/O144 |
| 7 | ROT5 | I/O139 |
| 8 | ROT6 | I/O137 |
| 9 | ROT7 | I/O134 |
| 10 | undocumented | undocumented |
| 11 | GRUEN0 | I/O132 |
| 12 | GRUEN1 | I/O128 |
| 13 | GRUEN2 | I/O126 |
| 14 | GRUEN3 | I/O122 |
| 15 | GRUEN4 | I/O119 |
| 16 | GRUEN5 | I/O115 |
| 17 | GRUEN6 | I/O112 |
| 18 | GRUEN7 | I/O108 |
| 19 | BLAU0 | I/O106 |
| 20 | BLAU1 | I/O100 |
| 21 | BLAU2 | I/O98 |
| 22 | BLAU3 | I/O96 |
| 23 | BLAU4 | I/O93 |

The signals at pin header CON21 are also used for the LEDs.

Double assignment CON21:

| X3 Pin | LED | FPGA Pin |
|---|---|---|
| 1 | LED0 | I/O153 |
| 2 | LED1 | I/O151 |
| 3 | LED2 | I/O147 |
| 4 | LED3 | I/O145 |
| 5 | LED4 | I/O140 |
| 6 | LED5 | I/O138 |
| 7 | LED6 | I/O135 |

| | | |
|---|---|---|
| 8 | LED7 | I/O133 |

## 2.3 Switches + LEDs

There are 8 switches and 8 LEDs, which are connected to the FPGA via pins. The following table shows the routing to the FPGA:

| Switch | Signal | FPGA Pin |
|---|---|---|
| S1 | SWTAST0 | I/O123 |
| S2 | SWTAST1 | I/O120 |
| S3 | SWTAST2 | I/O116 |
| S4 | SWTAST3 | I/O113 |
| S5 | SWTAST4 | I/O109 |
| S6 | SWTAST5 | I/O107 |
| S7 | SWTAST6 | I/O102 |
| S8 | SWTAST7 | I/O99 |
| LED0 | LED0 | I/O153 |
| LED1 | LED1 | I/O151 |
| LED2 | LED2 | I/O147 |
| LED3 | LED3 | I/O145 |
| LED4 | LED4 | I/O140 |
| LED5 | LED5 | I/O138 |
| LED6 | LED6 | I/O135 |
| LED7 | LED7 | I/O133 |

## 2.4 AUX I/O High-Speed

The RCA sockets CON8, CON9 and CON10 are I/O pins of the FPGA adapted to 50 Ohm. This allows high-frequency digital signals to be led out on coaxial lines.

Sockets CON10 CON11 are connected to the VGA sync signals.

The routing is as follows:

| Chinch-Buchse | Signal Video | FPGA Pin |
|---|---|---|
| CON10 | HSYNC | I/O36 |
| CON11 | VSYNC | I/O40 |
| CON12 | AUX | I/O129 |

## 2.5 3-fold 8 Bit DAC (Video)

The 3-fold DAC can be used to output VGA signals, among other things.

An ADV7125 from Analog Devices is used as DAC.

The analog RGB signals are available at the VGA socket CON6 and at the RCA sockets CON7 ... CON11 on.

The routing is as follows:

| SUB-D socket Pin | Chinch socket | Signal | FPGA Pin | Type |
|---|---|---|---|---|
| 1 | CON7 | ROT (R) | - | Analogue |
| 2 | CON8 | GRUEN (G) | - | Analogue |
| 3 | CON9 | BLAU (B) | - | Analogue |
| | CON10 | HSYNC | I/O36 | Digital |
| 13 | CON11 | VSYNC | I/O40 | Digital |

## *2.6 USB-Controller*

A Cypress CY7C68013A is used as USB controller. This allows a data transfer according to USB high-speed standard. The transmission to the FPGA is done in parallel via a 16-bit data bus (FD0 ... FD 15) using the following interface signals:

| FPGA Pin | Signal / CY7C68013A |
|---|---|
| I/O4 | FD0 |
| I/O2 | FD1 |
| I/O203 | FD2 |
| I/O200 | FD3 |
| I/O197 | FD4 |
| I/O193 | FD5 |
| I/O190 | FD6 |
| I/O187 | FD7 |
| I/O178 | FD8 |
| I/O172 | FD9 |
| I/O168 | FD10 |
| I/O165 | FD11 |
| I/O164 | FD12 |
| I/O161 | FD13 |
| I/O162 | FD14 |
| I/O163 | FD15 |
| I/O5 | FIFOADR0 |
| I/O3 | FIFOADR1 |
| I/O205 | FLAGA |
| I/O199 | FLAGB |
| I/O202 | FLAGC |
| I/O196 | SLOE |
| I/O192 | SLRD |
| I/O189 | SLWR |

| | |
|---|---|
| I/O185 | IF_CLK |

CON3 is designed as a USB type-B socket.

## 2.7 RS232 / UART

The signals RX/TX at the SUBD-9 socket CON0 are connected to the FPGA via a level converter (MAX232A).

The signals on the FPGA are:

| FPGA Pin | Signal |
|---|---|
| I/O9 | RX0 |
| I/O12 | TX0 |

## 2.8 Audio

The 2-channel delta-sigma codec PCM3060 from Texas Instruments is used as audio interface. It has 24 bit data word width, up to 192 kS/s sampling rate and two input and two output channels each.

The connection at the FPGA is done via an I2S interface for the audio data with the following routing:

| FPGA Pin | Signal |
|---|---|
| I/O25 | DOUT |
| I/O31 | LRCK |
| I/O34 | MCLK |
| I/O29 | SCLK |
| I/O55 | DIN |

The analog signals are low-pass filtered by operational amplifiers and reach the outside via the 3.5 mm jack sockets CON13 (input) and CON14 (output).

## 2.9 RF Frontend

The RF front-end consists of a receive (RX) and a transmit (TX) path that can be operated simultaneously. The signal paths can be configured with jumpers.

The signal conversion from analog to digital is done with IC12, AD9235A or IC14, AD9762B.

Both the RX and TX paths are equipped with separate crystal local oscillators. External LO signals can also be fed in via jumpers.

**RX path:**

The operating modes baseband, subsampling and down-converter are possible.

Baseband operation:

For baseband operation, a low-pass filter is looped in through J22 and J21. Subsequently, a balanced transformer can be optionally looped in with the jumpers J16, J13, J17/J18 for better signal performance at higher frequencies.

Subsampling:

(Optional without bandpass)

J20 and J19 can be used to insert a bandpass filter to select higher-frequency signal bands with subsampling.

Down-converter:

A down converter - consisting of the IC11 SA612A mixer and a bandpass filter - can be looped in via J14 and J13. The IF signal (10.7 MHz) reaches the ADC via an amplifier.

The local oscillator oscillates at a frequency of 80 MHz. The reception frequency is thus 90.7 MHz. Alternatively, an external local oscillator signal can be fed in.

**TX path:**

The operating modes baseband, oversampling and up-converter are possible.

Baseband operation:

For baseband operation, a low-pass filter is looped in through J39, J16. As an option, a balanced transformer can be inserted with the jumpers J41, J31, J36 for better signal performance at higher frequencies.

