|  |
| --- |
| ; export symbols  XDEF Entry, mainLoop, IOSetup, delay, delayLoop, moveLEDsRight, moveLEDsLeft, playerGuessOver, playerGuessUnder, playerGuessRight, flashLEDs, label1, label2  ; we use export 'Entry' as symbol. This allows us to  ; reference 'Entry' either 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  ; include derivative specific macros  INCLUDE 'MC9S08GB60.inc'  ; variable/data section  TIMER\_STATUS EQU $30  TIMER\_COUNTER\_HIGH EQU $82  TIMER\_COUNTER\_LOW EQU $83  TIMER\_MODULUS\_HIGH EQU $33  TIMER\_MODULUS\_LOW EQU $34  LOCK\_COMBINATION EQU $1000  USER\_GUESS\_COMBINATION EQU $1020  ; code section  MyCode: SECTION  main:  Entry:  LDHX #\_\_SEG\_END\_SSTACK ; initialize the stack pointer  TXS  CLI ; enable interrupts    IOSetup:  CLRA ; special instruction to load a ZERO value into A  LDA #$FF ; enable pull-up (all bits have pull up enabled)  STA $43 ; configure PORT F data direction (all bits OUTPUT)  mainLoop:  LDA USER\_GUESS\_COMBINATION ;loads A with the user guess  CMPA LOCK\_COMBINATION ;compares user guess with lock combo  BEQ playerGuessRight ;if equal go to flash leds  BLO playerGuessUnder ;if less then go to move LED right  BRA playerGuessOver ;if more go to move LED left  BRA mainLoop  playerGuessRight:  JSR flashLEDs  BRA mainLoop    playerGuessUnder:    JSR moveLEDsRight  BRA mainLoop    playerGuessOver:  JSR moveLEDsLeft  BRA mainLoop  moveLEDsRight:  LDA #%1111 ; turn off the LEDS  STA $40 ; store the off values in A  ASR $40 ; bit shift A right |

Lab 6

# Purpose

The purpose of this lab is to become closer with assembly programming and developing for the 6808 development platform as well as the workflow involved in such.

# Objective

This lab gives you:

* More experience with the 6808 development board, environment, and lifecycle
* More practice doing assembly programming and using tools such as delays and sub routines

# Procedure

1. Open up the CodeWarrior IDE and copy the code from figure. In.
2. Then, using your delay code from last week, copy past code into your new codebase and run It on the 6808.
3. Using the delay function from our last lab, try to make all of the LEDs blink at once. Then, when you’ve got that working, move that into a new function called flashLEDs.
4. Now, try making a new function which goes and turns all of the LEDs on and then turns them off one at a time from left to right. When you’re done, call the function moveLEDsRight
5. Using the existing function from the last step, try reversing the direction and, when you’re done, put it into a function called moveLEDsLeft
6. Now that we’ve got our basic LED animations done, it’s time to move onto some real programming! Create two placeholders in ram for storing a user combination and a lock combination.

|  |
| --- |
| JSR delay ; delay and repeat  ASR $40 ;  JSR delay ;  ASR $40 ;  JSR delay ;  ASR $40 ;  JSR delay ;  CLRA ; clear A  RTS ; return to main loop to repeat      moveLEDsLeft:  LDA #%1111 ; turn off the LEDs  STA $40 ; store the off values in 0x40  ASL $40 ; bit shift A left  JSR delay ; delay and repeat  ASL $40 ;  JSR delay ;  ASL $40 ;  JSR delay ;  ASL $40 ;  JSR delay ;  CLRA ; clear A  RTS ; return to main loop and repeat      flashLEDs:  LDA #%1111 ; turn off the LEDs  STA $40 ; store the off values in 0x40  JSR delay ; delay  LDA #%0000 ; turn all the LEDs on  STA $40 ; repeat  JSR delay ;  LDA #%1111 ;  STA $40 ;  JSR delay ;  LDA #%0000 ;  STA $40 ;  JSR delay ;  LDA #%1111 ;  STA $40 ;  JSR delay ;  LDA #%0000 ;  STA $40 ;  JSR delay ;  LDA #%1111 ;  STA $40 ;  JSR delay ;  LDA #%0000 ;  STA $40 ;  JSR delay ;  CLRA ; clear A  RTS ; return to main loop and repeat    delay:  LDA #$17 ; get current timer status  STA TIMER\_STATUS ; write the result back  LDA #$01 ; set the high mod  STA TIMER\_MODULUS\_HIGH  LDA #$40 ; set the low mod  STA TIMER\_MODULUS\_LOW  JSR delayLoop  RTS      delayLoop:  LDA TIMER\_STATUS ; get the timer status into accumulator  AND #$80 ; check the high bit in accumulator  BEQ delayLoop ; if result is zero, then high bit not yet set, so continue waiting  RTS |

1. Then, create functions for if the guess is too high or too low. Use the LED animation functions we just created to make a nice effect.
2. Then, in the main loop, put in some conditional logic to decipher whether the user guess combination is lower or higher than the lock combination. Attach the output of this conditional logic with our newly created functions.
3. Get your teacher to pick your lock! (A tip for next time though is not to let her near your memory map!)