APPLICATION COMMON OPERATING ENVIRONMENT (APPCOE)

TRAINING GUIDE

Version 1.0 – March 12, 2013

MapuSoft Technologies
1301 Azalea Road
Mobile, AL 36693
www.mapusoft.com

Copyright

For additional assistance, please contact us at:
MapuSoft Technologies
1301 Azalea Road
Mobile, Alabama 36693
251.665.0280
251.660.0288 FAX
support@mapusoft.com
info@mapusoft.com
http://www.mapusoft.com

Table of Contents

1. General Introduction

1.1 What is AppCOE ?

AppCOE (Application Common Operating Environment) is a framework of common architecture that promotes interoperability and cross-platform capabilities among systems and devices. It is built on the powerful open source Eclipse-based framework and integrates all of MapuSoft's tools: OS Changer, Cross-OS Development Platform, Cross-OS Hypervisor, Linux OK, OS Simulator, App/Platform Profiler, OS Version UpKit and Ada-C/C++ Changer. Embedded C, C++ and Ada applications can be standardized on AppCOE to allow the applications to interoperate and run seamlessly on a single platform.

1.2 Installing AppCOE

Downloading AppCOE

You can download a free demo or trial version as per your requirement from our website. After downloading the installer file proceed as follows.

Download URL: http://www.mapusoft.com/downloads/

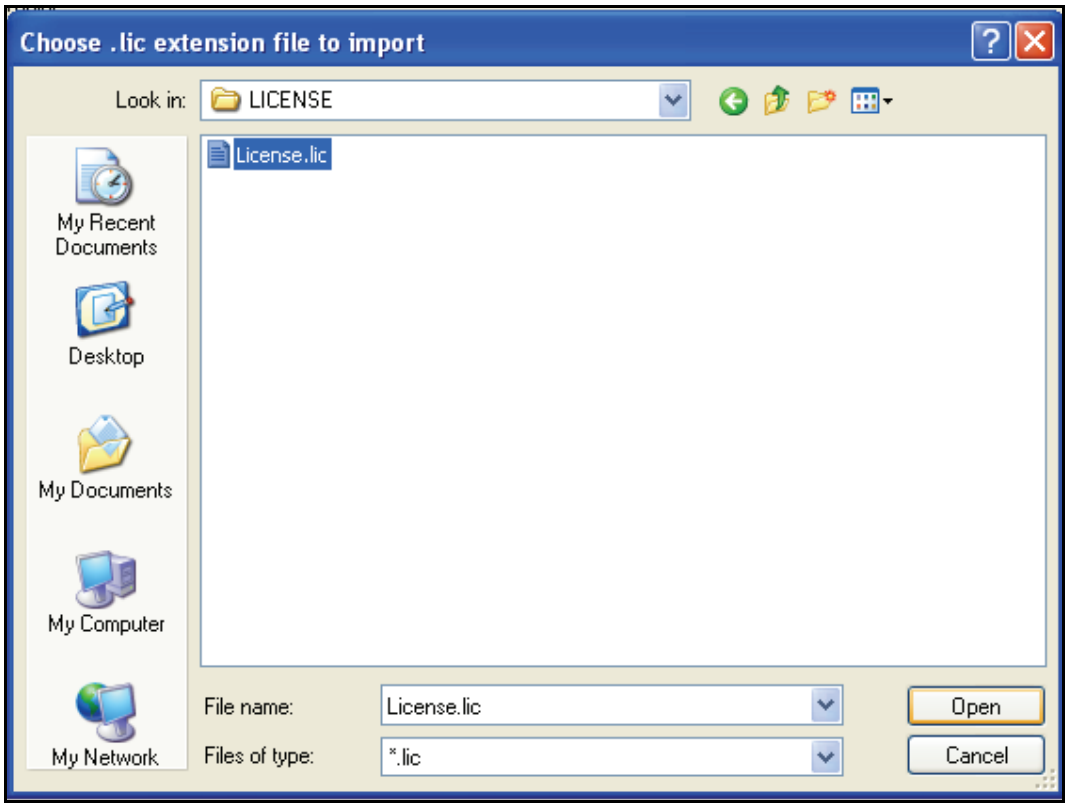
Installing in Windows

1. Double click on the installer file.
2. Installer will ask for a directory to install AppCOE release. Browse to the directory or provide a directory name when prompted
3. Once installed, reboot the computer. AppCOE will not run properly without re-boot
4. Now Run **appcoe.exe** in AppCOE < installdir > or launch the AppCOE application from the windows desktop **menu: Programs>AppCOE**

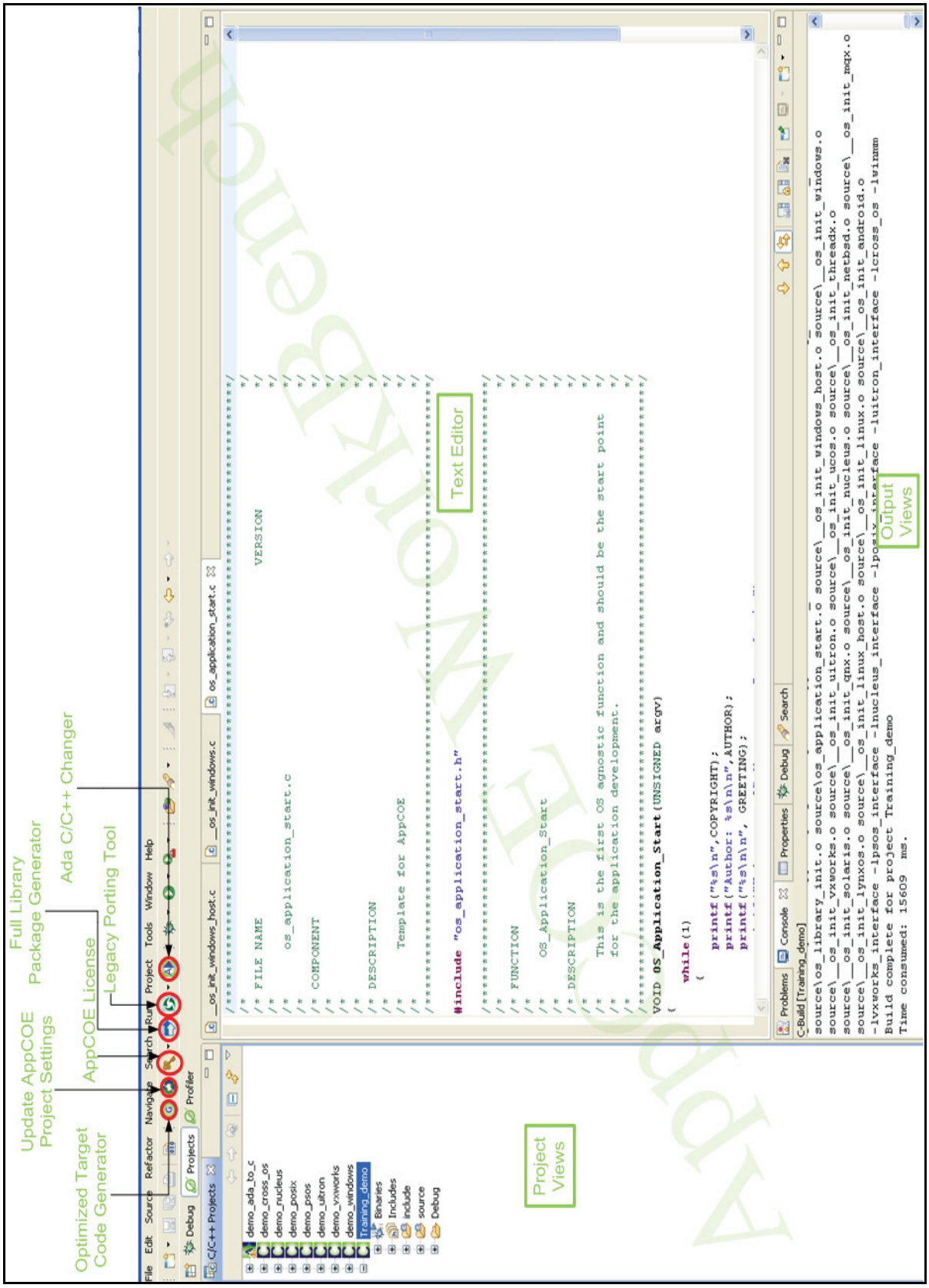
In order to get additional functionalities and upgrade your AppCOE to full version, ask for license from Mapusoft Support. To install the license key proceed as below.

1.3 Installing license key

Launch AppCOE**.** Click on "  " icon. Browse to the path where the license key is stored on your local drive and then select install. AppCOE will process the key and will keep a copy of the license file inside the <installdir/license> folder


Choose .lic extension file to import
Look in:
LICENSE
License.lic
My Recent Documents
Desktop
My Documents
My Computer
My Network
File name:
License.lic
Files of type:
*.lic
Open
Cancel

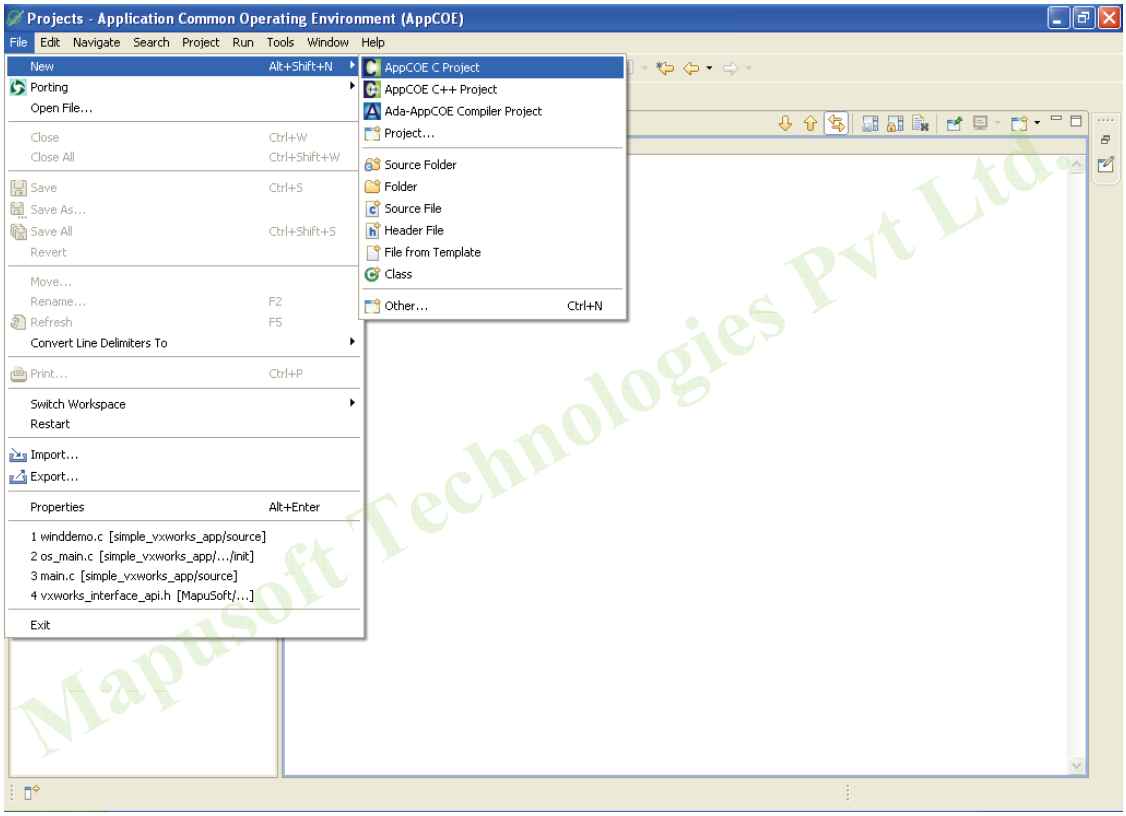
1.4 Introduction to AppCOE IDE



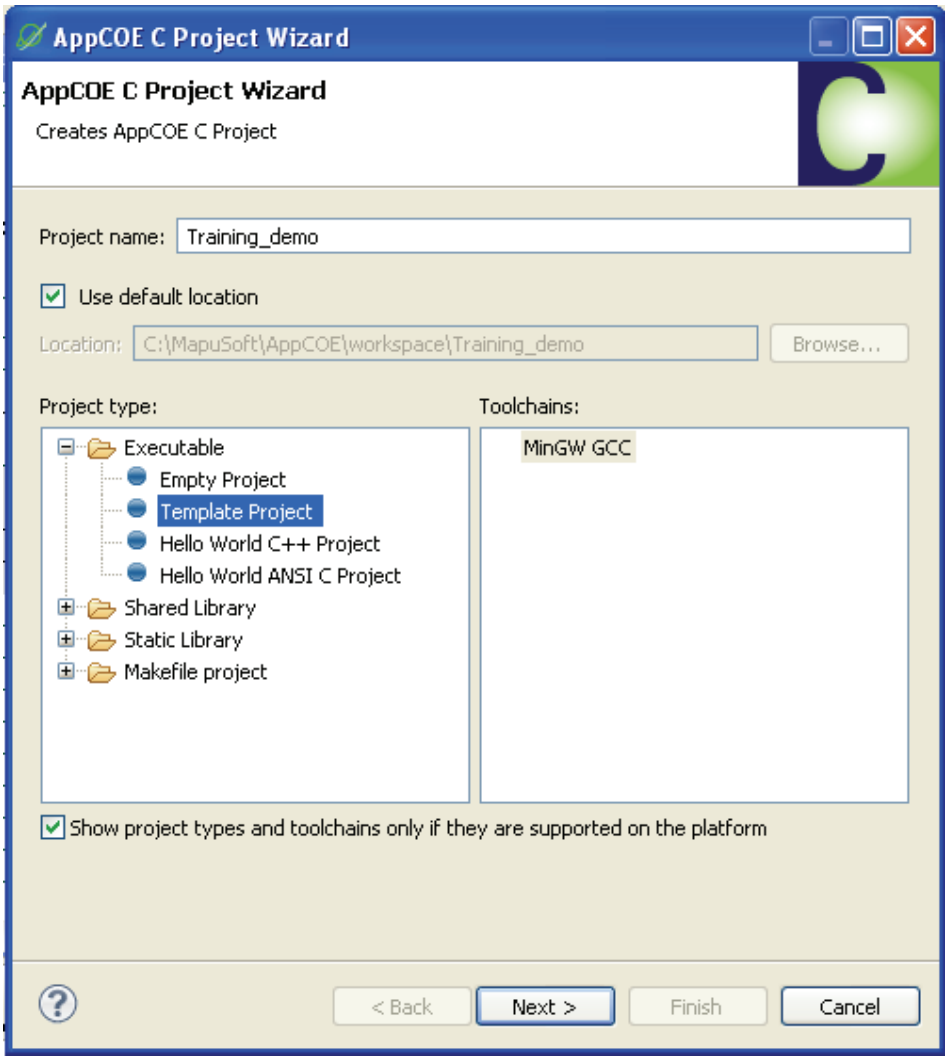
2. Develop C/C++ Projects

2.1 Creating an AppCOE C/C++ project

1. In AppCOE workbench, goto File > New > **AppCOE C/C++ Project.**

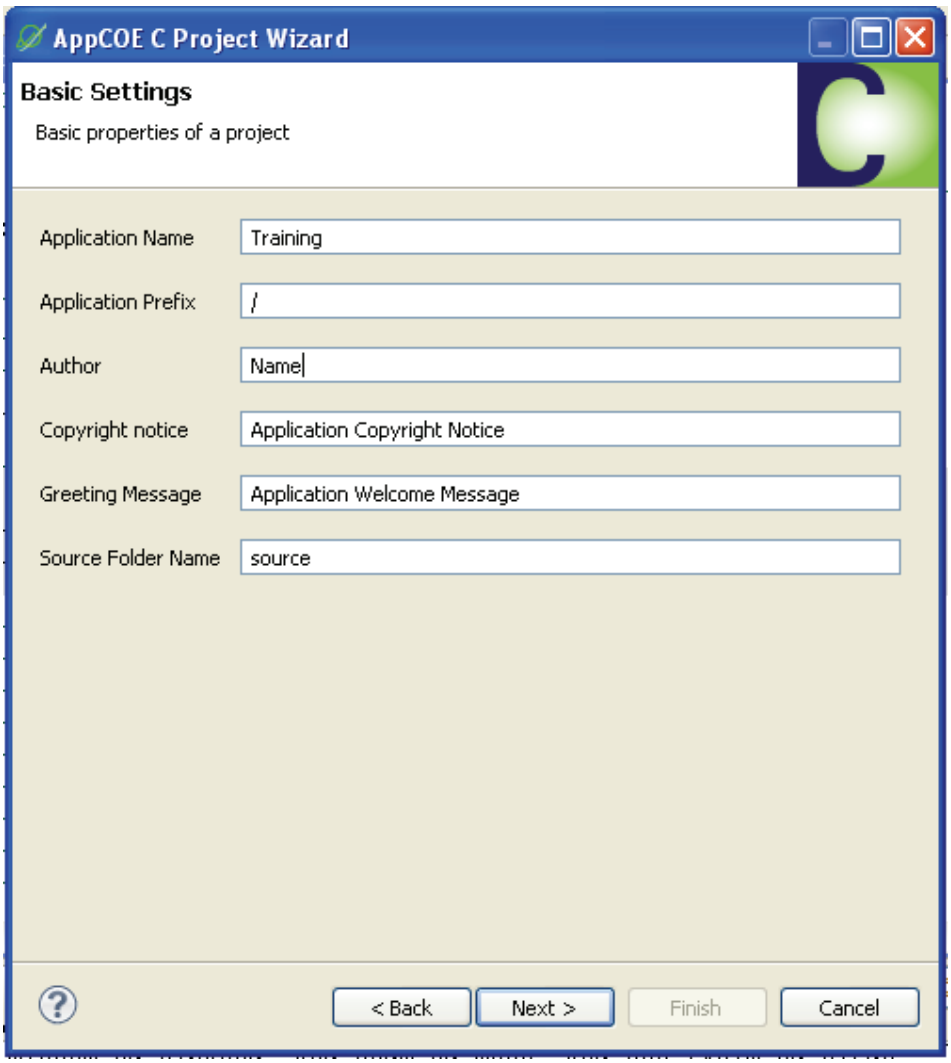


2. On AppCOE C/C++ Project Wizard window, type a project name and give a location next to **Project Name** text box.



3. Under Project Types, expand the Executable menu(You can also create library project of makefile project just the same way we do in eclipse), here we are creating an **Executable Template Project**.
    a) Now Choose the type of project you want to create.
    b) The difference between Empty and Template project is just that in **Empty project** you need to write your code from scratch, i.e. from "**main**". But in **Template Project** AppCOE does everything for you. So it will put up all the required files for a demo application that is ready to build and run. Further if you want to have your own application running, you can just customize **Template Project** source code as per your needs.

