LAB 09:

Timers and Interrupts

Provided Files

- main.c
- lab09lib.c
- lab09lib.h
- digits.bmp
- digits.c
- digits.h

Files to Edit/Add

- main.c
- Makefile
- .vscode
    - tasks.json

Instructions

In this lab, you will be completing several different TODOs, which will, piece by piece, complete a stopwatch that you can stop and restart. Your code may not compile until you complete an entire TODO block, at which point the game should compile with a new component of the final outcome (unless otherwise specified).

Note: Make sure to copy over your Makefile and .vscode/tasks.json from one of your previous assignments.

TODO 1 – Setting Up Interrupts

Let's set up interrupts! Navigate to the setupInterrupts() function in main.c.

TODO 1.0

- In setupInterrupts(), disable all interrupts
    - **HINT:** Which interrupt register turns all interrupts on or off?

TODO 1.1

- Set up the interrupt handler register. The interrupt handler register is a macro that can be found in lab09lib.h and should point to the interruptHandler function.

TODO 1.2

- Enable VBlank interrupts
    - **HINT:** Which interrupt register enables specific interrupts? Which subordinate interrupt control register is used to enable VBlank?

TODO 1.3

- Enable button interrupts. We will use Button A, the select button, and the start button for this lab.
    - **HINT:** Which interrupt register enables specific interrupts? Which subordinate interrupt control register is used to enable Button input? Also, remember that you will need to set each button you are using in addition to enabling button input.

TODO 1.4

- Re-enable all interrupts
    - **HINT:** Which interrupt register turns all interrupts on or off?

Build and run. There will not be any visible changes, but you can reference the “Setting up an Interrupt” instructions from lecture and recitation to make sure you have completed TODO 1 correctly.

TODO 2 – Enabling and Handling Timer Interrupts

Now you’ll focus on enabling timer interrupts, specifically. Make sure to reference the timer macros in lab09lib.h if you are stuck! Navigate to the enableTimerInterrupts() function in main.c

TODO 2.0

- You will use timer 0, timer 1, and timer 2 for this lab. Enable these timers in the interrupt enable register. Note: you do not need to set the control register for each timer in this step!

TODO 2.1

- Now, you’ll set the control register and the data register for timer 0, which triggers once per centisecond.
- First, set the value of the control register for timer 0 equal to 0.
- Next, set the value of the data register for timer 0 equal to (65536 - 164).
- Now, set the control register for timer 0 such that the timer frequency is 1024 cycles, the timer is on, and the timer raises an interrupt on overflow.

TODO 2.2

- Next you’ll set the control register and the data register for timer 1, which triggers once per second.
- First, set the value of the control register for timer 1 equal to 0.
- Next, set the value of the data register for timer 1 equal to (65536 - 16384).
- Now, set the control register for timer 1 such that the timer frequency is 1024 cycles, the timer is on, and the timer raises an interrupt on overflow (same as timer 0).

TODO 2.3

- Finally, set the control register and the data register for timer 2, which triggers once per minute.
- First, set the value of the control register for timer 2 equal to 0.
- Next, set the value of the data register for timer 2 equal to (65536 - 60).
- Now, set the control register for timer 2 such that the timer cascades from timer 1, the timer is on, and the timer raises an interrupt on overflow.

Next, we want to handle timer interrupts. Navigate to the interruptHandler() function in main.c.

TODO 2.4

- In interruptHandler(), disable all interrupts.
    - **HINT:** Look at TODO 1.0!

TODO 2.5

- You will handle the timer interrupts. You will be setting the conditions for three if statements. These if statements are in UNCOMMENT 2.0, 2.1, and 2.2 respectively. As the condition for each if statement, you will check if the interrupt flag register has the bit set for each of the three timers (more details in code comments). After you add each condition, uncomment the appropriate UNCOMMENT.

TODO 2.6

- Now you will handle the VBlank interrupt. If the interrupt flag register has the bit set for VBlank, you will call the displayTime() function and DMA the shadowOAM into the OAM.

TODO 2.7

- Tell the GBA that the interrupt has been handled. Remember that this involves updating the value of the interrupt flag register.

TODO 2.8

- Re-enable all interrupts
    - **HINT:** Look at TODO 1.4!

Build and run. You should see a white stopwatch on a black background. The stopwatch should be incrementing and should display minutes, seconds, and centiseconds.

TODO 3 – Handling Button Interrupts

Now you'll handle button interrupts. Navigate to the interruptHandler() function in main.c

TODO 3.0

- Uncomment UNCOMMENT 3.0. This code changes the background color if Button A is pressed or held.

TODO 3.1

- Now you will handle the start button interrupt. If start is pressed, the stopwatch should restart.

    - **HINT:** Your condition for handling the interrupt should be similar to the condition for handling the Button A interrupt in TODO 3.0, just with the start button instead.
- To restart the stopwatch, first turn the timer off in each timer's control register.
- Next, set each time variable (defined at the top of main.c) equal to 0.
- Finally, you will set each timer's control register as outlined in the last part of TODOs 2.1 - 2.3 (e.g. set the control register for timer 0 such that the timer frequency is 1024 cycles, the timer is on, and the timer raises an interrupt on overflow).

TODO 3.2

- Now you will handle the select button interrupt. If select is pressed, the stopwatch should stop.

Build and run. You should be able to adjust the color of your background if you press or hold down A. If you press start, your stopwatch should restart to 00:00:00 and immediately start incrementing again. If you press select, your stopwatch should stop incrementing.

You will know if it runs correctly if:

- You can press or hold down A to adjust the color of your background.
- You can press start to restart your stopwatch and have it immediately begin incrementing again.
- You can press select to make your stopwatch stop incrementing. The only way to make the stopwatch run again from this state is to press start to reset the stopwatch and begin incrementing again.

Tips:

- Follow each TODO in order, and only move forward if everything is correct.
- Reference lab09lib for register and flag macros!

Submission Instructions:

Ensure that **cleaning** and building/running your project still gives the expected results. **Please reference the last page of Lab02.pdf for instructions on how to perform a "clean" command.**

Zip up your entire project folder, including all source files, the Makefile, and everything produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab09_LastnameFirstname, for example:

"Lab09_KongDixie.zip"

It is your responsibility to ensure that all the appropriate files have been submitted, and that your submitted zip can be opened and everything cleans, builds, and runs as expected.