Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Report week 13: LCD, Keyboard and PWM integration.

Laboratory of Microcontrollers

Gilberto Ochoa Ruiz

Perla Vanessa Jaime Gaytán ITE

A00344428

Instituto Tecnológico de Estudios Superiores de Monterrey Campus Guadalajara

Zapopan, Jalisco, México.

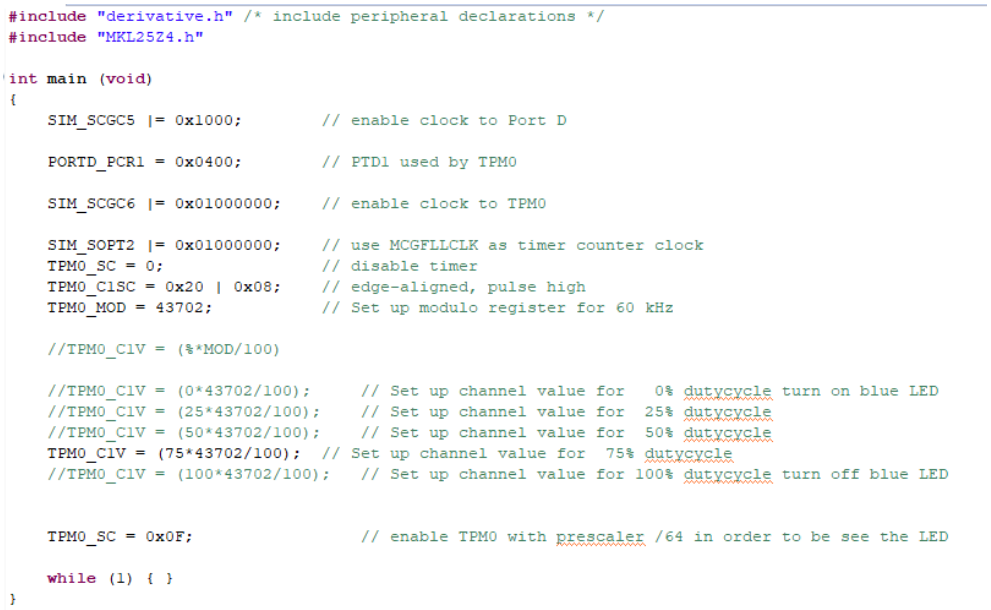
May 27, 2020

Part 1.

Modify the first example seen in class to generate PWM signals with a frequency of 60Hz (TPMx_MOD = 43702) but different duty cycles: 0, 25%, 50%,75% and 100% (since we are using non -inverted PWM we will have the LED completely on for 0 DT and off for 100%.

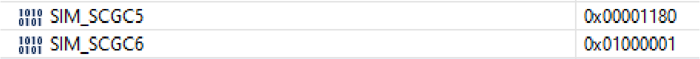
You must calculate the values for the TPMx_CnV register for the different duty cycle values.

Code:

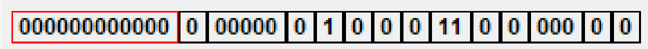


```c
#include "derivative.h" /* include peripheral declarations */
#include "MKL25Z4.h"

int main (void)
{
    SIM_SCGC5 |= 0x1000;        // enable clock to Port D

    PORTD_PCR1 = 0x0400;        // PTD1 used by TPM0

    SIM_SCGC6 |= 0x01000000;    // enable clock to TPM0

    SIM_SOPT2 |= 0x01000000;    // use MCGFLLCLK as timer counter clock
    TPM0_SC = 0;                // disable timer
    TPM0_C1SC = 0x20 | 0x08;    // edge-aligned, pulse high
    TPM0_MOD = 43702;           // Set up modulo register for 60 kHz

    //TPM0_C1V = (%*MOD/100)

    //TPM0_C1V = (0*43702/100);     // Set up channel value for   0% dutycycle turn on blue LED
    //TPM0_C1V = (25*43702/100);    // Set up channel value for  25% dutycycle
    //TPM0_C1V = (50*43702/100);    // Set up channel value for  50% dutycycle
    TPM0_C1V = (75*43702/100);  // Set up channel value for  75% dutycycle
    //TPM0_C1V = (100*43702/100);   // Set up channel value for 100% dutycycle turn off blue LED


    TPM0_SC = 0x0F;                 // enable TPM0 with prescaler /64 in order to be see the LED

    while (1) { }
}
```

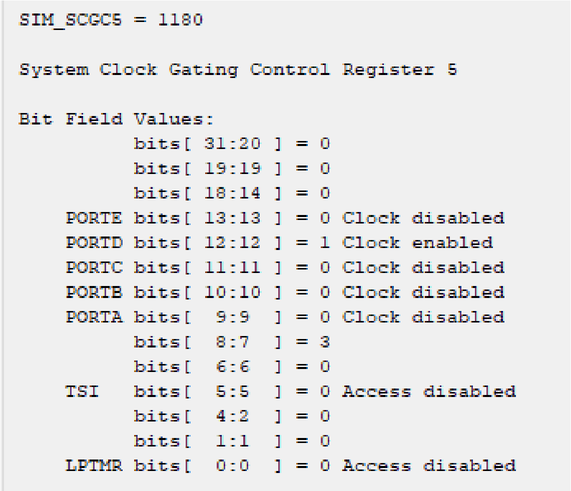
Registers:



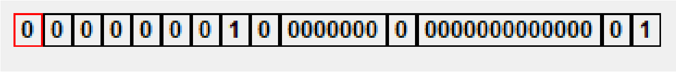
SIM_SCGC5	0x00001180
SIM_SCGC6	0x01000001



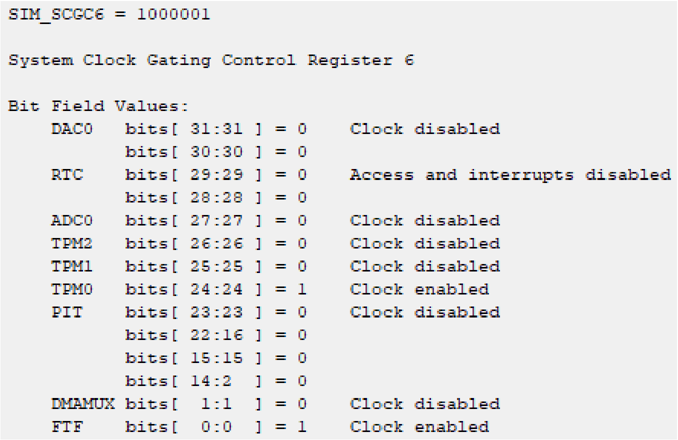
000000000000	0	00000	0	1	0	0	0	11	0	0	000	0	0



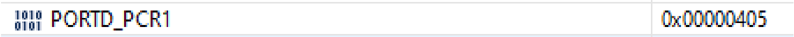
```
SIM_SCGC5 = 1180

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 ] = 0 Clock disabled
    PORTD bits[ 12:12 ] = 1 Clock enabled
    PORTC bits[ 11:11 ] = 0 Clock disabled
    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
```



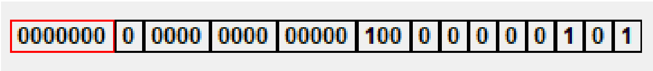
0	0	0	0	0	0	0	1	0	0000000	0	0000000000000	0	1



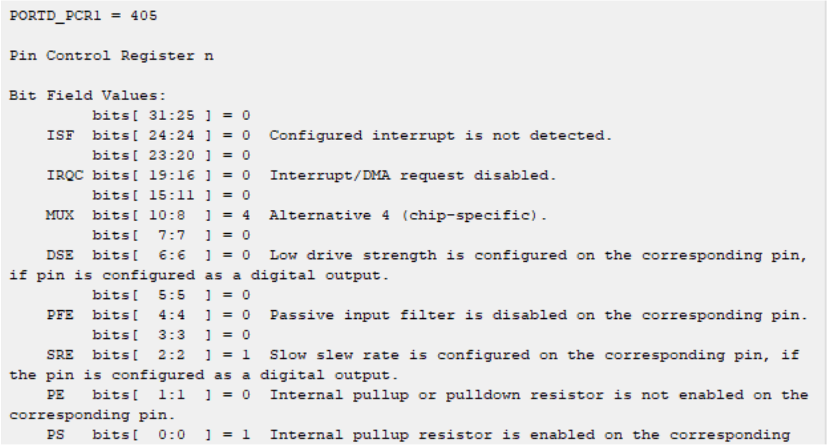
```
SIM_SCGC6 = 1000001

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
```



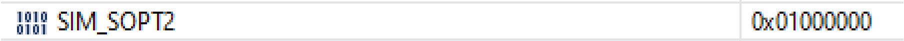
PORTD_PCR1 0x00000405



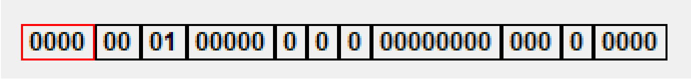
0000000 | 0 | 0000 | 0000 | 00000 | 100 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1



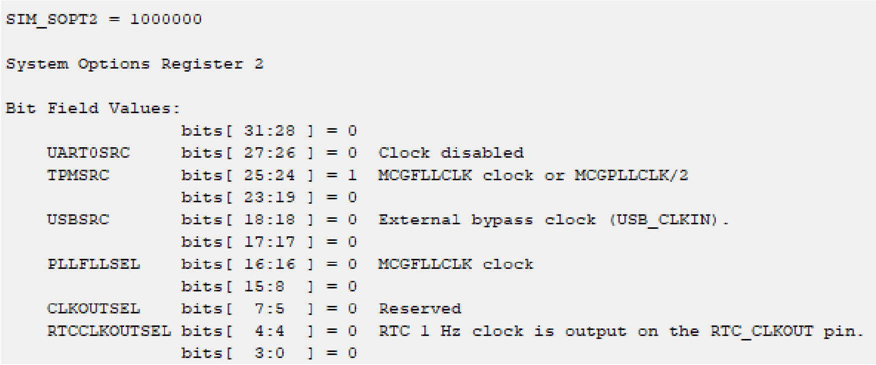
```
PORTD_PCR1 = 405

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  ] = 4  Alternative 4 (chip-specific).
         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
```



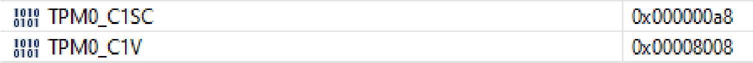
SIM_SOPT2 0x01000000



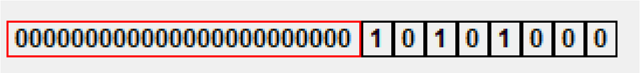
0000 | 00 | 01 | 00000 | 0 | 0 | 0 | 00000000 | 000 | 0 | 0000



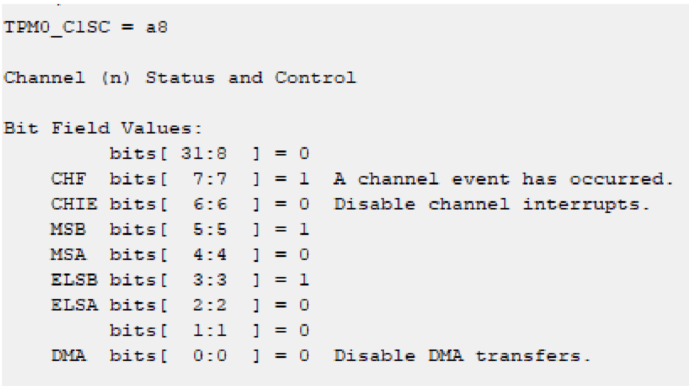
```
SIM_SOPT2 = 1000000

System Options Register 2

Bit Field Values:
                bits[ 31:28 ] = 0
    UART0SRC    bits[ 27:26 ] = 0  Clock disabled
    TPMSRC      bits[ 25:24 ] = 1  MCGFLLCLK clock or MCGPLLCLK/2
                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
```



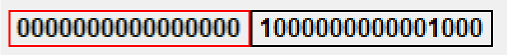
TPM0_C1SC	0x000000a8
TPM0_C1V	0x00008008



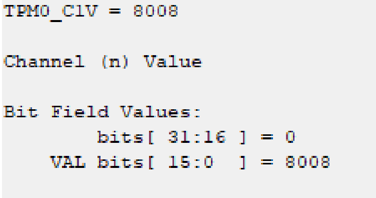
```
000000000000000000000000 1 0 1 0 1 0 0 0
```



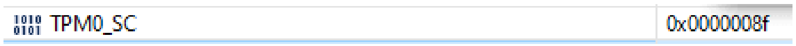
```
TPM0_C1SC = a8

Channel (n) Status and Control

Bit Field Values:
         bits[ 31:8  ] = 0
    CHF  bits[  7:7  ] = 1  A channel event has occurred.
    CHIE bits[  6:6  ] = 0  Disable channel interrupts.
    MSB  bits[  5:5  ] = 1
    MSA  bits[  4:4  ] = 0
    ELSB bits[  3:3  ] = 1
    ELSA bits[  2:2  ] = 0
         bits[  1:1  ] = 0
    DMA  bits[  0:0  ] = 0  Disable DMA transfers.
```



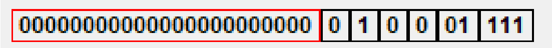
```
0000000000000000 1000000000001000
```



