Build Environment Setup - Eclipse

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

Table of Contents

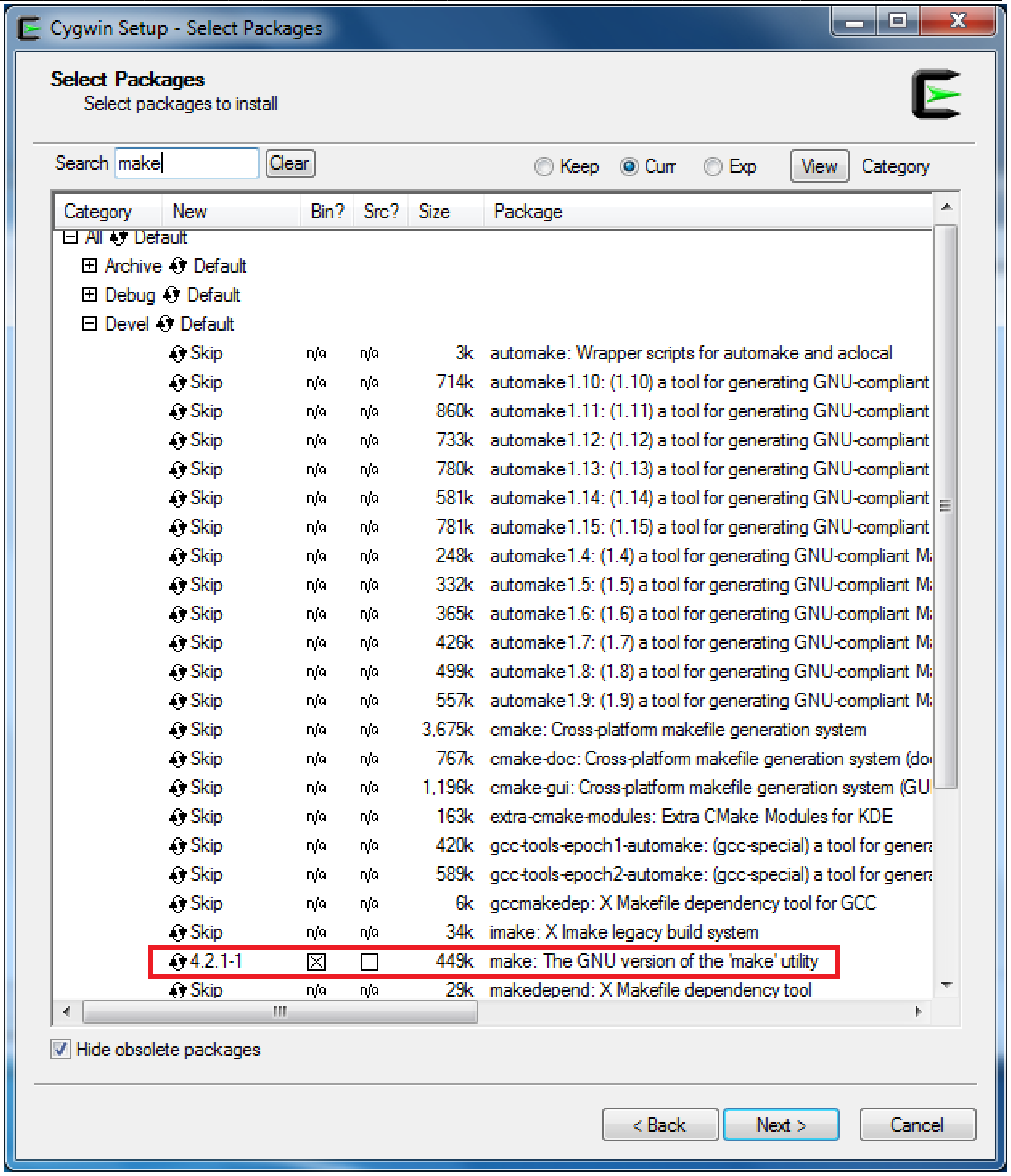
1 Introduction

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:


Cygwin Setup - Select Packages

Select Packages
Select packages to install

Search: make | Clear | ○ Keep ◉ Curr ○ Exp | View: Category

Category	New	Bin?	Src?	Size	Package
⊟ All Default					
⊞ Archive Default					
⊞ Debug Default					
⊟ Devel Default					
	Skip	n/a	n/a	3k	automake: Wrapper scripts for automake and aclocal
	Skip	n/a	n/a	714k	automake1.10: (1.10) a tool for generating GNU-compliant
	Skip	n/a	n/a	860k	automake1.11: (1.11) a tool for generating GNU-compliant
	Skip	n/a	n/a	733k	automake1.12: (1.12) a tool for generating GNU-compliant
	Skip	n/a	n/a	780k	automake1.13: (1.13) a tool for generating GNU-compliant
	Skip	n/a	n/a	581k	automake1.14: (1.14) a tool for generating GNU-compliant
	Skip	n/a	n/a	781k	automake1.15: (1.15) a tool for generating GNU-compliant
	Skip	n/a	n/a	248k	automake1.4: (1.4) a tool for generating GNU-compliant M
	Skip	n/a	n/a	332k	automake1.5: (1.5) a tool for generating GNU-compliant M
	Skip	n/a	n/a	365k	automake1.6: (1.6) a tool for generating GNU-compliant M
	Skip	n/a	n/a	426k	automake1.7: (1.7) a tool for generating GNU-compliant M
	Skip	n/a	n/a	499k	automake1.8: (1.8) a tool for generating GNU-compliant M
	Skip	n/a	n/a	557k	automake1.9: (1.9) a tool for generating GNU-compliant M
	Skip	n/a	n/a	3,675k	cmake: Cross-platform makefile generation system
	Skip	n/a	n/a	767k	cmake-doc: Cross-platform makefile generation system (do
	Skip	n/a	n/a	1,196k	cmake-gui: Cross-platform makefile generation system (GU
	Skip	n/a	n/a	163k	extra-cmake-modules: Extra CMake Modules for KDE
	Skip	n/a	n/a	420k	gcc-tools-epoch1-automake: (gcc-special) a tool for gener
	Skip	n/a	n/a	589k	gcc-tools-epoch2-automake: (gcc-special) a tool for gener
	Skip	n/a	n/a	6k	gccmakedep: X Makefile dependency tool for GCC
	Skip	n/a	n/a	34k	imake: X Imake legacy build system
	4.2.1-1	☒	☐	449k	make: The GNU version of the 'make' utility
	Skip	n/a	n/a	29k	makedepend: X Makefile dependency tool

☑ Hide obsolete packages

< Back | Next > | Cancel

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.

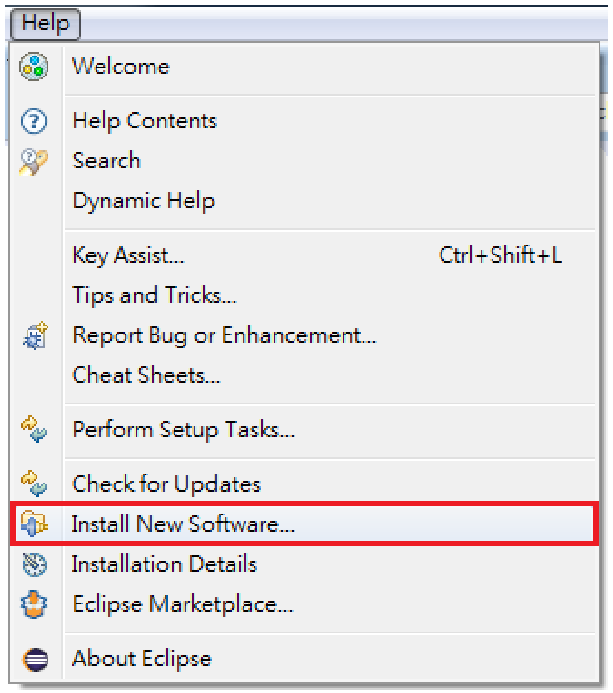
2.2 Eclipse environment

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.

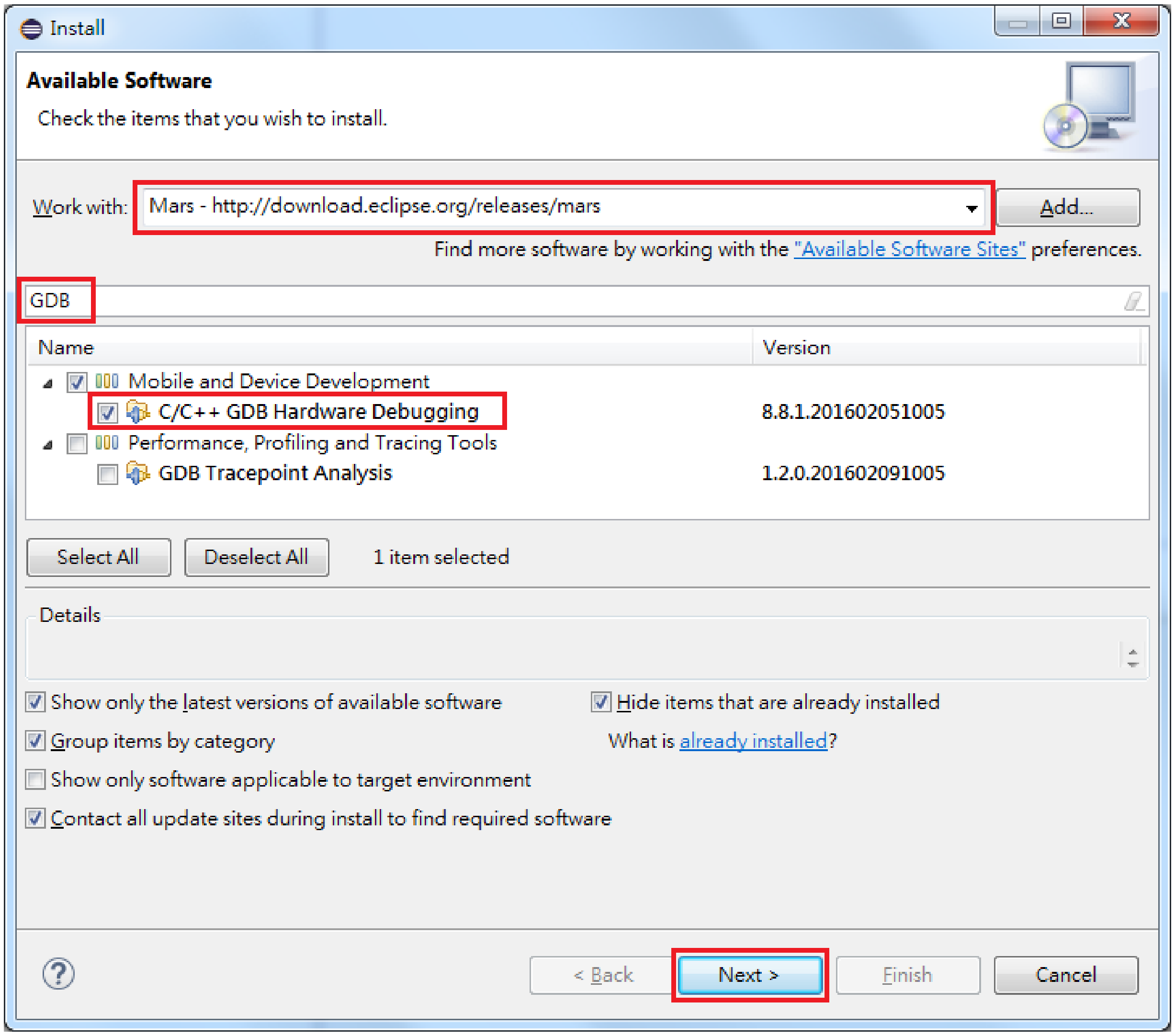
2.2.2 Plugin installation

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.



Work with: Mars - http://download.eclipse.org/releases/mars

Find more software by working with the "Available Software Sites" preferences.

GDB

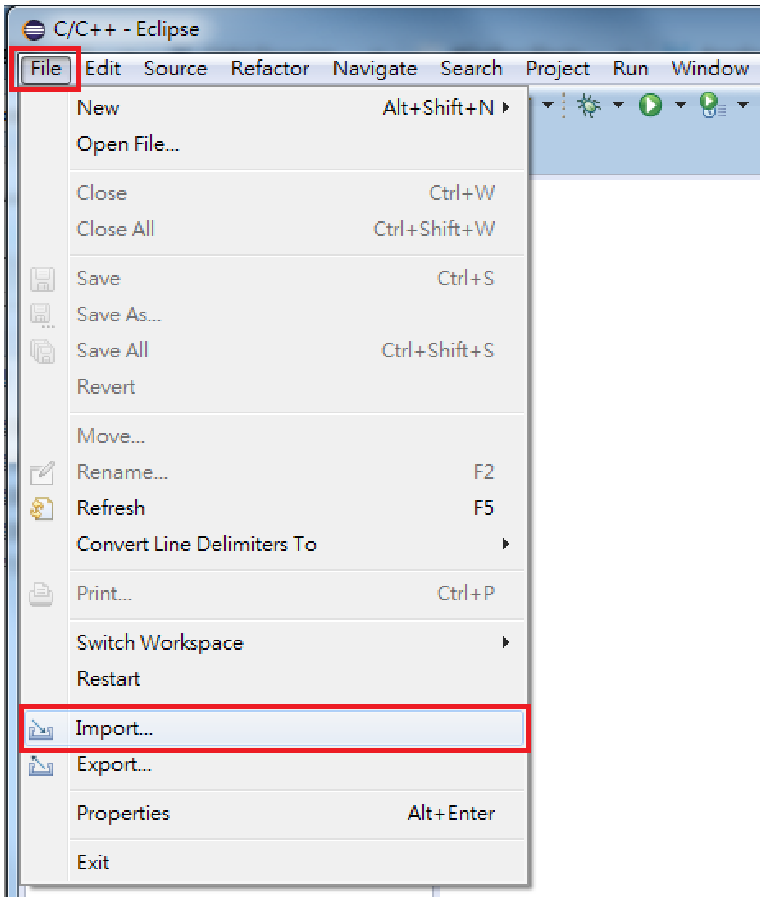
Name	Version
[✓] Mobile and Device Development	
[✓] C/C++ GDB Hardware Debugging	8.8.1.201602051005
[ ] Performance, Profiling and Tracing Tools	
[ ] GDB Tracepoint Analysis	1.2.0.201602091005

1 item selected

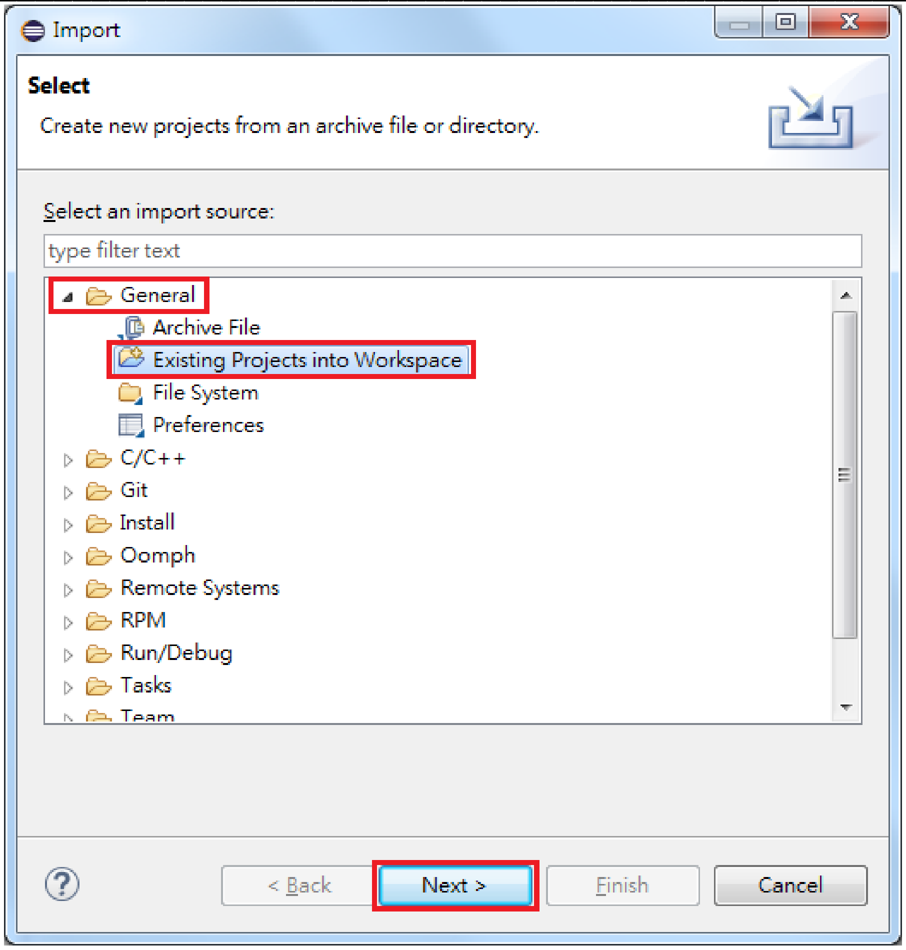
[✓] Show only the latest versions of available software
[✓] Hide items that are already installed
[✓] Group items by category
What is already installed?
[ ] Show only software applicable to target environment
[✓] Contact all update sites during install to find required software

