Intro to Microcontrollers

Class 3: Input: Buttons and Analog-to-Digital Conversion

September 29, 2008

Outline

Outline

Review

Show and Tell

- Anyone make anything cool they want to show?

Review

Show and Tell

- Anyone make anything cool they want to show?

Review

Show and Tell

- Anyone make anything cool they want to show?

Output

- Learned how to set up pins for output

Review

Show and Tell

- Anyone make anything cool they want to show?

Output

- Learned how to set up pins for output
- How to write to them using bit-math

Review

Show and Tell

- Anyone make anything cool they want to show?

Output

- Learned how to set up pins for output
- How to write to them using bit-math
- Did a little PWM at the end

Review

Show and Tell

- Anyone make anything cool they want to show?

Output

- Learned how to set up pins for output
- How to write to them using bit-math
- Did a little PWM at the end
- So far, done LED stuff

Review

Show and Tell

- Anyone make anything cool they want to show?

Output

- Learned how to set up pins for output
- How to write to them using bit-math
- Did a little PWM at the end
- So far, done LED stuff
- Today, let's do very simple audio

Audio

What is sound?

- Sound: repetitive compression/decompression of the air around you

Audio

What is sound?

- Sound: repetitive compression/decompression of the air around you
- Speaker: has an electromagnet inside – moves a cone forward and back depending on current running through it

Audio

What is sound?

- Sound: repetitive compression/decompression of the air around you
- Speaker: has an electromagnet inside – moves a cone forward and back depending on current running through it
- Our simple sound plan: Use the 5v/0v output we know from last week to make current flow through a speaker and make noise

Audio

What is sound?

- Sound: repetitive compression/decompression of the air around you
- Speaker: has an electromagnet inside – moves a cone forward and back depending on current running through it
- Our simple sound plan: Use the 5v/0v output we know from last week to make current flow through a speaker and make noise
- Pleasant audio frequencies from 30 Hz to 4200 Hz: 33mS to 283μS per cycle = 16mS to 140μS on/off times

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS
- How?

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS
- How?
- Make a loop that takes a fixed amount of time, toggle every n'th time through

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS
- How?
- Make a loop that takes a fixed amount of time,
  toggle every n'th time through
- See scale.h – a bunch of macros to help make musical notes

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS
- How?
- Make a loop that takes a fixed amount of time,
  toggle every n'th time through
- See scale.h – a bunch of macros to help make musical notes
- Middle C: Around 2mS on/off times.
  2mS / 200 = 10μS per loop

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS
- How?
- Make a loop that takes a fixed amount of time,
  toggle every n'th time through
- See scale.h – a bunch of macros to help make musical notes
- Middle C: Around 2mS on/off times.
  2mS / 200 = 10μS per loop
- Should *just* work if we're not doing too much math

Simple Organ

Setting up for sound

- So want to toggle a pin every 140μS to 16mS
- How?
- Make a loop that takes a fixed amount of time,
  toggle every n'th time through
- See scale.h – a bunch of macros to help make musical notes
- Middle C: Around 2mS on/off times.
  2mS / 200 = 10μS per loop
- Should *just* work if we're not doing too much math
- There is a better way to do it with timers, next class...

Outline

Initialization for Input

Too easy

- To initialize for output set bit to one
  DDRx = _BV(whatever)

Initialization for Input

Too easy

- To initialize for output set bit to one
  DDRx = _BV(whatever)
- For input, want to set the bit to zero instead.

Initialization for Input

Too easy

- To initialize for output set bit to one
  DDRx = _BV(whatever)
- For input, want to set the bit to zero instead.
- But zero is the default value. Done!

Initialization for Input II

One wrinkle: Initialize a pullup

- A pullup resistor ties the input pin to 5v (internally) when it's not pulled low from outside

Initialization for Input II

One wrinkle: Initialize a pullup

- A pullup resistor ties the input pin to 5v (internally) when it's not pulled low from outside
- Often want a pullup with input

Initialization for Input II

One wrinkle: Initialize a pullup

- A pullup resistor ties the input pin to 5v (internally) when it's not pulled low from outside
- Often want a pullup with input
- Why? Simplest input circuit is a switch from pin to ground

Initialization for Input II

One wrinkle: Initialize a pullup

- A pullup resistor ties the input pin to 5v (internally) when it's not pulled low from outside
- Often want a pullup with input
- Why? Simplest input circuit is a switch from pin to ground
- AVR's PORTn does double-duty.
  In output mode, controls output.
  In input mode, selects the pullup

Initialization for Input II

One wrinkle: Initialize a pullup

- A pullup resistor ties the input pin to 5v (internally) when it's not pulled low from outside
- Often want a pullup with input
- Why? Simplest input circuit is a switch from pin to ground
- AVR's PORTn does double-duty.
  In output mode, controls output.
  In input mode, selects the pullup
- So often set PORTn to one to enable the pullup:
  `PORTB |= _BV(PB3);`

Reading the Input

Reading the input register

- Input values in the PINx register

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading one pin: the most common case

- `PIND & _BV(PD3);`

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading one pin: the most common case

- `PIND & _BV(PD3);`
- If PD3 has more than 1.25v on it, we'll get 00001000

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading one pin: the most common case

- `PIND & _BV(PD3);`
- If PD3 has more than 1.25v on it, we'll get 00001000
- If PD3 has less than 1.25v on it, we'll get 00000000

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading one pin: the most common case

- `PIND & _BV(PD3);`
- If PD3 has more than 1.25v on it, we'll get 00001000
- If PD3 has less than 1.25v on it, we'll get 00000000
- Can use as a test of pin state: if(x){...}

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading one pin: the most common case

- `PIND & _BV(PD3);`
- If PD3 has more than 1.25v on it, we'll get 00001000
- If PD3 has less than 1.25v on it, we'll get 00000000
- Can use as a test of pin state: if(x){...}
- So let's go to the simpleOrgan project to see it in action

Reading the Input

Reading the input register

- Input values in the PINx register
- Can read them like `readIn = PINB;`
- `readIn` will contain an 8-bit number, each bit corresponding to the voltage state of all 8 of its pins.

Reading one pin: the most common case

- `PIND & _BV(PD3);`
- If PD3 has more than 1.25v on it, we'll get 00001000
- If PD3 has less than 1.25v on it, we'll get 00000000
- Can use as a test of pin state: if(x){...}
- So let's go to the simpleOrgan project to see it in action
- Remember negative logic!

Outline

The Real World

Switching Noise

- In reality, switches make/break contact a bunch of times as you press it

The Real World

Switching Noise

- In reality, switches make/break contact a bunch of times as you press it
- Two pieces of metal touching, bending, with different resistance all over

The Real World

Switching Noise

- In reality, switches make/break contact a bunch of times as you press 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

The Real World

Switching Noise

- In reality, switches make/break contact a bunch of times as you press 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

The Real World

Switching Noise

- In reality, switches make/break contact a bunch of times as you press 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

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, just check back later

Outline

The End

◂ Outline