Memo to: Randy Larimer
From: Matthew Handley
Date: February 27, 2014
Regarding: EE 465-01, Lab 2 – Liquid Crystal Display and Keypad

Summary:

This lab built on the previous one, by an LCD to the bus. The heartbeat LED was moved to one of the 8 LEDs driven by the DFFs on the bus. The goal was to listen for key presses on the keypad and display the key pressed on the LCD and 4of the DFF-driven LEDs.

Preliminary Solutions:

As before, the TPM module was used to toggle the heartbeat led. However, to improve the response speed of the keypad the scanning and interpretation of keypad data was moved to the main loop. Figure 1 of Appendix A shows the top level design for this lab. The S08 would power up and initialize the required modules (including the LCD), before continuously scaning the keypad and responding to changes. As shown in Figure 1, the TPM interrupt service routine would toggle the heartbeat LED and resetting the TPM overflow flag.

Setup:

To begin programming, the LCD was added to the bread board as well as several components required by the LCD. In order to provide the contrast control required by the LCD, a negative voltage supply and potentiometer were added. Because the LCD's Enable line latches data from the bus on a falling edge, a logic inverter was placed between the 74AHC138 decoder and LCD. The upper 4-bits of the LCD's data bus were tied to the existing data bus from the previous lab and the remaining 4 data lines were left floating. The RS and R/W were tied directly to PTA1 and PTA0 on the S08.

Solution:

The bus_read, bus_write, scan_keypad, led_write, and SUB_delay subroutines from previous labs were added first. Then, three new subroutines were added lcd_write, lcd_init, and lcd_char. The lcd_init and lcd_write subroutines were heavily based on the 4-bit example code given Motorola AN1745. Flow charts for the lcd_init and lcd_char subroutines are shown in Figure 4 of Appendix A. The lcd_char subroutine was developed to simplify the process of writing a character to the LCD and keep track of the LCD's cursor. Whenever lcd_char is called and the cursor is off of the LCD, the LCD is cleared before the new character is written.

In order to interpret the data from the keypad and respond accordingly, the interpret_keypad subroutine was written. The flow chart for this subroutine is shown in Figure 3 of Appendix A. Additionally, the scan_keypad subroutine had to be modified to keep track of current key press data as well as the last set of key press data. The modified flow chart for this subroutine is shown in Figure 2 of Appendix A. The interpret_keypad subroutine checks to see if a key changed from not pressed to pressed. If a key was pressed, it writes the corresponding character to the LCD, and loads the corresponding binary code into 4 bits of led_data.

Summary Comments:

While this lab was fairly straightforward, there were several issues that I ran into. The most time consuming part of this lab was initializing the LCD, however by closely following the AN1745 app note, eventually the LCD was initializing properly. Another interesting problem occurred when I tried writing the bus from both the main loop and TPM ISR. It seems that if the main loop is in the middle of writing to the bus when the TPM ISR occurs and starts writing to the bus, the device which the main loop was writing to will have odd data written to it. The solution was to modify the led_data variable in the ISR, but only write to the bus within the main loop. This also allows both the TPM ISR to toggle the hearbeat while the interpret_keypad modifies the 4 bits of key data in led_data.

