Shooting Game

Components of Basys3 board Used:

1. PModOLED:

The PmodOLED uses a standard 12-pin connector to display output on a 128x32 pixel organic LED (OLED) panel. The graphic display panel uses the Solomon Systech SSD1306 display controller. An SPI interface is used to configure the display, as well as to send the bitmap data to the device. The PmodOLED displays the last image drawn on the screen until it is powered down or a new image is drawn to the display. Refreshing and updating is handled internally. The PmodOLED has two field-effect transistors (FETs) that control the display's two power supplies. The VDDC control toggles the power to the logic of the display. The VBATC control toggles the power to the OLED display itself.
These two pins have pull-up resistors that turn off their respective power supplies when they're not being driven. These pins are configured as outputs and driven low to turn on the power supply. The PmodOLED has a particular power-on/power-off sequence that must be followed to prolong the life of the display.

Power-on sequence:

1. Apply power to VDD.
2. Send Display Off command.
3. Initialize display to desired operating mode.
4. Clear screen.
5. Apply power to VBAT.
6. Delay 100ms.
7. Send Display On command

S.No.	Step	Explanation
1	Apply power to VDD.	Make VDDC signal (pin 10 in interface) '0'
2	Send Display Off command.	Display Off commandis AEh.
3	Initialize display to desired operating mode.	Make RES signal (pin 8 in interface) '0' for at least 3µs (microseconds) and then make it '1'.
4	Clear screen.	
5	Apply power to VBAT.	Make VBATC signal (pin 9 in interface) '0'
6	Delay 100ms.	Wait for 100 ms (milliseconds).
7	Send Display On command.	Display On command is AFh.

Shooting Game

Display memory

The signals driving the 128 columns and 32 rows of the display are referred to as segments (SEG0 to SEG127) and commons (COM0 to COM 63). The graphic display memory (GDDRAM) in SSD1306 stores a 128x32 bit map to be displayed. The memory is divided into 4 "pages" (PAGE0 to PAGE3) of size 128x8. Thus, a page corresponds to 8 rows of the display which may be used for one text line, if the information we propose to display contains text. Data and commands are sent to SSD1306 through SPI interface. Signal D/C (pin 7 in interface) indicates whether data is being sent or command (D/C = '0' for command and '1' for data).

SSD1306 commands:

We would need to use these commands primarily to set the memory addressing modes and range of addresses. There are 3 addressing modes, selectable by command 20h – (i) Page addressing mode, (ii) Horizontal addressing mode and (iii) Vertical addressing mode.

If We propose to fill the entire memory, We may set the addressing mode to Vertical or Horizontal (Page mode is the default), leaving the *column start* and *column end* addresses to their default values (00h and 7Fh), but setting the *page start* and *page end* addresses to 00h and 03h using the command 22h since there are only 32 rows in our display. After doing this, data for 128x32 pixels (512 bytes) can be sent across the SPI interface. As and when the information to be displayed changes, modified data can be sent to the display device. If only a part of the display is to change, data for an appropriate rectangular region only may be updated by setting the column and page start and end addresses accordingly.

2. Seven segment display:

There are four displays as shown. Each of these four consists of 7 LEDs (Light Emitting Diodes) forming 7 segments. These diodes have a common anode and individual cathodes. To display a digit, it is required to give a '1' as input to the anode and a '0' or '1' to each segment depending upon whether that segment needs to be lighted ('0') or not ('1').

Initially we used the seven segment display to display the name of our game "SHOT" , then after the game starts the seven segment display is used to display the score of the player after every screen passes by, i.e., at the end of the game the seven segment display displayed the total score of the player.

Shooting Game

3. LEDs:

The LEDs present on the Basys board are given a value '1' when we want to light them up and '0' otherwise.

In our project, we used the LEDs to display the progress of our game. The game contains a total of 16 different screens, which is equal to the total number of LEDs present on the board. When the first screen is displayed, the first LED lights up, and at the time the second-screen displays, the first and the second LED lights up together, and so on.

4. Switches and Push buttons: first three switches correspond to low, medium and high level. On the second screen, we would be asked to select the mode among low, medium and high.

Descriptions of push buttons are given below:

a. Centre Button : Reset
b. Up button : PB1
c. Right Button : PB2
d. Down button : PB3
e. Left button : PB4

PROCESS :

The controller file contains two components. First component is to initialize the pmodOLED and the second one to run the code. So first of all, we start the pmodOLED by running the first component (init). This gives an output signal to the second component which works as a catalyst and provokes the second component. Description of these init and example components are given below:

1. **Init component :** It takes a signal from controller and gets activated. Two subcomponents are used in this. First is delay, and second is SPIinterface. Init component basically initialize the OLED by giving power-on sequence to the OLED. Description of the two subcomponent is given below :
    a. **Delay :** It takes input in milliseconds and delays the system for the given time.
    b. **SPIinterface :** It takes a byte as an input and sends each bit of the input to the pmodOLED in series.

