**11/10/2020**

Joshua Frazer

Lab 10

Analog to Digital Converter (ADC)

**Introduction:**

With this lab, we learn a lot about the Analog to Digital Converter (ADC). For part 1, we focused more on the configuration of the ADC for the horizontal axis of the joystick. Part 2 was similar to Part 1, but instead of just one axis, we had to configure the ADC for the horizontal and vertical axis of the joystick.

**Part 1:**

This part, like most other lab, required a lot of reading and understanding. Once I took in all the information given, it was quite simple to combine all that knowledge and finish the task being asked. The only difficulty I received from this part was configuring certain fields for the ADC and to be honest, I forgot to call the Initialize\_ADC(); function. Once I found my mistake, this lab was straightforward.

**#include** <msp430FR6989.h>

**#define** FLAGS UCA1IFG // Contains the transmit & receive flags

**#define** RXFLAG UCRXIFG // Receive flag

**#define** TXFLAG UCTXIFG // Transmit flag

**#define** TXBUFFER UCA1TXBUF // Transmit buffer

**#define** RXBUFFER UCA1RXBUF // Receive buffer

**#define** redLED BIT0 // Red LED at P1.0

**void** **Initialize\_ADC**();

**void** **Initialize\_UART**(**void**);

**void** **uart\_write\_char**(**unsigned** **char** ch);

**void** **uart\_write\_uint16**(**unsigned** **int** n);

**void** **uart\_write\_string**(**char** \*str);

// Booster Pack: HOR(X): J1.2 --> Horizontal direction: Jumper 1 pin 2

// Booster Pack: HOR(Y): J3.26 --> Vertical direction: Jumper 1 pin 26

// Launchpad: Horizontal J1.2: A10/P9.2 --> Analog input 10 / Port 9.2

// Launchpad: Vertical J3.26: A4/P8.7 --> Analog input 4 / Port 8.7

// A10 functionality: P9DIR=x, P9SEL1=1, P9SEL0=1

// A4 functionality: P8DIR=x, P8SEL1=1, P8SEL0=1

// 10.1 Equation for Sample-and-Hold Time

// (10kOhm + 10kOhm)\*(15pF+1pF)\*ln(2^(3+2)) = 3 microseconds

**int** **main**(**void**)

{

**volatile** **unsigned** **int** i;

**unsigned** **int** x = 0;

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

PM5CTL0 &= ~LOCKLPM5; // Enable the GPIO pins

P1DIR |= redLED; // Direct pin as output

P1OUT &= ~redLED; // Turn LED Off

Initialize\_ADC();

Initialize\_UART();

**while**(1)

{

ADC12CTL0 |= ADC12SC;

**while** (ADC12CTL1 & ADC12BUSY == 1) {}

x = ADC12MEM0;

uart\_write\_uint16(x);

uart\_write\_char('\n');

uart\_write\_char('\r');

P1OUT ^= redLED;

**for** (i = 0; i < 45000; i++) {}

}

}

**void** **Initialize\_ADC**() {

// Divert the pins to analog functionality

// X-axis: A10/P9.2, for A10 (P9DIR=x, P9SEL1=1, P9SEL0=1)

P9SEL1 |= BIT2;

P9SEL0 |= BIT2;

// Turn on the ADC module

ADC12CTL0 |= ADC12ON;

// Turn off ENC (Enable Conversion) bit while modifying the configuration

ADC12CTL0 &= ~ADC12ENC;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL0\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12SHT0 (select the number of cycles that you determined)

ADC12CTL0 |= ADC12SHT0\_3;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12SHS (select ADC12SC bit as the trigger)

// Set ADC12SHP bit

// Set ADC12DIV (select the divider you determined)

// Set ADC12SSEL (select MODOSC)

ADC12CTL1 |= ADC12SHS\_0 | ADC12DIV\_0 | ADC12SSEL\_3 | ADC12SHP;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12RES (select 12-bit resolution)

// Set ADC12DF (select unsigned binary format)------

ADC12CTL2 |= ADC12RES\_1;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Leave all fields at default values

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12MCTL0\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12VRSEL (select VR+=AVCC, VR-=AVSS)

// Set ADC12INCH (select channel A10)

// Turn on ENC (Enable Conversion) bit at the end of the configuration

ADC12MCTL0 |= ADC12VRSEL\_0 | ADC12INCH\_10;

ADC12CTL0 |= ADC12ENC;

**return**;

}

**void** **Initialize\_UART**(**void**){

// Divert pins to UART functionality

P3SEL1 &= ~(BIT4|BIT5);

P3SEL0 |= (BIT4|BIT5);

// Use SMCLK clock; leave other settings default

UCA1CTLW0 |= UCSSEL\_2;

// Configure the clock dividers and modulators

// UCBR=6, UCBRF=13, UCBRS=0x22, UCOS16=1 (oversampling)

UCA1BRW = 6;

UCA1MCTLW = UCBRS5|UCBRF3|UCOS16;

// Exit the reset state (so transmission/reception can begin)

UCA1CTLW0 &= ~UCSWRST;

