**LAB: Measuring Analog Input using ADC**

**I. Overview**

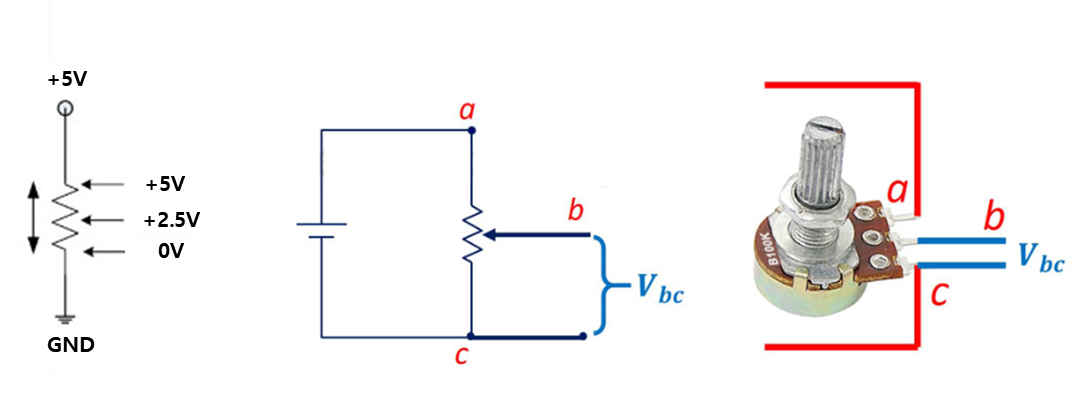
In this lab, we will learn how to configure and use ‘Analog mode’ of GPIO input of the MCU. Then, we will learn how to use ADC to measure the value of potentiometer and to understand the concept of ADC resolution. The sensor needs the VCC and GND of MCU pin.

The objectives of this lab are learning how to

* Read and configure registers of ADC
* Create your own functions for configuring and reading ADC

**Preparation**:

* 5 kΩ potentiometer, 1 kΩ resistor, DMM, breadboard
* You need to study the following registers: ADC register in ‘STM Reference Manual pg. 212 - 239’’
* You need to study the potentiometer and resolution about ADC pin.

  
**Figure 1. Connection of a potentiometer**

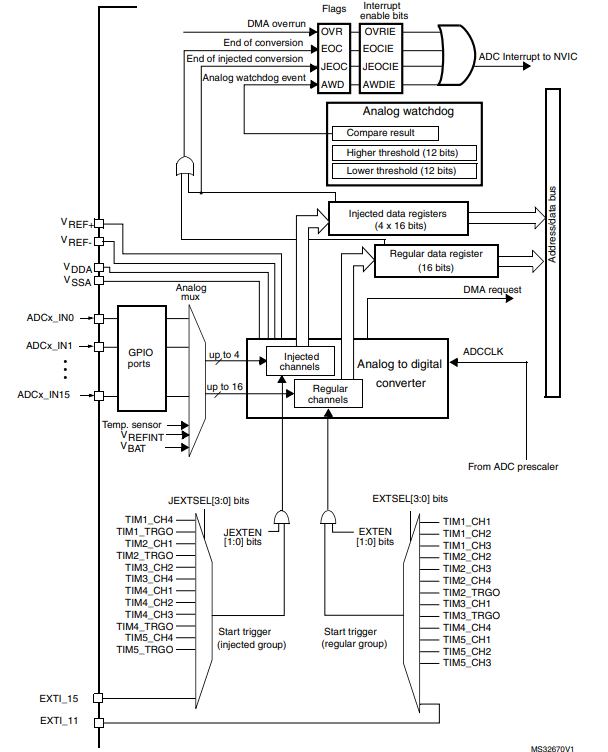
**II. Pre-Lab**

**A. TIMER Register**

* List of TIMx registers for this LAB

|  |  |  |
| --- | --- | --- |
| Type | Register Name | Description |
| RCC | RCC\_AHB1ENR | Enable the clock of GPIO Port |
| RCC\_APB2ENR | Enable the clock of ADC |
| GPIO | GPIOx\_MODER | Analog |
| GPIOx\_PUPDR | No Pull-Up, No Pull-Down: |
| ADCx | ADCx\_ CCR | Select the prescaler of ADC clock frequency |
| ADCx\_CRx | Start one ADC conversion with resolution setting |
| Choose the data alignment (right/left). |
| ADCx\_SQRx | Configure the sequence of conversation |
| ADCx\_SMPRx | Decide the sampling time of conversation |
| ADCx\_SR | Read the EOC (end of conversion) bit |
| ADCx\_DR | Read the converted data from ADC pin |

