# Procedure to Debug using GDB

Following is the procedure to debug the VM-based applications using GDB:

1. Open the SDK folder in Ubuntu terminal and type the following command to start OpenOCD:

|  |
| --- |
| sudo openocd -s ./conf -f ftdi.cfg -f t2.cfg |

Console output:

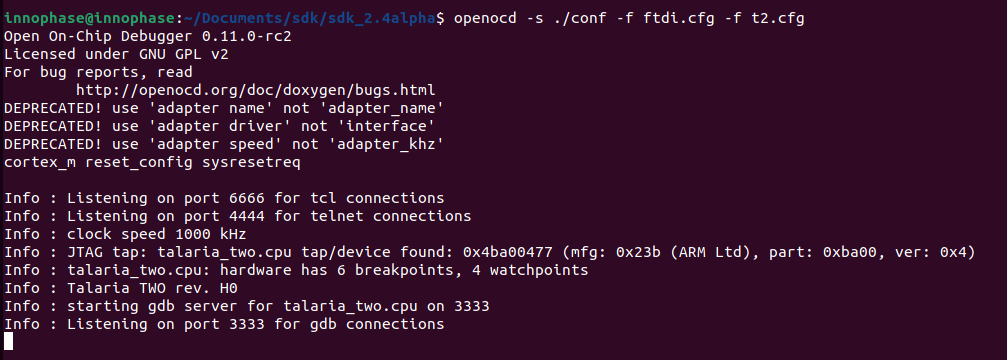


Figure 3: Starting openOCD

1. In a separate terminal, flash the virtual image from the SDK directory.

For example: Consider wifi\_connect.elf.

|  |
| --- |
| ./script/boot.py --reset=evk42\_bl --speed=2457600 --gdb examples/using\_wifi/out/wifi\_connect.elf |

**Note**: For the GDB to work, ELF needs to be loaded. By default, the SDK package contains ELF files in the bin folder (which are stripped ELF files). Hence, user needs to build the sample application, generate the ELF file (by default, it gets generated in the out folder) and load this ELF for debugging.

Execute make for using\_wifi example application (sdk\_2.4/examples/using\_wifi) to generate the ELFs under the out folder.

