**EGRE 364**

**Lab #8 Report**

**Date:**

12/5/2019

**Team Members:**

Dakota Bernacki

Jonathan O’Dell

## Introduction

Lab 8 was designed to teach the following:

* The basics of stepper motors; control and wiring
* Assembly coding
* Assembly subprograms

## Design

The code associated with this project has been zipped and uploaded to Blackboard.

The file name is: Lab5\_DakotaBernacki\_JonathanO\_Dell.zip

Primary files:

* main.s
* startup\_stm32l476xx.s

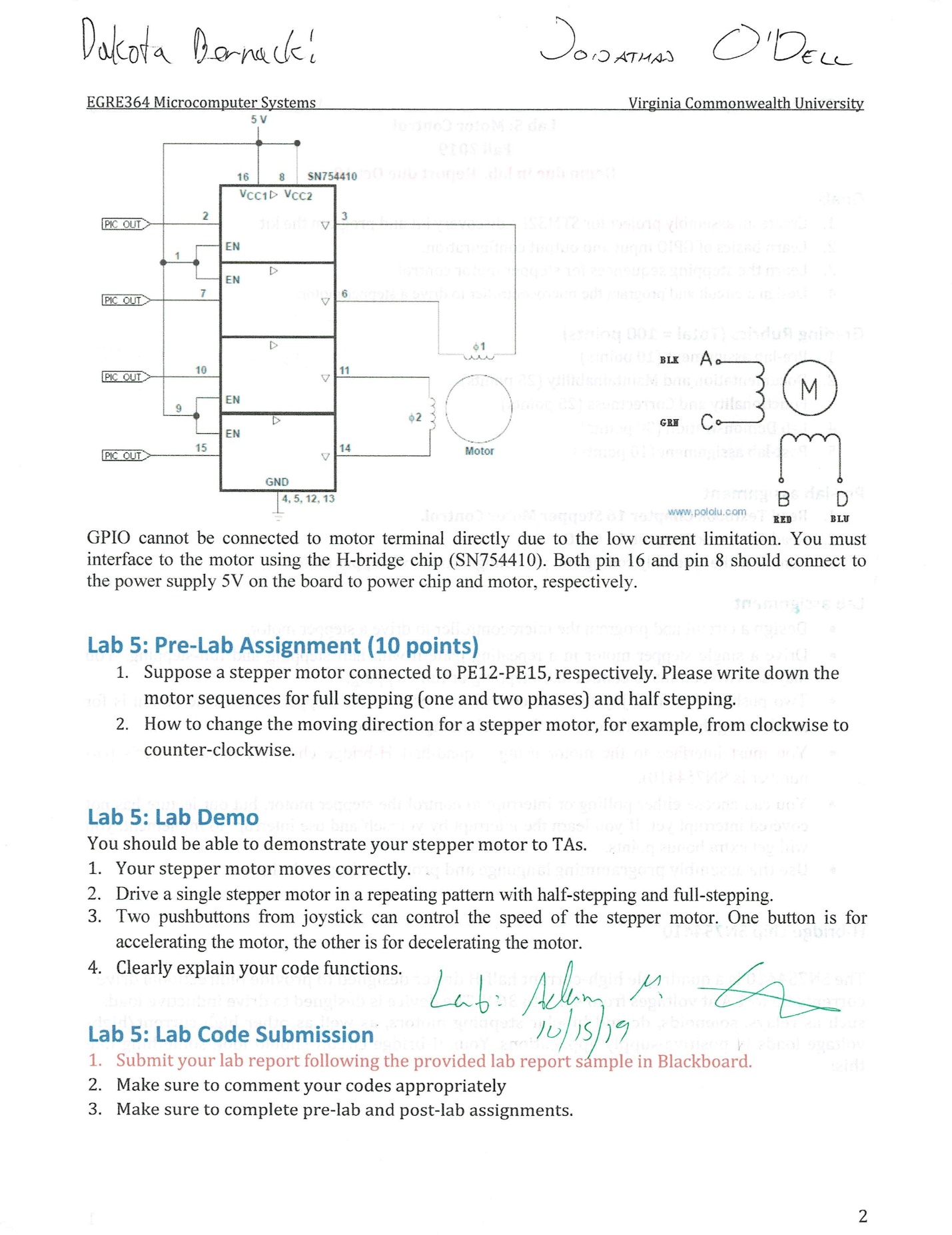
See **Appendix A** for code snippets,

