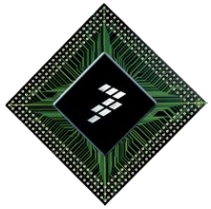
Last update: August 2013

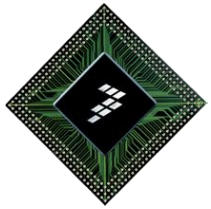


CW for Microcontrollers v10 and MQX™

Contents

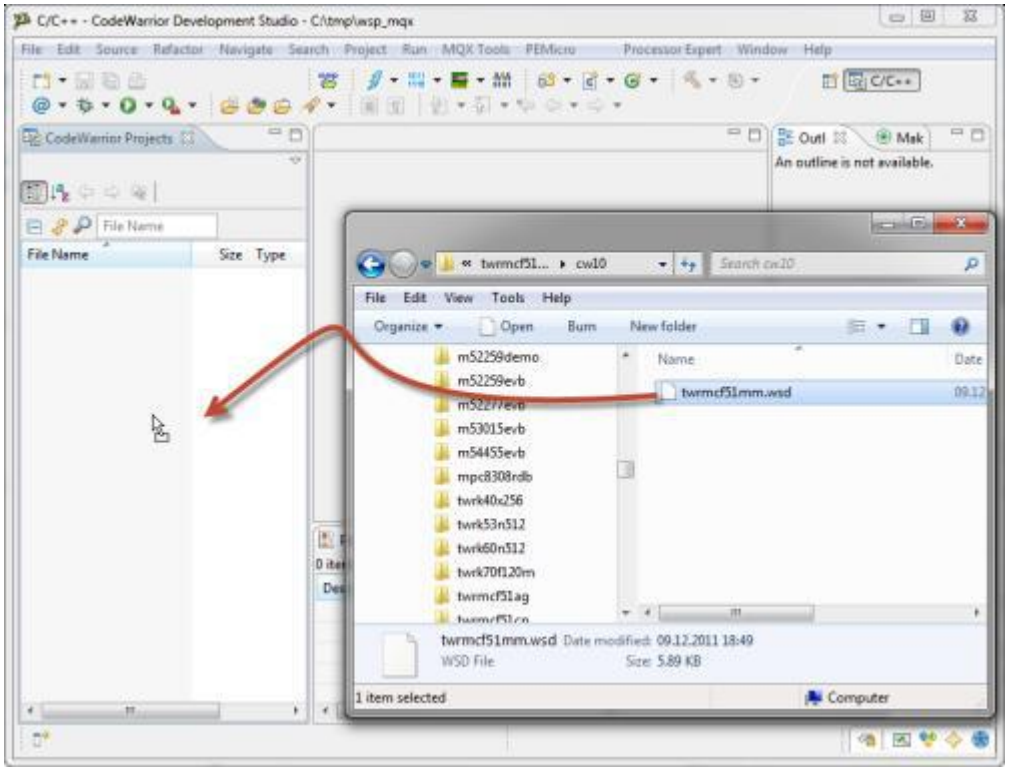
- Import MQX Libraries
- Build MQX libraries
- Import and Debug MXQ Hello World Project
- New MQX project
- Debugging with J-Link
- CW10.x, MQX and Processor Expert
- CW10.x, MQX and PE : New LDD driver

Import MQX Libraries

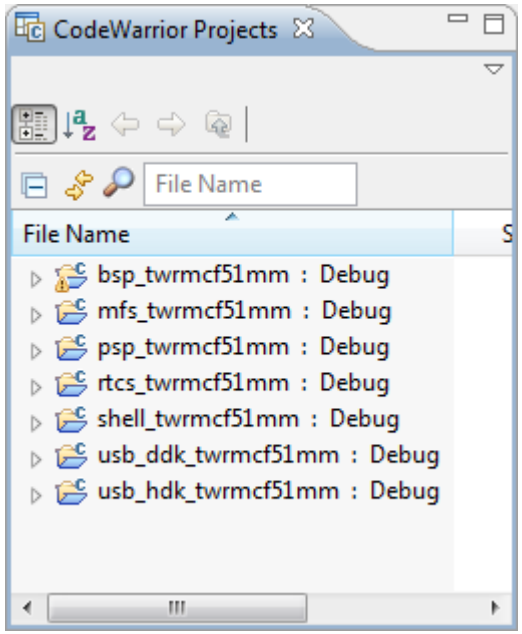


Import MQX Libraries

- Navigate to `C:\Freescale\Freescale MQX X.X\config\<board_name>` and drag <board>.wsd to the CodeWarrior

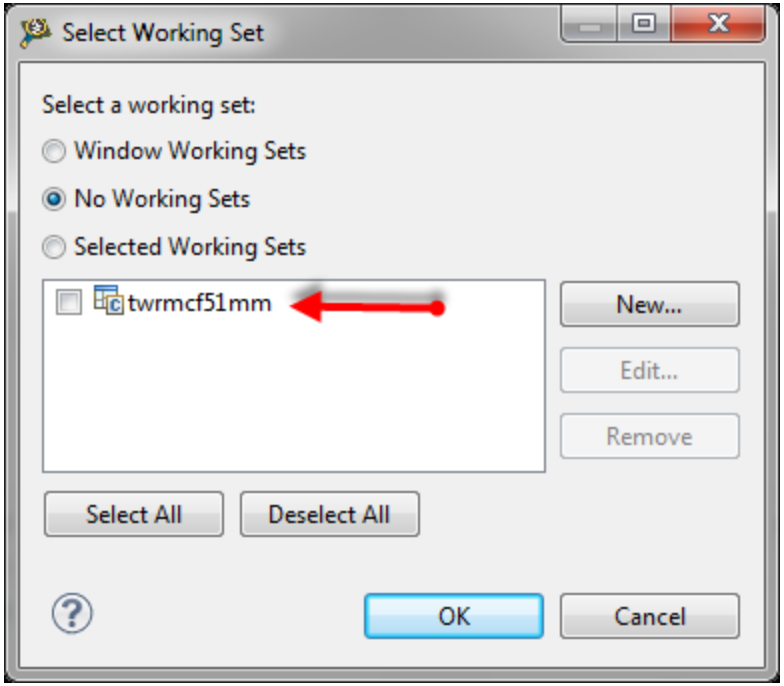


- All BSP libraries will be loaded to your environment automatically



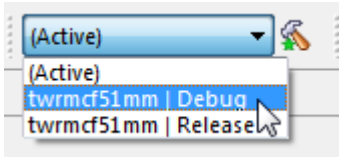
Import MQX libraries

- Both, the projects, and the Working Set configuration have been imported.

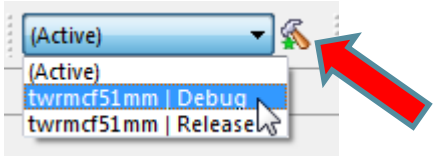


Building MQX Libraries

- Use MQX toolbar to select desired configuration you wish to build.

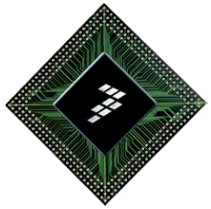


- Hit the icon to build all MQX libraries for a selected working set as shown below:



Note:
Debug configuration of MQX libraries, workingset, has the compiler optimization set to the lowest level for all imported projects. The **Release** configuration uses the highest possible compiler optimization setting.

Import and Debug MXQ Hello World Project

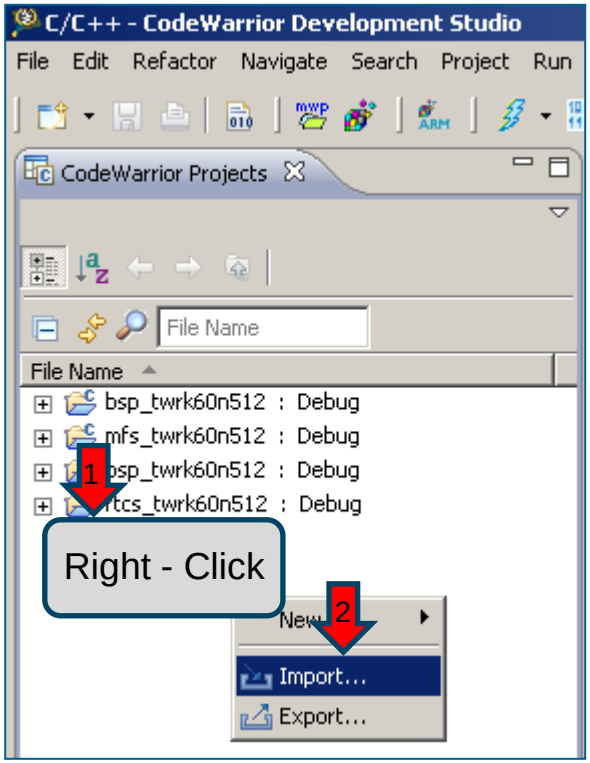


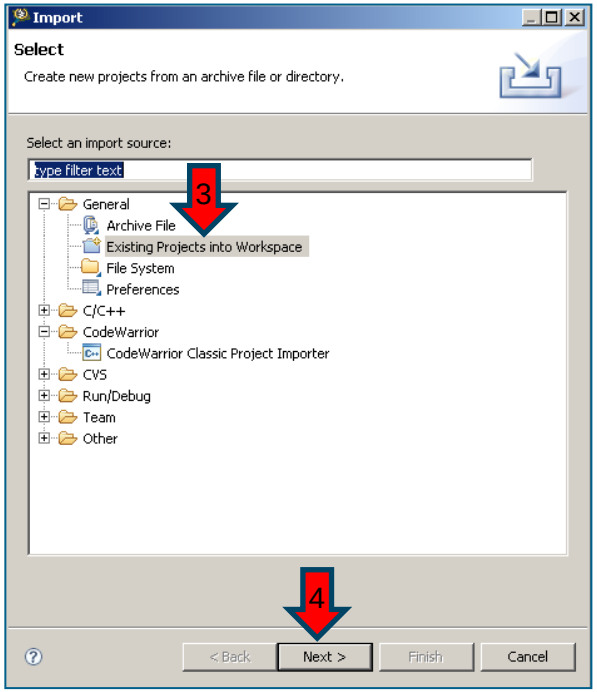
ARM POWERED®

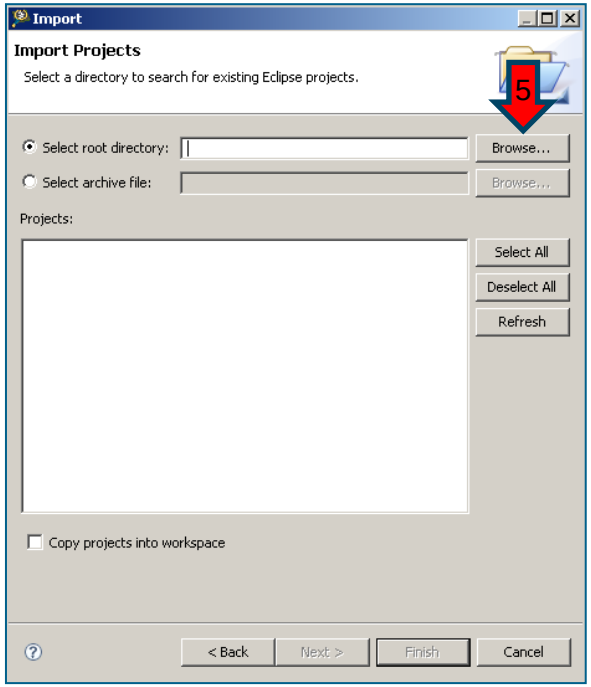
freescale™ semiconductor

Import ‘Hello World’ MQX example

- Right-Click on Project Explorer and Import.
- Select Existing Projects into Workspace and Browse.

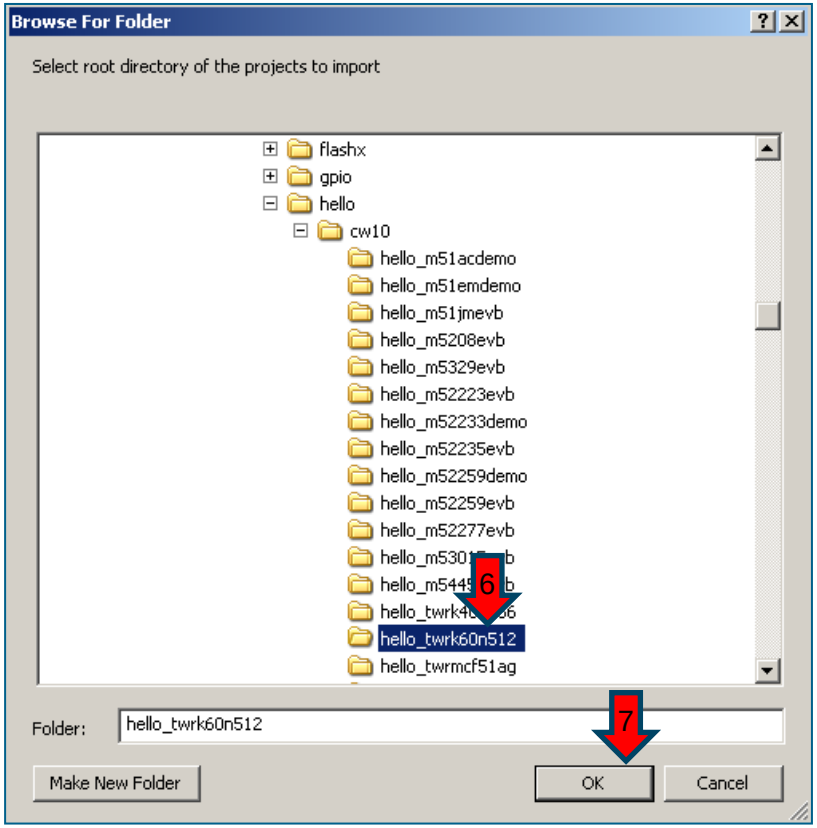


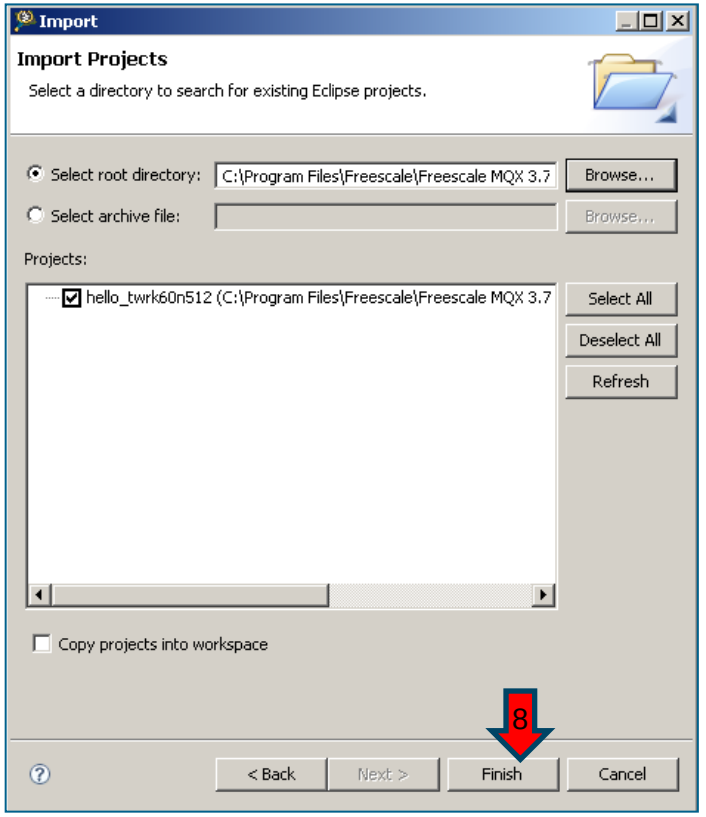




Import ‘Hello World’ MQX example

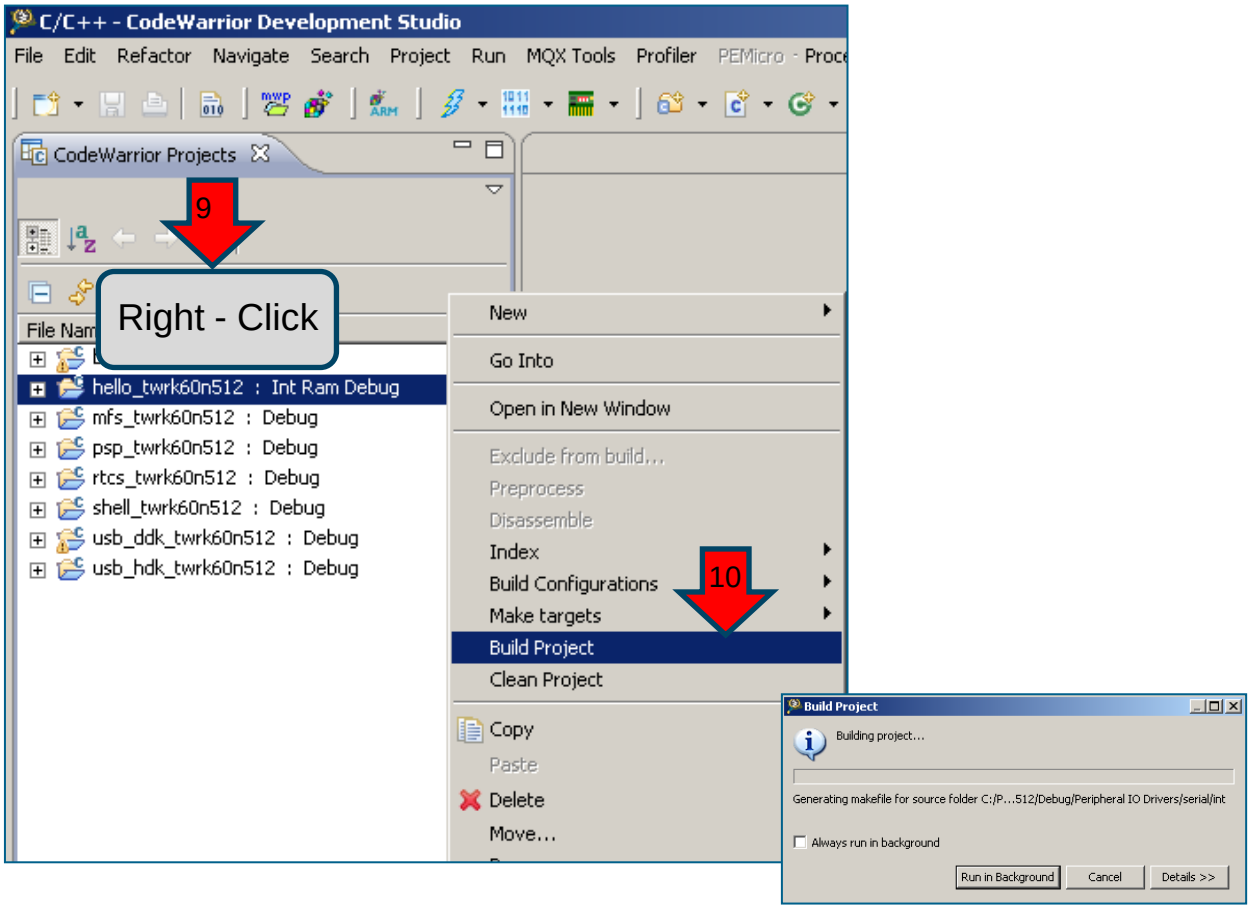
- Select *<install mqx folder>\mqx\examples\hello\CW10*\hello_twrk60n512





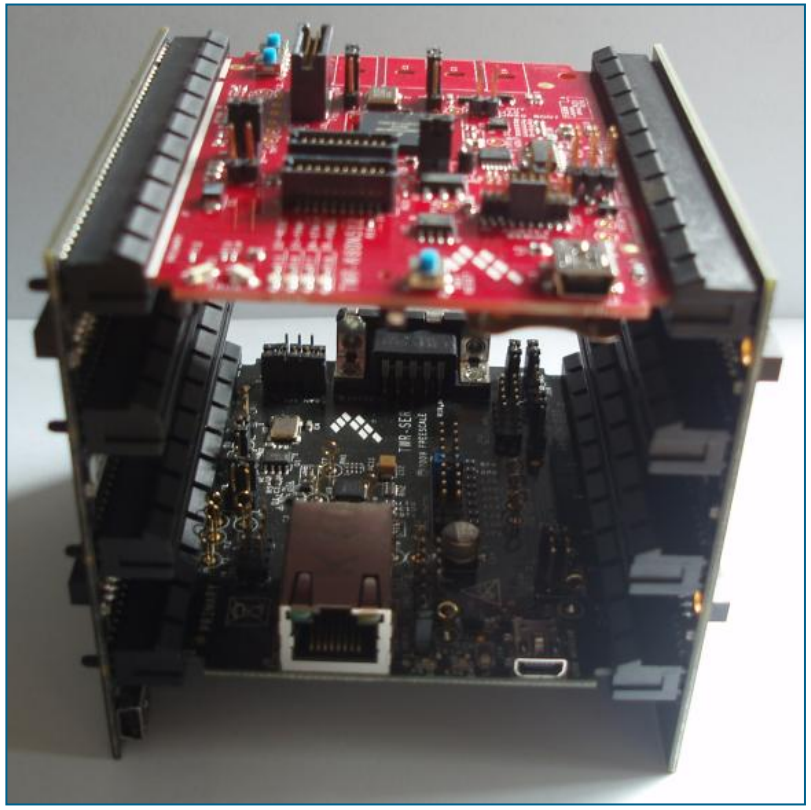
Build ‘Hello World’ MQX example

- Right-Click on Project Explorer **hello_twrk60n512** and select Build Project.



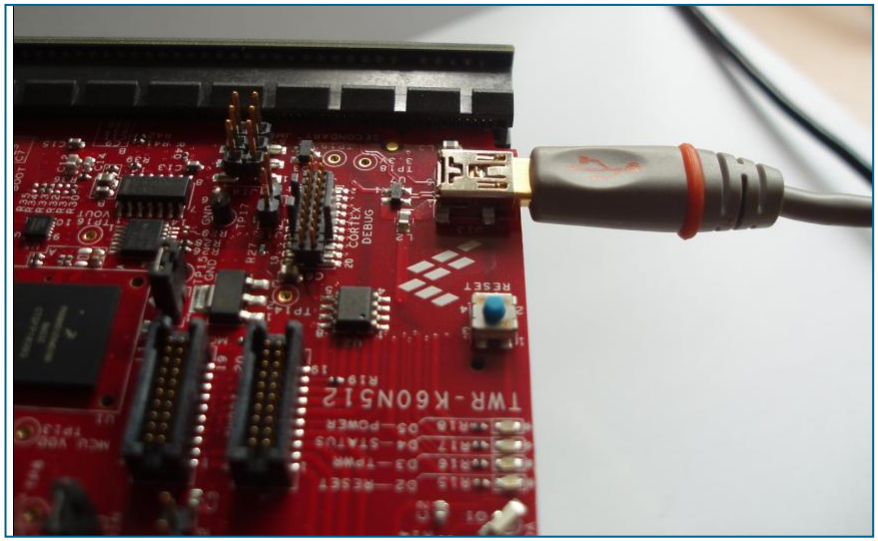
Prepare your hardware

- Prepare your Tower System:
  - Connect **TWR-SER** and **TWR-K60N512** to **TWR-ELEV** (primary and secondary).



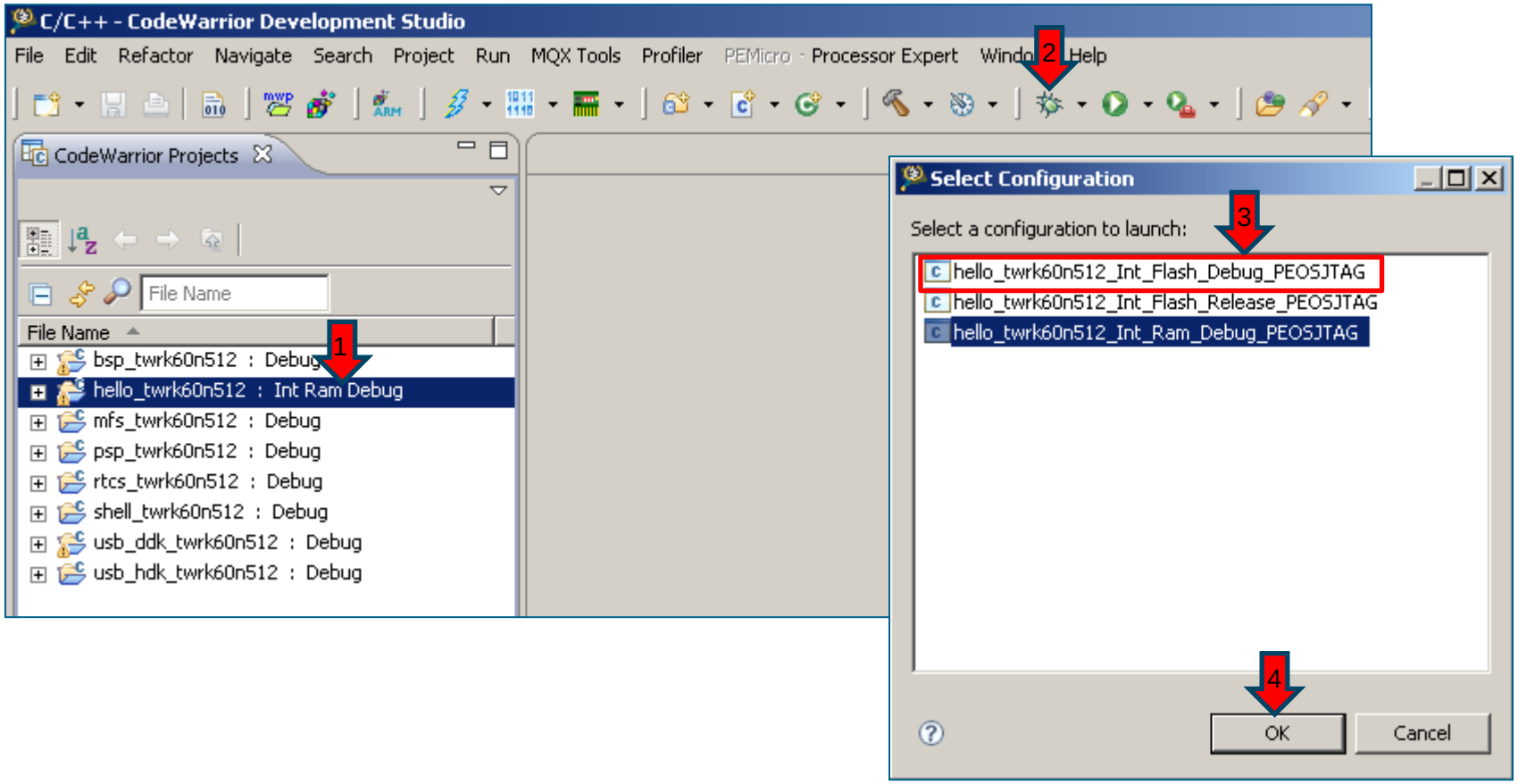
Prepare your hardware

- Connect USB Cable to the **TWR-K60N512** (J13) and to the laptop.



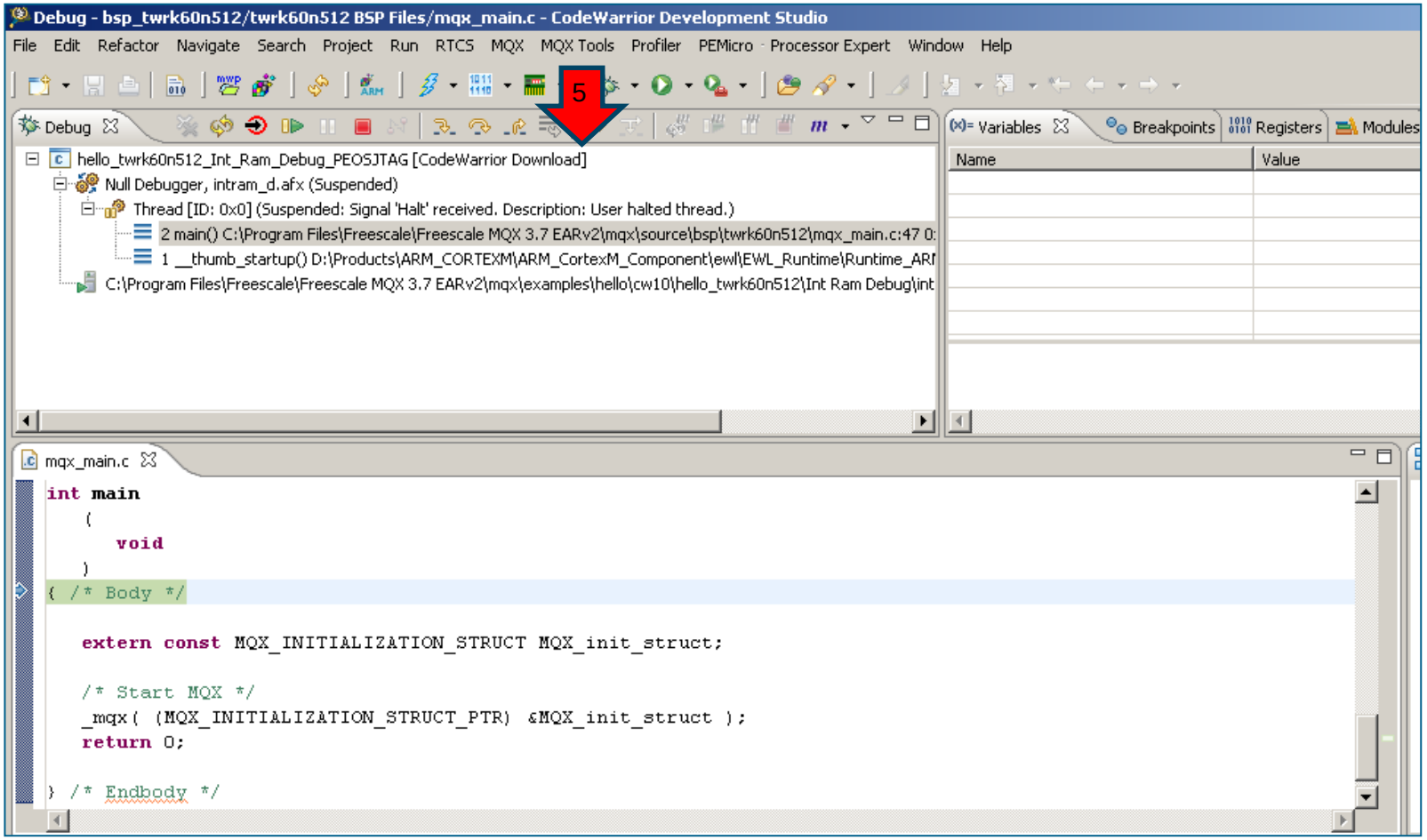
Debug MQX ‘Hello World’ example

- Select **hello_twrk60n512** project and Click ‘Debug icon.’
- Select **hello_twrk60n512_Int_Flash_Debug_PEOSJTAG** connection.



