Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Report week 7: Timers and Counters Management

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.

April 20, 2020

1. **Include some research about how the timers and counters operate in the corresponding section. Provide enough detail to demonstrate that you understand the basics and how the concepts relate**

   A timer is a specialized type of clock which is used to measure time intervals. A timer that counts from zero upwards for measuring time elapsed is often called a stopwatch. It is a device that counts down from a specified time interval and used to generate a time delay. A counter is a device that stores (and sometimes displays) the number of times a particular event or process occurred, with respect to a clock signal. It is used to count the events happening outside the microcontroller. In electronics, counters can be implemented quite easily using register-type circuits such as a flip-flop.

   There are some points that can help us differentiate a counter from a timer, which we will see next.

   Counters: The register is incremented considering 1 to 0 transition at its corresponding to an external input pin, maximum count rate is 1/24 of the oscillator frequency and a counter uses an external signal to count pulses.

   Timers: The register incremented for every machine cycle, maximum count rate is 1/12 of the oscillator frequency, a timer uses the frequency of the internal clock, and generates delay.

2. **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. An example is Fritzing, but there are others**

   There are not external connections for this laboratory since we're using the board's LED.

3. **Include some research about possible applications and timers in an application (especially in your final project).**

Considering our final project as the vending machine there are many applications for counters and timers on it. One of the main ones could be coin counting, trying to count bills and coins manually, especially in the quantities found in vending machines, is difficult to do. Time constraints and errors pretty much make that impossible. Counters may be purely mechanical or use electronic components. The machines typically provide a total count of all money, or count off specific batch sizes for wrapping and storage. A typical counter of presorted coins uses a bowl with flat spinning disc at the bottom to distribute coins around the bowl perimeter. An opening in the edge of the bowl is only wide enough to accept one coin at a time. Coins either pass through a light-beam counter, or are pushed through a spring-loaded cam that only accepts one coin at a time. Good standard for coin counters counting speed is 300 coins per minute.

On the other hand, an important application for timers on a vending machine might be one that turns on the “low power” on the machine after a certain amount of time of no activity in a proximity sensor. Vending energy efficiency is undoubtedly a value that sets the difference in the sector. When speaking of lighting consumption, the number of machines that include timers or sensors that turn off the light or stop the system once the machines reach an optimal temperature are on the rise.

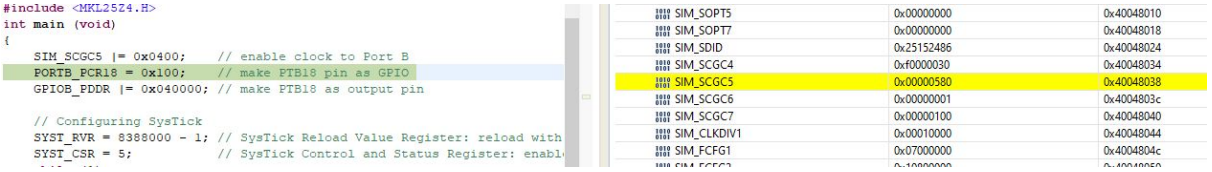
Part 1.

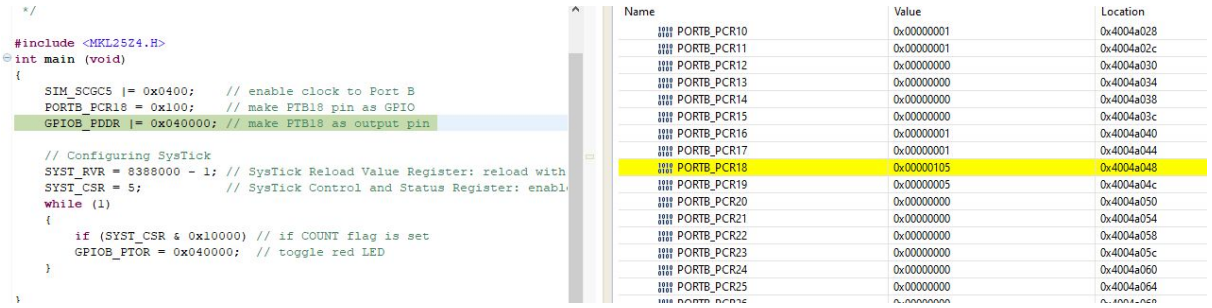
Make use of the SysTick timer to toggle the led in the KL25z board. A counter should be used and its value must be shifted 4 places to the right so that the changes can be slow enough to be visible in the LED of the board (connect the output to the red LED, PTB18)

