

GUIDELINE TO CODE AND DEBUG

BEAGLEBONE WITH VSCode

Contents

I. Prerequisites	2
II. Install software and dependency	3
III. Setup environment for the Beaglebone	5
IV. How to code and debug on BBB	8
V. How to set up the environment for kernel development on VSCode	10
VI. Debug Kernel for BBB by command line on VSCode	16
Reference:	19

I. Prerequisites

A full rootfs has all necessary packages (gcc, gdb, binutils and eglibc, ...)

- On host, make suitable rootfs for target device

```
$ wget https://debian.beagleboard.org/images/bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img.xz
```

```
$ unxz bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img.xz
```

```
$ fdisk -l bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img
```

```
$ sudo mount -o loop,offset=4194304 bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img /mnt
```

```
hoang@hoang-PC:/source/rootfs$ fdisk -l bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img
Disk bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img: 3,5 GiB, 3774873600 bytes, 7372800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x9bd889f0

Device                                Boot Start      End Sectors  Size Id Type
bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img1 *    8192 7372799 7364608  3,5G 83 Linux
hoang@hoang-PC:/source/rootfs$ sudo mount -o loop,offset=4194304 bone-debian-9.12-imgtec-armhf-2020-04-06-4gb.img /mnt
hoang@hoang-PC:/source/rootfs$ ls /mnt/
bbb-uEnv.txt  boot  etc  ID.txt  lost+found  mnt  nfs-uEnv.txt  opt  root  sbin  sys  usr
bin          dev  home  lib    media      nfs-uEnv.txt  proc  run  srv  tmpo  var
```

Note: offset=4194304 = Start sector * 512 = 8192*512

- Assume that, /dev/sdb2 is second partition of BBB's Sdcard. Copy all content of rootfs to sdb2.

```
$ sudo mount /dev/sdb2 /source/mount/
```

```
$ sudo rm -rf /source/mount/*
```

```
$ sudo cp -rp /mnt/* /source/mount/
```

- Install Kernel module

```
$ cd /source/linux/
```

```
$ sudo make INSTALL_MOD_PATH=/source/mount/ modules_install
```

```
$ sync
```

- Insert Sdcard to board and boot to OS
- Install texinfo package (have makeinfo tool) on target device

```
$ wget https://ftp.gnu.org/gnu/texinfo/texinfo-6.6.tar.gz
```

```
$ tar xvfz texinfo-6.6.tar.gz
```

```
$ cd texinfo-6.6
```

```
$ ./configure --prefix=/usr/local/texinfo/6_6
```

```
$ make
```

```
$ make install
```

```
$ export PATH=$PATH:/usr/local/texinfo/6_6/bin/
```

- Install gdb (have gdb, gdbserver,... tools) on target device (BBB)

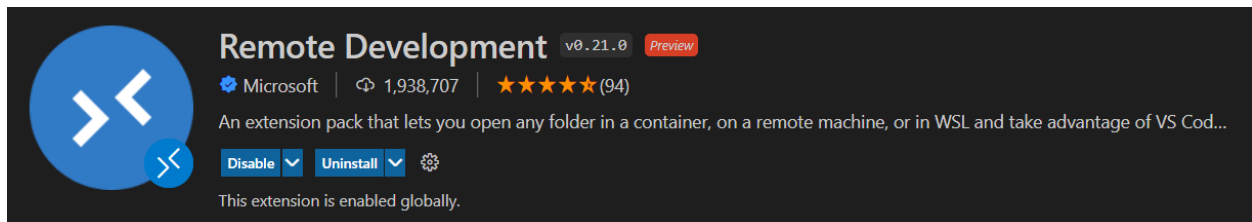
```
$ wget "http://ftp.gnu.org/gnu/gdb/gdb-7.11.tar.gz"
$ tar -xvzf gdb-7.11.tar.gz
$ cd gdb-7.11
$ ./configure
$ make
$ make install
$ gdb --version
```

Note: the `wget` command requires successful network connection from BBB to internet. Instead of that you can download the `.tar.gz` file and copy to rootfs of BBB.

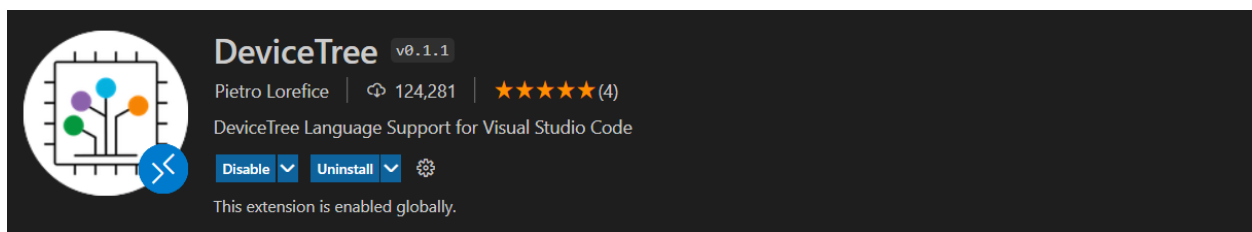
II. Install software and dependency

To get started, you need to:

1. Install an [OpenSSH compatible SSH client](https://code.visualstudio.com/docs/remote/troubleshooting#_installing-a-supported-ssh-client) if one is not already present. See the guide at https://code.visualstudio.com/docs/remote/troubleshooting#_installing-a-supported-ssh-client
2. Install [Visual Studio Code](#), [Cygwin](#) (if VSC is run on Windows OS)
3. Install these extension packs on VSC Local:
 - Remote Development v0.21.0

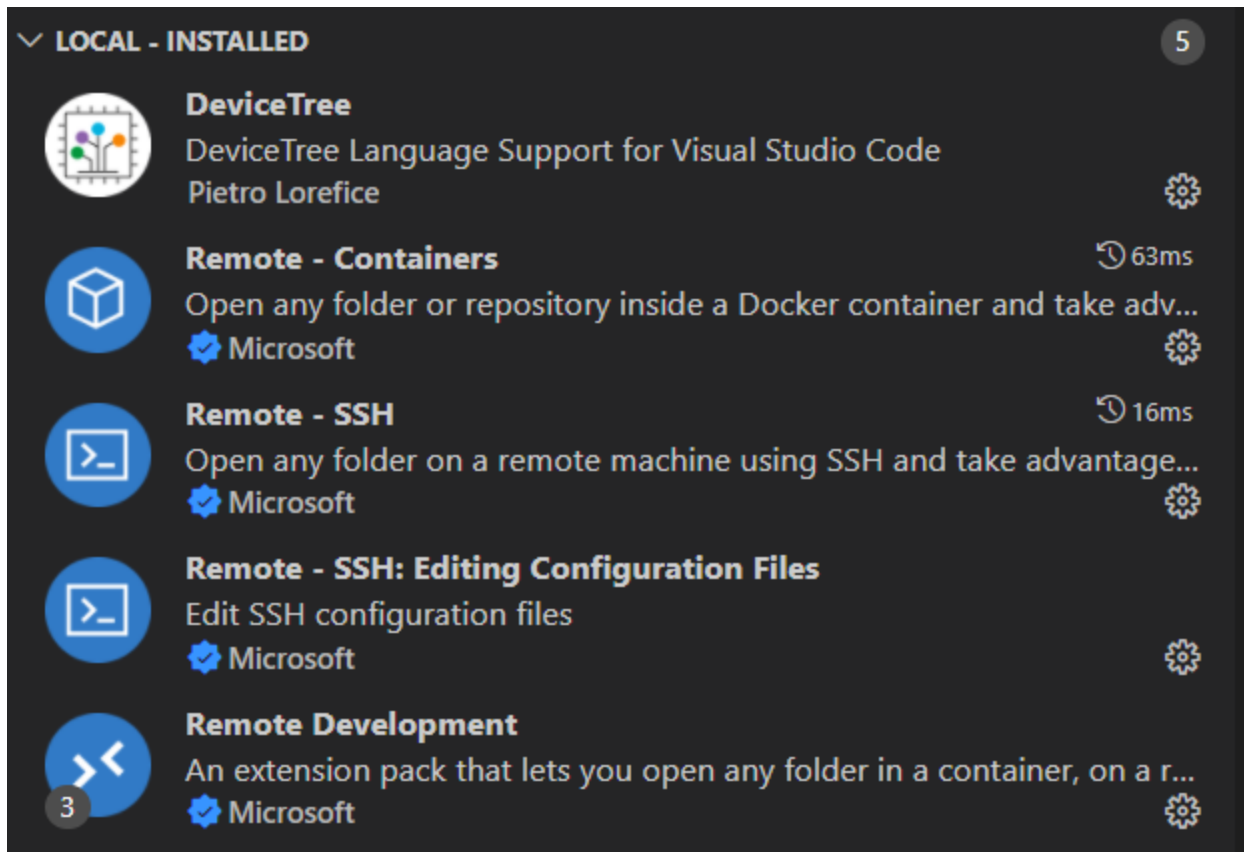


- DeviceTree v0.1.1



Note: Step 4 and 5 are only implemented after remote to target machine (ubuntu, BBB) successfully. See the chapter III.

