Chase Griswold

ECE 6780-003

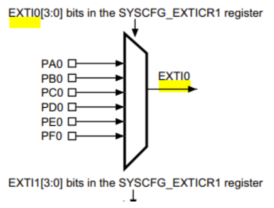
TA: Bailey Martin

Post-Lab Questions 02

2.2 — Postlab 2. Please answer the following questions and hand in as your postlab for Lab 2.

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

Figure 24. External interrupt/event GPIO mapping


EXTI0[3:0] bits in the SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
EXTI0
EXTI1[3:0] bits in the SYSCFG_EXTICR1 register

We can't use both pins for external interrupts at the same time because they are on the same multiplexer. EXTI0 is tied to all 0 pins A-F.

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

NVIC highest priority is Level 0, and lowest priority Level 3, giving four possible priority levels.

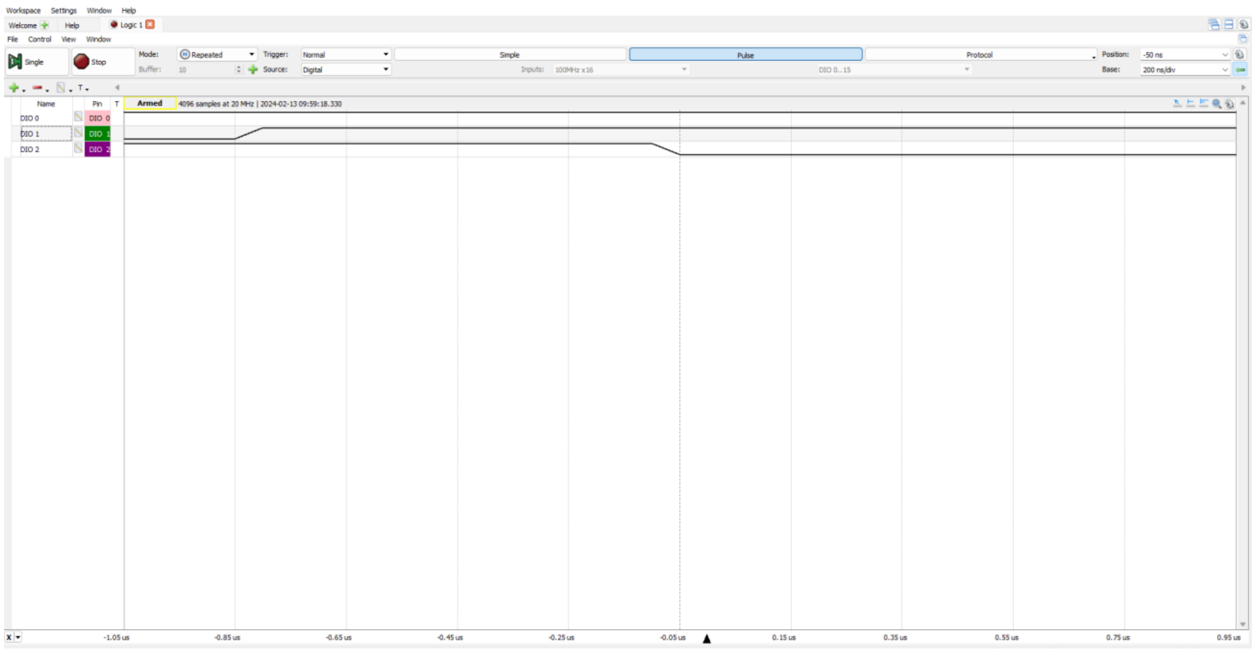
3. How many bits does the NVIC have reserved in its priority (IPR) registers for each interrupt (including non-implemented bits)? Which bits in the group are implemented?

The Interrupt Priority Registers (IPRs) on the STM32F0 configure interrupt priorities, each containing four 8-bit regions. However, the NVIC within the STM32F0 implements only the uppermost two bits from each region, resulting in four configurable priority levels (0-3). This effectively means that only bits [7:6] (the 2 most significant bits) of the 8-bit priority field provided by the NVIC IPR are utilized for setting interrupt priorities, while bits [5:0] are effectively disregarded as they are read as 0 and ignore writes.

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.

I ended up changing the Delay in the Handler function to 150,000 instead of 1.5 million so I could more easily capture the delay, which ended up being about ~0.80useconds. I used a setting of 200ms Pulse (rising edge trigger) on the Analog Discovery to get the capture.

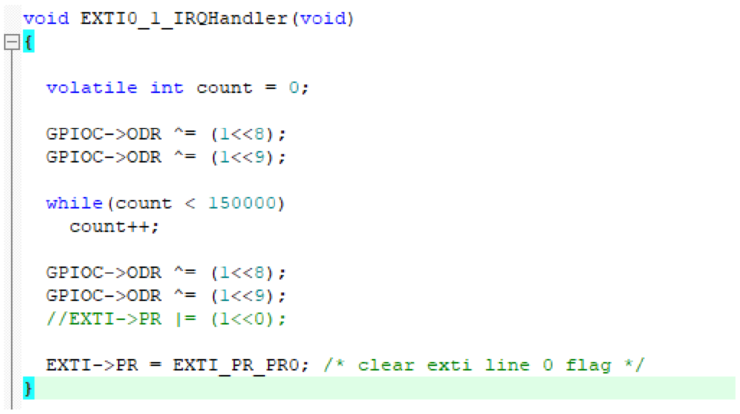
DIO 0 was tied to the user button, and DIO 1 was on PC8, and DIO 2 was on PC9.



Mode: Repeated
Trigger: Normal
Buffer: 10
Source: Digital
Inputs: 100MHz x16
Position: -50 ns
Base: 200 ns/div

Armed 4096 samples at 20 MHz | 2024-02-13 09:59:18.330

And this was the delay code in the handler I used (just for the screen capture, I changed it back to 1.5 million for submission).



```
void EXTI0_1_IRQHandler(void)
{

  volatile int count = 0;

  GPIOC->ODR ^= (1<<8);
  GPIOC->ODR ^= (1<<9);

  while(count < 150000)
    count++;

  GPIOC->ODR ^= (1<<8);
  GPIOC->ODR ^= (1<<9);
  //EXTI->PR |= (1<<0);

  EXTI->PR = EXTI_PR_PR0; /* clear exti line 0 flag */
}
```

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

If you don't clear the flag, it will constantly call the handler. That is the purpose of the flag.