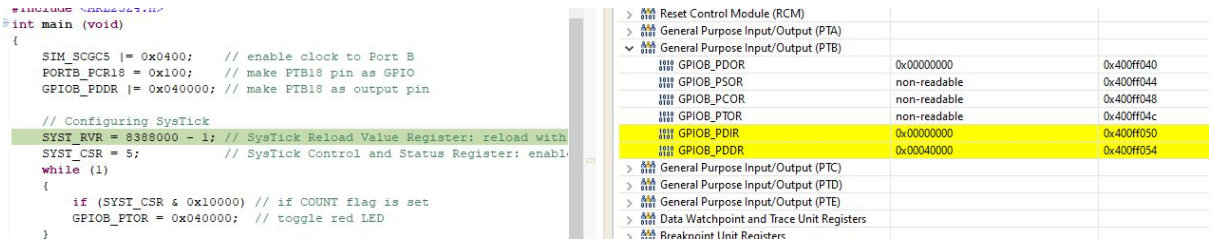
Code:

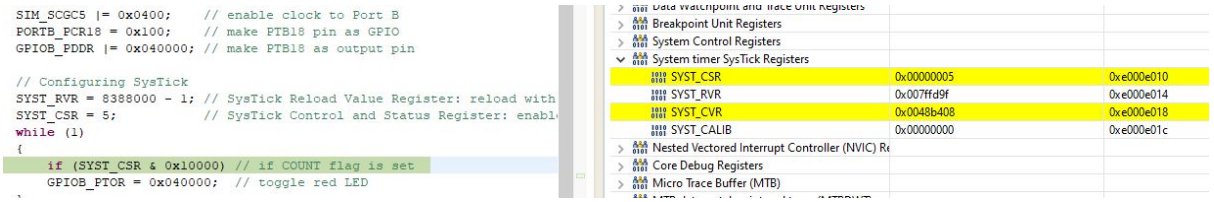
```
#include <MKL25Z4.H>
int main (void)
{
    SIM_SCGC5 |= 0x0400;     // enable clock to Port B
    PORTB_PCR18 = 0x100;     // make PTB18 pin as GPIO
    GPIOB_PDDR |= 0x040000; // make PTB18 as output pin

    // Configuring SysTick
    SYST_RVR = 8388000 - 1; // SysTick Reload Value Register: reload with number of clocks per 200 ms
    SYST_CSR = 5;           // SysTick Control and Status Register: enable it, no interrupt, use system clock
    while (1)
    {
        if (SYST_CSR & 0x10000) // if COUNT flag is set
        GPIOB_PTOR = 0x040000;  // toggle red LED
    }

}
```

Registers:









Part 2.

Toggling green LED using SysTick delay. This program should make use of the SysTick to generate one second delay to toggle the green LED. System clock is running at 41.94 MHz. SysTick is configure to count down from 41939 to give a 1 ms delay. For every 1000 delays (1 ms * 1000 = 1 sec), toggle the green LED once. The green LED is connected to PTB19.

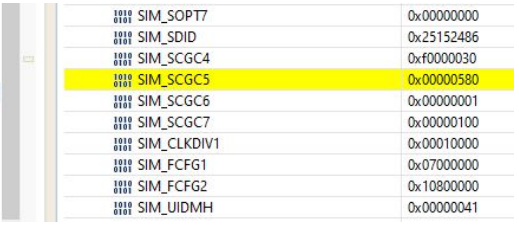
Code:

This code is used to toogle the green LED with a void called delays which is used with the sysclk in a way of doing an internal timer to delay there until the internal clk reaches the desire value.