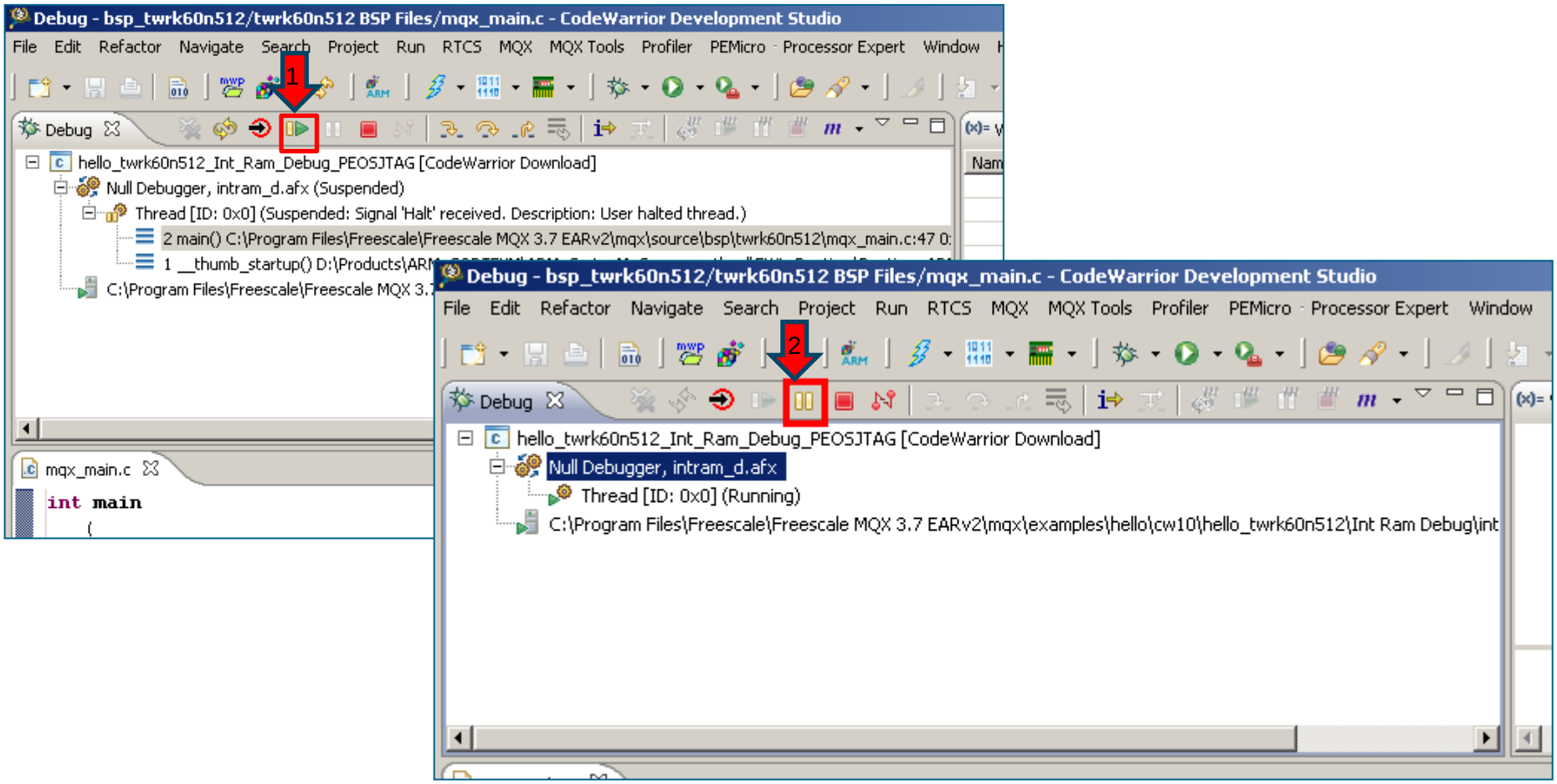
Debug MQX ‘Hello World’ example

- You are ready to Run and Debug the project.



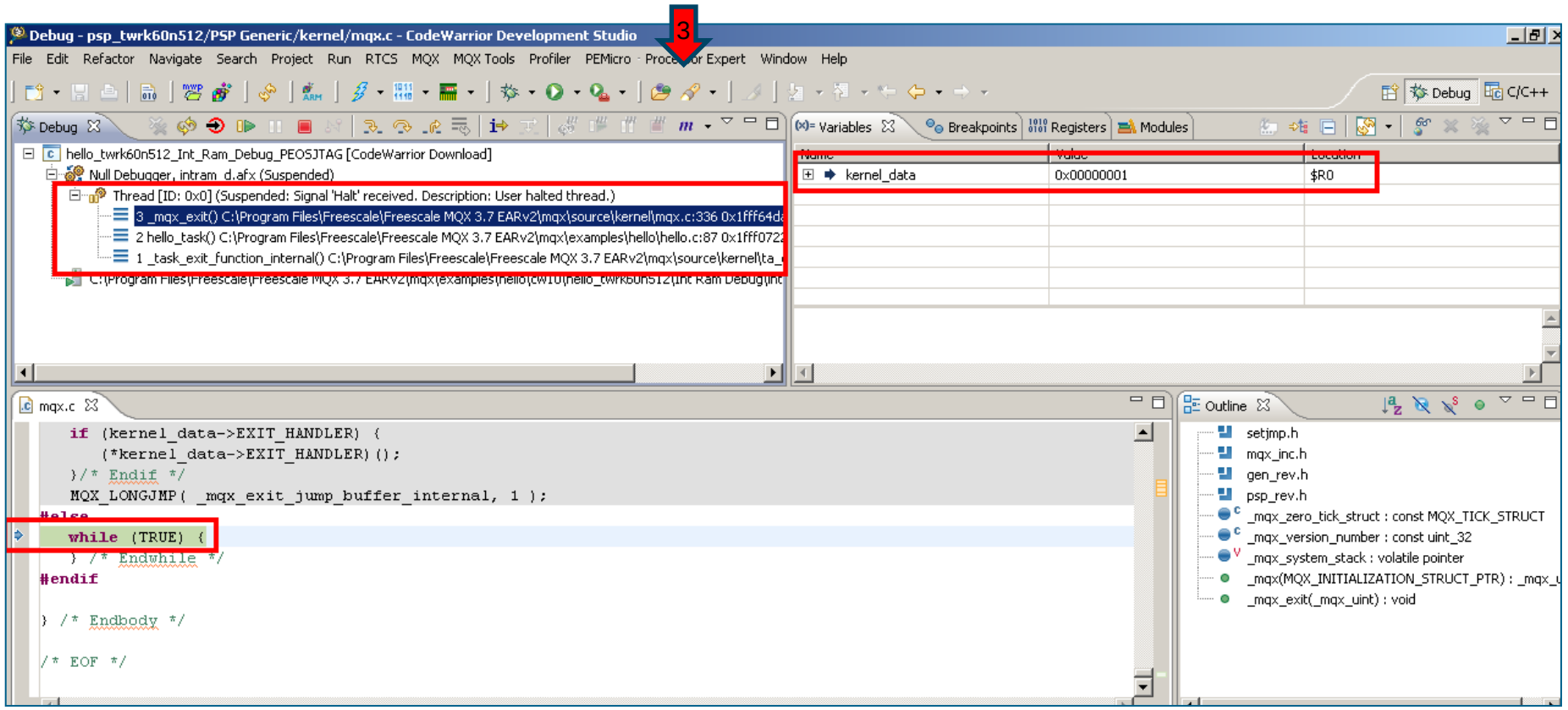
Run MQX ‘Hello World’ example

- Execute the code ‘Resume’ icon and ‘Pause’ execution.



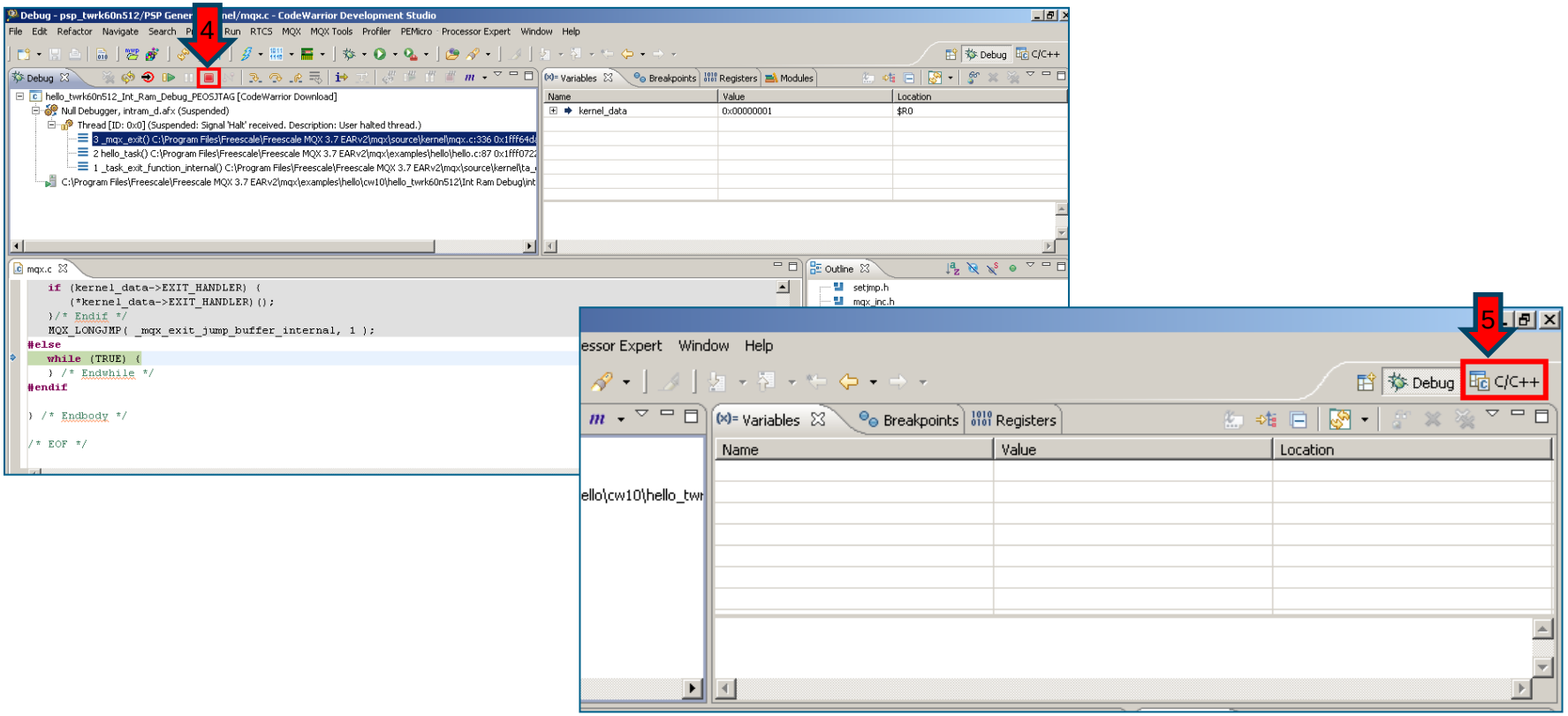
Run MQX ‘Hello World’ example

- You can explore the Debugging Eclipse perspective.

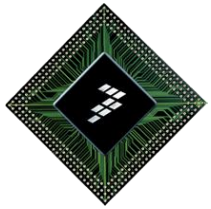


Run MQX ‘Hello World’ example

- Terminate the Debugging session and change Eclipse perspective.
- You have Run and Debug your first MQX CW10 project.

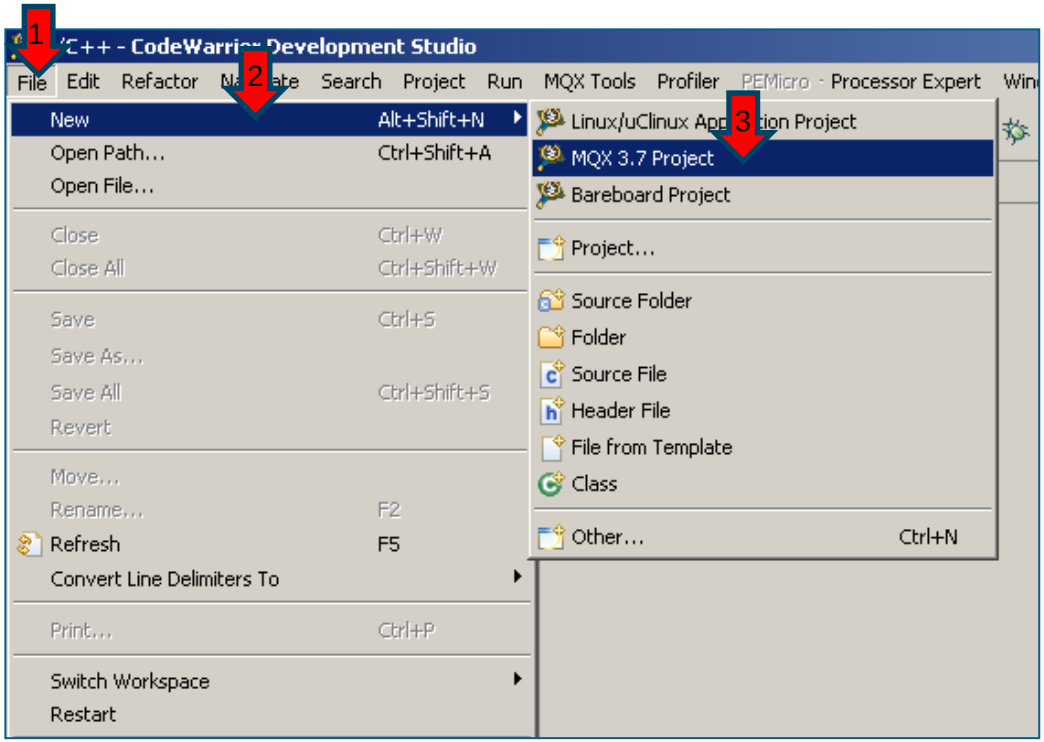


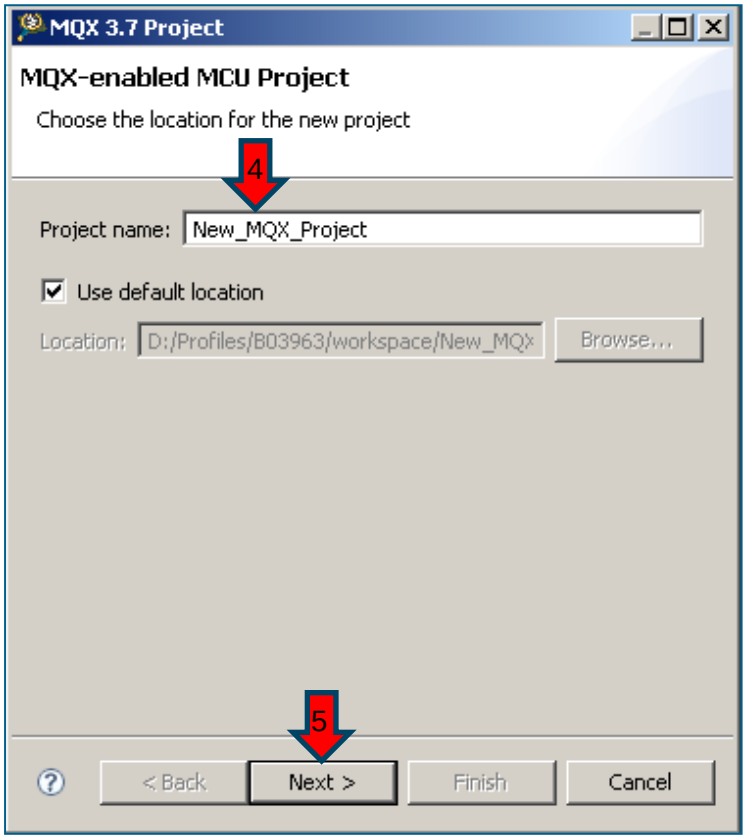
New MQX project



New MQX Project

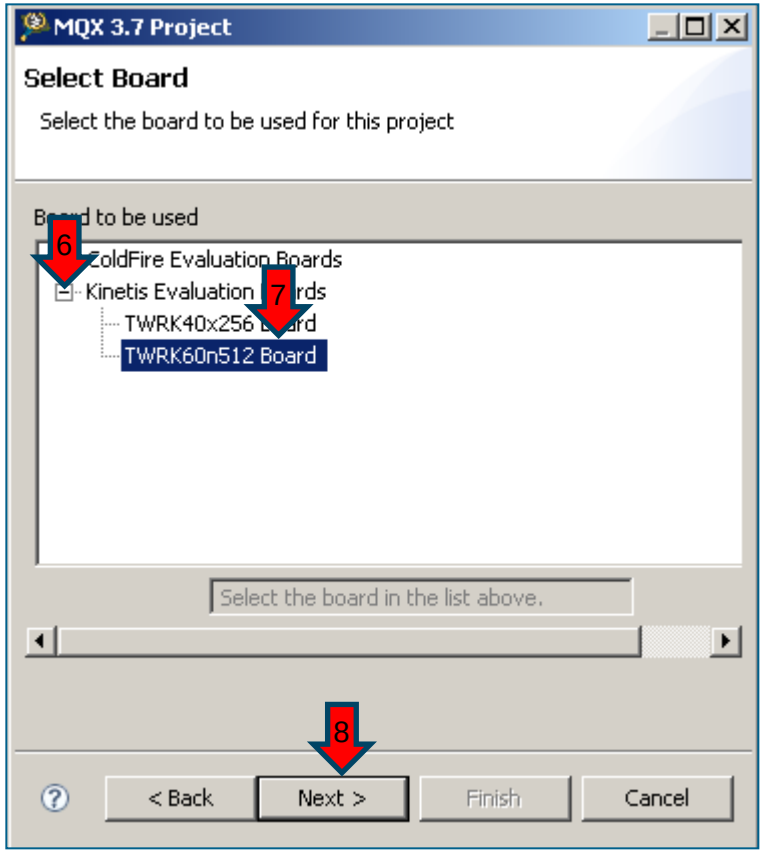
- File -> New -> MQX Project
- Give it a name and click Next.

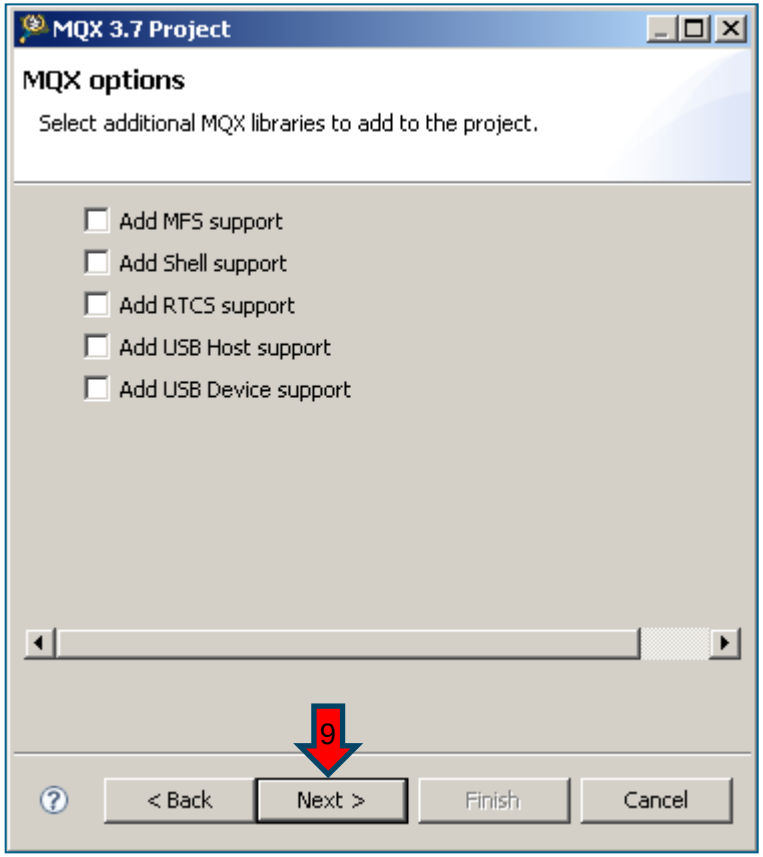




New MQX Project

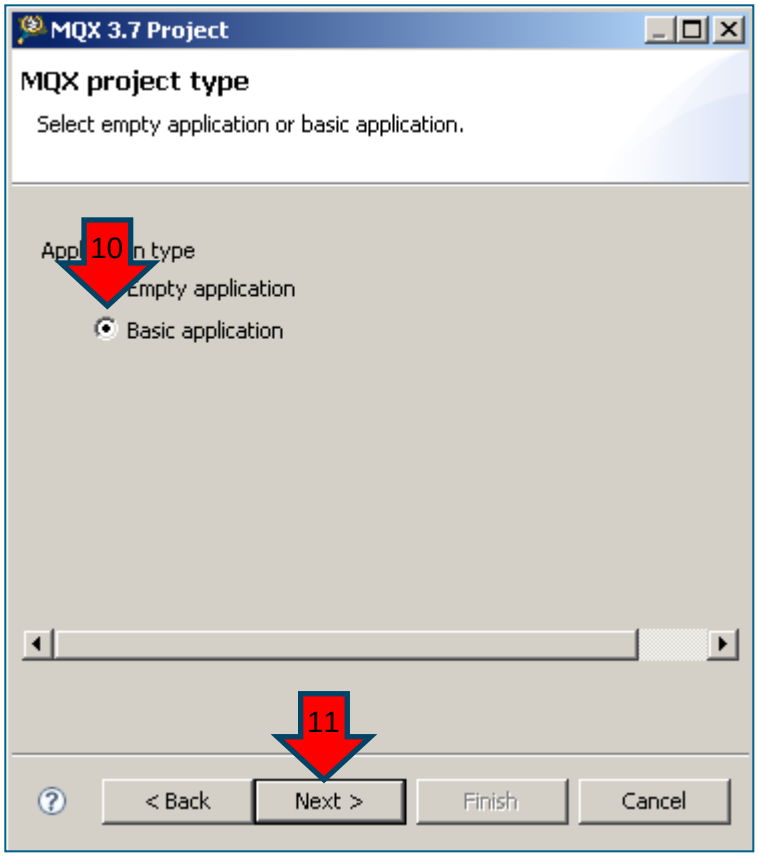
- Select **TWRK60n512** Board.

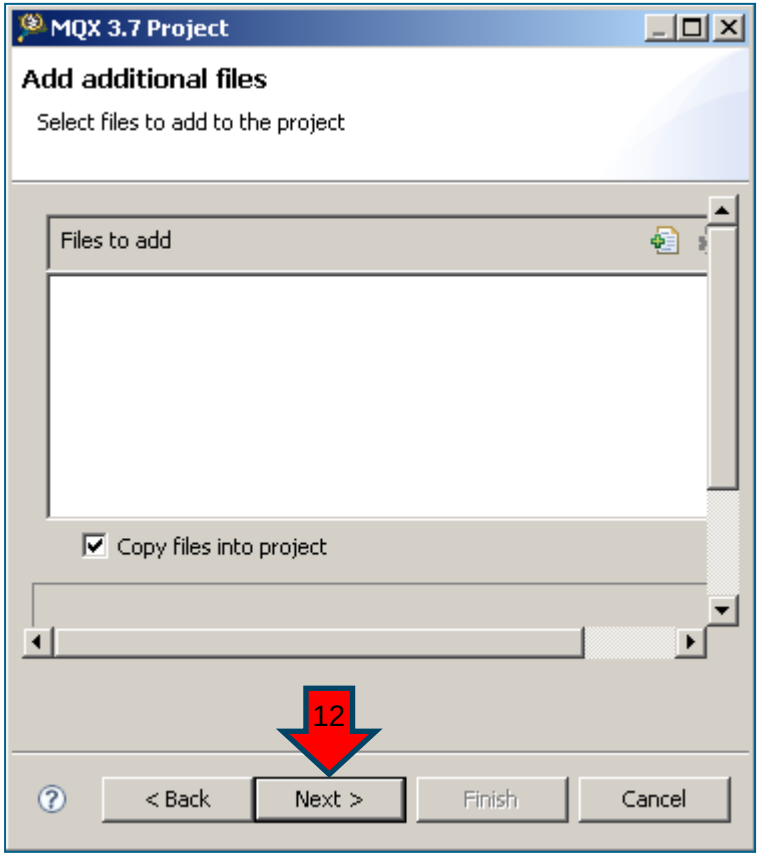




New MQX Project

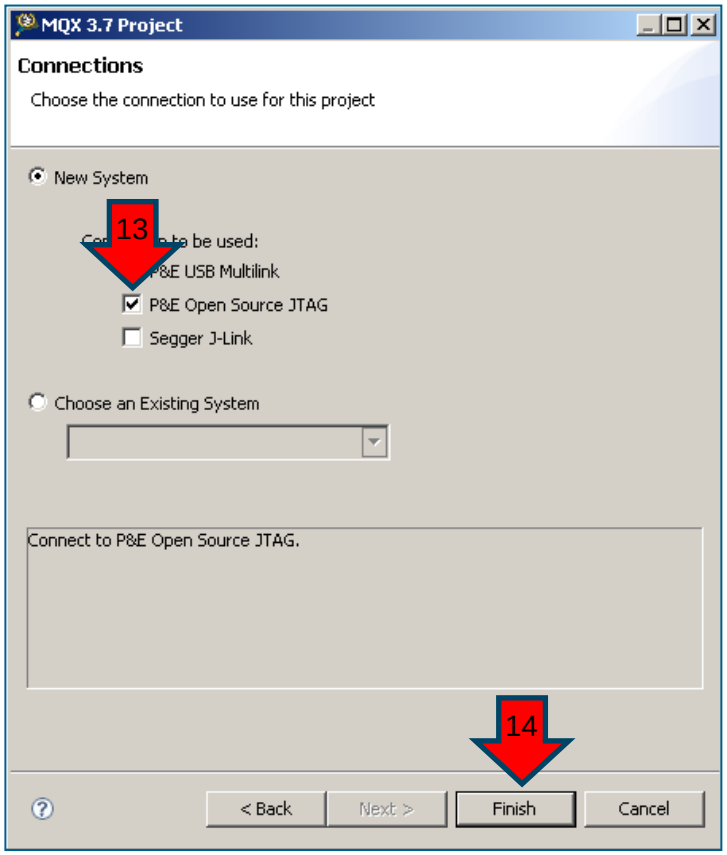
- Select Basic application.

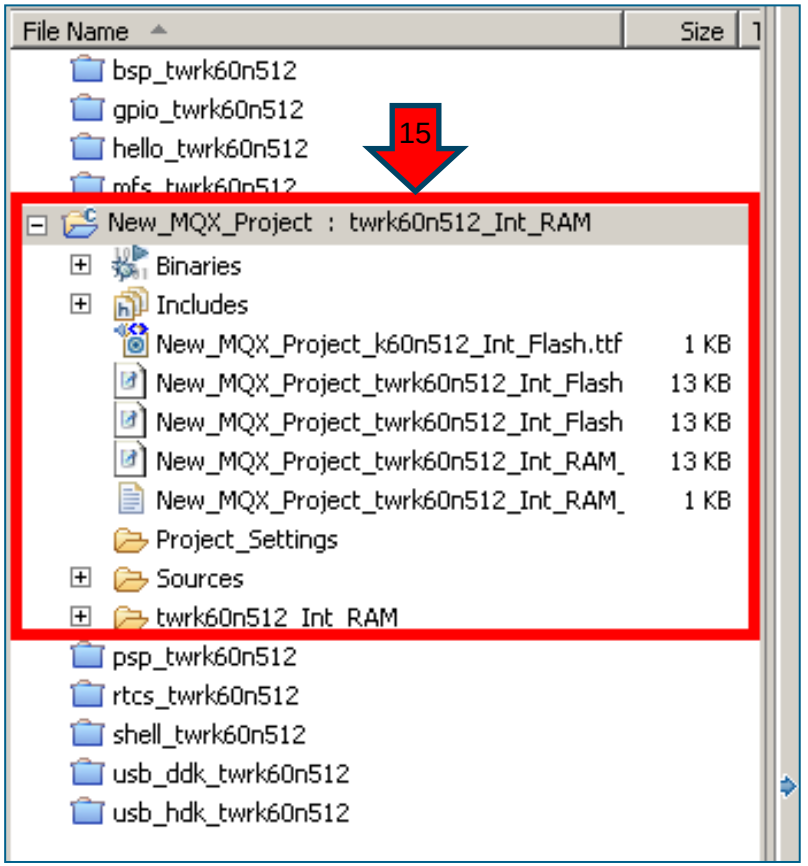




New MQX Project

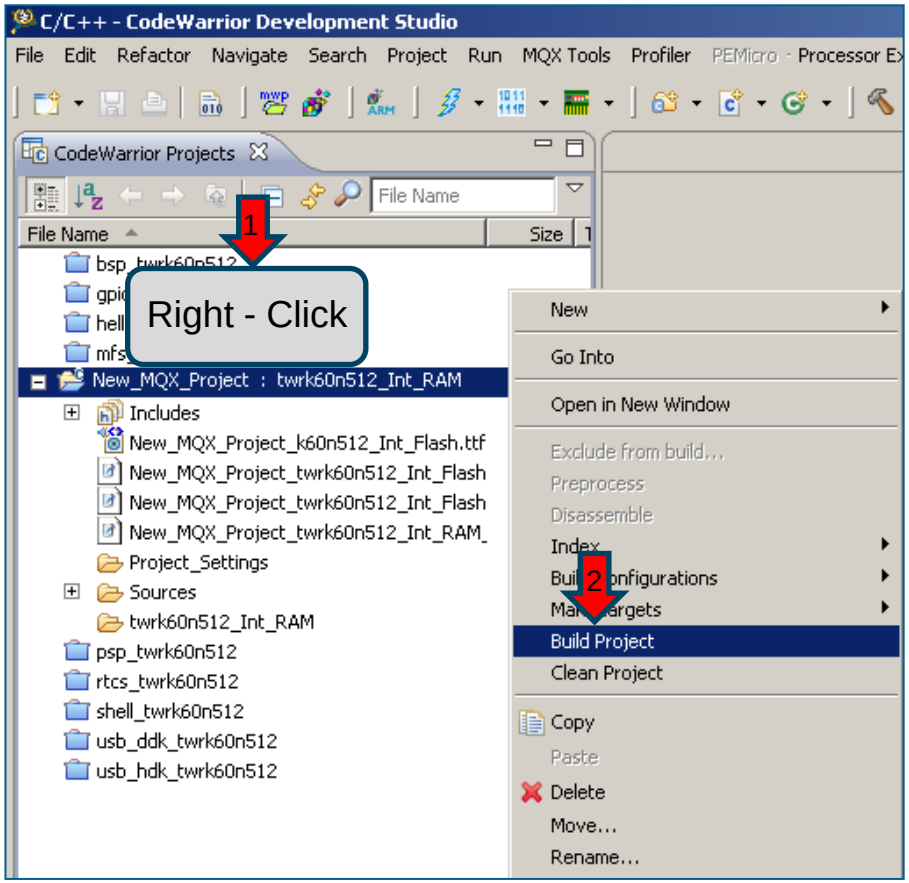
- Select P&E Open Source JTAG.
- A project is created.

