AN10862

LPC1000 software development toolchain

Rev. 04 — 21 May 2010 Application note

Document information

Info	Content
Keywords	LPC1000, GNU-GCC
Abstract	How to use the GNU toolchain with the Cortex LPC1000 series

Revision history

Rev	Date	Description
04	20100501	Include Linker flag --gc-sections
03	20100125	Include “SWD” in debug config and IAR test examples
02	20091214	Include Keil test examples
01	20090827	Initial version

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

LPC1000 is the new chip series that is based on the ARM Cortex processor family with many advanced features.

This document will show how to use the toolchain in LPC1000 series such as: LPC17xx (Cortex-M3), LPC13xx (Cortex-M3) and LPC11xx (Cortex-M0) families.

In this document, we will illustrate the setup of microcontroller LPC1768 on Keil MCB1700 evaluation board, using the GNU Toolchain. All examples code are also compiled and tested in Keil and IAR development platform.

2. Software update requirement

Here is a list of setup files should be installed:

- Java Runtime Environment (required by IDE – Eclipse)
- CodeSourcery Toolchain
- C/C++ Eclipse IDE
- Zylin CDT plugin for Eclipse.
- Segger J-link GDB server and driver.
- Flash Magic.

2.1 Java Runtime Enviroment

First, we must check for Java Runtime Environment on system since Eclipse IDE is written partially in JAVA.

- Go to the command prompt, type command “Java –version” and look at the result.

Fig 1. JAVA version check

- If windows cannot recognize this command or the version is not equal to or higher than 1.6.0_01, you will need to download and install JRE in the next step.

Downloading Java Runtime Enviroment

- Go to the website: http://www.nxp.com/redirect/java.sun.com
- Look for “Download” item, follow these links and find the setup file “Java Run Time Environment” (JRE) as shown in Fig 2.

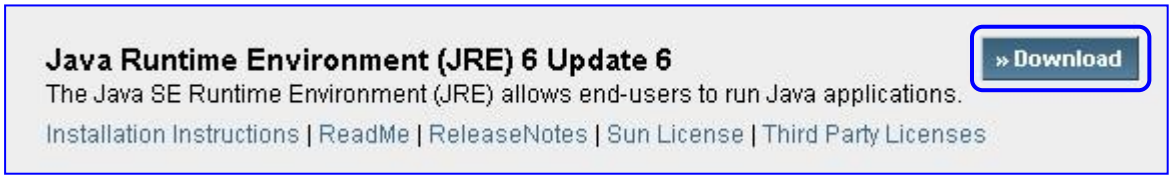


Fig 2. Download JAVA Runtime Environment

The version found on the website may be latest, and hence different from the example given in this application note.

2.2 CodeSourcery toolchain

Go to http://www.nxp.com/redirect/codesourcery.com/release830 and download the latest package release for Windows platform, and then install this package in your system after removing the previous version of CodeSourcery Lite.

Please note that the GNU version in “makeconfig” file must be changed to correspond to this release version.

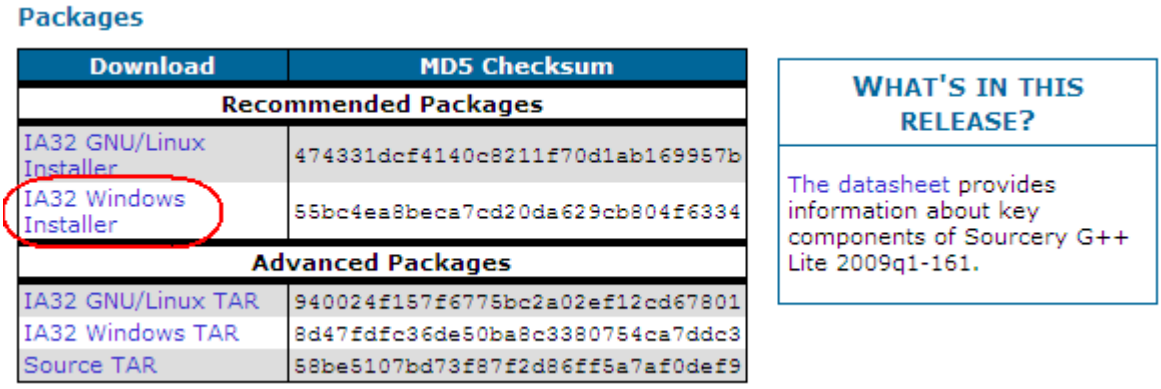


Fig 3. Update CodeSourcery toolchain

2.3 Eclipse IDE for C/C++

Go to this website to download the Integrated Development Environment – IDE of Eclipse:

http://www.nxp.com/redirect/eclipse.org/downloads

Fig 4. Choose “Eclipse IDE for C/C++ Developers” item to download

Install Eclipse IDE after downloading and set all options to default during installation.

2.4 Zylin CDT plug-in

Go through Help → Software update…

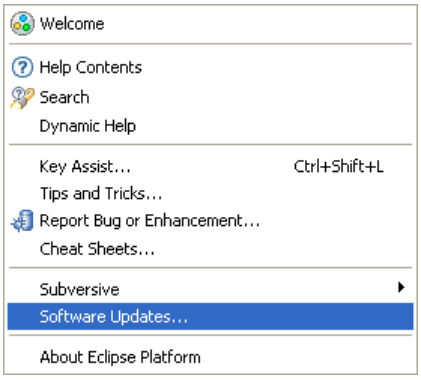


Fig 5. Choose Software Update in Eclipse

If the site http://download.eclipse.org/tools/cdt/releases/ganymede exists in Available Software, choose all items inside there, and then click Install… Follow remaining steps to complete the update.

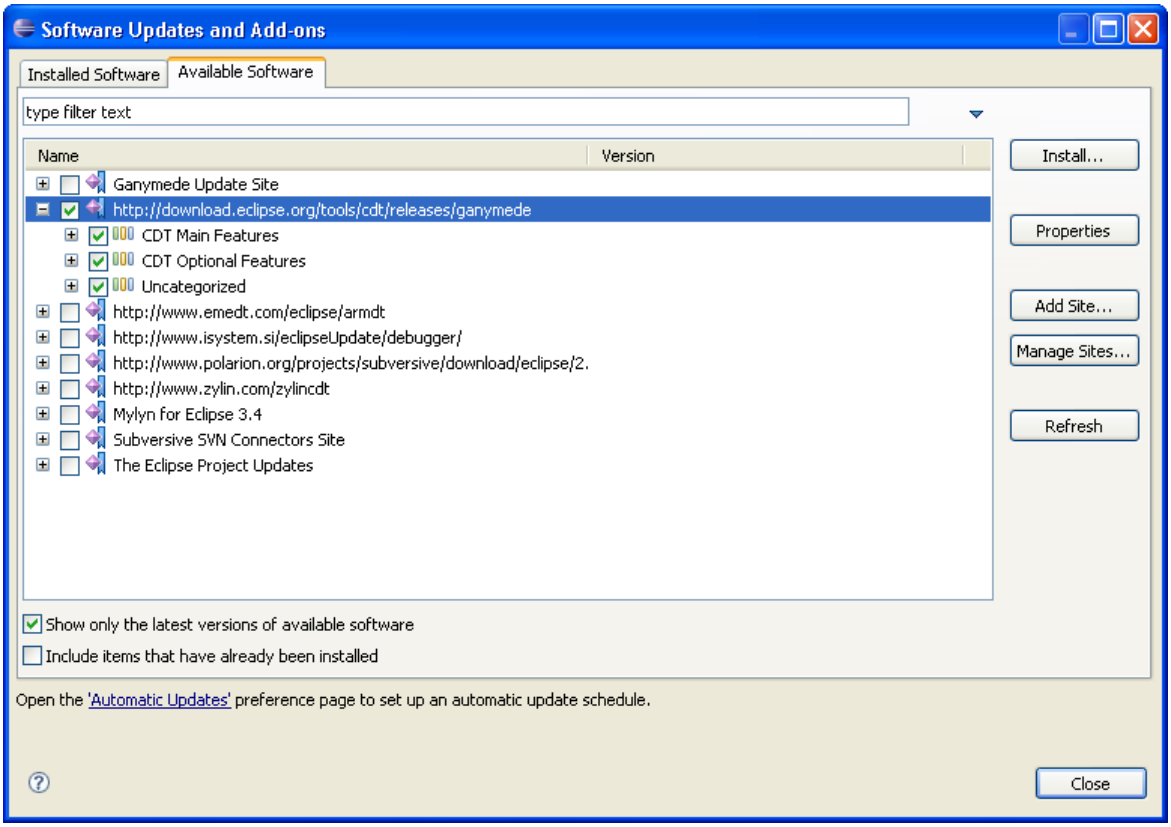


Fig 6. Check all items inside http://download.eclipse.org/tools/cdt/releases/ganymede sub-item

If this site address does not exist, click Add Site and fill this address in to dialog http://www.nxp.com/redirect/download.eclipse.org/ganymede

For the Zylin Embedded CDT plug-in, because its repository has been removed and changed to http://opensource.zylin.com/zylincdt, we need to re-target this link in Eclipse software update.

Click the Add Site button as shown in Fig 6, fill in the location field with the new site address, and follow the remaining steps to complete the update.

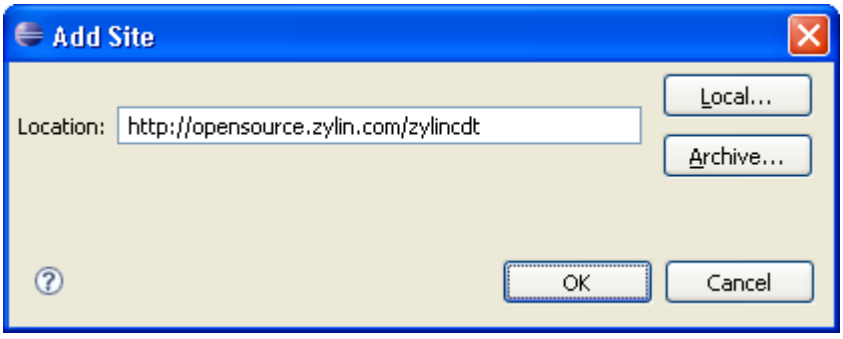


Fig 7. Update Zylincdt with new repository

2.5 Segger J-link

Go through http://www.nxp.com/redirect/segger.com/download_jlink.html to download the latest J-Link software.

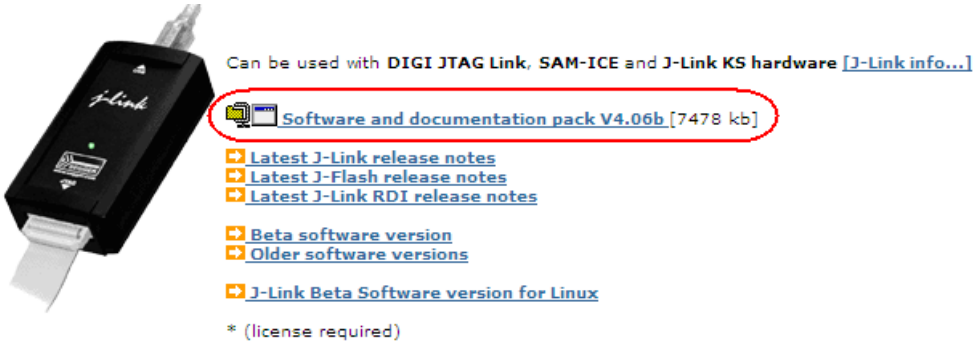


Fig 8. Update J-link software from SEGGER

2.6 Flash Magic

Flash Magic has supported flash downloading for LPC17xx series.

http://www.nxp.com/redirect/flashmagictool.com/

3. Running Eclipse for the first time

3.1 Eclipse workspace setup

- Initialize Eclipse on the desktop:

Fig 9. Initialize eclipse icon on the desktop

- Set workplace for the first time here. In this case, a new workspace named “nxpdrv” (also the name of the new folder) will be created in the root of the C: drive; all the projects will be located in “C:\nxpdrv\”.

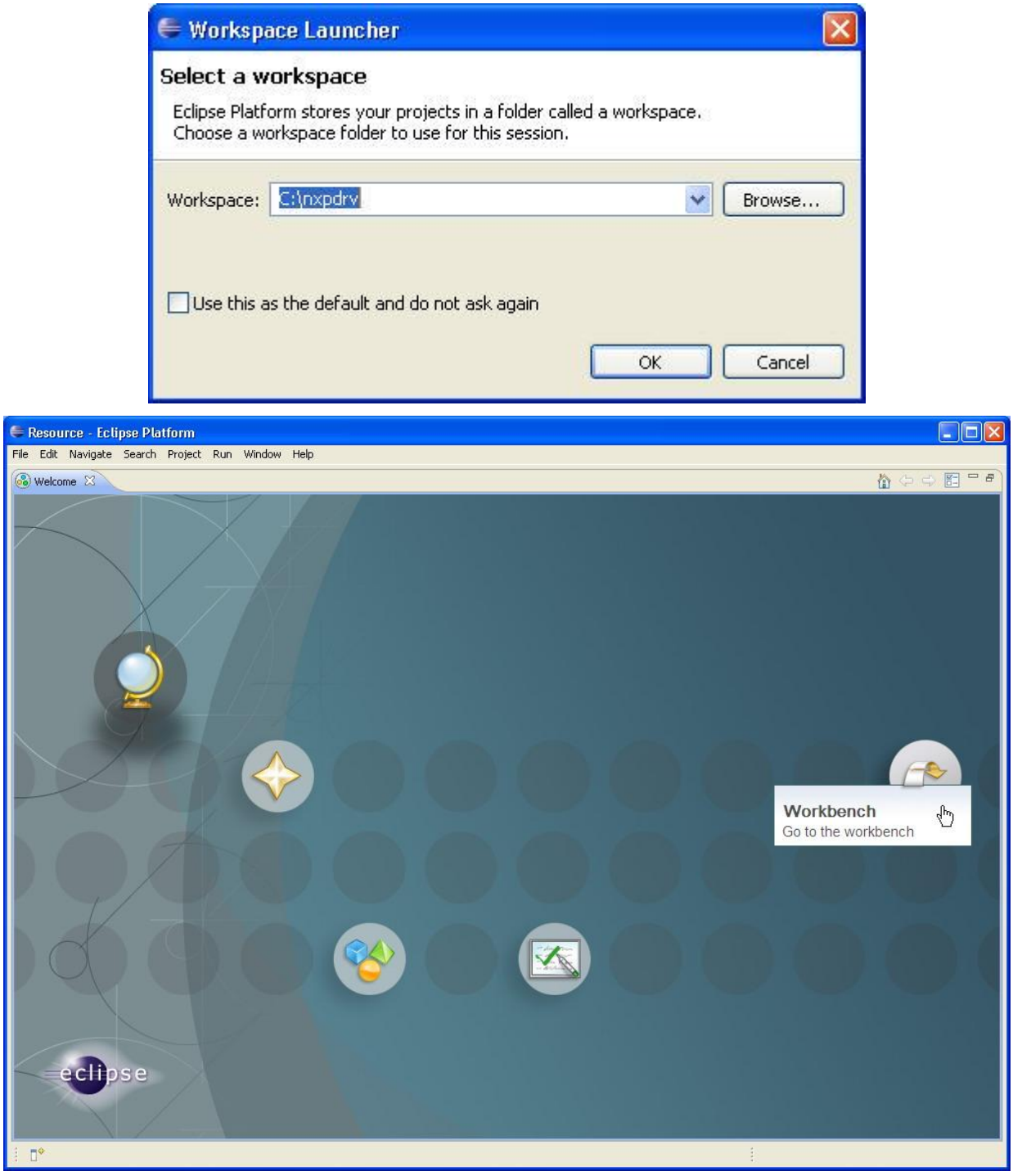


Fig 10. Set workplace and Eclipse for the first time

- Open perspective: Window → Open Perspective → Other ...

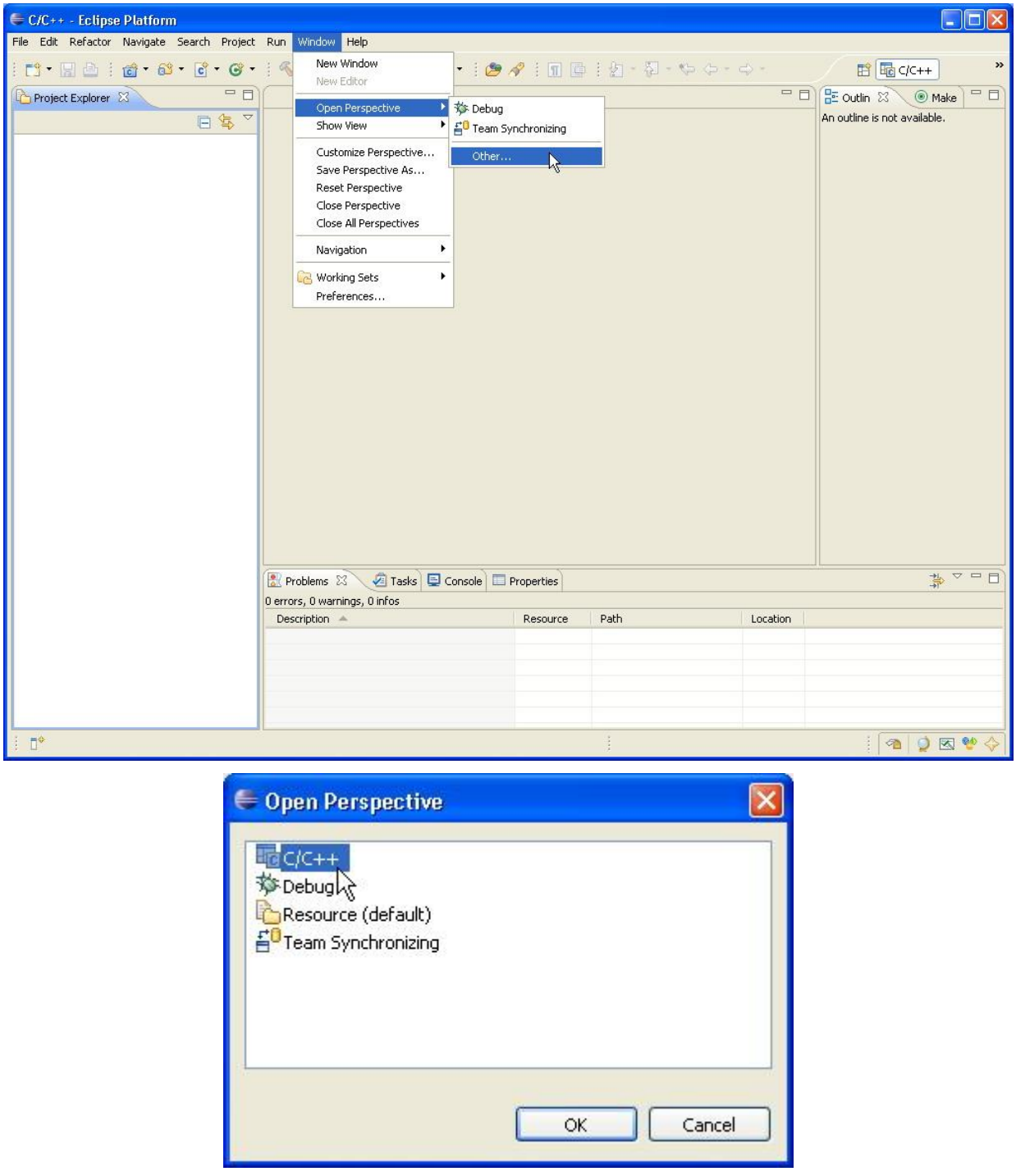


Fig 11. Open perspective for C/C++

3.2 Setting up External tools for Eclipse

The Workbench provides the mechanism for running tools that are not part of it. The examples shows how the following tools are added:

- Flash magic.

