Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Report week 7: LCD, Keyboard and Timers integration.

Laboratory of Microcontrollers

Gilberto Ochoa Ruiz

Perla Vanessa Jaime Gaytán ITE

A00344428

Nicol Gómez Tisnado ITE

A00227180

Instituto Tecnológico de Estudios Superiores de Monterrey Campus Guadalajara

Zapopan, Jalisco, México.

May 6, 2020

3. Make some research in how to connect the board to the external components. You can use a virtual program to schematize the connection between the board and the external components (LCD screen and keyboard, external components). An example is Fritzing, but there are others

> The KL25Z board has input and output pins which allow the board to receive information coming from the outside, in this case the key matrix, and also allow it to provide information in this case throughout the LCD for the upcoming document we will be able to see a couple of schematics representing the connection between the board and external components.

Part 1. Menu and output management. Write a very simple program that displays the following message in the LCD screen

PRESS BUTTON

R: 1 G: 2 R: 3

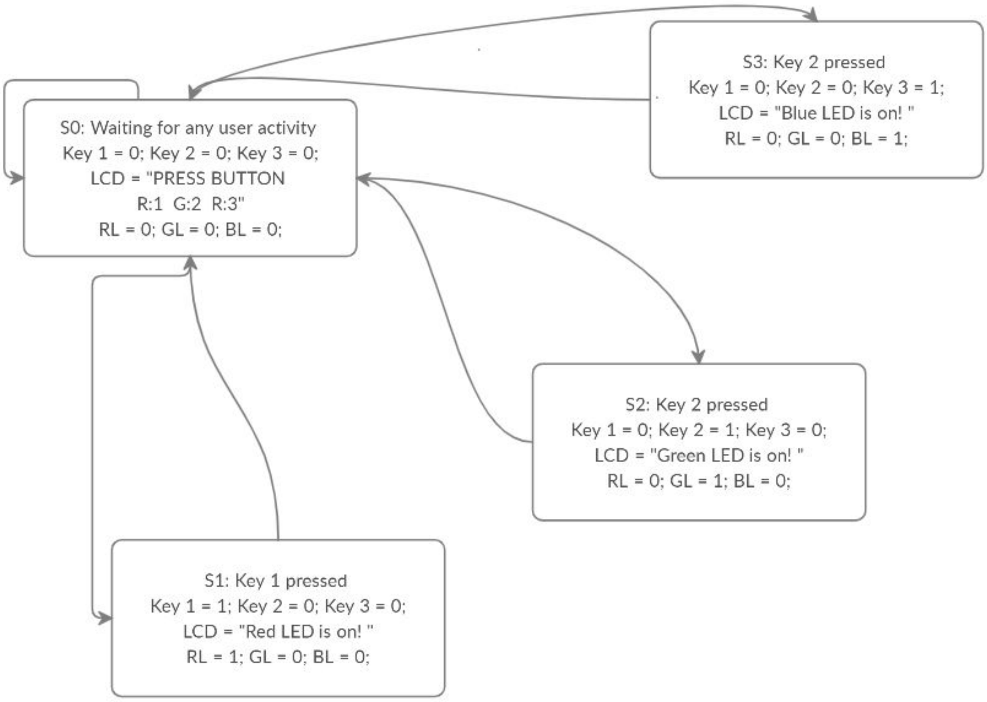
Then write a function that recovers the data from the get_key() function (plus a decoder) and switch on a led depending on which key was pressed (1: Red, 2: Blue, 3: Green) and display

RED/BLUE/GREEN

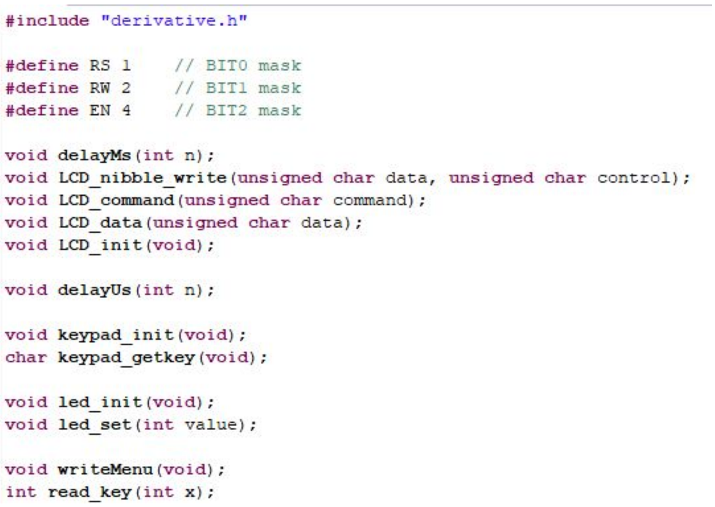
LED IS ON!

The led should remain on for a few seconds, then go off and display the initial menu again.

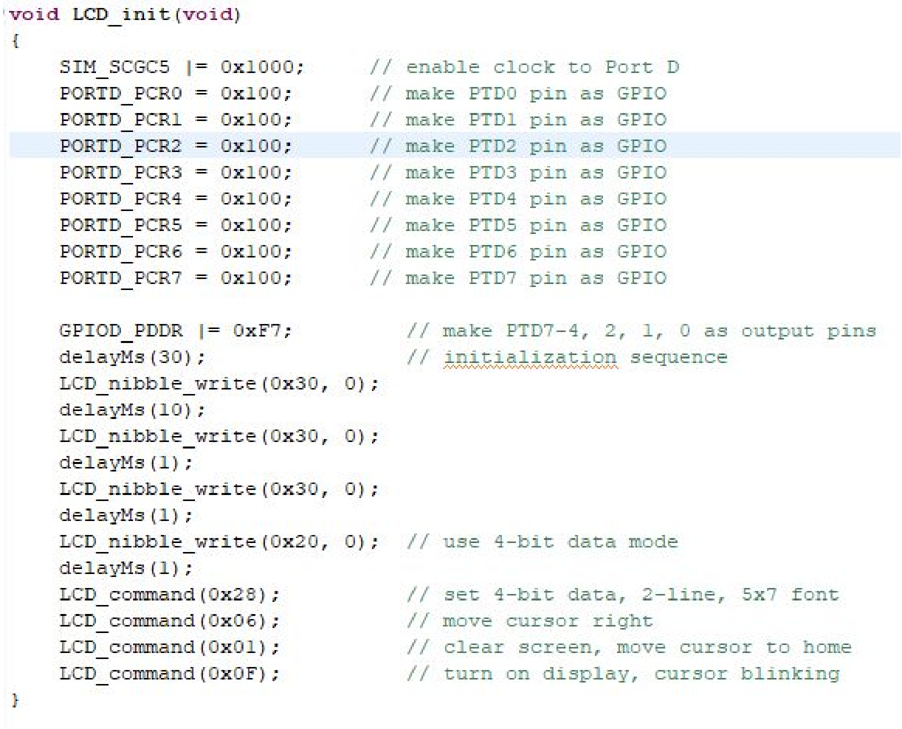
State Machine or Flow Diagram:



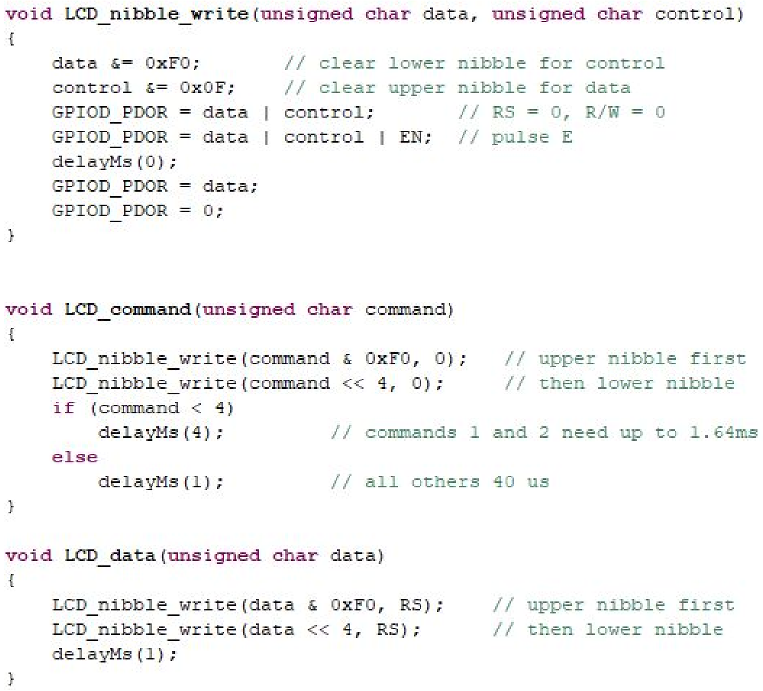
Code:



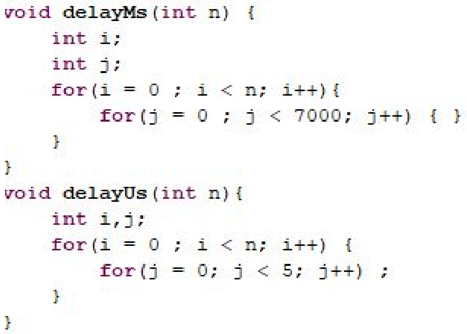
```c
#include "derivative.h"

#define RS 1    // BIT0 mask
#define RW 2    // BIT1 mask
#define EN 4    // BIT2 mask

void delayMs(int n);
void LCD_nibble_write(unsigned char data, unsigned char control);
void LCD_command(unsigned char command);
void LCD_data(unsigned char data);
void LCD_init(void);

void delayUs(int n);

void keypad_init(void);
char keypad_getkey(void);

void led_init(void);
void led_set(int value);

void writeMenu(void);
int read_key(int x);
```



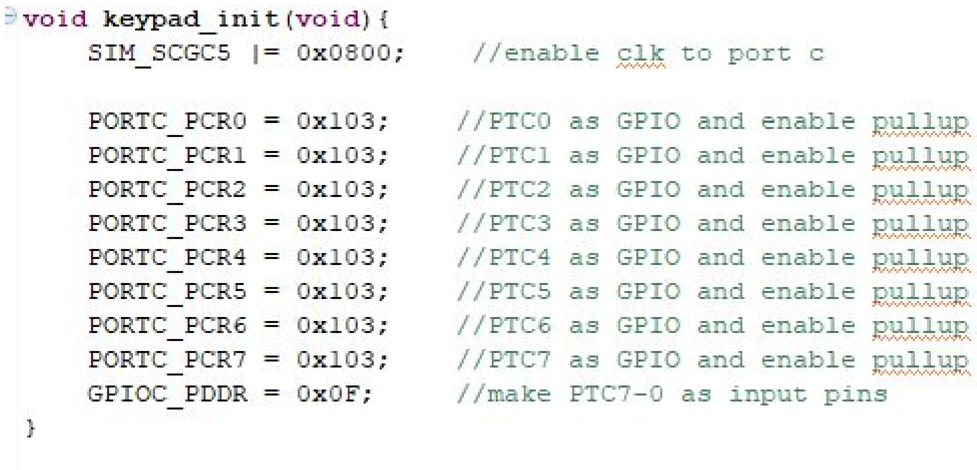
```
void LCD_init(void)
{
    SIM_SCGC5 |= 0x1000;      // enable clock to Port D
    PORTD_PCR0 = 0x100;       // make PTD0 pin as GPIO
    PORTD_PCR1 = 0x100;       // make PTD1 pin as GPIO
    PORTD_PCR2 = 0x100;       // make PTD2 pin as GPIO
    PORTD_PCR3 = 0x100;       // make PTD3 pin as GPIO
    PORTD_PCR4 = 0x100;       // make PTD4 pin as GPIO
    PORTD_PCR5 = 0x100;       // make PTD5 pin as GPIO
    PORTD_PCR6 = 0x100;       // make PTD6 pin as GPIO
    PORTD_PCR7 = 0x100;       // make PTD7 pin as GPIO

    GPIOD_PDDR |= 0xF7;          // make PTD7-4, 2, 1, 0 as output pins
    delayMs(30);                 // initialization sequence
    LCD_nibble_write(0x30, 0);
    delayMs(10);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x20, 0);   // use 4-bit data mode
    delayMs(1);
    LCD_command(0x28);           // set 4-bit data, 2-line, 5x7 font
    LCD_command(0x06);           // move cursor right
    LCD_command(0x01);           // clear screen, move cursor to home
    LCD_command(0x0F);           // turn on display, cursor blinking
}
```



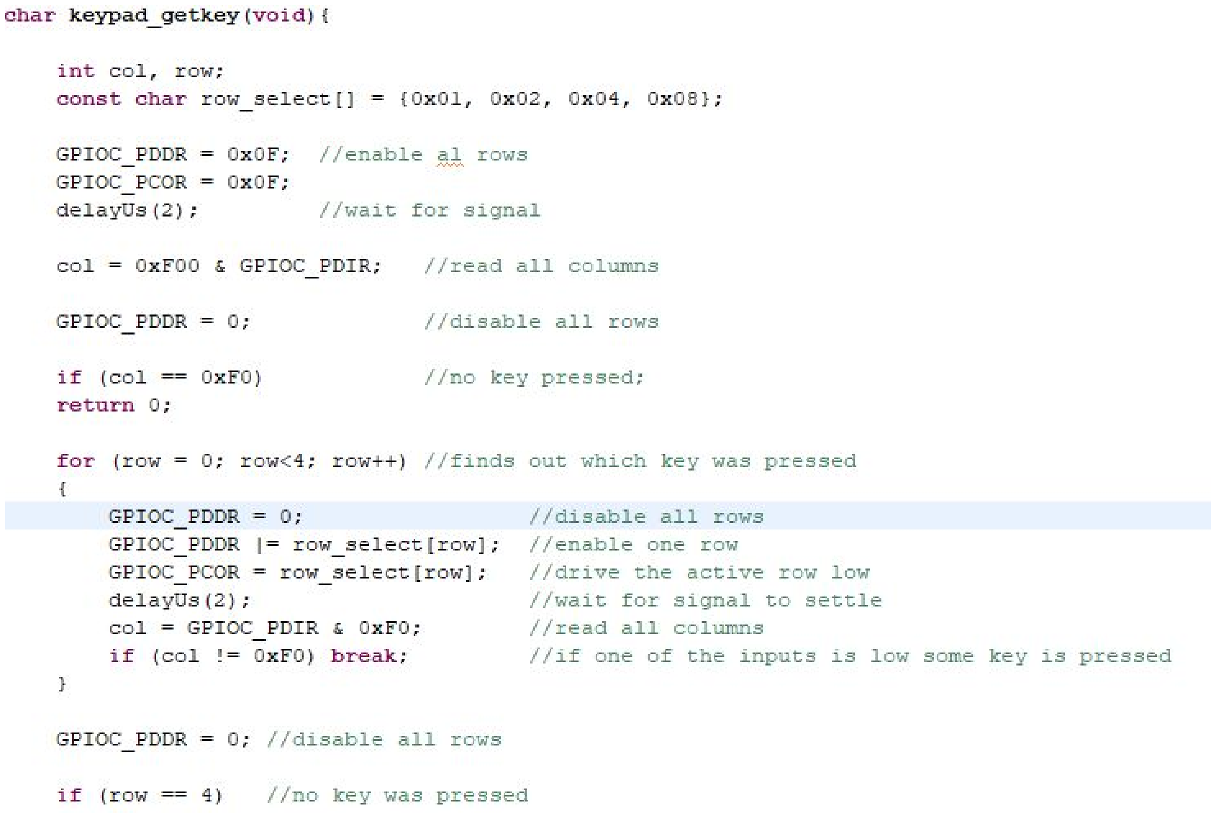
```
void LCD_nibble_write(unsigned char data, unsigned char control)
{
    data &= 0xF0;       // clear lower nibble for control
    control &= 0x0F;    // clear upper nibble for data
    GPIOD_PDOR = data | control;       // RS = 0, R/W = 0
    GPIOD_PDOR = data | control | EN;  // pulse E
    delayMs(0);
    GPIOD_PDOR = data;
    GPIOD_PDOR = 0;
}


void LCD_command(unsigned char command)
{
    LCD_nibble_write(command & 0xF0, 0);   // upper nibble first
    LCD_nibble_write(command << 4, 0);     // then lower nibble
    if (command < 4)
        delayMs(4);         // commands 1 and 2 need up to 1.64ms
    else
        delayMs(1);         // all others 40 us
}

void LCD_data(unsigned char data)
{
    LCD_nibble_write(data & 0xF0, RS);    // upper nibble first
    LCD_nibble_write(data << 4, RS);      // then lower nibble
    delayMs(1);
}
```



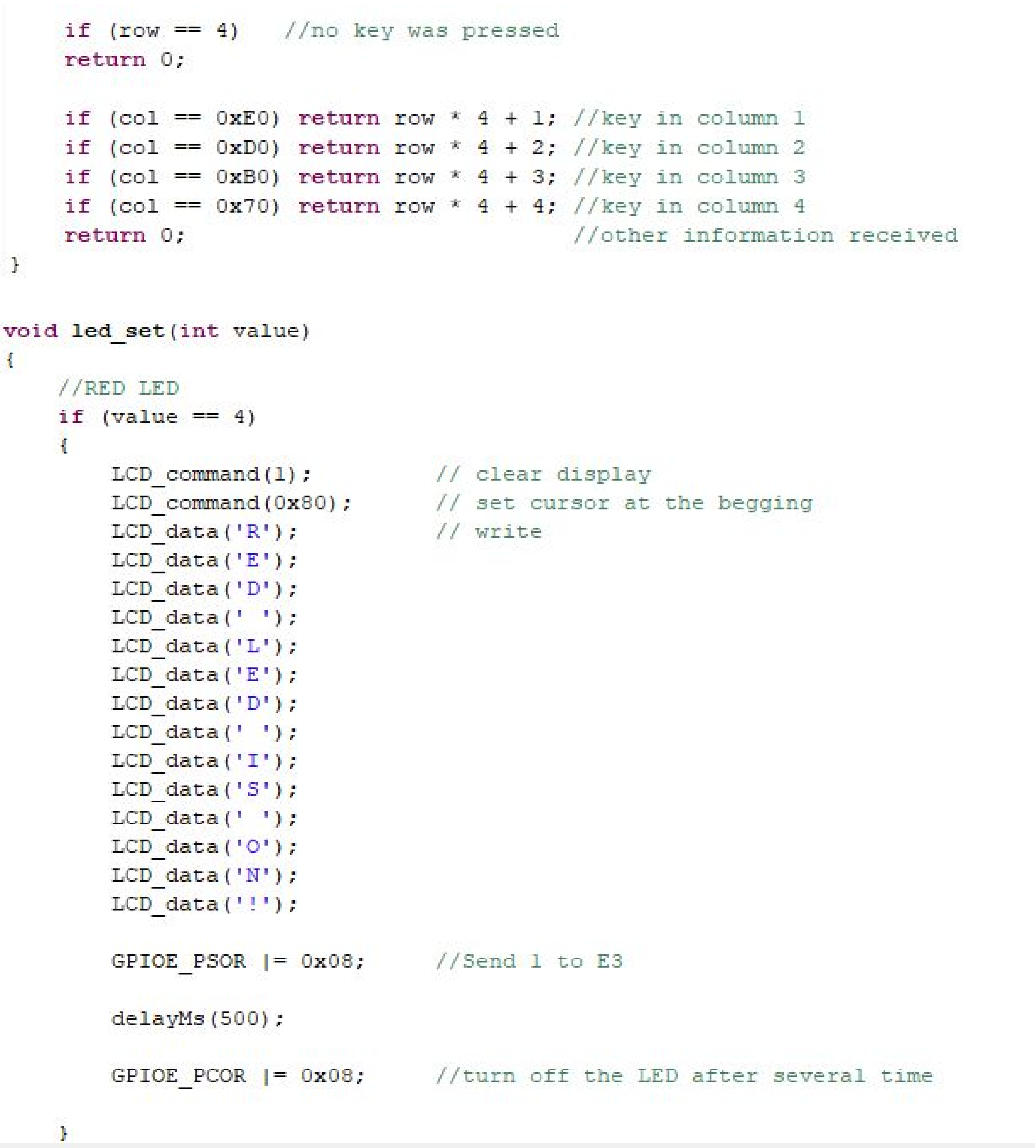
```
void delayMs(int n) {
    int i;
    int j;
    for(i = 0 ; i < n; i++){
        for(j = 0 ; j < 7000; j++) { }
    }
}
void delayUs(int n){
    int i,j;
    for(i = 0 ; i < n; i++) {
        for(j = 0; j < 5; j++) ;
    }
}
```