* Single ADC block diagram



* Process of ADC register initiation (for ADC mode to measure the voltage of potentiometer)

|  |
| --- |
| **GPIO Pin setting**  1. Set RCC (**AHB1ENR**)  2. Analog mode selection for Pin\_y in GPIOx  3. Set the GPIO pin configuration as No pull up, No pull down  **ADC setting**   1. Enable ADC peripheral clock (**RCC\_APB2ENR)** 2. Configure clock prescaler. **(ADC\_ CCR)** 3. Configure ADC data resolution and alignment. **(ADC1\_CR1, CR2)** 4. Enable continuous conversion mode **(ADC1\_CR2)** 5. Configure the number of conversions. **(ADC1\_SQR1)** 6. Configure the channel and sequence of conversion. **(ADC1\_SQRn)** 7. Configure the sampling time with choosing the number of cycles. **(ADC1\_SMPRn)** 8. Enable ADC and start conversion. **(ADC1\_CR2)**   **Read ADC value (Polling)**   1. Wait until the conversion ends **(ADC1\_SR)** 2. Read the data register of ADC **(ADC1\_DR)**   **Read ADC value (Interrupt)**   1. Enable EOC interrupt. **(ADC1\_CR1)** 2. Set the priority of ADC\_IRQn and enable interrupt. (**NVIC\_Setpriority, NVIC\_EnableIRQ)** 3. Create user code in ‘void ADC\_IRQHandler()’ to read the converted data from ADC pin. |

**B. Register Setting**

**1. GPIO Pin Initialization**

* ADC pin: **Port A pin 0** / Analog / No pull-up & No pull-down.
* Read appendix for ADC pin map

**2. ADC Initialization**

* ADC: Prescaler /2, 12-bit resolution, right alignment, continuous conversation mode, one channel scan in regular group, channel sample time selection 84-cycles
* Use your own functions created in previous lab for GPIO configuration.
* **RCC\_APB2ENR:** Enable ADC peripheral clock

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  | **20** | **19** | **18** | **17** | **16** |  | **14** | **13** | **12** | **11** |  |  | **8** |  |  | **5** | **4** |  |  |  | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC\_ CCR:** ADC prescaler to select ADC clock frequency (PCLK2 divided by 2)

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  | **23** | **22** |  |  |  |  | **17** | **16** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC1\_CR1:** 12-bit resolution

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  | **26** | **25** | **24** | **23** | **22** |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC1\_CR2:** Right alignment, Continuous conversion mode enable

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |  |  |  |  | **11** | **10** | **9** | **8** |  |  |  |  |  |  | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC1\_SQR1:** 1 conversion in the regular channel

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |  |  |  | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC1\_SQR3:** Configure the 1st conversion to channel 0

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC1\_SMPR2:** Channel 0 sampling time - 84 ADC clock cycles

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **ADC1\_CR2:** Enable ADC / Start conversion

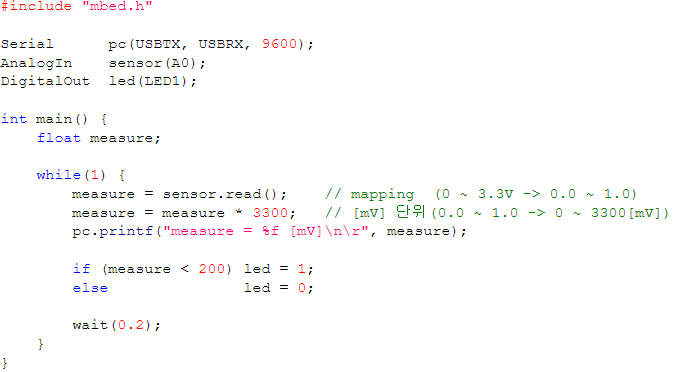
|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |  |  |  |  | **11** | **10** | **9** | **8** |  |  |  |  |  |  | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**III. Tutorial**

**A. Mbed online compiler**

* Open <https://www.mbed.com/en/> and create new program ‘Tutorial\_ADC’
* Open ‘main.cpp’ and delete the example codes.
* Write the following code on ‘main.cpp’.



