Automotive Racing Video Data Logger

ECE445 Fall 2012

Design Review

TEAM 5: Andrew Wesly, Nick Greenway, Tung Do

TA: Igor Fedorov

October 1st, 2012

Contents

I. Introduction

1) Statement of Purpose:

The project goal is to develop a data logger video overlay that can be used in a car. The final product should be able to acquire near real-time engine data through the On-Board Diagnostic Port (OBDII) installed in a car. This product will also take video of a ride through the car's windshield with user supplied standard definition camera. Next, it will overlay OBD data onto the captured video and store it in a portable medium. The video stored in the media should be in a computer readable video format like JPEG2000. Raw OBDII data should also be stored as a file in the storage media. The driver can select playback video on the spot with overlay or replay the video with data at a time later.

This product will aid in automotive racing and tuning by being more intuitive for driver's to use and have better functions than existing products on the market.

2) Benefits and Features:

Currently, products like Dashware create a data video overlay AFTER a user uploads their video and data to a home computer. This feature is not suitable for amateur racers because they may need to review video and data immediately after a race. Also, this style of operation makes system set-up difficult because they cannot test their logging systems in real-time.

Our logger solves this problem with the abilities to:

- Record video and create an overlay in real-time and
- Use logger hardware to playback captured video.
- Capture standalone OBDII data

According to a few internet reviews, some users prefer to have a data logger record video with OBDII data overlayed on the screen, whereas other users prefer to have raw video recorded and race data recorded separately. Currently, there are products like Track vision that record the raw data onto an overlay but do not allow for separate video/OBDII streams.

Our product solves this problem by allowing the user to manually select a recording mode. They can pick either OBD II on video overlay raw video recorded with an associated OBDII data file for later analysis.

We will also implement an intuitive interface for drivers to use with mechanical switches. Interface with the data logger should be intuitive for fast set-up times and can be done by feel. Some products on the market require a computer to set-up the logger, which is not conducive for race conditions when time between race events is critical for success.

II. Design

2.1 Block Diagrams

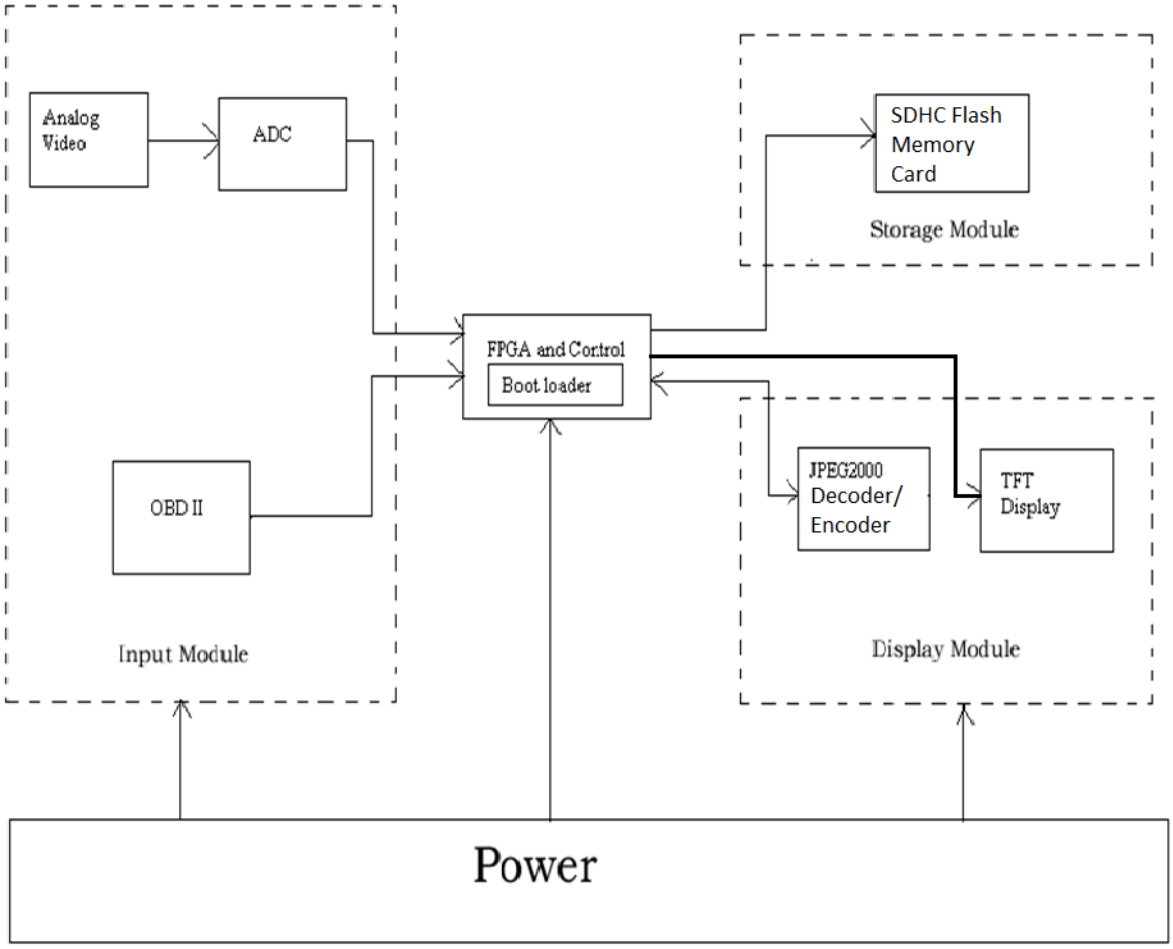


Figure 1: Top Level Block diagram

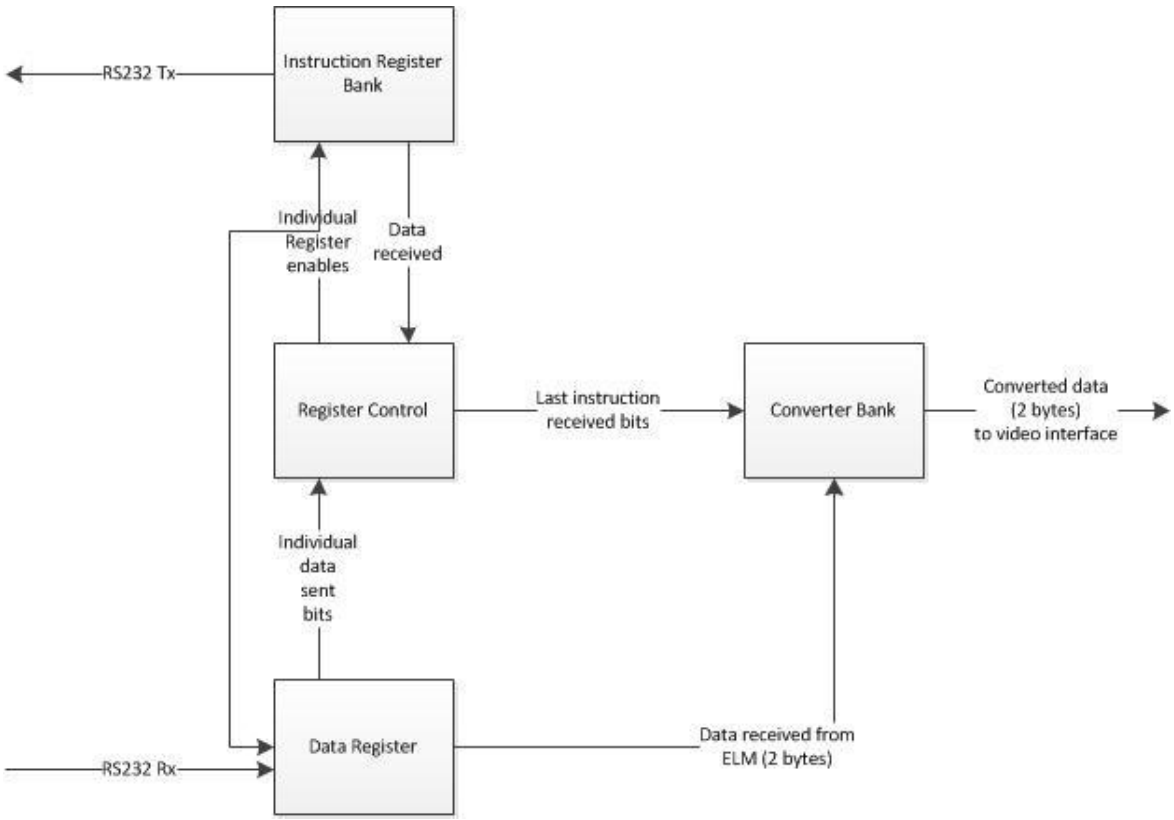


Figure 2: OBD II Block diagram

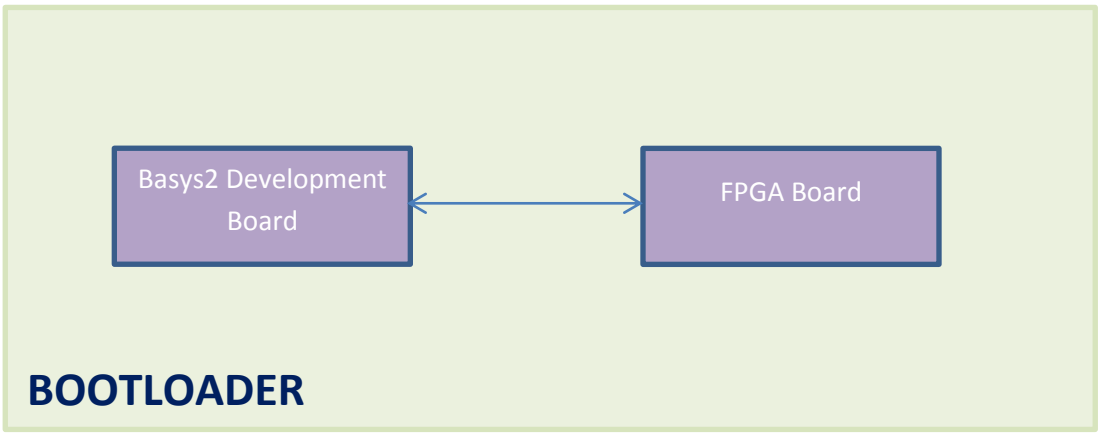


Figure 3: Bootloader Block diagram

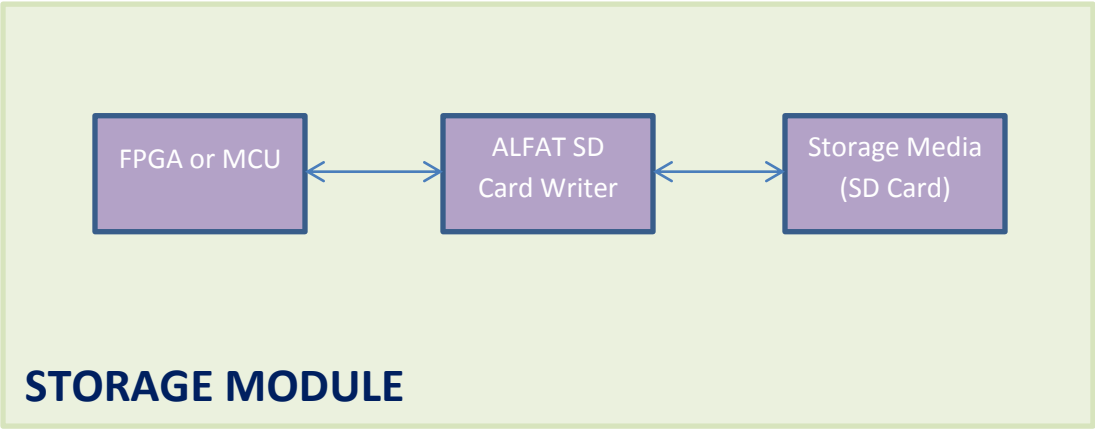


Figure 4: Storage Module Block diagram

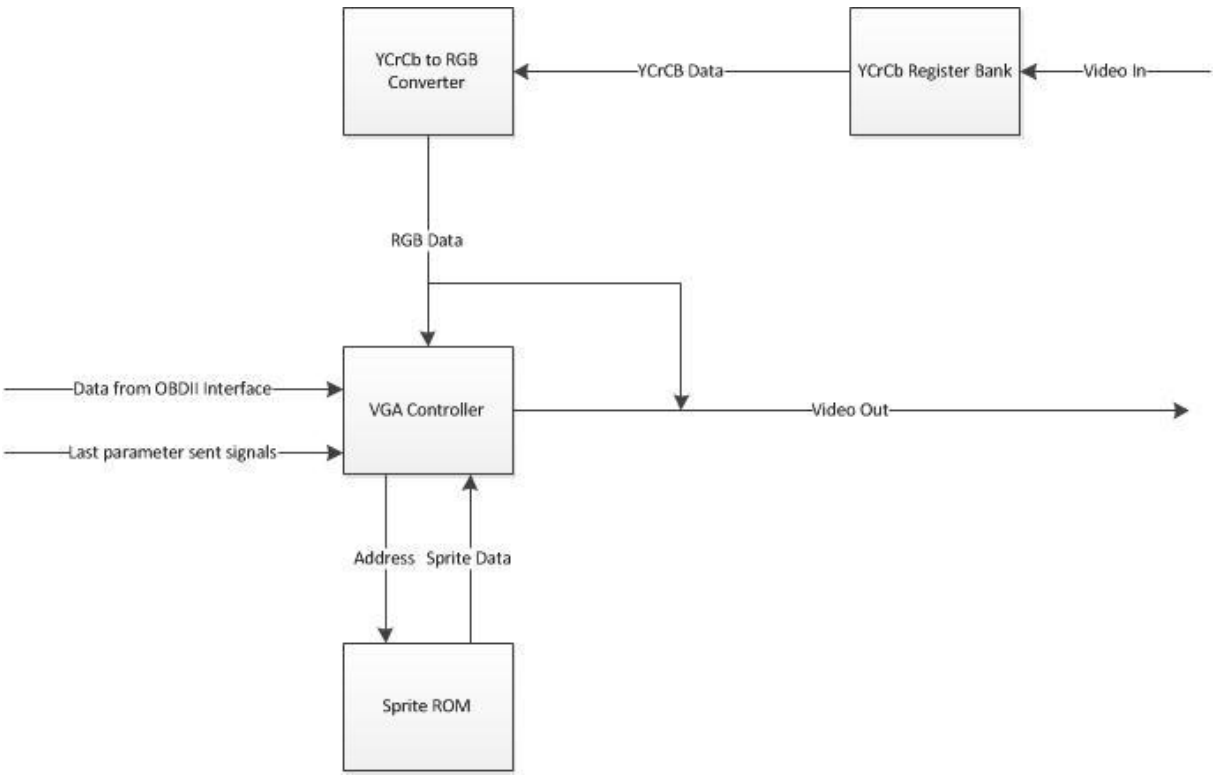


Figure 5: Video Display Module Block diagram

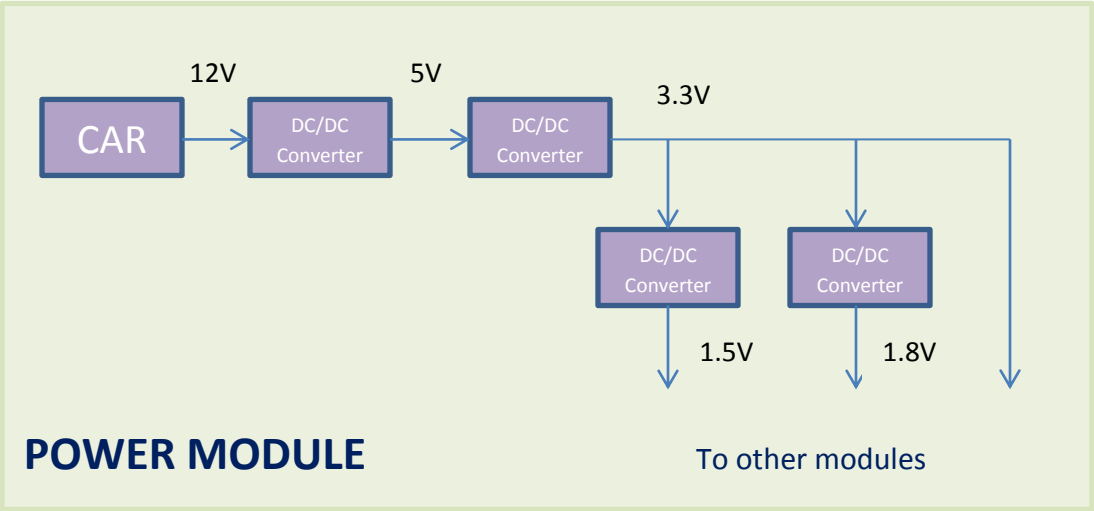


Figure 6: Power Module Block diagram

2.2 Block Descriptions

2.2.1 Overall Summary:

1) **Input Data Module:** This module obtains input data such as information from the OBD II and the video images from the camera. The module then converts them into digital form using an ADC and transmits this digital data to the FPGA.
    a. **Analog Video:** The analog video obtained from the camera on the windshield. The image will be sent to the ADC to be converted into digital form.
    b. **ADC:** The Analog-to-Digital converter that converts standard definition analog video input into a digital form so it can be used by the FPGA. ADC input is analog video, which is converted to digital images and are then routed to the FPGA.
    c. **OBD II:** The On-Board Diagnostic system on a car is where information such as the MIL (malfunction indicator light), DTC (diagnostic trouble code), I/M (inspection and maintenance) info, etc. can be extracted from. We will focus on polling operating data like engine RPM, temperature, etc. The information obtained from this block is sent to the FPGA.
2) **FPGA and Control Module:** This module is the "brain" of the entire circuit. It receives the OBD II data and video images from the input module, processes them, overlays the data onto the video stream. It then routes the overlaid video to the storage and display modules. This entity also manages video stream compression/decompression. This block also boots the system.
    a. **Boot loader:** The boot loader is a nonvolatile memory circuit and co-processor that configures the FPGA for use when system is powered on.
    b. **JPEG2000 Decoder/Encoder:** A device that compresses/decompresses digital video for extended storage and computer access. This block receives processed video and data from the FPGA and routes decompressed output back to the FPGA so that video can be routed to other entities.
    c. **FPGA**: The integrated circuit that will be used to design the digital system to process the images and data obtained from the OBD II. This block obtains video from the camera and data from the OBD II, process them, and routes signal through a video compressor/decompressor. FPGA also streams processed to a storage device (the MMC) and controls data display for the attached TFT screen.
3) **Storage Module:** This module receives the overlaid video from the FPGA and control module and stores it on a storage device for later use. A compact flash MMC may be used as the storage device. User can also replay video from the storage medium.
    a. **SDHC Flash Memory Card**: The flash memory data storage device where the video and data obtained will be stored. This block obtains the processed video and data from the FPGA.
    b. **USB**: It is possible to achieve higher storage speeds with an FUSB PHY chip over USB interface. This item may be used depending on time remaining for project completion.
4) **Display Module:** This module displays the overlaid video on a LCD screen from storage through the FPGA/control block.
    a. **TFT Display:** The LCD display where the processed video obtained from the FPGA will be displayed.
5) **Power Module:** This module supplies power to the FPGA and Control Module.

2.2.2 Input Module

OBD II Interface:

Register Control
Inputs: There are individual signals from each register specifying the last instruction that was sent or that the last requested data was received. Only one of these signals is ever high at any given time. There is also a reset input to restart and initialize the entity.

Outputs: Each instruction register and the data register have individual enables, only one of which is high at any given time. Also, enables are sent to the converter entities depending on the latest data that was received from the OBDII.