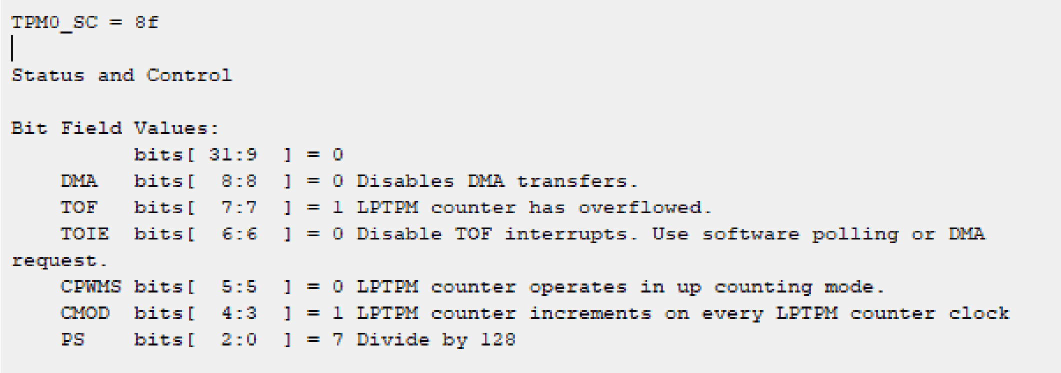
```
TPM0_C1V = 8008

Channel (n) Value

Bit Field Values:
        bits[ 31:16 ] = 0
    VAL bits[ 15:0  ] = 8008
```



TPM0_SC	0x0000008f



```
00000000000000000000000 0 1 0 0 01 111
```

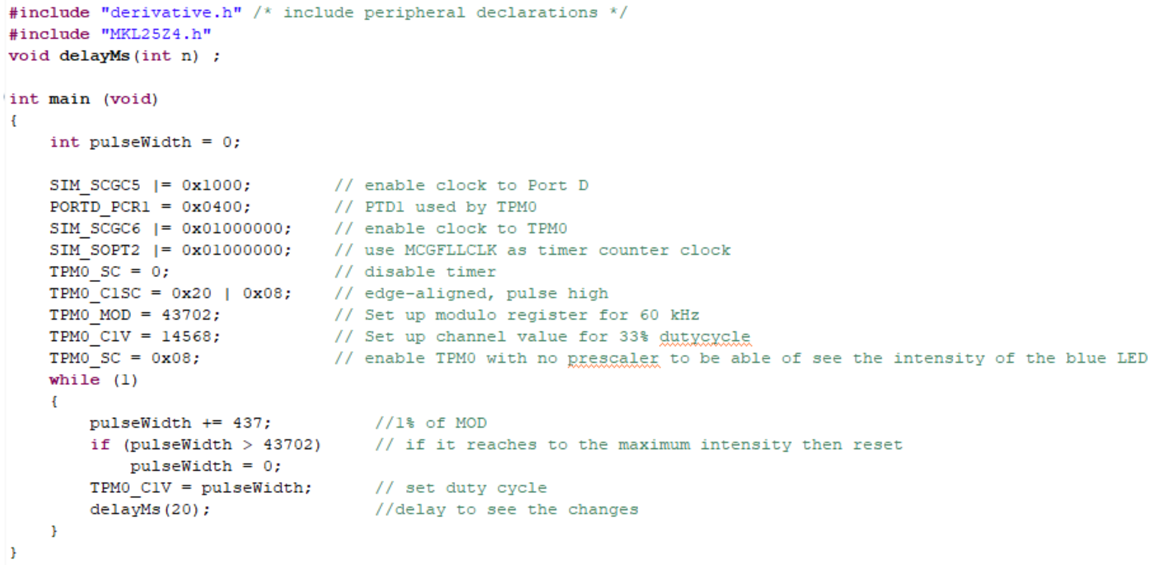


```
TPM0_SC = 8f

Status and Control

Bit Field Values:
          bits[ 31:9  ] = 0
    DMA   bits[  8:8  ] = 0 Disables DMA transfers.
    TOF   bits[  7:7  ] = 1 LPTPM counter has 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  ] = 7 Divide by 128
```

Part 2.

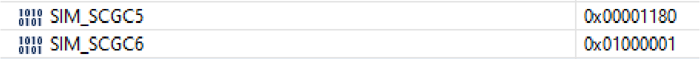
Implement the code for the second example seen in class, in which the PWM signal is changed by increments of 1% in the CnV register. Observe how the intensity of the LED decreases as we increment the duty cycle (because the LED is active low).

Code:

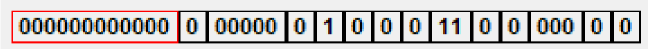


```c
#include "derivative.h" /* include peripheral declarations */
#include "MKL25Z4.h"
void delayMs(int n) ;

int main (void)
{
    int pulseWidth = 0;

    SIM_SCGC5 |= 0x1000;        // enable clock to Port D
    PORTD_PCR1 = 0x0400;        // PTD1 used by TPM0
    SIM_SCGC6 |= 0x01000000;    // enable clock to TPM0
    SIM_SOPT2 |= 0x01000000;    // use MCGFLLCLK as timer counter clock
    TPM0_SC = 0;                // disable timer
    TPM0_C1SC = 0x20 | 0x08;    // edge-aligned, pulse high
    TPM0_MOD = 43702;           // Set up modulo register for 60 kHz
    TPM0_C1V = 14568;           // Set up channel value for 33% dutycycle
    TPM0_SC = 0x08;             // enable TPM0 with no prescaler to be able of see the intensity of the blue LED
    while (1)
    {
        pulseWidth += 437;          //1% of MOD
        if (pulseWidth > 43702)     // if it reaches to the maximum intensity then reset
            pulseWidth = 0;
        TPM0_C1V = pulseWidth;      // set duty cycle
        delayMs(20);                //delay to see the changes
    }
}
```

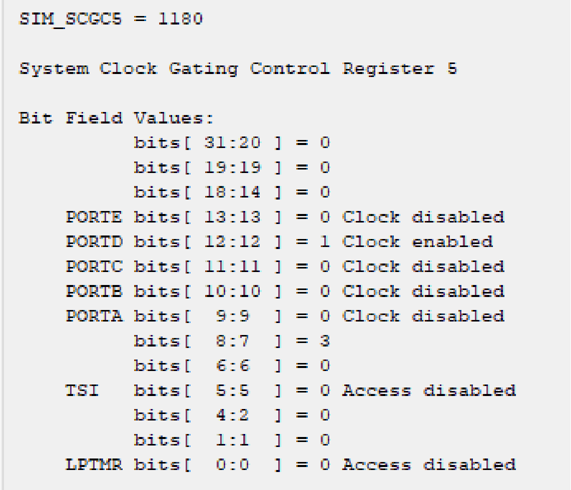
Registers:



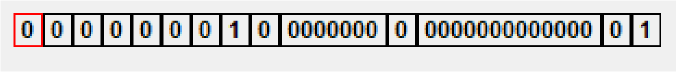
SIM_SCGC5	0x00001180
SIM_SCGC6	0x01000001



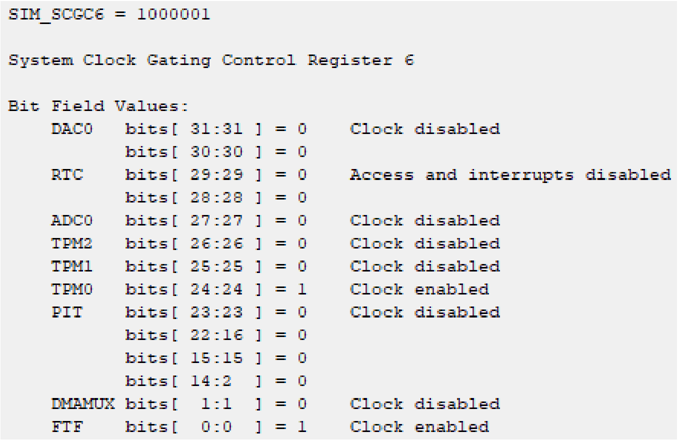
```
000000000000 | 0 | 00000 | 0 | 1 | 0 | 0 | 0 | 11 | 0 | 0 | 000 | 0 | 0
```



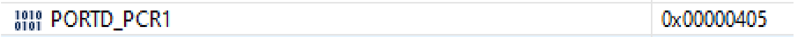
```
SIM_SCGC5 = 1180

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 ] = 0 Clock disabled
    PORTD bits[ 12:12 ] = 1 Clock enabled
    PORTC bits[ 11:11 ] = 0 Clock disabled
    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
```



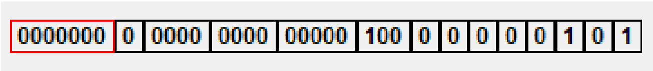
```
0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0000000 | 0 | 0000000000000 | 0 | 1
```



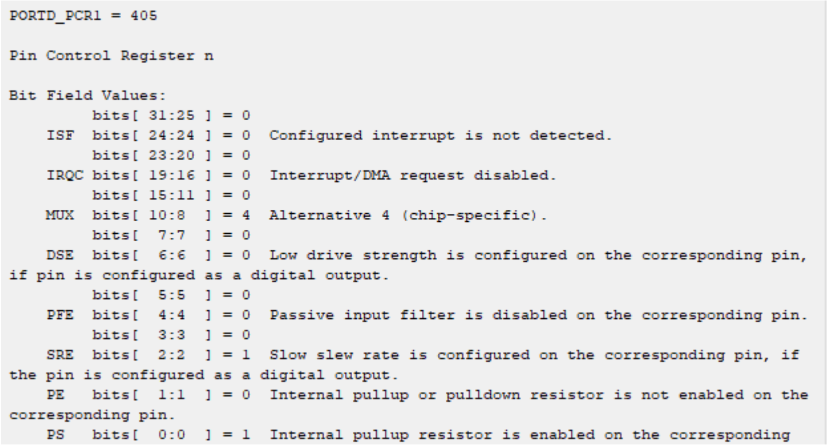
```
SIM_SCGC6 = 1000001

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
```



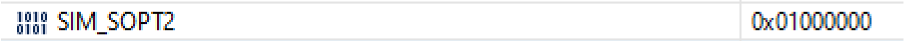
PORTD_PCR1 0x00000405



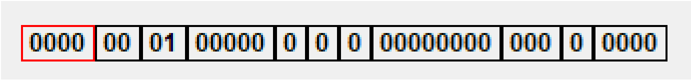
0000000 | 0 | 0000 | 0000 | 00000 | 100 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1



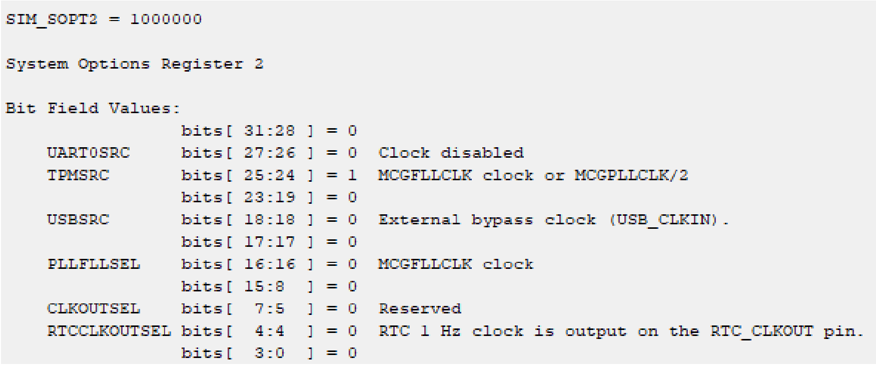
```
PORTD_PCR1 = 405

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  ] = 4  Alternative 4 (chip-specific).
         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
```



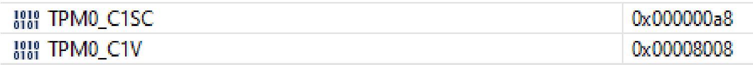
SIM_SOPT2 0x01000000



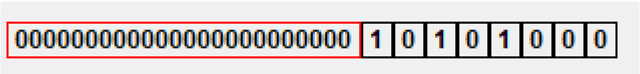
0000 | 00 | 01 | 00000 | 0 | 0 | 0 | 00000000 | 000 | 0 | 0000



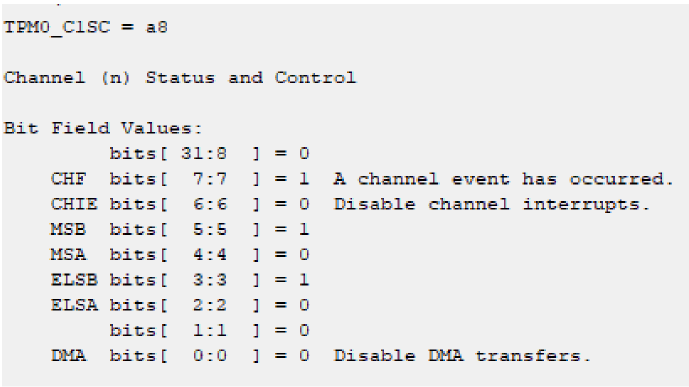
```
SIM_SOPT2 = 1000000

System Options Register 2

Bit Field Values:
                bits[ 31:28 ] = 0
    UART0SRC    bits[ 27:26 ] = 0  Clock disabled
    TPMSRC      bits[ 25:24 ] = 1  MCGFLLCLK clock or MCGPLLCLK/2
                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
```



