



Build Environment Setup - GCC

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

Table of Contents

1	Introduction	4
2	How to get GCC environment	4
2.1	Windows	4
2.2	Linux.....	6
3	How to build and download code.....	7
3.1	Build code.....	8
3.2	Debugger setting.....	9
3.2.1	OpenOCD/CMSIS-DAP.....	9
3.2.1.1	Windows	9
3.2.1.2	Linux.....	12
3.2.2	Jlink	16
3.2.2.1	Windows	16
3.2.2.2	Linux.....	18
3.3	Download code to flash	21
3.4	Enter GDB debugger	22
3.5	Download and debug in RAM	22
4	Command list	23
5	GDB debugger basic usage.....	24
5.1	Stop and continue	24
5.1.1	Breakpoint.....	24
5.1.2	Watchpoint	24
5.1.3	Print breakpoints and watchpoints	25
5.1.4	Delete breakpoints.....	25
5.1.5	Continue.....	25
5.1.6	Step	26
5.1.7	Next.....	26
5.1.8	Quit	26
5.2	Examine stack, source file and data.....	26

5.2.1	Backtrace.....	26
5.2.2	Print source lines.....	27
5.2.3	Examine data.....	28
6	Troubleshooting	28
6.1	Unable to execute run_openocd.bat normally on Windows	28
6.2	How to install the newest version of OpenOCD on Ubuntu	29
6.3	Download procedure hang for a long time.....	32

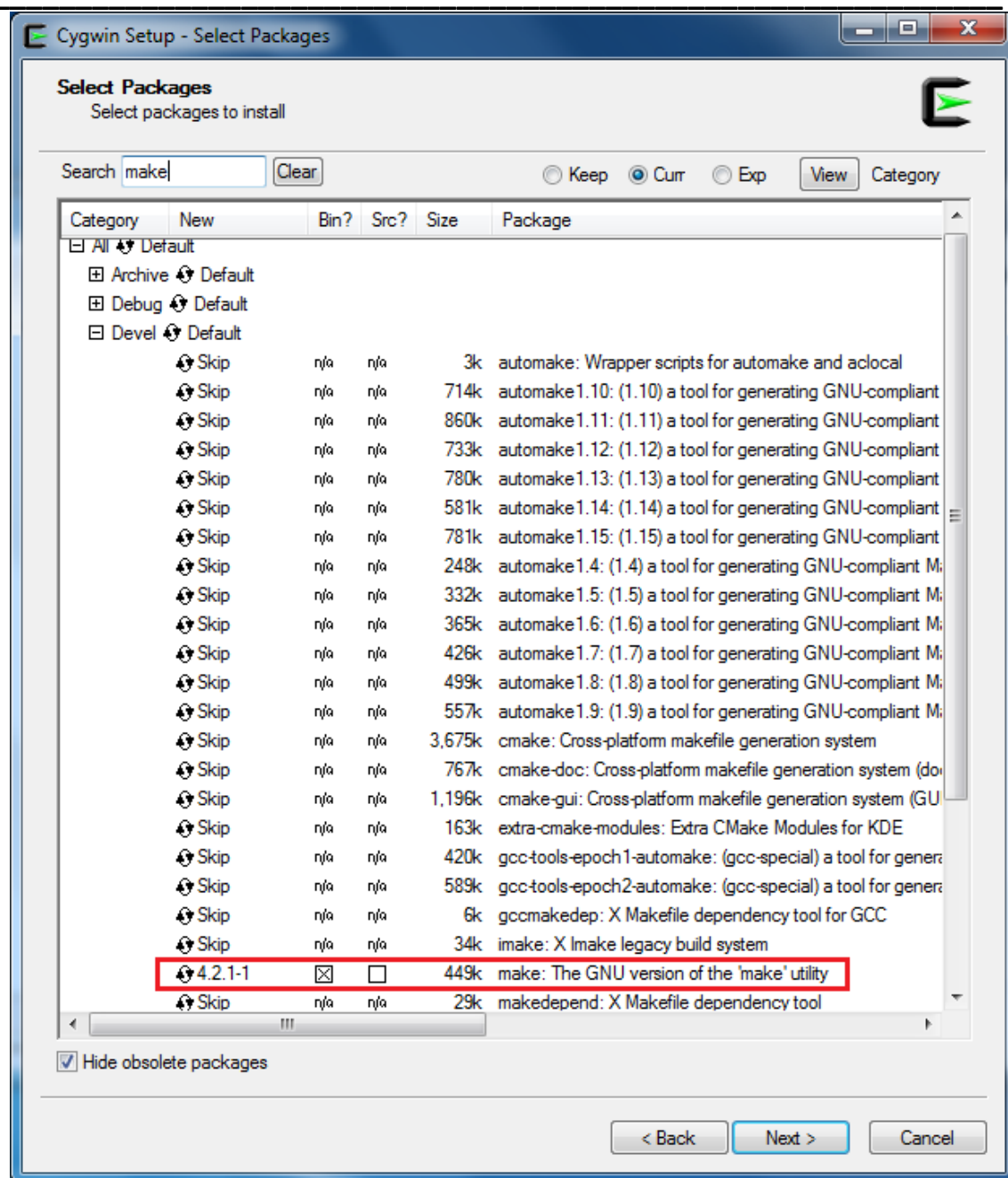
1 Introduction

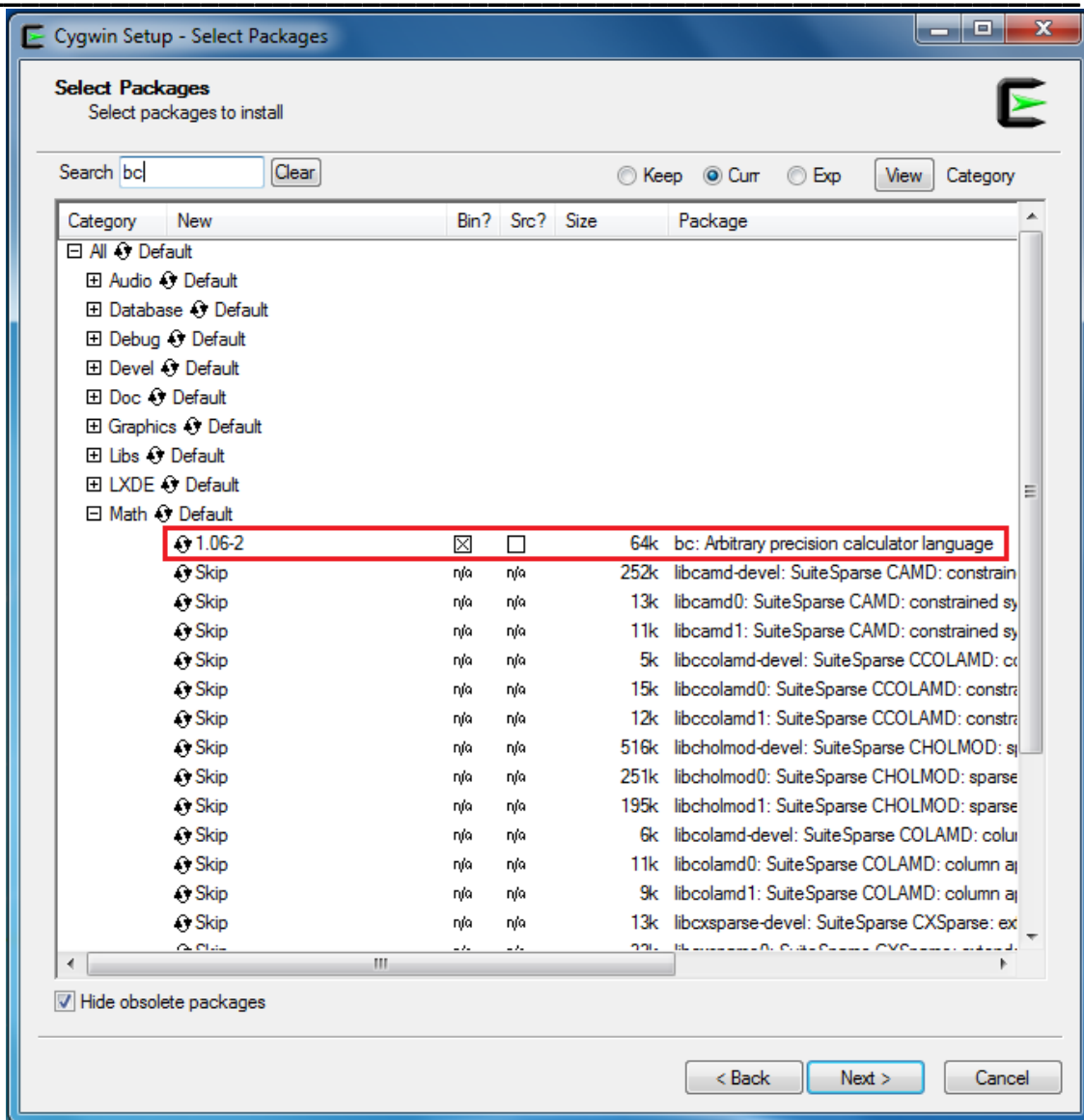
This document illustrates how to build Realtek Wi-Fi SDK under GCC environment. We will focus on both Windows platform and Linux distribution in this document. For Windows, we use Windows 7 64-bit as our platform. And for Linux distribution, we use Ubuntu 16.04 64-bit as our platform. Note that the build and download procedure are quite similar between Windows and Linux operating system.

