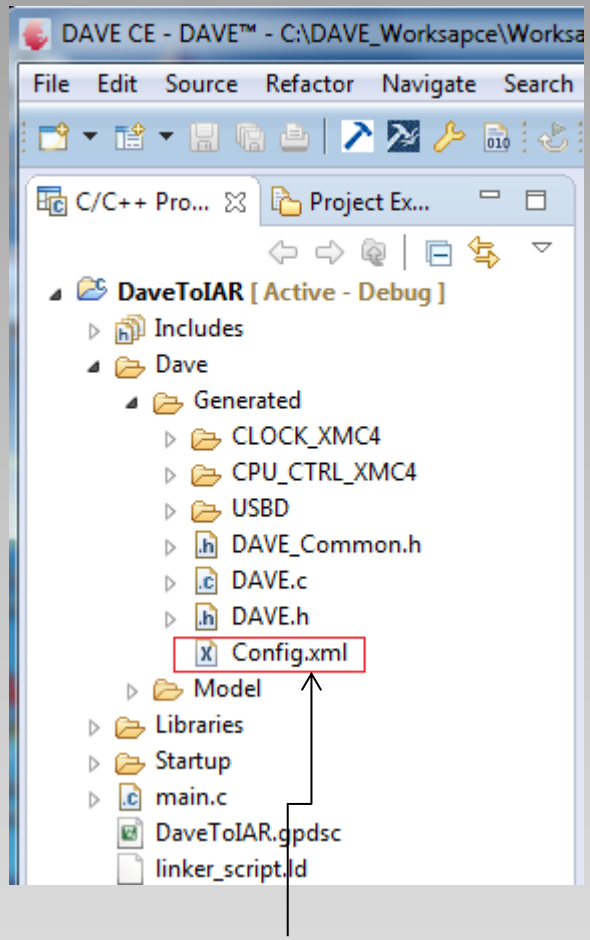
IAR Embedded Workbench for ARM

Lab Instructions

Munich, May 12 2015

DAVE™ v4 integration concept

DAVE project outline

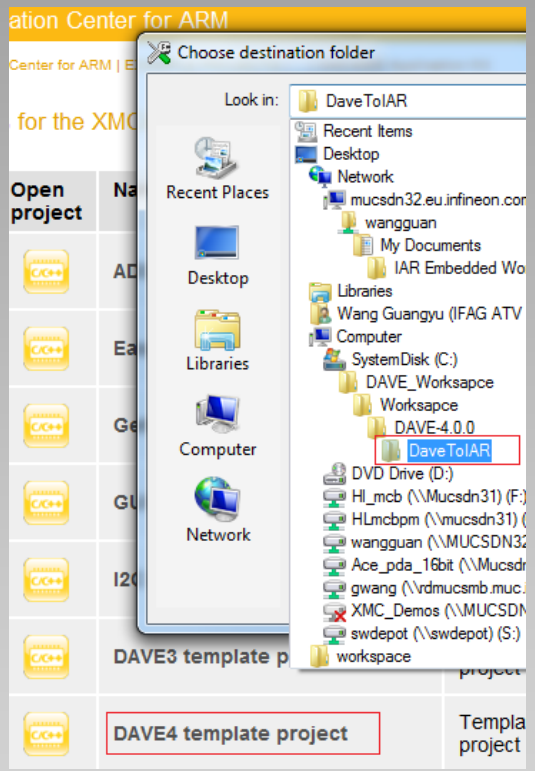


Config.xml file references only the generated sources and header files

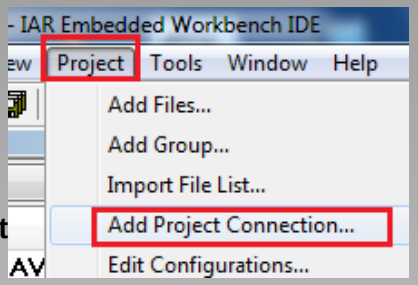
- DAVE v4 is a free eclipse based development platform that can generate application libraries from DAVE Apps.
- The generated library sources are included in a DAVE v4 project.
- DAVE can generate a Config.xml file that contains the path and names of all generated library sources (c source files and header files).
- IAR Embedded Workbench for ARM uses the information in Config.xml file to import the DAVE library sources and header files into IAR project .
- The Config.xml file references only the generated sources and headers, It is expected that the user develops the user code afterwards in the IAR Embedded Workbench project
- Not the full DAVE v4 project is ported. To port the full DAVE project the existing user code has to be referenced manually