2.2.3 Import Ameba SDK application

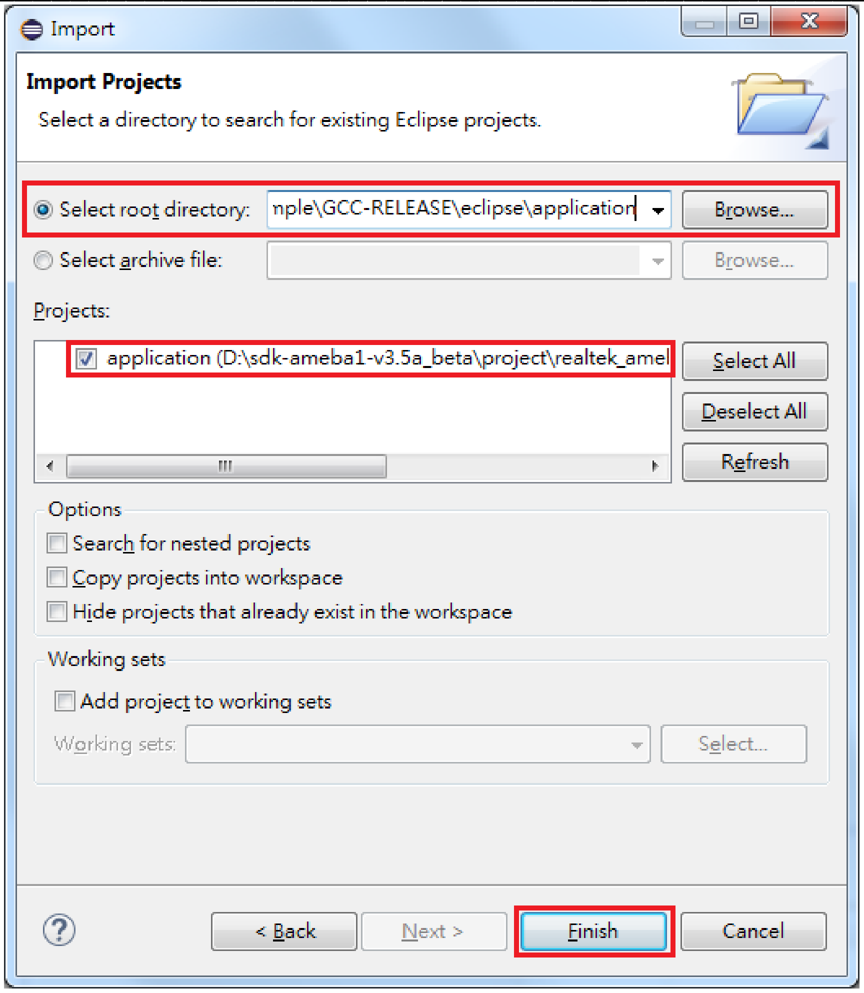
After the plugin installation, we now import the application project into Eclipse workspace. Please click “File -> Import…” on the upper left corner of Eclipse IDE:



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.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

Select root directory: nple\GCC-RELEASE\eclipse\application

Select archive file:

Projects:

[✓] application (D:\sdk-ameba1-v3.5a_beta\project\realtek_amel

Options

[ ] Search for nested projects

[ ] Copy projects into workspace

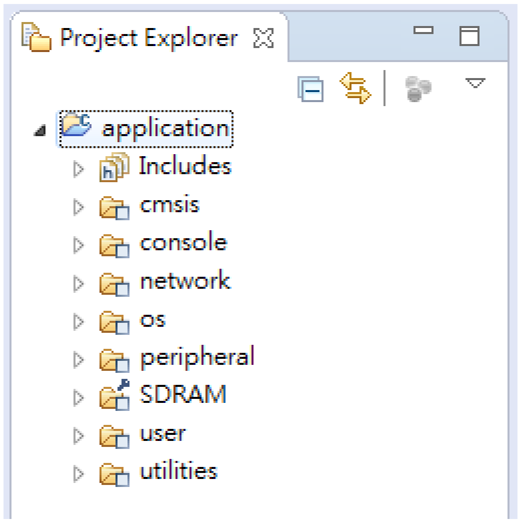
[ ] Hide projects that already exist in the workspace

Working sets

[ ] Add project to working sets

Working sets:

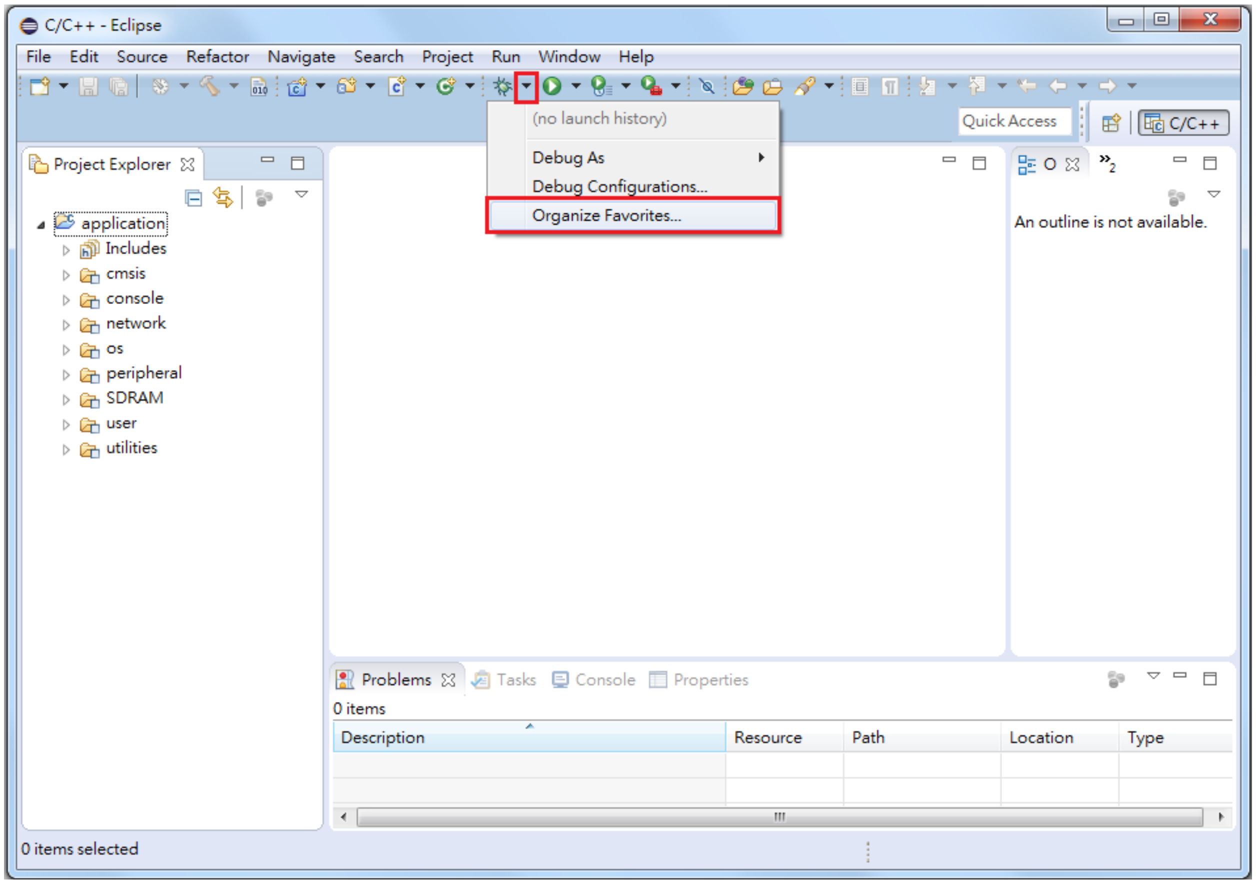
There should be an “application” project in “Project Explorer” view. The source files of SDK are listed under their corresponding directories in the project.



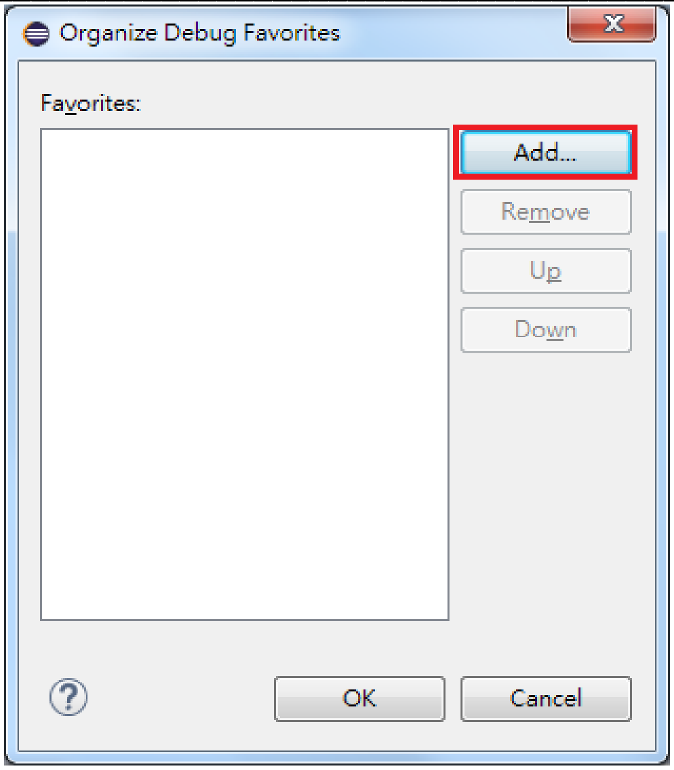
2.2.4 Configure debug and external tool launchers

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.

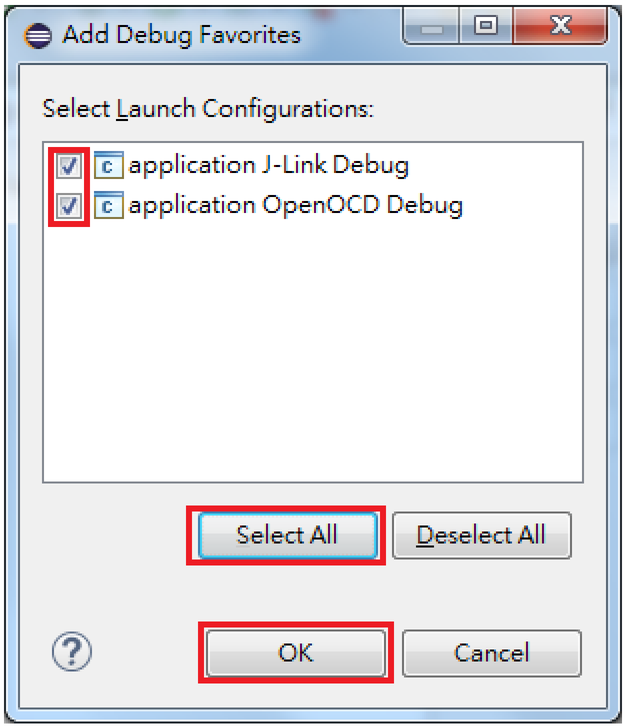
For debug launchers, please click “Debug -> Organize Favorites…”:



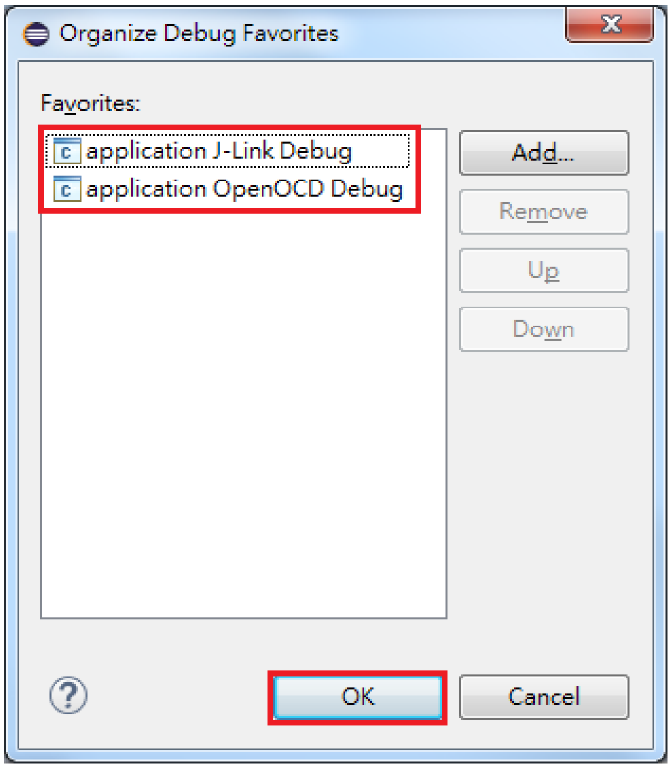
If the popped out window does not contains “application J-Link Debug” and “application OpenOCD Debug” launchers like below figure, click the “Add…” button to add them.



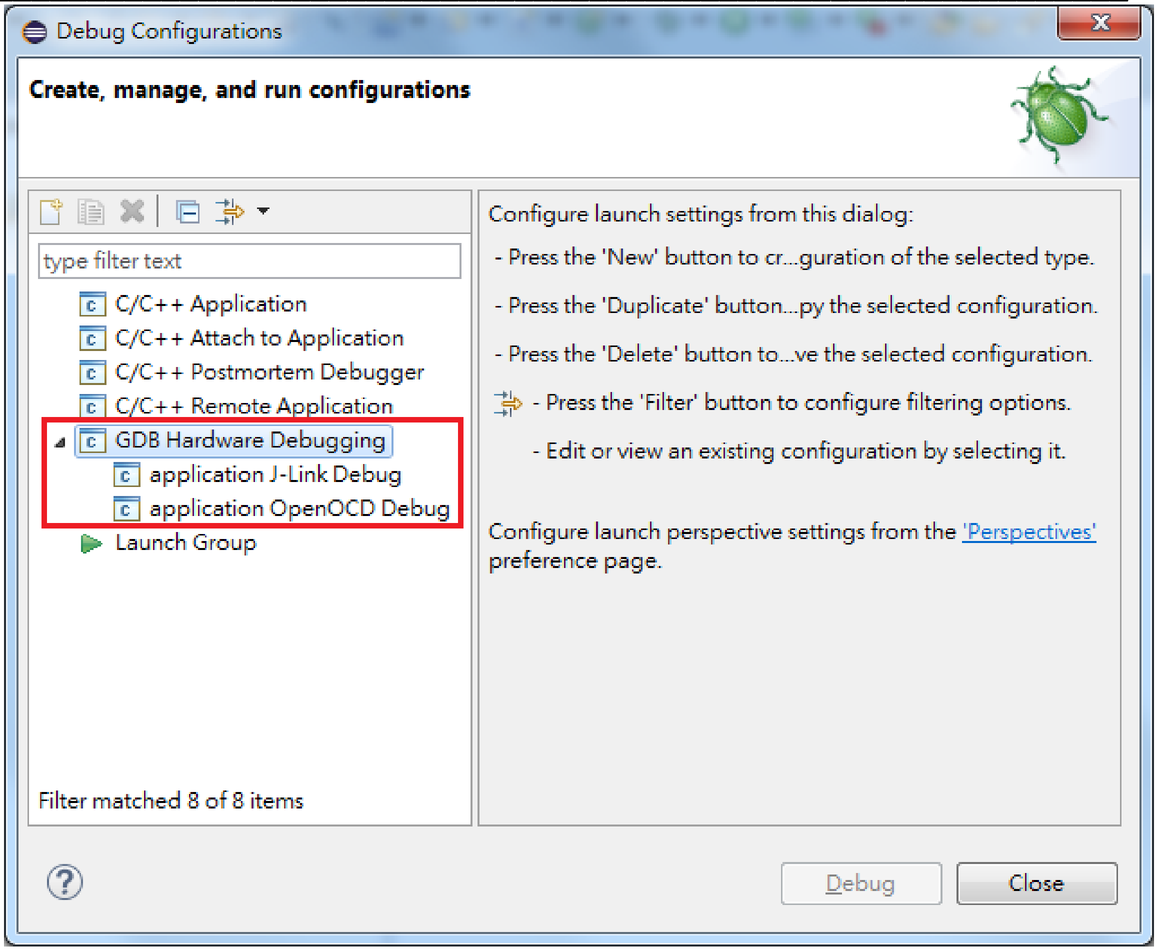
Click “Select All” to select both “application J-Link Debug” and “application OpenOCD Debug” launchers and click “OK”.



