KOCAELİ UNIVERSITY ENGINEERING FACULTY

ARDUINO LCD KEYPAD SHIELD
REFERENCE DOCUMENT

Furkan YARDIMCI
210207089

DEPARTMENT: Electronic and Communacition Engineering
TEACHER: Associate Profesor Dr. ANIL ÇELEBİ

KOCAELİ 2022

TABLE OF CONTENTS

LIST OF IMAGES

LIST OF TABLES

Arduino LCD keypad shield reference document

PURPOSE

The purpose of this document is to provide a document containing the necessary features about the design and use of LCD Keypad shield, prepared by Kocaeli University Students FURKAN YARDIMCI.

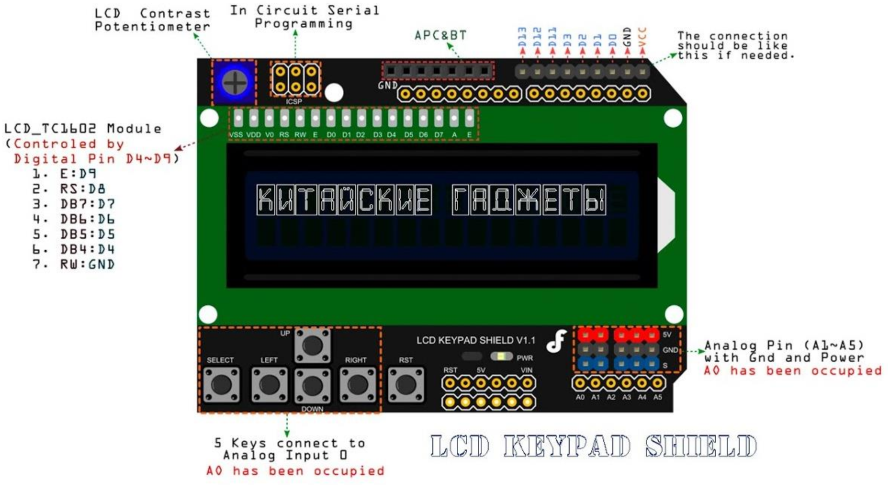
1. INTRO/OVERWIEV

LCD Keypad shield for Arduino or Freeduino board. It includes a 2x16 LCD display and 6 momentary push buttons. Pins 4, 5, 6, 7, 8, 9 and 10 are used to interface with the LCD. Analog Pin 0 is used to read the push buttons. The LCD shield supports contrast adjustment and backlit on/off functions. It also expands analog pins for easy analog sensor reading and display.
The LCD Keypad shield is developed for Arduino compatible boards, to provide a user-friendly interface that allows users to go through the menu, make selections etc. It consists of a 1602 white character blue backlight LCD. The keypad consists of 5 keys — select, up, right, down and left. To save the digital IO pins, the keypad interface uses only one ADC channel. The key value is read through a 5 stage voltage divider.

1.1 FEATURES:

- Operating Voltage:5V
- 5 Push buttons to supply a custom menu control panel
- RST button for resetting Arduino program (in our case PYNQ-Z2)
- Integrate a potentiometer for adjusting the backlight
- Expanded available I/O pins
- Expanded Analog Pinout with standard DFRobot configuration for fast sensor extension

The buttons for the right, left, front and back on the module are connected to the A0 pin in common. You can understand which button was pressed by processing the data coming to the A0 pin. In this way, you will also save on pins. In addition, LCD works in 4-bit mode. Detailed information about 4-bit mode is given in the next section.


LCD Contrast Potentiometer
In Circuit Serial Programming
APC&BT
D13 D12 D11 D3 D2 D1 D0 GND VCC
The connection should be like this if needed.
ICSP
GND
LCD_TC1602 Module (Controled by Digital Pin D4~D9)
1. E:D9
2. RS:D8
3. DB7:D7
4. DB6:D6
5. DB5:D5
6. DB4:D4
7. RW:GND
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A E
КИТАЙСКИЕ ГАДЖЕТЫ
UP
SELECT
LEFT
RIGHT
DOWN
RST
LCD KEYPAD SHIELD V1.1
PWR
RST 5V VIN
5V
GND
S
A0 A1 A2 A3 A4 A5
Analog Pin (A1~A5) with Gnd and Power
A0 has been occupied
5 Keys connect to Analog Input 0
A0 has been occupied
LCD KEYPAD SHIELD

Img 1: LCD keypad shield

2. LCD DISPLAY

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

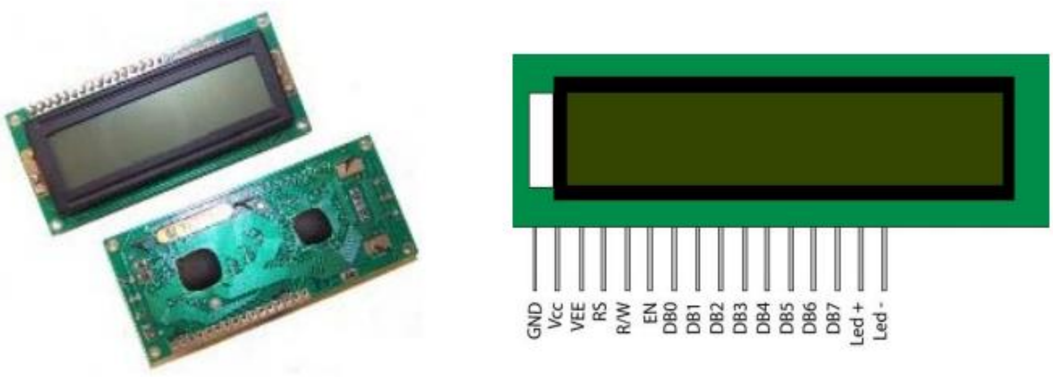


Image 2: 2x16 LCD Display

2.1 PIN CONFIGURATION

SM PMOD Reference Designator	Function
Pin 1 (Vss)	Function as Ground Terminal.
Pin 2 (Vcc)	Function as Positive Supply (2.7V to 5.5V).
Pin 3 (Vdd)	Function as Contrast adjustment (Ground to Vcc).
Pin 4 (RS)	Function as Register Select (If 0 is refer to Instruction Register and if 1 is refer to Data Register).
Pin 5 (R/W)	Its function to Read or Write Signal (if 1 mean to Read and if 0 mean to Write).
Pin 6 (E)	Function as Enable.
Pin 7 to Pin 14 (DB0 – DB7)	: Refer to Bi-directional data bus, data transfer is performed one, thru DB0 to DB7, in this case of interface data length is 8- bits; and twice, through DB4 to DB7 in this case of interface data length is 4-bits (Upper nibble first and then Lower nibble).
Pin 15 (K)	Function to Back light LED cathode terminal.
Pin 16 (A):	Function to Back light LED anode terminal.

Table 1: pins of LCD display

2.2 CONTROL AND DISPLAY COMMANDS

There is a character LCD controller of Hitachi company named HD44780 on most character LCDs available in the market. This controller acts as a bridge between the LCD and the FPGA. In other words, we do not directly interfere with the pixels on the LCD with the FPGA. We ensure that the characters we want are displayed through the controller. Since the HD44780 is a general-purpose controller, most character LCD manufacturers use this controller in their LCDs of various sizes and features.

3 bits are very important for sent data or instruction

RS: Register Select pin. In the logic 0 state, a command is sent to the HD44780 from the bus. If logic 1 is set, data is written or read from the data bus to the HDD780. It is necessary to write or receive commands to make adjustments to the HD44780, and to send or receive character data.

R/W: Read write pin. In the logic 1 state, reading is taken from HD44780. If the logic is 0, the HD44780 is written to. Since the HD44780 is generally written to, this pin is usually connected directly to gnd in applications.

EN: It is the enable pin. In case of logic 1, read-write operation is performed to HD44780. Cannot be done in the logic 0 state. D0:D7: The bus of HD44780 controller is 8 bits wide, used for reading, write operations.

To send instructions to LCD via controller, RS and R/W bits must be 0 and to send data RS must be 1. In this mode, we can send commands to LCD's instruction register. The commands are 8 bit wide and given in the list below.