## Functionality and Correctness

As specified; the motor rotates without interruption and Its speed can be increased and decreased via the up/down joystick buttons. Additionally, the direction of rotation can be reversed by clicking the center joystick button and the step type [one phase, two phase, half-step] can be cycled by using the left and right joystick buttons. All button inputs are handled via interrupts.

## Lab Demonstration

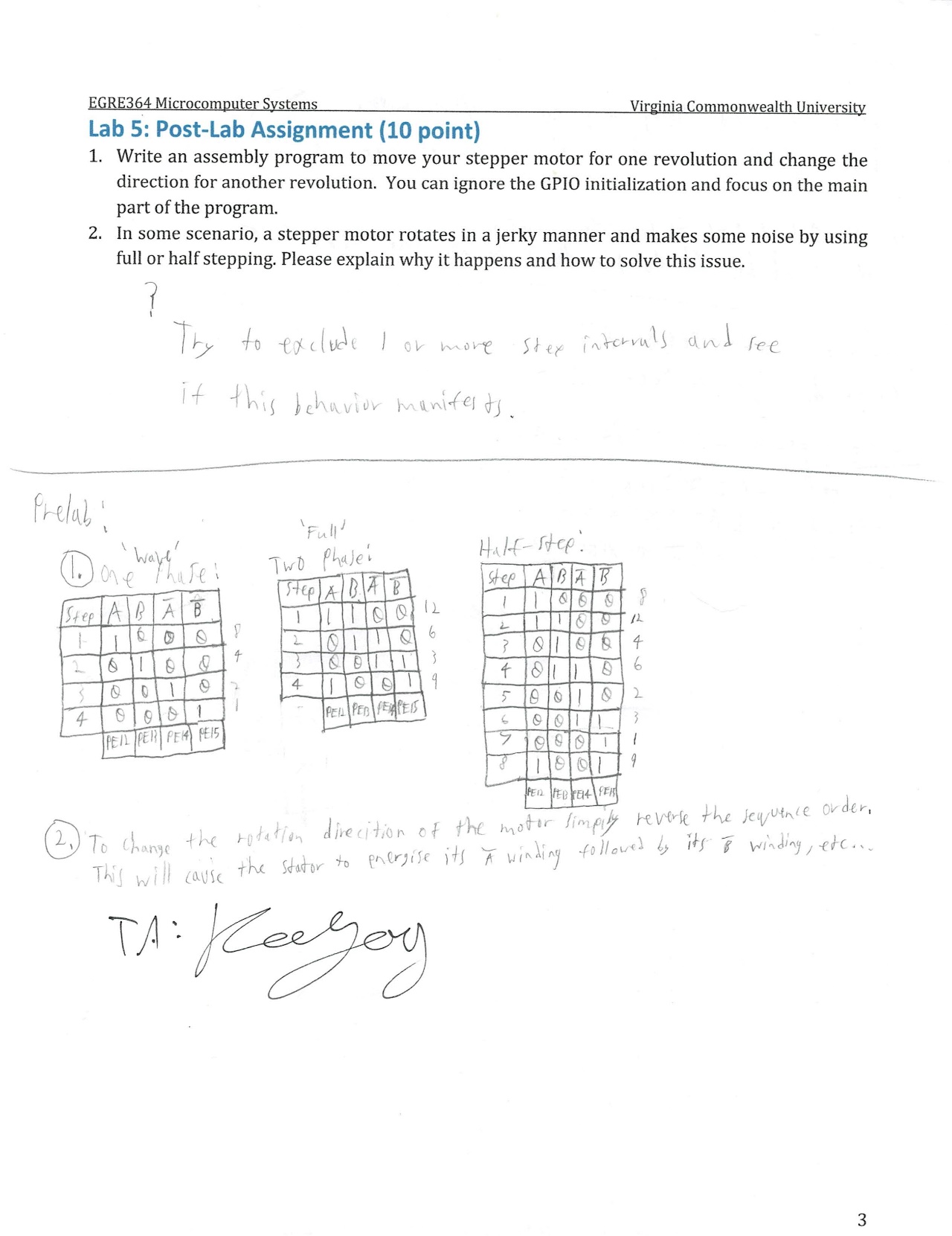
The aforementioned correct behavior was demonstrated to a TA, and he signed off on it at approximately 1530hrs 10/15/2019.



