BAREMETAL PROGRAMMING FOR ARM TARGET MCU USING GNU TOOLS AND USE ECLIPSE AS IDE

This documentation is created by VKSALIAN for practising programming of ARM MCU and then using Eclipse IDE for further development without using any official IDE. Here all the development environments are kept simple and are setup from scratch. The below are the details of setup.

OS	Ubuntu 18.04 LTS
Code Editor Used	Visual Studio Code v1.53.2
Cross−Compiler Toolchain	arm−none−eabi−gcc 10.2.1 20201103
GNU Make Tool	V4.1
MCU	TM4C123GH6PM
MCU Evaluation Board	Tiva C series TM4C123G (MCU:TM4C123GH6PM)Launchpad
MCU Architecture	ARM Cortex M4

Below are the things to be achieved,

- Understand the compilation of C program for ARM embedded target.
- Learn how to write microcontroller startup file.
- Learn how to write 'C' startup code which will then call main().
- Understand different sections of relocatable object (.o) files.
- Learn how to write linker script from scratch and section placements.
- Learn how to link multiple .o files using linker script and convert into one executable file (.elf, bin, hex).
- Use of OpenOCD and GDB for debugging.
- Port the project into Eclipse IDE and develop further.

Note: This documentation assumes that you have no development tools installed earlier. If any tools are already installed, you may continue with that or uninstall all of them and follow the this documentation.

Step 1: Installation of toolchain

We have Linux host machine (in our case) on which we have to create executables that can run on different machine (ARM MCU) using cross-toolchain. If we are creating executables for different machine, then it is called cross compilation. So, we need to install cross-compiler toolchains which are nothing but collection of binaries which allow to compile, assemble and link applications. It also comes with binaries that can be used to debug the application, convert the format of the application, disassemble the application, get the size information of the various sections inside the application, etc.

For ARM, we can use two available tool-chains,

1. GNU Tools for ARM embedded processors. (free and opensource).
2. armcc from ARM. (requires licensing).

We are using **GNU** Tools for ARM embedded processors here.

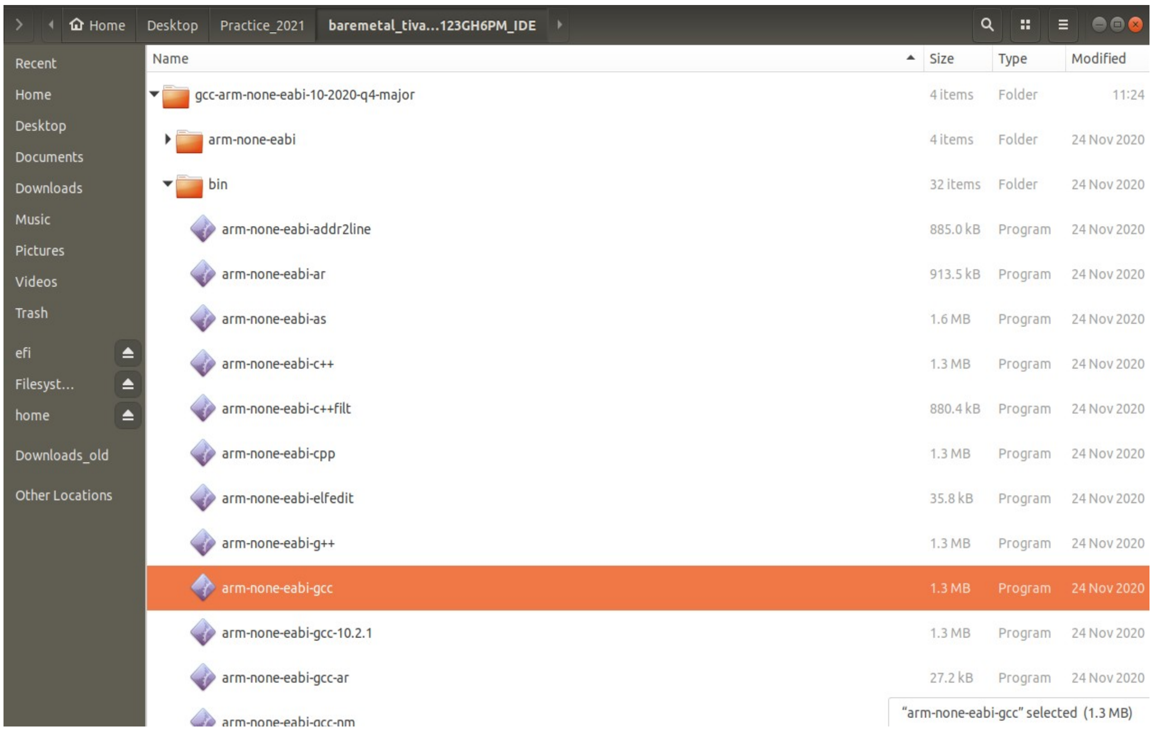
1.0 **Install GNU Tools for ARM embedded processors**:

- Goto the site https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads.
- Download the latest release x86_64 for linux: **gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2**
- Extract the above .tar.bz2 file.
- Copy the extarcted folder to some known location. I copied into "/home/vksalian/Desktop/Practice_2021/baremetal_tiva_TM4C123GH6PM_IDE" folder.
- Now, in order to add the bin folder into the global path (so that each compiler binaries can be called directly from any path) just edit the ~/**.bashrc** file and add the below line.

```
export
PATH=/home/vksalian/Desktop/Practice_2021/baremetal_tiva_TM4C123GH6PM_IDE
/gcc-arm-none-eabi-10-2020-q4-major/bin:$PATH
```

- Then, issue command “**source ~/.bashrc**“ and restart the terminal window. Now, you can issue commands like arm–none–eabi–gcc, arm–none–eabi–gdb, etc.

