Making a Simple Graphic Card & a Simple Sound Card

Team 1

2018707082 강지원
2018707065 고종완
2018707048 신동호

KWANGWOON UNIVERSITY
전자통신공학과

Department of Electronics & Communications Engineering, KWANGWOON University

Contents

1 Objective

2 Role Division

3 Time Line

4 Demonstration Order

- Simple Graphic Card
- Simple Sound Card
- Software Development

5 Summary of the Project

6 Final Test

7 Demo

8 Further Improvements

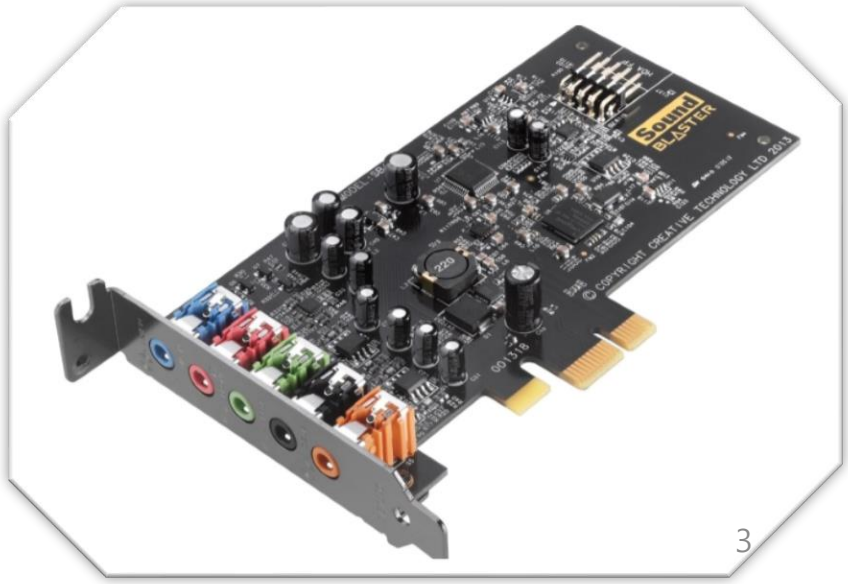
9 Cost

10 References

1 Objective

1. Design a elementary graphic card and print a picture on the monitor screen.

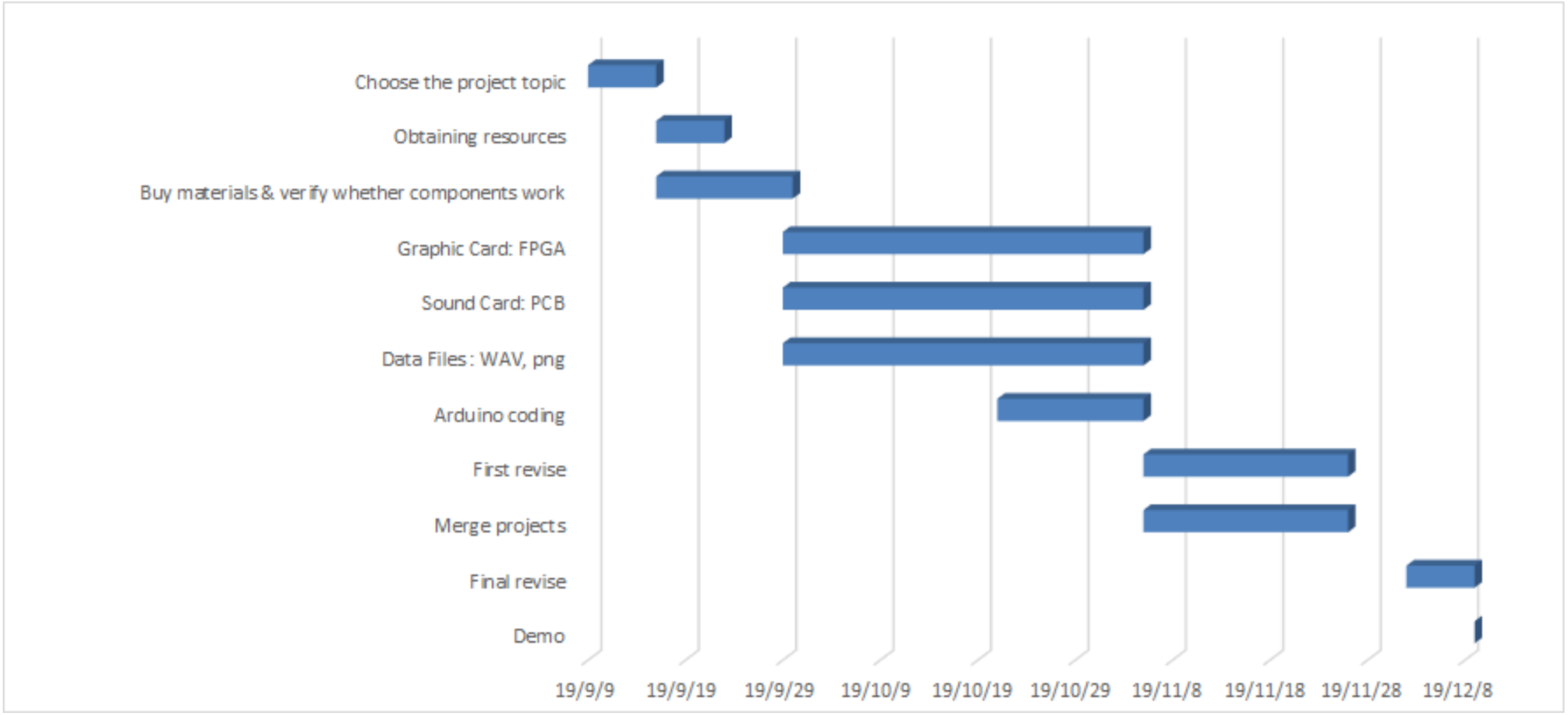
2. Design a elementary sound card and make a sound related to the picture shown on the monitor.



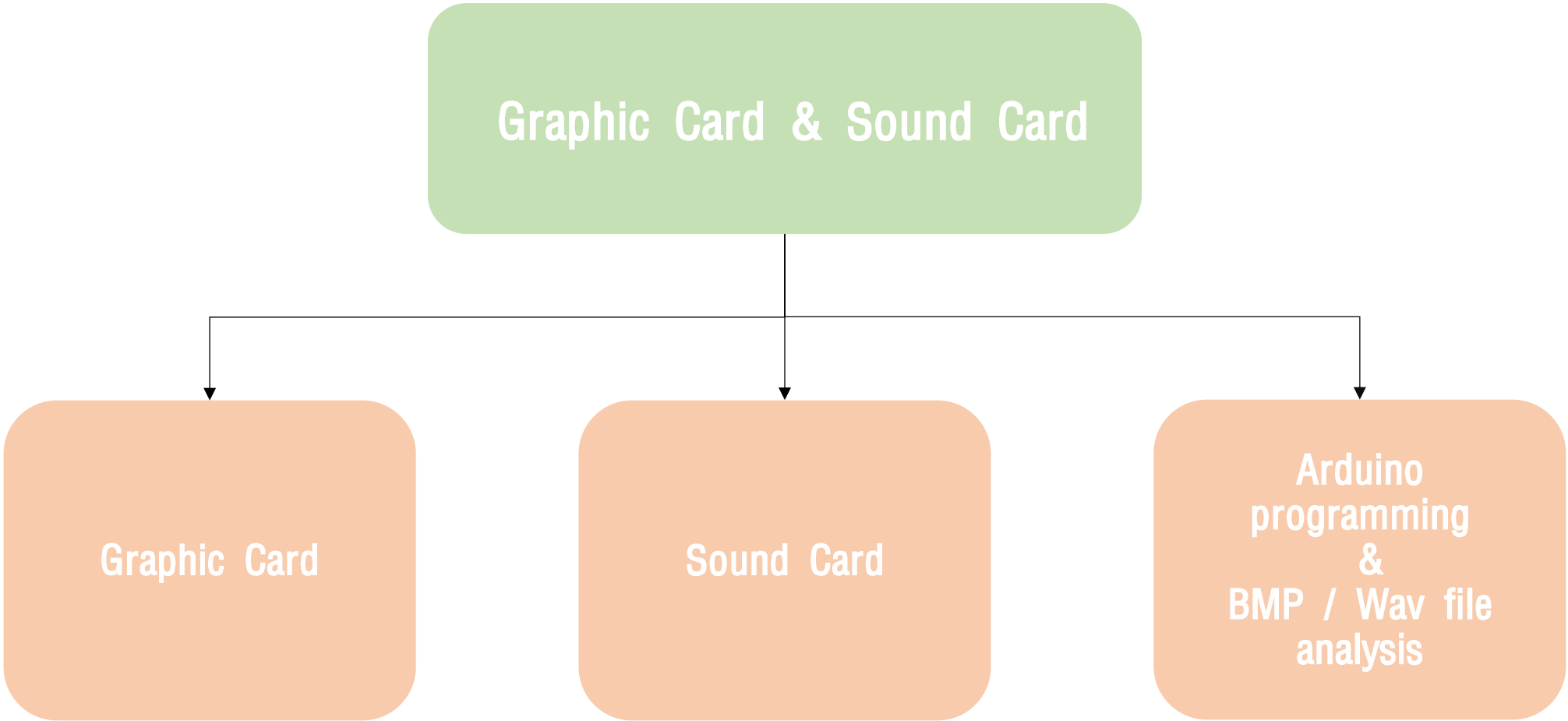
2 Role Division

Name	Role
강지원	1. Design a simple sound card using analog circuits. 2. Design a protocol for communications between graphics card and sound card. 3. PCB board design.
고종완	1. Design a graphic card capable to print 200 * 600 pixel picture using FPGA. 2. Design a protocol for communications between graphics card and sound card.
신동호	1. Decompose WAV & PNG files. 2. Manipulate data for graphic card and sound card. 3. Arduino programming

3 Time Line


Choose the project topic
Obtaining resources
Buy materials & verify whether components work
Graphic Card: FPGA
Sound Card: PCB
Data Files : WAV, png
Arduino coding
First revise
Merge projects
Final revise
Demo
19/9/9
19/9/19
19/9/29
19/10/9
19/10/19
19/10/29
19/11/8
19/11/18
19/11/28
19/12/8

4 Demonstration Order



Simple Graphic Card

VGA Timing

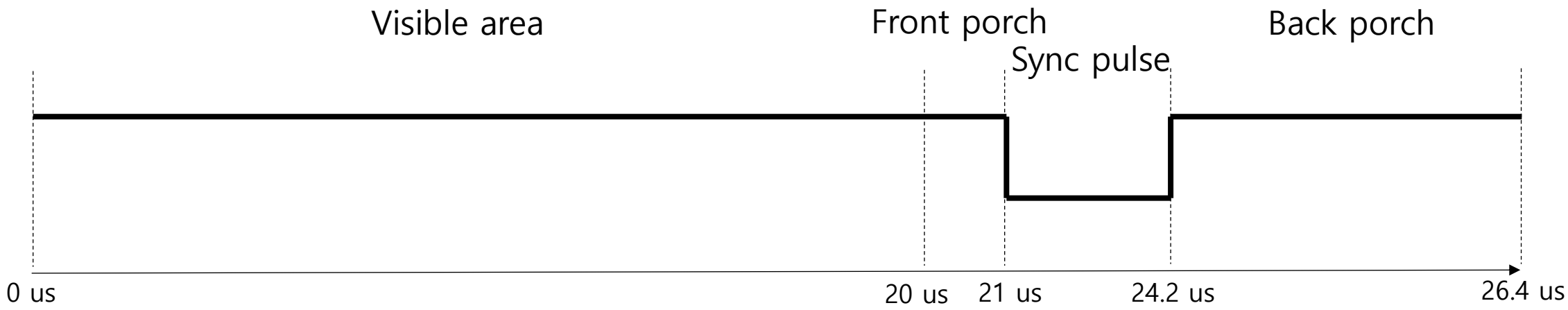
General timing	
Screen refresh rate	60 Hz
Vertical refresh	37.878787878788 kHz
Pixel frequency	10.0 MHz

Horizontal timing (Line)			Vertical timing (Frame)		
Scanline part	Pixels	Time [us]	Frame part	Lines	Time [ms]
Visible area	200	20	Visible area	600	15.84
Front porch	10	1	Front porch	1	0.0264
Sync pulse	32	3.2	Sync pulse	4	0.1056
Back porch	22	2.2	Back porch	23	0.6072
Whole line	264	26.4	Whole frame	628	16.5792

VGA Signal 800 x 600 @ 60Hz timing

**Actually 800 x 600, because of memory shortage we only use 200 pixels*

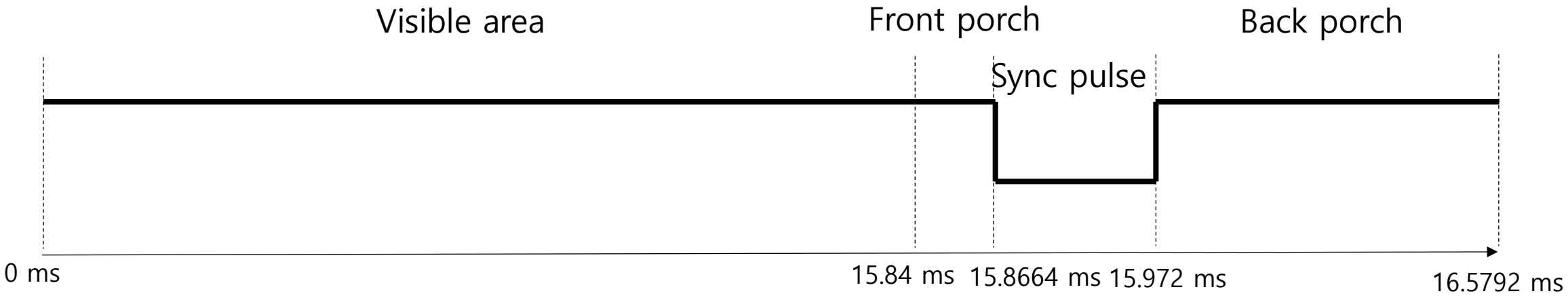
VGA Timing (Cont.)



Horizontal timing (Line)		
Scanline part	Pixels	Time [us]
Visible area	200	20
Front porch	10	1
Sync pulse	32	3.2
Back porch	22	2.2
Whole line	264	26.4

Horizontal Timing

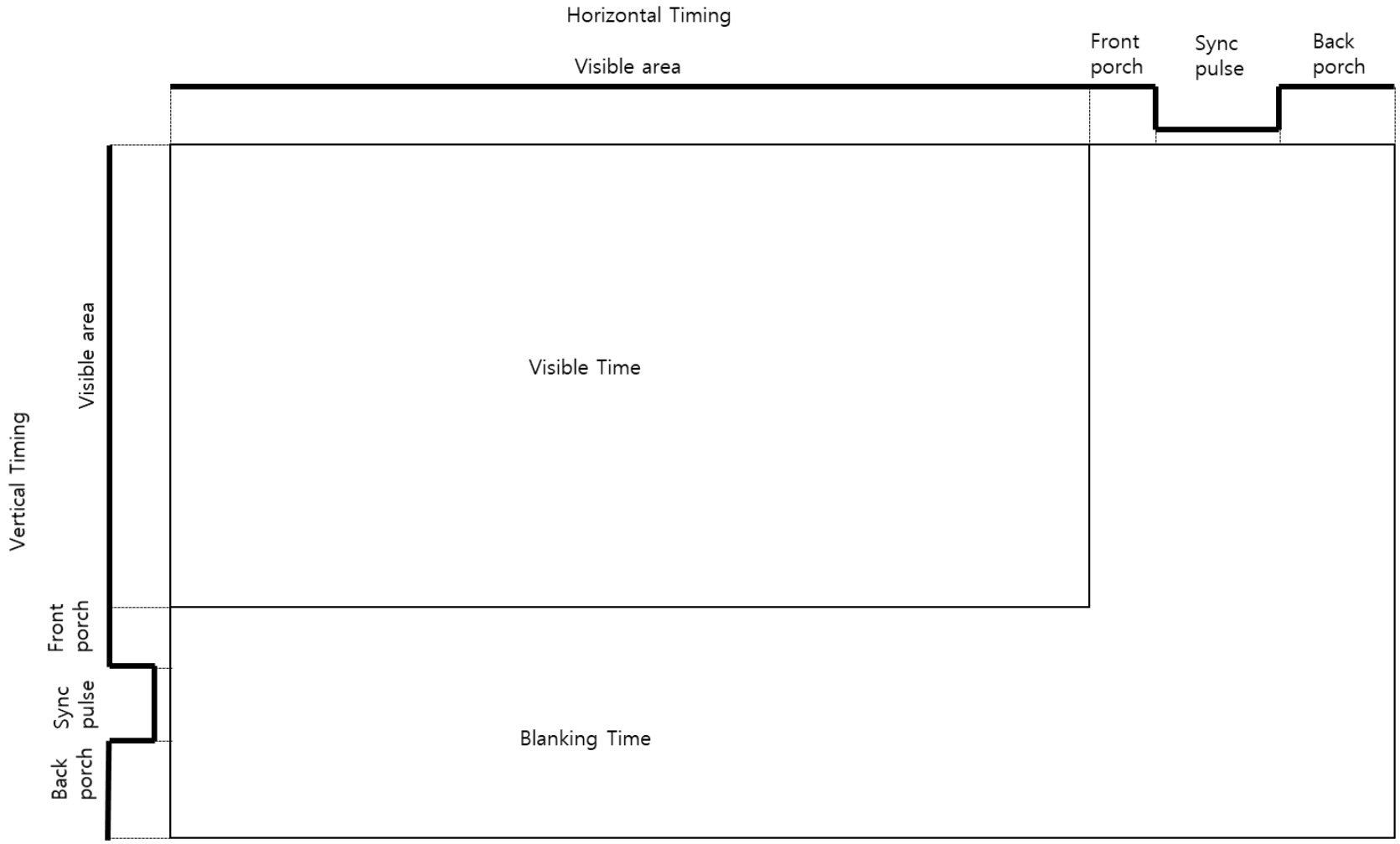
VGA Timing (Cont.)



Vertical timing (Frame)		
Frame part	Lines	Time [ms]
Visible area	600	15.84
Front porch	1	0.0264
Sync pulse	4	0.1056
Back porch	23	0.6072
Whole frame	628	16.5792

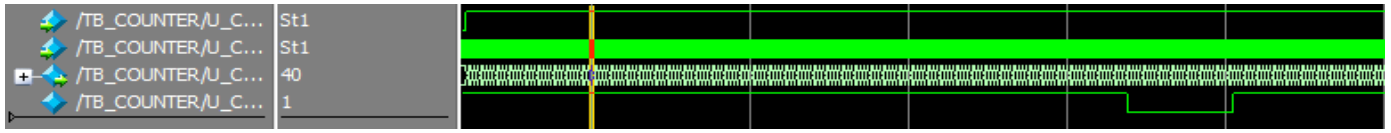
Vertical Timing

VGA Timing (Cont.)

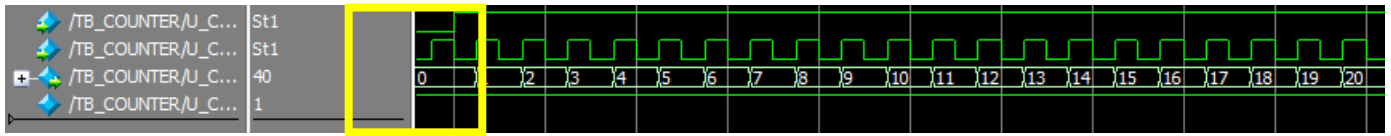


VGA Timing Diagram

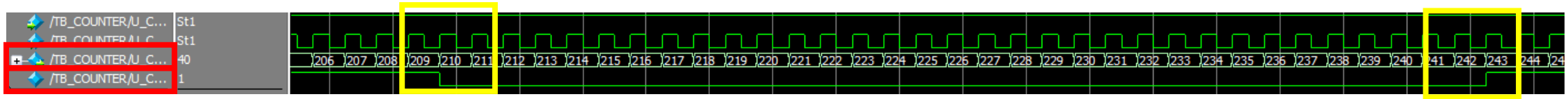
VGA Timing Simulation (Horizontal Timing)



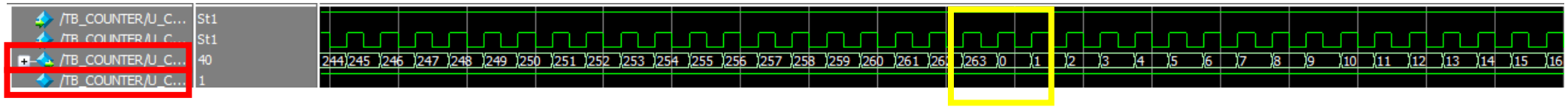
Horizontal timing overall simulation result



Horizontal timing simulation starting point ***(Count from 0)***

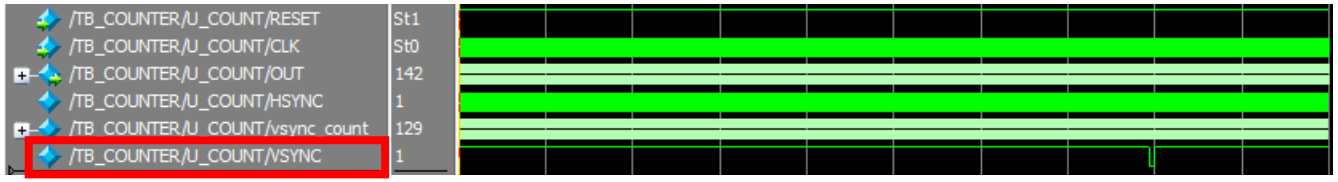


Horizontal timing simulation changing point ***(Caution on 209 & 242)***

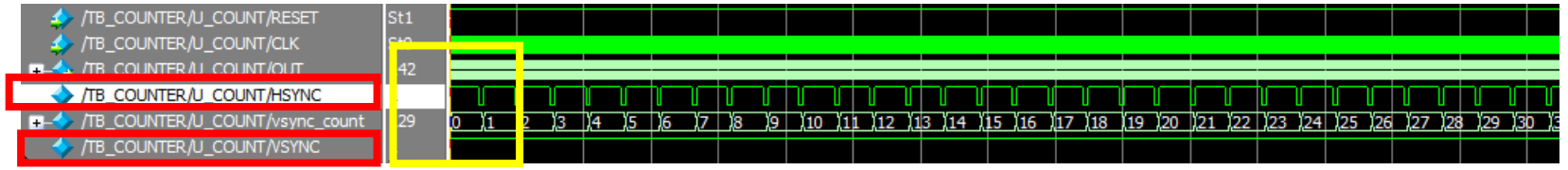


Horizontal timing simulation ending point ***(Count to 263)***

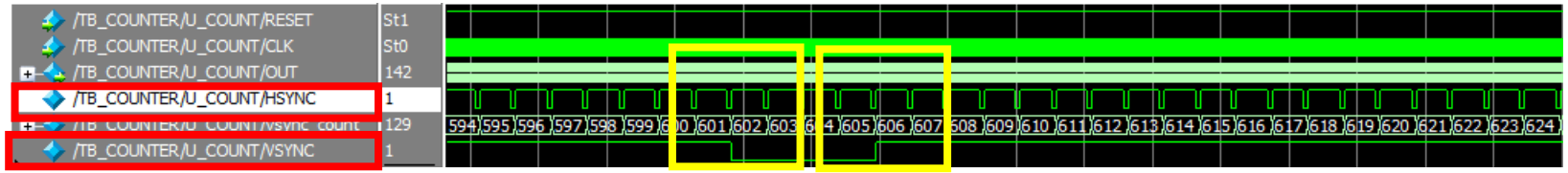
VGA Timing Simulation (Vertical Timing)



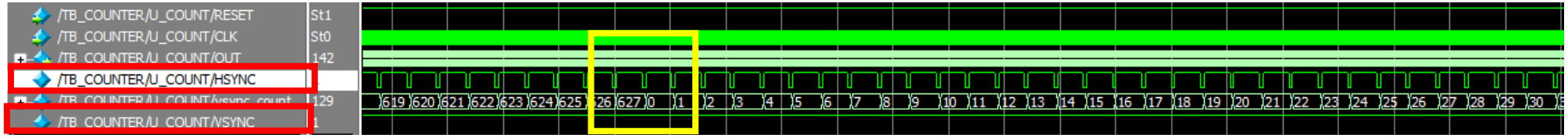
Vertical timing overall simulation result



Vertical timing simulation result starting point ***(Count from 0)***

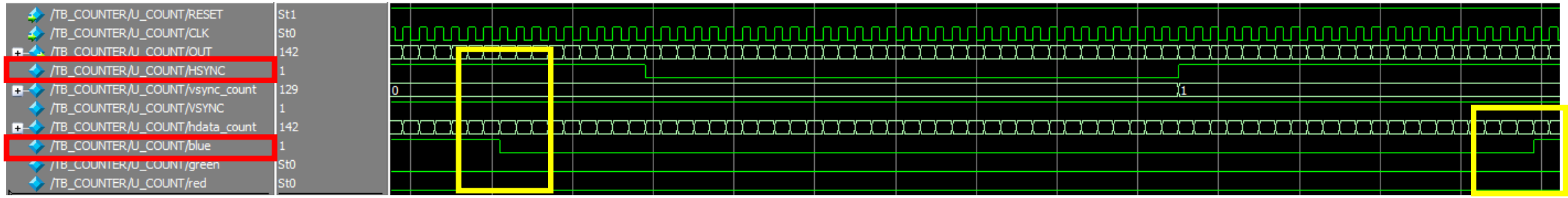


Vertical timing simulation result changing point ***(Caution on 601 & 605)***

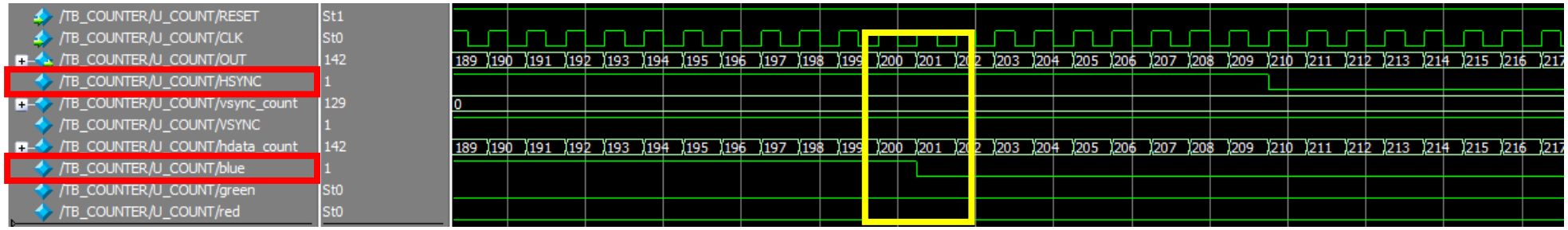


Vertical timing simulation ending point ***(Count to 627)***

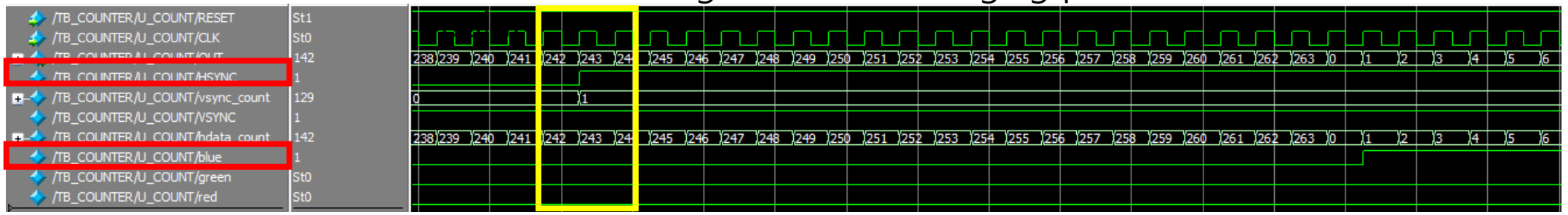
VGA Timing Simulation (Data : Color Blue)



