Lab 2: Clocks and Timers

Instructor's Guide

Lab Introduction

This lab introduces basic concepts of synchronous circuitry and clocks along with one synchronous component: timers. It would be a good idea to cover synchronous circuits in more detail outside of the lab in a lecture, as the topic is too complex to teach in one lab session.

Instructor Review

Part A

The schematic is provided in its entirety and the code section simply requires initialization code for the timer. Just follow the testing instructions in Part 3 of the procedure. Note that Part B of the procedure does not remove functionality of Part A, so a review of Part A can be delayed if desired.

Part B

The schematic for this part is also provided in its entirety. However, this time the code is fully up to the student to write. Here is a sample firmware solution:

```
#include "stdio.h"
#include <project.h>

#define TRUE  1
#define FALSE 0
#define STOPWATCH_FREQ 10000

asm (".global _printf_float");  //adds ~8000 bytes to the program

int main()
{
      //Set to TRUE if the stopwatch has started, else FALSE
      uint8 started_b = FALSE;
      //Temporary storage for status register
      uint8 reg;
      //Capture/counter read value
      uint32 capture;
      //Capture/counter converted to seconds
      float seconds;
      //Temporary string for sprintf to use
      char tstr[16];

      //Start all components and init display
      SecondTimer_Start();
      Stopwatch_Start();
```

```
    Display_Start();
    StopwatchStart_Read();
    Display_Position(0, 0);
    Display_PrintString("Stopped");
  for(;;)
  {
        if(!started_b)
        {
            //If the stopwatch isn't known to have started yet,
            //see if the start button has been pressed. If it has
            //been pressed, the stopwatch has started.
            if(StopwatchStart_Read())
            {
                started_b = TRUE;
                Display_ClearDisplay();
                Display_Position(0, 0);
                Display_PrintString("Started");
            }
        }
        else
        {
            //Else, check to see if the stopwatch has been stopped.
            reg = Stopwatch_ReadStatusRegister();

            //If the stopwatch has been stopped, first read the capture
            if(reg & Stopwatch_STATUS_CAPTURE)
            {
                capture = Stopwatch_ReadCapture();
                started_b = FALSE;
            }
            //Else, just read the counter
            else
            {
                capture = Stopwatch_ReadCounter();
            }

            //Convert the capture/counter to seconds and print it
            capture = Stopwatch_ReadPeriod() - capture;
            if(capture != 0)
            {
                seconds = capture / (float)STOPWATCH_FREQ;
                Display_Position(1, 0);
                sprintf(tstr, "%1.4f", seconds);
                Display_PrintString(tstr);
            }

            //If the stopwatch has stopped, reset it
            if(!started_b)
            {
                StopwatchReset_Write(1);
                StopwatchStart_Read();
                Display_Position(0, 0);
                Display_PrintString("Stopped");
            }
        }
  }
}
```