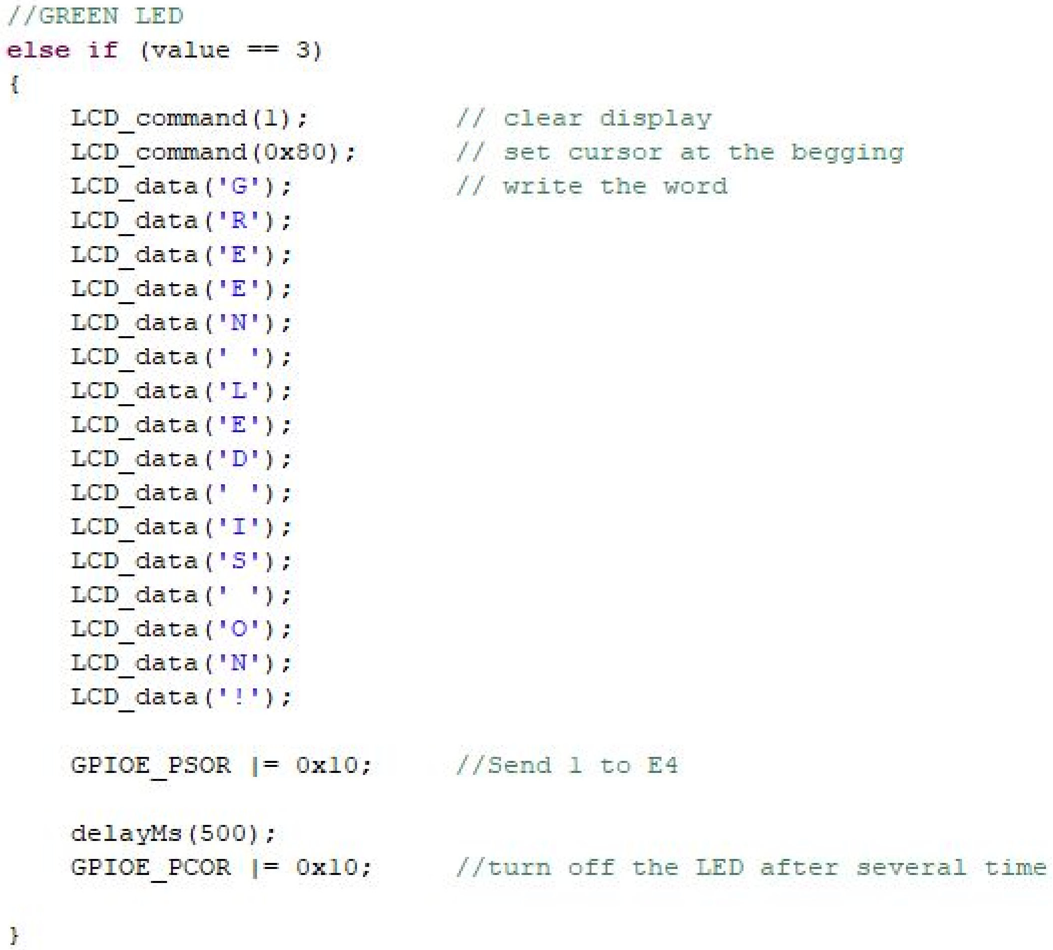
```
void keypad_init(void){
    SIM_SCGC5 |= 0x0800;    //enable clk to port c

    PORTC_PCR0 = 0x103;    //PTC0 as GPIO and enable pullup
    PORTC_PCR1 = 0x103;    //PTC1 as GPIO and enable pullup
    PORTC_PCR2 = 0x103;    //PTC2 as GPIO and enable pullup
    PORTC_PCR3 = 0x103;    //PTC3 as GPIO and enable pullup
    PORTC_PCR4 = 0x103;    //PTC4 as GPIO and enable pullup
    PORTC_PCR5 = 0x103;    //PTC5 as GPIO and enable pullup
    PORTC_PCR6 = 0x103;    //PTC6 as GPIO and enable pullup
    PORTC_PCR7 = 0x103;    //PTC7 as GPIO and enable pullup
    GPIOC_PDDR = 0x0F;     //make PTC7-0 as input pins
}
```



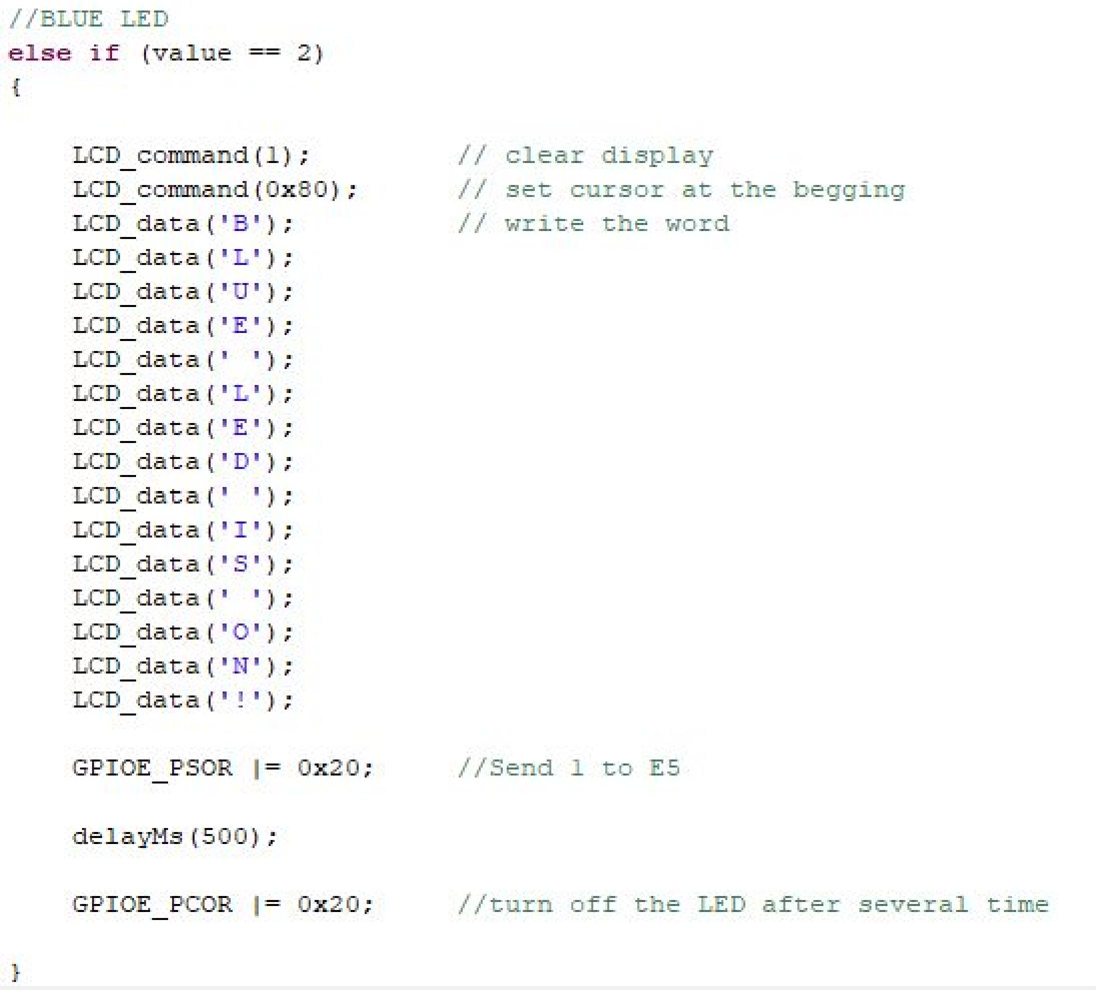
```
char keypad_getkey(void){

    int col, row;
    const char row_select[] = {0x01, 0x02, 0x04, 0x08};

    GPIOC_PDDR = 0x0F;  //enable al rows
    GPIOC_PCOR = 0x0F;
    delayUs(2);         //wait for signal

    col = 0xF00 & GPIOC_PDIR;   //read all columns

    GPIOC_PDDR = 0;             //disable all rows

    if (col == 0xF0)            //no key pressed;
    return 0;

    for (row = 0; row<4; row++) //finds out which key was pressed
    {
        GPIOC_PDDR = 0;                 //disable all rows
        GPIOC_PDDR |= row_select[row];  //enable one row
        GPIOC_PCOR = row_select[row];   //drive the active row low
        delayUs(2);                     //wait for signal to settle
        col = GPIOC_PDIR & 0xF0;        //read all columns
        if (col != 0xF0) break;         //if one of the inputs is low some key is pressed
    }

    GPIOC_PDDR = 0; //disable all rows

    if (row == 4)   //no key was pressed
```



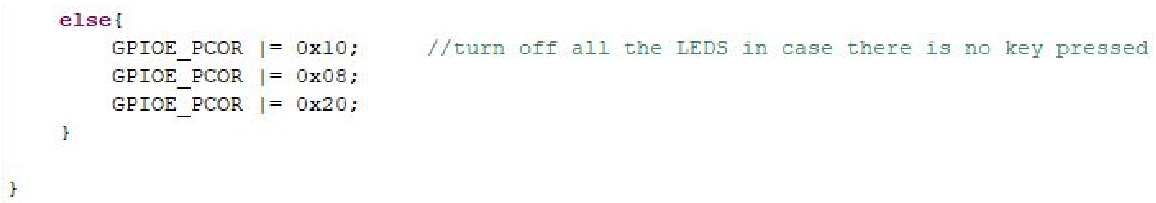
```
    if (row == 4)   //no key was pressed
    return 0;

    if (col == 0xE0) return row * 4 + 1; //key in column 1
    if (col == 0xD0) return row * 4 + 2; //key in column 2
    if (col == 0xB0) return row * 4 + 3; //key in column 3
    if (col == 0x70) return row * 4 + 4; //key in column 4
    return 0;                            //other information received
}

void led_set(int value)
{
    //RED LED
    if (value == 4)
    {
        LCD_command(1);         // clear display
        LCD_command(0x80);      // set cursor at the begging
        LCD_data('R');          // write
        LCD_data('E');
        LCD_data('D');
        LCD_data(' ');
        LCD_data('L');
        LCD_data('E');
        LCD_data('D');
        LCD_data(' ');
        LCD_data('I');
        LCD_data('S');
        LCD_data(' ');
        LCD_data('O');
        LCD_data('N');
        LCD_data('!');

        GPIOE_PSOR |= 0x08;     //Send 1 to E3

        delayMs(500);

        GPIOE_PCOR |= 0x08;     //turn off the LED after several time

    }
```



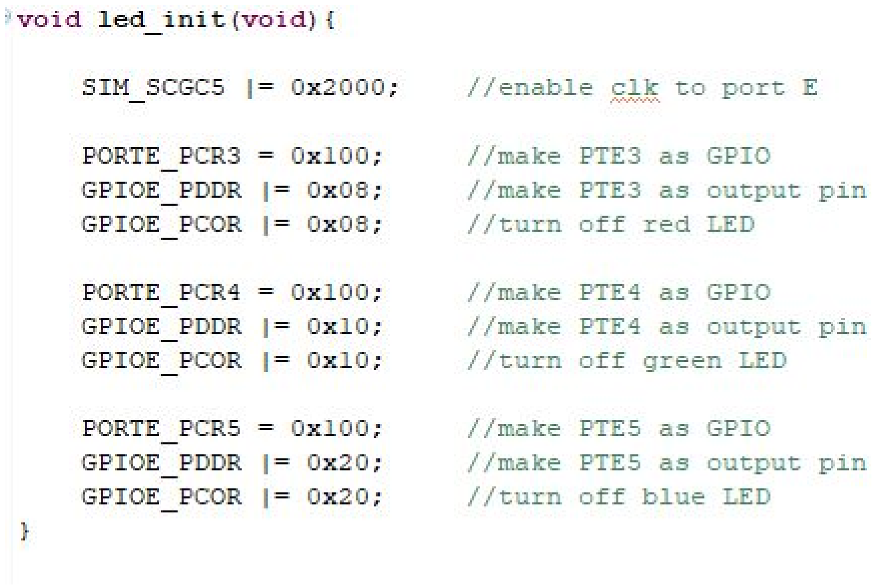
```
//GREEN LED
else if (value == 3)
{
    LCD_command(1);         // clear display
    LCD_command(0x80);      // set cursor at the begging
    LCD_data('G');          // write the word
    LCD_data('R');
    LCD_data('E');
    LCD_data('E');
    LCD_data('N');
    LCD_data(' ');
    LCD_data('L');
    LCD_data('E');
    LCD_data('D');
    LCD_data(' ');
    LCD_data('I');
    LCD_data('S');
    LCD_data(' ');
    LCD_data('O');
    LCD_data('N');
    LCD_data('!');

    GPIOE_PSOR |= 0x10;     //Send 1 to E4

    delayMs(500);
    GPIOE_PCOR |= 0x10;     //turn off the LED after several time

}
```



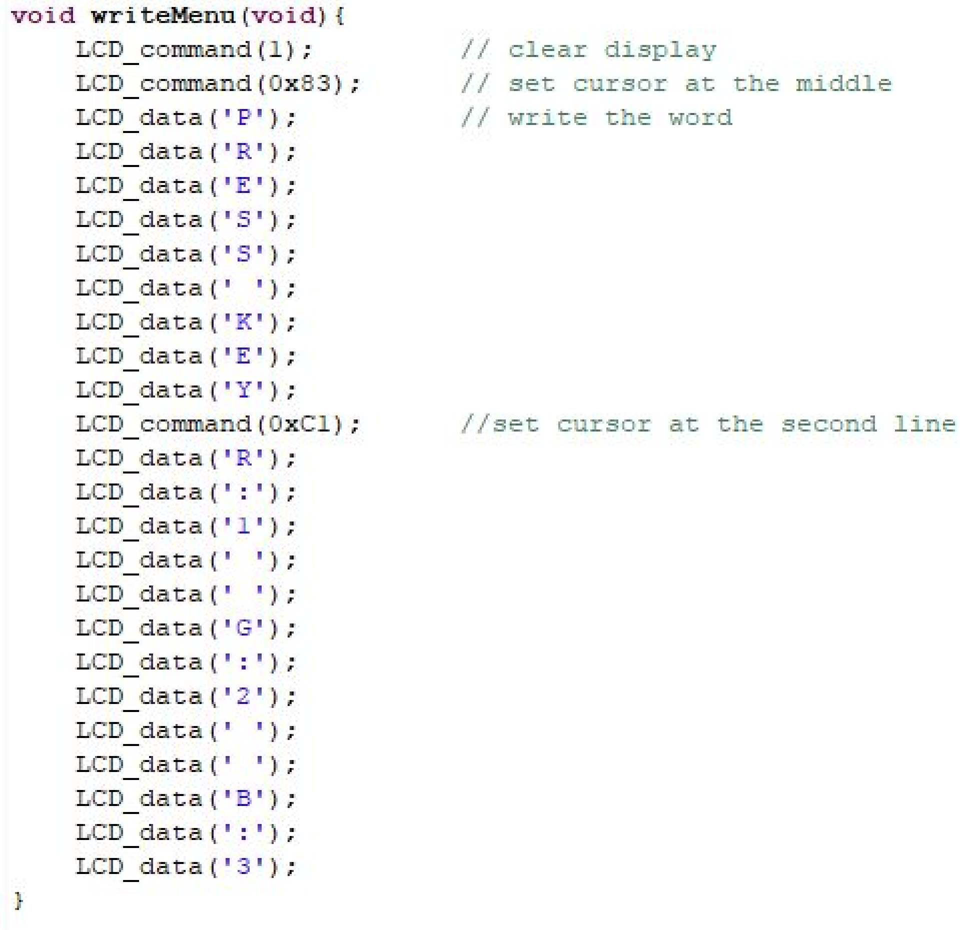
```
//BLUE LED
else if (value == 2)
{

    LCD_command(1);          // clear display
    LCD_command(0x80);       // set cursor at the begging
    LCD_data('B');           // write the word
    LCD_data('L');
    LCD_data('U');
    LCD_data('E');
    LCD_data(' ');
    LCD_data('L');
    LCD_data('E');
    LCD_data('D');
    LCD_data(' ');
    LCD_data('I');
    LCD_data('S');
    LCD_data(' ');
    LCD_data('O');
    LCD_data('N');
    LCD_data('!');

    GPIOE_PSOR |= 0x20;      //Send 1 to E5

    delayMs(500);

    GPIOE_PCOR |= 0x20;      //turn off the LED after several time

}
```



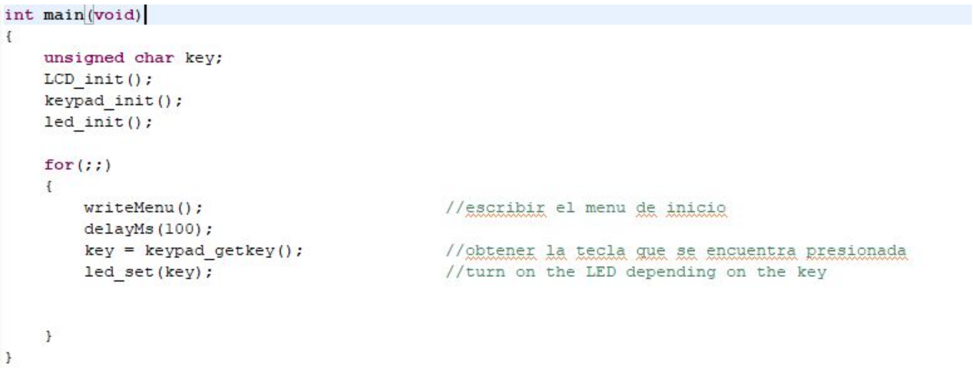
```
    else{
        GPIOE_PCOR |= 0x10;     //turn off all the LEDS in case there is no key pressed
        GPIOE_PCOR |= 0x08;
        GPIOE_PCOR |= 0x20;
    }

}
```



```c
void led_init(void){

    SIM_SCGC5 |= 0x2000;     //enable clk to port E

    PORTE_PCR3 = 0x100;      //make PTE3 as GPIO
    GPIOE_PDDR |= 0x08;      //make PTE3 as output pin
    GPIOE_PCOR |= 0x08;      //turn off red LED

    PORTE_PCR4 = 0x100;      //make PTE4 as GPIO
    GPIOE_PDDR |= 0x10;      //make PTE4 as output pin
    GPIOE_PCOR |= 0x10;      //turn off green LED

    PORTE_PCR5 = 0x100;      //make PTE5 as GPIO
    GPIOE_PDDR |= 0x20;      //make PTE5 as output pin
    GPIOE_PCOR |= 0x20;      //turn off blue LED
}
```

