Build Environment Setup - Eclipse

This document illustrates how to build Realtek Wi-Fi SDK under Eclipse environment.

Table of Contents

1 Introduction 3
2 Environment setup 3
2.1 Cygwin package 3
2.2 Eclipse environment 5
2.2.1 Eclipse IDE installation 5
2.2.2 Plugin installation 5
2.2.3 Import Ameba SDK application 7
2.2.4 Configure debug and external tool launchers 10
2.3 Debugger setting 17
2.3.1 OpenOCD/CMSIS-DAP 17
2.3.2 J-Link 21
3 How to build, download and enter debug mode 24
3.1 Build code 24
3.2 Download code to flash 28
3.3 Download and debug in RAM 32
4 Troubleshooting 34
4.1 Unable to execute run_openocd.bat normally on Windows 34
4.2 Make not found in PATH error during project building 36
4.3 Build fail due to relative path header file not found 37
4.4 Download procedure hang for a long time 38
4.5 Next step not correct in debug mode 38

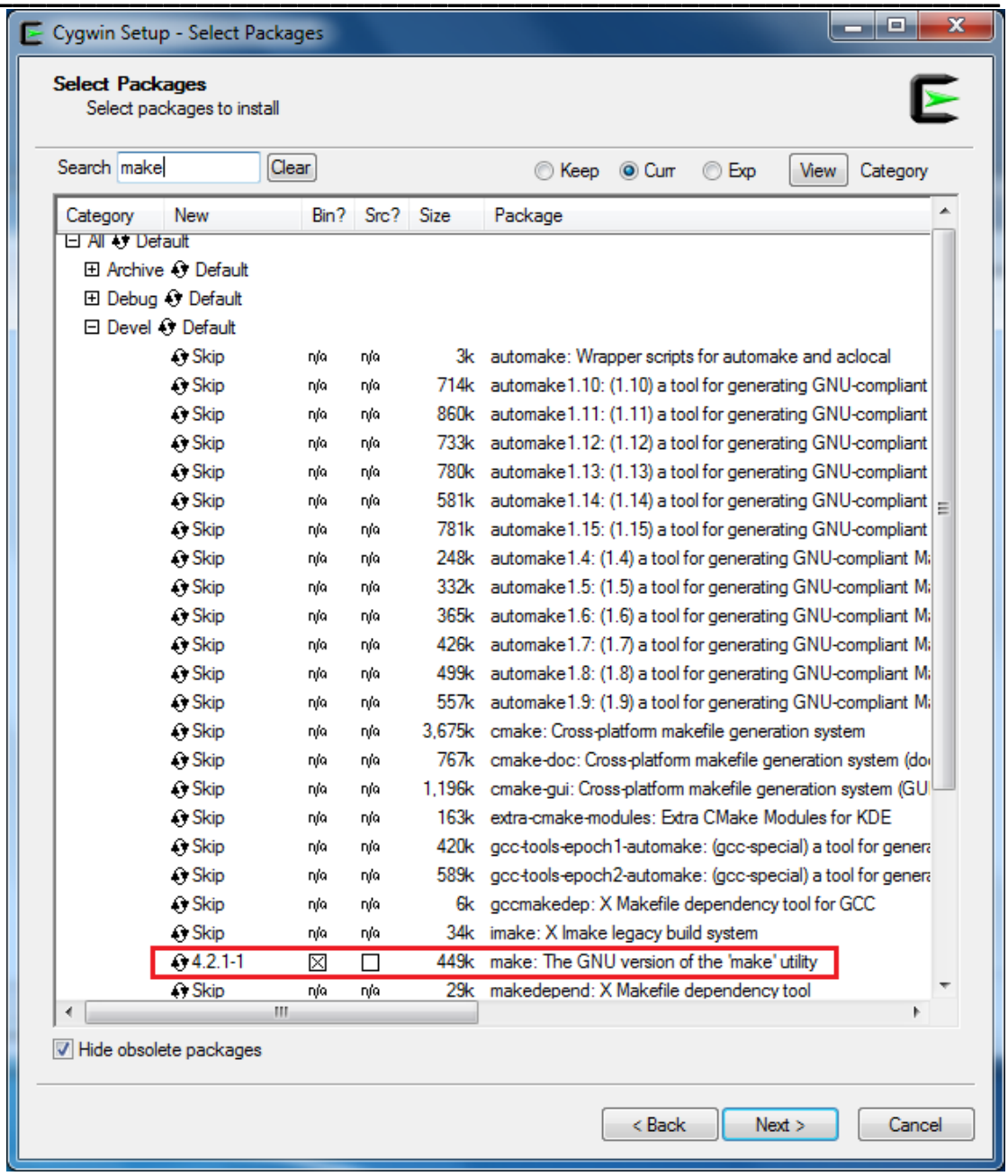
1 Introduction

This document illustrates how to build Realtek Wi-Fi SDK and download it to flash under Eclipse + GDB environment. In this document, we use Windows 7 64-bit as our platform but it should be applicable for other Windows platforms. Note that for now, **Linux platform is not supported yet**.

2 Environment setup

2.1 Cygwin package

Since we use GNU make builder to build our SDK, we need to install Cygwin for the GNU tools. Cygwin is a large collection of GNU and open source tools which provide functionality similar to a Linux distribution on Windows. Please check http://cygwin.com and download the Cygwin package for your Windows platform. During the installation of Cygwin package, please include '**Devel -> make**' utilities on the Select Packages step:



After the installation, please add the Cygwin installation path to system path. For example, append “C:\cygwin64\bin\” to the value of Path system variable which located at Control Panel -> System and Security -> System -> Advanced System Settings -> Advanced System Settings -> Advanced tab -> Environment Variables -> Path.

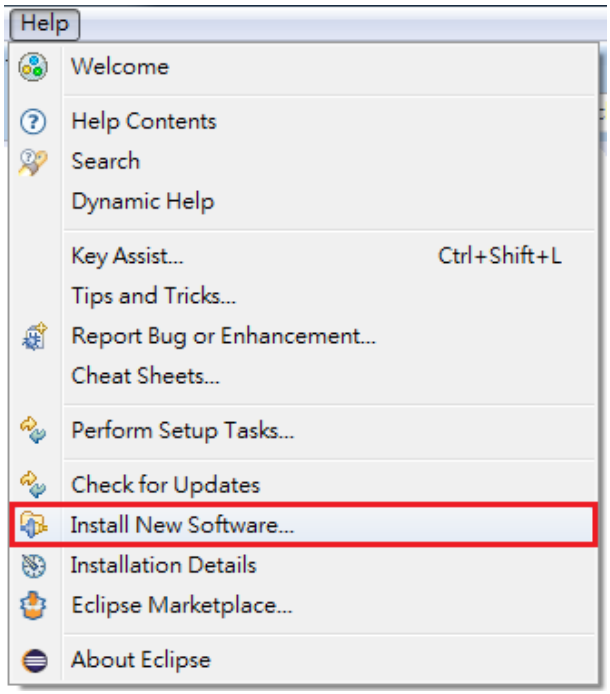
2.2 Eclipse environment

2.2.1 Eclipse IDE installation

For the Eclipse environment, we need to get “Eclipse IDE for C/C++ developers”. Please make sure you have installed Java Runtime Environment (JRE) before installing Eclipse environment. In this document we use JRE 8 as the java platform but you can simply get the newest version from http://www.oracle.com/technetwork/java/javase/downloads/index.html. And for “Eclipse IDE for C/C++ developers”, we use the Mars release version which you can download from http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/marsr. After the download procedure finished, un-compress it and execute the eclipse.exe inside the unzipped eclipse/ folder to start the Eclipse IDE.

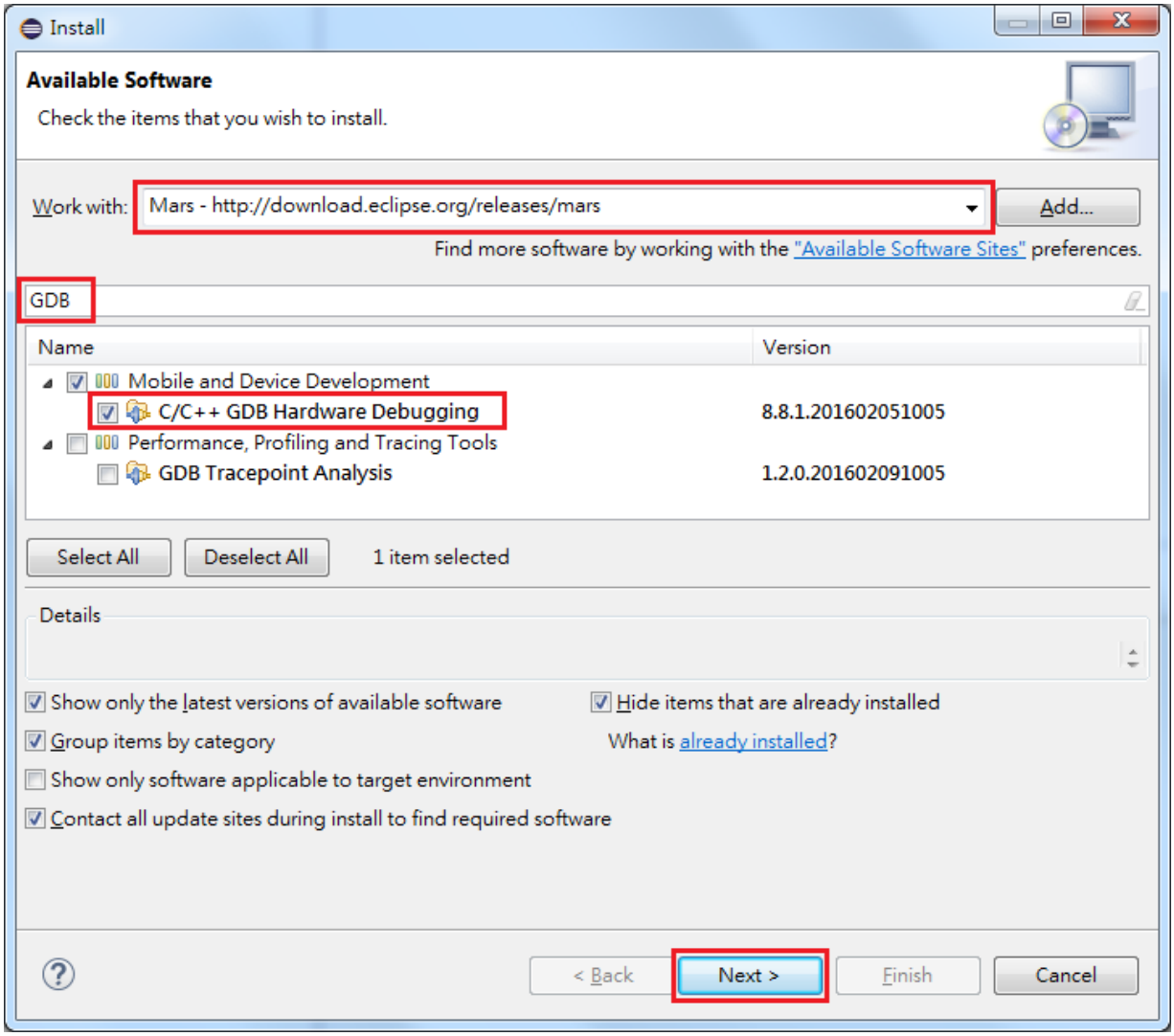
2.2.2 Plugin installation

Now, we need to install “C/C++ GDB Hardware Debugging” plugin since we want to use GDB debugger in Eclipse environment. Please click “Help -> Install New Software…” on the top of the Eclipse IDE.



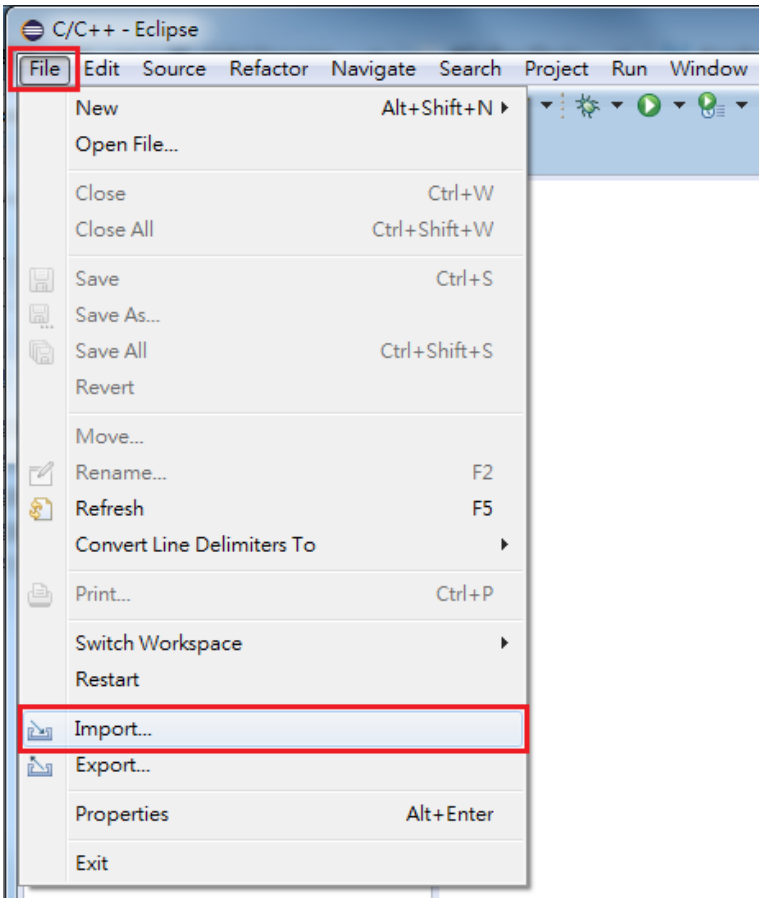
In the popped out window, you should choose an available server side on the section “Work with:” to get the available software. You can simply click the dropdown button to choose a software side. For example, we select the side called “Mars” and the middle section will shows the available software you can install in a few seconds. Please find the software named “C/C++

GDB Hardware Debugging" and select it. You can type "GDB" inside the "type filter text" box to help you locate the plug-in we need (like below figure). Make sure you select the "C/C++ GDB Hardware Debugging" and then finish its installation process by simply keep clicking "Next" button and accept its license agreement. It will take a few minutes to install it and at the end it might ask you to restart Eclipse IDE to make it work.

