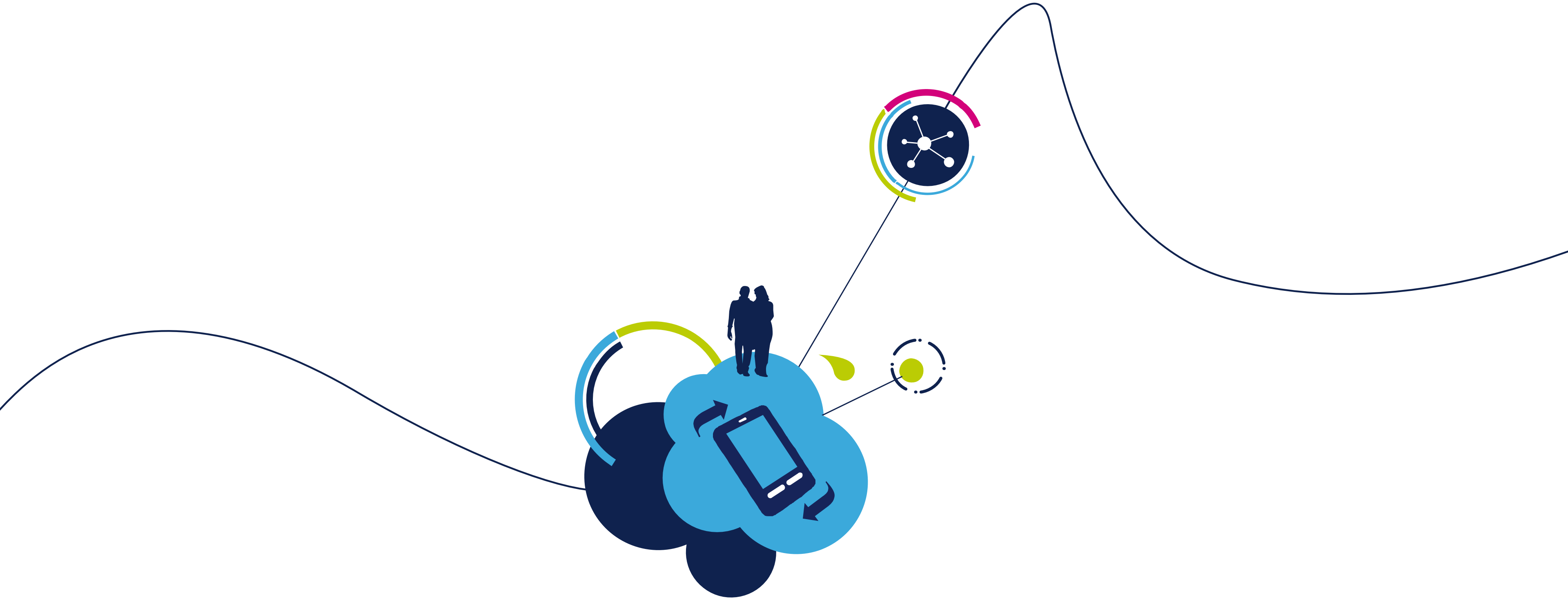
STM32

Training Hands on and Tools overview



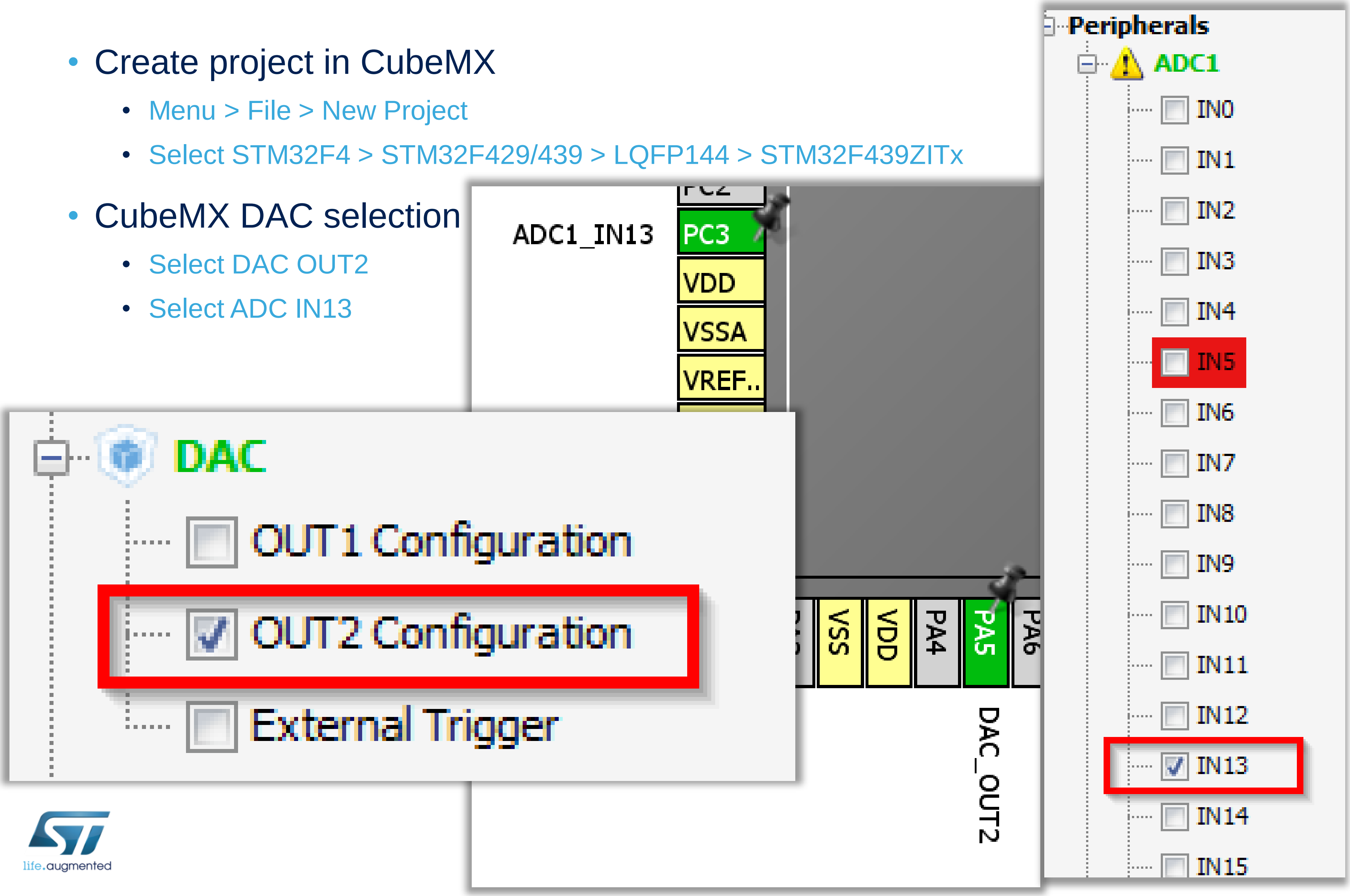
4.1.1 ADC Poll lab

4.1.1 Use ADC in polling mode

- Objective
  - Use the DAC part from previous lab
  - Learn how to setup ADC in CubeMX
  - How to Generate Code in CubeMX and use HAL functions
- Goal
  - Configure ADC in poll in CubeMX and Generate Code
  - Learn how to start ADC and measure the DAC
  - Verify the measured wave in STMStudio (http://www.st.com/web/en/catalog/tools/PF251373 require JAVA)

4.1.1 Use ADC in polling mode

- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F4 > STM32F429/439 > LQFP144 > STM32F439ZITx
- CubeMX DAC selection
  - Select DAC OUT2
  - Select ADC IN13

