Analog Read (A/D)

Last updated 10/30/18

Analog Read

- Need to read an analog value
  - voltage level
  - temperature
  - light sensor

Analog Read

- Reading analog signals

1) Must use an Analog pin

   pins marked A0 – A15 in blue in the pinmap (not the part)

2) Must configure the pin as an input pin

3) Must configure the pin to ADC mode

   PxSEL0 and PxSEL1 set to 1

4) Must configure the A/D to run a conversion (A0 with this code)

   adc0_setup()

5) Must Run the conversion (A0 with this code)

   adc0_convert()

6) Must read the results

Analog Read

- Reading analog signals

  - adc0_setup()
    - Sets up ADC pin 0 (A0) to read from an analog input and perform a 12 bit conversion
  - adc0_convert()
    - Samples the signal, does the conversion and returns a number (int) between 0 and 4095 (12 bits of resolution)

  - The number represents the value of the input
    - 0v → 0
    - 3.3V → 4095
    - everything in-between measures linearly

    - Each step is (3.3V – 0V)/4096 steps → 812uV/step

Analog Read

- Reading analog signals
  - 1.2V signal → 1.2V/(812uV/step) = 1489.5 steps → 1489 returned
  - 428 returned → 428 steps * 812uV/step = 344mV signal

Analog Read

- Reading analog signals

```
int adc0_setup(void){
    // Function to setup Analog input A0
    // for use in A/D conversion

    // Setup ADC Input 0
    // Pin 30 --> P5.5
    P5->SEL0 |= 0x20;   // Select alternate mode 11
    P5->SEL1 |= 0x20;
    P5->DIR &= ~0x20;   // input
    P5->REN &= ~0x20;   // No pull u/d

    // ADC Setup
    // You must enable the Analog 0 pin...
    ADC14->CTL0 = 0x04000210;       // S/H timer, 16clk S/H, ADC ON
    ADC14->CTL1 = 0x00000020;       // 12-bit conversion
    ADC14->MCTL[0] = 0x00000000;    // Vref=AVCC, ADC0 input

    return 0;
} // end adc0_setup
```

```
int adc0_convert(void){
    // Function to perform a single
    // A/D conversion on Analog input 0

    // Start sampling/conversion
    ADC14->CTL0 |= 0x00000003; // enable ADC, start conversion

    // Wait for conversion to complete
    // Conversion is complete when ADC0 flag is set
    while (!ADC14->IFGR0){
        ;
    }

    // returning a full int instead of a uint16_t for simplicity
    return ADC14->MEM[0];
} // end adc0_convert
```