The result of VSC after installing successfully:



4. Install these extension packs on SSH – target device (BBB):

Install extension by .vsix file.

Download these files and copy them to target device

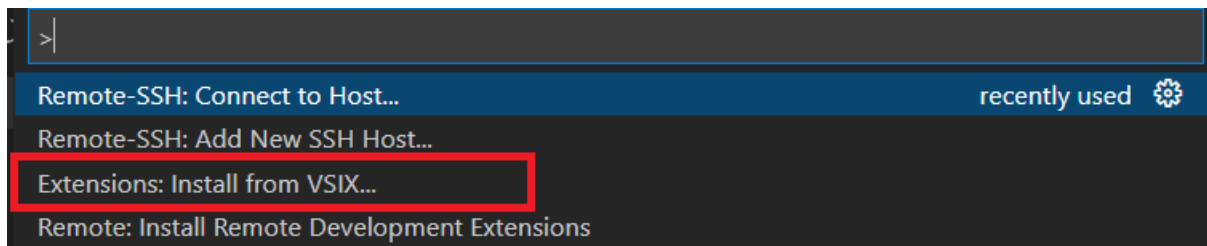
- C/C++ v1.7.1

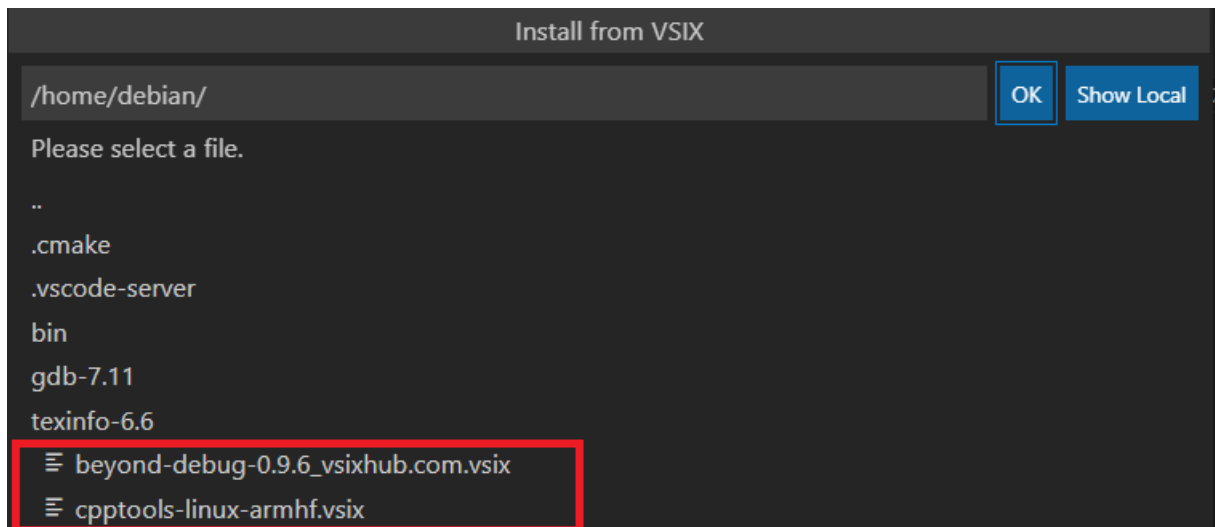
\$ `wget https://github.com/microsoft/vscode-cpptools/releases/download/1.7.1/cpptools-linux-armhf.vsix`

- GDB Debugger – Beyond

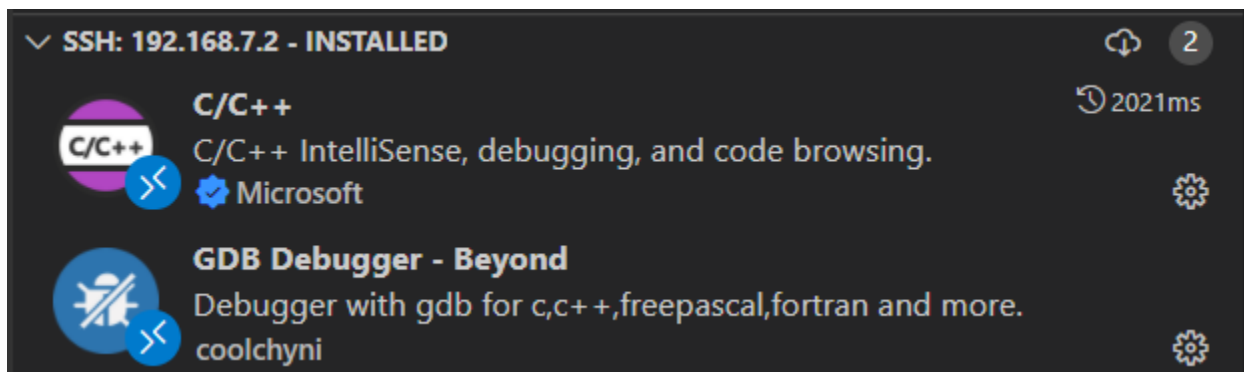
Download at <https://www.vsixhub.com/vsix/38158/>

Install Steps:

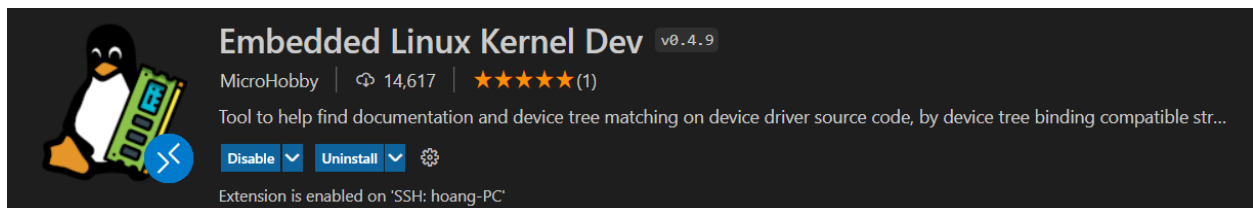




The results after installing successfully:



5. Install these extension packs on SSH – develop machine (Ubuntu, ...)
 - Embedded Linux Kernel Dev v0.4.9



III. Setup environment for the Beaglebone

- Connect network cable from Beaglebone to local PC.
- Check user name and host name of OS on the Beaglebone:

```

debian@beaglebone:~$ whoami
debian
debian@beaglebone:~$
debian@beaglebone:~$ cat /etc/hostname
beaglebone
debian@beaglebone:~$ █

