STM32CubeIDE basics

ADC lab: DMA + TIM configuration using HAL library

life.augmented

STM32 CubeIDE

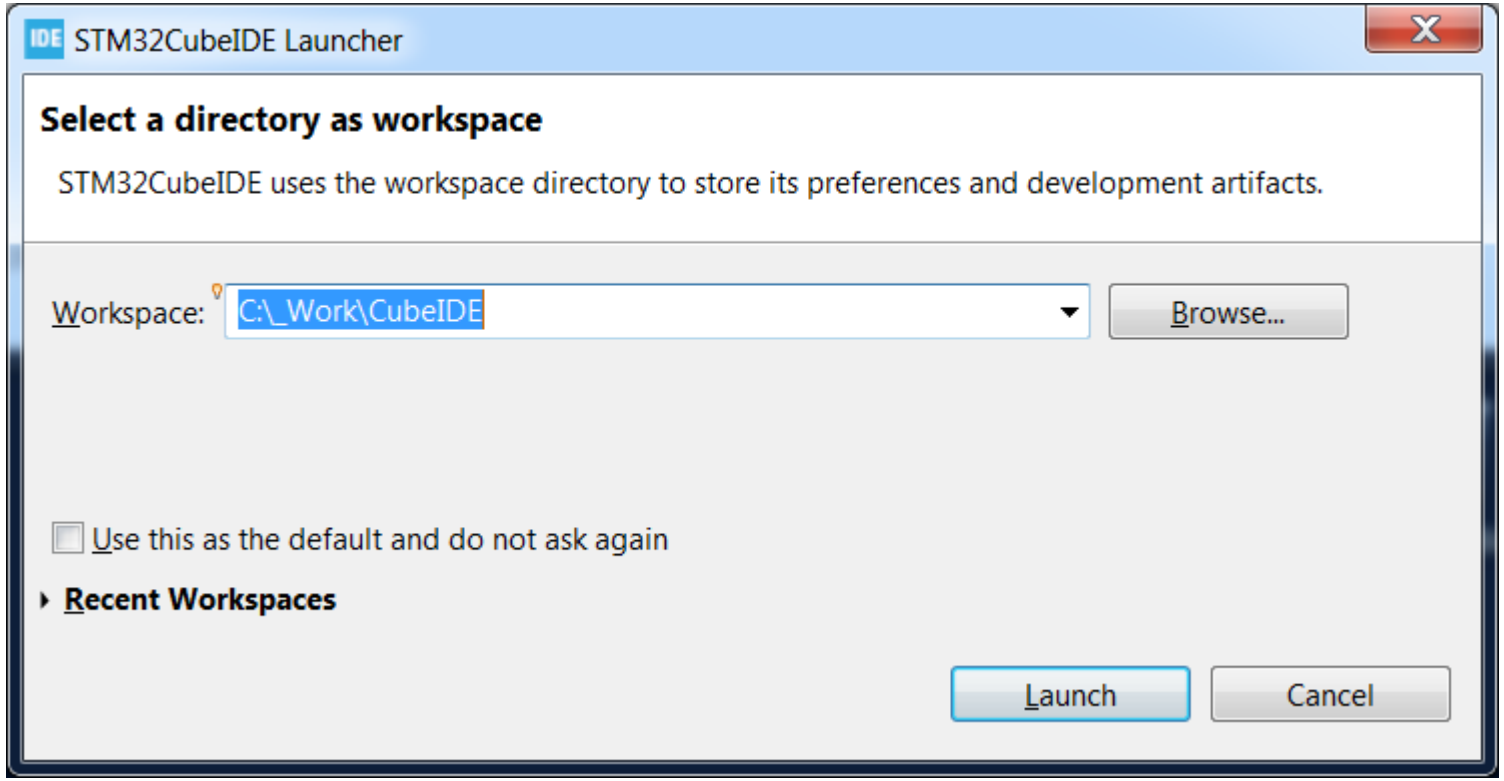
Lab: ADC+DMA+TIM configuration

Objective:

- Using ADC collect data from selected analog channel (i.e. internal temperature sensor),
- Each conversion should be triggered by TIM2 each 1 second
- Data should be transferred via DMA to internal buffer (8 results long)
- When buffer is full, TIM2 should be stopped and interrupt should be raised to allow ADC data postprocessing