Oversampling:

(Optional without bandpass)

A bandpass filter can be looped in through J33, J31. The signal is converted to higher-frequency signal bands by means of oversampling.

Up-converter:

An up-converter - consisting of the IC13 SA612A mixer and a bandpass filter - can be looped in via J26. The output signal is converted by 10.7 MHz and the local oscillator into the transmitted signal.

The equipped local oscillator has a frequency of 80 MHz. The transmission frequency is thus 90.7 MHz. Alternatively, a local oscillator signal can also be fed in externally.

# 3 Bring up

You need:

- FPGA development environment Xilinx *ISE Design Suite*
- JTAG Probe (e.g. Xilinx *Platform Cable USB II*)
- RS-232 cable

The SDR-500 is powered by the included power supply. Plug this into a socket

and insert the hollow plug into the CON3 socket. The two LEDs D11 (+5 V, green) and D14 (+3.3 V, green) are used to check the operating voltages.

The FPGA is programmed via the connector marked "JTAG" on the FPGA board using the "XILINX Platform Cable USB" or a compatible JTAG probe. The FPGA is configured with the development environment "XILINX ISE". For the following examples the freely available and free "Web Edition" is sufficient.

# 4 Examples

The supplied examples test and explain the basic functions of the subassemblies of the SDR-500 and serve as a starting point for own projects.

For this purpose load the project "template.xise" into the Xilinx ISE development environment.

The examples are created in graphical input and VHDL. A User Constraint File (UCF) is provided for mapping the FPGA pins. Figure 7 shows the circuit diagram *Top.sch* of the FPGA configuration.

The compiled result can be uploaded to to FPGA using the program *iMPACT*.[1]

Install the free *ISE Design Suite* software from Xilinx (download from www.xilinx.com, search for *ISE*).

For the usage of ISE and iMPACT, please read the corresponding documentation from Xilinx.

The positions of the LEDs, switches, sockets, etc. referenced in the example descriptions are marked in color in Figure 6.

## *4.1 Serial interface*

In this example, a simple, bidirectional communication between the FPGA and an RS232 terminal is implemented.

The function is implemented in the *serial* block.

Description:

Once switch S4 is switched from OFF to ON, the text "SDR-500" is output followed by a tone <BEL> and <CR><LF>.

Entering the characters 0 to 7 in the terminal causes the corresponding LEDs 0 to 7 to light up.

Connect an RS-232 cable between the terminal (or PC with terminal emulation) and the socket CON0 of the SDR-500.

Set the parameters of the terminal to 9600 Baud, 8 Bit, No Parity.

1 To be started via Tools -> iMPACT

Set switch S4 to position *OFF.*

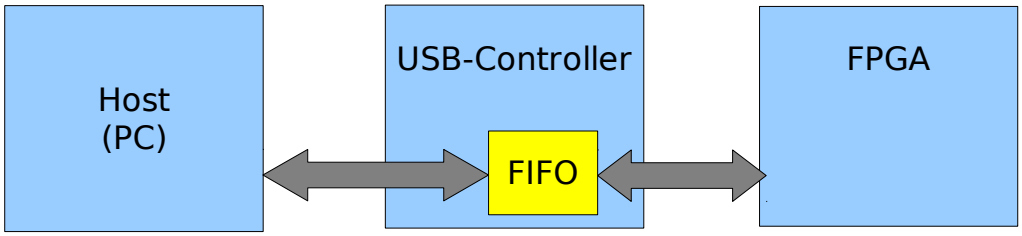
Plug in the SDR-500.

Set switch S4 to position *ON.*

Observe the output in the terminal. The character string "SDR-500" is output. Now enter any number 0 - 7 in the terminal. Observe the illumination of the corresponding LED 1 - 8 on the SDR-500.

## 4.2 USB

The USB example is implemented with the USB controller CY7C68013A and the FPGA. With the USB controller only the FIFO is used. The CPU itself serves only for configuration. The data transfer takes place with USB High-Speed.




Description:

The byte that is written to endpoint 2 by the host (e.g. PC) passes the FIFO and is available to the FPGA at the data bus. The logic in the FPGA reads this byte and outputs the bits to LEDs 0 to 7. On the USB controller runs the program *SlaveFIFO*, which configures the integrated FIFO.

The communication between USB controller and FPGA is done by the data and control signals of the FIFO.

The communication is performed as described in chapter "Slave FIFO Asynchronous Read" - see [DATASHEET_CY], page 47.

The flow control uses the signals

- FLAGA (signaling *empty of* the FIFO to the FPGA)
- SLRD (read request of the FPGA to the FIFO)
- SLOE (request of the FPGA to the FIFO to drive the data bus)

The control of the FIFO in the USB controller outputs these signals to signal the *Empty* state and start or stop the readout requests of the FPGA accordingly.

## 4.2.1 Installation on PC[2]

### *4.2.1.1 USB driver*

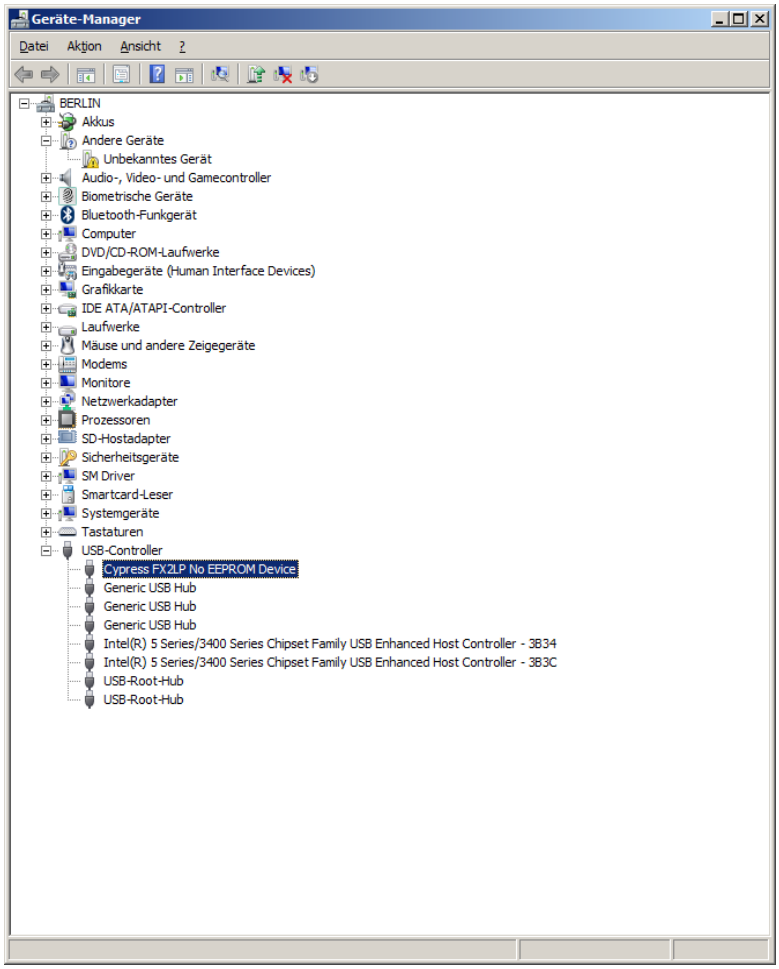
To control USB devices from the PC, the appropriate driver for the USB controller must first be installed. In addition, you will need the *CyConsole* software to control the USB controller and the free development environment *ISE Design Suite* from Xilinx.