2 How to get GCC environment

2.1 Windows

On Windows, you can use Cygwin as the GCC environment. 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, you should include '**Devel -> make**' and '**Math -> bc**' utilities on the Select Packages step:





2.2 Linux

On Linux, there are some packages should be installed for our GCC environment. The packages include **libc6-i386** (GNU C library), **lib32ncurses5** (32-bit terminal handling. If you are using 32-bit platform, install **libncurses5** instead), **make**, **bc**, and **gawk**. Some of these packages might have been pre-installed in your operating system. Please use package manager to check and install them. And for the last three packages, you can also type its corresponding version command on terminal like below figures to check whether it existed. If not, please make these packages installed.

\$ make -v

```
realtek@realtek-VirtualBox:~$ make -v
GNU Make 4.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

\$ bc -v

```
realtek@realtek-VirtualBox:~$ bc -v
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
```

\$ gawk --v

```
realtek@realtek-VirtualBox:~$ gawk --v
GNU Awk 4.1.3, API: 1.1 (GNU MPFR 3.1.4, GNU MP 6.1.0)
Copyright (C) 1989, 1991-2015 Free Software Foundation.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see http://www.gnu.org/licenses/.
```

3 How to build and download code

In this section, we illustrate how to build, download, and enter GDB debugger mode. First, we need to switch to gcc project directory.

For **Windows**, please open Cygwin terminal and use `cd` command to change directory to GCC-RELEASE/ directory of SDK. Note that you need to add “cygdrive” prefix in front of the SDK location so that Cygwin can access your file system:

```
$ cd /cygdrive/SDK_LOC/project/realtek_ameba1_va0_example/GCC-RELEASE
```

For **Linux**, open its own terminal and use `cd` command to change directory to GCC-RELEASE/ directory of SDK:

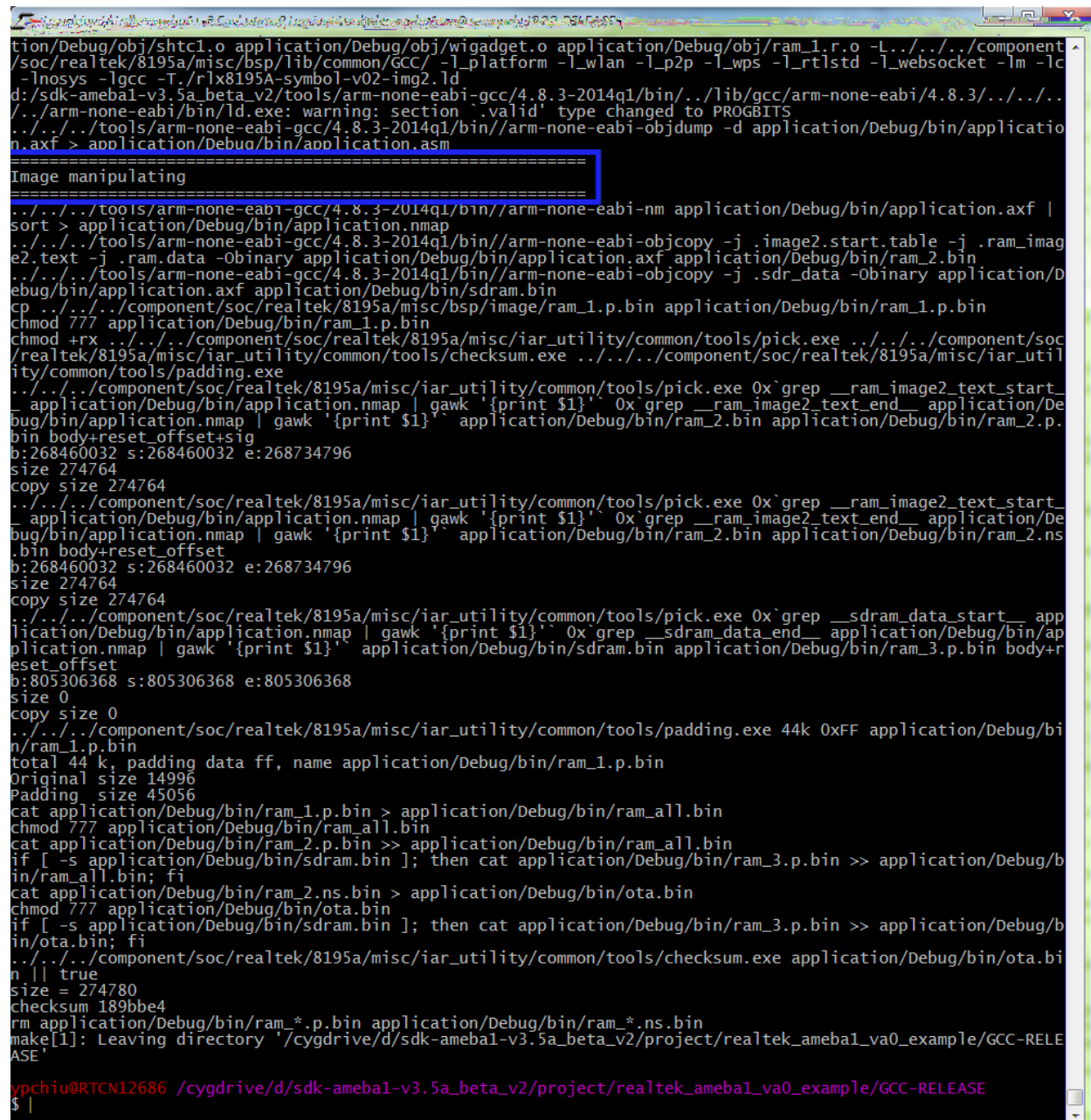
```
$ cd /SDK_LOC/project/realtek_ameba1_va0_example/GCC-RELEASE
```

3.1 Build code

To build the SDK, simply use *make* command under GCC-RELEASE/ directory on Cygwin (Windows) or terminal (Linux):

```
$ make
```

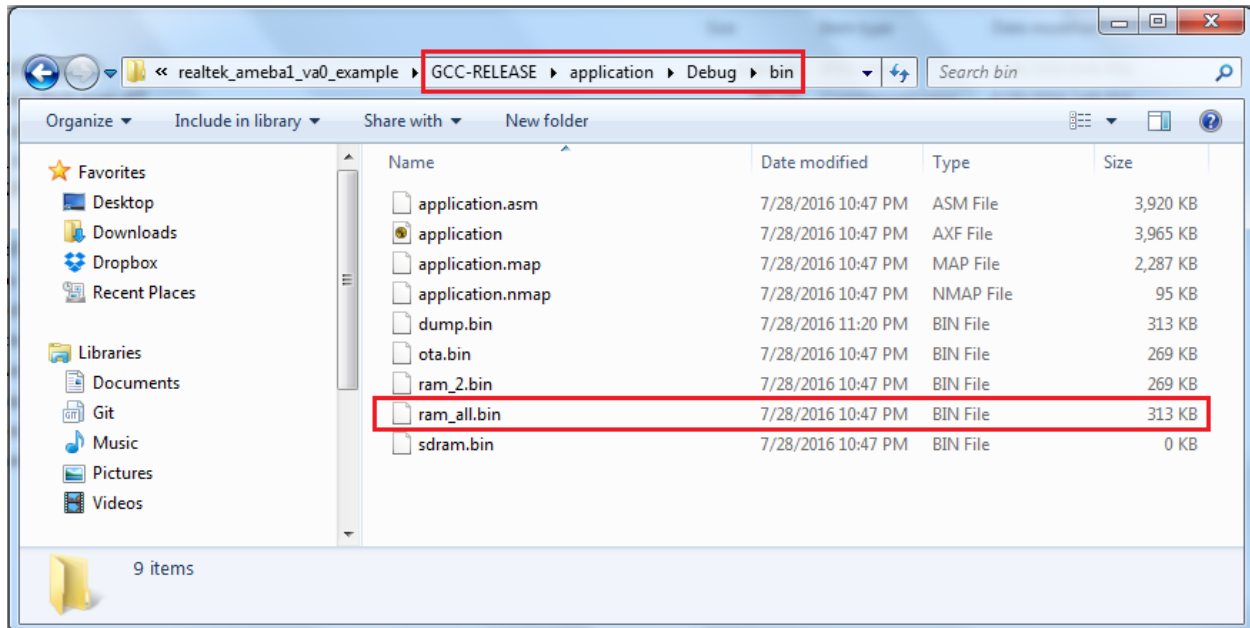
If the terminal contains “Image manipulating” output message means that the image has been built successfully.



