Adrian Ruiz

CPE 403

Lab 5

Task 00:

Code:

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/debug.h"

#include "driverlib/sysctl.h"

#include "driverlib/adc.h"

int main**(**void**)**

**{**

uint32\_t ui32ADC0Value**[**4**];**

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

SysCtlClockSet**(**SYSCTL\_SYSDIV\_5**|**SYSCTL\_USE\_PLL**|**SYSCTL\_OSC\_MAIN**|**SYSCTL\_XTAL\_16MHZ**);**

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_ADC0**);**

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**);**

**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**;**

**}**

**}**

Task 01:

Altered Code:

//enable port f

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_GPIOF**);**

GPIOPinTypeGPIOOutput**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**);**

This is to enable the LEDs to turn

//Configure the ADC sequencer to sample at 1Msps

ADCSequenceConfigure**(**ADC0\_BASE**,** 3**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

//Configure the ADC sequencer steps to sample temperture

//Sample the tempature and configure the interrupt flag

//Since ss3 can only take one sample at a time, i set up the flag at the same time

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 3**,** 0**,** ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

//Enable the Sequencer

ADCSequenceEnable**(**ADC0\_BASE**,**3**);**

//Clear the ADC Flag

ADCIntClear**(**ADC0\_BASE**,** 3**);**

//ADC conversion with software

ADCProcessorTrigger**(**ADC0\_BASE**,** 3**);**

//wait conversion to be complete

**while(!**ADCIntStatus**(**ADC0\_BASE**,**3**,**false**))**

**{**

**}**

This snippet shows that SS3 was used instead of SS1. SS3 can have one sequence step so I removed the other steps and had the flag reset after one step. It wasn’t as accurate as SS1. If more steps were included, it would crash.

**if(**ui32TempValueF **>** 72**)**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 4**);**

**}**

**else**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 0**);**

**}**

This is where the temperature was checked be greater than 72. If it was then it would turn on the LED. If not, then the LED would turn off.

Code:

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/debug.h"

#include "driverlib/sysctl.h"

#include "driverlib/adc.h"

#define TARGET\_IS\_BLIZZARD\_RB1

#include "driverlib/rom.h"

#include "driverlib/gpio.h"

#ifdef DEBUG

void\_error\_**(**char **\***pcFilename**,** uint32\_t ui32Line**)**

**{}**

#endif

int main**(**void**)**

**{**

uint32\_t ui32ADC0Value**[**4**];** //an array that stores data from the adc

//variables used to hold tempature info

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

//run clock at 40MHz

SysCtlClockSet**(**SYSCTL\_SYSDIV\_5**|**SYSCTL\_USE\_PLL**|**SYSCTL\_XTAL\_16MHZ**);**

//enable the ADC peripheral

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_ADC0**);**

//enable port f

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_GPIOF**);**

GPIOPinTypeGPIOOutput**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**);**

//ADC Sample Averaged

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 64**);**

//Configure the ADC sequencer to sample at 1Msps

ADCSequenceConfigure**(**ADC0\_BASE**,** 3**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

//Configure the ADC sequencer steps to sample temperture

//Sample the tempature and configure the interrupt flag

//Since ss3 can only take one sample at a time, i set up the flag at the same time

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 3**,** 0**,** ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

//Enable the Sequencer

ADCSequenceEnable**(**ADC0\_BASE**,**3**);**

**while(**1**)**

**{**

//Clear the ADC Flag

ADCIntClear**(**ADC0\_BASE**,** 3**);**

//ADC conversion with software

ADCProcessorTrigger**(**ADC0\_BASE**,** 3**);**

//wait conversion to be complete

**while(!**ADCIntStatus**(**ADC0\_BASE**,**3**,**false**))**

**{**

**}**

//Copy ADC samples into the array

ADCSequenceDataGet**(**ADC0\_BASE**,**3**,** ui32ADC0Value**);**

//Calculate the average tempature samples

ui32TempAvg **=** **(**ui32ADC0Value**[**0**]+**ui32ADC0Value**[**1**]+**ui32ADC0Value**[**2**]+**ui32ADC0Value**[**3**]** **+** 2**)/**4**;**

//Calculate the Tempature in Celcuis from the samples

ui32TempValueC **=** **(**1475 **-** **((**2475 **\*** ui32TempAvg**))** **/** 4096**)** **/** 10**;**

//Calculate The Temp in Far

ui32TempValueF **=** **((**ui32TempValueC **\*** 9**)** **+** 160**)** **/** 5**;**

//GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 8);

**if(**ui32TempValueF **>** 72**)**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 4**);**

**}**

**else**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 0**);**

**}**

**}**

**}**

Task 03:

Altered Code:

//ADC Sample Averaged

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 32**);**

//Configure the ADC sequencer to sample at 1Msps

ADCSequenceConfigure**(**ADC0\_BASE**,** 1**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

//Configure the ADC sequencer steps to sample temperture

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 0**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 1**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 2**,** ADC\_CTL\_TS**);**

//Sample the tempature and configure the interrupt flag

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 3**,** ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

I changed the Sequencer back to SS1 for more accuracy. I also change the ADC sample Average to 32 as in the directions.