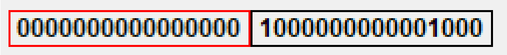
TPM0_C1SC	0x000000a8
TPM0_C1V	0x00008008



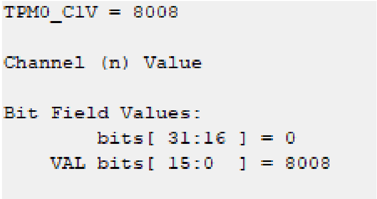
000000000000000000000000	1	0	1	0	1	0	0	0



```
TPM0_C1SC = a8

Channel (n) Status and Control

Bit Field Values:
         bits[ 31:8  ] = 0
    CHF  bits[  7:7  ] = 1  A channel event has occurred.
    CHIE bits[  6:6  ] = 0  Disable channel interrupts.
    MSB  bits[  5:5  ] = 1
    MSA  bits[  4:4  ] = 0
    ELSB bits[  3:3  ] = 1
    ELSA bits[  2:2  ] = 0
         bits[  1:1  ] = 0
    DMA  bits[  0:0  ] = 0  Disable DMA transfers.
```

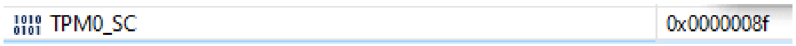


0000000000000000	1000000000001000

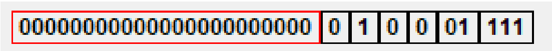


```
TPM0_C1V = 8008

Channel (n) Value

Bit Field Values:
        bits[ 31:16 ] = 0
    VAL bits[ 15:0  ] = 8008
```

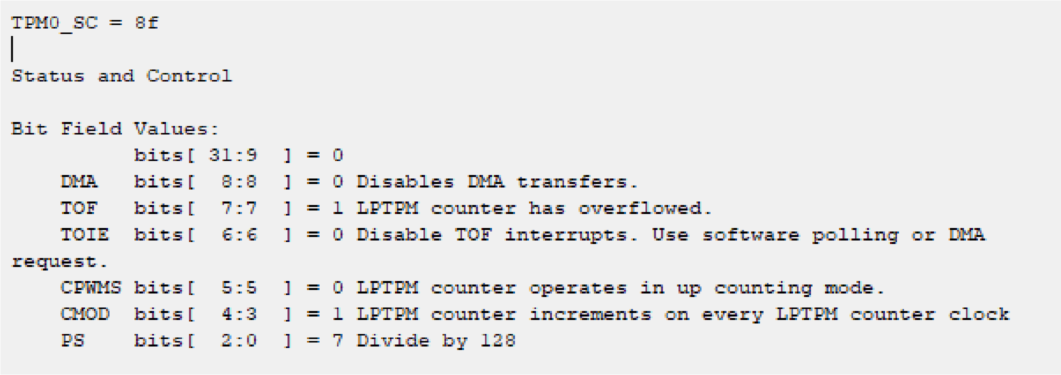
changes through the time



TPM0_SC	0x0000008f



00000000000000000000000	0	1	0	0	01	111



```
TPM0_SC = 8f

Status and Control

Bit Field Values:
          bits[ 31:9  ] = 0
    DMA   bits[  8:8  ] = 0 Disables DMA transfers.
    TOF   bits[  7:7  ] = 1 LPTPM counter has 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  ] = 7 Divide by 128
```

Part 3.

Simple industrial control with PWM. Imagine that we want to create an industrial grinder. Usually, just like a with house blender, these systems have several power configurations depending on how "hard" are the components (i.e. gravel, stones) we wish to grind and thus the motor changes speed (and thus torque) to do the work. In this sense, we will create a simple application that can change the speed depending on a setting defined by the user

1. **When you start your application, the message next message should be displayed.**

   Set the mode input mode

   Mode 1: M Mode 2: A

   Mode M stands for manual and mode 2 stands for Automatic. We will see what each mode implies. You can use each mode by pressing whichever button in the keyboard you choose and pressing # toe execute the rest of your application

2. **In manual mode, the idea is that you integrate the first part of this lab but modifying the code for generating inverted PWM signals. Thus, the LCD should display**

   Select Speed

   1: L 2:M 3:MH 4:H

   (standing for Low, Medium, Medium High and High)

   After pressing the one of the keys the corresponding CnV register values should be sent to a function the initializes the PWM (duty cycles of 25%, 50%, 75% and 100%). Please send the value as an argument to the function (if you use if statements the code becomes very long and it is very inefficient). The

LED is this case will go from very low intensity to completely on (as the case of a motor). If you have a fan or a small motor you can try as we do not require an H bridge.

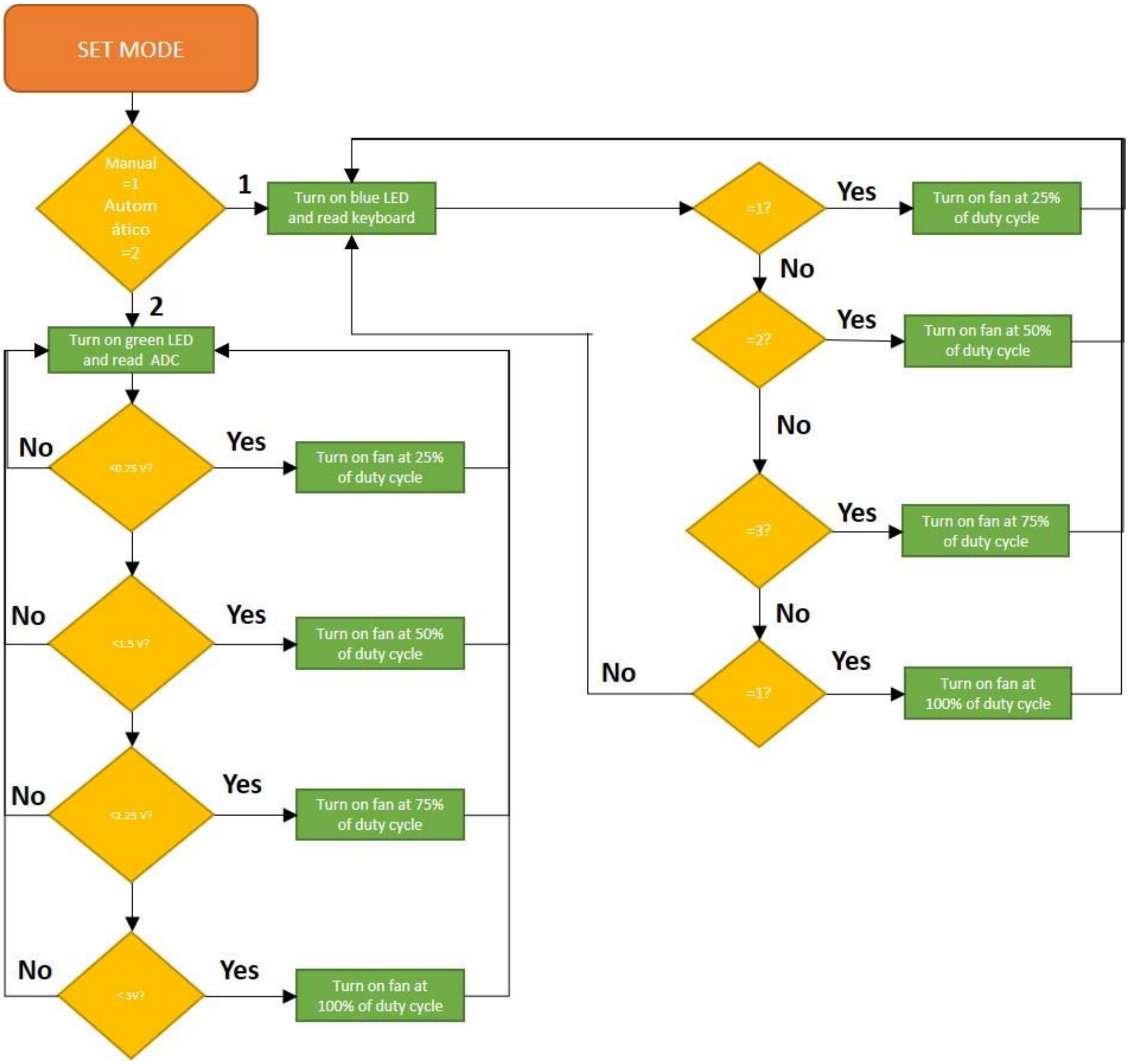
3. After this, if we want to modify the speed of the motor, we can use a push button to send an interrupt to the MCU and display the above menu again. A second button can be used to stop the system the motor at any given time (emergency signal)

Just as a note, the ADC could be used here to monitor the motor current or heat and send a signal to the microcontroller if there is an overload (the system cannot grind something for instance) and increase the power or switch the motor off. Something in the context is explored in the next part of the lab (optional)

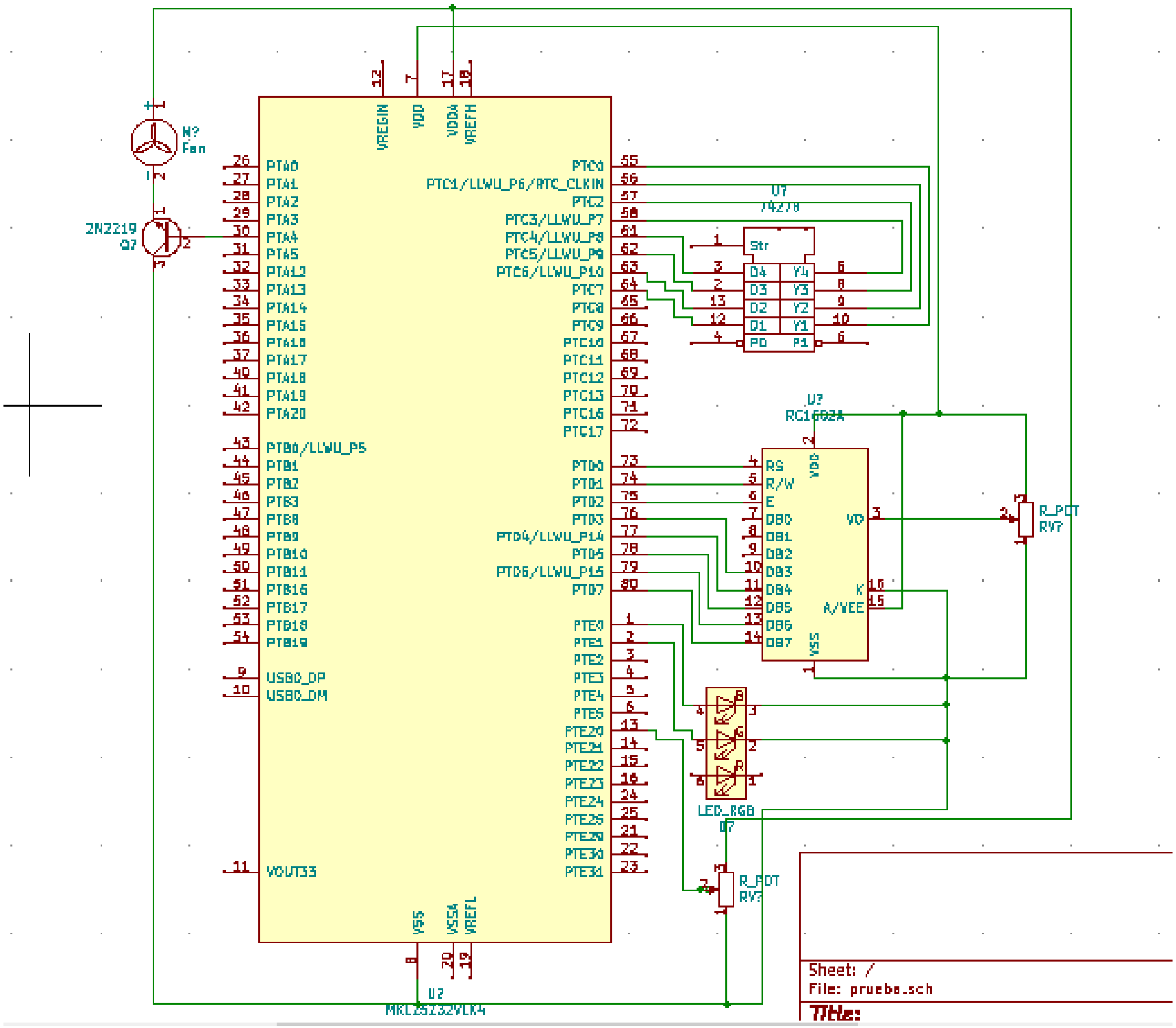
Automatic Mode (extra points)

1. This system is very similar to the previous case: there are 4 predefined power levels, which are chosen depending on the value of the CnV register. However, in this case, we monitor continuously the value of the ADC connected to a potentiometer (simulating the load of the motor). If the value is between 0 – 0.75V the motor should run on Mode 1; between 0.76 and 1.5V on Mode 2; between 1.51 and 2.25V on mode 3 and finally, between 2.25 and 3V on Mode 4. We also should use a button to go the main menu and a second button to stop the motor.

