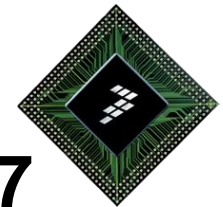


March 2011

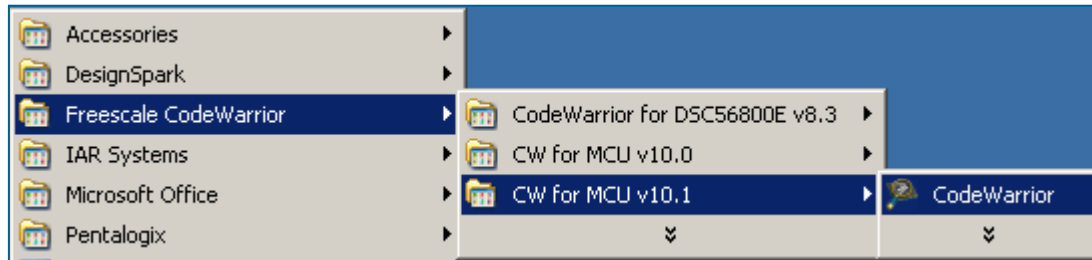
CW for Microcontrollers v10.1 and MQX 3.7



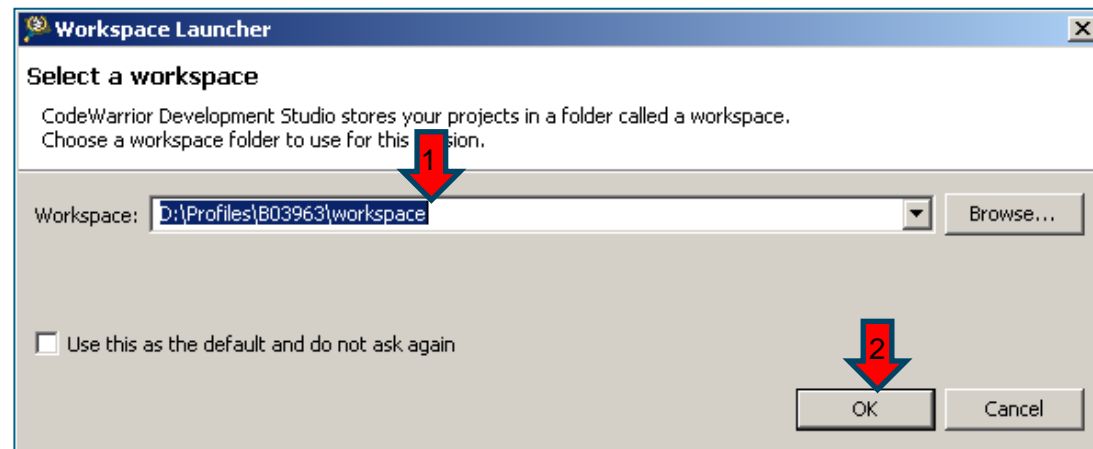
Luis Casado
EMEA FAE

- ▶ Import MQX Libraries
- ▶ Build MQX libraries
- ▶ Import and Debug MQX Hello World Project
- ▶ New MQX project
- ▶ Debugging with J-Link
- ▶ Eclipse Working Set
- ▶ CW10.1, MQX 3.7 and Processor Expert
- ▶ CW10.1, MQX 3.7 and PE : New LDD driver

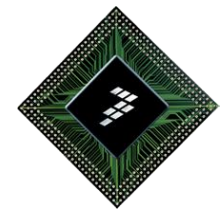
► Open CodeWarrior for MCU v10.1



► Select your Workspace and press OK

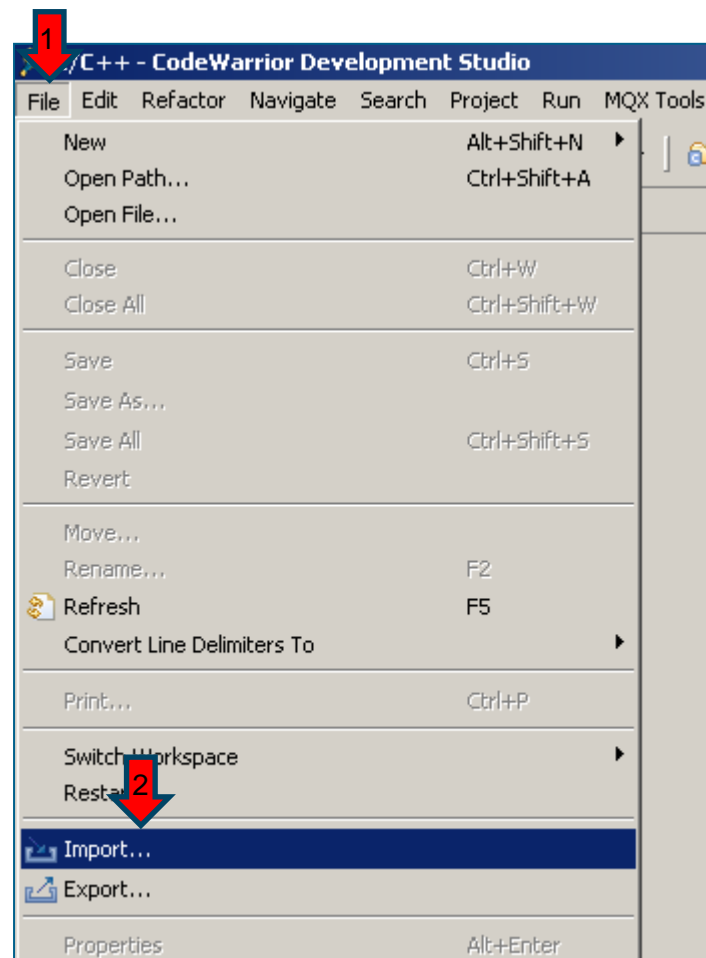
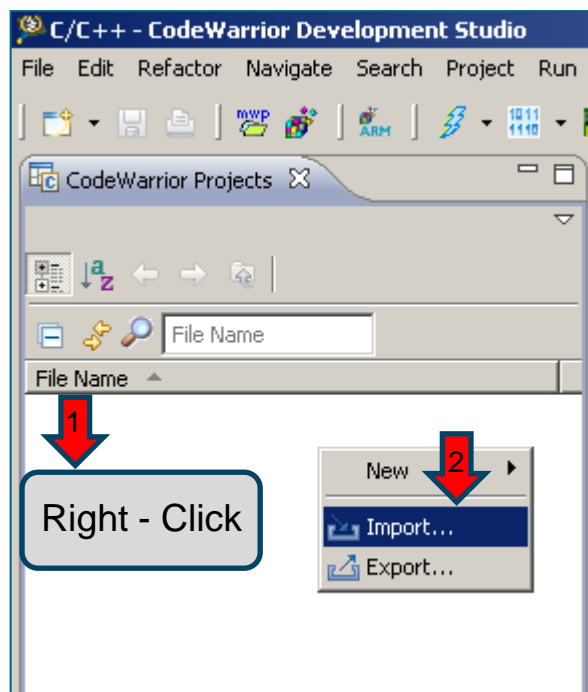


Import MQX Libraries

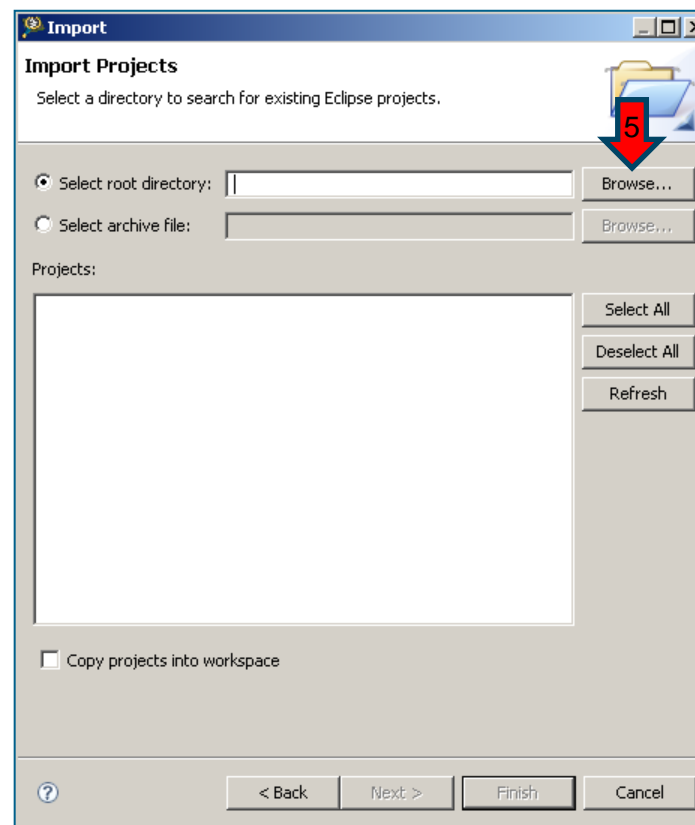
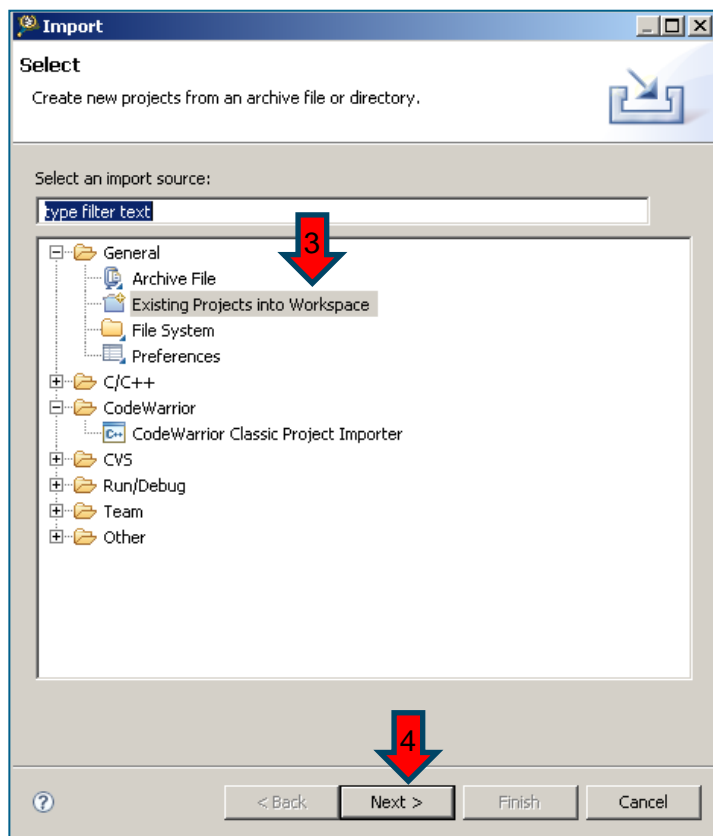


Import MQX Libraries

- Right-Click on Project Explorer and Import (or) File -> Import

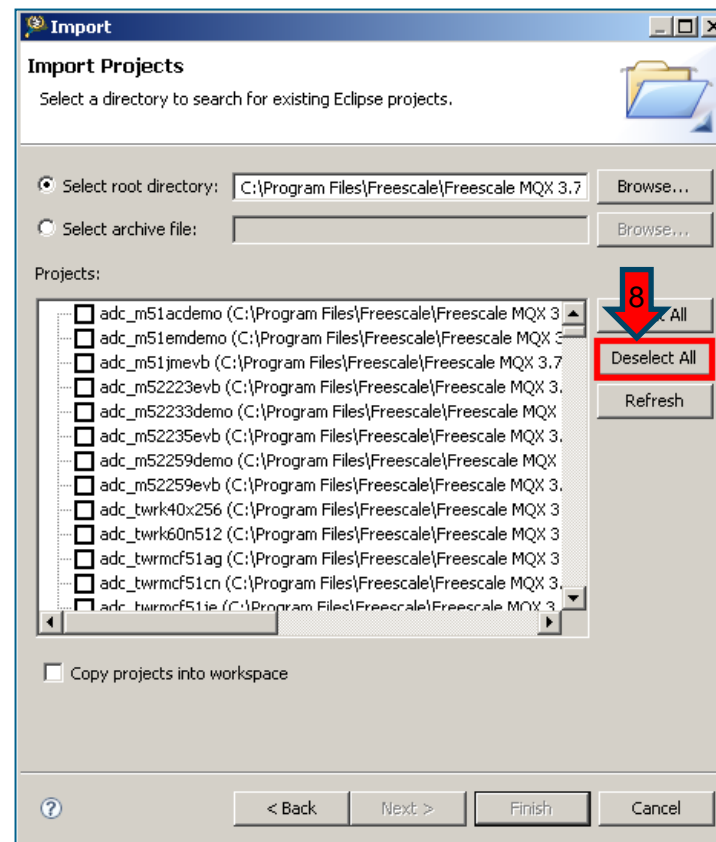
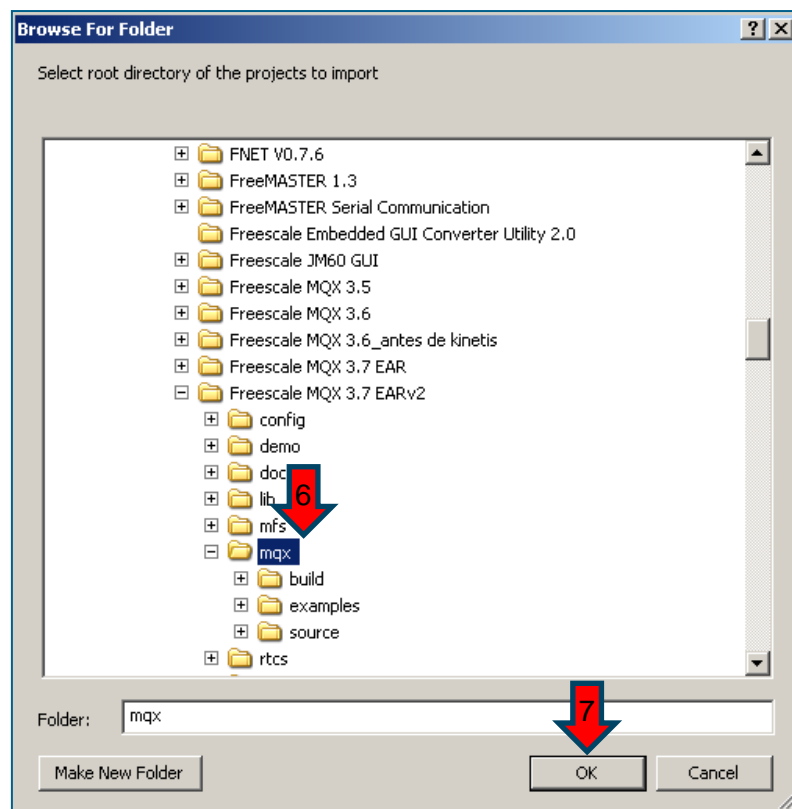


► Select Existing Projects into Workspace and Browse

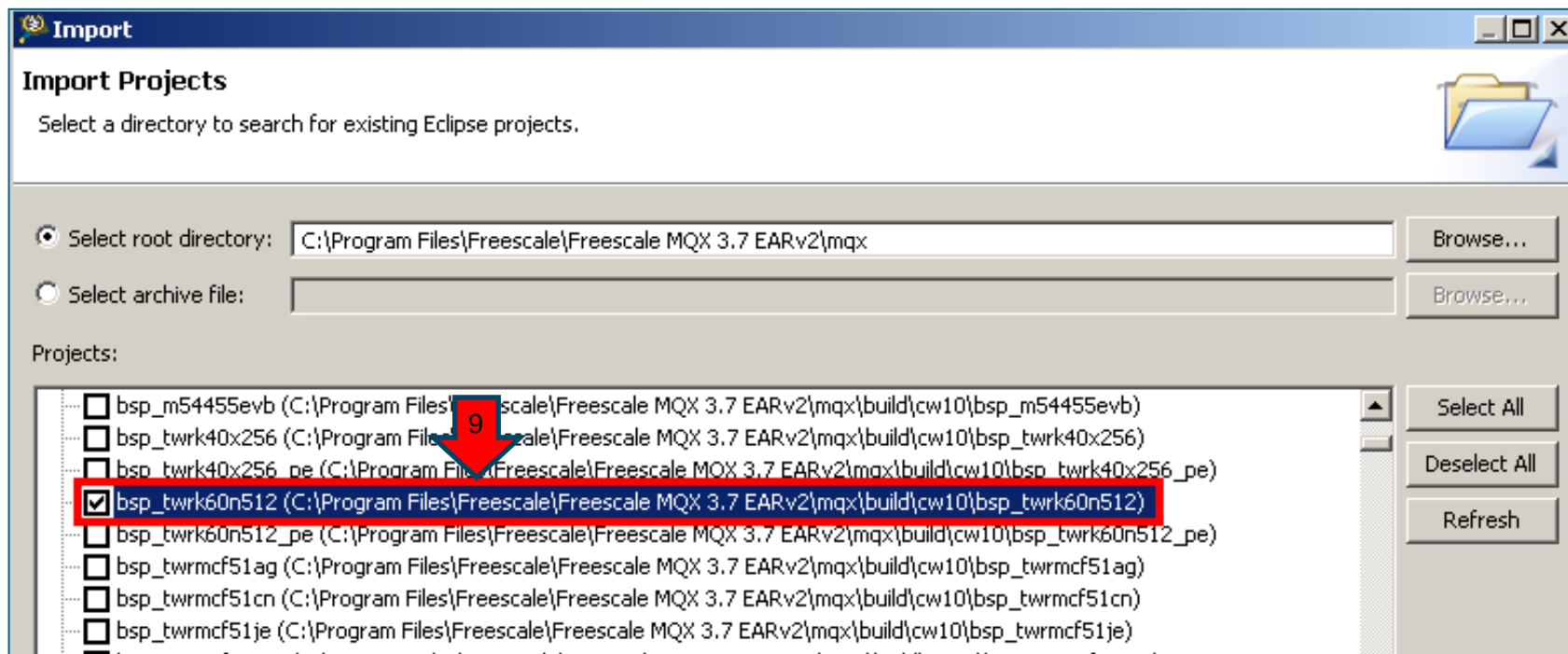


Import MQX Libraries

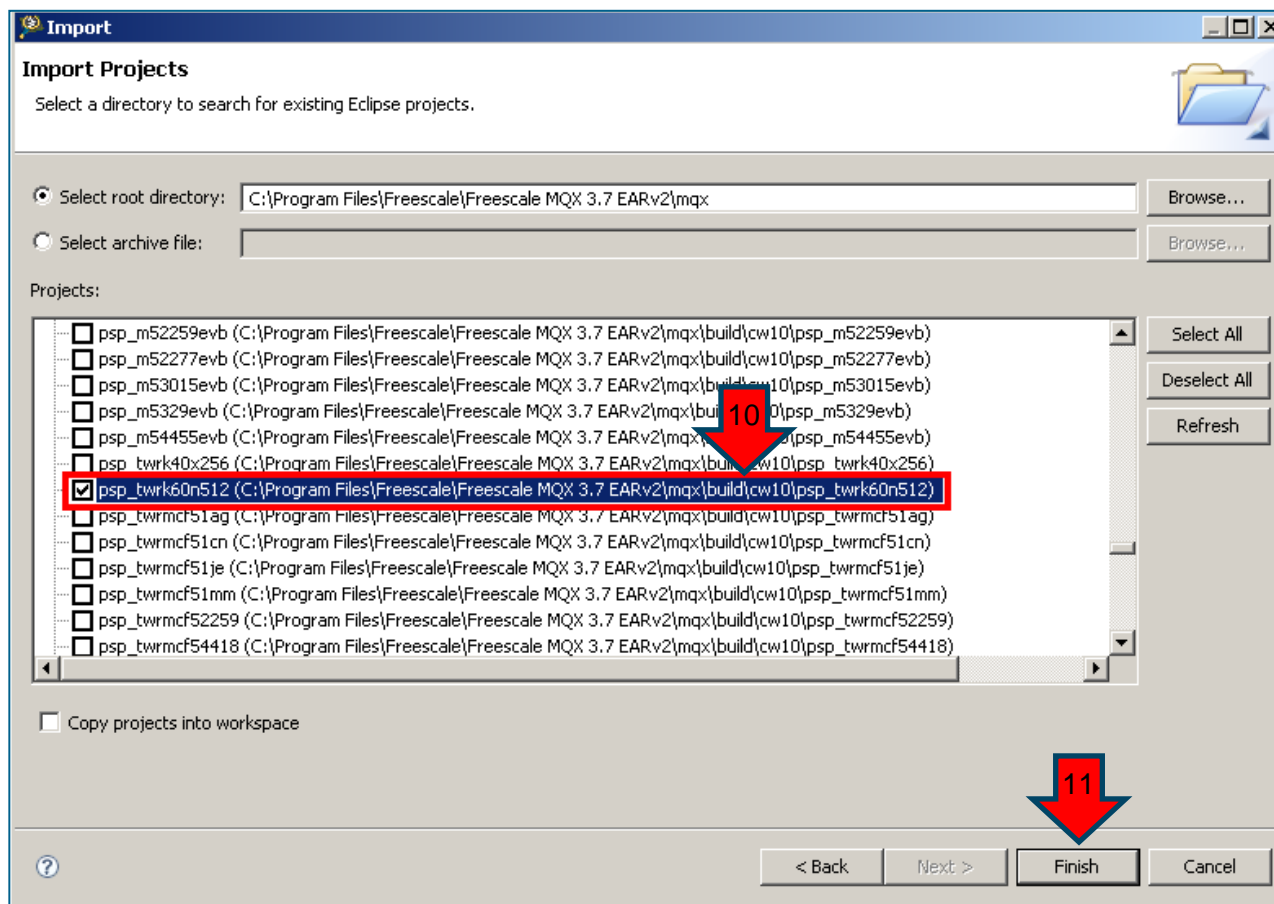
- Select *<install mqx folder>\mqx*



► Select **bsp_twrk60n512** project

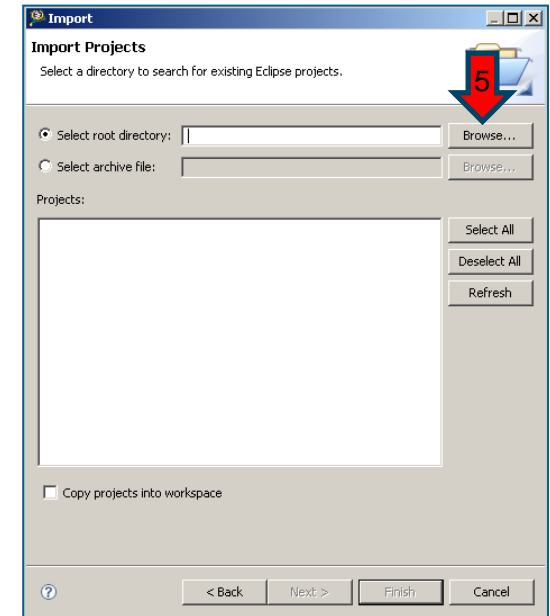
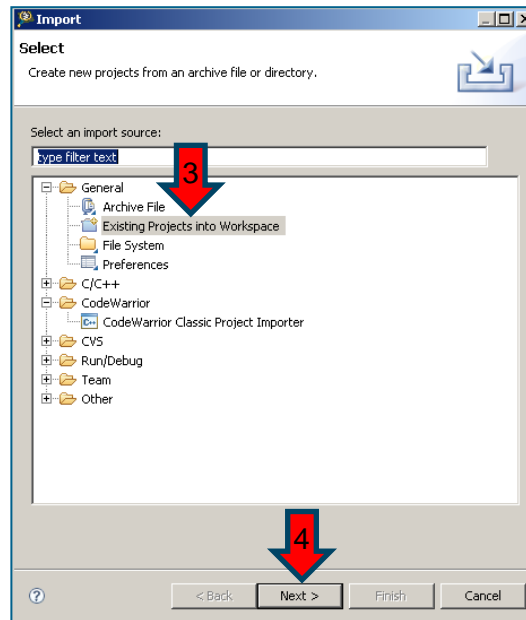
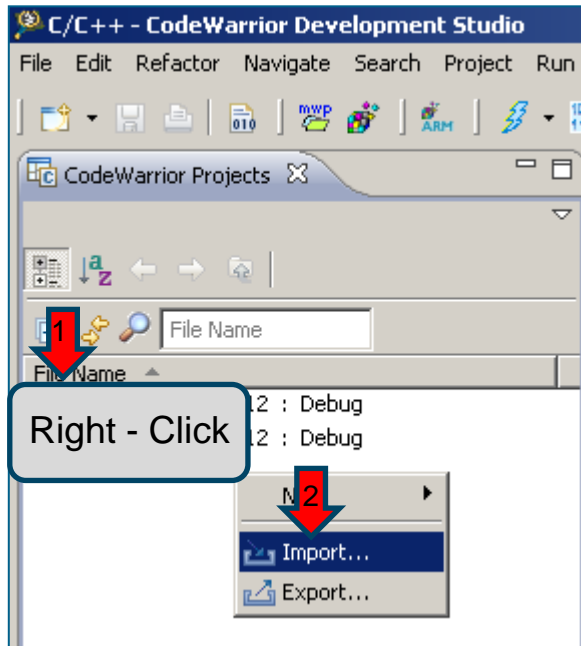


► Select **psp_twrk60n512** project



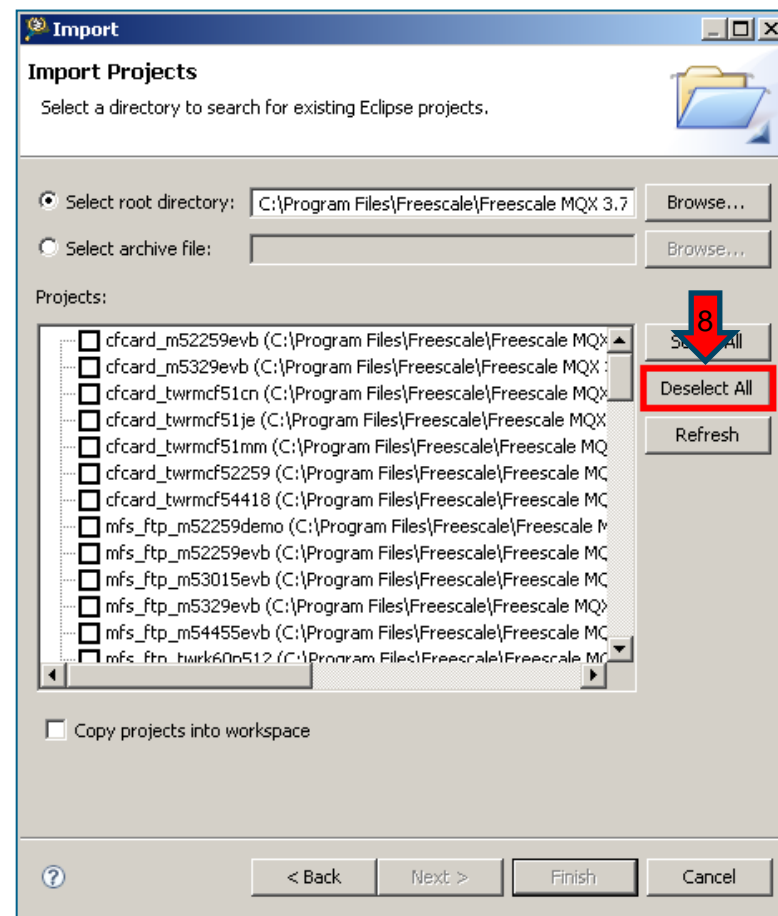
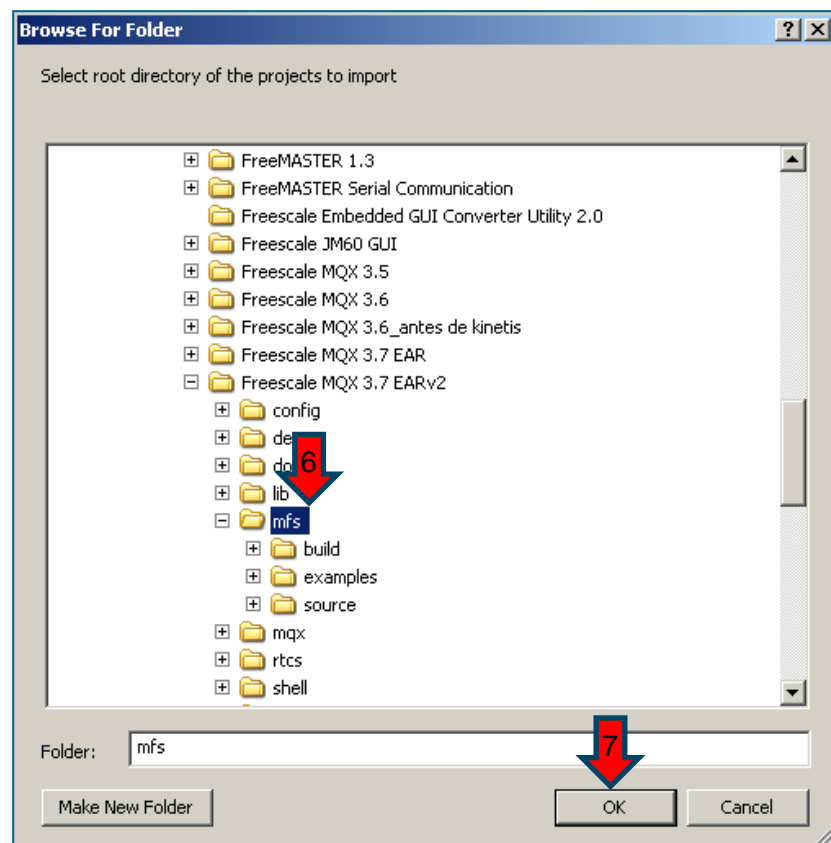
Import MQX Libraries

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

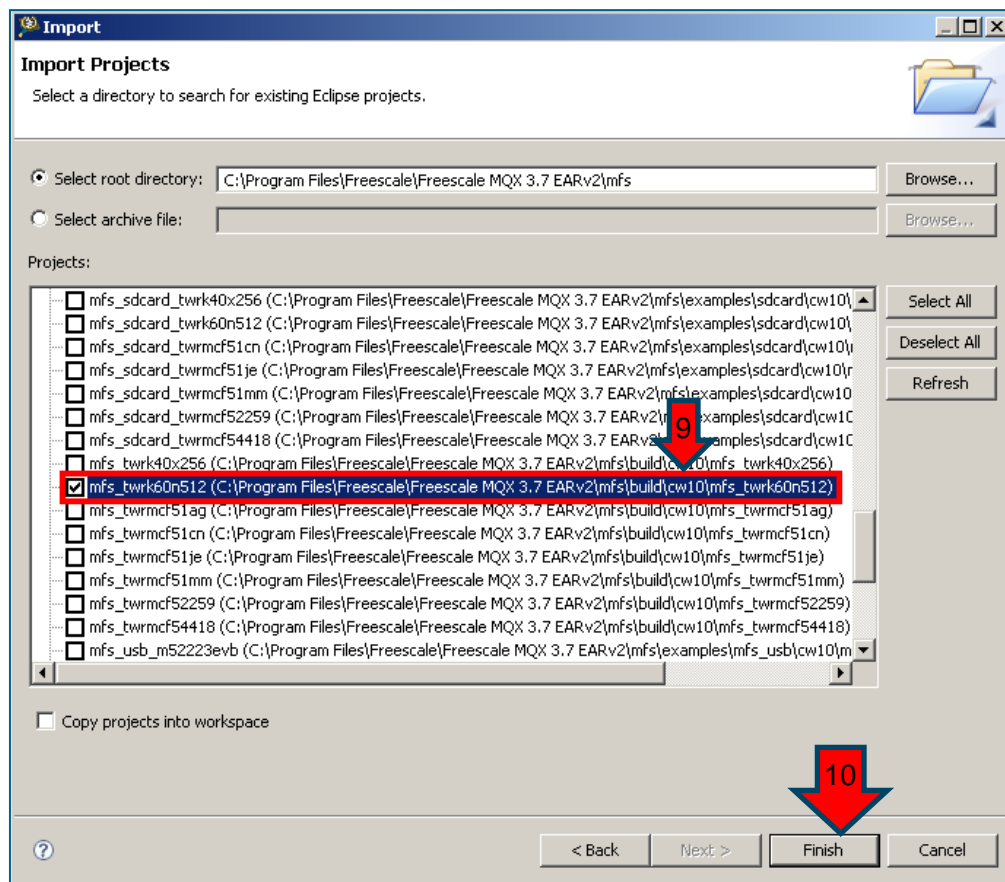


Import MQX Libraries

► Select *<install mqx folder>\mfs*

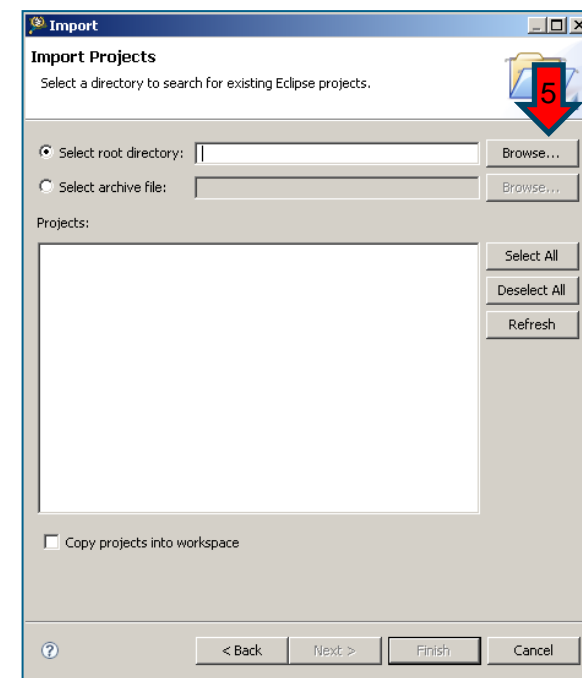
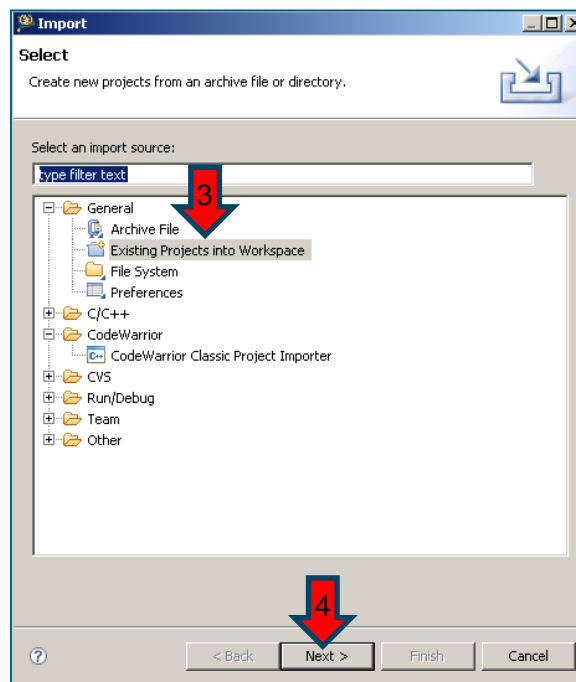
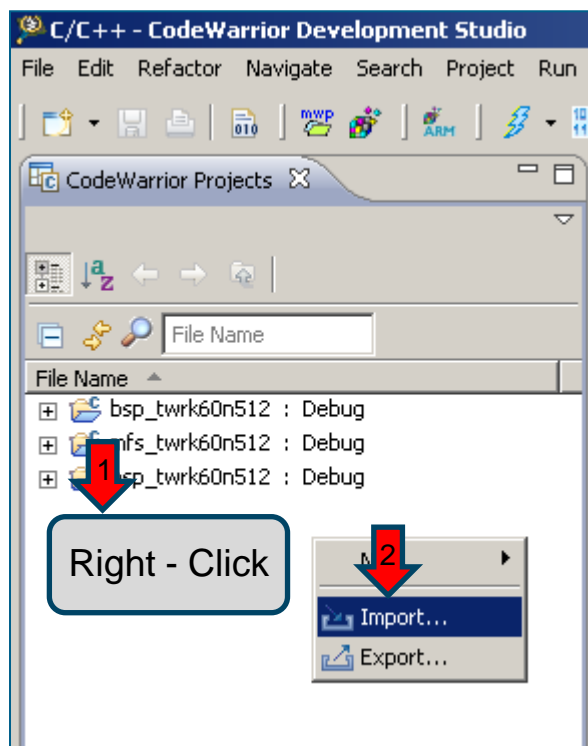


► Select **mfs_twrk60n512** project



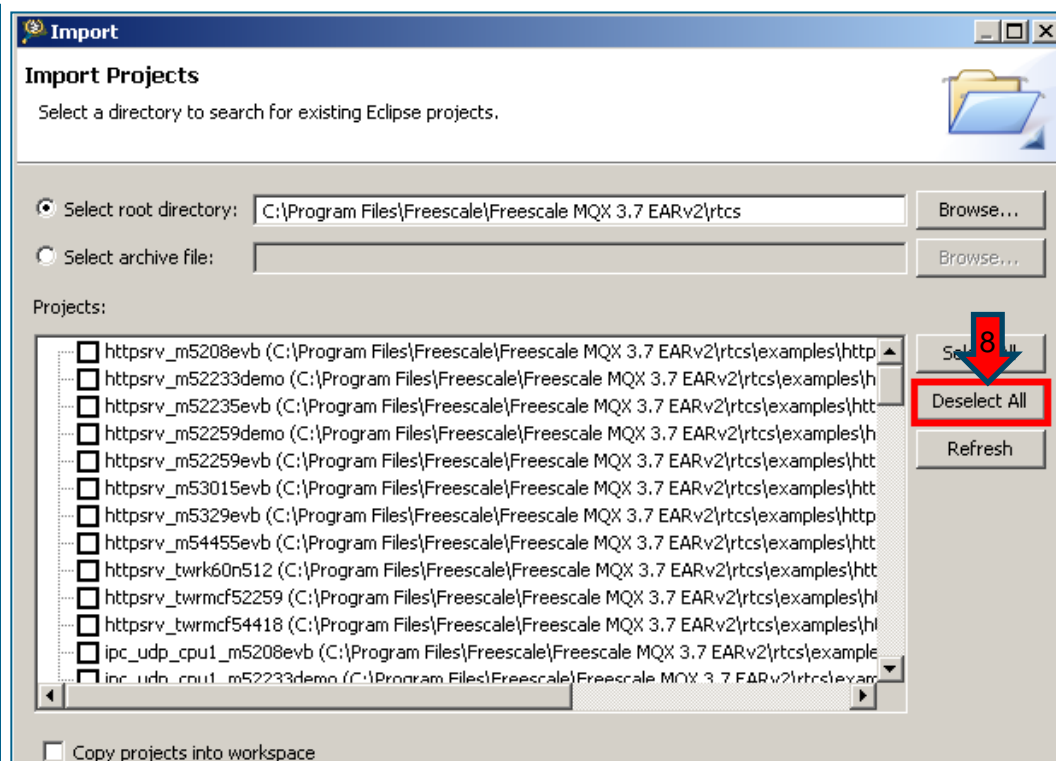
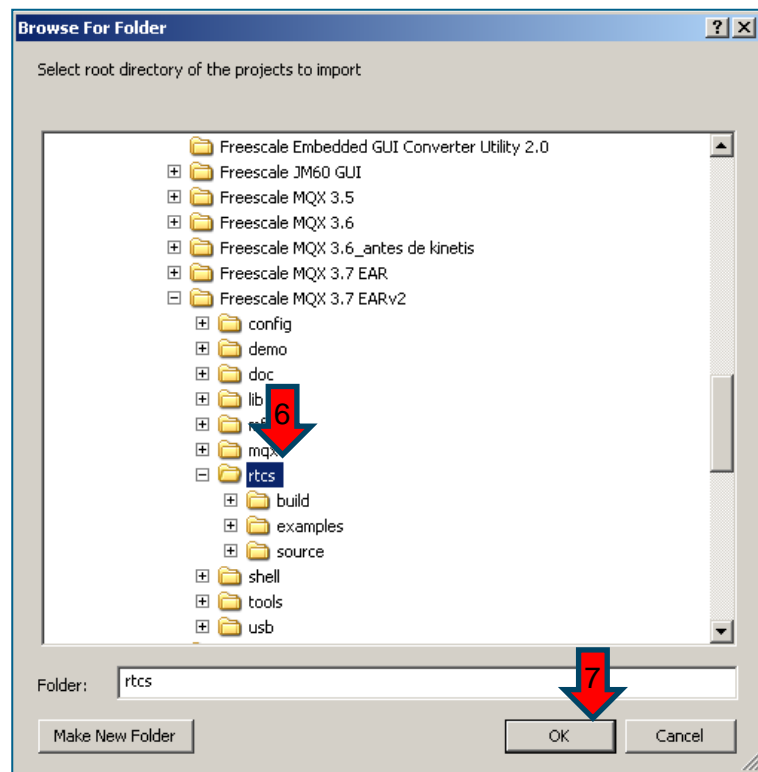
Import MQX Libraries

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

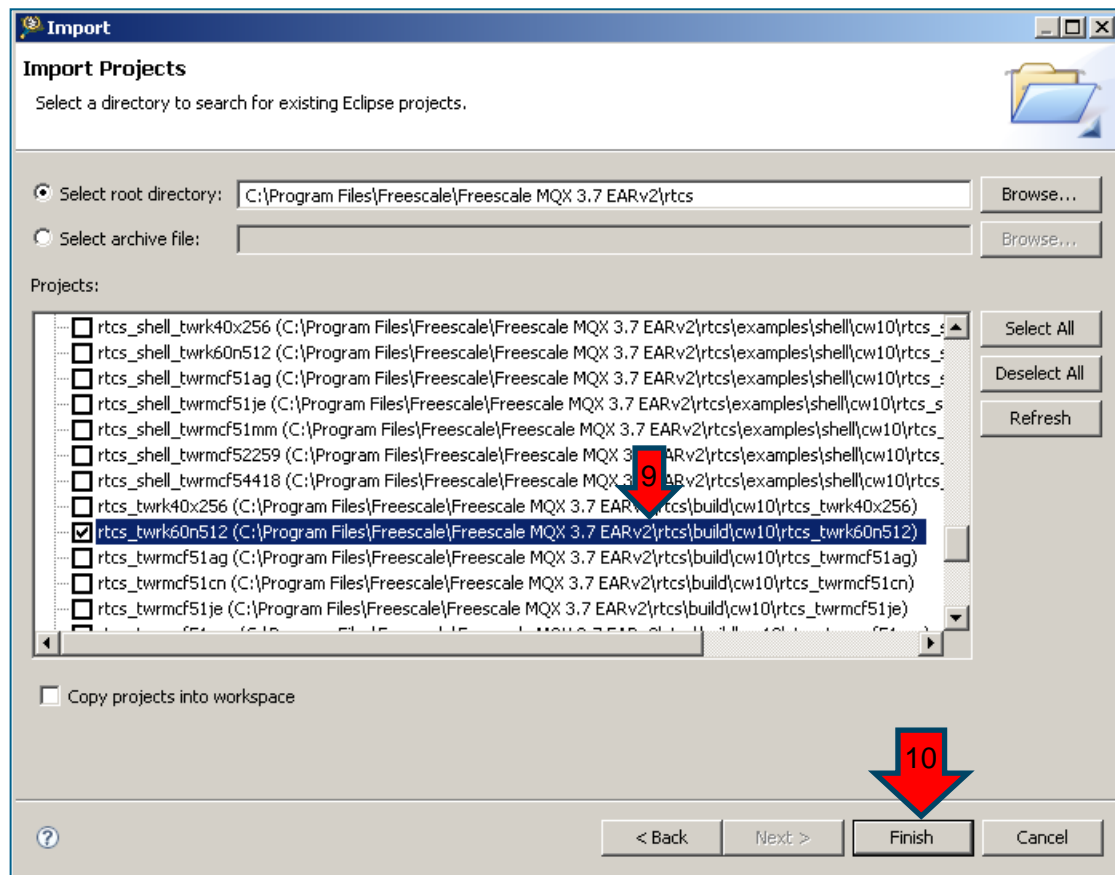


Import MQX Libraries

► Select *<install mqx folder>* rtcs

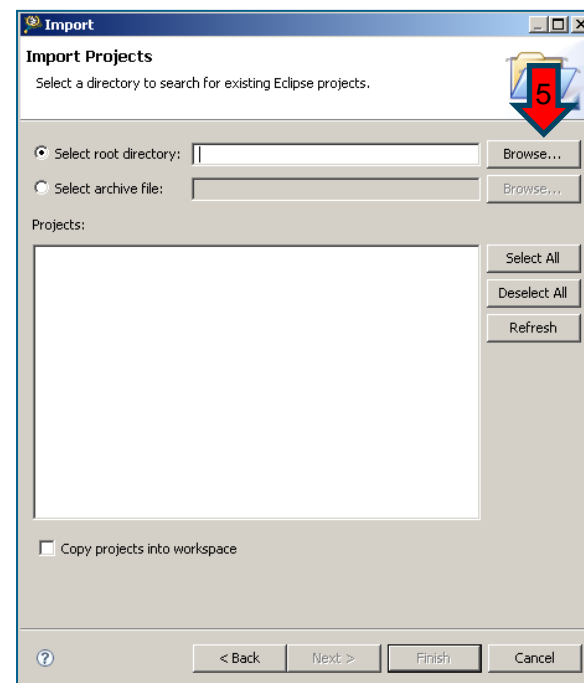
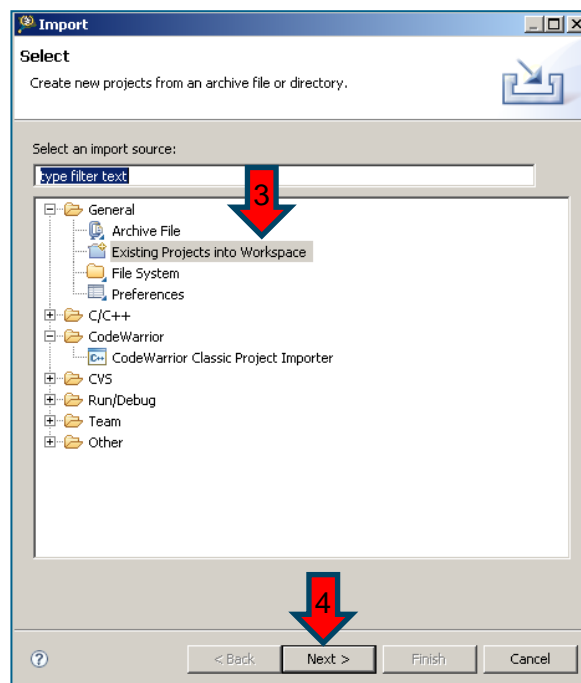
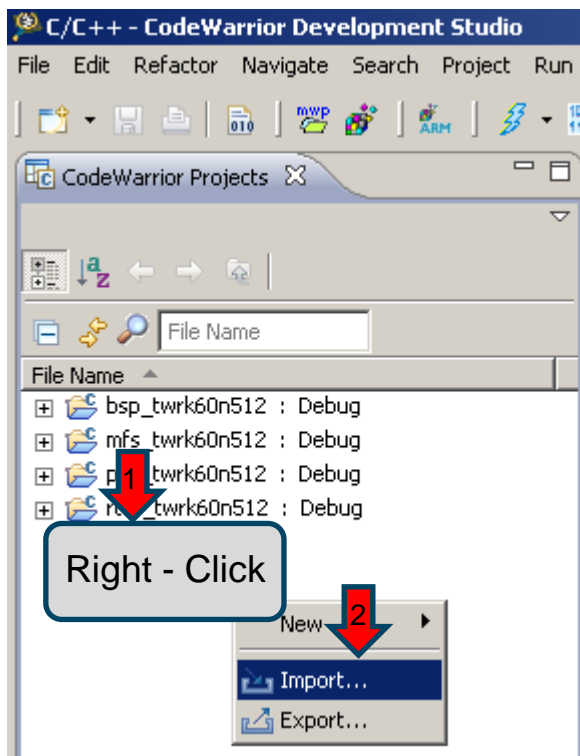


► Select **rtcs_twrk60n512** project

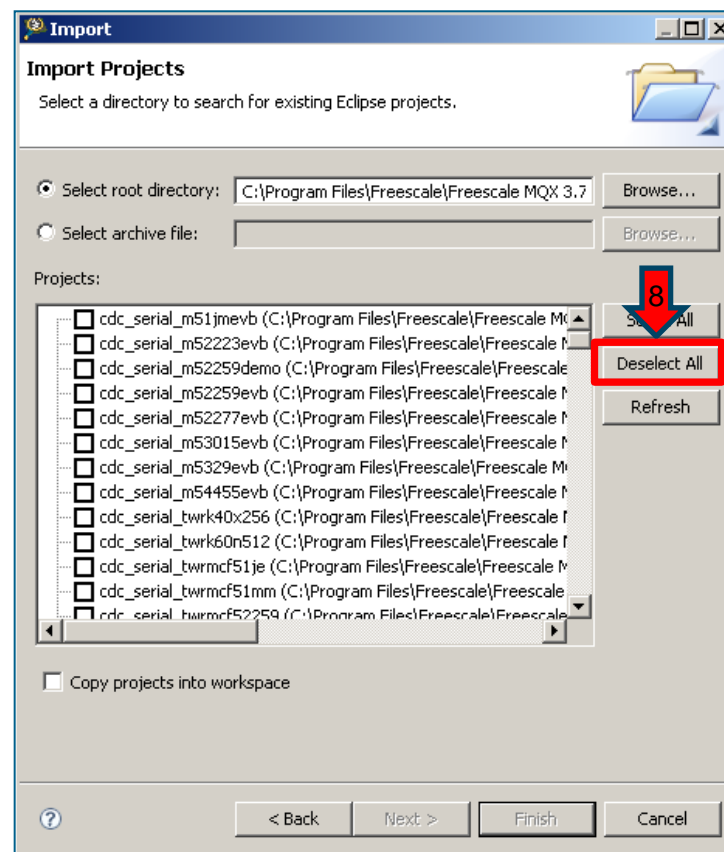
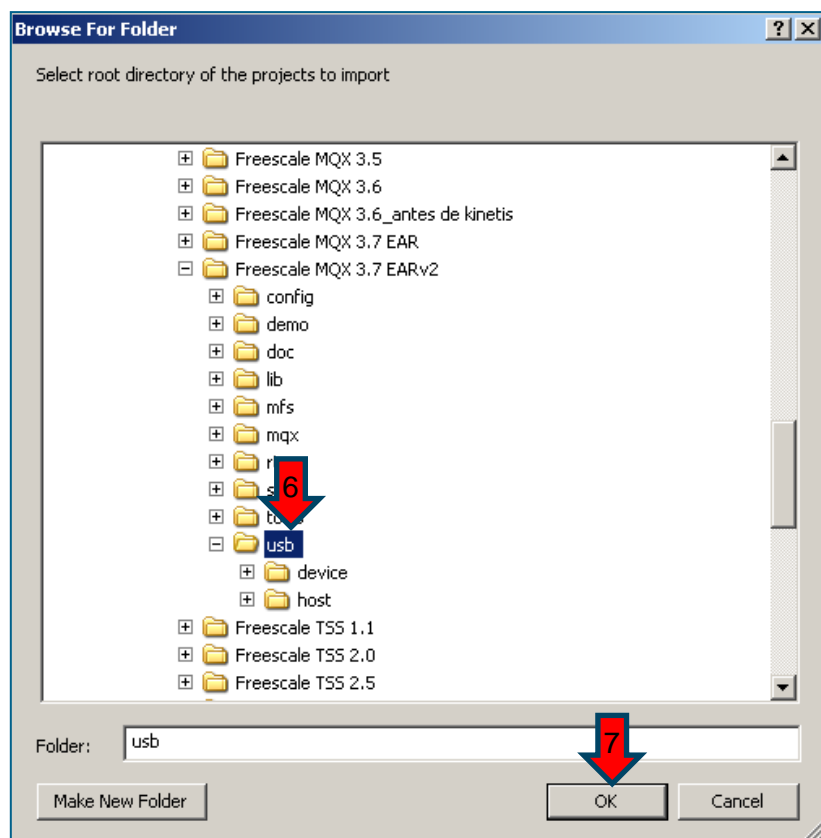