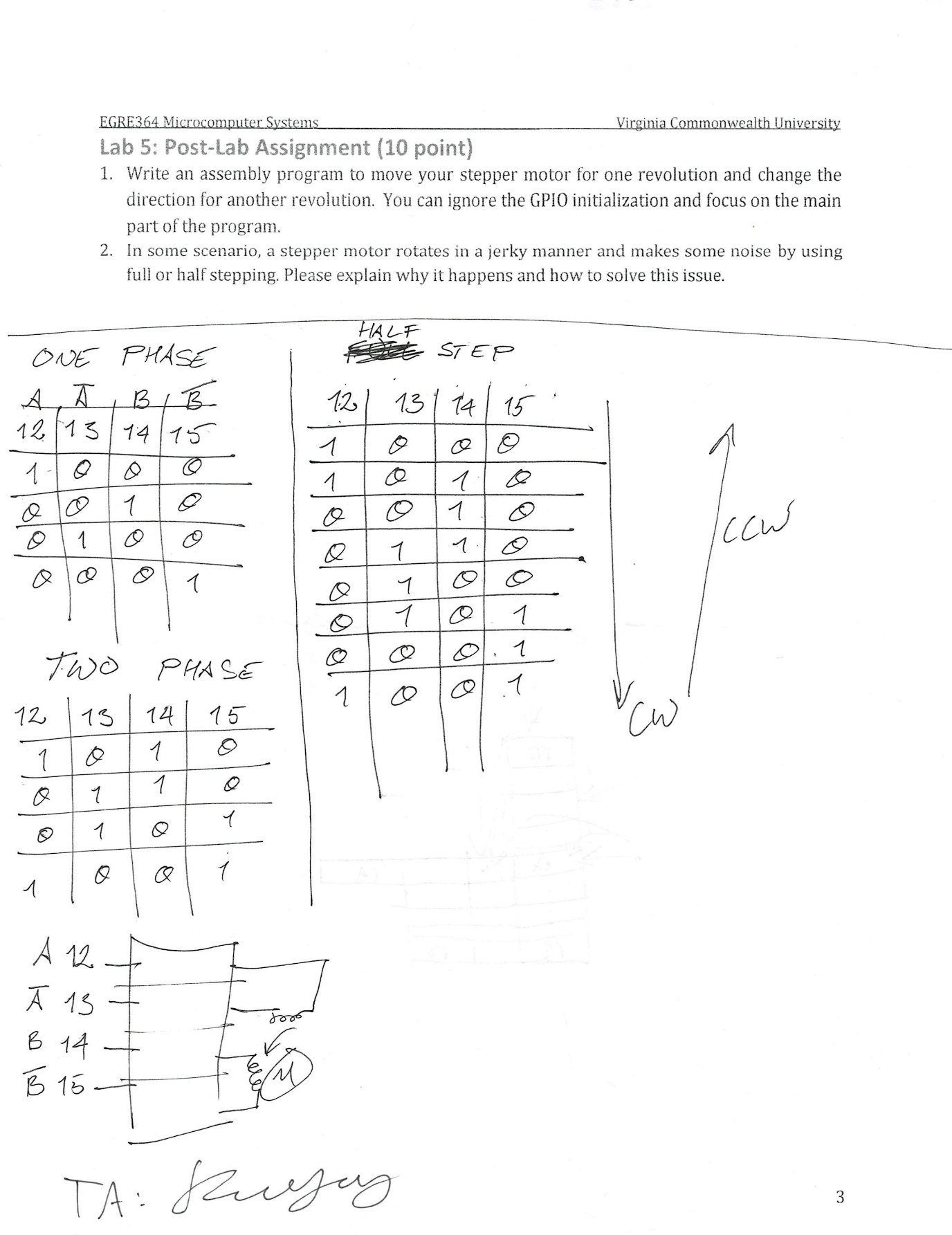
## Pre-lab assignment

Scanned pre-lab assignments:

Bernacki:



O’Dell:



## Post-lab assignment

1. Assembly to run motor for one revolution CW and then one revolution CCW:

align

\_\_main proc

export \_\_main

entry

;Register setup for motor control options

ldr r3, =step ;store base adr of our step vector

mov r4, #0 ;adr offset to sel setp vector index

bl motor ;Branch to the motor loop

s b s

endp

align

;Runs motor forever; parameters only changed via interrupt call

motor PROC

export motor

ldr r0, =GPIOE\_BASE

ldr r2, =200 ;number of steps per 360deg rotation;

bl loopA

ldr r2, =200

bl loopB

bl EXIT

EXIT ;Infinity loop to stop rotation

NOP

b EXIT

loopA

push{lr}

ldrb r1, [r3, r4] ;Stores the current step into r1

lsl r1, #12 ;Shift the bits left to line up with PE12-14

str r1, [r0, #GPIOx\_ODR] ;Store to GPIOE\_ODR

bl DLY

cmp r2, #0

addne r4, r4, #1 ;increment step array position

cmp r4, #4 ;Set's step array posiotion max to 3

movpl r4, #0

subs r2, #1 ;decrement rotation counter

bpl loopA ;if rotation counter is >=0 loop

pop{lr}

bx lr ;else return to motor

loopB

push{lr}

ldrb r1, [r3, r4] ;Stores the current step into r1

lsl r1, #12 ;Shift the bits left to line up with PE12-14

str r1, [r0, #GPIOx\_ODR] ;Store to GPIOE\_ODR

bl DLY

cmp r2, #0

subne r4, r4, #1 ;decrement step array position

cmp r4, #0 ;Set's step array posiotion min to 0

movmi r4, #3

subs r2, #1

bpl loopB

pop{lr}

bx lr

DLY

push{lr}

mov r6, #0x800 ;Default step delay

again

NOP

subs r6, #1 ;run through delay

