
Brooklyn Board by Silica Architech Documentation

Release .0

Silica

Mar 16, 2017

Contents

1	Installing Codewarrior on WIN7 or WIN8	3
1.1	Codewarrior on WINXP	6
2	Quick start guide	9
2.1	Hardware requirements	9
2.2	Software requirements	10
2.3	Hardware setup	10
2.4	Brooklyn Board MQX FW setup	14
2.5	IMPORTING AND BUILDING MQX LIBRARY	20
2.6	IMPORTING Pmod1_6 FIRMWARE	22
2.7	BUILDING Pmod1_6 FIRMWARE	27
2.8	Running Brooklyn Board MQX FW	33
3	Firmware details	37
3.1	Main project files from Maxim	37
3.2	General include files	37
3.3	Main Project files added	39
3.4	MQX tasks brief	41
4	Firmware changes	43
5	Tips and Tricks	47
6	More about Pmod	55
6.1	Important notice	55
6.2	Emulation of MAX3232	56

Version 1.0.0

Copyright (C)2016 Avnet Silica company



Silica Brooklyn Board is useful system to evaluate MAXIM Pmod device and is designed for use with Freescale TWR-K70F120M tower system

This software release is working with MQX4.0 Rtos

You can find and download TWR-K70 documentation by clicking:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=TWR-K70F120M&tid=m32TWR

Firmware application was developed with CodeWarrior MCU v10.3 Special Edition. **It's strongly recommended to use Codewarrior 10.3 to build this project.**

No MQX installation is required. Project contain all MQX resources needed for full functionality.

Codewarrior v10.3 Special Edition is free downloadable from Freescale site. Go to:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-MCU10&fsp=1&tab=Design_Tools_Tab

Codewarrior v10.3 Special Edition is also downloadable [here](#)

Please for download select "offline" package. (note that download can take much time ...)

We suggest you to read the Quick Start Guide to setup your evaluation system

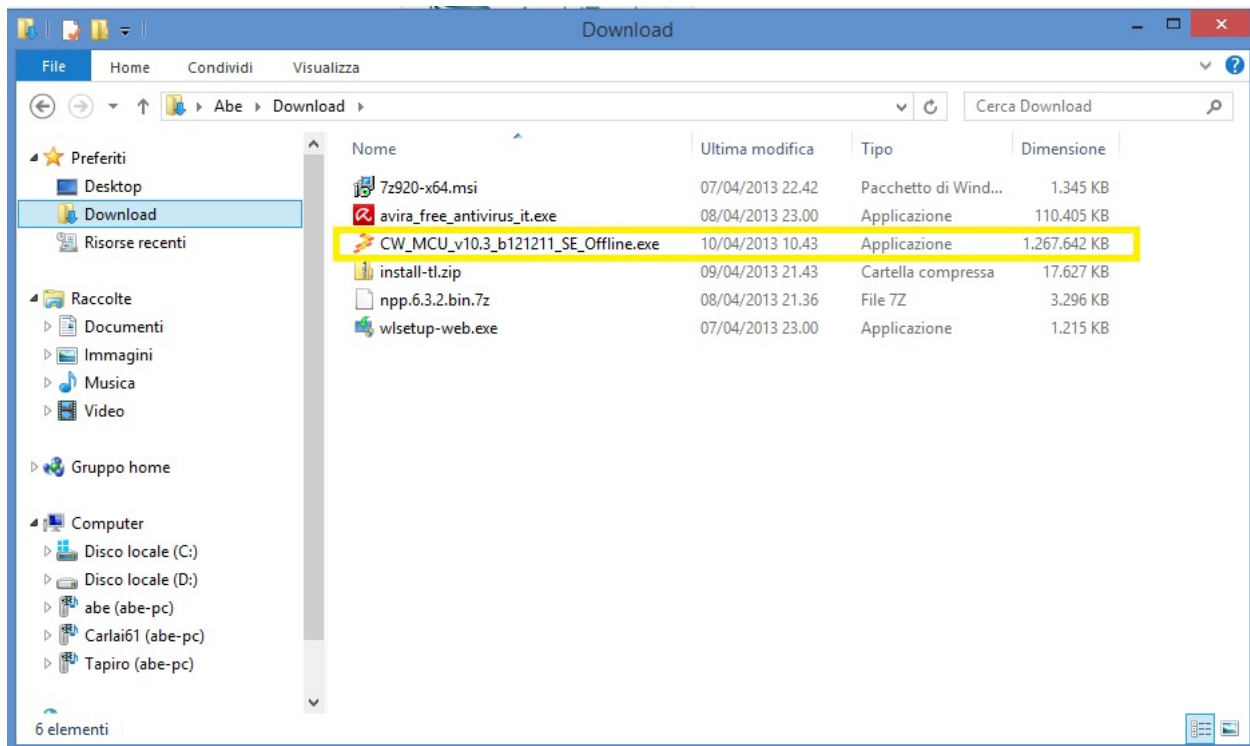
[Quick start guide](#)

This guide explains how to use this application and provides an overview of on the structure of the project firmware

CHAPTER 1

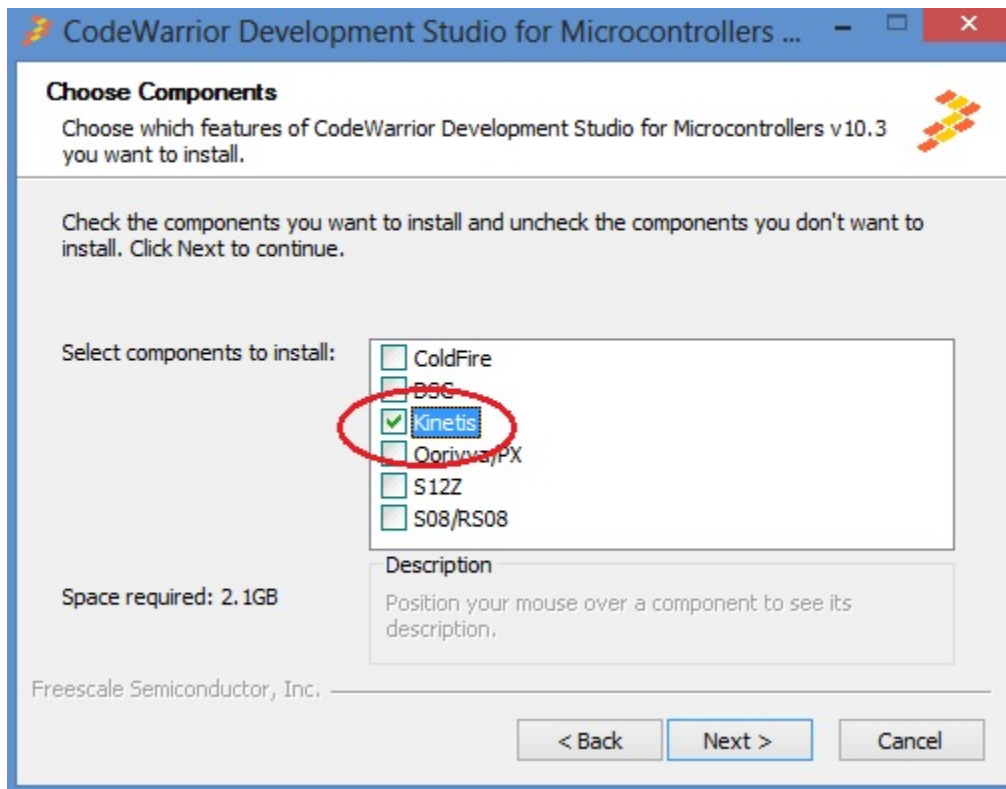
Installing Codewarrior on WIN7 or WIN8

Double click on 'CW_MCU_v10.3_b121211_SE_Offline.exe' that you have downloaded first, and follow installation instruction.

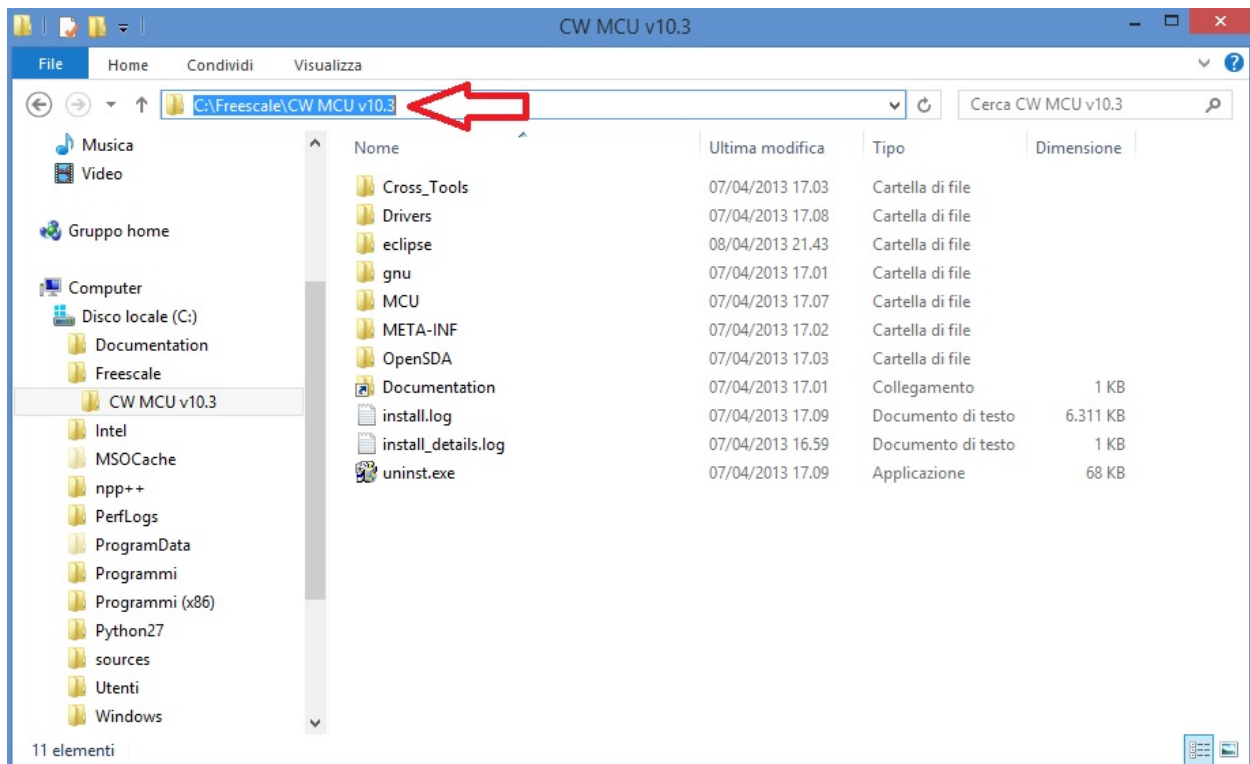


Quick start guide for Codewarrior install can be found at http://cache.freescale.com/files/soft_dev_tools/doc/quick_ref_guide/MCU_QS.pdf?fp=1

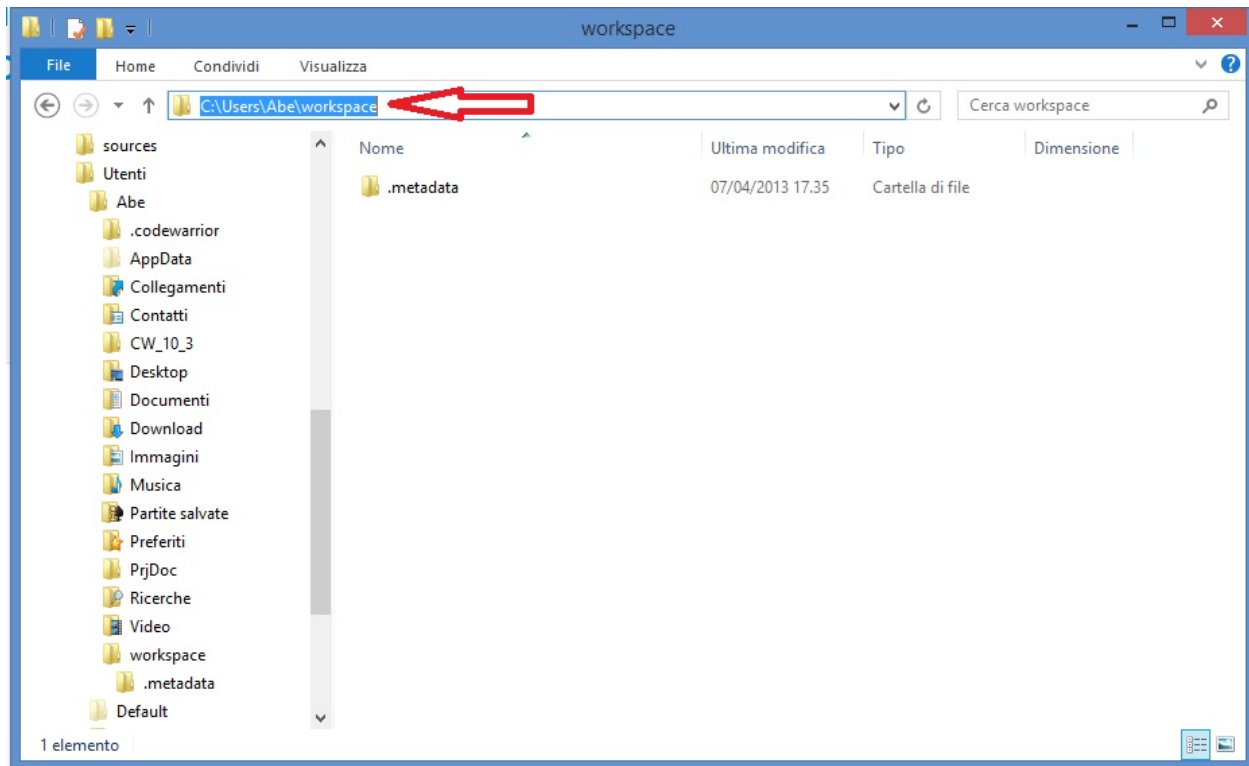
When setup ask for MCU type, select almost Kinetis as show below, then press NEXT button



If you have Windows7 or Windows8 (32 or 64 bit) Codewarrior will install into “Freescale\CW MCU v10.3” folder on the root of your system HDD.

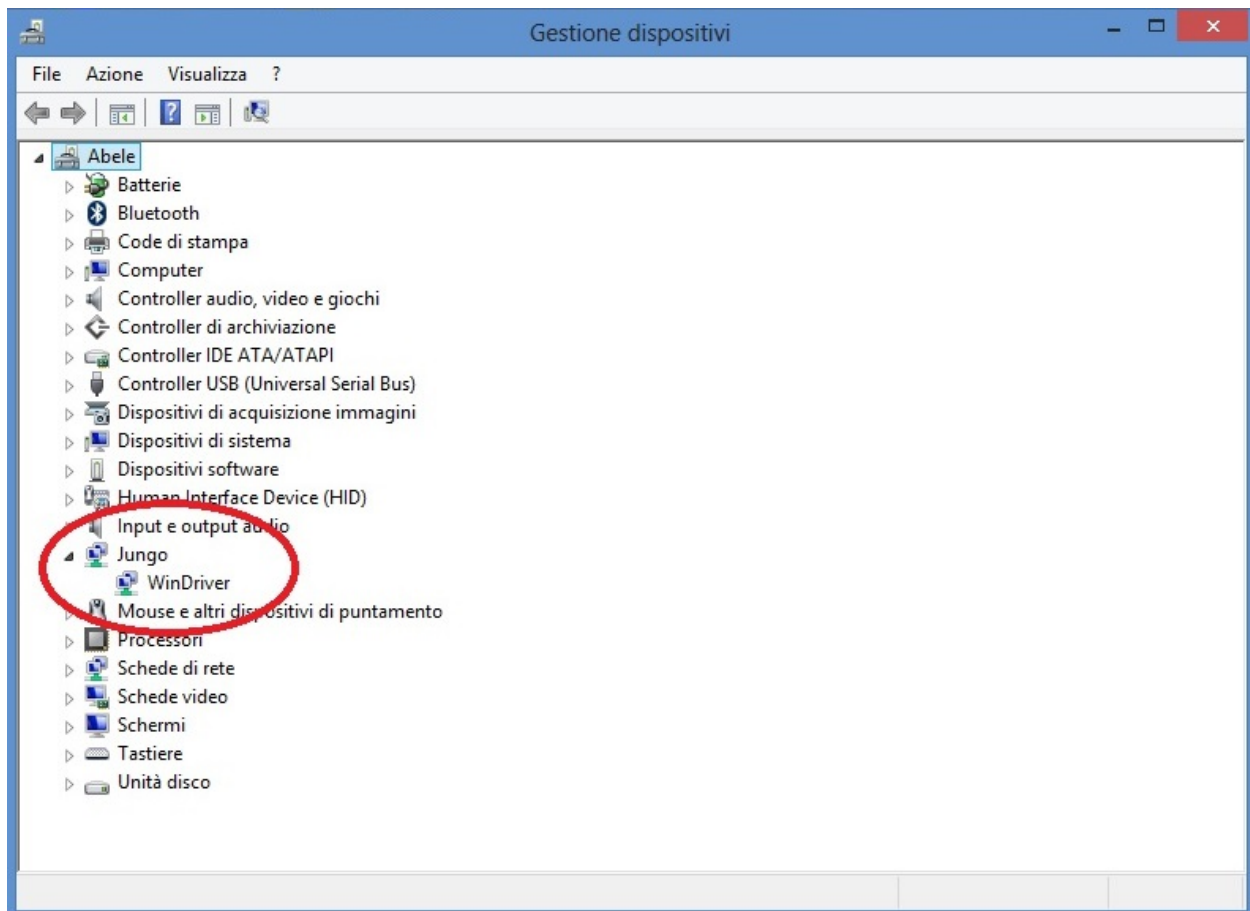


and create a default work folder named **workspace** in the path `C:\Users\your_user_name\workspace`



make shure that setup have been installed Jungo Driver. See your system configuration (righ-click on Computer -> Properties -> Device Manager)

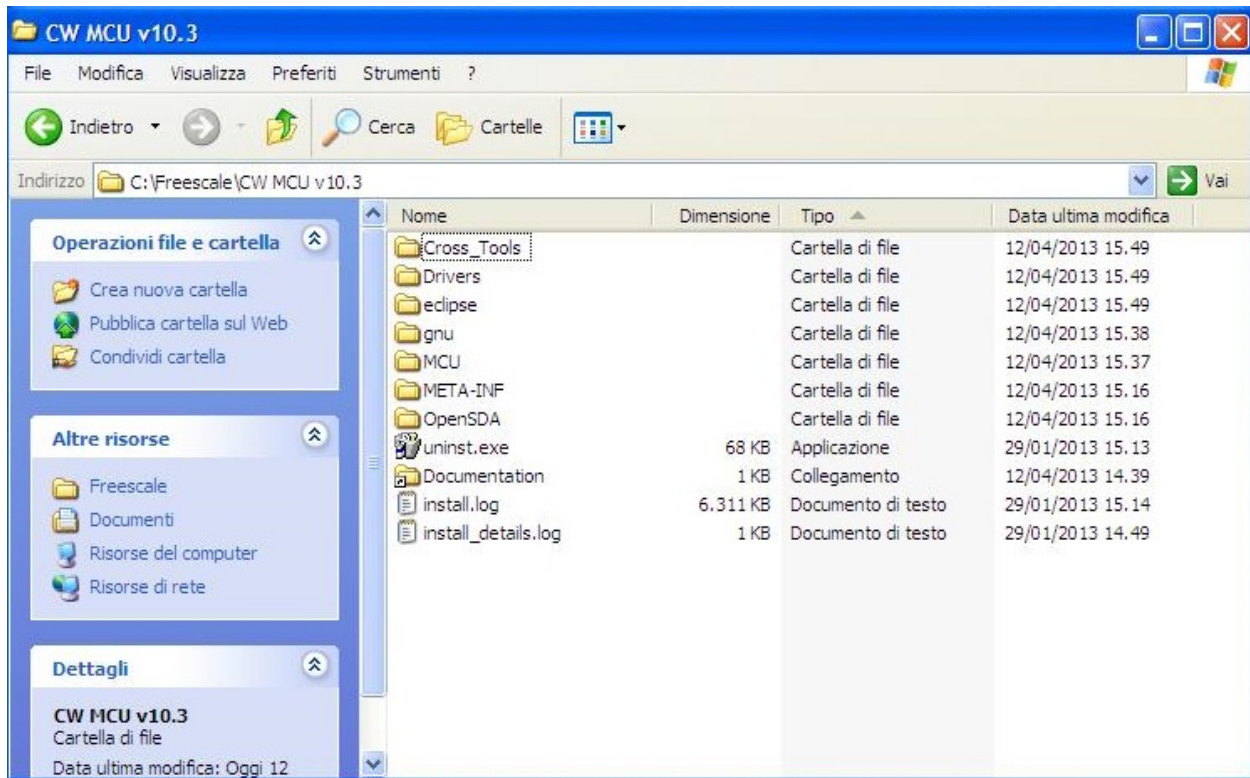
if you have any troubles about, read Codewarrior install Guide