Data timing simulation result

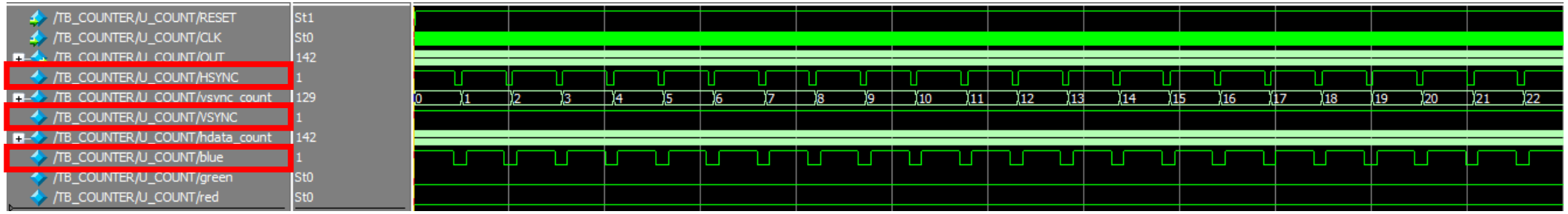


Horizontal timing simulation changing point

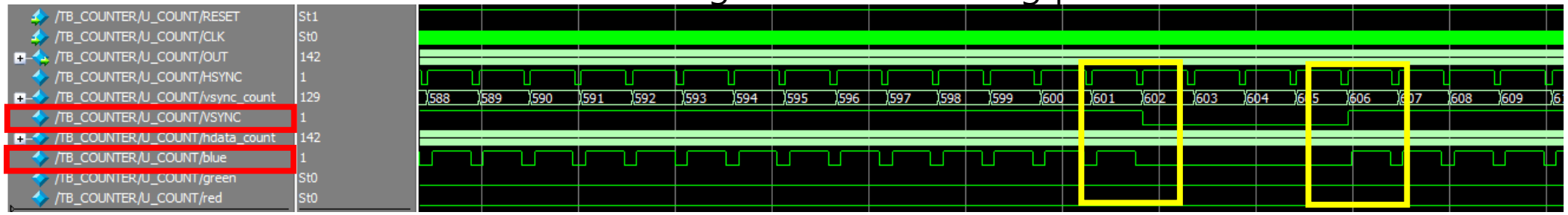


Horizontal timing simulation changing point

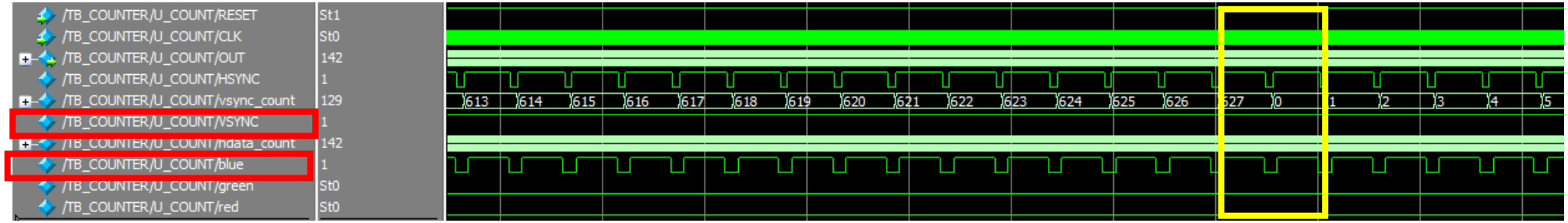
VGA Timing Simulation (Data : Color Blue)



Data timing simulation starting point

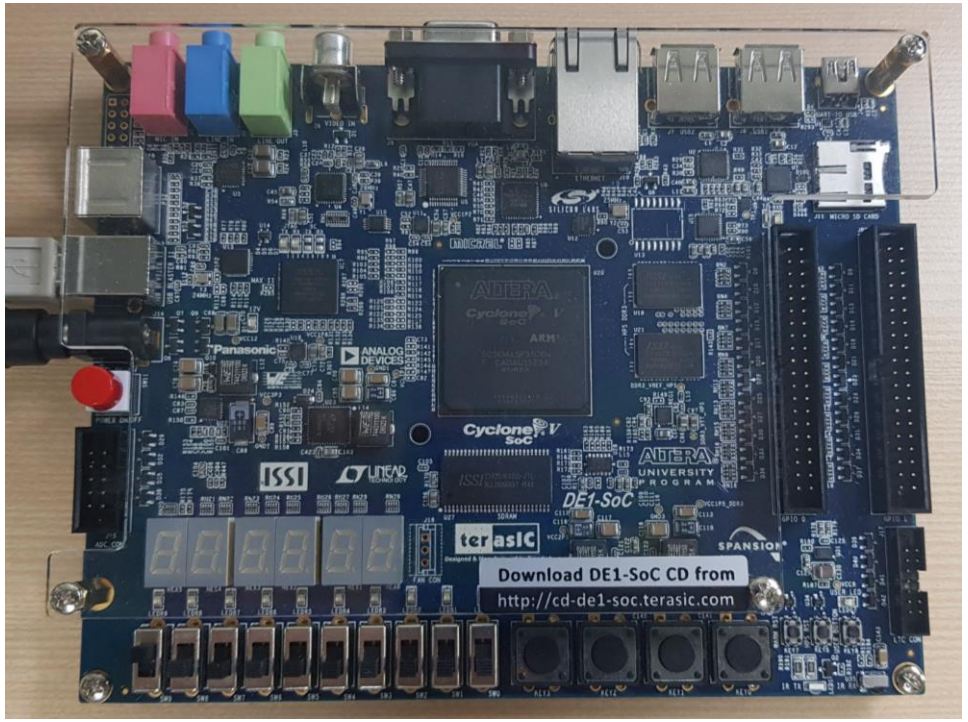


Vertical timing simulation changing point



Vertical timing simulation ending point

FPGA Board : DE1-Soc Spec



DE1-SoC Board Front

FPGA Device
- Cyclone V SoC 5CSEMA5F31C6 Device
- Dual-core ARM Cortex-A9 (HPS)
- 85K Programmable Logic Elements
- 4,450 Kbits embedded memory
- 6 Fractional PLLs
- 2 Hard Memory Controllers

Configuration and Debug
- Serial Configuration device – EPCS128 on FPGA
- On-Board USB Blaster II (Normal type B USB connector)

Sensors
- G-Sensor on HPS

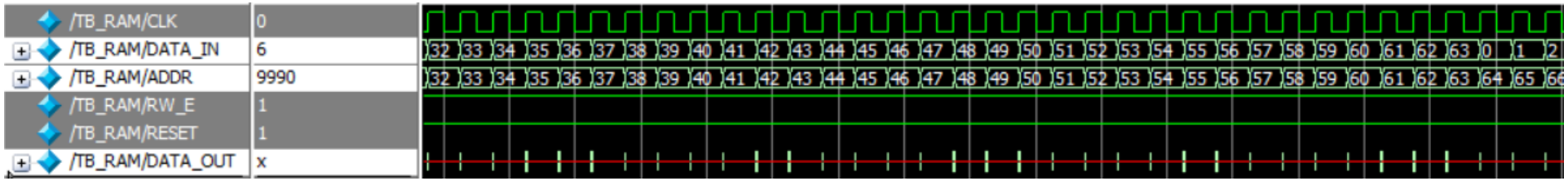
Memory Device
- 64MB (32Mx16) SDRAM on FPGA
- 1GB (2x256Mx16) DDR3 SDRAM on HPS
- Micro SD Card Socket on HPS

Communication
- Two Port USB 2.0 Host (ULPI interface with USB type A connector)
- USB to UART (micro USB type B connector)
- 10/100/1000 Ethernet
- PS/2 mouse/keyboard
- IR Emitter/Receiver

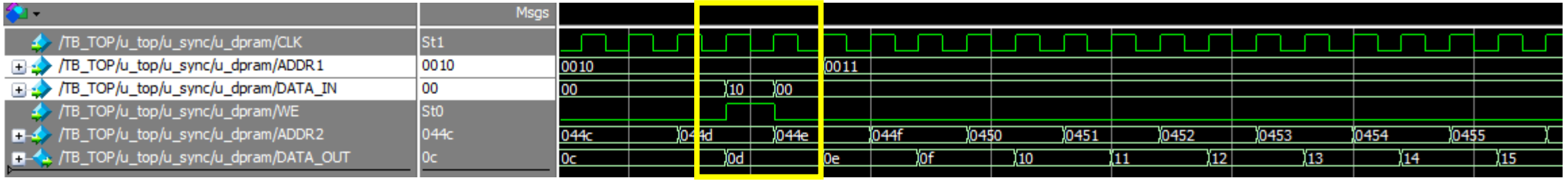
Power
- 12V DC input

DE1-SoC specifications

RAM Simulation (SRAM & DPRAM)



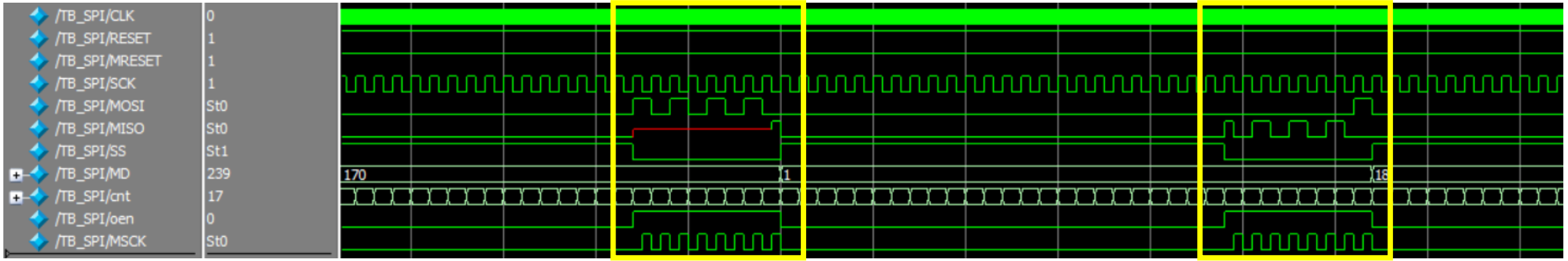
SRAM Timing simulation result



DPRAM Timing simulation result

**There is no output at the moment*

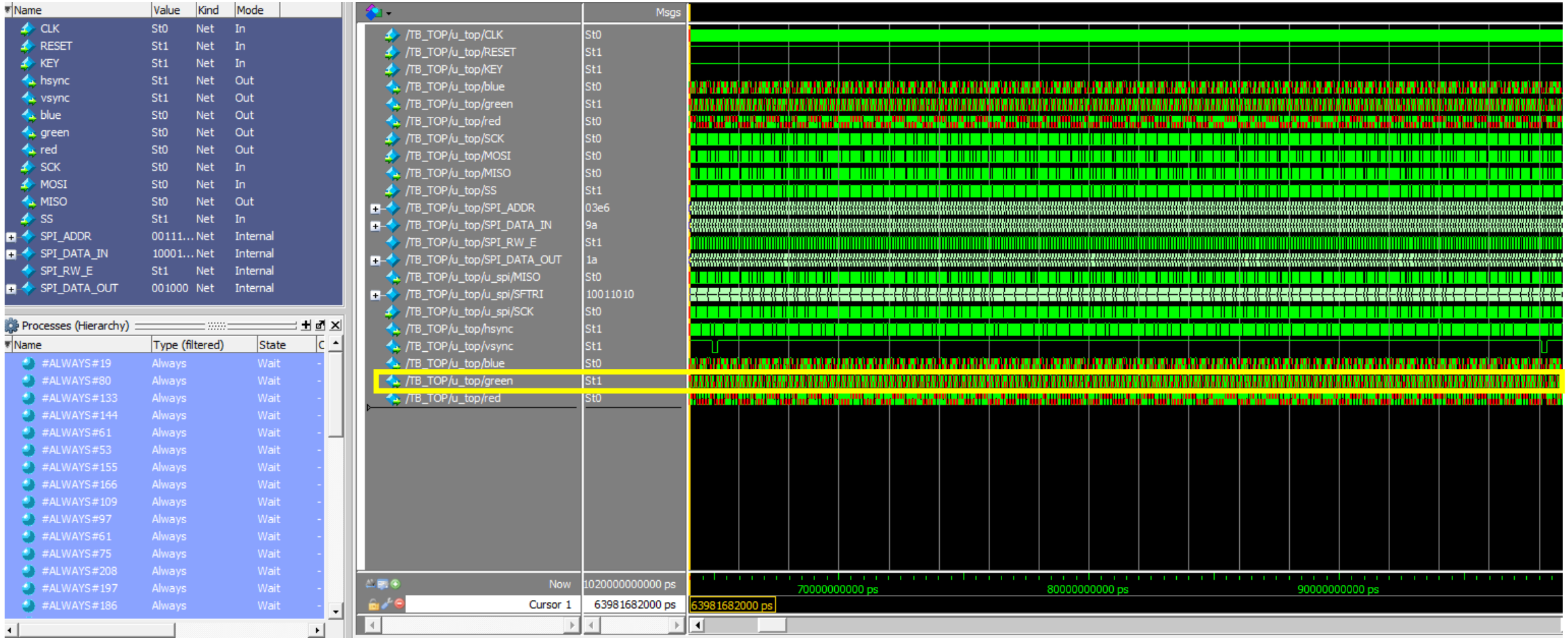
SPI Simulation (SPI Communication Protocol)



SPI Communication Protocol Timing

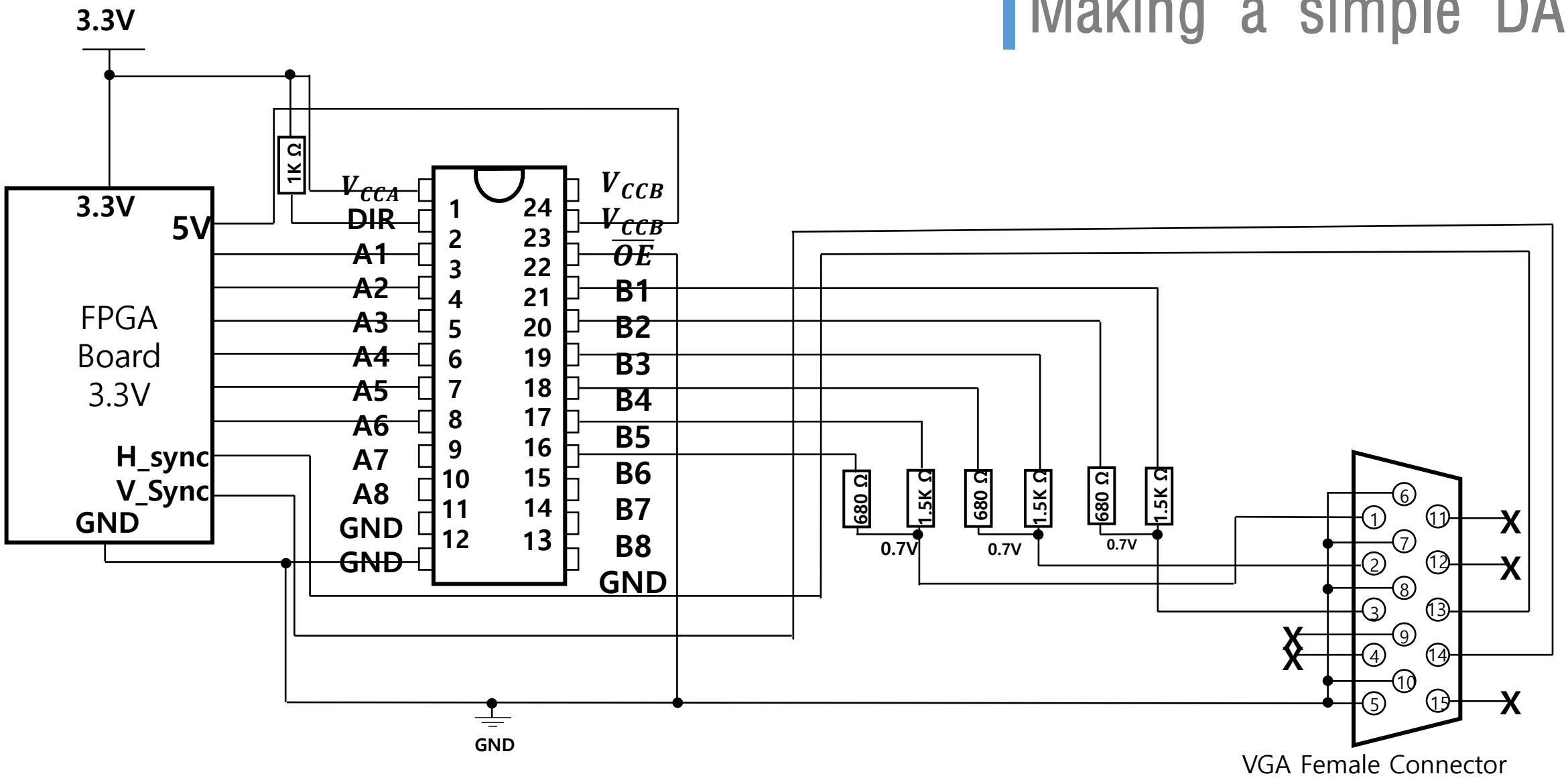
**Currently no data at first in the slave module*

Using Model Simulator to configurate timing (Integrating every element)

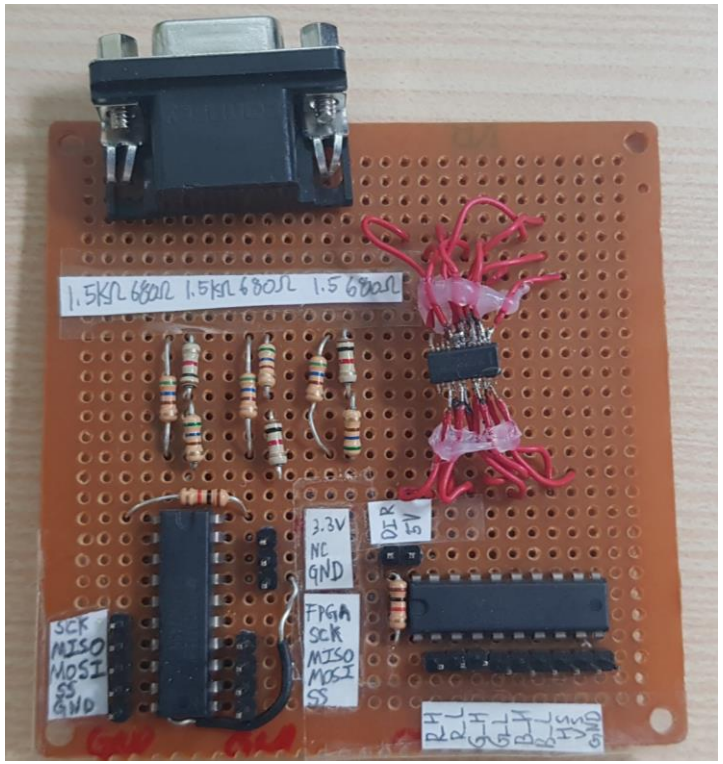


Integrated Total Timing

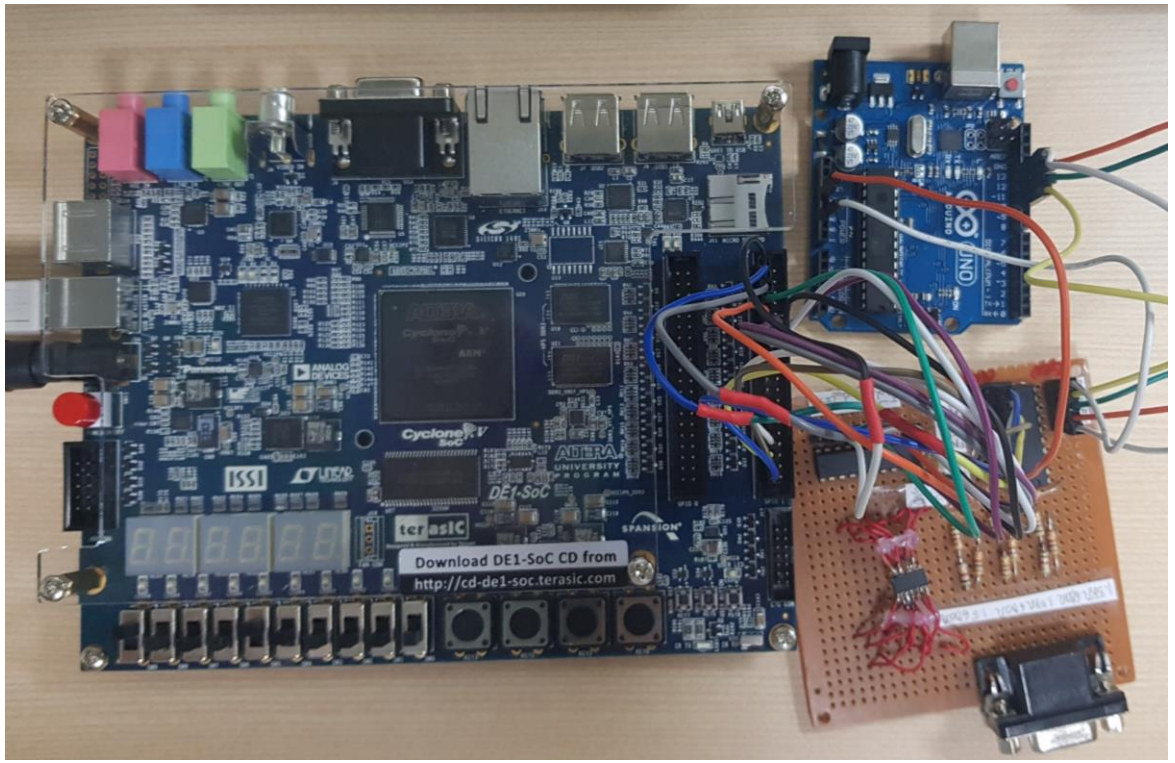
Making a simple DAC



Making a simple DAC (Cont.)

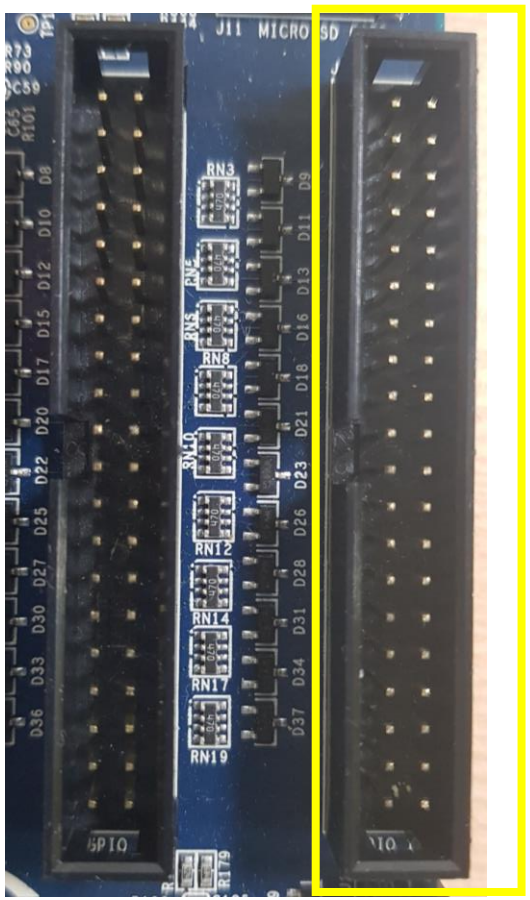


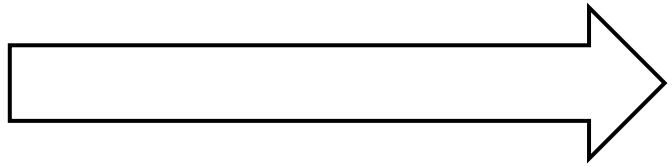
Simple DAC



All Wires Connected

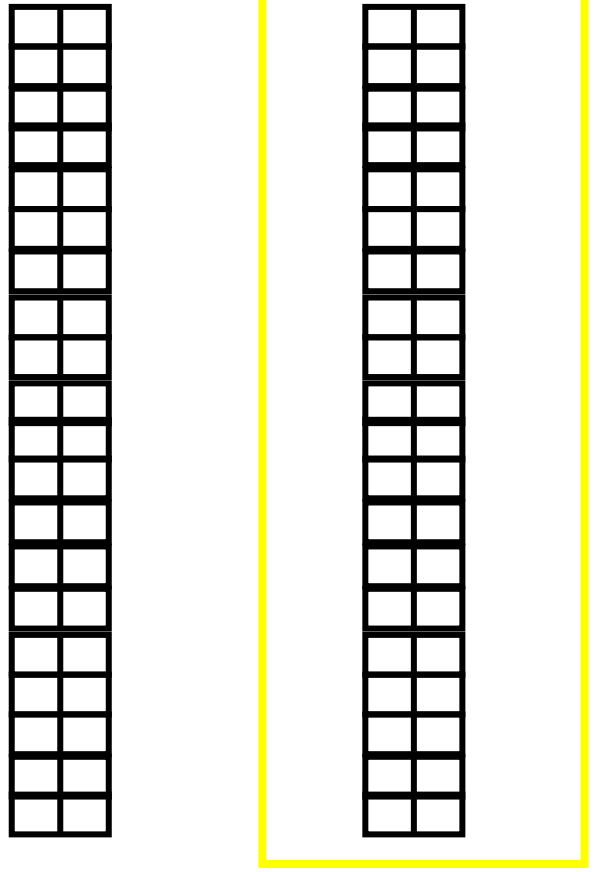
FPGA Pin Configuration





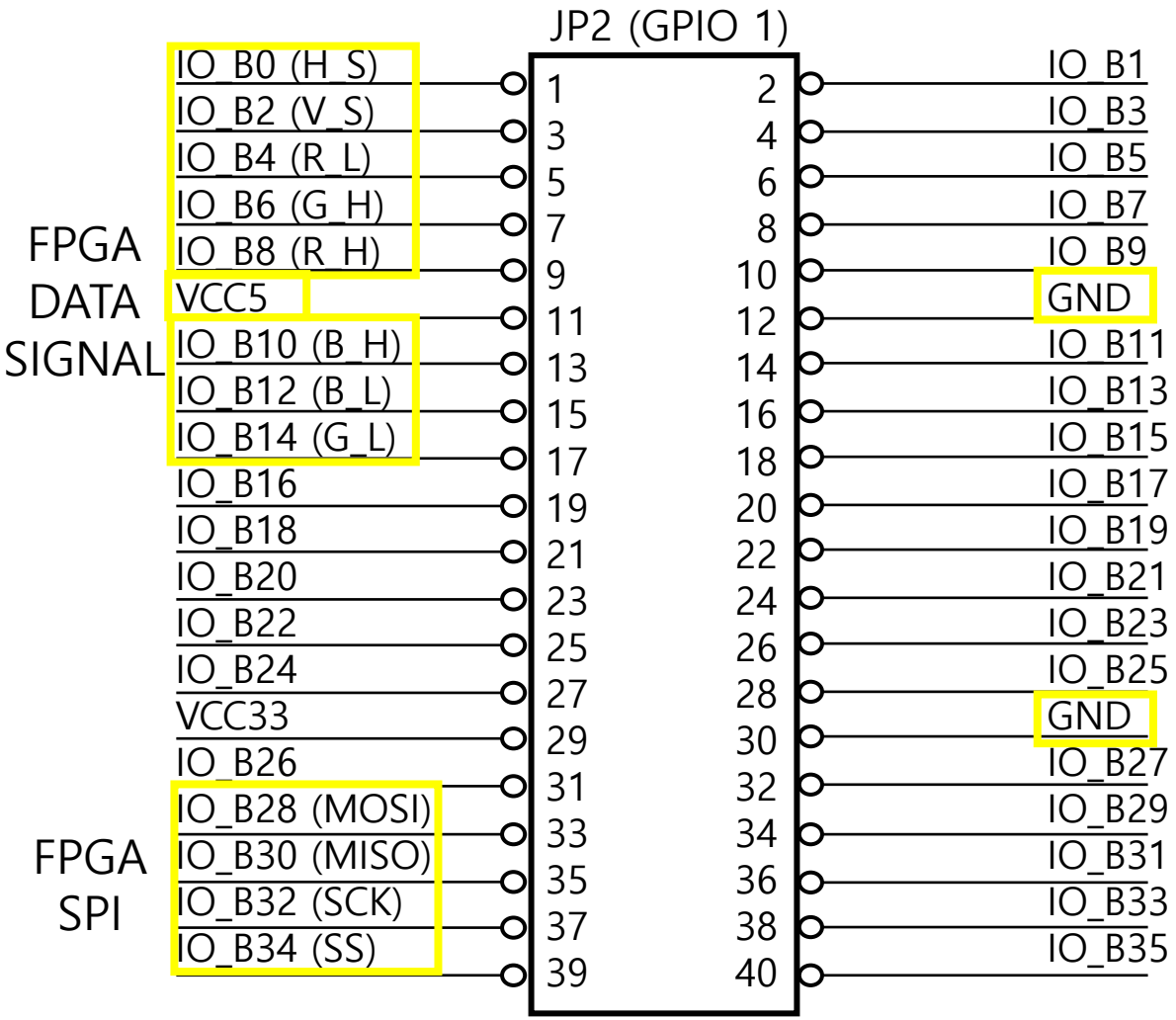
JP1 (GPIO_0)

JP2 (GPIO_1)



FPGA Pin Configuration

FPGA Pin Configuration (Cont.)



