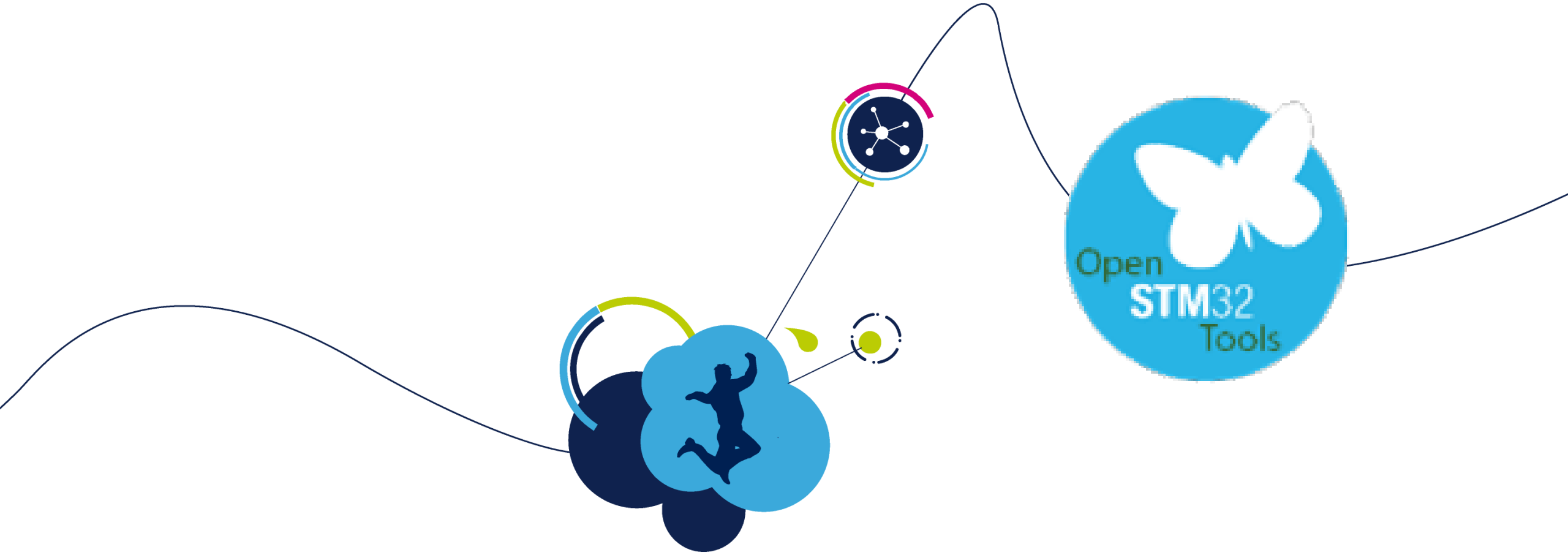
STM32 Ecosystem Workshop

T.O.M.A.S Team

life.augmented

- After successful code generation by STM32CubeMX this is the right time to import it into SW4STM32 toolchain for further processing



Handling the project in SW4STM32

life.augmented

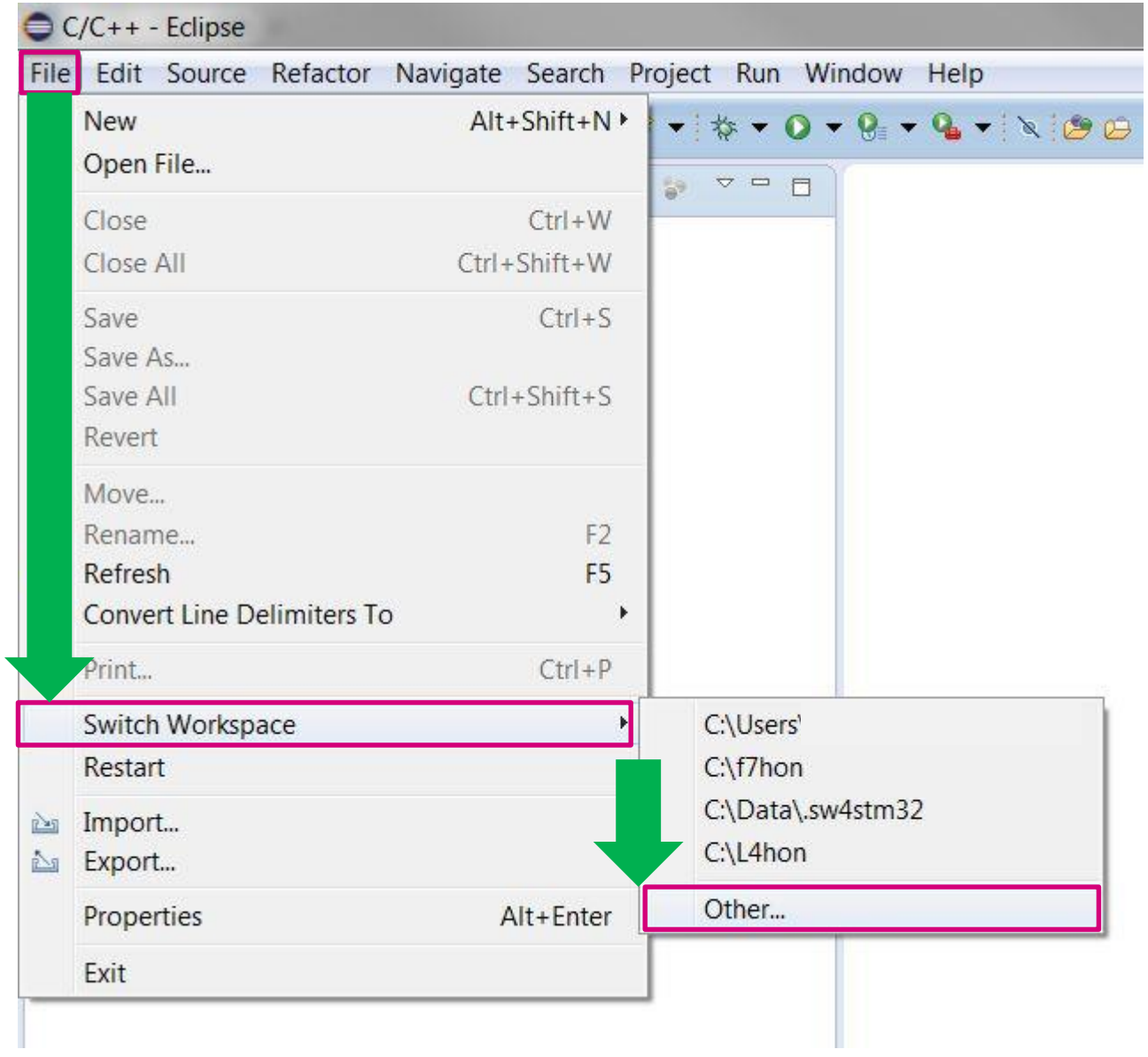
Our goals for this session

- Handling the projects generated by STM32CubeMX in SW4STM32
  - Import project generated by STM32CubeMX
  - Tune sources to run selected peripherals in desired algorithm
  - Build project
  - Configure debug session
  - Run debug session
  - Debug perspective
  - Watching the variables and registers content
  - Handling errors

Create a new workspace

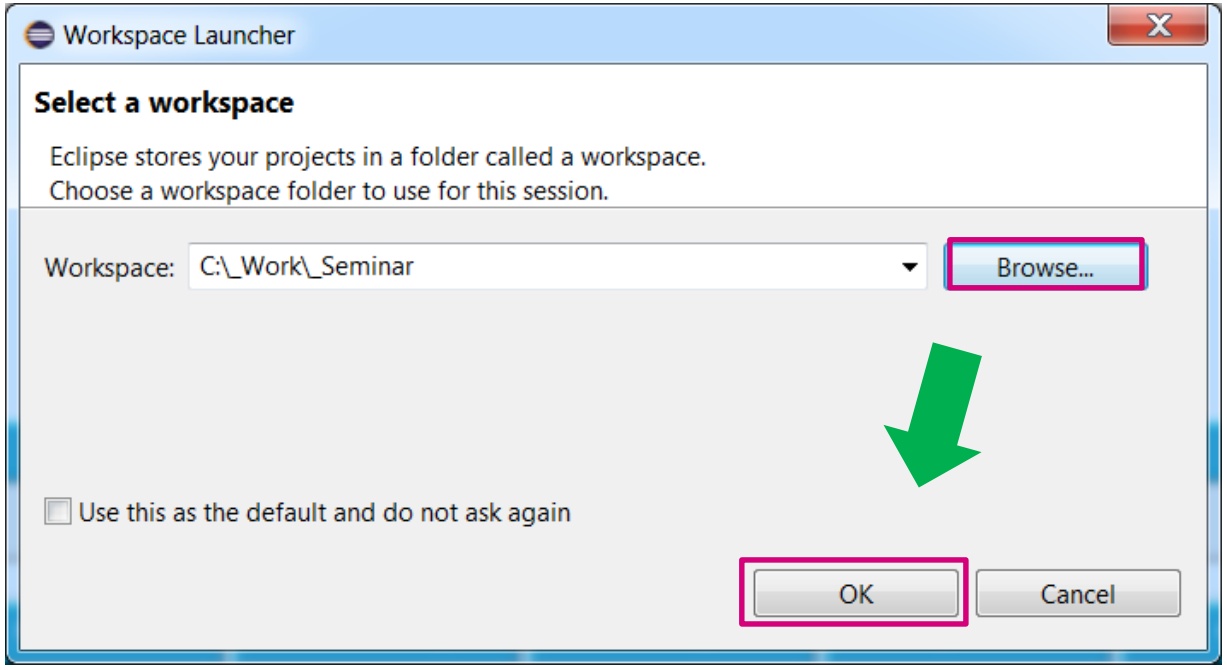
SW4STM32

- Start Workspace launcher if not done automatically by Eclipse.