Appendix A – Figures

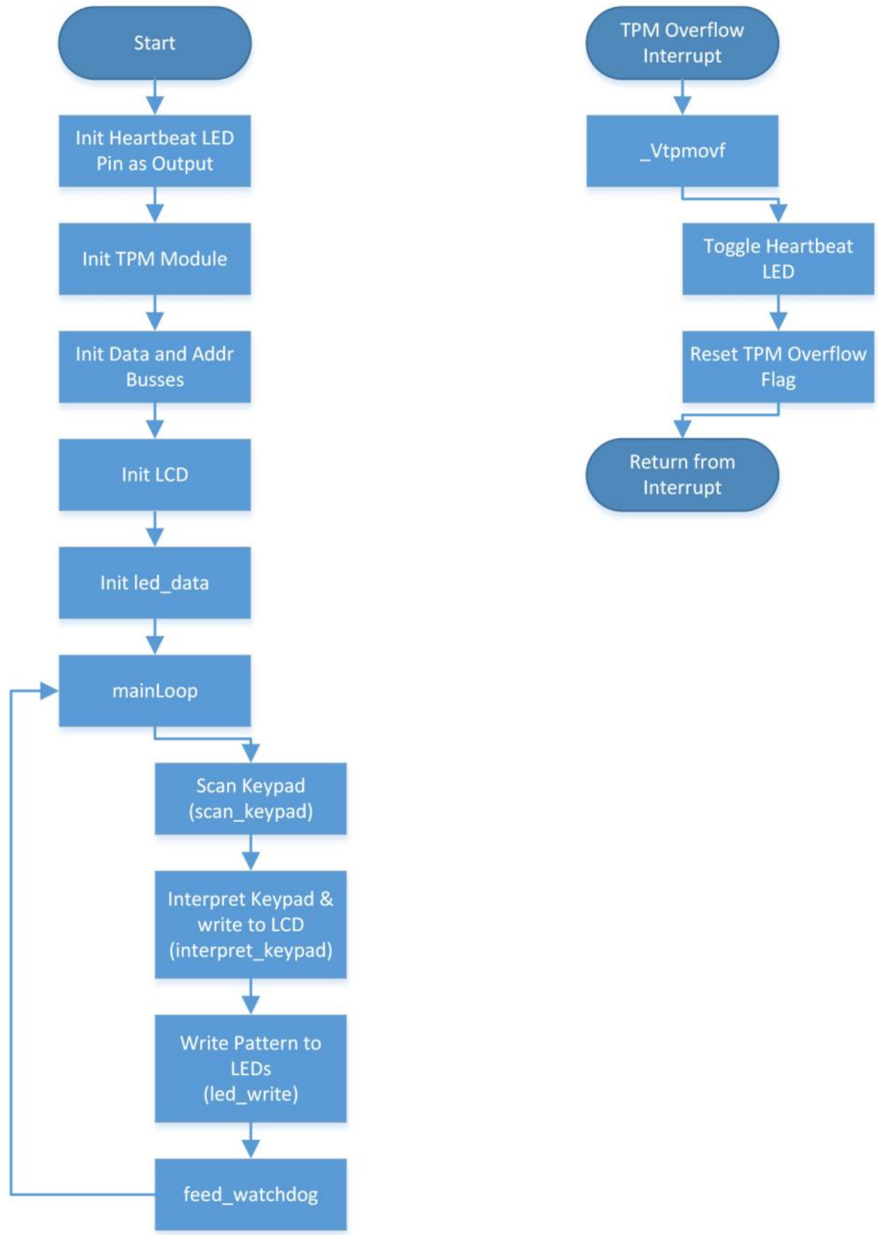

Figure 1: Top Level Flow Chart

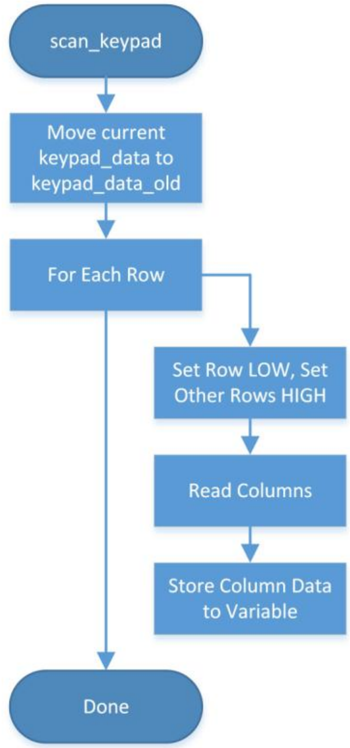

Figure 2: scan_keypad Subroutine Flow Chart

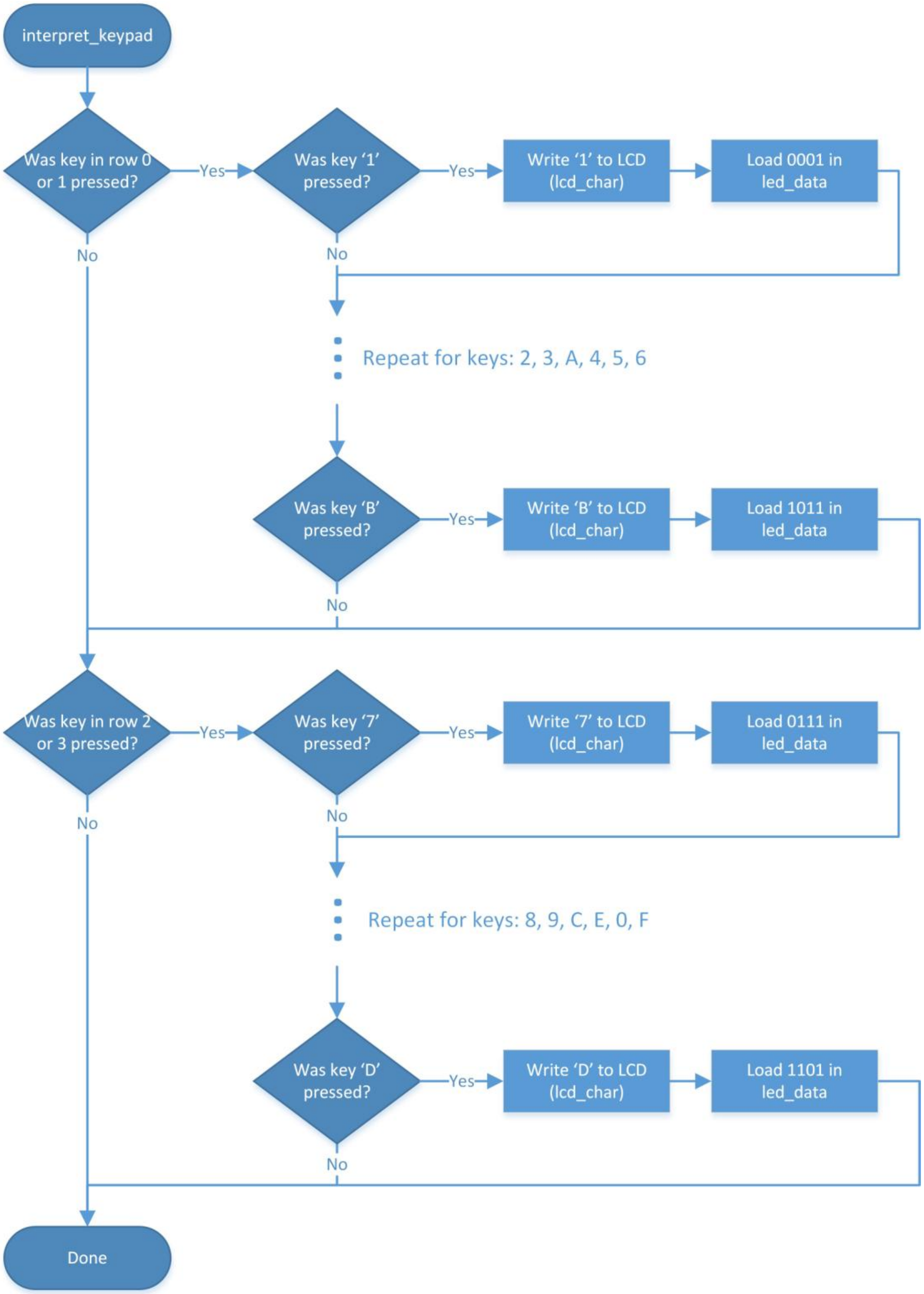

Figure 3: interpret_keypad Subroutine Flow Chart

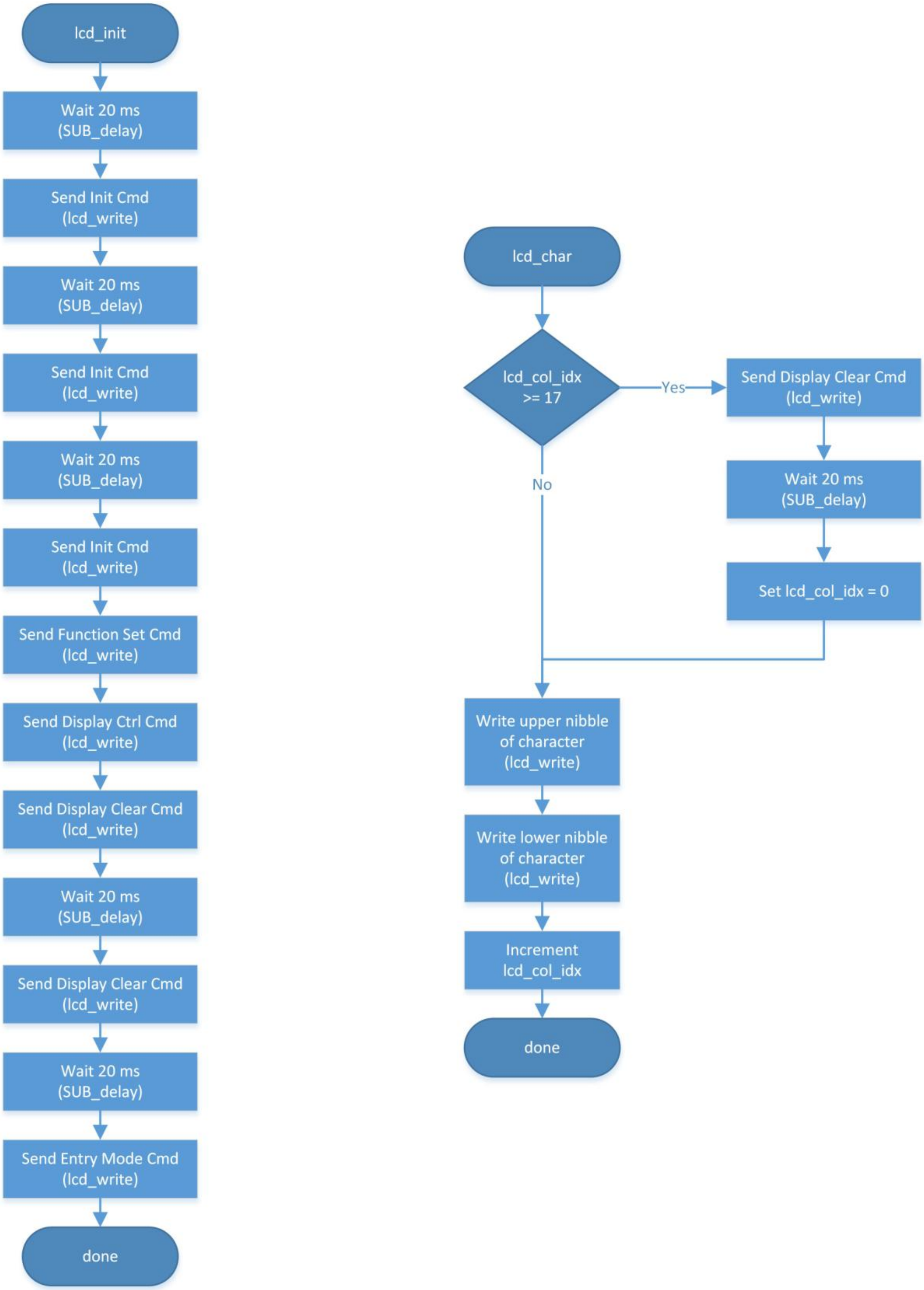

Figure 4: lcd_init and lcd_char Subroutine Flow Chart

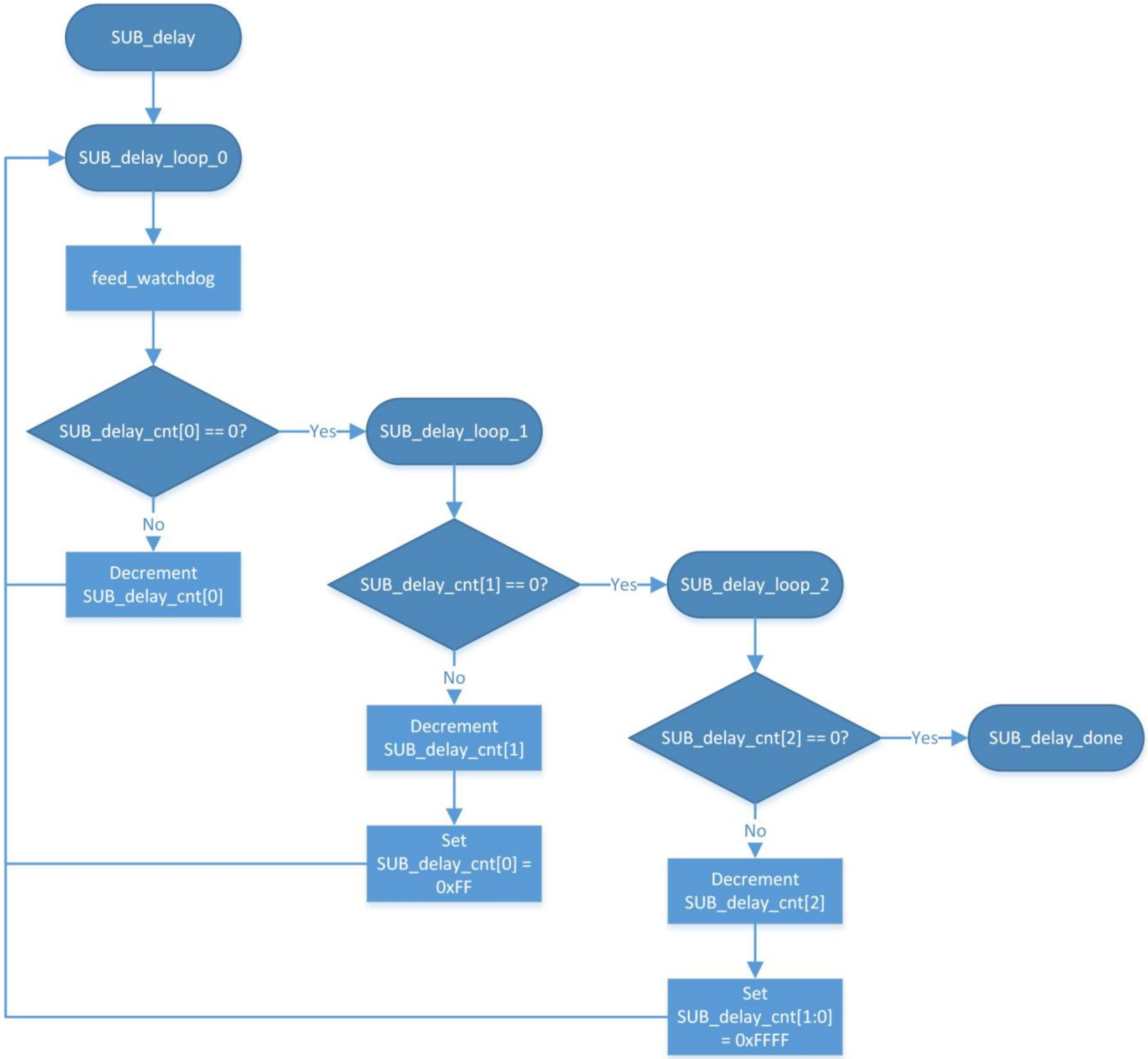

Figure 5: SUB_delay Subroutine Flow Chart

Appendix B – Source Code

```
;*************************************************************
;* Program Name: Lab#02 - LCD and Keypad
;* Author Names: Matthew Handley
;* Date: 2014-02-27
;* Description: Takes input from the keypad and displays it on
;*                      the first line of the LCD. When the line is
;*                      full, it is cleared before writing the next
;*                      character.
;*
;*************************************************************

mDataBus            EQU    $F0          ; Mask for the data bus pins on PortB
mAddrBus            EQU    $0F          ; Mask for the address bus pins on PortB


; Include derivative-specific definitions
            INCLUDE 'derivative.inc'


; export symbols
            XDEF _Startup, main, _Vtpmovf, bus_write, bus_read, scan_keypad,
led_write, lcd_init, SUB_delay
            ; we export both '_Startup' and 'main' as symbols. Either can
            ; be referenced in the linker .prm file or from C/C++ later on


            XREF __SEG_END_SSTACK   ; symbol defined by the linker for the end of the
stack


; variable/data section
MY_ZEROPAGE: SECTION  SHORT

                  bus_addr:          DS.B  1      ; only use lower 3 bits
                  bus_data:          DS.B  1      ; only use lower 4 bits

                  lcd_data:          DS.B  1      ; lower 4 bits = LCD data lines, bit 6
= RS, bit 5 = RW
                  lcd_char_data:     DS.B  1      ; used by lcd_char subroutine to store
a character
                  lcd_col_idx:       DS.B  1      ; index of the column of the LCD that
the cursor is currently in

                  keypad_data_0:     DS.B  1      ; bit flags representing what keys are
pressed on the 4x4 keypad
                  keypad_data_1:     DS.B  1

                  keypad_data_0_old:       DS.B  1      ; bit flags representing which
keys were pressed on the keypad, the last time it was scanned
                  keypad_data_1_old:       DS.B  1

                  keypad_data_cmp:   DS.B  1      ; tempory holder for keypad data
comparison in interpret_keypad


                  led_data:          DS.B  1      ; 8 bit value for the 8 LEDs

                  ; counter for SUB_delay subroutine
                  SUB_delay_cnt:           DS.B  3

MY_CONST: SECTION
```

```
; Constant Values and Tables Section


; code section
MyCode:     SECTION
main:
_Startup:
            LDHX   #__SEG_END_SSTACK ; initialize the stack pointer
            TXS

                  ;*** init TPM module - for heartbeat LED ***
                  ; TPMMODH:L Registers
                  LDA    #$00
                  STA          TPMMODH
                  LDA    #$00
                  STA          TPMMODL
                  ; TPMSC Register
                  LDA    #$4E                  ; TOIE clear, CLKS: Bus clock,
Prescale: 128
                  STA          TPMSC

                  ;*** init Data & Address Busses ***
                  LDA          mAddrBus        ; Set Address Bus pins as output by
default, leave data as input
                  STA          PTBDD
                  LDA          $00             ; Leave all of PortB as input at start
                  STA          PTBD

                  ;*** init LCD and RS, RW pins ***
                  JSR          lcd_init

                  LDA          #$00
                  STA          lcd_col_idx

                  ;*** init led_data variable ***
                  LDA          #$00
                  STA          led_data

                  CLI                          ; enable interrupts

mainLoop:
            ; read keypad
            JSR          scan_keypad

            ; interpret keypad data
            JSR          interpret_keypad

            ; Display led_data on leds
                  JSR          led_write

            feed_watchdog
            BRA          mainLoop
```

```
;**************************************************************
;* Subroutine Name: _Vtpmovf
;* Description: Interrupt service routine for the TPM overflow
;*                      interrupt. Toggles the heartbeat LED (PortA[0])
;*                      and resets TPM overflow flag.
;* Registers Modified: None
;* Entry Variables: None
;* Exit Variables: None
;**************************************************************
_Vtpmovf:
                  ; Toggle Heartbeat LED