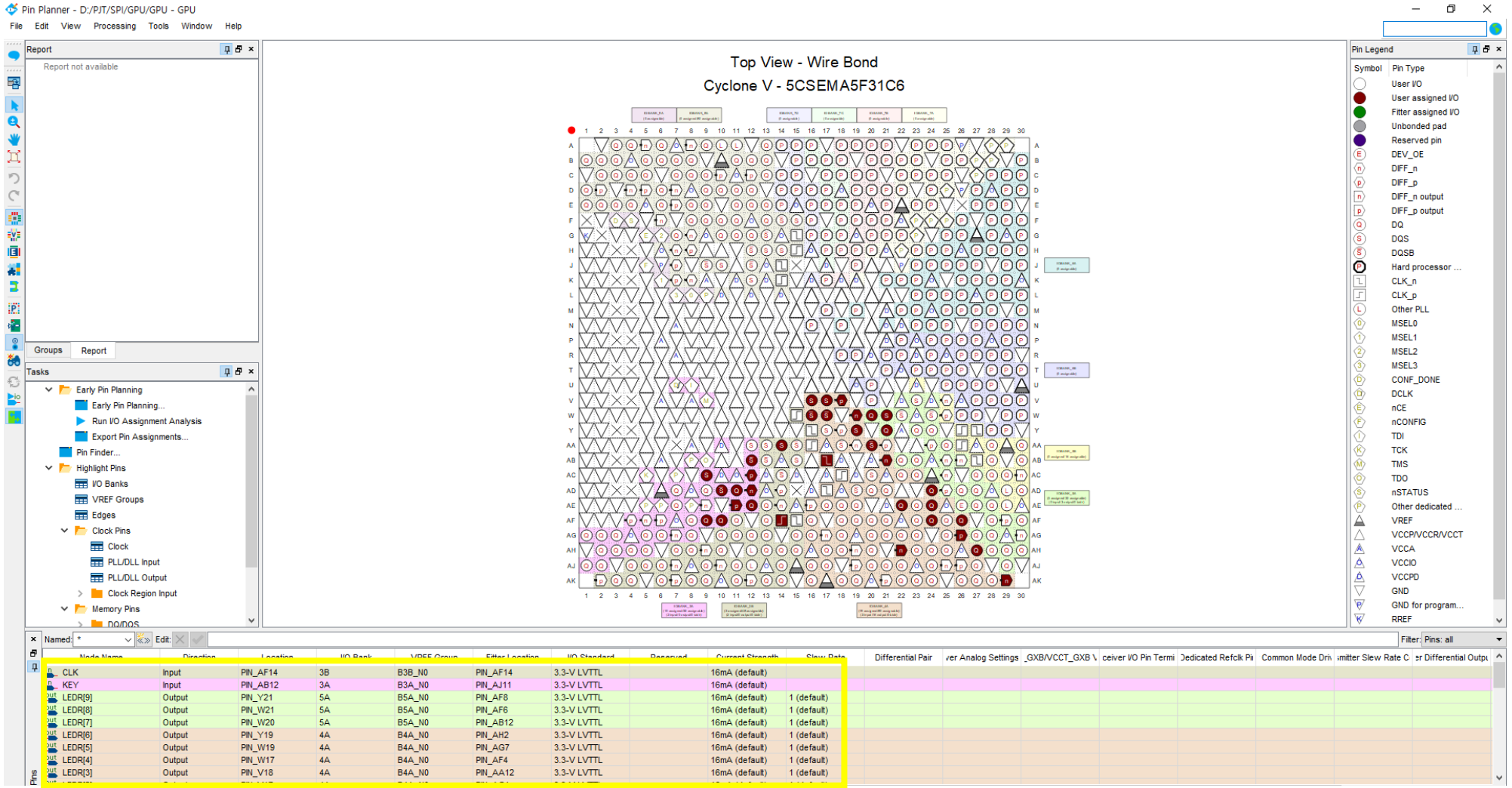
FPGA Pin Schematics

Internal Signal	I/O Pin Number	Verilog Code Number
R_L	IO_B14	AF16
R_H	IO_B12	AG16
G_L	IO_B10	AH18
G_H	IO_B8	AJ17
B_L	IO_B6	AK19
X	5V –11	X
B_H	IO_B4	AK16
HS	IO_B0	AC18
VS	IO_B2	AD17
X	GND	X

Internal Signal	I/O Pin Number	Verilog Code Number
SS	IO_B28	AH22
MOSI	IO_B30	AF24
MISO	IO_B32	AE22
SCK	IO_B34	AA20

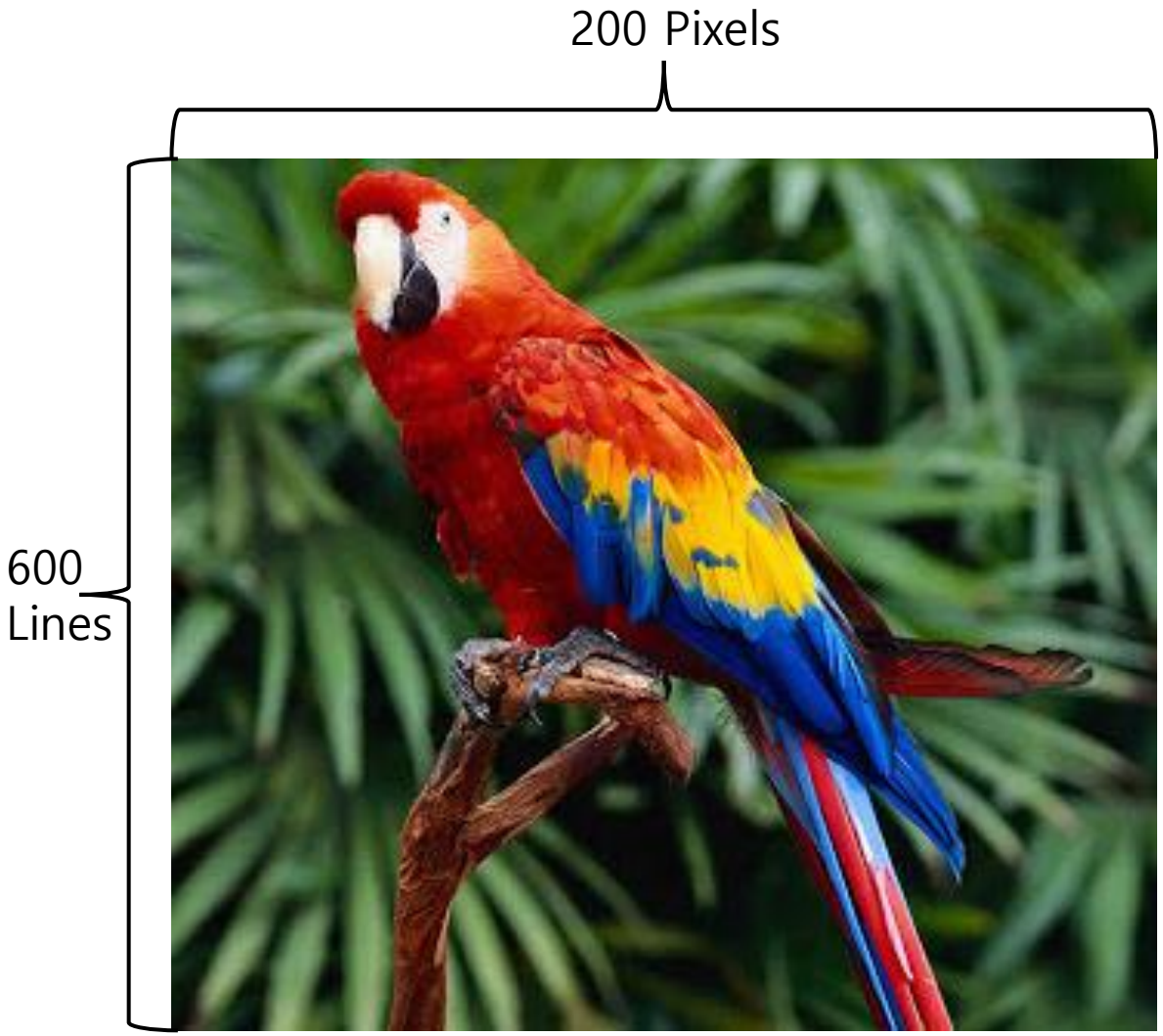
FPGA Pin Setup

FPGA Pin Configuration (Cont.)



Pin Assignment

Specification of Data

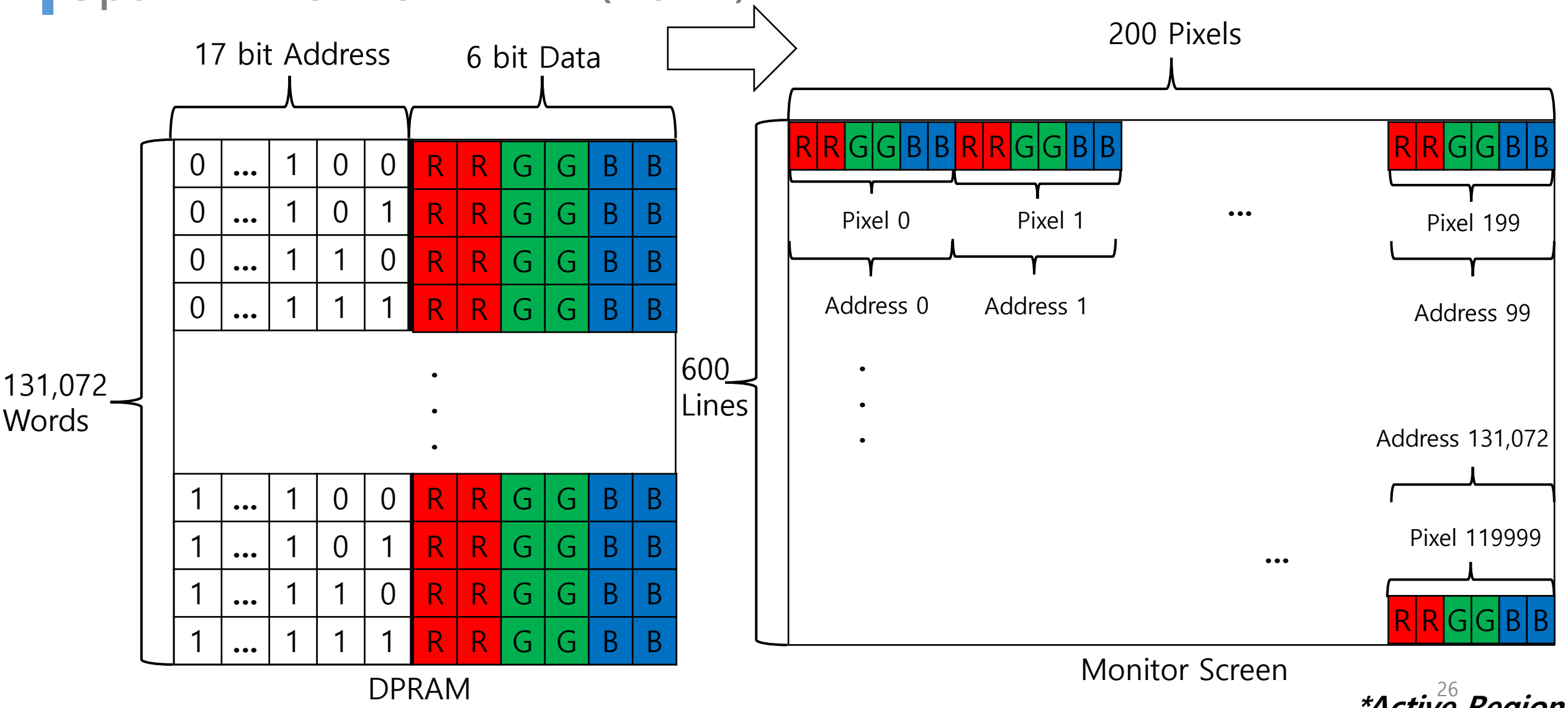


Example Picture

Example Picture		
Color	6 bit (R, G, B)	2^6 = 64 Colors
Horizontal Pixel X Vertical Line X Data	200 pixel (Active Region) X 600 lines (Active Region) X 6 bit RGB data	200 x 600 x 6 bit = 720 Kbit = 90 Kbyte
Data size	6 bit	R1 R0 G1 G0 B1 B0
Address size	17 bit Address (131,072 Words) =>2^n x m bit data	2^{17} x 6 = 786,432 bit

Example Picture Size

Specification of Data (Cont.)



*Active Region

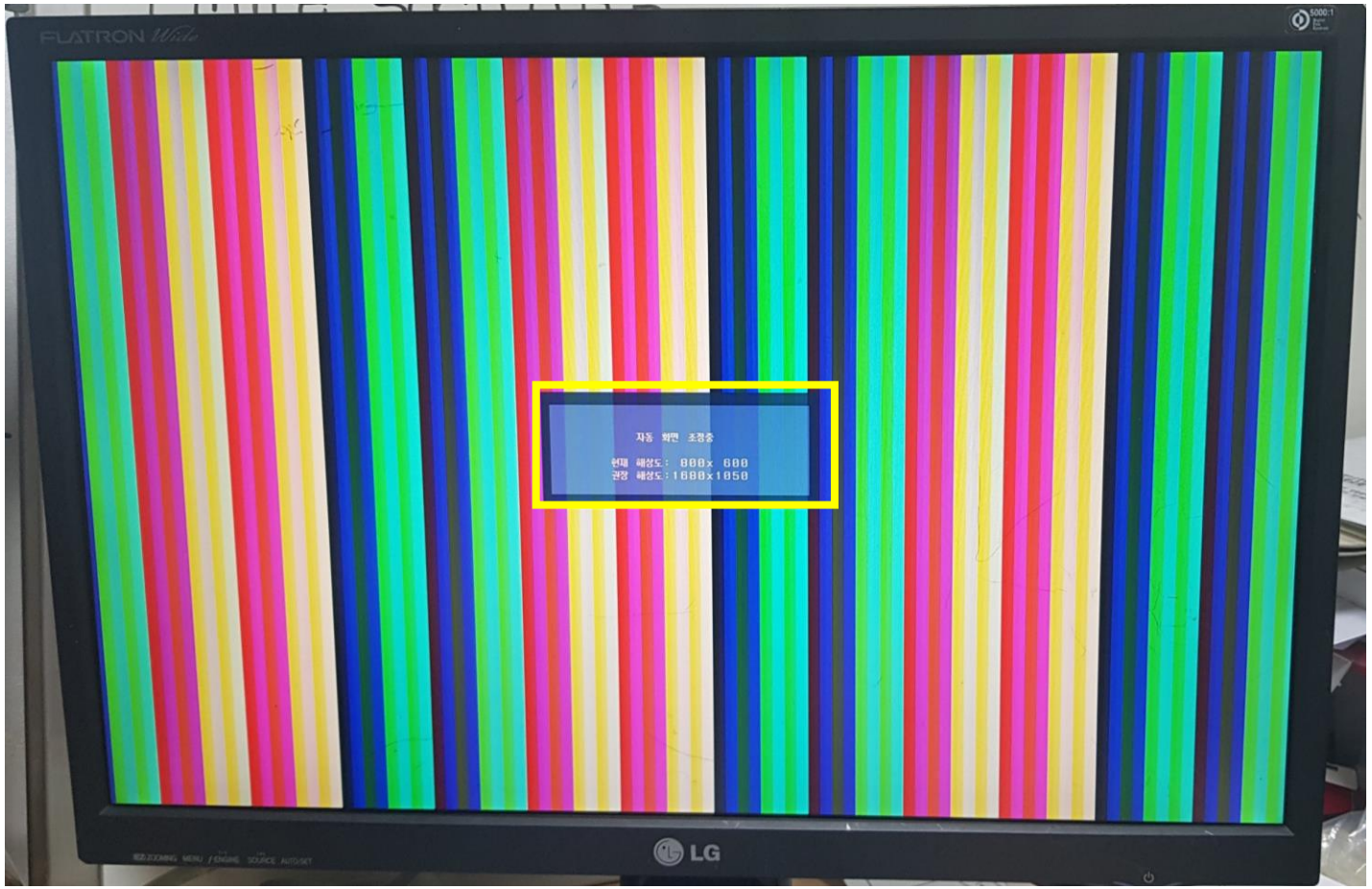
Specification of Data (Cont.)

Color Code

Binary	Hexadecimal	Color	Brightness
xx 11 00 00	0x30	Red	Bright ⇕ Dark
xx 10 00 00	0x20	Red	
xx 01 00 00	0x10	Red	
xx 00 00 00	0x00	Red Off	
xx 00 11 00	0x0c	Green	Bright ⇕ Dark
xx 00 10 00	0x08	Green	
xx 00 01 00	0x04	Green	
xx 00 00 00	0x00	Green Off	
xx 00 00 11	0x03	Blue	Bright ⇕ Dark
xx 00 00 10	0x02	Blue	
xx 00 00 01	0x01	Blue	
xx 00 00 00	0x00	Blue Off	

**We do not use LMB of the binary number(6 bits of data)*

Color Test



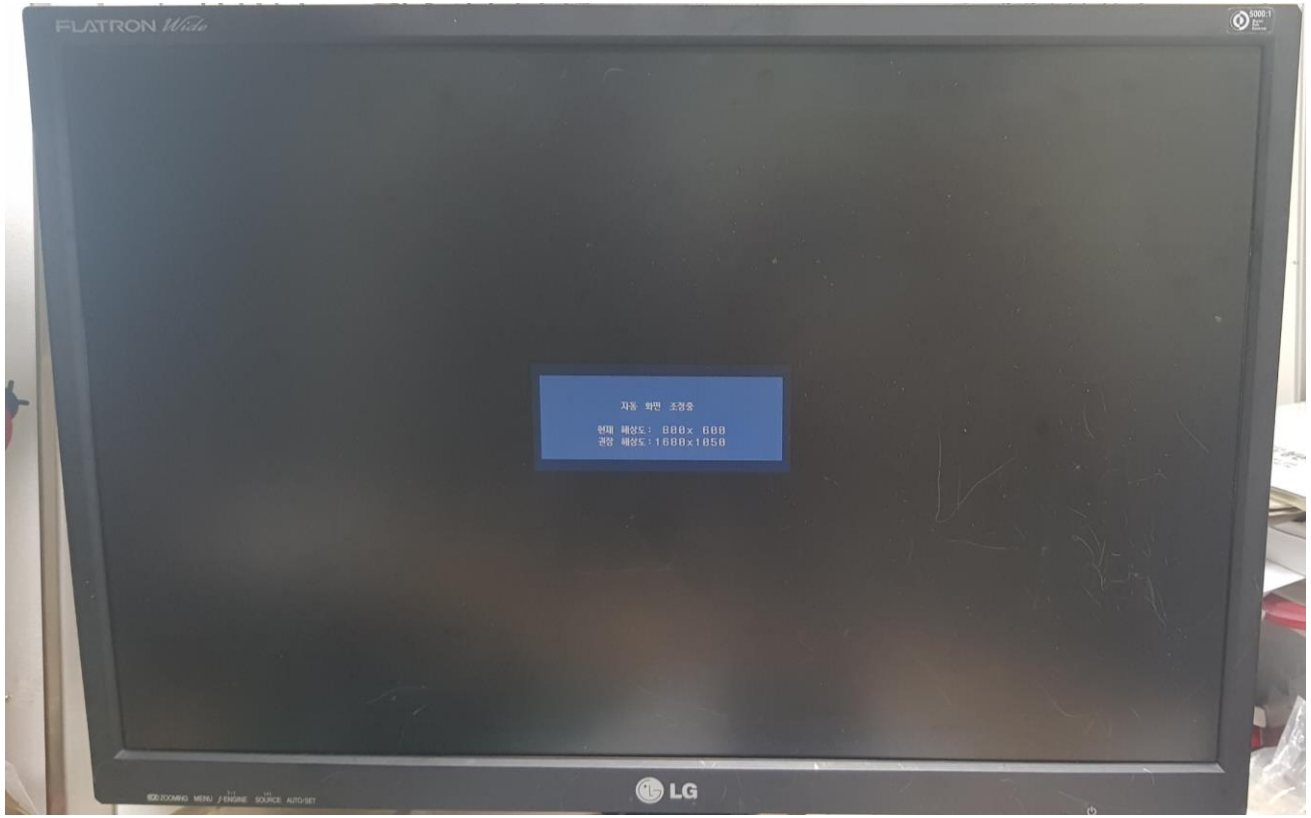
Color test

800 x 600 Screen detected

Screen Test With Arduino Uno

```
  digitalWrite(SS, LOW);
A = SPI.transfer(0x00); // ADDRESS_High *Default
A = SPI.transfer(0x00); // ADDRESS Low *Default

  for (j=0; j<200;j++){ // 1 line x 200
     for (i=0; i<200;i++) A = SPI.transfer(0x00);// 1 line
  }

  for (j=0; j<200;j++){ //1 line x 200
     for (i=0; i<200;i++) A = SPI.transfer(0x00); // 1 line
  }

  for (j=0; j<200;j++){ //1 line x 200
     for (i=0; i<200;i++) A = SPI.transfer(0x00); // 1 line
  }

digitalWrite(SS, HIGH);
```

Arduino Code



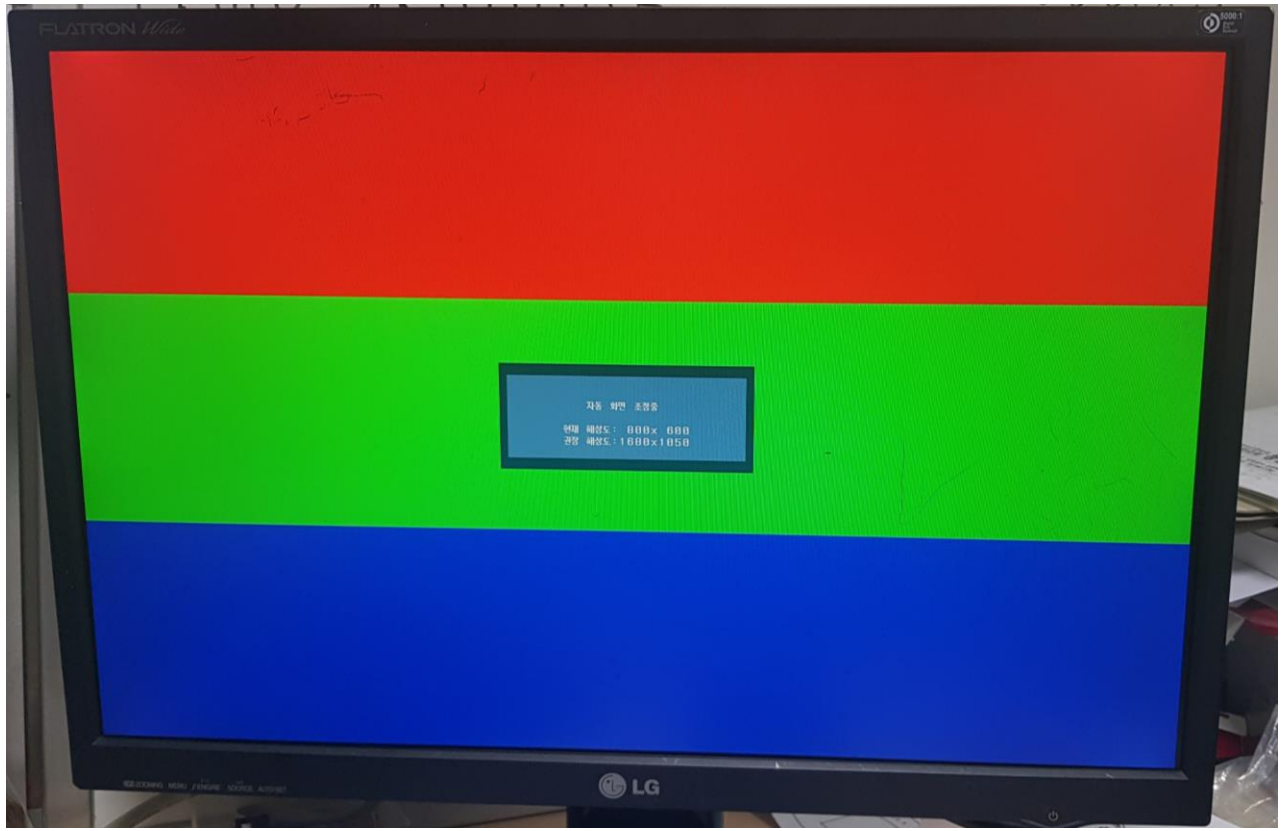
Blank Screen Test

800 x 600 Screen detected

Screen Test With Arduino Uno (Cont.)

```
  digitalWrite(SS, LOW);
  //delay(1);
A = SPI.transfer(0x00); // ADDRESS_High *Default
A = SPI.transfer(0x00); // ADDRESS Low *Default

  for (j=0; j<200;j++){ // 1 line x 200
     for (i=0; i<200;i++) A = SPI.transfer(0x30);// 1 line
  }

  for (j=0; j<200;j++){ //1 line x 200
     for (i=0; i<200;i++) A = SPI.transfer(0x0C); // 1 line
  }

  for (j=0; j<200;j++){ //1 line x 200
     for (i=0; i<200;i++) A = SPI.transfer(0x03); // 1 line
  }

digitalWrite(SS, HIGH);
```

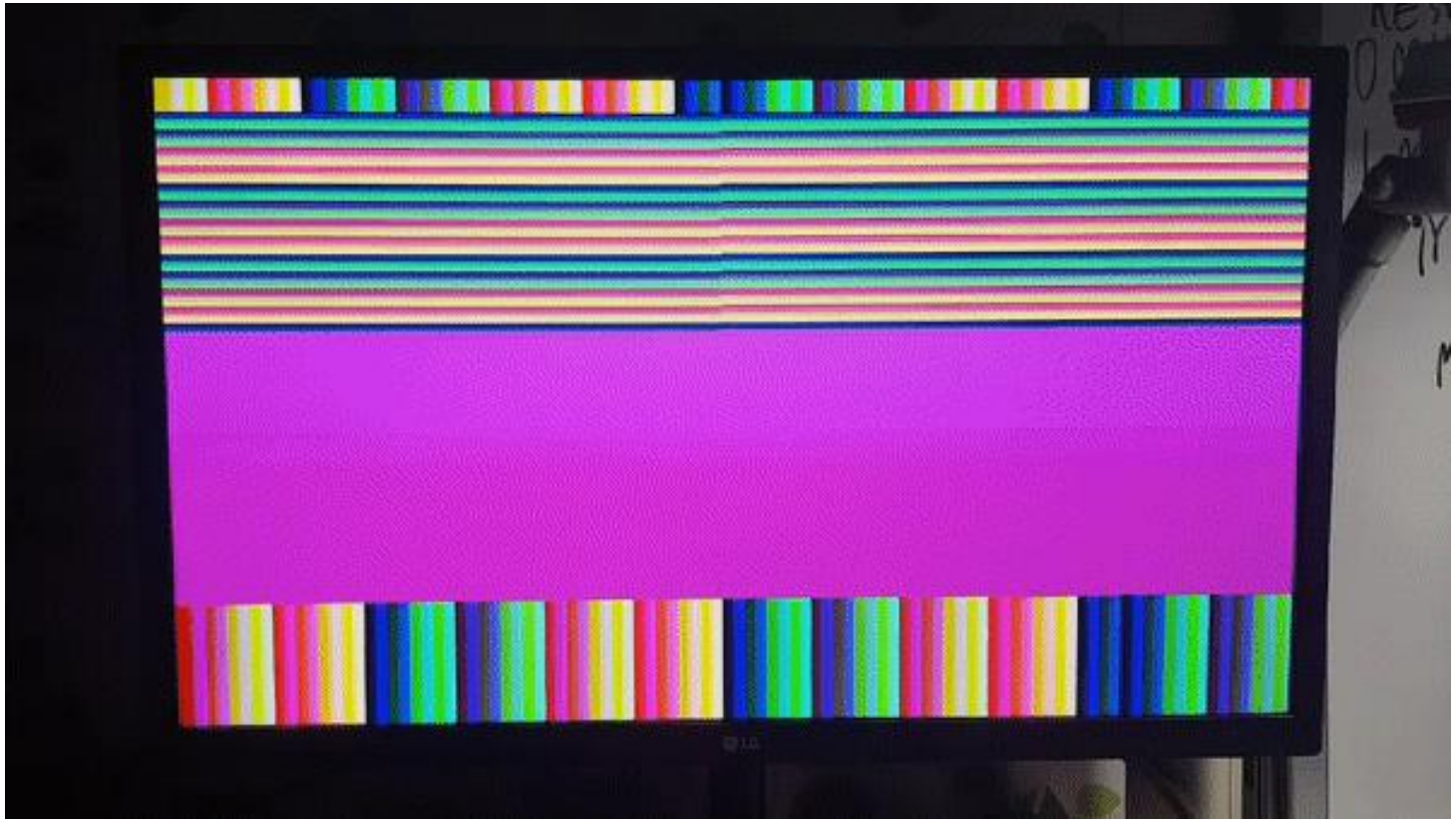
Arduino Code



RGB Screen Test

800 x 600 Screen detected

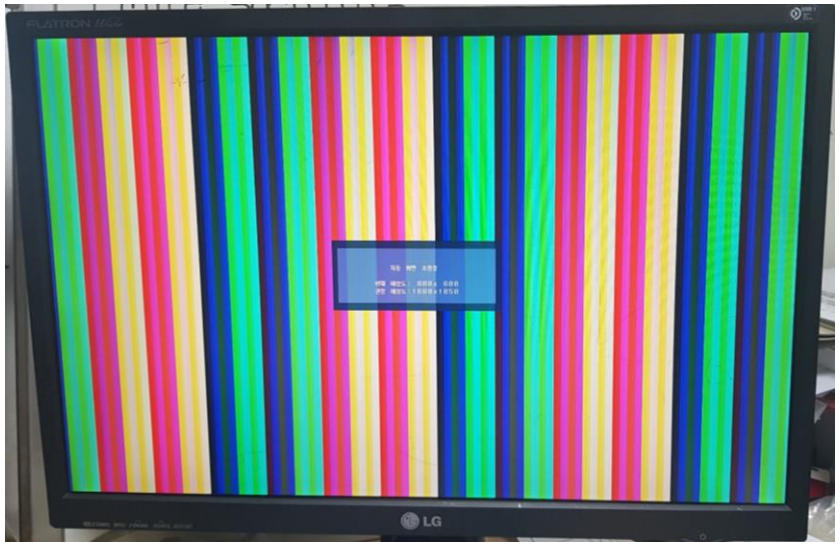
Screen Test With Arduino Uno (Cont.)