Import MQX Libraries

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

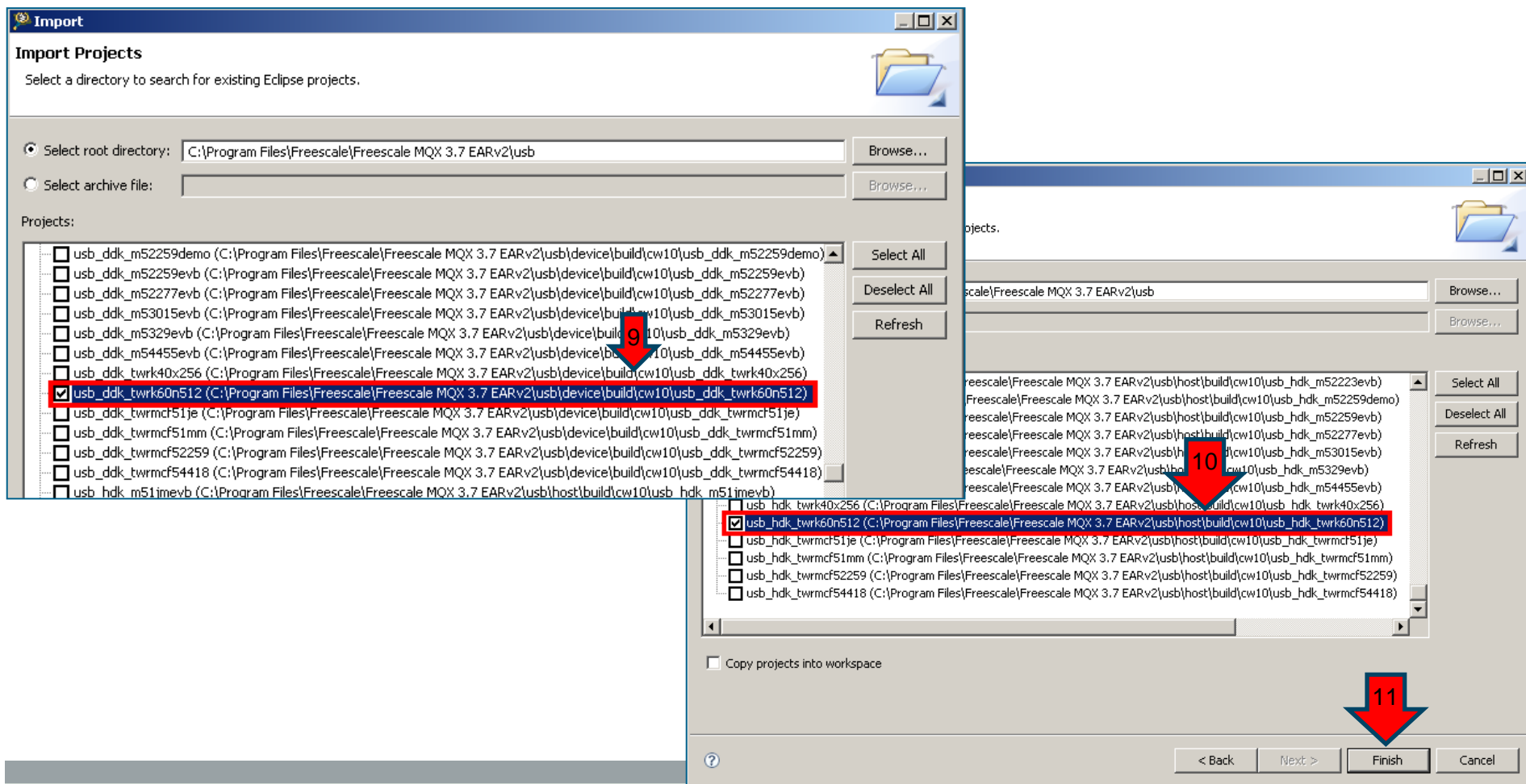


► Select *<install mqx folder>*usb



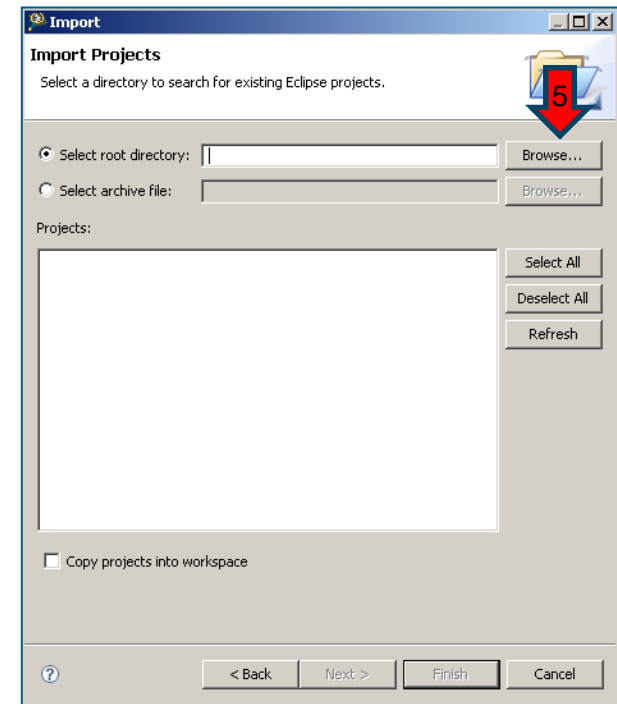
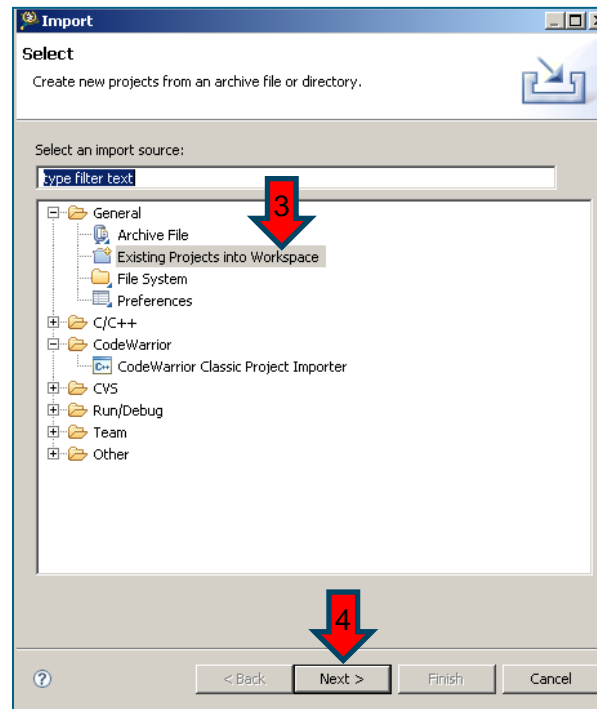
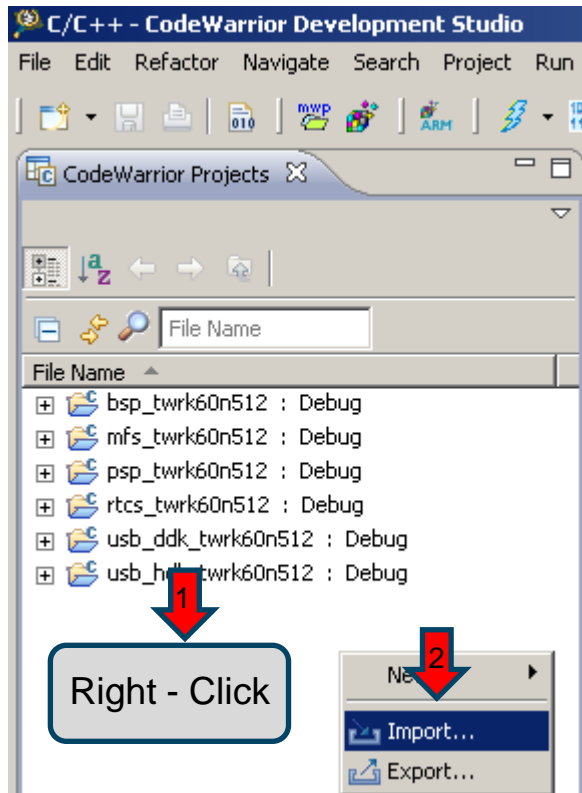
Import MQX Libraries

- Select **usb_ddk_twrk60n512** and **usb_hdk_twrk60n512** projects



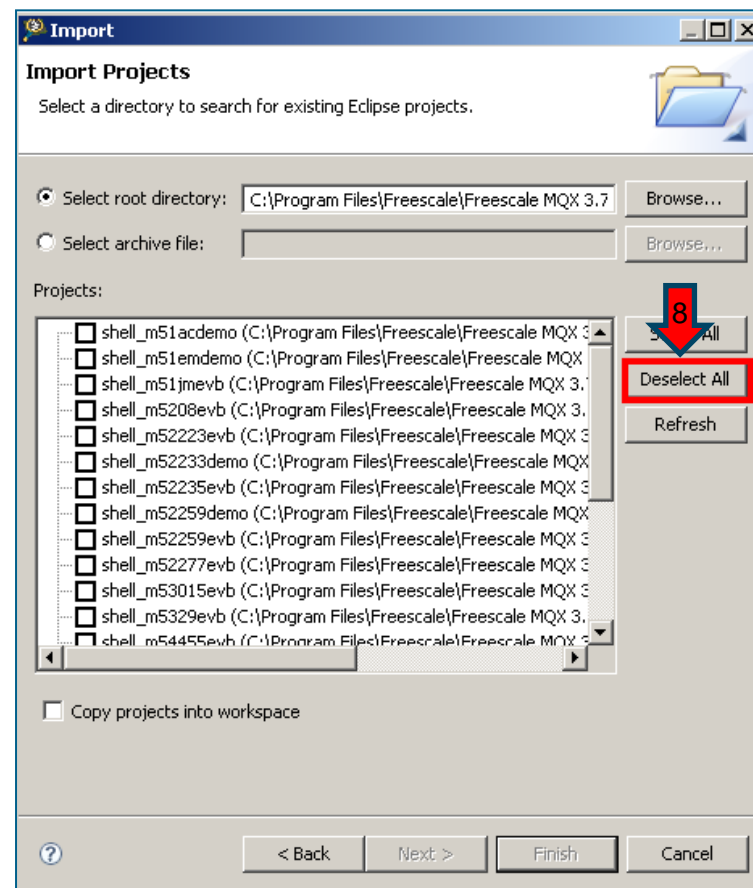
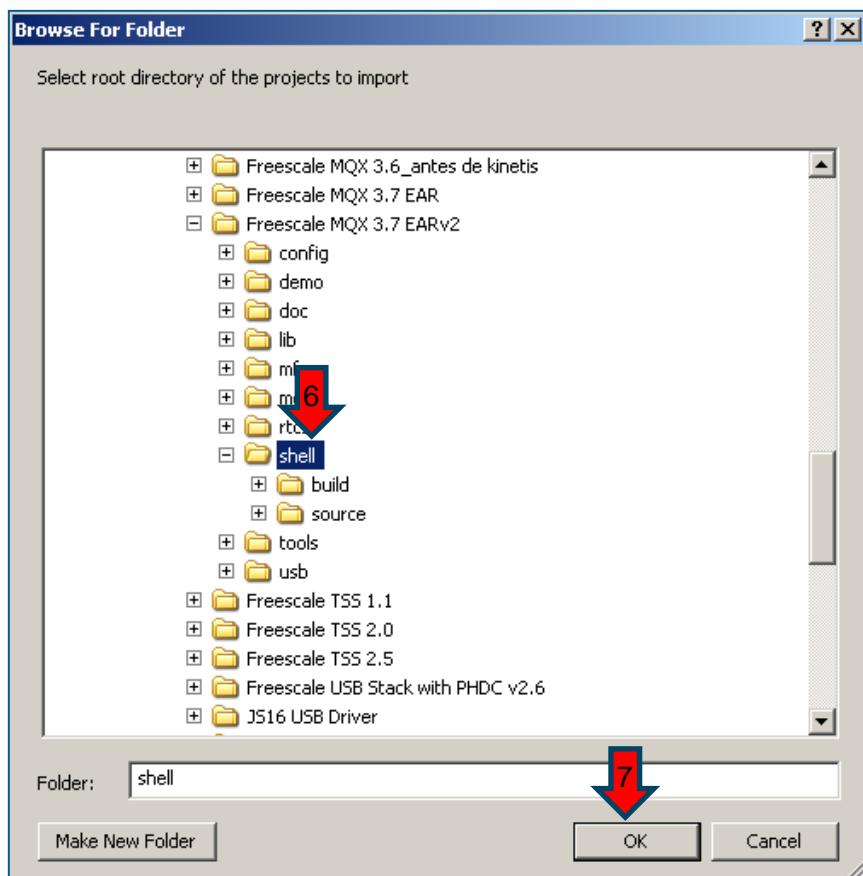
Import MQX Libraries

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

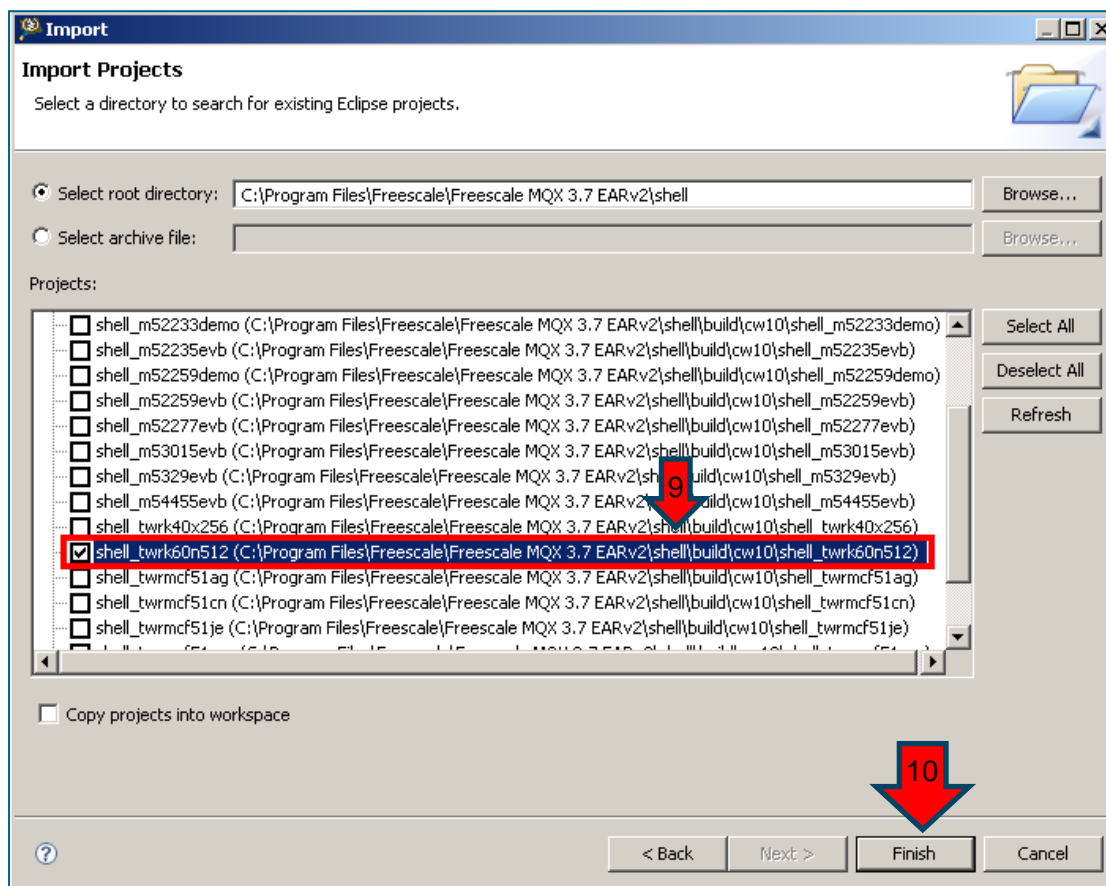


Import MQX Libraries

► Select *<install mqx folder>* \shell

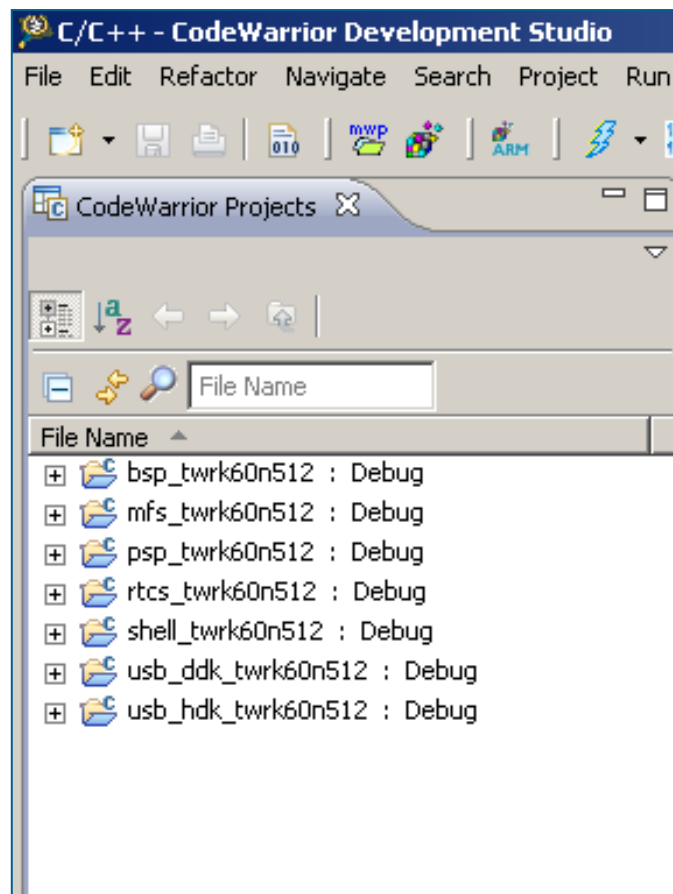


► Select **shell_twrk60n512** project

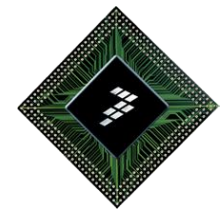


Import MQX Libraries

- You have now all the libraries imported in your workspace

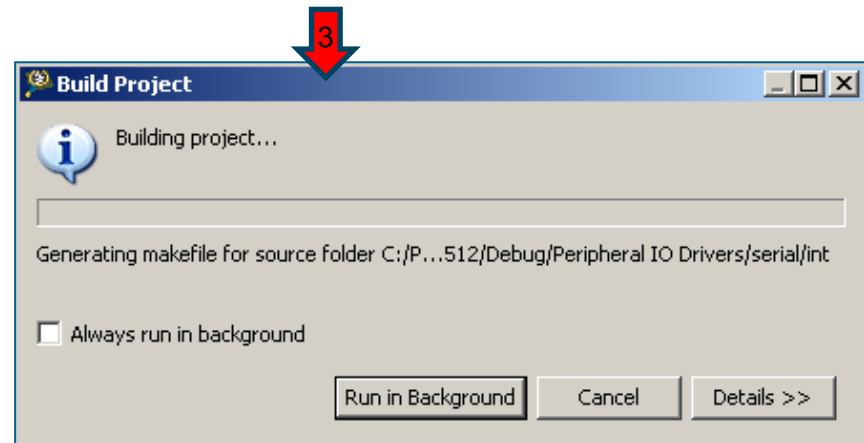
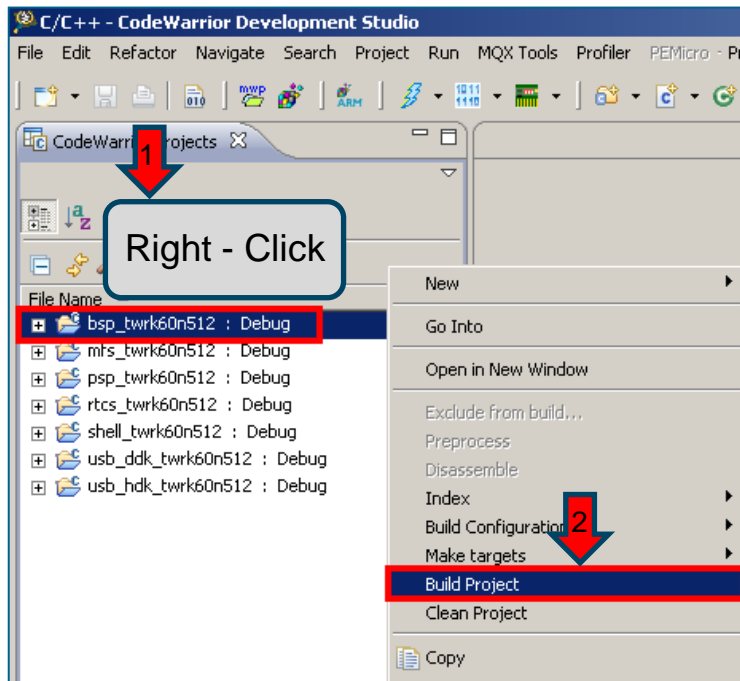


Build MQX libraries



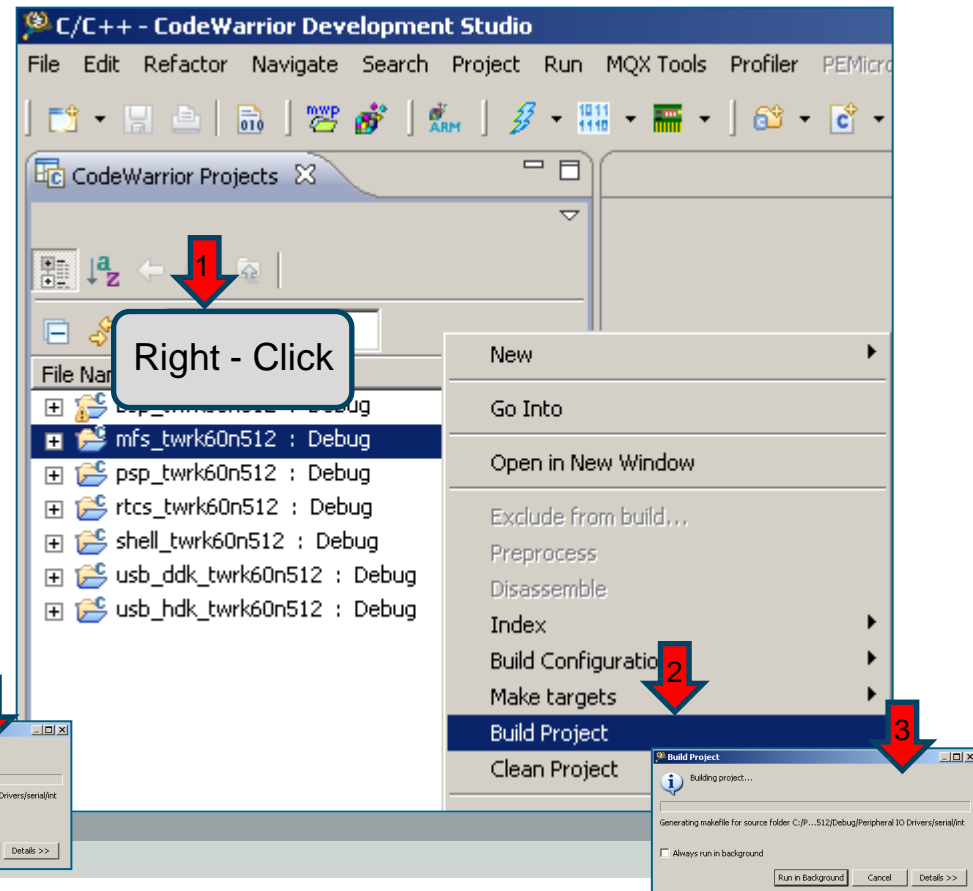
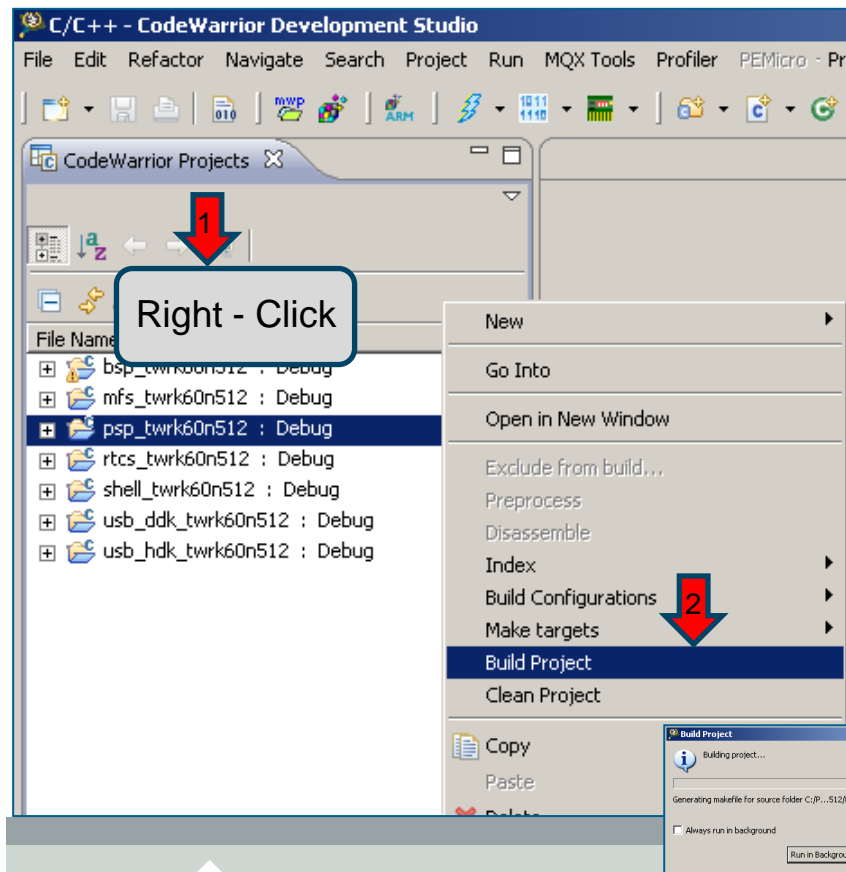
Build MQX libraries

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



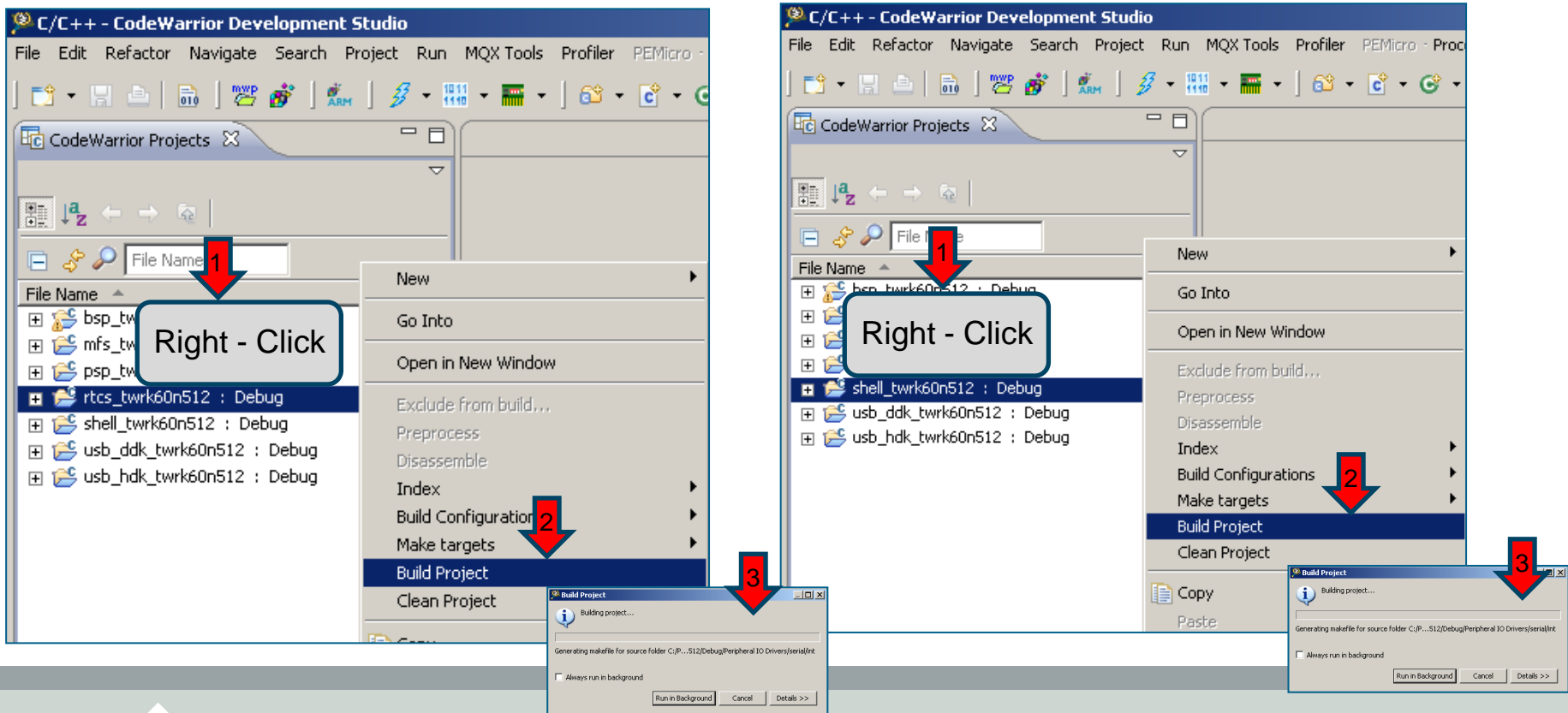
Build MQX libraries

- ▶ Right-Click on Project Explorer **psp_twrk60n512** and Build Project
- ▶ Right-Click on Project Explorer **mfs_twrk60n512** and Build Project



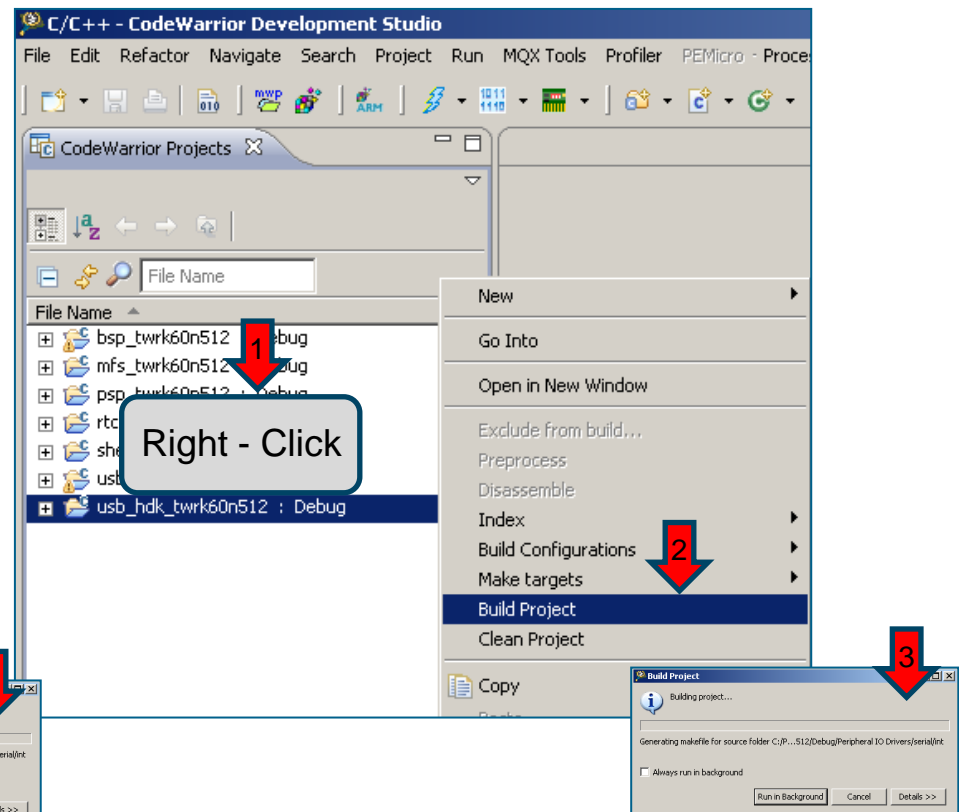
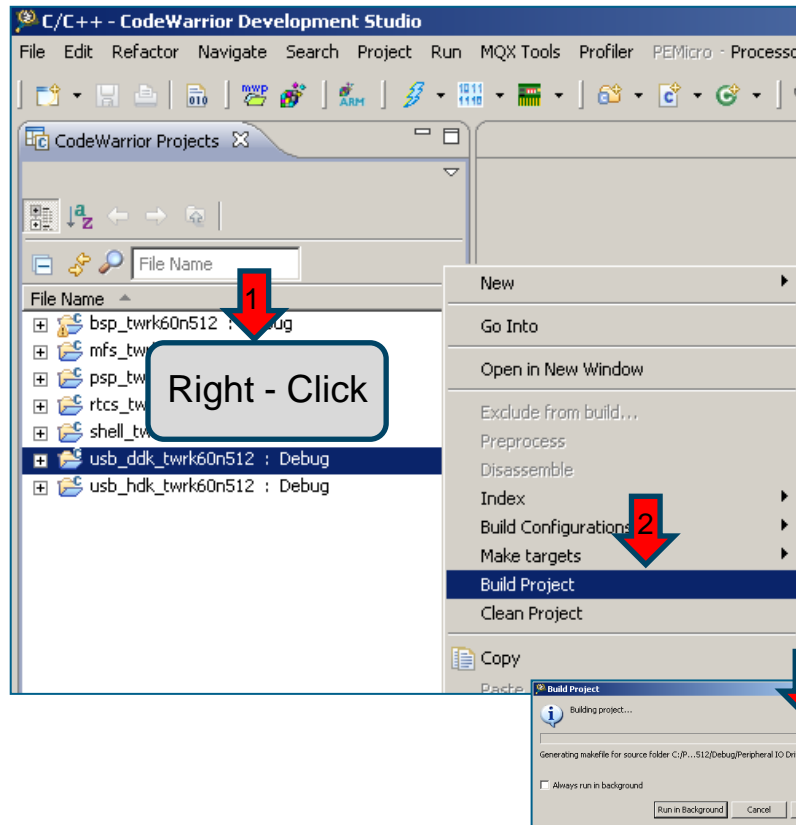
Build MQX libraries

- ▶ Right-Click on Project Explorer **rtcs_twrk60n512** and Build Project
- ▶ Right-Click on Project Explorer **shell_twrk60n512** and Build Project

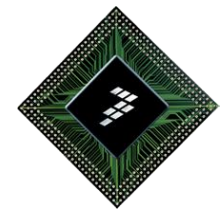


Build MQX libraries

- ▶ Right-Click on Project Explorer **usb_ddk_twrk60n512** and Build Project
- ▶ Right-Click on Project Explorer **usb_hdk_twrk60n512** and Build Project

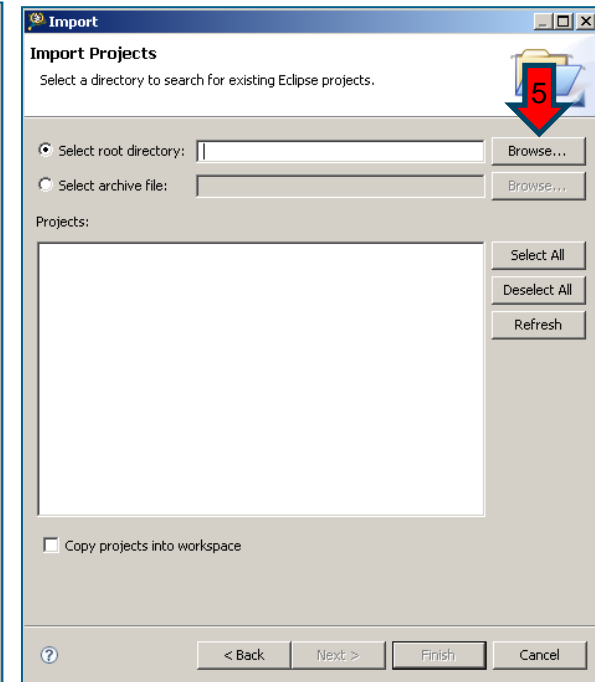
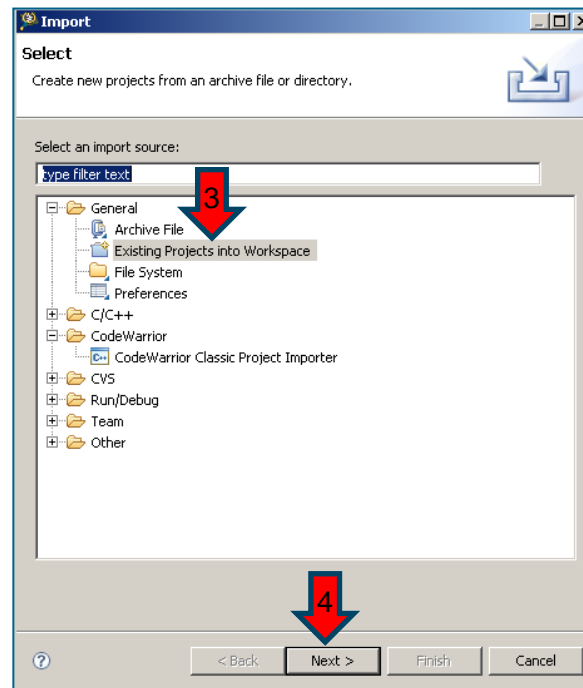
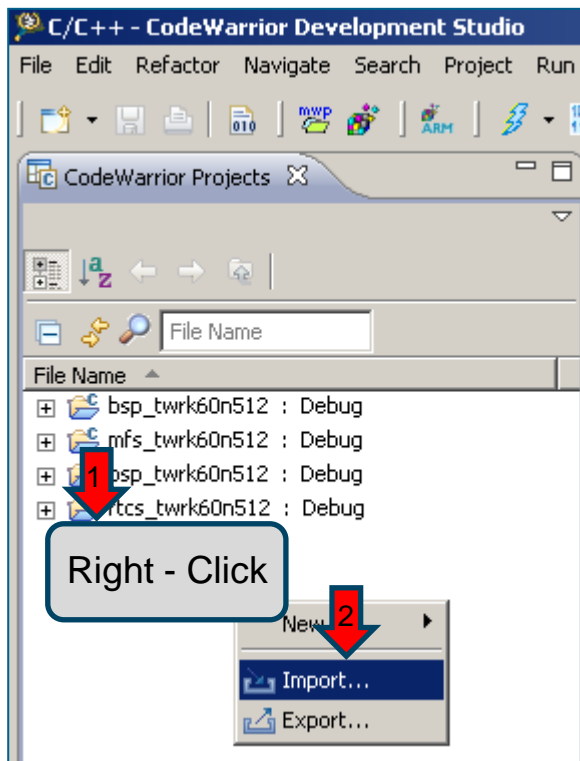


Import and Debug MQX Hello World Project



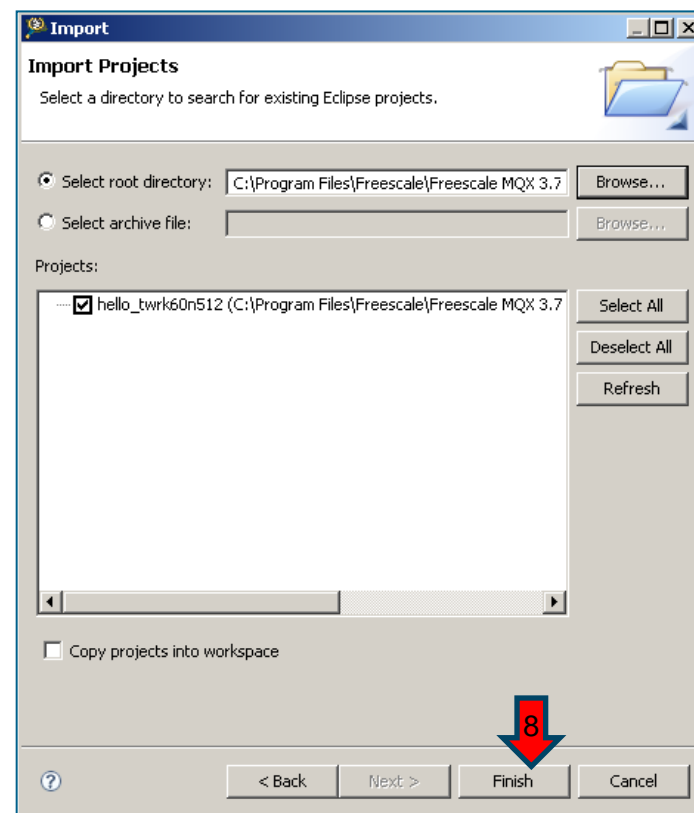
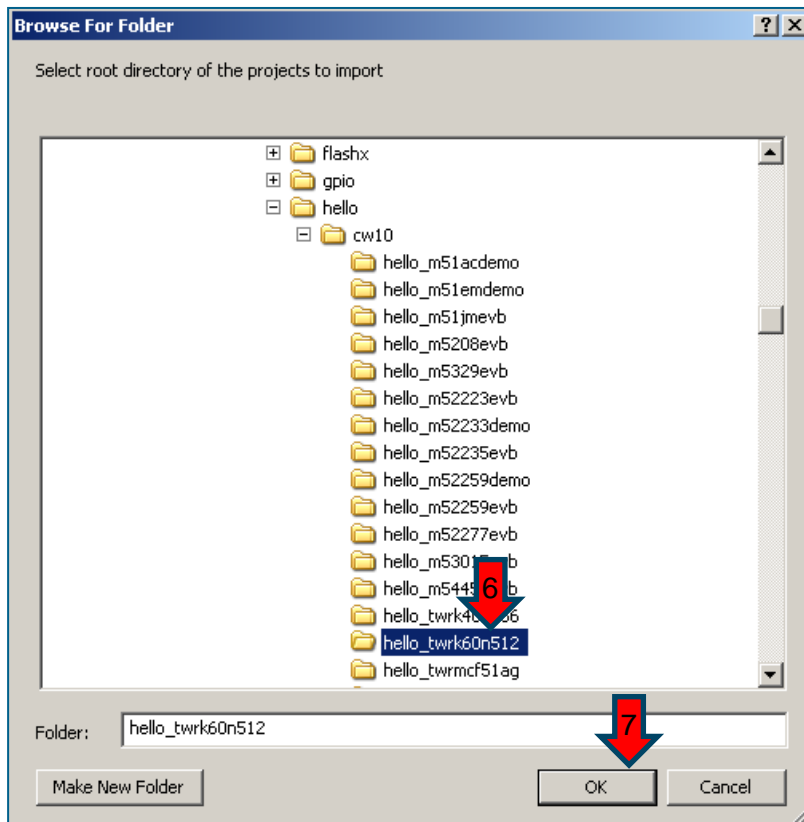
Import 'Hello World' MQX example

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



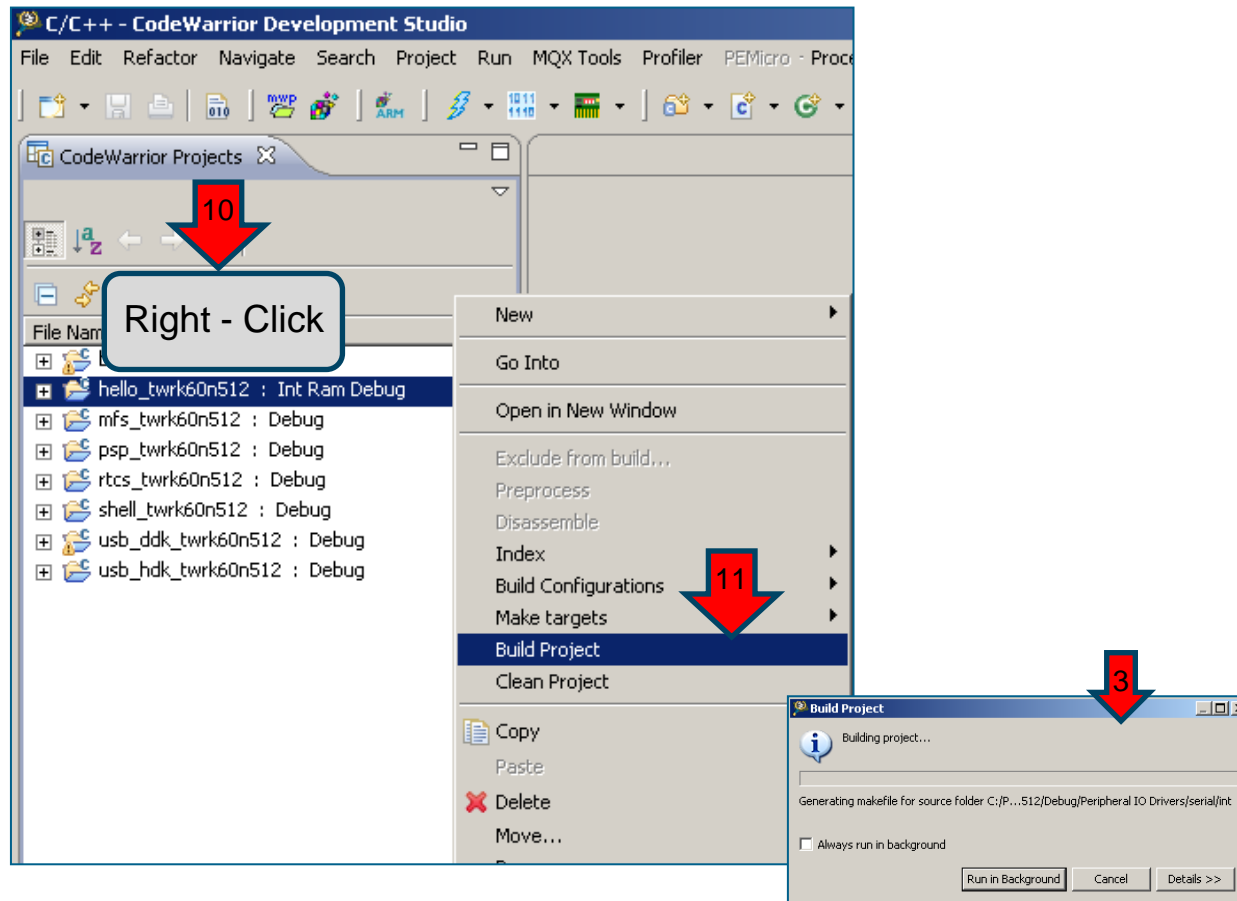
Import 'Hello World' MQX example

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



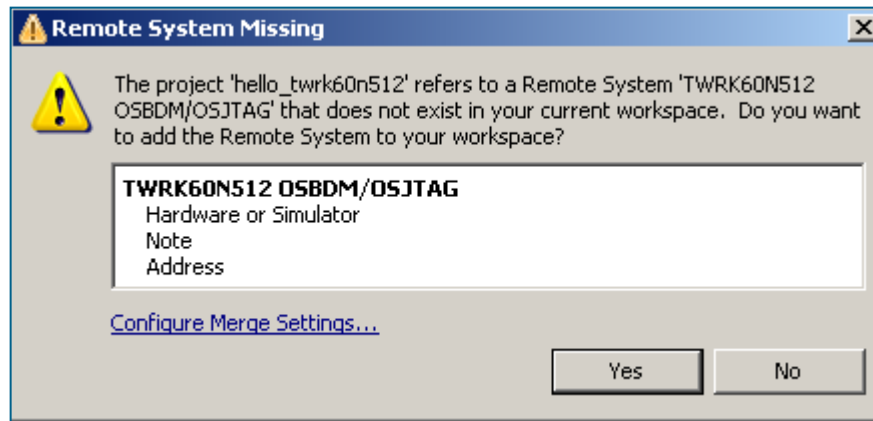
Build 'Hello World' MQX example

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



Build 'Hello World' MQX example

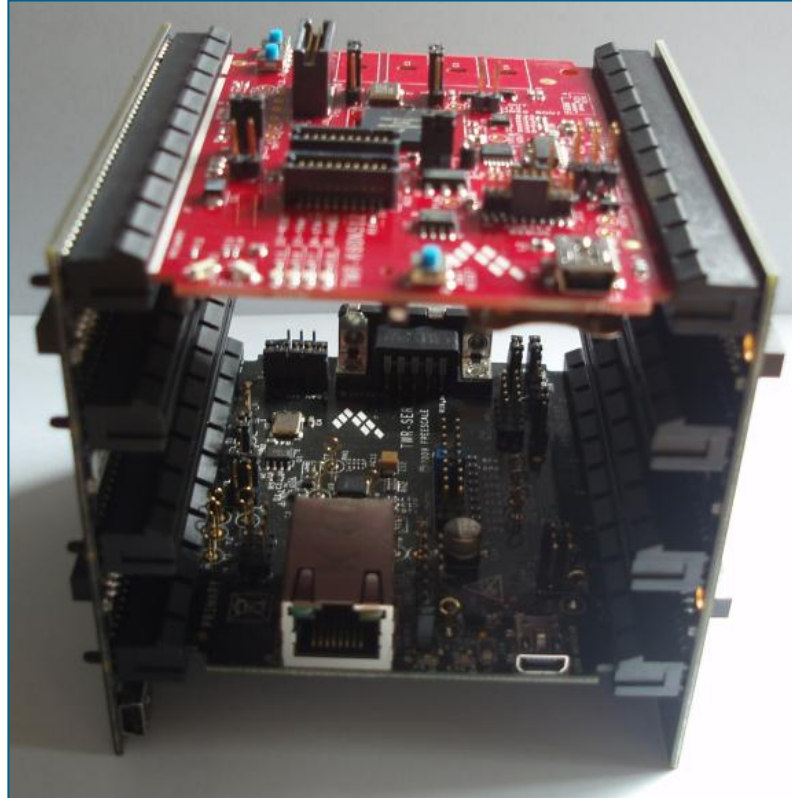
- ▶ If you receive this message, your connection is not configured properly. We will do it manually later



Prepare your hardware

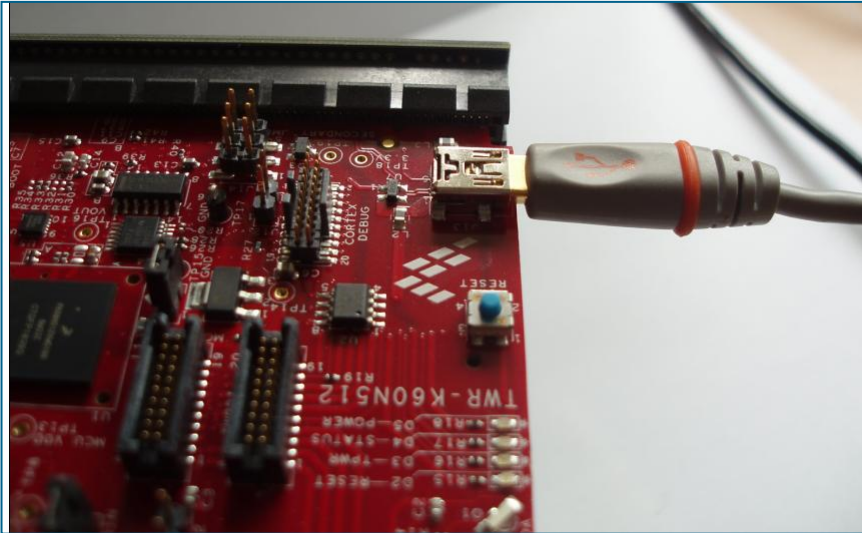
► Prepare your Tower System:

- Connect **TWR-SER** and **TWR-K60N512** to **TWR-ELEV** (Primary and Secondary)



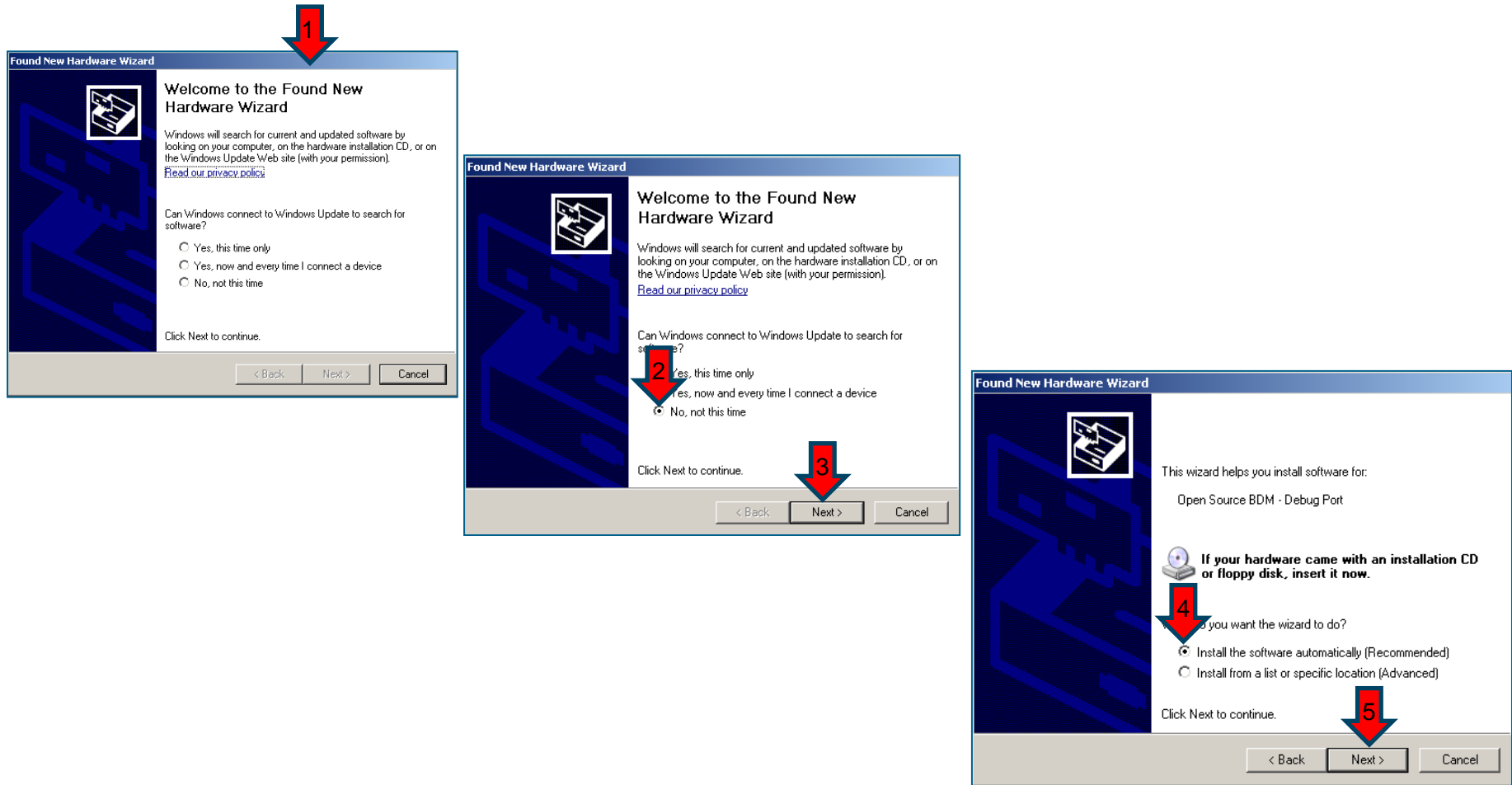
Prepare your hardware

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

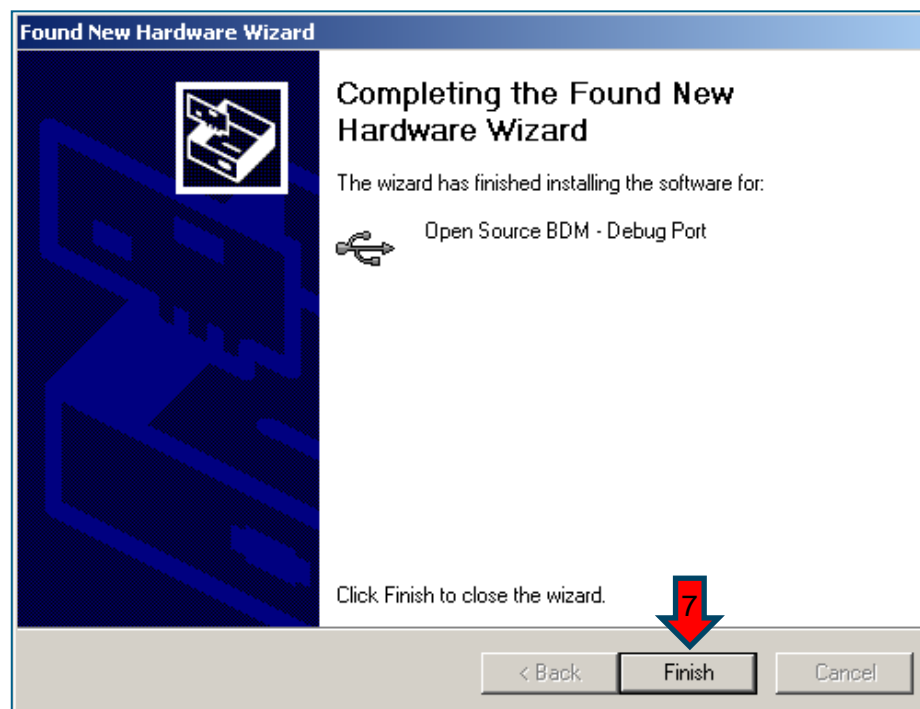
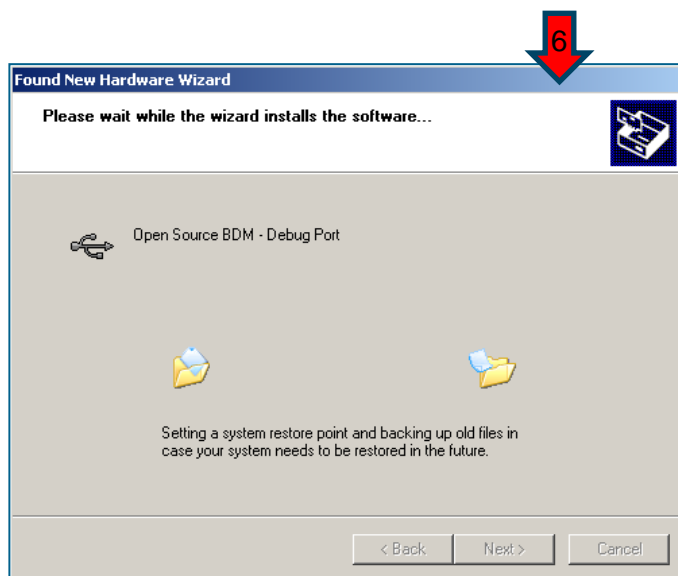


