The Analog to Digital Converter

University of California, Riverside — EE/CS120B Prelab Assignment 1

Datasheets

So far, we've used the term "datasheet" a few times. So what is it? The *datasheet* is the all-in-one bible of any electronic device. They detail everything one would need to know when using that component—what it's used for, how to use it, how not to use it, how big it is, who made it, etc. If you ever have a question about your device, you reference the datasheet. The teams that work on writing these documents are truly built different. They put in the effort so we can spend more time on our work, and less time deciphering theirs. Just imagine trying to use one of these devices without *any* documentation. Some companies do that, and you'll be forced to build your product off of starter code they provide. And if you have issues? You'll be forced to wait on a call with one of their engineers to help you debug your issue—which is fine when you have the time, but you'll more often find yourself under a deadline, which makes development *incredibly* stressful.

Here's the datasheet for the ATmega328P. It's nearly 300 pages long. Now, none of us have gone through each and every page of this document. The way datasheets are structured, they function like dictionaries. You can search for the functionality (e.g. PWM, DDRs, etc.) that you need and see exactly how to use them. Unlike a dictionary, the "definition" isn't a fraction of the page long. Often times, you'll find additional terms that are explained elsewhere in the datasheet, so you'll need to get used to jumping around. Being able to navigate and extract relevant information quickly is a skill you will be asked to hone over the course of these labs.

For those of you wondering why the datasheet is written so cryptically—it's not. If you persist in trying to understand the datasheet, you'll see it's written for accuracy, detail, and (ironically) conciseness. If it appears daunting, the only thing you need to keep in mind is to follow through, you'll see that it's actually easy to use and once you get the hang of them, you'll see your power grow 

The Analog-to-Digital Converter (ADC)

Microcontrollers are digital machines, they only operate on digital information—not analog. However, there are many components that encode information in analog; for example, potentiometers, photoresistors, and thermistors, to name a few. In order to understand the data produced by such components, microcontrollers need a circuit to map continuous, analog voltages to discrete, digital values. Such circuits are called analog to digital converters (ADCs). Fortunately, we don't need to assemble any additional circuitry. The ATmega328P has an internal ADC that performs this conversion for us; all we need to do is to configure the ADC, identify which pin we want to use, and hook up our component. In the next lab, you'll be using the ADC to interface with a photoresistor—a device whose resistance varies with light intensity. This prelab should not take more than 30 minutes to complete. If you're struggling to complete this, that's okay, reach out to me during OH. I'll likely go over this during Discussion as well.

Directions

For this Prelab, your objective is to understand and configure the appropriate registers in the ATmega328P for converting analog signals into digital values. You'll use this code in next week's lab. On the next page, you'll find a collection of questions to be completed. These are meant to serve as a guide to navigating the datasheet and configuring the necessary registers. You'll submit this document along with your this week's lab (Week 1 Lab).

Questions

Open the datasheet and jump to page 205. This section covers everything we need to know about the ADC. Pages 205-216 provide an overview of the ADC's various features, and also how the ADC works under the hood. This is useful, as knowing how the ADC works will help in configuring it.

1. What is free running mode?

2. How many cycles does it take for an ADC conversion to complete? _____

3. What is the resolution of the ADC? _____

4. Which two registers stores the converted ADC result? _____

5. What is a prescaler (feel free to google)? What is it used for? (Your Arduino runs at 16MHz)

The ADC module is controlled by three registers: `ADMUX`, `ADCSRA`, and `ADCSRB`. Each register has eight bits, with each bit serving some specific function. Look through pages 217-220 to understand that function, then choose a value for each bit so that the ADC *run in free running mode, uses channel 1 for the input, and AVCC for the reference voltage*. Finally, explain why you chose that value.

6.

ADMUX — ADC Multiplexer Selection Register	Bit		Value	Function/Reasoning
	7	`REFS1`		
	6	`REFS0`		
	5	`ADLAR`	`0`	Right align the 10-bit ADC result across the 16-bit registers.
	4	Res.	`0`	
	3	`MUX3`		
	2	`MUX2`		
	1	`MUX1`		
	0	`MUX0`		

7.

ADCSRA — ADC Control & Status Register A	Bit		Value	Function/Reasoning
	7	ADEN		
	6	ADSC		
	5	ADATE		
	4	ADIF		
	3	ADIE		
	2	ADPS2		
	1	ADPS1		
	0	ADPS0		

8.

ADCSRB — ADC Control & Status Register B	Bit		Value	Function/Reasoning
	7	Res.	0	
	6	Res.	0	
	5	Res.	0	
	4	Res.	0	
	3	Res.	0	
	2	ADTS2		
	1	ADTS1		
	0	ADTS0		

Continued on next page.

9. Configure the following registers with the values derived above. Then extract the result from the ADCH and ADCL registers and store them in the variable: `value`. *Note: Page 219 of the datasheet specifies the order in which ADCH and ADCL is read from* ***matters****.*

```
int main() {

    ADMUX  = 0x________;
    ADCSRA = 0x________;
    ADCSRB = 0x________;

    while (true) {

      unsigned int value = ________________;

    }

    return 0;
}
```