- GDB Server
- Serial terminal (existing on Windows)

Flash magic tool

Run → External Tools → Open External Tools Dialog…

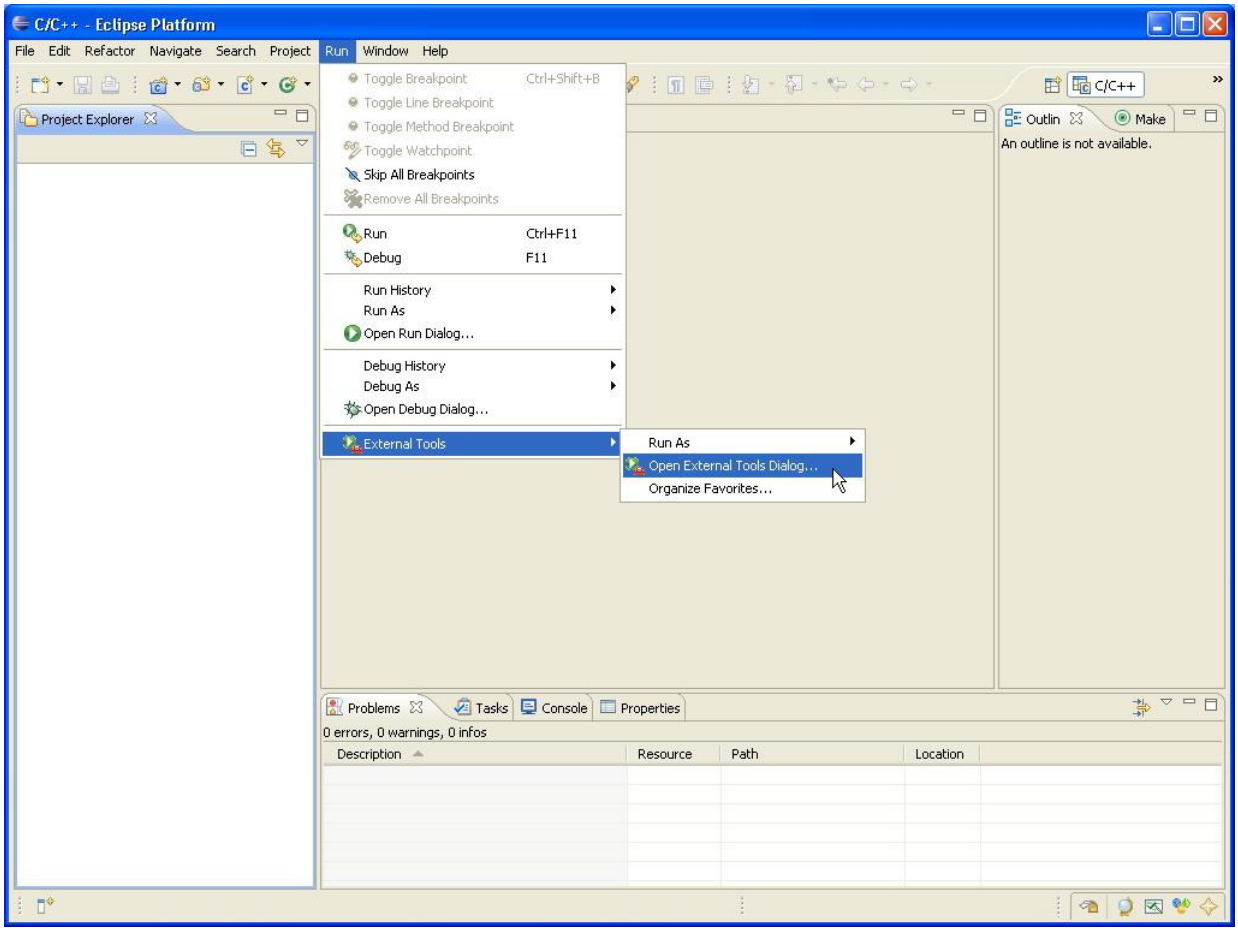


Fig 12. Steps to setting Flash magic as external tool (Step 1)

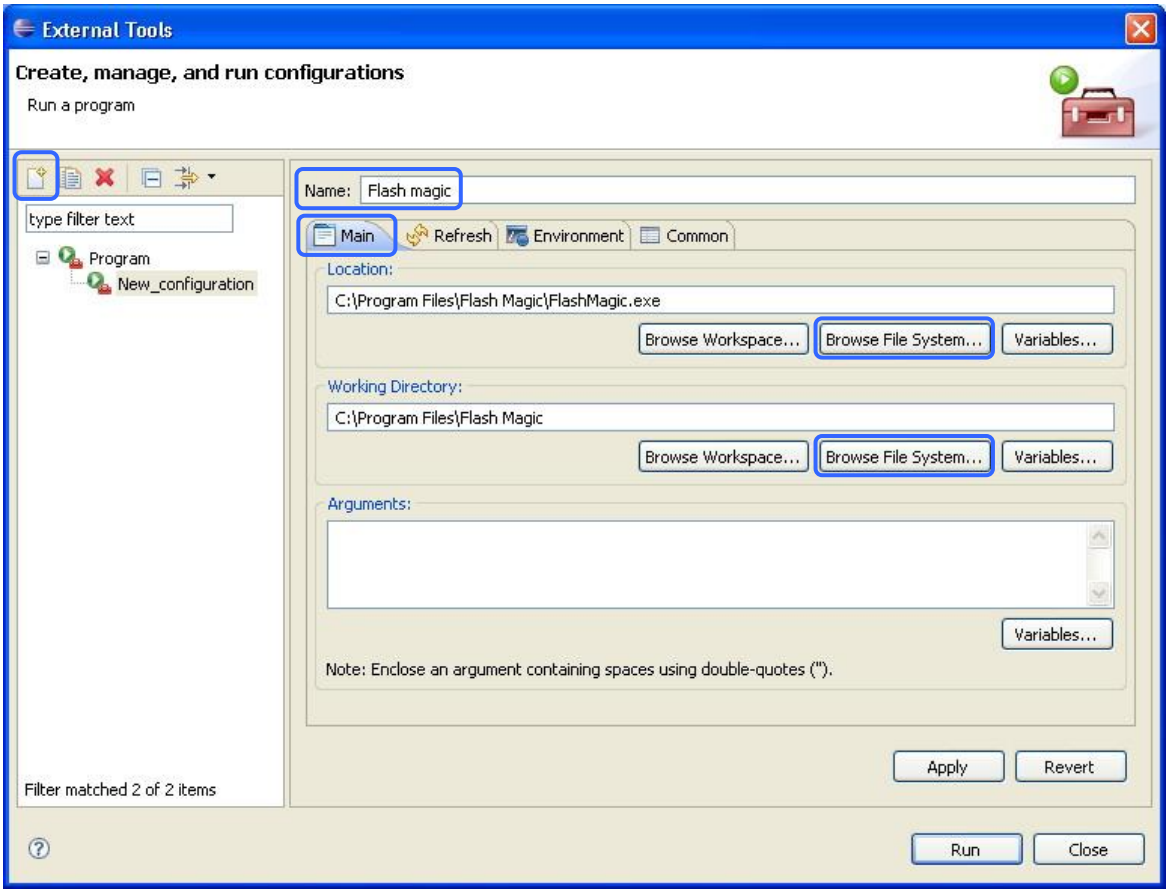


Fig 13. Steps to setting Flash magic as external tool (Step 2)

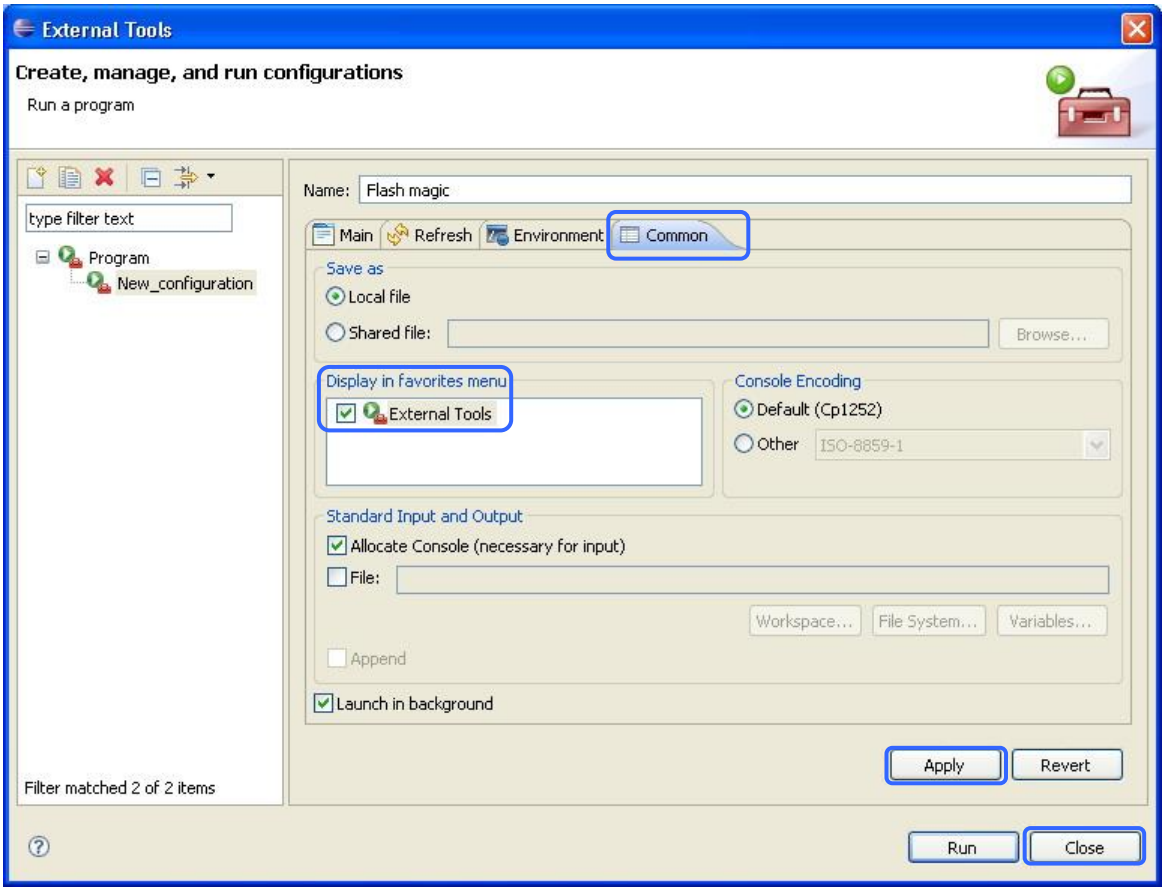


Fig 14. Steps to setting Flash magic as external tool (Step 3)

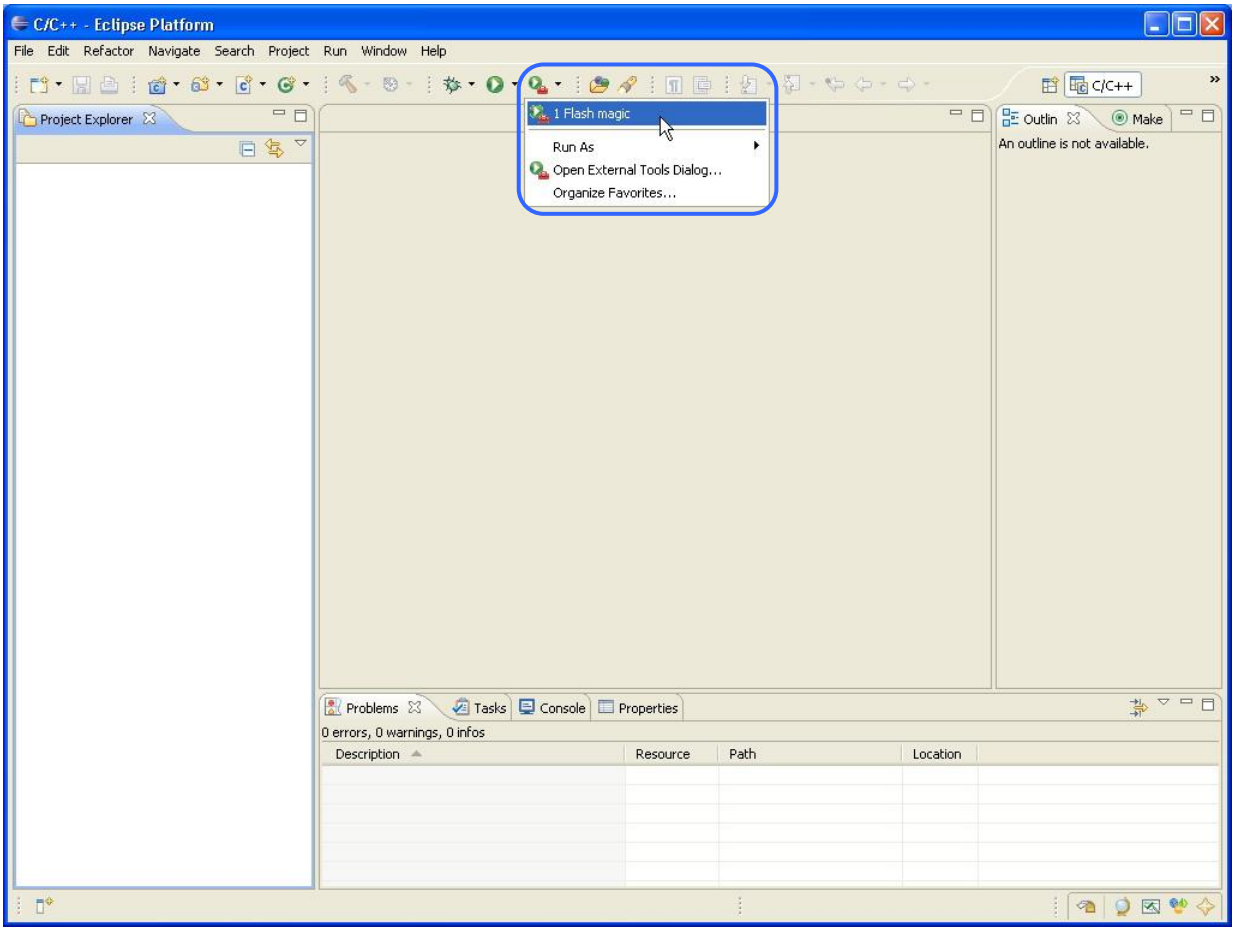


Fig 15. The result after setting

Follow the above steps to add other tools

GDB server

Located at directory that Jlink was installed, such as: “C:\Program Files\SEGGER\JlinkARM_Vxxx\JlinkGDBServer.exe”

Where Vxxx is the Jlink version that using

Ex: JlinkARM_V408, JLinkARM_V409j

Note: If you want to use SWD debug mode (for LPC13xx and LPC11xx devices only support SWD), include “-if SWD” in Argument tab in External Tools Configurations of gdb program as shown in Fig 16.

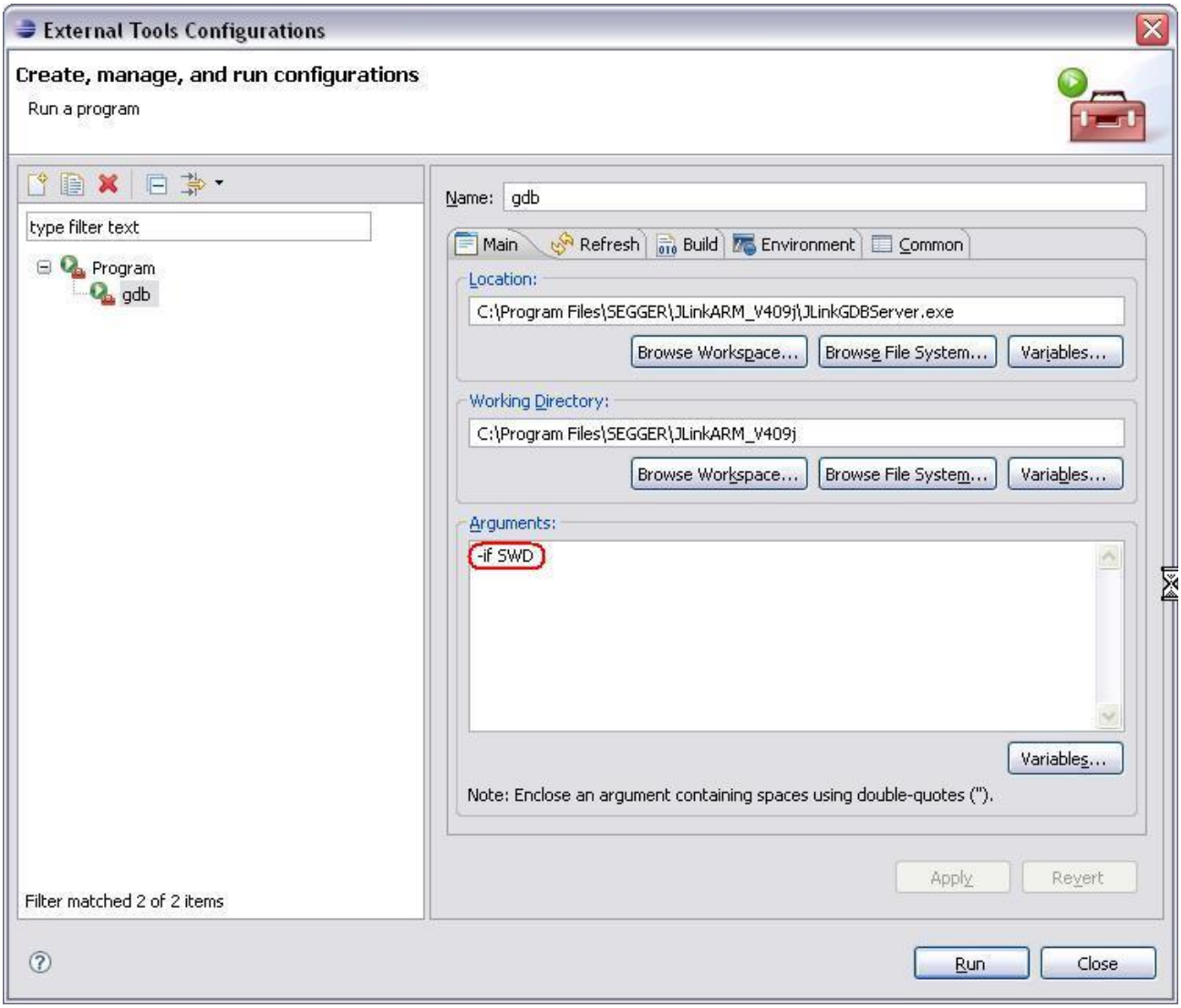


Fig 16. Configuration for cater SWD option

Serial terminal

Located at C:\Program Files\Windows NT".

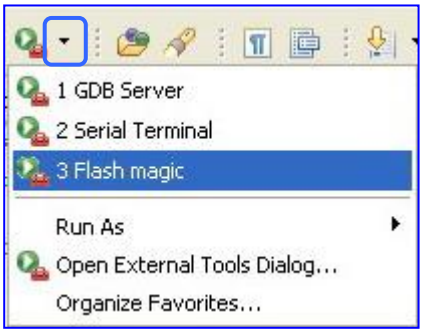


Fig 17. The result after all external tools setting

4. Creating and working with LPC1000CMSIS project

These following steps demonstrate how to build examples in project with CodeSourcery Lite/GNU toolchain in Eclipse IDE:

The project we used to represent here is LPC1700CMSIS. Other LPC1000CMSIS project are correlative.

4.1 Create new workspace

In this case, a "nxpdrv" workspace is created on C:\ drive.

4.2 Create new project

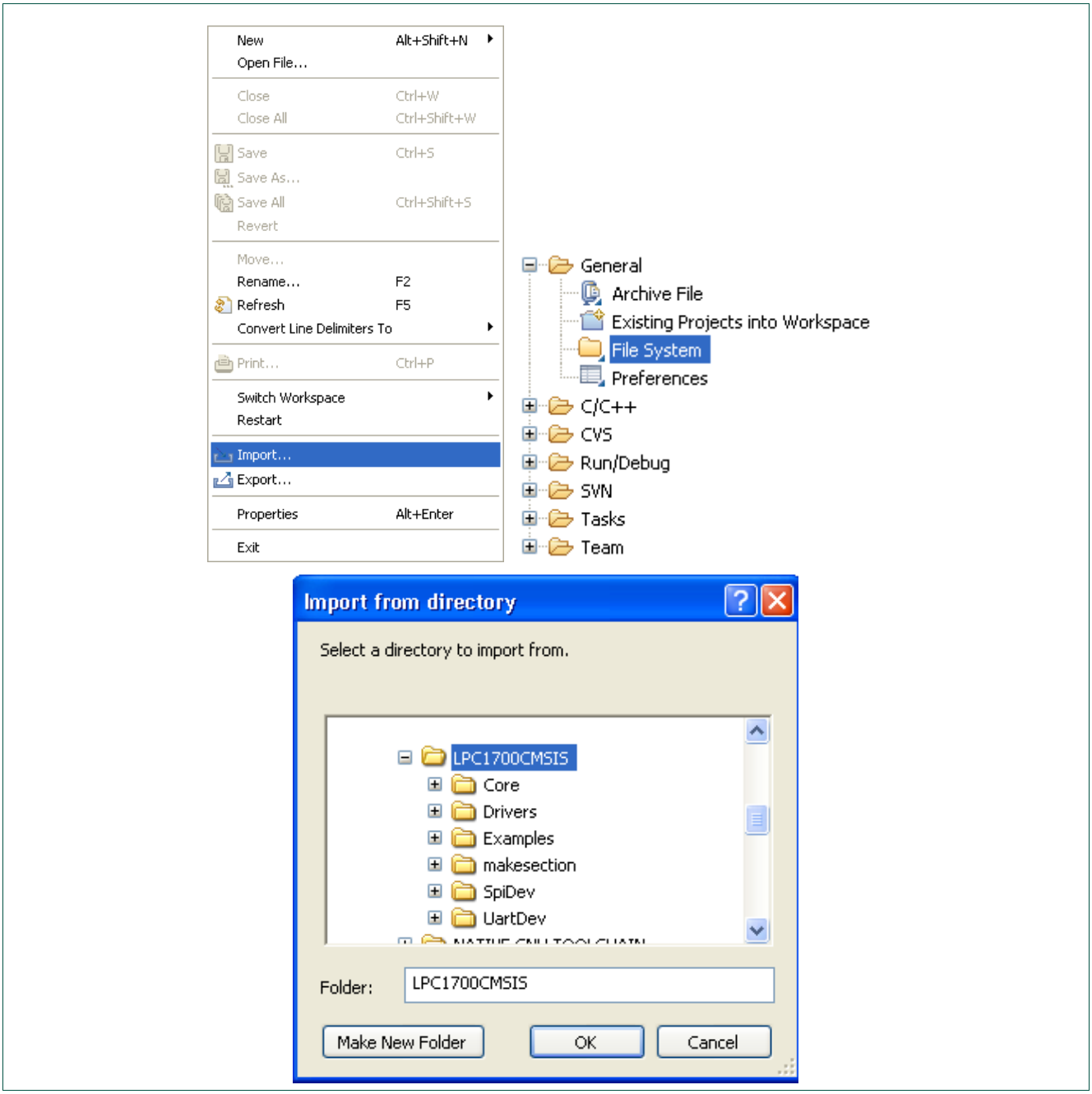
In this case, a new project is created with the name "LPC1700CMSIS".