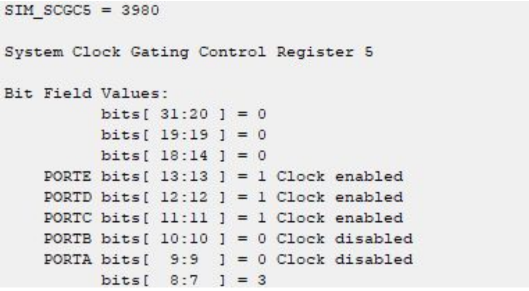


```c
void writeMenu(void){
    LCD_command(1);          // clear display
    LCD_command(0x83);       // set cursor at the middle
    LCD_data('P');           // write the word
    LCD_data('R');
    LCD_data('E');
    LCD_data('S');
    LCD_data('S');
    LCD_data(' ');
    LCD_data('K');
    LCD_data('E');
    LCD_data('Y');
    LCD_command(0xC1);       //set cursor at the second line
    LCD_data('R');
    LCD_data(':');
    LCD_data('1');
    LCD_data(' ');
    LCD_data(' ');
    LCD_data('G');
    LCD_data(':');
    LCD_data('2');
    LCD_data(' ');
    LCD_data(' ');
    LCD_data('B');
    LCD_data(':');
    LCD_data('3');
}
```

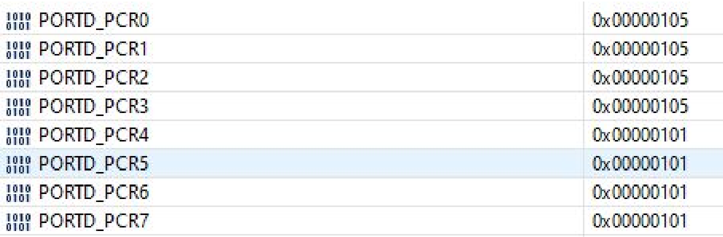


```
int main(void)
{
    unsigned char key;
    LCD_init();
    keypad_init();
    led_init();

    for(;;)
    {
        writeMenu();                        //escribir el menu de inicio
        delayMs(100);
        key = keypad_getkey();              //obtener la tecla que se encuentra presionada
        led_set(key);                       //turn on the LED depending on the key


    }
}
```

Registers:

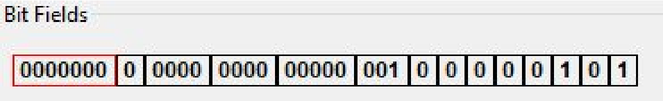


```
SIM_SCGC5 = 3980

System Clock Gating Control Register 5

Bit Field Values:
          bits[ 31:20 ] = 0
          bits[ 19:19 ] = 0
          bits[ 18:14 ] = 0
    PORTE bits[ 13:13 ] = 1 Clock enabled
    PORTD bits[ 12:12 ] = 1 Clock enabled
    PORTC bits[ 11:11 ] = 1 Clock enabled
    PORTB bits[ 10:10 ] = 0 Clock disabled
    PORTA bits[  9:9  ] = 0 Clock disabled
          bits[  8:7  ] = 3
```



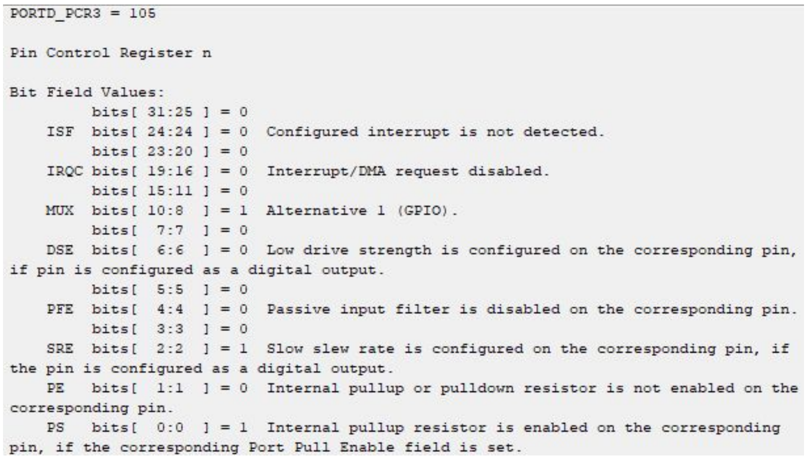
PORTD_PCR0	0x00000105
PORTD_PCR1	0x00000105
PORTD_PCR2	0x00000105
PORTD_PCR3	0x00000105
PORTD_PCR4	0x00000101
PORTD_PCR5	0x00000101
PORTD_PCR6	0x00000101
PORTD_PCR7	0x00000101

For PORTD_PCR0 to PORTD_PCR3 the configuration of the registers is the next.



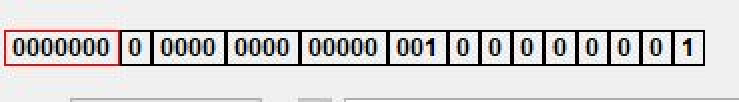
Bit Fields

0000000	0	0000	0000	00000	001	0	0	0	0	0	1	0	1

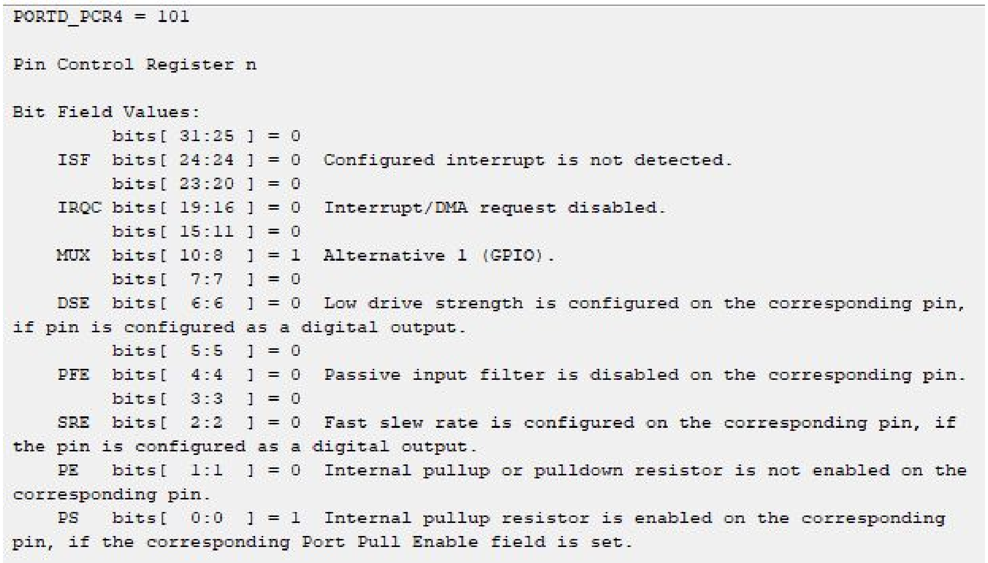


```
PORTD_PCR3 = 105

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 1  Slow slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 0  Internal pullup or pulldown resistor is not enabled on the
corresponding pin.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

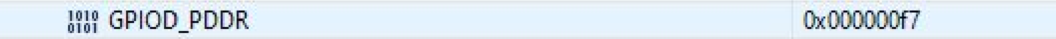
For PORTD_PCR4 to PORTD_PCR7 the configuration of the registers is the next.



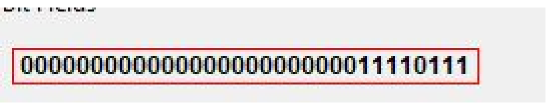
```
0000000 0 0000 0000 00000 001 0 0 0 0 0 0 0 1
```



```
PORTD_PCR4 = 101

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 0  Fast slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 0  Internal pullup or pulldown resistor is not enabled on the
corresponding pin.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

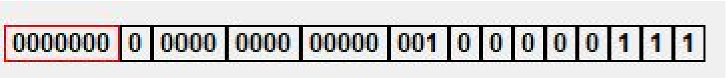


```
GPIOD_PDDR                                   0x000000f7
```

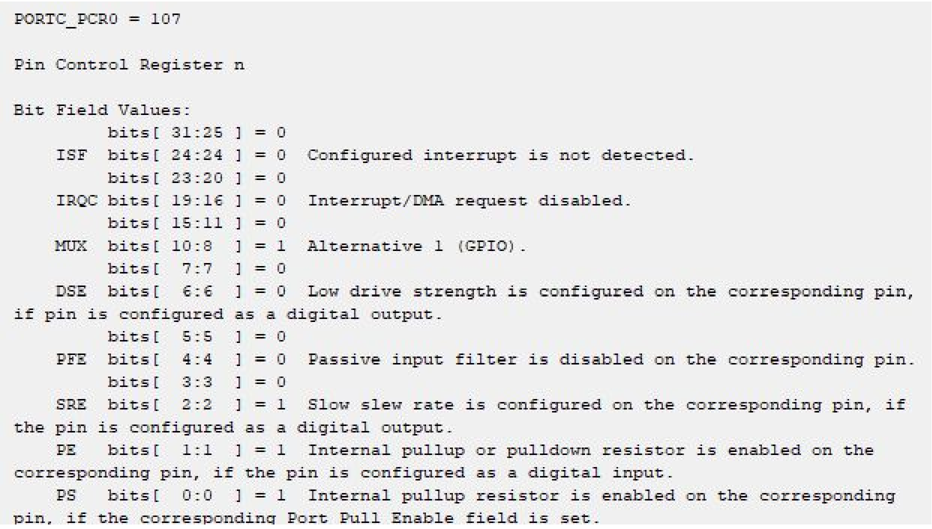


```
00000000000000000000000011110111
```

For PORTC_PCR0 to PORTC_PCR2 the configuration of the registers is the next.

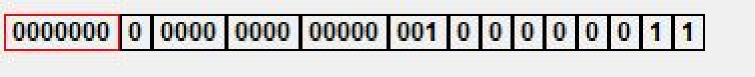


```
0000000 0 0000 0000 00000 001 0 0 0 0 0 1 1 1
```

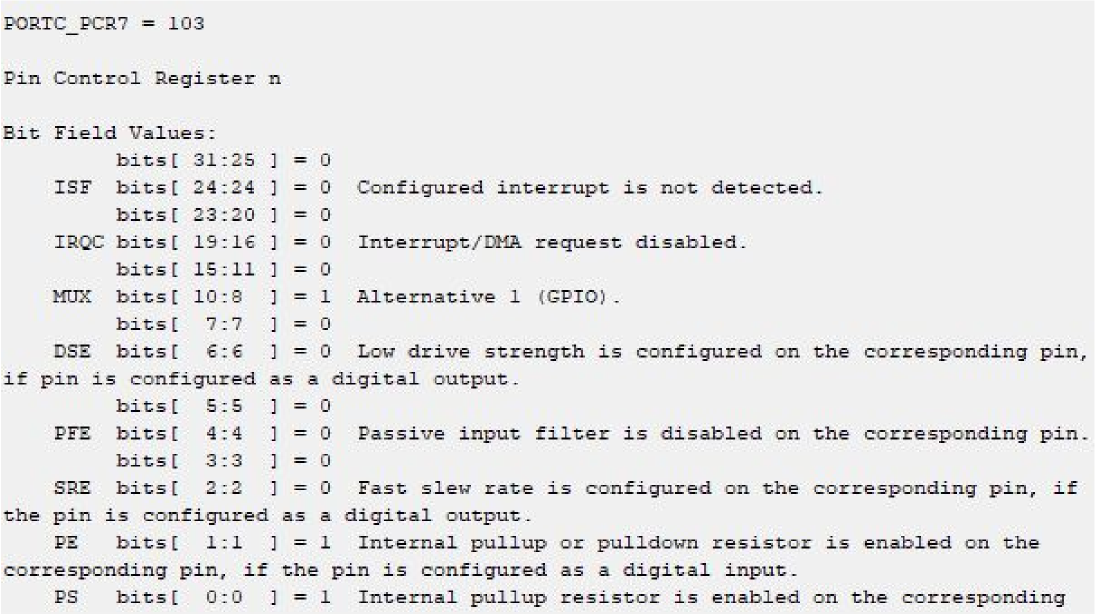


```
PORTC_PCR0 = 107

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 1  Slow slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 1  Internal pullup or pulldown resistor is enabled on the
corresponding pin, if the pin is configured as a digital input.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

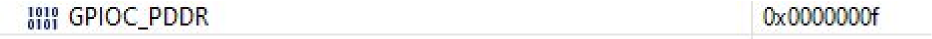
For PORTC_PCR3 to PORTC_CR7 the configuration of the registers is the next.



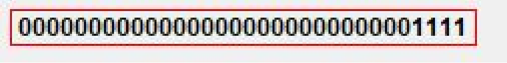
```
0000000 | 0 | 0000 | 0000 | 00000 | 001 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1
```



```
PORTC_PCR7 = 103

Pin Control Register n

Bit Field Values:
           bits[ 31:25 ] = 0
     ISF   bits[ 24:24 ] = 0  Configured interrupt is not detected.
           bits[ 23:20 ] = 0
     IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
           bits[ 15:11 ] = 0
     MUX   bits[ 10:8  ] = 1  Alternative 1 (GPIO).
           bits[  7:7  ] = 0
     DSE   bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
           bits[  5:5  ] = 0
     PFE   bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
           bits[  3:3  ] = 0
     SRE   bits[  2:2  ] = 0  Fast slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
     PE    bits[  1:1  ] = 1  Internal pullup or pulldown resistor is enabled on the
corresponding pin, if the pin is configured as a digital input.
     PS    bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
```

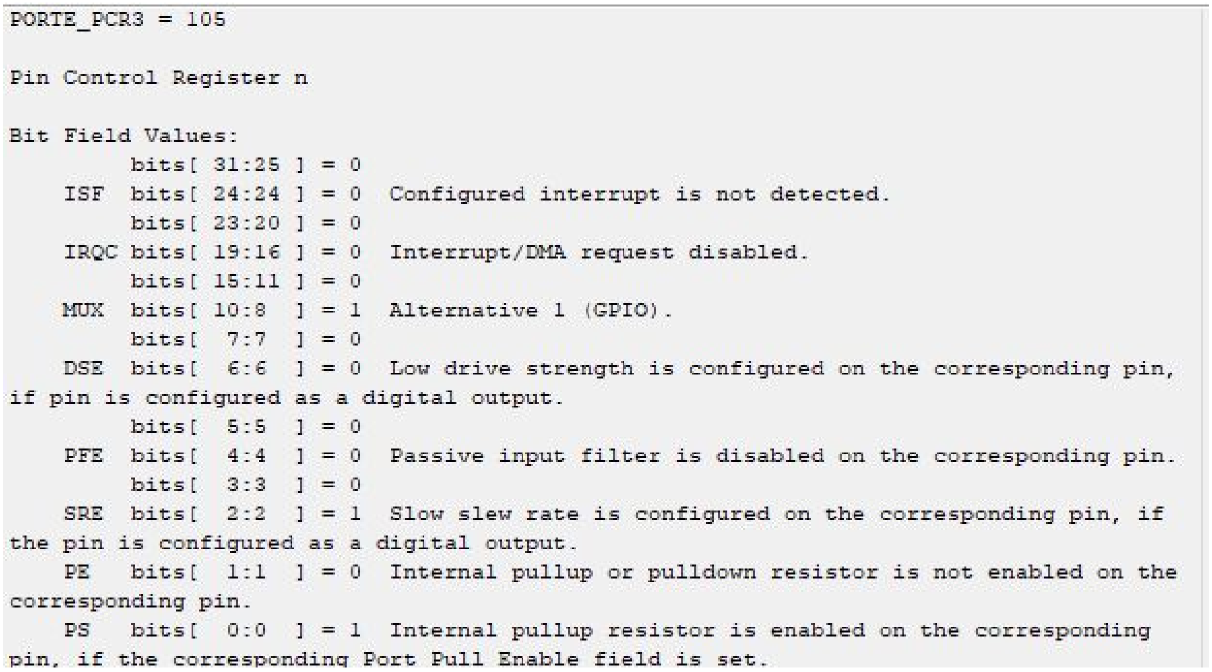


```
GPIOC_PDDR                                  0x0000000f
```

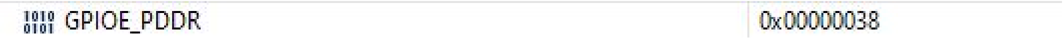


```
00000000000000000000000000001111
```

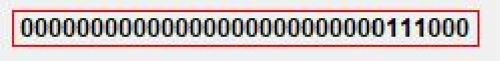
For PORTE_PCR3 to PORTE_PCR5 the configuration of the registers is the next.