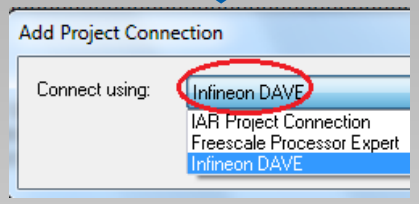
DAVE™ v4 integration concept cont'd

Open DAVE Template project and save a local copy

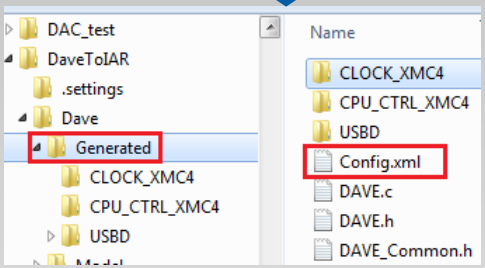


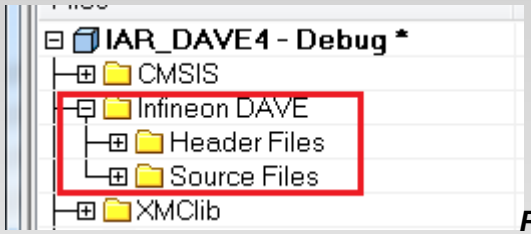
Add Infineon DAVE project connection





Go to config.xml file





Respective project view in IAR EWARM

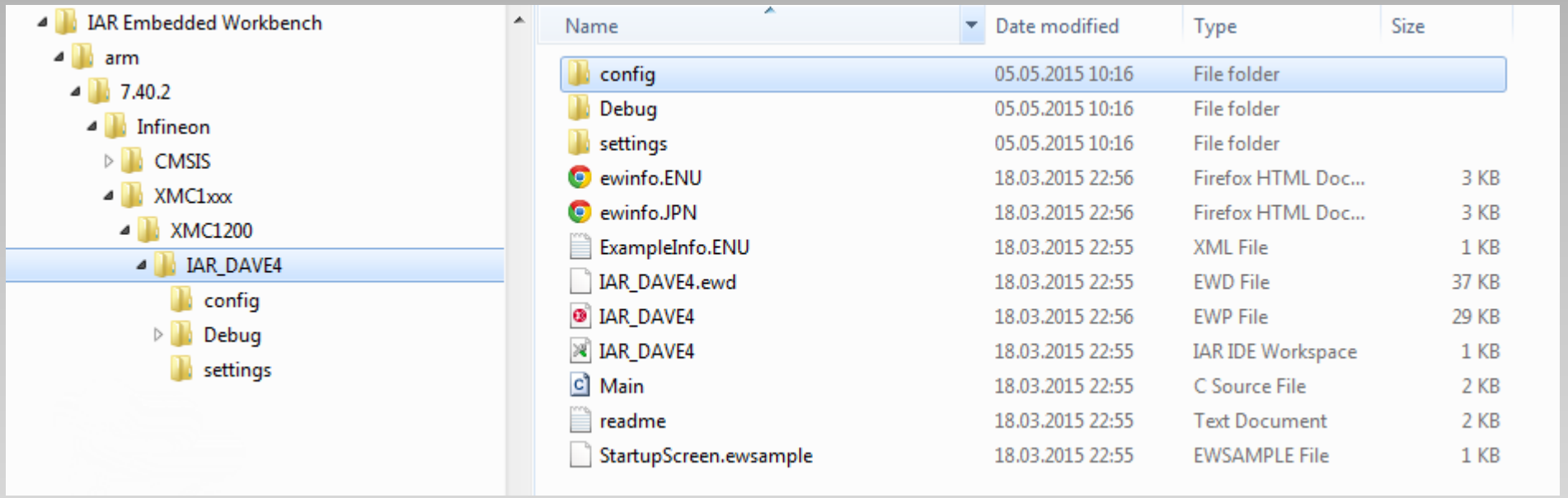
- In the Infineon examples there is a **DAVE4** template project for each device family.
- In IAR Information Center, go to the Infineon examples, open DAVE4 template project and save a local copy in a folder of your choice
- In the Template project add the project connection using Config.xml file.
- If the template project is saved in the project folder of the DAVE (eclipse) project, build in DAVE will now create build error. This can be avoided excluding these files from build in the DAVE project