4.3 Extract the package

In this case, the "zip" package is extracted on D:\ driver, a folder named "LPC1700CMSIS" will be generated on D:\ drive.

4.4 Import package resource file into project

Import all the source files in project package (File → Import… → Choose File System → Browse to D:\LPC1700CMSIS)



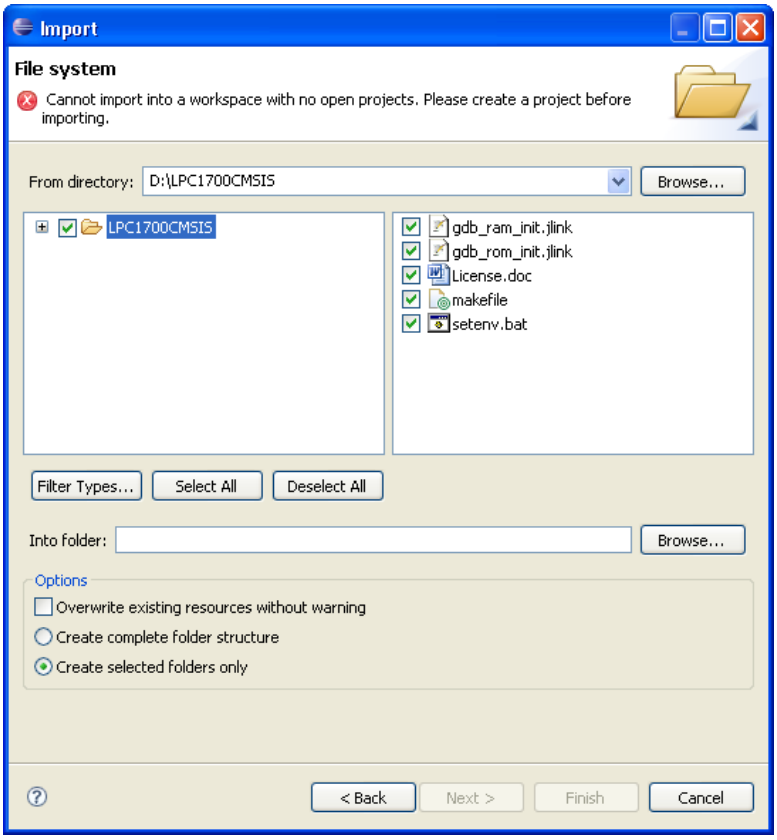


Fig 18. Import File system in resource package

The result should be like this on the left window (project window):

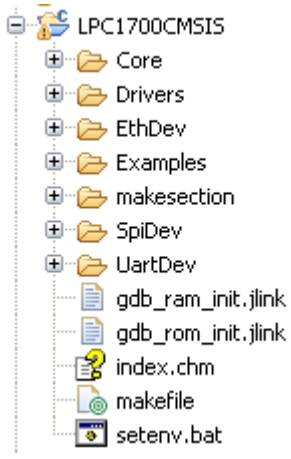


Fig 19. Result at the left window (Project window) after importing

Configure “make” environment: Open “makeconfig” file and edit the “PROJ_ROOT” symbol as shown in Fig 20:

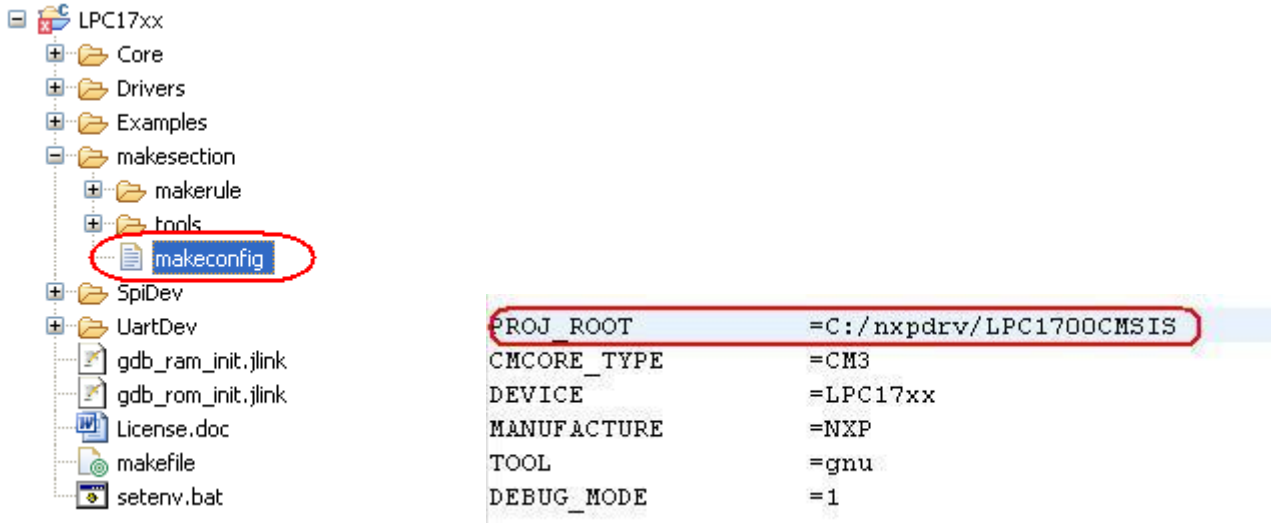


Fig 20. Configure "make" environment in "makeconfig" file

(In this case, "nxpdrv" means the eclipse workspace that was created first, and "LPC1700CMSIS" is the name of this project; note the forward slash style is used).

Close the Eclipse IDE.

4.5 Configure environment of "make utility"

In this case, all "make tool" resides in C:\nxpdrv\LPC1700CMSIS\makesection\tools

In order to make the system recognize the command located in private tool that its path is not included in the existing PATH variable of the system and user environment, the user must set the path of private tool to the "path" variable of the system variable environment.

In each project after importing the file system, all of the required tools are in the in "tools" folder; the user must set the path of the "tools" at the head of the "path" in system variables.

Note: Make sure all old paths of previous similar utility tools are removed in both user and system variables.

- On desktop, go to Start → Control Panel → System…

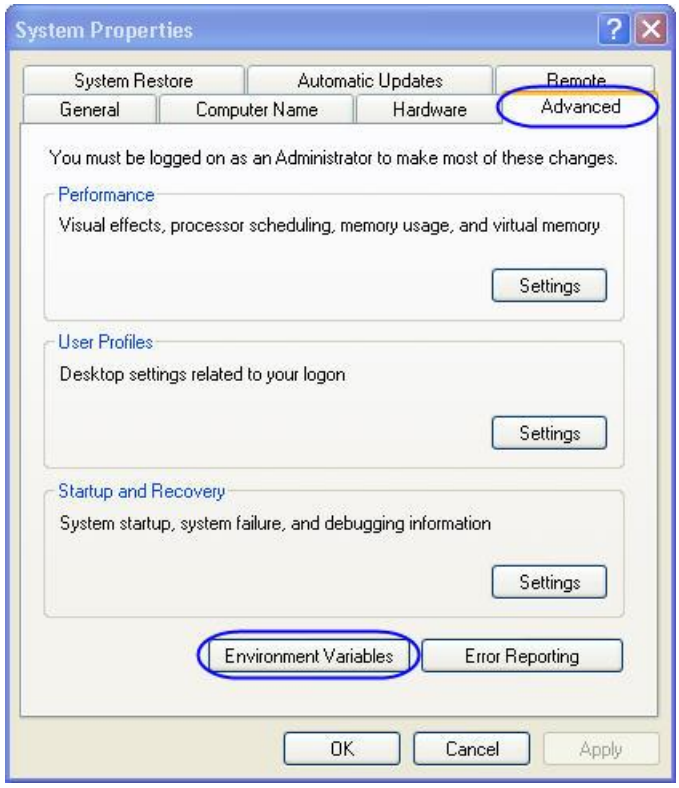


Fig 21. Environment configuration (Step 1)

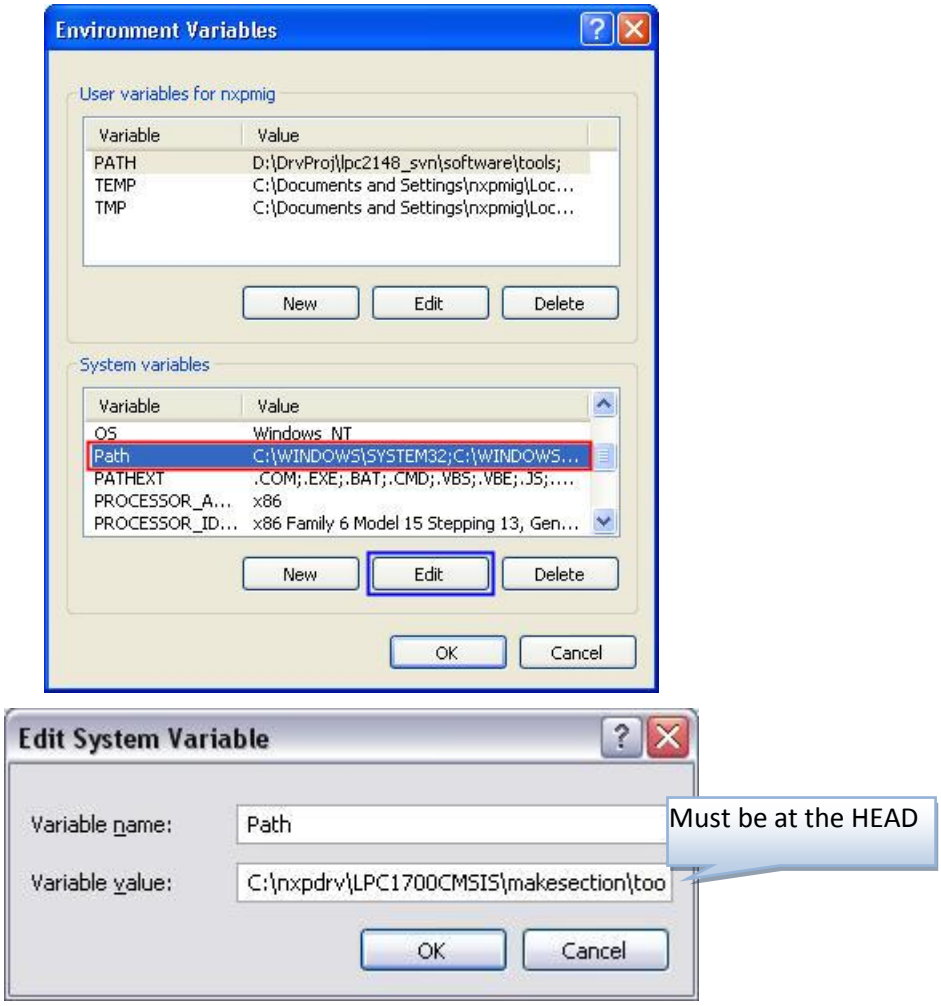


Fig 22. Environment configuration (Step 2)

4.6 Compile project

Re-open Eclipse, and then go through Project →Property… to configure the properties for the “LPC1700CMSIS” project as shown in Fig 23:

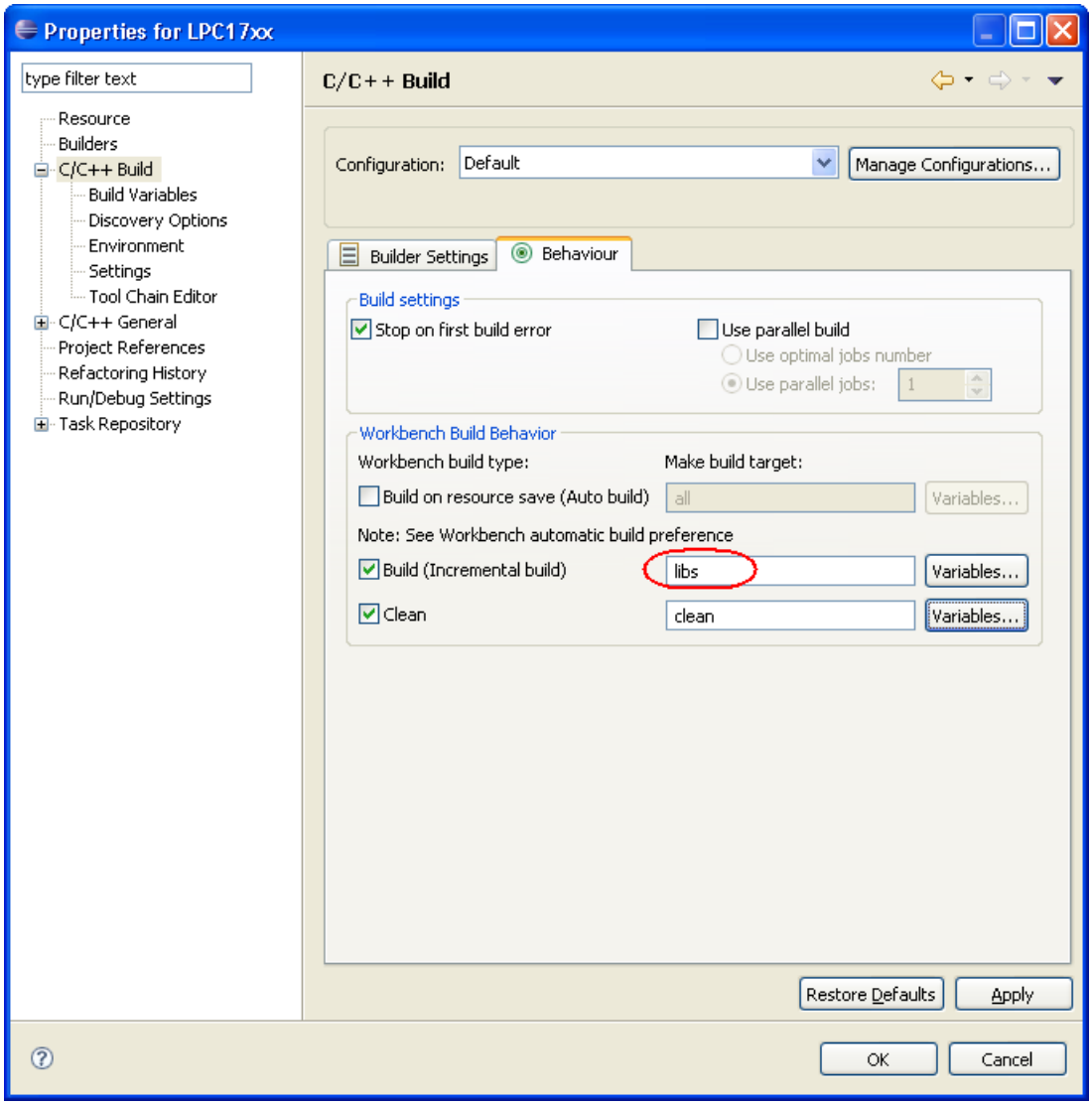


Fig 23. Configure “libs” for compiling entire project mode

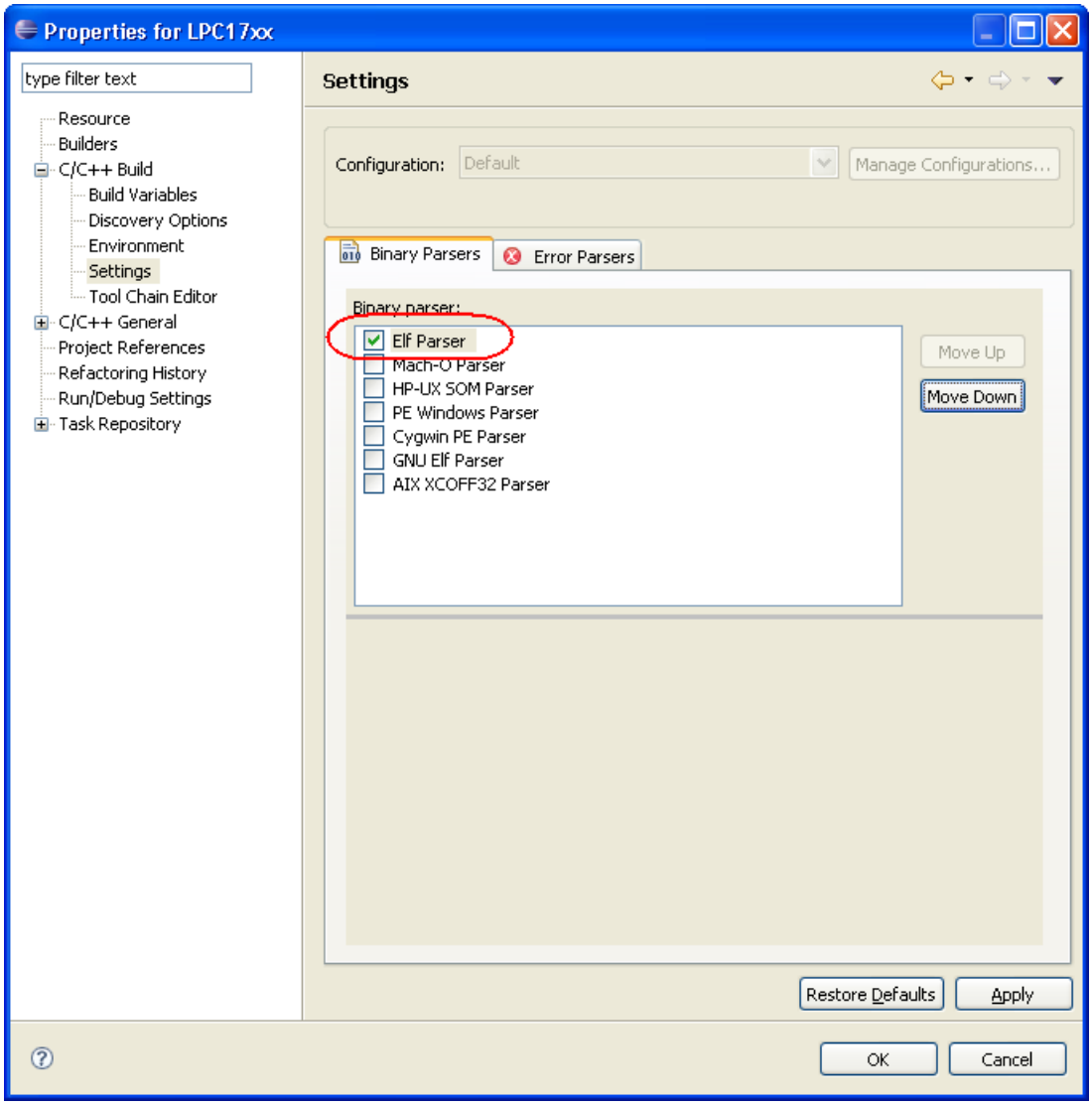


Fig 24. Check this option to allow .elf file to be recognized

Now, build entire the project…

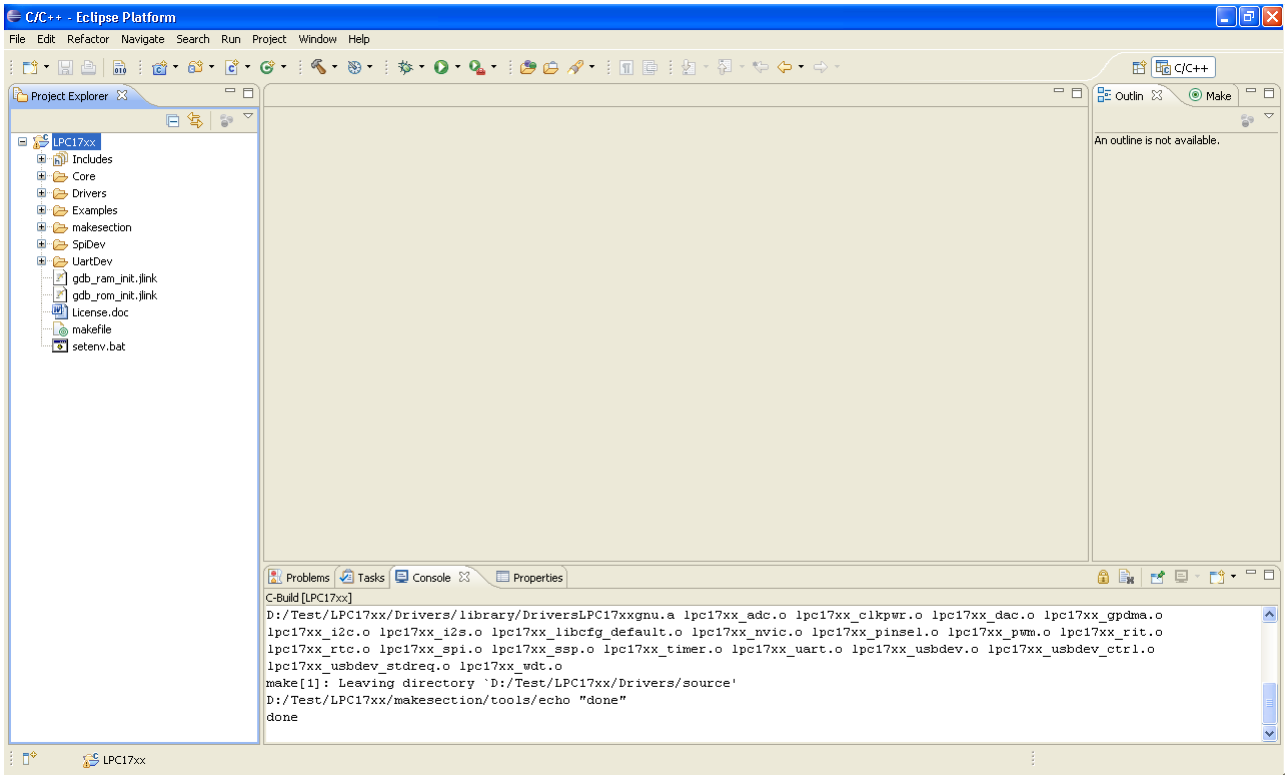


Fig 25. Result in Console window after building project

To build the example, right click on the root of that example → Make target → Build…

