STM32WL55
Dual core project creation, coding and debug

T.O.M.A.S. Team

Goals of the session

- Demonstrate how to start dual-core project for STM32WL55J device on STM32CubeIDE
- Show how to configure peripherals and assign them to selected core
- Demonstrate how to prepare the code and build the dual core project on STM32CubeIDE
- Practice with dual core debug using STLink on STM32CubeIDE

life.augmented

Prerequisites

- NUCLEO-WL55JC2 board
- Micro USB cable
- PC with preinstalled the following software:
  - STM32CubeIDE (in version at least 1.5.0)
  - STM32WL Cube library (in version at least 1.0.0)

Application details – configuration part

- Start dual-core project for STM32WL55J device
- Assign selected peripherals (according to the table on the right) to particular cores
- Enable interrupts on EXTI lines where B1-B3 buttons are connected
- Enable internal pull-ups on B1-B3 lines
- Keep default clock settings (4 MHz from MSI) for both cores
- Generate the code for STM32CubeIDE

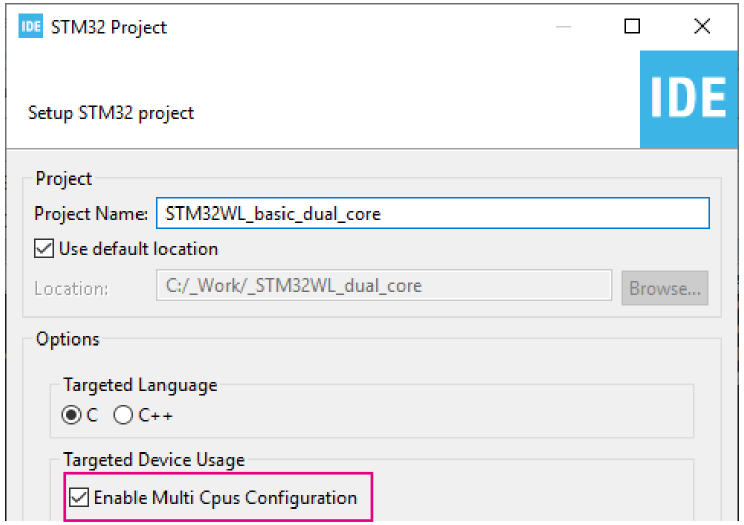
	pins	CM0PLUS	CM4
B1 button	PA0	+	
B2 button	PA1		+
B3 button	PC6	+	
LED1	PB15		+
LED2	PB9	+	
LED3	PB11		+

STM32WL55 dual core project creation

Application details – configuration hints

Some project preparation hints:

- Enable multi CPUs configuration at new project creation phase:



STM32 Project

Setup STM32 project

Project

Project Name: STM32WL_basic_dual_core

☑ Use default location

Location: C:/_Work/_STM32WL_dual_core

Browse...

Options

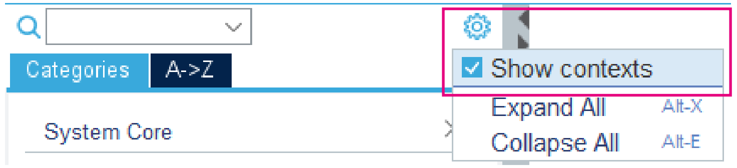
Targeted Language

● C ○ C++

Targeted Device Usage

☑ Enable Multi Cpus Configuration

- Within peripherals configuration enable „Show contexts”



Categories A->Z

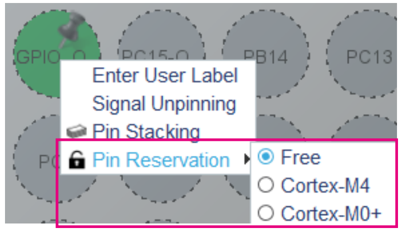
System Core

☑ Show contexts

Expand All Alt-X

Collapse All Alt-E

- You can assign particular pin to any of the core by clicking right button on mouse and select „Pin Reservation”.



Enter User Label

Signal Unpinning

Pin Stacking

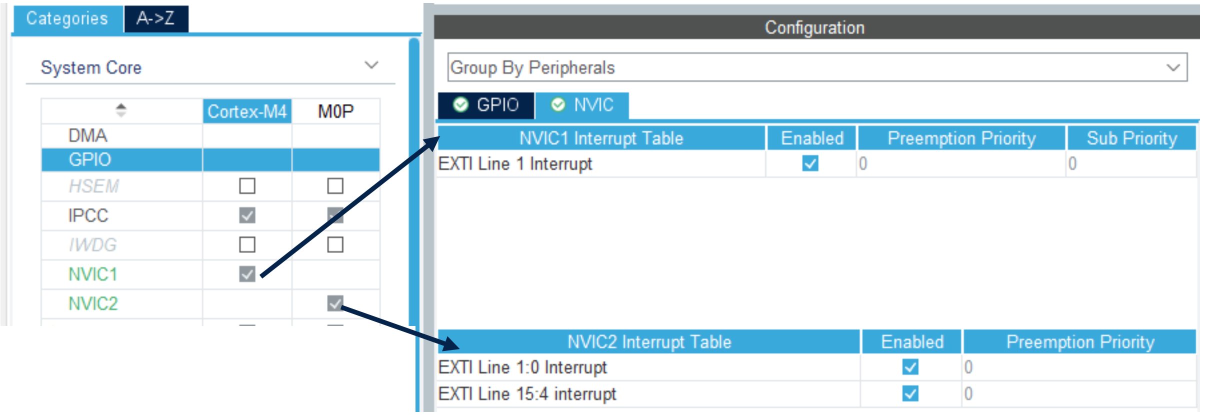
Pin Reservation

● Free

○ Cortex-M4

○ Cortex-M0+

- We have separate NVIC for each core



Categories A->Z

System Core

	Cortex-M4	M0P
DMA		
GPIO		
HSEM	□	□
IPCC	☑	☑
IWDG	□	□
NVIC1	☑	
NVIC2		☑

Configuration

Group By Peripherals

GPIO NVIC

NVIC1 Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI Line 1 Interrupt	☑	0	0

