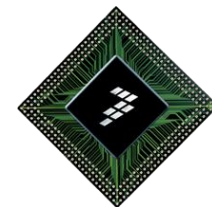


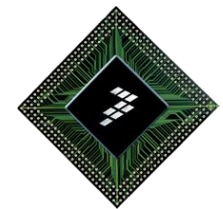
Last update: August 2014
Document Number: MQXCWPP

CW for Microcontrollers v10 and MQX™ RTOS



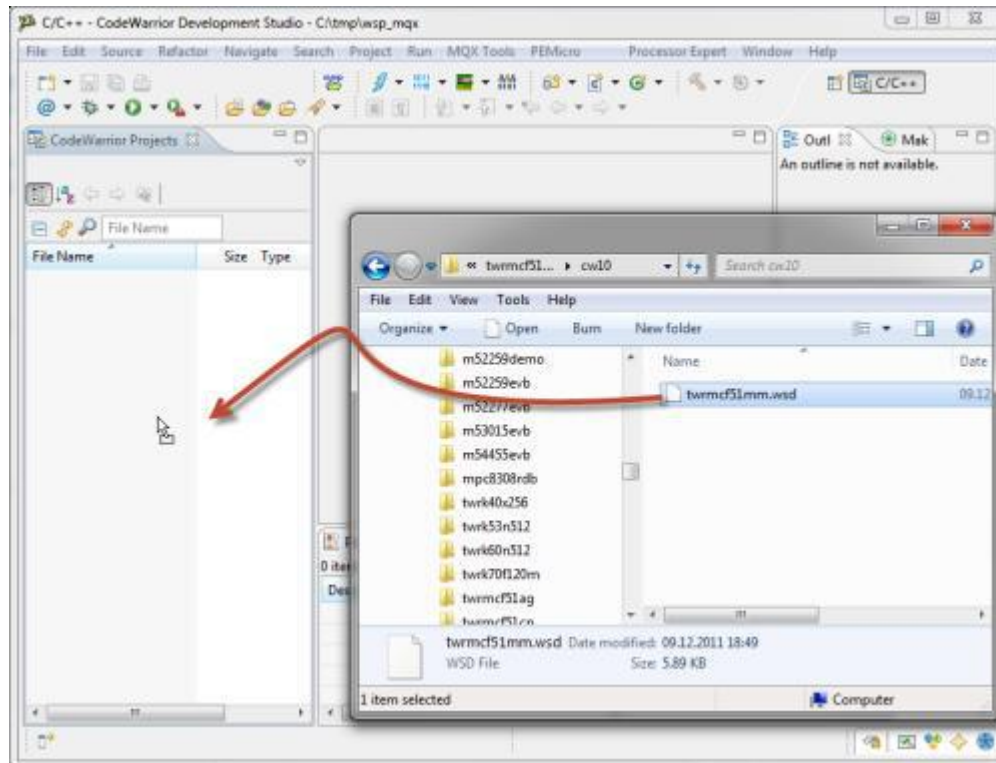
- ▶ Import MQX Libraries
- ▶ Build MQX libraries
- ▶ Import and Debug MXQ Hello World Project
- ▶ New MQX project
- ▶ Debugging with J-Link
- ▶ CW10.x, MQX RTOS and Processor Expert
- ▶ CW10.x, MQX RTOS and PE : New LDD driver

Import MQX Libraries

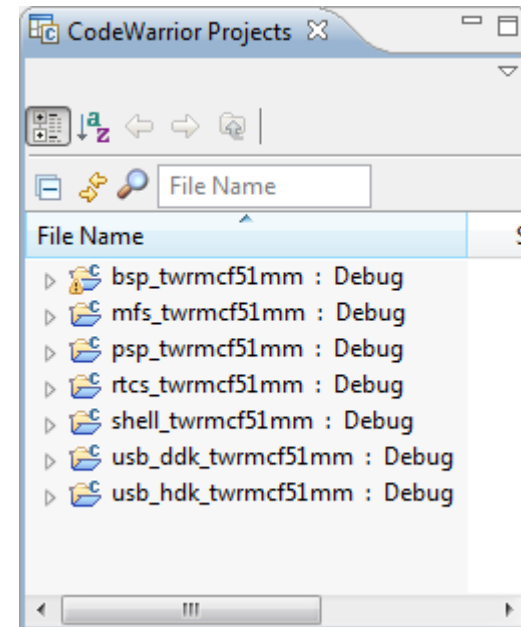


Import MQX Libraries

- ▶ Navigate to `C:\Freescale\Freescale MQXX.X\build\<board_name>\cw10gcc\` and drag `build_libs.wsd` to the CodeWarrior

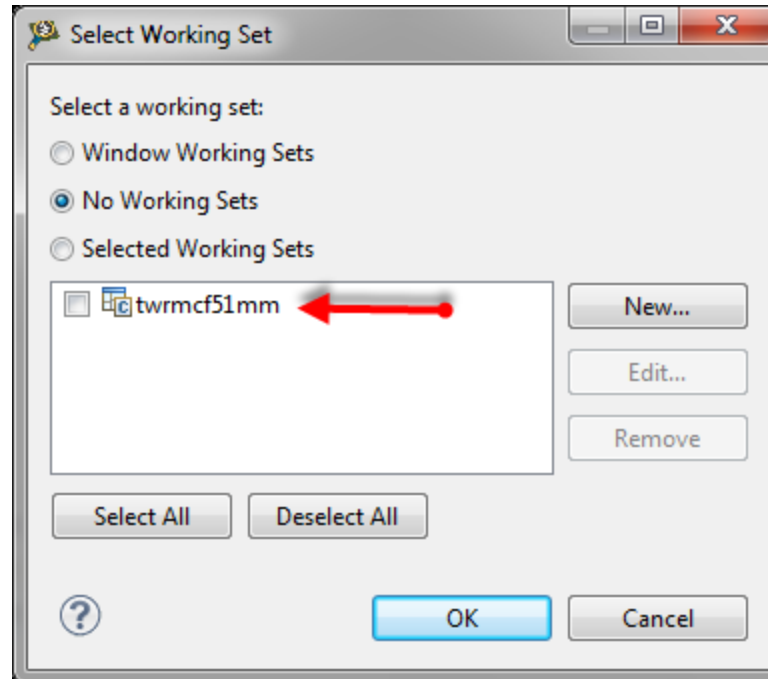


- ▶ All BSP libraries will be loaded to your environment automatically



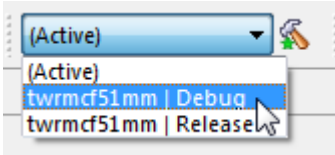
Import MQX libraries

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

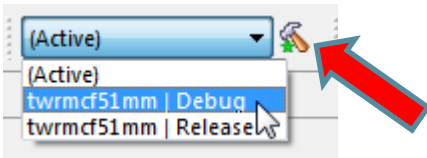


Building MQX Libraries

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



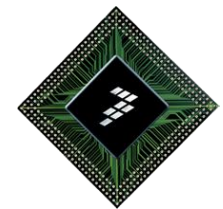
- Hit the icon to build all MQX libraries for a selected working set as shown below:



Note:

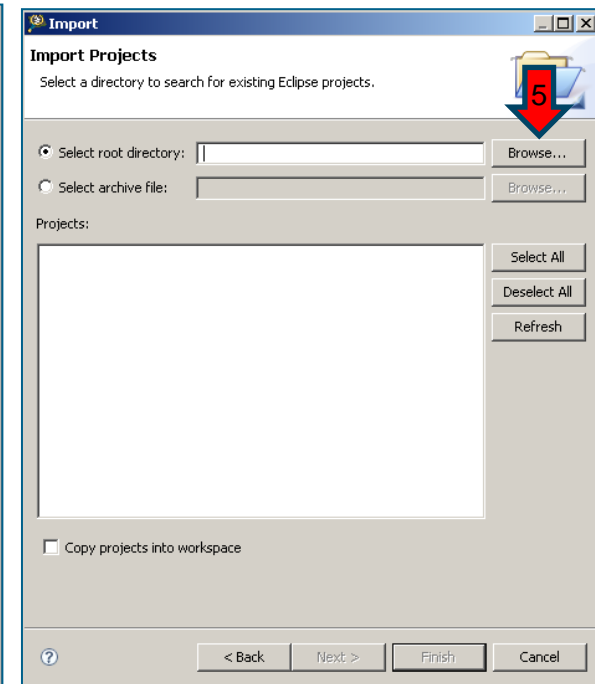
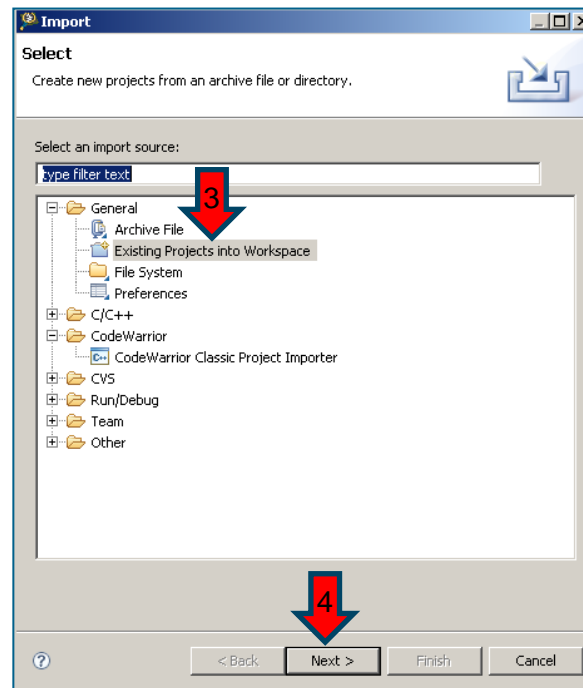
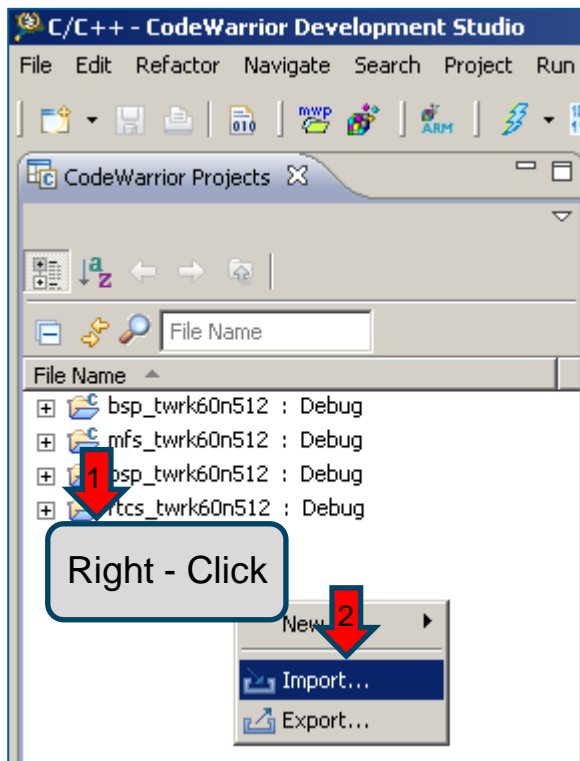
Debug configuration of MQX libraries, workingset, has the compiler optimization set to the lowest level for all imported projects. The **Release** configuration uses the highest possible compiler optimization setting.

Import and Debug MXQ Hello World Project



Import 'Hello World' MQX example

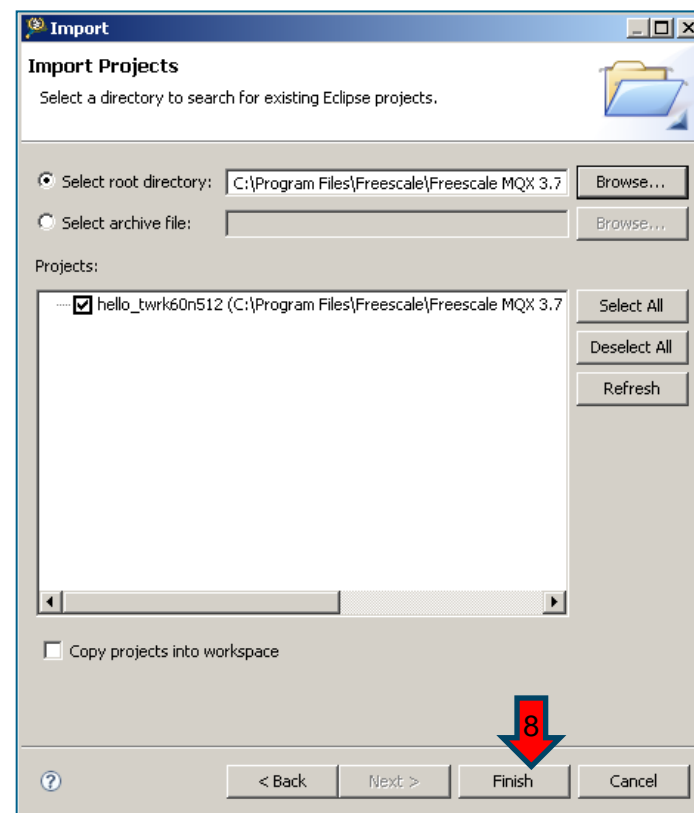
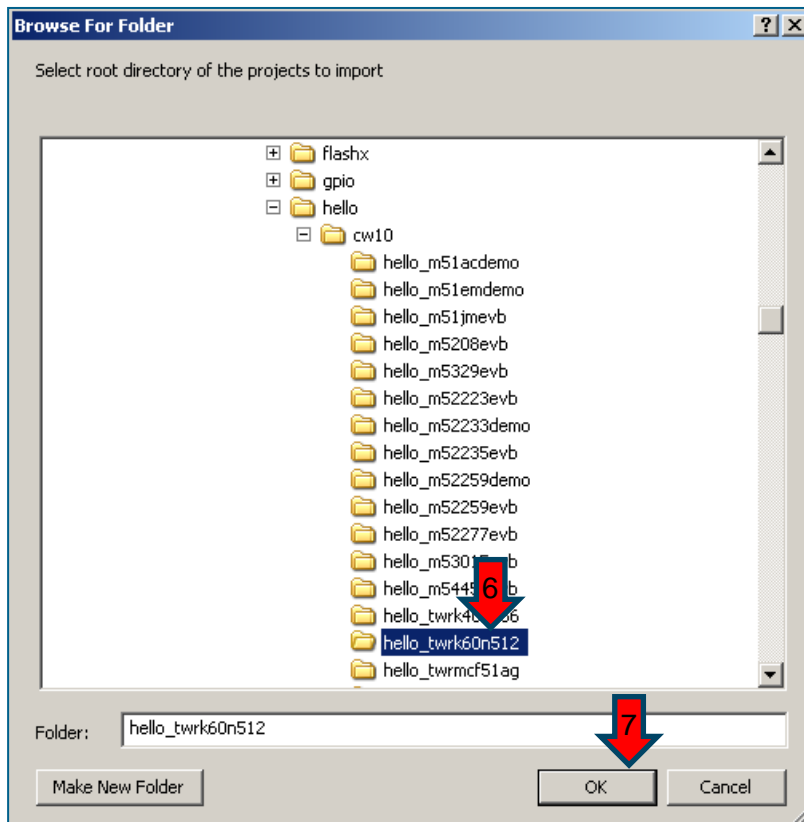
- ▶ Right-Click on Project Explorer and Import.
- ▶ Select Existing Projects into Workspace and Browse.



Import 'Hello World' MQX example

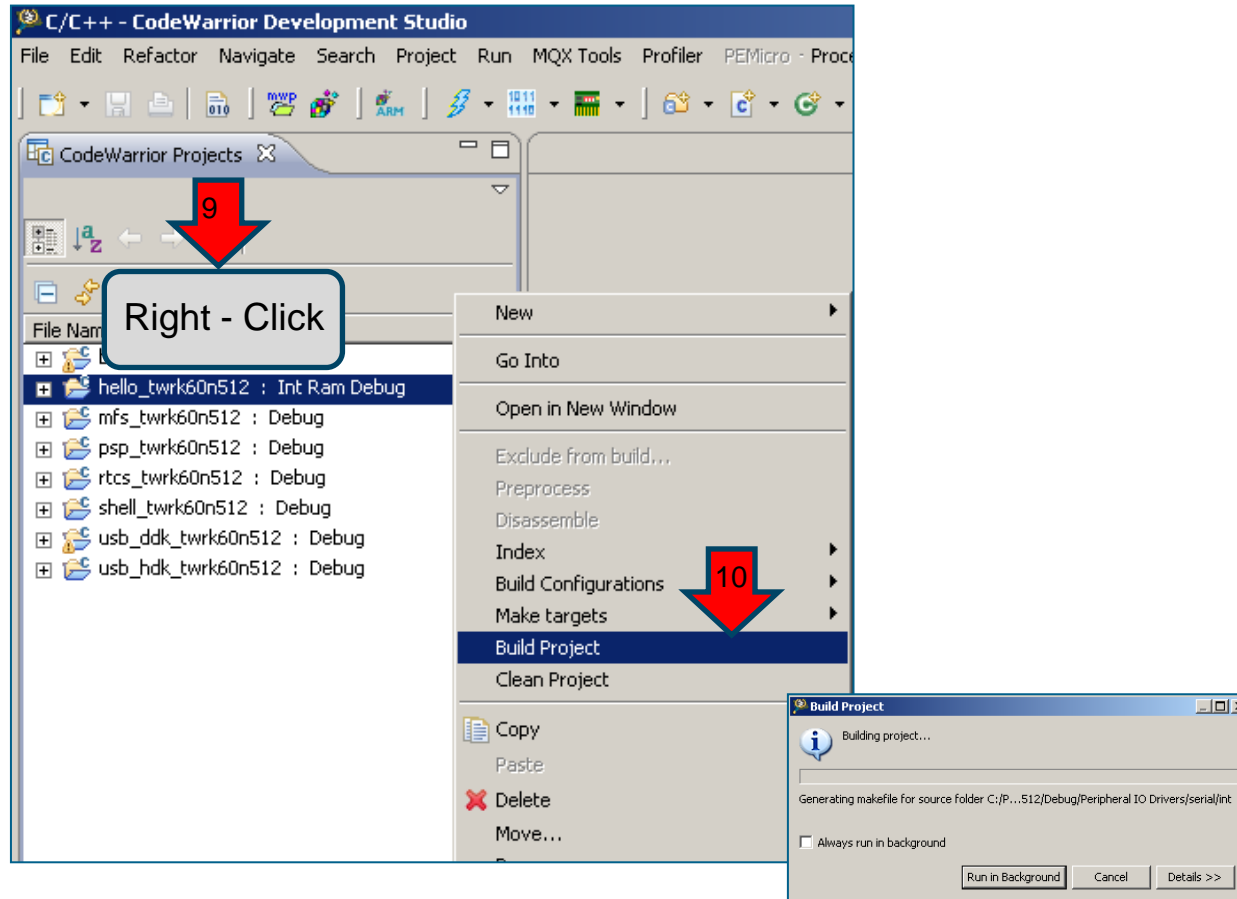
► Select

<install mqx folder>\mqx\examples\hello\build\cw10gcc\hello_twrk60n512



Build 'Hello World' MQX example

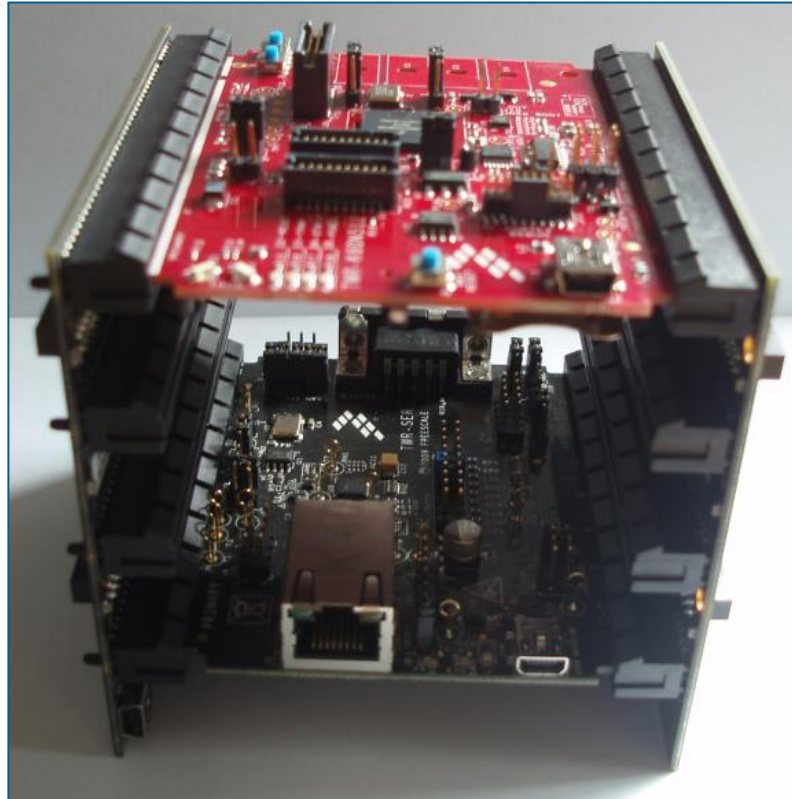
- Right-Click on Project Explorer **hello_twrk60n512** and select Build Project.



Prepare your hardware

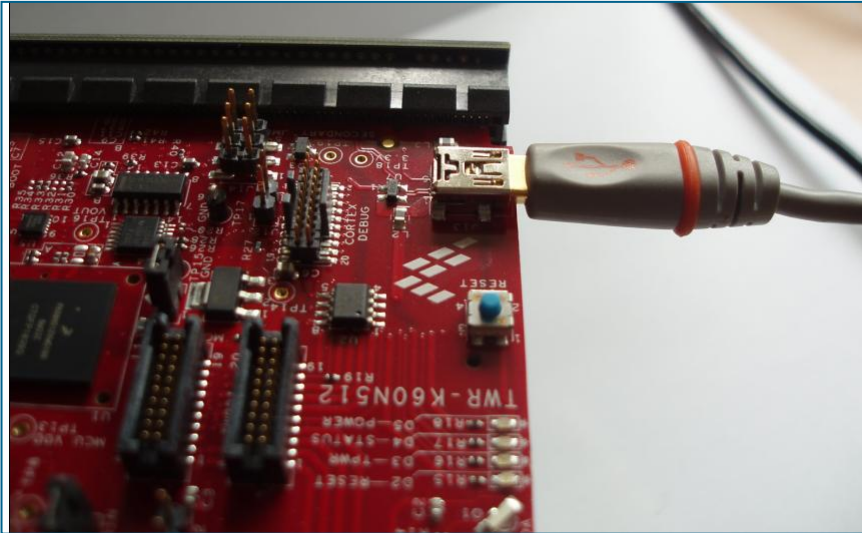
► Prepare your Tower System:

- Connect **TWR-SER** and **TWR-K60N512** to **TWR-ELEV** (primary and secondary).



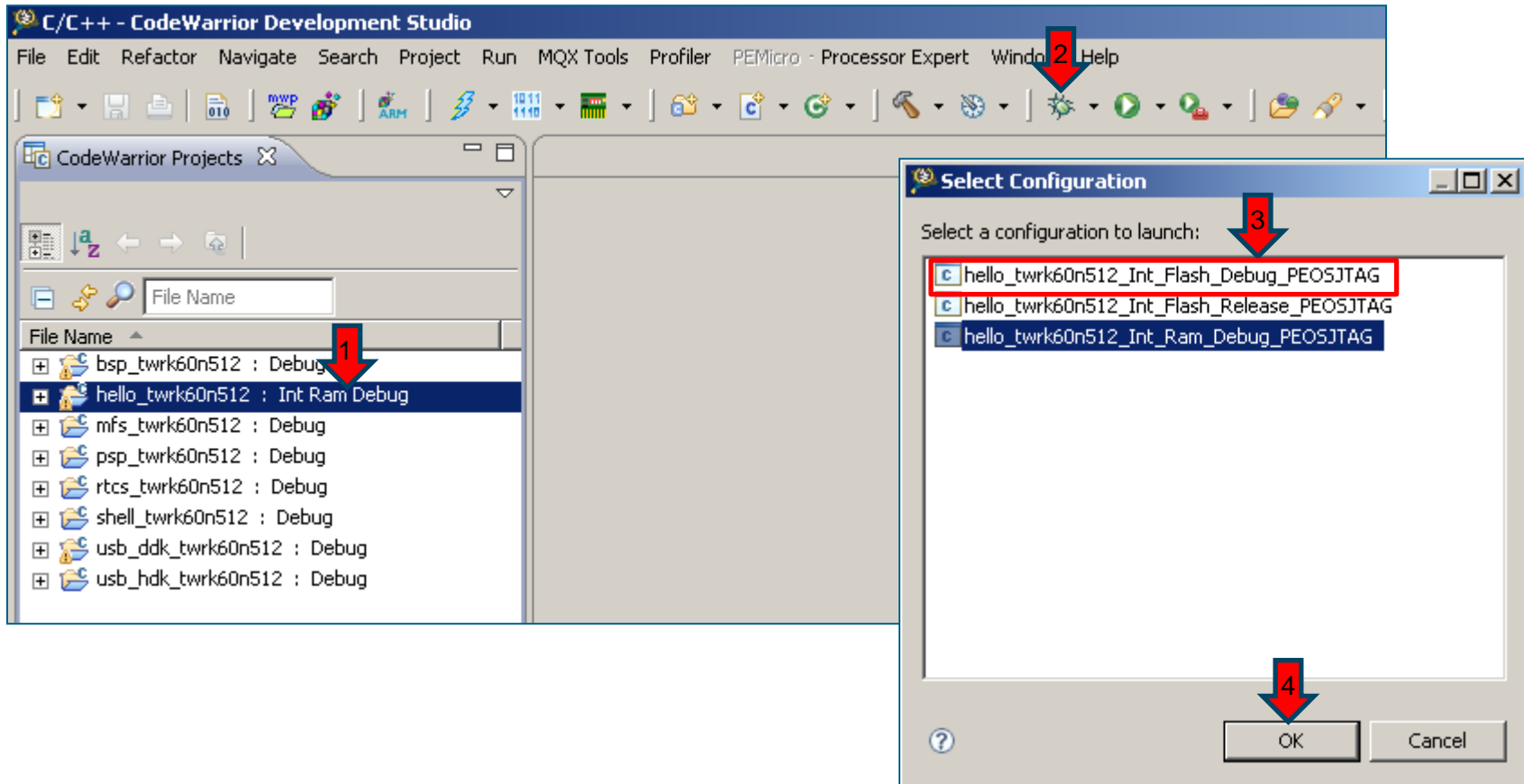
Prepare your hardware

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



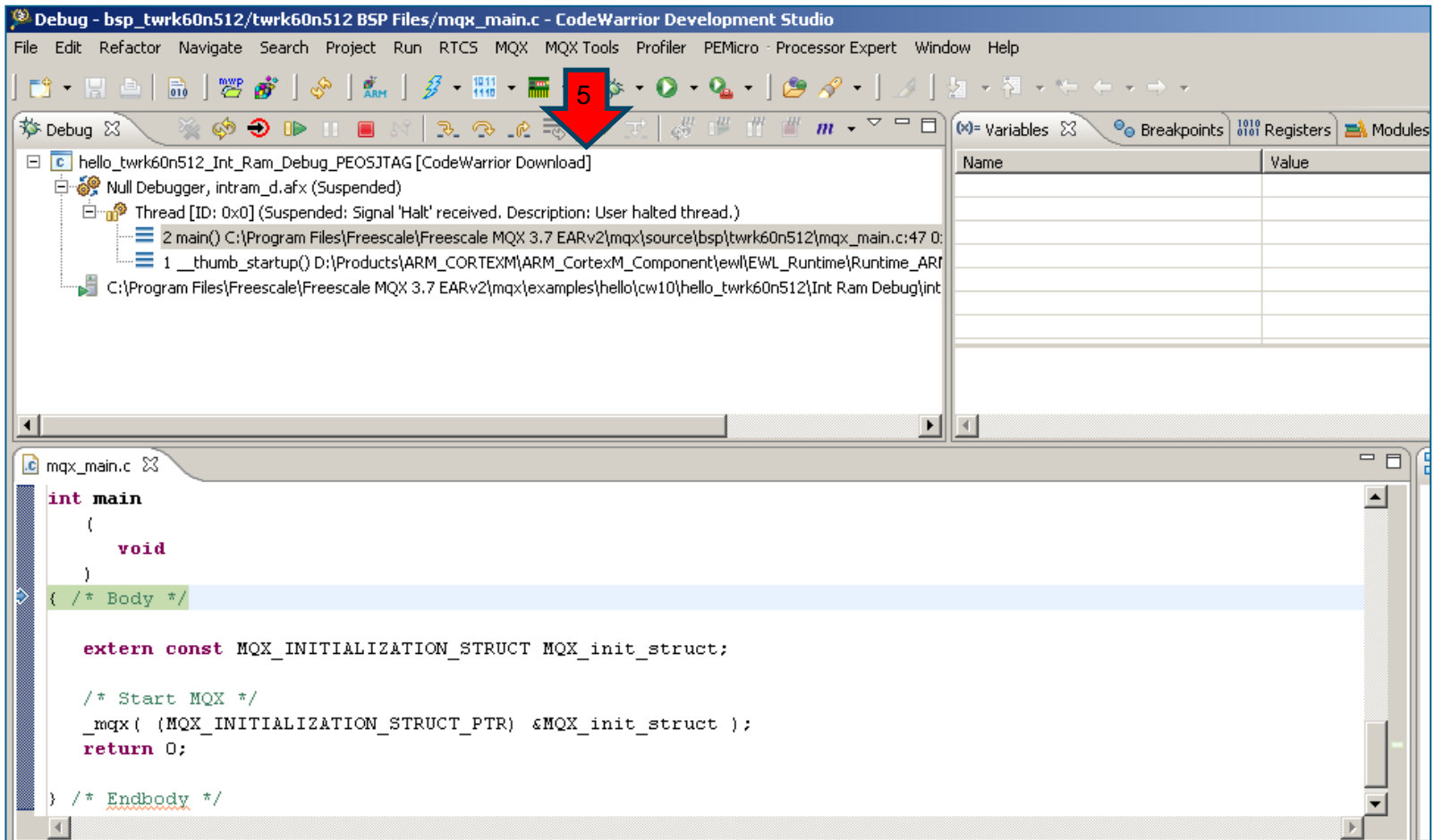
Debug MQX 'Hello World' example

- ▶ Select **hello_twrk60n512** project and Click 'Debug icon.'
- ▶ Select **hello_twrk60n512_Int_Flash_Debug_PEOSJTAG** connection.