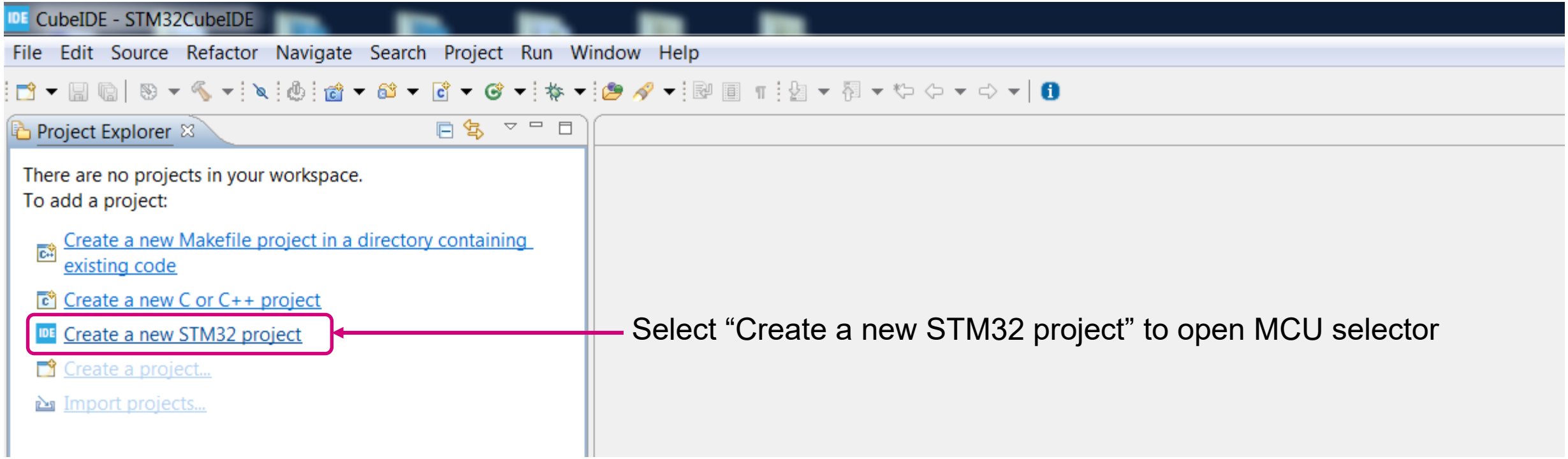
Start a new workspace

- Run STM32CubeIDE
- Select a folder to store a workspace



Create a new project

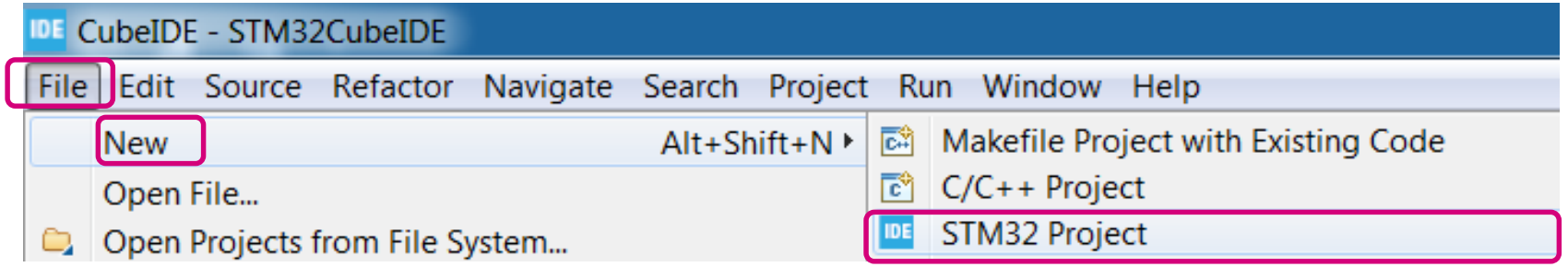
- Click on “Create a new STM32 project”