                  LDA         led_data                ; load current LED pattern
                  EOR         #$80                    ; toggle bit 7
                  STA         led_data                ; Store pattern to var

                  ; clear TPM ch0 flag
                  LDA         TPMSC                   ; read register
                  AND         #$4E                    ; clear CH0F bit, but leav
others alone
                  STA         TPMSC                   ; write back register

;*** done ***

                  ;Return from Interrupt
                  RTI

;**************************************************************
```

```
;***********************************************************
;* Subroutine Name: led_write
;* Description: Writes the 8 bits of led_data two the 8 LEDs
;*                      on the DFFs at address 0 and 1 on the bus.
;*
;* Registers Modified: None
;* Entry Variables: led_data
;* Exit Variables: None
;***********************************************************
led_write:
                  ; preserve accumulator A
                  PSHA

;*** write lower nibble LEDs ***
                  ; set the address
            LDA   #$00
            STA         bus_addr

            ; set the data
            LDA   led_data
            AND         #$0F
            STA         bus_data

            ; write the data
            JSR         bus_write

;*** write upper nibble LEDs ***
                  ; set the address
            LDA   #$01
            STA         bus_addr

            ; set the data
            LDA   led_data
            NSA
            AND         #$0F
            STA         bus_data

            ; write the data
            JSR         bus_write

;*** done ***
                  ; restore accumulator A
                  PULA
                  RTS

;***********************************************************
```

```
;***********************************************************
;* Subroutine Name: bus_read
;* Description: Reads data from the device whose address is
;*                     the lower 3 bits of bus_addr, and store the
;*                     data to the lower 4 bits of bus_data.
;*
;* Registers Modified: None
;* Entry Variables: bus_addr
;* Exit Variables: bus_data
;***********************************************************
bus_read:
                  ; preserve accumulator A
                  PSHA

                  ; make address bus output, data bus an input
            LDA         #mAddrBus
            STA         PTBDD

                  ; pull the address low
            LDA   bus_addr                  ; load address
            AND         #$07                      ; mask off the lower 3 bits to be
sure, will leave G2A low
            STA         PTBD                      ; write data to address bus, and clear
data bus

            ; read data from the bus
            LDA         PTBD
            NSA                                   ; shift data down to the lower 4 bits
            AND         #$0F                      ; mask off the lower 4 bits to be sure
            STA         bus_data                  ;

                  ; pull the address high
            LDA   #$08                      ; G2A_not high
            STA         PTBD                      ; write, clears address bus

                  ; restore accumulator A
                  PULA

                  ; return from subroutine bus_read
                  RTS

;***********************************************************
```

```
;**************************************************************
;* Subroutine Name: bus_write
;* Description: Writes the lower 4 bits of bus_data to the
;*                      device on whose address is the lower 3 bits
;*                      of bus_addr.
;* Registers Modified: None
;* Entry Variables: bus_addr, bus_data
;* Exit Variables: None
;**************************************************************
bus_write:
                  ; preserve accumulator A
                  PSHA

                  ; make data and address busses outputs
            LDA         #$FF
            STA         PTBDD

            ; prep data for the bus
            LDA         bus_data
            NSA                                 ; swap the lower 4 bits to be the
upper 4 bits
            AND         #$F0                    ; mask off the upper 4 bits to be sure

                  ; prep the addr, G2A_not low, Yx goes low
            ORA         bus_addr                ; add in the address
            STA         PTBD                    ; write data and address bus, with
G2A_not low

            ORA         #$08                    ; leave data and address, set G2A_not
high - Yx goes high
            STA         PTBD

                  ; restore accumulator A
                  PULA