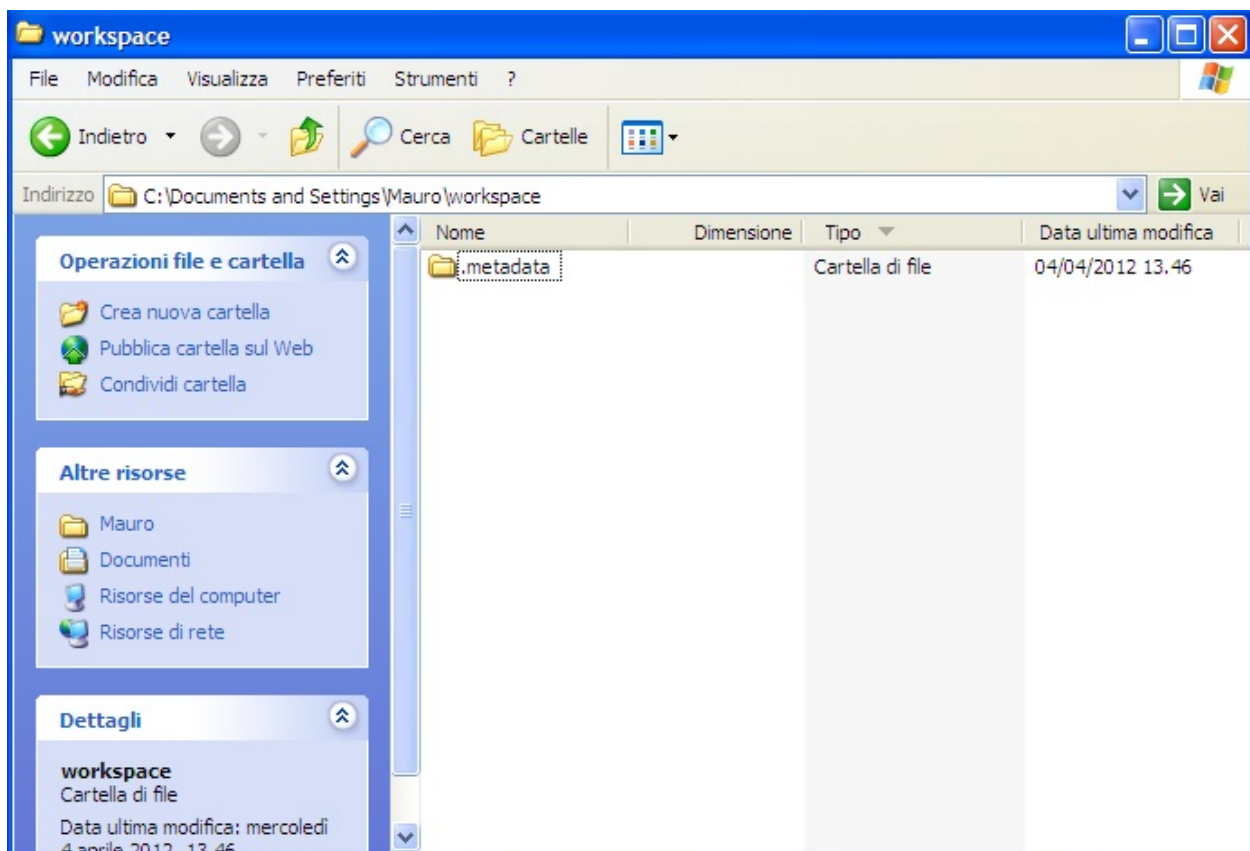
Codewarrior on WINXP

For WinXP, after installation, you have:

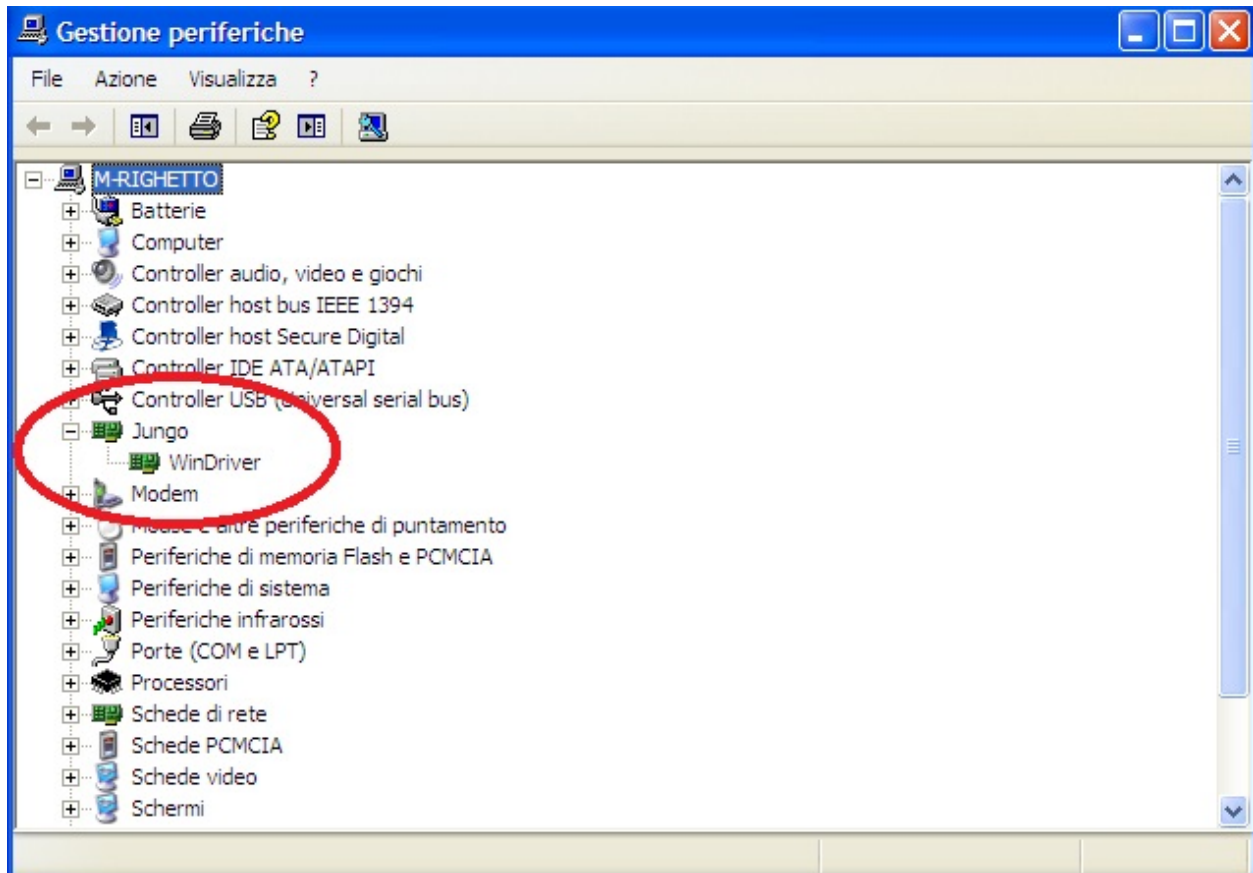
- installation folder



- default work folder in `c:\Documents and Settings\your_user_name\workspace`

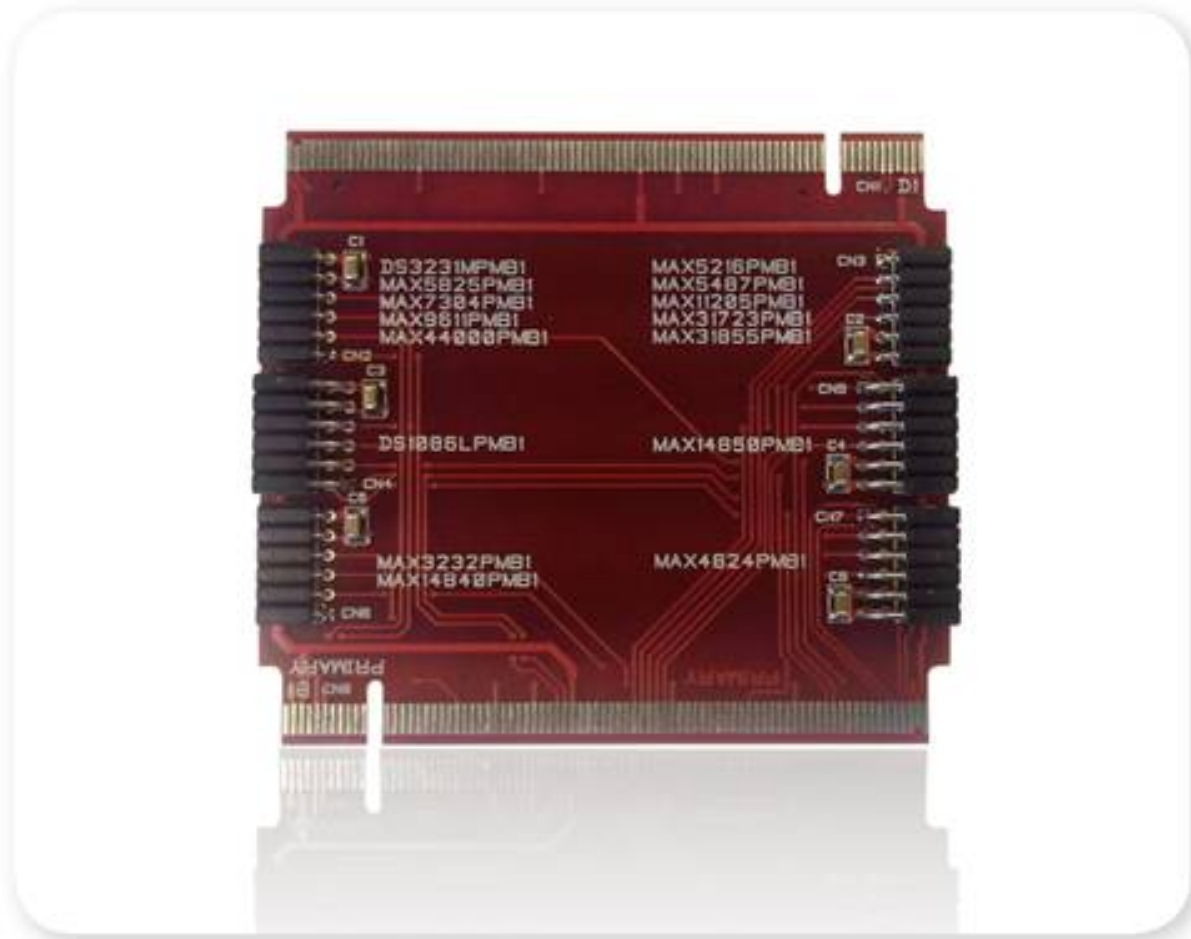


- Jungo driver:



Hardware requirements

- Tower system for Kinetis K70F120M (with TWR-SER expansion)
- Mini USB type-B cable
- Silica BrooklynBoard
- PC with at least one RS232 serial port and terminal software (two serial port for MAX3232 emulation)
- RS232 DB9 serial cable (modem type)
- Maxim Analog Essential Collection

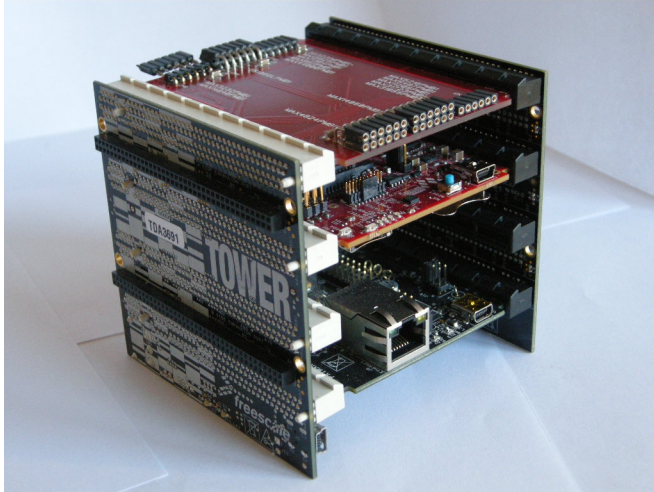


Software requirements

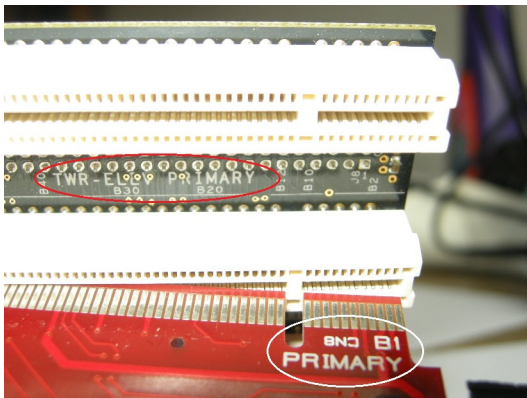
- CodeWarrior MCU v10.3 Special Edition ([download here](#)).
- Brooklyn Board application firmware for TWR-K70F120M system ([download here](#))

Hardware setup

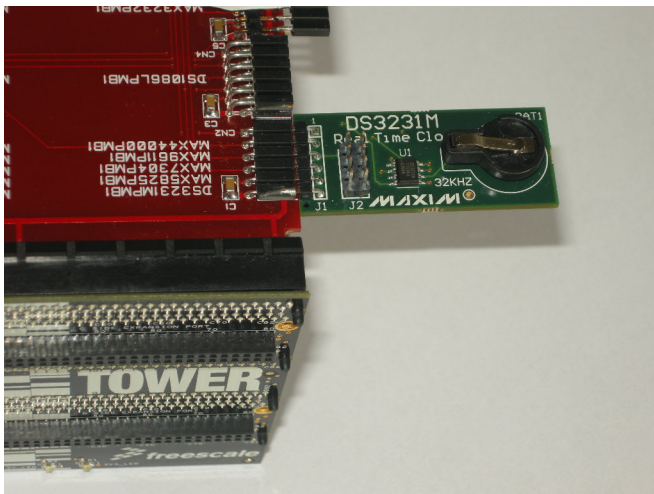
- Assemble tower system TWR-K70F120M and Brooklyn Board as in figure below.



Don't care slot position, but be careful to connect Primary and Secondary connector properly. Take care at reference signed near PCI board connectors.



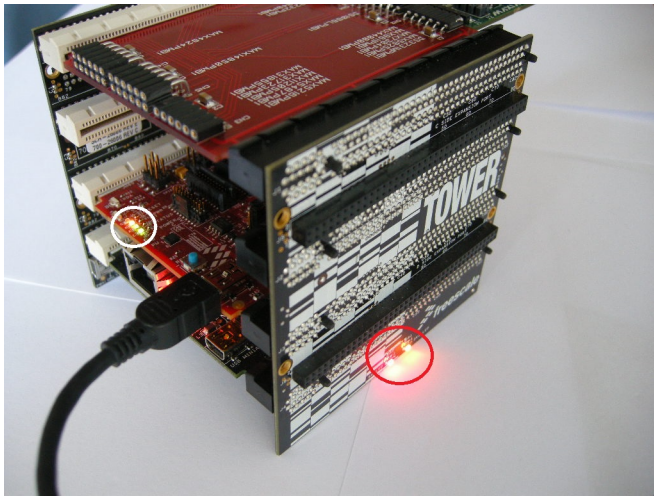
- Plug a Pmod Device (i.e. DS3231M Real Time Clock) inside properly connector.



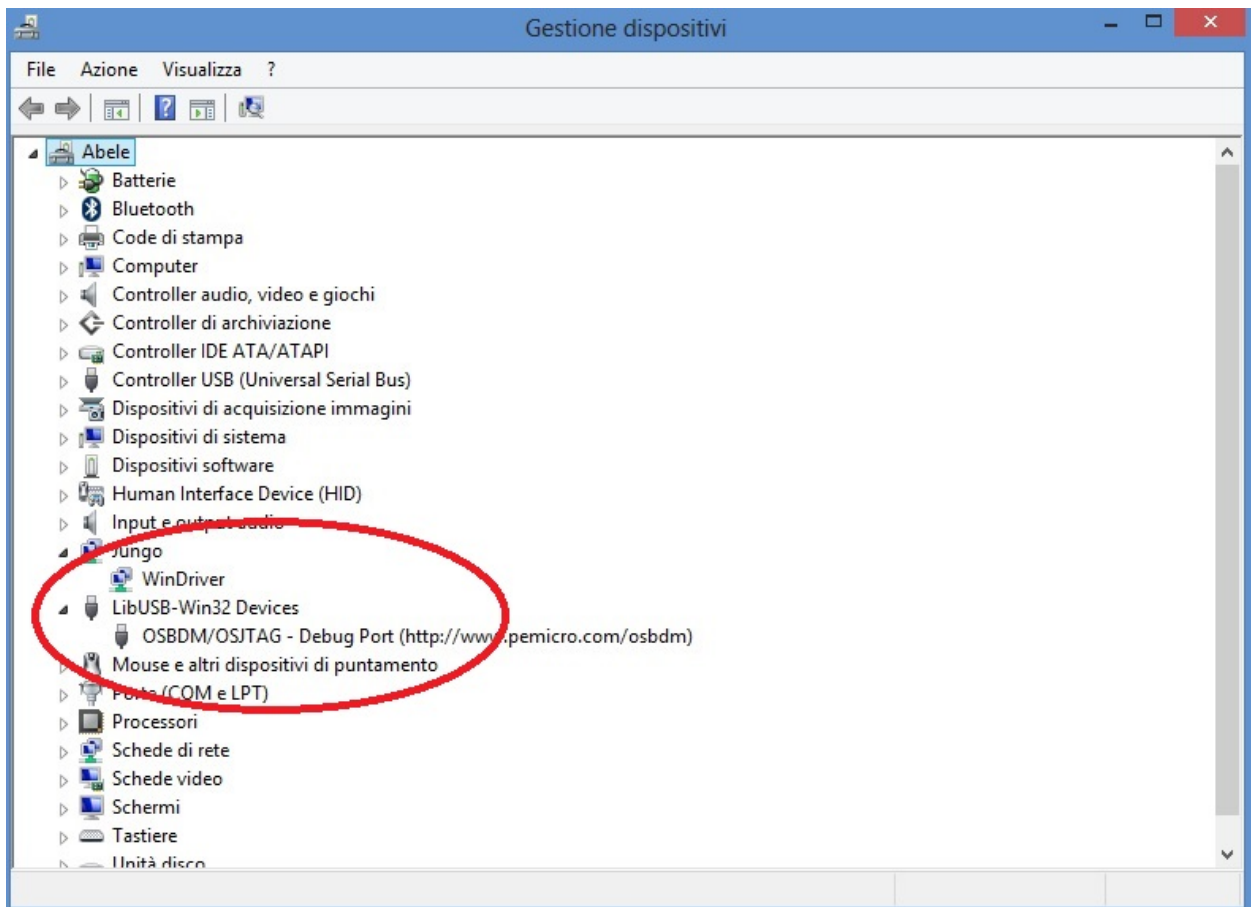
→ Be careful to see device reference next to connector. Each connector is designed for one or more devices and

will only accept dedicated modules.

- Plug Mini USB type-B cable into Cpu Board plug and connect to PC with Codewarrior. TWR power led will on.

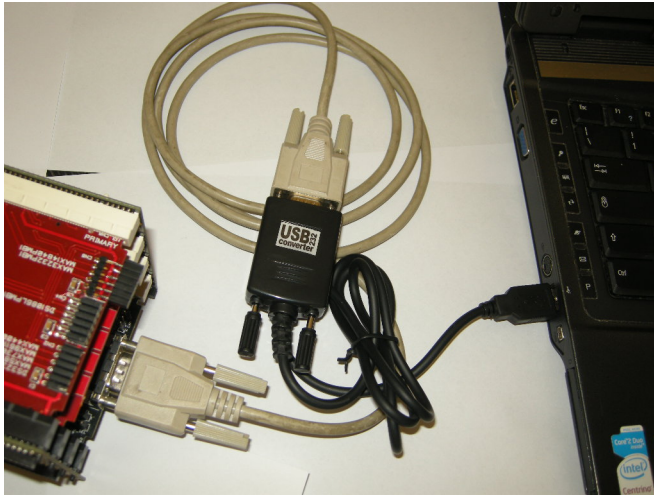


- If you see device tab, you will find OSBDM/OSJTAG debug port. If it doesn't occur, **unplug USB cable, start Codewarrior and then plug USB cable**. Tower system will be found and OSBDM/OSJTAG driver will be loaded. Close Codewarrior suite.



