LAB 5: Our First Interrupts

1. Updating some files

If the file

```
XilinxWorkspace\standalone_bsp_0\ps7_cortexa9_0\libsrc\standalone_v6_6\src\asm_vectors.S
```

does not have the line

```
.weak FIQHandler
```

around line 70 slightly below the comment block at the top of the file, please replace that file and also the file

```
U:\XilinxWorkspace\standalone_bsp_0\ps7_cortexa9_0\lib\libxil.a
```

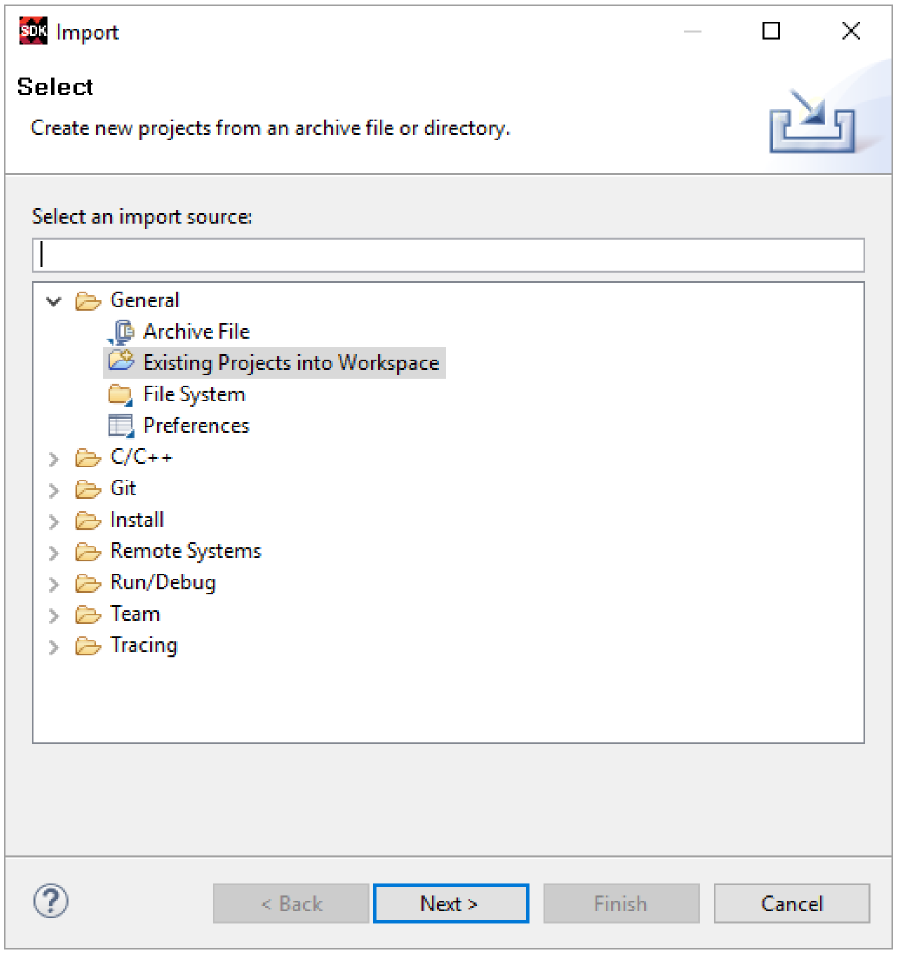
with the files from Canvas next to this document.

2. Importing the Lab 5 Project

A Lab 5 project has been created with some starting code included. Download the Lab5.zip file from Canvas and import it into the ENSC 254 workspace that you set up in Lab 4 as follows:

File -> Import

Select ‘Existing Projects into Workspace’ under the ‘General’ category.

