ECE371 Log

**PART 1:**

11/14/2020

I first read and follow the instruction in the project design PDF and follow it. Step 1 is to read the BBB system manual and determine what GPIO connect to the 4 user LED and figure out the logic on how to turn it on. I first look at the table and content and look for LED as keyword then go through each page to find the information I want. I found what I’m looking for at page 67 which it lists a table of user LED 0-3 and what GPIO it uses. LED 0-3 is using GPIO1\_21-GPIO1\_24, and the LED is being control by GPIO1 port. After that I need to find what the logic to turn on the LED but I was confuse on how to find that since I’m confuse on what am I suppose to look for. I read page 159-165 in the textbook to see if can find some hint using the example in the book. I see some code in page 162-163 that have code trying to turn on GPIO1\_22, so I think I can copy it and see if it works. I copy those code into code composer and see if it works, I run the program, and nothing is turning on.

Version 1.0:

**.text**

**.global** \_start

**\_start:**

ldr R0,=0x02 @value to enable clock for a GPIO module

ldr R1,=0x44E00408 @address of CM\_WKUP\_GPIO0\_CLKCTRL register

str R0,[R1] @Write to register

ldr R1,=0x44E000AC @address of CM\_WKUP\_GPIO1\_CLKCTRL register

str R0,[R1] @Write to register

ldr R1,=0x44E000B0 @address of CM\_WKUP\_GPIO2\_CLKCTRL register

str R0,[R1] @Write to register

ldr R1,=0x44E000B4 @address of CM\_WKUP\_GPIO3\_CLKCTRL register

str R0,[R1] @Write to register

ldr R0,=0xFFBFFFFF @load word to program GPIO1\_22 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

str R2,[R1] @write back to GPIO1\_OE register

**mov** R5,0x00400000 @Load value to set GPIO1\_22 to logic high when out

ldr R6,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

str R5,[R6] @write to GPIO1\_SETDATAOUT register

ldr R0,=0xFFBFFFFF @load word to program GPIO1\_22 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

str R2,[R1] @write back to GPIO1\_OE register

.END

11/15/2020

After I fail to turn on the LED I decide that I should go back to the text book and reread to see if I miss anything. I found that I forgot to turn on the clock for the GPIO module which was show in page 159, so I copy the code that is on page 159 to my BBB but nothing turning on as I run the BBB. At first, I thought there is something wrong with my BBB since I really don’t know why my light not turning on. So, I reset the BBB by power it off and unplug all the core and reconnect everything back after letting it rest for few minutes. After I connect everything back the LED still will not turn on, so I decide to go back to the book and reread to see what I miss starting from page 159 again. I skip through the figure and focus on the text only for the pages trying to think as I read what did I mess up in my code since I feel like I didn’t miss anything. So, I thought maybe I should use this chance and figure out how the code that I have work, like how did they got the address and why they do these steps. Maybe this will help me understand on what I am missing, so start looking at each line of the code and figuring out how they got the address. I start with the enable clock for the GPIO module on page 159 and see how they got the address, as I read from 159 to 160 I still don’t quite get how they got the address but for now I’ll just accept that those are the address I need for the clock and to turn on the clock I write #0x02 to the address, and I realize that I shouldn’t be including the other GPIO in my code since I only need GPIO1. So, I did some code clean up by deleting the enabling of other GPIO that is not GPIO1 and then I run the program again with hope that it will run but nothing turning on.

Version 1.1:

**.text**

**.global** \_start

**\_start:**

ldr R0,=0x02 @value to enable clock for a GPIO module

ldr R1,=0x44E000AC @address of CM\_WKUP\_GPIO1\_CLKCTRL register

str R0,[R1] @Write to register

ldr R0,=0xFFBFFFFF @load word to program GPIO1\_22 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

str R2,[R1] @write back to GPIO1\_OE register

**mov** R5,0x00400000 @Load value to set GPIO1\_22 to logic high when out

ldr R6,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

str R5,[R6] @write to GPIO1\_SETDATAOUT register

ldr R0,=0xFFBFFFFF @load word to program GPIO1\_22 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

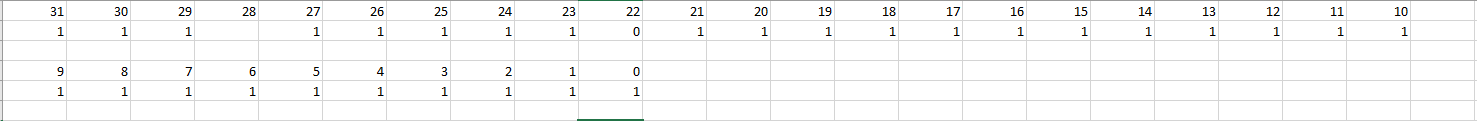
ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

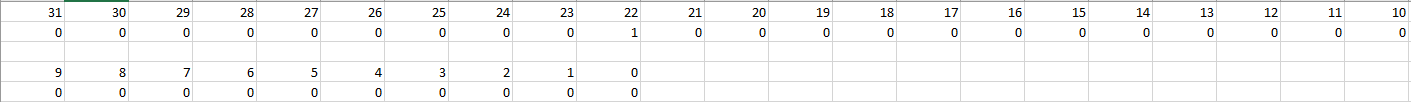
str R2,[R1] @write back to GPIO1\_OE register

.END

Then I go to page 162 which have the next part of my code and see how they got 0xFFBFFFFF for GPIO1\_22, and I start reading at page 161. Right there I notice figure 4-12 that have a table that say GPIO\_OE register data sheet and I look through that table little confuse on what it supposes to do. As I read on to figure 4-13, I realize what the table on figure 4-12 is for and see how they got the address for GPIO1\_22. It seems like you put 1 to all the bit but the 1 bit that correspond to the port you going to use for turning on the light. So, for GPIO1\_22 I have to put a 0 for bit 22 to make it and output. Then convert the 32 bits into hex which get me 0xFFBFFFFF for the address.



After that I look at the address of the GPIO1\_OE register then I found the answer in page 162 which it say that you take the base address of GPIO1 which is 0x4804C000 and you add 0x134 to it to make 0x4804C134. Then I go to page 163 next where they turn on the GPIO1\_22, then I look at the load value that put GPIO1\_22 to logic high and see how that work. I found out it work like how I find 0xFFBFFFFF but this time I put everything at 0 and only the register I want to turn on to 1 then convert it to hex which I got 0x00400000.



After that is LDR R6, =0x4804C194, I know that the 0x4804C000 come from the base GPIO address, but I do not recognize where that x194 come from. So, I start to flip from page 159 again and then I see a table in page 161 which have the GPIO control register on the bottom it have 194h which is the address of GPIO\_SETDATAOUT for the GPIO and I also see 134h which is the address for GPIO\_OE. Nothing look out of the ordinary, so I compile the program again and try to run it again but still nothing turns on.

11/16/2020

Since I fail to turn on the light ever since yesterday I decide that maybe it might be something wrong with my file so I decide to copy and paste the code to a note and then delete the project. Then I create a new project, and then copy and paste the code back this time. I load the program and instead of hitting the run button I decide to use the step through button. I step through each line of the file and when it hit STR R5,[R6] which is when you load the value that turn on GPIO1\_22 to high and you write it to the SETDATAOUT this time the light turn on. Seem like there was something wrong with the old file that I create, so with the LED1 turn on I decide that time to test if I can turn on LED0. I switch out the address that enable GPIO1\_22 to GPIO1\_21 then I also change the value that would make GPIO1\_21 set to high.

Version 1.1:

**.text**

**.global** \_start

**\_start:**

ldr R0,=0x02 @value to enable clock for a GPIO module

ldr R1,=0x44E000AC @address of CM\_WKUP\_GPIO1\_CLKCTRL register

str R0,[R1] @Write to register

ldr R0,=0xFFDFFFFF @load word to program GPIO1\_21 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

str R2,[R1] @write back to GPIO1\_OE register

**mov** R5,0x00200000 @Load value to set GPIO1\_21 to logic high when out

ldr R6,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

str R5,[R6] @write to GPIO1\_SETDATAOUT register

ldr R0,=0xFFBFFFFF @load word to program GPIO1\_22 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

str R2,[R1] @write back to GPIO1\_OE register

.END

I run this and the light from GPIO1\_22 turn off but the GPIO1\_21 never turn on which I was confuse then I decide to reverse the code back to original to see if it work but even when I revert it back it for some reason even the old code don’t work anymore.

11/17/2020

After meeting up with the TA, I learn how to solve the issue of why my file never run properly yesterday which is all I need is some little debugging by try to reconnect the device, unplugging to reset the device and try to rebuild from scratch. Also I make change to the code making it shorter, when I load the word to SETDATAOUT I don’t have to reread it out to GPIO1\_OE register anymore.

Version 1.2:

**.text**

**.global** \_start

**\_start:**

ldr R0,=0x02 @value to enable clock for a GPIO module

ldr R1,=0x44E000AC @address of CM\_WKUP\_GPIO1\_CLKCTRL register

str R0,[R1] @Write to register

ldr R0,=0xFFDFFFFF @load word to program GPIO1\_21 as output

ldr R1,=0x4804C134 @address of GPIO1\_OE

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @modify word read in

str R2,[R1] @write back to GPIO1\_OE register

**mov** R5,0x00200000 @Load value to set GPIO1\_21 to logic high when out

ldr R6,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

str R5,[R6] @write to GPIO1\_SETDATAOUT register

.END

I then add in the code to make GPIO1\_22 turn on also, and both light turn on when I add in the code.

Version 1.3:

**.text**

**.global** \_start

**\_start:**

**ldr** R0, #0x02 @value to enable clock for GPIO module

ldr R1, =0x44E000AC @address of CM\_PER\_GPIO clock

str R0,[R1] @wake up the clock

ldr R0,=0xFFDFFFFF @load word to GPIO21 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @Load address of GPIO1\_SETDATAOUT reg

str R5,[R6] @Write to GPIO1\_SETDATAOUT reg

ldr R0,=0xFFBFFFFF @load word to GPIO22 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00400000

ldr R6,=0x4804C194

str R5,[R6]

.END

Then while the thing still run the way I like, I add in both the code for turning on GPIO1\_23 and GPIO1\_24.