- plug the standard serial DB9 cable into serial connector on Tower System

- connect serial cable to terminal PC (equipped with terminal SW)



- On your terminal PC setup COMx parameter:

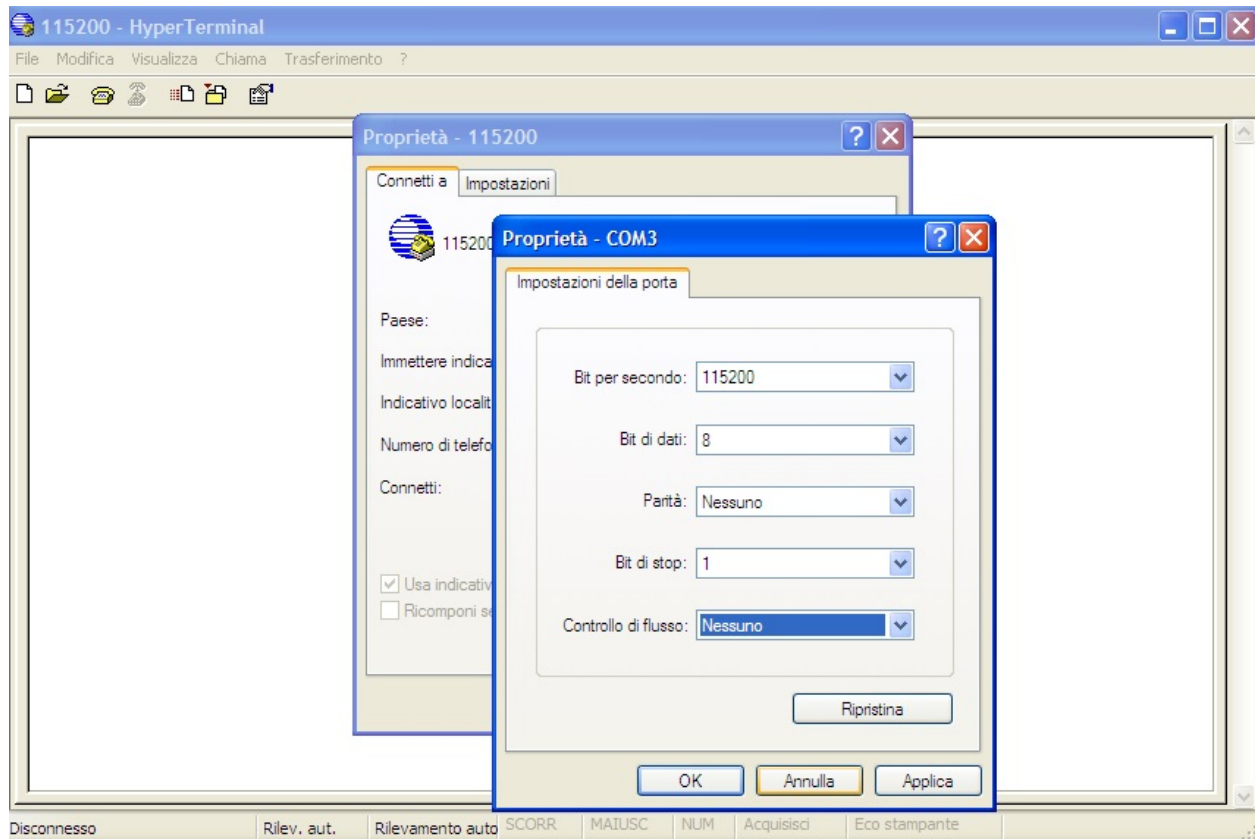
speed = 115200 baud

data with = 8

parity = none

stop bit = 1

flow control = none

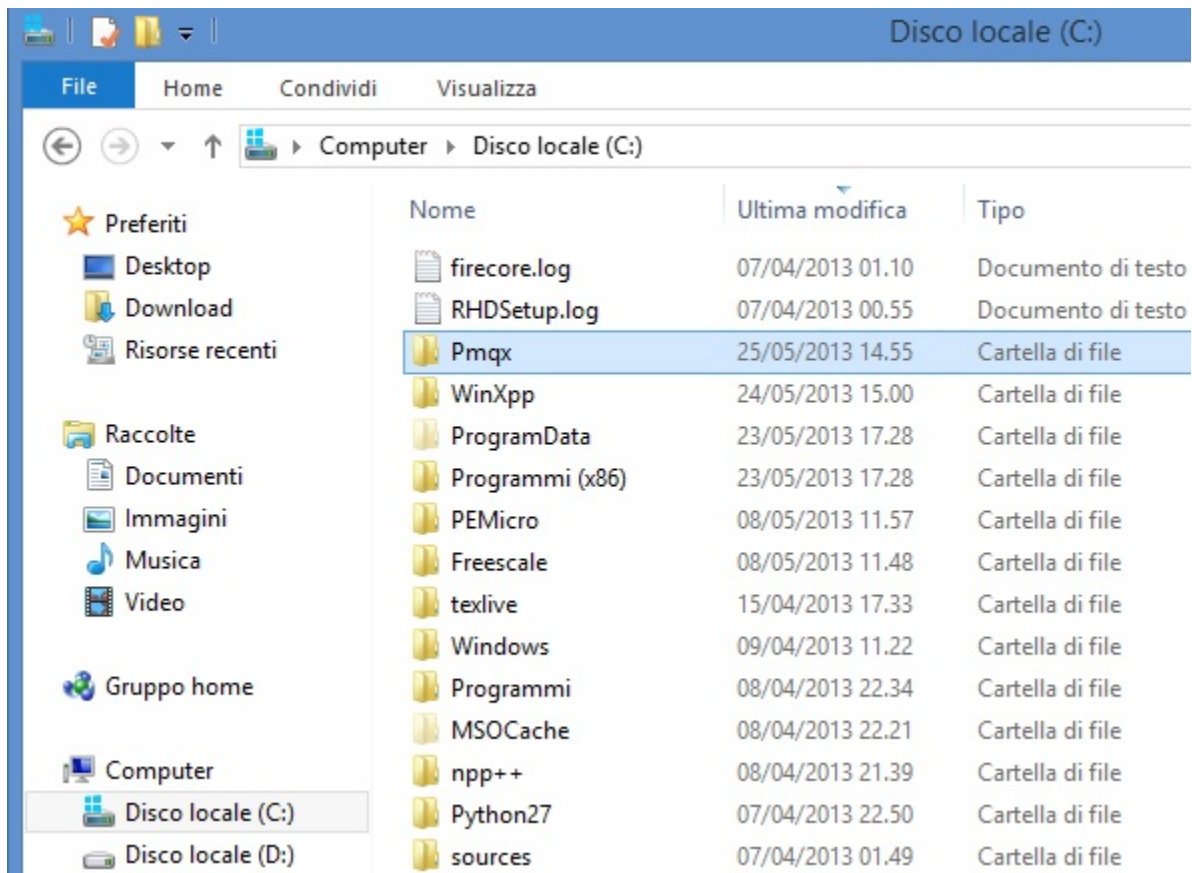


Now you are ready for install FW project.

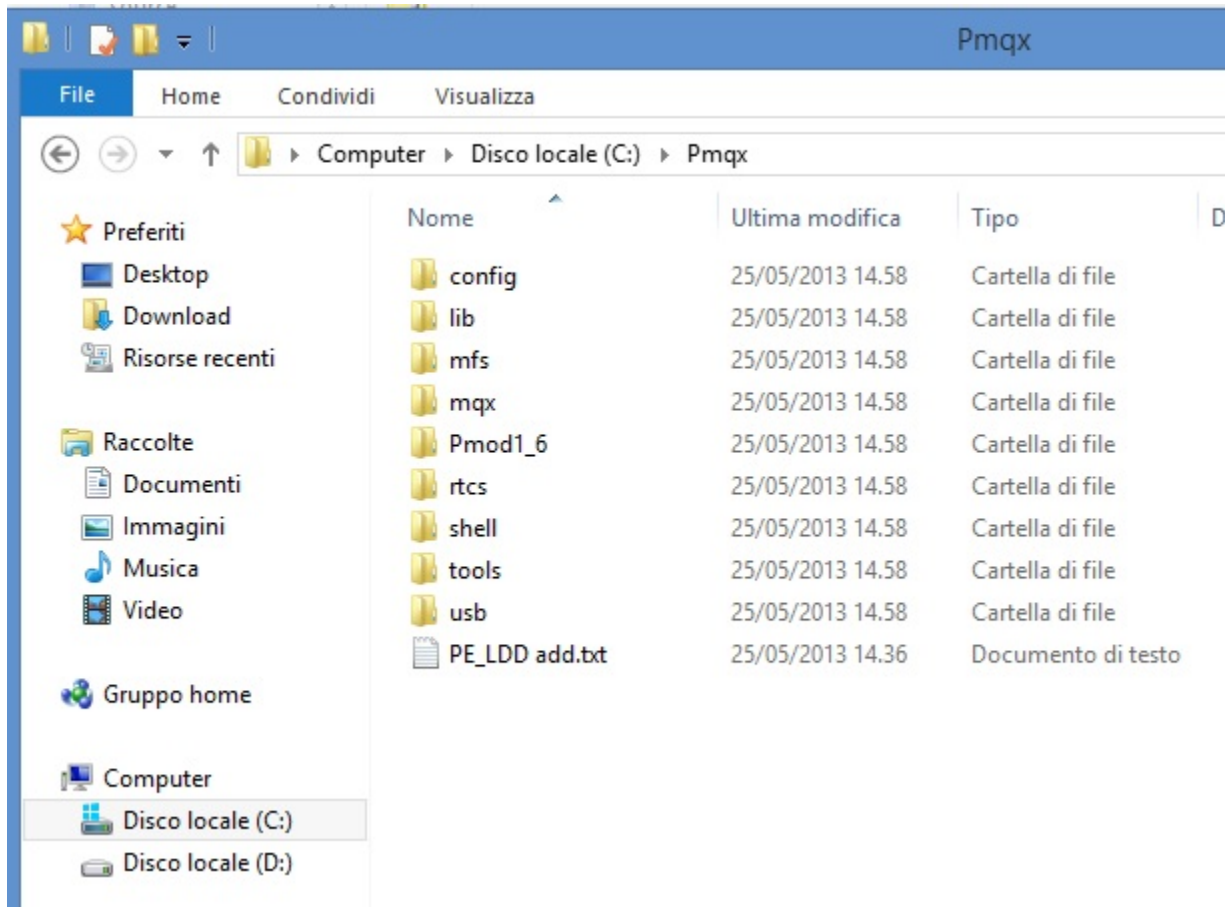
Brooklyn Board MQX FW setup

- In the root of your HDD C:\ create new folder named “**Pmqx**”.

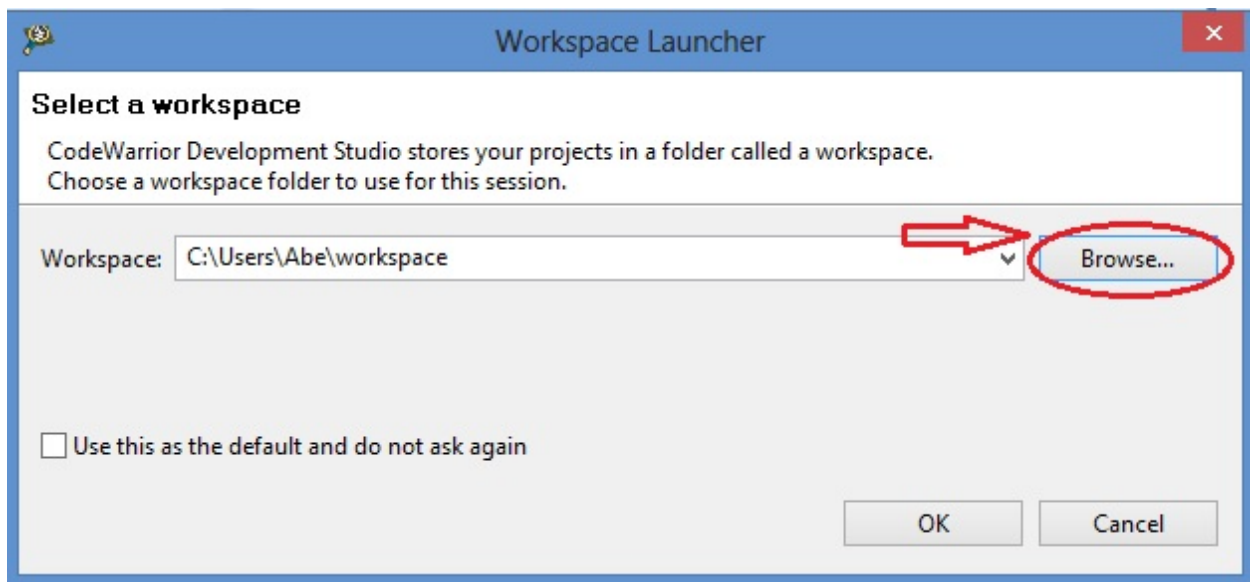
Important: Pay attention that the pathname is case-sensitive



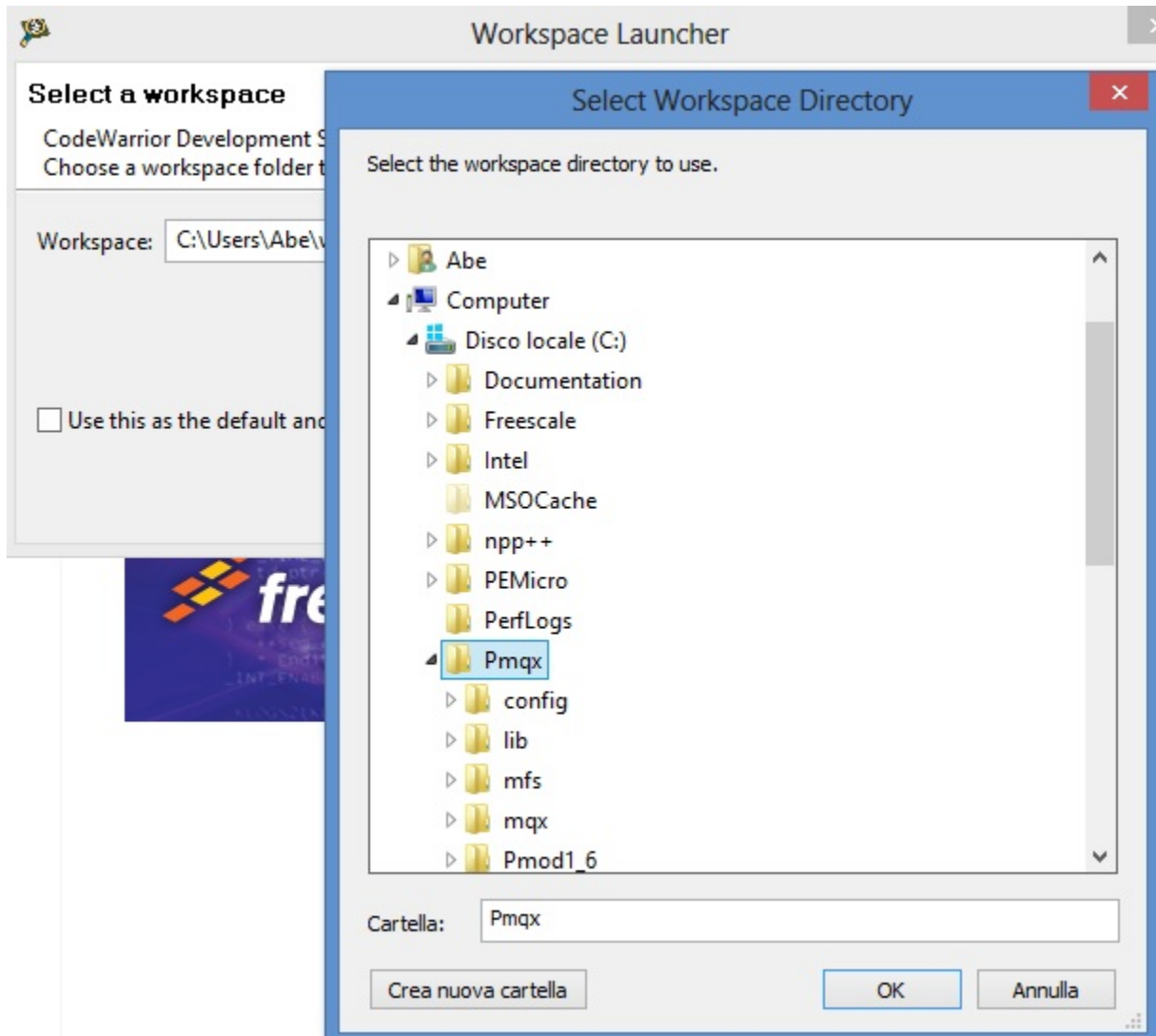
- Unzip all files from Pmod_MQX.zip into the folder C:\Pmqx just created



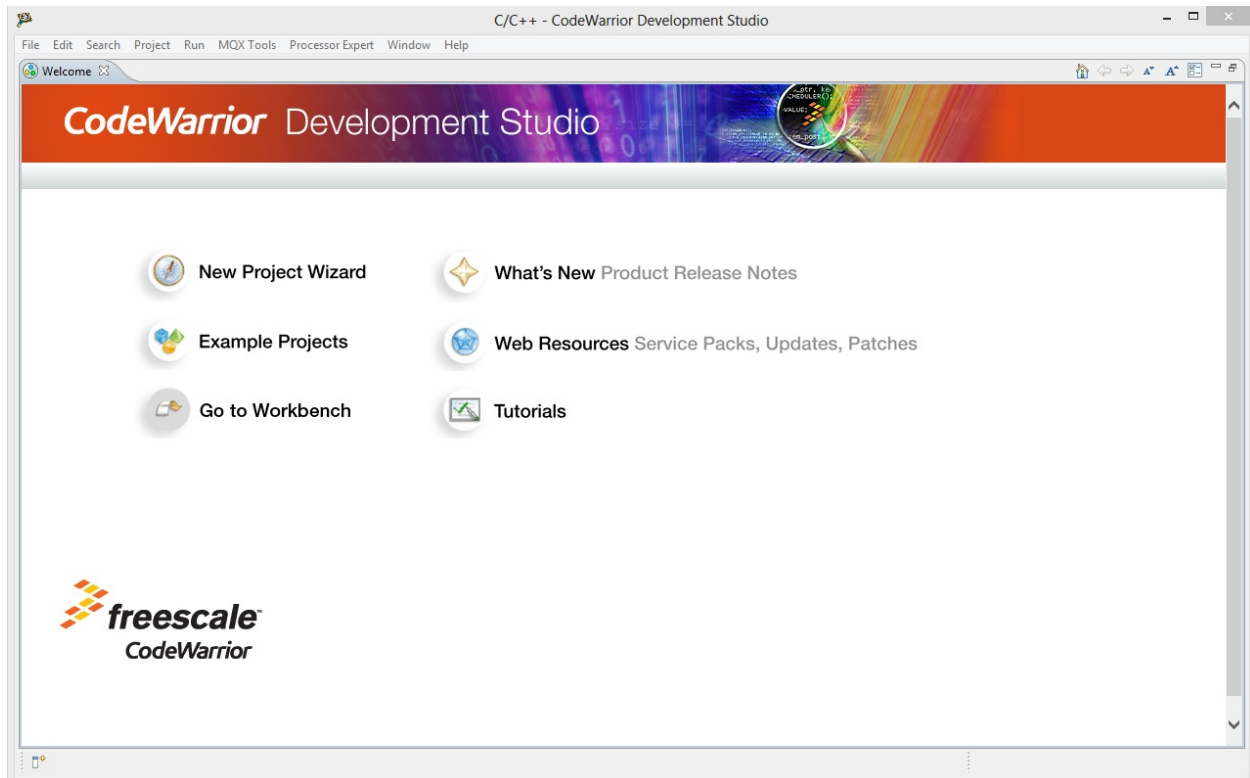
- If you have already opened Codewarrior, exit and restart it. The “Workspace launcher” tab will open. Click on “brouse” button.



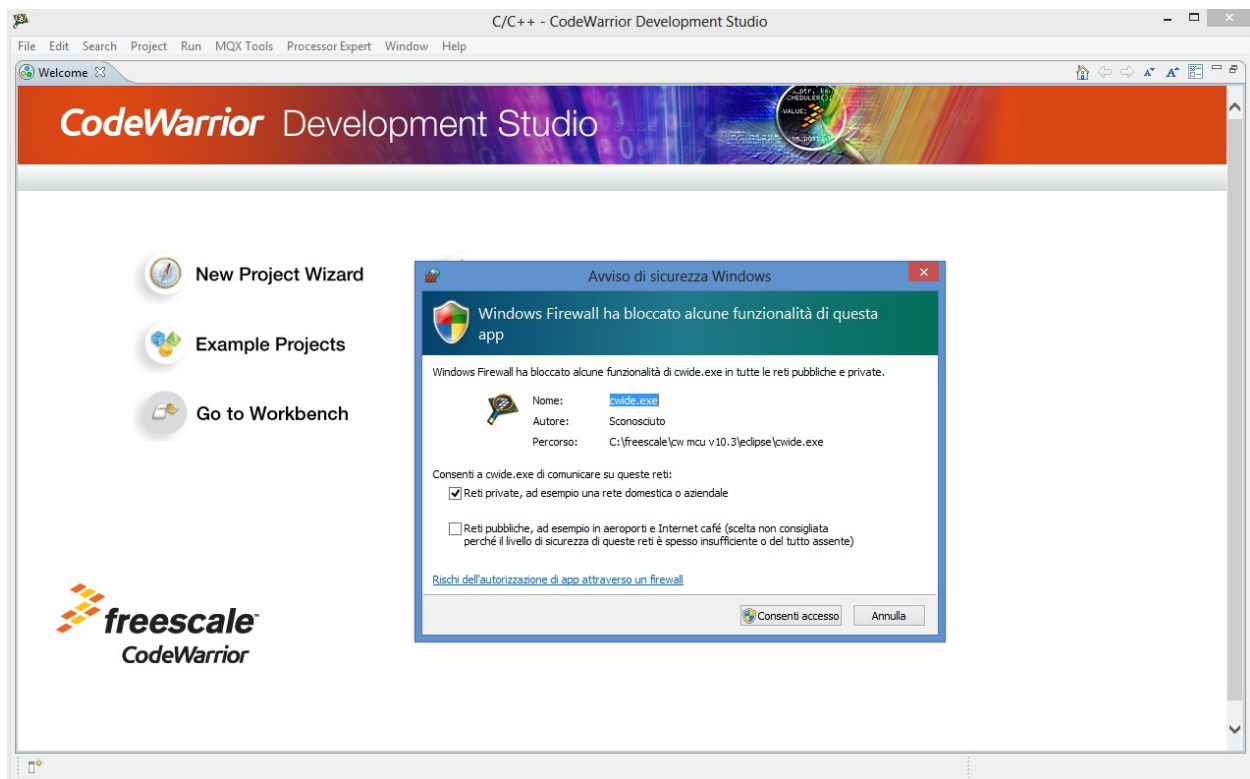
- navigate and select **C:\Pmqx** as workspace, then click OK