* Click on ‘**Compile’** button. Then, the binary file will be created and downloaded on your computer.
* Connect the MCU board to your PC. Copy and paste the downloaded binary file to the drive “NODE\_F411RE (E:)”.
* Open ‘Tera Term’ and verify the performance. Check the output voltage of photo sensor.

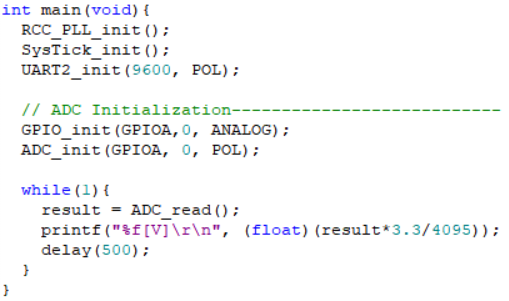
**IV. Exercise**

1. **Create your own functions for ADC mode**

* Create your own function to initialize ADC pin.
* Below are some of the examples of necessary functions. Attach your codes in Appendix.
* If necessary, you can also create other functions to configure timer input capture mode.

|  |  |  |
| --- | --- | --- |
| **File** | **Example Code** | **Description** |
| myGPIO.h | #define ANALOG 3 | You can use this variable in ‘analog mode’ configuration |
| **File** | **Example Code** | **Description** |
| myADC.h, myADC.c | uint32\_t ADC\_pinmap(GPIO\_TypeDef \*port, uint32\_t pin); | Return the number of ADC channel depending on GPIO port and pin.  Refer to Appendix **2. ADC GPIO pinout** |
| void ADC\_init(GPIO\_TypeDef \*port, uint32\_t int pin); | // Default Setting Set the ADC clock frequency as PCLK2/2 Resolution: 12-bit Right Alignment and Continuous conversion mode  Configure 1st conversion to the selected channel ID in regular sequence. (*Use function ADC\_pinmap()*) Sampling time: 84 clock cycles. POL(polling) |
| *Other configuration functions* | *Other configuration functions* |
| uint32\_t ADC\_read(); | Read the converted data by polling method |

Sample code:



1. **Create a program to measure the analog voltage using a potentiometer**

* The program needs to

1. Configure ADC on port A pin 0.

* ADC clock frequency as PCLK2/2
* Mode: Continuous conversion mode
* Data format: 12-bit resolution and right alignment
* Sequence: Configure 1st conversion to channel n in regular sequence.
* Sample time: 84 clock cycles

1. Initialize LED(port A, pin 5) pin.
2. Connect a potentiometer with 3.3V, GND and port A pin 0.
3. Check the voltage in with DMM.
4. Measure the analog voltage by reading the ADC data register.
5. Display measured voltage in [V] on serial monitor of Tera-Term.
6. Turn off LED if measured voltage is lower than 1[V]. Also, turn on LED if measured voltage is higher than 2[V].
7. **Create a program to measure multiple analog voltage channels**