Version 1.4:

**.text**

**.global** \_start

**\_start:**

**mov** R0, #0x02 @value to enable clock for GPIO module

ldr R1, =0x44E000AC @address of CM\_PER\_GPIO clock

str R0,[R1] @wake up the clock

ldr R0,=0xFFDFFFFF @load word to GPIO21 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @Load address of GPIO1\_SETDATAOUT reg

str R5,[R6] @Write to GPIO1\_SETDATAOUT reg

ldr R0,=0xFFBFFFFF @load word to GPIO22 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00400000

ldr R6,=0x4804C194

str R5,[R6]

ldr R0,=0xFF7FFFFF @load word to GPIO23 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00800000

ldr R6,=0x4804C194

str R5,[R6]

ldr R0,=0xFEFFFFFF @load word to GPIO24 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x01000000

ldr R6,=0x4804C194

str R5,[R6]

I turn this and all the light turn on but they all instantly turning on not like GPIO1\_21 turn on then GPIO1\_22 and so forth. Then I decide to see if I can shorten the code down, so I start decide what if instead of enable each GPIO individually, I enable them all at once by using 0xF1EFFFFF. This way all the GPIO will be enable as an output then all I need to do is just turn them on to high.

version 1.5:

**.text**

**.global** \_start

**\_start:**

**mov** R0, #0x02 @value to enable clock for GPIO module

ldr R1, =0x44E000AC @address of CM\_PER\_GPIO clock

str R0,[R1] @wake up the clock

ldr R0,=0xF1EFFFFF @load word to GPIO21-24 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @Load address of GPIO1\_SETDATAOUT reg

str R5,[R6] @Write to GPIO1\_SETDATAOUT reg

**mov** R5,#0x00400000 @enable GPIO 22

ldr R6,=0x4804C194 @load the address of GPIO1\_SETDATAOUT

str R5,[R6] @Turn on the light by sent out the signal

**mov** R5,#0x00800000 @enable GPIO 23

ldr R6,=0x4804C194

str R5,[R6]

**mov** R5,#0x01000000 @enable GPIO 24

ldr R6,=0x4804C194

str R5,[R6]

**mov** R5,#0xF1EFFFFF @turn off all the GPIO 21-24

ldr r6,=0x4804C190 @load the address of CLEARDATAOUTPUT

str R5,[R6] @turn off all light

.END

I run the code, and everything turn on the way it should be.

11/19/2020

I add in a loop that would delay for 2 seconds before the next light turn on, and every light turn on after every 2 seconds. All four light turn on at the end but none of them turning off.

Version 1.6:

**.text**

**.global** \_start

**\_start:**

**LIGHT:**

**mov** R0, #0x02 @value to enable clock for GPIO module

ldr R1, =0x44E000AC @address of CM\_PER\_GPIO clock

str R0,[R1] @wake up the clock

ldr R0,=0xF1EFFFFF @load word to GPIO21 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @Load address of GPIO1\_SETDATAOUT reg

str R5,[R6] @Write to GPIO1\_SETDATAOUT reg

**mov** R7,#0x00400000

**loop:**

**nop**

subs R7,#1

bne loop

**mov** R5,#0x00400000 @enable GPIO 22

ldr R6,=0x4804C194 @load the address of GPIO1\_SETDATAOUT

str R5,[R6] @Turn on the light by sent out the signal

**mov** R7,#0x00400000

**loop1:**

**nop**

subs R7,#1

bne loop1

**mov** R5,#0x00800000 @enable GPIO 23

ldr R6,=0x4804C194

str R5,[R6]

**mov** R7,#0x00400000

**loop2:**

**nop**

subs R7,#1

bne loop2

**mov** R7,#0x00400000

**loop3:**

**nop**

subs R7,#1

bne loop3

**mov** R5,#0x01000000 @enable GPIO 24

ldr R6,=0x4804C194

str R5,[R6]

**nop**

**mov** R7,#0x00400000

**loop4:**

**nop**

subs R7,#1

bne loop4

**mov** R5,#0xF1EFFFFF @turn off all the GPIO 21-24

ldr r6,=0x4804C190 @load the address of CLEARDATAOUTPUT

str R5,[R6] @turn off all light

**nop**

b LIGHT

.END

11/23/22

I run version 1.6 again today just to make sure it is running and all the light turning on all at once. The light should be turning on one by one every 2 seconds not all of them turning on at once. I try to add back in the enable instruction for each GPIO1 to make sure, but that did not fix it still. All the light still all turning on, so I reset the device again and plug back all the power but it still didn’t not fix the problem and now I’m running in the same problem I been having which is the code not running. I been having this problem ever since the project start which everything work one day and then nothing work the next day and no code was running. I decide that I should do something about this problem, so I decide to unplug everything again and try a different port for the USB cable this time which I use the back port instead of the front port. Once I have the port switch everything is running perfectly after that, I unplug again and use the old port this time and the same problem arise which is the code is not running. So I use the back port again and this time the program run perfectly still.

Version 1.7:

**.text**

**.global** \_start

**\_start:**

**LIGHT:**

**mov** R0, #0x02 @value to enable clock for GPIO module

ldr R1, =0x44E000AC @address of CM\_PER\_GPIO clock

str R0,[R1] @wake up the clock

ldr R0,=0xFFDFFFFF @load word to GPIO21 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @Load address of GPIO1\_SETDATAOUT reg

str R5,[R6] @Write to GPIO1\_SETDATAOUT reg

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop @loop until R7 is 0

**mov** R5,#0xFFDFFFFF @set GPIO 21 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 21

ldr R0,=0xFFBFFFFF @enable GPIO22

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00400000 @enable GPIO 22

ldr R6,=0x4804C194 @load the address of GPIO1\_SETDATAOUT

str R5,[R6] @Turn on the light by sent out the signal

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop1:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop1 @loop until R7 is 0

ldr R0,=0xFF7FFFFF @Enable GPIO23

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00800000 @enable GPIO 23

ldr R6,=0x4804C194

str R5,[R6]

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop2:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop2 @loop until R7 is 0

ldr R0,=0xFEFFFFFF @enable GPIO24

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x01000000 @enable GPIO 24

ldr R6,=0x4804C194

str R5,[R6]

**nop**

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop3:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop3 @loop until R7 is 0

**mov** R5,#0xF1EFFFFF @set GPIO 21 - 24 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 24

@display from 24 - 21 after all LED is shut off

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop4:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop4 @loop until R7 is 0

ldr R0,=0xFEFFFFFF @enable GPIO24

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x01000000 @enable GPIO 24

ldr R6,=0x4804C194

str R5,[R6]

**nop**

**mov** R5,#0xF1EFFFFF @set GPIO 21 - 24 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 24

.END

I run it a couple time to make sure that the code is running and all the light turning on one by one every 2 seconds. Then after I make sure that all the light turning from GPIO1\_21-GPIO1\_24, I then make it so that it light up from GPIO1\_24 – GPIO1\_21.

Version 1.8:

**.text**

**.global** \_start

**\_start:**

**LIGHT:**

**mov** R0, #0x02 @value to enable clock for GPIO module

ldr R1, =0x44E000AC @address of CM\_PER\_GPIO clock

str R0,[R1] @wake up the clock

ldr R0,=0xFFDFFFFF @load word to GPIO21 as output

ldr R1,=0x4804C134 @address of GPIO\_OE register

ldr R2,[R1] @read GPIO1\_OE register

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @Load address of GPIO1\_SETDATAOUT reg

str R5,[R6] @Write to GPIO1\_SETDATAOUT reg

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop @loop until R7 is 0

**mov** R5,#0xFFDFFFFF @set GPIO 21 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 21

ldr R0,=0xFFBFFFFF @enable GPIO22

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00400000 @enable GPIO 22

ldr R6,=0x4804C194 @load the address of GPIO1\_SETDATAOUT

str R5,[R6] @Turn on the light by sent out the signal

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop1:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop1 @loop until R7 is 0

**mov** R5,#0xFFBFFFFF @set GPIO 22 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 22

ldr R0,=0xFF7FFFFF @Enable GPIO23

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00800000 @enable GPIO 23

ldr R6,=0x4804C194

str R5,[R6]

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop2:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop2 @loop until R7 is 0

**mov** R5,#0xFF7FFFFF @set GPIO 23 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 23

ldr R0,=0xFEFFFFFF @enable GPIO24

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x01000000 @enable GPIO 24

ldr R6,=0x4804C194

str R5,[R6]

**nop**

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop3:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop3 @loop until R7 is 0

**mov** R5,#0xFEFFFFFF @set GPIO 24 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 24

@display from 24 - 21 after all LED is shut off

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop4:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop4 @loop until R7 is 0

ldr R0,=0xFEFFFFFF @enable GPIO24

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x01000000 @enable GPIO 24

ldr R6,=0x4804C194

str R5,[R6]

**nop**

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop5:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop5 @loop until R7 is 0

**mov** R5,#0xFEFFFFFF @set GPIO 24 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 24

ldr R0,=0xFF7FFFFF @Enable GPIO23

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00800000 @enable GPIO 23

ldr R6,=0x4804C194

str R5,[R6]

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop6:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop6 @loop until R7 is 0

**mov** R5,#0xFF7FFFFF @set GPIO 23 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 23

ldr R0,=0xFFBFFFFF @enable GPIO22

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00400000 @enable GPIO 22

ldr R6,=0x4804C194 @load the address of GPIO1\_SETDATAOUT

str R5,[R6] @Turn on the light by sent out the signal

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop7:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop7 @loop until R7 is 0

**mov** R5,#0xFFBFFFFF @set GPIO 22 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 22

ldr R0,=0xFFDFFFFF @enable GPIO21

**and** R2,R2,R0 @mod word read in

str R2,[R1] @write back to GPIO1\_OE

**mov** R5,#0x00200000 @enable GPIO 21

ldr R6,=0x4804C194 @load the address of GPIO1\_SETDATAOUT

str R5,[R6] @Turn on the light by sent out the signal

**mov** R7,#0x00400000 @set 2 seconds delay by having 40k ns

**loop8:**

**nop**

subs R7,#1 @minus 1ns from 40k ns

bne loop8 @loop until R7 is 0

**mov** R5,#0xFFDFFFFF @set GPIO 21 to turn off

ldr R6,=0x4804C190 @sent the data to CLEARDATAOUT

str R5,[R6] @turn off GPIO 21

b LIGHT

.END

I run this a few times to see if the light goes from GPIO1\_21 to GPIO1\_24 and then from 24 back to 21. After I confirm that everything running the way that it should be, I unplug everything and then wait for a few hours. I plug everything back and open compose studio and run the code again and this time it passes all my test.

**PART 2**

11/25/2020

I build the circuit base on the TA Tuesday open lab which he show me during the lab. After I got the circuit build, I start to copy down the code from page and all I change was the GPIO address from 12 to 21. I create a circuit that fit the description which is a 47k resistor between a button and the power source that will link the whole circuit to ground once the button is press. Since I don’t have an 47k resistor I have to connect a series of resistor to create an 47k, and then I hook it up with the button. I look up on how the button work and connect the power source and the ground base on how the button are layout. After I got the circuit layout I start to copy the code from the book into Code Composer. Then I start mark down place that say GPIO1\_12 and GPIO1\_14 in the book code, I make change to those places. There are only 6 places where I need to make change to which is the very first place is mov R7, #0x00001000 which I change the address of GPIO1\_12 to GPIO1\_21 so mov R7, #0x00200000. Then after that is where they enable GPIO1\_12, LDR R7, 0xFFFFEFFF instead of having it enable 12 I change it so that it enable GPIO1\_21 instead by changing the address to LDR R7,0xFFDFFFFF. The next place is the falling edge register, for Part2 we need to use GPIO1\_30 because we using pin 21 on the BBB which is link to GPIO1\_30. So instead of having it on MOV R2, #0x00004000. I find the correct value by using the table on Part1 that I use to find the value of the GPIO1\_21-24, which is flip the bit to 1 to use it as an input and 0 as an output. So I change from MOV R2,#0x00004000 to MOV R2, #0x40000000 which is the address to make GPIO1\_30 the input that will detect the falling edge. The next place to change is in the INT\_DIRECTOR where you testing the bit 14 instead of testing bit 14 I want to test bit 21 instead since that the port we be detecting the change which is also GPIO1\_30. The change would be from TST R1, #0x00004000 to TST R1, #0x40000000 which will be GPIO1\_30 bit 21. The next place I need to make change to that I have mark is on the BUTTON\_SVC where instead of turning off GPIO1\_14 interrupt request. I want it to turn off GPIO1\_30 interrupt request instead of GPIO1\_14 so I change MOV R1,#0x00004000 to MOV R1, #0x40000000. Finally, is still within BUTTON\_SVC is I need to change that from loading GPIO1\_12 to SETDATAOUT I need to load GPIO1\_21 instead. Which the change I make is MOV R1, #0x00001000 to MOV R1, #0x20000000. All the change I need to make for Part2, I then start copy each line of the code into Code Composer.