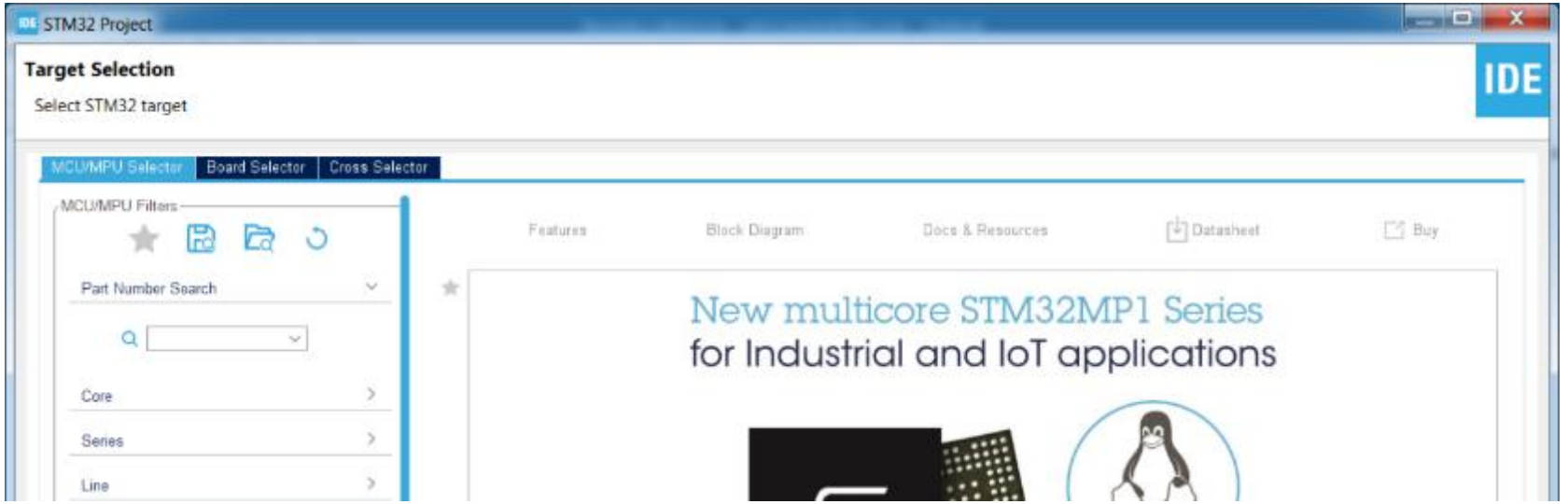
.. Click here to continue

Start a new project within workspace

- Go to File -> New -> STM32 Project

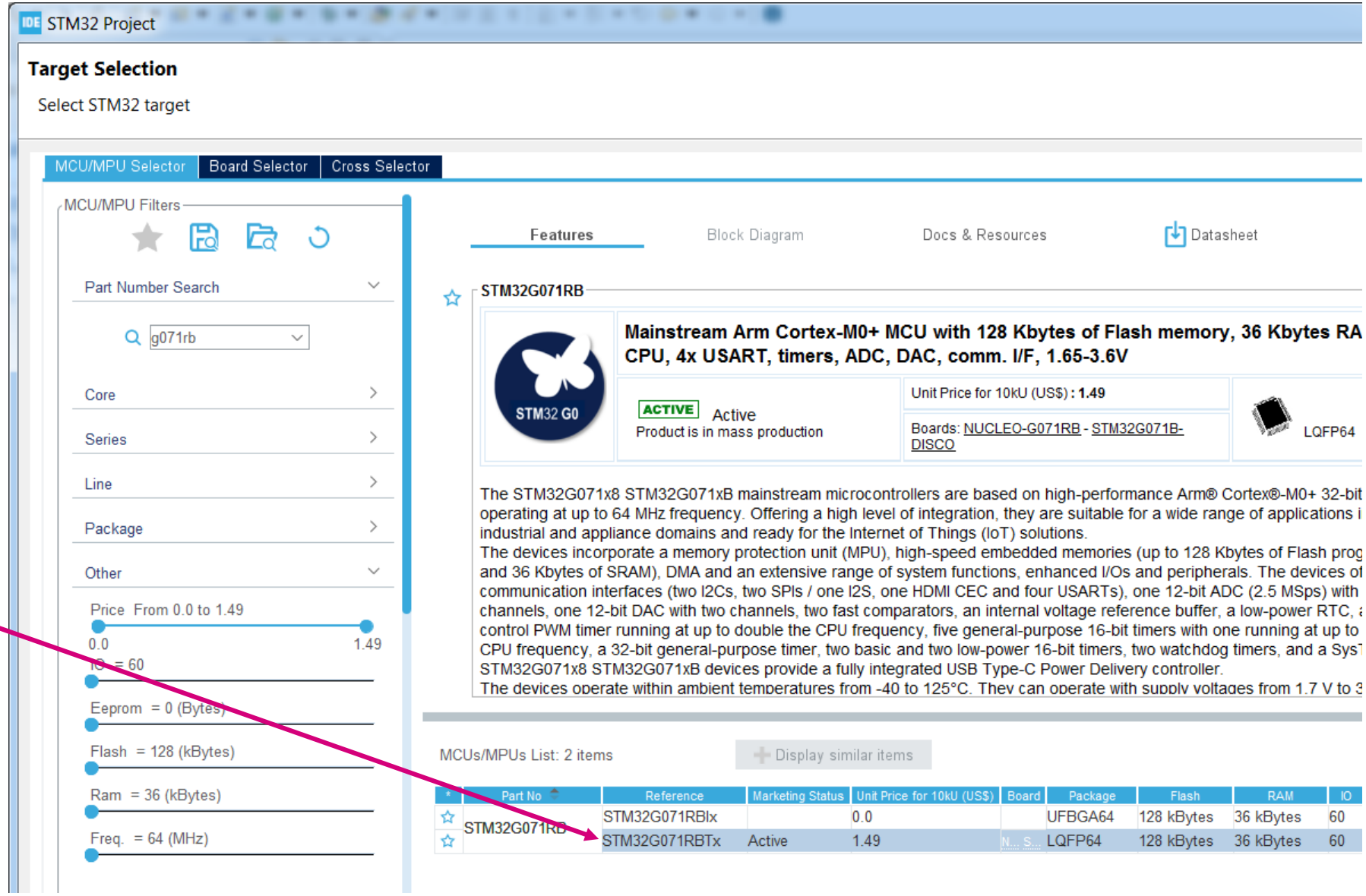


- As a result “target selection” window will be displayed



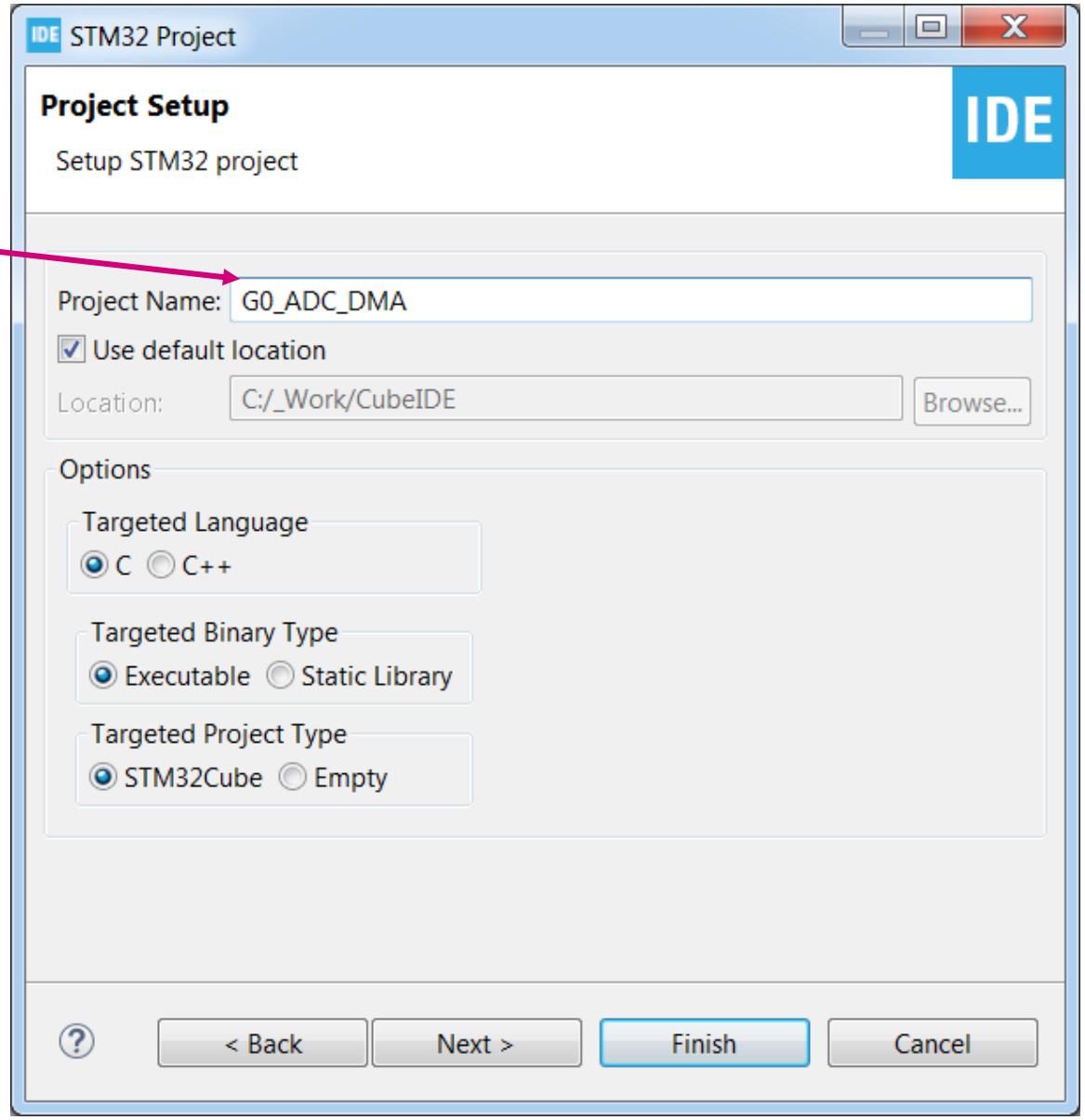
Select target MCU: STM32G071RBTx

- It is possible to view on main MCU features, download its documentation