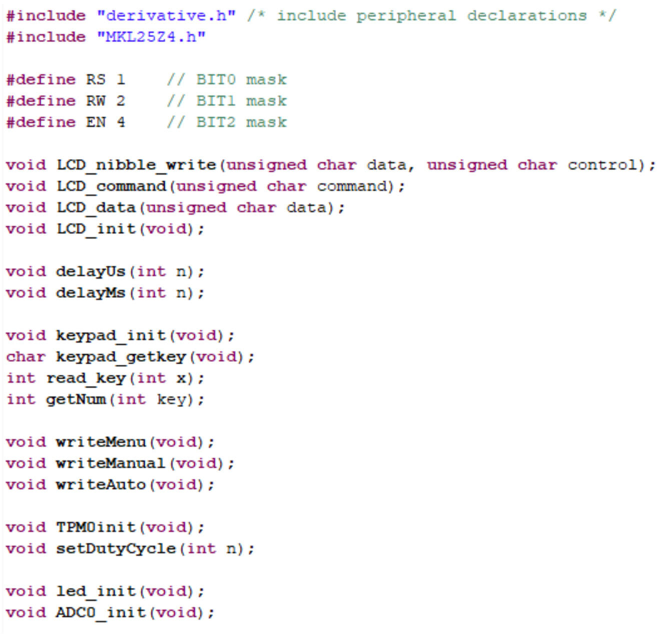
Flow Diagram.


SET MODE
Manual =1 Automático =2
1
2
Turn on blue LED and read keyboard
=1?
Yes
Turn on fan at 25% of duty cycle
No
=2?
Yes
Turn on fan at 50% of duty cycle
No
=3?
Yes
Turn on fan at 75% of duty cycle
No
=1?
Yes
Turn on fan at 100% of duty cycle
No
Turn on green LED and read ADC
<0.75 V?
Yes
Turn on fan at 25% of duty cycle
No
<1.5 V?
Yes
Turn on fan at 50% of duty cycle
No
<2.25 V?
Yes
Turn on fan at 75% of duty cycle
No
< 3V?
Yes
Turn on fan at 100% of duty cycle
No

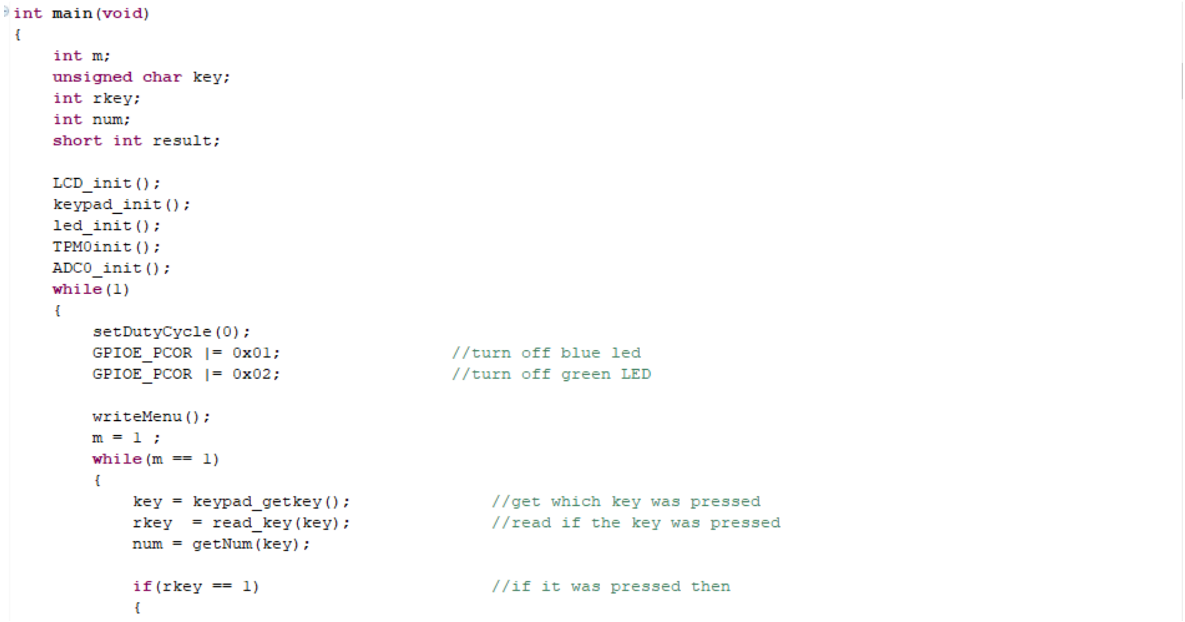
Schematic.


M?
Fan
2N2219
Q?
VREGIN
VDD
VDDA
VREFH
PTA0
PTA1
PTA2
PTA3
PTA4
PTA5
PTA12
PTA13
PTA14
PTA15
PTA16
PTA17
PTA18
PTA19
PTA20
PTB0/LLWU_P5
PTB1
PTB2
PTB3
PTB8
PTB9
PTB10
PTB11
PTB16
PTB17
PTB18
PTB19
USB0_DP
USB0_DM
VOUT33
PTC0
PTC1/LLWU_P6/RTC_CLKIN
PTC2
PTC3/LLWU_P7
PTC4/LLWU_P8
PTC5/LLWU_P9
PTC6/LLWU_P10
PTC7
PTC8
PTC9
PTC10
PTC11
PTC12
PTC13
PTC16
PTC17
PTD0
PTD1
PTD2
PTD3
PTD4/LLWU_P14
PTD5
PTD6/LLWU_P15
PTD7
PTE0
PTE1
PTE2
PTE3
PTE4
PTE5
PTE20
PTE21
PTE22
PTE23
PTE24
PTE25
PTE29
PTE30
PTE31
VSS
VSSA
VREFL
U?
MKL25Z32VLK4
U?
74278
Str
D4
D3
D2
D1
PD
Y4
Y3
Y2
Y1
P1
U?
RC1602A
VDD
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
VO
K
A/VEE
VSS
R_POT
RV?
LED_RGB
D?
B
G
R
R_POT
RV?
Sheet: /
File: prueba.sch
Title:

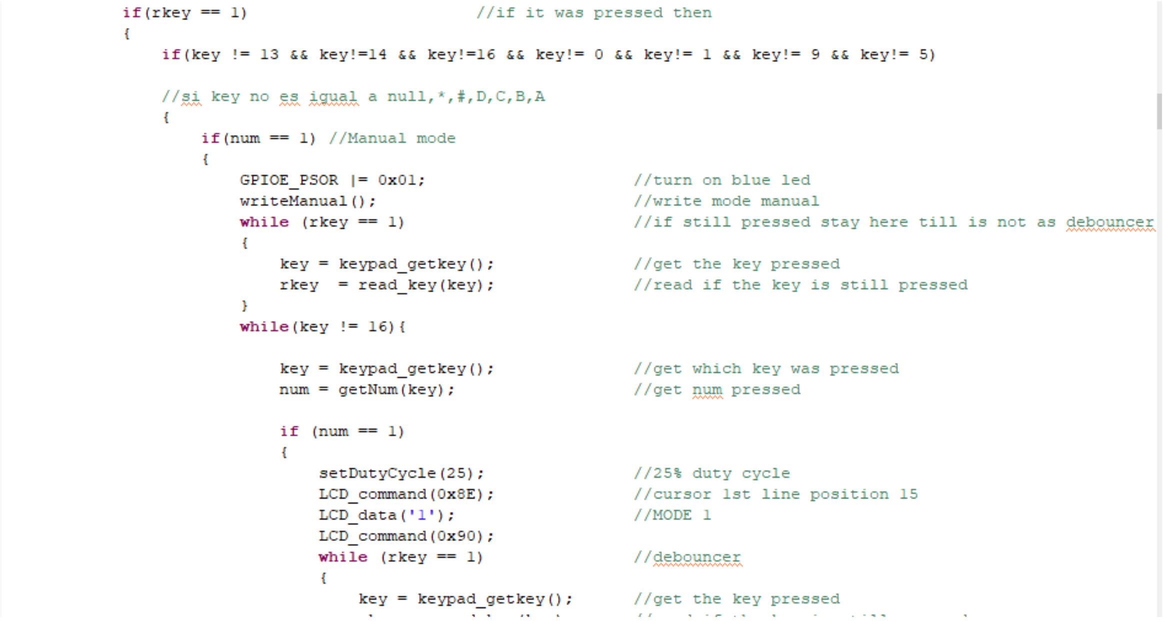
Code:



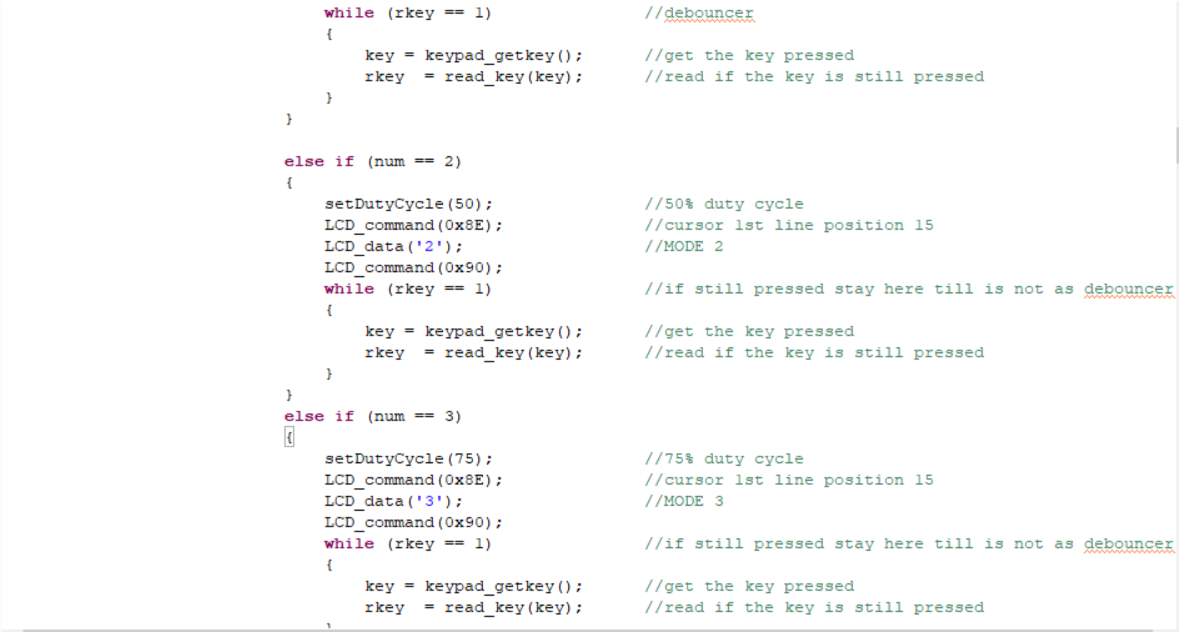
```
#include "derivative.h" /* include peripheral declarations */
#include "MKL25Z4.h"

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

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 delayMs(int n);

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

void writeMenu(void);
void writeManual(void);
void writeAuto(void);

void TPM0init(void);
void setDutyCycle(int n);

void led_init(void);
void ADC0_init(void);
```



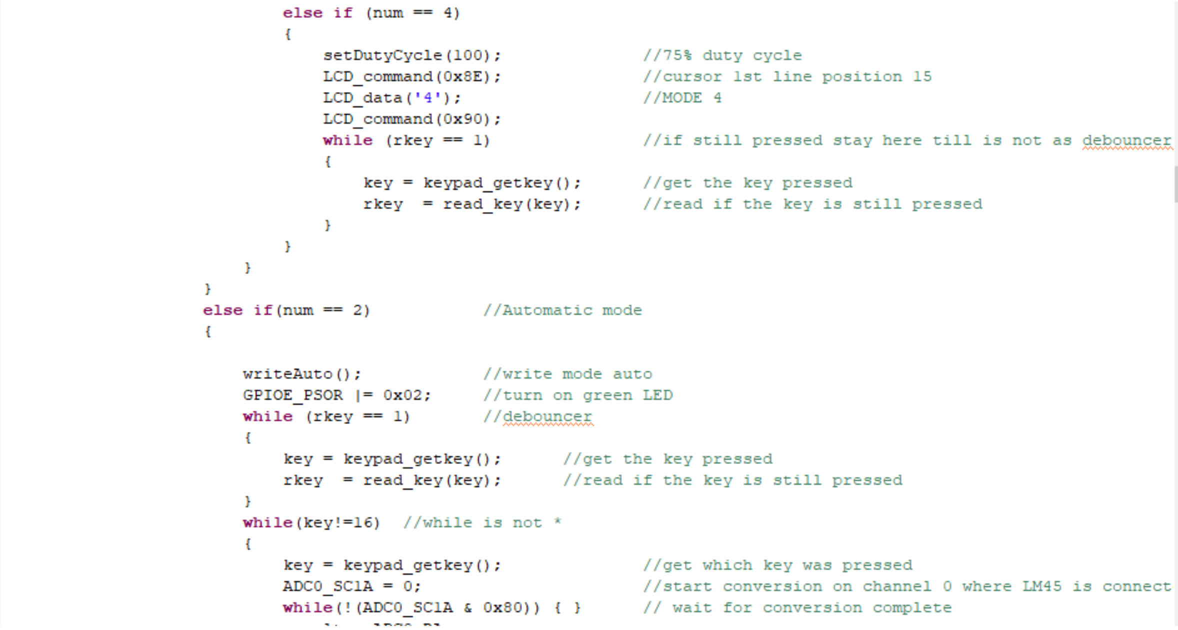
```
int main(void)
{
    int m;
    unsigned char key;
    int rkey;
    int num;
    short int result;

    LCD_init();
    keypad_init();
    led_init();
    TPM0init();
    ADC0_init();
    while(1)
    {
        setDutyCycle(0);
        GPIOE_PCOR |= 0x01;                 //turn off blue led
        GPIOE_PCOR |= 0x02;                 //turn off green LED

        writeMenu();
        m = 1 ;
        while(m == 1)
        {
            key = keypad_getkey();              //get which key was pressed
            rkey  = read_key(key);              //read if the key was pressed
            num = getNum(key);

            if(rkey == 1)                       //if it was pressed then
            {
```