Prerequisites to Follow the Tutorial

- IAR EWARM version 7.40 or higher
- Download the Infineon example projects in “IAR Information Center for ARM” in IAR EWARM
- **open** the “Infineon Examples”
- **Go to**:
  - **For XMC4xxx**: XMC4xxx->XMC4xxx Application Kit
  - **For XMC1xxx**: XMC1xxx->XMC1xxx Boot Kit board
- Open “**DAVE4** template project”
- Then, do the following step by step.....

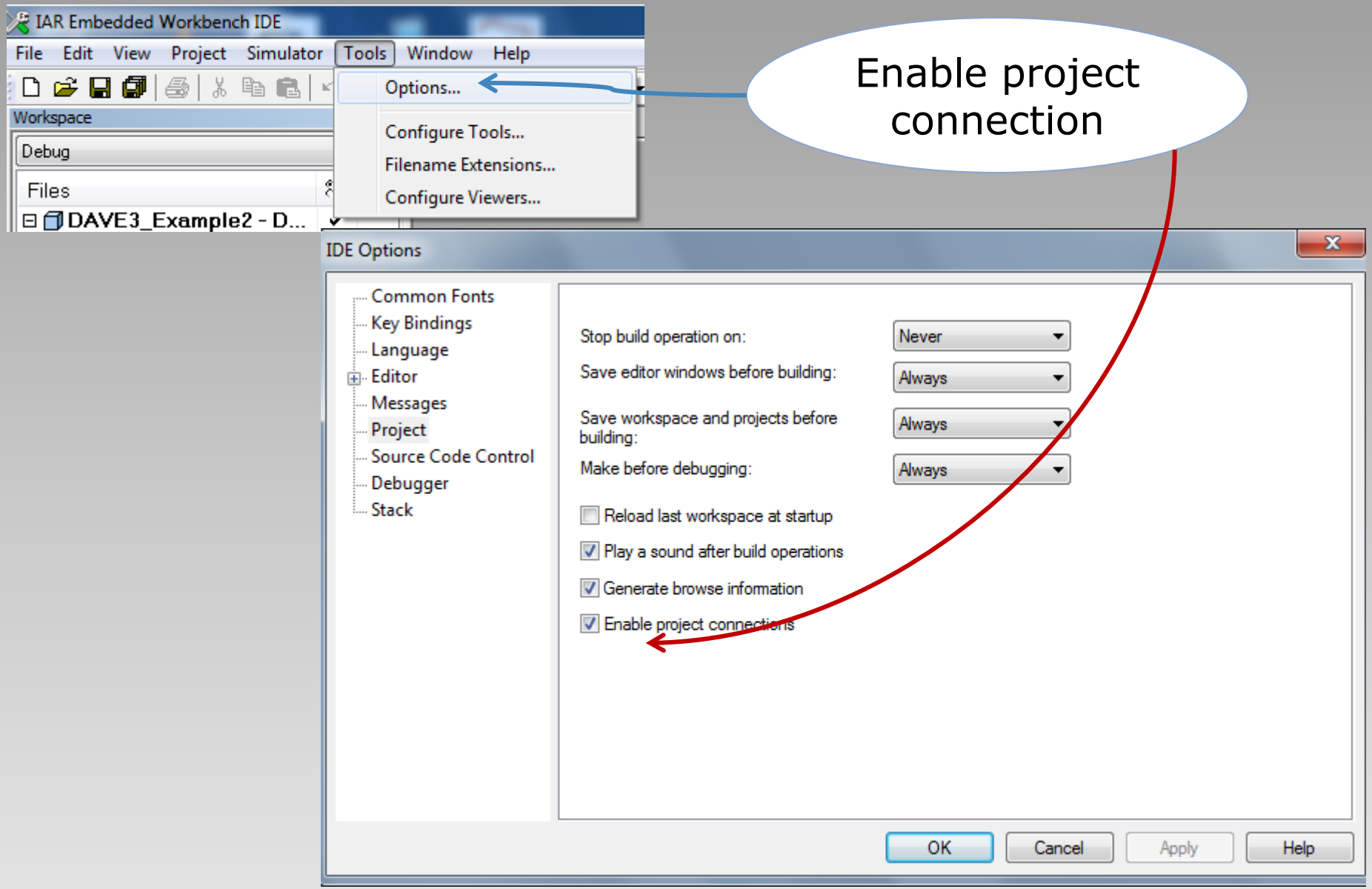
Step 1: Save Template Project

- Choose a **destination** folder of your choice for a copy of the template project
- The default folder is MyDocuments\IAR EmbeddedWorkbench\arm\7.40.2\Infineon\XMC1xxx\XMC1200\IAR_DAVE4

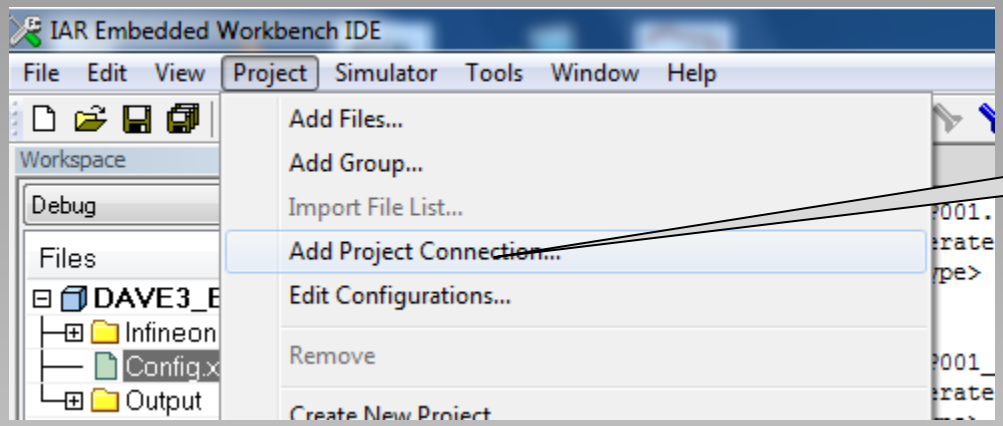


- **Note**: If a DAVE example project is used it has to be imported in DAVE and code has to regenerated before template project is saved (update of project path in config.xml file)

Step 2: Enable Project Connection (if not enabled)

