El figa esta bien sabroso ñam ñam

Reporte Practica 2 - Timers and Counters

Diseño en sistemas de chip

Diego Armando Limón de León A01638247

Jose Miguel Figarola Prado A01632557

Giancarlo Franco Carrillo A01638108

Tecnológico de Monterrey

20 de abril del 2021

Timers and Counters

A **timer** is a specialized type of clock that measures time intervals, when it counts from zero upwards, it is often called a stopwatch. It is important to generate time delays. Meanwhile a **counter** is a device that stores and in some cases displays the number of times a certain task was performed with the correlation of a specific clock.

Timer	Counter
The register incremented for every machine cycle.	The register is incremented considering 1 to 0 transition at its corresponding to an external input pin (T0, T1).
Maximum count rate is 1/12 of the oscillator frequency.	Maximum count rate is 1/24 of the oscillator frequency.
A timer uses the frequency of the internal clock, and generates delay.	A counter uses an external signal to count pulses.

Applications

Timers:

1. Clocks.
2. Watches.
3. Phones.
4. Reference signals.
5. Particular delays.
6. Triggering flip flops.
7. Road Lights setters.
8. Switching off the TV or radio automatically..

Counters:

1. Calculator.
2. Processors.
3. Real-Time Clock.
4. Washing Machines.
5. Microwaves.
6. Set time in camera for a picture.
7. Flashing light indicators in cars.
8. Car parking control.
9. Keyboard controller.
10. Machine moving control.

Activity and Analisis Development

In this lab, we will see how to configure and operate the timers and counters of the ARM M0 MCU contained in the KL25Z board. We won't need any extra hardware for this lab, only the board and we will display the results in our board's LEDs.

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

Include the code for each of the sections in the lab. The code should be commented and the report should include a short description of each function and how it works, including images of the registers that have been configured for enabling the different functionalities in your code.

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

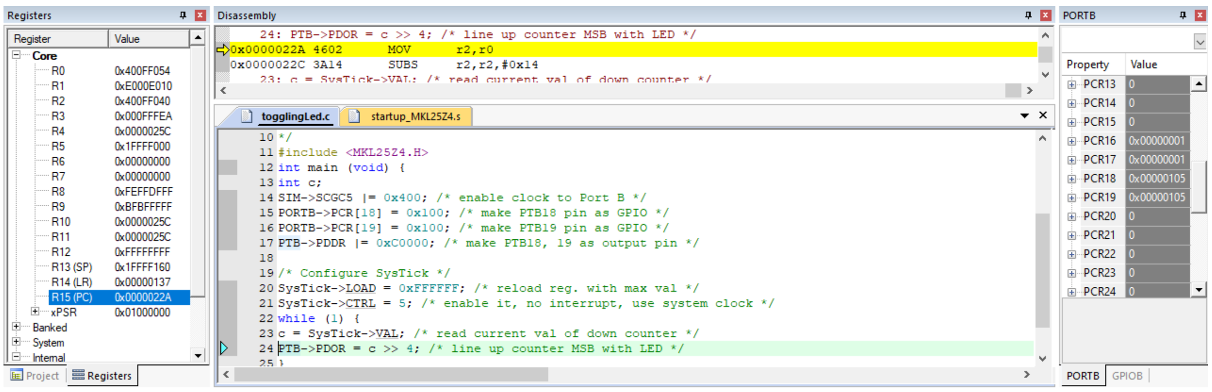
Include some research about possible applications and timers in an application (especially in your final project).

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).