NVIC2 Interrupt Table	Enabled	Preemption Priority
EXTI Line 1:0 Interrupt	☑	0
EXTI Line 15:4 interrupt	☑	0

Application details – code generation

STM32CubeMX and STM32CubeIDE during project creation:

1. is generating code for both cores (marked CM0PLUS and CM4)
2. within ARM CortexM4 core there is a line which activates second core

   ```
   HAL_PWREx_ReleaseCore(PWR_CORE_CPU2);
   ```

3. creates two linker files with the following memory usage settings:
   - ARM CortexM4 : FLASH starts at 0x0800 0000 (128k size), RAM starts at 0x2000 0000 (32k size)
   - ARM CortexM0+ : FLASH starts at 0x0802 0000 (128k size), RAM starts at 0x2000 8000 (32k size)

Dual core project modification

life.augmented

Application details – coding part

- Once per second we will increment *var_CM4* and *var_CM0P* variables within main while() loop for both cores
- The rest of user code will be stored within external interrupts callbacks:
    - B1 button press should toggle LED2
    - B2 button press should toggle LED1 and LED3
    - B3 button press should toggle LED2
- „weak" definition of the callback can be found within **stm32g0xx_hal_gpio.c** file (to be tracked from EXTI IRQ handler within **stm32g0xx_it.c** file):

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

	pins	CM0PLUS	CM4
B1 button	PA0	+	
B2 button	PA1		+
B3 button	PC6	+	
LED1	PB15		+
LED2	PB9	+	
LED3	PB11		+

life.augmented

Application details – coding part

- Once per second we will increment *var_CM4* and *var_CM0P* variables within main while() loop for both cores

```c
/* USER CODE BEGIN PV */
uint16_t var_CM0P=0;

....

 /* USER CODE BEGIN WHILE */
  while (1)
  {
  var_CM0P++;
  HAL_Delay(1000);
 /* USER CODE END WHILE */
```

main.c file within ARM CortexM0+ code

```c
/* USER CODE BEGIN PV */
uint16_t var_CM4=0;

....

 /* USER CODE BEGIN WHILE */
  while (1)
  {
  var_CM4++;
  HAL_Delay(1000);
 /* USER CODE END WHILE */
```

main.c file within ARM CortexM4 code

Application details – coding part

- We need to implement external interrupts callbacks for both cores within **main.c** files (i.e. within USER CODE 4 section)

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
 if(GPIO_PIN_0 == GPIO_Pin)
 {
  HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
 }
 else if(GPIO_PIN_6 == GPIO_Pin)
 {
  HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
 }
}
/* USER CODE END 4 */
```

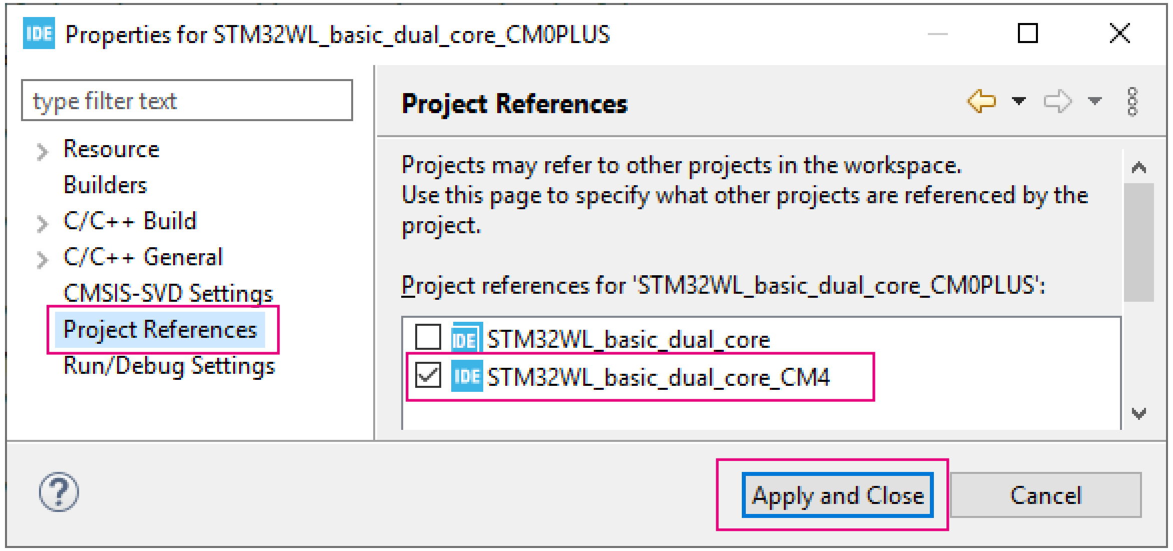
main.c file within ARM CortexM0+ code

