Basic Analog

Welcome back to Cypress Academy. This is PSoC 6 101. In this video I am going to show you how to use some of the analog features of the PSoC 6. Specifically, the SAR ADC, the voltage reference, and the OpAmp which for some strange reason we call the CTM or sometimes the CTB in our documentation.

To use the analog resources, we really need an analog signal for an input to the PSoC 6. Conveniently enough, on the CY8CKIT-028-EPD E-Ink shield that came with your CY8CKIT-062-BLE there is a thermistor which is perfect for taking analog measurements. We'll use that.

So, let’s get started! Create a new project, I’ll call it 2-6-ADC. For this project I want to read the voltages of the thermistor circuit using the ADC and then convert those readings into a temperature and finally print all of the data onto on a terminal connected to the UART.

If you haven’t used a thermistor before, it is a temperature dependent variable resistor. In other words, if you know the current through it, then you can calculate the temperature with a crazy nasty formula - more on that in a minute. In order to use a thermistor, you put it in series with another precision resistor - one that doesn’t change very much with temperature - typically 0.1%. Alright, let me show you a picture from the E-ink shield schematic.

You can see that the thermistor is connected to A0 and A1, and the reference resistor is connected to A2 and A3… and the intermediate node in the resistor stack between them is shorted.

Typically, the best way to do a thermistor measurement is ratiometric… meaning the absolute value of the voltages aren’t important. Rather, only the ratio of the voltages matters. This is how you get rid of common mode noise and offset in the measurement. The SAR ADC in the PSoC 6 can be configured to do differential measurements which enable these ratiometric calculations.

Let's get to the schematic.

First, Drop the UART onto the schematic. Remember, we're just going to be printing out the values of our calculations. Next you should drop the ADC and four analog pins.

Now I am going to show you something new. In all of the examples that I have shown you so far you have added components to your project. All of these components have been INSIDE of the PSoC. Well just like putting a comment in your C-program, the PSoC Creator team gave you a similar ability in the schematic. You can add “Off-chip components” - sometimes we call them “annotation components” - to your schematic. These components don’t actually DO anything to your PSoC Creator project, but they help you understand what's in your system and what your schematic outside of the PSoC looks like. These components and associated wires, they're always blue - and let me say it again - they don’t do anything to your PSoC Creator project. They are just there for documentation.

Let's add annotation components for the thermistor and a reference resistor to our schematic so you can see what's going on. First, click on the “Off chip” tab in the component catalog… you find a bunch of different annotation components which you can use for documentation. Let drag and drop a resistor and a thermistor… notice how they're blue … remember the blue wires and components are for documentation only.

Now, edit the resistor, change the value to 10K and turn off the instance name.

Next, edit the thermistor… set to 10K and turn off its instance name.

Now, let's look at the schematic again. Notice that there is no power source for the resistor stack. So, how does that work? Well we are going to power the stack from the PSoC by driving a logic high onto that A0 pin. All right Alan, you mean to say that the pins can be both an analog input and a digital output at the same time? Yup. Sure enough. Let me show you.

Edit the first pin and change the name to A0, turn on the external connection, turn on the digital output and set it to high… what this does is makes it so that this pin is both a digital output as well as an analog input. So I can drive a one or a 3.3V onto it but at the same time it can also an analog input which can be routed to the differential input on the SAR. How cool is that? The other new thing is that by turning on the external connection it will give you a terminal to hook the annotation components to.

Now remember from the schematic that there was not a ground on the stack. Well we'll use the PSoC to connect a ground, also known as a logic zero. How do we do that? The same way we did the high side. So, edit the pin and change the name to A3, turn on the external connection and turn on the digital output and set the initial state to low.

The other two pins are normal analog pins… but let's change their names to A1 and A2 and turn on the external connections.

Now let's wire it all up… notice that while I am doing this the wires that I connect to the external pins are blue… remember that they don’t do anything in the project… they are just there for documentation.

The next thing that I want to do is configure the ADC. Notice that it was already setup for 2 channel differential measurements. First, I'll double click on it. Now with the default settings, the ADC can only do differential measurements from minus vref to plus vref, and that's only plus or minus 1.2V, but we know we need to measure higher voltages than that. So first change the VREF to Vdda to increase the range. When I do this, I am able to measure plus or minus 3.3V - that'll work. Next, I want to turn on averaging, then select 256 samples. Averaging is effectively putting on a big low pass filter which gets rid of noise. Notice that it will be slowing down the speed of the SAR because it has to take all of those samples.