```
PORTE_PCR3 = 105

Pin Control Register n

Bit Field Values:
           bits[ 31:25 ] = 0
     ISF   bits[ 24:24 ] = 0  Configured interrupt is not detected.
           bits[ 23:20 ] = 0
     IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
           bits[ 15:11 ] = 0
     MUX   bits[ 10:8  ] = 1  Alternative 1 (GPIO).
           bits[  7:7  ] = 0
     DSE   bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
           bits[  5:5  ] = 0
     PFE   bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
           bits[  3:3  ] = 0
     SRE   bits[  2:2  ] = 1  Slow slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
     PE    bits[  1:1  ] = 0  Internal pullup or pulldown resistor is not enabled on the
corresponding pin.
     PS    bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

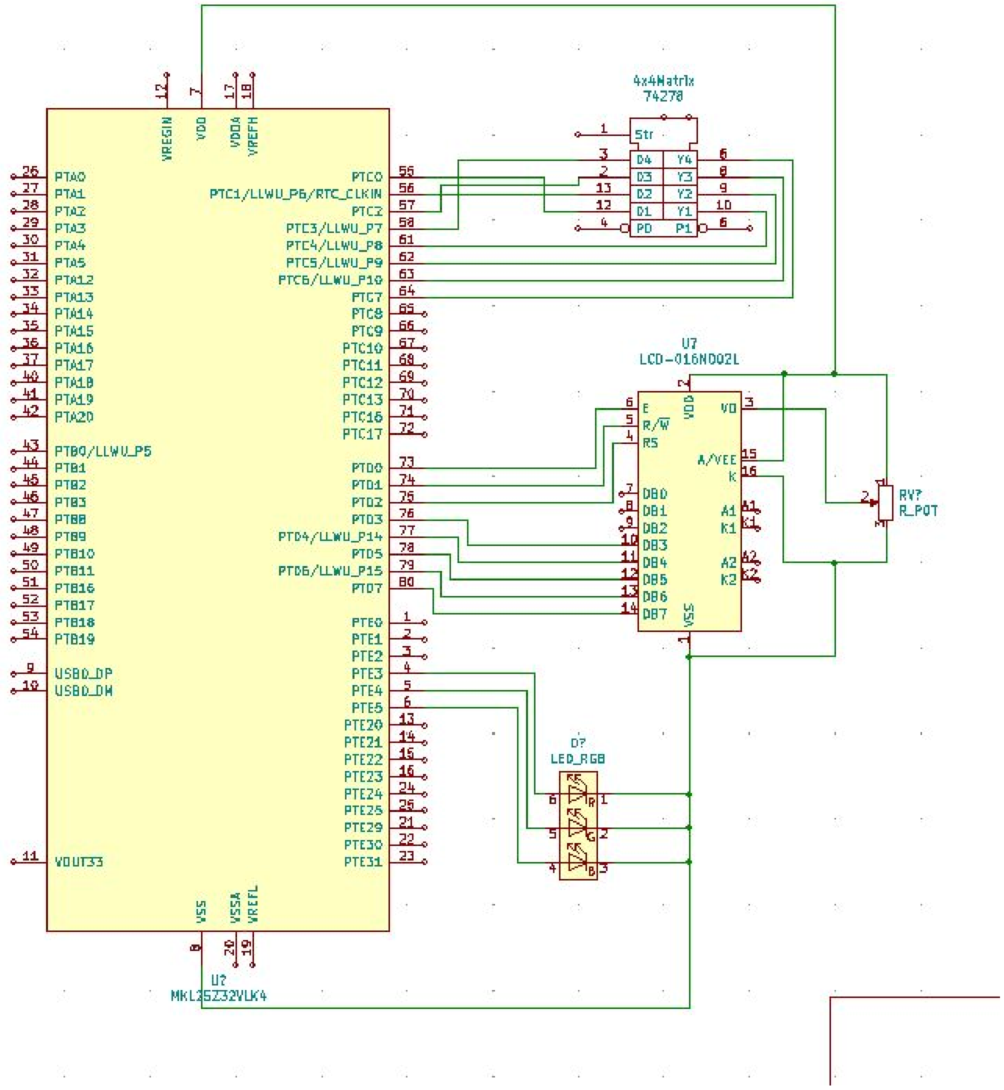


GPIOE_PDDR 0x00000038



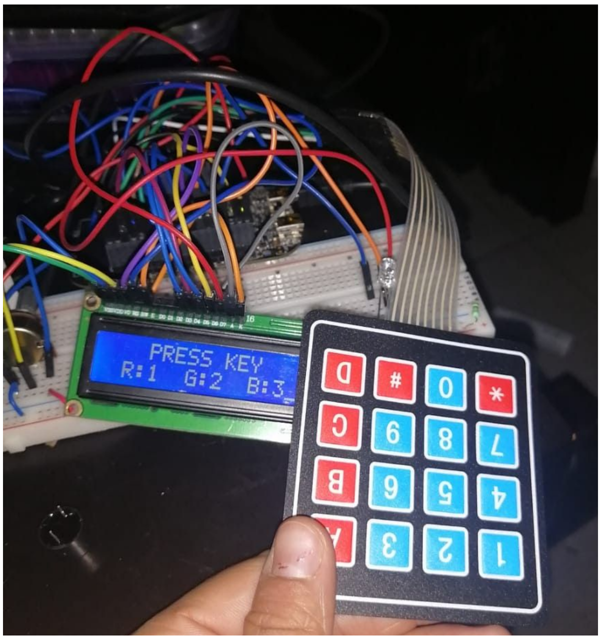
00000000000000000000000000111000

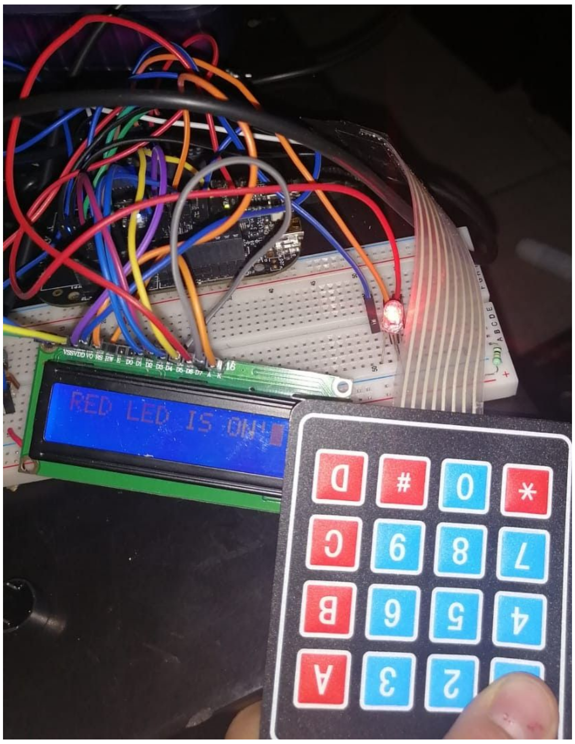
Schematic:

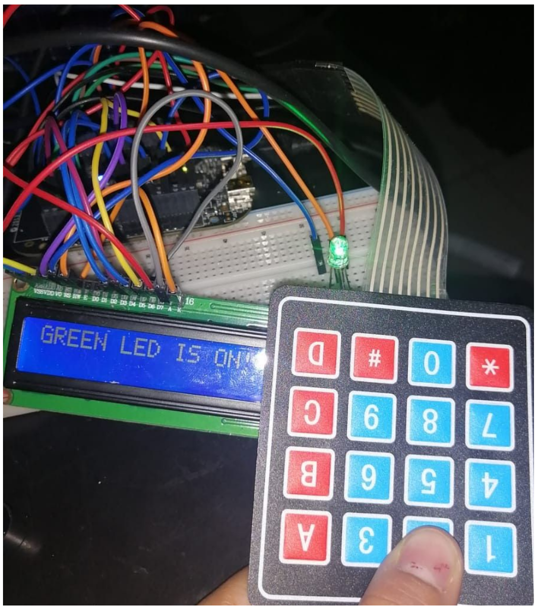


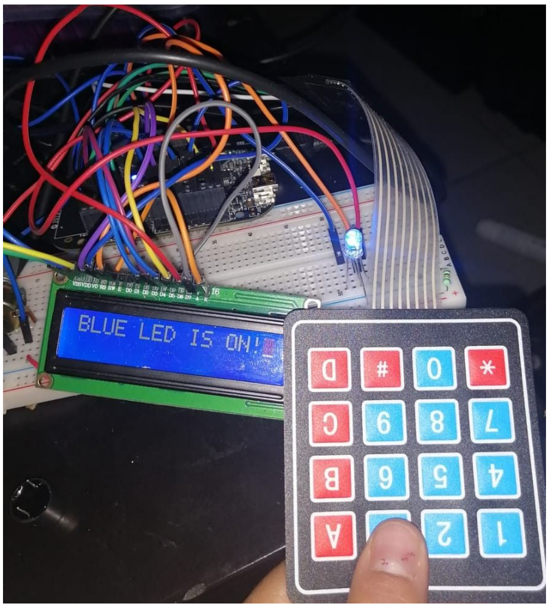
Video: https://youtu.be/3iGjBPr8FgA

Pictures of the circuit:


PRESS KEY
R:1 G:2 B:3


RED LED IS ON!

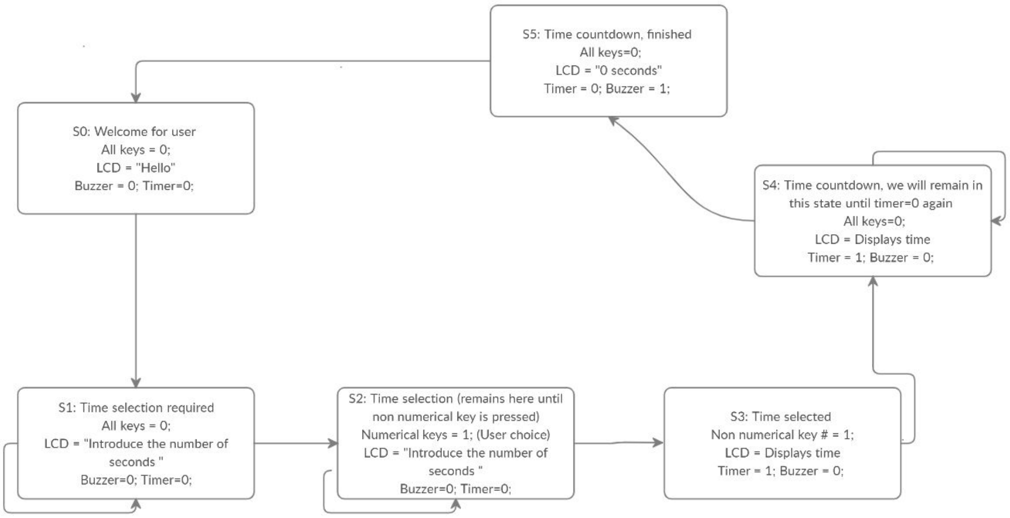

GREEN LED IS ON!


BLUE LED IS ON!

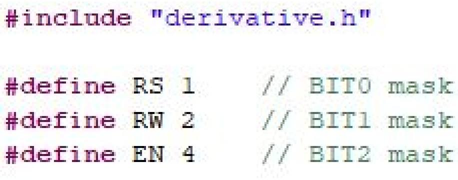
Part 2. Ascending Timer: The goal of the program is to implement a simple timer, as the ones used in sports or music. The code should proceed as follows:

1. **It should print a hello and stay there for 5 seconds (you can use simple delay for this), then a second message should appear in the first line asking for the user to introduce the number of seconds.**
2. **To make this interesting, the code should be able to accept at least two digits, so you need to introduce the get_key(function) into a while loop that only stops when you press another not numerical key (for instance the * or # key.)**
3. **The introduced value should set the TMP_MODULO register in one timer and start it, either immediately after the termination key was selected or by pressing the same key again.**
4. **Then the LCD should show a message Counting and showing the current count value.**
5. **Once the timer has elapsed, you can use a buzzer (if you have it) or a led to mark that the timer went to zero.**

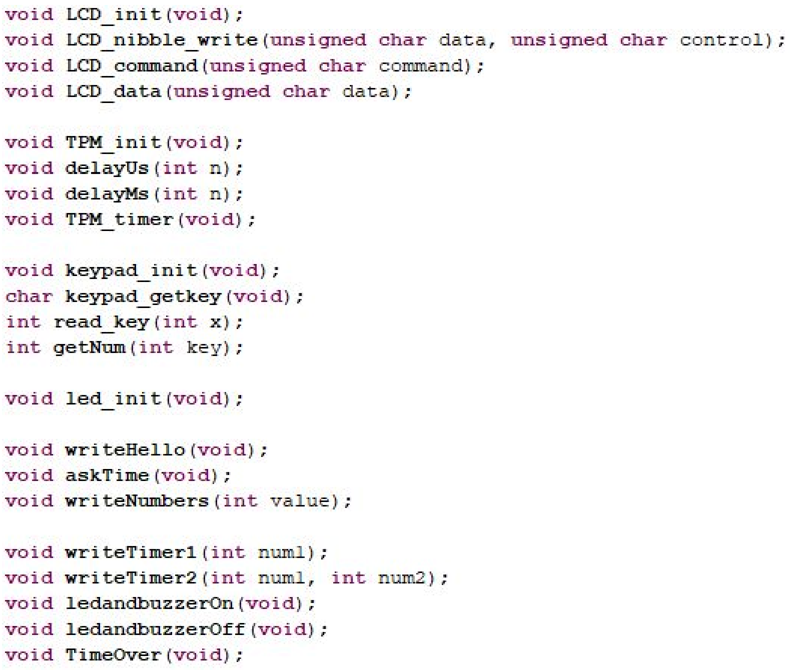
State Machine or Flow Diagram:


S5: Time countdown, finished
All keys=0;
LCD = "0 seconds"
Timer = 0; Buzzer = 1;
S0: Welcome for user
All keys = 0;
LCD = "Hello"
Buzzer = 0; Timer=0;
S4: Time countdown, we will remain in this state until timer=0 again
All keys=0;
LCD = Displays time
Timer = 1; Buzzer = 0;
S1: Time selection required
All keys = 0;
LCD = "Introduce the number of seconds "
Buzzer=0; Timer=0;
S2: Time selection (remains here until non numerical key is pressed)
Numerical keys = 1; (User choice)
LCD = "Introduce the number of seconds "
Buzzer=0; Timer=0;
S3: Time selected
Non numerical key # = 1;
LCD = Displays time
Timer = 1; Buzzer = 0;

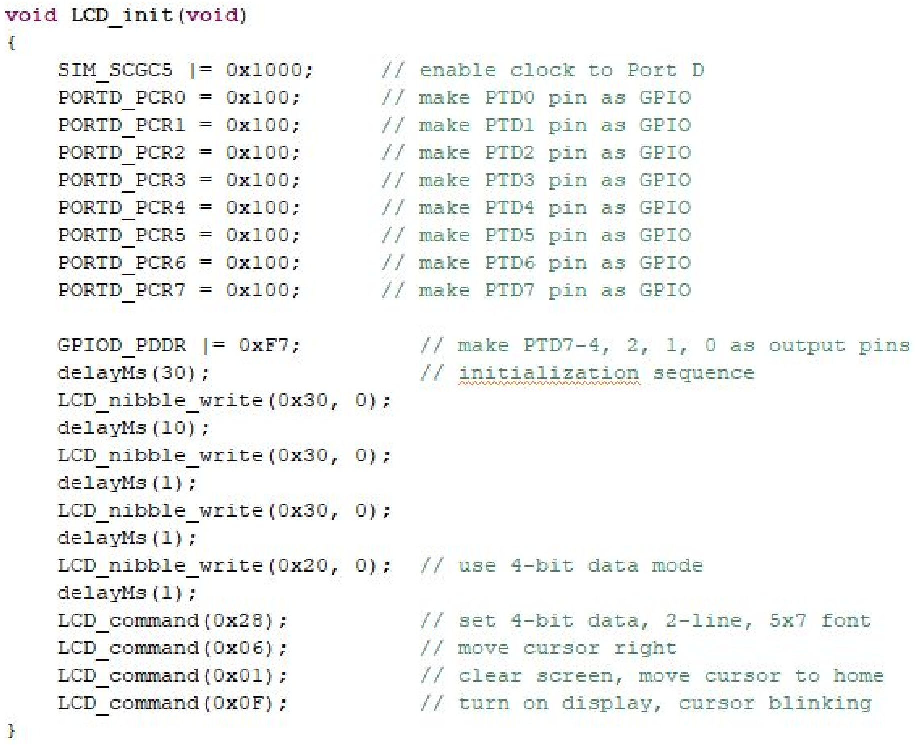
Code:



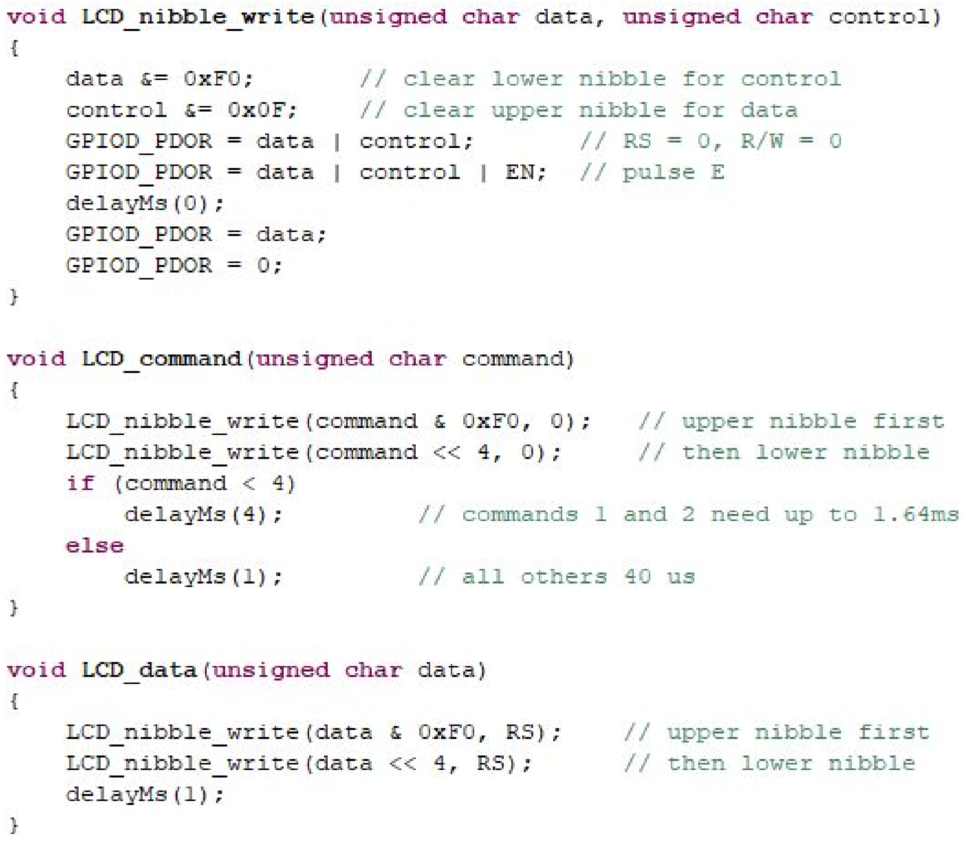
```
#include "derivative.h"

#define RS 1     // BIT0 mask
#define RW 2     // BIT1 mask
#define EN 4     // BIT2 mask
```



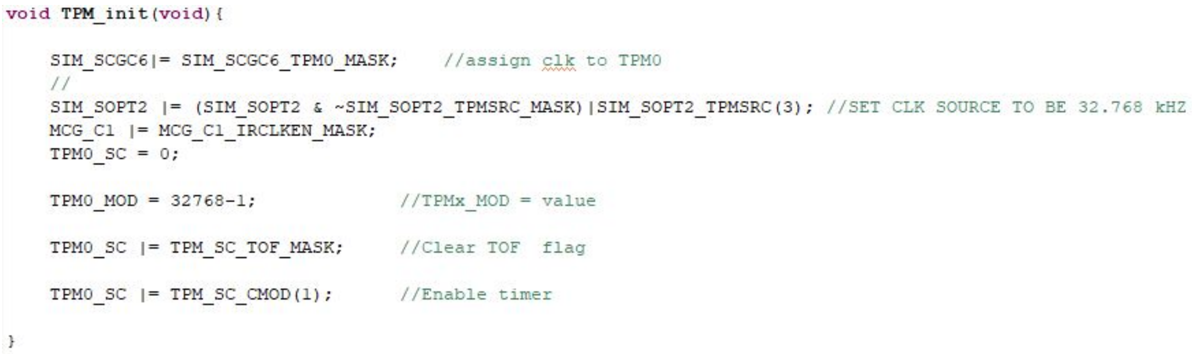
```
void LCD_init(void);
void LCD_nibble_write(unsigned char data, unsigned char control);
void LCD_command(unsigned char command);
void LCD_data(unsigned char data);

void TPM_init(void);
void delayUs(int n);
void delayMs(int n);
void TPM_timer(void);

void keypad_init(void);
char keypad_getkey(void);
int read_key(int x);
int getNum(int key);

void led_init(void);

void writeHello(void);
void askTime(void);
void writeNumbers(int value);

void writeTimer1(int num1);
void writeTimer2(int num1, int num2);
void ledandbuzzerOn(void);
void ledandbuzzerOff(void);
void TimeOver(void);
```