```
tion/Debug/obj/shtc1.o application/Debug/obj/wigadget.o application/Debug/obj/ram_1.r.o -L../component
/soc/realtek/8195a/misc/bsp/lib/common/GCC/ -l_platform -l_wlan -l_p2p -l_wps -l_rtlstd -l_websocket -lm -lc
-lnosys -lgcc -T./rlx8195A-symbol-v02-img2.ld
d:/sdk-ameba1-v3.5a_beta_v2/tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/./lib/gcc/arm-none-eabi/4.8.3/./../..
./arm-none-eabi/bin/ld.exe: warning: section '.valid' type changed to PROGBITS
./../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-objdump -d application/Debug/bin/applicatio
n.axf > application/Debug/bin/application.asm
Image manipulating
./../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-nm application/Debug/bin/application.axf |
sort > application/Debug/bin/application.nmap
./../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-objcopy -j .image2.start.table -j .ram_imag
e2.text -j .ram.data -Obinary application/Debug/bin/application.axf application/Debug/bin/ram_2.bin
./../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-objcopy -j .sdr_data -Obinary application/D
ebug/bin/application.axf application/Debug/bin/sdram.bin
cp ./../component/soc/realtek/8195a/misc/bsp/image/ram_1.p.bin application/Debug/bin/ram_1.p.bin
chmod 777 application/Debug/bin/ram_1.p.bin
chmod +rx ./../component/soc/realtek/8195a/misc/iar_utility/common/tools/pick.exe ./../component/soc
/realtek/8195a/misc/iar_utility/common/tools/checksum.exe ./../component/soc/realtek/8195a/misc/iar_util
ity/common/tools/padding.exe
./../component/soc/realtek/8195a/misc/iar_utility/common/tools/pick.exe 0x`grep __ram_image2_text_start_
application/Debug/bin/application.nmap | gawk '{print $1}' 0x`grep __ram_image2_text_end__ application/De
bug/bin/application.nmap | gawk '{print $1}' application/Debug/bin/ram_2.bin application/Debug/bin/ram_2.p.
bin body+reset_offset+sig
b:268460032 s:268460032 e:268734796
size 274764
copy size 274764
./../component/soc/realtek/8195a/misc/iar_utility/common/tools/pick.exe 0x`grep __ram_image2_text_start_
application/Debug/bin/application.nmap | gawk '{print $1}' 0x`grep __ram_image2_text_end__ application/De
bug/bin/application.nmap | gawk '{print $1}' application/Debug/bin/ram_2.bin application/Debug/bin/ram_2.ns
.bin body+reset_offset
b:268460032 s:268460032 e:268734796
size 274764
copy size 274764
./../component/soc/realtek/8195a/misc/iar_utility/common/tools/pick.exe 0x`grep __sdram_data_start__ app
lication/Debug/bin/application.nmap | gawk '{print $1}' 0x`grep __sdram_data_end__ application/Debug/bin/ap
plication.nmap | gawk '{print $1}' application/Debug/bin/sdram.bin application/Debug/bin/ram_3.p.bin body+r
eset_offset
b:805306368 s:805306368 e:805306368
size 0
copy size 0
./../component/soc/realtek/8195a/misc/iar_utility/common/tools/padding.exe 44k 0xFF application/Debug/bi
n/ram_1.p.bin
total 44 k, padding data ff, name application/Debug/bin/ram_1.p.bin
Original size 14996
Padding size 45056
cat application/Debug/bin/ram_1.p.bin > application/Debug/bin/ram_all.bin
chmod 777 application/Debug/bin/ram_all.bin
cat application/Debug/bin/ram_2.p.bin >> application/Debug/bin/ram_all.bin
if [ -s application/Debug/bin/sdram.bin ]; then cat application/Debug/bin/ram_3.p.bin >> application/Debug/b
in/ram_all.bin; fi
cat application/Debug/bin/ram_2.ns.bin > application/Debug/bin/ota.bin
chmod 777 application/Debug/bin/ota.bin
if [ -s application/Debug/bin/sdram.bin ]; then cat application/Debug/bin/ram_3.p.bin >> application/Debug/b
in/ota.bin; fi
./../component/soc/realtek/8195a/misc/iar_utility/common/tools/checksum.exe application/Debug/bin/ota.bi
n || true
size = 274780
checksum 189bbe4
rm application/Debug/bin/ram_*.bin application/Debug/bin/ram_*.ns.bin
make[1]: Leaving directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELE
ASE'
ypchuiu@RTCN12686 /cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
$ |
```


If somehow it built failed, you can try to type *\$make clean* and then redo the make procedure.

After successfully build, there should be a directory named “application” created under GCC-RELEASE/ directory. The image file is located in application/Debug/bin/:



3.2 Debugger setting

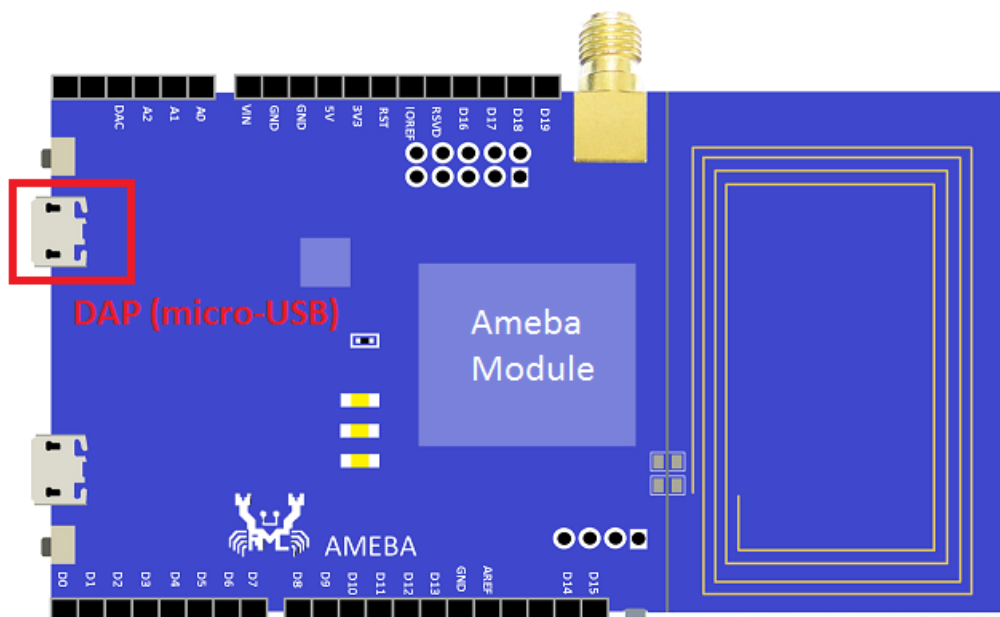
Ameba Device Board supports CMSIS-DAP and J-Link for code download and enter debugger mode with GCC. The settings for these two different debuggers are described below.

3.2.1 OpenOCD/CMSIS-DAP

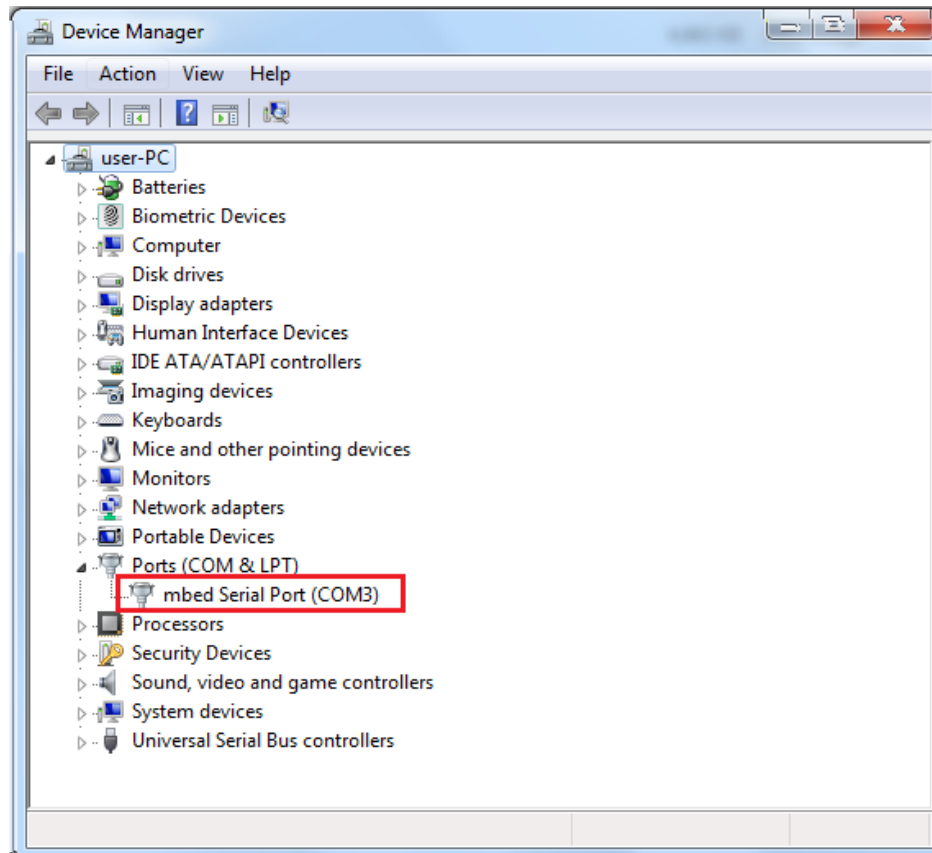
3.2.1.1 Windows

Ameba Device Board supports CMSIS-DAP debugger. We can use OpenOCD/CMSIS-DAP to download the software and enter GDB debugger mode under GCC 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.

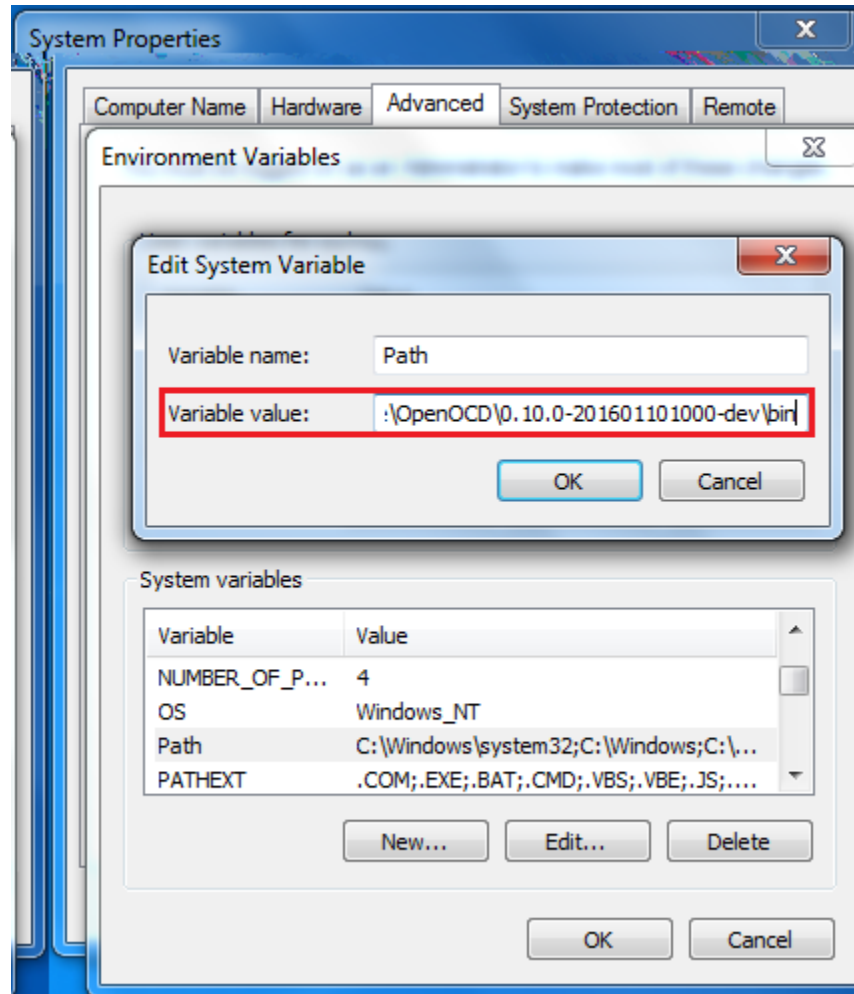
Connect board to the PC with micro-USB cable:



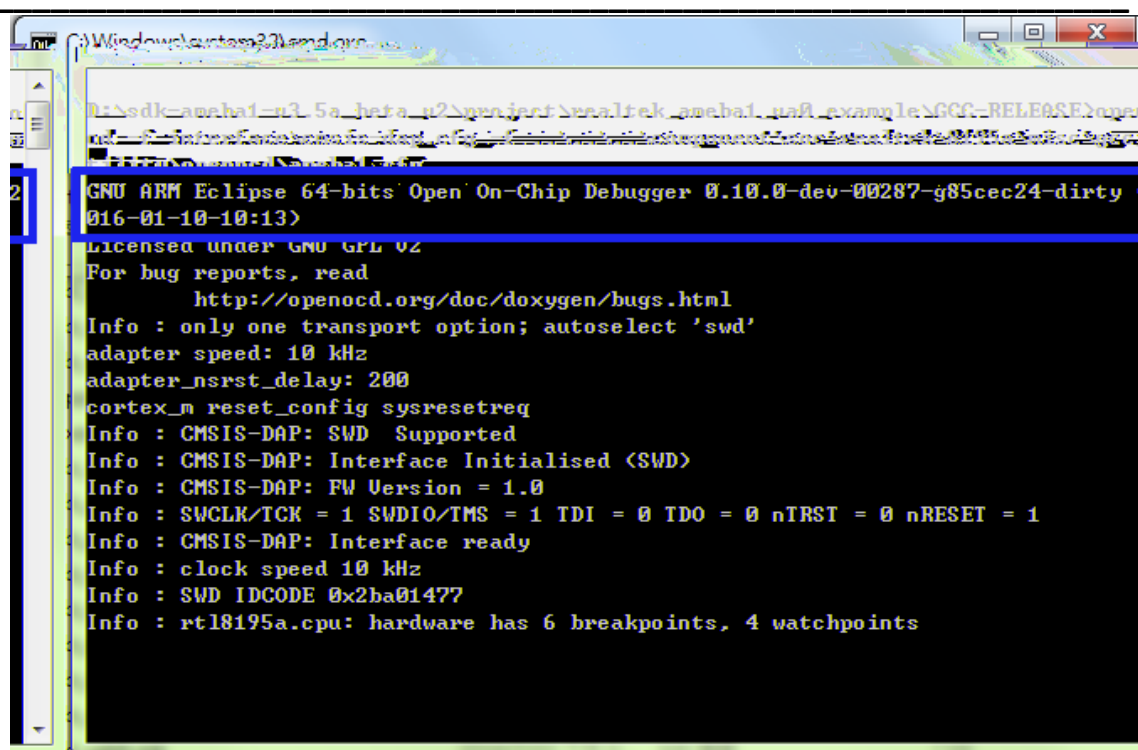
If Serial to USB driver has been installed and the board is connected to PC, there should be mbed Serial Port shown in Device Manager.



It also requires installing OpenOCD on your platform. Please check <http://openocd.org> to get the binary package (<https://github.com/gnuarmclipse/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).



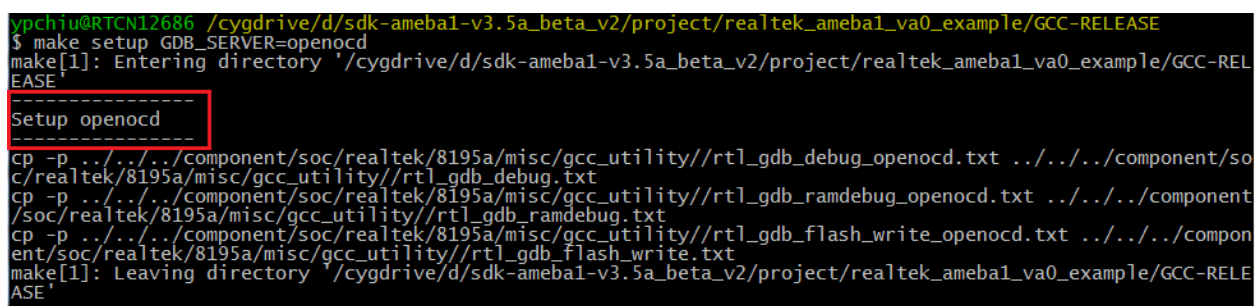
If OpenOCD has been installed correctly, execute GCC-RELEASE/run_openocd.bat to start GDB server and you should see some messages like below figure. This window should **NOT** be closed if you want to download software or enter GDB debugger. (Note that you also can execute run_openocd.sh script on Cygwin terminal rather than execute run_openocd.bat batch file.)