(Note: In this case, the example is placed at C:\nxpdrv\LPC1700CMSIS\Examples\, for each peripheral. There are some sub-folders that contain specific examples for various purposes to test this peripheral)

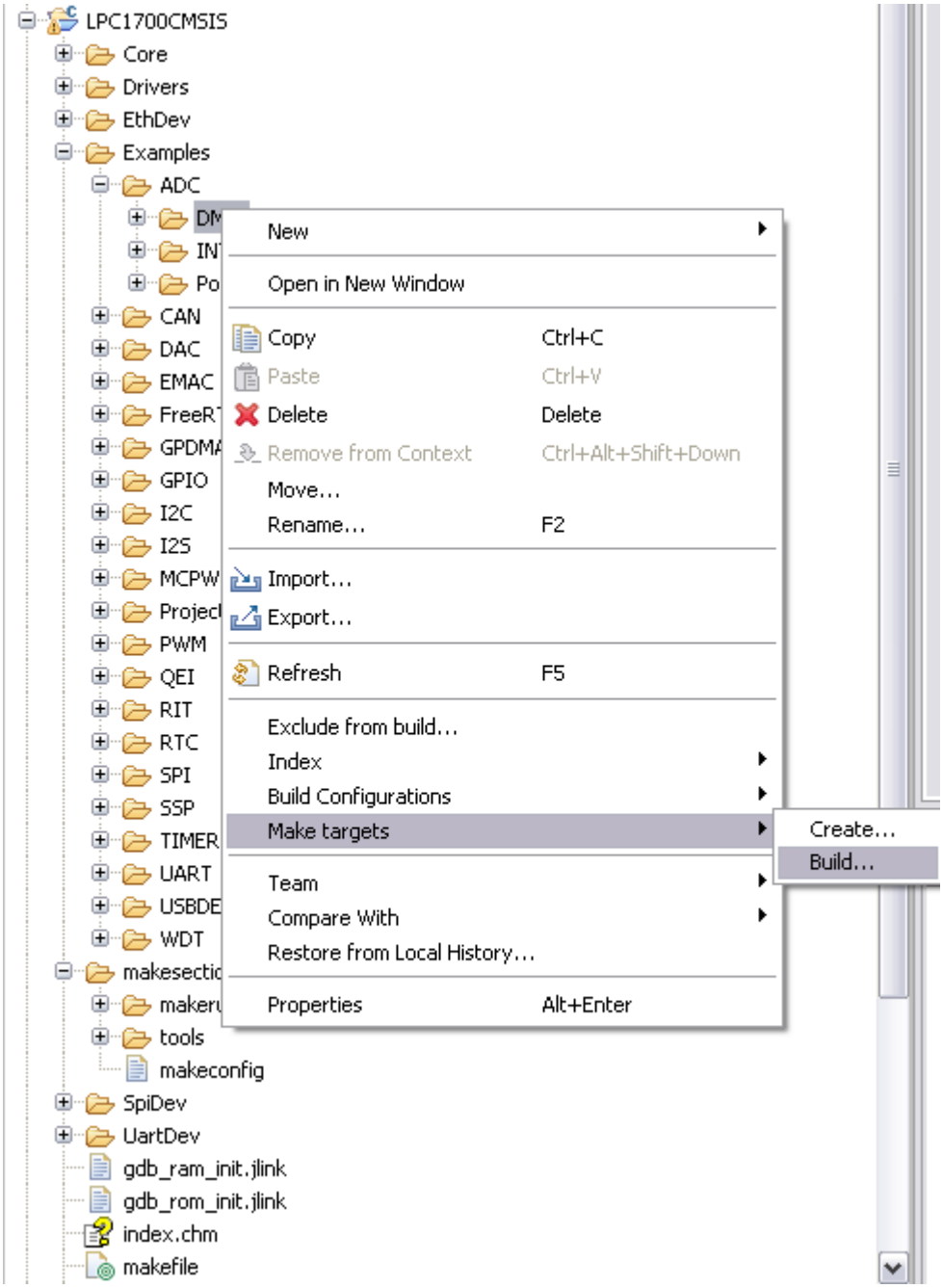


Fig 26. Create “make target” for each example (1)

Create these “make target”: “ram” and “rom”…

- RAM mode, target name is 'ram'
- ROM mode, target name is 'rom'

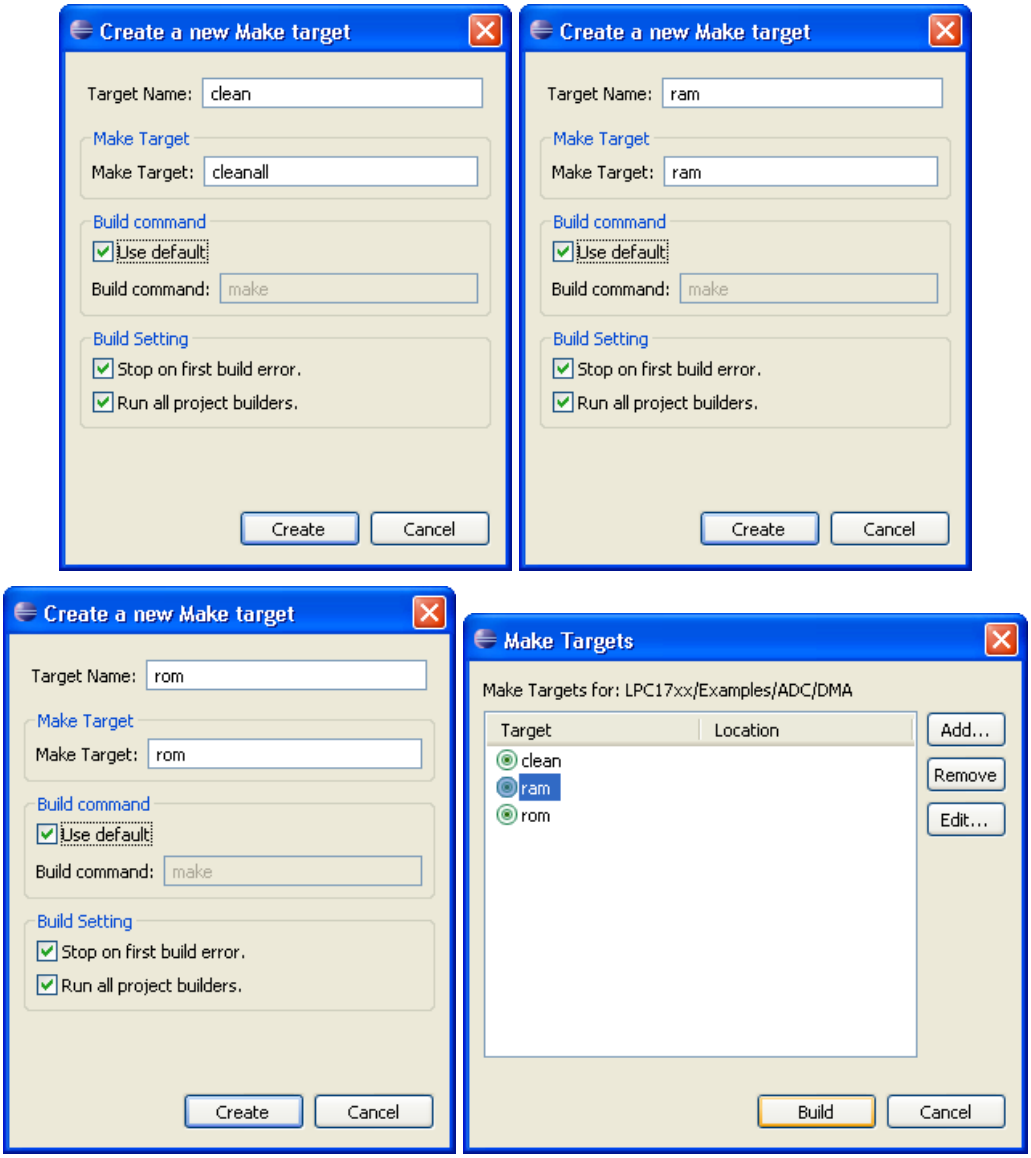


Fig 27. Create “make target” for each example (2)

Now, build the example in ram mode…

The result should like the example shown in Fig 28 after building successfully at the left window (Project window).

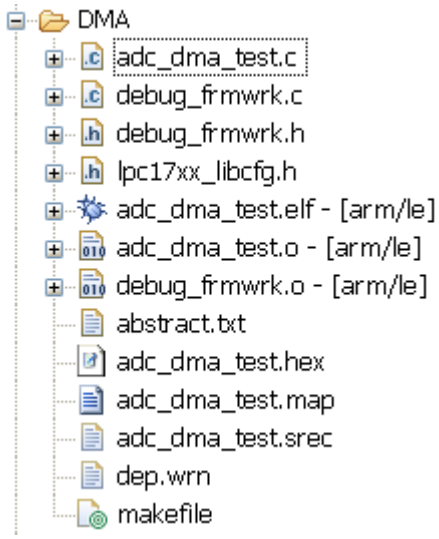


Fig 28. Result after building the “DMA” example

Tip

Open “Make Targets” viewer to manage projects easier.

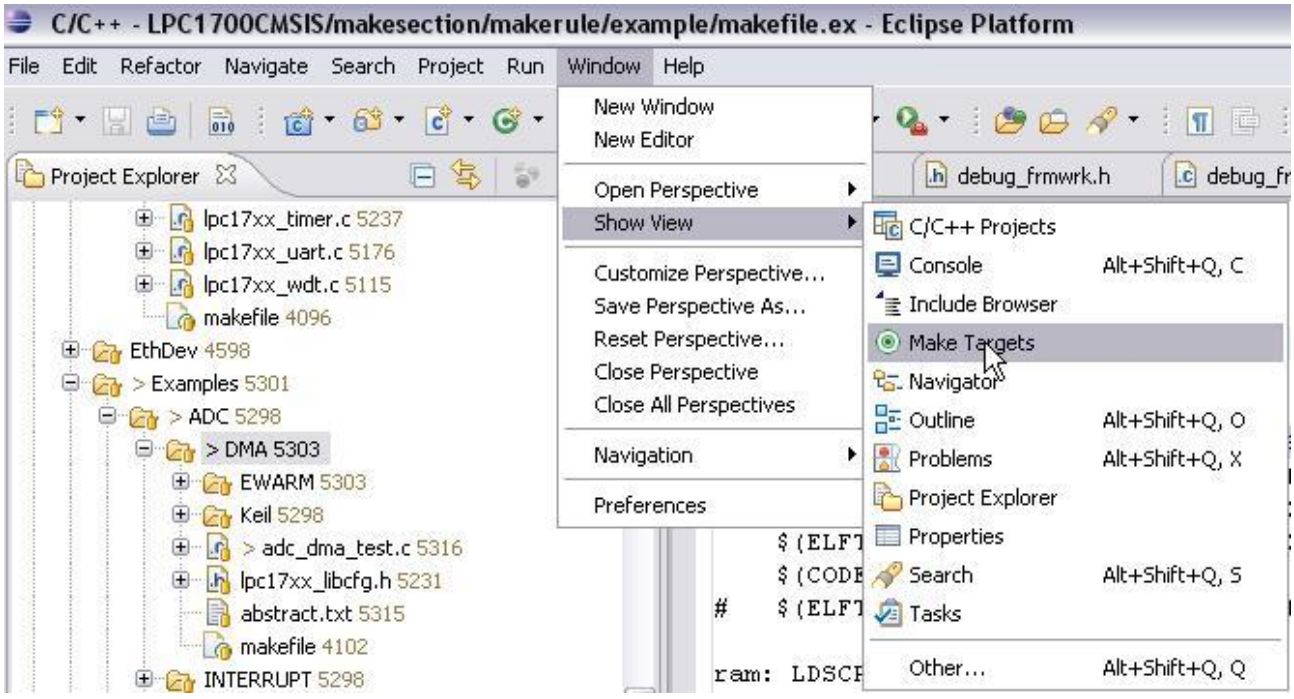


Fig 29. Open “Make Targets” view

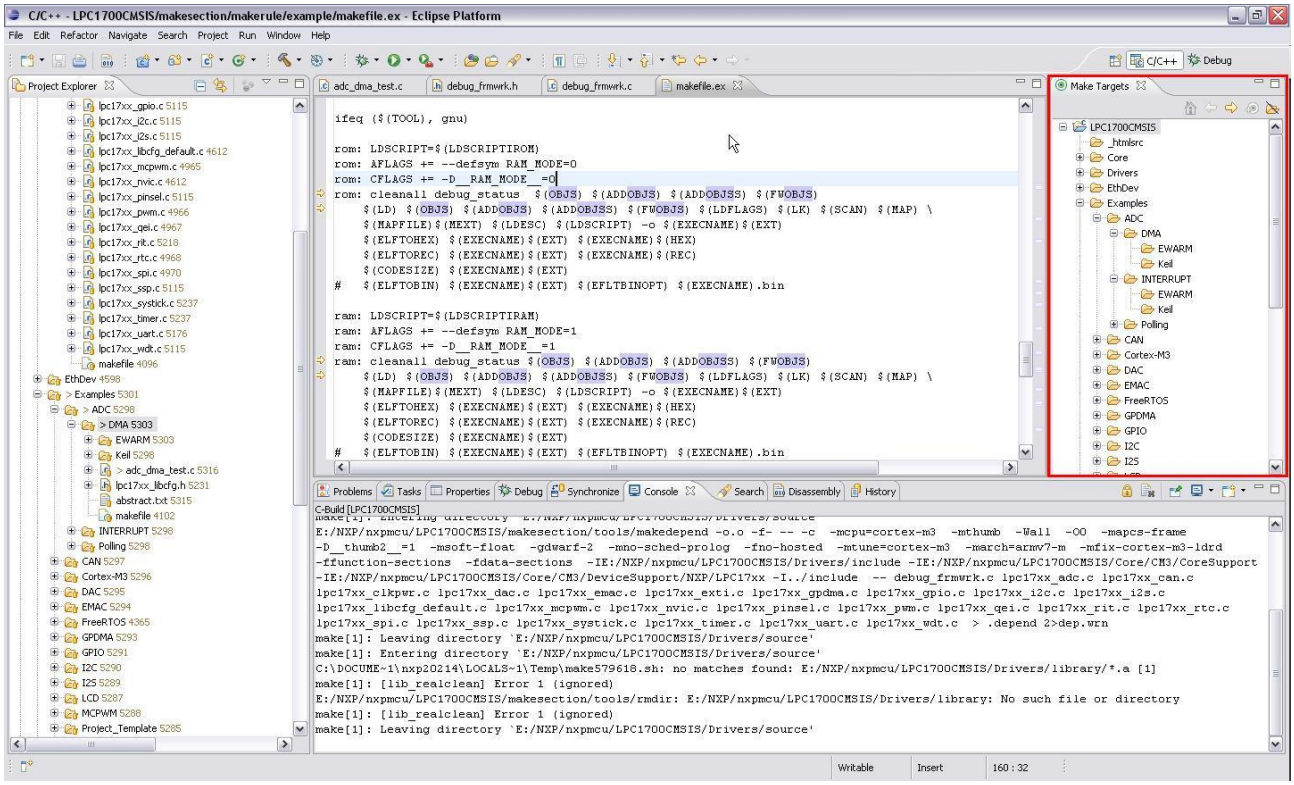


Fig 30. “Make Targets” view

- Right click in example on “Make Targets” view to create target
- Double click in example target to build/clean project.

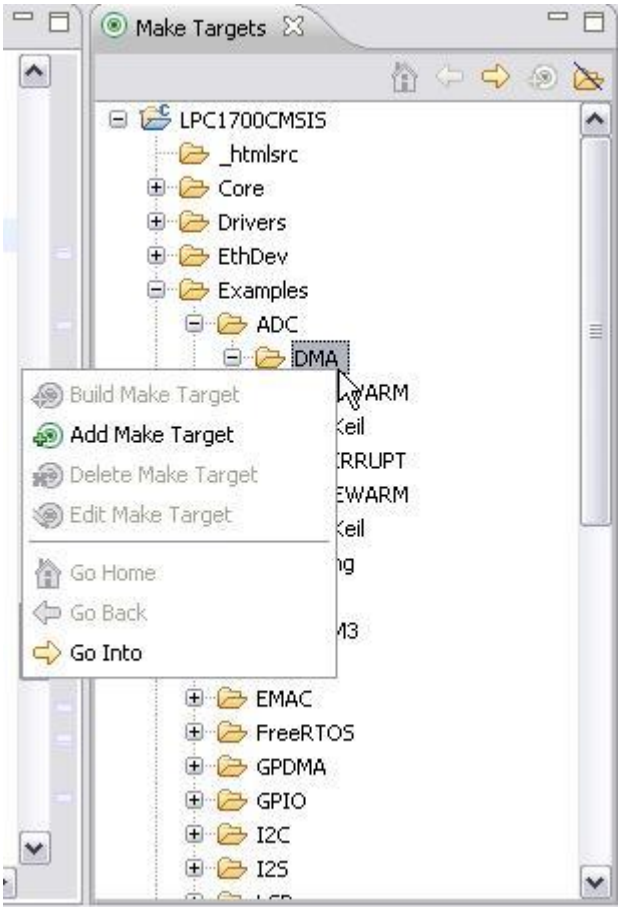


Fig 31. Create make target

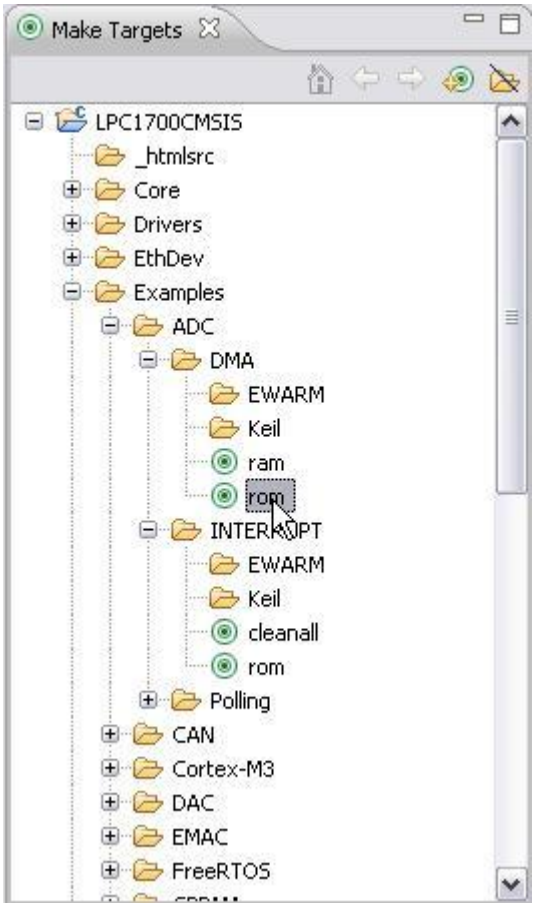


Fig 32. Run target

4.7 Remove unused sections

"—gc-section" is the flag supported by GNU to remove unused sections. It could reduce code size.

