Lab 9: Pulse Modulation

Victor Yuan

Microcontrollers and Embedded Systems

April 20, 2017

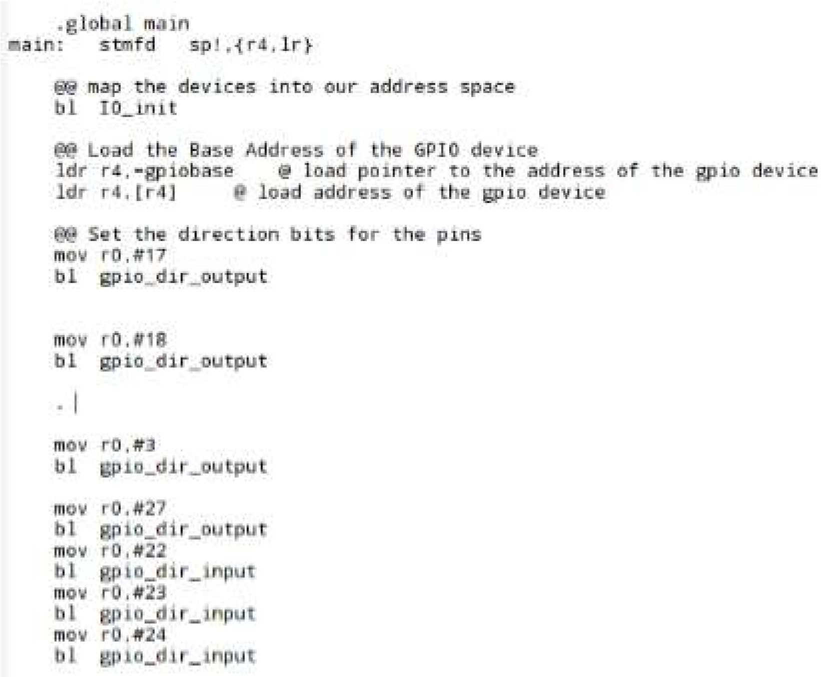
Introduction:

The goal of this lab was to configure the clock, PWM so that input from pinouts changes the brightness of a lightbulb .

Procedure:

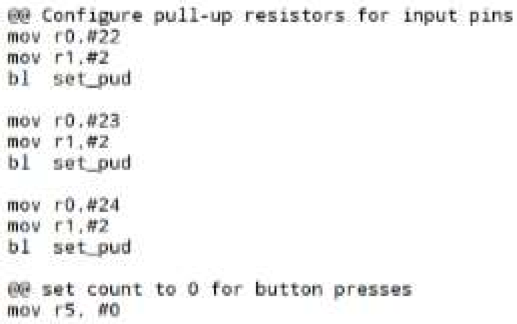
Download the base code from Canvas. Then configure GPIO 12 for alternate function 0. Then configure the PWM. Configure PWM0 from 0 to 100% duty cycle. Wire the pinouts to the breadboard, switches, and led.

Code:



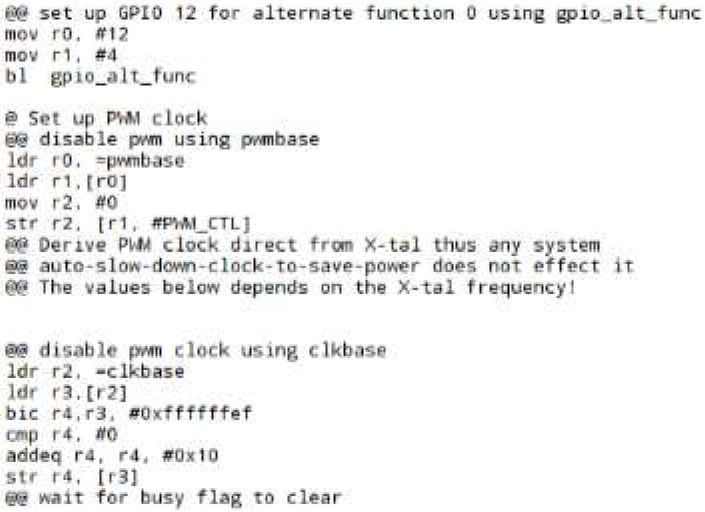
```
    .global main
main:   stmfd   sp!,{r4,lr}

    @@ map the devices into our address space
    bl  IO_init

    @@ Load the Base Address of the GPIO device
    ldr r4,=gpiobase    @ load pointer to the address of the gpio device
    ldr r4,[r4]     @ load address of the gpio device

    @@ Set the direction bits for the pins
    mov r0,#17
    bl  gpio_dir_output


    mov r0,#18
    bl  gpio_dir_output

    .

    mov r0,#3
    bl  gpio_dir_output

    mov r0,#27
    bl  gpio_dir_output
    mov r0,#22
    bl  gpio_dir_input
    mov r0,#23
    bl  gpio_dir_input
    mov r0,#24
    bl  gpio_dir_input
```