Description: This entity consists of a state machine that performs a loop that first initializes the ELM327 chip and then loops through a series of states that send and then received data from the ELM327. The states go from transmitting data, to receiving data, then back to sending, etc. First, an enable is sent to one of the instruction registers to allow a certain instruction to be sent to the OBDII. Then, once a signal is received from the register indicating that the data has been sent, the data register is enabled so as to receive the requested information from the OBDII. Once this data is received, a signal from the register is made high and sent to the register control, indicating that the data was received properly. Then the control moves to the next instruction and enables the appropriate register, but also sends a signal to the converter bank and the video interface indicating what the last received information was, and enabling the correct conversion entity depending on that data. Note that only one register enable and one converter enable are ever high at any given time to ensure that only one instruction is sent or received at a time and that only one parameter is being converted at a time to ensure that no data gets garbled and to ensure that all the data is sent and received in a clear order.

Instruction Register Bank
Inputs: Each register has an enable sent from the register control, a clock based on the baud rate, and a reset for initialization purposes.

Outputs: Data is outputted and transmitted via RS-232 UART. Also, a signal from each individual register is sent to the register control, which is low unless a complete instruction has been sent.

Description: When a register is first initialized, a 4 or 6 byte data signal is set, depending on the instruction being sent. If the enable is active, then the most significant bit of the data signal is outputted. Then, the signal is shifted to the left by one bit, and a counter increases by one. This continues until the counter reaches the number of bits in the instruction. Then, the signal going to the control register becomes active, indicating that the message has been sent. This causes the state in the control register to change, causing the enable to go low and the register to reinitialize to its original value.

Instructions we plan to use:
x0105: Engine coolant temperature
x010A: Fuel pressure
x010C: Engine RPM
x010D: Vehicle speed

x010F: Intake air temperature
x0111: Throttle position
x0146: Ambient air temperature
x015C: Engine oil temperature

Note: These may be changed depending on parameters available for the car we test. The one we use may not be compatible with some of the instructions listed.

Data Register

Inputs: One of the inputs in the data being received from the ELM327 via RS-232 UART. Also, there are two enables for this register; one indicating that the data being received is one byte long, the other indicating that it is two bytes long. There is also a reset for initialization and a clock signal based on the baud rate.
Outputs: The data received from the OBDII is sent to the converter bank, and there is also a signal sent to the register control that is high when a complete message has been received.
Description: When enabled, this register receives data from the OBDII one bit at a time. When a bit is received, the register shifts the bits to "make room" for the next one. Also, a counter is used to keep track of the amount of data received. Once all of the data is received, a signal is sent to the register control to indicate such. Notice that the counter counts to different values depending on whether one or two bytes are being received.

Converter Bank

Inputs: The data received from the OBDII is the primary input. This is one or two bytes depending on the information received. Also, each converter entity has an enable and a reset.
Outputs: Two bytes of data are outputted to the video interface. This data is the actual/converter value of the data received.
Description: When enabled, the converter uses an arithmetic equation to change the data received into the actual, factual information from the engine. The equations used are explained in the calculations section. These new values are then sent to the video interface. Note that if the converter is not enabled, its output is set to high impedance, so that only one output is being transmitted at any time.

2.2.3 FPGA and Control Module

a. Boot loader

According to the Cyclone II Device Handbook, Cyclone II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, all configuration data will be lost once the device powers off. Therefore, there's a need to implement a boot loader to download configuration data to the Cyclone II devices every time the device powers up.
In our design, we'll use the Digilent Basys2 development board as the FPGA's boot loader.
The Cyclone II (or the EP2C35 processor) supports 3 configuration schemes: AS (active serial), PS (passive serial) and JTAG-based configuration. To select a configuration scheme, the MSEL [ ] must be connected to the appropriate digital logic.

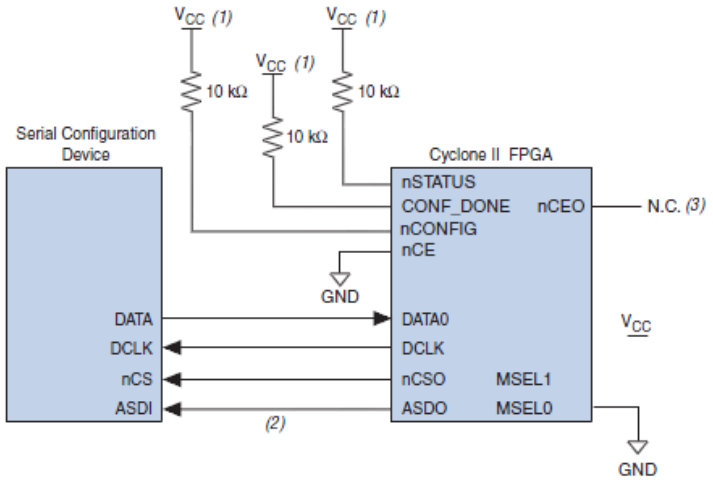
Table 13–1. Cyclone II Configuration Schemes

Configuration Scheme	MSEL1	MSEL0
AS (20 MHz)	0	0
PS	0	1
Fast AS (40 MHz) *(1)*	1	0
JTAG-based Configuration *(2)*	*(3)*	*(3)*

Figure 7: Cyclone II Configuration Schemes (Cyclone II Device Handbook, Volume 1, p. 356)

For our design, the AS scheme will be used, since it has the ability to configure the device at a high frequency (40MHz).

Figure 13–3. Single Device AS Configuration



Notes to Figure 13–3:
(1) Connect the pull-up resistors to a 3.3-V supply.
(2) Cyclone II devices use the ASDO to ASDI path to control the configuration device.
(3) The nCEO pin can be left unconnected or used as a user I/O pin when it does not feed another device's nCE pin.

Figure 8: Single Device AS Configuration (Cyclone II Device Handbook, Volume 1, p. 362)

The EP2C35 essentially goes through 3 states (RESET, CONFIGURATION and INITIALIZATION) before USER MODE, as depicted below,

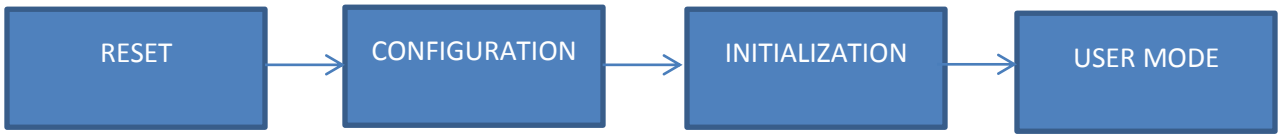
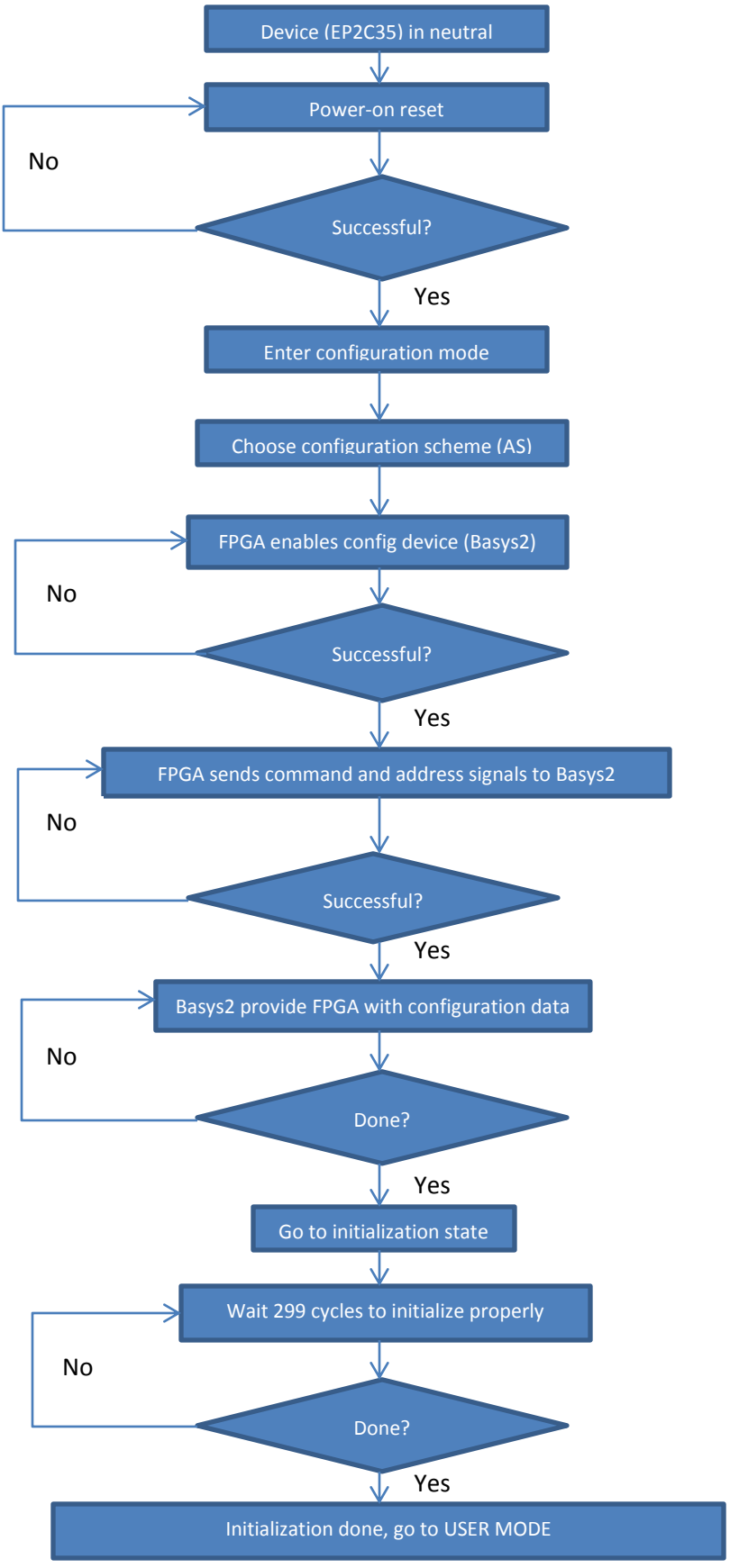


Figure 9: Cyclone II device configuration states

The EP2C35configuration flowchart is as follows.



2.2.4 Storage Module

After the video is processed with the overlaid data, it will be stored on a storage device. We'll choose the Transcend 16 GB SDHC Class 10 Flash Memory Card as our storage device. To simplify the process of writing the file to disk (which involves writing complex memory card and USB drivers and manipulate buggy file system), we'll use the ALFAT OEM Board – FAT32 SD card writer from GHI Electronics.

Figure 10: ALFAT OEM Board (ALFAT SoC Processor User Manual, p. 45)

The ALFAT SoC processor gives the FPGA a simple way to access storage medias such as SD cards and USB Mass Storage devices in a very short time. Its key features are as follows:

- LFN (Long File Name) supports.
- FAT16 and FAT32 systems.
- Friendly user-interface through UART, SPI, I2C.
- No limits on media size, file size, or file/folder count.
- Up to 8 simultaneous file access.

For our design, we choose to implement it using the UART interface. A simple schematic of how the ALFAT SoC interacts with other devices is as follows,

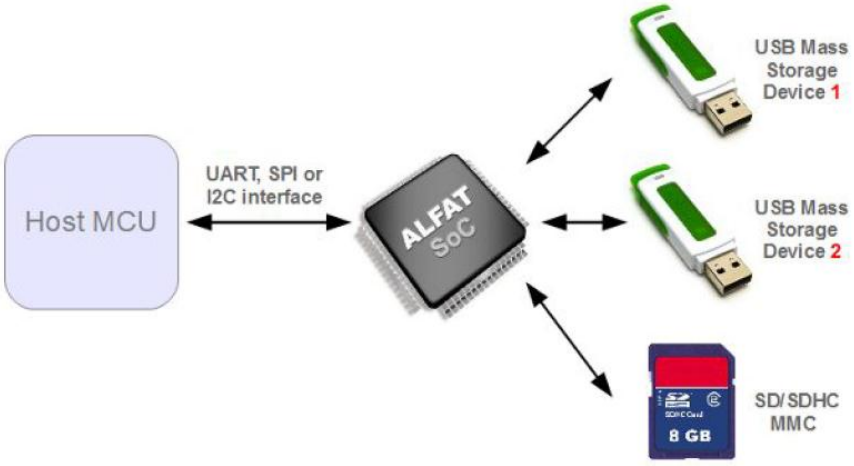


Figure 11: ALFAT SoC interface (ALFAT SoC Processor User Manual, p. 5)

After power-up, the Host MCU (our FPGA board) controls the ALFAT SoC by sending in commands in human-readable ASCII format. The command list is as follows,

Command	Description	Command	Description
V	Get version number	I	Initialize MMC/SD or USB
Z	Low power	O	Open file to a free handle
T	Initialize Real Time Clock	W	Write to a file
S	Set current time and date	R	Read from a file
G	Get current time and date	F	Flush file
B	Change Baud rate	C	Close file
#	Enable echo	P	File seek
J	Read status register	Y	File tell
E	Test media	D	Delete file or folder
K	Get free size	?	Find file or folder
@	Initialize directory list	M	Copy From File to Another
N	Get next directory entry	A	Rename file
Q	Format	L	Fast Write to file (SPI only)

Figure 12: ALFAT Command Set (ALFAT SoC Processor User Manual, p. 19)

The command will be send via a UART interface, which uses 3 hardware signals,

- UART_TX signal to send data out from ALFAT
- UART_RX signal to receive data to ALFAT
- UART_BUSY signal which should be monitored while sending data to ALFAT. When hi, no more data should be transmitted to ALFAT until it gets lo.

Besides, commands are terminated with line-feed, and the user must read back the responses for each command properly and check whether the command was successful.
To write a file to a storage device, we need the following commands:

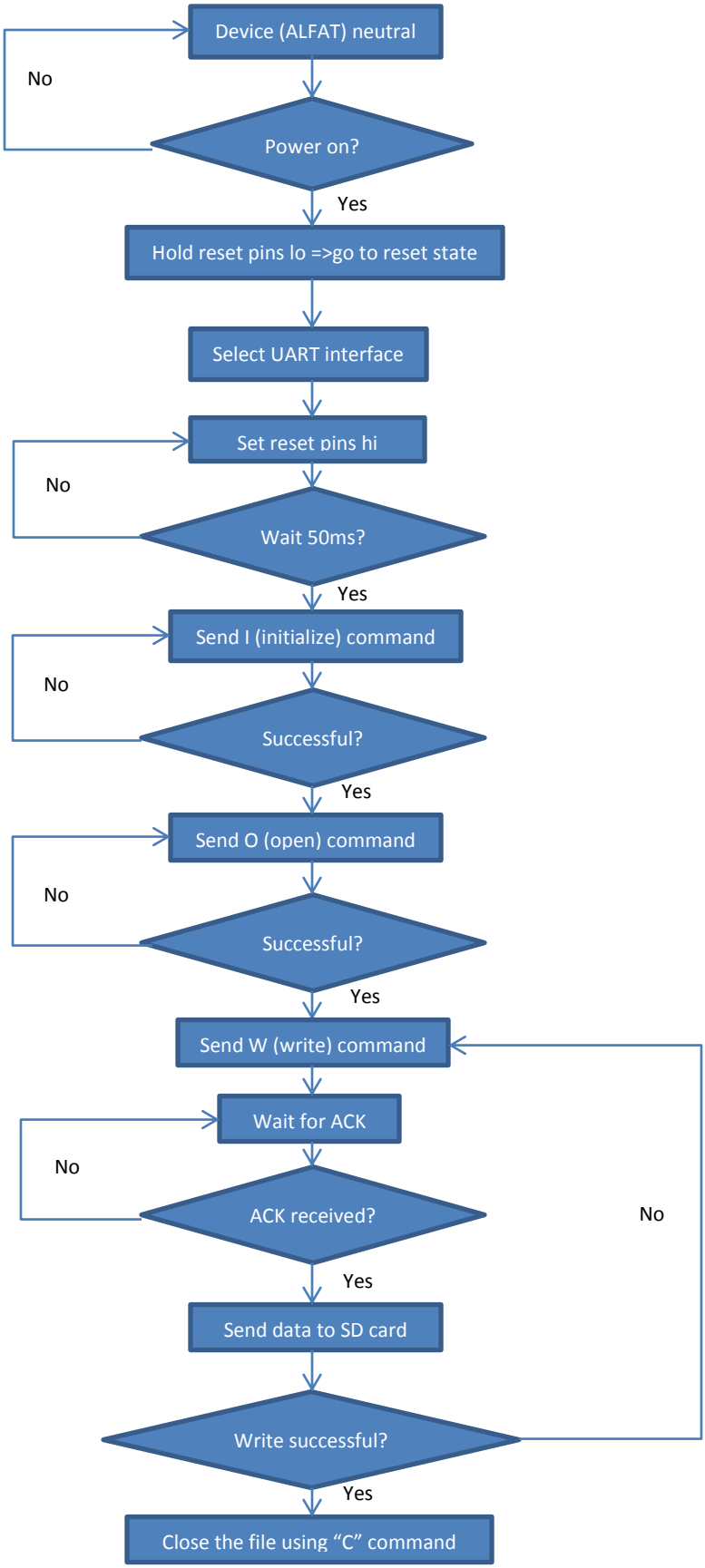
- I – Initialize and mount the storage device. If this command is not called first, the file system can be corrupted, and other file related commands will fail.
- O – Open file for Read, Write and Append, and give write privileges to it. If the file already existed, it will be erased and rewritten.
- W – Write to file through a file handle assigned to an open file with write mode. This command is accomplished through 3 steps:
  1) Send W command with file handle and the data size
  2) Wait til you get the acknowledge
  3) Send the data

To make sure the data is written to a file, the file must be flushed or closed when done or there will be a risk of losing data or corrupting the file system if the storage media was removed or if there was a power loss.

- F – Flush the data of an opened file. This command is useful to make sure all the data are physically saved in the media.
- C – Close file. This command issues a flush internally and then release the file handle.

The flowchart for a WRITE operation is as follows,

To store a file on an SD card:



2.2.5 Display Module (Software)

Video is inputted into the module from the ADC chip as luma and chroma data, which is stored in a register. This data is then converted to the RGB color space and is sent to the VGA Controller. The data from the OBDII module is also sent to the VGA Controller, which uses a ROM to call upon and display the correct character sprites. Both the raw RGB video data and that of the video with overlaid data are outputted.

YCrCb Register Bank

Inputs: 1 byte of data from the ADC chip. Also, there is an enable for each register, and a clock based on the video input.

Outputs: There are 3 bytes of outputs: one for luma and two for chroma values.

Description: 1 byte is sent from the ADC at one time, but the information changes between the luma value and the chroma value, which is two bytes. There are 3 registers then to hold the last value sent for each of those. There is also a counter that controls the enable for each register, which is timed by the input clock. Each of these registers then outputs that byte to the YCrCb to RGB converter.

