**Experiment #8: Count Up/Down Timer Using The SPI Subsystem and LCD Display**

**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: Monday, March 18th, 2013

Date Submitted: Thursday, March 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: Count Up/Down Timer Using The SPI Subsystem and LCD Display

; Experiment Description: This system is a timer that is capable of starting and

; pausing operation, as well as reversing the count direction.

; It will count from 00 to 99 or 99 to 00 and when it reaches 00

; again, it will blink 3 times and then reset the system. You

; can enter number values on the keypad at any time to change the

; timing values. All data displayed on LCD.

; Date: 3/15/2013

; Updated: 3/18/2013

; Version: 1

; Programmer: Mitchell Hedditch

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

; Define symbolic constants

REGBAS EQU $0000 ; REGISTER BLOCK STARTS AT $0000

PortA EQU $0000 ; PortA address (relative to Regbase i.e. offset)

DDRA EQU $0002 ; PortA Data Direction control register offset

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

DDRM EQU $0252 ; PortM Data Direction control register offset

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

DDRT EQU $0242 ; Actual Data Direction Register for PortT

PortE EQU $0008 ; PortE LABEL (XIRQ' INTERRUPT)

; TIMER SYMBOLIC CONSTANTS

TSCR1 EQU $0046 ; TIMER SYSTEM CONTROL REGISTER - WITH FAST FLAGS

TSCR2 EQU $004D ; TIMER SYSTEM CONTROL REGISTER 2 - NO FAST FLAGS

TFLG1 EQU $004E ; TIMER INTERRUPT FLAG1 REGISTER

TFLG2 EQU $004F ; TIMER INTERRUPT FLAG2 REGISTER

TIOS EQU $0040 ; TIMER INTERRUPT OUTPUT COMPARE

TCNT EQU $0044 ; TIMER COUNTER REGISTER - 16 BIT, INPUT CAPTURE/OUTPUT COMPARE REQUIRED

TC0 EQU $0050 ; TIME I/O COMPARE SELECT 0 REGISTER TO LOCATION $50 HEX

TC1 EQU $0052 ; TIME I/O COMPARE SELECT 1 REGISTER TO LOCATION $52 HEX

TIE EQU $004C ; TIMER TCi INTERRUPT ENABLE REGISTER

; INTERRUPT CONSTANTS

IRQCR EQU $001E ; IRQ CONTROL REGISTER ADDRESS LABEL

; SERIAL COMMUNICATION INTERFACE

SPCR1 EQU $00D8

SPCR2 EQU $00D9

SPIB EQU $00DA

SPSR EQU $00DB

SPDR EQU $00DD

ENABLE EQU $02 ; LCD ENABLE at PM1

RCK EQU $08 ; RCK connect to PM3

;UNKNOWN

INITRG EQU $0011

INITRM EQU $0010

PLLCTL EQU $003A

; CLOCKS

CLKSEL EQU $0039

CRGFLG EQU $0037

SYNR EQU $0034

REFDV EQU $0035

COPCTL EQU $003C ; COMPUTER OPERATING PROPERLY CONTROL LOCATION

TEST EQU $3800 ; DEFINE LOCATION FOR TEST BYTE STORAGE FOR DEBUGGING

SAVE\_X EQU $3802 ; Defines location for the storage of the X index register

SAVE\_Y EQU $3804 ; Defines location for the storage of the Y index register

DIR\_FLAG EQU $3806 ; DEFINES LOCATION FOR STORAGE OF COUNTER DIRECTION FLAG

; FOR INTERRUPTS FLAG = 0->COUNT UP; 1->COUNT DOWN

PAUSE EQU $3808 ; DEFINES LOCATION FOR STORAGE OF START/PAUSE FLAG

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

TMR\_FLAG EQU $3810 ; DEFINES LOCATION FOR STORAGE OF TIMER FLAG

; FLAG= 0->NOTHING; 1->TIMER FIRED

INVALID\_KEY EQU $3812 ; DEFINES LOCATION FOR STORAGE OF TIMER FLAG 2

; FLAG= 0->NOTHING; 1->TIMER FIRED

TIME\_COUNT EQU $3814 ; MEM ADDRESS TO STORE TIME FOR SECONDS

XIRQ\_FLAG EQU $3816 ; PAUSE FOR XIRQ (1 MSEC)

NUM\_FLAG EQU $3818 ; A FLAG THAT GOES TO 1 IF A KEY IS PRESSED ON THE PAD

CUR\_PAD\_VAL EQU $3820 ; USED TO HOUSE THE VALUE FOR THE CURRENT KEYPAD ITERATION

COUNT\_VAL EQU $3822 ; STORE THE COUNT VALUE HERE

CUR\_COLUMN EQU $3824 ; STORAGE LOCATION FOR VARIABLE OF CURRENT COLUMN

RS EQU $01 ; REGISTER SELECT (RS) AT PM0 (0=COMMAND, 1=DATA)

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

; 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

SEI ; TURN OFF INTERRUPTS

; INITIALIZE ALL SYSTEM PORTS/INTERRUPTS/DDRS/FLAGS/ETC

; SETUP S BIT ON INTERRUPTS

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

ANDCC #$BF ; SET THE X-BIT TO USE XIRQ' AS A STANDARD INTERRUPT

JSR INIT ; INITIALIZE ALL OF OUR VARIABLES, FLAGS, ETC.