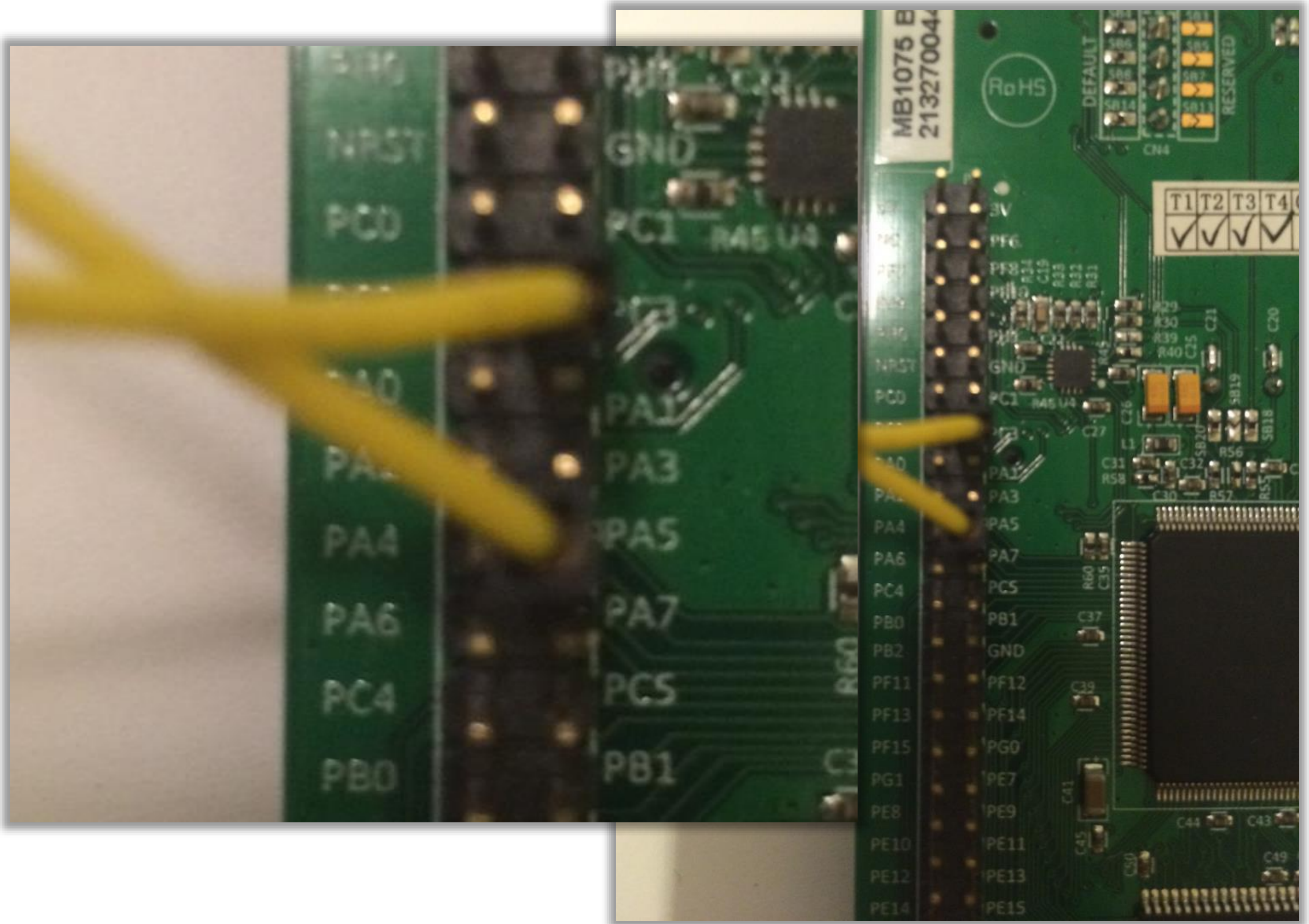


DAC
- ☐ OUT1 Configuration
- ☑ OUT2 Configuration
- ☐ External Trigger

Peripherals
- ADC1
  - ☐ IN0
  - ☐ IN1
  - ☐ IN2
  - ☐ IN3
  - ☐ IN4
  - ☐ IN5
  - ☐ IN6
  - ☐ IN7
  - ☐ IN8
  - ☐ IN9
  - ☐ IN10
  - ☐ IN11
  - ☐ IN12
  - ☑ IN13
  - ☐ IN14
  - ☐ IN15

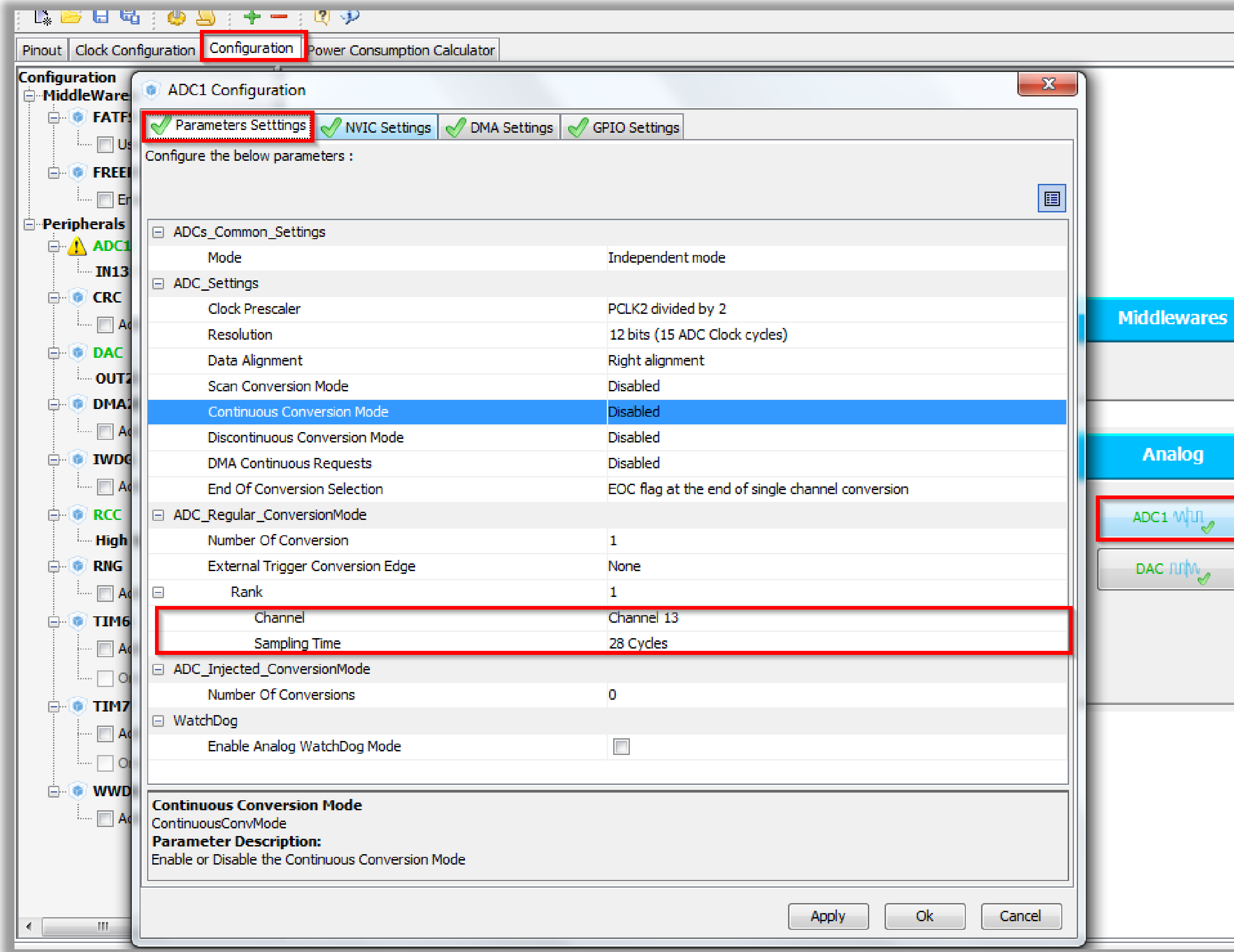
4.1.1 Use ADC in polling mode

- Hardware connection
  - Connect DAC out2 PA5 and ADC1 IN13 PC3 together



4.1.1 Use ADC in polling mode

- CubeMX ADC configuration
  - TAB>Configuration>Analog>ADC1>Parametr Settings
  - Set ADC1
  - Set sampling time for CH13
  - Button OK
- DAC from previous example



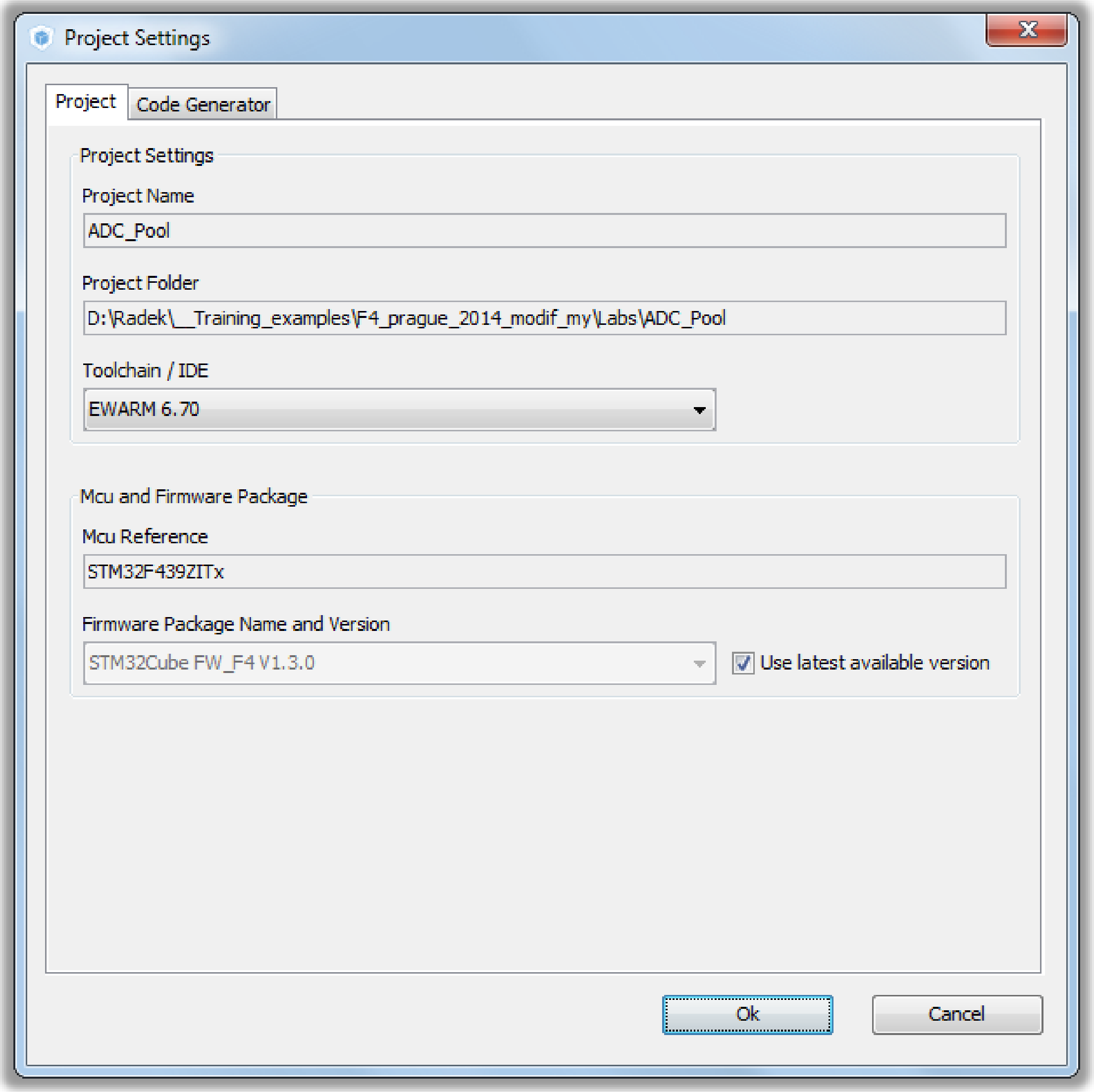
Configure the below parameters :

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 2
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	1
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 13
Sampling Time	28 Cycles
ADC_Injected_ConversionMode	
Number Of Conversions	0
WatchDog	
Enable Analog WatchDog Mode	□

Continuous Conversion Mode
ContinuousConvMode
Parameter Description:
Enable or Disable the Continuous Conversion Mode

4.1.1 Use ADC in polling mode

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Now we can Generate Code
  - Menu > Project > Generate Code



Project Settings

Project Settings

Project Name: ADC_Pool

Project Folder: D:\Radek\__Training_examples\F4_prague_2014_modif_my\Labs\ADC_Pool