The tree view of the folder is as given below,



Home | Desktop | Practice_2021 | baremetal_tiva...123GH6PM_IDE

Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
efi
Filesyst...
home
Downloads_old
Other Locations

Name	Size	Type	Modified
gcc-arm-none-eabi-10-2020-q4-major	4 items	Folder	11:24
arm-none-eabi	4 items	Folder	24 Nov 2020
bin	32 items	Folder	24 Nov 2020
arm-none-eabi-addr2line	885.0 kB	Program	24 Nov 2020
arm-none-eabi-ar	913.5 kB	Program	24 Nov 2020
arm-none-eabi-as	1.6 MB	Program	24 Nov 2020
arm-none-eabi-c++	1.3 MB	Program	24 Nov 2020
arm-none-eabi-c++filt	880.4 kB	Program	24 Nov 2020
arm-none-eabi-cpp	1.3 MB	Program	24 Nov 2020
arm-none-eabi-elfedit	35.8 kB	Program	24 Nov 2020
arm-none-eabi-g++	1.3 MB	Program	24 Nov 2020
arm-none-eabi-gcc	1.3 MB	Program	24 Nov 2020
arm-none-eabi-gcc-10.2.1	1.3 MB	Program	24 Nov 2020
arm-none-eabi-gcc-ar	27.2 kB	Program	24 Nov 2020
arm-none-eabi-gcc-nm			

“arm-none-eabi-gcc” selected (1.3 MB)

The important binaries of cross–compiler are,

Compiler, Assembler and Linker	arm–none–eabi–gcc
Linker	arm–none–eabi–gcc
Assembler	Arm–none–eabi–as
Elf File Analyzer	arm–none–eabi–objdump arm–none–eabi–readelf arm–none–eabi–nm
Format Converter	arm–none–eabi–objcopy

1.1 **Install GNU make**:

GNU make is a tool which controls the generation of executables and other non–source files of a program from the program’s source files. Make gets its knowledge of how to build your program from a file called the **makefile**. We have to write the rules how the source files in project folder are compiled and output generated.

In Ubuntu Linux, we can install GNU Make tool using terminal command as below,

```
sudo apt install make
```

To check the version of installed GNU make tool,

```
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.
vksalian@vksalian-HP-Notebook:~$
```

1.2 **Install OpenOCD**:

To download executable you need to connect **target** (TIVAC123G Launchpad) to the **PC or Host** via **Debug Adaptor**. The programming through debug adaptor is also know as **in–circuit** programming / Debugging. The Debug Adaptor is also known as in–circuit programmer/debugger. The debug adaptor will convert the host protocol to native target protocol. e.g. a target might support SWD or JTAG based debug protocol (target interface) and it understands only the supported protocols, where as a Host understands interfaces such as USB or serialport or parallel port. The debug adaptor converts host interface to target interface protocol. On host side you have to run some application that talks to the debug adaptor and there by you can talk to the target by sending some commands to the target. One such application is **openocd**.

- The **openocd** stands for open on–chip debugger which aims to provide debugging, in–system programming and boundary–scan testing for embedded target devices.

- It is free and opensource application that allows you to program, debug, and analyze your application using GDB.
- It supports various target boards based on different processor architecture. OpenOCD supports many types of debug adapters such as USB–based, parallel port–based, and other standalone boxes that run openOCD internally.
- You can debug various CPU architectures via GDB protocol.

It allows you to program the internal flash of the microcontroller using the host application openOCD. To install **openocd** tool under Ubuntu Linux, issue below command in terminal.

```
sudo apt install openocd
```

To check the version of installed OpenOCD tool,

```
openocd -v
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
vksalian@vksalian-HP-Notebook:~$
```

1.3 **Set USB permissions**:

The OpenOCD supports TIVAC123G Launchpad’s ICDI (In–circuit Debug Interface) using libusb drivers. But by default, the OpenOCD needs permission to interact with the USB devices as normal user. So use the below file named “60–**openocd.rules**”.

```
# Copy this file to /etc/udev/rules.d/
# If rules fail to reload automatically, you can refresh udev rules
# with the command "udevadm control --reload"

ACTION!="add|change", GOTO="openocd_rules_end"

SUBSYSTEM=="gpio", MODE="0660", GROUP="plugdev", TAG+="uaccess"

SUBSYSTEM!="usb|tty|hidraw", GOTO="openocd_rules_end"

# Please keep this list sorted by VID:PID

# opendous and estick
ATTRS{idVendor}=="03eb",            ATTRS{idProduct}=="204f",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Original FT232/FT245 VID:PID
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="6001",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Original FT2232 VID:PID
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="6010",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Original FT4232 VID:PID
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="6011",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Original FT232H VID:PID
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="6014",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# DISTORTEC JTAG-lock-pick Tiny 2
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="8220",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TUMPA, TUMPA Lite
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="8a98",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="8a99",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# XDS100v2
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="a6d0",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Xverve Signalyzer Tool (DT-USB-ST), Signalyzer LITE (DT-USB-SLITE)
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="bca0",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="bca1",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TI/Luminary Stellaris Evaluation Board FTDI (several)
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="bcd9",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TI/Luminary Stellaris In-Circuit Debug Interface FTDI (ICDI) Board
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="bcda",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# egnite Turtelizer 2
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="bdc8",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Section5 ICEbear
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="c140",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="c141",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Amontec JTAGkey and JTAGkey-tiny
ATTRS{idVendor}=="0403",            ATTRS{idProduct}=="cff8",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
```

```
# Nuvoton NuLink
ATTRS{idVendor}=="0416", ATTRS{idProduct}=="511b", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0416", ATTRS{idProduct}=="511c", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0416", ATTRS{idProduct}=="511d", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0416", ATTRS{idProduct}=="5200", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0416", ATTRS{idProduct}=="5201", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TI ICDI
ATTRS{idVendor}=="0451", ATTRS{idProduct}=="c32a", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# STMicroelectronics ST-LINK V1
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="3744", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# STMicroelectronics ST-LINK/V2
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="3748", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# STMicroelectronics ST-LINK/V2.1
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="374b", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="3752", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# STMicroelectronics STLINK-V3
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="374d", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="374e", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="374f", MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0483", ATTRS{idProduct}=="3753", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Cypress SuperSpeed Explorer Kit
ATTRS{idVendor}=="04b4", ATTRS{idProduct}=="0007", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Cypress KitProg in KitProg mode
ATTRS{idVendor}=="04b4", ATTRS{idProduct}=="f139", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Cypress KitProg in CMSIS-DAP mode
ATTRS{idVendor}=="04b4", ATTRS{idProduct}=="f138", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Infineon DAP miniWiggler v3
ATTRS{idVendor}=="058b", ATTRS{idProduct}=="0043", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Hilscher NXHX Boards
ATTRS{idVendor}=="0640", ATTRS{idProduct}=="0028", MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Hitex STR9-comStick
ATTRS{idVendor}=="0640",           ATTRS{idProduct}=="002c",           MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Hitex STM32-PerformanceStick
ATTRS{idVendor}=="0640",           ATTRS{idProduct}=="002d",           MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Altera USB Blaster
ATTRS{idVendor}=="09fb",           ATTRS{idProduct}=="6001",           MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Amontec JTAGkey-HiSpeed
ATTRS{idVendor}=="0fbb",           ATTRS{idProduct}=="1000",           MODE="660",
GROUP="plugdev", TAG+="uaccess"

# SEGGER J-Link
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0101",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0102",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0103",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0104",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0105",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0107",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="0108",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1010",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1011",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1012",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1013",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1014",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1015",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1016",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1017",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1018",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1051",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1366",           ATTRS{idProduct}=="1061",           MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Raisonance RLink
ATTRS{idVendor}=="138e",           ATTRS{idProduct}=="9000",           MODE="660",
GROUP="plugdev", TAG+="uaccess"
```

```
# Debug Board for Neo1973
ATTRS{idVendor}=="1457",            ATTRS{idProduct}=="5118",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Olimex ARM-USB-OCD
ATTRS{idVendor}=="15ba",            ATTRS{idProduct}=="0003",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Olimex ARM-USB-OCD-TINY
ATTRS{idVendor}=="15ba",            ATTRS{idProduct}=="0004",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Olimex ARM-JTAG-EW
ATTRS{idVendor}=="15ba",            ATTRS{idProduct}=="001e",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Olimex ARM-USB-OCD-TINY-H
ATTRS{idVendor}=="15ba",            ATTRS{idProduct}=="002a",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Olimex ARM-USB-OCD-H
ATTRS{idVendor}=="15ba",            ATTRS{idProduct}=="002b",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# USBprog with OpenOCD firmware
ATTRS{idVendor}=="1781",            ATTRS{idProduct}=="0c63",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TI/Luminary Stellaris In-Circuit Debug Interface (ICDI) Board
ATTRS{idVendor}=="1cbe",            ATTRS{idProduct}=="00fd",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TI XDS110 Debug Probe (Launchpads and Standalone)
ATTRS{idVendor}=="0451",            ATTRS{idProduct}=="bef3",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="0451",            ATTRS{idProduct}=="bef4",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="1cbe",            ATTRS{idProduct}=="02a5",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# TI Tiva-based ICDI and XDS110 probes in DFU mode
ATTRS{idVendor}=="1cbe",            ATTRS{idProduct}=="00ff",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# PLS USB/JTAG Adapter for SPC5xxx
ATTRS{idVendor}=="263d",            ATTRS{idProduct}=="4001",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Ambiq Micro EVK and Debug boards.
ATTRS{idVendor}=="2aec",            ATTRS{idProduct}=="6010",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="2aec",            ATTRS{idProduct}=="6011",            MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="2aec",            ATTRS{idProduct}=="1106",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Marvell Sheevaplug
ATTRS{idVendor}=="9e88",            ATTRS{idProduct}=="9e8f",            MODE="660",
GROUP="plugdev", TAG+="uaccess"

# Keil Software, Inc. ULink
ATTRS{idVendor}=="c251",          ATTRS{idProduct}=="2710",          MODE="660",
GROUP="plugdev", TAG+="uaccess"
ATTRS{idVendor}=="c251",          ATTRS{idProduct}=="2750",          MODE="660",
GROUP="plugdev", TAG+="uaccess"

# CMSIS-DAP compatible adapters
ATTRS{product}=="*CMSIS-DAP*", MODE="660", GROUP="plugdev", TAG+="uaccess"

LABEL="openocd_rules_end"
```

Copy the above file to "**/etc/udev/rules.d/**" folder. Then refresh the udev rules by issuing below command in terminal.

```
sudo cp 60-openocd.rules /etc/udev/rules.d/
sudo udevadm control --reload
```

1.4 **Install Eclipse Embedded CDT (C/C++)**:

1.4.1 **Download & Extract Eclipse tool**:

The eclipse tool can be downloaded from the site https://projects.eclipse.org/projects/iot.embed-cdt/downloads/ . First of all download the eclipse bundle from the above site (Linux). In my case the filename was "eclipse-embedcpp-2020-12-R-linux-gtk-x86_64.tar.gz". Extract into a folder and launch the binary "eclipse".

