Introduction to Mechatronics (ME/AE 6705)

Lab Assignment 6

Data Acquisition on the MSP432P4111

Objective

This purpose of this lab is to learn how to use onboard flash memory to save data acquired through analog to digital conversion or other means. A secondary objective is to learn how to print binary data in ASCII format using the `sprintf()` function.

Deliverables and Grading

To get credit for this lab assignment you must:

1. Submit a typed report as a PDF file to Canvas, answering the questions at the end of the lab (2 pages max, can be shorter). This is due at the beginning of class on the due date specified on Canvas. (30 points)

2. Demonstrate proper operation of your code to TAs or instructor during office hours or demo hours. You must write your own code for this lab – no lab groups are allowed. (50 points)

3. Submit the commented final version of your code on Canvas (single .c file containing your main() function, do not submit header files, etc. You should only submit a single file.) (Pass/Fail)

Setup

This lab requires Code Composer Studio, the MSP432, and the temperature sensor circuit built in Lab 1. The lab uses onboard features of the device such as flash memory, ADC, and UART. The MSP432 has 256KB of flash main memory and 16KB of flash information memory. This lab makes use of main memory to store data acquired by the ADC.

Problem Statement

The goal of this lab is to use the microcontroller as a data acquisition device. The program you write will have two modes: data acquisition mode, and data output mode. In data acquisition mode, the program will collect a total of 30 data samples from the ADC at 1 sec intervals and store it in flash memory. In data output mode, the program will print out the stored data from flash memory into the putty terminal window.

Background

Flash Memory:

The memory space on the MSP432 is a 4GB address space divided into 512 MB zones. The first 512MB zone is called the Code zone. The Code zone contains the ROM, SRAM and Flash memory regions. The flash memory is a 4MB region that starts at address 0x0000_0000, i.e. at the beginning of the entire memory region. This flash memory is then divided into Main Memory, Information Memory and Reserved Memory. The main memory starts at address 0x0000_0000. It is a 2048KB region that holds the code and data for user applications.

Main flash memory is further divided into two parts, called Banks, each of 1MB. Both banks are further subdivided into 4KB memory spaces or sectors. Each sector is 4096 Bytes (4KB).

Each sector in main flash memory can be individually programmed and erased. Driverlib API's are available for programming and erasing flash memory sectors, as discussed in the lecture on flash memory. The output of these API's is true if the operation is successful and false if the operation fails. If the operation fails, it is a good practice to trap the microcontroller in an infinite loop which does nothing. It is rare that failure occurs but nevertheless this is good practice.

Sectors must first be unprotected in order to be erased and programmed. Likewise, sectors should be protected once they are programmed. See the class lecture on flash memory for more details.

Hardware:

The circuit assembled/built in Lab 1 should be interfaced with the MSP432 similar to the setup in Lab 5. Adjust the potentiometer used to tune the gain of the op-amp to set the gain such that the output voltage is always less than 3.3V for the full temperature range. Connect the output of the temperature sensor signal conditioning circuit to the pin corresponding to the ADC module used. Furthermore, **connect a ground wire from the signal conditioning circuit to the MSP432.**

Software:

Your program should proceed in the following steps:

1. First, you should wait in a loop until the user presses one of the buttons. Pressing button S1 should place the system in data acquisition mode, while pressing S2 should place the system in data output mode.

2. When the button is pressed, the mode is selected. If data acquisition mode is selected, the MCU should collect ADC measurements from the temperature sensor at a rate of 1 Hz (as in Lab 5). It should store the data in a standard array of floating point values. After it has collected a total of 30 measurements, it should

load (program) the collected data into flash memory. Program flash memory at the memory address `0x1FE000`. After programming is complete, the program should then enter an infinite loop which does nothing, and an LED can light up.

3. If the user chooses data output mode, the data from memory location `0x1FE000` (which points to the flash memory location where you wrote your data) should be printed to the putty terminal via the UART interface from Lab 4. Use the `sprintf()` function to convert the stored binary numbers into Ascii values. When you print the data, print the temperature as a floating point number with one decimal place (this can be achieved using the format descriptor "%.1f"). After you print the data, enter an infinite while loop that does nothing.

Note: If you reprogram the board between acquiring the data and printing it out, all data in flash memory will be lost (since the programmer erases it). Instead, use the RST button on the board (S3) to reboot it when you want to switch modes.