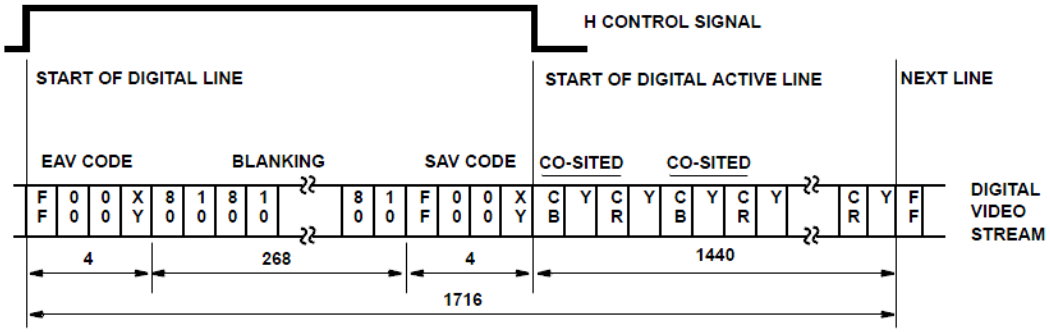


FIGURE 1. BT.656 8-BIT PARALLEL INTERFACE DATA FORMAT FOR 525/60 VIDEO SYSTEMS

Figure 13: BT.656 8-bit parallel interface data format for 525/60 video systems (from AN9728 Application Notes)

YCrCb to RGB Converter

Inputs: The Y, Cr, and Cb bytes from the YCrCb register bank. There is also a clock, as well as horizontal and vertical syncs.

Outputs: 3 bytes are outputted: one for red, one for green, and one for blue.

Description: the Y, Cr, and Cb values are all constantly inputted. Then, they are converted into the RGB color space. The conversion calculations are explained in the next section. Once the conversions have been performed, the RGB values are sent to the VGA Controller as 3 separate bytes.

VGA Controller

Inputs: The converted RGB values are received by the VGA Controller. Also, the data from the Converter Bank in the OBDII Interface is inputted, as well as the corresponding signals indicating the last parameter from the OBDII that was received and converted. Also, data is received from the Sprite ROM for overlaying text onto the display. The horizontal sync and the vertical sync are also inputted for timing purposes.

Outputs: A two byte signal is sent to the Sprite ROM for summoning certain sprites, which are all letter and number characters, to be displayed. Also, the video signal is outputted to the encoder one byte at a time.

Description: The VGA Controller will output information for each individual pixel going from left to right, then up to down. It will output the input for most of these pixels. However, at some points this entity will call upon the Sprite ROM, which then sends back 8 data bytes, each corresponding to 8 pixels on a single row of pixels. For each pixel, if the bit from the Sprite ROM is low, then the video outputted will simply be the RGB value inputted. However, if the value from the ROM is high, then the pixel will be changed to a different color. This will create characters on the screen. Some of these will be constant to describe the parameters, such as RPM, air temperature, etc. Others will be determined by the integer values of the data sent from the OBDII. These will be the numeric characters, and an if/then statement will be used to display the appropriate character.

Sprite ROM

Inputs: The address being summoned by the VGA Controller is inputted.

Outputs: The ROM outputs the data bytes from the addresses received.

Description: This entity acts as a simple read-only ROM, which outputs prewritten data to the VGA, depending on the address requested. One address is read from at a time, and one address is outputted at one time. Each address and each line of data are both 1 byte each.

2.2.6 Power Module

The Automotive Racing Video Data Logger device gets power from the 12V car battery. However, 12V is higher than needed for our device, so DC/DC voltage converters will be included in the power module to step down 12V to other useful voltages.

Specifically, here are the power requirements for all the components from the device:

Devices	Voltage(s) needed
Digilent Basys2 Development Board	3.5V-5.5V
ALFAT OEM Board	3.3V
FPGA board	3.3V, 1.8V, 1.5V, 1.2V
TFT LCD display	5V

From the 12V car battery, the voltage will be stepped down to 5V, and then 5V will be stepped down one more time to get 3.3V to supply power to the ALFAT OEM Board, the FPGA Board, etc.

3.3V will then be stepped down again to 1.8V and 1.5V to supply power to logic signals on the FPGA board. The power module block diagram from section 2.1 is copied below to illustrate how the car power supply will be converted to use on the device:

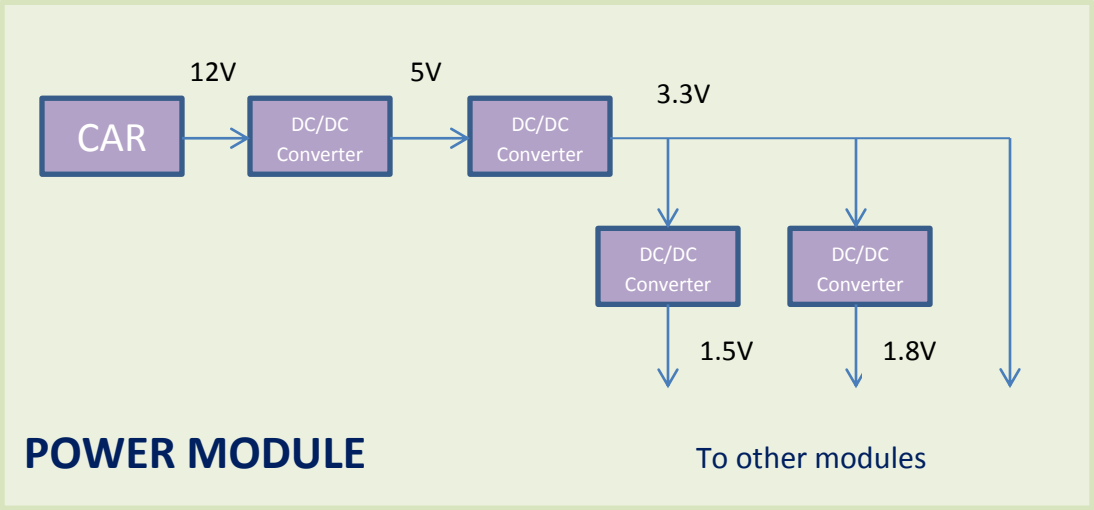


Figure 14: Power Module Block diagram

Overall the digital flowchart of the system is as follows,

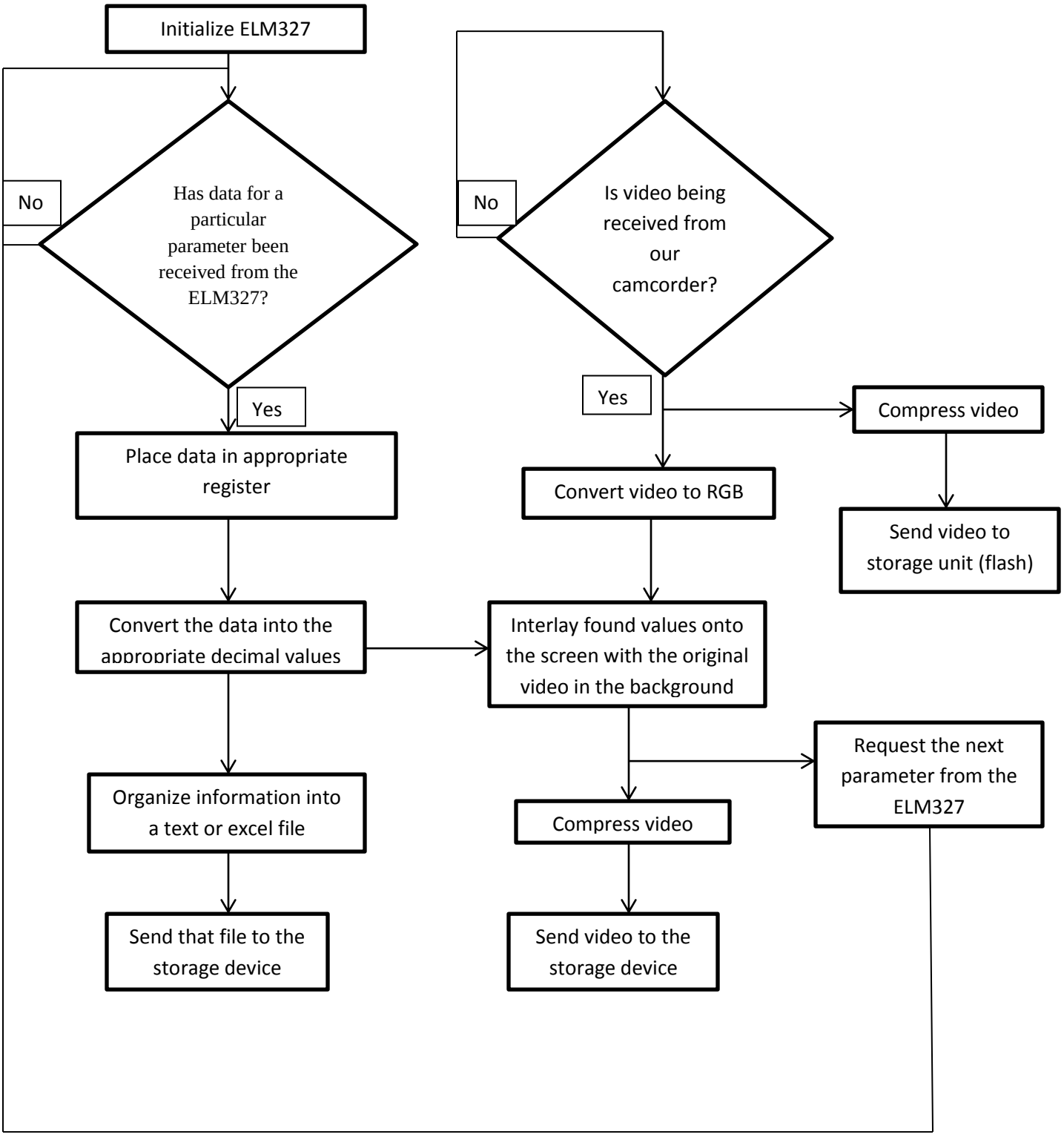


Figure 15: Overall Digital Flowchart

2.3 Schematic of Overall System

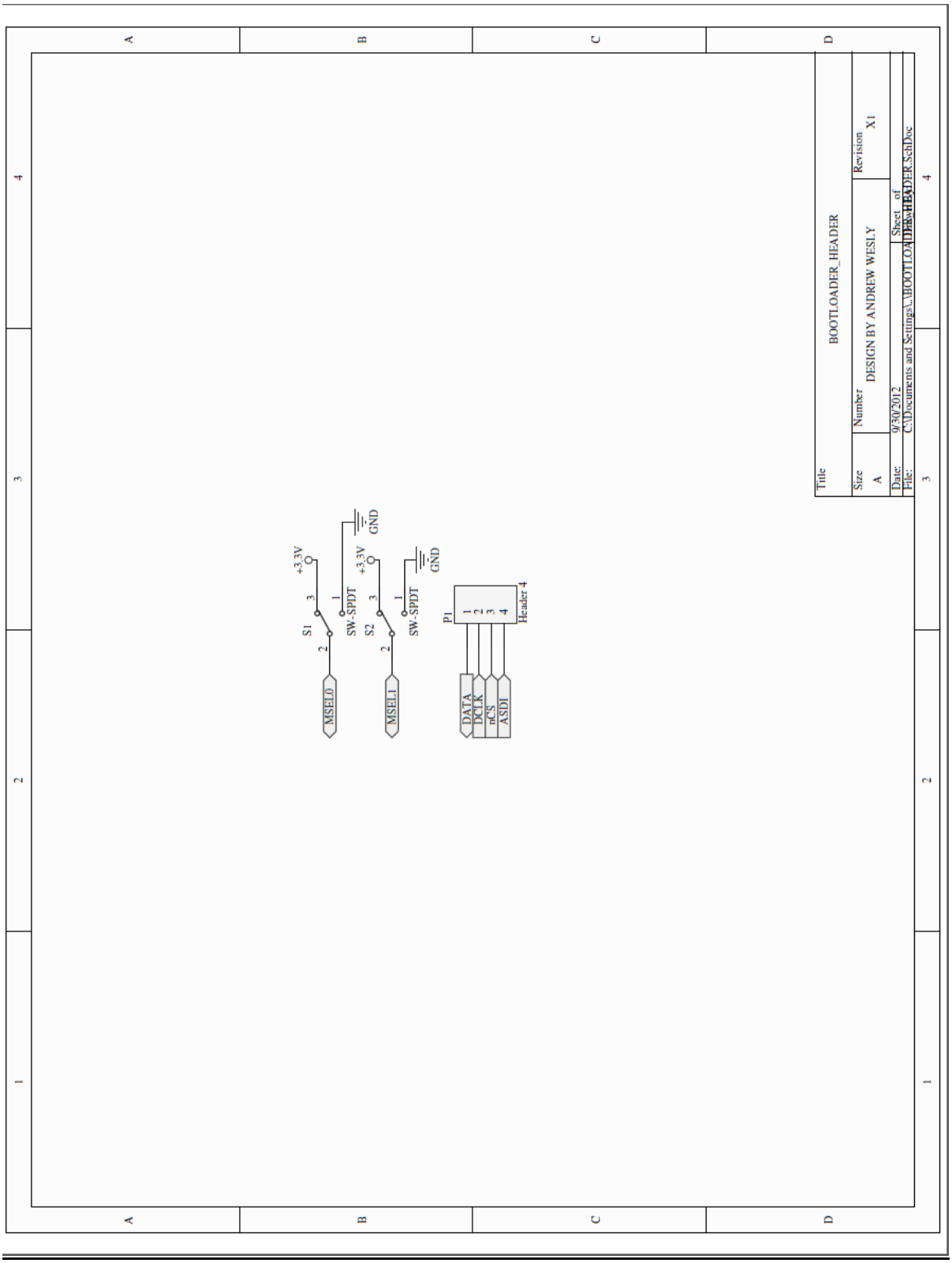


Figure 16: Schematic of the Bootloader. There is a 4 pin header, P1 for Active Serial Configuration. It connects to the Basys2 bootloader configured board. Switches MSEL0 and MSEL1 are used to set boot configuration schemes such as Active Serial, Passive Serial and JTAG.

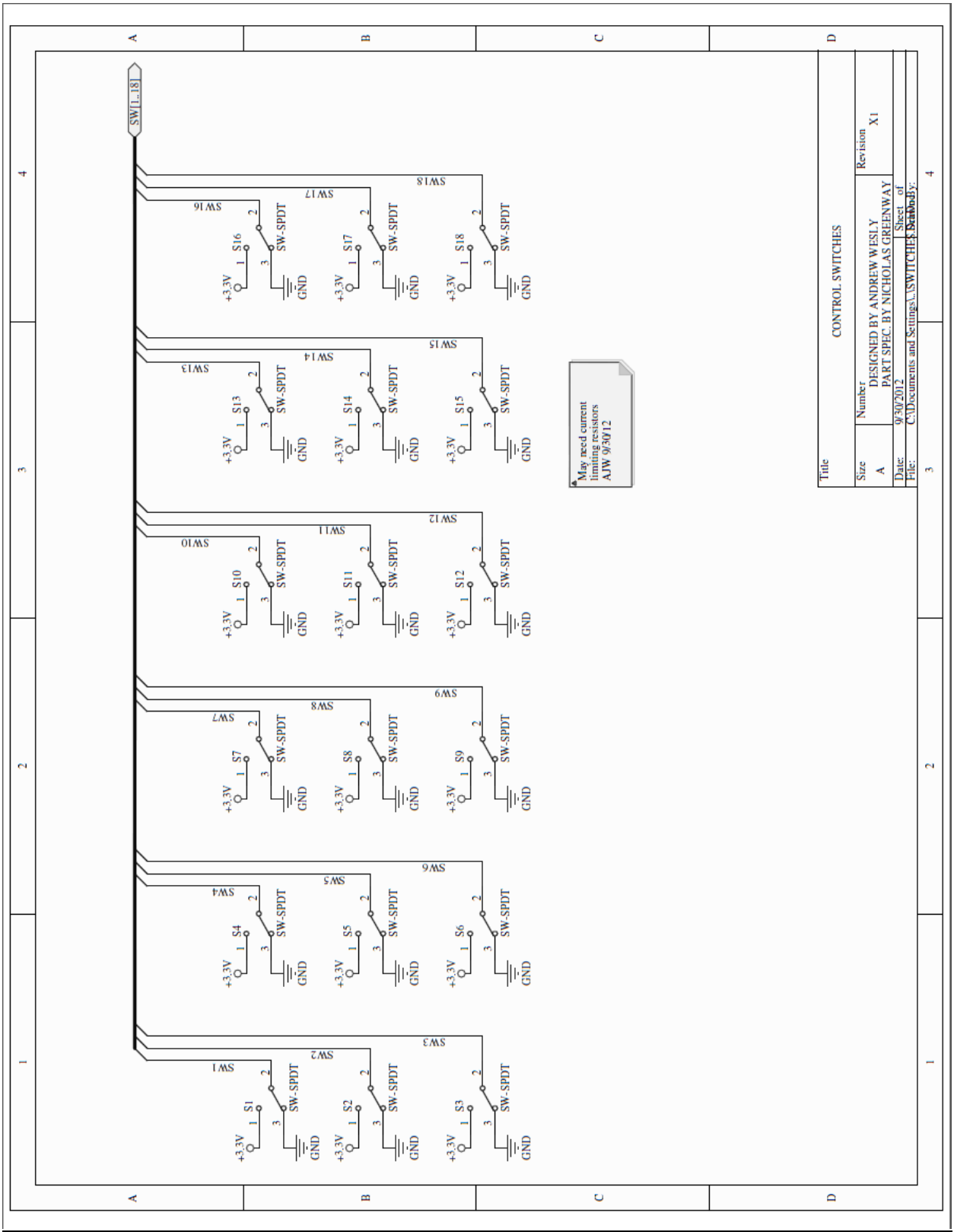


Figure 19: Schematic of the Control Switches. Only a few switches are required for Play,Stop,Record, Next File, Previous File and New File settings. We design for 16 switches because we can use them for future functions. Software changes are cheap, hardware modifications are expensive!