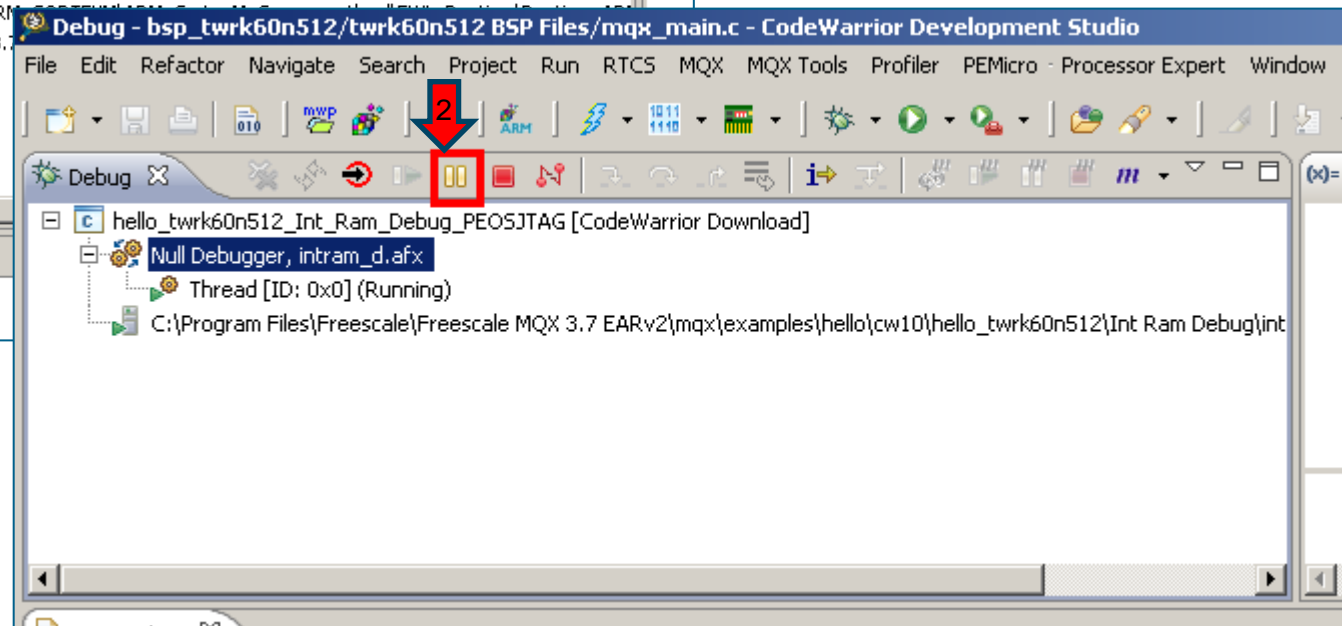
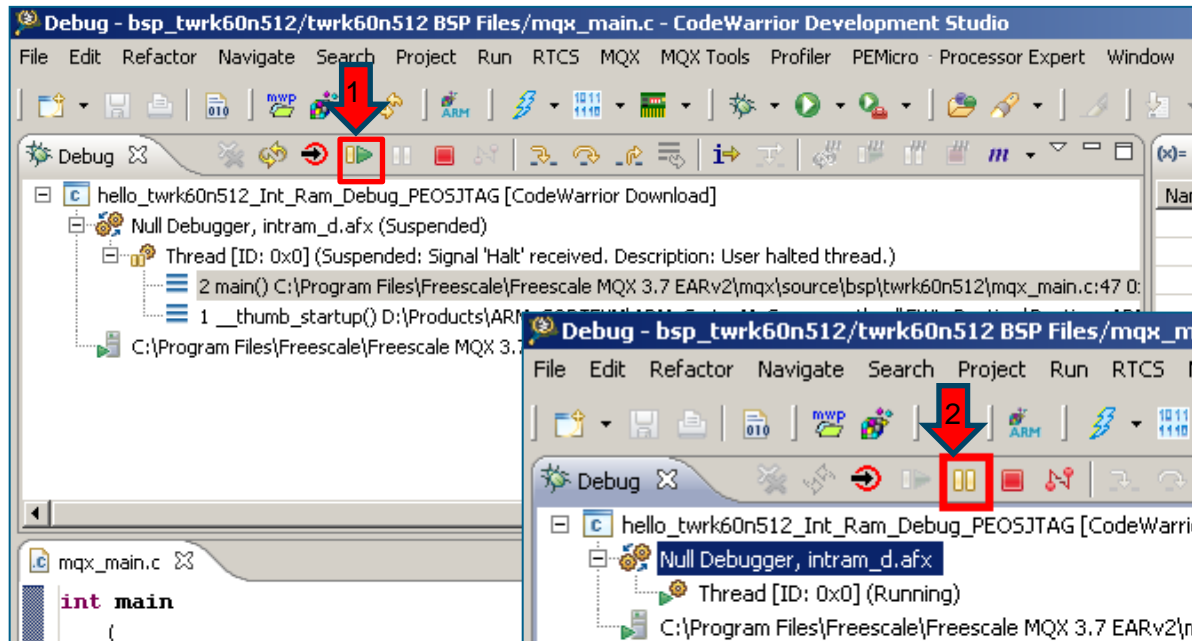
Debug MQX 'Hello World' example

- You are ready to Run and Debug the project.



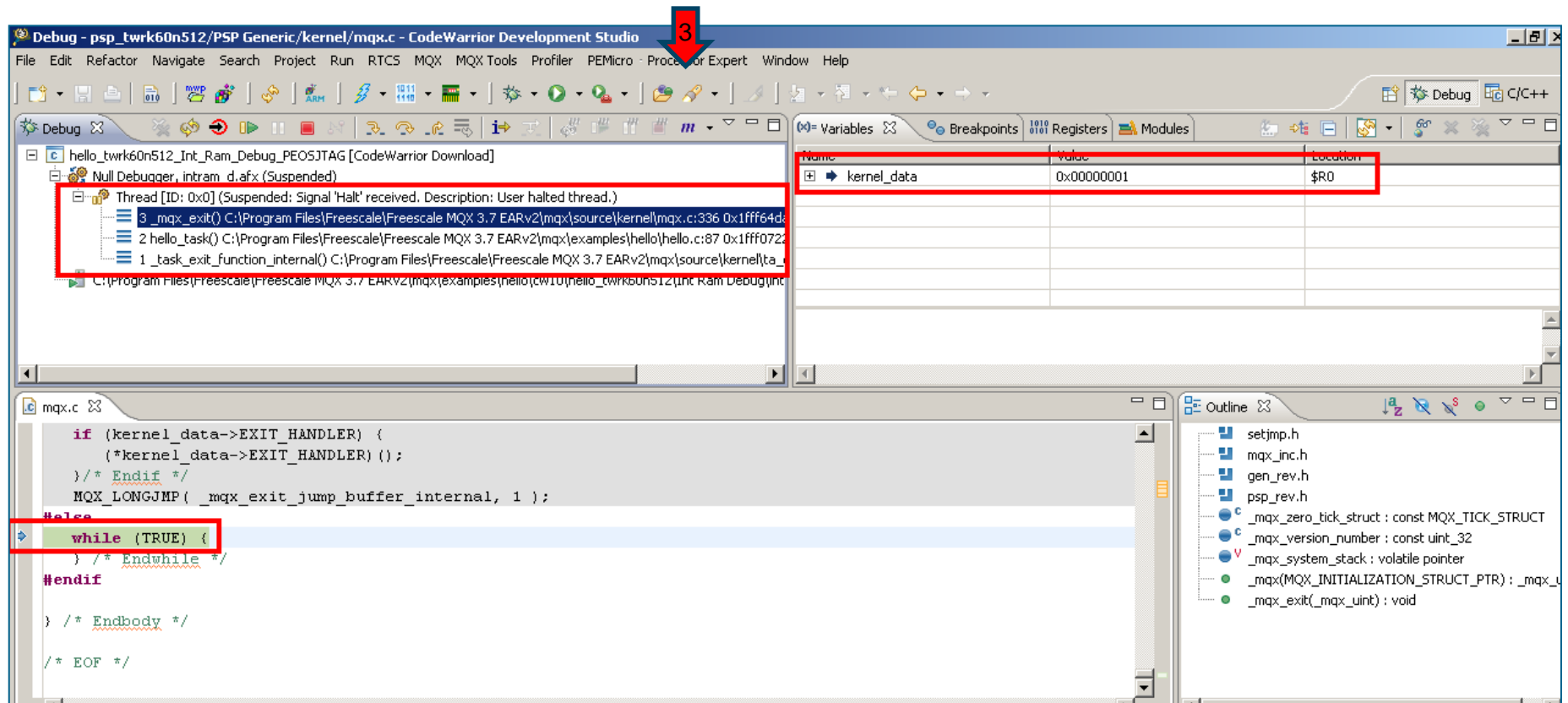
Run MQX 'Hello World' example

- ▶ Execute the code 'Resume' icon and 'Pause' execution.



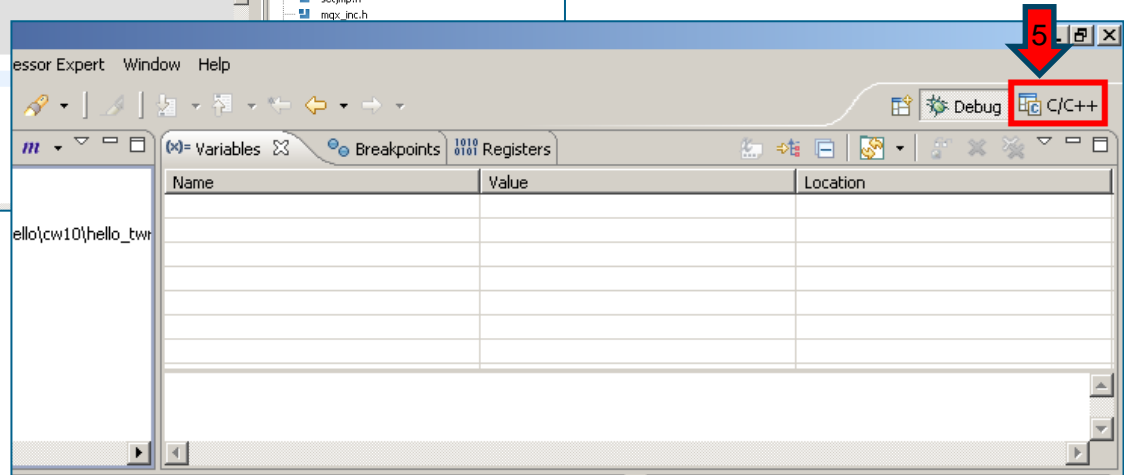
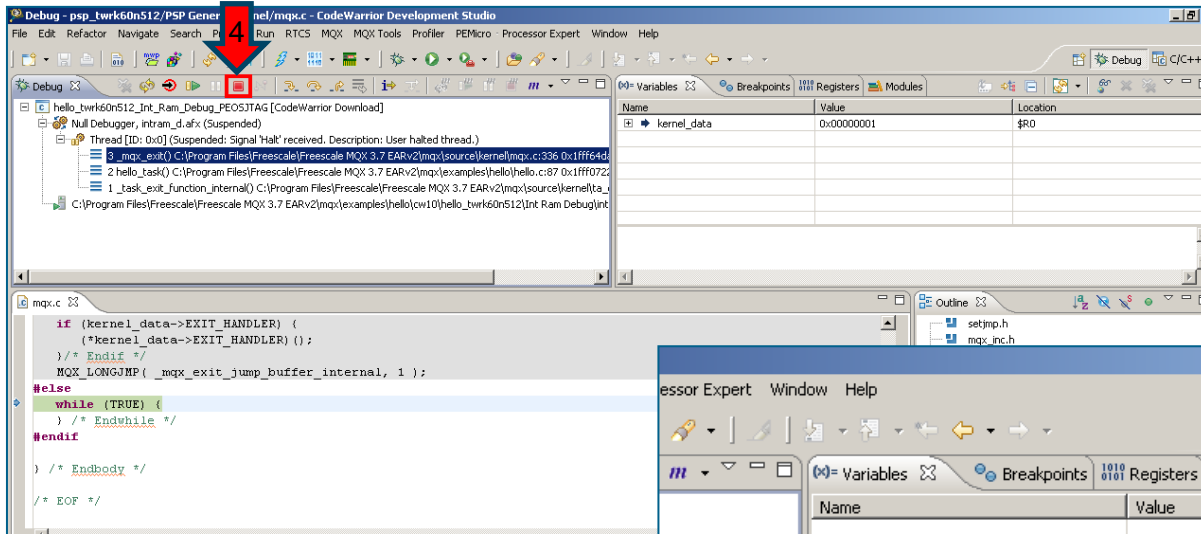
Run MQX 'Hello World' example

- You can explore the Debugging Eclipse perspective.

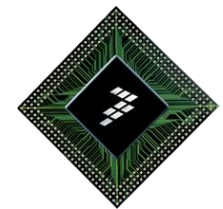


Run MQX 'Hello World' example

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

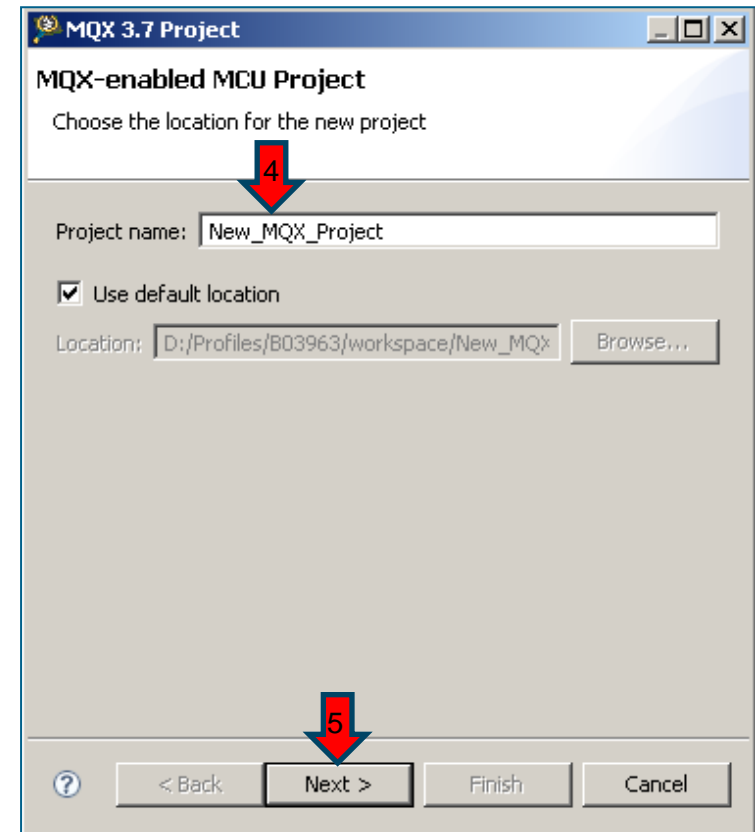
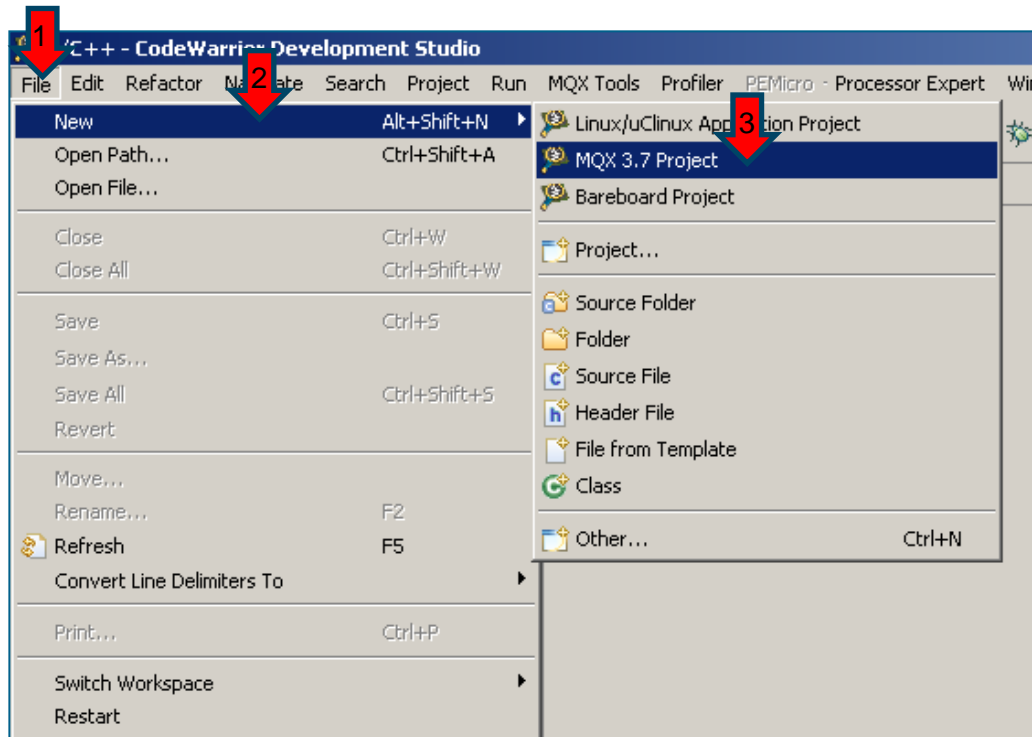


New MQX project



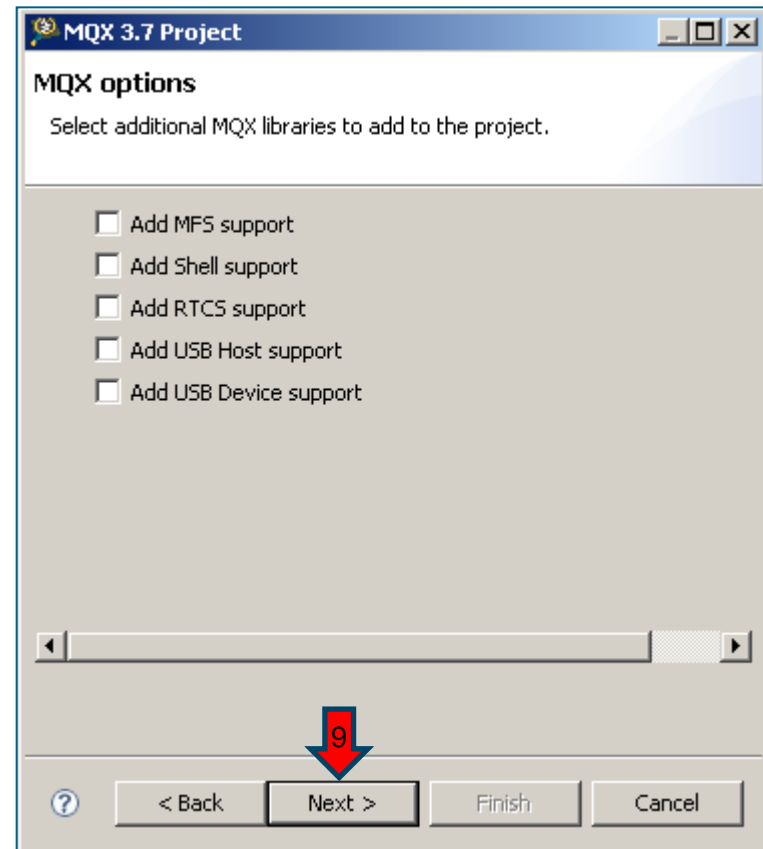
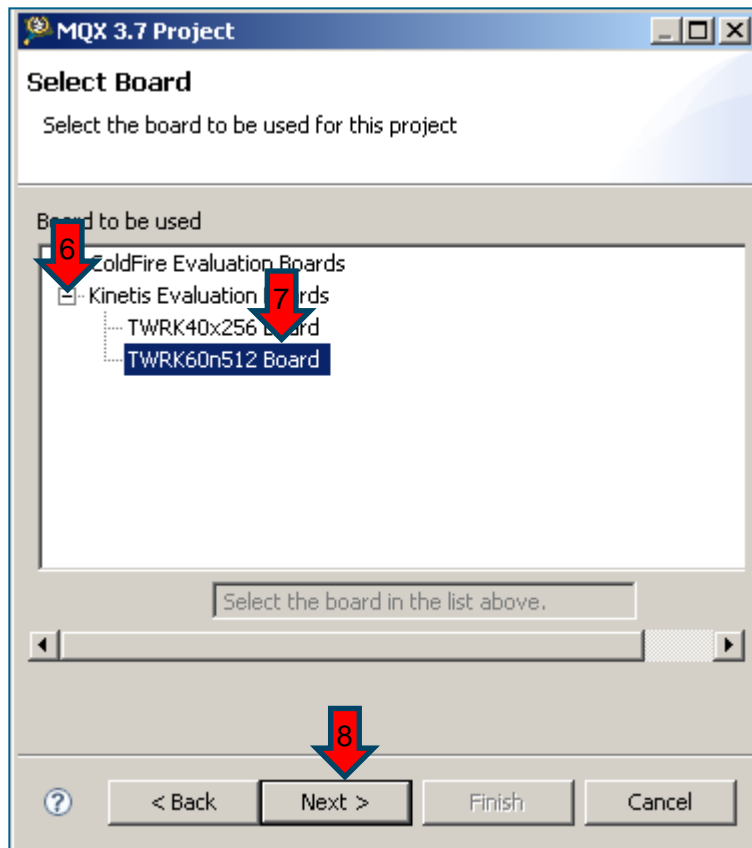
New MQX Project

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



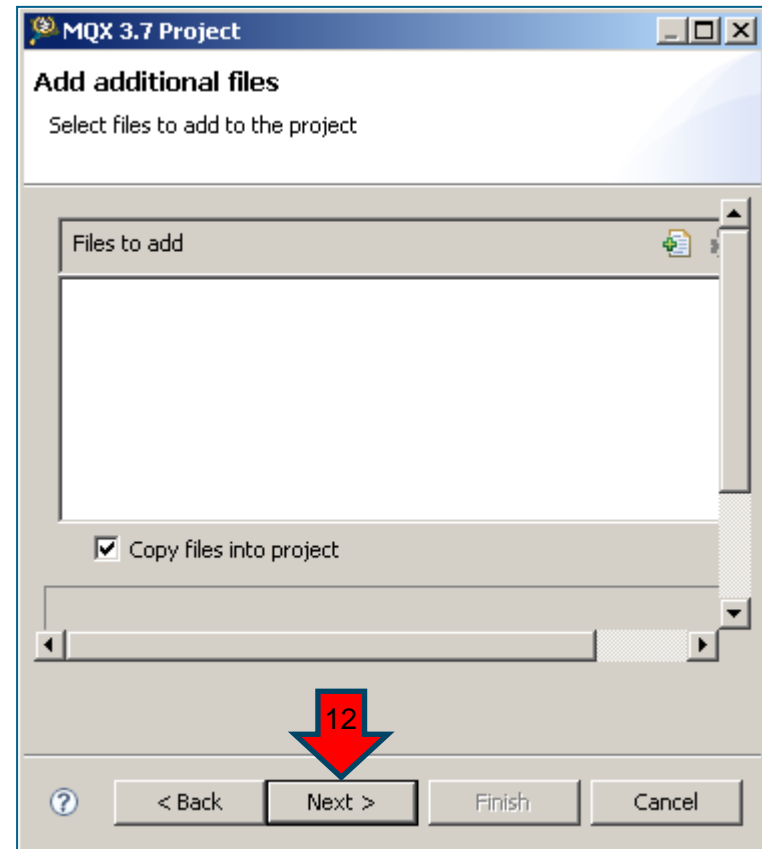
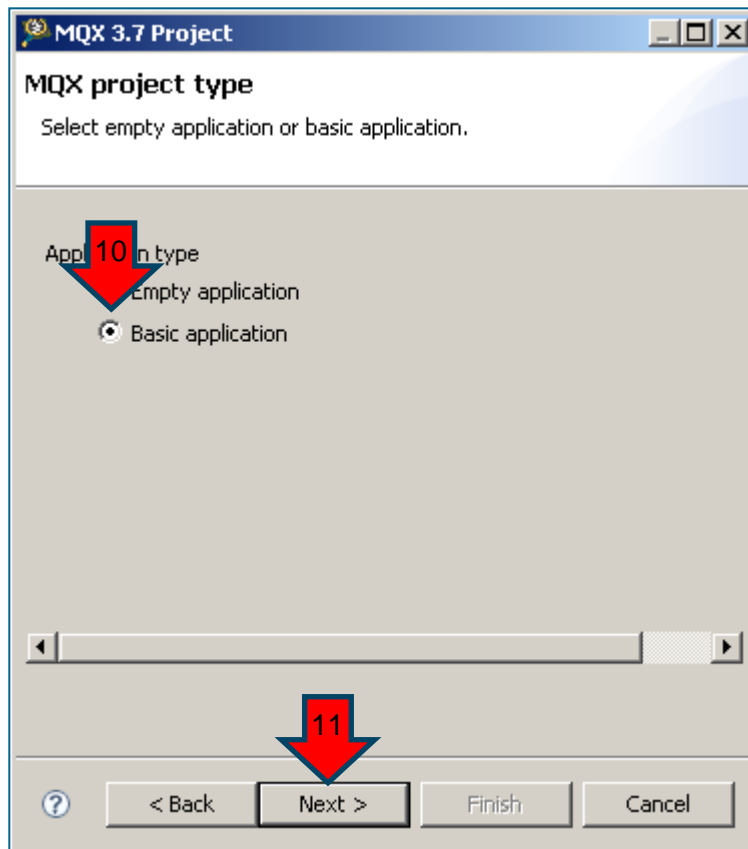
New MQX Project

- Select **TWRK60n512** Board.