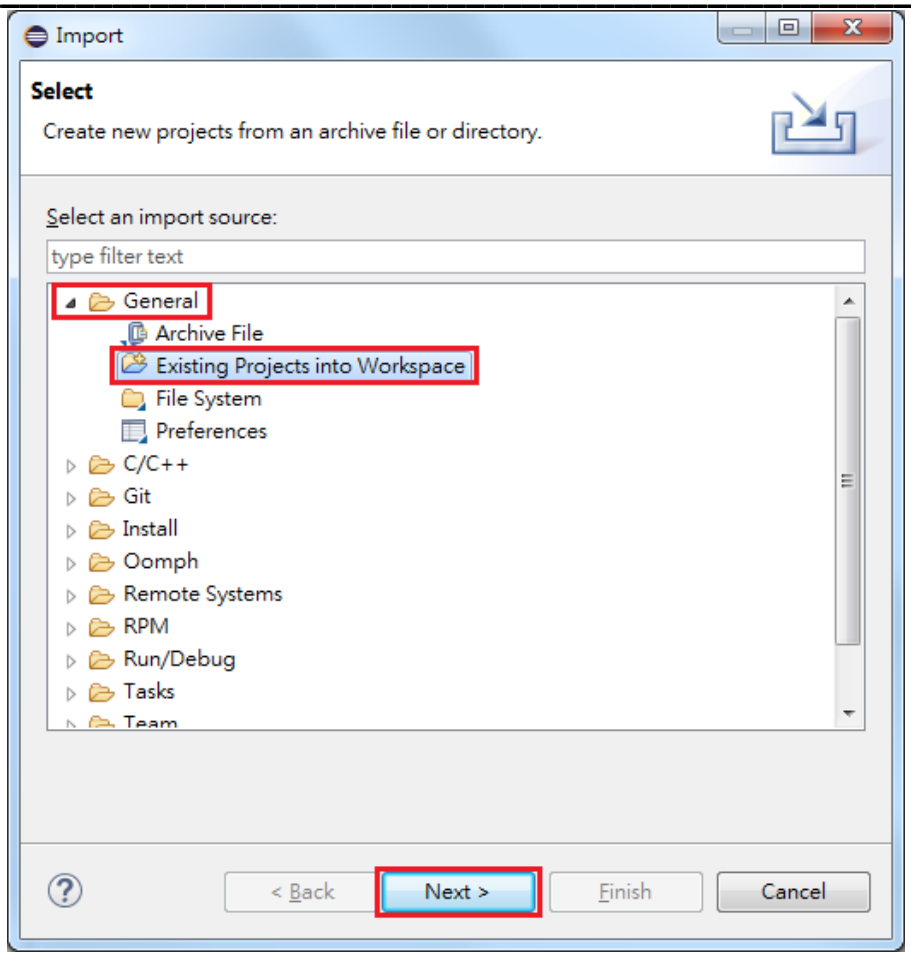


2.2.3 Import Ameba SDK application

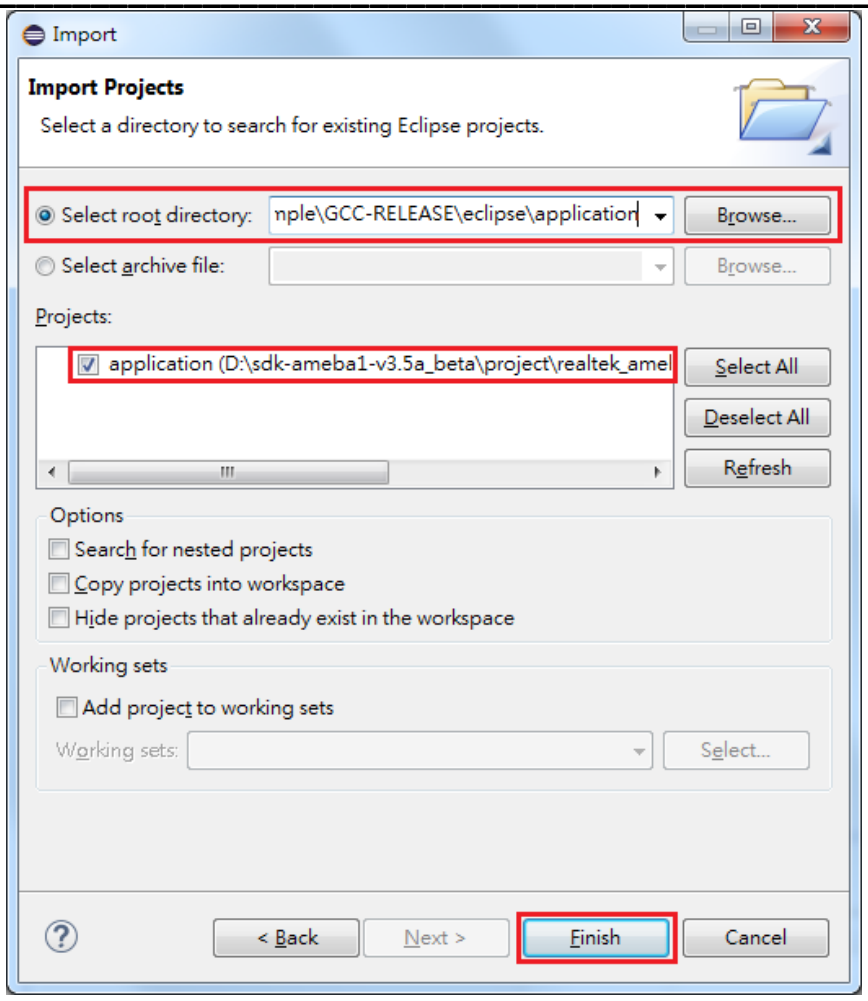
After the plugin installation, we now import the application project into Eclipse workspace. Please click “File -> Import…” on the upper left corner of Eclipse IDE:



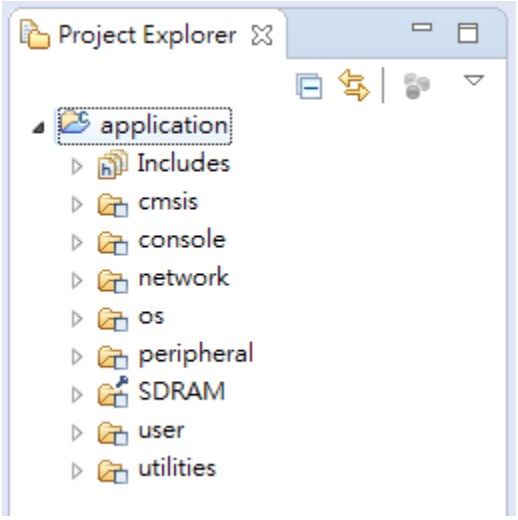
It should pop out an “Import” window, please select “General -> Existing Projects into Workspace” and click “Next” button:



In next window you should select the project directory like following figure. Please check "Select root directory" and click the "Browse…" button on the right-hand side to select **\SDK_LOC**\project\realtek_ameba1_va0_example\GCC-RELEASE\eclipse\application folder. You should be able to see "application" is showed in the "Projects" section at the middle of the window. Click "Finish" button to finish the import procedure.



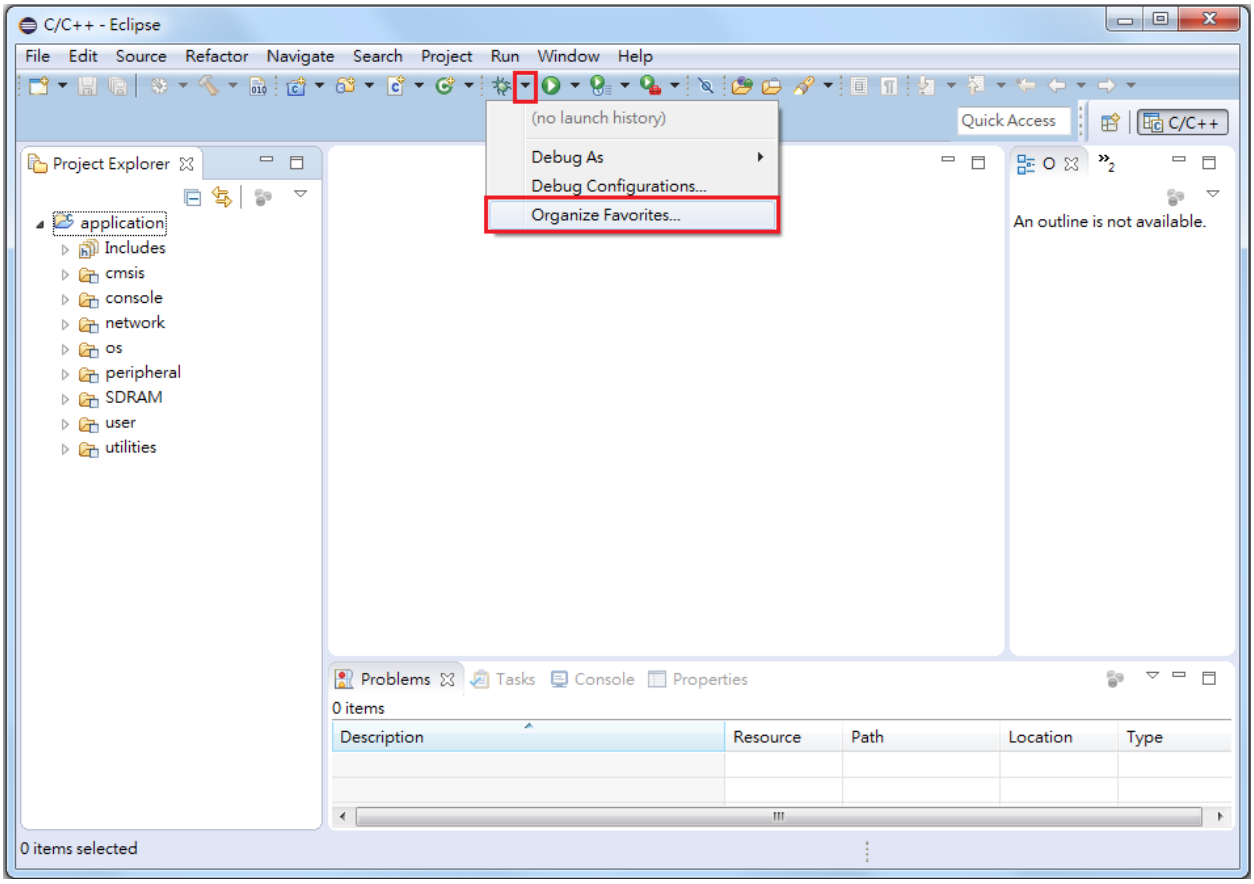
There should be an “application” project in “Project Explorer” view. The source files of SDK are listed under their corresponding directories in the project.



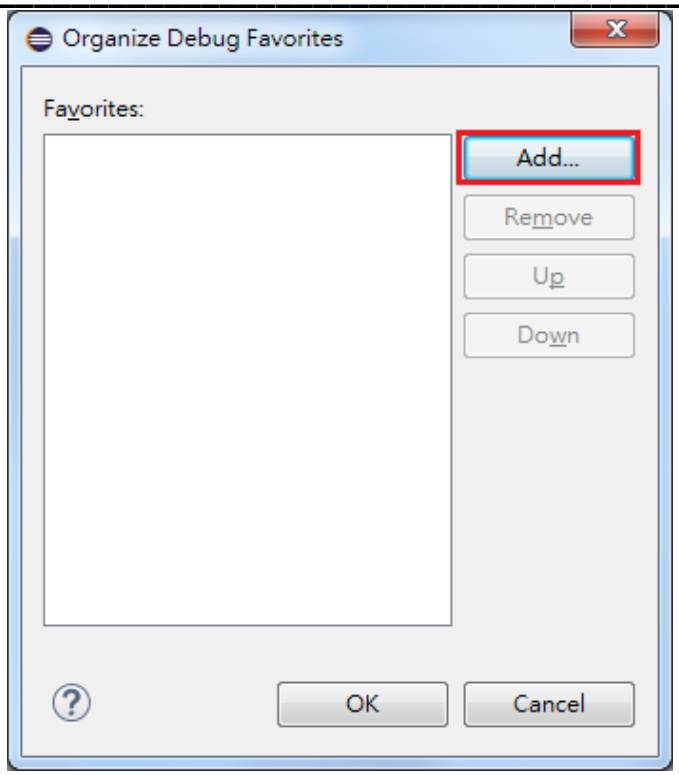
2.2.4 Configure debug and external tool launchers

After the import process finished, we now need to configure debug and external tools for ram debug and flash download. These debug and external tools launchers are located in *.setting/* folder. Please add these launchers to your Eclipse workspace by following the below steps.

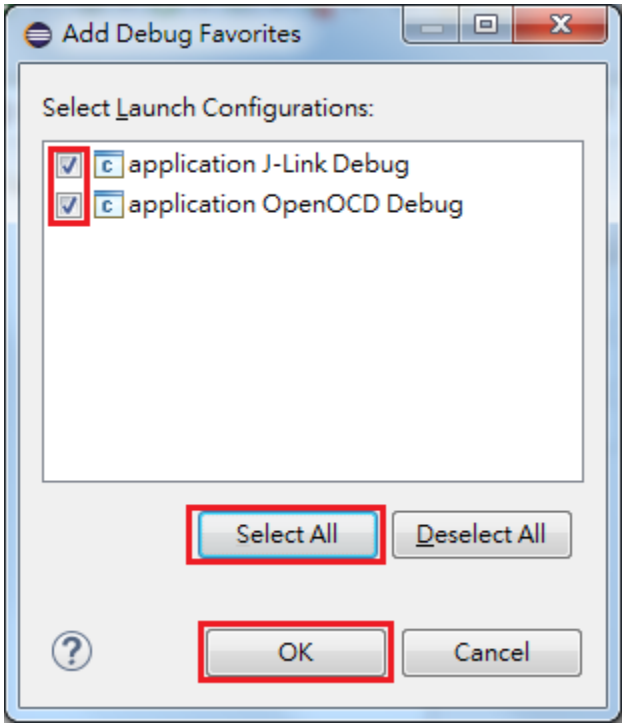
For debug launchers, please click “Debug -> Organize Favorites…”:



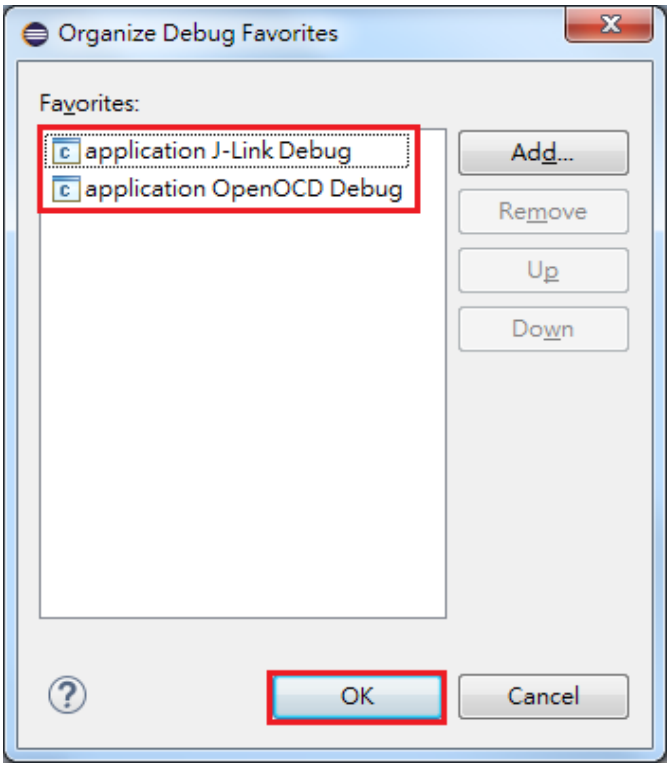
If the popped out window does not contains “application J-Link Debug” and “application OpenOCD Debug” launchers like below figure, click the “Add…” button to add them.