Toolchain / IDE: EWARM 6.70

Mcu and Firmware Package

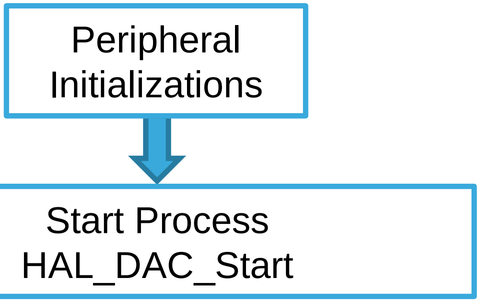
Mcu Reference: STM32F439ZITx

Firmware Package Name and Version: STM32Cube FW_F4 V1.3.0

☑ Use latest available version

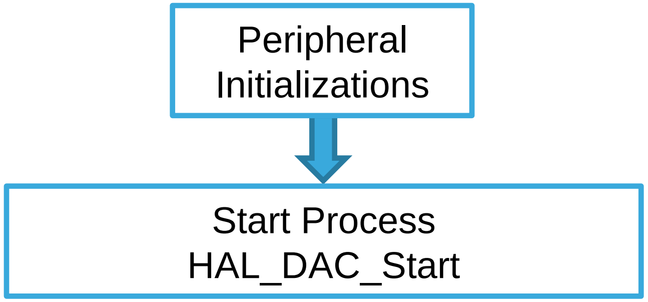
4.1.1 Use ADC in polling mode

- Start process ADC(same for DMA, DAC, TIM)
  - Non blocking start process



Peripheral Initializations

Start Process HAL_DAC_Start



Peripheral Initializations

Start Process HAL_DAC_Start

4.1.1 Use ADC in polling mode

- Open the project in our IDE
  - The functions we want to put into main.c
  - Between */* USER CODE BEGIN 2 */* and */* USER CODE END 2 */* tags
  - and */* USER CODE BEGIN 3 */* and */* USER CODE END 3 */* tags
- For DAC start use function
  - HAL_DAC_Start(DAC_HandleTypeDef* hdac, uint32_t Channel)
  - HAL_ADC_PollForConversion(ADC_HandleTypeDef* hadc, uint32_t Timeout)
  - HAL_ADC_GetValue(ADC_HandleTypeDef* hadc)
- DAC functions
  - HAL_DAC_Start(DAC_HandleTypeDef* hdac, uint32_t Channel)
  - HAL_DAC_SetValue(DAC_HandleTypeDef* hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)

4.1.1 Use ADC in polling mode

- Solution
  - Variables

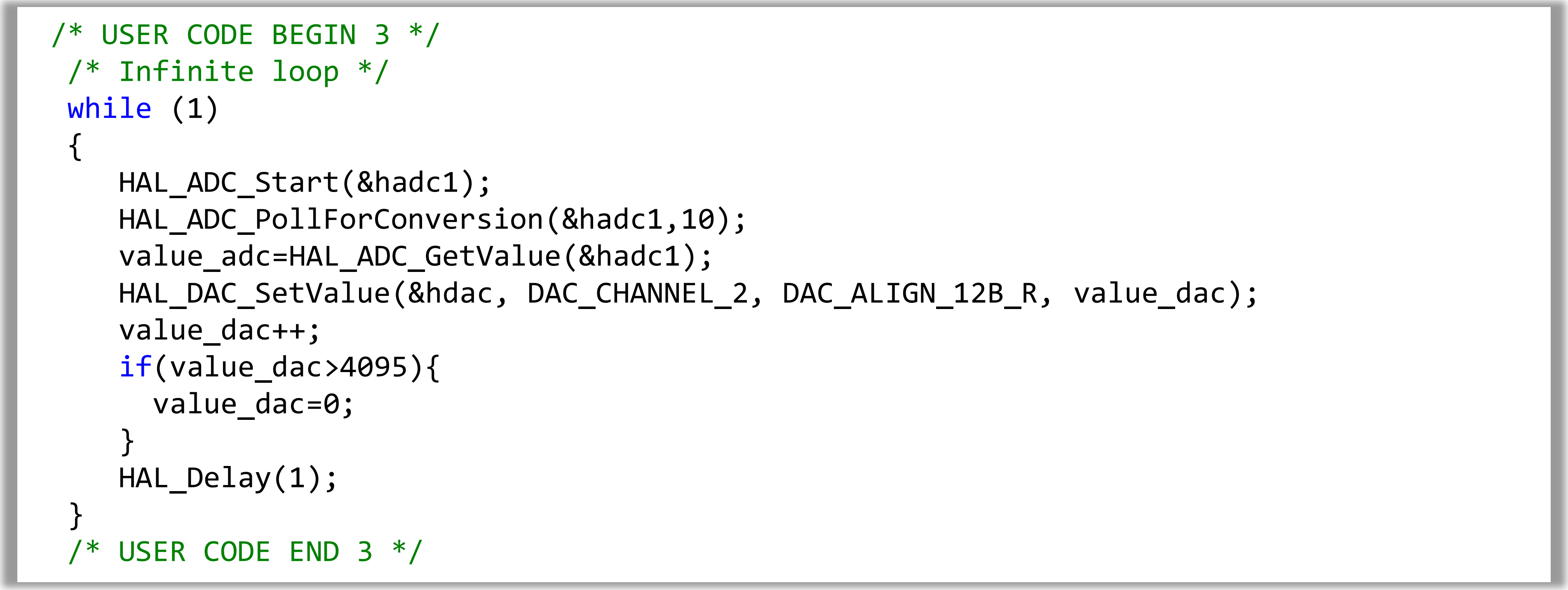
```
/* USER CODE BEGIN PV */
uint32_t value_adc;
uint32_t value_dac=0;
/* USER CODE END PV */
```

  - DAC setup and start

```
 /* USER CODE BEGIN 2 */
  HAL_DAC_Start(&hdac,DAC_CHANNEL_2);
  HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R, value_dac);
  /* USER CODE END 2 */
```

4.1.1 Use ADC in polling mode

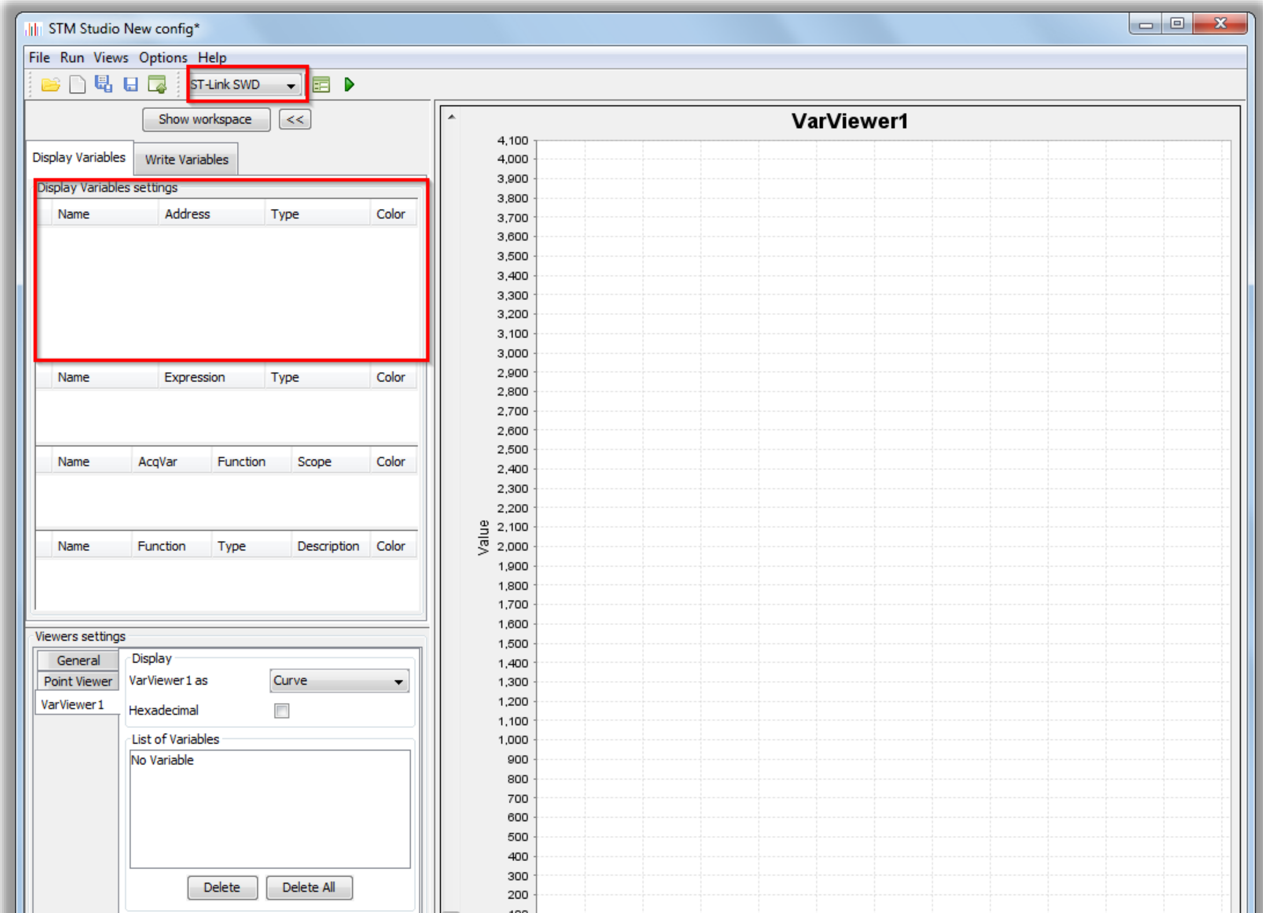
- Solution
  - Main loop with DAC set and ADC set



```
/* USER CODE BEGIN 3 */
 /* Infinite loop */
 while (1)
 {
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,10);
    value_adc=HAL_ADC_GetValue(&hadc1);
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R, value_dac);
    value_dac++;
    if(value_dac>4095){
      value_dac=0;
    }
    HAL_Delay(1);
 }
 /* USER CODE END 3 */
```

