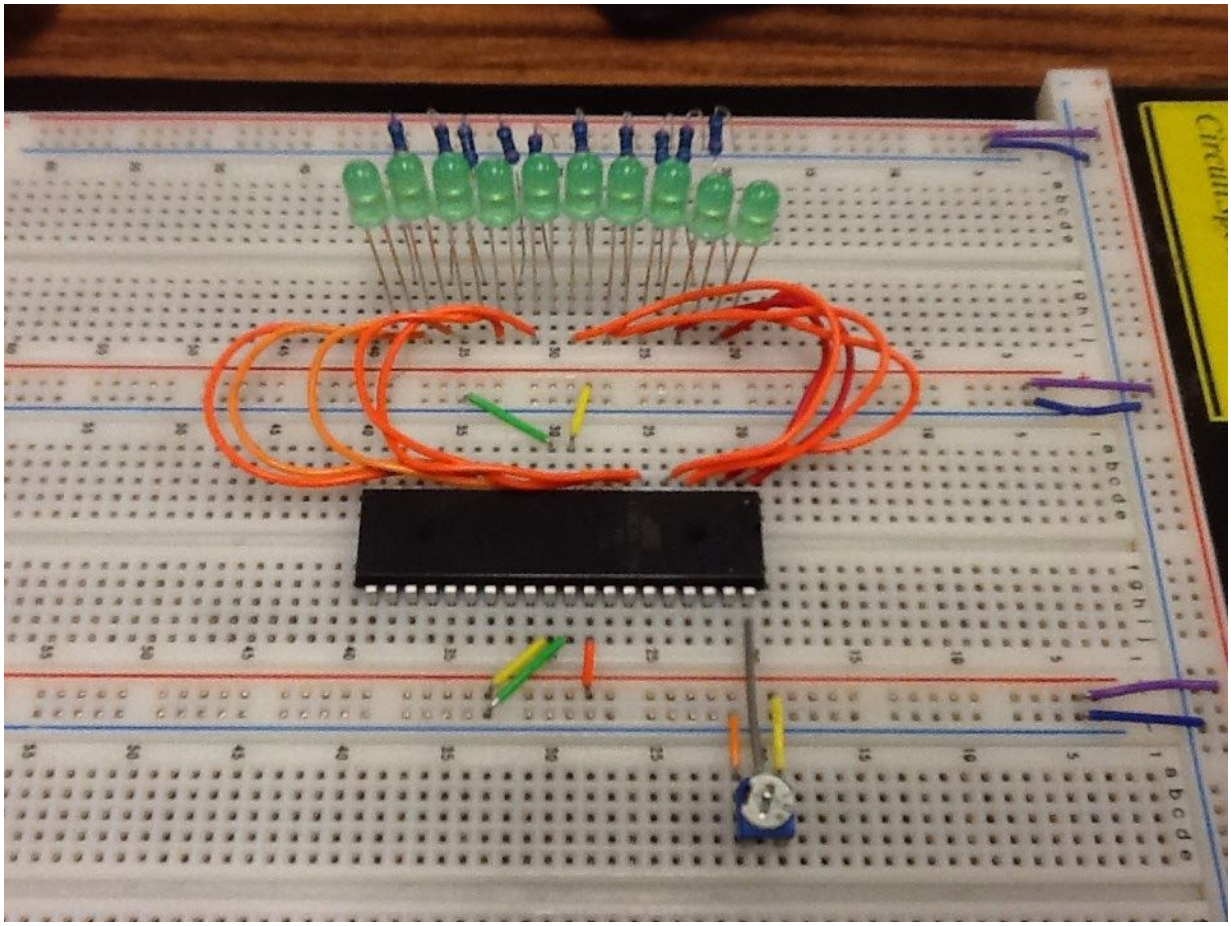
Lab 8: ADC

UCR EE/CS120B

Pre-lab

Come prepared with breadboard wired like the one below. The connections are as follows:

Input: PA[0] connected to potentiometer
Output: PB[7:0] connected to LEDs
PD[1:0] connected to LEDs