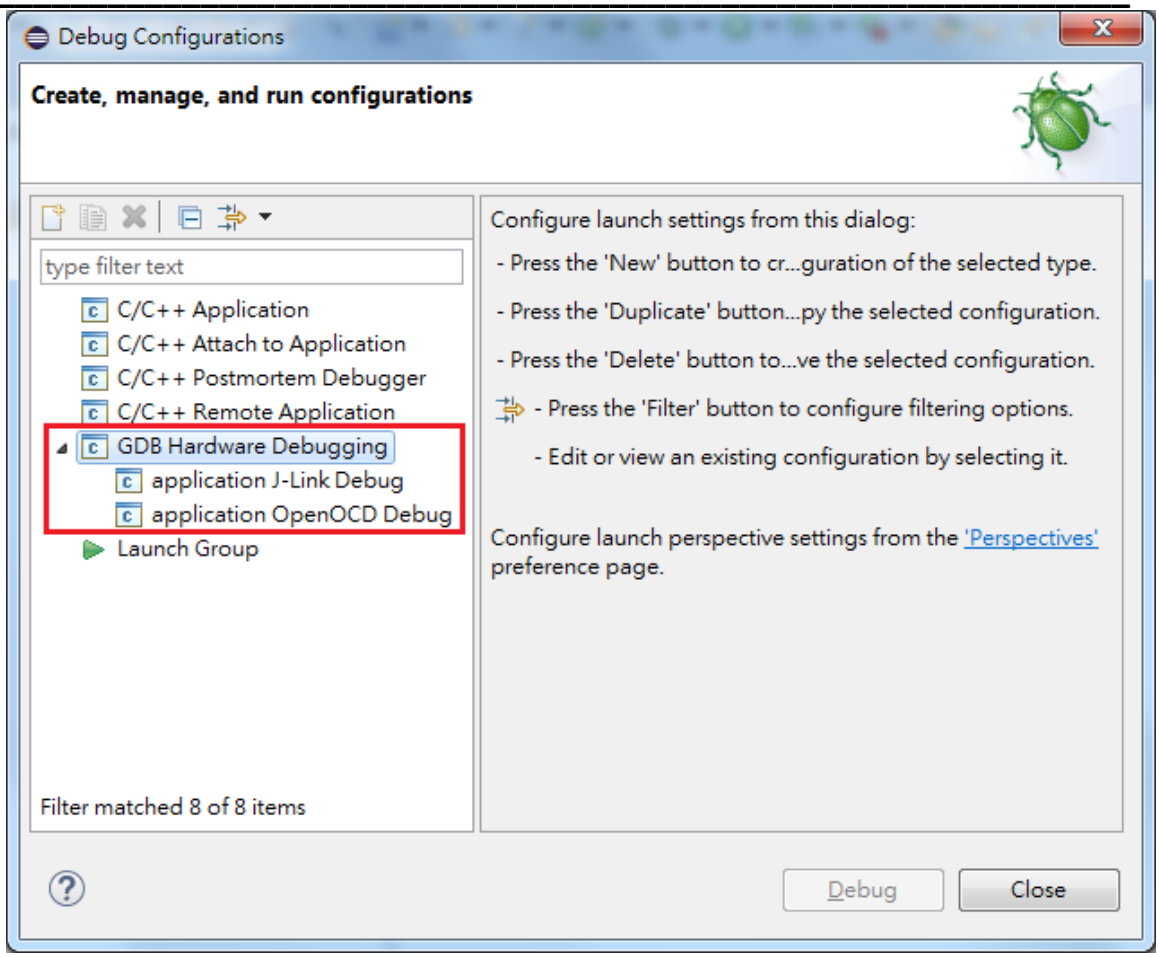
Click “Select All” to select both “application J-Link Debug” and “application OpenOCD Debug” launchers and click “OK”.



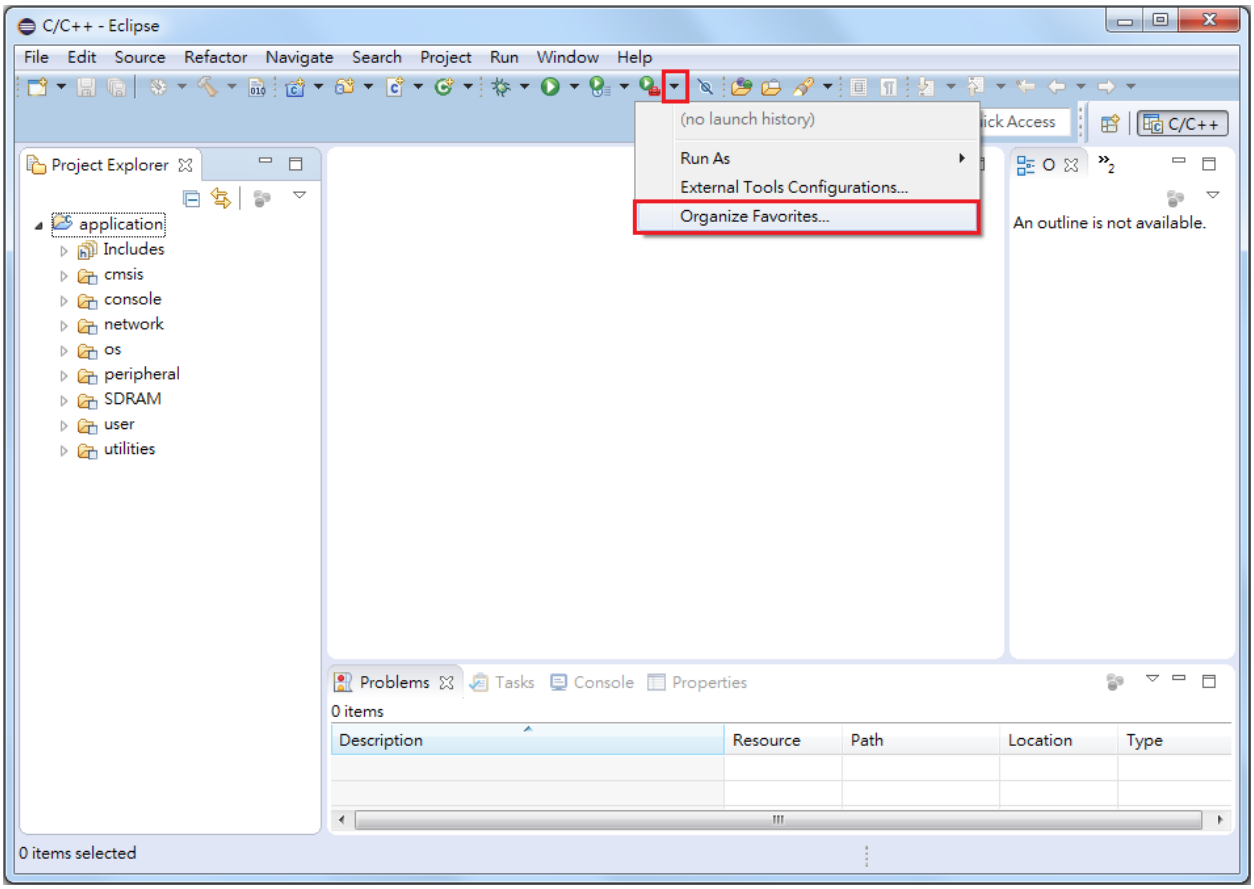
The two debug launchers should be included in “Organize Debug Favorites” window. Click “OK” to finish the setting.



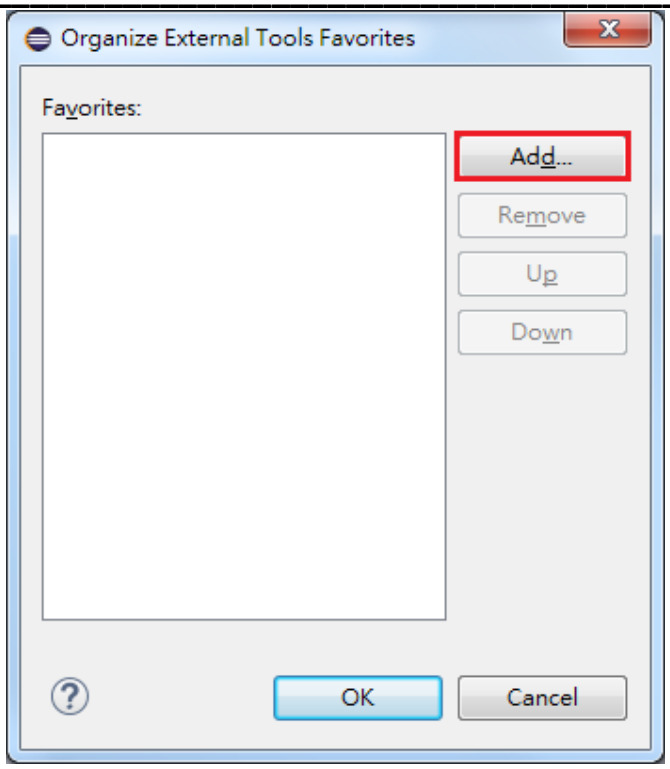
After the debug launchers setting, you can check “Run -> Debug Configurations…” on the top of IDE and you should see two debug configurations which are for J-Link debugger and OpenOCD/CMSIS-DAP debugger under the “GDB Hardware Debugging” entry. These two debug configurations will be introduced later in Sec. 3.3.


Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
application J-Link Debug
application OpenOCD Debug
Launch Group
Filter matched 8 of 8 items
Configure launch settings from this dialog:
- Press the 'New' button to cr...guration of the selected type.
- Press the 'Duplicate' button...py the selected configuration.
- Press the 'Delete' button to...ve the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
Configure launch perspective settings from the 'Perspectives' preference page.
Debug
Close

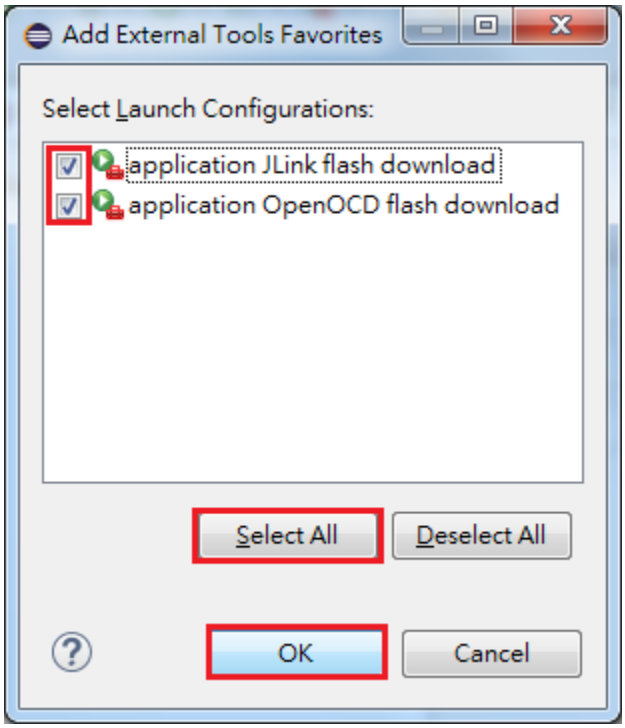
And for external tools launchers, the setting steps are quite similar to debug launchers setting. First, please click “External Tools -> Organize Favorites…”:



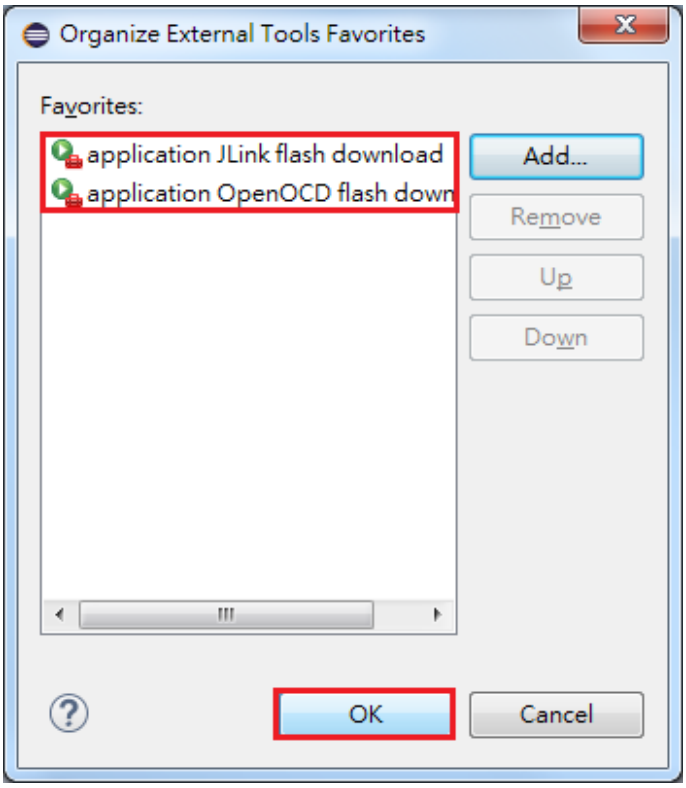
If the popped out window does not contains “application JLink flash download” and “application OpenOCD flash download” launchers like below figure, click the “Add…” button to add them.



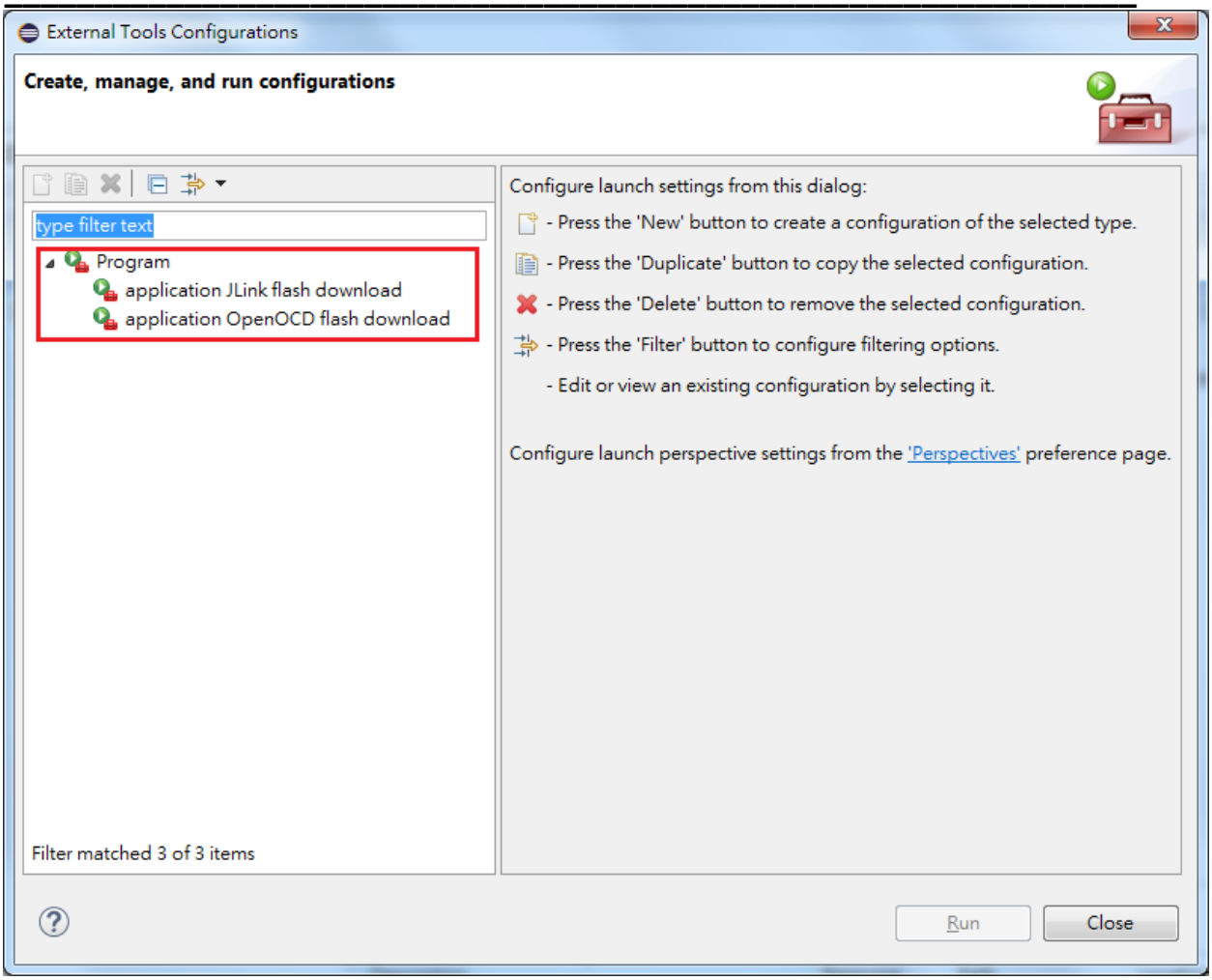
Click “Select All” to select both “application JLink flash download” and “application OpenOCD flash download” launchers and click “OK”.



The two external tools launchers should be included in “Organize External Tools Favorites” window. Click “OK” to finish the setting.



After the external tool launchers setting, you can check “Run -> External Tools -> External Tools Configurations…” on the top of IDE and you should see two external tool configurations which are for J-Link download and OpenOCD/CMSIS-DAP download under the “Program” entry. These two download configurations will be introduced later in Sec.3.2.



