LAB HW8: 2 Sources of FIQ Interrupts -- updated

First, every student should individually upload their work from HW7 to CourSys. This is just to have a record of your work. Your work for HW6, HW7, and HW8 will be evaluated during the Practical Exam at the end of the course.

This week we will continue working with the simulated STR730 EVALBOARD, and will start programming even more hardware on the board. In particular, we will be programming External Interrupts.

> First of all, make a copy of the Keil project where you finished last week's lab. Copy the whole directory tree and rename the destination directory to HW8-EI.

Open the new Keil Project in uVision. We are going to be making some edits to allow an FIQ interrupt to be triggered either from a TIM0 interrupt or from an INT0 external interrupt. On the simulated EVALBOARD, INT0 is connected to a pushbutton switch on GPIO1 at Pin 8 (i.e. P1.8).

We will keep the same program behavior as before for the TIM0 interrupt, but we want to make some change to the program output when a rising edge is detected from the pushbutton switch at P1.8. My suggestion is to slow down the strobing by adding 0x10 or so to the TIM0 prescaler for each rising edge. When the prescaler overflows, set it to a value that does strobing that looks okay and is not too fast. A number of changes will be required to achieve all of this…

1) Configure Pin 8 of GPIO1 (Pin 1.8) to be an IN_TRI_TTL (1-0-0) input pin (recall Table 16 in Section 6.1 on page 68 of the RM0001 document).
2) Configure INT0 external interrupt to be triggered with a rising edge (see Table 22 in Section 7.8.2 on Page 102 of the RM0001 document).
3) Change the FIQ interrupt handler. When an FIQ interrupt occurs, the handler must look at the EIC_FIPR and figure out if one or the other or both of the FIQ sources are trying to interrupt. The actions performed will depend on such determination. Actions on the TIM0 prescaler, as discussed above, must be added if INT0 is the interrupt source. The correct interrupt source(s) must also be cleared in the EIC_FIPR (hint: the value read in from EIC_FIPR can just be written back to the register to clear all sources read in).
4) Change the initialization of EIC_FIER so that both sources of interrupts will be enabled.

In the template uVision project that I distributed for HW6, I changed STR73x.s in a couple ways regarding the FIQ. I introduced the subroutine FIQ_init, and I introduced an include file FIQ.inc, which gets included at the bottom of the vector table around line 169 of STR73x.s.

Modify FIQ.inc to make the FIQ interrupt as fast as possible, especially for the most common source of FIQ interrupts. Consider each instruction that gets executed when an FIQ interrupt is generated and decide if that instruction is really needed for our use of the FIQ interrupt and whether you can make that instruction faster. You are free to comment out or delete as many lines of FIQ.inc as you want and to move as much code as you want into FIQ.inc. Don't forget that you can initialize registers in the range R8 to R12 in the subroutine FIQ_init. Make sure that at the end of processing an FIQ interrupt that the processor goes back to the previous mode (probably System mode) from FIQ mode. In order for the code to build nicely, please leave an FIQ_Handler subroutine in the file assembly.s, even though it will no longer be used. The remaining FIQ_Handler can contain only the instruction "mov pc, lr" if you would like. Please copy the first 20 or so comment lines from the file assembly.s to the top of FIQ.inc and update in both files for HW8 those lines with Helpers and Resources and so forth.

Note: you will probably want to change the line in STR73x.s from…

```
                INCLUDE FIQ.inc
```