Install OSJTAG Drivers

- ▶ Windows will detect the new USB device. Do next steps

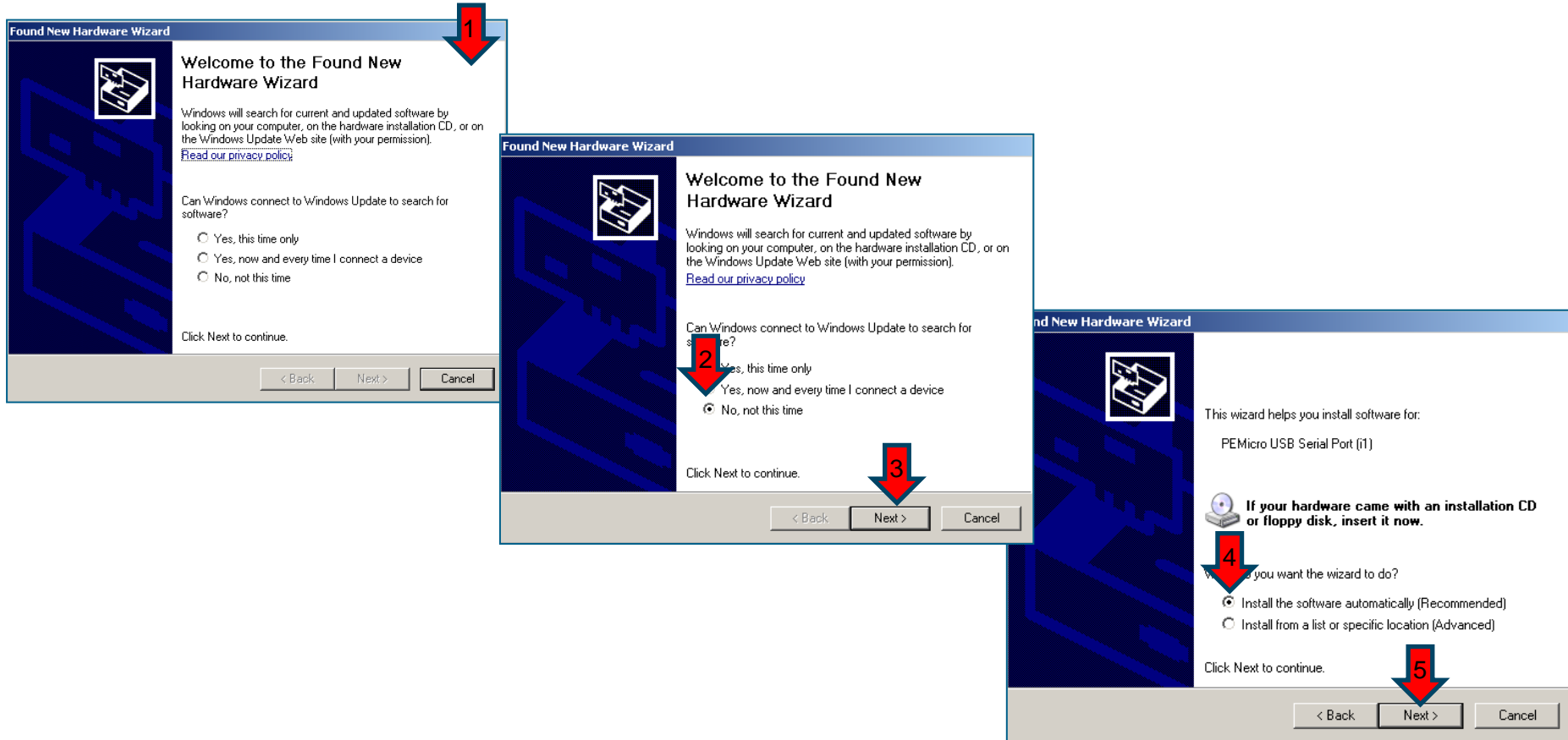


► Finish driver installation

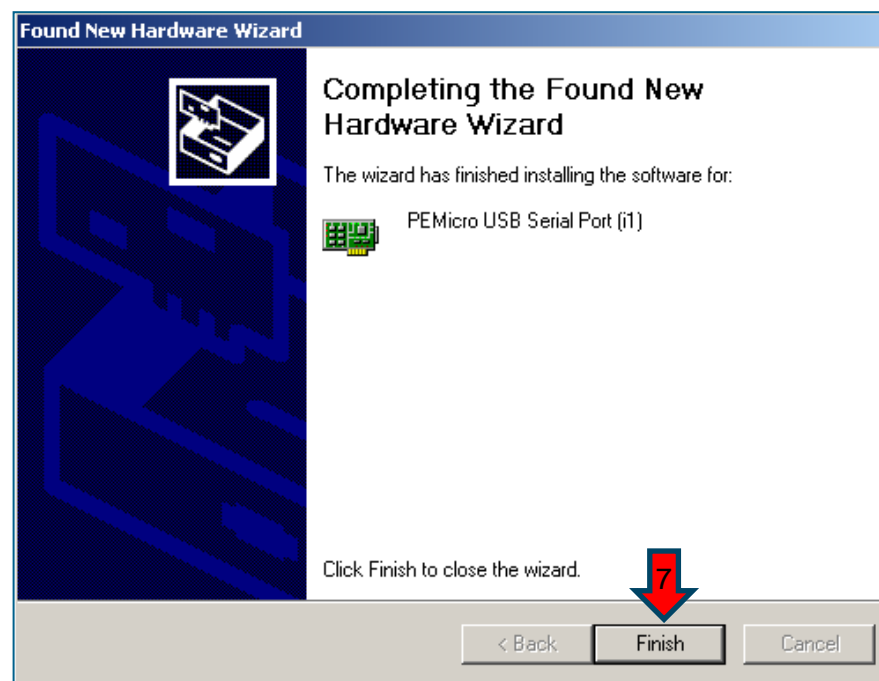
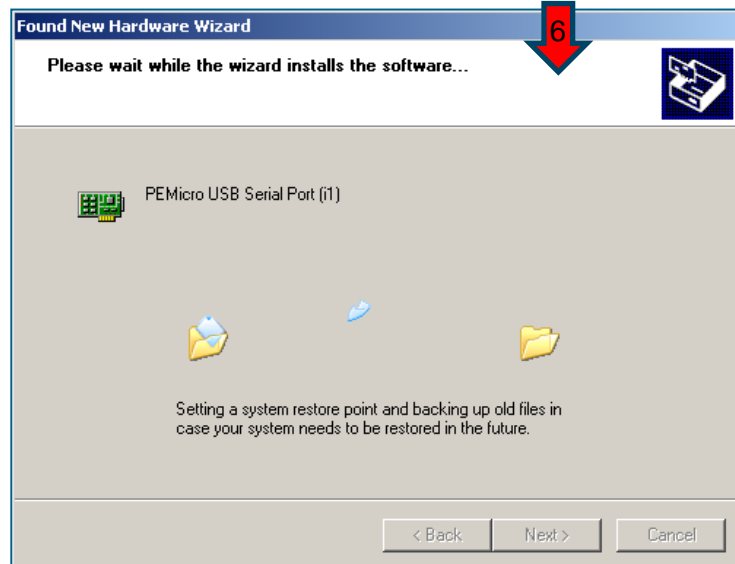


Install OSJTAG Drivers

- ▶ OSJTAG includes two profiles, OSBDM and USB Serial Port.
Install driver for USB Serial Port

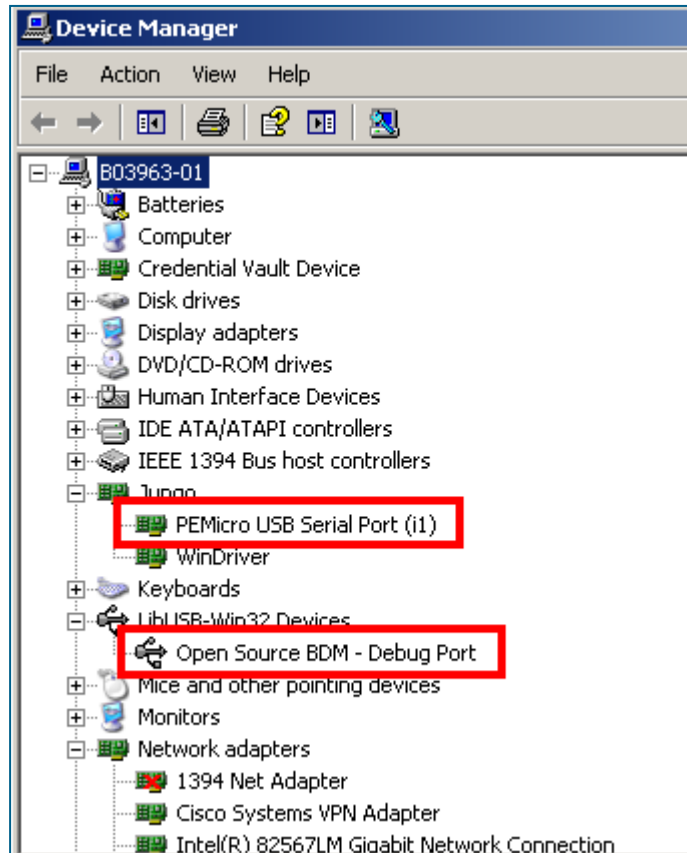


► Finish driver installation



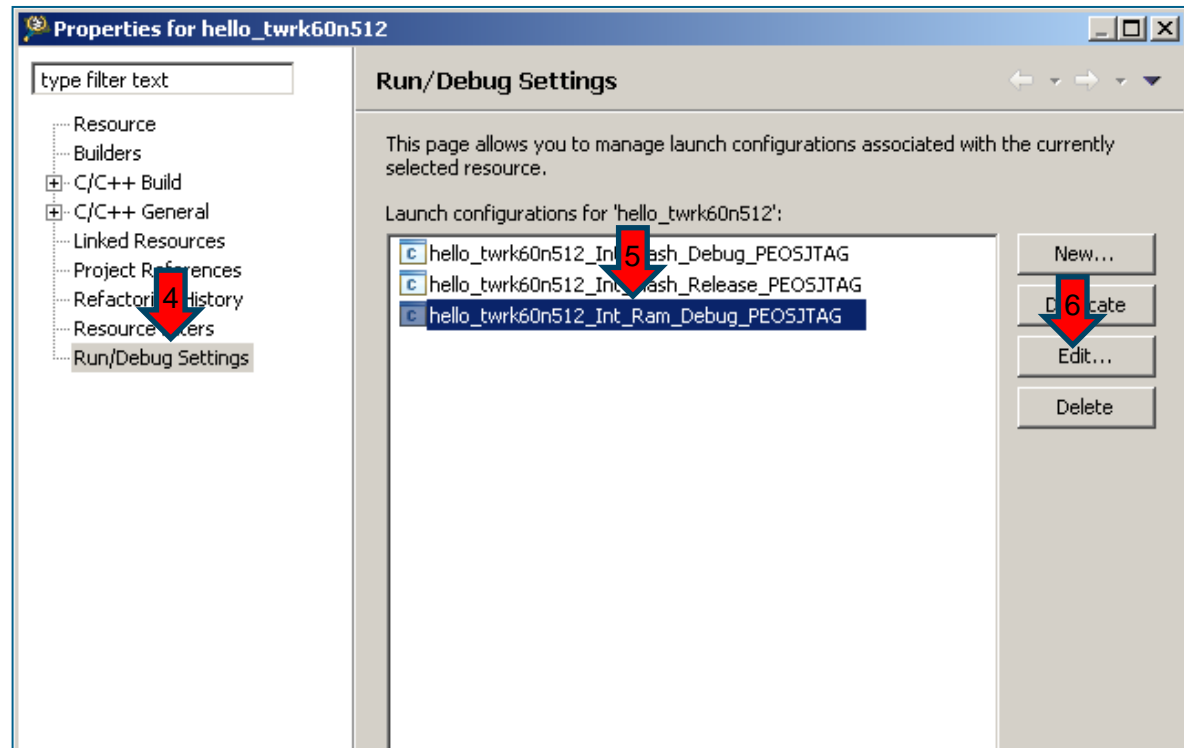
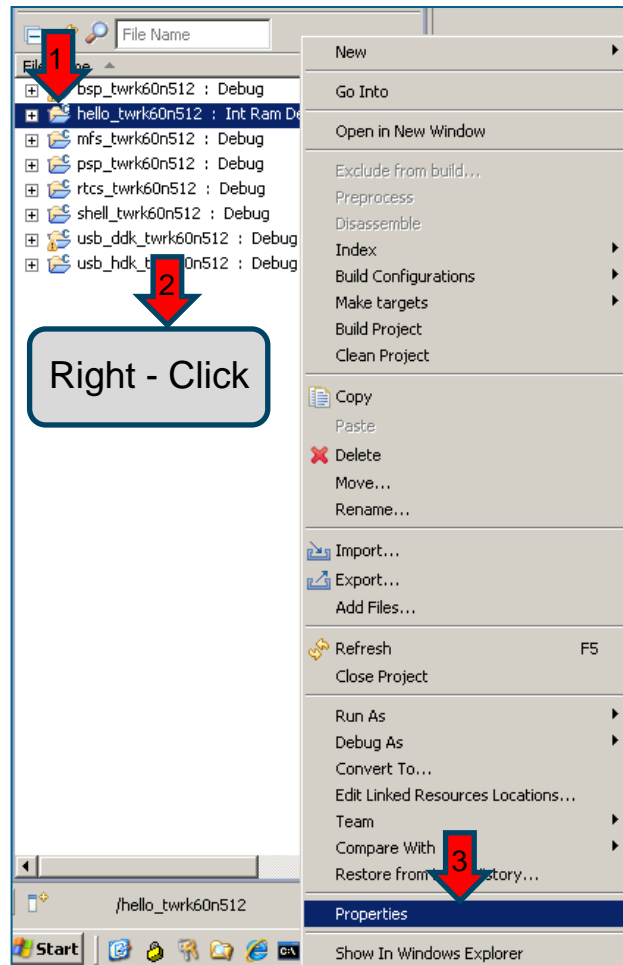
Install OSJTAG Drivers

- You can see in Device Manager the two new USB devices installed



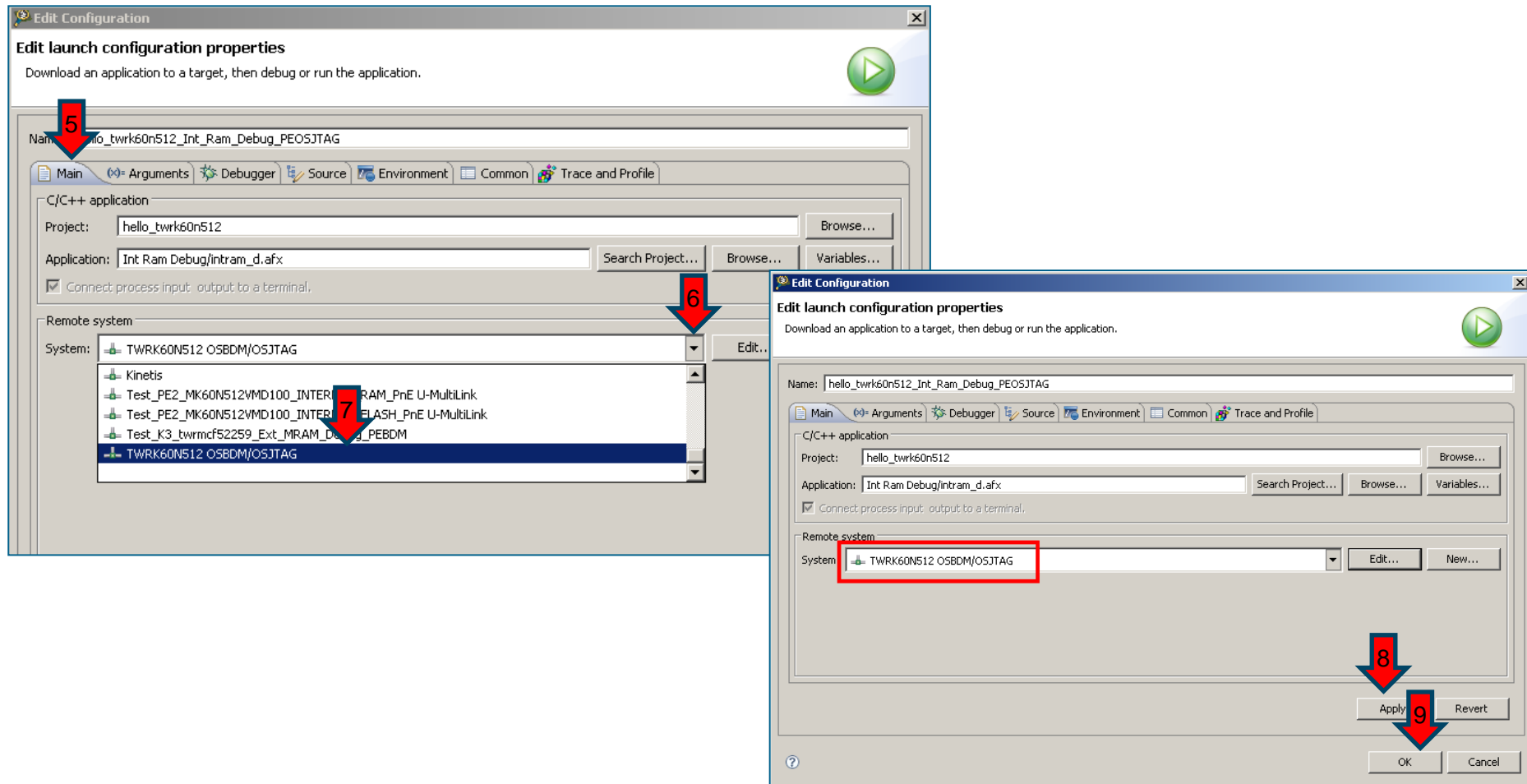
Change Connection Settings

► Configure Connection Settings of the project



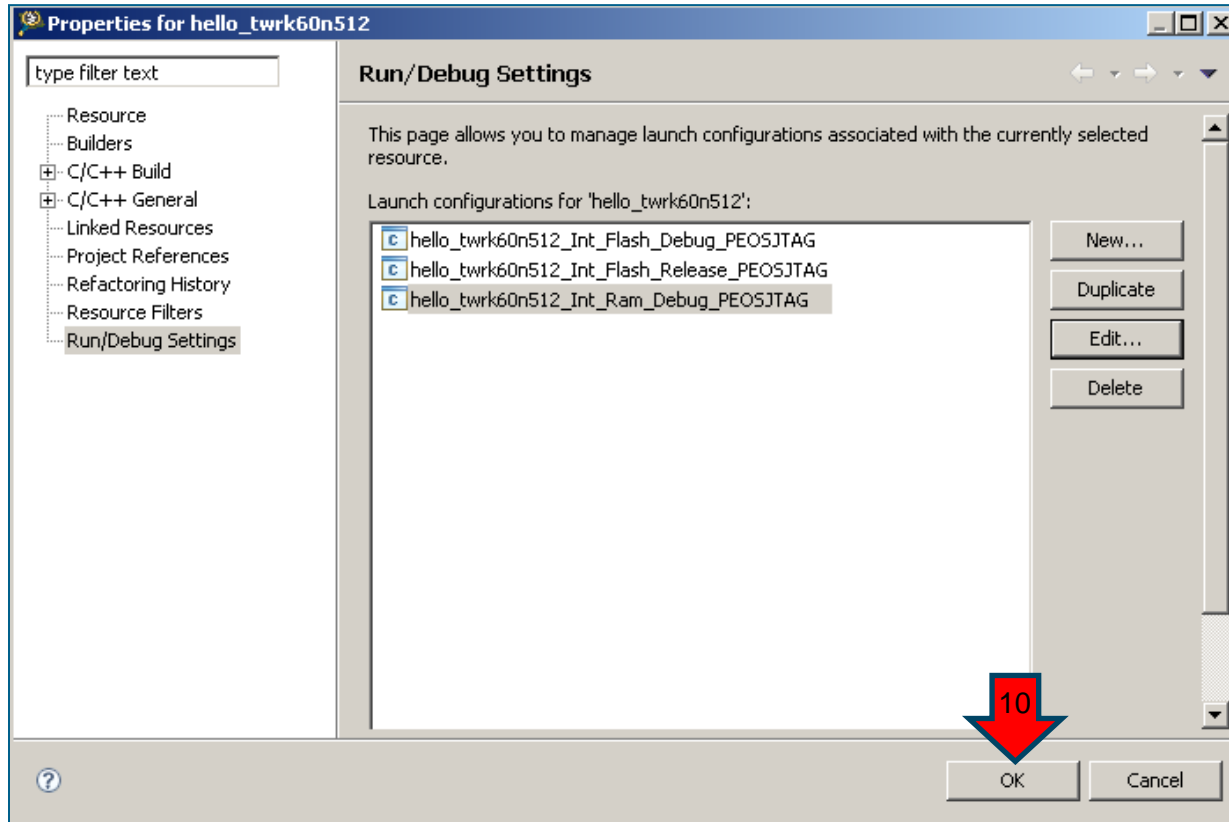
Change Connection Settings

► Select **TWRK60N512 OSBDM/OSJTAG** Remote System



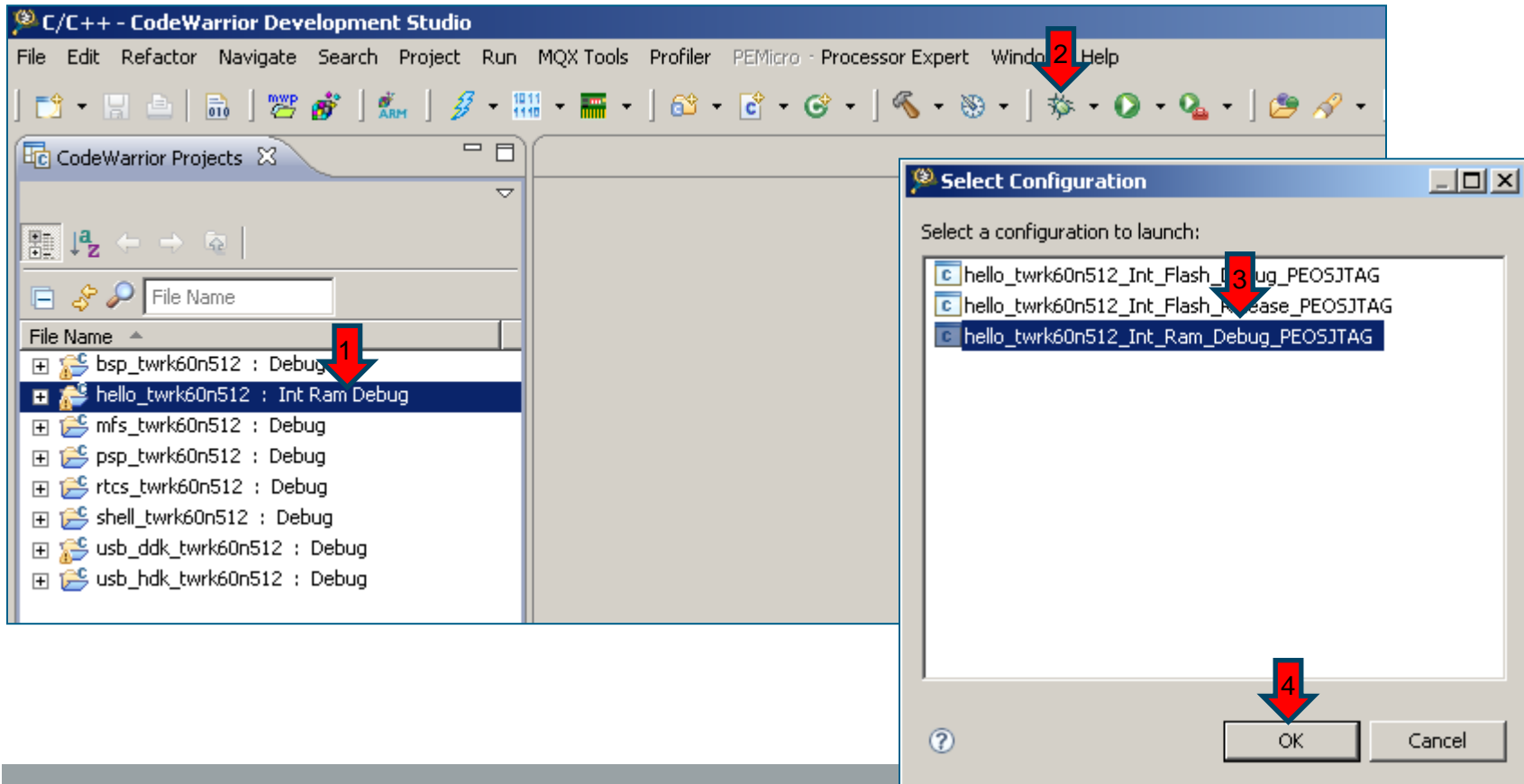
Change Connection Settings

► Click OK



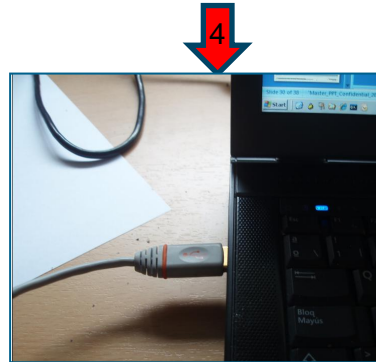
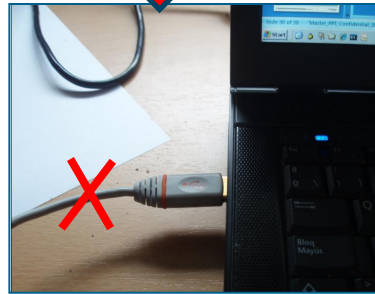
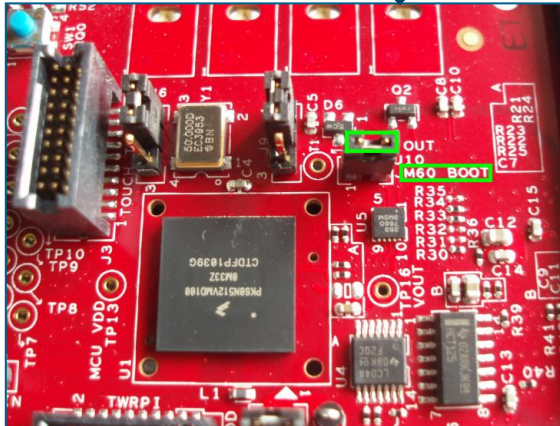
Debug MQX 'Hello World' example

- ▶ Select **hello_twrk60n512** project and Click 'Debug icon'
- ▶ Select **hello_twrk60n512_Int_Ram_Debug_PEOSJTAG** Connection



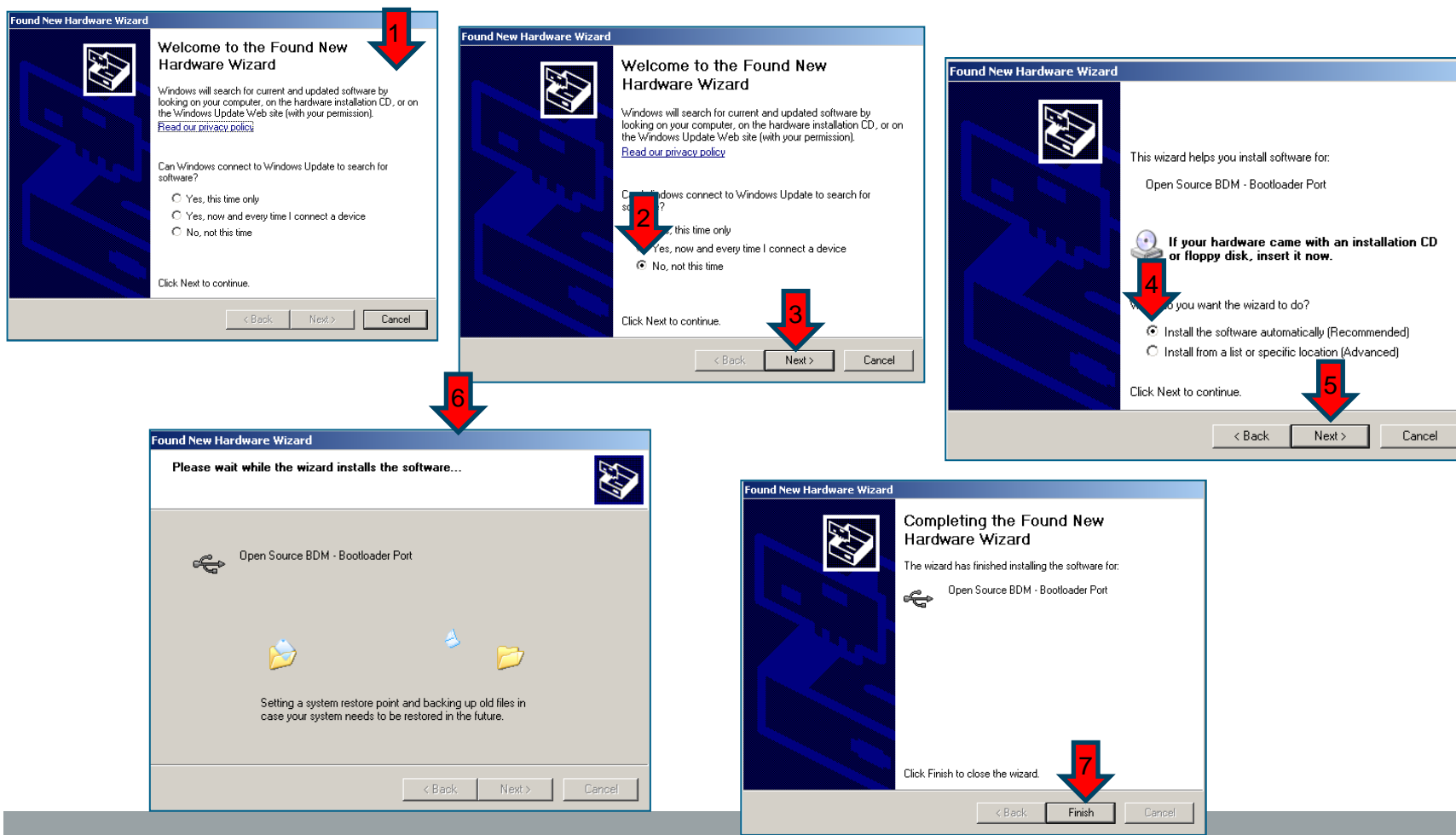
Update OSJTAG firmware

- ▶ An update of OSJTAG firmware could be needed.
- ▶ Unplug USB cable
- ▶ Insert Jumper
- ▶ Reconnect USB



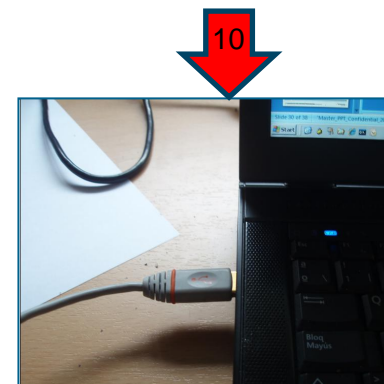
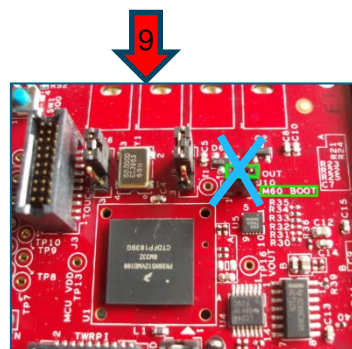
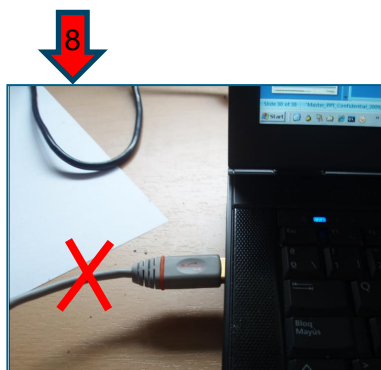
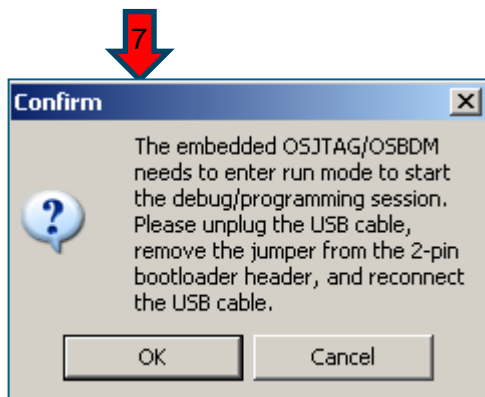
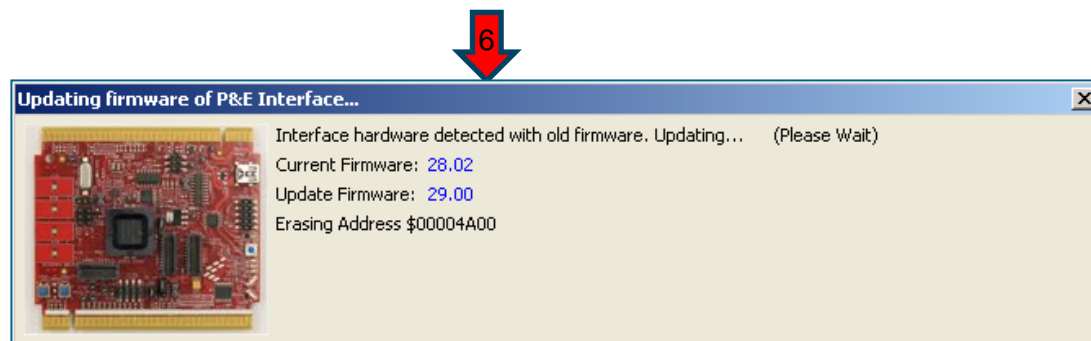
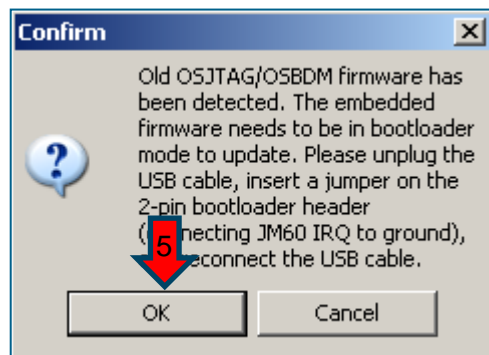
Update OSJTAG firmware

► Install Bootloader USB driver



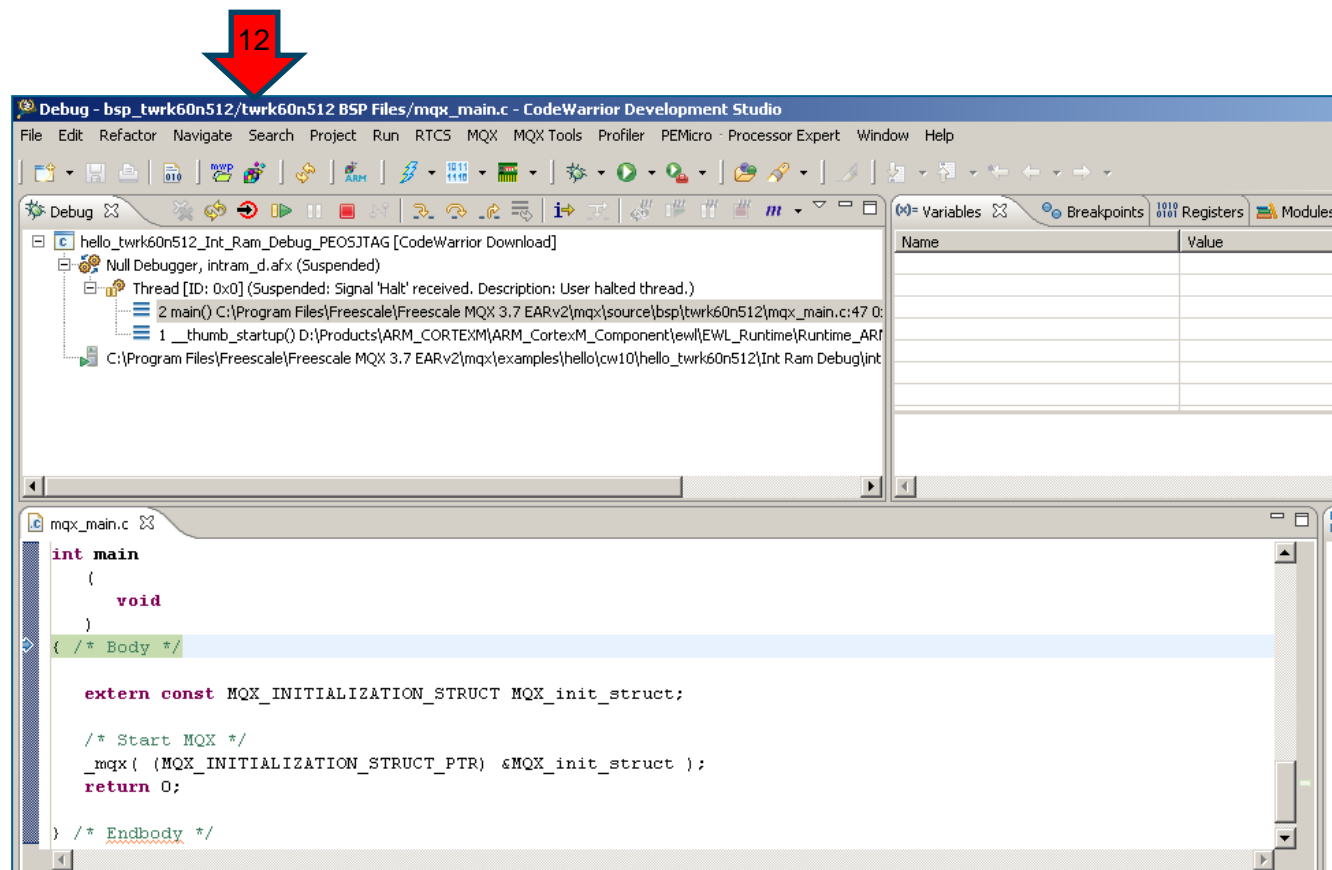
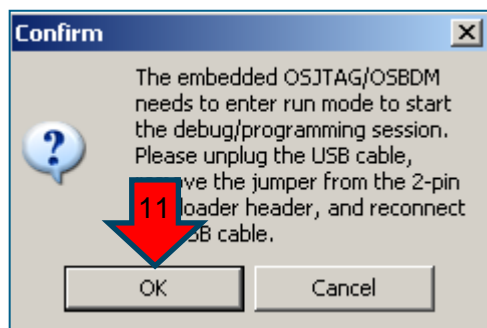
Update OSJTAG firmware

- Click OK and finish firmware update



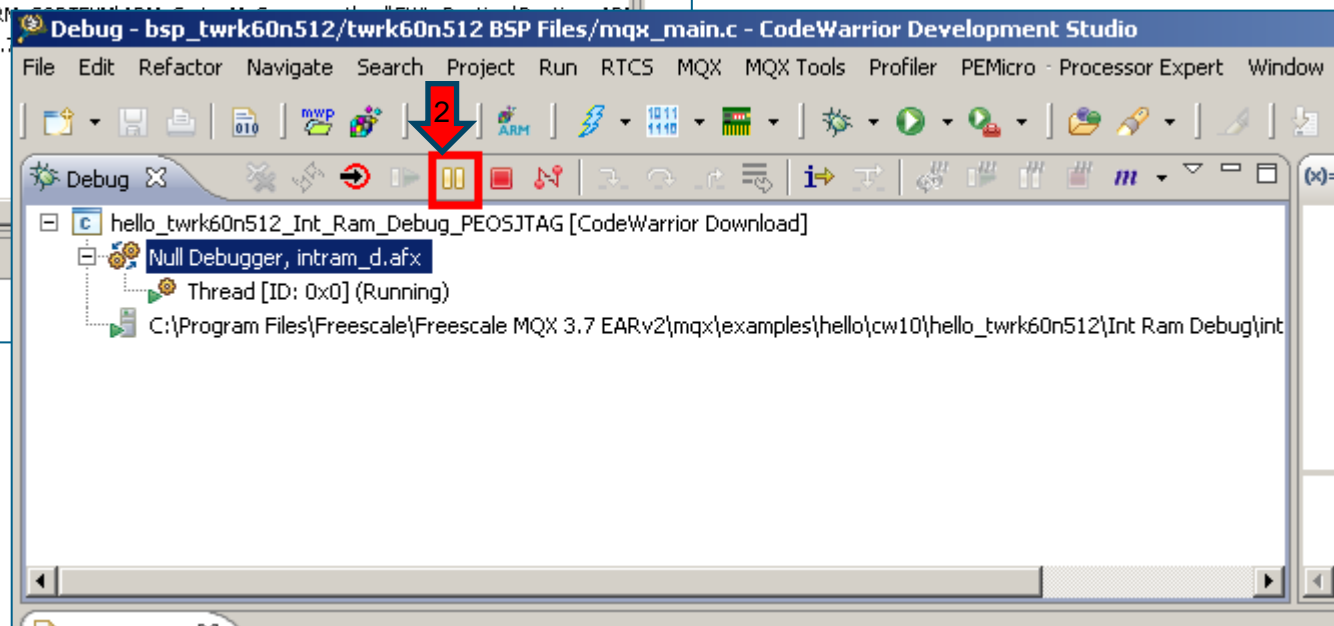
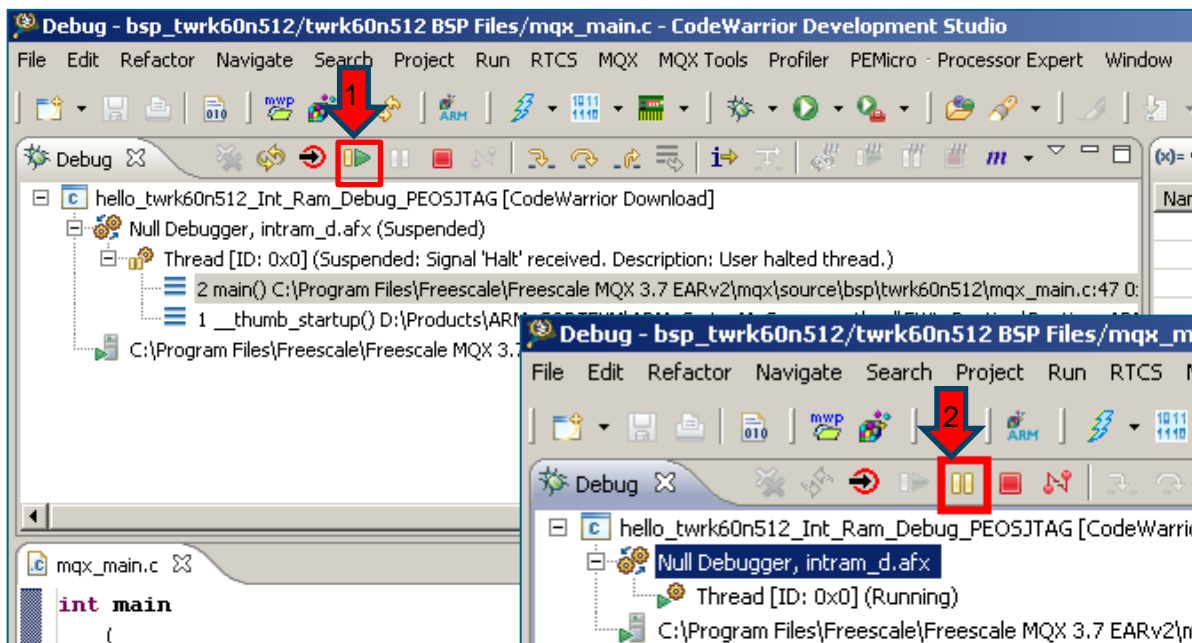
Update OSJTAG firmware

- You are ready to Run and Debug the project



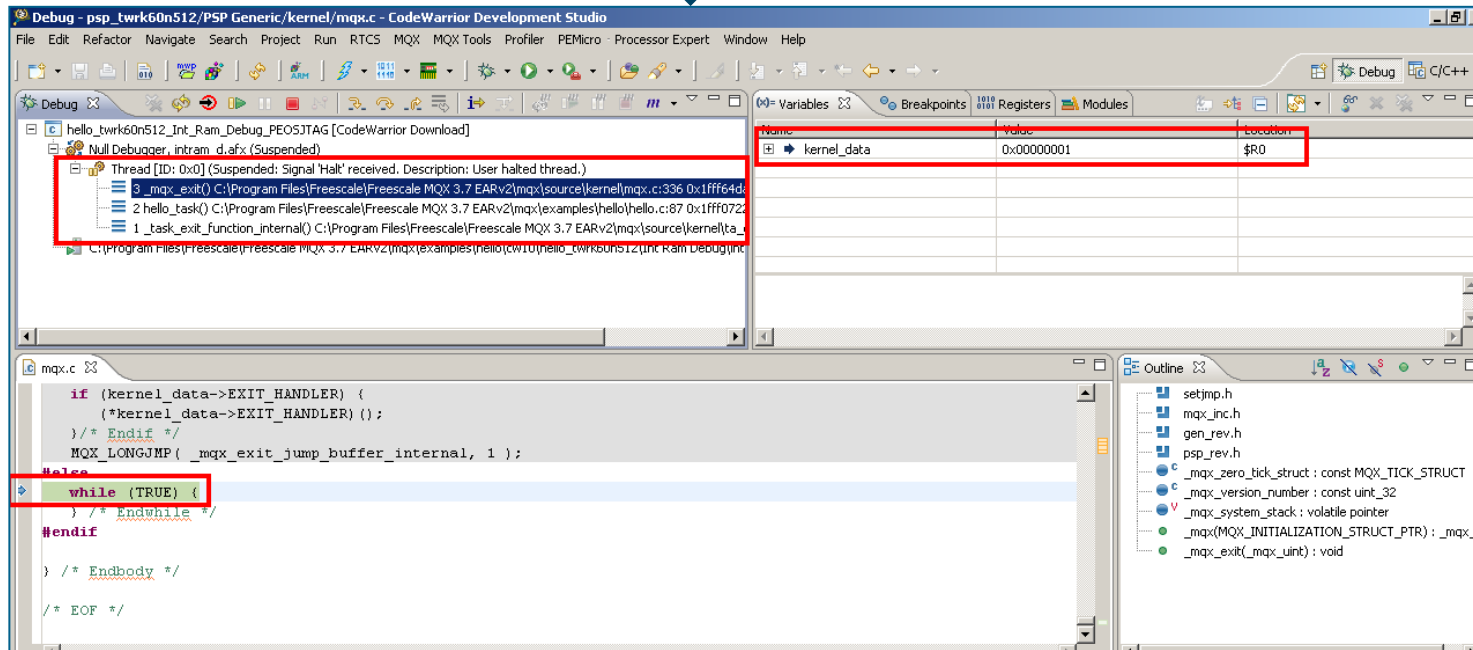
Run MQX 'Hello World' example

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



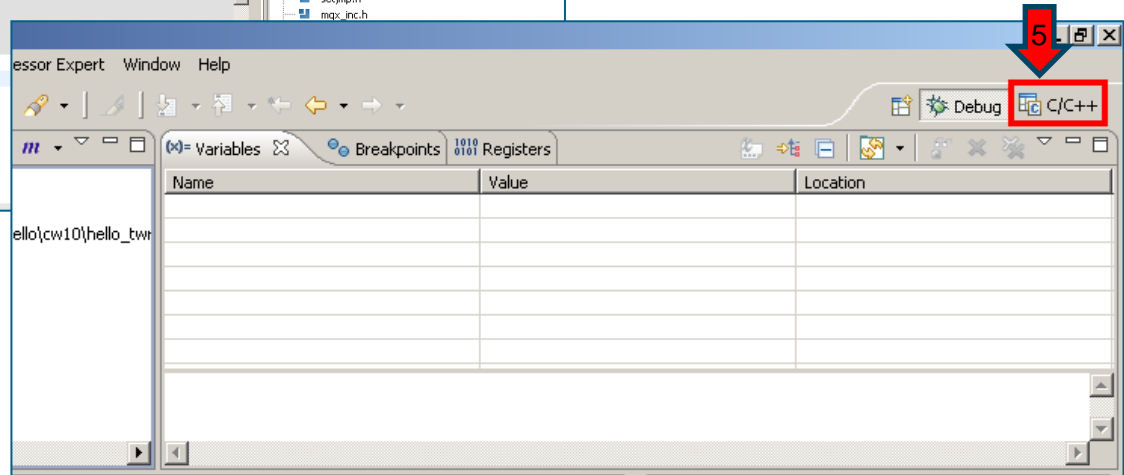
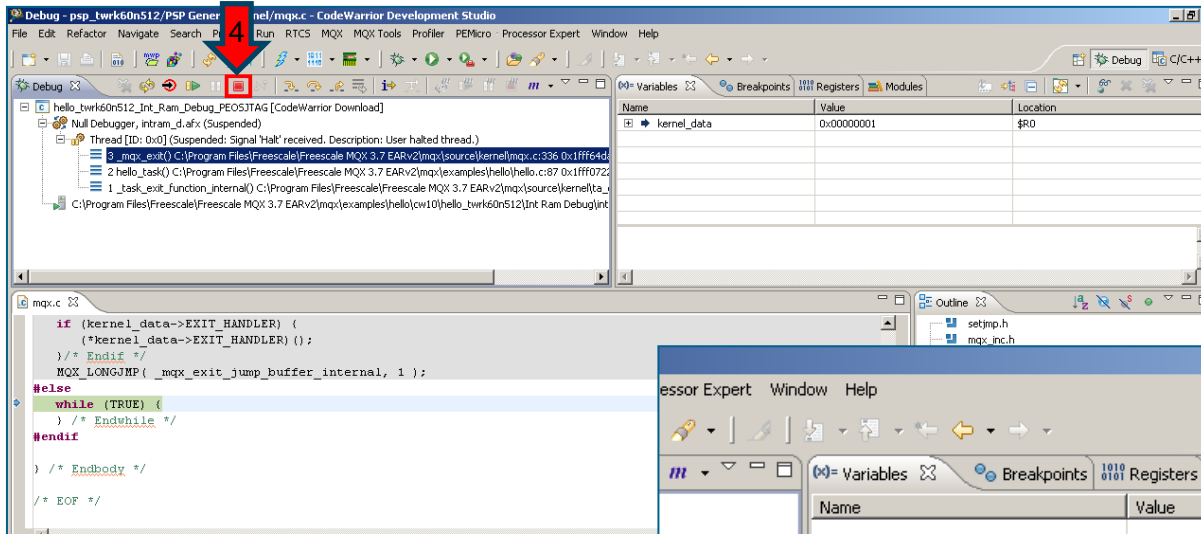
Run MQX 'Hello World' example

- You can explore the Debugging Eclipse perspective

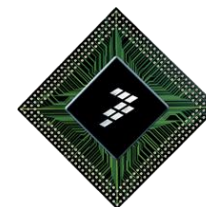


Run MQX 'Hello World' example

- ▶ Terminate the Debugging session and change Eclipse perspective
- ▶ You have Run and Debug your first MQX CW10.1 project