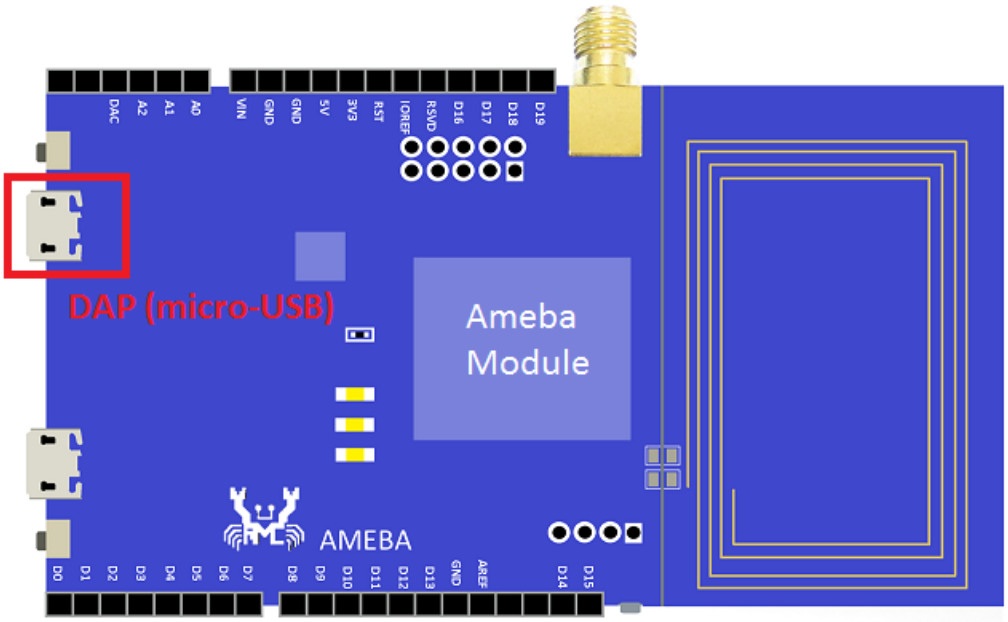
2.3 Debugger setting

Ameba Device Board supports CMSIS-DAP and J-Link for code download and enter debugger mode with GDB. Although we have integrated the download and debug procedure into Eclipse IDE, you still need to start its corresponding GDB server in advance. The settings for these two different debuggers are described below.

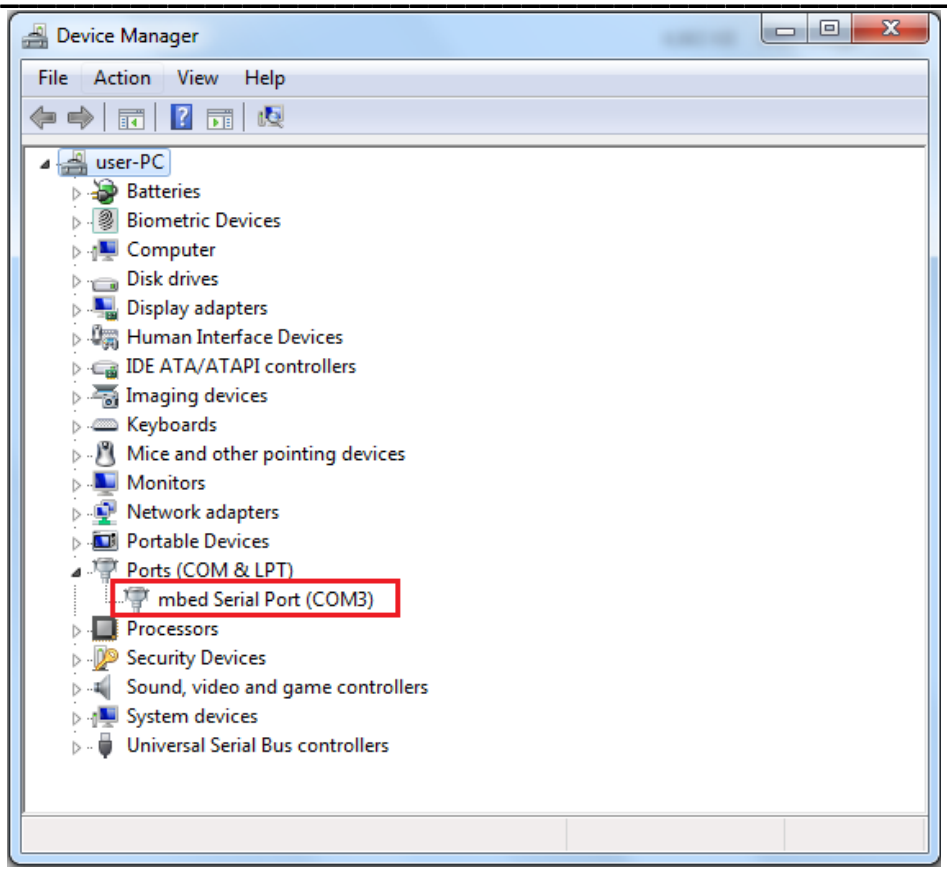
2.3.1 OpenOCD/CMSIS-DAP

Ameba Device Board supports CMSIS-DAP debugger. We can use OpenOCD/CMSIS-DAP to download the software and enter debug mode under Eclipse + GDB environment. It requires installing “serial to USB driver” at first. Serial to USB driver can be found in tools/serial_to_usb/mbedWinSerial_16466.zip.

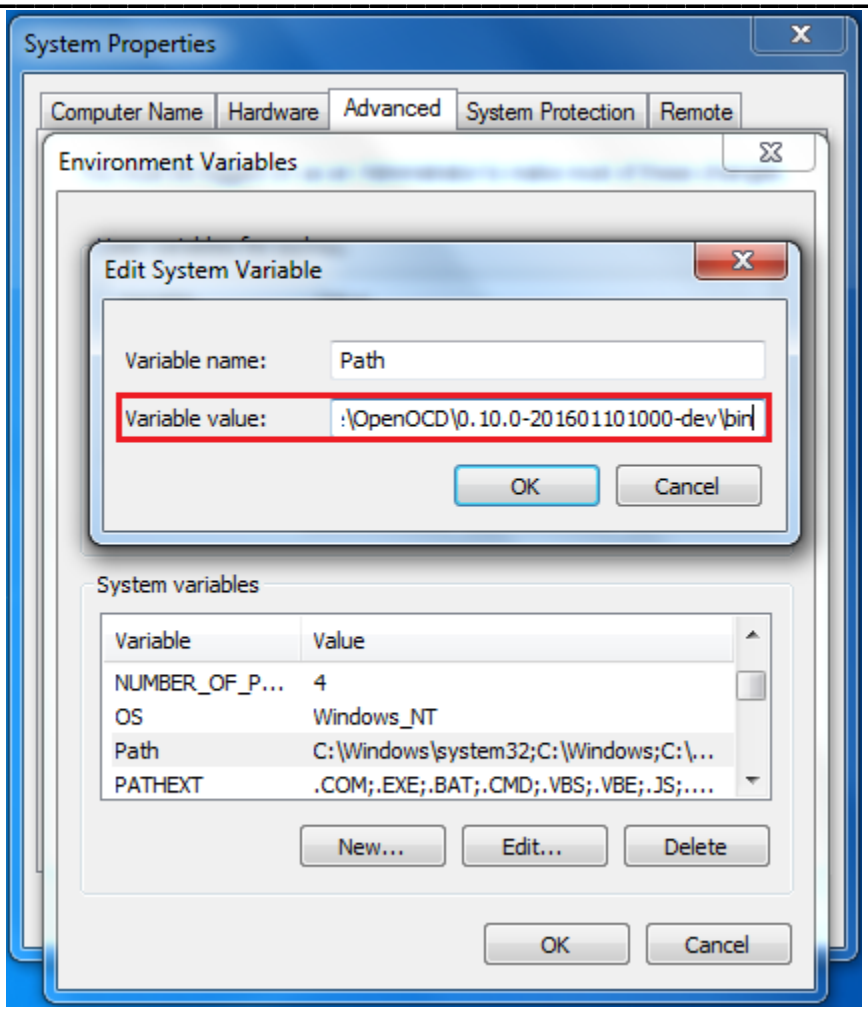
Connect board to the PC with micro-USB cable:



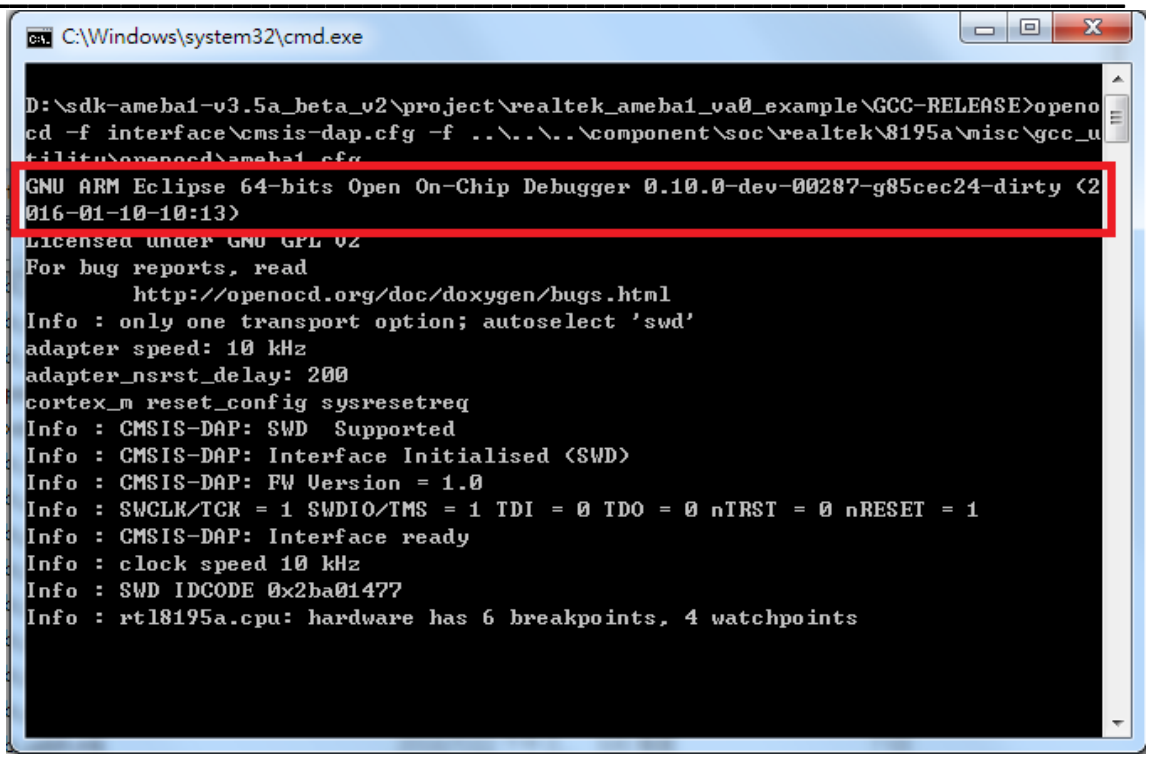
If Serial to USB driver has been installed and the board is connected to PC, there should be mbed Serial Port shown in Device Manager.



It also requires installing OpenOCD on your platform. Please check http://openocd.org to get the binary package (https://github.com/gnuarmeclipse/openocd/releases). Then install OpenOCD and add the bin files to Environment Variables Path (Control Panel -> System and Security -> System -> Advanced System Settings -> Advanced tab -> Environment Variables -> Path).

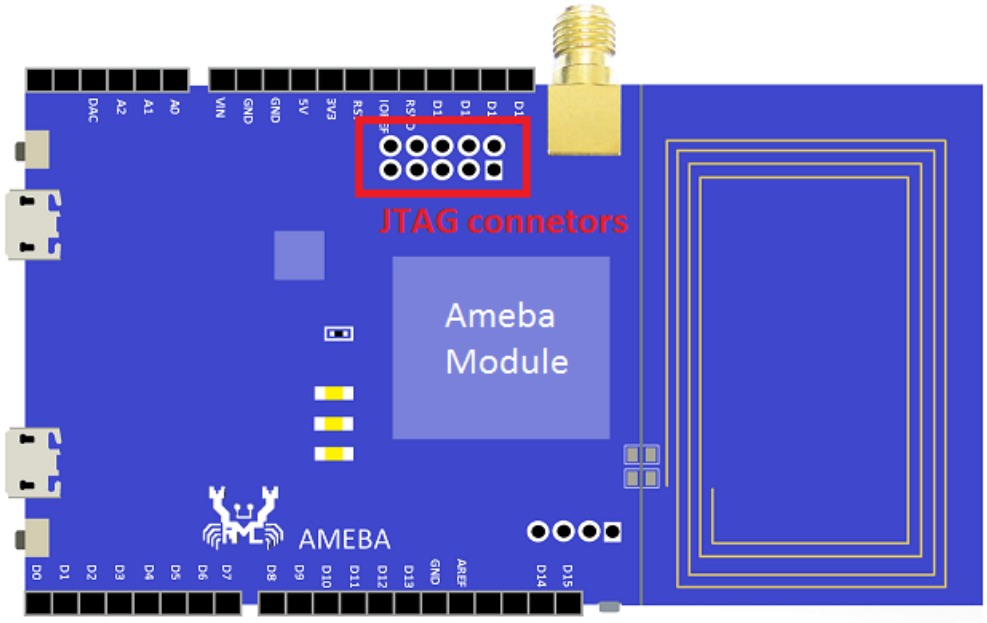


If OpenOCD has been installed correctly, execute project\realtek_ameba1_va0_example\GCC-RELEASE\run_openocd.bat to start GDB server and you should see some messages like below figure. This window should **NOT** be closed if you want to download software or enter debugger.



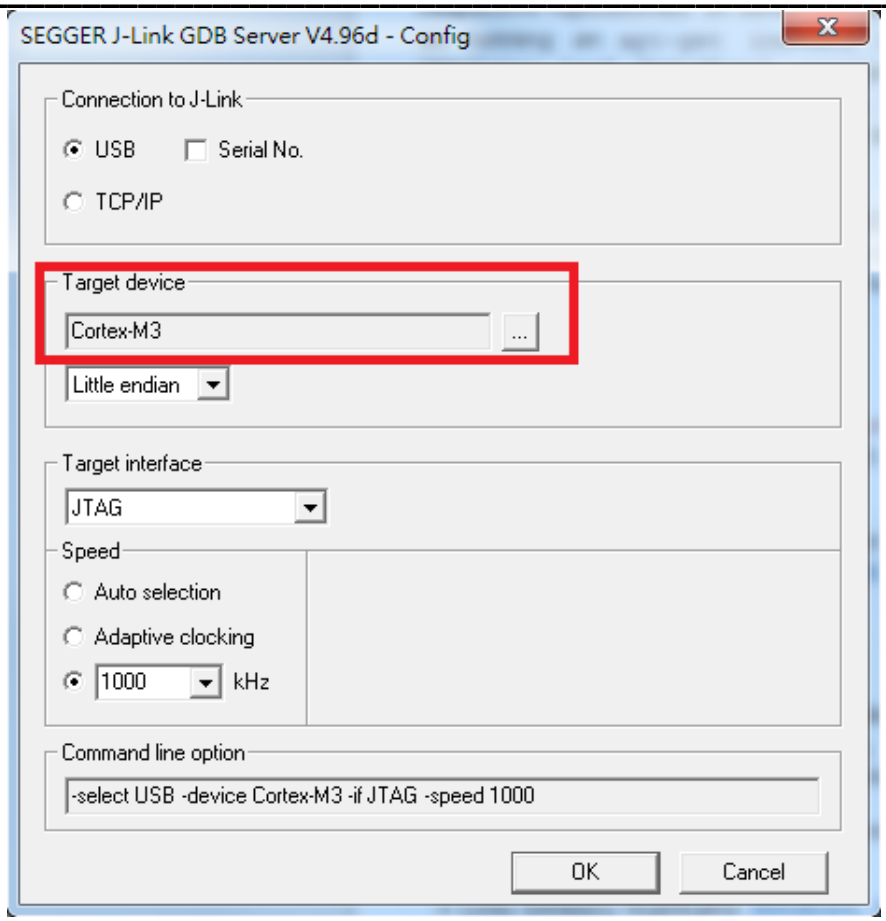
2.3.2 J-Link

Ameba Device Board also supports J-Link debugger. To use J-Link debugger we need to do some hardware configuration. Please weld JTAG connectors to HDK board and connect with pitch 2.54mm 2x5pins connector. The JTAG pin definitions are listed on the bottom side. And it is recommended to weld the connector on the bottom side. After finish this configuration, please connect it to PC side. Note that if you are using Virtual Machine as your platform, please make sure the USB connection setting between VM host and client is correct so that the VM client can detect the device.