JSR InitLCD ; INITIALIZE THE LCD

; ALL VARIABLES ARE INITIALIZED SO WE'RE READY FOR INTERRUPTS

CLI ; TURN ON INTERRUPTS

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

;

; MAIN PROGRAM CODE IS HERE

;

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

JSR DIRECTIONS ; DRAW DIRECTIONS

JSR DRAW\_SCREEN ; DRAW FIRST SCREEN FOR THE FIRST TIME

POLL: MOVB #$00,INVALID\_KEY ; RESET INVALID KEY FLAG

MOVB #$00,NUM\_FLAG ; CLEAR THE NUM FLAG TO WAIT FOR A NEW KEY

JSR GET\_KEY ; CHECK THE KEYPAD FOR A PRESSED VALUE

BRCLR NUM\_FLAG,$01,NO\_KEY ; IF NO KEY HAS BEEN PRESSED THEN MOVE ON THE THE NO\_KEY LINE

JSR BSPACE ; GO TO THE BSPACE SUB TO SEE IF USER PRESSED 'C'

JSR CHECK\_KEY ; CHECK TO SEE IF THE KEY IS VALID

BRSET INVALID\_KEY,$01,POLL ; GO BACK AND POLL AGAIN IF WE'VE GOT A BAD KEY

JSR LOAD\_NUMBER ; IF A NUMBER KEY HAS BEEN PRESSED THEN LOAD THE NEW NUMBER

MOVB #$00,NUM\_FLAG ; CLEAR THE NUM FLAG TO WAIT FOR A NEW KEY

NO\_KEY BRSET PAUSE,$01,POLL ; WAIT AT POLL WHILE THE IRQ' (PAUSE) INTERRUPT FLAG IS SET

BRCLR PortE,$01,\* ; BRANCH HERE UNTIL THE XIRQ PORT IS HIGH AGAIN

BRCLR TMR\_FLAG,$01,POLL ; IF THE TIME FLAG ISN'T SET BRANCH BACK TO POLL

BRSET DIR\_FLAG,$01,CDOWN ; IF THE DIRECTION FLAG IS SET, THEN COUNT DOWN

JSR COUNT\_UP ; INCREMENT THE COUNT VALUE

BRA UPDATE\_DISP ; BRANCE TO CONTINUE

CDOWN JSR COUNT\_DOWN ; DECREMENT THE COUNT VALUE

UPDATE\_DISP JSR DRAW\_SCREEN ; LET'S UPDATE THE SCREEN AGAIN

MOVB #$00,TMR\_FLAG ; CLEAR THE TIMER FLAG

LDD COUNT\_VAL ; LOAD THE COUNT VALUE INTO D

CPD #$00 ; CHECK TO SEE IF WE'RE AT ZERO

BNE CONTINUE ; IF WE'RE NOT AT ZERO THEN CONTINUE

JSR BLINK ; BLINK 3 TIMES IF WE'RE AT ZERO

JSR INIT ; RESTART OUR SYSTEM AND REINITIATE ALL FLAGS/VALS

CONTINUE JSR DRAW\_SCREEN ; LET'S UPDATE THE SCREEN JUST IN CASE

BRA POLL ; GO BACK START PROCESSING AT POLL AGAIN!

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

; PROGRAM INITIALIZATION

INIT: ; SETUP THE DATA DIRECTON REGISTERS AND INITIALIZE PORT A & PORT T

MOVB #$F0,DDRT ; SET PortT PINS 4-7 TO OUTBOUND AND PINS 0-3 TO INBOUND

MOVB #$00,PortT ; SET ALL PortT PINS TO LOW

; SET UP SERIAL PROGRAM INTERFACE SYSTEM

MOVB #$22,SPIB ; SPI CLOCKS A 1/24 OF E-CLOCK

MOVB #$3B,DDRM ; SETUP PortM DATA DIRECTION

MOVB #$50,SPCR1 ; ENABLE SPI AND SET MODE AS MASTER

MOVB #$00,SPCR2 ; RESETS SPCR2 TO $00 (ALSO DOES AT RESET)