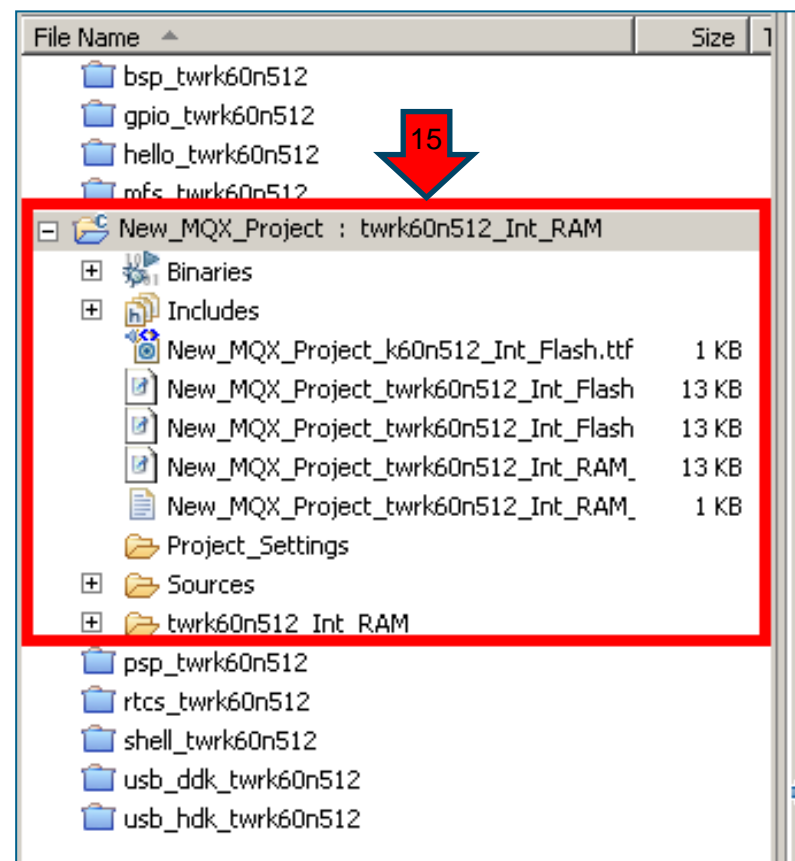
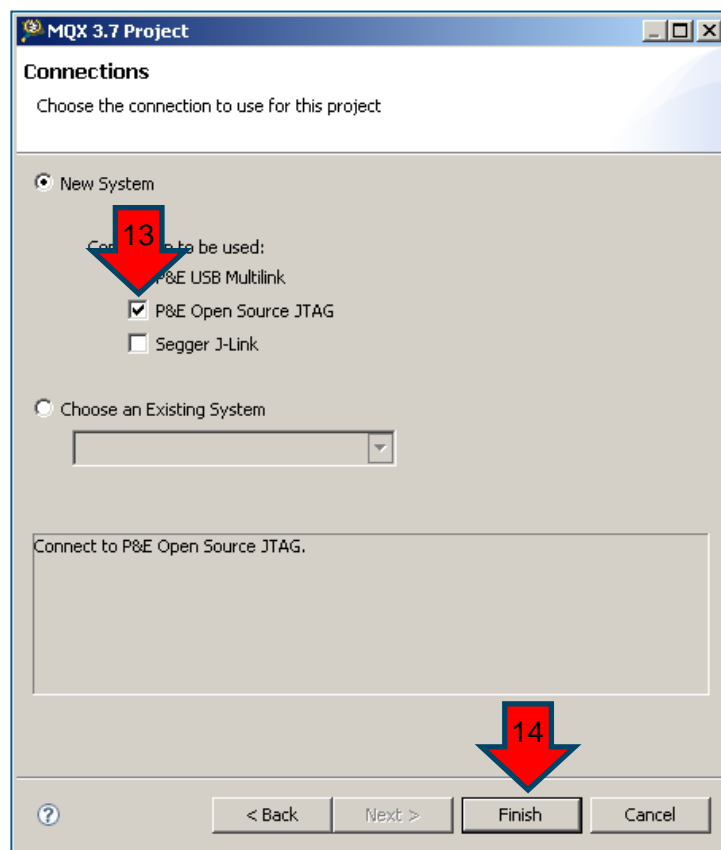
New MQX Project

- Select Basic application.



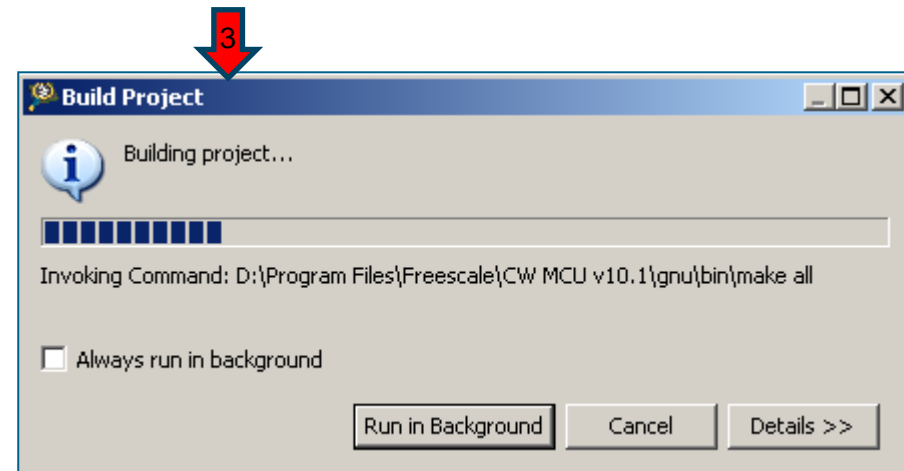
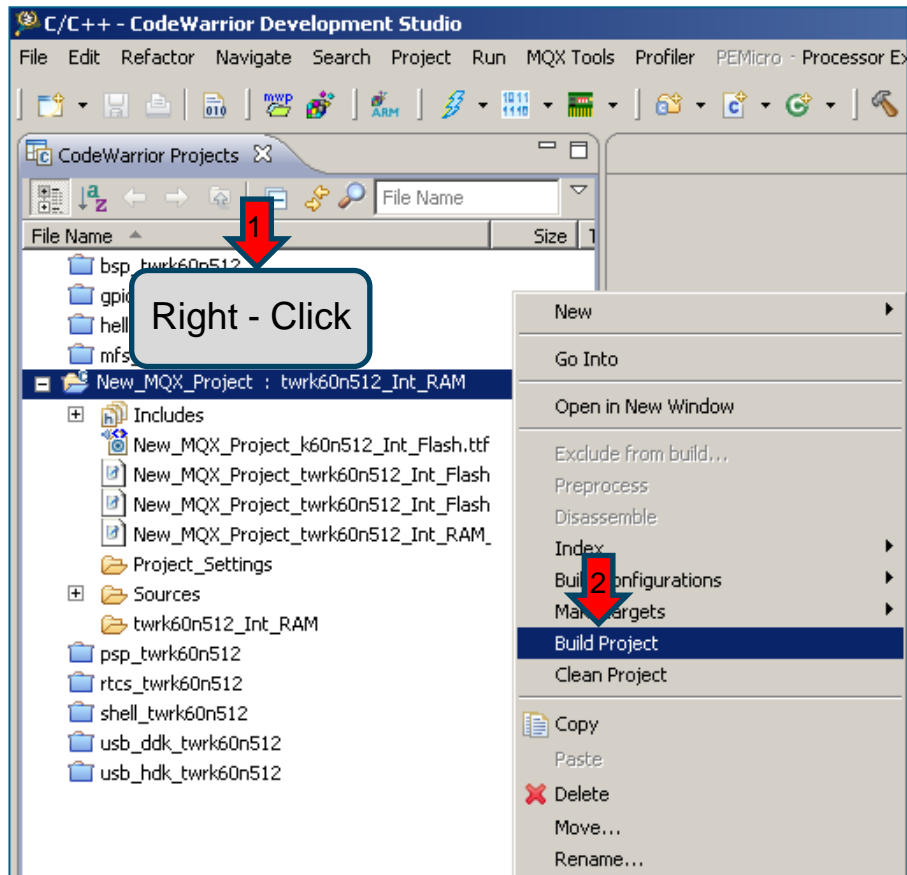
New MQX Project

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



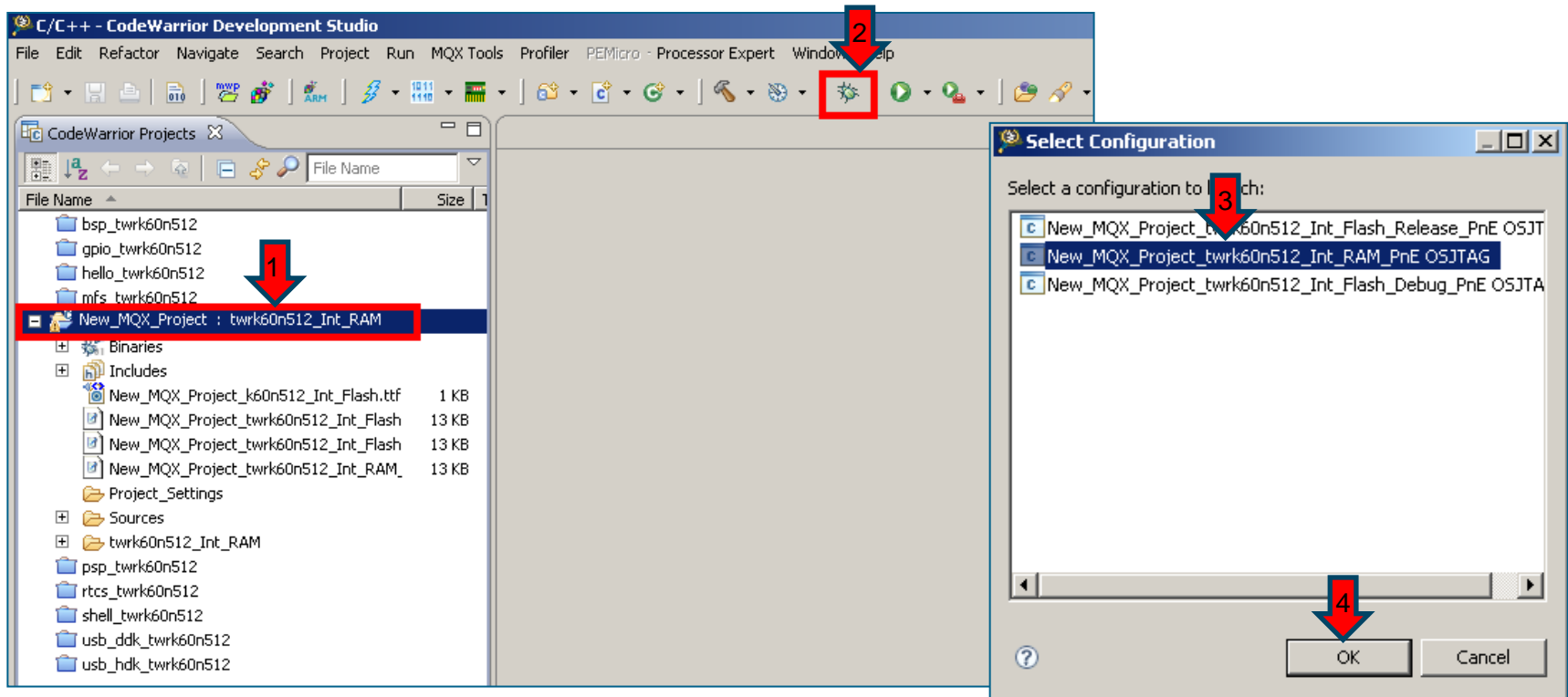
Build New MQX Project

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



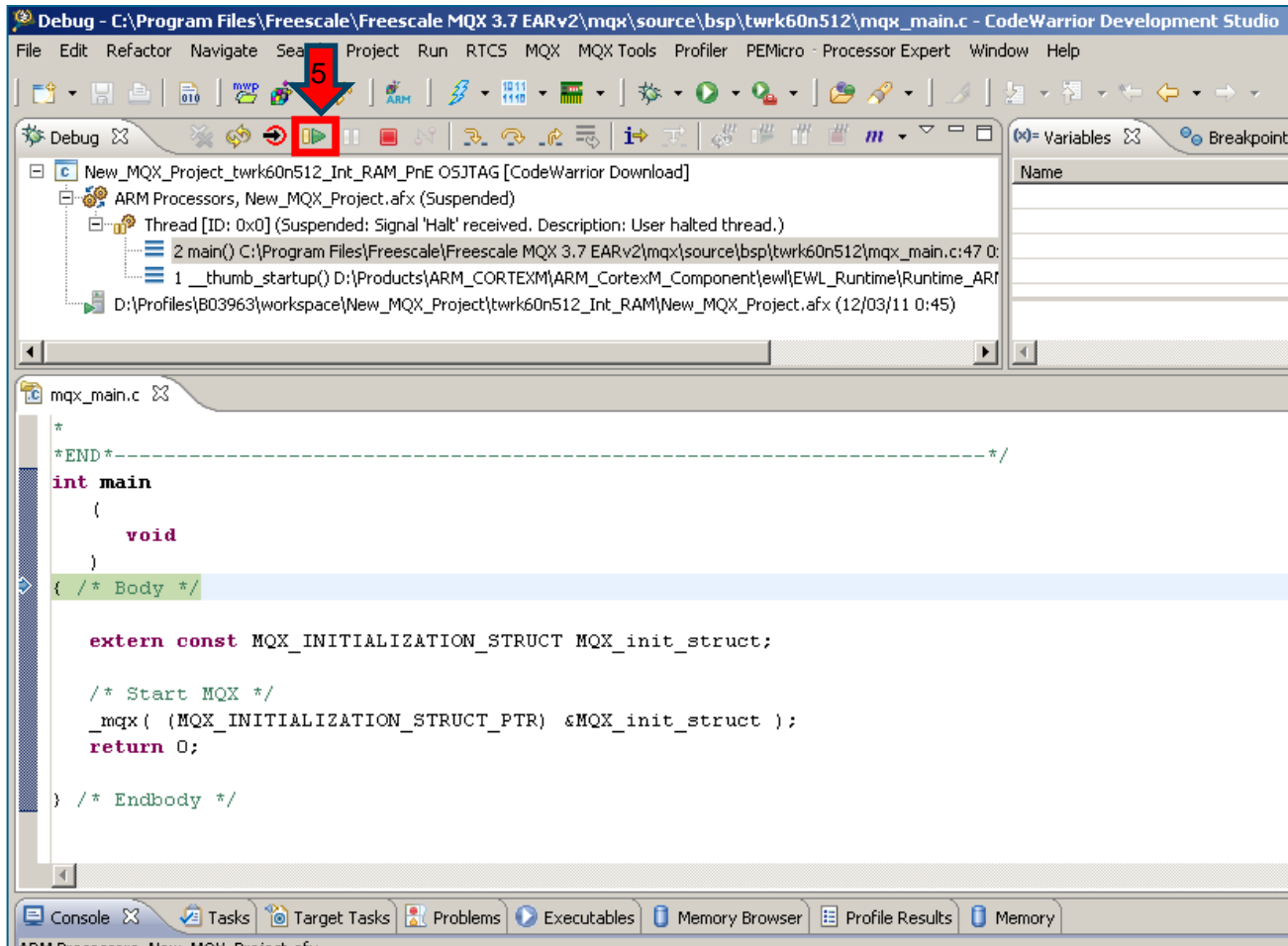
Debug New MQX Project

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



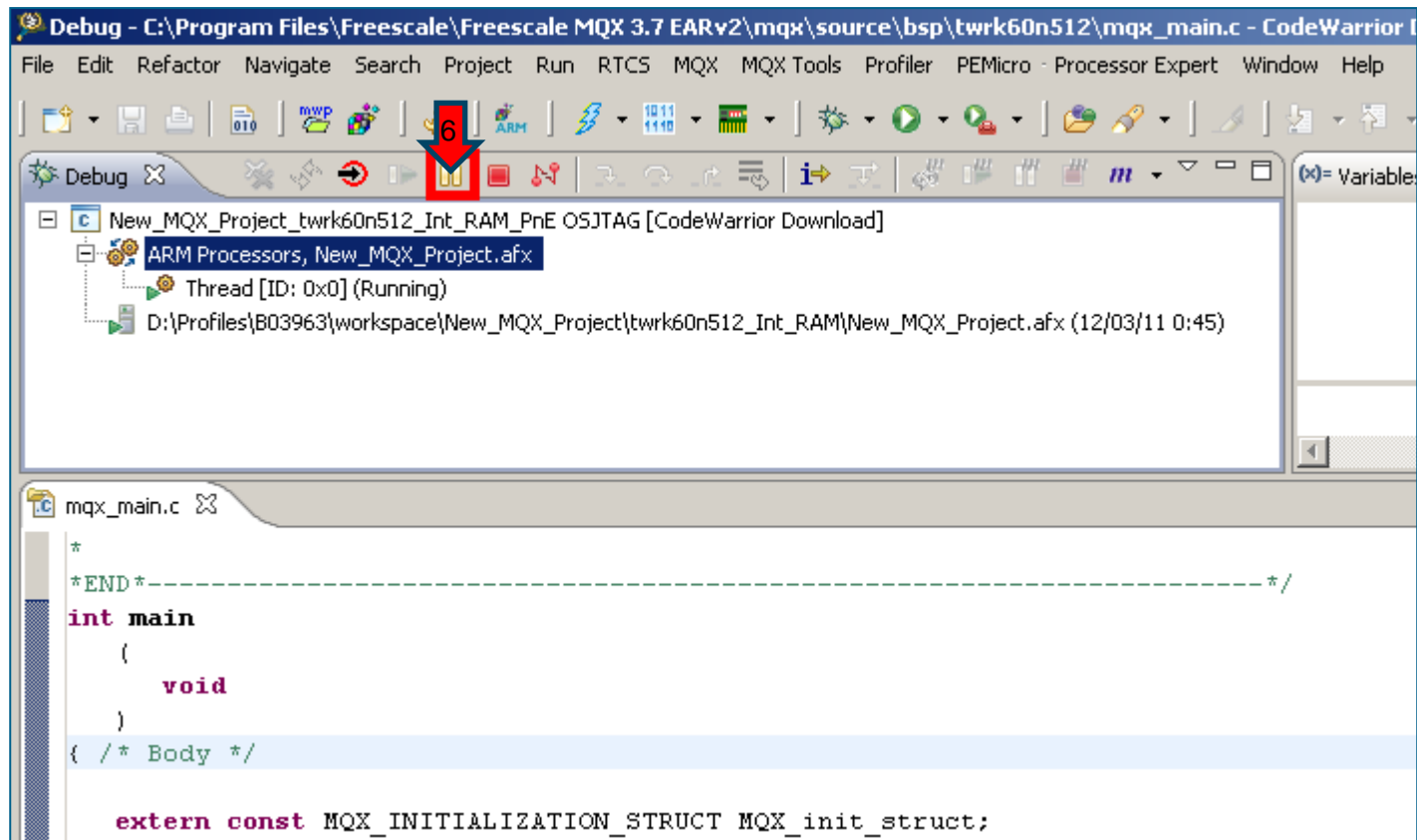
Run New MQX Project

- ▶ Execute the code 'Run' icon.



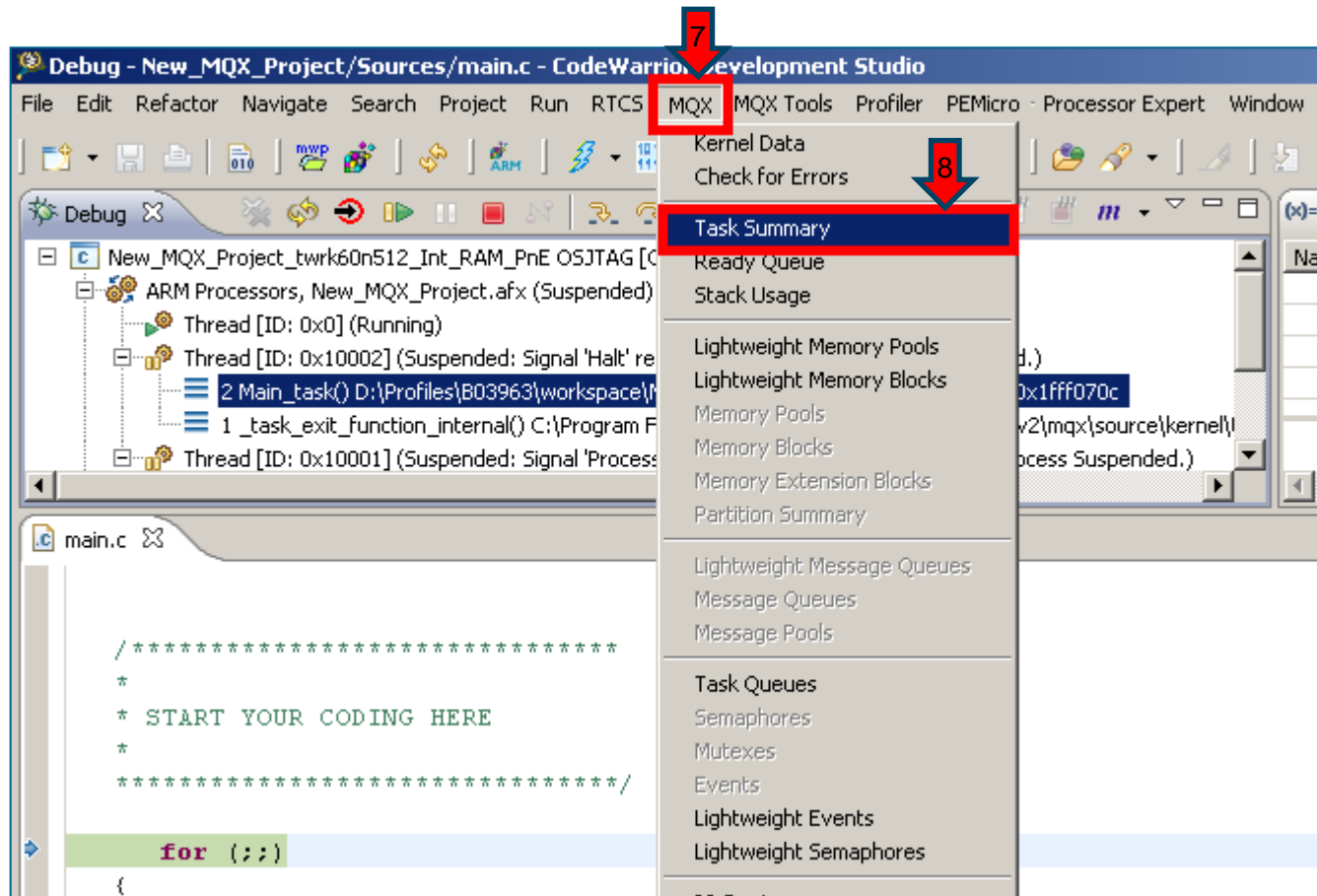
Run New MQX Project

- Pause execution.



TAD: Task Summary

► MQX -> Task Summary



TAD: Task Summary

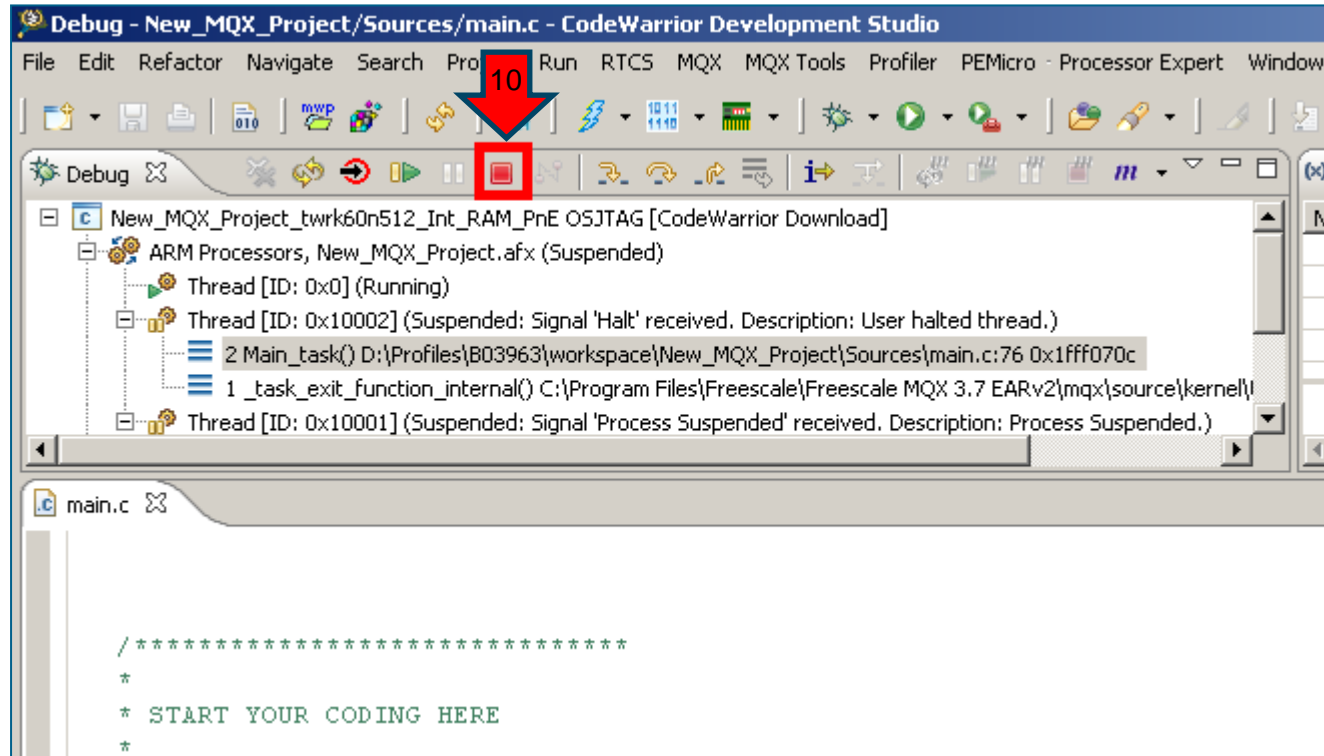
- Observe Tasks in your Application.