After this, Init component sends a signal to the Example component to provoke. During this time period, Seven segments shows the name of the game "Shot"

2. **Example component** : This is the main part of this project. In this part, we created 14 screens, which comes one after one. First of all, we set the brightness and contrast

mode. Then we set page addressing mode and set its range to the maximum range. For the dot matrix of alphabets, we import character library (charlib.coe and a component charlib.vhd) Then we show the first screen – Alphabet which consists the first screen :

Welcome to the game

SHOT

After the delay of 4000 milliseconds, we enters to the second screen:

Select a mode :

Low : Switch 1

Medium : Switch 2

High : Switch 3

It demands an input from switches 1, 2 or 3. As we give an input, it moves to the game.

Gaming screens :

So our game is that we have to count the number of digits which appears on the OLED screen and give the input corresponding to the number of digits. If we give the right input our score increases by 1. till last screen, Seven segments were showing “Shot”. But by this screen, it starts showing the current score.

There are total 12 gaming screens. Which consists different patterns of different digits. These screens changes with the time delay of 1 second for high mode, 2 second for medium mode and 4 seconds for low mode. It we kept pressing the correct push button until the next screen appears, we get one point. Time delay and inputs are given by these two subcomponents :

a. **Delay :** It takes input in millisecond and delays the system for the given time.
b. **SPIinterface** : It takes a byte as an input and sends each bit of the input to the pmodOLED in series.

Shooting Game

12 gaming screens are as follows :

Screen 1 :

Screen 2 :

Screen 3 :

Screen 4 :

Screen 5 :

Shooting Game

Screen 6 :

Screen 7 :

Screen 8 :

Screen 9 :

Screen 10 :

Shooting Game

After these gaming screens, Scorecard is shown, for scores, we created 12 screens for different scores. If we score 10 then the scorecard screen will be :

Your Score is :

10

This screen remains up to 4 seconds and then the last screen shows:

…THANK YOU…

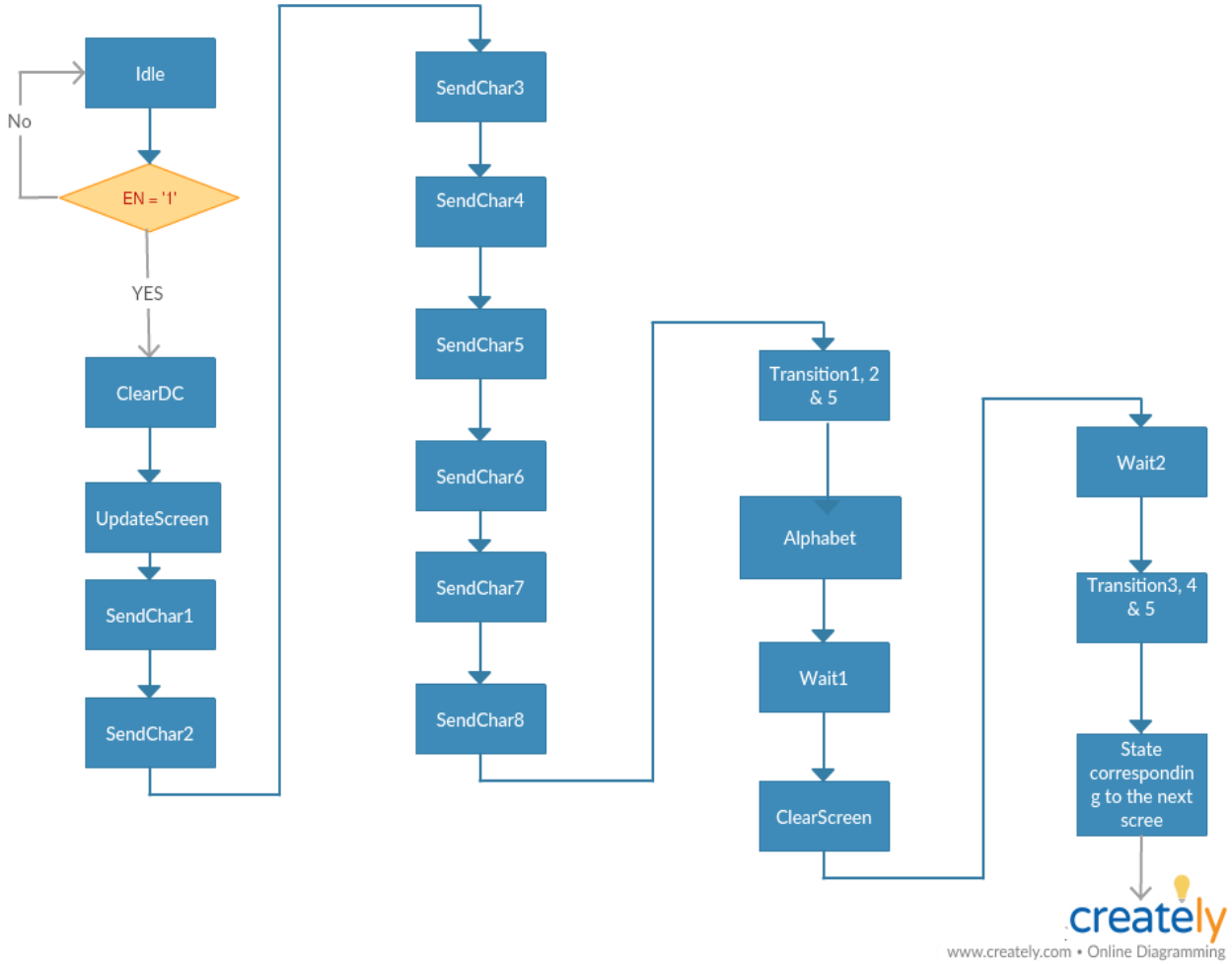
This screens remains until we press the reset button.