Output Analisis: This first code uses de SysTick timer to toggle the led from the KL25z board and it shows the 3 different colors (blude, red, yellow). By shifting the counter 4 places to the right, the blinky motion will be slower so it is perceptible to the human eye. The SysTick timer has 24 bits and we can define the ports PTB18 and PTB19 as GPIO. The SysTick value should be at its maximum value, then it is enabled and uses the system clock.

```c
/* Toggling LEDs using SysTick counter

This program let the SysTick counter run freely and dumps the counter values to the tri-color LEDs continuously.

The counter value is shifted 4 places to the right so that the changes of LEDs will be slow enough to be visible.

SysTick counter has 24 bits.
The red LED is connected to PTB18.
The green LED is connected to PTB19.
*/

#include <MKL25Z4.H>
void delay1ms(void);
int main (void) {
int c;
SIM->SCGC5 |= 0x400; /* enable clock to Port B */
PORTB->PCR[18] = 0x100; /* make PTB18 pin as GPIO */
PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
PTB->PDDR |= 0xC0000; /* make PTB18, 19 as output pin */

/* Configure SysTick */
SysTick->LOAD = 0xFFFFFF; /* reload reg. with max val */
SysTick->CTRL = 5; /* enable it, no interrupt, use system clock */
while (1) {
c = SysTick->VAL; /* read current val of down counter */
PTB->PDOR = c >> 4; /* line up counter MSB with LED */
}
}

void delay1ms(void) {
SysTick->LOAD = 41939;
SysTick->CTRL = 0x5; /* Enable the timer and choose sysclk as the clock source */
while((SysTick->CTRL & 0x10000) == 0) /* wait until the COUNT flag is set */ { }
SysTick->CTRL = 0; /* Stop the timer (Enable = 0) */ }
```


Registers

Register	Value
Core	
R0	0x400FF054
R1	0xE000E010
R2	0x400FF040
R3	0x000FFFEA
R4	0x0000025C
R5	0x1FFFF000
R6	0x00000000
R7	0x00000000
R8	0xFEFFDFFF
R9	0xBFBFFFFF
R10	0x0000025C
R11	0x0000025C
R12	0xFFFFFFFF
R13 (SP)	0x1FFFF160
R14 (LR)	0x00000137
R15 (PC)	0x0000022A
xPSR	0x01000000
Banked	
System	
Internal	

Disassembly

```
    24: PTB->PDOR = c >> 4; /* line up counter MSB with LED */
0x0000022A 4602      MOV      r2,r0
0x0000022C 3A14      SUBS     r2,r2,#0x14
    23: c = SysTick->VAL; /* read current val of down counter */
```

togglingLed.c | startup_MKL25Z4.s

```c
*/
#include <MKL25Z4.H>
int main (void) {
int c;
SIM->SCGC5 |= 0x400; /* enable clock to Port B */
PORTB->PCR[18] = 0x100; /* make PTB18 pin as GPIO */
PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
PTB->PDDR |= 0xC0000; /* make PTB18, 19 as output pin */

/* Configure SysTick */
SysTick->LOAD = 0xFFFFFF; /* reload reg. with max val */
SysTick->CTRL = 5; /* enable it, no interrupt, use system clock */
while (1) {
c = SysTick->VAL; /* read current val of down counter */
PTB->PDOR = c >> 4; /* line up counter MSB with LED */
}
```

PORTB

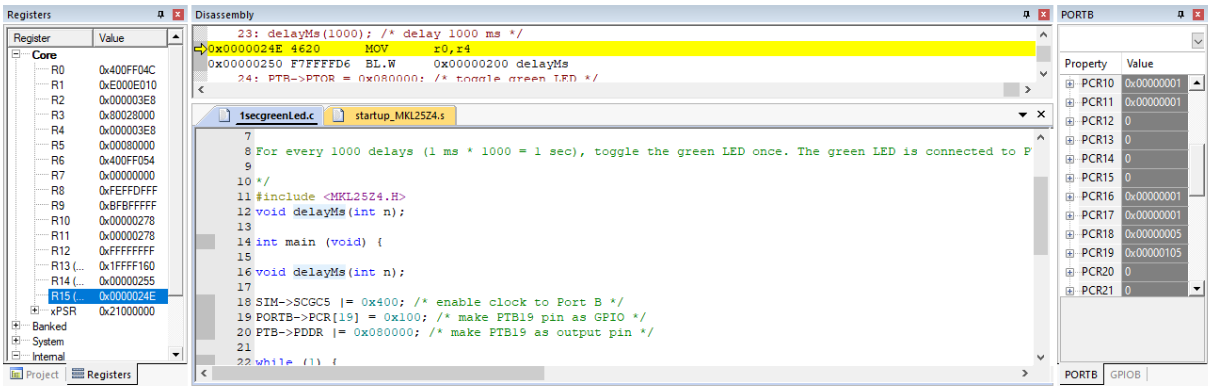
Property	Value
PCR13	0
PCR14	0
PCR15	0
PCR16	0x00000001
PCR17	0x00000001
PCR18	0x00000105
PCR19	0x00000105
PCR20	0
PCR21	0
PCR22	0
PCR23	0
PCR24	0

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 configured 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.

