

I. Introduction:

This lab experiment focuses on using a Graphical LCD (GLCD) to display custom content, including hand-drawn images and a two-digit timer. The GLCD, with its 128 x 64-pixel resolution and parallel interface, serves as the display platform. The process begins by creating a monochrome BMP file using the Paint application in Windows, which is then converted into a pixel array using the GLCD bitmap editor by Mikroelektronika. The next phase involves exploring animation by sequentially displaying a series of images. Finally, the GLCD is used to show a three-digit descending counter. Through this lab, participants gain practical experience in interfacing with and programming GLCDs, enhancing their relevant skills.

II. Procedure:

1. Part a: Programmable timer with 3 push buttons:

For this task, we programmed the graphics LCD (GLCD) to display a fixed image. We used the Paint application in Windows to create an image with dimensions matching the GLCD's resolution of 128 x 64 pixels. After creating the image, we saved it in monochrome BMP format and imported it into the GLCD bitmap editor provided by Mikroelektronika.

❖ GLCD interfacing with PIC18F45K22.

LCDs are used to display monochromatic graphical elements such as text, images, and other content. The EasyPIC 7 demo board includes the necessary connections and interfaces for operating GLCDs with a resolution of 128x64 pixels.

The GLCD pins and their functions are as follows:

- **CS1, CS2:** Controller Chip Select lines.
- **VCC:** +5V display power supply.
- **GND:** Reference ground.
- **Vo:** GLCD contrast adjustment.
- **RS:** Data (High) / Instruction (Low) selection line.
- **R/W:** Determines whether the display is in Read or Write mode.
- **E:** Display Enable line.
- **D7... D0:** Data lines.
- **RST:** Display reset line.
- **Vee:** Reference voltage for the GLCD contrast potentiometer P3.
- **LED+:** Connection to the back-light LED anode.
- **LED-:** Connection to the back-light LED cathode.

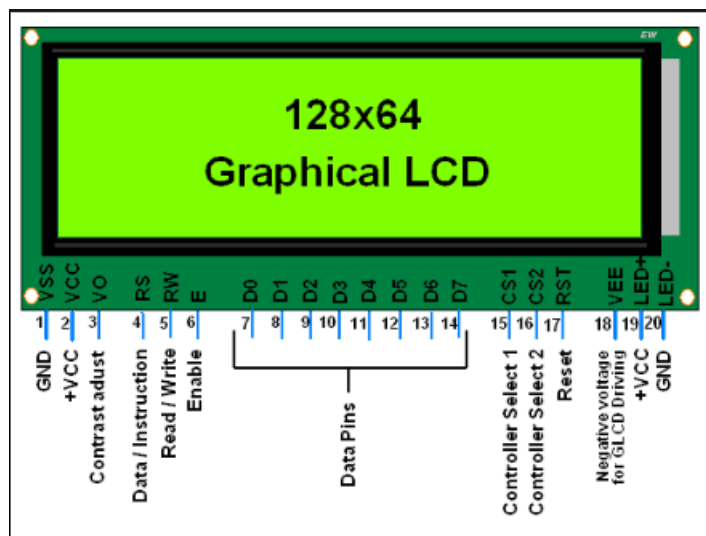


Figure 1:GLCD 128x64

The GLCD.h header file provides a library of functions and definitions to facilitate the interfacing and control of GLCD (Graphical Liquid Crystal Display) modules with a

microcontroller. It includes function prototypes for initializing the GLCD, sending commands and data to the display, and controlling various display features:

- **InitGLCD()**: Initializes the LCD screen and sets up all control signals.
- **DispRomStr(LnxChy, (Rom) "-----")**: Displays characters stored in the Read-Only Memory (ROM) of the microcontroller. ROM is non-volatile memory used for storing permanent program instructions and data that do not change during program execution.
- **DispVarStr(char digits[], LnxChy, size of the array in bytes)**: Displays a variable string, overwriting the characters at the specified position.
- **Bin2Asc(unsigned char binaryValue, char* asciiArray)**: Converts a binary value into its corresponding ASCII characters and stores the result in an array.