The screenshot shows the 'MQX Task Summary' window. The task list is as follows:

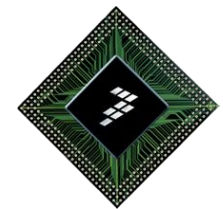
Task Name	Task ID	TD	Priority	State	Task Error Code
<input checked="" type="checkbox"/> _mqx_idle_task	0x10001	0x20000ffc	10	Ready	OK (0x0000)
Examine Task					
<input checked="" type="checkbox"/> Task Identification	Name: _mqx_idle_task				
<input checked="" type="checkbox"/> Scheduling	Flags: None				
<input checked="" type="checkbox"/> Task Status	State: Ready				
<input checked="" type="checkbox"/> main	0x10002	0x200011bc	9	Active	OK (0x0000)
Examine Task					
<input checked="" type="checkbox"/> Task Identification	Name: main				
<input checked="" type="checkbox"/> Scheduling	Flags: AutoStart				
<input checked="" type="checkbox"/> Task Status	State: Active				

Run New MQX Project

- Terminate execution.

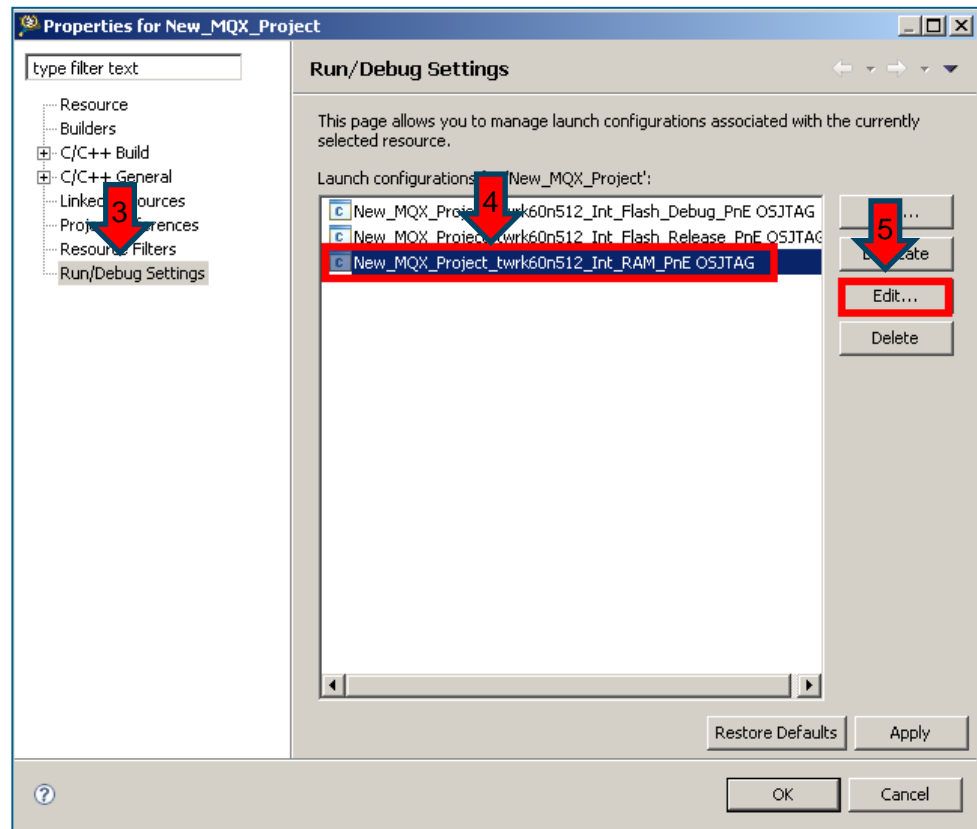
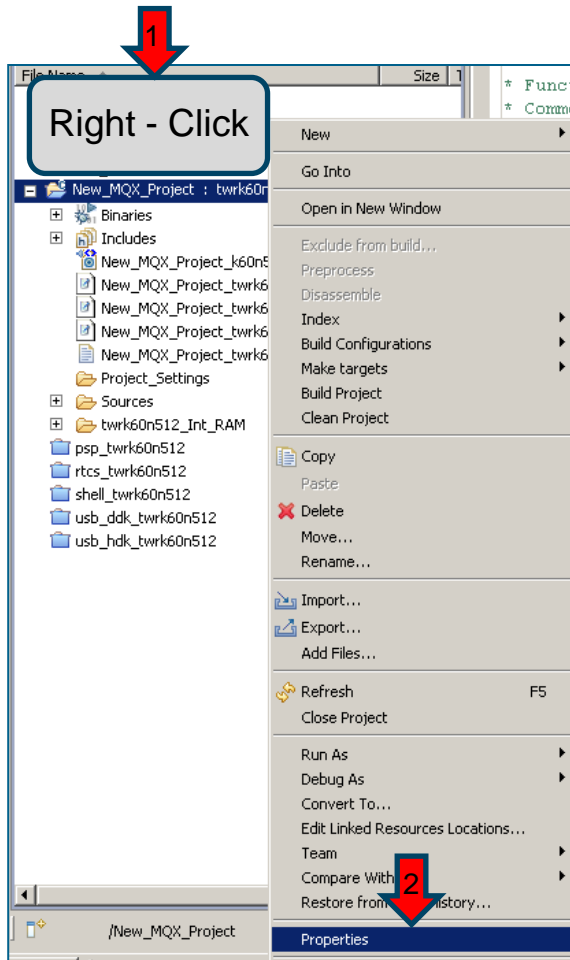


Debugging with J-Link



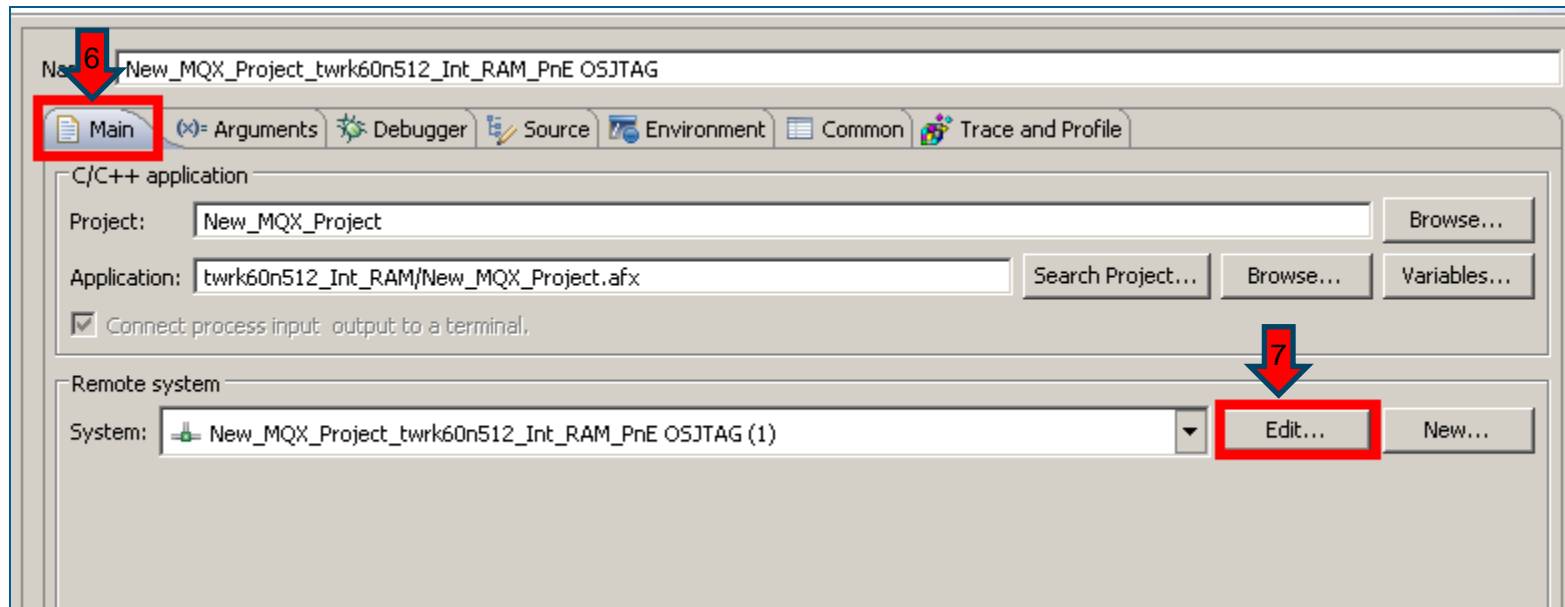
Change Connection Type

- Edit the Connection Settings of the project.



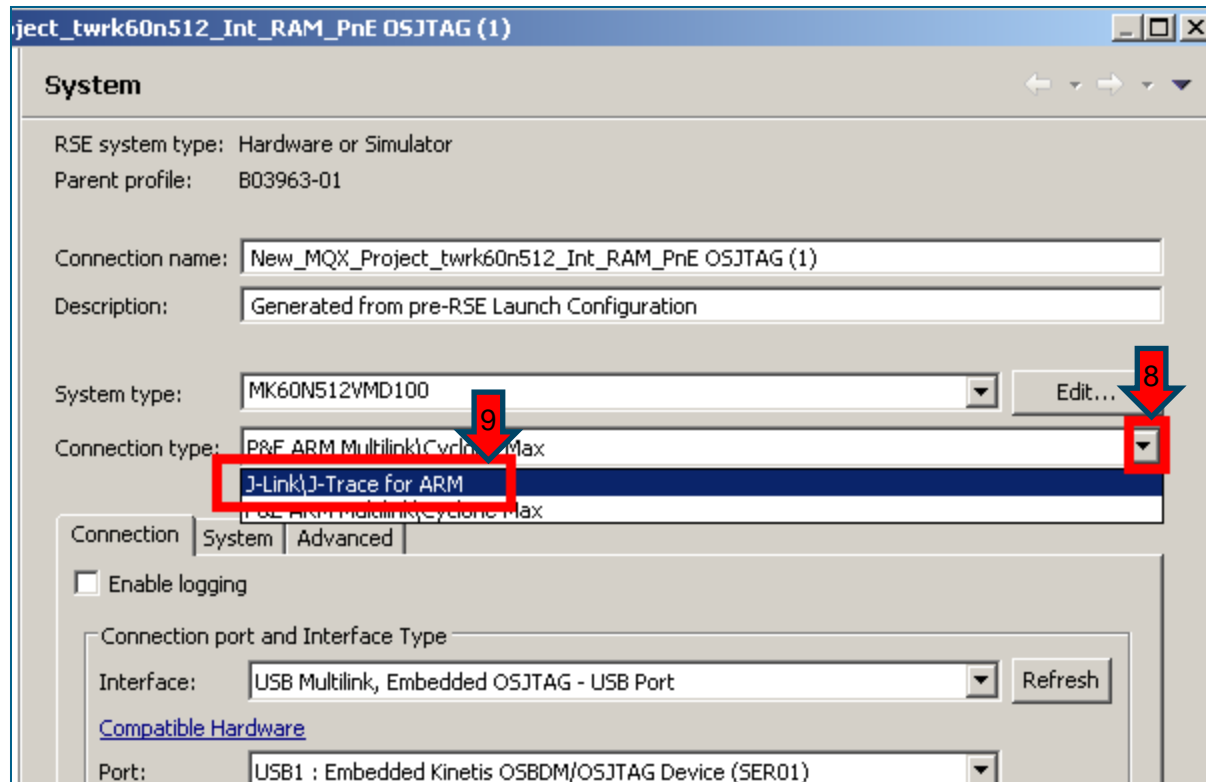
Change Connection Type

- Edit the Remote System.



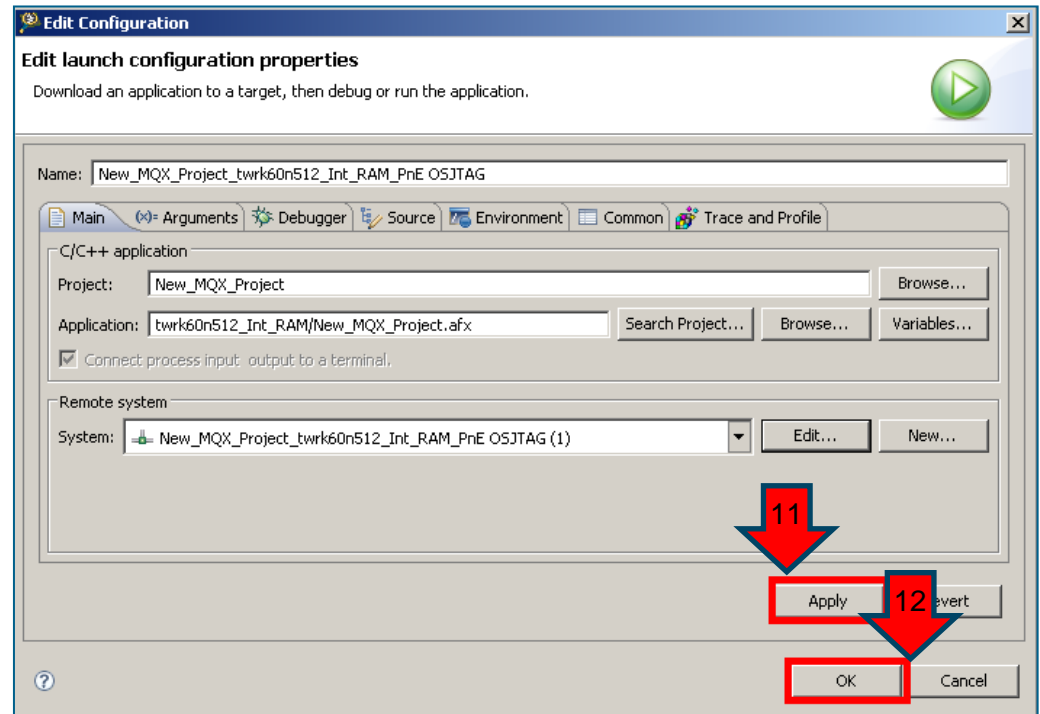
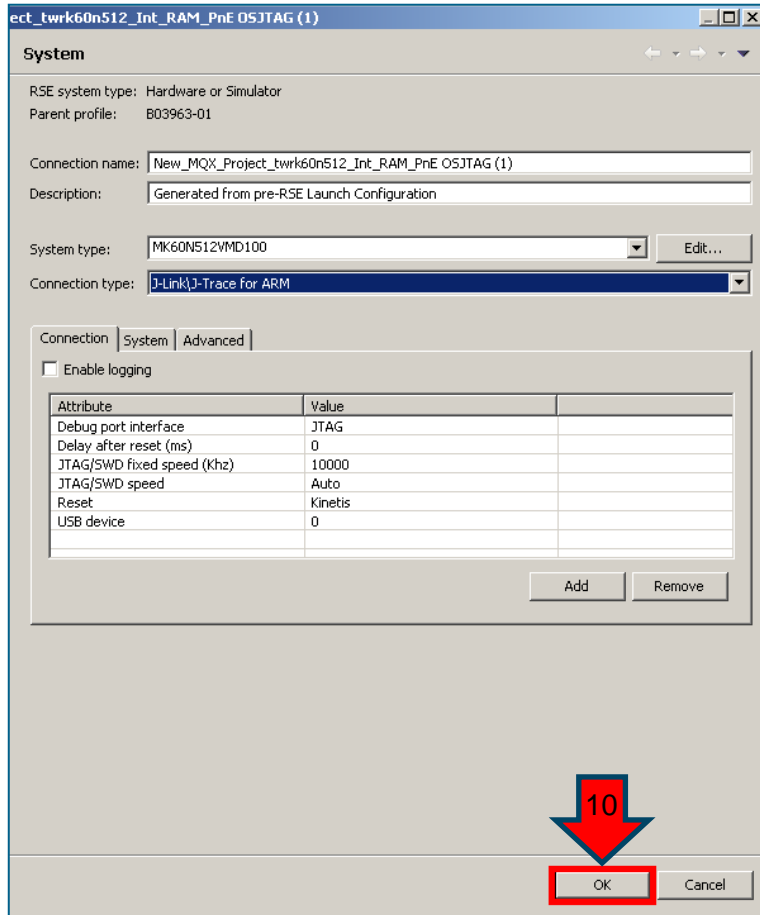
Change Connection Type

- Select **J-Link\J-Trace for ARM®**



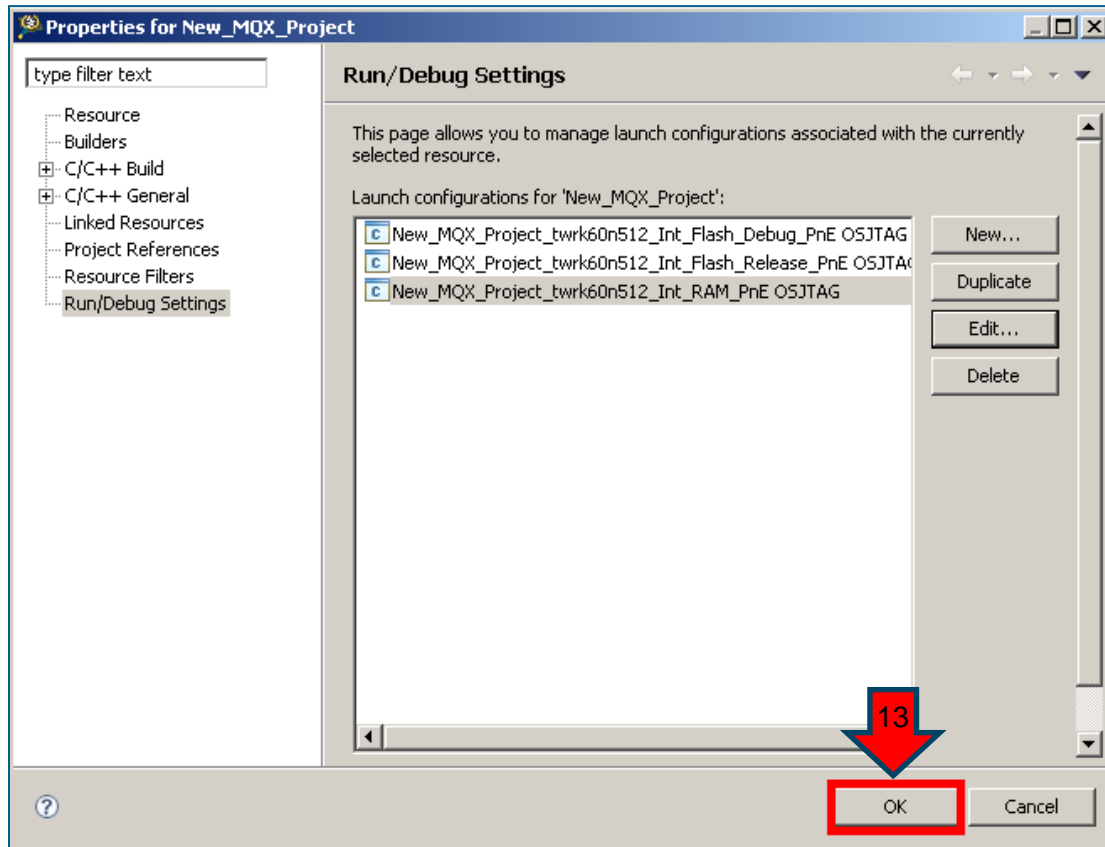
Change Connection Type

► Confirm changes.



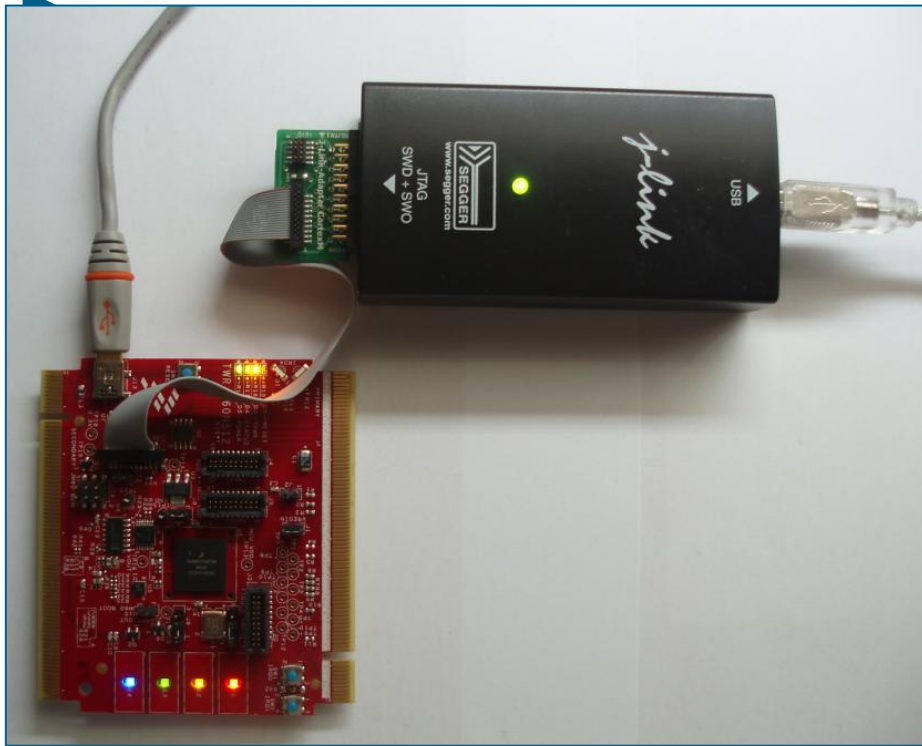
Change Connection Type

- Click OK.



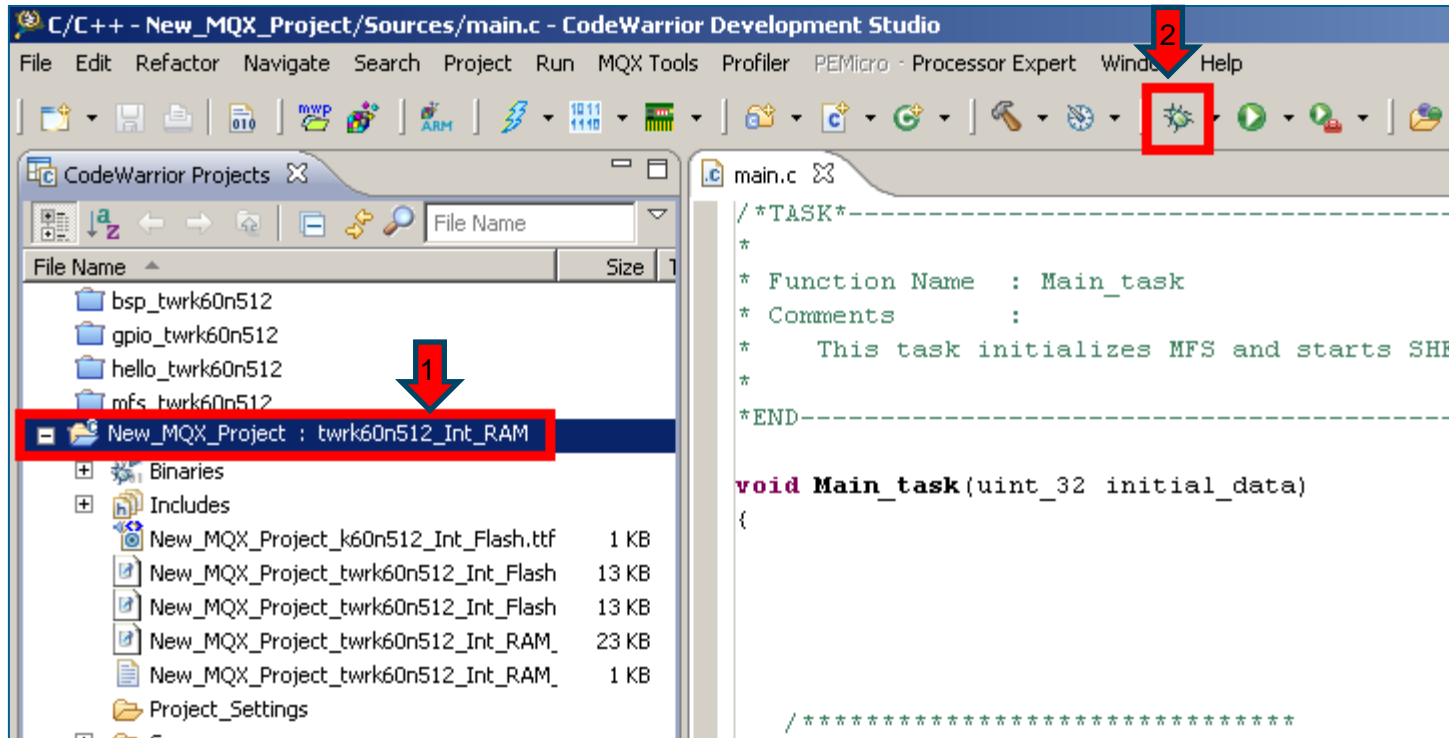
Debug with J-Link

- ▶ 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.