Version 1.0:

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

LDR R13,=STACK1 @Point to base of STACK for svc mod

**ADD** R13,R13,#0x1000 @Point to top of STACK

CSP #0x12 @Switch to IRQ mode

LDR R13,=STACK2 @Point to IRQ mode

**ADD** R13,R13,#0x1000 @Point to top of STACK

CPS #0x13 @Back to SVC mode

@Turn on GPIO CLK

LDR R0,=0x02 @Enable clock for GPIO

LDR R1,=0x44E00A0C @Address of GPIO1\_CLKCTRL register

STR R0,[R1] @Enable GPIO1

@Turn off GPIO21 just to make sure that the light is off when the program first run

LDR R0,=0x4804C000 @Base address for GPIO1

**ADD** R4, R0,#0x190 @Load in the address of CLEARDATAOUT by adding the base to 0x190

**MOV** R7,#0x00200000 @Address of GPIO1\_21

STR R7,[R4] @Turn off GPIO\_21

@make GPIO1\_21 as an output

**ADD** R1,R0,#0x0134 @Make the GPIO1\_OE register address

LDR R6,[R1] @READ current GPIO1 output Enable register

LDR R7,=0xFFDFFFFF @word to enable GPIO1\_21 as output

**AND** R6,R7,R6 @clear bit 21

STR R6,[R1] @write to GPIO1 output register

@Detect falling edge on GPIO1\_30 which is pin 21

**ADD** R1,R0,#0x14C @R1 = address of GPIO1\_FALLINGDETECT register

**MOV** R2,#0x40000000 @Load value for bit 30

LDR R3,[R1] @Read GPIO1\_FALLINGDETECT register

ORR R3,R3,R2 @Modify (set bit 21)