Now we could see the welcome window of Codewarrior Development Suite

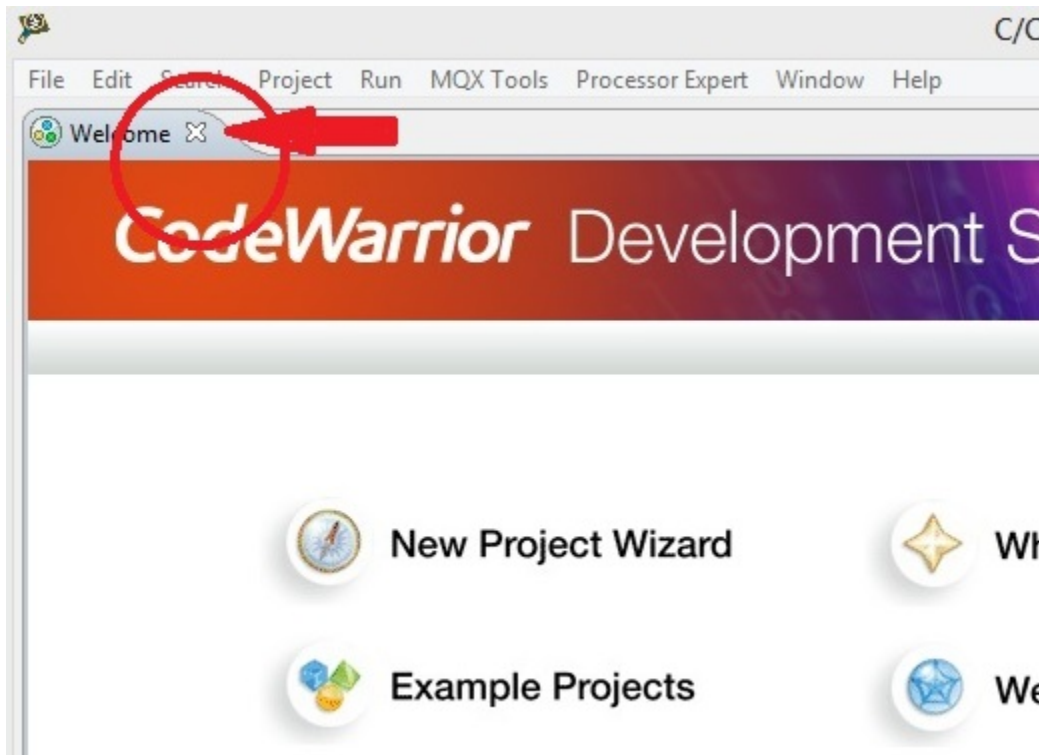


maybe will open firewall popup as below

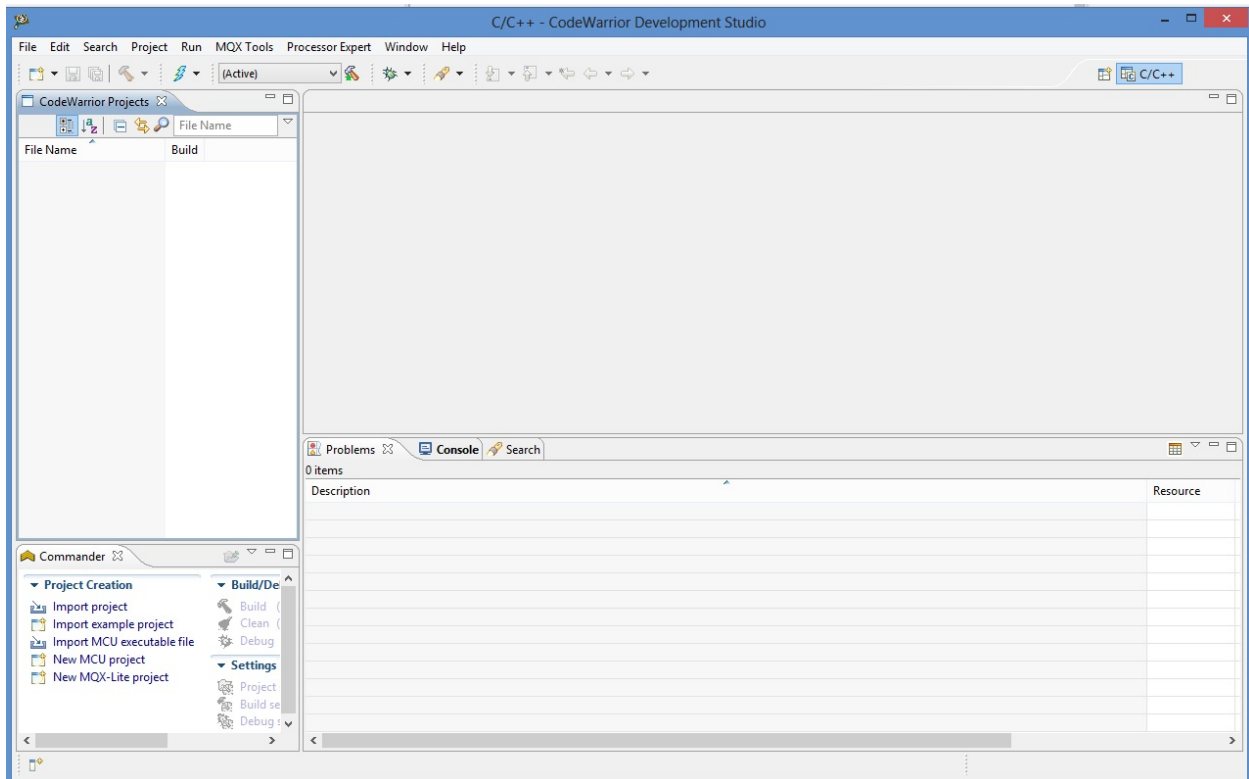


if yes, left-click on **enable access** and proceed

- close the welcome window by clicking 'X' in the Welcome tab



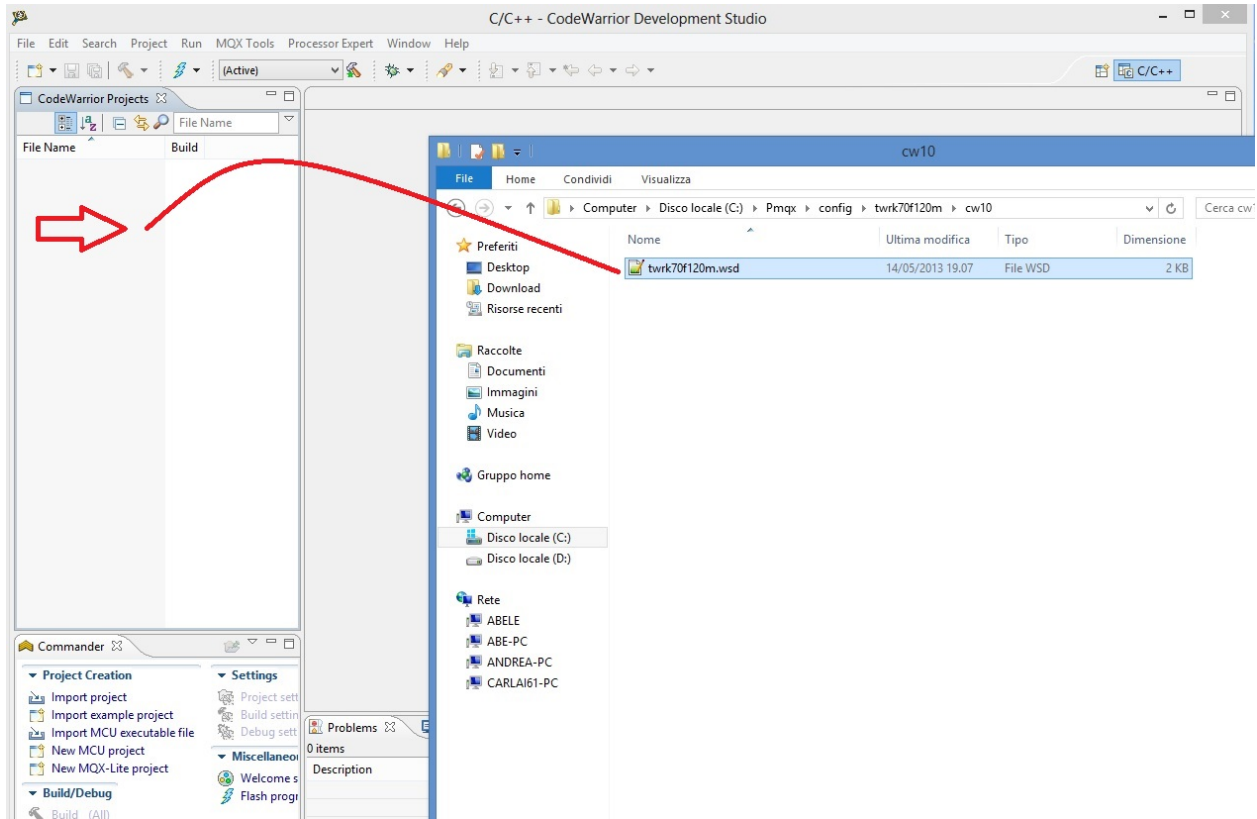
Now we can see the Codewarrior main window



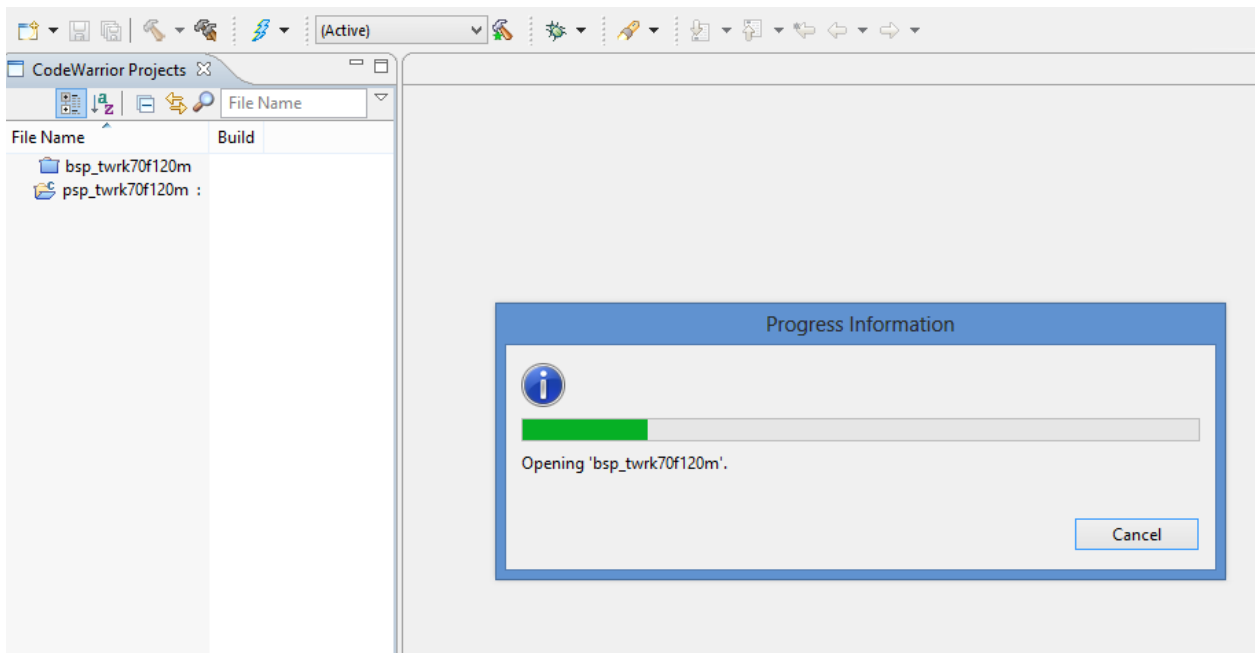
Codewarrior is ready to setup MQX project

IMPORTING AND BUILDING MQX LIBRARY

- Navigate to `C:\Pmqx\config\twrk70f120m\cw10` and drag the file `twrk70f120m.wsd` into Codewarrior Project window



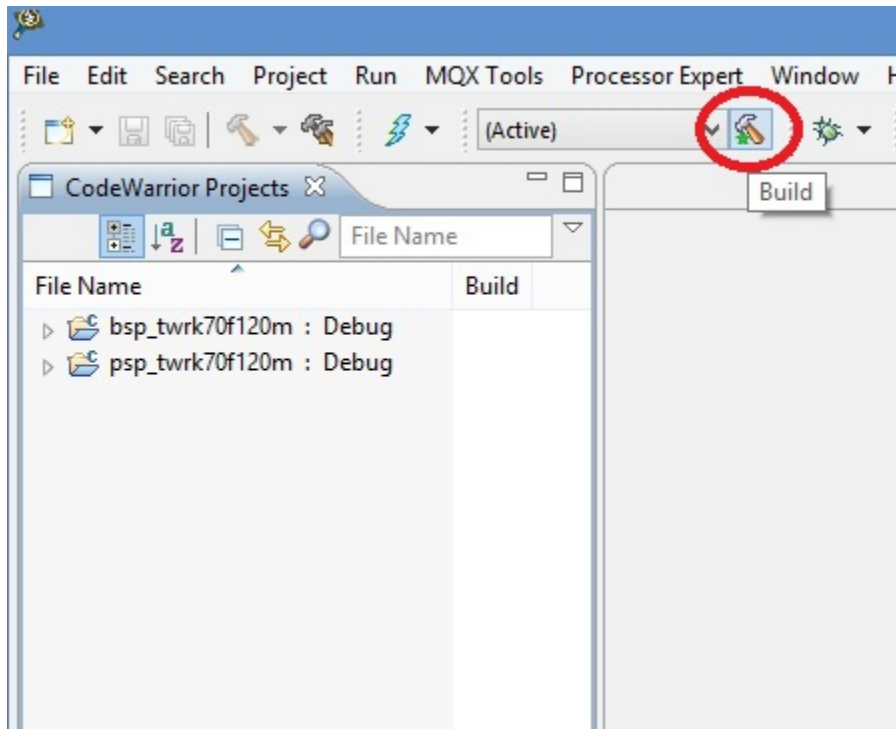
BSP and PSP library (needed for this project) will be loaded automatically



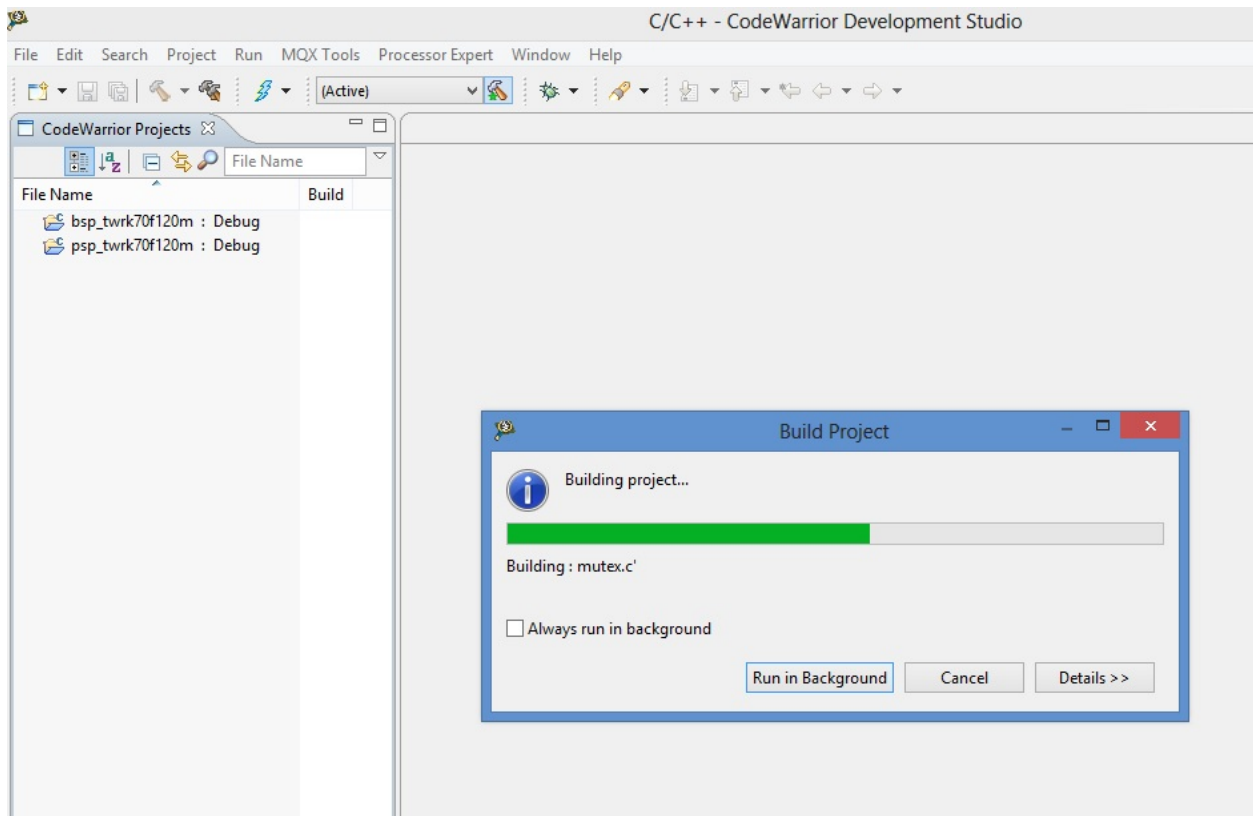
Now you can see library in Project Window.