1.4.2 **Install Eclipse Embedded CDT (C/C++) plug-in**:

The eclipse may not come with built in Embededd CDT plug-ins. You need to isnatll those by going to **Marketplace** in eclipse. Goto **Help->Eclipse Marketplace...** Then search for "Eclipse Embedded C/C++". You will get a plug-in named "**Eclipse Embedded C/C++** 6.1.2" with description as "The Eclipse Embedded CDT (C/C++ Development Tools, formerlyGNU MCU/ARM Eclipse". This plug-in is an open source project that includes a family of Eclipse plug-ins and tools for multi-platform embedded Arm & RISC-V development, based on GNU toolchains. Select and install the plug-in.

1.4.3 **Create new project for TM4C123GH6PM Microcontroller**:

Now, you create a new project by using **File–>New Project…** Then Select **C Project** and click Next. Give Project a name “tiva”. Then select **Empty Project** under **Executable.** Select the Toolchains as “Arm Cross GCC”. Click next. Then click Next. Now, Select the path for Arm Compiler. (In my case, /home/vksalian/Desktop/Practice_2021/baremetal_tiva_TM4C123GH6PM_IDE/gcc–arm–none–eabi–10–2020–q4–major/bin). Then Click Finish. Now you have an empty project.

Let us **copy** our previous project into the current project. The files that need to be copied are,

led.c
led.h
main.c
startup_tm4c123gh6pm.c
tm4c123gh6pm.h
tm4c123gh6pm.ld

1.4.4 **Configure the compiler options for the TM4C123GH6PM project**:

In this section we have to configure the project so that the compiler takes the correct arguements during build.

Right click on the **project**, then select **Properties**. Goto **C/C++ Build–>Settings**. Then Select the **Tool Settings**–>**Target Processor**. Select the options as below,

Arm family (–mcpu)	Cortex–m4
Architecture (–march)	ToolChain default
Instruction Set	Thumb (–mthumb)
Endianness	Little endian (–mlittle–endian)
Float ABI	FP instructions (hard)

FPU Type	fpv4-sp-16

Then Select the **Tool Settings**->**Optimization**. Select the options as below,

Optimization Level	None (-O0)
Message Length (-fmessage-length=0)	*Keep CHECKED*
'char' is signed (-fsigned-char)	*Keep CHECKED*
Function sections (-ffunction-sections)	*Keep CHECKED*
Data sections (-fdata-sections)	*Keep CHECKED*

Then Select the **Tool Settings->Warnings**. Select the options as below,

Enable All common Warnings (-Wall)	*Keep CHECKED*

Then Select the **Tool Settings->GNU Arm Cross Linker->General**. Add the Linker command file (linker script) file by selecting from the workspace. In our case the linker script is **tm4c123gh6pm.ld**. Then **CHECK** the option "Remove unused sections (-Xlinker -gc-sections)". Also **CHECK** the option "No startup or defualt libs (-nostdlib)".

Click on **Apply and Close** button to save the above settings. Now, try building the project to see if you can build the project. You should get log messages as below,

```
15:08:06 **** Build of configuration Debug for project tiva ****
make all
Building file: ../led.c
Invoking: GNU Arm Cross C Compiler
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mlittle-endian -mfloat-abi=hard -
mfpu=fpv4-sp-d16 -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -
fdata-sections -Wall  -g3 -std=gnu11 -MMD -MP -MF"led.d" -MT"led.o" -c -o
"led.o" "../led.c"
Finished building: ../led.c

Building file: ../main.c
Invoking: GNU Arm Cross C Compiler
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mlittle-endian -mfloat-abi=hard -
mfpu=fpv4-sp-d16 -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -
fdata-sections -Wall  -g3 -std=gnu11 -MMD -MP -MF"main.d" -MT"main.o" -c -o
```

```
"main.o" "../main.c"
Finished building: ../main.c

Building file: ../startup_tm4c123gh6pm.c
Invoking: GNU Arm Cross C Compiler
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mlittle-endian -mfloat-abi=hard -
mfpu=fpv4-sp-d16 -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -
fdata-sections -Wall  -g3 -std=gnu11 -MMD -MP -MF"startup_tm4c123gh6pm.d" -
MT"startup_tm4c123gh6pm.o" -c -o "startup_tm4c123gh6pm.o"
"../startup_tm4c123gh6pm.c"
Finished building: ../startup_tm4c123gh6pm.c

Building target: tiva.elf
Invoking: GNU Arm Cross C Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mlittle-endian -mfloat-abi=hard -
mfpu=fpv4-sp-d16 -O0 -fmessage-length=0 -fsigned-char -ffunction-sections -
fdata-sections -Wall  -g3 -T
"/home/vksalian/Desktop/Practice_2021/baremetal_tiva_TM4C123GH6PM_IDE/
eclipse_workspace/tiva/tm4c123gh6pm.ld" -nostdlib -Xlinker --gc-sections -Wl,-
Map,"tiva.map" -o "tiva.elf"  ./led.o ./main.o ./startup_tm4c123gh6pm.o
Finished building target: tiva.elf

Invoking: GNU Arm Cross Create Flash Image
arm-none-eabi-objcopy -O ihex "tiva.elf"  "tiva.hex"
Finished building: tiva.hex

Invoking: GNU Arm Cross Print Size
arm-none-eabi-size --format=berkeley "tiva.elf"
   text        data         bss         dec         hex     filename
    508           0           0         508         1fc     tiva.elf
Finished building: tiva.siz


15:08:08 Build Finished. 0 errors, 0 warnings. (took 1s.484ms)
```