Output Analisis: The second code has the intentions to toggle the green LED every second, as the first programme, we are using the system clock that runs at 41.94 MHz, with that said, it requires a count down from 41939 to have a 1 ms delay. That means, with every 1000 delays we get a 1 second delay and the LED is toggled once. The PTB19 is made as a GPIO.

```c
/* Toggling green LED using SysTick delay

* This program uses 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.

*/

#include <MKL25Z4.H>

int main (void) {

void delayMs(int n);

SIM->SCGC5 |= 0x400; /* enable clock to Port B */
PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
PTB->PDDR |= 0x080000; /* make PTB19 as output pin */

while (1) {
delayMs(1000); /* delay 1000 ms */
PTB->PTOR = 0x080000; /* toggle green LED */
} }

void delayMs(int n) {
int i;
SysTick->LOAD = 41940 - 1;
SysTick->CTRL = 0x5; /* Enable the timer and choose sysclk as the clock source */

for(i = 0; i < n; i++) {
while((SysTick->CTRL & 0x10000) == 0)
 /* wait until the COUTN flag is set */
{ }
}
SysTick->CTRL = 0;
/* Stop the timer (Enable = 0) */
}

```



Register	Value
Core	
R0	0x400FF04C
R1	0xE000E010
R2	0x000003E8
R3	0x80028000
R4	0x000003E8
R5	0x00080000
R6	0x400FF054
R7	0x00000000
R8	0xFEFFDFFF
R9	0xBFBFFFFF
R10	0x00000278
R11	0x00000278
R12	0xFFFFFFFF
R13 (...	0x1FFFF160
R14 (...	0x00000255
R15 (...	0x0000024E
xPSR	0x21000000
Banked	
System	
Internal	

```
    23: delayMs(1000); /* delay 1000 ms */
0x0000024E 4620      MOV      r0,r4
0x00000250 F7FFFFD6  BL.W     0x00000200 delayMs
    24: PTB->PTOR = 0x080000; /* toggle green LED */
```