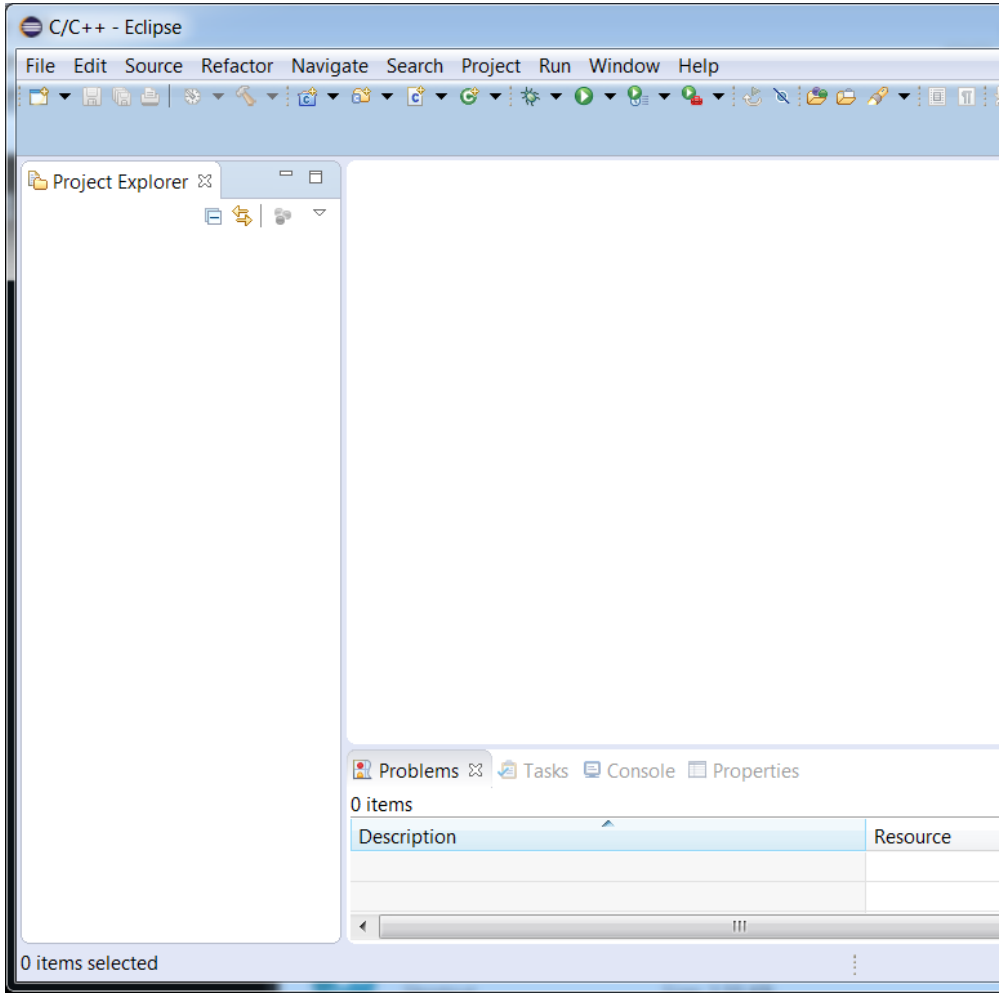


Create a new workspace
SW4STM32

- Create new workspace in the desired location – but not in the same folder where the project which will be imported is located (it must be one level above the project)

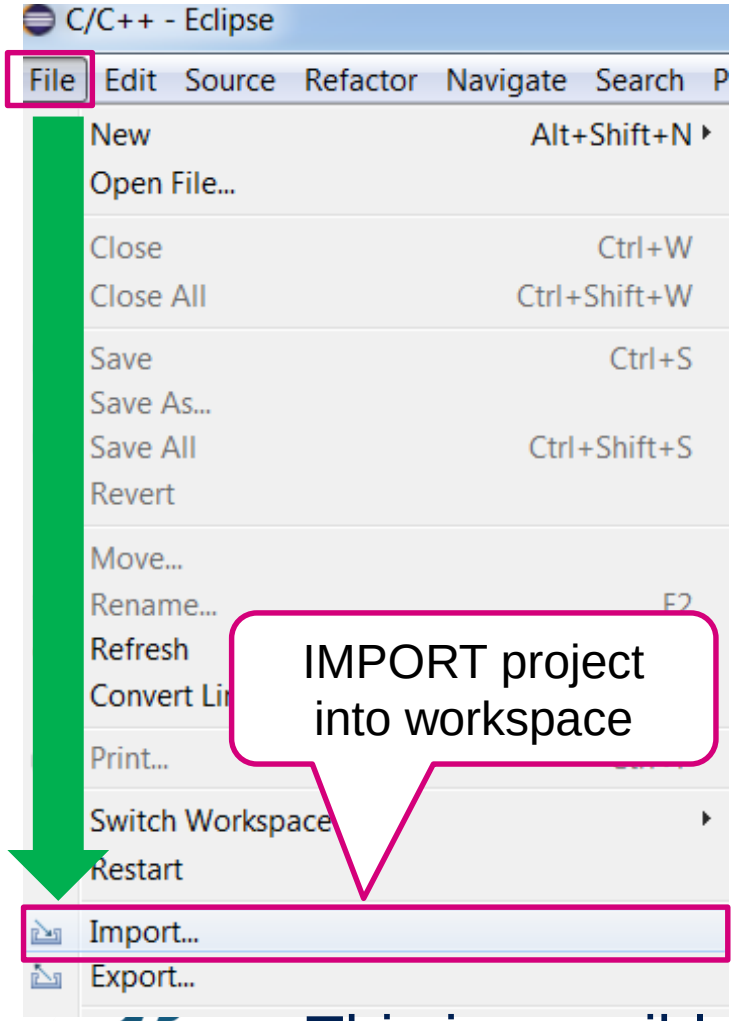


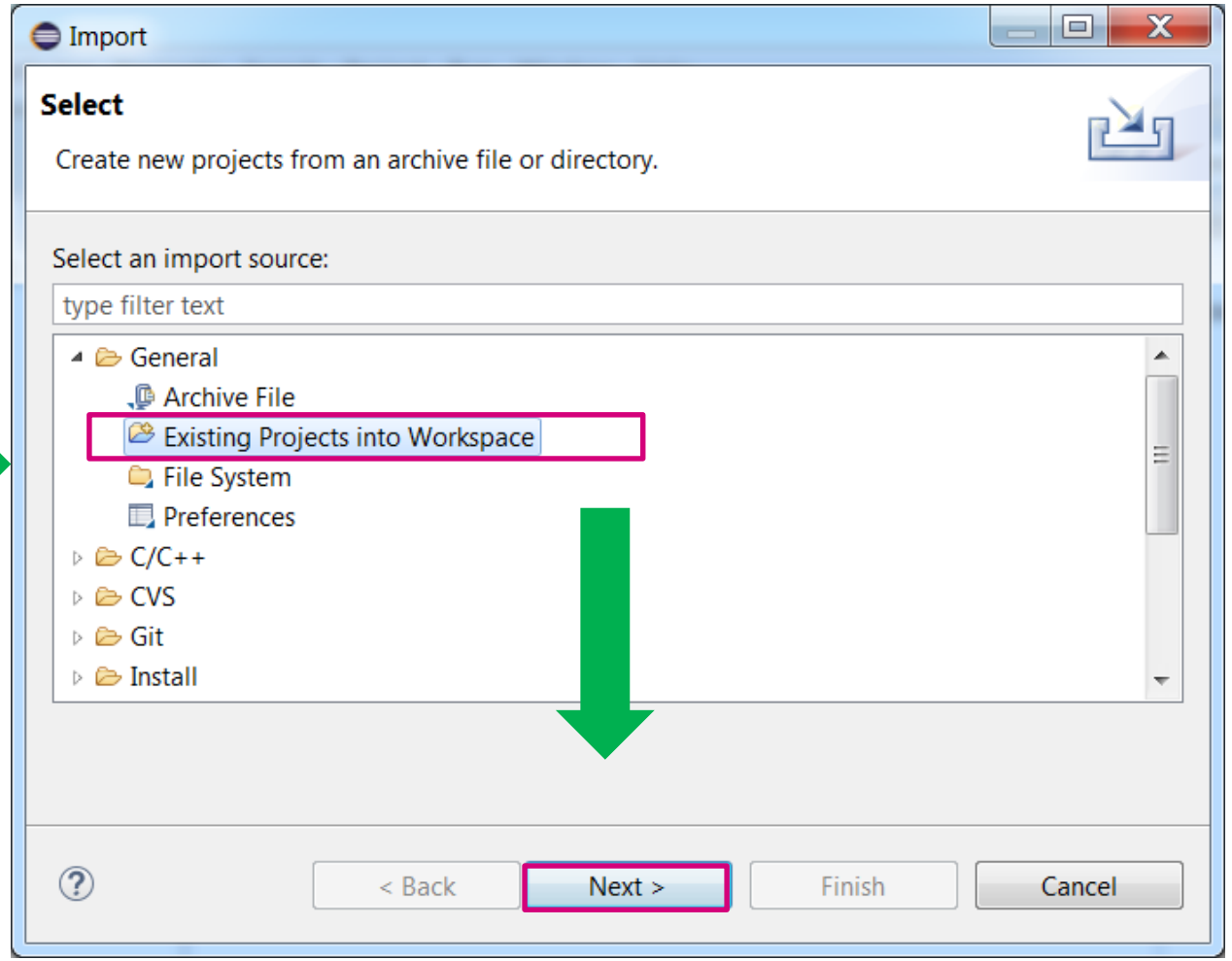
- An empty workspace will be generated



Import the project into the workspace 1/3

Import “L4_Blinky” project into empty workspace following below steps:

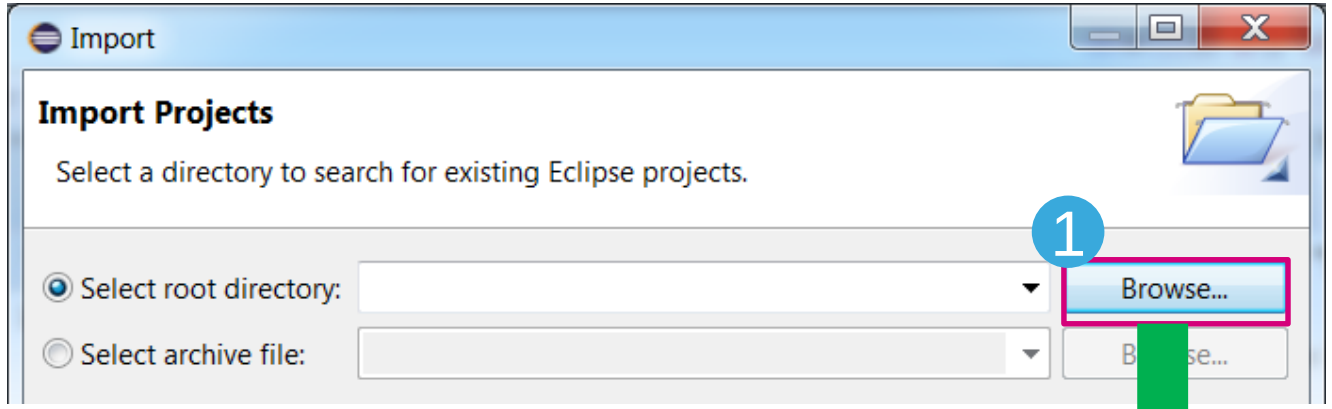




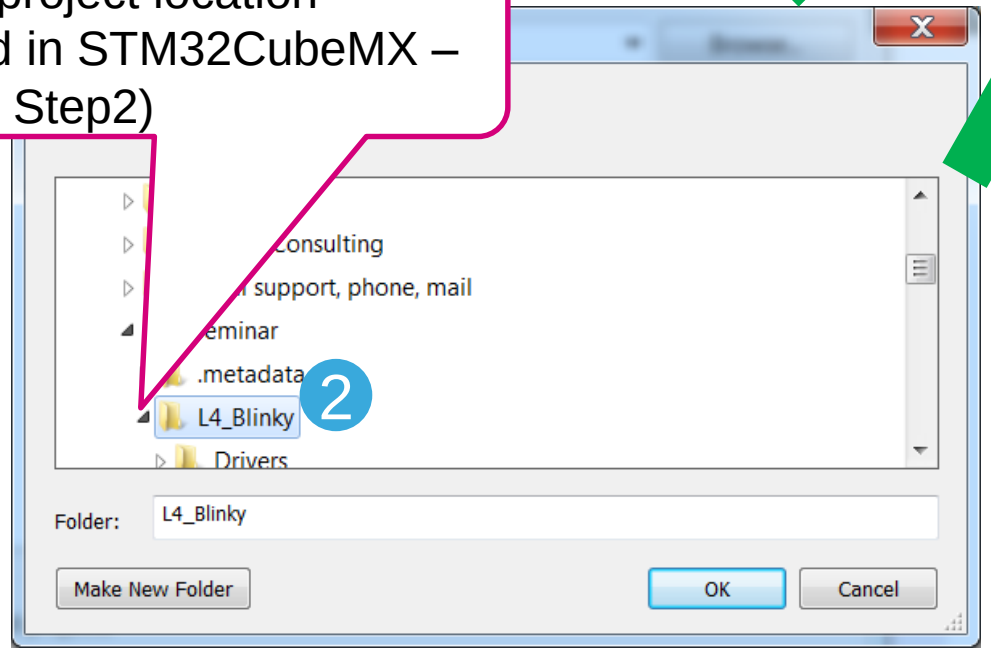
This is possible to import multiple projects into a single workspace

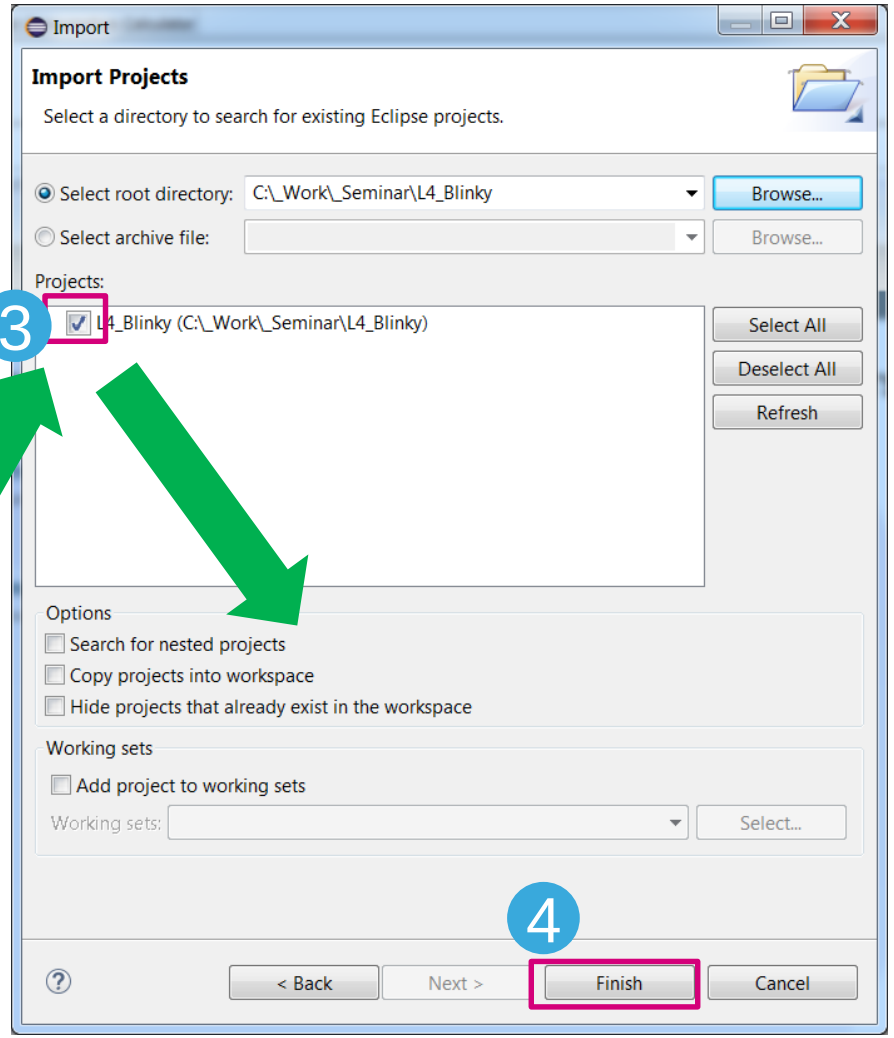
Import the project into the workspace 2/3

In this example L4_Blinky project will be processed.



Select project location (as configured in STM32CubeMX – Step2)



