**Experiment #4: 24-Second Shot Clock**

**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 17th, 2013

Date Submitted: Tuesday, February 19th, 2013

1. Logic Diagram
2. Schematic Diagram
   1. See attached sheet following this page.
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: 24 Second Shot Clock

; Experiment Description: This system is a 24 second shot clock which counts down

; from 24 seconds to 0 seconds and has the ability to pause

; and restart when the button is pressed. When the clock

; reaches 00, it will blink three times indicating the

; timer is done and the clock will reset.

; Date: 2/15/2013

; Updated: 2/15/2013

; 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

IRQCR EQU $001E ; IRQ CONTROL REGISTER ADDRESS LABEL

Regbas EQU $0000 ; Register block starts at $0000

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

FLAG1 EQU $3802 ; DEFINES LOCATION FOR STORAGE OF FLAG1 FOR INTERRUPTS

; FLAG = 0->PAUSE; 1->RUN

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

; 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 ALL 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 S BIT ON INTERRUPTS

BSET IRQCR,$C0 ; TURN ON IRQ' INTERRUPT AND SET TO EDGE TRIGGERED

; 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

; INITIALIZE THE ENTIRE SYSTEM VARIABLES AND BEGIN OPERATIONAL CODE

BEGIN: MOVB #$00,FLAG1 ; INITIALIZE THE SYSTEM IN PAUSE MODE

LDX #TABLE+2 ; Load the beginning address of the table into x(2)

LDY #TABLE+4 ; Load the beginning address of the table into y(4)

JSR SET\_TENS ; CALL SET TENS TO START LCD1

JSR SET\_ONES ; CALL SET ONES TO START LCD2

POLL: BRCLR FLAG1,$01,POLL ; WAIT AT POLL UNTIL THE IRQ' INTERRUPT FLAG IS SET

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

BNE ONES ; IF ONES IS AT 9, THEN

JSR SET\_TENS ; INCREMENT THE TENS LCD

ONES: JSR SET\_ONES ; INCREMENT THE ONES LCD

BCLR PortM,$0C ; SET PortM PINS 2 & 3 TO LOW

CPX #TABLE+9 ; Compare X to TABLE+9

BNE NOBUZZ ; Branch if X=09 to END\_RESET

CPY #TABLE+9 ; COMPARE Y TO TABLE+9

BNE NOBUZZ ; BRANCH TO NOBUZZ WHEN NOT AT 9 FOR ONES

JSR BUZZ ; JUMP BUZZ SUBROUTINE

JSR BEGIN ; JUMP TO BEGIN SUBROUTINE

NOBUZZ: STY SAVE\_Y ; SAVE THE VALUE OF Y TO MEMORY DURING THE DELAY

JSR Sec\_Delay ; DELAY OPERATION BY CLOCK COUNTING FOR 1 SECOND

LDY SAVE\_Y ; RELOAD Y FROM MEMORY SINCE THE DELAY CHANGED IT

JMP POLL ; GO BACK TO POLLING LABEL AND START AGAIN

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

SET\_TENS:

LDAA 1,X- ; Load A with X (post-decrement 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-1 ; Compare index X to TABLE-1

BNE CON\_X ; As long as index X is != TABLE-1 skip reset to 0

LDX #TABLE+9 ; GO BACK TO 9 SINCE WE'RE COUTING DOWN

CON\_X: RTS ; Return from subroutine

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

SET\_ONES:

LDAA 1,Y- ; Load A with Y (post-decrement 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-1 ; Compare index Y to TABLE-1

BNE CON\_Y ; As long as index Y in != TABLE-1 skip reset to 0

LDY #TABLE+9 ; GO BACK TO 9 SINCE WE'RE COUNTING DOWN

CON\_Y: RTS ; Return from subroutine

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

BUZZ:

BSET PortM,$0C ; SET BOTH Portm PINS TO HIGH FOR OUTPUT

LDAA #$00 ; LOAD A WITH ZERO SO DISPLAY GOES BLANK

STAA PortT ; OUTPUT VALUE IN A TO PortT

JSR Buzzer\_Delay ; SLIGHT DELAY FOR BLINK EFFECT

LDAA TABLE ; LOAD A WITH TABLE VALUE TO DISPLAY 00

STAA PortT ; OUTPUT VALUE IN A TO PortT

JSR Buzzer\_Delay ; SLIGHT DELAY FOR BLINK EFFECT

BRA BUZZ

RTS ; RETURN FROM SUBROUTINE

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

Sec\_Delay:

LDAB #100 ; Outer Loop counter - 1 clock cycle

A1: LDY #30000 ; Inside Loop Counter 2 clock cycles

A0: BRCLR FLAG1,$01,A0 ; WAIT AT POLL UNTIL THE IRQ' INTERRUPT FLAG IS SET

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.

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

Buzzer\_Delay:

LDAB #100 ; Outer Loop counter - 1 clock cycle

A2: LDY #6000 ; Inside Loop Counter 2 clock cycles

A3: LBRN A3 ; 3 clock cycles \

DEY ; 1 clock cycles | 8 clock cycles in loop

LBNE A3 ; 4 clock cycles /

DECB ; 1 clock cycles

BNE A2 ; 3 clock cycles

RTS ; Return from subroutine - 5 clock cycles

; when we get here we have

; ([(8\*6000) + (2) + (1) + (3)]\*100) + 1 + 5

; 4,800,600 clock cycles or approx 1/6TH sec.

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

ISR\_IRQ: COM FLAG1 ; TOGGLE THE START/PAUSE FLAG

RTI ; RETURN FROM INTERRUPT

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

ORG $FFF2 ; IRQ' VECTOR ADDRESS

FDB ISR\_IRQ ; ISR\_IRQ IS A LABEL THE ADDRESS OF THE LABEL

; IS THE VECTOR

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

; 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 “24” in paused mode. The clock will not begin until you press the Start/Pause button.
      2. The system operates using the “Start/Pause” button located opposite the reset button (on the right if the USB cable is closest to you). Each time the “Start/Pause” button is pressed; the system will either Start/Resume or Pause the countdown.
      3. You can Pause the operation at any time by pressing the “Start/Pause” button. To restart the count, simply press the “Start/Pause.”
      4. When the shot clock reaches “00” the display will blink “00” until you press the Reset button.
      5. 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 project requirements as explained on the ECE 367 website for experiment #4. A 24 second shot clock which starts in the “paused” mode. It can be Restarted/Paused at any time during operation. The “buzzer” is indicated by blinking “00.”
   2. What were the most difficult issues in realizing the system?
      1. Learning how to start the clock at 24, but count down to zero. This was different from the previous experiment which counted up to 99.
      2. The most difficult part in realizing this system involved learning and using system interrupts. Combining turning interrupts on both globally and locally, initializing interrupts, setting the interrupt subroutine up and designing the interrupt subroutine. During my interrupt subroutine implementation, I accidentally created an infinite loop which took me about 2 hours to debug.
   3. Were you able to add extra features? If so, explain them.
      1. No special features were added to this experiment.
   4. What would you have done differently if you were to do this project again?
      1. If I had more time to work on this project, I would make the code more efficient as well as organized the wiring better.
   5. What did you learn from working on this project?
      1. The most significant thing that I learned was the setup and operation of system interrupts and how powerful they can be. Due to the fact that they use almost no CPU power/time they don’t interfere with program operation which can free the programmer to process other code while the system waits for user input. Interrupts can also free up other pins on the system board.