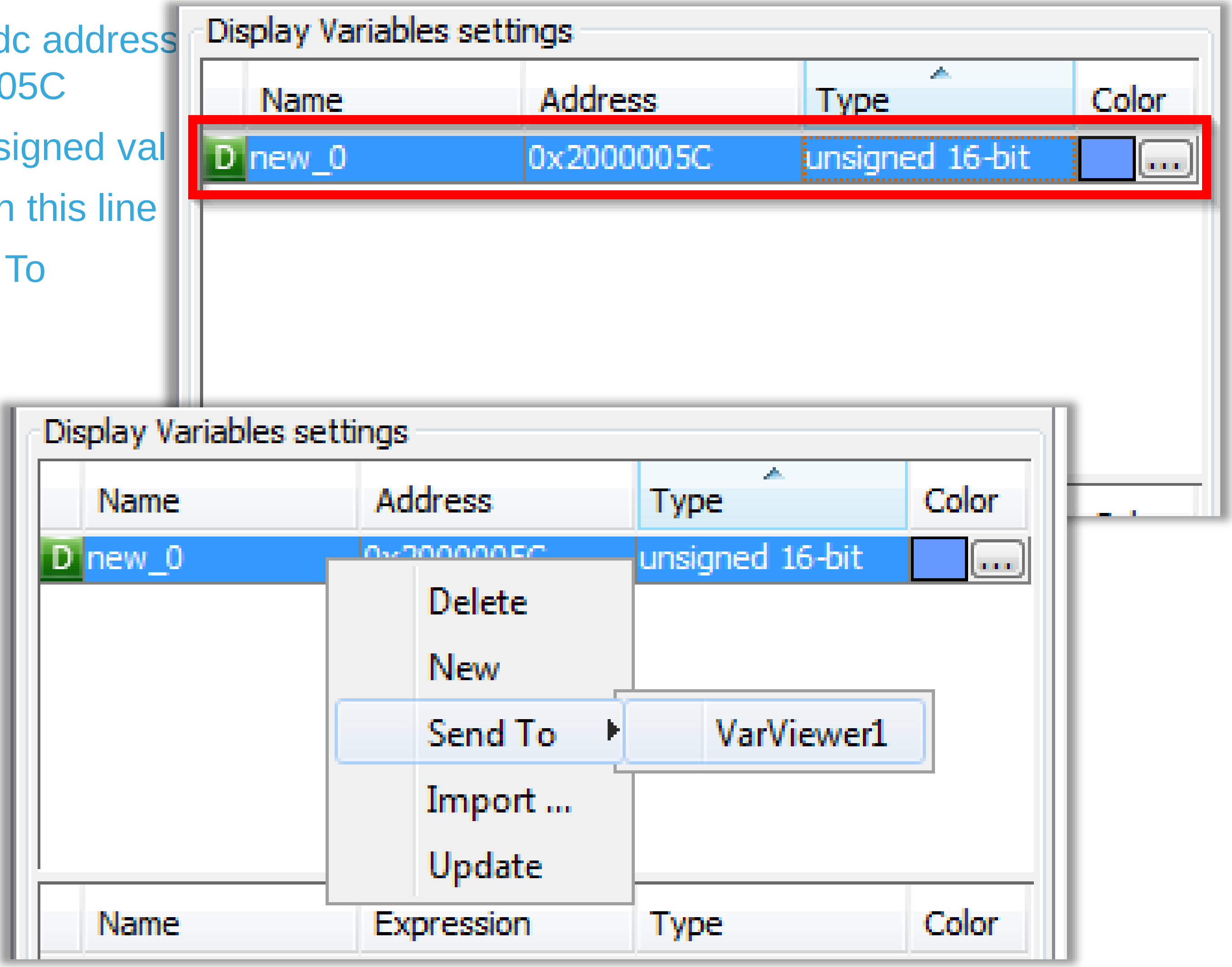
4.1.1 Use ADC in polling mode

- Test the functionality
  - We need the address of variable `value_adc`
  - This can be found usually in debug mode in watch, my address is 0x2000005C (depends on compilator and optimizations)
- Start the STMStudio
  - Set the ST Link SWD
  - Right click into Display variable settings
  - Select NEW



4.1.1 Use ADC in polling mode

- STM studio settings
  - Set value_adc address my 0x2000005C
  - Set 16bit unsigned val
  - Right click on this line
  - Select Send To VarViewer1

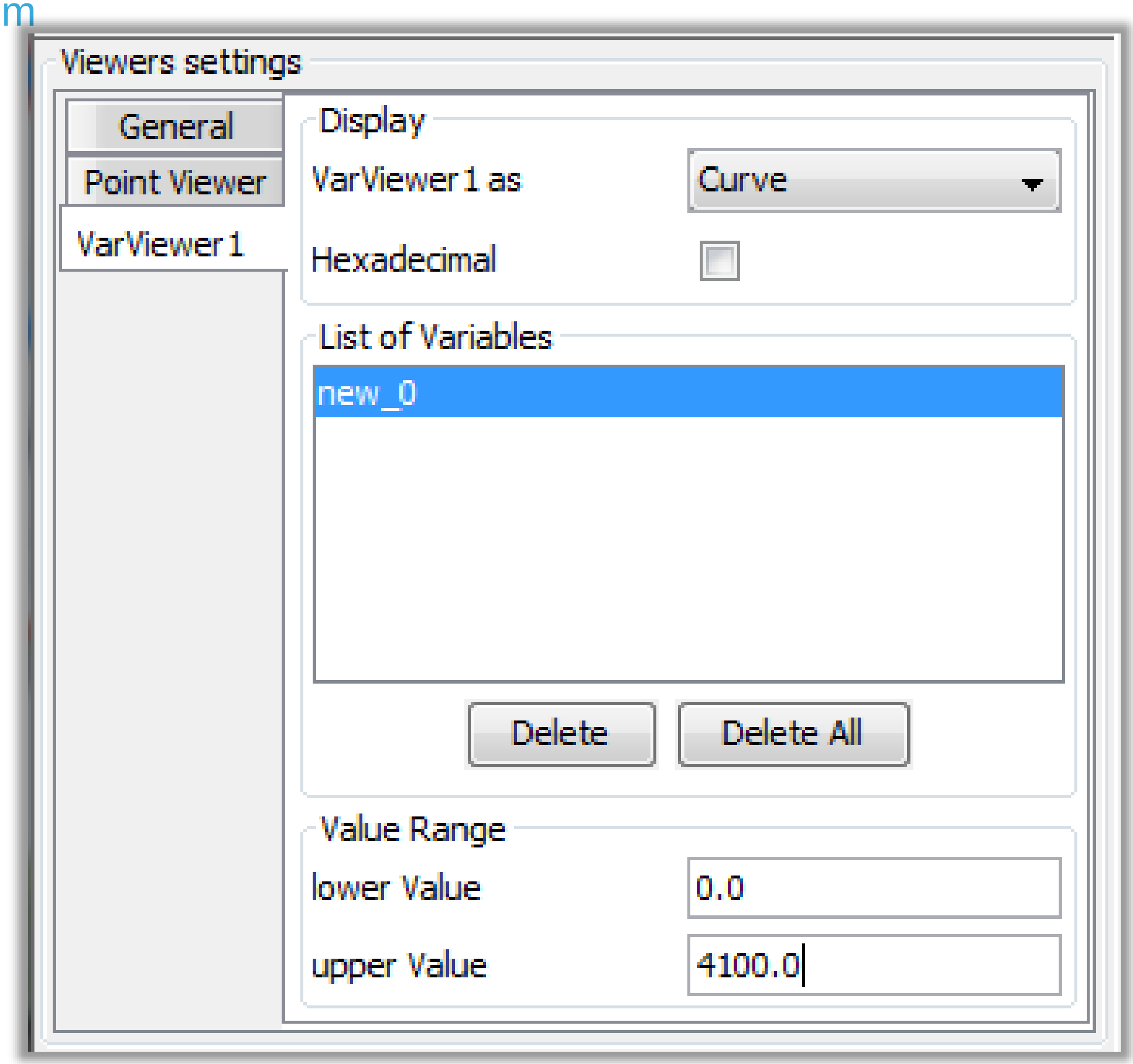


Display Variables settings

	Name	Address	Type	Color
D	new_0	0x2000005C	unsigned 16-bit	🟦 ...

4.1.1 Use ADC in polling mode

- STM studio settings
  - Viewers settings is on bottom
  - Set the correct upper value to 4096(12bit)