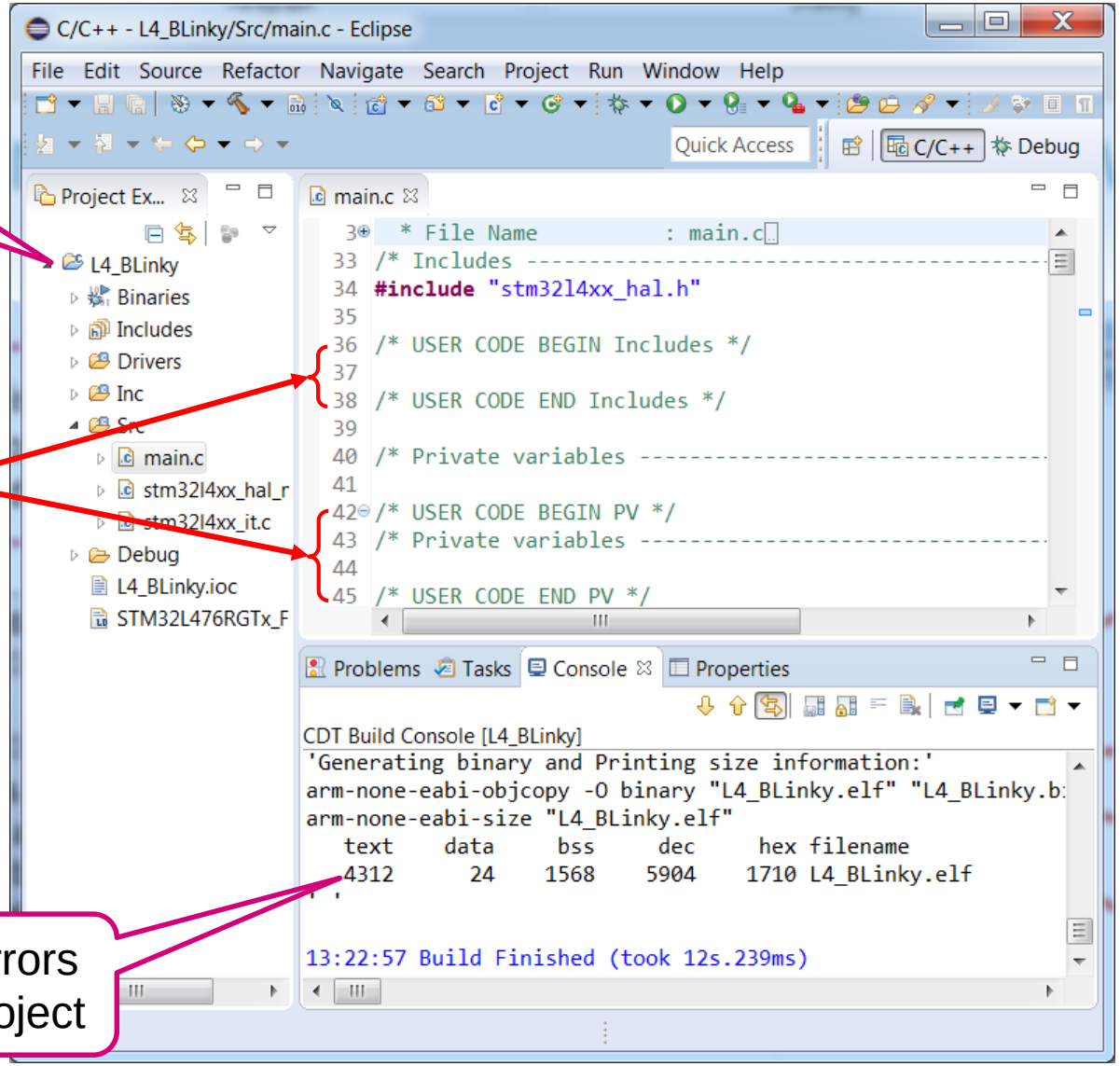


life.augmented

Import the project into the workspace 3/3

Once project is included into the workspace, its folder structure becomes visible in Project Explorer

Places dedicated for user code are marked by
`/* USER CODE … BEGIN*/`
and
`/* USER CODE … END*/`
comment lines.
These places are protected from being removed during code re-generation by STM32CubeMX.
This is **possible** to define another user code places in **.c** source files but **not possible** in **.h** header files.



Warnings and errors after build the project

- STM32CubeMX generated project is only a skeleton which should be filled with some code from our side
- To make green LED (connected to properly configured PA5 pin) we should continuously invoke GPIO toggle function with the proper delay to make blink visible

Modifying the code

blinking green LED (PA5)

Tasks (within while(1) loop in main.c):

1. Add GPIO pin toggle function for PA5 pin. Which function we can use here?

   ?

2. Add 500ms delay between each change of the GPIO pin state. Which function we can use here?

   ?

Hints:

- All HAL function begins with **HAL_PPP_** prefix (PPP – short name of the peripheral, i.e. GPIO)
- Please try to use Content Assistant (**Ctrl+SPACE**) in Eclipse

Modifying the code

blinking green LED (PA5) - solution

Solutions (within while(1) loop in main.c):

1. Add GPIO pin toggle function for PA5 pin. Which function we can use here?

```
HAL_GPIO_TogglePin();
```

2. Add 500ms delay between each change of the GPIO pin state. Which function we can use here?

```
HAL_Delay();
```

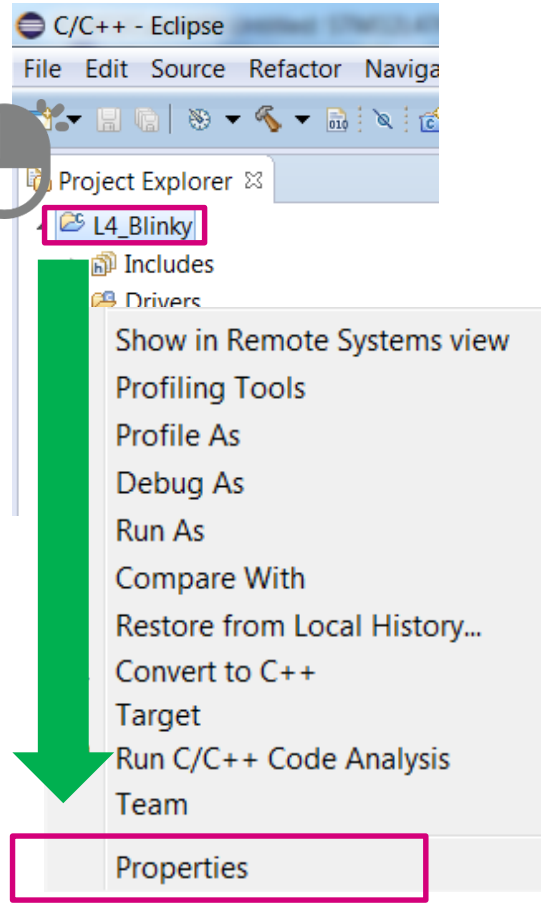
```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
  HAL_Delay(500);
}
/* USER CODE END 3 */
```

Useful project settings in SW4STM32

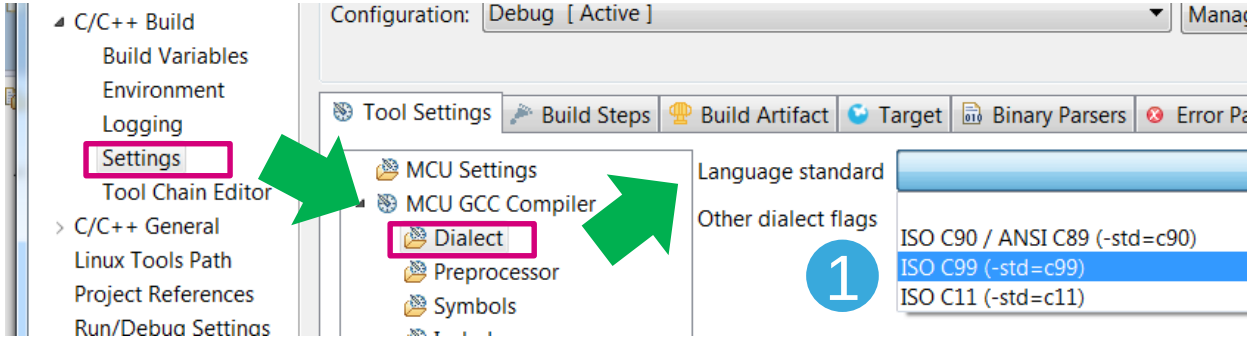
configuring C dialect and parallel build

Project->Properties



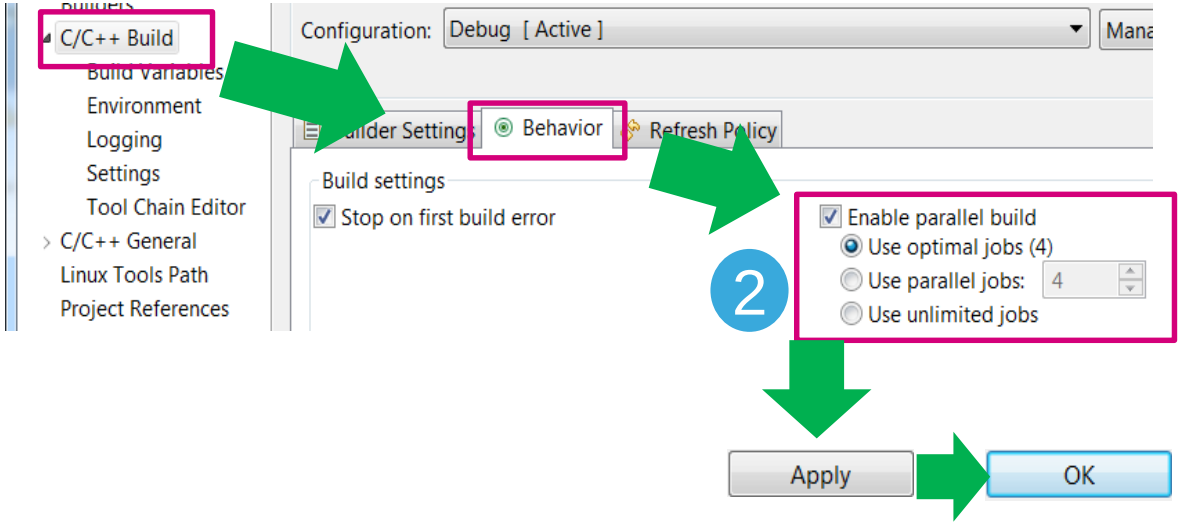
C/C++ Build->Settings->Tools Settings tab->MCU GCC Compiler->Dialect

1. Configure C standard to **C99** to avoid possible compilation errors



C/C++ Build->Behavior tab

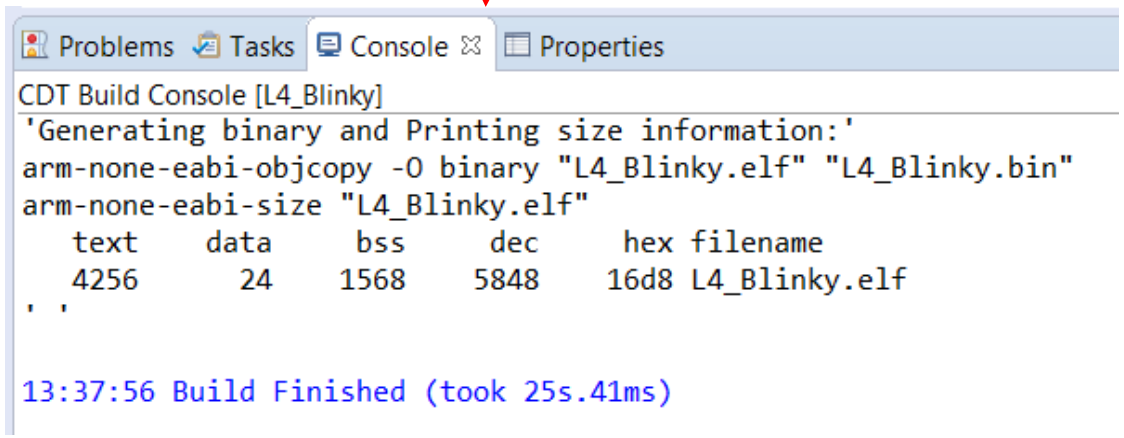
2. Check **Enable parallel build** to make use of your machine potential and to shorten compilation time