}

**void** **uart\_write\_string**(**char** \*str)

{

**volatile** **unsigned** **int** i;

**for** (i = 0; i<strlen(str); i++)

{

uart\_write\_char(str[i]);

}

}

**void** **uart\_write\_uint16**(**unsigned** **int** n){

**unsigned** **int** r[6] = {0};

**int** i = 0, j;

**if** (n == 0)

uart\_write\_char('0');

**while** (n != 0)

{

r[i++] = n % 10;

n /= 10;

}

**for** (j = i-1; j >= 0; --j)

uart\_write\_char(r[j] + '0');

}

**void** **uart\_write\_char**(**unsigned** **char** ch){

// Wait for any ongoing transmission to complete

**while** ( (FLAGS & TXFLAG)==0 ) {}

// Write the byte to the transmit buffer

TXBUFFER = ch;

}

**Questions:**

1. What are the values of the ADC’s RI and CI? If these values have a range show the range. Did you use the lower or upper range of these values? Justify your choice.
   1. **CI: Nominal = 10pF, Max = 15 pF**
   2. **RI:   
      When greater than 2V:   
      Nominal = 0.5kΩ, Max 4kΩ  
      When less than 2V:  
      Nominal = 1kΩ, Max 10kΩ**
   3. **I chose the maximum for both CI and RI because we wanted to hit all possible outcomes.**
2. What is the minimum sample-and-hold time? Show how you computed this duration.
   1. **(10kOhm + 10kOhm)\*(15pF+1pF)\*ln(2^(3+2)) = 3 microseconds**
3. What divider of MODOSC did you use? How many cycles did you set the sample-and-hold time? Show your computation.
   1. **I used the first divider of MODOSC (/1). I set the cycles to 32 ADC12CLK cycles.**
4. Observe the values returned by the ADC when the joystick is in the center, all the way to the left and all the way to the right. Interpret the results and indicate if they are correct.
   1. **They are correct. Left = ~470, Right = ~4095**

**Part 2:**

After everything in part 1 is configured correctly, this part is rudimentary. All that is needed is to add the vertical axis and configure the registers once more for the ADC.

**#include** <msp430FR6989.h>

**#define** FLAGS UCA1IFG // Contains the transmit & receive flags

**#define** RXFLAG UCRXIFG // Receive flag

**#define** TXFLAG UCTXIFG // Transmit flag

**#define** TXBUFFER UCA1TXBUF // Transmit buffer

**#define** RXBUFFER UCA1RXBUF // Receive buffer

**#define** redLED BIT0 // Red LED at P1.0

**void** **Initialize\_ADC**();

**void** **Initialize\_UART**(**void**);

**void** **uart\_write\_char**(**unsigned** **char** ch);

**void** **uart\_write\_uint16**(**unsigned** **int** n);

**void** **uart\_write\_string**(**char** \*str);

// Booster Pack: HOR(X): J1.2 --> Horizontal direction: Jumper 1 pin 2

// Booster Pack: HOR(Y): J3.26 --> Vertical direction: Jumper 1 pin 26

// Launchpad: Horizontal J1.2: A10/P9.2 --> Analog input 10 / Port 9.2

// Launchpad: Vertical J3.26: A4/P8.7 --> Analog input 4 / Port 8.7

// A10 functionality: P9DIR=x, P9SEL1=1, P9SEL0=1

// A4 functionality: P8DIR=x, P8SEL1=1, P8SEL0=1

// 10.1 Equation for Sample-and-Hold Time

// (10kOhm + 10kOhm)\*(15pF+1pF)\*ln(2^(3+2)) = 3 microseconds

**int** **main**(**void**)

{

**volatile** **unsigned** **int** i;

**unsigned** **int** x = 0, y = 0;

**char** xaxis[] = "X-Axis: ", yaxis[] = "Y-Axis: ";

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

PM5CTL0 &= ~LOCKLPM5; // Enable the GPIO pins

P1DIR |= redLED; // Direct pin as output

P1OUT &= ~redLED; // Turn LED Off

Initialize\_ADC();

Initialize\_UART();

**while**(1)

{

ADC12CTL0 |= ADC12SC;

**while** (ADC12CTL1 & ADC12BUSY == 1) {}

x = ADC12MEM0;

y = ADC12MEM1;

uart\_write\_string(xaxis);

uart\_write\_uint16(x);

uart\_write\_char('\n');

uart\_write\_char('\r');

uart\_write\_string(yaxis);

uart\_write\_uint16(y);

uart\_write\_char('\n');

uart\_write\_char('\r');

uart\_write\_char('\n');

P1OUT ^= redLED;

**for** (i = 0; i < 45000; i++) {}

}

}

**void** **Initialize\_ADC**() {

// Divert the pins to analog functionality

// X-axis: A10/P9.2, for A10 (P9DIR=x, P9SEL1=1, P9SEL0=1)

P9SEL1 |= BIT2;

P9SEL0 |= BIT2;

// Y-axis: A4/P8.7, for A4 (P8DIR=x, P8SEL1=1, P8SEL0=1)

P8SEL1 |= BIT7;

P8SEL0 |= BIT7;

// Turn on the ADC module

ADC12CTL0 |= ADC12ON;