Viewers settings

VarViewer1 as: Curve

Hexadecimal: □

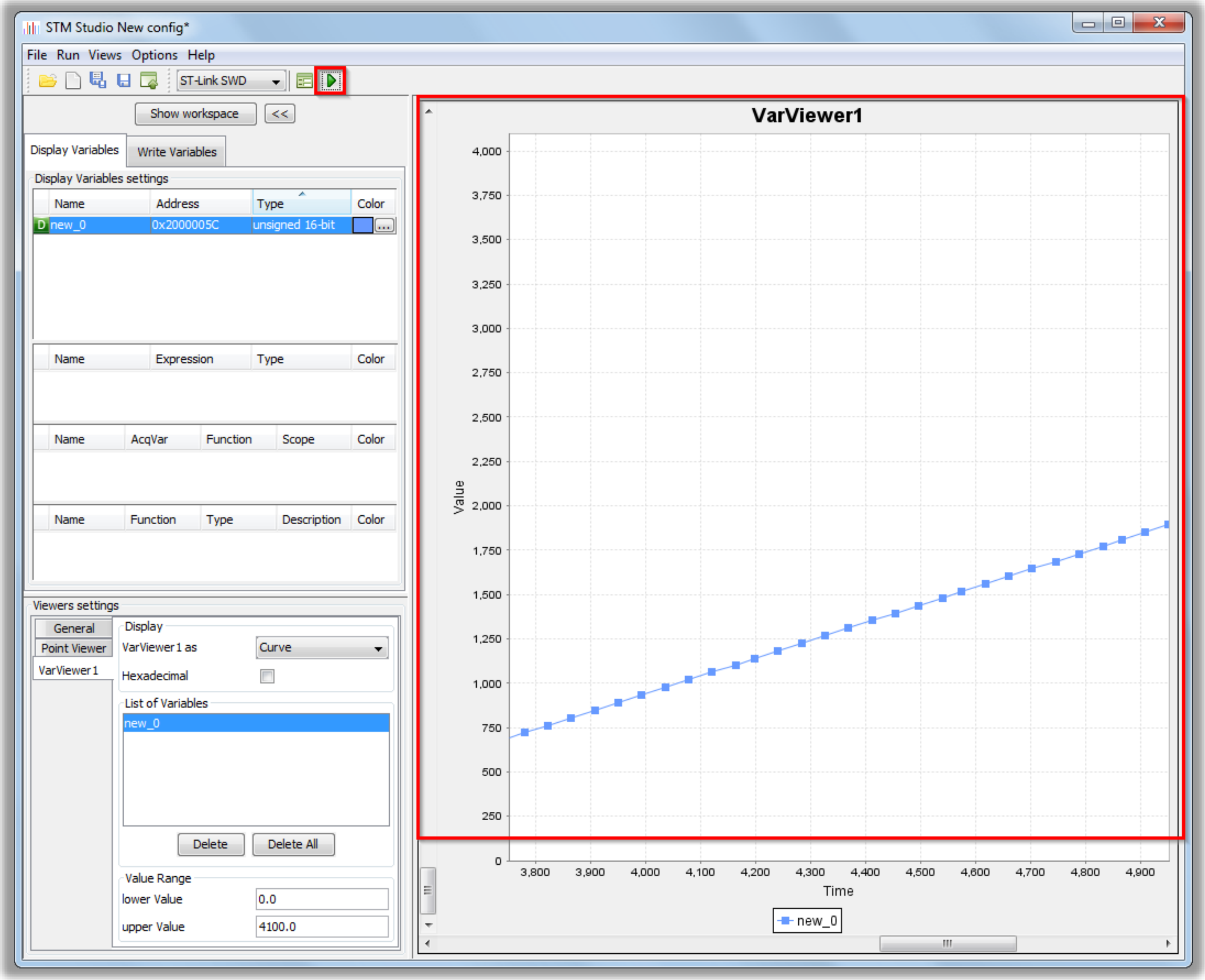
new_0

lower Value: 0.0

upper Value: 4100.0

4.1.1 Use ADC in polling mode

- STM studio settings
  - Now press green play button
  - And you will see content of value_adc



Display Variables settings

	Name	Address	Type	Color
D	new_0	0x2000005C	unsigned 16-bit	...

Name	Expression	Type	Color

Name	AcqVar	Function	Scope	Color

Name	Function	Type	Description	Color

Viewers settings

General
Point Viewer
VarViewer1

Display

VarViewer1 as: Curve

Hexadecimal □

List of Variables

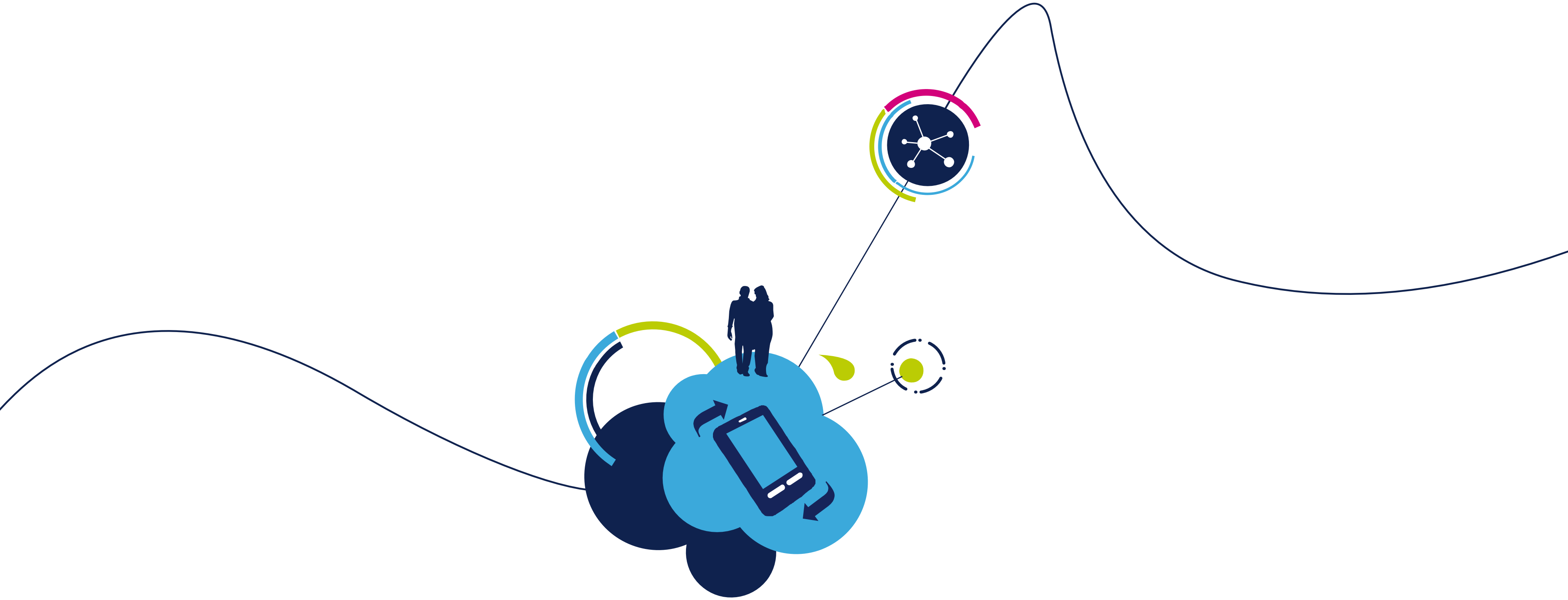
new_0

Delete
Delete All

Value Range

lower Value: 0.0

upper Value: 4100.0



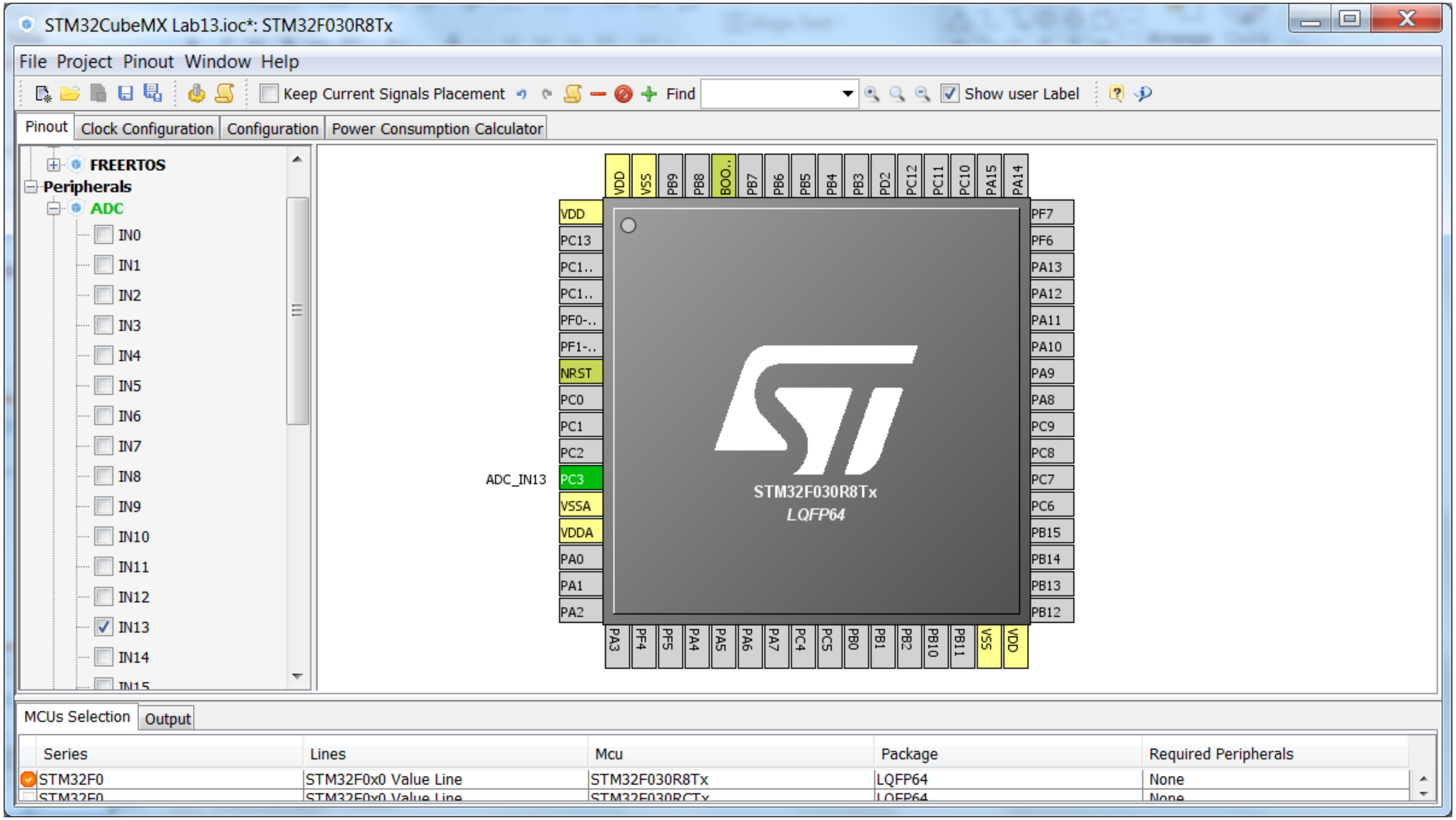
4.1.2 ADC Interrupt lab

4.1.2 Use ADC with interrupt

- Objective
  - Learn how to setup ADC with interrupt in CubeMX
  - How to Generate Code in CubeMX and use HAL functions
- Goal
  - Configure ADC in interrupt in CubeMX and Generate Code
  - Learn how to start ADC
  - Verify the measured wave in STMStudio (http://www.st.com/web/en/catalog/tools/PF251373 require JAVA)

4.1.2 Use ADC with interrupt

- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F0 > STM32F030 > LQFP64 > STM32F030R8