Uncomment this flag in 'make.LPC17xx.gnu' placed at '..\makesection\LPC17xx' folder.

```
LK                =  -static -mcpu=cortex-m3 -mthumb -mthumb-interwork
#LK               = -static
#LK               += -Wl,--start-group $(TARGET_FWLIB_LIB)
LK                += -Wl,--start-group
LK                      += -L$(THUMB2GNULIB) -L$(THUMB2GNULIB2)
LK                += -lc -lg -lstdc++ -lsupc++
LK                      += -lgcc -lm
LK                += -Wl,--end-group
LK                += -Wl,--gc-sections -Wl,--print-gc-sections
```

Fig 33. Uncomment '—gc-secions' flag to reduce code

Warning: Using this flag can affect debugger and cause it run incorrectly.

Recommend to use it just in “run standalone” mode.

Keil and IAR remove unused sections automatically. So don’t care about it if run examples in Keil and IAR

4.8 Execute example

4.8.1 ROM mode

Use Flash Magic to burn an image file (.hex) file (that can be observed only after building in ROM mode).

4.8.2 RAM mode

Example can be run in RAM mode only with debugger.

Debugging in RAM mode requires fewer steps than ROM mode because the image file after building in RAM mode is loaded directly into the RAM section without any external tools.

4.9 Debug the project

Basically, each example can be built in RAM mode or ROM (FLASH) mode.

4.9.1 RAM mode

Note: “make ram” must be completed before debugging in RAM mode

In this case, the “uart_interrupt” example is used to debug in RAM mode.

- Setting up Debug session: Go to menu Run → Open Debug Dialog…

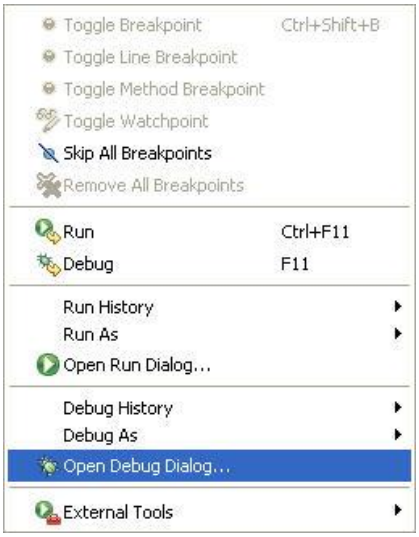


Fig 34. Open Debug dialog

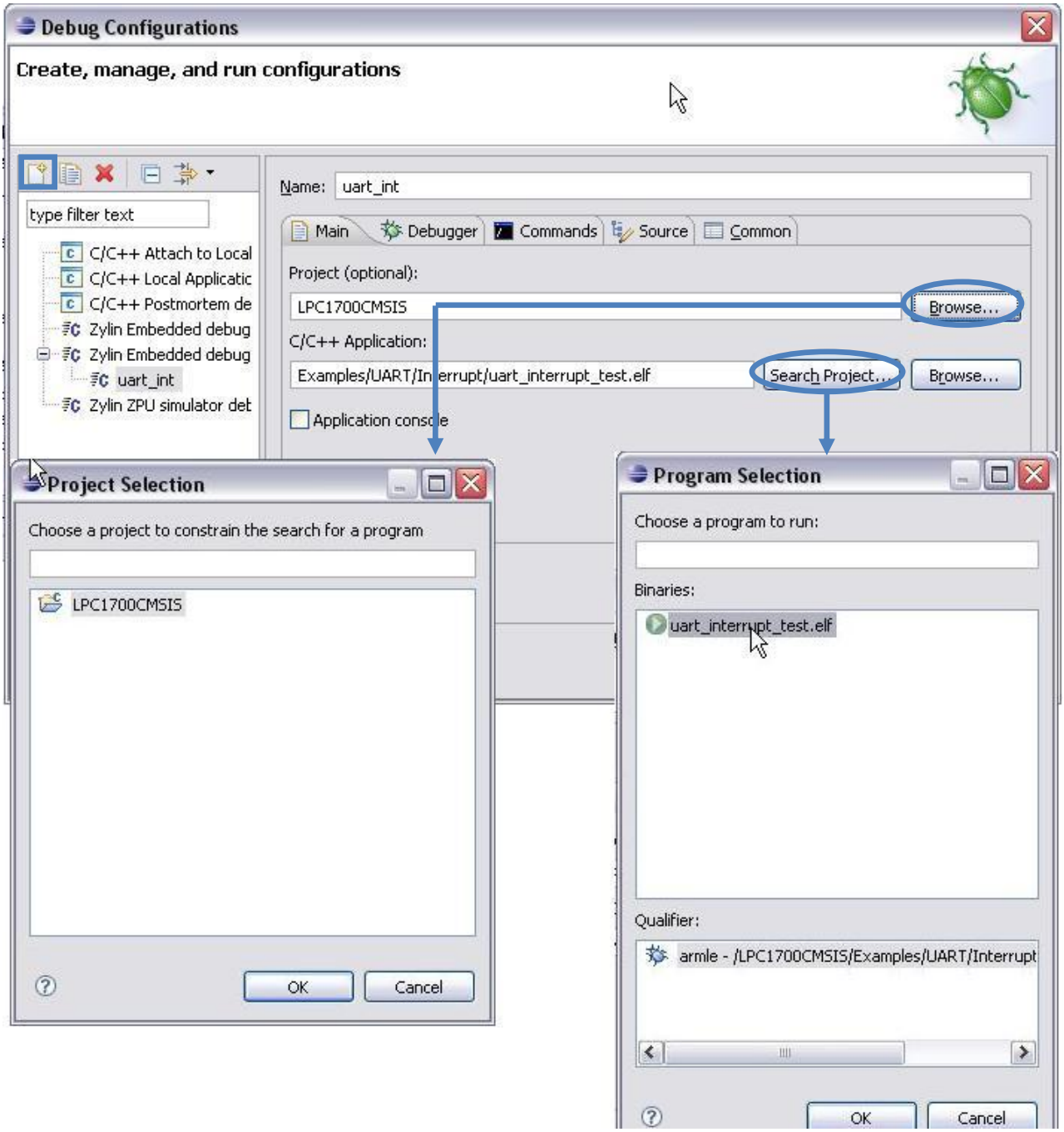


Fig 35. Create debug session (Step 1)

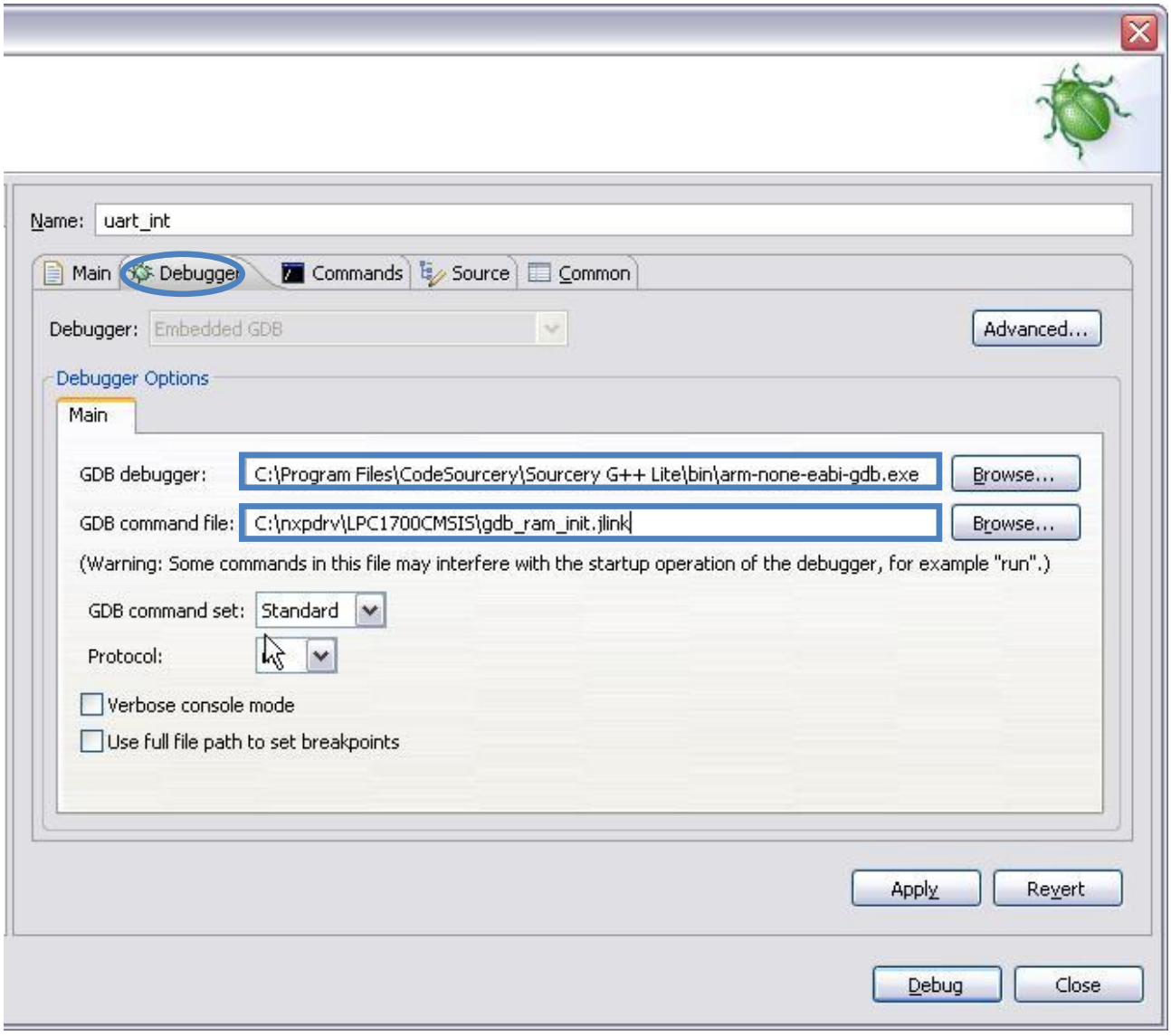


Fig 36. Create debug session (Step 2)

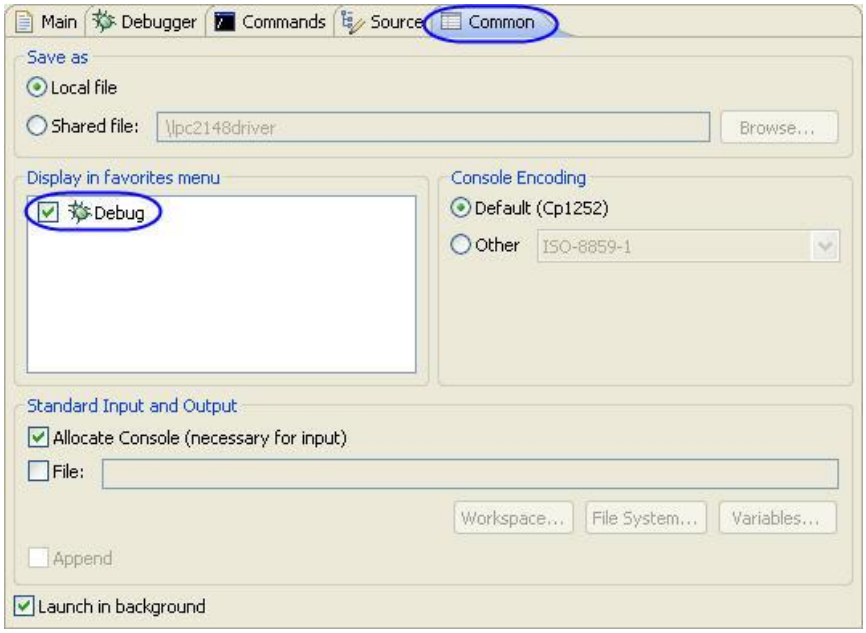


Fig 37. Create debug session (Step 3)

- Debugging in RAM mode (Note: Open “uart_interrupt_test.c” file to set breakpoint in next step).