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

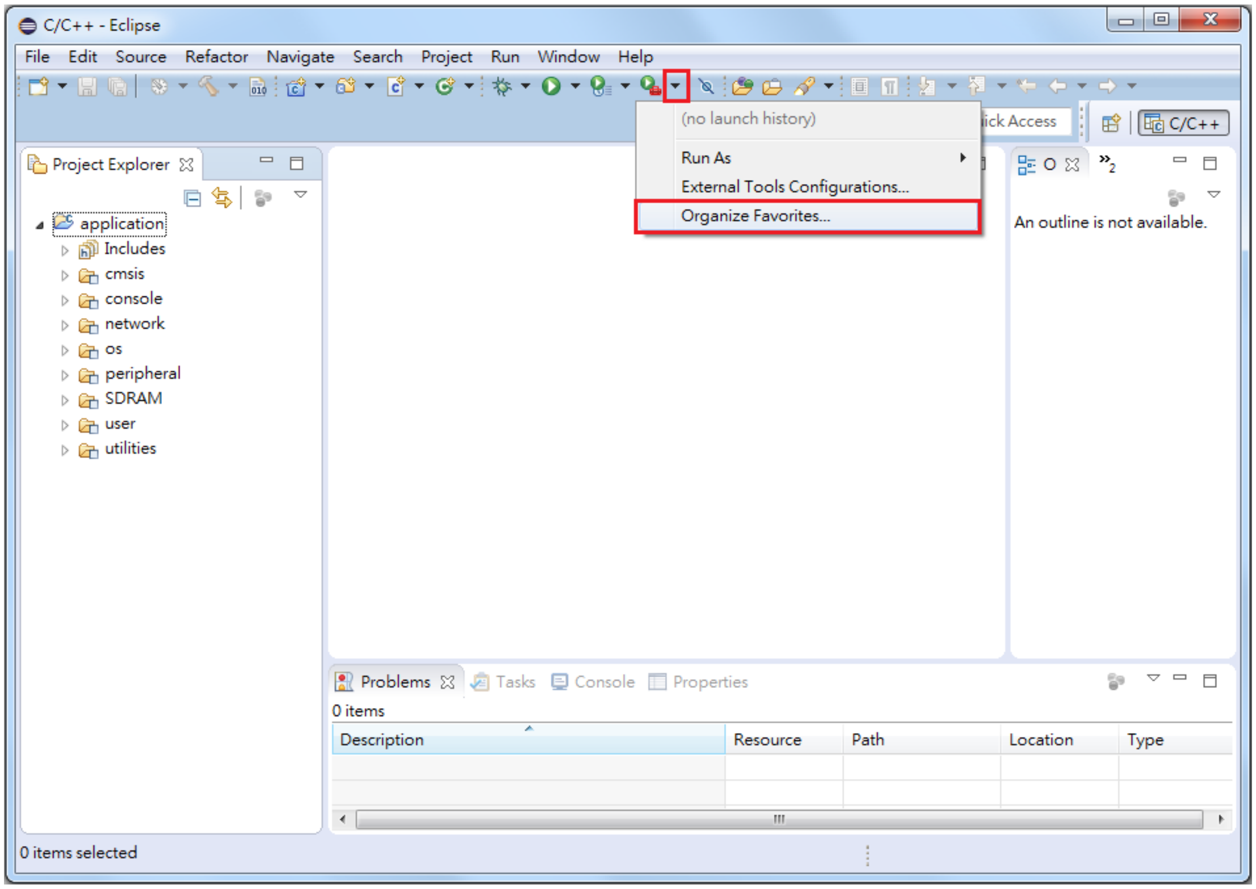
Filter matched 8 of 8 items

Configure launch settings from this dialog:

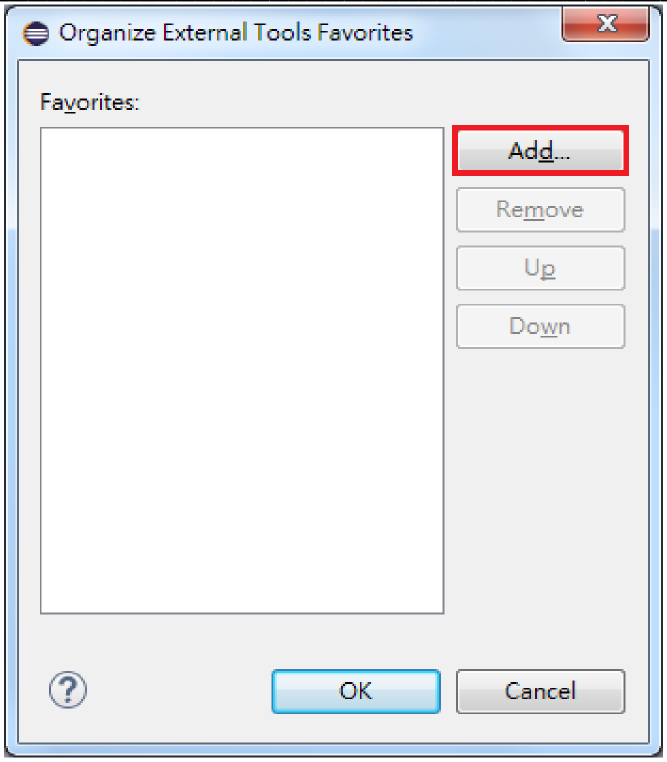
- Press the 'New' button to cr...guration of the selected type.
- Press the 'Duplicate' button...py the selected configuration.
- Press the 'Delete' button to...ve the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

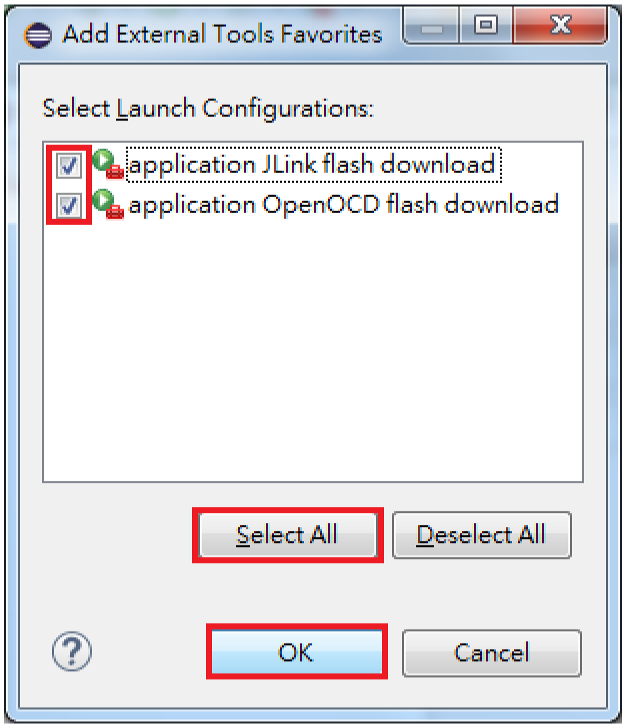
And for external tools launchers, the setting steps are quite similar to debug launchers setting. First, please click “External Tools -> Organize Favorites…”:


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
(no launch history)
Run As
External Tools Configurations...
Organize Favorites...
Project Explorer
application
Includes
cmsis
console
network
os
peripheral
SDRAM
user
utilities
An outline is not available.
Problems Tasks Console Properties
0 items
Description Resource Path Location Type
0 items selected

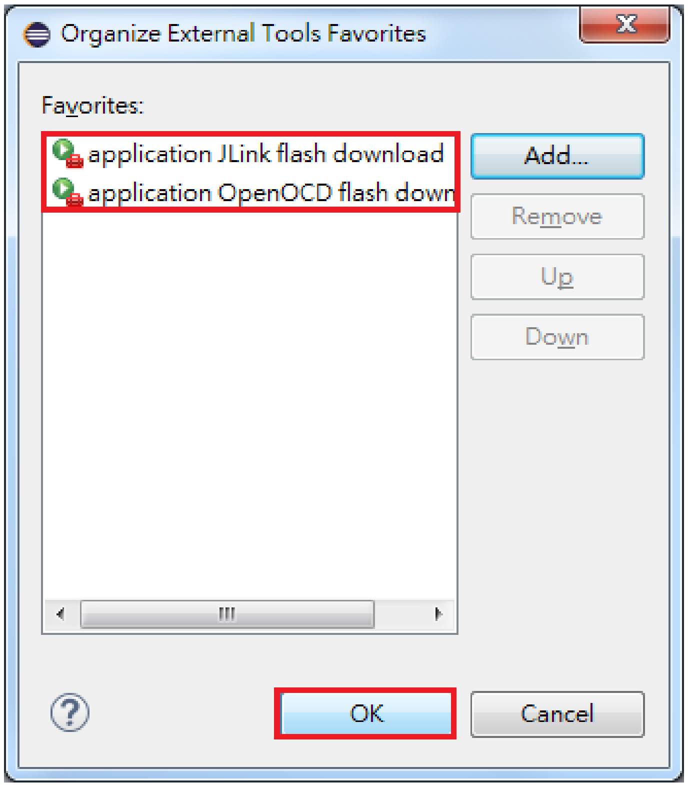
If the popped out window does not contains “application JLink flash download” and “application OpenOCD flash download” launchers like below figure, click the “Add…” button to add them.



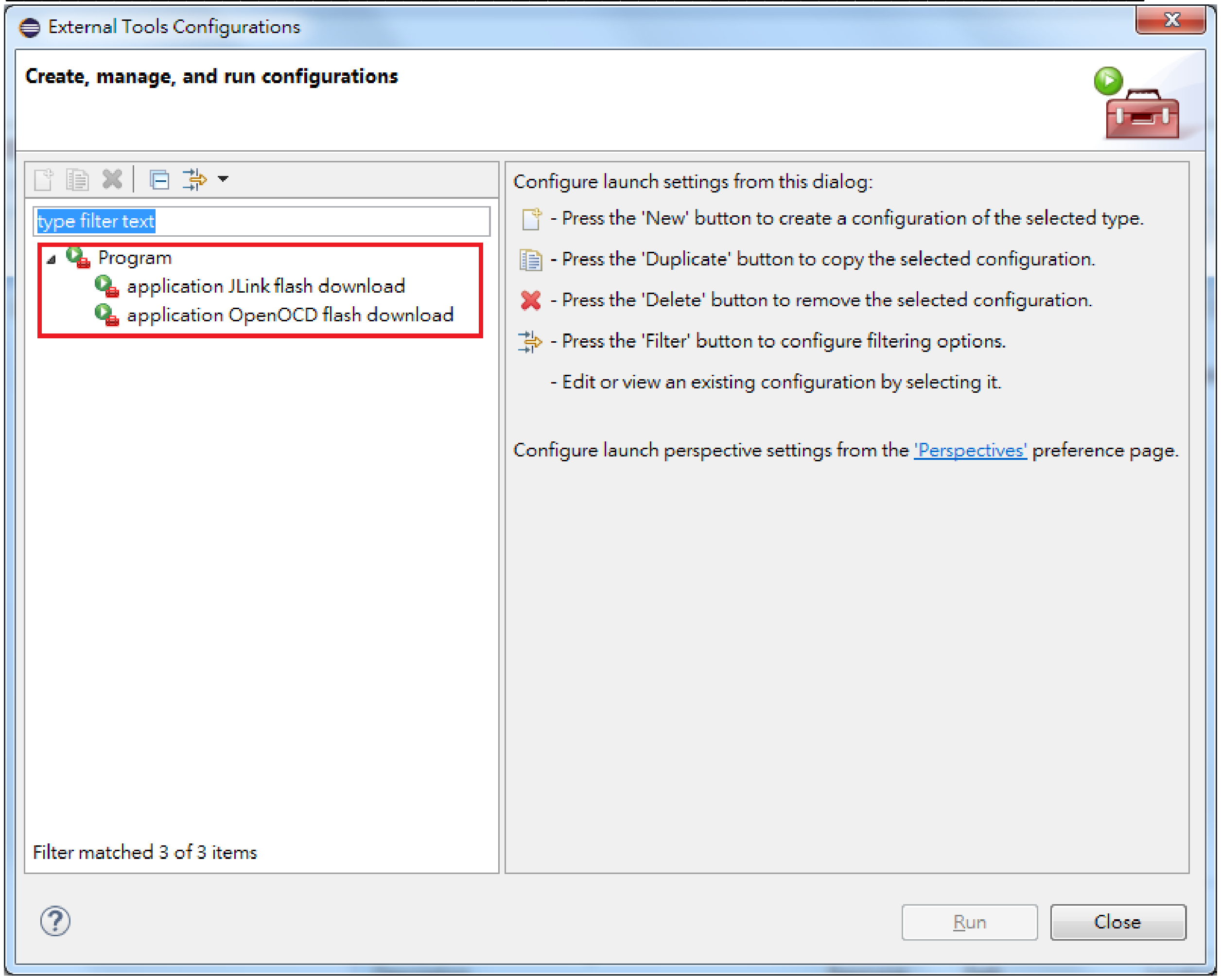
Click “Select All” to select both “application JLink flash download” and “application OpenOCD flash download” launchers and click “OK”.


Add External Tools Favorites
Select Launch Configurations:
application JLink flash download
application OpenOCD flash download
Select All
Deselect All
OK
Cancel

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


Organize External Tools Favorites
Favorites:
application JLink flash download
application OpenOCD flash down
Add...
Remove
Up
Down
OK
Cancel

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.



External Tools Configurations

Create, manage, and run configurations

type filter text

- Program
  - application JLink flash download
  - application OpenOCD flash download

Filter matched 3 of 3 items

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove 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.

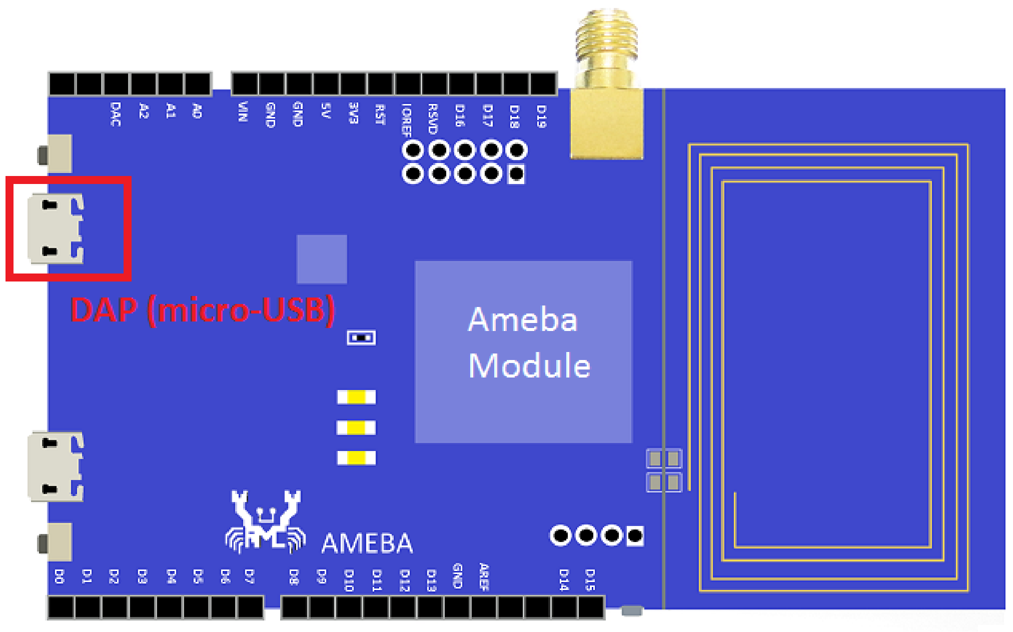
2.3 Debugger setting

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.

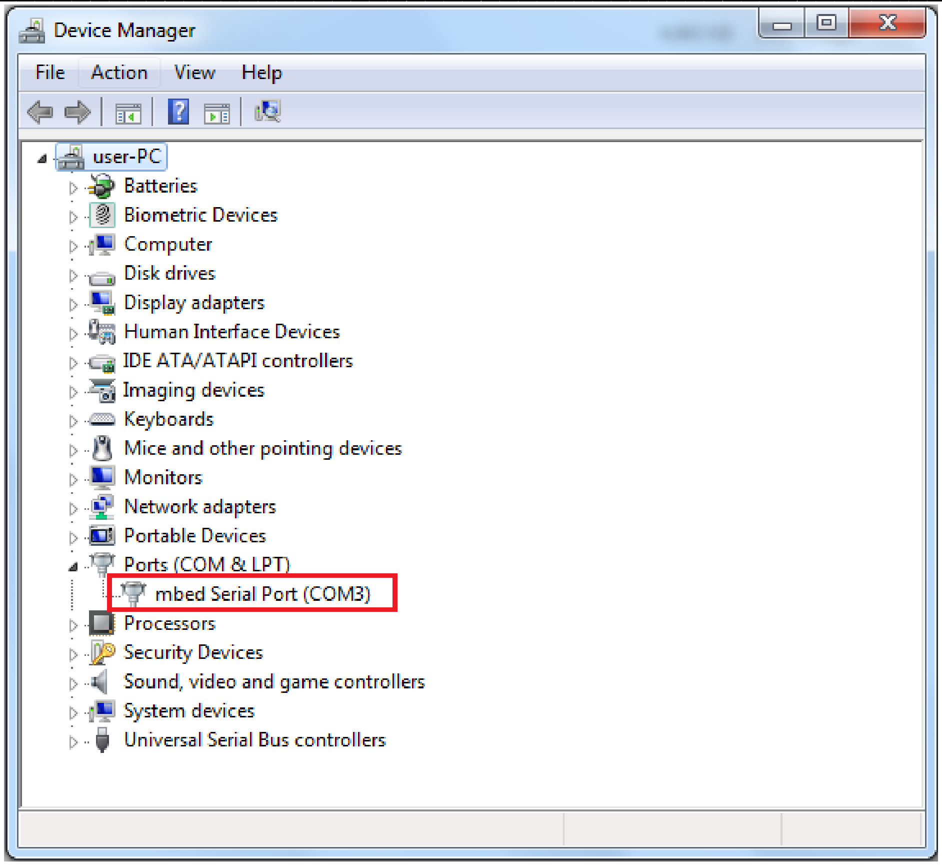
2.3.1 OpenOCD/CMSIS-DAP

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:

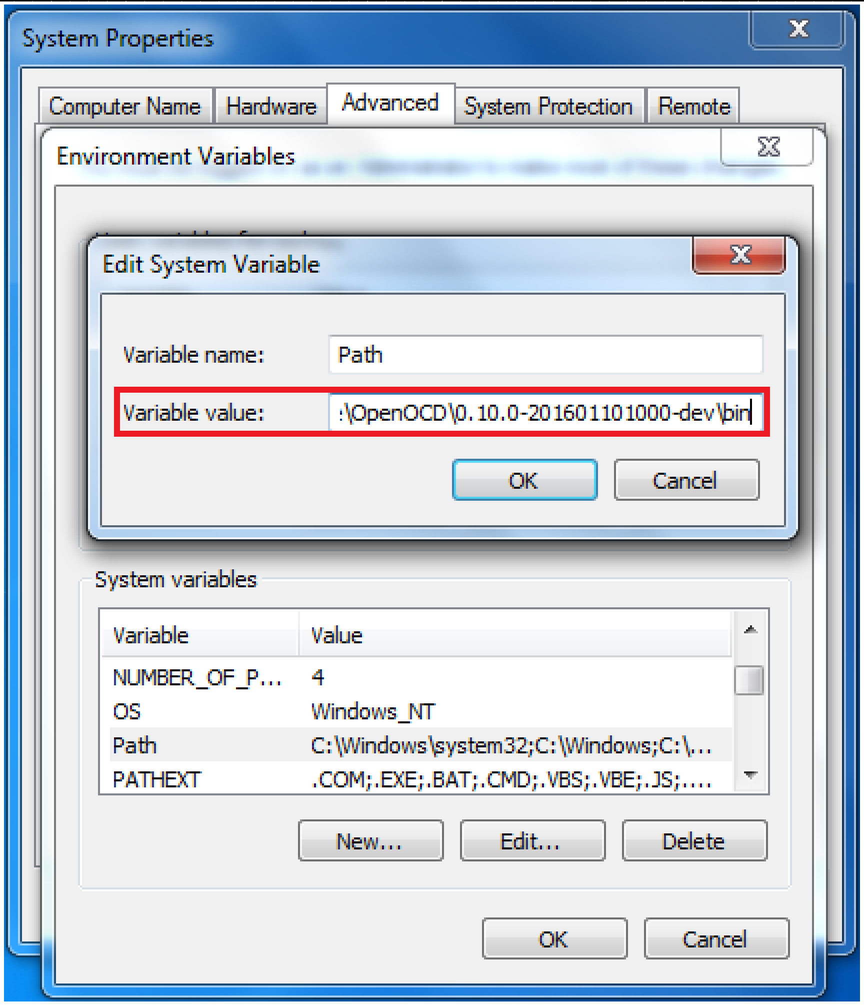

DAP (micro-USB)
Ameba Module
AMEBA
DAC A2 A1 A0 VIN GND GND 5V 3V3 RST IOREF RSVD D16 D17 D18 D19
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 GND AREF D14 D15

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.



user-PC
- Batteries
- Biometric Devices
- Computer
- Disk drives
- Display adapters
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Imaging devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Portable Devices
- Ports (COM & LPT)
  - mbed Serial Port (COM3)
- Processors
- Security Devices
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers

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).