BSET PortM,RCK ; SET RCK TO IDLE HIGH

BCLR PortM, ENABLE ; ENABLE to Idle LOW

; SET UP TIMER COUNT INFORMATION AND PRESCALE INITIALIZE THE COUNTER

MOVB #$06,TSCR2 ; CONFIGURE PRESCALE FACTOR 64

MOVB #$01,TIOS ; ENABLE OC0 FOR OUTPUT COMPARE

MOVB #$90,TSCR1 ; ENABLE TCNT & FAST FLAGS CLEAR

MOVB #$01,TIE ; ENABLE TC1 INTERRUPT

LDD TCNT ; FIRST GET CURRENT TCNT

ADDD #3750 ; INCREMENT TCNT COUNT BY 3750 AND STORE INTO TC0

STD TC0 ; WE WILL HAVE A SUCCESSFUL COMPARE IN 375 CLICKS

MOVB #$01,TFLG1 ; OF TCNT. BETTER BE SURE FLAG C0F IS CLEAR TO START

; INITIALIZE PROGRAM DEFINED VARIABLES

MOVB #$01,PAUSE ; INITIALIZE IN THE SYSTEM IN PAUSE MODE

MOVB #$00,DIR\_FLAG ; INITIALIZE THE SYSTEM IN COUNT UP MODE

MOVB #$00,TMR\_FLAG ; INITIALIZE THE TIMER FLAG TO LOW

LDD #$0000 ; INITIALIZE THE COUNT TO 0

STD COUNT\_VAL ; STORE THE COUNT VALUE OF D TO MEMORY

MOVB #$00,TIME\_COUNT ; SET TIME\_COUNT TO 0

MOVB #$00,NUM\_FLAG ; SET NUM\_FLAG TO 0 TO

MOVB #$00,INVALID\_KEY ; RESET INVALID KEY FLAG

;SET UP INTRO TEXT TO LCD AND PAUSE HERE

JSR DRAW\_SCREEN ; DURING RESTART, WE'LL NEED TO REDRAW SCREEN

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: TO RETRIEVE A PRESSED KEY FROM A MATRIX KEYBOARD, IF THIS ACTION HAPPENS, SET A FLAG

InitLCD: JSR delay3 ; WE NEED A SHORT DELAY HERE

BCLR PortM,RS ; SEND A COMMAND

LDAA #$30 ; Could be $38 too, 2 LINES AND 5X7 MATRIX

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

JSR delay3 ; need extra delay at startup

LDAA #$30 ; Could be $38 too, 2 LINES AND 5X7 MATRIX

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

JSR delay3 ; WE NEED A SHORT DELAY HERE

LDAA #$30 ; Could be $38 too, 2 LINES AND 5X7 MATRIX

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

JSR delay3 ; DELAY A LITTLE BIT

LDAA #$38 ; Use 8 - words (command or data) and

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

JSR delay3 ; NEED SHORT DELAY TO WAIT FOR COMMAND TO COMPLETE

LDAA #$0C ; Turn on the display

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

JSR delay3 ; NEED SHORT DELAY TO WAIT FOR COMMAND TO COMPLETE

LDAA #$01 ; clear the display and put the cursor

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

JSR delay ; clear command needs more time

JSR delay ; to execute

JSR delay ; NEED SHORT DELAY TO WAIT FOR COMMAND TO COMPLETE

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: LOAD A BIT INTO THE LCD (RS = 0 for commands OR RS = 1 FOR PRINT)

LCD\_INPUT:

SPI\_EF: BRCLR SPSR,$20,SPI\_EF ; WAIT FOR REGISTER EMPTY FLAG (SPIEF)

STAA SPDR ; OUTPUT COMMAND VIA SPI TO SIPO

CKFLG1 BRCLR SPSR,$80,CKFLG1 ; WAIT FOR SPI FLAG

LDAA SPDR ; LOAD FROM SPI TO CLEAR FLAG

NOP ; WAIT

BCLR PortM, RCK ; PULSE RCK

NOP ; WAIT

NOP ; WAIT

BSET PortM, RCK ; COMMAND NOW AVAILABEL FOR LCD

NOP ; WAIT

NOP ; PROBABLY DON'T NEED TO WAIT

NOP ; BUT WE WILL, JUST IN CASE...

BSET PortM, ENABLE ; FIRE ENABLE

NOP ; WE SHOULD WAIT AGAIN

NOP ; UNTIL IT'S FINISHED

BCLR PortM, ENABLE ; ENABLE OFF

JSR delay ; GIVE THE LCD TIME TO TAKE COMMAND IN

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: TO RETRIEVE A PRESSED KEY FROM A MATRIX KEYBOARD, IF THIS ACTION HAPPENS, SET A FLAG