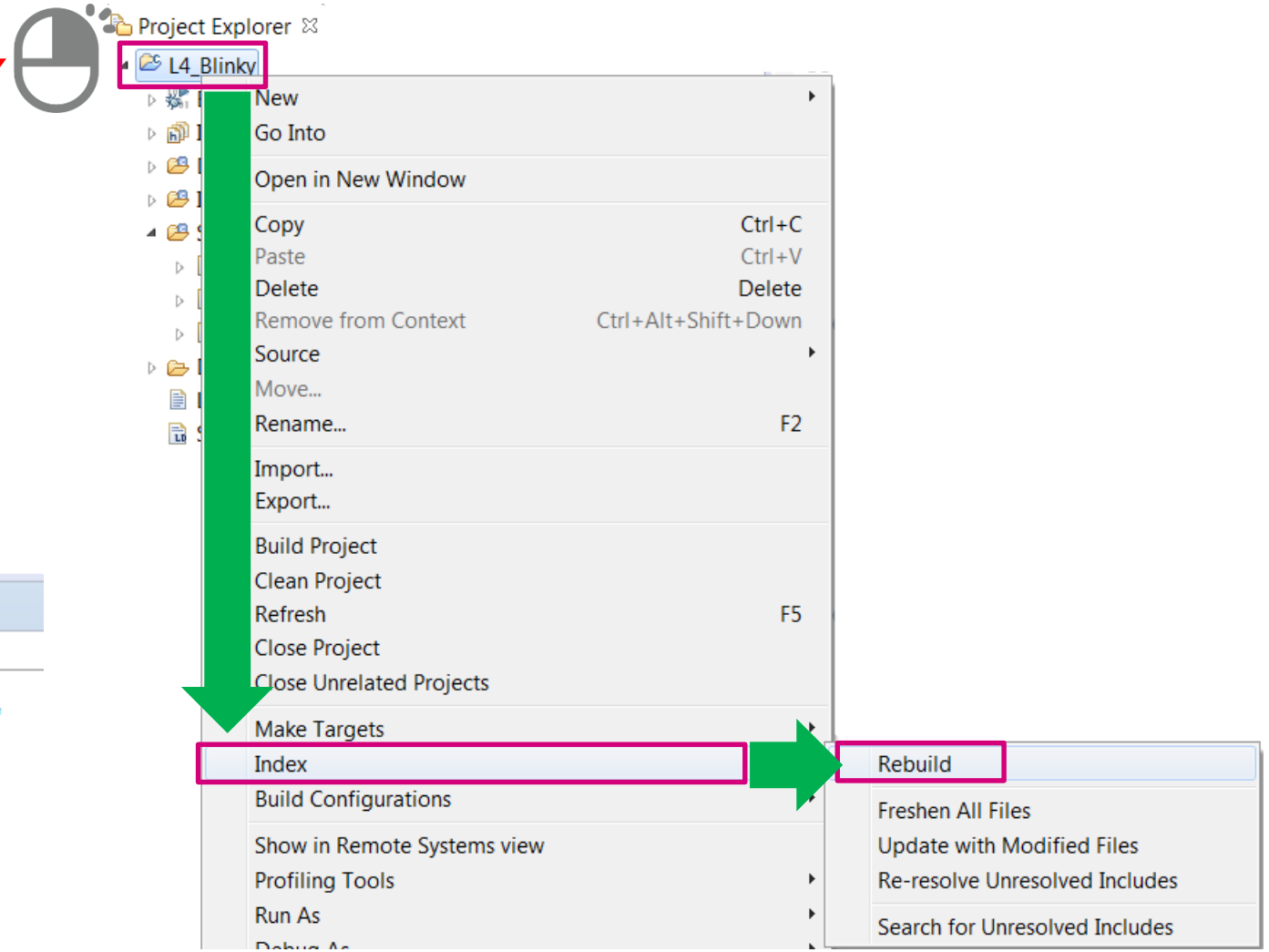


life.augmented

Building the project in SW4STM32

- To build the project press **Ctrl+B** or click Make All icon
- In case of multiple compilation errors, re-run Indexing of the project
- After proper build there are information about code/data space usage in Console window displayed

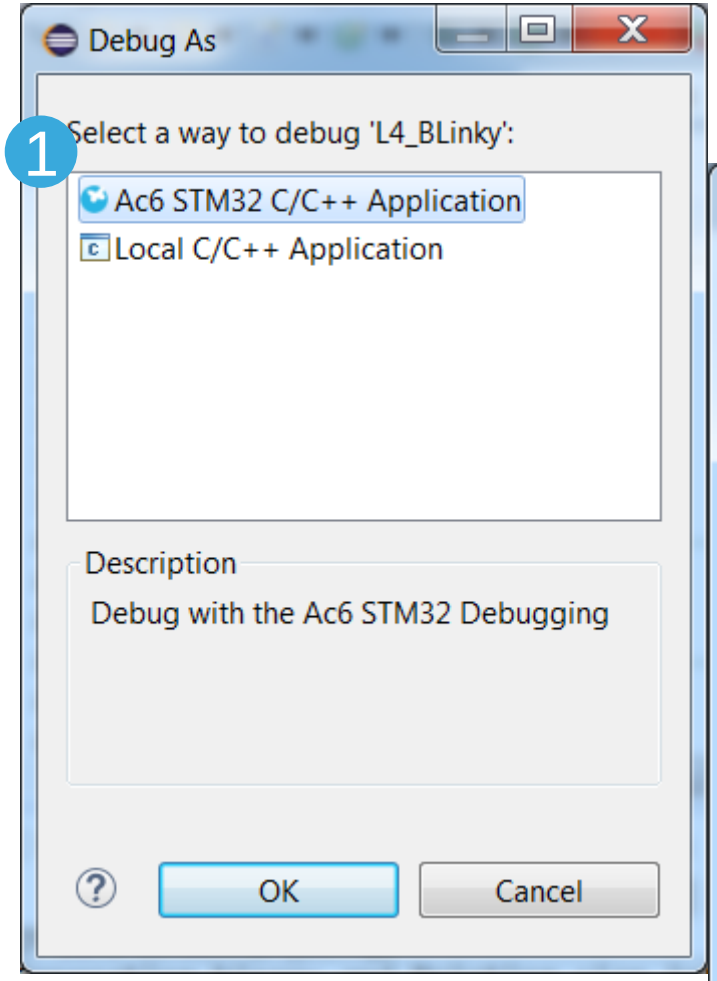


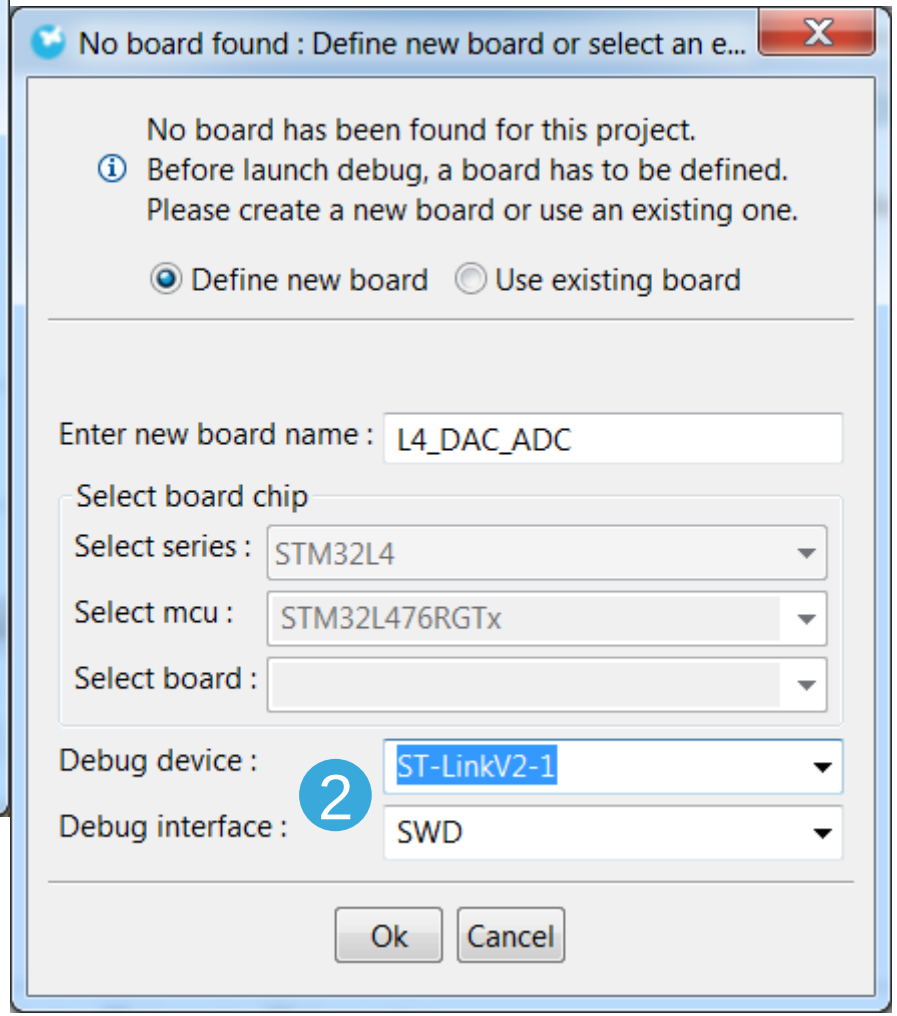


Configure the debug session in SW4STM32

for single project in the workspace

- Before running debug session this is necessary to configure it for current project
- In case there is a single project in the workspace this is enough to click the “bug” icon and:
  1. Select “Ac6 STM32 C/C++ Application” line and click ‘OK’
     In case the project was generated on existing/defined board (like NUCLEO-L476RG in our example) debug will run automatically
  2. Otherwise (we will practice it in L4_DAC_ADC example later) it is necessary to configure debug device (STLinkV2-1 in our case) and debug interface (SWD in our case) and click ‘OK’
- Next step would be to run the debug session (see the next slide)