pins 22,23,24 set to input for the switches



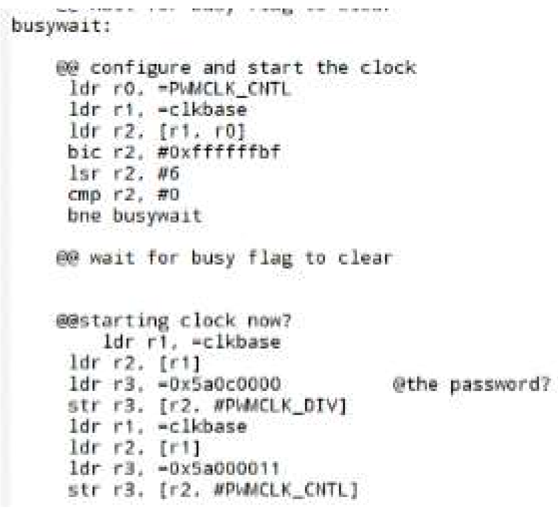
```
@@ Configure pull-up resistors for input pins
mov r0,#22
mov r1,#2
bl  set_pud

mov r0,#23
mov r1,#2
bl  set_pud

mov r0,#24
mov r1,#2
bl  set_pud

@@ set count to 0 for button presses
mov r5, #0
```

Given Base code



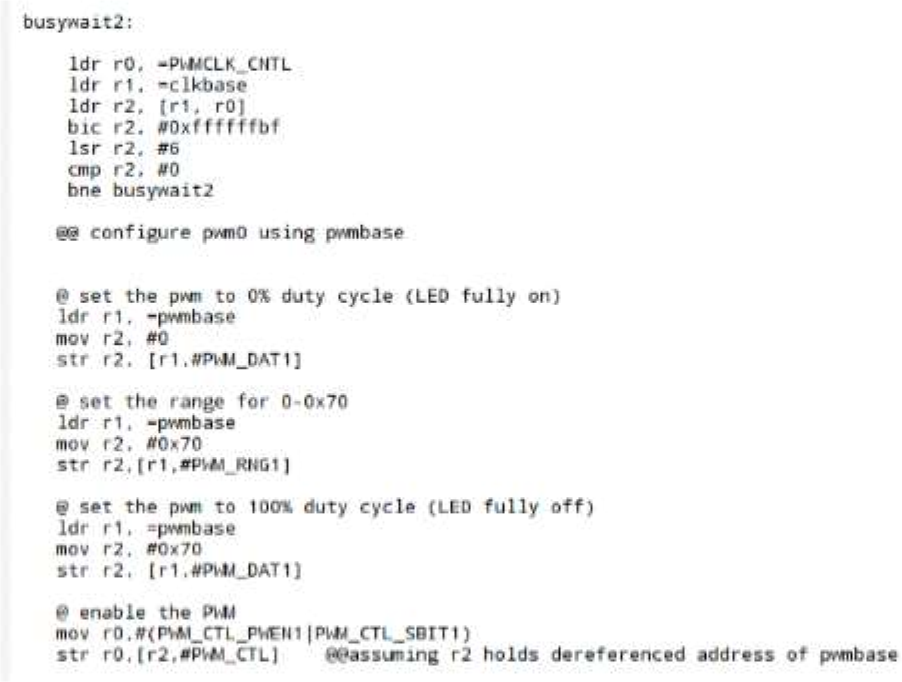
```
@@ set up GPIO 12 for alternate function 0 using gpio_alt_func
mov r0, #12
mov r1, #4
bl  gpio_alt_func

@ Set up PWM clock
@@ disable pwm using pwmbase
ldr r0, =pwmbase
ldr r1,[r0]
mov r2, #0
str r2, [r1, #PWM_CTL]
@@ Derive PWM clock direct from X-tal thus any system
@@ auto-slow-down-clock-to-save-power does not effect it
@@ The values below depends on the X-tal frequency!


@@ disable pwm clock using clkbase
ldr r2, =clkbase
ldr r3,[r2]
bic r4,r3, #0xffffffef
cmp r4, #0
addeq r4, r4, #0x10
str r4, [r3]
@@ wait for busy flag to clear
```