; AND STORE THE VALUE

GET\_KEY: LDX #KP\_VALUE ; LOAD X WITH MEM ADDRESS FOR KP\_VALUE

STX CUR\_PAD\_VAL ; STORE THE ADDRESS OF THE FIRST KEYPAD VALUE

LDX #ROW ; LOAD X WITH THE INITIAL VALUE AT THE ROW ADDRESS

LDY #COLUMN ; LOAD Y WITH THE INITIAL VALUE AT THE COLUMN ADDRESS

; NOW WE BEGIN OUR LOOPING

NEXT\_ROW LDAA 1,X+ ; LOAD ACCUM A WITH CURRENT ROW VALUE POST INCREMENT

NEXT\_COLUMN LDAB 1,Y+ ; LOAD ACCUM Y WITH CURRENT COLUMN VALUE POST INCREMENT

STAA PortT ; SET THE CURRENT ROW TO HIGH VALUE

STAB CUR\_COLUMN ; STORE THE CURRENT COLUMN VALUE

PSHA ; PUSH ONTO THE STACK OR IT WILL BE LOST

PSHB ; PUSH B ONTO THE STACK OR IT WILL BE LOST

NOP ; WAIT SOME TIME FOR PIN TO GO HI

NOP ; WAIT SOME TIME FOR PIN TO GO HI

NOP ; WAIT SOME TIME FOR PIN TO GO HI

ABA ; ADD B TO A TO GET ALL PINS THAT SHOULD BE HIGH

LDAB PortT ; LOAD THE VALUE IN PortT INTO ACCUM B

CBA ; CHECK THE CURRENT BIT IN PortT TO OUR CURRENT COLUMN

BEQ KEY\_PRESSED ; IF THE KEY IS PRESSED THEN MAKE IT SO!

LDD CUR\_PAD\_VAL ; LOAD THE CUR\_PAD\_VAL INTO D

ADDD #1 ; ADD 1 TO D

STD CUR\_PAD\_VAL ; STORE D BACK INTO THE PAD VALUE

PULB ; GET B BACK FROM THE STACK FIRST

PULA ; NOW RESTORE A FROM THE STACK

CPY #COLUMN+4 ; CHECK TO SEE IF WE'RE AT THE END OF THE COLUMNS

BNE NEXT\_COLUMN ; IF NOT, THEN GO BACK AND TRY NEXT COLUMN

LDY #COLUMN ; IF WE ARE THEN RESET THE COLUMNS

CPX #ROW+4 ; CHECK TO SEE IF WE'RE AT THE END OF THE ROWS

BNE NEXT\_ROW ; IF WE'RE NOT AT END OF ROWS, GO TO NEXT ROW

RTS ; RETURN FROM THE SUBROUTINE IF WE'VE PROCESS ALL ROWS AND COLUMNS

KEY\_PRESSED PULB ; GET B BACK FROM THE STACK FIRST

PULA ; NOW RESTORE A FROM THE STACK

MOVB #$01,NUM\_FLAG ; SET NUM\_FLAG SINCE A NUMBER WAS PRESSED

JSR KEY\_RELEASE ; NOW WE NEED TO WAIT UNTIL THE KEYS ARE RELEASED

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: WAIT UNTIL A PRESSED KEY IS RELEASED TO ELIMINATE BOUNCE AND DOUBLE PRESSING

KEY\_RELEASE: MOVB #$F0,PortT ; SET ROWS 4,5,6,7 OF PortT TO HIGH

NOP ; SHORT TIME WAITING FOR PINS TO GO HIGH

BRCLR PortT,$0F,FINISH ; WHEN COLUMN 1-4 (PM0-PM3) IS CLEAR THEN ALL KEYS

; HAVE BEEN RELEASED

BRA KEY\_RELEASE ; BRANCH BACK TO KEY RELEASE

FINISH RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: TO CHECK AND MAKE SURE WE HAVE A VALID KEY PRESSED

CHECK\_KEY: LDX CUR\_PAD\_VAL ; GET THE CURRENT KEYPAD VALUE ADDRESS

LDAA X ; LOAD THE KEYPAD VALUE ADDRESS

CMPA #$09 ; WAS THIS KEY AN INVALID KEY?

BGT INVALID ; IF IT WAS THEN SET THE FLAG

RTS ; IF NOT RETURN FROM SUBROUTINE

INVALID MOVB #$01,INVALID\_KEY ; SET THE INVALID KEY FLAG

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: THIS SUBROUTINE IS USED TO LOAD A NEW DIGIT INTO THE LCD AND THE COUNT VALUE

LOAD\_NUMBER: JSR PREP\_VALS ; LETS LET THE PREP\_VALS SUB SPLIT THE NUMBER

TFR Y,X ; TRANSFER THE ONES VALUE INTO THE X INDEX

LDD #$000A ; LOAD D WITH DECIMAL 10

EMUL ; MULTIPLY THE TENS VALUE BY 10 AND PLACE IN D

STD COUNT\_VAL ; STORE OUR TENS VALUE INTO COUNT\_VAL

LDY CUR\_PAD\_VAL ; LOAD THE EFFECTIVE ADDRESS INTO Y (NEW VALUE)

LDAA #$00 ; CLEAR A OUT BY WRITING ZEROS TO IT

LDAB Y ; LOAD B WITH THE ADDRESS IN Y

ADDD COUNT\_VAL ; ADD OUR KEYPAD VALUE TO THE TENS VALUE

STD COUNT\_VAL ; STORE THE NEW VALUE INTO COUNT\_VAL

JSR DRAW\_SCREEN ; MAKE SURE WE REDRAW THE SCREEN NOW

RTS ; RETURN FROM SUBROUTINE

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

; NOTE: DO NOT CHANGE THIS SUBROUTINE UNLESS YOU MODIFY LOAD\_NUMBER!!