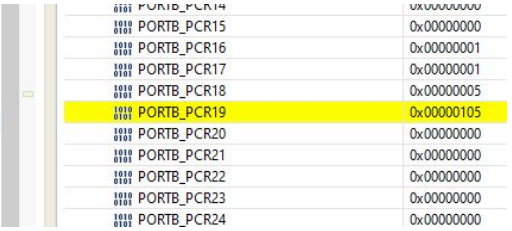
```
#include "MKL25Z4.h"

int main (void) {

    void delayMs(int n);
    SIM_SCGC5 |= 0x400;     // enable clock to Port B
    PORTB_PCR19 = 0x100;    // make PTB19 pin as GPIO
    GPIOB_PDDR |= 0x080000; // make PTB19 as output pin
    while (1)
    {
        delayMs(1000);          // delay 1000 ms
        GPIOB_PTOR = 0x080000;  // toggle green LED
    }
}

void delayMs(int n)
{
    int i;
    SYST_RVR = 41940 - 1;
    SYST_CSR = 0x5;                     // Enable the timer and choose sysclk as the clock source
    for(i = 0; i < n; i++)
    {
        while((SYST_CSR & 0x10000) == 0) // wait until the COUTN flag is set
        {
        }
    }
    SYST_CSR = 0;   // Stop the timer (Enable = 0)
}
```

Registers:

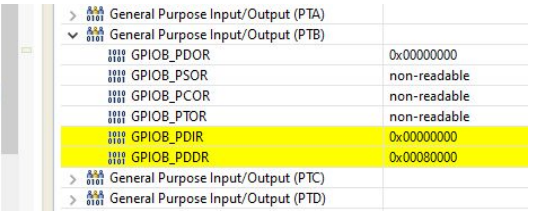
```
int main (void) {

    void delayMs(int n);
    SIM_SCGC5 |= 0x400;     // enable clock to Port B
    PORTB_PCR19 = 0x100;    // make PTB19 pin as GPIO
    GPIOB_PDDR |= 0x080000; // make PTB19 pin as output pin
    while (1)
    {
        delayMs(1000);          // delay 1000 ms
        GPIOB_PTOR = 0x080000;  // toggle green LED
    }
```




```
int main (void) {

    void delayMs(int n);
    SIM_SCGC5 |= 0x40[n = No value available] clock to Port B
    PORTB_PCR19 = 0x100;    // make PTB19 pin as GPIO
    GPIOB_PDDR |= 0x080000; // make PTB19 pin as output pin
    while (1)
    {
        delayMs(1000);          // delay 1000 ms
        GPIOB_PTOR = 0x080000;  // toggle green LED
```



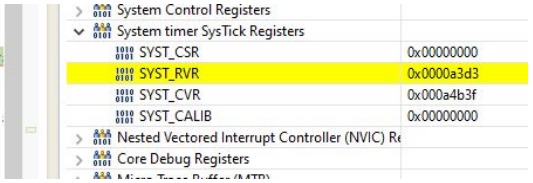

```
    PORTB_PCR19 = 0x100;    // make PTB19 pin as GPIO
    GPIOB_PDDR |= 0x080000; // make PTB19 pin as output pin
    while (1)
    {
        delayMs(1000);          // delay 1000 ms
        GPIOB_PTOR = 0x080000;  // toggle green LED
    }
}

void delayMs(int n)
```



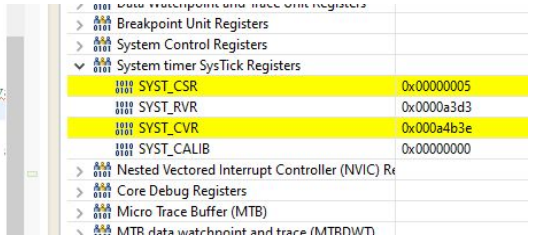

```
    {
        delayMs(1000);          // delay 1000 ms
        GPIOB_PTOR = 0x080000;  // toggle gree
    }
```

The word 'ms'

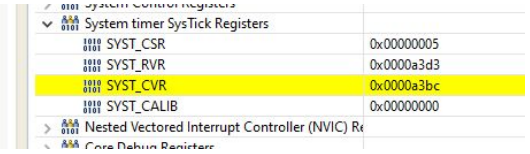
```
    int i;
    SYST_RVR = 41940 - 1;
    SYST_CSR = 0x5;                     // Enable the timer and choose sy
    for(i = 0; i < n; i++)
    {
        while((SYST_CSR & 0x10000) == 0) // wait until the COUNT flag is
        {
        }
```