```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
 if(GPIO_PIN_1 == GPIO_Pin)
 {
  HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
  HAL_GPIO_TogglePin(LED3_GPIO_Port, LED3_Pin);
 }
}
/* USER CODE END 4 */
```

main.c file within ARM CortexM4 code

Application build configuration

- Both core contains its independent projects, but we can configure more automatic build which will always build code from both projects in the same order (suggested way: CM4 first, then CM0PLUS)
- To perform such configuration, please click on right button on mouse over CM0PLUS project name and select Properties, then Project References and select CM4 project



Properties for STM32WL_basic_dual_core_CM0PLUS

Project References

Projects may refer to other projects in the workspace.
Use this page to specify what other projects are referenced by the project.

Project references for 'STM32WL_basic_dual_core_CM0PLUS':

☐ STM32WL_basic_dual_core
☑ STM32WL_basic_dual_core_CM4

- Now if you click on build button once your active project is CM0PLUS will trigger build of CM4 project first, then CM0PLUS one

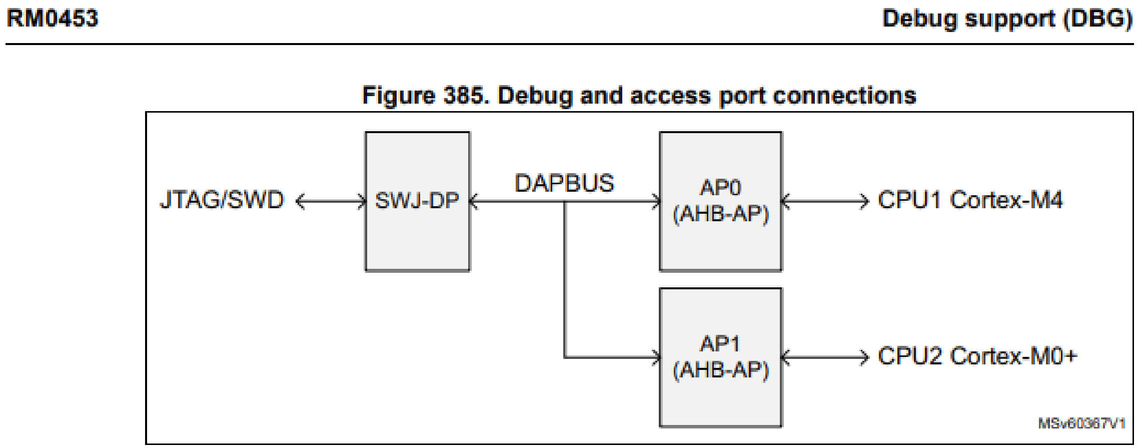
life.augmented

Dual core project debug configuration

life.augmented

STM32WL55 debug ports extract from Reference Manual (RM0453)

- Within dual core STM32WL55 MCU we have 2 debug ports:
  - Port 0 – dedicated to CortexM4 core
  - Port 1 – dedicated to CortexM0+ core



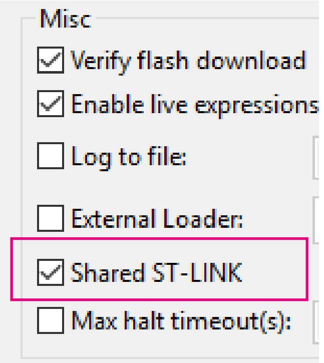
RM0453

Debug support (DBG)

Figure 385. Debug and access port connections

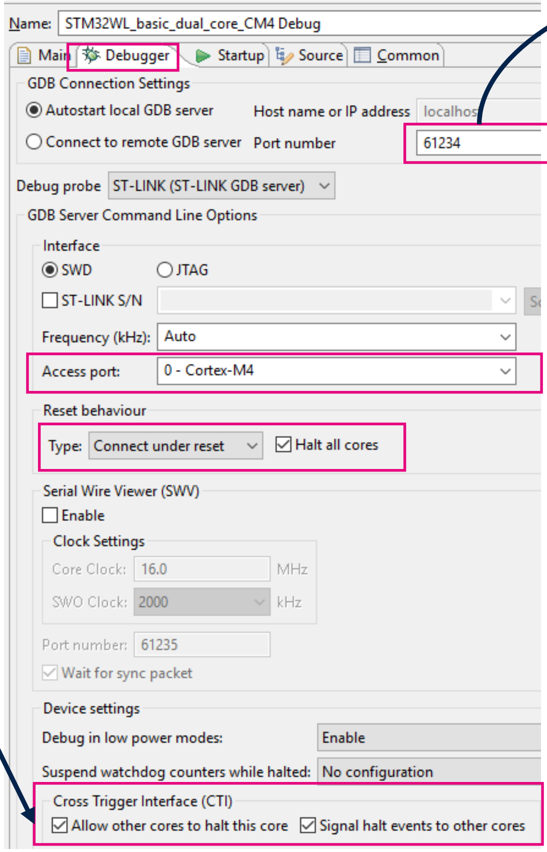
STM32WL55 debug ports configuration of debug session in STM32CubeIDE

- We need to specify correctly GDB Port number :
  - For ARM CortexM4 core -> Port number **61234** (default)
  - For ARM CortexM0+ core -> Port number = port selected for ARM CortexM4 **+ 1 (or higher)**
- Select access port:
  - „0 – Cortex-M4” for CortexM4 part
  - „1 – CortexM0plus” for CortexM0+ part
- Select reset behaviour only for CorexM4
- In both configuration select both options within CTI part
