ADC/Timer-Counter/Interrupt

Introduction

A. This document is an overview to setup a simple inter-system flow using the ADC, CPU Timer, and interrupts specifically using driverlib for F23879D processor

B. Understand the capabilities of the ADC/Timer/Interrupts systems far expand what will be covered in this lab/project

C. Using the driverlib for this lab is optional, to use the provided functions:

   a. Unzip the ADC_TIMER_DRIVERLIB.zip

   b. Copy driverlib folder into your project

   c. Replace the 'F2837xD_Adc.c' in your common/source with the provided file

   d. Replace the 'F2837xD_Adc_defines.h' in your common/include with the provided file

   e. Ensure to include the driverlib folder into your project properties -> include options

   f. You will need to include each adc.h, cputimer.h, and interrupt.h in your main code

   g. The code in the examples folder will not work with the provided files, however, it is useful to see how the functions are used with the various driverlib defines

ADC System for software trigger on ADCA SOC 0 for ADCINA0

A. Skim through ADC system in tech manual. Starts on page 1391.

   a. Pay close attention to the ADCCTL1, ADCCTL2, ADCSOCFRC1, and ADCSOC0CTL registers as these will be necessary for setting up the ADC System and the SOC0

B. ADC Registers

```
AdcaRegs.ADCCTL2.all = 6;
AdcaRegs.ADCCTL2.bit.PRESCALE = 6; //divide main clock by 4
```

   1. ADC registers can be accessed by specific bits or entirely as shown above.

C. Configuration for a software trigger SOC

   1. The prescaler of the ADC clock is used to determine the acquisition period of conversion for a specific channel. The prescaler directly divides down the system clock into the ADC clock

      a. If using driverlib, the function ADC_setPrescaler can be used

2. Set the mode of your ADC module
    a. Lab 5 just requires single-ended mode which is 12-bit resolution
    b. If using driverlib, the function ADC_setMode can be used
3. Enable the ADC
    a. If using driverlib, the function ADC_enableConverter can be used
4. Perform a millisecond delay to allow the ADC system time to boot up
5. Setup the SOC to convert with a software trigger for channel 0 and configure the number of ADC clock pulses for the acquisition window
    a. The driverlib function ADC_setupSOC can be used for this

CPU Timer1

A. Read the tech manual about timer-counter and interrupt systems
    1. CPU timers start on page 117
    2. Interrupts start on page 89
    3. CPU timer registers start on page 161
    4. PIE ctrl regs start on page 167
B. Setting up CPU timer1 to interrupt (special as it is the only interrupt in group 13)
    1. InitCpuTimers(); is a controlsuite function that will initialize the timer structs and setup the timers in their default states
    2. Setup the CPUtimer0 for a 0.1 S delay
        a. ConfigCpuTimer(Timer,Freq,Period) is a controlsuite function that will configure the timer with a specific period based on system frequency in MHz
        b. CPUTimer_setPeriod, CPUTimer_setPreScaler functions can be used in driverlib
    3. Enable inteerupt group 13 in the Interrupt Enable Register (IER)
        a. Interrupt_enable(INT_TIMER1); can be used with driverlib
    4. Place a pointer to your interrupt service routine(ISR) in the PIE vector table
        a. Interrupt_register(INT_TIMER1,handler) can be used with driverlib
    5. Enable interrupts by setting the ENPIE bit on the PIE control registers

a. Interrupt_initModule() can be used with driverlib

6. Enable global interrupts by using the “EINT;” macro function
   a. Interrupt_enableMaster() can be used with driverlib
7. Start the timer by setting the TSS bit on the timer control register
   a. CPUTimer_startTimer(CPUTIMER1_BASE) can be used to start the timer in driverlib

```
131
132 interrupt void TIMER0_INT(){
133
134 }
135
```

ISR SHELL. Pass &TIMER0_INT into PIE vector table (yours should use timer 1

TIPS:

- Driverlib has all the defines for inputs to functions(and many functions) in its respective header files: check adc.h, cputimer.h, interrupt.h
- Ensure to acknowledge the group 13 interrupt inside of your ISR: Interrupt_clearACKGroup(INTERRUPT_CPU_INT1); can be used with driverlib