Experiment 12

Analog Interfacing

Lab Objective

In this lab, we will learn to configure analog-to-digital converter (ADC) and sample sequencers available in LM4F120 based Stellaris Launchpad. We will measure the temperature using on-chip temperature sensor and display the result on a 7-segment display.

ADC in LM4F120

LM4F120 MCU contains two identical 12-bit ADC modules with a capability of 12 shared input channels and hardware averaging upto 64 samples. Each module is controlled by a number of registers and offers a variety of options. Both the modules have four sequencers. In this lab, we will use sequencer 3 since it captures only one sample per trigger and stores the sample into the corresponding FIFO. The sampling rate can be varied from 125 KSPS to 1 MSPS.

ADC Initialization

In this experiment, we will be using ADC0 and sample sequencer 3 to sample the data of on-chip temperature sensor and display it on seven segment module. The initialization and configuration instructions for the ADC module can be found on pg. 776 of the datasheet. A brief description of these steps is given below. For further details datasheet must be consulted.

1. As with all peripherals, first initialization step is to enble the clock signal for the appropriate GPIO pin and ADC module using RCGCGPIO and RCGCUART registers respectively. In this experiment, we are using on-chip temperature sensor so, we don't need to enable the clock signal for any analog input (AIN) pin.

2. Enable the alternate functionality by asserting the appropriate bits in GPIOAFSEL register for the analog signal. Also configure the pin as an input by clearing the respective bits in GPIODEN register.

3. Disable the analog isolation circuit by asserting the appropriate bits for the respective analog input (AIN) pins in GPIOAMSEL register.

In this experiment we can skip the above mentioned steps to configure a GPIO pin as an analog input because we will be using on-chip temperature sensor as analog input. Now, we discuss the steps to configure sample sequencer.

1. Set the sampling rate for the ADC by writing appropriate value to the ADCPC register.
2. Disable the sample sequencer by asserting the corresponding bit in ADCACTSS register. We disable all the peripherals before configuration to avoid any erroneous execution which may result in unintended results.
3. Select the trigger event from ADCEMUX register. ADC in LM4F120 MCU provides four different options of software, analog comparator, timer or GPIO which are completely programmable for each sample.
4. Configure the corresponding input source in the ADCSSMUXn register for each sample in sample sequence. For example, to configure AIN3 as an input source for the first sample in sample sequencer 3 we should write 0x03 to the lower nibble of ADCSSMUX3 register.
5. Write the corresponding nibble in ADCSSCTL register for configuring sample control bits for each sample in sample sequencer. In the last sample, END bit must be asserted to mark the end of analog sample otherwise ADC may exhibit unpredictable behaviour. To use internal temperature sensor ensure that TS bit is asserted in the corresponding nibble.
6. If interrupts are to be used, corresponding MASK bit should be asserted in ADCIM register. In this experiment, we will not use interrupts so 0 should be written to this bit.
7. After configuring the sample sequencer with the required parameters now activate the sample sequencer by writing 1 to corresponding bit of the sample sequencer.

After configuration ADC is triggered by writing 0x0008 to ADC0PSSI register. When the conversion is complete, bit 3 of ADC0RIS register is set to 1 automatically by the hardware. So, by polling this bit, we will know when the conversion is done and the result is ready. Then, the 12-bit ADC result is read from ADC0_SSFIFO3 register.