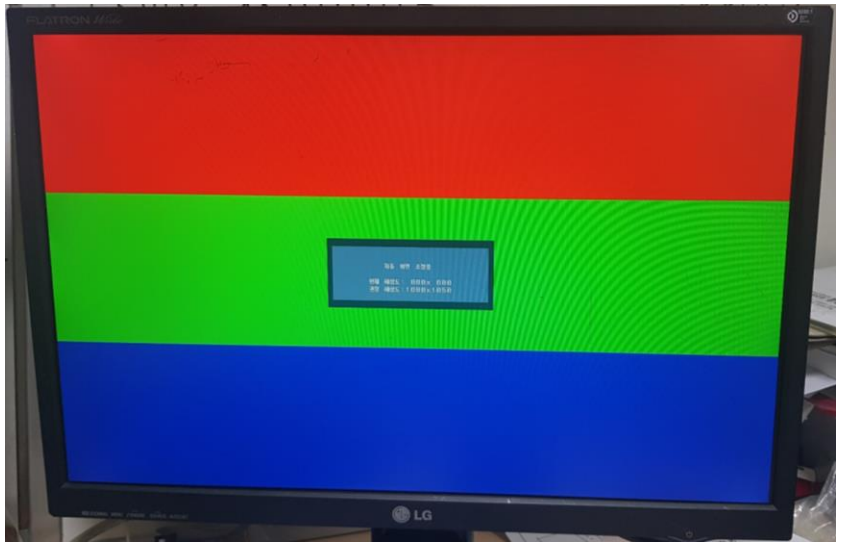


Test Video

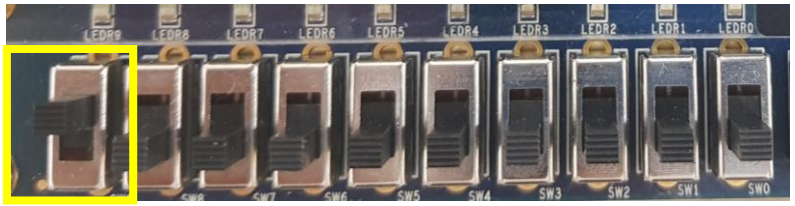
This screen is refreshed 1000 times and stops

Other Features on the FPGA Board

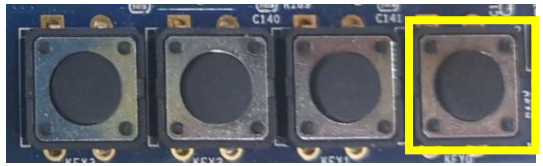




Color Test On

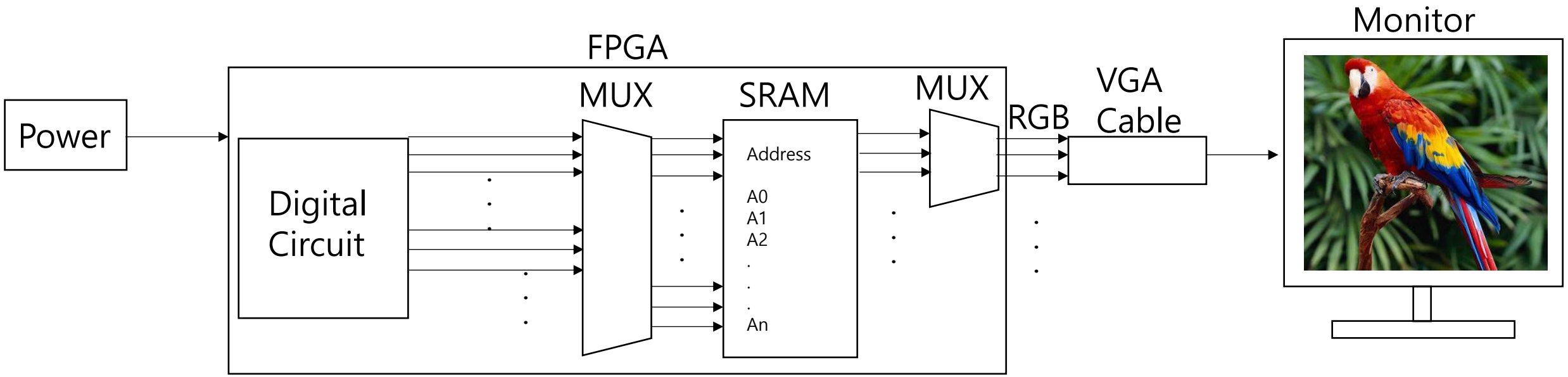


Color Test Off(Data in)



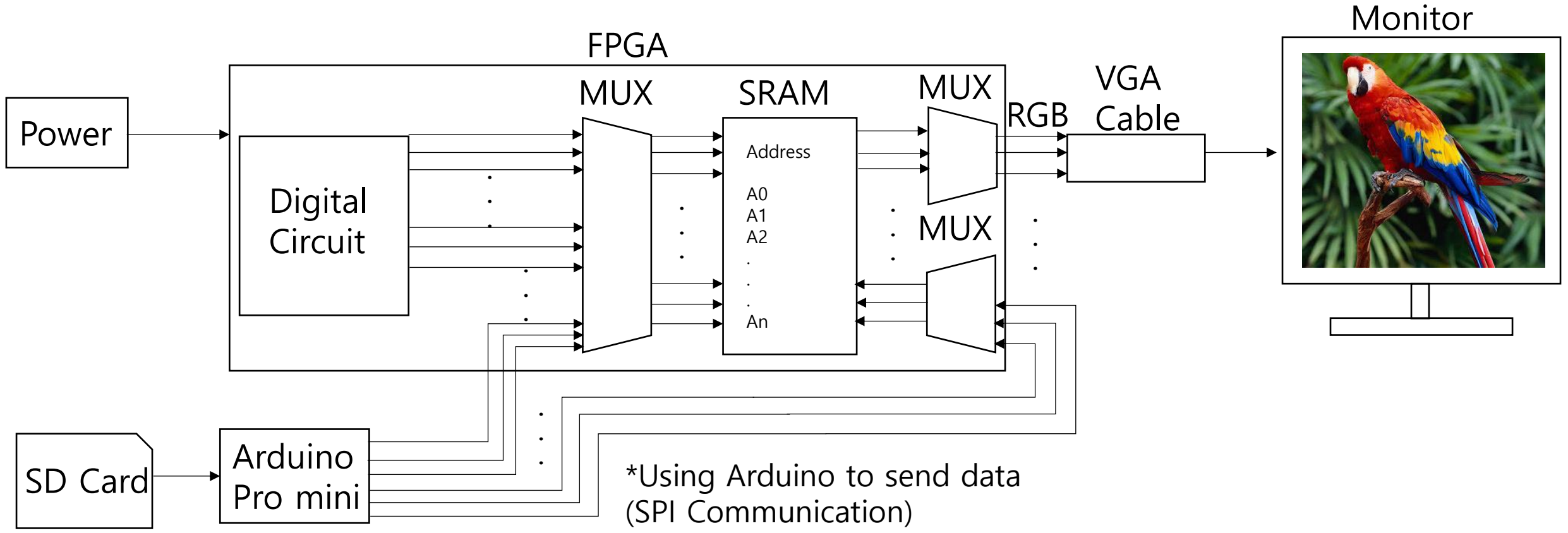
Reset Key

FPGA H/W Block Diagram Option1

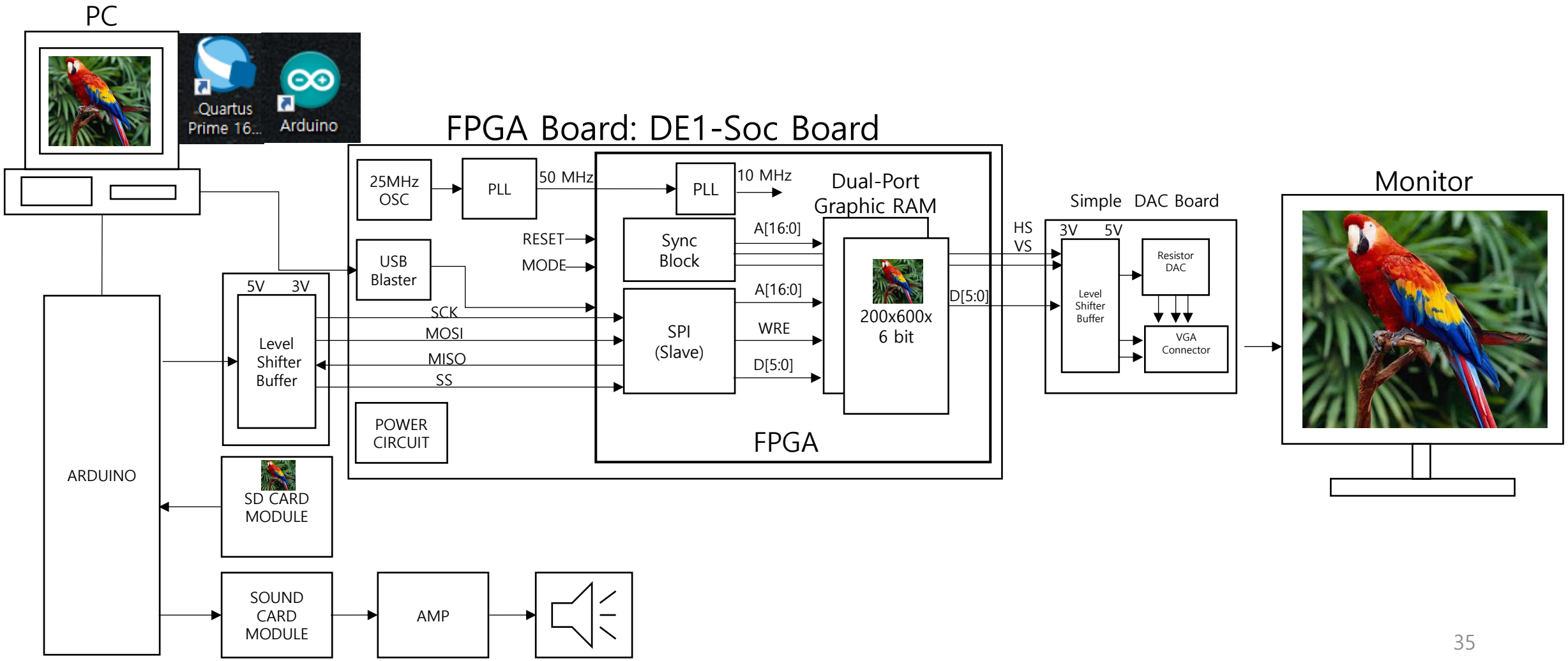


*Directly loading data to SRAM

FPGA H/W Block Diagram Option2 (Selected)



Block Diagram of Graphic Card Part



Debugging

Arduino

Not a big problem

Arduino

Arduino

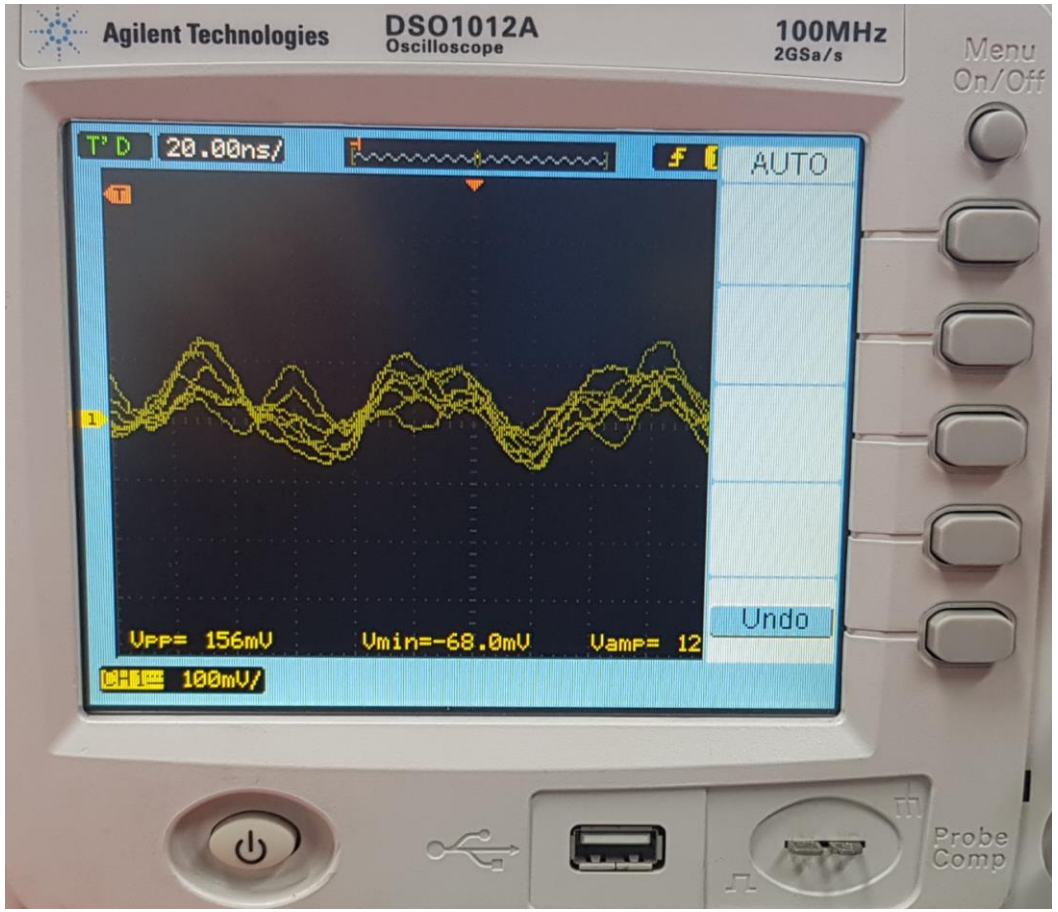
Don't Know if it's Possible

? ? ?

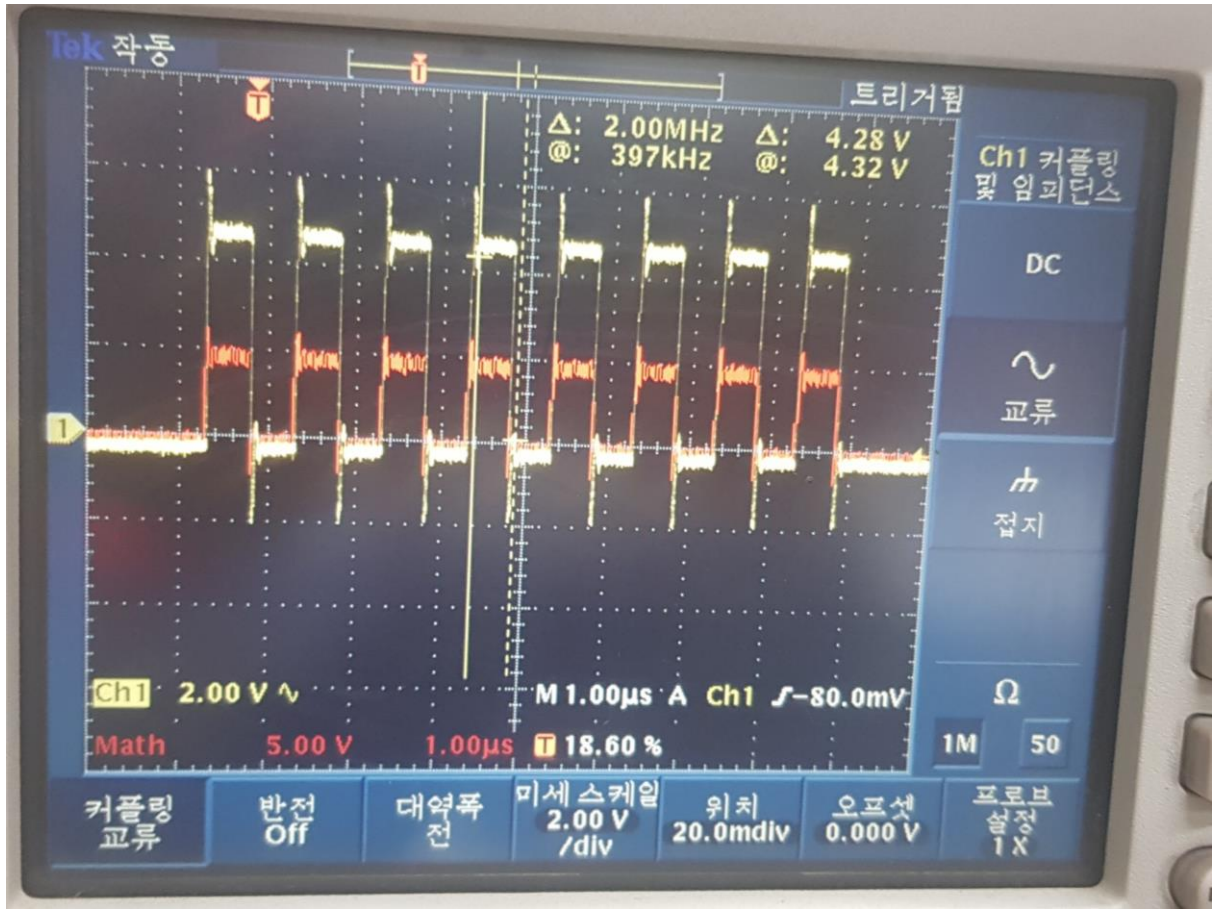
FPGA

Courtesy of Prof. Hyuk Jun Oh's Grad. Students

Debugging (Cont.)



SPI Clock Detection Problem



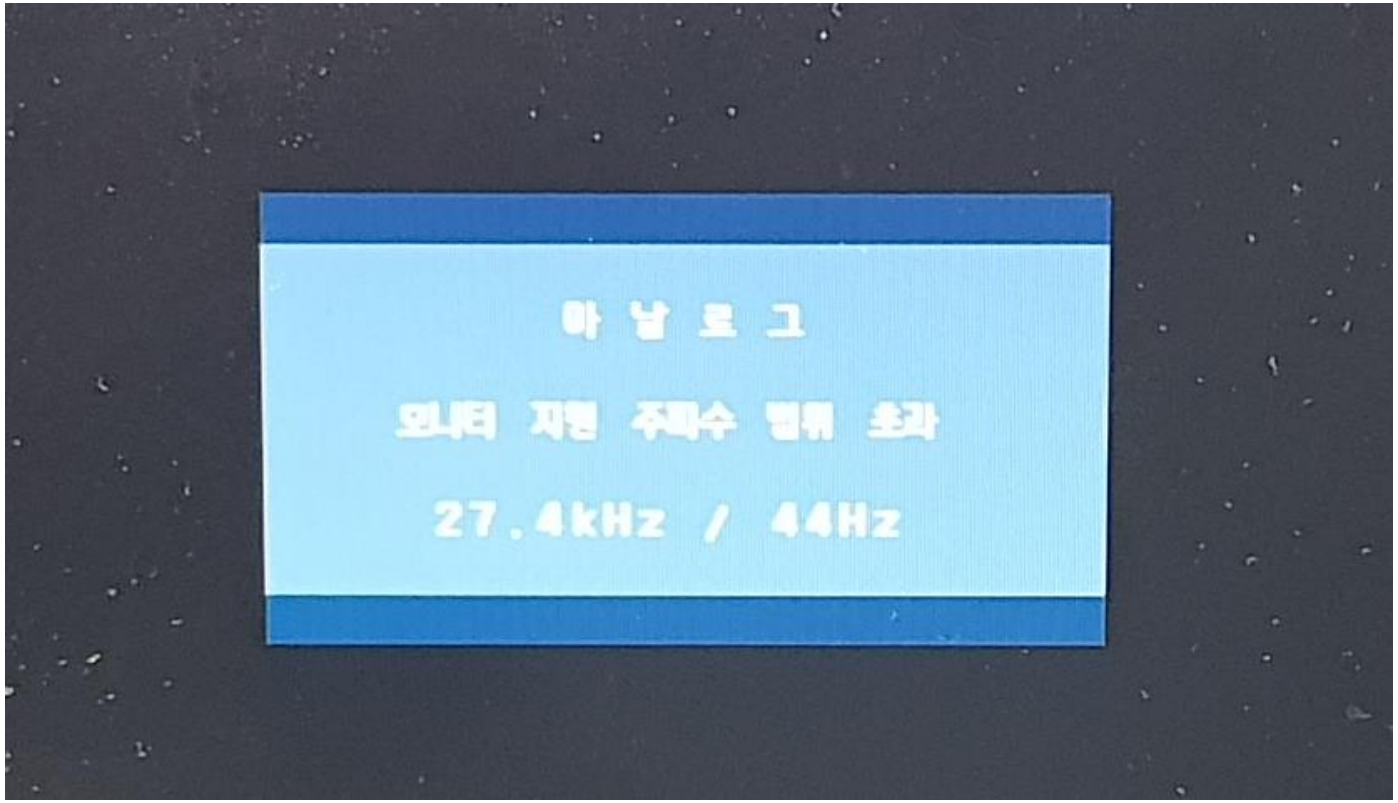
SPI Clock Detection Resolved using Trigger

Courtesy of Prof. Hyuk Jun Oh's Grad. Students

Debugging (Cont.)

Miscalculated Timing on H_sync & V_sync Pulses

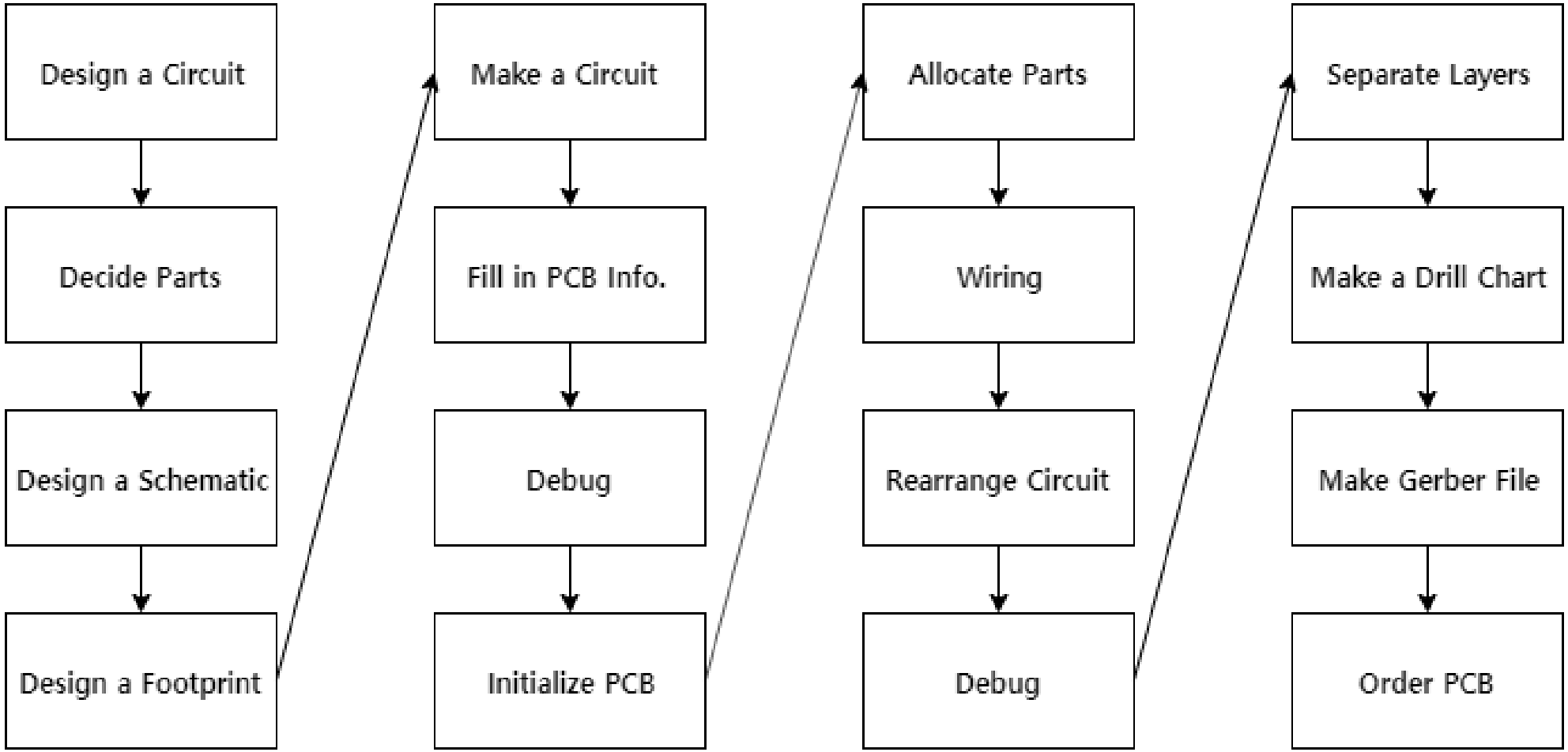
Debugging (Cont.)