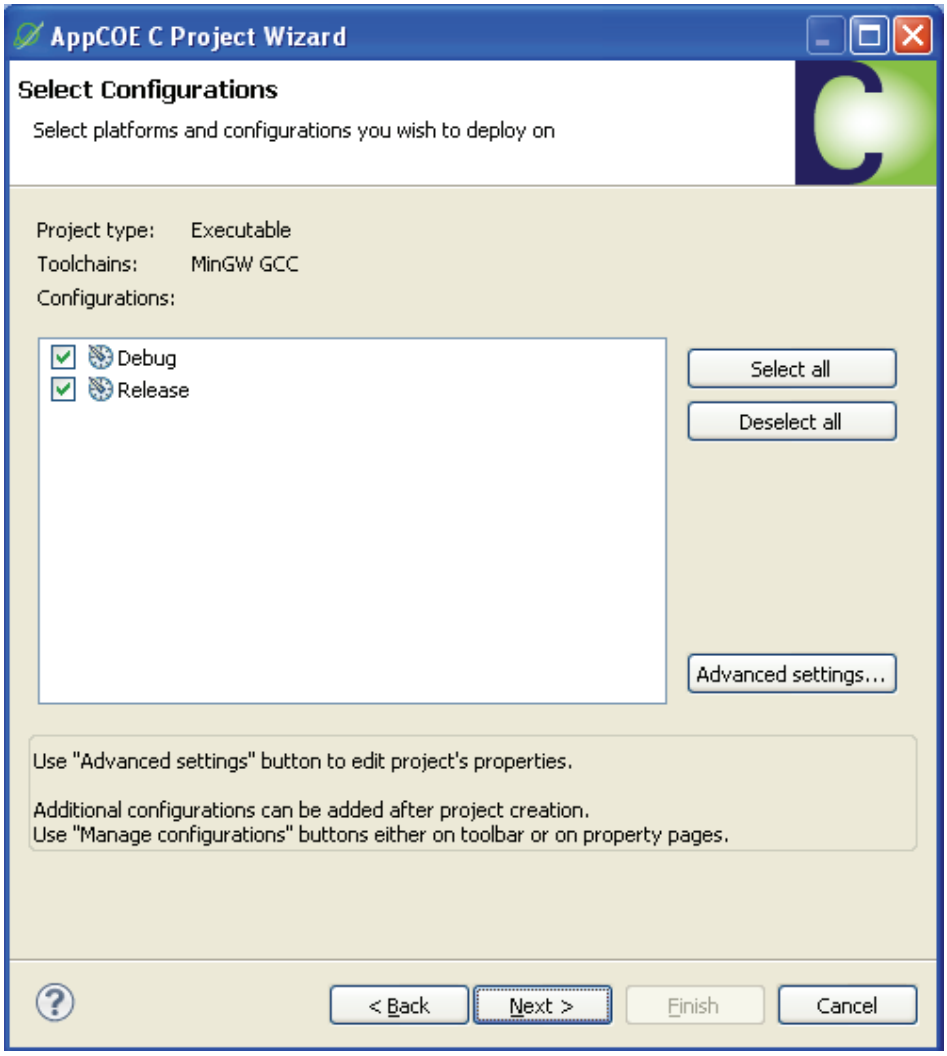
4. On Basic Settings window, define the basic properties of your project and click Next.



4.1 The Application Name should be within 1 to 8 characters.

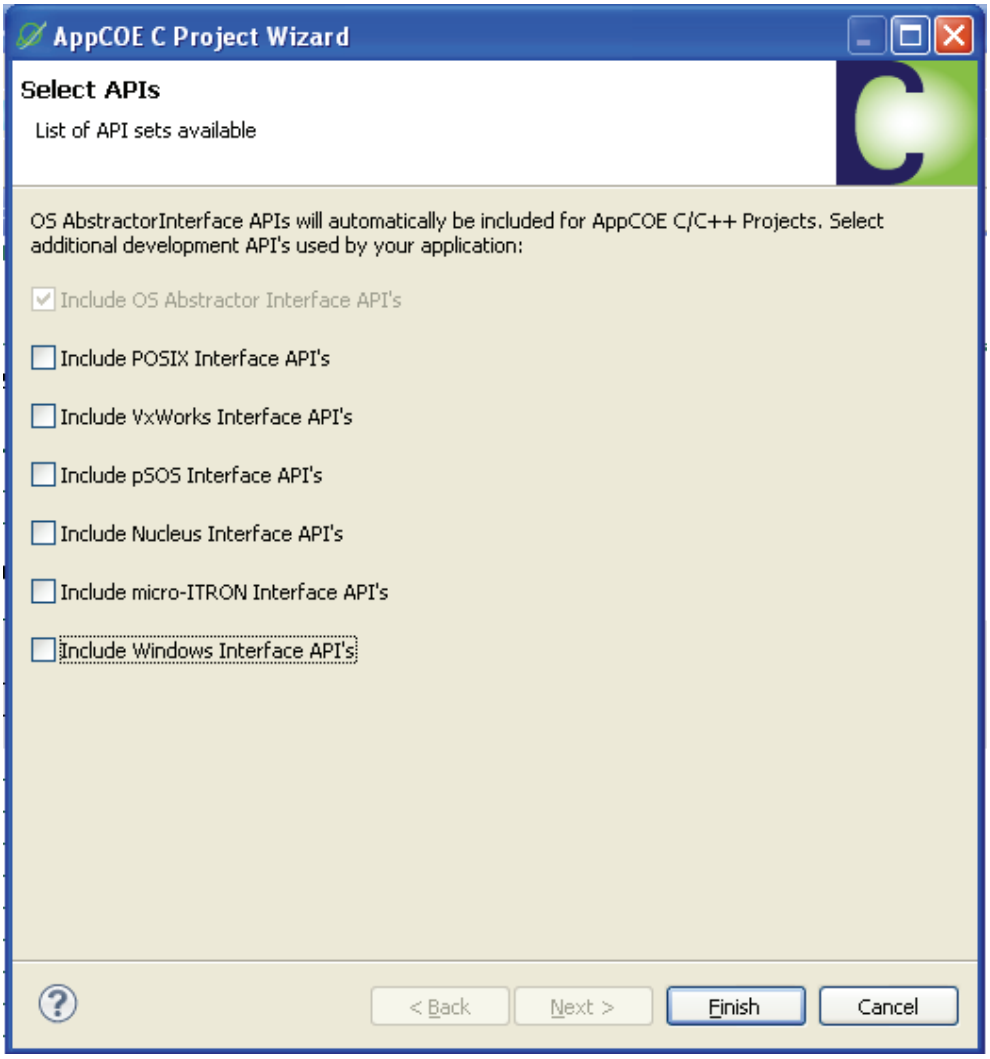
4.2 The Application Prefix is the path for all your resources. For e.x. in medical applications you may have prefix as **"/cabinet-A/Shelf-1/Box-5**..." and so on. By default it is set to root.

5. On Select Configurations window, select the platforms and configurations for deployment and click Next

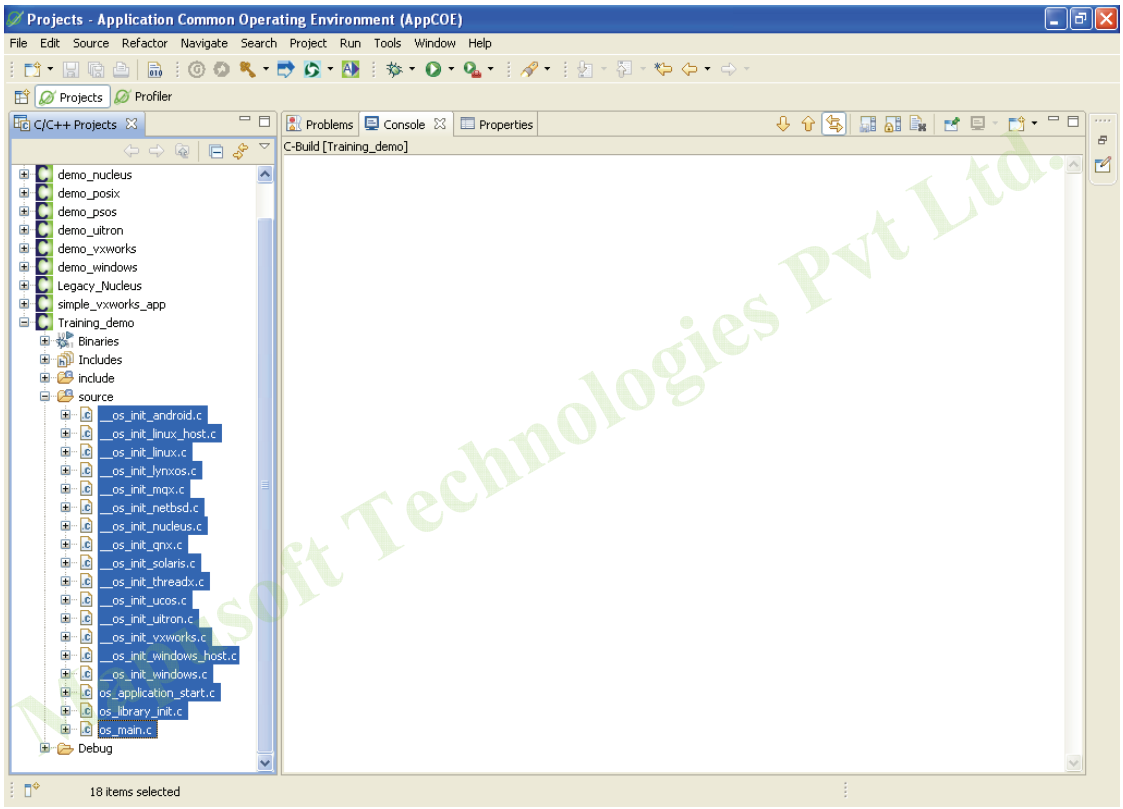


6. On Select APIs window, select the required AppCOE development APIs and click **Finish**

Please Note: If your want to include support for multiple interfaces then you can accordingly check the desired checkboxes.



→Now an AppCOE template project will be created as below.



You can view the following template files for your project on the left pane of the window. The description of these files is as follows:

- **_os_init_<os name>.c or _os_init_<os name>_host.c** – This is entry function for the native operating system (in this case it is windows). This is where you should put any of your operating system specific code. For instance, if you want to add a signal handler on <os name> host, you could do it here before calling OS_Main() in the respective "_os_init_<os name>.c" function. When optimizing, you will need to write an equivalent function for your target operating system**.**

- **os_application_start.c–** This function is the first OS agnostic function and should be the start point for the application development**.**

- **os_library_init.c–** This function initializes the required Interface products and creates the entry threads for each product.

- **os_main.c–** This function initializes the OS Abstractor Interface layer and calls OS_Application_Wait_For_End which will suspend until OS_Application_Free or OS_Delete_Process is called. It also spawns the first OS Independents thread which is the true entry point for OS Abstractor Interface.

The application code starts in the os_library_init.c file, the user defined entry function and name of the application for OS Abstractor Interface can be specified in:

#define OS_ABSTRACTOR_BASE_ENTRY_FUNCTION

#define OS_APPLICATION_START_TASK_NAME

For ex. in Vxworks interface, the user defined entry function and stack size can be specified in:

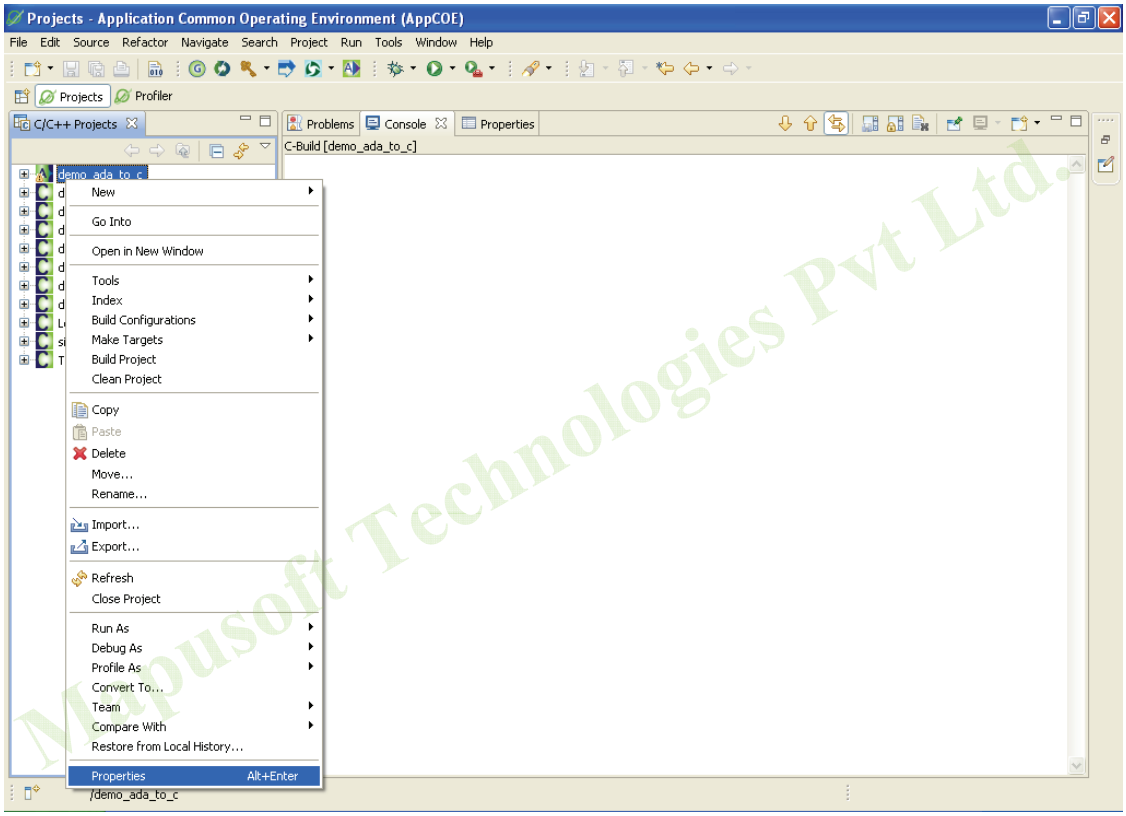
#define VXWORKS_ENTRY_FUNCTION

#define VXWORKS_ENTRY_FUNCTION_STACK_SIZE

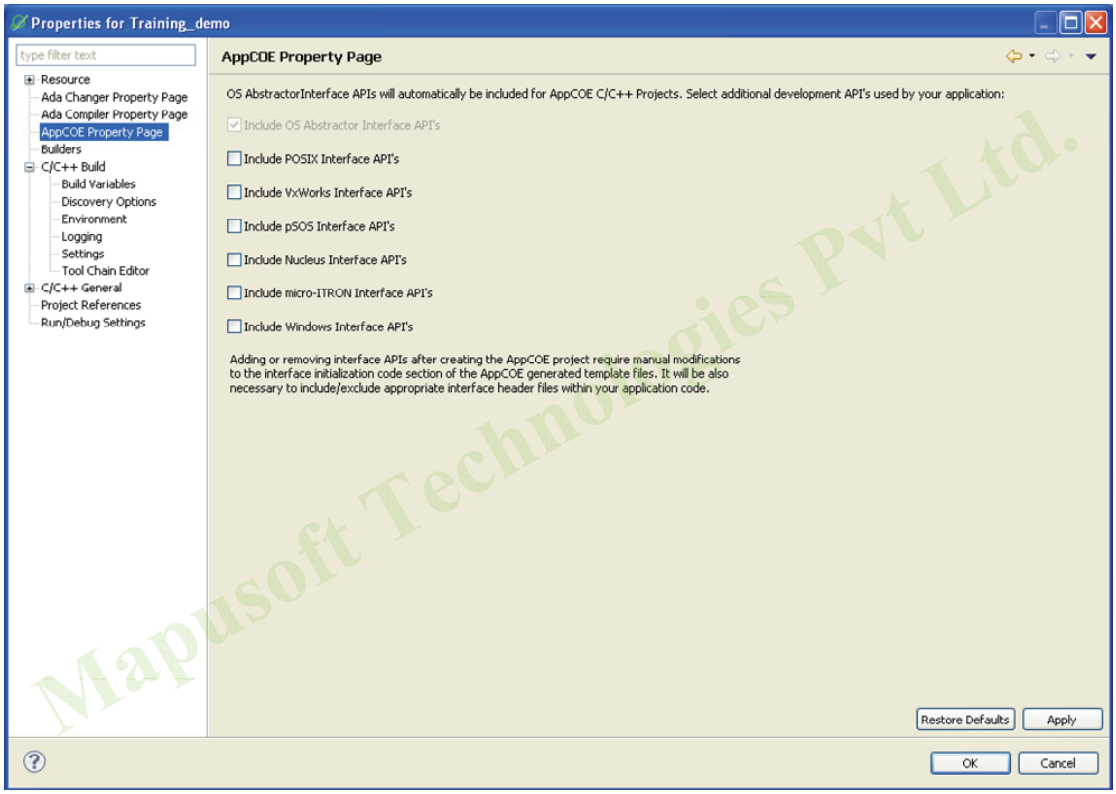
Similarly, this is how it works for all the remaining changers/abstractors.

2.2 Configuring Project Settings

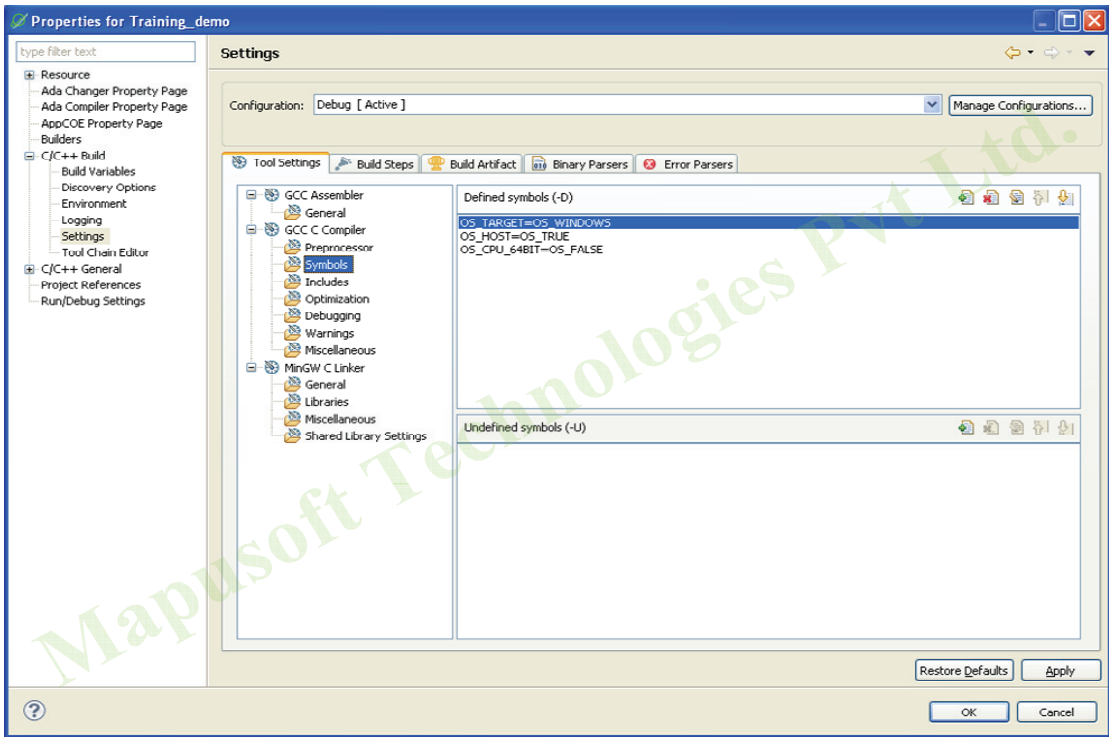
1. Right click on the respective project and click on properties as shown below



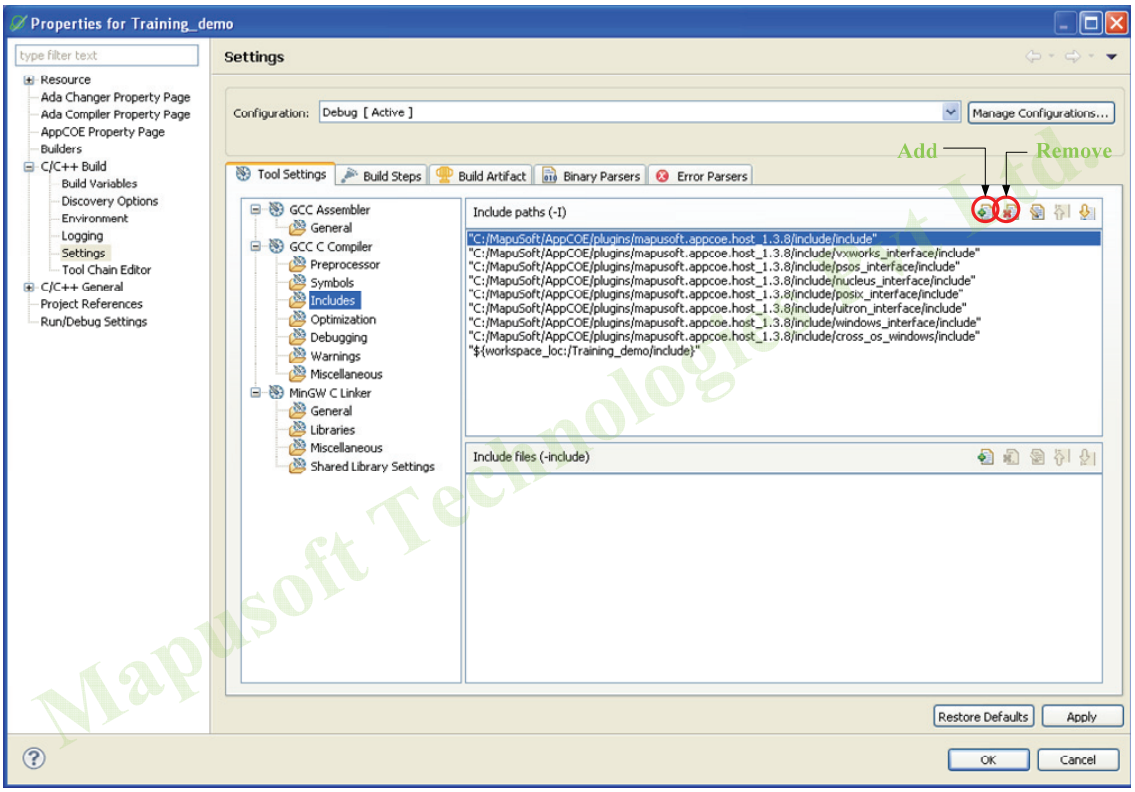
2. The AppCOE property page will appear as below:
    a. Here you can again add/remove available development API`s as in "6th step of Creating an AppCOE C / C++ project" by checking / unchecking the desired checkboxes.
    b. In order to view / modify various other settings of the project you can scroll between the respective nodes on the left.