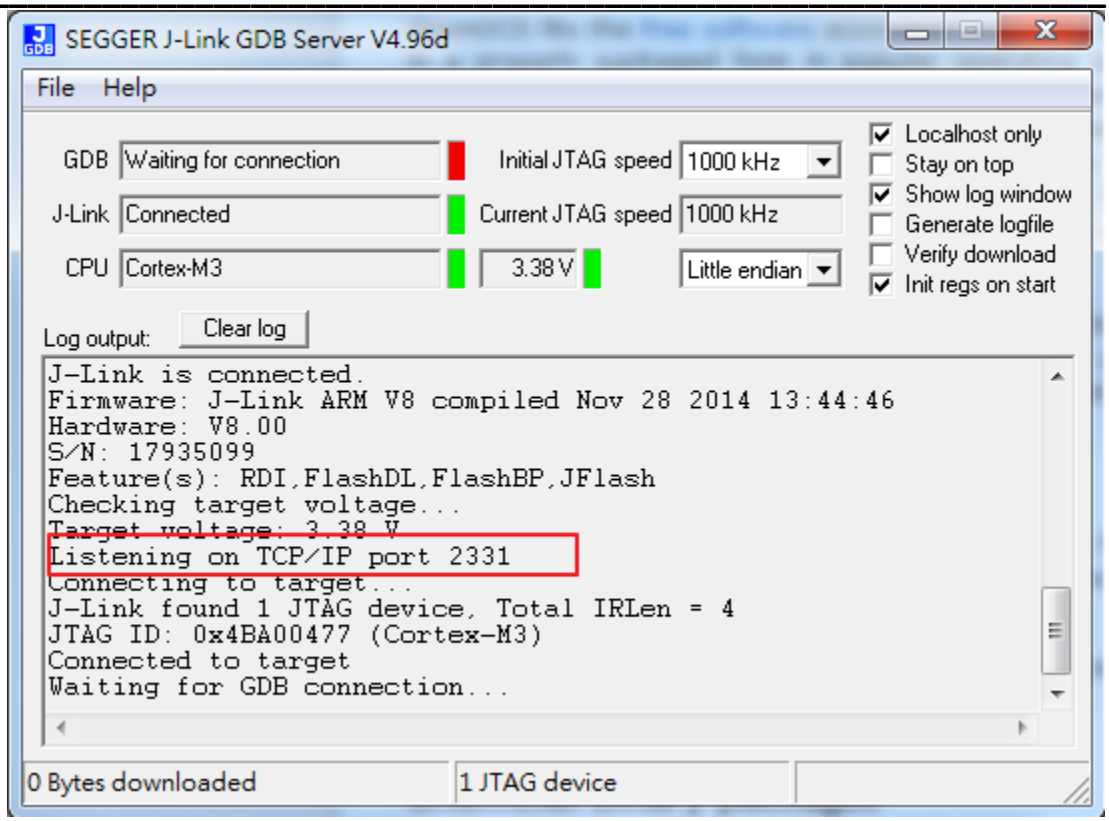
2.54mm 2x5pins connector (or use Dupont Line)

Besides the hardware configuration, it also requires installing J-Link GDB server. For Windows, please check http://www.segger.com and download “J-Link Software and Documentation Pack” (https://www.segger.com/downloads/jlink). After the installation of the software pack, you should see a tool named “J-Link GDB Server”. Execute the J-Link GDB Server tool and choose the target device to Cortex-M3 to start GDB server:



The started J-Link GDB server should looks like below figure. And this window should **NOT** be closed if you want to download software or enter GDB debugger mode.

In the log console, make sure the TCP/IP port is **2331** which should be the same as default setting in component\soc\realtek\8195a\misc\gcc_utility\eclipse\rtl_gdb_flash_write.txt and the J-Link debug configuration so that Eclipse can connect to the GDB server successfully.

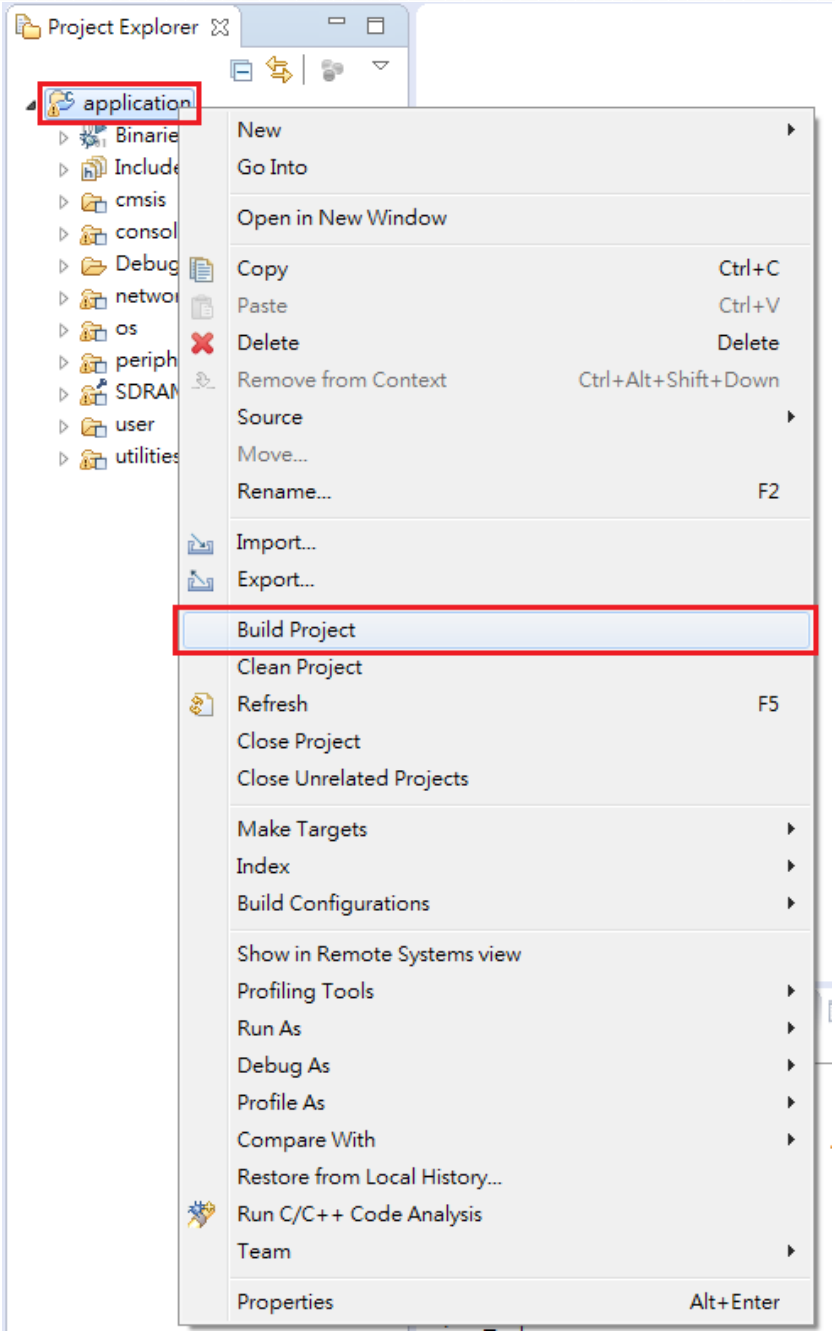


3 How to build, download and enter debug mode

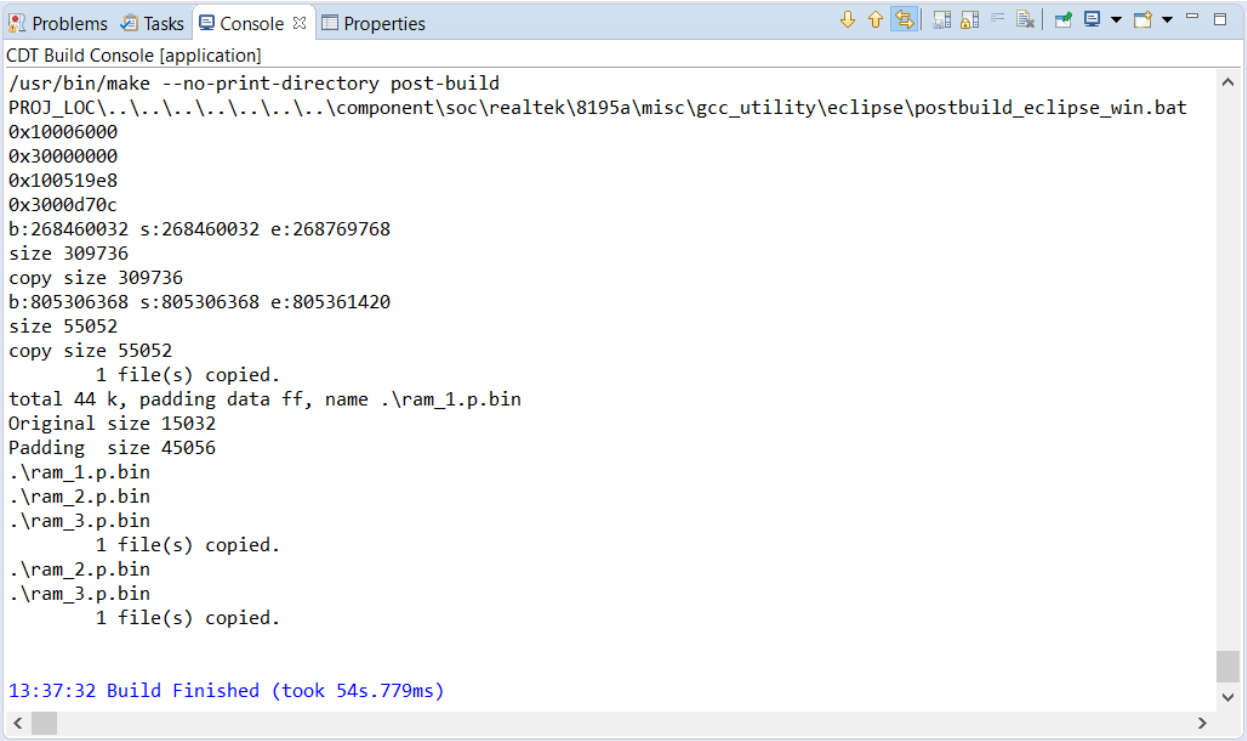
In this section, we illustrate how to build, download, and enter debug mode. Note that in Sec. 3.2 the code is downloaded to flash but in Sec. 3.3 the code is written into RAM before we enter debug mode.

3.1 Build code

To build the SDK, please right click the “application” project in the “Project Explorer” view and click “Build Project”.



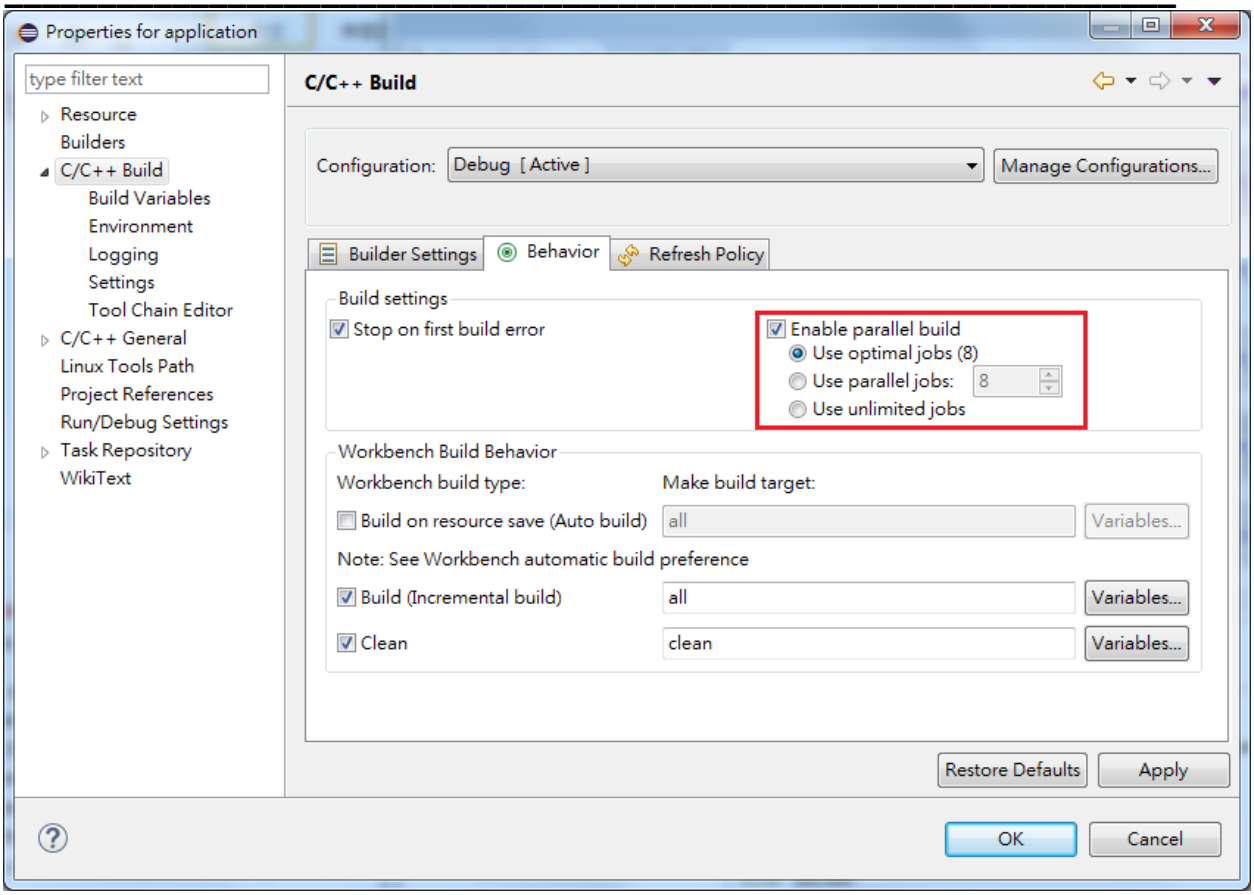
If the application built successfully, the console should display some message like following figure:



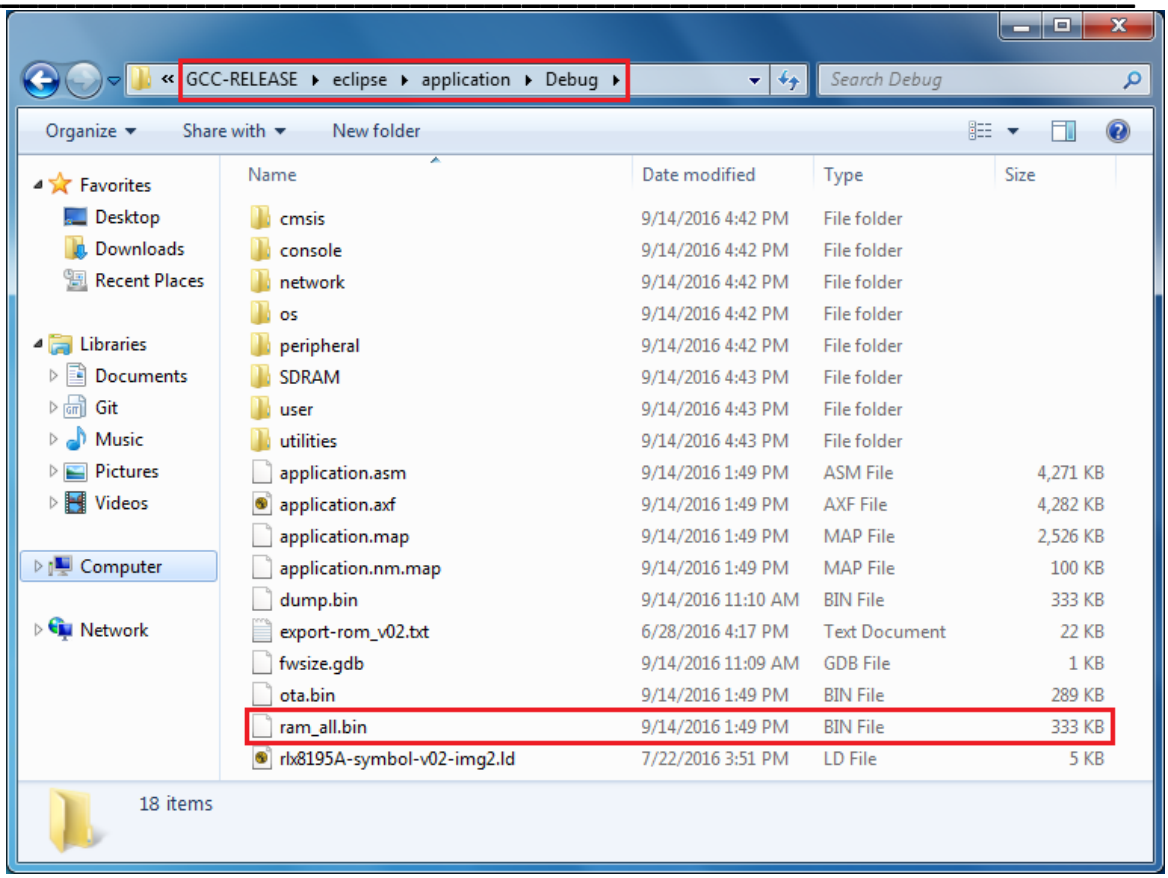
Note. If you get "**Program "make" not found in PATH**" error message in the console means that somehow the Cygwin path is not set correctly. Please refer to Sec. 4.2 to troubleshoot it. And if you get some header file not found error, please refer to Sec. 4.3 to fix it.