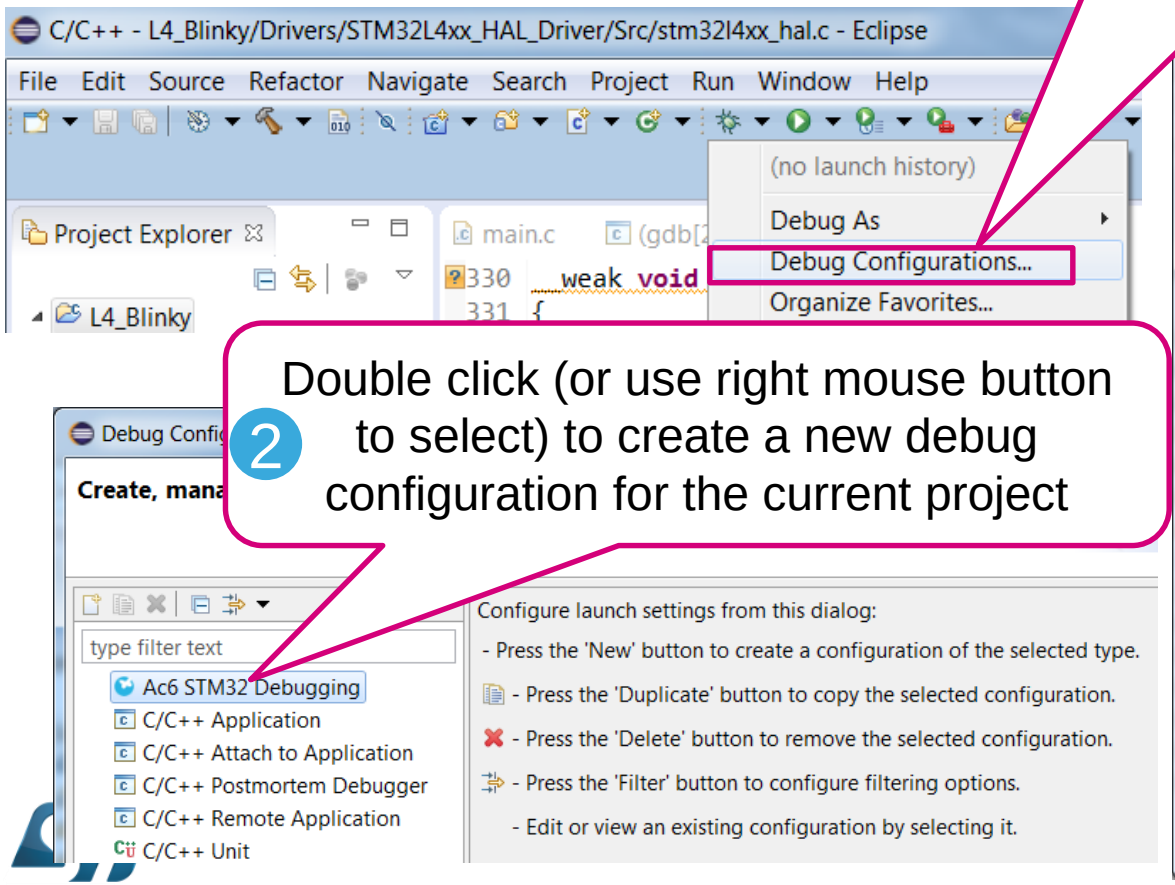
Configure the debug session in SW4STM32
for multiple projects in the workspace 1/2

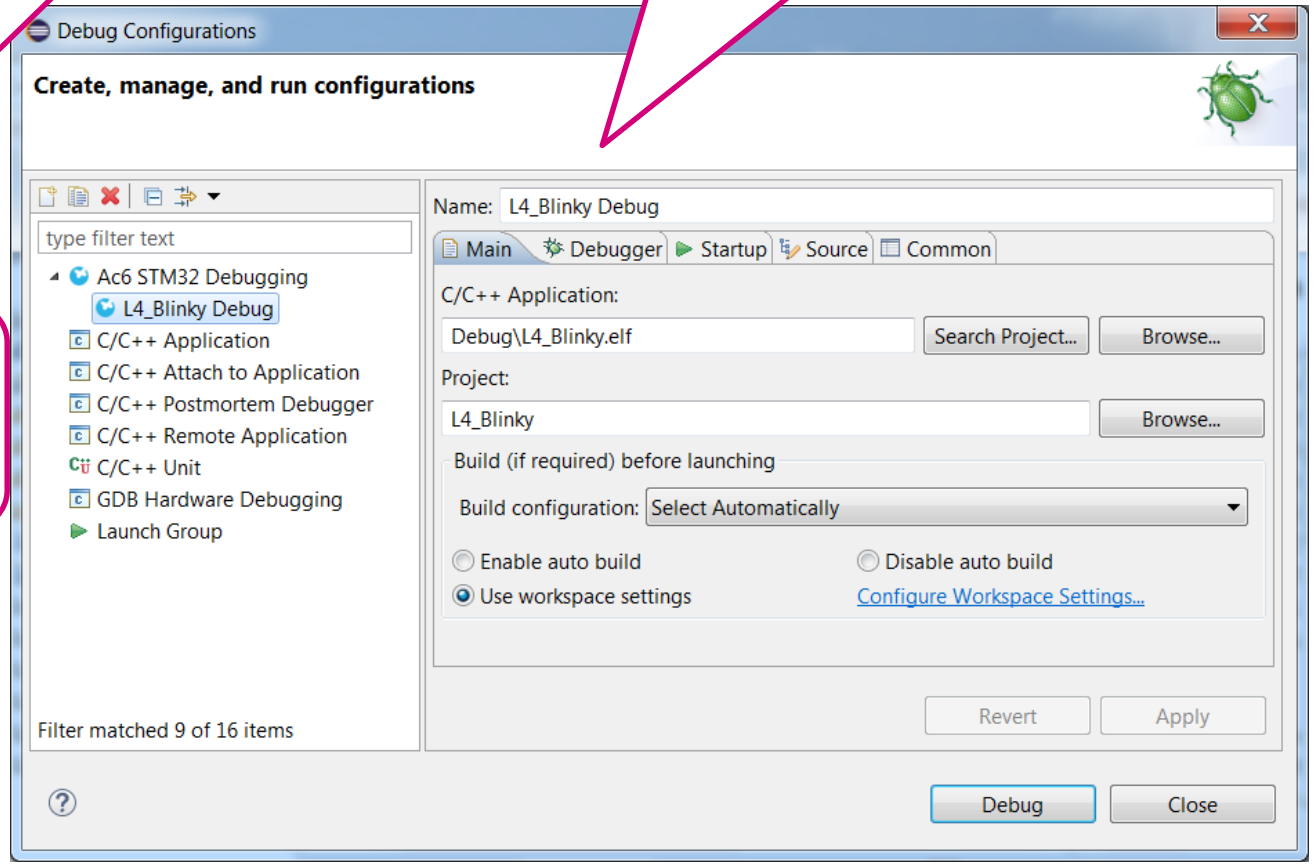
Before debugging current project for the first time, this is necessary to configure its debug session