STR R3,[R1] @Write back

**ADD** R1,R0,#0x34 @Address of GPIO1\_IRQSTATUS\_SET\_0 register

STR R2, [R1] @Enable GPIO1\_21 request on POINTRPEND1

@initialize INTC

LDR R1,=0x482000E8 @Address of INTC\_MIR\_CLEAR3 register

**MOV** R2,#0x04 @value to unmask INTC INT 98, GPIOINT1A

STR R2,[R1] @Write to INTC\_MIR\_CLEAR3 register

@Make sure processor IRQ enabled in CPSR

MRS R3,CPSR @Copy CPSR to R3

BIC R3,#0x80 @clear bit 7

MSR CPSR\_c,R3 @Write back to CPSR

@Wait for interrupt

**Loop:** **NOP**

**B** Loop

**INT\_DIRECTOR:**

STMFD SP!,{R0-R3,LR} @Push registers on stack

LDR R0,=0x482000F8 @Address of INTC-PENDING\_IRQ3 register

LDR R1,[R0] @read INTC-PENDING\_IRQ3 register

TST R1,#0x00000004 @test bit 2

BEQ PASS\_ON @Not from GPIOINT1A, go to back to wait loop, else

LDR R0,=0x4804C02C @load GPIO1\_IRQSTATUS\_0 register address

LDR R1,[R0] @read STATUS register

TST R1,#0x40000000 @Check if bit 21 = 1

BNE BUTTON\_SVC @if bit 21 = 1, then button pushed

BEQ PASS\_ON @if bit 21 = 0, then go back to wait loop

**PASS\_ON:**

LDMFD SP!,{R0-R3,LR} @restore register

SUBS PC,LR,#4 @pass execution on to wait Loop for now

**BUTTON\_SVC:**

**MOV** R1,#0x40000000 @Value turns off GPIO1\_21 interrupt request and also turn off INTC interrupt request

STR R1,[R0] @write to GPIO1\_IRQSTATUS\_0 register

@Turn off NEWIRQA bit in INTC\_CONTROL, so processor can respondto new IRQ

LDR R0,=0x48200048 @address of INTC\_CONTROL register

**MOV** R1,#01 @value to clear bit 0

STR R1,[R0] @write to INTC\_CONTROL register

@Turn on LED on GPIO1\_21

LDR R0,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

**MOV** R1,#0x00200000 @load address of GPIO1\_21 to turn on

STR R1,[R0] @write to GPIO1\_21SETDATAOUT register

@wait for 2 seconds

**MOV** R2,#0x00400000

**Loop1:**

**NOP**

SUBS R2,#1

BNE Loop1

@Turn off the LED GPIO1\_21

LDR R0,=0x4804C190 @load address of GPIO1\_CLEARDATAOUT register

STR R1,[R0] @write 0x00200000 GPIO1\_21 turn on address to GPIO1\_CLEARDATAOUT

@return to wait loop

LDMFD SP!,{R0-R3,LR} @restore refisters

SUBS PC,LR,#4 @return from IRQ interrupt procedure

**.align** 2

**SYS\_IRQ:** .WORD 0 @location to store system IRQ address

**.data**

**.align** 2

**STACK1:** .rept 1024

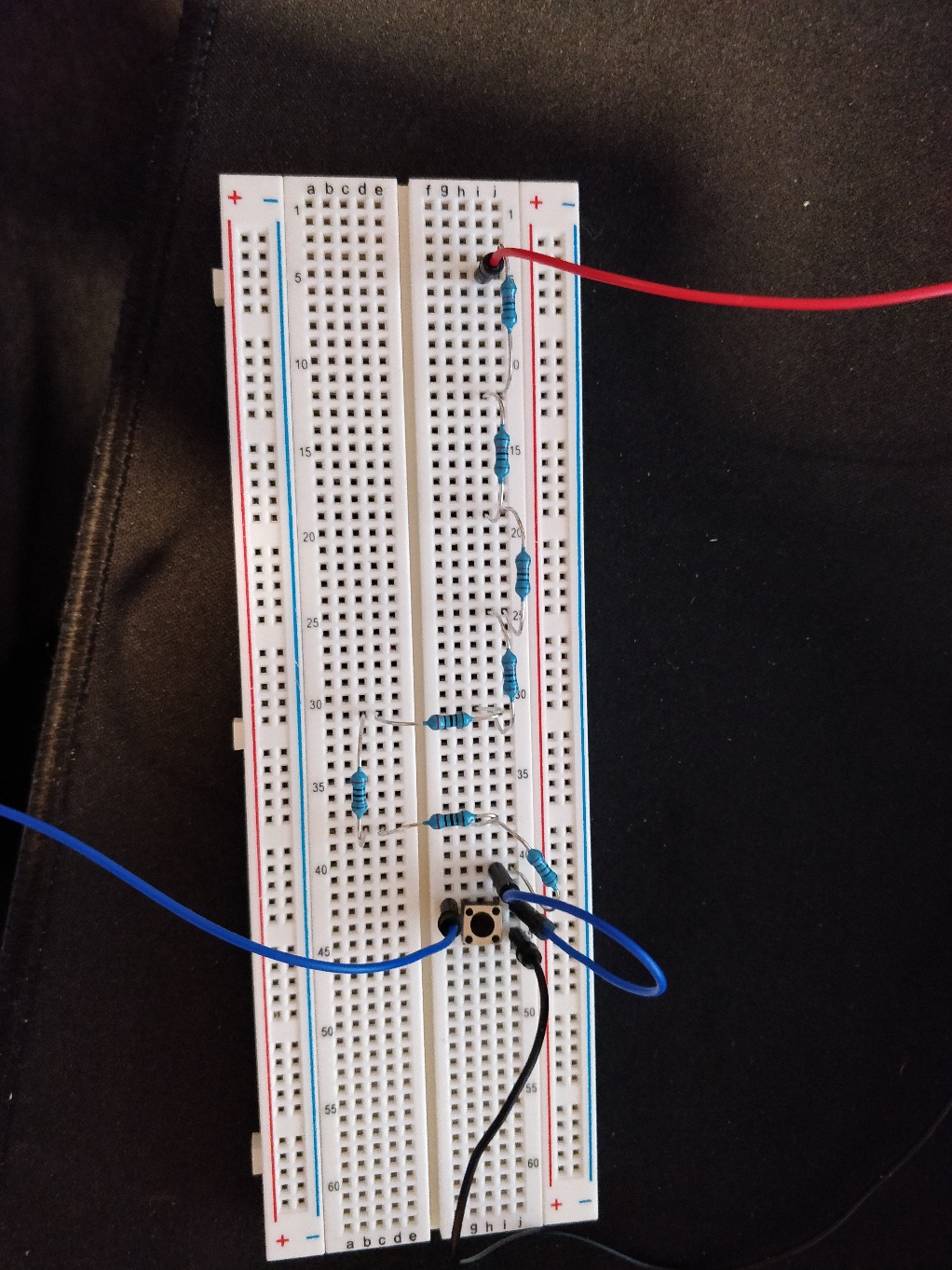
**.word** 0x0000

.endr

**STACK2:** .rept 1024

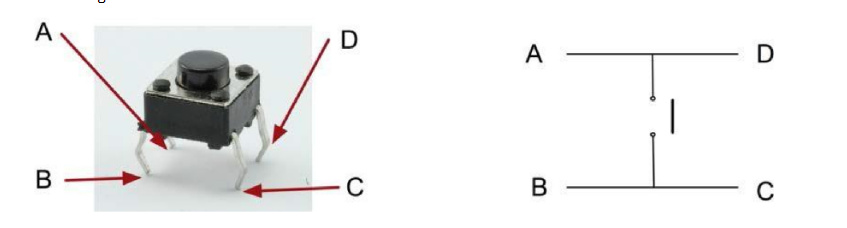
**.word** 0x0000

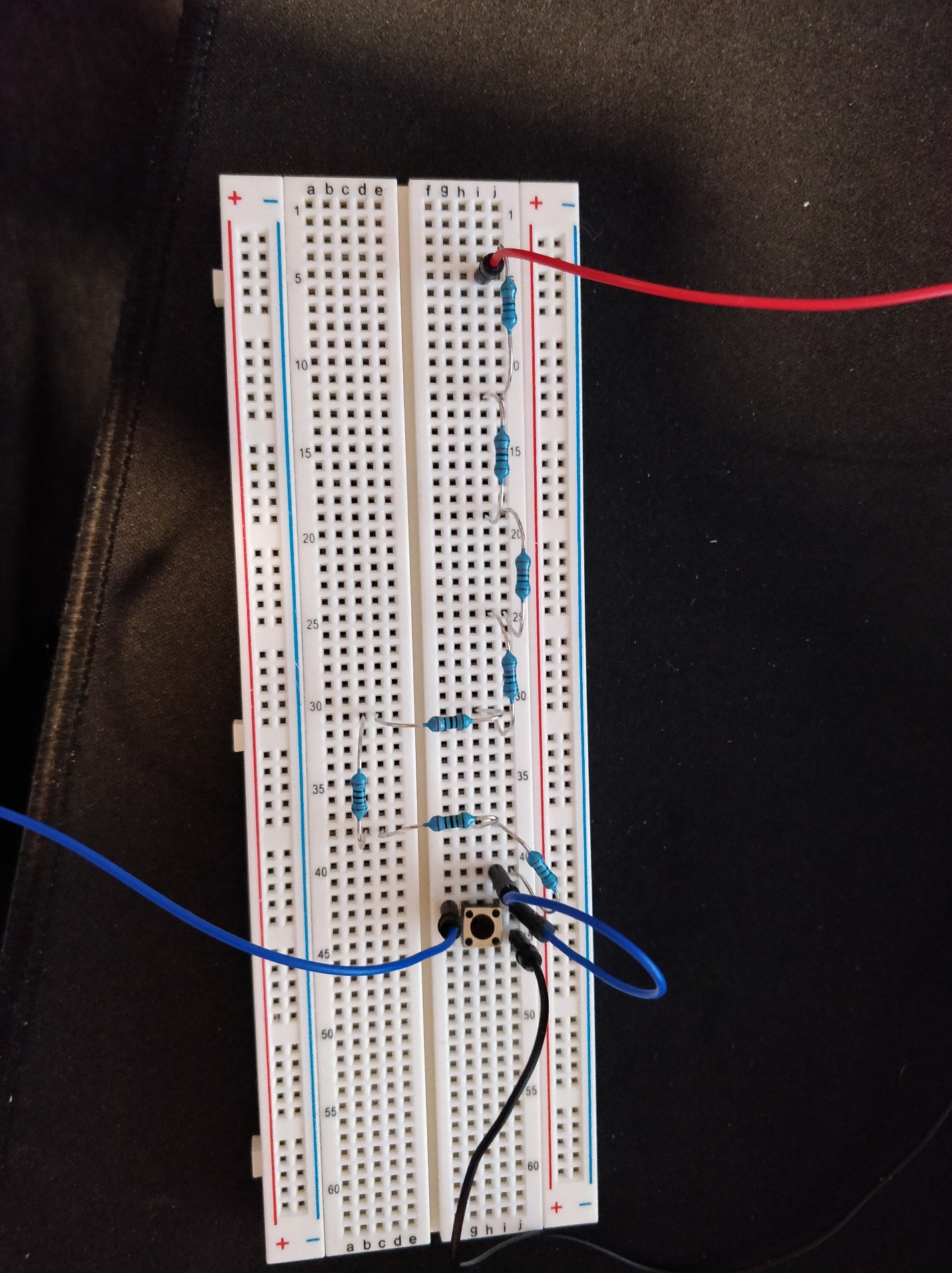
.endr



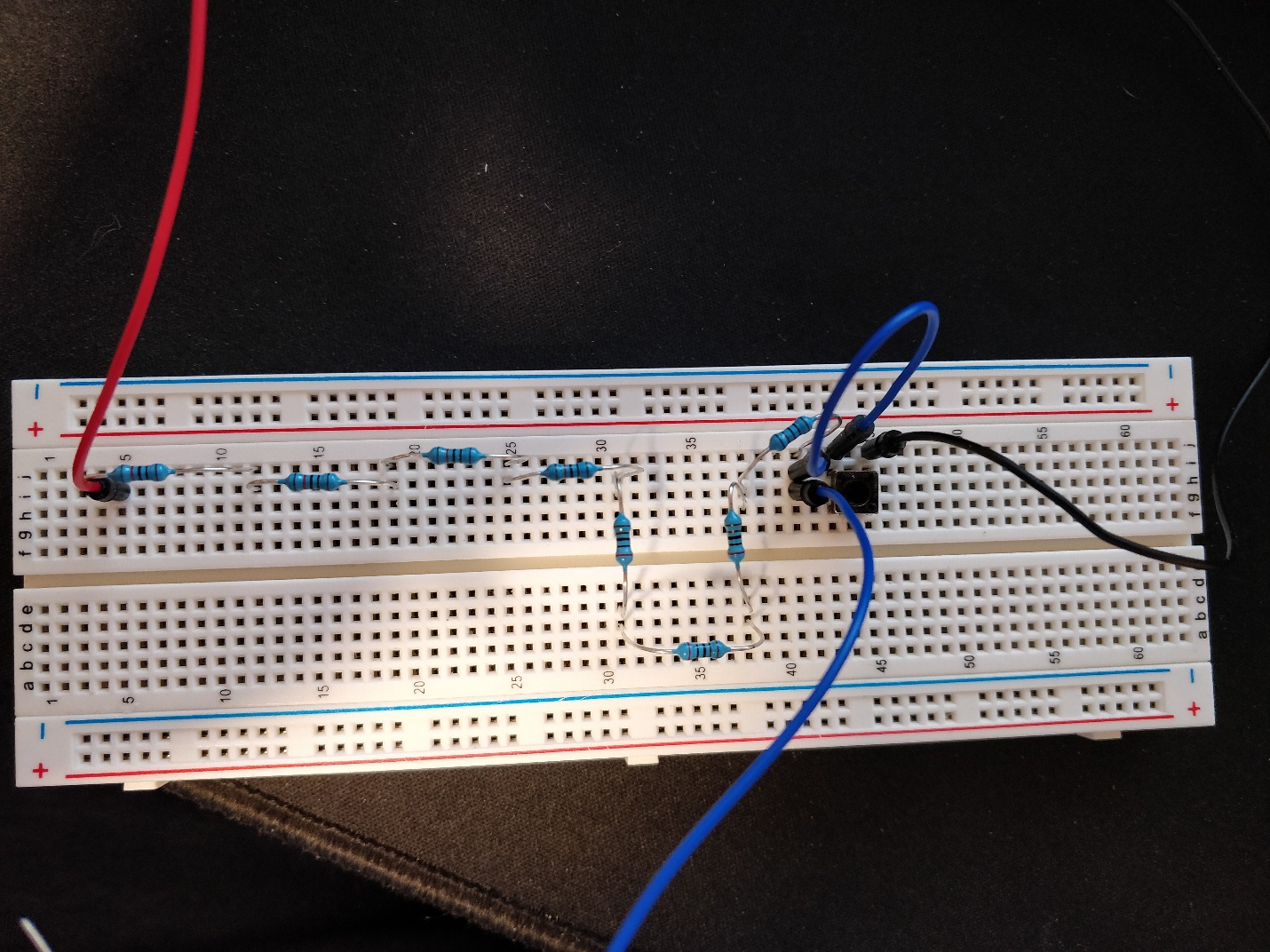
11/26/2020

I try to run the code with the circuit after got them both finish and when I compile I got an compile error of unreferenced to INT\_DIRECTION which I don’t know why. I start looking through my main and go down to where I create the INT\_ DIRECTOR function to see if there anything wrong with it, but there was nothing wrong with it. So, then it hit me that I must have spell the function wrong, so I went to the start up file and look at where I hook up the interrupt. Apparently, there was a misspelling from me which the function should be INT\_INDIRECTOR not INT\_INDIRECTION so I got it fix and everything compile again with no bug. I run the code again and nothing turning on and off, so I first try to look at my circuit and see if there anything wrong with it. I use an electrometer to try and measure the voltage in the circuit to make sure that I am correctly create my circuit. When I try to get a reading from the blue and ground circuit I got no reading, but I do have a reading of 3.3V when I try to read between my red and black. Which make me think that I must have connect the circuit wrong for the blue line, so I decide to fix it. I pull up the button circuit again to see if I did something wrong.

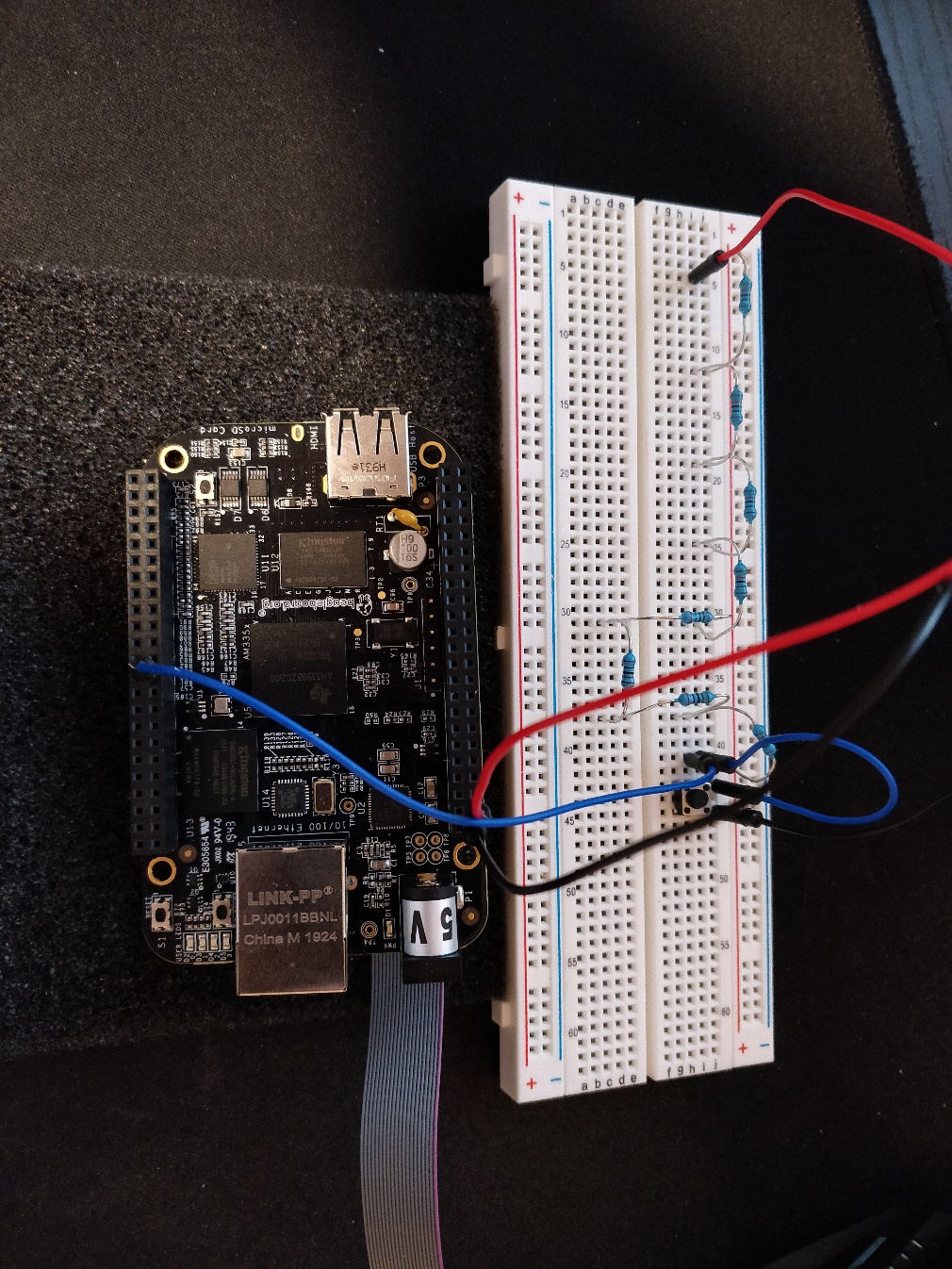


I also watch the video on Zoom of TA office hour to make sure that I did the circuit correctly. Which it turns out I did my circuit wrong as I watching the lecture, what I did wrong was my button connection. The blue line is not supposed to be connect to B point like how I have it set up. 