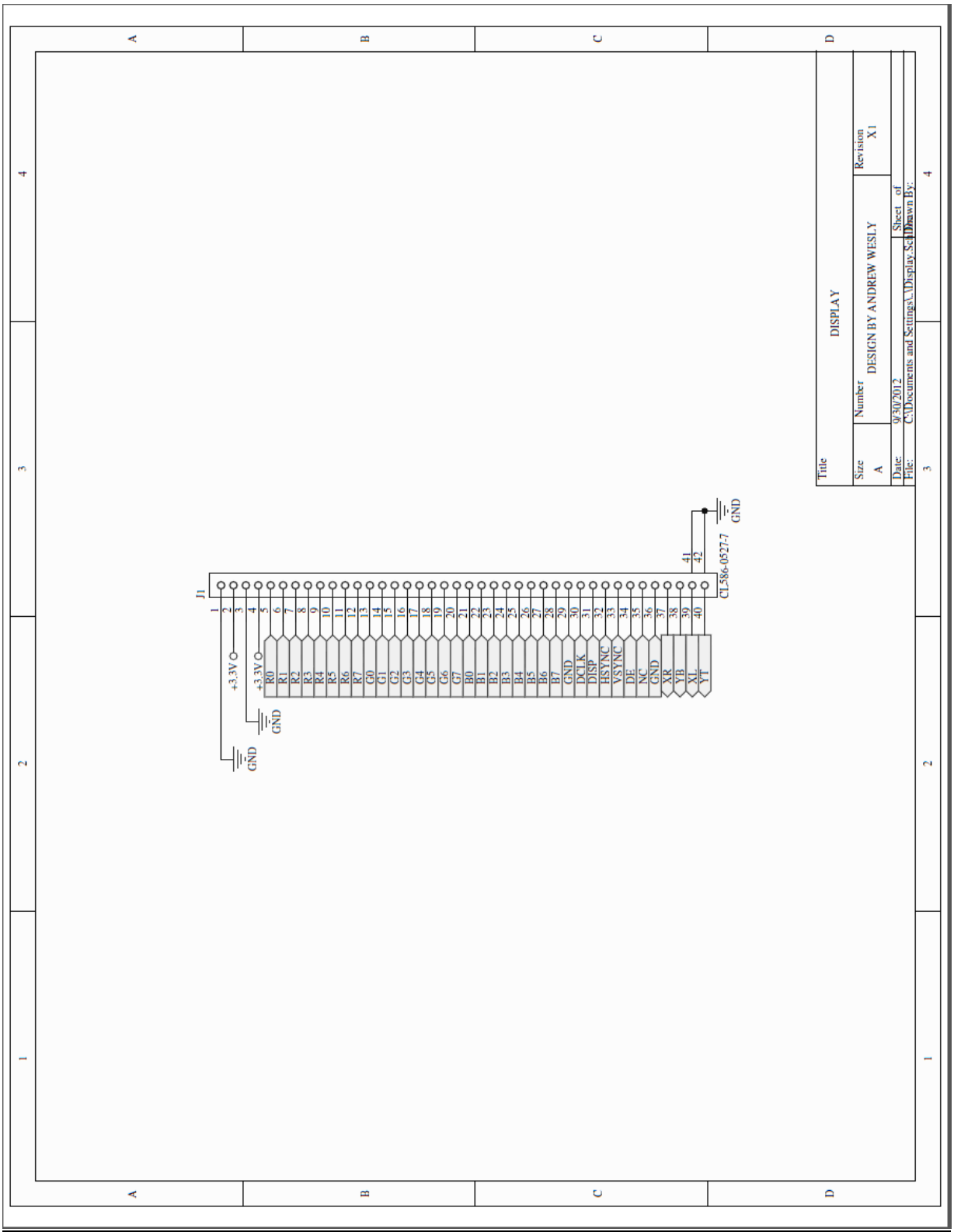


Figure 20: Schematic of the TFT Display. This schematic is a 40-pin TFT header that connects to the Kentec LCD display module. All signals are 3.3V compatible and the screen backlight is powered by on-board 3.3V power. Present are 8-bit RGB signals and HSYNC/VSYNC lines for video timing information

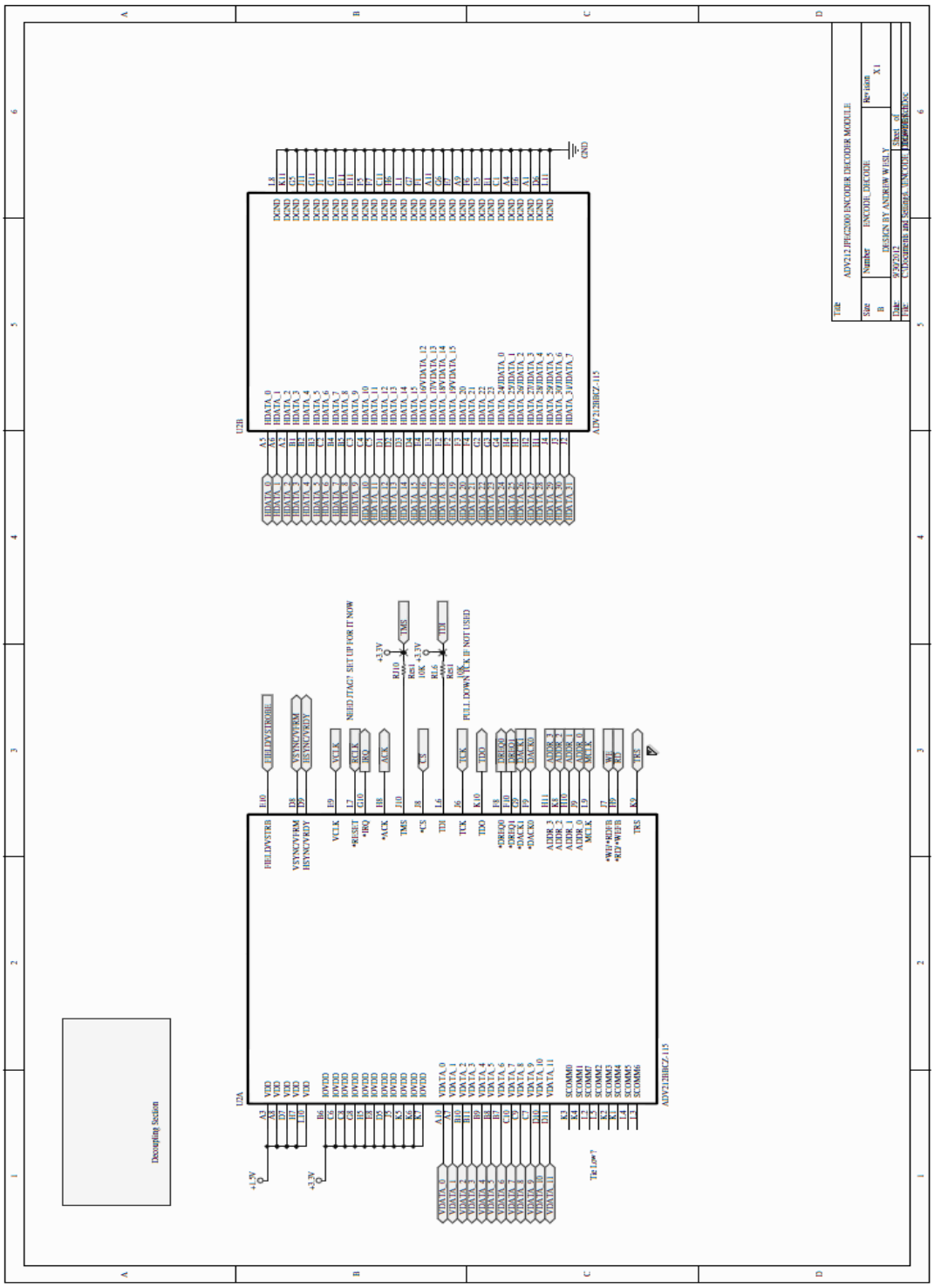


Figure 21: Schematic of the Decoder/Encoder. The Encoder/Decoder schematic depicts an Analog Devicse ADV212 JPEG2000 encoder/decoder. Module input and output include HDATA (Host Data) bus that is used to configure the device and transport video to the FPGA. VDATA lines are also used to stream video through the device. Also present in this module are decoupling capacitors that are used to decouple this device from board power. This way, power draw at another part of the board does not cause this component to fail, nor does a large instantaneous power draw from this component cause

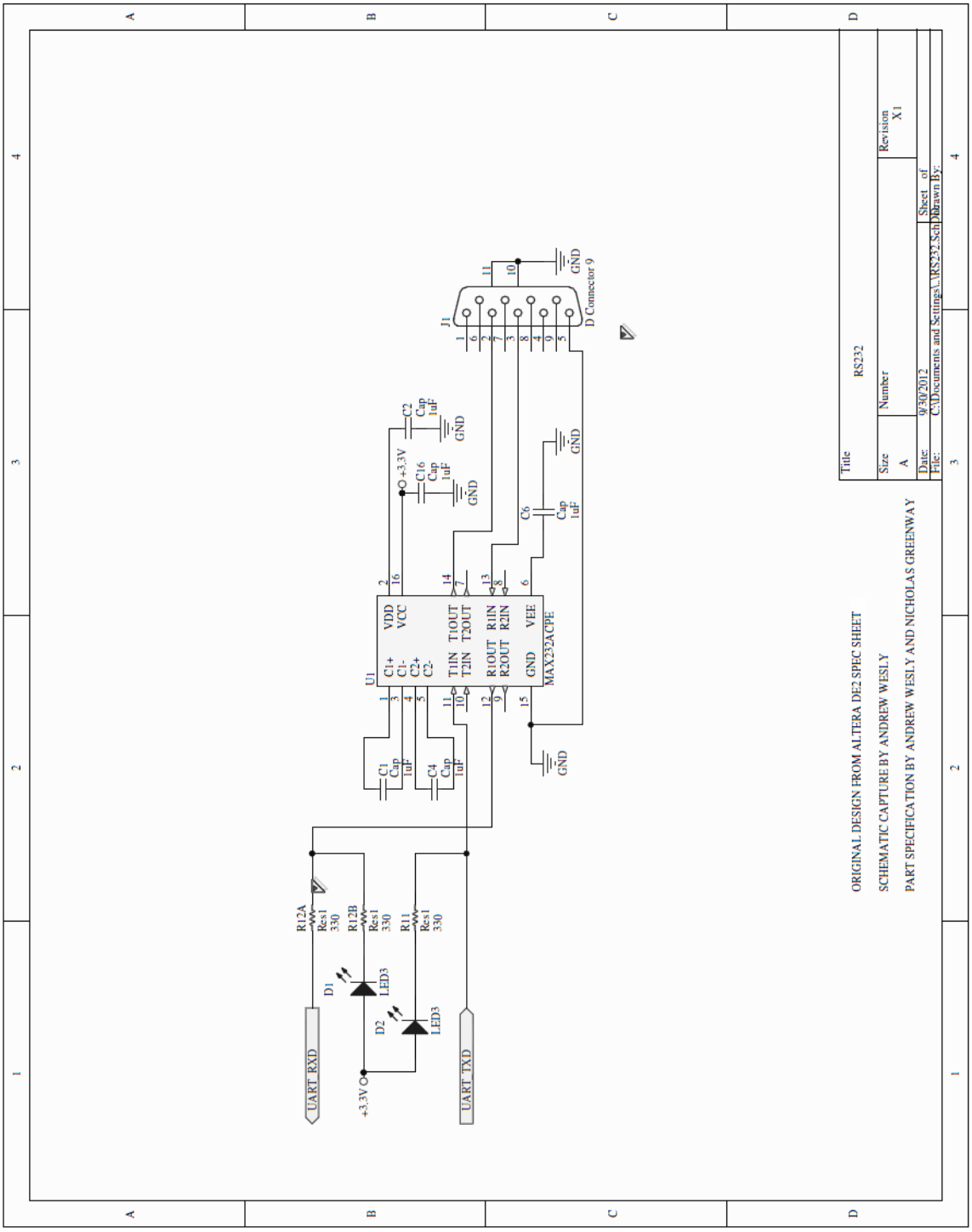


Figure 22: Schematic of the RS232 module. The RS232 module takes 3.3V CMOS signals from the on-board FPGA and converts them to +/-12V levels used for RS-232 communications with the OBDII module. Input is UART protocol signaling from the FPGA that is input to the MAX232ACPE level converter chip. This chip is powered by on-board 3.3V power that is stepped up to +/-12V by IC charge pumps. Output from the MAX232ACPE is then routed to a DB9 connector that is connected to the OBDII communications module.

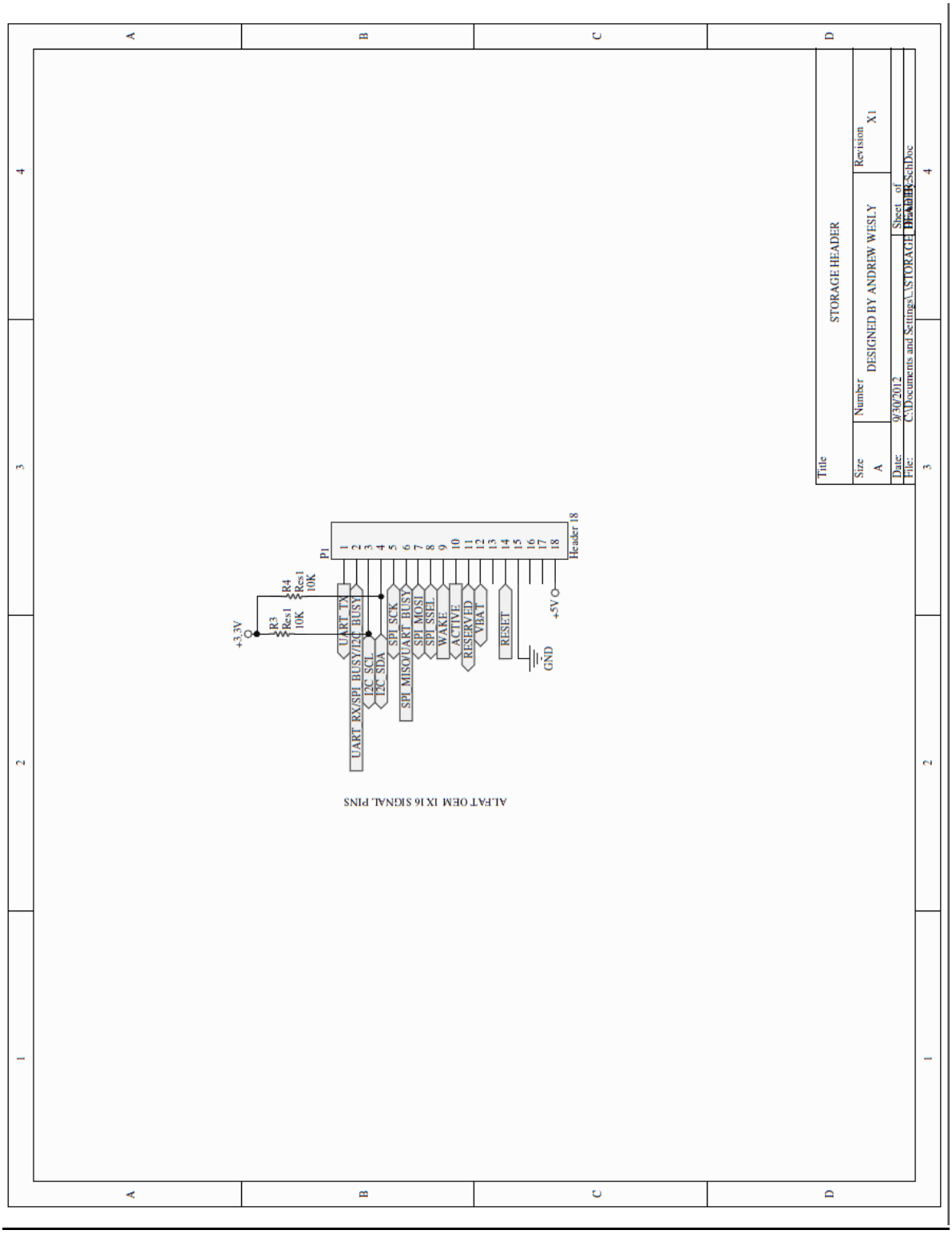


Figure 23: Schematic of the Storage Header. The Storage header allows for I2C, UART and SPI connectivity to the the ALFAT storage unit. We intend to use one protocol, however during the course of this design another choice may be needed, therefore all types of signal protocols are chosen for this project. Not the 10KOhm pull-up resistors – they are used to pull-up the I2C bus per manufacturer recommendations.

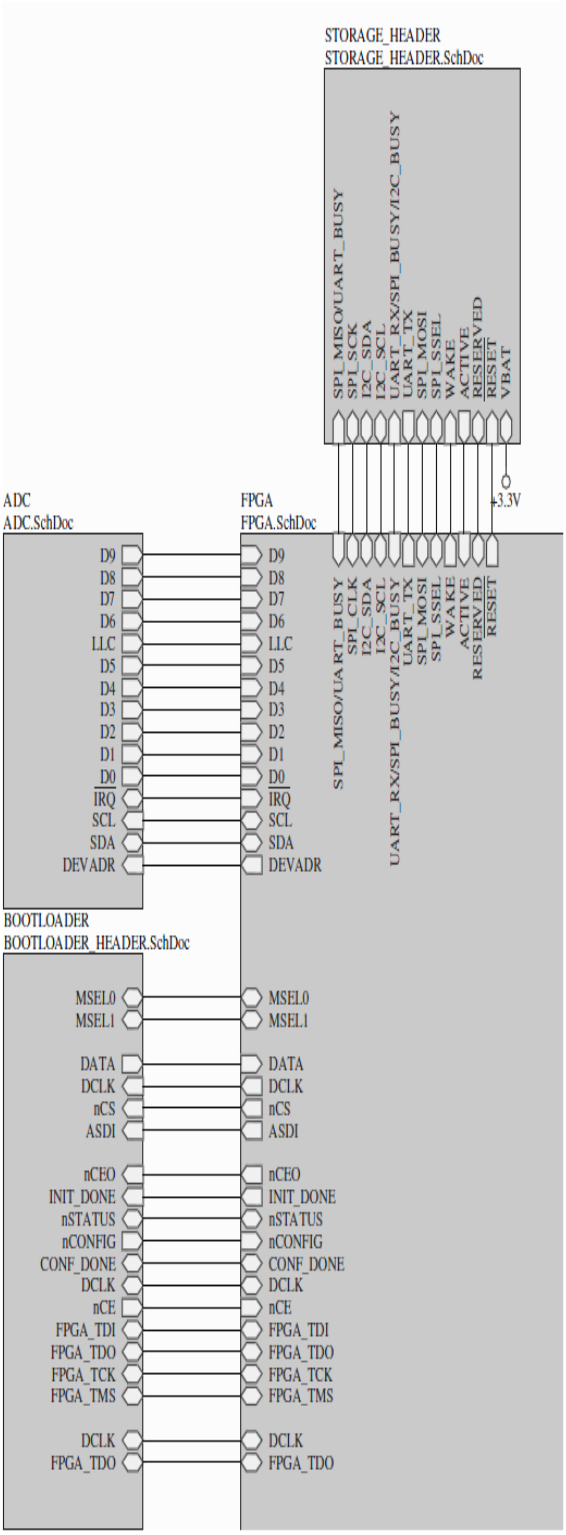


Figure 24: Top Level Schematic of ADC – Storage – Bootloader

This figure shows a hierarchical schematic connection between ADC, Bootloader, Storage and FPGA modules.

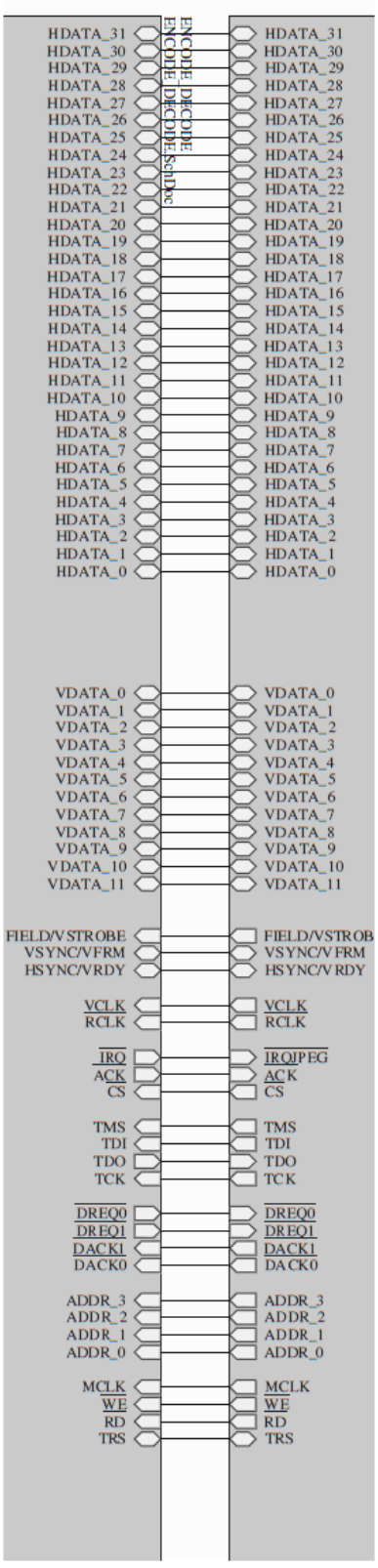


Figure 25: Top Level Schematic of Encoder/Decoder

This figure shows connections between the FPGA and ADV212 JPEG2000 ENCODER/DECODER module.