```

- Check SSH connect from Local PC to Beaglebone by command: **ssh beaglebone -l debian (or ssh debian@192.168.7.2)**

```

C:\Users\admin>ssh beaglebone -l debian
The authenticity of host 'beaglebone (fe80::8a3f:4aff:fe24:54b6%6)' can't be established.
ECDSA key fingerprint is SHA256:3vshIoIA+rglG7phcXiBKRhQRmdERBaLnXkOn5akXEs.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'beaglebone,fe80::8a3f:4aff:fe24:54b6%6' (ECDSA) to the list of known hosts.
Debian GNU/Linux 10

BeagleBoard.org Debian Buster IoT Image 2020-04-06

Support: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

default username:password is [debian:1]

debian@beaglebone's password:

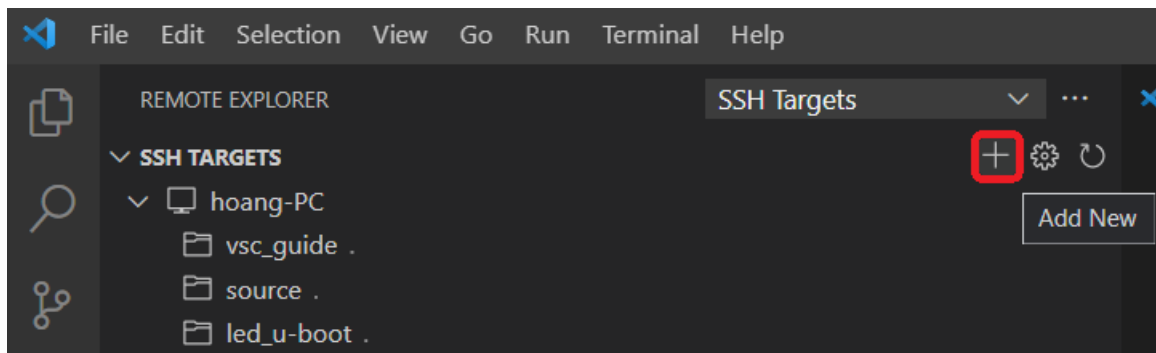
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Apr  8 19:11:59 2020 from 192.168.7.1
debian@beaglebone:~$

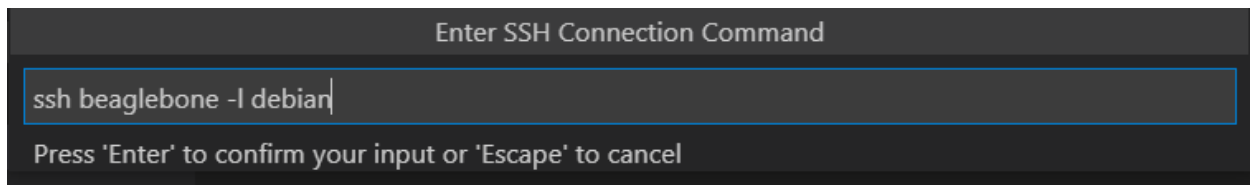
```

- Open VSCode and setup Remote Development using SSH

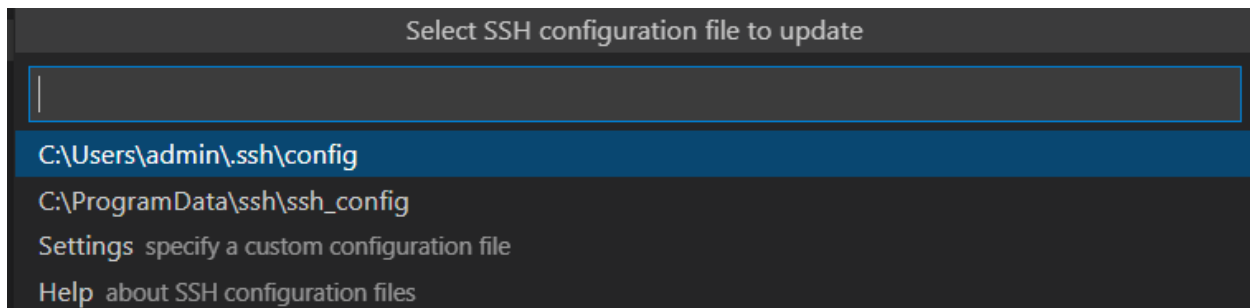
Start by selecting **Remote-SSH: Add New SSH Host...** from the Command Palette (**F1, Ctrl+Shift+P**) or clicking on the **Add New** icon in the SSH **Remote Explorer** in the Activity Bar.



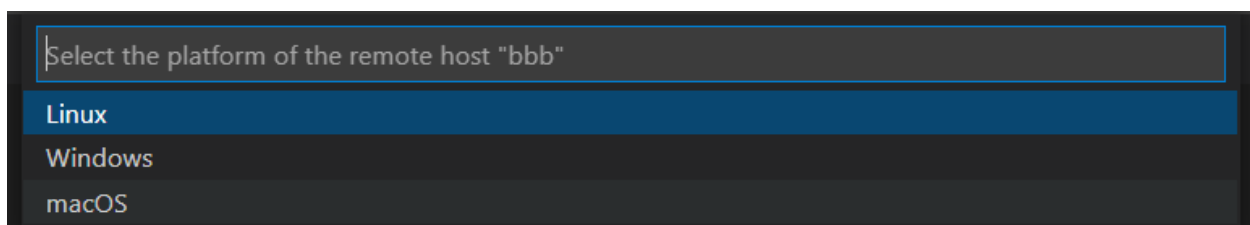
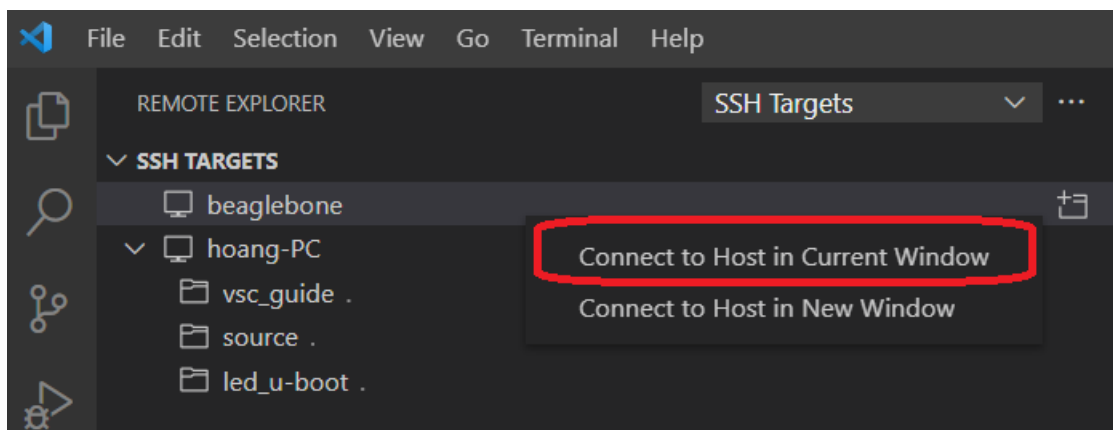
You'll then be asked to enter the SSH connection information. You can either enter a host name:



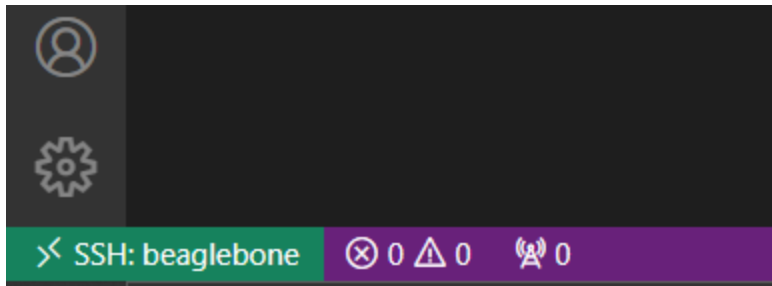
Finally, you'll be asked to pick a config file to use. You can also set the `"remote.SSH.configFile"` property in your User `settings.json` file if you want to use a different config file than those listed. The extension takes care of the rest!



From this point forward, the host will appear in the list of hosts when you select **Remote-SSH: Connect to Host...** from the Command Palette (F1, Ctrl+Shift+P) or in the **SSH Targets** section of the **Remote Explorer**.