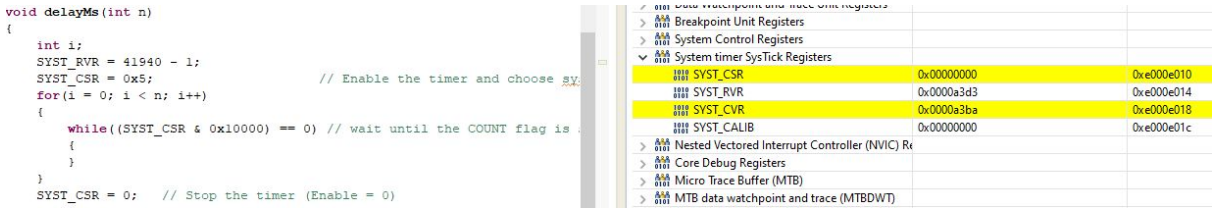



```
void delayMs(int n)
{
    int i;
    SYST_RVR = 41940 - 1;
    SYST_CSR = 0x5;                     // Enable the timer and choose sy
    for(i = 0; i < n; i++)
    {
        while((SYST_CSR & 0x10000) == 0) // wait until the COUNT flag is
        {
        }
    }
    SYST_CSR = 0;   // Stop the timer (Enable = 0)
```




```
    int i;
    SYST_RVR = 41940 - 1;
    SYST_CSR = 0x5;                     // Enable the timer and choose sy
    for(i = 0; i < n; i++)
    {
        while((SYST_CSR & 0x10000) == 0) // wait until the COUNT flag is
        {
```





```
void delayMs(int n)
{
    int i;
    SYST_RVR = 41940 - 1;
    SYST_CSR = 0x5;                    // Enable the timer and choose sy
    for(i = 0; i < n; i++)
    {
        while((SYST_CSR & 0x10000) == 0) // wait until the COUNT flag is
        {
        }
    }
    SYST_CSR = 0;   // Stop the timer (Enable = 0)
```

Breakpoint Unit Registers		
System Control Registers		
System timer SysTick Registers		
SYST_CSR	0x00000000	0xe000e010
SYST_RVR	0x0000a3d3	0xe000e014
SYST_CVR	0x0000a3ba	0xe000e018
SYST_CALIB	0x00000000	0xe000e01c
Nested Vectored Interrupt Controller (NVIC) Re		
Core Debug Registers		
Micro Trace Buffer (MTB)		
MTB data watchpoint and trace (MTBDWT)		

```
        delayMs(1000);          // delay 1000 ms
        GPIOB_PTOR = 0x080000;  // toggle green LED
    }
}

void delayMs(int n)
{
    int i;
```

General Purpose Input/Output (PTB)		
GPIOB_PDOR	0x00080000	0x400ff040
GPIOB_PSOR	non-readable	0x400ff044
GPIOB_PCOR	non-readable	0x400ff048
GPIOB_PTOR	non-readable	0x400ff04c
GPIOB_PDIR	0x00080000	0x400ff050
GPIOB_PDDR	0x00080000	0x400ff054

Part 3.

Toggling blue LED using TPM0 delay (prescaler). This program has to make use of the TPM0 to generate maximal delay to toggle the blue LED. MCGFLLCLK (41.94 MHz) is used as timer counter clock. Prescaler must be set to divide by 128 and the Modulo register to 65,535. The timer counter overflows at 41.94 MHz / 128 / 65,536 = 5.0 Hz.

The blue LED is connected to PTD1.

Remember the steps for configuring the counter are

1) enable the clock to TPMx module in SIM_SCGC6,

2) select the clock source for timer counter in SIM_SOPT2,

3) disable timer while the configuration is being modified,

4) set the mode as up-counter timer mode with TPMx_SC register,

5) load TPMx_MOD register with proper value,

6) clear TOF flag,

7) enable timer,

8) wait for TOF flag to go HIGH.

Answer the following questions

(a) Show time delay calculation for the program

(b) calculate the largest delay size without prescaler

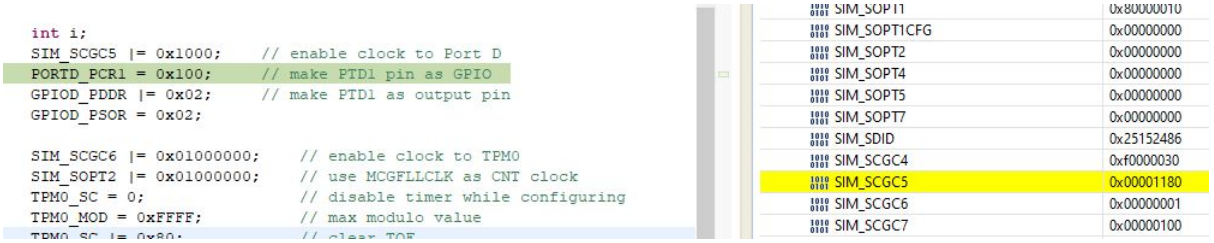
(c) Find the TPMx_MOD value to generate a delay of 0.1 second. Use the prescaler of 128.