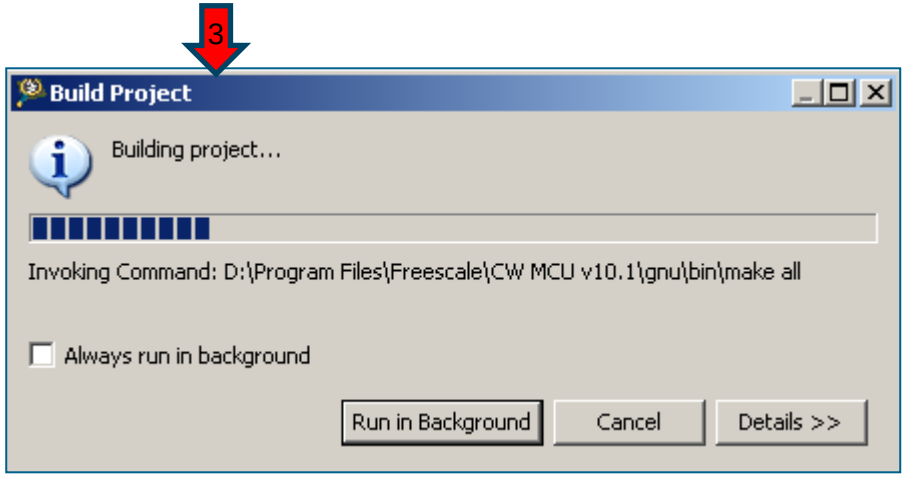




Build New MQX Project

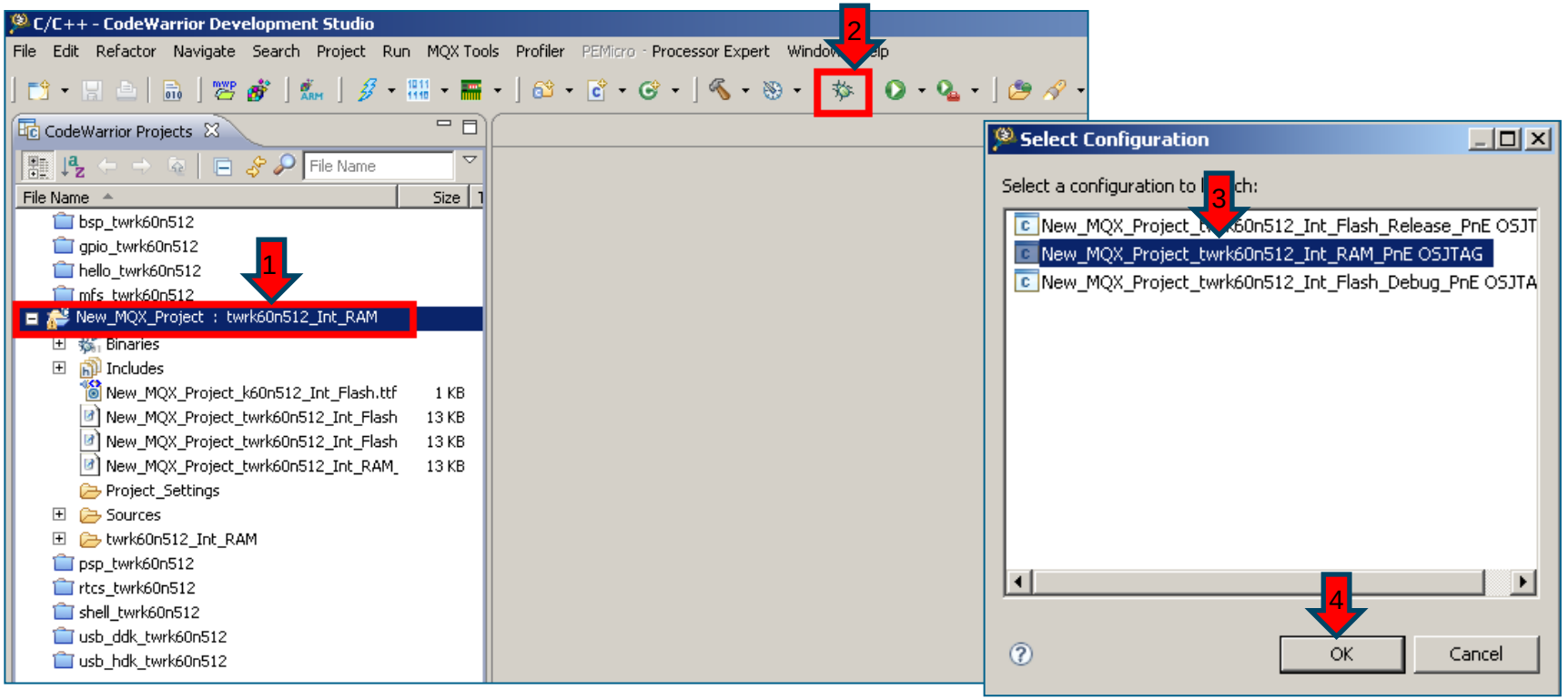
- Right-Click on Project Explorer **New_MQX_Project** and Build Project.





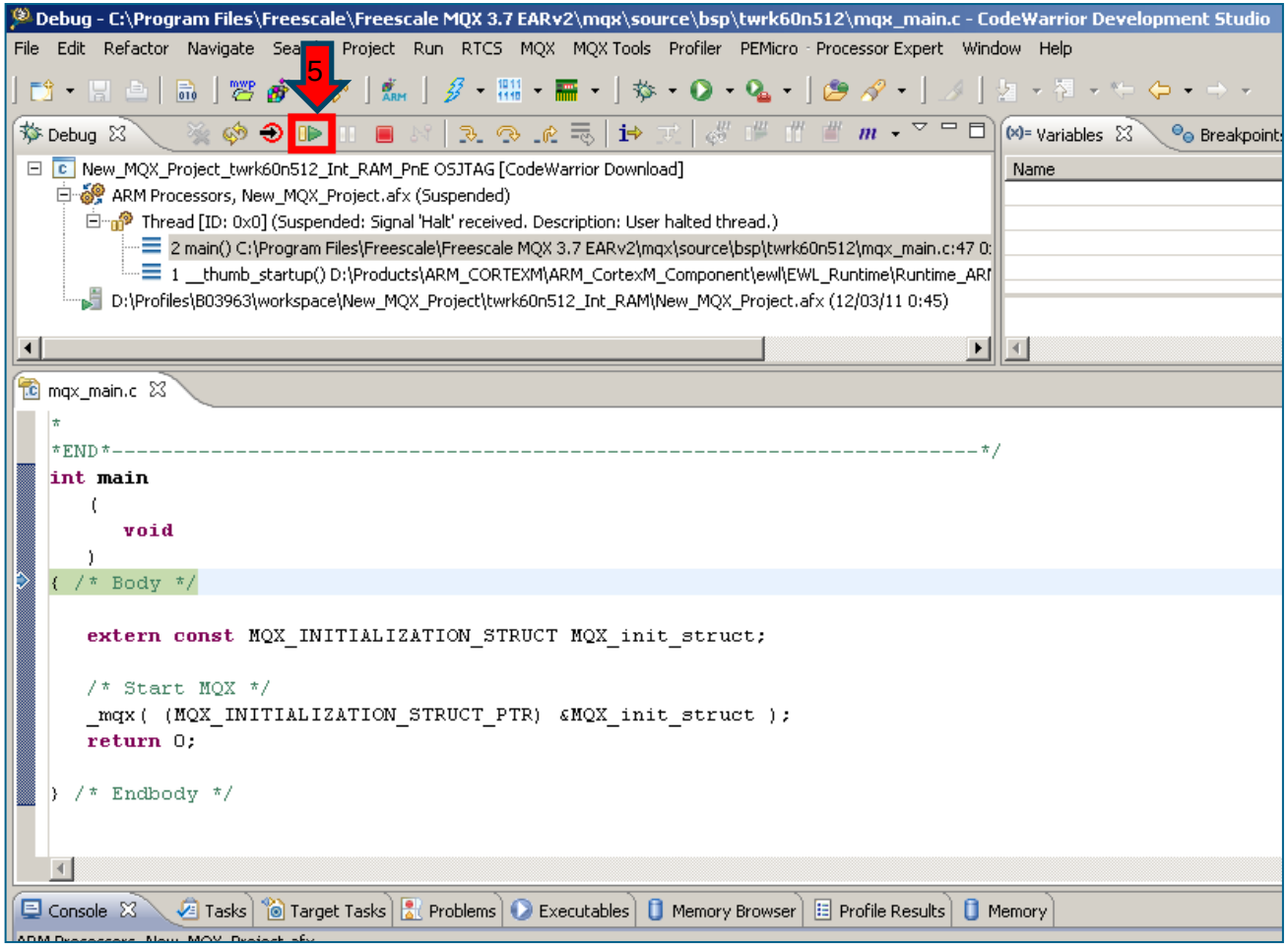
Debug New MQX Project

- Select **New_MQX_Project : twrk60n512_Int_RAM**
- Select **New_MQX_Project_twrk60n512_Int_Ram_PnE OSJTAG**



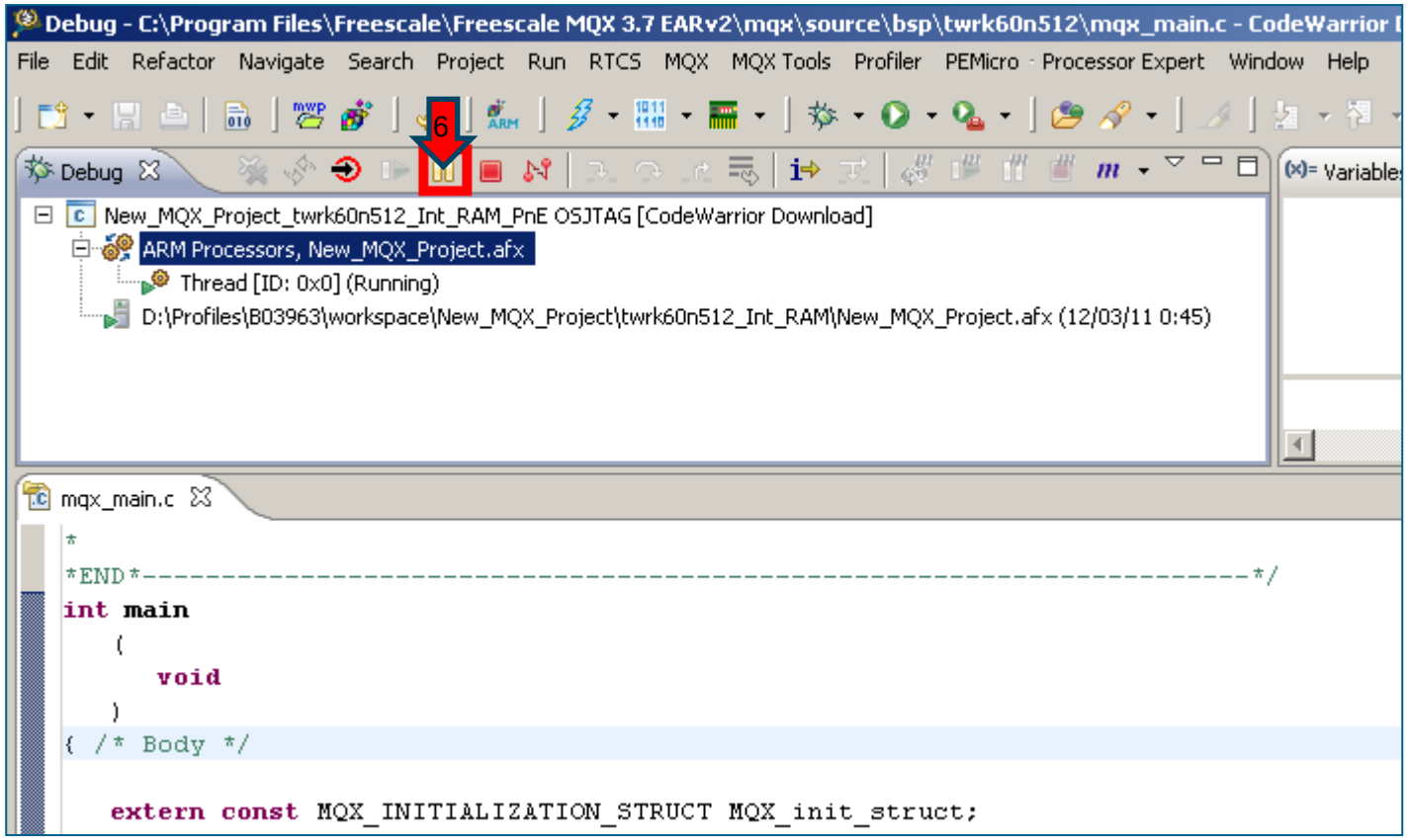
Run New MQX Project

- Execute the code '**Run**' icon.



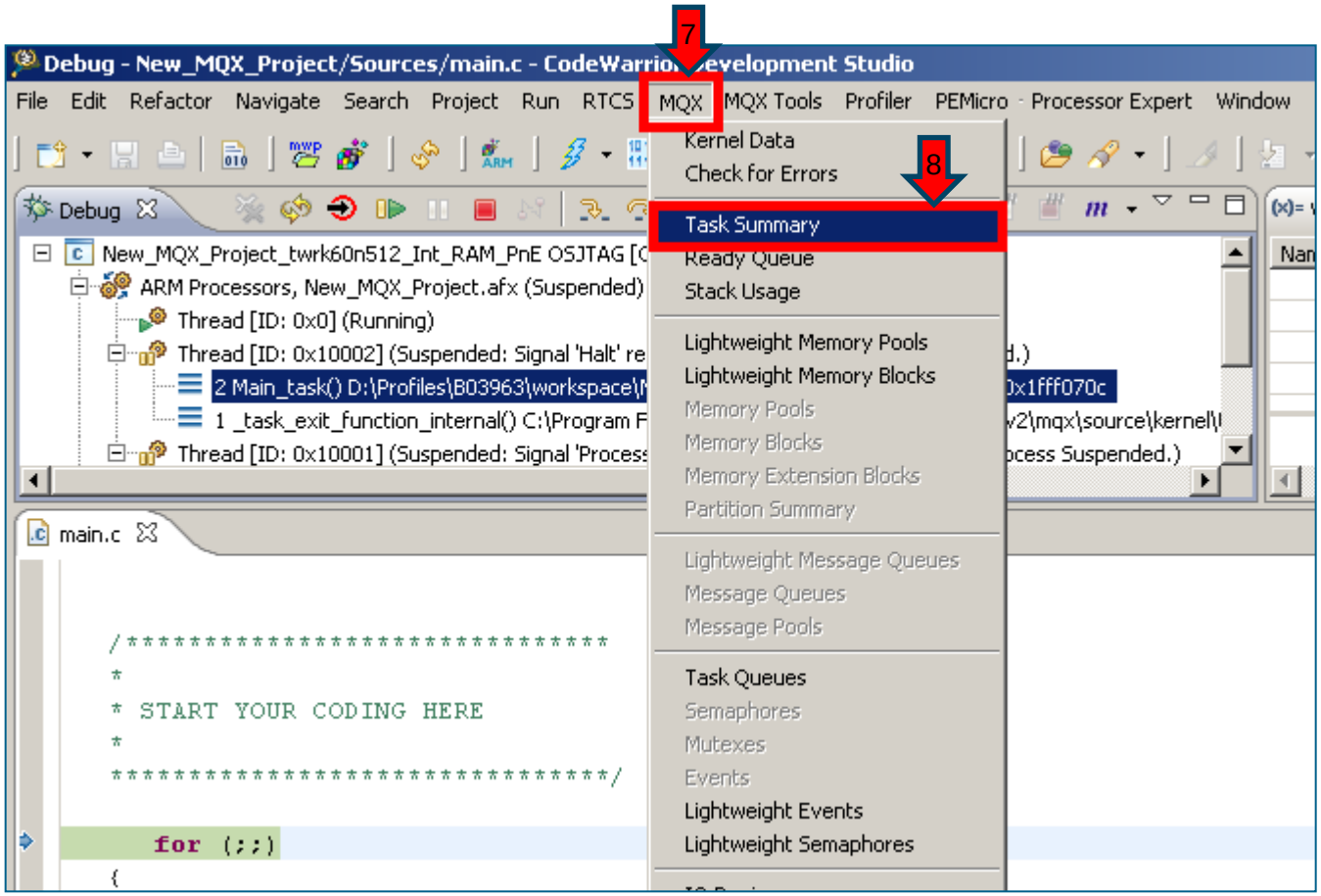
Run New MQX Project

- Pause execution.



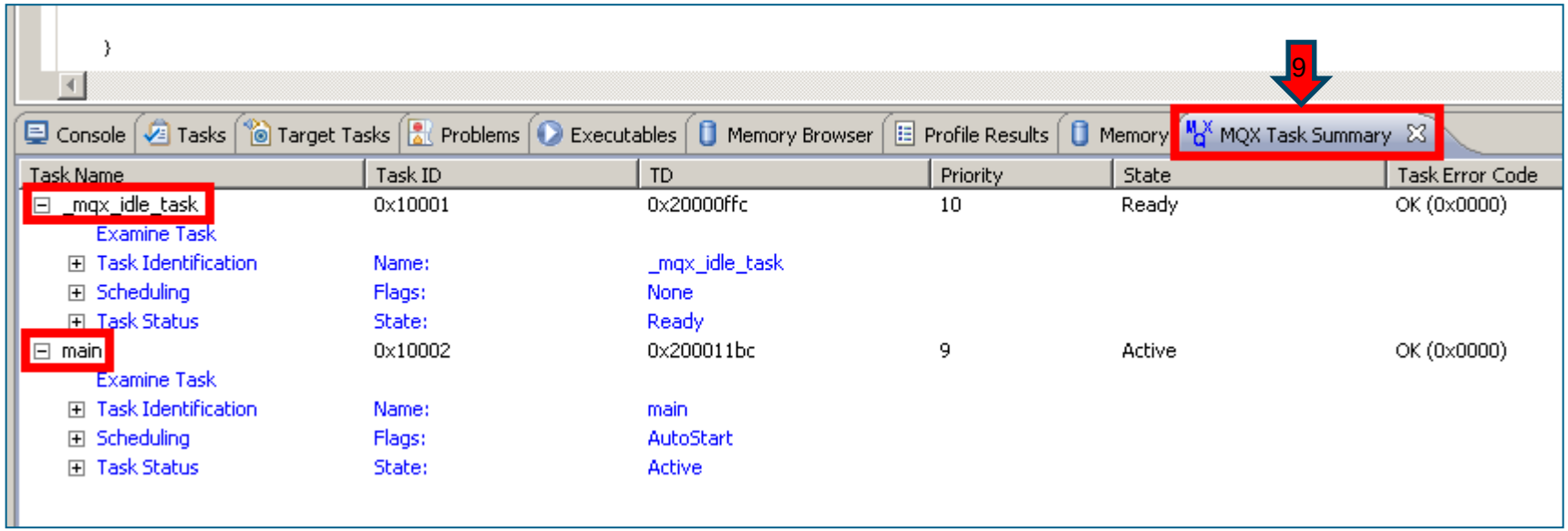
TAD: Task Summary

- MQX -> Task Summary



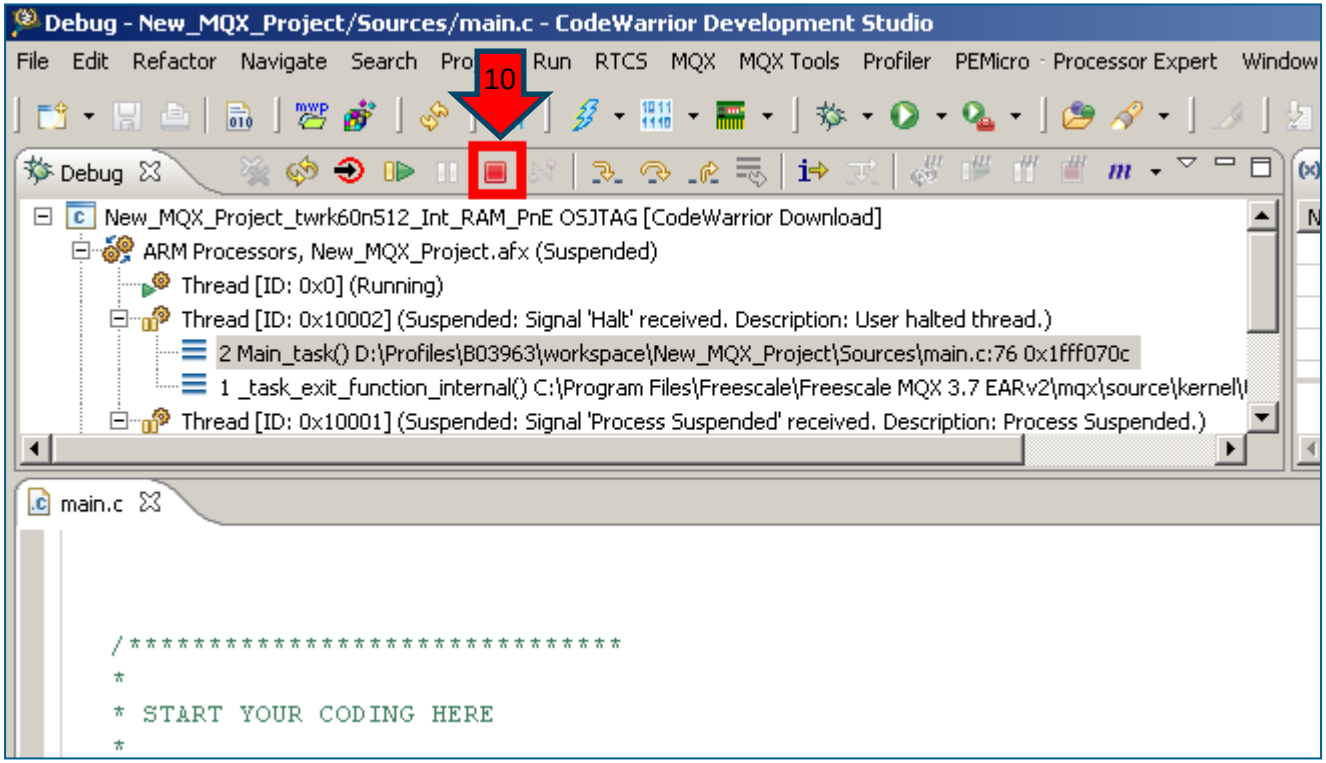
TAD: Task Summary

- Observe Tasks in your Application.

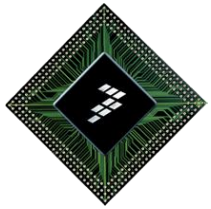


Run New MQX Project

- Terminate execution.

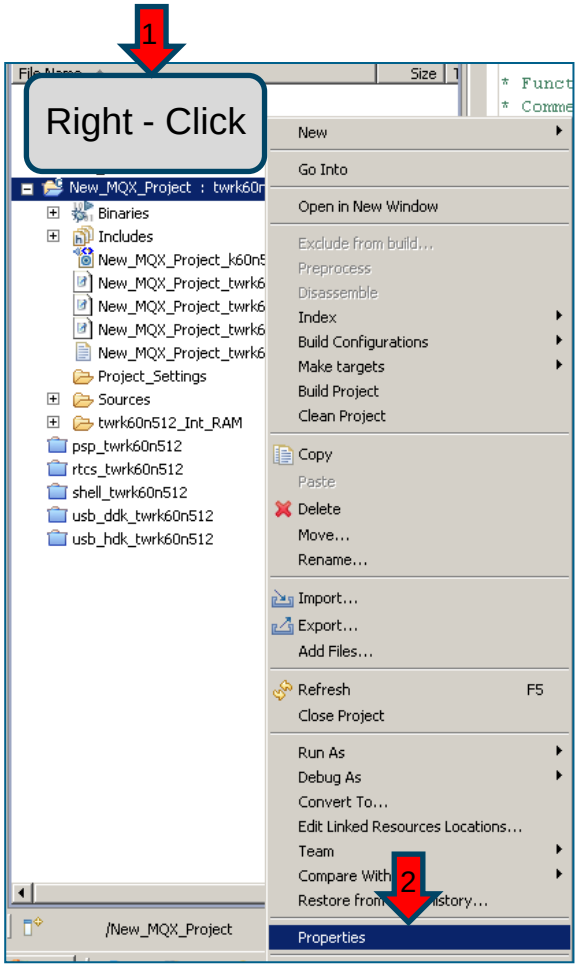


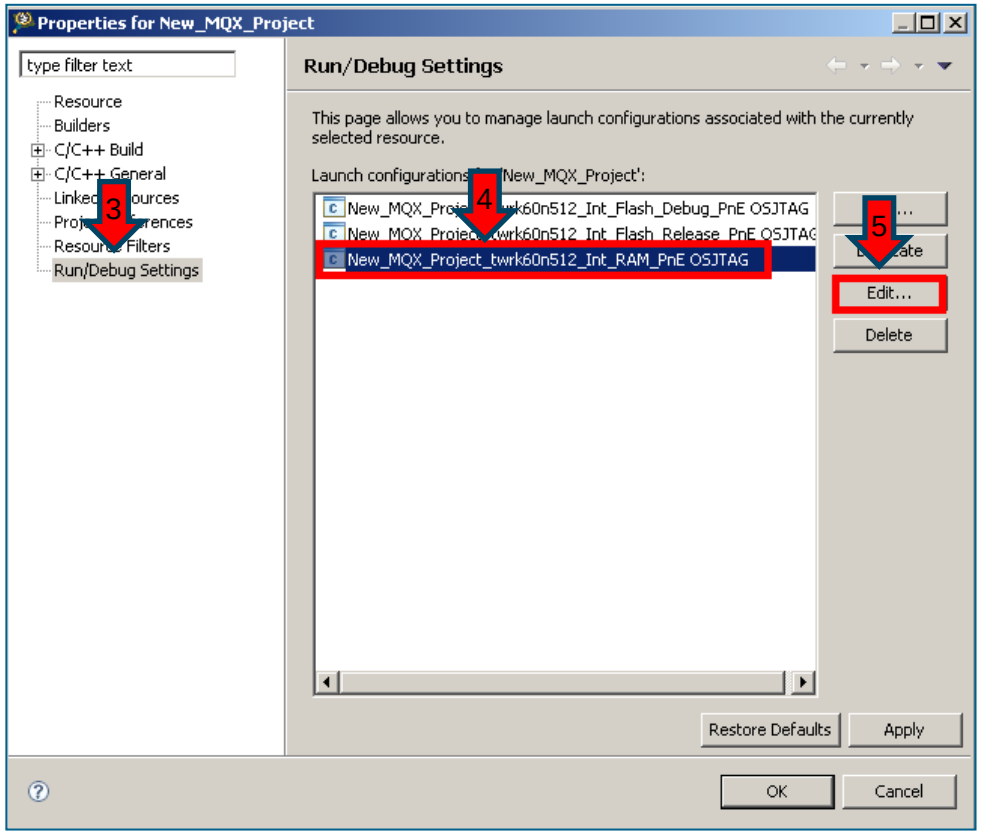
Debugging with J-Link



Change Connection Type

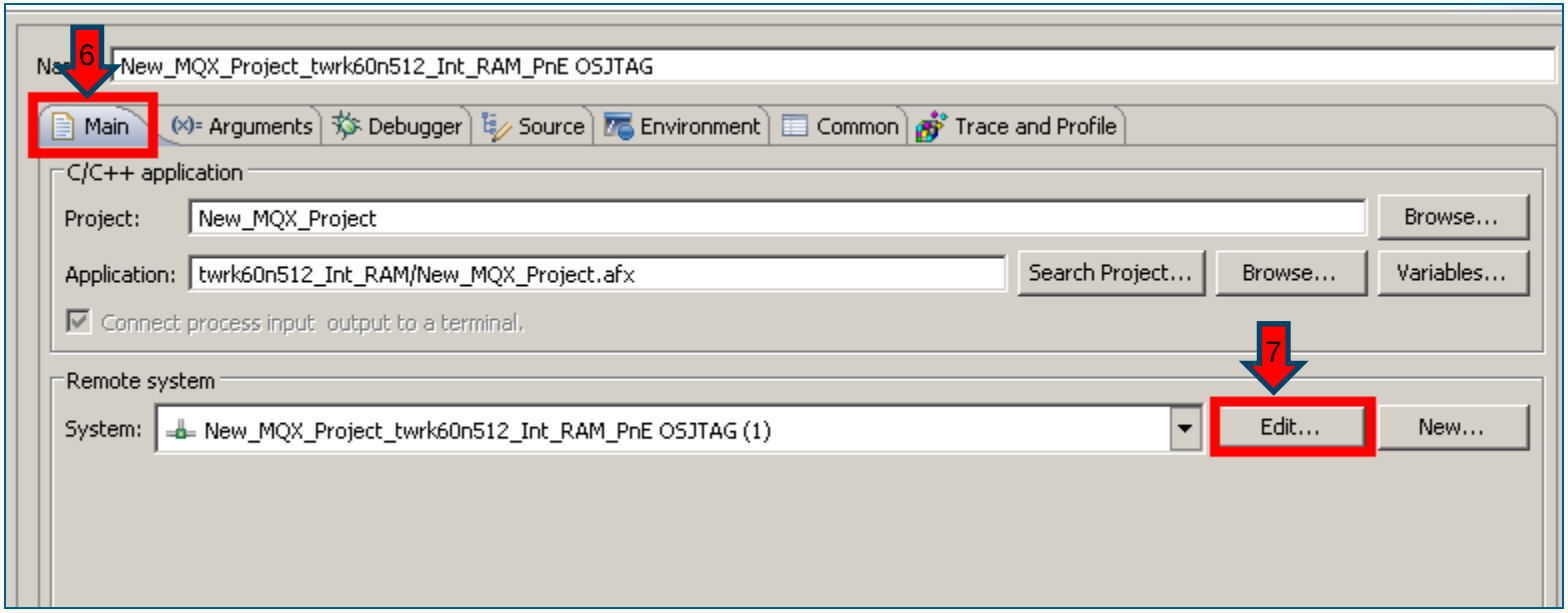
- Edit the Connection Settings of the project.





