

Lab 8: ADC

Farnam Adelkhani

Explanation of Device Settings:

The goal of this lab was to learn how to make use of ADC to continuously sample analog signals from analog input pins or the internal temperature sensor. The skills we learned are ADC configuration and analog input pins, handling of ADC triggered interrupts, and real-time data collection as well.

The sample project “ADC_temperature” will be modified to use ADC0, SS3 in the modified implementation instead of ADC0 and SS1 to sample the signals from the internal temperature sensor. Only one sample can be stored at SS3 FIFO so the user does not need to calculate ui32TempAvg anymore. Instead our group will just calculate ui32TempValueC and ui32TempValueF based on the single sample from the ADC0 SS3 FIFO. The project will need to be run in debug mode and the values of defined variables will need to be watched in the watch window. Our group is familiar with how to add a variable to watch from previous lab assignments.

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"
#include "driverlib/interrupt.h"
#include "inc/tm4c123gh6pm.h"

//*****
//!
//! In this project we use ADC0, SS3 to measure the data from the on-chip
//! temperature sensor. Then we enabled analog input pin PE3 and connected
//! it to another project to see the fluctuation of high and low frequencies.
//!
//*****

uint32_t ui32ADC0Value[1];
```

```

//volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

//ADC0 initializaiton
void ADC0_Init(void)
{

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ); // configure the system clock to be 40MHz
SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0); //activate the clock of ADC0
SysCtlDelay(2); //insert a few cycles after enabling the peripheral to allow the clock to be fully activated.

        ADCSequenceDisable(ADC0_BASE, 3); //disable ADC0 before the configuration is complete
        ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0); // will use ADC0, SS3, processor-trigger, priority 0
        //ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_TS); //ADC0 SS3 Step 0, sample from internal temperature sensor
        //ADCSequenceStepConfigure(ADC0_BASE, 3, 1, ADC_CTL_TS); //ADC0 SS3 Step 1, sample from internal temperature sensor
        //ADCSequenceStepConfigure(ADC0_BASE, 3, 2, ADC_CTL_TS); //ADC0 SS3 Step 2, sample from internal temperature sensor
        //ADC0 SS3 Step 0, sample from internal temperature sensor, completion of this step will set RIS, last sample of the sequence

ADCSequenceStepConfigure(ADC0_BASE,3,0,ADC_CTL_CH0|ADC_CTL_IE|ADC_CTL_END)
;
    //change ADC_CTL_TS to ADC_CTL_CH0 which means the analog input signal
    // will be from Ain0

        IntPrioritySet(INT_ADC0SS3, 0x00); // configure ADC0 SS3 interrupt priority as 0
        IntEnable(INT_ADC0SS3); // enable interrupt 31 in NVIC (ADC0 SS3)
        ADCIntEnableEx(ADC0_BASE, ADC_INT_SS3); // arm interrupt of ADC0 SS3

        ADCSequenceEnable(ADC0_BASE, 3); //enable ADC0

```

```

}

//interrupt handler
void ADC0_Handler(void)
{

    ADCIntClear(ADC0_BASE, 3);
    ADCProcessorTrigger(ADC0_BASE, 3);
    ADCSequenceDataGet(ADC0_BASE, 3, ui32ADC0Value);
//ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = ui32ADC0Value[0];
    ui32TempValueF = (ui32TempValueC);
}

int main(void)
{
    ADC0_Init();
    IntMasterEnable();           // globally enable interrupt
    ADCProcessorTrigger(ADC0_BASE, 3);