After pressing reset button, the game restarts with 0 score and goes to the first Alphabet screen.

Printing Characters : We use ASCII values of the corresponding characters to print those on OLED display. For this, we import *charlib.coe and charlib.vhd* files which gives the values of a block matrix of 4 x 16, in which the character is supposed to print.

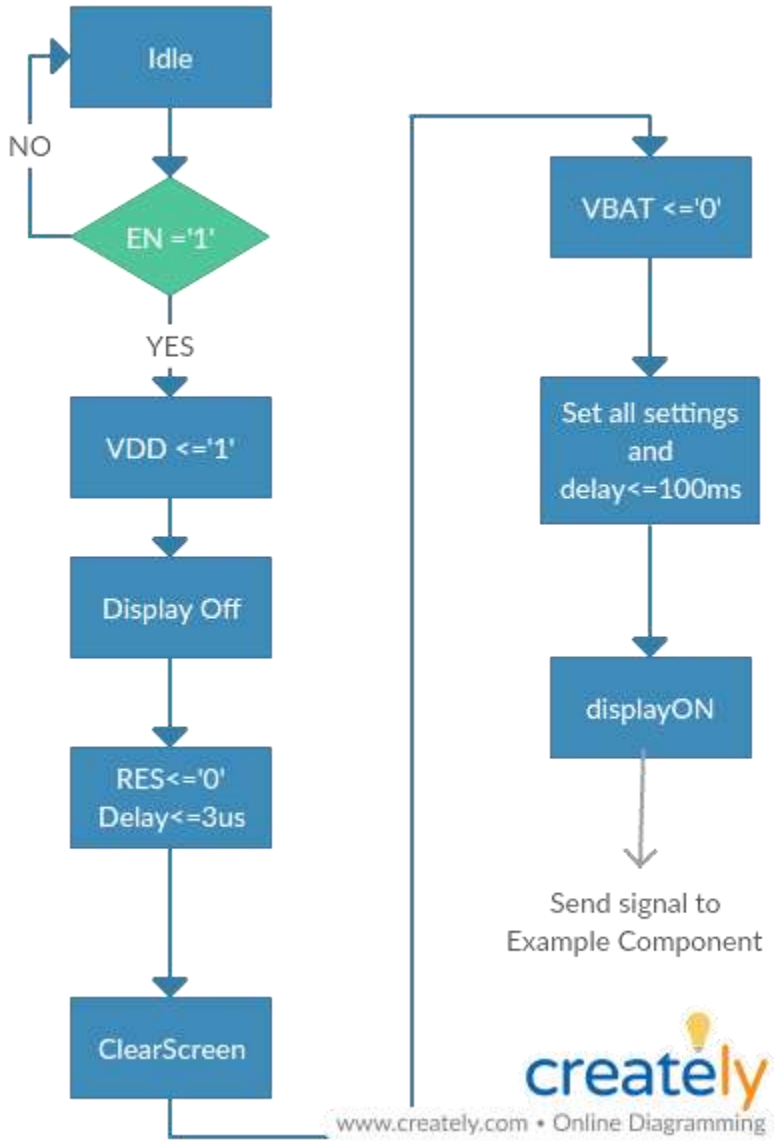
Shooting Game

ASM chart for printing screens:



Shooting Game

ASM chart of Initilization :



PROJECT SUBMITTED BY:

MANOJ KUMAR 2018CS50411

NIKITA BHAMU 2018CS50413

charlib.coe and charlib.vhd files : internet

ASM charts created with www.creately.com