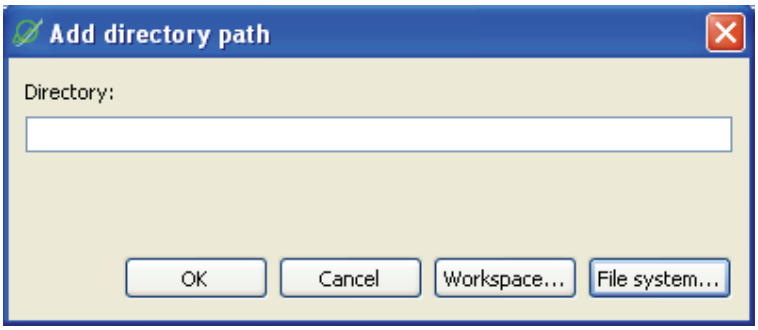
3. AppCOE uses some flags to automatically compile the required part of code. To view these flags goto C/C++ build > Settings and then in the Tool Settings tab goto GCC C Compiler > Symbols.



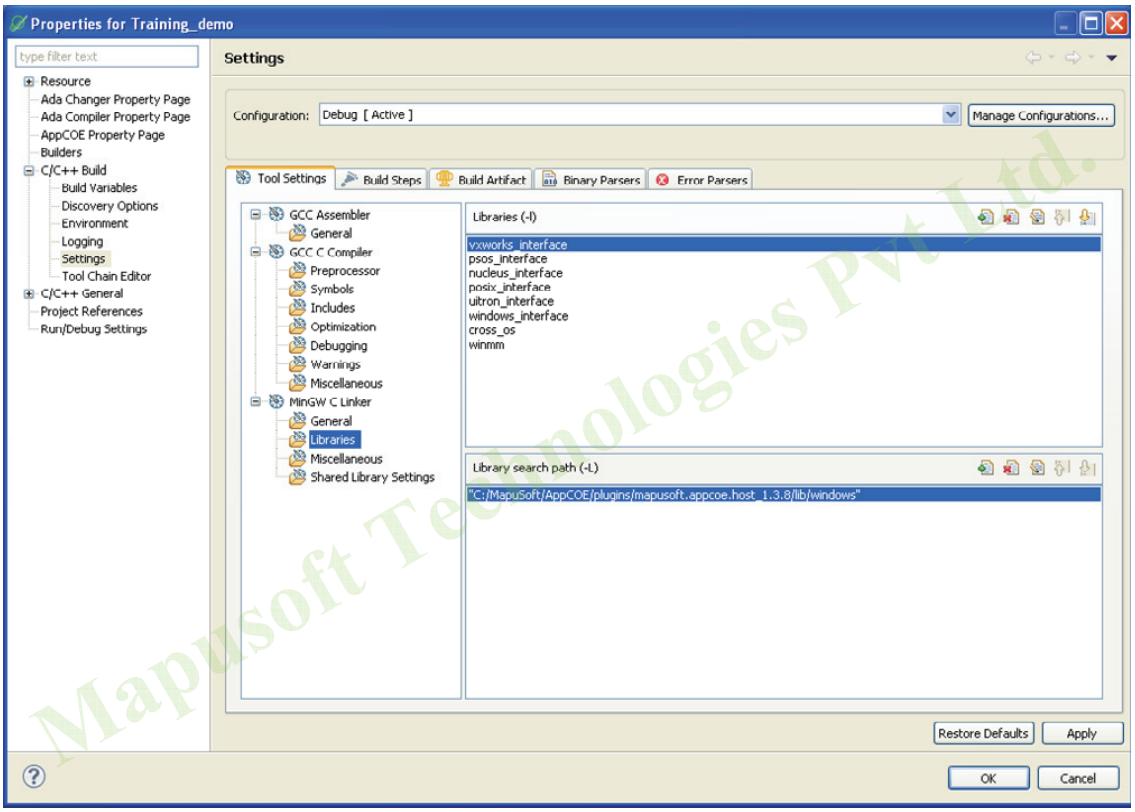
4. To view / modify include files or paths goto **C/C++ build > Settings** and then in the **Tool Settings** tab goto **GCC C Compiler > Includes**
    a. In the figure below you can see include paths of all the development API`s that you included earlier
    b. In order to remove any path select the corresponding path and click on **Remove** button



    c. In order to add any path, click on **Add** button, "Add directory path" Dialog box will appear.
    d. Enter the appropriate directory path or select it from workspace or directly select it from file system. After entering the path click on OK.



5. To view / modify the Libraries goto **C/C++ build > Settings** and then in the **Tool Settings** tab goto **GCC C Compiler > Libraries.**
    a. In the figure below you will be able to see libraries of all the development API`s that you included earlier.
    b. In order to add/remove any library proceed similarly as done in Step 4 above to add include paths.



6. Once all the settings have been done. Click on **Apply** and then **OK.**

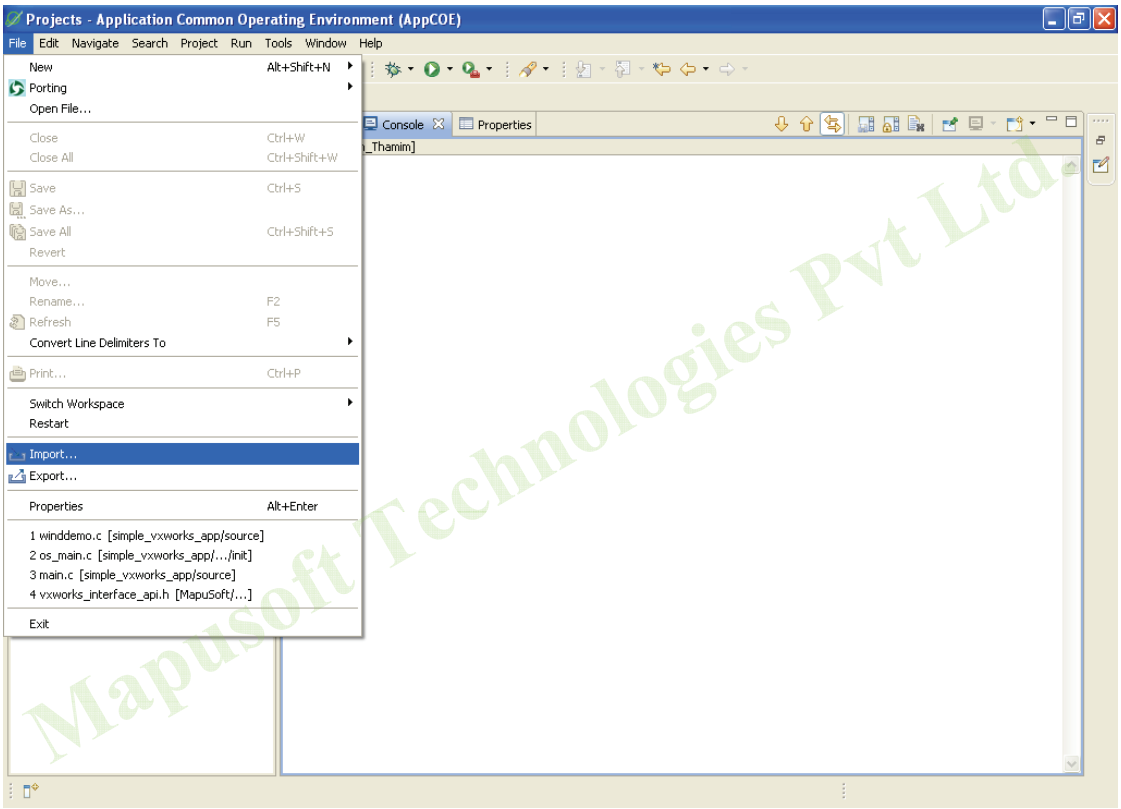
Please Note: It has been observed that many developers are not able to compile and run their projects even after having added the appropriate libraries and include paths.

I. Here one reason may be wrong path, you can edit the include paths, libraries or library search paths by using Edit ( ) button, next to remove button.

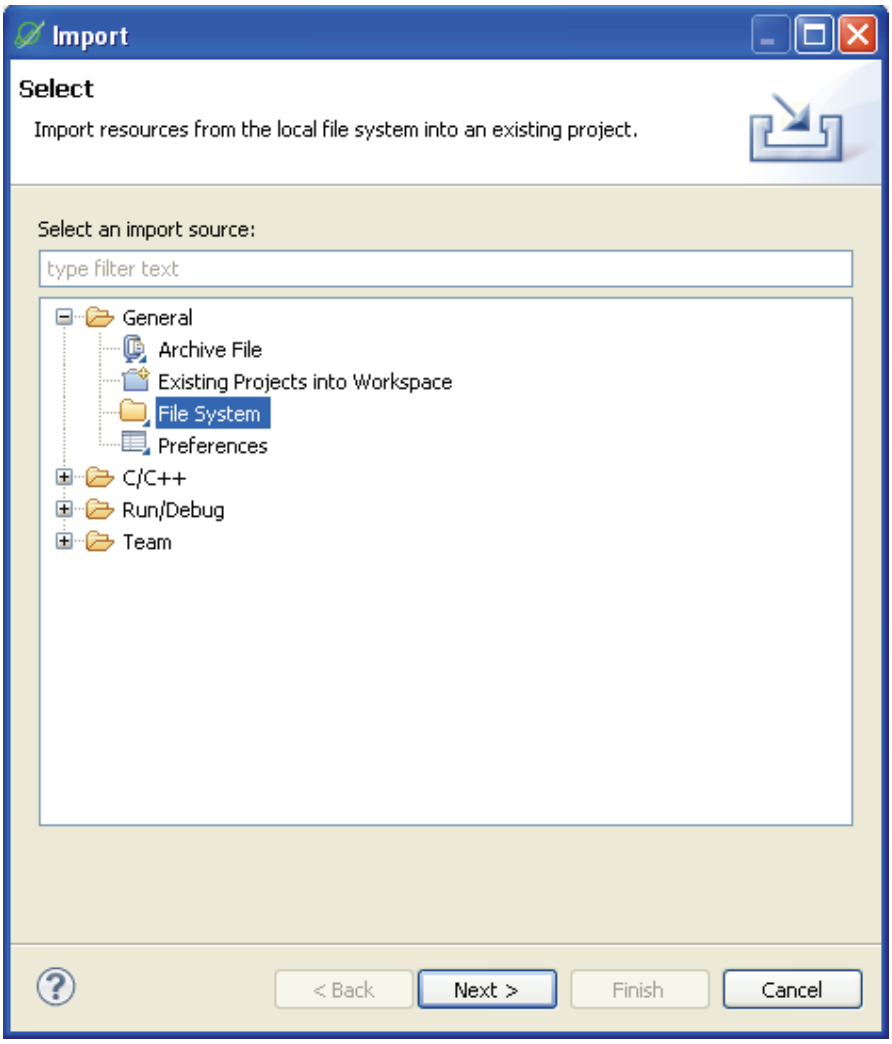
II. Another situation may be that the paths and libraries are not in the desired sequence as they are required during build. So you need to shift the desired library or their include paths up/down using and buttons next to edit button.

2.3 Importing existing projects/source files into Workspace

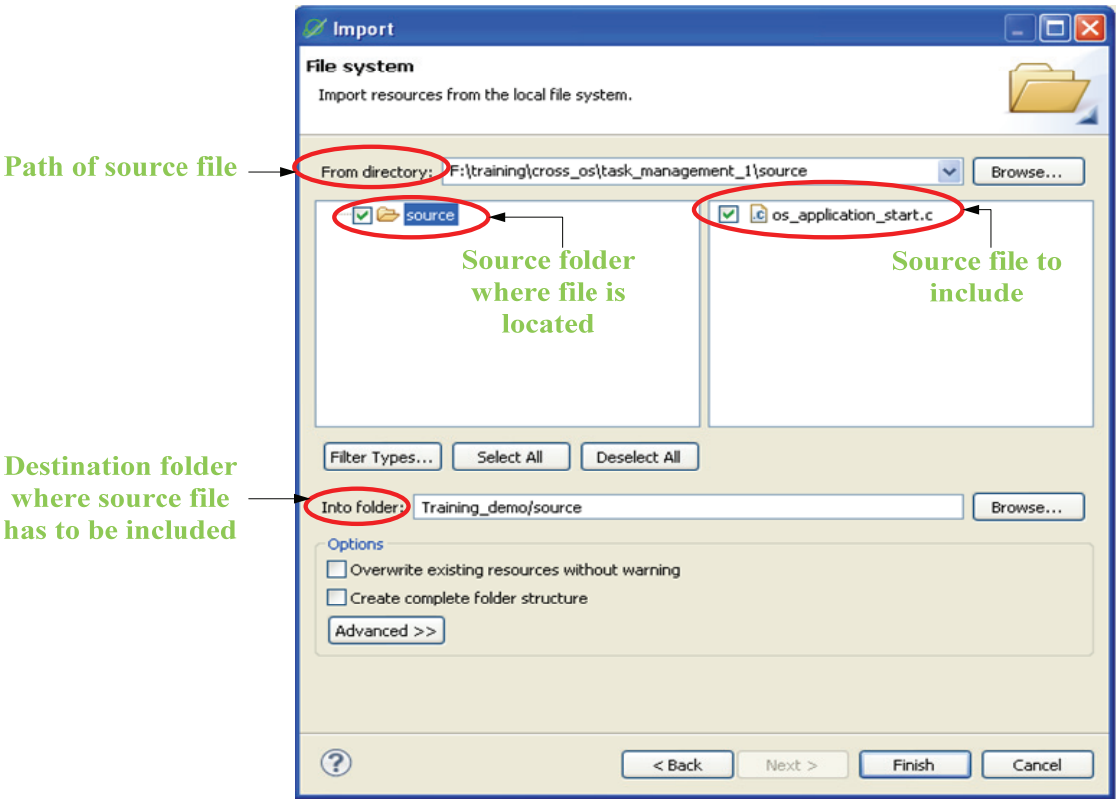
1. To import Project or source files goto File > Import.



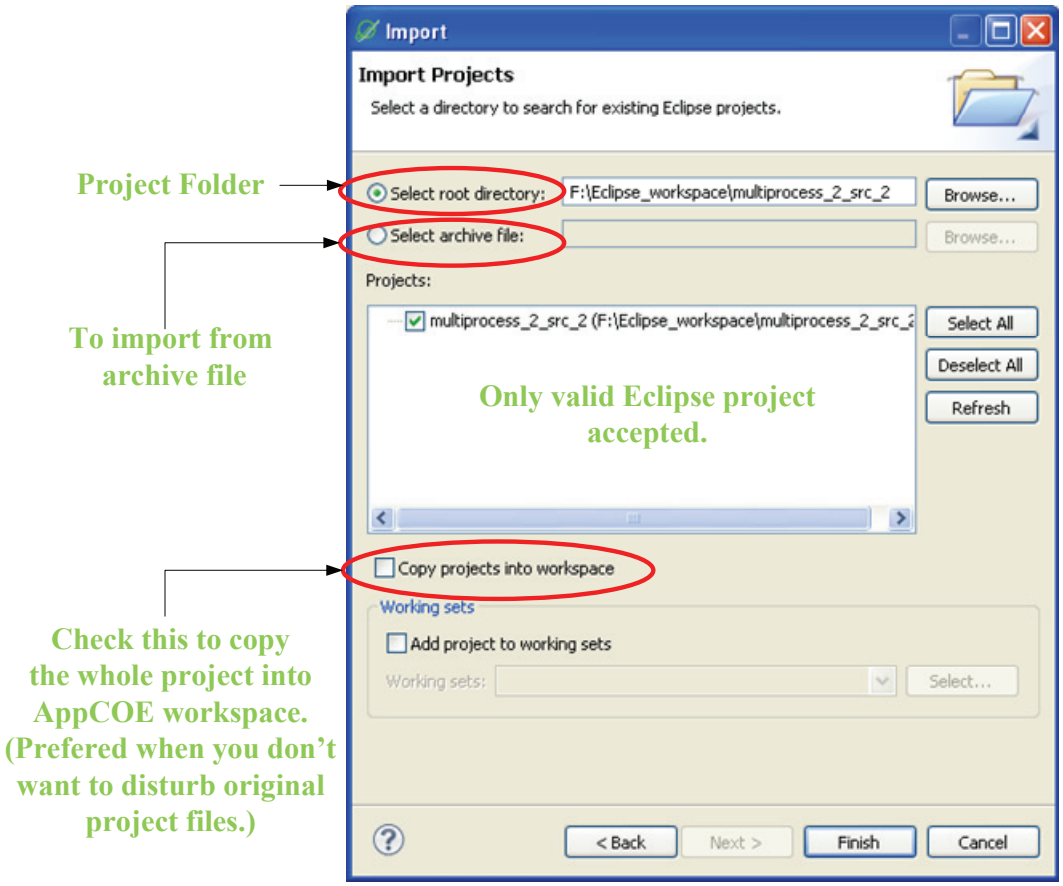
2. To Import existing source files, choose **File System** from the **Import window** and click on **Next**.



3. In the proceeding enter the required fields as required and click on **Finish.**



4. To Import existing project, choose **Existing Projects into Workspace** from the **Import window** and click on **Next**.
    a. In the proceeding window enter the required fields and click on Finish. The selected project will be imported into the workspace.

