EE447 INTRODUCTION TO MICROPROCESSORS LABORATORY PROJECT

FINAL REPORT

Muttalip Caner TOL
2031466
Çağnur TEKEREKOĞLU
2031425

1) Introduction

Battleship Game Console Project consists of three main modules: Initialization, Cursor Module and Time Module. The objective is to mimic famous Battleship game with TM4C123GH6PM microcontroller, Nokia 5110 LCD screen and two potentiometers.

2) Framework

The ASM chart of the project is shown in the Figure 1.

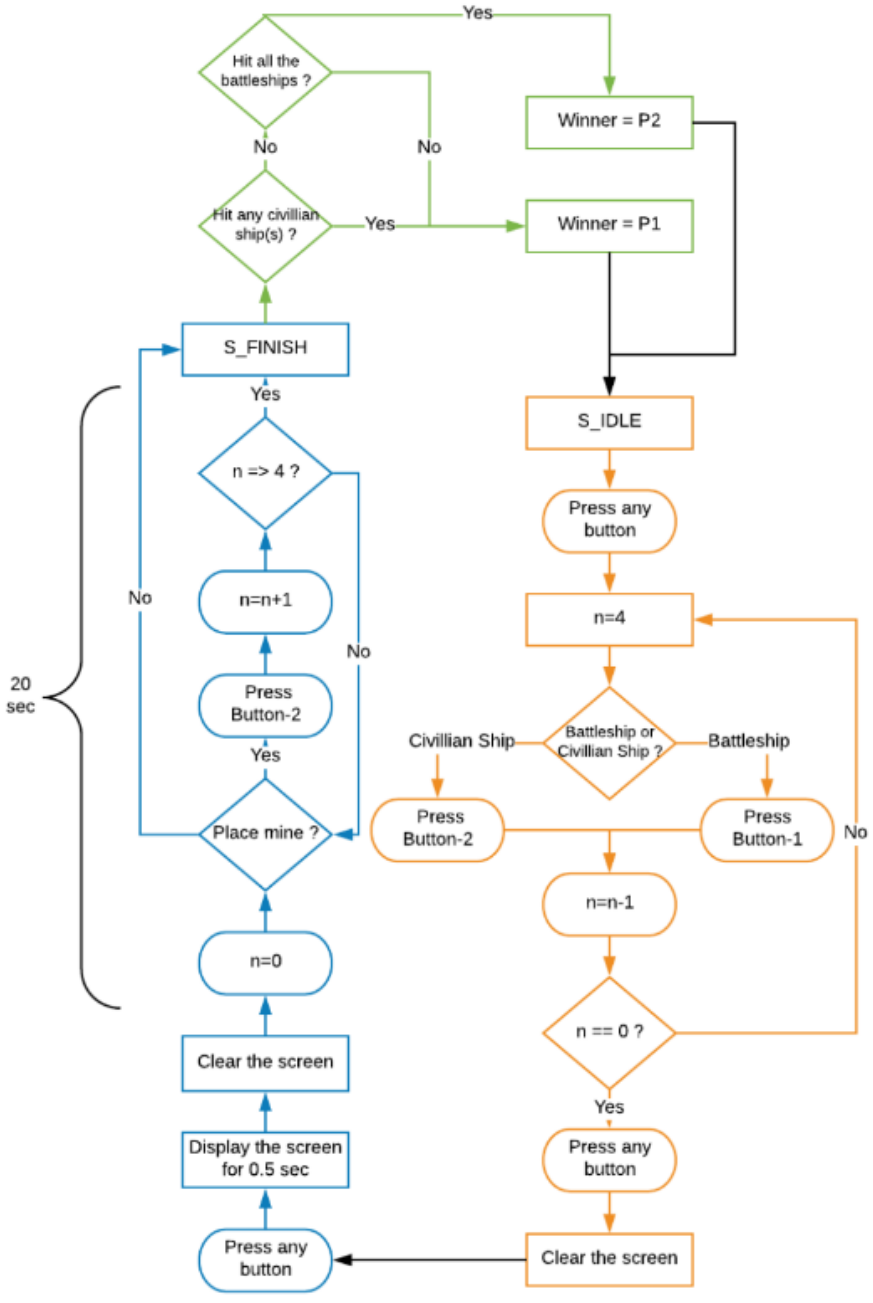

Figure 1: The ASM Chart of the Project

3) Submodules

Initialization Module

I. Analog to Digital Converter

ADC unit is used for getting X and Y coordinates of the cursor on the screen. In order to make ADC unit functional, we use PE3 as input for ADC0 and PE2 for ADC1.