That blue line on B should be connect to A not B, so I switch where that blue line suppose to be .



Which is there, I measure the voltage between that and the black line again, and this time I got some reading. That confirm my circuit finally working with there is now voltage through that point. I press the button and the voltage go to 0 which further confirm that the circuit is working the way it should be. After that I hook everything back up to the BBB again and run my code again, but the LED0 still will not turn on. I go back to the circuit and measure every point again to make sure that there is absolutely nothing wrong with the circuit, and the circuit pass all my test again. So, which conclude to me that the only thing that could be wrong is the code that is the reason since I know it not the circuit anymore. I start looking at the code to see if there anything wrong with it, but I cannot spot anything wrong with it. I then start looking around the part that I make change to the address since I only have made a few changes to the code from it original. I start looking at the GPIO1\_21 address and the GPIO\_30 which is the pin. Then I start looking at the address I sent to SETDATAOUT but nothing looks wrong to me. Everything is correct since I make sure to check the address of Part 2 to my Part 1 of the project to make sure that I use the correct address. The next thing I did was I start looking through the code from top to bottom to make sure that I didn’t mess up on any spelling or miss something when I was copy from the code in the book. Then I see that within the first 10 line in the \_start I already mess up in the code which is the CPS but I spell it wrong. I then start to go through my whole code again line by line and I spotted that I have make a lot of mistake in copying the code. There part where I miss spell and type in the wrong address which is probably the reason why the code isn’t working. After I make sure that nothing is wrong with the code anymore and that everything has been fix in the code. I load the program in again and run the code then I press the button and LED0 turn on and then off after 2 seconds. I have finally complete Part 2 of the project and it turn out the whole time it was my circuit was being wrong and that there is typo in the code that I copy. I press the button a few more time to make sure that the code is running correctly and the LED0 turn on and off after 2 seconds every time. So, with this I conclude that my Part 2 is complete.



Version 1.1:

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

LDR R13,=STACK1 @Point to base of STACK for svc mod

**ADD** R13,R13,#0x1000 @Point to top of STACK

CPS #0x12 @Switch to IRQ mode

LDR R13,=STACK2 @Point to IRQ mode

**ADD** R13,R13,#0x1000 @Point to top of STACK

CPS #0x13 @Back to SVC mode

@Turn on GPIO CLK

LDR R0,=0x02 @Enable clock for GPIO

LDR R1,=0x44E000AC @Address of GPIO1\_CLKCTRL register

STR R0,[R1] @Enable GPIO1

@Turn off GPIO21 just to make sure that the light is off when the program first run

LDR R0,=0x4804C000 @Base address for GPIO1

**ADD** R4, R0,#0x190 @Load in the address of CLEARDATAOUT by adding the base to 0x190

**MOV** R7,#0x00200000 @Address of GPIO1\_21

STR R7,[R4] @Turn off GPIO\_21

@make GPIO1\_21 as an output

**ADD** R1,R0,#0x0134 @Make the GPIO1\_OE register address

LDR R6,[R1] @READ current GPIO1 output Enable register

LDR R7,=0xFFDFFFFF @word to enable GPIO1\_21 as output

**AND** R6,R7,R6 @clear bit 21

STR R6,[R1] @write to GPIO1 output register

@Detect falling edge on GPIO1\_30 which is pin 21

**ADD** R1,R0,#0x14C @R1 = address of GPIO1\_FALLINGDETECT register

**MOV** R2,#0x40000000 @Load value for bit 30

LDR R3,[R1] @Read GPIO1\_FALLINGDETECT register

ORR R3,R3,R2 @Modify (set bit 21)