BUILDING MQX LIBRARY

- Click on hammer icon (with green star) red-circled as in figure below

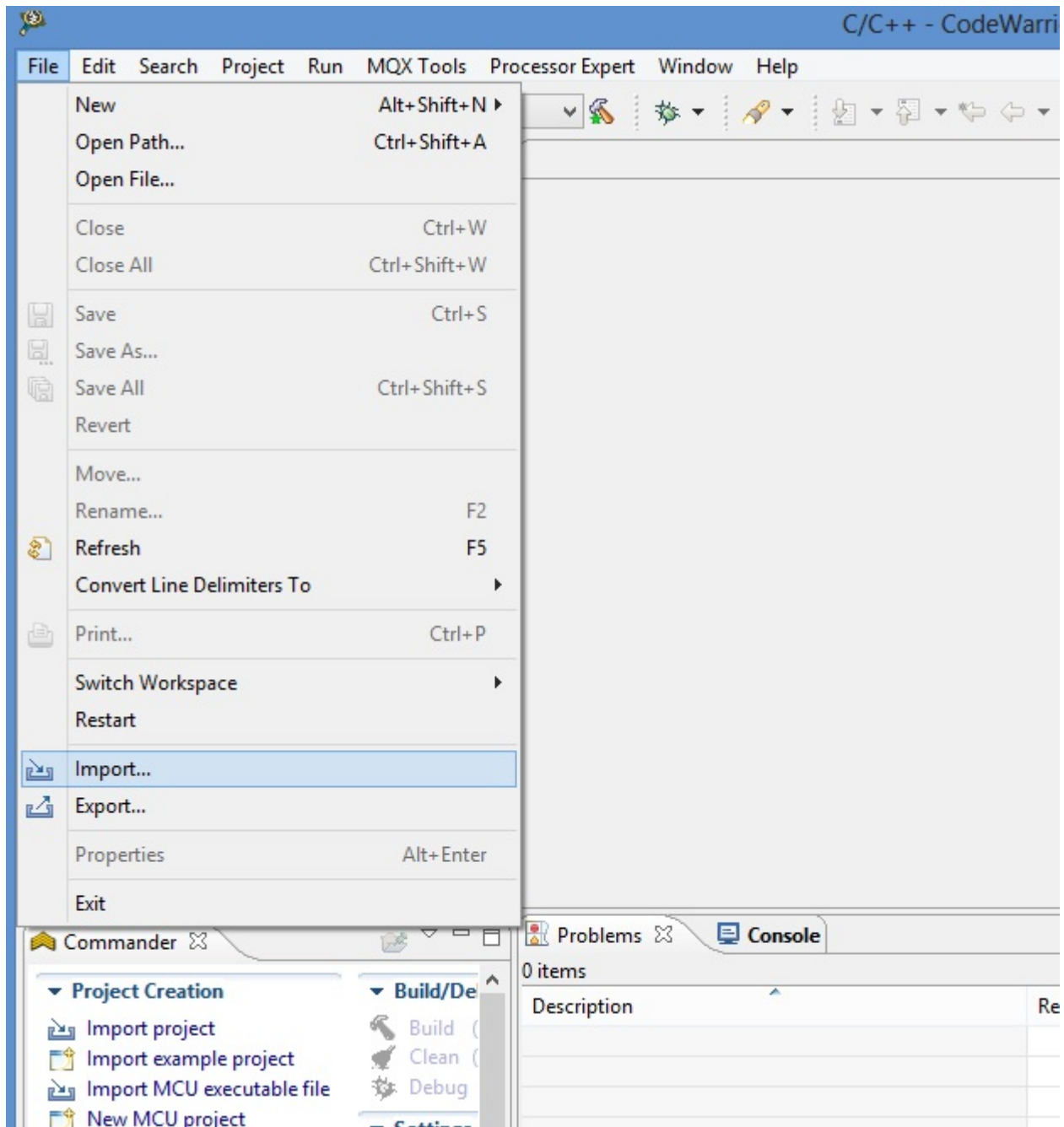


- BSP and PSP library are now ready for the project

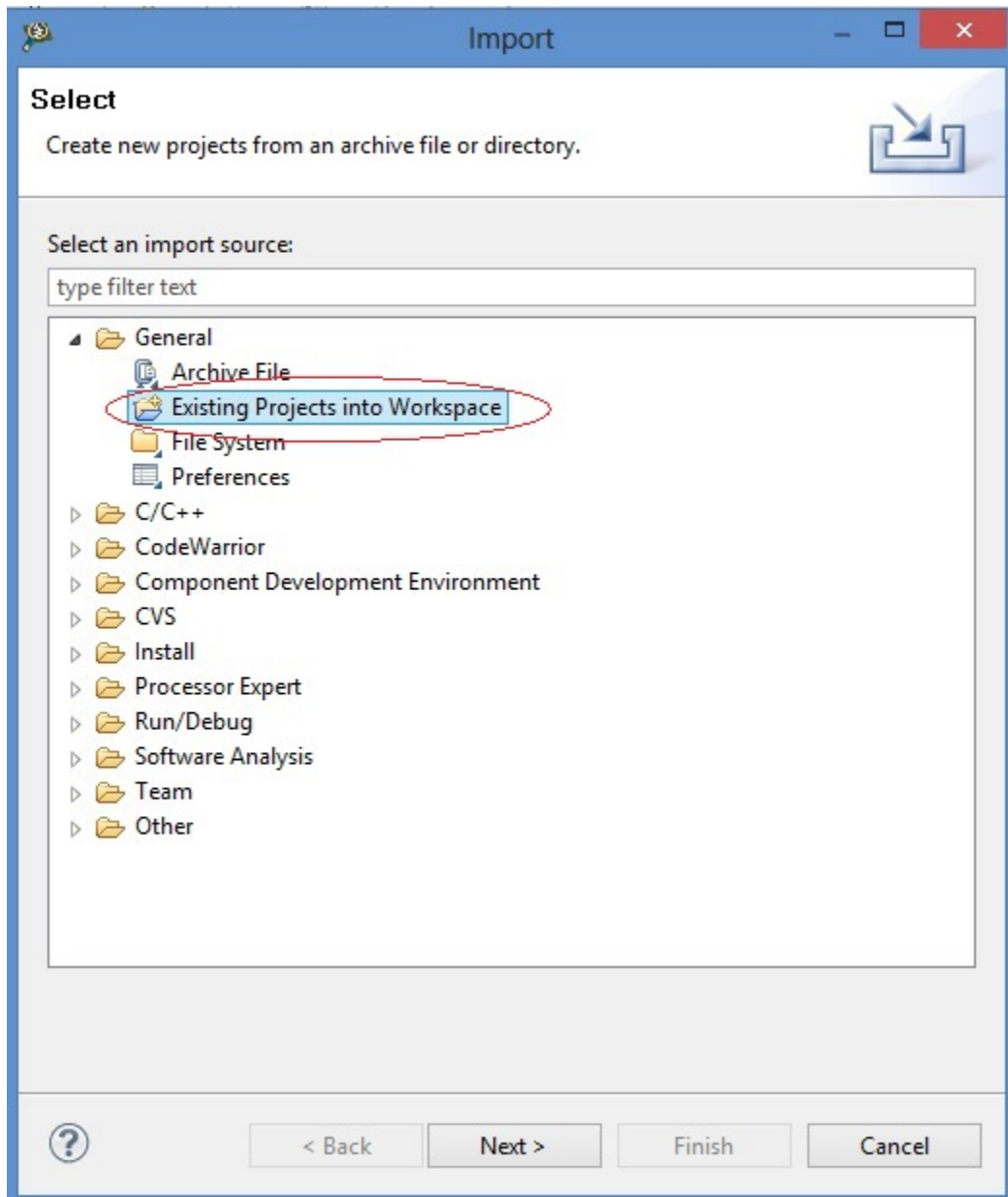


IMPORTING Pmod1_6 FIRMWARE

- Select File -> Import and click

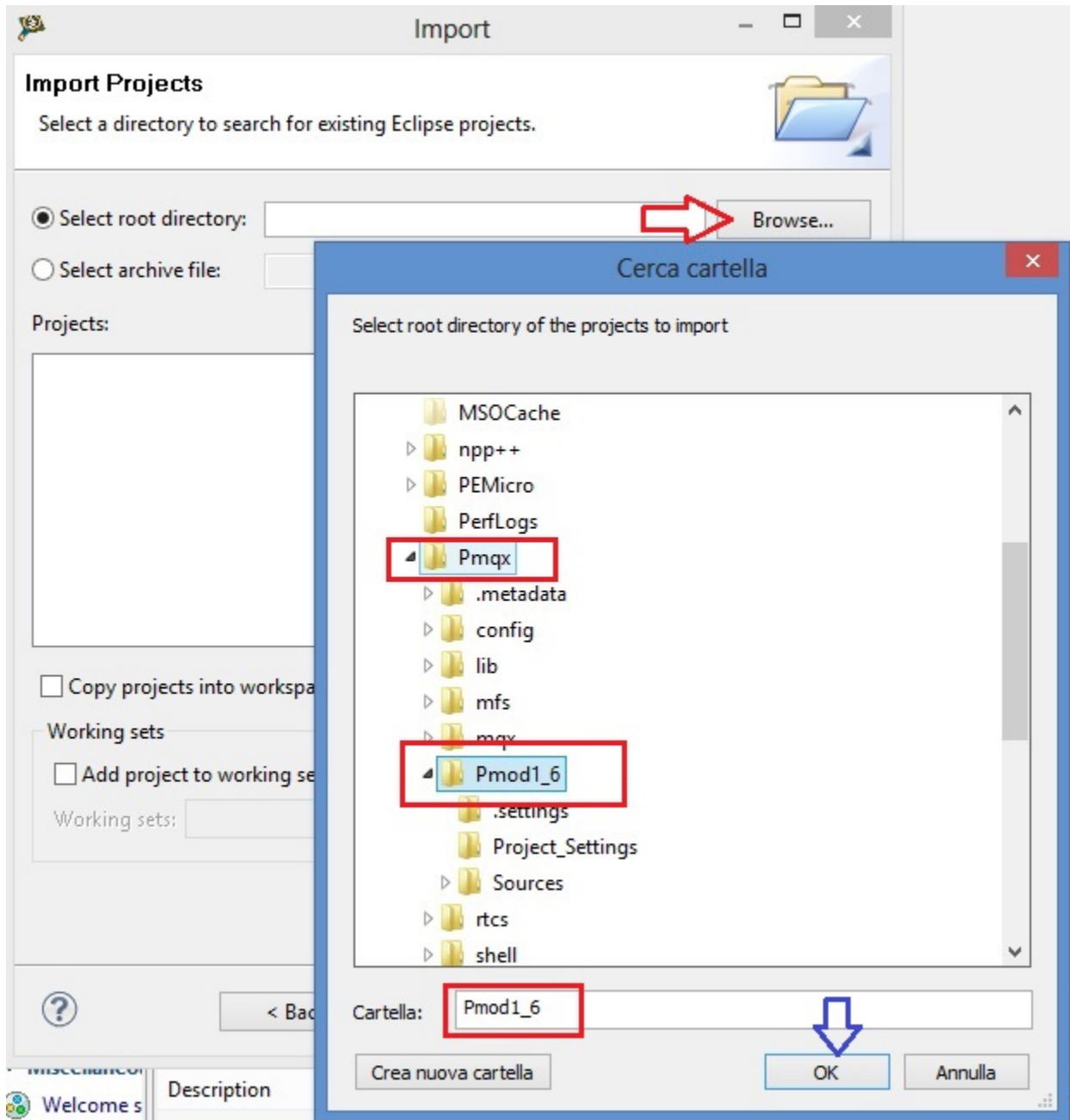


- in the next tab select “Existing Project into Workspace” and click “NEXT”

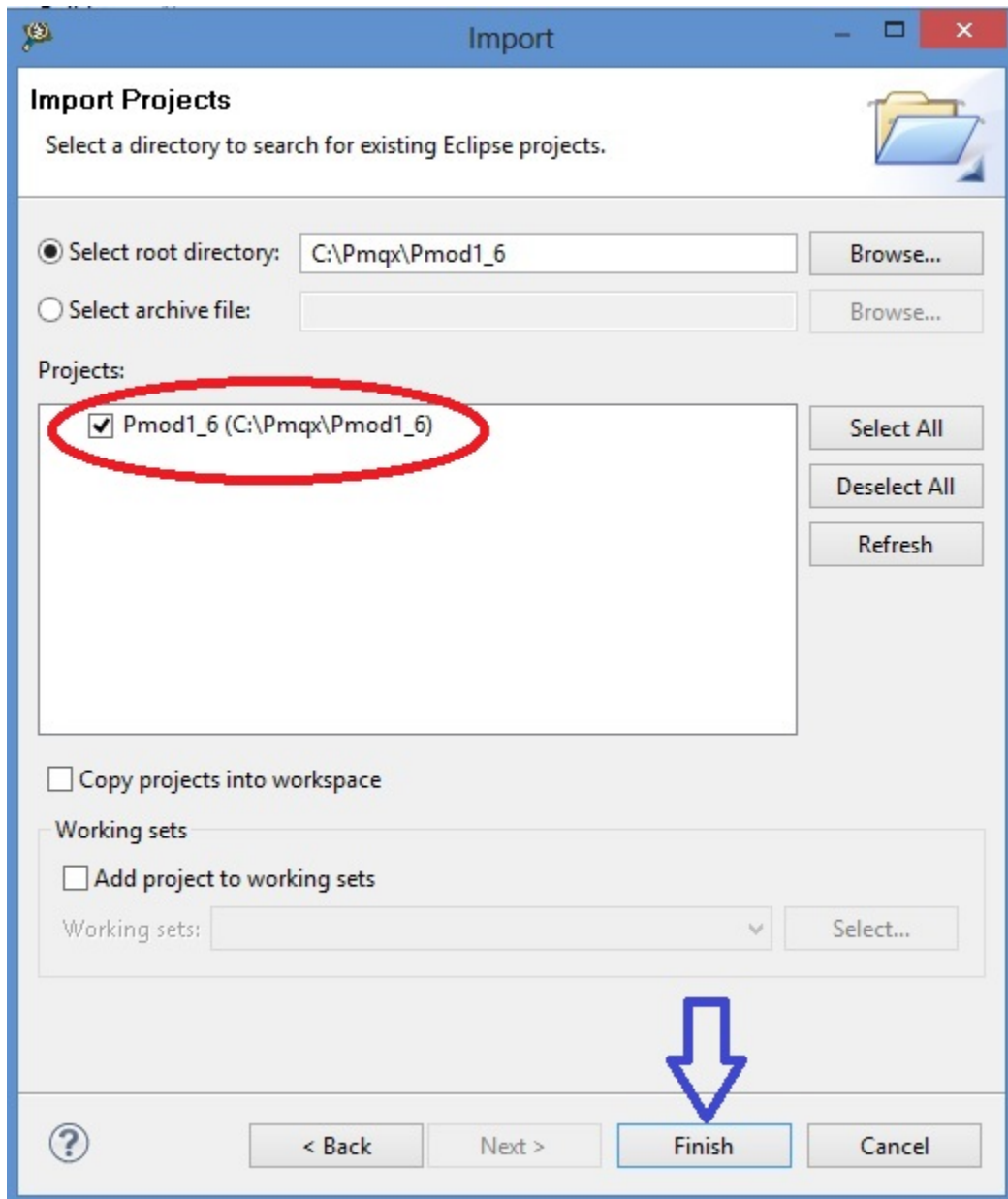


- in the next window make the following step

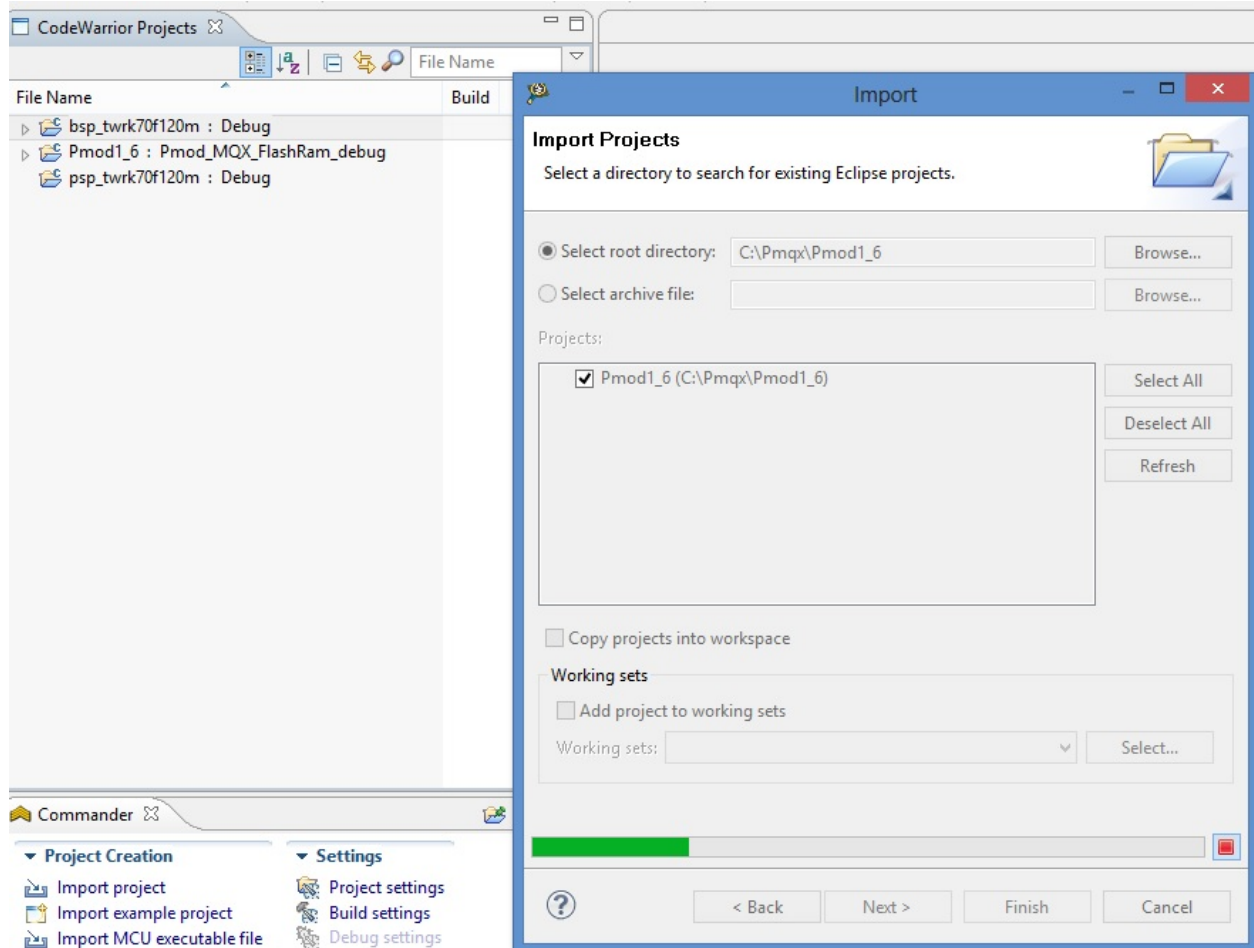
- 1 - click on **Browse** button.
- 2 - select folder "C:\Pmqx\Pmod1_6" as below.
- 3 - click on OK button.



- select checkbox “Pmod1_6(C:\Pmqx\Pmod1_6)”
- click “Finish”

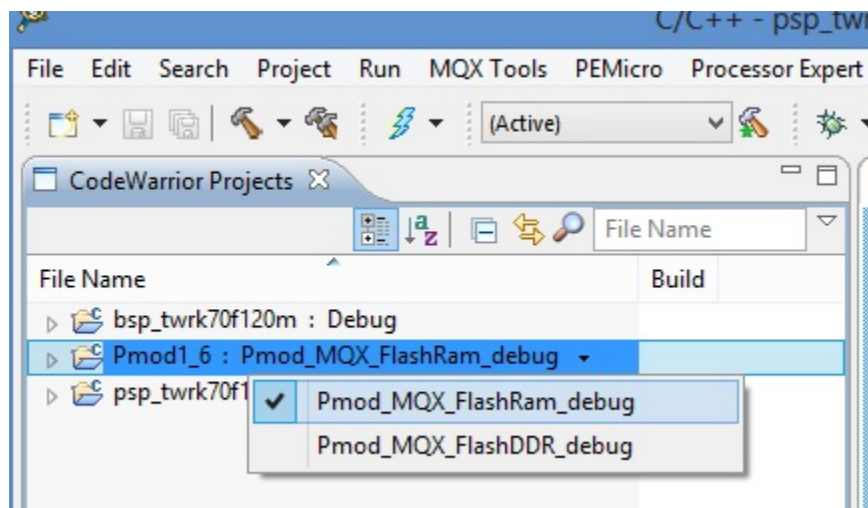


The firmware Pmod1_6 will be imported

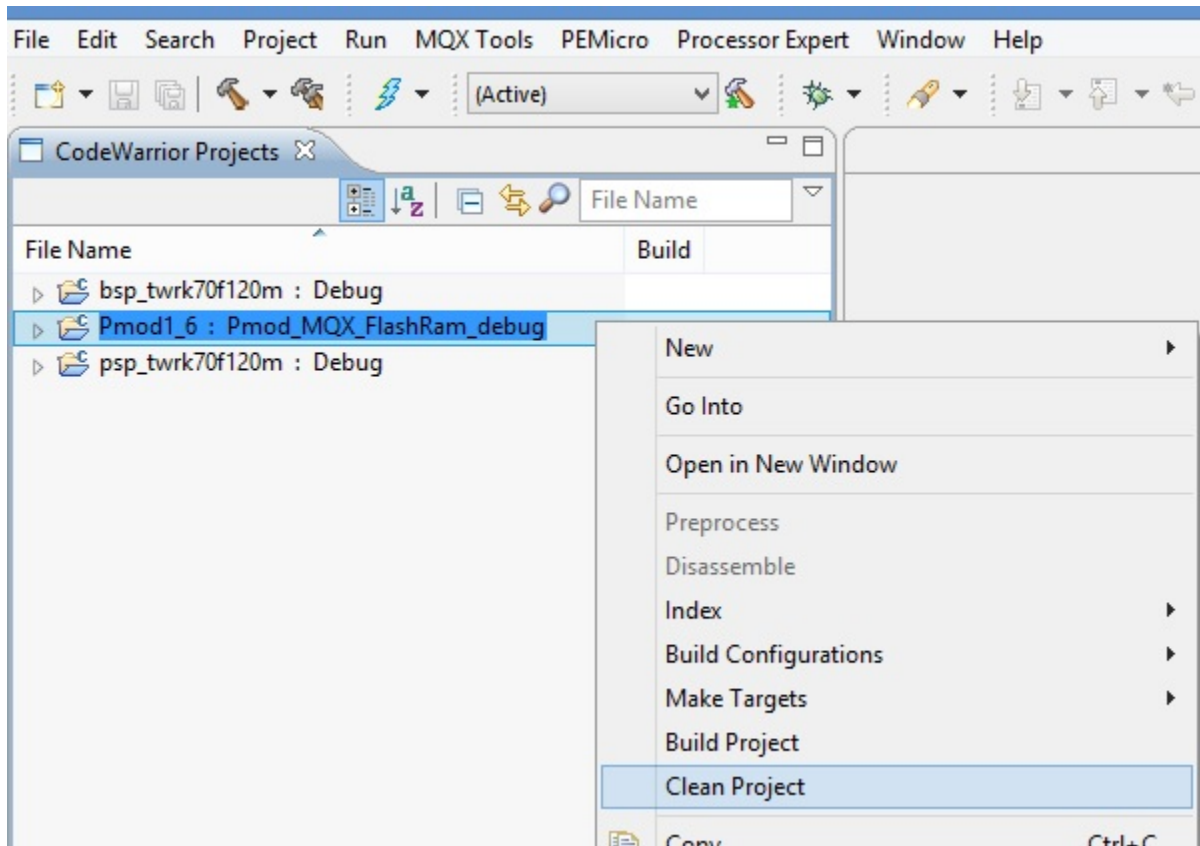


BUILDING Pmod1_6 FIRMWARE

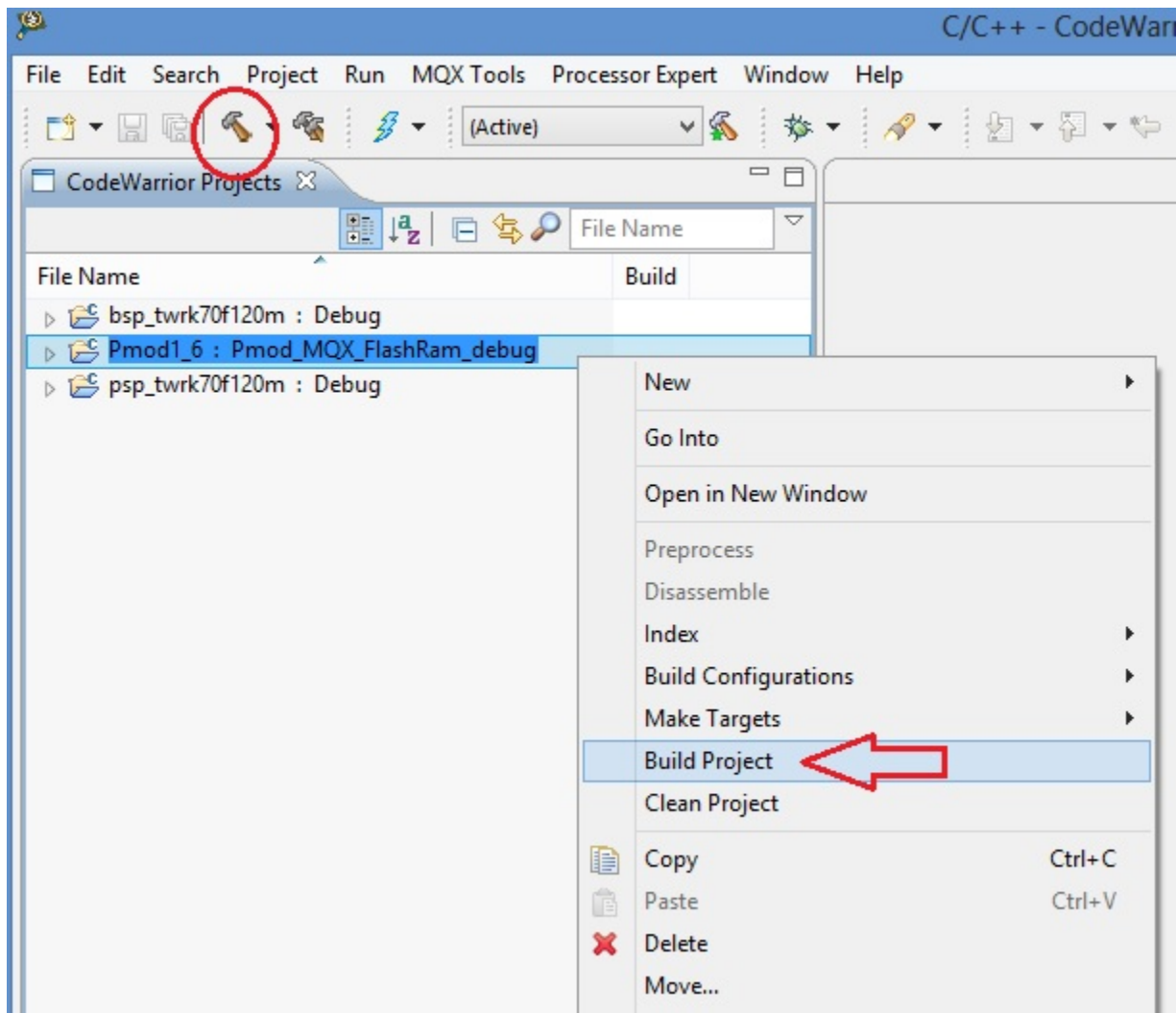
- see Codewarrior Project tab and select the project “Pmod1_6”, move mouse cursor on the right side an arrow will appear. Click on arrow and a popup menu will open. Check “Pmod_MQX_FlashRam_debug” and click over