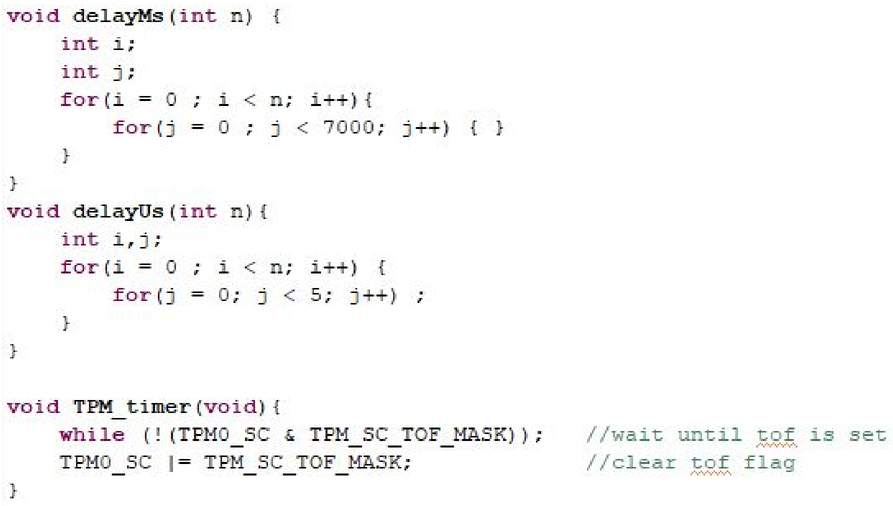
```c
void LCD_init(void)
{
    SIM_SCGC5 |= 0x1000;     // enable clock to Port D
    PORTD_PCR0 = 0x100;      // make PTD0 pin as GPIO
    PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
    PORTD_PCR2 = 0x100;      // make PTD2 pin as GPIO
    PORTD_PCR3 = 0x100;      // make PTD3 pin as GPIO
    PORTD_PCR4 = 0x100;      // make PTD4 pin as GPIO
    PORTD_PCR5 = 0x100;      // make PTD5 pin as GPIO
    PORTD_PCR6 = 0x100;      // make PTD6 pin as GPIO
    PORTD_PCR7 = 0x100;      // make PTD7 pin as GPIO

    GPIOD_PDDR |= 0xF7;         // make PTD7-4, 2, 1, 0 as output pins
    delayMs(30);                // initialization sequence
    LCD_nibble_write(0x30, 0);
    delayMs(10);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x20, 0);  // use 4-bit data mode
    delayMs(1);
    LCD_command(0x28);          // set 4-bit data, 2-line, 5x7 font
    LCD_command(0x06);          // move cursor right
    LCD_command(0x01);          // clear screen, move cursor to home
    LCD_command(0x0F);          // turn on display, cursor blinking
}
```



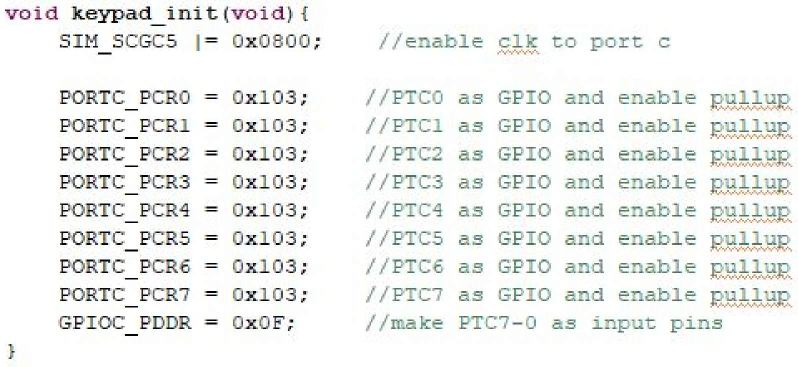
```c
void LCD_nibble_write(unsigned char data, unsigned char control)
{
    data &= 0xF0;       // clear lower nibble for control
    control &= 0x0F;    // clear upper nibble for data
    GPIOD_PDOR = data | control;       // RS = 0, R/W = 0
    GPIOD_PDOR = data | control | EN;  // pulse E
    delayMs(0);
    GPIOD_PDOR = data;
    GPIOD_PDOR = 0;
}

void LCD_command(unsigned char command)
{
    LCD_nibble_write(command & 0xF0, 0);   // upper nibble first
    LCD_nibble_write(command << 4, 0);     // then lower nibble
    if (command < 4)
        delayMs(4);         // commands 1 and 2 need up to 1.64ms
    else
        delayMs(1);         // all others 40 us
}

void LCD_data(unsigned char data)
{
    LCD_nibble_write(data & 0xF0, RS);    // upper nibble first
    LCD_nibble_write(data << 4, RS);      // then lower nibble
    delayMs(1);
}
```



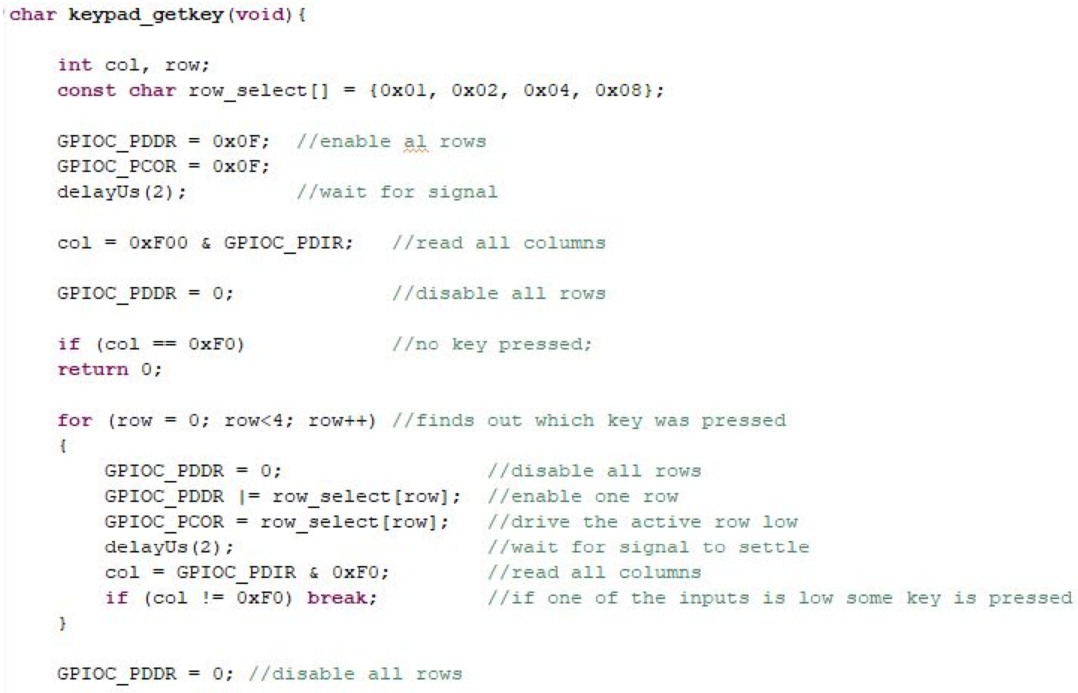
```
void TPM_init(void){

    SIM_SCGC6|= SIM_SCGC6_TPM0_MASK;    //assign clk to TPM0
    //
    SIM_SOPT2 |= (SIM_SOPT2 & ~SIM_SOPT2_TPMSRC_MASK)|SIM_SOPT2_TPMSRC(3); //SET CLK SOURCE TO BE 32.768 kHZ
    MCG_C1 |= MCG_C1_IRCLKEN_MASK;
    TPM0_SC = 0;

    TPM0_MOD = 32768-1;             //TPMx_MOD = value

    TPM0_SC |= TPM_SC_TOF_MASK;     //Clear TOF  flag

    TPM0_SC |= TPM_SC_CMOD(1);      //Enable timer

}
```



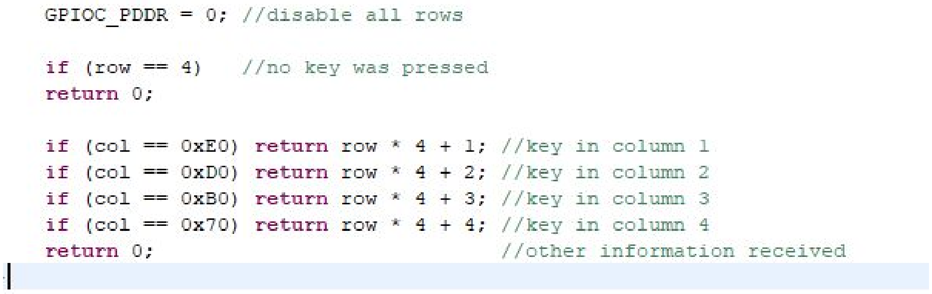
```
void delayMs(int n) {
    int i;
    int j;
    for(i = 0 ; i < n; i++){
        for(j = 0 ; j < 7000; j++) { }
    }
}
void delayUs(int n){
    int i,j;
    for(i = 0 ; i < n; i++) {
        for(j = 0; j < 5; j++) ;
    }
}

void TPM_timer(void){
    while (!(TPM0_SC & TPM_SC_TOF_MASK));   //wait until tof is set
    TPM0_SC |= TPM_SC_TOF_MASK;             //clear tof flag
}
```



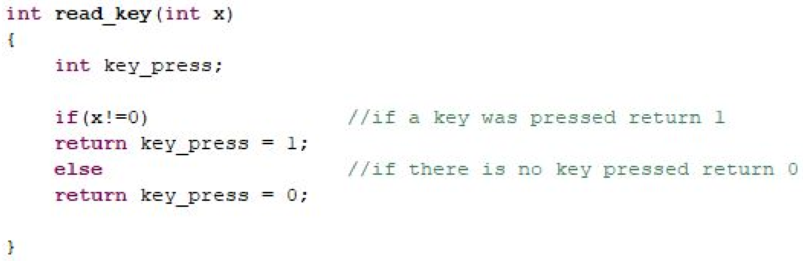
```
void keypad_init(void){
    SIM_SCGC5 |= 0x0800;    //enable clk to port c

    PORTC_PCR0 = 0x103;    //PTC0 as GPIO and enable pullup
    PORTC_PCR1 = 0x103;    //PTC1 as GPIO and enable pullup
    PORTC_PCR2 = 0x103;    //PTC2 as GPIO and enable pullup
    PORTC_PCR3 = 0x103;    //PTC3 as GPIO and enable pullup
    PORTC_PCR4 = 0x103;    //PTC4 as GPIO and enable pullup
    PORTC_PCR5 = 0x103;    //PTC5 as GPIO and enable pullup
    PORTC_PCR6 = 0x103;    //PTC6 as GPIO and enable pullup
    PORTC_PCR7 = 0x103;    //PTC7 as GPIO and enable pullup
    GPIOC_PDDR = 0x0F;     //make PTC7-0 as input pins
}
```



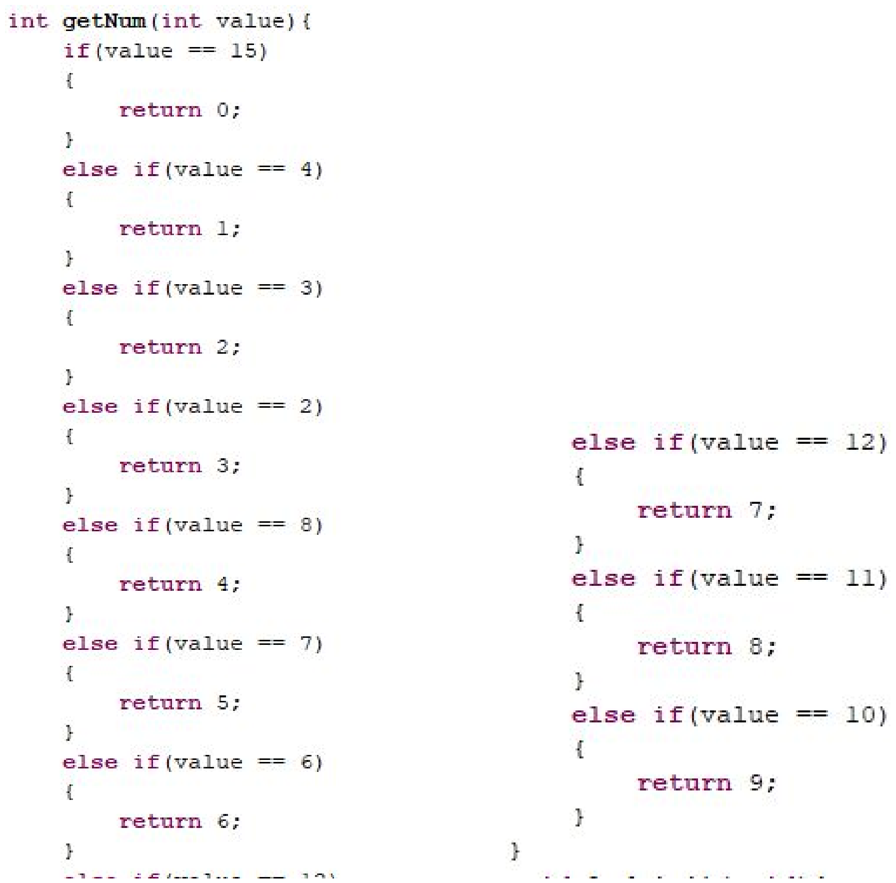
```
char keypad_getkey(void){

    int col, row;
    const char row_select[] = {0x01, 0x02, 0x04, 0x08};

    GPIOC_PDDR = 0x0F;  //enable al rows
    GPIOC_PCOR = 0x0F;
    delayUs(2);         //wait for signal

    col = 0xF00 & GPIOC_PDIR;   //read all columns

    GPIOC_PDDR = 0;             //disable all rows

    if (col == 0xF0)            //no key pressed;
    return 0;

    for (row = 0; row<4; row++) //finds out which key was pressed
    {
        GPIOC_PDDR = 0;                 //disable all rows
        GPIOC_PDDR |= row_select[row];  //enable one row
        GPIOC_PCOR = row_select[row];   //drive the active row low
        delayUs(2);                     //wait for signal to settle
        col = GPIOC_PDIR & 0xF0;        //read all columns
        if (col != 0xF0) break;         //if one of the inputs is low some key is pressed
    }

    GPIOC_PDDR = 0; //disable all rows
```



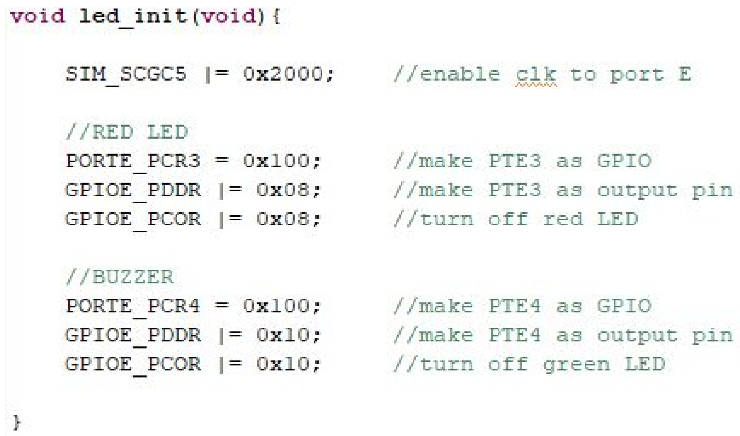
```
    GPIOC_PDDR = 0; //disable all rows

    if (row == 4)   //no key was pressed
    return 0;

    if (col == 0xE0) return row * 4 + 1; //key in column 1
    if (col == 0xD0) return row * 4 + 2; //key in column 2
    if (col == 0xB0) return row * 4 + 3; //key in column 3
    if (col == 0x70) return row * 4 + 4; //key in column 4
    return 0;                            //other information received
```



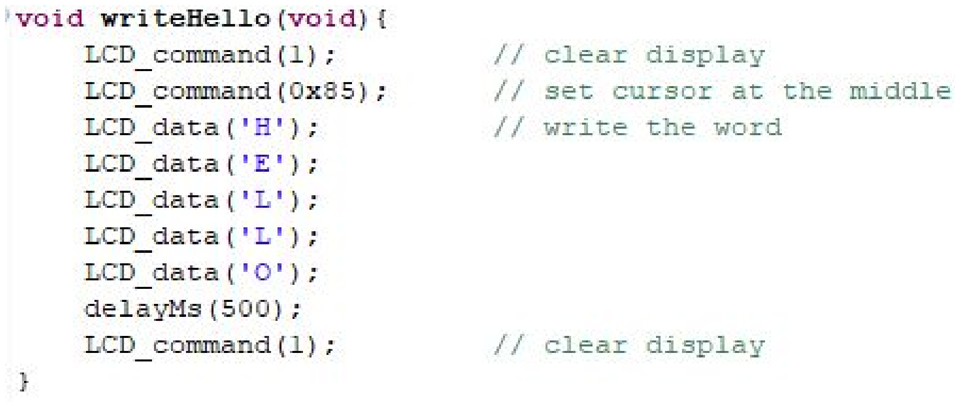
```
int read_key(int x)
{
    int key_press;

    if(x!=0)                //if a key was pressed return 1
    return key_press = 1;
    else                    //if there is no key pressed return 0
    return key_press = 0;

}
```



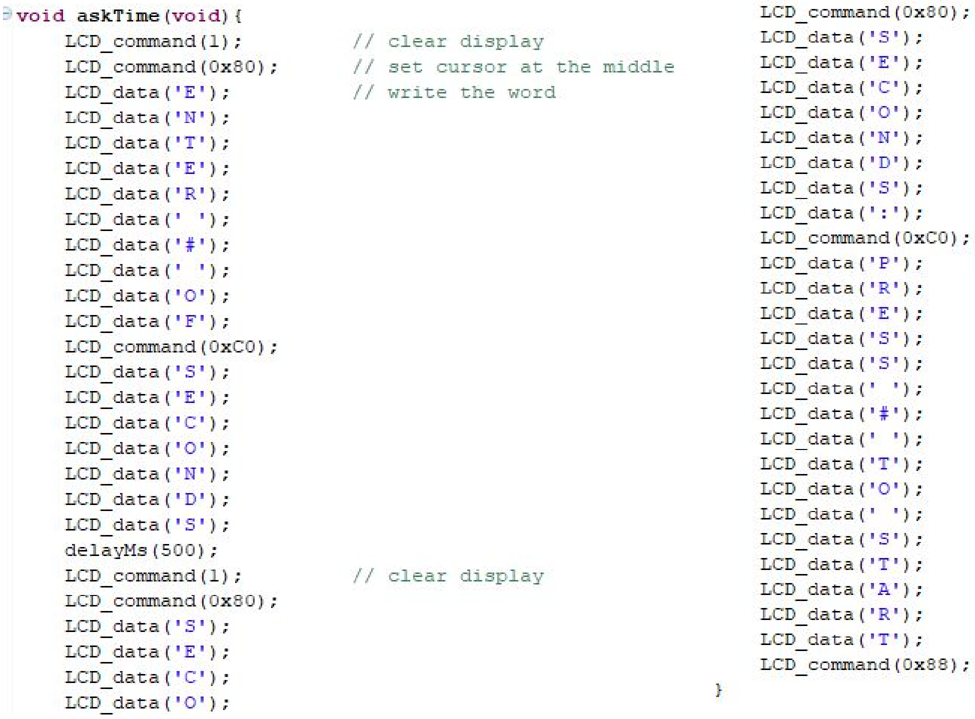
```
int getNum(int value){
    if(value == 15)
    {
        return 0;
    }
    else if(value == 4)
    {
        return 1;
    }
    else if(value == 3)
    {
        return 2;
    }
    else if(value == 2)
    {
        return 3;
    }
    else if(value == 8)
    {
        return 4;
    }
    else if(value == 7)
    {
        return 5;
    }
    else if(value == 6)
    {
        return 6;
    }
    else if(value == 12)
    {
        return 7;
    }
    else if(value == 11)
    {
        return 8;
    }
    else if(value == 10)
    {
        return 9;
    }
}
```