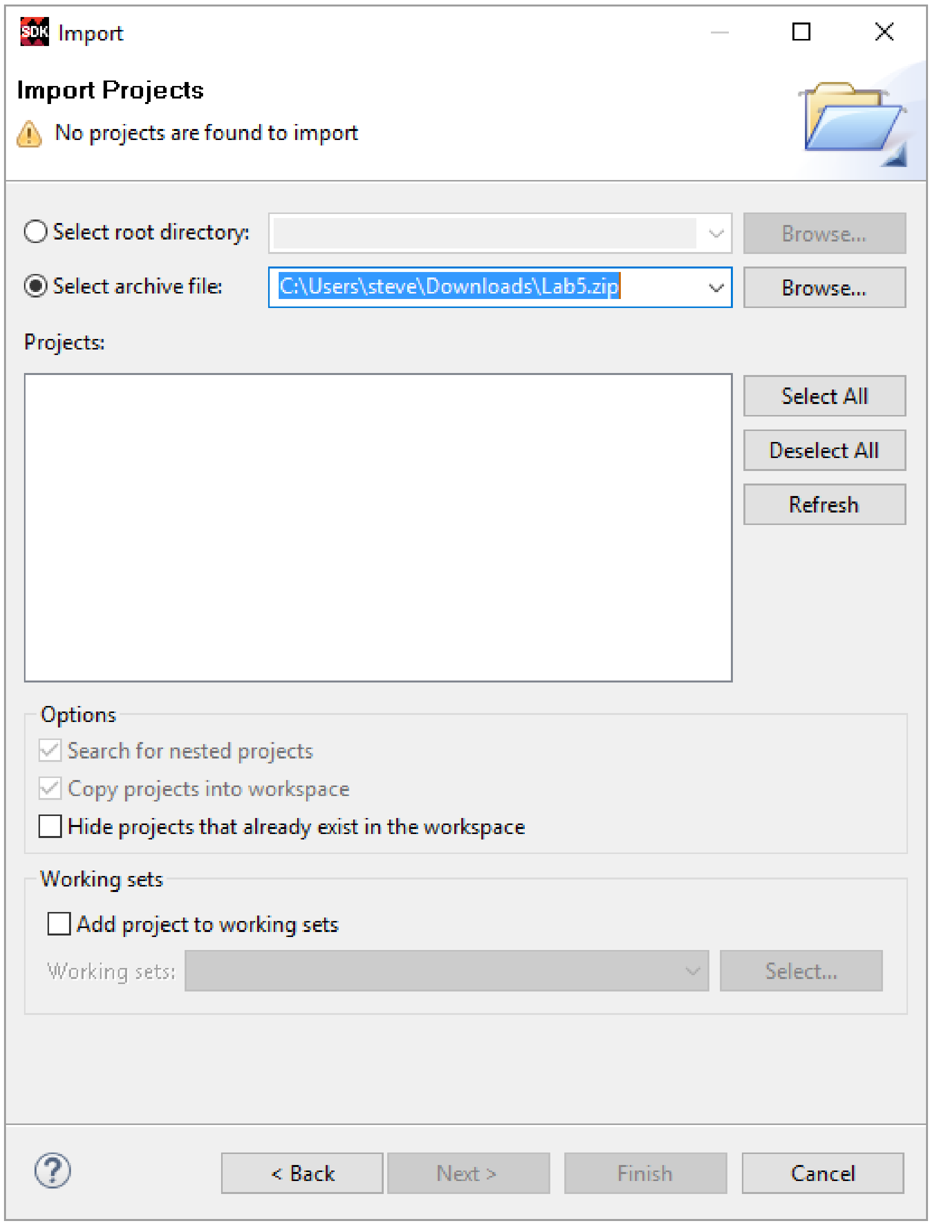

Import

Select

Create new projects from an archive file or directory.

Select an import source:

Select 'Select archive file' as shown below and browse to your downloaded copy of Lab5.zip. Select 'Finish'.



Import

Import Projects

No projects are found to import

Select root directory:

Select archive file: C:\Users\steve\Downloads\Lab5.zip

Projects:

Options

- [✓] Search for nested projects
- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

A new Lab 5 project should appear in the Project Explorer pane next to your previous Lab 4 project. Inside you'll find five primary source files: 'main.c', 'asm.S', 'OLED.S', 'fib.c' and 'asuAdd.S'.

The 'main.c' file is identical to the one from Lab 4. Its primary function is simply to call our *asm_main* subroutine located in 'asm.s'.