                  ; return from subroutine bus_write
                  RTS

;**************************************************************
```

```
;**************************************************************
;* Subroutine Name: scan_keypad
;* Description: Scans the greyhill 4x4 keypad, and saves the
;*                      result to variable.
;*                      Note that this method will overwrite values in
;*                      the bus_addr and bus_data variables.
;*
;* Registers Modified: None
;* Entry Variables: None
;* Exit Variables: keypad_data_0, keypad_data_1
;**************************************************************
scan_keypad:
                  ; preserve registers
                  PSHA

;*** save old value of keypad_data, before we overwrite it

                  LDA          keypad_data_0
                  STA          keypad_data_0_old
                  LDA          keypad_data_1
                  STA          keypad_data_1_old

;*** scan row 0 ***

            ;* set row 0 to low, other rows to high *

            ; set address of keypad driver DFF
            LDA    #$02
            STA          bus_addr

            ; set the data
            LDA    #%00001110
            STA          bus_data

            ; write the data
            JSR          bus_write

            ;* read data from row *

                  ; set the address
            LDA    #$03
            STA          bus_addr

            ; read the data
            JSR          bus_read

            ;* save row data to variable *

                  LDA          bus_data          ; load in data nibble
                  COMA                           ; compliment bits, so 1=button press
                  AND          #$0F              ; mask off the lower 4 bits
                  STA          keypad_data_0     ; store to vairable


;*** scan row 1 ***

            ;* set row 1 to low, other rows to high *

            ; set address of keypad driver DFF
            LDA    #$02
            STA          bus_addr
```

```
            ; set the data
            LDA   #%00001101
            STA         bus_data

            ; write the data
            JSR         bus_write

            ;* read data from row *

                  ; set the address
            LDA   #$03
            STA         bus_addr

            ; read the data
            JSR         bus_read

            ;* save row data to variable *

                  LDA         bus_data          ; load in data nibble
                  COMA                          ; compliment bits, so 1=button press
                  NSA                                 ; swap our data to the upper
nibble
                  AND         #$F0              ; mask off the data
                  ORA         keypad_data_0     ; add the lower 4 bits in
                  STA         keypad_data_0     ; store to vairable


;*** scan row 2 ***

            ;* set row 2 to low, other rows to high *

            ; set address of keypad driver DFF
            LDA   #$02
            STA         bus_addr

            ; set the data
            LDA   #%00001011
            STA         bus_data

            ; write the data
            JSR         bus_write

            ;* read data from row *

                  ; set the address
            LDA   #$03
            STA         bus_addr

            ; read the data
            JSR         bus_read

            ;* save row data to variable *

                  LDA         bus_data          ; load in data nibble
                  COMA                          ; compliment bits, so 1=button press
                  AND         #$0F              ; mask off the lower 4 bits
                  STA         keypad_data_1     ; store to vairable


;*** scan row 3 ***

            ;* set row 3 to low, other rows to high *
```

```
            ; set address of keypad driver DFF
            LDA    #$02
            STA          bus_addr

            ; set the data
            LDA    #%00000111
            STA          bus_data

            ; write the data
            JSR          bus_write

            ;* read data from row *

                  ; set the address
            LDA    #$03
            STA          bus_addr

            ; read the data
            JSR          bus_read

            ;* save row data to variable *

                  LDA          bus_data            ; load in data nibble
                  COMA                             ; compliment bits, so 1=button press
                  NSA                                    ; swap our data to the upper
nibble
                  AND          #$F0                ; mask off the data
                  ORA          keypad_data_1       ; add the lower 4 bits in
                  STA          keypad_data_1       ; store to vairable


;*** done ***