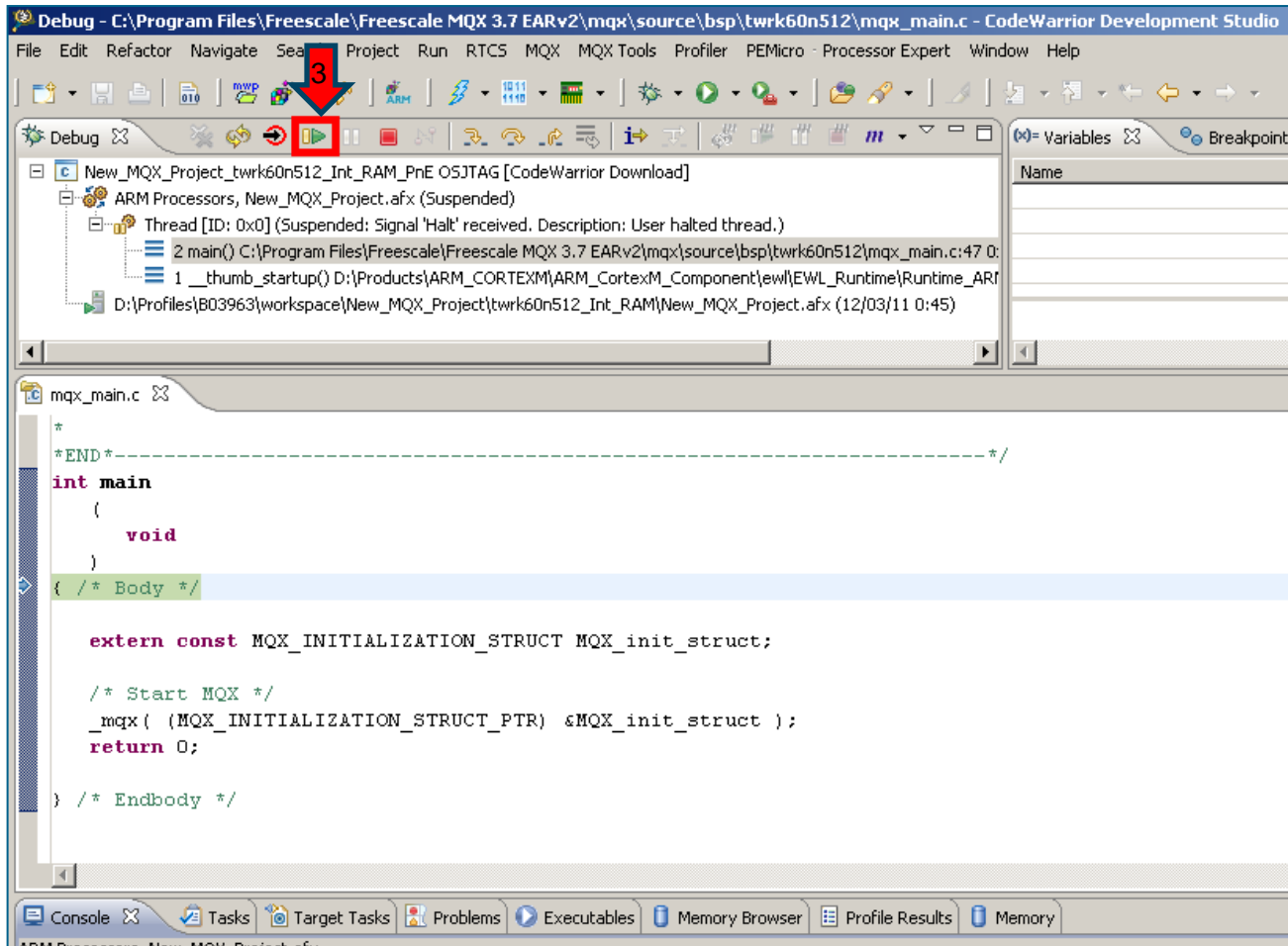
Change Connection Type

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



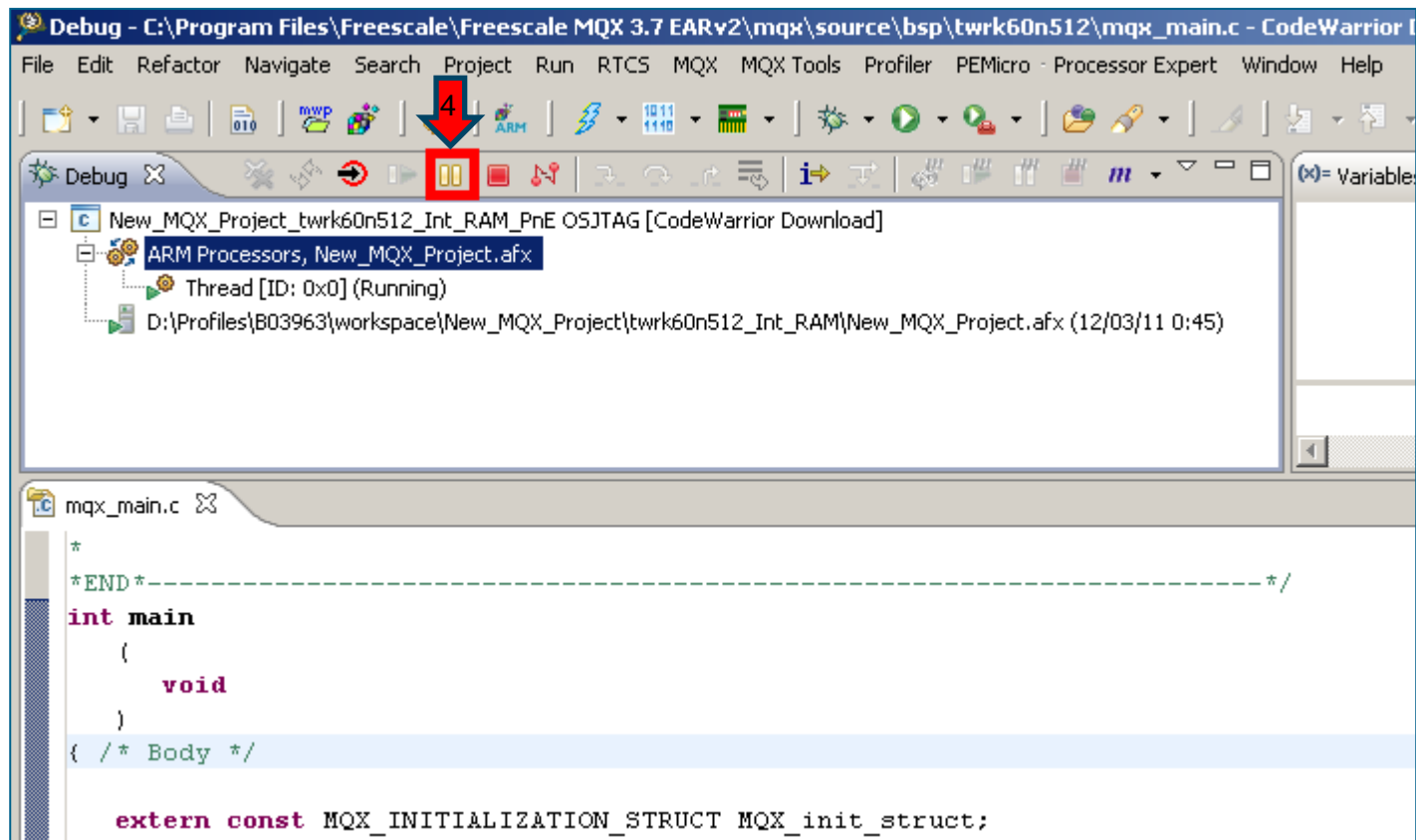
Run New MQX Project

- ▶ Execute the code 'Resume' icon.

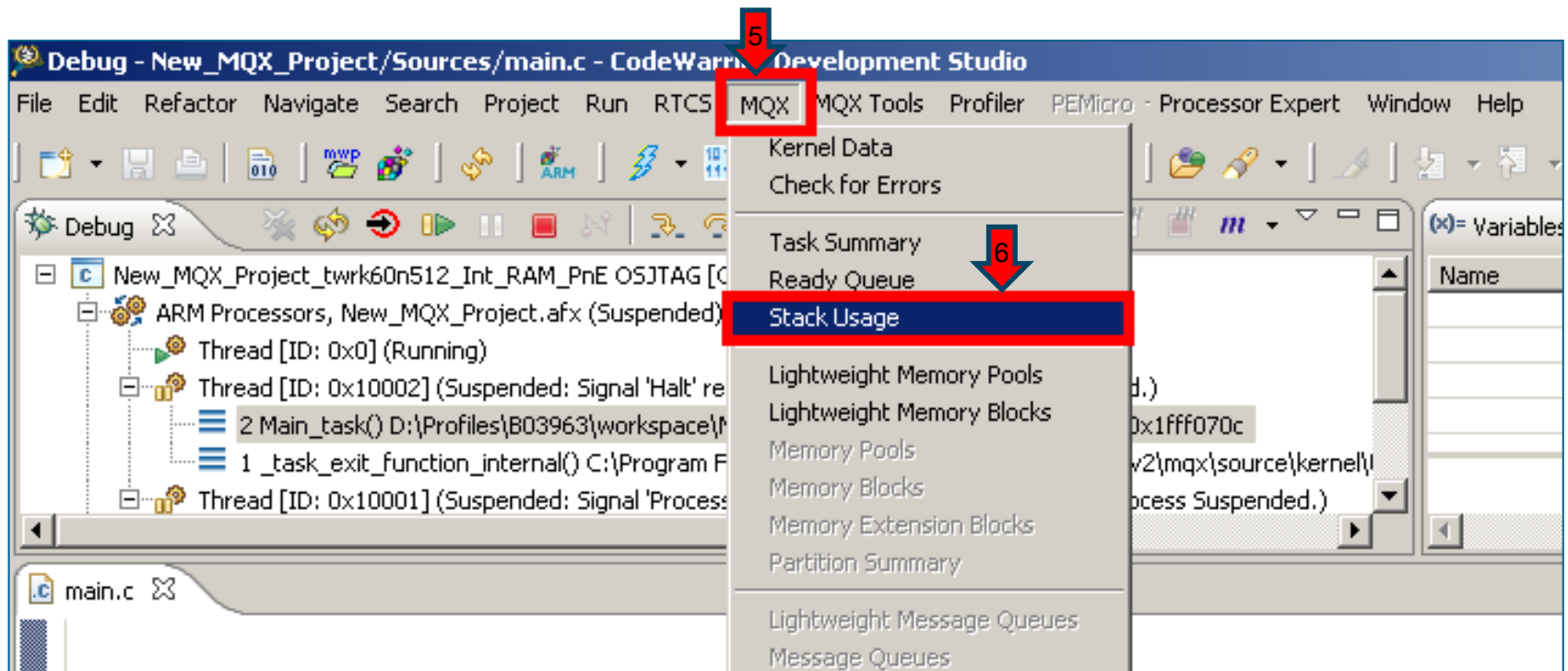


Run New MQX Project

- Pause execution.

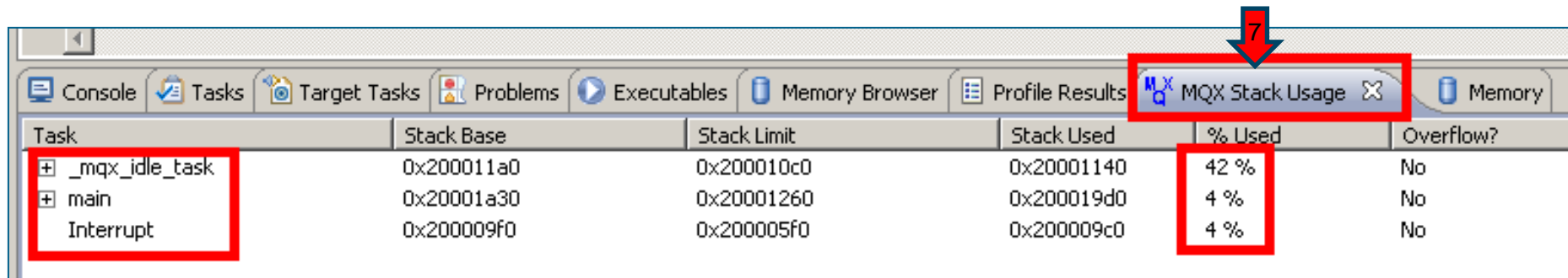


- MQX -> Stack Usage.



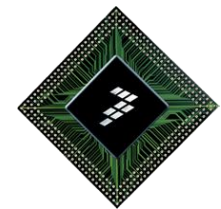
TAD: Stack Usage

► Observe Stack Data.



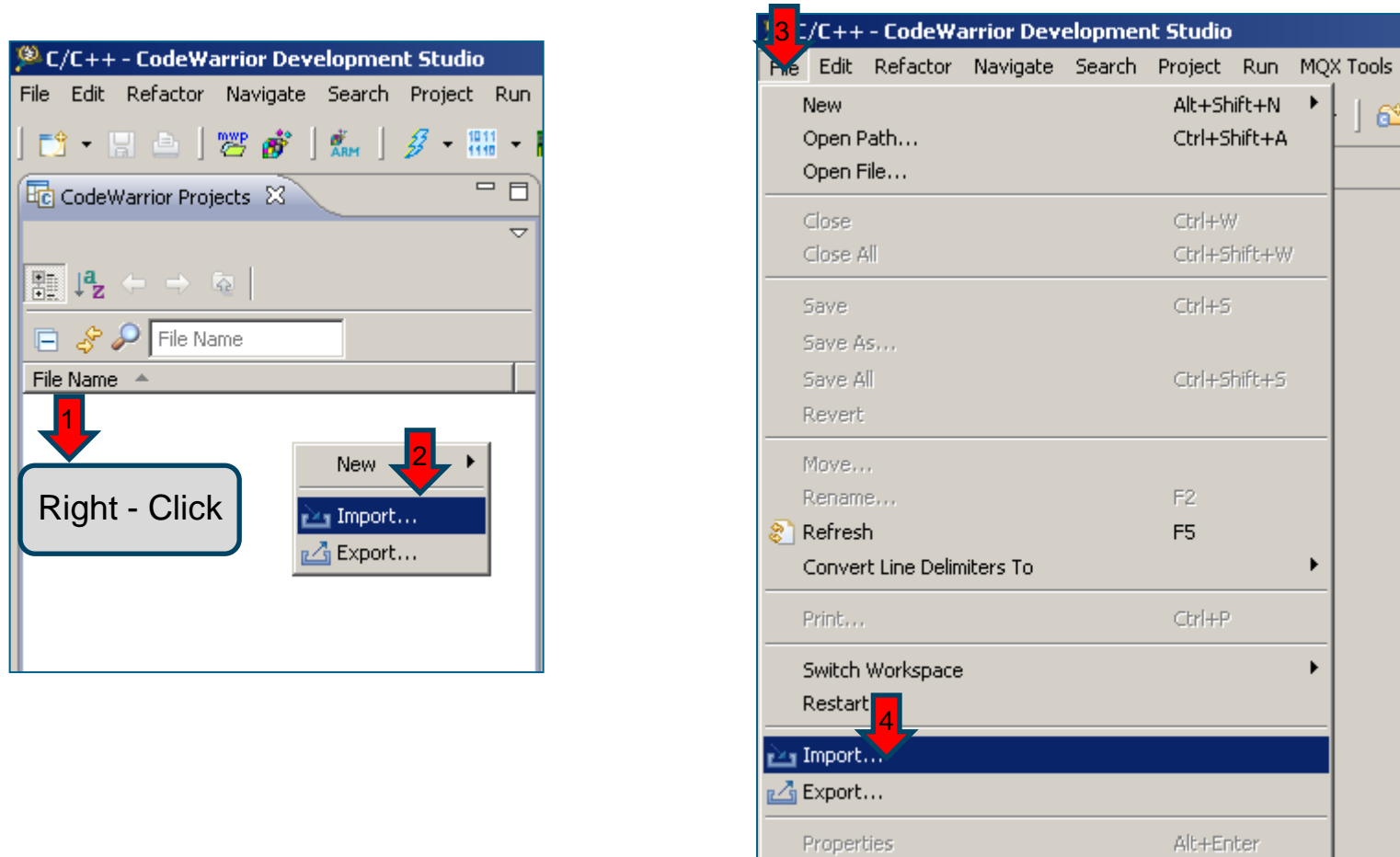
Task	Stack Base	Stack Limit	Stack Used	% Used	Overflow?
+ _mqx_idle_task	0x200011a0	0x200010c0	0x20001140	42 %	No
+ main	0x20001a30	0x20001260	0x200019d0	4 %	No
Interrupt	0x200009f0	0x200005f0	0x200009c0	4 %	No

CW10.x, MQX RTOS and Processor Expert



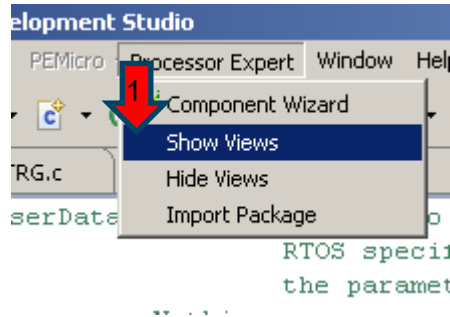
Import MQX BSP

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

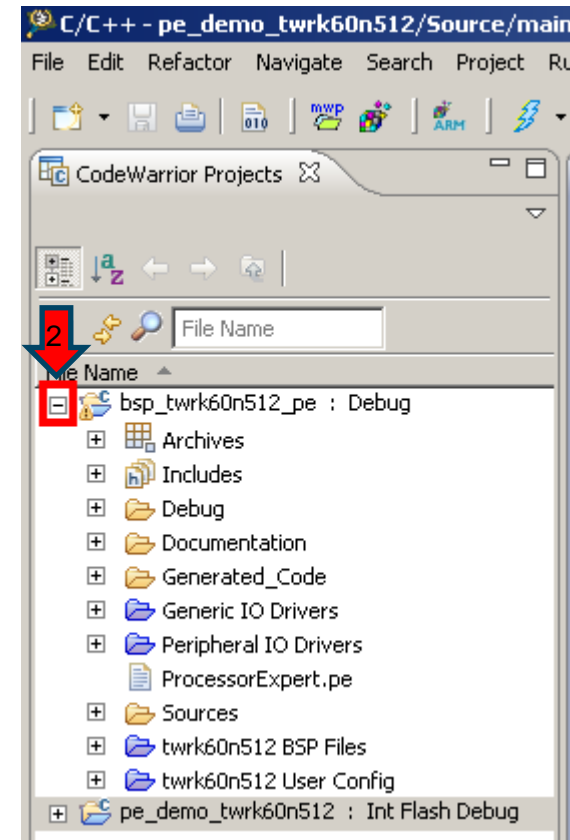
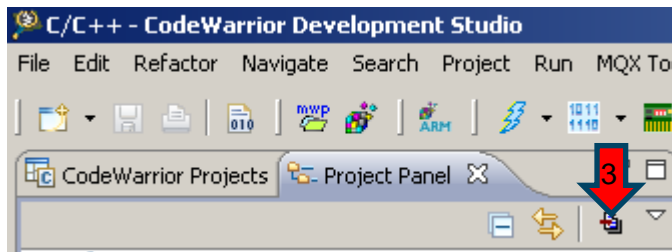


Processor Expert in MQX BSP

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



- ▶ Generate code:



Processor Expert in MQX BSP

- Click on PE components to watch the properties.

7 CodeWarrior Projects Project Panel

8 bsp_twrk60n512_pe

9 ProcessorExpert

Configuration Inspector

Name	Value
Optimizations	
Ignore range checking	no
Ignore enable test	no
Utilize after reset values	no
Complete initialization in Peripheral Init.Component	no

Properties Methods Events

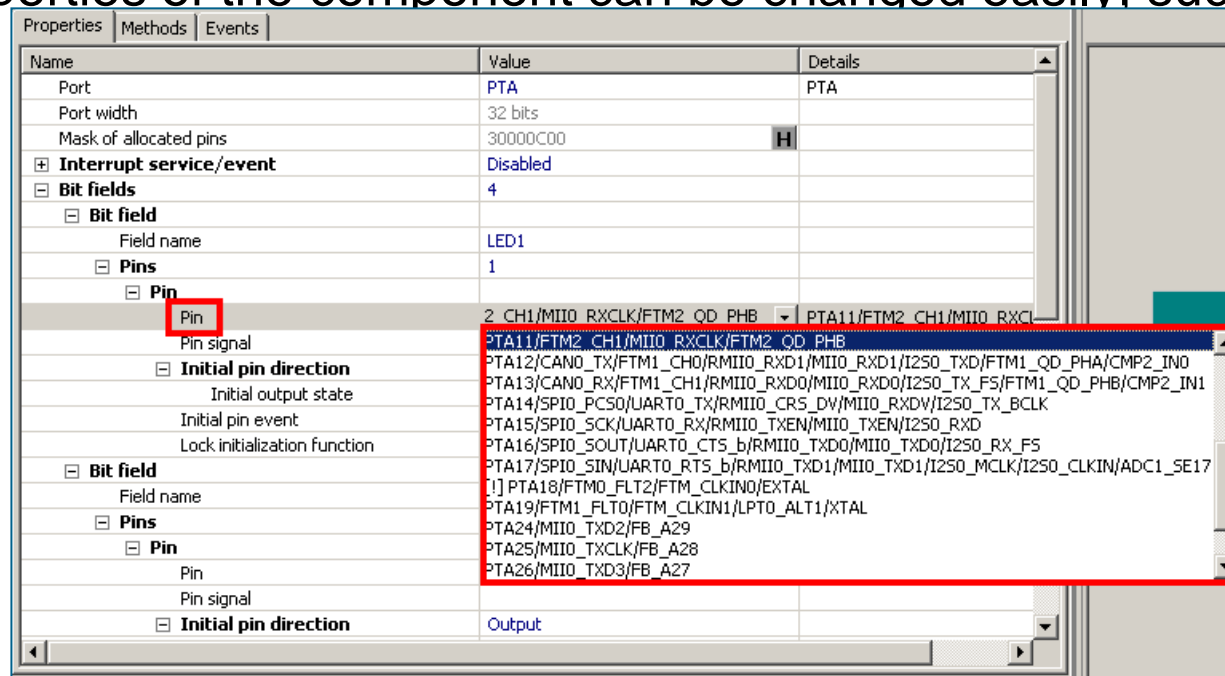
Name	Value
Component name	MQX1
Settings	
RTOS adaptor	
Memory allocation	
User function for memory allocation	no
User function for memory deallocation	no
User function name	
Critical section	
User function for entering critical section	no
User function for exiting critical section	no

Properties Methods Events

Name	Value	Details
Port	PTA	PTA
Port width	32 bits	
Mask of allocated pins	30000C00	H
Interrupt service/event	Disabled	
Bit fields	4	
Bit field		
Field name	LED1	
Pins	1	
Pin		
Pin	PTA11/FTM2_CH1/MII0_RXCLK/FTM2...	PTA11/FTM2_CH1/MII0_RXCLK
Pin signal	LED_D17	
Initial pin direction	Output	
Initial output state	1	
Initial pin event	Disabled	
Lock initialization function	no	
Bit field		
Field name	LED2	
Pins	1	
Pin		
Pin	PTA28/MII0_TXER/FB_A25	PTA28/MII0_TXER/FB_A25
Pin signal		
Initial pin direction	Output	Signal name of the pin.

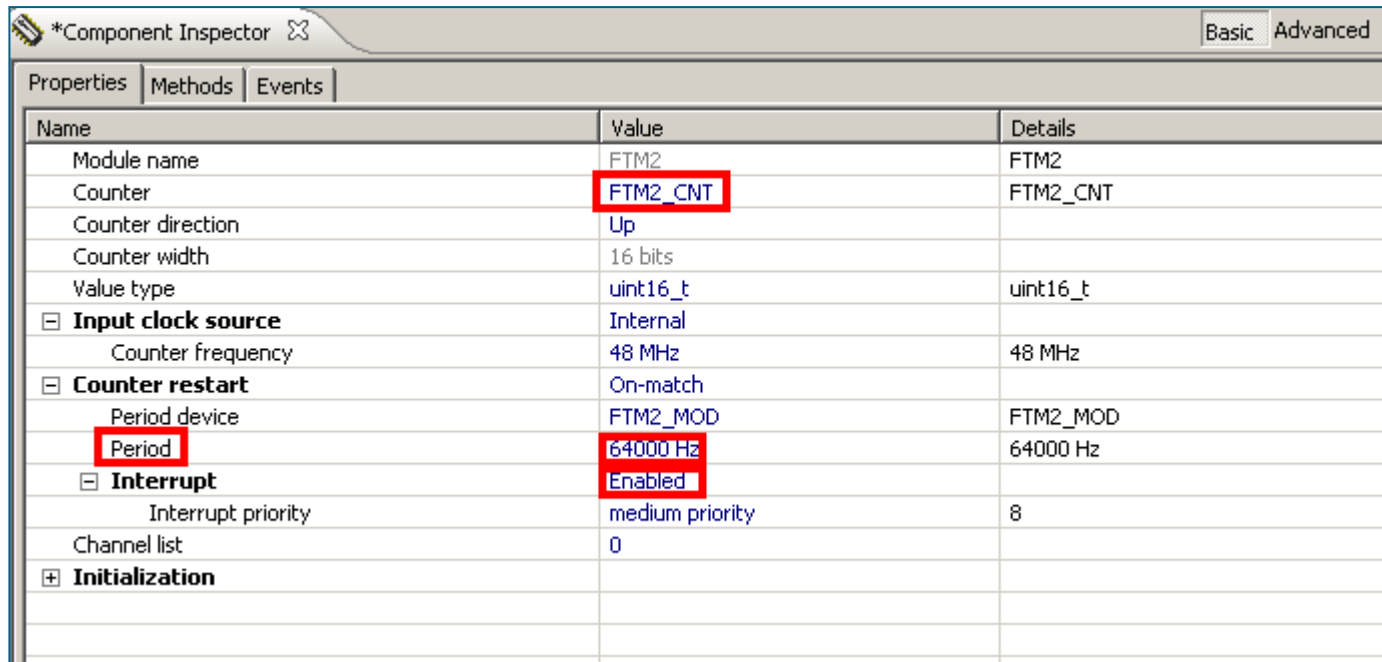
Processor Expert in MQX BSP

- ▶ 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.