- In both configurations select „Shared ST-LINK”



Misc
☑ Verify flash download
☑ Enable live expressions
☐ Log to file:
☐ External Loader:
☑ Shared ST-LINK
☐ Max halt timeout(s):

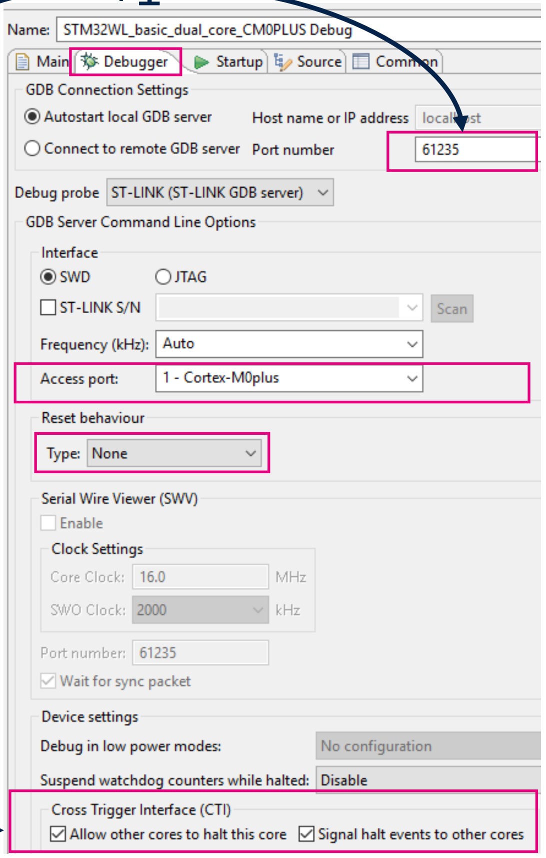
+1



Name: STM32WL_basic_dual_core_CM4 Debug

GDB Connection Settings
● Autostart local GDB server   Host name or IP address: localhos
○ Connect to remote GDB server   Port number: 61234
Debug probe: ST-LINK (ST-LINK GDB server)
GDB Server Command Line Options
Interface
● SWD ○ JTAG
☐ ST-LINK S/N
Frequency (kHz): Auto
Access port: 0 - Cortex-M4
Reset behaviour
Type: Connect under reset ☑ Halt all cores
Serial Wire Viewer (SWV)
☐ Enable
Clock Settings
Core Clock: 16.0 MHz
SWO Clock: 2000 kHz
Port number: 61235
☑ Wait for sync packet
Device settings
Debug in low power modes: Enable
Suspend watchdog counters while halted: No configuration
Cross Trigger Interface (CTI)
☑ Allow other cores to halt this core ☑ Signal halt events to other cores

ARM CortexM4 core



Name: STM32WL_basic_dual_core_CM0PLUS Debug

GDB Connection Settings
● Autostart local GDB server   Host name or IP address: localhost
○ Connect to remote GDB server   Port number: 61235
Debug probe: ST-LINK (ST-LINK GDB server)
GDB Server Command Line Options
Interface
● SWD ○ JTAG
☐ ST-LINK S/N   Scan
Frequency (kHz): Auto
Access port: 1 - Cortex-M0plus
Reset behaviour
Type: None
Serial Wire Viewer (SWV)
☐ Enable
Clock Settings
Core Clock: 16.0 MHz
SWO Clock: 2000 kHz
Port number: 61235
☑ Wait for sync packet
Device settings
Debug in low power modes: No configuration
Suspend watchdog counters while halted: Disable
Cross Trigger Interface (CTI)
☑ Allow other cores to halt this core ☑ Signal halt events to other cores

ARM CortexM0+ core

STM32WL55 debug configuration startup configuration in STM32CubeIDE - code

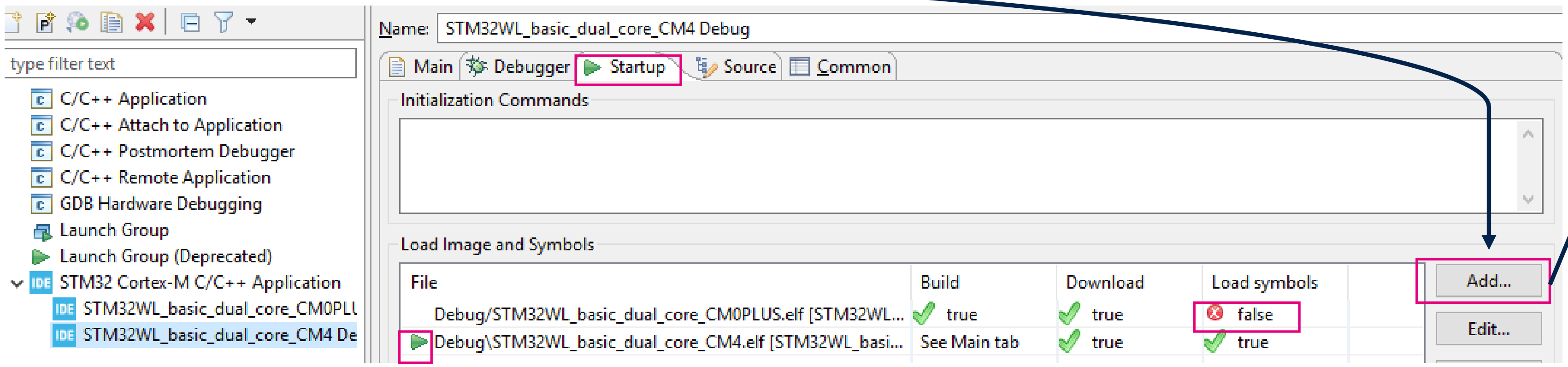
- Within dual core STM32WL55 MCU ARMCortexM4 is starting automatically after the reset and ARMCortexM0+ should be turned on
- It can be done i.e. Using the code:

```
HAL_PWREx_ReleaseCore(PWR_CORE_CPU2);
```

- This line of code is automatically added by STM32CubeMX and STM32CubeIDE to ARMCortexM4 code once we select dual core project creation

STM32WL55 debug configuration
how to configure startup debug for dual core

- Within debug startup configuration for ARMCortexM4 it is necessary to add ARMCortexM0+ part to be build and flashed together



type filter text

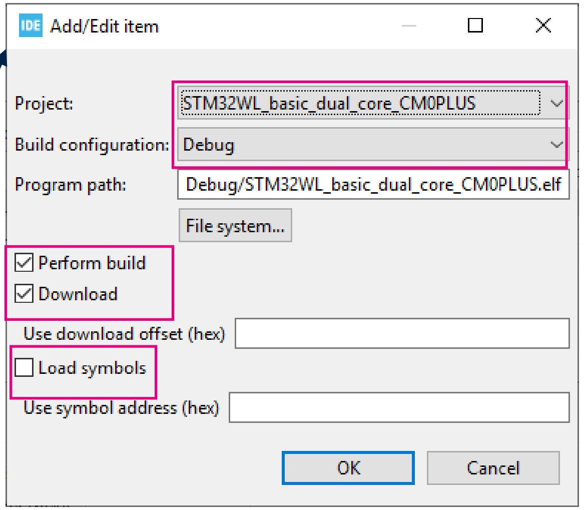
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)
- STM32 Cortex-M C/C++ Application
  - STM32WL_basic_dual_core_CM0PLU
  - STM32WL_basic_dual_core_CM4 De

Name: STM32WL_basic_dual_core_CM4 Debug

Initialization Commands

Load Image and Symbols

File	Build	Download	Load symbols
Debug/STM32WL_basic_dual_core_CM0PLUS.elf [STM32WL...	✓ true	✓ true	☒ false
Debug\STM32WL_basic_dual_core_CM4.elf [STM32WL_basi...	See Main tab	✓ true	✓ true



Add/Edit item

Project: STM32WL_basic_dual_core_CM0PLUS

Build configuration: Debug

Program path: Debug/STM32WL_basic_dual_core_CM0PLUS.elf

☑ Perform build

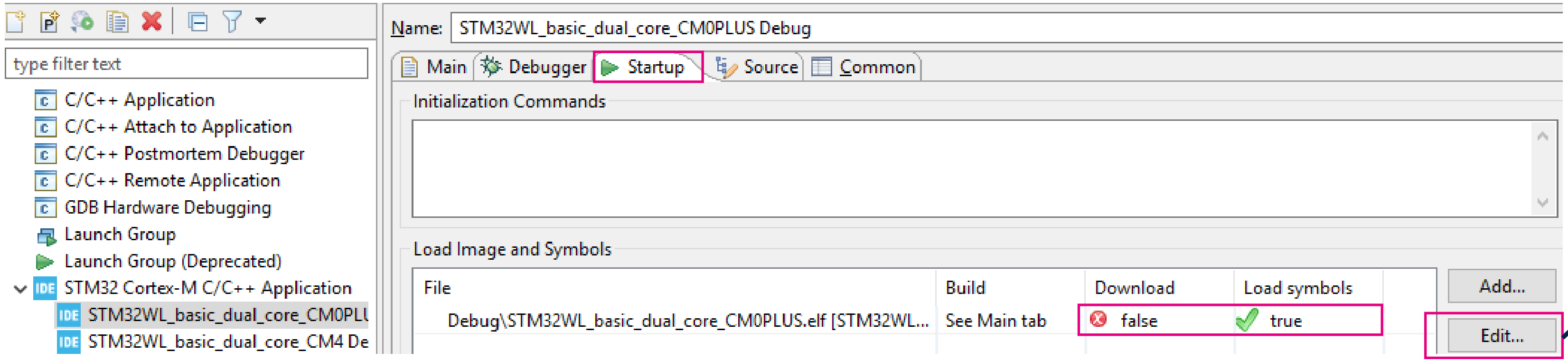
☑ Download

Use download offset (hex)

☐ Load symbols

Use symbol address (hex)

- Within debug startup configuration for ARM CortexM0+ we need to load only symbols for debug session (as code is already loaded within CM4 debug start)