                  ; restore registers
                  PULA

                  ; return from subroutine scan_keypad
                  RTS

;***************************************************************
```

```
;*************************************************************
;* Subroutine Name: interpret_keypad
;* Description: Interpres the data from the keypad and writes
;*                      to the LCD and led_data based on keypad input.
;*
;* Registers Modified: None
;* Entry Variables: None
;* Exit Variables: None
;*************************************************************
interpret_keypad:

                  ; preserve registers
                  PSHA

;*** was a key pressed in the first 2 rows ? ***

                  LDA         keypad_data_0_old
                  COMA
                  AND         keypad_data_0
                  CBEQA #$00, interpret_keypad_lower_rows_jump

                  ; key was pressed
                  STA         keypad_data_cmp

interpret_keypad_1:

                  ; was '1' pressed ?
                  LDA         keypad_data_cmp
                  AND         #%11111110
                  BNE         interpret_keypad_2

                  ; write a '1' to the LCD
                  LDA         #'1'
                  JSR         lcd_char

                  ; put 0x1 on LEDs
                  LDA         led_data
                  AND         #$F0
                  ORA         #$01
                  STA         led_data

interpret_keypad_2:

                  ; was '2' pressed ?
                  LDA         keypad_data_cmp
                  AND         #%11111101
                  BNE         interpret_keypad_3

                  ; write a '2' to the LCD
                  LDA         #'2'
                  JSR         lcd_char

                  ; put 0x1 on LEDs
                  LDA         led_data
                  AND         #$F0
                  ORA         #$02
                  STA         led_data

interpret_keypad_3:

                  ; was '3' pressed ?
                  LDA         keypad_data_cmp
```

```
                AND         #%11111011
                BNE         interpret_keypad_A

                ; write a '3' to the LCD
                LDA         #'3'
                JSR         lcd_char

                ; put 0x3 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$03
                STA         led_data


interpret_keypad_A:

                ; was 'A' pressed ?
                LDA         keypad_data_cmp
                AND         #%11110111
                BNE         interpret_keypad_4

                ; write a 'A' to the LCD
                LDA         #'A'
                JSR         lcd_char

                ; put 0xA on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$0A
                STA         led_data

                BRA         interpret_keypad_4
interpret_keypad_lower_rows_jump:
                BRA         interpret_keypad_lower_rows


interpret_keypad_4:

                ; was '4' pressed ?
                LDA         keypad_data_cmp
                AND         #%11101111
                BNE         interpret_keypad_5

                ; write a '4' to the LCD
                LDA         #'4'
                JSR         lcd_char

                ; put 0x4 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$04
                STA         led_data


interpret_keypad_5:

                ; was '5' pressed ?
                LDA         keypad_data_cmp
                AND         #%11011111
                BNE         interpret_keypad_6

                ; write a '5' to the LCD
```

```
                LDA         #'5'
                JSR         lcd_char

                ; put 0x5 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$05
                STA         led_data

interpret_keypad_6:

                ; was '6' pressed ?
                LDA         keypad_data_cmp
                AND         #%10111111
                BNE         interpret_keypad_B

                ; write a '6' to the LCD
                LDA         #'6'
                JSR         lcd_char

                ; put 0x6 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$06
                STA         led_data

interpret_keypad_B:

                ; was 'B' pressed ?
                LDA         keypad_data_cmp
                AND         #%01111111
                BNE         interpret_keypad_lower_rows

                ; write a 'B' to the LCD
                LDA         #'B'
                JSR         lcd_char

                ; put 0xB on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$0B
                STA         led_data


interpret_keypad_lower_rows:
;*** was a key pressed in the second 2 rows ? ***

                LDA         keypad_data_1_old
                COMA