STR R3,[R1] @Write back

**ADD** R1,R0,#0x34 @Address of GPIO1\_IRQSTATUS\_SET\_0 register

STR R2, [R1] @Enable GPIO1\_21 request on POINTRPEND1

@initialize INTC

LDR R1,=0x482000E8 @Address of INTC\_MIR\_CLEAR3 register

**MOV** R2,#0x04 @value to unmask INTC INT 98, GPIOINT1A

STR R2,[R1] @Write to INTC\_MIR\_CLEAR3 register

@Make sure processor IRQ enabled in CPSR

MRS R3,CPSR @Copy CPSR to R3

BIC R3,#0x80 @clear bit 7

MSR CPSR\_c,R3 @Write back to CPSR

@Wait for interrupt

**Loop:** **NOP**

**B** Loop

**INT\_DIRECTOR:**

STMFD SP!,{R0-R3,LR} @Push registers on stack

LDR R0,=0x482000F8 @Address of INTC-PENDING\_IRQ3 register

LDR R1,[R0] @read INTC-PENDING\_IRQ3 register

TST R1,#0x00000004 @test bit 2

BEQ PASS\_ON @Not from GPIOINT1A, go to back to wait loop, else

LDR R0,=0x4804C02C @load GPIO1\_IRQSTATUS\_0 register address

LDR R1,[R0] @read STATUS register

TST R1,#0x40000000 @Check if bit 21 = 1

BNE BUTTON\_SVC @if bit 21 = 1, then button pushed

BEQ PASS\_ON @if bit 21 = 0, then go back to wait loop

**PASS\_ON:**

LDMFD SP!,{R0-R3,LR} @restore register

SUBS PC,LR,#4 @pass execution on to wait Loop for now

**BUTTON\_SVC:**

**MOV** R1,#0x40000000 @Value turns off GPIO1\_21 interrupt request and also turn off INTC interrupt request

STR R1,[R0] @write to GPIO1\_IRQSTATUS\_0 register

@Turn off NEWIRQA bit in INTC\_CONTROL, so processor can respondto new IRQ

LDR R0,=0x48200048 @address of INTC\_CONTROL register

**MOV** R1,#01 @value to clear bit 0

STR R1,[R0] @write to INTC\_CONTROL register

@Turn on LED on GPIO1\_21

LDR R0,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

**MOV** R1,#0x00200000 @load address of GPIO1\_21 to turn on

STR R1,[R0] @write to GPIO1\_21SETDATAOUT register

@wait for 2 seconds

**MOV** R2,#0x00400000

**Loop1:**

**NOP**

SUBS R2,#1

BNE Loop1

@Turn off the LED GPIO1\_21

LDR R0,=0x4804C190 @load address of GPIO1\_CLEARDATAOUT register

STR R1,[R0] @write 0x00200000 GPIO1\_21 turn on address to GPIO1\_CLEARDATAOUT

@return to wait loop

LDMFD SP!,{R0-R3,LR} @restore refisters

SUBS PC,LR,#4 @return from IRQ interrupt procedure

**.align** 2

**SYS\_IRQ:** .WORD 0 @location to store system IRQ address

**.data**

**.align** 2

**STACK1:** .rept 1024

**.word** 0x0000

.endr

**STACK2:** .rept 1024

**.word** 0x0000

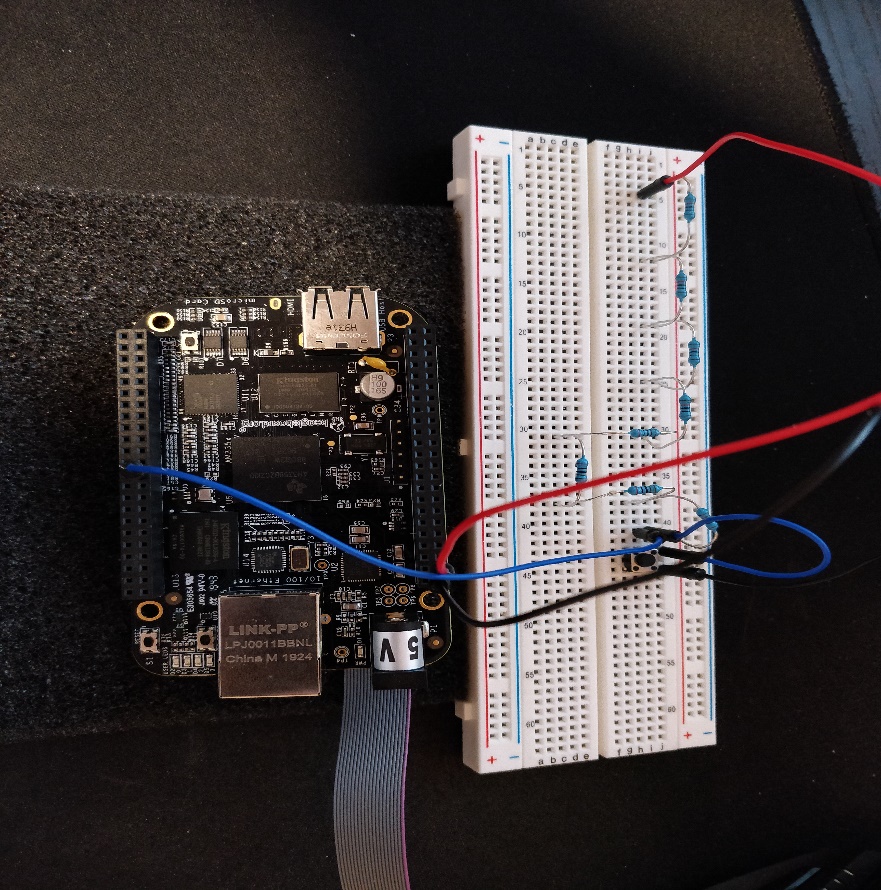
.endr

.END

**PART 3**

12/5/2020

Since Part 3 is like Part 2 of the project so I decide that first I would start by finding the address of everything first and make the change in the book. Then I would start to type everything from the book to code composer. So, the very first thing I do is I scan the code from the book and start marking where I need to make change. How I decide what is to change is I mark the line where the comment say GPIO1\_12, GPIO1\_14, Timer2 any line have that in the comment I mark them all down. Then I read page 238-241trying to understand how they got the address the timer5 and everything. The first change I make is that instead of enabling GPIO1\_12 I enable GPIO1\_21 instead so LDR R7, =0xFFFFEFFF to LDR R7, =0xFFDFFFFF. The next place is to have it to use GPIO1\_30 instead of 14 so I change the value from MOV R2, #0x40000000. After that I must find the unmask INT for timer5 instead of timer 4, to find this number I read page 240 where it tells me to look for the number in Appendix D in the book which is page 279 in the book. The INT I am looking for is 93 for timer5, and then I determine that INT93 is in INTC\_MIR\_CLEAR2 because 96-127 is for CLEAR3 and that 64-95 is for CLEAR2 since 93 is within CLEAR2. I do not need to change the address for that in the book, but I only must change MOV R2, #0x10 to unmask bit 29 so the change would be MOV R2, #0x20000000 for bit 29 which correspond to INT 93. After that is at Turn on timer2, I have it instead of turning on timer2 I change it so that it will be turning on timer5 instead since that the requirement. To do that I first change the CM\_PER\_TIMER2\_CLKCTRL to CM\_PER\_TIMER5\_CLKCTRL by looking up the address of it in the Sitara Data book. I use CTRL + f to bring up the search function and then type in what I need to search for. The search takes me straight to CM\_PER\_TIMER5\_CLKCTRL address, which is 0x44E000EC once I have the address, I replace LDR R1,0x44E00080 with LDR R1,0x44E000EC. Next change will be the address of PRCMCLKSEL\_TIMER2, I read page 241 for this and it tell me to look for CM\_DPLL which I type that in the search bar and then start going down the search. The first thing I see was the base address of CM\_DPLL which is 0x44E00500, and when I go down by one more, I see the CM\_DPLL register and then I look for timer5 which I found on the table. Timer5 have an offset of 0x18 which all I need to do was combine that with the base address to make 0x44E00518, so I change LDR R1,0x44E00508 to LDR R1, 0x44E00518. Next I need to find the base address of timer5 instead of timer2, I found it on page 273 on the book on the L4\_PER Peripheral memory Map table and I see DMTIMER5 on it that have the address of 0x48046000. So then I change LDR R1, =0x48040000 which is the base address of timer2 to LDR R1, =0x48046000 which is the base address of timer5. The next place I need to change is in the INT\_DIRECTOR which instead of having it test bit 14 I change it to be testing bit 30 for GPIO1\_30 so from TST R1, #0x00000004 to TST R1, #0x40000000. Next is in the TCHK, instead of the address of timer2 IRQSTATUS register I want timer5 instead and all I need to do is take the base address of timer5 which is 0x48046000 + the offset of 0x28 which is the IRQSTATUS register so LDR R1, 0x48040028 to LDR R1, #0x48046028. Then I move down to BUTTON\_SVC which there are 3 things that needs to be change, and the first thing is have it change from loading GPIO1\_12 to SETDATAOUT to GPIO1\_21. So, I switch MOV R1, 0x00001000 to MOV R1, #0x00200000. Second, I must change the load value that set the timer to reload and start which make the light keep on blinking every 0.5 second to just have it to wait for 0.5, and to fix the blinking problem I just must change the value from #0x03 to #0x01 so MOV R2, #0x03 to MOV R2, #0x01. Third, I must change the address of timer2 TCLR register to timer5 instead, it was easy to be done as the offset for the TCLR register is 0x38 and so all I need to do is add it to the base address of Timer5 which make 0x48046000+0x38 so from LDR R1, =0x48040038 to LDR R1, =0x48046038. Last place I need to make the change is on LED which instead of loading the address of timer2 IRQSTATUS I must switch 0x48040028 to 0x48046028. The address of timer5 offset for IRQSTATUS can be find on page 281 of the book in Appendix F which the offset is 0x28 which all I need is to add this to timer5 base address so 0x48046000 + 0x28 which will give me 0x48046028. Then I switch LDR R1, =0x48040028 to LDR R1, =0x48046028, after I everything address of timer2 change to timer5 and all the GPIO address is correct. I double check to make sure that everything is correct by going through all the address again, and double check where I found the offset of each and everyone of them. Once I confident that everything is correct, I start copy the code into Code Composer. When I finish typing all the code into Code Composer, I build the code and there was couple error that pop up which they all say unreferenced to TOFF. I check my code again and see that I have miss spell TOFF, so I went and fix it then rebuild the code again. There was still an error which is unreferenced to PASS\_ON, and then I look through my code again and I notice that I have miss some part of the code in the book when I copy to Code Composer, so I go back and add PASS\_ON to the code. Then rebuild the whole code and everything is being build correctly but since I want to make sure that nothing gone wrong like in Part2 of the project which was me misspelling stuff. So I went back to the code and then go through each line by line and compare it to the book until I reach the last line of the code in the book. When I am sure that nothing is wrong with the code, I just copy from the book to Code Composer, I then build the code again to make sure nothing is wrong and then I start the hooking up interrupt vector process by adding my INT\_DIRECTOR to the interrupt table. The circuit I build for Part3 is the same as Part2 since Part2 and Part3 is the same except that I have to use a timer instead of a loop to delay the light by 2 seconds.



I load and then run the code, I press the button, but nothing lights on, so I press the button couple more time but LED0 didn’t turn on. I then check all the code again to make sure nothing is miss spell again by going through each line by line again. But there was not anything I can see that is wrong, so I check where I load the address to turn on GPIO1\_21 again and then I see on the BUTTON\_SVC I mess up the address of GPIO1\_21 that I accidentally put 0x20000000 instead of 0x00200000, so I change that real quick and rebuild the program. Then I run the code again this time the light turns on, but it didn’t not turn off after 2 seconds have pass. Since light turn on but it did not turn off so I thought I must have done my timer initialize wrong, I went back and double check all my timer5 address and see if there anything I did wrong which there is not anything that I can see that is wrong. I then look through the register tab in the Code Composer tab, and then I run the program line by line this time to see if I mess up the register somehow. I run the program line by line as and when it get to the timer initialize I find where the register for TIMER5 in the tab and then start looking through all the register as I run the code. All the register is being update like it should be in the tab which mean that the address that I have is correct, so I set a hardware break point in INT\_DIRECTOR to make sure that everything is running correctly. I run the program and press the button, then step through each line of the code from INT\_DIRECTOR and then I notice that my code go to BUTTON\_SVC turn on the light and then go straight back to the infinite loops again. Which make me think there must be something wrong with my initialize of timer5 again.

Version 1.0:

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

ldr R13,=STACK1

**add** R13, R13,#0x1000

cps #0x12

ldr R13,=STACK2

**add** R13,R13,#0x1000

cps #0x13

@turn on GPIO1 CLK

ldr R0,=0x02

ldr R1,=0x44E000AC

str R0,[R1]

@make sure GPIO1\_21 is off

ldr R0,=0x4804C000

**add** R4,R0,#0x190

**mov** R7,#0x00200000

str R7,[R4]

@make GPIO1\_21 as an output

**add** R1,R0,#0x0134

ldr R6,[R1]

ldr R7,=0xFFDFFFFF

**and** R6,R7,R6

str R6,[R1]

@detect falling edge on GPIO1\_30 on pin 21

**add** R1,R0,#0x14C

**mov** R2,#0x40000000

ldr R3,[R1]

orr R3,R3,R2

str R3,[R1]

**add** R1,R0,#0x34

str R2,[R1]

@initialize INTC

ldr R1,=0x48200000

**mov** R2,#0x2

str R2,[R1,#0x10]

**mov** R2,#0x20000000 @ <-------------

str R2,[R1,#0xC8]

**mov** R2,#0x04

str R2,[R1,#0xE8]

@turn on Timer5 CLK

**mov** R2,#0x2

ldr R1,=0x44E000EC @ <----------------

str R2,[R1]

ldr R1,=0x44E00518 @ <----------------

str R2,[R1]

@initialize timer 2 registers, with count, overflow interrupt generation

ldr R1,=0x48046000 @ <----------------

**mov** R2,#0x1

str R2,[R1,#0x10]

**mov** R2,#0x2

str R2,[R1,#0x2C]

ldr R2,=0xFFFF0000

str R2,[R1,#0x40]

str R2,[R1,#0x3C]

@make sure processor IRQ enabled in CPSR

MRS R3, CPSR

BIC R3,#0x80

MSR CPSR\_c, R3

**LOOP:** **NOP**

**B** LOOP

**INT\_DIRECTOR:**

STMFD SP!,{R0-R3,LR}

ldr R1,=0x482000F8

ldr R2,[R1]

TST R2,#0x00000004

BEQ TCHK

ldr R0,=0x4804C02C

ldr R1,[R0]

TST R1,#0x40000000 @<------------

bne BUTTON\_SVC

ldr R0,=0x48200048

**mov** R1,#01

str R1,[R0]

LDMFD SP!,{R0-R3,LR}

subs PC,LR,#4

**TCHK:**

ldr R1,=0x482000D8

ldr R0,[R1]

TST R0,#0x20000000 @<------------

BEQ PASS\_ON

ldr R1,=0x48046028 @ <-----------

ldr R0,[R1]

TST R0,#0x2

BNE LED

**PASS\_ON:**

ldr R0,=0x48200048

**mov** R1,#01

str R1,[R0]

LDMFD SP!,{R0-R3,LR}

subs PC,LR,#4

LDMFD SP!,{R0-R3,LR}

subs PC,LR,#4

**BUTTON\_SVC:**

**mov** R1,#0x40000000

str R1,[R0]

@turn on LED0 GPIO1\_21

ldr R0,=0x4804C194

**mov** R1,#0x00200000

str R1,[R0]

ldr R1,=0x48046000

**mov** R2,#0x01 @<---------- load it before it hit this

ldr R1,=0x48046038

str R2,[R1]

@turn off NEWIRQA bit in INTC\_CONTROL, so can respond to new IRQ

ldr R0,=0x48200048

**mov** R1,#01

str R1,[R0]

LDMFD SP!,{R0-R3,LR}

subs PC,LR,#4

**LED:**

@turn off timer 5 interrupt request and enable INTC for next IRQ

ldr R1,=0x48046028 @ <--------------

**mov** R2,#0x2

str R2,[R1]

@toggle LED

ldr R1,=0x4804C000

ldr R2,[R1,#0x013C]

TST R2,#0x1000

**mov** R2,#0x1000

BNE TOFF

str R2,[R1,#0x194]

**B** BACK

**TOFF:**

ldr R2,=0x00200000

str R2,[R1,#0x190]

**BACK:**

ldr R1,=0x48200048

**mov** R2,#0x01

str R2,[R1]

ldmfd SP!,{R0-R3,LR}

subs PC,LR,#4

**.data**

**.align** 2

**STACK1:** .rept 1024

**.word** 0x0000

.endr

**STACK2:** .rept 1024

**.word** 0x0000

.endr

.END

12/8/2020

I load the program again and then run it again to see if there anything differences from 3 days ago. I press the button and the light turn on but after 2 seconds have pass the light remain on. So I then start looking through the code again and double check the place that I have change the address to make sure that everything is correct and then I start thinking why the light not turning off. The only thing I can think of is that I must have do my timer wrong, but I don’t where did I go wrong on. After rerun, the program couple time and the result are still the same, I conclude that I should ask for the TA to see if he knows what is wrong.

