



MANHATTAN  
COLLEGE

# Lab 2: Hibernation & Wakeup

On RTC Interrupt

School of Engineering

Electrical and Computer  
Engineering Department

ECEG 721-61 Embedded  
Systems

Jamie Quinn & Lucia  
Rosado-Fournier

October 3<sup>rd</sup>, 2020



## Table of Contents

<b>1.1 Objective .....</b>	<b>3</b>
<b>1.2 Introduction.....</b>	<b>3</b>
<b>1.3 Component Requirements .....</b>	<b>3</b>
<b>1.4 Software .....</b>	<b>4</b>
Flowchart.....	4
C code.....	5
<b>1.5 Procedure.....</b>	<b>6</b>
<b>1.6 Observation .....</b>	<b>6</b>
<b>1.7 Summary.....</b>	<b>7</b>

## 1.1 Objective

The first objective of this experiment was to figure out the solution to allow the code to work properly. Then, to understand how the hibernation module manages to place the device in a lower power state and then restores the power to the device on RTC (Real-Time Clock) interrupt.

## 1.2 Introduction

In this experiment, we turned on the hibernation module of the TM4C123GH6PM processor using the C code. The hibernation module shown in **Figure 1** provides logic to switch power off to the main processor and its peripherals while the processor is idle. Once in hibernation, the module is woken up by either an external signal or the built-in-Real-Time Clock. The RTC is used in the RTC Match-Seconds mode.

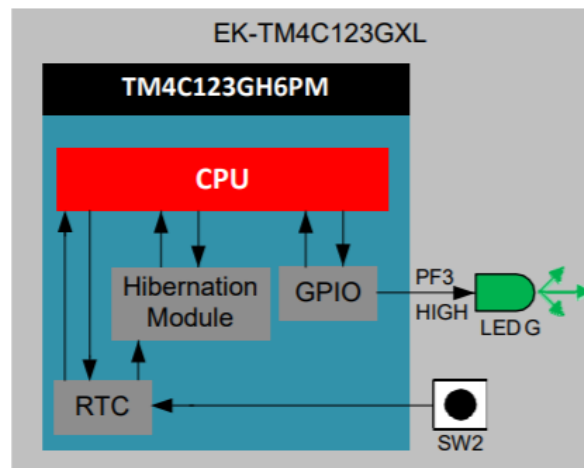


Figure 1 - Functional Block Diagram.

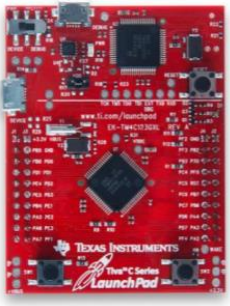
## 1.3 Component Requirements

### 1.3.1 Software Requirements

1. Keil uVision4
2. TivaWare\_C\_Series

### 1.3.2 Hardware Requirements

Table 1-2: Components Required for the Experiment

S.No	Components	Specification	Image
1.	Tiva LaunchPad	EK-TM4C123GXL LaunchPad	
2.	USB Cable		

### 1.4 Software

The software for the experiment is written in C and developed using the Keil Integrated Development Environment (IDE). The software is programmed into the target device TM4C123GH6PM on the EK-TM4C123GXL using the USB interface.

#### Flowchart

The flowchart for the Hibernate Mode and Wake up using RTC is shown in **Figure 2**.

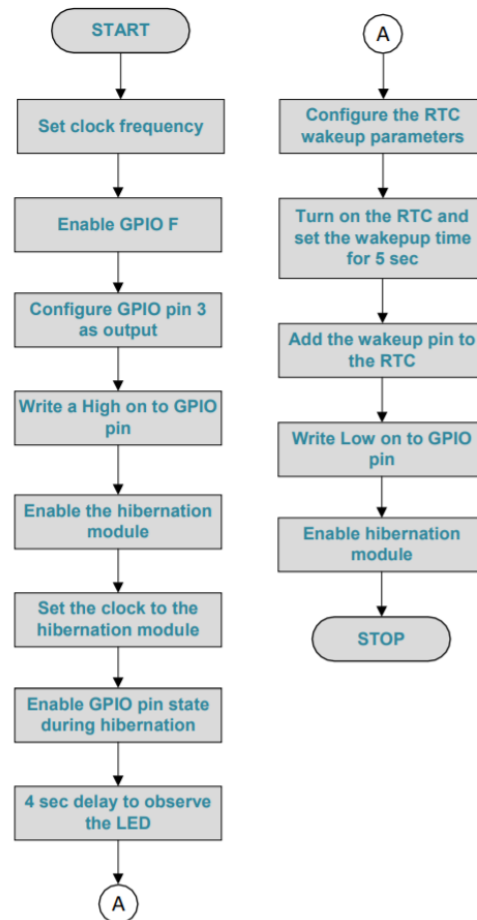


Figure 2 - Flowchart for Hibernate Mode and Wake up using RTC.

First the C code sets the system clock to 40MHz. The GPIO Port F of TM4C123GH6PM processor is enabled and pin 3 (PF3) configured as output. A HIGH is written on the GPIO pin to turn the green LED on.

