EGR 322 Microcontroller System Design

Lab 6 – Digital Clock Using PSoC Timer and LCD Tool Box

Prepared for: Dr. Foist

Christopher Parisi

College of Engineering
California Baptist University

03/21/12

Summary

This lab teaches students how to use the LED on the PSoC Evall board. Students will learn how to use the LCD User Module to create programs that show strings and integers on the screen. Students will also implement previous lab material by using the Timer to create a clock.

Design

Exercise 1:

I began by creating a project and writing the main program to display a string and an integer. I accomplished this by using the example code given and modifying it meet my needs. The program I wrote is shown below. The Program was built, downloaded, and testing. It worked correctly and was approved by the professor.

```
;-----------------------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------------------

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

area  bss(RAM)

      hour: BLK 1

      min:  BLK 1

area  text(ROM, REL)

_main:

      mov [hour], 7
      mov [min], 14

      cmp [min], 9
      jz resume
      add [min], 6

resume:
      lcall LCD_1_Start

      mov   A, 0
      mov X, 4
      lcall LCD_1_Position

      mov A, >sRomString1
      mov X, <sRomString1
      lcall LCD_1_PrCString

      mov A, 1
      mov X, 4

      lcall LCD_1_Position
      mov A, [hour]
      mov X, 0

      lcall LCD_1_PrHexByte

      mov A, 1
      mov X, 6
      lcall LCD_1_Position

      mov A, >sRomString2
      mov X, <sRomString2
      lcall LCD_1_PrCString

      mov A, 1
      mov X, 7
      lcall LCD_1_Position

      mov A, [min]
      mov X, 0
      lcall LCD_1_PrHexByte

area  lit

sRomString1:
DS "My Clock"
db 00h

sRomString2:
DS ":"
db 00h
```

Exercise 2:
I then wrote a program that generates an interrupt every 1 second. I used the timer module rather than the PSoC Sleep Timer. I used the 16-bit Timer Module. My code is shown below.

```
;----------------------------------------------------------------------------
-
; Assembly main line
;----------------------------------------------------------------------------
-

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

area  text(ROM, REL)

_main:

      mov REG [INT_MSK1], 0x02
      M8C_EnableGInt
      lcall Timer16_1_EnableInt
      lcall Timer16_1_Start
      mov REG[PRT1DR], 0

     loop: jmp loop
```

Exercise 3:
I then wrote code that implements a digital clock on the LCD by combining my excerice 1 and 2. I used the interrupt signal to change the seconds value every one second and display it on the screen. For the systems settings, I chose VC1=10, VC2=10 and VC3 divided by 30 to obtain a 1kHz signal.

```
;----------------------------------------------------------------------------
; Assembly main line
;----------------------------------------------------------------------------
include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

area  bss(RAM)
      hour: BLK 1
      min:  BLK 1

area  text(ROM, REL)

_main:

      mov REG[INT_MSK1], 0x02
      M8C_EnableGInt
      lcall Timer16_1_EnableInt
      lcall Timer16_1_Start
      mov REG[PRT1DR], 0

loop:
      lcall LCD_1_Start

      mov   A, 0
      mov X, 4
      lcall LCD_1_Position

      mov A, >sRomString1
      mov X, <sRomString1
      lcall LCD_1_PrCString

      mov A, 1
      mov X, 4
      lcall LCD_1_Position

      mov A, [hour]
      mov X, 0

      lcall LCD_1_PrHexByte

      mov A, 1
      mov X, 6
      lcall LCD_1_Position

      mov A, >sRomString2
      mov X, <sRomString2
      lcall LCD_1_PrCString

      mov A, 1
      mov X, 7
      lcall LCD_1_Position

      mov A, [min]
      mov X, 0
      lcall LCD_1_PrHexByte

      jmp loop

area  lit

sRomString1:
DS "My Clock"
db 00h
sRomString2:
DS ":"
db 00h

;-------------------------------
; SleepTimer ISR
;-------------------------------
include "m8c.inc"
export SleepTimerISR

SleepTimerISR:

      inc [min]
      mov A, [min]
      and A, 0x0F
      cmp A, 10
      jz addmin
      jmp compare

addmin:
      add [min], 6

compare:
      cmp [min], 0x60
      jz minutes
      reti

hours:
      mov [min], 0
      inc [hour]
      mov A, [hour]
      and A, 0x0F
      cmp A, 10
      jz addhour
      reti

addhour:
      add [hour], 6
     reti
```

Discussion

I did not encounter any technical issues with this lab but I did have a hard time figuring out the code for exercise 2 and 3. After working on it for a while, I was able to solve the problem.

Conclusion

This lab has taught me how to use the LCD User Module and how to use it to display various data on the screen. I have also learned how to use the interrupt timer in my projects to create a clock. Knowing how to program an LCD is a useful tool and this lab has given me the knowledge to do so.