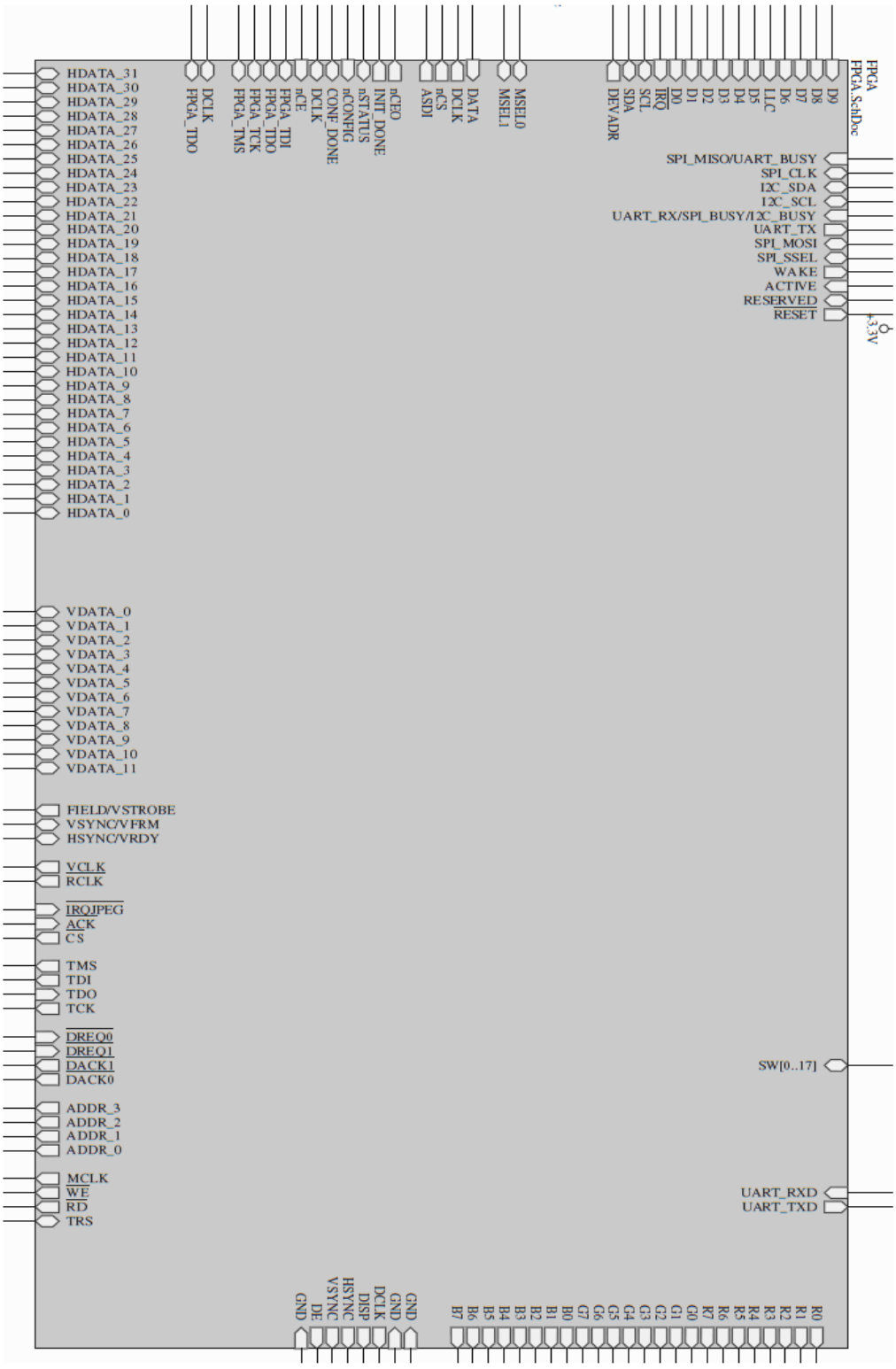


Figure 26: Top Level Schematic of FPGA

This schematic shows the inputs and outputs from the FPGA module to all other devices

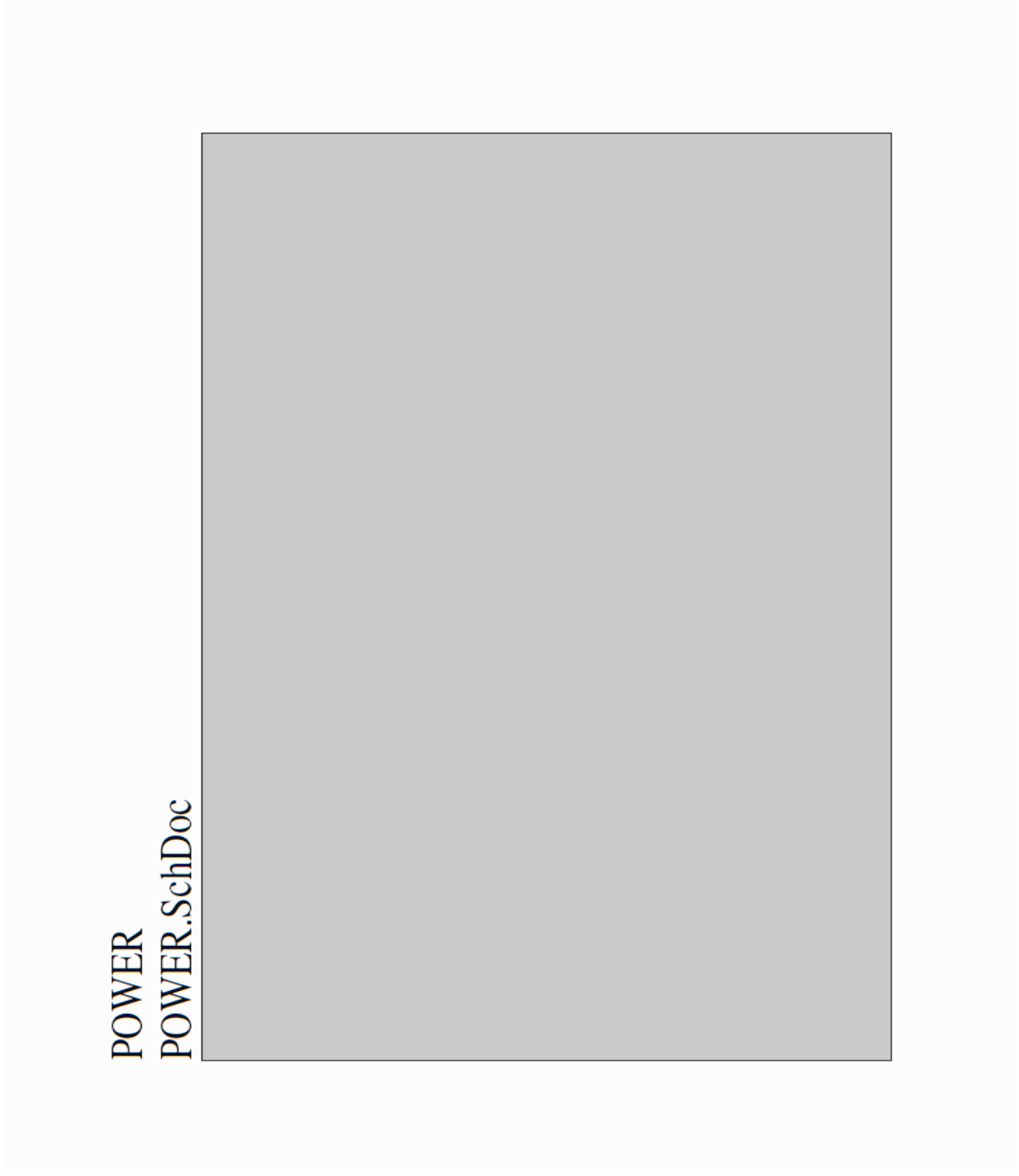


Figure 27: Top Level Schematic of Power Module

The power module is comprised of DC/DC converters that take 12V Automotive voltage and supply 1.5V, 1.2V, 3.3V and 1.8V to appropriate on-board modules.

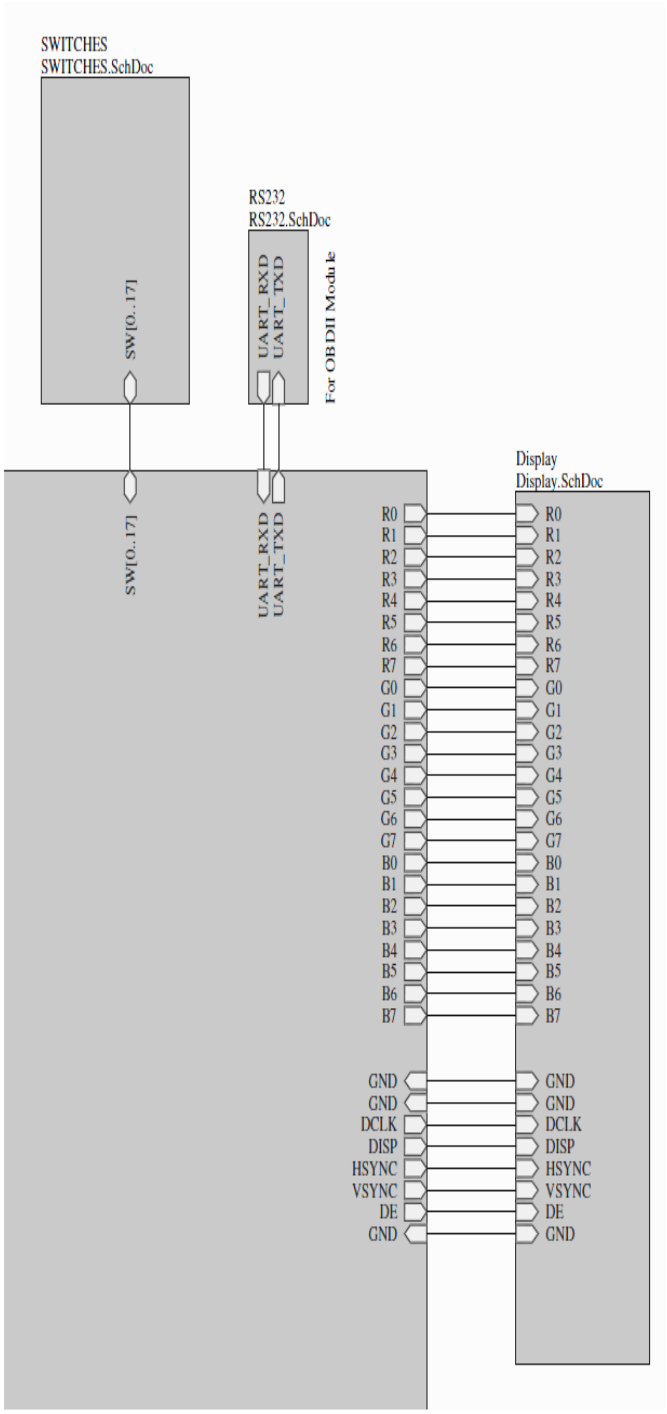


Figure 28: Top Level Schematic of Switches – RS232 – Display

This figure shows connectiveity between control switches, RS232 communications module and the Display module

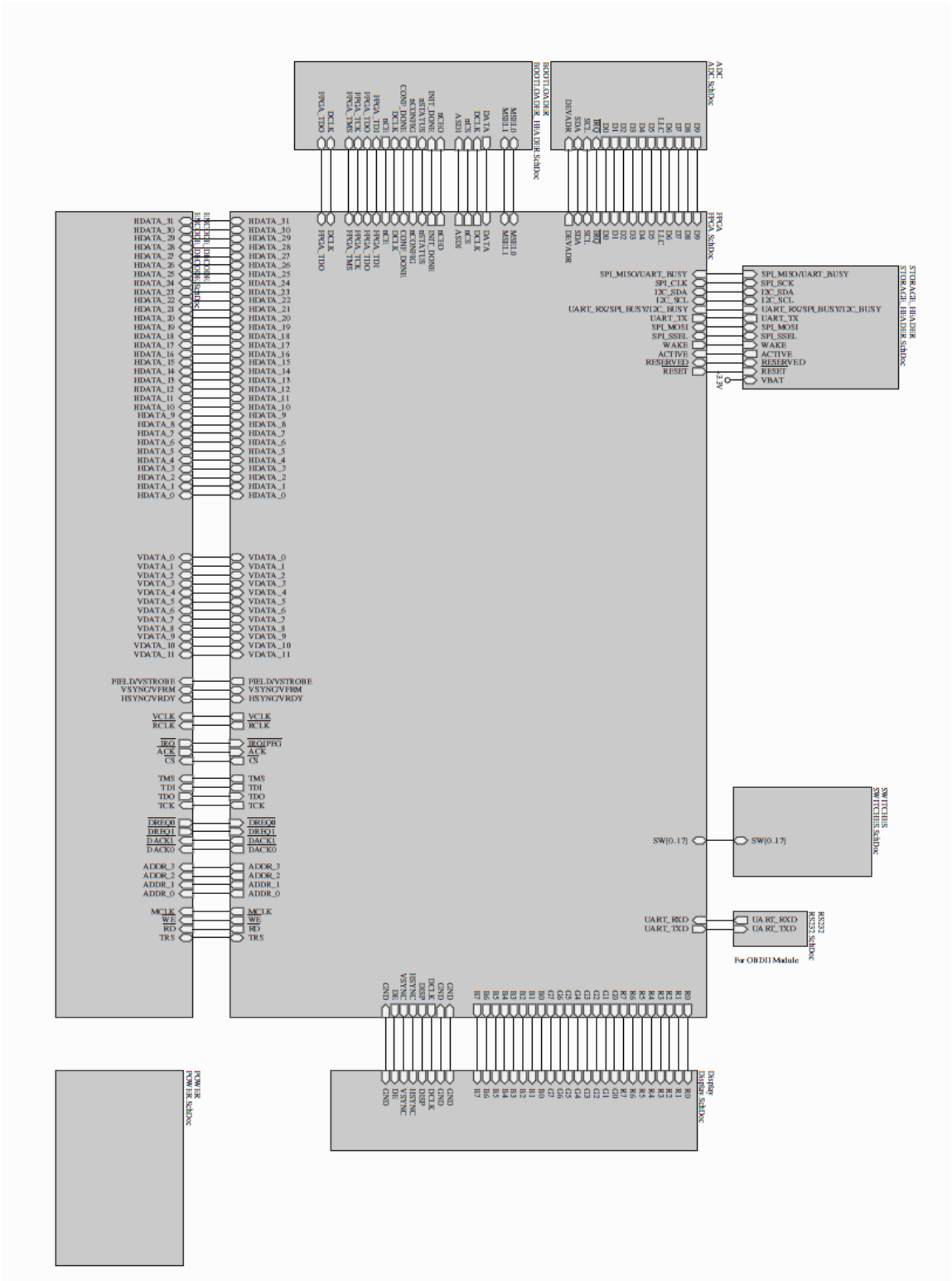


Figure 29: Top Level Schematic of Switches – RS232 – Display

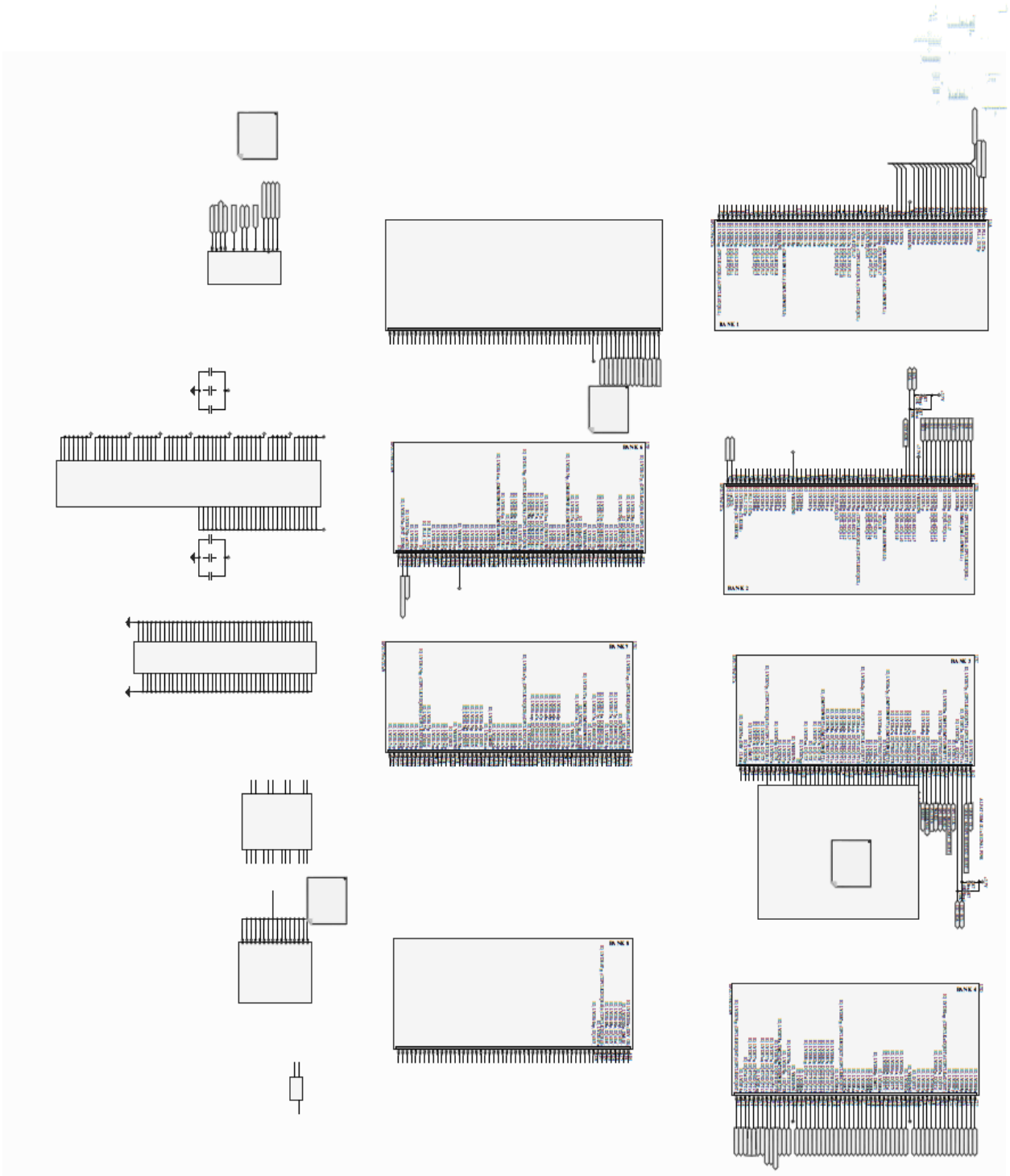


Figure 30: Top Level Schematic of FPGA The EP2C35F672C6 FPGA has 8 IO banks and 4 on-chip PLLS. It also has 672 pins as displayed in the top-level diagram above. Schematics that follow are zoomed in on FPGA sections.

