Experiment #5

24 Second Shot Clock with Interrupts

ECE 367

Spring 2012

Shuaib Shameem

673551999

February 23rd, 2012

TA: Yasaman Keshtkarjahromi

Schematic, Program Code, and Logic Diagram attached to the end of report.

User Manual:

To reset the program, press the top button.

To start/pause the clock, press the bottom button

Conclusion:

Getting the timer to work was hit or miss. For some reason, the values given in class didn’t work well. With some guess and check, I was able to get the program to run properly. This lab was otherwise easy.

; Author: Shuaib Shameem

; UIN: 673551999

; ECE 367 -Microprocessor-Based Design

; Experiment 4 - 24 Second Shot Clock

; 02/16/2012

;

; Purpose: Design a 24 second count down timer similar to the shot

; clock used in NBA basketball games. The reset button resets the

; count. The pause/stop button pauses and starts the counter.

;PAY ATTENTION TO THE ALIGNMENT BELOW

;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)

; Define symbolic constants

PORTT EQU $240 ;Define Register Locations

PORTM EQU $250

DDRT EQU $242

DDRM EQU $252

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

TC5 EQU $5A

TIE EQU $4C

;

; The ORG statment below would normally be followed by variable definitions

; There are no variables needed for this project.

;

ORG $3800 ; Beginning of RAM for Variables

COUNT: DS.W 1

FLAG1: DS.B 1

TIMEONE:DS.W 1

TIMETEN:DS.W 1

;

; The main code begins here. Note the START Label

;

ORG $4000 ; Beginning of Flash EEPROM

START LDS #$3FCE ; 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

;

;

MOVB #$01, TSCR2 ; set up TSCR2

MOVB #$20, TIOS ; set up TIOS for output

MOVB #$90, TSCR1 ; set up TSCR1

MOVB #$20, TIE ; start interrupt setting

LDD TCNT ; load current count

ADDD #INCREMENT ; increment it by set amount

STD TC5 ; save to TC5 interrupt

MOVB #$20, TFLG1 ; clear flag

;

CLI ; Turn ON Interrupts

LDAA #$00 ; load AA with 0

STAA FLAG1 ; store run flag with 0

LDY #$0002 ; load Y with 2

STY TIMETEN ; store tens counter with 2

LDY #$0005 ; load Y with 5

STY TIMEONE ; store ones counter with 5

LDAA #$FF ; Make PortT Outbound

STAA DDRT

LDAA #$03 ; Make PortM pins 1 and 2 Outbound

STAA DDRM

;

; Initial Reset Location

;

JSR UPDATE ; setup display

TOP: LDD MAXCOUNT ;set up count

STD COUNT

HERE: BRCLR FLAG1, $01, HERE ;check run flag for running

LDD COUNT ;check count value

BNE HERE ; if not zero, check run flag again

JSR UPDATE ; else update display

JSR DONEYET ; check time values

BRA TOP ; run loop again

UPDATE: LDY TIMEONE ; load ones counter

BEQ RESET1 ; if 0, reset to A, tens counter-1

BACK0: DEY ; decrement Y

STY TIMEONE ; store in ones counter

JSR ONES ; update ones display

JSR TEN ; update tens display

RTS ; return

RESET1: LDY TIMETEN ; load time ten

DEY ; decrement time ten

STY TIMETEN ; store time ten

LDY #$000A ; load time one reset value

BRA BACK0 ; return to UPDATE

DONEYET:LDY TIMETEN ; Check time ten for 0

BEQ DONEONE ; if 0, check time one

BACK1: RTS ; else return

DONEONE:LDY TIMEONE ; check time one for 0

BNE BACK1 ; return if not 0

JSR FLASH ; flash display if 0

ONES: LDY TIMEONE ;load ones value

LDAA TABLE, Y ;get LED code for value

STAA PORTT ;output code

BSET PORTM, $01 ;enable latch

NOP

NOP

BCLR PORTM, $01 ;disable latch

RTS ; return

TEN: LDY TIMETEN ;lead tens value

LDAA TABLE, Y ;get LED code for value

STAA PORTT ;output code

BSET PORTM, $02 ;enable latch

NOP

NOP

BCLR PORTM, $02 ;disable latch

RTS ;retun

FLASH BCLR PORTT, $FF ; Clears out PortT

JSR OPENB ; Opens both M1 and M2 for Latch Enable

JSR SDELAY ; delay the program

JSR SDELAY ; Delay the program

BSET PORTT, $3F ; Sets the value $7E in portT

JSR OPENB ; Opens both M1 and M2 for Latch Enable

JSR SDELAY ; Delay the program

JSR SDELAY ; delay the program

BRA FLASH ; Branch to "flashy" for infinite loop

;

; Open both ports

;

OPENB BSET PORTM, $03 ; Set Bits of PortM to 11

NOP

NOP

BCLR PORTM, $FF ; Clear all the bits of PortM

NOP

RTS

;

; We use the CPU clock cycles to create a delay

;

; Delay of about 1 Sec with the switching control

;

SDELAY: PSHY

LDY #65535 ; Loop counter = 65535 - 2 clock cycles

A0: LBRN A0 ; 3 clock cycles \

DEY ; 1 clock cycles | 8 clock cycles in loop

LBNE A0 ; 4 clock cycles / Time = 8\*<Y>/(24\*10\*\*6) + 2 =

; ; [8X65535 + 2]/24000000 ~= 20msec

PULY

RTS

;

; End of counter code

ISR\_IRQ:COM FLAG1 ; Complement Run Flag

JSR SDELAY ; Short delay to counter debounce

RTI ; return to program

ISR\_TC5:LDD TC5 ; Load Counter 5

ADDD #INCREMENT ; Increment by set amount

STD TC5 ; Store Counter 5

MOVB #$20, TFLG1 ; Insure flag is cleared

BRCLR FLAG1, $01, DONE ; check run flag

LDD COUNT ; if running, load count

SUBD #$0001 ; decrement count

STD COUNT ; store count

DONE: RTI ; return to program

ORG $5000

;

; Table of 7 segment LED values as bits 7-0 as gfedcba0.

;

TABLE: DC.B $3F, $06, $5B, $4F, $66, $6D, $7D, $07, $7F, $67, $00

;

; Order :0, 1, 2, 3, 4, 5, 6, 7, 8, 9, off

MAXCOUNT:DC.W $0250 ;Count size when program starts/number changes

INCREMENT: DC.W $0177 ;Amount to increase TC5 by

; Define TC5 Interrupt Vector

ORG $FFE4

FDB ISR\_TC5

; Define IRQ' Interrupt Vector

ORG $FFF2

FDB ISR\_IRQ

; Define Power-On Reset Interrupt Vector

; 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 of Interrupt code

END ; (Optional) End of source code