



# Tutorials for STM32F103RB

Nicolas Barbot, Edouard BURTZ

2017

# Tutorials for STM32F103RB

2

- Create an OpenSTM32 Project
- Create a STM32CubeMX Project
- GPIO
- UART
- Timer
- Analog to Digital Conversion
- Digital to Analog Conversion
- FLASH

- Ressources

- Nucleo F103RB

- [http://www.st.com/content/st\\_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo/nucleo-f103rb.html](http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo/nucleo-f103rb.html)
    - This page contains user manual UM1724 describing the hardware of Nucleo boards

- STM32F103RB

- [http://www.st.com/content/st\\_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f1-series/stm32f103/stm32f103rb.html](http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f1-series/stm32f103/stm32f103rb.html)
    - This page contains reference manual RM0008 describing STM32F103RB

- STM32 Cube F1

- [http://www.st.com/content/st\\_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software/stm32cubef1.html](http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software/stm32cubef1.html)
    - This page contains UM1850 describing the HAL library

- Open STM32

- <http://www.openstm32.org/HomePage>
    - Open STM32 is a free Eclipse plugin for programming STM32 MCU (registration needed).

- Eclipse

- <https://www.eclipse.org>
    - Eclipse is the IDE software

- Hardware Shield

- ##### TBD #####
    - This page describe the shield hardware for Nucleo board (Arduino connector)



# Using ADC

- Goals:
  - Identify adc from the datasheet
  - Configure and use timers with the ADC library
  - Use the ADC to verify an analogic level and trigg an alarm
- Requirement
  - Create an OpenSTM32 Project
  - GPIO

- Open Reference Manuel RM0008 [2]
  - This document describes the STMF103RB.
  - Two 12 bits Analog to Digital Converter (ADC1 & ADC2)
  - 16 external channels for each ADC
  - Conversion in single shot or scan mode, possibility of DMA
- In this tutorial, we will configure the ADC1 in mode Single continuous (simplest mode) to acquire voltage value.

# Create the Project 7

- Create a new project for STM32F103RB using HAL library
  - See “CreateOpenSTM32Project” presentation
- Add the SystemClock\_Config function in main.c
  - Set HCLK to 64 MHz
- Call SystemClock\_Config and HAL\_Init function in main().

- Open Hardware shield schématic document [4]
  - This document describes the hardware layout of shield connected on arduino connector of the Nucleo boards.
- This shield provide different hardware input and output
  - Two adjustable resistors “POT1” and “POT2” which provides 2 analog voltages between 0 and 3,3V
  - A bicolour Led (Red and orange) “LED1” and “LED2”
- Configure the GPIO:
  - Configure the GPIO output for driving the bicolour Led
  - Configure the GPIO Input for acquire an analogic voltage on “POT1” (GPIO\_MODE\_ANALOG, GPIO\_NOPULL, GPIO\_SPEED\_FREQ\_HIGH)
  - Make sure that the choosen GPIO\_Pin correspond to the STM32 port pin (User Manuel 1724 « STM32 Nucleo-64 boards, page 39)



- Open User Manuel 1850 « Description of STM32F1xx HAL drivers » [3]
  - Chapter 4 (p. 82) describes the use of ADC
  - ADC must be configured at RCC top level. In `HAL_ADC_MspInit` declare a `RCC_PeriphCLKInitTypeDef` variable and set the field as describe in [doc3] page 85. then call `HAL_RCCEx_PeriphCLKConfig()` function.
  - Call `__HAL_RCC_ADCx_CLK_ENABLE()` to enabling clock for the corresponding ADC.
  - You must select the operation mode: Declare an `ADC_HandleTypeDef` global variable . In Main init, affect it with « data alignement right », « scan disable », « continuous mode » and « software trigg » for configuring the ADC in desired mode. Set the corresponding instance.
  - Call `HAL_ADC_Init()`
  - You must select the conversion channel and sample time: Declare an `ADC_ChannelConfTypeDef` variable. In Main init, affect it for selecting the corresponding channel , « `ADC_REGULAR_RANK_1` » and « `ADC_SAMPLETIME_41CYCLES_5` »
  - Call `HAL_ADC_ConfigChannel()`
  - Call `HAL_ADC_Start()`
  - In the main loop call `HAL_ADC_PollForConversion ()` to wait for end of conversion then call `HAL_ADC_GetValue()` for getting the converted value

- Test your software
  - Put a breakpoint on the next instruction after reading the converted value. Verify the value in the result variable compared to the analogical value on the analog input. Make several test.
- Manage an alarm according to voltage value on analog input
  - Complete your program with the specification below. Then test it.

Vin	Orange LED	Red LED
$V_{in} < 1V$	OFF	OFF
$1V \leq V_{in} \leq 2V$	ON	OFF
$V_{in} > 2V$	OFF	ON

- Improve your software in adding an hysteresis for the red LED (see specification below). Then test it.

Vin	Red LED
Increase and $V_{in} > 2V$	ON
Decrease and $V_{in} \leq 1.5V$	OFF

- Thanks to this tutorial, you are now able to:
  - Configure an use ADC
  - Convert analogical value for numeric treatment