STM32CubeMX Lab13.ioc*: STM32F030R8Tx

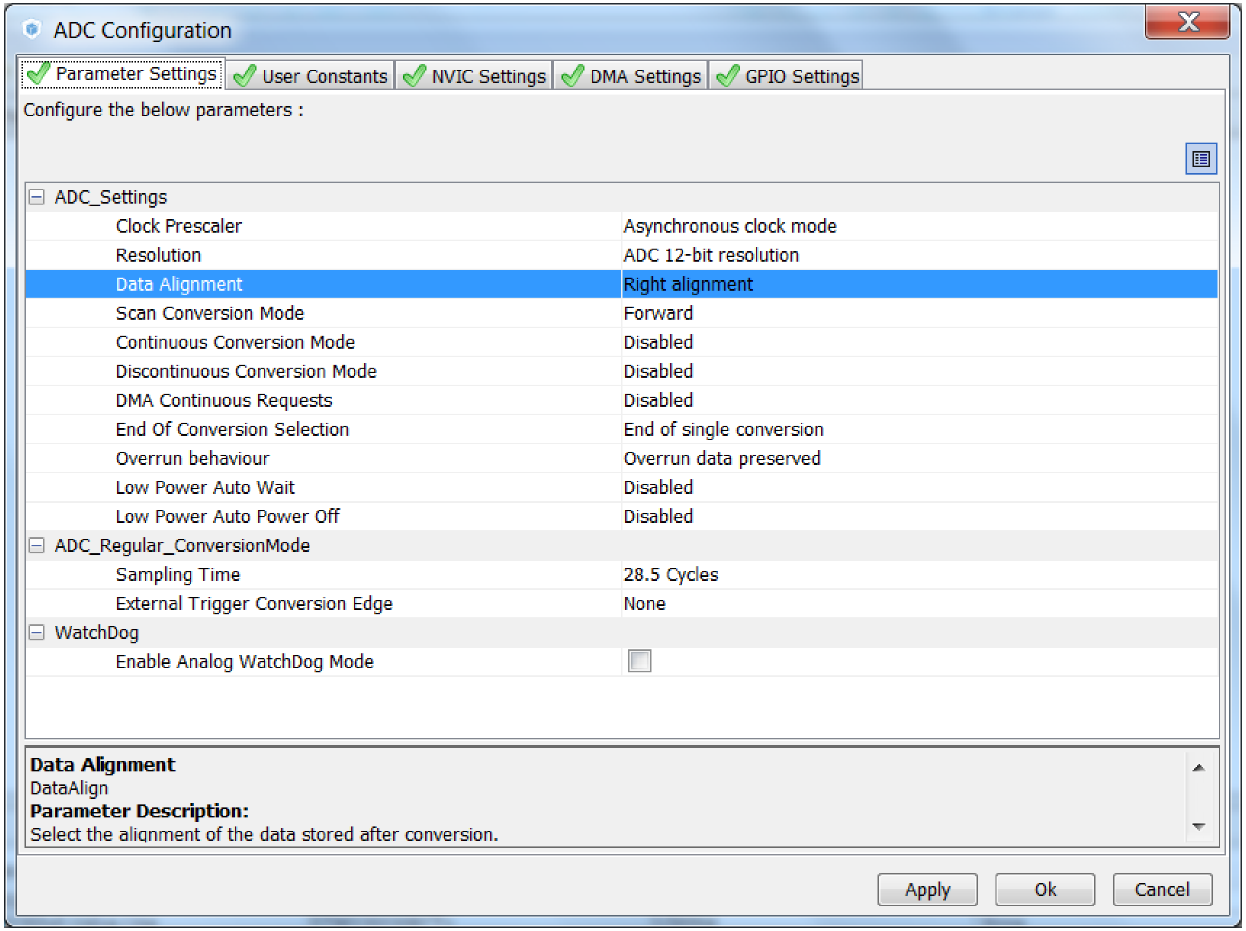
- FREERTOS
- Peripherals
  - ADC
    - □ IN0
    - □ IN1
    - □ IN2
    - □ IN3
    - □ IN4
    - □ IN5
    - □ IN6
    - □ IN7
    - □ IN8
    - □ IN9
    - □ IN10
    - □ IN11
    - □ IN12
    - ☑ IN13
    - □ IN14
    - □ IN15

ADC_IN13 PC3

	Series	Lines	Mcu	Package	Required Peripherals
●	STM32F0	STM32F0x0 Value Line	STM32F030R8Tx	LQFP64	None
	STM32F0	STM32F0x0 Value Line	STM32F030RCTx	LQFP64	None

4.1.2 Use ADC with interrupt

- CubeMX ADC configuration
  - TAB>Configuration>Analog>ADC1>Parametr Settings
  - Set ADC1
  - Set sampling time for CH13
  - Button OK



ADC Configuration

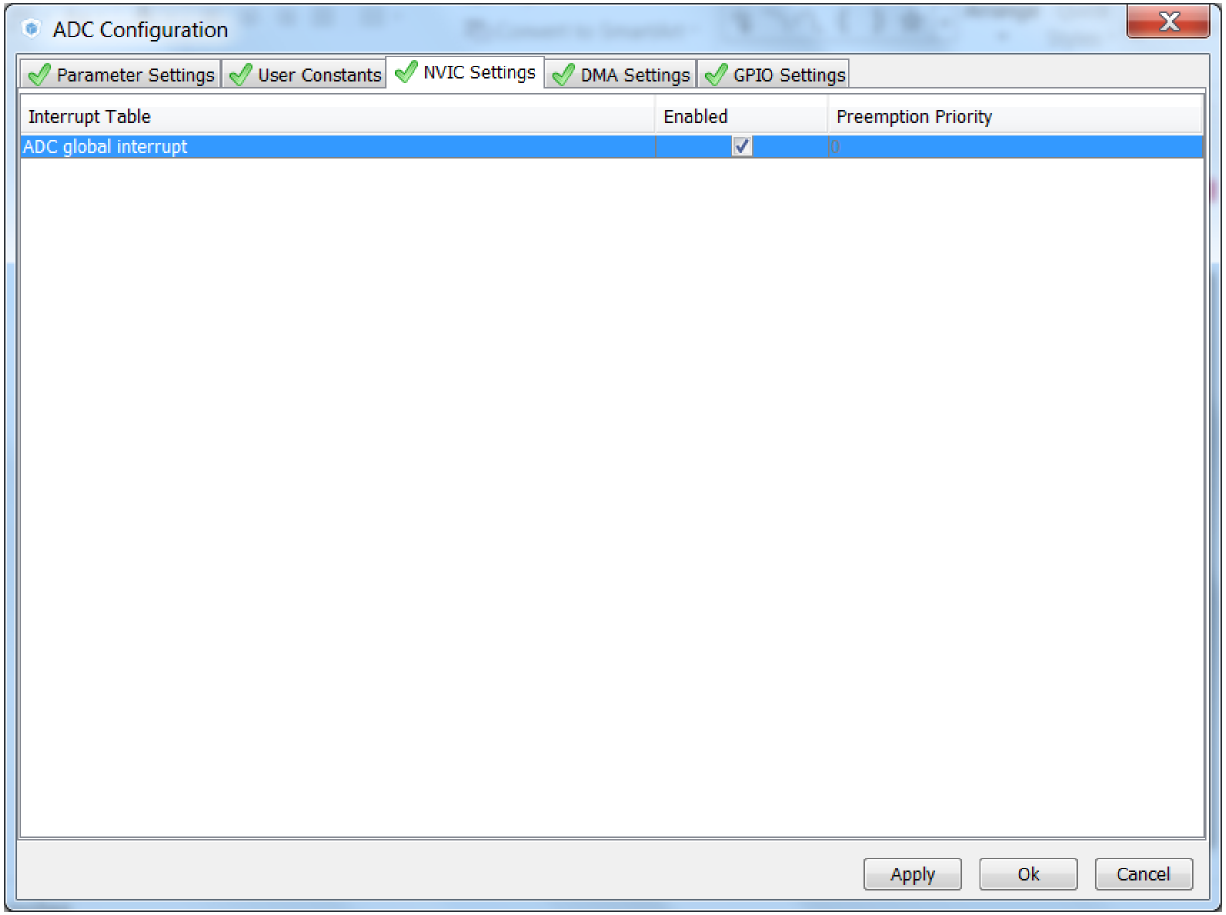
Configure the below parameters :

⊟ ADC_Settings	
Clock Prescaler	Asynchronous clock mode
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Scan Conversion Mode	Forward
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	End of single conversion
Overrun behaviour	Overrun data preserved
Low Power Auto Wait	Disabled
Low Power Auto Power Off	Disabled
⊟ ADC_Regular_ConversionMode	
Sampling Time	28.5 Cycles
External Trigger Conversion Edge	None
⊟ WatchDog	
Enable Analog WatchDog Mode	□

Data Alignment
DataAlign
Parameter Description:
Select the alignment of the data stored after conversion.

4.1.2 Use ADC with interrupt

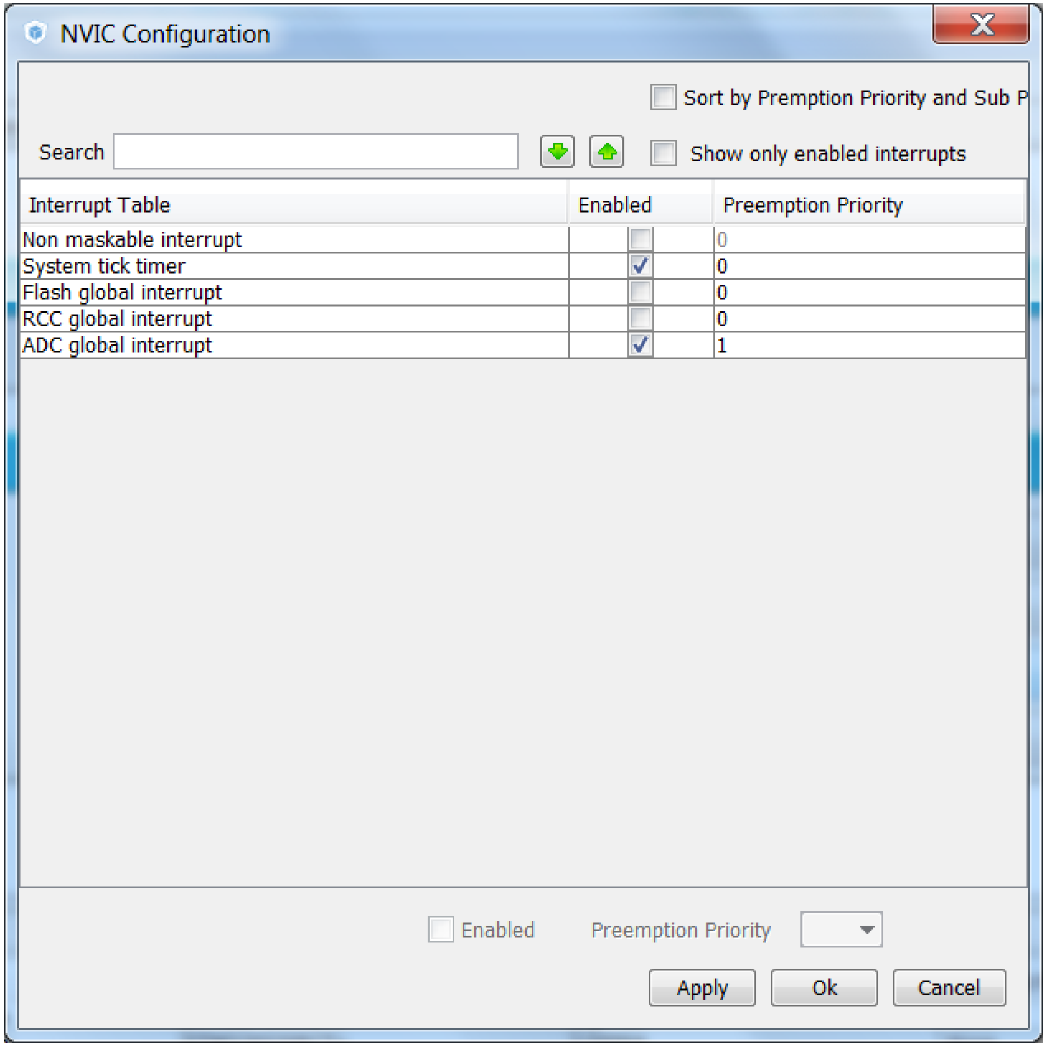
- CubeMX ADC configuration
  - TAB>NVIC settings
  - Enable ADC1 interrupt
  - Button OK