Code:

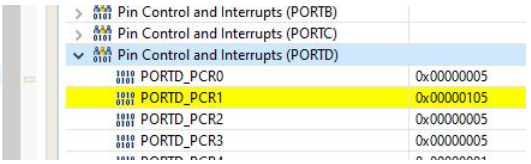
This code uses internal flags of the clock in order to delay the time that we desire and toggle the led every each time that we need.

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

#include "MKL25Z4.h"

int main (void)
{
    int i;
    SIM_SCGC5 |= 0x1000;     // enable clock to Port D
    PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
    GPIOD_PDDR |= 0x02;      // make PTD1 as output pin


    SIM_SCGC6 |= 0x01000000;     // enable clock to TPM0
    SIM_SOPT2 |= 0x01000000;     // use MCGFLLCLK as CNT clock
    TPM0_SC = 0;                 // disable timer while configuring
    TPM0_MOD = 0xFFFF;           // max modulo value
    TPM0_SC |= 0x80;             // clear TOF
    TPM0_SC |= 0x08;             // enable timer free-running mode

    while (1) {
        for(i = 0; i < 320; i++) {          // repeat timeout for 320 times
            while((TPM0_SC & 0x80) == 0) {} // wait until the TOF is set
            TPM0_SC |= 0x80;                // clear TOF
        }
        GPIOD_PTOR = 0x02;                  // toggle blue LED

    }
}
```

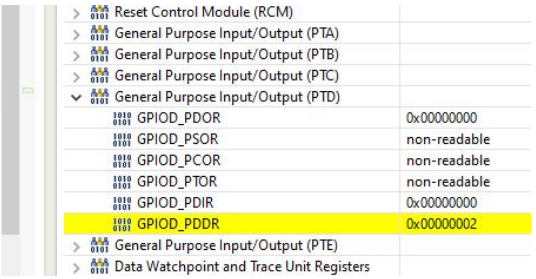
Registers:



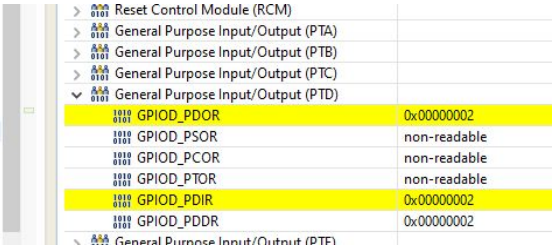
```
int i;
SIM_SCGC5 |= 0x1000;     // enable clock to Port D
PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
GPIOD_PDDR |= 0x02;      // make PTD1 as output pin
GPIOD_PSOR = 0x02;

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



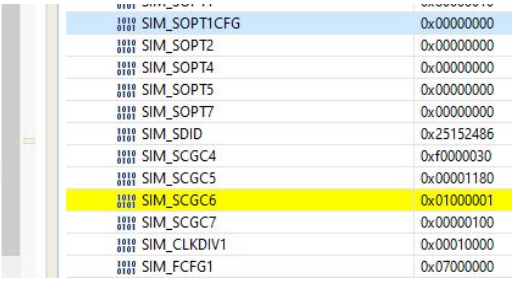

```
int i;
SIM_SCGC5 |= 0x1000;     // enable clock to Port D
PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
GPIOD_PDDR |= 0x02;      // make PTD1 as output pin
GPIOD_PSOR = 0x02;

SIM_SCGC6 |= 0x01000000;     // enable clock to TPM0
SIM_SOPT2 |= 0x01000000;     // use MCGFLLCLK as CNT clock
TPM0_SC = 0;                 // disable timer while configuring
TPM0_MOD = 0xFFFF;           // max modulo value
TPM0_SC |= 0x80;             // clear TOF
TPM0_SC |= 0x08;             // enable timer free-running mode

while (1) {
```



