Zebra FX Series Embedded C/CPP SDK User Guide Linux

Version 1.0.1

Table of Content

INTRODUCTION

The 'Zebra FXSeries Embedded C/CPP SDK User Guide Linux' describes the detailed steps about how to use the FX Series Embedded native C/CPP SDK to develop the RFID sample application from scratch, debugging it and packaging it as debian package using Eclipse IDE based on Ubuntu 16.04/Fedora 27 (64 bits, x86) host and executing the RFID sample application in target 'FX9600/FX7500' Readers.

This user guide describes the following:

- **Zebra Native C/C++ SDK** in general describes how to create, build, and debug an embedded C/C++ application.
- **Embedded Sample RFID C/C++ Application** from scratch, create, build and debug and how to create debian package, how to create Start/Stop script files for the deployment of debian package and install it through Web interface of target 'Reader FX9600/FX7500' from Ubuntu 16.04/Fedora 27 64bit x86 Host.
- Debian packaging of **embedded C/C++ Linux SDK package** for newer version.

  Note: Uninstall any older Zebra SDK if installed on Linux host machine.

REVISION HISTORY

REV	DESCRIPTION	DATE	AUTHOR
1.0	Steps and procedure to develop, debug and package embedded application for Zebra C/C++ sample application	04-Jan-2019	
1.1	• Updated steps with added gdb support • Modified steps for strict host key checking SCP for gdb file transfer	09-Apr-2019	
1.2	Added C++ steps Added scp key for Ubuntu server Added Debian package creation	03-May-2019	
1.3	Added Debian package installation process	24-May-2019	
1.4	● dos2unix conversion executed on script files in debian installables ● Updated the SDK file names in new format ● Corrected hyphens to underscore in section 9 for basicdebtest-2.0-1 ● Java 8 installation added in prerequisites	23-June-2019	
1.5	● Modified the document in removable of repeated build/debug steps in section 7. ● Section 6 updated for C/C++ application compiler's include path, library path, libraries, compiler/linker flags. ● Modified c/c++ application projects of SDK in single workspace instead of individual workspace and user guide contents are updated for same. ● Procedure to execute RFID sample application at RFID reader.	30-April-2020	

PRE-REQUISITES

- Host Machine running with Ubuntu 16.04/Fedora 27 (64-bits x86)
- Host Machine with minimum of 8GB RAM (16GB recommended) preferred with 40GB free space, Intel Core i7 CPU
- Installation files:

  Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux_V1.0.1.tar.gz

  (tarred**/**compressed file**)** or

  ZebraFXSeriesEmbeddedSDKCCPPLinux_1.0.1.deb (debian package)

  provided by Zebra.
- Target Machine 'RFID Reader FX Series FX9600/FX7500' with firmware Version 3.x.x or higher
- Ensure that both target/host service port (22) is up
- Ensure latest Java (java 8) is installed in the host
- RFID API documentation (RFID3_SDK_C_Help.chm) in the doc folder can be viewed using kchmviewer. This can be installed as below.

  Ubuntu 16.04:

  sudo apt-get update

  sudo apt-get install kchmviewer

  Fedora:

  yum install kchmviewer-qt

4.0 SDK Install

This section describes the steps involved for installing Zebra Native C/C++ SDK on Linux host Ubuntu 16.04 machine.

4.1 Install C/C++ SDK debian package (Ubuntu 16.04)

Note: Debian package installation is supported only with Ubuntu-Linux
Host machine used is Ubuntu-16.04

Download the deb file "ZebraFXSeriesEmbeddedSDKCCPPLinux_1.0.1.deb" from Zebra ftp site.

Install Command

- "sudo dpkg -i ZebraFXSeriesEmbeddedSDKCCPPLinux_1.0.1.deb"

Once installation is done, SDK will be installed in
/usr/share/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux directory
In Debian Zebra installed packages above path will become the default installation path

Using GUI browse to the new path
/usr/share/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux

Jump to Figure 4 in the section below and follow the same steps thereafter considering
/usr/share/Zebra-FXSeries-Embedded-Native-C-SDK-Linux
as the default install-path for Debian Zebra package installations only.

Note:
For ***removal*** *of Debian zebra C/C++ package is required, then come out of the installation path (***/usr/share/Zebra-FXSeries***...../) into some other directory and then use command*

- *"sudo dpkg -r zebrafxseriesembeddedsdkccpplinux"*

This will remove the installed Debian Zebra C/C++ package from the default path.