And if you want to speed up the compile procedure, you can enable the parallel build option to make it not compile one file after another, but instead spawn a set of parallel builds. But note that since we would un-compress the gcc tool chain (tools\arm-none-eabi-gcc\) during pre-build step, enabling parallel build might cause the compile process has been started but the tool chain is not ready yet and cause errors. Fortunately, the un-compress tool chain procedure only needs to be executed one time. So we suggest that for the first project build, do **NOT** enable parallel build. After first successful build, you can enable the parallel build options as default.

To enable parallel build, please right-click "application" project and select "Properties". Choose "C/C++ Build" -> "Behavior" tag and click "Enable parallel build" on the right side like following figure. After apply this configuration, re-compile the project and you should find that it would cost less time.

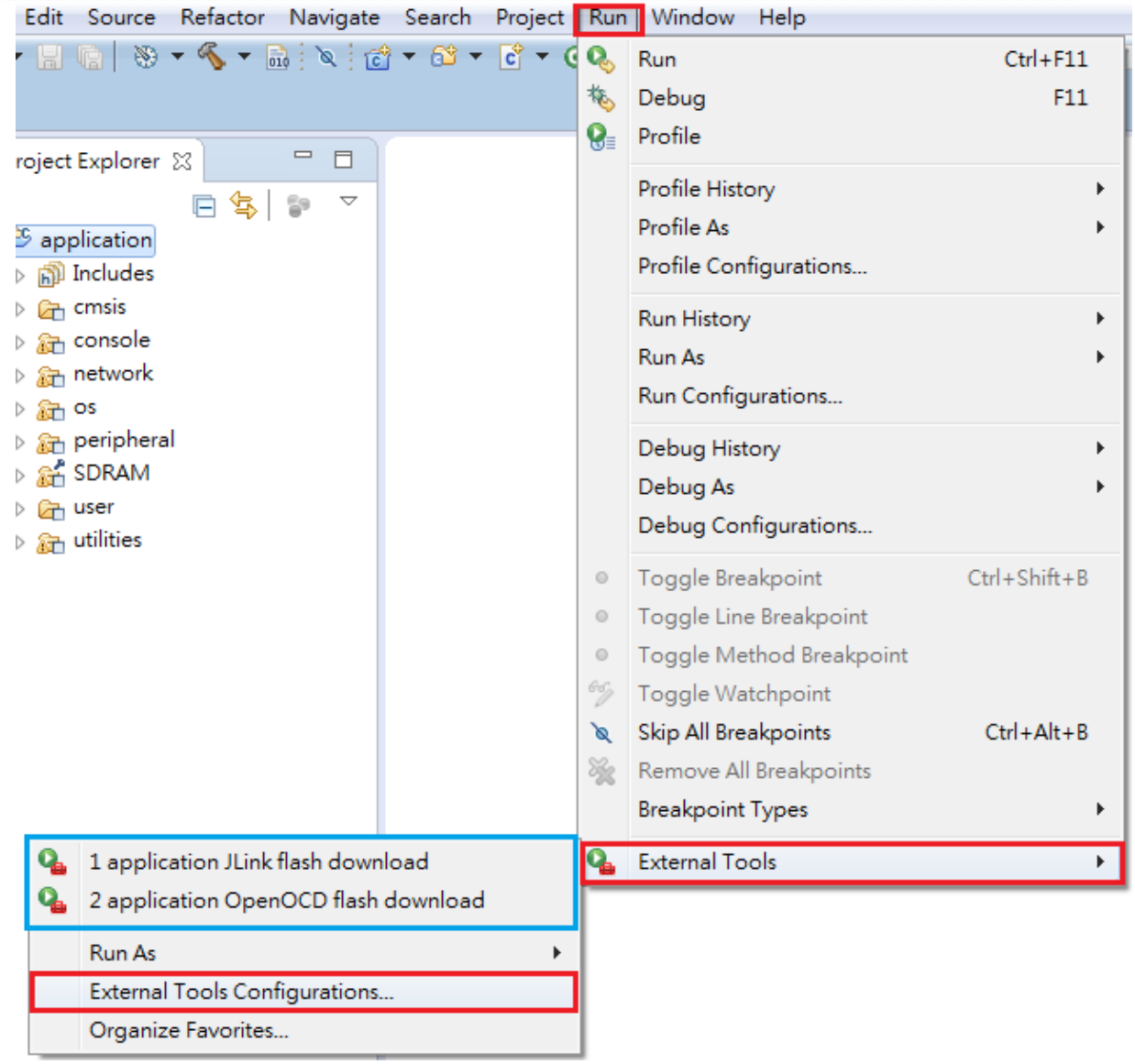


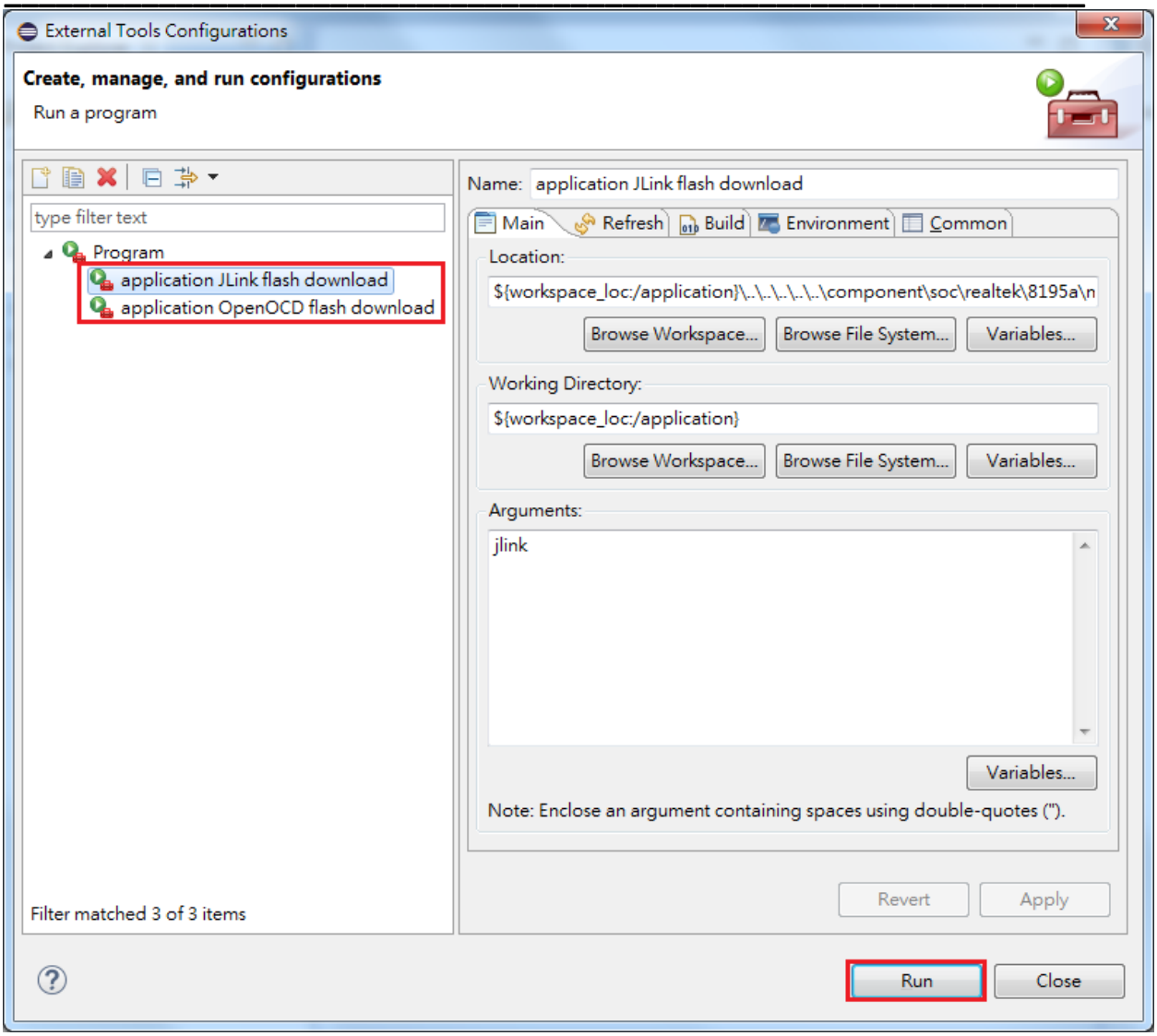
After successfully build, there should be a directory named “Debug” created under project\realtek_ameba1_va0_example\GCC-RELEASE\eclipse\application directory. The image file (ram_all.bin) is located in Debug\:



3.2 Download code to flash

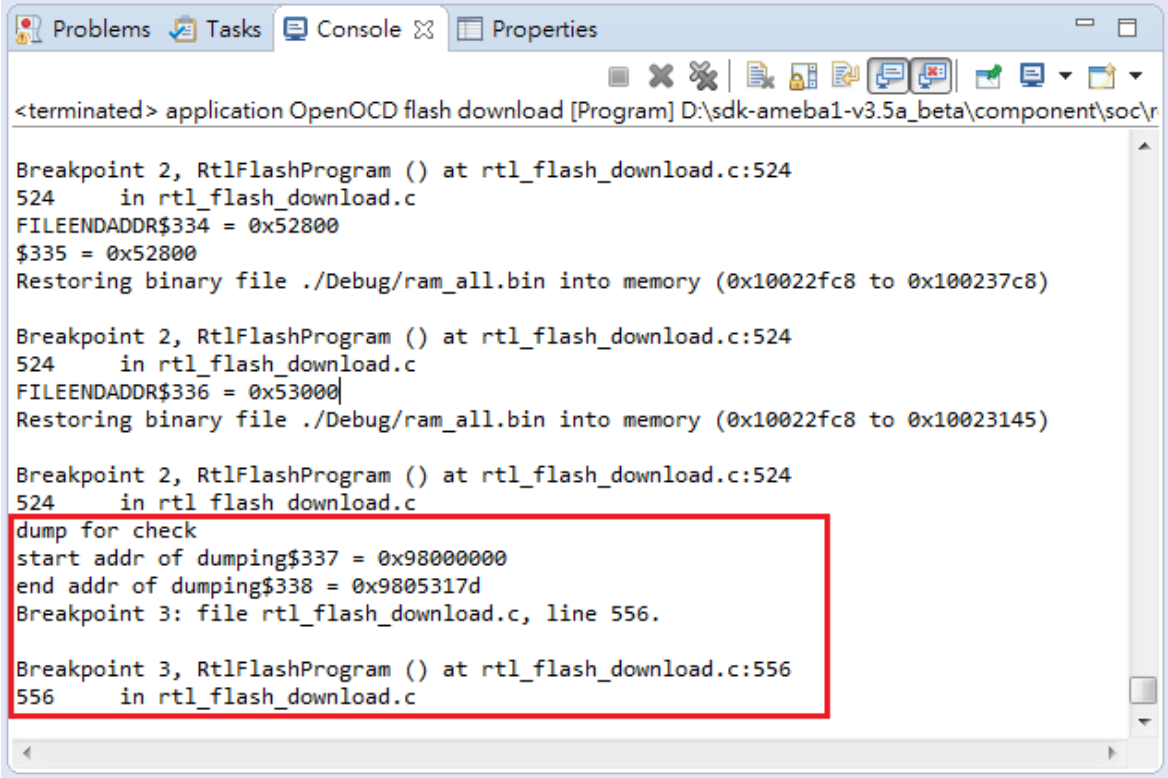
To download software into Ameba Device Board, please make sure steps mentioned in Sec. 2.3 are done and the GDB server is started whether you are using OpenOCD/CMSIS-DAP or J-Link as GDB server. Click “Run -> External Tools -> External Tools Configurations…” and then choose “application JLink flash download / application OpenOCD flash download” to start the download procedure. Or you can just click “Run -> External Tools -> application JLink flash download / application OpenOCD flash download” to start it.


Edit Source Refactor Navigate Search Project Run Window Help
roject Explorer
application
Includes
cmsis
console
network
os
peripheral
SDRAM
user
utilities
Run Ctrl+F11
Debug F11
Profile
Profile History
Profile As
Profile Configurations...
Run History
Run As
Run Configurations...
Debug History
Debug As
Debug Configurations...
Toggle Breakpoint Ctrl+Shift+B
Toggle Line Breakpoint
Toggle Method Breakpoint
Toggle Watchpoint
Skip All Breakpoints Ctrl+Alt+B
Remove All Breakpoints
Breakpoint Types
External Tools
1 application JLink flash download
2 application OpenOCD flash download
Run As
External Tools Configurations...
Organize Favorites...



The download procedure would take a few seconds to finish. And it might also build the project before downloading to flash depends on the project status. After successful download, you might see some message like below figure in the “Console” view.

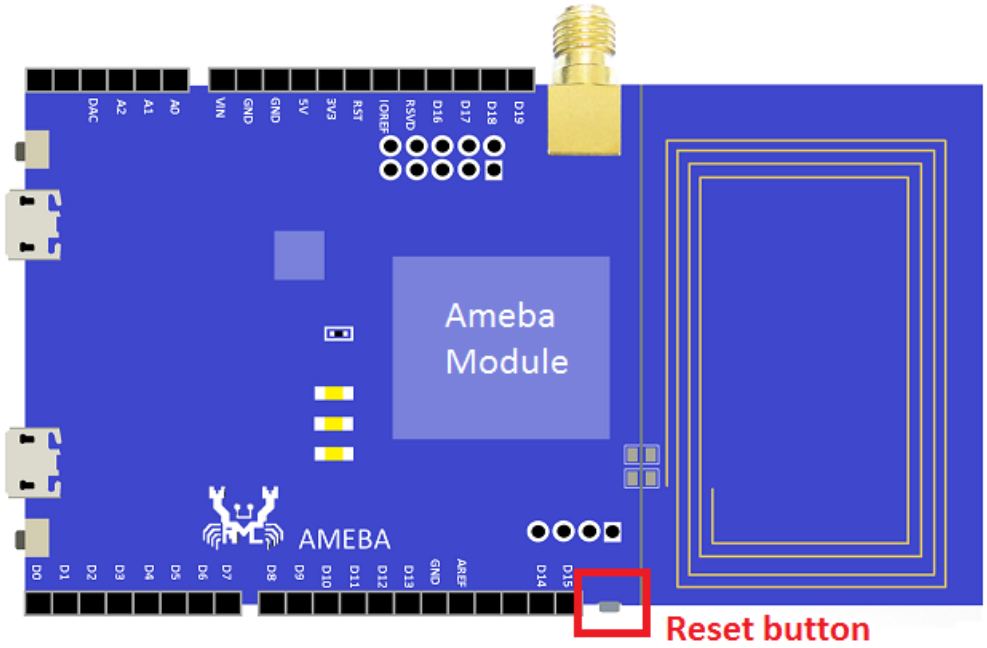
Note. If the download procedure hangs for a long time, you can check Sec. 4.4 to troubleshoot the issue by updating newest DAP firmware.