Name	Value	Details
Module name	FTM2	FTM2
Counter	FTM2_CNT	FTM2_CNT
Counter direction	Up	
Counter width	16 bits	
Value type	uint16_t	uint16_t
<input type="checkbox"/> Input clock source	Internal	
Counter frequency	48 MHz	48 MHz
<input type="checkbox"/> Counter restart	On-match	
Period device	FTM2_MOD	FTM2_MOD
Period	64000 Hz	64000 Hz
<input type="checkbox"/> Interrupt	Enabled	
Interrupt priority	medium priority	8
Channel list	0	
<input type="checkbox"/> Initialization		

Processor Expert in MQX BSP

- **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 type="checkbox"/> Input clock source	Internal		
Counter frequency	48 MHz	48 MHz	
<input type="checkbox"/> Counter restart	On-match		
Period device	FTM0_MOD	FTM0_MOD	
Period	4096 timer-ticks	4096 timer-ticks	
<input type="checkbox"/> Interrupt	Enabled		
Interrupt priority	medium priority	8	
<input type="checkbox"/> Channel list	1		
<input type="checkbox"/> Channel 0			
<input type="checkbox"/> Mode	Compare		
Compare	FTM0_COV	FTM0_COV	
Offset	1 timer-ticks	1 timer-ticks	
<input type="checkbox"/> 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...	
<input type="checkbox"/> Interrupt	Disabled		
<input type="checkbox"/> Initialization			
Enabled in init. code	no		
<input type="checkbox"/> Event mask			

Processor Expert in MQX BSP

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

The screenshot displays the Processor Expert interface. The main window has three tabs: 'Properties', 'Methods', and 'Events'. The 'Methods' tab is selected and highlighted with a red box. It contains a table with the following data:

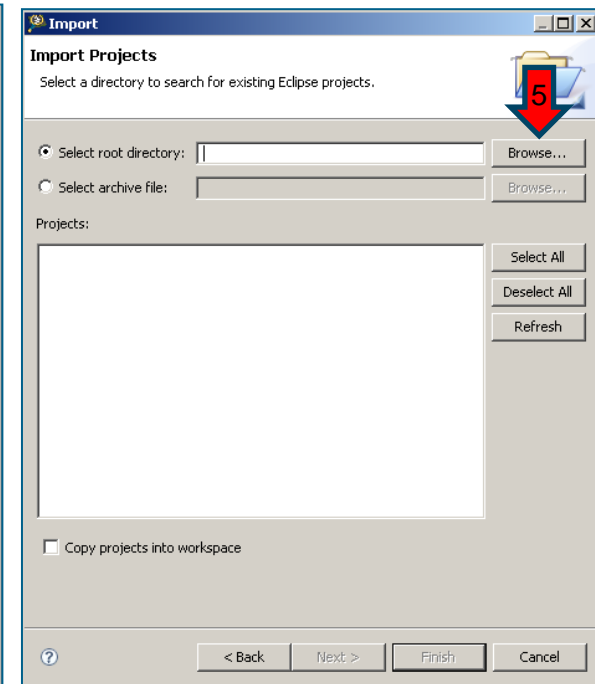
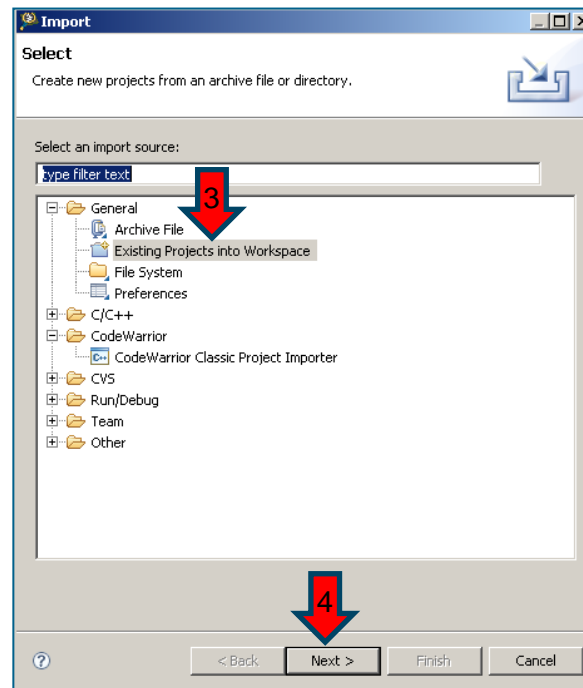
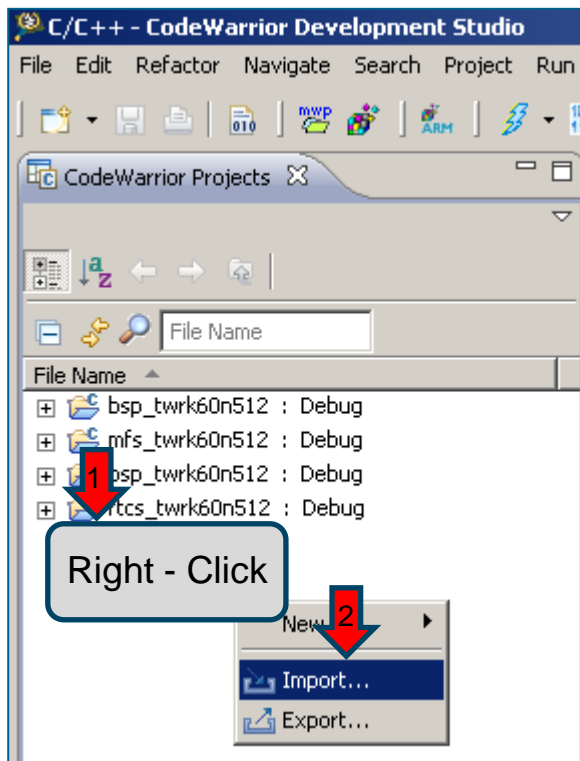
Name	Value
Init	generate code
Deinit	generate code
Enable	generate code
Disable	generate code
SetValue	generate code
SetBuffer	generate code
ForceSwTrigger	generate code
SetBufferMode	generate code
SetBufferSize	generate code
SetBufferWatermark	generate code
SetBufferReadPointer	generate code
GetBufferReadPointer	generate code

Overlaid on the main window is a smaller window titled '*Component Inspector'. It also has three tabs: 'Properties', 'Methods', and 'Events'. The 'Events' tab is selected and highlighted with a red box. It contains a table with the following data:

Name	Value	Details
+ OnBufferWatermark	generate code	
+ OnBufferEnd	generate code	
+ OnBufferStart	generate code	
+ OnComplete	don't generate code	
+ OnError	don't generate code	

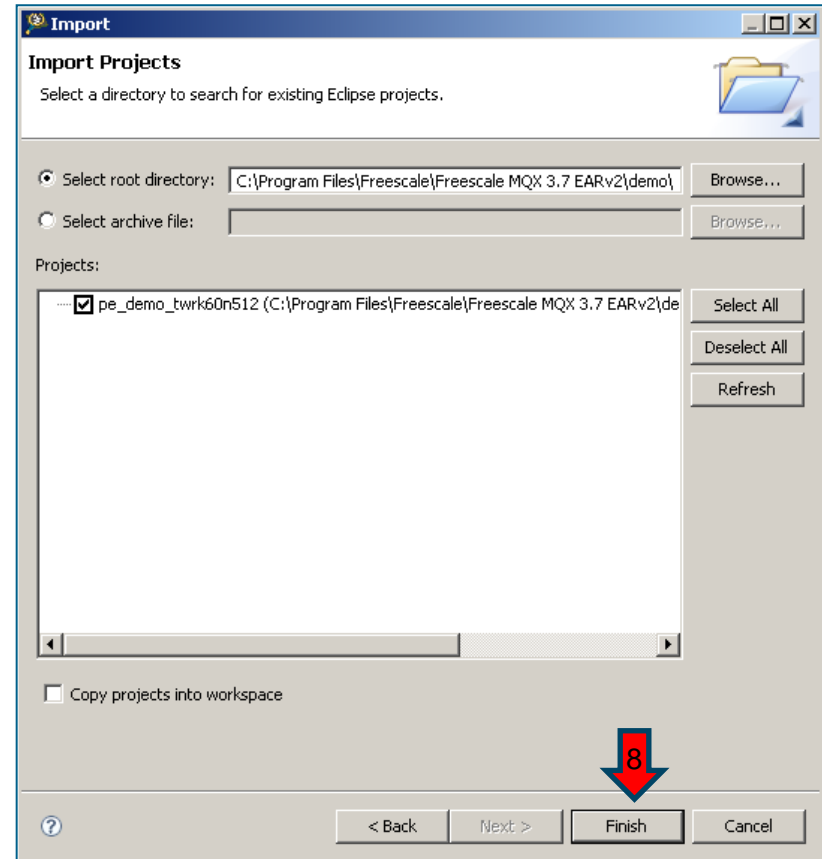
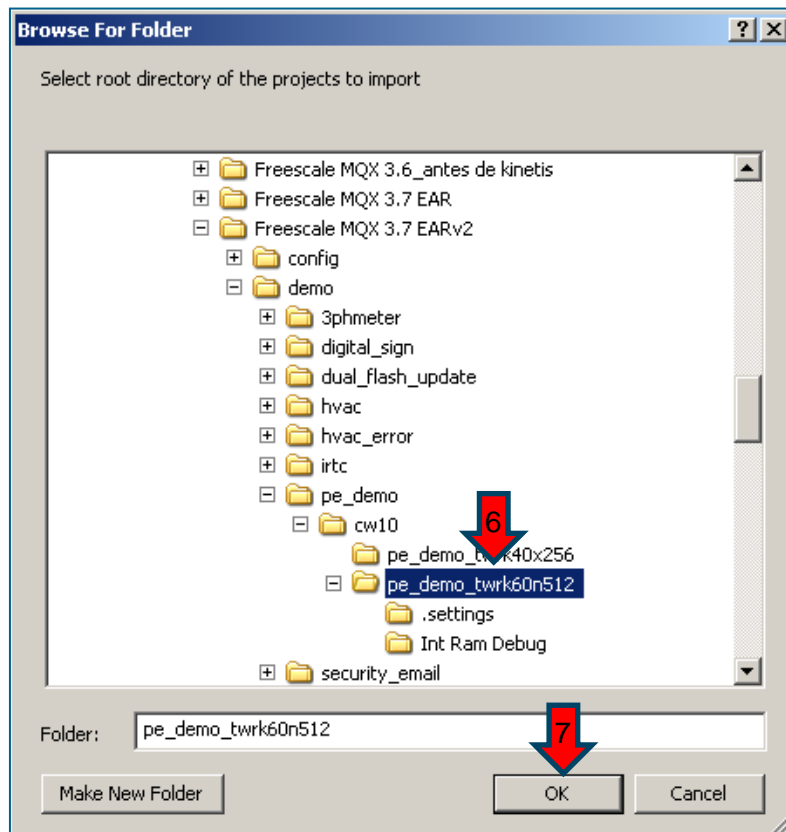
Import MQX PE Demo

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

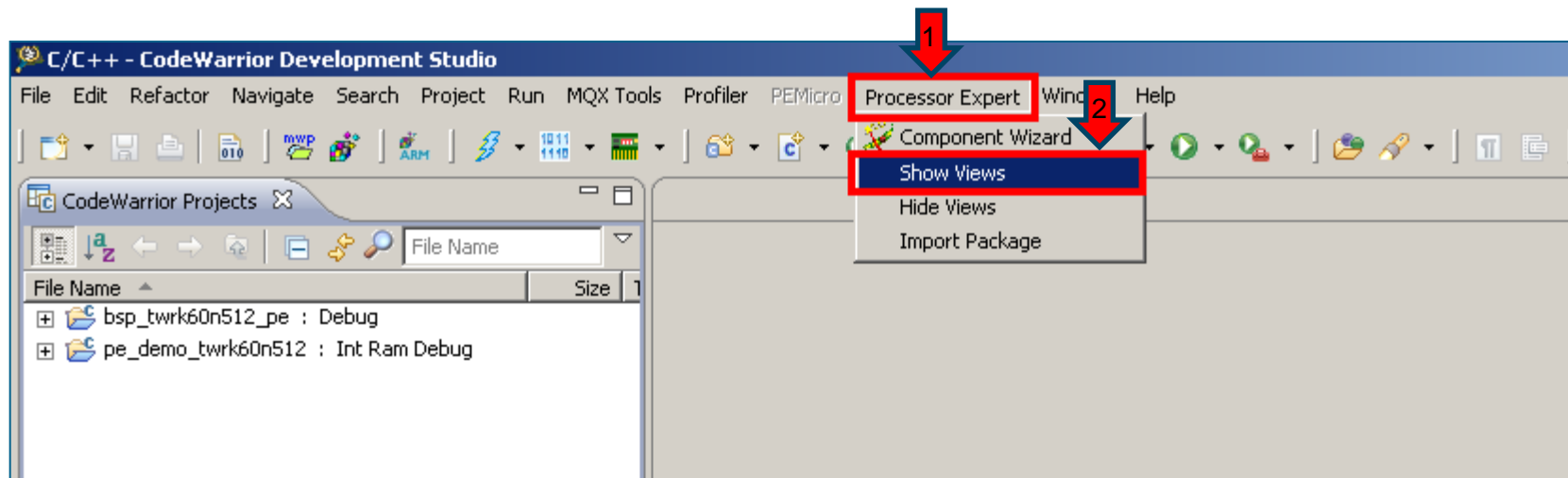


Import MQX PE Demo

- Select *<install mqx folder>\demo\pe_demo\build\cw10gcc*

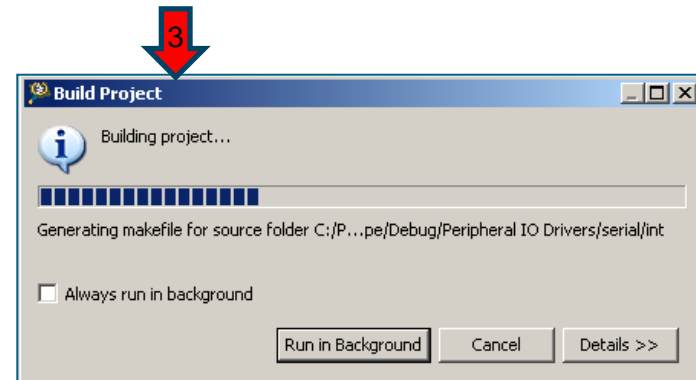
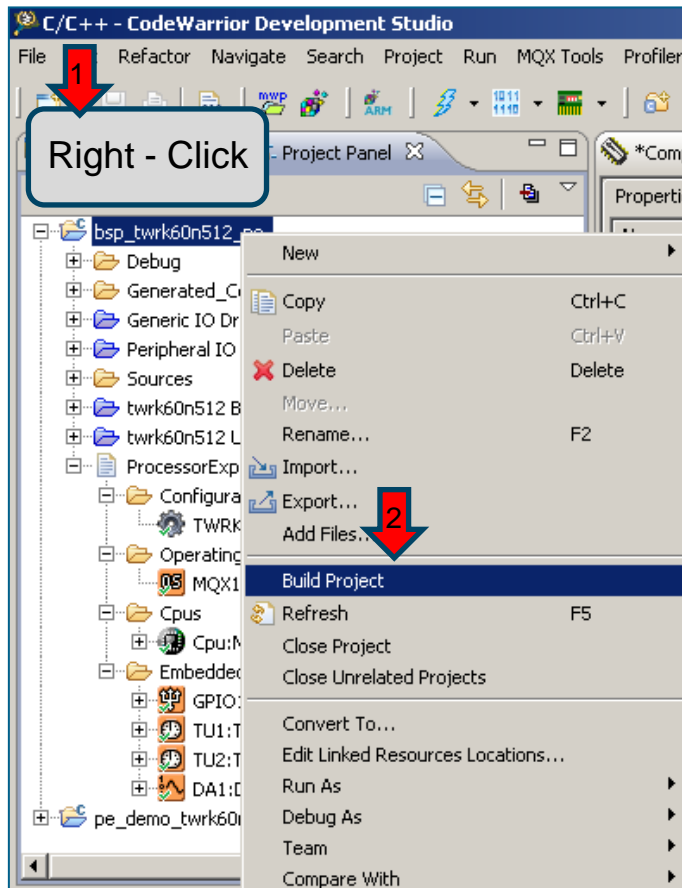


- Select in Menu : Processor Expert -> Show Views.



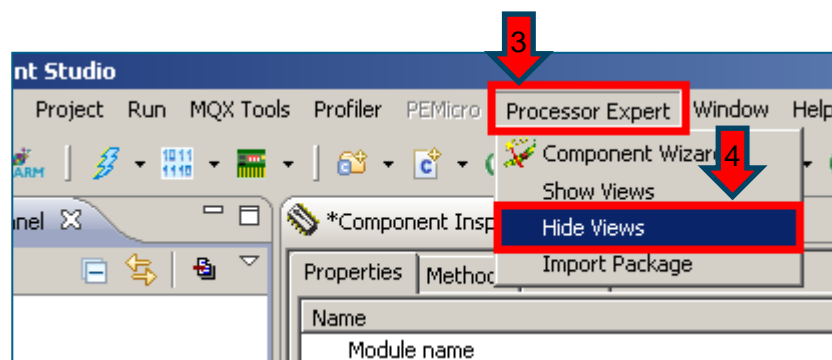
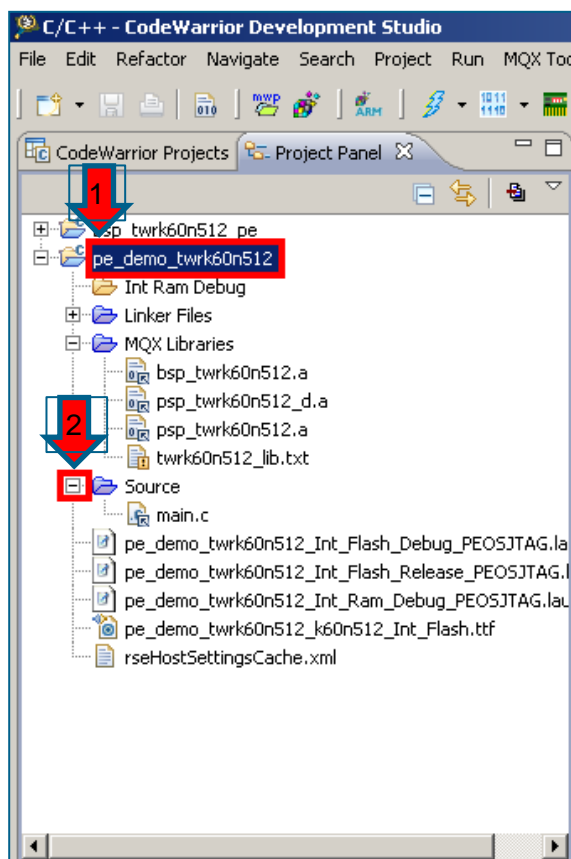
Build PE BSP

- Right-Click on Project Explorer **bsp_twrk60n512** and Build Project.



Build PE Demo

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



- ▶ 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 }
};
```

- ▶ 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;
```


- 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");
}
```

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

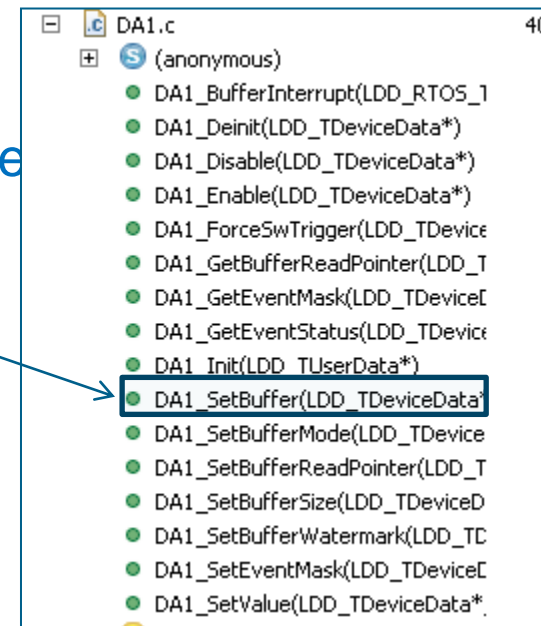
```
EWM_DeviceData = WDog1_Init(NULL);
if (EWM_DeviceData == NULL) {
    puts("failed");
    _task_block();
}
else {
    puts("done");
}
```

- 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_Buffer,  
DA1_INTERNAL_BUFFER_SIZE, 0);
```



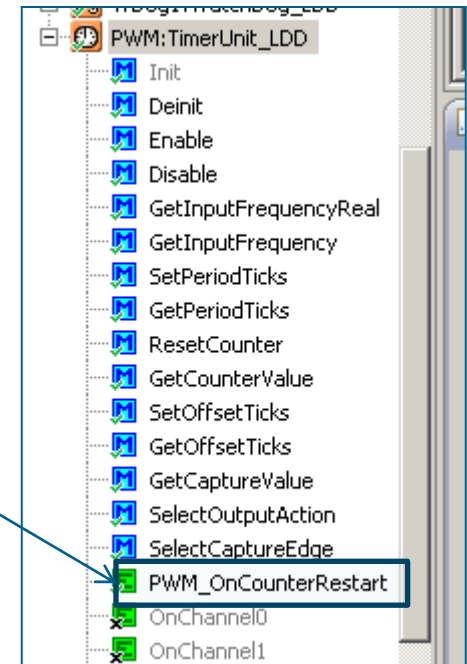
► Finally, implement the Events.

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


    PWM_Value += PWM_Step;

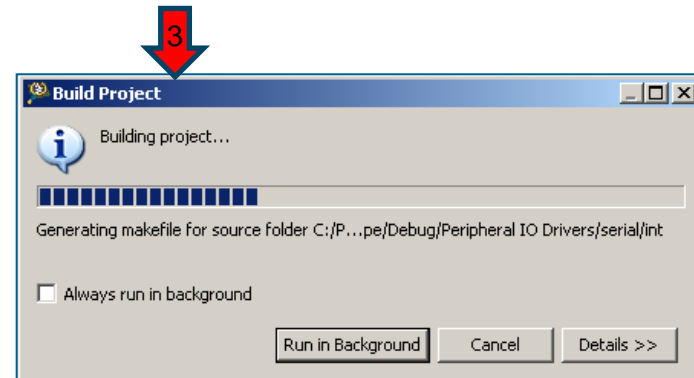
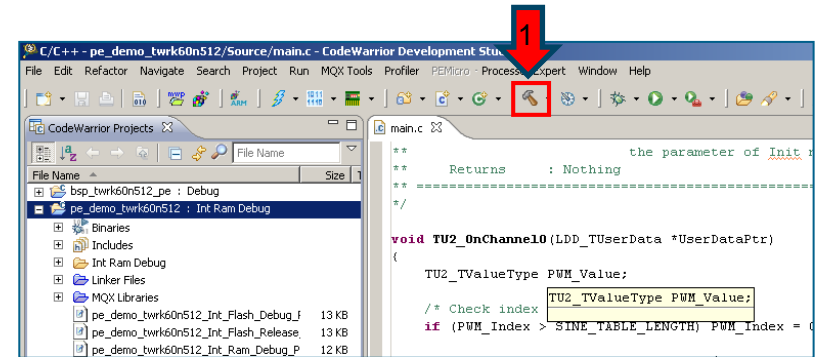
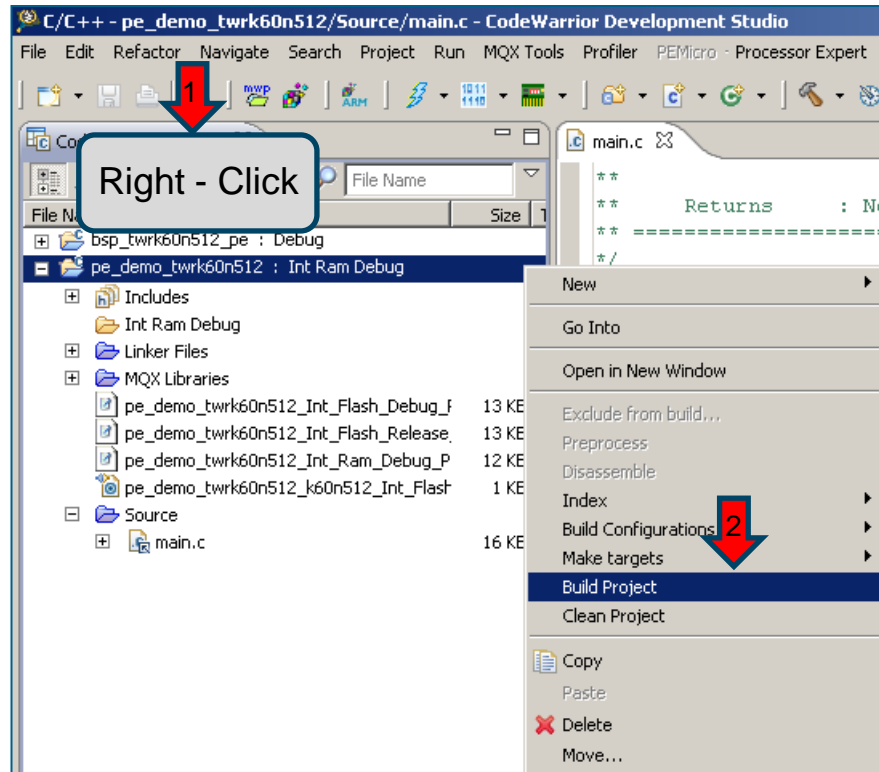
    if (PWM_Value > PWM_MaxValue) PWM_Value = 0;

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



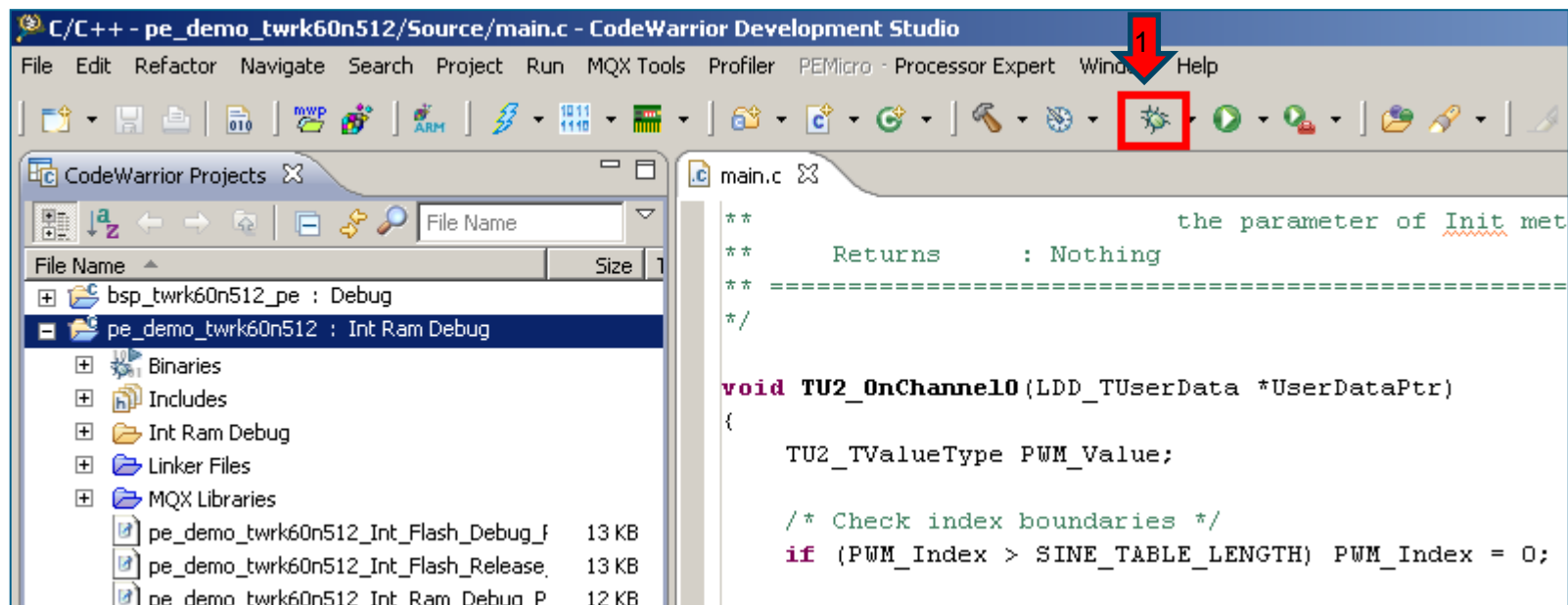
Build PE Demo

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



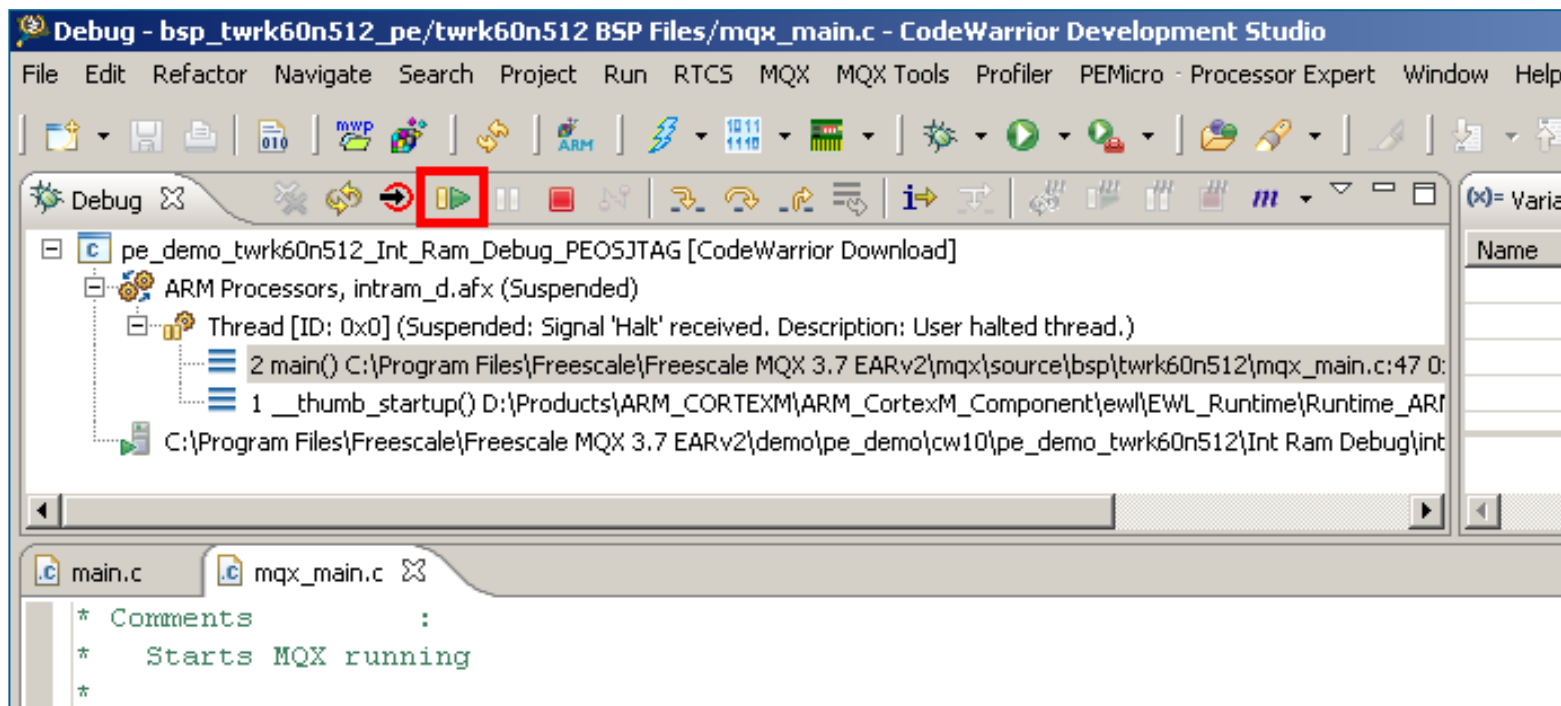
Run MQX PE Demo

- Click on the Debug icon.