; PURPOSE: TAKE THE VALUE IN COUNT\_VAL AND PARSE ITS ONES AND TENS DIGIT INTO THE X AND Y INDEX

; FOR USE IN THE DISPLAYS

PREP\_VALS: LDD COUNT\_VAL ; LOAD THE COUNT VALUE INTO D

CPD #$000A ; COMPARE X TO 10

LBLO UNDR\_TEN ; IF IT'S LESS THEN 10 MAKE IT ZERO

LDX #0010 ; PLACE TEN IN D

IDIV ; DIVIDE OUR NUMBER BY 10

TFR D,Y ; TRANSFER THE REMAINDER INTO Y

RTS ; RETURN FROM SUBROUTINE

UNDR\_TEN LDX 0 ; LOAD ZERO INTO X

TFR D,Y ; WE LEAVE Y AS IT IS

RTS ; RETURN FROM SUBROUTINE

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

BSPACE: LDX CUR\_PAD\_VAL ; GET THE KEY PRESSED VALUE ADDRESS

LDAA X ; LOAD THE KEY VALUE

CMPA #$0C ; WAS THIS KEY A 'C'?

BNE NO\_BSPC ; IF IT WAS THEN SET THE FLAG

LDD COUNT\_VAL ; LOAD THE COUNT VALUE INTO D

CPD #$000A ; COMPARE X TO 10

LBLO UNDR\_TEN2 ; IF IT'S LESS THEN 10 MAKE IT ZERO

LDX #0010 ; PLACE TEN IN D

IDIV ; DIVIDE OUR NUMBER BY 10

BRA ST\_CT ; GO TO STORE THE COUNT ST\_CT

UNDR\_TEN2 LDX 0 ; LOAD ZERO INTO X

ST\_CT STX COUNT\_VAL ; STORE OUR NEW COUNT VALUE

BCLR PortM,RS ; SEND A COMMAND TO LCD

LDAA #$10 ; SEND BACKSPACE CHARACTER TO DISPLAY

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

BSET PortM,RS ; SEND A COMMAND TO LCD

LDAA #$20 ; SEND BACKSPACE CHARACTER TO DISPLAY

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

BCLR PortM,RS ; SEND A COMMAND TO LCD

LDAA #$10 ; SEND BACKSPACE CHARACTER TO DISPLAY

JSR LCD\_INPUT ; OUTPUT CLEAR TO SIPO SERIALLY

NO\_BSPC RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: PRINT A STRING TO THE LCD (USES LCD\_INPUT)

PRINT\_STRING:

Loop1 LDAA 0,X ; LOAD A CHARACTER INTO ACMA

BEQ Done1 ; QUIT IF WE REACH A $00

JSR LCD\_INPUT ; AND OUTPUT THE CHARACTER

INX ; GO TO NEXT CHARACTER

BRA Loop1 ; PROCESS NEXT CHARACTER

Done1 RTS ; RETURN FROM SUBROUTINE

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

DRAW\_SCREEN: BCLR PortM, RS ; SEND A COMMAND TO LCD

LDAA #$01 ; CLEAR SCREEN COMMAND

JSR LCD\_INPUT ; SEND TO LCD

LDAA #$02 ; RETURN TO HOME COMMAND

JSR LCD\_INPUT ; SEND COMMAND

; CHECK WHETHER SYSTEM IS PAUSED OR NOT

BSET PortM, RS ; LET'S PRINT TO LCD

BRCLR PAUSE,$01,RUN ; WAIT AT POLL WHILE THE IRQ' (PAUSE) INTERRUPT FLAG IS SET

LDX #STRING1 ; IF WE'RE IN PAUSE MODE PRINT 'PAUSED'

BRA DIR ; GO TO DIR TO PRINT VALUE

RUN LDX #STRING2 ; IF WE'RE RUNNING PRINT 'RUN'

DIR JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB

; CHECK WHETHER WE'RE COUTNING UP OR DOWN

BRSET DIR\_FLAG,$01,DOWN ; IF THE DIRECTION FLAG IS SET, THEN COUNT DOWN

LDX #STRING3 ; IF WE'RE COUTING UP PRINT 'UP'

BRA PCOUNT ; GO TO PCOUNT TO PRINT

DOWN LDX #STRING4 ; IF WE'RE COUTING DOWN PRINT 'DOWN'

PCOUNT JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB TO PPRINT

BCLR PortM, RS ; SENT A COMMAND TO LCD

LDAA #$C0 ; GO TO SECOND LINE TO PRINT

JSR LCD\_INPUT ; SEND COMMAND

; PRINT THE CURRENT COUNT

BSET PortM, RS ; LET'S PRINT TO LCD

LDX #STRING5 ; PRINT 'COUNT'