To do this, install the *cyusb3* device driver included in the software package supplied. Installation is performed manually in the Device Manager and is explained in the following section:

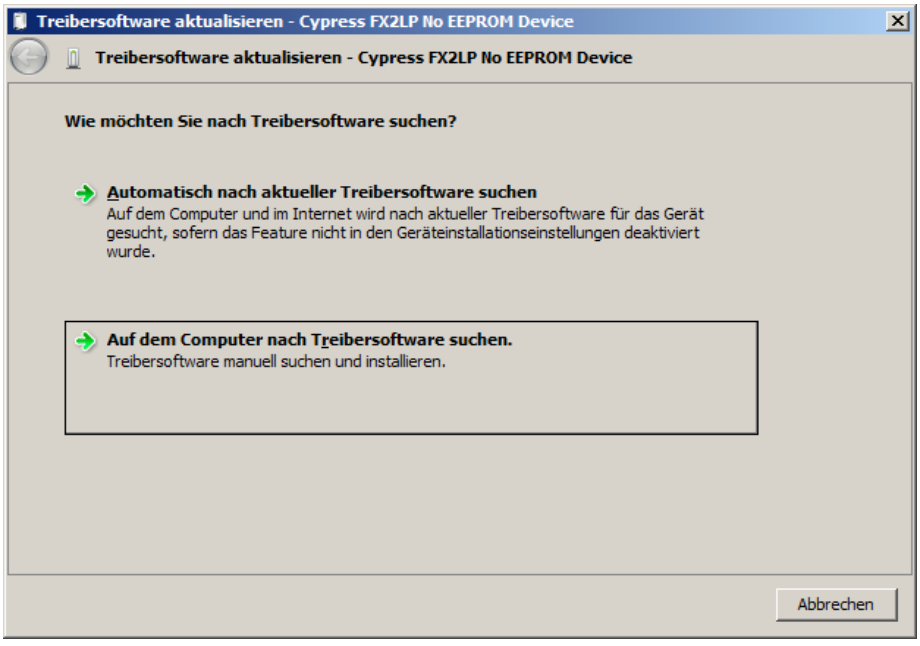
Basically, after RESET, the CY7C68013A USB controller reports to a host with the USB device properties (VID=0x04B4, PID=0x8613, Product=EZ-USB FX2, etc.). If a firmware is loaded into RAM, the USB device properties can be **changed as** desired. The former belongs to the group of registered manufacturers. Therefore, when the USB device is first registered, Windows tries to find a suitable USB driver on the Internet and install it automatically. To install the USB driver with the **changed** USB device properties, the installation must be performed manually.

The driver contains the files *cyusb3.sys*, *cyusb3.inf* and *cyusb.cat,* which are included in the supplied software. Turn on the SDR-500 and let it perform an automatic installation of the USB driver - whether successful or not. In System Preferences, open the Device Manager and identify the new USB device.

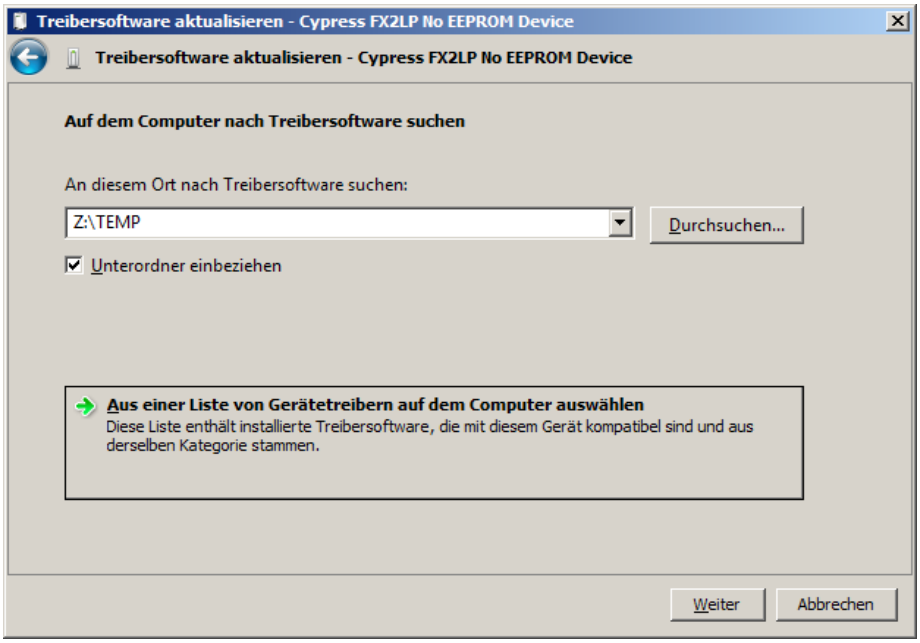
2 The example is executable under Microsoft Windows from version XP. Other operating systems are also supported by Cypress.



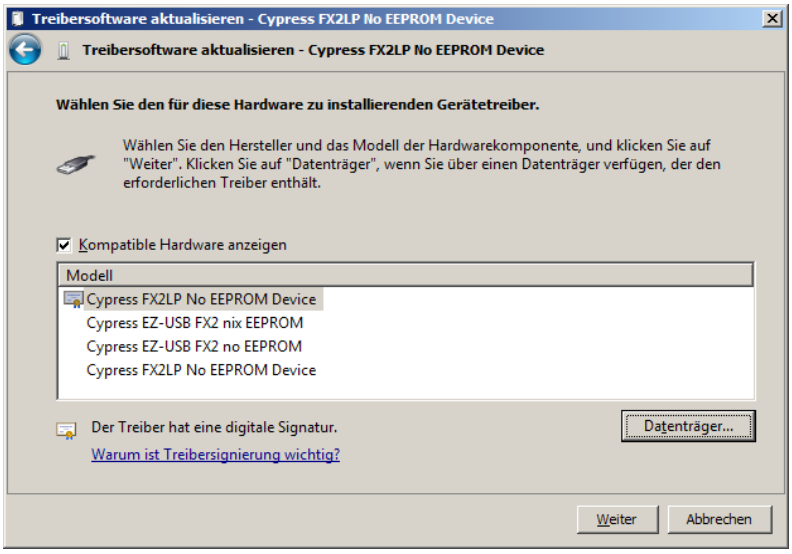

Right-click to open the context menu and select "Update driver software". In the new window select "Search for driver software on the computer".



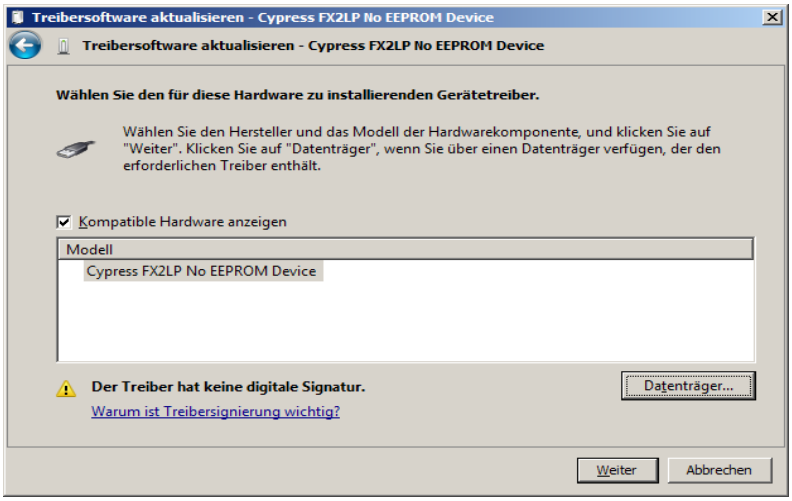
In the next window select "Select from a list of device drivers on the computer".