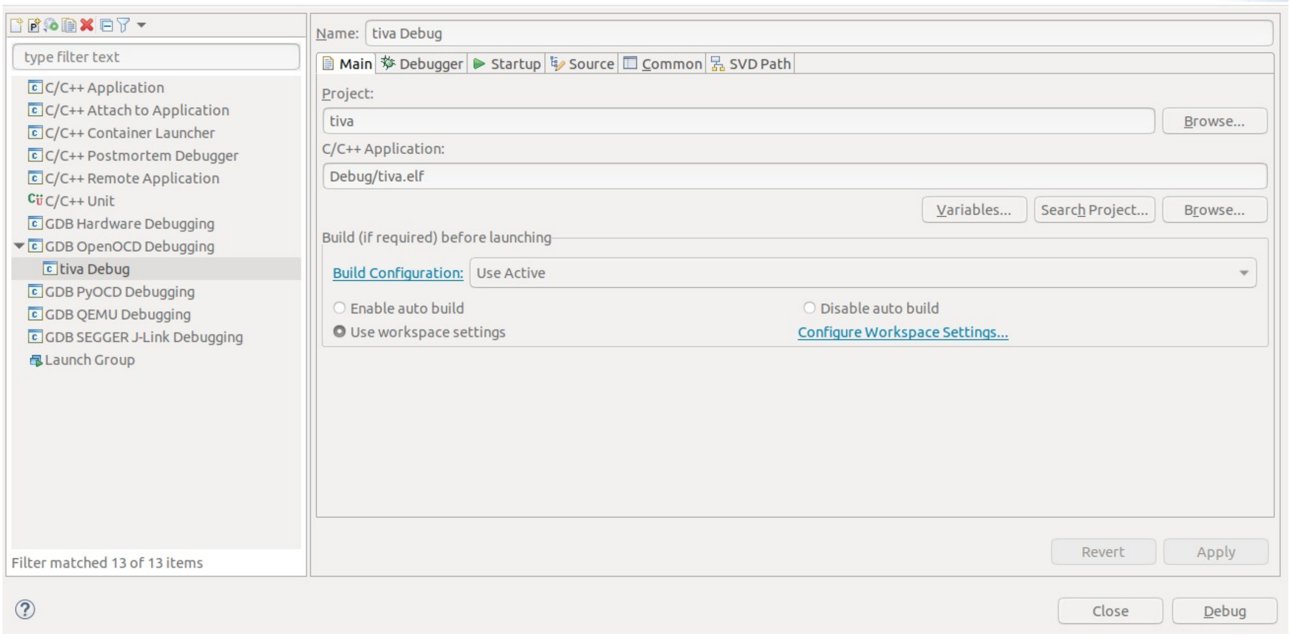
Goto **MCU–>Arm Toolchains Paths**. Select the toolchain folder path as below,

Toolchain folder	/home/vksalian/Desktop/Practice_2021/baremetal_tiva_TM4C123GH6PM_IDE/gcc-arm-none-eabi-10-2020-q4-major/bin
Toolchain name	Xpack GNU Arm Embedded GCC

1.4.5 **Configure OpenOCD Debugger for the TM4C123GH6PM project**:

Now, click on the ***DownArrow*** button next to **Debug** button and select **Debug Configurations**... Double Click on **GDB OpenOCD Debugging** to create new configuration. Rename the Configuration with name of your choice.

I have renamed it as **tiva Debug**. I have added the snapshots below.