                AND         keypad_data_1
                CBEQA #$00, interpret_keypad_done_jump

                ; key was pressed
                STA         keypad_data_cmp

interpret_keypad_7:

                ; was '7' pressed ?
                LDA         keypad_data_cmp
                AND         #%11111110
                BNE         interpret_keypad_8
```

```
                ; write a '7' to the LCD
                LDA         #'7'
                JSR         lcd_char

                ; put 0x7 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$07
                STA         led_data

interpret_keypad_8:

                ; was '8' pressed ?
                LDA         keypad_data_cmp
                AND         #%11111101
                BNE         interpret_keypad_9

                ; write a '8' to the LCD
                LDA         #'8'
                JSR         lcd_char

                ; put 0x8 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$08
                STA         led_data

interpret_keypad_9:

                ; was '9' pressed ?
                LDA         keypad_data_cmp
                AND         #%11111011
                BNE         interpret_keypad_C

                ; write a '9' to the LCD
                LDA         #'9'
                JSR         lcd_char

                ; put 0x9 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$09
                STA         led_data

interpret_keypad_C:

                ; was 'C' pressed ?
                LDA         keypad_data_cmp
                AND         #%11110111
                BNE         interpret_keypad_E

                ; write a 'C' to the LCD
                LDA         #'C'
                JSR         lcd_char

                ; put 0xC on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$0C
                STA         led_data
```

```
                BRA   interpret_keypad_E
interpret_keypad_done_jump:
                BRA         interpret_keypad_done


interpret_keypad_E:

                ; was 'E'/'*' pressed ?
                LDA         keypad_data_cmp
                AND         #%11101111
                BNE         interpret_keypad_0

                ; write a 'E' to the LCD
                LDA         #'E'
                JSR         lcd_char

                ; put 0xE on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$0E
                STA         led_data


interpret_keypad_0:

                ; was '0' pressed ?
                LDA         keypad_data_cmp
                AND         #%11011111
                BNE         interpret_keypad_F

                ; write a '0' to the LCD
                LDA         #'0'
                JSR         lcd_char

                ; put 0x0 on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$00
                STA         led_data


interpret_keypad_F:

                ; was 'F'/'#' pressed ?
                LDA         keypad_data_cmp
                AND         #%10111111
                BNE         interpret_keypad_D

                ; write a 'F' to the LCD
                LDA         #'F'
                JSR         lcd_char

                ; put 0xF on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$0F
                STA         led_data


interpret_keypad_D:

                ; was 'D' pressed ?
```

```
                LDA         keypad_data_cmp
                AND         #%01111111
                BNE         interpret_keypad_done

                ; write a 'D' to the LCD
                LDA         #'D'
                JSR         lcd_char

                ; put 0xD on LEDs
                LDA         led_data
                AND         #$F0
                ORA         #$0D
                STA         led_data


interpret_keypad_done:
;*** done ***

                ; restore registers
                PULA

                ; return from subroutine scan_keypad
                RTS

;***************************************************************
```

```
;**************************************************************
;* Subroutine Name: lcd_init
;* Description: Initilizes the LCD.
;*
;* Registers Modified: None
;* Entry Variables: None
;* Exit Variables: None
;**************************************************************
lcd_init:
                  ; preserve registers
                  PSHA

;*** init RS and RW pins as outputs
                  LDA         PTADD
                  ORA         #$03
                  STA         PTADD

;*** wait for 20 ms
                  ; load address of SUB_delay_cnt
                  LDHX #SUB_delay_cnt

                  ; configure loop delays: 0x001388 = 20 ms
                  LDA         #$00
                  STA         2,X
                  LDA         #$13
                  STA         1,X