Counts the wrong amount of pixels

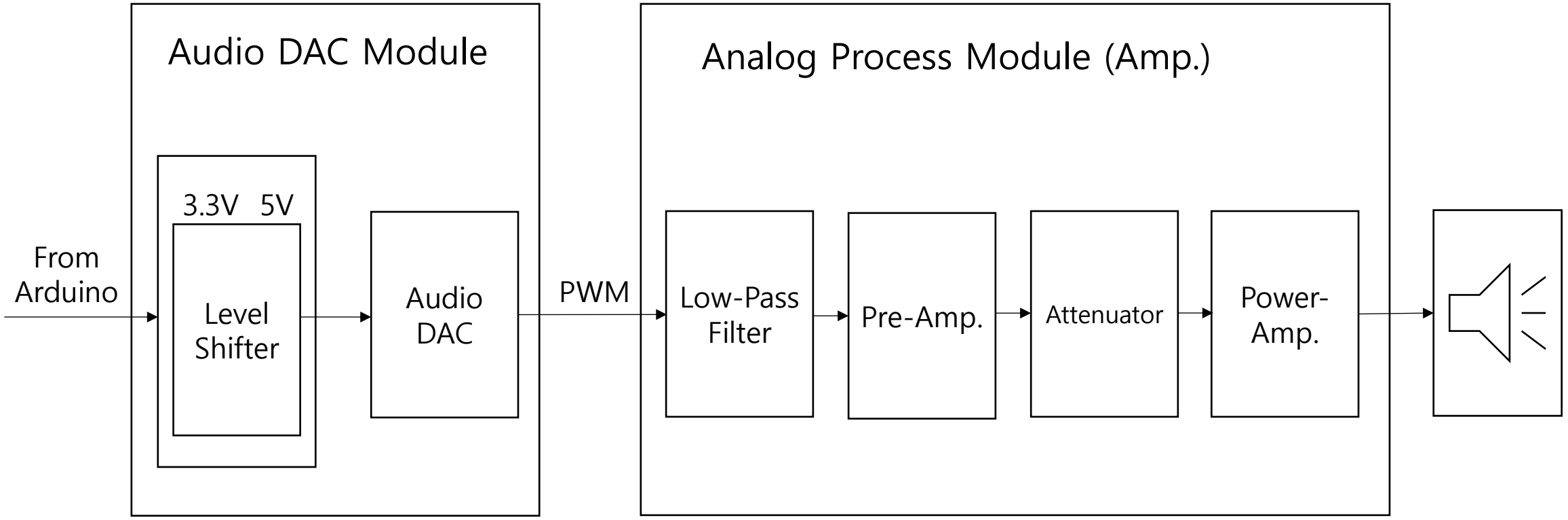
Simple Sound Card

PCB Building Process

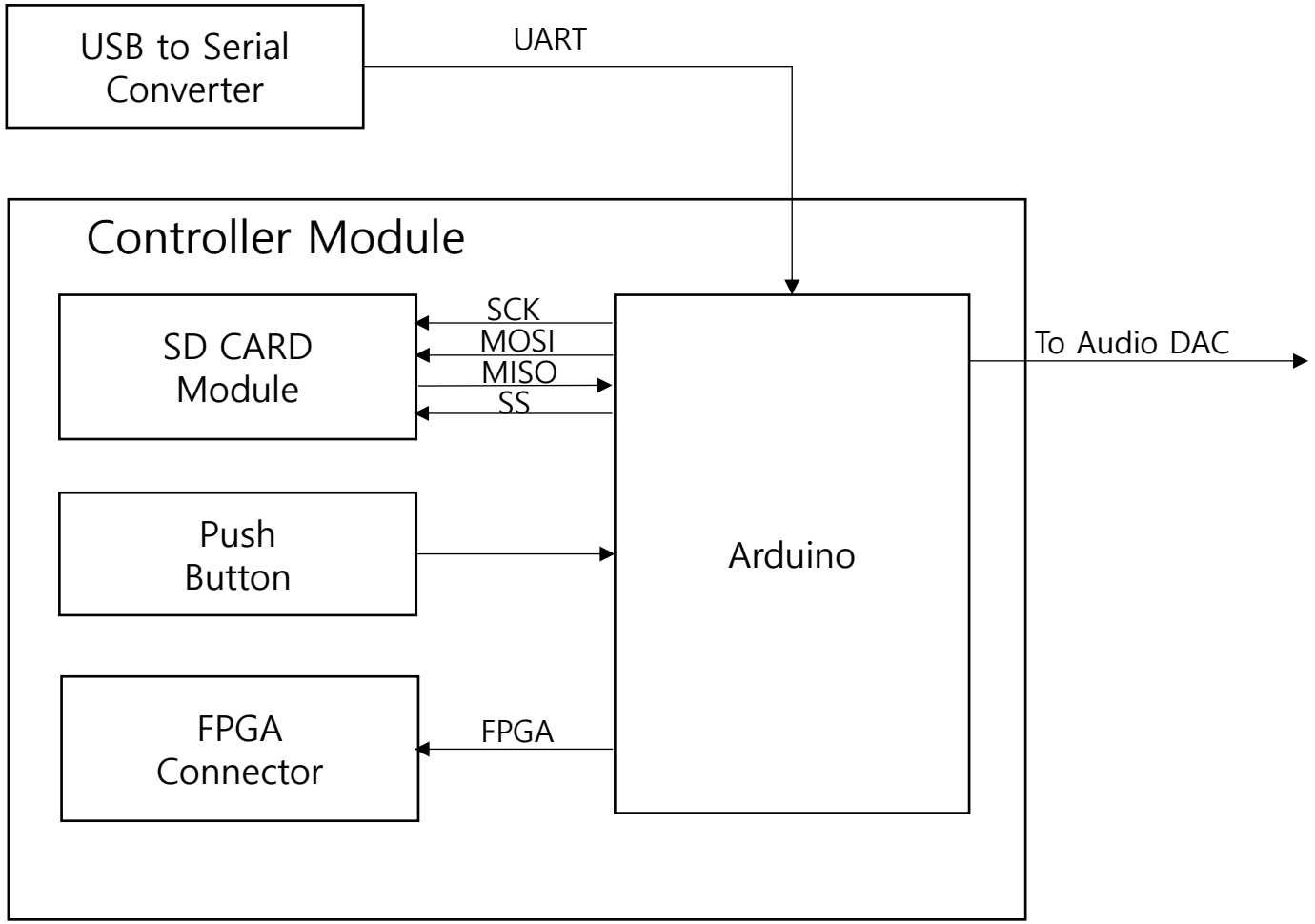


***Designed on OrCAD**

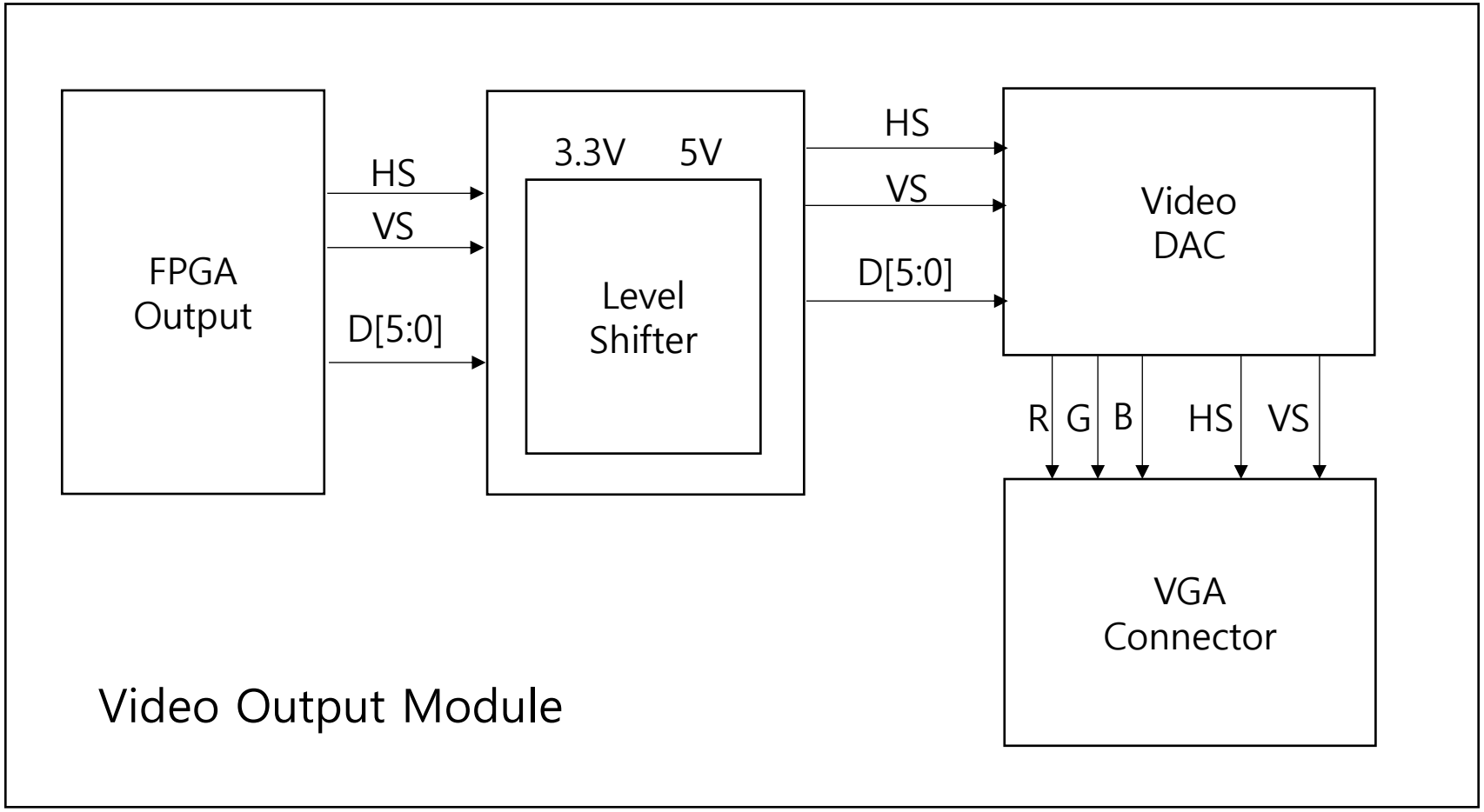
Audio Module



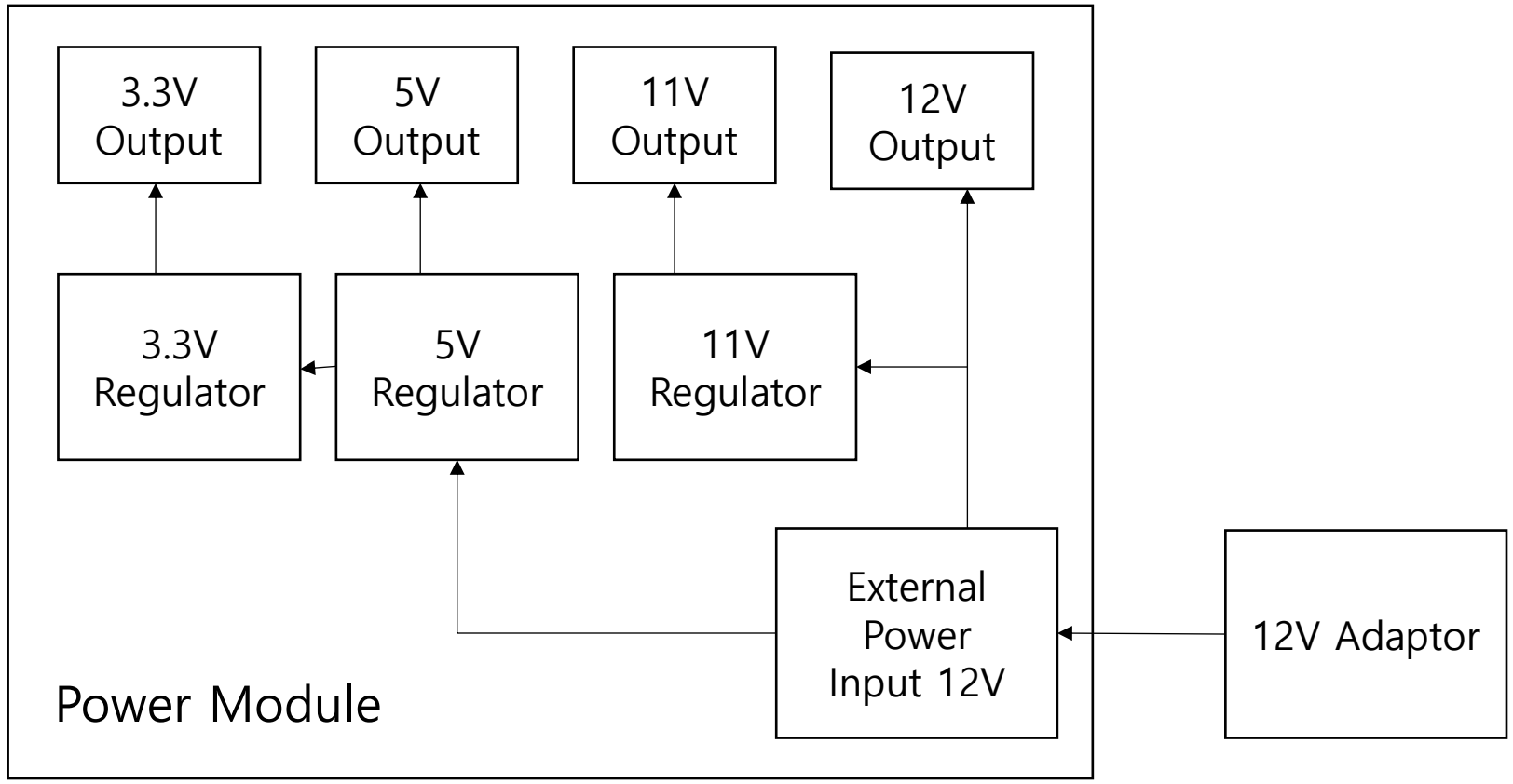
Controller Module



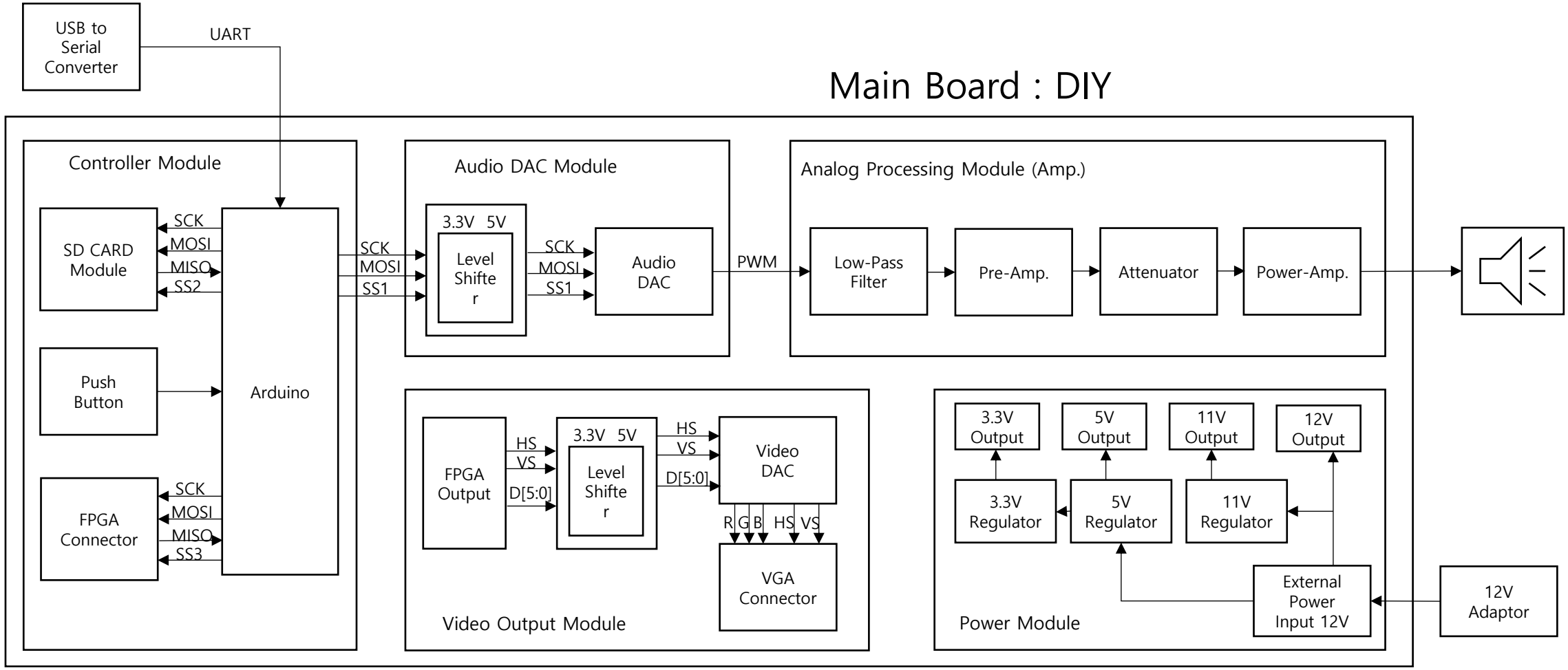
Video Card Module



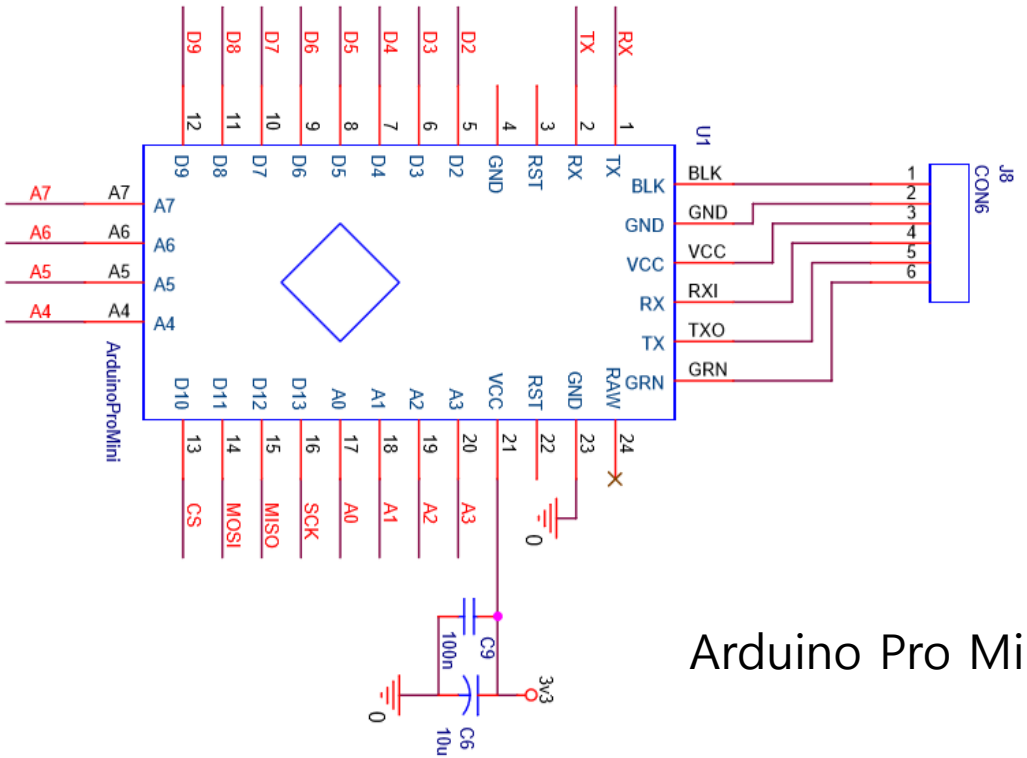
Power Source Module



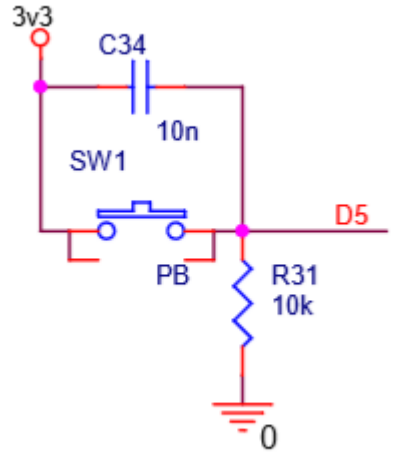
Main Board Diagram



Controller Part

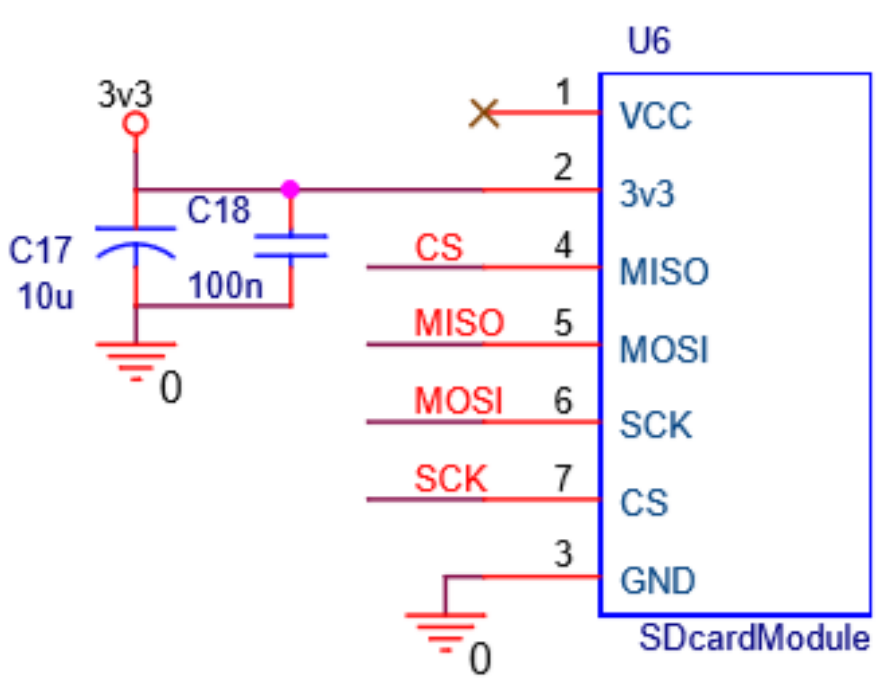


Arduino Pro Mini

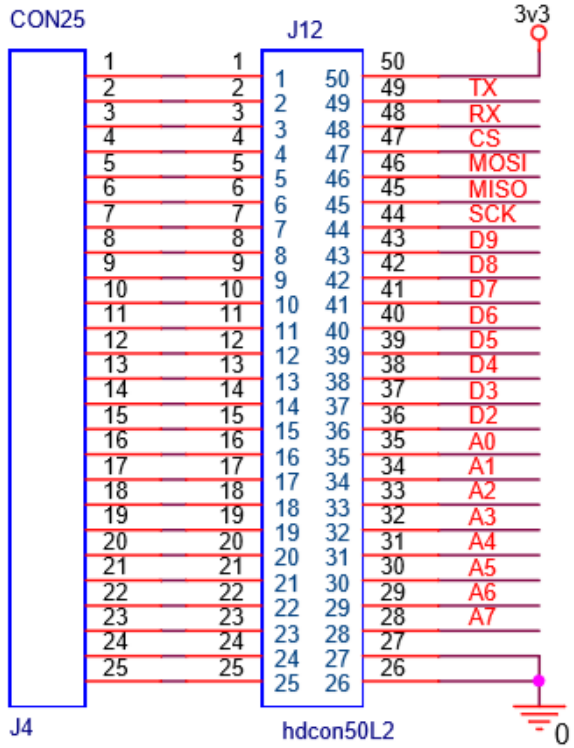


Input Switch

Controller Part

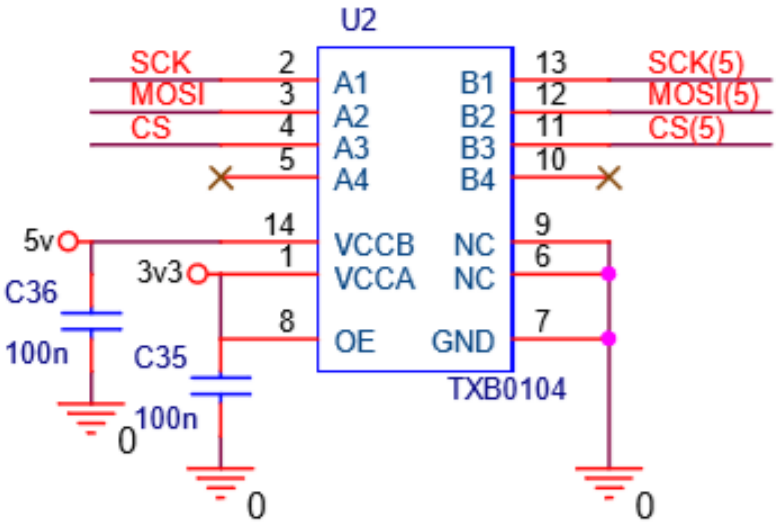


SD-Card Module

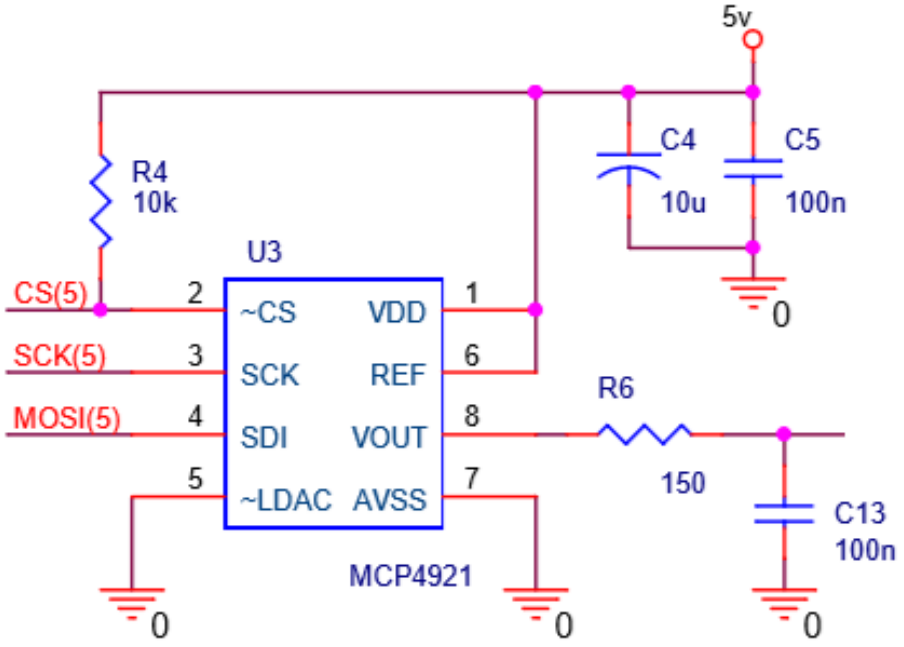


FPGA & Arduino Connector

DAC Part (Digital to Analog Converter)