Setting up GPIO 12 for output and alternate function 0, setting up PWM



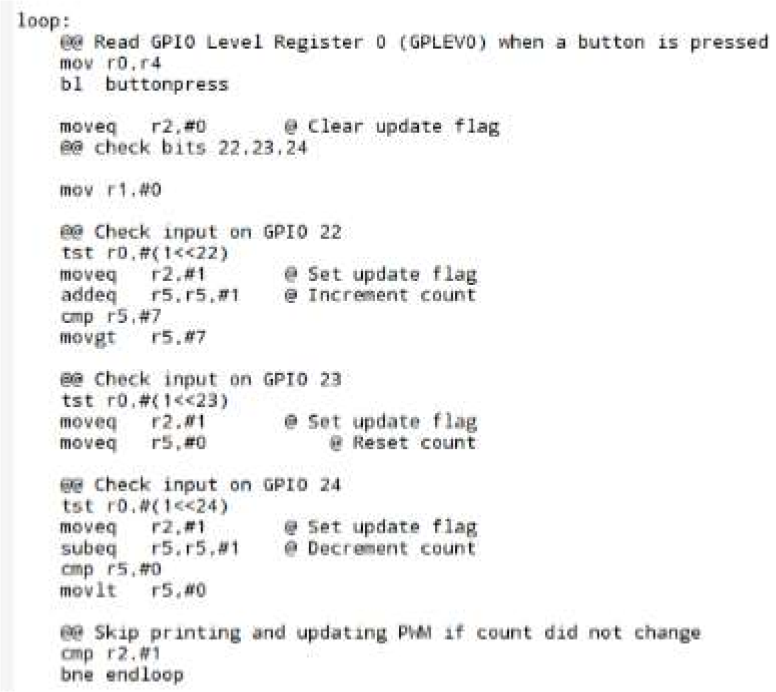
```
busywait:

    @@ configure and start the clock
     ldr r0, =PWMCLK_CNTL
     ldr r1, =clkbase
     ldr r2, [r1, r0]
     bic r2, #0xffffffbf
     lsr r2, #6
     cmp r2, #0
     bne busywait

    @@ wait for busy flag to clear


    @@starting clock now?
        ldr r1, =clkbase
     ldr r2, [r1]
     ldr r3, =0x5a0c0000          @the password?
     str r3, [r2, #PWMCLK_DIV]
     ldr r1, =clkbase
     ldr r2, [r1]
     ldr r3, =0x5a000011
     str r3, [r2, #PWMCLK_CNTL]
```