- To start a new project we need to double click on the part number



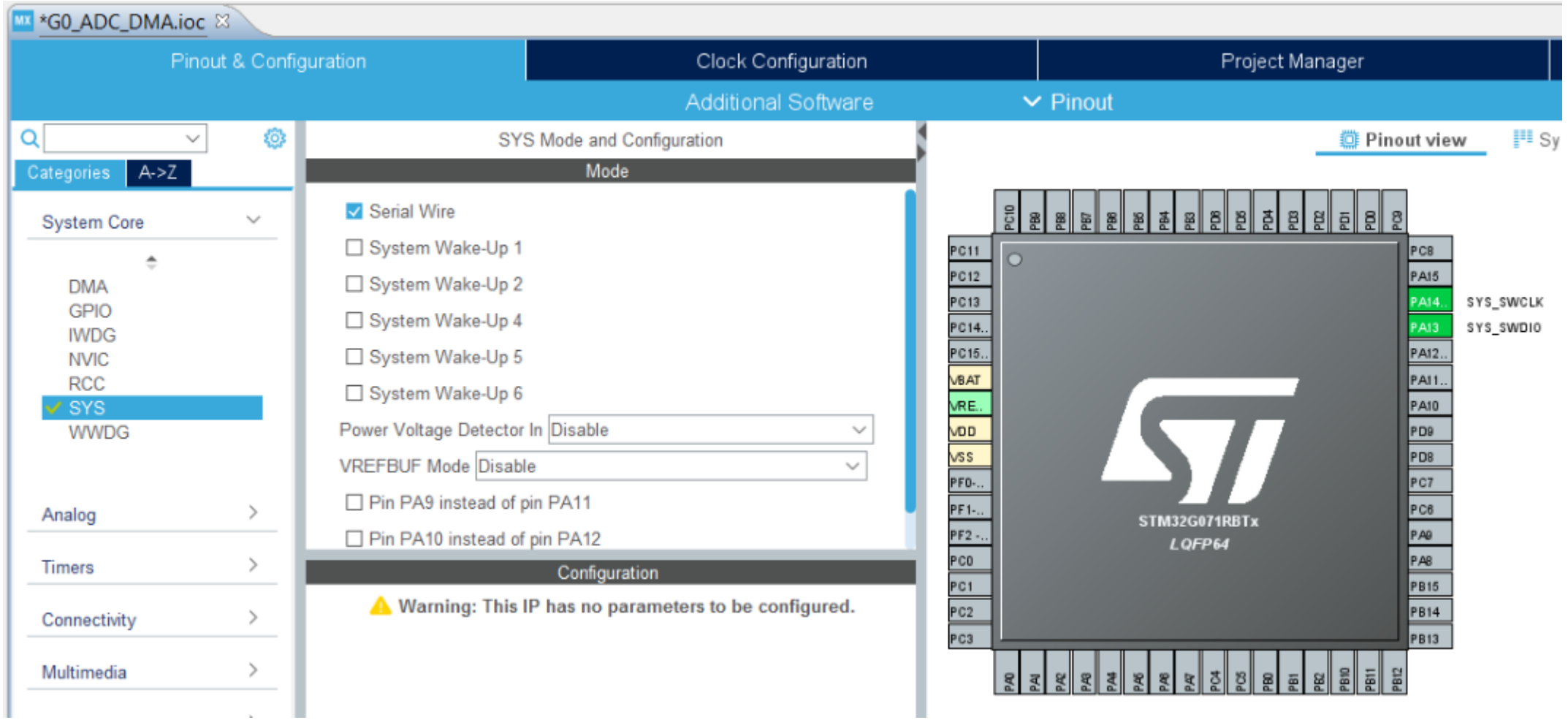
Enter project name

- Specify project name, optionally its location (if different from workspace one)
- Additionally we can specify target language (C or C++), binary type (executable or static library) and project type (generated by STM32CubeMX or an empty one)