                  LDA         #$88
                  STA         0,X

                  ; jump to the delay loop
                  JSR         SUB_delay

;*** Send Init Command

                  LDA         #$03
                  JSR         lcd_write

;*** Wait for 20 ms
                  ; load address of SUB_delay_cnt
                  LDHX #SUB_delay_cnt

                  ; configure loop delays: 0x001388 = 20 ms
                  LDA         #$00
                  STA         2,X
                  LDA         #$13
                  STA         1,X
                  LDA         #$88
                  STA         0,X

                  ; jump to the delay loop
                  JSR         SUB_delay

;*** Send Init command

                  LDA         #$03
                  JSR         lcd_write

;*** Wait for 20 ms
                  ; load address of SUB_delay_cnt
                  LDHX #SUB_delay_cnt

                  ; configure loop delays: 0x001388 = 20 ms
```

```
                LDA         #$00
                STA         2,X
                LDA         #$13
                STA         1,X
                LDA         #$88
                STA         0,X

                ; jump to the delay loop
                JSR         SUB_delay

;*** Send Init command

                LDA         #$03
                JSR         lcd_write

;*** Send Function set command

                LDA         #$02
                JSR         lcd_write

                LDA         #$02
                JSR         lcd_write

                LDA         #$08
                JSR         lcd_write ; goes blank here

;*** Send display ctrl command

                LDA         #$00
                JSR         lcd_write

                LDA         #$0F
                JSR         lcd_write

;*** Send display clear command

                LDA         #$00
                JSR         lcd_write

;*** Wait for 20 ms
                ; load address of SUB_delay_cnt
                LDHX #SUB_delay_cnt

                ; configure loop delays: 0x001388 = 20 ms
                LDA         #$00
                STA         2,X
                LDA         #$13
                STA         1,X
                LDA         #$88
                STA         0,X

                ; jump to the delay loop
                JSR         SUB_delay

;*** Send display clear command

                LDA         #$01
                JSR         lcd_write

;*** Wait for 20 ms
                ; load address of SUB_delay_cnt
                LDHX #SUB_delay_cnt
```

```
                ; configure loop delays: 0x001388 = 20 ms
                LDA         #$00
                STA         2,X
                LDA         #$13
                STA         1,X
                LDA         #$88
                STA         0,X

                ; jump to the delay loop
                JSR         SUB_delay

;*** Send entry mode command

                LDA         #$00
                JSR         lcd_write

                LDA         #$06
                JSR         lcd_write

;*** done ***

                ; restore registers
                PULA

                ; return from subroutine lcd_init
                RTS

;***************************************************************
```

```
;************************************************************
;* Subroutine Name: lcd_write
;* Description: Sends data to the LCD.
;*
;* Registers Modified: Accu A
;* Entry Variables: Accu A
;* Exit Variables:
;************************************************************
lcd_write:
                  ; store param to var for latter
                  STA          lcd_data

                  ; clear RS and RW pins on PTAD
                  LDA    PTAD
                  AND          #$FC
                  STA          PTAD

                  ; put RS an RW on PTAD
                  LDA          lcd_data
                  NSA
                  AND    #$03
                  ORA          PTAD
                  STA          PTAD

                  ; prep bus data
                  LDA          lcd_data
                  AND          #$0F
                  STA          bus_data
                  ; prep bus addr
                  LDA          #$04
                  STA          bus_addr
                  ; write data to bus (and clock the addr)
                  JSR          bus_write


;*** Wait for 40 us
                  ; load address of SUB_delay_cnt
                  LDHX #SUB_delay_cnt

                  ; configure loop delays: 0x00000A = 40 us
                  LDA          #$00
                  STA          2,X
                  LDA          #$00
                  STA          1,X
                  LDA          #$0A
                  STA          0,X

                  ; jump to the delay loop
                  JSR          SUB_delay

