ADC

Files: FuelCell\_ADC.c/FuelCell\_ADC.h/analog\_defs.h

Functions:

Void ADCInit(void);

* Configures ADC.
* gpio\_enable\_module(ADCIFA\_GPIO\_MAP, sizeof(ADCIFA\_GPIO\_MAP) / sizeof(ADCIFA\_GPIO\_MAP[0]));
  + sets analog pins to analog function
    - ADCIFA\_GPIO\_MAP
      * Defined in analog\_defs.h
      * Lists all the pins and their functions in a giant structure of gpio\_map\_t type
    - The second argument has to do with how many pins you are setting. I just copied it from an example
* adcifa\_configure(&AVR32\_ADCIFA, &adcifa\_opt, sysclk\_get\_pbc\_hz());
  + configures the ADC
    - &AVR32\_ADCIFA
      * Memory address of the ADC module in the uc3c chip
    - &adcifa\_opt
      * Address of configuration structure defined in analog\_defs.h
      * Sets frequency to 0.5MHz (max is 1.5 MHz?)
      * Sets conversion reference source to ADCIFA\_REF06VDD
        + Ref = 0.6 \* 5 = 3V
      * Sample and hold disable = 0
        + Sample and hold is enabled
      * Single sequencer mode = 0
        + We have the option of combining both sequencers into one giant sequencer
        + Not using this because running two sequencers at once is faster
      * Free running mode enable = 0 (disabled)
        + Free running mode causes sequencer to run in a constant loop
        + With it disabled you have to call StartADC\_Sequencers() every time you want a conversion
      * Sleep mode enable = 0 (sleep mode disabled)
    - Systclk\_get\_pbc\_hz()
      * ADC gets clock signal from pbc clock
      * This function gets clock rate and passes it to adcifa configuration function
  + adcifa\_configure\_sequencer(&AVR32\_ADCIFA, 0, &adcifa\_sequence\_opt, adcifa\_sequence\_conversion\_option\_0);
    - &AVR32\_ADCIFA
      * Address of ADC module
    - 0 (1 for sequencer 1)
      * Sequencer number (0 or 1)
    - &adcifa\_sequence\_opt
      * Defined in analog\_defs.h
      * Sets resolution, result byte alignment etc. see datasheet
    - adcifa\_sequence\_conversion\_option\_0 (or 1)
      * sets what pins the sequencers read and in which order they read them
      * see datasheet
      * each reading requires you to define a positive reference (ie an input pin) and a negative reference (ie ground)
      * note: half the inputs are “negative” inputs. (need to set pin as negative reference and ground as positive reference) (will also return a negative value ie: you expect 500 but will get -500)
      * You also have the ability to set the gain of each input. I have no idea how that works
  + ADC is now good to go

Void StartADC\_Sequencers(void);

* Starts a new analog conversion sequence.

Void ReadADC\_Sequencers(void);

* Retrieves the conversion results and stores them in adc\_values\_0 and adc\_vals\_1 arrays.

Int convert\_temp(int temp\_reading)

* Converts thermistor reading to temperature in millis Kelvin
* At this time it is simply doing a linear curve fit using table from datasheet and some voltage divider calculations
* I might add the math formulas here later

Int get\_FCTEMP1(void) also FCTEMP2 AMBTEMP[0:3]

* Gets thermistor reading from appropriate adc\_vals array runs it throught convert\_temp and returns temperature

Int get\_TANKPRES(void)

* Gets tank pressure from adc\_vals array
* Does conversion to actual pressure value
* Will explain conversion later

Int get\_FCPRES(void)

* Same as tankpres

Int get\_FCCURR(void)

Void zero\_FCCUR(void)

Int get\_CAPVOLT(void)

Int get\_FCVOLT(void)

Revision 1 done by Reegan McAllister