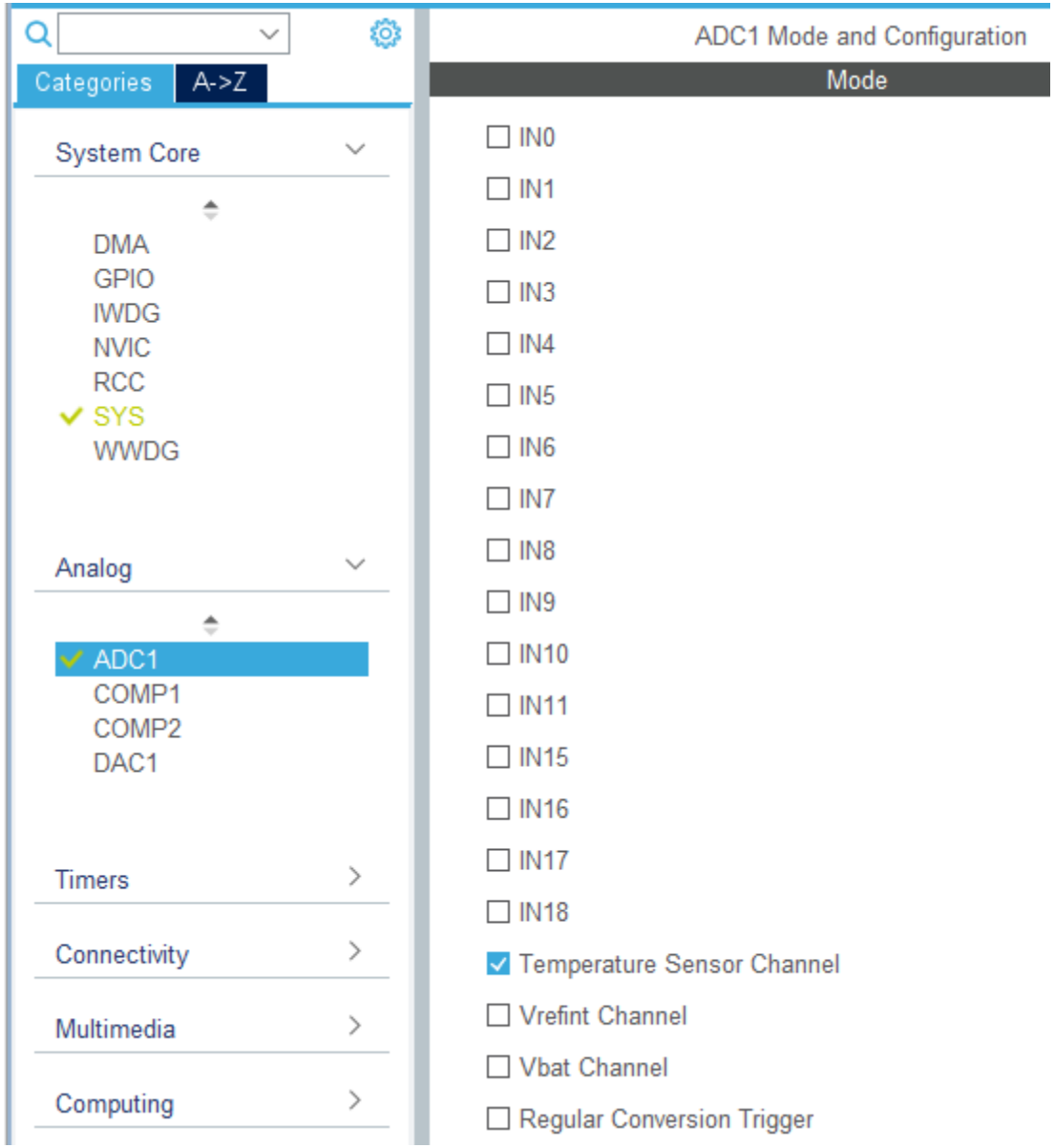
Enabling Serial Wire debug interface

- Select “Serial Wire” from System Core -> SYS peripheral group
- As a result PA13 and PA14 will be assigned to SWD interface