//Enable Timer1, set period, and enable interrupt

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_TIMER1**);**

TimerConfigure**(**TIMER1\_BASE**,**TIMER\_CFG\_PERIODIC**);**

TimerLoadSet**(**TIMER1\_BASE**,** TIMER\_A**,(**SysCtlClockGet**()/**2**)-**1**);**

IntEnable**(**INT\_TIMER1A**);**

TimerIntEnable**(**TIMER1\_BASE**,** TIMER\_TIMA\_TIMEOUT**);**

IntMasterEnable**();**

This is where the Timer1 is set up. Its period of .5 seconds and interrupt is setup and enabled.

void Timer1IntHandler**(**void**)**

**{**

TimerIntClear**(**TIMER1\_BASE**,** TIMER\_TIMA\_TIMEOUT**);**

uint32\_t ui32ADC0Value**[**4**];** //an array that stores data from the adc

//variables used to hold tempature info

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

//Clear the ADC Flag

ADCIntClear**(**ADC0\_BASE**,** 1**);**

//ADC conversion with software

ADCProcessorTrigger**(**ADC0\_BASE**,** 1**);**

//wait conversion to be complete

**while(!**ADCIntStatus**(**ADC0\_BASE**,**1**,**false**))**

**{**

**}**

//Copy ADC samples into the array

ADCSequenceDataGet**(**ADC0\_BASE**,**1**,** ui32ADC0Value**);**

//Calculate the average tempature samples

ui32TempAvg **=** **(**ui32ADC0Value**[**0**]+**ui32ADC0Value**[**1**]+**ui32ADC0Value**[**2**]+**ui32ADC0Value**[**3**]** **+** 2**)/**4**;**

//Calculate the Tempature in Celcuis from the samples

ui32TempValueC **=** **(**1475 **-** **((**2475 **\*** ui32TempAvg**))** **/** 4096**)** **/** 10**;**

//Calculate The Temp in Far

ui32TempValueF **=** **((**ui32TempValueC **\*** 9**)** **+** 160**)** **/** 5**;**

**if(**ui32TempValueF **>** 69**)**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 4**);**

**}**

**else**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 0**);**

**}**

**}**

This is the Timer1 interrupt sub routine. The flag is cleared and everything inside the while loop and the variables is moved into the interrupt.

Code:

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "inc/tm4c123gh6pm.h"

#include "driverlib/debug.h"

#include "driverlib/sysctl.h"

#include "driverlib/adc.h"

#define TARGET\_IS\_BLIZZARD\_RB1

#include "driverlib/rom.h"

#include "driverlib/gpio.h"

#include "driverlib/timer.h"

#include "driverlib/interrupt.h"

#ifdef DEBUG

void\_error\_**(**char **\***pcFilename**,** uint32\_t ui32Line**)**

**{}**

#endif

int main**(**void**)**

**{**

//run clock at 40MHz

SysCtlClockSet**(**SYSCTL\_SYSDIV\_5**|**SYSCTL\_USE\_PLL**|**SYSCTL\_XTAL\_16MHZ**);**

//enable the ADC peripheral

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_ADC0**);**

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_GPIOF**);**

GPIOPinTypeGPIOOutput**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**);**

//ADC Sample Averaged

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 32**);**

//Configure the ADC sequencer to sample at 1Msps

ADCSequenceConfigure**(**ADC0\_BASE**,** 1**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

//Configure the ADC sequencer steps to sample temperture

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 0**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 1**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 2**,** ADC\_CTL\_TS**);**

//Sample the tempature and configure the interrupt flag

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 3**,** ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

//Enable Timer1, set period, and enable interrupt

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_TIMER1**);**

TimerConfigure**(**TIMER1\_BASE**,**TIMER\_CFG\_PERIODIC**);**

TimerLoadSet**(**TIMER1\_BASE**,** TIMER\_A**,(**SysCtlClockGet**()/**2**)-**1**);**

IntEnable**(**INT\_TIMER1A**);**

TimerIntEnable**(**TIMER1\_BASE**,** TIMER\_TIMA\_TIMEOUT**);**

IntMasterEnable**();**

//Enable the Sequencer

ADCSequenceEnable**(**ADC0\_BASE**,**1**);**

TimerEnable**(**TIMER1\_BASE**,** TIMER\_A**);**

**while(**1**)**

**{**

**}**

**}**

void Timer1IntHandler**(**void**)**

**{**

TimerIntClear**(**TIMER1\_BASE**,** TIMER\_TIMA\_TIMEOUT**);**

uint32\_t ui32ADC0Value**[**4**];** //an array that stores data from the adc

//variables used to hold tempature info

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

//Clear the ADC Flag

ADCIntClear**(**ADC0\_BASE**,** 1**);**

//ADC conversion with software

ADCProcessorTrigger**(**ADC0\_BASE**,** 1**);**

//wait conversion to be complete

**while(!**ADCIntStatus**(**ADC0\_BASE**,**1**,**false**))**

**{**

**}**

//Copy ADC samples into the array

ADCSequenceDataGet**(**ADC0\_BASE**,**1**,** ui32ADC0Value**);**

//Calculate the average tempature samples

ui32TempAvg **=** **(**ui32ADC0Value**[**0**]+**ui32ADC0Value**[**1**]+**ui32ADC0Value**[**2**]+**ui32ADC0Value**[**3**]** **+** 2**)/**4**;**

//Calculate the Tempature in Celcuis from the samples

ui32TempValueC **=** **(**1475 **-** **((**2475 **\*** ui32TempAvg**))** **/** 4096**)** **/** 10**;**

//Calculate The Temp in Far

ui32TempValueF **=** **((**ui32TempValueC **\*** 9**)** **+** 160**)** **/** 5**;**

**if(**ui32TempValueF **>** 72**)**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 4**);**

**}**

**else**

**{**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**|**GPIO\_PIN\_2**|**GPIO\_PIN\_3**,** 0**);**

**}**

**}**