1secgreenLed.c | startup_MKL25Z4.s

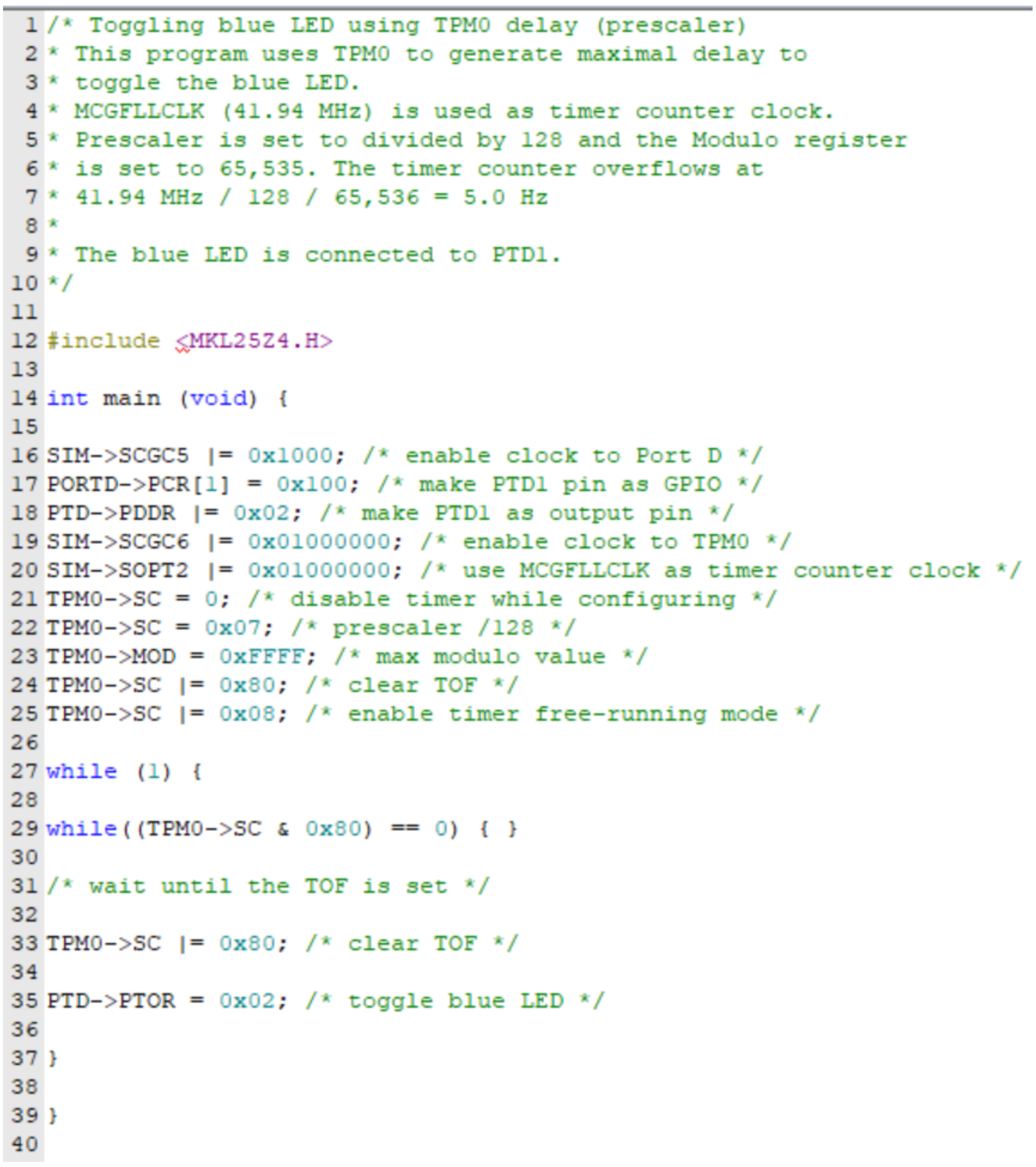
```c
 7
 8 For every 1000 delays (1 ms * 1000 = 1 sec), toggle the green LED once. The green LED is connected to P
 9
10 */
11 #include <MKL25Z4.H>
12 void delayMs(int n);
13
14 int main (void) {
15
16 void delayMs(int n);
17
18 SIM->SCGC5 |= 0x400; /* enable clock to Port B */
19 PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
20 PTB->PDDR |= 0x080000; /* make PTB19 as output pin */
21
22 while (1) {
```

Property	Value
PCR10	0x00000001
PCR11	0x00000001
PCR12	0
PCR13	0
PCR14	0
PCR15	0
PCR16	0x00000001
PCR17	0x00000001
PCR18	0x00000005
PCR19	0x00000105
PCR20	0
PCR21	0

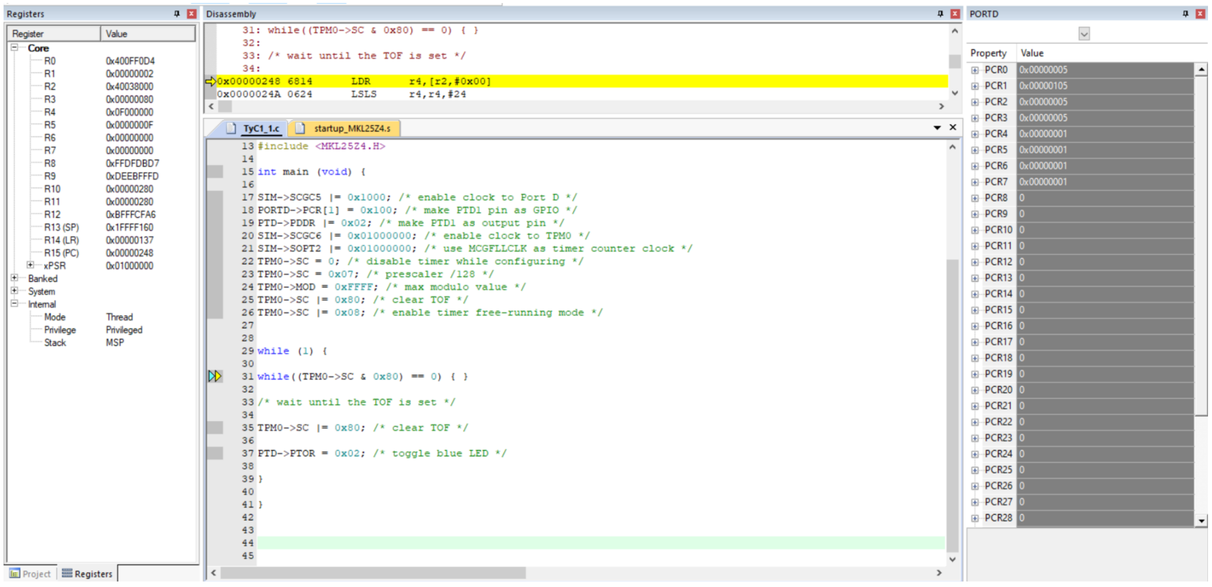
Part 3.