bpl again

pop{lr}

bx lr

area constants, data, readonly

align 1

;Data array containing the values for:

;One Phase stepping in the UPPER 4bits

;Two Phase stepping in the LOWER 4bits

;Half-Stepping is achieved by first reading the

;upper 4 bits then reading the lower 4bits.

step DCB 0x15,0x46,0x2A,0x89

1. Noise and jerky movement in the motor are caused primarily by running the motor too fast and/or by incorrectly wiring it.

When the step sequence is sent to the motor too quickly, the magnetic fields on the coils don’t have time to fully collapse. In extreme cases this will result in a humming noise, which changes as the frequency of the step sequence changes, and the motor not being able to rotate at all. If the frequency is on the borderline, of what the motor can handle, then occasional stutters in the motor’s rotation will occur. To rectify this issue, find the maximum frequency of step sequence the at which the motor can run smoothly and limit its speed to what was that.

If the user accidently mixes the phase wires together, ie: instead of wiring the 4bit bus as A,A`,B,`B you accidently do something like: A,B,`A ,`B, the motor will generate noise and not rotate. This can be rectified in tow ways:

First you can change the step sequence you send on your bus to account for this.

Or you could simply rewire it correctly.

## Conclusions

Lab #5’s objectives were achieved and demonstrated to a TA. No major issues were encountered. Interrupts were implemented to handle all button inputs. And the stepper motor was successfully wired up and run.

## Appendix A