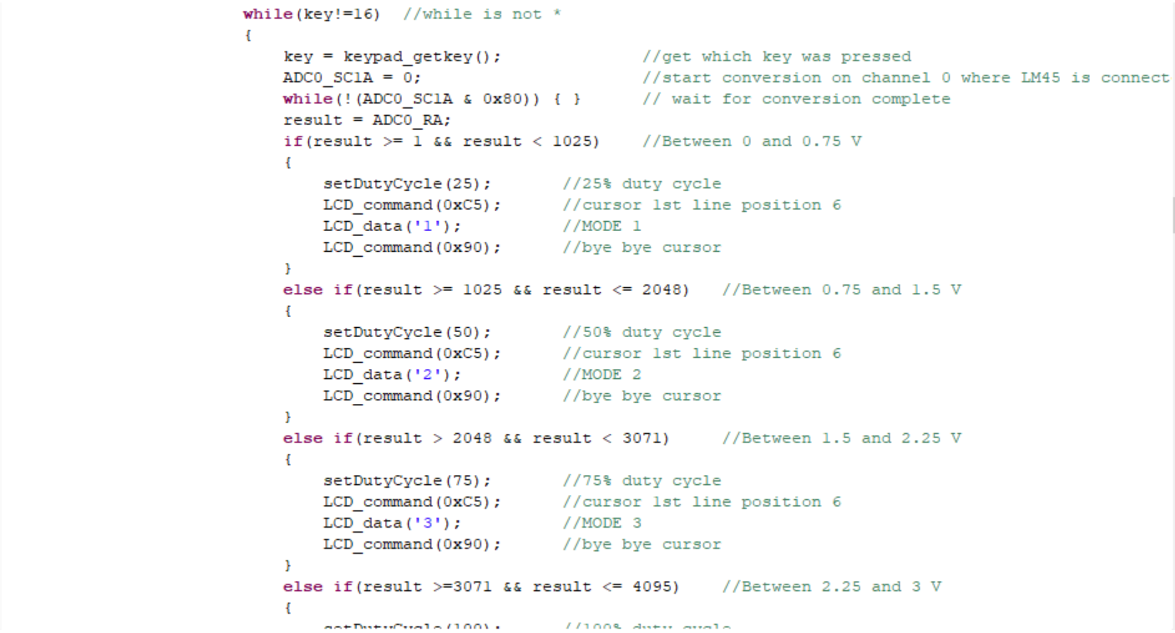
```
if(rkey == 1)                       //if it was pressed then
{
    if(key != 13 && key!=14 && key!=16 && key!= 0 && key!= 1 && key!= 9 && key!= 5)

    //si key no es igual a null,*,#,D,C,B,A
    {
        if(num == 1) //Manual mode
        {
            GPIOE_PSOR |= 0x01;                     //turn on blue led
            writeManual();                          //write mode manual
            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
            }
            while(key != 16){

                key = keypad_getkey();              //get which key was pressed
                num = getNum(key);                  //get num pressed

                if (num == 1)
                {
                    setDutyCycle(25);               //25% duty cycle
                    LCD_command(0x8E);              //cursor 1st line position 15
                    LCD_data('1');                  //MODE 1
                    LCD_command(0x90);
                    while (rkey == 1)               //debouncer
                    {
                        key = keypad_getkey();      //get the key pressed
```



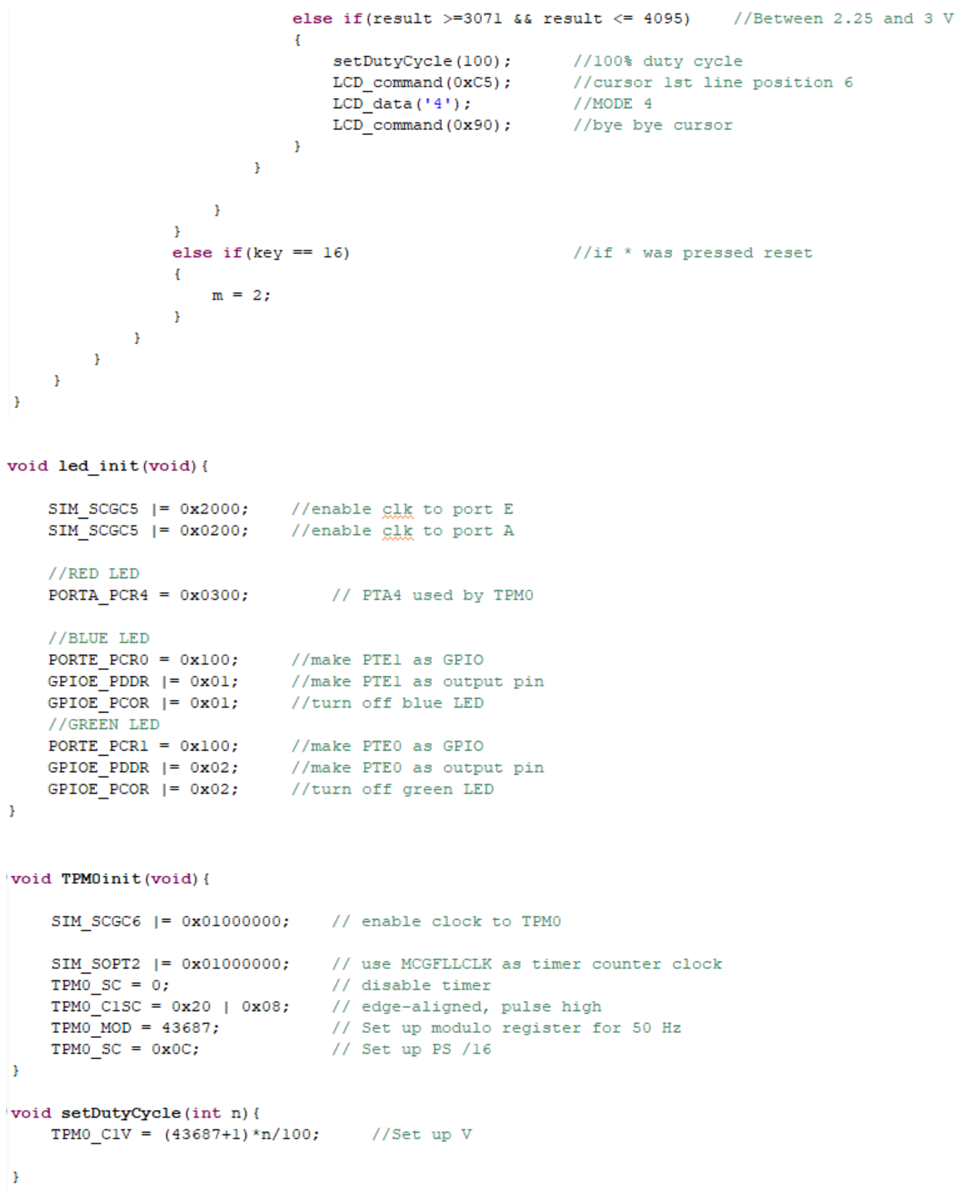
```
                    while (rkey == 1)               //debouncer
                    {
                        key = keypad_getkey();      //get the key pressed
                        rkey  = read_key(key);      //read if the key is still pressed
                    }
                }

                else if (num == 2)
                {
                    setDutyCycle(50);               //50% duty cycle
                    LCD_command(0x8E);              //cursor 1st line position 15
                    LCD_data('2');                  //MODE 2
                    LCD_command(0x90);
                    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 (num == 3)
                {
                    setDutyCycle(75);               //75% duty cycle
                    LCD_command(0x8E);              //cursor 1st line position 15
                    LCD_data('3');                  //MODE 3
                    LCD_command(0x90);
                    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 (num == 4)
        {
            setDutyCycle(100);                  //75% duty cycle
            LCD_command(0x8E);                  //cursor 1st line position 15
            LCD_data('4');                      //MODE 4
            LCD_command(0x90);
            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(num == 2)           //Automatic mode
{

    writeAuto();            //write mode auto
    GPIOE_PSOR |= 0x02;     //turn on green LED
    while (rkey == 1)       //debouncer
    {
        key = keypad_getkey();      //get the key pressed
        rkey  = read_key(key);      //read if the key is still pressed
    }
    while(key!=16)  //while is not *
    {
        key = keypad_getkey();              //get which key was pressed
        ADC0_SC1A = 0;                      //start conversion on channel 0 where LM45 is connect
        while(!(ADC0_SC1A & 0x80)) { }      // wait for conversion complete
```



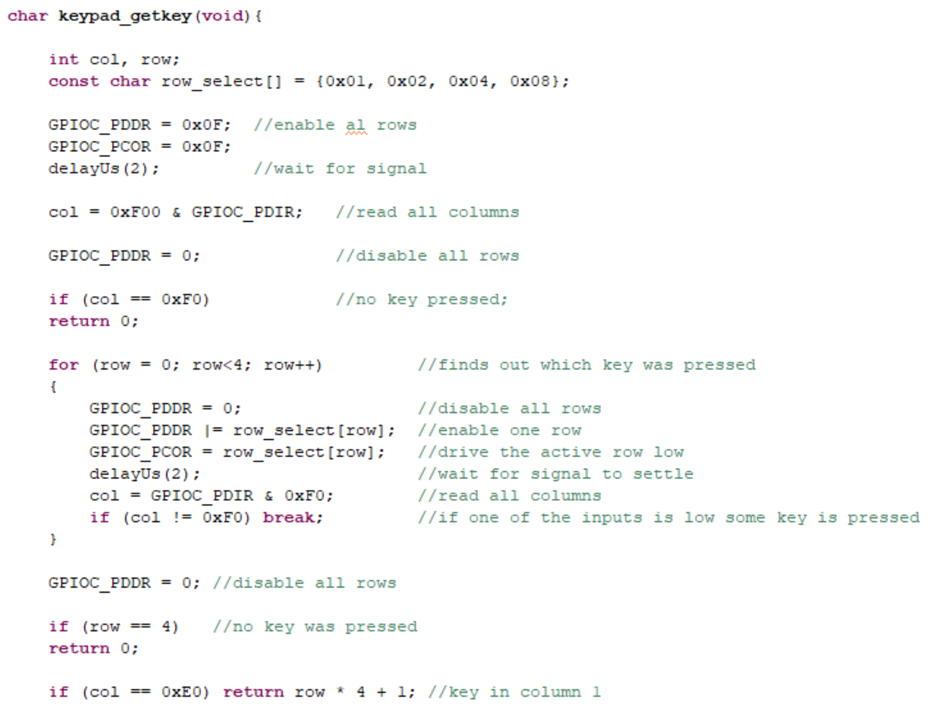
```
    while(key!=16)  //while is not *
    {
        key = keypad_getkey();              //get which key was pressed
        ADC0_SC1A = 0;                      //start conversion on channel 0 where LM45 is connect
        while(!(ADC0_SC1A & 0x80)) { }      // wait for conversion complete
        result = ADC0_RA;
        if(result >= 1 && result < 1025)    //Between 0 and 0.75 V
        {
            setDutyCycle(25);       //25% duty cycle
            LCD_command(0xC5);      //cursor 1st line position 6
            LCD_data('1');          //MODE 1
            LCD_command(0x90);      //bye bye cursor
        }
        else if(result >= 1025 && result <= 2048)   //Between 0.75 and 1.5 V
        {
            setDutyCycle(50);       //50% duty cycle
            LCD_command(0xC5);      //cursor 1st line position 6
            LCD_data('2');          //MODE 2
            LCD_command(0x90);      //bye bye cursor
        }
        else if(result > 2048 && result < 3071)     //Between 1.5 and 2.25 V
        {
            setDutyCycle(75);       //75% duty cycle
            LCD_command(0xC5);      //cursor 1st line position 6
            LCD_data('3');          //MODE 3
            LCD_command(0x90);      //bye bye cursor
        }
        else if(result >=3071 && result <= 4095)    //Between 2.25 and 3 V
        {
            setDutyCycle(100);      //100% duty cycle
```



