ECEN 3002

Real Time Operating Systems Lab #2

GPIO, Timers, and Interrupts

Spring 2020

Objective: Utilize GPIO and Timers to demonstrate the use of interrupts as compared to polling from a main loop for both code flow and power consumption.

In this lab, you will utilize the LEDs, pushbuttons, and capacitive touch slider on the EFM32PG12 Pearl Gecko Starter Kit. The lab consists of two parts. In the first part of the lab, you will use a main loop to sample the pushbuttons and touch slider and to illuminate the LEDs based on the user input.

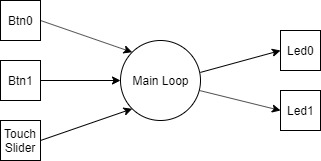


Figure 1: Main Loop Implementation

In the second part of the lab, interrupts will be used in place of polling from the main loop to sample the inputs and to drive the output LEDs. In the second part, the device should be placed into sleep mode when not awakened by the interrupts. You will compare the power consumption between the main loop implementation and the interrupt driven implementation.

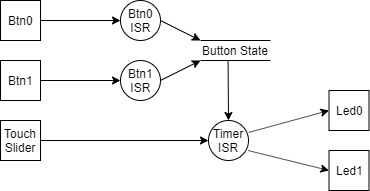


Figure 2: Interrupt-Driven Implementation

References:

1. Pearl Gecko Starter Kit User’s Guide: <https://www.silabs.com/documents/public/user-guides/ug257-stk3402-usersguide.pdf>
2. Pearl Gecko Reference Manual: <https://www.silabs.com/documents/public/reference-manuals/efm32pg12-rm.pdf>
3. Pearl Gecko Software Documentation: <http://devtools.silabs.com/dl/documentation/doxygen/5.7/efm32pg12/html/index.html>

Preparation:

1. Review the documentation on GPIO, Timers, LEDs, and Capacitive Touch Slider
2. Review the BSP functions for controlling the LEDs
3. Review the SDK functions for sampling the pushbuttons and capacitive touch slider
4. Review the implementation of an interrupt handler (see clock demo listed below)
5. Download and run the following Software Examples to become familiar with the hardware and interrupt handling required by this lab:
   1. SLSTK3402A\_blink – Demonstrates use of the LEDs
   2. SLSTK3402A\_touch – Demonstrates use of the capacitive touch slider
   3. SLSTK3402A\_clock – Demonstrates use of the pushbuttons and interrupts

Procedure:

Part I:

1. Create a new project in Simplicity Studio. You may either create a project from scratch, or else make a copy of one of the software examples. Name the lab RTOS\_Lab2\_GPIO\_Timers\_Interrupts.
2. Define global variables to separately store the state of each pushbutton (pressed or not pressed) and the direction of the capacitive touch slider (left or right).
3. Create a function to sample pushbutton 0 and a separate function to sample pushbutton 1. The code should only acknowledge one button being pressed at a time (i.e., pressing both buttons at the same time is treated the same as when no buttons are pressed).
4. Create a function to determine the position of a finger on the capacitive touch slider. At a minimum, 4 positions on the slider should be sensed (two left of center and two right of center). The code should determine if the finger is left of center or right of center, but not allow both at the same time (i.e., if a finger is placed on both the left side and right side, then the code should treat it as if no finger is placed). For this lab, either of the two positions left of center should be treated the same, and likewise for right of center.
5. Create a function that drives each LED based on the value of pushbutton 0, pushbutton 1, and the slider direction (using the global variables defined above).
   1. LED0 - Turn on while Btn0 is pressed or while the slider position is left of center; turn off otherwise.
   2. LED1 - Turn on while Btn1 is pressed or while the slider position is right of center; turn off otherwise.
6. From the main loop, call each function to get the state of each input and to update the corresponding global variable, then call the function that drives each LED. Delay 100 ms between each update.
7. Put a compile-time switch (e.g., #if !(LAB2\_USE\_INTERRUPT) …. #endif) around the code in the main loop that samples the inputs and drives the LEDs. Leave this code intact when developing Part II below.