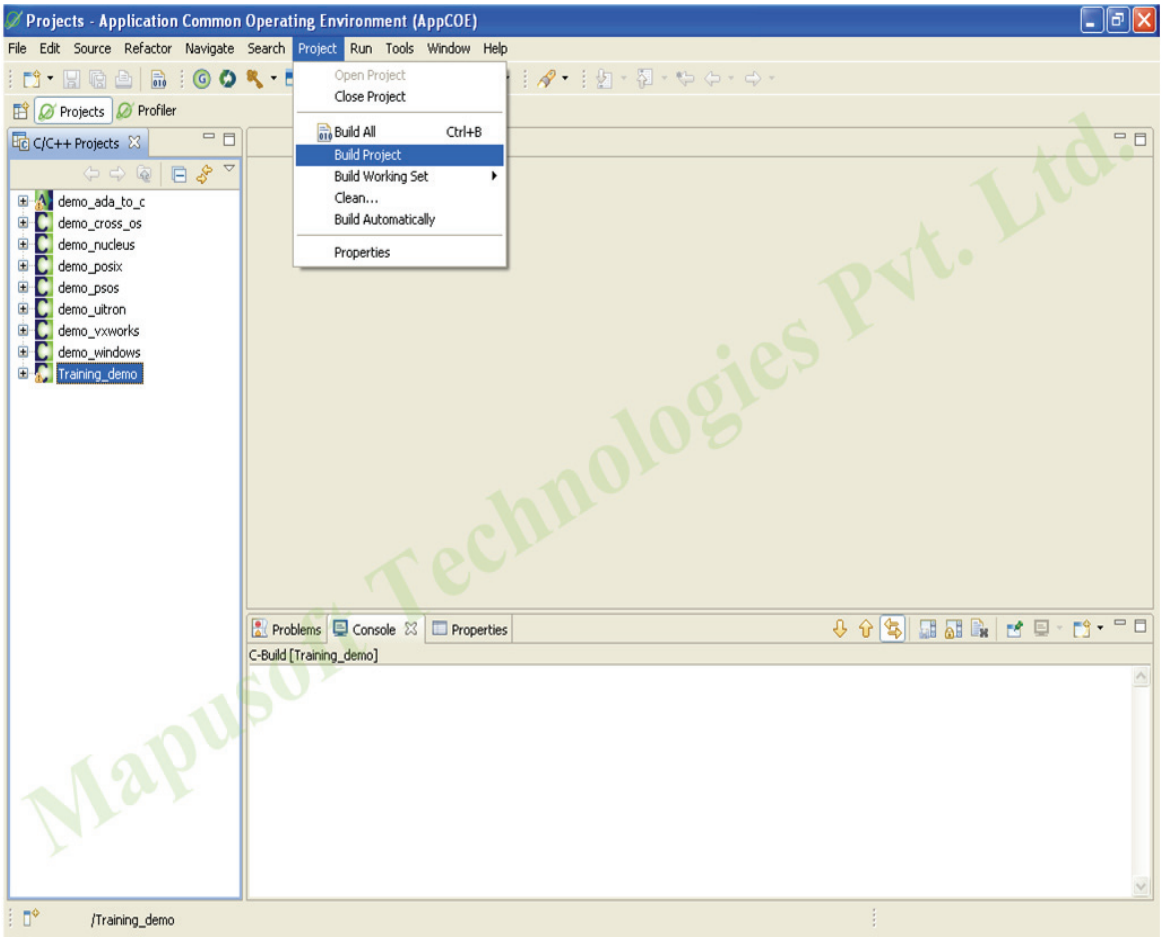


Please Note: While importing please make sure that the you are selecting a valid C/C++ file or a proper project with ".project "file. Otherwise AppCOE won`t be able to import any other invalid file.

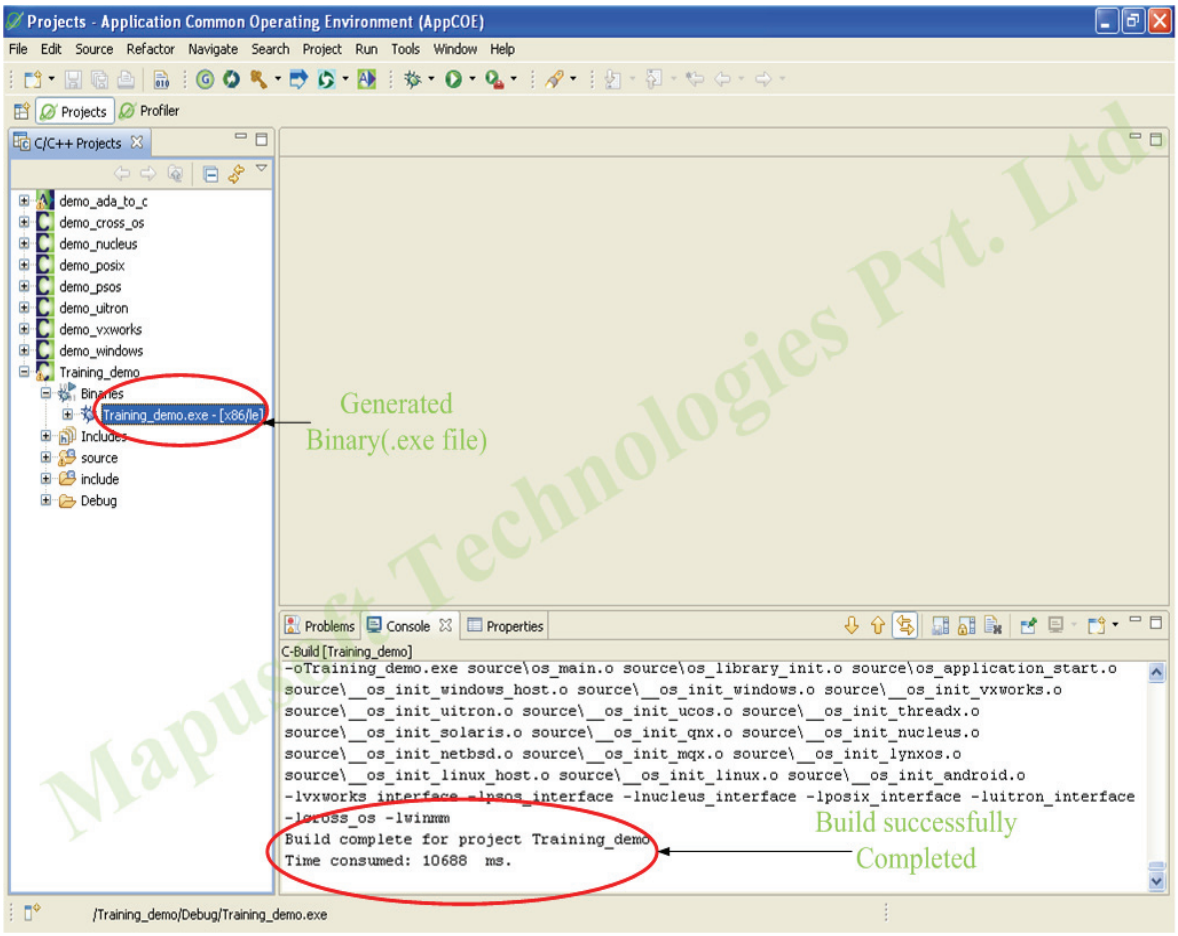
2.3 Building, Running and Debugging Projects.

Building Projects

1. Once all the settings have been done, we have to build and "exe". For this just select your project and **Goto Project > Build Project.**
    a. The Console window will show the progress during build operation.

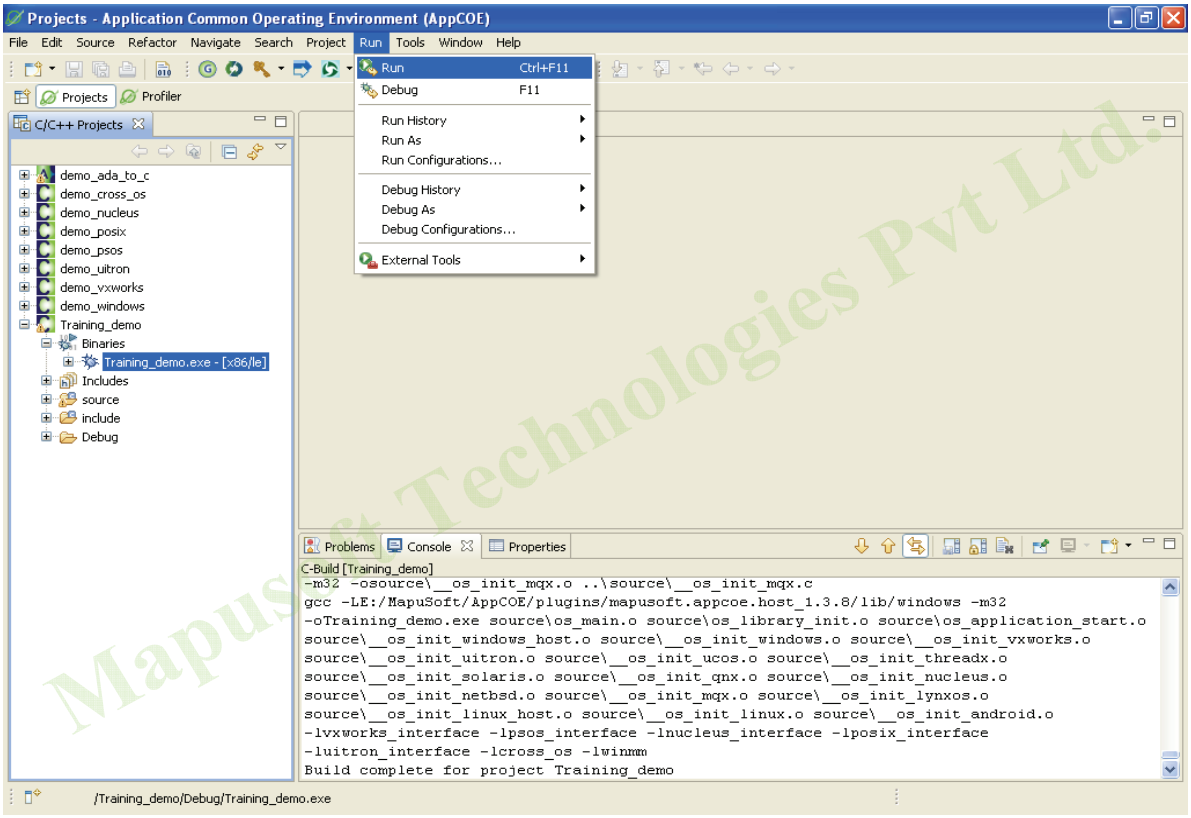


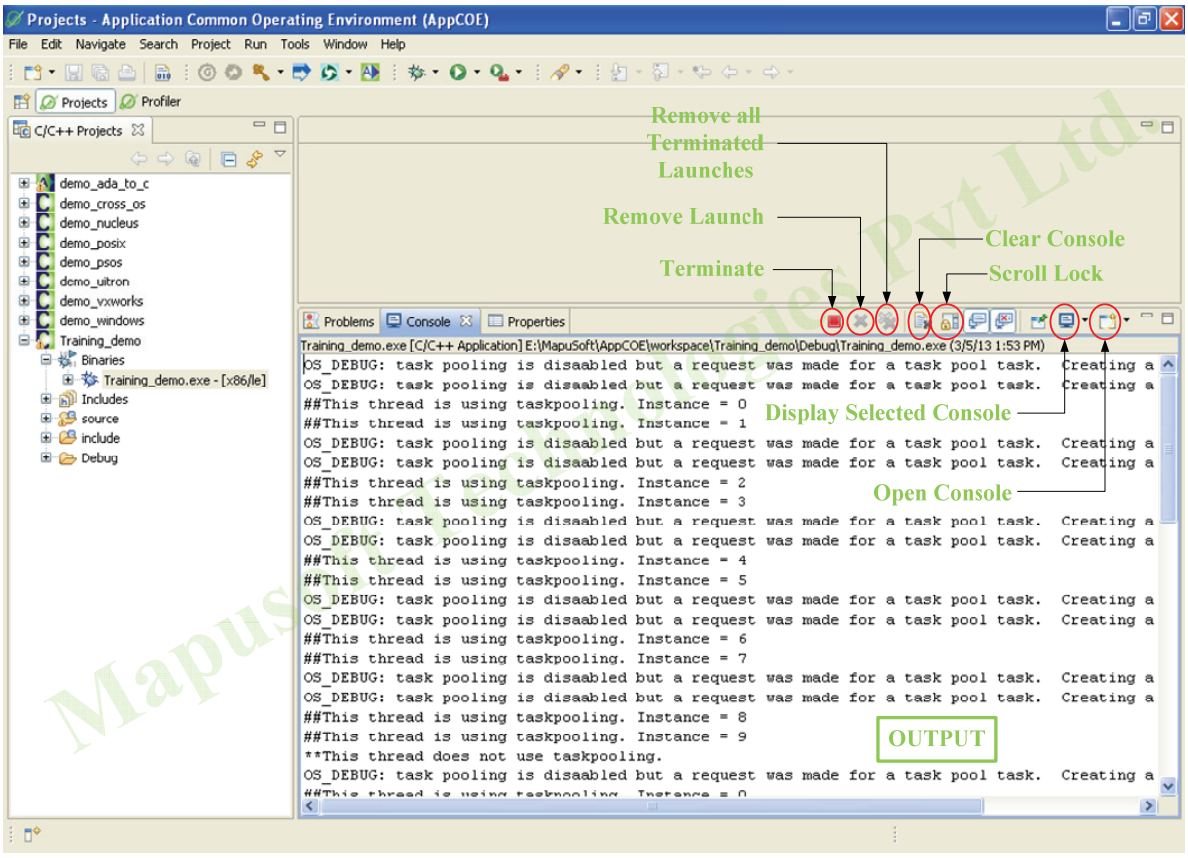
2. If there is some problem in the settings or some compiler error, the build will be stopped automatically. The console will report the same along with the error due to which the build has been stopped.
3. If no problem is encountered, the build will end successfully and a binary file (.exe file) will be generated. The console will show the acknowledgment of the same.



Running Projects

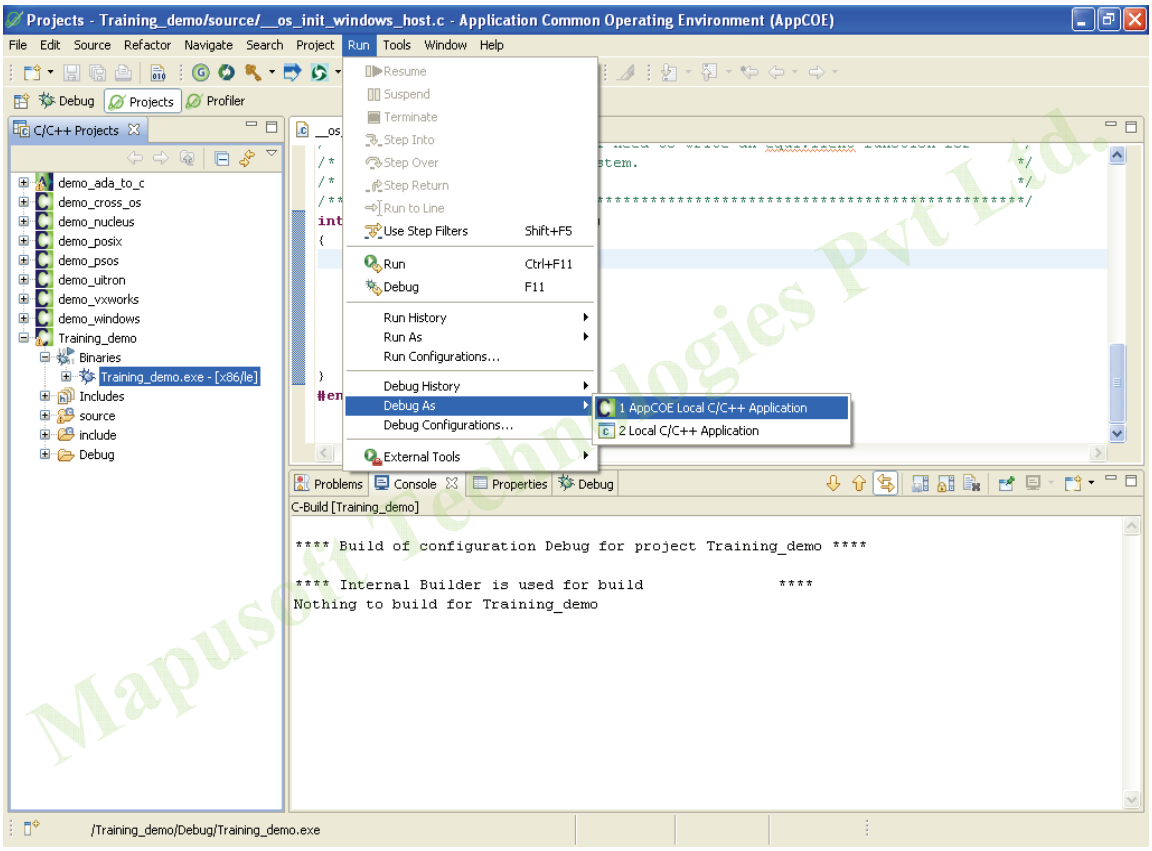
To run a built project, select the project **Goto Run > Run or click on**  **button.** The project will start running and the output will be displayed into the **Console Window.**



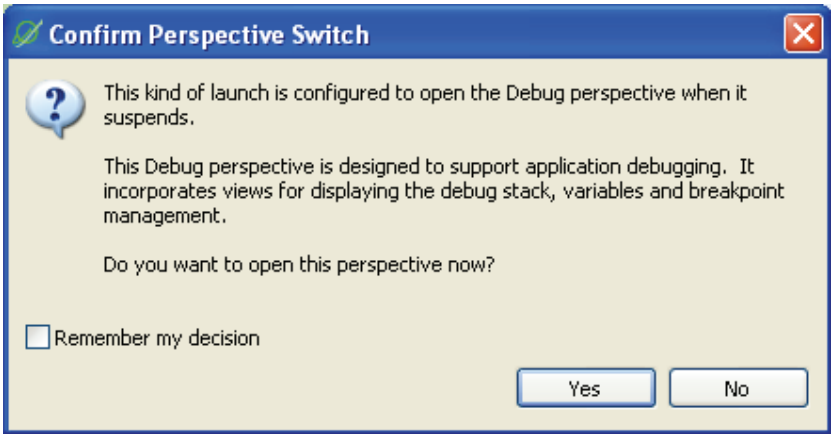


Debugging the project

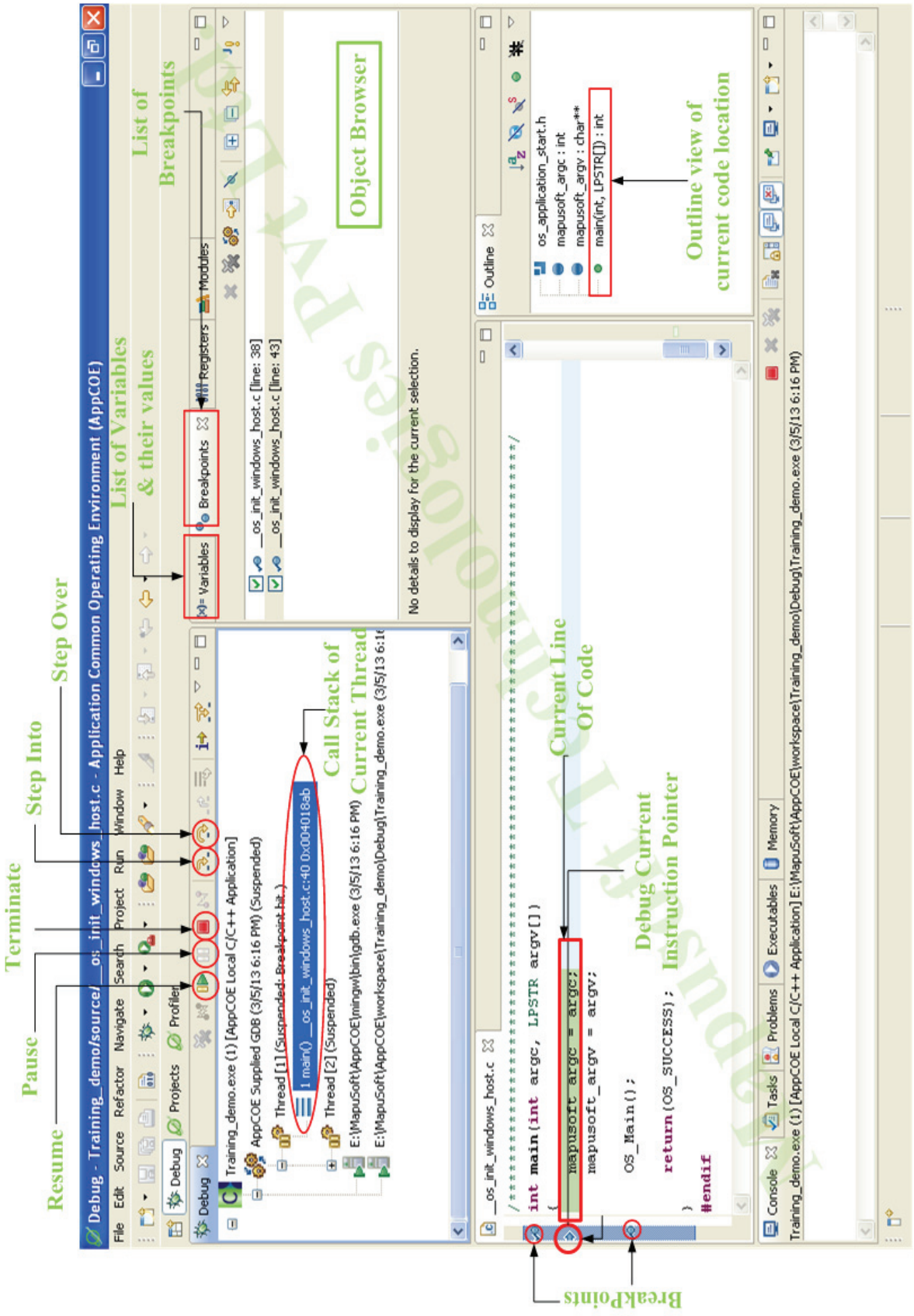
To run the application in debugging mode, select the respective project and **Goto Run > Debug As > AppCOE Local C/C++ Application or click on**  **button..**



a. A **"Confirm Perspective Switch"** dialog Box appears. Click "Yes" to open the debug perspective.