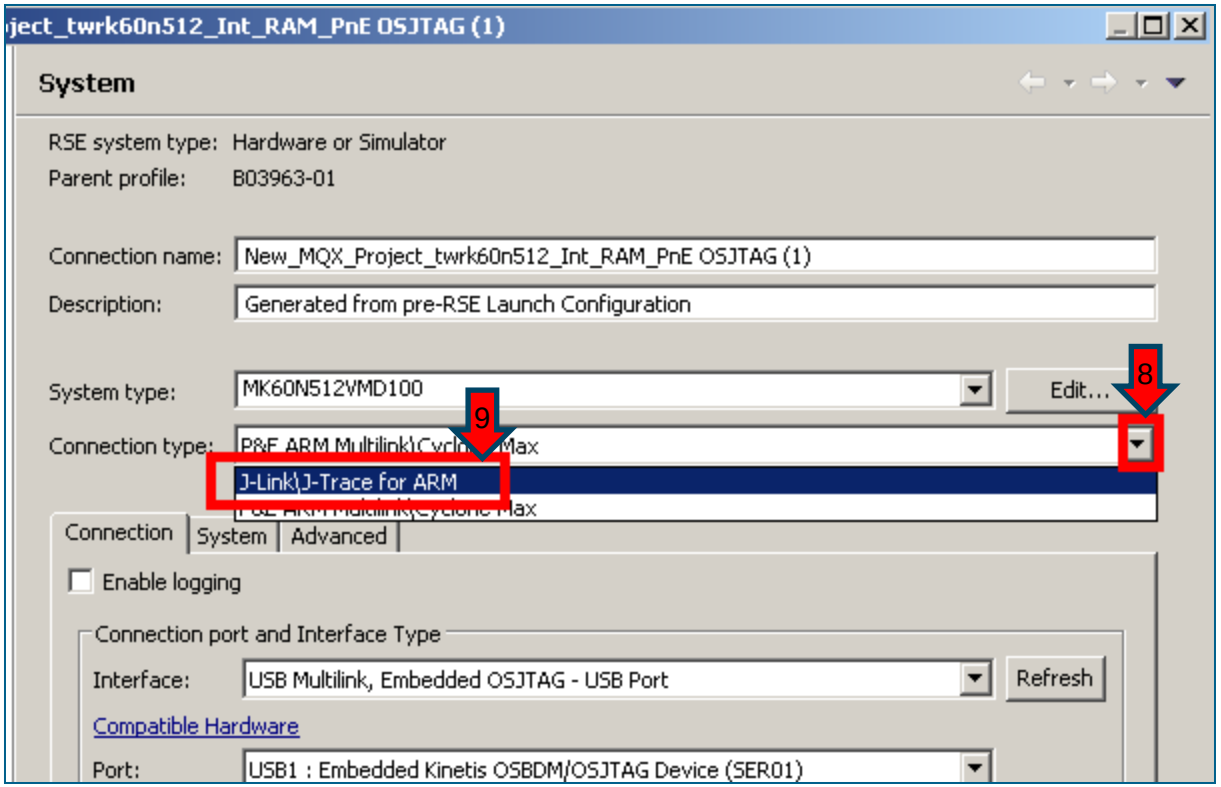
Change Connection Type

- Edit the Remote System.



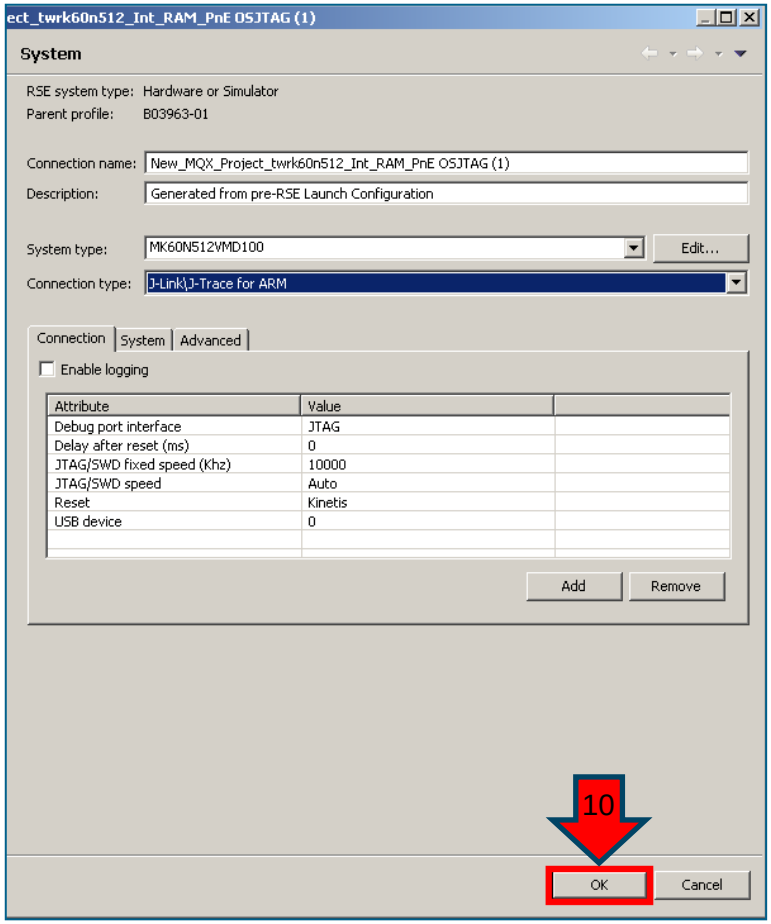
Change Connection Type

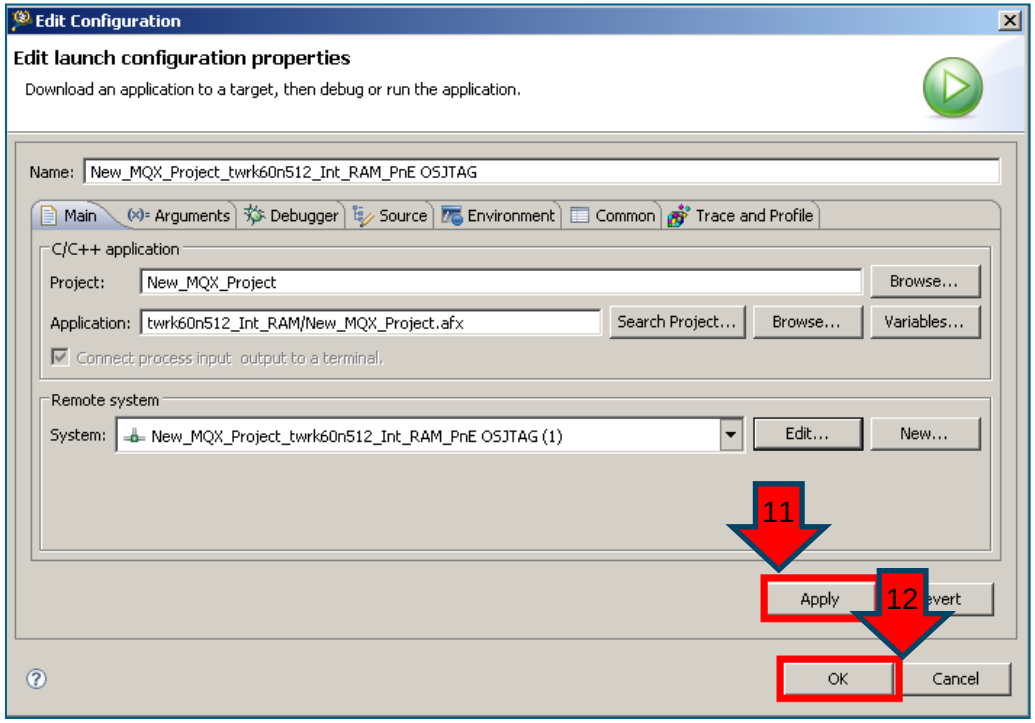
- Select **J-Link\J-Trace for ARM®**



Change Connection Type

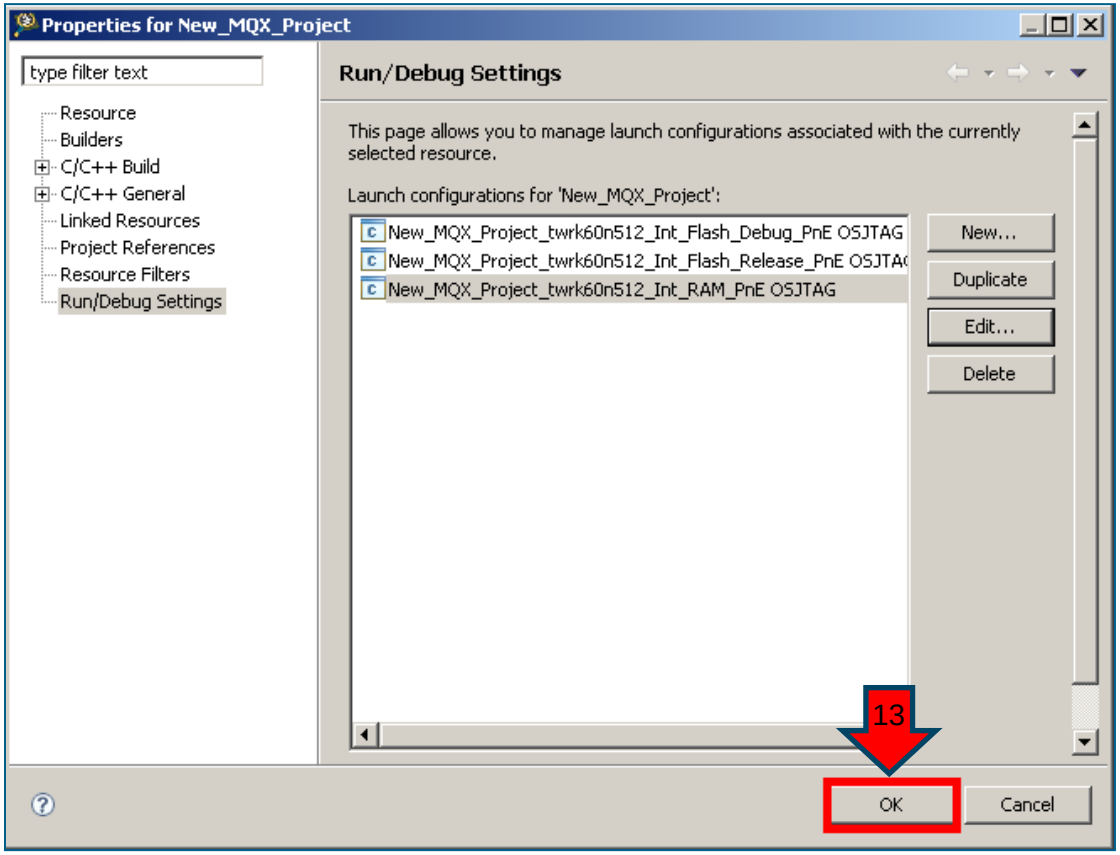
- Confirm changes.





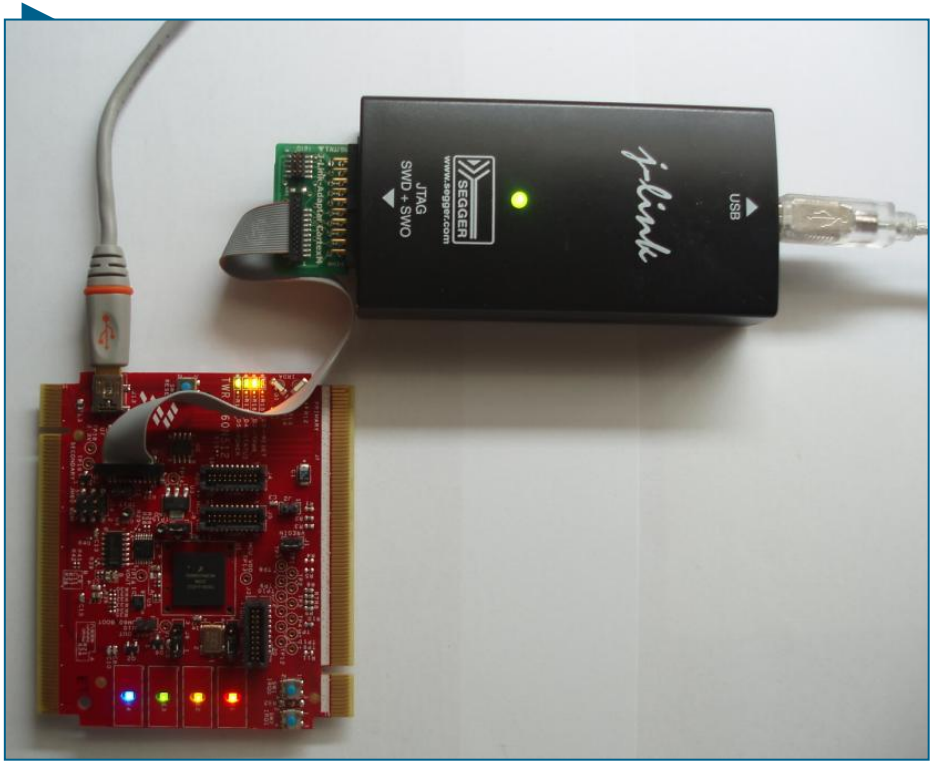
Change Connection Type

- Click OK.



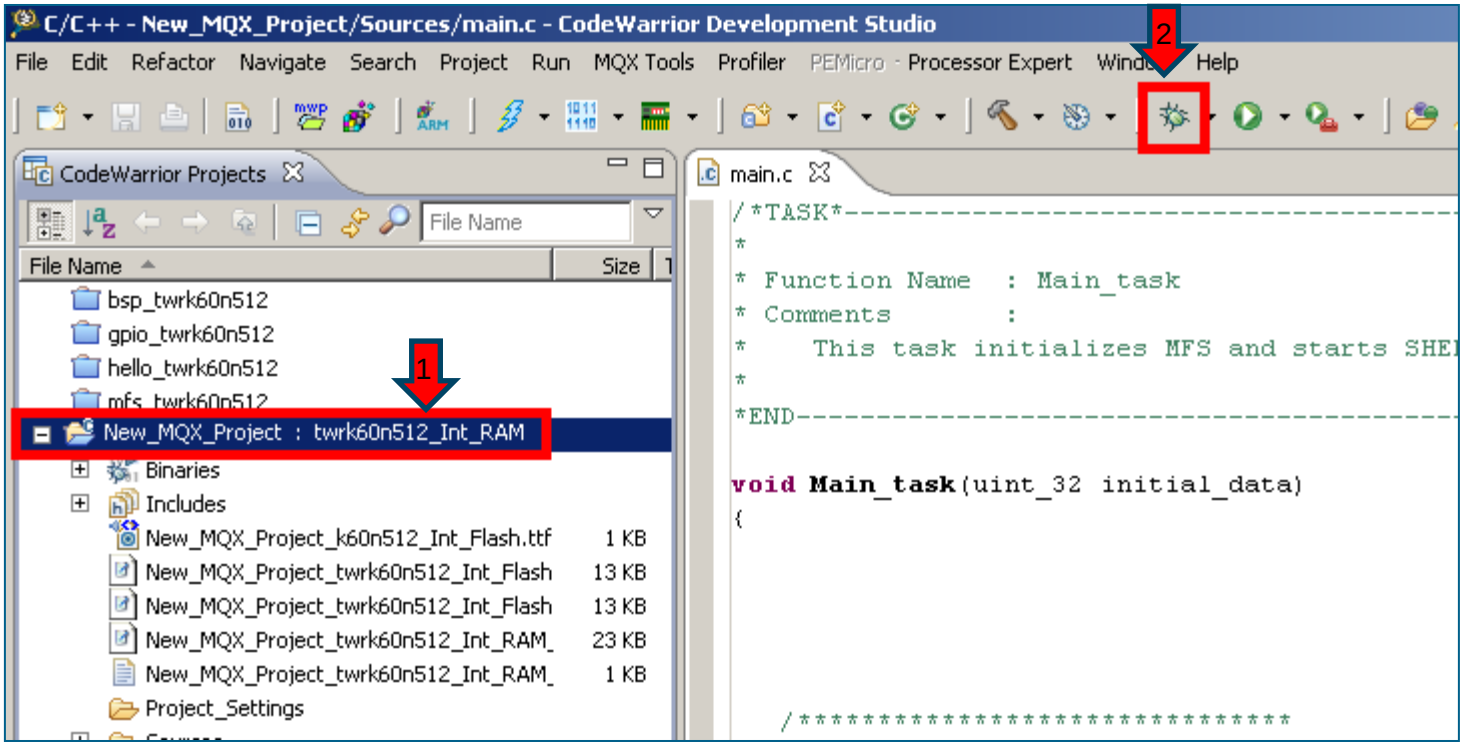
Debug with J-Link

- Connect J-Link target cable to TWR-K60N512 (J11).
- Connect USB J-Link cable to laptop.
- Connect USB Cable to TWR-K60N512 (J13) and laptop.



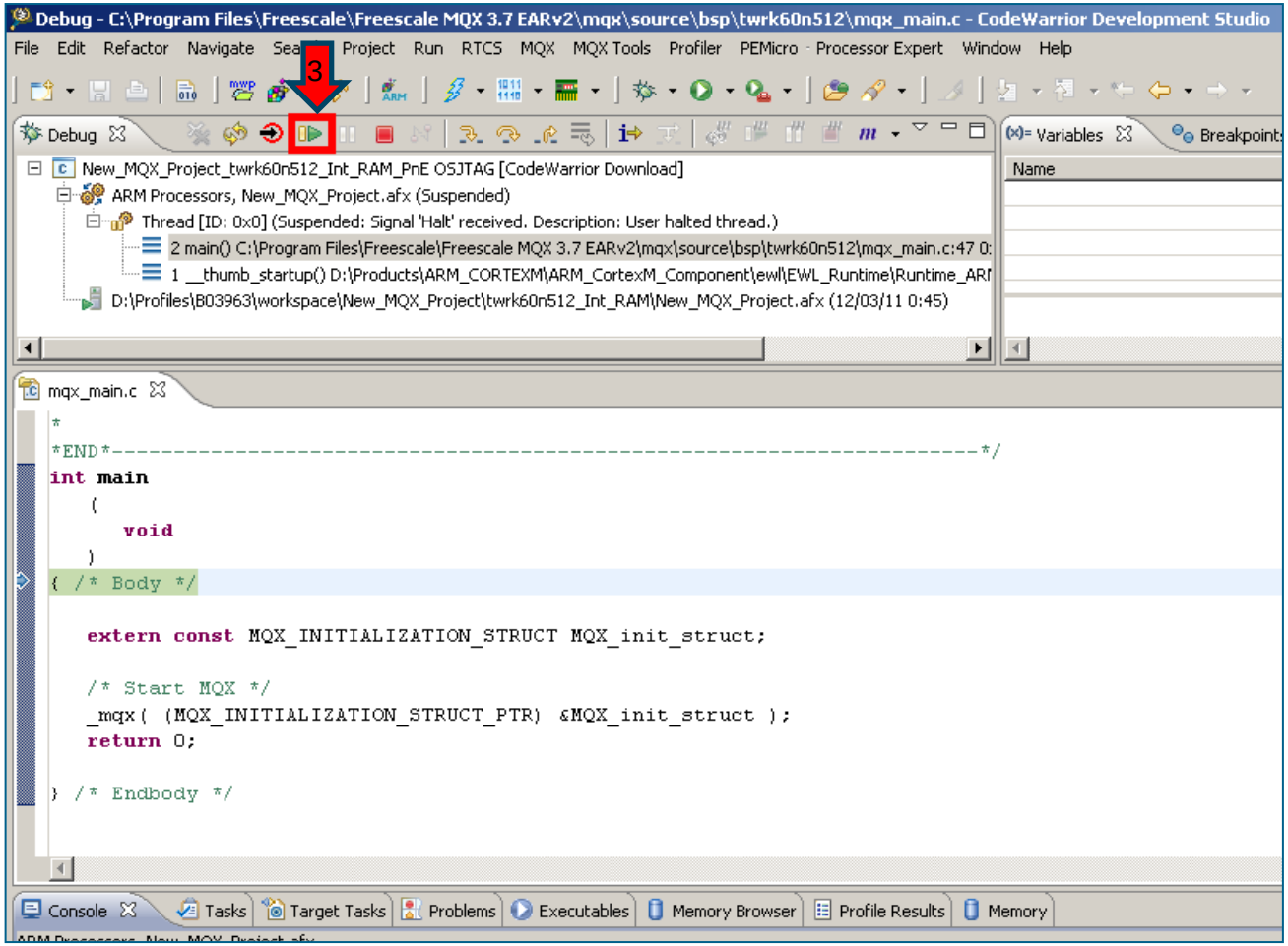
Change Connection Type

- Select **New_MQX_Project : twrk60n512_Int_RAM**
- Click Debug.



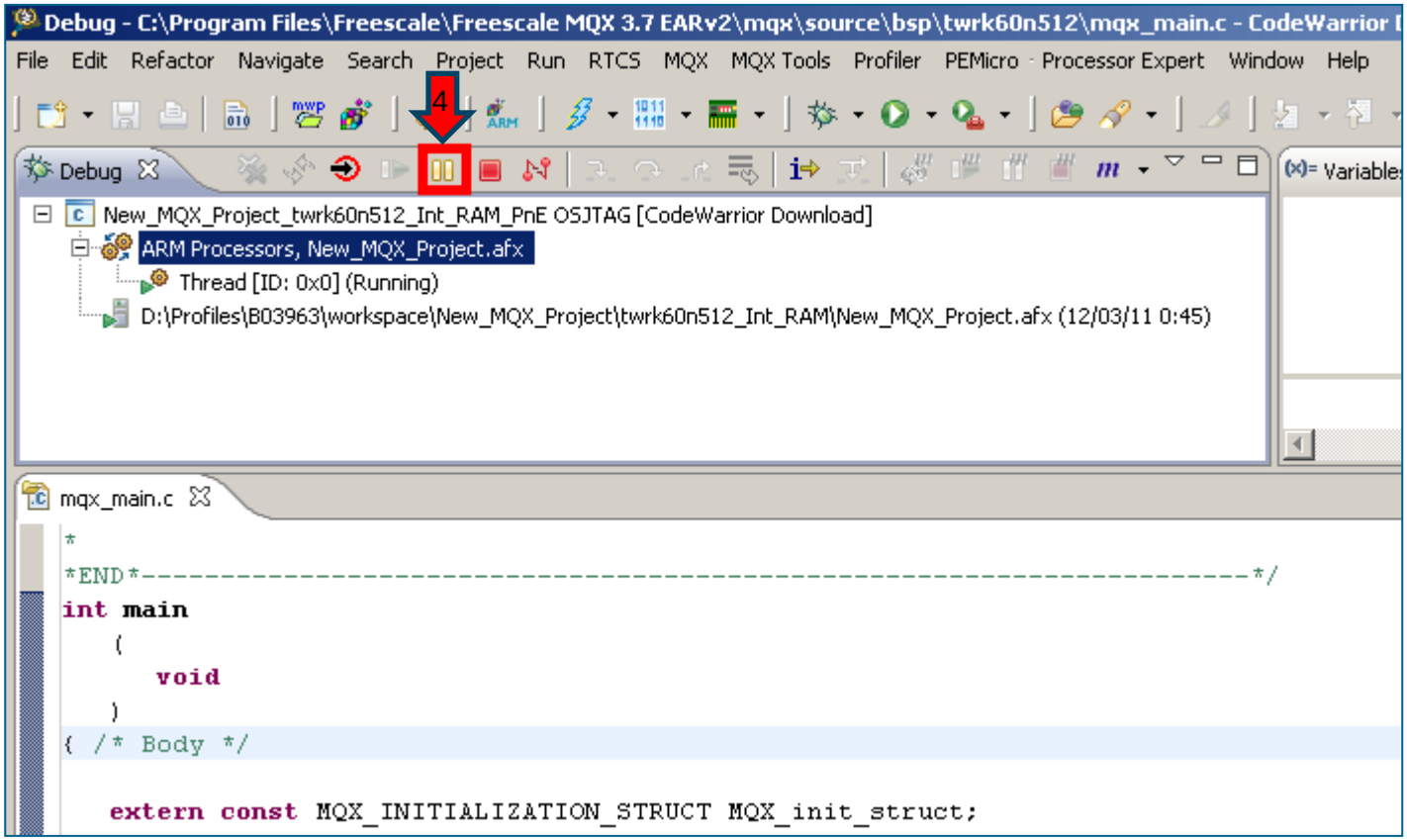
Run New MQX Project

- Execute the code '**Resume**' icon.



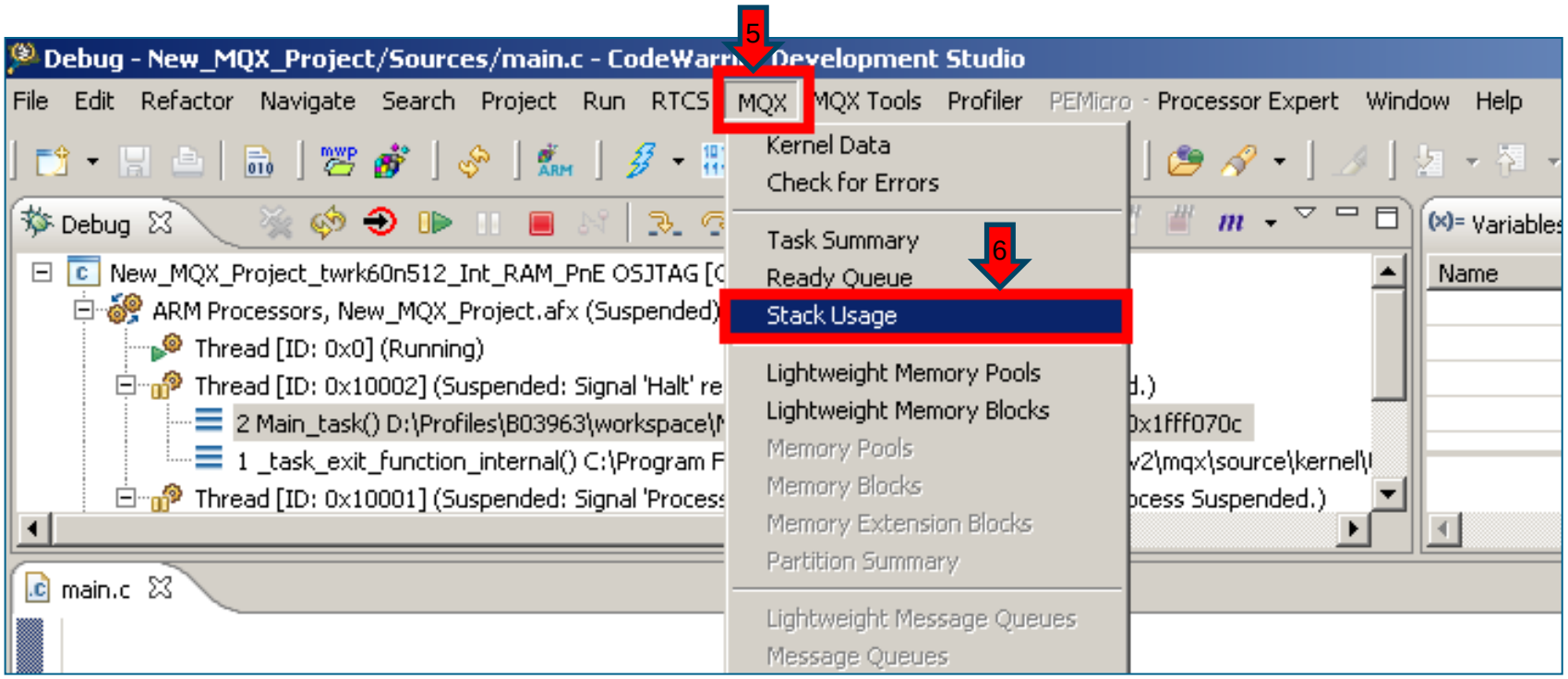
Run New MQX Project

- Pause execution.



