Lab 9

Analog-to-Digital Converter

FLORIDA POLYTECHNIC UNIVERSITY

EEL 4746C / EEL 5756C: Microcomputers

Fall 2018

Student Name:	
Student ID:	
Section:	

Contents

1 Objective

The objective of this lab is to familiarize the students with Analog to Digital converter and demonstrate ability to read the micro-controller datasheet.

2 Introduction

Analog to digital converters are important modules that allow applications to used sensors of non-electrical quantities. An analog signal can be converted into a digital signal or multiple binary signals. In order to convert an analog signal into a digital signal, the following steps must be applied:

1. The analog signal must be sampled, i.e., the signal must be moved from continuous time to discrete time. If the reconstruction of the analog signal from the digital signal is required, then the sampling rate must be performed at a rate which is at least twice the highest frequency that present in the analog signal. This sampling rate is known as Nyquist frequency.
2. The sampled signal is then quantized. In quantization, the voltage range of the sampled signal is divided into L levels and the sample voltage is rounded to the nearest level. These levels are numbered from 0 to $L-1$. Since we use binary system, we will say that $L = 2^n$, where n is the number of bits in which the voltage value of the sample is stored. We also refere to Quantization transform the signal from analog to digital.
3.

2.1 ATmega328P A2DC Subsystem

ATmega328P has a 10-bit Analog-to-Digital Converter (ADC) with six channels. The channels are multiplexed a set of bits in one of the registers. The ATmega328P doesn't support the required rate to sample audio and video signals. ATmega328p has six registers (DIDR is divided over two registers) to control the ADC module. The C code that represents the ADC in ATmega328P is given by:

```
typedef struct adc {
  uint8_t ADCL;
  uint8_t ADCH;
  uint8_t ADCSRA;
  uint8_t ADCSRB;
  uint8_t ADMUX;
  uint8_t not_used;
  uint8_t DIDR0;
  uint8_t DIDR1;
} adc_t;

volatile adc_t *adcs=(adc_t *) 0x0078;
```

Refer to section 28 of the datasheet.

3 Procedure

Step 1. Create two outputs LED circuits.

Step 2. The code portion listed below is used to get samples from the ADC. By referring to section 28 of the datasheet explain the code.

Step 3. Write a C-program, using the code sections in the lab, to divide the value received from the ADC by 8 and send the result to the LEDs.

Step 4. Test the C-program by connecting external voltage source to ADC0. Note 1: Voltage should not exceed 4.5V. Note 2: Make sure to connect the GND of the external voltage source the GND on your board. Note 3: The TA must check your circuit before testing.

```
adcs->ADCSRA=0x87;
adcs->ADMUX=0x40;
while (1) {
            adcs->ADCSRA= adcs->ADCSRA | 0x40;
            while ((adcs->ADCSRA&0x10) == 0 );
            value=(adcs->ADCH & 0x03) * 256 +
                                  adcs->ADCL;

           // adcs->ADCSRA|=0x10;
}
```