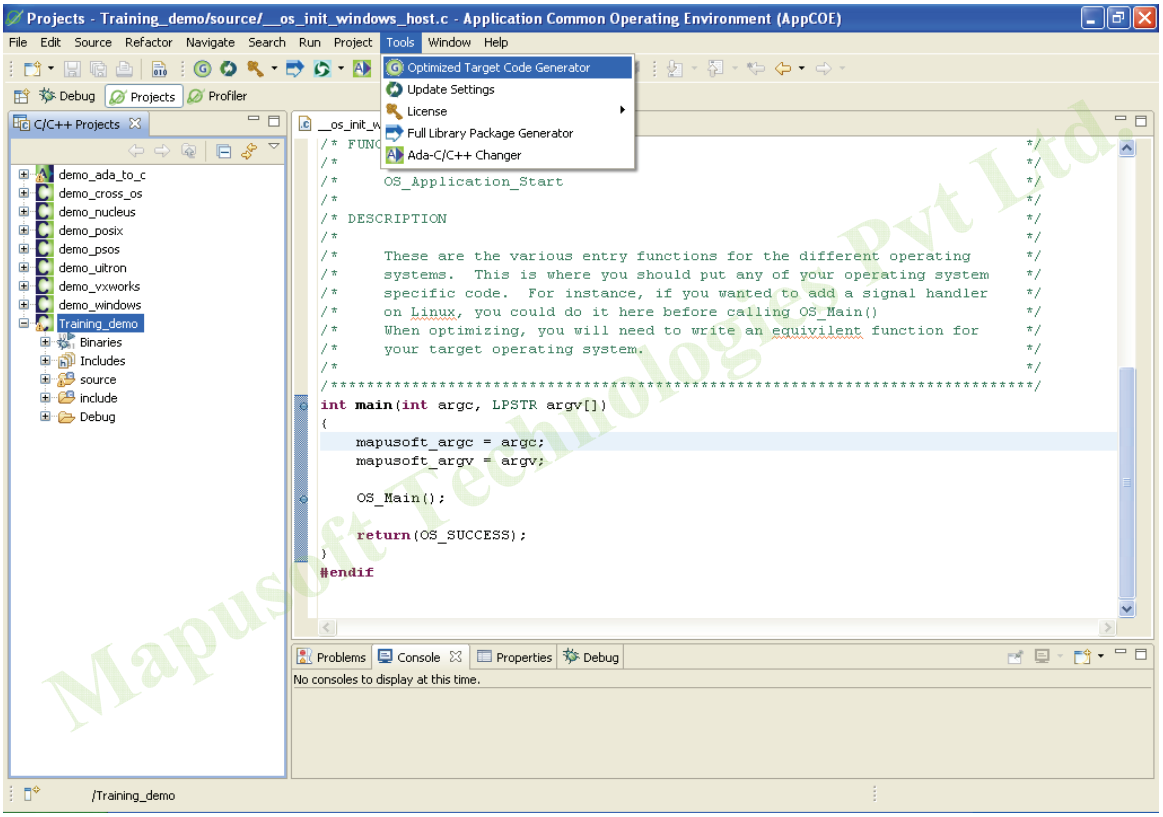
b. The debug perspective will open as below


Terminate
Step Into
Step Over
Pause
Resume
List of Variables & their values
List of Breakpoints
Object Browser
Call Stack of Current Thread
Current Line Of Code
Debug Current Instruction Pointer
Outline view of current code location
BreakPoints
Debug - Training_demo/source/__os_init_windows_host.c - Application Common Operating Environment (AppCOE)
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Projects
Profiler
Variables
Breakpoints
Registers
Modules
Outline
No details to display for the current selection.
Tasks
Problems
Executables
Memory
Console
os_application_start.h
mapusoft_argc : int
mapusoft_argv : char**
main(int, LPSTR[]) : int
int main(int argc, LPSTR argv[])
mapusoft_argc = argc;
mapusoft_argv = argv;
OS_Main();
return(OS_SUCCESS);
#endif

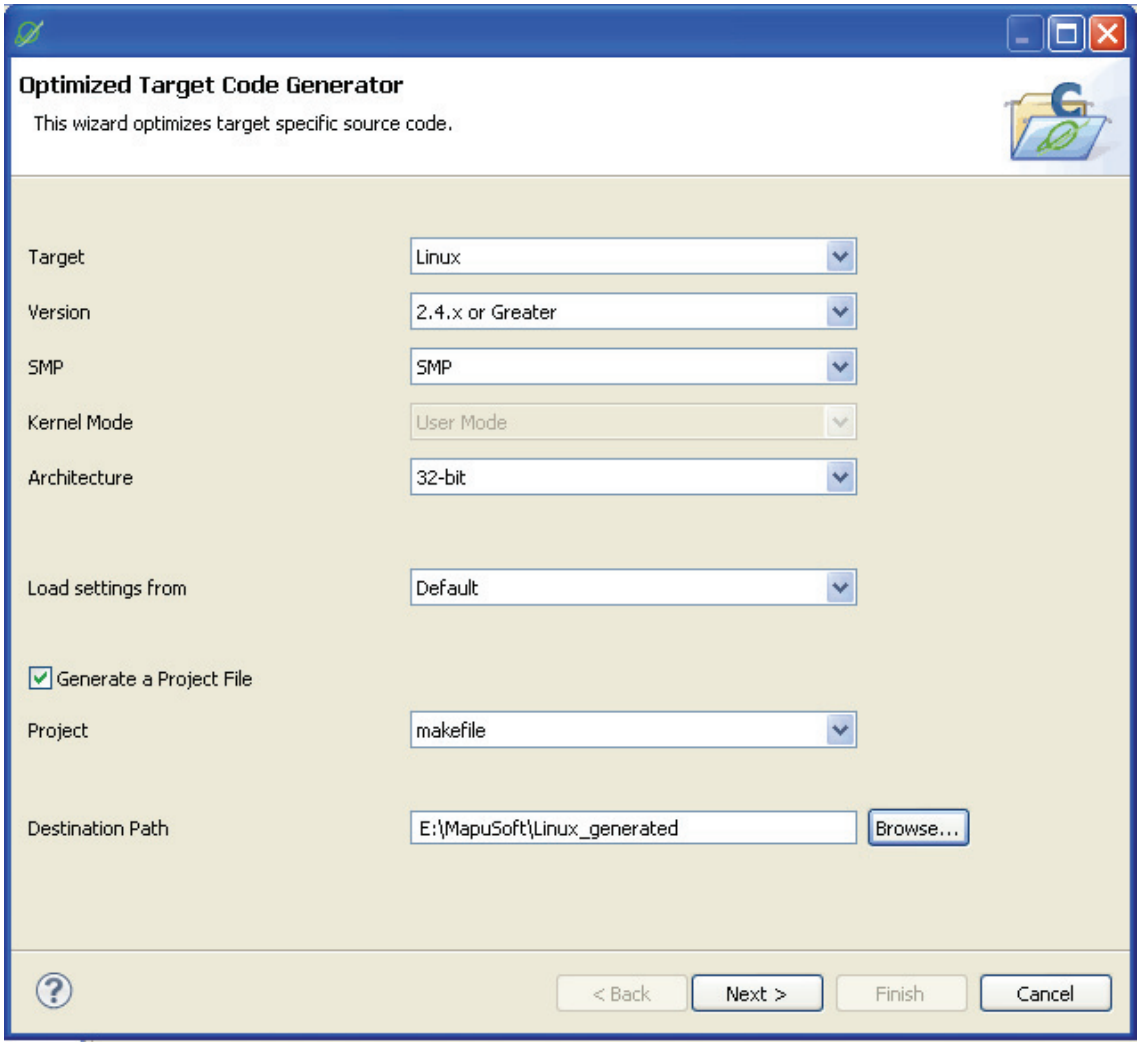
3. Target Code Generation

3.1 Optimize Target Code Generation

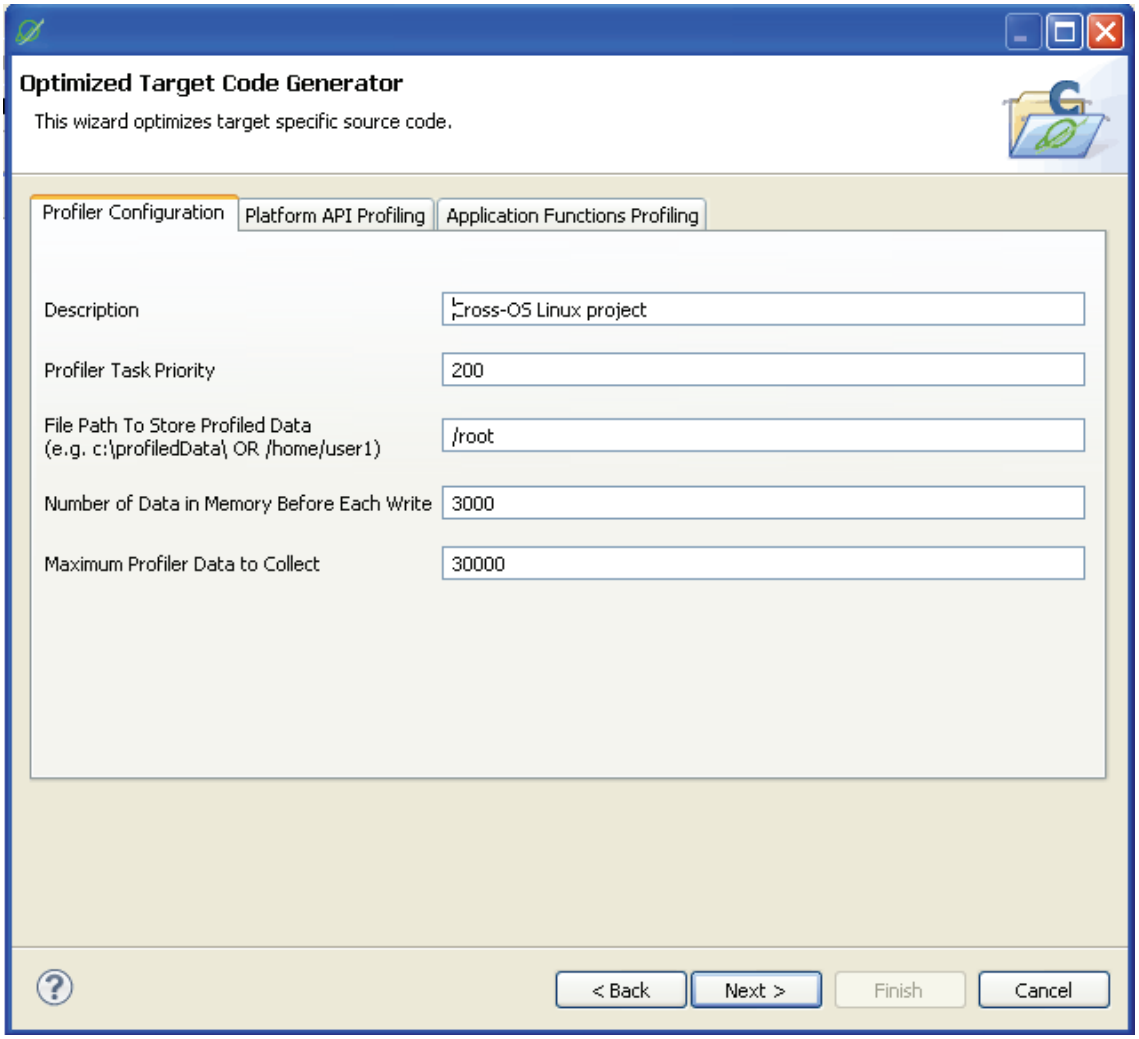
1. To migrate your current source code to different OS, select the project and **Goto Tools > Optimized Target Code Generator or click on**  **button.**



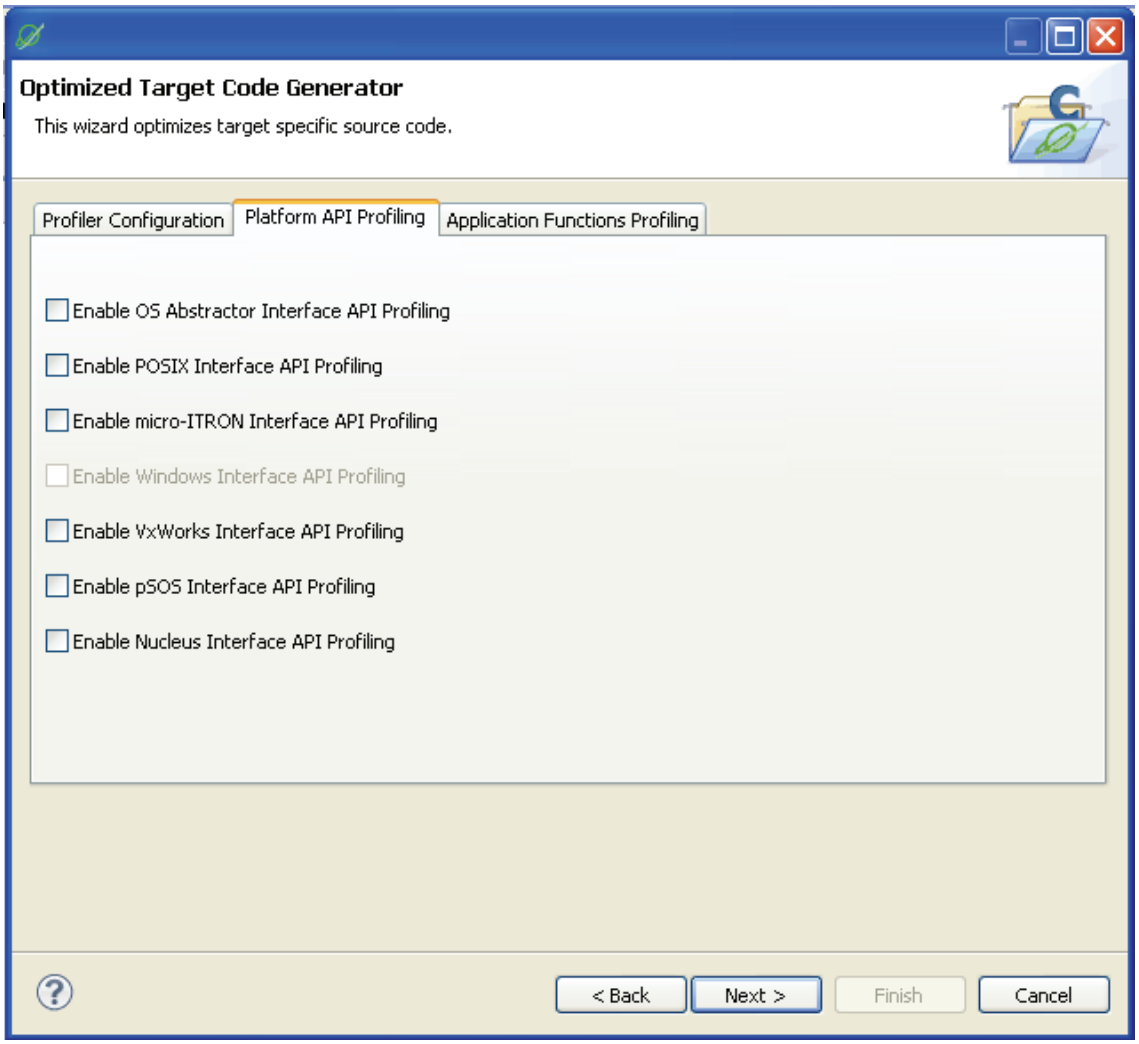
2. The "**Optimized Target Code Generator**" window will appear. Enter the appropriate data into the required fields and press **Next.**



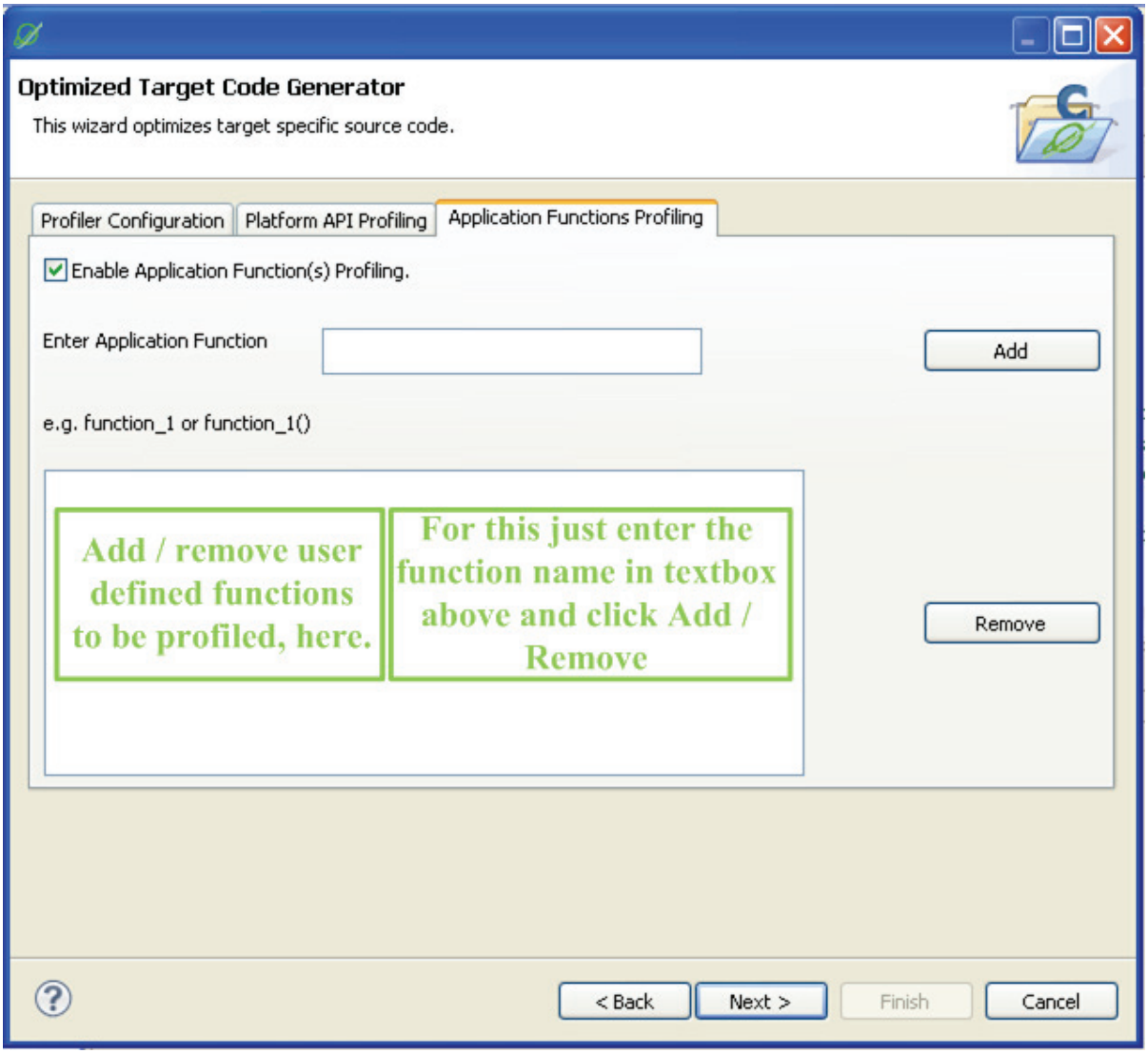
3. A new wizard for Profiler settings will appear. Define your profiler data specifications in the Profiler Configuration Tab.



