Pre-lab Assignment 4

Due: Fri, 25 Sep 2020 23:59:59 (approximately 76 days ago)
[Score: 19 / 25 points possible]

An interrupt is a hardware-invoked subroutine. Some form of trigger causes execution of the normal program to be suspended. A subset of the *state* of the CPU is saved on the stack. In particular, registers R0-R3, R12, LR, PC, and xPSR are saved. This stack content is called an exception frame. A special value is loaded into the LR register to indicate that the stack contains an exception frame. This tells a subsequent **`BX LR`** or **`POP {...,LR}`** instruction to load the exception frame contents back into their respective registers. In this way, the interrupted execution can be treated just like a subroutine that can be *returned* from.

The type of trigger decides what form of *exeception vector* the CPU should read from the vector table. The 32-bit value read from the vector table is put into the PC and execution continues in this new subroutine. Since the subroutine is meant to handle some kind of interrupt, it is called an Interrupt Service Routine (ISR). Since R0-R3, R12, LR, PC, and xPSR have been saved, the subroutine can modify all of the registers that the ABI normally allows. If it modifies R4-R7, they should be saved on the stack in an initial **`PUSH {R4-R7,LR}`** instruction at the start of the subroutine.

Academic Integrity Statement [0 ... -100 points]

By typing my name, below, I hereby certify that the work on this prelab is my own and that I have not copied the work of any other student (past or present) while completing it. I understand that if I fail to honor this agreement, I will receive a score of zero for the lab, a one letter drop in my final course grade, and be subject to possible disciplinary action.

Raghuram Selvaraj 

---

(1) [1 point]

Look at startup/startup_stm32.s in any Standard Peripheral project in System Workbench. The g_pfnVectors label indicates the beginning of the exception vector table. A comment above it describes the .section information needed to make the linker place it at the proper location in memory.

What is the address range of the STM32 exception vector table? (State it in the form of two hexadecimal numbers like 0x***FROM*** - 0x***TO***) Include the reset vector and initial stack pointer value. Include the space up to, and including, the word for USB_IRQHandler. There are a variety of ways to describe the span of the vector table, so this one is not necessarily the same value you saw in the lecture notes.

0x00 - 0xBC 

---

(2) [1 point]

Each value label used in a vector table entry in startup/startup_stm32.s is defined further down in the file with a **.weak** directive. What does this directive mean? If two different subroutines are defined in different .s files *with the same name*, and one is given a .weak designation and the other is given a .global designation, what will happen?

```
This means that the handler is defined in another module, so a developer can
customize what the machine does in different modules when the handler is
triggered. The global definition executes and the weak definition does not
prevail.
```

---

(3) [1 point]

Each value label used in a vector table entry in startup/startup_stm32.s is defined further down in the file with a **.thumb_set** directive. What does this mean? What would happen if it were omitted?

```
It creates a label that acts as an alias for another label, while also marking the
aliased label as a thumb function entry point (16-bit encoding). Here, it allows
the currently-weak but future-defined handlers to override the default handler. If
omitted, the specified handlers won't trigger. This also ensures that the LSB is
set to 1. An error would occur if the directive is omitted because the linker
would consider the code to be ARM instead of thumb instructions.
```

---

(4) [1 point]

In addition to declaring it .global, we recommend, for any label of a subroutine used as an ISR, to also declare it with the following directive:

```
.type some_handler, %function
```

What does this do? What would happen if it were omitted?

```
This ensures that the LSB for each reference to some_handler is set to 1. Without
it, the linker would consider it to be ARM instructions instead of thumb
instructions and an error would occur.
```

---

(5) [1 point]

Why are things like "**`.thumb_set`**" and "**`.type ..., %function`**" needed? I.e., what happens if you omit them? Try doing so with an ISR.

As mentioned before, these directives are needed to set the LSB. Without them, the linker will interpret the instructions as ARM instructions instead of thumb instructions as needed.

---

(6) [1 point]

How many distinct exception vectors does an STM32F0 microcontroller support? (Hints: Take a look at the g_pfnVectors table in startup/startup_stm32.s and count them. Note that the first word, _estack, is not an exception, but the starting value of the SP register. Any entries that are zero do not represent a supported handler.) Some of them may not be supported by the specific microcontroller we use for ECE 362. For instance, we have no USB peripheral in the STM32F091RCT6, but there may be one available in other versions of the STM32F0. Therefore, any "F0" family chip has a defined vector location for it.

37

---

(7) [1 point]

For the next few questions, consult the documentation for the SysTick timer (STK) subsystem in section 4.4 of the STM32F0xxx Cortex-M0 programming manual. The NVIC and STK subsystems are "core peripherals" and are only fully documented in the programming manual. You will not find much information about them in the Family Reference Manual.

Which bit in which register allows the SysTick down-counter to run?

TICKINT in STK_CSR

---

(8) [1 point]

What is the address of the STK_CVR register? (Indicate the absolute address, not the offset from the SysTick register base.)