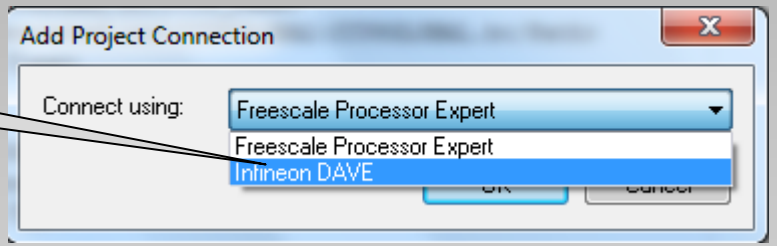


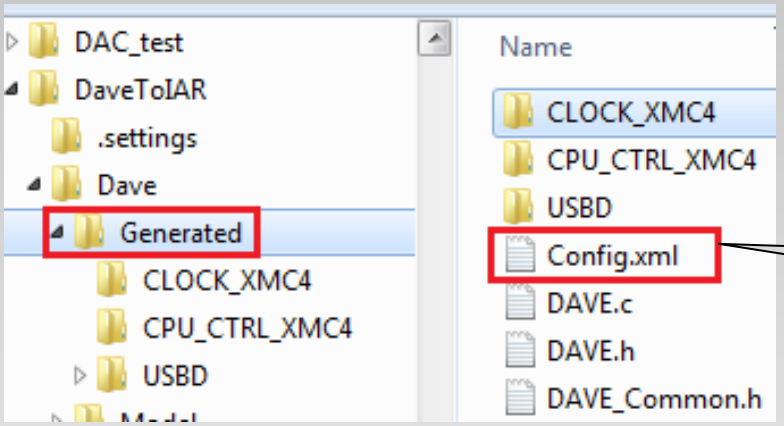
Step 3: Add Project Connection



Step 3.1: Add project connection

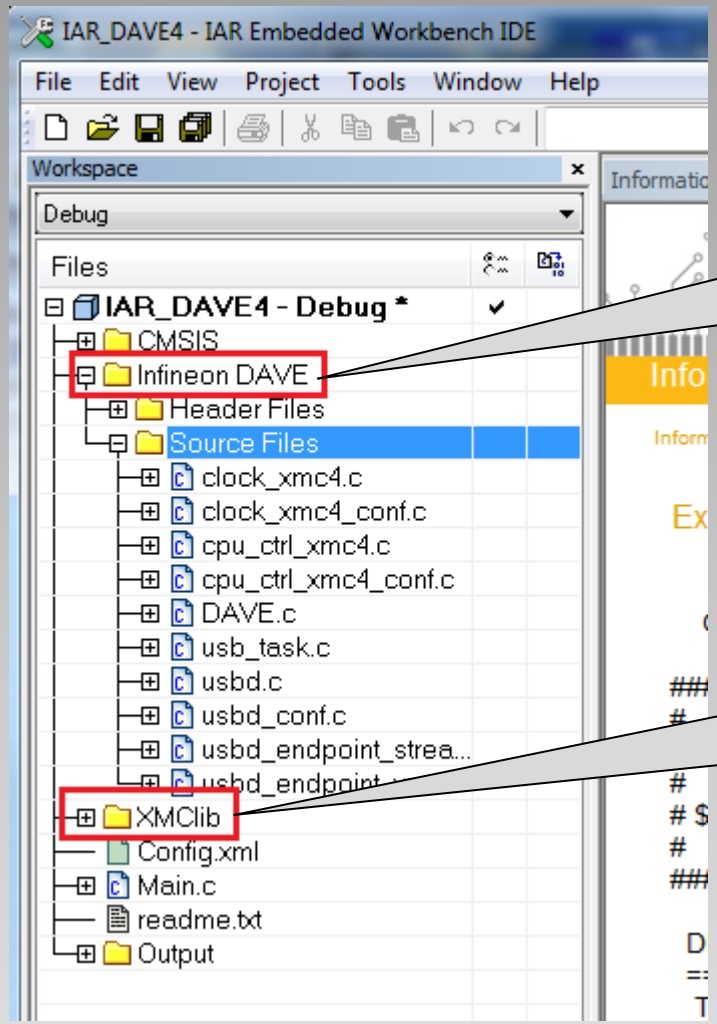
Step 3.2: Infineon DAVE





Step 3.3:
Go to ..\Dave\Generated\
Open Config.xml

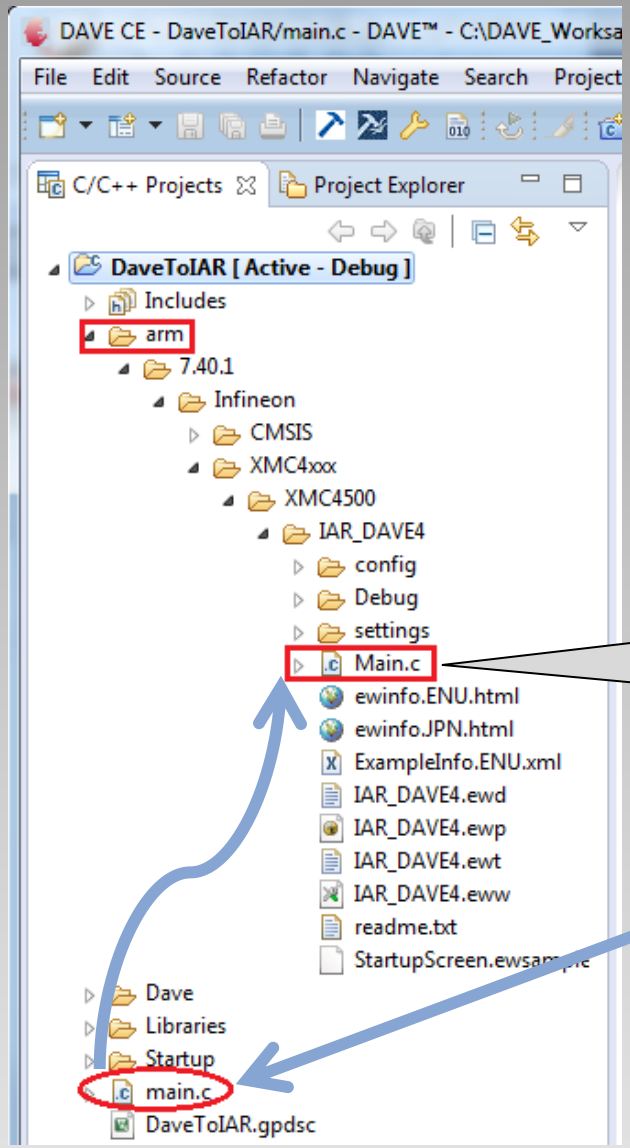
Step 4: Add Project Connection (cont'd)



After add project connection, all **DAVE4** source files and header files are imported into IAR project and displayed here.

From **DAVE4** the LLD library XMClib is an integral part in DAVE project. The XMClib source codes are already in Template project.