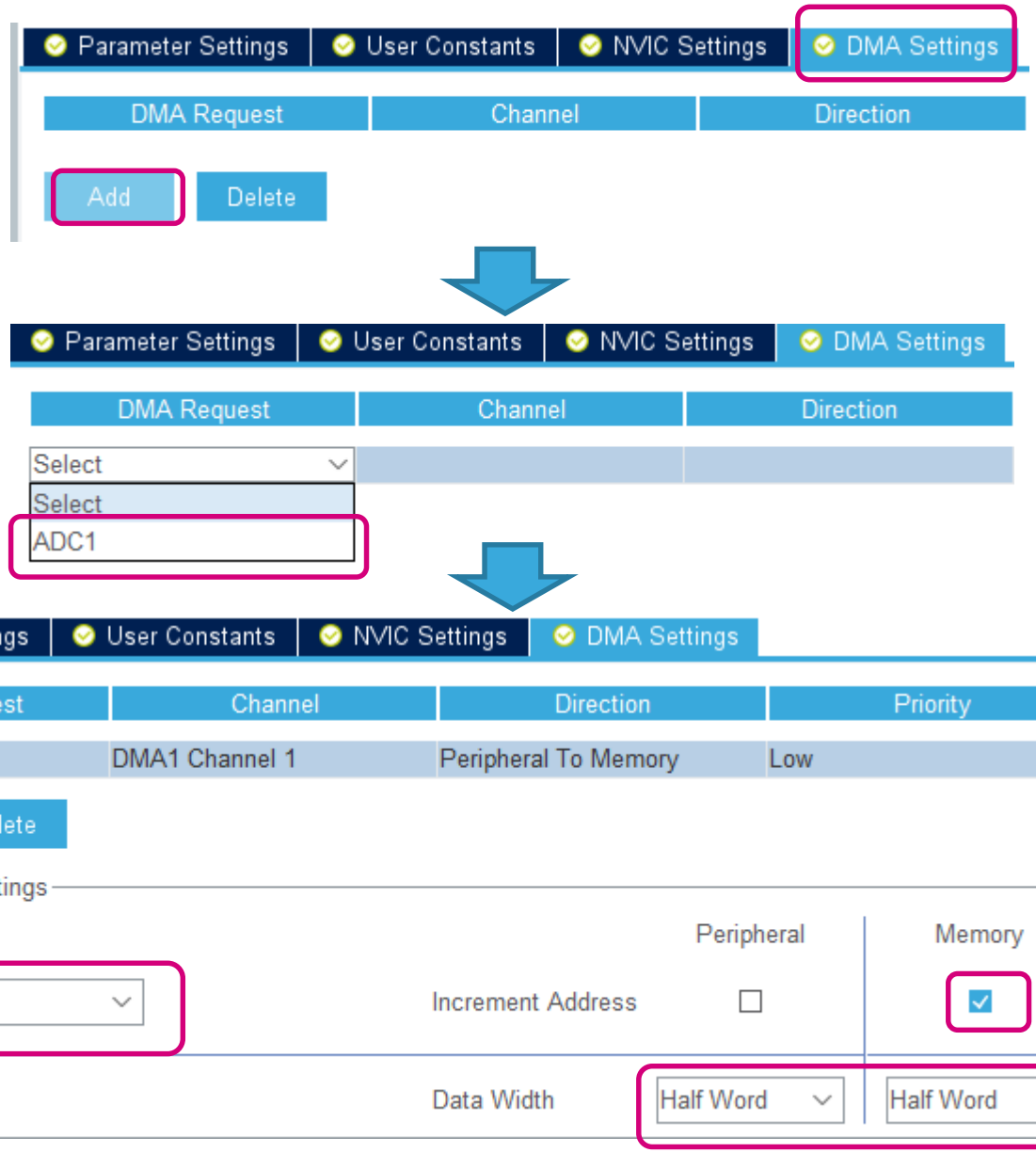
ADC channel selection

- Select ADC1 within Analog peripherals group and enable “Temperature Sensor Channel” within Mode window



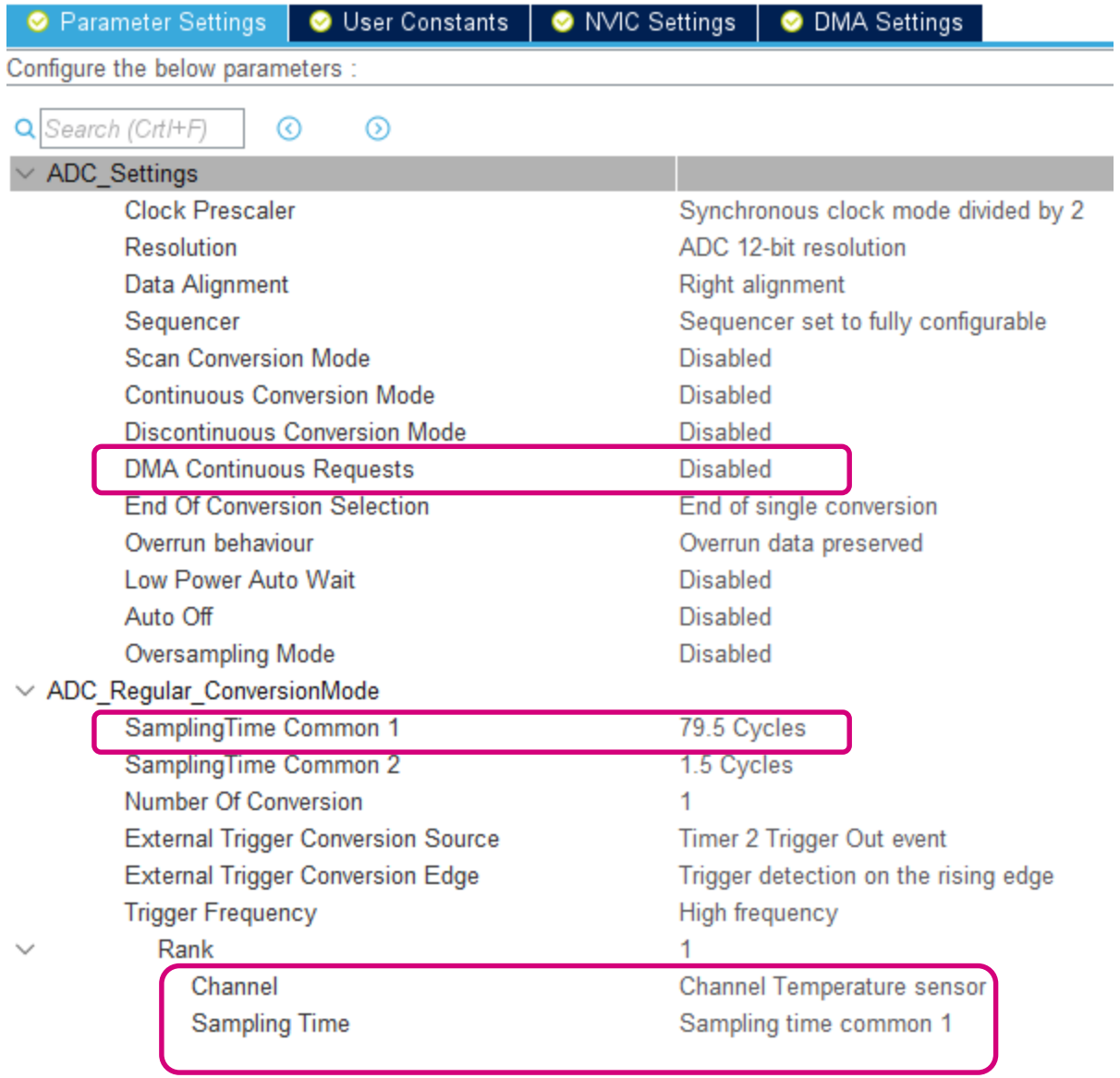
ADC+DMA configuration

- In ADC1 configuration, DMA Settings tab, please Add a new DMA request
- Select ADC1 from the list
- Set its configuration to:
  - Normal Mode
  - Increment on Memory side (data buffer)
  - Half Word data width on both sides (as we will operate on 12bit data)



ADC configuration

- In ADC1 configuration, Parameter Settings tab configure:
  - DMA Continuous Requests: Disabled (as we are not using Circular DMA mode)
  - SamplingTime Common1 to 79.5 cycles
  - External Trigger Source: Timer2 Trigger Out event (this event is programmed within Timer2 configuration)
  - External Trigger Edge: rising edge
  - Sampling time common 1 as sampling time for temperature sensor channel