Sr.No.	Hex Code	Command to LCD instruction Register
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift the cursor position to the right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to the beginning (1st line)
17	C0	Force cursor to the beginning (2nd line)
18	38	2 lines and 5×7 matrix (8bit mode)
19	28	2 lines and 5×7 matrix (4bit mode)

Table 2: LCD instructions

2.3 INITIALIZATION OF LCD

According to datasheet some process must be followed beginning of the device start to function properly. This process steps are given below. To send instructions RS bit should be 0 and to write R/W bit should be 0 as mentioned above and enable pin must be high at least 230ns to send or write any data to lcd. Waits are given due to this information.

1	EN = 0	Wait 15ms or longer before enable the device.
2	EN = 1	Send 0x3 for 240ns.
3	EN = 0	Wait 4.1ms or longer.
4	EN = 1	Send 0x3 for 240ns.
5	EN = 0	Wait 100us or longer.
6	EN = 1	Send 0x3 for 240ns.
7	EN = 0	Wait 40us or longer.
8	EN = 1	Send 0x2 for 240ns.
9	EN = 0	Wait 40us or longer.

Table 3: LCD initialization

After initialization process, we can send data or instruction to LCD. First, we will send instructions. Used instructions are listed below.

1	0E	Display on, cursor blinking
2	06	Increment cursor (shift cursor to right)
3	80	Force cursor to the beginning (1st line)
4	28	2 lines and 5×7 matrix (4bit mode)

Table 4: LCD instructions used

2.4 USAGE OF 4-BIT MODE

8 bit data length is expensive in terms of pin usage so LCD used in 4 bit mode in this project. To do this 0x28 instruction must be send to LCD when RS and R/W bits are 0. In this mode, data is sent in nibbles. The MSB nibble must be sent first, followed by the LSB nibble. Regardless of whether the information sent is data or instructions, a certain process must be followed. After sending MSB nibble, wait 1 microsecond and then send LSB nibble and wait at least 40 microseconds at the end of each 8-bit data packet. This process must be followed to send instruction or data to LCD.

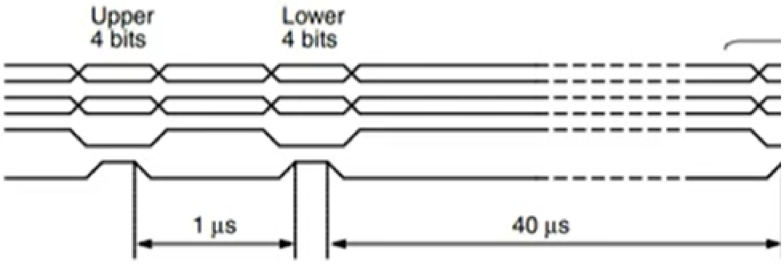
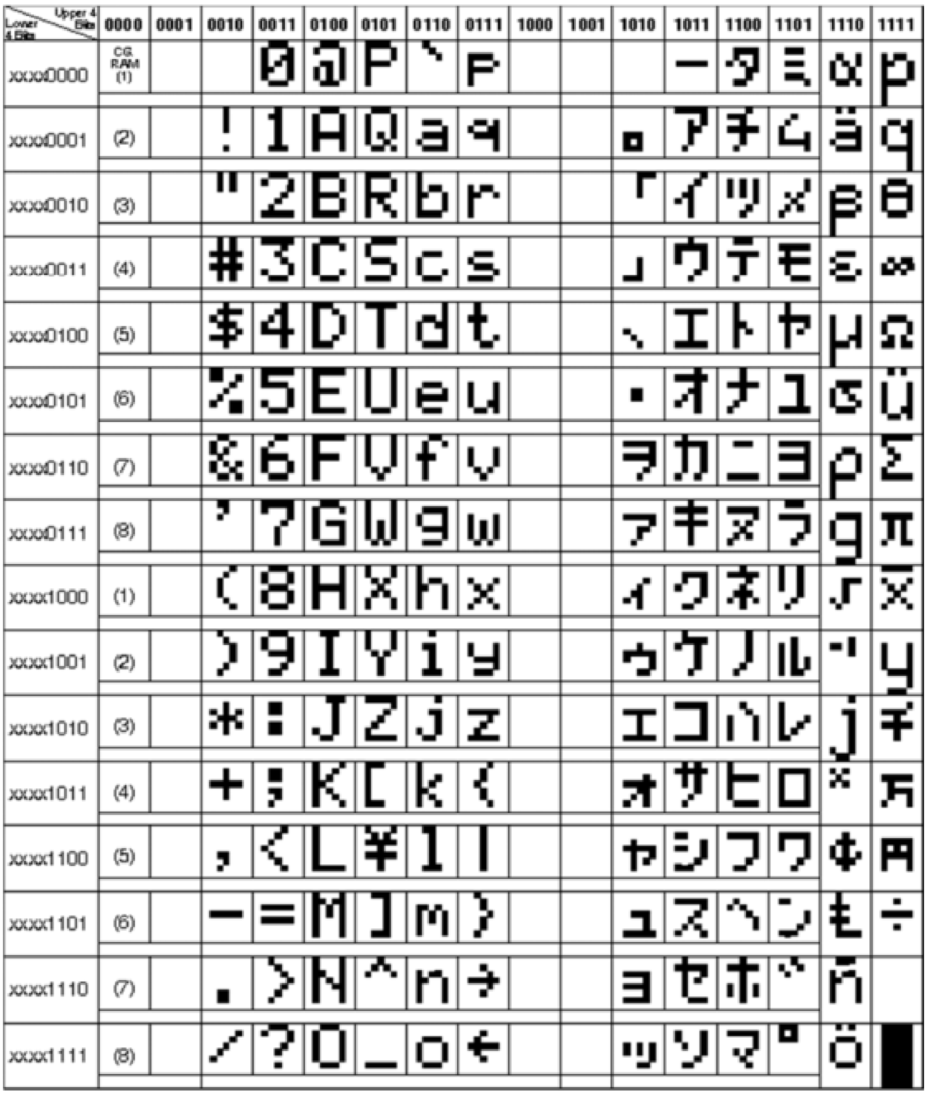


Image 3: 4-bit mode working process

2.5 ALPHA NUMERIC TABLE

The 2x16 LCD display has the power to write 2x16=32 characters in total. It uses a 5x7 matrix to write each character to the screen. The upper and lower nibble values of these characters are given below. To be able to read these characters on the screen, RS = 1 and R/W = 0 and when RS = 1, we send data to LCD's data register as mentioned early.



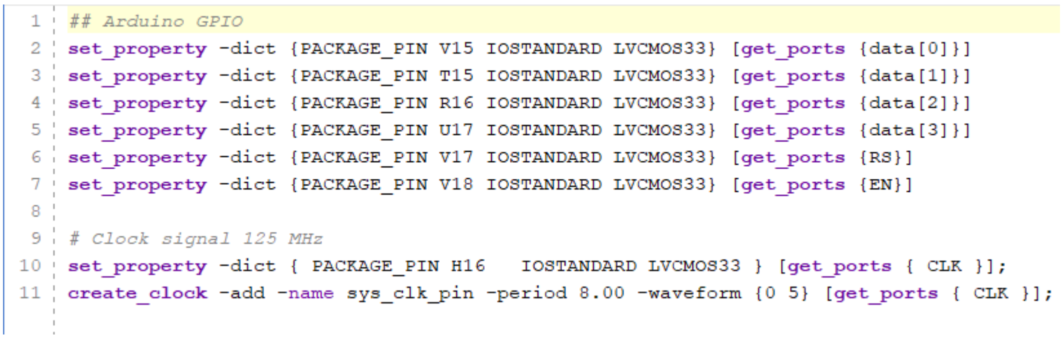
Upper 4 Bits Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
xxxx0010	(3)		"	2	B	R	b	r			「	イ	ツ	メ	β	θ
xxxx0011	(4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
xxxx0100	(5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
xxxx0110	(7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
xxxx1000	(1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
xxxx1001	(2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	⁻¹	y
xxxx1010	(3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
xxxx1011	(4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	ˣ	万
xxxx1100	(5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
xxxx1101	(6)		-	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
xxxx1110	(7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
xxxx1111	(8)		/	?	O	_	o	←			ッ	ソ	マ	゜	ö	█

