| **VIDEO P6-2-6-ADC-OPAMP-REF Version 1** | | |
| --- | --- | --- |
| Cell | **Visuals** | **Audio** |
| 1 |  | Welcome back to Cypress Academy, 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 CTB in our documentation. |
| 2 | VIDEO:  Show the E-Ink shield and point to the thermistor. | 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 kit there is a thermistor which is perfect for taking analog measurements. |
| 3 | SCREEN CAPTURE:  P6-2-6-ADC\_capture1.trec | 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. |
| 4 | SCREEN CAPTURE:  P6-2-6-ADC\_capture2.trec  (schematic of E-Ink shield) | 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 point 1%. 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. |
| 5 |  | 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. |
| 6 | SCREEN CAPTURE:  P6-2-6-ADC\_capture3.trec | Let's get to the schematic.  First, Drop the UART onto the schematic. Next 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 called “annotation componetns”. These components don’t actually DO anything to your PSoC creator project… but they help you understand what your system looks like. These components and associated wires are always in blue… and let me say it again… they don’t do anything to your project. They are just there for documentation.  Let's add annotation components for the thermistor and reference resistor to our schematic so that 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 are 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 the 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 to high… what this does is makes it so that this pin is both a digital output … so I can drive a one or a 3.3V onto it… but at the same time it is 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.  Now… remember from the schematic that there was not a ground on the stack. Well we will use the PSoC to connect a ground… also known as a logic low. How do we do that? 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 out and set the initial state to low.  The other two pins are normal analog input pins… but let's change their names to A1 and A2 and turn on the external connections.  Now let's wire it 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. |
| 7 | SCREEN CAPTURE:  P6-2-6-ADC\_capture4.trec | 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 it. Now with the default settings, the ADC can only do differential measurements from minus vref to plus vref… and that is only plus or minus 1.2V... but we know that we need to measure higher voltages… So first change the VREF to the Vdda input to increase the range. When I do this, I am able to measure plus or minus 3.3V … that will 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 slowed down the speed of the SAR.  Now that the schematic is done, I'll assign the pins… UART 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 |
| 8 | SCREEN CAPTURE:  P6-2-6-ADC\_capture5.trec | 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… at least for now… so how do I get it there? Well… let's make a PSoC 4 project... file-new project … PSoC 4… I'll call it p4therm… now add the thermistor to the project… now run generate application… and after a few seconds you will see a directory called Thermistor in the generated source… that is exactly what we need. I'll copy Thermistor.h and then paste it into my CM4 Header Files folder… and then copy Thermistor.c and paste it into my cm4 Source Files folder. |
| 9 | SCREEN CAPTURE:  P6-2-6-ADC\_capture6.trec | Now… I will make the 2-6-ADC project the active project again so that I don’t accidently hose myself… then I'll make one small change to Thermistor.h… 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… |
| 10 | SCREEN CAPTURE:  P6-2-6-ADC\_capture7.trec | And finally, I'll edit some firmware. I'll open up main\_cm4.c 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 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 that I can print out the voltage.  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 convert that value into a temperature by calling another library function – again copied from the PSoC 4 project - this function actually returns temperature in 100ths of a degree Celcuis… .and I'd like to have it in degrees Celcuis… so divide by 100.  Finally, I'll print out the whole thing… and do a delay and then loop back to the start.  Now for the moment of truth. Hit program. |
| 11 | SCREEN CAPTURE:  P6-2-6-ADC\_capture8.trec | When I open up the terminal… I can see that 5 times a second, I am seeing the voltages… and temperature… how cool is that? |
| 12 | VIDEO:  Video of Alan putting finger on thermistor.  Video of Alan measuring the temperature with a Fluke meter. Need to be able to see values on the UART at the same time since we want to compare the same values in both places.  Video of Alan measuring voltages. Need to be able to see values on the UART at the same time since we want to compare the same values in both places. | 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 value as the PSoC 6…  Finally let's measure the actual voltages… and look they are nearly the same as well. |
| 13 |  | What happens if you have a noisy power supply? Well this whole thing depends on your measurement of the reference resistor and the thermistor being taken with exactly the same supply voltage… if the current changes because of power supply noise, you will end up with a less accurate reading. The averaging that we are doing in the ADC helps to filter out 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 use the internal reference of the chip and an OpAmp to improve things. |
| 14 | SCREEN CAPTURE:  P6-2-6-ADC\_capture9.trec | 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.  Inside of the chip there is a very accurate reference voltage called VREF. In fact, this is the signal that 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. The default value is 1.2V which is perfect for what we need, so we will leave it at that setting.  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.  Make it a follower… also known as an analog buffer… and change it to output to a pin.  Wire it to A0.  This time we know that the maximum voltage across the thermistor or the resistor will be less than 1.2V… so we 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.  Program it. |
| 15 | SCREEN CAPTURE:  P6-2-6-ADC\_capture10.trec | When I look at the terminal program I see that I get input voltages around 0.6V at room temperature which is exactly what we expected. |
| 16 | TEXT ON SCREEN:  Cypress Developer Community  community.cypress.com  VIDEO:  Show video of ARH email and twitter windows. | You can post your comments and questions in our PSoC 6 community or as always you are welcome to email me at alan\_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions. |