```
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
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
```

On the Cygwin terminal you should type below command before you using OpenOCD/CMSIS-DAP to download software or enter GDB debugger:

```
$ make setup GDB_SERVER=openocd
```

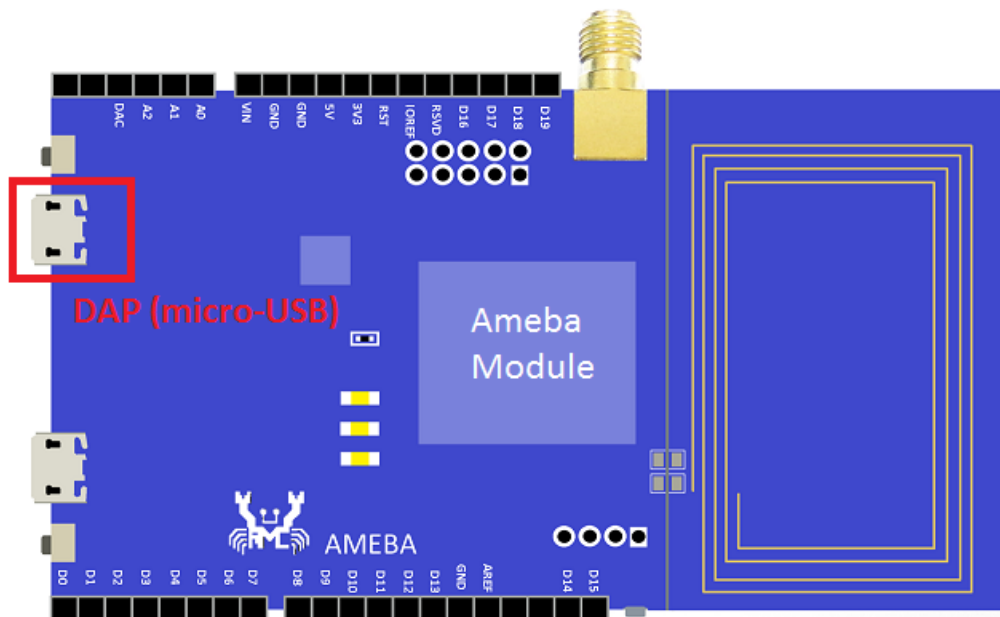