// Turn off ENC (Enable Conversion) bit while modifying the configuration

ADC12CTL0 &= ~ADC12ENC;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL0\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set the bit ADC12MSC (Multiple Sample and Conversion)\*

// Set ADC12SHT0 (select the number of cycles that you determined)

ADC12CTL0 |= ADC12SHT0\_3 | ADC12MSC;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12CONSEQ (select sequence-of-channels)\*

// Set ADC12SHS (select ADC12SC bit as the trigger)

// Set ADC12SHP bit

// Set ADC12DIV (select the divider you determined)

// Set ADC12SSEL (select MODOSC)

ADC12CTL1 |= ADC12SHS\_0 | ADC12DIV\_0 | ADC12SSEL\_3 | ADC12SHP | ADC12CONSEQ\_1;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12RES (select 12-bit resolution)

// Set ADC12DF (select unsigned binary format)

ADC12CTL2 |= ADC12RES\_1;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12CTL3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12CSTARTADD to 0 (first conversion in ADC12MEM0)\*

// Leave all fields at default values

ADC12CTL3 |= ADC12CSTARTADD\_0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12MCTL0\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12VRSEL (select VR+=AVCC, VR-=AVSS)

// Set ADC12INCH (select channel A10)

ADC12MCTL0 |= ADC12VRSEL\_0 | ADC12INCH\_10;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ADC12MCTL1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Set ADC12VRSEL (select VR+=AVCC, VR-=AVSS)\*

// Set ADC12INCH (select the analog channel that you found)\*

// Set ADC12EOS (last conversion in ADC12MEM1)\*

ADC12MCTL1 |= ADC12VRSEL\_0 | ADC12INCH\_4 | ADC12EOS;

// Turn on ENC (Enable Conversion) bit at the end of the configuration

ADC12CTL0 |= ADC12ENC;

**return**;

}

**void** **Initialize\_UART**(**void**){

// Divert pins to UART functionality

P3SEL1 &= ~(BIT4|BIT5);

P3SEL0 |= (BIT4|BIT5);

// Use SMCLK clock; leave other settings default

UCA1CTLW0 |= UCSSEL\_2;

// Configure the clock dividers and modulators

// UCBR=6, UCBRF=13, UCBRS=0x22, UCOS16=1 (oversampling)

UCA1BRW = 6;

UCA1MCTLW = UCBRS5|UCBRF3|UCOS16;

// Exit the reset state (so transmission/reception can begin)

UCA1CTLW0 &= ~UCSWRST;

}

**void** **uart\_write\_string**(**char** \*str)

{

**volatile** **unsigned** **int** i;

**for** (i = 0; i<strlen(str); i++)

{

uart\_write\_char(str[i]);

}

}

**void** **uart\_write\_uint16**(**unsigned** **int** n){

**unsigned** **int** r[6] = {0};

**int** i = 0, j;

**if** (n == 0)

uart\_write\_char('0');

**while** (n != 0)

{

r[i++] = n % 10;

n /= 10;

}

**for** (j = i-1; j >= 0; --j)

uart\_write\_char(r[j] + '0');

}

**void** **uart\_write\_char**(**unsigned** **char** ch){

// Wait for any ongoing transmission to complete

**while** ( (FLAGS & TXFLAG)==0 ) {}

// Write the byte to the transmit buffer

TXBUFFER = ch;

}

**Questions:**

1. Turn the joystick in the four main dimensions (left, right, top, bottom) and combinations thereof. Interpret the results and indicate if they are correct.
   1. **Yes, they are correct.  
      Left:  
      X-Axis = ~440  
      Y-Axis = ~1670  
      Right:  
      X-Axis = ~4090  
      Y-Axis = ~2050  
      Top:  
      X-Axis = ~1935  
      Y-Axis = ~4090  
      Bottom:  
      X-Axis = ~1935  
      Y-Axis = ~490**

**Student Q&A:**

1. How many cycles does it take the ADC to convert a 12-bit result? (look in the configuration register that contains ADC12RES).
   1. **14 Clock cycles.**
2. The conversion time you found in the previous question does not include the sample-and-hold time. Find the total conversion time of your setup (sample-and-hold time and conversion time). Give the total cycles and the duration.
   1. **30 Clock cycles and 3 microseconds total duration.**
3. In this experiment, we set our reference voltages VR+=AVCC (Analog Vcc) and VR−=AVSS (Analog Vss). What voltage values do these signals have? Look in the MCU datasheet(slas789c) in Table 5.3. Assume that Vcc=3.3V and Vss=0
   1. **The voltages would be 3.3V for Vcc and 0V for Vss.**
4. It’s possible for the ADC12B module to use reference voltages that are generated by the module REFA (Reference A). What voltage levels does REFA provide? (look in theFR6xx Family User’s Guide on p. 859 and p. 869).
   1. **REFA provides 1.2V, 2.0V, and 2.5V.**

**Conclusion:**

This lab was surprisingly simple, with only having two parts. The usual part 1 carried most of the information, you can obviously see that this is the most significant part. After doing this lab, I got a bigger understanding on how much configurations are needed to enable the ADC. Honestly, I am quite mind blown after this realization.