Then the C code enables the Hibernation module and sets the clock to the hibernation module. It enables the GPIO pin state to be retained during hibernation and provides a 4-second delay for the user to observe the LED which is ON. The time interval for which the LED is ON is programmed by specifying the delay count value in the `SysCtlDelay()` function.

### C code

The code below follows the logic outlined in the flow chart. Though the code was given, it is heavily commented to explain the logic going through it.

```
#include <stdint.h>
#include <stdbool.h>
#include "utils/ustdlib.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/hibernate.h"
#include "driverlib/gpio.h"
int main(void)
{
    //Set the clock frequency
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    //Enables GPIO F
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //Configure pin 3 as the output
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    //Write a high to the pin to turn on the LED
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);
    //Enable the hibernation module
    SysCtlPeripheralEnable(SYSCTL_PERIPH_HIBERNATE);
    //Set the clock to the hibernation module
    HibernateEnableExpClk(SysCtlClockGet());
    //Enable GPIO pin state during hibernation
    HibernateGPIORetentionEnable();
    //4 second delay
    SysCtlDelay(64000000);
    //Configure the RTC wake up parameters
    HibernateRTCSet(0);
    //Turn on the RTC
    HibernateRTCEnable();
    //Set the wakeup time to 5 seconds
    HibernateRTCMatchSet(0,5);
    //Add the wakeup pin
    HibernateWakeSet(HIBERNATE_WAKE_PIN | HIBERNATE_WAKE_RTC);
    //Write low to the pin to turn off the LED
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3, 0x00);
    //Enable hibernation module
    HibernateRequest();
    while(1)
    {
    }
}
```

## 1.5 Procedure

The steps to run the code above on the microcontroller are below:

1. Connect the microcontroller to the computer using the USB cable. Make sure it is connected to the debug port on the board and the switch is flipped to debug as well.
2. Add the necessary files and libraries to the project to ensure the libraries and files referenced in the `main.c` can run, like it was done in the previous lab.
3. Under the C/C++ portion of the flash configuration tools, ensure “C99 Mode” is checked. The reason for this will be elaborated upon in the observation section. Also ensure other settings like the path to the necessary files are configured here as well, like it was done in the previous lab.
4. Build and load the code to the board, ensuring there are no errors. Then press the reset button on the board.
5. Check to make sure the LED turns on for 4 seconds and turns off for 5 to ensure the hibernation is working. Also try pressing switch 2 to wake up the processor to ensure that works as well.

## 1.6 Observation

As expected, the LED flashes green for 4 seconds and turns off for 5 seconds as seen in **Figure 3** below.

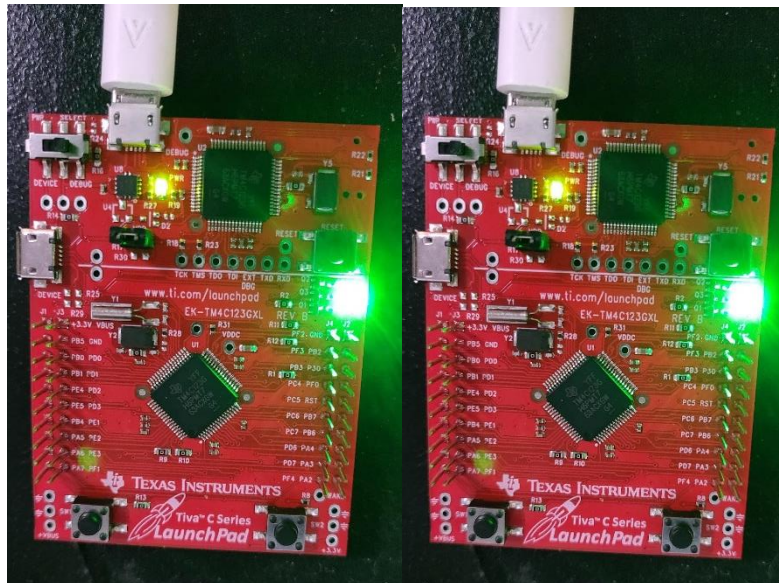


Figure 3 - Board with the LED on and the LED off.

The lab procedure ran relatively smoothly since the steps were similar to that of the last lab. However, there was one issue that occurred during the procedure. The error we received stated:

```
C:\ti\TivaWare_C_Series-2.1.0.12573\utils\ustdlib.h(56): error: #18:
expected a ")"
```

This is because in the `ustdlib.h` file, it uses the ‘restrict’ keyword, which is only supported in an older version of the C language, C99. To fix this problem, we had to add step 3 in the



procedure process outlined above. Checking off the “C99 Mode” box tells Keil to build the code using C99. After doing so, the program was able to build and load to the board

### 1.7 Summary

The experiment was generally straightforward, even with the small wrench regarding the C99 version of the code. Overall, it gave good practice on how to debug our own errors and get a better understanding of the hibernation module built into the microcontroller. Altogether, the completion of the lab and the report took about an hour.