II. Systick Timer

In order to create interrupt in every second for the “Time Module”, the Systick timer is used. In order to start the module, we put flag. At the end of it, it ends

III. Serial Peripheral Interface

We need one-way communication between microcontroller and the LCD screen. In order to establish this connection, we use SPI as a synchronous serial interface. In the SPI configuration, the microcontroller will be master, and the LCD screen will be the slave. Therefore, the LCD screen uses the clock generated by the microcontroller.

IV. LCD Configuration

Nokia 5110 LCD screen must be configured before displaying anything. There are a set of commands that should be given in an order. Only after sending these commands, we can send data bits.

V. Locked GPIO Pin

In order to unlocked PF0, GPIO_LOCK and GPIO_CR of the Port F is configured (TI).

VI. Clear Memory

Battleships, civilian ships and mines are stored in the memory. For this purposes, 256 bytes for each is cleared(=0x00).

Cursor Module

The cursor can be placed on the screen in 8 different cases. The Figure 2 represents these cases:

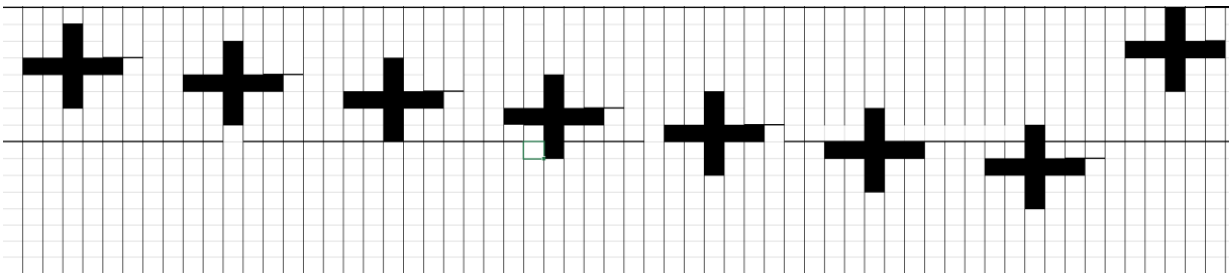

Figure 2 : Cursor Cases

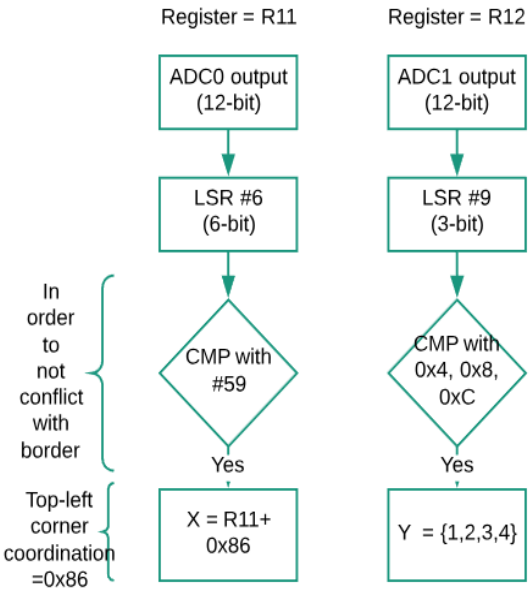

Figure 3 : Calculation Process of the coordination of the Cursor

X coordination calculation and Y coordination calculation is given in Figure 3. After the calculations, extra calculation is required to handle these cases in Cursor module. The 12-bit result from ADC1 is converted to 5-bit value which can represent max 32. This is divided by 8. The remainder gives one of these cases. For example, if the remainder is 1, first cursor model in Figure 2 is displayed.