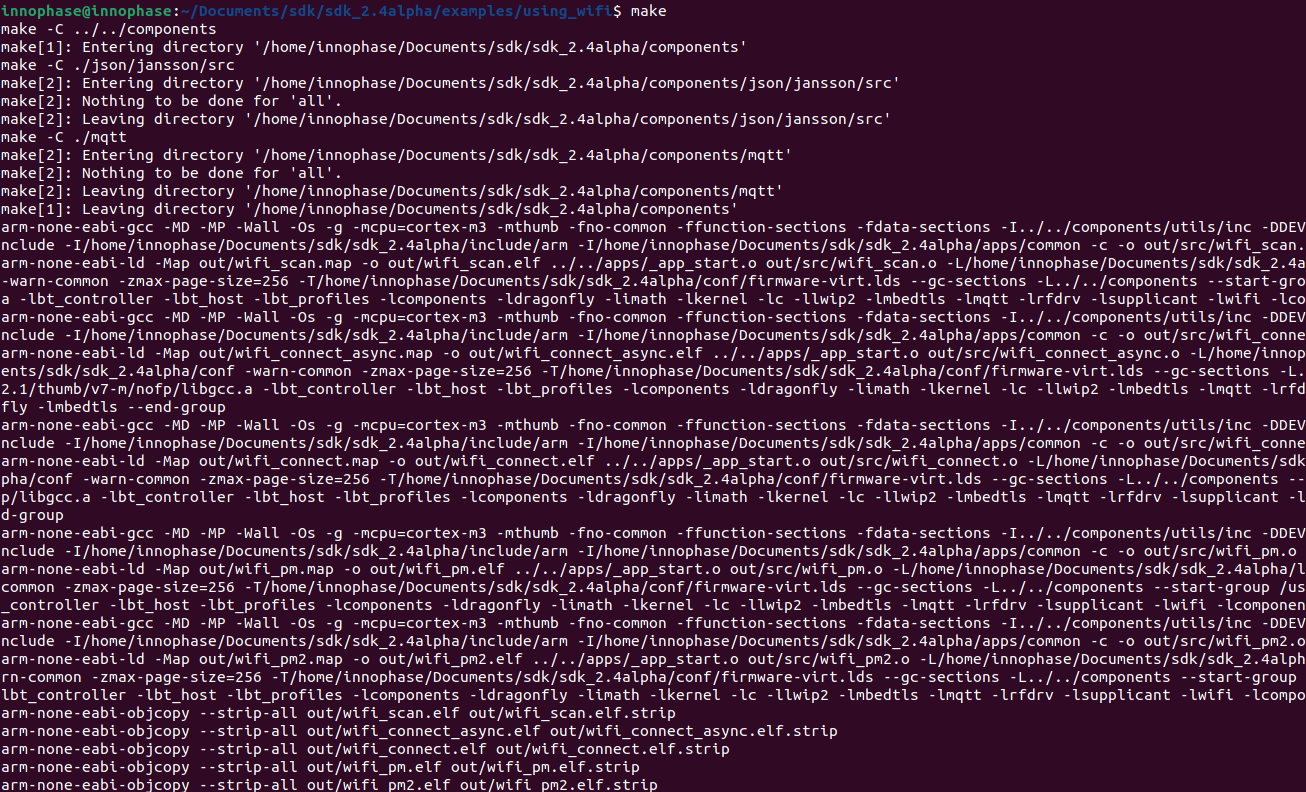


Figure 4: make command output for using\_wifi

Following is the output for a sample application wifi\_connect.elf. The total number of bytes displayed in Figure 5 will vary with the ELF of the application being flashed.

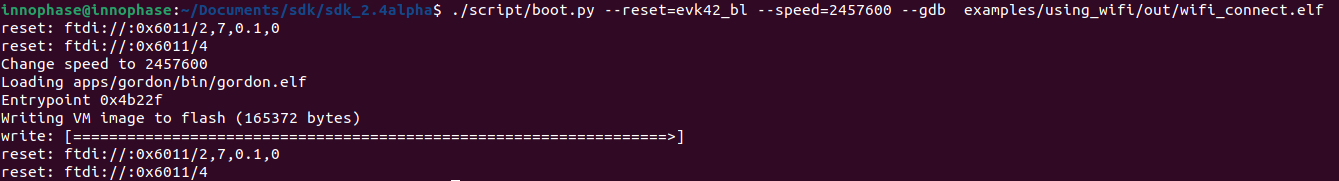


Figure 5: Flashing the application from SDK directory

.gdbinit initialization file contains the information on Talaria TWO’s memory regions and the required scripts of the GDB sources. gdbinit file is present under the apps folder. To start the GDB session, gdb-multiarch should be started from this folder.

Command line method of configuring the gdbinit file

GDB method relies on the GDB scripts. Initially, GDB needs to be configured to allow auto-load.

|  |
| --- |
| echo "set auto-load safe-path /" > ~/.gdbinit |

Manual method of configuring the gdbinit file:

If there are any warnings as shown in Figure 6, the gdb-multiarch does not work for GDB commands. Hence, create a file named gdbinit in the home directory to allow auto-load.