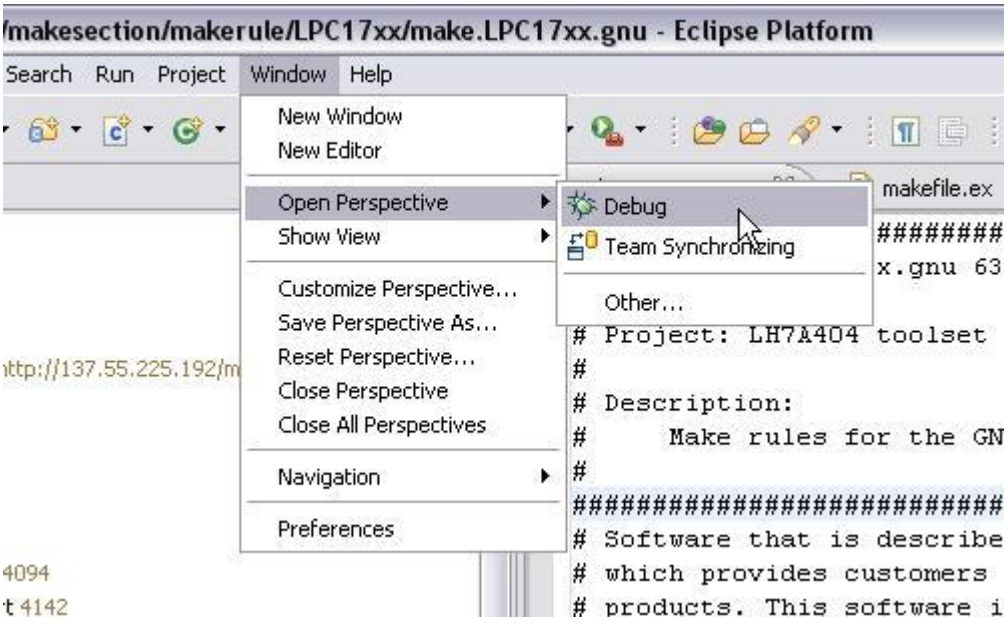


Fig 38. Open Perspective to debug window

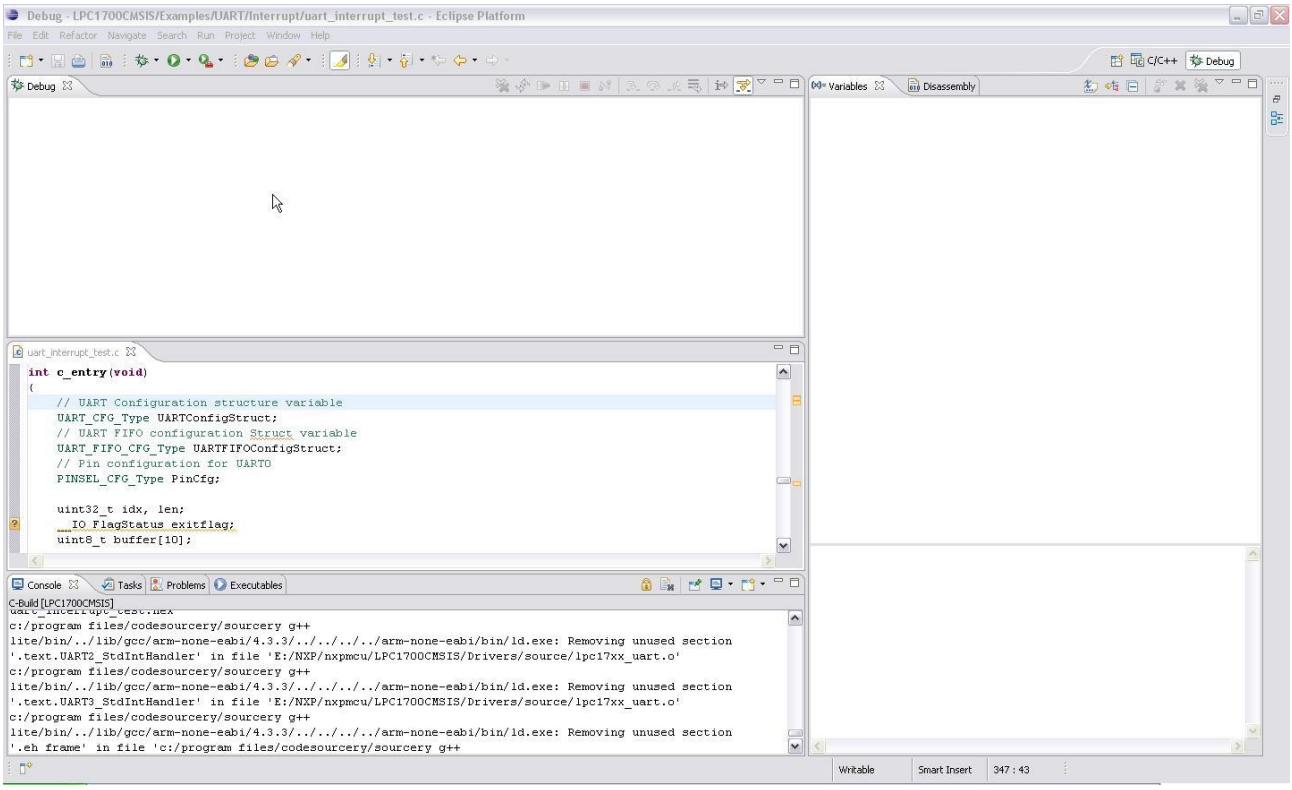


Fig 39. Overview of debug window

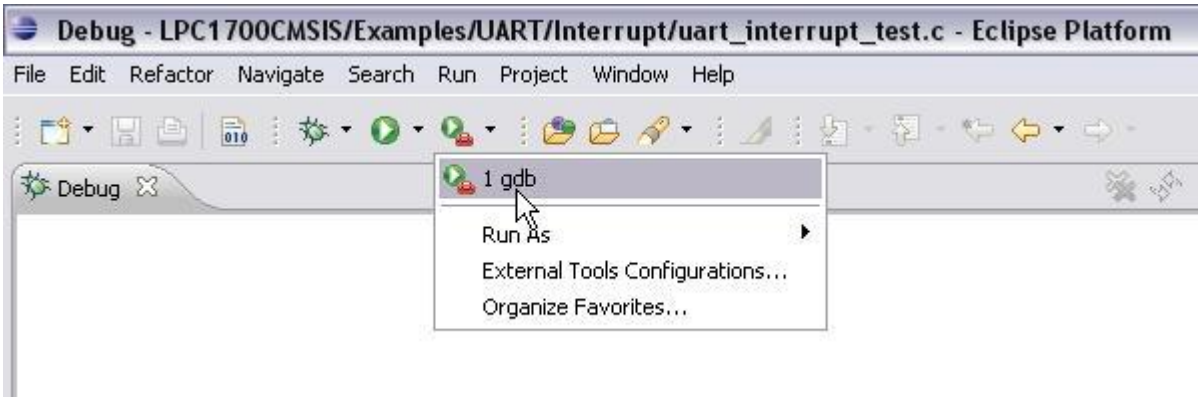


Fig 40. Open GDB server in external tool list

Fig 41. Start in free mode

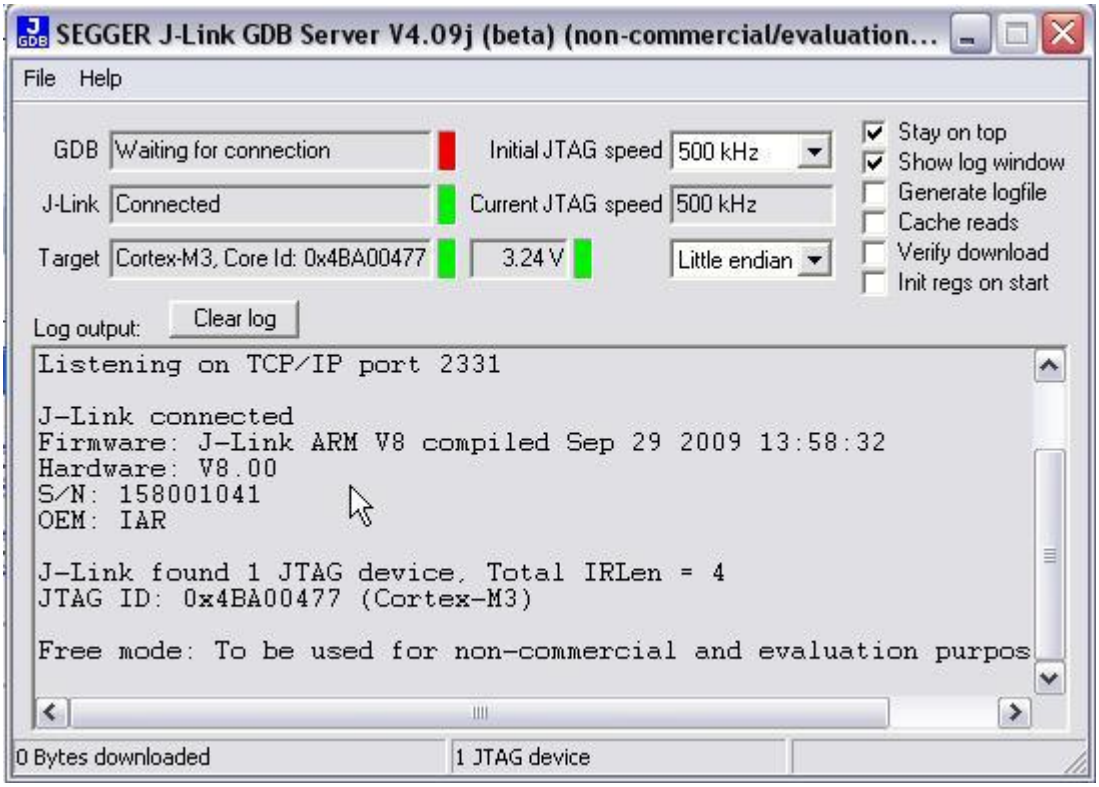


Fig 42. Configure parameter on J-link GDB server

Note: Remember configure for SEGGER J-link run in “little endian” mode

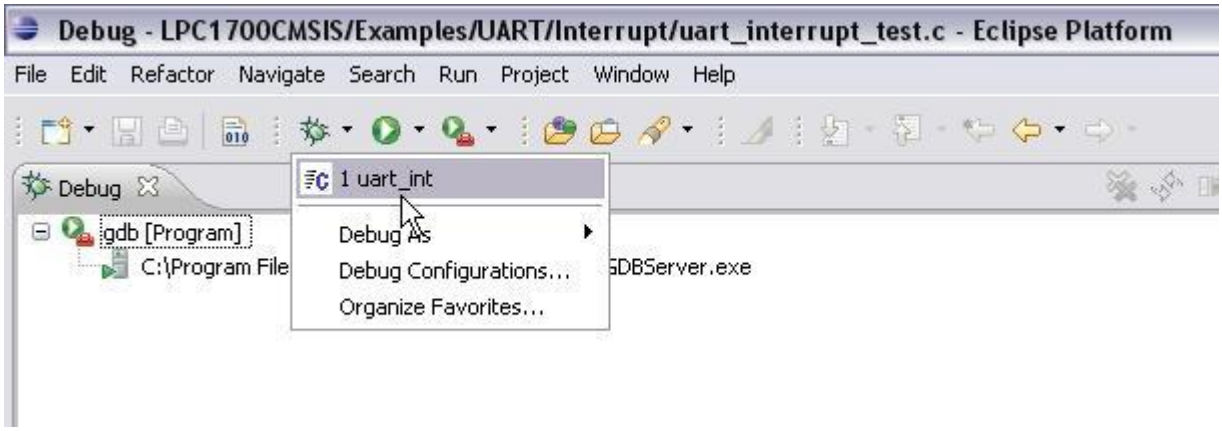


Fig 43. Open uart_interrupt_test debug session

Fig 44. All function buttons

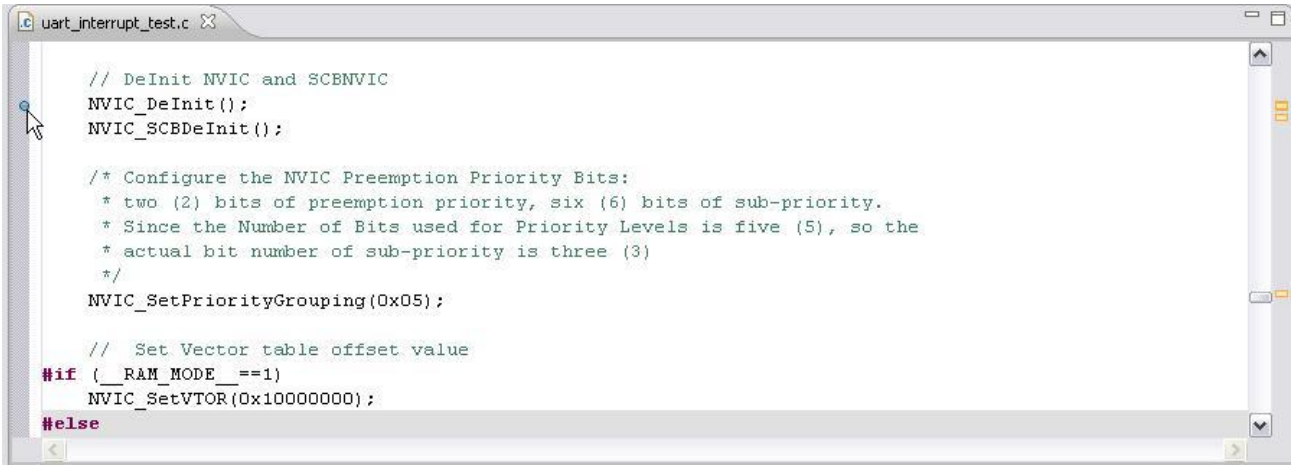


Fig 45. Set a break-point on uart_int.c file

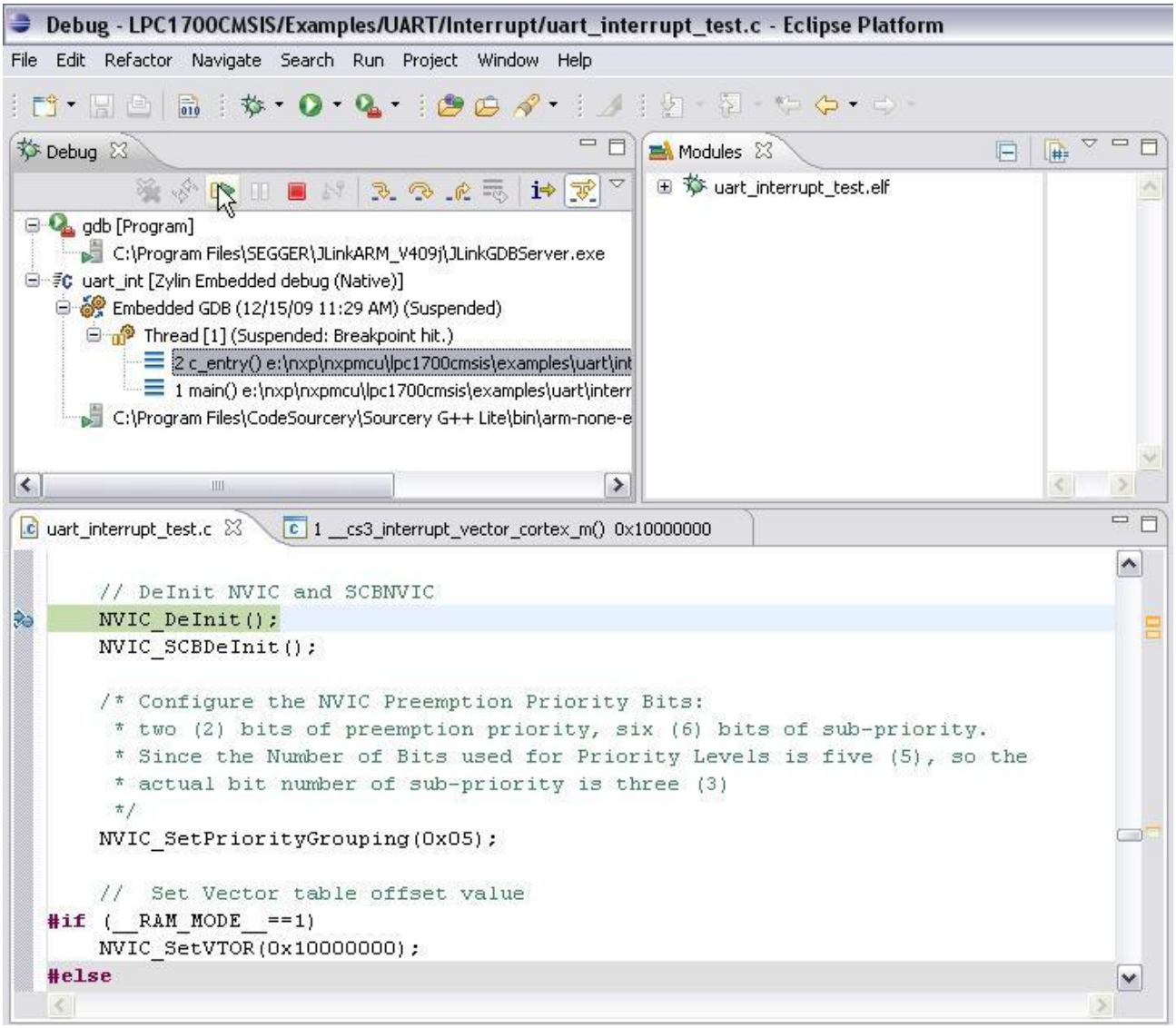


Fig 46. Run to the first break point

4.9.2 ROM mode

- Build the example in ROM mode first.
- Use Flash Magic to burn an image file (.hex) into the chip
- Start a debug session in ROM mode.

Steps to configure debugging session in ROM mode are the same with RAM mode.

In fact, Flash downloading using J-link is possible but this requires some license from SEGGER.

4.10 Clean project

Make target for clean project:

- Clean: should be named as 'cleanall' in each example, this target should be executed before any 'make' session!!!

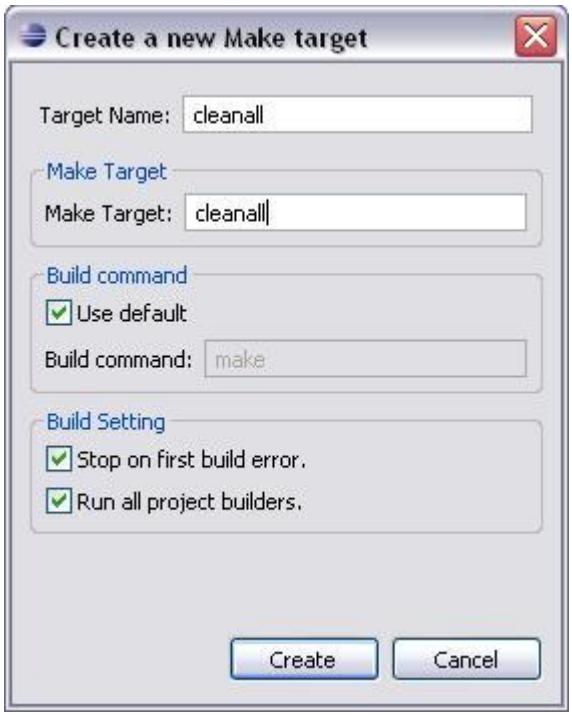


Fig 47. Make “cleanall” target

After clean, these example’s files are removed:

- *.o : object files (include driver’s object)
- *.elf: ELF files
- *.hex: HEX files
- *.map: MAP files
- *.srec: SREC files

Beside, “*.bin,*.dbo,*.dla, *.dnm, *.wrn, *.c.orig, *.h.orig, *.depend” files also be removed if existed.

4.11 Compiling in command line

The package project is supported to build without Eclipse IDE.

In this case, the “zip” package is extracted on C:\nxpdrv, and a folder named “LPC1700CMSIS” will be generated on C:\nxpdrv.

Modify the value of symbol “PROJ_ROOT” in the “makeconfig” file corresponding to the current location of project root (note that a forward slash is used in this case):

PROJ_ROOT = C:\nxpdrv\LPC1700CMSIS

Set environment: Refer to chapter 4.5.

Example that used to demo in this case is UART1_FullModem

4.11.1 Ram

Fig 48. Build example in RAM mode (Command prompt window)

4.11.2 Rom

Fig 49. Build example in ROM mode (Command prompt window)

4.11.3 Clean

Fig 50. Clean the example (Command prompt window)

5. Using LPC1000CMSIS driver package with extension tools

The project we used to represent here is LPC1700CMSIS. Other LPC1000CMSIS project are correlative.

5.1 Keil RealView – MDK

In each example has attached Keil project in Keil folder. This folder contents project file and all configure files that need to run this example on Keil platform. It also already included all required files (startup, core, driver source files).

Here is the steps to run an exits Keil example or import an demo source file into Keil project sample.

5.1.1 Keil example project

- Run *.uvproj file in Keil folder. Such as follow '.\Example\ADC\Polling\Keil' to open adc_polling.uvproj file
- Chose correct target to run (RAM/ROM mode)

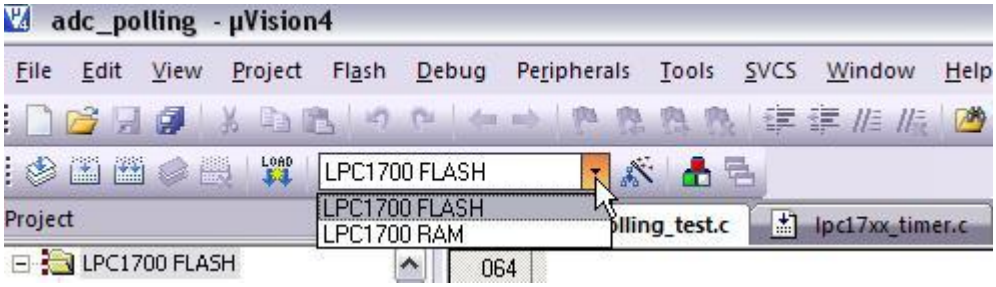


Fig 51. Choose target project

- Click 'Build' icon or 'Project > Build target' to compile this example

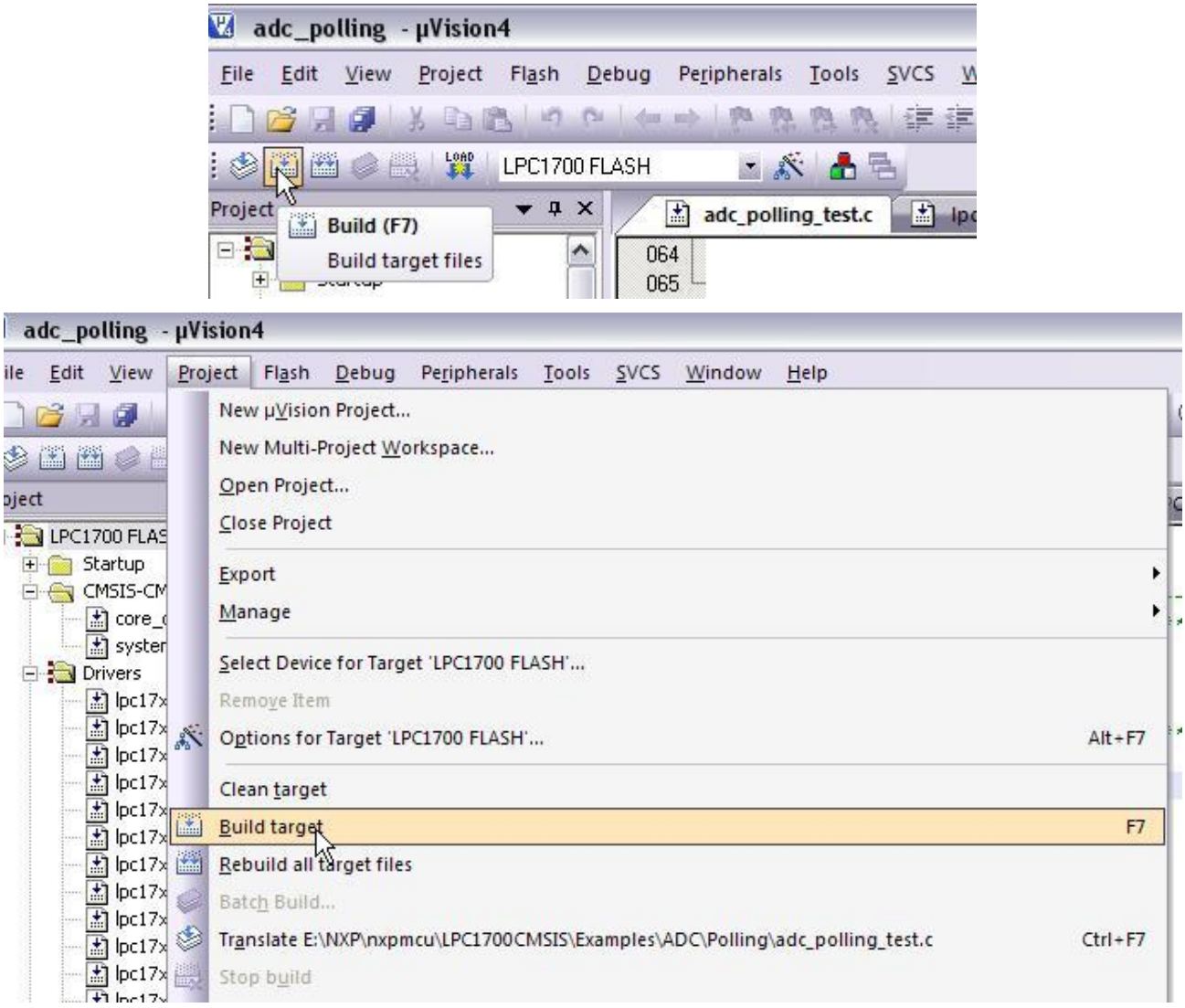


Fig 52. Choose target project

- Reference ‘readme.txt’ to get more information about hardware configuration and how to run this example.
- If choose ROM target, click ‘Download’ icon or ‘Flash > Download’ to burn image file into board
- reset board and run

Note: Project set ULINK Cortex Debugger as default. But you also can burn by Flash Magic or other flash download tools.

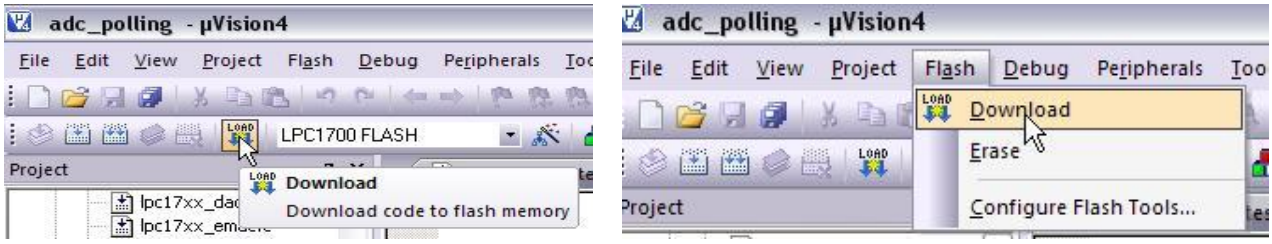


Fig 53. Build project

- If choose RAM target or want to debug in ROM mode, click “Debug > Start/Stop Debug Session” to start debug operation

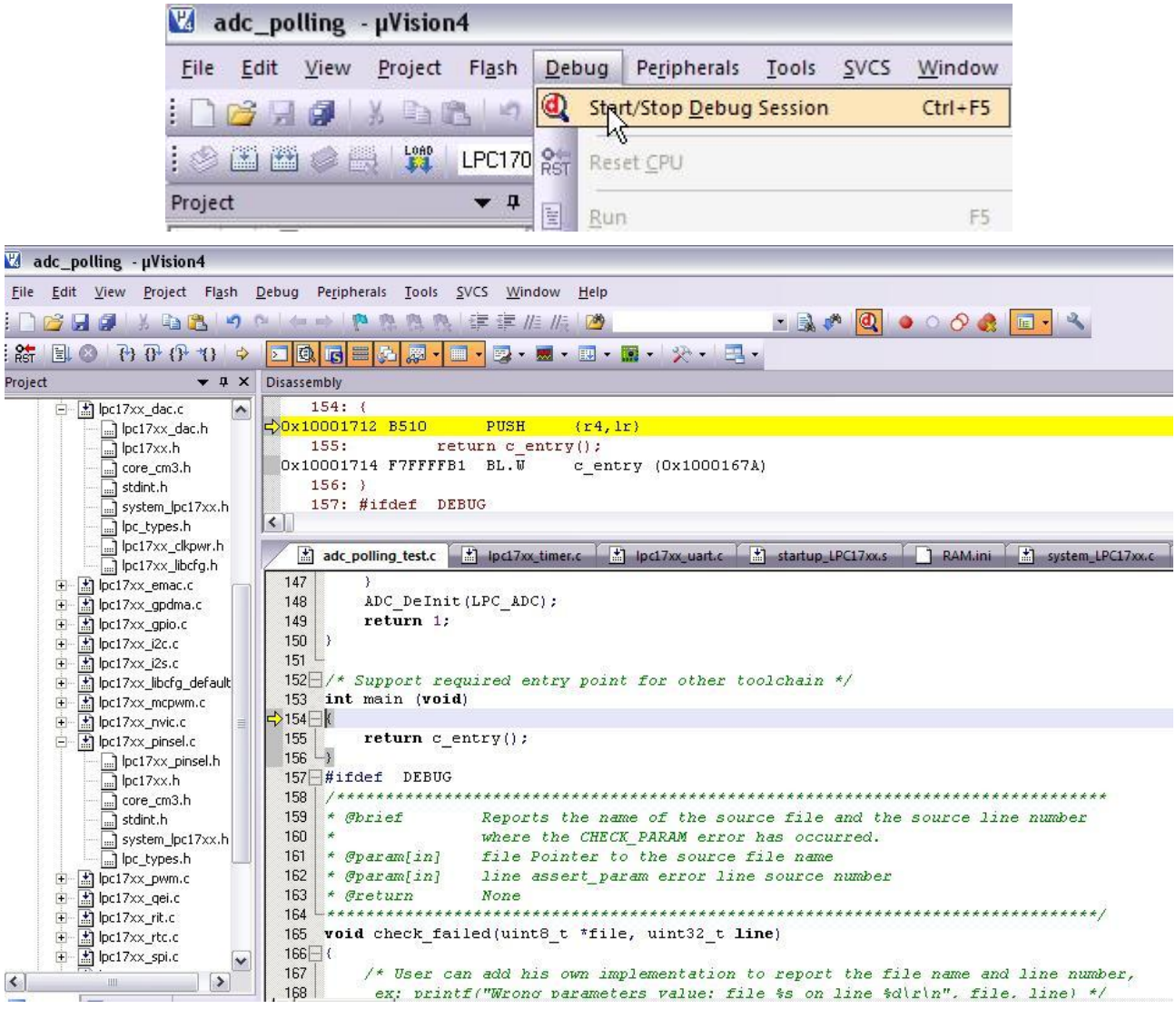


Fig 54. Build project

5.1.2 Keil project template

There's a Keil Project template inside .\Examples\Project_Template\Keil_RealView_MDK directory.