Pick the best in the west

Figure 2: BMP picture used

A. MPLAB Code:

```
#include <GLCD.h>
#include <p18cxxx.h>
#include <delays.h>

const far rom char picture1 [1024] = {
    0, 0, 0, 0, 0, 0, 0, 0, 192, 64, 192, 64, 64, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 192, 128, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192,
    128, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 128, 0, 0, 0, 0, 0, 0, 0, 0,
```

$$\};$$

5 Kris Antony Atallah

```
}
```

Explanation of the code:

1. Libraries Used:

- "p18cxxx.h"
- "delays.h"
- "GLCD.h": Used to control the GLCD.

2. Global Variables Used:

The global variable picture1 is declared as ***const far rom char picture1[1024];*** Even though the GLCD has a resolution of 128 x 64 pixels, totaling 8192 pixels, we only needed to construct an array of 1024 elements. Each element is of type char, with each char occupying 1 byte (8 bits). Consequently, each element of the array represents 8 pixels on the GLCD. The complete type of the array elements is const far rom char because they are constants intended for storage in ROM memory. The inclusion of far indicates that a pointer should be capable of addressing a larger memory space compared to a standard pointer.

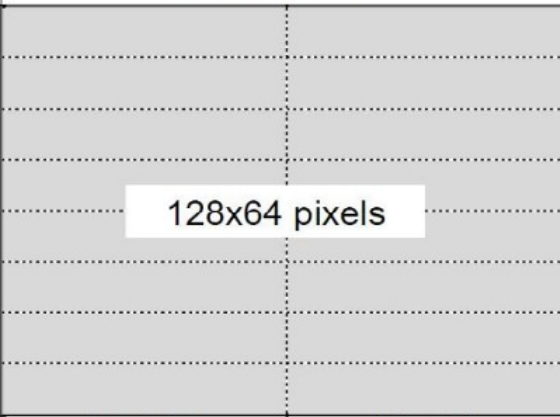
Display Memory Map		
Page (X) address	data	LCD Display (front view)
0	D0 : D7	
1	D0 : D7	
2	D0 : D7	
3	D0 : D7	
4	D0 : D7	
5	D0 : D7	
6	D0 : D7	
7	D0 : D7	
Column(Y) Address		00h → 3Fh
Chip Select		CS1=1, CS2=0
		00h → 3Fh
		CS1=0, CS2=1

Figure 2:GLCD Memory Map

$$\text{GLCD Pixels} = 128 * 64 = 8192$$

$$8 \text{ pixels/byte: } \frac{8192}{8} = 1024$$

B. Demo Board Implementation:

Our project utilizes the EasyPIC v7 Demo Board, which requires the mikroProg Suite™ for PIC® to program it. We connected a GLCD to the board. To power on the GLCD, first turn on the VDD (+5V) by setting SW4 pin 6 to the ON position. Next, adjust the backlight of the LCD using the potentiometer for contrast. To connect the buzzer as mentioned in the code, attach the yellow jumper to RE1 for the piezo buzzer.



2. Part b: Multiple frames using GLCD

A. MPLAB Code:

Kris Antony Atallah

};

$\};$

```
const far rom char picture4 [1024] = {  
    0, 0, 0, 0, 0, 0, 0, 0, 192, 64, 192, 64, 64, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 192, 128, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192,  
    128, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 127, 2, 3, 1, 0, 64, 114, 0, 0, 0, 120, 72, 64, 0, 0, 126, 56,  
    44, 100, 0, 0, 0, 0, 4, 6, 255, 2, 127, 10, 8, 24, 112, 0, 56, 104, 88, 0, 0, 0, 0, 127,  
    216, 136, 216, 112, 0, 112, 208, 176, 0, 48, 56, 104, 72, 193, 1, 255, 1, 1, 0, 0, 244, 0, 224,  
    48, 48, 224, 0, 0, 4, 6, 255, 2, 127, 10, 8, 24, 112, 0, 56, 104, 88, 0, 0, 0, 2, 14, 120,  
    192, 0, 192, 112, 192, 0, 128, 254, 2, 0, 252, 164, 44, 24, 0, 128, 56, 40, 42, 226, 2, 127,  
    194, 2, 2, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 32, 192, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,  
    1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 15, 221, 105, 83, 98, 66, 194, 134, 196, 124, 248,  
    140, 198, 194, 163, 145, 217, 105, 61, 31, 15, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 128, 192, 64, 64, 64, 64, 64, 64, 192, 128, 0, 0, 0, 0, 0, 0, 0, 0, 128, 128, 192, 64,  
    64, 64, 192, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 222, 242, 161, 161, 163, 62, 0, 128, 135, 252, 139, 76, 218,  
    239, 129, 128, 0, 31, 49, 35, 46, 24, 8, 24, 48, 224, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    240, 28, 15, 1, 16, 16, 16, 16, 16, 16, 0, 0, 129, 199, 124, 48, 120, 140, 2, 3, 1, 16, 16,  
    16, 252, 16, 16, 16, 0, 0, 3, 206, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 31, 16, 240, 120, 63, 28, 22, 241, 0, 15, 24, 16, 16, 16, 25,  
    15, 3, 4, 24, 120, 204, 198, 194, 126, 6, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 6,  
    132, 132, 4, 4, 4, 4, 4, 4, 2, 3, 1, 0, 224, 128, 0, 1, 1, 2, 6, 4, 68, 4, 4, 4, 4, 4,
```