0xE000E018

---

(9) [1 point]

Write a simple ISR for the interrupt generated by SysTick. You should make it just like the one you saw in the lecture notes, except that your ISR should toggle PC6. You do not need to initialize the SysTick subsystem---just write the ISR. For the code you write here, you may assume PC6 is already set up for output. Assume the availability of any .equ statements for symbols like "GPIOC" and "ODR" that represent the base address of the

GPIOC port registers and the offset of the Output Data Register, respectively. (This is easy enough to try on your own for practice before your lab. Make the red LED blink, and build your confidence. Keep in mind that a Standard Peripheral project in System Workbench makes the CPU run at 48 MHz. The **maximum** value you can store in the SysTick reload value register is 16,777,215. That would generate a SysTick interrupt every (16,777,215 + 1)/48,000,000 ≅ 0.35 seconds.)

```
.type SysTick_Handler, %function
.global SysTick_Handler
SysTick_Handler:
        ldr r0, =GPIOc
        ldr r1, [r0, #ODR]
        movs r2, #1
        lsls r2, r2, #6
        orrs r1, r2
        str r1, [r0, #ODR]
        bx lr
```

---

(10) [1 point]

How times per second would the LED *toggle* if the STK_RVR held the value 59,999, and the CLKSOURCE bit of the STK_CSR was set to 0?

If this question seems unanswerable, you might have to go digging through the Family Reference Manual to find out how the external clock source is synthesized, or you can read the entirety of the lab document.

100 

---

(11) [1 point]

The rate of the LED blinking will be half the toggle rate. How many times per second will the LED in question 10 make a full cycle through on-off?

50 

---

(12) [1 point]

What value would you use for the STK_RVR if you wanted the LED in question 10 to blink on at a rate of *exactly* 320 times per second? (Remember that the SysTick counter cycle is one longer than the value in the STK_RVR.)

18,749 

---

(13) [1 point]

If the CLKSOURCE bit in STK_CSR is set to zero, the STK_CVR is set to 8,999,999, and the STK_RVR is set to 11,999,999, how many seconds will elapse from the moment the counter is started until the SysTick_Handler ISR is invoked for the first time?

0.5 ✗

---

(14) [1 point]

How many seconds, at most, can the duration between SysTick_Handler interrupts be if the CLKSOURCE bit in STK_CSR is set to zero? (State your answer with four sigificant digits. i.e., x.xxx)

2.796 ✓

---

(15) [1 point]

Why is it that we need to enable the EXTI interrupt in the NVIC ISER register but not for the SysTick interrupt?

To let the system acknowledge that the event has been handled for NVIC ISER so that execution goes back to the main program. ✗

---

(16) [1 point]

Which register is used to un-mask the EXTI interrupt for a particular pin?

EXTI_IMR ✓

---

(17) [1 point]

Name the register that can be used to set the EXTI interrupt to be triggered on a falling edge?

EXTI_FTSR ✓

(18) [1 point]

Which EXTI interrupt handler do pins 2 and 3 of any port invoke?

Hint: Look at the vector table in startup/startup_stm32.s to find the exact name of the interrupt handler. Copy and paste it below. You must always use the ***exact*** name of the ISR to override the (weakly definined) default handler. If you don't use the exact name, the compilation won't even give you a warning. It's a good idea to always copy and paste the value from startup/startup_stm32.s file. Did we mention you can always find the ***exact*** name in startup/startup_stm32.s? Lots of students forget this, and they try to type it in from memory or a guess. **When you waste a precious hour of your life trying to understand why your ISR is never invoked, and finally realize it's because you mistyped it, then you will know what your instructor feels like at least twice per semester.** Copy it and paste it. ...or waste your life. It is your choice.

EXTI2_3_IRQHandler 

(19) [1 point]

What register must you write to to change the port used for an EXTI on pin 3? (E.g., if you wanted to use PB3 instead of PA3 for an EXTI.)

SYSCFG_EXTICR3 

(20) [1 point]

The subsystem for the register you named in the previous question must have a clock enabled for it before it can be modified. This is similar to how you must set the GPIOCEN bit in the RCC_AHBENR to enable the clock to Port C. (Almost every subsystem in the STM32 other than the NVIC and SysTick needs to have a clock enabled before it will do anything.) What bit in what register must you set to '1' to enable that clock before you update the register in the previous question? (Write the symbol name of the register and bit.)

RCC_APB2ENR > SYSCFGCOMPEN 

(21) [1 point]

What is the full name for the interrupt service routine that will be invoked for an external interrupt on pin 6 of a GPIO port?

EXTI4_15_IRQHandler 

(22) [1 point]

What is the full name for the interrupt service routine that will be invoked for an external interrupt on pin 12 of a GPIO port?

EXTI4_15_IRQHandler 

---

(23) [1 point]

What is the *interrupt number* for the interrupt in the previous question? (See Table 37 on page 217 of the Family Reference Manual.)

7 

---

(24) [1 point]

What is the default priority value in the NVIC_IPR for the EXTI0_1 interrupt?

0 

---

(25) [1 point]

What is the *relative* priority value for the EXTI0_1 interrupt compared to any other interrupt with the same value in the NVIC_IPR? (See Table 37 of the Family Reference Manual again.)

12