Part II:

1. Use the SysTick timer to sample the touch slider and to drive the LEDs at a 100 ms period. From the SysTick interrupt handler, every 100 ms:
   1. Call the functions developed in Part I above to sample the touch slider and update the global variable.
   2. Call the function to drive the LEDs.
2. Create the interrupt handlers for the GPIO that is connected to Btn0 and Btn1. The interrupts should be enabled for both rising edge and falling edge. From each interrupt handler, call the function defined in Part I above to get the state of the pushbutton and update the corresponding global variable.
3. Make a call to enter Energy Mode 1 from the main loop (EMU\_EnterEM1()). After initialization, the main function should enter a loop that repeatedly calls only EMU\_EnterEM1(). This call will keep the system in sleep mode when no interrupts are active.
4. Put a compile-time switch (e.g., #if (LAB2\_USE\_INTERRUPT) …. #endif) around the code that is unique to the interrupt-driven implementation for Part II. Make any adjustments to the code as necessary such that only the code that is unique to each implementation is under a switch. The code that samples each input and drives the LEDs should be common, as well as any common initialization code.
5. Using the Energy Profiler:
   1. Measure the baseline power consumption in mW when no buttons are pressed and no finger is placed on the touch slider.
   2. Control the LEDs with each pushbutton and record the power consumption in mW when each LED is lit.
   3. Control the LEDs with the touch slider and record the power consumption in mW when each LED is lit.
   4. Light both LEDs using a combination of pushbuttons and the touch slider and record the power consumption in mW.
   5. Use the compile-time switch to build, run, and compare the power consumption for the polling method in Part I versus the interrupt-driven method in Part II.
   6. For the interrupt-driven method, reduce the frequency of the SysTick timer and compare the baseline power consumption.

Functional Tests:

The following functional tests should pass for both Part I and Part II

1. At reset, observe no LEDs are lit
2. Press Btn0 and observe only LED0 is lit
   1. Release Btn0 and observe LED0 is no longer lit
3. Press Btn1 and observe only LED1 is lit
   1. Release Btn1 and observe LED1 is no longer lit
4. Press both Btn0 and Btn1 simultaneously and observe that neither LED is lit
   1. Release only Btn0 and observe that only LED1 is lit
   2. Release both buttons and observe that no LED is lit
   3. Press both buttons again and release only Btn1 and observe that only LED0 is lit
   4. Release both buttons and observe that no LED is lit
5. Place your finger on the touch slider and slide your finger to the left
   1. Observe that LED0 is lit
6. Create a test case analogous to FT.5 above to test LED1.
7. With your finger on the left side of the touch slider
   1. Observe that LED0 is lit
   2. Press Btn0 and observe that only LED0 remains lit
   3. Press both Btn0 and Btn1 and observe that only LED0 remains lit
   4. Press Btn1 and observe that both LED0 and LED1 are lit
   5. Release Btn1 and observe that only LED0 is lit
   6. Place a second finger on the right side of the touch slider and observe that no LED is lit
8. Create a test case analogous to FT.8 above that tests the right side of the touch slider.

Grading:

1. 25 points equals “100%” for this lab.
2. All project files and source code must be submitted, such that results can be duplicated by the grader. ZERO score if all project files and source code are not submitted.
3. Basic Construction:
   1. Project builds without warnings and successfully executes as your cited results indicate when compile-time switch is set for “main loop” mode: 3 pts
   2. Able to build without warnings and successfully run as your cited results indicate when compile-time switch is set for interrupt mode: 3 pts
   3. No code that is not unique to the mode is under a compile-time switch: 2 pts
4. Power Measurements:
   1. Measurements provided for each system state as specified: 3 pts
   2. Analysis and comparison of power consumption for main loop method vs. interrupt-driven method: 2 pts
5. Functional Testing:
   1. Appropriate wording of student-specified Functional Test #6: 1 pt
   2. Appropriate wording of student-specified Functional Test #9: 1 pt
   3. Each passing functional test: 0.5 pt each (4 pt maximum)
6. Good Coding:
   1. Header documentation has been provided for all functions: 2 pt
   2. Header documentation follows Doxygen guidelines: 2 pt
   3. Appropriate use of comments inside functions: 2 pt
7. Bonus points (will add to the above-listed 25 pts if the lab is turned in on time—which can push your lab#2 grade above 100%)
   1. For the interrupt-driven implementation, what is the change in baseline power consumption (no buttons pressed and slider not touched) when the SysTick period is increased to 10 ms? Next, try 100 ms and larger intervals. What is the trade-off of increasing the SysTick period? (2 pt)
   2. Is the capacitive sense circuit using Successive Approximation or Delta Modulation? (1 pt)

Point Deductions for Late Submission: -5 points per day