This project is configured to add all required startup file (asm file used for RealView compiler), ARM Cortex-M3 core files and driver source files.

Following target properties have been configured:

- Target device (LPC1768)
- Memory layout with linker.
- Output and list directory.
- C/C++ definitions and Common C/C++ Include Paths (.\Core\CM3\CoreSupport; .\Core\CM3\DeviceSupport\NXP\LPC17xx; .\Drivers\include)
- Flash programming and debug tool.

The main program is template.c for demonstration. However, this demo program can be easily removed to add other main program source files for user's purposes.

Here're all steps to import demo (main program) source file into exist Keil RealView-MDK project sample:

- Open Keil RealView project template

Follow .\Examples\Project_Template\Keil_RealView_MDK to open LPC1700CMSIS.uvproj (or .uv2 with older Keil project version) project file.

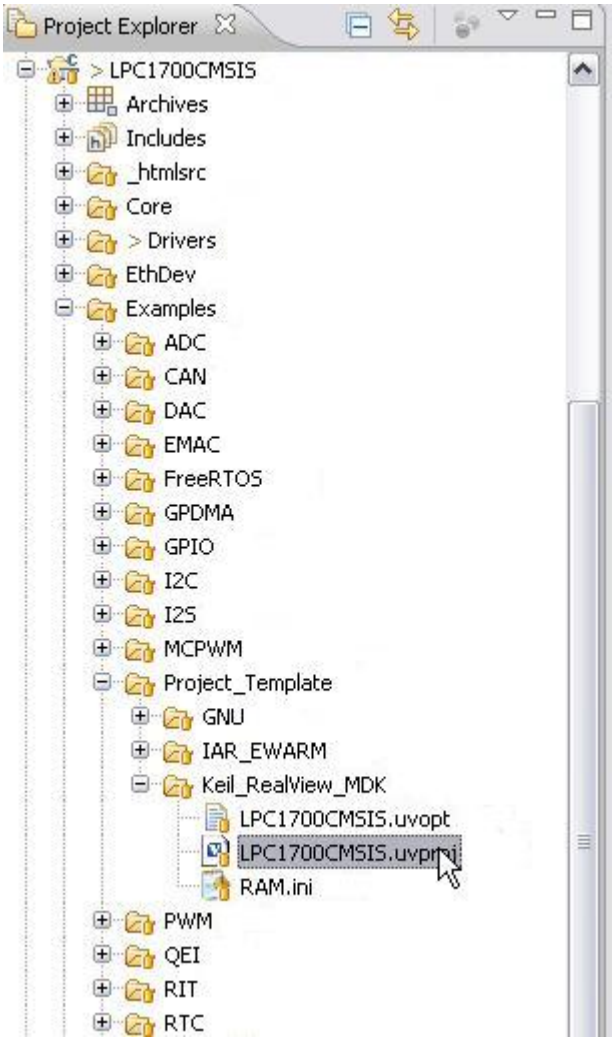


Fig 55. Open Keil RealView-MDK project sample

- Choose target mode

Setting RAM/ROM mode by choosing target:

- ROM mode: chose ‘LPC1700 ROM’ target
- RAM mode: chose ‘LPC1700 RAM’ target

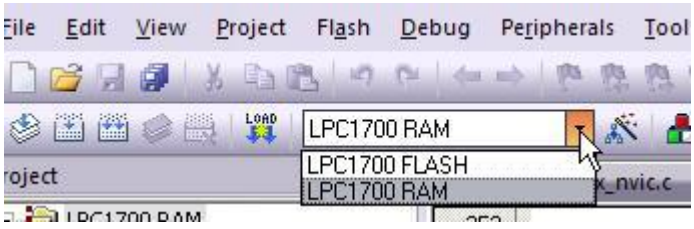


Fig 56. Choose target project

- Add main program source files

ADC Polling example is used in this case. Add all required source files (*.c) in .\Examples\ADC\Polling directory.

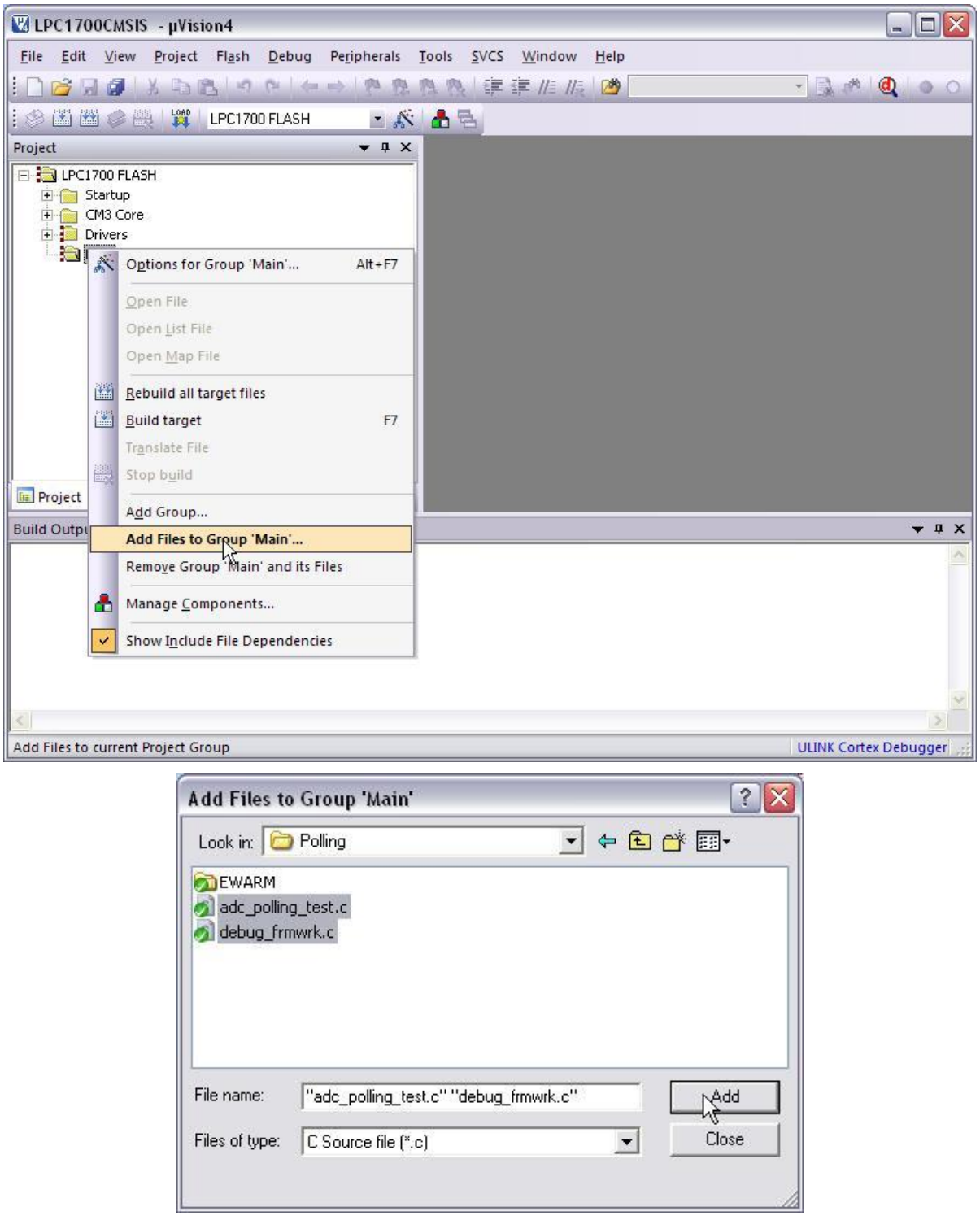


Fig 57. Add main program source file to project

- Add included Path

Add .\Examples\ADC\Polling into C/C++ Include Path of project.

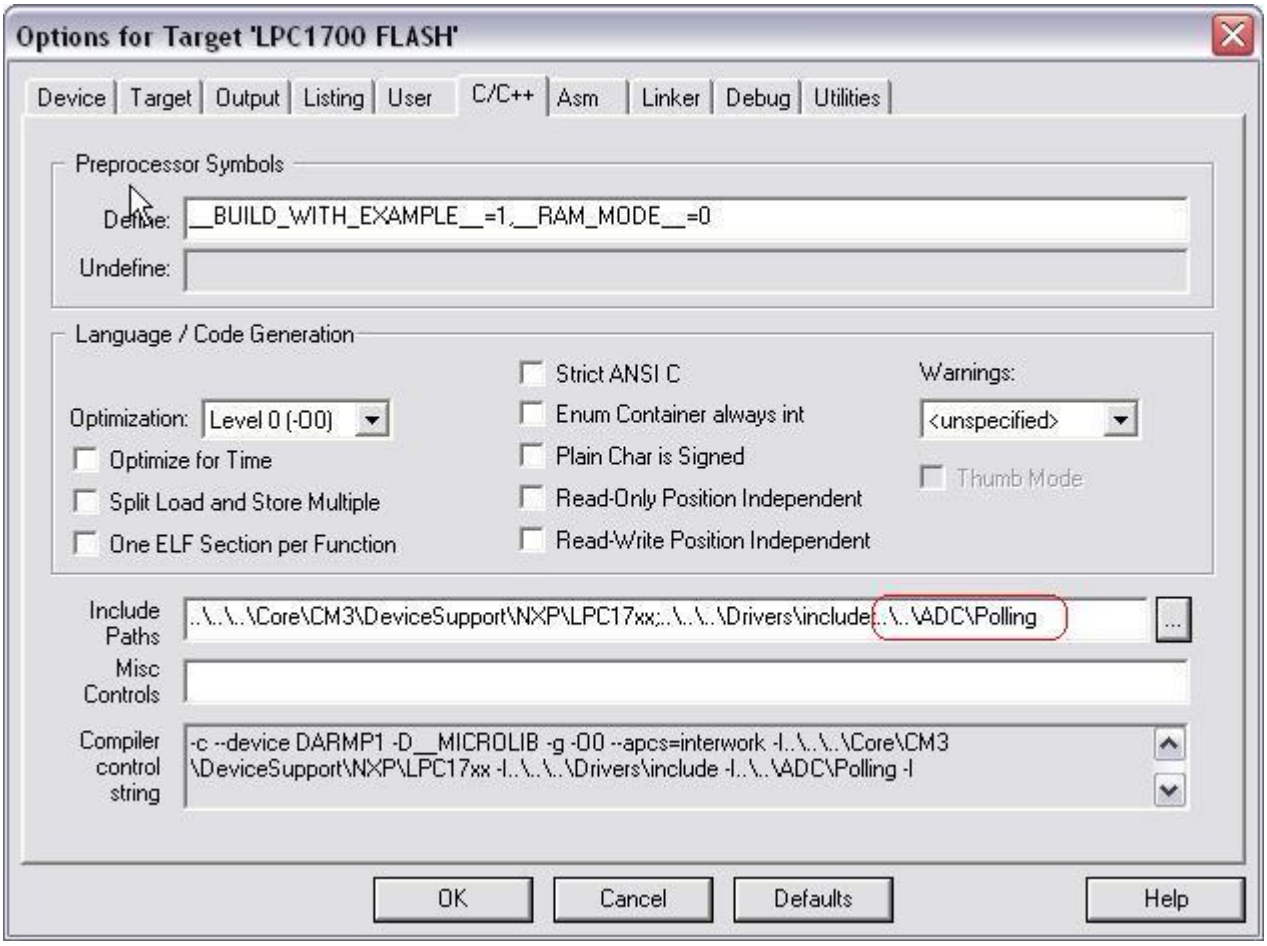


Fig 58. Add include path

Note:

There's a file named "lpc17xx_libcfg.h" in each example directory. This file used to enable/disable individual peripheral driver in driver source code for compiler to build.

Specified symbol should be defined in order to enable corresponding peripheral driver used in main program.

For example:

If ADC, UART0 and TIMER are used in this demo program, corresponded symbol for these two peripheral drivers should be defined:

#define _ADC

#define _UART

#define _UART0

#define _TIM

- Build project and run

The rest is very common. Execute build then burn image file into working board and let it run!

5.2 IAR Embedded Workbench IDE

In each example has attached IAR project in EWARM folder. It allows you run this example in IAR Embedded Workbench IDE. This project is configured to add all required startup file (asm file used for IAR compiler), ARM Cortex-M3 core files and driver source files.

Here're all steps to run exist IAR examples or import demo source file into exist IAR project sample.

5.2.1 IAR example project

Run .eww file in EWARM folder. Such as: follow .\Example\ADC\Polling\EWARM to open adc_polling.eww file.

Click "Dowload and Debug" icon to start building and let it run.

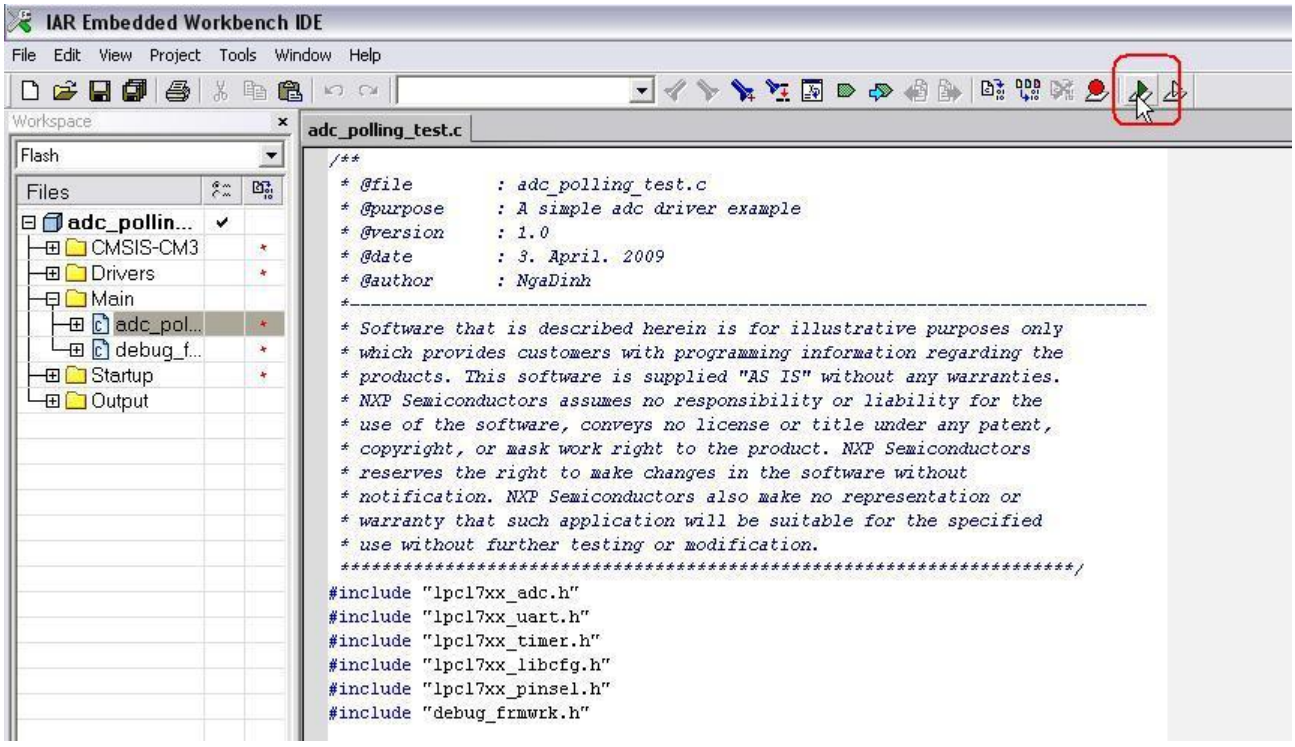


Fig 59. Run IAR example

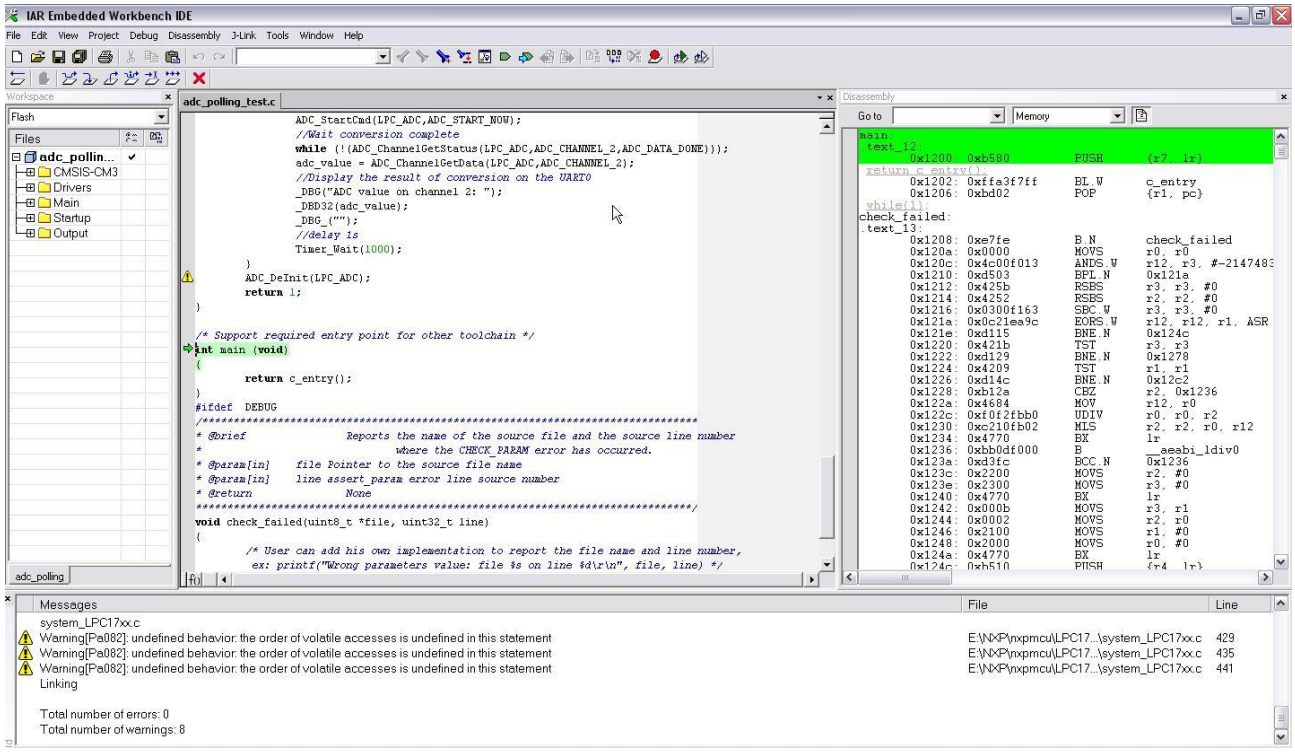


Fig 60. Debug IAR window

Switch to RAM workspace if want to run in RAM mode.