```
ypchui@RTCN12686 /cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
$ make setup GDB_SERVER=openocd
make[1]: Entering directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE'
Setup openocd
cp -p ../../../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_debug_openocd.txt ../../../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_debug.txt
cp -p ../../../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_ramdebug_openocd.txt ../../../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_ramdebug.txt
cp -p ../../../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_flash_write_openocd.txt ../../../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_flash_write.txt
make[1]: Leaving directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE'
```

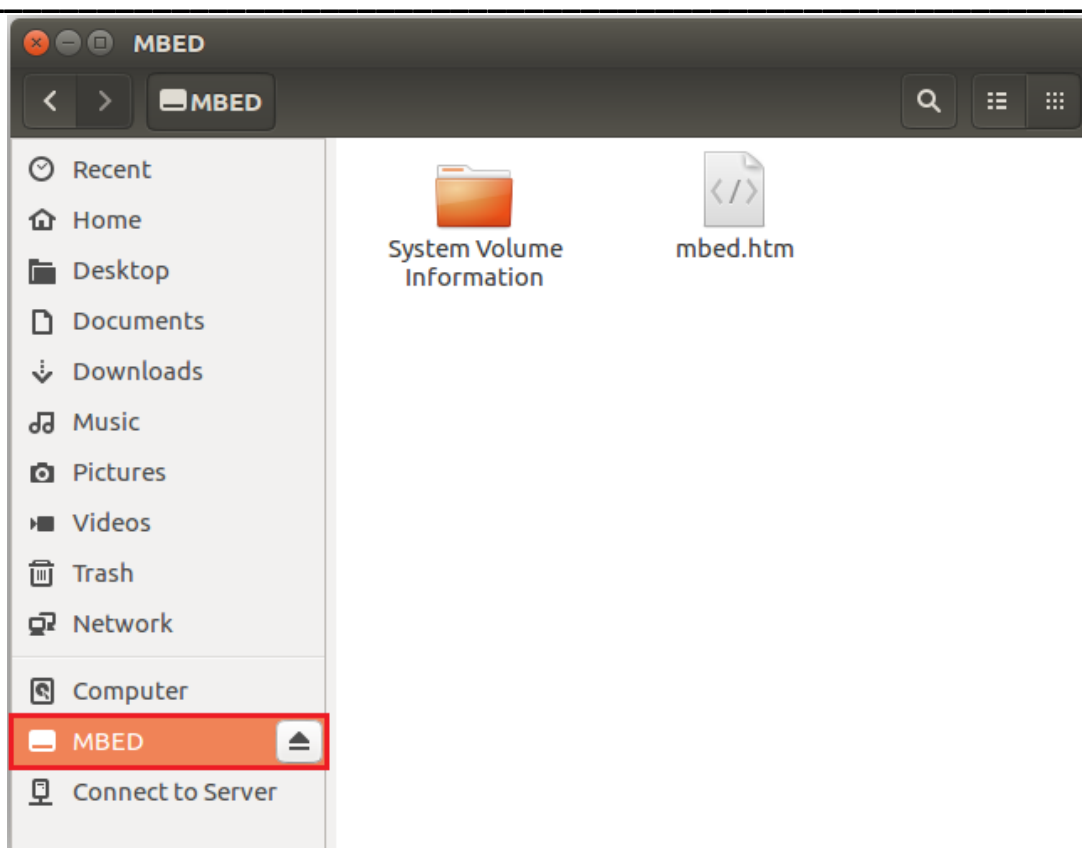
3.2.1.2 Linux

Ameba Device Board supports CMSIS-DAP debugger. We can use OpenOCD/CMSIS-DAP to download the software and enter GBD debugger mode under GCC environment.

Connect board to the PC with micro-USB cable:



If the board is connected to PC, there should be MBED drive shown in file explorer. Note that if you are using Virtual Machine as your Linux platform, please make sure the USB connection setting between VM host and client is correct so that the VM client can detect MBED drive.



Now that the MBED drive can be detected by our platform, we need to install OpenOCD package as our GDB server. You can use package manager to install it. To check whether it existed, use its version command to check:

```
realtek@realtek-VirtualBox:~$ openocd -v
Open On-Chip Debugger 0.9.0 (2015-09-02-10:42)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.org/doc/doxygen/bugs.html
```

Note. We suggest the version of OpenOCD you installed should be newer than (or equal to) 0.9.0, which is available on package manager of Ubuntu 16.04. If the version of OpenOCD you installed is older than it, the connection might not be successful. You can refer Sec. 6.2 to know how to configure and build the newer version of OpenOCD on your platform.

After OpenOCD installation, open a new terminal and run the GCC-RELEASE/run_openocd.sh script. You should see some messages like below figure and the GDB server has been started. If you see some error message contains “unable to open CMSIS-DAP device”, it might be caused

Note that this script should **NOT** be suspended if you want to download software or enter GDB debugger.

```
$ sh run openocd.sh
```

```
realtek@realtek-VirtualBox:~/sdk-ameba1-v3.5a_beta_v4/project/realtek_ameba1_va0
_example/GCC-RELEASE$ sh run_openocd.sh
Found openocd running, Kill it
run_openocd.sh: 9: kill: Illegal number: realtek
Open On-Chip Debugger 0.9.0 (2015-09-02-10:42)
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.org/doc/latest/tutorial.html#ReportingABug
Info : Only one adapter found: Realtek USB-Blaster v2.0
adapter speed: 10 kHz
adapter reset delay: 200
device to connect: realtek_ameba1 (auto-detect)
Info : realtek_ameba1: 32K RAM, 16KB program ROM
Info : realtek_ameba1: 100000000Hz (100MHz)
Info : realtek_ameba1: 16KB program ROM (16)
Info : SWCLK/CLK = 1 (SWCLK/CLK = 1, CPU = 0, DCD = 0, JTAG = 0, SWCLK = 0, SWCLK = 1)
Info : realtek_ameba1: JTAG core ready
Info : clock speed 10 MHz
Info : SWD interface established
Info : initbus4: add hardware hw 6 breakpoints, 4 watchpoints
```

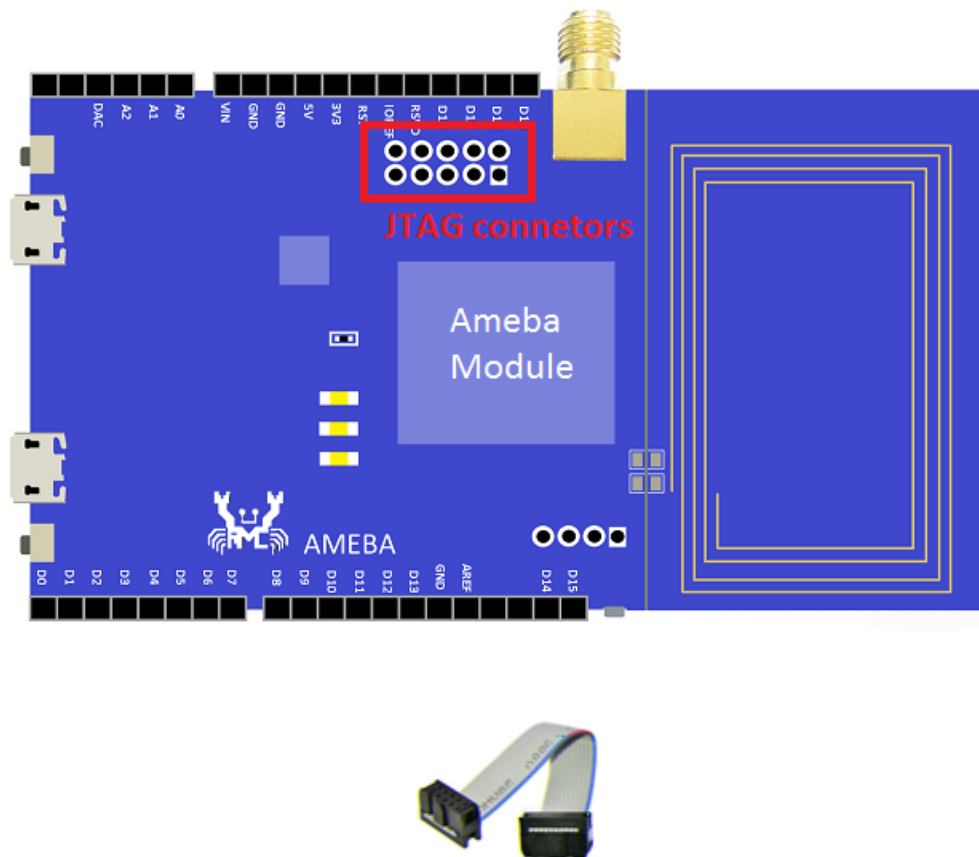
On the project terminal you should type below command before you using OpenOCD/CMSIS-DAP to download software or enter GDB debugger:

```
$ make setup GDB SERVER=openocd
```

```
realtek@realtek-VirtualBox:~/sdk-ameba1-v3.5a_beta_v4/project/realtek_ameba1_va0
_example/GCC-RELEASE$ make setup GDB_SERVER=openocd
make[1]: Entering directory '/home/realtek/sdk-ameba1-v3.5a_beta_v4/project/real
tek_ameba1_va0_example/GCC-RELEASE'
```

3.2.2 Jlink

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.

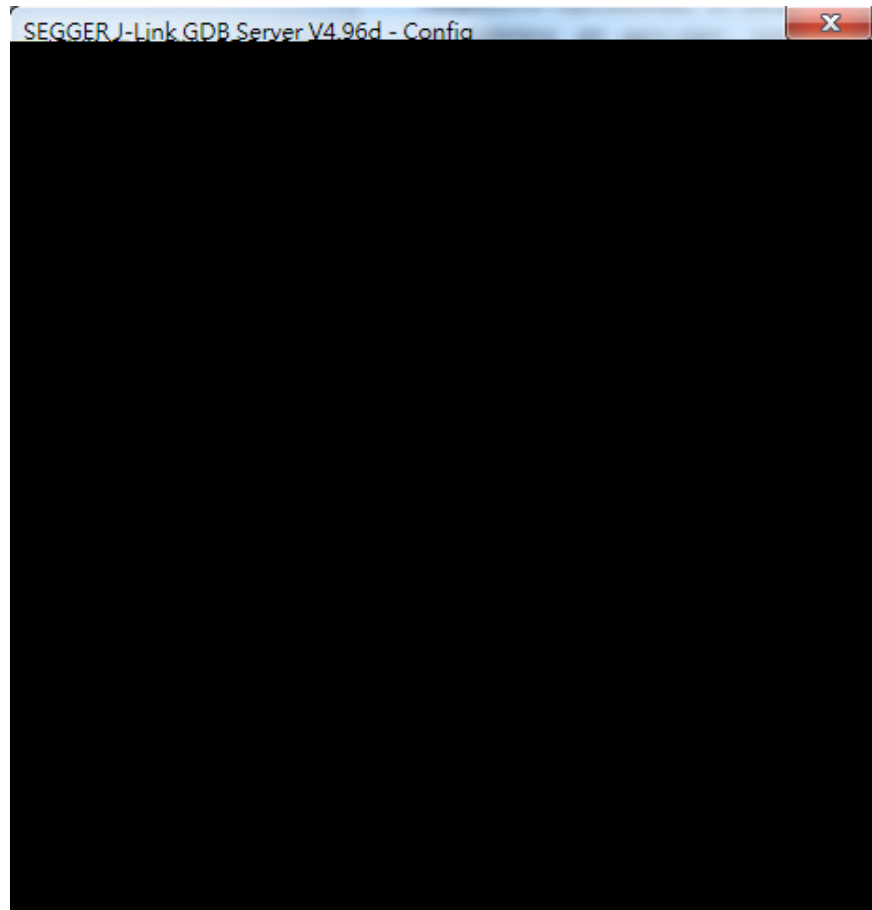


2.54mm 2x5pins connector (or use Dupont Line)

3.2.2.1 Windows

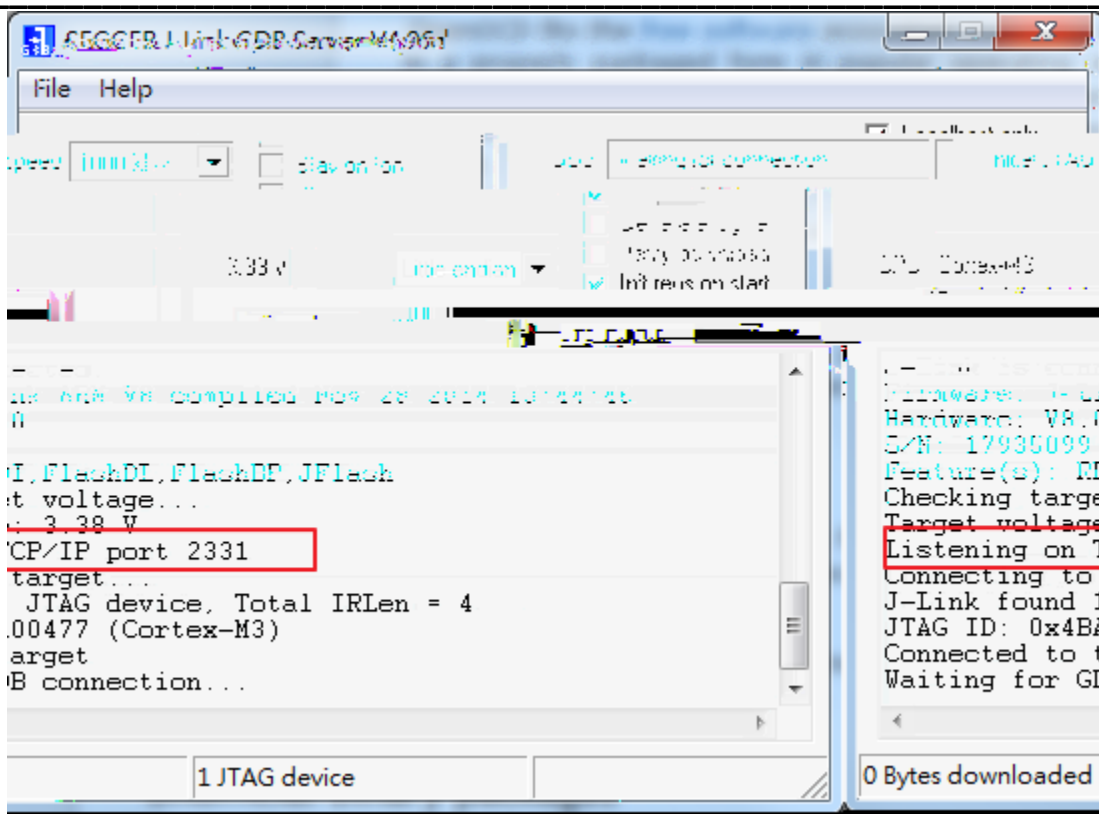
Besides the hardware configuration, it also requires installing J-Link GDB server. For Windows, please check <http://www.segger.com> and download “J-Link Software and Documentation Pack” (<https://www.segger.com/downloads/jlink>). After the installation of the software pack, you

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 look 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\rtl_gdb_flash_write.txt” so the Cygwin can connect to the GDB server successfully.



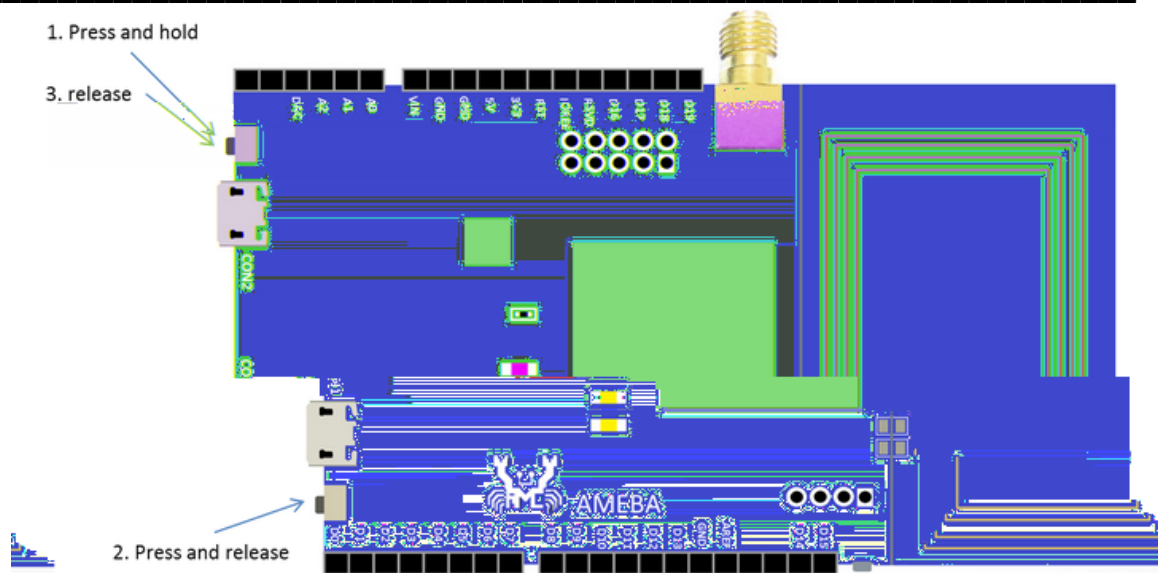
On the Cygwin terminal you should type below command before you using J-Link to download software or enter GDB debugger:

```
$ make setup GDB_SERVER=jlink
```

```
ypchiu@RTCN12686 /cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
$ make setup GDB_SERVER=jlink
make[1]: Entering directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE'
Setup jlink
cp -p ../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_debug_jlink.txt ../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_debug.txt
cp -p ../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_ramdebug_jlink.txt ../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_ramdebug.txt
cp -p ../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_flash_write_jlink.txt ../../component/soc/realtek/8195a/misc/gcc_utility/rtl_gdb_flash_write.txt
make[1]: Leaving directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE'
```

3.2.2.2 Linux

For Linux, we need to disable DAP at first so that the DAP signal won't conflict with JTAG signal. Please follow the steps shown at below figure to disable it. If it success, you should see a drive named "CRP DISABLD" instead of "MBED". (This can refer to the first half content of <http://www.amebaiot.com/en/change-dap-firmware/>)



And for J-Link GDB server, please check <http://www.segger.com> and download “J-Link Software and Documentation Pack” (<https://www.segger.com/downloads/jlink>). We suggest using Debian package manager to install the Debian version:

```
$ dpkg -i jlink_6.0.7_x86_64.deb
```

After the installation of the software pack, there should be a tool named “JLinkGDBServer” under JLink directory. Take Ubuntu 16.04 as example, the JLinkGDBServer can be found at /opt/SEGGER/JLink/ directory. Please open a new terminal and type following command to start GDB server. Note that this terminal should **NOT** be closed if you want to download software or enter GDB debugger mode.