4. On **Platform API Profiling** tab, select the check box to enable your appropriate Interface API Profiling

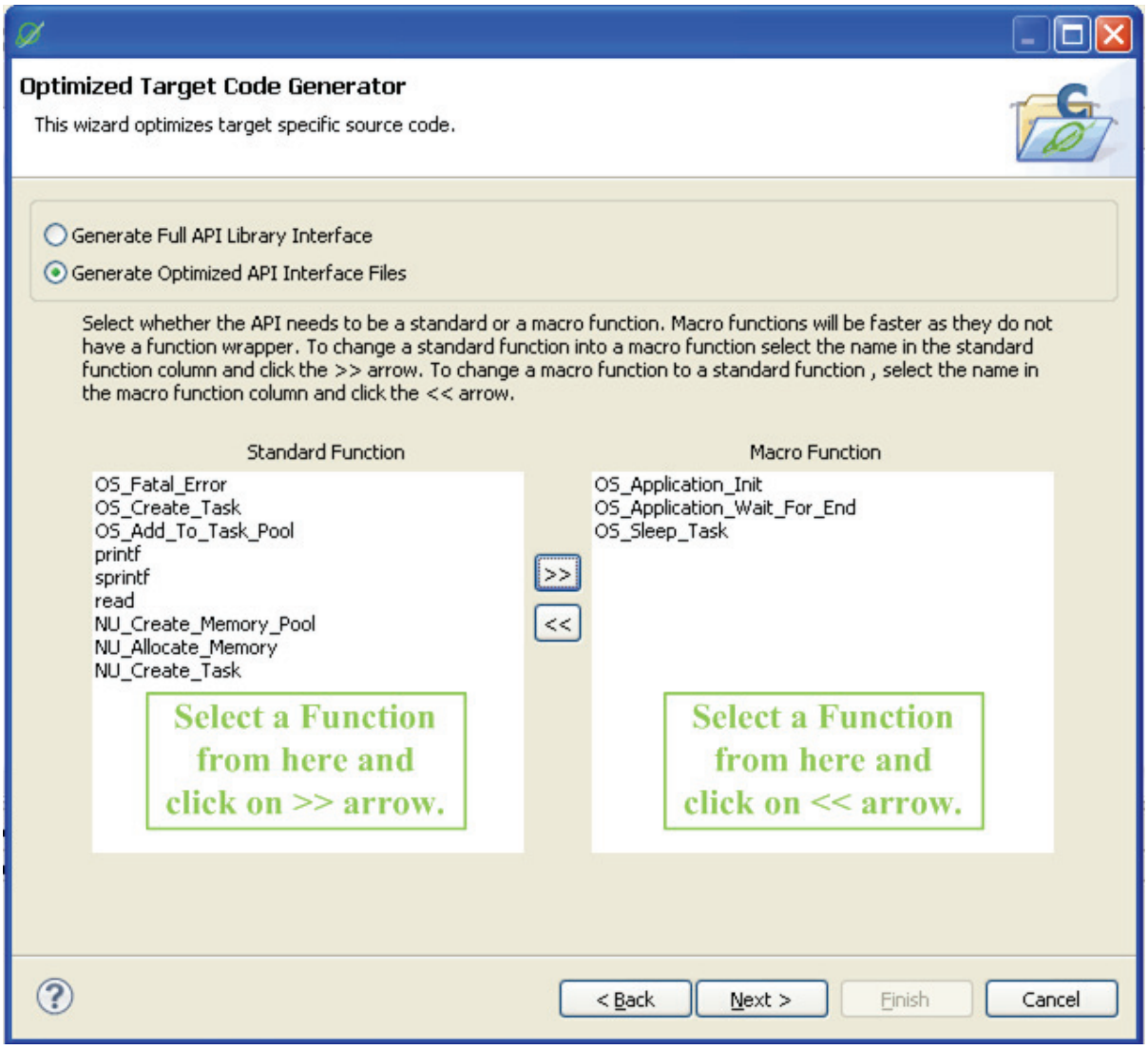


5. On **Application Functions Profiling** tab, you can also perform profiling for your specific APIs. Once done click on Next.



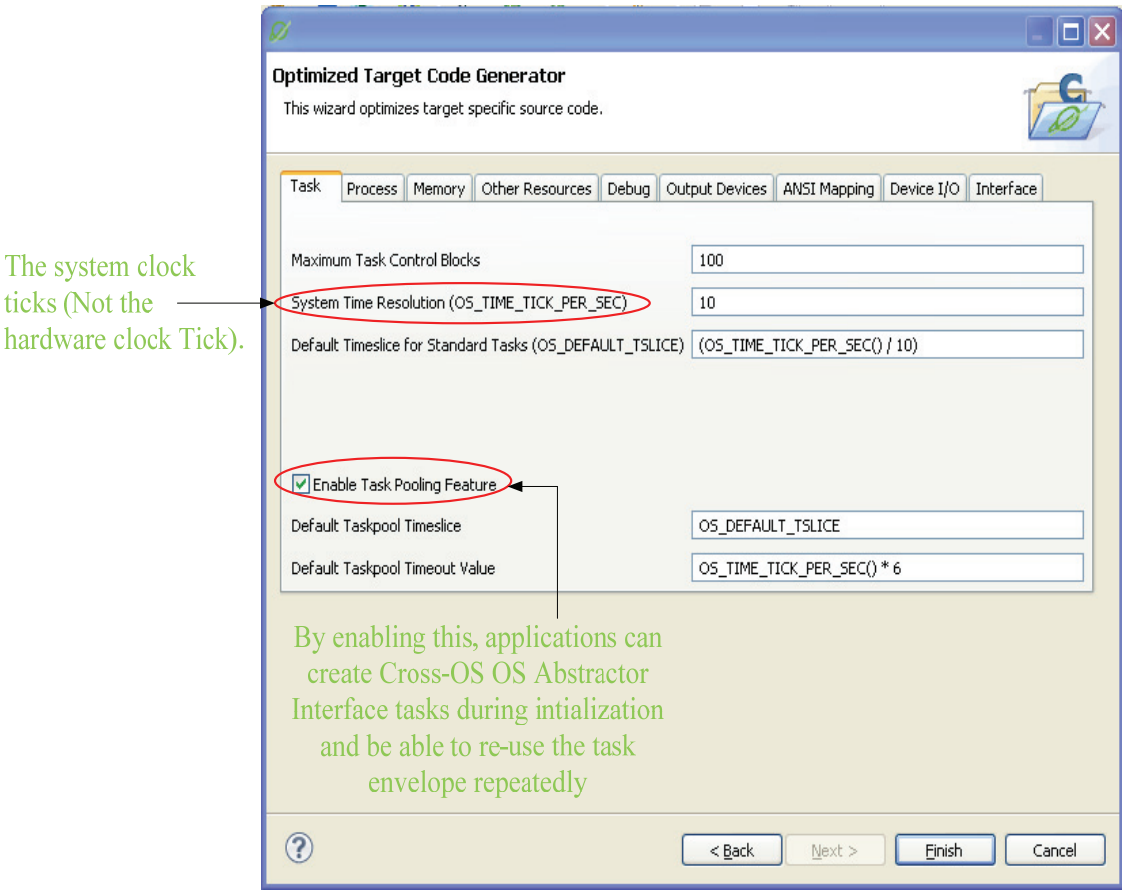
6. On **API Optimization** tab, you can select either to generate Full API Library Interface or Optimized API interface.

In API Optimization, you can select the API's which needs to be a standard function or a macro function and move the selected API using the double arrow button as shown below. **Macro functions will execute faster but will increase the memory footprint of the application.** Once done click on **Next.**



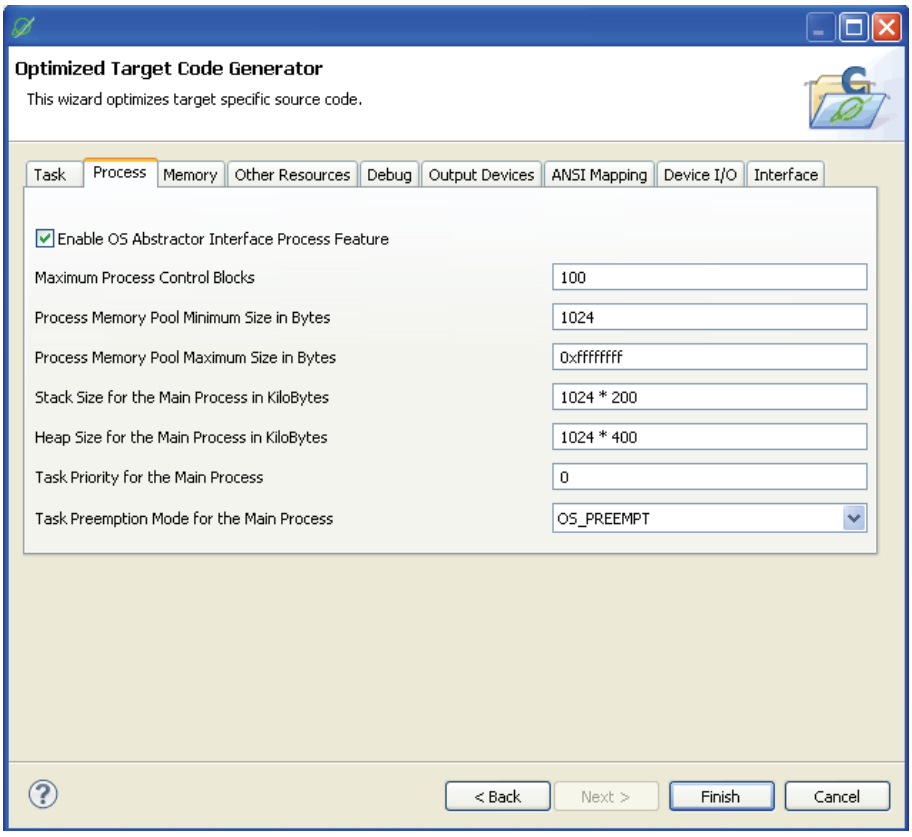
7. Next you are directed to Task Configuration Wizard as below:

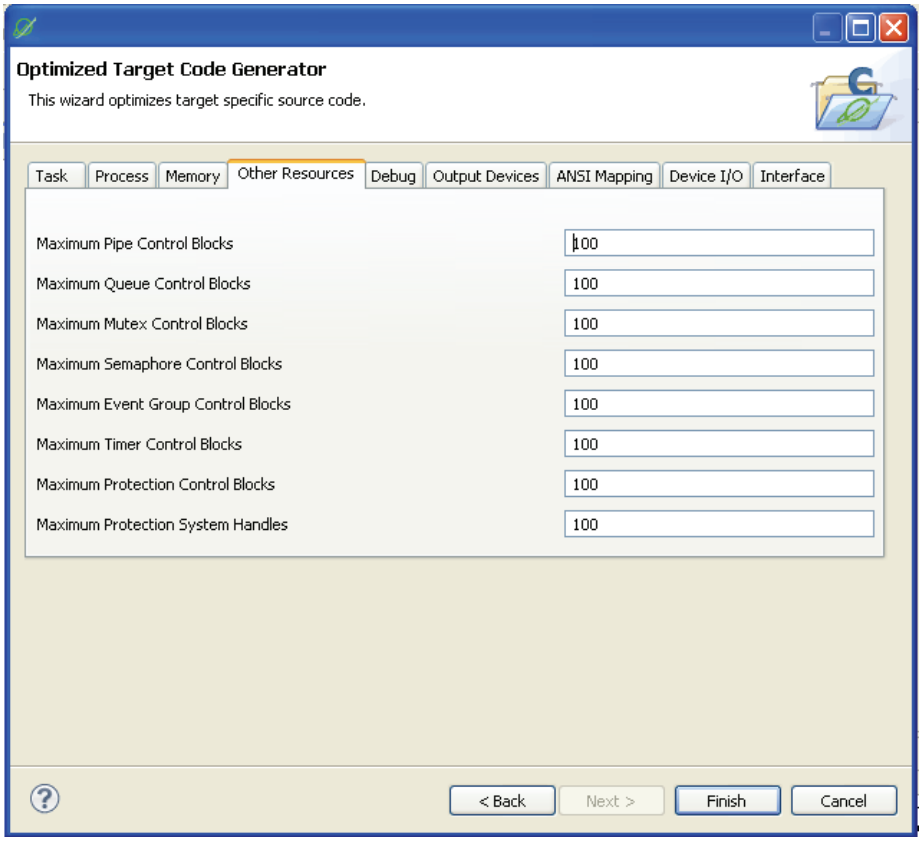
   Configure the following below settings as per your requirements.

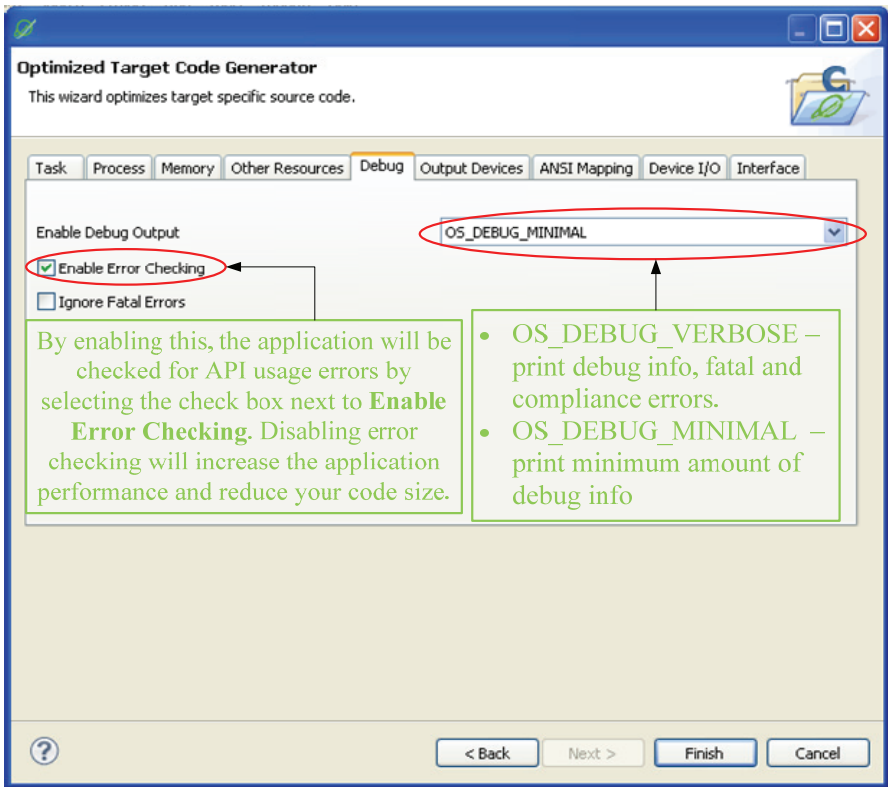


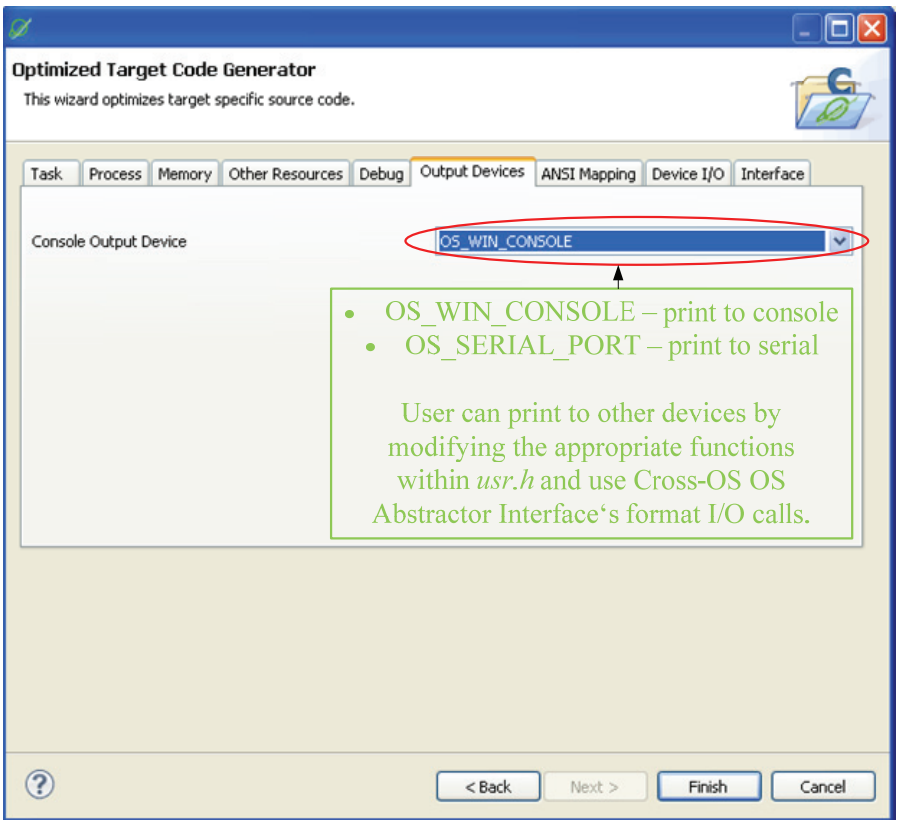
The system clock ticks (Not the hardware clock Tick).

