EECS 388

LABORATORY EXERCISE V

ANALOG-TO-DIGITAL CONVERTER

November 1, 2018

Benjamin Streit

GTA: Ibikunle Oluwanisola

Laboratory Overview

This laboratory involves the writing, compiling, and demonstrating of a new task on the Tiva TM4C1294 evaluation board. This task reads the Analog-to-Digital Converter (ADC) on-board the evaluation board and display the read values using PuTTY.

Background

UART and PuTTY will be used to print a ADC values once every second. PuTTY is prepared by setting the COM port to whichever COM port is being used on the workstation by the evaluation board, and setting the serial connection speed to 115200. This will open a terminal accepting output from programs being run on the evaluation board. UART works similarly to a standard `printf` statement in C syntax, and example use is as follows:

```
// Import APIs
#include "Drivers/UARTStdio_Initialization.h"
#include "Drivers/uartstdio.h"
// Initialize UART
UARTStdio_Initialization();
// Print to PuTTY
UARTprintf( "FreeRTOS Starting!\n" );
```

Procedure & Results

Design of the task `Task_ADC.c` is in three parts: (i) Initializing UART, (ii) Configuring the ADC, and (iii) Sampling, Reading, and Printing the ADC value. Overview of each part is below:

(i) Initialization of UART is done via the following call:

```
    UARTStdio_Initialization();
```

Now the program is prepared to print to PuTTY.

(ii) Configuration of the ADC involves enabling the peripheral, and configuring and enabling the ADC sequence.

```
    SysCtlPeripheralEnable( SYSCTL_PERIPH_ADC0 );

    ADCSequenceConfigure( ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 1 );
    ADCSequenceStepConfigure( ADC0_BASE, 0, 0,
                              ADC_CTL_IE | ADC_CTL_END | ADC_CTL_CH9 );

    ADCSequenceEnable( ADC0_BASE, 0 );
```

Now the program is prepared to read potentiometer voltage on the board.

(iii) Reading the ADC consists of triggering a sample sequence, waiting until said sequence is complete, reading the sequence values from the ADC, and printing to PuTTY. This is done via the following calls:

```
    ADCProcessorTrigger(ADC0_BASE, 0);
    while(!ADCIntStatus(ADC0_BASE, 0, false)) {
    }
    ADCSequenceDataGet(ADC0_BASE, 0, &ulValue);
    UARTprintf( "ADC:%i\n", ulValue);
```

Note the sampled sequence is being set to a previously declared `unsigned long` variable by the name of `ulValue`.

Lastly, as all of part (iii) takes place within an infinite while loop, the task delays for one second before beginning at the top of the loop again using the following call:

```
vTaskDelay( pdMS_TO_TICKS(1000) );
```

ANALYSIS

The ADC on the Tiva TM4C1294 evaluation board generates 12-bits for a conversion, this information can be used to make additional sense of the this task's generated output in PuTTY. The range of possible values that the ADC can represent is computed as follows:

$$0 \leq x < 2^{12}$$
$$0 \leq x < 4096$$

It is clear that the lowest value the ADC can represent is 0, and the highest value is 4095, as only integer values are generated. With this in mind, one can expect PuTTY to read `0` when the potentiometer is turned all the way "down", and `4095` when the potentiometer is turned all the way "up", as well as every integer value between `0` and `4095` along the way.

Conclusions

This laboratory exercises observes the reading of the ADC on the Tiva TM4C1294 evaluation board once per second, without any interaction from a user. The implemented logic is broken into easily understandable steps and is consequently both appropriately modular and effective. Thus, this logic could be extended to include interaction from the user, or perhaps reading of an additional peripheral as a means of introducing new complexities to the reading and processing of the ADC.

Code

```c
#include <stddef.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdarg.h>

#include "FreeRTOS.h"
#include "task.h"

#include "Drivers/UARTStdio_Initialization.h"
#include "Drivers/uartstdio.h"

#include "Driverlib/adc.h"
#include "Driverlib/sysctl.h"
#include "inc/hw_memmap.h"

extern void Task_ADC( void *pvParameters ) {

    unsigned long ulValue;

    //
    // Initialize UART
    //
    UARTStdio_Initialization();

    //
    // Configure ADC<9> to read potentiometer voltage.
    //
    SysCtlPeripheralEnable( SYSCTL_PERIPH_ADC0 );

    ADCSequenceConfigure( ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 1 );
    ADCSequenceStepConfigure( ADC0_BASE, 0, 0,
                              ADC_CTL_IE | ADC_CTL_END | ADC_CTL_CH9 );

    ADCSequenceEnable( ADC0_BASE, 0 );

    while ( 1 ) {
        //
        // Trigger the sample sequence.
        //
        ADCProcessorTrigger(ADC0_BASE, 0);

        //
        // Wait until the sample sequence has completed.
        //
        while(!ADCIntStatus(ADC0_BASE, 0, false)) {

        }

        //
        // Read the value from the ADC.
        //
        ADCSequenceDataGet(ADC0_BASE, 0, &ulValue);

        //
        // Print
        //
        UARTprintf( "ADC:%i\n", ulValue);

        //
        // Delay
        //
        vTaskDelay( pdMS_TO_TICKS(1000) );
    }
}
```