Interrupts

Overview:
In this lab, you will write a simple stopwatch program that uses SysTick Interrupts to keep track of time. You will use polling to monitor the Start and Stop buttons on the MSP432 LaunchPad.

Stopwatch Using SysTick Interrupts and Switch Inputs
Create a new sketch in Energia and save it to your Cosmos Workspace.

The flowcharts for the main program and the SysTick ISR are shown in Figure 1. Your stopwatch should measure time to the hundredth of a second. The time will be displayed as *seconds.hundredths*. You do not have to convert the time to show minutes when seconds exceeds 59. Of course, hundredths should always be between 0 and 99.

You must use global variables in order to keep track of seconds and hundredths. These can be individual variables or you can use a structure and have seconds and hundredths as members of the structure. In either case, these variables should be declared as **volatile**.

The initialization block should include properly configuring GPIO pins for switch inputs and outputs to control the RGB LED, turning off all LEDs, and setting the SysTick period. Pushbutton S1 (P1.1) will be the Start button and pushbutton S2 (P1.4) will be the Stop button.

You should ***not*** print to the serial terminal while your stopwatch is running because Serial.print() disables interrupts, which will make your stopwatch timing inaccurate. Also, it is bad practice to print from an ISR since your ISR should be as fast as possible. Therefore, your program will not display the running time on the console. Instead, the program will update the RGB LED color once per second when the stopwatch is running in order to provide a visual indication that SysTick interrupts are working and that the elapsed time is being updated. You can use your RGB_output() function from previous labs to control the RGB LED.

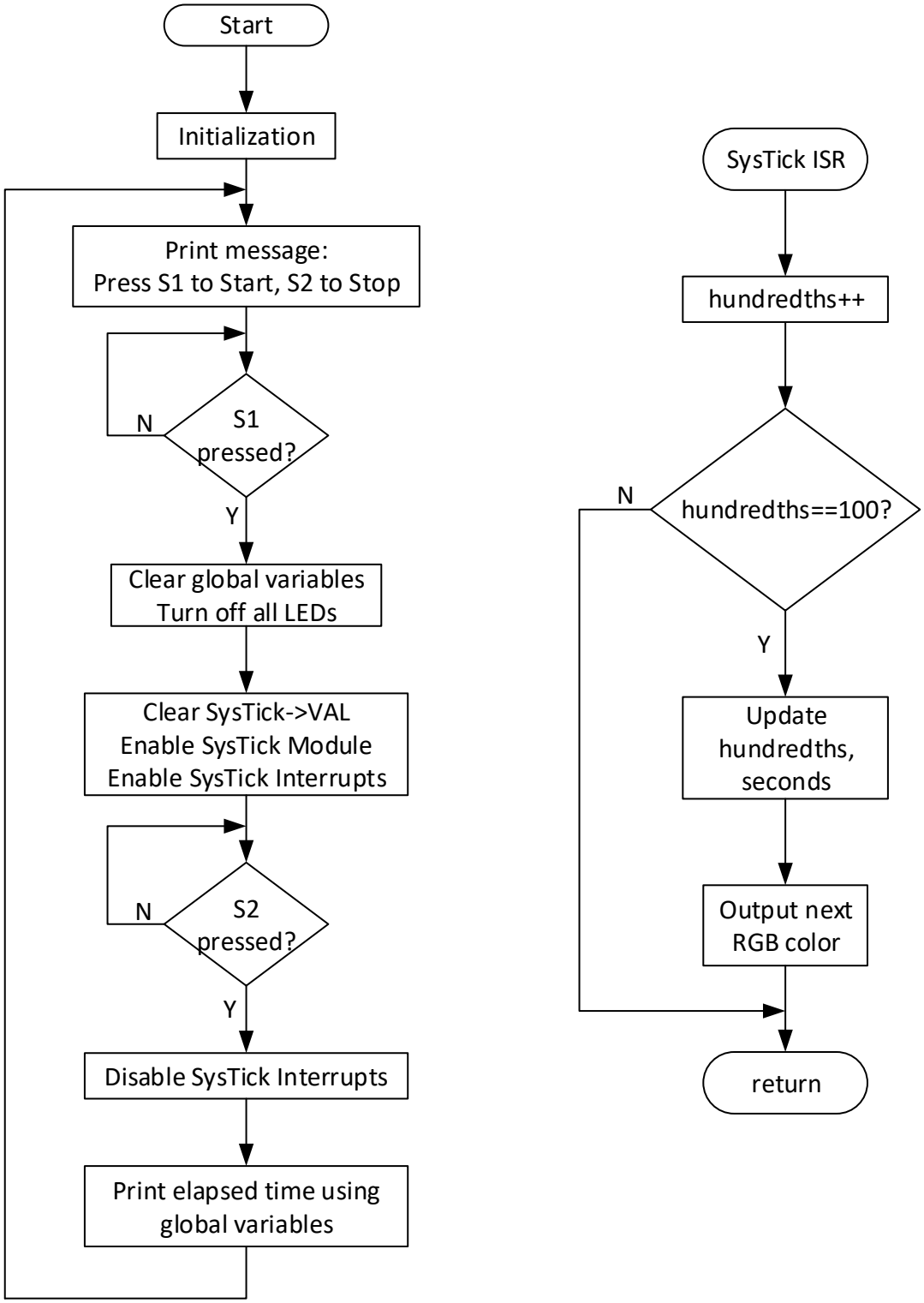

Figure 1: Flowcharts for main function and SysTick ISR

```
// Built-in Interrupt handler function name
void SysTick_Handler (void){

    /*
     * Implement stopwach
     */
}
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  // Push button 1 is an input (start button)
  GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1, GPIO_PIN1);

  // Push button 2 is an input (stop button)
  GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1, GPIO_PIN4);

  // Set RGB Led as output and set low
  GPIO_setAsOutputPin(GPIO_PORT_P2, GPIO_PIN0 | GPIO_PIN1 | GPIO_PIN2);
  GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN0 | GPIO_PIN1 | GPIO_PIN2);

  SysTick_registerInterrupt(SysTick_Handler);
  SysTick_setPeriod(480000);

  // Disable Interrupt for now
  SysTick_disableInterrupt();

}
```

Figure 2: Interrupt Setup for Systick Timer

```
void loop() {

  Serial.println("Press S1 to Start, Press S2 to Stop");

  /*
   * Poll SW1 Here
   * You must debounce this switch
   */


  /*
   * Counting has started - set LED low, reset all variables
   * You must properly reset the Systick Registers / Enable Interrupts
   */

  /*
   * Poll SW2 Here
   * You must debounce this switch
   */

  // Disable Interrupt
  SysTick_disableInterrupt();

  Serial.print(seconds);
  Serial.print(",");
  Serial.print(hundredths);
  Serial.print("\n");
}
```

Figure 3: Looping Function for Systick Stopwatch