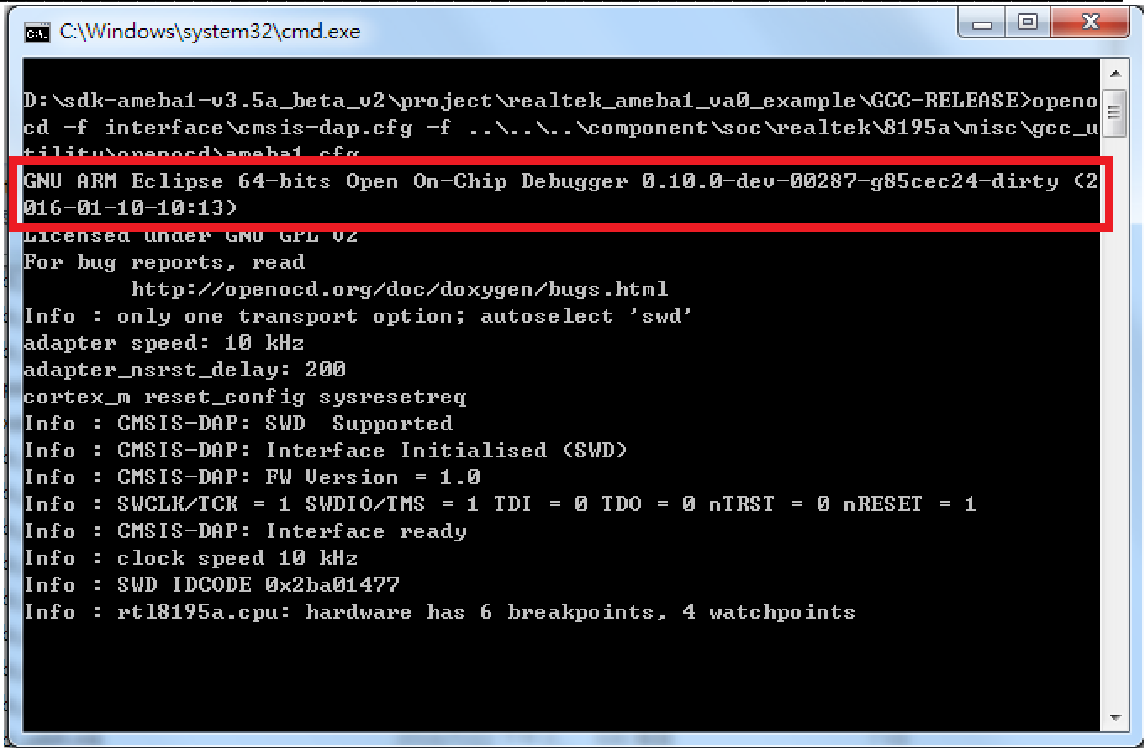


Variable name: Path

Variable value: :\OpenOCD\0.10.0-201601101000-dev\bin

Variable	Value
NUMBER_OF_P...	4
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

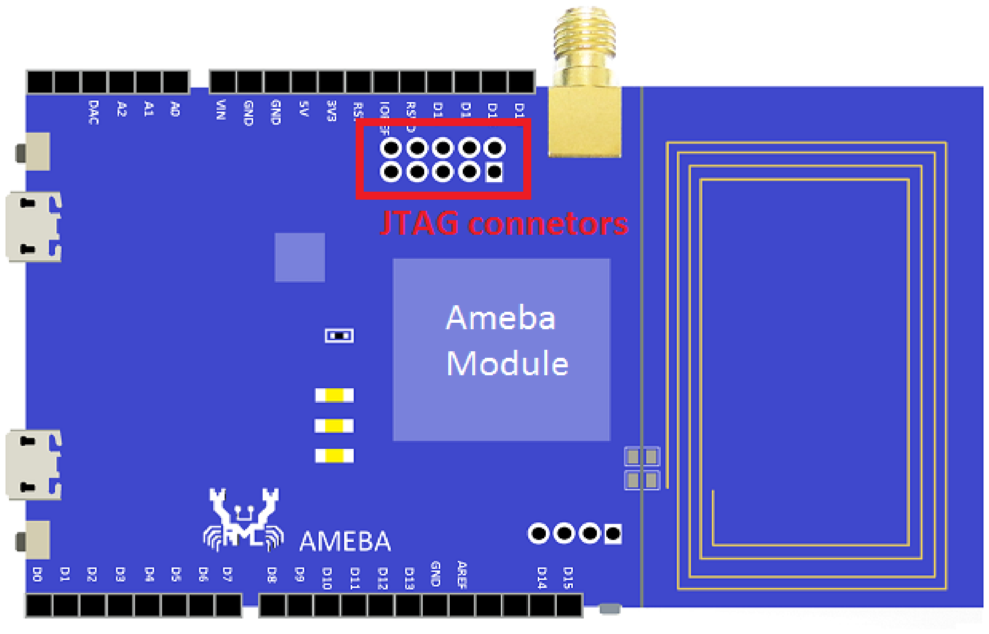
If OpenOCD has been installed correctly, execute project\realtek_ameba1_va0_example\GCC-RELEASE\run_openocd.bat to start GDB server and you should see some messages like below figure. This window should **NOT** be closed if you want to download software or enter debugger.


C:\Windows\system32\cmd.exe

```
D:\sdk-ameba1-v3.5a_beta_v2\project\realtek_ameba1_va0_example\GCC-RELEASE>openo
cd -f interface\cmsis-dap.cfg -f ..\..\..\component\soc\realtek\8195a\misc\gcc_u
tility\openocd\ameba1.cfg
GNU ARM Eclipse 64-bits Open On-Chip Debugger 0.10.0-dev-00287-g85cec24-dirty (2
016-01-10-10:13)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'swd'
adapter speed: 10 kHz
adapter_nsrst_delay: 200
cortex_m reset_config sysresetreq
Info : CMSIS-DAP: SWD  Supported
Info : CMSIS-DAP: Interface Initialised (SWD)
Info : CMSIS-DAP: FW Version = 1.0
Info : SWCLK/TCK = 1 SWDIO/TMS = 1 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1
Info : CMSIS-DAP: Interface ready
Info : clock speed 10 kHz
Info : SWD IDCODE 0x2ba01477
Info : rtl8195a.cpu: hardware has 6 breakpoints, 4 watchpoints
```

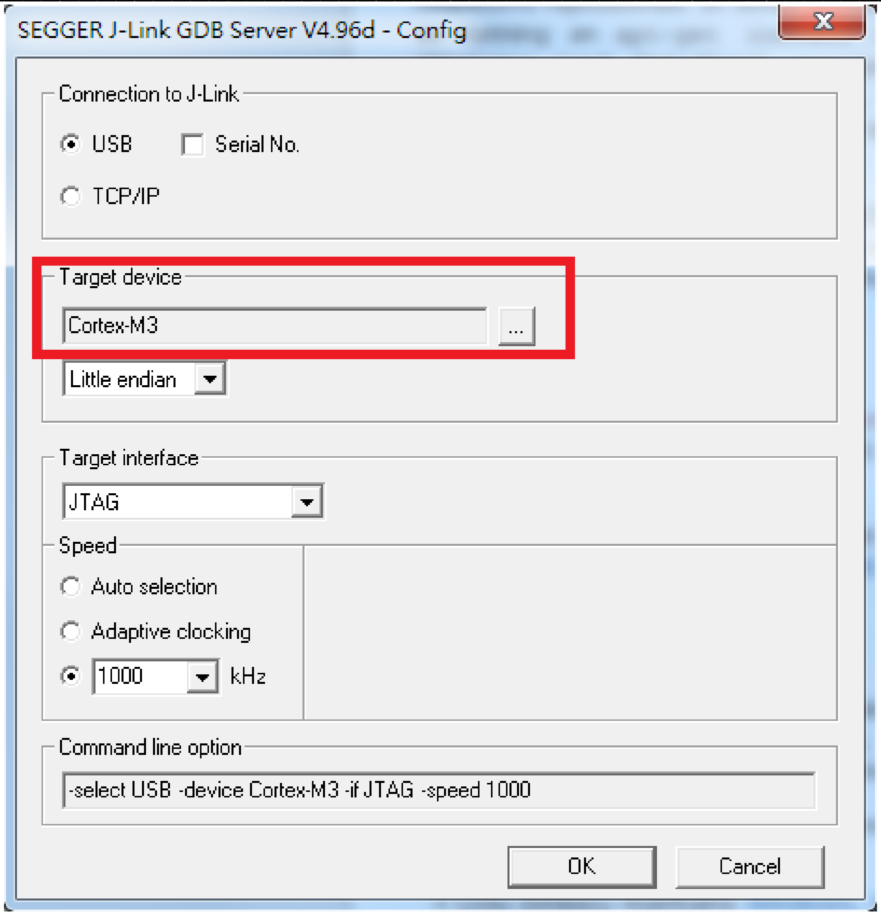
2.3.2 J-Link

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:


SEGGER J-Link GDB Server V4.96d - Config

Connection to J-Link

USB [✓] Serial No. [ ]

TCP/IP

Target device

Cortex-M3

Little endian

Target interface

JTAG

Speed

Auto selection

Adaptive clocking

[✓] 1000 kHz

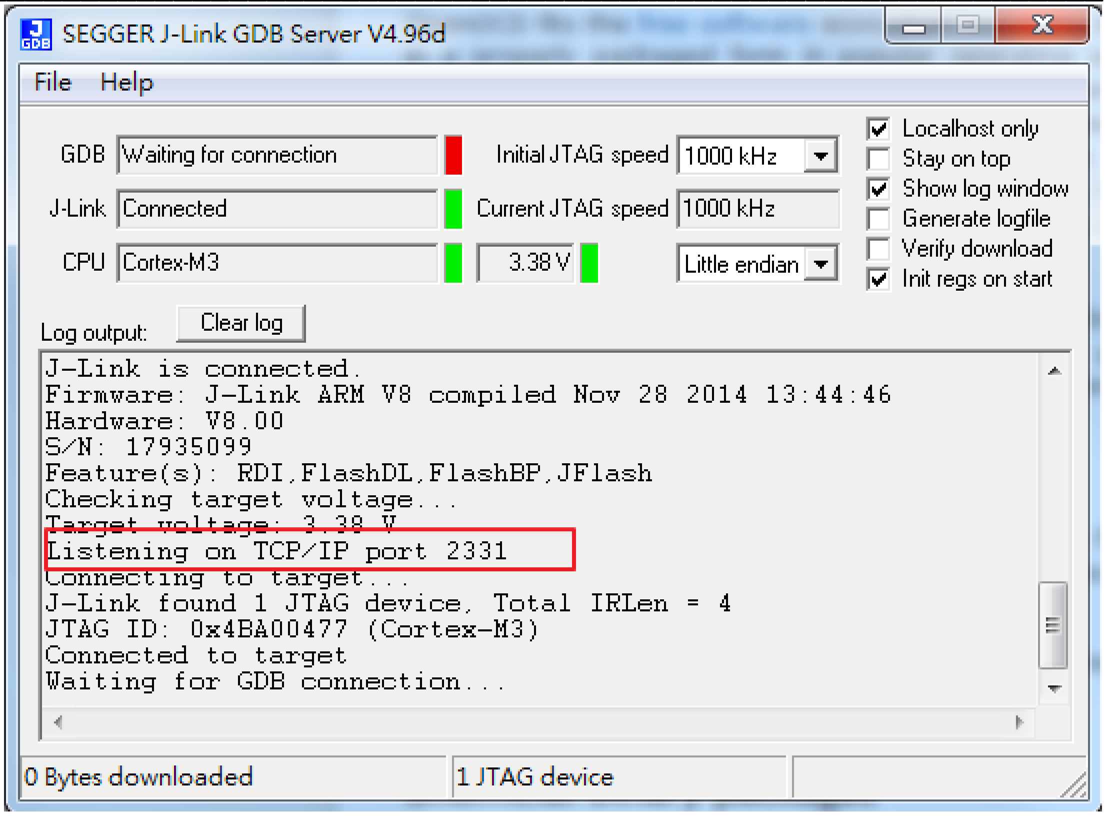
Command line option

-select USB -device Cortex-M3 -if JTAG -speed 1000

OK Cancel

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.



GDB: Waiting for connection 🟥
J-Link: Connected 🟩
CPU: Cortex-M3 🟩
3.38 V 🟩
Initial JTAG speed: 1000 kHz
Current JTAG speed: 1000 kHz
Little endian

[✓] Localhost only
[ ] Stay on top
[✓] Show log window
[ ] Generate logfile
[ ] Verify download
[✓] Init regs on start

Log output:

```
J-Link is connected.
Firmware: J-Link ARM V8 compiled Nov 28 2014 13:44:46
Hardware: V8.00
S/N: 17935099
Feature(s): RDI,FlashDL,FlashBP,JFlash
Checking target voltage...
Target voltage: 3.38 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 1 JTAG device, Total IRLen = 4
JTAG ID: 0x4BA00477 (Cortex-M3)
Connected to target
Waiting for GDB connection...
```

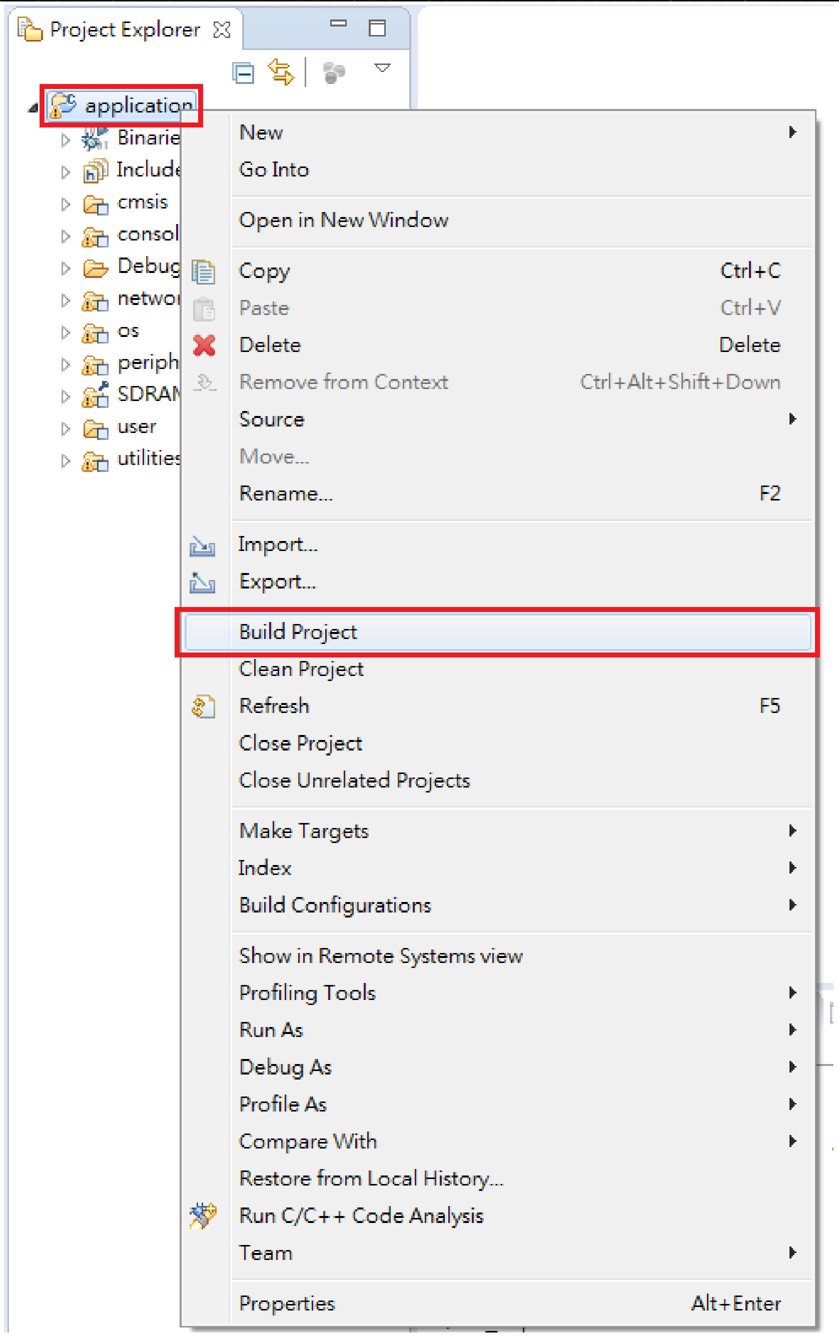
0 Bytes downloaded | 1 JTAG device

3 How to build, download and enter debug mode

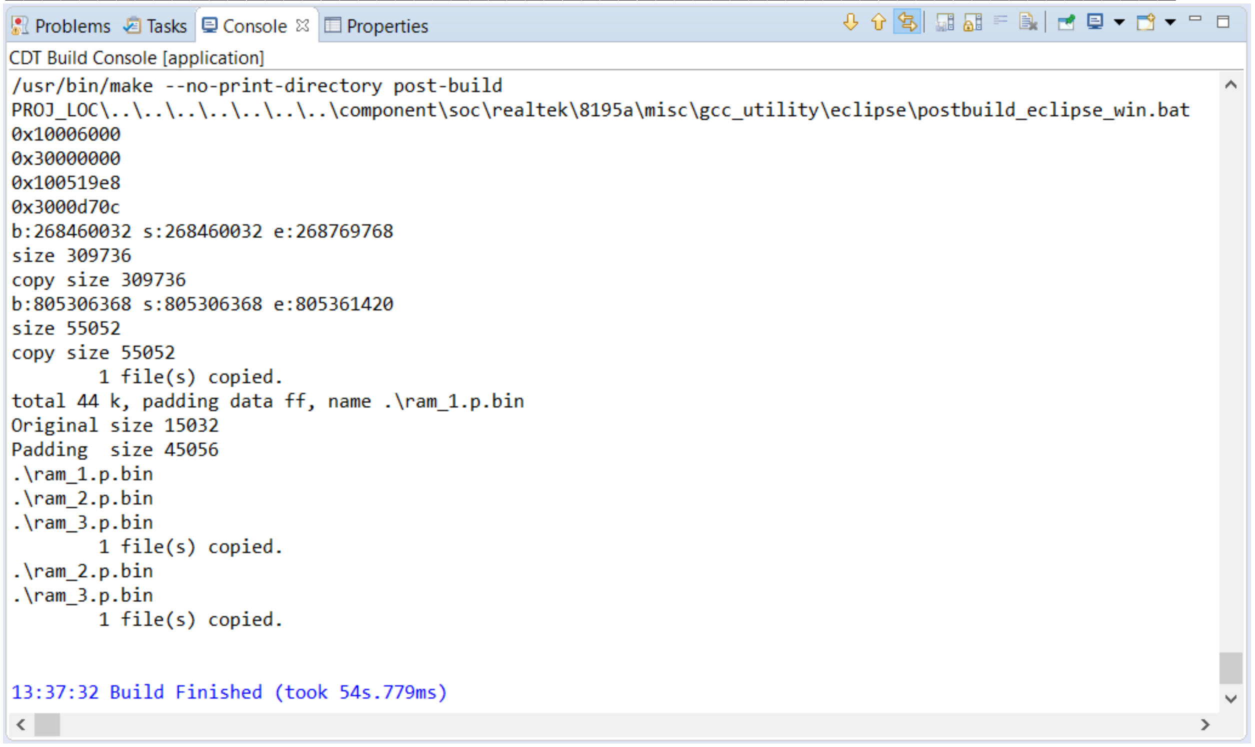
In this section, we illustrate how to build, download, and enter debug mode. Note that in Sec. 3.2 the code is downloaded to flash but in Sec. 3.3 the code is written into RAM before we enter debug mode.