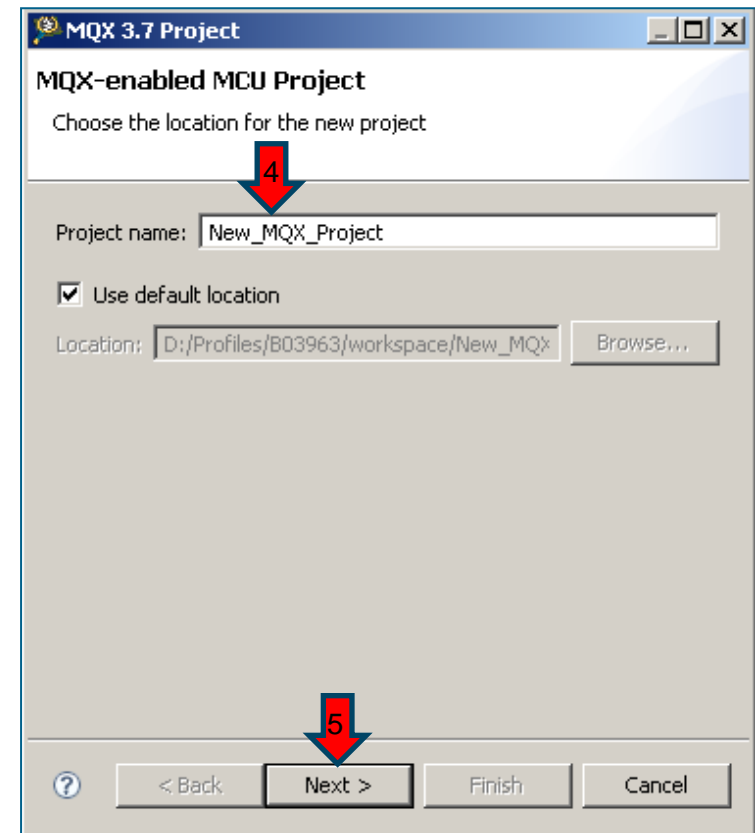
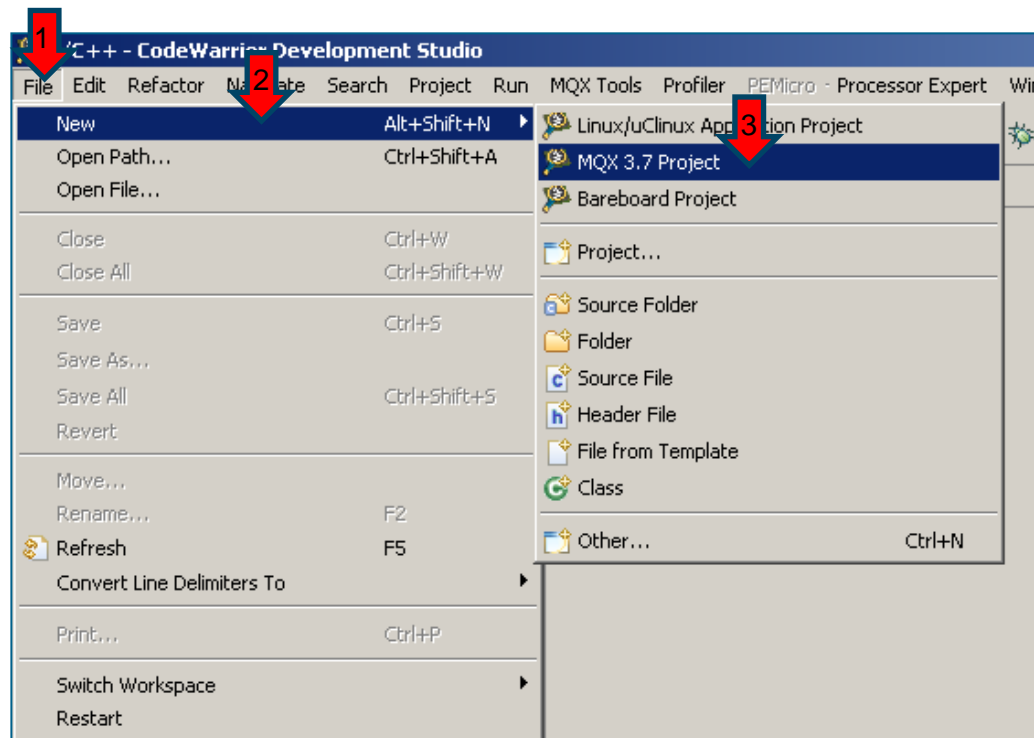


New MQX project



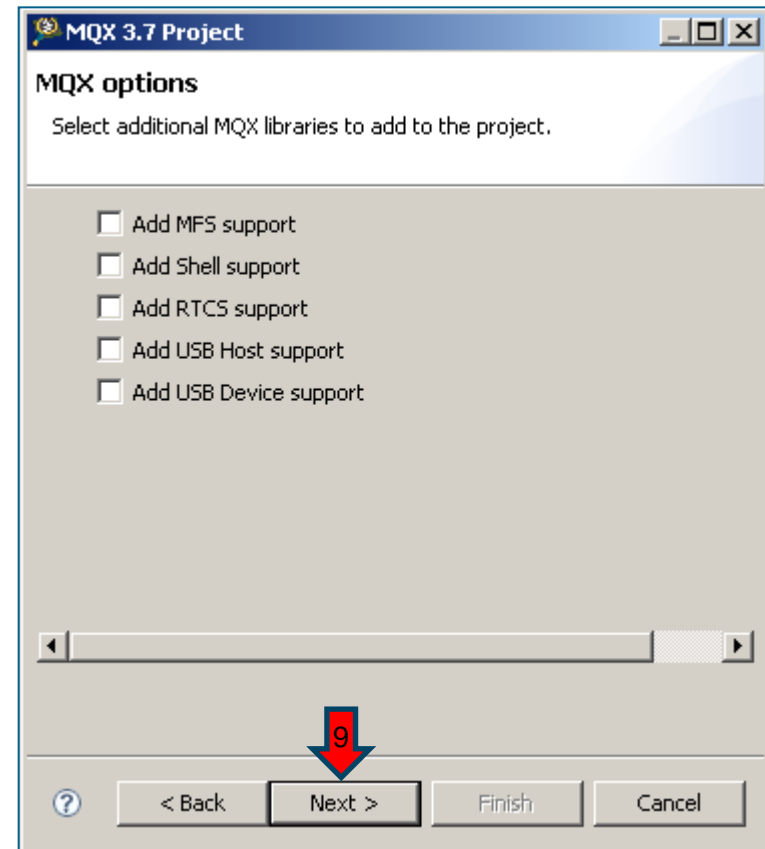
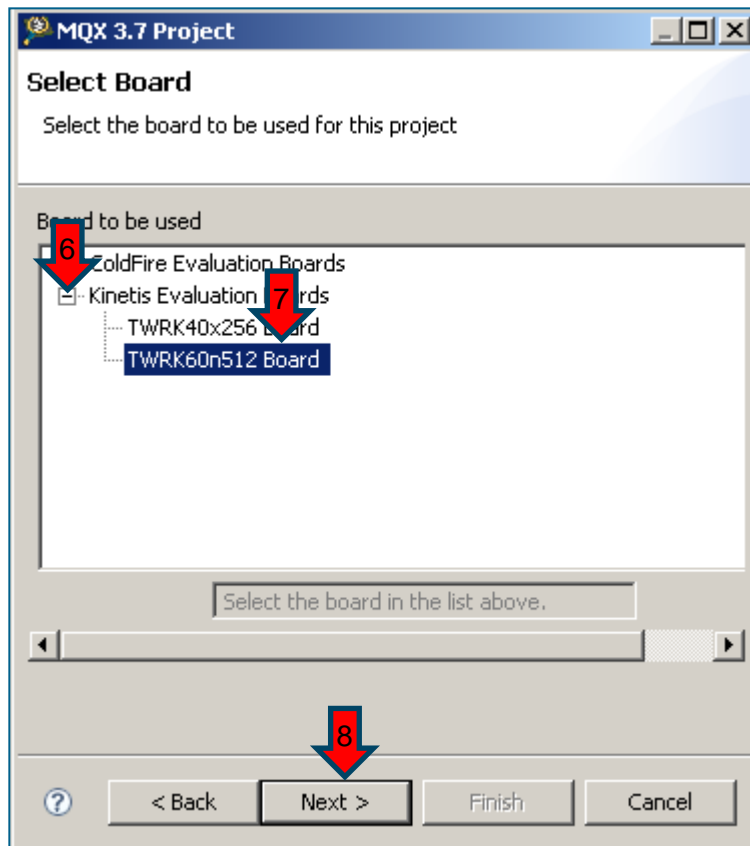
New MQX Project

- ▶ File -> New -> MQX 3.7 Project
- ▶ Give a name and click Next

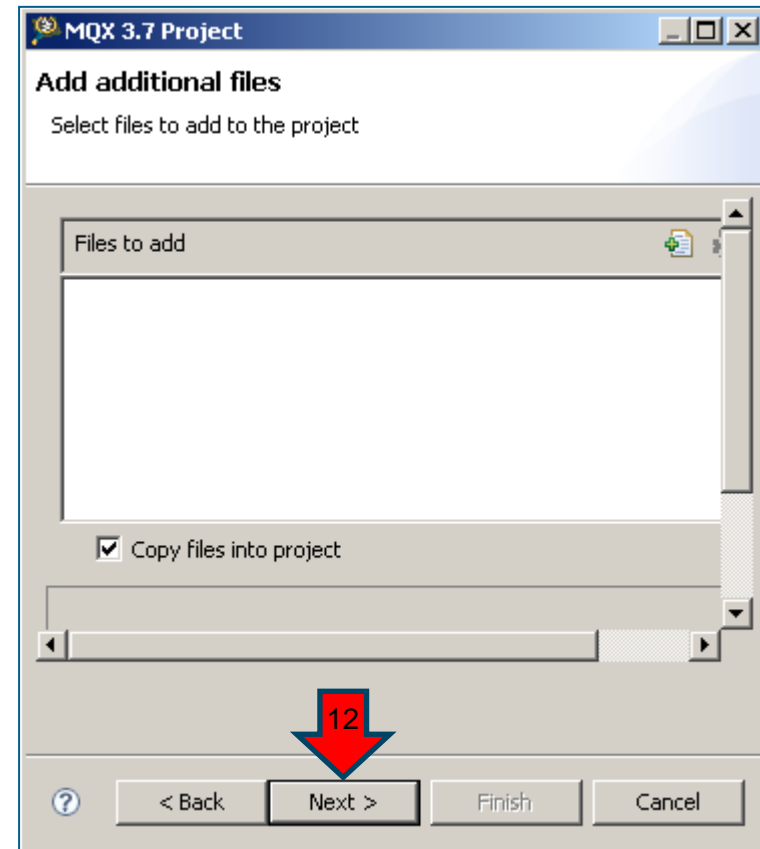
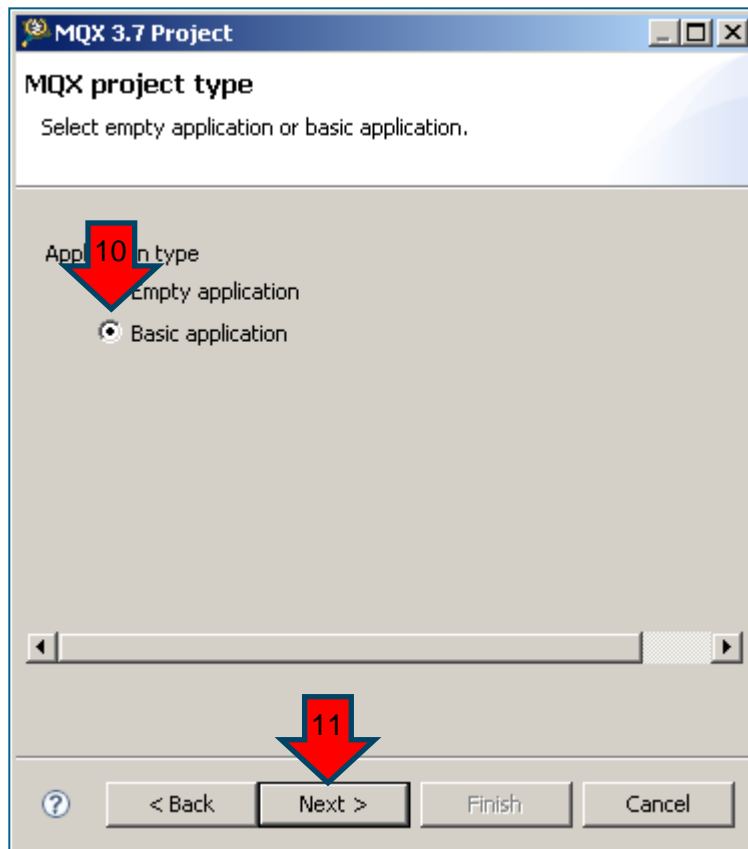


New MQX Project

► Select **TWRK60n512** Board

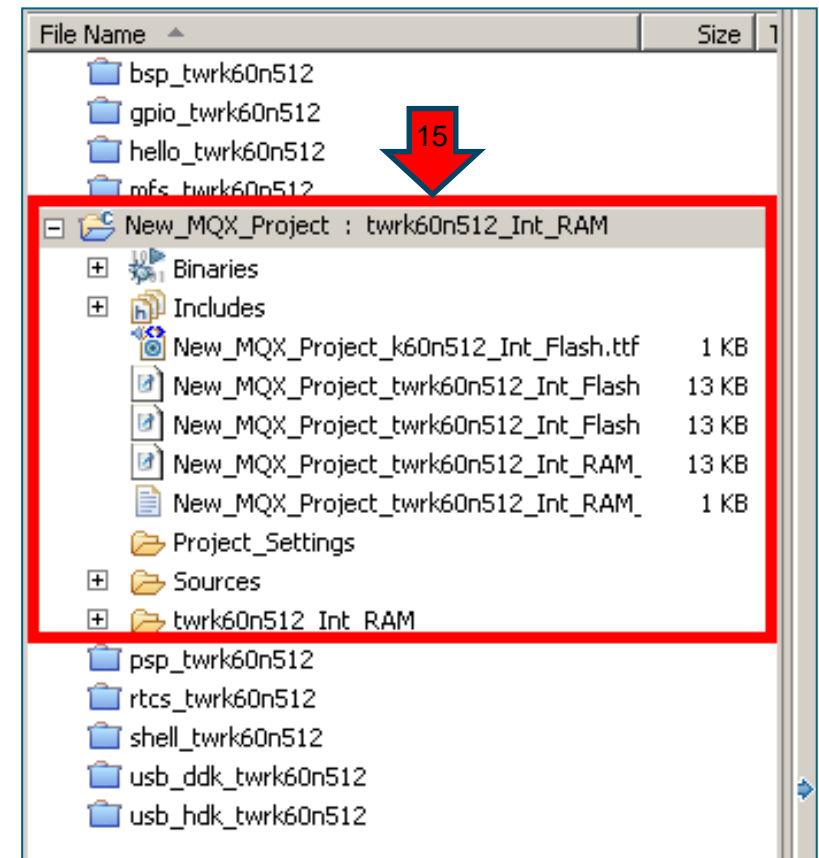
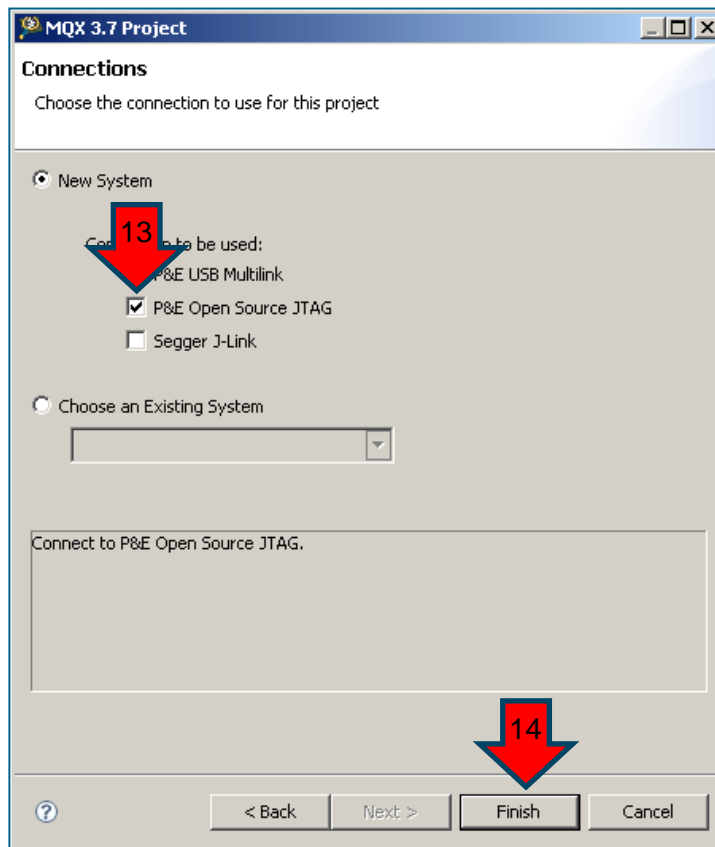


► Select Basic application



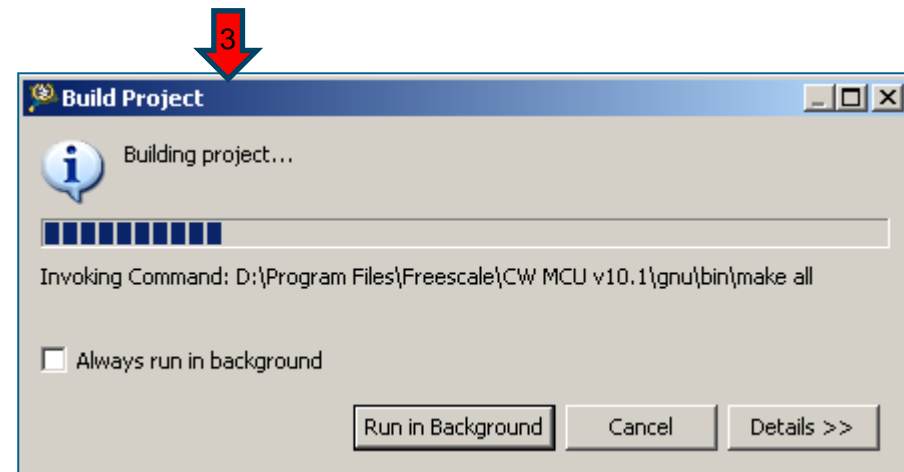
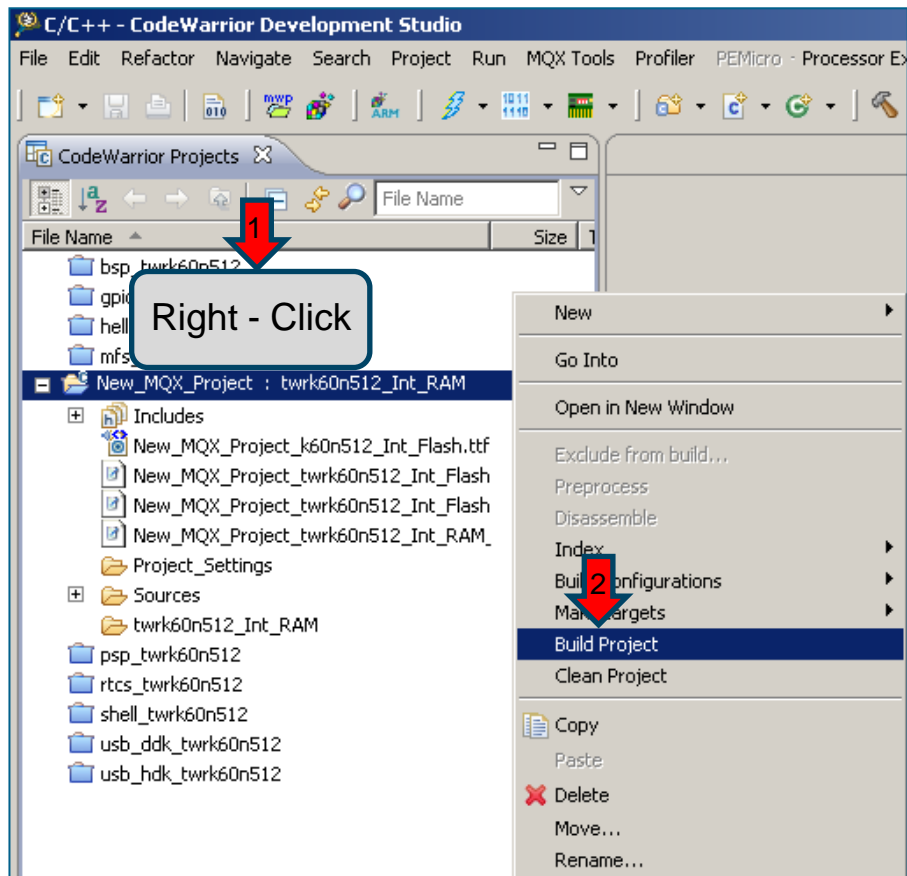
New MQX Project

- ▶ Select P&E Open Source JTAG
- ▶ Project is created



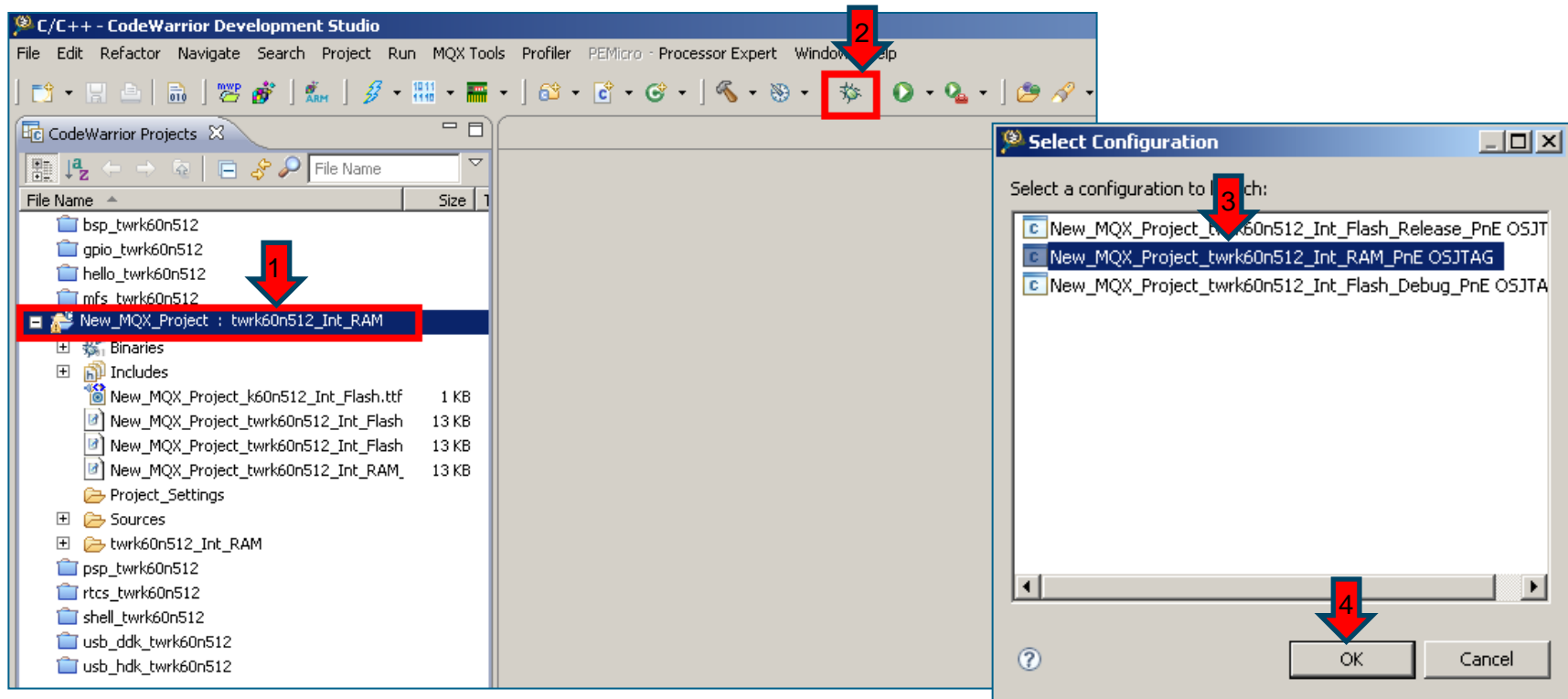
Build New MQX Project

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

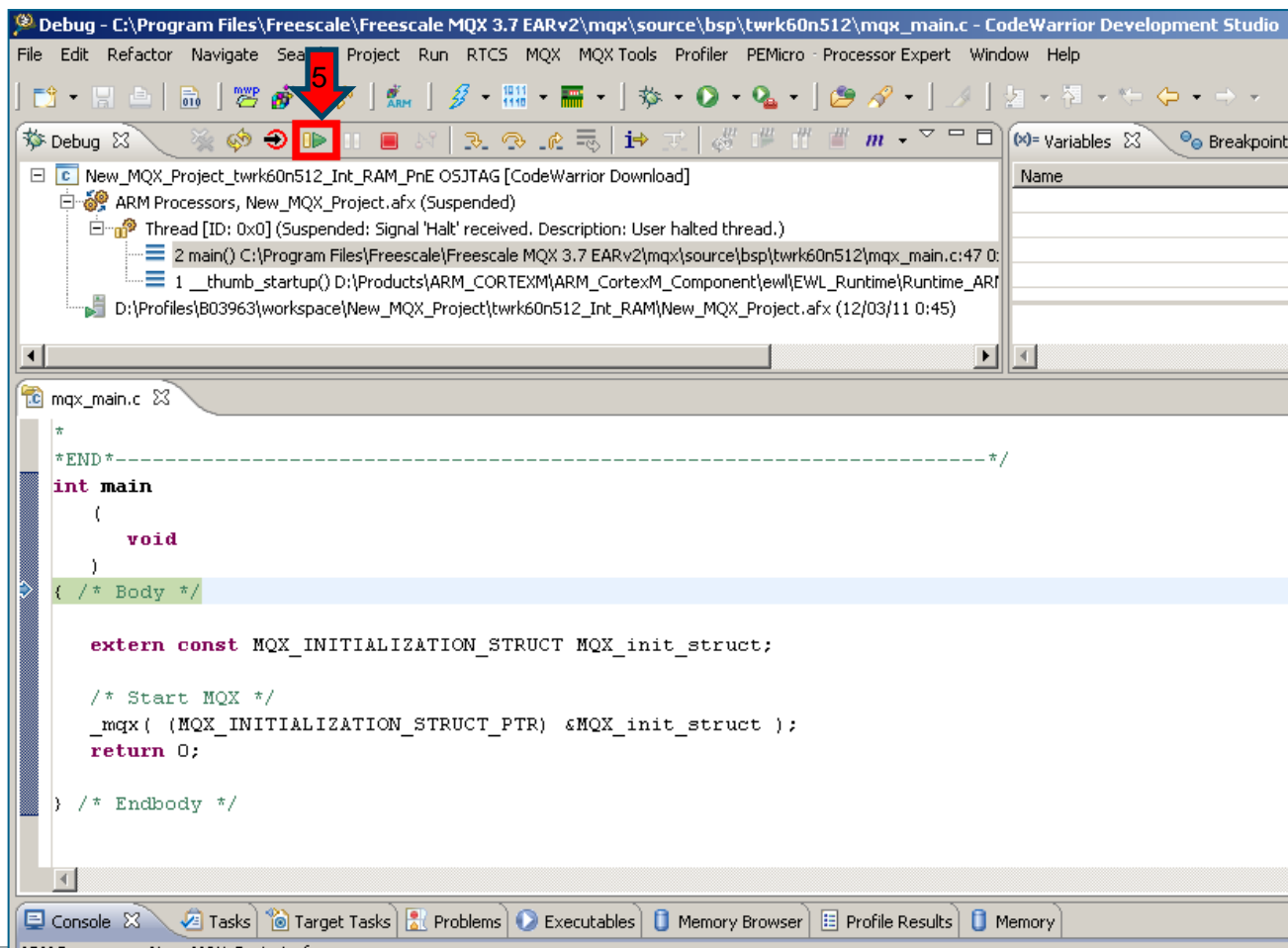


Debug New MQX Project

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

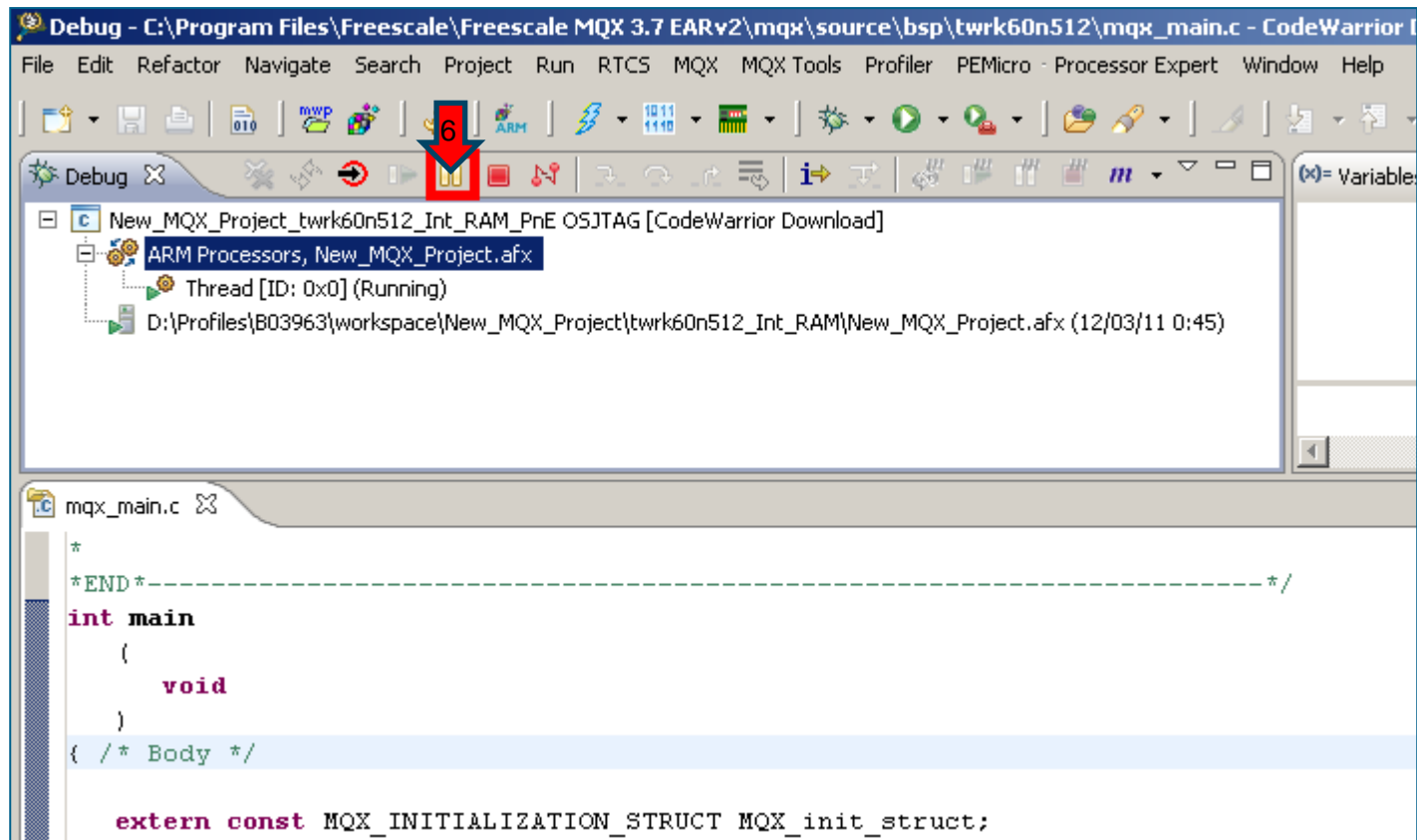


► Execute the code 'Run' icon

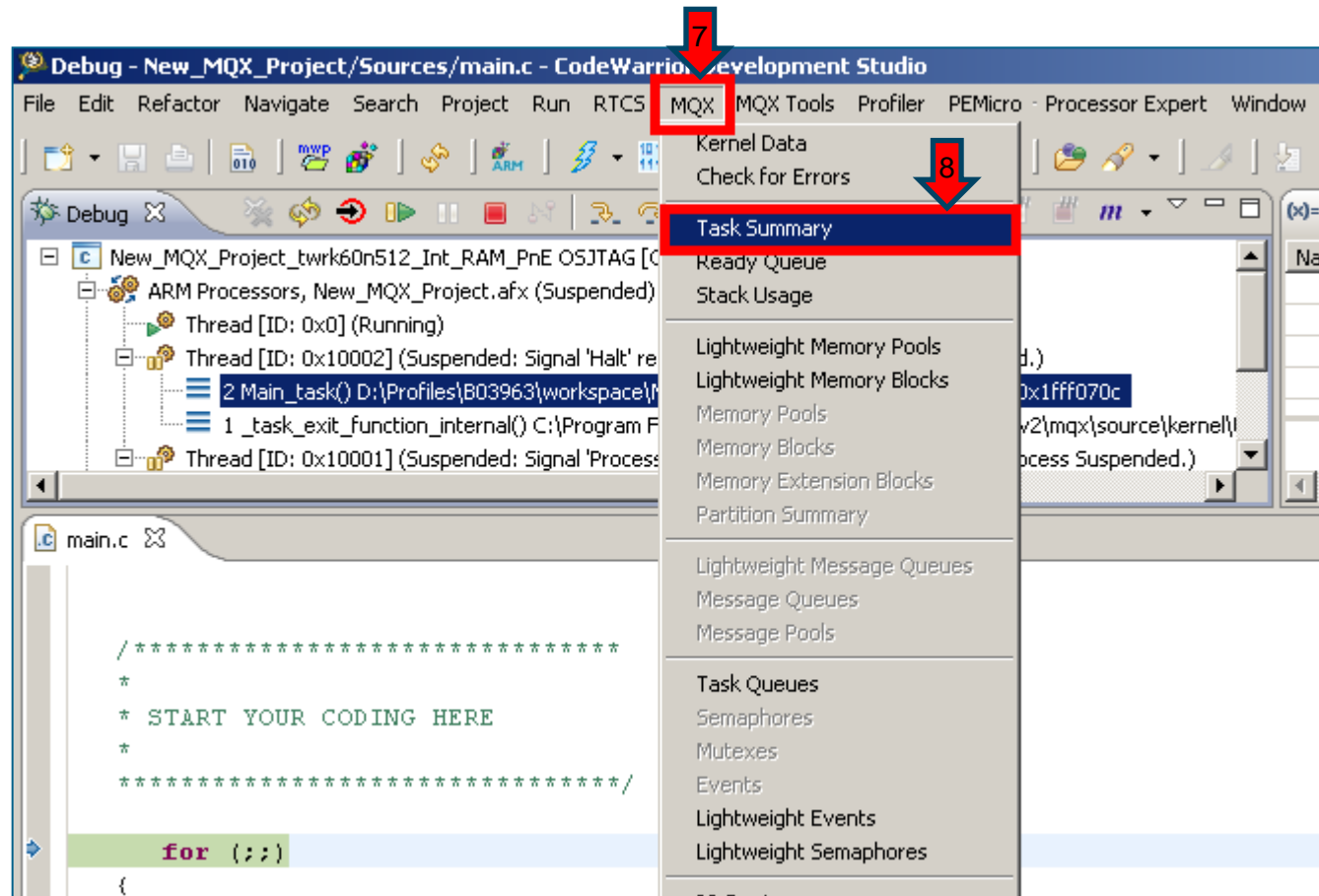


Run New MQX Project

► Pause execution

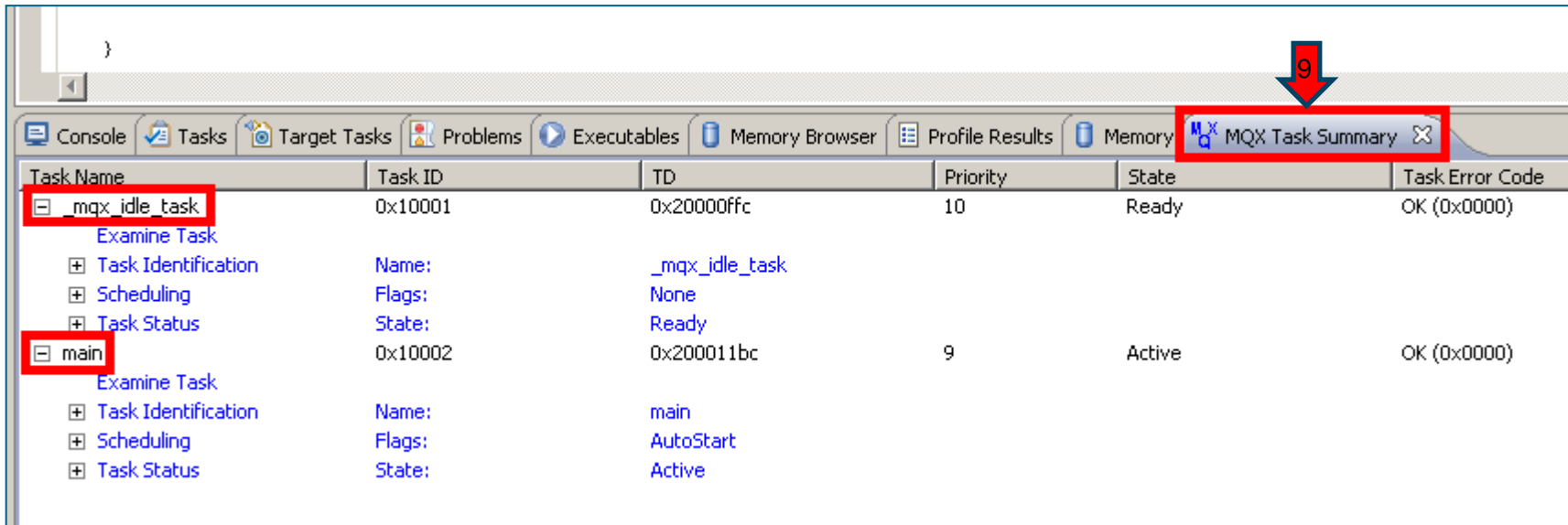


► MQX -> Task Summary



TAD: Task Summary

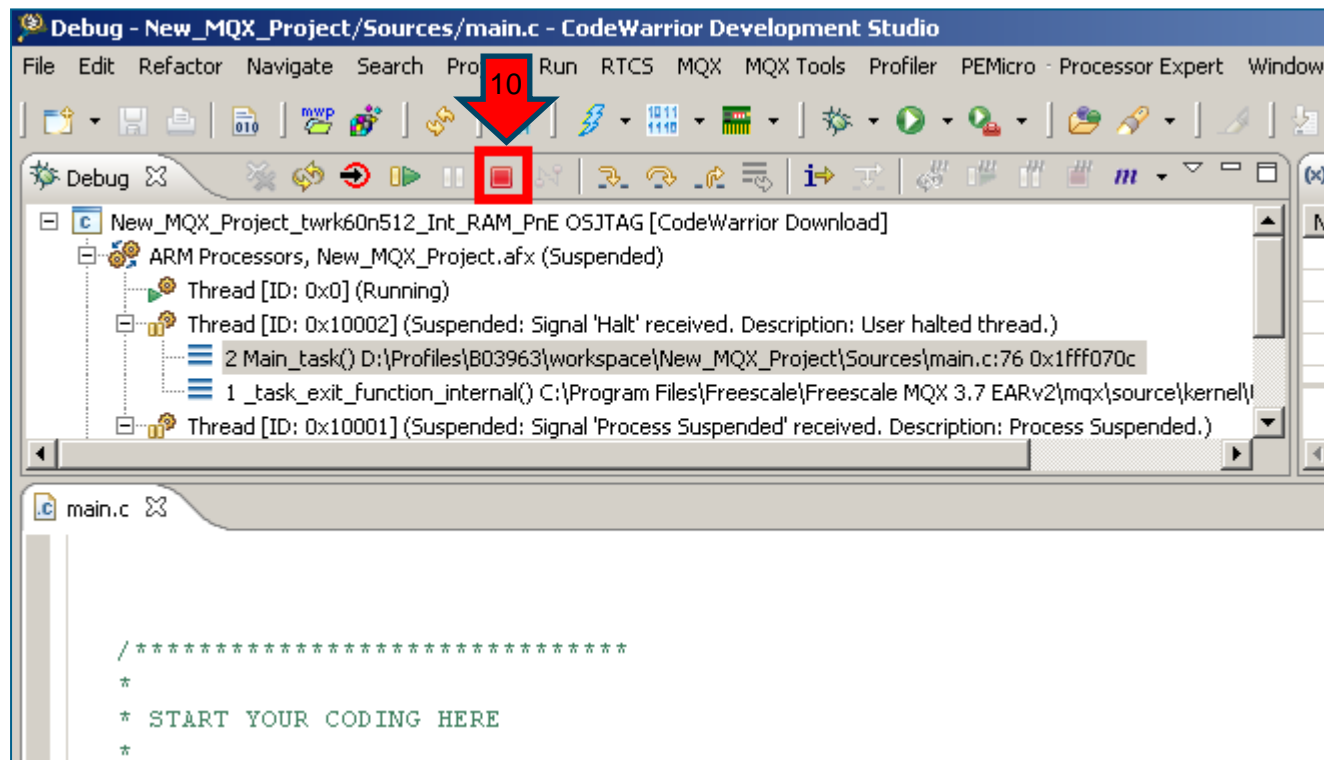
► Observe Tasks in your Application



The screenshot shows the 'MQX Task Summary' window in a development environment. The window has a toolbar with icons for Console, Tasks, Target Tasks, Problems, Executables, Memory Browser, Profile Results, and Memory. The main area displays a table of tasks:

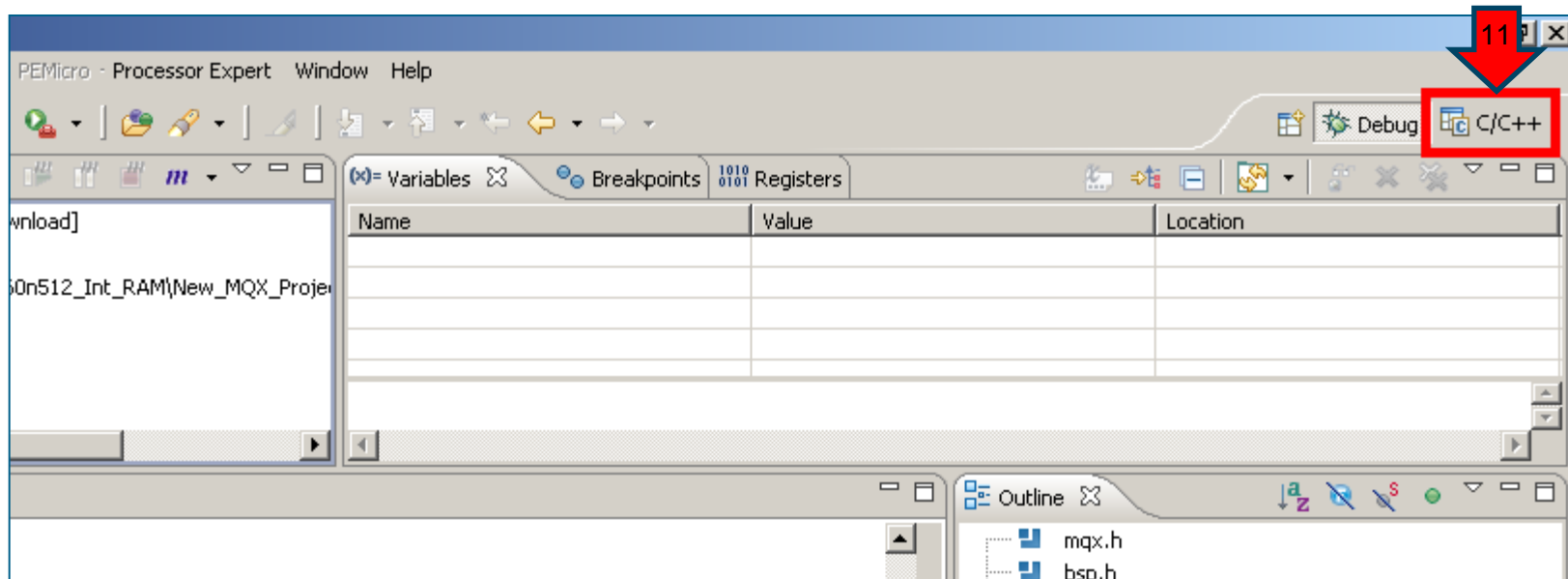
Task Name	Task ID	TD	Priority	State	Task Error Code
<input checked="" type="checkbox"/> _mqx_idle_task	0x10001	0x20000ffc	10	Ready	OK (0x0000)
Examine Task					
<input checked="" type="checkbox"/> Task Identification	Name:	_mqx_idle_task			
<input checked="" type="checkbox"/> Scheduling	Flags:	None	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

► Terminate execution

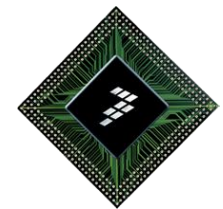


Run New MQX Project

► Change to C/C++ Perspective

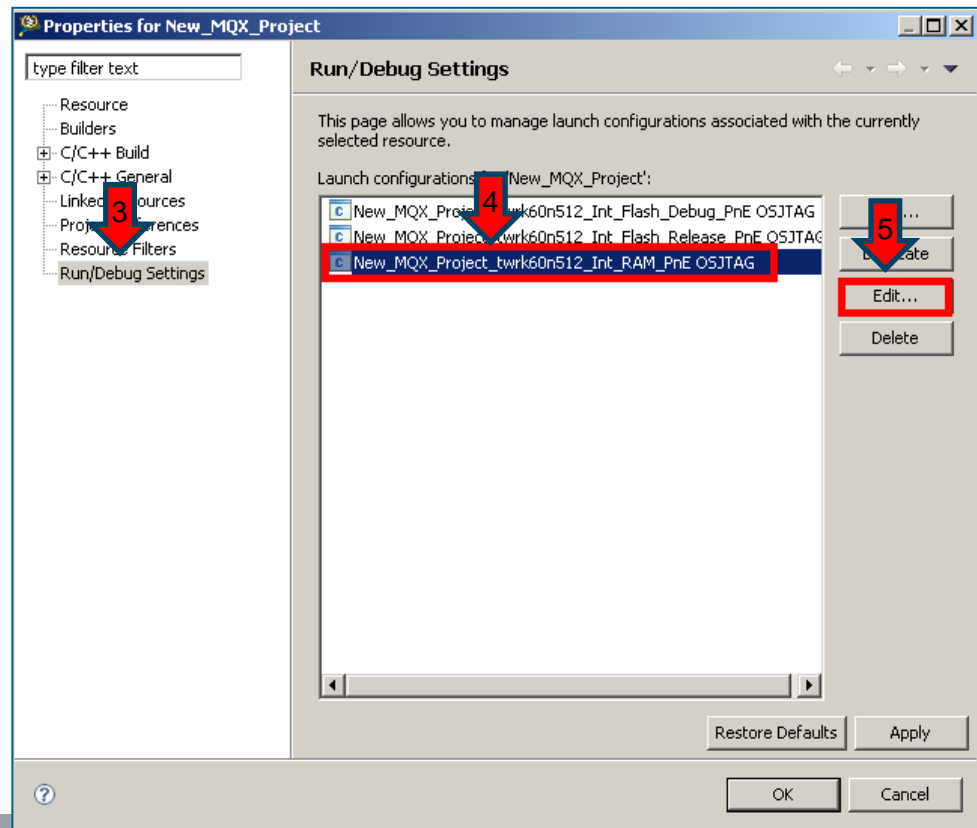
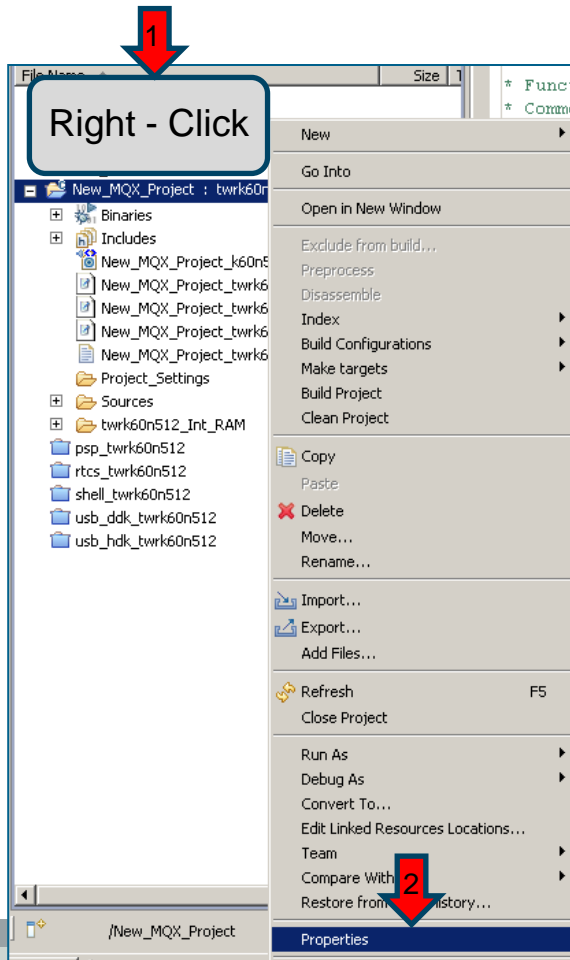


Debugging with J-Link



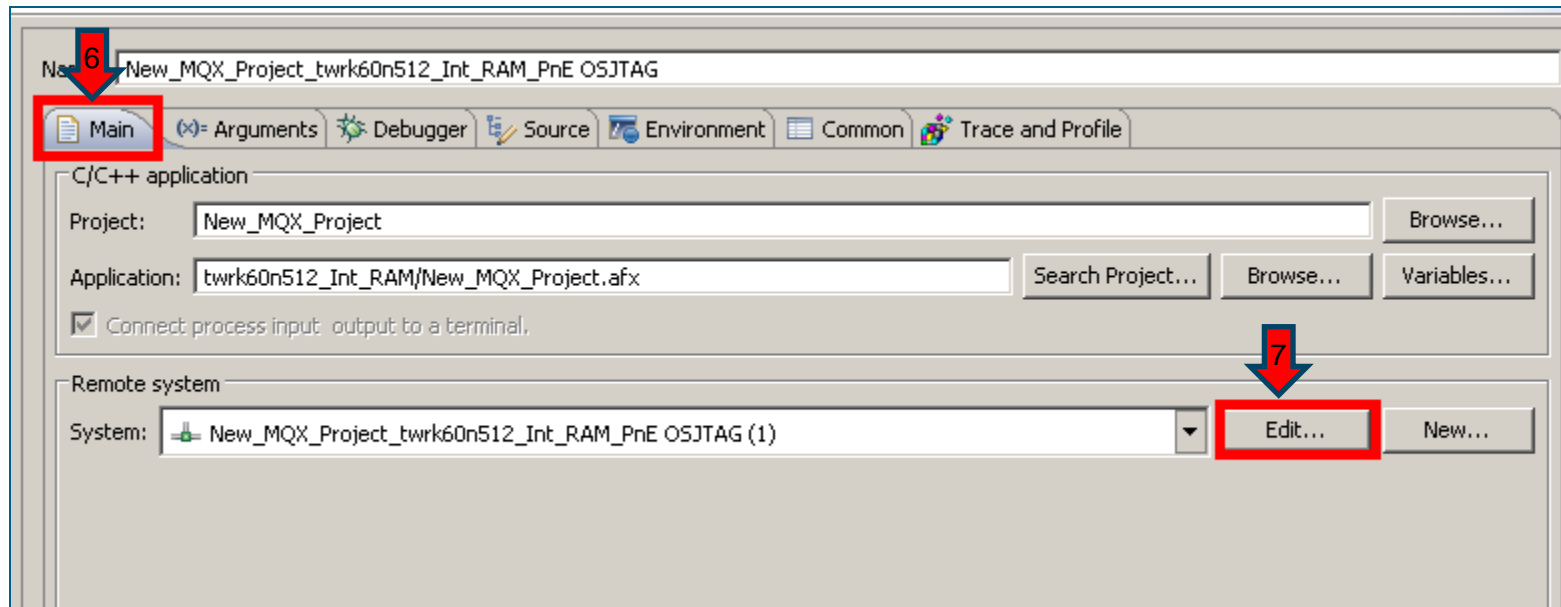
Change Connection Type

► Edit Connection Settings of the project



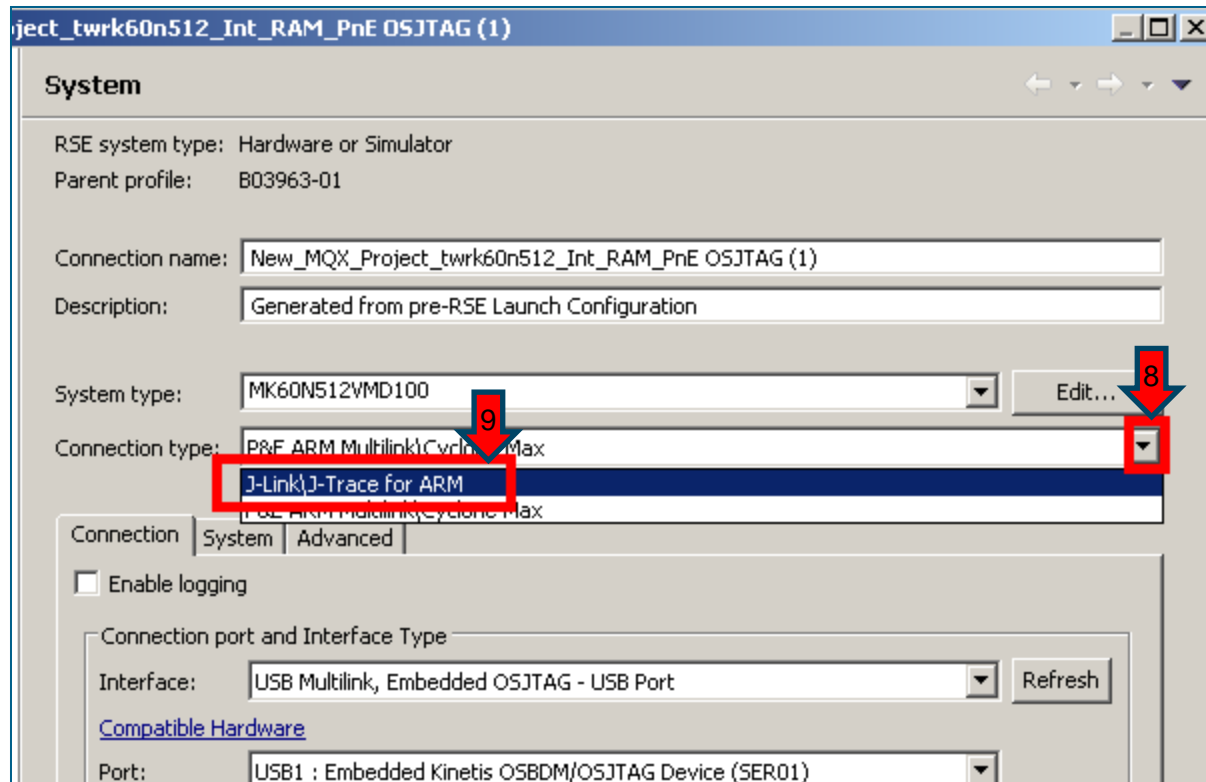
Change Connection Type

► Edit Remote System



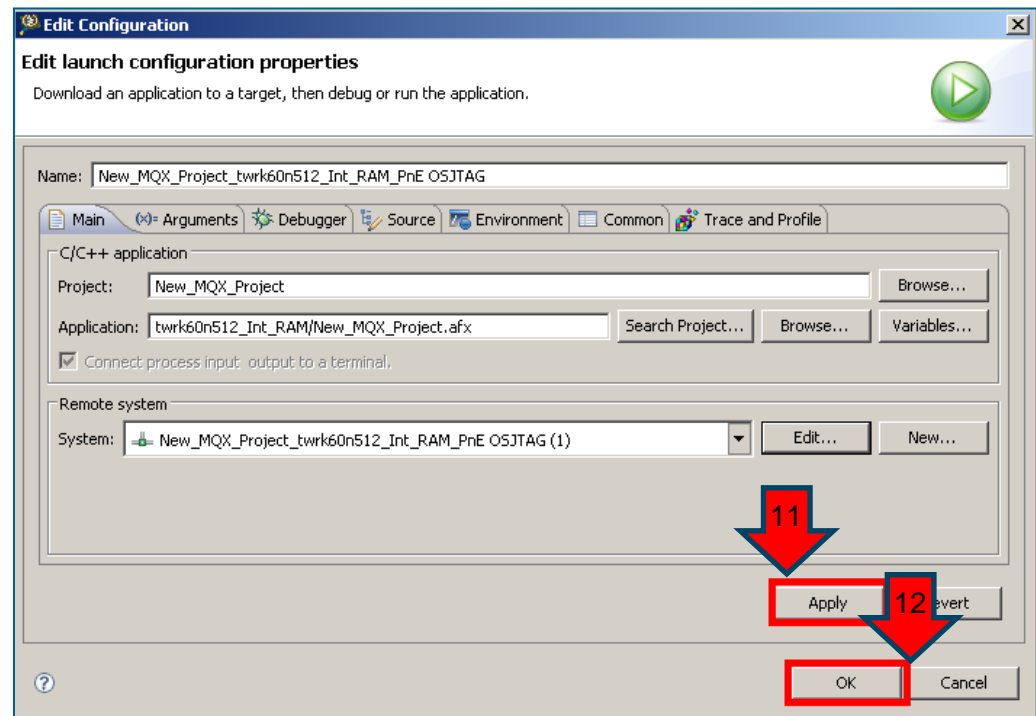
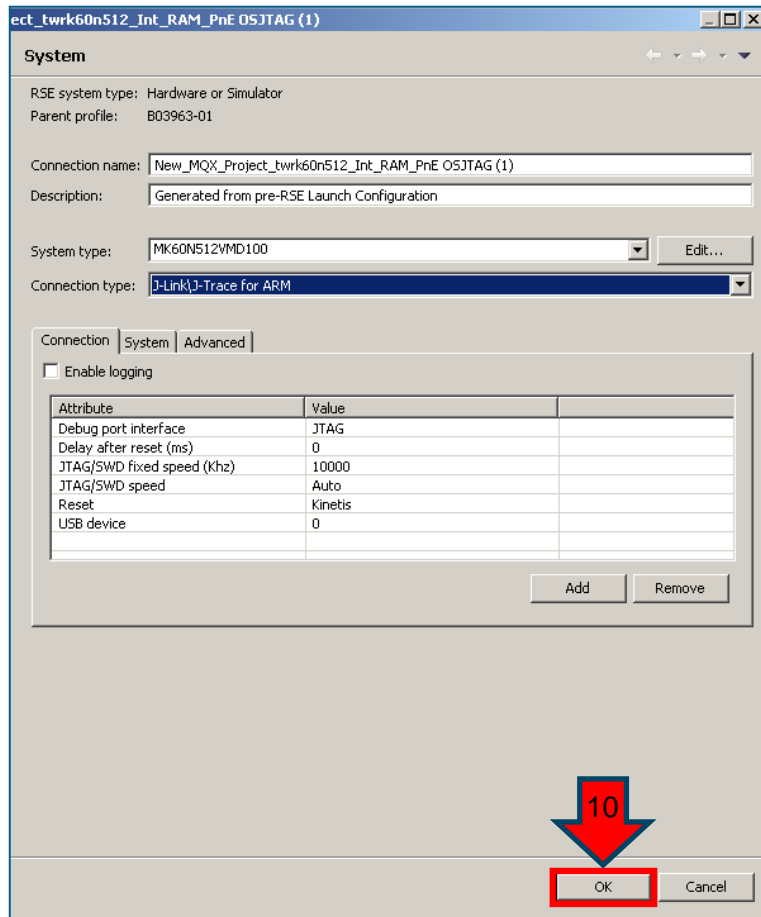
Change Connection Type

