**Experiment #3: Now Serving**

**ECE 367 – Microprocessor Design (Spring 2013)**

**PROFESSOR:** Robert Becker

**T.A.:** Chenjie Tang

MWF – 10:00AM – 11:50PM

T Lab: 8:00AM – 10:50AM

Prepared by: Mitchell Hedditch

UIN: 677318273

Date Prepared: Sunday, February 10th, 2013

Date Submitted: Tuesday, February 12th, 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: Now Serving

; Date: 2/10/2013

; Updated: 2/10/2013

; Version: 1

; Programmer: Mitchell Hedditch

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

; Define symbolic constants

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

PortM EQU $0250 ; PortM offset (actual address of PortM)

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

DDRA EQU $02 ; PortA Data Direction control register offset

DDRM EQU $0252 ; PortM Data Direction control register offset

DDRT EQU $0242 ; Actual Data Direction Register for PortT

INITRG EQU $11

INITRM EQU $10

CLKSEL EQU $39

PLLCTL EQU $3A

CRGFLG EQU $37

SYNR EQU $34

REFDV EQU $35

COPCTL EQU $3C

TSCR1 EQU $46

TSCR2 EQU $4D

TIOS EQU $40

TCNT EQU $44

TC0 EQU $50

TFLG1 EQU $4E

Regbas EQU $0000 ; Register block starts at $0000

SAVE\_Y EQU $3800 ; Defines location for the storage of the Y index register

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; The ORG statment below is followed by variable definitions

; THIS IS THE BEGINNING SETUP CODE

;

ORG $3800 ; Beginning of RAM for Variables

;

; The main code begins here. Note the START Label

;

ORG $4000 ; Beginning of Flash EEPROM

START LDS #$3FC0 ; Top of the Stack

SEI ; Turn Off Interrupts

MOVB #$00, INITRG ; I/O and Control Registers Start at $0000

MOVB #$39, INITRM ; RAM ends at $3FFF

;

; We Need To Set Up The PLL So that the E-Clock = 24MHz

;

BCLR CLKSEL,$80 ; disengage PLL from system

BSET PLLCTL,$40 ; turn on PLL

MOVB #$2,SYNR ; set PLL multiplier

MOVB #$0,REFDV ; set PLL divider

NOP ; No OP

NOP ; NO OP

PLP BRCLR CRGFLG,$08,PLP ; while (!(crg.crgflg.bit.lock==1))

BSET CLKSEL,$80 ; engage PLL

;

;

;

CLI ; Turn ON Interrupts

;

; End of setup code. You will always need the above setup code for every experiment

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Begin Code

; Initialize the 68HC11

LDY #Regbas ; Initialize register base address

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

; Setup the data directon for PortA and PortT

BCLR DDRM,$FF ; Set all pins of PortM to inbound

BSET DDRM,$0C ; PortM pins 2 & 3 are outbound

BCLR PortM,$0C ; Set pins 2 & 3 of PortM to low

BSET DDRT,$FF ; set all PortT pins to outbound

BCLR PortT,$FF ; Make Sure all PortT pins are low

LDX #TABLE ; Load the beginning address of the table into x

LDY #TABLE ; Load the beginning address of the table into y

JSR SET\_TENS ; Call set tens to start LCD1

JSR SET\_ONES ; Call set ones start LCD2

POLL: BRSET PortM,$01, POLL ; If the increment button is not pushed, branch here

CPY #TABLE ; Compare ones index (Y) to 9

BNE ONES ; If ones place not 9 branch to ONES so we only

; Change the ones LCD

JSR SET\_TENS ; Increment the tens LCD otherwise

ONES: JSR SET\_ONES ; Call set ones

BCLR PortM,$0C ; Set PortM pins 2 and 3 back to low

STY SAVE\_Y ; Save the value of Y b/c it will change in delay

JSR Sec\_Delay ; Delay a bit to let the user let go of button

LDY SAVE\_Y ; Reload Y from it's saved location after delay

JMP POLL ; Jump back and begin polling M0 again

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SET\_TENS:

LDAA 1,X+ ; Load A with X (post-increment X)

BCLR PortM,$0C ; Set pins 2 & 3 of PortM to low (just to make sure)