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:

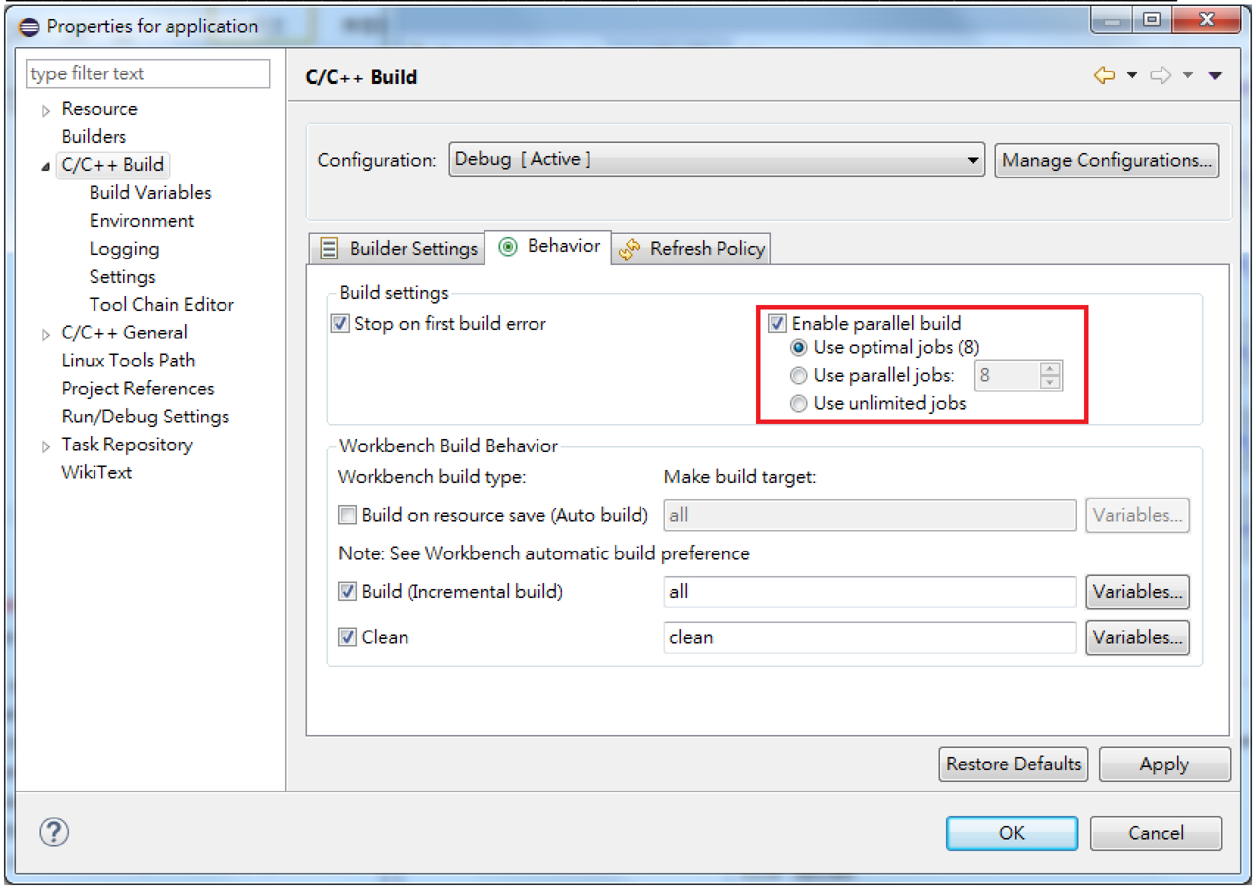


```
/usr/bin/make --no-print-directory post-build
PROJ_LOC\..\..\..\..\..\..\..\component\soc\realtek\8195a\misc\gcc_utility\eclipse\postbuild_eclipse_win.bat
0x10006000
0x30000000
0x100519e8
0x3000d70c
b:268460032 s:268460032 e:268769768
size 309736
copy size 309736
b:805306368 s:805306368 e:805361420
size 55052
copy size 55052
        1 file(s) copied.
total 44 k, padding data ff, name .\ram_1.p.bin
Original size 15032
Padding  size 45056
.\ram_1.p.bin
.\ram_2.p.bin
.\ram_3.p.bin
        1 file(s) copied.
.\ram_2.p.bin
.\ram_3.p.bin
        1 file(s) copied.


13:37:32 Build Finished (took 54s.779ms)
```

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.


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

C/C++ Build

Configuration: Debug [Active]   Manage Configurations...

Builder Settings | Behavior | Refresh Policy

Build settings

- [✓] Stop on first build error
- [✓] Enable parallel build
  - ● Use optimal jobs (8)
  - ○ Use parallel jobs: 8
  - ○ Use unlimited jobs

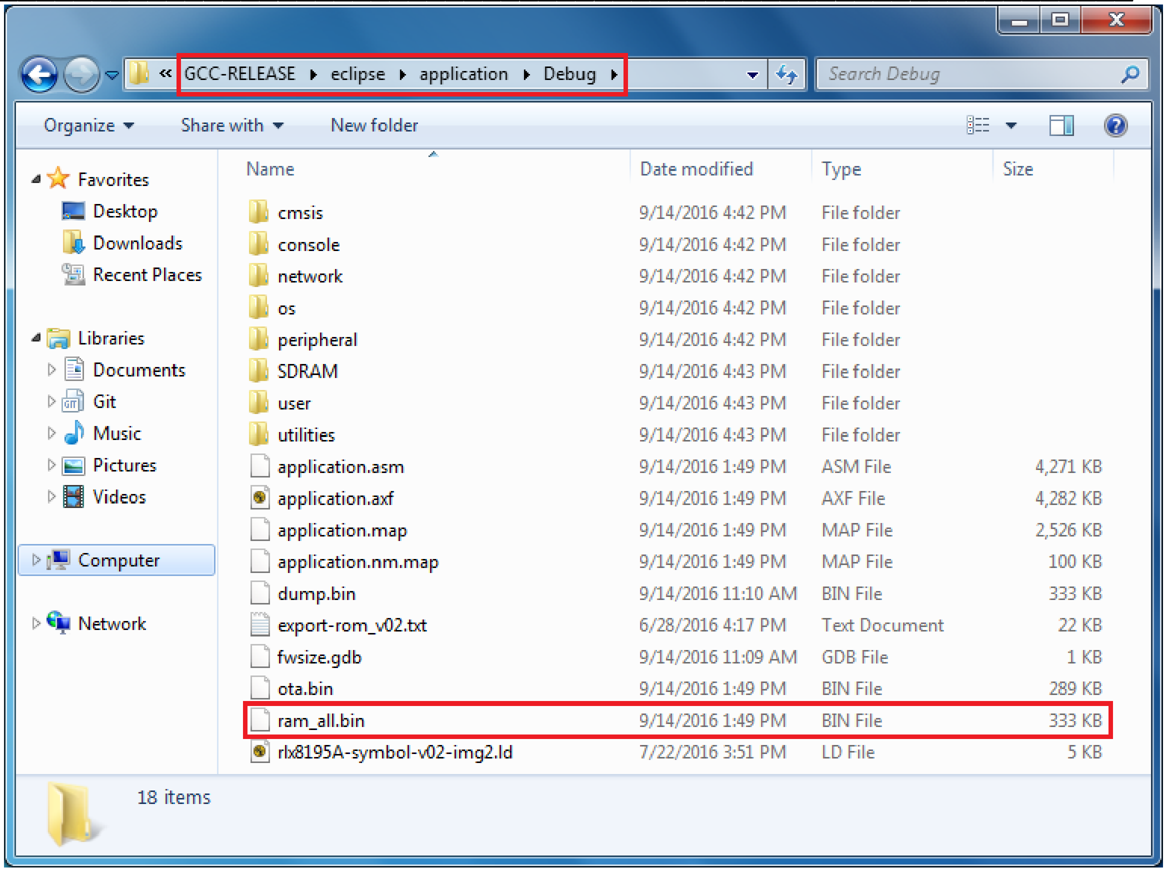
Workbench Build Behavior

Workbench build type:	Make build target:
[ ] Build on resource save (Auto build)	all
Note: See Workbench automatic build preference	
[✓] Build (Incremental build)	all
[✓] Clean	clean

Restore Defaults   Apply

OK   Cancel

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\:

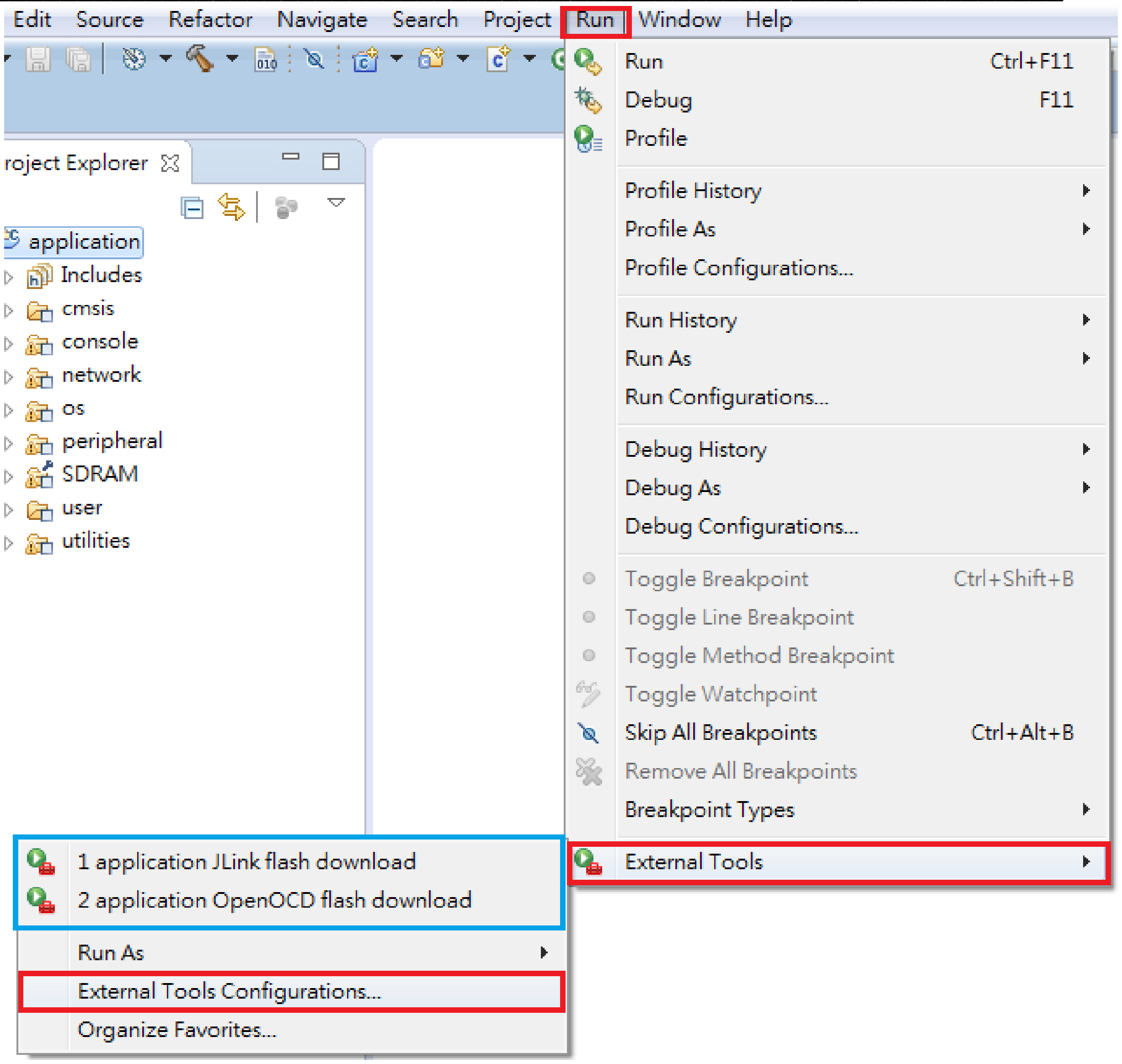


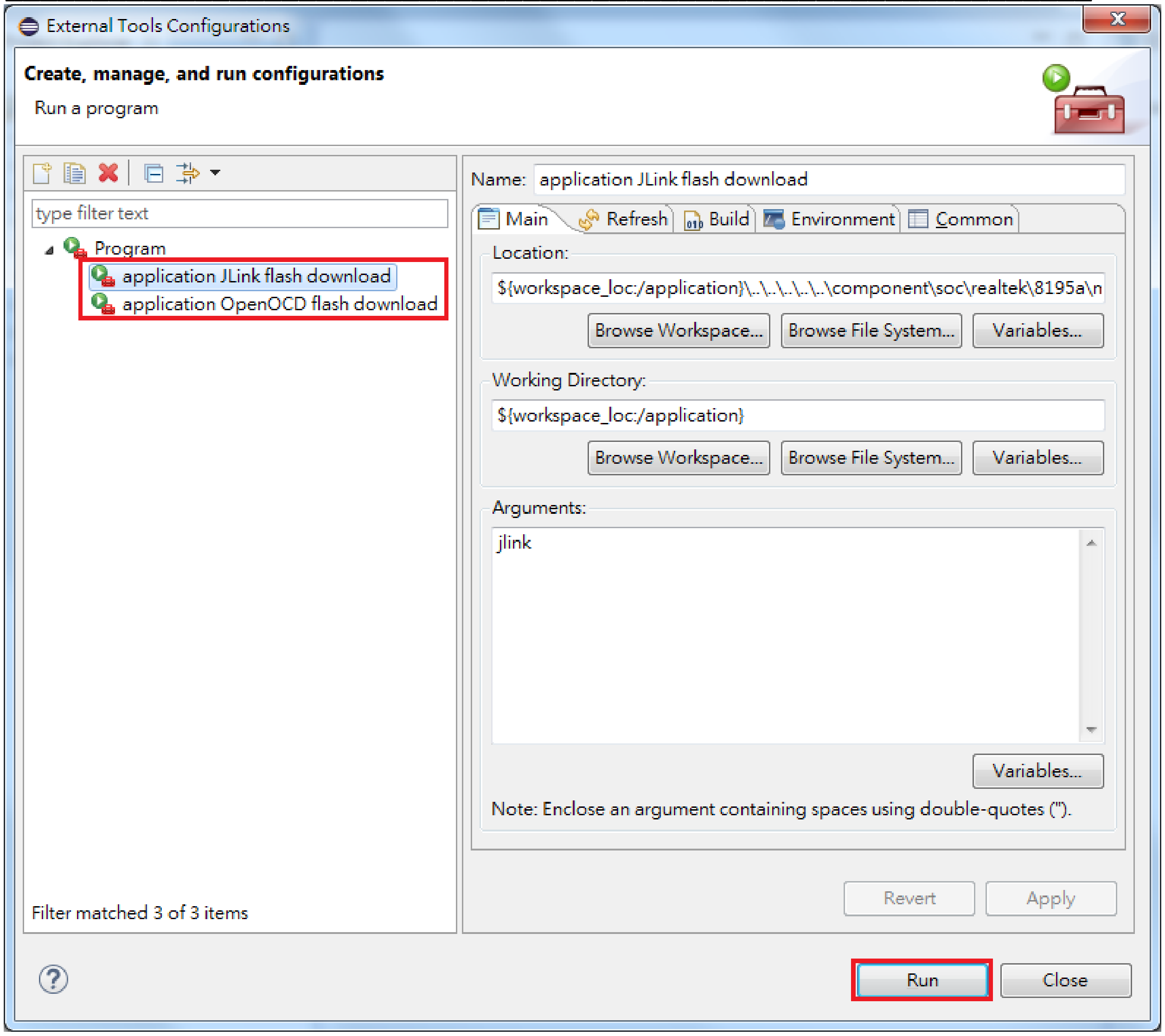
Name	Date modified	Type	Size
cmsis	9/14/2016 4:42 PM	File folder	
console	9/14/2016 4:42 PM	File folder	
network	9/14/2016 4:42 PM	File folder	
os	9/14/2016 4:42 PM	File folder	
peripheral	9/14/2016 4:42 PM	File folder	
SDRAM	9/14/2016 4:43 PM	File folder	
user	9/14/2016 4:43 PM	File folder	
utilities	9/14/2016 4:43 PM	File folder	
application.asm	9/14/2016 1:49 PM	ASM File	4,271 KB
application.axf	9/14/2016 1:49 PM	AXF File	4,282 KB
application.map	9/14/2016 1:49 PM	MAP File	2,526 KB
application.nm.map	9/14/2016 1:49 PM	MAP File	100 KB
dump.bin	9/14/2016 11:10 AM	BIN File	333 KB
export-rom_v02.txt	6/28/2016 4:17 PM	Text Document	22 KB
fwsize.gdb	9/14/2016 11:09 AM	GDB File	1 KB
ota.bin	9/14/2016 1:49 PM	BIN File	289 KB
ram_all.bin	9/14/2016 1:49 PM	BIN File	333 KB
rlx8195A-symbol-v02-img2.ld	7/22/2016 3:51 PM	LD File	5 KB

18 items

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...