► Select J-Link\J-Trace for ARM



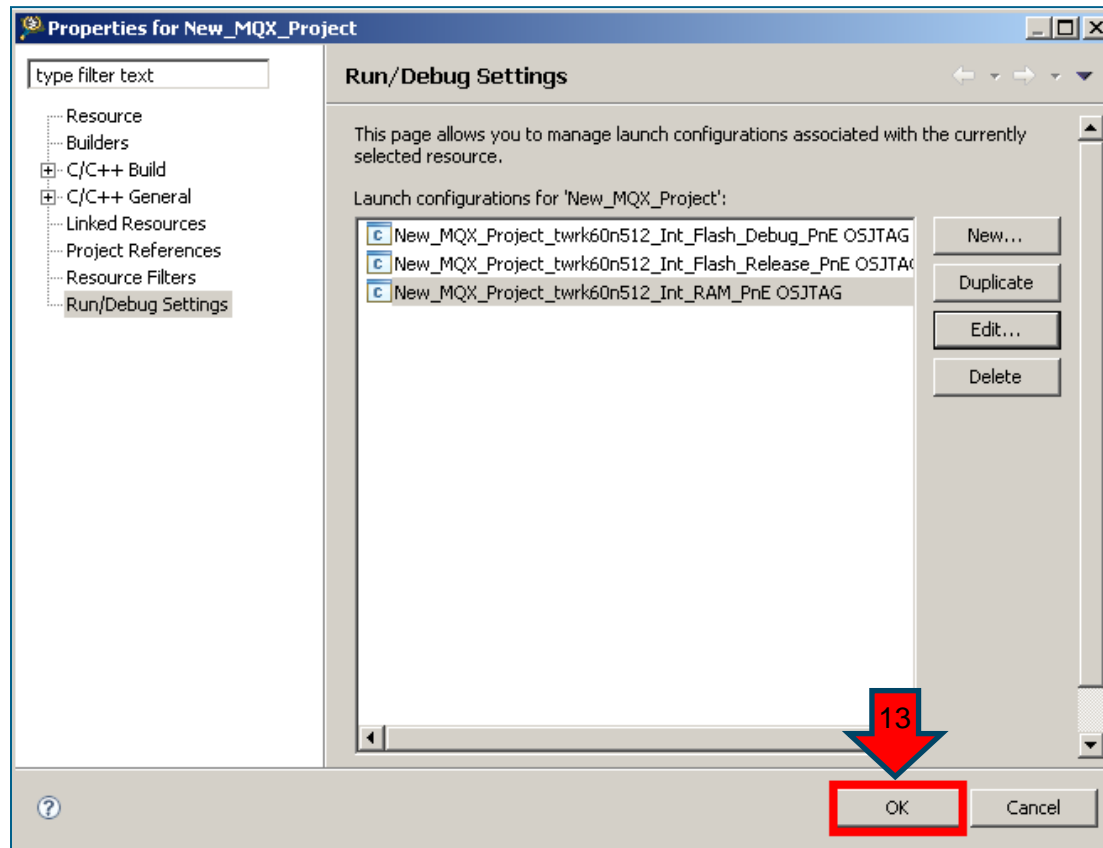
Change Connection Type

► Confirm changes



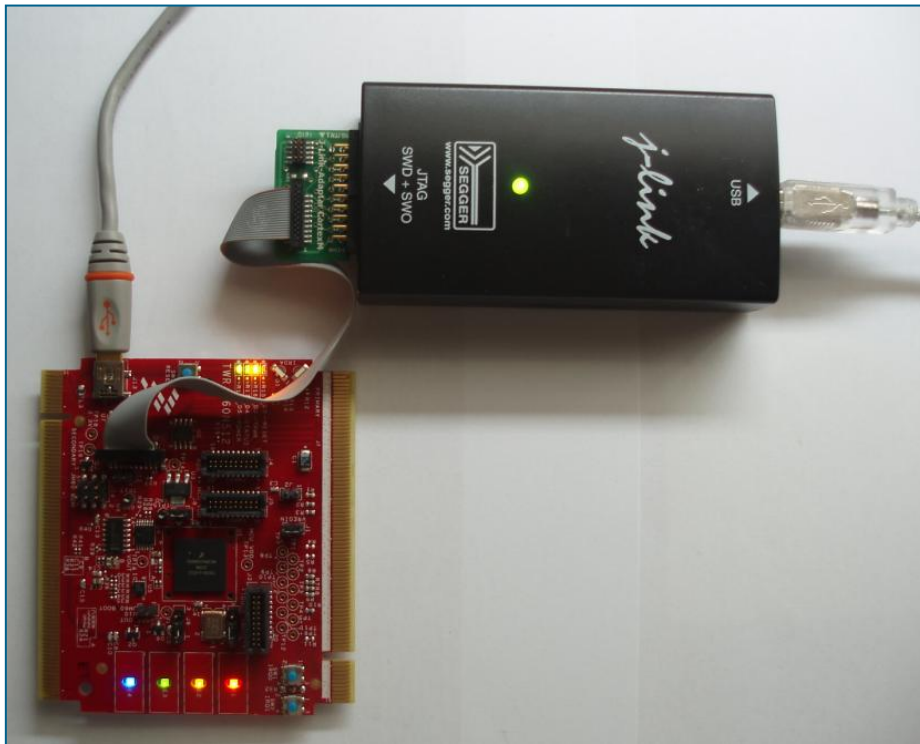
Change Connection Type

► Click OK



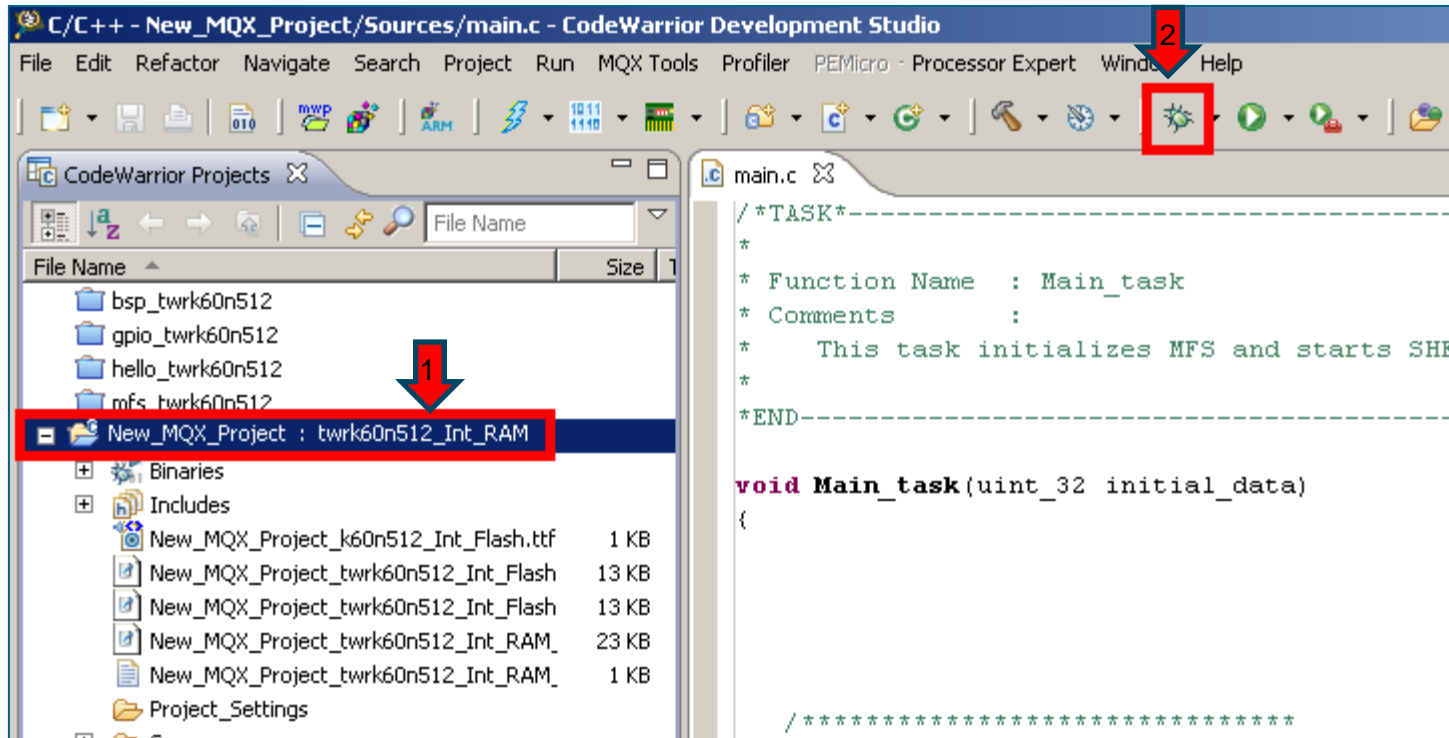
Debug with J-Link

- ▶ 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



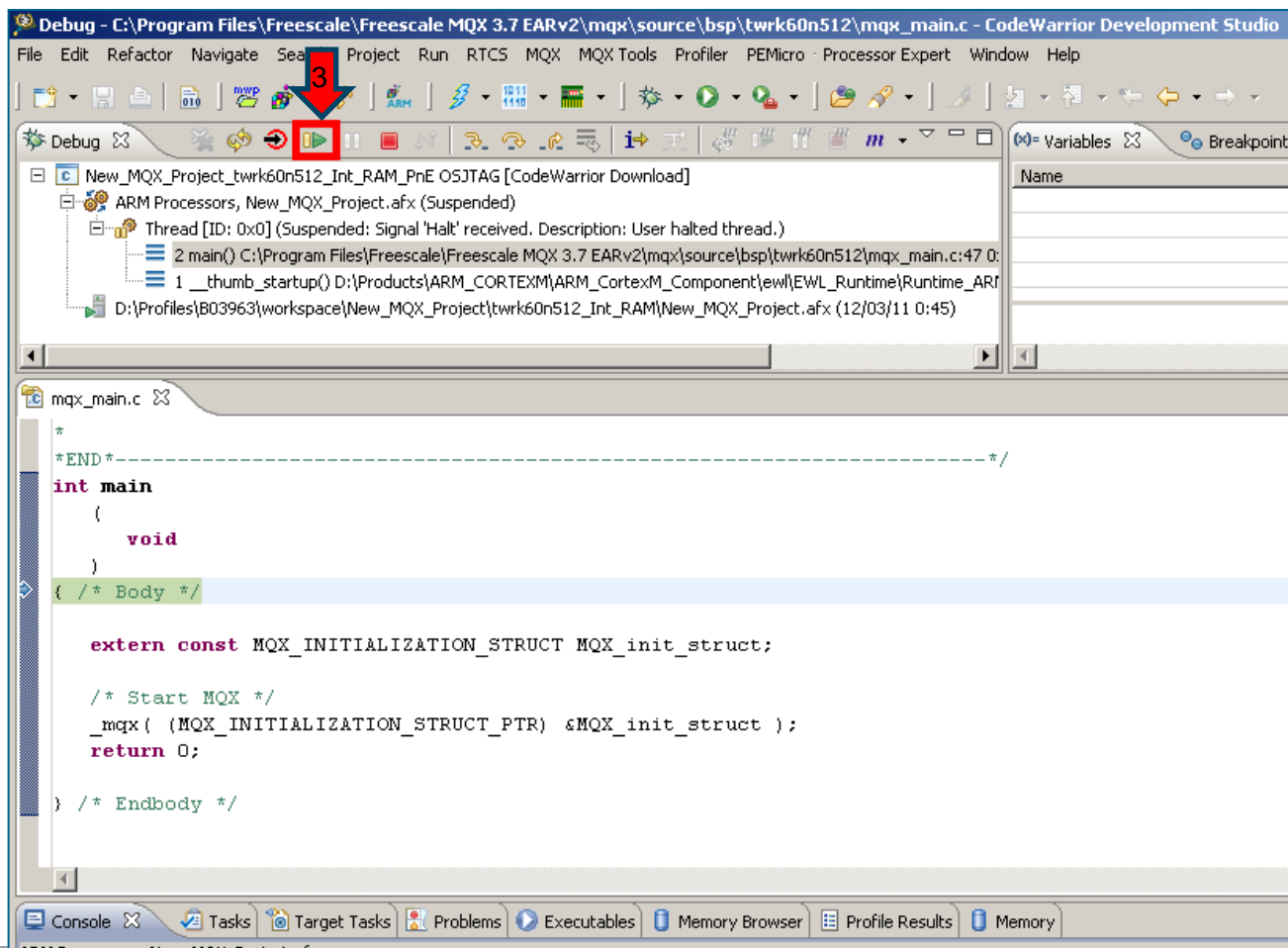
Change Connection Type

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

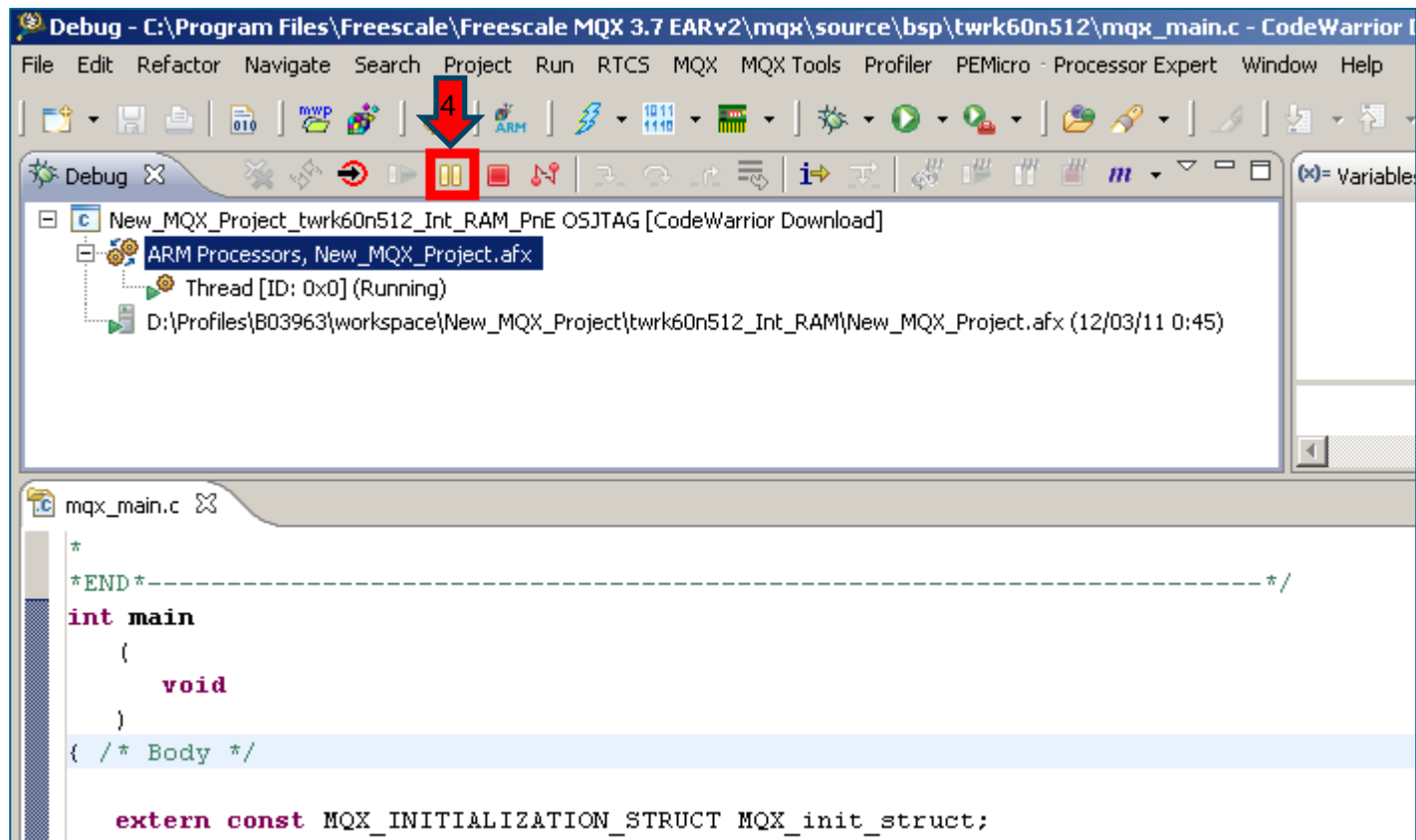


Run New MQX Project

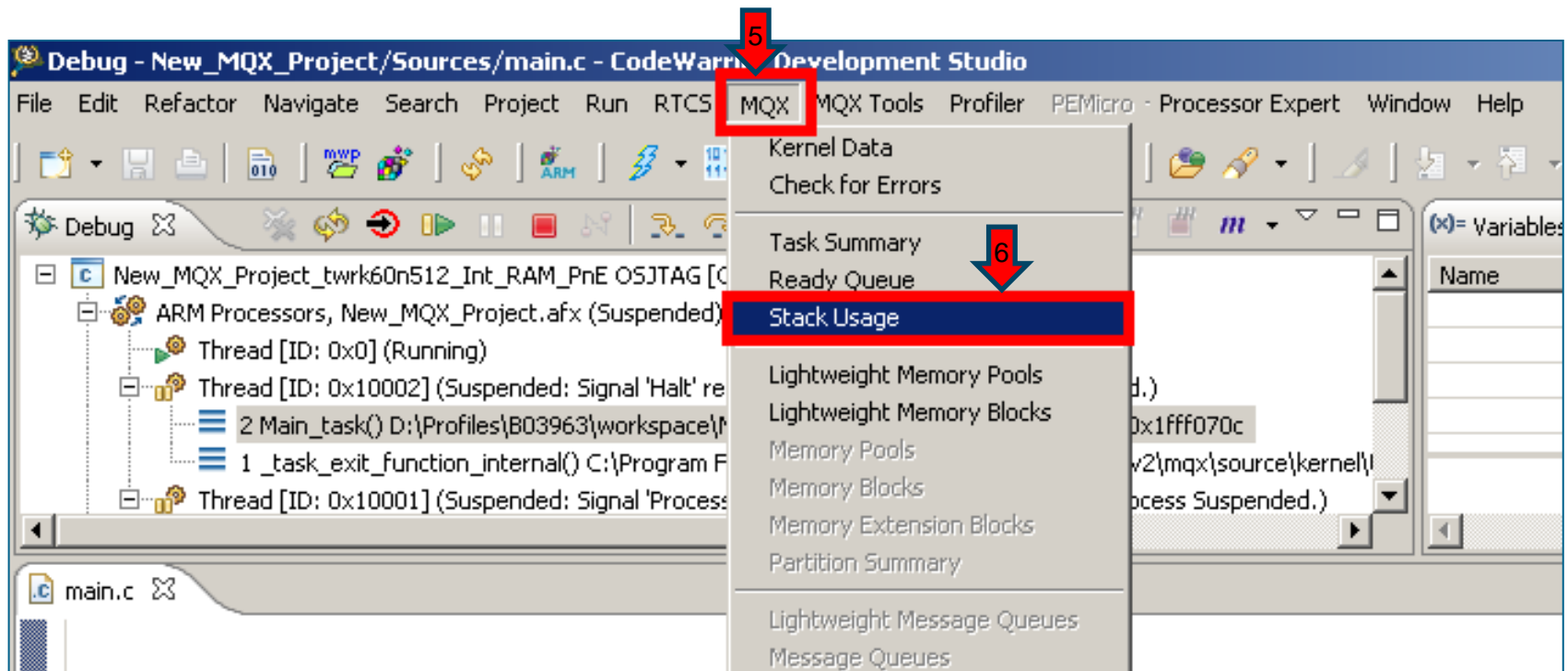
► Execute the code 'Resume' icon



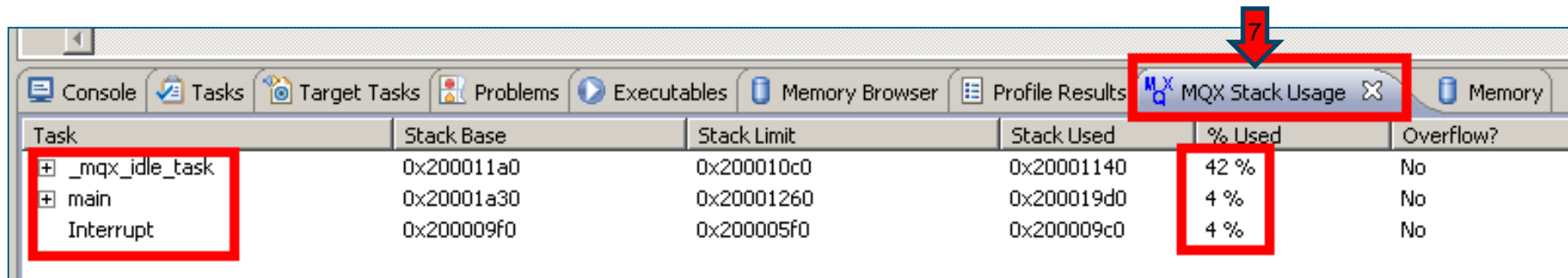
► Pause execution



► MQX -> Stack Usage

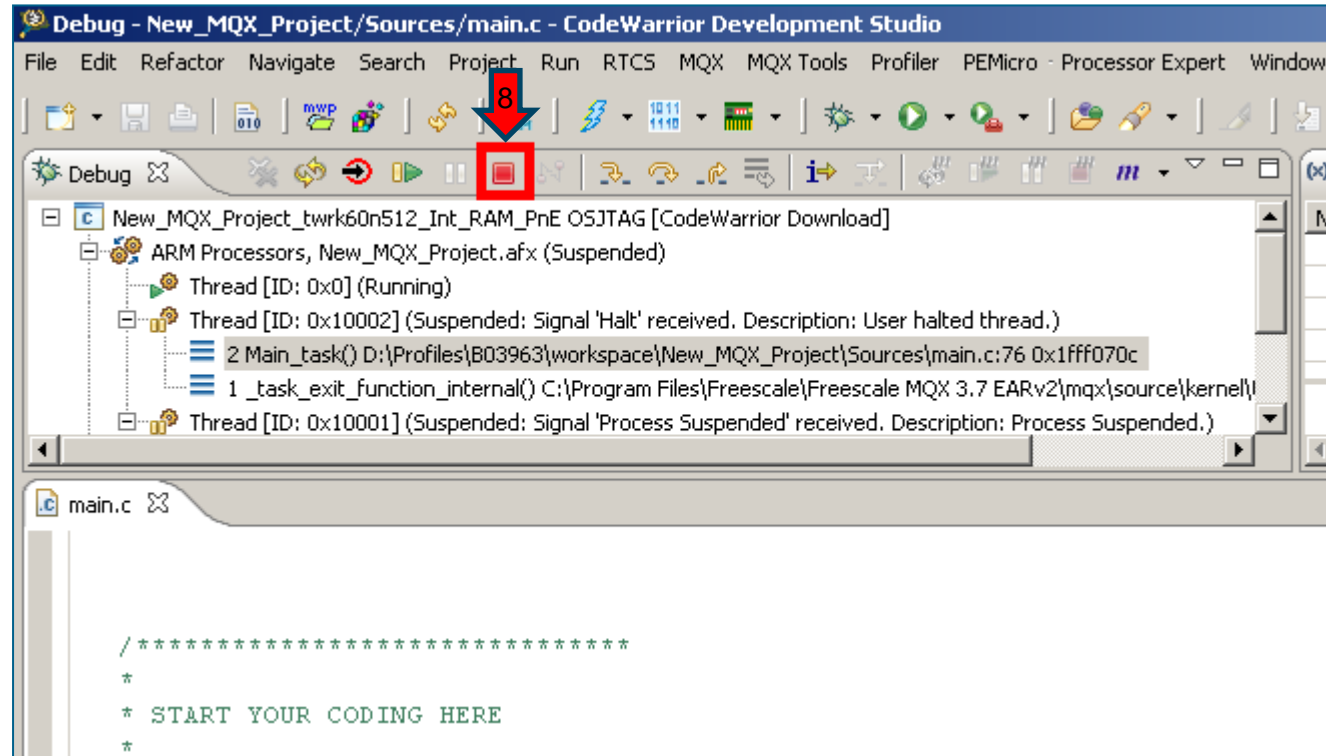


► Observe Stack Data



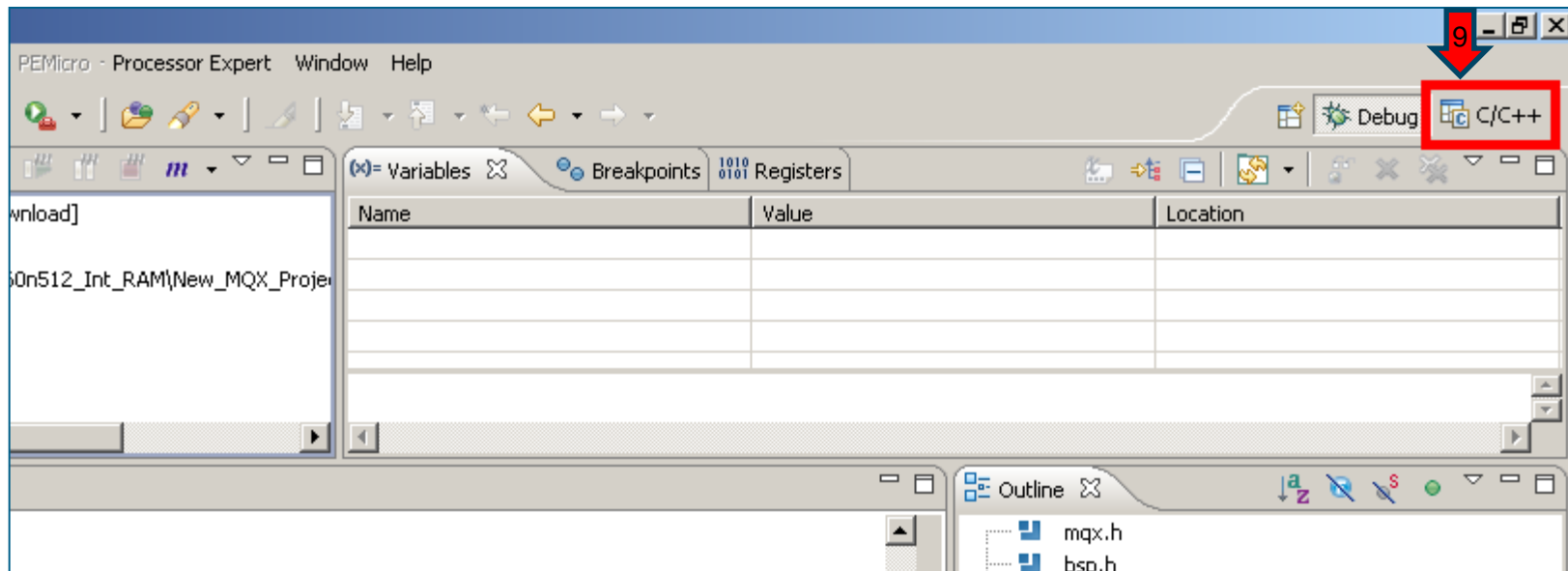
Task	Stack Base	Stack Limit	Stack Used	% Used	Overflow?
+ _mqx_idle_task	0x200011a0	0x200010c0	0x20001140	42 %	No
+ main	0x20001a30	0x20001260	0x200019d0	4 %	No
Interrupt	0x200009f0	0x200005f0	0x200009c0	4 %	No

► Terminate execution

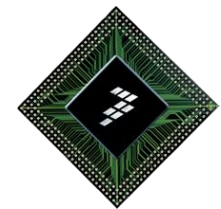


Run New MQX Project

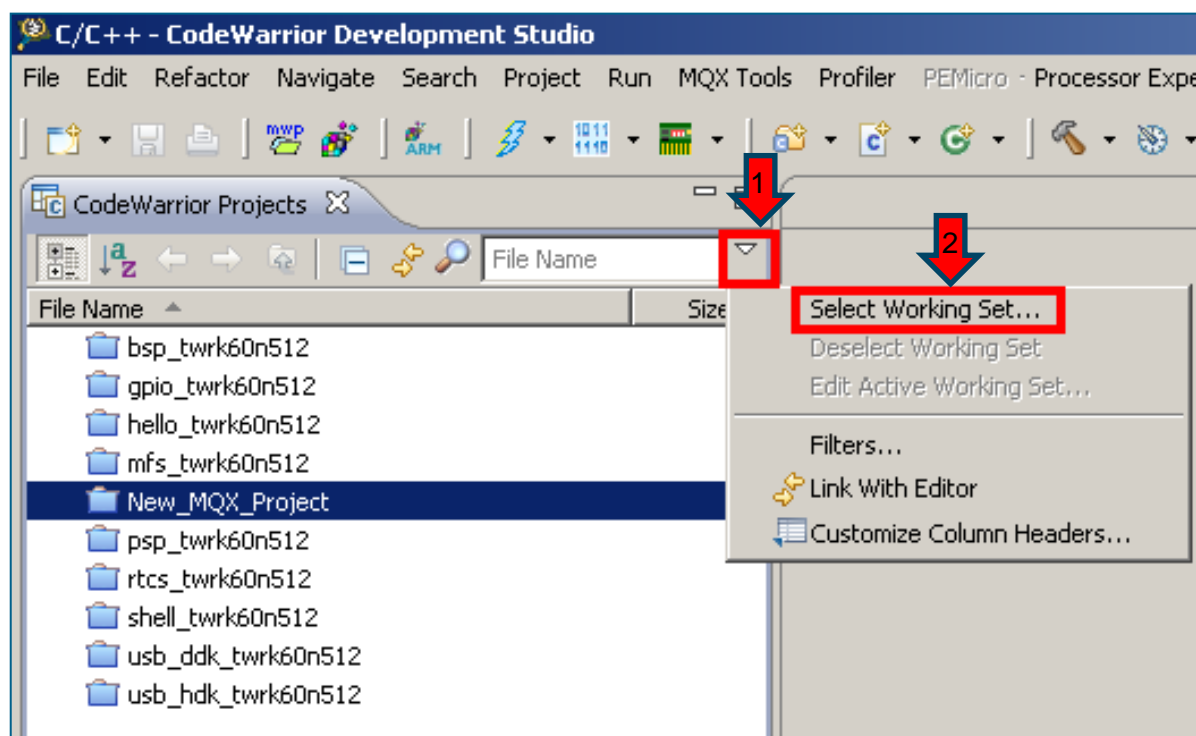
► Change to C/C++ Perspective



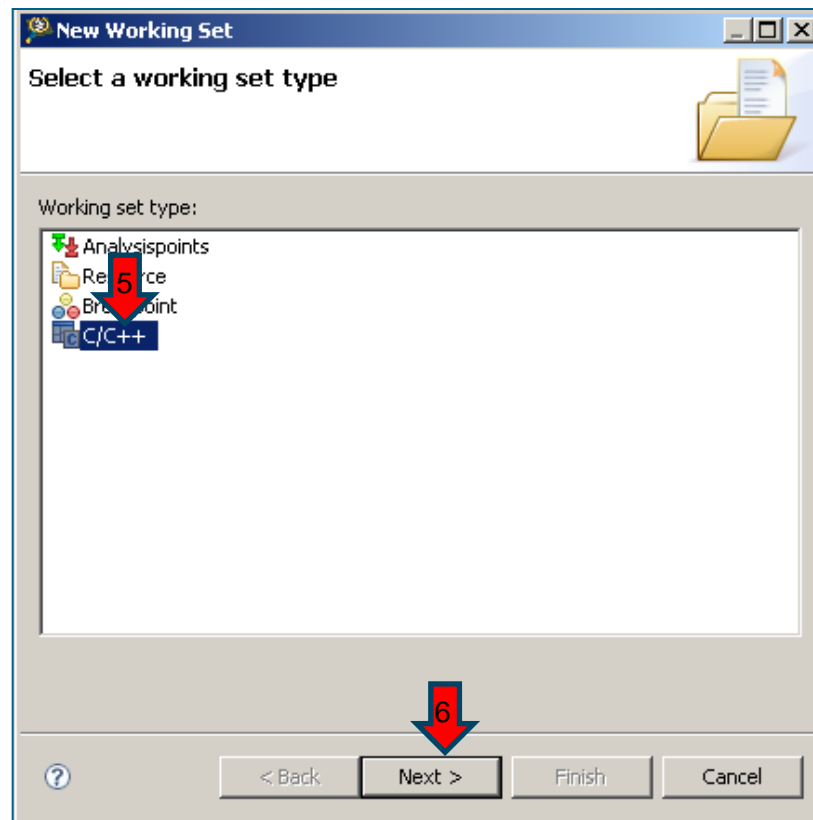
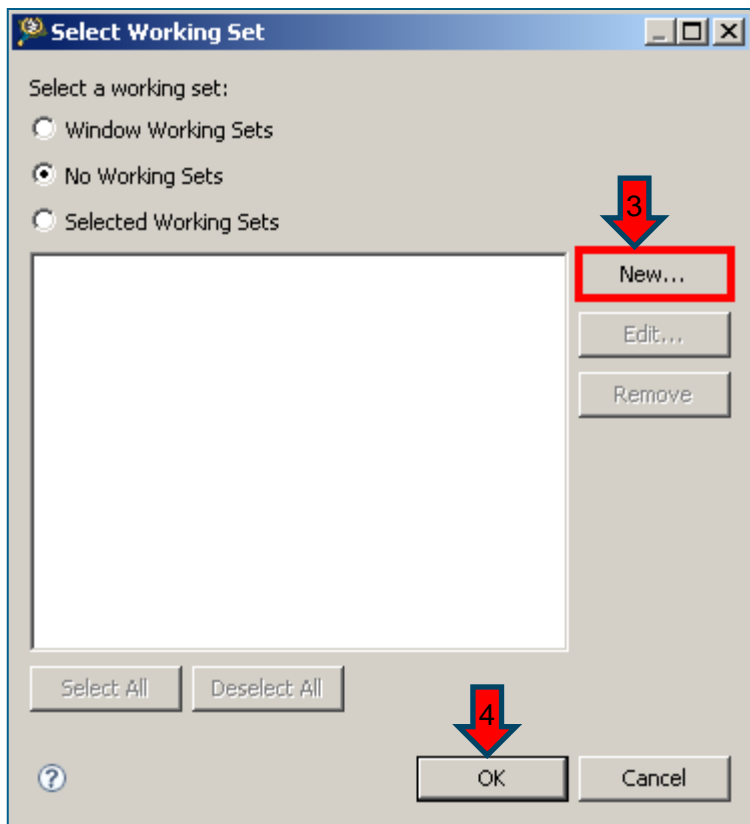
Eclipse Working Set



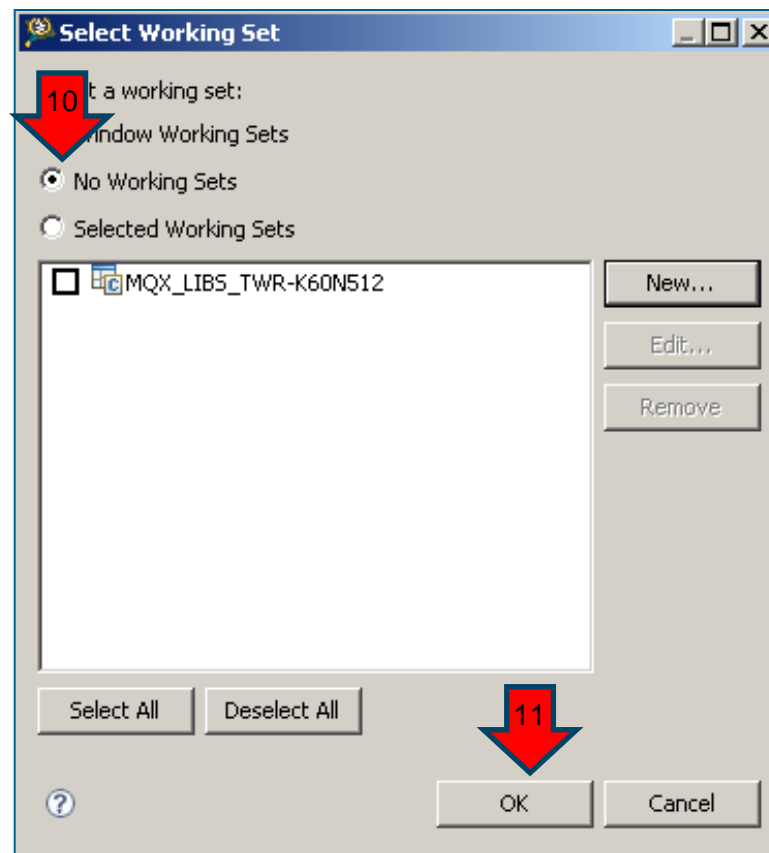
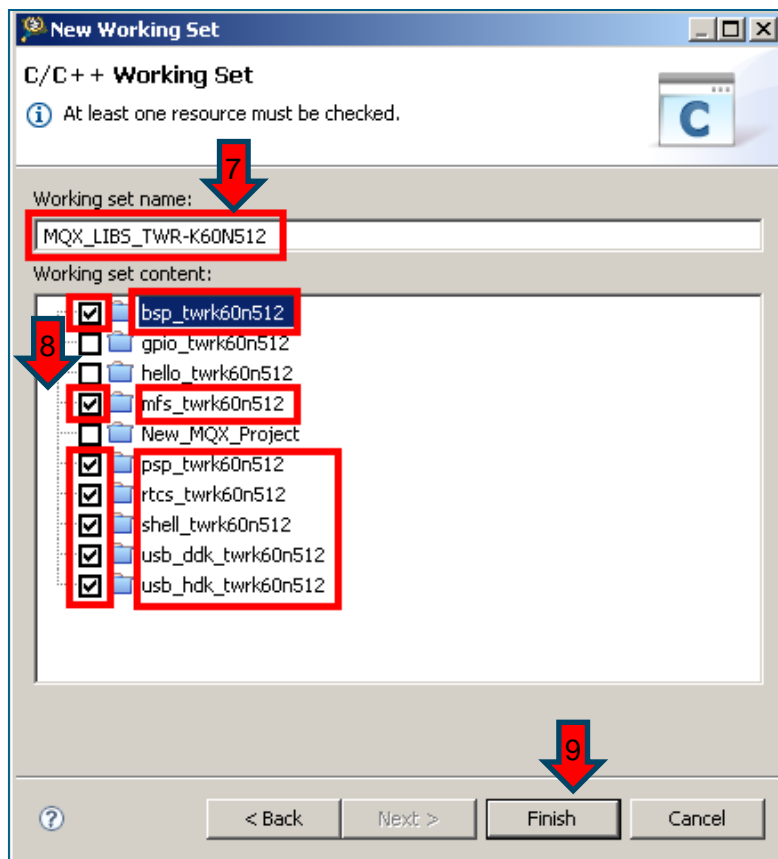
- ▶ If you are managing many projects and some MQX libraries for different Freescale architectures, is useful to create a “Working Set”
- ▶ Click in the CodeWarrior Projects view or Project Explorer view toolbar



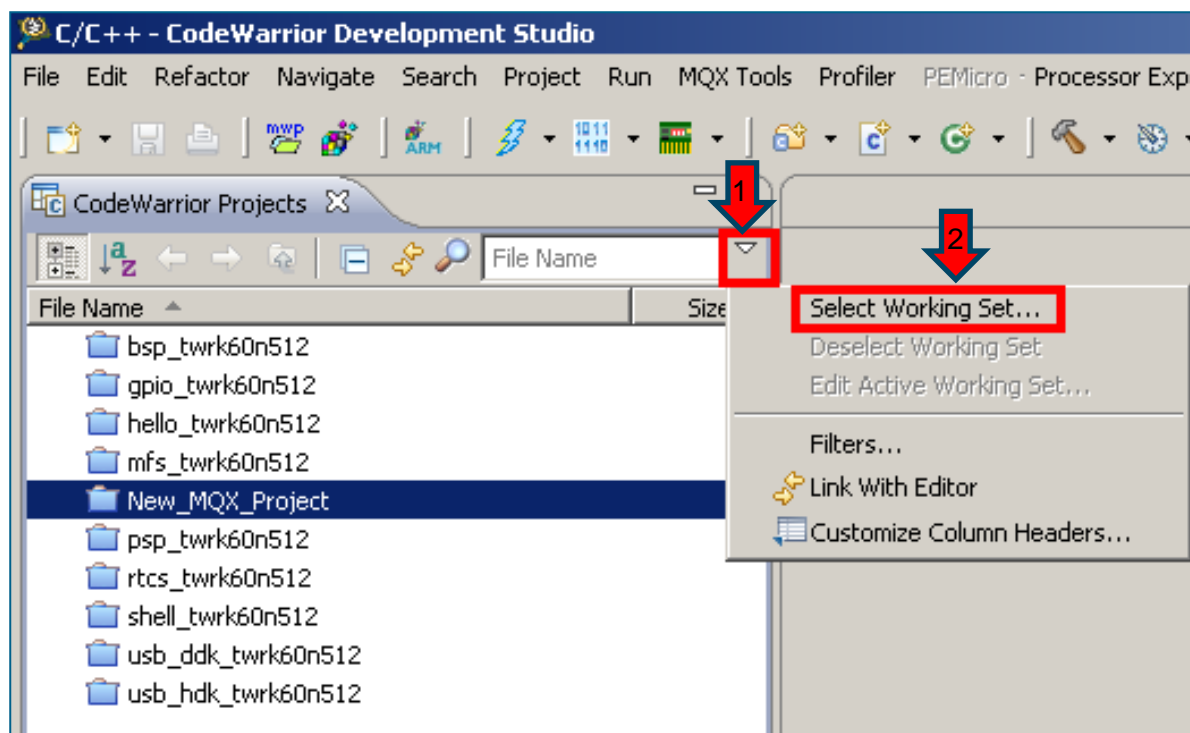
- ▶ Click New and OK
- ▶ Select C/C++ and Next



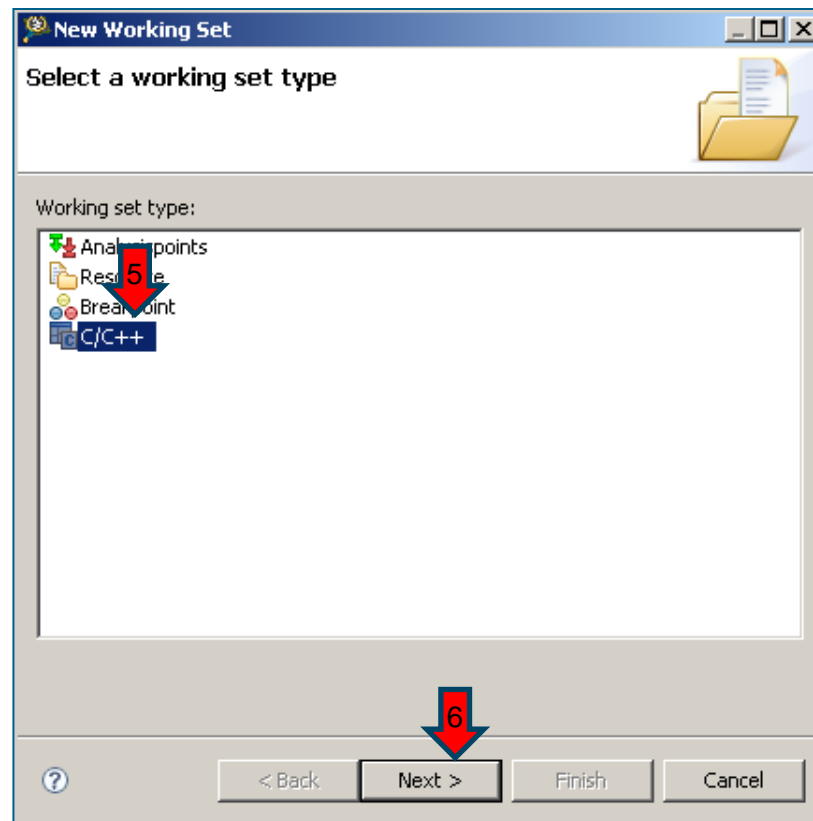
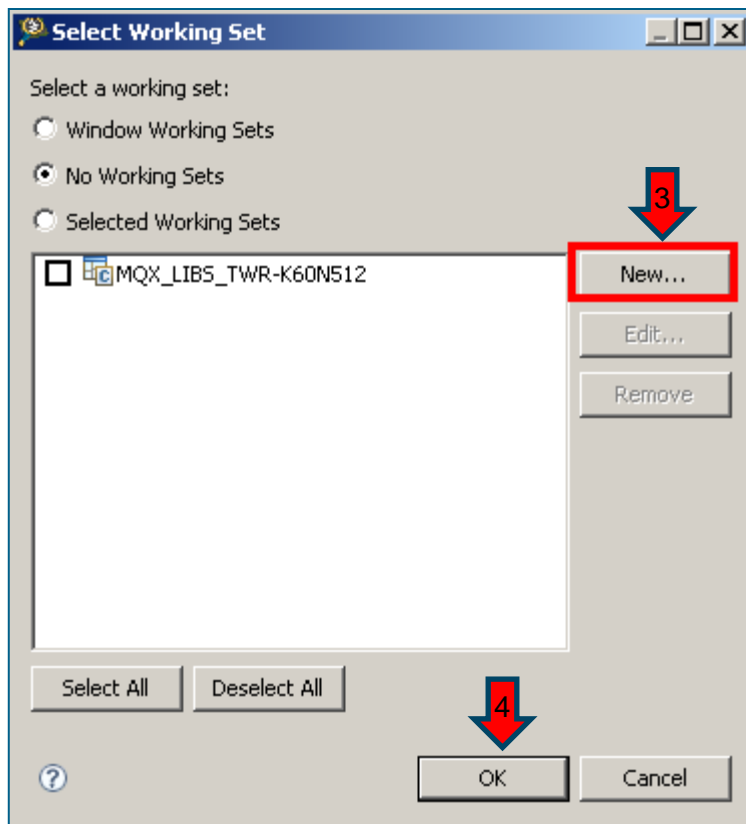
- ▶ Give a Name : **MQX_LIBS_TWR-K60N512**
- ▶ Select Projects to add and Finish



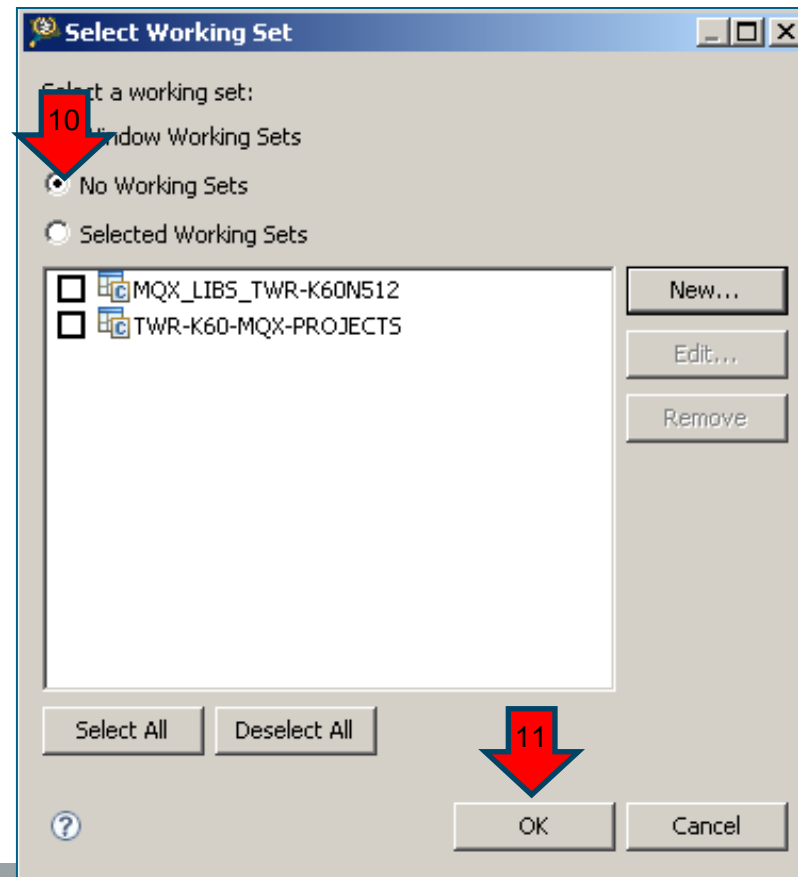
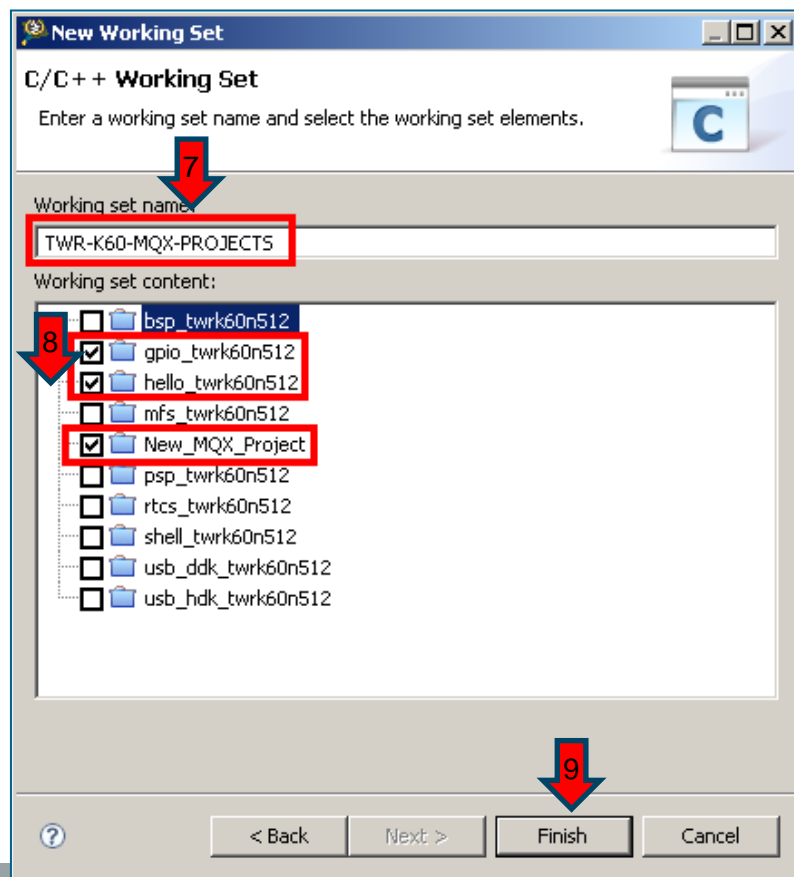
- ▶ Click in the CodeWarrior Projects view or Project Explorer view toolbar
- ▶ Select Working Set...