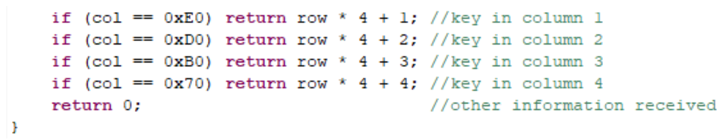
```
                            else if(result >=3071 && result <= 4095)    //Between 2.25 and 3 V
                            {
                                setDutyCycle(100);      //100% duty cycle
                                LCD_command(0xC5);      //cursor 1st line position 6
                                LCD_data('4');          //MODE 4
                                LCD_command(0x90);      //bye bye cursor
                            }
                        }

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

void led_init(void){

    SIM_SCGC5 |= 0x2000;    //enable clk to port E
    SIM_SCGC5 |= 0x0200;    //enable clk to port A

    //RED LED
    PORTA_PCR4 = 0x0300;        // PTA4 used by TPM0

    //BLUE LED
    PORTE_PCR0 = 0x100;     //make PTE1 as GPIO
    GPIOE_PDDR |= 0x01;     //make PTE1 as output pin
    GPIOE_PCOR |= 0x01;     //turn off blue LED
    //GREEN LED
    PORTE_PCR1 = 0x100;     //make PTE0 as GPIO
    GPIOE_PDDR |= 0x02;     //make PTE0 as output pin
    GPIOE_PCOR |= 0x02;     //turn off green LED
}


void TPM0init(void){

    SIM_SCGC6 |= 0x01000000;    // enable clock to TPM0

    SIM_SOPT2 |= 0x01000000;    // use MCGFLLCLK as timer counter clock
    TPM0_SC = 0;                // disable timer
    TPM0_C1SC = 0x20 | 0x08;    // edge-aligned, pulse high
    TPM0_MOD = 43687;           // Set up modulo register for 50 Hz
    TPM0_SC = 0x0C;             // Set up PS /16
}

void setDutyCycle(int n){
    TPM0_C1V = (43687+1)*n/100;     //Set up V

}
```

```
void writeManual(void)      // write mode :IDLE
{
    LCD_command(1);         // clear display
    LCD_command(0x80);      // set cursor at the begging first line
    LCD_data('S');
    LCD_data('E');
    LCD_data('T');
    LCD_data(' ');
    LCD_data('S');
    LCD_data('P');
    LCD_data('E');
    LCD_data('E');
    LCD_data('D');
    LCD_command(0xC0);      //set cursor at the begging second line
    LCD_data('1');
    LCD_data(':');
    LCD_data('L');
    LCD_data(' ');
    LCD_data('2');
    LCD_data(':');
    LCD_data('M');
    LCD_data(' ');
    LCD_data('3');
    LCD_data(':');
    LCD_data('M');
    LCD_data('H');
    LCD_data(' ');
    LCD_data('4');
    LCD_data(':');
    LCD_data('H');
}

void writeAuto(void)        // write mode :IDLE
{
    LCD_command(1);         // clear display
    LCD_command(0x80);      // set cursor at the begging first line
    LCD_data('A');
    LCD_data('U');
    LCD_data('T');
    LCD_data('O');
    LCD_data('M');
    LCD_data('A');
    LCD_data('T');
    LCD_data('I');
    LCD_data('C');
    LCD_command(0xC0);      //set cursor at the begging secon line
    LCD_data('M');
    LCD_data('O');
    LCD_data('D');
    LCD_data('E');
    LCD_command(0x90);      //bye bye cursor

}
```

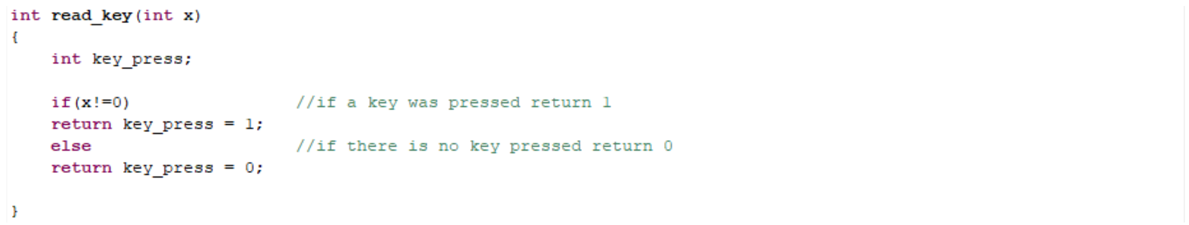
```
void writeMenu(void)        // write mode menu
{
    LCD_command(1);         // clear display
    LCD_command(0x80);      // set cursor at the begging first line
    LCD_data('S');
    LCD_data('E');
    LCD_data('T');
    LCD_data(' ');
    LCD_data('M');
    LCD_data('O');
    LCD_data('D');
    LCD_data('E');
    LCD_command(0xC0);      //set cursor at the begging first line
    LCD_data('1');
    LCD_data(':');
    LCD_data('M');
    LCD_data('a');
    LCD_data('n');
    LCD_data(' ');
    LCD_data('2');
    LCD_data(':');
    LCD_data('A');
    LCD_data('u');
    LCD_data('t');
    LCD_data('o');
    LCD_command(0x90);
}


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
}

void ADC0_init(void)
{
    SIM_SCGC5 |= 0x2000;             // clock to PORTE
    PORTE_PCR20 = 0;                 // PTE20 analog input
    SIM_SCGC6 |= 0x8000000;          // clock to ADC0
    ADC0_SC2 &= ~0x40;               // software trigger
    /* clock div by 4, long sample time, single ended 12 bit, bus clock */
    ADC0_CFG1 = 0x40 | 0x10 | 0x04 | 0x00;
}
```



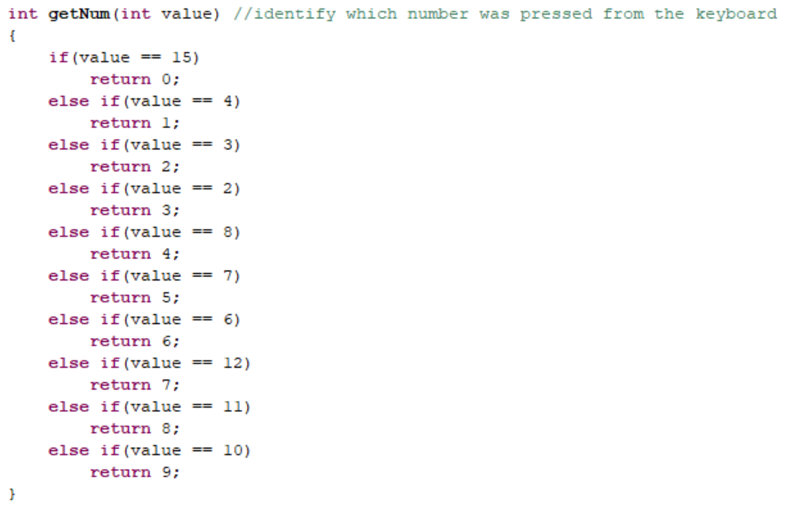
```
char keypad_getkey(void){

    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
    return 0;

    if (col == 0xE0) return row * 4 + 1; //key in column 1
```



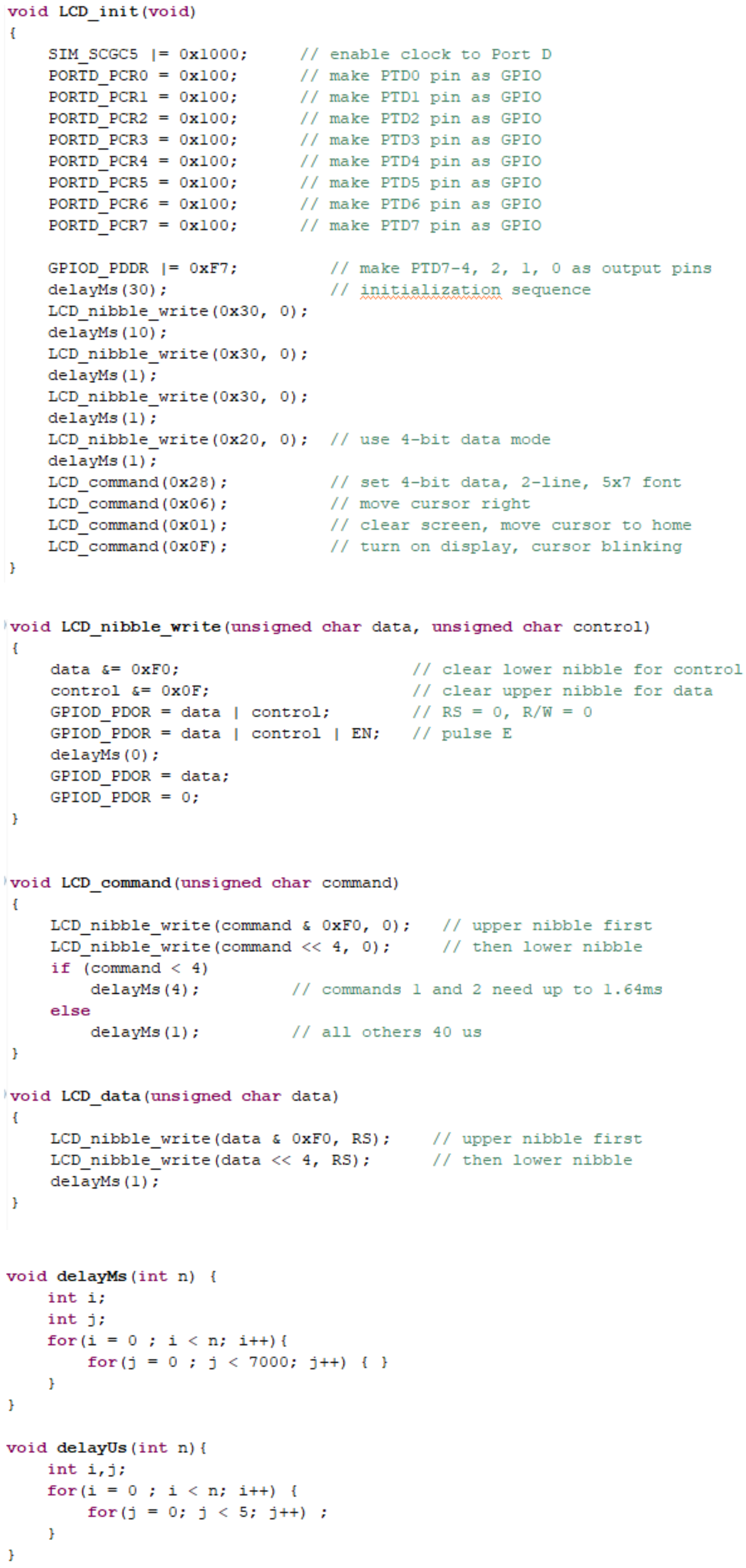
```
    if (col == 0xE0) return row * 4 + 1; //key in column 1
    if (col == 0xD0) return row * 4 + 2; //key in column 2
    if (col == 0xB0) return row * 4 + 3; //key in column 3
    if (col == 0x70) return row * 4 + 4; //key in column 4
    return 0;                            //other information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

}
```

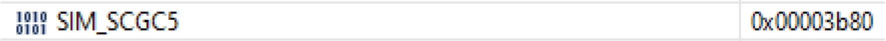


```
int getNum(int value) //identify which number was pressed from the keyboard
{
    if(value == 15)
        return 0;
    else if(value == 4)
        return 1;
    else if(value == 3)
        return 2;
    else if(value == 2)
        return 3;
    else if(value == 8)
        return 4;
    else if(value == 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;
}
```

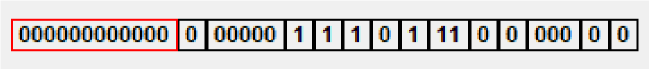


```c
void LCD_init(void)
{
    SIM_SCGC5 |= 0x1000;     // enable clock to Port D
    PORTD_PCR0 = 0x100;      // make PTD0 pin as GPIO
    PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
    PORTD_PCR2 = 0x100;      // make PTD2 pin as GPIO
    PORTD_PCR3 = 0x100;      // make PTD3 pin as GPIO
    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
}


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++) ;
    }
}
```

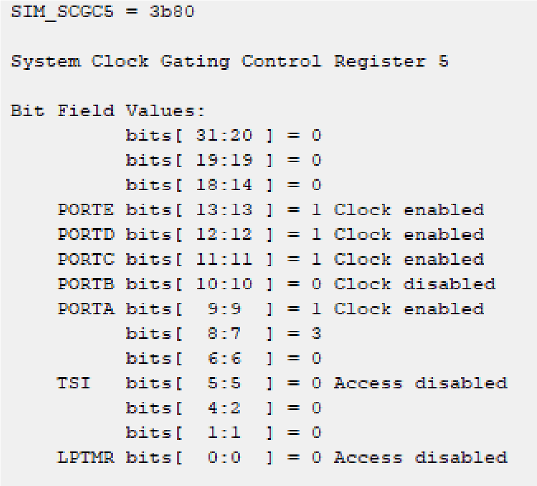
Registers:



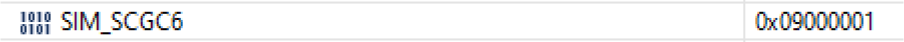
```
SIM_SCGC5                                   0x00003b80
```



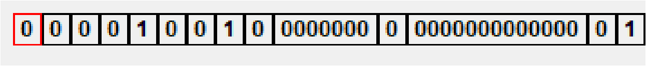
```
| 000000000000 | 0 | 00000 | 1 | 1 | 1 | 0 | 1 | 11 | 0 | 0 | 000 | 0 | 0 |
```