JSR PRINT\_STRING ; LET'S PRINT THE STRING NOW

JSR UPDATE\_TENS\_DISPLAY ; PRINT 10'S VALUE

JSR UPDATE\_ONES\_DISPLAY ; PRINT 1'S VALUE

BCLR PortM, RS ; SEND A COMMAND TO LCD

LDAA #$0E ; LCD DISPLAY ON, CURSOR BLINKING

JSR LCD\_INPUT ; PRINT COMMAND TO LCD

RTS ; RETURN FROM SUBROUTINE

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

DIRECTIONS: BCLR PortM, RS ; SEND A COMMAND TO LCD

LDAA #$01 ; CLEAR SCREEN COMMAND

JSR LCD\_INPUT ; SEND TO LCD

LDAA #$02 ; RETURN TO HOME COMMAND

JSR LCD\_INPUT ; SEND COMMAND

BSET PortM, RS ; LET'S PRINT TO LCD

LDX #STRING6 ; IF WE'RE IN PAUSE MODE PRINT 'PAUSED'

JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB

BCLR PortM, RS ; SENT A COMMAND TO LCD

LDAA #$C0 ; GO TO SECOND LINE TO PRINT

JSR LCD\_INPUT ; SEND COMMAND

LDX #STRING7 ; IF WE'RE COUTING DOWN PRINT 'DOWN'

JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB TO PPRINT

JSR delay2 ; DELAY A BIT

JSR delay2 ; DELAY A BIT

BCLR PortM, RS ; SEND A COMMAND TO LCD

LDAA #$01 ; CLEAR SCREEN COMMAND

JSR LCD\_INPUT ; SEND TO LCD

LDAA #$02 ; RETURN TO HOME COMMAND

JSR LCD\_INPUT ; SEND COMMAND

BSET PortM, RS ; LET'S PRINT TO LCD

LDX #STRING6 ; IF WE'RE IN PAUSE MODE PRINT 'PAUSED'

JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB

BCLR PortM, RS ; SENT A COMMAND TO LCD

LDAA #$C0 ; GO TO SECOND LINE TO PRINT

JSR LCD\_INPUT ; SEND COMMAND

LDX #STRING7 ; IF WE'RE COUTING DOWN PRINT 'DOWN'

JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB TO PPRINT

JSR delay2 ; DELAY A BIT

JSR delay2 ; DELAY A BIT

BCLR PortM, RS ; SEND A COMMAND TO LCD

LDAA #$01 ; CLEAR SCREEN COMMAND

JSR LCD\_INPUT ; SEND TO LCD

LDAA #$02 ; RETURN TO HOME COMMAND

JSR LCD\_INPUT ; SEND COMMAND

BSET PortM, RS ; LET'S PRINT TO LCD

LDX #STRING6 ; IF WE'RE IN PAUSE MODE PRINT 'PAUSED'

JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB

BCLR PortM, RS ; SENT A COMMAND TO LCD

LDAA #$C0 ; GO TO SECOND LINE TO PRINT

JSR LCD\_INPUT ; SEND COMMAND

LDX #STRING7 ; IF WE'RE COUTING DOWN PRINT 'DOWN'

JSR PRINT\_STRING ; GO TO PRINT\_STRING SUB TO PPRINT

JSR delay2 ; DELAY A BIT

JSR delay2 ; DELAY A BIT

RTS ; RETURN FROM SUBROUTINE ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; PURPOSE: TAKE THE VALUE IN THE X INDEX AND DISPLAY IT IN THE TENS LCD

UPDATE\_TENS\_DISPLAY:

JSR PREP\_VALS ; PREPARE OUR COUNT VALUE FOR OUTPUT

TFR X,B ; MOVE X INTO B

LDX #ASCII ; LOAD THE BEGINNING ADDRESS OF TABLE INTO X

ABX ; ADD B TO THE X INDEX

LDAA X ; LOAD THE ADDRESS OF INDEX X INTO ACCUM A

BSET PortM, RS ; PRINT CHARACTER TO LCD

JSR LCD\_INPUT ; SEND CHARACTER TO LCD

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: TAKE THE VALUE IN THE Y INDEX AND DISPLAY IT IN THE ONES LCD

UPDATE\_ONES\_DISPLAY:

JSR PREP\_VALS ; PREPARE OUR COUNT VALUE FOR OUTPUT

TFR Y,B ; MOVE Y INTO B SO WE CAN USE Y

LDY #ASCII ; LOAD THE BEGINNING ADDRESS OF TABLE INTO Y

ABY ; ADD B TO THE Y INDEX

LDAA Y ; LOAD THE ADDRESS OF INDEX Y INTO ACCUM A

BSET PortM, RS ; PRINT CHARACTER TO LCD

JSR LCD\_INPUT ; SEND CHARACTER TO LCD

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE INCREMENT THE VALUE OF COUNT\_VAL UNTIL WE REACH 99, THEN RESET TO 00

COUNT\_UP:

LDY COUNT\_VAL ; LOAD THE COUNT VALUE INTO Y

INY ; INCREMENT THE NUMBER BY 1

CPY #100 ; COMPARE IT TO 100

BEQ U\_RESET ; IF WE'RE ATT 100, THEN RESET THE NUMBER

STY COUNT\_VAL ; STORE THE COUNT VALUE BACK INTO Y

RTS ; RETURN FROM SUBROUTINE

U\_RESET LDY 0 ; RESET Y TO 0

STY COUNT\_VAL ; STORE THE COUNT VALUE BACK INTO Y

RTS ; RETURN FROM SUBROUTINE

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

; PURPOSE: DECREMENT THE VALUE OF COUNT\_VAL UNTIL WE REACH 0, THEN RESET TO 99

COUNT\_DOWN:

LDY COUNT\_VAL ; LOAD THE COUNT VALUE INTO Y

CPY #0 ; COMPARE IT TO 0

BEQ D\_RESET ; IF WE'RE AT ZERO, THEN RESET IT

DEY ; DECREMENT THE NUMBER BY Y

STY COUNT\_VAL ; STORE THE COUNT VALUE BACK INTO Y

RTS ; RETURN FROM SUBROUTINE

D\_RESET LDY 99 ; RESET Y TO 0

STY COUNT\_VAL ; STORE THE COUNT VALUE BACK INTO Y

RTS ; RETURN FROM SUBROUTINE

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

BLINK: LDY #0 ; SET Y TO ZERO

BLINKING PSHY ; PUSH Y ONTO THE STACK

BCLR PortM, RS ; SEND A COMMAND TO LCD

LDAA #$08 ; TURN LCD OFF COMMAND

JSR LCD\_INPUT ; SEND COMMAND TO LCD

BRCLR TMR\_FLAG,$01,\* ; WAIT A SECOND HERE

MOVB #$00,TMR\_FLAG ; CLEAR THE TIMER FLAG

LDAA #$0C ; DISPLAY ON COMMAND

JSR LCD\_INPUT ; SEND COMMAND TO LCD

BRCLR TMR\_FLAG,$01,\* ; WAIT A SECOND HERE

MOVB #$00,TMR\_FLAG ; CLEAR THE TIMER FLAG

PULY ; PULL Y OFF STACK

INY ; INCREMENT Y

CPY #3 ; SEE IF WE'VE BLINKED 3 TIMES

BNE BLINKING ; IF NOT THEN BLINK AGAIN!

RTS ; RETURN FROM SUBROUTINE

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

delay LDY #8000 ; COMMAND DELAY ROUTINE. WAY TO LONG. OVERKILL!

A2: DEY ; BUT WE DO NEED TO WAIT FOR THE LCD CONTROLLER

BNE A2 ; TO DO IT'S THING. HOW MUCH TIME?

RTS ; RETURN FROM SUBROUTINE

delay2 LDY #$F000 ; LONG DELAY ROUTINE. ADJUST AS NEEDED.

PSHA ; SAVE ACMA

A3: LDAA #$8F ; LONG DELAY LOAD ACMA WITH 8F (NESTED LOOP)

AB: DECA ; DECREMENT A

BNE AB ; BRANCH TO AB IF NOT EQUAL

DEY ; DECREMENT Y

BNE A3 ; BRANCH TO A3 IF NOT EQUAL

PULA ; GET ACMA BACK

RTS ; RETURN FROM SUBROUTINE

delay3 LDAA #$0F ; LOAD 15 (F) INTO ACMA

AA6: LDY #$FFFF ; LOAD Y WITH FFFF (Blink Delay routine.)

A6: DEY ; DECREMENT Y

BNE A6 ; BRANCH TO A6 IF NOT EQUAL

DECA ; DECREMENT A

BNE AA6 ; BRANCH TO AA6 IF NOT EQUAL

RTS ; RETURN FROM SUBROUTINE

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

; TC0 INTERRUPT SUBROUTINE

ISR\_TC0: LDD TC0 ; INTERRUPT READS THE FLAG SO THIS WRITE CLEARS THE FLAG

ADDD #3750 ; ADD THE EQUIVALENT .1 SECOND CNT TO REGISTER D

STD TC0 ; UPDATE TC0 MEMORY TO NEW VALUE

BRSET PAUSE,$01,PAUSED ; IF PAUSED DON'T UPDATE TIME\_COUNT!!

