Intro to Microcontrollers

Class 4: Input II: Debouncing and Analog-to-Digital Conversion

October 6, 2008

Outline

- Review and Today's Setup
- Debouncing
- Analog to Digital

Outline

Review and Today's Setup

Debouncing

Analog to Digital

Review

Show and Tell

- Anyone make anything cool they want to show?

Output

- Learned how to set up pins for input
- Saw an if() statement
- Saw more bit-masking examples (for input this time)
- Talked a bit about audio, and got buzzers buzzing
- Quiz: What is a pullup resistor for?
  How do you enable it?

My Screwup Last Time

Setting it Straight

- Here's why you need to think hard (sometimes) about bit logic
- Read in PINB – 8 bits, one for each pin
- Want to test when PB0 goes low (is pressed)
- (PINB & _BV(PB0))
  → 00000000 if PB0 is low, 00000001 if PB0 high
- What I did: (wrong): ~(PINB & _BV(PB0))
- What would have worked: !(PINB & _BV(PB0))
- It inverts the last bit, alright, but also all of the others too
- Instead: ~PINB & _BV(PB0) does what we want:
  ~PINB → xxxxxxx1 if PB0 is low
  & _BV(PB0) masks/zeros all but PB0

Outline

The Real World

Switching Noise

- In reality, switches make/break contact a bunch of times as you press it or release it
- Two pieces of metal touching, bending, with different resistance all over
- If you're trying to make a per-button-press device, this can cause troubles
- Symptom: Get multiple presses for what you thought was a single press
- Solution: Debouncing

Debouncing

Patience!

- The trick is to see if the button is still pressed some time after it was first pressed
- Couple ways to do this:
  if you've already got a timing loop, you can keep track of how many times through, and re-test
- Or if you're not concerned with real-time performance, you can just wait a bit and double-check

Outline

Reading in from the Outside World

From Black-and-white to Greyscale

- Have a voltage on a pin, and you want to know what it is
- So far, just know on/off
- Want to convert the voltage to a number that the chip can use
- Analog to digital conversion
- AVRs have a single 10-bit ADC, which it can use on many pins
- Setup is the tricky part...

ADC Theory

How does it work?

- Inside the chip, it has a multi-way switch (multiplexer)
- When you take a reading, it compares whichever pin the switch is connected to
- It starts by comparing the pin voltage to a voltage reference of 1/2 Vcc (2.5V in our case)
- If it's higher, it divides the source by 2 and compares again
- If it's lower, it divides the reference by 2 and compares
- Stores the results of ten comparisons in binary form in two registers
- Start comparison, wait until done, then read it out

Voltage Divider

A bit of circuit theory

- Electricity is like water
- Voltage = water pressure. Measured in volts.
- Current = the flow of water. Measured in amps (charge / sec).
- Resistors are like thin pipes
  they restrict the flow of water, and you end up with less pressure downstream of them
- A voltage divider is just two resistors in a row
- Easiest case: equal resistors.
  The voltage in-between them is $\frac{1}{2}$ of the voltage across both

Light Detection

Make a voltage divider from the LDR

- So we've got a good source of 5v
- And we've got a light-dependent resistor
- If we make a voltage divider with the LDR in it, the voltage in the divider will depend on the light
- Then just read that off, play notes accordingly
- Voila. Light-dependent theremin.

ADC Initialization

Things to do

- Set up our ADC pin for input
- Turn off the traditional digital sensing stuff
- Point the multiplexer at our ADC pin
- Enable the ADC
- Wait for the conversion to finish
- Read out the value

The End

◀ Outline