- ▶ Click New and OK
- ▶ Select C/C++ and Next

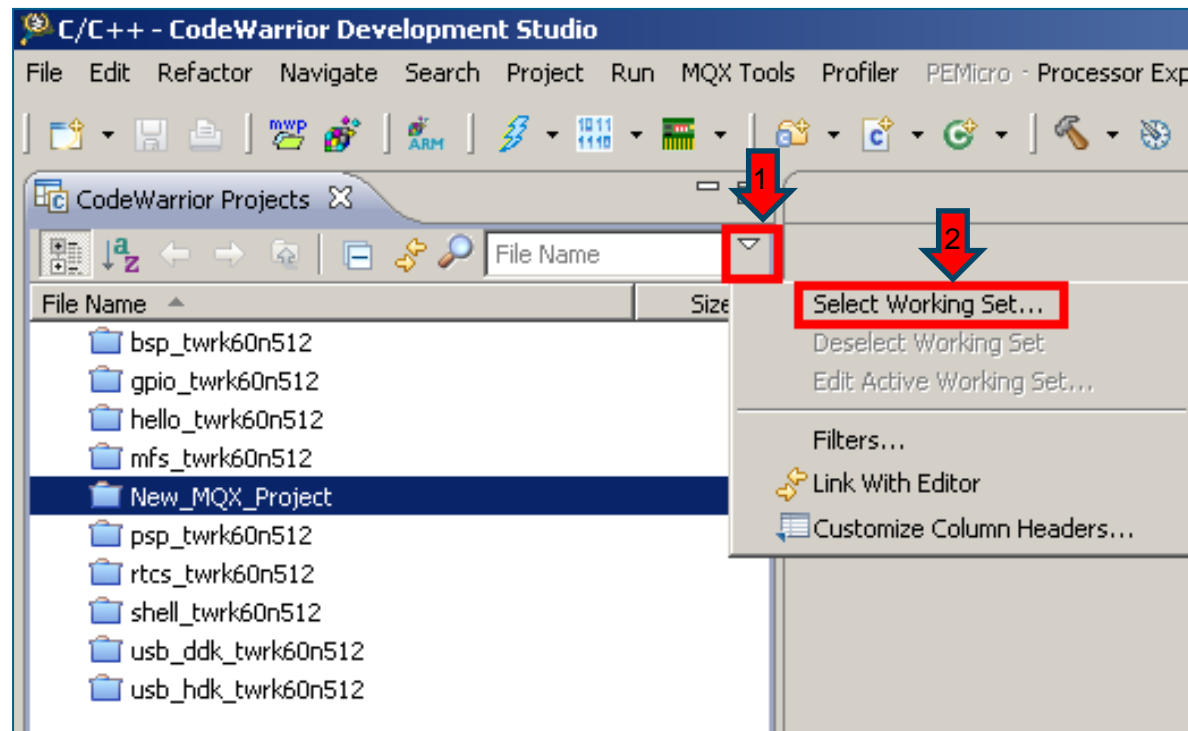


- ▶ Give a Name : **TWR-K60-MQX-PROJECTS**
- ▶ Select Projects to add and Finish



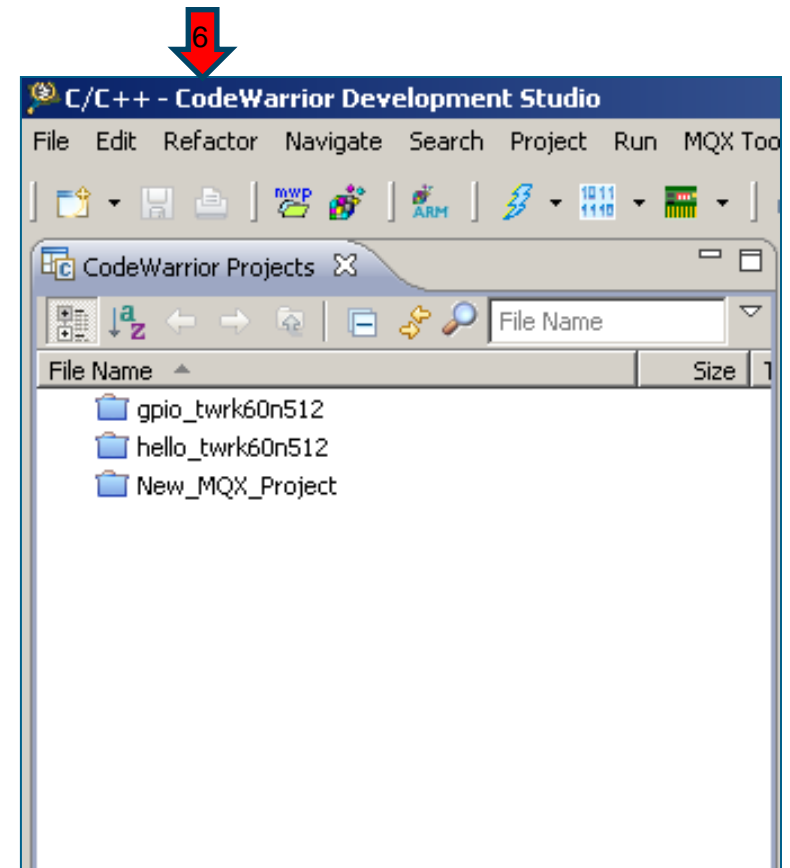
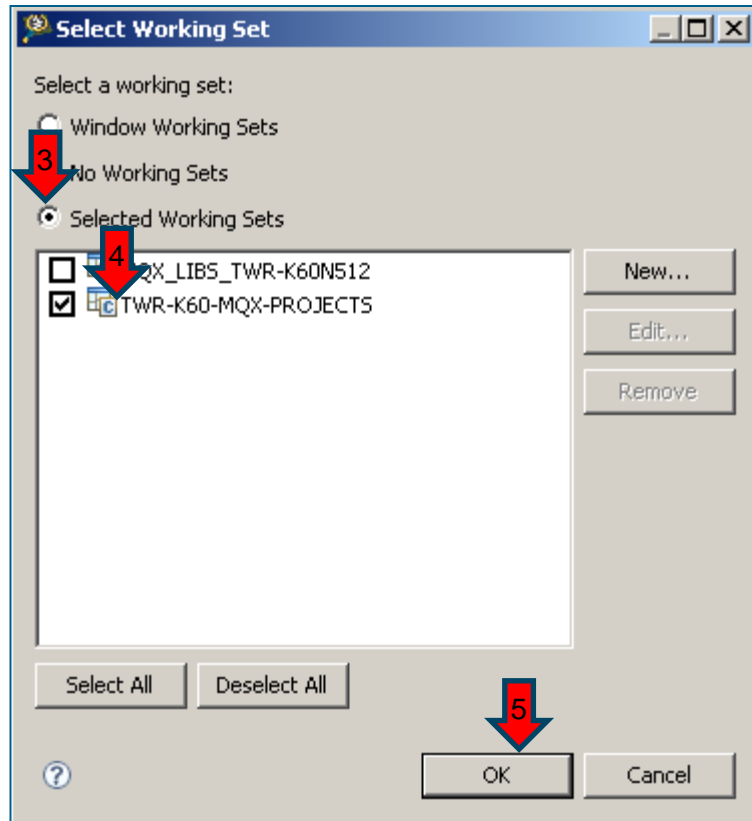
Switch Working Set

- ▶ Click in the CodeWarrior Projects view or Project Explorer view toolbar
- ▶ Select Working Set...



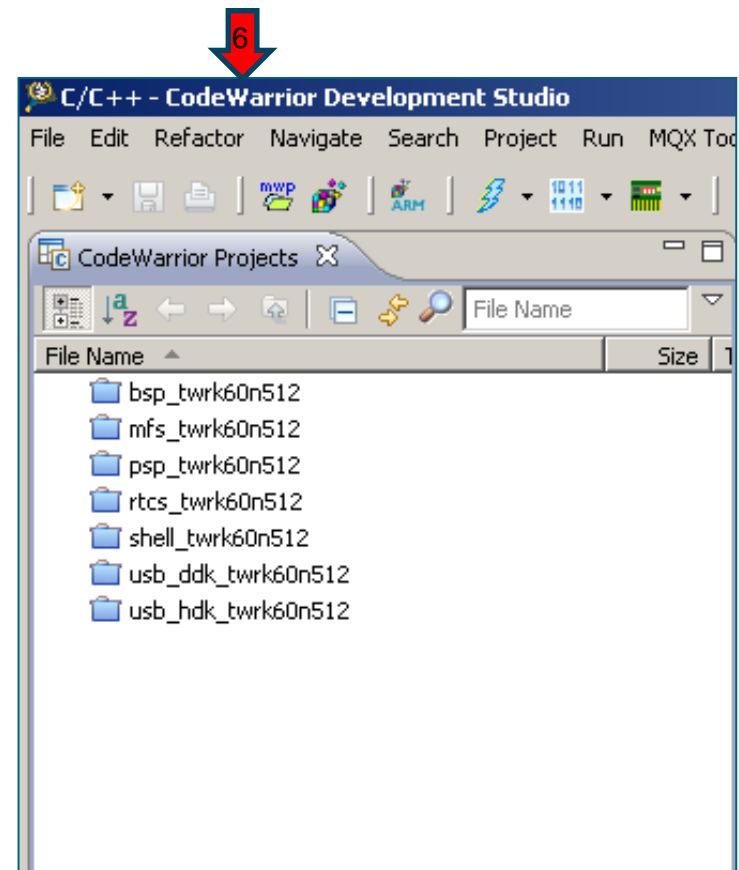
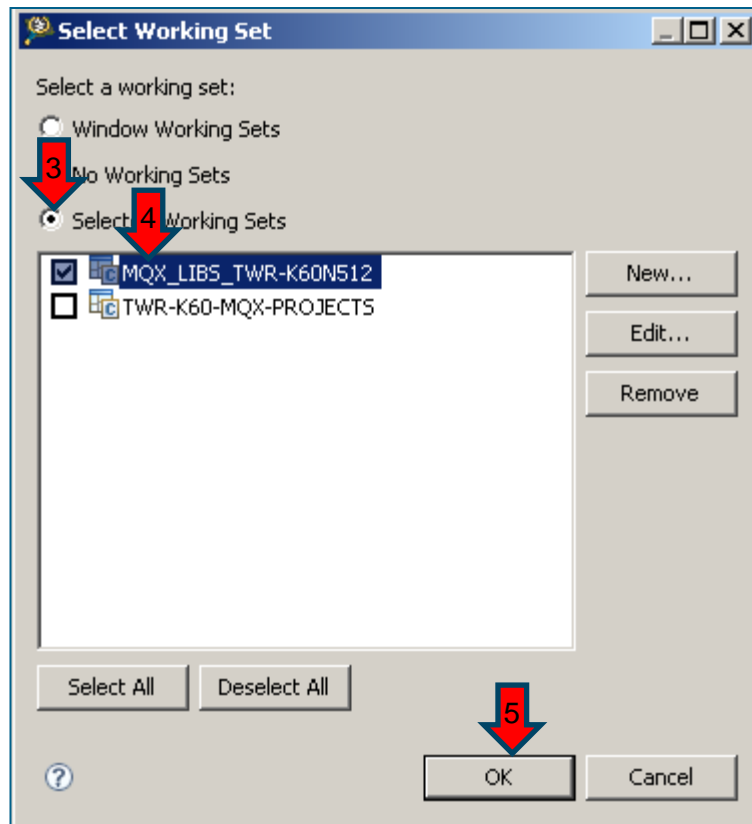
Switch Working Set

► Select **TWR-K60-MQX-PROJECTS**



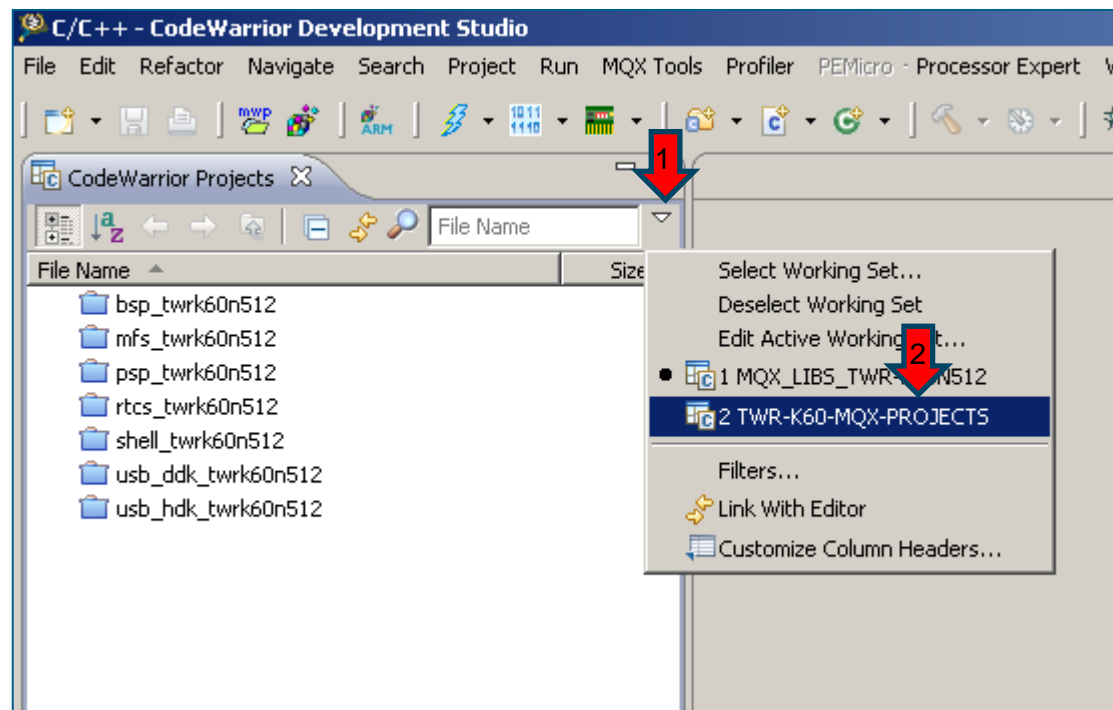
Switch Working Set

► Or Select **MQX_LIBS_TWR-K60N512**

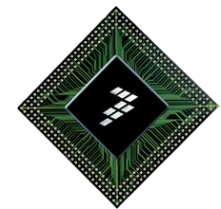


Switch Working Set

- ▶ You can switch between your Working Sets
- ▶ Click in the CodeWarrior Projects view or Project Explorer view toolbar
- ▶ Select Working Set



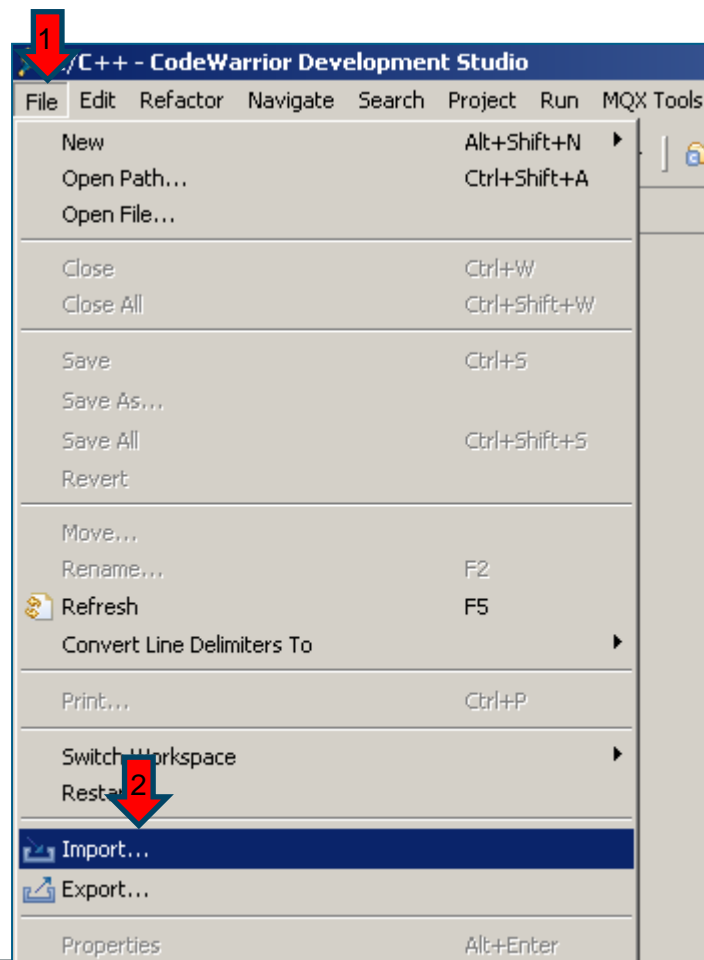
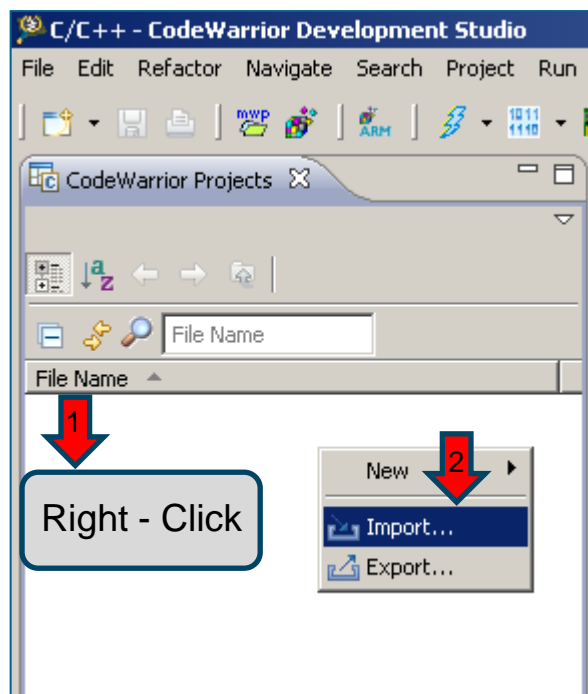
CW10.1, MQX 3.7 and Processor Expert



- ▶ Kinetis BSP projects are CodeWarrior 10.1 Processor Expert Ready
- ▶ Processor Expert drivers are enabled in MQX RTOS environment
- ▶ Two BSPs with Processor Expert drivers prepared
 - `<mqx>/build/twrk40_pe`
 - and
 - `<mqx>/build/twrk60_pe`

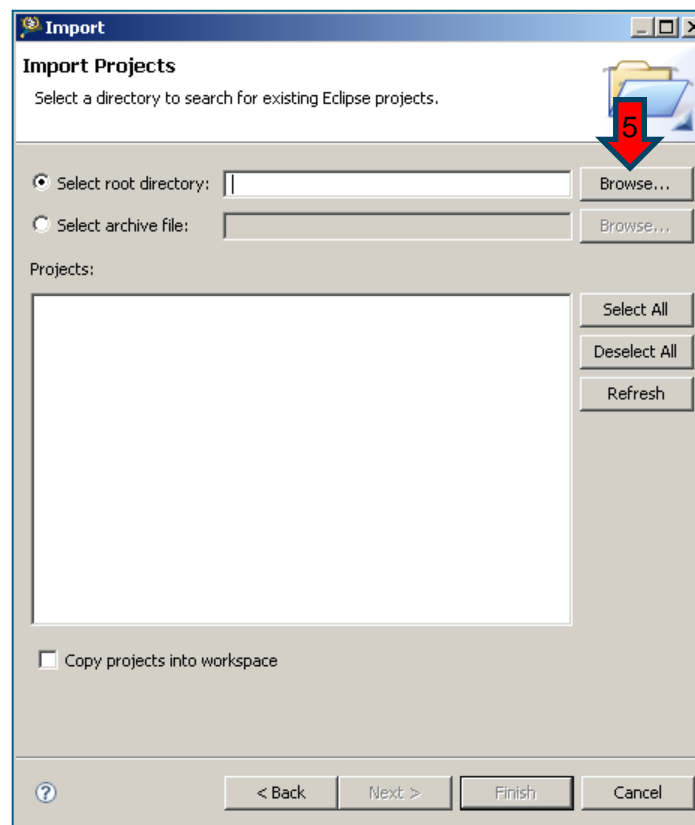
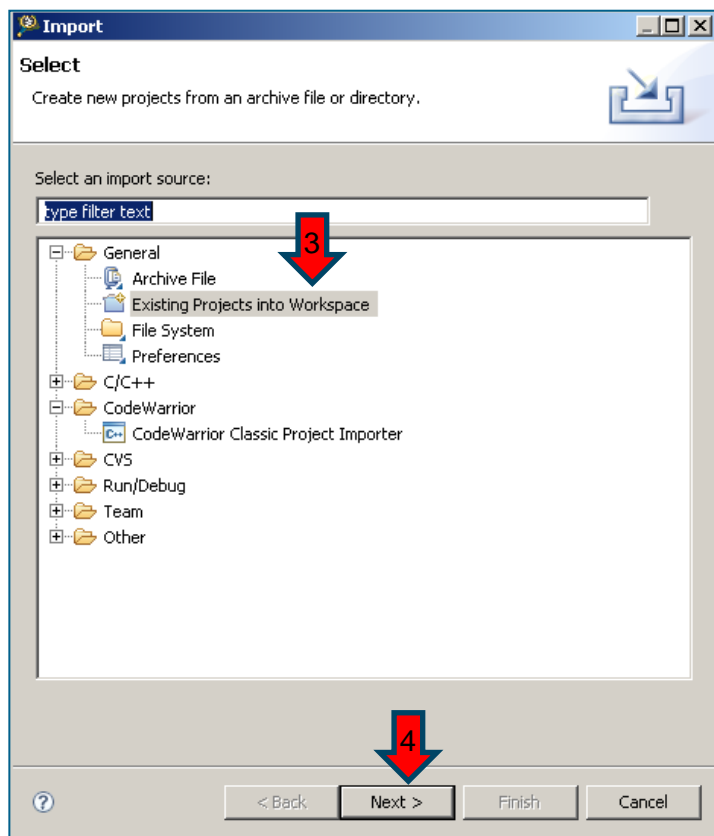
Import MQX PE Ready BSP

- Right-Click on Project Explorer and Import (or) File -> Import



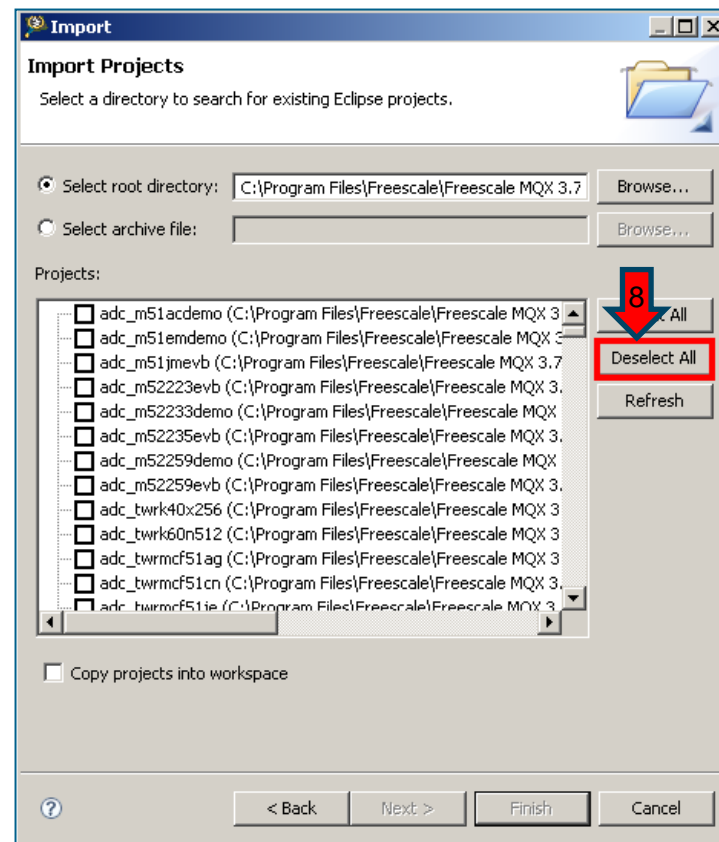
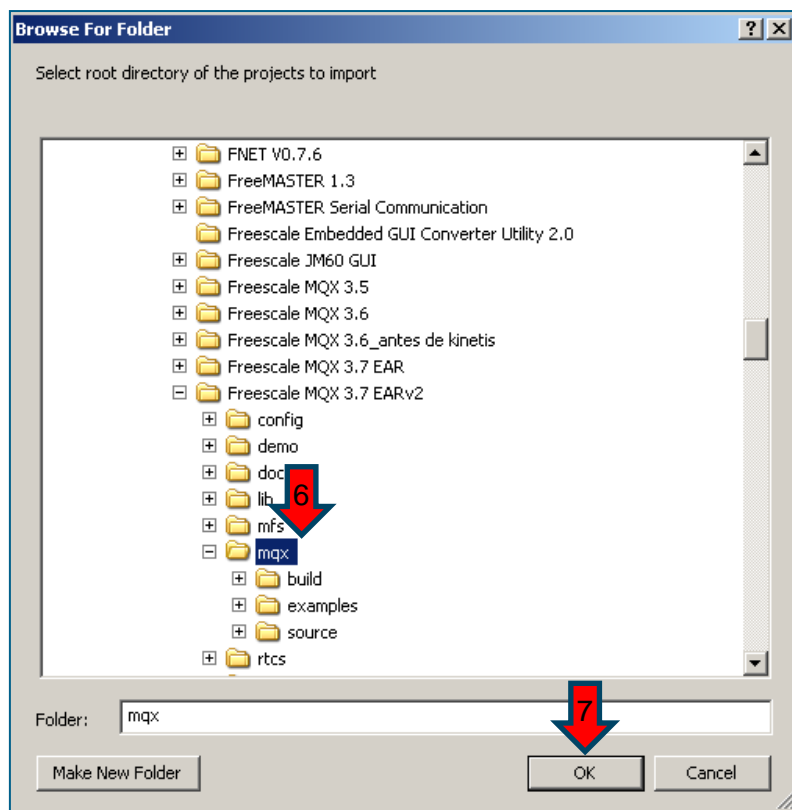
Import MQX PE Ready BSP

► Select Existing Projects into Workspace and Browse



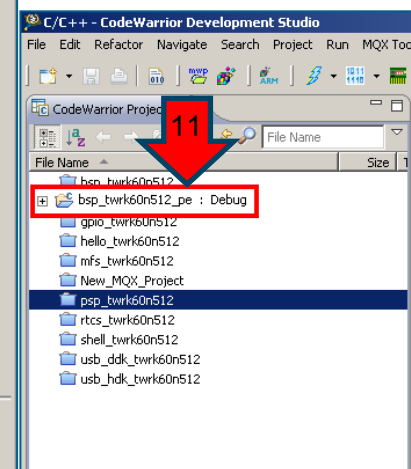
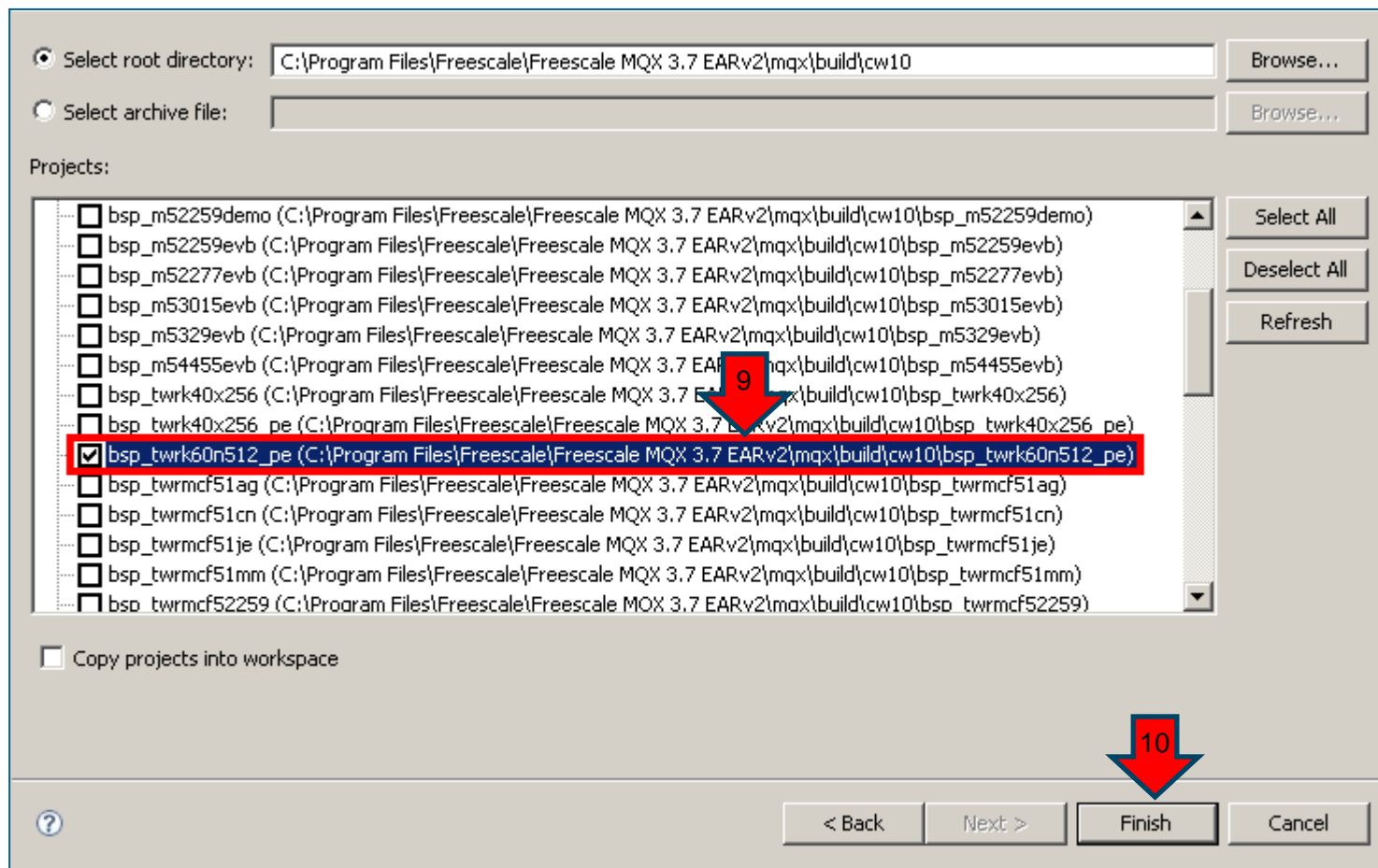
Import MQX PE Ready BSP

► Select *<install mqx folder>mqx*



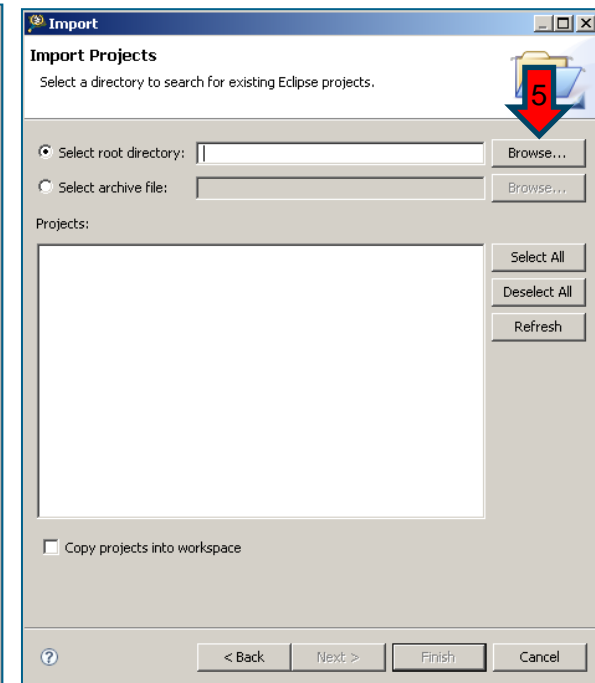
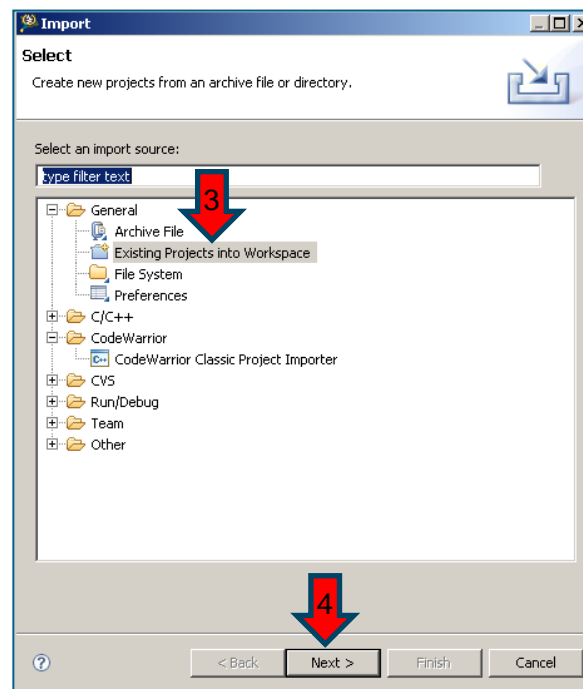
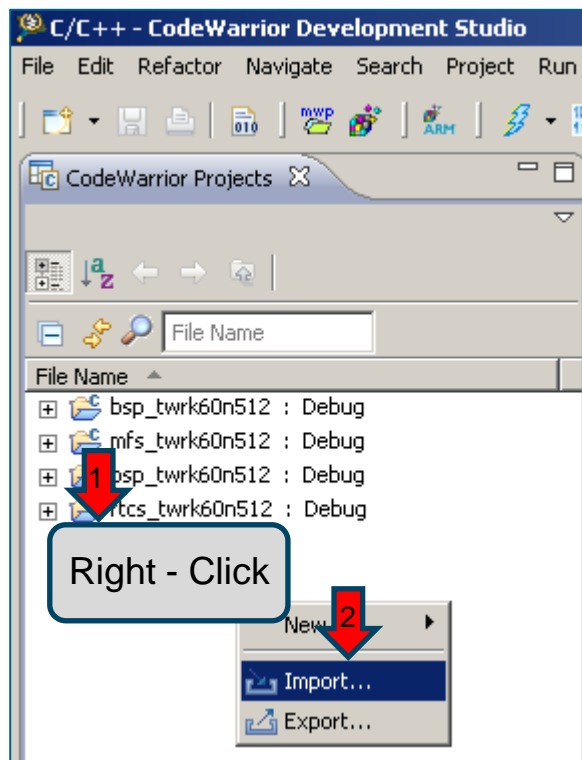
Import MQX PE Ready BSP

► Select **bsp_twrk60n512_pe** project



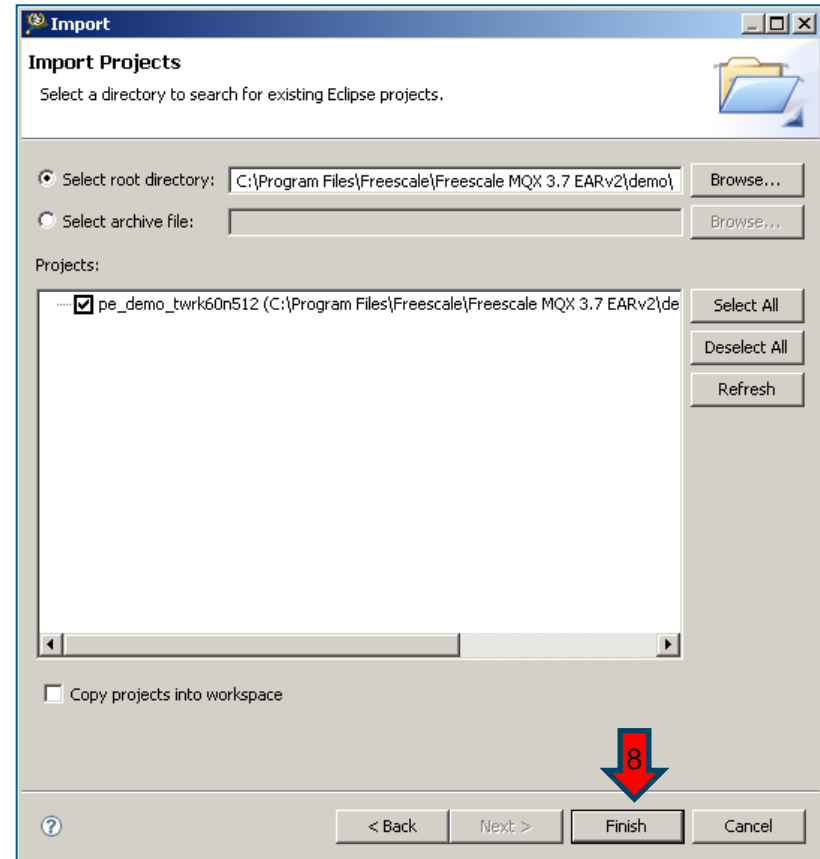
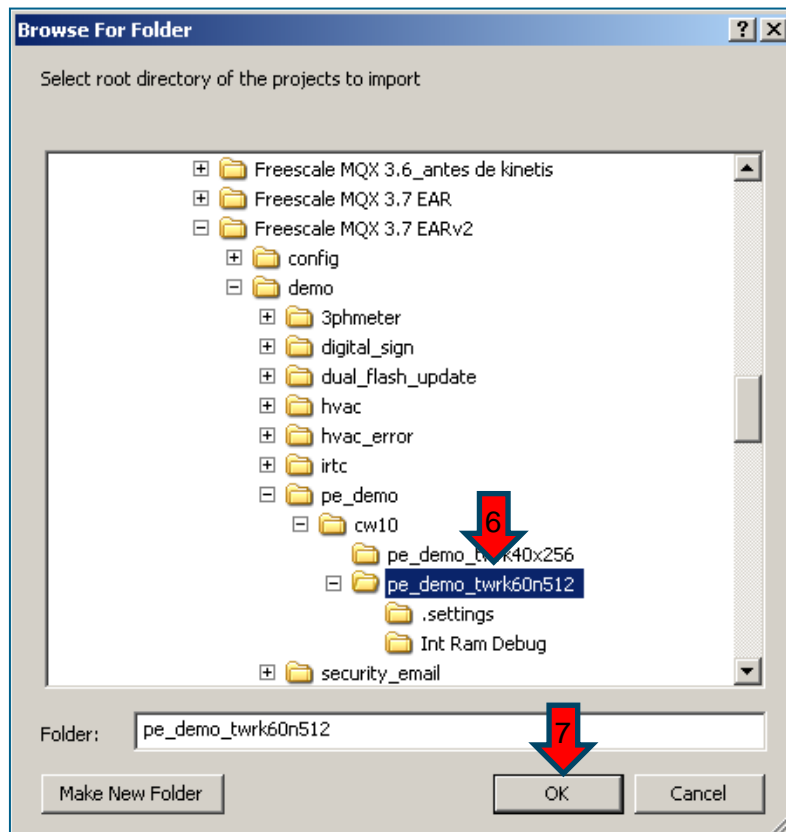
Import MQX PE Demo

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



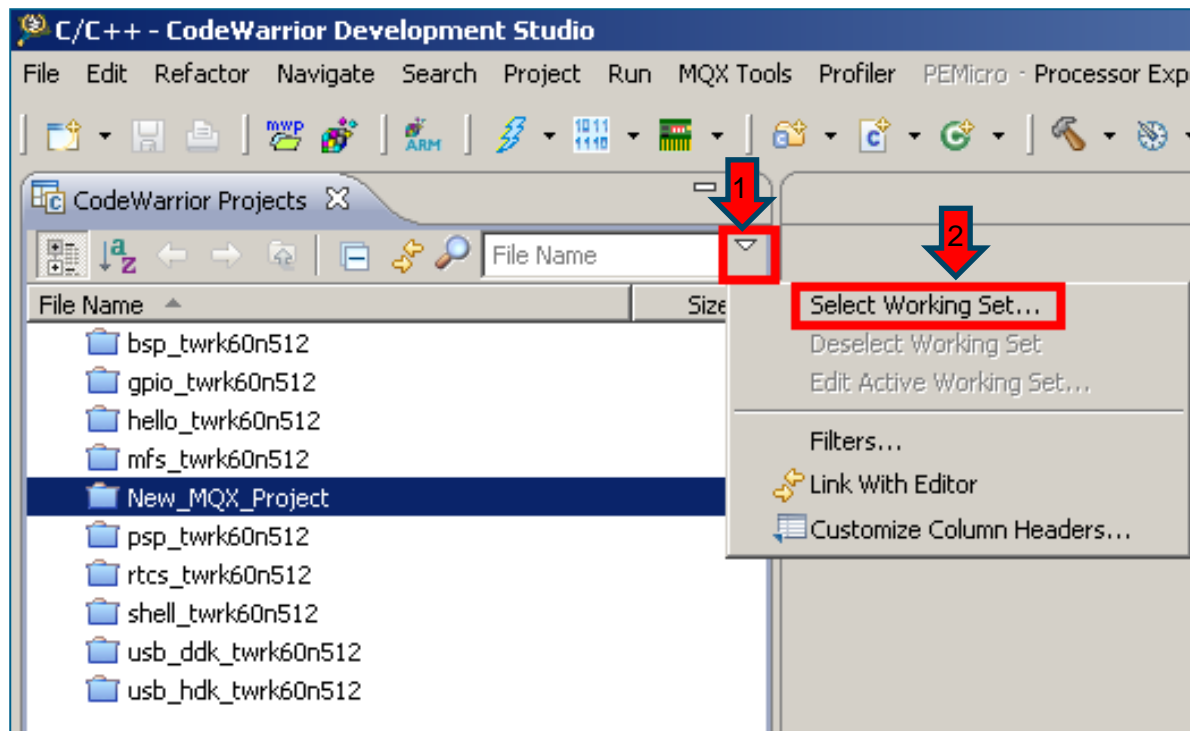
Import MQX PE Demo

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

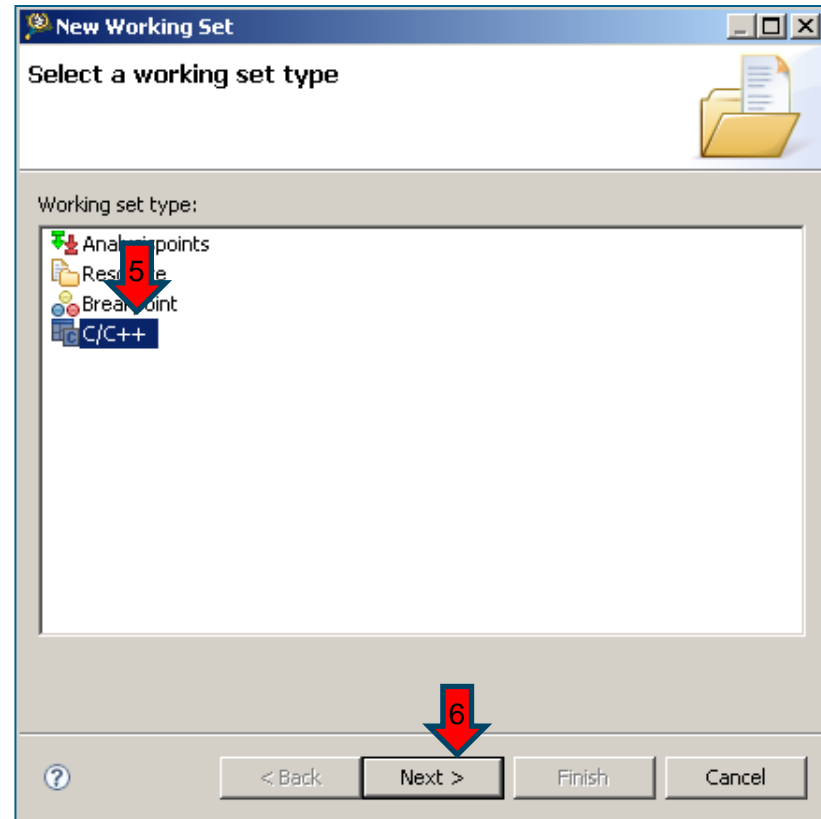
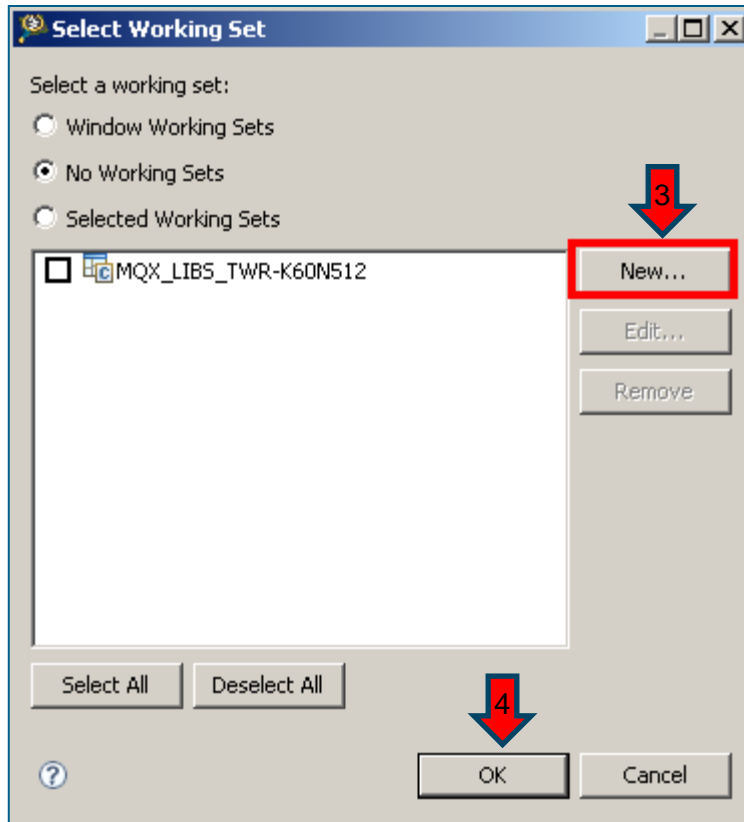


MQX PE Working Set

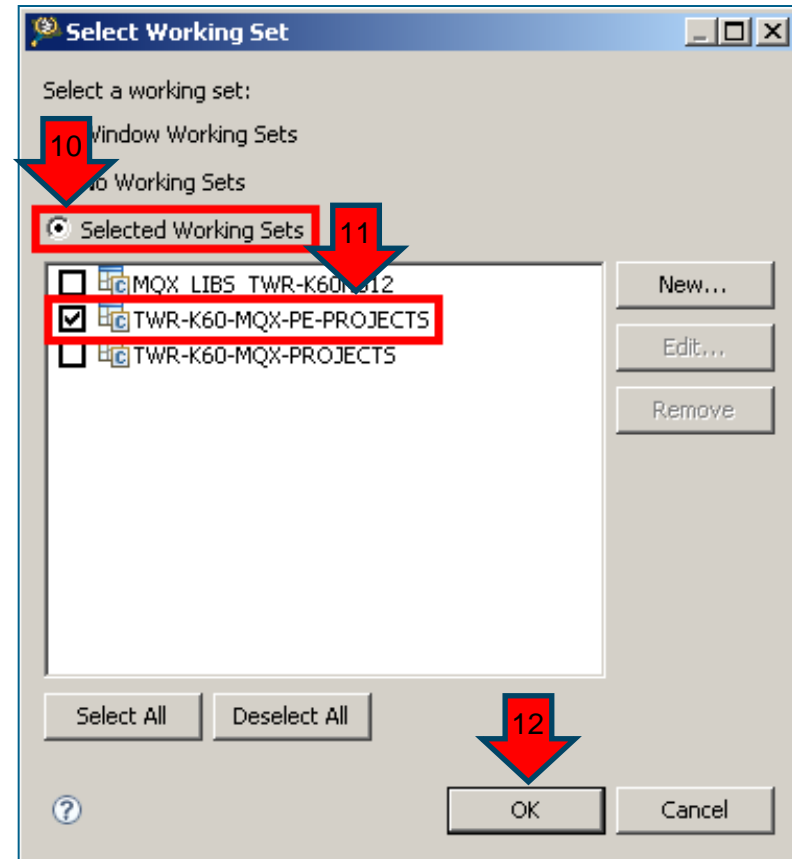
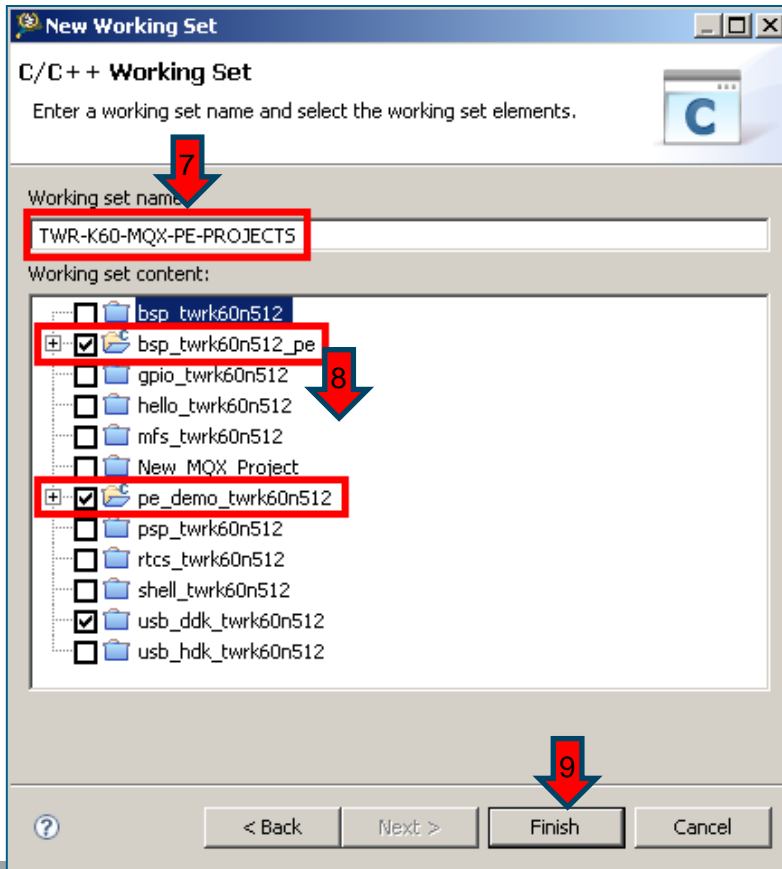
- ▶ Click in the CodeWarrior Projects view or Project Explorer view toolbar
- ▶ Select Working Set...