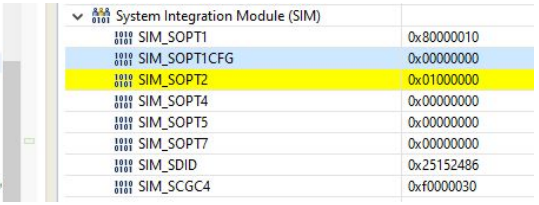

```
int i;
SIM_SCGC5 |= 0x1000;     // enable clock to Port D
PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
GPIOD_PDDR |= 0x02;      // make PTD1 as output pin
GPIOD_PSOR = 0x02;

SIM_SCGC6 |= 0x01000000;     // enable clock to TPM0
SIM_SOPT2 |= 0x01000000;     // use MCGFLLCLK as CNT clock
TPM0_SC = 0;                 // disable timer while configuring
TPM0_MOD = 0xFFFF;           // max modulo value
TPM0_SC |= 0x80;             // clear TOF
TPM0_SC |= 0x08;             // enable timer free-running mode
```



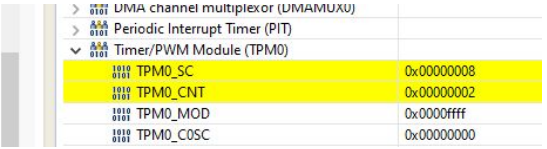

```
int i;
SIM_SCGC5 |= 0x1000;     // enable clock to Port D
PORTD_PCR1 = 0x100;      // make PTD1 pin as GPIO
GPIOD_PDDR |= 0x02;      // make PTD1 as output pin
GPIOD_PSOR = 0x02;

SIM_SCGC6 |= 0x01000000;     // enable clock to TPM0
SIM_SOPT2 |= 0x01000000;     // use MCGFLLCLK as CNT clock
TPM0_SC = 0;                 // disable timer while configuring
TPM0_MOD = 0xFFFF;           // max modulo value
TPM0_SC |= 0x80;             // clear TOF
TPM0_SC |= 0x08;             // enable timer free-running mode
```




```
SIM_SCGC6 |= 0x01000000;     // enable clock to TPM0
SIM_SOPT2 |= 0x01000000;     // use MCGFLLCLK as CNT clock
TPM0_SC = 0;                 // disable timer while configuring
TPM0_MOD = 0xFFFF;           // max modulo value
TPM0_SC |= 0x80;             // clear TOF
TPM0_SC |= 0x08;             // enable timer free-running mode

while (1) {
    for(i = 0; i < 320; i++) {          // repeat timeout for 320 tim
```




```
SIM_SCGC6 |= 0x01000000;     // enable clock to TPM0
SIM_SOPT2 |= 0x01000000;     // use MCGFLLCLK as CNT clock
TPM0_SC = 0;                 // disable timer while configuring
TPM0_MOD = 0xFFFF;           // max modulo value
TPM0_SC |= 0x80;             // clear TOF
TPM0_SC |= 0x08;             // enable timer free-running mode
```



Part 4.

Longer time interval. Toggling blue LED using TPM0 delay. The program must use TPM0 to generate long delay to toggle the blue LED. MCGIRCLK (32.768 kHz) is used as timer counter clock. Prescaler is set to divided by 4 and the Modulo register is set to 40,959. The timer counter overflows at 32,768 Hz / 40,960 / 4 = 0.2 Hz. The blue LED is connected to PTD1.

Code:

This program also used the internal clock of the microcontroller to delay the time that we want to. This code is used in order that the delay last longer.

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

    PORTD_PCR1 = 0x100;           // make PTD1 pin as GPIO
    GPIOD_PDDR |= 0x02;           // make PTD1 as output pin


    SIM_SCGC6 |= 0x01000000;      // enable clock to TPM0
    SIM_SOPT2 |= 0x03000000;      // use MCGIRCLK as timer counter clock

    TPM0_SC = 0;                  // disable timer while configuring
    TPM0_SC = 0x02;               // prescaler 4
    TPM0_MOD = 40960 - 1;         // modulo value
    TPM0_SC |= 0x80;              // clear TOF
    TPM0_SC |= 0x08;              // enable timer free-running mode

    while (1) {
            while((TPM0_SC & 0x80)==0){}     // wait until the TOF is set
            TPM0_SC |= 0x80;                 // clear TOF
            GPIOD_PTOR = 0x02;                // toggle blue LED