BSET PortM,$04 ; Set pin 2 to high for latch for TEN's LCD

STAA PortT ; Output new value for LCD to PortT

CPX #TABLE+10 ; Compare index X to TABLE+10

BNE CON\_X ; As long as index X is != 10 skip the reset to 0

LDX #TABLE ; Reset the index X to the base value of TABLE

CON\_X: RTS ; Return from subroutine

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SET\_ONES:

LDAA 1,Y+ ; Load A with Y (post-increment Y)

BCLR PortM,$0C ; Set pins 2 & 3 of PortM to low (just to make sure)

BSET PortM,$08 ; Set pin 3 to high for latch for TEN's LCD

STAA PortT ; Output new value for LCD to PortT

CPY #TABLE+10 ; Compare index Y to TABLE+10

BNE CON\_Y ; As long as index Y in != 10 skip the reset to 0

LDY #TABLE ; Reset the index Y to the base value of TABLE

CON\_Y: RTS ; Return from subroutine

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Sec\_Delay:

LDAB #100 ; Outer Loop counter - 1 clock cycle

A1: LDY #20000 ; 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 /

DECB ; 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 $5500 ; The look-up table is at $5000

TABLE: DC.B $6F, $03, $5D, $57, $33 ; Define data table of mappings to each of the

DC.B $76, $7E, $43, $7F, $73 ; segments of the 7-segment LED displays

; Memory locations correspond to their values

; i.e. $5500 = 0, $5501 = 1, etc

; 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. Start Up
      1. To start this system, use the USB to miniUSB cable and connect to computer. If you have a USB-outlet adapter, power may be supplied this way as well.
      2. Verify the system is in “Run” mode (the switch on the microcontroller board).
      3. Press the reset button to begin (this is the button located closest to the power terminals).
   2. Operation
      1. After pressing the reset button, the system will initialize to “00”.
      2. The system operates using the “Increment” button located opposite the reset button. Each time the increment button is pressed, the system will increment the display by 1.
      3. The system has a maximum count of 99, after which if the increment button is pressed again it will restart at zero.
      4. The system can be restarted/reinitialized at any time by simply pressing the reset button during operation.
   3. Shut Down
      1. To shut the system down, disconnect the power source (USB cable) from the breadboard.
2. Conclusion.
   1. How well does your project meet the specifications?
      1. It meets the requirements as stated in the experiment description.
         1. Initializes to 00
         2. Counts to 99
         3. Cycles back to 00 after 99
         4. Can be reset at any time to 00
         5. Uses PT1-PT7 as output
         6. Uses PM0 as increment
         7. Uses PM2 & PM3 as latch enable
   2. What were the most difficult issues in realizing the system?
      1. First would be the timer delay, specifically timing it so that the user could reliably press the button and release causing only one increment.
      2. Due to the layout of the latches (the way the gates were arranged), wiring turned out to be very problematic. I ultimately was unhappy with the way it looked, however time restrictions prevented me from cleaning up the wires like I had planned.
   3. Were you able to add extra features? If so, explain them.
      1. I was not able to add any extra features for this particular experiment.
   4. What would you have done differently if you were to do this project again?
      1. To start, after having written the code, I would have tried to condense it even more. It is quite compact, but I believe I could do it with even less lines of code and commands.
      2. Second, instead of doing my wiring diagram first like I had planned, I just dove into it and began hooking up the connections. At first I wired the latches backwards, but after realizing my mistakes the system worked correctly. If I had written a wiring diagram first and then begun wiring, I wouldn’t have made that mistake and I would have been able to organize my wires much better.
   5. What did you learn from working on this project?
      1. I learned how valuable it is to start coding with pseudo-code before doing the actual coding of a project.
      2. I learned the benefits of putting a wiring diagram together before connecting the devices.
      3. I gained a better understanding of multiple commands including:
         1. Reasons for storing data in tables such as the corresponding LED’s of a seven segment LED for numbers.
         2. Subroutines and proper structuring.
         3. Switch de-bouncing techniques.
         4. Switching pins from inbound<->outbound
         5. Proper techniques and reasons for initializing and re-initializing pins.
         6. PUSY
         7. BCLR
         8. BSET
         9. BRCLR
         10. BRSET
         11. RTS
         12. JSR
         13. LDY
         14. LDX