    while(1)
    {

    }
}

```

Explanation of Device Settings:

For the second part of this lab assignment the project that was implemented in task 1 will instead of sampling signal from the internal temperature sensor, it will begin sampling signals from the analog input port Ain0 using ADC0 SS3 in the modified implementation that was setup. ADC SS will need to have it's configuration modified and this is done by means of replacing the parameter ADC_CTL_TS with ADC_CTL_CH0 in the function ADCSequenceStepConfigure().

The project is then evaluated by connecting PE3 on the board to the output on another launchpad where another project is running and changing the value on this output pin PF1 on that other board and connected to the analog input pin via the P#3 pin on this board. It is imperative to connect the ground pins of the two boards too. There was no need for us to borrow the male to male wires from Dr Anwar since we already owned wires.

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"
#include "driverlib/interrupt.h"
#include "inc/tm4c123gh6pm.h"

//*****
//!
//! In this project we use ADC0, SS3 to measure the data from the on-chip
//! temperature sensor. Then we enabled analog input pin PE3 and connected
//! it to another project to see the fluctuation of high and low frequencies.
//!
//*****

uint32_t ui32ADC0Value[1];
//volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;

//ADC0 initialization
void ADC0_Init(void)
```

```

{

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ); // configure the system clock to be 40MHz
SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0); //activate the clock of ADC0
SysCtlDelay(2); //insert a few cycles after enabling the peripheral to allow the clock to be fully activated.

    ADCSequenceDisable(ADC0_BASE, 3); //disable ADC0 before the configuration is complete
    ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0); // will use ADC0, SS3, processor-trigger, priority 0
    //ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_TS); //ADC0 SS3 Step 0, sample from internal temperature sensor
    //ADCSequenceStepConfigure(ADC0_BASE, 3, 1, ADC_CTL_TS); //ADC0 SS3 Step 1, sample from internal temperature sensor
    //ADCSequenceStepConfigure(ADC0_BASE, 3, 2, ADC_CTL_TS); //ADC0 SS3 Step 2, sample from internal temperature sensor
    //ADC0 SS3 Step 0, sample from internal temperature sensor, completion of this step will set RIS, last sample of the sequence

ADCSequenceStepConfigure(ADC0_BASE,3,0,ADC_CTL_CH0|ADC_CTL_IE|ADC_CTL_END)
;
    //change ADC_CTL_TS to ADC_CTL_CH0 which means the analog input signal
    // will be from Ain0
    IntPrioritySet(INT_ADC0SS3, 0x00); // configure ADC0 SS3 interrupt priority as 0
    IntEnable(INT_ADC0SS3); // enable interrupt 31 in NVIC (ADC0 SS3)
    ADCIntEnableEx(ADC0_BASE, ADC_INT_SS3); // arm interrupt of ADC0 SS3

    ADCSequenceEnable(ADC0_BASE, 3); //enable ADC0
}

//interrupt handler
void ADC0_Handler(void)
{

```

```

        ADCIntClear(ADC0_BASE, 3);
        ADCProcessorTrigger(ADC0_BASE, 3);
        ADCSequenceDataGet(ADC0_BASE, 3, ui32ADC0Value);
//ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
        ui32TempValueC = ui32ADC0Value[0];
        ui32TempValueF = (ui32TempValueC);
    }

int main(void)
{
    ADC0_Init();
    IntMasterEnable();           // globally enable interrupt
    ADCProcessorTrigger(ADC0_BASE, 3);

    while(1)
    {

    }
}

```

Answer to question in Task 3:

If you don't want a high sampling frequency for the application, you would need to create a periodic interrupt that would trigger the sampling rate of ADC at every 1000 Hz. This just means you would need to implement a timer interrupt._____

Discussion and Suggestions:

Given a sample project to modify, this project was straight forward. Out groups were given the necessary implementations that need to be made and asked to modify the project in accordance to the directions given. Task one of this lab assignment was to modify the sample project in accordance to the given directions. Changes needed to be made to the program to allow the modifications to take action. The explanation of the necessary modifications that need to take place are given in a straightforward manner within the text. The implementations were made to the program and everything worked as expected.

There are no suggestions that we can think of regarding this lab assignment. The sample project provided was modified and so this eliminated the possibility of any major

misconceptions and errors in this lab assignment. It was a bit of a luxury to finish the class on this lab assignment. The reason why is that this lab assignment was rewarding in the regard that our group found this program was great for applying to other tasks that our group would like to achieve.