Jiajian Chen

Github root directory:
https://github.com/IMSB007/2019Fall

Date Submitted:

Task 00: Execute provided code

Youtube Link:

Task 01:

Youtube Link: https://www.youtube.com/watch?v=P24fV8Rr4Oo

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
int main(void)
 {
  int32_t ui32ADC0Value[4];
  volatile uint32_t ui32TempAvg;
  volatile uint32_t ui32TempValueC;
  volatile uint32_t ui32TempValueF;
  uint32_t pui32DataTx[NUM_SSI_DATA];
  uint32_t pui32DataRx[NUM_SSI_DATA];
  uint32_t ui32Index;
  SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);
  InitConsole();
  SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
  ADCHardwareOversampleConfigure(ADC0_BASE, 32);
  ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
  ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
  ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
  ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
  ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
  ADCSequenceEnable(ADC0_BASE, 1);
  UARTprintf("SSI ->\n");
  UARTprintf(" Mode: SPI\n");
  UARTprintf(" Data: 8-bit\n\n");
  SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
  GPIOPinConfigure(GPIO_PA2_SSI0CLK);
   GPIOPinConfigure(GPIO_PA3_SSI0FSS);
   GPIOPinConfigure(GPIO_PA4_SSI0RX);
   GPIOPinConfigure(GPIO_PA5_SSI0TX);
   GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_5 | GPIO_PIN_4 | GPIO_PIN_3 |
GPIO_PIN_2);
   SSIConfigSetExpClk(SSI0_BASE, SysCtlClockGet(), SSI_FRF_MOTO_MODE_0,
SSI_MODE_MASTER, 1000000, 8);
   // Enable the SSI0 module.
   SSIEnable(SSI0_BASE);
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Github root directory:
https://github.com/IMSB007/2019Fall

```
   while(1)
   {

       ADCIntClear(ADC0_BASE, 1);
       ADCProcessorTrigger(ADC0_BASE, 1);
       while(!ADCIntStatus(ADC0_BASE, 1, false))
       {
       }
       ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
       ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
       ui32TempValueC = (1475 -((2475 * ui32TempAvg)) / 4096)/10;
       ui32TempValueF = ((ui32TempValueC * 9)+ 160) / 5;
       while(SSIDataGetNonBlocking(SSI0_BASE, &pui32DataRx[0]))
       {
       }
       pui32DataTx[0] = ui32TempValueC;
       pui32DataTx[1] = ui32TempValueF;
       //pui32DataTx[2] = 'i';
       UARTprintf("Sent:\n ");

       for(ui32Index = 0; ui32Index < NUM_SSI_DATA; ui32Index++)
       {

           UARTprintf("%d ", pui32DataTx[ui32Index]);
           SSIDataPut(SSI0_BASE, pui32DataTx[ui32Index]);
       }
       while(SSIBusy(SSI0_BASE))
       {
       }
       UARTprintf("\nReceived:\n ");
       for(ui32Index = 0; ui32Index < NUM_SSI_DATA; ui32Index++)
       {
           SSIDataGet(SSI0_BASE, &pui32DataRx[ui32Index]);
           pui32DataRx[ui32Index] &= 0x00FF;
           UARTprintf("%d ", pui32DataRx[ui32Index]);
       }
    }
    // Return no errors
    return(0);
}
```

Task 02:

Youtube Link: https://www.youtube.com/watch?v=mTD3BCsvsNg

Modified Schematic (if applicable):

Modified Code:

```
// Insert code here
int main(void)
{
    FPULazyStackingEnable();// 80MHz
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Github root directory:

https://github.com/IMSB007/2019Fall

```
    SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ
|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlDelay(50000);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);
    SysCtlDelay(50000);
    GPIOPinConfigure(GPIO_PA5_SSI0TX);
    GPIOPinConfigure(GPIO_PA2_SSI0CLK);
    GPIOPinConfigure(GPIO_PA4_SSI0RX);
    GPIOPinConfigure(GPIO_PA3_SSI0FSS);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_5);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_2);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_4);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_3);//20 MHz data rate
    SSIConfigSetExpClk(SSI0_BASE, 80000000, SSI_FRF_MOTO_MODE_0, SSI_MODE_MASTER,
2400000, 9);
    SSIEnable(SSI0_BASE);
    while(1)
    {

        fill_frame_buffer(255, 0, 0, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        fill_frame_buffer(0, 255, 0, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        fill_frame_buffer(0, 0, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        fill_frame_buffer(255, 255, 0, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        fill_frame_buffer(255, 0, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        fill_frame_buffer(0, 255, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        fill_frame_buffer(255, 255, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
    }
    return 0;
}
void send_data(uint8_t* data, uint8_t num_leds)
{
    uint32_t i, j, curr_lut_index=0, curr_rgb;
    for(i = 0; i < (num_leds*3); i = i + 3)
    {
        curr_rgb = (((uint32_t)data[i + 2]) << 16) | (((uint32_t)data[i + 1]) << 8) |
data[i];
        for(j = 0; j < 24; j = j + 3)
        {
            curr_lut_index = ((curr_rgb>>j) & 0b111);
            SSIDataPut(SSI0_BASE, ssi_lut[curr_lut_index]);
       }
    }
    SysCtlDelay(50000000); // delay more then 50us
}
void fill_frame_buffer(uint8_t r, uint8_t g, uint8_t b, uint32_t num_leds)
{
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Github root directory:

https://github.com/IMSB007/2019Fall

```
    uint32_t i;
    uint8_t* frame_buffer_index = frame_buffer;
    for(i = 0; i < num_leds; i++)
    {
        *(frame_buffer_index++) = g;
        *(frame_buffer_index++) = r;
        *(frame_buffer_index++) = b;
    }
}
--------------------------------------------------------------------------------
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Jiajian Chen

Github root directory:
https://github.com/IMSB007/2019Fall

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.