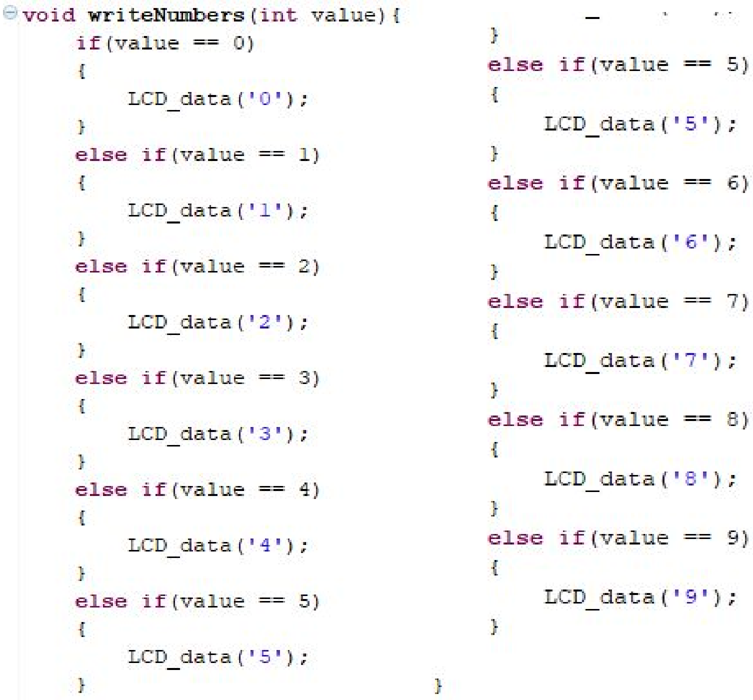
```
void led_init(void){

    SIM_SCGC5 |= 0x2000;     //enable clk to port E

    //RED LED
    PORTE_PCR3 = 0x100;      //make PTE3 as GPIO
    GPIOE_PDDR |= 0x08;      //make PTE3 as output pin
    GPIOE_PCOR |= 0x08;      //turn off red LED

    //BUZZER
    PORTE_PCR4 = 0x100;      //make PTE4 as GPIO
    GPIOE_PDDR |= 0x10;      //make PTE4 as output pin
    GPIOE_PCOR |= 0x10;      //turn off green LED

}
```



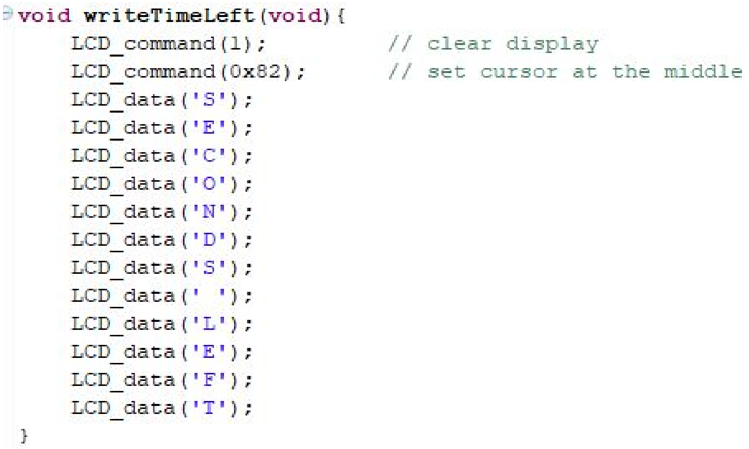
```
void writeHello(void){
    LCD_command(1);           // clear display
    LCD_command(0x85);        // set cursor at the middle
    LCD_data('H');            // write the word
    LCD_data('E');
    LCD_data('L');
    LCD_data('L');
    LCD_data('O');
    delayMs(500);
    LCD_command(1);           // clear display
}
```



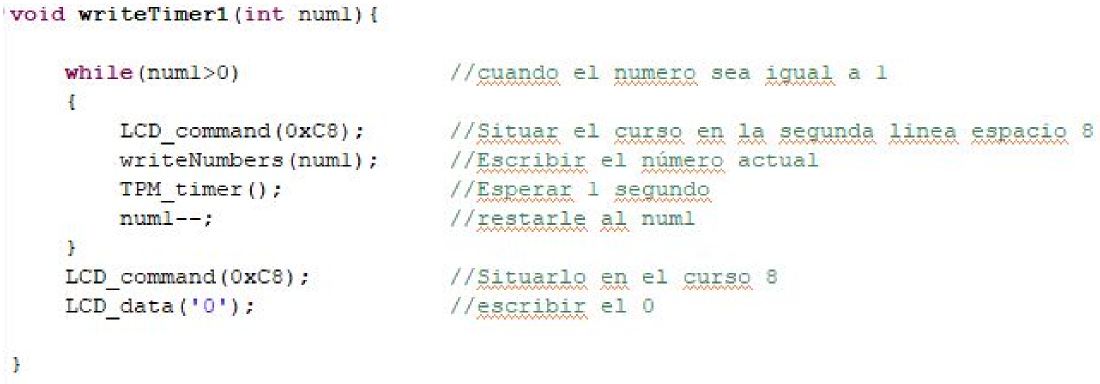
```
void askTime(void){
    LCD_command(1);         // clear display
    LCD_command(0x80);      // set cursor at the middle
    LCD_data('E');          // write the word
    LCD_data('N');
    LCD_data('T');
    LCD_data('E');
    LCD_data('R');
    LCD_data(' ');
    LCD_data('#');
    LCD_data(' ');
    LCD_data('O');
    LCD_data('F');
    LCD_command(0xC0);
    LCD_data('S');
    LCD_data('E');
    LCD_data('C');
    LCD_data('O');
    LCD_data('N');
    LCD_data('D');
    LCD_data('S');
    delayMs(500);
    LCD_command(1);         // clear display
    LCD_command(0x80);
    LCD_data('S');
    LCD_data('E');
    LCD_data('C');
    LCD_data('O');
    LCD_command(0x80);
    LCD_data('S');
    LCD_data('E');
    LCD_data('C');
    LCD_data('O');
    LCD_data('N');
    LCD_data('D');
    LCD_data('S');
    LCD_data(':');
    LCD_command(0xC0);
    LCD_data('P');
    LCD_data('R');
    LCD_data('E');
    LCD_data('S');
    LCD_data('S');
    LCD_data(' ');
    LCD_data('#');
    LCD_data(' ');
    LCD_data('T');
    LCD_data('O');
    LCD_data(' ');
    LCD_data('S');
    LCD_data('T');
    LCD_data('A');
    LCD_data('R');
    LCD_data('T');
    LCD_command(0x88);
}
```



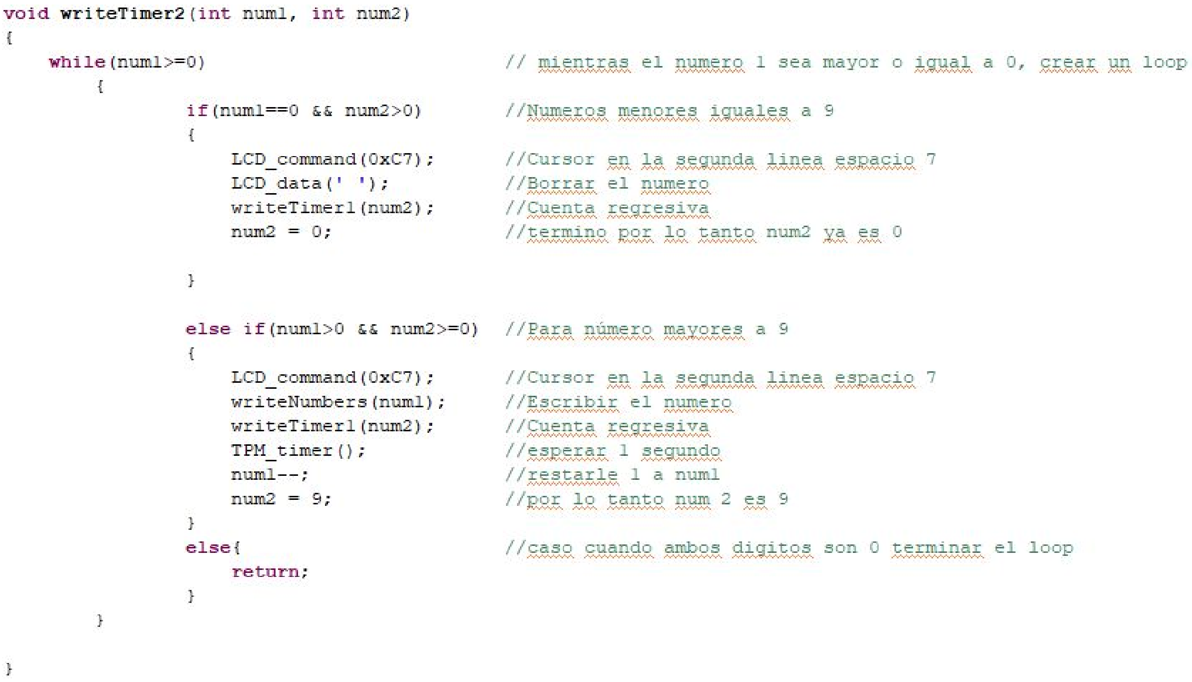
```
void writeNumbers(int value){
    if(value == 0)
    {
        LCD_data('0');
    }
    else if(value == 1)
    {
        LCD_data('1');
    }
    else if(value == 2)
    {
        LCD_data('2');
    }
    else if(value == 3)
    {
        LCD_data('3');
    }
    else if(value == 4)
    {
        LCD_data('4');
    }
    else if(value == 5)
    {
        LCD_data('5');
    }
}
    }
    else if(value == 5)
    {
        LCD_data('5');
    }
    else if(value == 6)
    {
        LCD_data('6');
    }
    else if(value == 7)
    {
        LCD_data('7');
    }
    else if(value == 8)
    {
        LCD_data('8');
    }
    else if(value == 9)
    {
        LCD_data('9');
    }
```



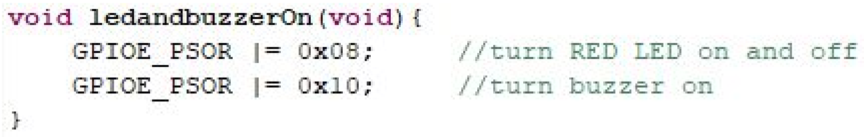
```
void writeTimeLeft(void){
    LCD_command(1);         // clear display
    LCD_command(0x82);      // set cursor at the middle
    LCD_data('S');
    LCD_data('E');
    LCD_data('C');
    LCD_data('O');
    LCD_data('N');
    LCD_data('D');
    LCD_data('S');
    LCD_data(' ');
    LCD_data('L');
    LCD_data('E');
    LCD_data('F');
    LCD_data('T');
}
```



```
void writeTimer1(int num1){

    while(num1>0)                   //cuando el numero sea igual a 1
    {
        LCD_command(0xC8);          //Situar el curso en la segunda linea espacio 8
        writeNumbers(num1);         //Escribir el número actual
        TPM_timer();                //Esperar 1 segundo
        num1--;                     //restarle al num1
    }
    LCD_command(0xC8);              //Situarlo en el curso 8
    LCD_data('0');                  //escribir el 0

}
```

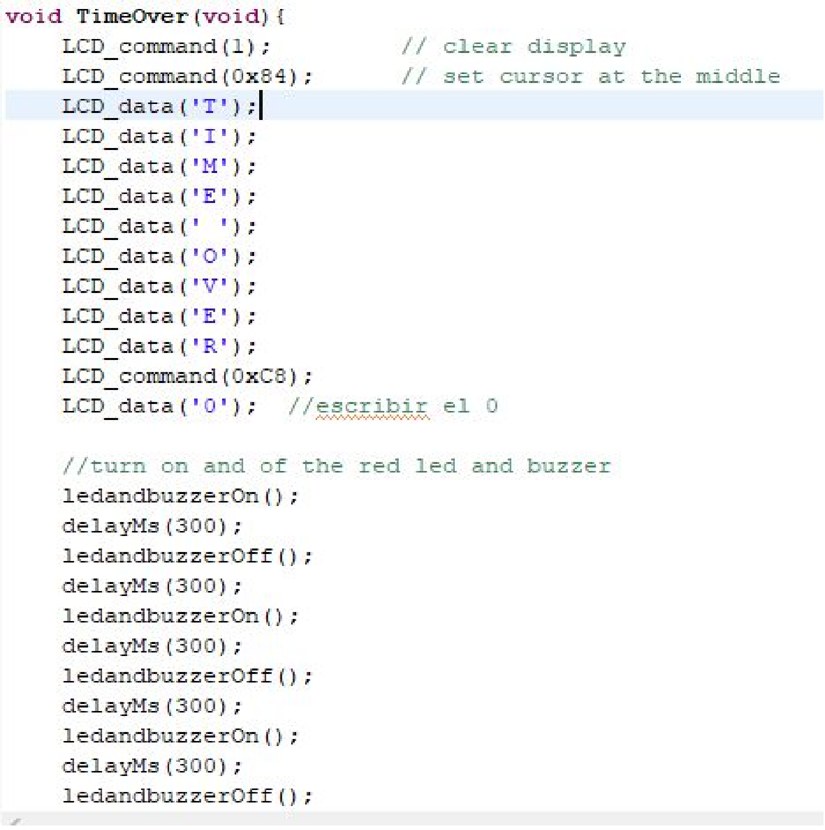


```
void writeTimer2(int num1, int num2)
{
    while(num1>=0)                                  // mientras el numero 1 sea mayor o igual a 0, crear un loop
        {
                if(num1==0 && num2>0)               //Numeros menores iguales a 9
                {
                    LCD_command(0xC7);              //Cursor en la segunda linea espacio 7
                    LCD_data(' ');                  //Borrar el numero
                    writeTimer1(num2);              //Cuenta regresiva
                    num2 = 0;                       //termino por lo tanto num2 ya es 0

                }

                else if(num1>0 && num2>=0)          //Para número mayores a 9
                {
                    LCD_command(0xC7);              //Cursor en la segunda linea espacio 7
                    writeNumbers(num1);             //Escribir el numero
                    writeTimer1(num2);              //Cuenta regresiva
                    TPM_timer();                    //esperar 1 segundo
                    num1--;                         //restarle 1 a num1
                    num2 = 9;                       //por lo tanto num 2 es 9
                }
                else{                               //caso cuando ambos digitos son 0 terminar el loop
                    return;
                }
        }

}
```



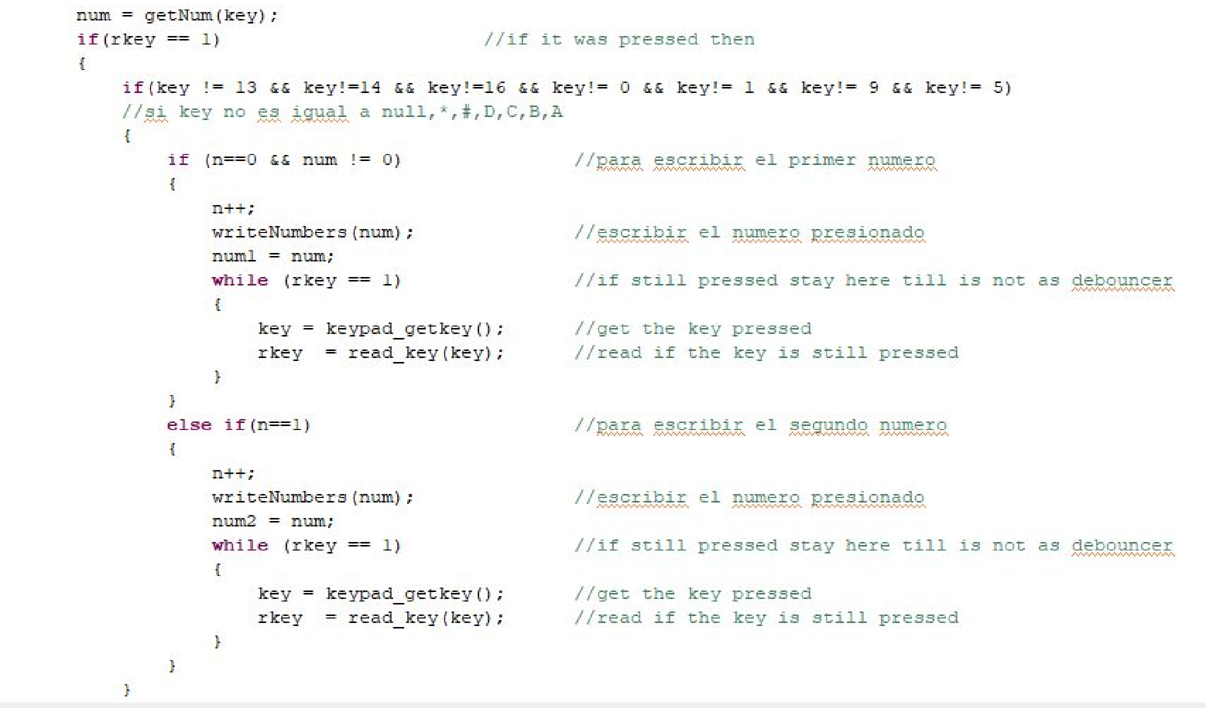
```
void ledandbuzzerOn(void){
    GPIOE_PSOR |= 0x08;     //turn RED LED on and off
    GPIOE_PSOR |= 0x10;     //turn buzzer on
}
```

```
void ledandbuzzerOff(void){
    GPIOE_PCOR |= 0x08;     //turn RED LED on and off
    GPIOE_PCOR |= 0x10;     //turn buzzer on
}
```