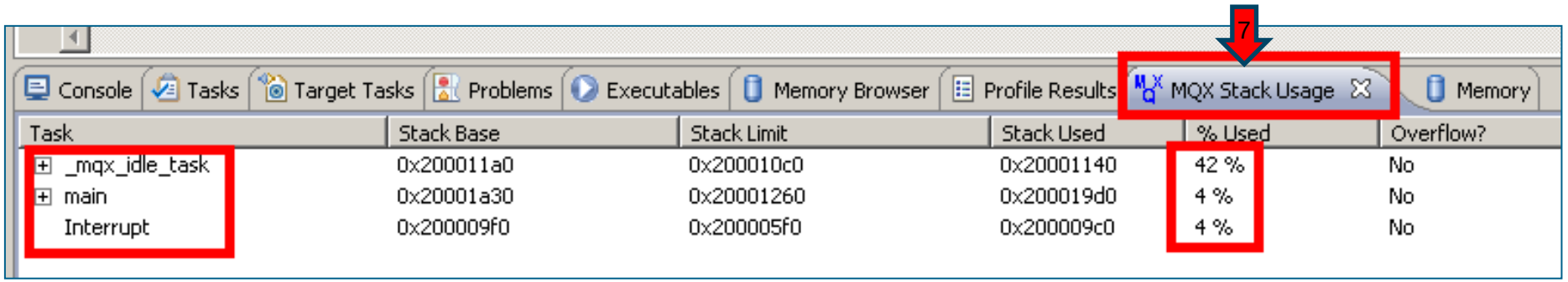
TAD: Stack Usage

- MQX -> Stack Usage.

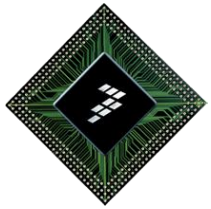


TAD: Stack Usage

- Observe Stack Data.

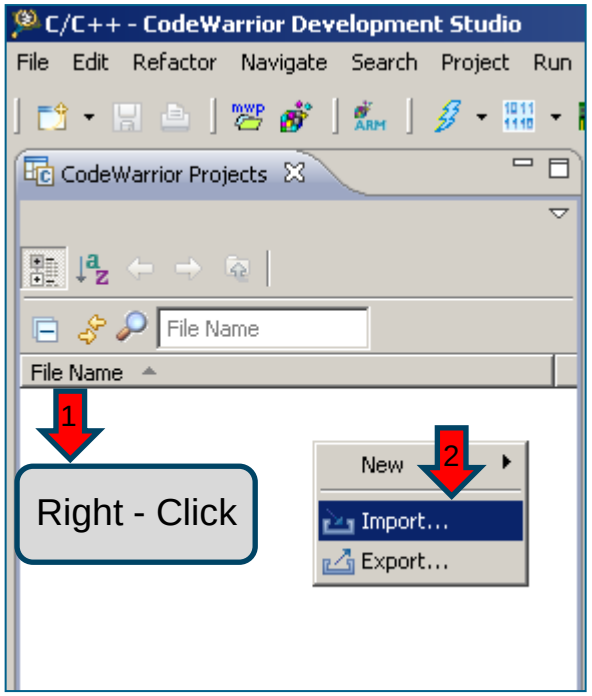


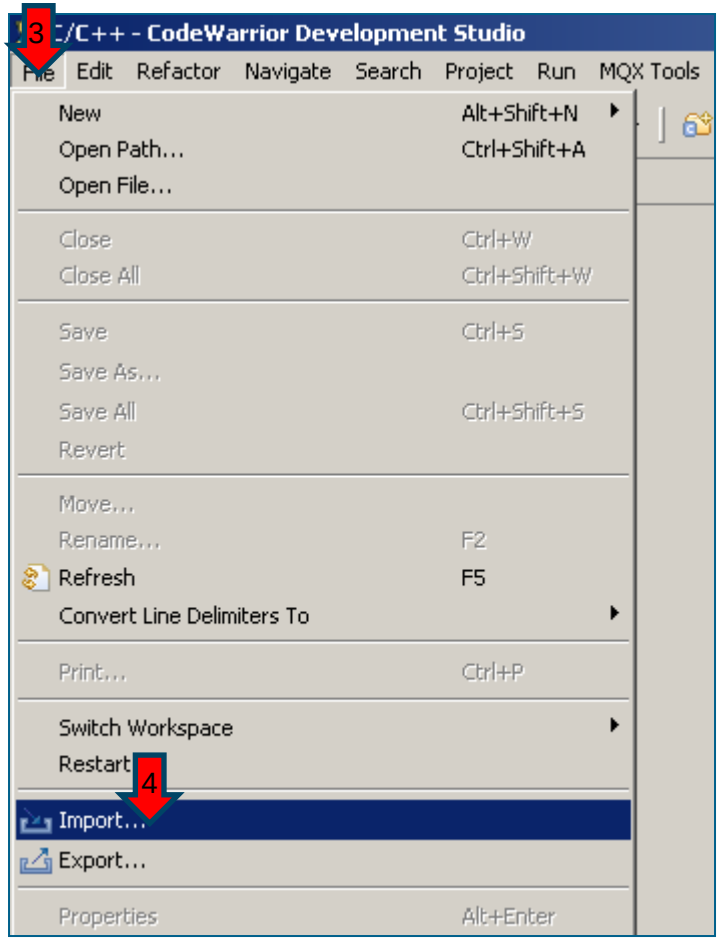
CW10.x, MQX and Processor Expert



Import MQX BSP

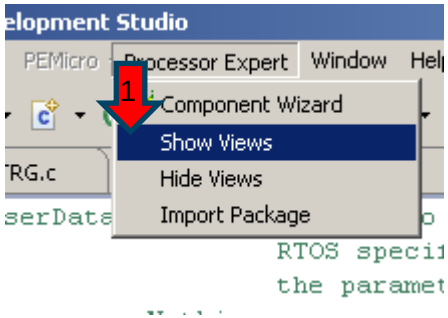
- Right-Click on Project Explorer and Import (or) File -> Import.
- All Kinetis BSP projects are Processor Expert Ready.

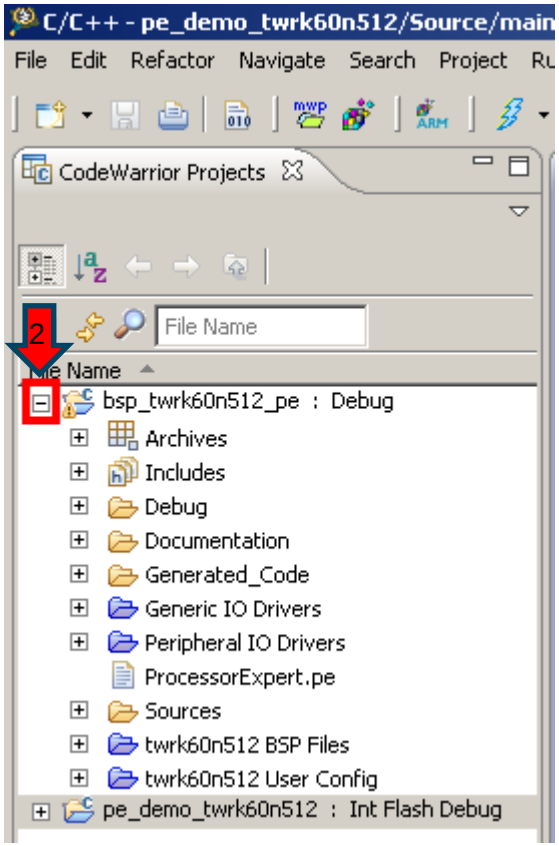




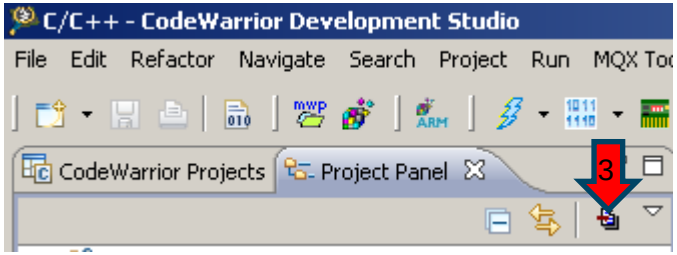
Processor Expert in MQX BSP

- Expand **bsp_twrk60n512 project view:**
- Show Processor Expert View:



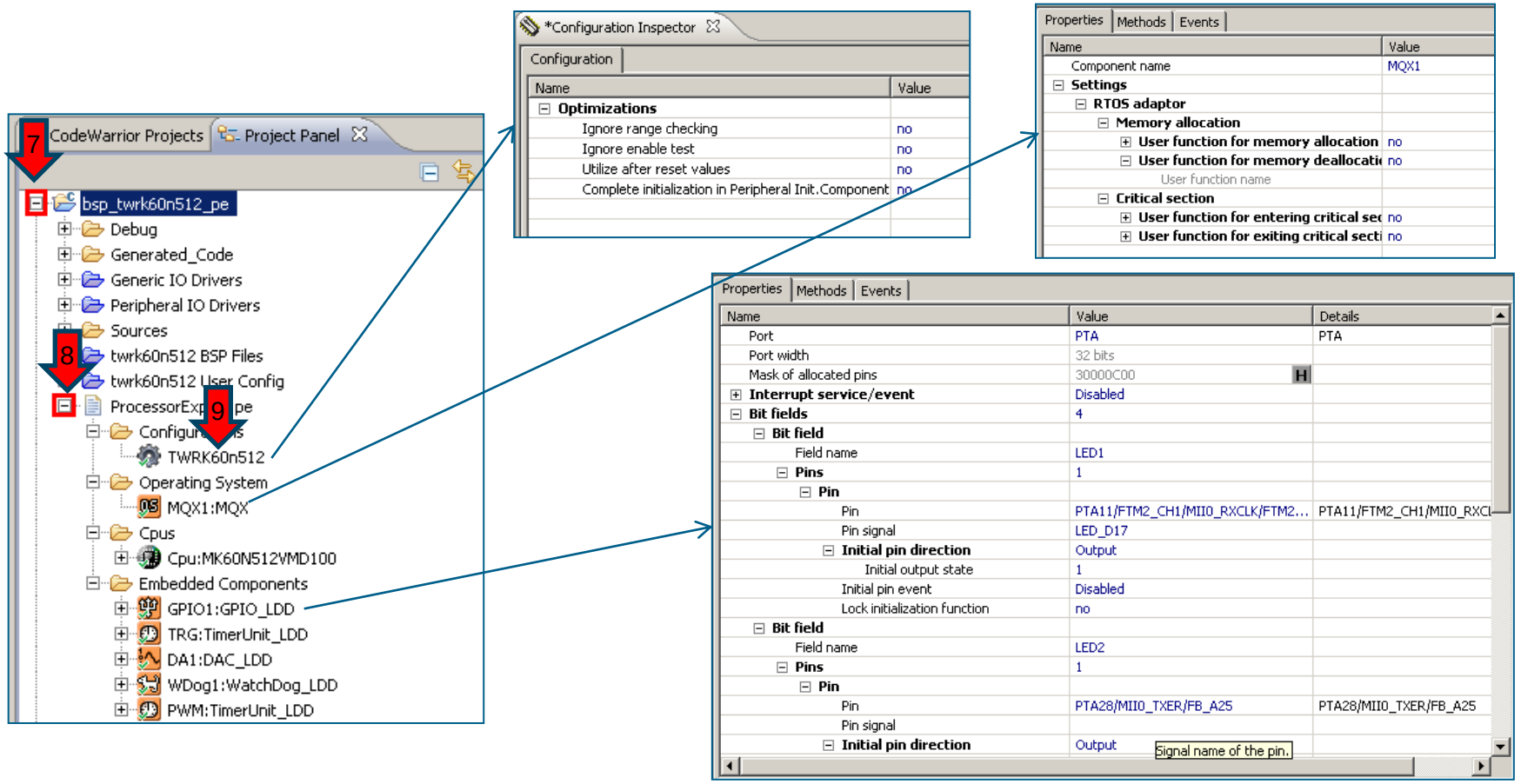


- Generate code:



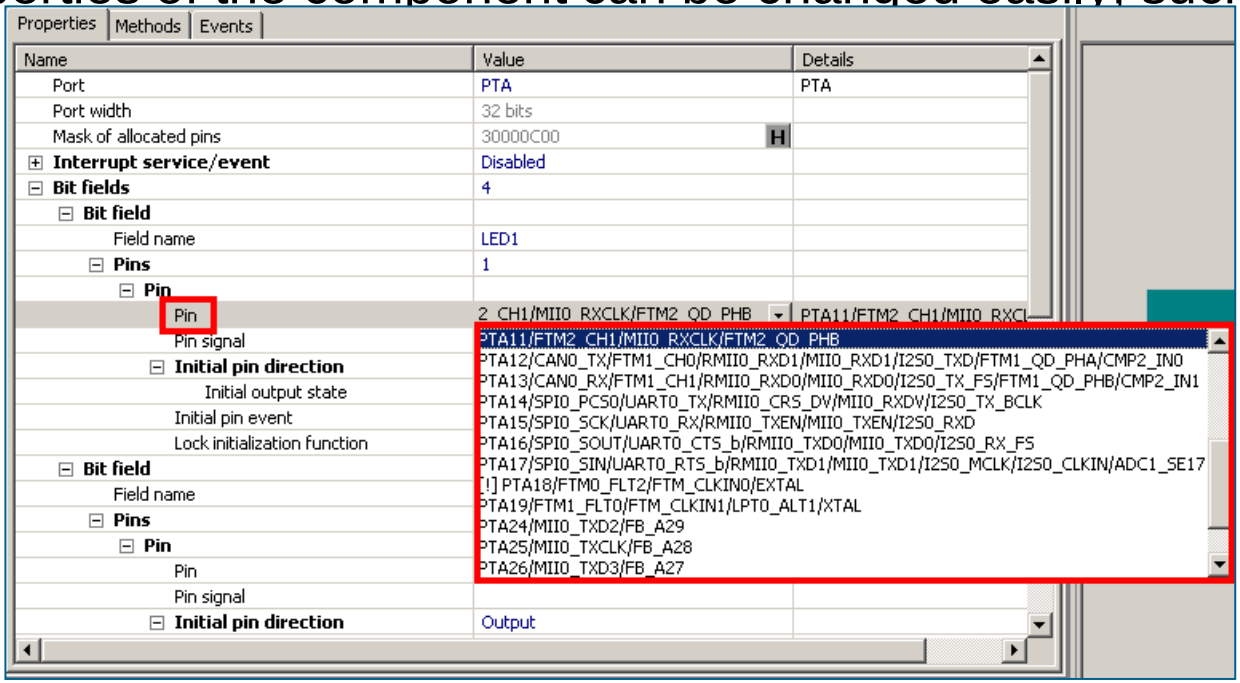
Processor Expert in MQX BSP

- Click on PE components to watch the properties.



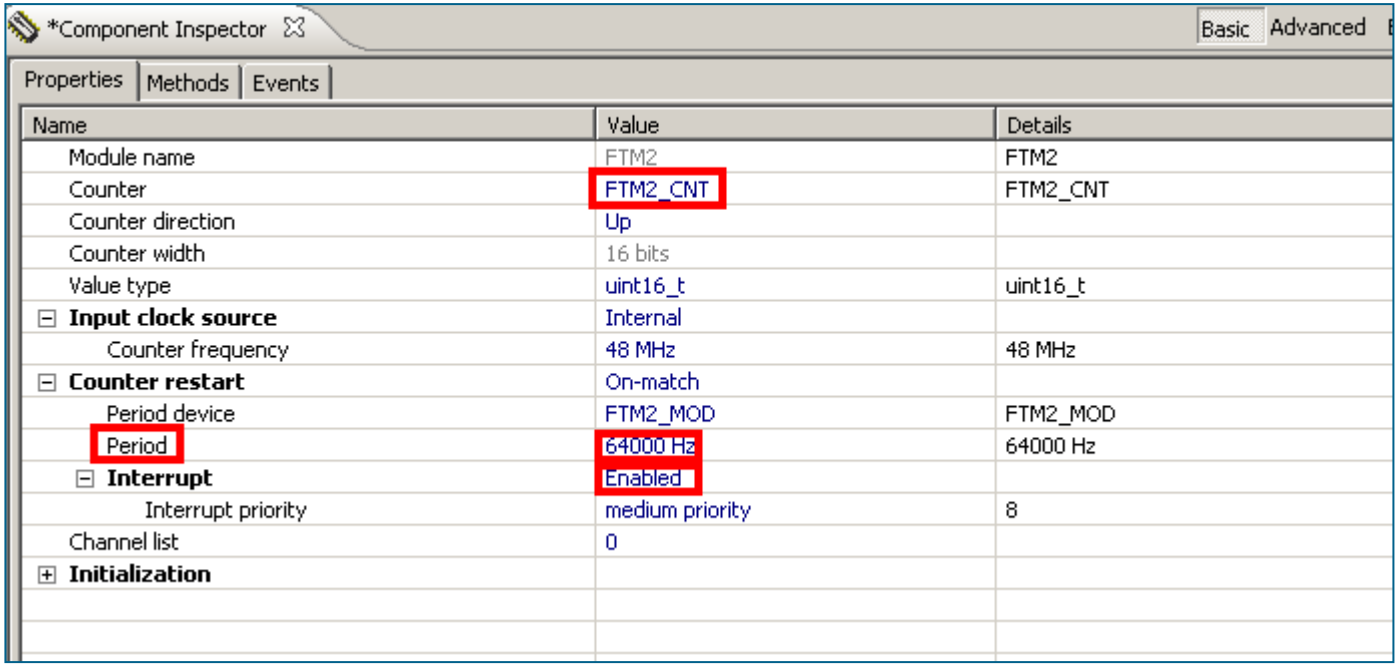
Processor Expert in MQX BSP

- Processor Expert gives you a easy way to add device drivers to the MQX BSP.
- In the BSP example two Timers, GPIO, WatchDog, and DAC are included.
- Properties of the component can be changed easily, such as GPIO pin.



Processor Expert in MQX BSP

- **GPIO1** component in BSP is driving LED's in Tower board.
- **TRG** Timer will generate a 64KHz interrupt.



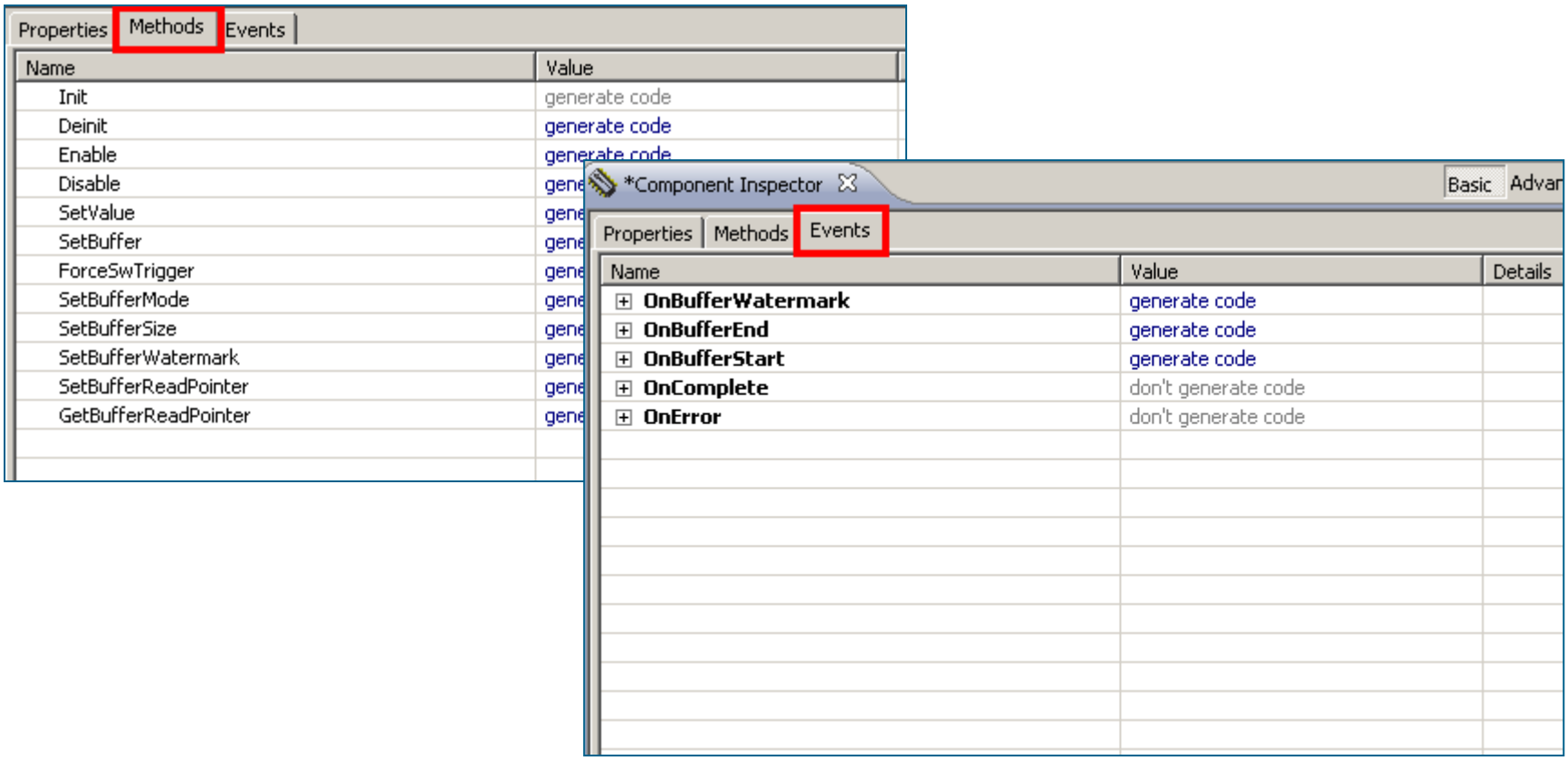
- **PWM** configures Channel 0 in Flex Timer 0 a PWM of 4096 timer-ticks

Properties | Methods | Events

Name	Value	Details
Module name	FTM0	FTM0
Counter	FTM0_CNT	FTM0_CNT
Counter direction	Up	
Counter width	16 bits	
Value type	uint16_t	uint16_t
Input clock source	Internal	
Counter frequency	48 MHz	48 MHz
Counter restart	On-match	
Period device	FTM0_MOD	FTM0_MOD
Period	4096 timer-ticks	4096 timer-ticks
Interrupt	Enabled	
Interrupt priority	medium priority	8
Channel list	1	
Channel 0		
Mode	Compare	
Compare	FTM0_C0V	FTM0_C0V
Offset	1 timer-ticks	1 timer-ticks
Output on compare	Set	
Output on overrun	Clear	
Initial state	Low	
Output pin	PTC1/SPI0_PCS3/UART1_RTS_b/FTM...	PTC1/SPI0_PCS3/UART1_RTS_b/FTM...
Interrupt	Disabled	
Initialization		
Enabled in init. code	no	
Event mask		

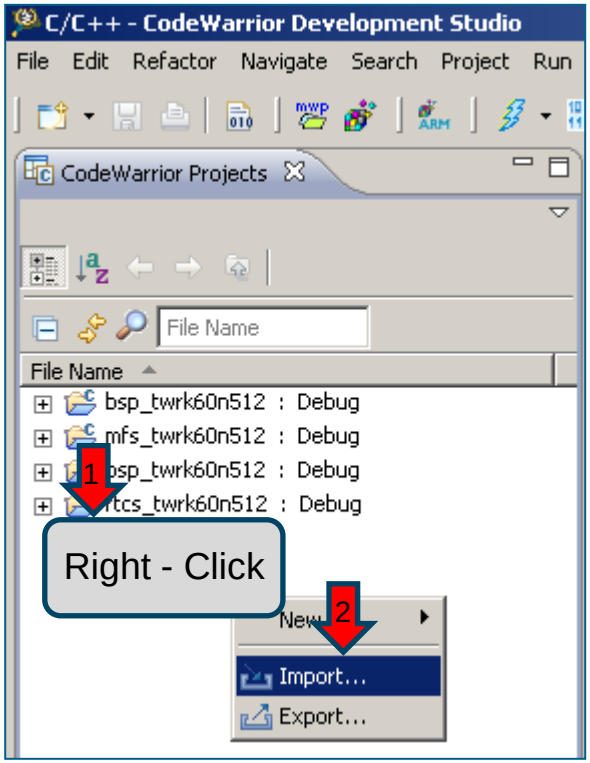
Processor Expert in MQX BSP

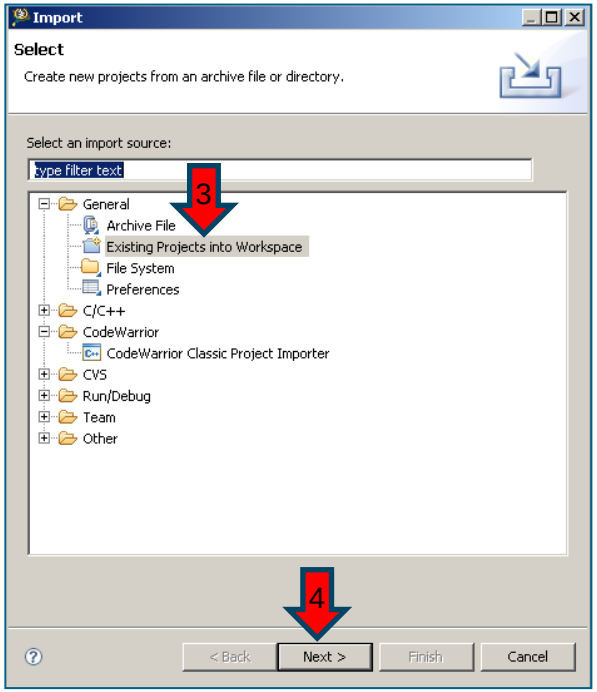
- Besides Properties, Components also include Methods and Events that we can enable or disable.

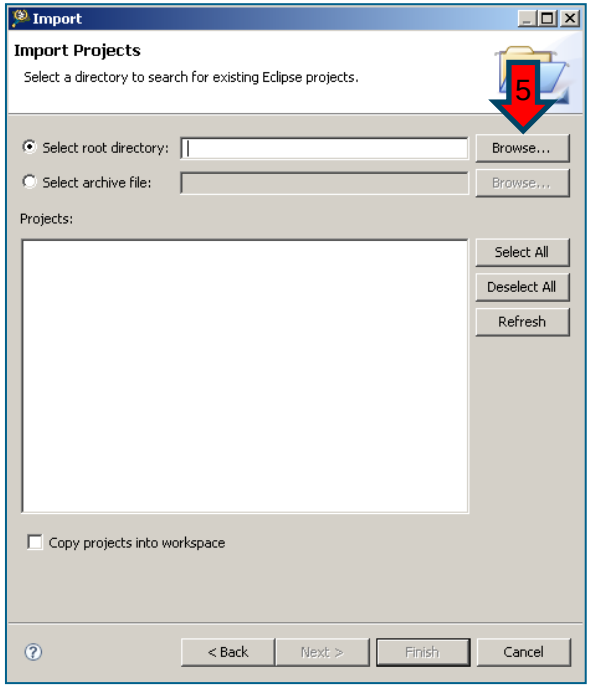


Import MQX PE Demo

- Right-Click on Project Explorer and Import.
- Select Existing Projects into Workspace and Browse.