PSHA ; SAVE A ON THE STACK

LDAA TIME\_COUNT ; LOAD THE VALUE OF TIME\_COUNT INTO A

CMPA #100 ; IF TIME\_COUNT = 100 THEN WE HAVE 1 SECOND

BNE TMR\_UPDATE ; IF WE'RE NOT AT 100 YET, GOTO TMR\_UPDATE LINE

MOVB #$01,TMR\_FLAG ; TURN ON OUR TIMER FLAG

MOVB #$00,TIME\_COUNT ; RESET OUR TIMER COUNT BACK TO ZERO

PULA ; PUL A BACK OFF THE STACK

PAUSED RTI ; RETURN FROM THE INTERRUPT

TMR\_UPDATE ADDA #01 ; INCREMENT THE VALUE IN A

STAA TIME\_COUNT ; STORE A BACK INTO TIME\_COUNT

PULA ; PULL A BACK OFF THE STACK

RTI ; RETURN FROM THE INTERRUPT

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

; IRQ' INTERRUPT SUBROUTINE

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

JSR DRAW\_SCREEN

RTI ; RETURN FROM INTERRUPT

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

; XIRQ' INTERRUPT SUBROUTINE

ISR\_XIRQ: COM DIR\_FLAG ; TOGGLE THE DIRECTION FLAG

JSR DRAW\_SCREEN

RTI ; RETURN FROM INTERRUPT

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

ORG $FFF2 ; IRQ' VECTOR ADDRESS

FDB ISR\_IRQ ; ISR\_IRQ IS A LABEL FOR THE INTERRUPT SUBROUTINE

ORG $FFF4 ; XIRQ' VECTOR ADDRESS

FDB ISR\_XIRQ ; ISR\_XIRQ' IS A LABEL FOR THE INTERRUPT SUBROUTINE

ORG $FFEE ; VECTOR ADDRESS FOR TC0 INTERRUPT

FDB ISR\_TC0 ; ISR\_TIMER IS A LABEL FOR THE INTERRUPT SUBROUTINE

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

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

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

TABLE: DC.B $00, $01, $02, $03, $04 ; Define data table of mappings to each of the

DC.B $05, $06, $07, $08, $09 ; matrix keypad values.

DC.B $0A, $0B, $0C, $0D, $0E ; Memory locations correspond to their values

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

ASCII: DC.B $30, $31, $32, $33, $34 ; Define data table of mappings to each of the

DC.B $35, $36, $37, $38, $39 ; ascii values for the keypad

DC.B $41, $42, $43, $44, $45 ; Memory locations correspond to their values

DC.B $46 ; i.e. $5500 = 0, $5501 = 1, etc

ROW: DC.B $10, $20, $40, $80 ; PortT OUTPUT VALUES FOR MATRIX KEYPAD ROWS

COLUMN: DC.B $01, $02, $04, $08 ; PortM INPUT VALUES FOR MATRIX KEYPDA COLUMNS

KP\_VALUE: DC.B $01, $02, $03, $0A ; KEY VALUES FROM KEYPAD FOR ITERATING THROUGH

DC.B $04, $05, $06, $0B

DC.B $07, $08, $09, $0C

DC.B $00, $0F, $0E, $0D

STRING1 FCC "PAUSED " ; CREATE A STRING FOR PAUSED

DC.B $00

STRING2 FCC "RUN " ; CREATE A STRING WITH THE RUN

DC.B $00

STRING3 FCC " UP" ; CREATE A STRING WITH THE UP

DC.B $00

STRING4 FCC "DOWN" ; CREATE A STRING WITH THE DOWN

DC.B $00

STRING5 FCC " COUNT: " ; CREATE A STRING FOR THE TIME LINE

DC.B $00

; 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 (left-most black button).
   2. Operation
      1. After pressing the reset button, the system directions will be displayed on the screen.
      2. The system will next display “PAUSED UP” on the first line. On the second line will be the prompt, “COUNT: 00” where it will be awaiting input. The system defaults in the “Paused” and “Count Up” modes.
      3. Enter a number using the numeric portion of the keypad by pressing and releasing a button. Each time a new keypad number is entered, the system will take the value entered and move it into the ones place LED. If the ones place LED has a value in it, it will be moved to the tens place LED. A new number can be entered at any time during operation.
      4. If no values are input, the system will initialize to 00 and begin counting up after the Start/Pause button is pressed.
      5. The system operates using the “Run/Pause” button which is the far right black button. Each time the “Run/Pause” button is pressed; the system will either Run/Resume or Pause the countdown. This will be indicated on the display as well, you will either see “RUN” or “PAUSED” on the LCD screen.
      6. The direction of the count (up or down) may be changed at any time by pressing and releasing the middle black button. When the system is in count up mode, “UP” will be displayed in the screen and “DOWN” will be displayed when the system is counting down.
      7. When the shot clock reaches “00” the display will blink “00” 3 times and then reset to the initial screen indicating it is ready for input again.
      8. The system can be restarted/reinitialized at any time by simply pressing the reset button during operation.
      9. Button Layout:
         1. Left: Reset
         2. Middle: Change Direction
         3. Right: Start/Pause
   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 #8.
   2. What were the most difficult issues in realizing the system?
      1. The first difficult part of this experiment was determining the correct command codes for the LCD screen (although this wasn’t too hard as they were given) as well as the ASCII values for the characters on the screen.
      2. The other difficult task was transitioning the code from previous experiments, modifying it and then repurposing it for this experiment.
   3. Were you able to add extra features? If so, explain them.
      1. The only extra feature added for this experiment was reusable code for printing words. This allows for a simple subroutine call when one wants to print a word instead of printing it out each time.
   4. What would you have done differently if you were to do this project again?
      1. If I had more time, I probably would have added more information to the LCD screen and made it look better.
   5. What did you learn from working on this project?
      1. The two main things I learned from this project were how to operate the LCD screen properly and how to implement a “Backspace” key properly in the code.