Post-Lab 2: Interrupts

Justin Francis
Wednesday, Feb 12

Overview

This lab introduces the concept of interrupt-driven programming and guides through the configuration of interrupt-oriented peripherals; the exercises herein provide a foundation for utilizing interrupts in an embedded application. They introduce the practice of enabling, configuring parameters and writing handler routines to service peripheral interrupt requests. After completing this lab, you will understand how to use interrupts effectively without impacting the main application or each other.

1 Post-lab Questions

1. Why can't you use both pins PA0 and PC0 for external interrupts at the same time?

   Answer: They cannot be used at the same time for external interrupts because they are multiplexed through the same SYSCFG/EXTI register (SYSCFG_EXTICR1, Periph Manual pg 214). So only one can be vectored (handled) at a time.

2. What software priority level gives the highest priority? What level gives the lowest?

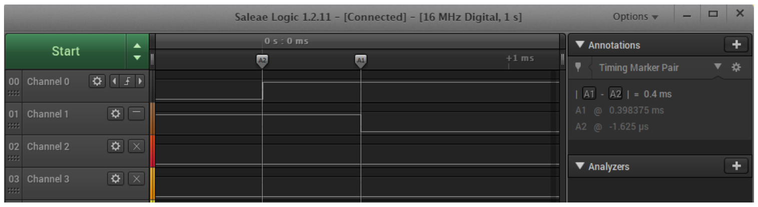
   Answer: The more negative the priority level, the higher the priority in software, e.g. -3 is the highest reserved priority, 0 is the highest user priority, and 3 is the lowest user priority, (PeriphRef-Manual.pdf, pg 209).

3. How many bits does the NVIC have reserved in its priority (IPR) registers for each interrupt (including non-implemented bits)? Which bits in the group are implemented?

   Answer: The IPR registers provide an 8-bit priority field for each interrupt. The processor implements only bits[7:6] of each field, (Core Manual, pg 73).

4. What was the latency between pushing the Discovery board button and the LED change (interrupt handler start) that you measured with the logic analyzer? Make sure to include a screenshot in the post-lab submission.

   Answer: The latency is about 0.4 [ms]. This can be seen in the top right corner of Figure 1.


Saleae Logic 1.2.11 - [Connected] - [16 MHz Digital, 1 s]

Annotations
Timing Marker Pair
| A1 - A2 | = 0.4 ms
A1 @ 0.398375 ms
A2 @ -1.625 μs
Analyzers

Figure 1: The logic analyzer GUI showing the time from button press to interrupt service, triggered at the button press and timed for 1 [ms].

5. Why do you need to clear status flag bits in peripherals when servicing their interrupts?

Answer: ”Most peripherals have a status register containing flag bits for pending interrupt requests; however, even in those without dedicated registers, most interrupts set status flags within their peripheral. These flags are necessary to generate interrupt requests. Typically you will need to clear the matching status bit manually for the interrupt condition that you are handling; otherwise, the interrupt will repeat continuously because the request never acknowledges as complete,” (Lab2.pdf, pg 11). Like the manual says, it is important otherwise the interrupt will be continuously serviced.