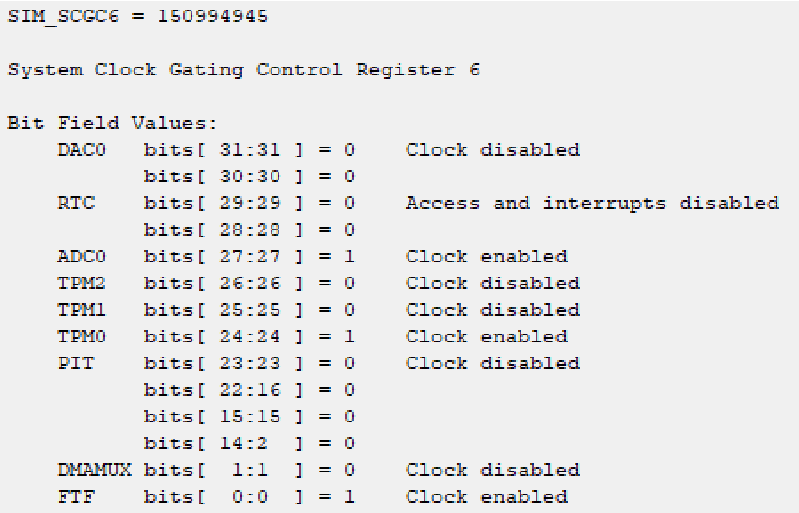
```
SIM_SCGC5 = 3b80

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  ] = 1 Clock enabled
          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
```



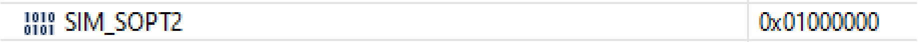
```
SIM_SCGC6                                   0x09000001
```



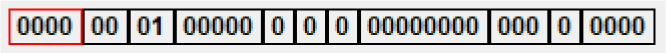
```
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0000000 | 0 | 0000000000000 | 0 | 1 |
```



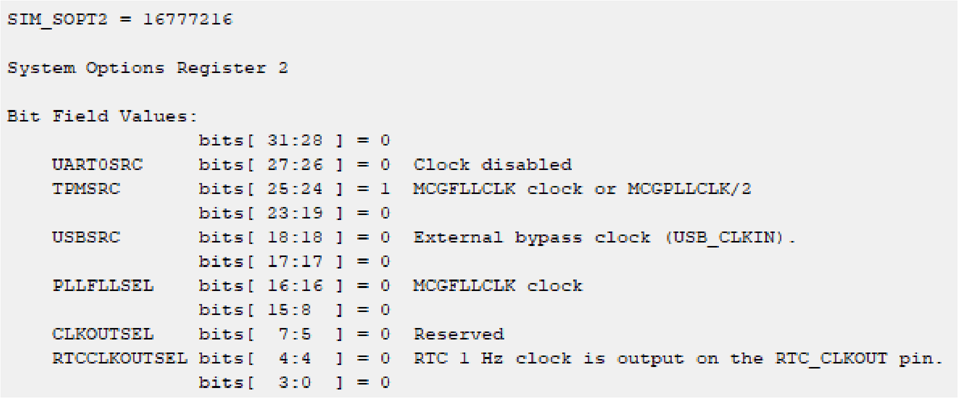
```
SIM_SCGC6 = 150994945

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 ] = 1    Clock enabled
    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
```



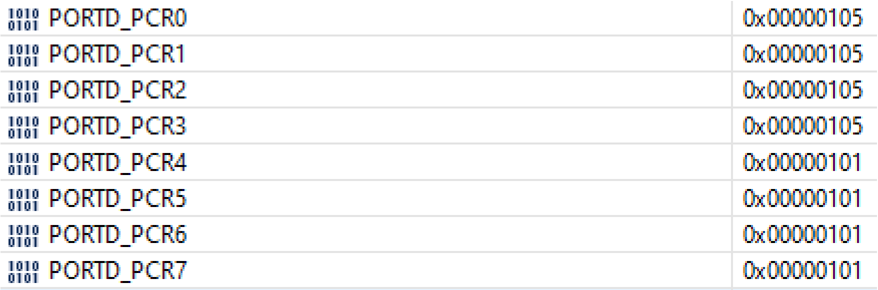
SIM_SOPT2	0x01000000



```
0000 | 00 | 01 | 00000 | 0 | 0 | 0 | 00000000 | 000 | 0 | 0000
```

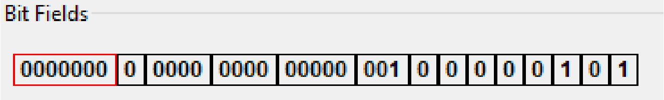


```
SIM_SOPT2 = 16777216

System Options Register 2

Bit Field Values:
                 bits[ 31:28 ] = 0
    UART0SRC     bits[ 27:26 ] = 0  Clock disabled
    TPMSRC       bits[ 25:24 ] = 1  MCGFLLCLK clock or MCGPLLCLK/2
                 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
```



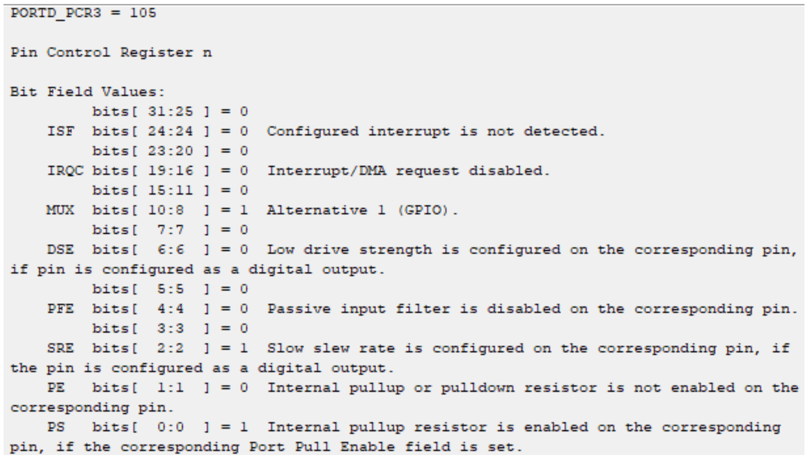
PORTD_PCR0	0x00000105
PORTD_PCR1	0x00000105
PORTD_PCR2	0x00000105
PORTD_PCR3	0x00000105
PORTD_PCR4	0x00000101
PORTD_PCR5	0x00000101
PORTD_PCR6	0x00000101
PORTD_PCR7	0x00000101

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



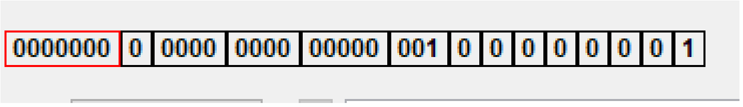
Bit Fields

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

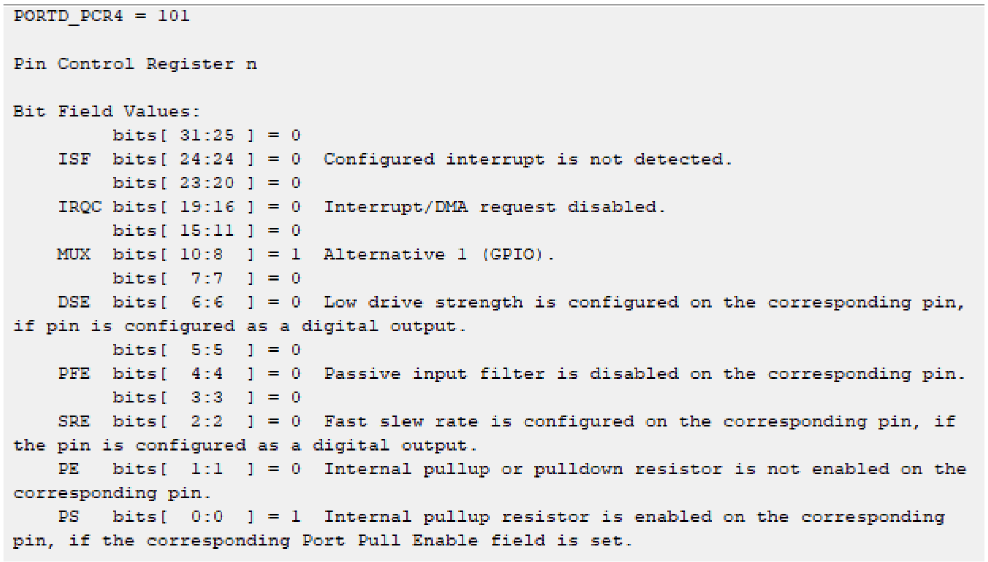


```
PORTD_PCR3 = 105

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.
```

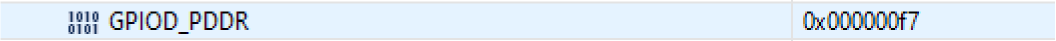
For PORTD_PCR4 to PORTD_PCR7 the configuration of the registers is the next.



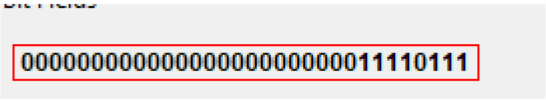
0000000	0	0000	0000	00000	001	0	0	0	0	0	0	0	1



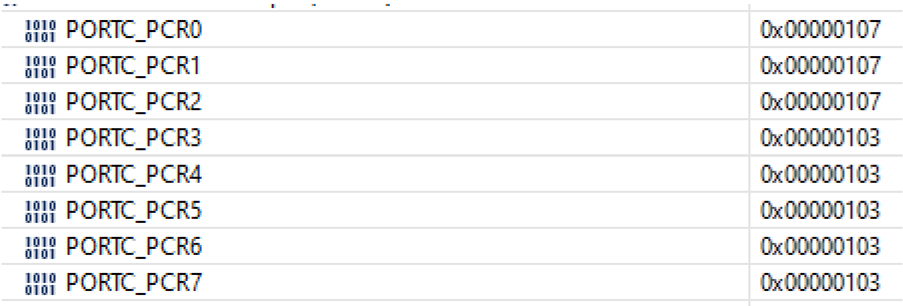
```
PORTD_PCR4 = 101

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

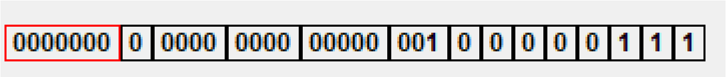


00000000000000000000000011110111

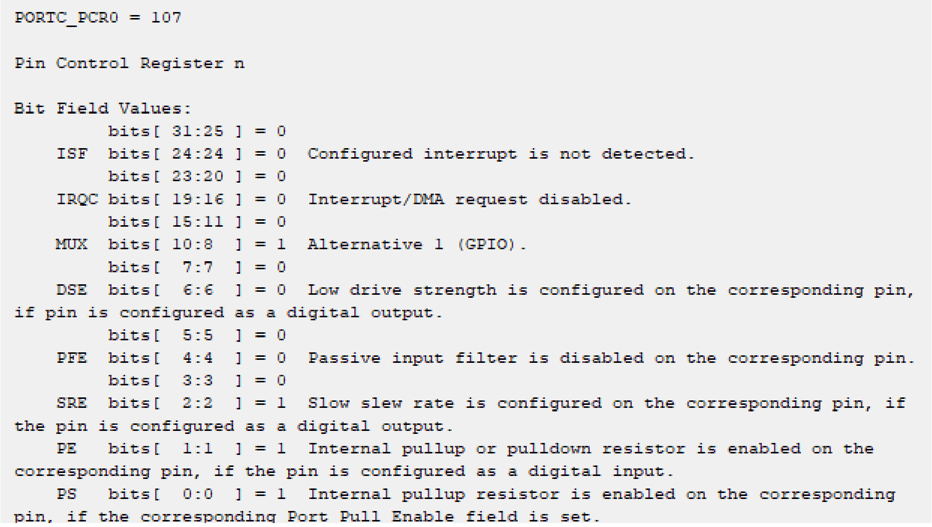


PORTC_PCR0	0x00000107
PORTC_PCR1	0x00000107
PORTC_PCR2	0x00000107
PORTC_PCR3	0x00000103
PORTC_PCR4	0x00000103
PORTC_PCR5	0x00000103
PORTC_PCR6	0x00000103
PORTC_PCR7	0x00000103

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

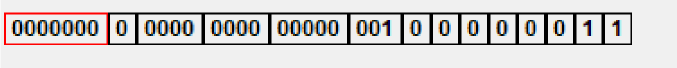


0000000 0 0000 0000 00000 001 0 0 0 0 0 1 1 1

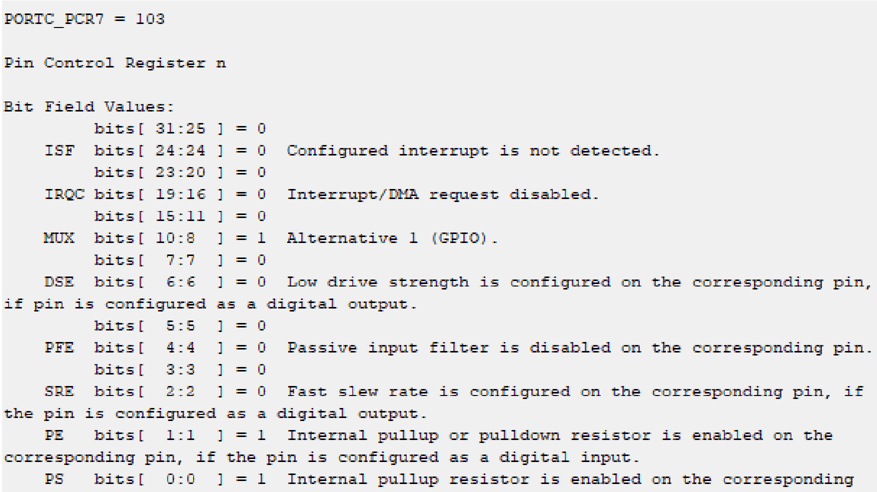