Img 4: Alphanumeric table

3. VERILOG HDL:

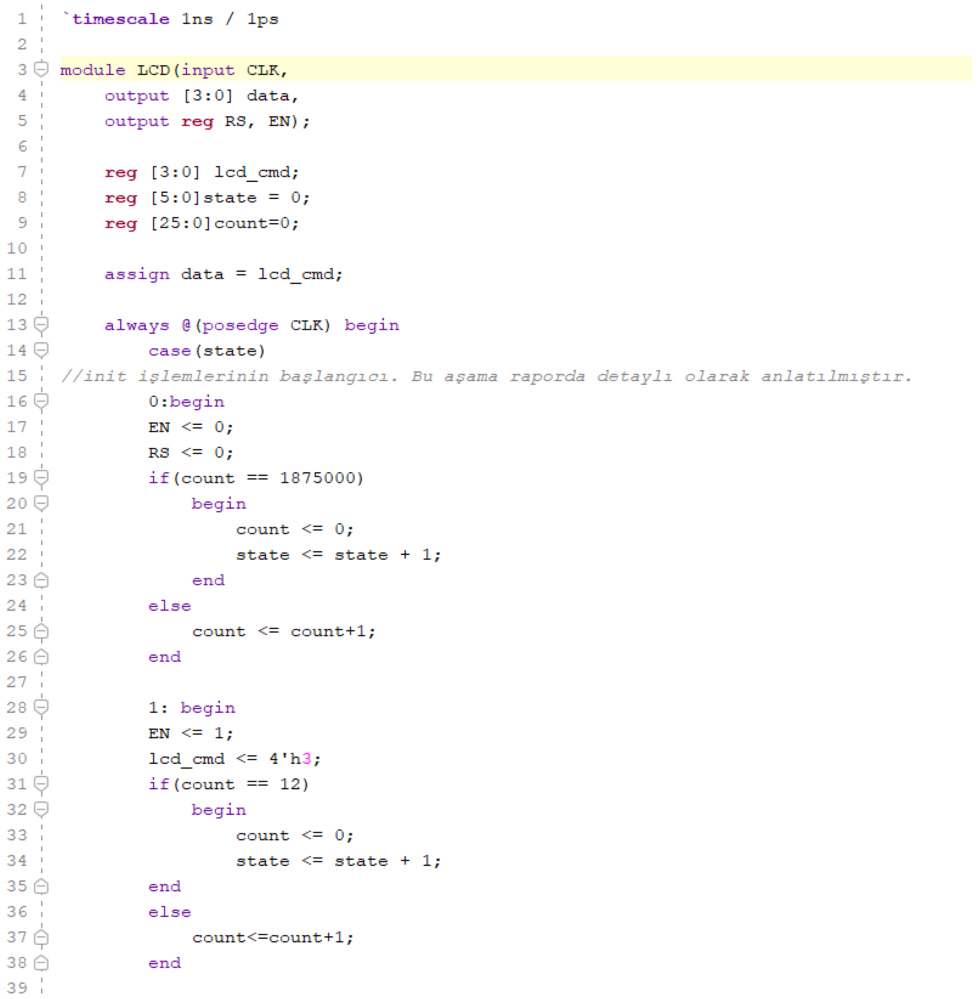
Procedural design was used for the initial and control processes of the LCD mentioned in the previous section. The always block is sensitive to the rising edge of the clock signal and is triggered on each rising edge. Appropriate waiting times are given to the datasheet of the LCD between each state, and these waiting times are given specifically for the PYNQ-Z2 card. For different cards, these times should be recalculated depending on the card's clock speed. The code is as follows.

3.1 XDC FILE

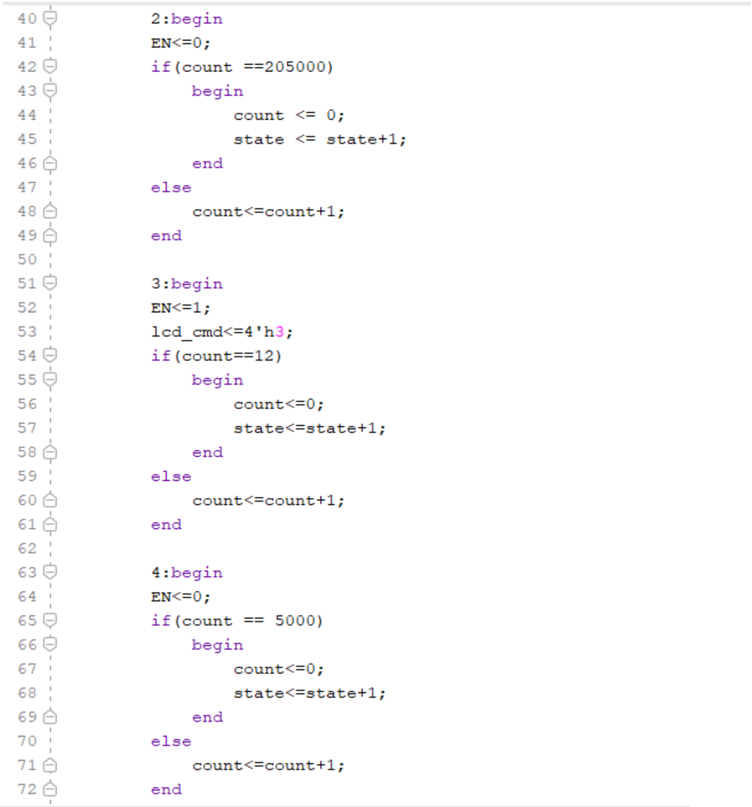


```
## Arduino GPIO
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {data[0]}]
set_property -dict {PACKAGE_PIN T15 IOSTANDARD LVCMOS33} [get_ports {data[1]}]
set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports {data[2]}]
set_property -dict {PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports {data[3]}]
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {RS}]
set_property -dict {PACKAGE_PIN V18 IOSTANDARD LVCMOS33} [get_ports {EN}]

# Clock signal 125 MHz
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { CLK }];
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 5} [get_ports { CLK }];
```