In the following window press the button "Disk..." .



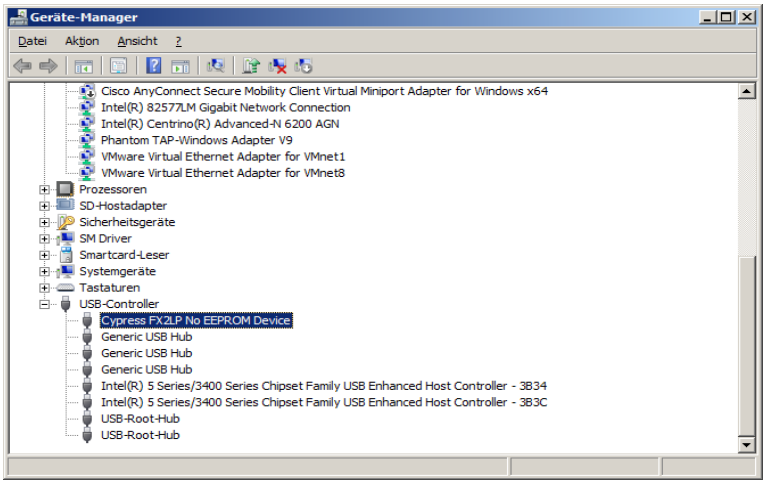

In the selection that opens, search the software package supplied for the directory in which the "cyusb3.*" files modified for this example are located. Select the file "cyusb3.inf" suitable for your operating system version and then press the "OK" button. In the new window select the model "Cypress FX2LP No EEPROM Device" and press the *Next* button.




The driver software is now installed. A warning that the driver is not signed should be acknowledged with "Install driver anyway".[3]

Windows now requires a restart, which you must perform now.

In the device manager the installed USB device must now be displayed without errors.[4]




### *4.2.1.2 USB application and FPGA design software*

The *CyConsole* program from Cypress is used as USB application. Among other things, it allows firmware download to RAM and EEPROM, as well as data transfers. It is included in the supplied software package.

Start the *CyConsole* software (also included) to control the USB controller.

3 Reason: The file cyusb.inf contains the USB device properties VID=0x0547, PID=0x1002, which are not certified. This is only relevant if the system is sold publicly as a device. This is irrelevant for the examples.

4 An exclamation mark on the symbol indicates an error. The device driver will not work in this case. In this case repeat the driver installation and exactly as described before.

Start Xilinx ISE.

### 4.2.2 Programming the firmware of the USB controller

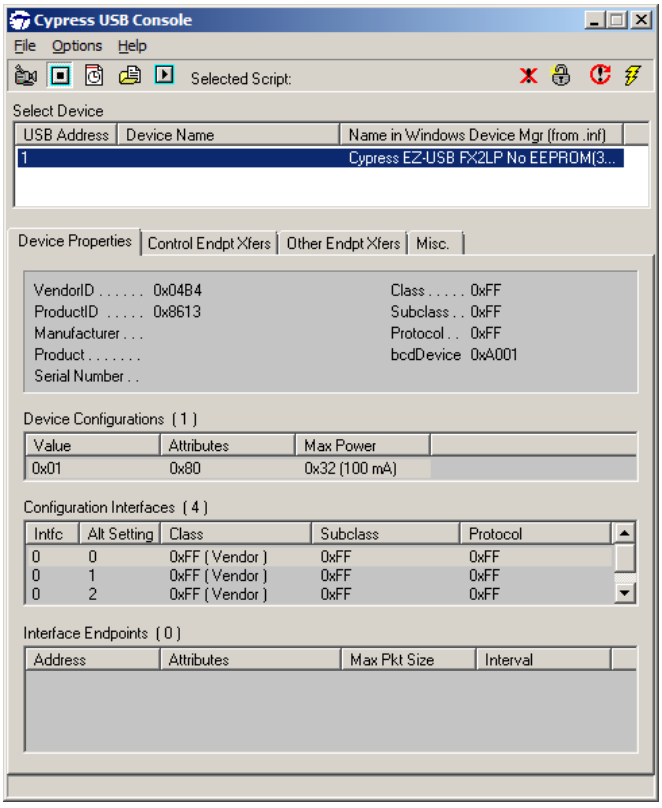
The USB controller CY7C68013A contains a RAM where the firmware must be loaded. The data is volatile, i.e. after RESET all data is lost and the firmware must be reloaded. Alternatively it can be stored in the EEPROM on the board. The USB controller recognizes a valid program in the EEPROM and boots it after RESET.

The firmware performs an initialization of the FIFO.

The firmware has been created with a software project under the development environment *KEIL C51* and can be changed or extended as required. For this example only the binary file Slave_FIFO.hex has to be loaded into the USB controller.

*Start the CyConsole program*

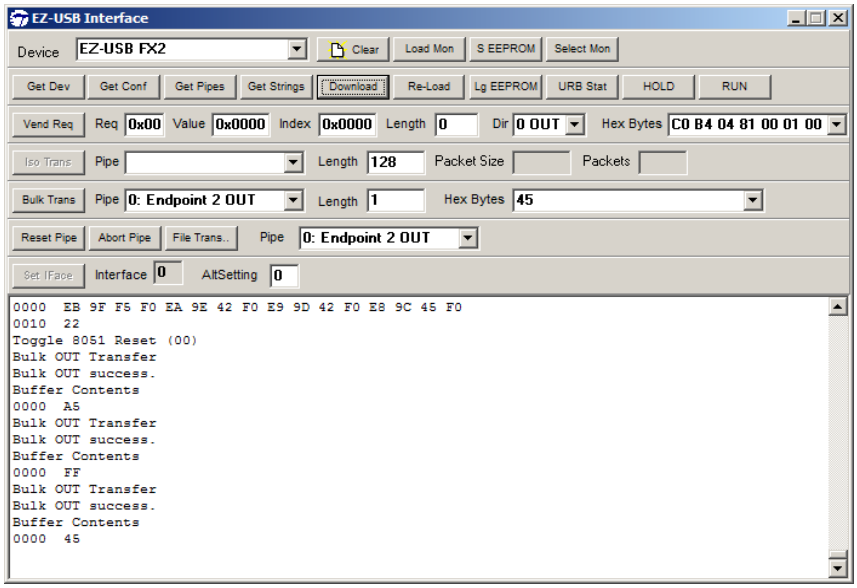
Connect the SDR-500 to the PC with the supplied USB cable and switch on the SDR-500




The data in the tab *Device Properties* (VendorID, ProductID, Manufacturer, ... ) must appear as shown in the picture.