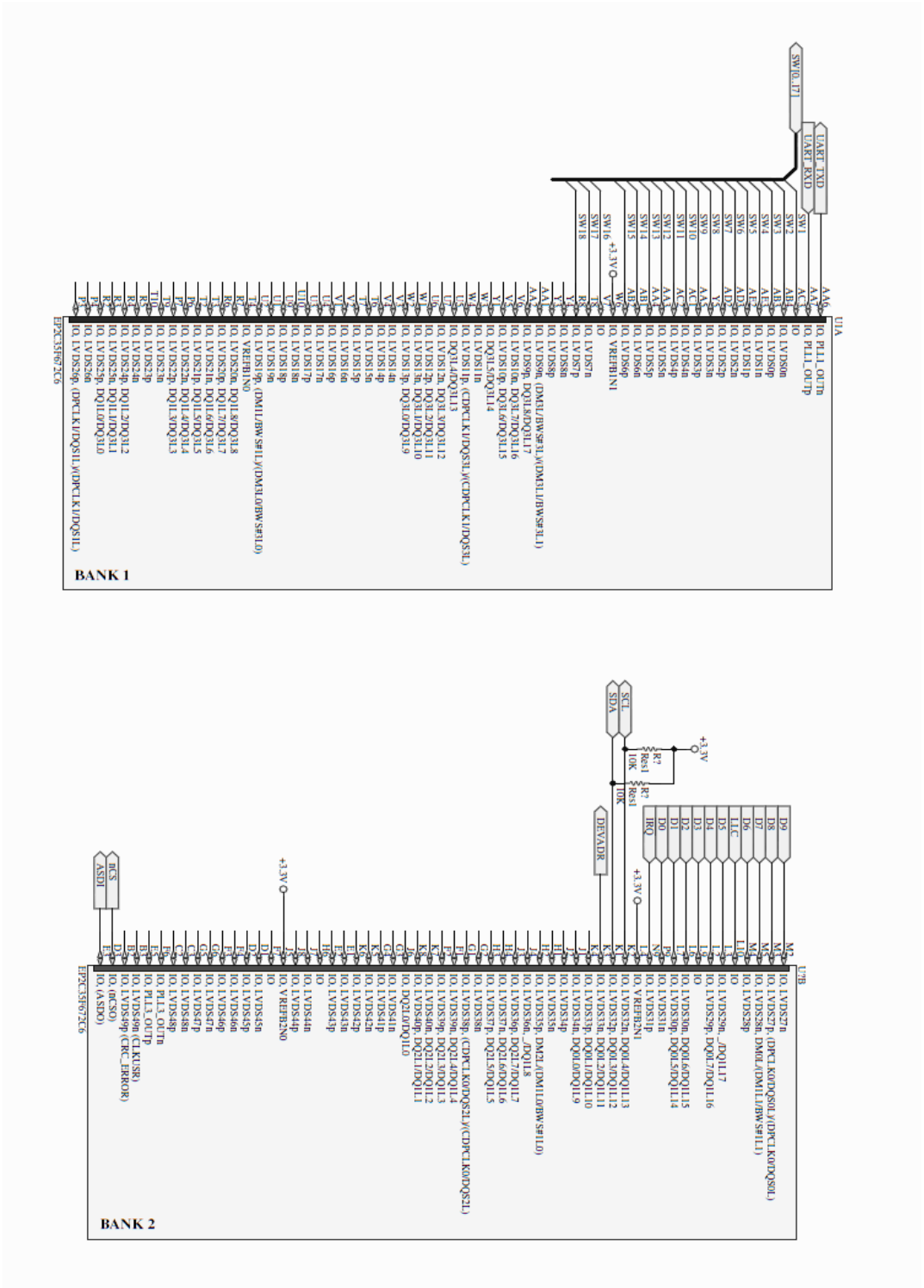


Figure 31: Schematic of FPGA bank 1 and bank 2. Schematic banks one and two are connected to 3.3V for IO power. IO power is 3.3V for signaling voltage. Bank one is connected to switches for control inputs. Banks two is connected to the ADC. Pinouts may change during the PCB layout stage for ease of FPGA signal extraction.

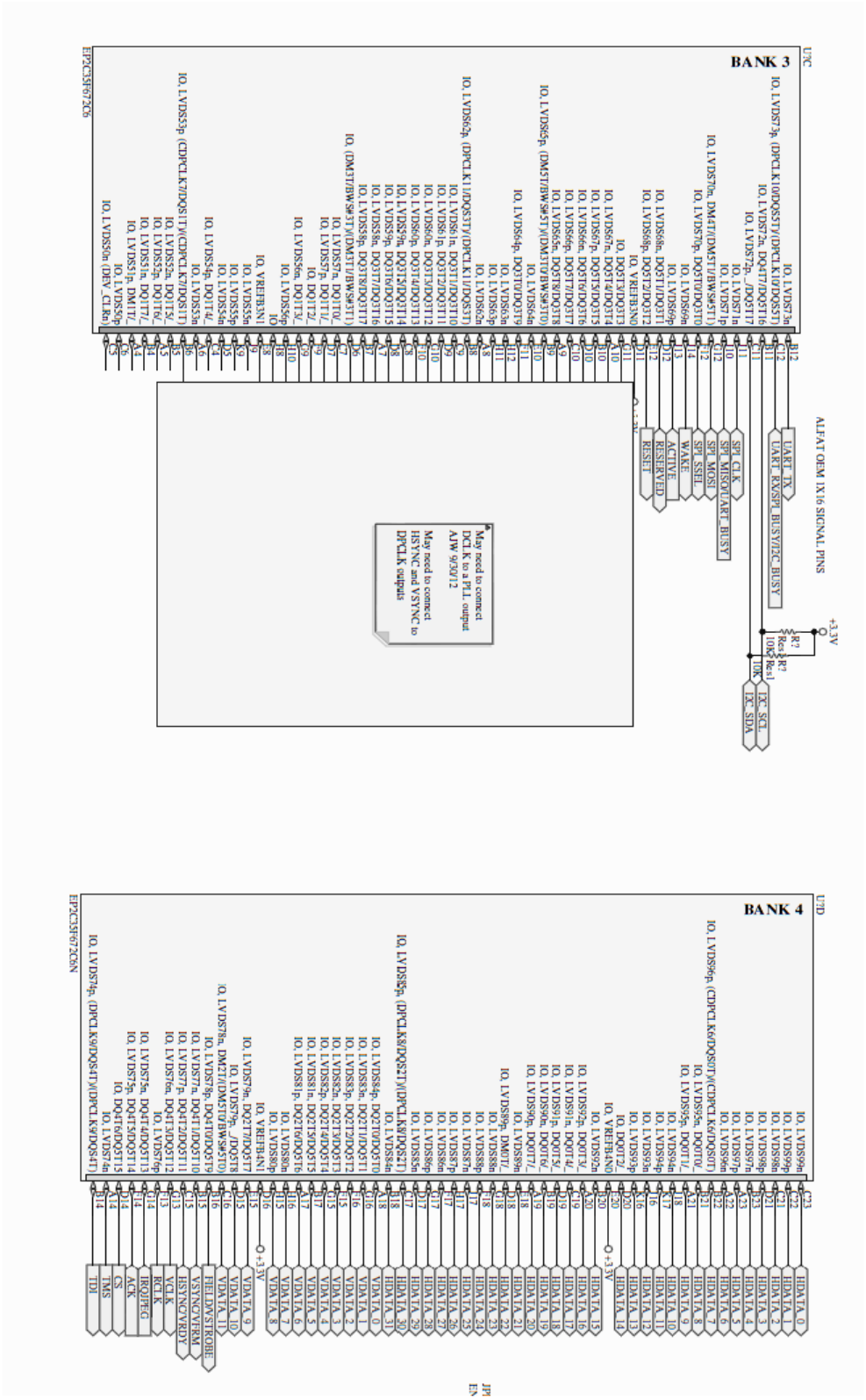


Figure 32: Schematic of FPGA bank 3 and bank 4 are both connected to +3.3V for signaling.Bank3 is connected to the ALFAT storage module and Bank 4 is connected to the ADV212 JPEG 2000 ENCODER/DECODER Module.

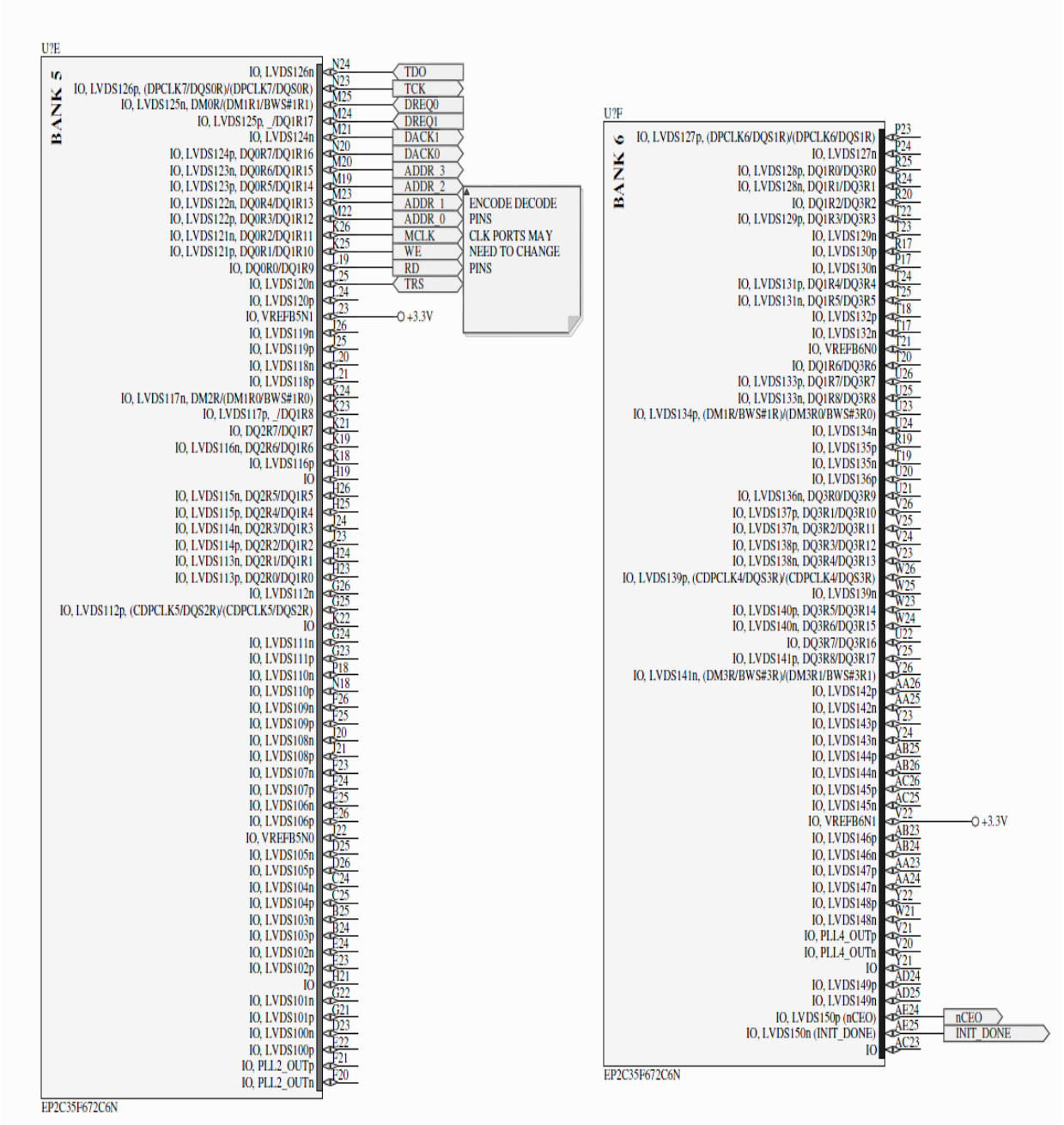


Figure 33: Schematic of FPGA bank 5 and bank 6. These banks are connected to +3.3V for signal levels. Bank 5 has some Encoder/Decoder pins. Bank 6 is has pins nCEO and INIT_DONE that are connected to the Bootloader module for FPGA programming.

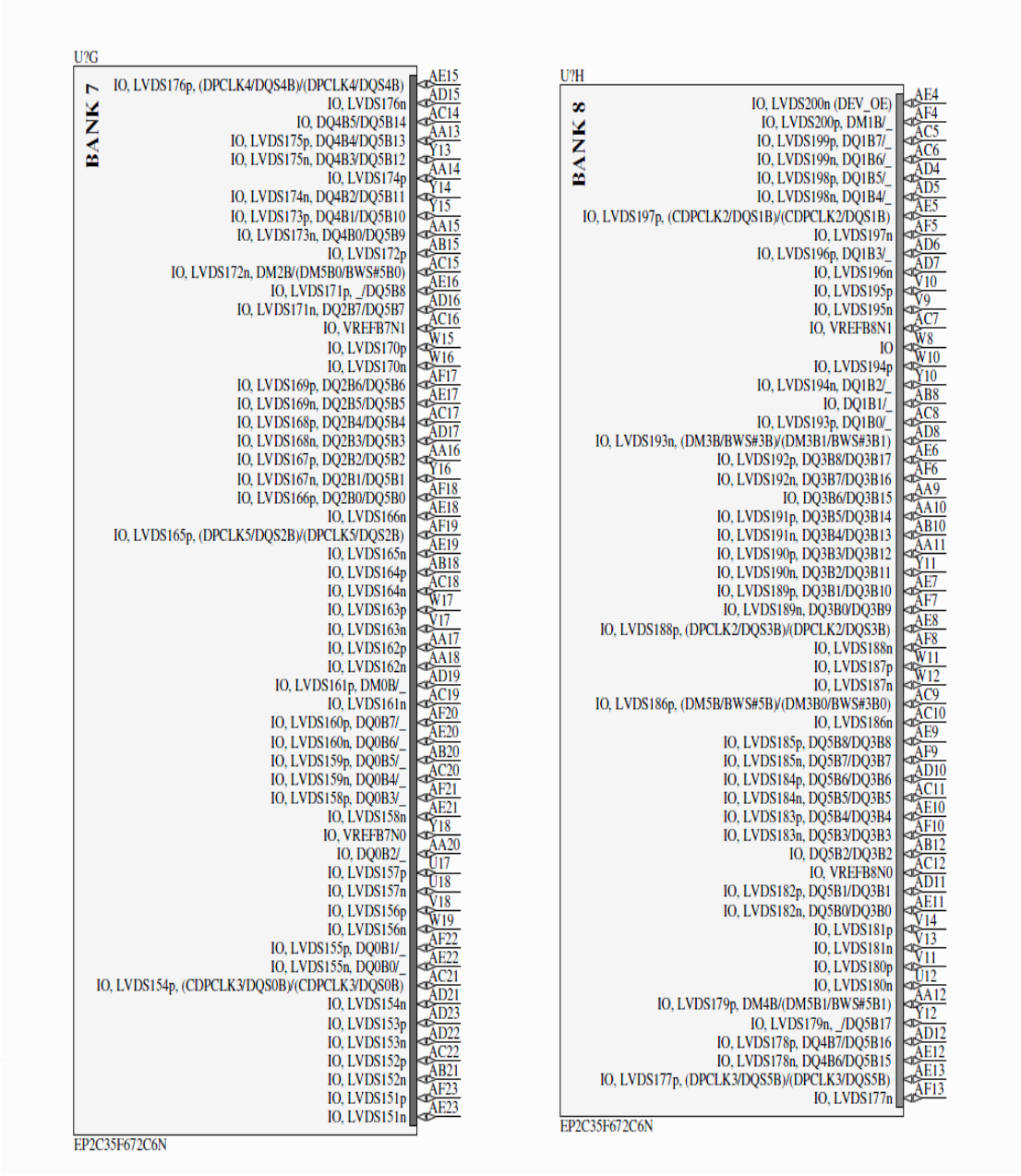


Figure 34: Schematic of FPGA bank 7 and bank 8. These IOBANKS are left unconnected

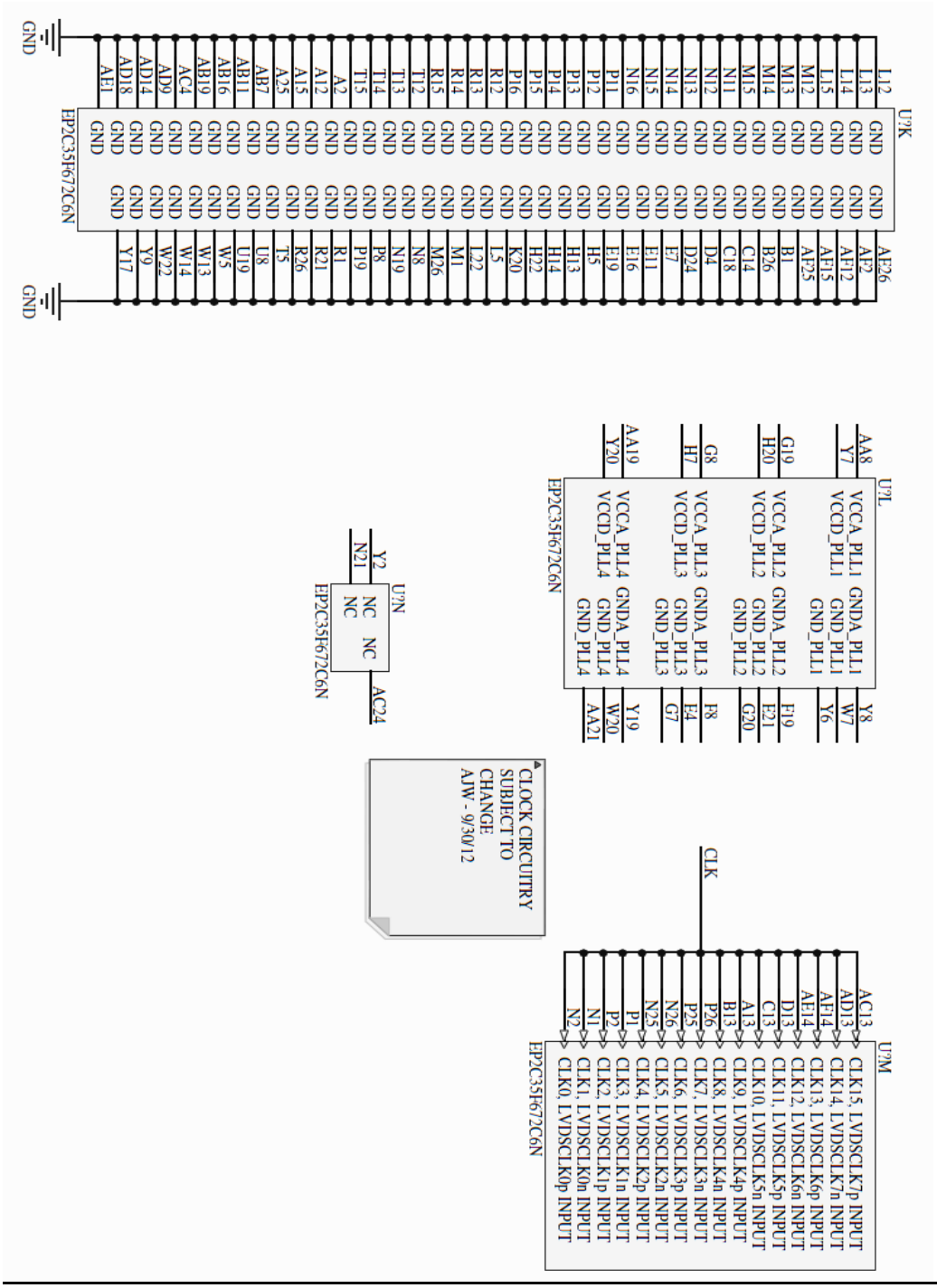


Figure 35: Schematic of FPGA Clock – PLL – GND . Current design requires one clock signal that is connected to FPGA global clock network. Internal FPGA PLLs may need to be powered on for correct operation. This design aspect is subject to change during the building process.