Now that the schematic is done, I'll assign the pins. The UART RX and TX to P5.0 and P5.1, A0 to P10.0, A1 to P10.1 A2 to P10.2 and A3 to P10.3.

I want to use printf, so I'll change the build settings to include stdio retargeting.

Now, run Generate application.

I told you earlier that the actual temperature value of a thermistor is calculated with a big gnarly equation. Well, in PSoC 4 we have a thermistor calculator library, but I noticed that the library isn’t in PSoC 6 – well at least for now – do you hear that Misha? Let's get the thermistor library - so how do I get it there? Well, let's make a PSoC 4 project. Do File-New Project … PSoC 4… Let's see here, I'll call it p4therm… now I'll add the thermistor to the project… now run Generate Application… and after a few seconds you'll see a directory called Thermistor in the Generated Source. That's exactly what we need. I'll copy Thermistor.h and paste it into my CM4 Header Files folder, and then copy Thermistor.c and paste it into my cm4 Source Files folder.

Now, I'll make the 2-6-ADC project the active project again so that I don’t accidently hose myself and program the wrong thing. Then I'll make one small change to Thermistor.h. Specifically I'll delete these two includes and just include “project.h”.

Next, I'll change stdio\_user.h to include project.h and use UART\_1\_HW.

And finally, I'll edit some firmware. I'll open up the main\_cm4 and include stdio and Thermistor.h.

Then in the main function I need to start the UART, start the ADC, and tell the ADC to start running continuously.

In the main loop I need to declare a couple of floats to be the voltage of the two differential inputs and I need counts for the integer values that come back from the SAR.

Then I read ADC channel 0 and assign it to the countReference - remember channel 0 is the reference resistor - then do the same thing for the thermistor on channel 1.

Next, I'll convert the counts from the two channels into volts. This isn’t needed, but I do it just so I can print out the voltages.

I'll call the thermistor library function to find out the resistance of the thermistor. This is part of what I copied over from the PSoC 4 project.

Then I'll convert that value into a temperature by calling another library function – again copied from the PSoC 4 project - this function actually returns the temperature in 100ths of a degree Celsius and I'd really like to have it in degrees Celsius so I'll divide by 100.

Finally, I'll print out the whole mess onto the screen, do a delay and then loop back to the start.

Now for the moment of truth. Hit program.

When I open up the terminal I can see that 5 times a second, I am seeing the voltages and the temperature. How cool is that?

Now when I put my finger on the thermistor, it warms up, and yes, I can see the new values.

Next, I'll take my Fluke meter with a thermocouple, and look there, the Fluke says nearly the same thing as the PSoC 6.

Finally let's measure the actual voltages… and look they are nearly the same as well.

What happens if you have a noisy power supply? Well this whole thing is dependent on your measurement of the reference resistor and the thermistor being taken at exactly the same supply voltage. If the current changes because of the power supply noise, you'll end up with a less accurate reading. The averaging that we are doing in the ADC helps to filter out the noise, but I'd like to change it so that the system is not referenced to the external analog supply. Let me show you how you can do this by using the internal reference of the chip and an OpAmp to improve the overall measurements.

First, copy the project and then paste it back into the same workspace.

Now rename it to 2-6-ADC\_OPAMP\_VREF.

Change A0 back to just an analog pin. Fix the external component wire, and then do this…

Inside of the chip there is a very accurate, very stable analog reference called VREF. In fact, this is also the signal the SAR uses to get accurate measurements. But you can access this signal in your design. To do that add the Vref signal to your schematic.

The Vref voltage can be selected in the System settings in the Design Wide Resources if you want something other than the default value of 1.2V. But that 1.2 is perfect for what we need, so we will leave it at that.

To get the Vref signal to an output pin, first I'll buffer it with an OpAmp. To do that, add an OpAmp to the project.

Change the settings and make it a follower… also known as an analog buffer… and change it to output to a pin. Then wire the output of the follower to A0.

This time we know that the maximum voltage across the thermistor or the resistor will be much less than 1.2V so we can change the ADC Vref setting to be the system bandgap which will give us a measurement that is more immune to power supply noise.

Now we need to start up the OpAmp in the main\_cm4.c.

Alright, hit Program.

When I look at the terminal program I can see that I get input voltages of around 0.6 V at room temperature which is exactly what we expected.

Alright, I've shown you how to use the analog to digital converter, the built in OpAmp, so we got some more good stuff coming so keep paying attention.

As always, you can post your comments and questions in our PSoC 6 community or you are welcome to email me at alan\_hawse@cypress.com or tweet me @askioexpert with your comments, your suggestions, your criticisms or your questions. Thank you.