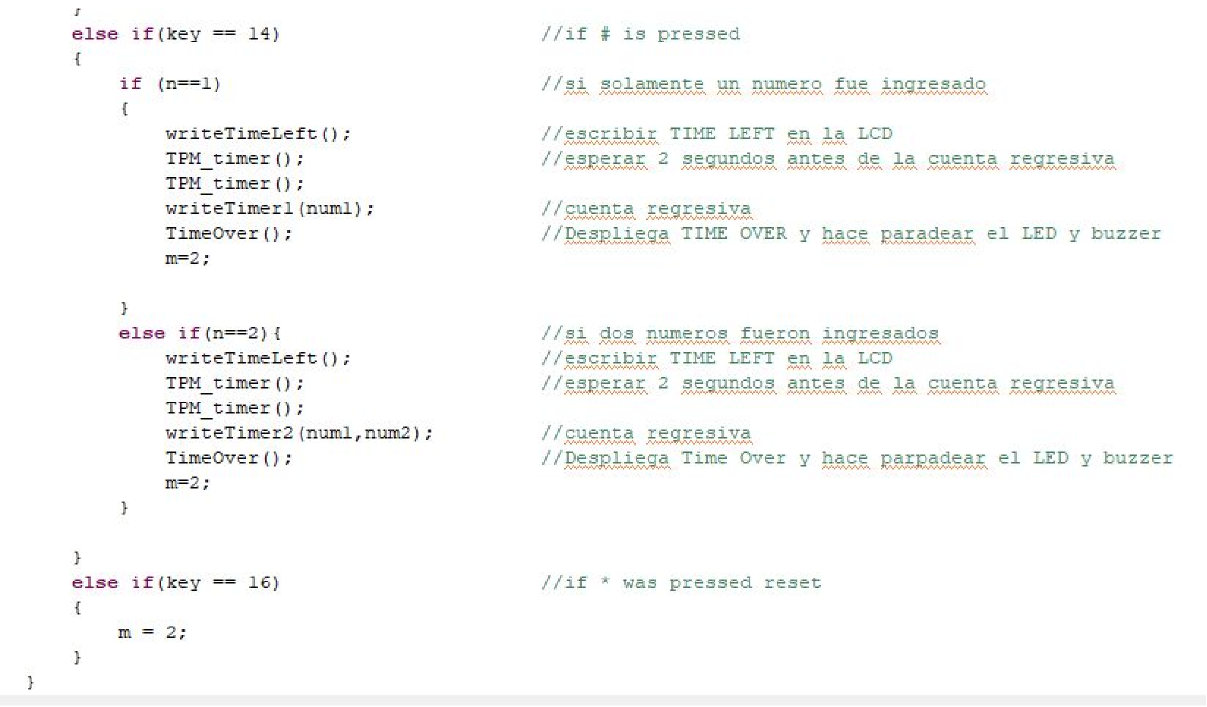


```
void TimeOver(void){
    LCD_command(1);         // clear display
    LCD_command(0x84);      // set cursor at the middle
    LCD_data('T');
    LCD_data('I');
    LCD_data('M');
    LCD_data('E');
    LCD_data(' ');
    LCD_data('O');
    LCD_data('V');
    LCD_data('E');
    LCD_data('R');
    LCD_command(0xC8);
    LCD_data('0');  //escribir el 0

    //turn on and of the red led and buzzer
    ledandbuzzerOn();
    delayMs(300);
    ledandbuzzerOff();
    delayMs(300);
    ledandbuzzerOn();
    delayMs(300);
    ledandbuzzerOff();
    delayMs(300);
    ledandbuzzerOn();
    delayMs(300);
    ledandbuzzerOff();
```

```
int main(void)
{
    unsigned char key;
    int rkey;
    int m;
    int n;
    int num;
    int num1;
    int num2;
    LCD_init();
    keypad_init();
    led_init();
    TPM_init();

    for(;;)
    {
        m = 1;
        n = 0;                                  //números escritos
        writeHello();                           //escribir Hello
        askTime();                              //ask for time
        while(m == 1){
            key = keypad_getkey();              //get which key was pressed
            rkey  = read_key(key);              //read if the key was pressed
            num = getNum(key);
```

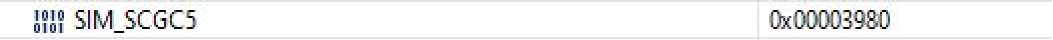


```
num = getNum(key);
if(rkey == 1)                           //if it was pressed then
{
    if(key != 13 && key!=14 && key!=16 && key!= 0 && key!= 1 && key!= 9 && key!= 5)
    //si key no es igual a null,*,#,D,C,B,A
    {
        if (n==0 && num != 0)                   //para escribir el primer numero
        {
            n++;
            writeNumbers(num);                  //escribir el numero presionado
            num1 = num;
            while (rkey == 1)                   //if still pressed stay here till is not as debouncer
            {
                key = keypad_getkey();          //get the key pressed
                rkey  = read_key(key);          //read if the key is still pressed
            }
        }
        else if(n==1)                           //para escribir el segundo numero
        {
            n++;
            writeNumbers(num);                  //escribir el numero presionado
            num2 = num;
            while (rkey == 1)                   //if still pressed stay here till is not as debouncer
            {
                key = keypad_getkey();          //get the key pressed
                rkey  = read_key(key);          //read if the key is still pressed
            }
        }
    }
```

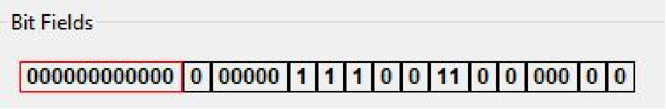


```
    }
    else if(key == 14)                           //if # is pressed
    {
        if (n==1)                                //si solamente un numero fue ingresado
        {
            writeTimeLeft();                     //escribir TIME LEFT en la LCD
            TPM_timer();                         //esperar 2 segundos antes de la cuenta regresiva
            TPM_timer();
            writeTimer1(num1);                   //cuenta regresiva
            TimeOver();                          //Despliega TIME OVER y hace paradear el LED y buzzer
            m=2;

        }
        else if(n==2){                           //si dos numeros fueron ingresados
            writeTimeLeft();                     //escribir TIME LEFT en la LCD
            TPM_timer();                         //esperar 2 segundos antes de la cuenta regresiva
            TPM_timer();
            writeTimer2(num1,num2);              //cuenta regresiva
            TimeOver();                          //Despliega Time Over y hace parpadear el LED y buzzer
            m=2;
        }

    }
    else if(key == 16)                           //if * was pressed reset
    {
        m = 2;
    }
}
```

Registers:

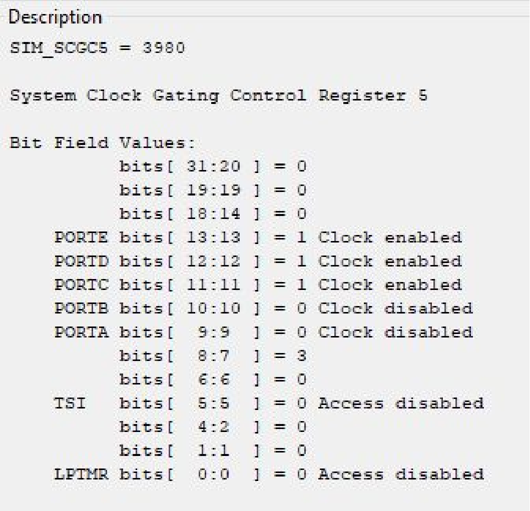


SIM_SCGC5 0x00003980



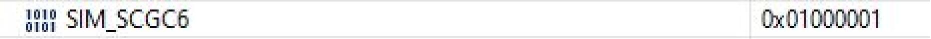
Bit Fields

```
000000000000 | 0 | 00000 | 1 | 1 | 1 | 0 | 0 | 11 | 0 | 0 | 000 | 0 | 0
```

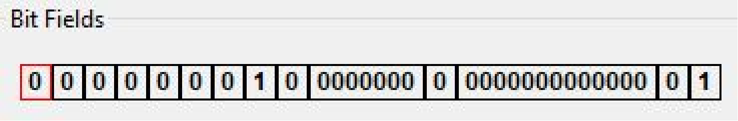


Description

```
SIM_SCGC5 = 3980

System Clock Gating Control Register 5

Bit Field Values:
          bits[ 31:20 ] = 0
          bits[ 19:19 ] = 0
          bits[ 18:14 ] = 0
    PORTE bits[ 13:13 ] = 1 Clock enabled
    PORTD bits[ 12:12 ] = 1 Clock enabled
    PORTC bits[ 11:11 ] = 1 Clock enabled
    PORTB bits[ 10:10 ] = 0 Clock disabled
    PORTA bits[  9:9  ] = 0 Clock disabled
          bits[  8:7  ] = 3
          bits[  6:6  ] = 0
    TSI   bits[  5:5  ] = 0 Access disabled
          bits[  4:2  ] = 0
          bits[  1:1  ] = 0
    LPTMR bits[  0:0  ] = 0 Access disabled
```

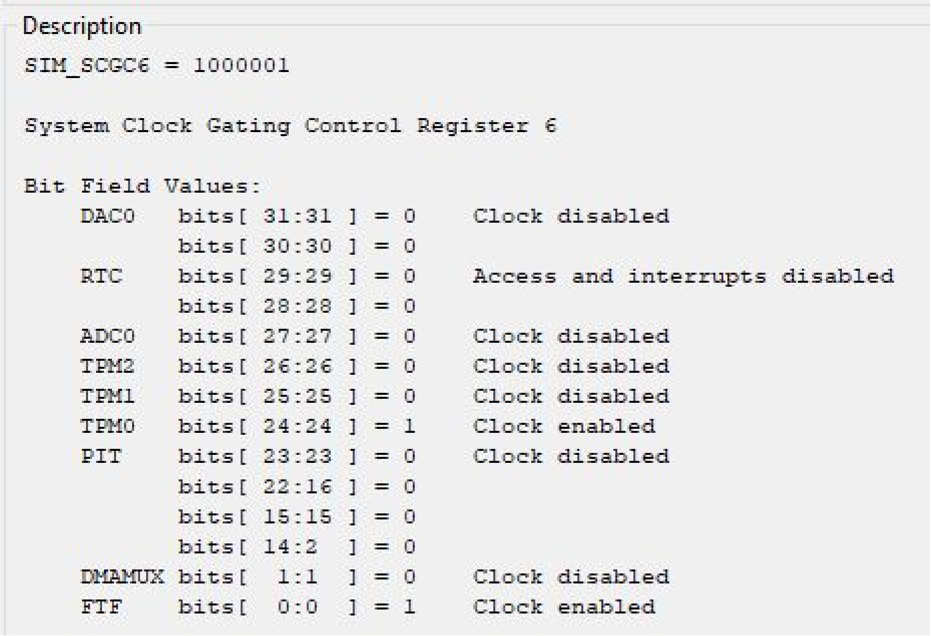


SIM_SCGC6 0x01000001



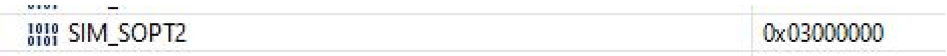
Bit Fields

```
0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0000000 | 0 | 0000000000000 | 0 | 1
```

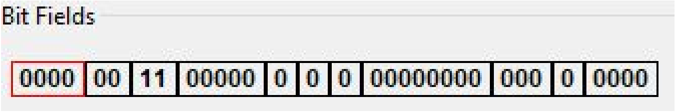


Description

```
SIM_SCGC6 = 1000001

System Clock Gating Control Register 6

Bit Field Values:
    DAC0   bits[ 31:31 ] = 0    Clock disabled
           bits[ 30:30 ] = 0
    RTC    bits[ 29:29 ] = 0    Access and interrupts disabled
           bits[ 28:28 ] = 0
    ADC0   bits[ 27:27 ] = 0    Clock disabled
    TPM2   bits[ 26:26 ] = 0    Clock disabled
    TPM1   bits[ 25:25 ] = 0    Clock disabled
    TPM0   bits[ 24:24 ] = 1    Clock enabled
    PIT    bits[ 23:23 ] = 0    Clock disabled
           bits[ 22:16 ] = 0
           bits[ 15:15 ] = 0
           bits[ 14:2  ] = 0
    DMAMUX bits[  1:1  ] = 0    Clock disabled
    FTF    bits[  0:0  ] = 1    Clock enabled
```

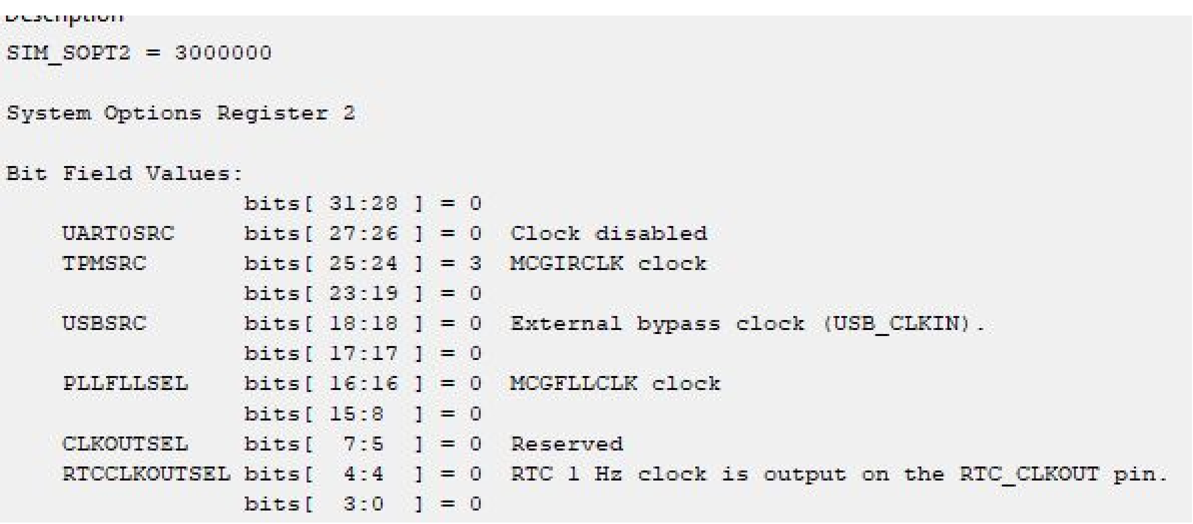


SIM_SOPT2 0x03000000



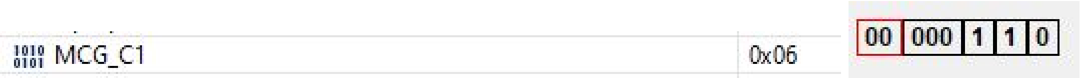
Bit Fields

0000	00	11	00000	0	0	0	00000000	000	0	0000



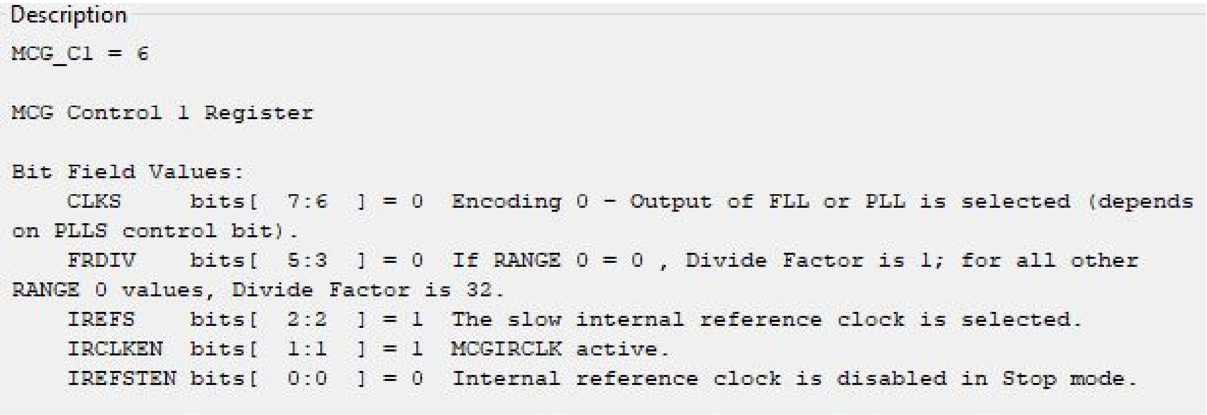
Description

```
SIM_SOPT2 = 3000000

System Options Register 2

Bit Field Values:
                 bits[ 31:28 ] = 0
    UART0SRC     bits[ 27:26 ] = 0  Clock disabled
    TPMSRC       bits[ 25:24 ] = 3  MCGIRCLK clock
                 bits[ 23:19 ] = 0
    USBSRC       bits[ 18:18 ] = 0  External bypass clock (USB_CLKIN).
                 bits[ 17:17 ] = 0
    PLLFLLSEL    bits[ 16:16 ] = 0  MCGFLLCLK clock
                 bits[ 15:8  ] = 0
    CLKOUTSEL    bits[  7:5  ] = 0  Reserved
    RTCCLKOUTSEL bits[  4:4  ] = 0  RTC 1 Hz clock is output on the RTC_CLKOUT pin.
                 bits[  3:0  ] = 0
```



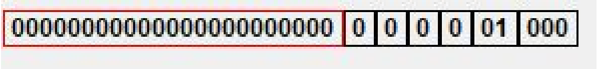
MCG_C1 0x06

00	000	1	1	0

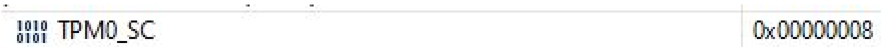


Description

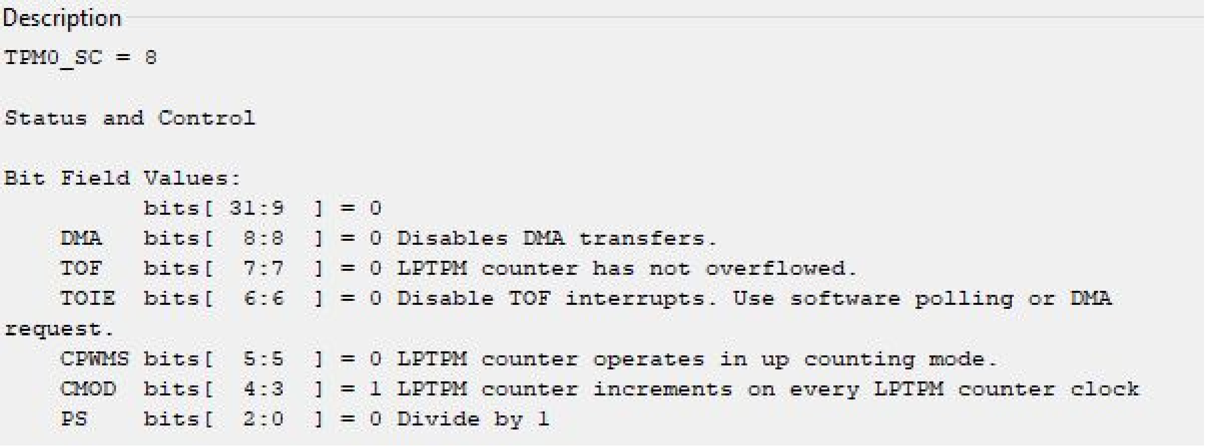
```
MCG_C1 = 6

MCG Control 1 Register

Bit Field Values:
    CLKS     bits[  7:6  ] = 0  Encoding 0 - Output of FLL or PLL is selected (depends
on PLLS control bit).
    FRDIV    bits[  5:3  ] = 0  If RANGE 0 = 0 , Divide Factor is 1; for all other
RANGE 0 values, Divide Factor is 32.
    IREFS    bits[  2:2  ] = 1  The slow internal reference clock is selected.
    IRCLKEN  bits[  1:1  ] = 1  MCGIRCLK active.
    IREFSTEN bits[  0:0  ] = 0  Internal reference clock is disabled in Stop mode.
```



00000000000000000000000	0	0	0	0	01	000

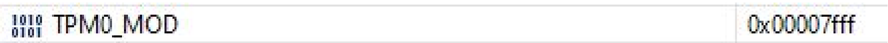


TPM0_SC 0x00000008

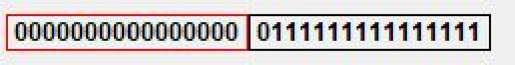


Description

```
TPM0_SC = 8

Status and Control

Bit Field Values:
          bits[ 31:9  ] = 0
    DMA   bits[  8:8  ] = 0 Disables DMA transfers.
    TOF   bits[  7:7  ] = 0 LPTPM counter has not overflowed.
    TOIE  bits[  6:6  ] = 0 Disable TOF interrupts. Use software polling or DMA
request.
    CPWMS bits[  5:5  ] = 0 LPTPM counter operates in up counting mode.
    CMOD  bits[  4:3  ] = 1 LPTPM counter increments on every LPTPM counter clock
    PS    bits[  2:0  ] = 0 Divide by 1
```

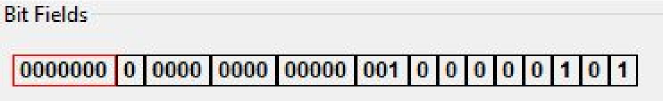


TPM0_MOD	0x00007fff



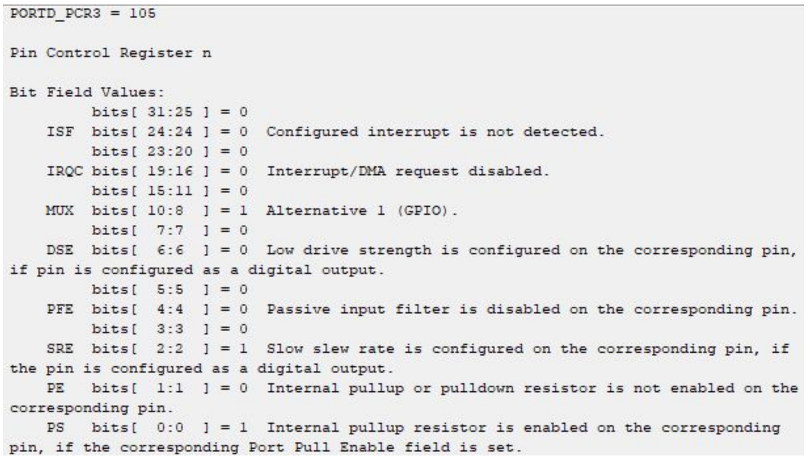
```
0000000000000000 | 0111111111111111
```

For PORTD_PCR0 to PORTD_PCR3 the configuration of the registers is the next.