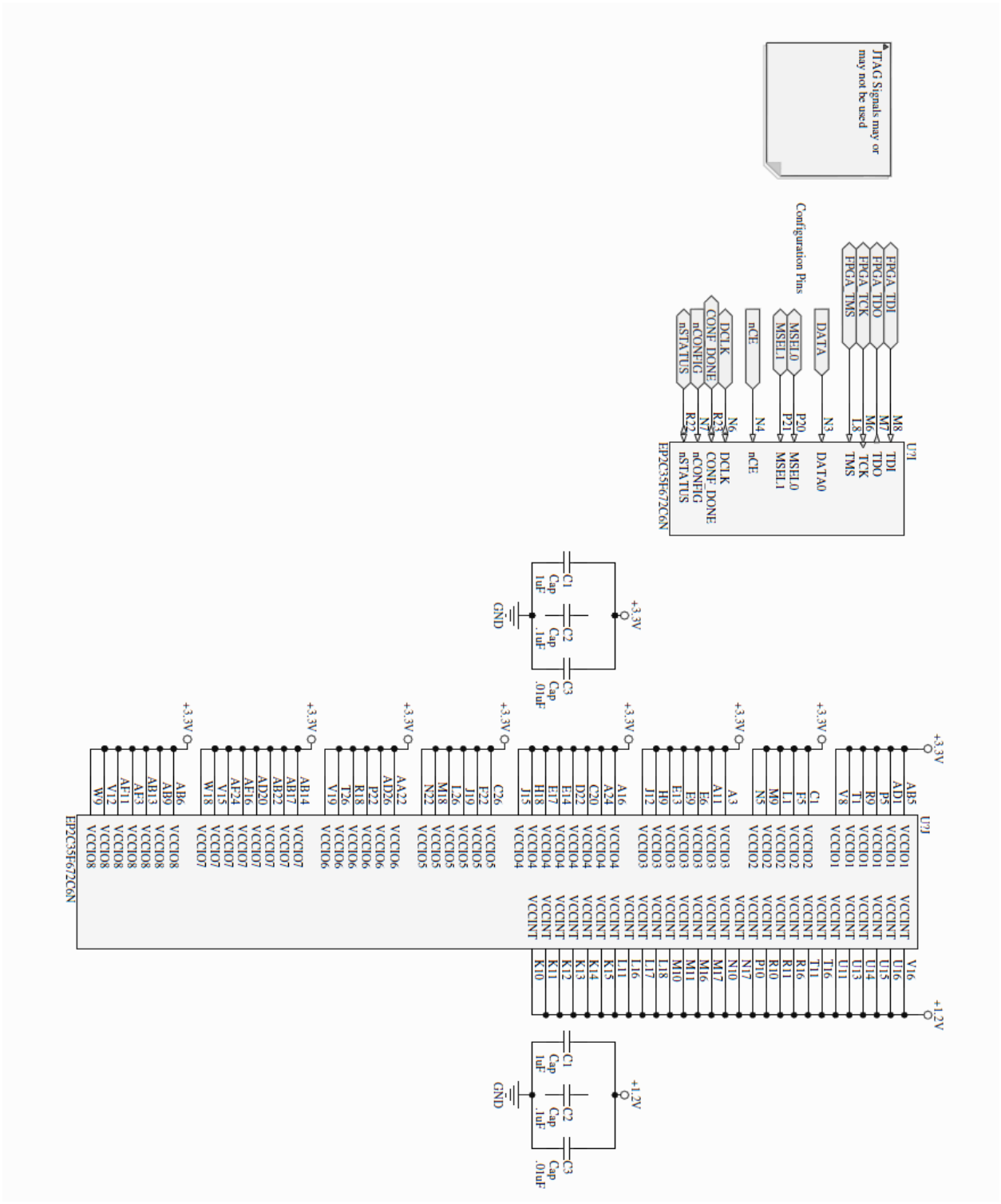


Figure 36: Schematic of FPGA JTAG Program Power

2.4 Simulations and Calculations

Color Conversion Calculations:

From Poynton's "Introduction to Digital Video" (p. 176, equations 9.6 and 9.7), the formula to convert RGB to YCbCr signal is,

$$\mathrm{YCbCr} = \mathrm{T} * \mathrm{RGB} + \mathrm{offset}$$

where,

$$\mathrm{T} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix}$$

and,

$$\mathrm{offset} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

This equation can be rewritten as,

$$\mathrm{RGB} = \mathrm{T}^{-1} * (\mathrm{YCbCr} - \mathrm{offset})$$

where,

$$\mathrm{T}^{-1} = \begin{bmatrix} 0.00456621 & 0 & 0.00625893 \\ 0.00456621 & -0.00153632 & -0.00318811 \\ 0.00456621 & 0.00791071 & 0 \end{bmatrix}$$

Color Conversion Simulation:

Using the derived equations, we implement our color conversion on MATLAB as follows,

```
function [ R, G, B , rgb] = YCbCr2RGB( Y, Cb, Cr )
%Function to convert from YCbCr to RGB used for simulation
    % Input checks
    if (Y>255 || Y<0 || Cb>255 || Cb<0 || Cr>255 || Cr<0)
        error('Invalid Input Range');
    end

    % Converting to fixed point
    max = 255;
    Y_r = (Y/max)*(2^7);
    Cb_r = (Cb/max)*(2^7);
    Cr_r = (Cr/max)*(2^7);

    % New fixed point inputs
    YCbCr = [Y_r; Cb_r; Cr_r];

    % T matrix and offset used for calculations
    T = [65.481 128.553 24.966; -37.797 -74.203 112; 112 -93.786
-18.214];
    offset = [16;128;128];
    % Inverting the T matrix
    Tinv = T^-1;

    % The final RGB map result
    rgb = Tinv*(YCbCr-offset);

    % Converting the fixed point to RGB results
    R = (rgb(1)/(2^7))*max;
    G = (rgb(2)/(2^7))*max;
    B = (rgb(3)/(2^7))*max;
```

Some results from this MATLAB code are as follows,

```
>> [R,G,B] = YCbCr2RGB(128, 128, 128)

R =

   -0.3560


G =

    1.0389


B =

   -0.5657
```

```
>> [R,G,B] = YCbCr2RGB(255, 128, 0)

R =

   -0.5772


G =

    2.0269


B =

    0.0142
```

```
>> [R,G,B] = YCbCr2RGB(255, 128, 64)

R =

   -0.1766


G =

    1.8229


B =

    0.0142
```

Data Rate:

Our data capture rate is limited by our storage rates. Maximum write speed for the storage module, ALFAT SoC is only 1400 Kbytes/s, so there will be a need for storage device speed-up and video capture down-sampling. The FUSB2805 transceiver can be soldered into the ALFAT SoC to increase the data rate transfer speed. On a FUSB2805, data can be transmitted and receive at high speed (480Mbps), full speed (12Mbps) and low speed (1.5Mbps) through a 12-bit (SDR) interface. In other words, 12Mbps is an achievable speed and data rate design calculations use this rate as a bench mark.

In this project, we use the MAX9526 as our ADC to convert Analog Video into digital signal. The MAX9526 is a 10-bit 4x oversampling (54Msps) ADC with true 10-but digital processing. To decrease bandwidth because of storage limitations, however, we will down-sample the video signals twelve times. We down-sample by 4 to achieve a video rate of 30 fps. After that, we further down-sample by 3 to achieve 10 fps. In other words, we will sample at a rate of,

$$\frac{54\ \text{Msps}}{12} = 4.5\ \text{Msps}$$

The FPGA gets the digital video signal from at this rate from the MAX9526 ADC, processes it, overlays it with the data obtained from the OBDII, and sends it to the ADV212 encoder. The ADV212 has the following data input rates:

Table 23. Maximum Pixel Data Input Rates (121-Ball Package)

Interface	Compression Mode	Input Format	Input Rate Limit Active Resolution (MSPS)[1]	Approx Min Output Rate, Compressed Data[2] (Mbps)	Approx Max Output Rate, Compressed Data[3] (Mbps)
HDATA	Irreversible	8-bit data	34	98	150
	Irreversible	10-bit data	34	98	150
	Irreversible	12-bit data	34	98	150
	Irreversible	16-bit data	34	98	150
	Reversible	8-bit data	30	98	150
	Reversible	10-bit data	24	98	150
	Reversible	12-bit data	20	98	150
	Reversible	14-bit data	17	98	150
VDATA	Irreversible	8-bit data	48	98	150
	Irreversible	10-bit data	48	98	150
	Irreversible	12-bit data	48	98	150
	Reversible	8-bit data	30	98	150
	Reversible	10-bit data	24	98	150
	Reversible	12-bit data	20	98	150

Figure 37: Maximum Pixel Data Input Rates for 121-Ball Package (ADV212 Datasheet, p. 32)

According to the table, for a VDATA irreversible 10-bit data, with an input of 48 Msps, it is guaranteed to have the output rate in the range of [98, 150] Mbps.

With our input coming in at a rate of 4.5Msps, the ratio of the actual input rate to the input rate limit is,

$$\frac{4.5}{48} = 0.09375$$

This gives us a new range of output rate of,

$$[98, 150]\text{Mbps} * 0.09375 = [9.2, 14]\ \text{Mbps}$$

OBDII Parameters:

The OBDII sends one or two bytes of information which then need additional calculations to determine the correct values of the parameters. These are done with the following equations, where A denotes the first byte of data and B denotes the second, if there is a second data byte.

Temperature:
    Data = A – 40 (degrees C)
Fuel Pressure:
    Data = A * 3 (kPa)
Engine RPM:
    Data = ((A * 256) + B) / 4 (rpm)
Vehicle Speed:
    Data = A (km/hr)
Throttle Position:
    Data = (A * 100) / 255 (%)

III. Requirements and Verification

3.1 Requirements

3.1.1 Requirements Summary:

1) **Input Module:** This module should successfully obtain the captured video from the camera and convert it into digital form using an ADC. Besides, this module also sends the OBD II data to the FPGA and control module without errors.
2) **FPGA and Control Module:** This module should successfully overlay the data obtained from the OBD II on the video received from the input module.
3) **Storage Module:** This module should successfully store the overlaid video without corrupting the file system at a reasonable frame rate.
4) **Display Module:** This module should display the overlaid video without any glitches or freezes.
5) **Power Module:** This module should supply enough steady power to the circuit.

3.1.2 Verifications Summary

1) **Input Module:** The video input can be tested simply by displaying the image from the camcorder onto a screen. The latest OBD II information will be stored in a register, so we can display the contents of the register onto the screen to be sure they are correct. If possible we will use an OBD II simulator to verify our measurements.
2) **FPGA and Control Module:** This can be verified by checking the information from the OBD II in relation to what is present on the screen. Since most of the information needs to have a calculation done to it, we will have to make sure that this calculation is done accurately.
   The signal inputs and outputs will be verified with a logic analyzer to ensure accuracy.
3) **Storage Module:** We will do a simple check to make sure that both of our videos (the raw video and the video with the overlay) are stored properly by playing them back off of the storage device. One method to check this is by displaying the videos through a computer and the other is local playback.
4) **Display Module:** We will verify that the real-time video is displayed on our screen.
5) **Power Module:** We will do a simple check to see if the power supply supplies the expected voltage and that the circuit stays powered on during an extended period of time.

3.1.3 Input Module

Requirements	Verifications
Camcorder: 1) Output video signal is analog SD NTSC-M signal	Camcorder: 1) Ensure analog source, the camcorder, is powered on and output is enabled by: a. Pressing 'ON' button on camera b. Default video setting is NTSC-M c. Connect minijack-to-RCA cable to camera video output d. Connect RCA output to the one RCA input connector on main circuit board e. Set scope probe to High-Z (1MOhm or higher) f. Press probe connector onto RCA connector center (signal) wire g. View scope pattern – an NTSC-M characteristic signal should be present on screen.
MAX9526: 2) Output video is ITU-R.656 uncompressed digital video	MAX9526: 2) Check ADC chip power: a. Connect three (3) oscilloscope probes' local GND to board GND (pin 23) closest to ADC GND pin b. Attach a probe to ADC power pin VAVDD (pin 5). Voltage should be +1.8V nominal. Voltage should not exceed 2V. Voltage should not be less than 1.5V c. Attach a probe to ADC power pin VDVDD (pin 22). Voltage should be +1.8V nominal. Voltage should not exceed 2V. Voltage should not be less than 1.5V d. Attach a probe to ADC power pin VDVDDIO (pin 24). Voltage should be +3.3V nominal. Voltage should not exceed 3.3V. Voltage should not be less than 3.0V
3) Ensure that the ADC chip is powered at the correct wattage	3) Input format settings a. Connect Logic Analyzer Ground to GND closes to ADC DGND (pin 23) b. Connect Logic Analyzer probe pin 1 to DEVADDR (pin 9) c. Connect another Logic Analyzer probe pin 2 to SDA (chip pin 12) d. Connect a third logic analyzer probe #3 to ADC SCL pin (#13) e. Connect 4th Logic Analyzer probe to ADC pin 14 and monitor NOT(IRQ) f. Set Logic Analyzer to record

	g. Reset FPGA (press reset button and record data h. Use the Logic Analyzer to ensure that the FPGA sends the correct signals to read from register x0D, the clock and output control register, and that the data read is x04, meaning the vertical and horizontal syncs are enabled.
4) Configure the MAX9526 correctly	4) Enable the horizontal and vertical syncs a. Set up the Logic Analyzer using the above instructions. b. Using the SLA and SLD signals, call upon the x0D register, which controls the output and the clock. Then write x04 to the register to enable the horizontal and vertical sync signals. Use the Logic Analyzer to ensure that this is done correctly.
<u>WIG-09555:</u> 5) Input data signal is OBDII data from car	<u>WIG-09555:</u> 5) Ensure car is powered on: a) Insert key into automotive ignition b) Turn key into Start c) Release key and allow car to idle
6) OBDII to ELM signaling is set to ISO 9141-2 for Asian make vehicles	6) Check automotive accessory power outlet for 12V: a. Turn multimeter on b. Place multimeter negative lead to outer shell of automotive accessory outlet (GND) c. Place positive multimeter lead to center pin of automotive accessory adapter d. Reading should be 12V nominal, voltages from 11.5V to 12.5V are acceptable. –AJW verify
7) Output data is OBDII info over RS-232 interface	7) Verify OBDII ISO 9141-2 signaling scheme is selected by: a. Connect OBDII GND and pins to headers on Logger board b. Connect Logic Analyzer Ground to OBDII GND c. Connect Logic Analyzer probes to the RS-232 Tx and Rx signals connecting the WIG-09555 and the FPGA. d. Use the logic analyzer to make sure OBDII module is powered on and is set correctly. Store sent information in a register, and use the hex display on the FPGA to

	check that the correct data is received and stored

3.1.4 FPGA and Control Module

Requirements	Verifications
FPGA: 1) Module passes ITU.656 video with overlay to ENCODER/DECODER chip during record	FPGA: 1) Ensure FPGA is powered on by the +1.2V power power supply and +3.3V for signal voltage levels by: a. Connect Multimeter Ground to onboard ground b. Connect Multimeter positive connector to VCCINT test pad (Test pads will be added during PCB Layout process) c. Read voltage – it should be 1.2.V nominal, minimum of 1.15V, maximum of 1.25V d. Iterate a-d for each test pad e. Connect Multimeter positive connector to FPGA VCCIO test pads (Test pads will be included in PCB Layout) f. Voltage should read 3.3V nominal. Minimum voltage is 3.135V. Maximum is 3.6V g. Iterate steps e-f for each test pad h. Connect multimeter positive probe to VCCA-PLL test pads (Pads will be included in PCB layout) i. Voltage should read 1.2V nominal, 1.15V minimum, 1.25V maximum
2) Module overlays OBD II data onto ITU.656 stream	2) Ensure digital timings and internal character mappings are at standard. Timing will be ensured by using a 100 to 200MHz clock driver, as well as the FPGA's internal 20MHz clock for configuration. Test timing signal by: a. Set oscilloscope probe input to High-Z (~1MOhm or greater) b. Place oscilloscope probe on timing chip output pin c. Confirm frequency d. Change clock chip settings
3) Module takes in OBD II data.	3) Ensure FPGA configures OBD II circuit

	module for correct operation. We can also test the individual OBDII UART board with a terminal program to ensure that it is receiving and transmitting data correctly. Test module by: a. Connect a Logic Analyzer to circuit ground b. Connect Logic Analyzer probe pins to UART signals. c. Test for signal transmission and patterns
4) Module passes compressed JPEG2000 stream to storage module during record	4) Ensure FPGA is set to route video to the storage device. We can then test that the video can be opened and viewed directly from said device Testing is done with: a. Connect logic analyzer to FPGA-to-ALFAT transmission traces
5) Module passes compressed JPEG2000 video from storage module to JPEG2000 ENCODER/DECODER chip during playback	5) Ensure FPGA is set to route video from ENCODER/DECODER module and has device settings configured properly a. Connect Logic Analyzer probes to HDATA and VDATA lines. Video data and configuration data should be present b. Connect Logc Analyzer to ADV212 configuration signal lines. Chip should be enalbled
6) Module processes decompressed JPEG2000 to ITU.656 video and passes it to display module during playback	6) Ensure FPGA is set to route video to the display, a. Trace each TFT connector with oscilloscope probe. Voltages should be LVCMOS +3.3V levels b. Connect Logic Analyzer to TFT signal pins. Test for video signal c. Qualitatively observe video stream on screen.
<u>Switches:</u> 7) User selects video playback and record functions such as (stop, play, record, next file, previous file)	<u>Switches:</u> 7) Test for switch power connectivity by: a. Place multimeter negative lead to switch ground pin 3 b. Place multimeter positive probe to switch power of 3.3V at pin 1 c. Iterate for each switch 8) Test for switch functionality: a. Place multimeter negative lead to switch ground on pin 3 b. Place multimeter positive lead to switch output on pin 2 c. Toggle switch to “OFF” – output should read 0V d. Toggle switch to “ON” – output

	should be 3.3V e. Iterate for each switch 9) Test for switch connection to FPGA: a. Since we cannot probe electrical connection from switch output to FPGA input, test is a software stimulus b. Set FPGA switch register to ON/OFF state for switch function [play, stop, etc] c. Compile code and check output d. If output changes with respect to software change and not switch state, switch is not connected e. If output does not change with respect to software stimulus change, software is broken. f. Iterate step 9 for each switch.
<u>Boot loader/Digilent Basys2:</u> 10) Ensure power inputted is correct	<u>Boot loader/Digilent Basys2:</u> 10) Use a multimeter to verify that the input voltage is between 3.135 and 3.465 V and that it draws to more than 150 mA of current
11) The Basys2 Development Board powers on	11) The Basys2 Development Board is supplied with power from a USB connector or an external 3 AA batteries giving it a 4.5V power supply a. A multimeter will be used to check if the batteries are supplying 4.5V ± 0.15 V
12) The AS (active-serial) scheme is selected for configuration	12) Ensure the digital logic for pins MSEL [ ] is 00 for the 20MHz AS configuration a. A logic probe will be used to check the digital logic of the MSEL [ ] pins
13) The FPGA and the Basys2 (used as bootloader) are on when being configured	13) Connect the pull-up resistors to a 3.3-V ± 0.15 V supply a. A multimeter will be used to measure the voltage and the resistance
14) The FPGA goes through all 3 states (RESET, CONFIGURATION, INITIALIZATION) before entering USER MODE	14) Ensure each state completes successfully before going to the next state by checking the appropriate pin signals a. A logic probe will be used to check the logic signal of the nSTATUS pin. If it is HI, the RESET state has been completed b. A logic probe will be used to check the logic signal of the