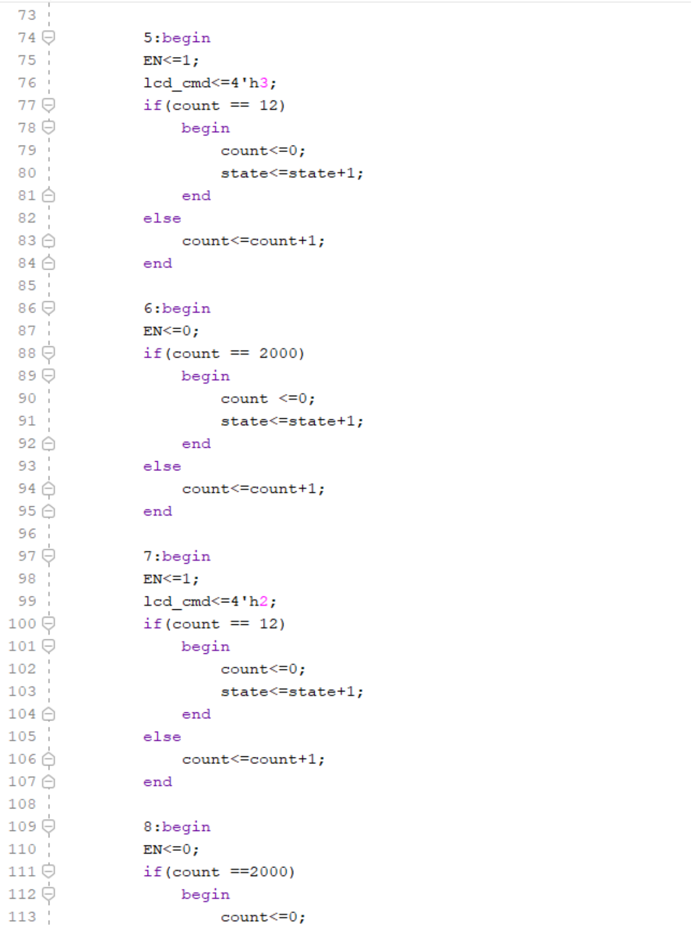
3.2 LCD.V



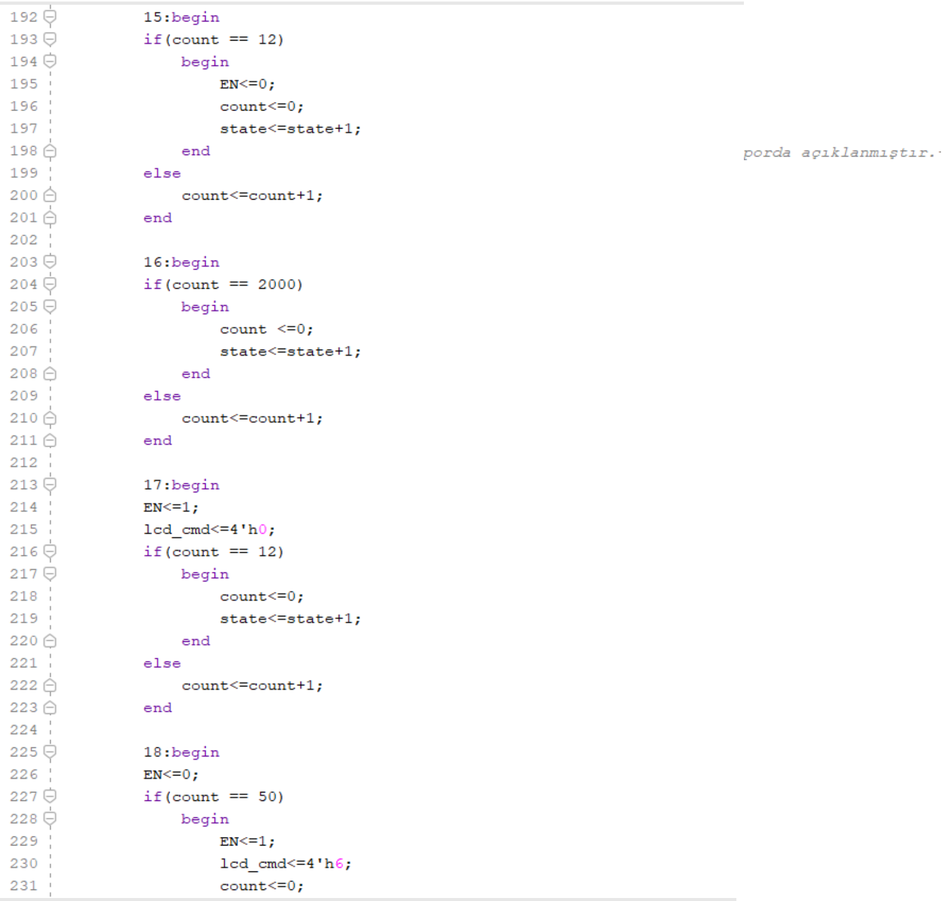
```
`timescale 1ns / 1ps

module LCD(input CLK,
    output [3:0] data,
    output reg RS, EN);

    reg [3:0] lcd_cmd;
    reg [5:0]state = 0;
    reg [25:0]count=0;

    assign data = lcd_cmd;

    always @(posedge CLK) begin
        case(state)
//init işlemlerinin başlangıcı. Bu aşama raporda detaylı olarak anlatılmıştır.
        0:begin
        EN <= 0;
        RS <= 0;
        if(count == 1875000)
            begin
                count <= 0;
                state <= state + 1;
            end
        else
            count <= count+1;
        end

        1: begin
        EN <= 1;
        lcd_cmd <= 4'h3;
        if(count == 12)
            begin
                count <= 0;
                state <= state + 1;
        end
        else
            count<=count+1;
        end

```



```
2:begin
EN<=0;
if(count ==205000)
    begin
        count <= 0;
        state <= state+1;
    end
else
    count<=count+1;
end

3:begin
EN<=1;
lcd_cmd<=4'h3;
if(count==12)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

4:begin
EN<=0;
if(count == 5000)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end
```

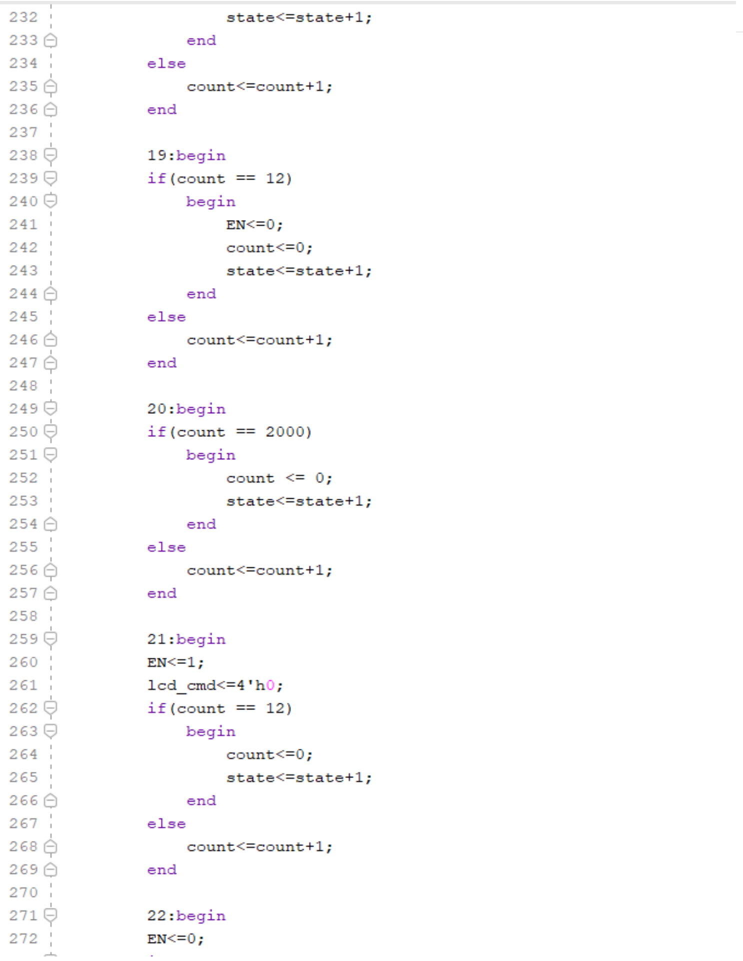


```

5:begin
EN<=1;
lcd_cmd<=4'h3;
if(count == 12)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

6:begin
EN<=0;
if(count == 2000)
    begin
        count <=0;
        state<=state+1;
    end
else
    count<=count+1;
end

7:begin
EN<=1;
lcd_cmd<=4'h2;
if(count == 12)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

8:begin
EN<=0;
if(count ==2000)
    begin
        count<=0;
```

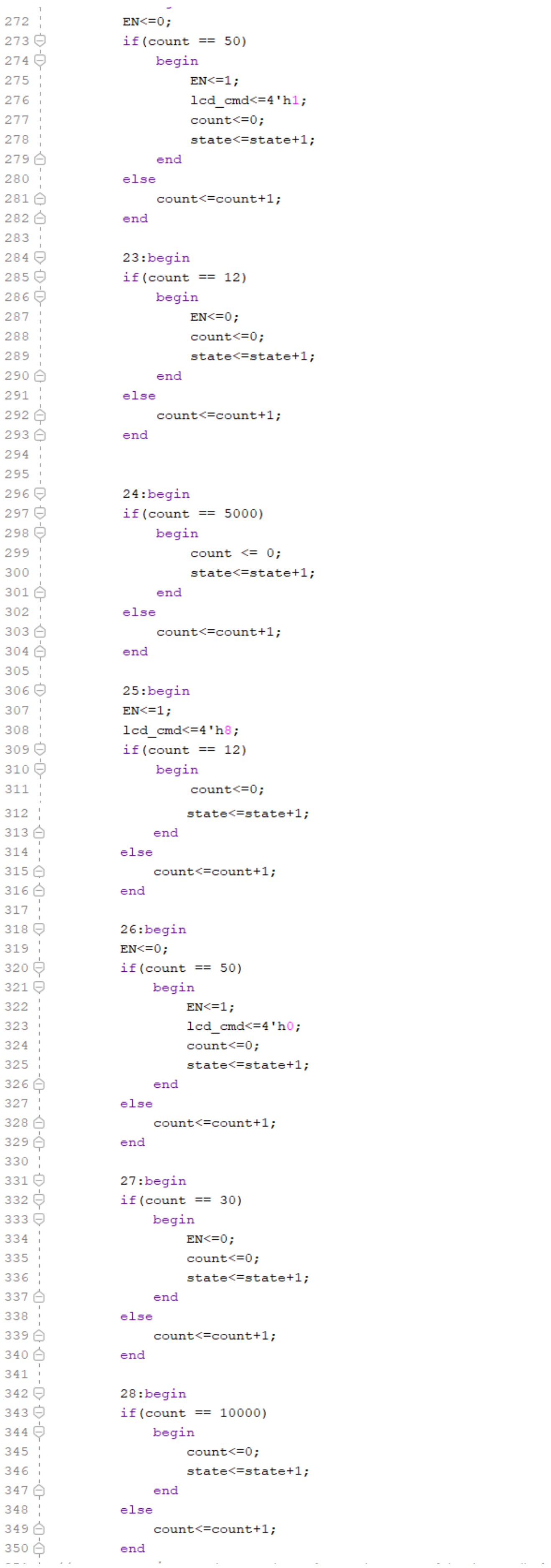


```
15:begin
if(count == 12)
    begin
        EN<=0;
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

16:begin
if(count == 2000)
    begin
        count <=0;
        state<=state+1;
    end
else
    count<=count+1;
end

17:begin
EN<=1;
lcd_cmd<=4'h0;
if(count == 12)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

18:begin
EN<=0;
if(count == 50)
    begin
        EN<=1;
        lcd_cmd<=4'h6;
        count<=0;
```