Pointing pixel requirement is satisfied with this module.

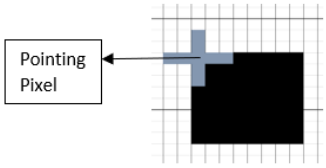

Figure 4 : Pointing Pixel

REMINDER: The battleship, civilian ship and mine selection cannot affect the displaying cursor. Because their places are stored in the memory and fetched from memory when we write on the screen. On the other hand, cursor is displayed simultaneously.

Selection of Ships and Mines Modules

This module is related to memory. 256-byte memory is cleared for ships and mines.256 is found with Equation 1 and 2:

$$\#\ bits = 32x64 \qquad (1)$$

$$\#\ byte = \frac{\#\ bits}{8} \qquad (2)$$

The background values while the selection parts fetched from the memory. This method provides simultaneous cursor movement. The playfield surrounded by a frame is stored in memory, byte by byte. In each sampling loop, the objects in the playfield are loaded from memory*. This method allows us to keep the objects in their place while the cursor is passing over them.* Another advantage of storing and reloading the playfield using memory, is deleting the outdated cursor from its old location.

In this module, the X and Y addresses of the cursor are taken from the registers which are loaded by the cursor module previously. According to the Y address of the pointing pixel of the cursor, there may be eight different conditions of the ships which can be seen in Figure 5 and 6. The method is very similar to what we did in cursor module. In this time, we store the first part of the ships on the bank which is pointed by the cursor. The lower part of the ship is stored on the following bank. Since we are storing the bytes to the memory in this case, the offset between two parts is equal to 56, i.e. if the

last byte of in the first bank is finishes on memory location X, the second part should start on memory location X+56.

The mine number can be up to 4 within 20 sec. It is controlled by flag in State4. After 4 mines are loaded, the player has to wait 20 sec.

The requirement is the fact that the pointing pixel of the cursor should point to the top-left corner of the bounding box of the ship icon is satistied with this and cursor modules.

Another reqiurement which is up-to-4-mine is controlled in this module.

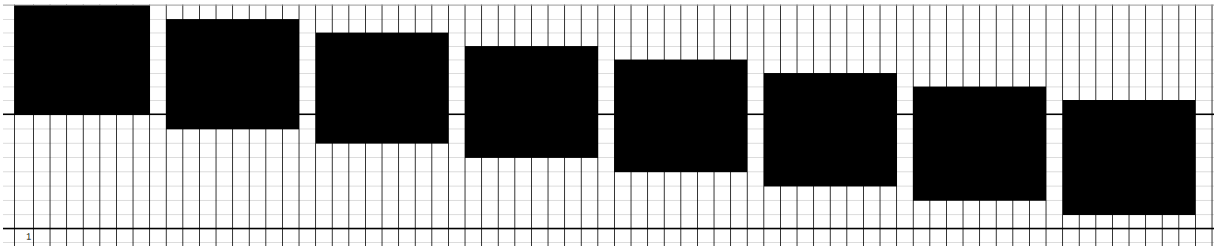

Figure 5 : Eight different locations of the battleship icon

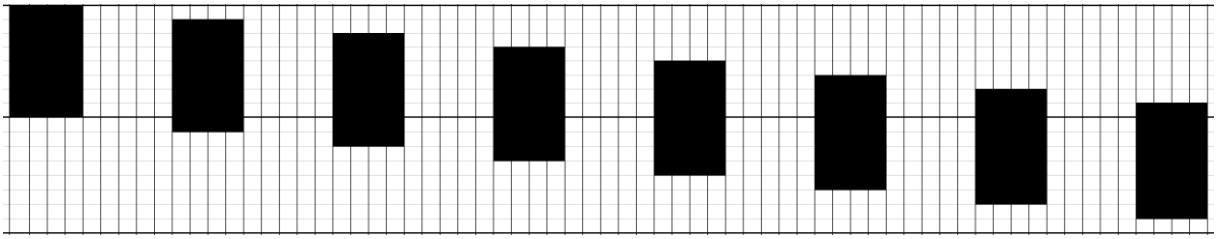

Figure 6 : Eight different locations of the civilian ship icon

Button Interrupt Module

As shown in the Figure 1, the state exchange is occurred by buttons. A flag is created to track the state flows. According to this flag, interrupt subroutine decides what it does. The general description of the states and their diagram are shown in Figure 4.