External Tools Configurations
Create, manage, and run configurations
Run a program
type filter text
Program
application JLink flash download
application OpenOCD flash download
Filter matched 3 of 3 items
Main
Refresh
Build
Environment
Common
Browse Workspace...
Browse File System...
Variables...
Revert
Apply
Run
Close

Name: application JLink flash download

Location: ${workspace_loc:/application}\..\..\..\..\..\component\soc\realtek\8195a\n

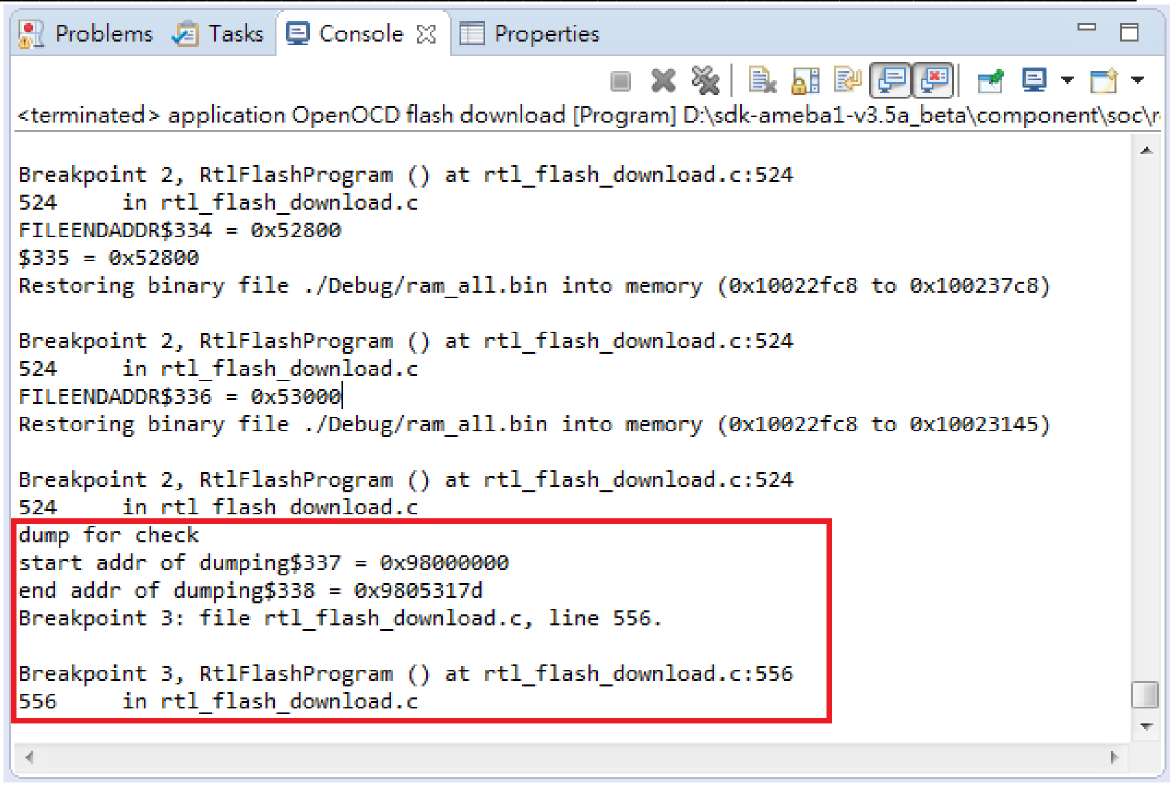
Working Directory: ${workspace_loc:/application}

Arguments: jlink

Note: Enclose an argument containing spaces using double-quotes (").

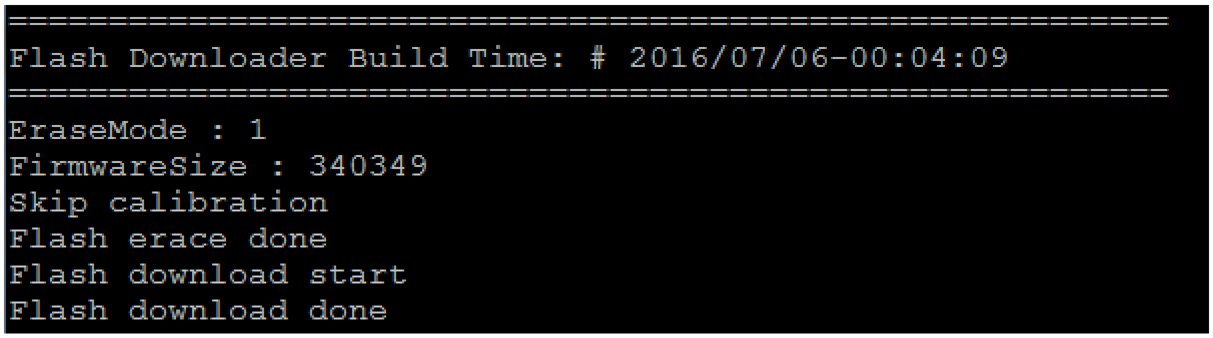
The download procedure would take a few seconds to finish. And it might also build the project before downloading to flash depends on the project status. After successful download, you might see some message like below figure in the "Console" view.

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.



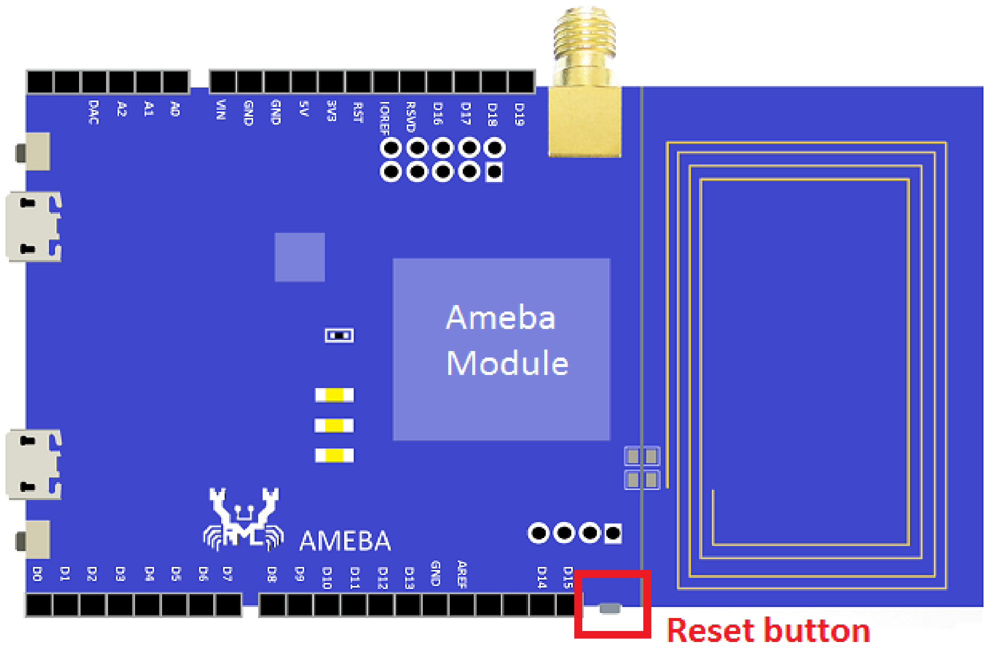
```
<terminated> application OpenOCD flash download [Program] D:\sdk-ameba1-v3.5a_beta\component\soc\r

Breakpoint 2, RtlFlashProgram () at rtl_flash_download.c:524
524      in rtl_flash_download.c
FILEENDADDR$334 = 0x52800
$335 = 0x52800
Restoring binary file ./Debug/ram_all.bin into memory (0x10022fc8 to 0x100237c8)

Breakpoint 2, RtlFlashProgram () at rtl_flash_download.c:524
524      in rtl_flash_download.c
FILEENDADDR$336 = 0x53000
Restoring binary file ./Debug/ram_all.bin into memory (0x10022fc8 to 0x10023145)

Breakpoint 2, RtlFlashProgram () at rtl_flash_download.c:524
524      in rtl_flash_download.c
dump for check
start addr of dumping$337 = 0x98000000
end addr of dumping$338 = 0x9805317d
Breakpoint 3: file rtl_flash_download.c, line 556.

Breakpoint 3, RtlFlashProgram () at rtl_flash_download.c:556
556      in rtl_flash_download.c
```

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



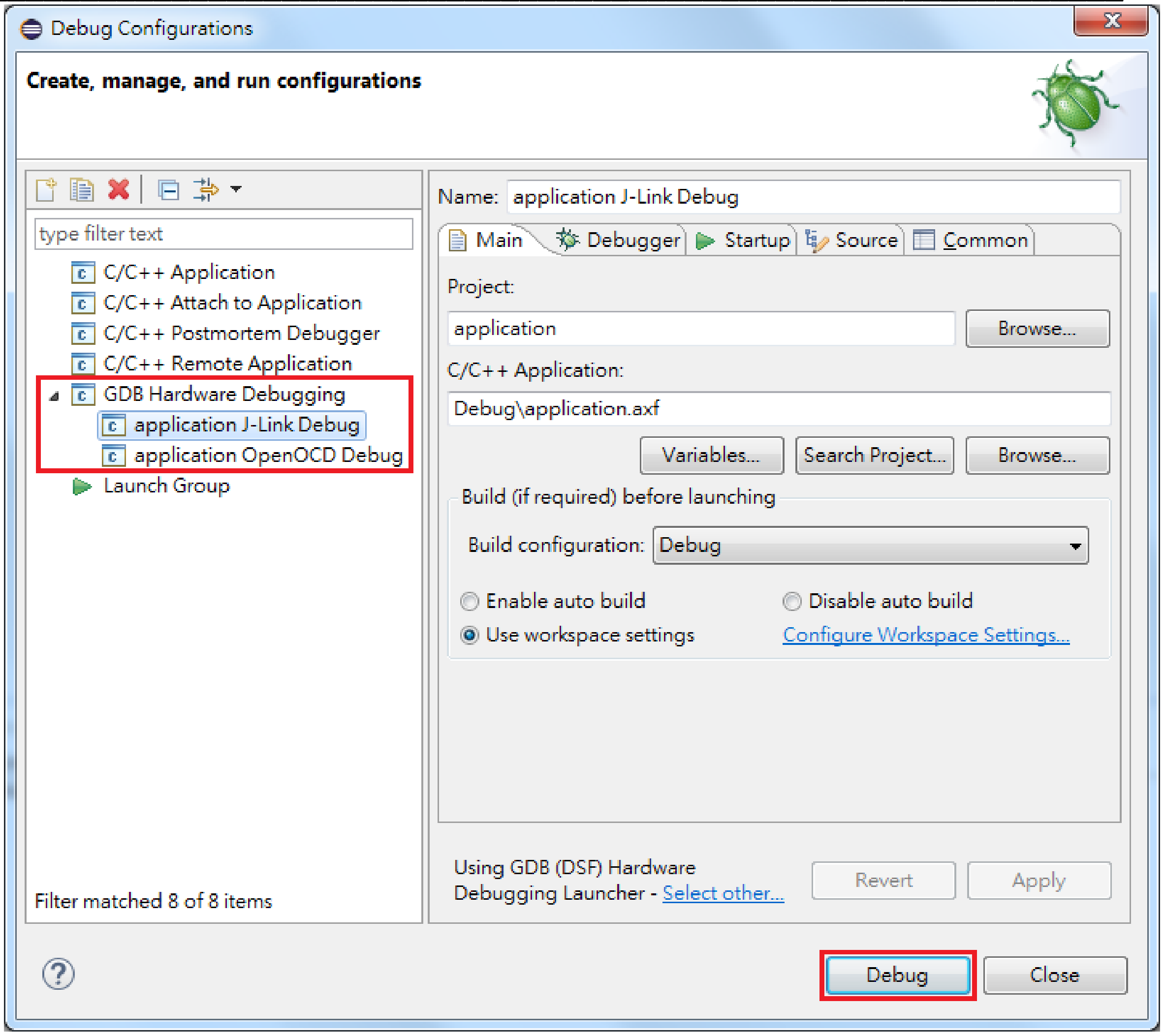
```
===========================================================
Flash Downloader Build Time: # 2016/07/06-00:04:09
===========================================================
EraseMode : 1
FirmwareSize : 340349
Skip calibration
Flash erace done
Flash download start
Flash download done
```

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.


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

Name: application J-Link Debug

Main | Debugger | Startup | Source | Common

Project: application

C/C++ Application: Debug\application.axf

Build (if required) before launching

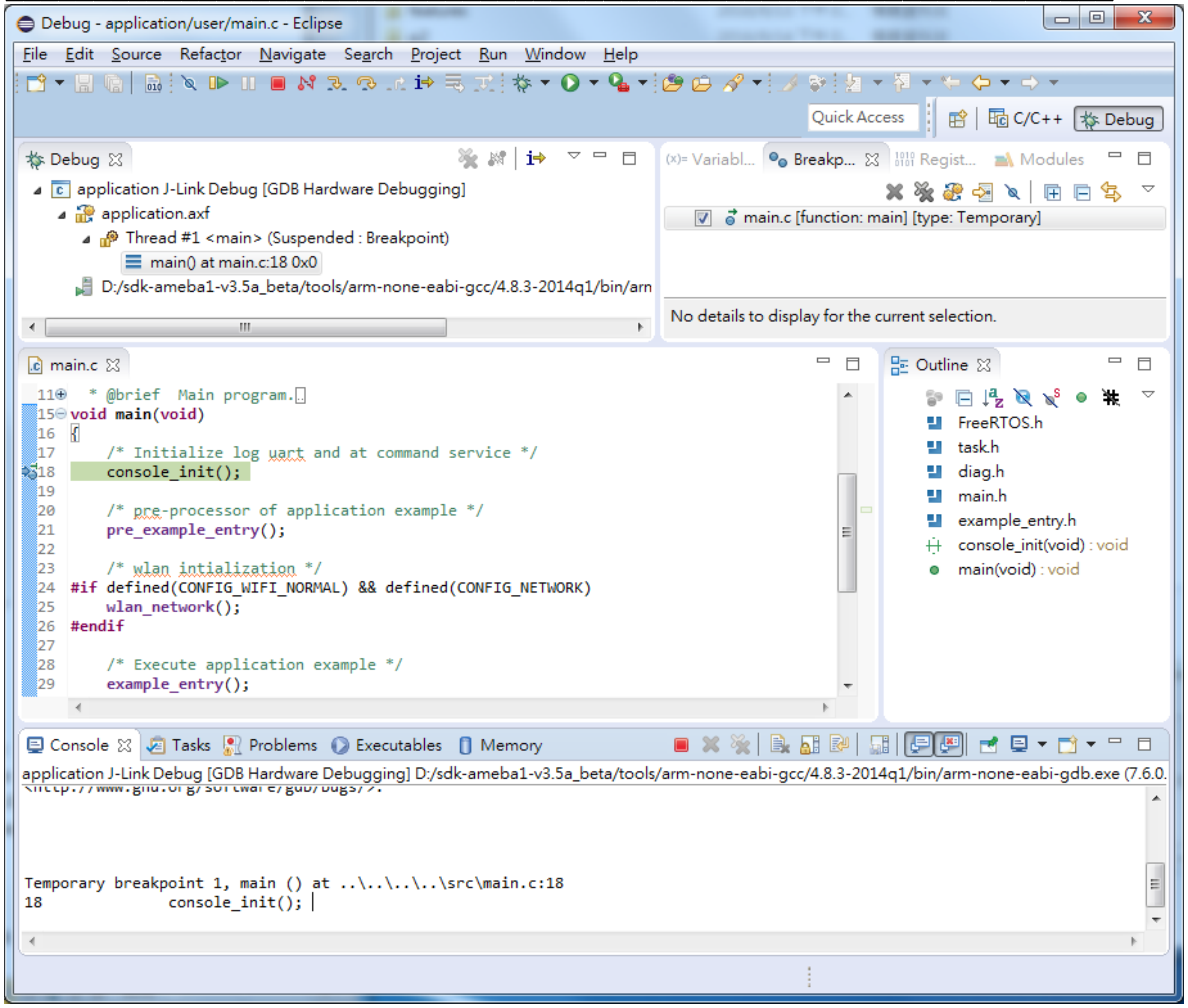
Build configuration: Debug

Enable auto build | Disable auto build

[✓ Use workspace settings] | Configure Workspace Settings...

Using GDB (DSF) Hardware Debugging Launcher - Select other...

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. 4.7 to troubleshoot it.


