Exercise 3 -- Measure an Analog Input (Part 1)

Introduction

The document **`Packages.pdf`** in **`ada-remoteio-tutorial/`** contains annotated package specifications for the packages that you will be using during this tutorial session. For this exercise you will need to reference the packages **`Analog`**, **`ADC.RemoteIO`**, and **`RemoteIO.LPC1114`**.

In this exercise you will begin working with ***analog*** sensors. The potentiometer (variable resistor) acts as a voltage divider that splits the LPC1114 I/O Processor board's 3.3V power supply voltage. At one extreme of rotation, the voltage at the center terminal (yellow wire) or wiper will be 0.0V. At the other extreme of rotation, the voltage will be 3.3V. By adjusting the rotation, you can select any voltage between 0.0 and 3.3V. This is the primary characteristic of all analog sensor signals: They are infinitely variable between some physical limits.

Computers, including your PC or Mac and including the LPC1114 microcontroller, are ***digital***, meaning all fundamental data values are limited to a discrete range, usually a power of two. The LPC1114 microcontroller is a 32-bit machine, meaning fundamental data values are constrained to the range 0 to 2^{32}, or 0 to 4294967295. Your PC or MAC is probably a 64-bit machine with a much larger (but still finite) range of fundmental data values.

In order to move data from the ***analog voltage domain*** to the ***digital data domain***, we will use a device called an Analog to Digital Converter or just ADC. The LPC1114 I/O Processor contains hardware support for 5 ADC channels, each with 10-bit resolution or 2^{10} or 1024 separate values. In effect, the LPC1114 ADC splits the voltage range 0.0 to 3.3V into 1024 subranges numbered from 0 to 1023. Any analog input voltage fitting inside the 511'th subrange will be reported as a discrete value of 511.

The discrete values ("bin numbers") corresponding to the subranges are called ***sampled data values*** and are an ***approximation*** of the original analog signal levels. The higher the resolution of the ADC (more bits, more bins), the better the approximation will be (and the more expensive the hardware will cost!) but sampled data is ***always*** just an approximation.

The Linux Simple I/O Library and the Remote I/O Protocol both define sampled data as 32-bit unsigned integer values. The package **`Analog`** defines an abstract interface called **`Analog.InputInterface`** for all ADC devices that return sampled data values. The package **`ADC.RemoteIO`** implements **`Analog.InputInterface`** as **`ADC.RemoteIO.InputSubclass`** to provide services for obtaining sampled data values from remote ADC inputs.

For this exercise, you will write an Ada program that will read a sampled data value once per second from Remote I/O ADC channel 1 and display it to the console. The displayed values will correspond to both the analog input voltage ***and*** the physical position of the potentiometer. (Controls for video game consoles and model aircraft often use a potentiometer fed into an ADC to capture the desired position or setting.)

Hardware Setup

photo goes here

Plug the potentiometer assembly from the tutorial hardware kit into Grove socket **`J2`** (Raspberry Pi Zero) or **`J4`** (BeagleBone Green). This attaches the center wiper (yellow wire) of the potentiometer to Remote I/O ADC channel 1.

Instructions

1. In **`gps`**, right click on the **`.`** directory in the **`Project`** tab and select **`New`** → **`Ada Main Unit`**.

2. In the **`Create Ada Main Unit`** dialog box, enter **`test_analog_input`** and click **`OK`**.

3. In the **`Confirmation`** dialog box, click **`No`**. A new tab with a stub for **`test_analog_input`** will appear.

4. You will want to copy bits and pieces from **`test_led.adb`** to **`test_analog_input.adb`**, (most of the **`with`** statements and the declaration of and assigment to **`remdev`**). You should also add "**`with RemoteIO.LPC1114`**" to obtain identifiers for the available Remote I/O resources. Replace "**`with GPIO.RemoteIO`**" with "**`with Analog`**" and "**`with ADC.RemoteIO`**".

5. The constructor **`ADC.RemoteIO.Create`** requires two parameters: A remote device object instance (*i.e.* **`remdev`**) and a Remote I/O ADC channel number (*i.e.* **`RemoteIO.LPC1114.AIN1`**) and returns an analog input object instance of type **`Analog.Input`**.

6. To capture an analog input sample, use the **`Get`** method of **`Analog.Input`**.

7. To display analog sampled data values, use **`Analog.Sample_IO.Put`**.

8. When you are ready to compile, do **`Build`** → **`Project`** → **`Build <current file>`**.

9. When you are ready to test, do View → OS Shell and then enter one of the commands:

`.\test_analog_input` ***or*** **`./test_analog_input`**

Hint: For help getting started, you can try peeking at:

`libsimpleio/ada/programs/remoteio/test_adc.adb`