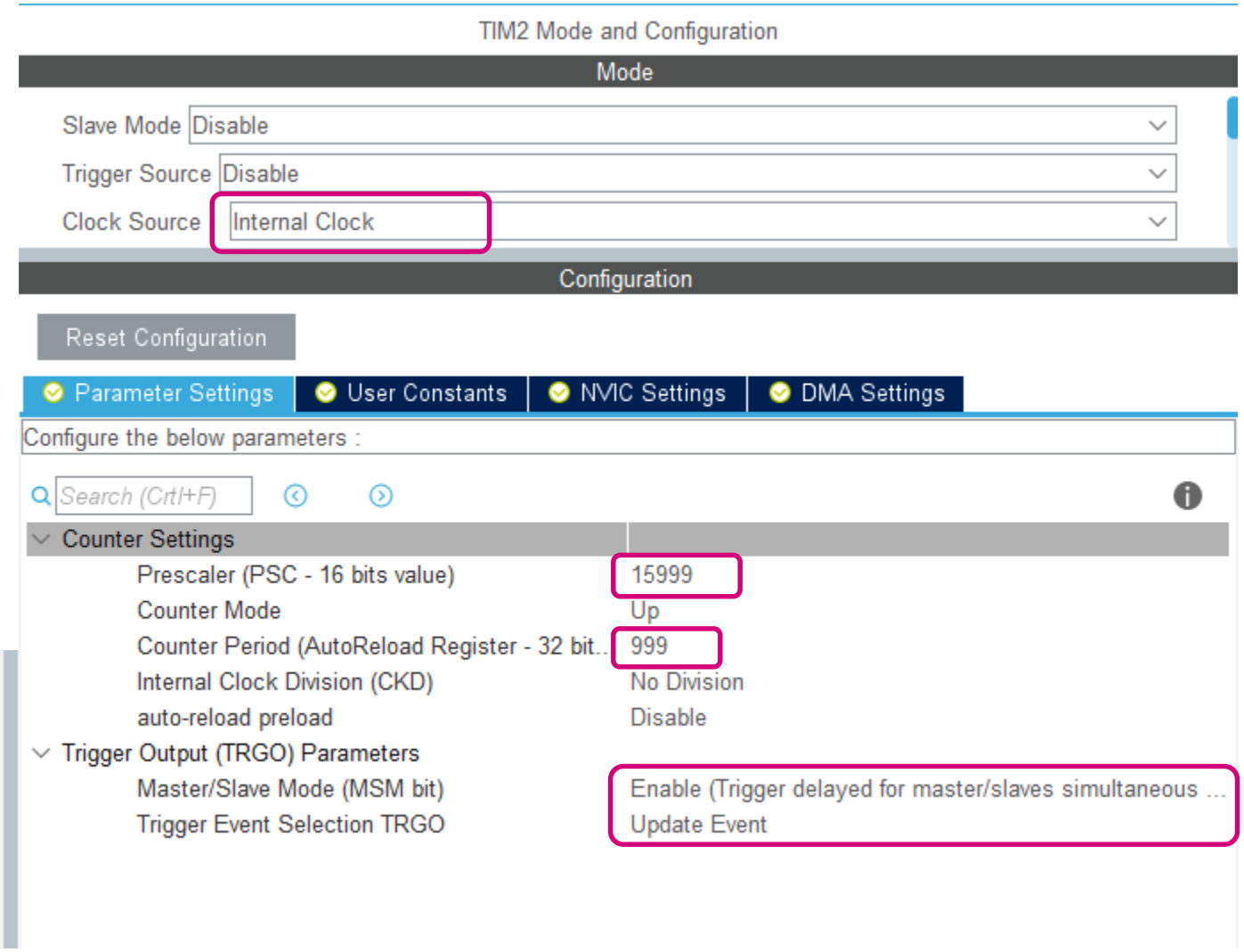


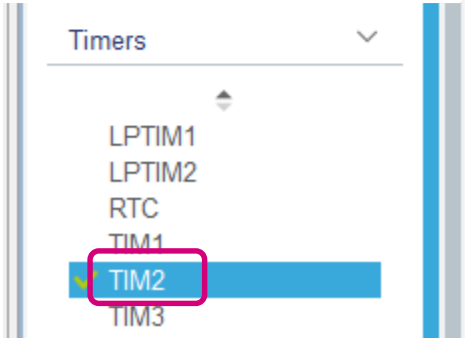
Timer Parameters Calculation

- TIM2 will be used to trigger ADC1 conversions each 1second
- System is working on default clock settings (HSI 16MHz, no PLL) and we use ADC clock as system clock / 2 => 8MHz
- We have Timer2 input clock 16 MHz (default system clock settings using HSI without PLL)
- We can set prescaler (PSC) to 15999 (actual divide is PSC+1) to have 1kHz on Timer2 counter input
- Then we need to set Timer2 period to 999 (as we are counting from 0) to have overflow (update event) each 1 second