Debug - application/user/main.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Debug
application J-Link Debug [GDB Hardware Debugging]
application.axf
Thread #1 <main> (Suspended : Breakpoint)
main() at main.c:18 0x0
D:/sdk-ameba1-v3.5a_beta/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arn
Variabl...
Breakp...
Regist...
Modules
main.c [function: main] [type: Temporary]
No details to display for the current selection.
main.c
Outline
FreeRTOS.h
task.h
diag.h
main.h
example_entry.h
console_init(void) : void
main(void) : void
Console
Tasks
Problems
Executables
Memory

```
 * @brief  Main program.
void main(void)
{
    /* Initialize log uart and at command service */
    console_init();

    /* pre-processor of application example */
    pre_example_entry();

    /* wlan intialization */
#if defined(CONFIG_WIFI_NORMAL) && defined(CONFIG_NETWORK)
    wlan_network();
#endif

    /* Execute application example */
    example_entry();
```

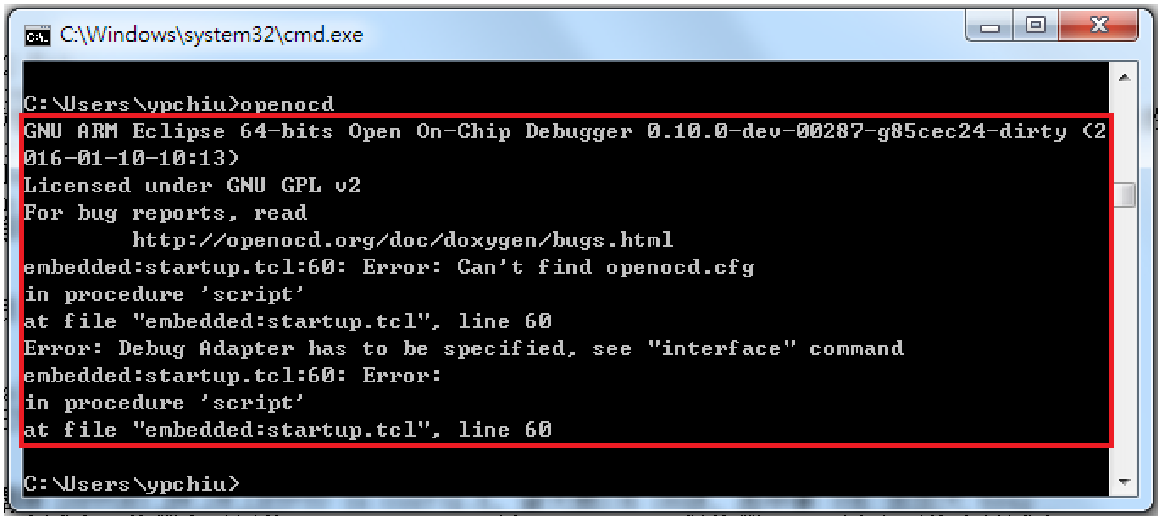
```
application J-Link Debug [GDB Hardware Debugging] D:/sdk-ameba1-v3.5a_beta/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-gdb.exe (7.6.0.
<http://www.gnu.org/software/gdb/bugs/>.



Temporary breakpoint 1, main () at ..\..\..\..\src\main.c:18
18          console_init();
```

4 Troubleshooting

4.1 Unable to execute run_openocd.bat normally on Windows

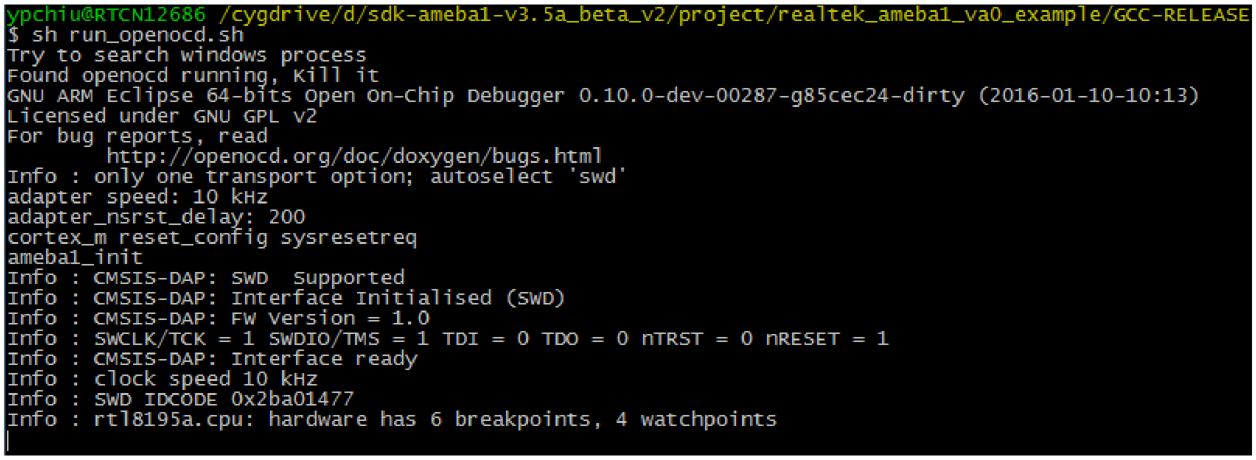
On Windows platform, if you cannot execute run_openocd.bat normally and the pop out window always crash, the reason might be OpenOCD has not been correctly installed or the connection between PC and Ameba has some problem.

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.



```
C:\Windows\system32\cmd.exe

C:\Users\ypchiu>openocd
GNU ARM Eclipse 64-bits Open On-Chip Debugger 0.10.0-dev-00287-g85cec24-dirty (2
016-01-10-10:13)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
embedded:startup.tcl:60: Error: Can't find openocd.cfg
in procedure 'script'
at file "embedded:startup.tcl", line 60
Error: Debug Adapter has to be specified, see "interface" command
embedded:startup.tcl:60: Error:
in procedure 'script'
at file "embedded:startup.tcl", line 60

C:\Users\ypchiu>
```

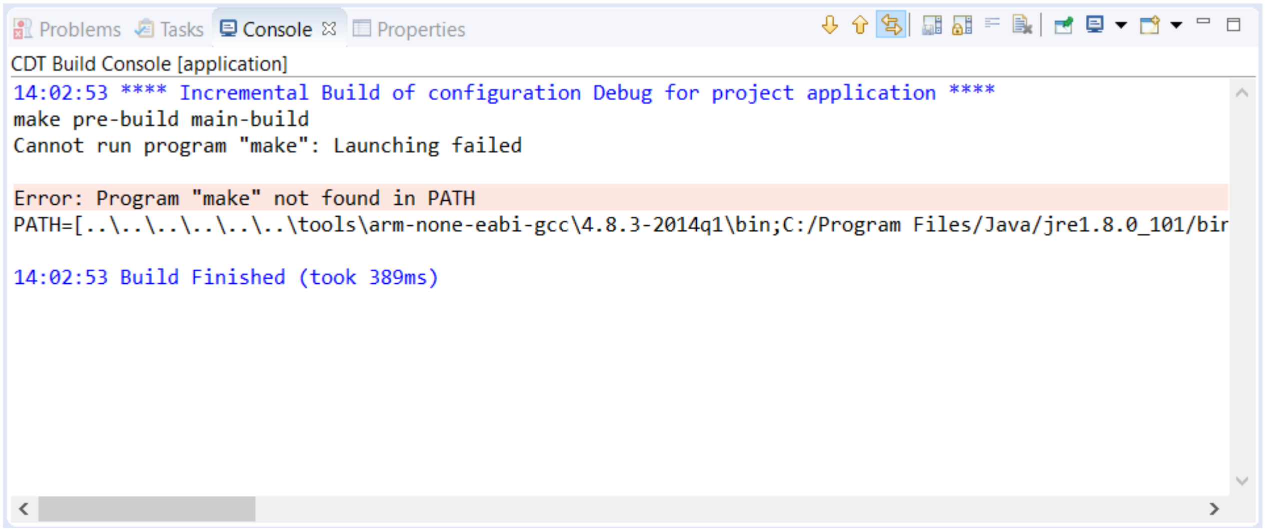
If OpenOCD has been installed correctly but the run_openocd.bat still cannot run normally, you can try to re-plug the connection between PC and Ameba board. You also can try to execute run_openocd.sh rather than run_openocd.bat if you have some Unix-like environment tools, e.g. Cygwin. Below figure illustrates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



```
ypchiu@RTCN12686 /cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
$ sh run_openocd.sh
Try to search windows process
Found openocd running, Kill it
GNU ARM Eclipse 64-bits Open On-Chip Debugger 0.10.0-dev-00287-g85cec24-dirty (2016-01-10-10:13)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'swd'
adapter speed: 10 kHz
adapter_nsrst_delay: 200
cortex_m reset_config sysresetreq
ameba1_init
Info : CMSIS-DAP: SWD  Supported
Info : CMSIS-DAP: Interface Initialised (SWD)
Info : CMSIS-DAP: FW Version = 1.0
Info : SWCLK/TCK = 1 SWDIO/TMS = 1 TDI = 0 TDO = 0 nTRST = 0 nRESET = 1
Info : CMSIS-DAP: Interface ready
Info : clock speed 10 kHz
Info : SWD IDCODE 0x2ba01477
Info : rtl8195a.cpu: hardware has 6 breakpoints, 4 watchpoints
```

4.2 Make not found in PATH error during project building

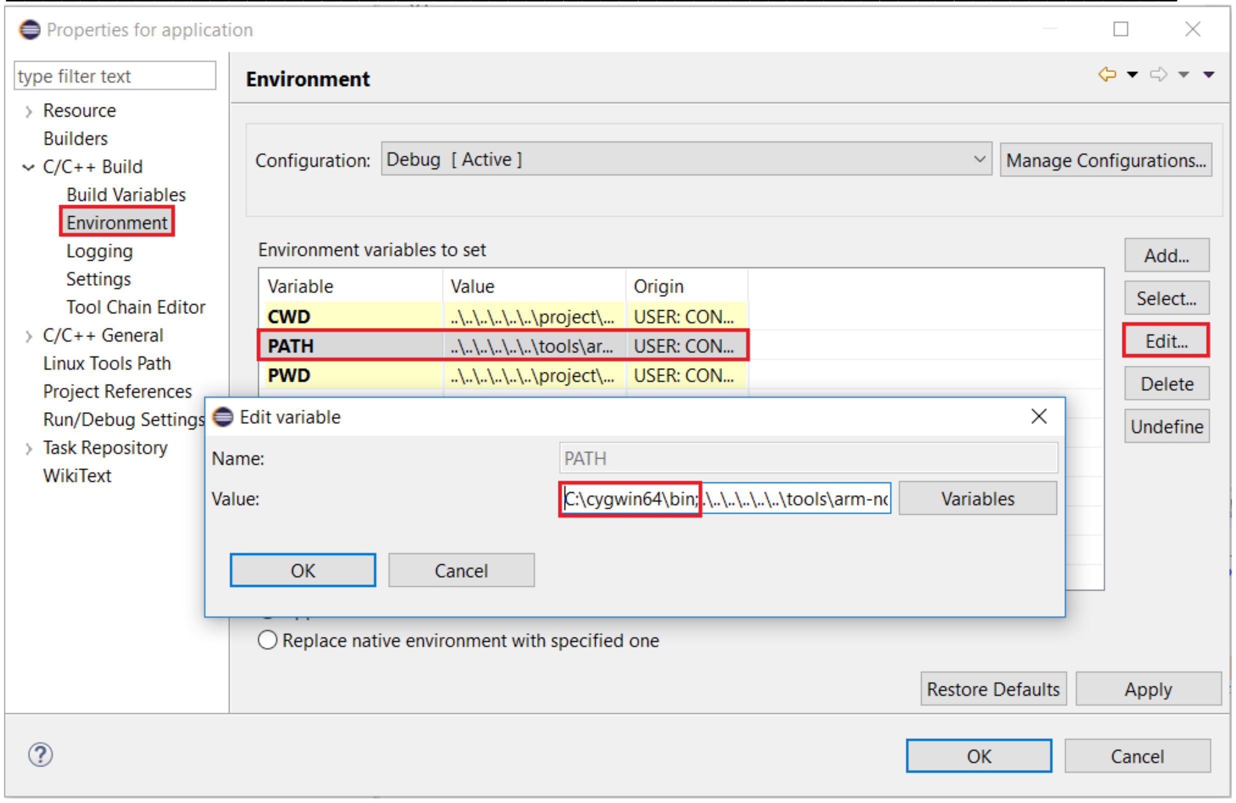
As we described in Sec. 3.1, if you encounter “Error: Program make not found in PATH” for building project means that the Cygwin path is not set correctly for Eclipse.



```
CDT Build Console [application]
14:02:53 **** Incremental Build of configuration Debug for project application ****
make pre-build main-build
Cannot run program "make": Launching failed

Error: Program "make" not found in PATH
PATH=[..\..\..\..\..\..\tools\arm-none-eabi-gcc\4.8.3-2014q1\bin;C:/Program Files/Java/jre1.8.0_101/bir

14:02:53 Build Finished (took 389ms)
```

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.



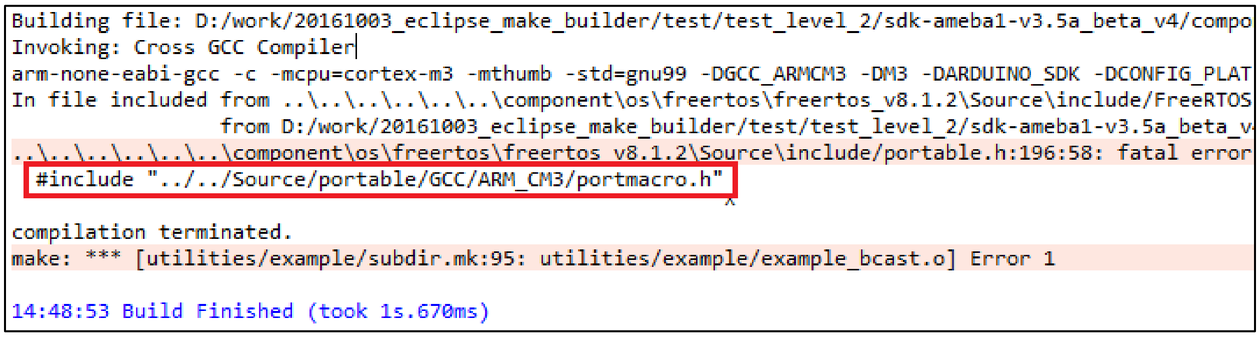
Variable	Value	Origin
CWD	..\..\..\..\..\project\...	USER: CON...
PATH	..\..\..\..\..\tools\ar...	USER: CON...
PWD	..\..\..\..\..\project\...	USER: CON...

Name: PATH

Value: C:\cygwin64\bin;..\..\..\..\..\tools\arm-nc

4.3 Build fail due to relative path header file not found

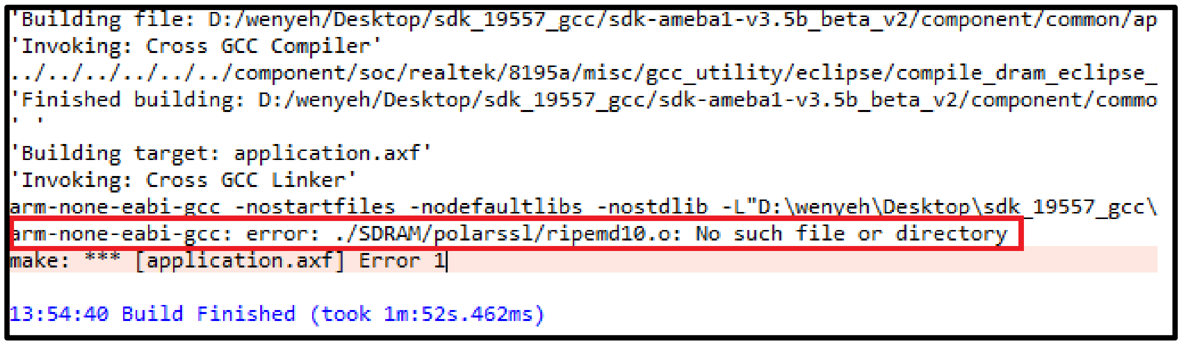
If the build project procedure terminates due the some header file not found and the header file is included with relative path, it may be caused by the long project path. The length of include path may exceed the maximum limit. For example, if you encounter a fatal error like: "fatal error: ../Source/portable/GCC/ARM_CM3/portmacro.h: 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 Eclipse link error

If the build project procedure terminates due to link procedure, and it is reported that “No such file or directory” in one file, which actually exists and just in lack of one character, it means that the Cygwin path is not set correctly for Eclipse or the “make” function you use is wrong version.



```
'Building file: D:/wenyeh/Desktop/sdk_19557_gcc/sdk-ameba1-v3.5b_beta_v2/component/common/ap
'Invoking: Cross GCC Compiler'
../../../../../../component/soc/realtek/8195a/misc/gcc_utility/eclipse/compile_dram_eclipse_
'Finished building: D:/wenyeh/Desktop/sdk_19557_gcc/sdk-ameba1-v3.5b_beta_v2/component/commo
' '
'Building target: application.axf'
'Invoking: Cross GCC Linker'
arm-none-eabi-gcc -nostartfiles -nodefaultlibs -nostdlib -L"D:\wenyeh\Desktop\sdk_19557_gcc\
arm-none-eabi-gcc: error: ./SDRAM/polarssl/ripemd10.o: No such file or directory
make: *** [application.axf] Error 1

13:54:40 Build Finished (took 1m:52s.462ms)
```

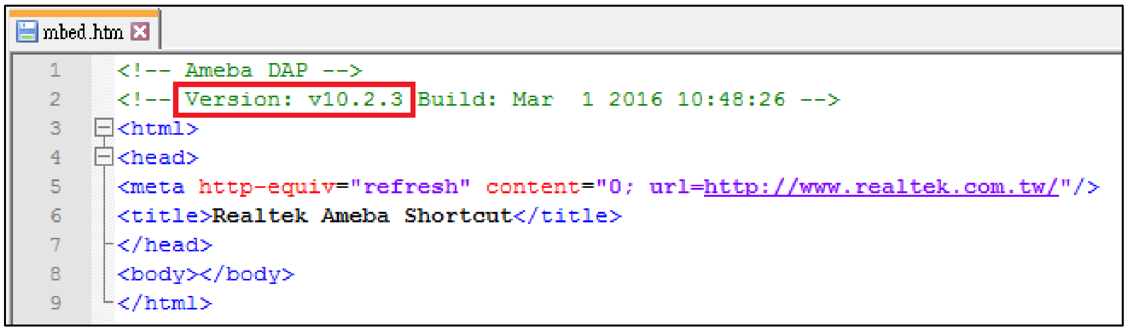
The reason of this problem is that calling wrong "make" will trigger cmd.exe in Windows. However, Using "make" in Cygwin as our document said can avoid this error because in Cygwin sh.exe will dominate instead of cmd.exe.