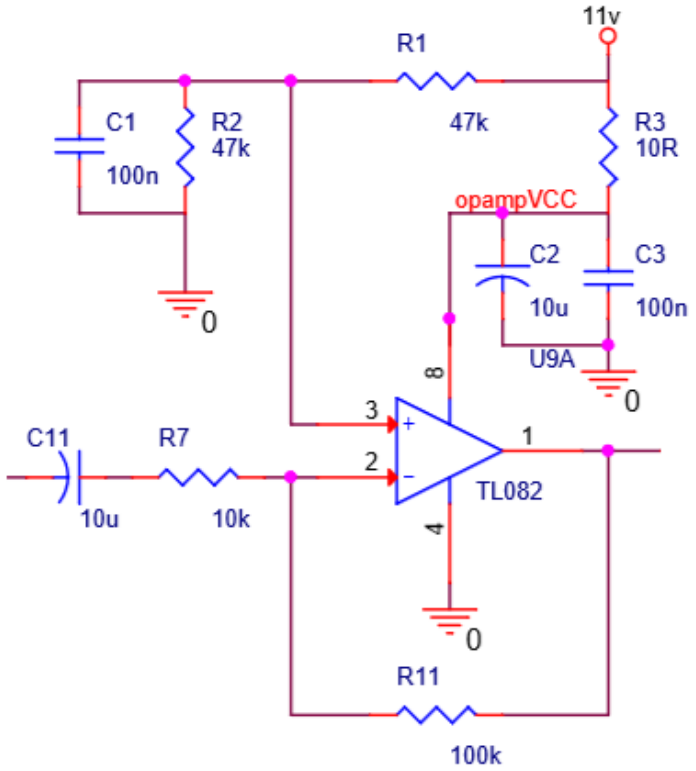


Level Shifting Circuit

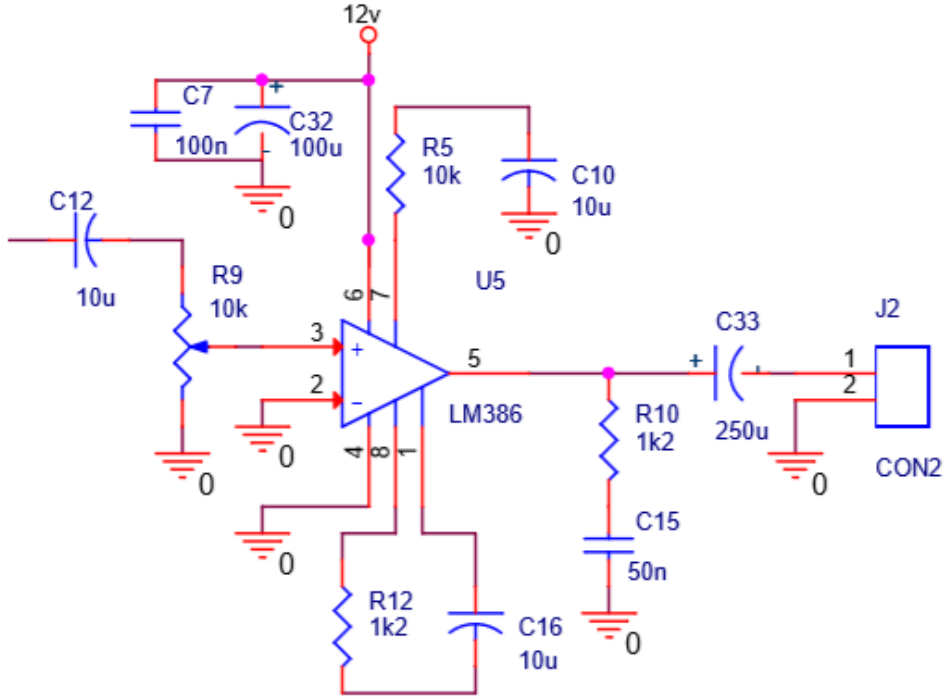


Audio DAC Circuit

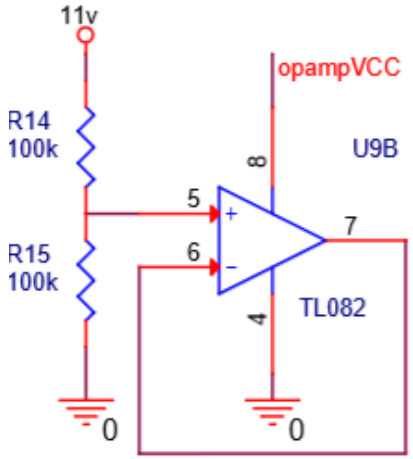
Amplifier Part



Pre - Amp

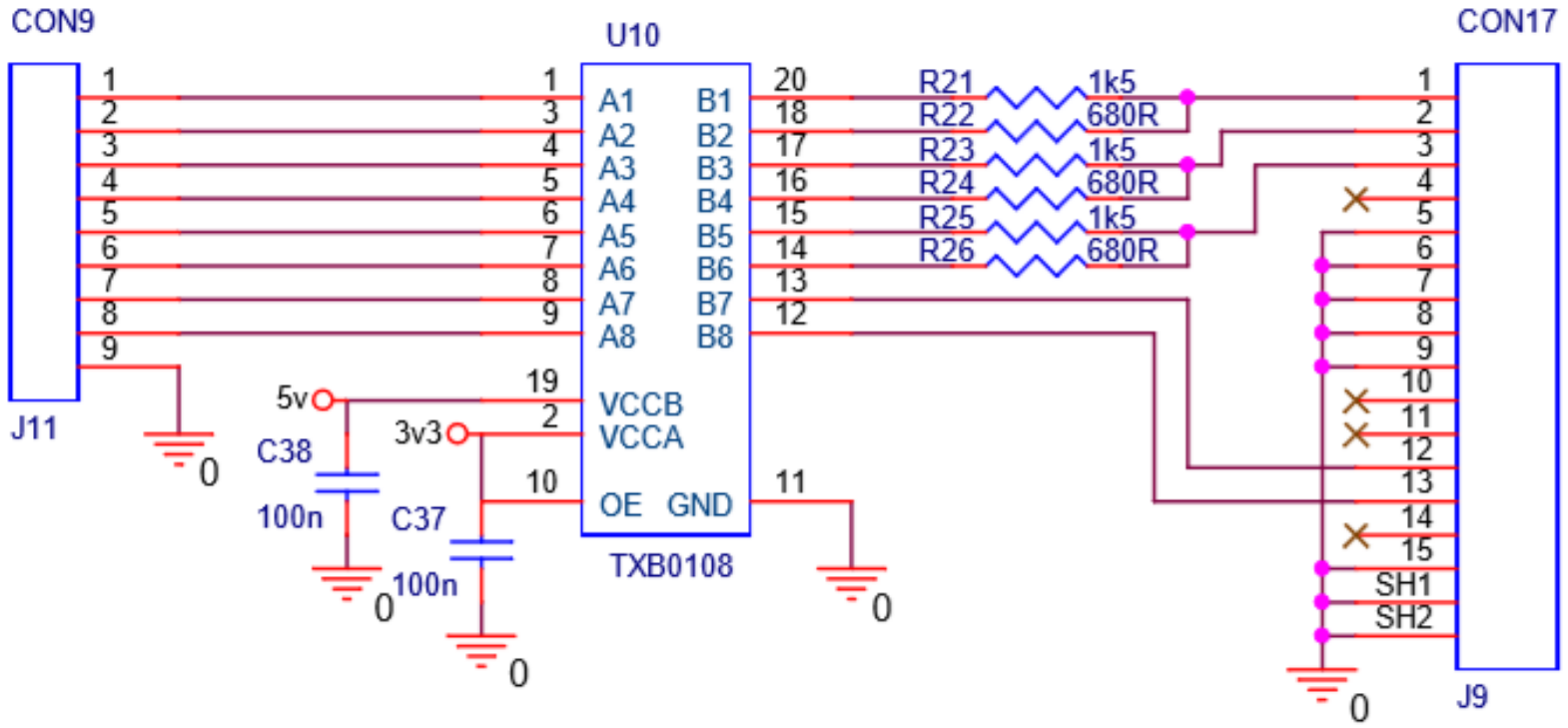


Power Amp



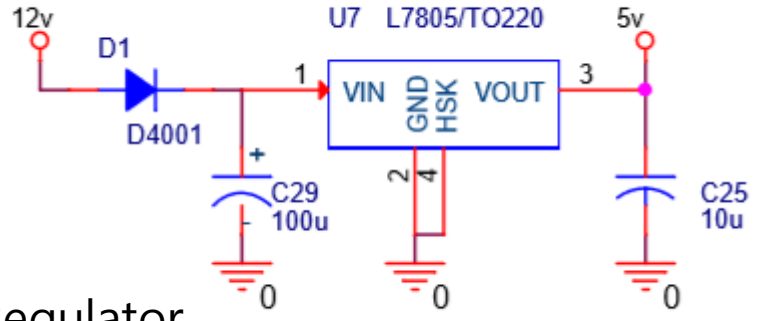
OPAMP(Not used)