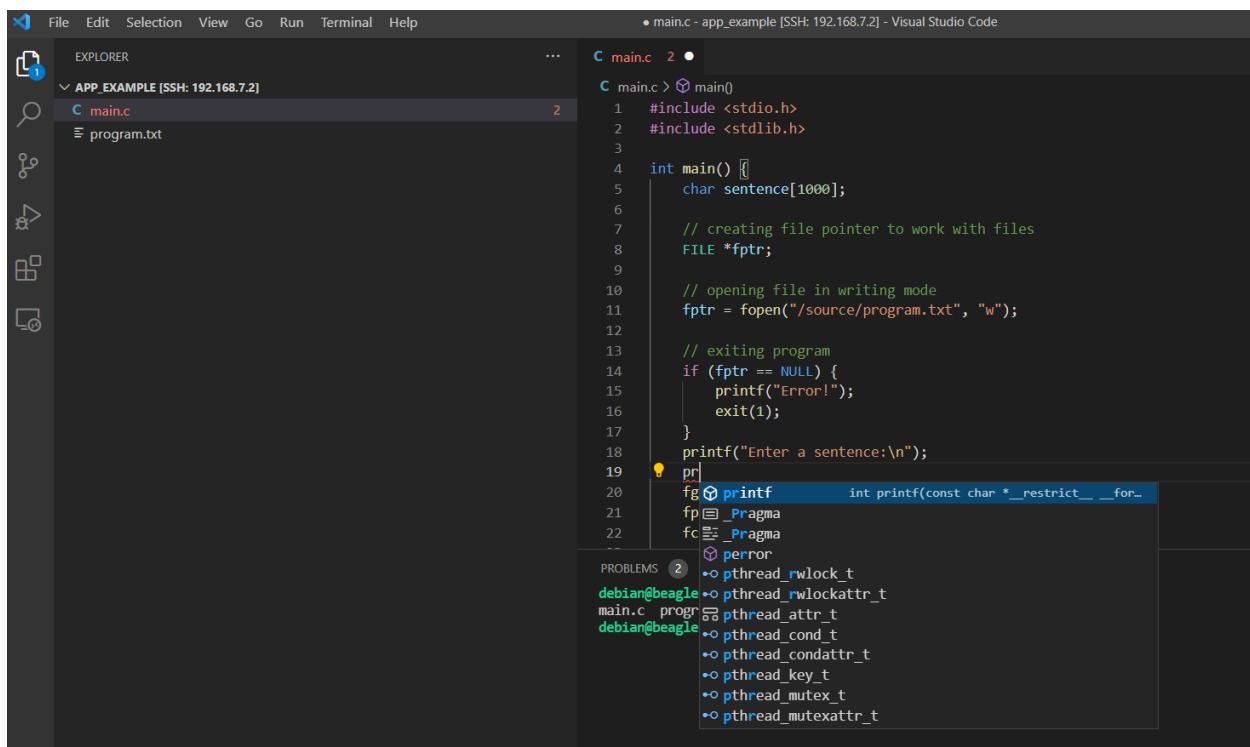
After you are connected, you'll be in an empty window. You can always refer to the Status bar to see which host you are connected to.



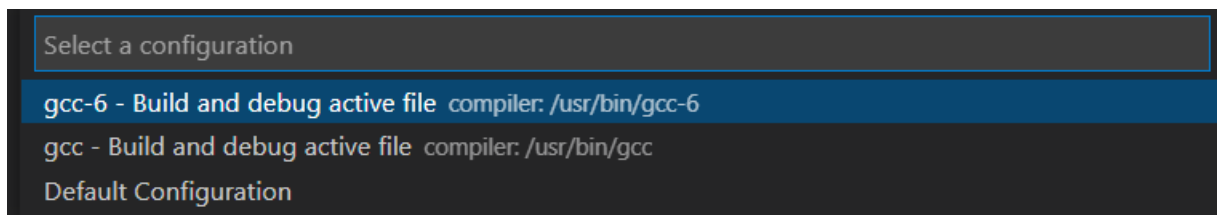
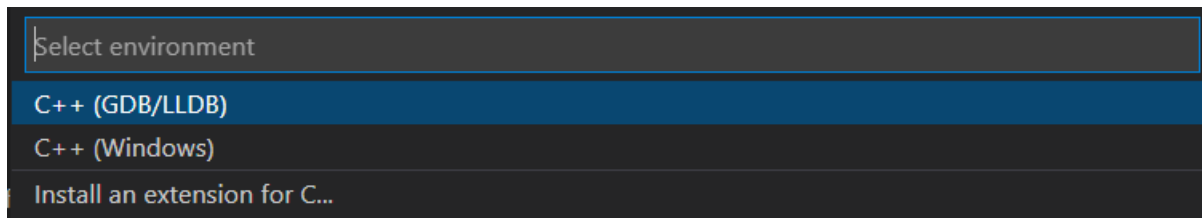
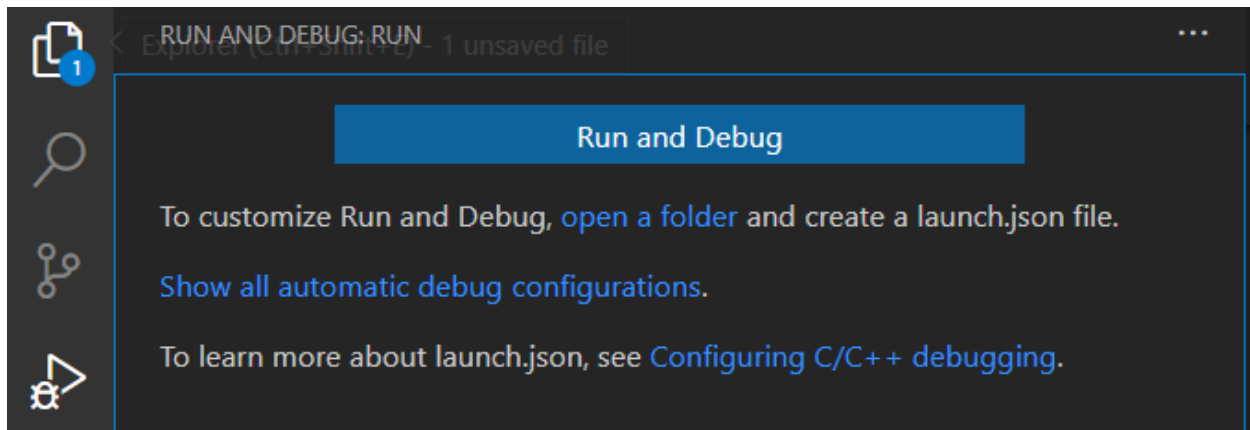
IV. How to code and debug on BBB

Use an example: **C Program to Write a Sentence to a File**. It also is located at “vsc_guide_BBB/source/app_example” folder.

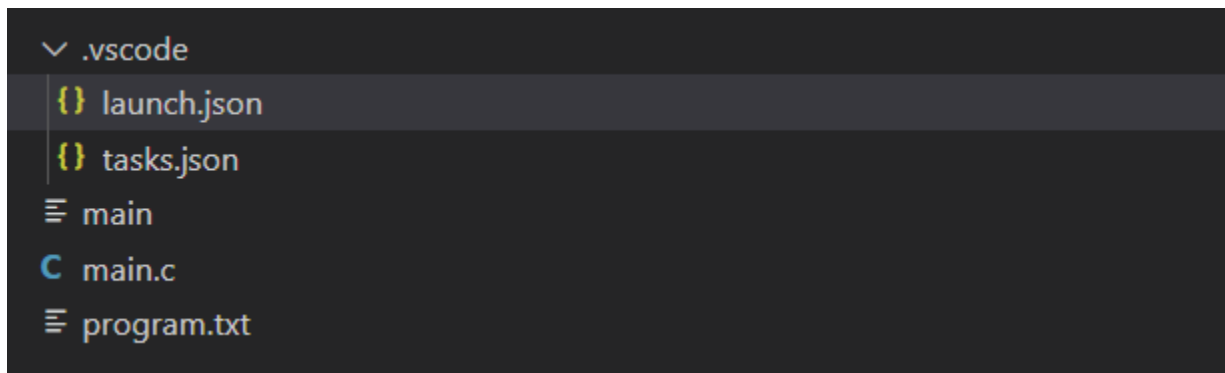
- Open folder -> /source/ app_example



- After completing the code then click Run & debug



After that, we will see:



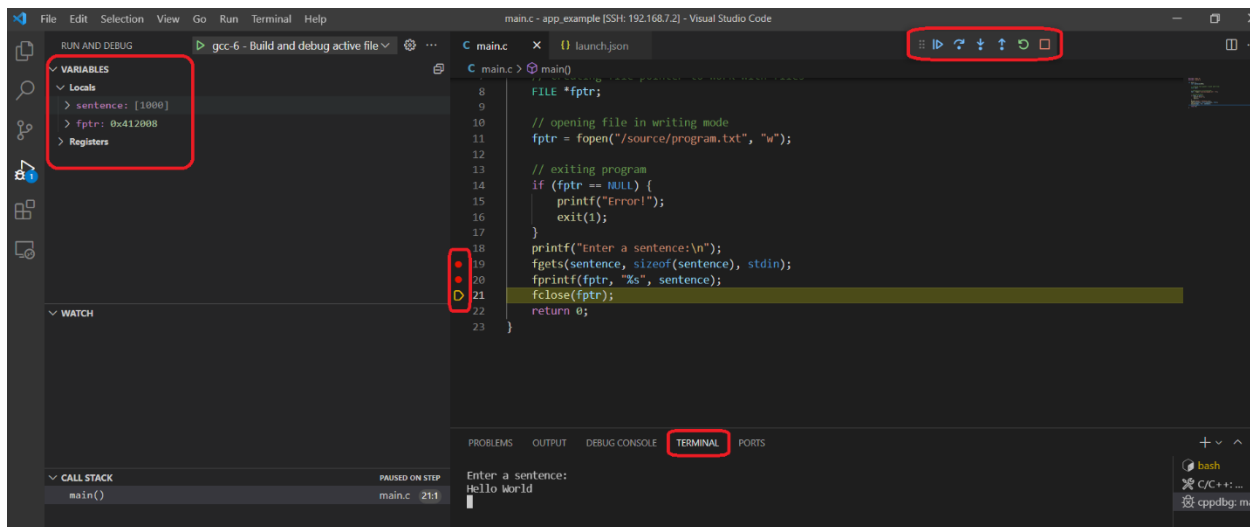
Modify the `"miDebuggerPath"` in `launch.json` file to point to correct path to **gdb** execute file on target file.

```
"miDebuggerPath": "/usr/bin/gdb"
```

=>

```
"miDebuggerPath": "/usr/local/bin/gdb"
```

- Set breakpoints and start debug the program



```

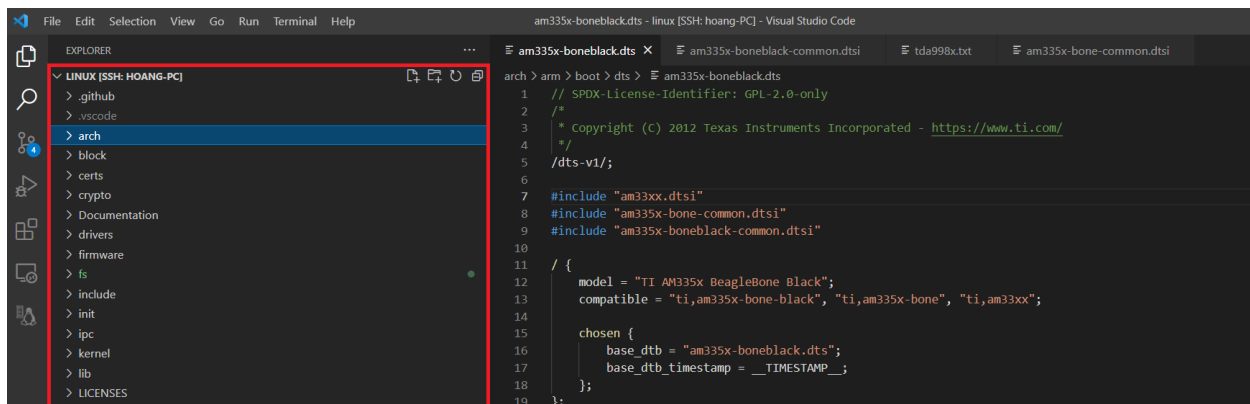
Enter a sentence:
hello world
[1] + Done
"/usr/local/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-I
n-arz3ewt4.o5a" 1>"/tmp/Microsoft-MIEngine-Out-qgtzuij2.f4e"
debian@beaglebone:/source/app_example$ ls
main main.c program.txt
debian@beaglebone:/source/app_example$ cat program.txt
hello world
debian@beaglebone:/source/app_example$

```

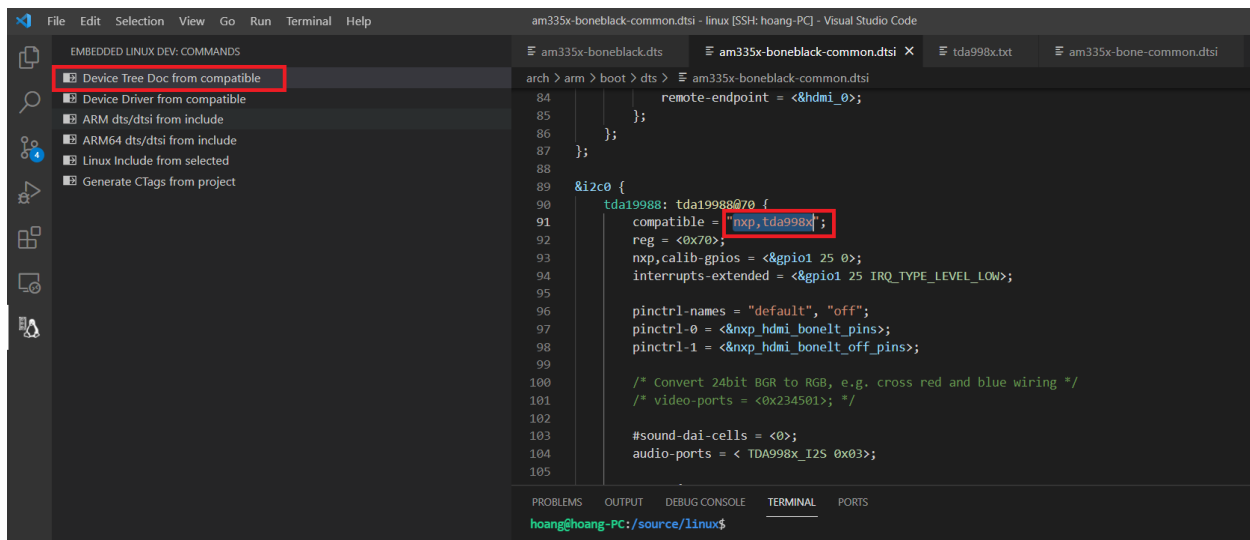
V. How to set up the environment for kernel development on VSCode

5.1. Some useful tools for kernel module development:

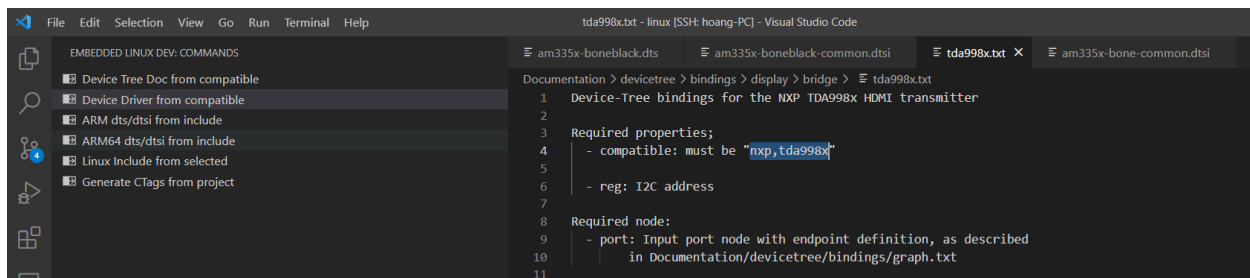
- Open Linux source code on Host machine



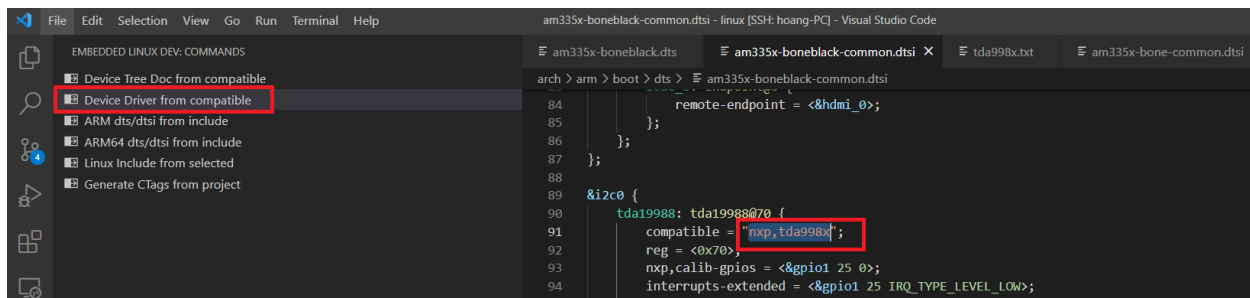
- Find Device Tree doc from compatible.