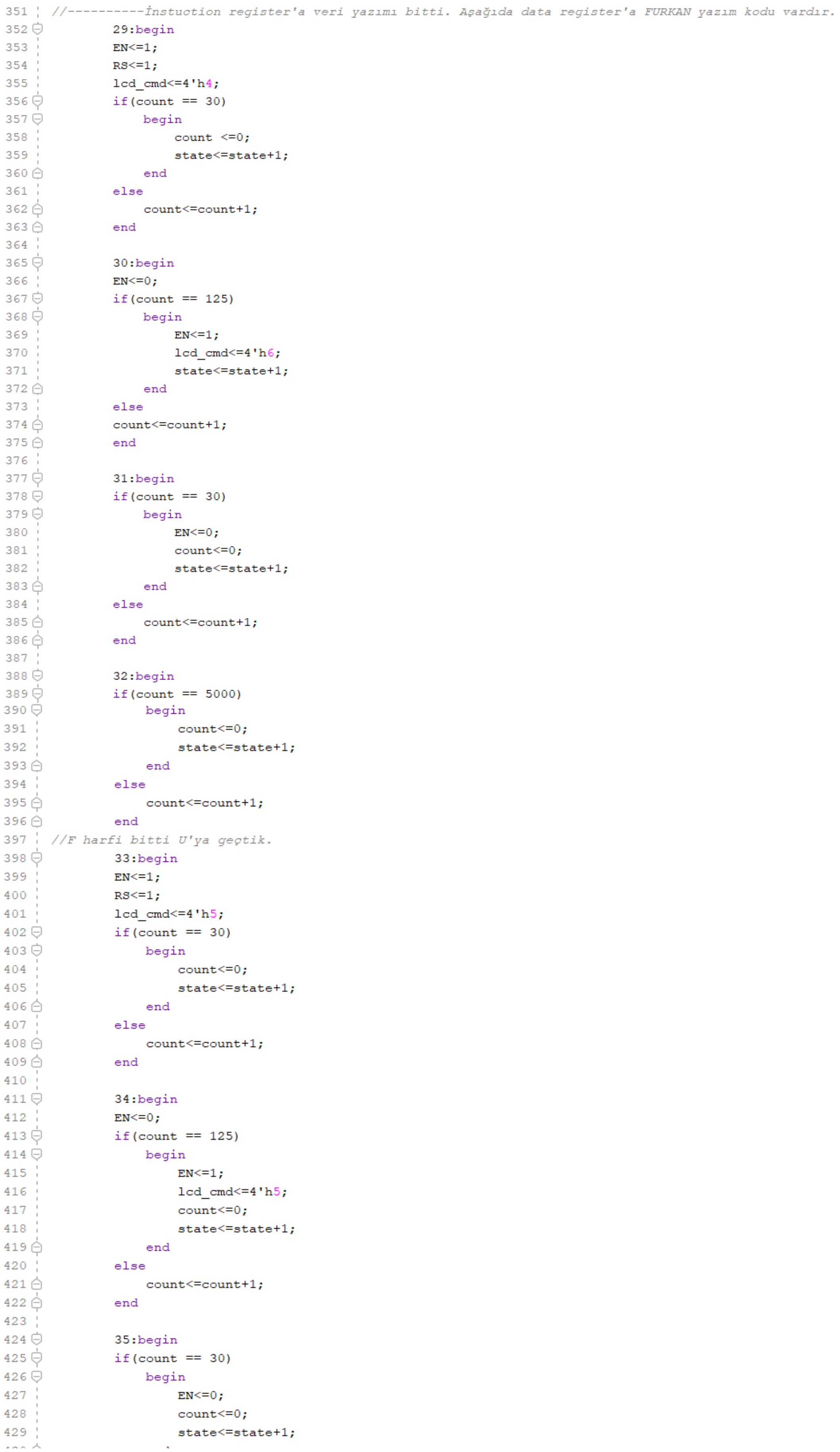
porda açıklanmıştır.·



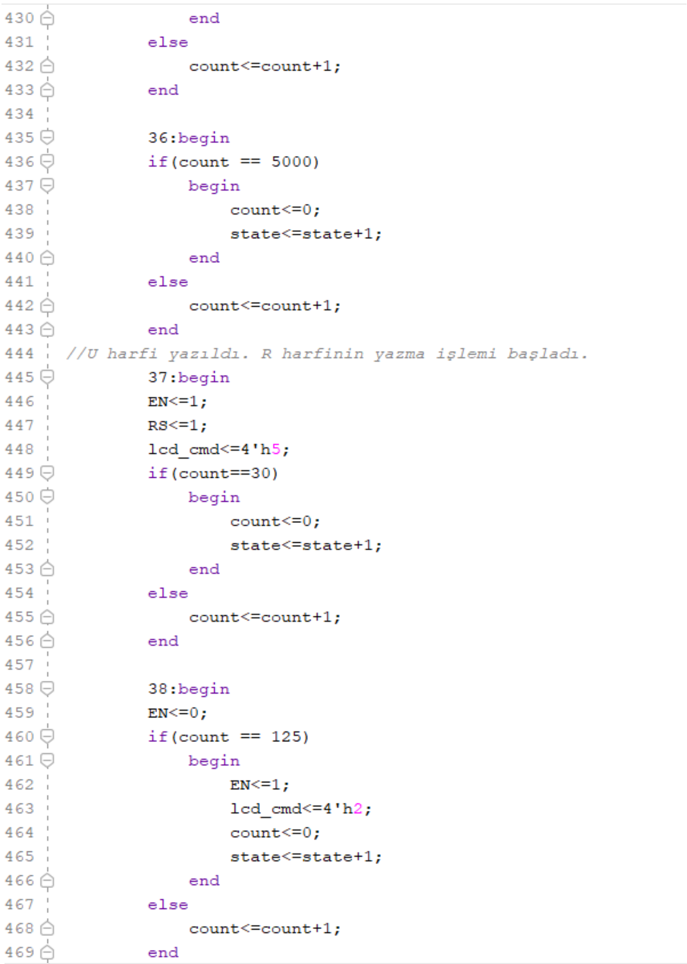
```
        state<=state+1;
    end
else
    count<=count+1;
end

19:begin
if(count == 12)
    begin
        EN<=0;
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

20:begin
if(count == 2000)
    begin
        count <= 0;
        state<=state+1;
    end
else
    count<=count+1;
end

21:begin
EN<=1;
lcd_cmd<=4'h0;
if(count == 12)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

22:begin
EN<=0;
```