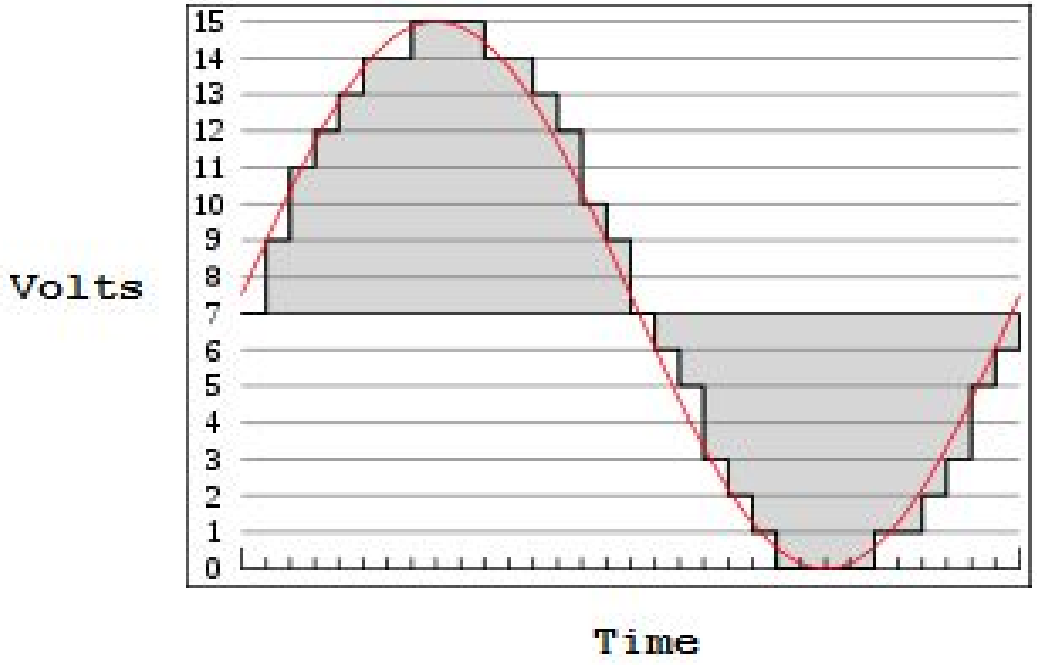


Introduction

A programmer often wants to read input from the environment, such as sound from a microphone, levels from a light sensor, temperature from a thermometer, etc. Sensors often convert environmental input into an analog signal output. The amplitude, and perhaps frequency, of the analog signal output usually indicates the intensity of the sensory input.
A microcontroller can react to stimulus detected by the sensor by interpreting the analog output signal's amplitude or frequency. For example, a microphone that converts sound waves into an analog signal can be connected to a microcontroller that enables a light if the sound is too loud (perhaps as an anti-burglar device).
An analog signal must first be converted into a digital value before a microcontroller can use it. Most microcontrollers provide analog-to-digital converters (ADCs) for this purpose. In this lab, you will utilize the ADC on a microcontroller to interface with a light sensor.

How the ADC works

Consider a sine wave analog signal like the red line in the image below. This particular sine wave has a DC bias of 7.5 Volts and an amplitude of +/- 7.5 Volts, creating a signal that oscillates between 0 and 15 Volts. An analog-to-digital converter discretizes the analog signal, yielding a digital number that represents the amplitude of the signal. The image below shows how the sine wave might be discretized into 16 levels using a 4-bit ADC. Note that the discretization of the signal results in 'steps' being formed when the signal is traced, (an effect usually called aliasing).



The ADC on the ATmega1284 has 10 bits of precision, thus offering 1024 individual possible levels that an analog signal can be represented as a digital value. The ATmega1284 uses the following equation to calculate the converted digital value of the signal amplitude, which is stored in the register ADC:

$$ADC = (V_{IN} * 1024) / V_{REF}$$

- The input voltage V_{IN} is the analog signal from a sensor.
- The reference voltage V_{REF} is used to scale the result.
- Higher values of ADC correspond to voltage levels closer to V_{REF}.