1 Select Debug Configuration option

2 Double click (or use right mouse button to select) to create a new debug configuration for the current project

3 All project parameters should be filled-in automatically

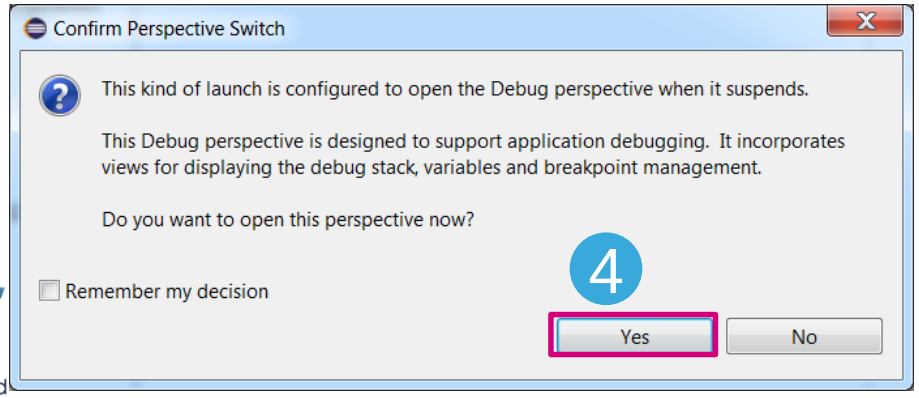


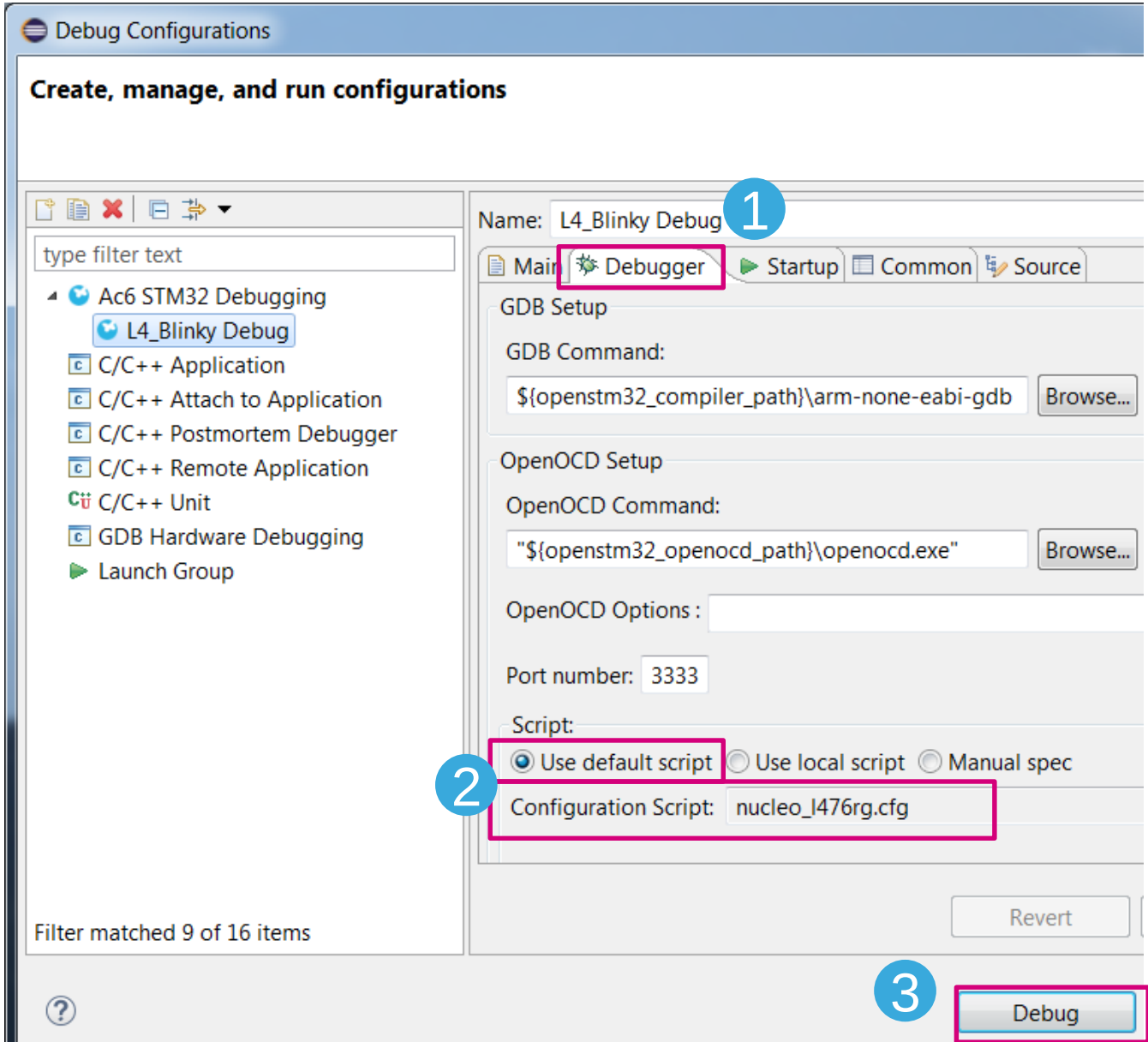


Run the debug session in SW4STM32

for multiple projects in the workspace 2/2

- Connect Nucleo board with miniUSB cable (ST-Link)
- In case of the projects generated for ST board, there should be selected board configuration script which specifies debug device and its interface (you can check it in Debugger tab)
- Debug perspective will be run (please select Yes in the information window)
- This is enough just to click a “bug” icon to enter debug session next time.

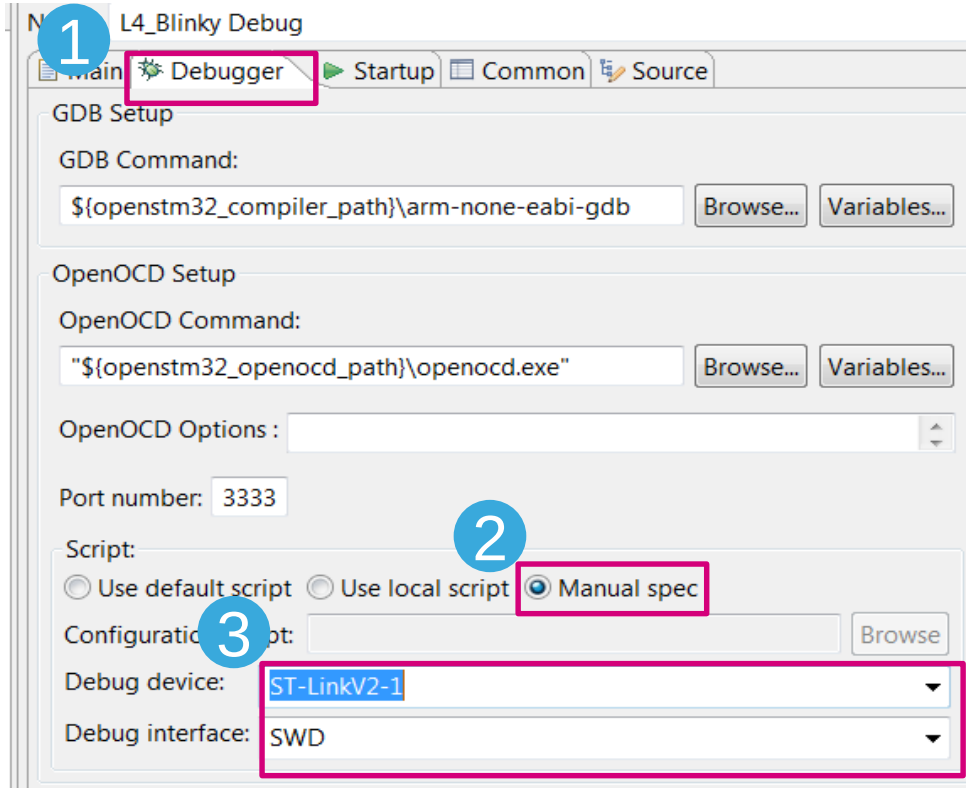


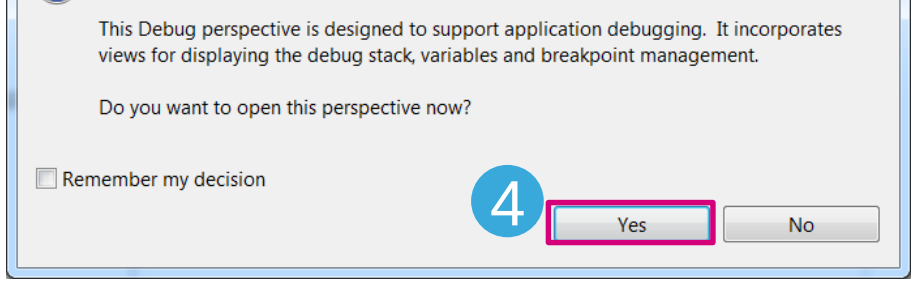


Run the debug session in SW4STM32

for multiple projects in the workspace, but no board specification

- Connect Nucleo board with miniUSB cable (ST-Link)
- Under **Debugger** tab select debug device (ST LinkV2-1 for Nucleo ones) and debug interface (SWD)
- Click **Apply** and then **Debug**
- Debug perspective will be run (please select Yes in the information window)
- This is enough just to click a “bug” icon to enter debug session next time.