Video Card Part

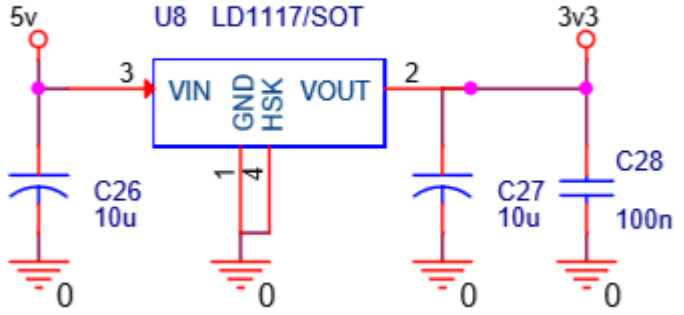


Shifting Level Circuit & VGA DAC

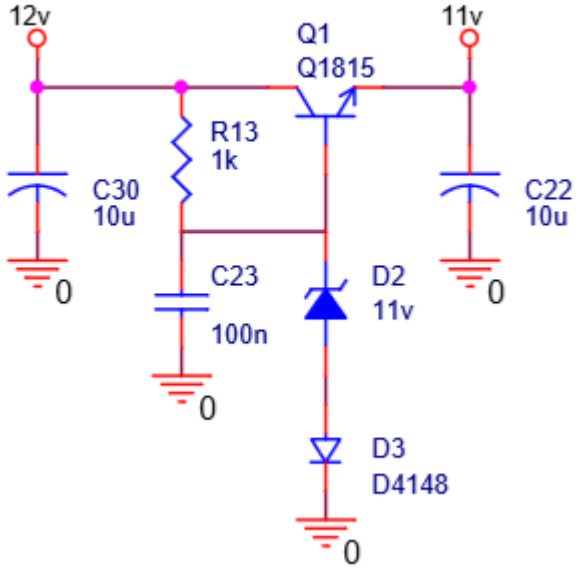
Power Part



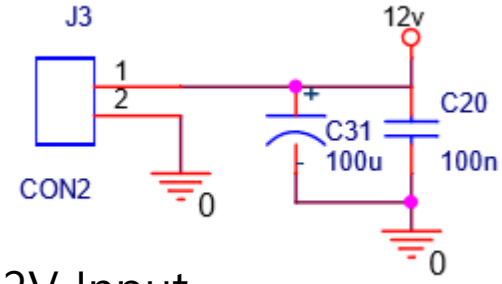
5V-Regulator



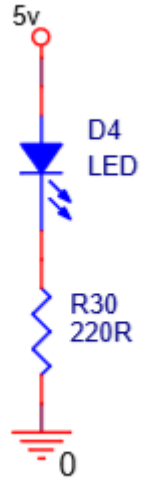
3.3V-Regulator



11V-Regulator

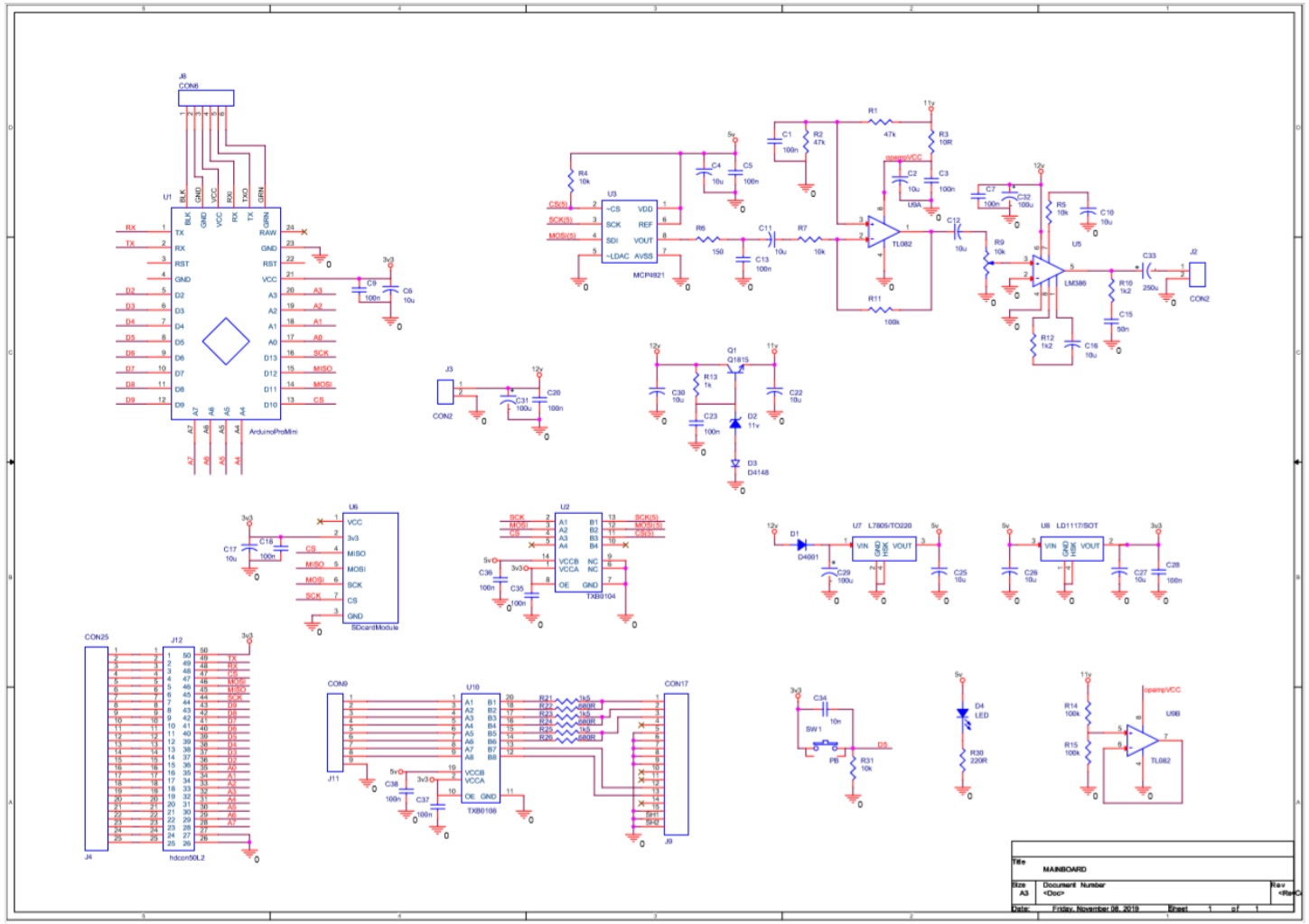


12V Input

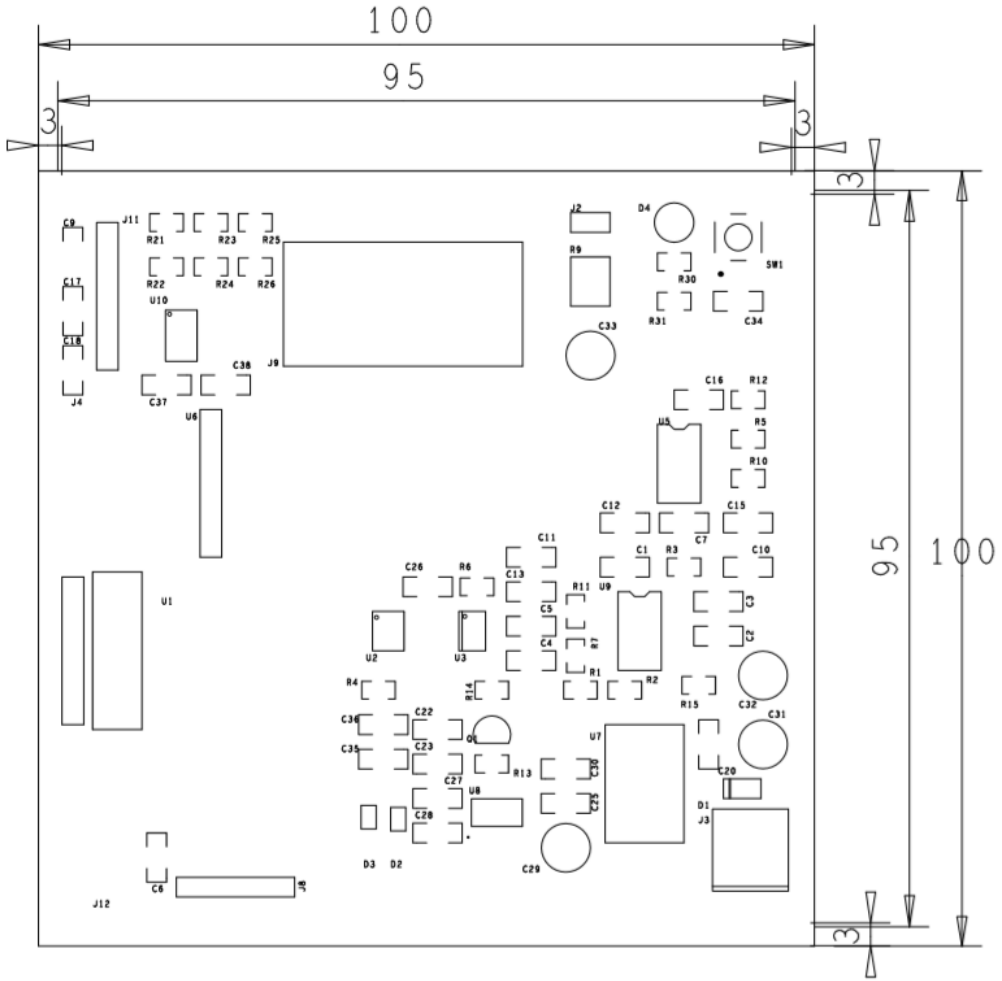


Power Check LED

The Whole Schematic

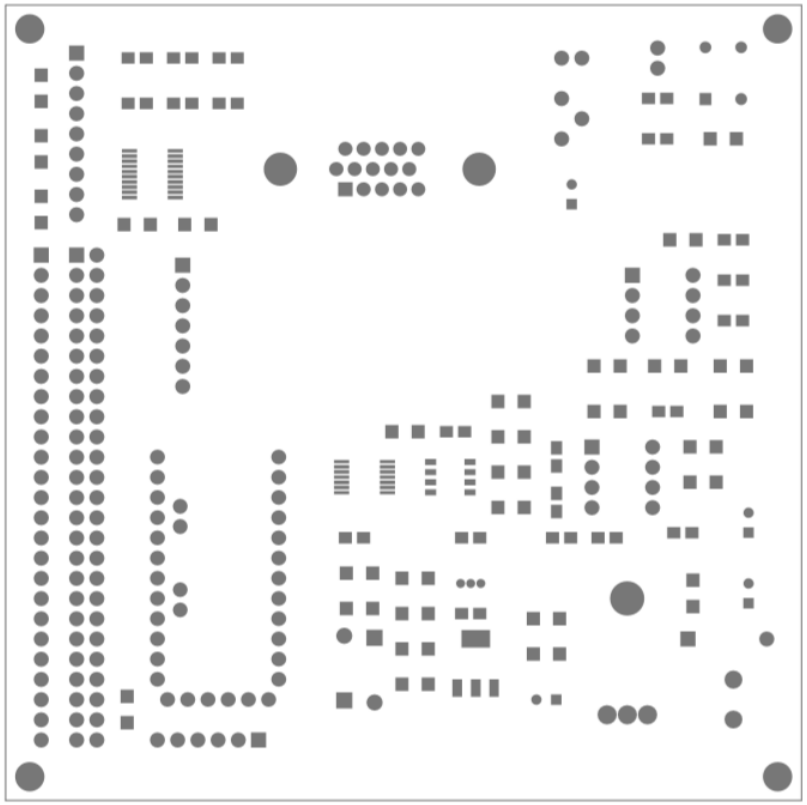


Making Drill Charts & Layer Partitions

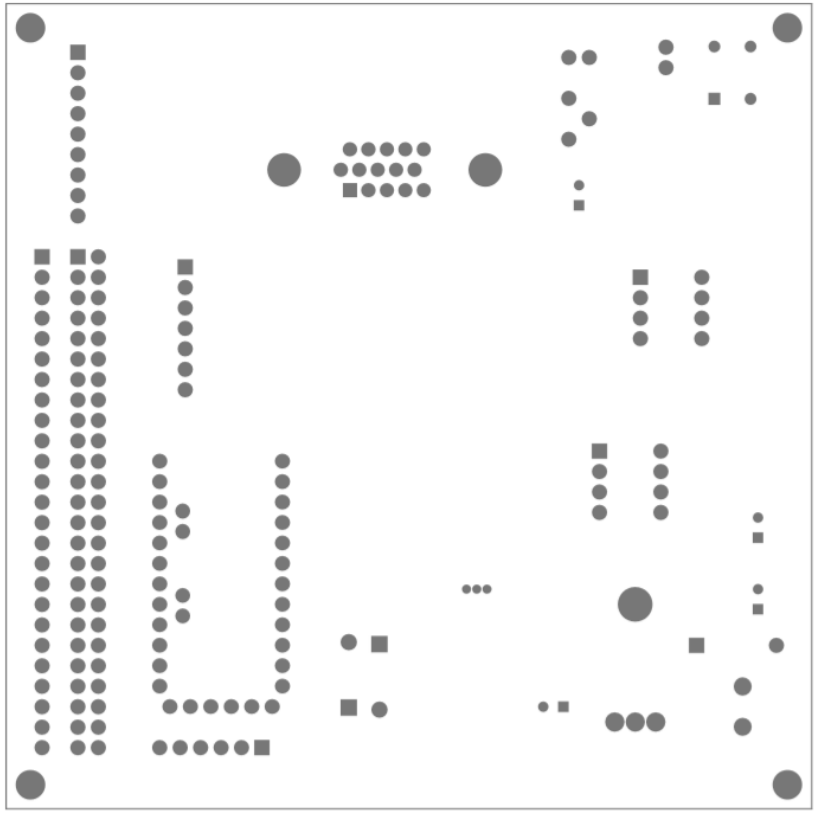


Silk Screen

Making Drill Charts & Layer Partitions

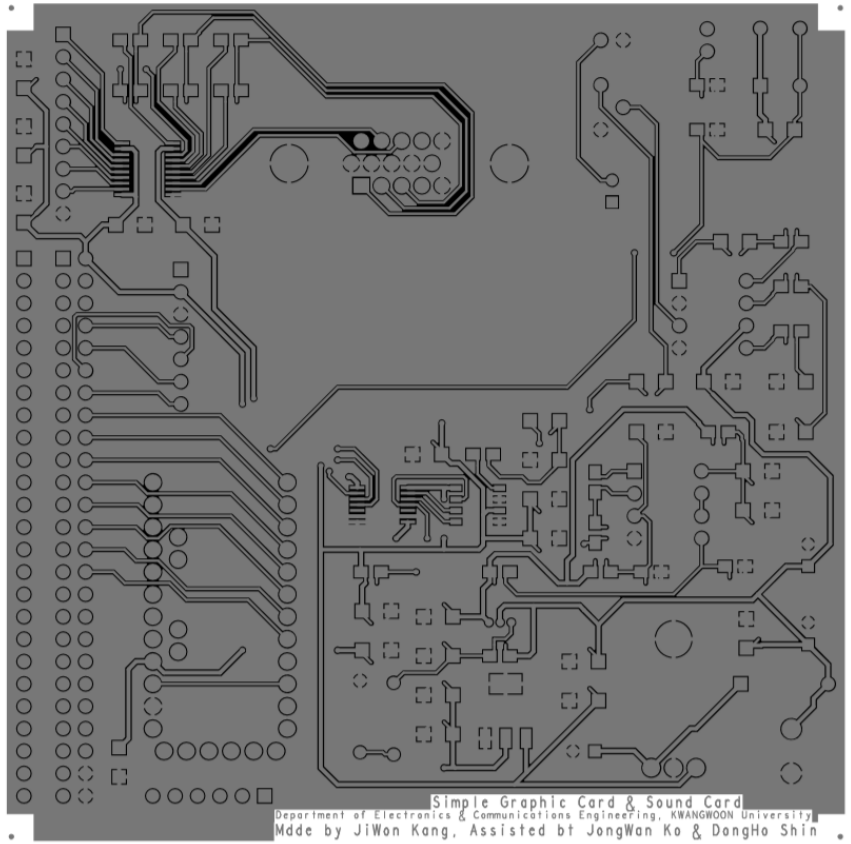


Top Solder-mask

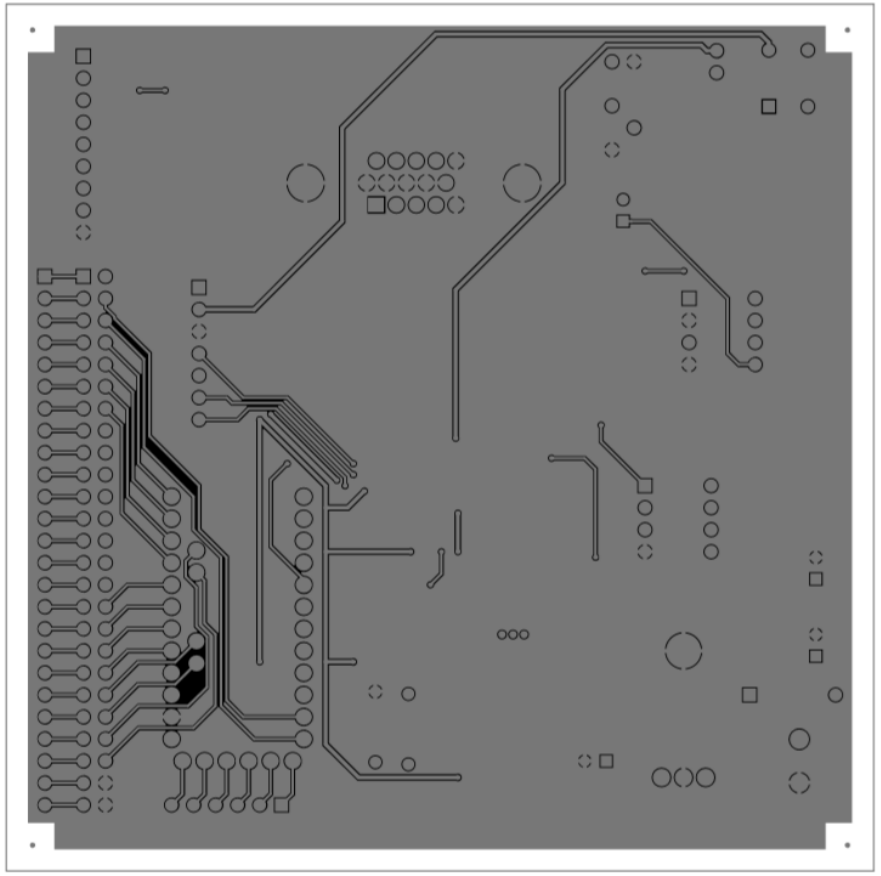


Bottom Solder-mask

Making Drill Charts & Layer Partitions

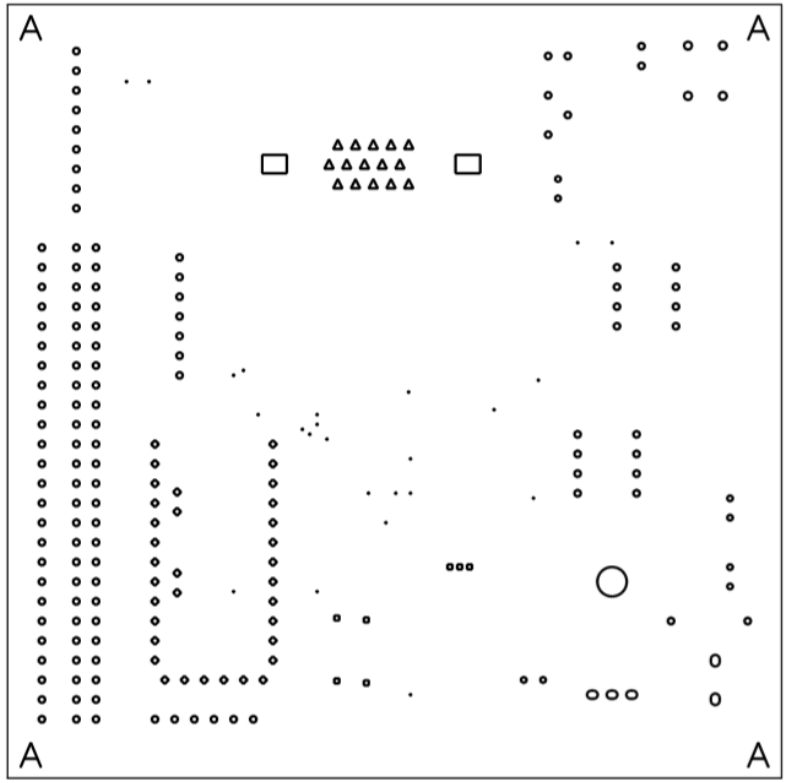


Top Copper



Bottom Copper

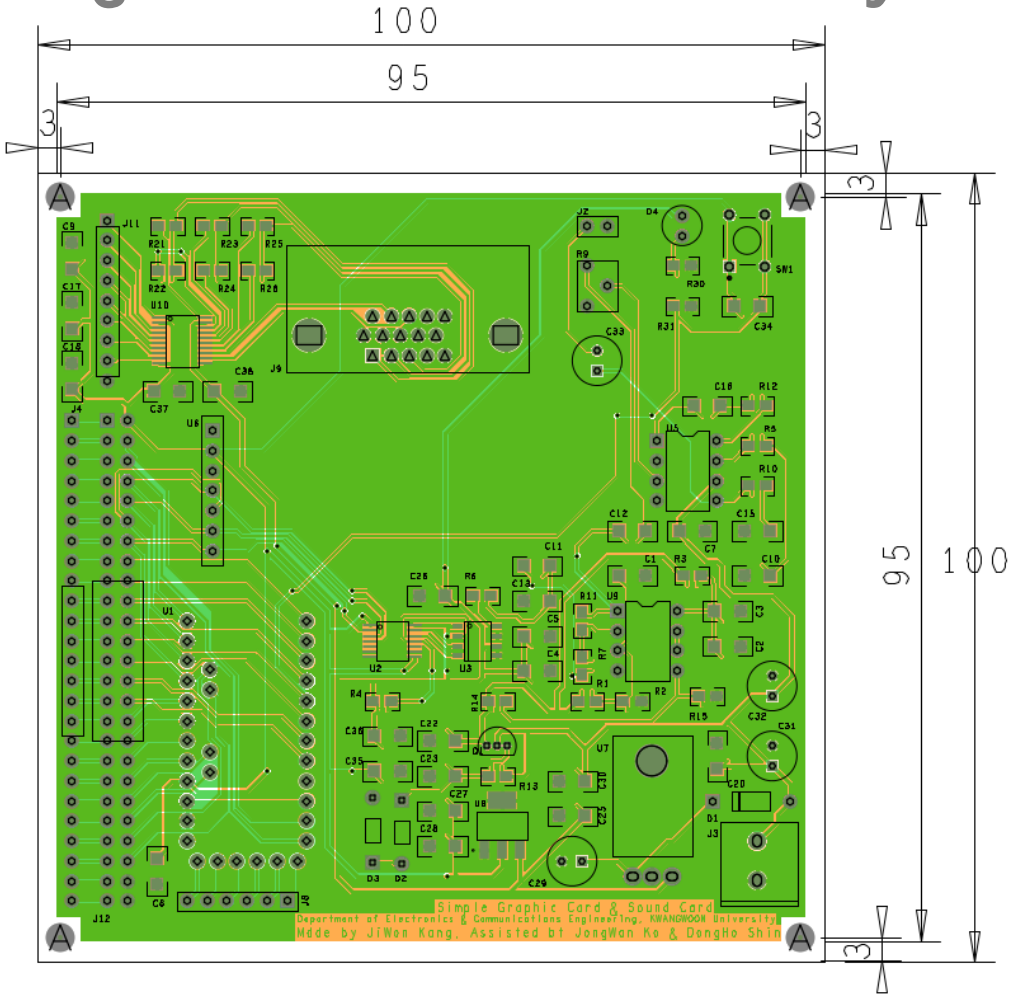
Making Drill Charts & Layer Partitions



DRILL CHART: TOP to BOTTOM			
ALL UNITS ARE IN MILLIMETERS			
FIGURE	SIZE	PLATED	QTY
·	0.3302	PLATED	24
◦	0.635	PLATED	7
◦	0.8001	PLATED	8
◦	0.9144	PLATED	122
◦	0.9901	PLATED	4
◦	1.016	PLATED	34
△	1.19	PLATED	15
○	1.397	PLATED	3
○	1.5001	PLATED	2
□	3.1801	PLATED	2
○	3.81	PLATED	1
A	3.175	NON-PLATED	4

Drill Info.

Making Drill Charts & Layer Partitions



DRILL CHART: TOP to BOTTOM			
ALL UNITS ARE IN MILLIMETERS			
FIGURE	SIZE	PLATED	QTY
·	0.3302	PLATED	24
▫	0.635	PLATED	7
◦	0.8001	PLATED	8
◦	0.9144	PLATED	122
◦	0.9901	PLATED	4
◇	1.016	PLATED	34
△	1.19	PLATED	15
○	1.397	PLATED	3
○	1.5001	PLATED	2
□	3.1801	PLATED	2
◯	3.81	PLATED	1
A	3.175	NON-PLATED	4

All Layer

NC Drill File

```
INTEGER-PLACES          5
DECIMAL-PLACES          3
X-OFFSET                0.000000
Y-OFFSET                0.000000
FEEDRATE                1
COORDINATES             ABSOLUTE
OUTPUT-UNITS            METRIC
TOOL-ORDER              INCREASING
REPEAT-CODES            YES
SUPPRESS-LEAD-ZEROES    NO
SUPPRESS-TRAIL-ZEROES   NO
SUPPRESS-EQUAL          NO
TOOL-SELECT             YES
OPTIMIZE_DRILLING       YES
ENHANCED_EXCELLON       YES
HEADER                  none
LEADER                  12
CODE                    ASCII
SEPARATE                NO
SEPARATE-ROUTING        NO
DRILLING                LAYER-PAIR
BACKDRILL               NO
CAVITY                  NO
```

nc_param.txt

```
0.3302  P  T01    0.000000    0.000000
0.6350  P  T02    0.000000    0.000000
0.8001  P  T03    0.000000    0.000000
0.9144  P  T04    0.000000    0.000000
0.9901  P  T05    0.000000    0.000000
1.0160  P  T06    0.000000    0.000000
1.1900  P  T07    0.000000    0.000000
1.3970  P  T08    0.000000    0.000000
1.5001  P  T09    0.000000    0.000000
3.1801  P  T10    0.000000    0.000000
3.8100  P  T11    0.000000    0.000000
3.1750  N  T12    0.000000    0.000000
```

nc_tools_auto.txt

Not used directly.

NC(Numerical Control) Drill File

```
M48
METRIC
T01C.3302
T02C.635
T03C.8001
T04C.9144
T05C.9901
T06C1.016
T07C1.19
T08C1.397
T09C1.5001
T10C3.1801
T11C3.81
T12C3.175
;LEADER: 12
;HEADER:
;CODE  : ASCII
;FILE  : TEST-1-2.drl for board #Taaaaab03104.tmp  ...  layers TOP and BOTTOM
;T01 Holesize 1. = 0.330200 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 24
;T02 Holesize 2. = 0.635000 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 7
;T03 Holesize 3. = 0.800100 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 8
;T04 Holesize 4. = 0.914400 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 122
;T05 Holesize 5. = 0.990100 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 4
;T06 Holesize 6. = 1.016000 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 34
;T07 Holesize 7. = 1.190000 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 15
;T08 Holesize 8. = 1.397000 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 3
;T09 Holesize 9. = 1.500100 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 2
;T10 Holesize 10. = 3.180100 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 2
;T11 Holesize 11. = 3.810000 Tolerance = +0.000000/-0.000000 PLATED MM Quantity = 1
;T12 Holesize 12. = 3.175000 Tolerance = +0.000000/-0.000000 NON_PLATED MM Quantity = 4
%
```

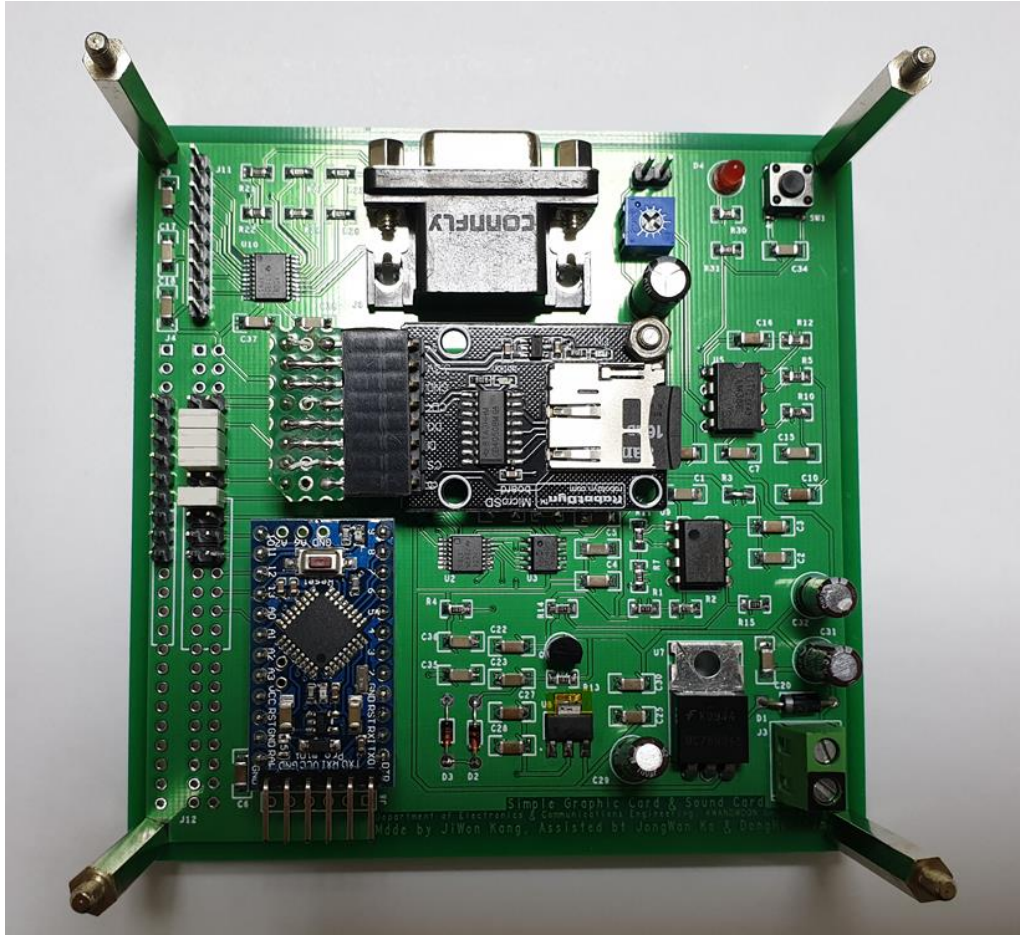
Drill Initializer

```
G90
T01
X-00043942Y00055753
X-00046863Y00055753
X-00031750Y00018415
X-00033020Y00017780
X-00022225Y00012700
X-00029845Y00012700
X-00022225Y00011430
X-00024130Y00010795
X-00023177Y00010160
X-00020955Y00009525
X-00033020Y-00010160
X-00022225Y-00010160
X-00010160Y-00023495
X-00013335Y-00001270
X-00012065Y00002540
X-00015597Y00002540
X-00010160Y00002540
X-00010160Y00006985
X-00010414Y00015621
X00005715Y00001905
X00000635Y00013335
X00006350Y00017145
X00015875Y00034925
X00011430Y00034925
T02
X-00015875Y-00021971
```

Drill Hole Coordinates

Both are in the Same File

Soldering Parts

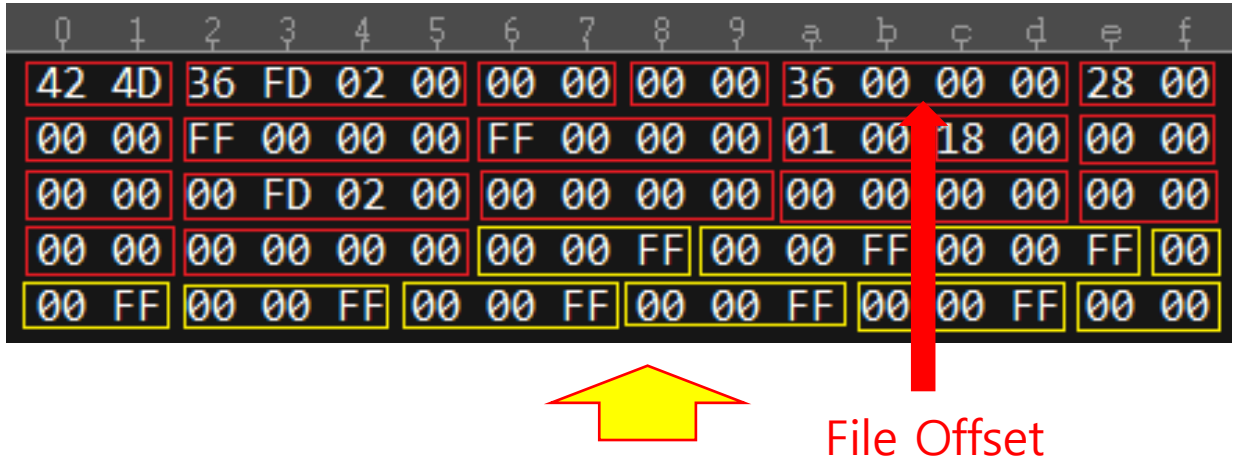


Soldered PCB Top View

Software Development

The Format of BMP

Bmp format file has 54kB bits of header, to make sure the computer understands about the file.

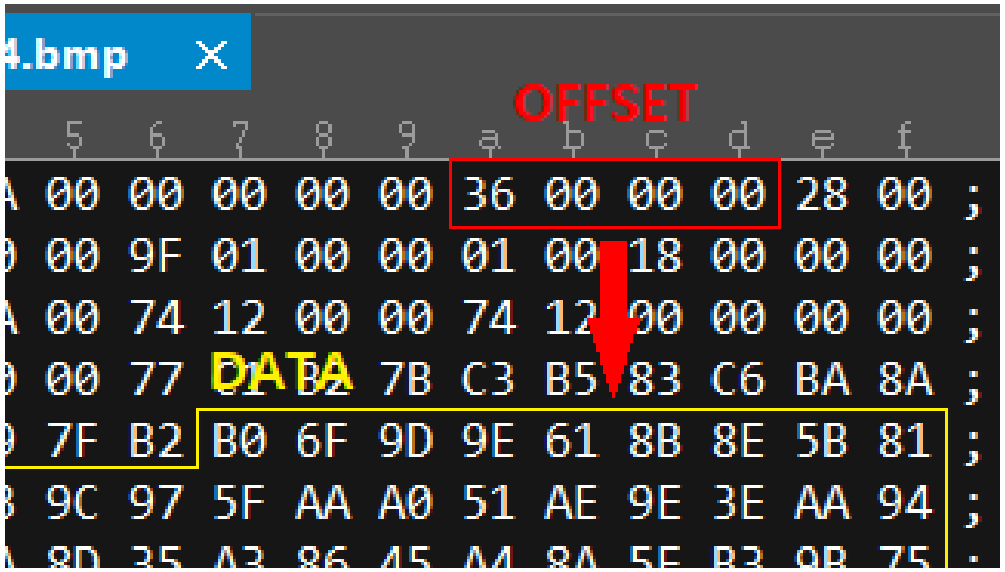


Bmp format saves color data as **BGR**, Not **RGB**

Extract Data from Image File

Extracting Hexadecimal Code by using UltraEditor. And extracting meaningful data by using offset [36 00 00 00].

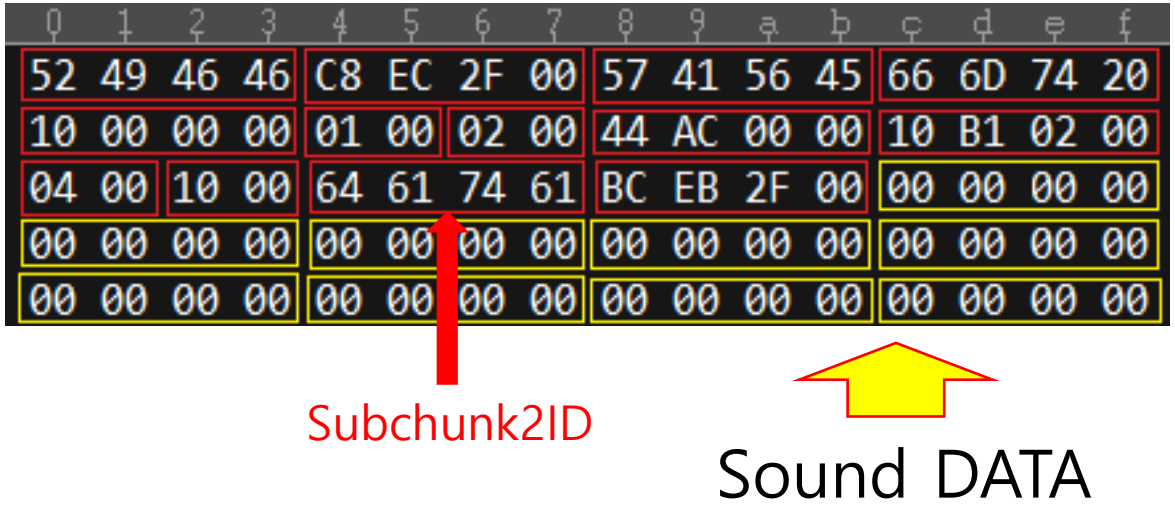
BMP format File



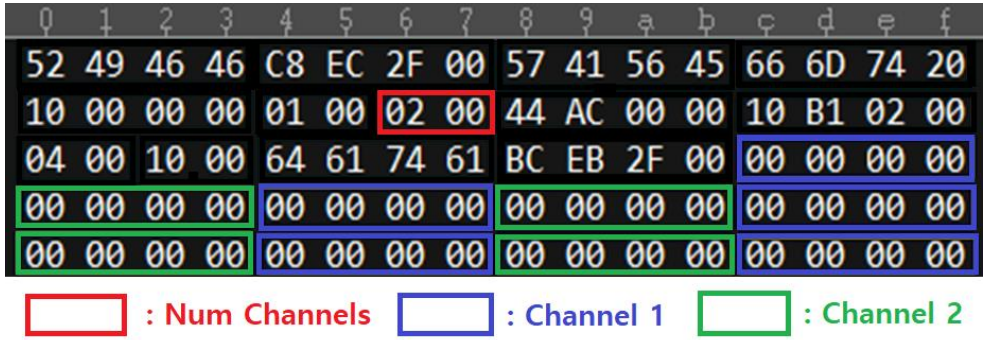
Hexadecimal File

Structure of WAV Format

Wav format file has 44kB bits of header, to make sure the computer understands about the file.

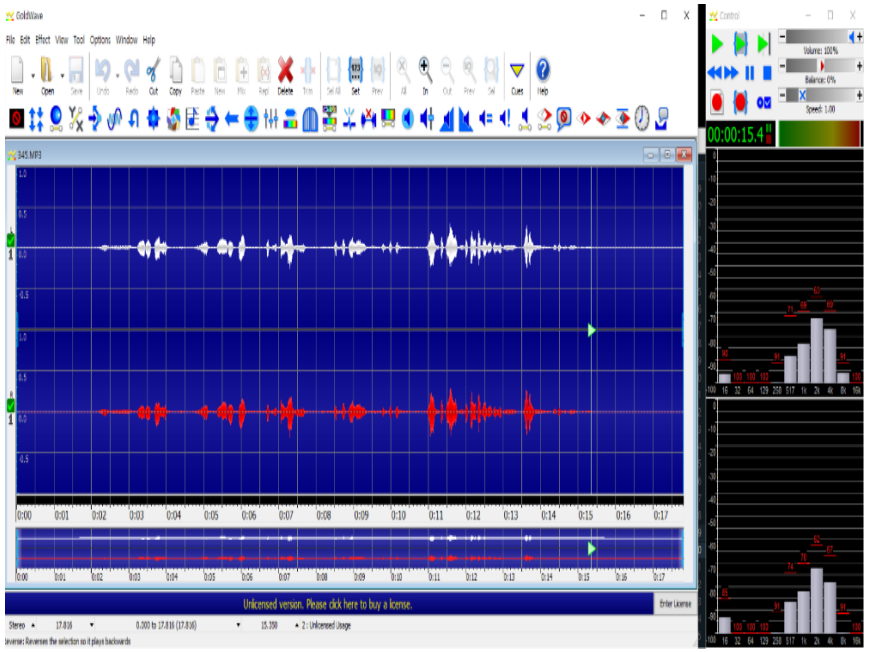


Wav format saves Sound data into two channels. They are different Bmp format from which Bmp files are not sound files in the first place.

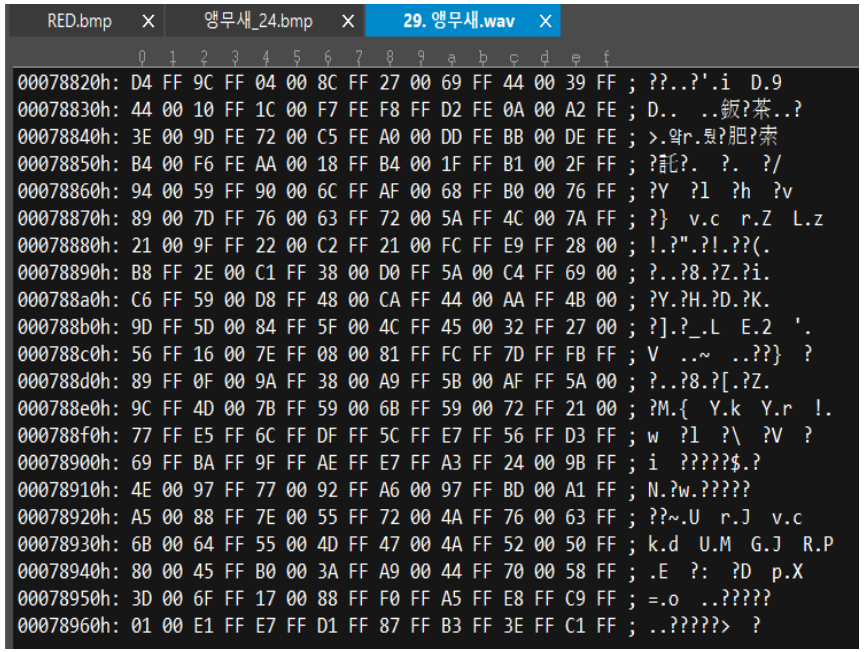


Extract Data from Audio File

Extracting Hexadecimal Code by using UltraEditor. And extracting data effectively by using Subchunk2ID [64 61 74 61].



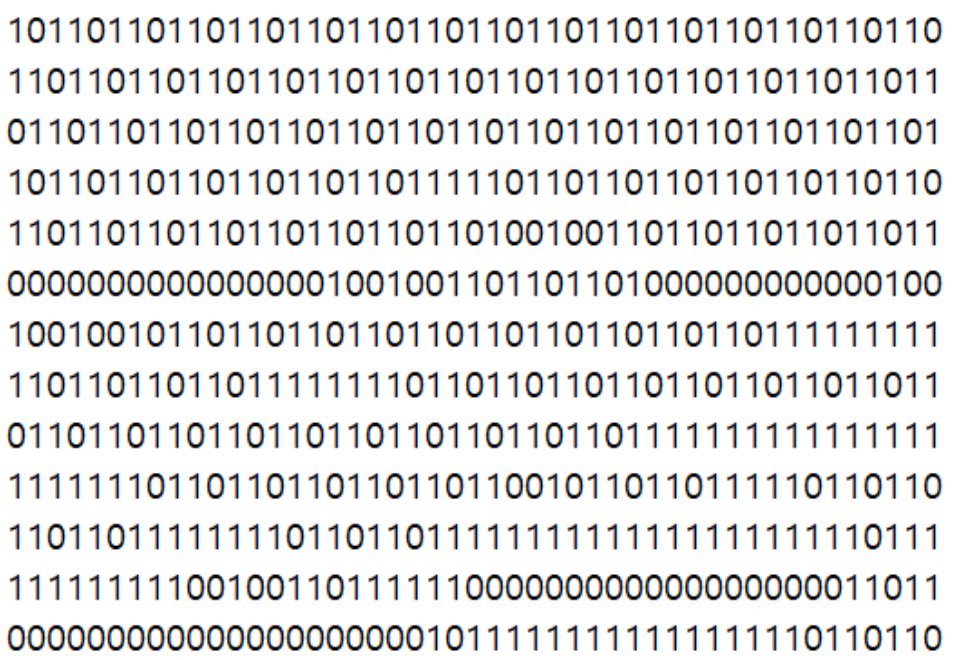
WAV format File



Hexadecimal File

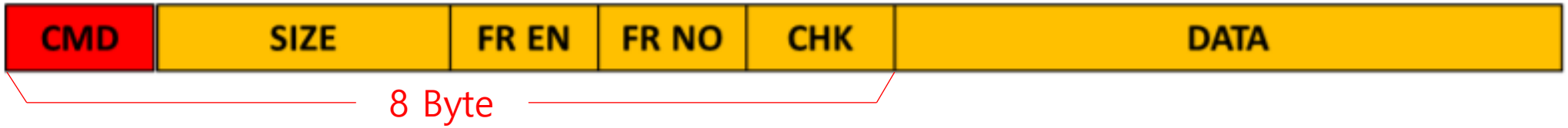
Change Data as Binary

We need to change Data as binary because FPGA & DAC cannot understand if we send message as Hexadecimal numbers.



Communication Protocol on SPI

SPI (Serial Peripheral Interface) has communication protocol to classify SPI communication.



Field	Length (in bytes)	Description
CMD	4	Identifies the command/data packet
SIZE	1	The size of the data in a data packet including 3 bytes for FR EN, FR NO and CHK fields
FR EN	1	Identifies if the data associated with the packet is fragmented
FR NO	1	The fragment number of the packet if the data is fragmented
CHKSUM	1	The checksum on the packet from the 4th byte onwards excluding this byte. Checksum is computed as a sum of the associated bytes.
DATA	Variable	The actual data associated with the packet

SPI Registers

SPI has a lots of registers. We need to know what register can change the SPI protocol.

EIMSK

Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	IINT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	PCIE	–	–	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

SPI Registers – SPCR & SPSR & SPDR

SPCR(SPI Control Register) has 8 bits which works different with each other.

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SPSR(SPI Status Register) has 3 bits which works different with each other.