                  ; done
                  RTS

;************************************************************
```

```
;***************************************************************
;* Subroutine Name: lcd_char
;* Description: Writes a character to the LCD.
;*                      If lcd_col_idx is off of the first line, the
;*                      LCD will be cleared and the new char will be
;*                      written to the first column of row 0
;*
;* Registers Modified: Accu A
;* Entry Variables: Accu A
;* Exit Variables:
;***************************************************************
lcd_char:

                  ; store input parameter
                  STA          lcd_char_data

                  ; lcd_col_idx < 17
                  LDA          lcd_col_idx
                  CMP          #$10
                  BNE          lcd_char_write_Char

                  ; lcd_col_idx >= 17, clear lcd

                  ; Send display clear command
                  LDA          #$00
                  JSR          lcd_write
                  LDA          #$01
                  JSR          lcd_write

                  ;*** Wait for 20 ms ***
                  LDHX #SUB_delay_cnt

                  ; configure loop delays: 0x001388 = 20 ms
                  LDA          #$00
                  STA          2,X
                  LDA          #$13
                  STA          1,X
                  LDA          #$88
                  STA          0,X

                  ; jump to the delay loop
                  JSR          SUB_delay

                  ;  reset lcd_col_idx
                  LDA          #$00
                  STA          lcd_col_idx

lcd_char_write_Char:

                  ; write upper nibble
                  LDA          lcd_char_data
                  NSA
                  AND          #$0F
                  ORA          #$20
                  JSR          lcd_write

                  ; write lower nibble
                  LDA          lcd_char_data
                  AND          #$0F
                  ORA          #$20
                  JSR          lcd_write
```

```
        ; increment lcd_col_idx
        LDA         lcd_col_idx
        INCA
        STA         lcd_col_idx

        ; done
        RTS

;*************************************************************
```

```
;***********************************************************************
;* Subroutine Name: SUB_delay
;* Description: Decrements SUB_delay_cnt until it reaches zero.
;*                      1 count in SUB_delay_cnt is approx 4.019 us
;*
;* Registers Modified: None.
;* Entry Variables: SUB_delay_cnt - 3 byte variable, determines length
;*                            of time the SUB_delay routine will take to execute.
;* Exit Variables: SUB_delay_cnt - will be zero at exit.
;***********************************************************************
SUB_delay:
                  ; save the existing values of registers
                  PSHH
                  PSHX
                  PSHA

                  ; load address of SUB_delay_cnt
                  LDHX #SUB_delay_cnt

SUB_delay_loop_0:

                  feed_watchdog

                  ; if byte[0] == 0
                  LDA   0, X
                  BEQ         SUB_delay_loop_1        ; jump to SUB_delay_outer_loop

                  ;else
                  DECA                                      ; decrement byte[0]
                  STA         0, X

                  ;repeat
                  BRA SUB_delay_loop_0

SUB_delay_loop_1:

                  ; if byte[1] == 0
                  LDA   1, X
                  BEQ         SUB_delay_loop_2        ; branch to done

                  ;else
                  DECA                                      ; decrement byte[1]
                  STA         1, X

                  LDA         #$FF                          ; reset byte[0]
                  STA         0,X

                  ;repeat
                  BRA SUB_delay_loop_0

SUB_delay_loop_2:

                  ; if byte[2] == 0
                  LDA   2, X
                  BEQ         SUB_delay_done                ; branch to done

                  ;else
                  DECA                                      ; decrement byte[2]
                  STA         2, X

                  LDA         #$FF                          ; reset byte[1]
                  STA         1, X
```

```
                LDA         #$FF                        ; reset byte[0]
                STA         0, X

                ;repeat
                BRA SUB_delay_loop_0

SUB_delay_done:

                ; restore registers to previous values
                PULA
                PULX
                PULH

                RTS
;*************************************************************
```