Name: tiva Debug

Project: tiva

C/C++ Application: Debug/tiva.elf

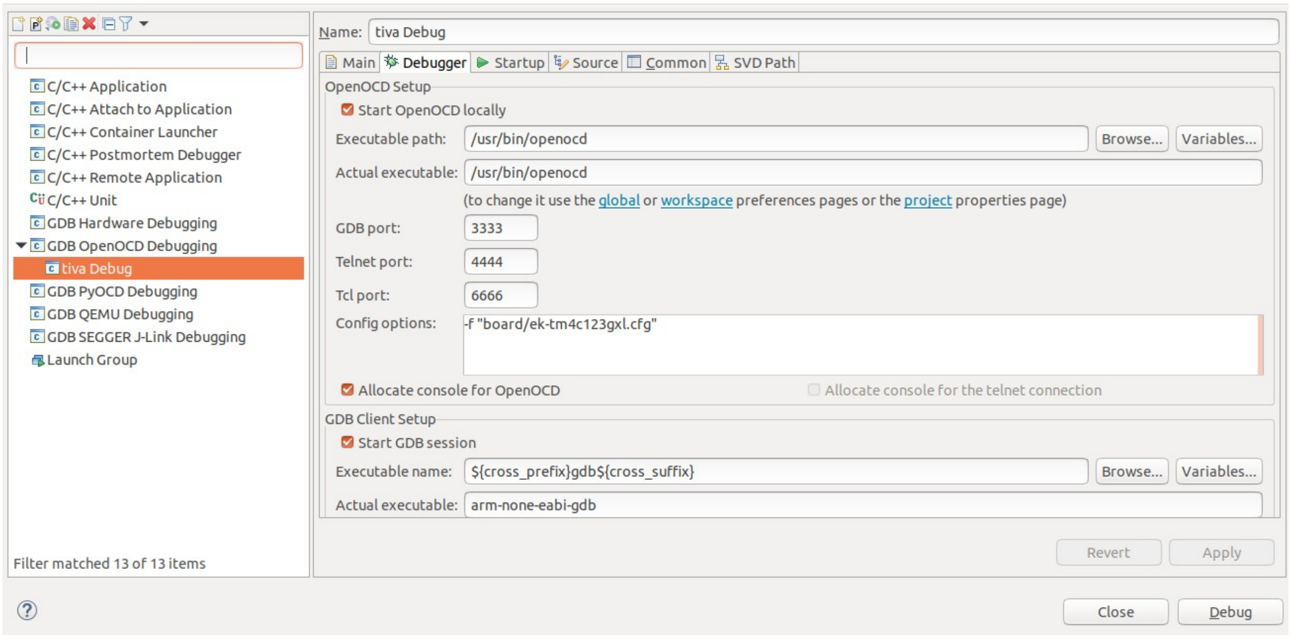
Build (if required) before launching

Build Configuration: Use Active

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

Filter matched 13 of 13 items



Name: tiva Debug

OpenOCD Setup

☑ Start OpenOCD locally

Executable path: /usr/bin/openocd

Actual executable: /usr/bin/openocd

(to change it use the global or workspace preferences pages or the project properties page)

GDB port: 3333

Telnet port: 4444

Tcl port: 6666

Config options: -f "board/ek-tm4c123gxl.cfg"

☑ Allocate console for OpenOCD ☐ Allocate console for the telnet connection

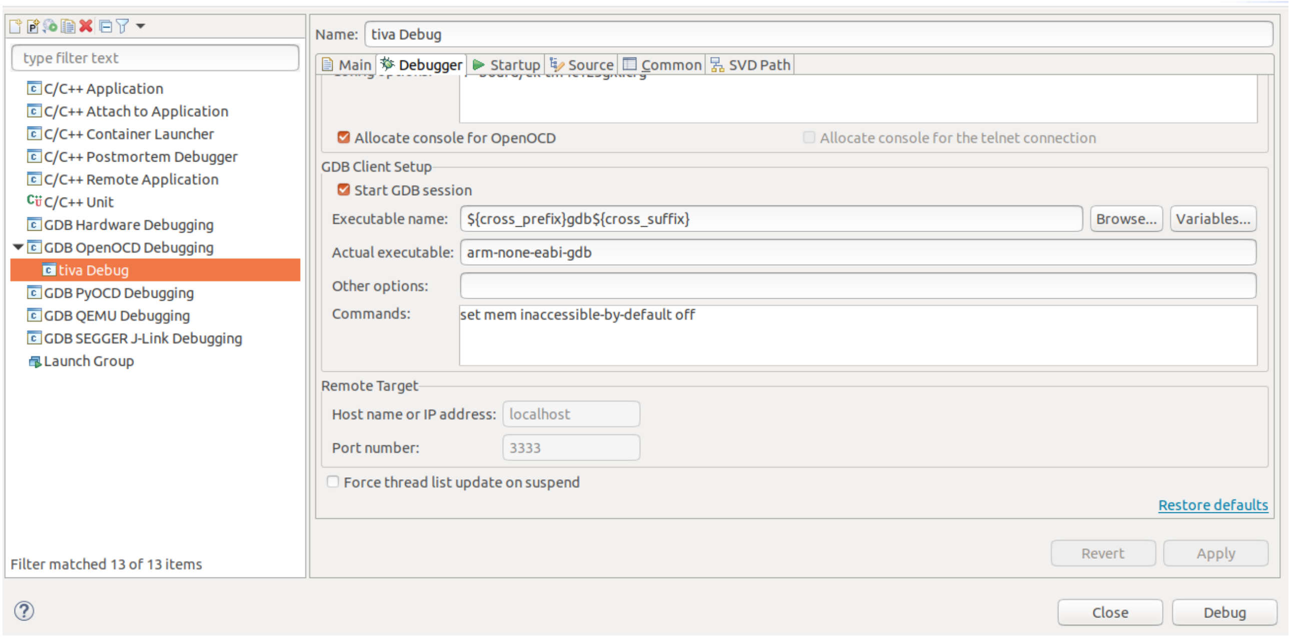
GDB Client Setup

☑ Start GDB session

Executable name: ${cross_prefix}gdb${cross_suffix}

Actual executable: arm-none-eabi-gdb

Filter matched 13 of 13 items



type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - tiva Debug
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 13 of 13 items

Name: tiva Debug

Main | Debugger | Startup | Source | Common | SVD Path

[✓] Allocate console for OpenOCD　　[ ] Allocate console for the telnet connection

GDB Client Setup

[✓] Start GDB session

Executable name: ${cross_prefix}gdb${cross_suffix}　Browse...　Variables...