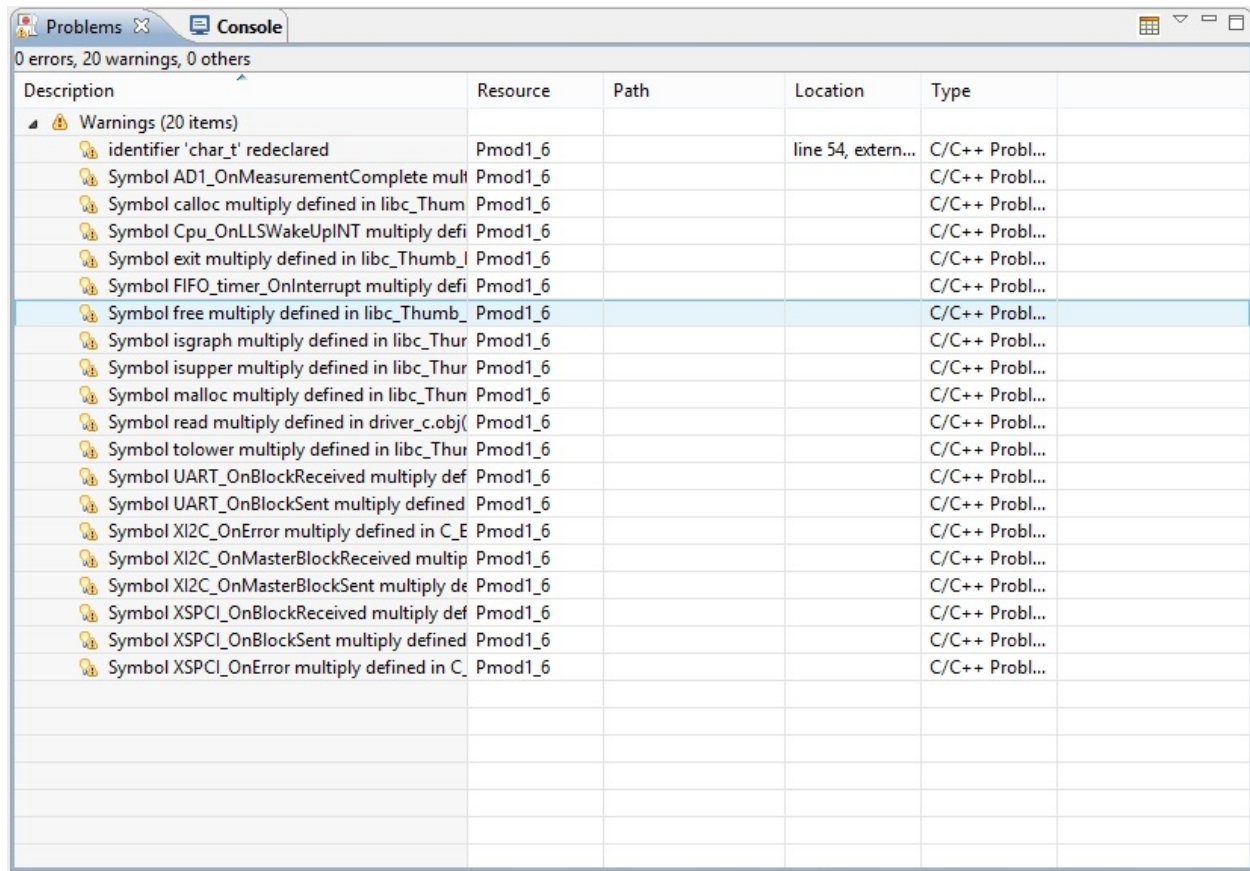
- select “Pmod1_6” project and right click over, select “Clean Project” and click



when process finish, click on single-hammer icon or right-click over the project and select “build” to build entire project

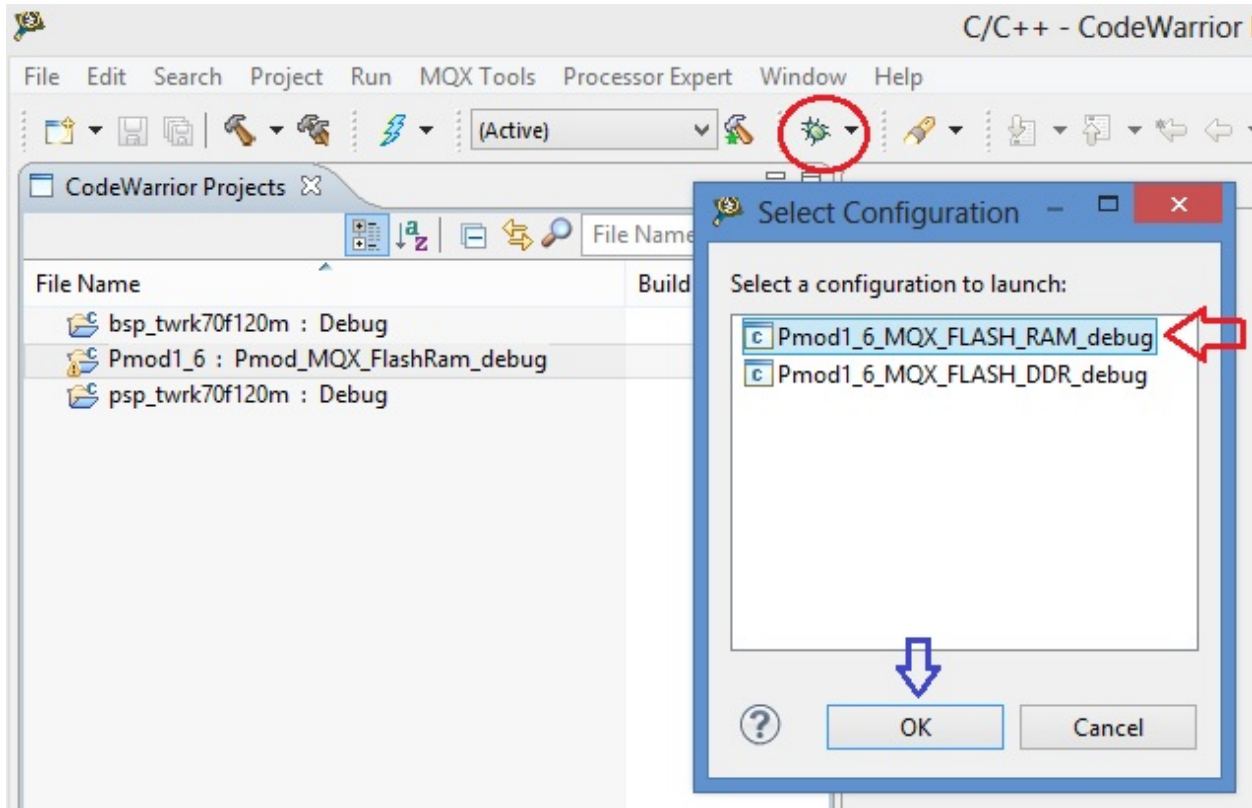


See the “problems” tab. There are 20 warnings because of file C_Events.c has same function as Events.c in BSP library. This is normally in MQX and doesn’t make trouble

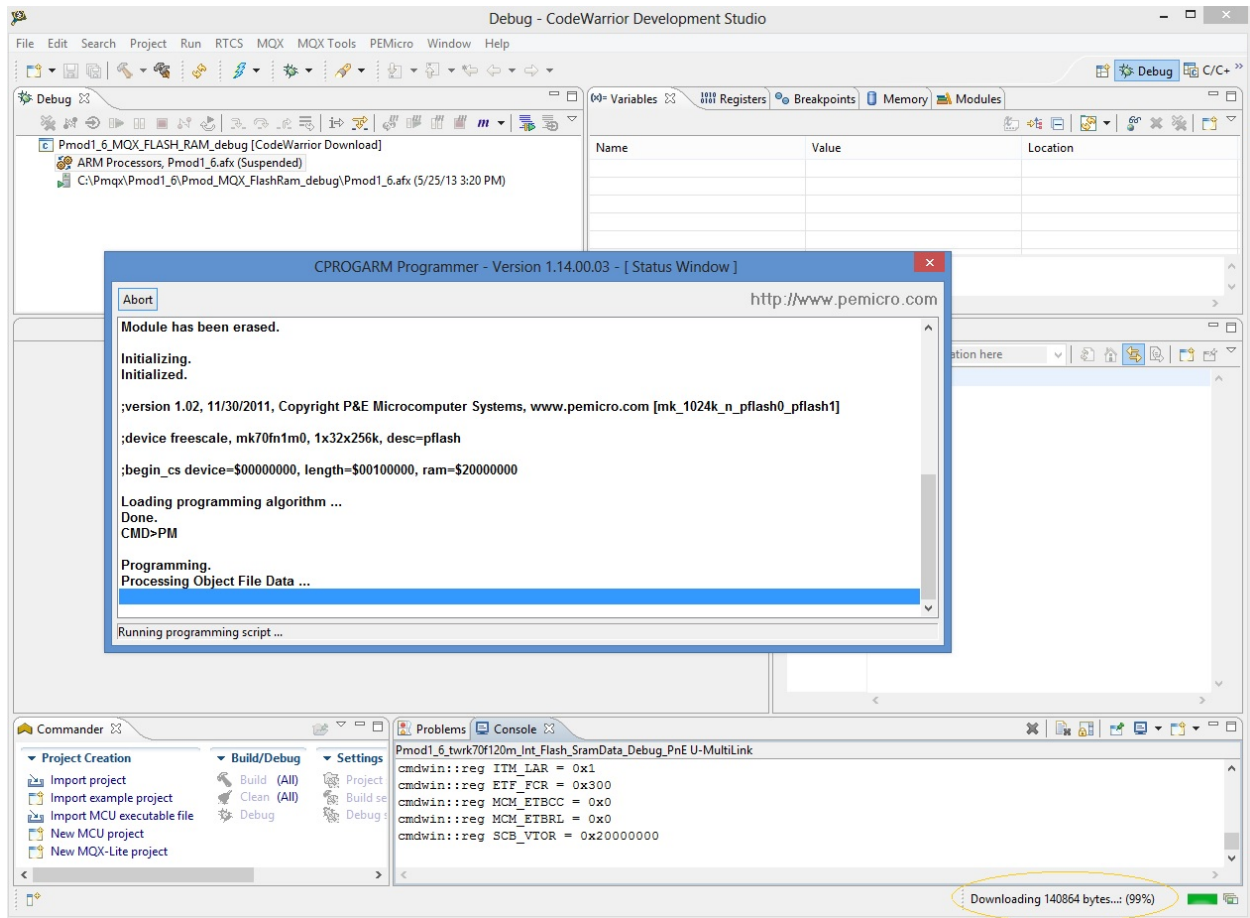


Description	Resource	Path	Location	Type
Warnings (20 items)				
identifier 'char_t' redeclared	Pmod1_6		line 54, extern...	C/C++ Probl...
Symbol AD1_OnMeasurementComplete multi	Pmod1_6			C/C++ Probl...
Symbol calloc multiply defined in libc_Thum	Pmod1_6			C/C++ Probl...
Symbol Cpu_OnLLSWakeUpINT multiply defi	Pmod1_6			C/C++ Probl...
Symbol exit multiply defined in libc_Thumb_I	Pmod1_6			C/C++ Probl...
Symbol FIFO_timer_OnInterrupt multiply defi	Pmod1_6			C/C++ Probl...
Symbol free multiply defined in libc_Thumb_	Pmod1_6			C/C++ Probl...
Symbol isgraph multiply defined in libc_Thur	Pmod1_6			C/C++ Probl...
Symbol isupper multiply defined in libc_Thur	Pmod1_6			C/C++ Probl...
Symbol malloc multiply defined in libc_Thun	Pmod1_6			C/C++ Probl...
Symbol read multiply defined in driver_c.obj(Pmod1_6			C/C++ Probl...
Symbol tolower multiply defined in libc_Thur	Pmod1_6			C/C++ Probl...
Symbol UART_OnBlockReceived multiply def	Pmod1_6			C/C++ Probl...
Symbol UART_OnBlockSent multiply defined	Pmod1_6			C/C++ Probl...
Symbol XI2C_OnError multiply defined in C_E	Pmod1_6			C/C++ Probl...
Symbol XI2C_OnMasterBlockReceived multip	Pmod1_6			C/C++ Probl...
Symbol XI2C_OnMasterBlockSent multiply de	Pmod1_6			C/C++ Probl...
Symbol XSPCI_OnBlockReceived multiply def	Pmod1_6			C/C++ Probl...
Symbol XSPCI_OnBlockSent multiply defined	Pmod1_6			C/C++ Probl...
Symbol XSPCI_OnError multiply defined in C_	Pmod1_6			C/C++ Probl...

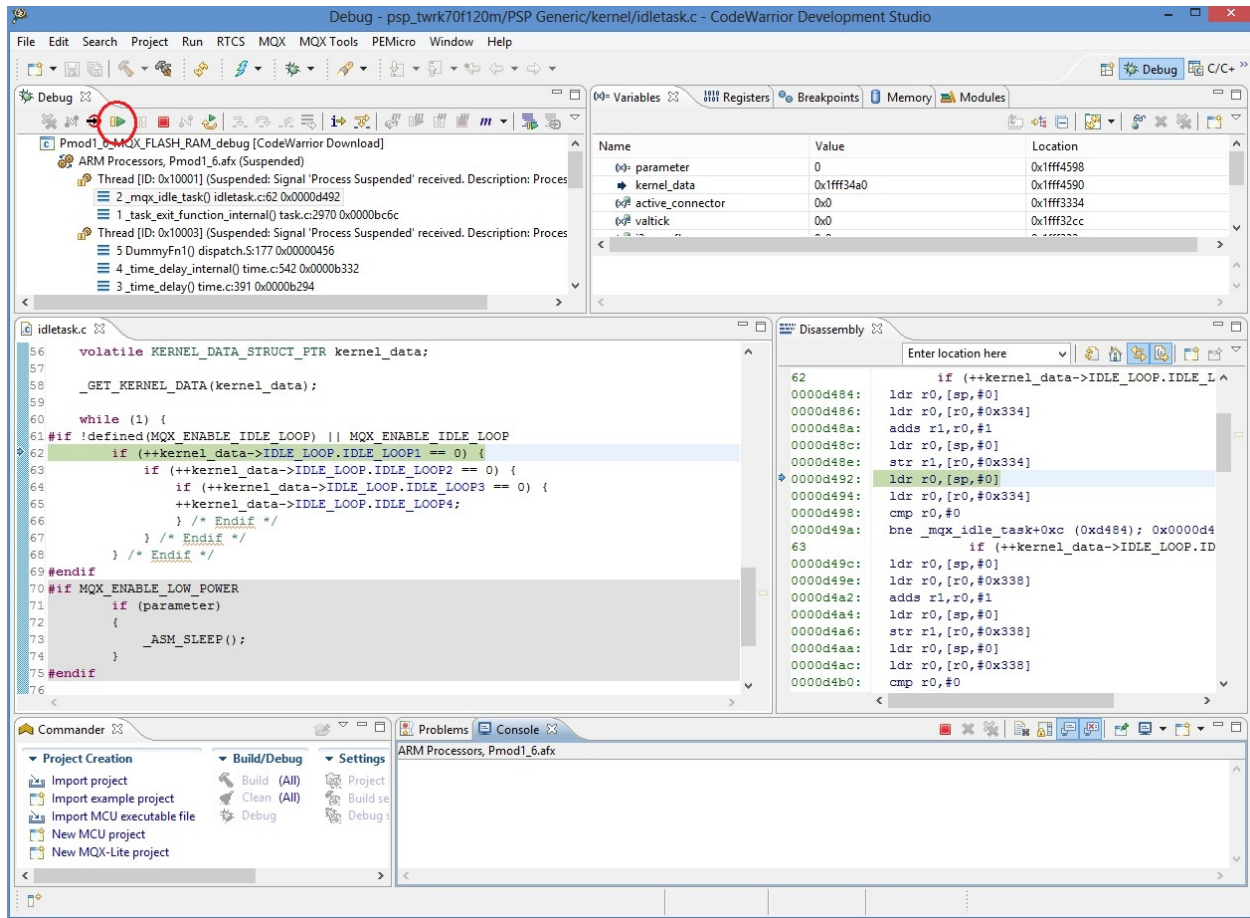
Click on **bug** icon red-circled. Popup menu will open. Select “Pmod1_6_MQX_FLASH_RAM_debug” and click



- Click again “bug” icon and debug will start with firmware download
during firmware download this tab will open



and when download finish you see the main debug windows of Codewarrior



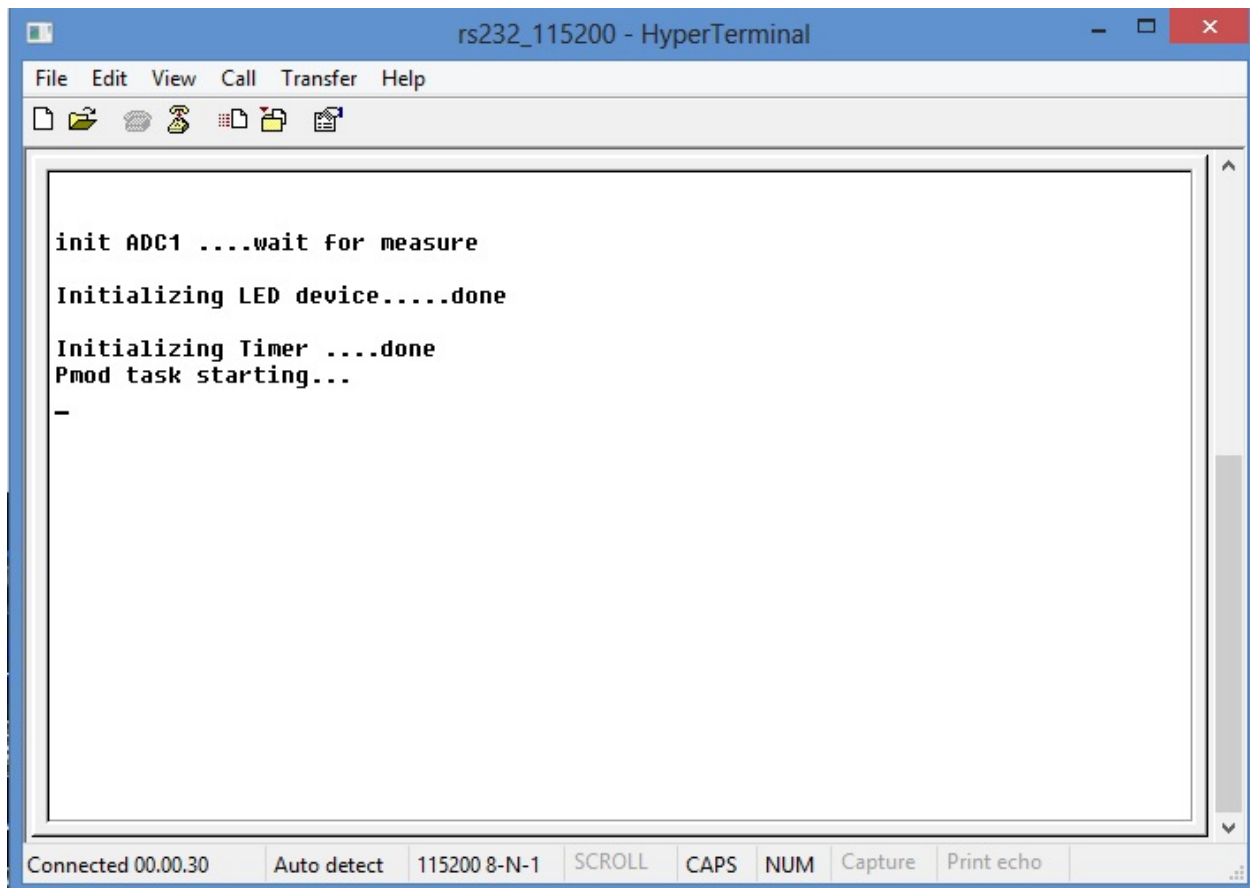
to start program you can press “F8” or click on Icon red-circled in image above

NOTE: for full Codewarrior functionality please refer to Freescale Official Guide

download here [Codewarrior Guide](#)

Running Brooklyn Board MQX FW

When you start program, in terminal window you can see for few seconds this screen



```
init ADC1 .....wait for measure
Initializing LED device.....done
Initializing Timer .....done
Pmod task starting...
-
```

this screen will inform about task init and task start

then, you see the Pmod start screen

2.8. Running Brooklyn Board MQX FW


```

rs232_115200 - HyperTerminal
File Edit View Call Transfer Help

//      # # # # # # # # # #      //
//      # # # # # # # # # #      //
//      # # # # # # # # # #      //
#####

Active PMOD Port = A
Press a key/number to test a Peripheral Module:
{0} DS1086L   Spread Spectrum Econ Oscillator
{1} DS3231M   I2C Real-Time Clock
{2} MAX3232   RS-232 Transceiver
{3} MAX4824   Octal Relay Driver
{4} MAX5216   SPI-Compatible, High performance 16 bit DAC
{5} MAX5487   Dual, 256-tap SPI Digital Potentiometer
{6} MAX5825   I2C Octal DAC
{7} MAX7304   I2C-Interfaced 16-Port, GPIO and LED Driver
{8} MAX9611   Current sense amplifier with OpAmp and ADC
{9} MAX11205  16 bit Delta-Sigma ADC with 2-Wire interface
{A} MAX14840  RS-485 Transceiver
{B} MAX31723  MAX31723 Digital Thermometer
{C} MAX31855  Thermocouple to Digital Converter
{D} MAX44000  I2C-Interfaced Ambient Light and Proximity Sensor
{E} PMOD      Increment Active PMOD

>> _

Connected 00.01.17  Auto detect  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo

```

Now select device menu (typing selection key in the terminal window) and follow menu option to test device.