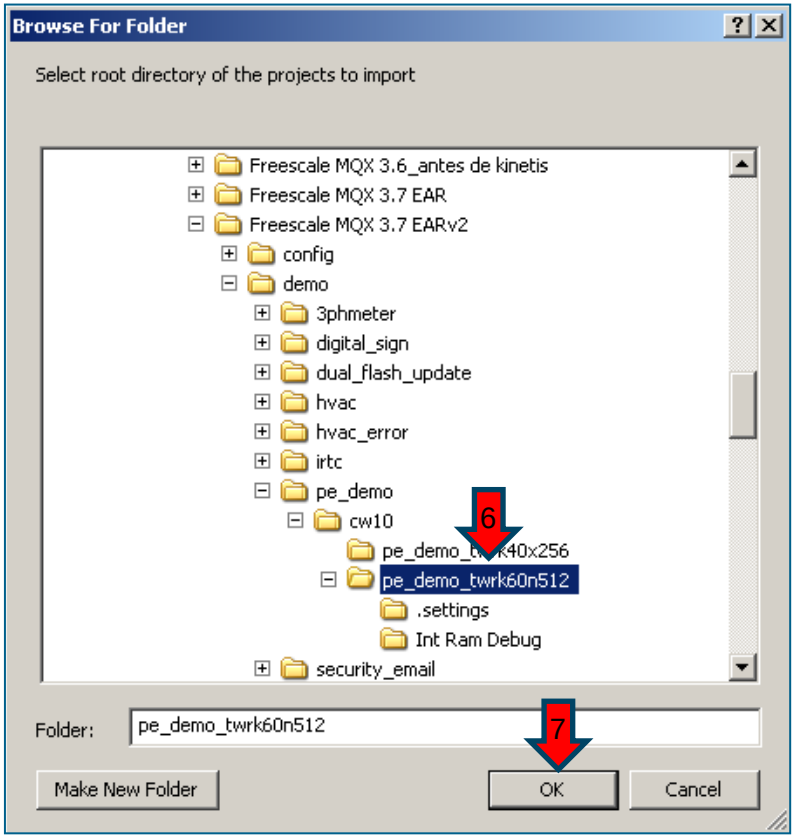


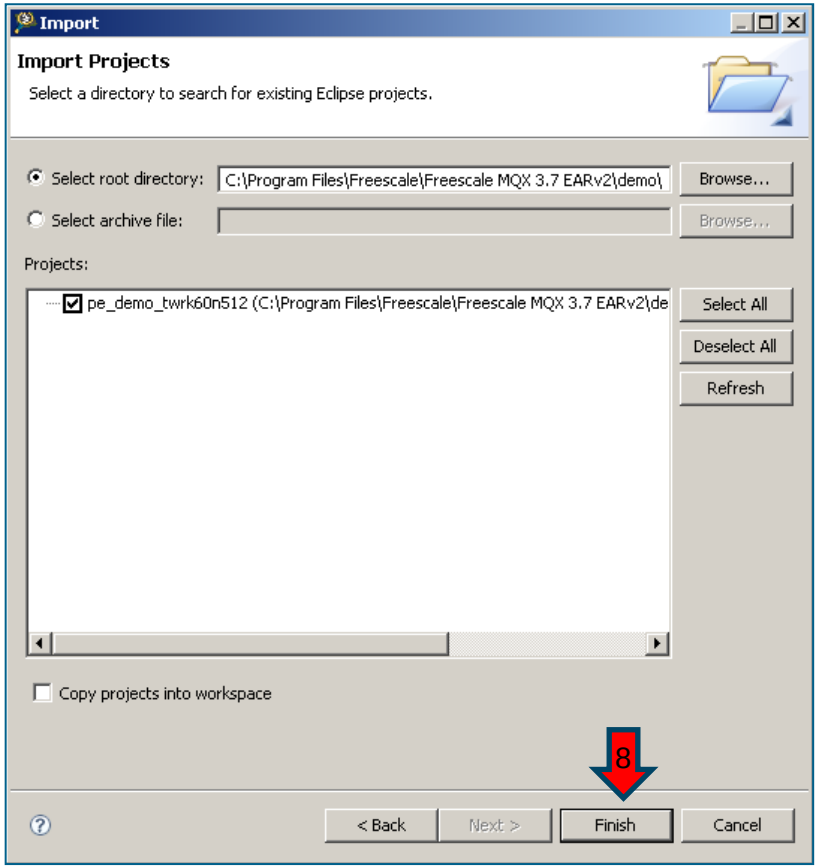




Import MQX PE Demo

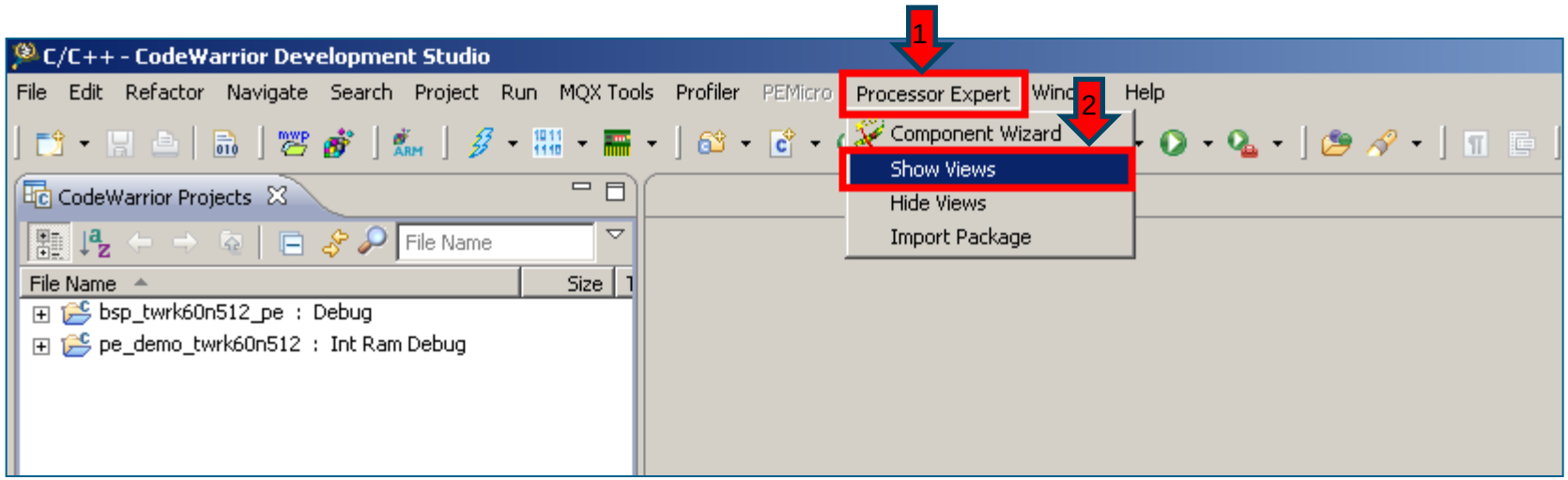
- Select *<install mqx folder>\mqx\pe_demo\pe_demo_twrk60n512*





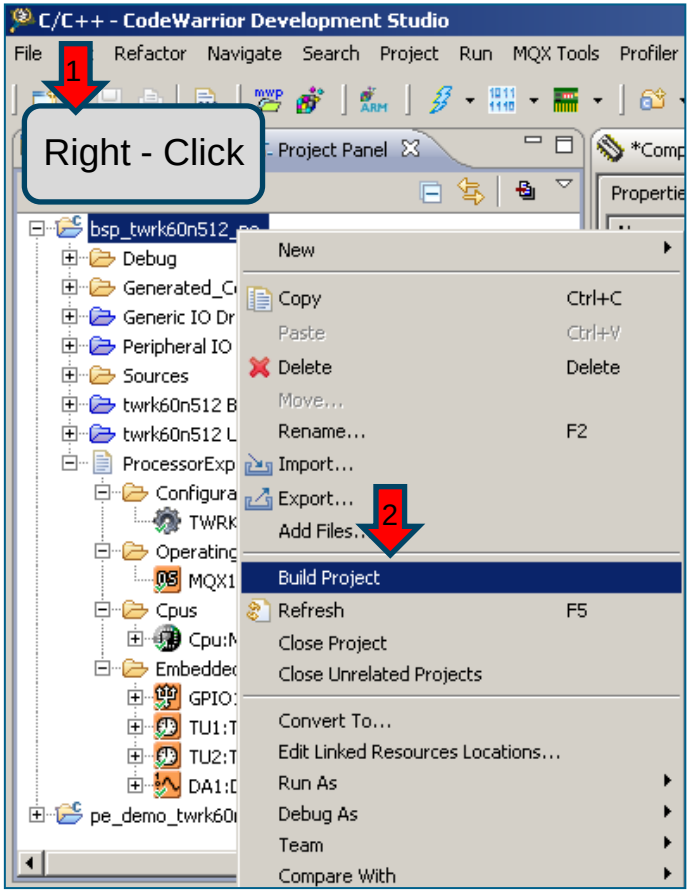
MQX PE Demo

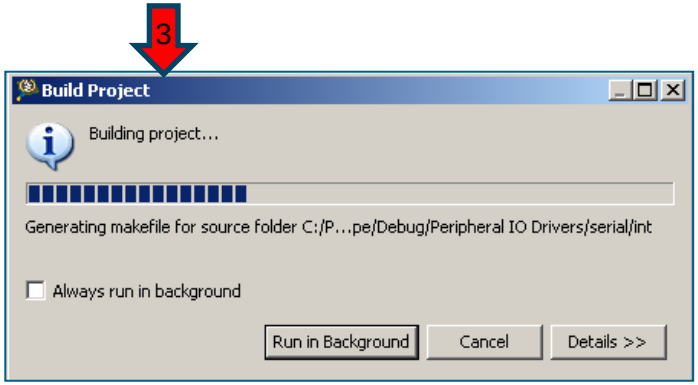
- Select in Menu : Processor Expert -> Show Views.



Build PE BSP

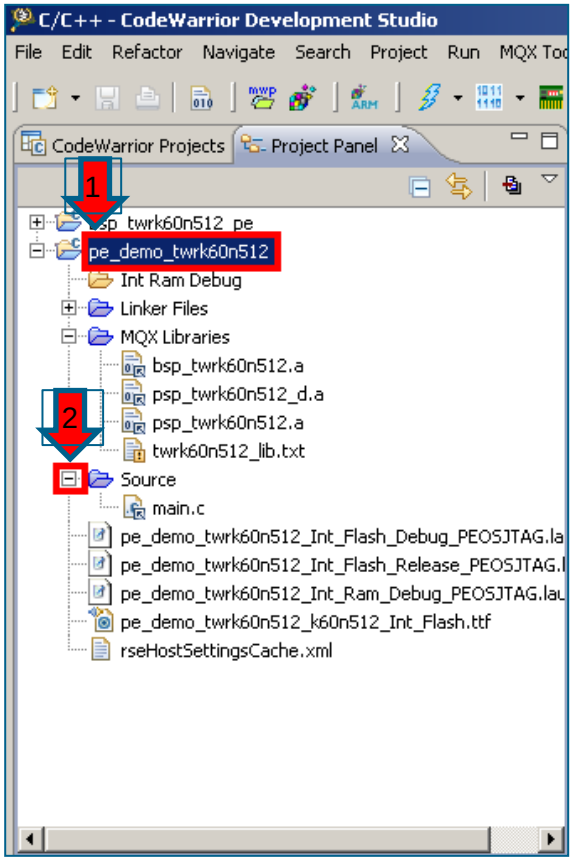
- Right-Click on Project Explorer **bsp_twrk60n512_pe** and Build Project.

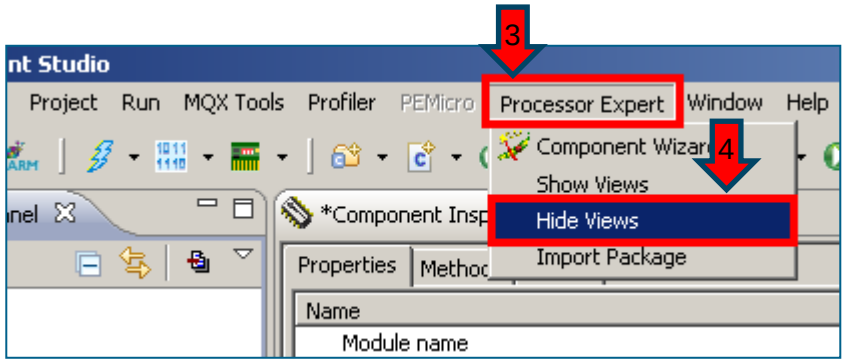




Build PE Demo

- Expand **pe_demo_twrk60n512** project view.
- Hide Processor Expert View.





MQX PE Demo

- Demo Application demonstrates how to use Processor Expert to configure MQX BSP:
  - It generates sine signal with given period on DACO pin.
  - PWM signal is generated using FlexTimer FTM0 Channel 0.
  - It toggles LEDs (D9-D11) on board using GPIO driver.
  - ewm_task task is periodically refreshing watchdog.

- Application creates four tasks as shown below:

```
/* Task template list */
const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
   /* Task Index,   Function,   Stack,  Priority,   Name,         Attributes,          Param,   Time Slice
    { DAC_TASK,     dac_task,    400,       8,      "DAC Task",  MQX_AUTO_START_TASK,   0,          0 },
    { PWM_TASK,     pwm_task,    400,       9,      "PWM Task",  MQX_AUTO_START_TASK,   0,          0 },
    { EWM_TASK,     ewm_task,    300,       10,     "EWM Task",  MQX_AUTO_START_TASK,   0,          0 },
    { LED_TASK,     led_task,    200,       11,     "LED Task",  MQX_AUTO_START_TASK,   0,          0 },
    { 0 }
};
```

MQX PE Demo

- Application uses PE LDD drivers.
- To use PE drivers, some ‘handler’ variables must be declared:

```
/* LED */
LDD_TDeviceData      *LED_DeviceData;
LDD_TError            LED_Error;

static int                   count = 1;
static int                   sign = 1;
static LDD_GPIO_TBitField    LED;
```

```
static vuint_32              pwm_task_count;
static LDD_TDeviceData      *PWM_DeviceData;
static LDD_TError            PWM_Error;
volatile PWM_TValueType      PWM_Value;
volatile PWM_TValueType      PWM_MaxValue;
volatile PWM_TValueType      PWM_Step;
```

```
/* DAC */
#define                       DA1_INTERNAL_BUFFER_SIZE    (16)
LDD_TDeviceData              *DA1_Device;
LDD_TUserData                *DA1_UserDataPtr;
LDD_TError                    DA1_Error;
LDD_DAC_TBufferWatermark      DA1_WatermarkValue = LDD_DAC_BUFFER_WATERMARK_L4;
```

MQX PE Demo

- Task must initialize the LDD components.

```
DA1_UserDataPtr = NULL;
DA1_Device      = DA1_Init(DA1_UserDataPtr);
if (DA1_Device == NULL)  {
    puts("failed");
    _task_block();
} else {
    puts("done");
```

```
EWM_DeviceData = WDog1_Init(NULL);
if (EWM_DeviceData == NULL)  {
puts("failed");
    _task_block();
}
else  {
    puts("done");
}
```

```
PWM_DeviceData = PWM_Init(NULL);
if (PWM_DeviceData == NULL)  {
puts("failed");
    _task_block();
}
else  {
    puts("done");
}
```

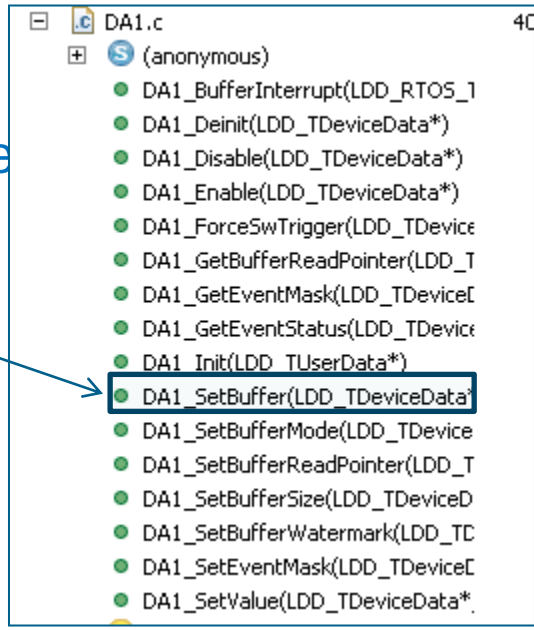
MQX PE Demo

- Enable the components:

```
PWM_Error = PWM_Enable(PWM_DeviceData);
EWM_Error = WDog1_Enable(EWM_DeviceData);
```

- Application can use the components Methods:

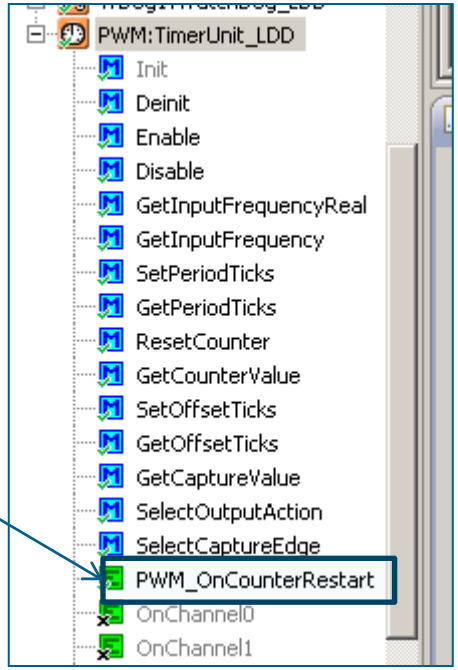
```
DA1_Error = DA1_SetBuffer(DA1_Device, GEN_Buffe
  DA1_INTERNAL_BUFFER_SIZE, 0);
```



MQX PE Demo

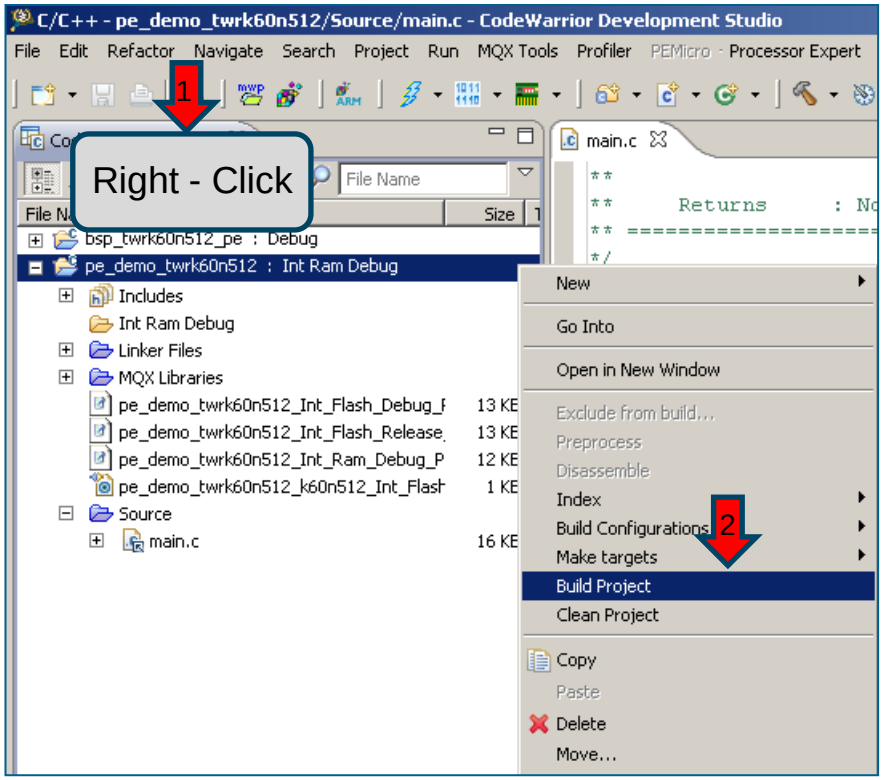
- Finally, implement the Events.

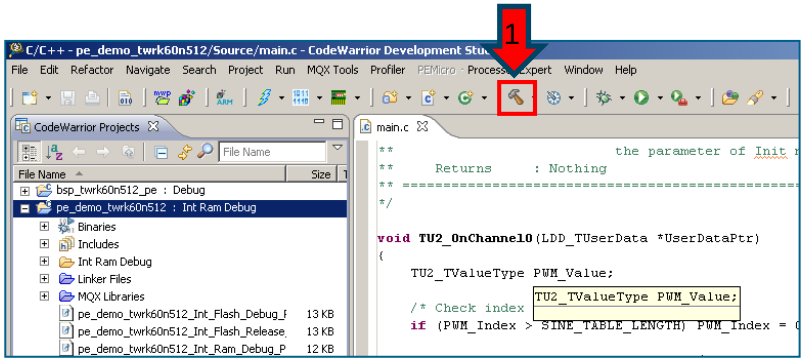
```
/*
void PWM_OnCounterRestart(LDD_TUserData *UserDataPtr)
{
    /* Increment PWM duty-cycle from 0-100% */

    PWM_Value += PWM_Step;


    if (PWM_Value > PWM_MaxValue) PWM_Value = 0;

    /* Set new PWM channel value */
    PWM_Error = PWM_SetOffsetTicks(PWM_DeviceData, 0, PWM_Value);
}
```

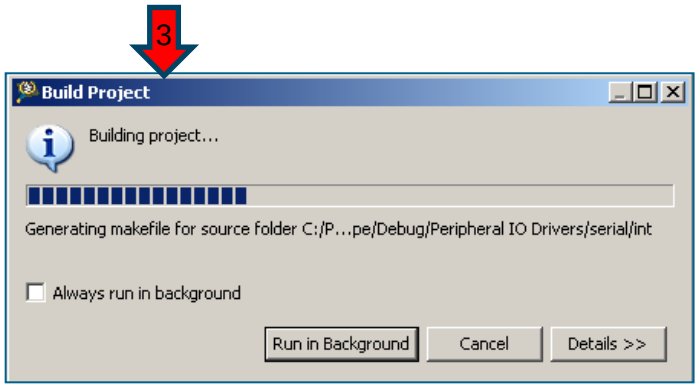


Build PE Demo

- Right-Click on the Project Explorer **pe_demo_twrk60n512** and Build Project or click on the icon 

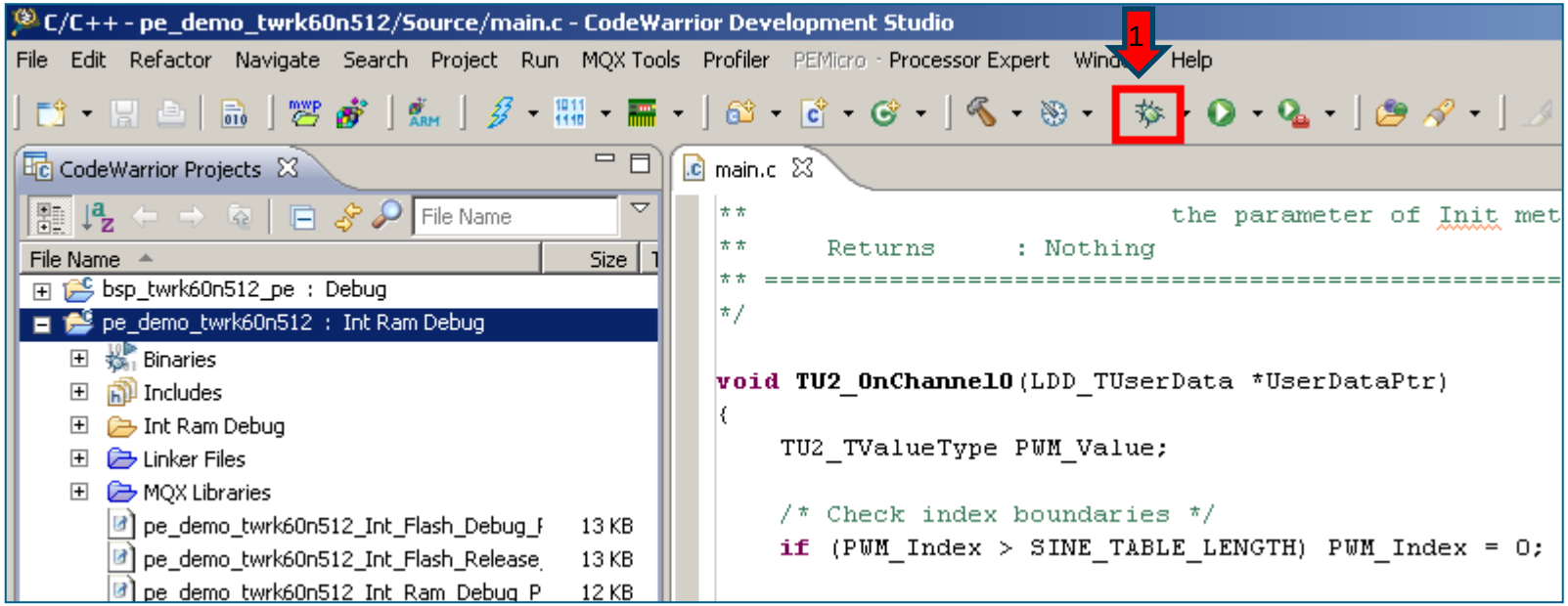






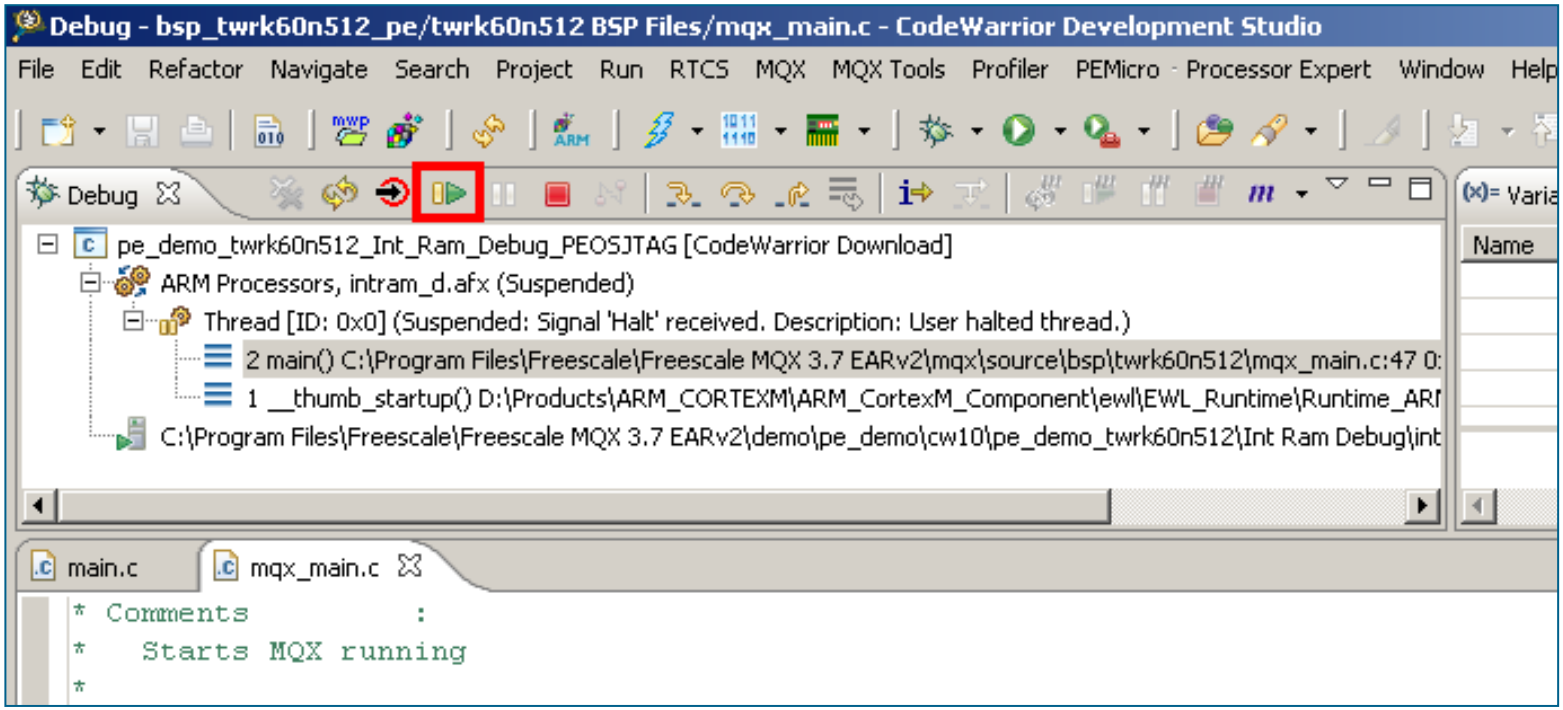
Run MQX PE Demo

- Click on the Debug icon.



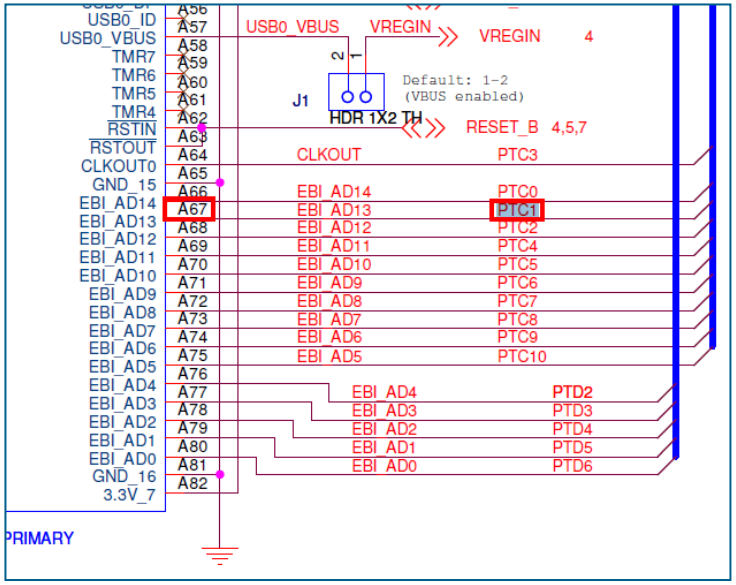
Run MQX PE Demo

- Click on the Resume (F8).



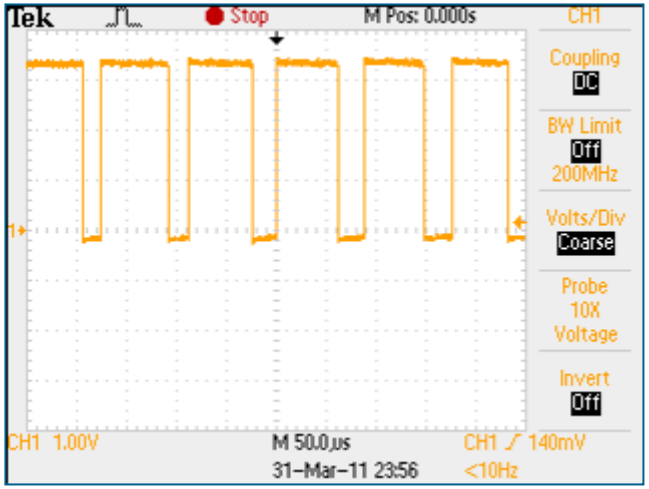
Run MQX PE Demo

- Check PWM output on A67.