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

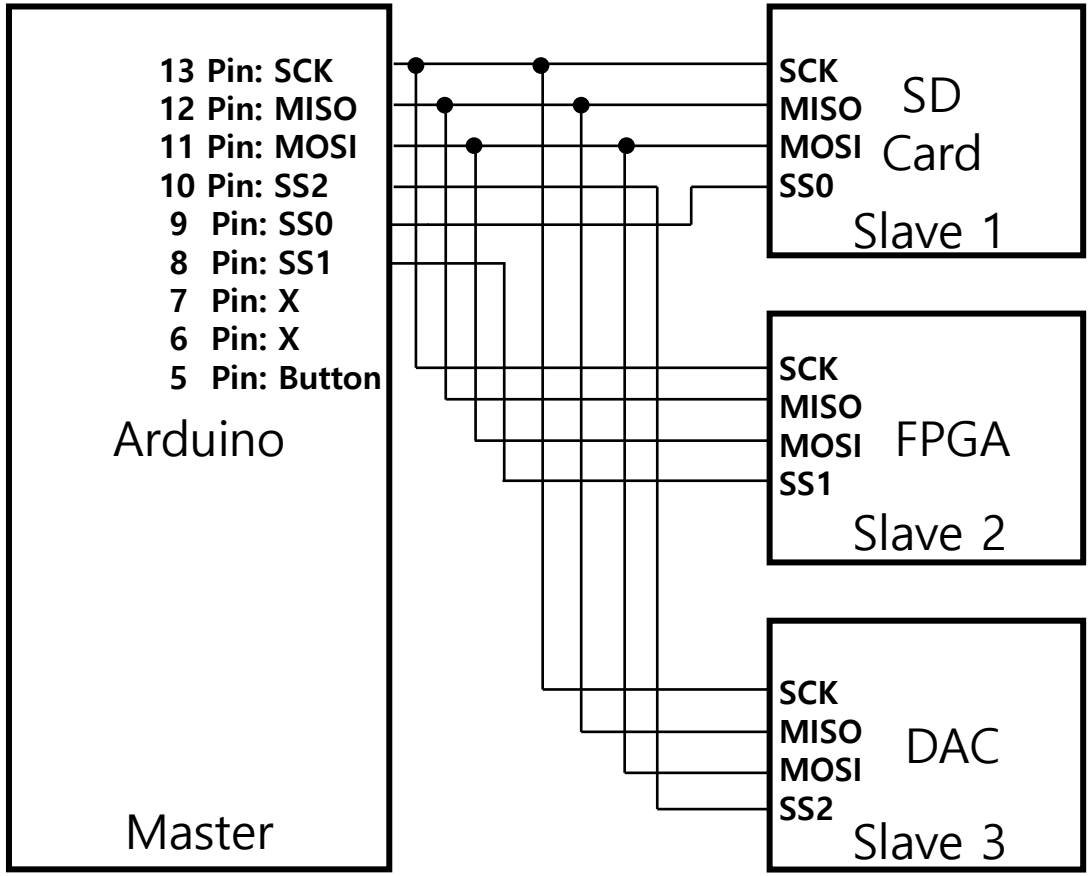
SPDR(SPI Data Register) saves DATA of what we want to transfer.

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Analysis SPI.h header

```
inline static void beginTransaction(SPISettings settings) {   // 통신을 시작할때 데이터 간섭을 허용하고 중복을 방지해줌.
  if (interruptMode > 0) {
    uint8_t sreg = SREG;  // 가장 최근에 사용한 값을 불러옴
    noInterrupts();

    #ifdef SPI_AVR_EIMSK  // 만약 SPI_AVR_EIMSK 값이 선언되어 있다면
    if (interruptMode == 1) { // interruptMOde가 1이라면
      interruptSave = SPI_AVR_EIMSK;  // interruptSave에 아까 SPI_AVR_EIMSK 값을 주고
      SPI_AVR_EIMSK &= ~interruptMask;    // SPI_AVR_EIMSK와 interruptMask의 not 연산값과 &연산을 수행하여 저장
      SREG = sreg;    // SREG에 sreg 값 입력
    } else
    #endif    // 그렇지 않으면
    {
      interruptSave = sreg;   // sreg 값을 그대로 interruptSave에 준다.
    }
  }

  #ifdef SPI_TRANSACTION_MISMATCH_LED // 만약 SPI_TRANSACTION_MISMATCH_LED 값이 선언되어 있다면
  if (inTransactionFlag) {    // flag 값이 true라면
    pinMode(SPI_TRANSACTION_MISMATCH_LED, OUTPUT);    // 지정된 SPI 핀을 데이터 출력 모드로 설정
    digitalWrite(SPI_TRANSACTION_MISMATCH_LED, HIGH); // 데이터를 출력
  }
  inTransactionFlag = 1;  // flag를 1로 설정함
  #endif

  SPCR = settings.spcr;   // SPCR에 위에서 세팅한 spcr 값을 줌
  SPSR = settings.spsr;   // SPSR에 위에서 세팅한 spsr 값을 줌
}

// Write to the SPI bus (MOSI pin) and also receive (MISO pin)
inline static uint8_t transfer(uint8_t data) {    // transfer 함수. 실제로 데이터를 전송하는 함수
  SPDR = data;    // SPDR에 데이터 저장
  asm volatile("nop");    // 타이밍이 틀어질때까지 대기
  while (!(SPSR & _BV(SPIF))) ; // 잠시 대기
  return SPDR; // SPDR 값을 반환함.
}
```

Block Diagram



Arduino & FPGA & DAC Pin Setup

Arduino(Master) – FPGA(Slave1) – DAC(Slave2) – SD card Module(Slave3) Connection.

Total SPI Code

PLEASE_BE_FINAL

```
#include <SD.h>
#include <SPI.h>
#include <math.h>

File bmpFile;
File wavFile;

int SDCARD_SS = 5;
int FPGA_SS = 8;
int DAC_SS = 10;

int Data_Counter = 0; // 데이터가 6비트 모였나 확인하는 공간
int Data_Receiver[8] = {0,}; // 한 바이트를 임시로 저장하는 공간
int Convert_Data = 0; // 한 바이트를 묶어서 임시로 보관하는 공간
int DataNum = 1; // 바이트 변환과정에 필요한 변수

int Total_DATA_Count = 0; // 전체 데이터를 카운트하는 변수
int Total_DATA [200000] = {0,}; // 전체 데이터 파일을 임시적으로 보관하는 변수

int BMPEND = 1; // BMP 파일 전송이 완료되었는지 체크하는 변수
int WAVEND = 1; // WAV 파일 전송이 완료되었는지 체크하는 변수

void setup (void)
{
  SPI.begin (); // SPI 통신 초기화
  digitalWrite(SDCARD_SS, HIGH); // 슬레이브가 선택되지 않은 상태로 유지
  // 안정적인 전송을 위해 분주비를 높여 전송 속도를 낮춤
  SPI.setClockDivider(SPI_CLOCK_DIV4);
  Serial.begin(9600);

  Serial.print("Initializing SD card...");  // SD카드 체크 과정 선언
  if (!SD.begin(4)) {
      Serial.println("initialization failed!"); // SD카드 체크 실패 선언
      return;
  }
  Serial.println("initialization done."); // SD카드 체크 완료 선언

  bmpFile = SD.open("bbny.txt"); // bbny 파일 오픈

  Serial.println("File Open Success");

}
void loop (void)
{
  while(BMPEND < 4)BMPEND += BMPFile();

  for(int i=0; i<200000; i++) Total_DATA[i] = 0; // WAV에도 쓰기위해
  Total_DATA_Count = 0; // WAV에도 쓰기위해

  while(WAVEND < 4)WAVEND += WAVFile();

  return;
}
```

Total SPI Code

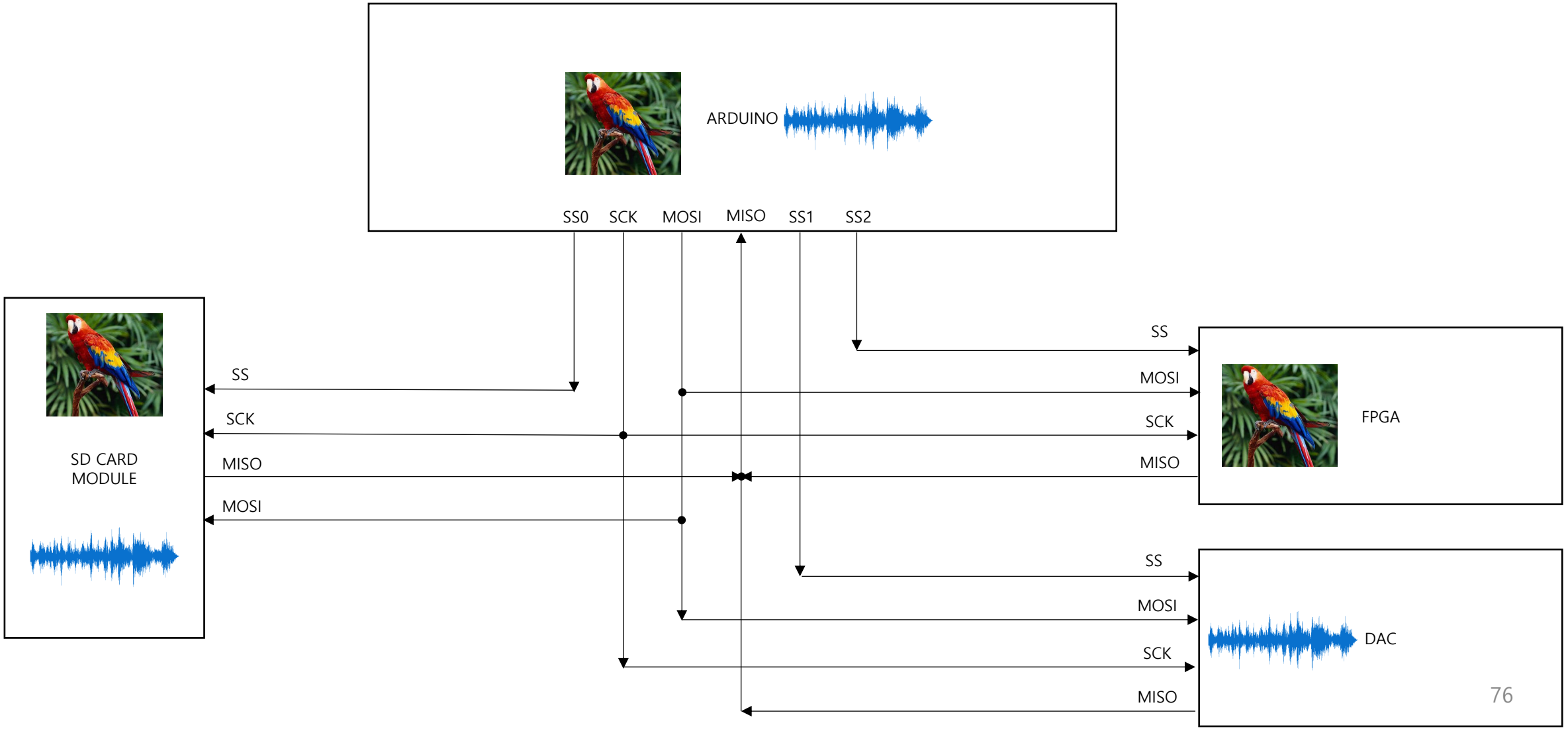
```
int BMPFile(void){  // BMP 파일을 변환하고 전송하는 함수

  Data_Receiver[Data_Counter] = bmpFile.read();
  Total_DATA_Count++;

  if(Data_Counter == 5){  // 만약 6번째 데이터까지 받았다면 이제부터 SD카드에서 얻은 데이터를 전송한다.
    for(int i=0; i<8; i++){  // 총 8번 반복
      if(Data_Receiver[7-i] == 49){
        for(int j=0; j<i; j++)DataNum *= 2;
        Convert_Data += DataNum;  // 받은 2진수 6개 + 더미 2개 총 8개의 데이터를 아스키화
        DataNum = 1;
      }
    }
    Total_DATA[Total_DATA_Count] = Convert_Data;
    Data_Counter = 0;
    Convert_Data = 0;
  }
  else Data_Counter++;

  if(Total_DATA_Count == 200000){
    for(int i=0; i<8; i++) SPI.transfer(0x00);  // 프로토콜
    for(int i=0; i<8; i++) SPI.transfer(0xFF);  // 프로토콜

    for(int i=0; i<200000; i++){
      digitalWrite(FPGA_SS, LOW);  // 슬레이브 선택
      char received = SPI.transfer(Total_DATA[i]); // Total_DATA를 전송하고 Slave의 값을 받아온다.
      digitalWrite(FPGA_SS, HIGH); // 슬레이브 선택 해제
    }
    return 1;
  }
  return 0;
}
```

Total SPI Code

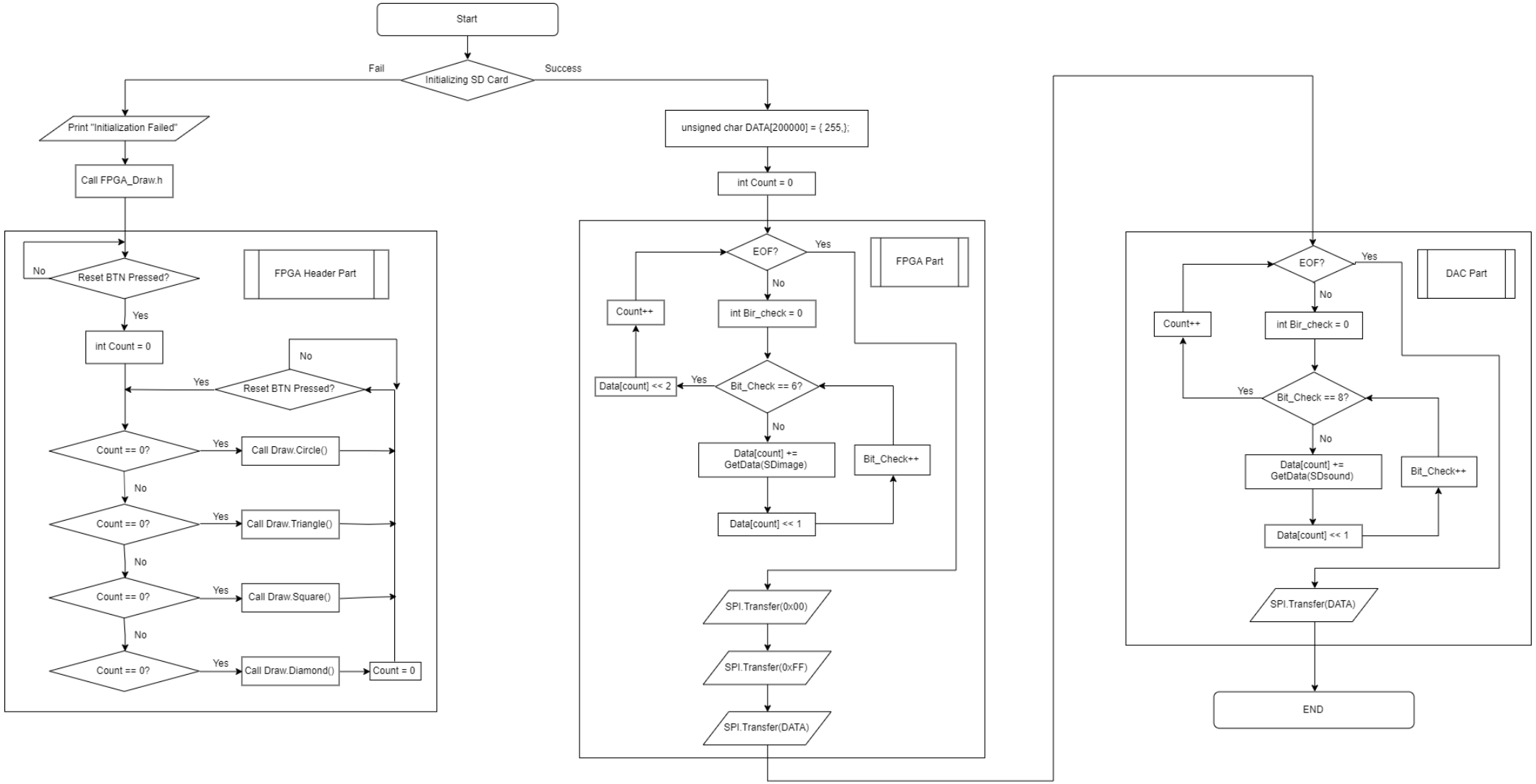
This code is not perfect. Because we don't know what type the DAC needs.

```
int WAVFile(void){  // WAV 파일을 변환하고 전송하는 함수

   Data_Receiver[Data_Counter] = wavFile.read();
   Total_DATA_Count++;

  if(Data_Counter == 5){  // 만약 6번째 데이터까지 받았다면 이제부터 SD카드에서 얻은 데이터를 전송한다.
    for(int i=0; i<8; i++){  // 총 8번 반복
      if(Data_Receiver[7-i] == 49){
        for(int j=0; j<i; j++)DataNum *= 2;
        Convert_Data += DataNum;  // 받은 2진수 6개 + 더미 2개 총 8개의 데이터를 아스키화
        DataNum = 1;
      }
    }
    Total_DATA[Total_DATA_Count] = Convert_Data;
    Data_Counter = 0;
    Convert_Data = 0;
  }
  else Data_Counter++;

    if(Total_DATA_Count == 200000){

      // DAC쪽은 프로토콜이 필요한지 아닌지 알 수 없어서 작성하지 않음. 만약 필요시 여기에 작성

      for(int i=0; i<200000; i++)
      {
        digitalWrite(DAC_SS, LOW);  // 슬레이브 선택
        char received = SPI.transfer(Total_DATA[i]); // Total_DATA를 전송하고 Slave의 값을 받아온다.
        digitalWrite(DAC_SS, HIGH); // 슬레이브 선택 해제
      }
      return 1;
    }
  return 0;
}
```

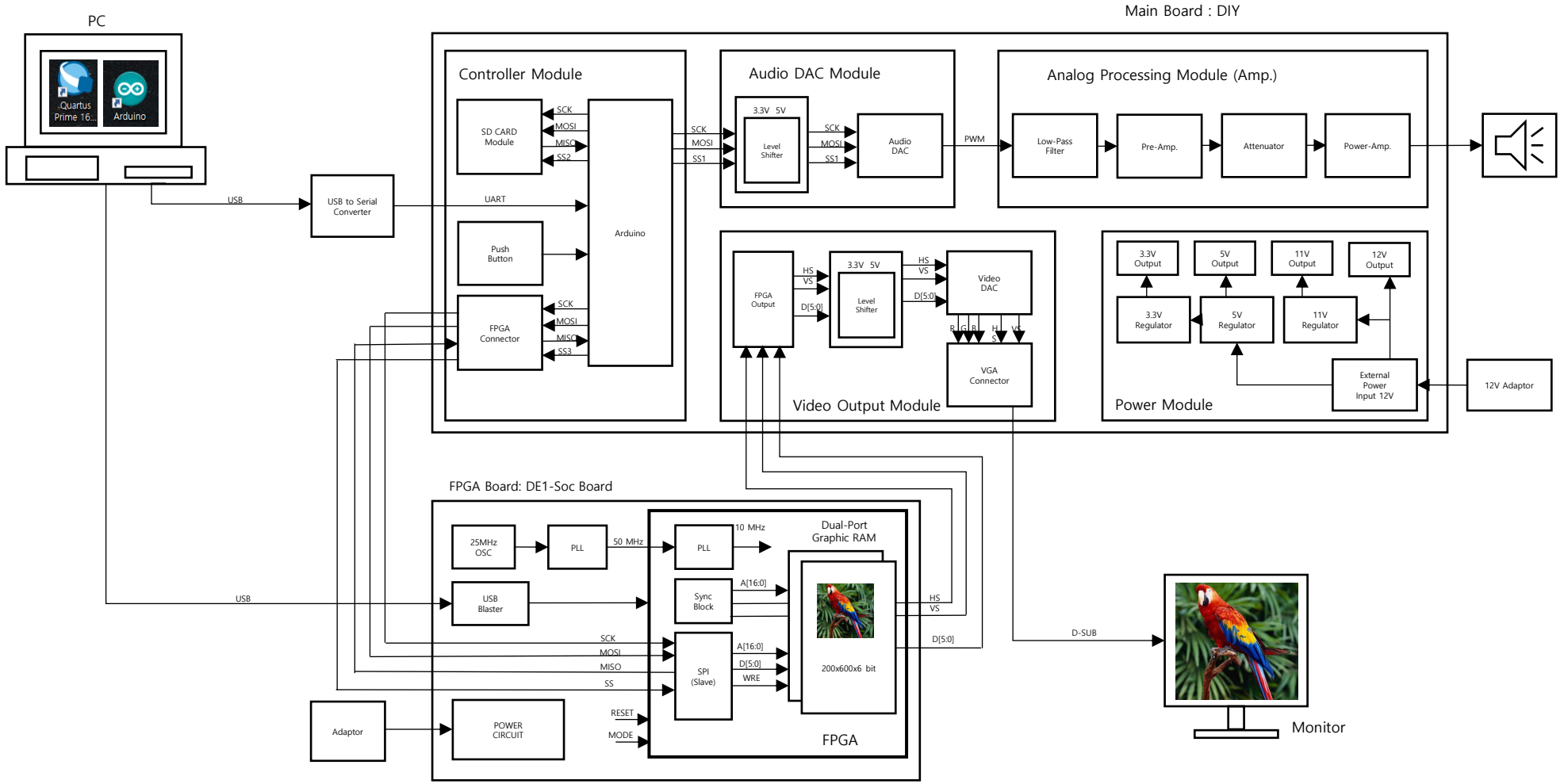
Total Block Diagram – Software Part



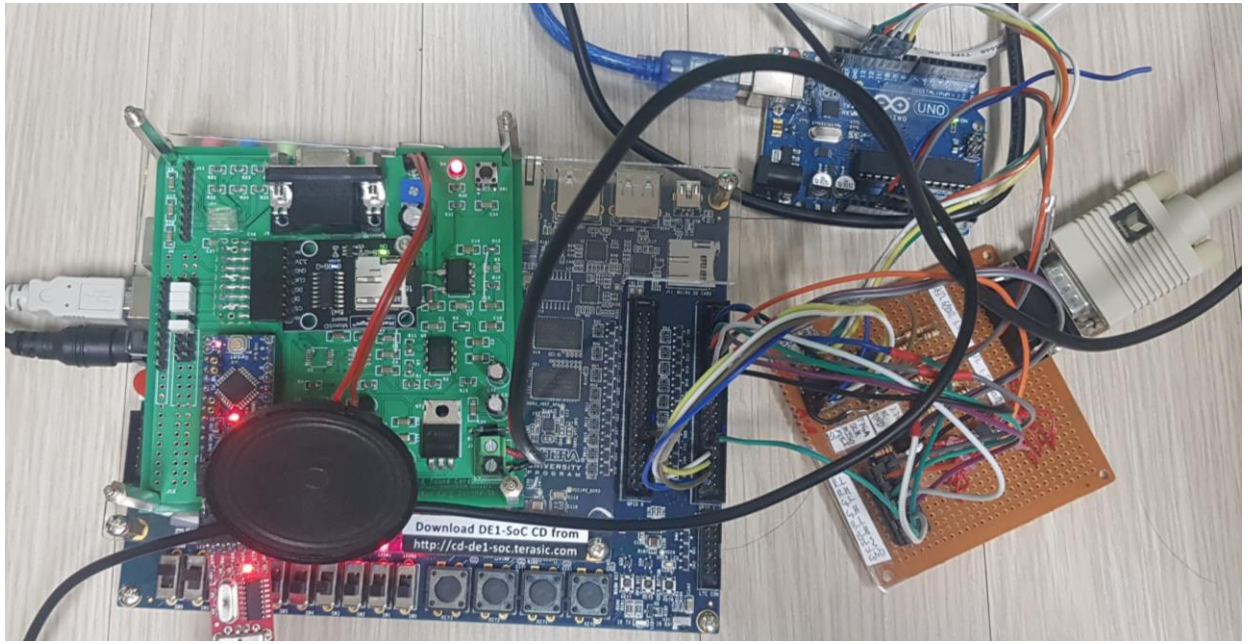
Total Block Diagram – Software Part



5 Summary of the Project



6 Final Test



All Modules Integrated

Test Images

7 Demo

Function List:

1. rgb( ); // prints R G B colors horizontally
2. color( ); // prints colors in a diagonal line
3. fashion ( ); // prints 5 squares with fashion
4. square ( ); // prints a square
5. ship( );// prints a moving ship
6. mcdonald( );// prints the “M” for McDonald
7. faceA( ); // face type A
8. faceB( ); // face type B

With SOUND when 7 & 8 is on screen!

8 Further Improvements

Problems

1. Failed to import picture(Bird Picture) from SD card.

2. External DAC(IC Chip) can't communicate(SPI) with arduino.

3. Arduino Pro mini and SD Card module are very unstable devices.

Alternatives

1. Made a library of Graphical images that can be computed inside Arduino Uno's memory

2. It takes too long to buy another IC chip, thus had no alternatives but to use internal DAC.

3. Will no longer use Arduino series in further projects.

9 Cost

Name	pcs.	Cost
PCB Bread Board (PP-A237)	1	₩ 1100
1/4W (10 SET)	6	₩ 600
Razer Sensor Module (PP-A128)	2	₩ 1000
HEADPIN 1X40 ROHS (11.6m/m)	2	₩ 220
HEADER 1X40 PSS ROHS (2.54m/m)	1	₩ 400
HEADPIN 2X40 ROHS (11.6m/m)	1	₩ 220
HEADER 2X40 PSS (2.54m/m)	1	₩ 620
LM1117S-5V HTC (SOT-223)	10	₩ 1100
LM1117S-3.3 HTC (SOT-223)	10	₩ 1100
TL082 TI (LINEAR)	2	₩ 500

9 Cost (Cont.)

Name	pcs.	Cost
1N5241=1N962A ZENER (11V 0.5W)	10	₩ 400
1/4W (10 SET)	6	₩ 600
JUMPER 2P 2.54 m/m (MJ01-MBG)	20	₩ 200
DSH03-15F ROHS (VGA-ANG)	3	₩ 1350
DC JACK (2.0) (DC005-2)	1	₩ 100
DG127-5.0-02P (HIGHT-10)	1	₩ 260
M-50 (SUPPORT)	4	₩ 560
M-7 (SUPPORT)	4	₩ 240
동판 – 양면 (동판)	2	₩ 4500
RAINBOW CABLE 50C(1M) (0.16/11)	1	₩ 4500

9 Cost (Cont.)

Name	pcs.	Cost
Chip Resistor 2012 size F Level 150Ω	100	₩ 550
Chip Resistor 2012 size J Level 680Ω	100	₩ 550
Chip Resistor 2012 size J Level 1.2kΩ	100	₩ 550
Chip Resistor 2012 size J Level 47kΩ	100	₩ 550
Chip Resistor 2012 size J Level 100kΩ	100	₩ 550
Chip Resistor 2012 size F Level 1MΩ	100	₩ 550
Chip Ceramic 3216 size K Level 82nF 50V	100	₩ 2090
Chip Resistor 2012 size J Level 100Ω	100	₩ 550
Chip Resistor 2012 size J Level 10kΩ	100	₩ 550
Chip Resistor 2012 size J Level 1kΩ	100	₩ 550

9 Cost (Cont.)

Name	pcs.	Cost
Chip Resistor 2012 size F Level 2.2kΩ	100	₩ 550
Chip Resistor 2012 size J Level 220Ω	100	₩ 550
Chip Resistor 2012 size J Level 2kΩ	100	₩ 550
Chip Resistor 2012 size J Level 3.3kΩ	100	₩ 550
Chip Resistor 2012 size J Level 4.7kΩ	100	₩ 550
Chip Resistor 2012 size F Level 470Ω	100	₩ 550
TXB0104PWR	1	₩ 1430
LMK316F106ZF-T (REEL CUT)	100	₩ 2200
Chip Resistor 2012 size J Level 10Ω	100	₩ 550
MCP4921T-E/SN	1	₩ 3091

9 Cost (Cont.)

Name	pcs.	Cost
74HC245 (DIP)	4	₩ 1760
1/4W 5% Axial Resistor 152J (1.5kΩ)	10	₩ 110
1/4W 5% Axial Resistor 681J (680Ω)	10	₩ 110
Chip Resistor 2012 size J Level 1.5kΩ	100	₩ 550
TXB0108PWR	2	₩ 4268
Kapton Tape 15mm x 33m	1	₩ 6600
Raspberry Pi 3 Model B + Heat Sink	1	₩ 45100
SPI MicroSD Card Module (RD097)	1	₩ 1210
Total		₩ 99639

10 References

- https://en.wikipedia.org/wiki/WAV
- https://en.wikipedia.org/wiki/Field-programmable_gate_array
- https://en.wikipedia.org/wiki/BMP_file_format
- https://m.blog.naver.com/ksseo63/221427433909
- https://cafe.naver.com/seogarae/1424
- https://cafe.naver.com/carroty/12008
- https://cafe.naver.com/carroty/12008
- https://mh-nexus.de/en/hxd/
- https://en.wikipedia.org/wiki/Verilog
- https://en.wikipedia.org/wiki/Digital-to-analog_converter
- https://en.wikipedia.org/wiki/Amplifier
- https://en.wikipedia.org/wiki/Lossless_compression
- https://en.wikipedia.org/wiki/MP3
- https://en.wikipedia.org/wiki/Video_card
- https://en.wikipedia.org/wiki/Sound_card
- https://blog.naver.com/okkju/130071016175

11 Q & A

Q & A