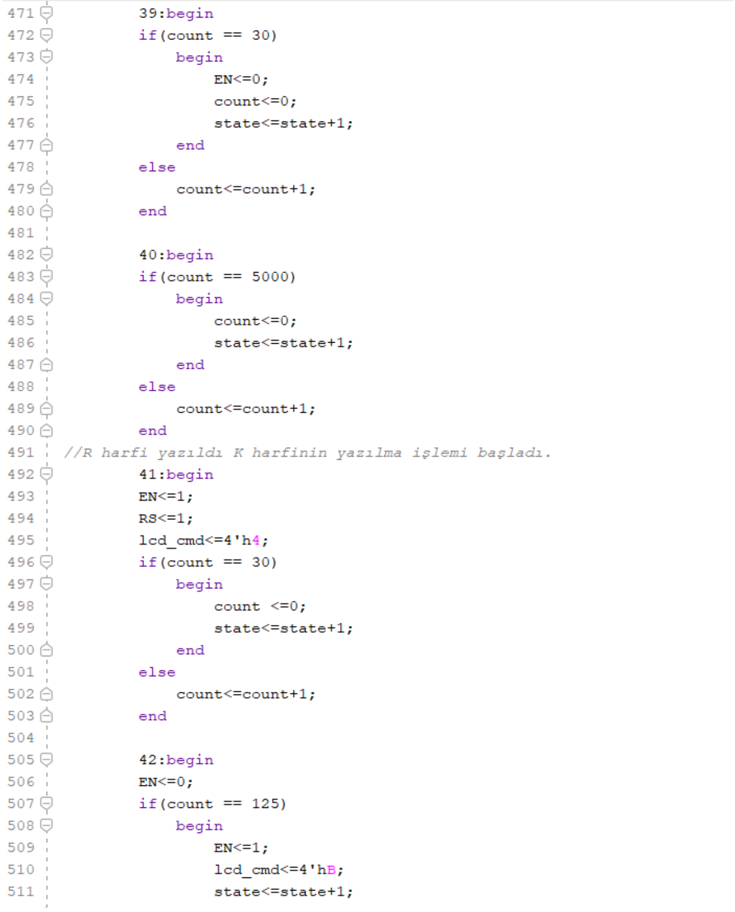
```
EN<=0;
if(count == 50)
    begin
        EN<=1;
        lcd_cmd<=4'h1;
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

23:begin
if(count == 12)
    begin
        EN<=0;
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end


24:begin
if(count == 5000)
    begin
        count <= 0;
        state<=state+1;
    end
else
    count<=count+1;
end

25:begin
EN<=1;
lcd_cmd<=4'h8;
if(count == 12)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

26:begin
EN<=0;
if(count == 50)
    begin
        EN<=1;
        lcd_cmd<=4'h0;
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

27:begin
if(count == 30)
    begin
        EN<=0;
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end

28:begin
if(count == 10000)
    begin
        count<=0;
        state<=state+1;
    end
else
    count<=count+1;
end
```



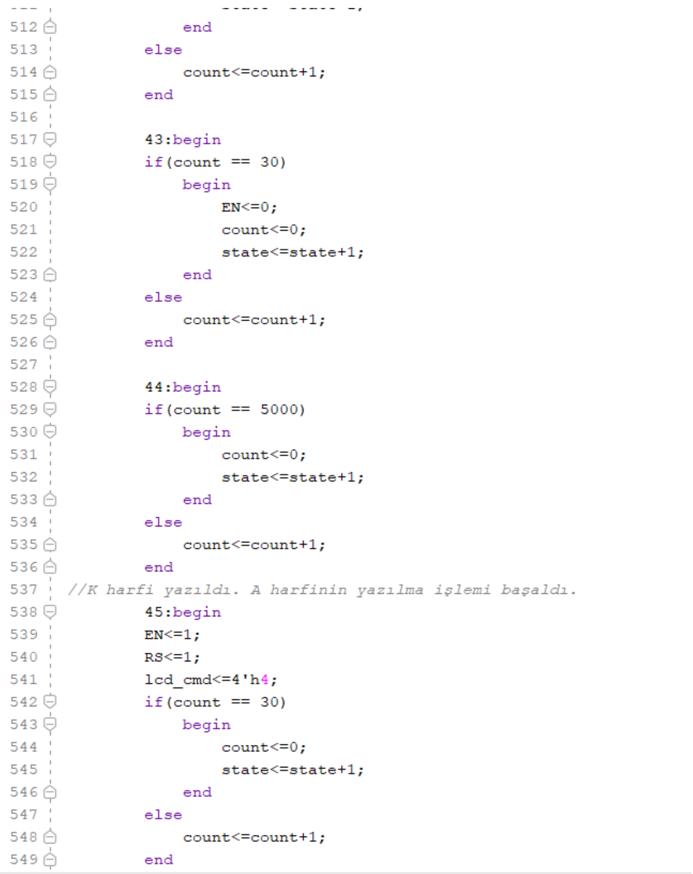
```
//----------İnstuction register'a veri yazımı bitti. Aşağıda data register'a FURKAN yazım kodu vardır.
        29:begin
        EN<=1;
        RS<=1;
        lcd_cmd<=4'h4;
        if(count == 30)
            begin
                count <=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        30:begin
        EN<=0;
        if(count == 125)
            begin
                EN<=1;
                lcd_cmd<=4'h6;
                state<=state+1;
            end
        else
        count<=count+1;
        end

        31:begin
        if(count == 30)
            begin
                EN<=0;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        32:begin
        if(count == 5000)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
//F harfi bitti U'ya geçtik.
        33:begin
        EN<=1;
        RS<=1;
        lcd_cmd<=4'h5;
        if(count == 30)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        34:begin
        EN<=0;
        if(count == 125)
            begin
                EN<=1;
                lcd_cmd<=4'h5;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        35:begin
        if(count == 30)
            begin
                EN<=0;
                count<=0;
                state<=state+1;
```



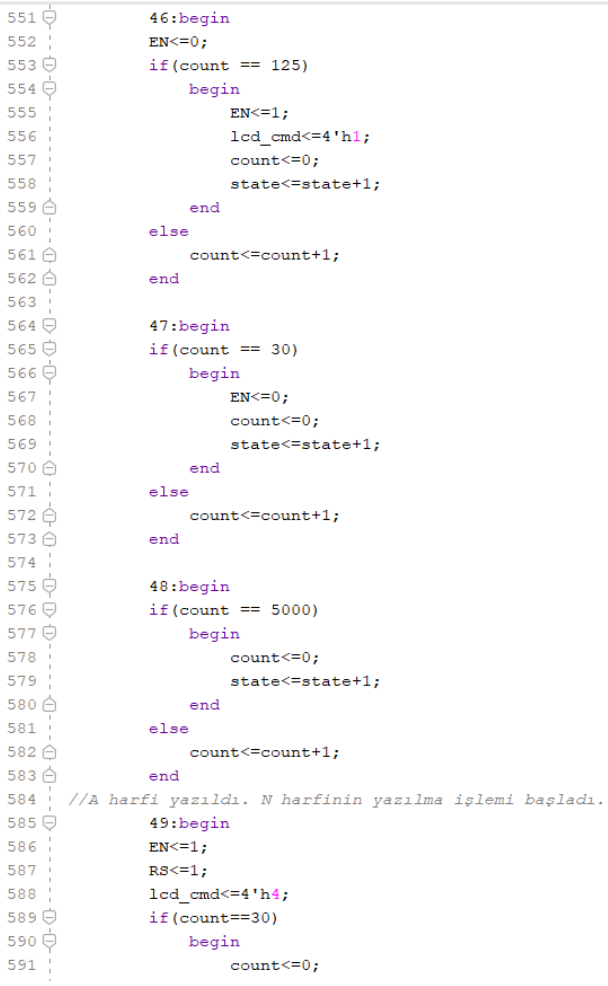
```
            end
        else
            count<=count+1;
        end

        36:begin
        if(count == 5000)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
//U harfi yazıldı. R harfinin yazma işlemi başladı.
        37:begin
        EN<=1;
        RS<=1;
        lcd_cmd<=4'h5;
        if(count==30)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        38:begin
        EN<=0;
        if(count == 125)
            begin
                EN<=1;
                lcd_cmd<=4'h2;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
```