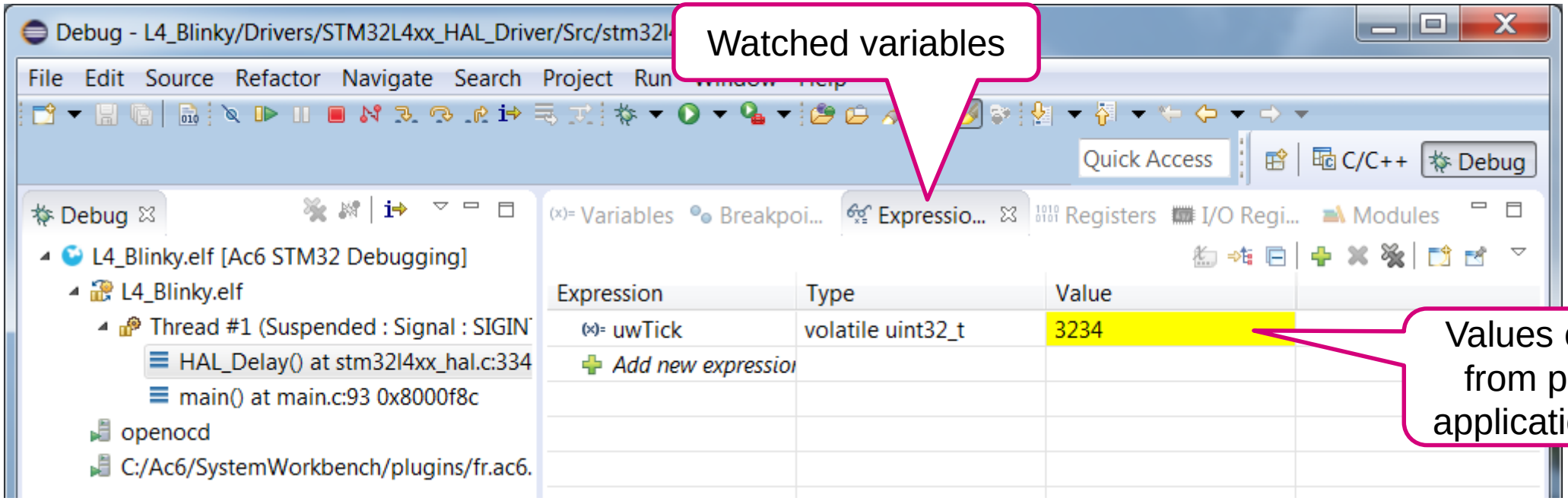




Debug session perspective

watching the variables

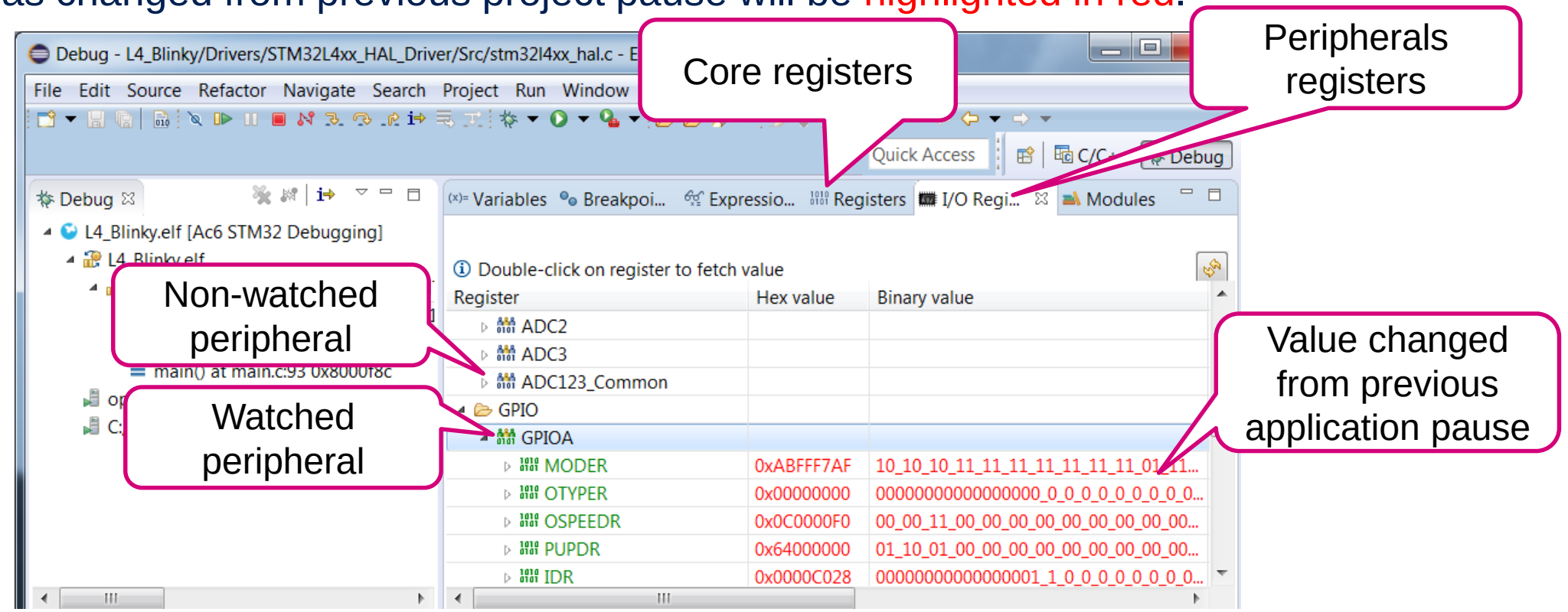
- This is possible to monitor CPU registers, peripherals registers and variables during debug session, but we need to pause the code execution (no live view is possible for the time being).
- To add variable to be monitored - highlight it, press right mouse button and select "**Add Watch Expression**". It will appear in Expressions tab then.
- Values which has changed from previous project pause will be presented on yellow background



Debug session perspective

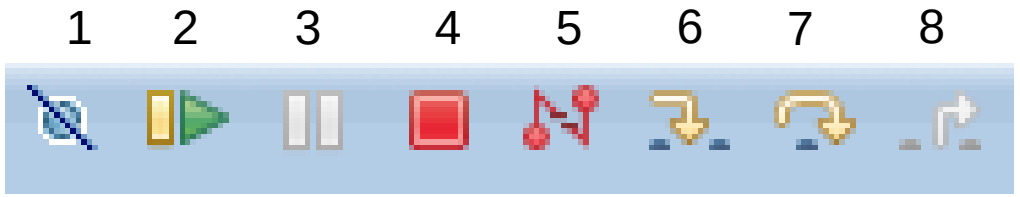
watching the registers content

- This is possible to monitor CPU registers, peripherals registers and variables during debug session, but we need to pause the code execution (no live view is possible for the time being).
- To add peripheral register to watch - click right mouse button and select “**Activate**”. Peripheral icon and its registers names will be highlighted in green and will contain “caught” values on next debug pause.
- Values which has changed from previous project pause will be highlighted in red.

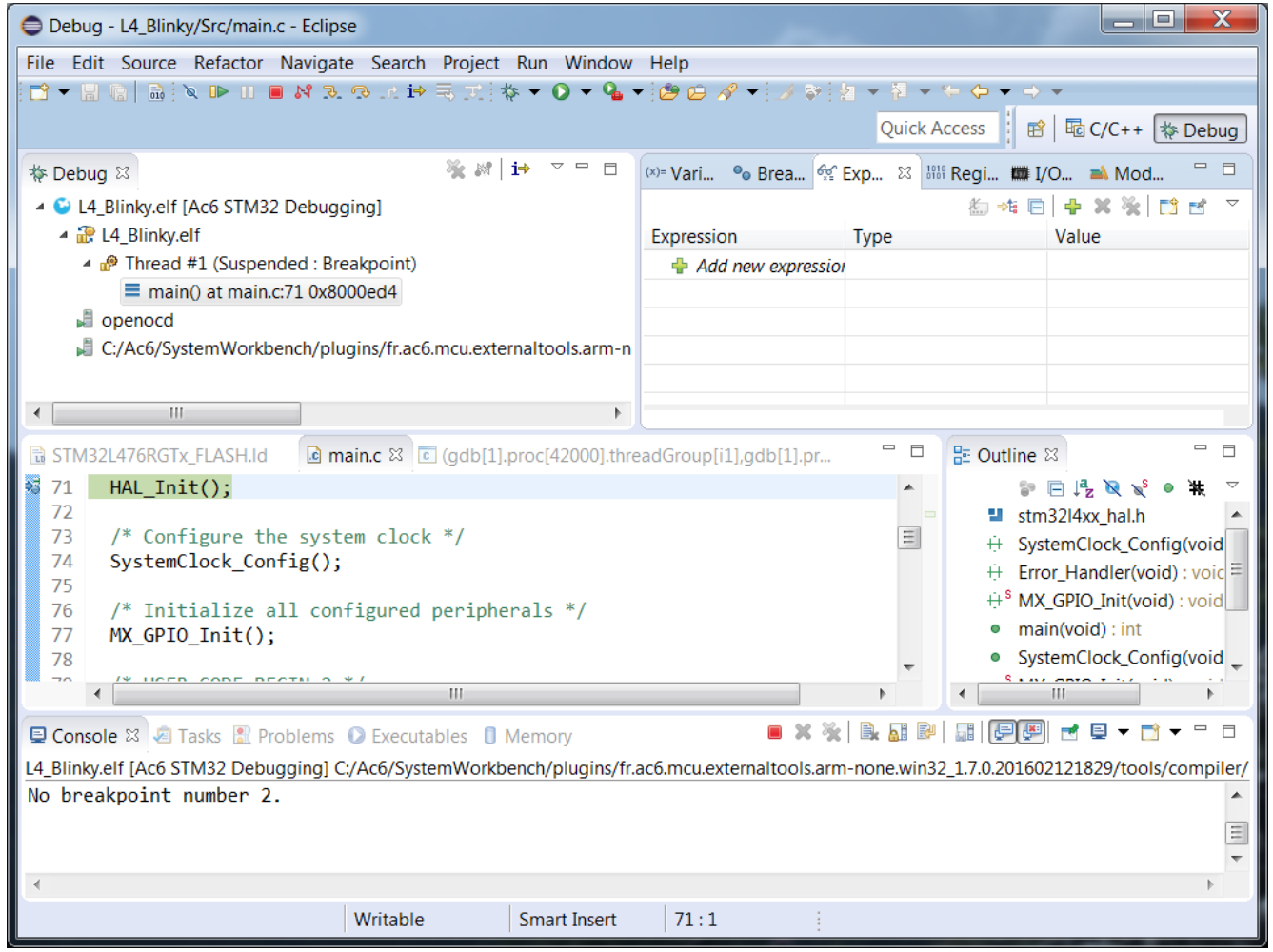


Handling the debug session

SW4STM32



1. Skip all breakpoints
2. Run/resume
3. Suspend
4. Terminate debug session
5. Disconnect from the target
6. Step into
7. Step Over
8. Step Return



windows configuration in debug perspective

What have we learnt?

- ✓Handling the projects generated by STM32CubeMX in SW4STM32
  - ✓Import project generated by STM32CubeMX
  - ✓Tune sources to run selected peripherals in desired algorithm
  - ✓Build project
  - ✓Configure debug session
  - ✓Run debug session
  - ✓Debug perspective
  - ✓Watching the variables and registers content
  - ✓Handling errors

Enjoy!

www.st.com/mcu

life.augmented