The 'fib.c' file should look familiar from your earlier assignment, and contains a solution to it. We're using it here to give the ARM some 'work' to do. It is called in the main loop of our assembly routine.

The ‘asm.s’ file is where we’ll start working in this lab. Subroutine *asm_main* first clears the OLED screen and does other initialization. It then enters a loop where it calculates and displays a value *maxSize*, calls the Fibonacci code, and then displays the return value on the OLED screen. In this lab you’ll be modifying the code for the interrupt handler and code related to the OLED display.

3. Build and Run the Starting Code

Compile the Lab 5 Code using ‘Project->Build Project’ or the appropriate icon or Project context menu. If the .elf file does not show up under “Binaries” in the project, refresh the project using the project’s context menu (or some other valid method). Do this again if necessary until “Binaries” shows up.

Launch the project in the debugger using the “System Debugger using Debug_Lab5.elf” launch configuration found in the drop-down list by the bug icon. If it is not in the drop-down list, find it in “Debug Configurations…”.

The launch configuration is configured to at first stop at the first ARM instruction the processor encounters, which is the first element of the vector table. If you resume execution, the debugger will next stop at the beginning of the main function, as also requested in the launch configuration. Try stepping into/through and then resuming the code on the Zedboard. You should be able to step into the assembly code files as before.

When the code is running, the OLED display should clear and numbers should appear below the phrase “Welcome”. Exercising any of the buttons will cause a binary count of interrupts to appear on the individual LEDs.

4. Modify the Interrupt Behavior

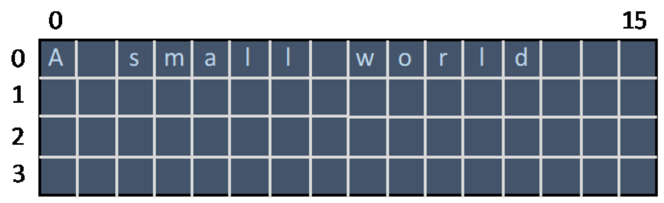
The *FIQHandler* routine in the ‘asm.S’ file is our FIQ interrupt handler routine, which is currently handling button interrupts. When a button is exercised, our normal main processing code gets interrupted and this handler routine gets executed. When it is called, the handler code provided simply increments once a count variable stored in a register, outputs the count to the LEDs, and then returns -- back to the interrupted code. You may have noticed that the count may increment several times when a button is pressed or released. How could we solve this? Think about this for the next lab.

You’ve probably noticed that there is often a delay between adjusting the slider switches and seeing *maxSize* get updated on the screen. This is because the characters only get updated after some often large Fibonacci number gets calculated in our main loop. As a result, the period of the screen update for *maxSize* depends on how much work is being done in the main loop – which can often be enough to create a noticeable lag. Compare this lag to the speed of the buttons adjusting the interrupt count.

Your First Task… Based on your work in the previous lab, modify the handler code to increment the count only when the ‘Up’ button is pressed, to decrement the count when the ‘Down’ button is pressed, and to clear the count when the ‘Centre’ button is pressed. You can assume for this lab that only one button will be pressed at a time.

5. Update interrupt count on OLED from within Interrupt Handler

In the main loop of the asm.S file you'll find code for calling a subroutine to display a number on the OLED display. The display can show four lines of text with each line being 16 characters long as shown below.



Your Next Task… Instead of just displaying the interrupt count on the individual LEDs, display the interrupt count from within the interrupt handler at the bottom of the OLED display. Be very careful with this. The FIQ interrupt handler has interrupted some other background code being executed, and is sharing registers R0 through R7 with that code.

6. Modify OLED Counter Display

The display is mapped into the processors memory space as an array of bytes with each byte corresponding to a character on the screen. You can think of this array as being organized into rows and columns as shown above. To write a character to the screen, simply write the ASCII code of the character to the corresponding memory array location.

The offset to the character array from the OLED controller base address is defined by us as OLED_CHAR_OFFSET.

Your Last Task… Based upon the code already provided to you, such as the code that displays "Welcome!", modify the project so that you can get a screen that looks something like the following:

```
ENSC 254 Lab 5.
maxSize: 512
return: 23601
interrupts: 42
```