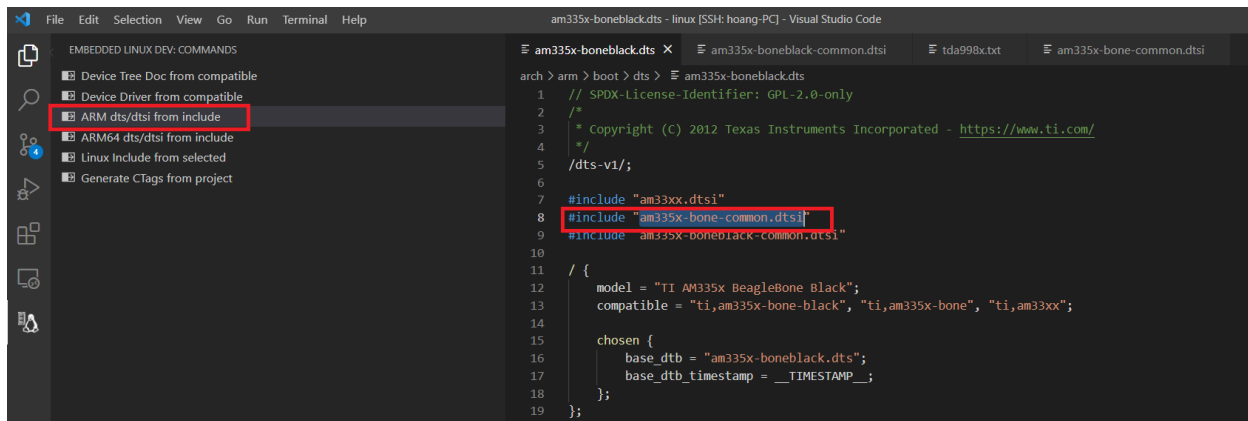
Example: Choose the compatible string (*nxp,tda998x*) then click the *Device Tree Doc from compatible* on Control panel. It will be pointed to the *tda998x.txt* Doc file that describes this device tree node.



- Find device driver from compatible string



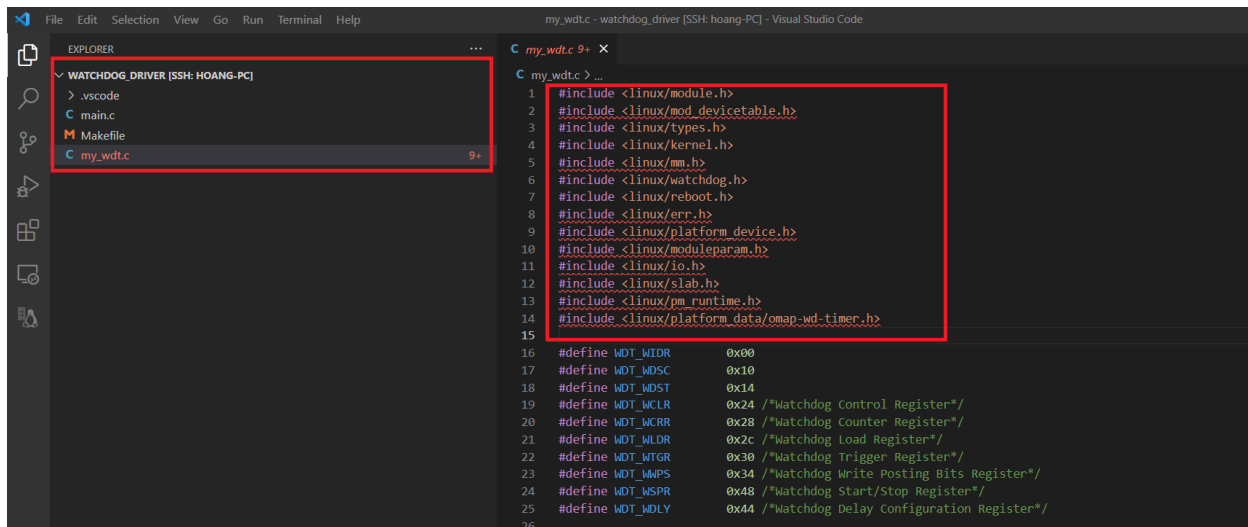
- Find the ARM dts/dtsi from include



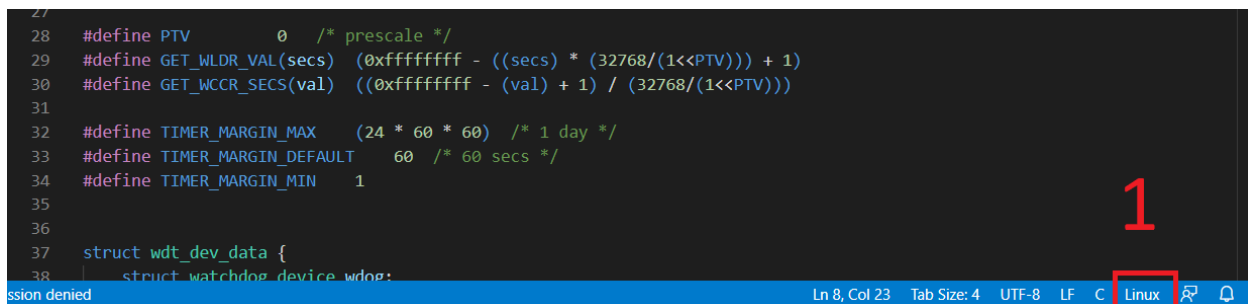
And some other useful tools. It also applies to the u-boot source code.

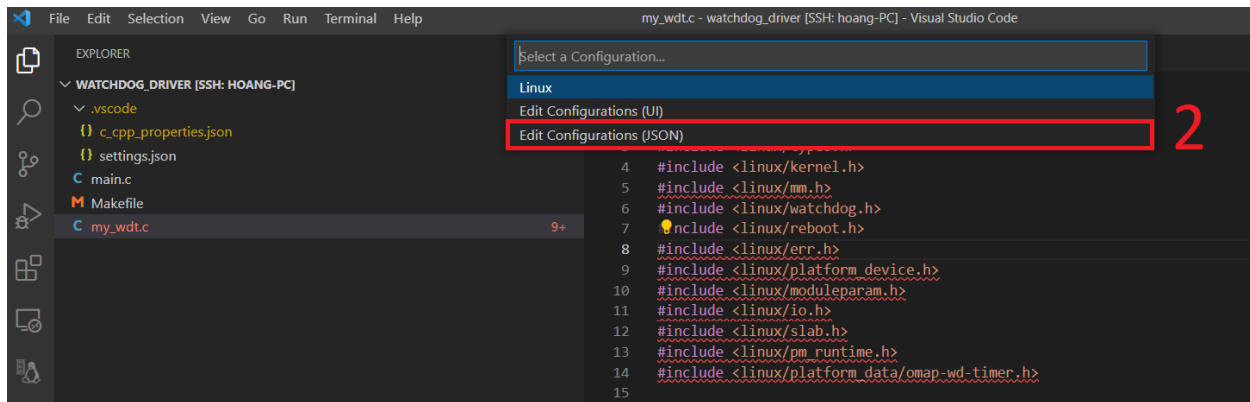
5.2. Build an out-of-tree kernel module

Example: open a watchdog driver source code:



- Configure the C/C++ configuration file to get the correct Linux kernel headers.