It's strongly recommended to change or insert Pmod Modules when Tower System is off (without power).

Then, turn off the power by disconnecting the Mini USB B-type cable, remove device (if present) and insert new module in properly connector.

Turn on the power by plug the Mini USB B-type cable. The program will restart. No reload from debug console is needed. Follow same steps used before to test new device

Brookling Board firmware comes from original Maxim Maxim Zenboard Platform project revision 1.6, by using the file listed below.

Main project files from Maxim

- MaximPmod.c
- menu.c.
- maximDeviceSpecificUtilities.c
- platform.c
- utilities.c

and related include files

- MaximPmod.h
- menu.h
- maximDeviceSpecificUtilities.h
- platform.h
- utilities.h
- platform_config.h

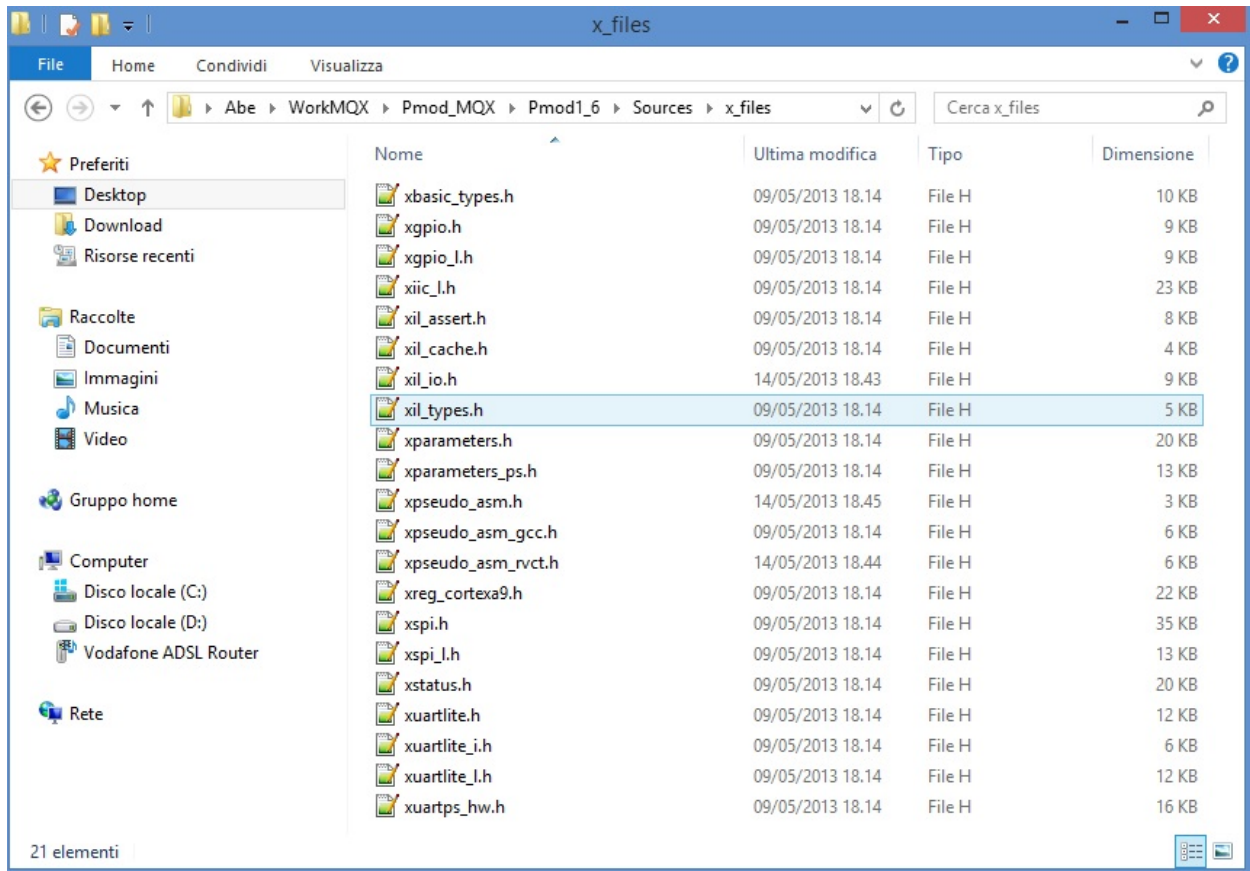
You can find all this file in the “Source” folder of the project

General include files

- xbasic_types.h

- xgpio.h
- xgpio_1.h
- xiic_1.h
- xil_assert.h
- xil_cache.h
- xil_io.h
- xil_types.h
- xparameters.h
- xparameters_ps.h
- xpseudo_asm.h
- xpseudo_asm_gcc.h
- xpseudo_asm_rctv.h
- xreg_cortex9.h
- xspi.h
- xspi_i.h
- xspi_1.h
- xstatus.h
- xuartlite.h
- xuartlite_i.h
- xuartlite_1.h
- xuartps_hw.h

You can find all this file in the “Source\x_files” folder of the project



Main Project files added

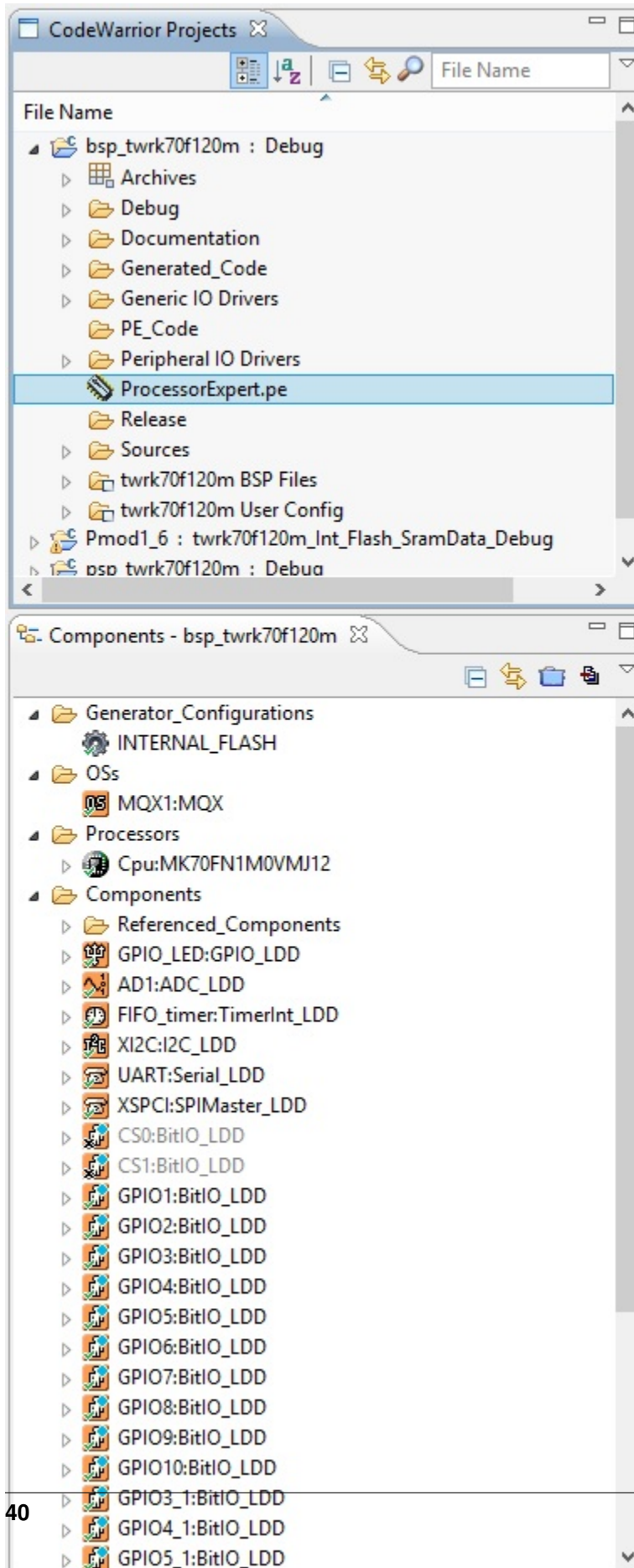
In source folder you find application specific files:

- main.c (MQX main function including task declaration and function)
- C_Events.c (ISR events function)
- driver.c (low-level function replacement)

and include files

- main.h (MQX main include)
- MaxFuncRedefinition.h (start menu function redefinition)
- def.h (general purpose definition)

This project is developed under Codewarrior 10.3 using **Processor Expert tools**. In the BSP project you can find folder “Generated_Code” witch contains files generated by Processor Expert. Opening BSP project you can see Processor Expert settings. It’s strongly recommended to make no changes in the configurations of the Processor Expert components, because of some ajustement needed in file after code generation. A detail of this changes is available on request.



MQX tasks brief

In file main.c there are 4 tasks:

Pmod_task: invokes init timer and main_pmod()

Led_task: yellow led blink (on tower cpu board)

PmodLed_task: orange led blink during Pmod_task active

adc_task: a task that read from ADC converter the value of potentiometer R52 and start/stop blue led blink.

```

1  /*****
2  *
3  *   This file contains MQX only stationery code.
4  *
5  *****/
6  #include "main.h"
7
8  #include "UART_PDD.H"
9  #include "GPIO_LED.h"
10 #include "def.h"
11
12
13 #if !BSPCFG_ENABLE_IO_SUBSYSTEM
14 #error This application requires BSPCFG_ENABLE_IO_SUBSYSTEM defined non-zero in user_config.h. Ple
15 #endif
16
17
18 #ifndef BSP_DEFAULT_IO_CHANNEL_DEFINED
19 #error This application requires BSP_DEFAULT_IO_CHANNEL to be not NULL. Please set corresponding I
20 #endif
21
22
23 TASK_TEMPLATE_STRUCT MQX_template_list[] =
24 {
25 /*   Task number,      Entry point,      Stack, Pri,      String,      Auto? */
26 {PMOD_TASK,          Pmod_task,          4096, 10,          "main",       MQX_AUTO_START_TASK},
27 {LED_TASK,           Led_task,           1500, 9,            "led",        MQX_AUTO_START_TASK},
28 {PMODLED_TASK,       PmodLed_task,       1500, 8,            "pmodled",    MQX_AUTO_START_TASK},
29 {ADC_TASK,           adc_task,           1500, 8,            "adc",        MQX_AUTO_START_TASK},
30 {0,                  0,                  0,      0,              0,            0,
31 };
32
33 LDD_TDeviceData      *LED_DeviceData;
34 LDD_TError           LED_Error;
35 LDD_TDeviceData      *T1Ptr;
36
37 bool mPmod = FALSE;

```


CHAPTER 4

Firmware changes

1 - added include file **MaxFuncRedefinition.h** at the top of **MaximPmod.c** file. This file must be the first include in list.

```
#include "MaxFuncRedefinition.h" <----

#include <stdio.h>
#include "platform.h"
#include "menu.h"
#include "utilities.h"
#include "maximDeviceSpecificUtilities.h"
#include "maximPMod.h"

#define MAJOR_REVISION 1
#define MINOR_REVISION 6
```

2 - renamed **main()** function inside **MaximPmod.c** file with new name **main_pmod()**.

```
int main_pmod()          <----
/**
 * \brief      Main() function for Analog Essentials example program.
 * \par       Details
 *            This function sets up and initializes the FPGA and hardware, displays
↪the root menu via
 *            Hyperterminal, then dispatches inidividual demo programs for specific
↪module based on
 *            user's keypress selection.
 *
 * \param      None
 *
 * \retval     Always TRUE
 */
{
    // Variables for the main() function
    u8 uchInput=0;
    int nMenuState=0;
```

```
int i=0;
char tempString[256];
```

3 - Added redefinition of “printf” function inside include file **maximPMOD.h**. This is needed for build this project without any other changes to send functionality messages through Tower Expansion Board serial interface

```
#ifndef MAXIMPMOD_H_
#define MAXIMPMOD_H_

#include "xgpio.h"
#include "xgpio_1.h"
#include "xparameters.h"
#include "xuartlite.h"
#include "xspi_1.h"
#include "xspi.h"
#include "xiic_1.h"
#include "utilities.h"
#include <string.h>
#include <stdio.h>
#include "platform.h"
#include "math.h"

**#define printf _io_printf          <----**

#define DEFAULT_HYPERTERMINAL_UART_ID XPAR_PS7_UART_1_DEVICE_ID  //!< macro used to
↳abstract Physical Port of Hyperterminal UART
#define DEFAULT_HYPERTERMINAL_UART_ADDRESS XPAR_PS7_UART_1_BASEADDR
```

4 - commented function **led_knight_rider** inside **MaximPmod.c** file to obtain application fast start.

```
// Toggle the LEDs so that the user knows the board is awake
XGpio_Initialize(&g_xGpioLed, XPAR_AXI_GPIO_LED_DEVICE_ID);
XGpio_SetDataDirection(&g_xGpioLed, 1, 0x00000000); // Set the LED peripheral
↳to outputs
// led_knight_rider(&g_xGpioLed,2);          <----
```

5 - changed costant definition **ABOUT_ONE_SECOND** inside **MaximPmod.h** file as follow:

```
#define ABOUT_ONE_SECOND 74067512/8    <----
//#define ABOUT_ONE_SECOND 74067512      //!< approx 1 second delay when used as
↳argument with function delay(numberCyclesToDelay)
// Update this if uBlaze/Zynq CPU core frequency is changed, or if the external
↳memory timing changes.
// Although emprirically tested to 1.0000003 seconds, it is not meant to be used for
↳precise timing purposes
```

6 - improved definition inside “utilities.h”, row 103, as follow

```
void set_seven_segment_character(u8 uchDigitNumber, u8 uchValue, u8 uchDecimalActive);
void print_seven_segment_number(float fNumber);
extern struct maximDateTime *t; //NEEDED FOR MQX CORRECT BUILD <----
//struct maximDateTime *t;
void print_seven_segment_time(struct maximDateTime *t);
```

7 - changed triangle wave ramp value for MAX5216 (file menu.c, row 765 anf 769)

```
case 12:
    printf("Triangle Wave started\r\n");
```



```

fflush(stdout);
for(i=0;i<300;i++)
{
    for(j=0;j<65535;j=j+23)
    for(j=0;j<65535;j=j+230)        <-- new ramp_
    {
        max_MAX5216_set_output_voltage(g_
    }
    for(j=65535;j>=0;j=j-23)
    for(j=65535;j>=0;j=j-230)        <-- new ramp_
    {
        max_MAX5216_set_output_voltage(g_
    }
}
nMenuState = 0;
break;

```

NOTE: All these changes are tested on revision 1.6 of Maxim project files and need to be checked on further new revisions

At the date of issue of this firmware version there are some care in the use of Cadewarrior 10.3. If you would make changes in BSP or PSP library or in Processor Expert components, please take care the following remarks. We know that it's just released a new Codewarrior version (10.4) but we have no tested the functionality of this project inside this revision.

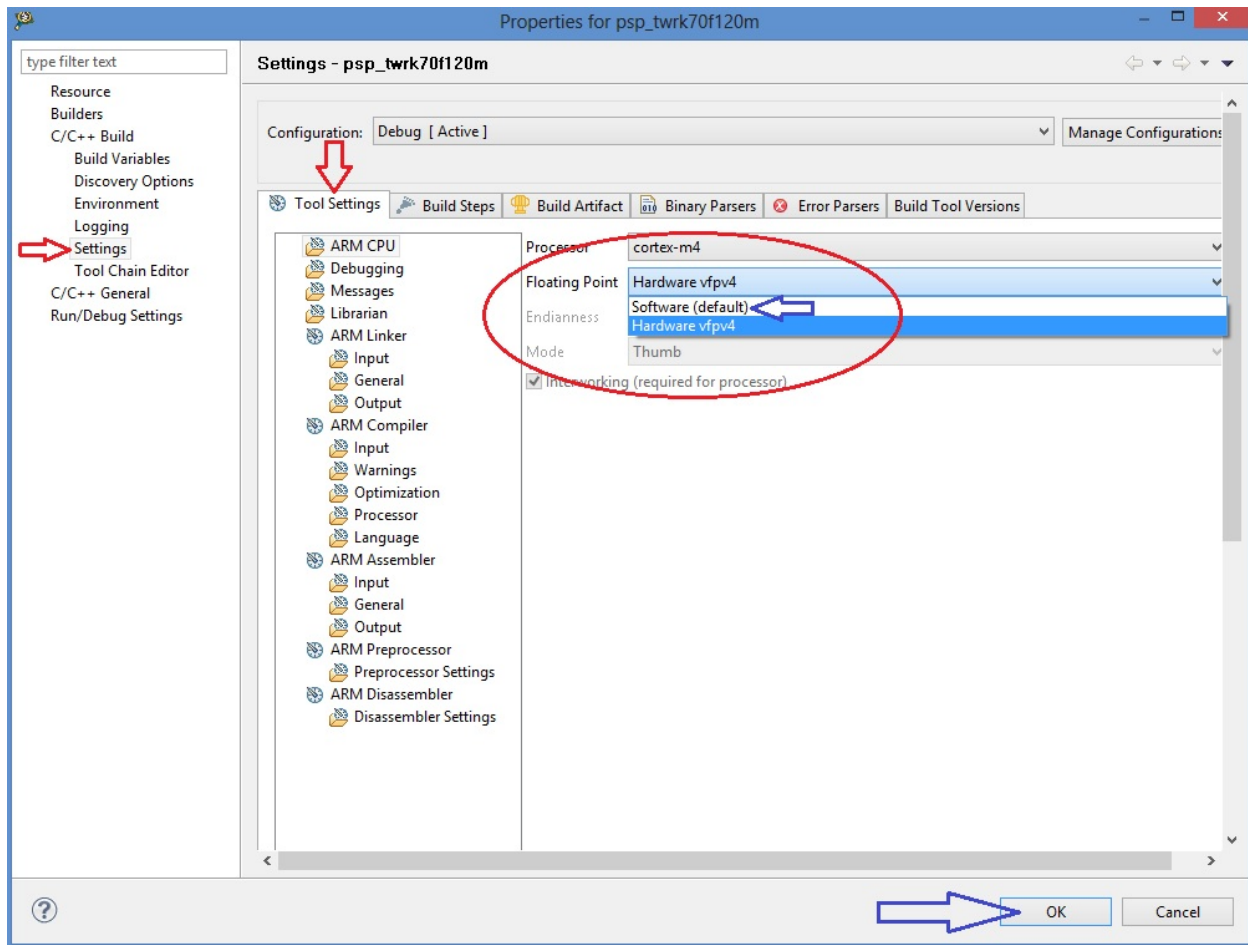
Printf Floating Point settings

In order to see temperature value of MAXIM Pmod DS3231M, MAX31723 and MAX31855 printf function need to be improved with floating poing features. Settings of this functionallity are included in PSP library.

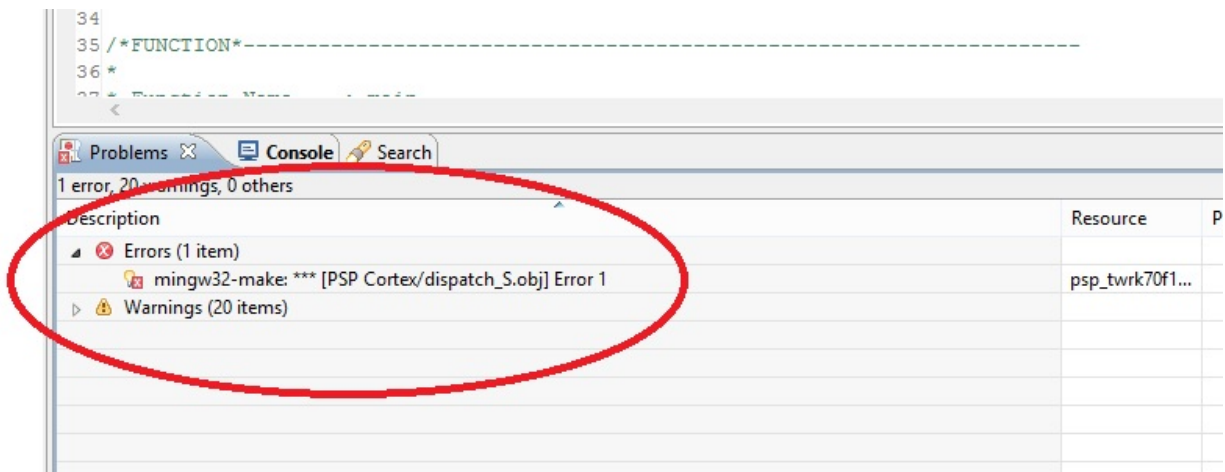
For the normally use of this project, there is no reason to clean and rebuild PSP library. If you need to make this operation, there is a possibility to have some truobles with **printf** function whit floating point variables. If it occurs, there is a trip for rebuild this library without floating point failure.

