

Lab02 - ADC

Introduction to Embedded Systems - University of Nebraska

Dat Nguyen

Contents

1	Introduction	3
2	Program Description	3
2.1	Program 1 - Measure ADC Value and Voltage	3
2.2	Program 2 - ADC Conversion Time Measurement	3
2.3	Program 3 - Read Voltage From AD0	4
3	Summary	4
4	Appendix	4
4.1	Program 1	4
4.2	Program 2	5
4.3	Program 3	5

1 Introduction

This lab provides hand-on activity and insight of Analog to Digital Converter(ADC) usages.

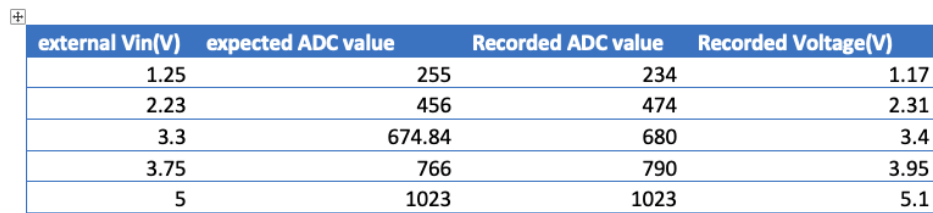
2 Program Description

2.1 Program 1 - Measure ADC Value and Voltage

1. Different voltages are produced using a voltage divider circuit with an 5.1 reference voltage. Figure 1 show a table that capture externally measured voltage V_{in} , expected ADC value, recorded ADC value using Arduino board, and the voltage the recorded ADC value represents. The ADC0 reading logic is encapsulated in the function readADC0() in the code and implemented as following

```

1  uint16_t readADC0() {
2      uint16_t result;
3      *pADCSRA = (*pADCSRA) | 0x40; // start conversion
4      while (((*pADCSRA) & 0x40) == 0x40) { // while still conversion
5          // do nothing. waiting ADSC bit to be cleared by hardware
6      }
7
8      result = *pADCL; // lower 8bits of ADC conversion (0x00??)
9      result = result | (((uint16_t)*pADCH) << 8);
10     result = result & 0x3FFF; // clear bits 15-10 (ADC res is only 10 bits)
11     return result;
12 }
```



external Vin(V)	expected ADC value	Recorded ADC value	Recorded Voltage(V)
1.25	255	234	1.17
2.23	456	474	2.31
3.3	674.84	680	3.4
3.75	766	790	3.95
5	1023	1023	5.1

Figure 1: Table capture list of voltage and corresponding ADC value

2. The V_{ref} measured is 5.1V. This value exceed the expected value of 5v.
3. The measurements don't match my expectations. The ADC value displayed for 5v is still correct since the ADC precision is only 10 bits. For other voltages, the error is fluctuate between 0.1 - 0.2 causing the ADC value to off by 10-20.

2.2 Program 2 - ADC Conversion Time Measurement

1. The conversion time is measured by continuously trigger PB0 high, read ADC0 value, then set PB0 back to low. When analyzing the signal from PB0 we can measure the high duty cycle to determine the time it take to do the ADC0 conversion. The readADC0() method logic is similar to experiement 01. Figure 2 shows the O-scope capture of PB0 pin. A soft delay of 1ms is added and implemented similar to experiement03 in Lab01.
2. The value measured is as expected. According to the data sheet, ADC reading takes 13 clock cycles. Since our ADC prescaler is configure as $\frac{16Mhz}{128} = 125kHz$, therefore each

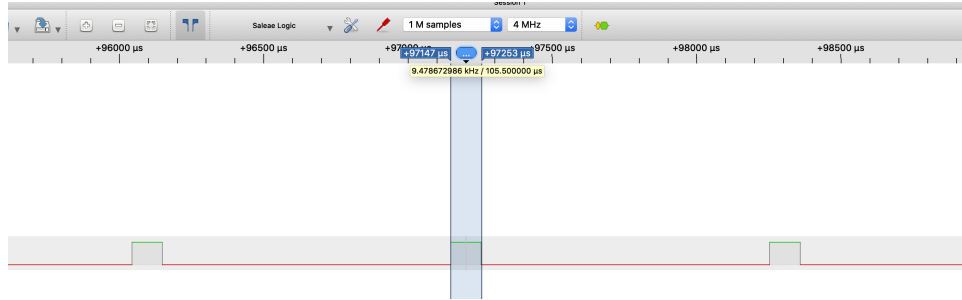


Figure 2: Table capture PB0 sign

cycle takes $\frac{1}{125kHz} = 8\mu s$ and 13 cycle will result in $8\mu s * 13 = 104\mu s \sim 105\mu s$ (measured value)

3. The fastest rate one could sample an input voltage is $\frac{1}{105\mu s} = 9523Hz$

2.3 Program 3 - Read Voltage From AD0

The voltage reading is acquired by first reading the ADC value from ADC0 then use the formula $V_{in} = \frac{(ADC+1)*V_{ref}}{1024}$. where V_{ref} is 5v and ADC is off by 1 therefore, we add 1. Program code is listed in appendix section and demo video can be found via <https://tinyurl.com/y3zk8nxw>.