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 a 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.

Output Analisis: If we compare this code with the past one, first of all the LED that is going to blink is the blue one, and this LED is blinking faster, while the other codes takes 1 second, this LED has another frequency in the Modulo register (65,535) and we are using the 41.94 clock, doing the correct math it gives a 5.0Hz ratio.



```c
/* Toggling blue LED using TPM0 delay (prescaler)
* This program uses TPM0 to generate maximal delay to
* toggle the blue LED.
* MCGFLLCLK (41.94 MHz) is used as timer counter clock.
* Prescaler is set to divided by 128 and the Modulo register
* is set to 65,535. The timer counter overflows at
* 41.94 MHz / 128 / 65,536 = 5.0 Hz
*
* The blue LED is connected to PTD1.
*/

#include <MKL25Z4.H>

int main (void) {

SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
PTD->PDDR |= 0x02; /* make PTD1 as output pin */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000; /* use MCGFLLCLK as timer counter clock */
TPM0->SC = 0; /* disable timer while configuring */
TPM0->SC = 0x07; /* prescaler /128 */
TPM0->MOD = 0xFFFF; /* max 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 */

PTD->PTOR = 0x02; /* toggle blue LED */

}

}

```



Registers

Register	Value
Core	
R0	0x400FF0D4
R1	0x00000002
R2	0x40038000
R3	0x00000080
R4	0x0F000000
R5	0x0000000F
R6	0x00000000
R7	0x00000000
R8	0xFFDFDBD7
R9	0xDEEBFFFD
R10	0x00000280
R11	0x00000280
R12	0xBFFFCFA6
R13 (SP)	0x1FFFF160
R14 (LR)	0x00000137
R15 (PC)	0x00000248
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP

Disassembly

```
    31: while((TPM0->SC & 0x80) == 0) { }
    32:
    33: /* wait until the TOF is set */
    34:
0x00000248 6814      LDR      r4,[r2,#0x00]
0x0000024A 0624      LSLS     r4,r4,#24
```

TyC1_1.c | startup_MKL25Z4.s

```
#include <MKL25Z4.H>

int main (void) {

SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
PTD->PDDR |= 0x02; /* make PTD1 as output pin */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000; /* use MCGFLLCLK as timer counter clock */
TPM0->SC = 0; /* disable timer while configuring */
TPM0->SC = 0x07; /* prescaler /128 */
TPM0->MOD = 0xFFFF; /* max 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 */

PTD->PTOR = 0x02; /* toggle blue LED */

}

}
```

PORTD

Property	Value
PCR0	0x00000005
PCR1	0x00000105
PCR2	0x00000005
PCR3	0x00000005
PCR4	0x00000001
PCR5	0x00000001
PCR6	0x00000001
PCR7	0x00000001
PCR8	0
PCR9	0
PCR10	0
PCR11	0
PCR12	0
PCR13	0
PCR14	0
PCR15	0
PCR16	0
PCR17	0
PCR18	0
PCR19	0
PCR20	0
PCR21	0
PCR22	0
PCR23	0
PCR24	0
PCR25	0
PCR26	0
PCR27	0
PCR28	0

Steps to configure counter

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.