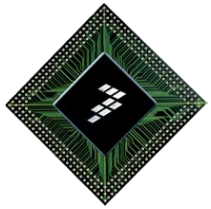


Properties | Methods | Events

Name	Value	Details
Module name	FTM0	FTM0
Counter	FTM0_CNT	FTM0_CNT
Counter direction	Up	
Counter width	16 bits	
Value type	uint16_t	uint16_t
Input clock source	Internal	
Counter frequency	48 MHz	48 MHz
Counter restart	On-match	
Period device	FTM0_MOD	FTM0_MOD
Period	4096 timer-ticks	4096 timer-ticks
Interrupt	Enabled	
Interrupt priority	medium priority	8
Channel list	1	
Channel 0		
Mode	Compare	
Compare	FTM0_C0V	FTM0_C0V
Offset	1 timer-ticks	1 timer-ticks
Output on compare	Set	
Output on overrun	Clear	
Initial state	Low	
Output pin	PTC1/SPI0_PCS3/UART1_RTS_b/FTM...	PTC1/SPI0_PCS3/UART1_RTS_b/FTM...
Interrupt	Disabled	
Initialization		
Enabled in init. code	no	



CW10.x, MQX and PE : New LDD driver

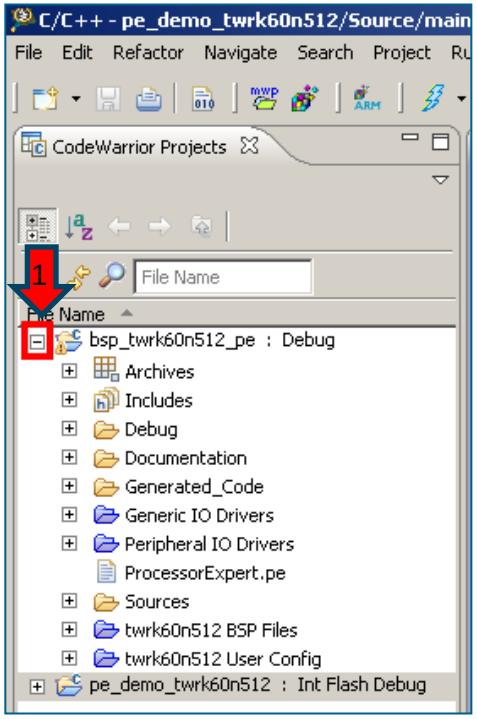


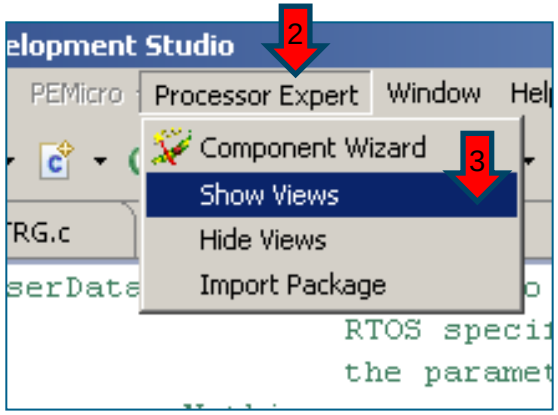
ARM POWERED®

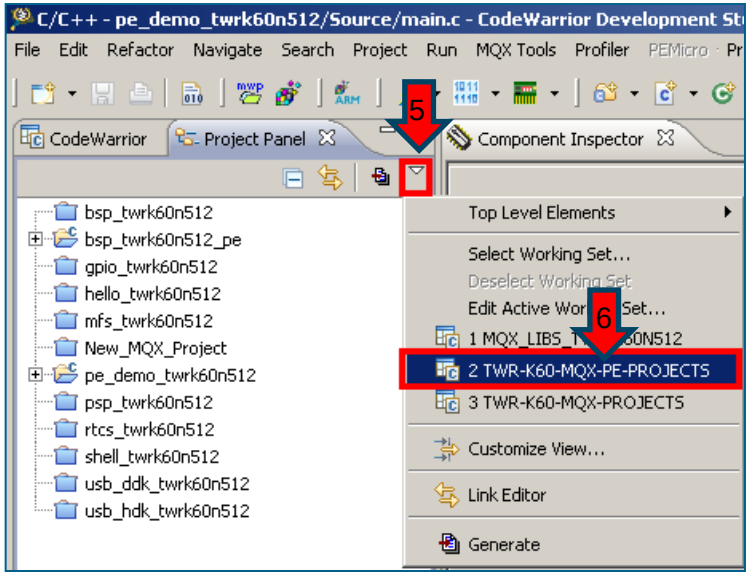
freescale™ semiconductor

New LLD Driver

- Expand **bsp_twrk60n512_pe** project view.
- Show Processor Expert View.
- Select PE Projects Working Set.

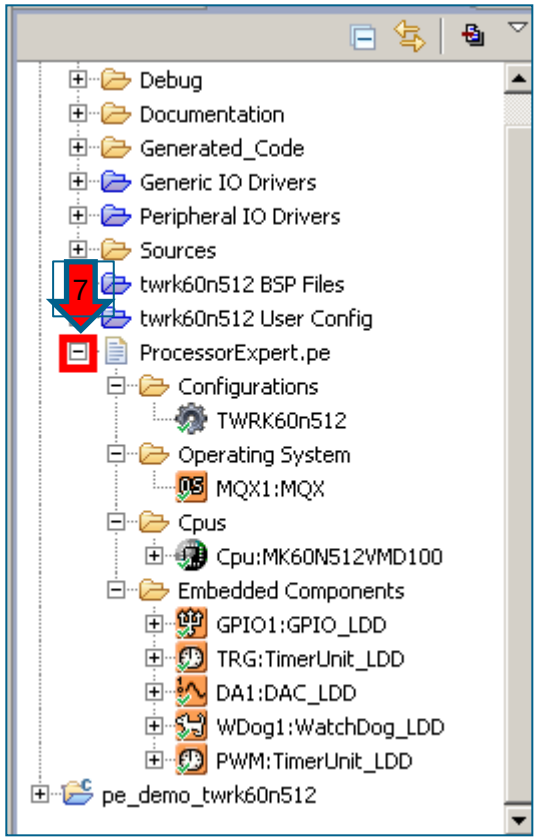


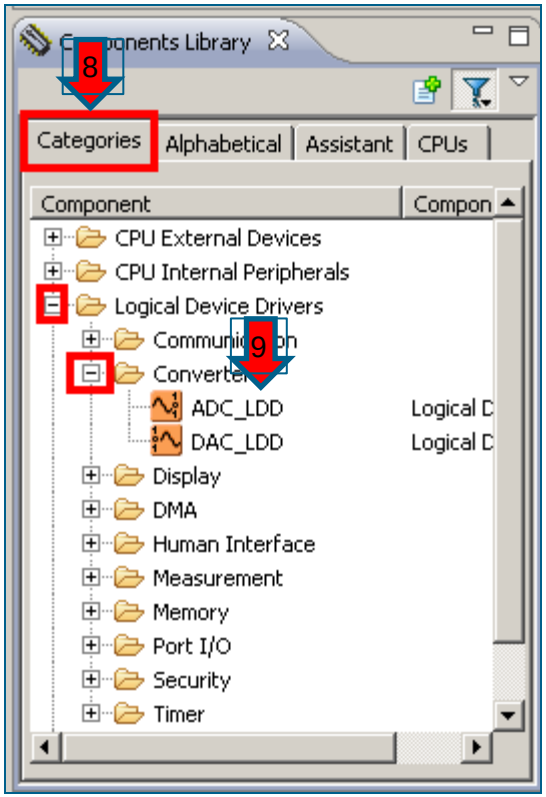




New LLD Driver

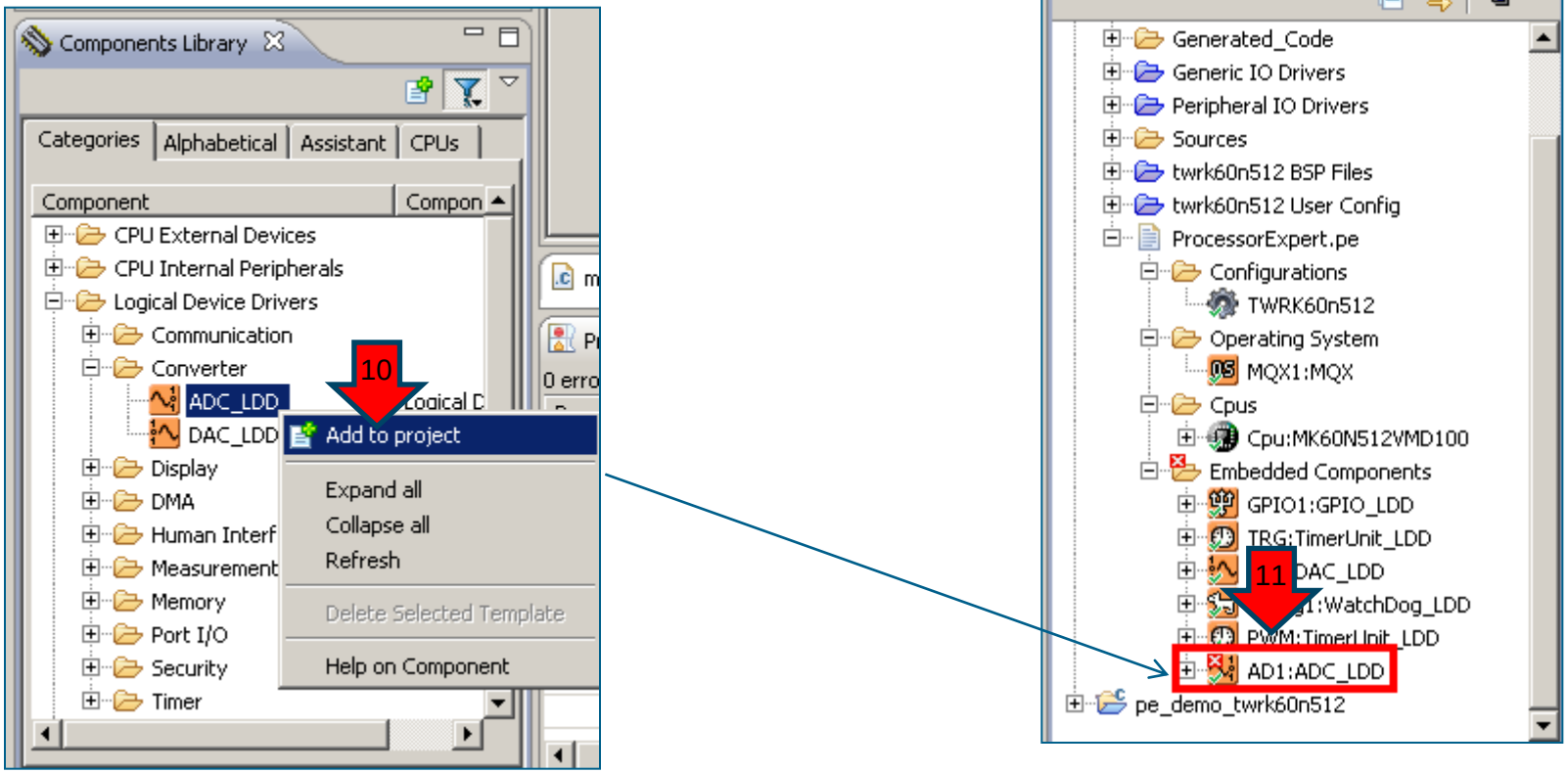
- Expand Processor Expert Project View.
- Search ADC_LDD in the Components Library window.





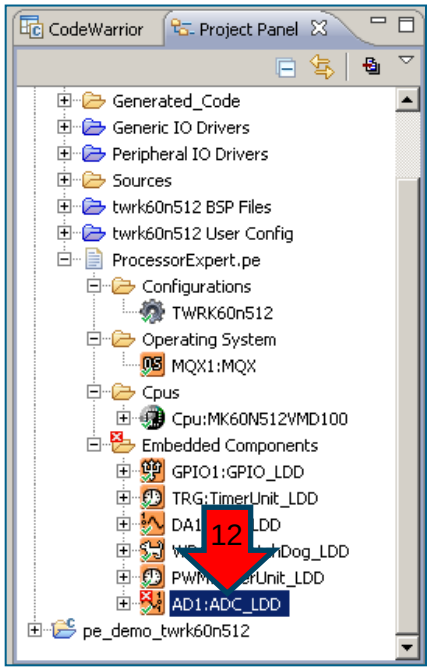
New LLD Driver

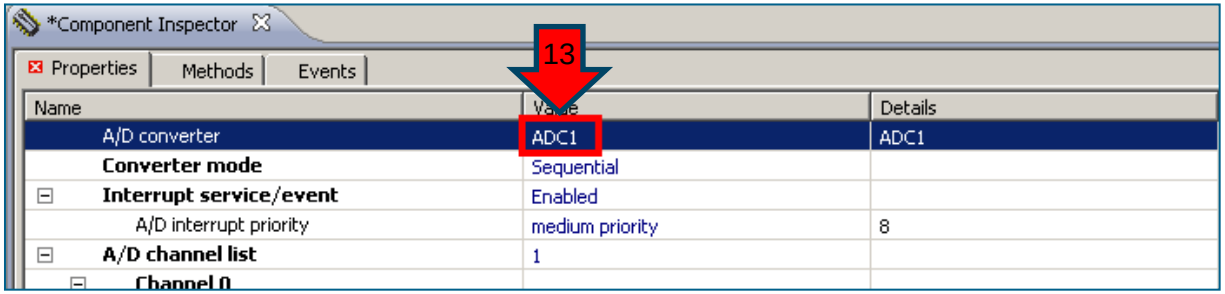
- Right click on the component.
- Select Add to project.

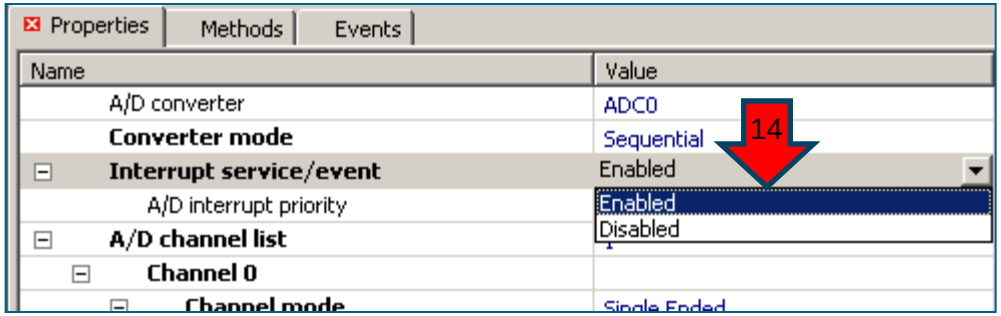


New LLD Driver

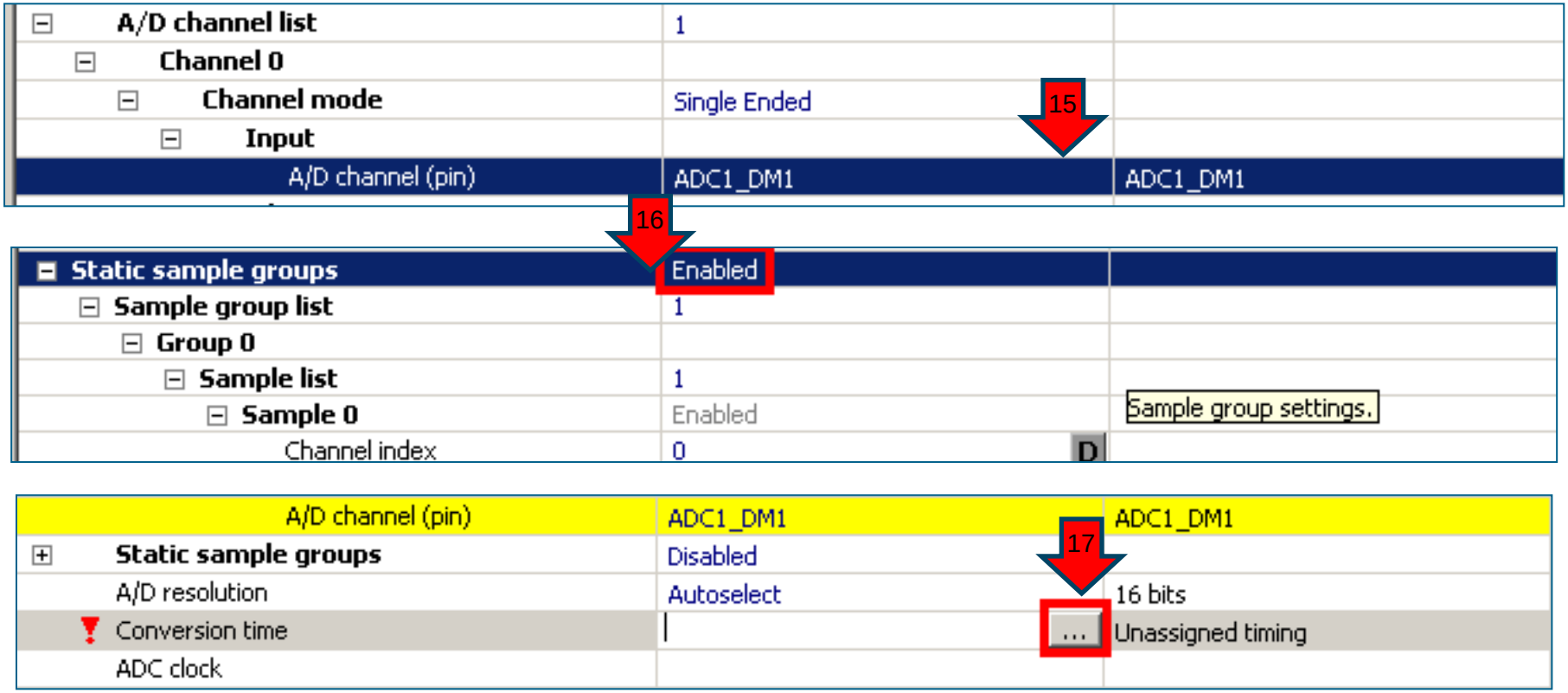
- Double click on ADC_LDD.
- Select ADC1.
- Enable Interrupt service.



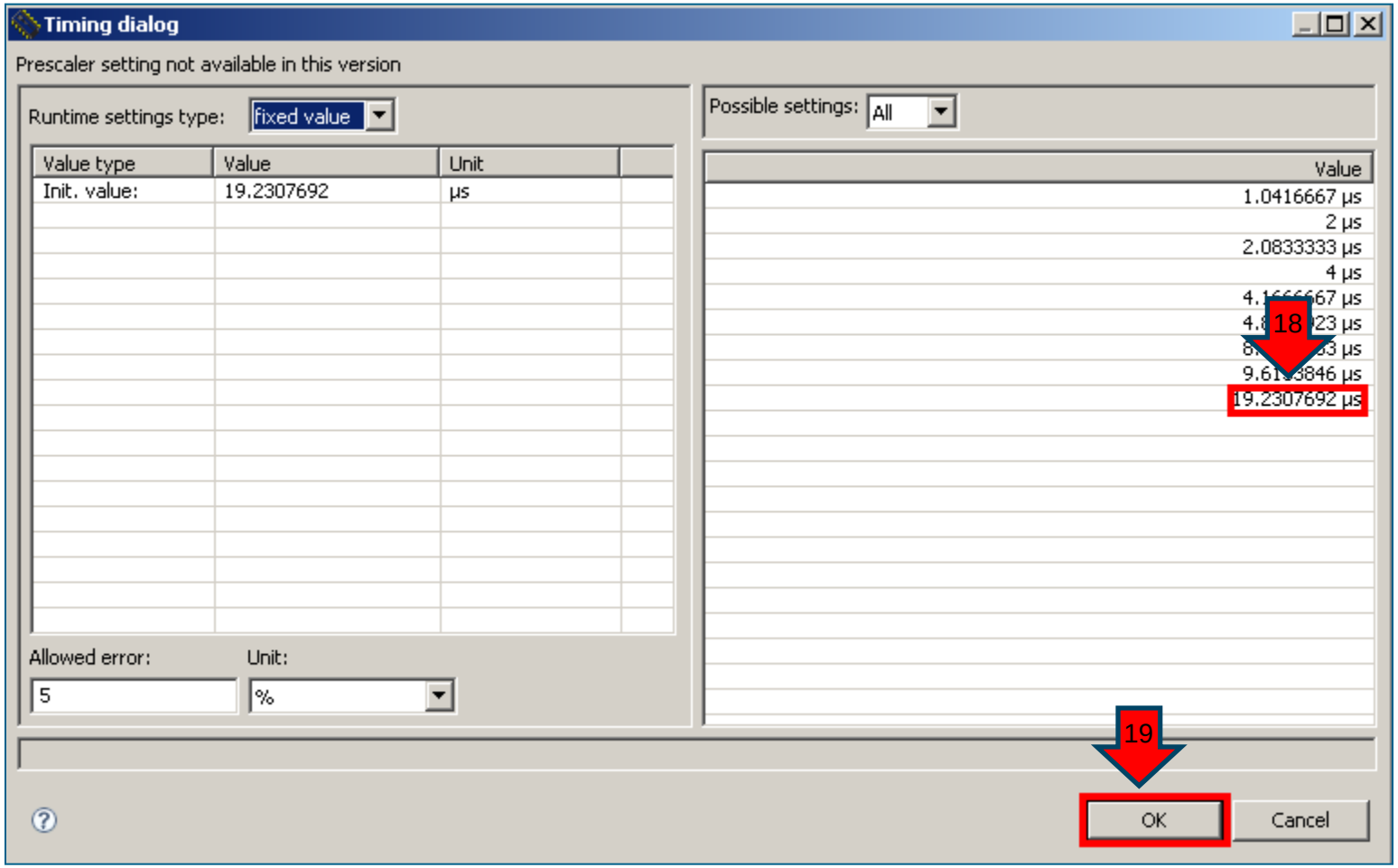




- Select **ADC1_DM1** Channel.
- Enable Static sample groups.
- Open Conversion Time Window.



- Select **19.23** us.



New LLD Driver

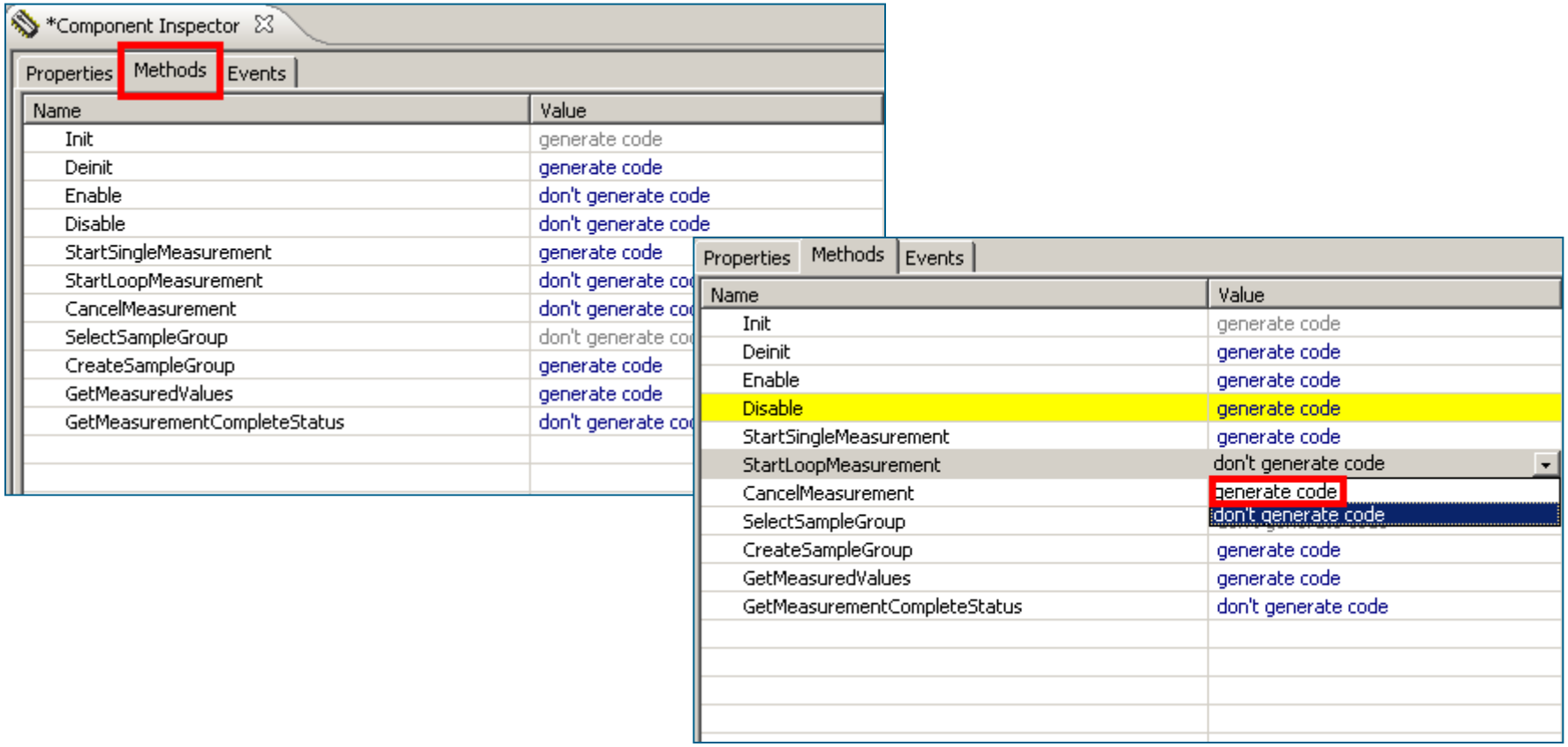
- ADC LLD Driver is configured.

Properties | Methods | Events

Name	Value	Details
A/D converter	ADC1	ADC1
Converter mode	Sequential	
Interrupt service/event	Enabled	
A/D interrupt priority	medium priority	8
A/D channel list	1	
Channel 0		
Channel mode	Single Ended	
Input		
A/D channel (pin)	ADC1_DM1	ADC1_DM1
Static sample groups	Enabled	
Sample group list	1	
Group 0		
Sample list	1	
Sample 0	Enabled	
Channel index	0 D	
A/D resolution	Autoselect	16 bits
Conversion time	4µs	4.167 µs
ADC clock	5.999 MHz (166.667 ns)	Clock conf. 0: 5.999 MHz (166.667 ns)
Single conversion time - Single-ended	10.104 us	Clock conf. 0: 10.104 us
Single conversion time - Differential	11.604 us	Clock conf. 0: 11.604 us
Additional conversion time - Single-ended	4.166 us	Clock conf. 0: 4.166 us
Additional conversion time - Differential	5.666 us	Clock conf. 0: 5.666 us
Result type	unsigned 16 bits, right justified	
Initialization		This property allows to select one of result
Enabled in init. code	yes	*Description for the current value* (unsigned
Event mask		
OnMeasurementComplete	Enabled	

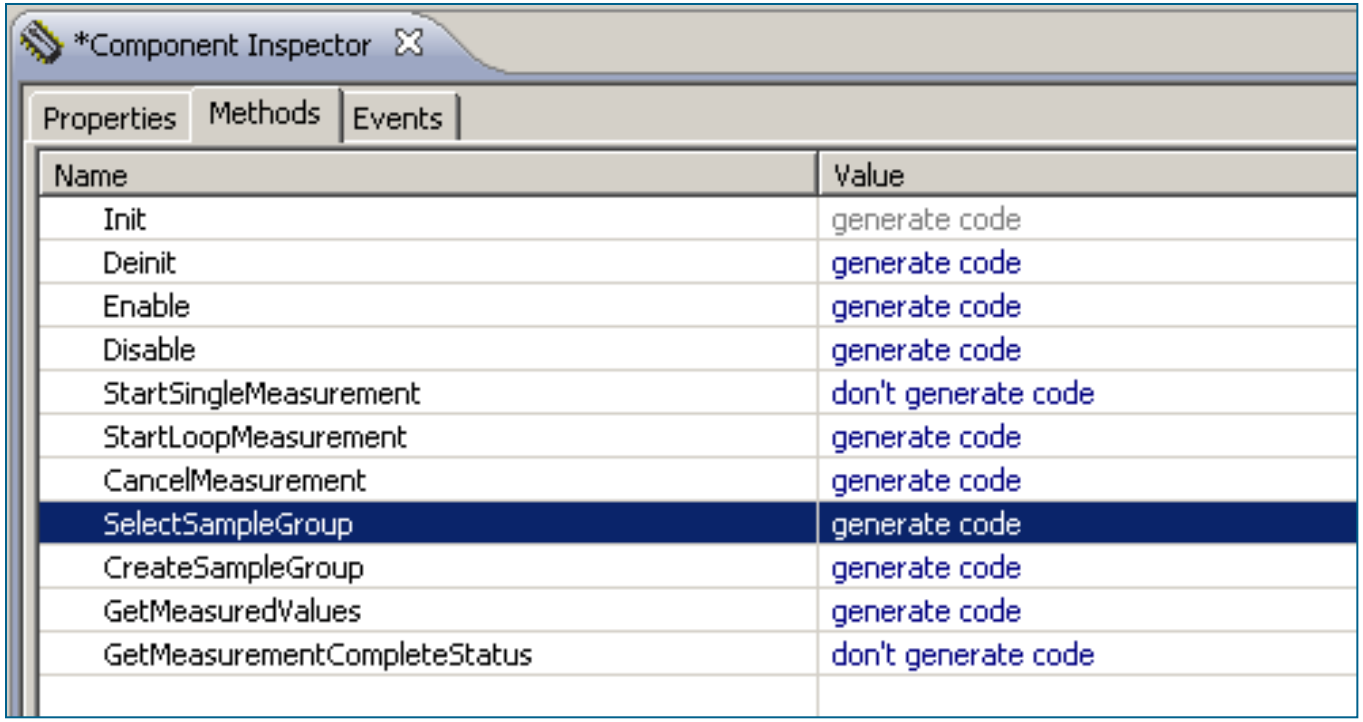
New LLD Driver

- Click Methods Tab.
- Click to generate code for methods.