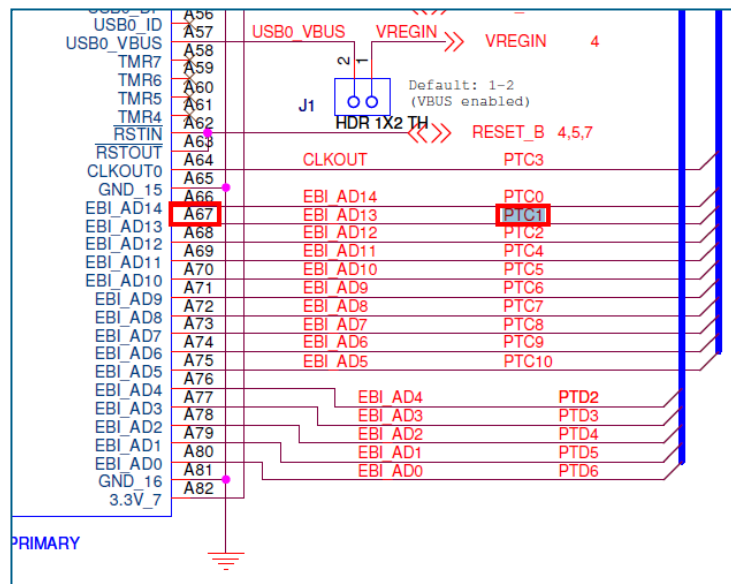
Run MQX PE Demo

- Click on the Resume (F8).

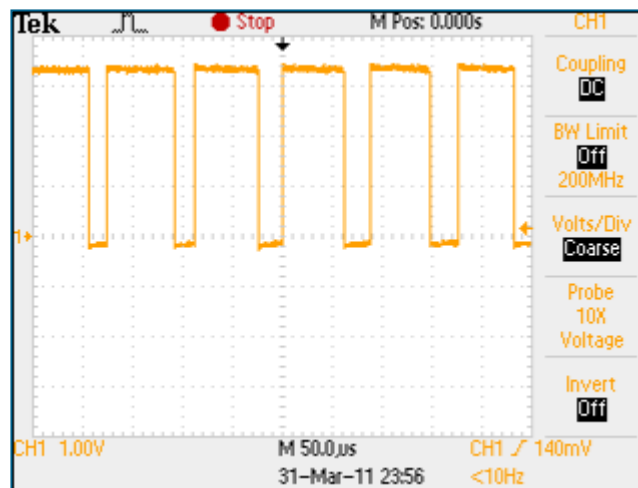


Run MQX PE Demo

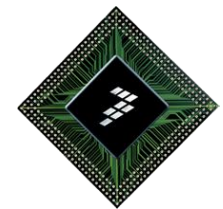
► Check PWM output on A67.



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		
Counter frequency	Internal	48 MHz
Counter restart		
Period device	FTM0_MOD	FTM0_MOD
Period	4096 timer-ticks	4096 timer-ticks
Interrupt		
Interrupt priority	medium priority	8
Channel list		
Channel 0		
Mode		
Compare	FTM0_C0V	FTM0_C0V
Offset	1 timer-ticks	1 timer-ticks
Output on compare		
Output on overrun	Set	
Initial state	Low	
Output pin	PTC1/SPI0_PCS3/UART1_RTS_b/FTM...	PTC1/SPI0_PCS3/UART1_RTS_b/FTM...
Interrupt		
Enabled in init. code	no	

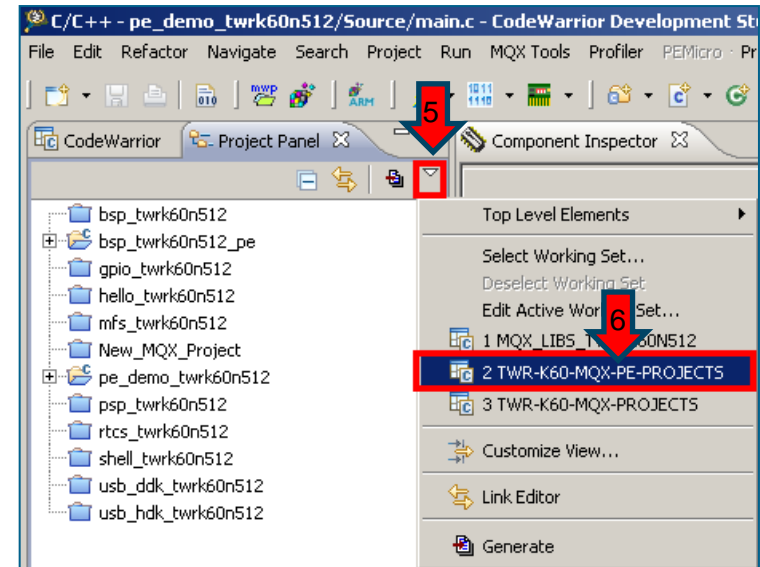
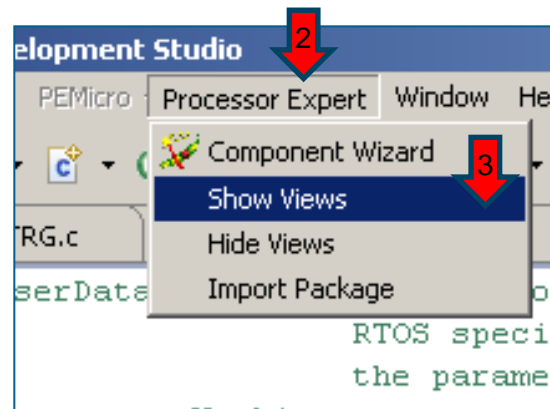
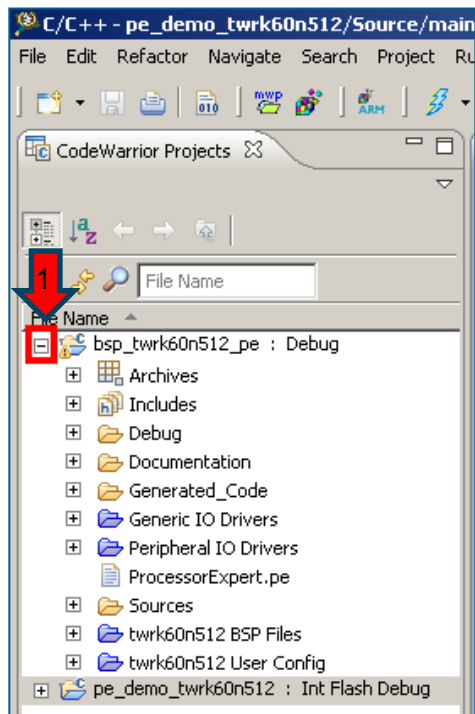


CW10.x, MQX RTOS and PE : New LDD driver



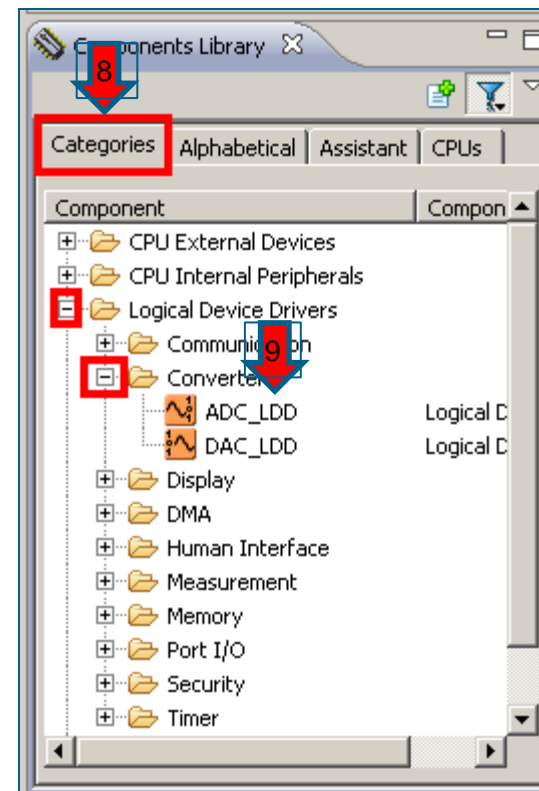
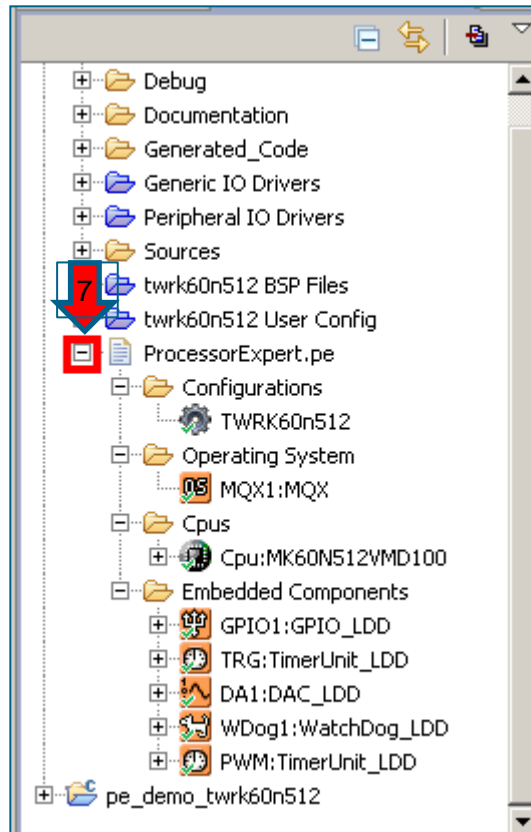
New LDD Driver

- ▶ Expand **bsp_twrk60n512** project view.
- ▶ Show Processor Expert View.
- ▶ Select PE Projects Working Set.



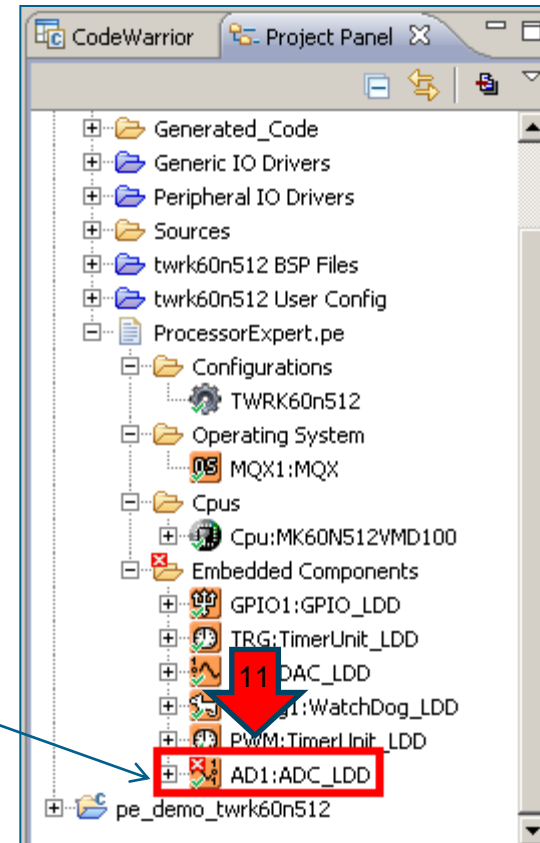
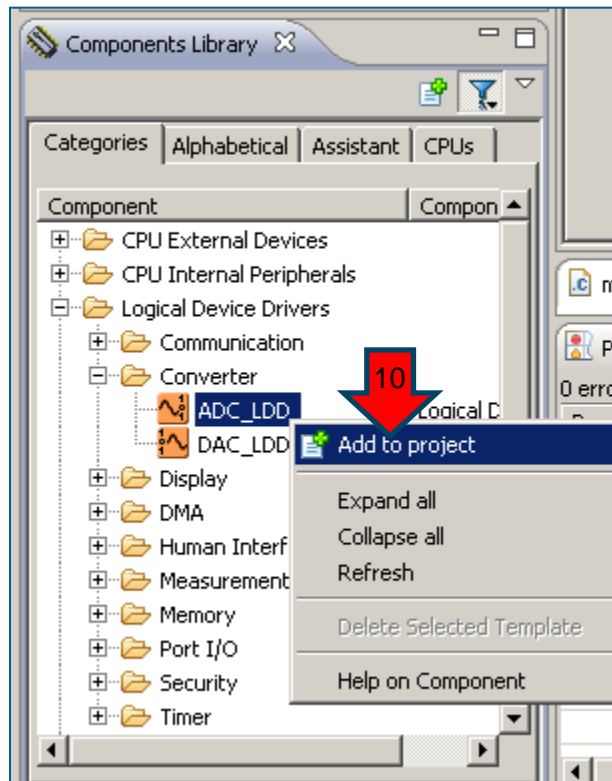
New LDD Driver

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



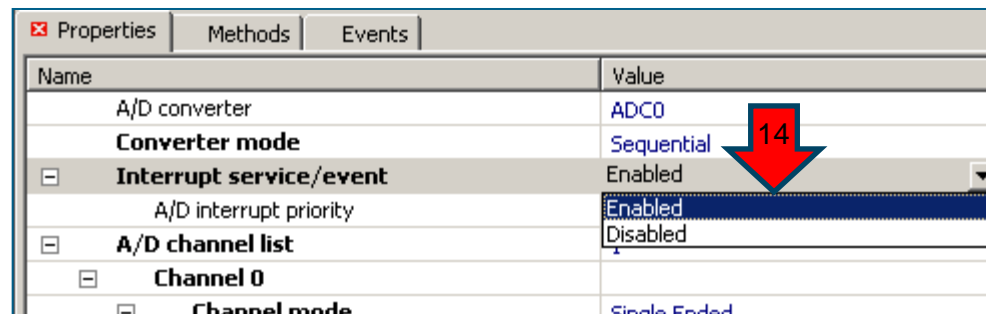
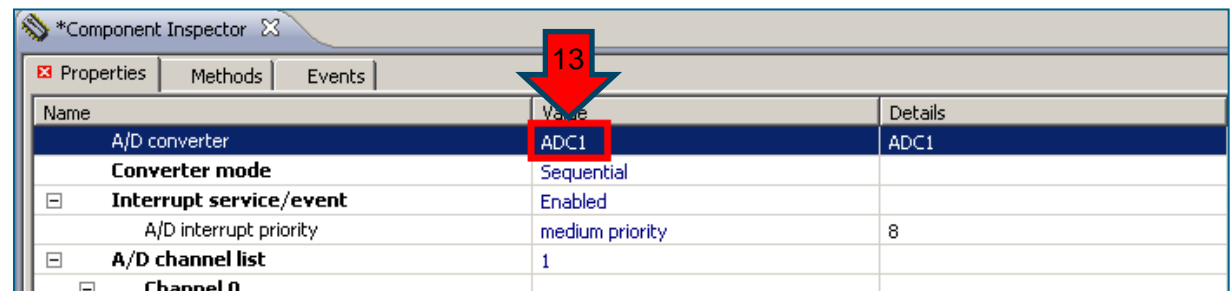
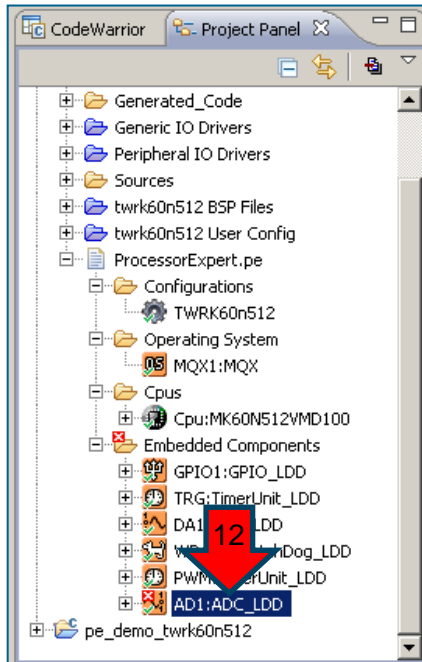
New LDD Driver

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



New LDD Driver

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



New LDD Driver

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

[-] 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	Sample group settings.
Channel index	0	D

A/D channel (pin)	ADC1_DM1	ADC1_DM1
[+] Static sample groups	Disabled	
A/D resolution	Autoselect	16 bits
! Conversion time		...
ADC clock		Unassigned timing

New LDD Driver

- Select **19.23** us.

[illegible]

► ADC LDD Driver is configured.

Properties	Methods	Events
Name	Value	Details
A/D converter	ADC1	ADC1
Converter mode	Sequential	
<input checked="" type="checkbox"/> Interrupt service/event	Enabled	
A/D interrupt priority	medium priority	8
<input checked="" type="checkbox"/> A/D channel list	1	
<input checked="" type="checkbox"/> Channel 0		
<input checked="" type="checkbox"/> Channel mode	Single Ended	
<input checked="" type="checkbox"/> Input		
A/D channel (pin)	ADC1_DM1	ADC1_DM1
<input checked="" type="checkbox"/> Static sample groups	Enabled	
<input checked="" type="checkbox"/> Sample group list	1	
<input checked="" type="checkbox"/> Group 0		
<input checked="" type="checkbox"/> Sample list	1	
<input checked="" type="checkbox"/> 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	
<input checked="" type="checkbox"/> Initialization		This property allows to select one of result
Enabled in init. code	yes	Description for the current value (unsigned
<input checked="" type="checkbox"/> Event mask		
OnMeasurementComplete	Enabled	

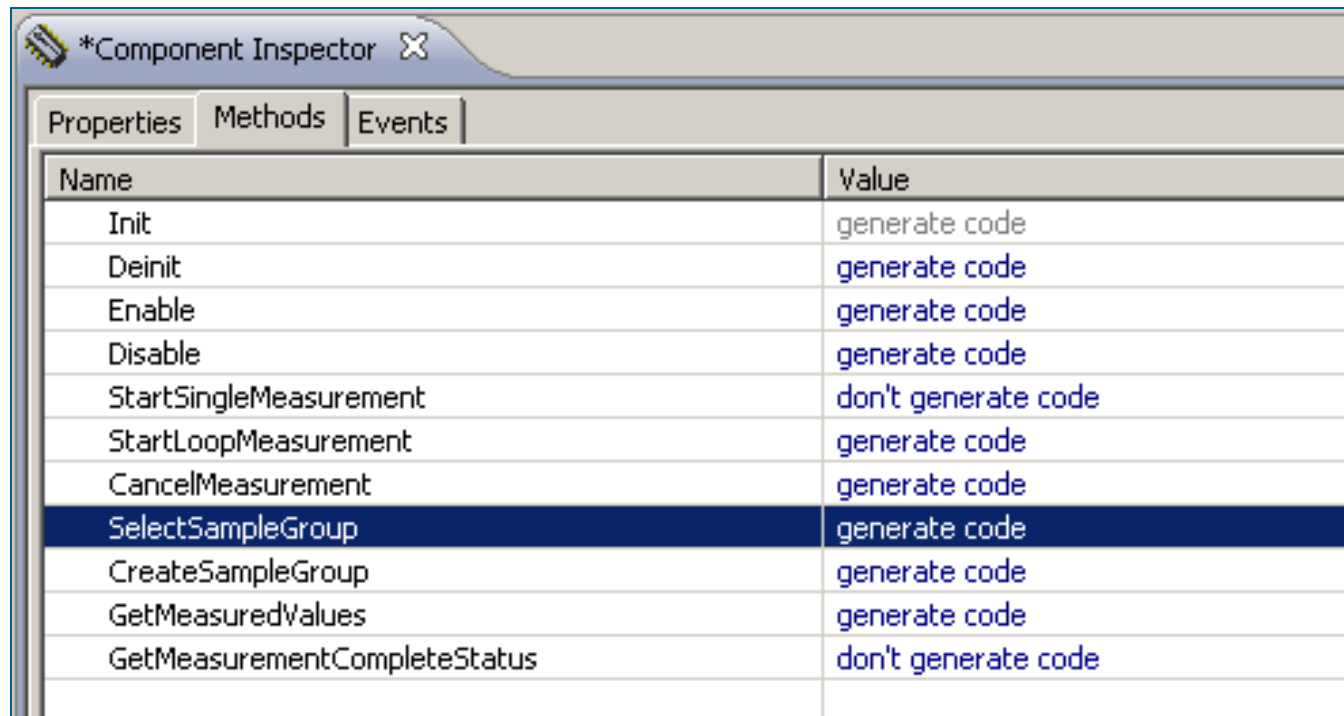
- ▶ Click Methods Tab.
- ▶ Click to generate code for methods.

*Component Inspector

Properties	Methods	Events
Name	Value	
Init	generate code	
Deinit	generate code	
Enable	don't generate code	
Disable	don't generate code	
StartSingleMeasurement	generate code	
StartLoopMeasurement	don't generate code	
CancelMeasurement	don't generate code	
SelectSampleGroup	don't generate code	
CreateSampleGroup	generate code	
GetMeasuredValues	generate code	
GetMeasurementCompleteStatus	don't generate code	

Properties	Methods	Events
Name	Value	
Init	generate code	
Deinit	generate code	
Enable	generate code	
Disable	generate code	
StartSingleMeasurement	generate code	
StartLoopMeasurement	don't generate code	
CancelMeasurement	generate code	
SelectSampleGroup	don't generate code	
CreateSampleGroup	generate code	
GetMeasuredValues	generate code	
GetMeasurementCompleteStatus	don't generate code	

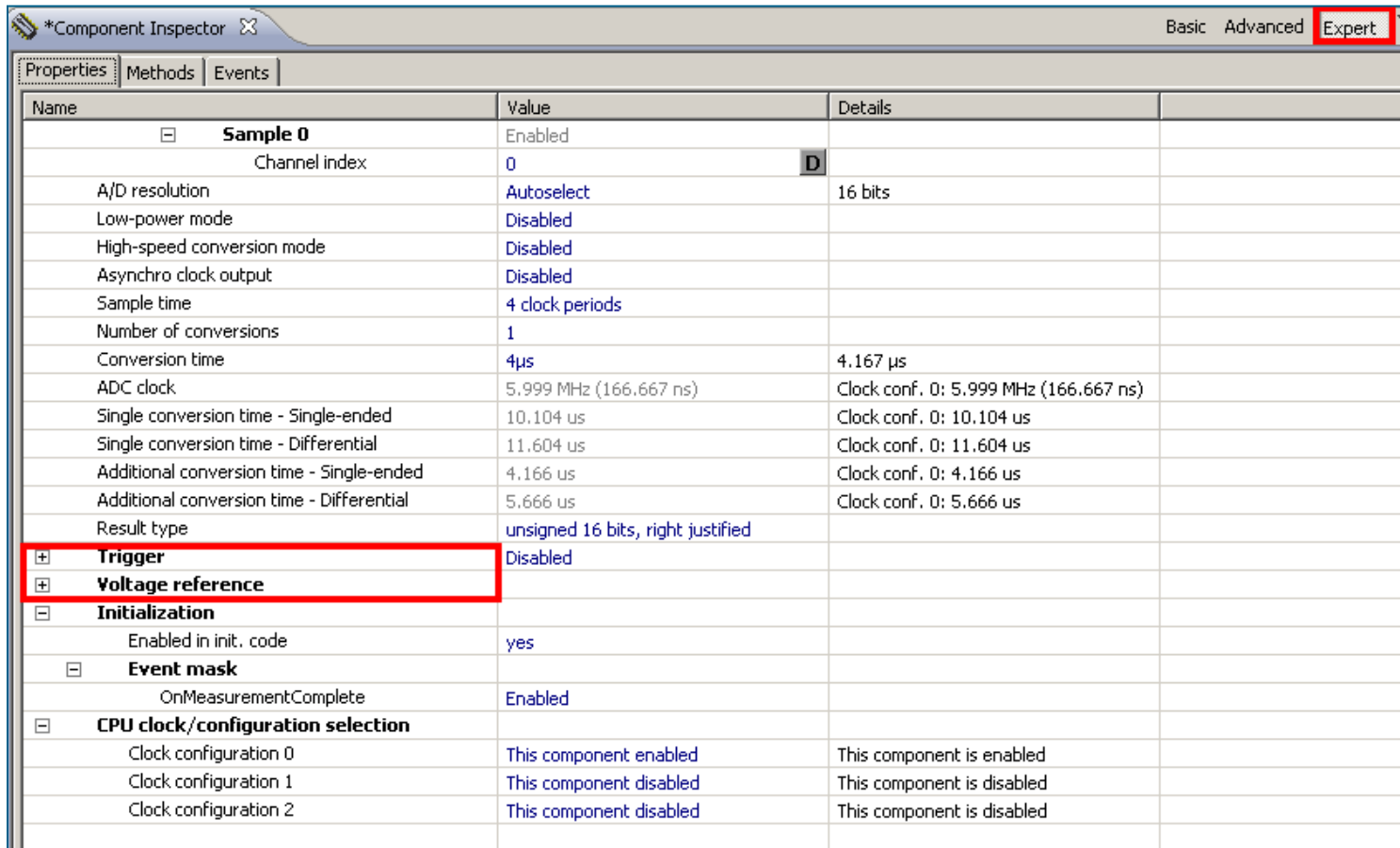
- Set 'generate code' for the next Methods:



The screenshot shows the *Component Inspector window with the Methods tab selected. The table below represents the data shown in the window.