Interrupt Table	Enabled	Preemption Priority
ADC global interrupt	☑	0

4.1.2 Use ADC with interrupt

- CubeMX NVIC configuration
  - Because we want use the Systick for delay in interrupt The ADC interrupt priority must be changed
  - TAB>Configuration>System>NVIC
  - Change ADC1 preemption priority to 1



NVIC Configuration

□ Sort by Premption Priority and Sub P

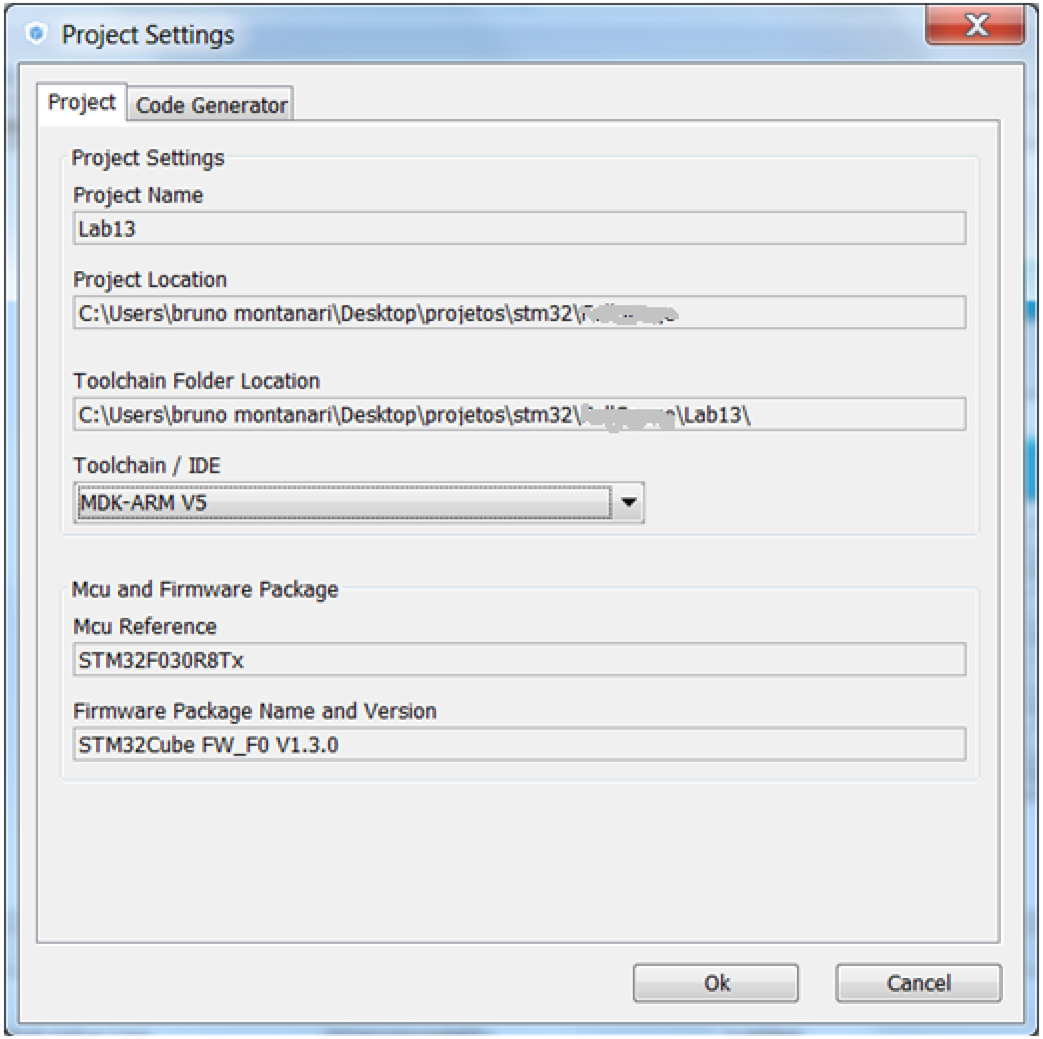
Search

□ Show only enabled interrupts

Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	□	0
System tick timer	☑	0
Flash global interrupt	□	0
RCC global interrupt	□	0
ADC global interrupt	☑	1

4.1.2 Use ADC with interrupt

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Now we can Generate Code
  - Menu > Project > Generate Code



Project Settings

Project Settings

Project Name: Lab13

Project Location: C:\Users\bruno montanari\Desktop\projetos\stm32\[redacted]

Toolchain Folder Location: C:\Users\bruno montanari\Desktop\projetos\stm32\[redacted]\Lab13\

Toolchain / IDE: MDK-ARM V5

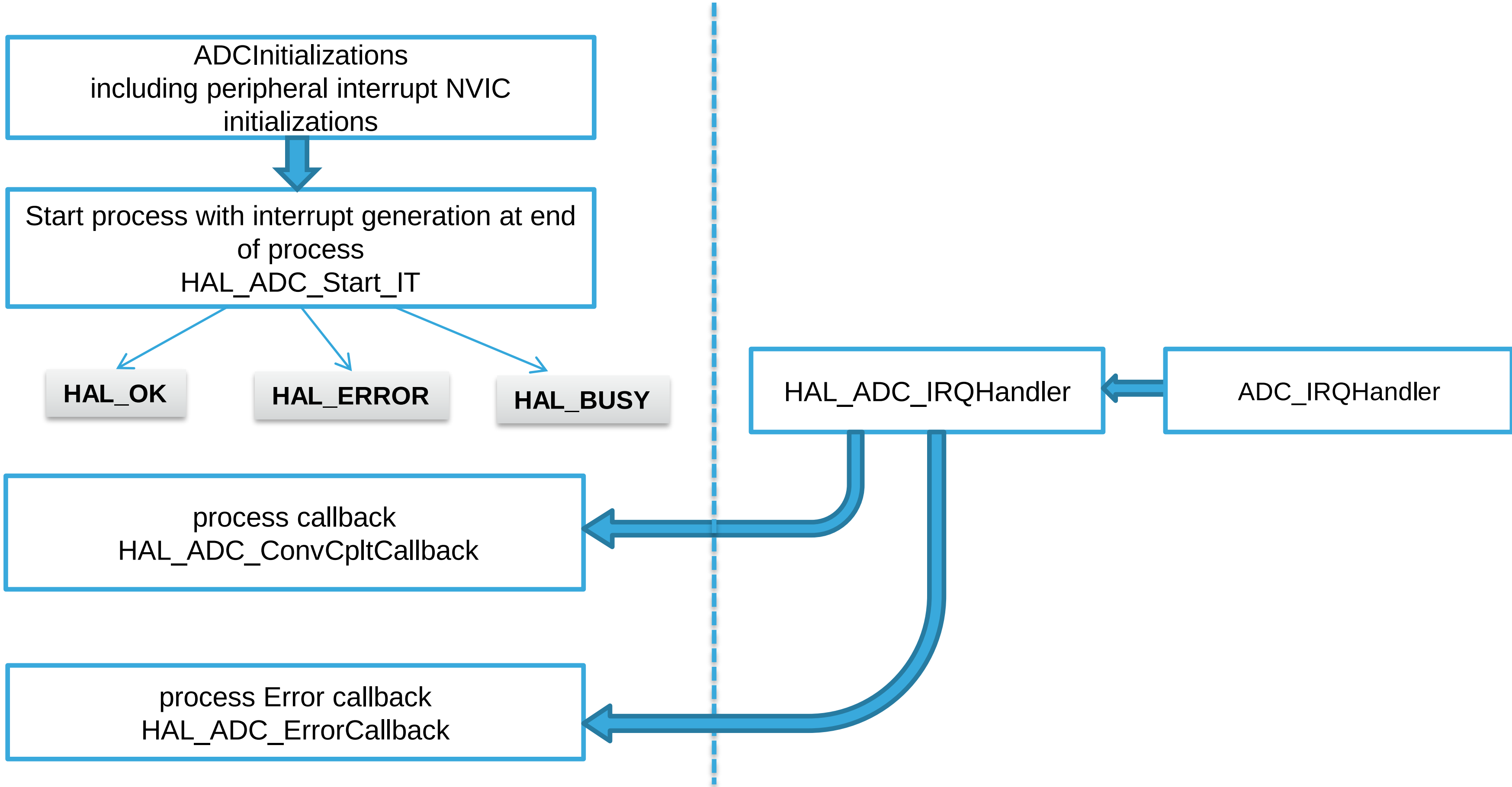
Mcu and Firmware Package

Mcu Reference: STM32F030R8Tx

Firmware Package Name and Version: STM32Cube FW_F0 V1.3.0

4.1.2 Use ADC with interrupt

HAL Library ADC with IT flow



ADCInitializations
including peripheral interrupt NVIC initializations

Start process with interrupt generation at end of process
HAL_ADC_Start_IT

HAL_OK

HAL_ERROR

HAL_BUSY

HAL_ADC_IRQHandler

ADC_IRQHandler

process callback
HAL_ADC_ConvCpltCallback

process Error callback
HAL_ADC_ErrorCallback

4.1.2 Use ADC with interrupt

- Open the project in our IDE
  - The functions we want to put into main.c
  - Between */* USER CODE BEGIN 2 */* and */* USER CODE END 2 */* tags
  - and */* USER CODE BEGIN 4 */* and */* USER CODE END 4 */* tags
- For DAC start use function
  - HAL_ADC_Start_IT(ADC_HandleTypeDef* hadc, uint32_t Channel)
  - HAL_ADC_GetValue(ADC_HandleTypeDef* hadc)
- ADC complete callback function
  - HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)