First Busy wait,



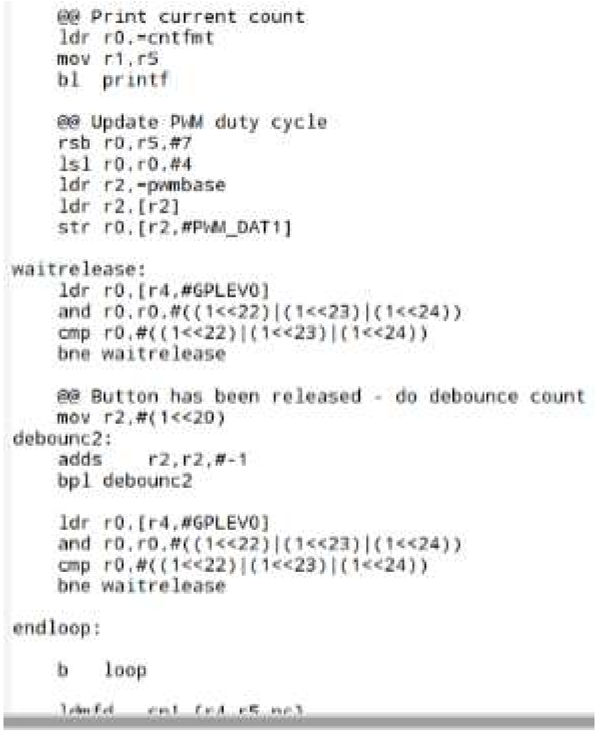
```
 busywait2:

     ldr r0, =PWMCLK_CNTL
     ldr r1, =clkbase
     ldr r2, [r1, r0]
     bic r2, #0xffffffbf
     lsr r2, #6
     cmp r2, #0
     bne busywait2

    @@ configure pwm0 using pwmbase


    @ set the pwm to 0% duty cycle (LED fully on)
    ldr r1, =pwmbase
    mov r2, #0
    str r2, [r1,#PWM_DAT1]

    @ set the range for 0-0x70
    ldr r1, =pwmbase
    mov r2, #0x70
    str r2,[r1,#PWM_RNG1]

    @ set the pwm to 100% duty cycle (LED fully off)
    ldr r1, =pwmbase
    mov r2, #0x70
    str r2, [r1,#PWM_DAT1]

    @ enable the PWM
    mov r0,#(PWM_CTL_PWEN1|PWM_CTL_SBIT1)
    str r0,[r2,#PWM_CTL]    @@assuming r2 holds dereferenced address of pwmbase
```

Second Busy wait



```
loop:
    @@ Read GPIO Level Register 0 (GPLEV0) when a button is pressed
    mov r0,r4
    bl  buttonpress

    moveq   r2,#0       @ Clear update flag
    @@ check bits 22,23,24

    mov r1,#0

    @@ Check input on GPIO 22
    tst r0,#(1<<22)
    moveq   r2,#1       @ Set update flag
    addeq   r5,r5,#1    @ Increment count
    cmp r5,#7
    movgt   r5,#7

    @@ Check input on GPIO 23
    tst r0,#(1<<23)
    moveq   r2,#1       @ Set update flag
    moveq   r5,#0           @ Reset count

    @@ Check input on GPIO 24
    tst r0,#(1<<24)
    moveq   r2,#1       @ Set update flag
    subeq   r5,r5,#1    @ Decrement count
    cmp r5,#0
    movlt   r5,#0

    @@ Skip printing and updating PWM if count did not change
    cmp r2,#1
    bne endloop
```