12/9/2020

After talking with the TA I found out there are couple place, I didn’t not fix yet which he point me to the right direction. One of the places I got thing wrong is on the LED which I was testing the wrong bit and that I forgot to change that since it was testing bit 12 but I do not want it to test bit 12. What I want to test was GPIO1\_21 to see if it on or not, which the whole time the program been testing GPIO1\_12 which make it my light do not turn off. So I go and did a quick fix by changing 2 lines which is TST R2, #0x1000 and MOV R2, #0x1000 to TST R2, #0x00200000 and MOV R2, #0x00200000. After that I have to change the count value of timer5 from 0xFFFFC000 which only going to wait for 0.5 seconds to 2 second which he point me to page 4447 that there an equation to calculate it. I go to the page and start reading from there but then I see a table which in the table list out the value is need for 1, 2, 3 seconds so I look at the table to see what the value for 2 seconds is which turn out to be 0xFFFF0000. So I went and change the part where I initialize timer5 and change LDR R2, =0xFFFFC000 to LDR R2, =0xFFFF0000 which is from 0.5 seconds to 2 seconds wait time. Then I need to do is to copy 3 lines of code from initialize which is.

ldr R2,=0xFFFF0000

str R2,[R1,#0x40]

str R2,[R1,#0x3C]

The reason is these 3 lines initialize the counter to 2 seconds and load it TLDR timer5 register and I need to write it to the TCRR register to start the wait 2 seconds timer. I just have to put it after I have loaded the value of GPIO1\_21 to SETDATAOUT which is when the light is turn on. The last thing I need to do is to switch the value that making the light blinking, which is MOV R2, #0x03 in the BUTTON\_SVC to MOV R2, #0x01 since 0x03 make the timer reload and start the clock again which make the light start blinking every 2 seconds and I don’t want that to happen, so I have to put #0x01 instead which only going to make it wait for 2 seconds only. After I have made all the change, I rebuild the circuit and plug everything back in again then I rebuild the program and then run the test again. I press the button and the light turn on and then it turns off after 2 seconds this time. I repeat pressing the button 5 more times to make sure that everything is running correctly and this time all 5 tests is pass.

Version 1.1:

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

ldr R13,=STACK1 @point to base of STACK for SVC mode

**add** R13, R13,#0x1000 @point to top of STACK

cps #0x12 @switch to IRQ mode

ldr R13,=STACK2 @point to IRQ STACK

**add** R13,R13,#0x1000 @point to top of STACK

cps #0x13 @ switch back to SVC mode

@turn on GPIO1 CLK

ldr R0,=0x02 @value to enable the clock for an GPIO module

ldr R1,=0x44E000AC @ address of CM\_PER\_GPIO1\_CLKCTRL register

str R0,[R1] @write 0x02 to register

@make sure GPIO1\_21 is off

ldr R0,=0x4804C000 @base address of GPIO1 register

**add** R4,R0,#0x190 @off set of CLEARDATAOUT, add the base address and the offset together

**mov** R7,#0x00200000 @load the value that would turn off GPIO1\_21 off to make sure that the LED0 isn't on

str R7,[R4] @ write the value to CLEARDATAOUT

@make GPIO1\_21 as an output

**add** R1,R0,#0x0134 @create GPIO1\_OE address

ldr R6,[R1] @read current GPIO1 output enable register

ldr R7,=0xFFDFFFFF @enable GPIO1\_21 as output

**and** R6,R7,R6 @modify bit 21

str R6,[R1] @write to GPIO1 output enable register

@detect falling edge on GPIO1\_30 on pin 21

**add** R1,R0,#0x14C @GPIO\_FALLINGDETECT register

**mov** R2,#0x40000000 @Load value for bit 30

ldr R3,[R1] @ read GPIO1\_FALLINGDETECT register

orr R3,R3,R2 @modify bit 30

str R3,[R1] @write back

**add** R1,R0,#0x34 @address of GPIO1\_IRQSTATUS\_SET\_0 register

str R2,[R1] @enable GPIO1\_30 request on POINTRPEND1

@initialize INTC

ldr R1,=0x48200000 @base address for INTC

**mov** R2,#0x2 @value to reset INTC

str R2,[R1,#0x10] @write to INTC Config register

**mov** R2,#0x20000000 @unmask INTC INT 93, Timer5 interrupt

str R2,[R1,#0xC8] @write to INTC\_MIR\_CLEAR2

**mov** R2,#0x04 @value to unmask INTC INT 98, GPIOINTA

str R2,[R1,#0xE8] @write to INTC\_MIR\_CLEAR3 register

@turn on Timer5 CLK

**mov** R2,#0x2 @value to enable Timer5 CLK

ldr R1,=0x44E000EC @address of CM\_PER\_TIMER5\_CLKCTRL

str R2,[R1] @turn on

ldr R1,=0x44E00518 @address of PRCMCLKSEL\_TIMER5 register

str R2,[R1] @select 32Khz CLK for Timer5

@initialize timer 2 registers, with count, overflow interrupt generation

ldr R1,=0x48046000 @Base address for Timer5 register

**mov** R2,#0x1 @value to reset Timer5

str R2,[R1,#0x10] @write to Timer5 CFG register

**mov** R2,#0x2 @value to enable overflow interrupt

str R2,[R1,#0x2C] @write to Timer5 IRQENABLE\_SET

ldr R2,=0xFFFF0000 @count value for 2 seconds

str R2,[R1,#0x40] @Timer5 TLDR load register

str R2,[R1,#0x3C] @write to Timer5 TCRR count register

@make sure processor IRQ enabled in CPSR

MRS R3, CPSR @copy CPSR to R3

BIC R3,#0x80 @clear bit 7

MSR CPSR\_c, R3 @write back to CPSR

@wait for interrupt

**LOOP:** **NOP**

**B** LOOP

**INT\_DIRECTOR:**

STMFD SP!,{R0-R3,LR} @push registers on stack

ldr R1,=0x482000F8 @address of INTC\_PENDING\_IRQ3 register

ldr R2,[R1] @read INTC\_PENDING\_IRQ3 register

TST R2,#0x00000004 @test bit 2

BEQ TCHK @not GPIOINT1A, check if Timer5, else

ldr R0,=0x4804C02C @GPIO\_IRQSTATUS\_0 register address

ldr R1,[R0] @read STATUS register to see if button

TST R1,#0x40000000 @check if bit 30 = 1

bne BUTTON\_SVC @ if bit 30 = 1, button push, service

ldr R0,=0x48200048 @else, go back. INTC\_CONTROL register

**mov** R1,#01 @value to clear bit 0

str R1,[R0] @write to INTC\_CONTROL register

LDMFD SP!,{R0-R3,LR} @restore register

subs PC,LR,#4 @pass execution to wait LOOP for now

**TCHK:**

ldr R1,=0x482000D8 @address of INTC PENDING\_1IRQ2 register

ldr R0,[R1] @read value

TST R0,#0x20000000 @check if interrupt from Timer5

BEQ PASS\_ON @no, return yes, check for overflow

ldr R1,=0x48046028 @address of Timer5 IRQSTATUS register

ldr R0,[R1] @read value

TST R0,#0x2 @check bit 1

BNE LED @if overflow, then go toggle LED

**PASS\_ON:** @else go back to wait loop

ldr R0,=0x48200048 @address of INTC\_CONTROL register

**mov** R1,#01 @value to clear bit 0

str R1,[R0] @write to INT\_CONTROL register

LDMFD SP!,{R0-R3,LR} @restore register

subs PC,LR,#4 @pass execution to wait LOOP for now

LDMFD SP!,{R0-R3,LR} @restore register

subs PC,LR,#4 @pass execution to wait LOOP for now

**BUTTON\_SVC:**

**mov** R1,#0x40000000 @value to turn off GPIO1\_30 IRQ request. This will turn off INTC IRQ request also

str R1,[R0] @write to GPIO1\_IRQSTATUS\_0 register

@turn on LED0 GPIO1\_21

ldr R0,=0x4804C194 @load address of GPIO1\_SETDATAOUT register

**mov** R1,#0x00200000 @load value to turn on GPIO1\_21

str R1,[R0] @write value to GPIO1\_SETDATAOUT register

ldr R1,=0x48046000 @address of Timer5 TCLR register

ldr R2,=0xFFFF0000 @value load for 2 seconds

str R2,[R1,#0x40] @Timer4 TLDR load register

str R2,[R1,#0x3C] @write to Timer5 TCRR count register

**mov** R2,#0x01 @value to make timer wait for 2 seconds

ldr R1,=0x48046038 @address of Timer5 TCLR register

str R2,[R1] @write to TCLR register

@turn off NEWIRQA bit in INTC\_CONTROL, so can respond to new IRQ

ldr R0,=0x48200048 @address of INTC\_CONTROL register

**mov** R1,#01 @value to clear bit 0

str R1,[R0] @write to INTC\_CONTROL register

LDMFD SP!,{R0-R3,LR} @restore register

subs PC,LR,#4 @pass execution on to wait LOOP for now

**LED:**

@turn off timer 5 interrupt request and enable INTC for next IRQ

ldr R1,=0x48046028 @load address of Timer5 IRQSTATUS register

**mov** R2,#0x2 @value to reset Timer5 Overflow IRQ request

str R2,[R1] @write

@toggle LED

ldr R1,=0x4804C000 @base address of GPIO1

ldr R2,[R1,#0x013C] @read value from GPIO\_DATAOUT

TST R2,#0x00200000 @check bit 21 where LED is connect

**mov** R2,#0x00200000 @value to set or clear bit 21

BNE TOFF @LED on, go turn off

str R2,[R1,#0x194] @LED off, turn on with GPTIO1\_SETDATAOUT

**B** BACK @back to wait LOOP

**TOFF:**

ldr R2,=0x00200000 @value to turn off GPIO1\_21

str R2,[R1,#0x190] @turn LED off with GPIO1\_CLEARDATAOUT

**BACK:**

ldr R1,=0x48200048 @address of INTC\_CONTROL register

**mov** R2,#0x01 @value to enable new IRQ response in INTC

str R2,[R1] @write

ldmfd SP!,{R0-R3,LR} @restore register

subs PC,LR,#4 @return from IRQ interrupt procedure

**.data**

**.align** 2

**STACK1:** .rept 1024

**.word** 0x0000

.endr

**STACK2:** .rept 1024

**.word** 0x0000

.endr

.END

**ALGORITHM**

Part 1 High level Algorithm:

MAINLINE

1. Set up to enable the clock for CM\_PER\_GPIO module
2. Load GPIO1\_21 to OE register
3. Enable GPIO1\_21 as output and load it to SETDATAOUT
4. Wait for 2 seconds
5. Turn off GPIO1\_21 by send the value to CLEARDATAOUT
6. Load GPIO1\_22 to OE register
7. Enable GPIO1\_22 as output and load it to SETDATAOUT
8. Wait for 2 seconds
9. Turn off GPIO1\_21 by send the value to CLEARDATAOUT
10. Load GPIO1\_23 to OE register
11. Enable GPIO1\_23 as output and load it to SETDATAOUT
12. Wait for 2 seconds
13. Turn off GPIO1\_23 by send the value to CLEARDATAOUT
14. Load GPIO1\_24 to OE register
15. Enable GPIO1\_24 as output and load it to SETDATAOUT
16. Wait for 2 seconds
17. Turn off GPIO1\_24 by send the value to CLEARDATAOUT
18. Then cycle the whole process again this time from GPIO1\_24 and go up to GPIO1\_21 instead of GPIO1\_21 to GPIO1\_24.