```
PORTC_PCR0 = 107

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.
```

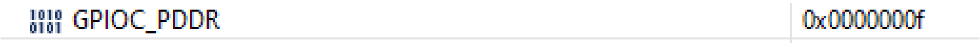
For PORTC_PCR3 to PORTC_PCR7 the configuration of the registers is the next.



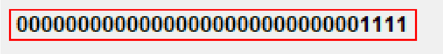
0000000 0 0000 0000 00000 001 0 0 0 0 0 0 1 1



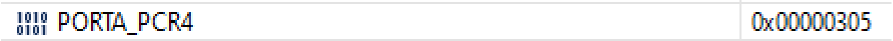
```
PORTC_PCR7 = 103

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
```



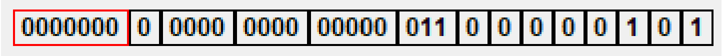
GPIOC_PDDR 0x0000000f



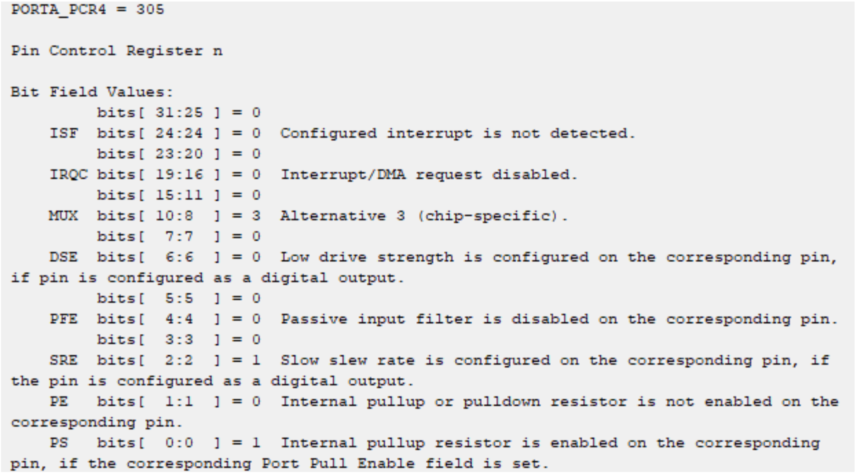
00000000000000000000000000001111



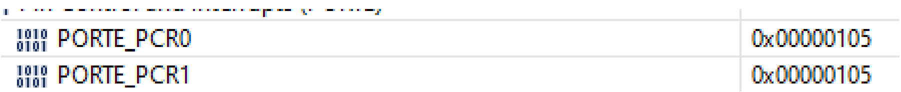
```
PORTA_PCR4    0x00000305
```



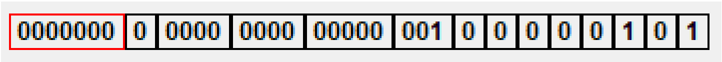
```
0000000 | 0 | 0000 | 0000 | 00000 | 011 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1
```



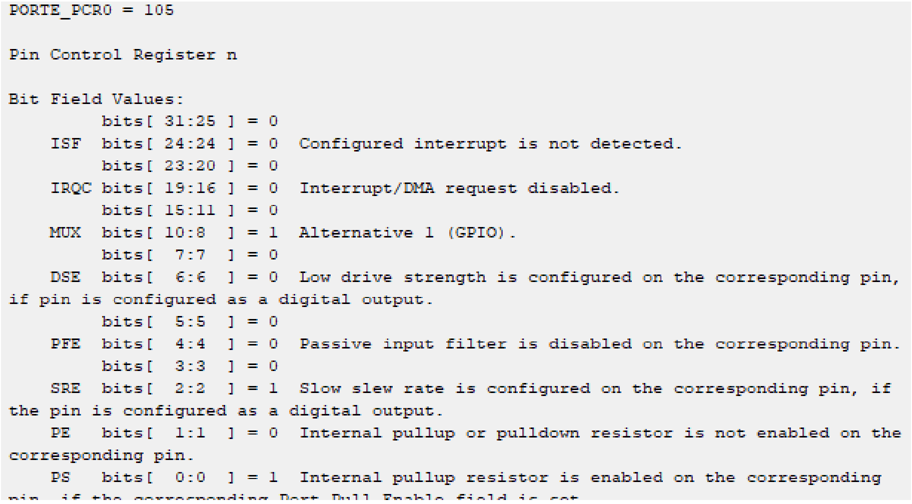
```
PORTA_PCR4 = 305

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  ] = 3  Alternative 3 (chip-specific).
         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.
```



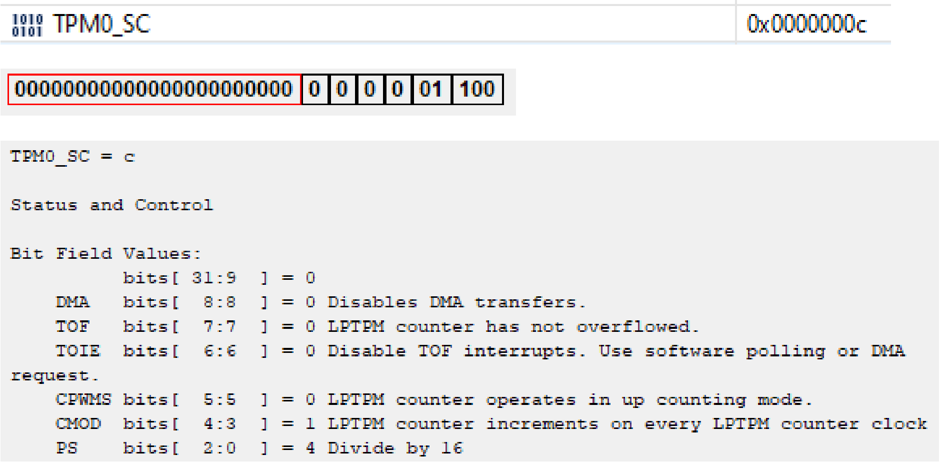
```
PORTE_PCR0    0x00000105
PORTE_PCR1    0x00000105
```



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



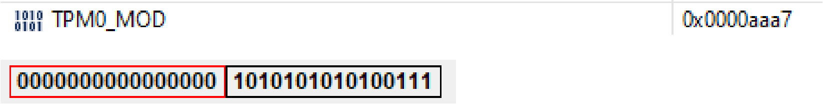
```
PORTE_PCR0 = 105

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.
```



TPM0_SC 0x0000000c

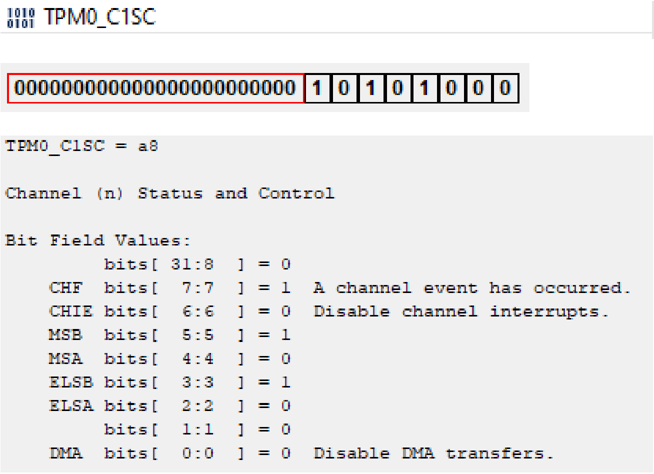
00000000000000000000000 | 0 | 0 | 0 | 0 | 01 | 100

```
TPM0_SC = c

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  ] = 4 Divide by 16
```



TPM0_MOD 0x0000aaa7

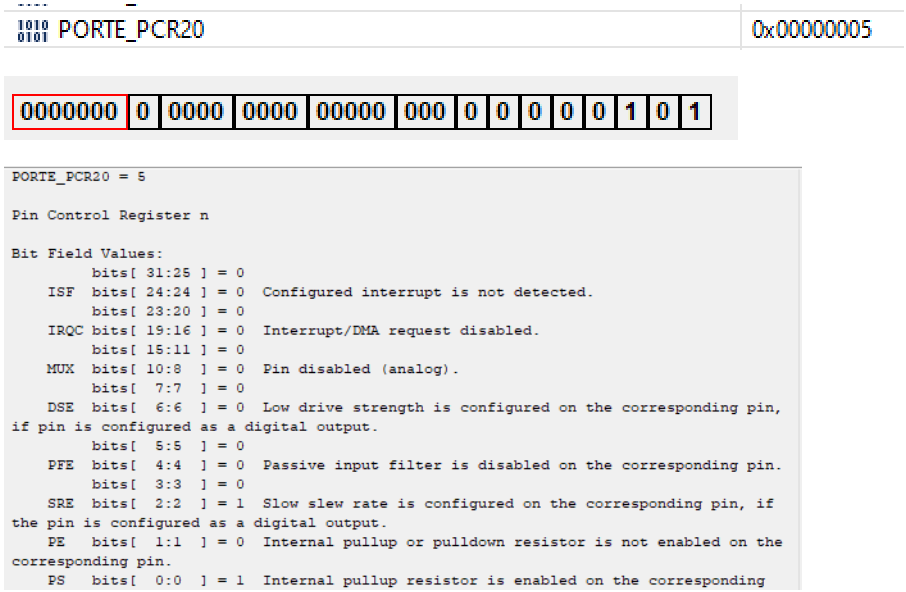
0000000000000000 | 1010101010100111



TPM0_C1SC 0x000000a8

000000000000000000000000 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0

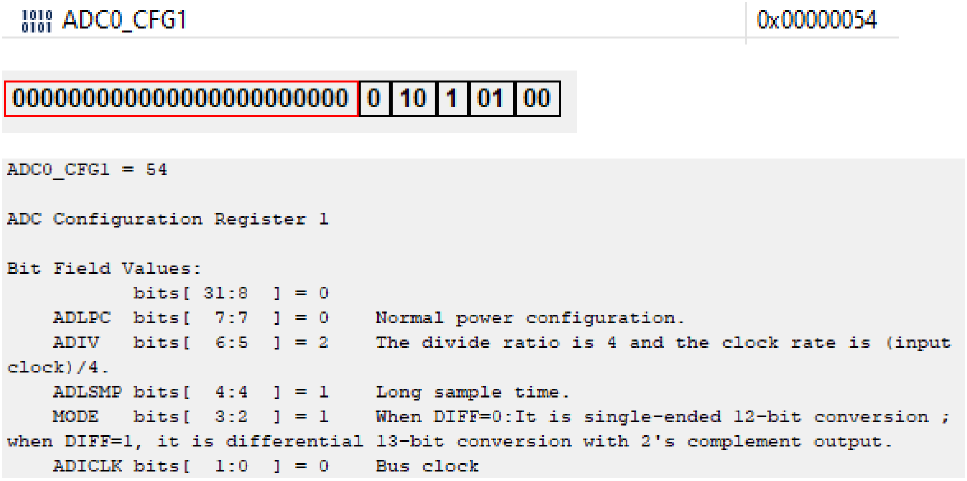
```
TPM0_C1SC = a8

Channel (n) Status and Control

Bit Field Values:
         bits[ 31:8  ] = 0
    CHF  bits[  7:7  ] = 1  A channel event has occurred.
    CHIE bits[  6:6  ] = 0  Disable channel interrupts.
    MSB  bits[  5:5  ] = 1
    MSA  bits[  4:4  ] = 0
    ELSB bits[  3:3  ] = 1
    ELSA bits[  2:2  ] = 0
         bits[  1:1  ] = 0
    DMA  bits[  0:0  ] = 0  Disable DMA transfers.
```



PORTE_PCR20 0x00000005

0000000 | 0 | 0000 | 0000 | 00000 | 000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1

```
PORTE_PCR20 = 5

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  ] = 0  Pin disabled (analog).
         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
```



ADC0_CFG1 0x00000054

000000000000000000000000 | 0 | 10 | 1 | 01 | 00

```
ADC0_CFG1 = 54

ADC Configuration Register 1

Bit Field Values:
           bits[ 31:8  ] = 0
    ADLPC  bits[  7:7  ] = 0    Normal power configuration.
    ADIV   bits[  6:5  ] = 2    The divide ratio is 4 and the clock rate is (input
clock)/4.
    ADLSMP bits[  4:4  ] = 1    Long sample time.
    MODE   bits[  3:2  ] = 1    When DIFF=0:It is single-ended 12-bit conversion ;
when DIFF=1, it is differential 13-bit conversion with 2's complement output.
    ADICLK bits[  1:0  ] = 0    Bus clock
```

Video:

https://youtu.be/cKKHyy2hGQA