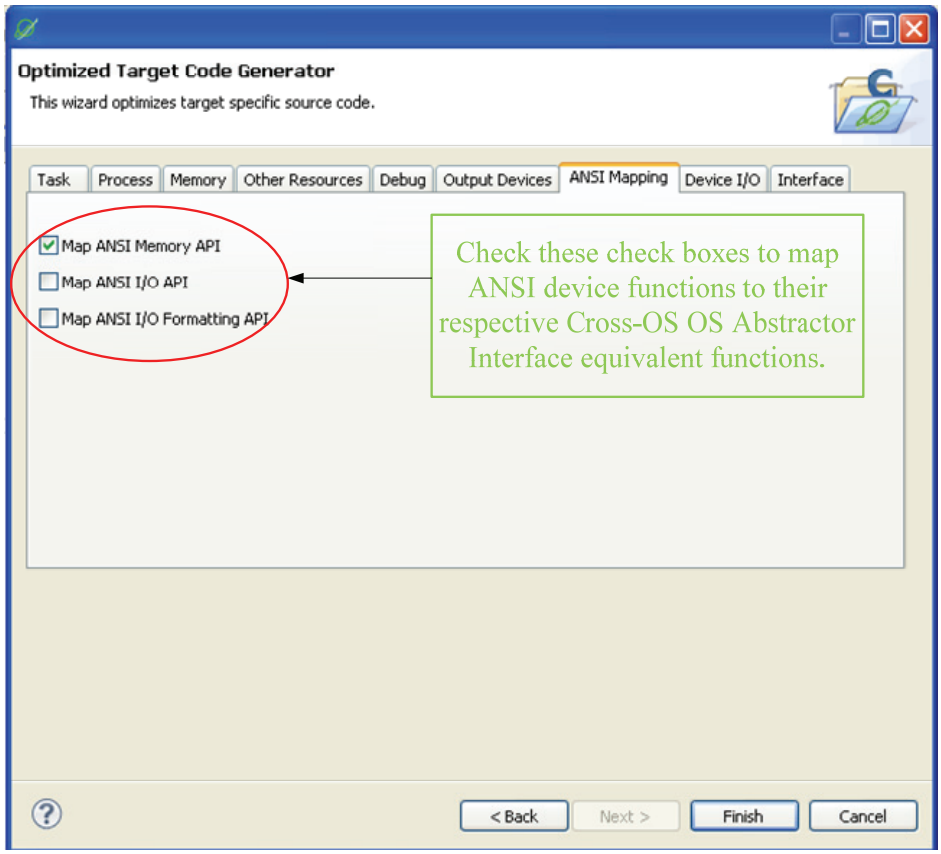
By enabling this, applications can create Cross-OS OS Abstractor Interface tasks during intialization and be able to re-use the task envelope repeatedly

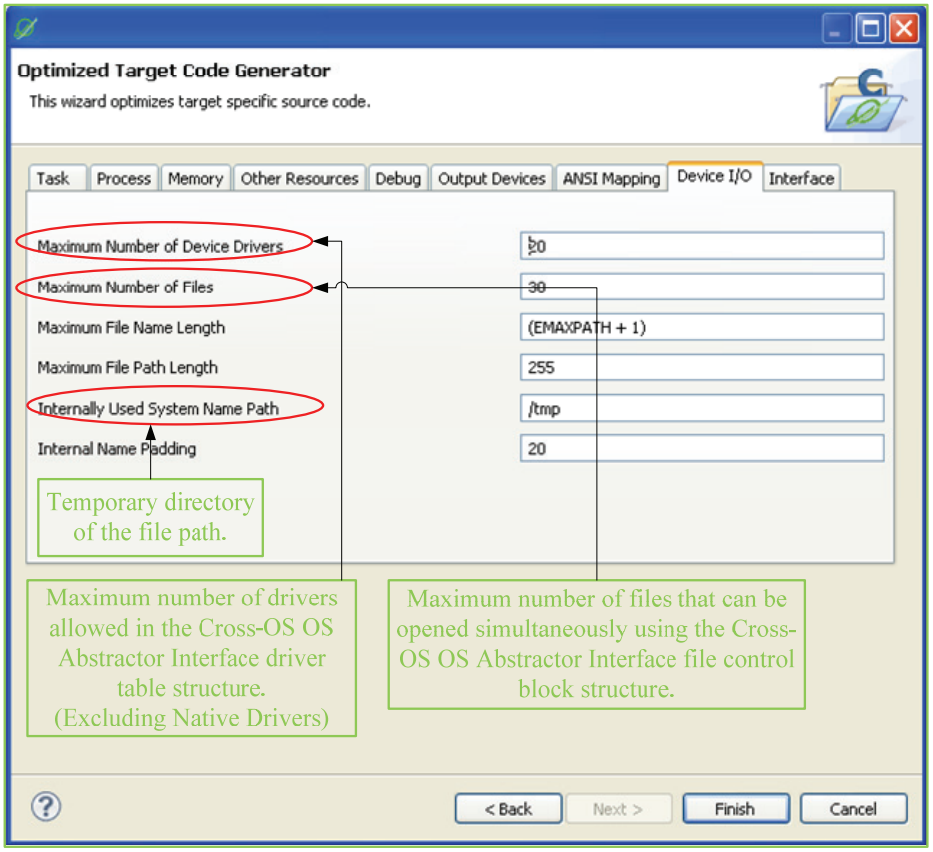

Optimized Target Code Generator
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
Interface
Enable OS Abstractor Interface Process Feature
Maximum Process Control Blocks
100
Process Memory Pool Minimum Size in Bytes
1024
Process Memory Pool Maximum Size in Bytes
0xffffffff
Stack Size for the Main Process in KiloBytes
1024 * 200
Heap Size for the Main Process in KiloBytes
1024 * 400
Task Priority for the Main Process
0
Task Preemption Mode for the Main Process
OS_PREEMPT
< Back
Next >
Finish
Cancel

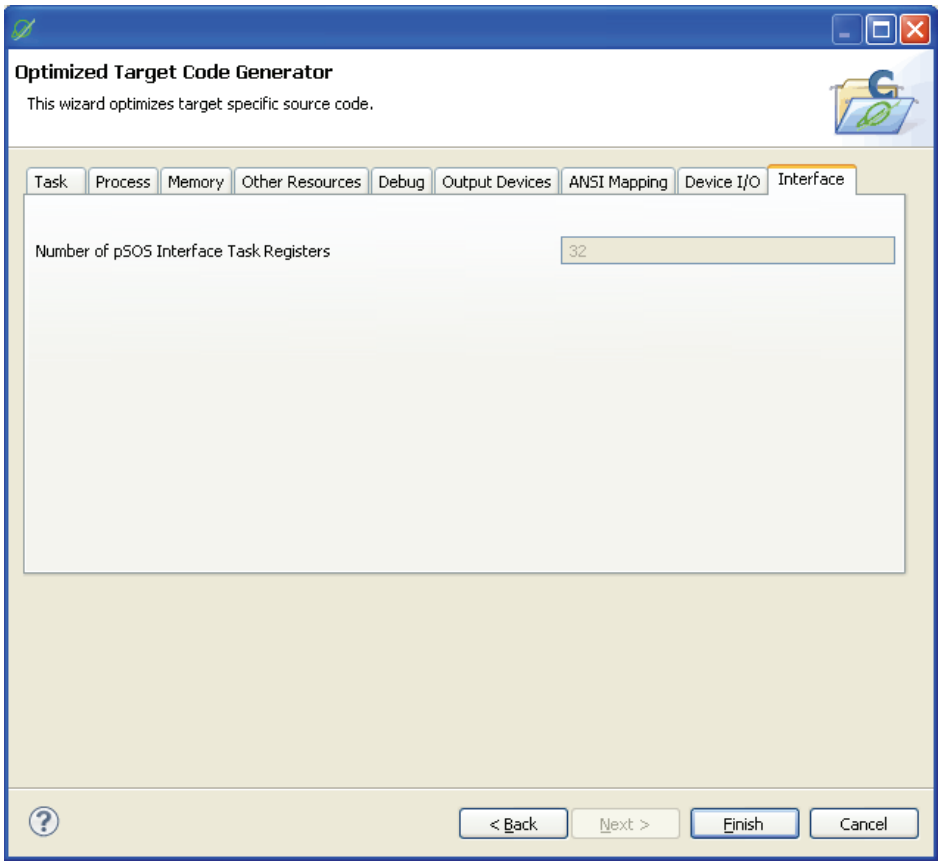

Optimized Target Code Generator
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
Interface
Maximum Pipe Control Blocks
100
Maximum Queue Control Blocks
100
Maximum Mutex Control Blocks
100
Maximum Semaphore Control Blocks
100
Maximum Event Group Control Blocks
100
Maximum Timer Control Blocks
100
Maximum Protection Control Blocks
100
Maximum Protection System Handles
100
< Back
Next >
Finish
Cancel


Optimized Target Code Generator
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
Interface
Enable Debug Output
OS_DEBUG_MINIMAL
Enable Error Checking
Ignore Fatal Errors
By enabling this, the application will be checked for API usage errors by selecting the check box next to Enable Error Checking. Disabling error checking will increase the application performance and reduce your code size.
OS_DEBUG_VERBOSE – print debug info, fatal and compliance errors.
OS_DEBUG_MINIMAL – print minimum amount of debug info
< Back
Next >
Finish
Cancel

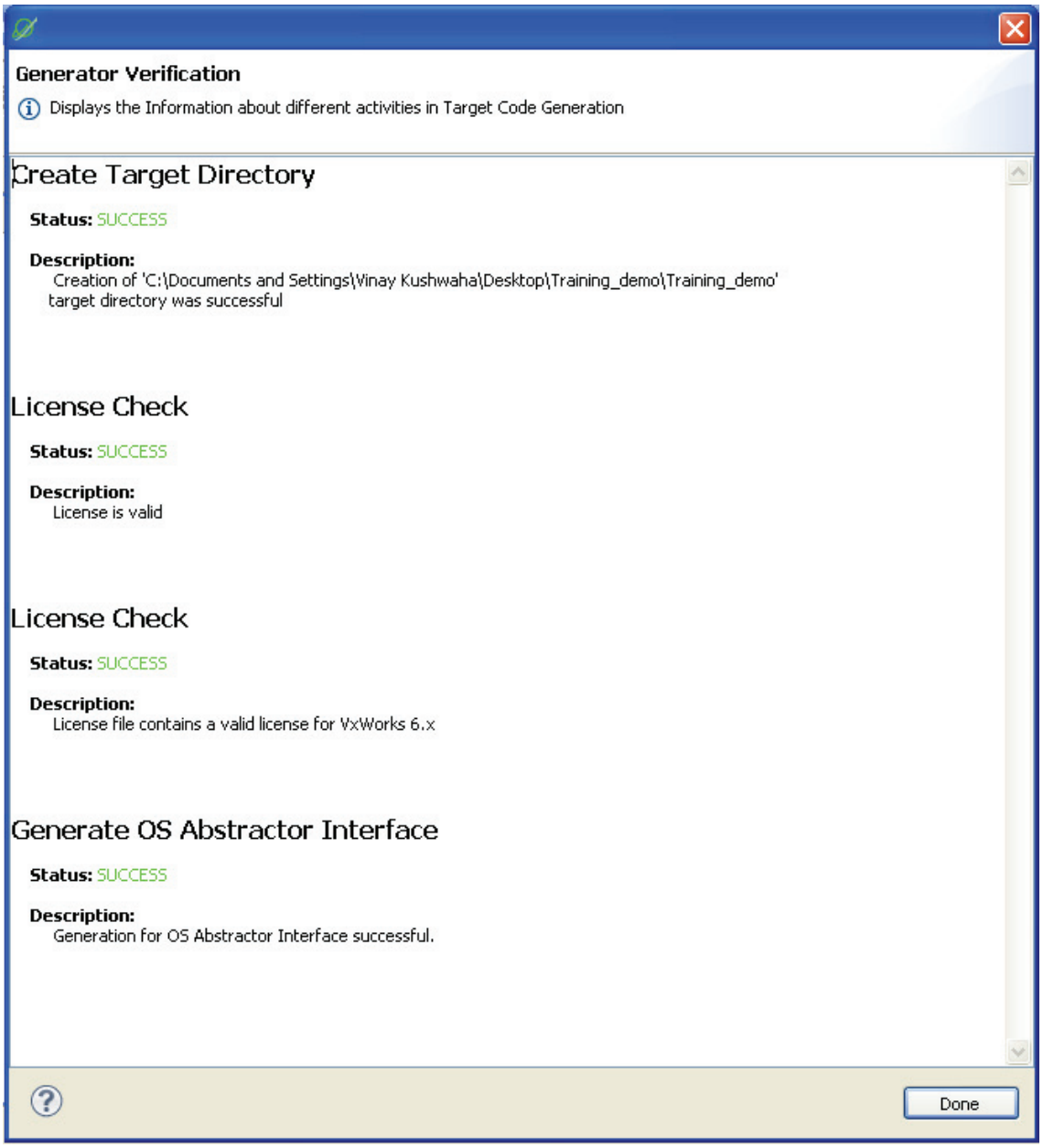

Optimized Target Code Generator
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
Interface
Console Output Device
OS_WIN_CONSOLE
OS_WIN_CONSOLE – print to console
OS_SERIAL_PORT – print to serial
User can print to other devices by modifying the appropriate functions within usr.h and use Cross-OS OS Abstractor Interface's format I/O calls.
< Back
Next >
Finish
Cancel


Optimized Target Code Generator
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
Interface
Map ANSI Memory API
Map ANSI I/O API
Map ANSI I/O Formatting API
Check these check boxes to map ANSI device functions to their respective Cross-OS OS Abstractor Interface equivalent functions.
< Back
Next >
Finish
Cancel


Optimized Target Code Generator
This wizard optimizes target specific source code.
Task
Process
Memory
Other Resources
Debug
Output Devices
ANSI Mapping
Device I/O
Interface
Maximum Number of Device Drivers
20
Maximum Number of Files
30
Maximum File Name Length
(EMAXPATH + 1)
Maximum File Path Length
255
Internally Used System Name Path
/tmp
Internal Name Padding
20
Temporary directory of the file path.
Maximum number of drivers allowed in the Cross-OS OS Abstractor Interface driver table structure. (Excluding Native Drivers)
Maximum number of files that can be opened simultaneously using the Cross-OS OS Abstractor Interface file control block structure.
< Back
Next >
Finish
Cancel



8. Once you are done, Click on Finish to Generate the Target Code. After generation of Code, a Generation Verification window will appear acknowledging the success of code generation. Click on done.

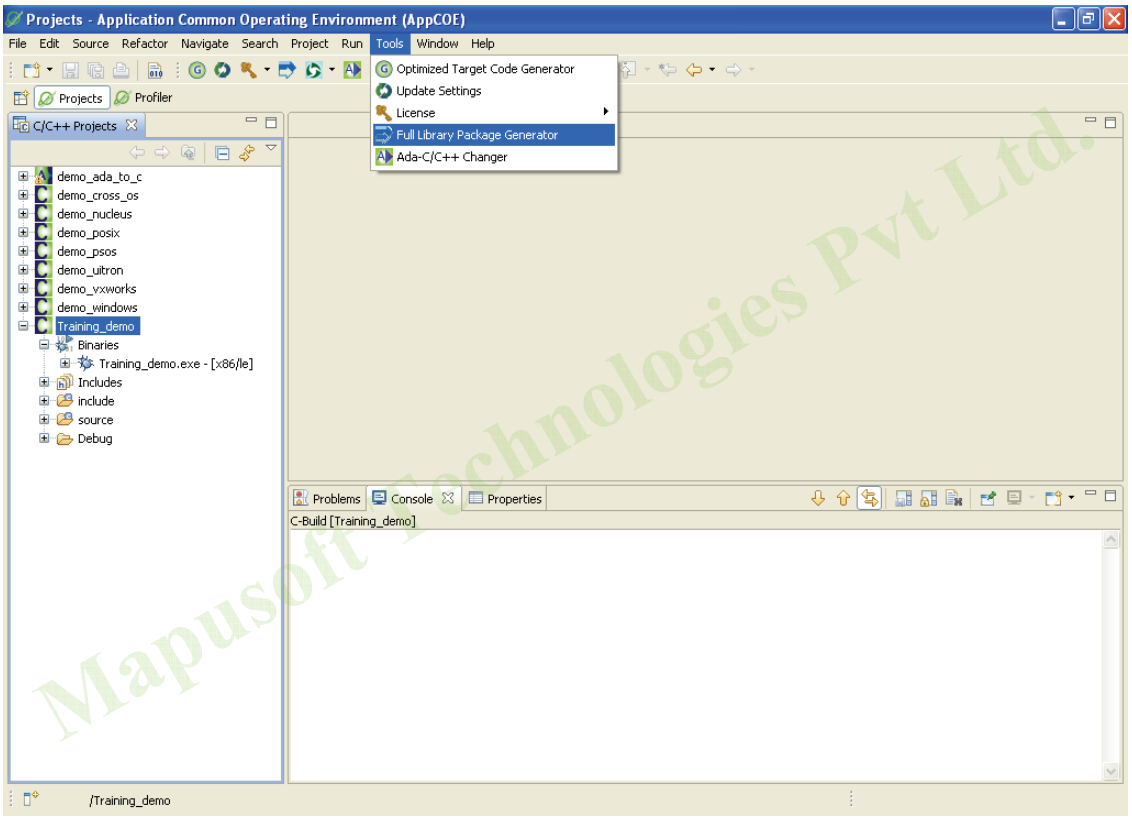


Now you have obtained code ready to build and run on your target platform.

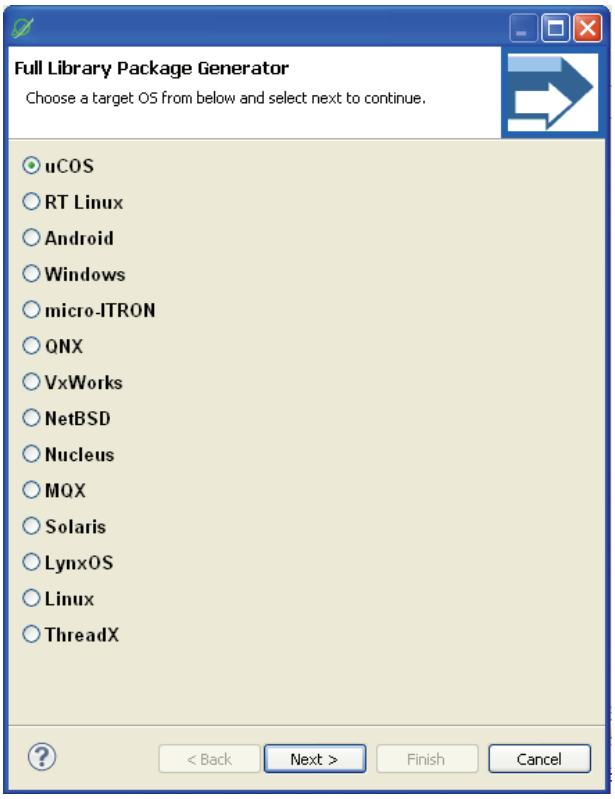
Please Note: While doing above exercises if you require much in depth explanation of the various attributes and parameters that you are setting in the GUI, then please refer to the topic **"Optimize Target Code Generation" on Pg 122 in "AppCOE User Manual".**

3.2 Full Package Library Generator

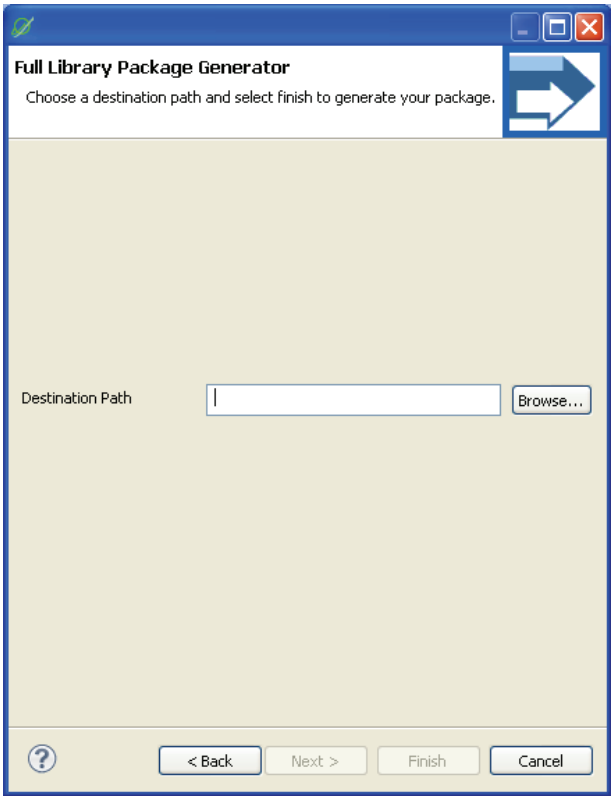
1. To generate full library package **Goto Tools > Full Library Package Generator.**



2. Then in proceeding window, select the Target OS for which you want to generate full library package.



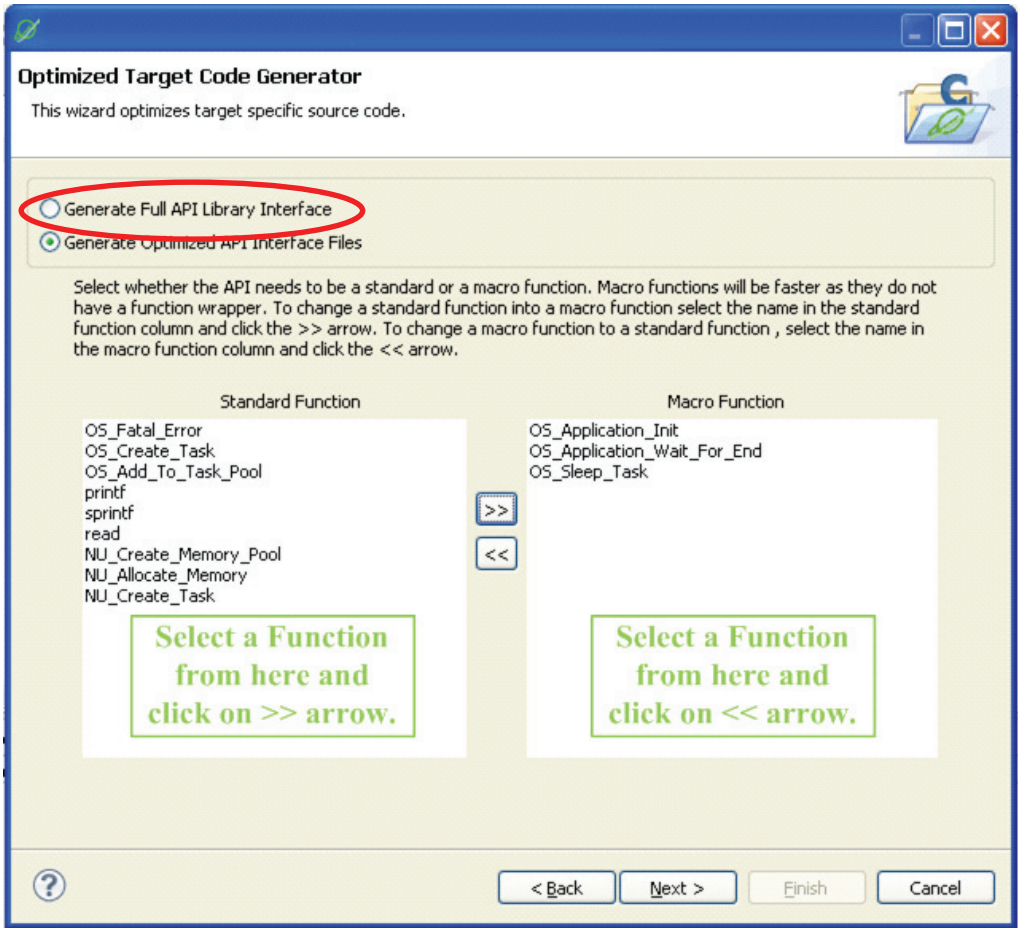
3. Then enter the Destination path to save the generated code and click on **Finish**



4. The finally generated files will be saved at the directory entered above.

Also while generating "Optimized Target Code", you can create "Full API Library Interface" as in below image. The difference is that by using "Full Library Package Generator" you get whole package to create libraries and develop applications using your own IDE. The project can be manually configured and scale by modifying the user configuration file.