Step 5: Replace `main.c` in Template as needed

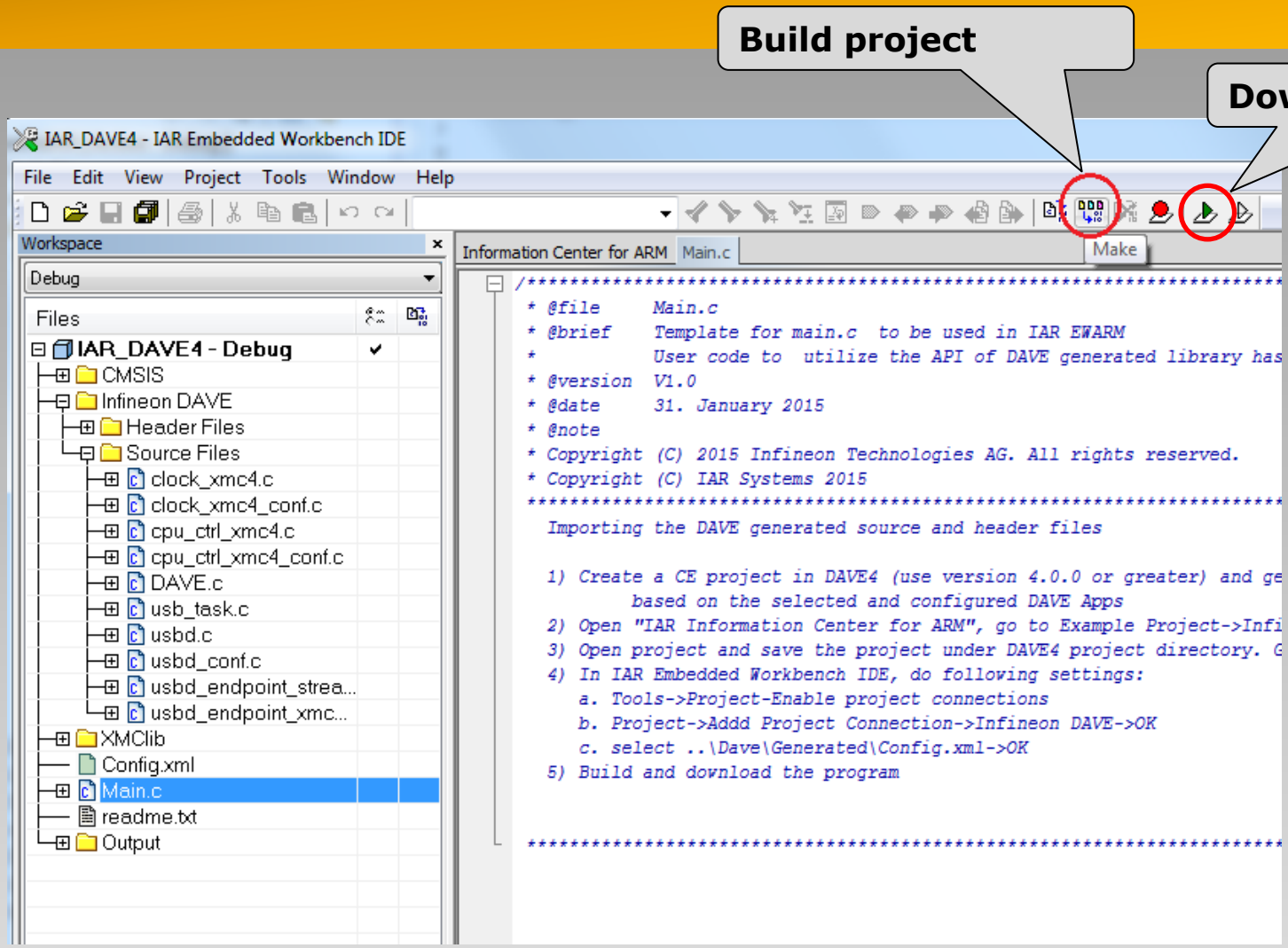


Note: DAVE Template project in IAR contains a *main.c* file for test purposes. **If the *main.c* file of the DAVE project should be used in IAR** , then ***main.c*** file should be copied to IAR project to replace the exiting *main.c* in the IAR Template project.

Step 5.1:

copy the **main.c** in the DAVE application project into IAR Template project to replace the *main.c* in the IAR Template project

Step 6: Build and download Project



- **Note**: the reason for warnings Pe050 is that the DAVE generated code terminates lines with LF only rather than CR/LF. These diagnostics can be ignored though.

Explore the IAR Embedded Workbench (1)

Apply to following change in main.c

```
void Adc_Measurement_Handler()
{
 uint32_t result;
 static uint32_t end_result;

 result = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_Potenziometer_handle);
 end_result=(result*Blink_LED.config_ptr->config_value>period_value)>>15;
 PWM_SetDutyCycle(&Blink_LED, end_result );
}
```

Explore the IAR Embedded Workbench (2)

- Stepping on C and Assembler level
- Code and log breakpoints
- Call Stack
- Live Watch
- Stack analysis and stack view
- Terminal I/O
- MISRA-C checker