4.1.2 Use ADC with interrupt

- Solution
  - Variables

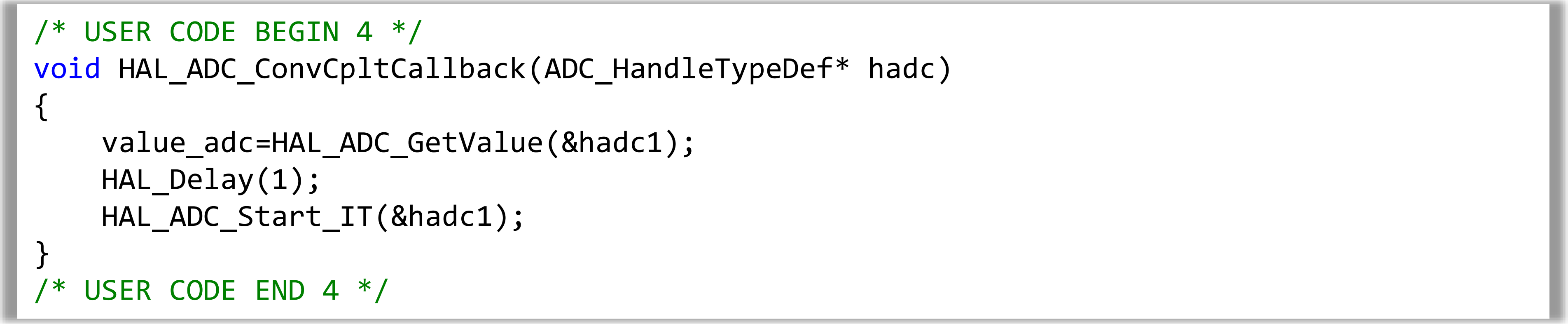
```
/* USER CODE BEGIN PV */
uint32_t value_adc;
/* USER CODE END PV */
```

  - Start ADC

```
 /* USER CODE BEGIN 2 */
HAL_ADC_Start_IT(&hadc1);
  /* USER CODE END 2 */
```

4.1.2 Use ADC with interrupt

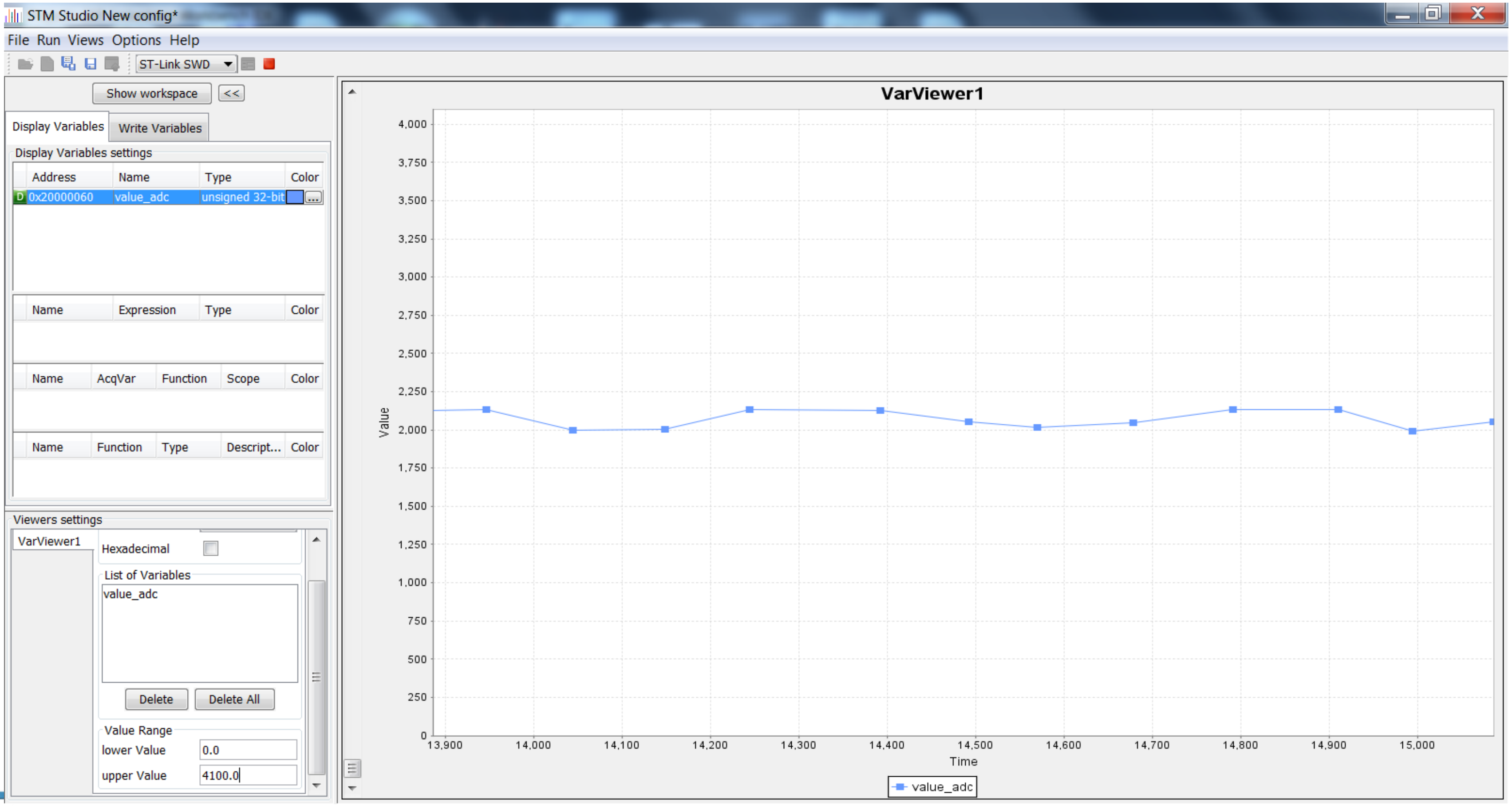
- Solution
  - ADC complete callback routine



```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    value_adc=HAL_ADC_GetValue(&hadc1);
    HAL_Delay(1);
    HAL_ADC_Start_IT(&hadc1);
}
/* USER CODE END 4 */
```

4.1.2 Use ADC with interrupt

- STM studio settings
  - Check functionality again with STMstudio



Address	Name	Type	Color
0x20000060	value_adc	unsigned 32-bit	🟦

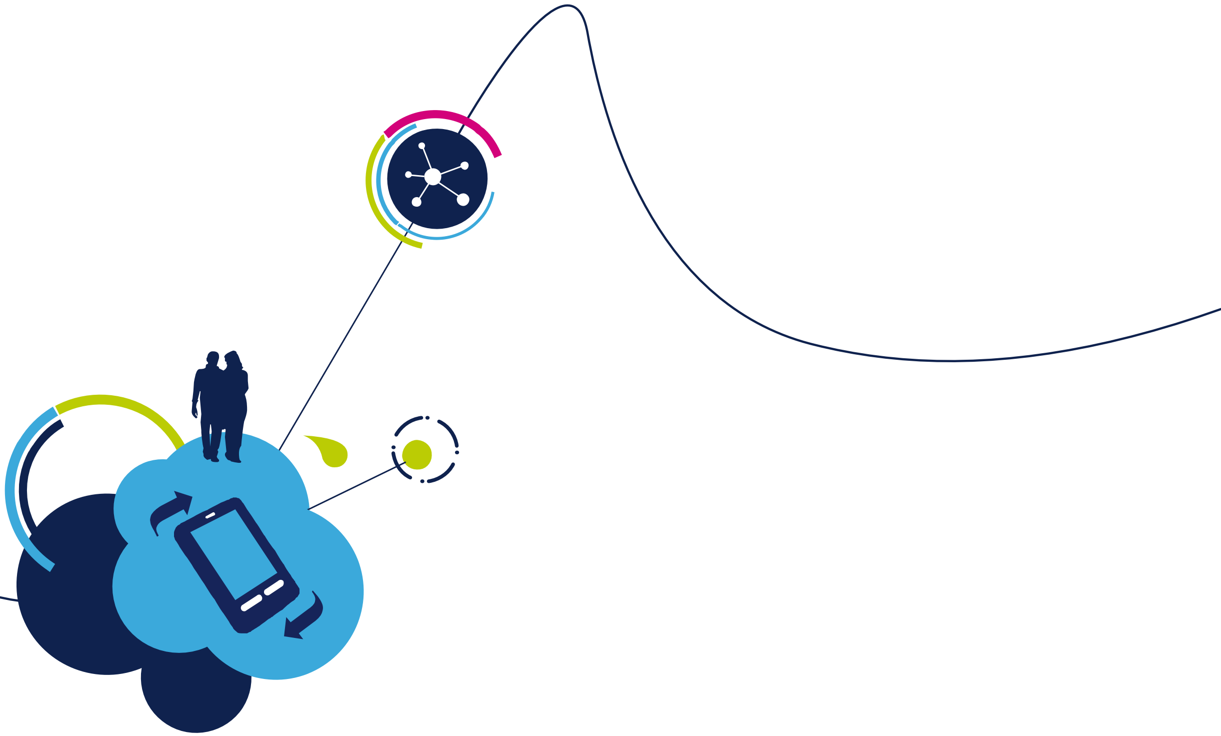
Hexadecimal □

List of Variables: value_adc

Value Range

lower Value: 0.0

upper Value: 4100.0



Appendix B Documents

life.augmented

B

CubeMX documentation

- CubeMX user manual UM1718
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00104712.pdf
- CubeMX release note RN0094
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00104712.pdf
- CubeMX technical note TN0072
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/technical_note/CD00214439.pdf

B

STM32F429i-Discovery documentation

- STM32F429i-Discovery page
  - http://www.st.com/web/en/catalog/tools/FM116/SC959/SS1532/LN1848/PF259090?s_searchtype=keyword
- STM32F429i-Discovery user manual with discovery schematics
  - http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00093903.pdf