For example, when the button is pressed, ships are loaded to memory according to pressed button. Because the main__ uses the ship memory for background values, the loaded ships are shown on the screen with the cursor movement. After this process, the state flag is increased. Another interrupt call will clean the screen.

Time Module

0.5 sec is produced by 5x(Delay100) subroutine in State3. 20 sec is produced by SysTick. It is initialized in State4. After 20 sec later (After printing 20), SysTick Handler is disabled.

The requirement for the time is achieved with these methods.

Decision Module

Because we store the ships and mines location in the memory, we compare each byte in this module. It checks whether there is any conflict with civilian ship and then whether the mine hits all the battleships. Generally, we use XOR in this module.

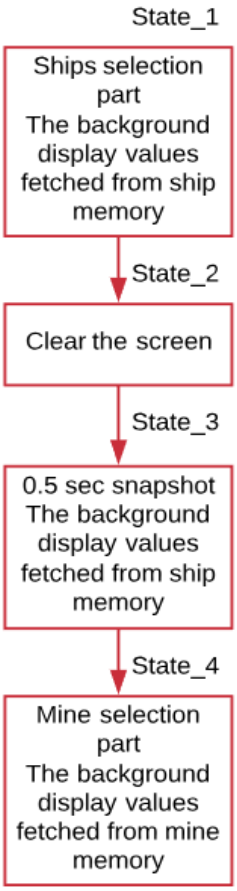

Figure 5 : Flow of the Interrupt Subroutine

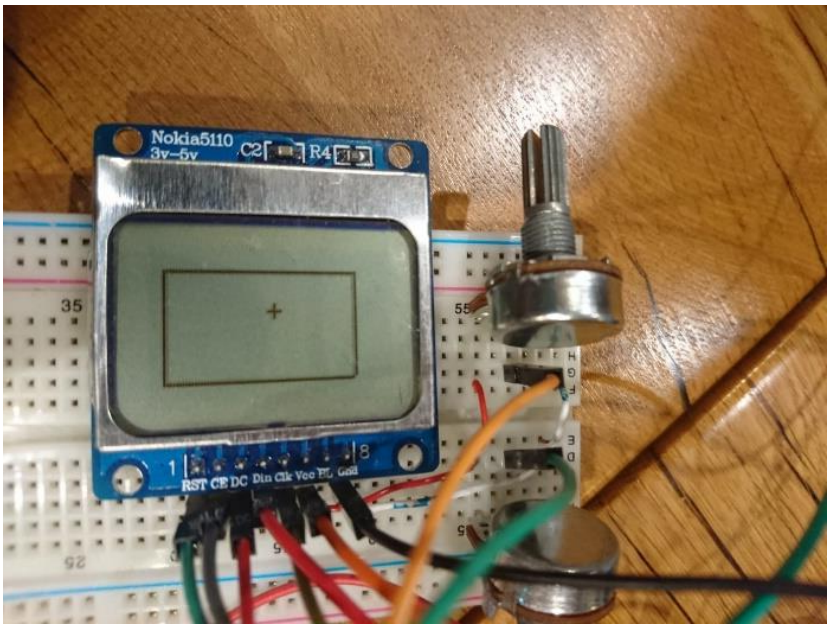

Figure 6 : Set-up circuit

4) Conclusion

Thanks to this project, we have opportunity to apply our knowledge in laboratories and lectures. Since the project requires many modules and multitasking software and combining all of them together at the end, we have attained a high level of knowledge on embedded systems.

References

TI. *Texas Instruments.* http://www.ti.com/lit/ds/spms376e/ received from