Actual executable: arm-none-eabi-gdb

Other options:

Commands: set mem inaccessible-by-default off

Remote Target

Host name or IP address: localhost

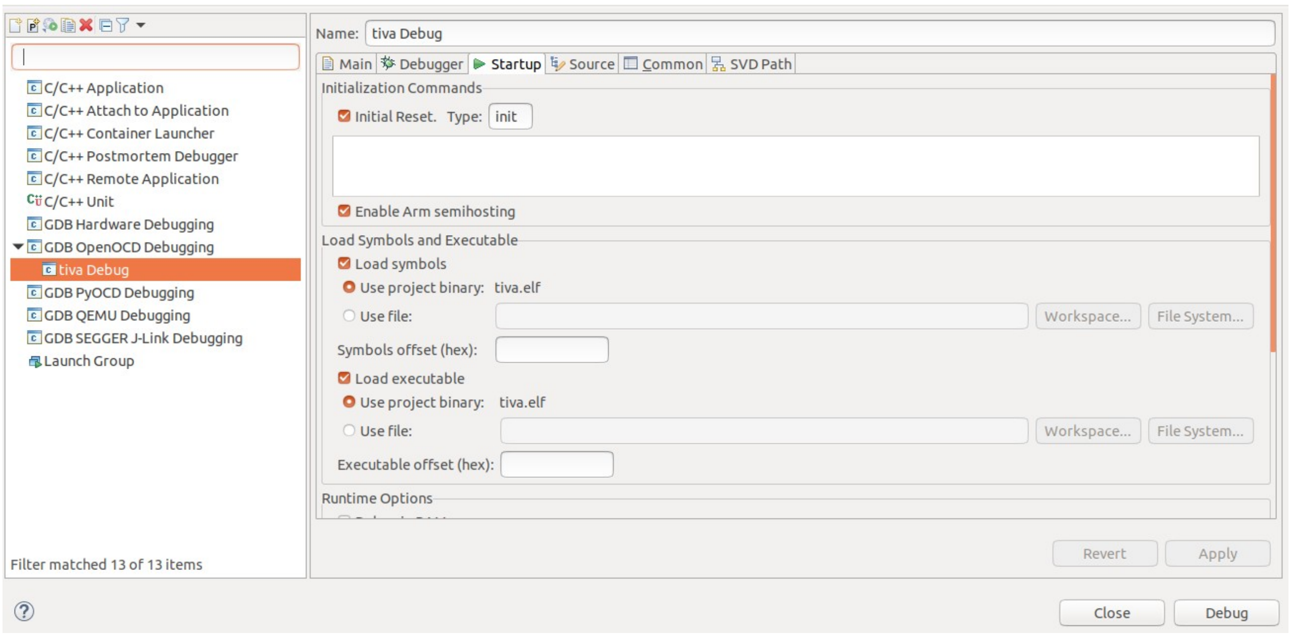
Port number: 3333

[ ] Force thread list update on suspend

Restore defaults

Revert　Apply

Close　Debug



- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - tiva Debug
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 13 of 13 items

Name: tiva Debug

Main | Debugger | Startup | Source | Common | SVD Path

Initialization Commands

[✓] Initial Reset.　Type: init

[✓] Enable Arm semihosting

Load Symbols and Executable

[✓] Load symbols

(●) Use project binary: tiva.elf

( ) Use file:　Workspace...　File System...

Symbols offset (hex):

[✓] Load executable

(●) Use project binary: tiva.elf

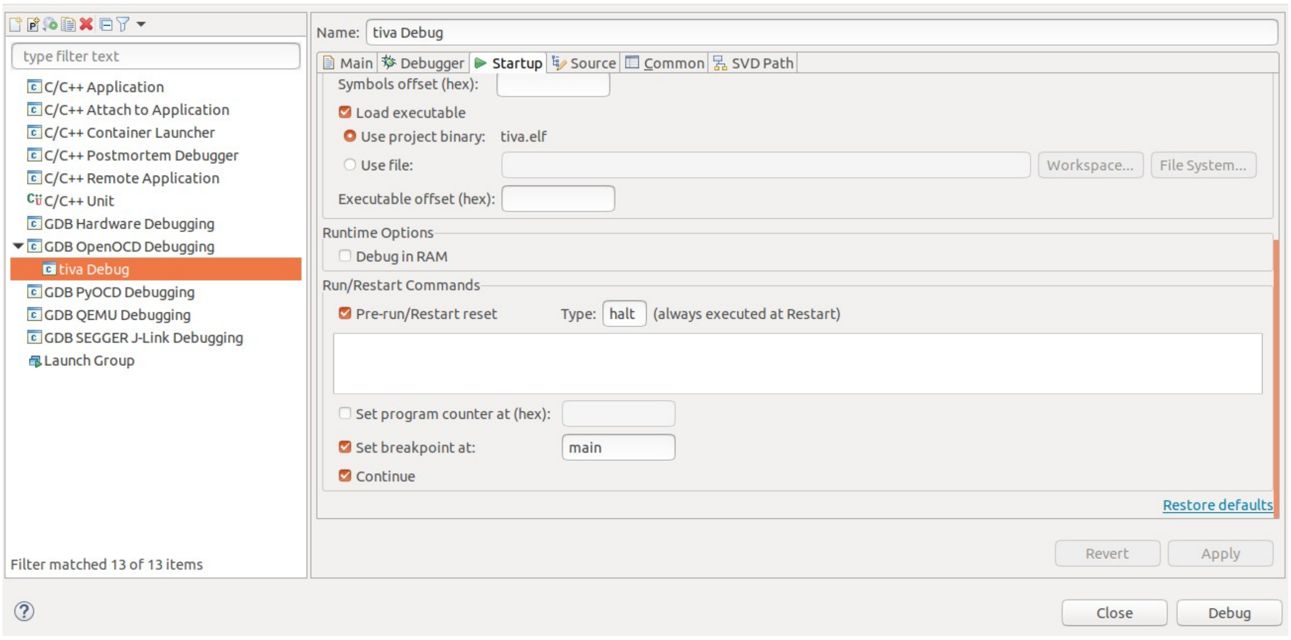
( ) Use file:　Workspace...　File System...