Modify the `c_cpp_properties.json` file:

```
"includePath": [
    "${workspaceFolder}/**"
],
```

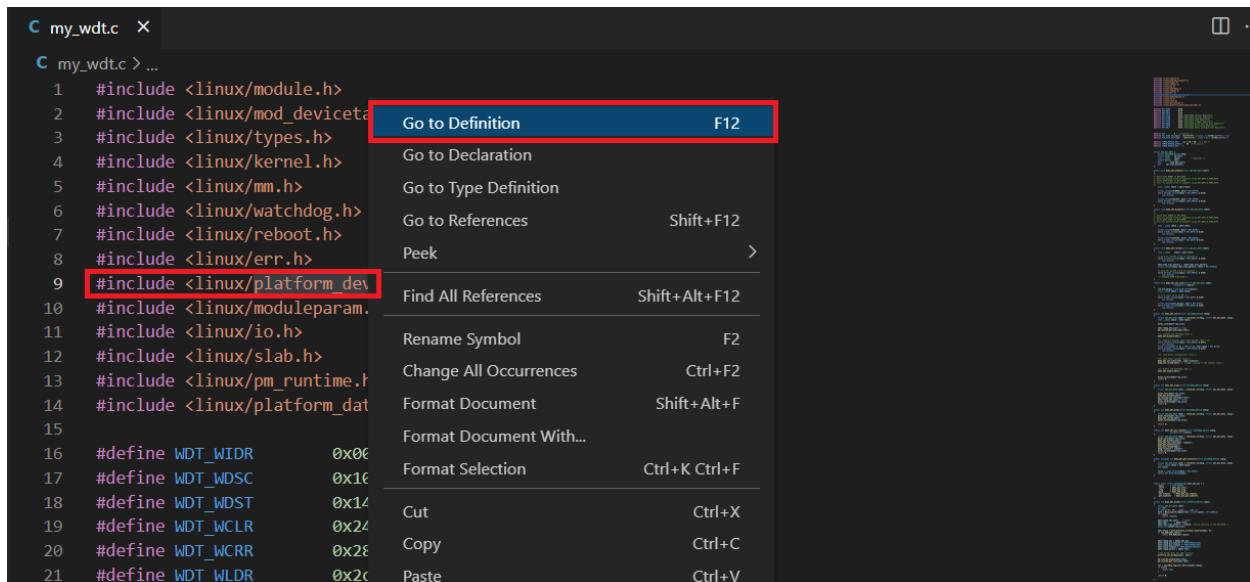
=>

```
"configurations": [
  {
    "name": "Linux",
    "includePath": [
      "/source/linux/**"
    ],
    "defines": [
      "__GNUC__",
      "__KERNEL__"
    ],
    "forcedInclude": [
      "/source/linux/arch/powerpc/boot/types.h"
    ],
    "compilerPath": "/usr/bin/gcc",
    "cStandard": "c99",
    "cppStandard": "gnu++14",
    "intelliSenseMode": "linux-gcc-x64"
  }
]
```

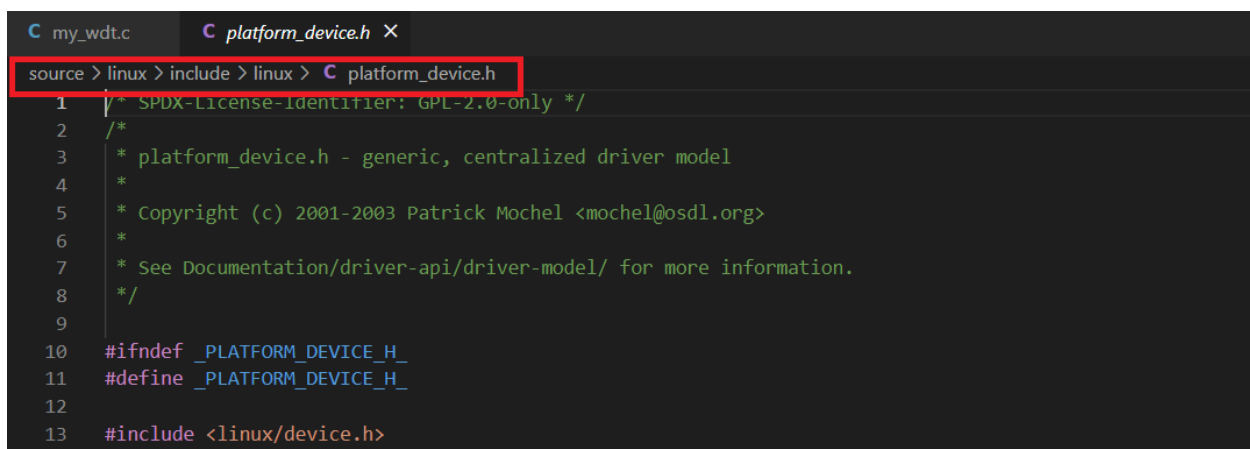
Note: `/source/linux` is the directory that you used to store the Linux source code.

Now you can point to correct the kernel header file or function prototype from your code.

- Find the kernel header definition



It will be automatically switched to a new tab window:



- Find and view function prototypes or variable definitions

```
C my_wdt.c • C platform_device.h
C my_wdt.c > demo_wdt_enable(wdt_dev_data *)
44 };
45
46 #define writel_relaxed writel_relaxed
47 static inline void writel_relaxed(u32 value, volatile void __iomem *addr)
48 {
49     __raw_writel(__cpu_to_le32(value), addr);
50 }
51 #define writel_relaxed writel_relaxed
52 4. P
53 /*
54    SPDX-License-Identifier: GPL-2.0-or-later
55    Expands to:
56    writel_relaxed
57    writel_relaxed(0xB8BB, base + WDT_WSPR);
58    while ((readl_relaxed(base + WDT_WWPS) & 0x10)
59        cpu_relax());
60    writel_relaxed(0x4444, base + WDT_WSPR);
61    while ((readl_relaxed(base + WDT_WWPS) & 0x10)
62        cpu_relax());
63 }
64
```

- Automated typing in Visual Studio Code

```
248 static void demo_wdt_shutdown(struct platform_device *pdev)
249 {
250     struct wdt_dev_data *wdev = platform_get_drvdata(pdev);
251     printk
252     mutex_
253     if (wd
254     de
255     pm
256     }
257     mutex_
258     static int
259     {
260     struct wdt_dev_data *wdev = platform_get_drvdata(pdev);
261     pr_emerg("%s, %d", __func__, __LINE__);
262     watchdog_unregister_device(&wdev->wdog);
263     return 0;
264 }
265
266
267
268
```

Note: to quickly jump to the definition, you can press Ctrl + Click on the object you want to find the definition.

```

wdev->base = devm_platform_ioremap_resource(&pdev->dev);
if (IS_ERR(wdev->base))
    return PTR_ERR(wdev->base);

wdev->wdog.ops = &demo_wdog_ops;
wdev->wdog.min_timeout = (24 * 60 * 60);
wdev->wdog.max_timeout = TIMER_MARGIN_MAX;
wdev->wdog.timeout = TIMER_MARGIN_DEFAULT;
wdev->wdog.parent = &pdev->dev;

```

```

#define TIMER_MARGIN_MAX (24 * 60 * 60) /* 1 day */
#define TIMER_MARGIN_DEFAULT 60 /* 60 secs */
#define TIMER_MARGIN_MAX (24 * 60 * 60)

```

Expands to:

press Ctrl + click mouse on object

VI. Debug Kernel for BBB by command line on VSCode

Prerequisites

Connect the BBB to the host via serial connection, in the example below, /dev/ttyUSB0 will be used as the detected serial device on the host.

The kernel to be debugged should be configured with KGDB support. Configure the kernel and re-build the kernel with the following included.

```

# CONFIG_STRICT_KERNEL_RWX is not set
CONFIG_FRAME_POINTER=y
CONFIG_KGDB=y
CONFIG_KGDB_SERIAL_CONSOLE=y
CONFIG_KGDB_KDB=y
CONFIG_KDB_KEYBOARD=y

```

The kernel also must be compiled with this macro enabled.

```

[*] General setup -> Load all symbols for debugging / ksymoops -> Include all symbols in kallsyms

```

Boot Arguments

```

=> setenv bootargs console=ttyO0,115200n8 root=/dev/mmcblk0p2 ro rootfstype=ext4 rootwait debug
earlyprintk mem=512M kgdboc=ttyO0,115200 kgdbwait

```

```

=> printenv bootargs

```

```

bootargs=console=ttyO0,115200n8 root=/dev/mmcblk0p2 ro rootfstype=ext4 rootwait debug
earlyprintk mem=512M kgdboc=ttyO0,115200 kgdbwait

```

Debugging a kernel module using KGDB

- From BBB:

```

echo "ttyS0,115200" > /sys/module/kgdboc/parameters/kgdboc
echo g > /proc/sysrq-trigger

```



```

root@beaglebone:/home/debian# echo ttyS0 > /sys/module/kgdboc/parameters/kgdboc
[ 200.874769] KGDB: Registered I/O driver kgdboc
root@beaglebone:/home/debian#
root@beaglebone:/home/debian# echo g > /proc/sysrq-trigger
[ 204.842355] sysrq: DEBUG

Entering kdb (current=0xdaf14000, pid 2504) on processor 0 due to Keyboard Entry
[0]kdb>