Source Code

```
// Register definitions for clock enable
#define SYSCTL_RCGCGPIO_R        (*((volatile unsigned long *)0x400FE608))
#define SYSCTL_RCGCADC_R         (*((volatile unsigned long *)0x400FE638))

// Register definitions for GPIO port A
#define  GPIO_PORTA_DATA_R       (*((volatile unsigned long *)0x400043FC))
#define  GPIO_PORTA_DIR_R        (*((volatile unsigned long *)0x40004400))
#define  GPIO_PORTA_DEN_R        (*((volatile unsigned long *)0x4000451C))

// Register definitions for GPIO port B
#define  GPIO_PORTB_DATA_R       (*((volatile unsigned long *)0x400053FC))
#define  GPIO_PORTB_DIR_R        (*((volatile unsigned long *)0x40005400))
#define  GPIO_PORTB_DEN_R        (*((volatile unsigned long *)0x4000551C))

// Register definitions for GPIO port F
```

```
#define  GPIO_PORTF_DATA_R      (*((volatile unsigned long *)0x400253FC))
#define  GPIO_PORTF_DIR_R       (*((volatile unsigned long *)0x40025400))
#define  GPIO_PORTF_DEN_R       (*((volatile unsigned long *)0x4002551C))

//Register definitions for ADC0 and Sample Sequencer 3
#define ADC0_PC_R               (*((volatile unsigned long *)0x40038FC4))
#define ADC0_SSPRI_R            (*((volatile unsigned long *)0x40038020))
#define ADC0_ACTSS_R            (*((volatile unsigned long *)0x40038000))
#define ADC0_IM_R               (*((volatile unsigned long *)0x40038008))
#define ADC0_RIS_R              (*((volatile unsigned long *)0x40038004))
#define ADC0_ISC_R              (*((volatile unsigned long *)0x4003800C))
#define ADC0_SAC_R              (*((volatile unsigned long *)0x40038030))
#define ADC0_PSSI_R             (*((volatile unsigned long *)0x40038028))
#define ADC0_SSCTL3_R           (*((volatile unsigned long *)0x400380A4))
#define ADC0_SSFIFO3_R          (*((volatile unsigned long *)0x400380A8))

unsigned char Lookup_7Seg_Disp[12] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,
                                0x92, 0x82, 0xF8, 0x80, 0x90, 0xC6};
unsigned char Temperature_Value[3] = {0, 0, 0xA};
unsigned char i, value=0;
unsigned int ADC_value = 0, temperature = 0;

void ADC_Init (){

  volatile unsigned long delay;

  //Enable clock

  //Enable clock for GPIO port A,B and F [pg. 314]
  SYSCTL_RCGCGPIO_R |= 0x23;
  //Enable clock for ADC0 [pg. 326]
  SYSCTL_RCGCADC_R |= 0x01;
  //Delay for clock signal to settle down
  delay = SYSCTL_RCGCADC_R;

  //GPIO A,B,F enable

  //Set the latch enable pins of 7 segment as output [pg. 622]
  GPIO_PORTA_DIR_R |= 0x38;
  //Set the digital operation for GPIO port A [pg. 641]
  GPIO_PORTA_DEN_R |= 0x38;

  GPIO_PORTB_DIR_R |= 0xFF;
  GPIO_PORTB_DEN_R |= 0xFF;
  GPIO_PORTB_DATA_R |= 0xFF;

  GPIO_PORTF_DIR_R |= 0x08;
  GPIO_PORTF_DEN_R |= 0x08;
  GPIO_PORTF_DATA_R |= 0x08;

  //ADC0 configure

  //Clear the sample rate [pg. 847]
  ADC0_PC_R &= 0x00;
  //Set sample rate equal to 125 ksps
  ADC0_PC_R |= 0x01;
  //Set the priority of sample sequencers.
  //Sample sequence 0 has highest and SS3 has lowest priority [pg. 798]
  ADC0_SSPRI_R |= 0x3210;
  //Disable sample sequence 3 befor configuration [pg. 780]
  ADC0_ACTSS_R &= ~0x08;
  //Enable TS0, IE0 and END0 bits [pg. 833]
  ADC0_SSCTL3_R |= 0x0E;
  //Enable 16x hardware oversampling [pg. 804]
  ADC0_SAC_R |= 0x4;
  //Disable Interrupt by writing 0 to corresponding bit [pg. 784]
  ADC0_IM_R &= ~0x08;
  //Activate sample sequencer
  ADC0_ACTSS_R |= 0x08;

}

void ADC_Temperature(void){

  //Software trigger [pg. 802]
  ADC0_PSSI_R |= 0x08;

  //Poll the status of corresponding RIS bit to wait
  //for the end of conversion [pg. 782]
  while((ADC0_RIS_R & 0x08)==0);
  //After conversion fetch the data from the FIFO [pg. 817]
  ADC_value = (ADC0_SSFIFO3_R & 0xFFF);
  //Calculate the value of temperature [pg. 771]
  temperature = (unsigned char) (147.5 - (ADC_value *3.3*75)/4096);

  Temperature_Value[1] = temperature%10;
  Temperature_Value[0] = temperature/10;

  //Clear RIS to start coversion again [pg. 787]
  ADC0_ISC_R |= 0x08;
}

//Function to introduce delay
void Delay(unsigned long count){
  int i;
  for (i = 0; i < count; i++)
  { }
}

int main (){

  //Initialize ADC and GPIO ports for seven segment display
  ADC_Init();

  while(1)
  {
    for(i = 0; i <= 2; i++)
    {
      value = Temperature_Value[i];
      GPIO_PORTB_DATA_R = Lookup_7Seg_Disp[value];

      // Select the digit on Port A
      GPIO_PORTA_DATA_R = (0x38 - (1 << (i+3)));
      Delay(1000);
    }
      ADC_Temperature();
  }
}
```