Executable offset (hex):

Runtime Options

Revert　Apply

Close　Debug



type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - tiva Debug
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 13 of 13 items

Name: tiva Debug

Main | Debugger | Startup | Source | Common | SVD Path

Symbols offset (hex):

☑ Load executable

● Use project binary: tiva.elf

○ Use file: [Workspace...] [File System...]

Executable offset (hex):

Runtime Options

☐ Debug in RAM

Run/Restart Commands

☑ Pre-run/Restart reset Type: halt (always executed at Restart)

☐ Set program counter at (hex):

☑ Set breakpoint at: main

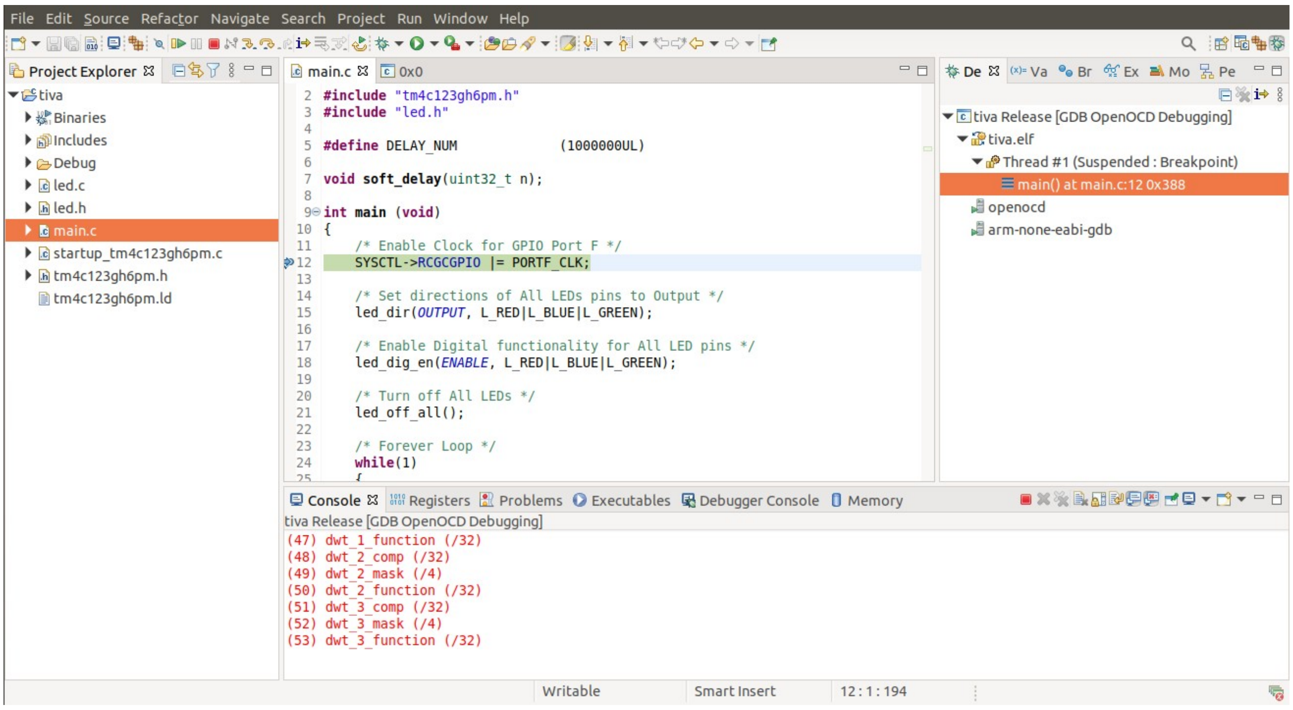
☑ Continue

Restore defaults

Revert Apply

Close Debug

NOTE: Sometimes, TIVA123G Launchpad behaves abnormally. That means the above settings did not work on the board for the first time. So, I tried enable/disable some options like "force thread list update on suspend", etc. Then it worked. But I really don't know how that happened before.



```c
#include "tm4c123gh6pm.h"
#include "led.h"

#define DELAY_NUM           (1000000UL)

void soft_delay(uint32_t n);

int main (void)
{
    /* Enable Clock for GPIO Port F */
    SYSCTL->RCGCGPIO  |= PORTF_CLK;

    /* Set directions of All LEDs pins to Output */
    led_dir(OUTPUT, L_RED|L_BLUE|L_GREEN);

    /* Enable Digital functionality for All LED pins */
    led_dig_en(ENABLE, L_RED|L_BLUE|L_GREEN);

    /* Turn off All LEDs */
    led_off_all();

    /* Forever Loop */
    while(1)
    {
```

tiva Release [GDB OpenOCD Debugging]
- tiva.elf
  - Thread #1 (Suspended : Breakpoint)
    - main() at main.c:12 0x388
- openocd
- arm-none-eabi-gdb

tiva Release [GDB OpenOCD Debugging]

```
(47) dwt_1_function (/32)
(48) dwt_2_comp (/32)
(49) dwt_2_mask (/4)
(50) dwt_2_function (/32)
(51) dwt_3_comp (/32)
(52) dwt_3_mask (/4)
(53) dwt_3_function (/32)
```

Writable Smart Insert 12 : 1 : 194

References:

1. https://igor-misic.blogspot.com/2017/10/debugging-ek-tm4c123gxl-with-visual.html

2. https://github.com/hans-jorg/tiva-tm4c123gxl-launchpad-cmsis/tree/master/01-Blink