* The program needs to

1. Connect a 3.3V source to a resistor of 1kΩ and potentiometer in series.
2. Configure ADC of Channel 0 and Channel 1.

* Same setting as Q2, except, it should scan with a group of 2 channels

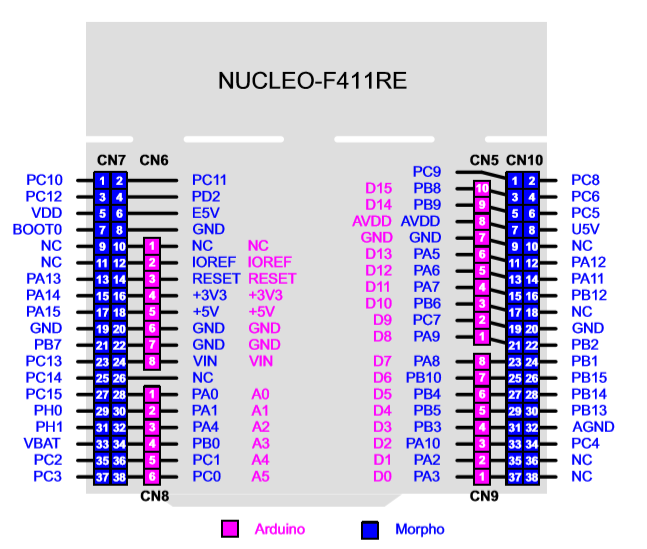
1. Measure Analog in Ch0 across the potentiometer-ground, Ch1 across the resistor-ground.
2. ADC register (ADC\_DR) should be read in ADC\_IRQHandler() in sequence of the channel order. It means you have to enable and use EOC interrupt. Also, set the order of conversion as(Ch0🡪Ch1) in ADC\_SRQ .
3. Also, measure each voltage with DMM
4. Display measured voltage in [V] on serial monitor of Tera-Term.

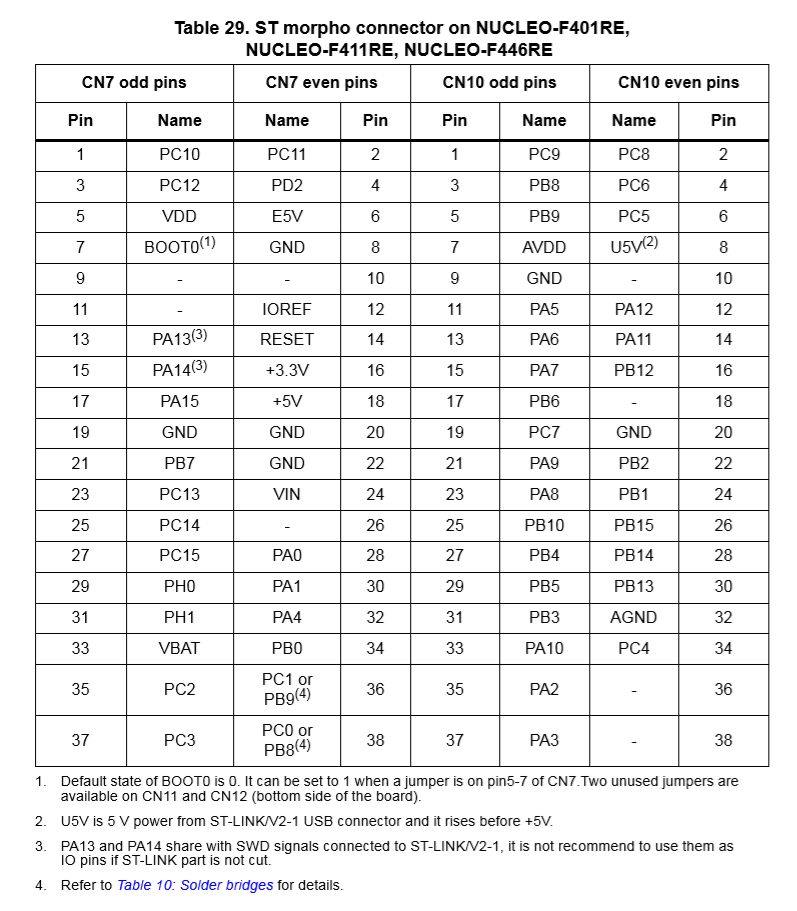
**V. Questions**

1. When the voltage output from the potentiometer-based voltage divider is lower than 1.0V, the LED is turned off. What constant value should the ADC DR register be compared with?
2. When the voltage output from the potentiometer-based voltage divider is higher than 2.0V, the LED is light up. What constant value should the ADC DR register be compared with?
3. In this lab, what is the accuracy of the potentiometer voltage you have measured with ADC? Show your experiment result and compare it with exact voltage measured by a DMM.

**Appendix**

1. **Pin Configuration of NUCLE-F411RE**

****

****

1. **ADC GPIO pinout for STM32f411**