The result of the conversion, stored in ADC, can be read at any time by the C program. A programmer simply writes code like:

```
unsigned short x = ADC;
// Value of ADC register is now stored in variable x.
```

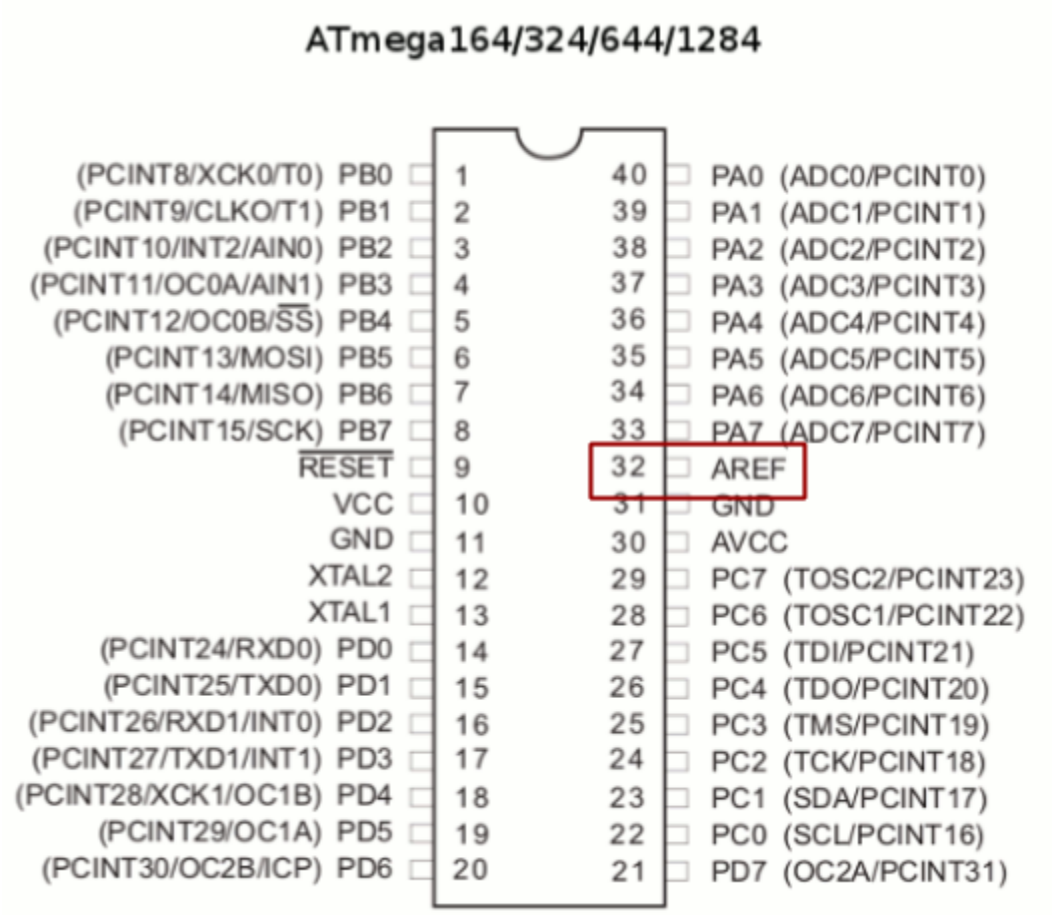
Initialization

Analog to digital conversion will only work with the ATmega1284 if specific flags are set within the microcontroller. The function below sets those flags. Write the function below to the start of your C program to initialize the A/D converter. Make sure to call the function within main, but before the while loop.

```
void ADC_init() {
    ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADATE);
    // ADEN: setting this bit enables analog-to-digital conversion.
    // ADSC: setting this bit starts the first conversion.
    // ADATE: setting this bit enables auto-triggering. Since we are
    //        in Free Running Mode, a new conversion will trigger whenever
    //        the previous conversion completes.
}
```

After the above function is called, analog to digital conversion will work as follows.

- V_{in} is connected to PA0.
- ***IMPORTANT*:** The AREF (V_{ref}) pin is connected directly to the +5 Volt power supply. AREF is the pin located between PA7 and the ground pin. A schematic of the ATmega1284 is given below.