type filter text

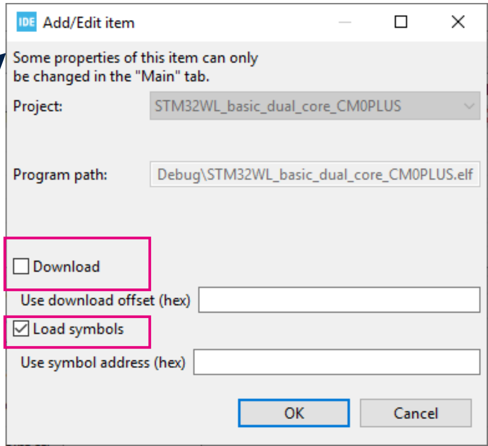
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)
- STM32 Cortex-M C/C++ Application
  - STM32WL_basic_dual_core_CM0PLU
  - STM32WL_basic_dual_core_CM4 De

Name: STM32WL_basic_dual_core_CM0PLUS Debug

Initialization Commands

Load Image and Symbols

File	Build	Download	Load symbols
Debug\STM32WL_basic_dual_core_CM0PLUS.elf [STM32WL...	See Main tab	☒ false	✓ true



Add/Edit item

Some properties of this item can only be changed in the "Main" tab.

Project: STM32WL_basic_dual_core_CM0PLUS

Program path: Debug\STM32WL_basic_dual_core_CM0PLUS.elf

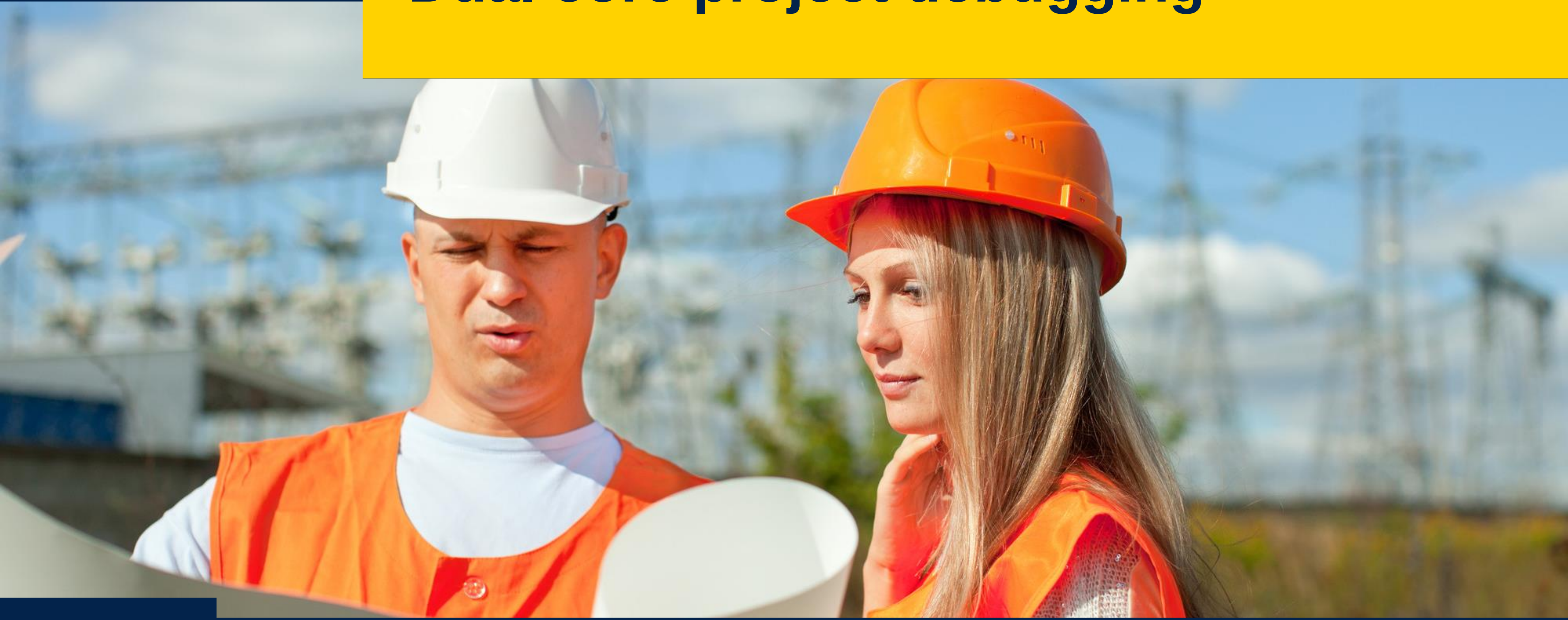
☐ Download

Use download offset (hex)

☑ Load symbols

Use symbol address (hex)

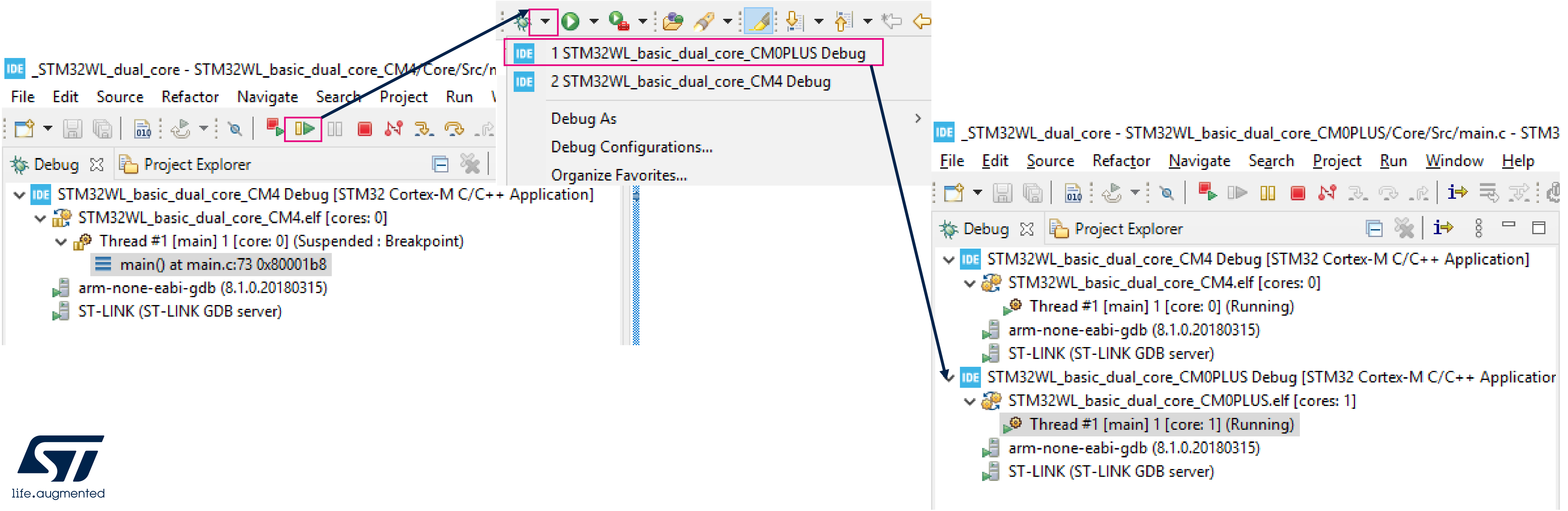
Dual core project debugging



life.augmented

STM32WL55 debug configuration how to start debug session for dual core

- Start debug session for ARM CortexM4 core
- Start code execution within debug perspective (to start ARM CortexM0+ core)
- Start debug session for ARM CortexM0+ core (it will load only symbols)



_STM32WL_dual_core - STM32WL_basic_dual_core_CM4/Core/Src/n

File Edit Source Refactor Navigate Search Project Run \

Debug | Project Explorer