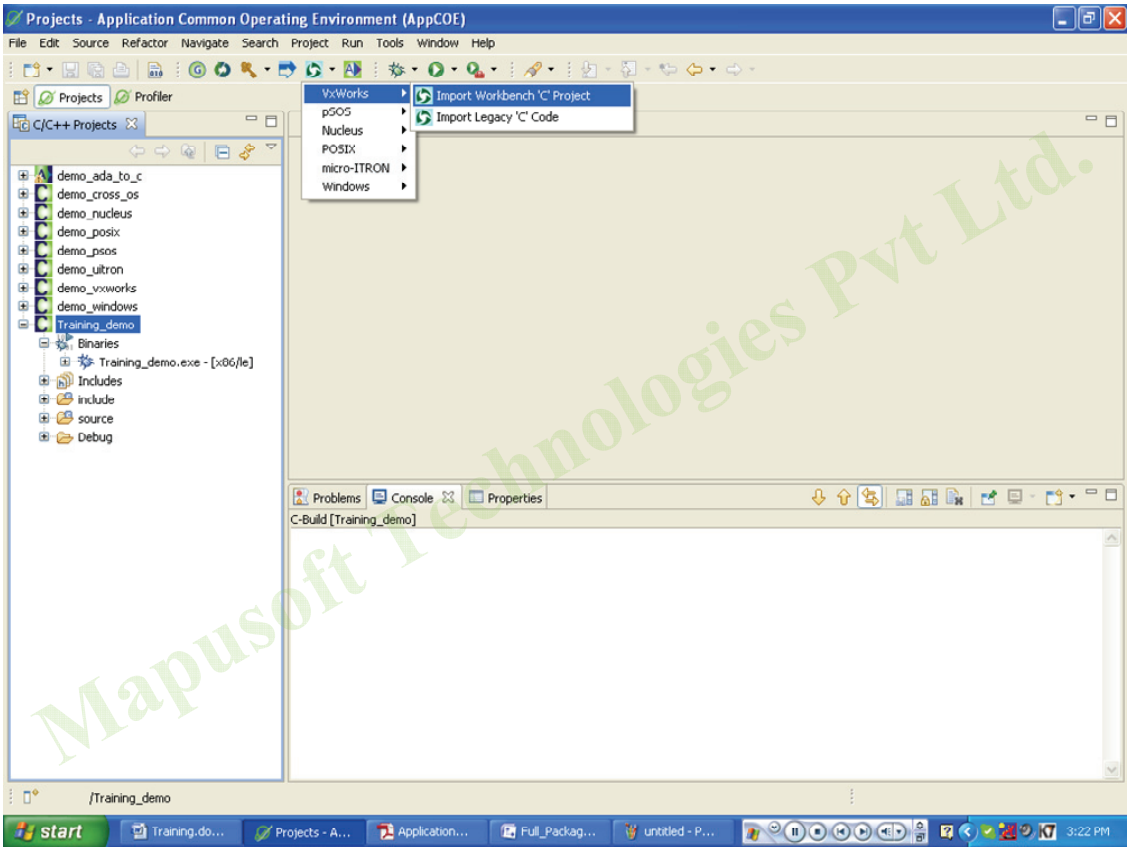
While using the feature as highlighted below you will get target specific Package. This will include you application; OS Abstractor files along with the Target specific Project file to directly import your project into the target Development platform.



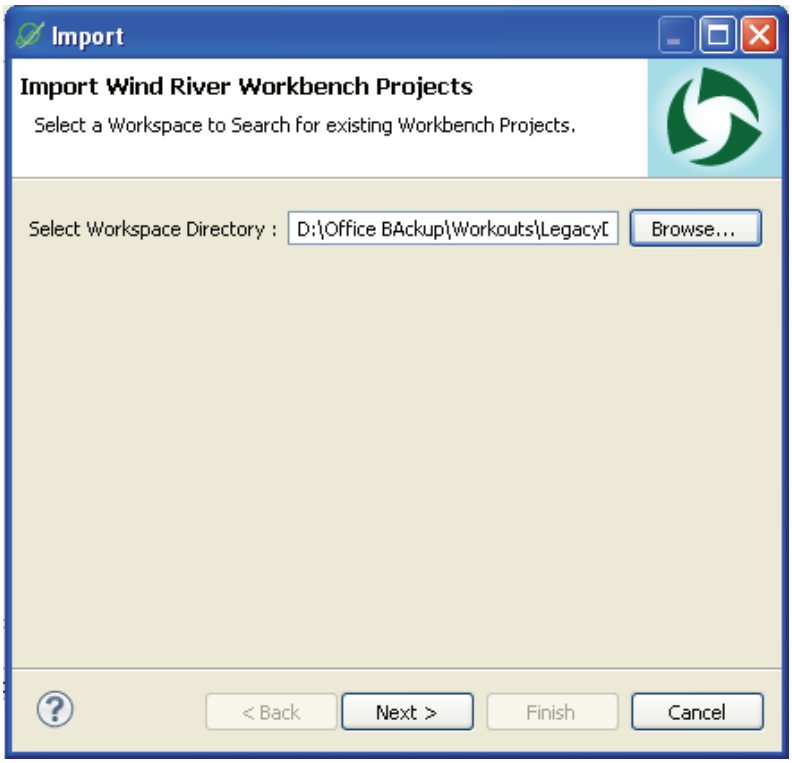
4. Legacy Porting Tool

4.1 Importing VxWorks workbench C/C++ Project.

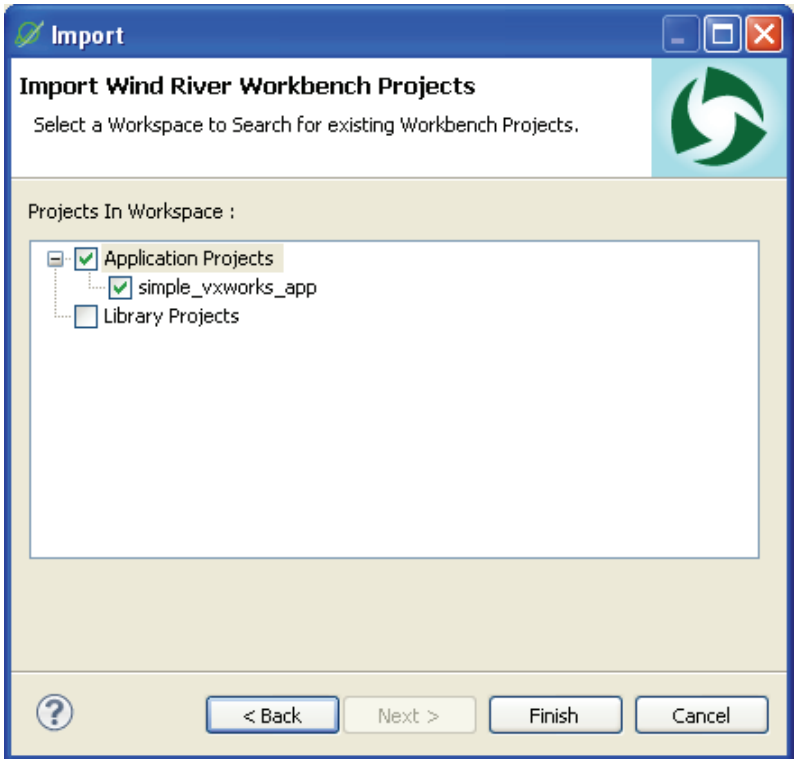
1. To import a VxWorks workbench C/C++ Project. Click on  button and then **Goto VxWorks > Import workbench 'C' Project.**



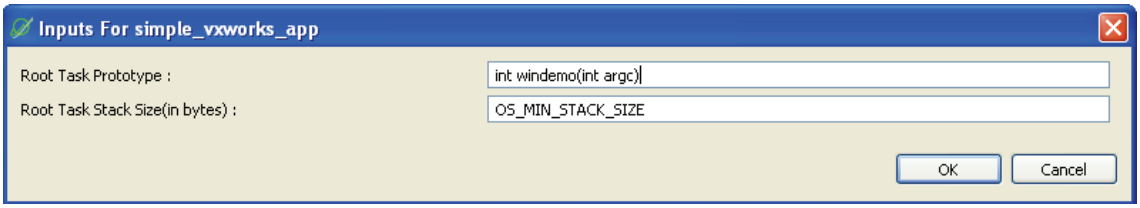
2. Then browse for the workspace directory where the VxWorks Workbench projects are present and click on **Next.**



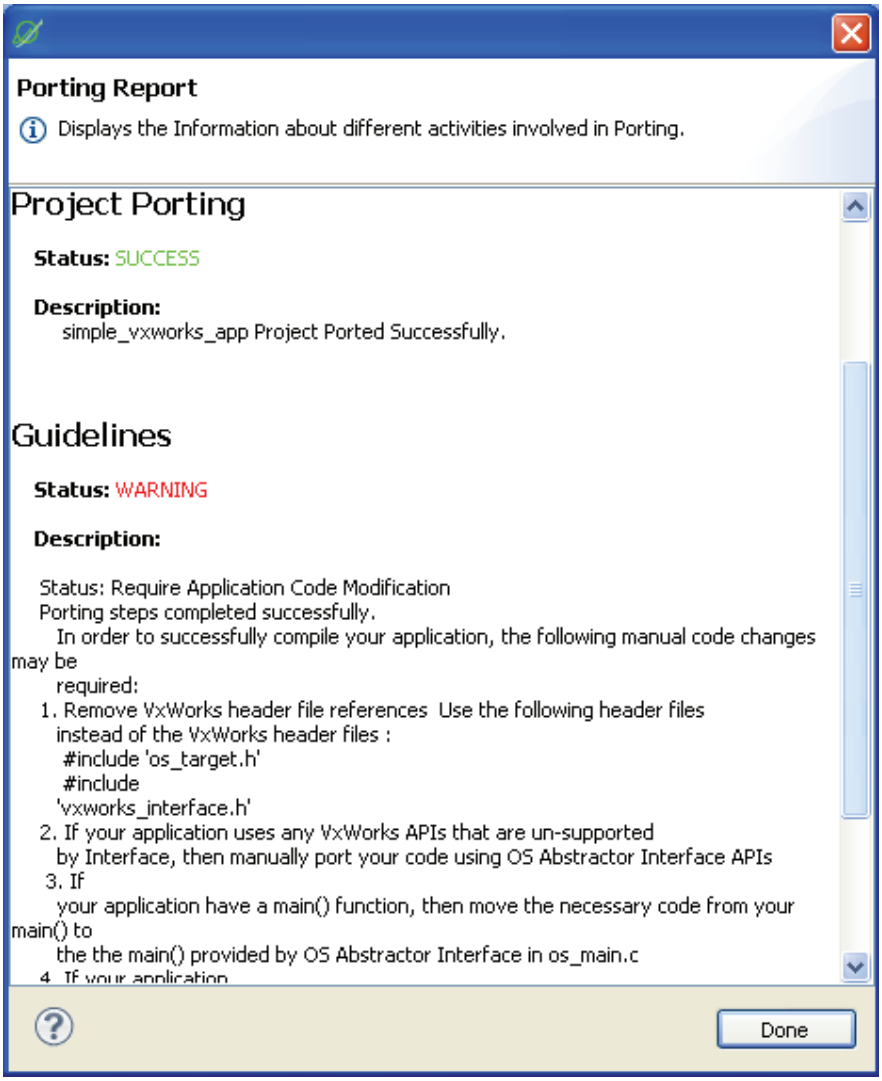
3. Select the appropriate Workbench projects and click **Next.**



4. Enter the **"Root Task Prototype"** and "**Root Task Stack Size (in bytes)"** in the following window. Then Click **OK** and then **Finish.**



5. A report will be generated, acknowledging the successful porting and describing certain fixes required. Read it and Click on **Done.**



6. Open you application files and put all **VxWorks** include files into comments. Also add Cross OS include files "os_target.h" and "vxworks_interface.h" as below. Save the file.

winddemo.c

```
/* BEGIN MAPUSOFT - comment out vxworks native header files */
#include "vxWorks.h"
#include "semLib.h"
#include "taskLib.h"
#include "msgQLib.h"
#include "wdLib.h"
#include "logLib.h"
#include "tickLib.h"
#include "sysLib.h"
#include "stdio.h"
#include "kernelLib.h"
/* END MAPUSOFT */

/* BEGIN MAPUSOFT - add mapusoft header files */
/*
#include "os_target.h"
#include "vxworks_interface.h"
*/
/* END MAPUSOFT */
```

Before

winddemo.c

```
/* BEGIN MAPUSOFT - comment out vxworks native header files */
/*
#include "vxWorks.h"
#include "semLib.h"
#include "taskLib.h"
#include "msgQLib.h"
#include "wdLib.h"
#include "logLib.h"
#include "tickLib.h"
#include "sysLib.h"
#include "stdio.h"
#include "kernelLib.h"
*/
/* END MAPUSOFT */

/* BEGIN MAPUSOFT - add mapusoft header files */
#include "os_target.h"
#include "vxworks_interface.h"
/* END MAPUSOFT */
```

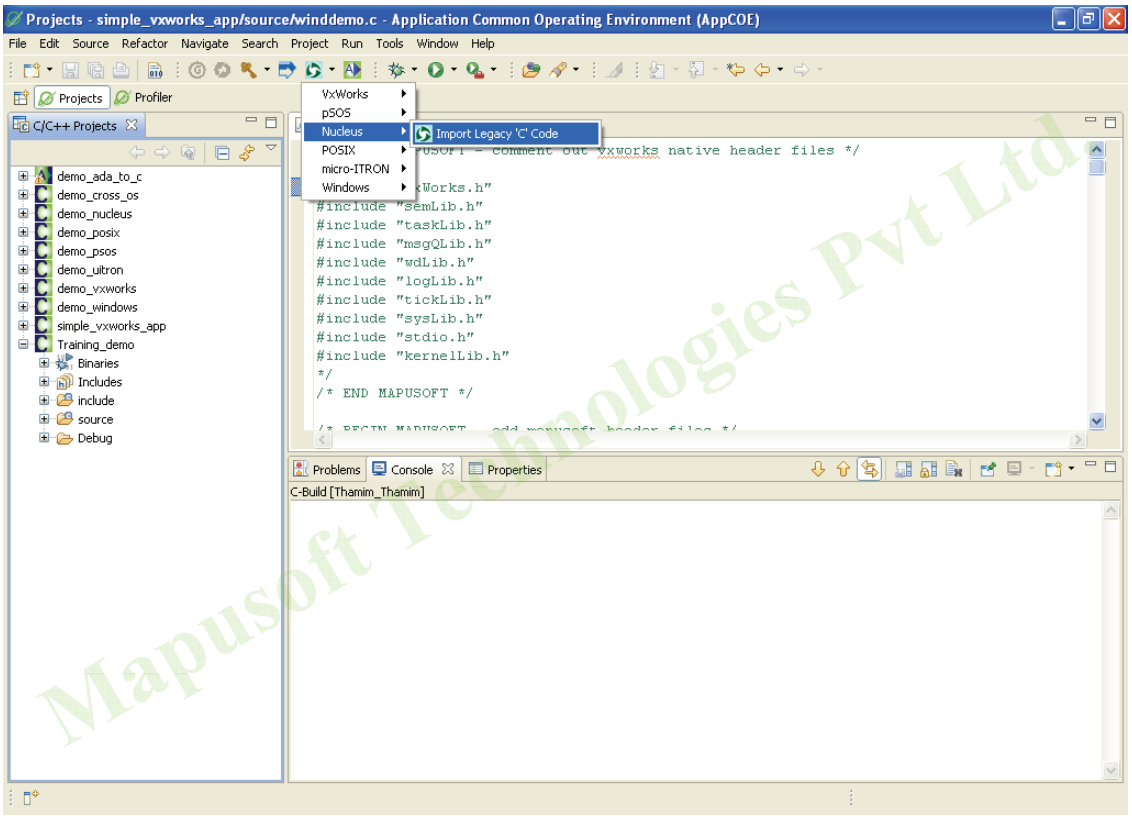
After

Please Note: Here you may also have to remove or rename your main, since AppCOE will be using OS Abstractor main. So if you are doing some stuff in main, then rename that and put it as **"Root Task Prototype" in step 4**. Otherwise if you are just calling another function from main, then just remove main and enter the function being called as **"Root Task Prototype" in step 4.**

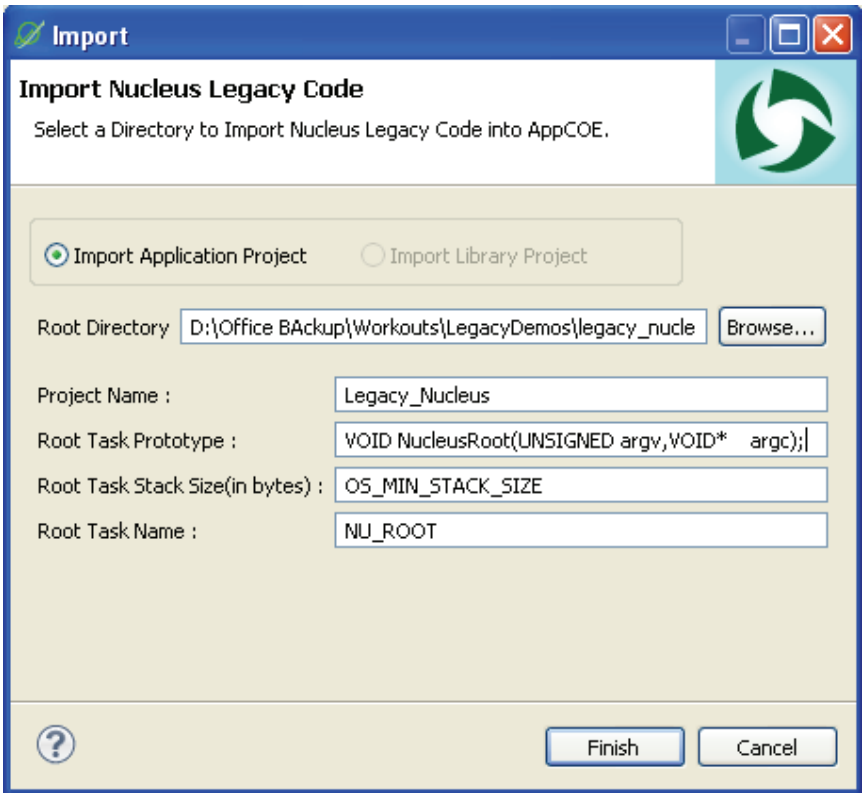
7. Now you are ready to build and run your VxWorks workbench project in the AppCOE. You can also migrate this code to the other OS, using tools described above.

4.2 Importing Legacy C/C++ Project.

1. To import a Legacy C/C++ code. Click on  button and then **Goto "<OS >" > Import workbench 'C' Project.**



2. Enter the required entries in the following window and click **Finish.** Now you are ready to build and run you Legacy code. You can also migrate this code to different OS using above decribed tools.



Visit www.mapusoft.com or submit ticket at support@mapusoft.com for further information.