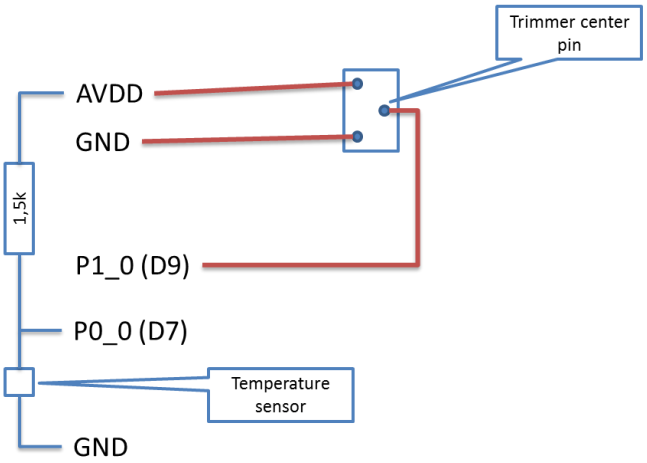
Introduction

ADC conversion

Wire a test board. Test board has a trimmer that is used to adjust the voltage into ADC channel 8 and a temperature sensor (+ auxiliary 1.5k resistor) that is connected to ADC channel 10.

Ask the instructor for trimmer, temperature sensors and the resistor.



Temperature sensor type is KTY81/210. Sensor has a positive thermal coefficient. The measuring circuit is a simple voltage divider where the voltage on the ADC input increases with temperature.

Exercise 1 - Introduction

Create a C++ project. Copy the example file from Tuubi to your project, modify the project to use ITM printing and test that you get proper readings from ADC. You should get values between 0 and 4095 by adjusting the trimmer. The temperature sensor should give values that are around 2500 in room temperature. The readings should get bigger when you warm the sensor in your hands.

Exercise 2 - Tuner

Write a program that helps user to match ADC-value from the channel with the trimmer to the value form channel with the temperature sensor. How close the values are to each other is indicated by blinking leds. When the trimmer reading is lower than the temperature reading the blue led is blinking. The blinking frequency depends on the difference between the channels. The bigger the difference the slower the blinking rate is. The red led blinks when the trimmer reading is higher than the temperature reading and the blinking frequency depends on the difference between channels. When the two readings match with reasonable accuracy then the green led is constantly on (blue and red are off).

Summary:

- Green led on when the readings match

- Red led blinks when trimmer reading is higher than temperature
- Blue led blinks when trimmer reading is lower than temperature
- Blinking frequency depends on difference:
    - big difference → low frequency
    - small difference → high frequency

Other requirements:

- ADC sample rate must be ~10 Hz (~100 ms intervals)
    - Use Sleep() for timing. It is not absolutely accurate but it is good enough for this purpose
    - Print ACD values and blink rate to ITM console
- Led blinking must be implemented in the Systick interrupt
- Blinking frequency must change smoothly
- Do not print anything in Systick_Handler (or any ISR in general)

Instructions:

- Read ADC values and do calculations in the main program
    - Set the variables that determine blinking frequency in the main
    - Remember to use critical sections when modifying variables, that are shared with an ISR, in the main program
- Do the led blinking in the ISR

Exercise 3 –Interrupt driven ADC and filtering

Change the previous program to use fully interrupt driven ADC sampling and add averaging to get more stable readings.

Move `Chip_ADC_StartSequencer(LPC_ADC0, ADC_SEQA_IDX);` to Systick_Handler. Use a counter in the Systick_Handler to start the sequencer at 100 ms intervals.

Add ADC interrupt handler and enable the ADC interrupts. Note that also this handler must be wrapped inside **extern "C" { }** for linker to find the handler. The ADC interrupt handler gets called when conversion is complete. The conversion takes very little time. With properly written ISR there is no risk of getting and overrun.

```
void ADC0A_IRQHandler(void)
{
            uint32_t pending;

            /* Get pending interrupts */
            pending = Chip_ADC_GetFlags(LPC_ADC0);

            /* Sequence A completion interrupt */
            if (pending & ADC_FLAGS_SEQA_INT_MASK) {
                        /* Read the ADC values here */
            }

            /* Clear any pending interrupts */
            Chip_ADC_ClearFlags(LPC_ADC0, pending);
}
```

Enable ADC interrupt in NVIC by adding the following line after `Chip_ADC_EnableInt` in your source file.

```
/* Enable related ADC NVIC interrupts */
NVIC_EnableIRQ(ADC0_SEQA_IRQn);
```

Calculate an average of three samples to reduce noise in the readings. Handle all printing in the main program. Printing in an ISR consumes too much time and will ruin all real time performance.