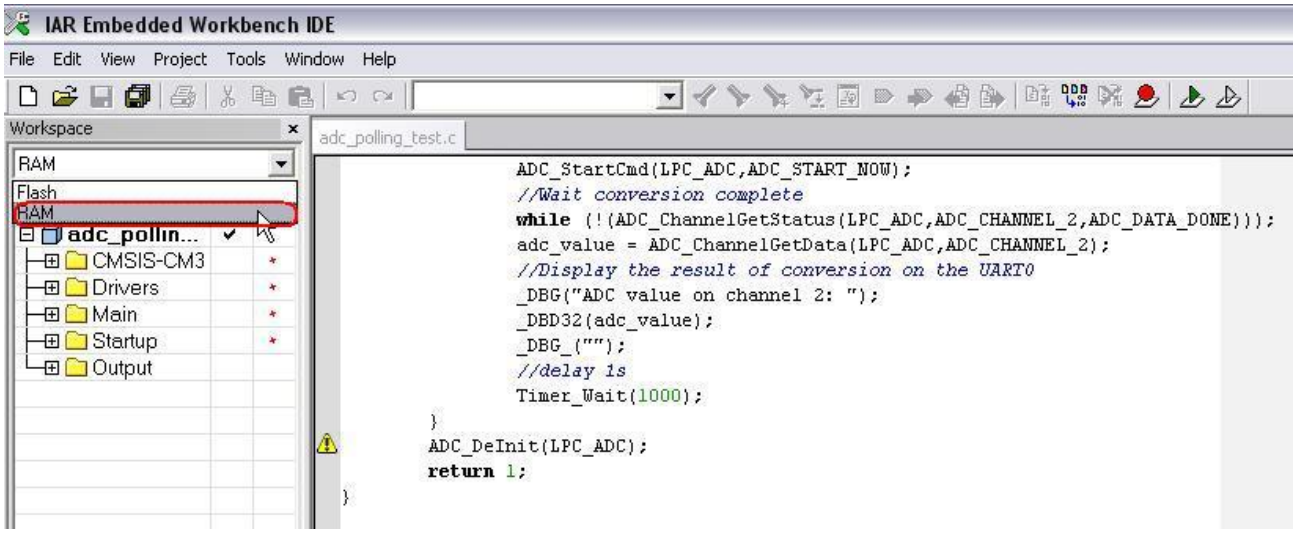


Fig 61. Switch to RAM mode

5.2.2 IAR project template

There's a IAR Project template inside .\Examples\Project_Template\IAR_EWARM directory. The main program is template.c for demonstration. However, this demo program can be easily removed to add other main program source files for user's purposes.

- Open IAR project

Follow .\Example\Project_Template\IAR_EWARM directory to open LPC1700CMSIS.eww file.

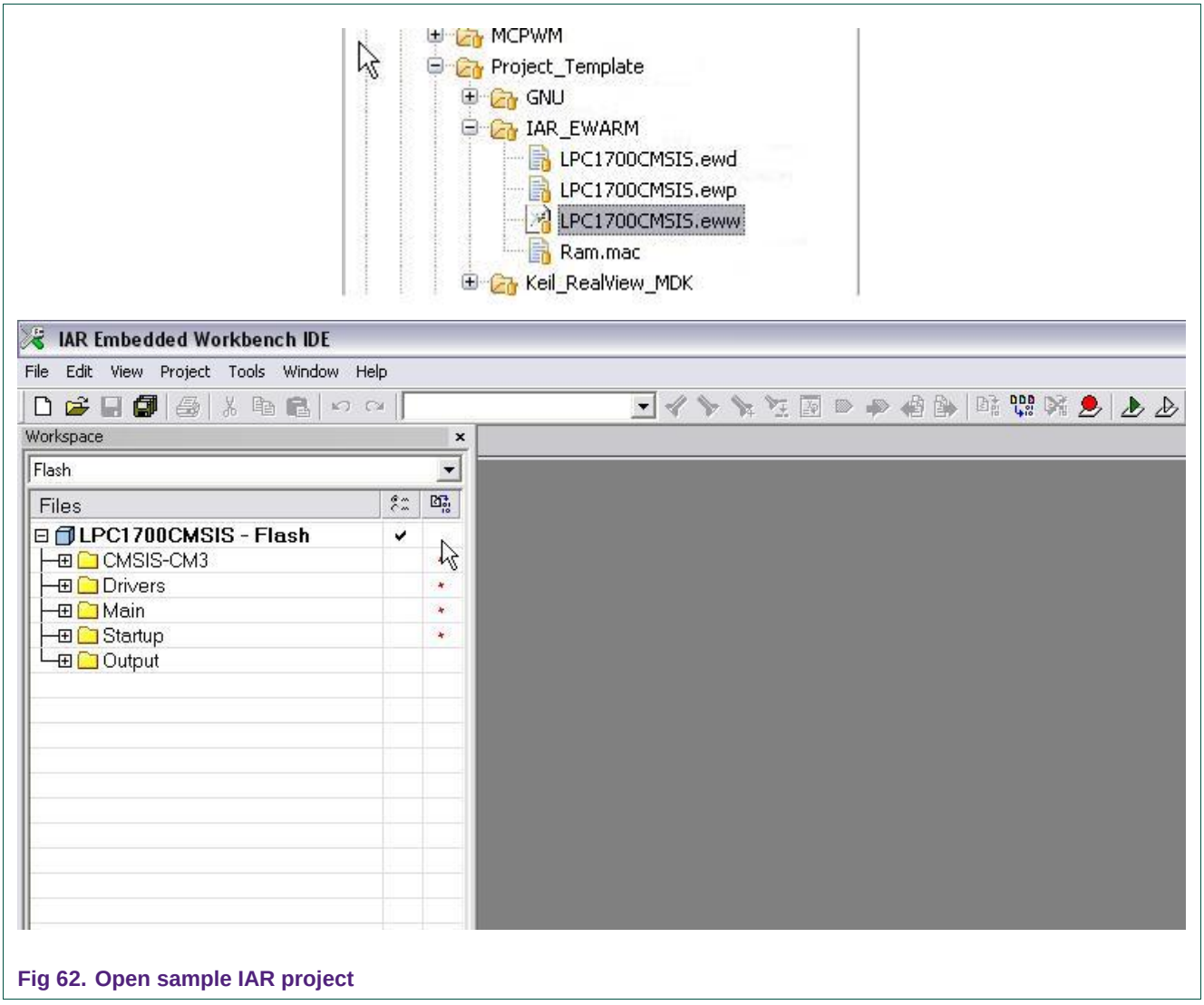
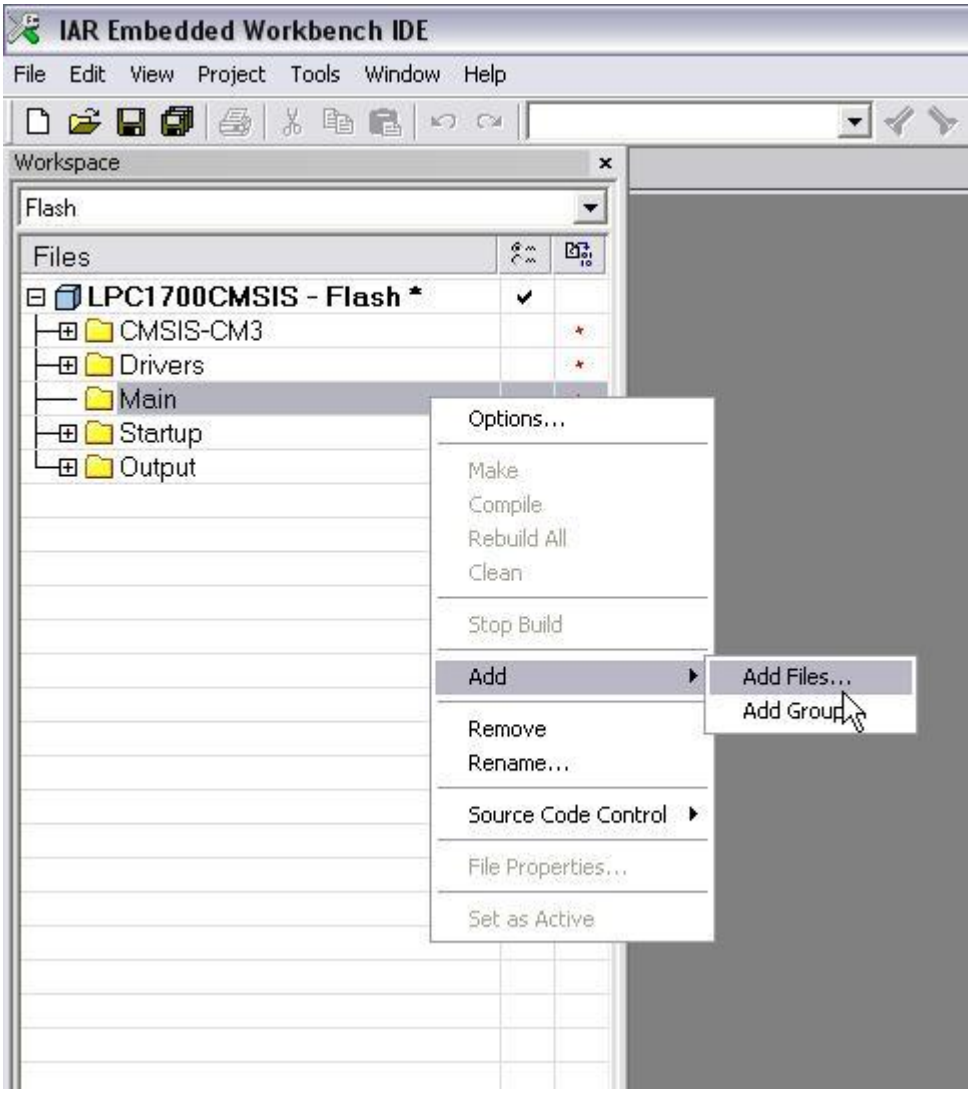


Fig 62. Open sample IAR project

- Add main source code

ADC Polling example is used in this case. Add all required source files (*.c) in .\Examples\ADC\Polling directory:



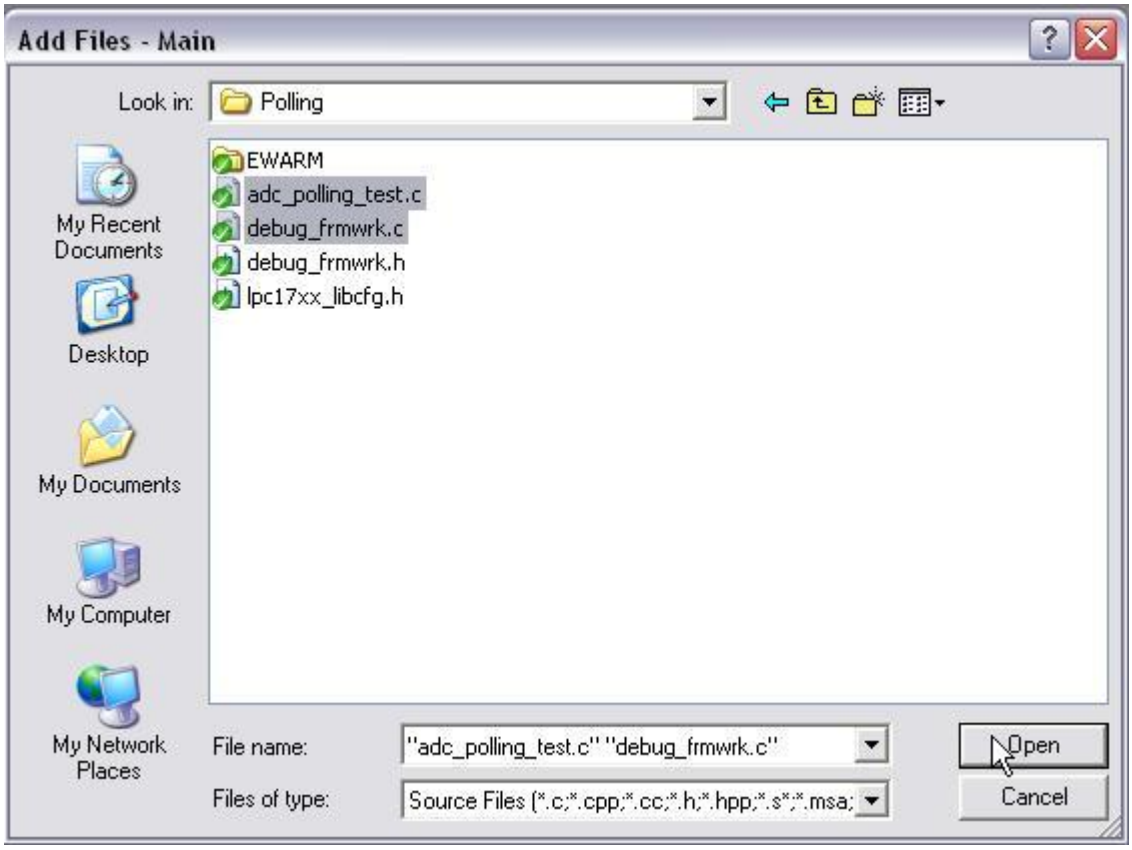


Fig 63. Add source code in IAR project

- Add include path

Open Option window by right click into LPC1700CMSIS-Flash workspace.

Add .\Examples\ADC\Polling into C/C++ Compiler/ Preprocessor tab as follows:

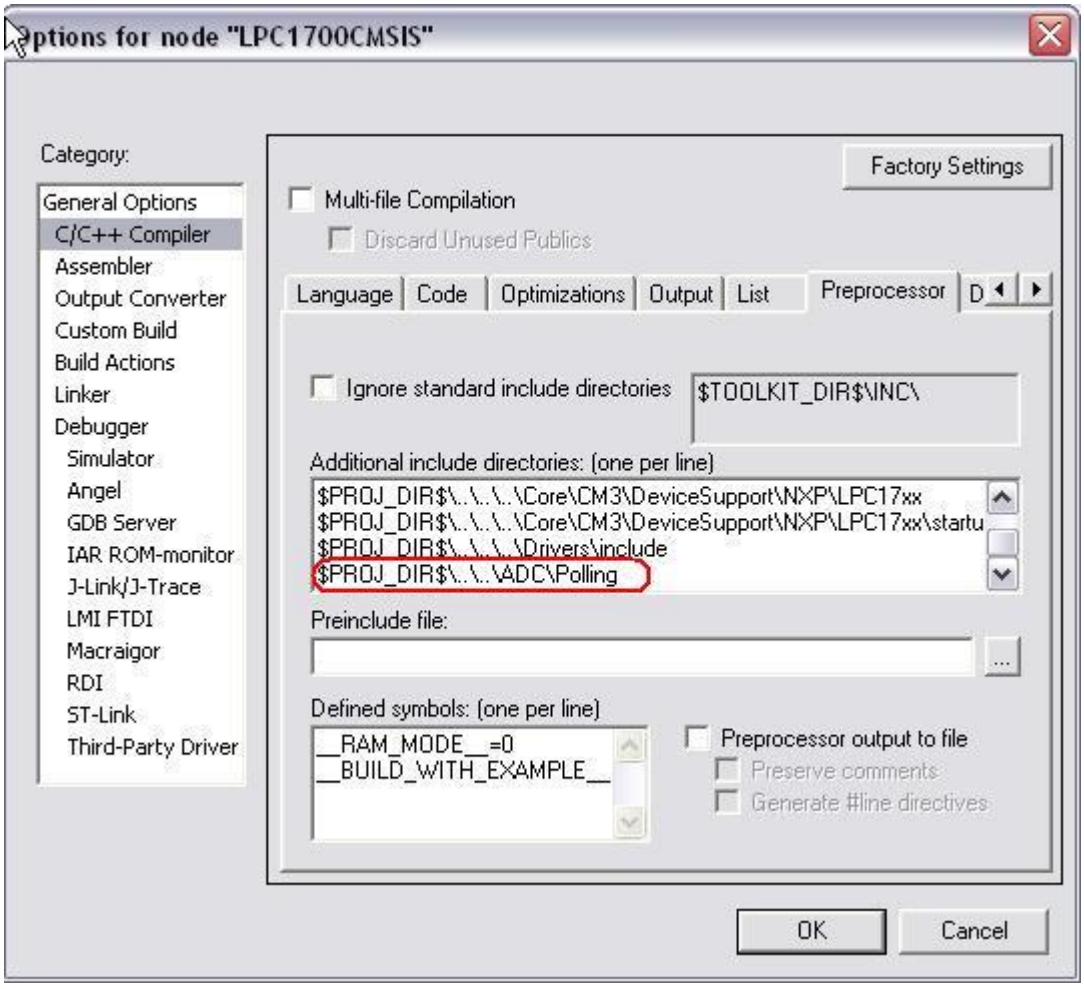


Fig 64. Add include path

Now, follow the same steps in “Use exist IAR example” to run it.

6. Appendix

Project Files Structure Diagram

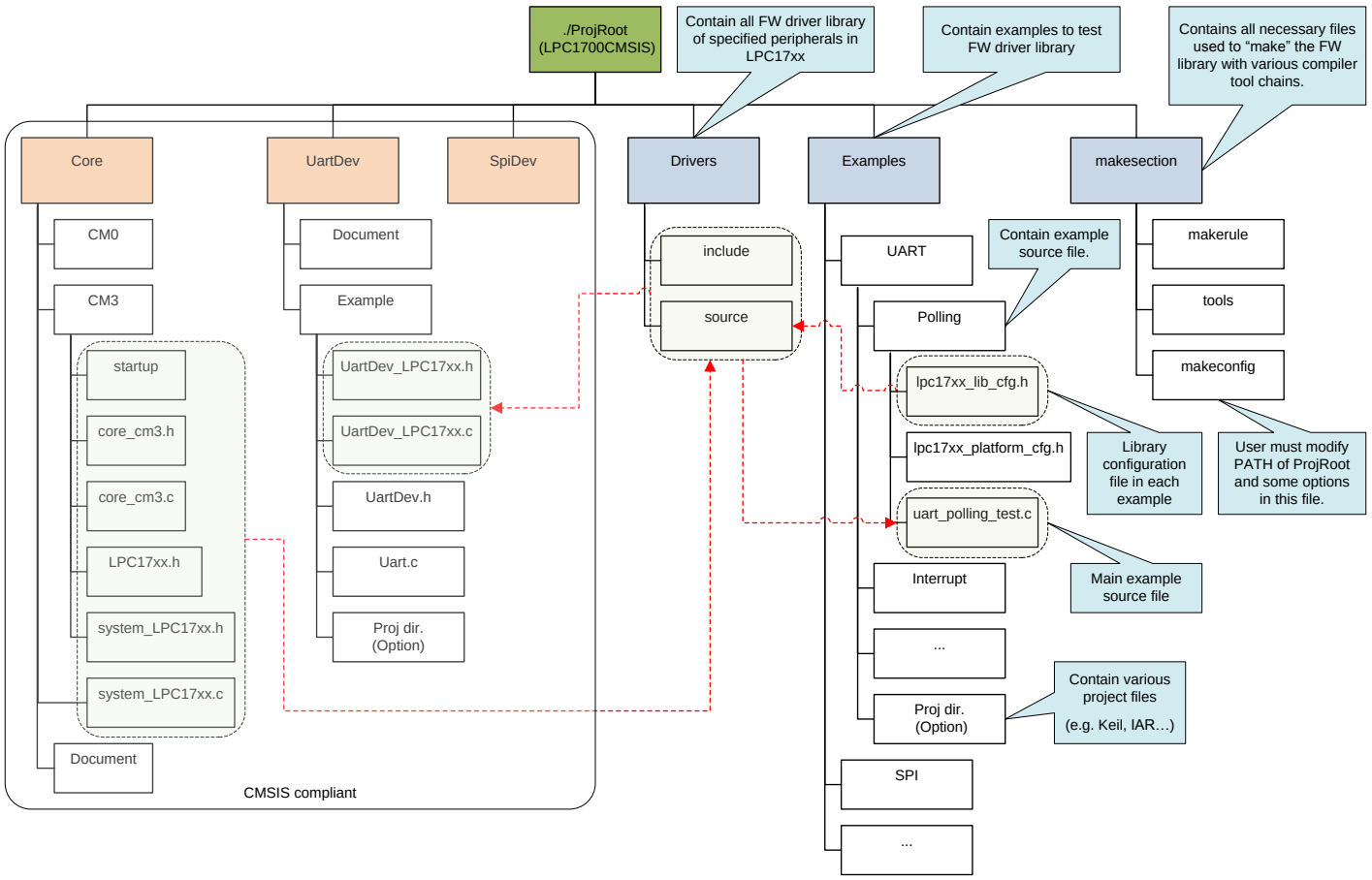


Fig 65. Project Files Structure Diagram

7. Legal information

7.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

7.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

8. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: 21 May 2010
Document identifier: AN10862_3