```
STM32WL_basic_dual_core_CM4 Debug [STM32 Cortex-M C/C++ Application]
  STM32WL_basic_dual_core_CM4.elf [cores: 0]
    Thread #1 [main] 1 [core: 0] (Suspended : Breakpoint)
      main() at main.c:73 0x80001b8
  arm-none-eabi-gdb (8.1.0.20180315)
  ST-LINK (ST-LINK GDB server)
```

1 STM32WL_basic_dual_core_CM0PLUS Debug

2 STM32WL_basic_dual_core_CM4 Debug

Debug As

Debug Configurations...

Organize Favorites...

_STM32WL_dual_core - STM32WL_basic_dual_core_CM0PLUS/Core/Src/main.c - STM3

File Edit Source Refactor Navigate Search Project Run Window Help

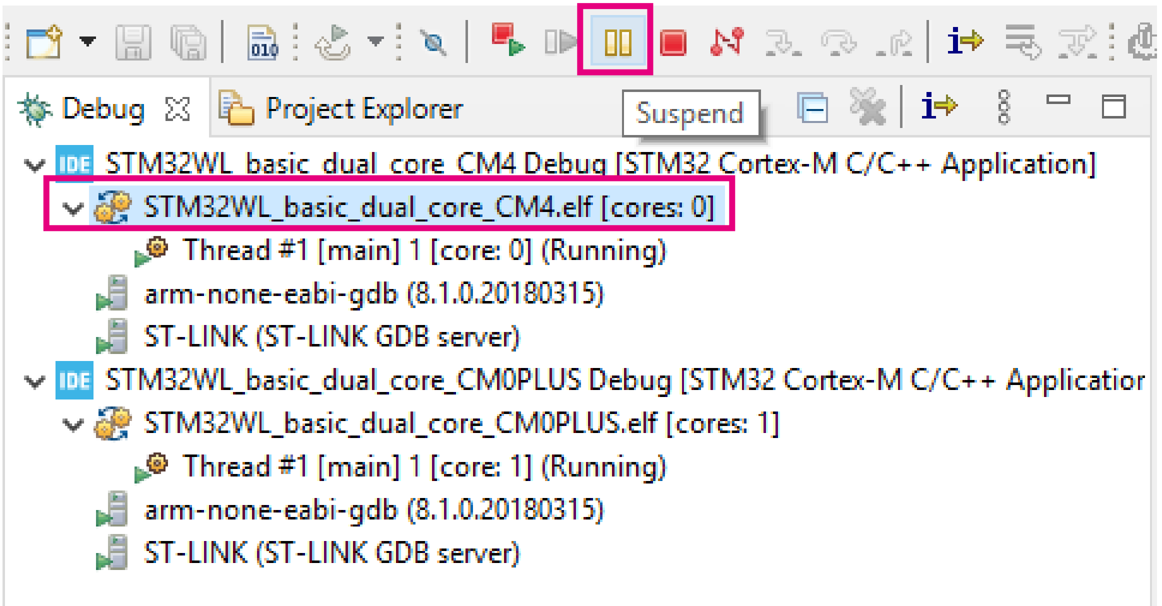
Debug | Project Explorer