Therefore, it will be helpful to check that their Cygwin works correctly. In eclispe, go to Project> Properties> C/C++ Build> Environment. You will see three fields, choose PATH. See if the PATH containing Cygwin/bin inside as Section 4.2 said. It is required that the PATH should not contain GnuWin, MinGW... that may have other "make".

It is noted that sometimes the change to the System PATH variable (made from My Computer> Properties> Advanced System Settings...) is NOT reflected in Eclipse, so eclipse should be checked correctly.

4.5 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.

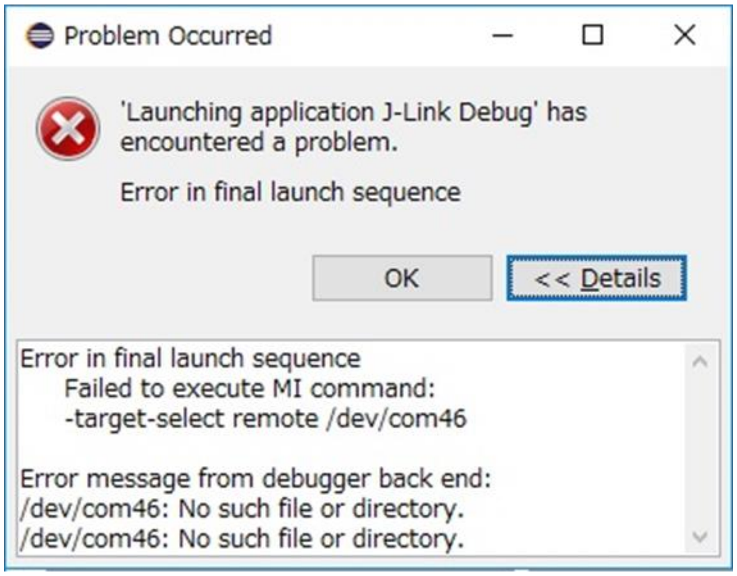


```
<!-- Ameba DAP -->
<!-- Version: v10.2.3 Build: Mar  1 2016 10:48:26 -->
<html>
<head>
<meta http-equiv="refresh" content="0; url=http://www.realtek.com.tw/"/>
<title>Realtek Ameba Shortcut</title>
</head>
<body></body>
</html>
```

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.

4.6 Target error for J-Link debug

In Sec. 3.3 we have described how to enter debug mode under Eclipse + GDB environment. If you start the "application J-Link Debug" but encounter some message which indicate target not found error such as:

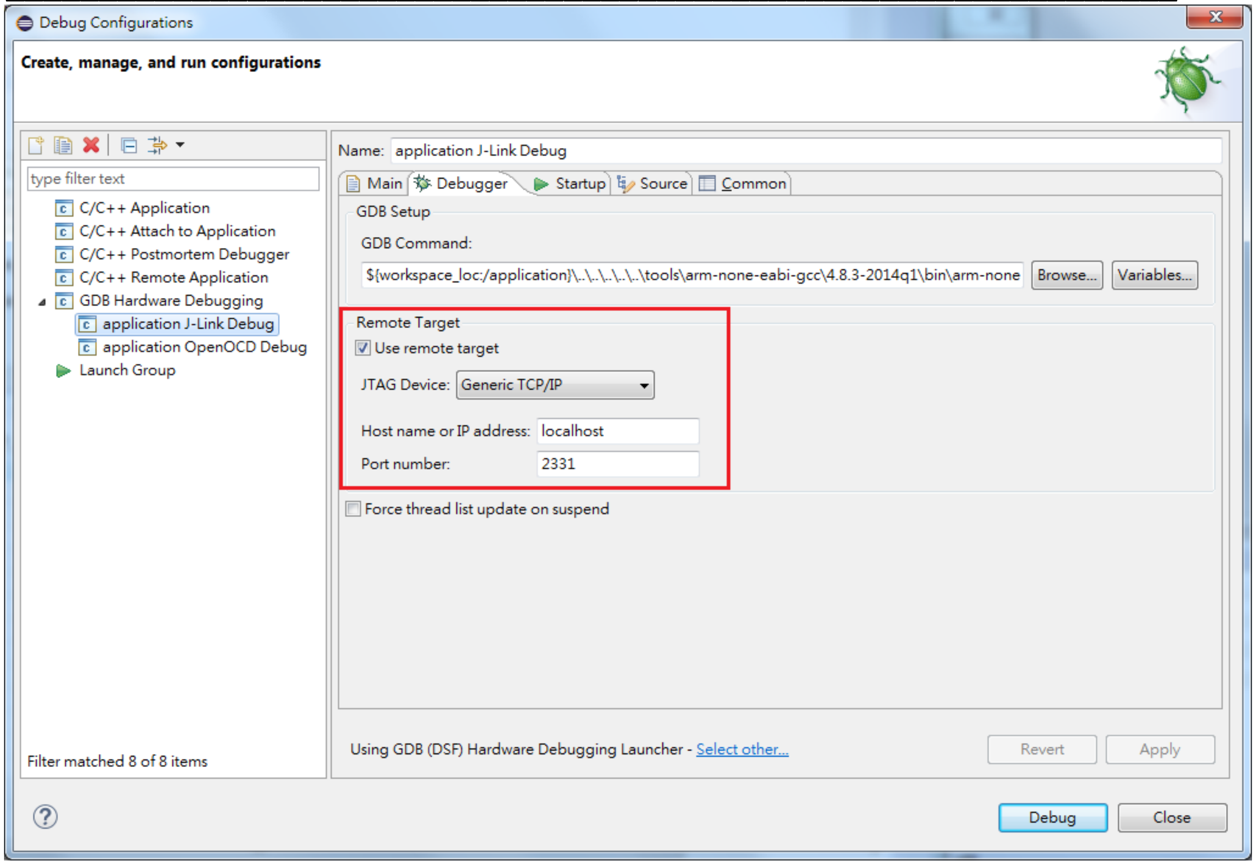


'Launching application J-Link Debug' has encountered a problem.

Error in final launch sequence

```
Error in final launch sequence
    Failed to execute MI command:
    -target-select remote /dev/com46

Error message from debugger back end:
/dev/com46: No such file or directory.
/dev/com46: No such file or directory.
```

Then you can open the debug configuration for "application J-Link Debug" and check whether the remote target setting is correct. To open the debug configuration, click "Run -> Debug Configurations…" on the top of Eclipse IDE and then choose "application J-Link Debug" under "GDB Hardware Debugging" entry in the left side of popped out window. Navigate to "Debugger" tag and check the "Remote Target" setting:



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

Name: application J-Link Debug

Main | Debugger | Startup | Source | Common

GDB Setup

GDB Command: ${workspace_loc:/application}\..\..\..\..\tools\arm-none-eabi-gcc\4.8.3-2014q1\bin\arm-none

Remote Target

[✓] Use remote target

JTAG Device: Generic TCP/IP

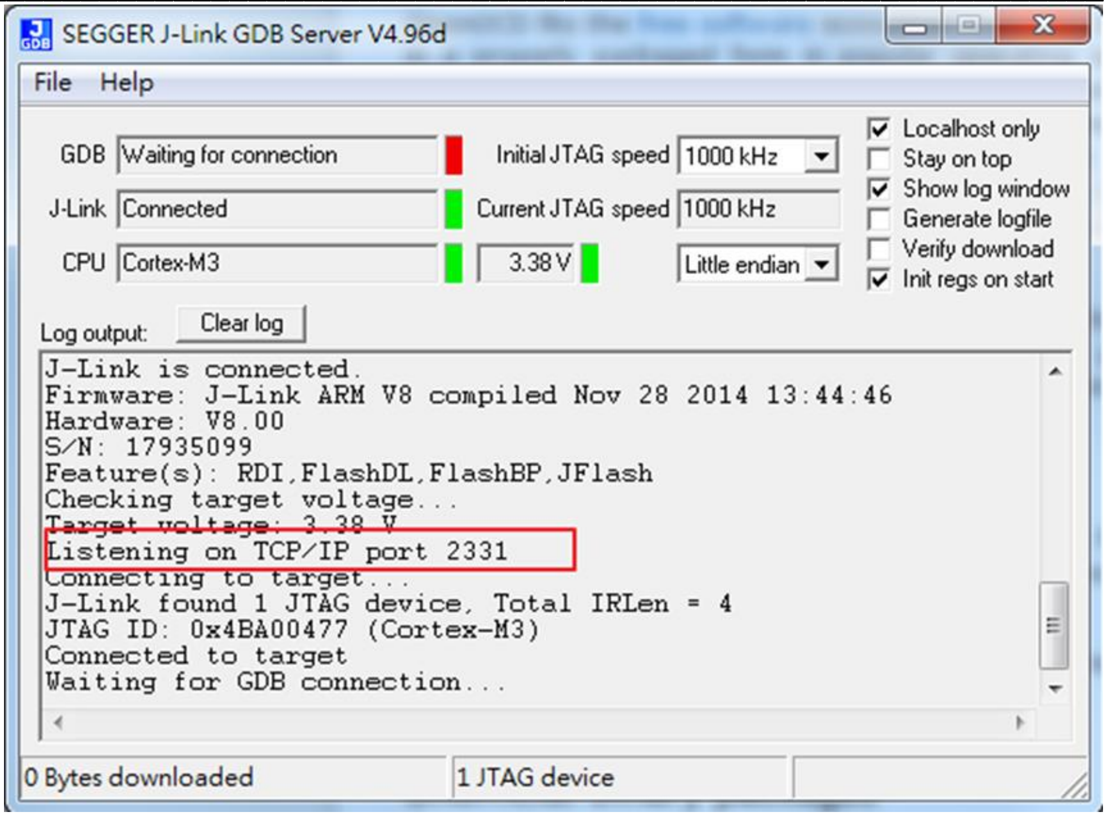
Host name or IP address: localhost

Port number: 2331

[ ] Force thread list update on suspend

Using GDB (DSF) Hardware Debugging Launcher - Select other...

The remote target should be set as same as above red region area, which is identical to GDB server listening port:



GDB: Waiting for connection
J-Link: Connected
CPU: Cortex-M3
Initial JTAG speed: 1000 kHz
Current JTAG speed: 1000 kHz
3.38 V
Little endian
[✓] Localhost only
[ ] Stay on top
[✓] Show log window
[ ] Generate logfile
[ ] Verify download
[✓] Init regs on start

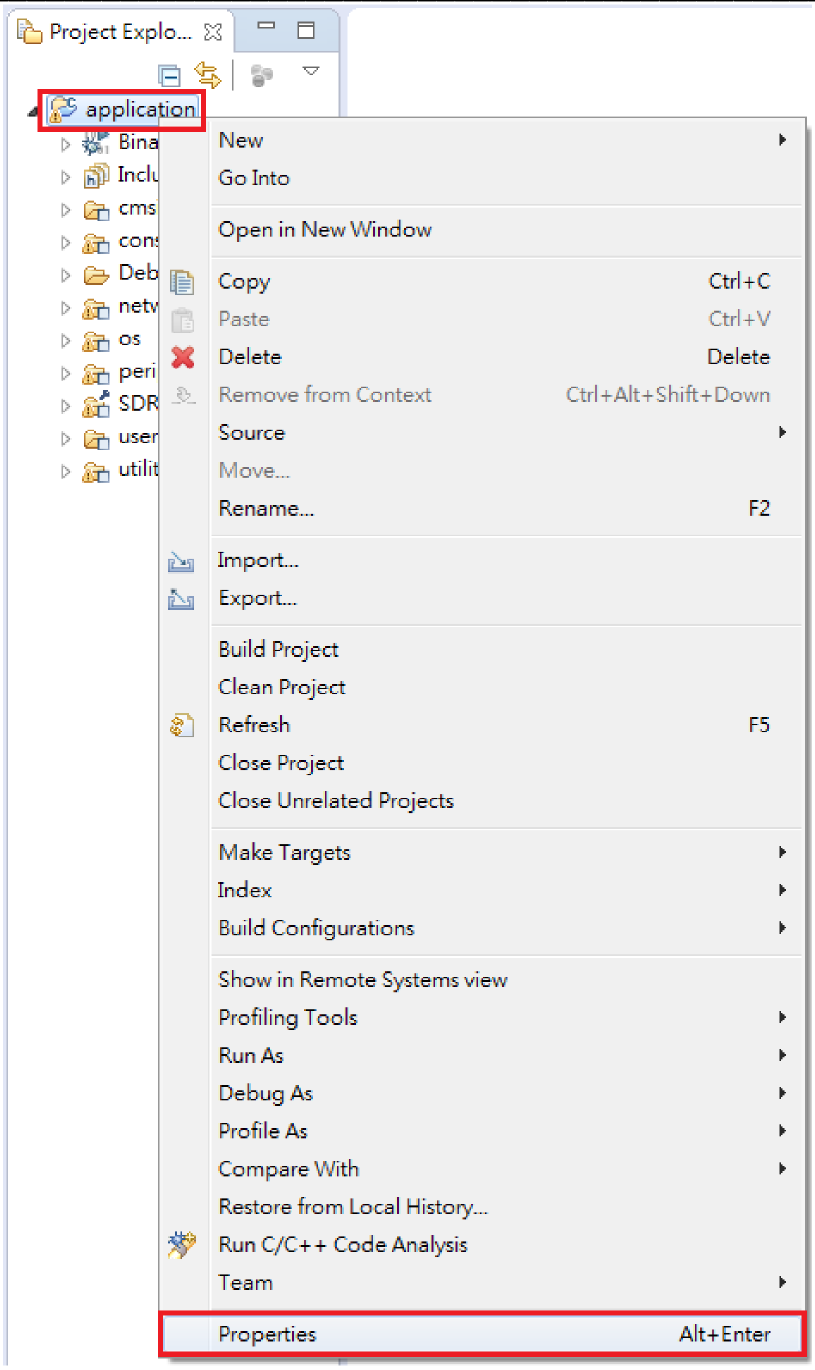
Log output:

```
J-Link is connected.
Firmware: J-Link ARM V8 compiled Nov 28 2014 13:44:46
Hardware: V8.00
S/N: 17935099
Feature(s): RDI,FlashDL,FlashBP,JFlash
Checking target voltage...
Target voltage: 3.38 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 1 JTAG device, Total IRLen = 4
JTAG ID: 0x4BA00477 (Cortex-M3)
Connected to target
Waiting for GDB connection...
```

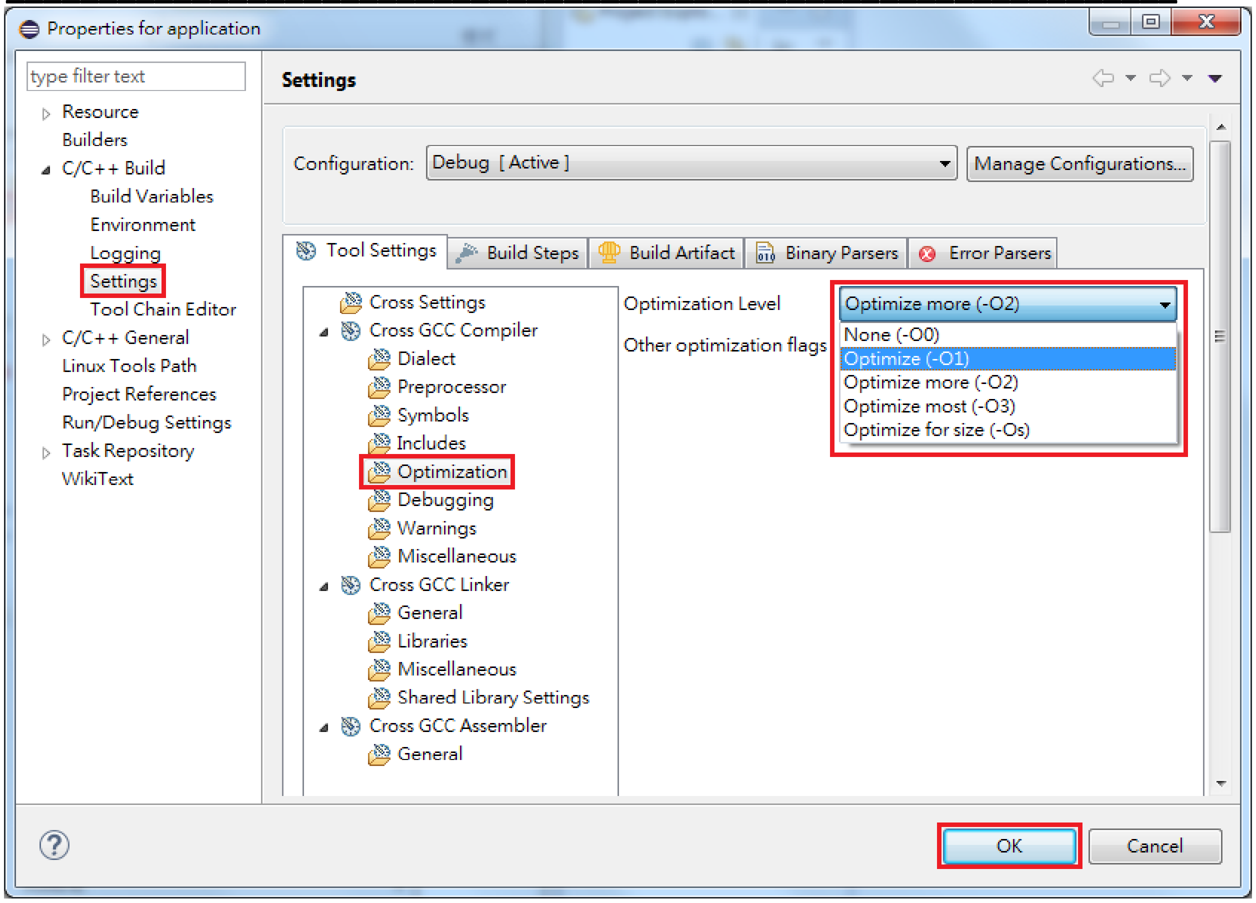
0 Bytes downloaded | 1 JTAG device

4.7 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: Optimize more (-O2)

Other optimization flags

- None (-O0)
- Optimize (-O1)
- Optimize more (-O2)
- Optimize most (-O3)
- Optimize for size (-Os)

OK   Cancel