	CONF_DONE pin. If it is HI, the CONFIGURATION state has been completed c. A logic probe will be used to check the logic signal of the INIT_DONE pin. If it goes form LOW to HI, the INITIALIZATION state has been completed
15) The Basys2 Development Board downloads the program to the FPGA board	15) Ensure the Basys2 Development Board can communicate with the FPGA Board and vice versa by initiating test hand-shake programs a. The ECE385 Altera DE2 Board will be used to test the program from the bootloader. If the bootloader successfully downloads the program to the Altera DE2 board, it should also be able to download program to the PCB board b. The Altera DE2 Board can also be used as a debugging tool to see if the configuration data has been sent to the PCB or not.
16) The FPGA board goes to USER MODE successfully	16) The FPGA board goes to USER MODE successfully. The user can use the program downloaded to the FPGA board from the Basys2 Development Board a. A "Hello World" user program will be written to check if the PCB goes to the user mode or not

3.1.5 Storage Module

Requirements	Verifications
TS16GSDHC10E: 1) Video stored in computer readable format	TS16GSDHC10E: 1) Ensure FPGA commands sent to storage SoC properly set-up file saves and the file saved in the standard video format a. We'll use the ECE385 Altera DE2 board to display and check the commands sent to the storage device

	b. To check if the file is readable, a PC will be used later to check if the file can be opened and played
2) Video read from device is realtime JPEG2000 format	2) Ensure FPGA control commands sent to storage SoC properly set-up file reads provided condition in step 1 is met. a. After checking that file save is successful, a PC will be used to analyze the format of the video file, bit rates of the video and the text file b. A PC will be used to ensure that the video file is in the JPEG2000 format c. The text file which stores the information from the OBD II will also be read on a PC to make sure that the data is correct and the file is readable
3) The user can playback the video stored on the storage media device at a time later	3) The video file on the storage media can be played on a PC or other devices a. If file saves and file reads are successful, the video file should be playable on a PC b. The text file should also be readable on a PC
<u>ALFAT OEM Board:</u>	<u>ALFAT OEM Board:</u> <u>Note:</u> The ECE385 Altera DE2 Board will be used to test the storage device individually (specifically, to check if the codes sent to the ALFAT OEM Board is in correct format, and to read the error codes returned from the ALFAT OEM Board). It won't be included in the final product. Rather, it will be used as a debugging tool.
4) The ALFAT OEM Board powers on	4) Ensure the ALFAT OEM Board is powered on with a 3.3V power source a. Use a multimeter to check that the ALFAT OEM Board is supplied with 3.3V ± 0.15V b. Use a oscilloscope to check if the 3.3V ± 0.15 V voltage supply is steady
5) The media device (SD card) receives enough power and works reliably	5) The media may not work on an unstable source. Make sure the power source to the storage media is reliable and there is a large enough capacitor as close as possible to the media power pins. 0.1uF and 22uF capacitors are recommended

	a. Use a oscilloscope to make sure that the supplied power voltage is steady at 3.3V ± 0.15 V (as of part 4) b. Use a multimeter or a capacitance meter to measure and make sure that the recommended capacitors 0.1uF and 22uF at the media power pins
6) The ALFAT OEM board receives the commands form the FPGA board successfully without error	6) Ensure the returned error codes are checked by the user every time a command is sent. a. The Altera DE2 will be used to check the returned error codes b. The DE2 board interfaces with the ALFAT OEM DE2 board and receives signals from it similarly to what has been done in ECE385 with the keyboard interface c. The returned error codes will be displayed on the LEDs on the DE2 board for verifications
7) The ALFAT OEM Board receives separate commands from the FPGA board and stack them up if needed	7) Make sure commands are terminated by line-feed a. The Altera DE2 board will be used to display and check the commands sent to the ALFAT OEM Board, and check if each command terminates with a line-feed b. When sending the commands, successive commands will be sent, and the error codes will be checked using the Altera DE2 board to see if the stacked up commands are received and completed by the ALFAT OEM board one command at a time
8) The ALFAT OEM Board can communicate with the FPGA Board. Data can be sent out to and received from the FPGA board	8) Ensure the UART_TX signal send data out from ALFAT UART_RX receive data to ALFAT Ensure no more data is sent when the UART_BUSY signal is high a. The Altera DE2 Board will be used to test this. The UART_TX, UART_RX, and the UART_BUSY signals are hooked to the DE2 Board so that those signals can be read and displayed by Altera DE2 Board b. The Altera DE2 board will display

	the UART_TX and UART_RX signals using the 7-segment LEDs or the on-board LCD display c. The Altera DE2 board will display the UART_BUSY signal using an on-board LED
9) The ALFAT OEM Board initialize and mount the storage device successfully	9) 0x00 (successful command) is sent back as error code for INITIALIZE instead of 0x10 or 0x11 (Initialize media failed) a. The Altera DE2 board will be used to check if UART_TX sends out 0x00 from the ALFAT OEM board for a successful command b. The Altera DE2 board will also be used to check the UART_RX sends out a correct INITALIZE command is sent out successfully to the ALFAT OEM Board device
10) The ALFAT OEM Board Opens a new file successfully for read	10) 0x00 (successful command) is sent back as error code for OPEN instead of 0x21 (Failed to open the file). INITIALIZE must be called first before OPEN a. The Altera DE2 board will be used to check if UART_TX sends out 0x00 from the ALFAT OEM board for a successful command b. The Altera DE2 board will also be used to check the UART_RX sends out a correct OPEN command is sent out successfully to the ALFAT OEM Board device
11) The ALFAT OEM Board Writes to file successfully	11) 0x00 (successful command) is sent back as error code for WRITE instead of 0x30-0x37 (file handle invalid). Wait to get acknowledge after sending the W command before actually sending the file. Ensure the sent data size matches the size declared in the command. Make sure the file is FLUSHed or CLOSEd when done a. The Altera DE2 board will be used to check if UART_TX sends out 0x00 from the ALFAT OEM board for a successful command b. The Altera DE2 board will also be used to check the UART_RX sends out a correct WRITE command is sent out successfully to the ALFAT OEM Board device
12) The ALFAT OEM Board Flushes or Closes file successfully	12) 0x00 is returned for CLOSE or FLUSH command a. The Altera DE2 board will be used

	to check if UART_TX sends out 0x00 from the ALFAT OEM board for a successful command b. The Altera DE2 board will also be used to check the UART_RX sends out a correct CLOSE or FLUSH command is sent out successfully to the ALFAT OEM Board device

3.1.6 Display Module

Requirements	Verifications
ADV212: 1) Displays decompressed video equivalent to capture rate 2) Video output size maximum of 800X480 resolution 3) Configure the ADV212 correctly 4) Ensure that the ADV212 is powered correctly 5) ADV212 must be initiated properly	ADV212: 1) Use an oscilloscope to ensure that the crystal used to configure the interface clock is oscillating at 27MHz. 2) Ensure JPEG2000 video is decompressed and scaled properly; ensure the whole video is on the screen by comparing it with the camcorder. 3) This can be checked by communicating with the chip through a CPU via a free software codec such as Kakadu and ensuring that the chip is operating in single component mode. 4) Power settings a. Check that V_{DD} (pins 3, 8, 40, 84, 120) is 1.5V ±0.175V by using a multimeter. b. Also, we must check that V_{DDIO} (pins 17, 28, 30, 38, 52, 74, 82, 93, 104, 105, and 106) is 3.3V ± 0.165V. 5) Initializing the chip a. Connect the first Logic Analyzer probe to ground. b. Connect the Logic Analyzer to the 4 address signals (pins 87, 88, 96, 97, and 107). c. Using the Logic Analyzer, first ensure that 0x400 is written to the EIRQIE at address 0x5. Then, the IRQ pin will go low, and we will check that EIRQFLG[10] is set using the FPGA and a simple code. We will also use the FPGA to read the application ID to ensure that the chip was correctly initialized.

K50DWN0-V1-F: 6) Ensure the correct power wattage is being sent to the display	K50DWN0-V1-F 6) Power Settings a. Use a multimeter to ensure that V_{CC} is between -0.3 and 5.0V and I_{CC} is between 25 and 35 mA. b. Also, V_{DD} must be between 3.0 and 3.6 V and I_{DD} has to be between 15 and 19 mA.

3.1.7 Power Module

Requirements	Verifications
Power: 1) Supply enough voltage to power on the board	Power: 1) Car outlet is live. A multimeter will be used to check if the voltage output from the car is 12V
2) Supply constant power to the board	2) The board doesn't power off intermittently. An oscilloscope will be used to check if the voltage signal is steady.
3) The car voltage is stepped down correctly to 5V, 3.3V, 1.8V, 1.2V and 1.5V	3) A multimeter will be used to check the voltage output of each DC/DC converter to see if 5V, 3.3V, 1.8V, 1.2V and 1.5V are achieved. The error tolerance should be about ± 0.15V
4) All the individual components power on and work consistently	4) Ensure each component is connected to its correct voltage supply by checking the voltage before connecting it to a component.

3.2 Tolerance Analysis

Oscillator Frequencies: Verify that the ADC oscillator crystal has a frequency of 27 MHz, plus or minus 50 ppm. We can verify this using either an oscilloscope, an ADC output with logic analyzer, or a frequency analyzer. We need to have oscillator frequency at 27MHz so that the ADC can lock onto an input NTSC video signal.

3.3 Ethical Issues

The purpose of this project is to develop a data logger device for racer, which helps them to better manage the condition of the car while racing. With such function, our device helps increase the safety and health of the driver, which is consistent with the first code of the IEEE Code of Ethics:

1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

One of the factors that cause car accidents is system defects. Our device helps prevent these kinds of accidents by giving drivers immediate information about the system s other car while also them with video of a driving car.

Throughout the development of the device, we will follow the third code closely, and only make claims and estimates based on real data acquired from our data logger.

3. to be honest and realistic in stating claims or estimates based on available data;

Working with a data logger device, the most importance factor is the accuracy of the information obtained from the system. We will be honest and will not falsify the data acquired from our test procedures.

After this project, we will have learned a great deal about various real-world industrial systems such as the On-board Diagnostic (OBD II) system, the SD Flash memory FAT 32 file system, or the ITU-R BT 656 protocol. This will improve our understanding of these technologies and their applications, and also improve our technical competence, as directed in the 5th and 6th codes of the IEEE Code of Ethics:

5. to improve the understanding of technology; its appropriate application, and potential consequences;

6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

Furthermore, while working on the project, we will build an environment that promote engineer professionalism, which welcomes constructive and honest criticisms, acknowledges errors, assists peer workers with their professional and academic developments, and credits appropriate contributions, as cited in the 7th and 10th codes of the IEEE Code of Ethics:

7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

Following the 9th code of the IEEE Code of Ethics,

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

We will make sure that the data logger device developed from this project will offer the users with data that is as accurate as possible, and will not provide them with false information, in order to avoid damaging that could occur to the users' health, their car or other properties.

IV. Cost and Schedule

4.1 Cost Analysis

a. Labor

Name	Rate	Hours	Total	Total x 2.5
Tung Do	$50/hr	180	$9000	$22500
Nick Greenway	$50/hr	180	$9000	$22500
Andrew Wesly	$50/hr	180	$9000	$22500
Total				$67500

b. Parts

Description	#	Manufacturer	Vendor	Cost/unit	Total Cost
ADV212BBCZRL-150 JPEG2000 Encoder/Decoder	1	Analog Devices	Digikey	$48.984	$48.984
Development Board	1	Altera DE2	NA	$269	0 - Have
ELM327 OBD Interpreter	1	Elm Electronics	ELM Electronics	$23.50	$23.50
MAX9526 Video Decoder	1	Maxim Integrated	Mouser	$7.97	$7.97
OBD-II UART	1	Sparkfun Electronics		$49.95	$49.95
Control Switches	6	C&K Compnents	Mouser	$6.25	$37.50
ALTERA EP2C35F672C6 FPGA	1	Altera	Altera	$149.50	$149.50
Digilent Basys2 Development Board – used as bootloader	1	DigilentInc	DigilentInc	$100.00	0 - Have
ALFAT-SD-337 – FAT32 SD card writer	1	GHI Electronics	GHI Electronics	$59.95	$59.95
K50DWN2-VI-FE – TFT Display	1	Kentec	Newark	$65.22	$65.22
10085901-6015elf– TFT Display Connector	1	FCI	Mouser	$1.46	$1.46
Transcend 16 GB SDHC Class 10 Flash Memory Card TS16GSDHC10E	1	Transcend	Amazon	$11.92	0 – Have
RC55LF-D-10K-B-B : Metal Film Resistor 1/4W 10K 0.1%	2	TT Electronics	Mouser	$1.32	$2.64
PR01000103300JR500 : Metal Film Resistor 1watt 330ohms 5%	3	Vishay	Mouser	$0.34	$1.02
LT CN5M-FBGB-25-	2	OSRAM	Mouser	$3.25	$6.50

1-Z: High Power LED - Green 1/2 Watt 528nm, 140mA					
173D105X9025UWE3 : Tantalum Capacitor, 1uF 25V 10% Axial	5	Vishay	DigiKey	$1.15	$5.75
DE09P064HTXLF: D-Sub Standard Connector 9P PIN SOLDER BUCKET	1	FCI	Mouser	$0.90	$0.90
1531A2L12: Toggle Switches SPDT ON-OFF SOLDER	18	Apem	Mouser	$3.42	$61.56
78SR-5/2-C: DC/DC Converters 8-32Vin 5Vout 2A SIP Switching Reg.	1	Murata	Mouser	$12.36	$12.36
NDTS0503C: DC/DC Converters 5Vin 3.3Vout 909mA Isolated 3W 24 pin	1	Murata	Mouser	$16.95	$16.95
LSN-1.8/16-W3-C: DC/DC Converters 28W 5V to 1.8V 16A	1	Murata	Mouser	$18.75	$18.75
LSN-1.2/10-D5-C: DC/DC Converters 12W 5V to 1.2V 10A	1	Murata	Mouser	$21.13	$21.13
PCB – All boards	1	PCBFABEXPRESS	PCBFABEXPRESS	$200	$200
Total					$791.59

Total Labor Cost	**Total Parts Cost**	**Total Project Cost**
$67,500	$791.59	**$68,291.59**

4.2 Schedule

Week	Task Description	Group Members
9/16	**Proposals Due**	
	Software design for storage and boot loader	Tung Do
	Software design for OBD II and ITU-R BT. 656	Nick Greenway
	Electronic hardware specifications/designs	Andrew Wesly
9/23	**Design Reviews Sign-up Closes**	
	Software development for storage interface	Tung Do
	Software development for OBD II interface	Nick Greenway
	Finalize the schematic	Andrew Wesly
9/30	**Design Reviews**	
	Test storage interface	Tung Do
	Test OBD II interface	Nick Greenway
	Board layout/order components	Andrew Wesly
10/7		
	Software development for Boot loader	Tung Do
	Software development for Video interface	Nick Greenway
	Submit board for fab	Andrew Wesly
10/14		
	Test boot loader	Tung Do
	Test video interface	Nick Greenway
	Assemble the boards	Andrew Wesly
10/21	**Individual Progress Reports Due**	
	Test overlaid video on storage device	Tung Do
	Verify correctness of OBD II data obtained	Nick Greenway
	Test assembled boards	Andrew Wesly
10/28		
	Assemble boot loader	Tung Do
	Assemble video	Nick Greenway
	Power board layout& Assemble FPGA	Andrew Wesly
11/4	**Mock-up Demos and Mock Presentation Sign-up closes**	
	Mock Presentation – Control Slides	Tung Do
	Mock Presentation – Test Slides	Nick Greenway
	Mock Presentation – Design Slides	Andrew Wesly
11/11	**Last Day to Request 1st Revision PCB Fabrication**	
	Final software testing	Tung Do
	Final software testing	Nick Greenway
	Assemble everything in an enclosure	Andrew Wesly
11/18	**Thanksgiving Break, Last day to Request Final Revision PCB fab**	
11/25	**Demo and Presentation Sign-up closes**	
	Final tests and verifications	Tung Do
	Final tests and verifications	Nick Greenway
	Final tests and verifications	Andrew Wesly

12/2	**Demos and Presentations**	
	Demo, Final Paper – Intro & Conclusion	Tung Do
	Presentation, Final Paper – Testing & Verification	Nick Greenway
	Demo, Final Paper – Design Procedure	Andrew Wesly
12/9	**Presentations, Checkout, Final Paper, Lab Notebooks**	
	Final Paper & Notebook review and hand-in	Tung Do
	Final Paper & Notebook review and hand-in	Nick Greenway
	Final Paper & Notebook review and hand-in	Andrew Wesly

4.1 Contingency Plan

- In case of exorbitant pcb costs or a crunch for time the following contingency plan may be executed to deliver on proposed Automotive Racing Video Data Logger.

- Replace FPGA IC, MAX9526 ADC IC, RS232 MODULE, and CONTROL switches with Altera DE2 board.

- Modify power supply module to supply 9V to development board

- Connect ALFAT module to board, OBDII module to board

- Create TFT display connections

- Possibly implement JPEG2000 encoder/decoder.

- Frame rates may be as low as 1fps

References:

78SR 2 Amp Series, Murata, Mansfield, MA, 2009. [Online]. Available: http://www.murata-ps.com/data/meters/mpm_78sr-2a_a00.pdf

ADV212, Rev. B, Analog Devices, Norwood, MA, 2010. [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADV212.pdf

ALFAT SoC Processor User Manual, Rev. 1.12, GHI Electronics, Macomb Township, MI, 2012. [Online]. Available: http://www.ghielectronics.com/downloads/ALFAT/ALFAT%20SoC%20Processor%20User%20Manual.pdf

BT.656 Video Interface for ICs, Intersil, Milpitas, CA, 2002. [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/an97/an9728.pdf

Charles A. Poynton, *A Technical Introduction to Digital Video*, John Wiley & Sons, Inc., 1996, p. 175-176

Cyclone II Device Handbook, Volume 1, Altera, San Jose, CA, 2008. [Online]. Available: http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf

Cyclone II EP2C35 PCI Development Board Reference Manual, Version 1, Altera, San Jose, CA, 2005. [Online]. Available: http://www.altera.com/literature/manual/rm_pci_dev_bd_cyclone2_ed.pdf

Digilent Basys 2Board Reference Manual, Digilent, Pullman, WA, 2010. [Online]. Available: https://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf

ELM327, ELM, Toronto, [Online]. Available: http://elmelectronics.com/DSheets/ELM327DS.pdf

FS-K50DWN0-V1-F, Rev. 2, Kentec Display, 2011. [Online]. Available: http://www.kentecdisplay.com/uploads/soft/Products_spec/K50DWN0-V1-F-01_KED_.pdf

IEEE Code of Ethics [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

MAX9526, Rev. 3, Maxim Integrated, San Jose, CA, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX9526.pdf

MPDTY Series, Murata, Mansfield, MA, [Online]. Available: http://search.murata.co.jp/Ceramy/image/img/A14X/M07E1.pdf

NDTS Series, Murata, Mansfield, MA, 2012 [Online]. Available: http://www.murata-ps.com/data/power/ncl/kdc_ndts.pdf

OBD-II UART [Online]. Available: https://www.sparkfun.com/products/9555

Recommendation ITU-R BT.601-5: Studio Encoding Parameters of Digital Television for Standard 4:3 and Widescreen 16:9 Aspect Ratios, 1995. [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/an97/an9728.pdf

Single Output LSN-10A Models, Murata, Mansfield, MA, 2009. [Online]. Available: http://www.murata-ps.com/data/power/lsn10a-d5.pdf

Single Output LSN-W3 Models, Murata, Mansfield, MA, 2009. [Online]. Available: http://www.murata-ps.com/data/power/lsn16a-w3.pdf