[illegible]

```
const far rom char picture6 [1024] = {  
    0, 0, 0, 0, 0, 0, 0, 0, 192, 64, 192, 64, 64, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 192, 128, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 192,  
    128, 0, 192, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 127, 2, 3, 1, 0, 64, 114, 0, 0, 0, 120, 72, 64, 0, 0, 126, 56,  
    44, 100, 0, 0, 0, 0, 4, 6, 255, 2, 127, 10, 8, 24, 112, 0, 56, 104, 88, 0, 0, 0, 0, 127,  
    216, 136, 216, 112, 0, 112, 208, 176, 0, 48, 56, 104, 72, 193, 1, 255, 1, 1, 0, 0, 244, 0, 224,  
    48, 48, 224, 0, 0, 4, 6, 255, 2, 127, 10, 8, 24, 112, 0, 56, 104, 88, 0, 0, 0, 2, 14, 120,  
    192, 0, 192, 112, 192, 0, 128, 254, 2, 0, 252, 164, 44, 24, 0, 128, 56, 40, 42, 226, 2, 127,  
    194, 2, 2, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 32, 192, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 96, 192, 128, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 128, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,  
    1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 15, 221, 105, 83, 98, 66, 194, 134, 196, 124, 248,  
    140, 198, 194, 163, 145, 217, 105, 61, 31, 15, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2,  
    4, 24, 160, 192, 192, 64, 64, 64, 64, 64, 192, 128, 0, 0, 0, 0, 0, 0, 0, 0, 128, 128, 192, 64,  
    64, 64, 192, 128, 128, 224, 32, 24, 4, 4, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    0,  
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 222, 242, 161, 161, 163, 62, 0, 128, 135, 252, 139, 76, 218,  
    239, 129, 128, 0, 31, 49, 35, 46, 24, 8, 24, 48, 224, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
    240, 28, 15, 1, 16, 16, 17, 18, 18, 28, 16, 32, 193, 199, 124, 48, 120, 140, 2, 3, 1, 16, 16,
```

[illegible]

[illegible]

```
void main(void){
    InitGLCD();
    while(1){
        DisplayPicture((ROM) picture1);
        Delay10KTCYx(3000);
        DisplayPicture((ROM) picture2);
        Delay10KTCYx(500);
        DisplayPicture((ROM) picture3);
        Delay10KTCYx(500);
        DisplayPicture((ROM) picture4);
        Delay10KTCYx(500);
        DisplayPicture((ROM) picture5);
        Delay10KTCYx(500);
        DisplayPicture((ROM) picture6);
        Delay10KTCYx(500);
        DisplayPicture((ROM) picture7);
        Delay10KTCYx(500);
        DisplayPicture((ROM) picture8);
        Delay10KTCYx(2000);
    }
}
```

In this part we used the AMPIRE128X6 as our GLCD.



For this part the same configuration used for the previous part is used here also.