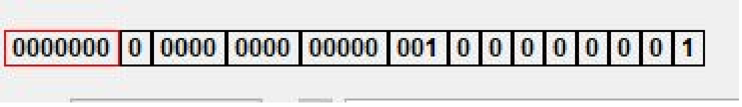
Bit Fields

```
0000000 | 0 | 0000 | 0000 | 00000 | 001 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1
```

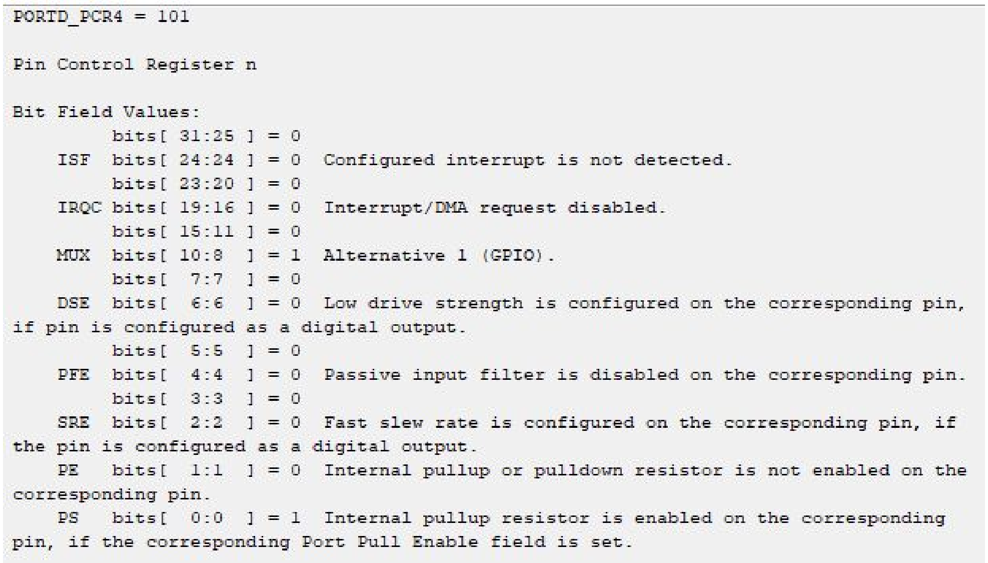


```
PORTD_PCR3 = 105

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 1  Slow slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 0  Internal pullup or pulldown resistor is not enabled on the
corresponding pin.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

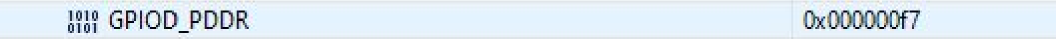
For PORTD_PCR4 to PORTD_PCR7 the configuration of the registers is the next.



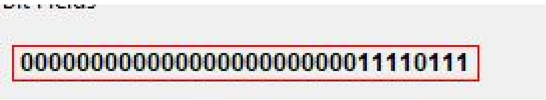
0000000	0	0000	0000	00000	001	0	0	0	0	0	0	0	1



```
PORTD_PCR4 = 101

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 0  Fast slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 0  Internal pullup or pulldown resistor is not enabled on the
corresponding pin.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

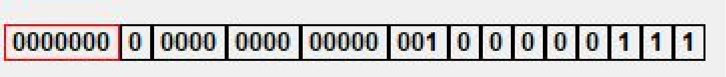


GPIOD_PDDR	0x000000f7

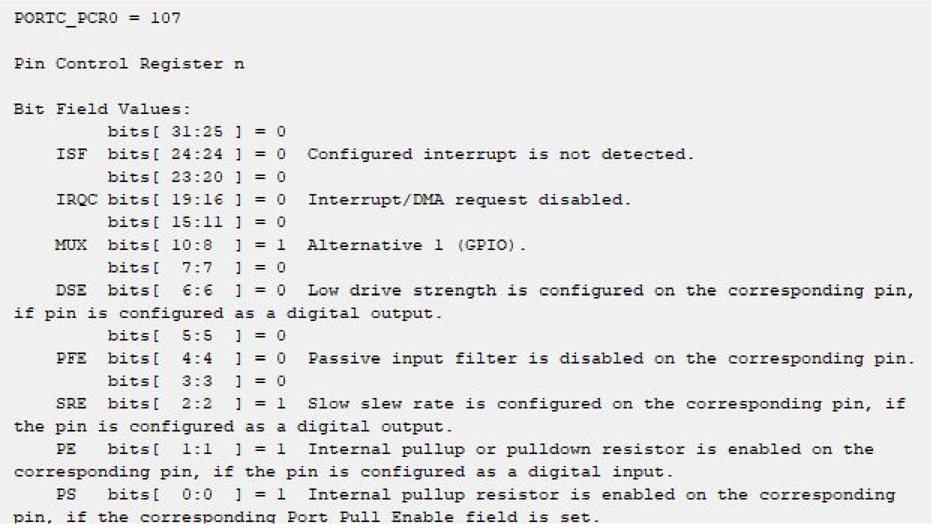


00000000000000000000000011110111

For PORTC_PCR0 to PORTC_PCR2 the configuration of the registers is the next.

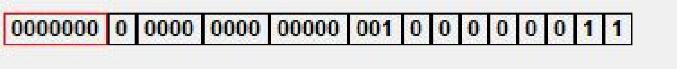


0000000	0	0000	0000	00000	001	0	0	0	0	0	1	1	1

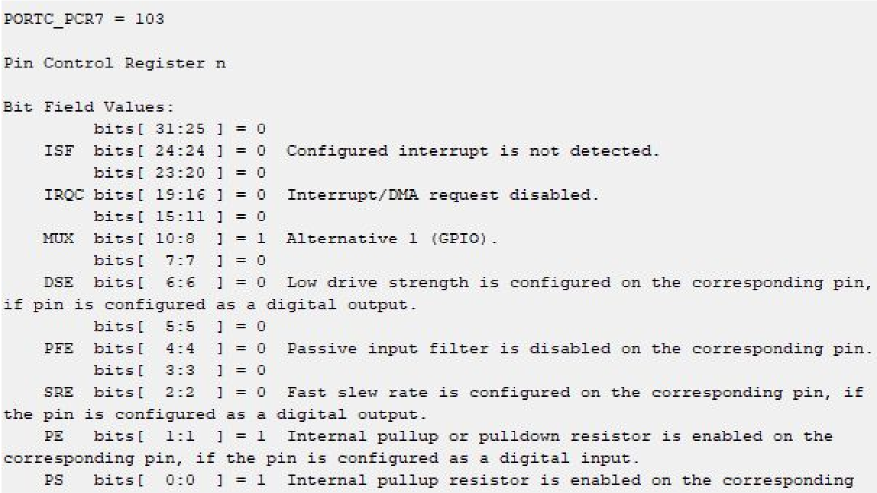


```
PORTC_PCR0 = 107

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 1  Slow slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 1  Internal pullup or pulldown resistor is enabled on the
corresponding pin, if the pin is configured as a digital input.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
pin, if the corresponding Port Pull Enable field is set.
```

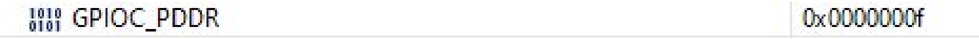
For PORTC_PCR3 to PORTC_CR7 the configuration of the registers is the next.



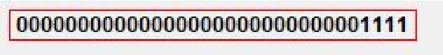
```
0000000 | 0 | 0000 | 0000 | 00000 | 001 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1
```



```
PORTC_PCR7 = 103

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 0  Fast slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 1  Internal pullup or pulldown resistor is enabled on the
corresponding pin, if the pin is configured as a digital input.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
```

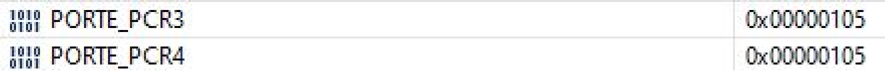


GPIOC_PDDR	0x0000000f

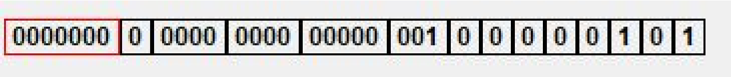


```
00000000000000000000000000001111
```

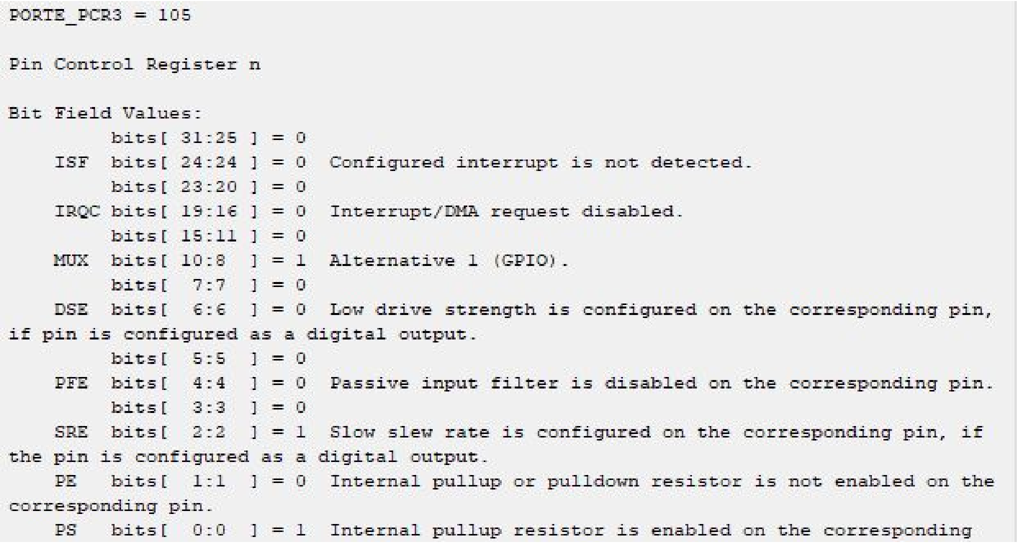
For PORTE_PCR3 to PORTE_CR4 the configuration of the registers is the next.



PORTE_PCR3	0x00000105
PORTE_PCR4	0x00000105

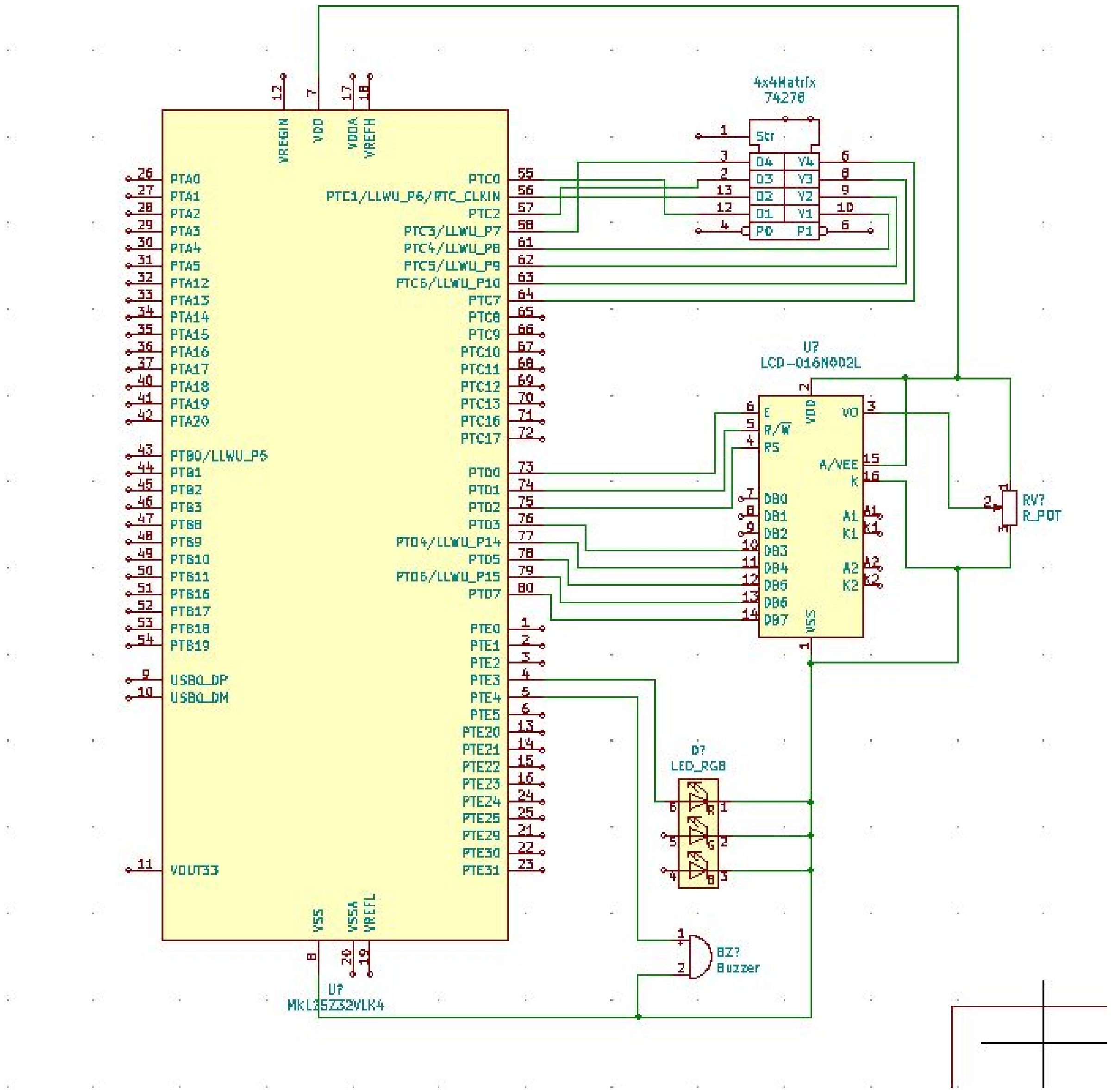


```
0000000 | 0 | 0000 | 0000 | 00000 | 001 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1
```



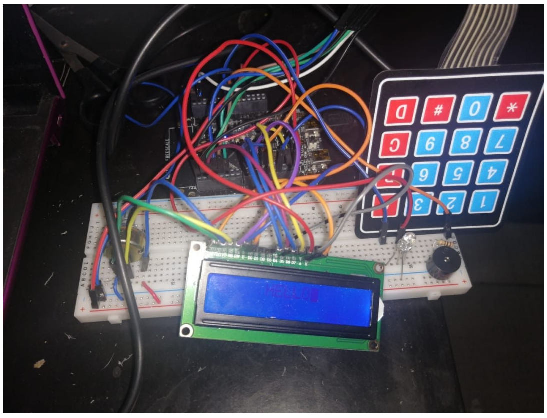
```
PORTE_PCR3 = 105

Pin Control Register n

Bit Field Values:
         bits[ 31:25 ] = 0
    ISF  bits[ 24:24 ] = 0  Configured interrupt is not detected.
         bits[ 23:20 ] = 0
    IRQC bits[ 19:16 ] = 0  Interrupt/DMA request disabled.
         bits[ 15:11 ] = 0
    MUX  bits[ 10:8  ] = 1  Alternative 1 (GPIO).
         bits[  7:7  ] = 0
    DSE  bits[  6:6  ] = 0  Low drive strength is configured on the corresponding pin,
if pin is configured as a digital output.
         bits[  5:5  ] = 0
    PFE  bits[  4:4  ] = 0  Passive input filter is disabled on the corresponding pin.
         bits[  3:3  ] = 0
    SRE  bits[  2:2  ] = 1  Slow slew rate is configured on the corresponding pin, if
the pin is configured as a digital output.
    PE   bits[  1:1  ] = 0  Internal pullup or pulldown resistor is not enabled on the
corresponding pin.
    PS   bits[  0:0  ] = 1  Internal pullup resistor is enabled on the corresponding
```

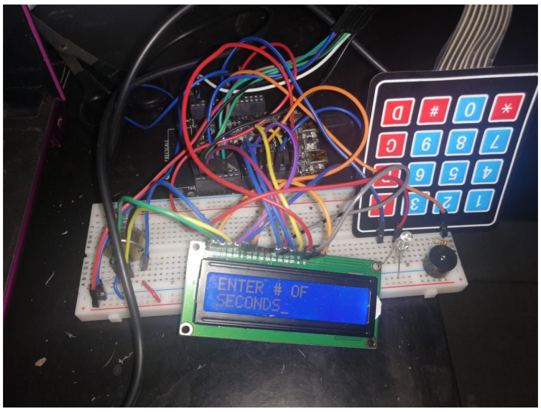
Schematic:



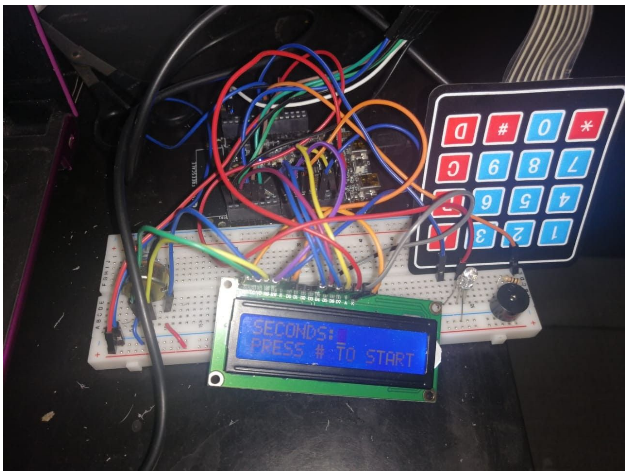
Video: https://youtu.be/rrnw49vgrLg

Picture of the circuit:





ENTER # OF
SECONDS_



SECONDS:
PRESS # TO START