CCS Project Setup and MSP432P411-specific Items:

The MSP432P4111 board we are using has a different flash memory controller than the MSP432P401R used in the lecture videos and previous sessions of the course. The MSP432P4111 will require a new version of driverlib to program flash memory. Below are the instructions you will need to complete the lab assignment with the MSP432P4111 – please follow them carefully:

1. Download the source.zip file from the Lab 6 folder under the Lab Resources folder on Canvas. Unzip this file somewhere on your computer. This contains an updated driverlib package (from the one we have used in previous labs). Also download the driverlib User Guide from the Lab 6 folder on Canvas. This is the new user guide for this updated version. **We will only use this updated driverlib version for Lab 6, not for the remaining labs in the course.**
2. Start a new CCS project for Lab 6. When creating the new project, you should select the target device as MSP432P4111. This is <u>different from previous labs</u> when you specified the target device as MSP432P401R.
3. Right click on the CCS project name and select Properties. In the Properties tab, select Paths and Symbols from the C/C++ General tab. Select the Includes button to add a new include path. Click Add…. Click File System from the box that comes up, and browse to where you unzipped the files from Canvas. Select the "source" folder, then click Open and then Okay. This will add the Include path to the new driverlib package provided to you for Lab 6. Note that you should select the "source" folder as the include path, not the "driverlib" folder or any other folder farther down the folder tree.
4. Before closing anything, click on the Libraries button. Click Add… and then File System again. Browse to the "source" folder, then to ti->devices->msp432p4xx->driverlib->ccs. Then click on the file msp432p4xx_driverlib.lib. Then click Open and then Okay.
5. Click on the Library Paths button. Click Add…, then File System, then browse to the "source" folder. Then click Open and Okay. You have now completed the project setup in CCS.

Note: If your CCS version does not have the dialog boxes described in Steps 3-5, follow the alternative steps below. If Steps 3-5 were successful, proceed to Step 6 below.

Alternative Step 3: Right click on the CCS project name and select Properties. In the Properties tab, select *Build ->Arm Compiler -> Include Options*. Click Add…, then click Browse, and browse to where you unzipped the files from Canvas. Select the "source" folder, then click Open and then Okay. This will add the Include path to the new driverlib package provided to you for Lab 6. Note that you should select the "source" folder as the include path, not the "driverlib" folder or any other folder farther down the folder tree.

Alternative Step 4: Before closing anything, click Add… again, then click Browse and browse to the "source" folder, then to ti->devices->msp432p4xx->driverlib. Then click Open and then Okay. There should now be the path to "source/" and the path to "source/ ti/devices/msp432p4xx/driverlib" in your include path.

Alternative Step 5: While still in the Properties tab, select *Build -> Arm Linker -> File Search Path -> Include library file or command file as input*. Click the Add button and browse to source->ti->devices->msp432p4xx->driverlib->ccs. Then click on the file msp432p4xx_driverlib.lib. You have now completed the project setup in CCS.

6. Create your code according to the instructions above. You should create a macro for the flash address you will be writing to as follows:

   ```
   #define FLASH_ADDR 0x1FE000
   ```

7. Instead of the functions shown in class, you should use the FlashCtl_A functions in driverlib. The functions you will need are:

   ```
   FlashCtl_A_unprotectMemory(…);
   
   FlashCtl_A_eraseSector(…)
   
   FlashCtl_A_programMemory(…)
   
   FlashCtl_A_protectMemory(…)
   ```

   Note that we will use the unprotectMemory function instead of the unprotectSector function shown in class. When calling the unprotectMemory memory, you should give `FLASH_ADDR` as both the first and second argument. This will just erase one 4096 byte segment starting at `FLASH_ADDR`. You should give this argument to all of the four functions above as the memory location which should be unprotected, erased, programmed, and protected.

8. The new driverlib package has a few different macros and structs for UART configuration and usage. When declaring the UART configuration struct, you should declare it with type `eUSCI_UART_ConfigV1` rather than `eUSCI_UART_Config` (which is what you used in Lab 4). This new configuration struct has an additional element at the end that should be set to `EUSCI_A_UART_8_BIT_LEN`. This is documented in the new driverlib user's guide provided for this lab which you downloaded from Canvas in Step 1.
9. When sending out data from UART, in Lab 4 you used the macro `UCA0IFG` for the transmit interrupt flag. With the new version of driverlib you are using for this lab, this macro should be changed to `EUSCI_A_UART_TRANSMIT_INTERRUPT_FLAG`.

Tips:

- Make sure to set the DCO and SMCLK to the appropriate rate depending on which mode you are in. You can switch DCO and SMCLK frequency at any time. Set up the clocks after the user selects the mode.
- Place the ADC in Manual Iteration mode (rather than Automatic Iteration mode) and place the toggle conversion function inside the timer ISR.
- After you reach 30 samples and are ready to write to flash memory, disable all interrupts to avoid the ADC continuing to sample.
- Store your data in flash memory as an array of floating point numbers.
- The memory address where we will be storing data should be defined in a macro as follows (as specified in step 6 above):

  ```
  #define FLASH_ADDR 0x1FE000
  ```

- The only interrupt you should use in this program is the Timer A capture compare interrupt, as in the previous ADC lab.

Requirements

1. Successfully demonstrate the outcome of the program and all the required functionalities to TAs or instructor.

2. Submit the commented final version of the code on Canvas.

3. Answer the below questions and submit the typed report on Canvas.

Questions

1. Compare the use of the internal DCO vs using a Crystal oscillator for sourcing a clock. For what applications would you want to use a crystal rather than a DCO, and vice versa?

2. Explain what is the difference between MCLK, SMCLK and ACLK.

3. What are the advantages of Timer A compared to SysTick?