

## 1 Introduction

In this lab, you will write a program that uses two prewritten modules – a time delay module, *OCDelay*, and an LCD module, *Lcd*. The program you will write performs a test for the LCD module by calling all of the functions designated below. In addition, you will develop this lab as a cooperative multitasking timeslice loop to update a clock.

## 2 Modules and Project Files

**Linker Command File and Modules.** The four modules you will use are the startup module, the vector module, *OCDelay.c*, and *Lcd.c*. You will have to copy the source and header files from the locations shown below to your project directory.

<b>9S12DP512 Module Files</b>
Y:/emicros/ProjS12/emodules/OCDelay/s12/Source/OCDelay.c
Y:/emicros/ProjS12/emodules/OCDelay/s12/Source/OCDelay.h
Y:/emicros/ProjS12/emodules/LCD/s12/Lcd/Source/Lcd.c
Y:/emicros/ProjS12/emodules/LCD/s12/Lcd/Source/Lcd.h
Y:/emicros/ProjS12/Startup Files/9S12DP256Start.c
Y:/emicros/ProjS12/Vector Files/S12DP256/s12dp256vecs.a12
<b>9S12DP256 Linker Command File for Stalalone Operation</b>
Y:/emicros/ProjS12/Linker Files/9S12DP256sm.prm

Once the vector file is copied into your project directory, you will have to configure the interrupt vectors. To do this, open the vector file and replace the `__exit` for the appropriate vector with the ISR name. You will also have to add the ISR name to the XREF list at the top of the vector definitions. In this case, the interrupt service routine that is used is *OC0Isr* from the *OCDelay* module.

In addition to these modules, you will be creating your own module that will have the main function and the LCD test code. It should be named something like *LcdTest.c* or *LcdDemo.c*. Also, don't forget your master includes file, the Makefile, and the project environment file.

## 3 Program Requirements

The overall structure of your program must be a timeslice loop that contains two tasks – one for the LCD demonstration code, and the other to update the clock. Because the demonstration code takes much longer than the required timeslice period, you must use state decomposition.

**Clock.** The clock short term accuracy must be within  $\pm 100$ ms. The current time must be stored in a structure of the following type:

```
typedef struct {
    INT8U hrs;
    INT8U mins;
    INT8U secs;
}TIME;
```

The time must always be displayed on the LCD in the upper right hand corner. This is a time-of-day clock so design the code to wrap the time after 12:59. There is no requirement to set the clock in this lab. As an exercise, you can put the time structure in the watch window of the Noral debugger and modify the contents to the current time.

## ETec454 Lab #3 – Standalone LCD Demonstration with Clock

**LCD Demonstration.** The second task involves demonstrating all of the LCD module public functions. While doing this, make sure not to erase or write over the time display – the exceptions are the *LcdInit()*, *LcdClrDisp()*, and *LcdClrLine()* for the top line. Test these functions before displaying the time. The following LCD functions must be called in your main test program. How they are called and the resulting LCD display is up to you, as long as the functions can be verified by a human watching the display – i.e. don't go too fast.

Function	Comments
void LcdInit(void)	Required LCD Initialization
void LcdClrDisp(void)	
void LcdClrLine(INT8U line)	Test on both lines
void LcdDispByte(INT8U *b)	
void LcdDispDecByte(INT8U*b, INT8U lz)	Test with leading zeros and without
void LcdDispStrg(INT8U_t *s)	
void LcdMoveCursor(INT8U row, INT8U col)	
void LcdDispTime(INT8U hrs, INT8U mins, INT8U secs)	
void LcdCursor(INT8U on, INT8U blink)	Test the four possible modes
void LcdBSpace(void)	
void LcdFSpace(void)	
void LcdDispChar(INT8U c)	

### 4 Design Process

The first step in designing this system is to calculate the required timeslice period. In this case, the determining requirement is the short term accuracy of the clock.

Next, implement the timeslice loop, including task skeletons, which include code for debugging helper signals. Debugging helper signals must be defined as follows:

Task	Debug Bit
TimeSlice()	PP7
ClockTask()	PP6
LcdDemoTask()	PP5

You must keep the helper signals in the final product.

Once this part is working, you can proceed with designing the tasks. It is important to keep an eye on the debugging signals while you add code to make sure the timeslice loop is working properly and no task is blocking.

In this program, you will have two tasks accessing the LCD. Because of this you will have to be very careful about the cursor location. If the LCD demonstration program is finished with a state, the program then updates the clock, note that the cursor is now at a different location. So, it must be reset for the next LCD demonstration state.

### 5 Write-up

The source code for your project must be released by midnight on the due date. The write-up must include the following material: (Due: Source: Feb 12, 2008, Write-up: Feb 15, 2008)

**Introduction**

**Program Description**

**Source Code and Header File(s) (hardcopy)**

**Comments and Conclusions**