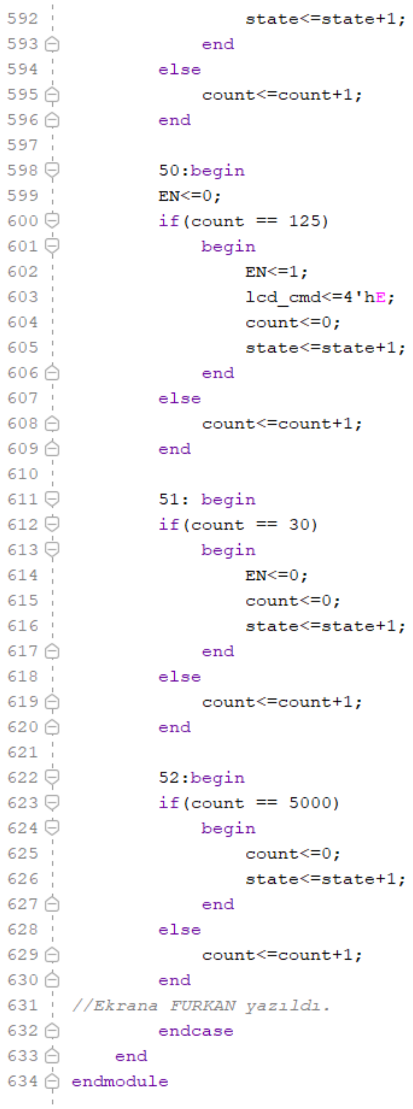
```
        39:begin
        if(count == 30)
            begin
                EN<=0;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        40:begin
        if(count == 5000)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
//R harfi yazıldı K harfinin yazılma işlemi başladı.
        41:begin
        EN<=1;
        RS<=1;
        lcd_cmd<=4'h4;
        if(count == 30)
            begin
                count <=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        42:begin
        EN<=0;
        if(count == 125)
            begin
                EN<=1;
                lcd_cmd<=4'hB;
                state<=state+1;
```



```
            end
        else
            count<=count+1;
        end

        43:begin
        if(count == 30)
            begin
                EN<=0;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        44:begin
        if(count == 5000)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
//K harfi yazıldı. A harfinin yazılma işlemi başaldı.
        45:begin
        EN<=1;
        RS<=1;
        lcd_cmd<=4'h4;
        if(count == 30)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
```

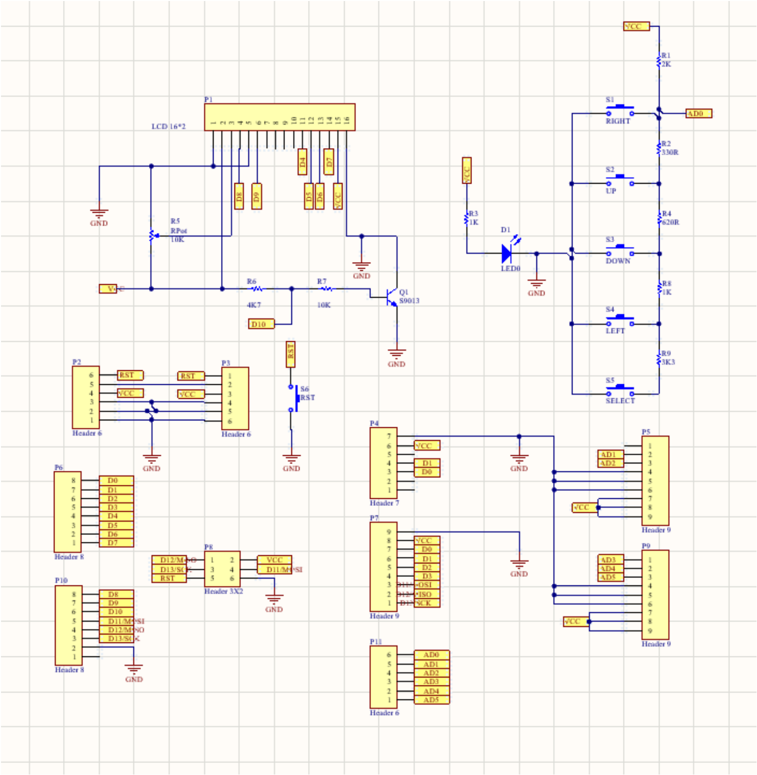


```
        46:begin
        EN<=0;
        if(count == 125)
            begin
                EN<=1;
                lcd_cmd<=4'h1;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        47:begin
        if(count == 30)
            begin
                EN<=0;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        48:begin
        if(count == 5000)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
//A harfi yazıldı. N harfinin yazılma işlemi başladı.
        49:begin
        EN<=1;
        RS<=1;
        lcd_cmd<=4'h4;
        if(count==30)
            begin
                count<=0;
```

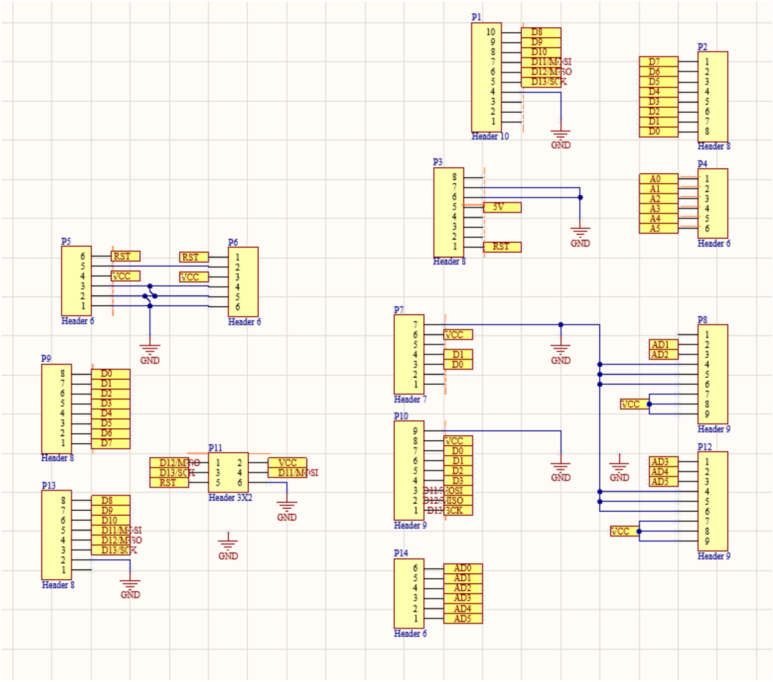


```
                state<=state+1;
            end
        else
            count<=count+1;
        end

        50:begin
        EN<=0;
        if(count == 125)
            begin
                EN<=1;
                lcd_cmd<=4'hE;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        51: begin
        if(count == 30)
            begin
                EN<=0;
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end

        52:begin
        if(count == 5000)
            begin
                count<=0;
                state<=state+1;
            end
        else
            count<=count+1;
        end
//Ekrana FURKAN yazıldı.
        endcase
    end
endmodule
```