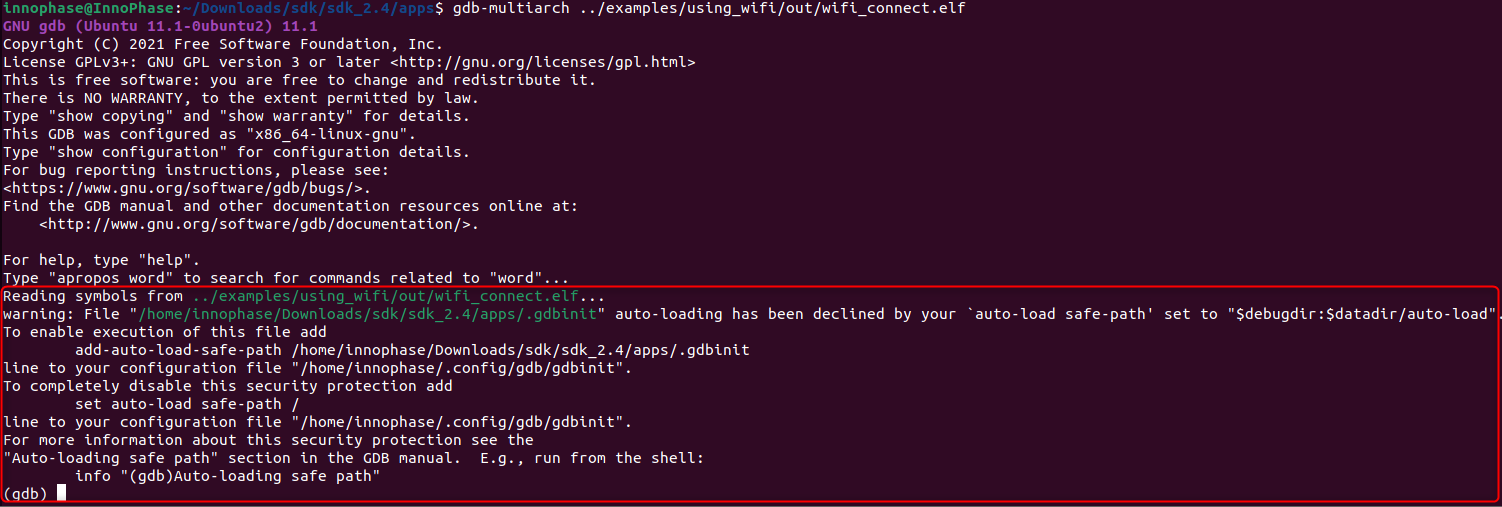


Figure 6: Warning for .gdbinit file

In the created gdbinit file add the following path:

add-auto-load-safe-path/home/innophase/Downloads/sdk/sdk\_2.4/apps/.gdbinit.

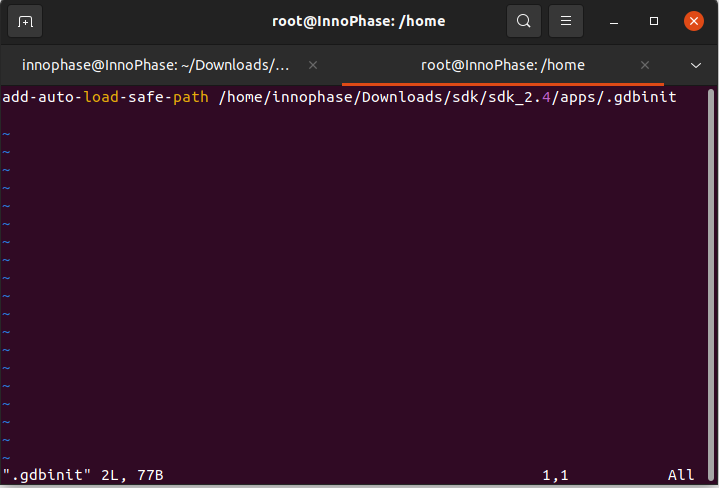


Figure 7: Configuring the gdbinit file

## Start the GDB Session

In a separate terminal, run the following command from the sdk\_x.y/apps directory. In this directory, there is a .gdbinit file that configures the GDB. Here, the RAM portion of the ELF gets loaded.

**Note**: x and y in sdk\_x.y refer to the SDK release version.

|  |
| --- |
| gdb-multiarch ../examples/using\_wifi/out/wifi\_connect.elf |

Console output:

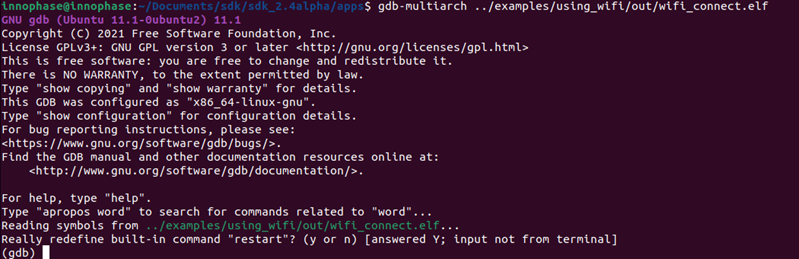


Figure 8: Running GDB

1. Connect to OpenOCD by running ocd in the GDB prompt.

|  |
| --- |
| ocd |

1. Set a break point at main:

|  |
| --- |
| b main |

1. Run the application by executing:

|  |
| --- |
| R |

1. The information on the break points set can be seen by issuing:

|  |
| --- |
| info b |

1. A break point at a line number of a particular source file can be set using:

|  |
| --- |
| b <filename>: <linenum> |

1. If the line to be executed is a function call, GDB will step into that function and start executing its code one line at a time.

|  |
| --- |
| s |

1. If the entire function needs to be executed with one keypress, type next or n.

|  |
| --- |
| next |

1. Continue running the program (after stopping, for example at a breakpoint).

|  |
| --- |
| continue |

1. Step out is the operation that resumes execution after the function the program is executing terminates. The debugger will stop at the statement after the function call.