A. MPLAB Code:

19

```
while(1) {  
    // Display the static picture stored in ROM  
    DisplayPicture((ROM) picture7);  
}
```

```

// Display the static text "Left time:  s" on the first line of the GLCD
DispRomStr(Ln1Ch0, (ROM) "Left time:  s");

// Start the countdown loop
while(1) {
    // Convert the seconds variable to a character array (digits)
    Bin2Asc(seconds, digits);
    // Display the countdown digits on the GLCD, starting from column 10 of the first line, with
    a length of 3
    DispVarStr(digits, Ln1Ch10, 3);
    // Decrement the seconds variable
    seconds--;
    // Delay for 500 milliseconds (50 * 10K cycles) before updating the display
    Delay10KTCYx(50);

    // Check if the countdown has reached zero
    if(seconds == 0)
        break; // Exit the countdown loop
}

// Display the blank picture stored in ROM
DisplayPicture((ROM) picture1);
// Delay for 500 milliseconds before resetting the seconds variable
Delay10KTCYx(50);
// Reset the seconds variable to 20 for the next countdown
seconds = 20;
}

```

4. Code Explanation:

For this code we used two arrays, picture1 holding the blank picture array, and picture7 holding the picture that we want to display array. The rest of the code has been explained through the comments.

B. Proteus simulation results:

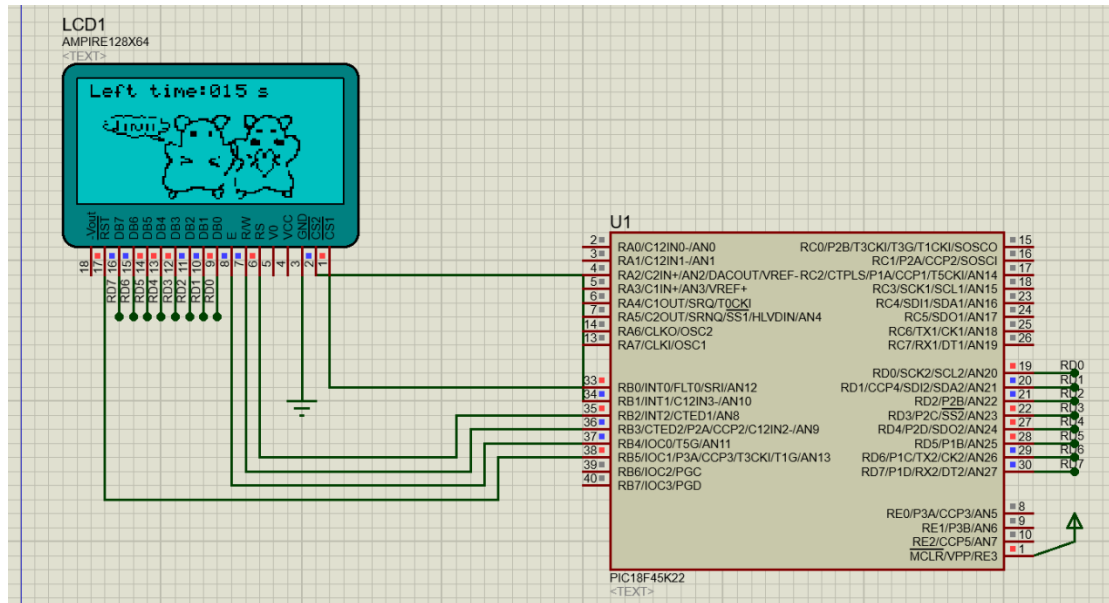


Figure 2 Timer before reaching zero

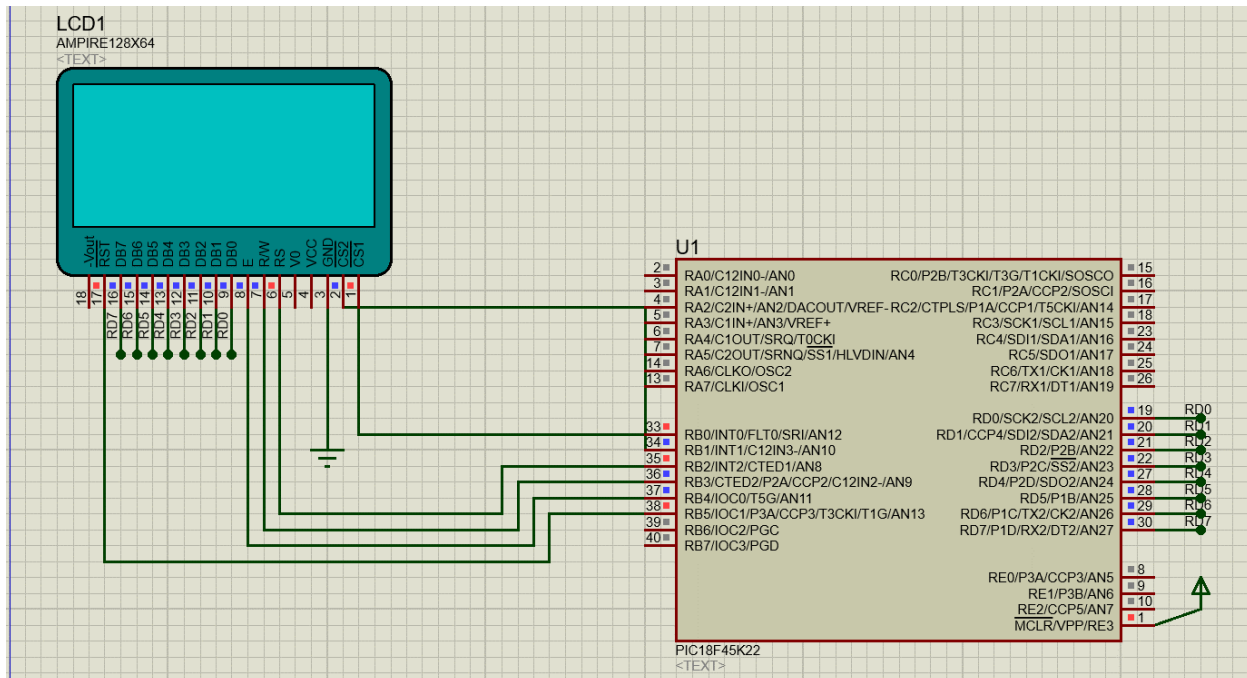


Figure 3 At t = 0

C. Demo Board Implementation:

The same configuration used for the hardware in the previous parts was used here also.

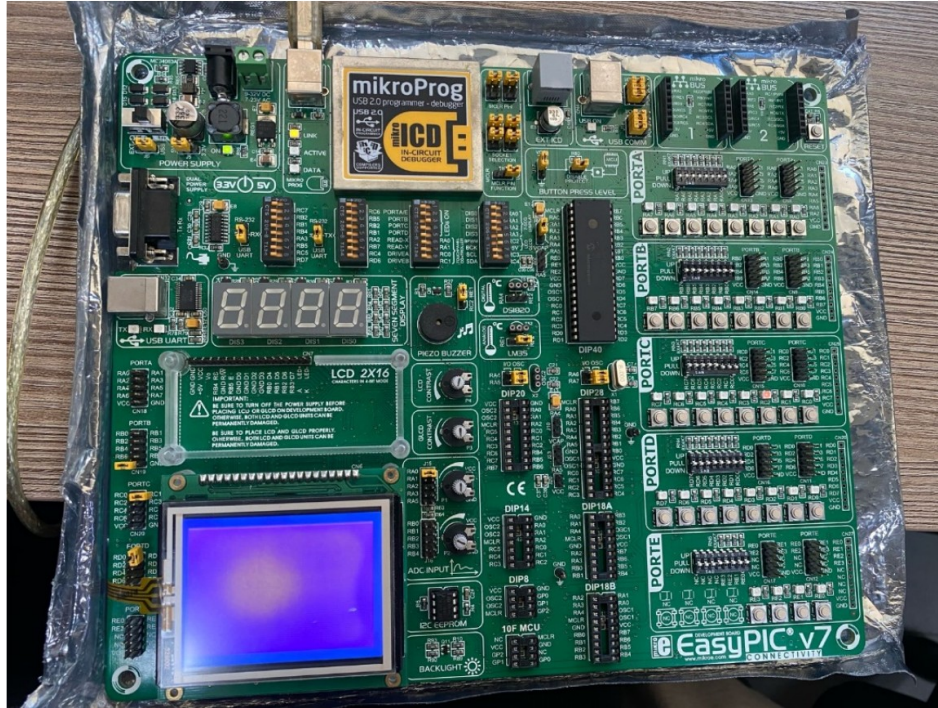


Figure 5 At $t = 0$

And the process will repeat itself.

III. Conclusion:

In summary, this laboratory session offered practical experience in interfacing with GLCD on the EasyPIC 7 demo board. We explored its functionalities by exhibiting a unique hand-drawn image, demonstrating the customized visual potential it provides. The experiment progressed to animation, where we displayed several images sequentially, illustrating the GLCD's dynamic features. Additionally, incorporating a 3-digit timer on the GLCD broadened our comprehension of both static and dynamic visual representations.