- **be very careful to the next steps**

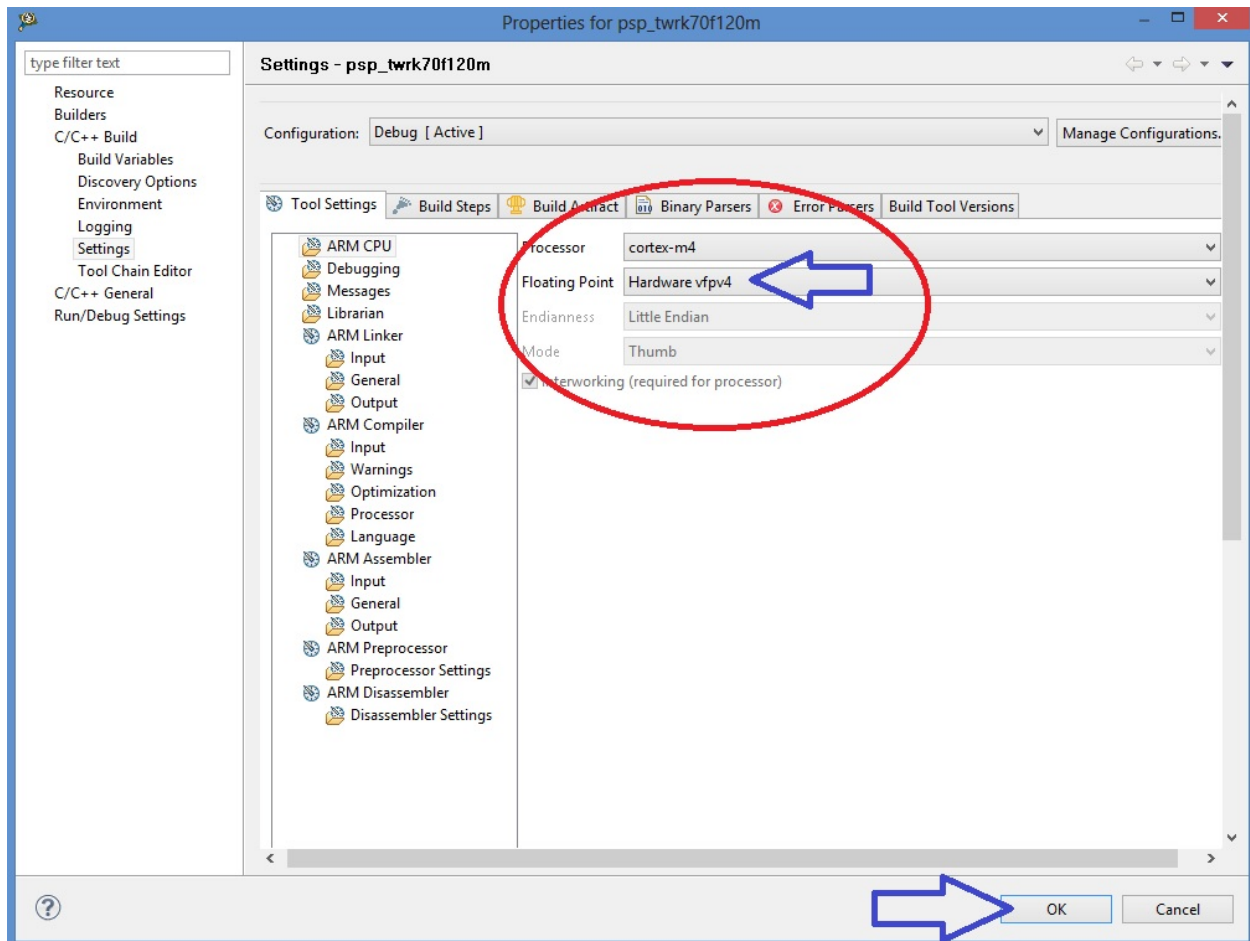
- 1 - Select PSP properties (right-click on project)
- 2 - Select C/C++ Build -> Settings
- 3 - Select and set Floating Point **“Software (default)”**
- 4 - Click **OK** button to confirm selections



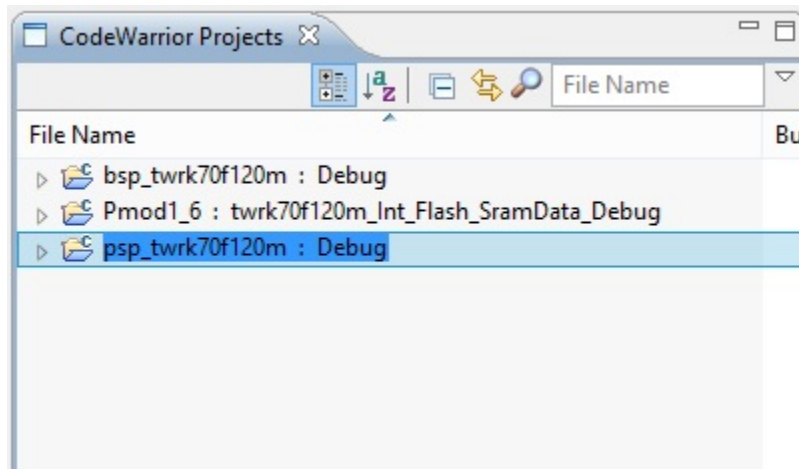
5 - Right click on PSP project, select build. When finished, in the Problems Windows there is one error. Don't care it and proceed with following steps 6-10.



- 6 - Select PSP properties (right-click on project)
- 7 - Select C/C++ Build -> Settings
- 8 - Select amd and set Floating Point **"Hardware vfpv4"**
- 9 - Click **OK** button to confirm selections



10 - Right click on PSP project, select build. When finished, in the Problems Windows there is NO error
BSP library are now built and ready for the project without any error

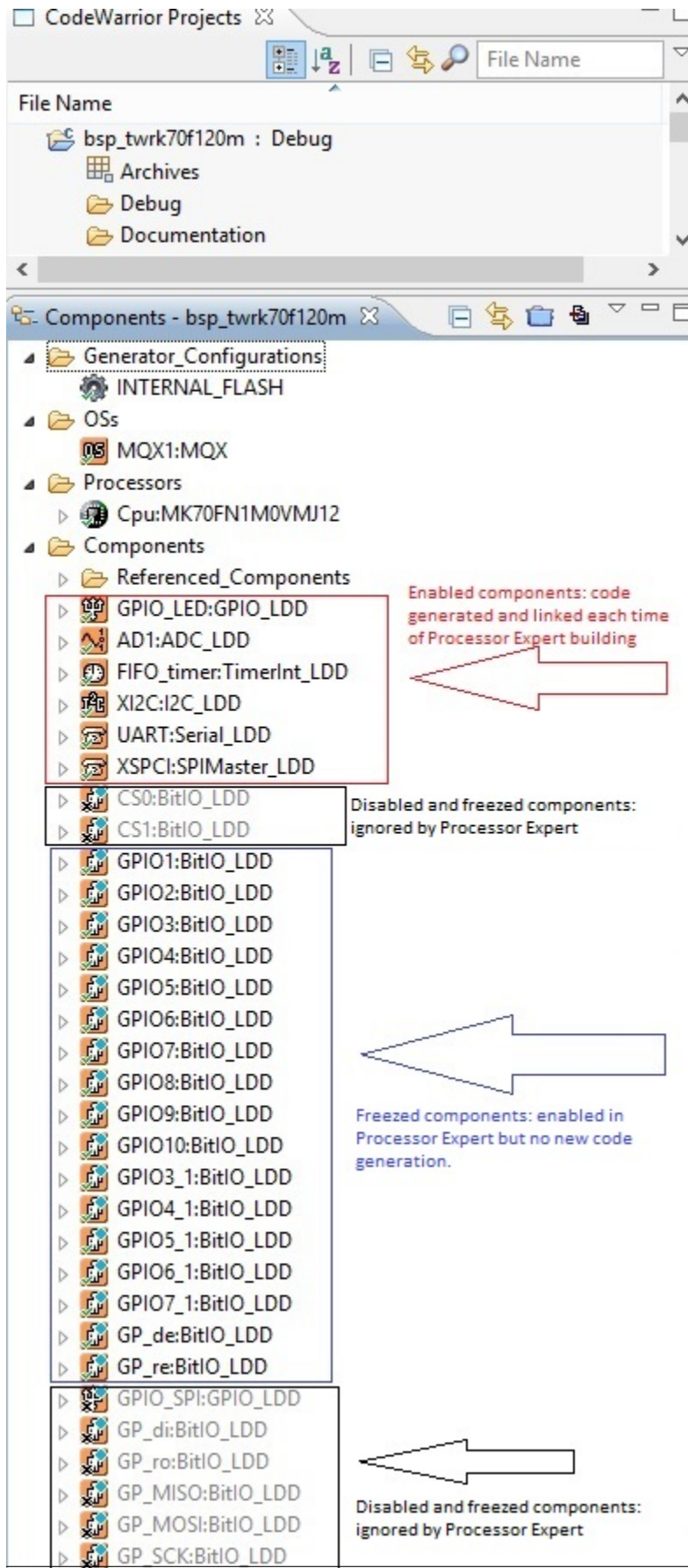


Cautions for Processor Expert suite

Pmod project is developed using HAL driver generated using Processor Expert suite. The suite is located inside BSP library.

Brief summary of Processor Expert components settings

See the figure below.



1- Components red framed are standard Processor Expert component. It's possible to make changes at any time. Code generation takes changes and make new files each time.

2- Components black framed are ignored by Code generation and linker. This mode can be used when you like to use same pins alternately for UART and GPIO function. With Processor Expert using more than one components for same pin make errors in code generation. To make this, you can enable first components using pins, generate code, disable components and enable the second one for new code generation. The result is tha you have more source code using same pins.

3- Component blue framed. Component “freezed”. Code generation is disabled. This is because Processor Expert in Codewarrior 10.3, in BitIo or BitsIO components, don't add mxq include files during code generation. The result of this is a general linker error during build of BSP library. If yuo make changes or add a new one, yuc must add “manually” the needed include file “mxq.h” in component code (.h file) and disable code generation.

ADDING DEFINITION IN PE_LDD FILES

When you make any changes os adds in Processor Expert components, you must proceed with new code generation. After this, because of disabled components in project, some manual adding are needed in PE_LDD.h and PE_LDD.c files (see folder Generated_Code in BSP project) Look at **PE_LDD add.txt** in project folder (C:\Pmqx). You must copy definition inside this file into PE_LDD.h as image below:

```

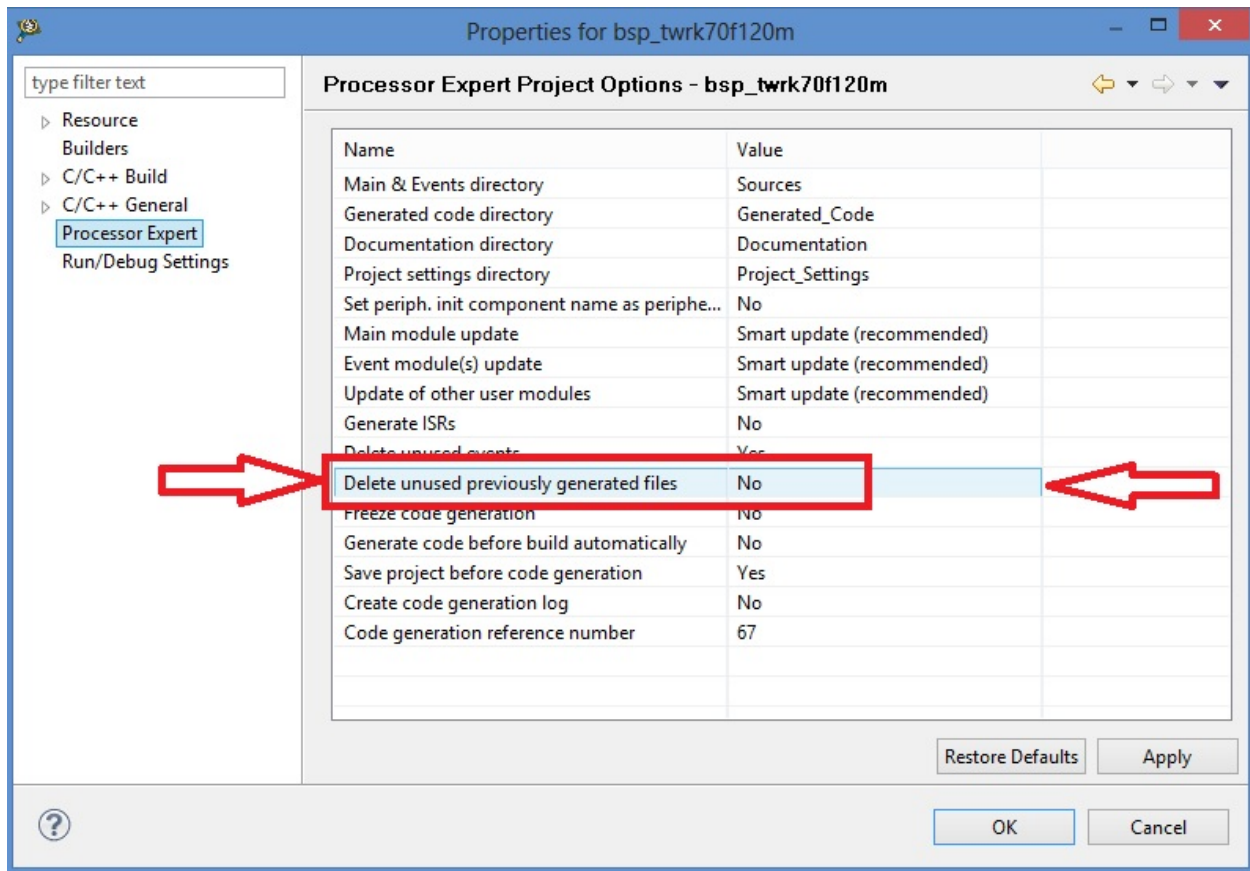
** =====
** LDD component ID specifying the component instance in the project. This ID
** is used internally as an index to the array of LDD device structures.
** =====
*/
#define PE_LDD_COMPONENT_GPIO_LED_ID          0x00U
#define PE_LDD_COMPONENT_AD1_ID               0x01U
#define PE_LDD_COMPONENT_TU1_ID               0x02U
#define PE_LDD_COMPONENT_FIFO_timer_ID       0x03U
#define PE_LDD_COMPONENT_XI2C_ID              0x04U
#define PE_LDD_COMPONENT_UART_ID              0x05U
#define PE_LDD_COMPONENT_XSPCI_ID             0x06U
#define PE_LDD_COMPONENT_GPIO1_ID             0x07U
#define PE_LDD_COMPONENT_GPIO2_ID             0x08U
#define PE_LDD_COMPONENT_GPIO3_ID             0x09U
#define PE_LDD_COMPONENT_GPIO4_ID             0x0AU
#define PE_LDD_COMPONENT_GPIO5_ID             0x0BU
#define PE_LDD_COMPONENT_GPIO6_ID             0x0CU
#define PE_LDD_COMPONENT_GPIO7_ID             0x0DU
#define PE_LDD_COMPONENT_GPIO8_ID             0x0EU
#define PE_LDD_COMPONENT_GPIO9_ID             0x0FU
#define PE_LDD_COMPONENT_GPIO10_ID            0x10U
#define PE_LDD_COMPONENT_GPIO3_1_ID           0x11U
#define PE_LDD_COMPONENT_GPIO4_1_ID           0x12U
#define PE_LDD_COMPONENT_GPIO5_1_ID           0x13U
#define PE_LDD_COMPONENT_GPIO6_1_ID           0x14U
#define PE_LDD_COMPONENT_GPIO7_1_ID           0x15U
#define PE_LDD_COMPONENT_GP_de_ID             0x16U
#define PE_LDD_COMPONENT_GP_re_ID             0x17U

#define PE_LDD_COMPONENT_CS0_ID                0x18U    <-- added definition
#define PE_LDD_COMPONENT_CS1_ID                0x19U    <-- added definition
#define PE_LDD_COMPONENT_GP_di_ID              0x1AU    <-- added definition
#define PE_LDD_COMPONENT_GP_ro_ID              0x1BU    <-- added definition
#define PE_LDD_COMPONENT_GPIO_SPI_ID           0x1CU    <-- added definition
#define PE_LDD_COMPONENT_GP_MISO_ID            0x1DU    <-- added definition
#define PE_LDD_COMPONENT_GP_MOSI_ID            0x1EU    <-- added definition
#define PE_LDD_COMPONENT_GP_SCK_ID             0x1FU    <-- added definition

```


The result of changes is show in figure below

IMPORTANT SETTING To obtain functionality as described above, you must go to BSP Properties → Processor Expert Option and set “Delete unused previously generated file = NO”. See figure below

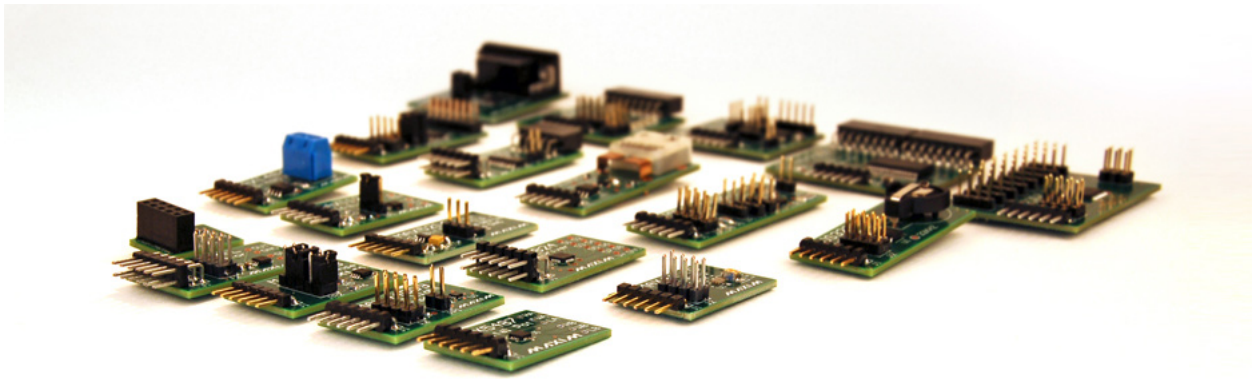


NOTE: on further revisions of Codewarrior these tips have to be checked and verified

CHAPTER 6

More about Pmod

Maxim Analog Essential Collection is a collection of plug-in peripheral modules (Pmod) You can find more informations visiting [Maxim Analog Essential Collection](#) site



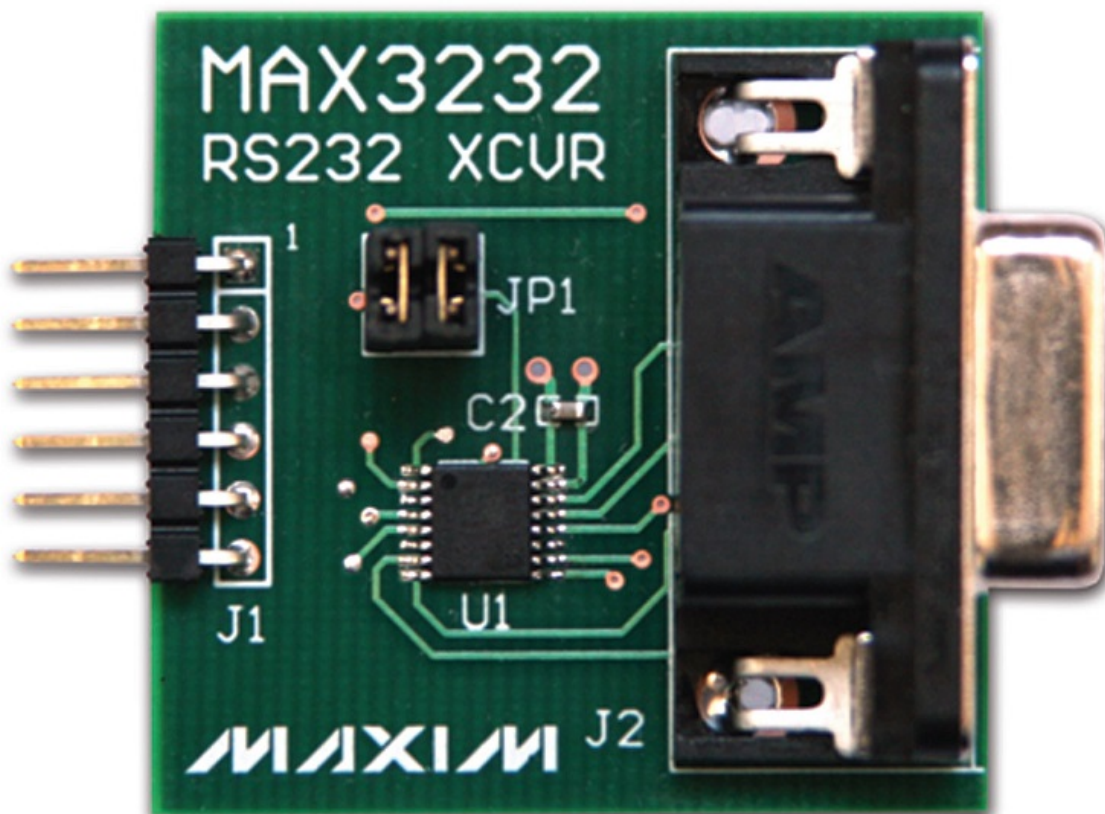
Important notice

At the date of issue of this review, Maxim Zenboard Platform Project files are available on version 1.6, and don't support MAX14850 Pmod module. As a result this version, that use original files from Maxim project, is not able to emulate this device.



Emulation of MAX3232

This device is a RS232 converter, and require 2 serial channel (each one connected to terminal software) for full test. The first one is used for commands and the second one must be connected, for complete testing purpose, to MAX3232 Pmod serial connector (by standard modem cable).



We also suggest you to see documentation [Maxim Pmod-Compatible Plug-In Peripheral Modules](#) for any specific

further detail.

- search

C

CwInst, 1

M

MaxFiles, 36