Part 1 Low level Algorithm:

1. Set up to enable the clock for CM\_PER\_GPIO module

* Load R0 with #0x02 for enable value
* Load 0x02 to 0x44E000AC

1. Load GPIO1\_21 to OE register

* Load 0xFFDFFFFF to 0x4804C134

1. Enable GPIO1\_21 as output and load it to SETDATAOUT

* Load 0x00200000 to 0x4804C194

1. Wait for 2 seconds

* Use a for loops and load in value 0x00400000 for 2 second to R7. Then subs #1 to that value until it hit 0, and end the loop once the value is 0.

1. Turn off GPIO1\_21 by send the value to CLEARDATAOUT

* Load the value 0x00200000 to 0x4804C190 which is CLEARDATAOUT register to turn off GPIO1\_21

1. Load GPIO1\_22 to OE register

* Load 0xFFBFFFFF to 0x4804C134

1. Enable GPIO1\_22 as output and load it to SETDATAOUT

* Load 0x00400000 to 0x4804C194

1. Wait for 2 seconds

* Use a for loops and load in value 0x00400000 for 2 second to R7. Then subs #1 to that value until it hit 0 and end the loop once the value is 0.

1. Turn off GPIO1\_22 by send the value to CLEARDATAOUT

* Load the value 0x00400000 to 0x4804C190 which is CLEARDATAOUT register to turn off GPIO1\_22

1. Load GPIO1\_23 to OE register

* Load 0xFF7FFFFF to 0x4804C134

1. Enable GPIO1\_23 as output and load it to SETDATAOUT

* Load 0x00800000 to 0x4804C194

1. Wait for 2 seconds

* Use a for loops and load in value 0x00400000 for 2 second to R7. Then subs #1 to that value until it hit 0 and end the loop once the value is 0.

1. Turn off GPIO1\_23 by send the value to CLEARDATAOUT

* Load the value 0x00800000 to 0x4804C190 which is CLEARDATAOUT register to turn off GPIO1\_23

1. Load GPIO1\_24 to OE register

* Load 0xFEFFFFFF to 0x4804C134

1. Enable GPIO1\_24 as output and load it to SETDATAOUT

* Load 0x01000000 to 0x4804C194

1. Wait for 2 seconds

* Use a for loops and load in value 0x00400000 for 2 second to R7. Then subs #1 to that value until it hit 0, and end the loop once the value is 0.

1. Turn off GPIO1\_24 by send the value to CLEARDATAOUT

* Load the value 0x01000000 to 0x4804C190 which is CLEARDATAOUT register to turn off GPIO1\_24

1. Cycle through the whole process again from GPIO1\_24 to GPIO1\_21 this time.

Part 2 High level Algorithm:

MAINLINE

1. Set up stack for Supervisor mode and Interrupt mode
2. Turn on GPIO CLK
3. Turn off GPIO1\_21 to make sure that it is off when the program first run
4. Make GPIO1\_21 as an output
5. Set up GPIO1\_30 as detect falling edge
6. Initialize INTC
7. Enable IRQ in CPSR
8. Loop until there is an interrupt happen

INT\_DIRECTOR

1. Save the register
2. Check for if the button is being press or not
3. If the button is not press, then go back to wait loop go to PASS\_ON
4. Check for bit 21 to see if it 1 or 0.
5. If it 1 then button is press then go to BUTTON\_SVC
6. Else go back to wait loops go PASS\_ON

PASS\_ON

1. Restore the register
2. Pass execution for now to wait LOOP.

BUTTON\_SVC

1. Turn off GPIO1\_30 interrupt request
2. Turn off NEWIRQ bit in INTC\_CONTROL
3. Turn on LED on GPIO1\_21
4. Wait for 2 seconds
5. Turn off LED on GPIO1\_21
6. Go back to wait loops

Part 2 Low level Algorithm:

MAINLINE

1. Set up stack for Supervisor mode and Interrupt mode
2. Turn on GPIO CLK

* Load 0x02 into 0x44E000AC

1. Turn off GPIO1\_21 to make sure that it is off when the program first run

* Load 0x00200000 into 0x4804C190 which is CLEARDATAOUT register

1. Make GPIO1\_21 as an output

* Load 0xFFDFFFFF into 0x4804C134 which is the OE register

1. Set up GPIO1\_30 as detect falling edge

* Load in the address of GPIO1\_FALLINGDETECT which is 4804C14C
* Load the value 0x40000000 which is bit 30
* Read from FALLINGDETECT register
* Sent 0x400000000 to 4804C034

1. Initialize INTC

* Load in address 0x482000E8 which is the address of INTC\_MIR\_CLEAR3
* Write 0x04 into the address of INTC\_MIR\_CLEAR3

1. Enable IRQ in CPSR

* Clear bit 7 by reading 0x80 into CPSR then write back

1. Loop until there is an interrupt happen

INT\_DIRECTOR

1. Save the register
2. Check for if the button is being press or not

* Test bit 2 0x00000004

1. If the button is not press, then go back to wait loop go to PASS\_ON
2. Check for bit 21 to see if it 1 or 0.

* Test bit 21 which is 0x40000000 to see if it 1 or 0

1. If it 1 then button is press then go to BUTTON\_SVC
2. Else go back to wait loops go PASS\_ON

PASS\_ON

1. Restore the register

* From R0-R3 on LR

1. Pass execution for now to wait LOOP.

* Subs #4 from LR

BUTTON\_SVC

1. Turn off GPIO1\_30 interrupt request

* Write 0x40000000 to GPIO1\_IRQSTATUS\_0

1. Turn off NEWIRQ bit in INTC\_CONTROL

* Load in 0x48200048
* Clear bit 0 by #01
* Write #01 to address 0x48200048

1. Turn on LED on GPIO1\_21

* Load 0x00200000 to 0x4804C194

1. Wait for 2 seconds
2. Turn off LED on GPIO1\_21

* Load 0x00200000 to 0x4804C190

1. Go back to wait loops

* LDMFD SP!, {R0-R3,LR}
* Subs PC, LR , #4

Part 3 High level Algorithm:

MAINLINE

1. Set up stack for Supervisor mode and IRQ mode
2. Turn on GPIO1 clock
3. Turn off GPIO1\_21 if it is on
4. Make GPIO1\_21 as output
5. Set up Falling Edge detect on GPIO1\_30
6. Initialize INTC, Timer5 on INTC #93, GPIO1\_30 on INTC #98
7. Turn on Timer5 clock
8. Initialize timer5 2 register, with count, overflow interrupt generation
9. Loop till interrupt

INT\_DIRECTOR

1. Save register
2. Check to see if there an interrupt from button
3. If yes, then go to BUTTON\_SVC, else go back to wait loop
4. Check for interrupt from timer 5
5. If there an interrupt from timer 5 then check for overflow else go back to wait loops

BUTTON\_SVC

1. Turn off GPIO1\_30 IRQ request
2. Turn on GPIO1\_21
3. Start timer 5 to start counting to 2 seconds
4. Turn off NEWIRQA bit in INTC\_CONTROL
5. Restore register and go back to wait loop

TCHK

1. Check if there an interrupt from timer 5
2. If no, then go back to wait loops else check for overflow
3. If there overflow, then go to LED
4. Else go back to wait loop
5. Restore register and go back to wait loop

LED

1. Turn off timer 5 interrupt request
2. Check for if GPIO1\_21 is on or not
3. If not on, then turn it on
4. Else turn it off

TOFF

1. Turn off GPIO1\_21
2. Restore register
3. Go back to wait loops

Part 3 Low level Algorithm:

MAINLINE

1. Set up stack for Supervisor mode and IRQ mode
2. Turn on GPIO1 clock

* Load 0x02 to 0x44E000AC

1. Turn off GPIO1\_21 if it is on

* Load 0x00200000 to 0x4804C190

1. Make GPIO1\_21 as output

* Load 0xFFDFFFFF to 48040C134

1. Set up Falling Edge detect on GPIO1\_30

* Move 0x40000000 to 4804C14C

1. Initialize INTC, Timer5 on INTC #93, GPIO1\_30 on INTC #98

* Load in base address for INT 0x48200000
* Move 0x2 to the INTC config which is 0x48200010
* Move 0x40000000 value of 93 to 0x482000C8 INTC\_MIR\_CLEAR2
* Move 0x04 to 0x482000E8 INTC\_MIR\_CLEAR3

1. Turn on Timer5 clock

* Load the value to enable timer5 0x2
* Load the value to address of CM\_PER\_TIMER5\_CLKCTRL which is 0x44E000EC
* Load 0x2 into 0x44E00518 which is the address of PRCMCLKSEL\_TIMER5

1. Initialize timer5 2 register, with count, overflow interrupt generation

* Load base address of Timer5 register 0x48046000
* Write the reset value 0x1 to 0x48046010 which is CFG register
* Write 0x2 to 0x4804602C to enable overflow
* Load in value for 2 second 0xFFFF0000
* Store the value for 2 seconds to 0x48046040 for TLDR load register
* And store it to 0x4804603C for TCRR count register

1. Loop till interrupt

INT\_DIRECTOR

1. Save register
2. Check to see if there an interrupt from button

* Test bit 30 which is 0x40000000 for if it 1 or 0
* If it 1 then button is press else it not press

1. If yes, then go to BUTTON\_SVC, else go back to wait loop
2. Check for interrupt from timer 5
3. If there an interrupt from timer 5 then check for overflow else go back to wait loops

BUTTON\_SVC

1. Turn off GPIO1\_30 IRQ request

* Load 0x40000000 to GPIO1\_IRQSTATUS\_0 register

1. Turn on GPIO1\_21

* Load 0x00200000 to 0x4804C194 to turn on the LED

1. Start timer 5 to start counting to 2 seconds

* Load in address of timer5 TCLR register 0x48046000
* Load in value for 2 second 0xFFFF0000
* Put the value in 0x48046040 for TLDR load register and 0x4804603C for TCRR register.

1. Turn off NEWIRQA bit in INTC\_CONTROL

* Move #01 to 0x48200048

1. Restore register and go back to wait loop

TCHK

1. Check if there an interrupt from timer 5

* Test bit 29 which is 0x20000000

1. If no, then go back to wait loops else check for overflow

* BEG PASS\_ON if no

1. If there overflow, then go to LED

* BNE LED if there an overflow

1. Else go back to wait loop
2. Restore register and go back to wait loop

LED

1. Turn off timer 5 interrupt request

* Load 0x02 to 0x48046028 which is TIMER5 IRQSTATUS register

1. Check for if GPIO1\_21 is on or not
2. If not on, then turn it on

* Load 0x00200000 to 0x4804C194

1. Else turn it off

* BNE TOFF

TOFF

1. Turn off GPIO1\_21

* Load 0x00200000 to 0x484C190 to turn off GPIO1\_21

1. Restore register
2. Go back to wait loops