**Experiment #2: Danger! Keep Right!**

**ECE 367 – Microprocessor Design (Spring 2012)**

PROFESSOR: Becker

MWF – 10:00AM – 11:50PM

T Lab: 8:00AM – 10:50AM

Prepared by: Mitchell Hedditch

UIN: 677318273

Date Prepared: Sunday, February 3rd, 2013

Date Submitted: Tuesday, February 5th, 2013

1. Logic Diagram
2. Schematic Diagram
3. 9S12 Assembler Program

; University of Illinois at Chicago, Dept. of Electrical and Computer Engineering

; ECE 367 -Microprocessor-Based Design

; Semester: Spring 2013

; Experiment Title: Danger! Keep Right!

; Date: 2/2/2013

; Version: 1

; Programmer: Mitchell Hedditch

; Lab Session: Tuesday 8AM-10:50AM

Regbas EQU $0000 ; Register block starts at $0000

PortA EQU $00 ; PortA address (relative to Regbase i.e. offset)

PortT EQU $0240 ; PortT offset (actual address of PortT)

DDRA EQU $02 ; PortA Data Direction control register offset

DDRT EQU $0242 ; Actual Data Direction Register for PortT

; Begin Code

; Initialize the 68HC11

ORG $4000 ; Place code in Flash EEPROM starting at $4000

START: LDS #$3FC0 ; Initialize stack pointer ALWAYS use this value if you

; intend to run debug. Else use $$4000

LDY #Regbas ; Initialize register base address

; Note that Regbas = $0000 so now <Y> = $0000

; Setup the data directon for PortA and PortT

BSET DDRA,$FF ; PortA pins are outbound

BSET DDRT,$FF ; PortT pins are outbound

BSET PortT,$00 ; Make Sure all PortT pins are low

; Start the program loop

LOOP: LDX #TABLE ; Initialize index X to beginnng of the table

NXT: LDAA #$11 ; Load our last table value into accumulator A

LDAB 1, X+ ; Load accumulator B with the value at X and

; then decrement X by 2

CBA ; Compare accum B to A

BEQ LOOP ; If we are at the end of our table, then restart the loop

STAB PortT ; Output the results to port t

JSR Sec\_Delay ; Jump to subroutine Sec\_Delay - 4 clock cycles

JMP NXT ; Continue displaying LEDs

Sec\_Delay:

LDAA #100 ; Outer Loop counter - 1 clock cycle

A1: LDY #3000 ; Inside Loop Counter 2 clock cycles

A0: LBRN A0 ; 3 clock cycles \

DEY ; 1 clock cycles | 8 clock cycles in loop

LBNE A0 ; 4 clock cycles /

DECA ; 1 clock cycles

BNE A1 ; 3 clock cycles

RTS ; Return from subroutine - 5 clock cycles

; when we get here we have

; ([(8\*30000) + (2) + (1) + (3)]\*100) + 1 + 5

; 24000606 clock cycles or approx 1 sec.

; Have the Assembler put the solution data in the look-up table

ORG $5000 ; The look-up table is at $5000

TABLE: DC.B $80, $C0, $E0, $F0, $F8, $FC ; Define data to be stored.

DC.B $FE, $FF, $00, $FF, $00, $FF ; i.e. the solutions

DC.B $00, $FF, $00, $FF, $11 ; This line includes the end of the blinking and the ;stop code

; End of code

; Define Power-On Reset Interrupt Vector - Required for all programs!

; AGAIN - OP CODES are at column 9

ORG $FFFE ; $FFFE, $FFFF = Power-On Reset Int. Vector Location

FDB START ; Specify instruction to execute on power up

END ; (Optional) End of source code

; Labels start in the first column (left most column = colunm 1)

; OP CODES are at column 9

; COMMENTS follow a ";" symbol

; Blank lines are allowed (Makes the code more readable)

1. USER MANUAL
   1. This system displays a “Keep Right Arrow” by lighting LEDs from successively from left to right and then blinking all lit LEDs four times.
   2. To operate the counter:
      1. Make sure the system is plugged into power via the USB cord to computer.
      2. To begin the program simply press the black reset button on the breadboard (this is the only button available to the user).
      3. The system will continue to cycle as long as it connected to power.
2. Conclusion.
   1. How well does your project meet the specifications?
      1. The project meets all specifications by having the ability to count down from 255 to 0.
   2. What were the most difficult issues in realizing the system?
      1. Because this system was laid out by the professor, there was no real difficulty in realizing this system.
   3. Were you able to add extra features? If so, explain them.
      1. No, but I was able to successfully modify the system to count faster and slower as well as count forwards and backwards.
   4. What would you have done differently if you were to do this project again?
      1. There was nothing I would have done differently for this project.
   5. What did you learn from working on this project?
      1. I learned how to set up the CodeWarrior environment and set up a new project, set up the power module and microcontroller to work from USB power, as well as load code from CodeWarrior onto the chipset.