And if you have started the Ameba serial console during the procedure you will see it also display the flash download message:

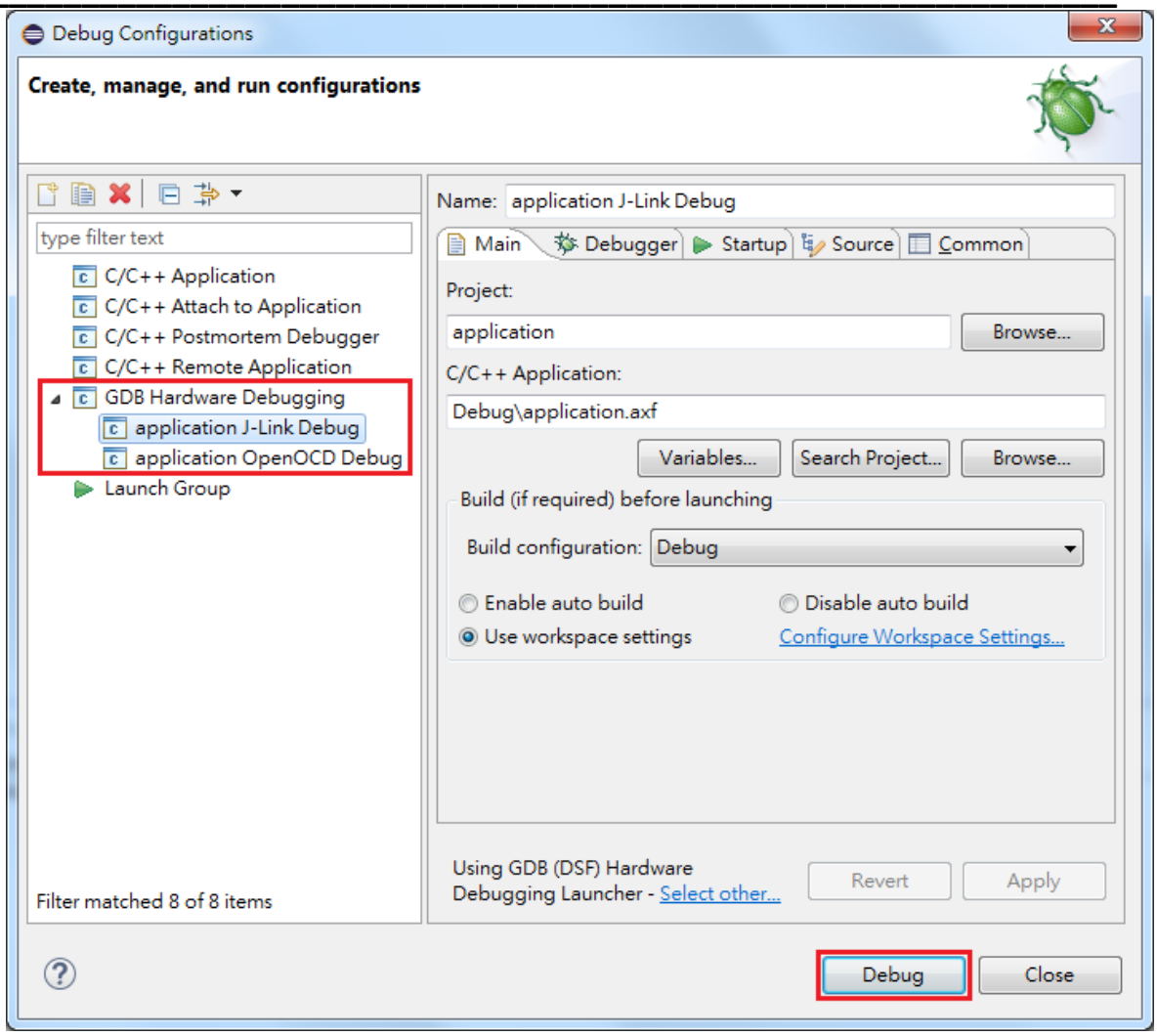
```
==========================================================
Flash Downloader Build Time: # 2016/07/06-00:04:09
==========================================================
EraseMode : 1
FirmwareSize : 340349
Skip calibration
Flash erace done
Flash download start
Flash download done
```

After the flash download finished, press the Reset button on device board and you should see that the device now is booted with new image.

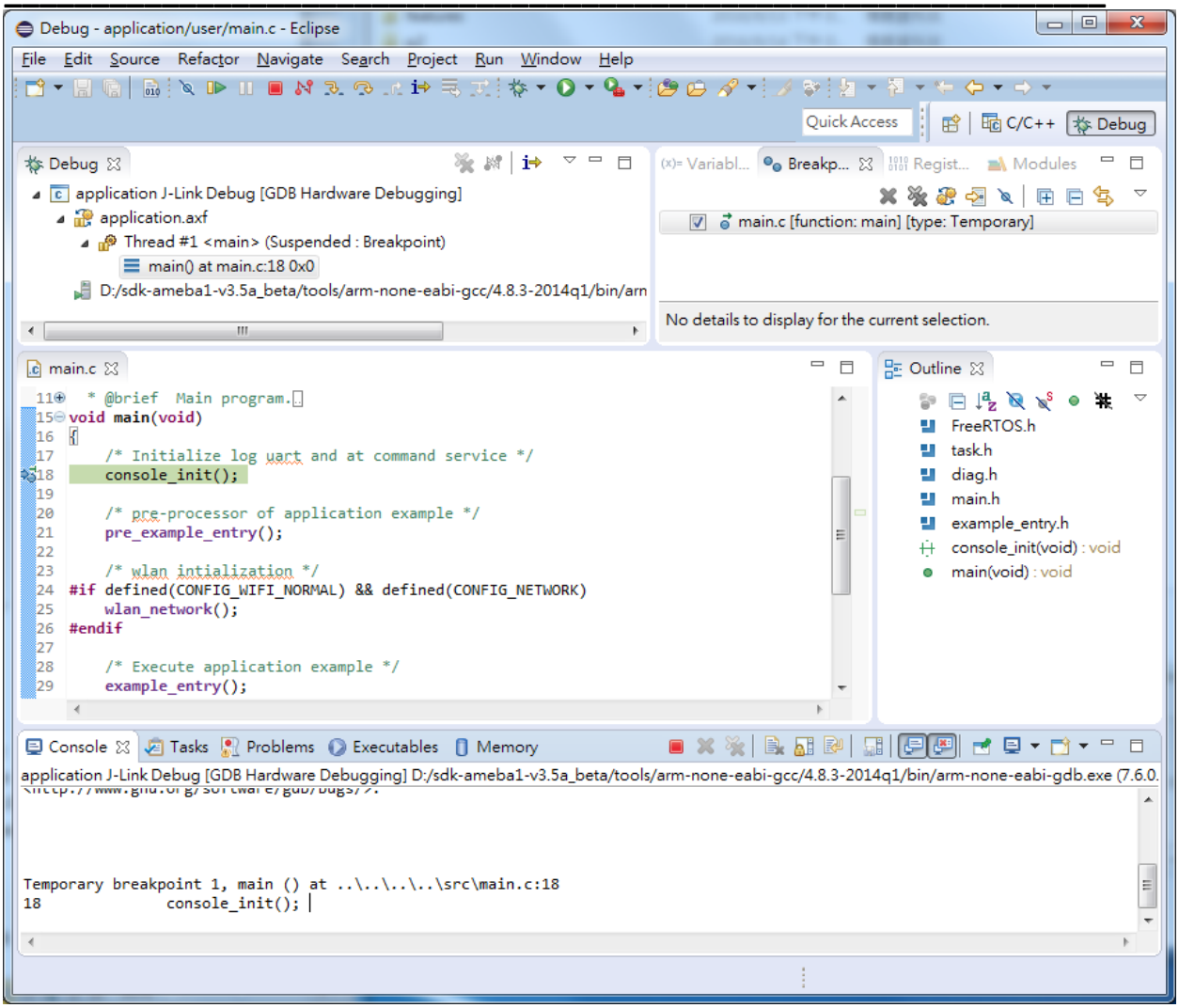


3.3 Download and debug in RAM

This section describes how to debug application project under Eclipse IDE. For the entire debug procedure, the code would be downloaded into RAM before entering debug mode. As we described in Sec. 2.2, we use “GDB Hardware debugging” plug-in to enter GDB debugger mode under Eclipse environment. Please make sure the GDB server whether is OpenOCD or J-Link has been started as we described in Sec. 2.3 before entering debug mode. To start the debug procedure, click “Run -> Debug Configurations…” on the top of Eclipse IDE and then choose “application J-Link Debug / application OpenOCD Debug” under “GDB Hardware Debugging” entry in the left side of popped out window. Click “Debug” button and the debug procedure would start.



After a few seconds, Eclipse would switch to “Debug” perspective as the following figure. Note that if you are using OpenOCD as GDB server, the procedure might take longer time than J-Link. As you can see in the figure, the program should be suspended at *console_init()* statement since we have inserted a breakpoint at *main* function by default. Note that if you find that the program counter would jump to incorrect place when you click “Step” button during the debug procedure, please refer to Sec. 4.5 to troubleshoot it.



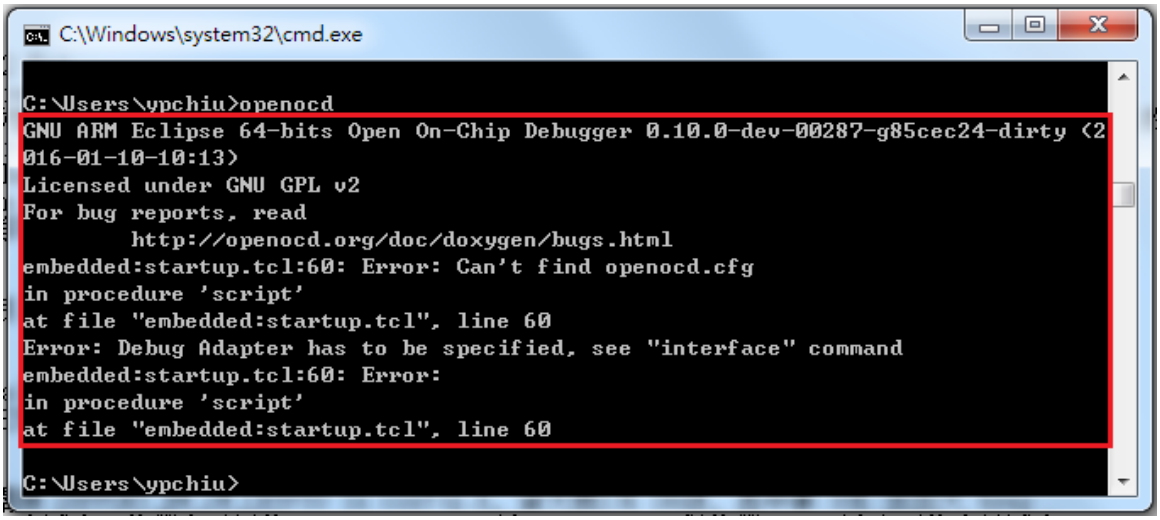
4 Troubleshooting

4.1 Unable to execute run_openocd.bat normally on Windows

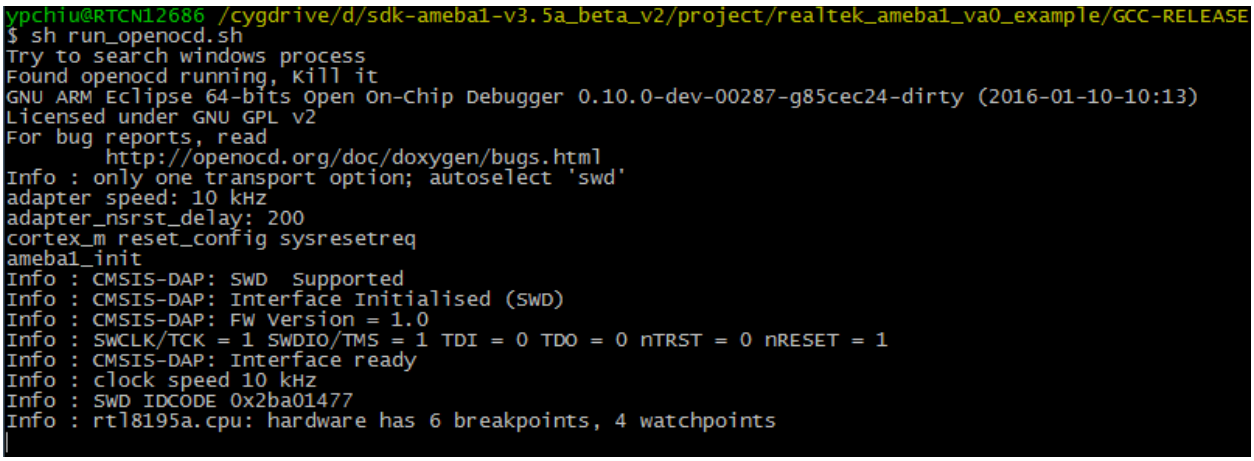
On Windows platform, if you cannot execute run_openocd.bat normally and the pop out window always crash, the reason might be OpenOCD has not been correctly installed or the connection between PC and Ameba has some problem.

To check whether OpenOCD has been correctly installed, you can simply type “openocd” on cmd window. You should see the debug message like following figure if the OpenOCD has been

installed right. If you see message like “openocd is not recognized as an internal or external command…” instead of above message, it means that OpenOCD did not installed correctly. In this case, please make sure the steps in Sec. 2.3.1 are all done correctly especially the environment variable path configuration part.

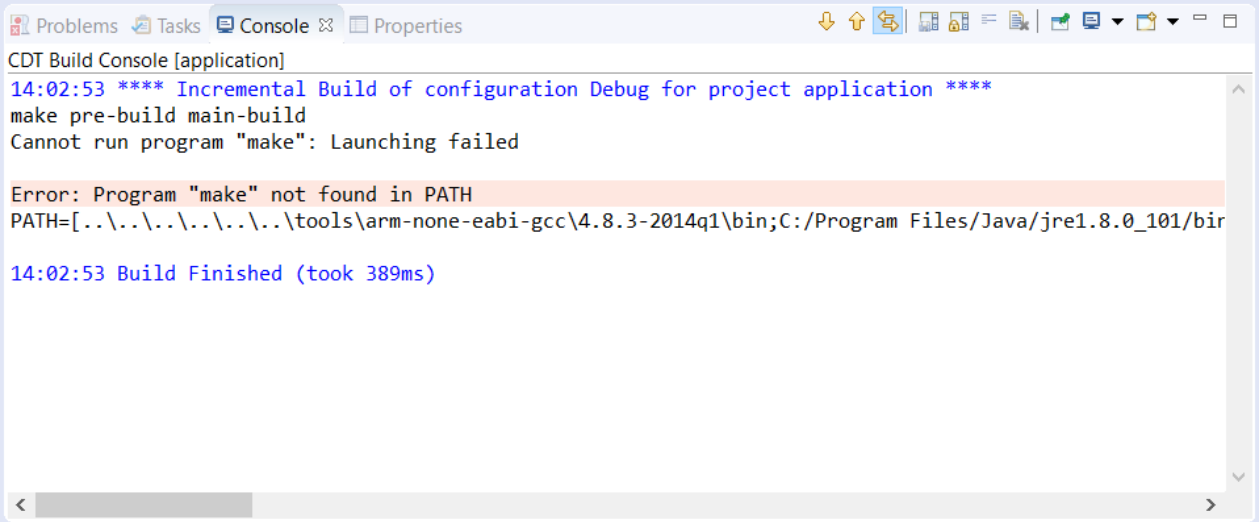


If OpenOCD has been installed correctly but the run_openocd.bat still cannot run normally, you can try to re-plug the connection between PC and Ameba board. You also can try to execute run_openocd.sh rather than run_openocd.bat if you have some Unix-like environment tools, e.g. Cygwin. Below figure illustrates using Cygwin to execute run_openocd.sh script.