        }
}
```

Registers:

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

    PORTD_PCR1 = 0x100; /* make PTD1 pin as GPIO */
    GPIOD_PDDR |= 0x02; /* make PTD1 as output pin */
    //GPIOD-_PSOR = 0x02;

    SIM_SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM_SOPT2 |= 0x03000000; /* use MCGIRCLK as timer counter clock */

    TPM0_SC = 0; /* disable timer while configuring */
    TPM0_SC = 0x02; /* prescaler /4 */
    TPM0_MOD = 40960 - 1; /* modulo value */
    TPM0_SC |= 0x80; /* clear TOF */
    TPM0_SC |= 0x08; /* enable timer free-running mode */
```

Name	Value	Location
General Purpose Input/Output (PTA)		
General Purpose Input/Output (PTB)		
General Purpose Input/Output (PTC)		
General Purpose Input/Output (PTD)		
GPIOD_PDOR	0x00000000	0x400ff0c0
GPIOD_PSOR	non-readable	0x400ff0c4
GPIOD_PCOR	non-readable	0x400ff0c8
GPIOD_PTOR	non-readable	0x400ff0cc
GPIOD_PDIR	0x00000000	0x400ff0d0
GPIOD_PDDR	0x00000002	0x400ff0d4
General Purpose Input/Output (PTE)		
Data Watchpoint and Trace Unit Registers		
Breakpoint Unit Registers		

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

    PORTD_PCR1 = 0x100; /* make PTD1 pin as GPIO */
    GPIOD_PDDR |= 0x02; /* make PTD1 as output pin */
    //GPIOD-_PSOR = 0x02;

    SIM_SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM_SOPT2 |= 0x03000000; /* use MCGIRCLK as timer counter clock */

    TPM0_SC = 0; /* disable timer while configuring */
    TPM0_SC = 0x02; /* prescaler /4 */
    TPM0_MOD = 40960 - 1; /* modulo value */
    TPM0_SC |= 0x80; /* clear TOF */
```

Name	Value	Location
SIM_SOPT2	0x00000000	0x40048004
SIM_SOPT4	0x00000000	0x4004800c
SIM_SOPT5	0x00000000	0x40048010
SIM_SOPT7	0x00000000	0x40048018
SIM_SDID	0x25152486	0x40048024
SIM_SCGC4	0xf0000030	0x40048034
SIM_SCGC5	0x00001180	0x40048038
SIM_SCGC6	0x01000001	0x4004803c
SIM_SCGC7	0x00000100	0x40048040
SIM_CLKDIV1	0x00010000	0x40048044
SIM_FCFG1	0x07000000	0x4004804c
SIM_FCFG2	0x10800000	0x40048050
SIM_UIDMH	0x00000041	0x40048058

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

    PORTD_PCR1 = 0x100; /* make PTD1 pin as GPIO */
    GPIOD_PDDR |= 0x02; /* make PTD1 as output pin */
    //GPIOD-_PSOR = 0x02;

    SIM_SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM_SOPT2 |= 0x03000000; /* use MCGIRCLK as timer counter clock */

    TPM0_SC = 0; /* disable timer while configuring */
    TPM0_SC = 0x02; /* prescaler /4 */
    TPM0_MOD = 40960 - 1; /* modulo value */
    TPM0_SC |= 0x80; /* clear TOF */
    TPM0_SC |= 0x08; /* enable timer free-running mode */

    while (1) {
```

Name	Value	Location
Secure Real Time Clock (RTC)		
Low Power Timer (LPTMR0)		
Touch sense input (TSI0)		
System Integration Module (SIM)		
SIM_SOPT1	0x80000010	0x40047000
SIM_SOPT1CFG	0x00000000	0x40047004
SIM_SOPT2	0x03000000	0x40048004
SIM_SOPT4	0x00000000	0x4004800c
SIM_SOPT5	0x00000000	0x40048010
SIM_SOPT7	0x00000000	0x40048018
SIM_SDID	0x25152486	0x40048024
SIM_SCGC4	0xf0000030	0x40048034
SIM_SCGC5	0x00001180	0x40048038

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

    PORTD_PCR1 = 0x100; /* make PTD1 pin as GPIO */
    GPIOD_PDDR |= 0x02; /* make PTD1 as output pin */
    //GPIOD-_PSOR = 0x02;