Explore the IAR Embedded Workbench (3)

- Switch to the XMC4500 (Cortex-M3) board and open the example project for the Relax kit
- Add the following code in main.c:

```
typedef enum { OFF = 0, ON = 1} state_t ;
state_t led1_state = ON,LED1=ON;
state_t led2_state = ON,LED2=ON;
…
    if(led1_state == ON){
            /* Toggle LED1 */
            P1_1_toggle();
            LED1 =( LED1==ON) ? OFF : ON;
    }

    if(led2_state == ON){
            /* Toggle LED2 */
            P1_0_toggle();
            LED2 =( LED2==ON) ? OFF : ON;
    }
```

- Set a *data log breakpoint* for LED1 and LED2
- View data and interrupt log in the timeline

Explore the IAR Embedded Workbench (4)

- Add the following code in main.c:

```
#include "GPIO.h"
#include <arm_itm.h>
…
if(button1_state == ON){
            ITM_EVENT8_WITH_PC(1,1);
            if(led1_state == ON){
…
if(button2_state == ON){
            ITM_EVENT8_WITH_PC(2,1);
```

- View the event log in the timeline

Explore the IAR Embedded Workbench (5)

- Add the following code in main.c:

```
void SysTick_Handler(void) {
    static uint32_t ticks = 0UL;
    static state_t button1_state = OFF;
    static state_t button2_state = OFF;

    ticks++;
    printf("ticks = %d\n", ticks);
```

- Compare the semihosted vs. SWO based stdout