L_03

Hardware I/O – Analog inputs control

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this laboratory work is to get familiarized with the setup and use of analog inputs of a LPC1769 microcontroller.

1.2 Objective

(1) Implementation of an embedded C application for controlling the digital outputs based on the value read from a potentiometer connected to the analog AD5 port of the microcontroller.

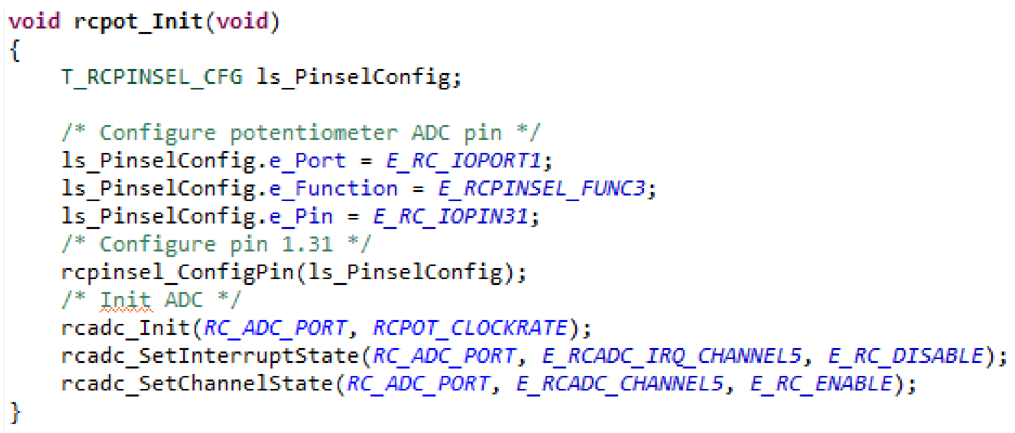
2. General Description

The AOAA Board contains a variable resistance (potentiometer), placed near the RGB LEDs, which is connected to the analog AD5 port of the microcontroller and is configurable for custom applications. The signal provided by the potentiometer is a variable voltage between 0V and 3V. The GPIO pins associated with this analog input is P1_31.

2.1 Software template description

The software template contains all the setups to run a standalone application for reading the analog AD5 input (potentiometer value).
This template has two important sections in the code:

- the GPIO analog input initialization function;
- the potentiometer task running continuously from "*tasks\rc_task_pot\rc_task_pot.c*"



```c
void rcpot_Init(void)
{
    T_RCPINSEL_CFG ls_PinselConfig;

    /* Configure potentiometer ADC pin */
    ls_PinselConfig.e_Port = E_RC_IOPORT1;
    ls_PinselConfig.e_Function = E_RCPINSEL_FUNC3;
    ls_PinselConfig.e_Pin = E_RC_IOPIN31;
    /* Configure pin 1.31 */
    rcpinsel_ConfigPin(ls_PinselConfig);
    /* Init ADC */
    rcadc_Init(RC_ADC_PORT, RCPOT_CLOCKRATE);
    rcadc_SetInterruptState(RC_ADC_PORT, E_RCADC_IRQ_CHANNEL5, E_RC_DISABLE);
    rcadc_SetChannelState(RC_ADC_PORT, E_RCADC_CHANNEL5, E_RC_ENABLE);
}
```

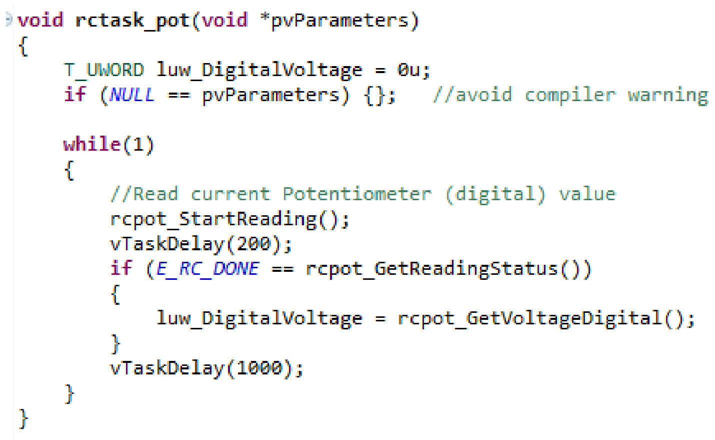
Figure 1. Potentiometer module initialization

In the "*rcpot_Init*" function, the application configures the GPIO pin P1_31 to accept analog input via AD5 channel (*E_RCADC_CHANNEL5*). The ls_PinselConfig structure holds the configuration of the pin function (ADC), which is set using the "*rcpinsel_ConfigPin*" function.

The ADC port is initialized using the driver function "*rcadc_Init*" having the sampling frequency as the second parameter. The ADC channel 5 is enabled using the function "*rcadc_SetChannelState*".

In this template application we don't use the ADC interrupts, therefore they are disabled at initialization using the function "*rcadc_SetInterruptState*".

The current value of the potentiometer is read in the "*rctask_pot*" function (Fig. 2)



```
void rctask_pot(void *pvParameters)
{
    T_UWORD luw_DigitalVoltage = 0u;
    if (NULL == pvParameters) {};   //avoid compiler warning

    while(1)
    {
        //Read current Potentiometer (digital) value
        rcpot_StartReading();
        vTaskDelay(200);
        if (E_RC_DONE == rcpot_GetReadingStatus())
        {
            luw_DigitalVoltage = rcpot_GetVoltageDigital();
        }
        vTaskDelay(1000);
    }
}
```

Figure 2. Potentiometer task

The analog-to-digital conversion process is started manually using "*rcpot_StartReading*" function. The conversion status can be checked using "*rcpot_GetReadingStatus*" function and it can have the values "*E_RC_DONE*" or "*E_RC_NOTDONE*".

Once the status of the reading is E_RC_DONE, the converted value can be read from the register using the function "*rcpot_GetVoltageDigital*". The values returned by this function can be between 0 and 4095 digital. The value "0" corresponds to an analog signal of 0V, while the 4095 value corresponds to 3.3V (analog-to-digital converter on 12 bits).

In order to calculate the voltage corresponding to a digital converted value, the following formula shall be used:

$$U_{analog} = \frac{3.3 \cdot U_{digital}}{4095} \tag{2.1}$$

where $U_{digital}$ is the value returned by the function "*rcpot_GetVoltageDigital*".

3. Requirements

Based on the template described in the previous chapter, update the application with the following requirements:

(1) The LEDs task shall contain a state machine which checks for 6 states – one state for each Led. In each state, the corresponding LED should be turned ON and the rest of the LEDs shall be turned OFF. Each state shall switch to NONE after the implementation.

(2) The "*modules\rc_pot\rc_pot.c*" file shall contain a function named "*T_UWORD rcpot_GetVoltageAnalog(T_UWORD luw_DigitalVoltage)*" which converts the digital values into millivolts values.

(3) The application shall read the potentiometer values and switch the LEDs state as follows:

	U_{analog} ***[mV]***	**State name**
1	[0, 500)	E_RCLEDS_STATE_RED1
2	[500, 1000)	E_RCLEDS_STATE_GREEN1
3	[1000, 1500)	E_RCLEDS_STATE_BLUE1
4	[1500, 2000)	E_RCLEDS_STATE_RED2
5	[2000, 2500)	E_RCLEDS_STATE_GREEN2
6	[2500, 3300)	E_RCLEDS_STATE_BLUE2