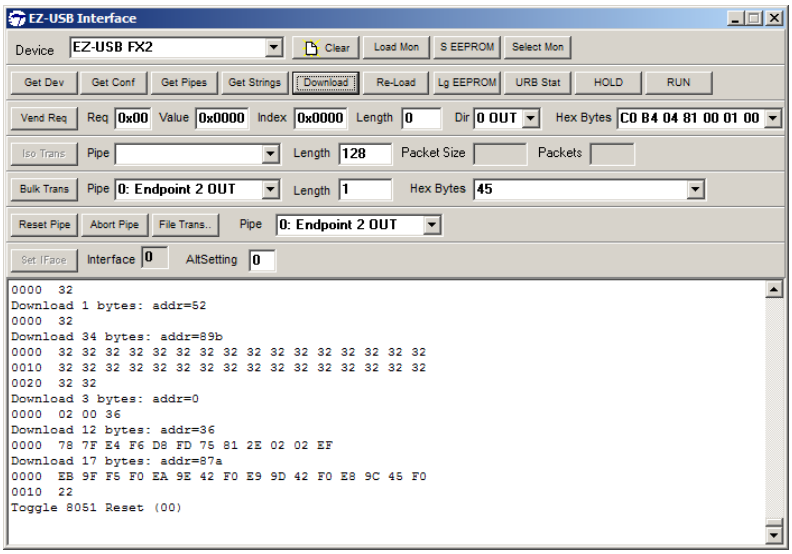
*In CyConsole, click on Options -> EZ-USB Interface*

The window shown opens.

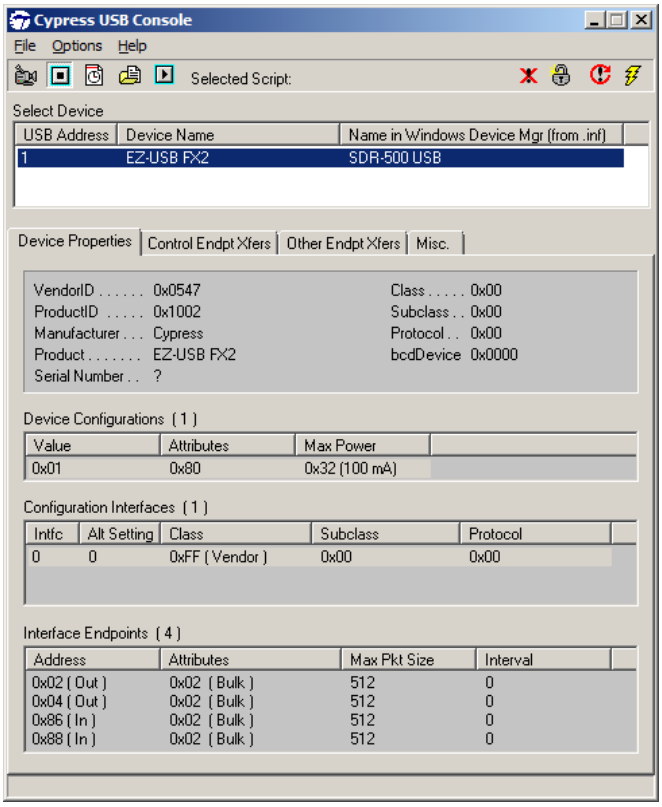



Press the *Download* button and select the file *Slave-FIFO.hex*. To program the EEPROM, press *Lg EEPROM* and select the file *Slave-FIFO.iic*.

The program data should be output in the output field. The window will look like this:




The loaded program is now running in the USB controller. A new registration of the USB device with new device parameters is now performed. In the device manager the name changes to "SDR-500", as shown in the following picture.




### 4.2.3 Load FPGA configuration

In the example the following unit is configured:

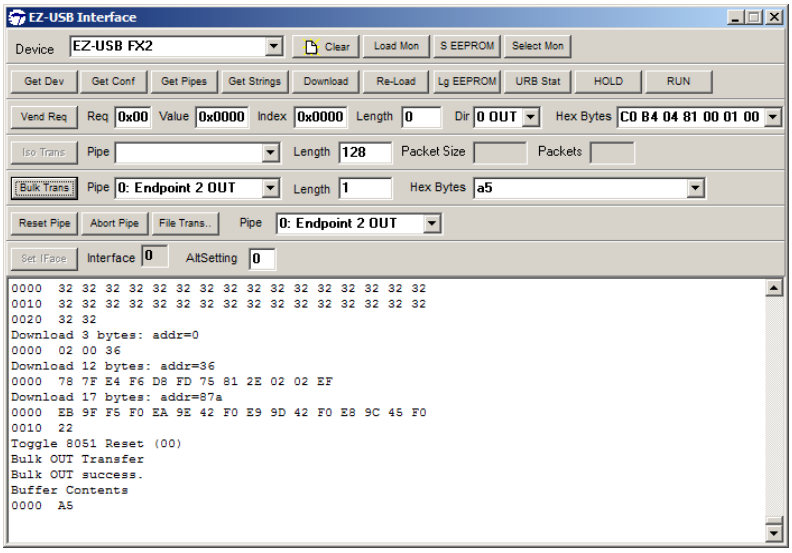
The function is implemented in the block *USB_LEDdemo.*

Input *FLAGA* signals that the FIFO of the USB controller is empty. If the host writes a byte to the USB controller, *FLAGA* becomes inactive. A process in the FPGA then reads the FIFO by setting the FIFO address (*FIFO-ADR*) to 0b00 and activating *SLOE* and *SLRD.* The byte written by the host is now present on the *FIFODATA* data bus and the LSBs 3:0 are output to the LEDs (*led*). After resetting the control signals *SLRD* and *SLOE FLAGA* is set again and the host can write a new byte.

### 4.2.4 Test

Start Xilinx *ISE Design Suite* and load the FPGA project

Now transfer a byte in the *CyConsole* and check the LEDs at the SDR-500. Set *Pipe 0: Endpoint 2 OUT, Length 1 Hex Bytes 01 (or 02, 03, 04 or 05, respectively)* in the line next to the *Bulk-Trans* button and press the *Bulk-Trans* button. In the text window in the lower area the transmitted byte must be shown as follows:




The LEDs on the FPGA sub-board L1 ... L4 must light up according this table.

| Hex Bytes | LED Status |
|---|---|
| 01 | L1 lights up |
| 02 | L2 lights up |
| 03 | L3 lights up |
| 04 | L4 lights up |
| 05 | L1 ... L4 light up in sequence |

## *4.3 VGA*

This example generates a video signal which is output via the video DAC at the VGA socket CON6. The test pattern can be viewed on a PC monitor or television with a VGA socket. The following variants are available as test patterns:

- Oscilloscope
- Colour bars

The function is implemented in the blocks *oszi* and the module marked *Audio_Source.*

Switch S7 determines the function of the oscilloscope (OFF) or color bar (ON).

Switch S2 determines the signal source for the oscilloscope. There is a choice between an internal sine wave generator and the input socket *CON13*.

The trigger of the oscilloscope is permanently set to positive zero crossing.

Connect a VGA cable between the CON6 socket and a monitor.

Set switch S7 to position ON.

Plug in the SDR-500.

The color bar test pattern appears on the monitor.

Set switch S7 to position OFF.