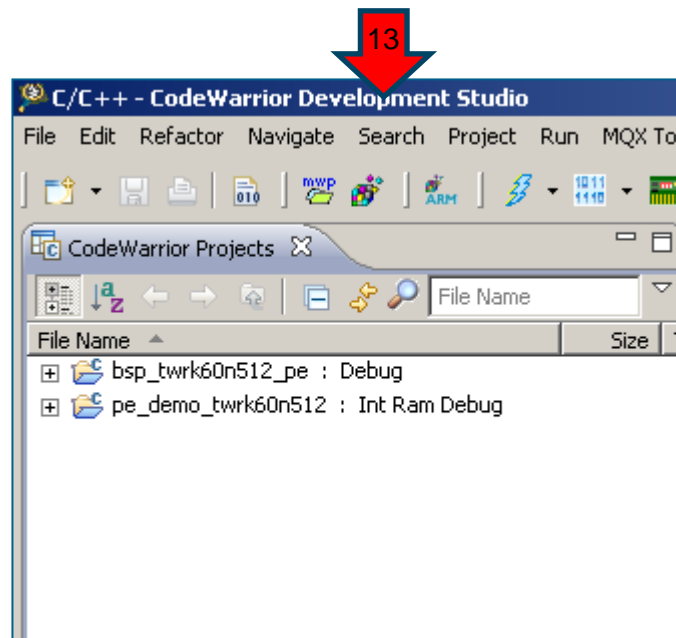
- ▶ Click New and OK
- ▶ Select C/C++ and Next



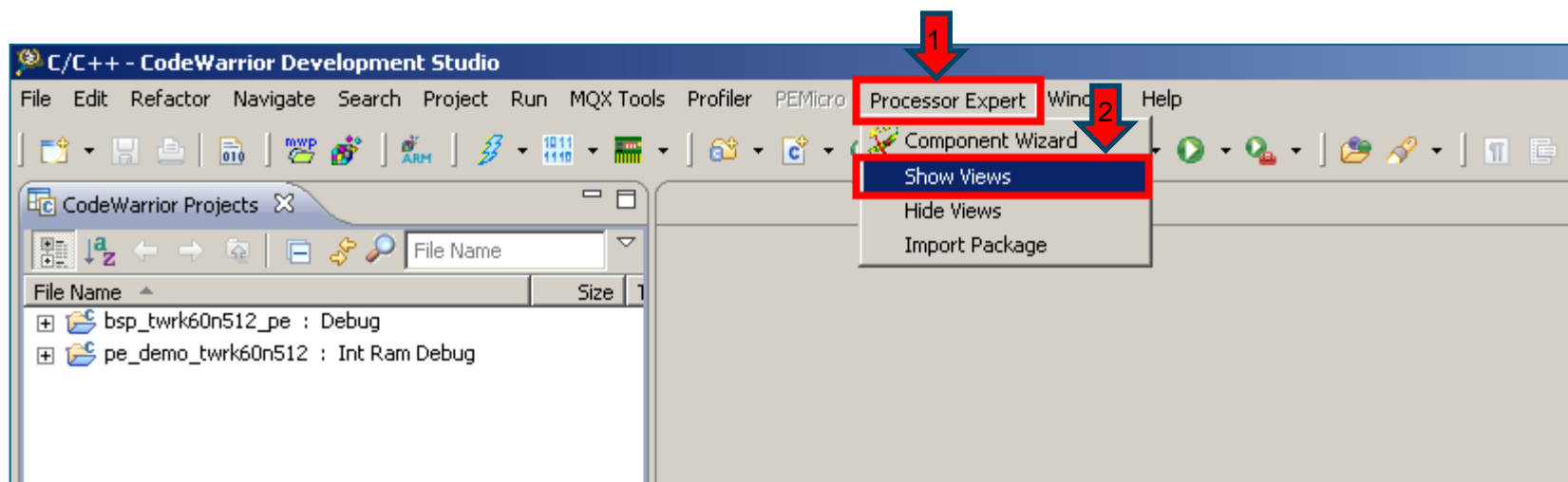
- ▶ Give a Name : **TWR-K60-MQX-PE-PROJECTS**
- ▶ Select Projects to add and Finish



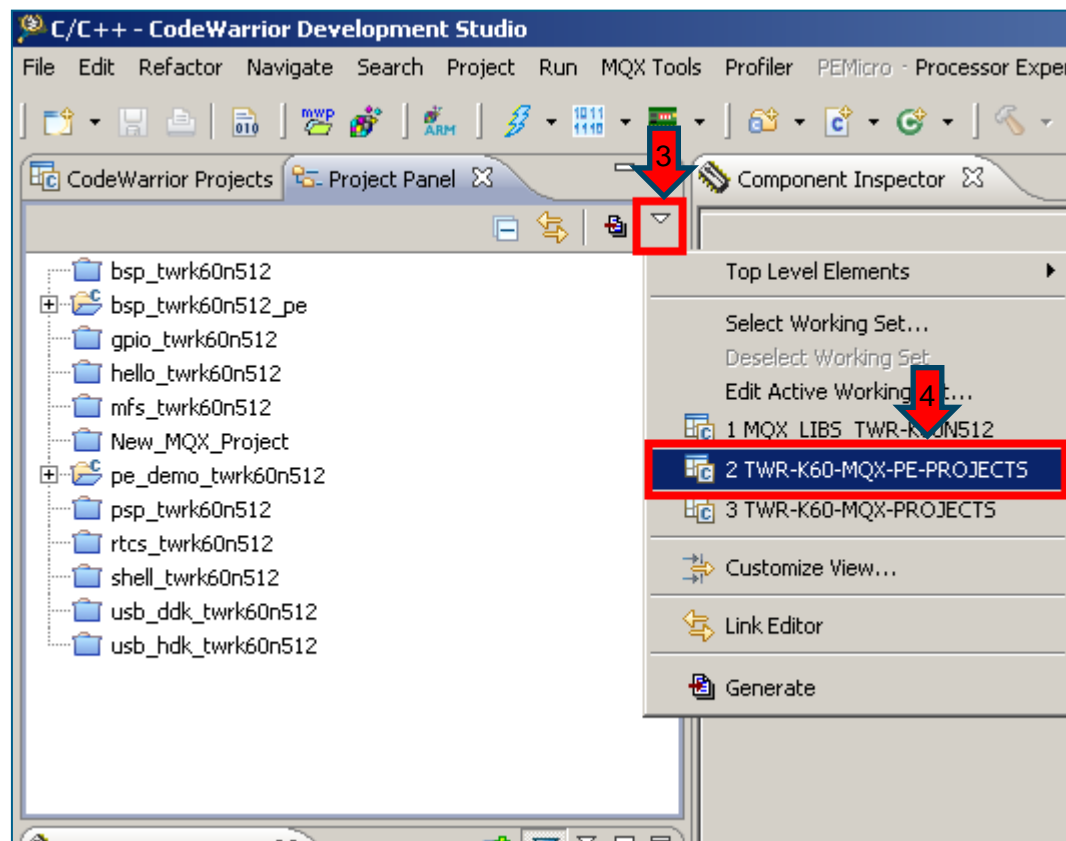
- We have PE BSP and PE demo in our Working Set



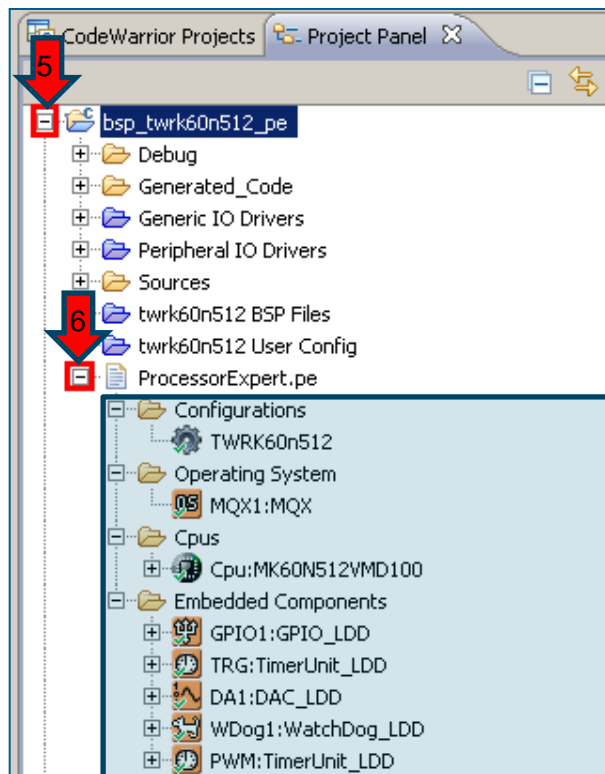
► Select in Menu : Processor Expert -> Show Views



- Select PE working Set to hide the rest of projects



- Expand **bsp_twrk60n512_pe** project view



Processor Expert Drivers

- Click on PE components to watch the properties

7 CodeWarrior Projects Project Panel

8 bsp_twrk60n512_pe

- Debug
- Generated_Code
- Generic IO Drivers
- Peripheral IO Drivers
- Sources
- twrk60n512 BSP Files
- twrk60n512 User Config
- ProcessorExpert
- Configuration
 - TWRK60N512
 - Operating System
 - MQX1:MQX
 - Cpus
 - Cpu:MK60N512VMD100
 - Embedded Components
 - GPIO1:GPIO_LDD
 - TRG:TimerUnit_LDD
 - DA1:DAC_LDD
 - WDog1:WatchDog_LDD
 - PWM:TimerUnit_LDD

9 *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

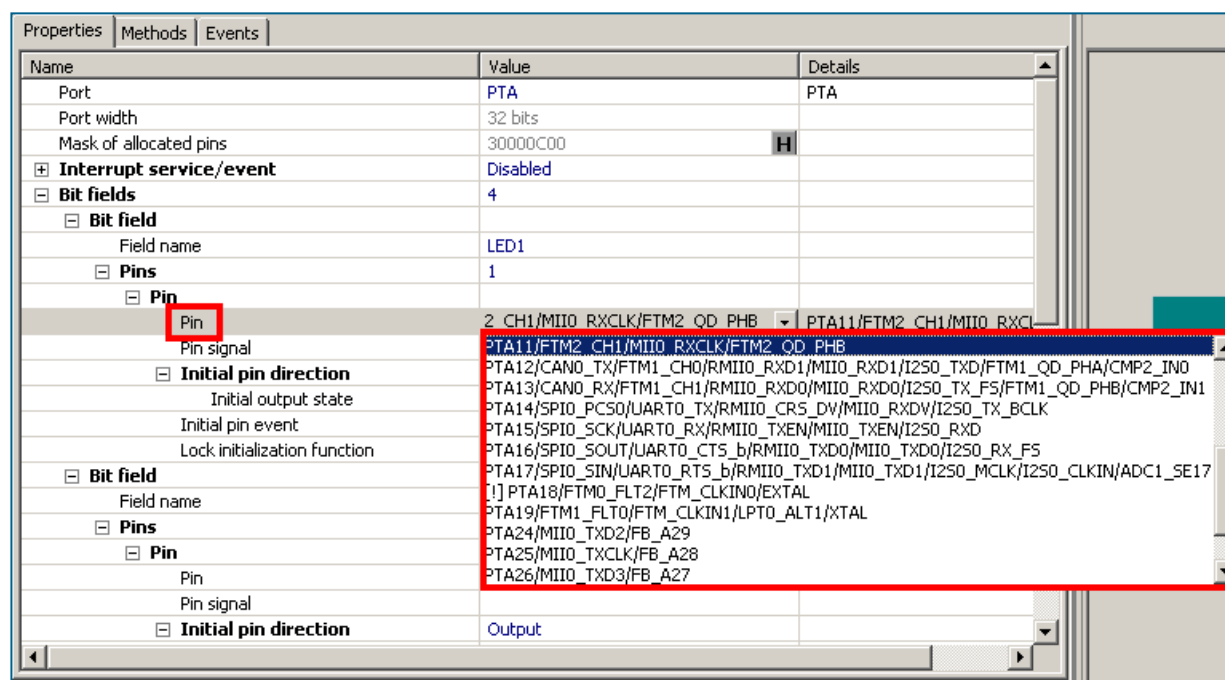
Properties Methods Events

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	

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



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

*Component Inspector

Basic Advanced

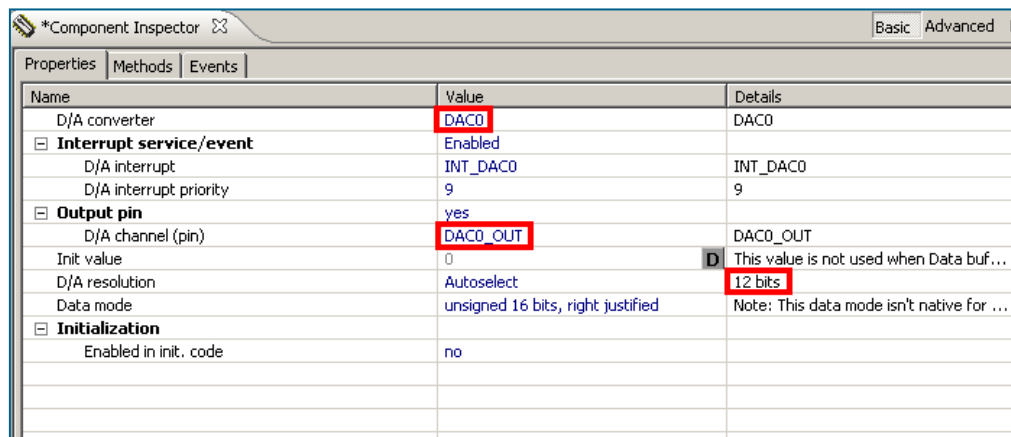
Properties Methods Events

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		

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

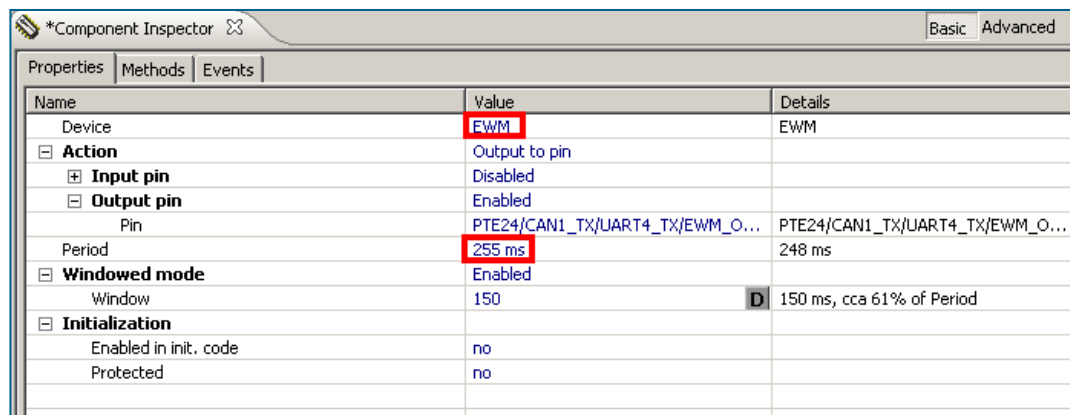
Properties Methods Events			
Name	Value	Details	
Module name	FTM0	FTM0	
Counter	FTM0_CNT	FTM0_CNT	
Counter direction	Up		
Counter width	16 bits		
Value type	uint16_t	uint16_t	
<input type="checkbox"/> Input clock source	Internal		
Counter frequency	48 MHz	48 MHz	
<input type="checkbox"/> Counter restart	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			

► DAC0 configures DAC0 with resolution of 12 bits



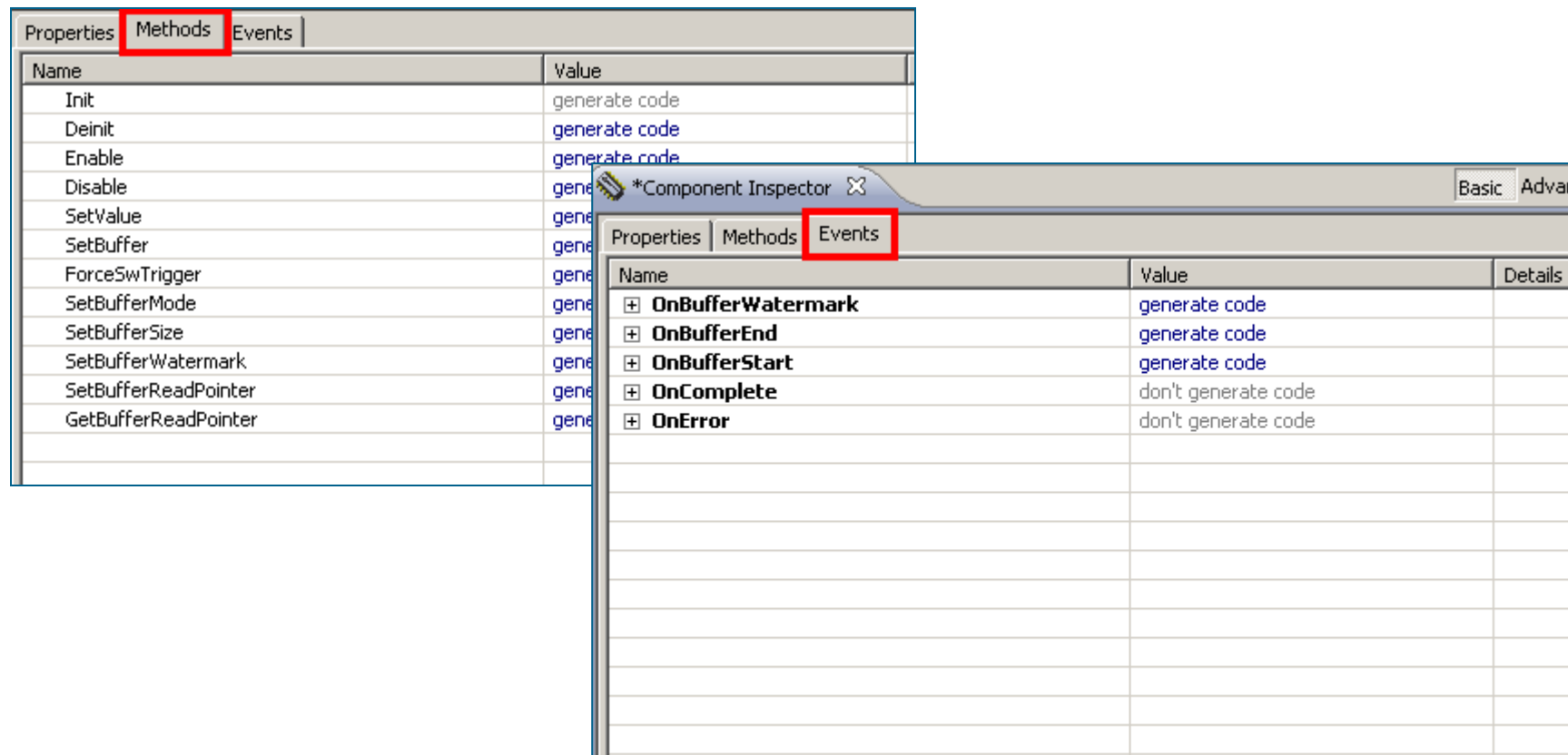
Name	Value	Details
D/A converter	DAC0	DAC0
Interrupt service/event	Enabled	
D/A interrupt	INT_DAC0	INT_DAC0
D/A interrupt priority	9	9
Output pin	yes	
D/A channel (pin)	DAC0_OUT	DAC0_OUT
Init value	0	D This value is not used when Data buf...
D/A resolution	Autoselect	12 bits
Data mode	unsigned 16 bits, right justified	Note: This data mode isn't native for ...
Initialization		
Enabled in init. code	no	

► WDog1 enables EWM module with period of 255 ms



Name	Value	Details
Device	EWM	EWM
Action	Output to pin	
Input pin	Disabled	
Output pin	Enabled	
Pin	PTE24/CAN1_TX/UART4_TX/EWM_O...	PTE24/CAN1_TX/UART4_TX/EWM_O...
Period	255 ms	248 ms
Windowed mode	Enabled	
Window	150	D 150 ms, cca 61% of Period
Initialization		
Enabled in init. code	no	
Protected	no	

- Components has not only Properties
- It includes **Methods** and **Events** that we can enable or disable

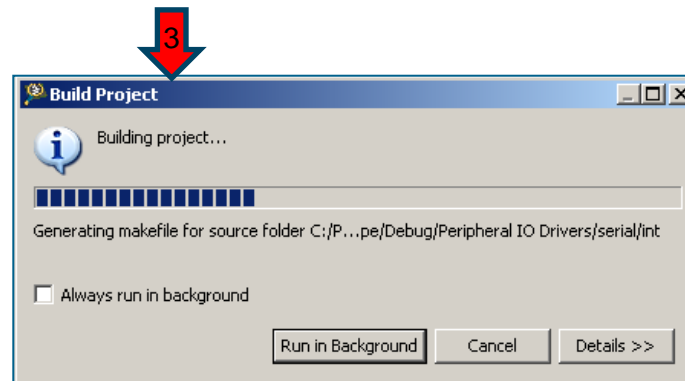
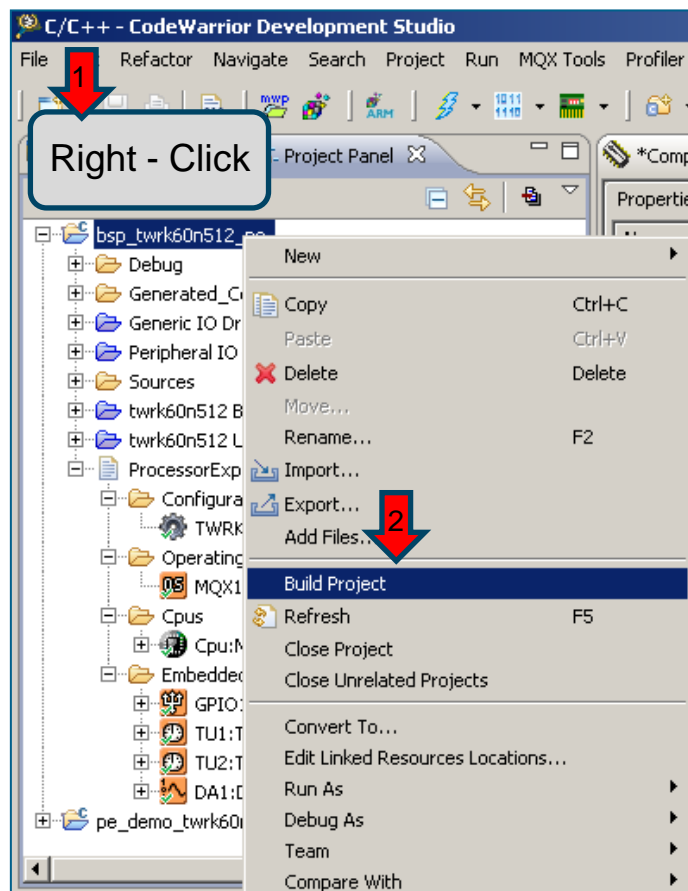


The screenshot displays two windows from the MQX PE demo. The main window shows the 'Properties' tab, with the 'Methods' tab highlighted by a red box. The 'Methods' tab lists various methods such as 'Init', 'Deinit', 'Enable', 'Disable', 'SetValue', 'SetBuffer', 'ForceSwTrigger', 'SetBufferMode', 'SetBufferSize', 'SetBufferWatermark', 'SetBufferReadPointer', and 'GetBufferReadPointer'. The 'Component Inspector' window, which is a smaller dialog box, shows the 'Events' tab highlighted by a red box. This window lists events: 'OnBufferWatermark', 'OnBufferEnd', 'OnBufferStart', 'OnComplete', and 'OnError'. Each event has a 'Value' column indicating whether to 'generate code' or 'don't generate code'.

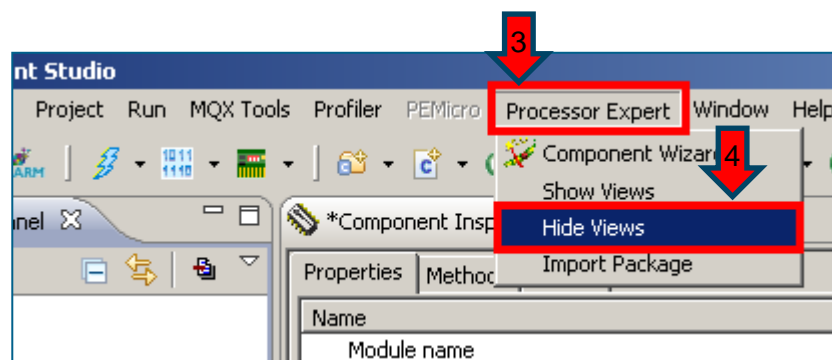
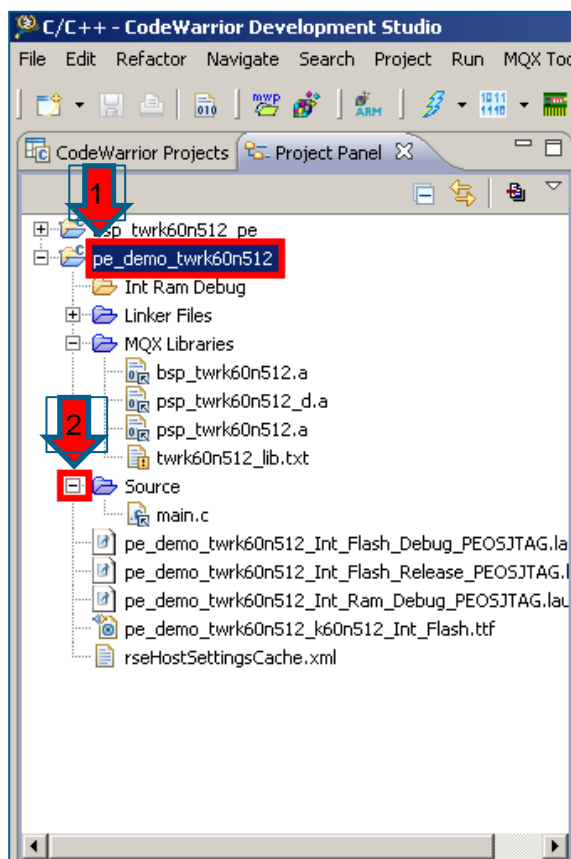
Name	Value
Init	generate code
Deinit	generate code
Enable	generate code
Disable	generate code
SetValue	generate code
SetBuffer	generate code
ForceSwTrigger	generate code
SetBufferMode	generate code
SetBufferSize	generate code
SetBufferWatermark	generate code
SetBufferReadPointer	generate code
GetBufferReadPointer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

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

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



- ▶ 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:

```

/* 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");
}
```

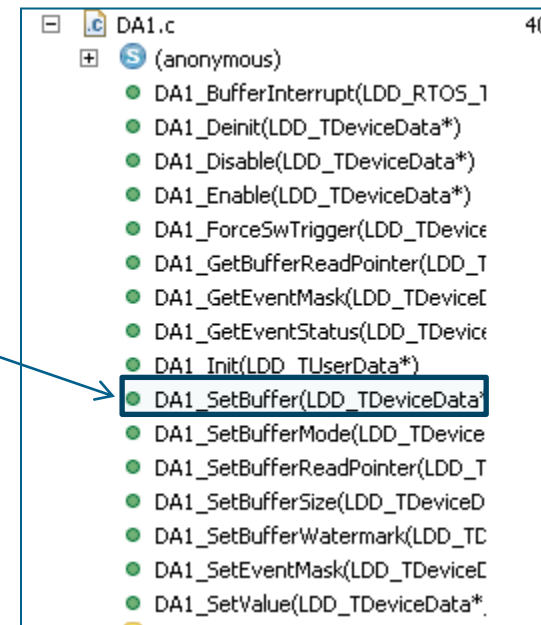
```
EWM_DeviceData = WDog1_Init(NULL);
if (EWM_DeviceData == NULL) {
    puts("failed");
    _task_block();
}
else {
    puts("done");
}
```