3 Summary

This lab introduced ADC peripheral along with its usage through experiment 1 and 3. ADC port can be used to read input voltage based on a reference voltage. Experiment 1 emphasized the residual that one can possibly face when performing ADC reading because of V_{in} precision.

Many practical applications can be derived from ADC like reading a temperature sensor. One can build a sensor to output a voltage based on the temperature and allow its client to read the voltage.

4 Appendix

4.1 Program 1

```

1  volatile uint8_t *pADMUX, // configure voltage ref and pin
2      *pADCSRA, // Control status register A
3      *pADCL,
4      *pADCH;
5
6  void experiment01() {
7      pADMUX = (uint8_t *)0x7C;
8      pADCSRA = (uint8_t *)0x7A;
9      pADCL = (uint8_t *)0x78;
10     pADCH = (uint8_t *)0x79;
11
12     uint16_t result;
13
14     *pADMUX = 0x40; // 0100 0000. 01: using Vcc. 0000: ADC0
15     *pADCSRA = 0x87; // 1000 0111. 1: ADC enable. 111: prescaler = clock/128
16
17     while (1) {

```

```

18     Serial.print("result = ");
19     Serial.println(readADC0());
20 }
21 }
22
23 uint16_t readADC0() {
24     uint16_t result;
25     *pADCSRA = (*pADCSRA) | 0x40; // start conversion
26     while (((*pADCSRA) & 0x40) == 0x40) { // while still conversion
27         // do nothing. waiting ADSC bit to be cleared by hardware
28     }
29
30     result = *pADCL; // lower 8bits of ADC conversion (0x00??)
31     result = result | (((uint16_t)*pADCH) << 8);
32     result = result & 0x3FFF; // clear bits 15-10 (ADC res is only 10 bits)
33     return result;
34 }

```

4.2 Program 2

```

1
2 volatile uint8_t *ioPORTB,
3                 *ioDDRB;
4
5 volatile uint8_t *pADMUX, // configure voltage ref and pin
6                 *pADCSRA, // Control status register A
7                 *pADCL,
8                 *pADCH;
9
10 void experiment02() { // measure ADC0 reading
11     pADMUX = (uint8_t *)0x7C;
12     pADCSRA = (uint8_t *)0x7A;
13     pADCL = (uint8_t *)0x78;
14     pADCH = (uint8_t *)0x79;
15
16     *pADMUX = 0x40; // 0100 0000. 01: using Vcc. 0000: ADC0
17     *pADCSRA = 0x87; // 1000 0111. 1: ADC enable. 111: prescaler = clock/128
18
19     ioPORTB = (uint8_t *)0x25;
20     ioDDRB = (uint8_t *) 0x24;
21
22     *ioDDRB = 0x01; // DDRB[7:0] 0000 0001 // set PB0 as output , PB0 - port 8 on arduino
23
24     while (1) {
25         *ioPORTB = 0x01; // set PB0 high
26         readADC0();
27         *ioPORTB = 0x00; // 0000 0000 .set PB1 low
28         myHardDelay(1);
29     }
30 }

```

4.3 Program 3

```

1 volatile uint8_t *pADMUX, // configure voltage ref and pin
2                 *pADCSRA, // Control status register A
3                 *pADCL,
4                 *pADCH;
5
6 void experiment03() { //measure ADC0 reading then convert it to volt
7     pADMUX = (uint8_t *)0x7C;
8     pADCSRA = (uint8_t *)0x7A;
9     pADCL = (uint8_t *)0x78;
10    pADCH = (uint8_t *)0x79;

```

```
11 |
12 | *pADMUX = 0x40; //0100 0000. 01: using Vcc. 0000: ADC0
13 | *pADCSRA = 0x87; // 1000 0111. 1. ADC enable. 111: prescaler = clock/128
14 |
15 | float adc_reading;
16 | while (1) {
17 |     adc_reading = (float) (readADC0() + readADC0()) / 2. + 1.;
18 |     float vin = adc_reading * 5. / 1024.; // vin = (ADC+1) * Vref / 1024
19 |     Serial.print("Voltage_reading_=");
20 |     Serial.println(vin);
21 |     myHardDelay(100);
22 | }
23 | }
```