4. SCHEMATICS

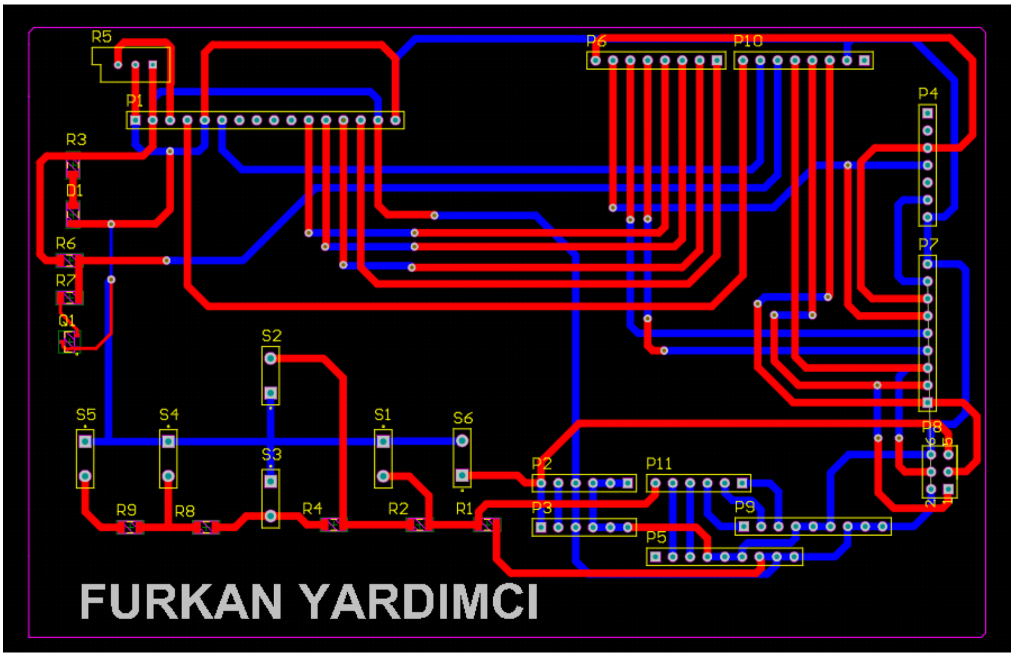


Img 5: Schematic1

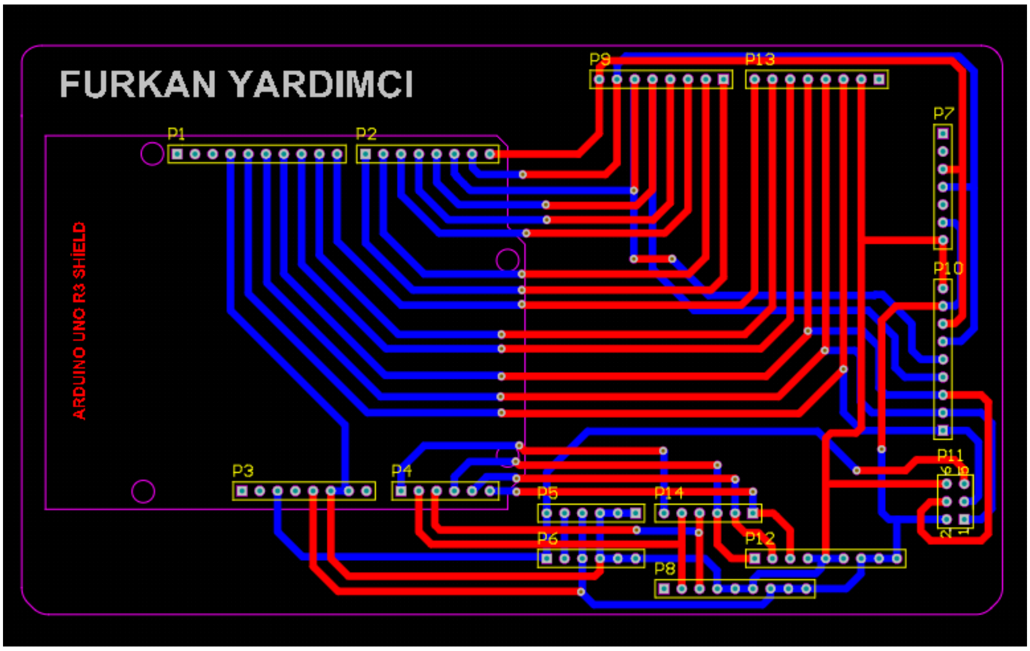


Img 6: Schematic2

5. PCBS

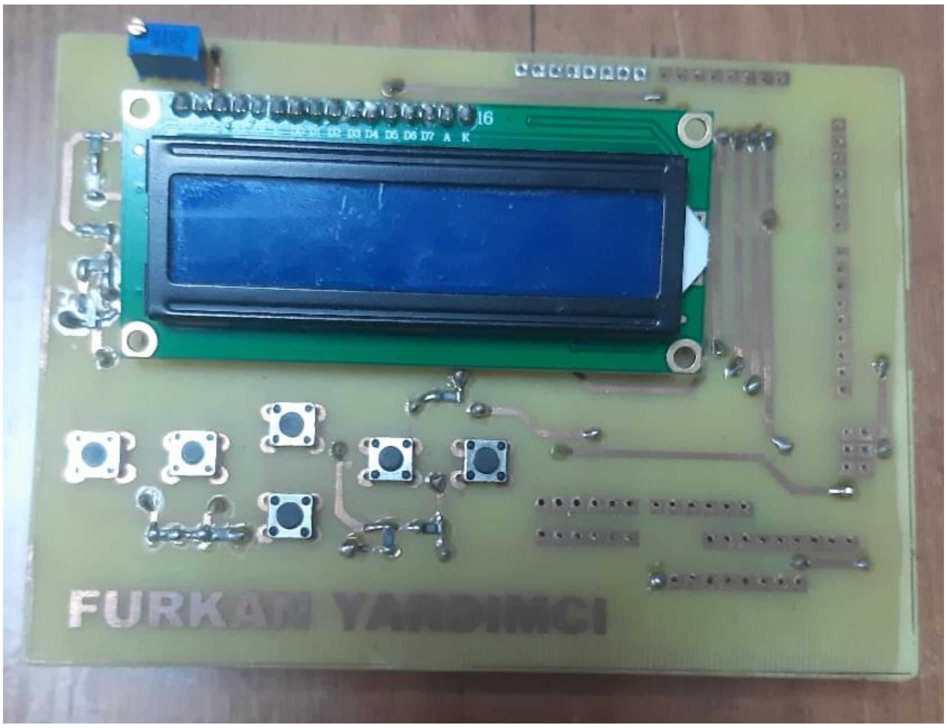


Img 7: PCB1

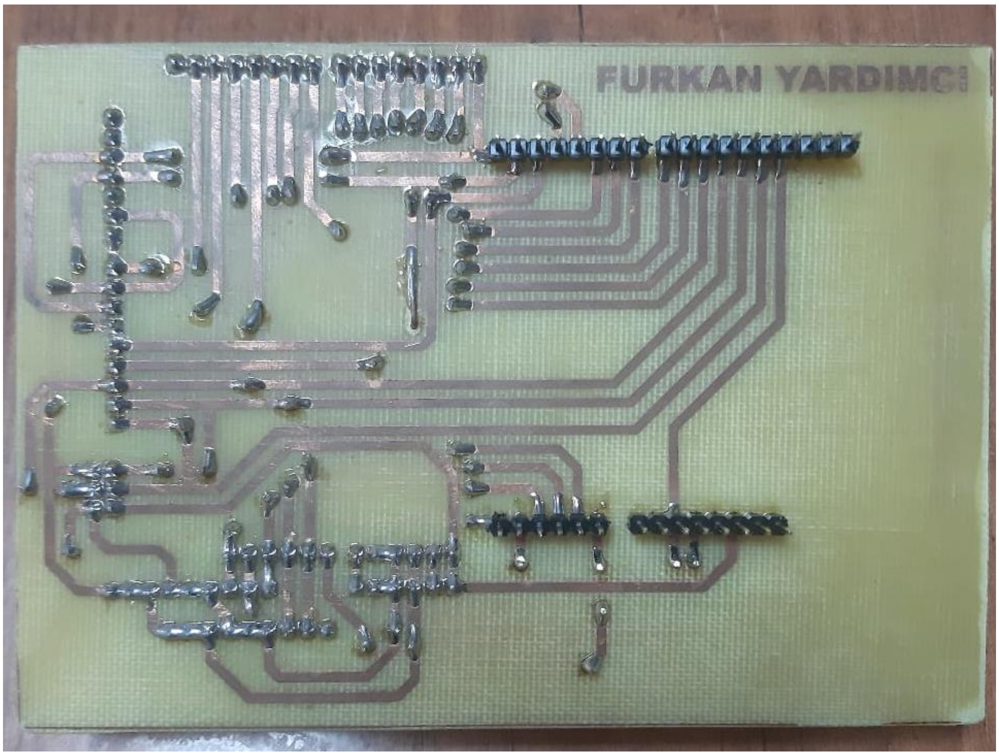


Img 8: PCB2

6. FINAL VIEW OF PROJECT



Img 9: Front view of the project



Img 10: Back view of the project

7.REFERENCES:

[1] https://www.youtube.com/watch?v=8YYZVVcnVpM&t=141s

[2] https://linuxhint.com/interface-lcd-4-bit-8-bit-modes-arduino/

[3] ARDUINO LCD KEYPAD SHIELD ÖRNEK UYGULAMA - Elektronik Bilgi Paylaşım Platformu (arduinocuyuz.blogspot.com)

[4] PYNQ - Python productivity for Zynq - Board

[5] PYNQ-Z2 Setup Guide — Python productivity for Zynq (Pynq)

[6] A PYNQ-Z2 Guide for Absolute Dummies — Part I: Fun with LEDs and Switches | by Umer Farooq | Medium (umer-farooq.com)

[7] 16x2 LCD Display Module - Pinout & Datasheet (circuitdigest.com)