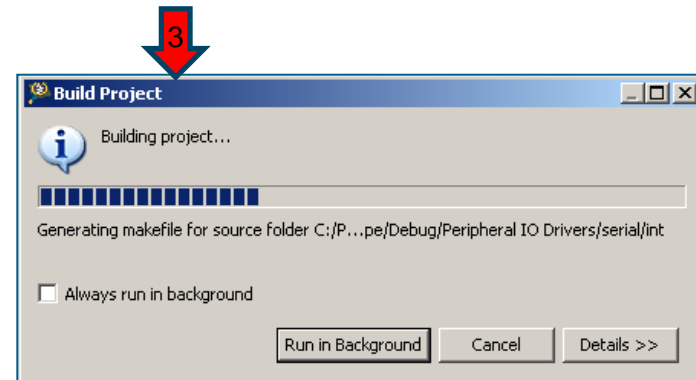
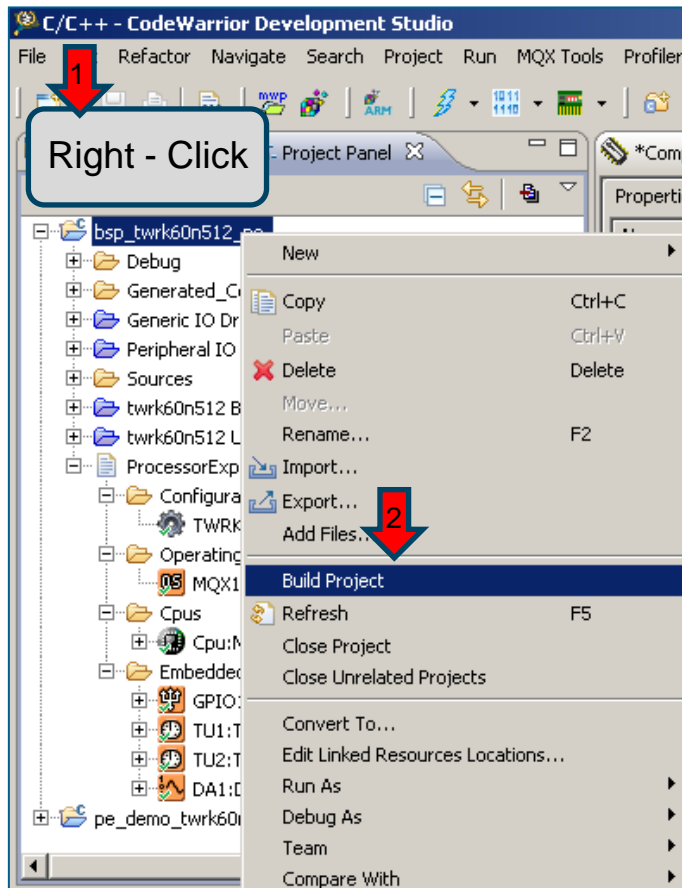
Name	Value
Init	generate code
Deinit	generate code
Enable	generate code
Disable	generate code
StartSingleMeasurement	don't generate code
StartLoopMeasurement	generate code
CancelMeasurement	generate code
SelectSampleGroup	generate code
CreateSampleGroup	generate code
GetMeasuredValues	generate code
GetMeasurementCompleteStatus	don't generate code

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



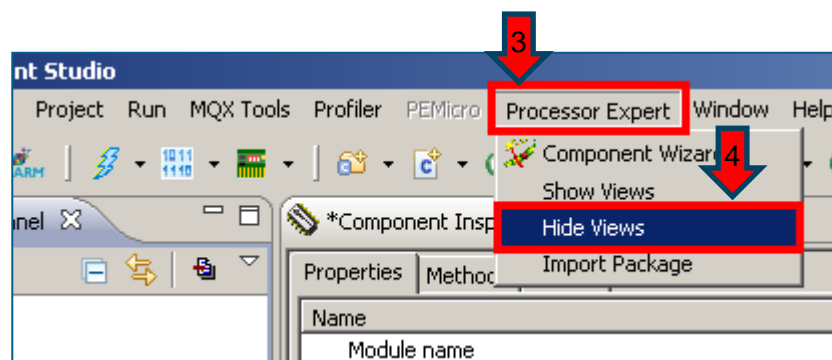
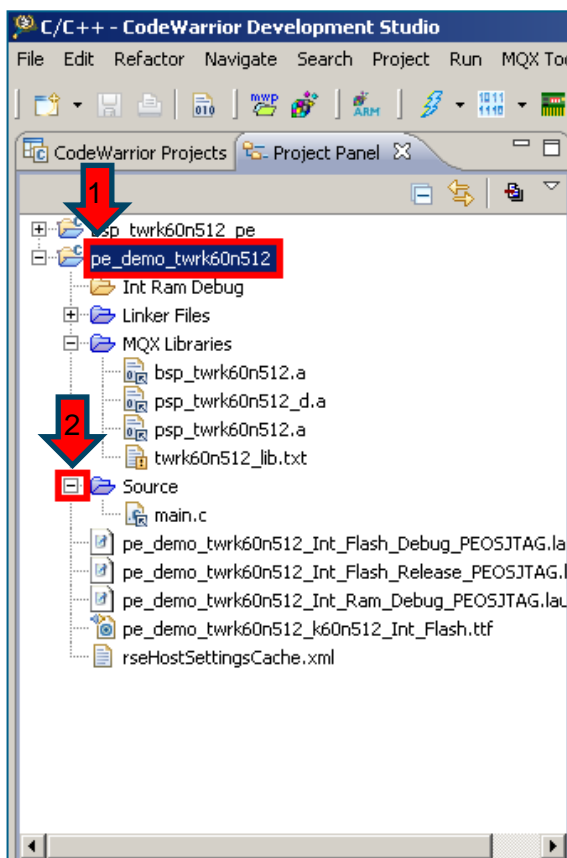
Build PE BSP

- Right-Click on the Project Explorer **bsp_twrk60n512** and Build Project.



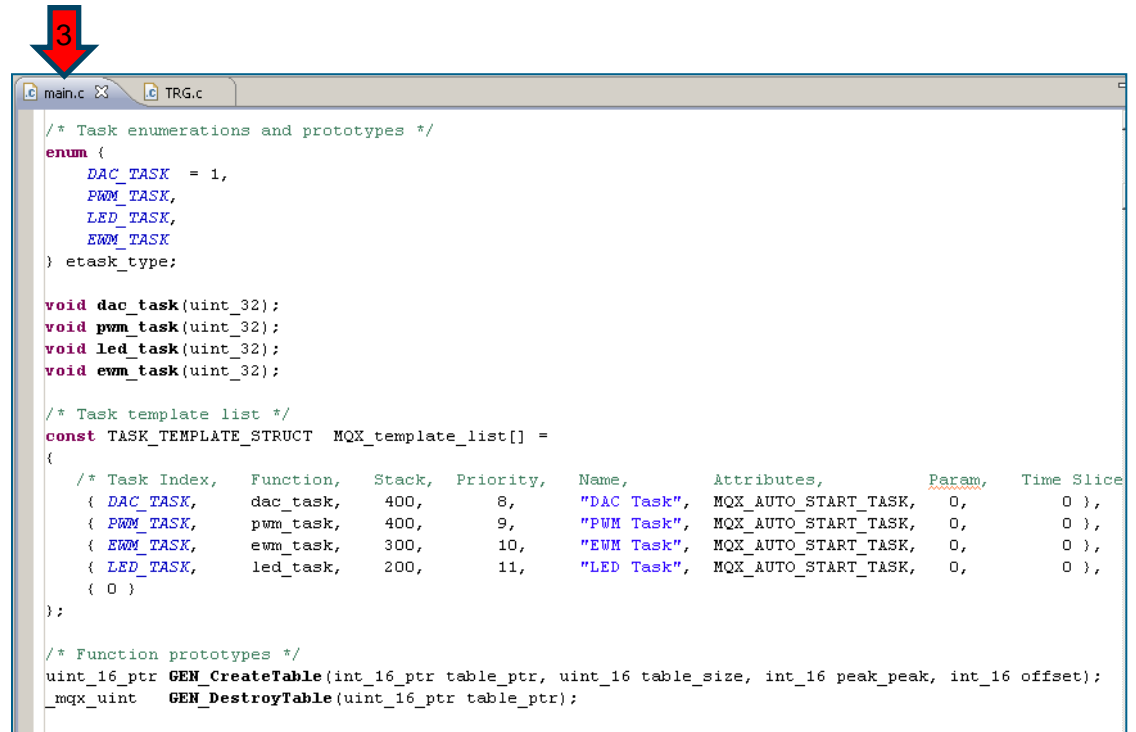
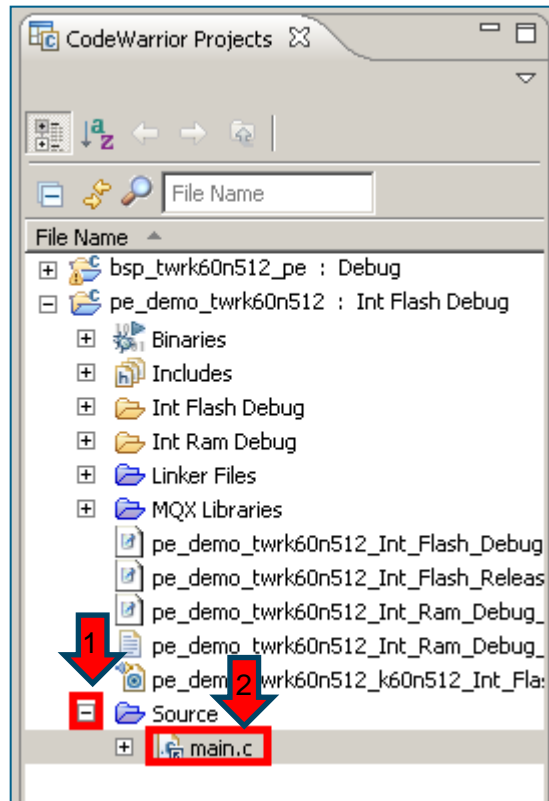
New LDD driver

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



New LDD Driver

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



► 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 */

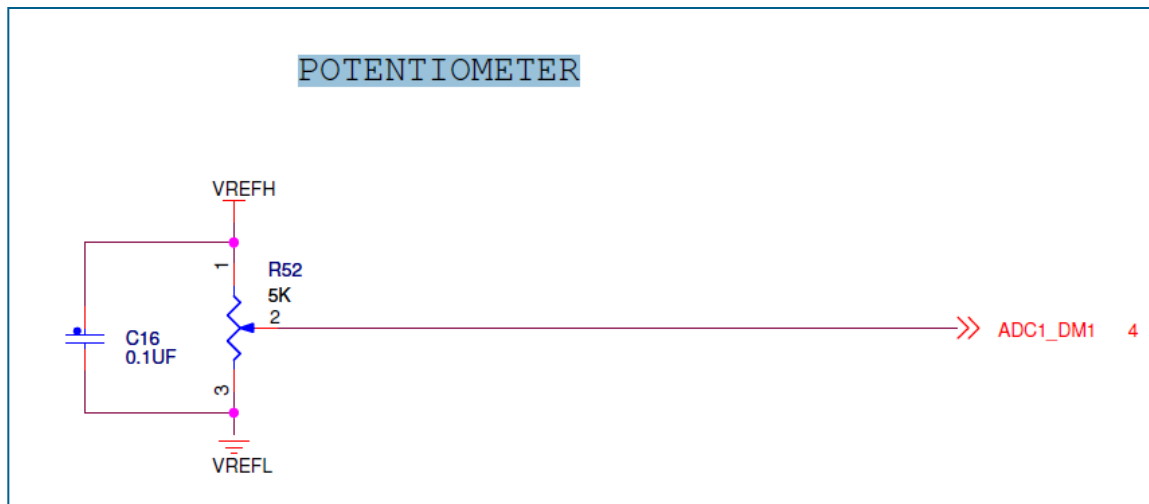
        if(MeasuredValues[0]>2000) GPIO1_ToggleFieldBits(LED_DeviceData, LED4, 1);
        _time_delay(200);
    }
}
```

- 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.



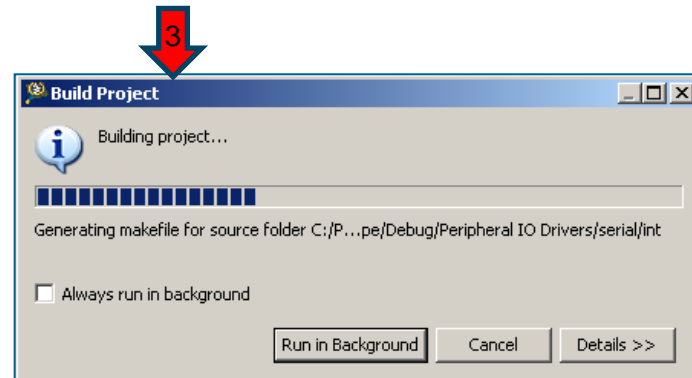
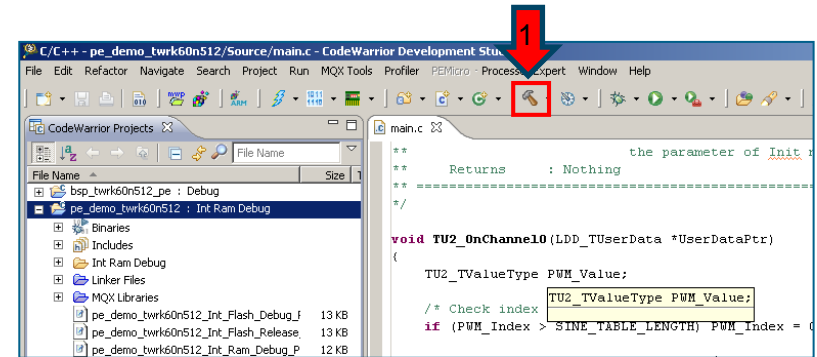
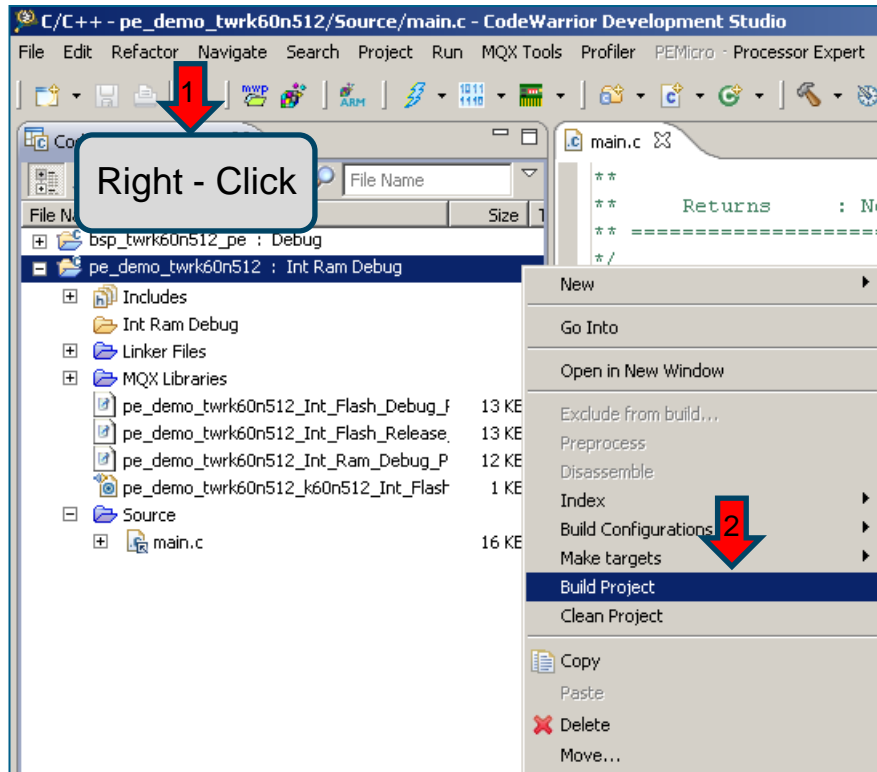
- ▶ When ADC value is greater than 20000, LED4 (Blue) toggles.

```
while(1)
{
/* Suspend task for 100ms */
    if (MeasuredValues[0]>20000) GPIO1_ToggleFieldBits(LED_DeviceData, LED4, 1);
    _time_delay(200);
}
```

- ▶ Moving potentiometer R52 can start/stop LED4 toggle.

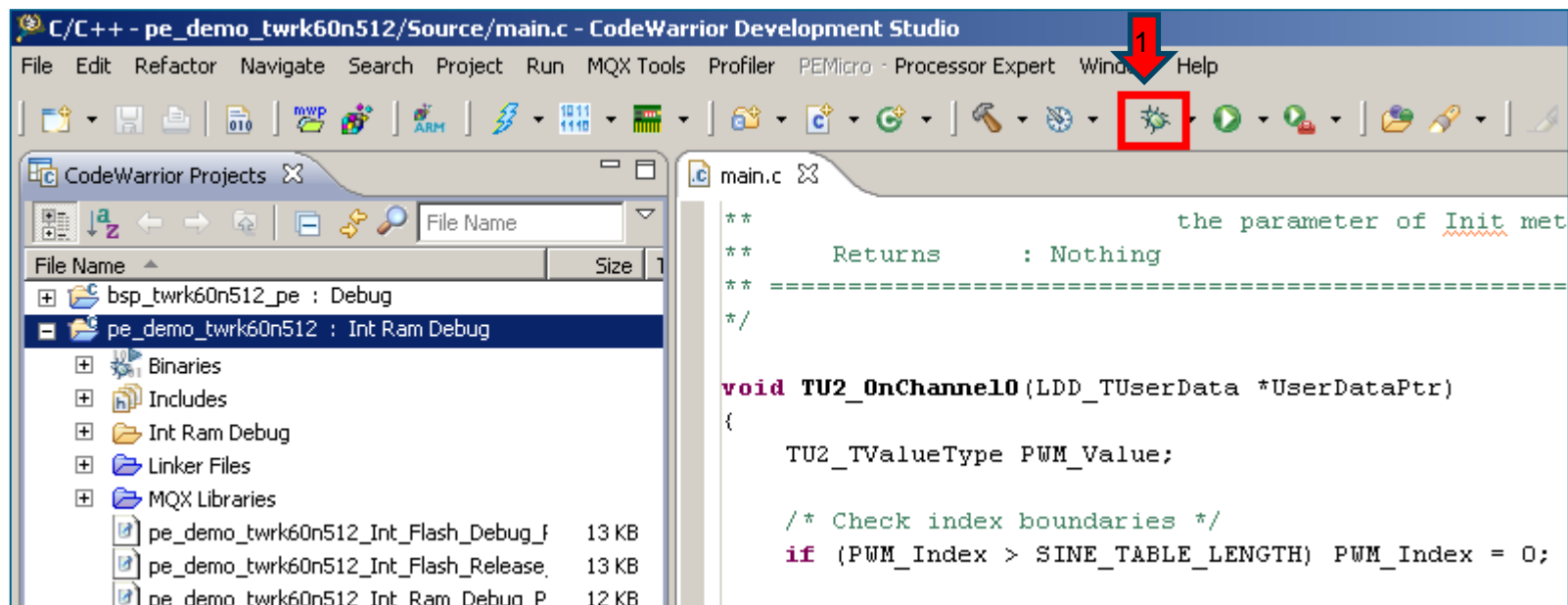
Build PE Demo

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



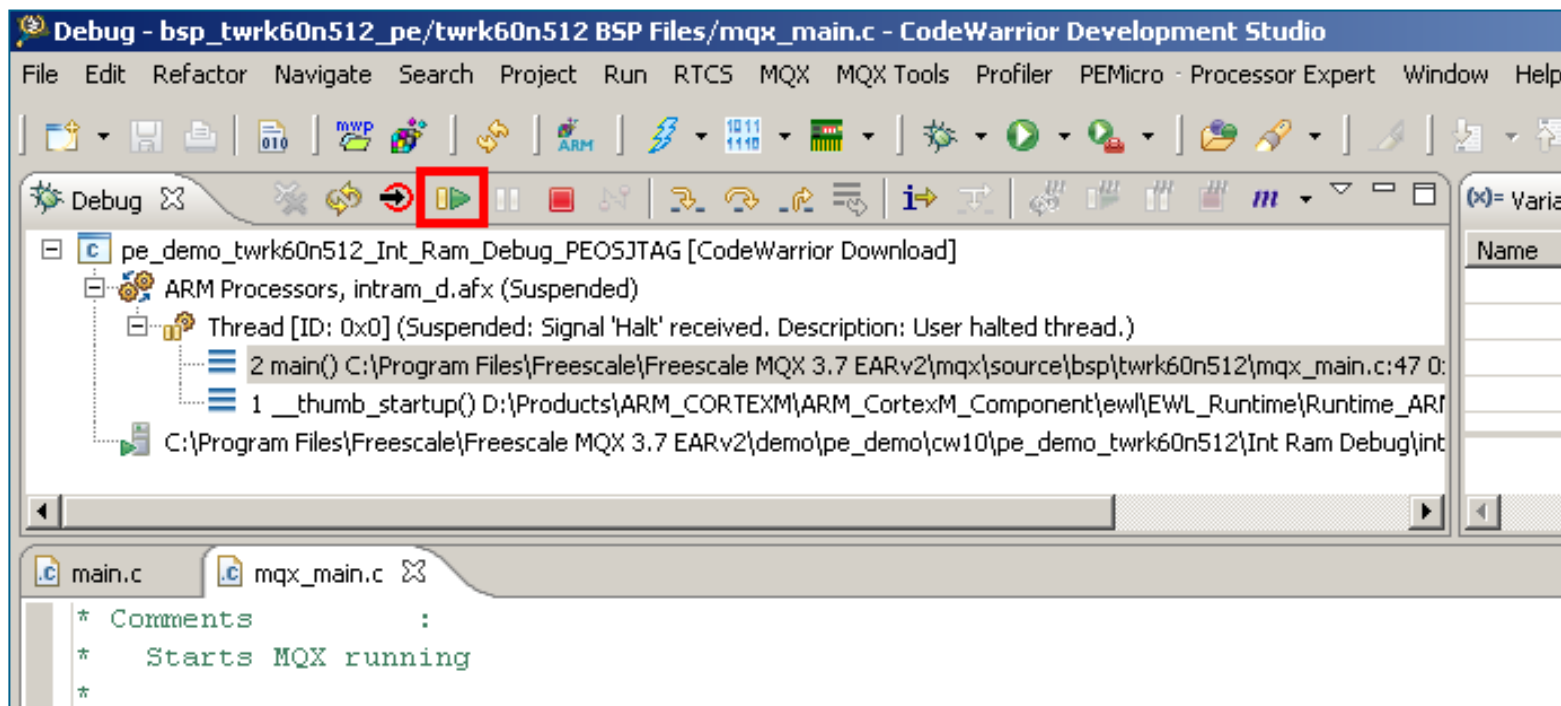
Run MQX PE Demo

- Click Debug icon.



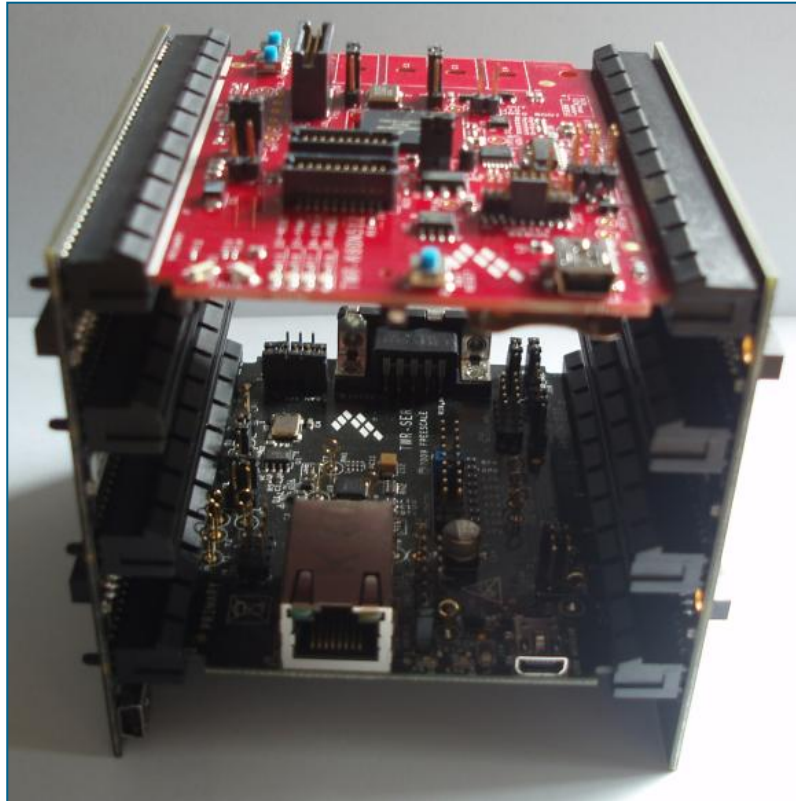
Run MQX PE Demo

- Click Resume (F8).



New LDD 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

The screenshot displays the Freescale Infocenter website. At the top left is the Freescale Semiconductor logo. The main heading is "Welcome to the Freescale Infocenter". Below this is a search bar with a "GO" button and a "Search scope: All topics" link. A "Contents" sidebar on the left lists various resources, including "CodeWarrior for Microcontrollers V10.x" and "Processor Expert Manuals". The main content area shows the breadcrumb path: "CodeWarrior for Microcontrollers V10.x > Processor Expert Manuals > Processor Expert Components Manual > Installed Components". The selected component is "Init_MDHA", with sub-links for "General Info", "Properties", "Methods", and "Init Code Usage". The "Component Init_MDHA for MCF" section describes the "Message Digest Hardware Accelerator", noting its component level as "Peripheral Initialization" and category as "CPU Internal Peripherals-Peripheral Initialization". It states that this component provides initialization of the MDHA module and that peripheral initialization components are intended for experienced users. A footer note mentions "PROCESSOR EXPERT is trademark of Freescale Semiconductor, Inc." and "Copyright 1997 - 2010 Freescale Semiconductor, Inc."

How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, Kinetis, Processor Expert, and CodeWarrior are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Tower is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM and ARM Powered logo are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.
© 2013 Freescale Semiconductor, Inc.

Document Number MQXCWPP
08/2014