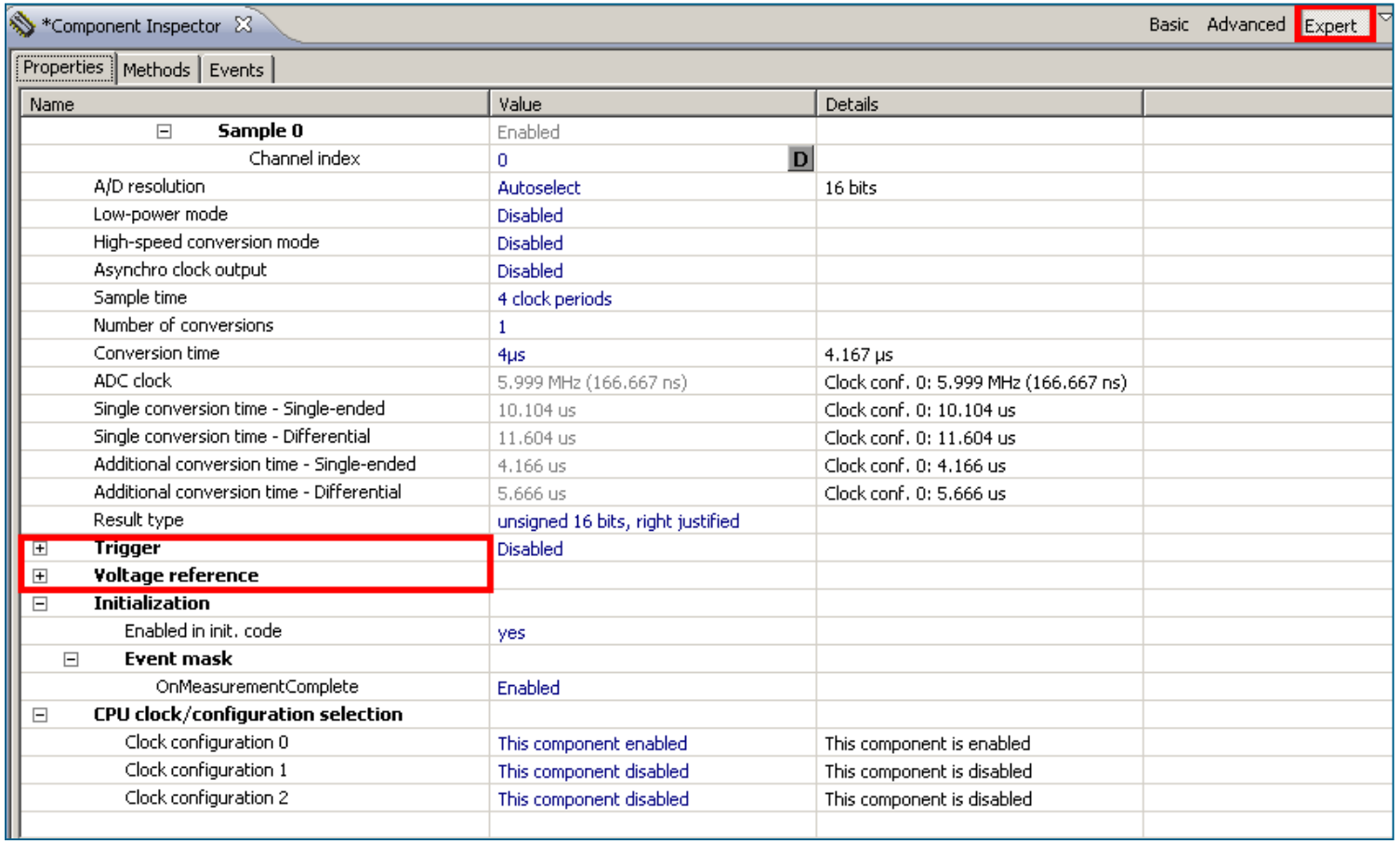
New LLD Driver

- Set ‘generate code’ for the next Methods:



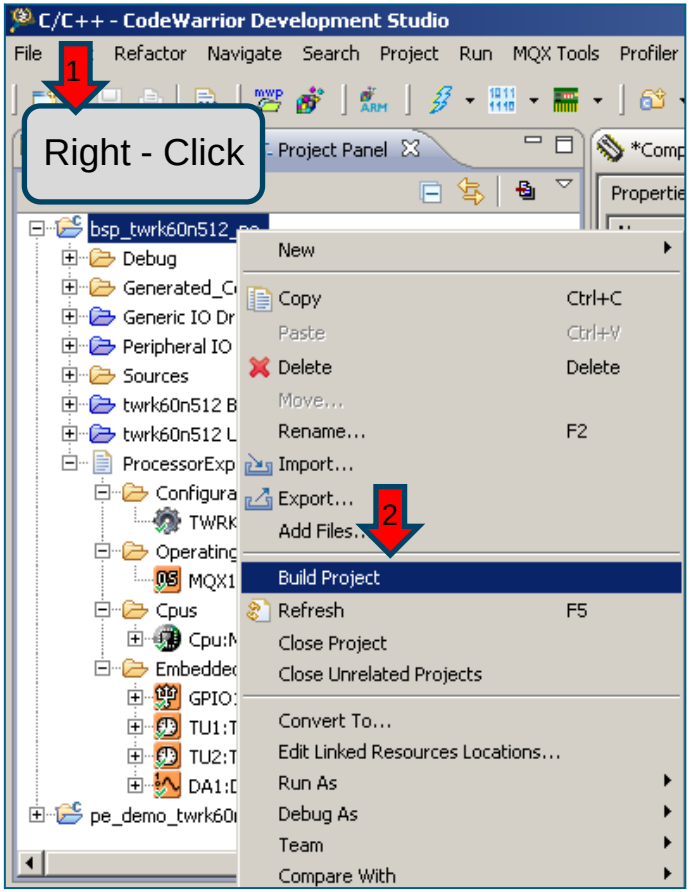
New LLD Driver

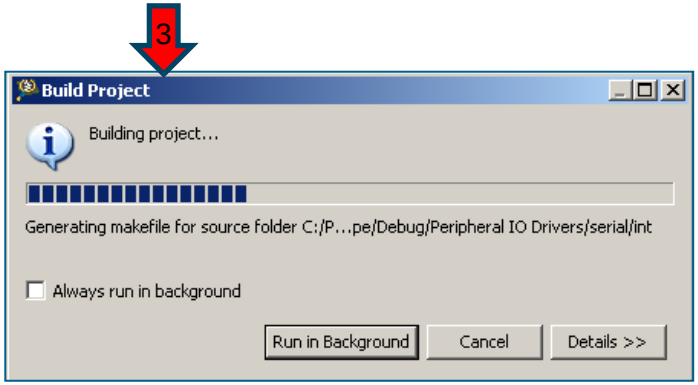
- You can configure more parameters of the components by selecting the Expert View.



Build PE BSP

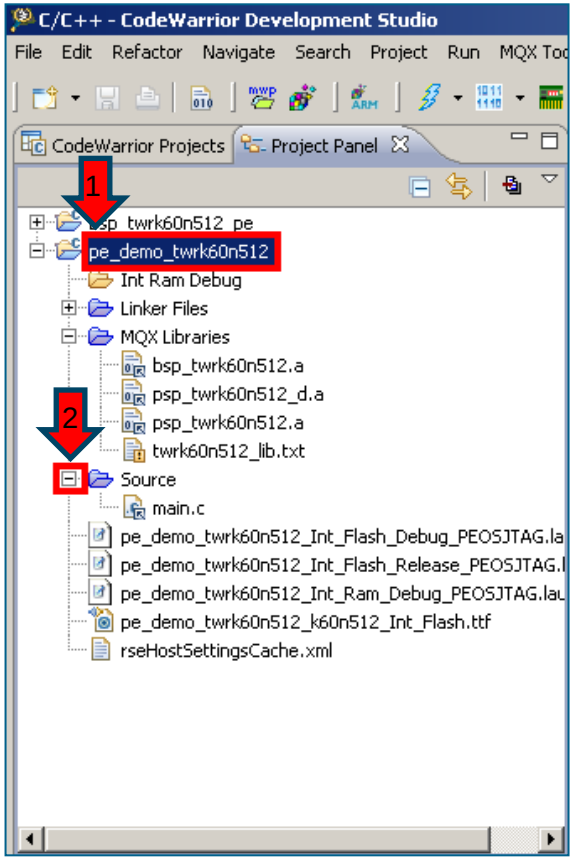
- Right-Click on the Project Explorer **bsp_twrk60n512_pe** and Build Project.

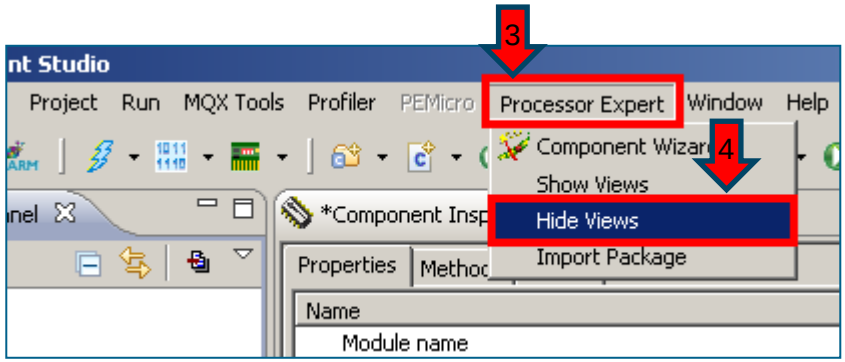




New LDD driver

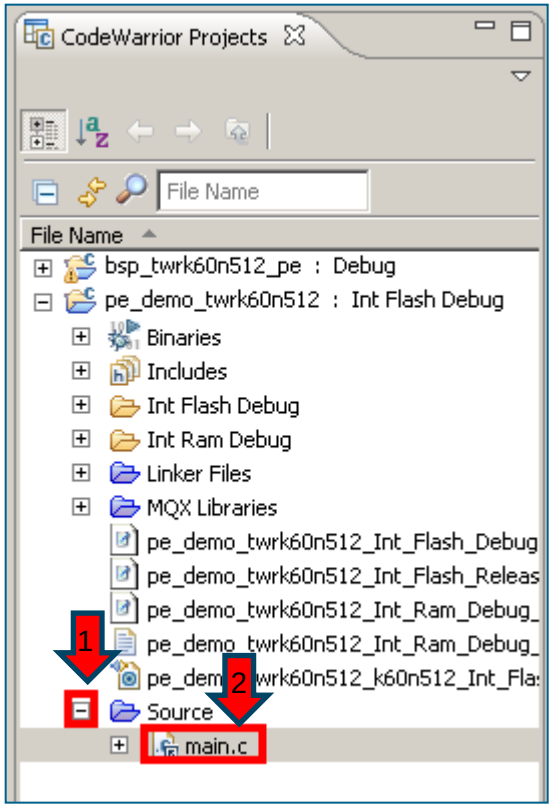
- Expand **pe_demo_twrk60n512** project view.
- Hide Processor Expert View.

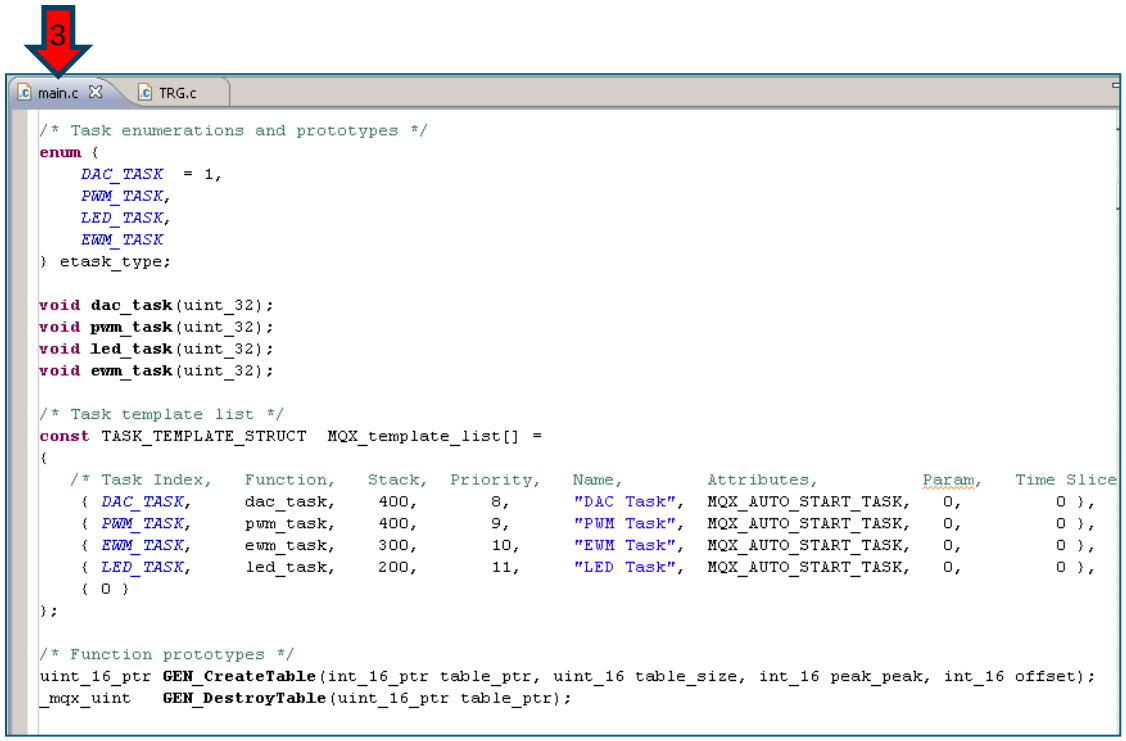




New LLD Driver

- Double click in **main.c** to view code.






```
/* Task enumerations and prototypes */
enum {
    DAC_TASK  = 1,
    PWM_TASK,
    LED_TASK,
    EWM_TASK
} etask_type;

void dac_task(uint_32);
void pwm_task(uint_32);
void led_task(uint_32);
void ewm_task(uint_32);

/* Task template list */
const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
   /* Task Index,   Function,   Stack,  Priority,   Name,        Attributes,          Param,   Time Slice
    { DAC_TASK,     dac_task,    400,       8,       "DAC Task", MQX_AUTO_START_TASK,  0,          0 },
    { PWM_TASK,     pwm_task,    400,       9,       "PWM Task", MQX_AUTO_START_TASK,  0,          0 },
    { EWM_TASK,     ewm_task,    300,       10,      "EWM Task", MQX_AUTO_START_TASK,  0,          0 },
    { LED_TASK,     led_task,    200,       11,      "LED Task", MQX_AUTO_START_TASK,  0,          0 },
    { 0 }
};

/* Function prototypes */
uint_16_ptr GEN_CreateTable(int_16_ptr table_ptr, uint_16 table_size, int_16 peak_peak, int_16 offset);
_mqx_uint   GEN_DestroyTable(uint_16_ptr table_ptr);
```

New LLD Driver

- Add new task : ADC.

```
/* Task enumerations and prototypes */
enum {
    DAC_TASK  = 1,
    PWM_TASK,
    LED_TASK,
    EWM_TASK,
    ADC_TASK
} etask_type;

void dac_task(uint_32);
void pwm_task(uint_32);
void led_task(uint_32);
void ewm_task(uint_32);
void adc_task(uint_32);

/* Task template list */
const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
   /* Task Index,   Function,   Stack,  Priority,   Name,         Attributes,           Param,   Time Slice
    { DAC_TASK,     dac_task,    400,       8,      "DAC Task",  MQX_AUTO_START_TASK,   0,          0 },
    { PWM_TASK,     pwm_task,    400,       9,      "PWM Task",  MQX_AUTO_START_TASK,   0,          0 },
    { EWM_TASK,     ewm_task,    300,      10,      "EWM Task",  MQX_AUTO_START_TASK,   0,          0 },
    { LED_TASK,     led_task,    200,      11,      "LED Task",  MQX_AUTO_START_TASK,   0,          0 },
    { ADC_TASK,     adc_task,    200,      12,      "ADC Task",  MQX_AUTO_START_TASK,   0,          0 },
    { 0 }
};
```

- Add Task function and code.

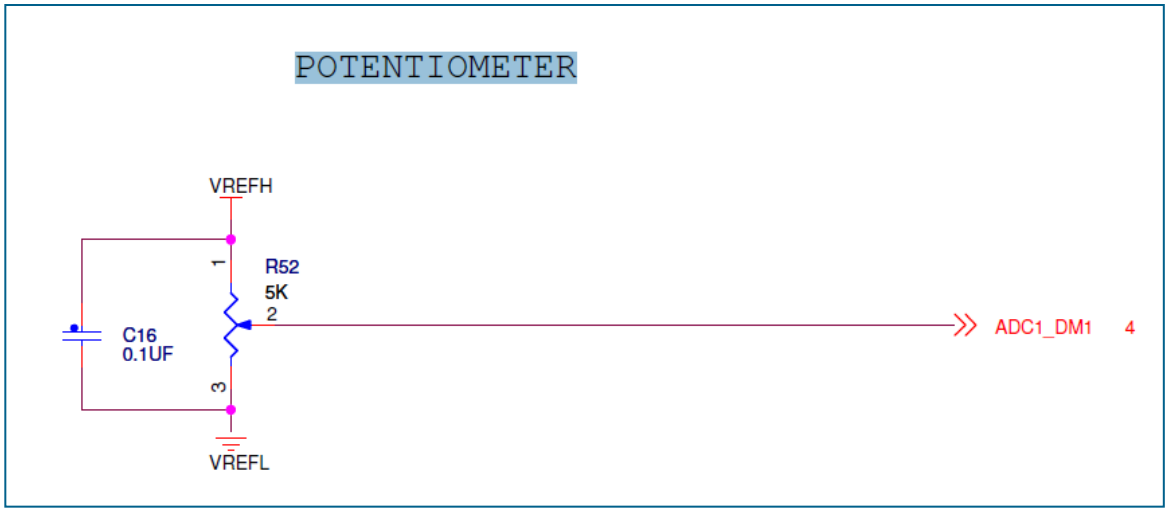
```
#define SAMPLE_GROUP_SIZE 1U
volatile AD1_TResultData MeasuredValues[SAMPLE_GROUP_SIZE];
LDD_TDeviceData *MyADCPtr;
LDD_TError Error;

void adc_task
    (
       uint_32 initial_data
    )
{
     MyADCPtr = AD1_Init((LDD_TUserData *)NULL);          /* Initialize the device */
     Error = AD1_SelectSampleGroup(MyADCPtr, 0U);         /* Select sample group 0 */
     Error = AD1_StartLoopMeasurement(MyADCPtr);          /* Start continuous measurement */
     Error = AD1_Enable(MyADCPtr);
     while(1)
     {
     /* Suspend task for 100ms */

          if(MeasuredValues[0]>2000)GPIO1_ToggleFieldBits(LED_DeviceData, LED4, 1);
          _time_delay(200);

     }
}
```

- Add **ADC1** Event function code.

8

```
void AD1_OnMeasurementComplete(LDD_TUserData *UserDataPtr)
{
  Error = AD1_GetMeasuredValues(MyADCPtr, (LDD_TData *)&MeasuredValues);  /* Read measured values */
}
/* EOF */
```

- ADC1 channel is connected to TWR-K60N512 Potentiometer.



New LLD Driver

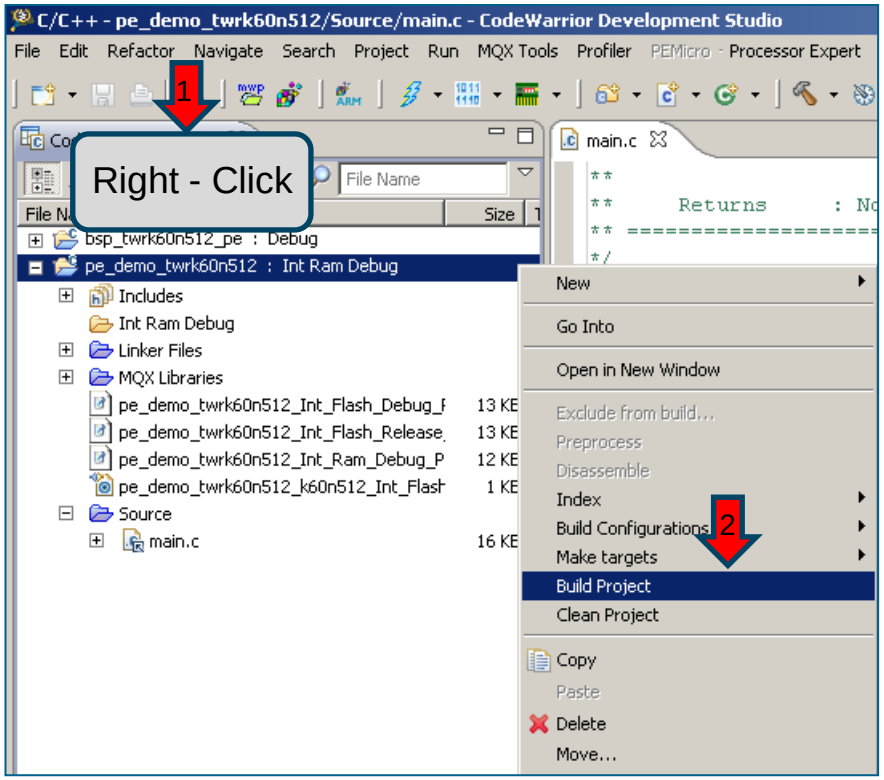
- When ADC value is greater than 20000, LED4 (Blue) toggles.

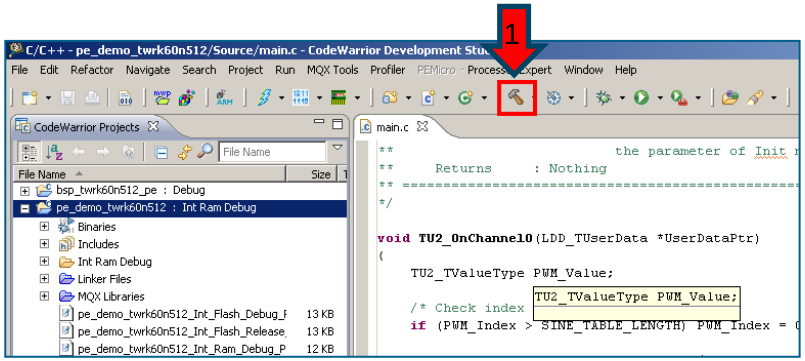
```
while(1)
{
/* Suspend task for 100ms */

    if(MeasuredValues[0]>2000)GPIO1_ToggleFieldBits(LED_DeviceData, LED4, 1);
     _time_delay(200);

}
```

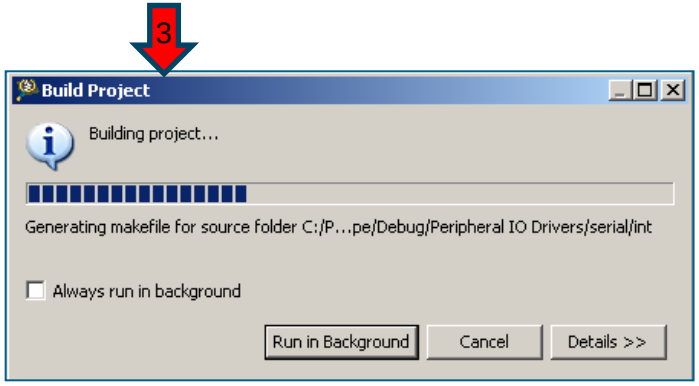
- Moving potentiometer R52 can start/stop LED4 toggle.

Build PE Demo

- Right-Click on the Project Explorer **pe_demo_twrk60n512** and Build the Project or click on the icon. 

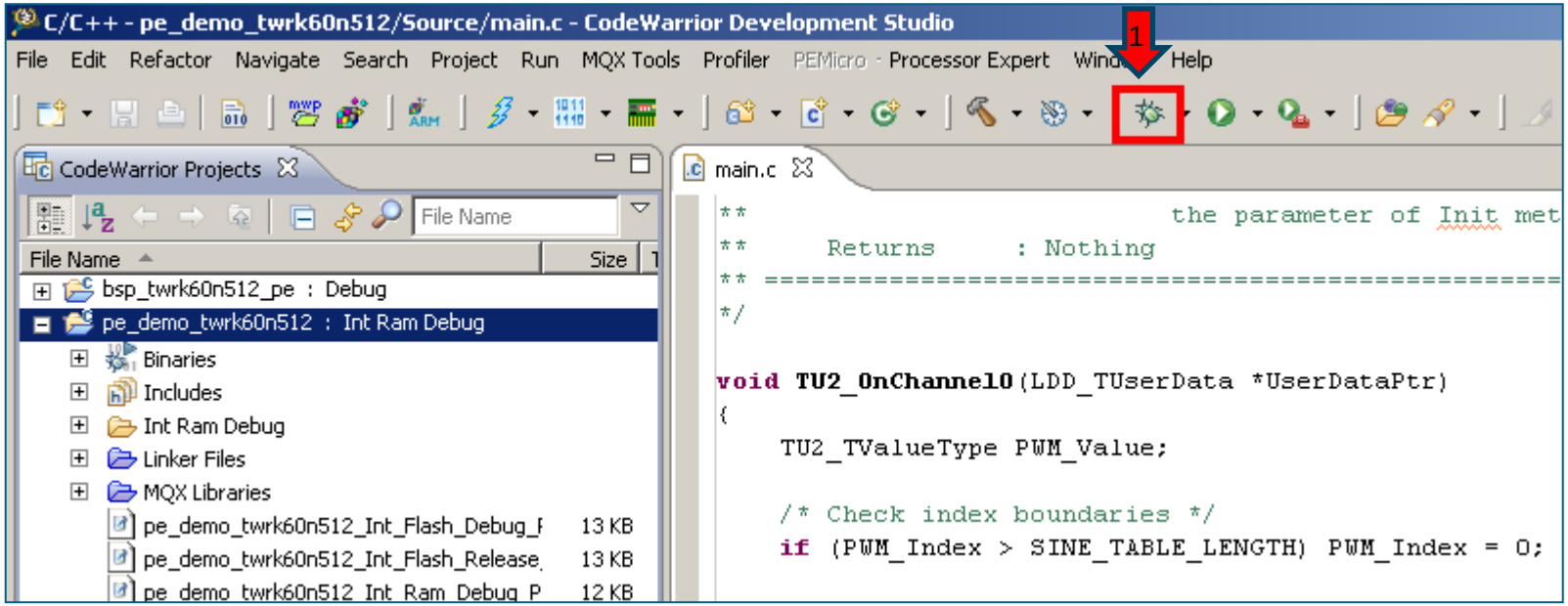






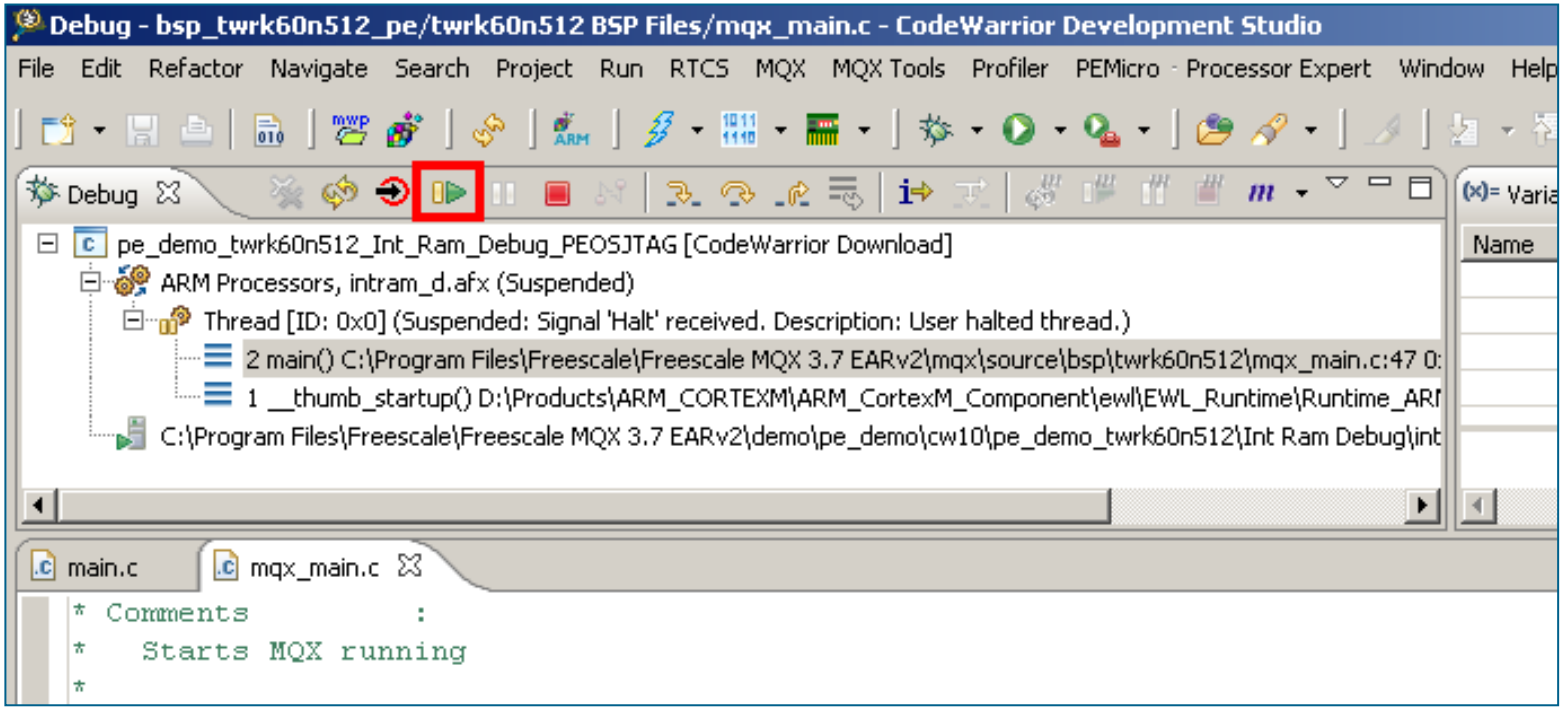
Run MQX PE Demo

- Click Debug icon.



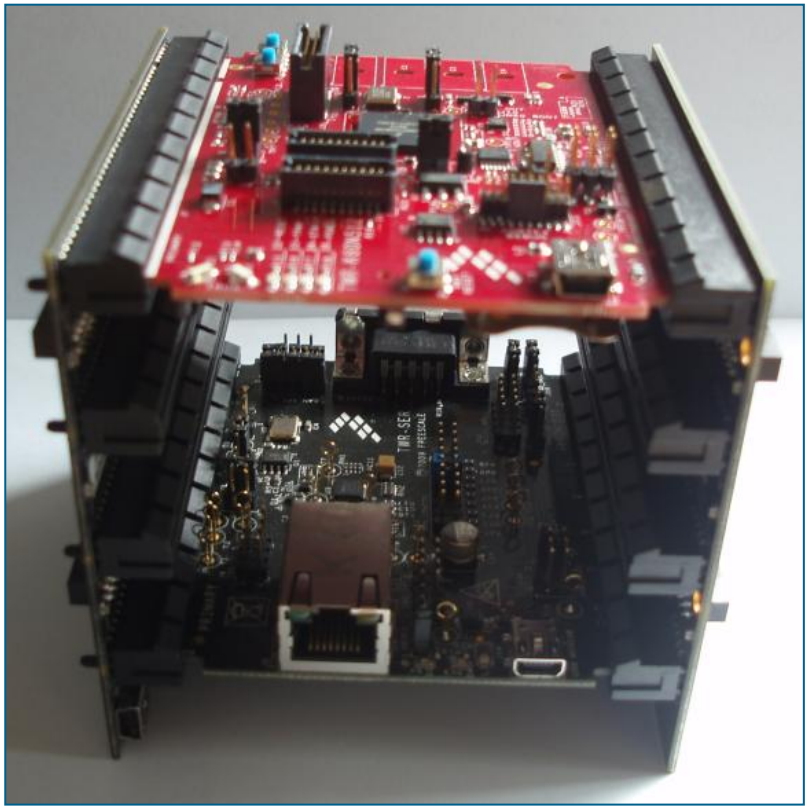
Run MQX PE Demo

- Click Resume (F8).



New LLD Driver

- Test the new functionality in the application and the new LDD driver.



- Use this link to access Freescale Infocenter:
- freescale.com/infocenter/index.jsp