Set switch S2 to position ON.

The monitor image now changes to the oscilloscope display. The internal sinus signal is displayed.

Connect a signal source to *CON13*.

Set switch S2 to position OFF.

The display now changes to the signal at socket *CON13*.

## *4.4 Audio*

This example generates a duotone signal with 1 kHz and 2 kHz and routes it or optionally the input signal at socket *CON13* to the output socket *CON14*.

The function is implemented in the module designated with *Audio_Source.*

Switch S1 switches between the internal signal generator and the input signal (loop through).

Switch S2 switches the internal sine wave generator or the supplied audio signal to the output.

Connect an audio playback device (e.g. active loudspeakers, headphones, PC) to the output socket *CON12*.

Set switch S1 to position OFF.

Set switch S2 to position ON.

Plug in the SDR-500.

You hear a duotone signal with 1 kHz in the left - and 2 kHz in the right channel.

Connect an audio cable between an audio source and the *CON11* input jack.

Set switch S1 to position OFF.

Set switch S2 to position OFF.

Plug in the SDR-500.

You hear the unchanged signal of the audio source.

## *4.5 RF RX/TX*

This example implements an FM transmitter in the FM frequency range. In the FPGA a carrier signal is FM-modulated with an audio signal and output at the ADC. The high-frequency signal is present at the TX socket *CON15.*

FPGA part:

The function is implemented in the *FM modulator* and *audio source* modules and is fed to the DAC output via a multiplexer.

The multiplexer is located at the output and can switch between the FM modulated signal, a 1 MHz sine signal and the input signal from the ADC (loop through).

The FM modulator modulates the audio signal to a carrier frequency of 10.7 MHz.

The input signal of the modulator can be a 1 kHz sine wave signal or the signal from the audio signal source.

Switch S2 selects between the internal sine wave generator or the externally fed audio signal as source.

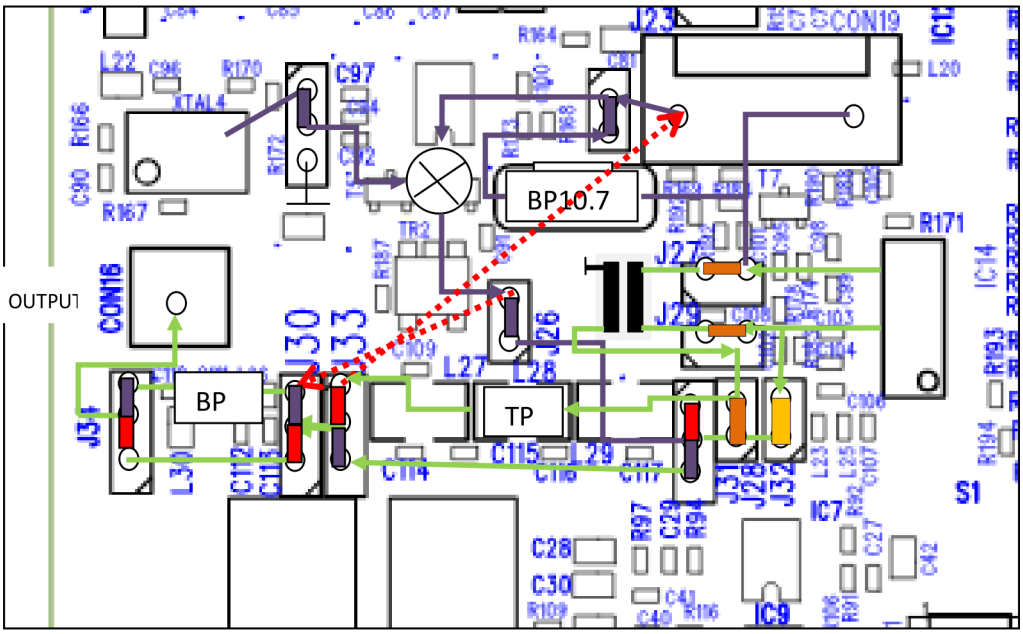
Hardware part:

After the D/A conversion by IC14, the signal goes to the up-converter, which mixes the signal with the LO frequency from 80 MHz to 90.7 MHz. The bandpass filtered RF signal is fed out at socket *CON17.*

Configure the following jumper:

J24 --> closed
J26 --> closed
J25 --> 1-2 connected
J28, J32 --> open
J31, J33 --> 2-3 connected
J30, J34 --> 1-2 connected

The following graph shows a better orientation at the PCB. Connect the jumper marked in magenta colored ▮ .




Connect an audio playback device (e.g. active loudspeaker, PC) to the output socket *CON12*.

Set switch S2 to position ON.

Connect a receiving device (e.g. radio receiver, spectrum analyzer or antenna) to the HF TX socket *CON17.* Switch this to FM and select 90.7 MHz as the center frequency.

Plug in the SDR-500.

You hear a signal with 1 kHz in the receiver.

Set switch S2 to position OFF.

You will hear the signal of the audio signal source fed into the receiver.