Setting up PWM0

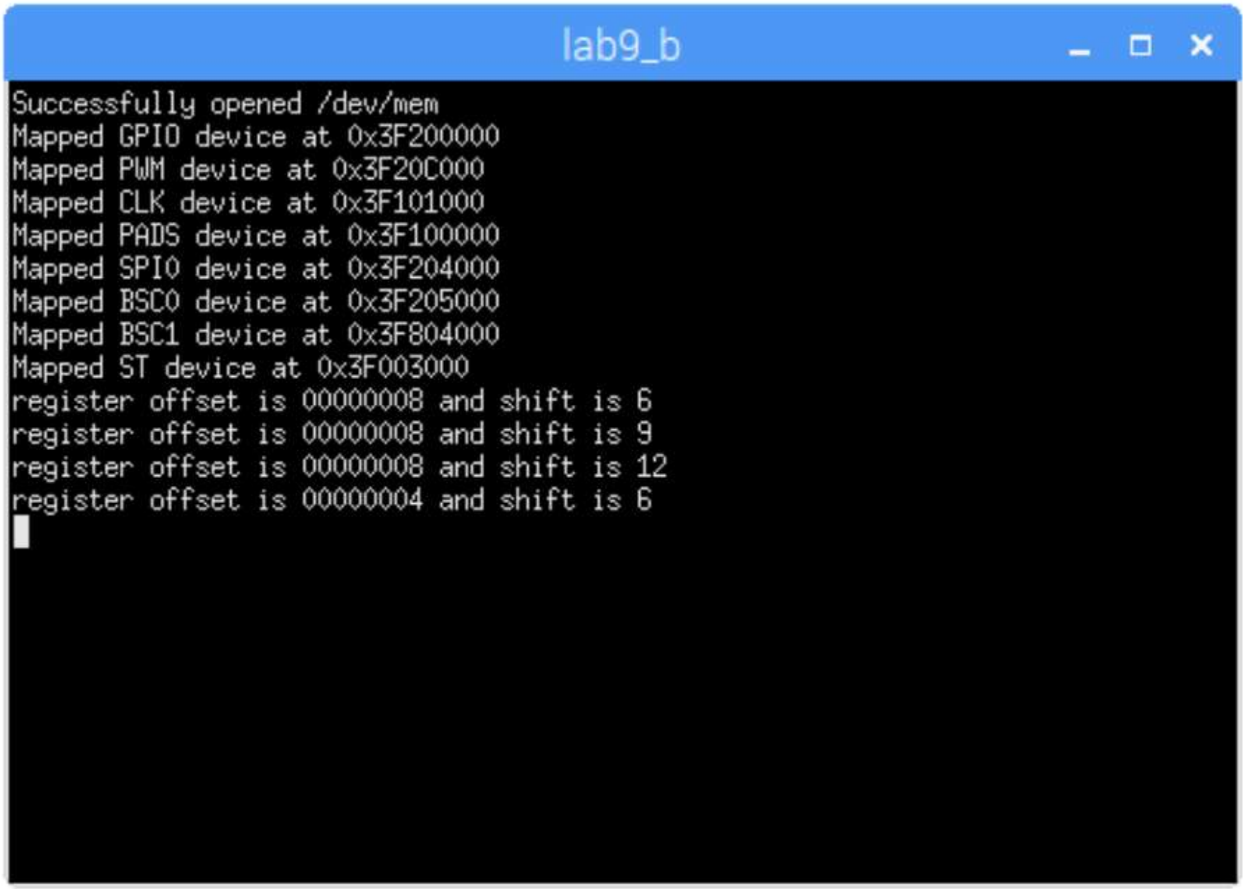


```
    @@ Print current count
    ldr r0,=cntfmt
    mov r1,r5
    bl  printf

    @@ Update PWM duty cycle
    rsb r0,r5,#7
    lsl r0,r0,#4
    ldr r2,=pwmbase
    ldr r2,[r2]
    str r0,[r2,#PWM_DAT1]

waitrelease:
    ldr r0,[r4,#GPLEV0]
    and r0,r0,#((1<<22)|(1<<23)|(1<<24))
    cmp r0,#((1<<22)|(1<<23)|(1<<24))
    bne waitrelease

    @@ Button has been released - do debounce count
    mov r2,#(1<<20)
debounc2:
    adds    r2,r2,#-1
    bpl debounc2

    ldr r0,[r4,#GPLEV0]
    and r0,r0,#((1<<22)|(1<<23)|(1<<24))
    cmp r0,#((1<<22)|(1<<23)|(1<<24))
    bne waitrelease

endloop:

    b   loop

    ldmfd   sp!,{r4,r5,pc}
```

Setting up the duty cycle 0x0 = 0%, up to 0x70 = 100%

Range from 0x0-0x70

Results:



```
Successfully opened /dev/mem
Mapped GPIO device at 0x3F200000
Mapped PWM device at 0x3F20C000
Mapped CLK device at 0x3F101000
Mapped PADS device at 0x3F100000
Mapped SPI0 device at 0x3F204000
Mapped BSC0 device at 0x3F205000
Mapped BSC1 device at 0x3F804000
Mapped ST device at 0x3F003000
register offset is 00000008 and shift is 6
register offset is 00000008 and shift is 9
register offset is 00000008 and shift is 12
register offset is 00000004 and shift is 6
```

Conclusion:

The code did not work in the end, this was due to an infinite loop in busy wait.