    SIM_SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM_SOPT2 |= 0x03000000; /* use MCGIRCLK as timer counter clock */

    TPM0_SC = 0; /* disable timer while configuring */
    TPM0_SC = 0x02; /* prescaler /4 */
    TPM0_MOD = 40960 - 1; /* modulo value */
    TPM0_SC |= 0x80; /* clear TOF */
    TPM0_SC |= 0x08; /* enable timer free-running mode */
```

Name	Value	Location
Periodic Interrupt Timer (PIT)		
Timer/PWM Module (TPM0)		
TPM0_SC	0x00000002	0x40038000
TPM0_CNT	0x00000000	0x40038004
TPM0_MOD	0x0000ffff	0x40038008
TPM0_C0SC	0x00000000	0x4003800c
TPM0_C0V	0x00000000	0x40038010
TPM0_C1SC	0x00000000	0x40038014
TPM0_C1V	0x00000000	0x40038018
TPM0_C2SC	0x00000000	0x4003801c
TPM0_C2V	0x00000000	0x40038020
TPM0_C3SC	0x00000000	0x40038024
TPM0_C3V	0x00000000	0x40038028

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

    PORTD_PCR1 = 0x100; /* make PTD1 pin as GPIO */
    GPIOD_PDDR |= 0x02; /* make PTD1 as output pin */
    //GPIOD-_PSOR = 0x02;

    SIM_SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM_SOPT2 |= 0x03000000; /* use MCGIRCLK as timer counter clock */

    TPM0_SC = 0; /* disable timer while configuring */
    TPM0_SC = 0x02; /* prescaler /4 */
    TPM0_MOD = 40960 - 1; /* modulo value */
    TPM0_SC |= 0x80; /* clear TOF */
    TPM0_SC |= 0x08; /* enable timer free-running mode */

    while (1) {
            while((TPM0_SC & 0x80)==0){}    // wait until the TOF is set
```

Name	Value	Location
Periodic Interrupt Timer (PIT)		
Timer/PWM Module (TPM0)		
TPM0_SC	0x00000002	0x40038000
TPM0_CNT	0x00000000	0x40038004
TPM0_MOD	0x00009fff	0x40038008
TPM0_C0SC	0x00000000	0x4003800c
TPM0_C0V	0x00000000	0x40038010
TPM0_C1SC	0x00000000	0x40038014
TPM0_C1V	0x00000000	0x40038018
TPM0_C2SC	0x00000000	0x4003801c
TPM0_C2V	0x00000000	0x40038020
TPM0_C3SC	0x00000000	0x40038024
TPM0_C3V	0x00000000	0x40038028

```
    PORTD_PCR1 = 0x100; /* make PTD1 pin as GPIO */
    GPIOD_PDDR |= 0x02; /* make PTD1 as output pin */
    //GPIOD-_PSOR = 0x02;

    SIM_SCGC6 |= 0x01000000; /* enable clock to TPM0 */
    SIM_SOPT2 |= 0x03000000; /* use MCGIRCLK as timer counter clock */

    TPM0_SC = 0; /* disable timer while configuring */
    TPM0_SC = 0x02; /* prescaler /4 */
    TPM0_MOD = 40960 - 1; /* modulo value */
    TPM0_SC |= 0x80; /* clear TOF */
    TPM0_SC |= 0x08; /* enable timer free-running mode */

    while (1) {
            while((TPM0_SC & 0x80)==0){}     // wait until the TOF is set
            TPM0_SC |= 0x80;                 // clear TOF
            GPIOD_PTOR = 0x02;               // toggle blue LED

        }
}
```

Name	Value	Location
Periodic Interrupt Timer (PIT)		
Timer/PWM Module (TPM0)		
TPM0_SC	0x0000000a	0x40038000
TPM0_CNT	0x00000000	0x40038004
TPM0_MOD	0x00009fff	0x40038008
TPM0_C0SC	0x00000000	0x4003800c
TPM0_C0V	0x00000000	0x40038010
TPM0_C1SC	0x00000000	0x40038014
TPM0_C1V	0x00000000	0x40038018
TPM0_C2SC	0x00000000	0x4003801c
TPM0_C2V	0x00000000	0x40038020
TPM0_C3SC	0x00000000	0x40038024
TPM0_C3V	0x00000000	0x40038028