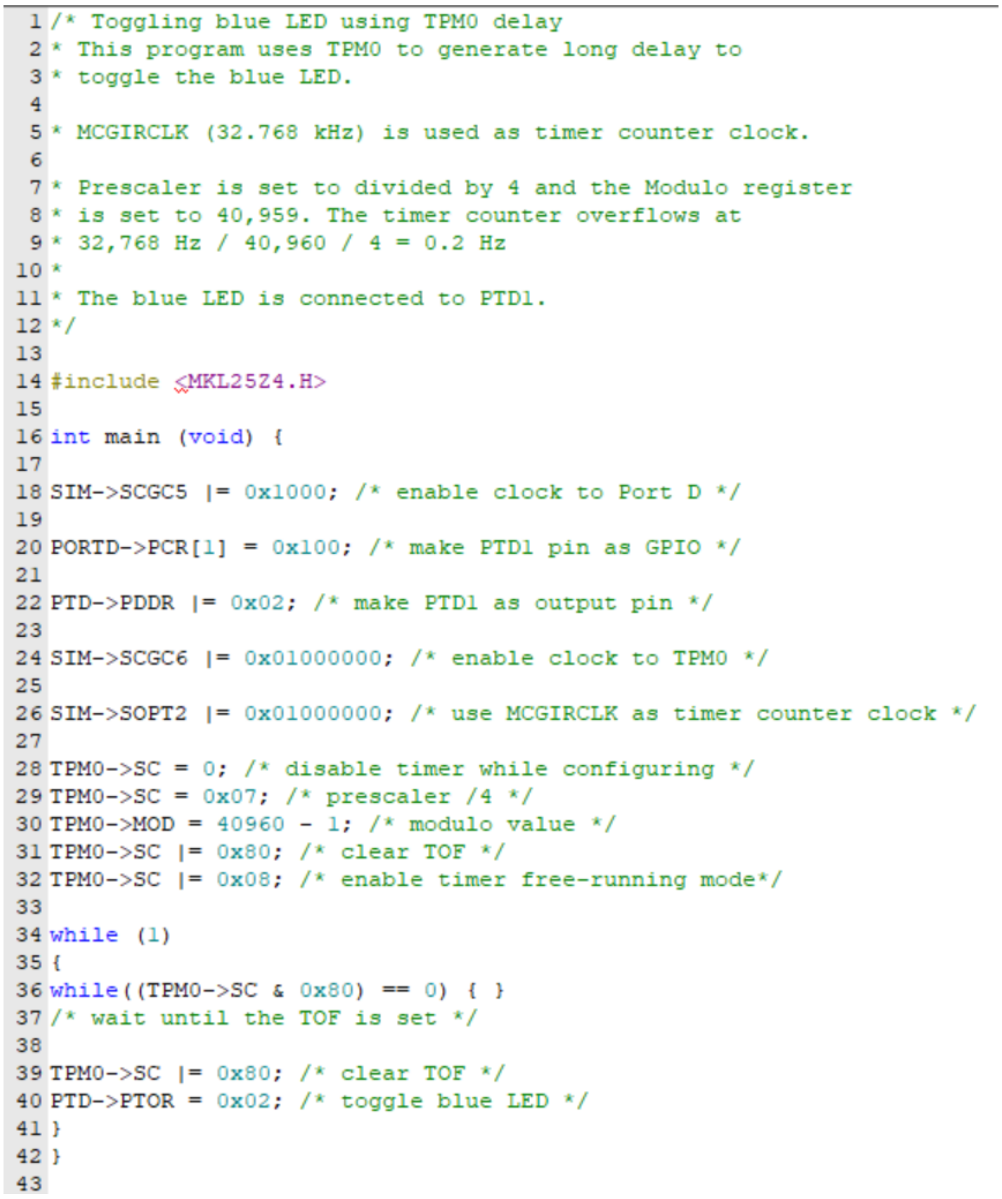
Follow up questions for the codes

(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.