# Annex

## *Circuit diagram*

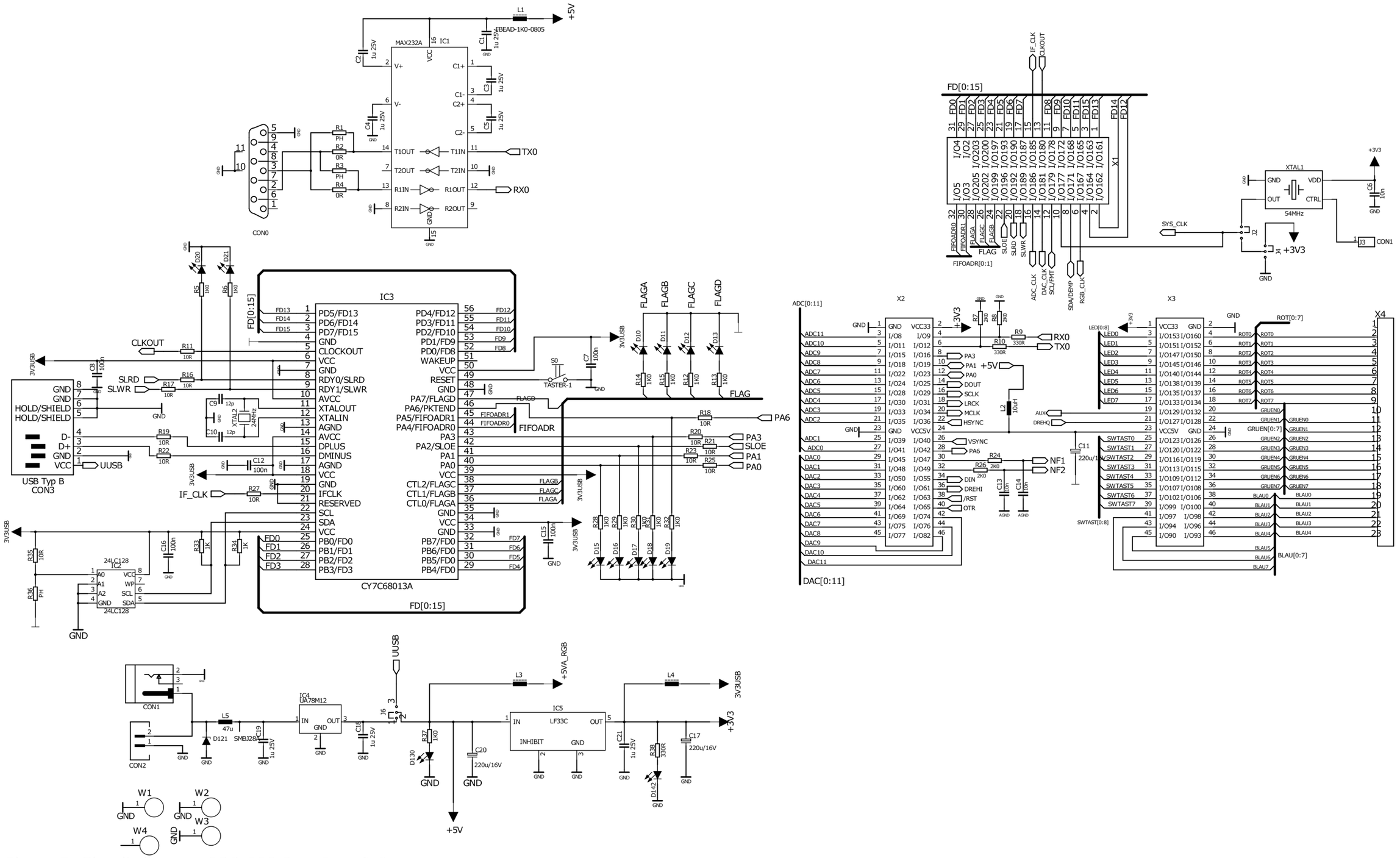



*Figure 3: Circuit diagram SDR-500 Rev. 3, p. 1/3*

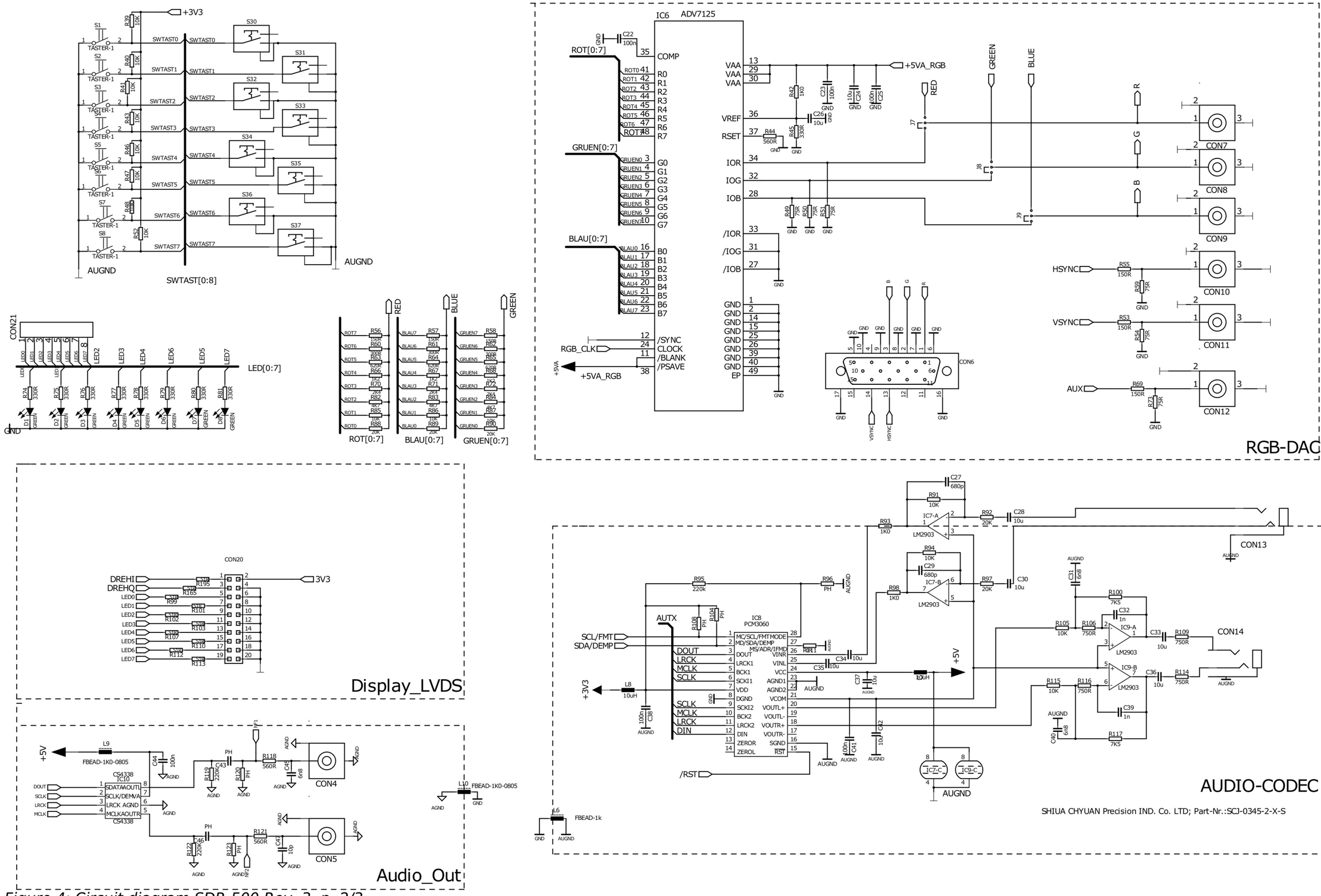



*Figure 4: Circuit diagram SDR-500 Rev. 3, p. 2/3*

*Figure 5: Circuit diagram SDR-500 Rev. 3, p. 3/3*

***Mounting print***

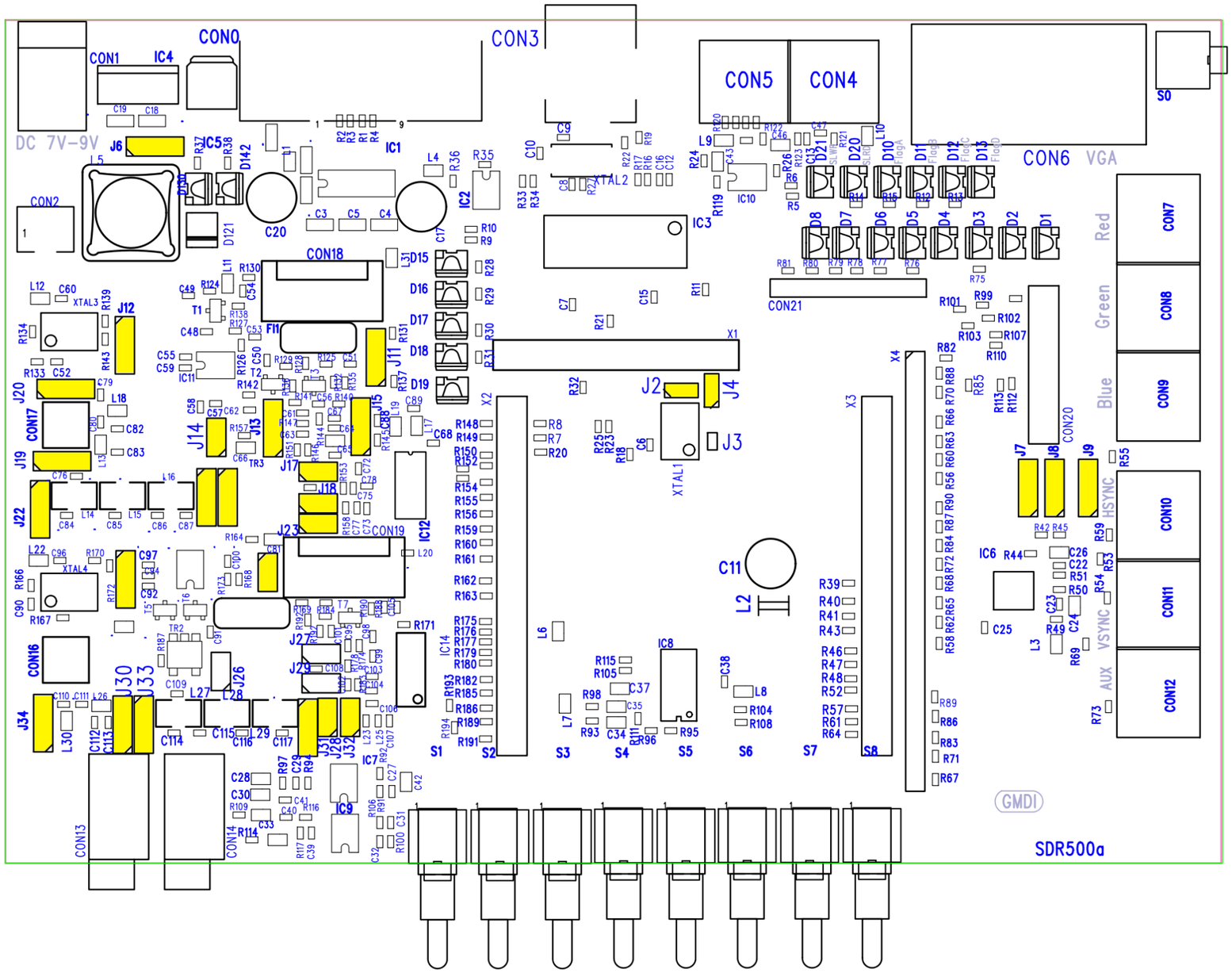



*Figure 6: Mounting print SDR-500 Rev. 3*

# *FPGA configuration diagram*

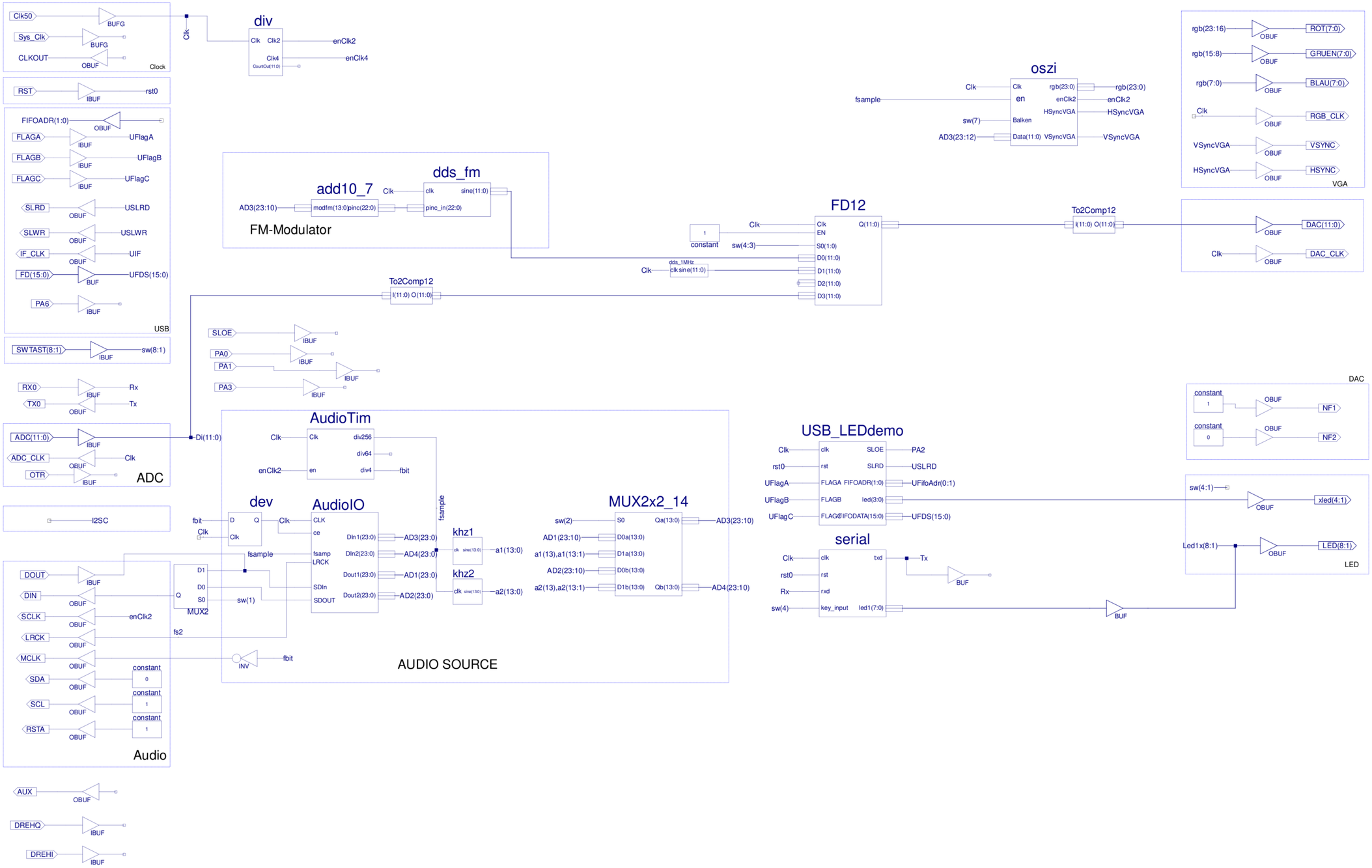



*Figure 7: FPGA configuration diagram*

# List of figures

# Bibliography

[DATASHEET_CY] Data sheet
CY7C68013A, CY7C68014A, CY7C68015A, CY7C68016A, EZ-USB® FX2LPTM USB Microcontroller High-Speed USB Peripheral Controller
Rev. *Z, April 26, 2017
Document Number: 38-08032
Cypress Semiconductor Corporation

HUBER SIGNAL PROCESSING

*... Made in Germay*

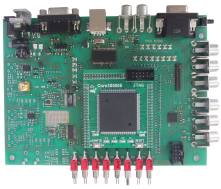
Signal Generators

Probes

Precision amplifiers

Measuring range extensions

DSP systems



HUBER SIGNAL PROCESSING
Unternbergstr. 13
83278 Traunstein
Germany

Phone: +49 (0)861.213978-0
Fax: +49 (0)861.213978-40

sales@huber-signal.com
www.huber-signal.com

Doc. Nr.: 0126-002
v1.1, 06.10.2020, subject to changes