```
STM32WL_basic_dual_core_CM4 Debug [STM32 Cortex-M C/C++ Application]
  STM32WL_basic_dual_core_CM4.elf [cores: 0]
    Thread #1 [main] 1 [core: 0] (Running)
  arm-none-eabi-gdb (8.1.0.20180315)
  ST-LINK (ST-LINK GDB server)
STM32WL_basic_dual_core_CM0PLUS Debug [STM32 Cortex-M C/C++ Applicatior
  STM32WL_basic_dual_core_CM0PLUS.elf [cores: 1]
    Thread #1 [main] 1 [core: 1] (Running)
  arm-none-eabi-gdb (8.1.0.20180315)
  ST-LINK (ST-LINK GDB server)
```

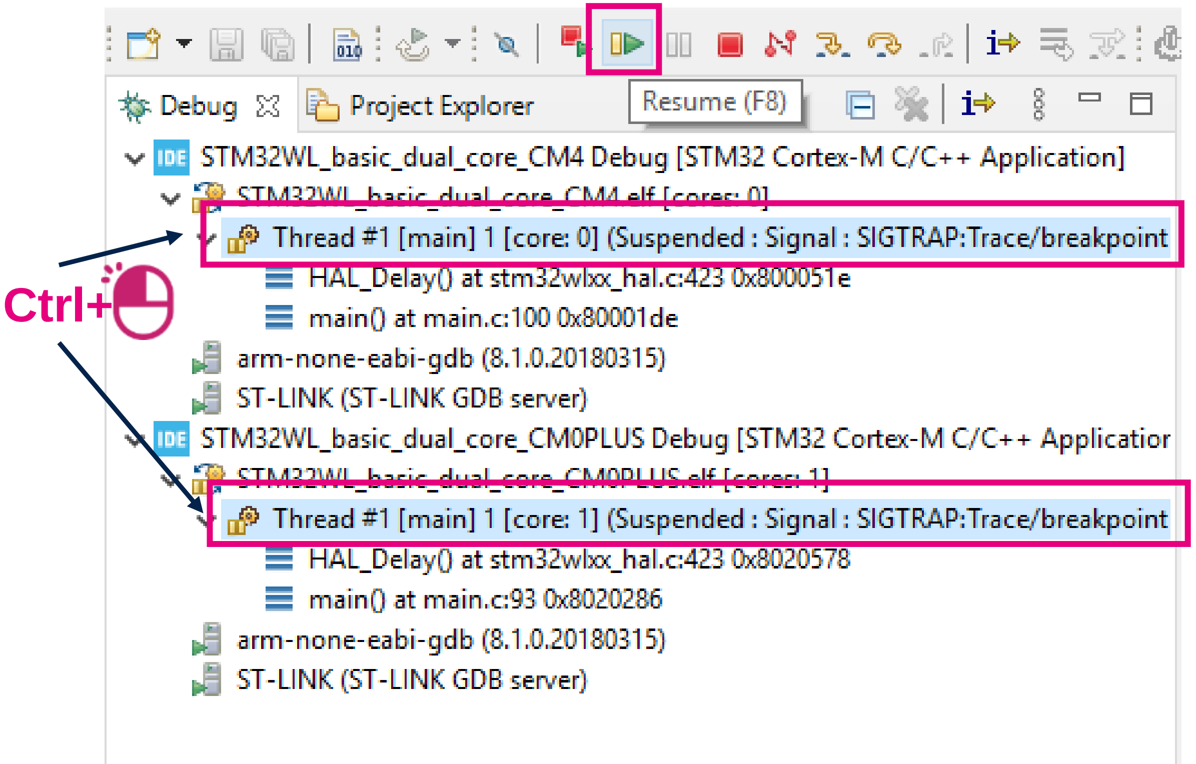
Synchronous pause and resume execution on both cores

- To suspend both cores stop at breakpoint or suspend using button any of the cores

- To resume both cores from suspend mode select both cores (threads) keeping Ctrl button pressed then press Resume button

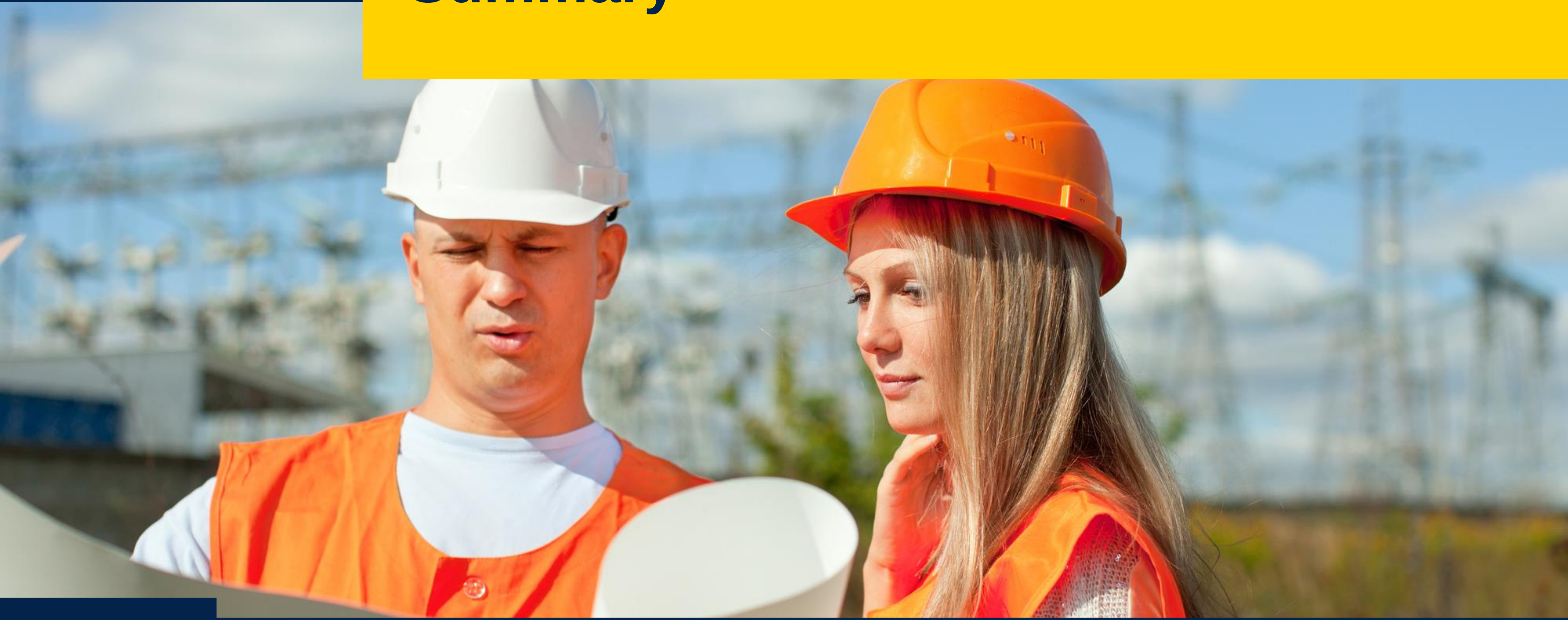


STM32WL_basic_dual_core_CM4 Debug [STM32 Cortex-M C/C++ Application]
STM32WL_basic_dual_core_CM4.elf [cores: 0]
Thread #1 [main] 1 [core: 0] (Running)
arm-none-eabi-gdb (8.1.0.20180315)
ST-LINK (ST-LINK GDB server)
STM32WL_basic_dual_core_CM0PLUS Debug [STM32 Cortex-M C/C++ Applicatior
STM32WL_basic_dual_core_CM0PLUS.elf [cores: 1]
Thread #1 [main] 1 [core: 1] (Running)
arm-none-eabi-gdb (8.1.0.20180315)
ST-LINK (ST-LINK GDB server)



STM32WL_basic_dual_core_CM4 Debug [STM32 Cortex-M C/C++ Application]
STM32WL_basic_dual_core_CM4.elf [cores: 0]
Thread #1 [main] 1 [core: 0] (Suspended : Signal : SIGTRAP:Trace/breakpoint
HAL_Delay() at stm32wlxx_hal.c:423 0x800051e
main() at main.c:100 0x80001de
arm-none-eabi-gdb (8.1.0.20180315)
ST-LINK (ST-LINK GDB server)
STM32WL_basic_dual_core_CM0PLUS Debug [STM32 Cortex-M C/C++ Applicatior
STM32WL_basic_dual_core_CM0PLUS.elf [cores: 1]
Thread #1 [main] 1 [core: 1] (Suspended : Signal : SIGTRAP:Trace/breakpoint
HAL_Delay() at stm32wlxx_hal.c:423 0x8020578
main() at main.c:93 0x8020286
arm-none-eabi-gdb (8.1.0.20180315)
ST-LINK (ST-LINK GDB server)

Summary



life.augmented

Further information

- AN5564 - Getting started with projects based on dual-core STM32WL microcontrollers in STM32CubeIDE

Thank you

life.augmented