```

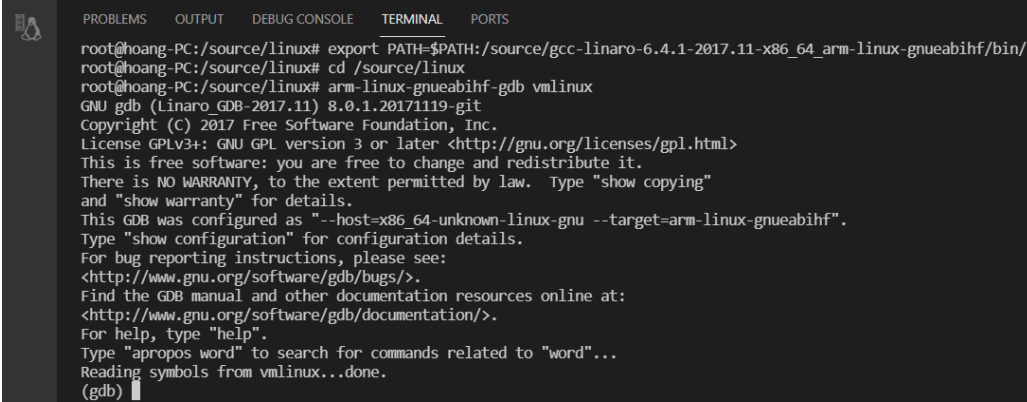
If you are using the same serial line for both console and debug, now is the time to disconnect the terminal application (e.g. minicom). Then, start the host debugger using the vmlinux elf from your Linux build as input file. Once connected, you can debug a kernel the way you would debug an application program.

- From VSCode's terminal (Connected to Host through SSH remote):

```

export PATH=$PATH:/source/gcc-linaro-6.4.1-2017.11-x86_64_arm-linux-gnueabi/bin/
cd /source/linux
arm-linux-gnueabi-gdb vmlinux

```



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
root@hoang-PC:/source/linux# export PATH=$PATH:/source/gcc-linaro-6.4.1-2017.11-x86_64_arm-linux-gnueabi/bin/
root@hoang-PC:/source/linux# cd /source/linux
root@hoang-PC:/source/linux# arm-linux-gnueabi-gdb vmlinux
GNU gdb (Linaro_GDB-2017.11) 8.0.1.20171119-git
Copyright (C) 2017 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=x86_64-unknown-linux-gnu --target=arm-linux-gnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from vmlinux...done.
(gdb)

```

- Now setup GDB:

```

(gdb) set serial baud 115200
(gdb) target remote /dev/ttyUSB0
(gdb) set serial baud 115200
(gdb) target remote /dev/ttyUSB1
Remote debugging using /dev/ttyUSB1
0xc01292c4 in arch_kgdb_breakpoint () at kernel/debug/debug_core.c:1137
1137          wmb(); /* Sync point before breakpoint */
(gdb)

```

- Step Debugging:

Example: Debugging a GPIO controller driver, in BBB this is *gpio-omap.c*, where *omap_gpio_get_direction* will be called anytime we export a pin as an GPIO from User Space:

```

(gdb) br gpio-omap.c:omap_gpio_get_direction
Breakpoint 1 at 0xc070b0c4: file drivers/gpio/gpio-omap.c, line 805.
(gdb) c
Continuing.

```

Now on the Host machine, setup an SSH connection with the target board after boot process is complete and export a pin as GPIO

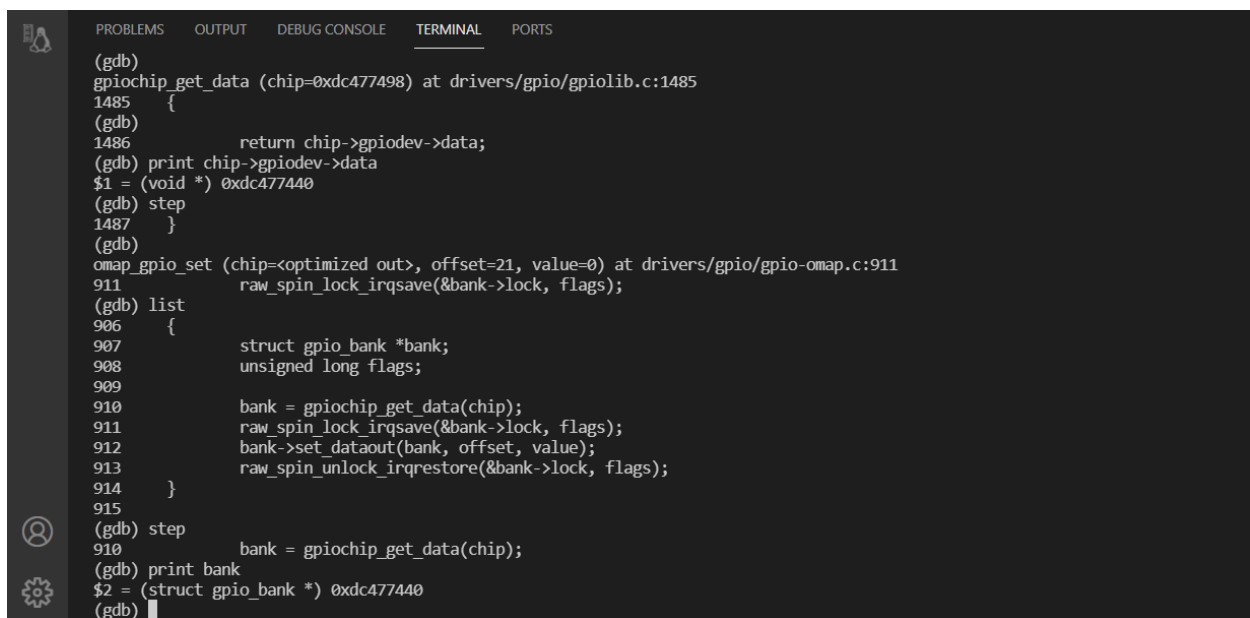
```
root@beaglebone:/home/debian# echo g > /proc/sysrq-trigger
root@beaglebone:/home/debian# echo 10 > /sys/class/gpio/export
```

Now the Kernel execution should stop at drivers/gpio/gpio-omap.c:

```
(gdb) br gpio-omap.c:omap_gpio_get_direction
Breakpoint 1 at 0xc070b0c4: file drivers/gpio/gpio-omap.c, line 805.
(gdb) c
Continuing.
[New Thread 2711]
[New Thread 2713]

Thread 107 hit Breakpoint 1, omap_gpio_get_direction (chip=0xdc457298, offset=10) at drivers/gpio/gpio-omap.c:805
805 {
(gdb)
```

Now you can step through the any function in the module and add breakpoints in the module source.



```
(gdb)
gpiochip_get_data (chip=0xdc477498) at drivers/gpio/gpiolib.c:1485
1485 {
(gdb)
1486     return chip->gpiodev->data;
(gdb) print chip->gpiodev->data
$1 = (void *) 0xdc477440
(gdb) step
1487 }
(gdb)
omap_gpio_set (chip=<optimized out>, offset=21, value=0) at drivers/gpio/gpio-omap.c:911
911     raw_spin_lock_irqsave(&bank->lock, flags);
(gdb) list
906 {
907     struct gpio_bank *bank;
908     unsigned long flags;
909
910     bank = gpiochip_get_data(chip);
911     raw_spin_lock_irqsave(&bank->lock, flags);
912     bank->set_dataout(bank, offset, value);
913     raw_spin_unlock_irqrestore(&bank->lock, flags);
914 }
915
(gdb) step
910     bank = gpiochip_get_data(chip);
(gdb) print bank
$2 = (struct gpio_bank *) 0xdc477440
(gdb)
```

Reference:

1. <https://code.visualstudio.com/docs>
2. <https://www.kernel.org/doc/html/v5.4/>
3. <https://www.kernel.org/doc/html/v5.4/dev-tools/kgdb.html>
4. [https://linuxlink.timesys.com/docs/how to use kgdb](https://linuxlink.timesys.com/docs/how_to_use/kgdb)
5. ...