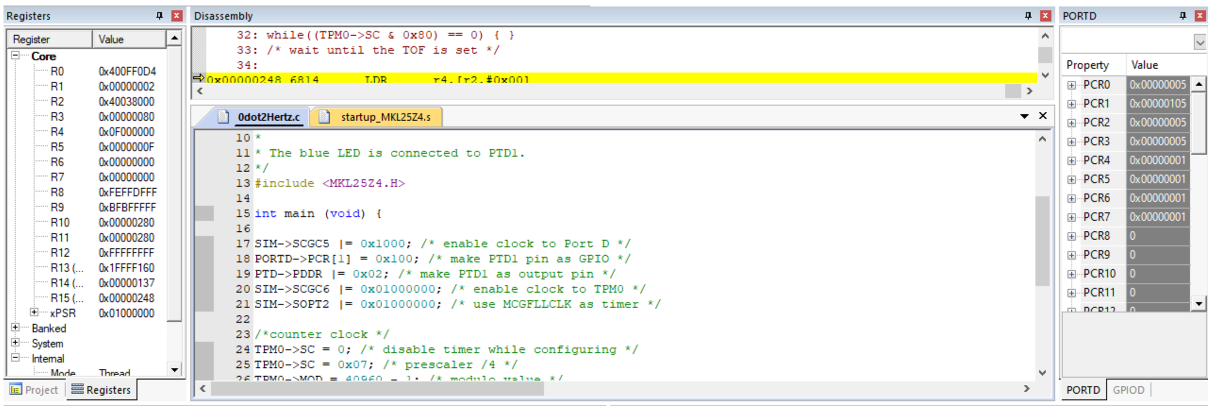
Part 4.

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 a timer counter clock. Prescaler is set to divided by 4 and the Modulo register is set to 40,959. The timer counter overflows at (41.94Hz) or 32,768 Hz / 40,960 / 4 = 0.2 Hz. The blue LED is connected to PTD1.

Output Analisis: The code has the intention to toggle the blue LED in a 0.2HZ ratio using the 32.768 clock, using this clock gave problems, so the LED neved toggled as requested. For it to be solved, the clock was changed to 41.94Hz, as a result we can see the toggle faster than the other codes.



```c
/* Toggling blue LED using TPM0 delay
* This program uses 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.
*/

#include <MKL25Z4.H>

int main (void) {

SIM->SCGC5 |= 0x1000; /* enable clock to Port D */

PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */

PTD->PDDR |= 0x02; /* make PTD1 as output pin */

SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */

SIM->SOPT2 |= 0x01000000; /* use MCGIRCLK as timer counter clock */

TPM0->SC = 0; /* disable timer while configuring */
TPM0->SC = 0x07; /* 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 */
PTD->PTOR = 0x02; /* toggle blue LED */
}
}

```



Registers

Register	Value
Core	
R0	0x400FF0D4
R1	0x00000002
R2	0x40038000
R3	0x00000080
R4	0x0F000000
R5	0x0000000F
R6	0x00000000
R7	0x00000000
R8	0xFEFFDFFF
R9	0xBFBFFFFF
R10	0x00000280
R11	0x00000280
R12	0xFFFFFFFF
R13 (...	0x1FFFF160
R14 (...	0x00000137
R15 (...	0x00000248
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread

Disassembly

```
    32: while((TPM0->SC & 0x80) == 0) { }
    33: /* wait until the TOF is set */
    34:
0x00000248 6814      LDR      r4,[r2,#0x00]
```

0dot2Hertz.c | startup_MKL25Z4.s

```c
*
* The blue LED is connected to PTD1.
*/
#include <MKL25Z4.H>

int main (void) {

SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
PTD->PDDR |= 0x02; /* make PTD1 as output pin */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000; /* use MCGFLLCLK as timer */

/*counter clock */
TPM0->SC = 0; /* disable timer while configuring */
TPM0->SC = 0x07; /* prescaler /4 */
TPM0->MOD = 40960 - 1; /* modulo value */
```

PORTD

Property	Value
PCR0	0x00000005
PCR1	0x00000105
PCR2	0x00000005
PCR3	0x00000005
PCR4	0x00000001
PCR5	0x00000001
PCR6	0x00000001
PCR7	0x00000001
PCR8	0
PCR9	0
PCR10	0
PCR11	0
PCR12	0

Visual Help

As part of the report, a video will be provided as reference of it being proven and downloaded to que KL25z board. The link below is a youtube video of the practice:

Link: https://youtu.be/joTjiN7gxdU

References

Tutorials Point. (2021). *Embedded Systems - Timer/Counter*. Retrieved from: https://www.tutorialspoint.com/embedded_systems/es_timer_counter.htm#:~:text=A%20timer%20uses%20the%20frequency,external%20signal%20to%20count%20pulses.

Partner. (2019). *Application Of Timer And Counter*. Our Time. Retrieved from: https://www.ourtime.org/application-of-timer-and-counter/#:~:text=In%20those%20applications%2C%20they%20measure,%2C%20set%20AC%20timer%2C%20etc.