to…

```
                #include        "FIQ.inc"
```

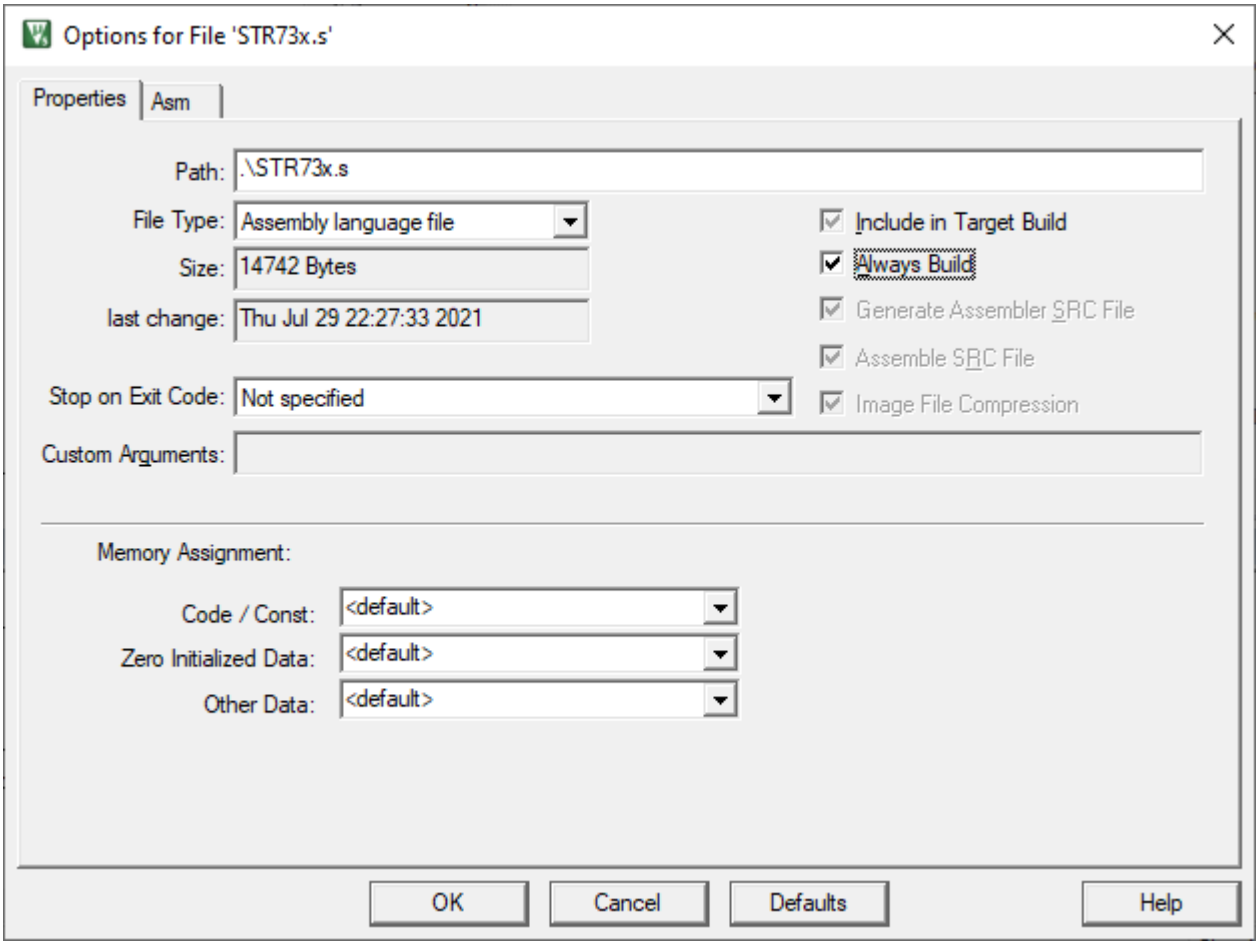
This will allow you to include relevant .h files towards the top of FIQ.inc (and also allow you a broader range of options to start comments in the code). So it will allow you have these lines in FIQ.inc…

```
#include "asm_include.h"
#include "73x_tim_l.h"
#include "73x_eic_l.h"
```

Note that the C-preprocessor's #include directives create a single relatively large file for the assembler to assemble, and that the C-preprocessor does not strip out the assembler's END directive and any text in a file after the END directive. Therefore, you should probably remove any END directive from any .inc file #included by the C-preprocessor. Whereas ARMASM's include directive will give you a warning if an include file does not have an END directive, having an END directive when #including a file using the C-preprocessor will often cause problems.

Further note that using #include to include a .inc file from a .s file seems to break the dependency checking of the uVision/ARM build system, and therefore making a change in the .inc file will not normally trigger the .s file to be built using the uVision Build (F7) command. For this reason, please

get the options for the file STR73x.s and click on “Always Build” until you get a black checkmark for that option. This will force that file and it’s FIC.inc, which is #included by it, to be assembled every time a build is performed. See the following window-shot.



Make a .zip archive of your uVision project and individually upload to CourSys when the submission page is enabled.

Please ask on Piazza if you have any questions. If you finish early and are bored, try extending the work done in this lab. For example, consider switching from FIQ to IRQ. Ask on Piazza if you want more ideas.

Craig