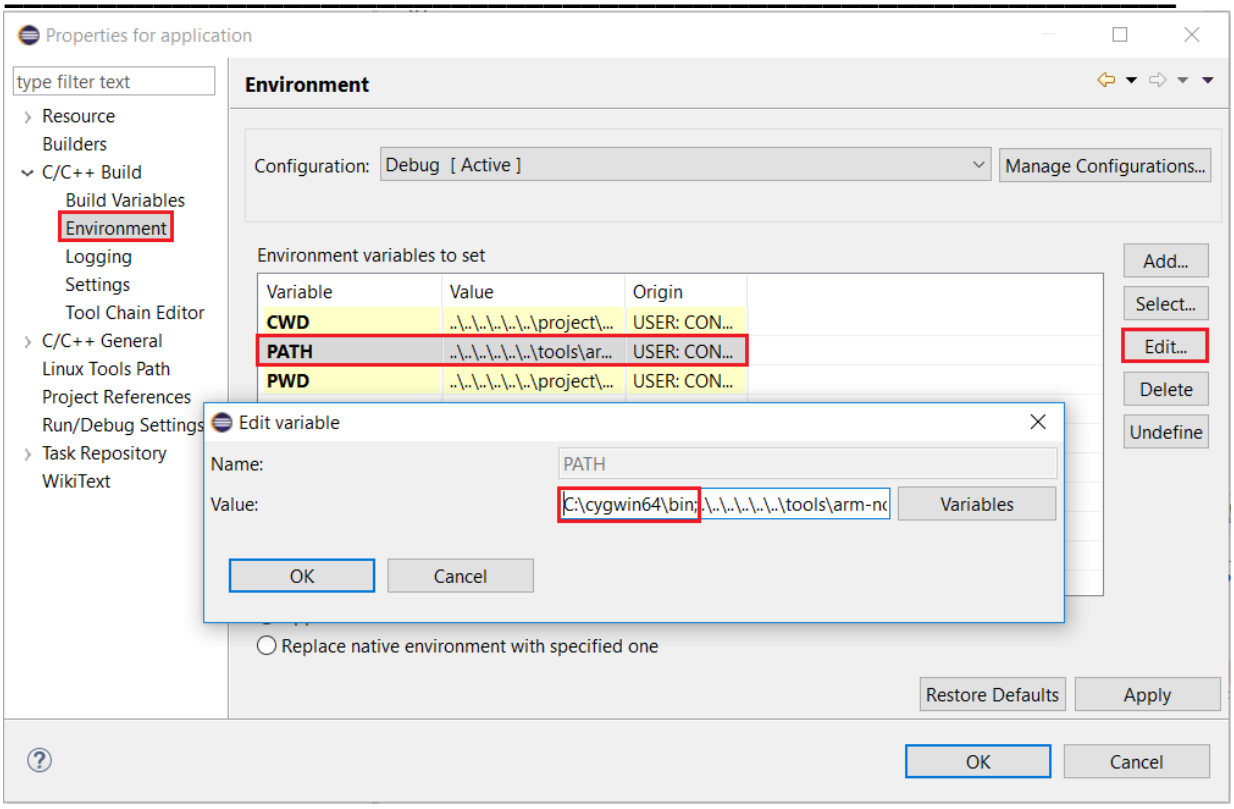
4.2 Make not found in PATH error during project building

As we described in Sec. 3.1, if you encounter “Error: Program make not found in PATH” for building project means that the Cygwin path is not set correctly for Eclipse.



Please make sure the path of Cygwin bin file directory (e.g. C:\cygwin64\bin) is appended to Environment Variable Path. After that you need to restart the Eclipse IDE and build the project again.

If the error still existed, you also can modify the Eclipse project environment variable directly. Please right-click “application” project and select “Properties”. In the popped out window, select “C/C++ Build” -> “Environment” and edit the “PATH” variable. For example, we append the bin files path “C:\cygwin64\bin\;” in front of the variable value like below figure.



4.3 Build fail due to relative path header file not found

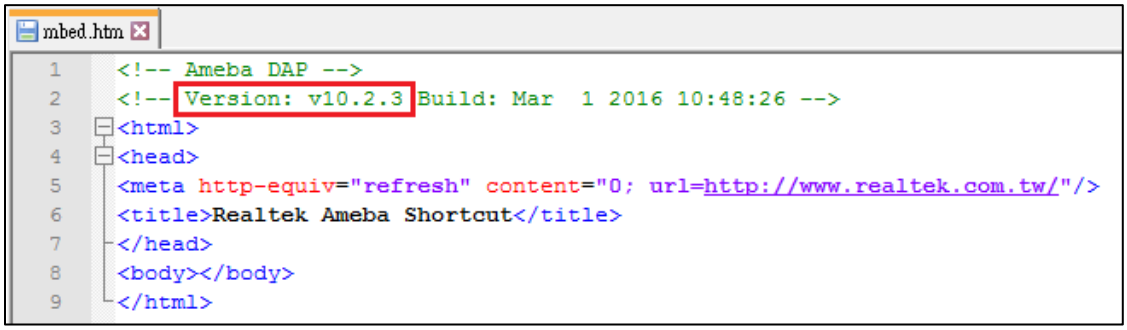
If the build project procedure terminates due the some header file not found and the header file is included with relative path, it may be caused by the long project path. The length of include path may exceed the maximum limit. For example, if you encounter a fatal error like: “fatal error: ../Source/portable/GCC/ARM_CM3/portmacro.h: No such file or directory”. You can try to move the SDK project to other directory with shorter path and import the project again. In this case, we move the project to D:\sdk\ and the error would disappear.

```
Building file: D:/work/20161003_eclipse_make_builder/test/test_level_2/sdk-ameba1-v3.5a_beta_v4/compo
Invoking: Cross GCC Compiler
arm-none-eabi-gcc -c -mcpu=cortex-m3 -mthumb -std=gnu99 -DGCC_ARMCM3 -DM3 -DARDUINO_SDK -DCONFIG_PLAT
In file included from ..\..\..\..\..\..\component\os\freertos\freertos_v8.1.2\Source\include/FreeRTOS
                 from D:/work/20161003_eclipse_make_builder/test/test_level_2/sdk-ameba1-v3.5a_beta_v
..\..\..\..\..\..\component\os\freertos\freertos_v8.1.2\Source\include/portable.h:196:58: fatal error
 #include "../../Source/portable/GCC/ARM_CM3/portmacro.h"
                                                          ^
compilation terminated.
make: *** [utilities/example/subdir.mk:95: utilities/example/example_bcast.o] Error 1

14:48:53 Build Finished (took 1s.670ms)
```

4.4 Download procedure hang for a long time

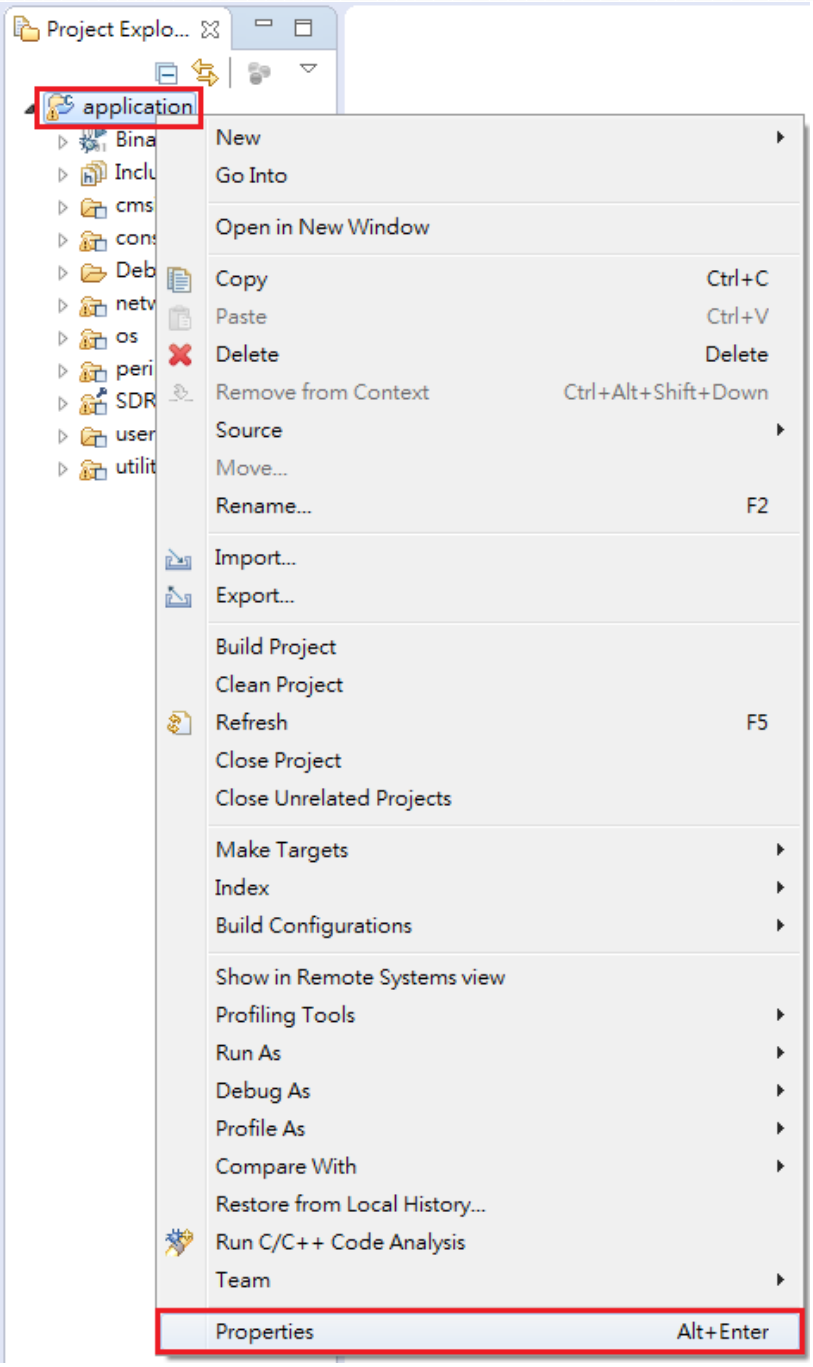
In Sec. 3.2, if the download procedure hang for a long time it might due to the DAP firmware problem on device board. The version of DAP firmware should be greater than (or equal to) v10.2.3 to make download procedure work. To check the version information, you can use text editor to check to content of mbed.htm in MBED drive.



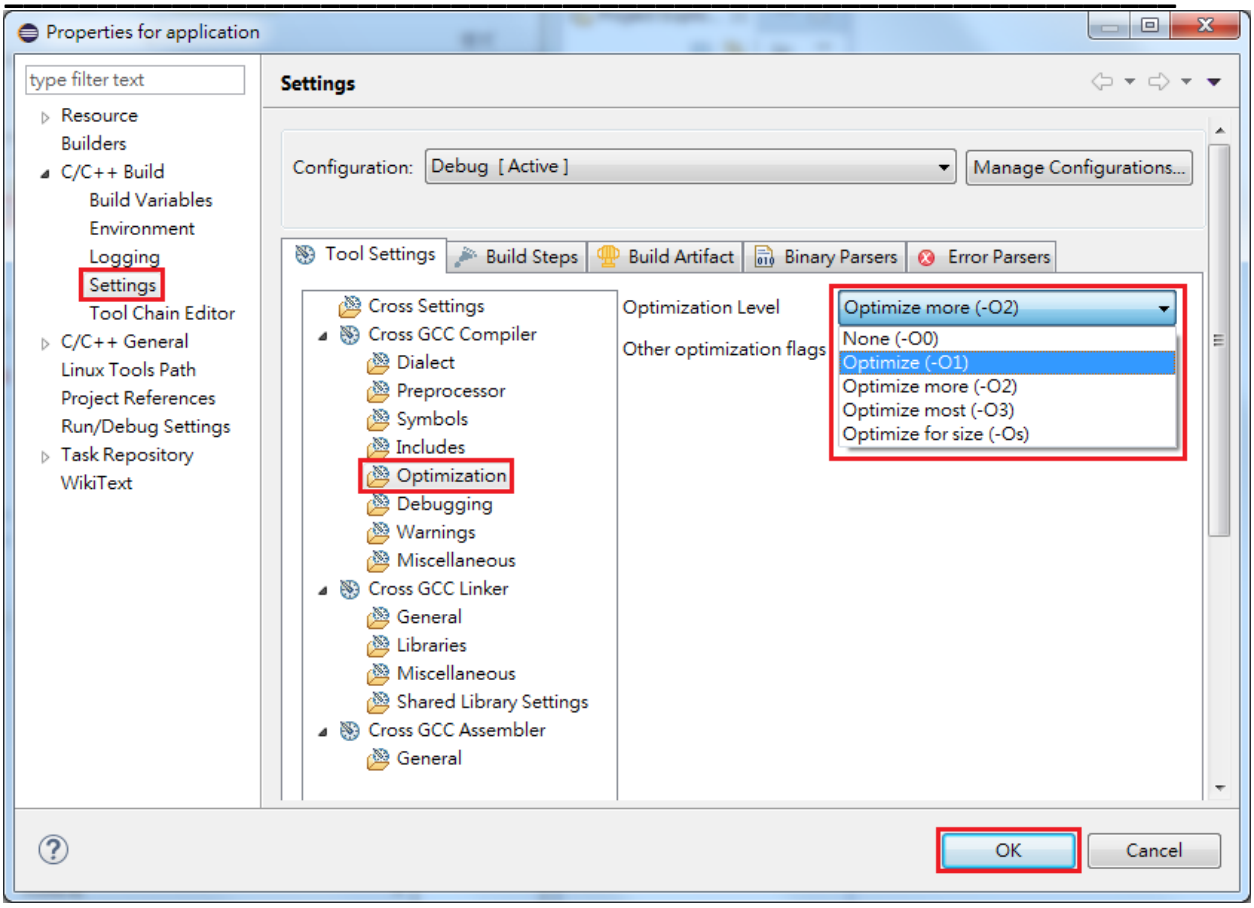
If you find that the DAP firmware version is out of date, you can refer http://www.amebaiot.com/en/change-dap-firmware/ to update the DAP firmware on device board.

4.5 Next step not correct in debug mode

In Sec. 3.3 we have described how to enter debug mode under Eclipse + GDB environment. If you find that the step to step tracing feature (“Step Into” / “Step Over”) sometimes would go to wrong program location it might be due to the compiler optimization. To prevent it, you can try to set the compiler optimization level to a lower value. To do this, please right click “application” project in “Project Explorer” view and select “Properties”.



Please select “C/C++ Build -> Settings” at the left side of popped out window and then chose “Tool Settings -> Cross GCC Compiler -> Optimization” at the right side. As in the following figure, the default optimization level is set to “Optimize more (-O2)” and you can change the value to “Optimize (-O1)” from the dropdown button. Click “OK” and re-enter the debug mode the next step feature should act correctly.


Properties for application
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Linux Tools Path
Project References
Run/Debug Settings
Task Repository
WikiText
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
Cross Settings
Cross GCC Compiler
Dialect
Preprocessor
Symbols
Includes
Optimization
Debugging
Warnings
Miscellaneous
Cross GCC Linker
General
Libraries
Miscellaneous
Shared Library Settings
Cross GCC Assembler
General
Optimization Level
Other optimization flags
Optimize more (-O2)
None (-O0)
Optimize (-O1)
Optimize more (-O2)
Optimize most (-O3)
Optimize for size (-Os)
OK
Cancel