► 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_BufferPtr,  
                           DA1_INTERNAL_BUFFER_SIZE,  
                           0);
```



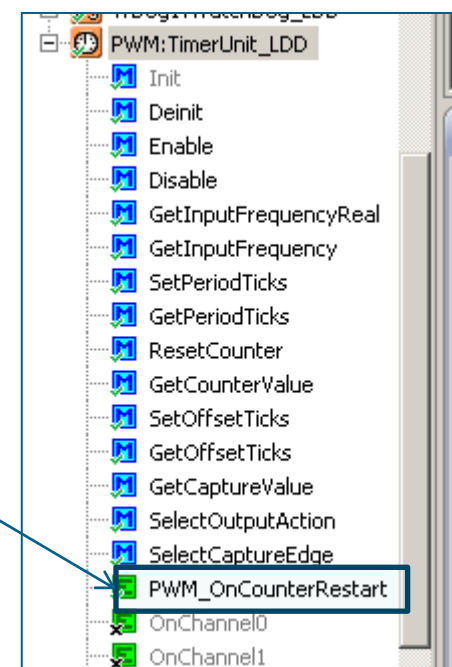
► And implement the Events

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

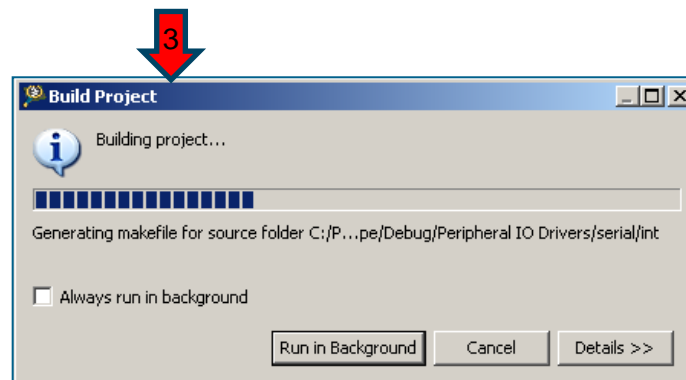
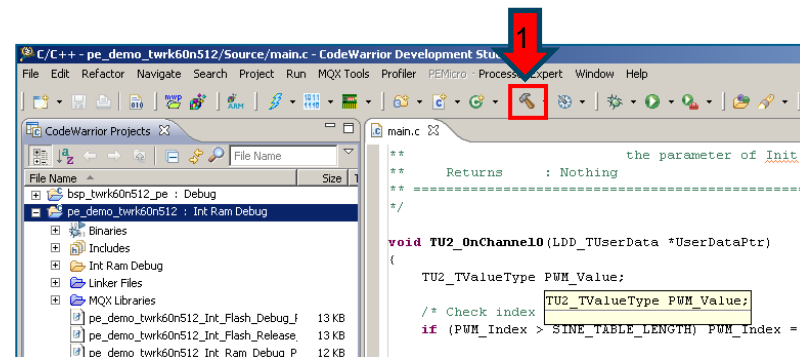
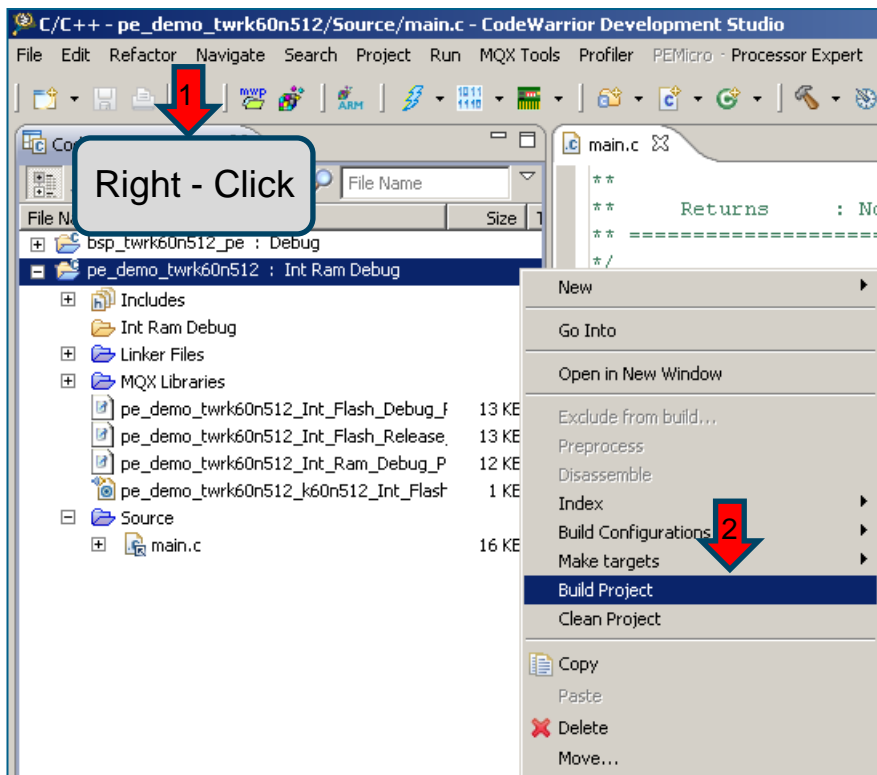
    PWM_Value += PWM_Step;

    if (PWM_Value > PWM_MaxValue) PWM_Value = 0;

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



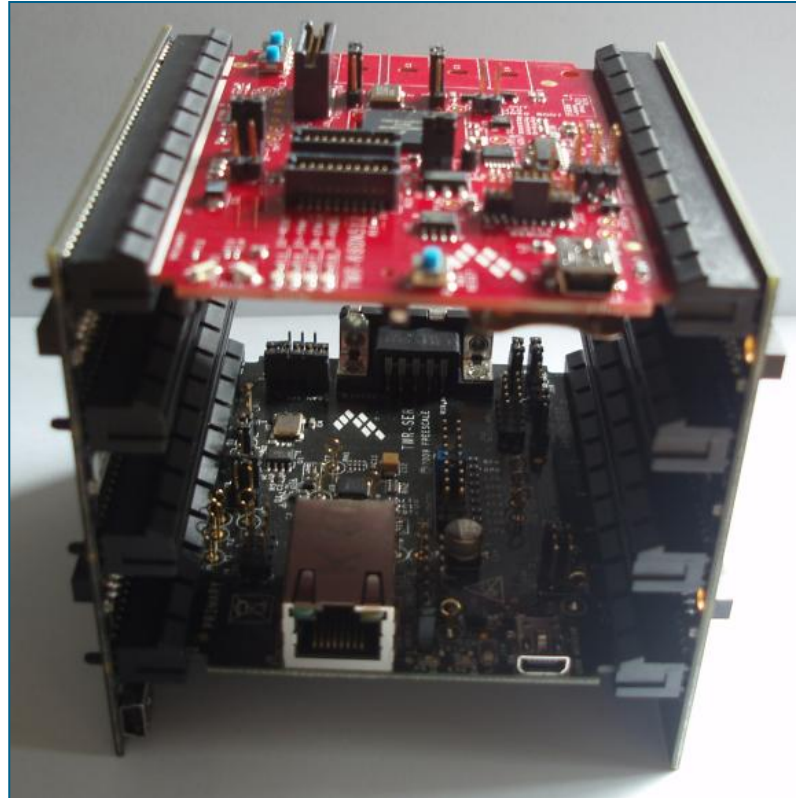
- ▶ Right-Click on Project Explorer **pe_demo_twrk60n512** and Build Project or click on icon 



Prepare your hardware

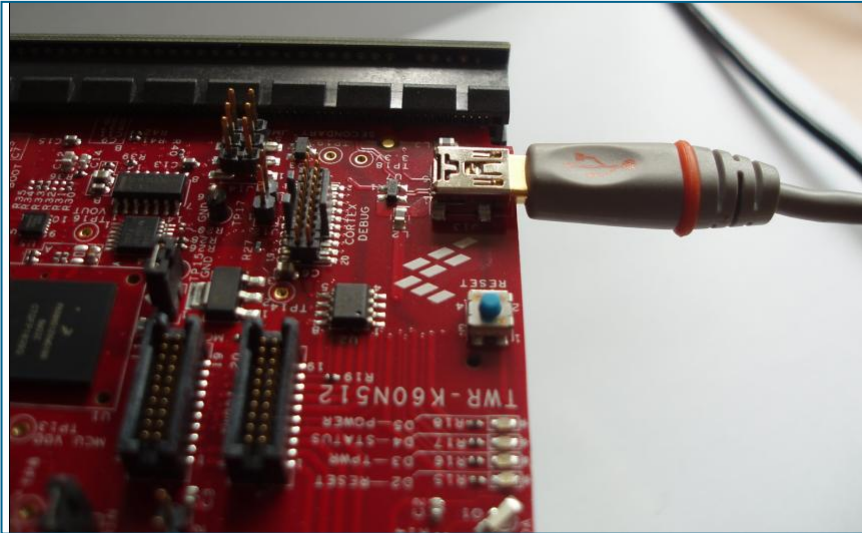
► Prepare your Tower System:

- Connect **TWR-SER** and **TWR-K60N512** to **TWR-ELEV** (Primary and Secondary)



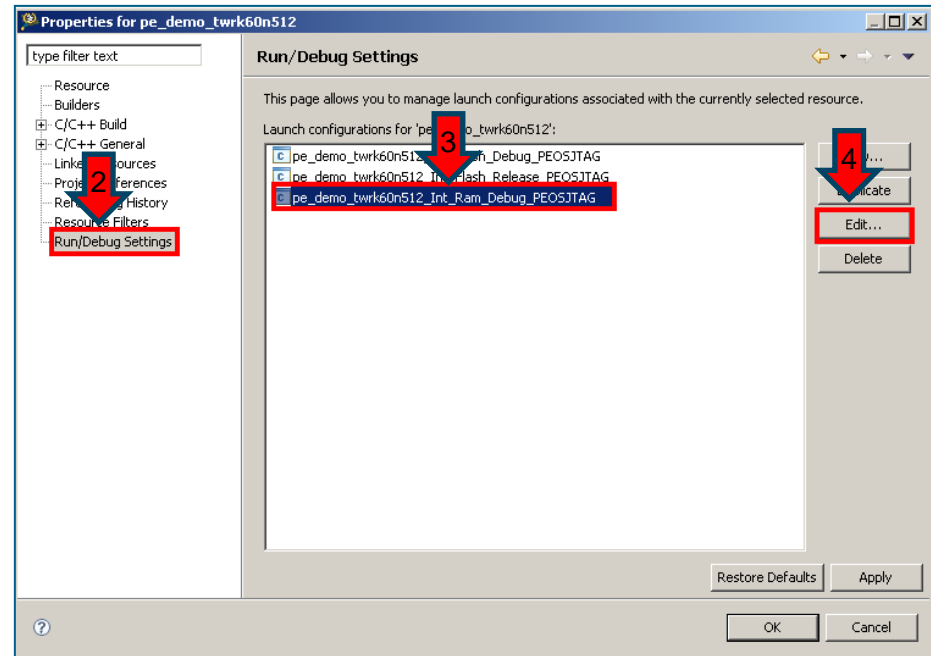
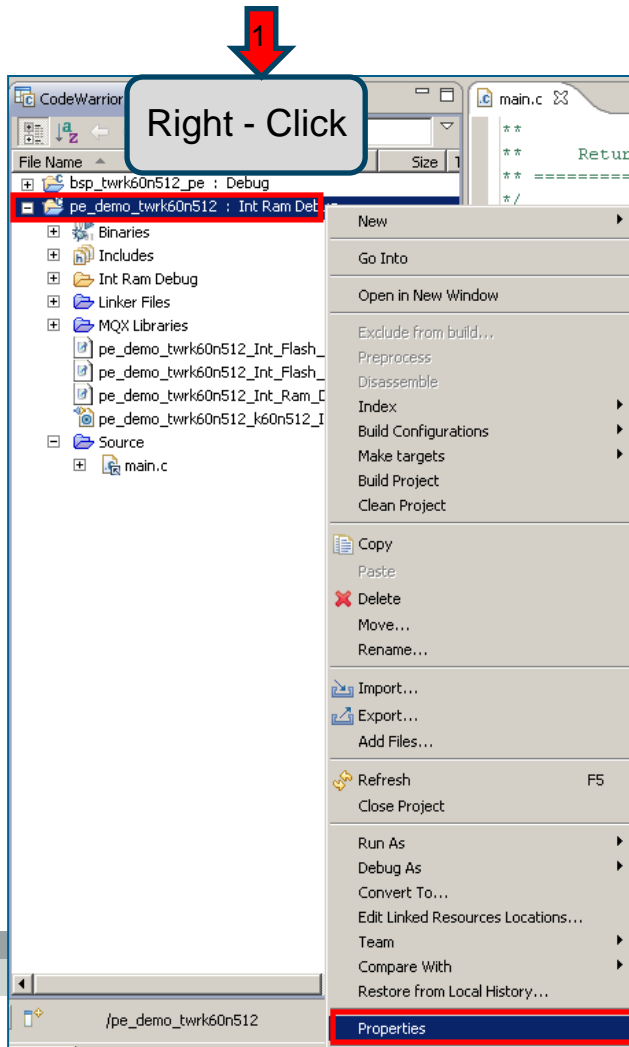
Prepare your hardware

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



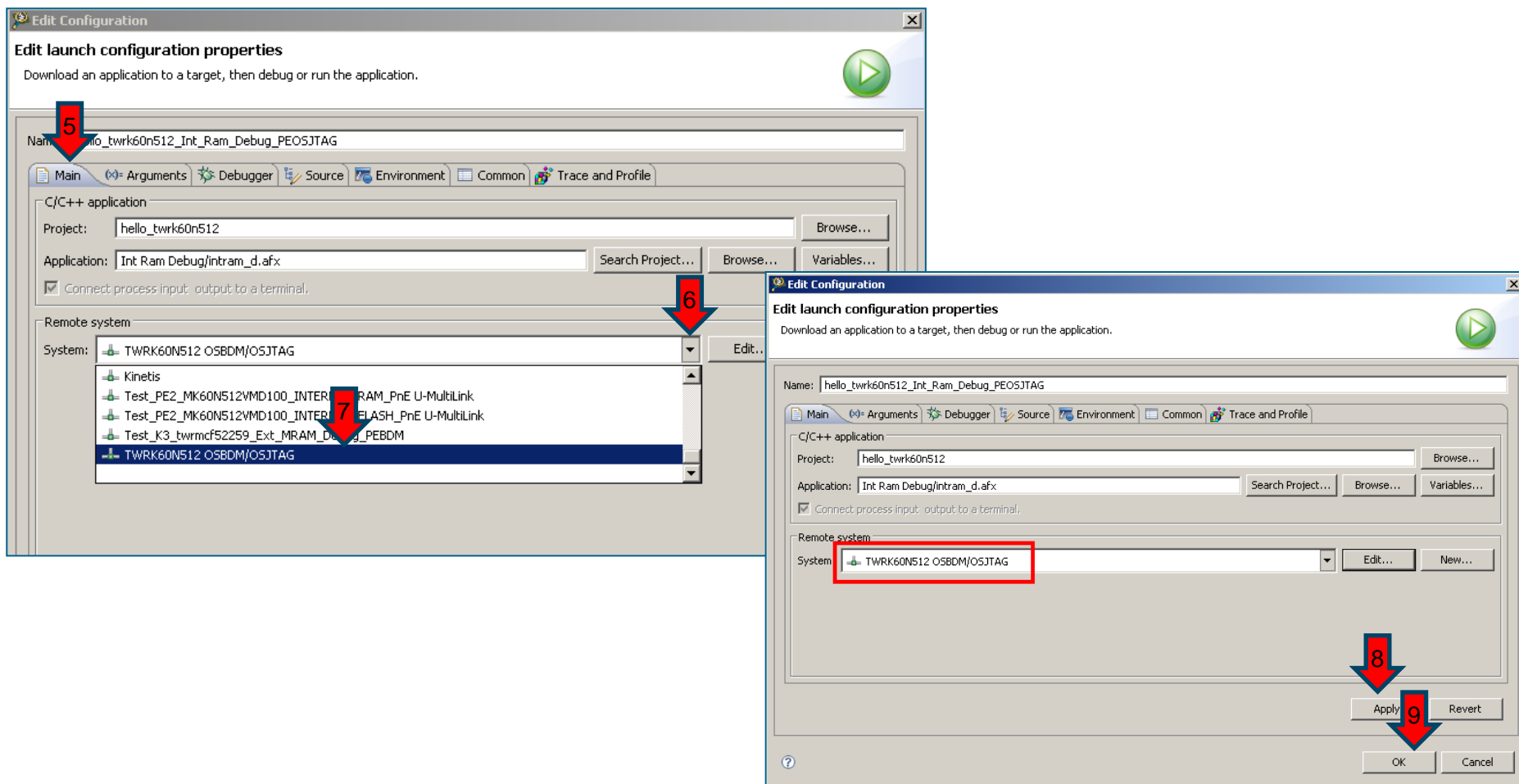
Debug Connection MQX PE Demo

- ▶ Right-Click on Project Explorer **pe_demo_twrk60n512** and Build Project
- ▶ Edit **pe_demo_twrk60n512_Int_Ram_Debug_PEOSJTAG**



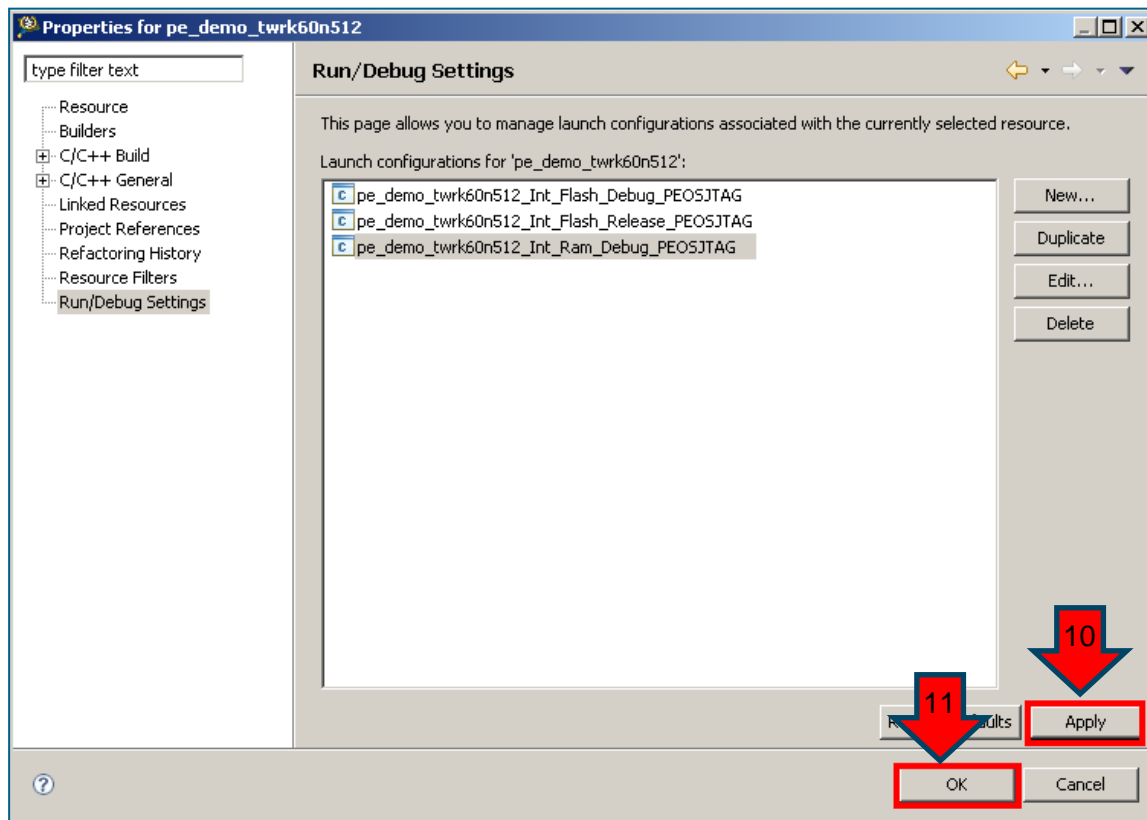
Debug Connection MQX PE Demo

► Select **TWRK60N512 OSBDM/OSJTAG** Remote System

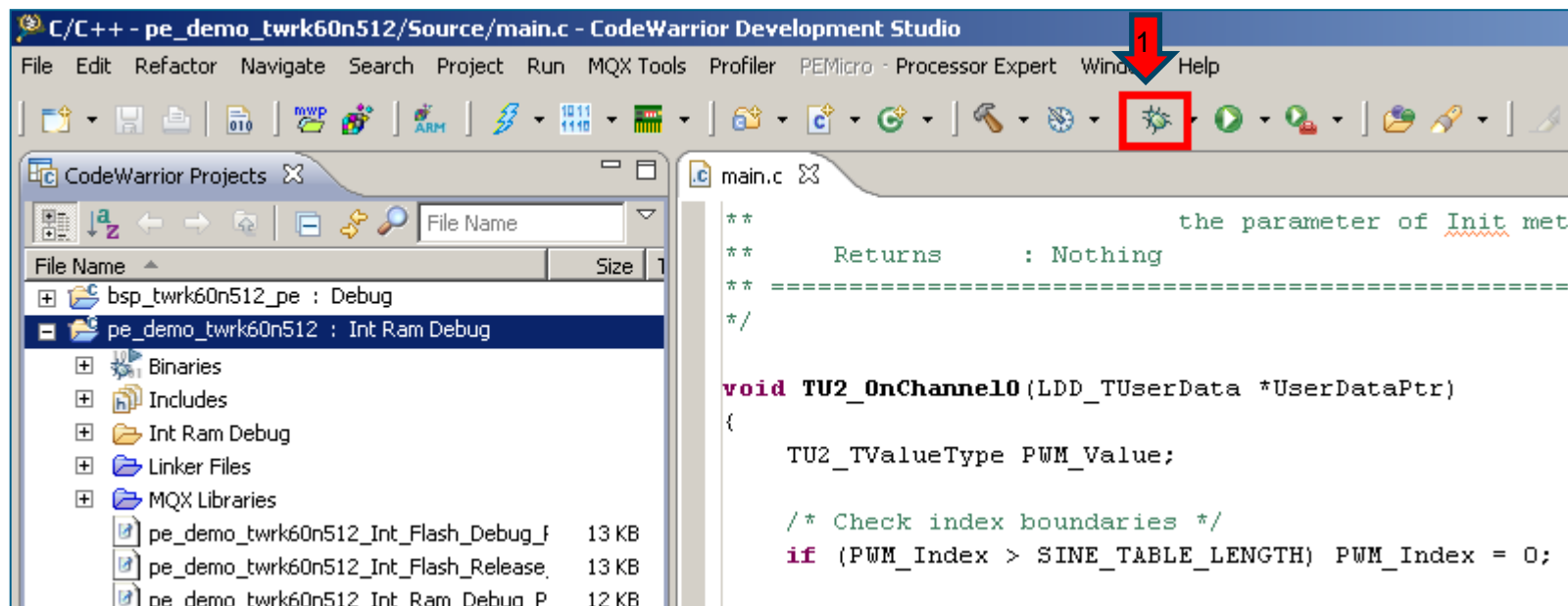


Debug Connection MQX PE Demo

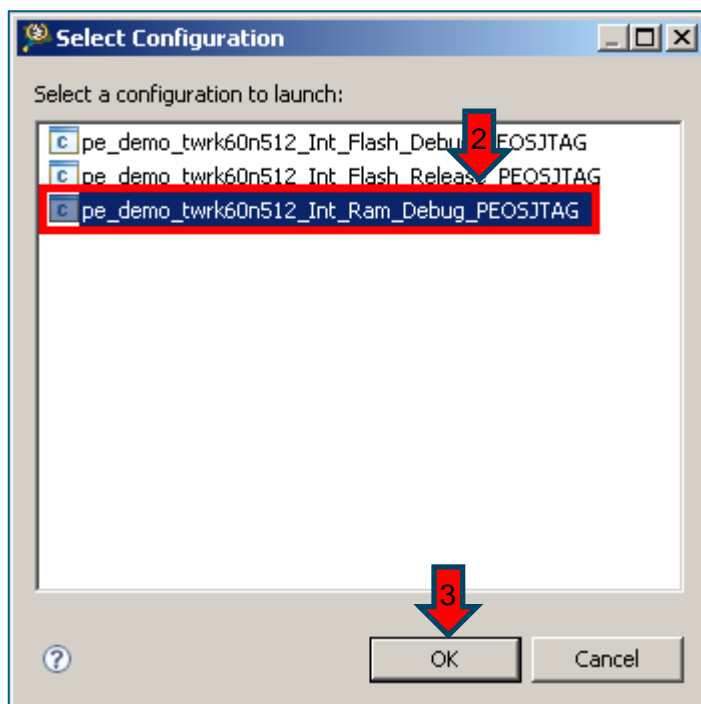
► Apply changes and OK



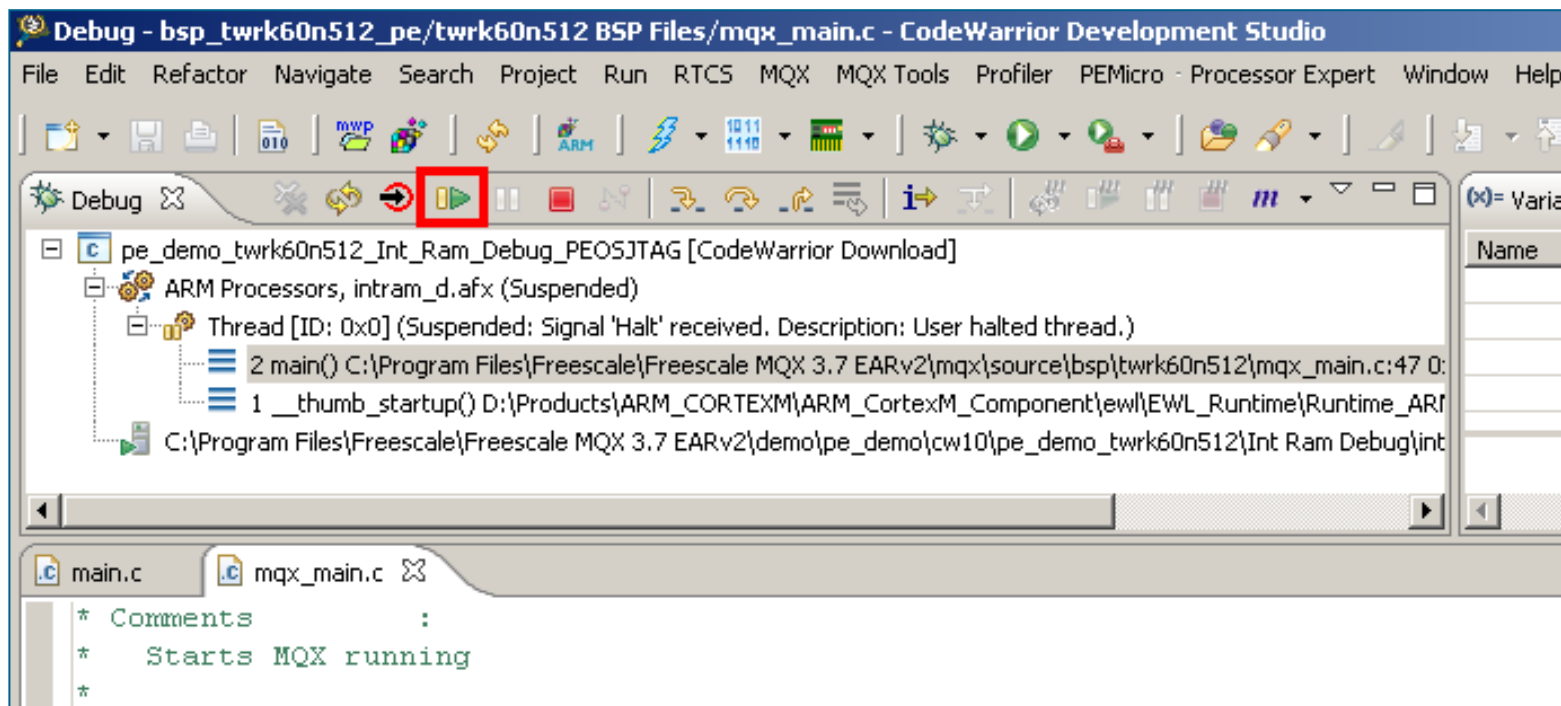
► Click Debug icon



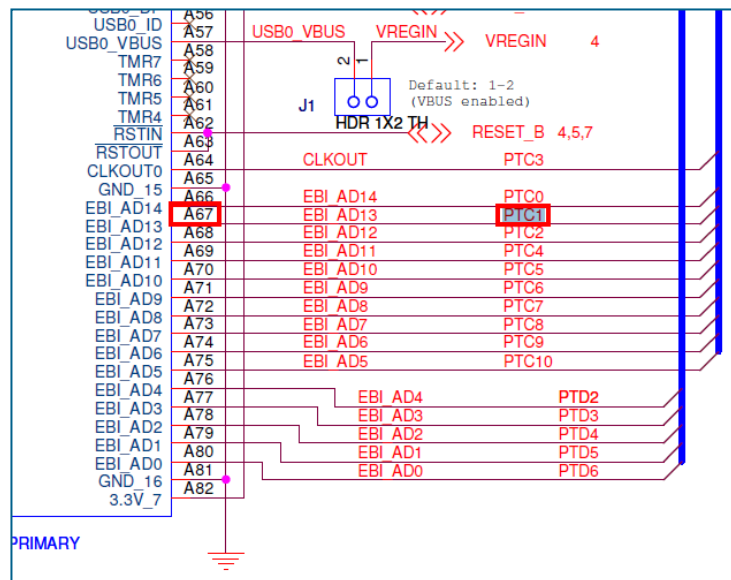
- Select **pe_demo_twrk60n512_Int_Ram_Debug_PEOSJTAG**



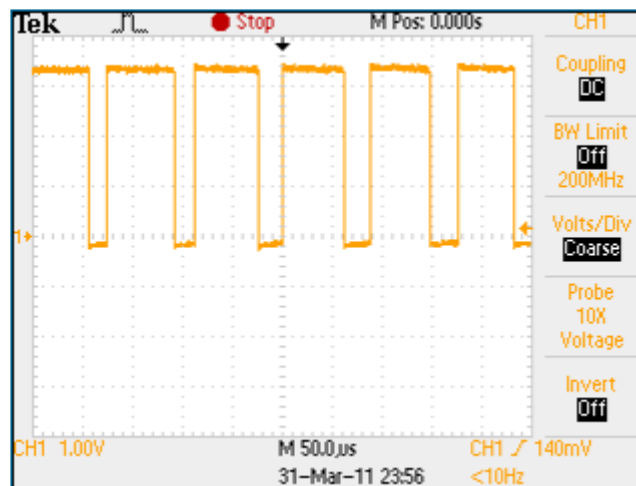
► Click Resume (F8)



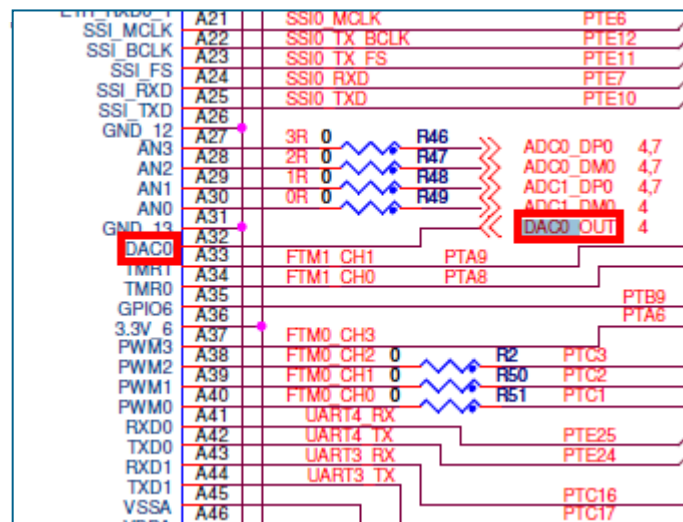
► Check PWM output on A67



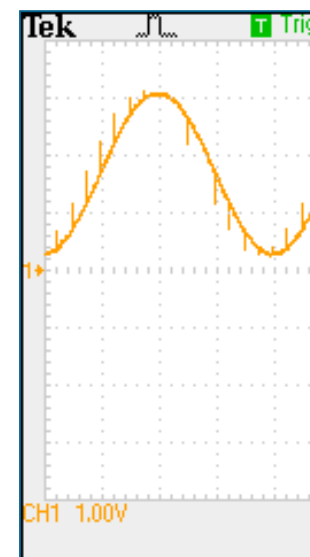
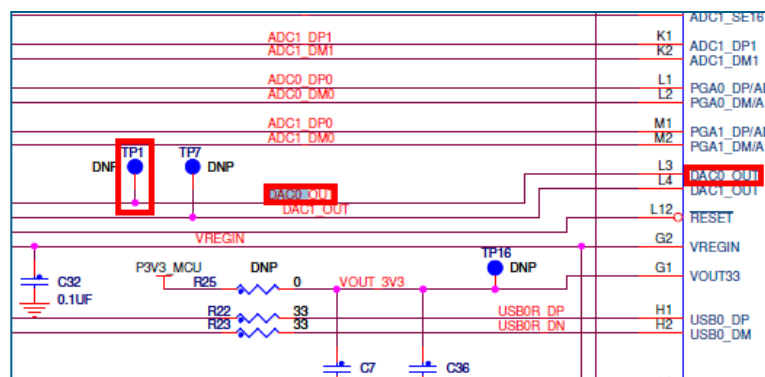
Properties	Methods	Events
Name	Value	Details
Module name	FTM0	FTM0
Counter	FTM0_CNT	FTM0_CNT
Counter direction	Up	
Counter width	16 bits	
Value type	uint16_t	uint16_t
Input clock source	Internal	
Counter frequency	48 MHz	48 MHz
Counter restart	On-match	
Period device	FTM0_MOD	FTM0_MOD
Period	4096 timer-ticks	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	



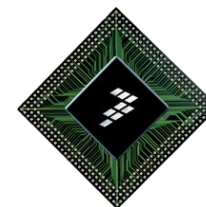
► Check DAC Output on TP1



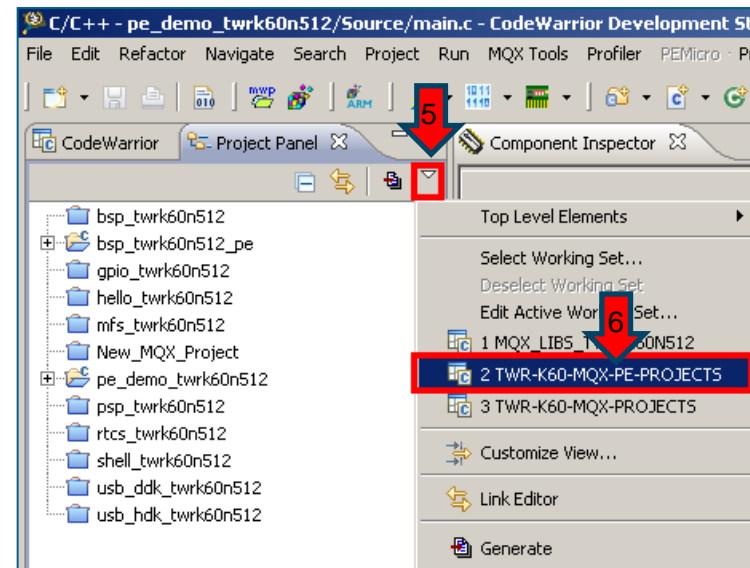
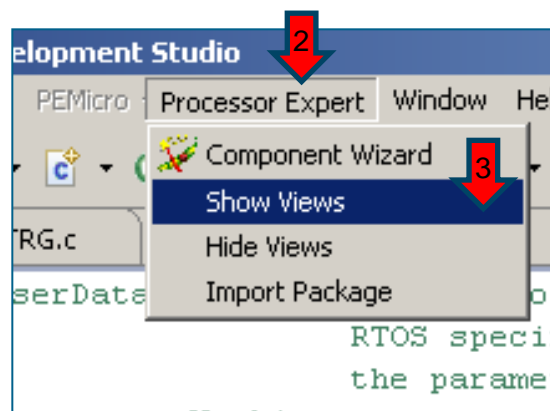
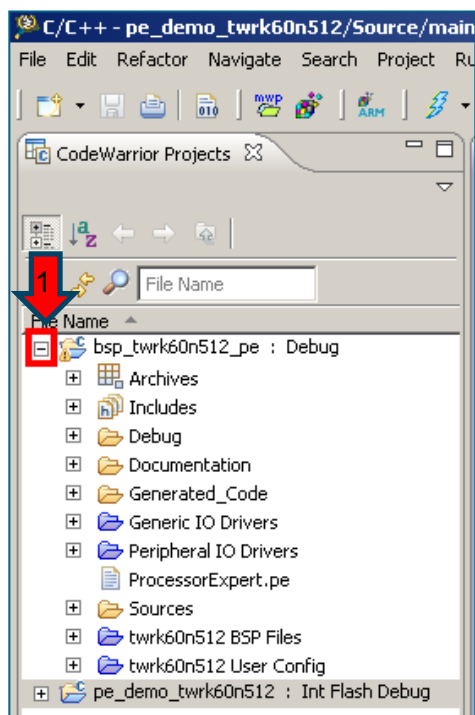
Name	Value
D/A converter	DAC0
<input checked="" type="checkbox"/> Interrupt service/event	Enabled
D/A interrupt	INT_DAC0
D/A interrupt priority	medium priority
<input checked="" type="checkbox"/> Output pin	yes
D/A channel (pin)	DAC0 OUT
Init value	0
D/A resolution	Autoselect
Data mode	unsigned 8 bits, right justified
<input checked="" type="checkbox"/> Initialization	no
Enabled in init. code	no



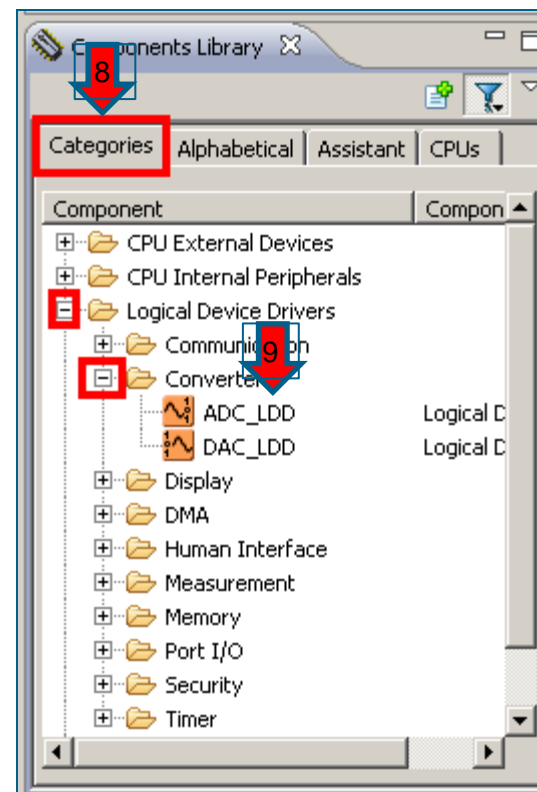
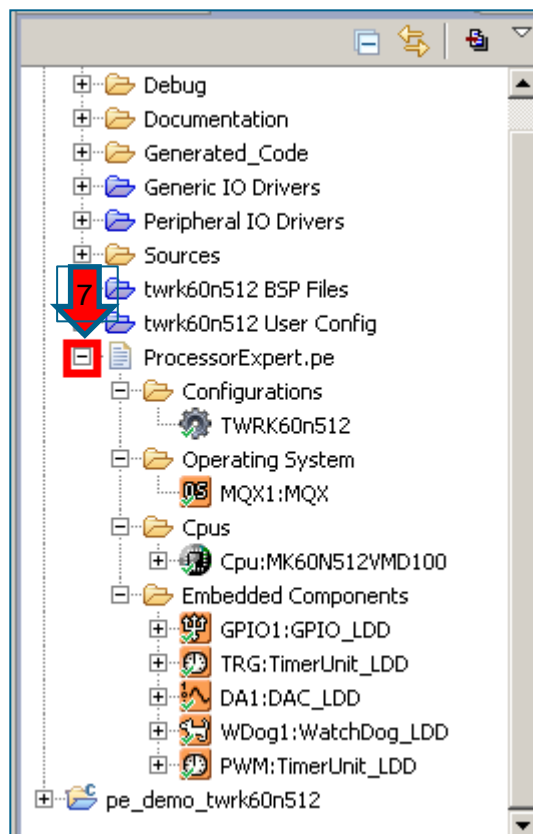
CW10.1, MQX 3.7 and PE : New LDD driver



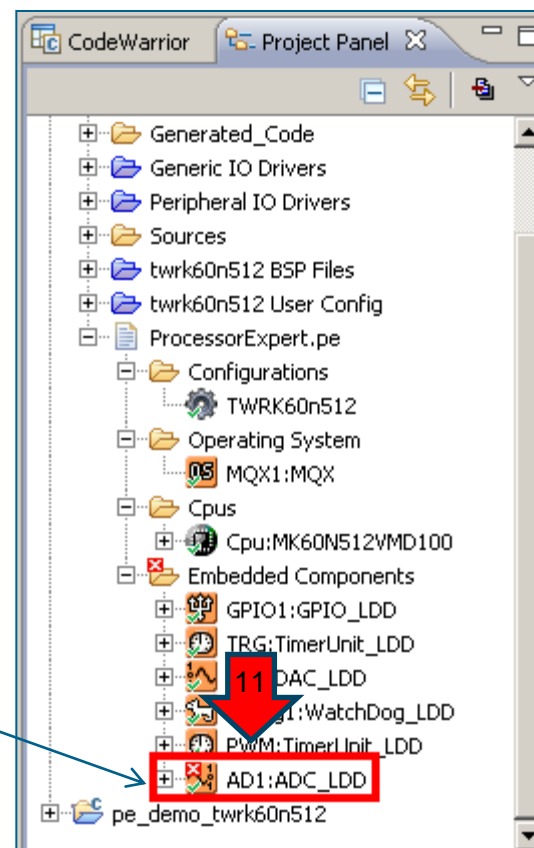
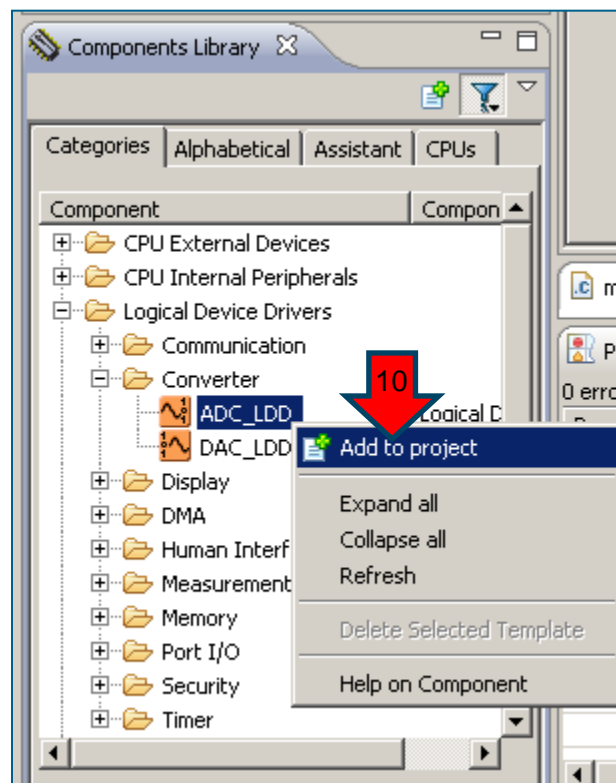
- ▶ Expand **bsp_twrk60n512_pe** project view
- ▶ Show Processor Expert View
- ▶ Select PE Projects Working Set



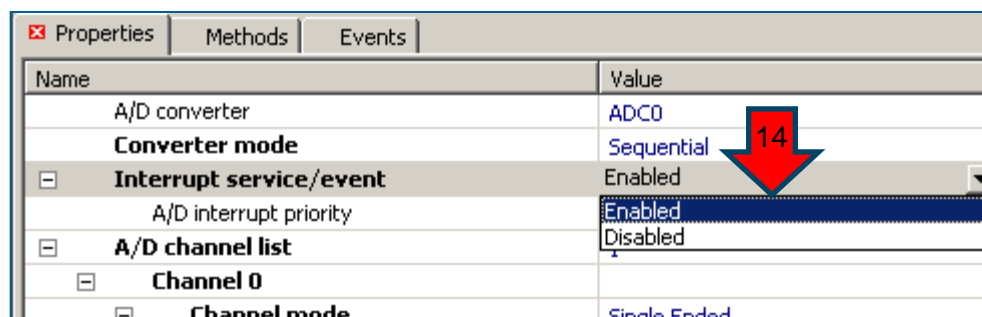
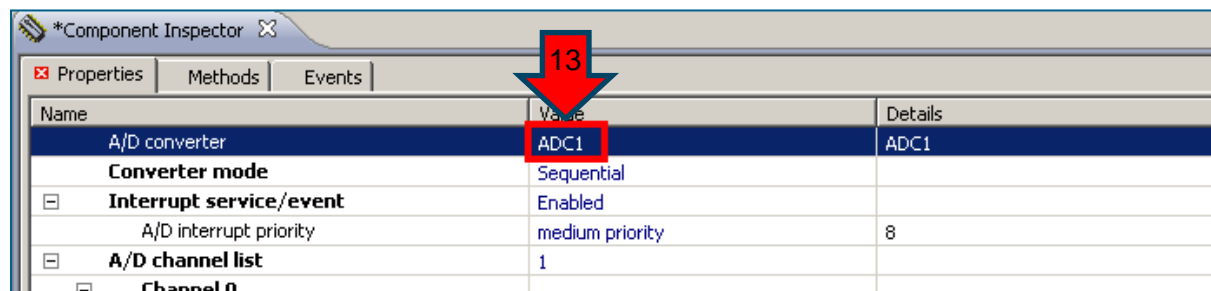
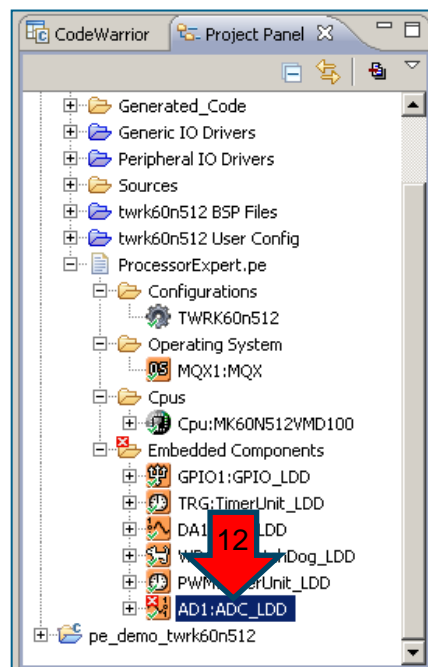
- ▶ Expand Processor Expert Project View
- ▶ Search ADC_LDD in Components Library window



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



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

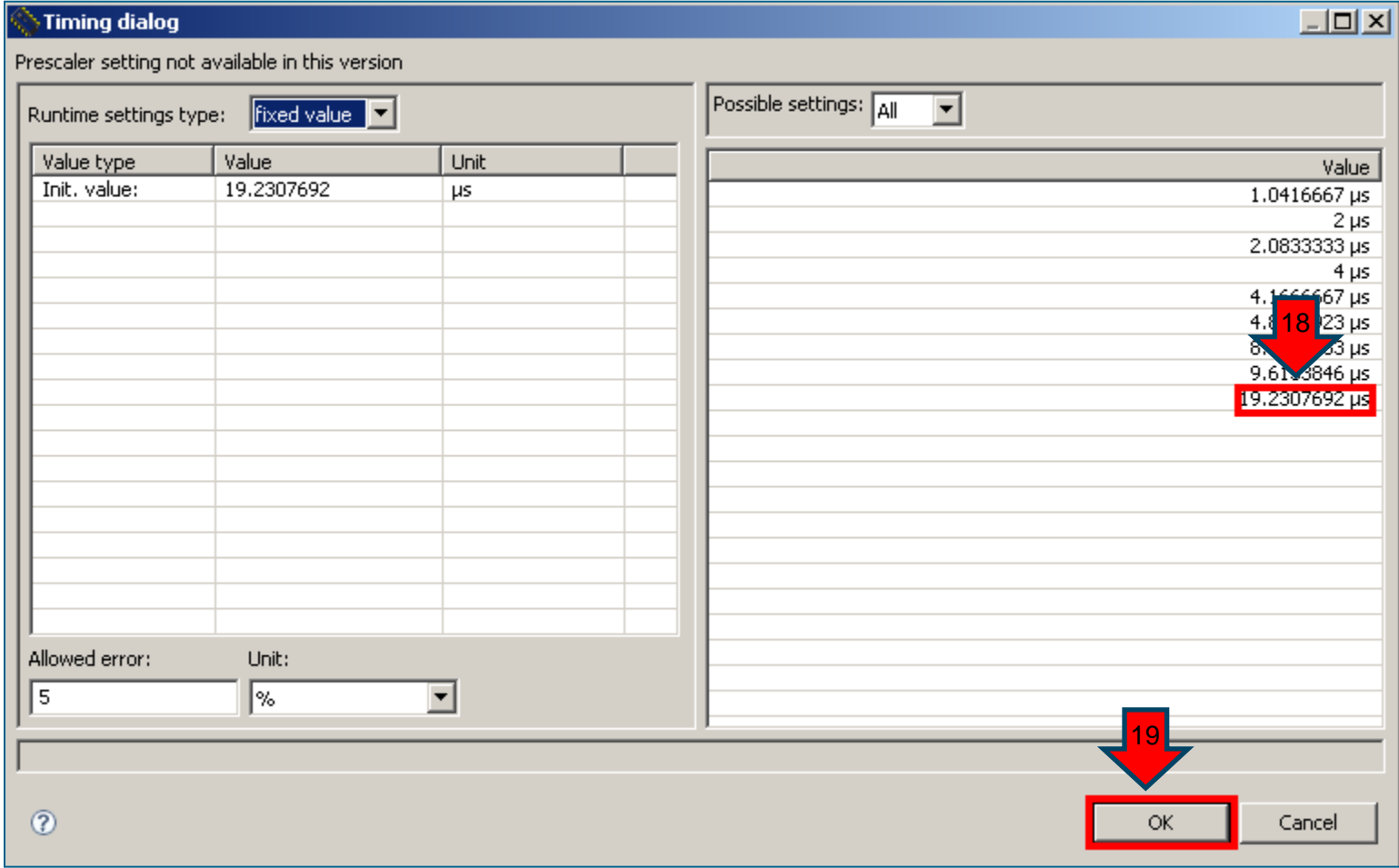


- ▶ Select **ADC1_DM1** Channel
- ▶ Enable Static sample groups
- ▶ Open Conversion Time Window

[-] 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



► 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 Description for the current value (unsigned
Enabled in init. code	yes	
<input checked="" type="checkbox"/> Event mask		
OnMeasurementComplete	Enabled	

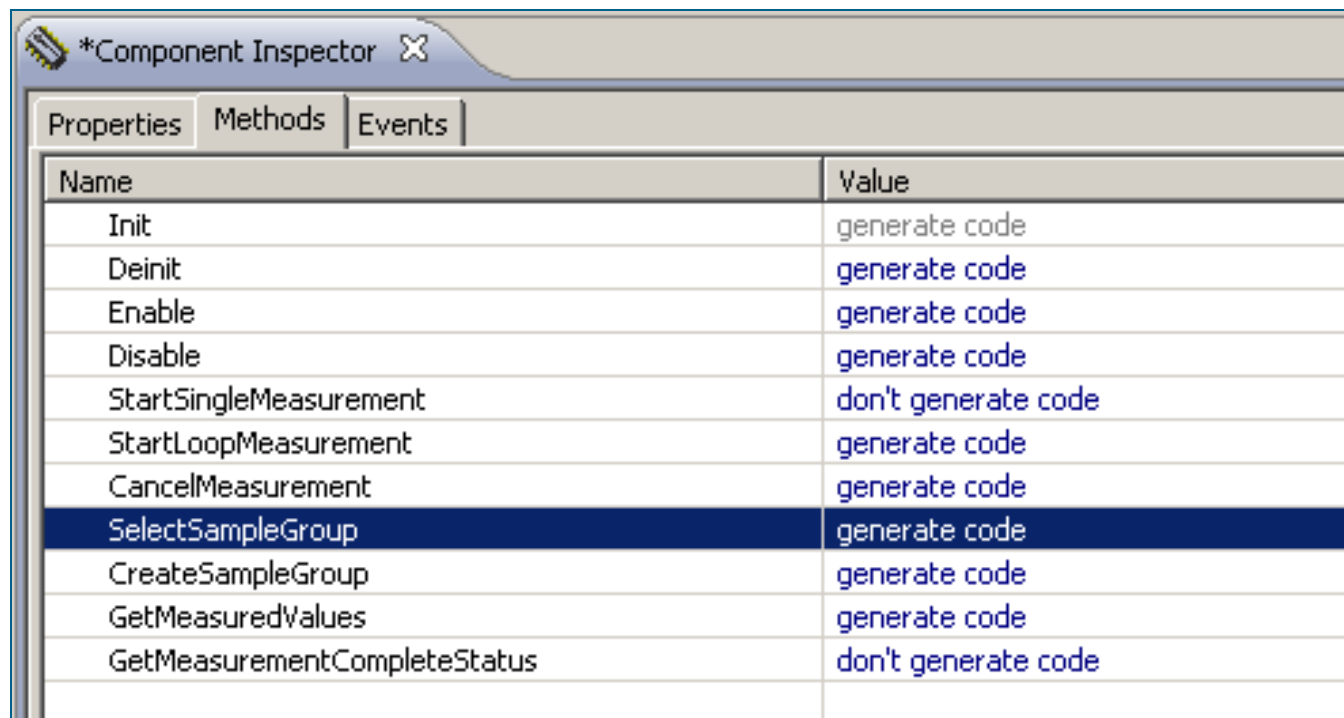
- ▶ Click Methods Tab
- ▶ Click to generate code for methods

*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	

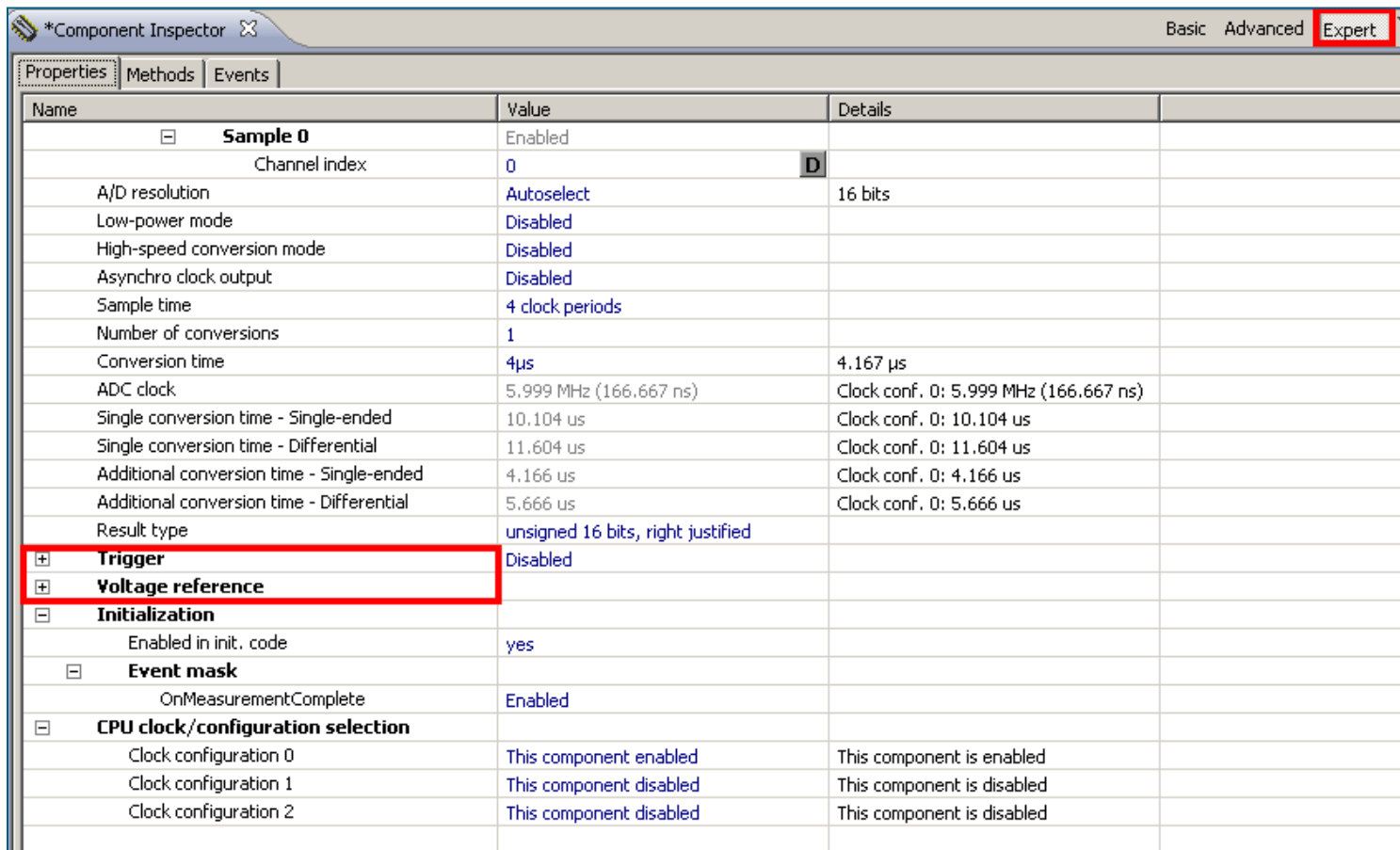
Properties	Methods	Events
Name	Value	
Init	generate code	
Deinit	generate code	
Enable	generate code	
Disable	generate code	
StartSingleMeasurement	generate code	
StartLoopMeasurement	don't generate code	
CancelMeasurement	generate code	
SelectSampleGroup	don't generate code	
CreateSampleGroup	generate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

- Set 'generate code' for the next Methods:



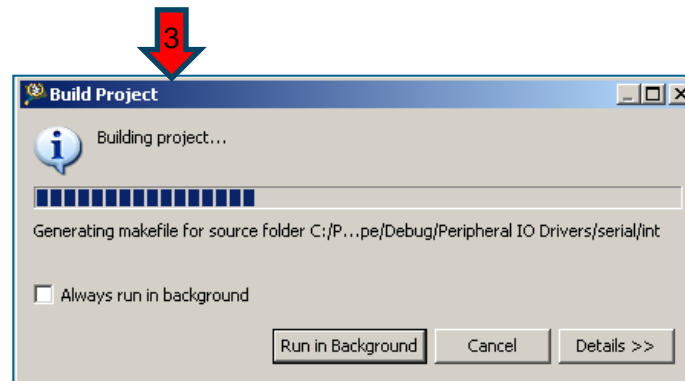
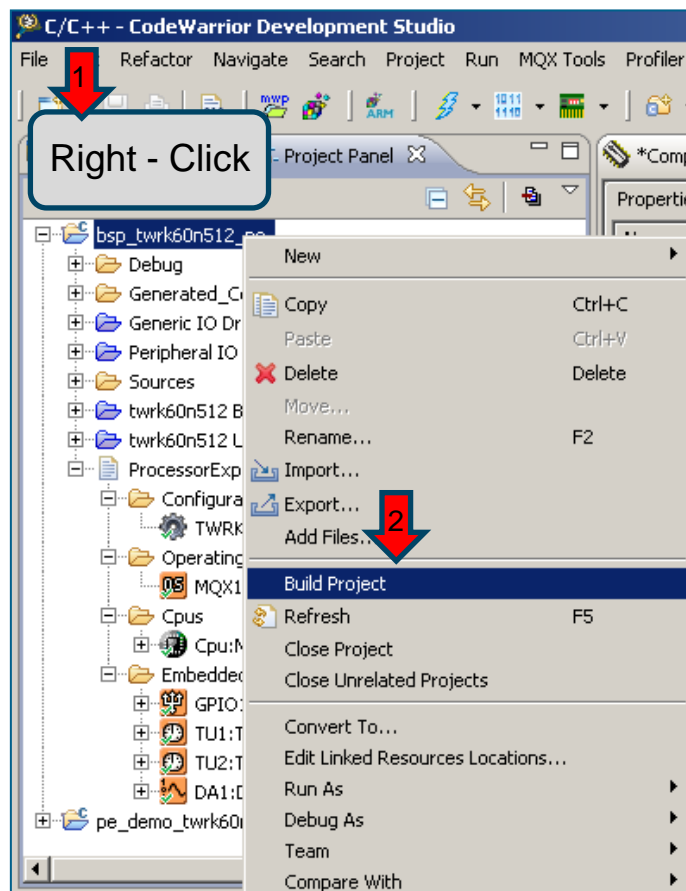
Name	Value
Init	generate code
Deinit	generate code
Enable	generate code
Disable	generate code
StartSingleMeasurement	don't generate code
StartLoopMeasurement	generate code
CancelMeasurement	generate code
SelectSampleGroup	generate code
CreateSampleGroup	generate code
GetMeasuredValues	generate code
GetMeasurementCompleteStatus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

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

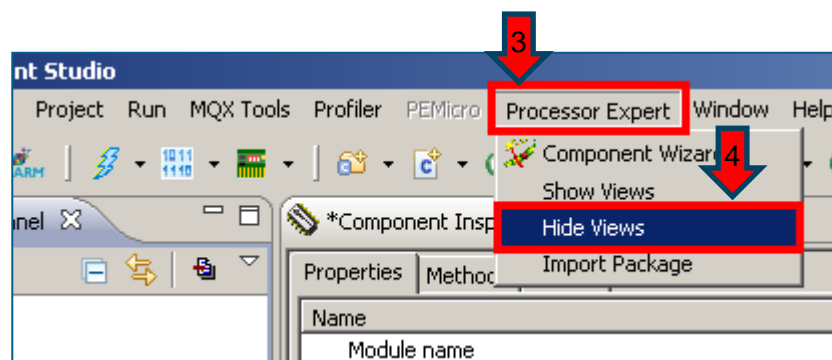
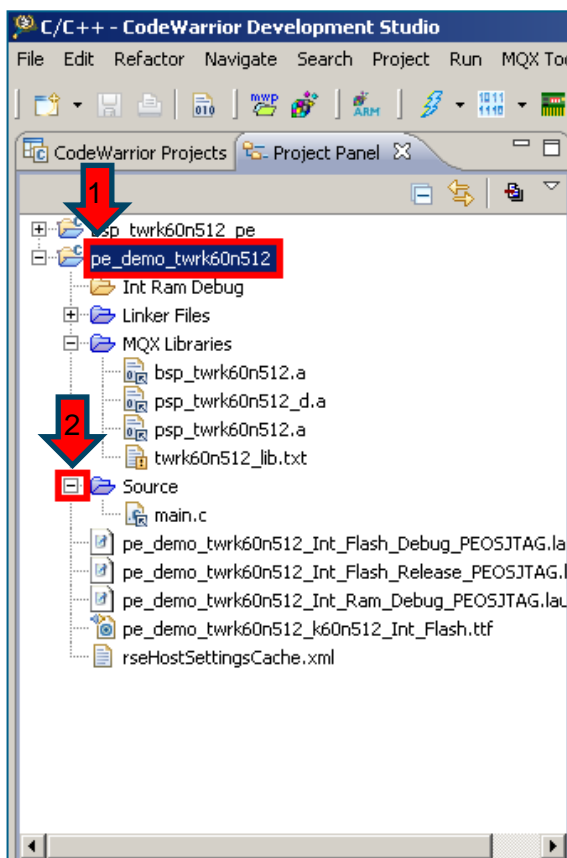


Name	Value	Details
Sample 0	Enabled	
Channel index	0	
A/D resolution	Autoselect	16 bits
Low-power mode	Disabled	
High-speed conversion mode	Disabled	
Asynchro clock output	Disabled	
Sample time	4 clock periods	
Number of conversions	1	
Conversion time	4µs	4.167 µs
ADC clock	5.999 MHz (166.667 ns)	Clock 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	
+ Trigger	Disabled	
+ Voltage reference		
- Initialization		
Enabled in init. code	yes	
- Event mask		
OnMeasurementComplete	Enabled	
- CPU clock/configuration selection		
Clock configuration 0	This component enabled	This component is enabled
Clock configuration 1	This component disabled	This component is disabled
Clock configuration 2	This component disabled	This component is disabled

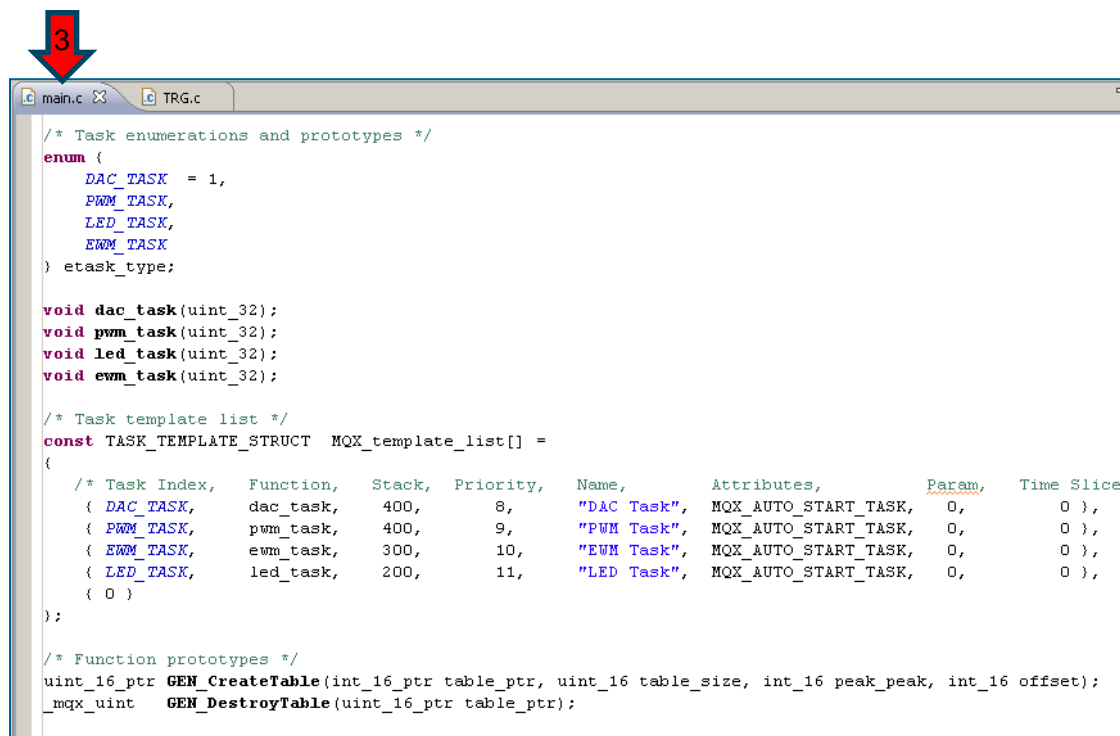
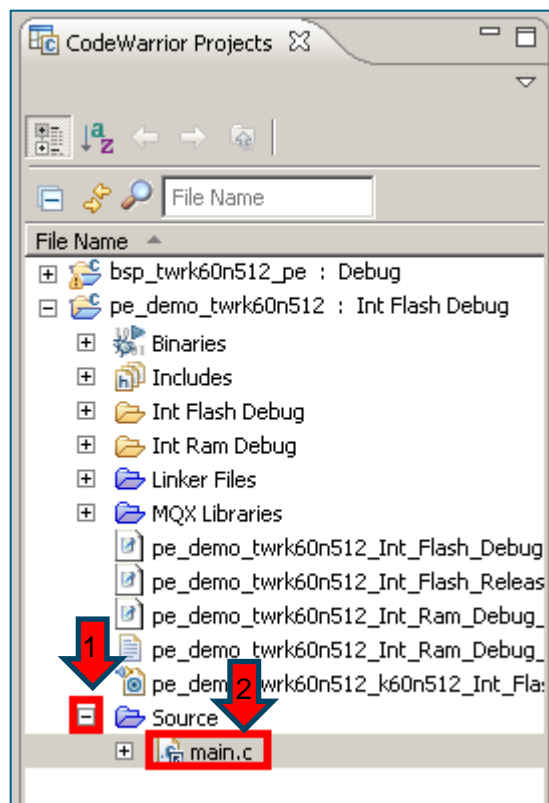
- Right-Click on Project Explorer **bsp_twrk60n512_pe** and Build Project



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



- 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_TUserData *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 */

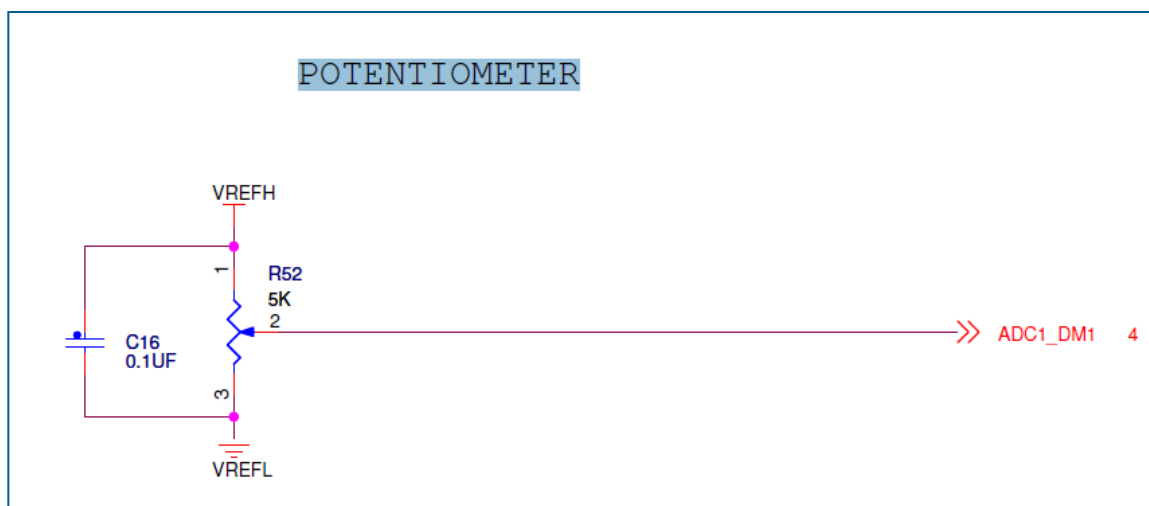
        if(MeasuredValues[0]>2000) GPIO1_ToggleFieldBits(LED_DeviceData, LED4, 1);
        _time_delay(200);
    }
}
```

► 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

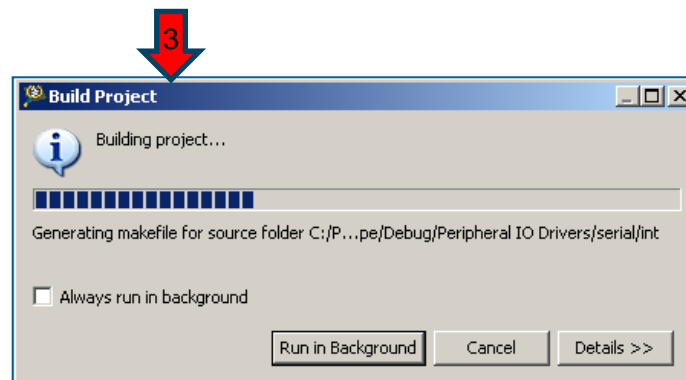
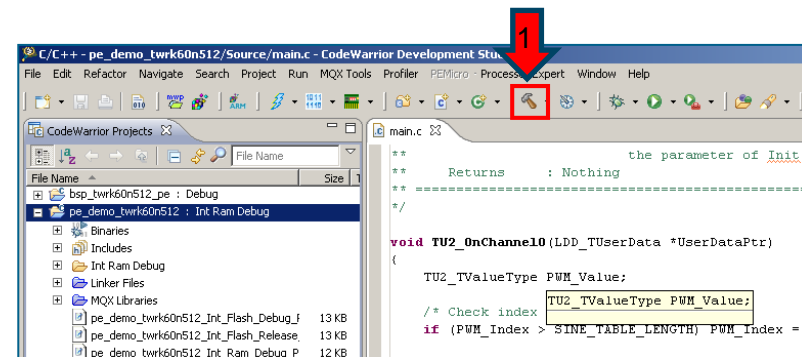
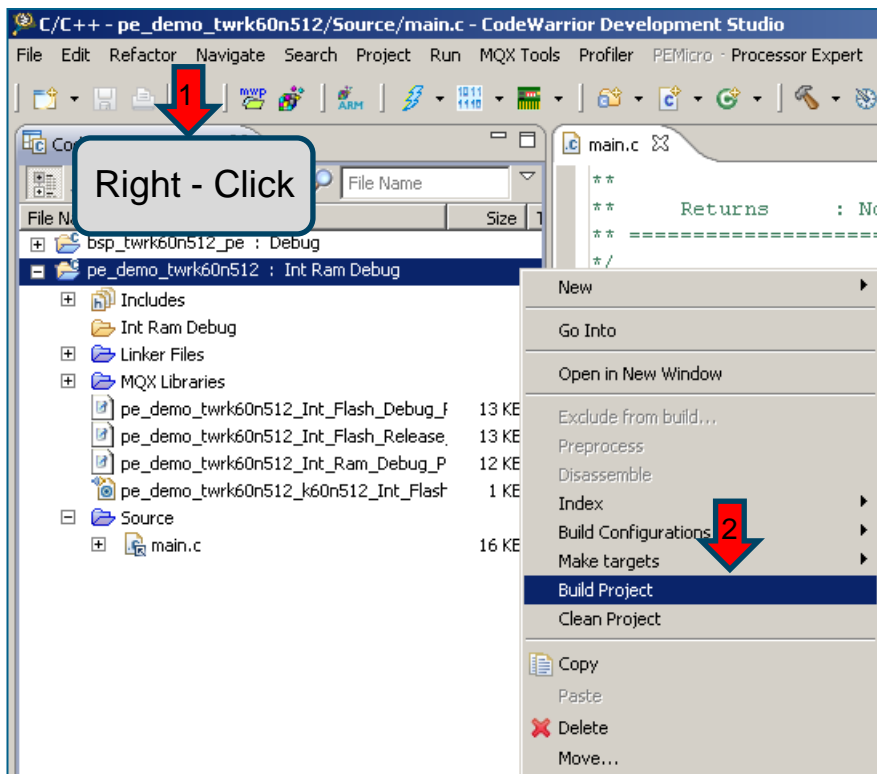


- ▶ When ADC value is greater than 20000, LED4 (Blue) toggles

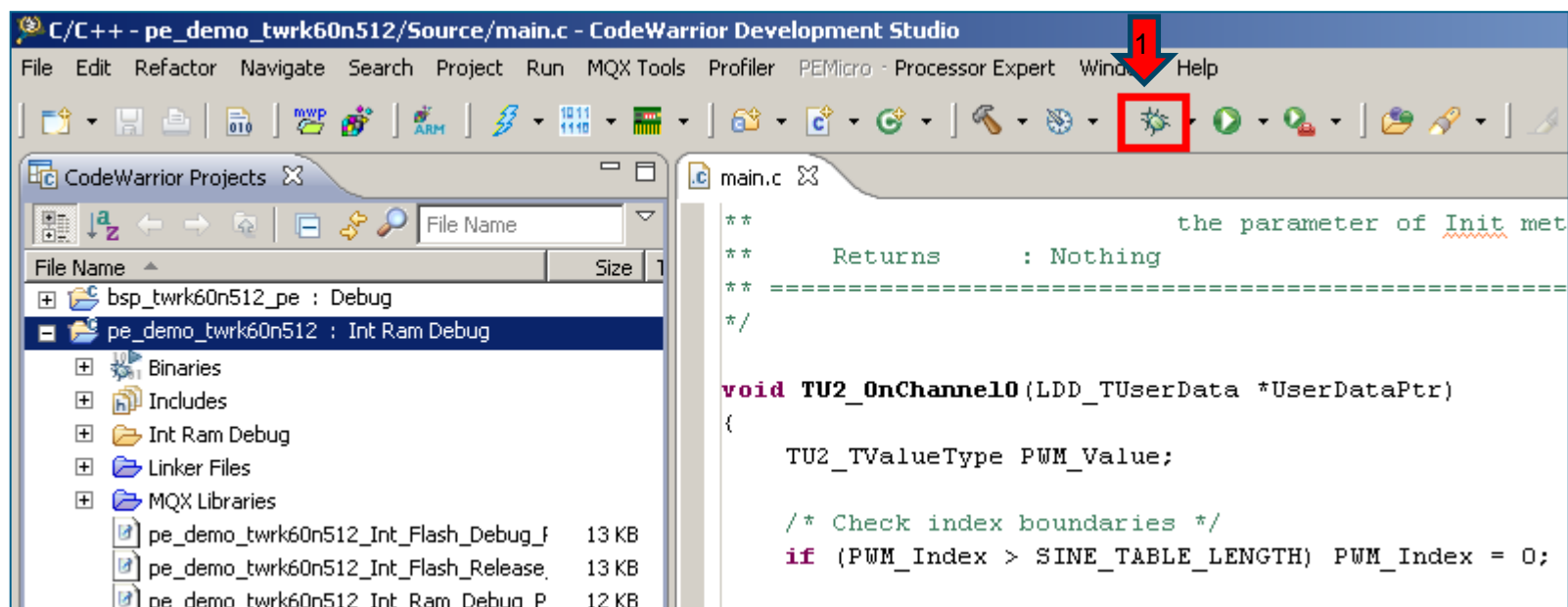
```
while(1)
{
    /* Suspend task for 100ms */
    if (MeasuredValues[0]>2000) GPIO1_ToggleFieldBits(LED_DeviceData, LED4, 1);
    _time_delay(200);
}
```

- ▶ Moving potentiometer R52 can start/stop LED4 toggle

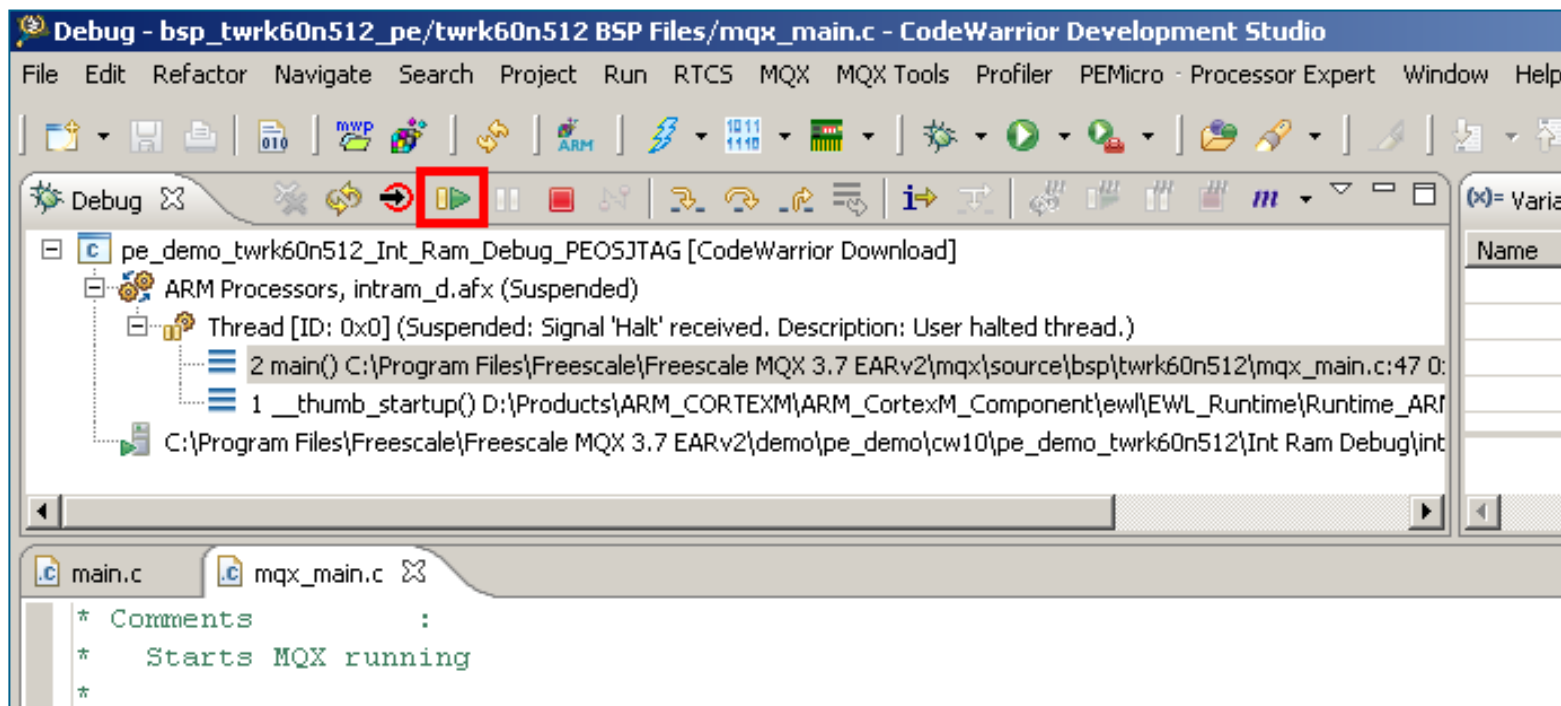
- ▶ Right-Click on Project Explorer **pe_demo_twrk60n512** and Build Project or click on icon 



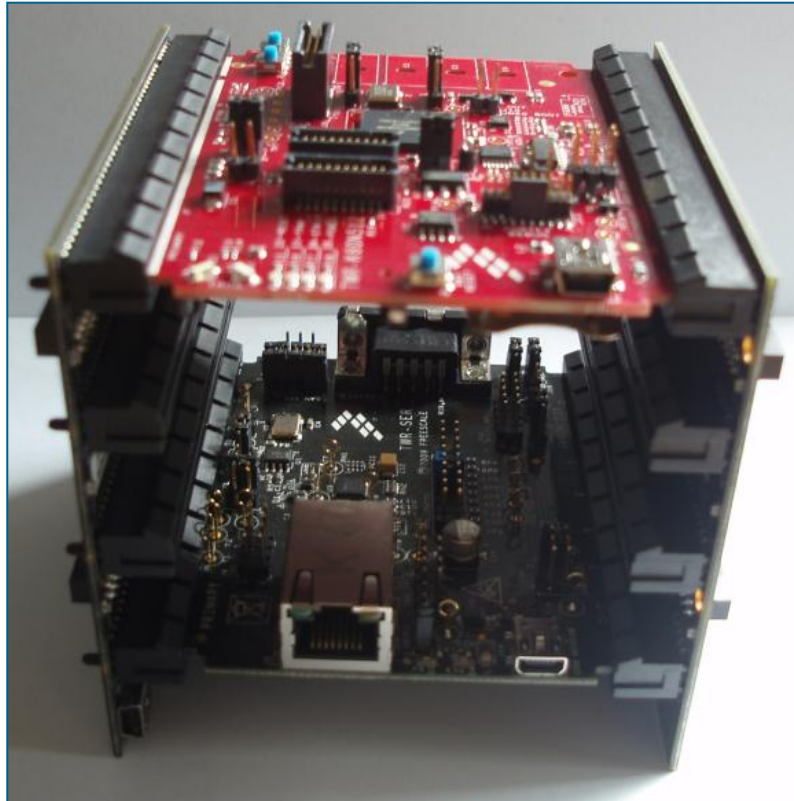
► Click Debug icon



► Click Resume (F8)



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



► <http://www.freescale.com/infocenter/Codewarrior/index.jsp>

freescale
semiconductor

Welcome to the Freescale Infocenter

Search: **GO** Search scope: All topics

Contents

- Welcome
- What's New?
28 February 2011
- CodeWarrior for Microcontrollers V10.x**
 - Getting Started
 - HCS08/RS08 for Microcontrollers
 - ColdFire for Microcontrollers
 - Power Architecture Processors for Microcontrollers
 - Kinetis for Microcontrollers
 - Common Manuals
 - Run Control Manuals
 - Processor Expert Manuals
 - Application Notes
 - Microcontrollers 10.x FAQ Guide
 - Introduction
 - IDE
 - Project Management
 - Project
 - Compiler
 - Linker

CodeWarrior for Microcontrollers V10.x > Processor Expert Manuals > Processor Expert Components Manual > Installed Components

Init_MDHA

[General Info](#)
[Properties](#)
[Methods](#)

Component Init_MDHA for MCF

Message Digest Hardware Accelerator

Component Level: Peripheral Initialization
Category: CPU Internal Peripherals-Peripheral Initialization

This component provides initialization of the MDHA module.

Peripheral Initialization Components provide a low-level hardware approach to initialize components. These components are intended for experienced users.

PROCESSOR EXPERT is trademark of Freescale Semiconductor, Inc.
Copyright 1997 - 2010 Freescale Semiconductor, Inc.

