Lab1 Report

ECE 4220

Chan Kim/chkz6d

9/17/20

**Objectives and Lab Description:**

There are two parts for this lab. One is to compile the code to Turn ON/OFF LEDs using Kernel Module and the other one is to detect Push-Button using an interrupt function. To be able to turn ON/OFF LEDs using Kernel Module, I have to access GPIO ports on the RaspberryPi. Next, in order to trigger the LEDs I write a code interrupt function to trigger a push button. After that, I write my own ISR to handle the interrupt. Familiarize with GPFSEL0,GPSET0,GPCLR0,GPAREN0,GPEDS0,GPPUD,GPPUDCLK0 and their address/bit/OFFSET.

**Implementation:**

For part 1, we simply need two functions for this. Function for to turn on LED and the other function for to turn off LED. In order to successfully execute this matter, I first configure GPSET0, and GPCLR0 as inputs and outputs through the GPFSEL0 register. Because I need to access the GPIO through the GPSET0, and GPCLR0 register. For part2, I mostly focused on Request\_irq and Free\_irq. I interrupt handling which means this function applies when the interrupt happens in my program, I tell the program to handle the problems. For free\_irq, this function removes an interrupt handler so basically a cleaning process. Whenever the program interrupt handler, it disables the interrupt first then I assign GPIO address afterwards. For initial module installation, I followed the steps

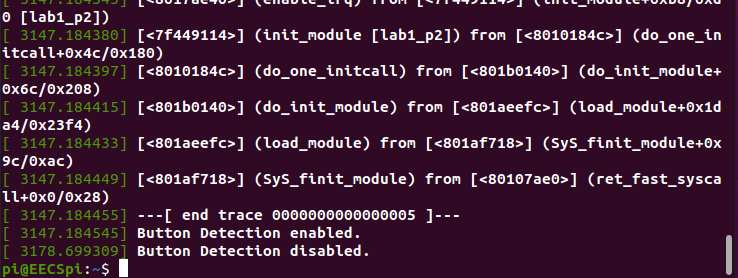
1. Set an input BCM 16-20
2. Configure to trigger interrupt
3. Set GPAREN0 bcm 16-20
4. ISR\_MY “Blind” (79)
5. Enable(79)

For final line, I simply free\_irq to free an interrupt allocated with request\_irq.

**Experiments and Results:**

To see if I have the right result, I make sure that I have the right address for GPIO registers by printing all the address and compare the last two digits of the address to Broadcom BCM2835 datasheet. On the part1 lab I found myself getting many errors due to carrying wrong address for GPIO registers, as well as the offset. I also tend to have wrong locations for bits so I run program more than 30 times to have the bits stay on the right location. Start it from initial module installation trigger the interrupt then clear up the irq.





**Discussion**

Discussion and Post-lab Questions

Through out this lab, I encounter so many errors and for many times. Using wrong address, having wrong location for bits, wrong offset value, and etc… This was easy to fix when I used printk and displaying the address to see if I have the right address for the GPIO registers. Most of the time I didn’t understand what I was doing in lab. I had so much trouble configuring the trigger to interrupt and didn’t even know why I needed to write them. This is my very first time using raspberry pi and I feel like I am always try to catch up with class.

From this lab, I learned how to configure inputs and outputs the GPFSEL register and access the GPIO ports through the GPSET,GPCLR. I also learn shifting bits to have the right location for bits to turn on the corresponding LED. It was a cool experience to learn a new materials.