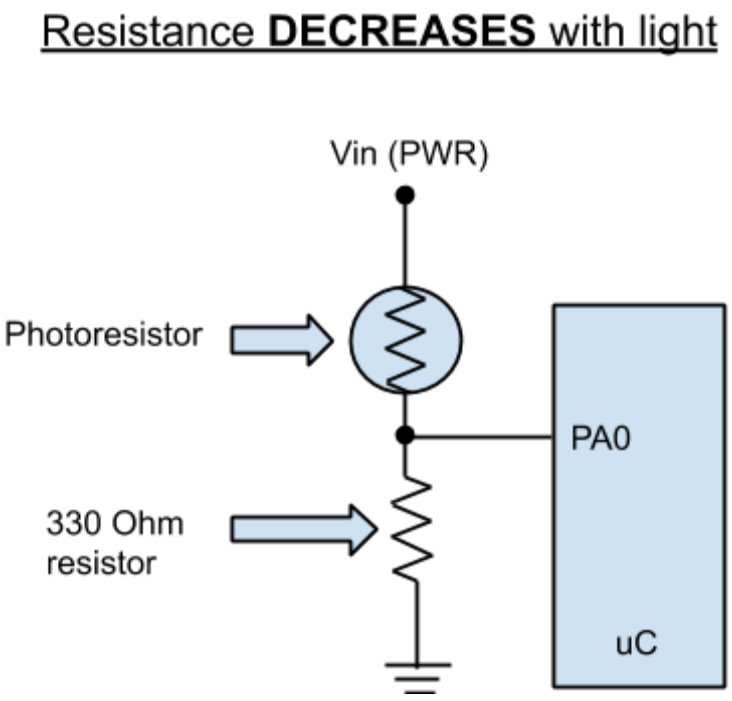
Exercises

1. Make sure your breadboard is wired according to the prelab. The potentiometer is used to adjust the voltage supplied to the microcontroller for ADC . Design a system that reads the 10-bit ADC result from the ADC register, and displays the result on a bank of 10 LEDs.
   Hints:
   a. Assuming the breadboard has been wired according to the prelab photo, display the lower 8 bits on port B, and the upper 2 bits on port D.
   b. Use a "short" variable to hold the ADC result.
   c. Use bit shifting and casting to align the proper bits to the proper ports. For example:

```
unsigned short my_short = 0xABCD;
unsigned char my_char = (char)my_short; // my_char = 0xCD
my_char = (char)(my_short >> 4); // my_char = 0xBC
```

Video Demonstration: http://youtu.be/DB4fXVLT9r0

2. A **photoresistor** is a resistor whose resistance varies based on how much light the photoresistor detects. An additional resistor needs to be connected in parallel with the photoresistor. Without the additional resistor, results from ADC will be too small to observe. A 330 ohm resistor is chosen because it is common in the lab kits. Connect the photoresistor to the microcontroller according to the diagram below.



NOTE: With the photoresistor connected in this way, more light equals higher ADC values.

Revise exercise 1 by replacing the potentiometer with a photoresistor and 330 ohm resistor. Take note of the highest ADC value displayed (`MAX`) when the photoresistor is exposed to light, and the lowest ADC value displayed (`MIN`) when the photoresistor is deprived of all light. These values will be used for the remaining lab exercises.

Video Demonstration: http://youtu.be/Zx2ulW9UJfc

3. Design a system where an LED is illuminated **only** if enough light is detected from the photo resistor. **Criteria:**
    a. If the result of the ADC is `>= MAX/2`, the LED is illuminated.
    b. If the result of the ADC is `< MAX/2`, the LED is turned off.

Note: `MAX` is the highest ADC value observed from exercise 2 of the lab.

Video Demonstration: http://youtu.be/Hx0FWGBI6-g

4. (**Challenge**) Design a system, using a bank of eight LEDs, where the number of LEDs illuminated is a representation of how much light is detected. For example, when more light is detected, more LEDs are illuminated. **Criteria:**
    a. The LEDs should illuminated in sequence from 0 to 7, based on the amount of light detected by the photoresistor.

Hints:

- Use the `MAX` ADC value observed from exercise 2 as the highest amount of light detectable by the photoresistor. Divide that number by eight to determine the thresholds for each LED.

Video Demonstration: http://youtu.be/n9ejT-PNJTI

Submission

Each student must submit their source files (`.c`) and any new/modified header file through Gradescope according to instructions in the lab submission guidelines.

Don’t forget to commit and push to Github before you logout!