|  |
| --- |
| finish |

Example 1: Following is the output while debugging the wifi\_connect.elf using GDB:

|  |
| --- |
| (gdb) ocd  0x00023f36 in ?? ()  (gdb) b main  Breakpoint 1 at 0x100020: file wifi\_connect/main.c, line 23.  Note: automatically using hardware breakpoints for read-only addresses.  (gdb) R  JTAG tap: talaria\_two.cpu tap/device found: 0x4ba00477 (mfg: 0x23b (ARM Ltd), part: 0xba00, ver: 0x4)  target halted due to debug-request, current mode: Thread  xPSR: 0x01000000 pc: 0x00020f90 msp: 0x00041a78  Loading section .text, size 0x1c538 lma 0x42000  Loading section .data, size 0x490 lma 0x5e538  Loading section .virt, size 0x24658 lma 0x2000000  Start address 0x0004ea1e, load size 266272  Transfer rate: 71 KB/sec, 14014 bytes/write.Breakpoint 1, main () at wifi\_connect/main.c:23  23 const char \*ssid = os\_get\_boot\_arg\_str("ssid");  (gdb) info b  Num Type Disp Enb Address What  1 breakpoint keep y 0x00100020 in main at wifi\_connect/main.c:23  breakpoint already hit 1 time  (gdb) del 1  (gdb) info b  No breakpoints or watchpoints.  (gdb) b main.c:29  Breakpoint 2 at 0x10003c: file wifi\_connect/main.c, line 29.  (gdb) R  JTAG tap: talaria\_two.cpu tap/device found: 0x4ba00477 (mfg: 0x23b (ARM Ltd), part: 0xba00, ver: 0x4)  target halted due to debug-request, current mode: Thread  xPSR: 0x01000000 pc: 0x00020f90 msp: 0x00041a78  Loading section .text, size 0x1c538 lma 0x42000  Loading section .data, size 0x490 lma 0x5e538  Loading section .virt, size 0x24658 lma 0x2000000  Start address 0x0004ea1e, load size 266272  Transfer rate: 71 KB/sec, 14014 bytes/write.Breakpoint 2, main () at wifi\_connect/main.c:29  29 os\_printf("Need to specify ssid and passphrase boot arguments\n");  (gdb) |

Example 2 : Following is the output while debugging the wcma.elf using GDB:

|  |
| --- |
| (gdb) ocd  0x00023f36 in ?? ()  (gdb) b wcma\_test.c:73  Breakpoint 1 at 0x10038c: file src/wcma\_test.c, line 99.  Note: automatically using hardware breakpoints for read-only addresses.  (gdb) R  JTAG tap: talaria\_two.cpu tap/device found: 0x4ba00477 (mfg: 0x23b (ARM Ltd), part: 0xba00, ver: 0x4)  target halted due to debug-request, current mode: Thread  xPSR: 0x01000000 pc: 0x00020f90 msp: 0x00041a78  Loading section .text, size 0x19f78 lma 0x42000  Loading section .data, size 0x470 lma 0x5bf78  Loading section .virt, size 0x1bcb4 lma 0x2000000  Start address 0x0004cb8e, load size 221340  Transfer rate: 58 KB/sec, 13833 bytes/write.  Breakpoint 1, wcma\_thread (arg=<optimized out>) at src/wcma\_test.c:99  99 wcma\_scan\_retry(h, 3, &ap\_manager);  (gdb) s  wcma\_scan\_retry (handle=0xbf8b0, retries=retries@entry=3, manager=manager@entry=0x5c3e8 <ap\_manager>) at src/wcma\_test.c:318  318 scan\_result = os\_alloc(max\_nets \* sizeof(void \*));  (gdb) b wcma\_test.c:109  Breakpoint 2 at 0x1003a8: file src/wcma\_test.c, line 109.  (gdb) R  JTAG tap: talaria\_two.cpu tap/device found: 0x4ba00477 (mfg: 0x23b (ARM Ltd), part: 0xba00, ver: 0x4)  target halted due to debug-request, current mode: Thread  xPSR: 0x01000000 pc: 0x00020f90 msp: 0x00041a78  Loading section .text, size 0x19f78 lma 0x42000  Loading section .data, size 0x470 lma 0x5bf78  Loading section .virt, size 0x1bcb4 lma 0x2000000  Start address 0x0004cb8e, load size 221340  Transfer rate: 58 KB/sec, 13833 bytes/write.  Breakpoint 1, wcma\_thread (arg=<optimized out>) at src/wcma\_test.c:99  99 wcma\_scan\_retry(h, 3, &ap\_manager);  (gdb) info b  Num Type Disp Enb Address What  1 breakpoint keep y 0x0010038c in wcma\_thread at src/wcma\_test.c:99  breakpoint already hit 2 times  2 breakpoint keep y 0x001003a8 in wcma\_thread at src/wcma\_test.c:109  (gdb) continue  Continuing.  Breakpoint 2, wcma\_thread (arg=<optimized out>) at src/wcma\_test.c:109  109 if(connection\_status == AP\_DISCONNECTED && reconnect\_next\_ap)  (gdb) next  116 if((os\_systime() - last\_disconnect\_time) > 35000000) /\* 35 seconds \*/ |