```
$ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m3
```

```
realtek@realtek-VirtualBox:~$ /opt/SEGGER/JLink/JLinkGDBServer -device cortex-m3
SEGGER J-Link GDB Server V6.00g Command Line Version

JLinkARM.dll V6.00g (DLL compiled Aug 17 2016 13:20:32)

-----GDB Server start settings-----
GDBInit file:                none
GDB Server Listening port:    2331
SWO raw output listening port: 2332
Terminal I/O port:          2333
Accept remote connection:    yes
Generate logfile:            off
Verify download:             off
Init regs on start:          off
Silent mode:                 off
Single run mode:             off
Target connection timeout:    0 ms
-----J-Link related settings-----
J-Link Host interface:       USB
J-Link script:               none
J-Link settings file:        none
-----Target related settings-----
Target device:               cortex-m3
Target interface:            JTAG
Target interface speed:      1000kHz
Target endian:               little

Connecting to J-Link...
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: 3.32 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...
```

The started J-Link GDB server should look like above figure. Please make sure the TCP/IP port is **2331** which should be the same as default setting in “component\soc\realtek\8195a\misc\gcc_utility\rtl_gdb_flash_write.txt”.

On the project terminal you should type below command before you using J-Link to download software or enter GDB debugger:

```
$ make setup GDB_SERVER=jlink
```

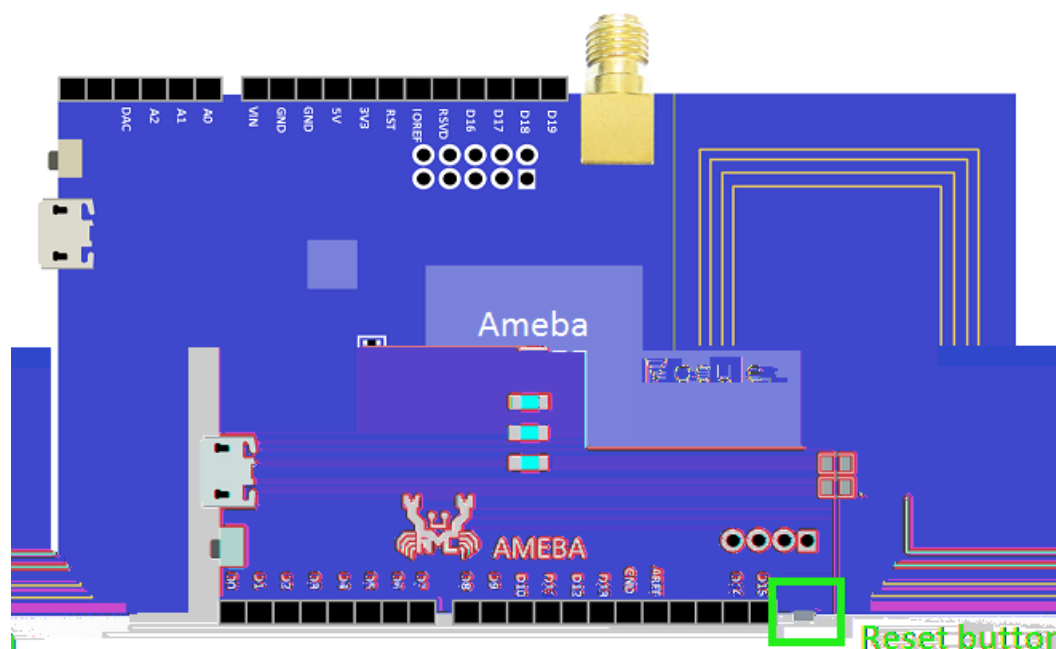
```
realtek@realtek-VirtualBox:~/sdk-ameba1-v3.5a_beta_v4/project/realtek_ameba1_va0
_example/GCC-RELEASE$ make setup GDB_SERVER=jlink
make[1]: Entering directory '/home/realtek/sdk-ameba1-v3.5a_beta_v4/project/real
tek_ameba1_va0_example/GCC-RELEASE'
```

3.3 Download code to flash

To download software into Ameba Device Board, please make sure steps mentioned in Sec.0 to Sec.3.2 are done and then type below command on Cygwin (Windows) or terminal (Linux).

\$ make flash

This command would download the software into flash and it would take a few seconds to finish. After successful download, please press the Reset button and you should see that the device now is booted with new image.



Note. If the download procedure hangs for a long time, you can check Sec. 6.3 to troubleshoot the issue by updating newest DAP firmware.

3.4 Enter GDB debugger

To enter GDB debugger mode, please make sure steps mentioned in Sec.3.1 to Sec. 3.3 are finished and then **reset the device** first. After reset chip, type below command on Cygwin (Windows) or terminal (Linux) to enter GDB:

\$ make debug

```
ypchiu@RTCN12686 /cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
$ make debug
if [ ! -d ../../tools/arm-none-eabi-gcc/4.8.3-2014q1 ] ; then tar -zxf ../../tools/arm-none-eabi-gcc/4
.8.3-2014q1.tar.gz -C ../../tools/arm-none-eabi-gcc/ ; fi
make[1]: Entering directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-REL
EASE'
../../component/soc/realtek/8195a/misc/gcc_utility//Check_Jtag.sh
0
0
/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
../../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-gdb -x ../../component/soc/realtek/8195a
/misc/gcc_utility//rtl_gdb_debug.txt
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20140228-cvs
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
0x10008d18 in ?? ()
Notification of completion for asynchronous execution commands is off.
rtl8195a.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000100 msp: 0x1fffffff
0x40000040: 00fcc702
0x40005000: 0000000d
Breakpoint 1: file ../src/main.c, line 18.

Breakpoint 1, main () at ../src/main.c:18
18      console_init();
(gdb) |
```

3.5 Download and debug in RAM

This section describes another command that can download the software into RAM and then enter GDB debug mode. Generally, this command is the combination of Sec.3.3 and Sec.3.4. But the command mentioned in Sec.3.3 is to download software into flash, while in this section we download software into RAM. To use this command please make sure steps mentioned in Sec.0 to Sec.3.2 are done and then type below command on Cygwin (Windows) or terminal (Linux).

\$ make ramdebug

You should see some messages like below in terminal indicates that you have entered the GDB debugger.

```

ypchiu@RTCN12686 /cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
$ make ramdebug
if [ ! -d ../tools/arm-none-eabi-gcc/4.8.3-2014q1 ] ; then tar -zxf ../tools/arm-none-eabi-gcc/4.8.3-2014q1.tar.gz -C ../tools/arm-none-eabi-gcc/ ; fi
make[1]: Entering directory '/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE'
../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-gdb -x ../component/soc/realtek/8195a/misc/gcc_utility//Check_Jtag.sh
0
0
/cygdrive/d/sdk-ameba1-v3.5a_beta_v2/project/realtek_ameba1_va0_example/GCC-RELEASE
../tools/arm-none-eabi-gcc/4.8.3-2014q1/bin/arm-none-eabi-gdb -x ../component/soc/realtek/8195a/misc/gcc_utility//rtl_gdb_ramdebug.txt
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20140228-cvs
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
0x10008d14 in ?? ()
Notification of completion for asynchronous execution commands is off.
rtl8195a.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000100 msp: 0x1ffffffc
0x40000040: 00fcc702
0x40000500: 00000000
Loading section bootloader size 0x3a98 lma 0x10000000
...
c/main.c:18:
Breakpoint 11, main () at ./src/18:
(gdb)

```

4 Command list

Command	Usage	Description
all	\$ make all	Compile project to generate <i>ram_all.bin</i>
clean	\$ make clean	Remove compile result (*.bin, *.o,...)
clean_all	\$ make clean_all	Remove compile result and Toolchains
flash	\$ make flash	Download <i>ram_all.bin</i> to flash
setup	\$ make setup GDB_SERVER=server (server=openocd or jlink)	Setup GDB_SERVER
debug	\$ make debug	Enter gdb debug
ramdebug	\$ make ramdebug	Write <i>ram_all.bin</i> to RAM then enter gdb debug

5 GDB debugger basic usage

GDB, the GNU project debugger, allows you to examine the program while it executes and it is helpful for catching bugs. In Sec. 3.4 and Sec. 3.5, we have described how to enter GDB debugger mode. And for this section, we will illustrate some basic usage of GDB commands. For further information about GDB debugger and its commands, please check <https://www.gnu.org/software/gdb/> and <https://sourceware.org/gdb/current/onlinedocs/gdb/>.

5.1 Stop and continue

5.1.1 Breakpoint

Breakpoints are set with the *break* command (abbreviated *b*). The usage can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Breaks.html>.

\$ break

```
(gdb) break example_entry
Breakpoint 2: file ../../component/common/example/example_entry.c, line 208.
(gdb) continue
Continuing.

Breakpoint 2, example_entry () at ../../component/common/example/example_entry.c:208
208      {
```

5.1.2 Watchpoint

You can use a watchpoint to stop execution whenever the value of an expression changes. The related commands include *watch*, *rwatch*, and *awatch*. And the usage of these commands can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Watchpoints.html>.

\$ watch

```
(gdb) watch wifi.security_type
Hardware watchpoint 9: wifi.security_type
(gdb) continue
Continuing.

Program received signal SIGTRAP, Trace/breakpoint trap.
FATWC (arg=<optimized out>) at ../../component/common/api/at_cmd/atcmd_wifi.c:850
850      wext_get_mode(WLAN0_NAME, &mode);
```

Note that please keep the range of watchpoints less than 20 bytes, or the watchpoints might dump some warning messages like below figure:


```
(gdb) watch wifi
Hardware watchpoint 11: wifi
(gdb) continue
Continuing.
Warning:
Could not insert hardware watchpoint 11.
Could not insert hardware breakpoints:
You may have requested too many hardware breakpoints/watchpoints.
```

5.1.3 Print breakpoints and watchpoints

To print a table of all breakpoints, watchpoints set and not deleted, use the *info* command. You can simply type *info* to know its usage.

\$ info

```
(gdb) info breakpoints
Num      Type          Disp Enb What
2        breakpoint      keep y   in example_entry at ../../component/common/example/example_entry.c:208
3        breakpoint      keep y   in fATWx at ../../component/common/api/at_cmd/atcmd_wifi.c:412
```

5.1.4 Delete breakpoints

To eliminate the breakpoints, use the *delete* command (abbreviated *d*). The usage can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/Delete-Breaks.html>.

\$ delete

```
(gdb) info breakpoints
Num      Type          Disp Enb What
4        breakpoint      keep y   in example_entry at ../../component/common/example/example_entry.c:208
5        breakpoint      keep y   in fATWx at ../../component/common/api/at_cmd/atcmd_wifi.c:412
(gdb) delete
(gdb) info breakpoints
No breakpoints or watchpoints.
```

5.1.5 Continue

To resume program execution, use the *continue* command (abbreviated *c*). The usage can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html>.

\$ continue

```
(gdb) continue
Continuing.
```

5.1.6 Step

To step into a function call, use the *step* command (abbreviated *s*). It will continue running your program until control reaches a different source line. The usage can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html>.