TIM2 configuration

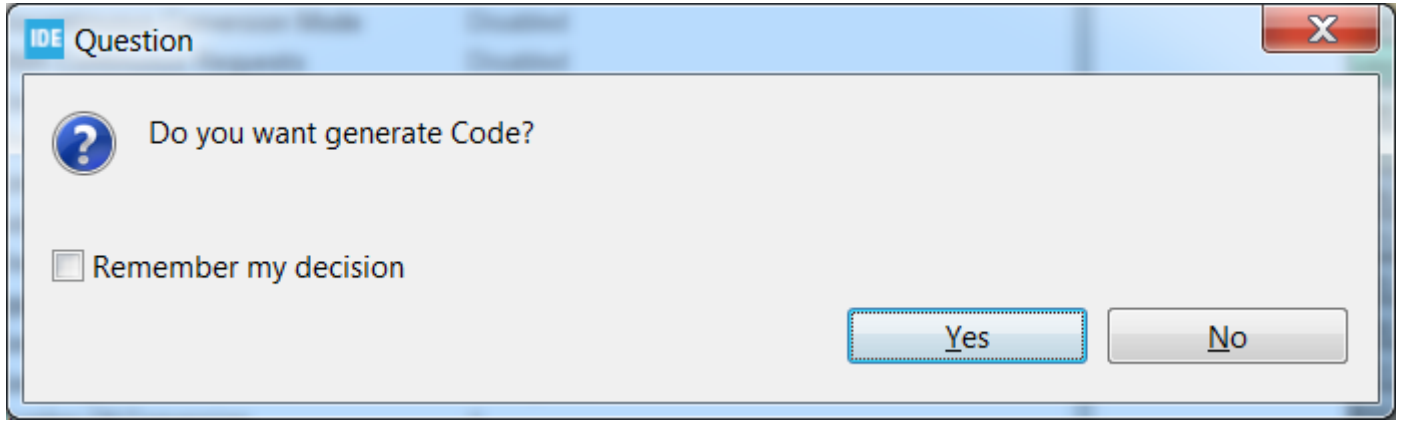
- Select Timers->TIM2
- Set clock source to Internal Clock
- Configure:
  - Prescaler to 15999
  - Counter period to 999
  - MSM bit: Enable
  - TRGO selection: Update Event





Generate the code

- Save the design settings to trigger code generation



Coding time – defines and variables

main.c file

- Define data buffer size and ADC reference voltage (in mV) used for temperature calculation

```
/* USER CODE BEGIN PD */
#define ADC_BUF_SIZE       8
#define __VREFANALOG_VOLTAGE__    3300//ADC reference voltage in mV
```

- Define data buffer for temperature measurements

```
/* USER CODE BEGIN PV */
uint16_t ADC_buffer[8];
```

Coding time – ADC calibration and start

main.c file

- Perform ADC calibration, start ADC and start TIM2 which would trigger ADC conversion each 1second

```
/* USER CODE BEGIN 2 */
  if(HAL_ADCEx_Calibration_Start(&hadc1) != HAL_OK)
       Error_Handler();

  if(HAL_ADC_Start_DMA(&hadc1 , (uint32_t *)ADC_buffer, ADC_BUF_SIZE) != HAL_OK)
       Error_Handler();

  if(HAL_TIM_Base_Start(&htim2) != HAL_OK)
       Error_Handler();
```

- In all cases we are monitoring proper function execution by checking return value

Coding time – ADC data collection and ADC stop

main.c file

- As we have configured DMA in Normal mode, it would be stopped once the buffer will be filled with the data
- Additionally a DMA transfer complete interrupt will be triggered which could be used to stop ADC or simply TIM2 which is triggering ADC.
- DMA transfer complete interrupt is linked with `HAL_ADC_ConvCpltCallback()`, so we can use the same callback in case we are using ADC in IRQ or in DMA mode

```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
  if(HAL_ADC_Stop_DMA(&hadc1) != HAL_OK)
       Error_Handler();
}
/* USER CODE END 4 */
```

… Let's check it

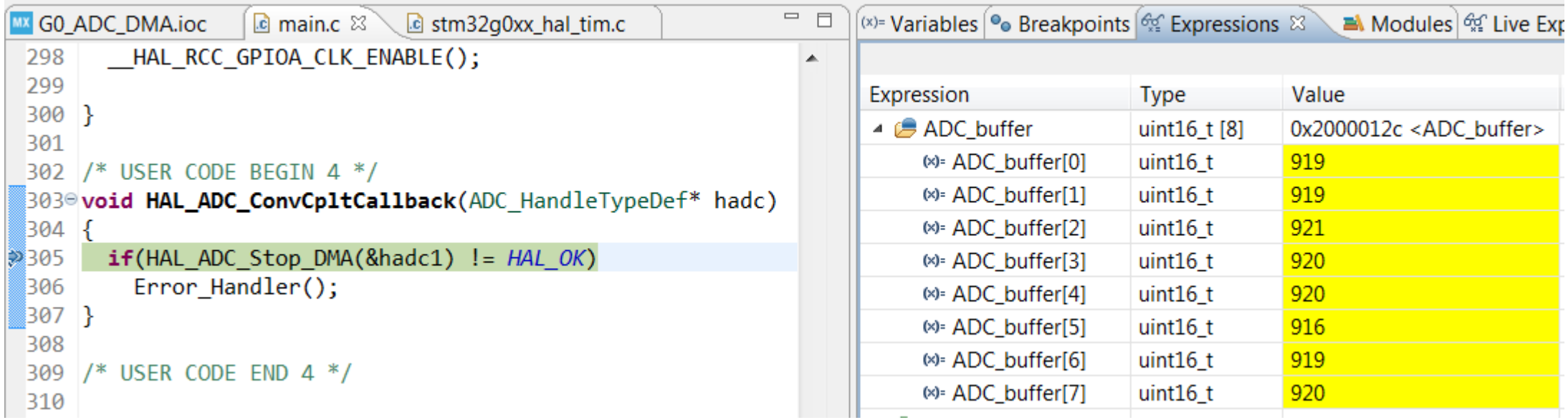
- After all code processing we can build the project, start debug session and run the application

- We can monitor ADC_buffer (while stopped on breakpoint or paused the debug session) by:
  - Clicking on its name (separate window with more details will be displayed)

  - Adding it to watch (left mouse button click on its name and select Add Watch Expression and then fill then name of the variable into "Add Watch Expression" window

Monitoring of variables in debug mode

- We can set a breakpoint within HAL_ADC_ConvCpltCallback() which would be called once complete buffer is filled with data.



- Now we need to recalculate measured values into deg C values. We can use macro `__LL_ADC_CALC_TEMPERATURE()`

STM32 CubeIDE

Thank you

/STM32

@ST_World

community.st.com

life.augmented

www.st.com/stm32g0, www.st.com/stm32cubeide