```
$ step
```

5.1.7 Next

To step through the program, use the *next* command (abbreviated *n*). The execution will stop when control reaches a different line of code at the original stack level. The usage can be found at [https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-](https://sourceware.org/gdb/current/onlinedocs/gdb/Continuing-and-Stepping.html)

```
(gdb) backtrace
#0  FATWx (arg=) at ../../../../component/common/api/at_cmd/atcmd_wifi.c:412
#1  log_handler (cmd=cmd@entry="ATW?") at ../../../../component/common/api/at_cmd/log_service.c:205
#2  log_service (param=<optimized out>) at ../../../../component/common/api/at_cmd/log_service.c:371
#3  ulPortSetInterruptMask () at ../../../../component/os/freertos/freertos_v8.1.2/Source/portable/GCC/ARM_CM3/port.c:419
#4  ulPortSetInterruptMask () at ../../../../component/os/freertos/freertos_v8.1.2/Source/portable/GCC/ARM_CM3/port.c:419
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
```

5.2.2 Print source lines

To print lines from a source file, use the *list* command (abbreviated *l*). The usage can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/List.html>.

\$ list

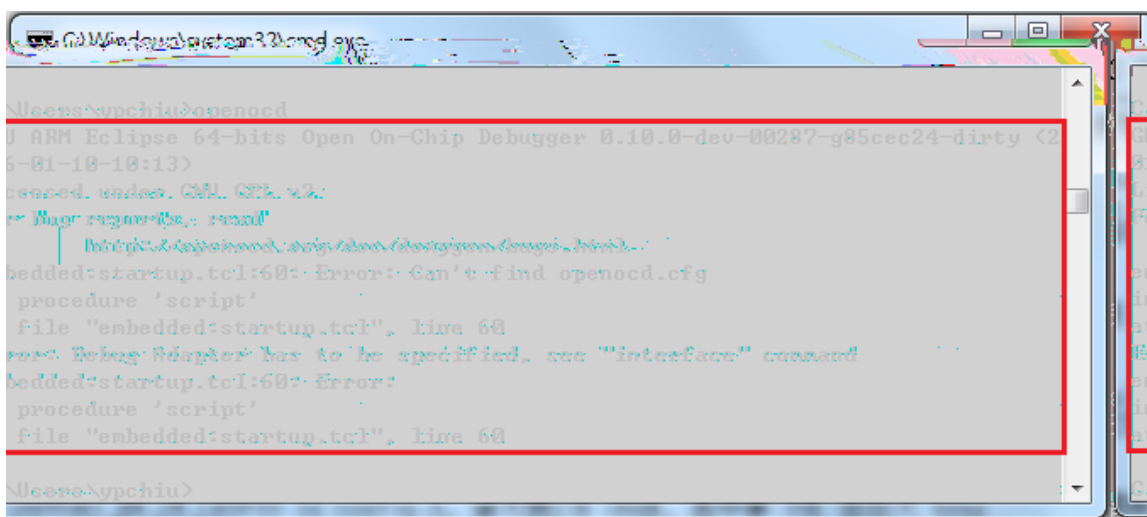
```
(gdb) list 15,39
15      void main(void)
16      {
17          /* Initialize log uart and at command service */
18          console_init();
19
20          /* pre-processor of application example */
21          pre_example_entry();
22
23          /* wlan initialization */
24          #if defined(CONFIG_WIFI_NORMAL) && defined(CONFIG_NETWORK)
25              wlan_network();
26          #endif
27
28          /* Execute application example */
29          example_entry();
30
31          /*Enable Schedule, Start Kernel*/
32          #if defined(CONFIG_KERNEL) && !TASK_SCHEDULER_DISABLED
33              #ifdef PLATFORM_FREERTOS
34                  vTaskStartScheduler();
35              #endif
36          #else
37              RtlConsolTaskRom(NULL);
38          #endif
39      }
```

To examine data in your program, you can use *print* command (abbreviated *p*). It evaluates and prints the value of an expression. The usage can be found at <https://sourceware.org/gdb/current/onlinedocs/gdb/Data.html>.

```
(gdb) print wifi.ssid
$8 = {len = 7 '\a', val = "Test_ap", '\000' <repeats 25 times>}
```

6.1 Unable to execute run_openocd.bat normally on Windows

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. 3.2.1 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. To do this, you need to open a new Cygwin window, locate the corresponding directory which contains run_openocd.sh, and type “sh run_openocd.sh” to execute the 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 : rt18195a.cpu: hardware has 6 breakpoints, 4 watchpoints
|
```

6.2 How to install the newest version of OpenOCD on Ubuntu

As mentioned in Sec. 3.2.1.2, if the version of OpenOCD you are using is not newer than (or equal to) 0.9.0, the OpenOCD/CMSIS-DAP connection might fail. However, if you are using Ubuntu 12.04 or 14.04, the OpenOCD you installed by the package manager might be the older one. Hence in this section, we provide a guide for compiling and installing the newest OpenOCD. The following steps have been tested under Ubuntu 12.04 and 14.04. And for other Linux OS, it should also be worked if you make proper changes based on your platform.

First, we assume that you have access to root privileges and you need to install some required packages. The packages include git, gcc build environment, usb-related libraries:

```
$ sudo apt-get install git build-essential g++ autotools-dev make libtool pkg-config
autoconf automake texinfo libudev-dev libusb-1.0-0-dev libfox-1.6-dev
```

Second, we need to install HIDAPI library before OpenOCD. HIDAPI is a library which allows applications to interface with USB devices. You can refer <http://www.signal11.us/oss/hidapi/> for more information about it. To install it, we are going to clone the git project and compile it:

```
$ cd ~/
$ git clone https://github.com/signal11/hidapi.git
$ cd hidapi/
$ ./bootstrap
$ ./configure
$ make
$ sudo make install
```

After typing above commands, the HIDAPI should be installed. But we still need to add the location of the hid library into system PATH variable. For Ubuntu, please use an editor to open ~/.profile file:

```
$ vim ~/.profile
```

And at the bottom of .profile, please add the following line:

```
PATH="$HOME/bin:/usr/local/lib:$PATH"
```

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

PATH="$HOME/bin:/usr/local/lib:$PATH"
```

To reload the PATH variable, you can use below command:

```
$ source ~/.profile
```

And you can use *echo* command to check the updated content of PATH variable:

```
$ echo $PATH
```

```
realtek@realtek-VirtualBox:~$ echo $PATH
/home/realtek/bin:/usr/local/lib:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

We also need to update our system shared library cache by following command:

```
$ sudo ldconfig
```

Finally, we are going to compile and install OpenOCD library after we installed HIDAPI:

```
$ cd ~/
```

```
$ git clone git://git.code.sf.net/p/openocd/code openocd-code
```

```
$ cd openocd-code/
```

```
$ ./bootstrap
```

Since we are using OpenOCD/CMSIS-DAP, we only enable its corresponding configuration:

```
$ ./configure --enable-cmsis-dap --disable-gccwarnings
```

```
$ make
```

```
$ sudo make install
```

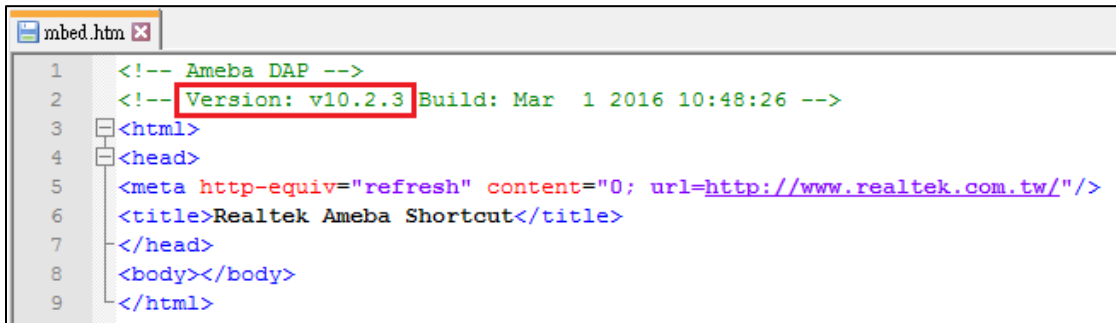
At this point, we have installed the newest OpenOCD library and the OpenOCD/CMSIS-DAP connection should be able to work. You can use *-v* command to check its version:

```
$ openocd -v
```

```
realtek@realtek-VirtualBox:~$ openocd -v
Open On-Chip Debugger 0.10.0-dev-00371-g81631e4-dirty (2016-08-29-13:50)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
```

6.3 Download procedure hang for a long time

In Sec. 3.3, 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.



```
1 <!-- Ameba DAP -->
2 <!-- Version: v10.2.3 Build: Mar 1 2016 10:48:26 -->
3 <html>
4 <head>
5 <meta http-equiv="refresh" content="0; url=http://www.realtek.com.tw/" />
6 <title>Realtek Ameba Shortcut</title>
7 </head>
8 <body></body>
9 </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.