Issue while installing deb package
Sometimes users may face the lock problem while trying to install using Debian package.

dpkg: error: dpkg status database is locked by another process

Solution to unlock and install
First run:

lsof /var/lib/dpkg/lock

Then make sure that process isn't running:

ps cax | grep PID

If it is running:

kill PID #wait kill -9 PID

Make sure process is done:

ps cax | grep PID

Then remove the lock file:

sudo rm /var/lib/dpkg/lock

Let dpkg fix itself:

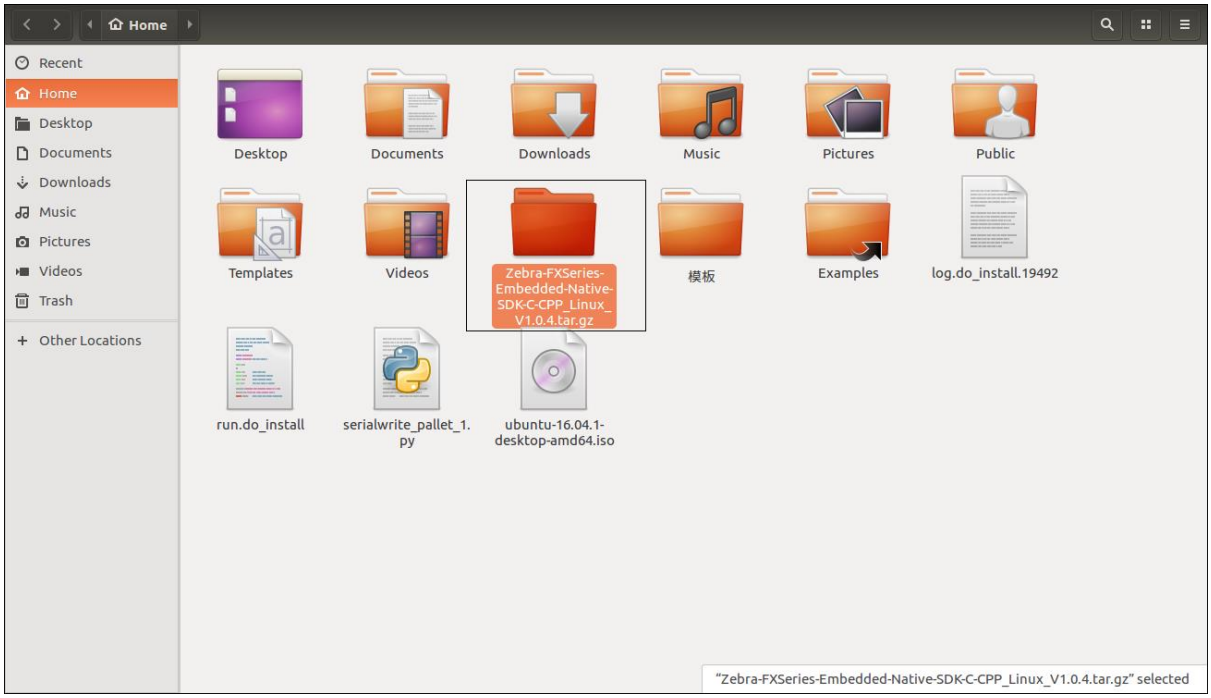
sudo dpkg --configure -a

After this dpkg installation should work fine

4.2 Install Native C/C++ SDK tar file (Fedora/Ubuntu)

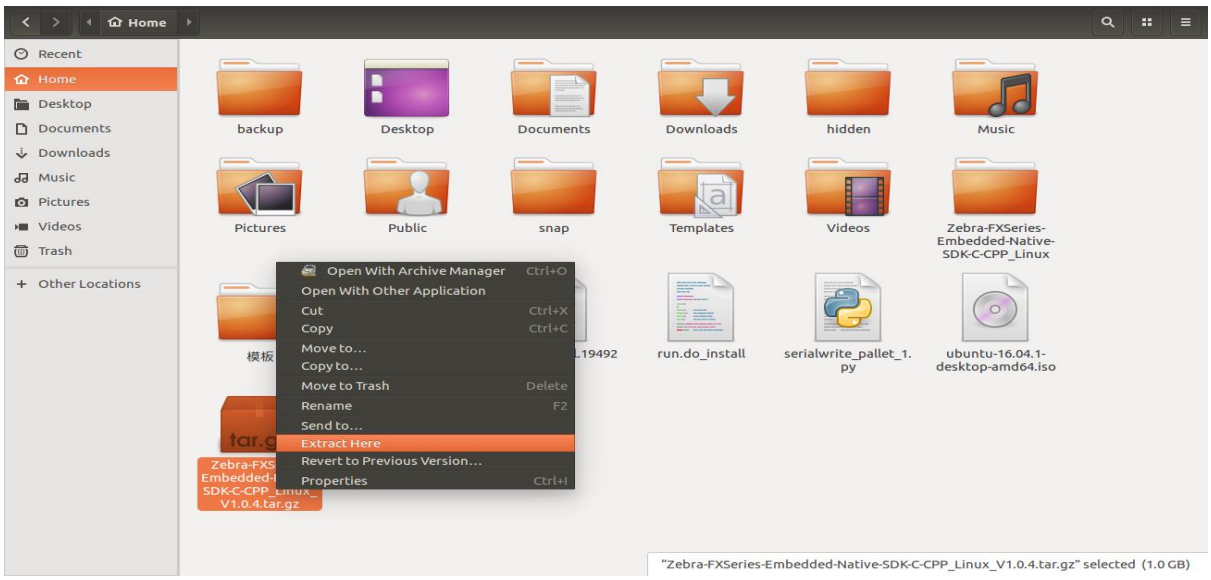
Tar file can be installed in Ubuntu/Fedora as follows.
Copy the Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux_V1.0.1.tar.gz file provided by Zebra in any of the host machine directory; this will be the install base directory (i.e. <installation-path>).
Extract the tar file as shown below:

Figure 1: Tar File



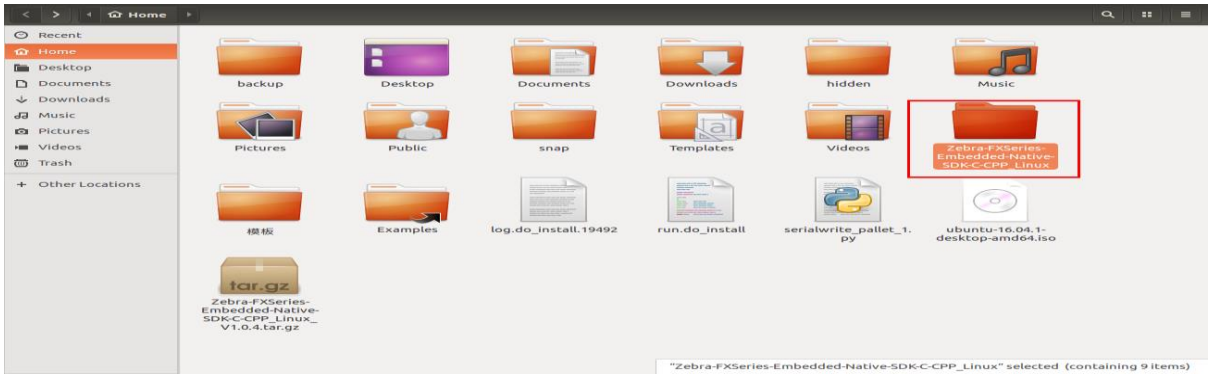
Right-click on the tar file, and select Extract Here as shown in Figure 2.

Figure 2: Untar SDK



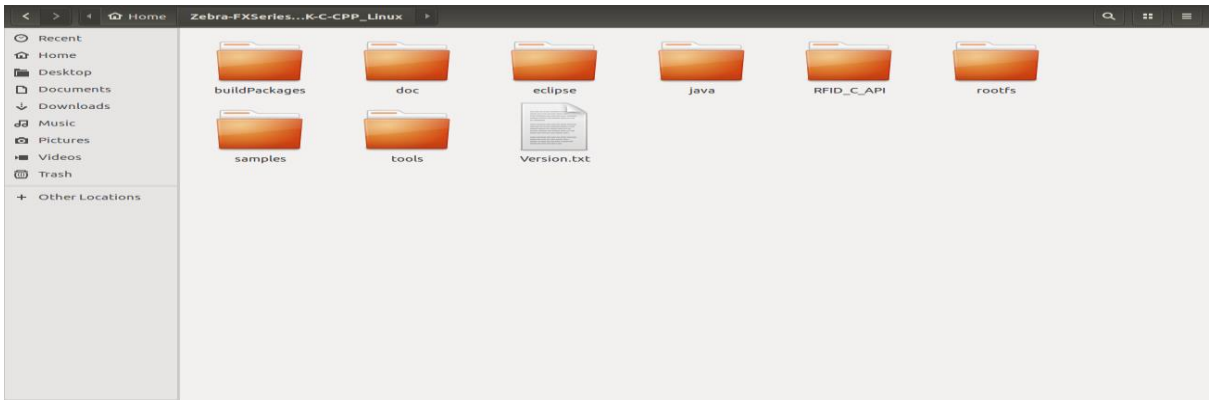
After successful extraction, the folder will be visible as shown below. Double-click the folder to see that the respective folders are present.

Figure 3: Untarred Directory



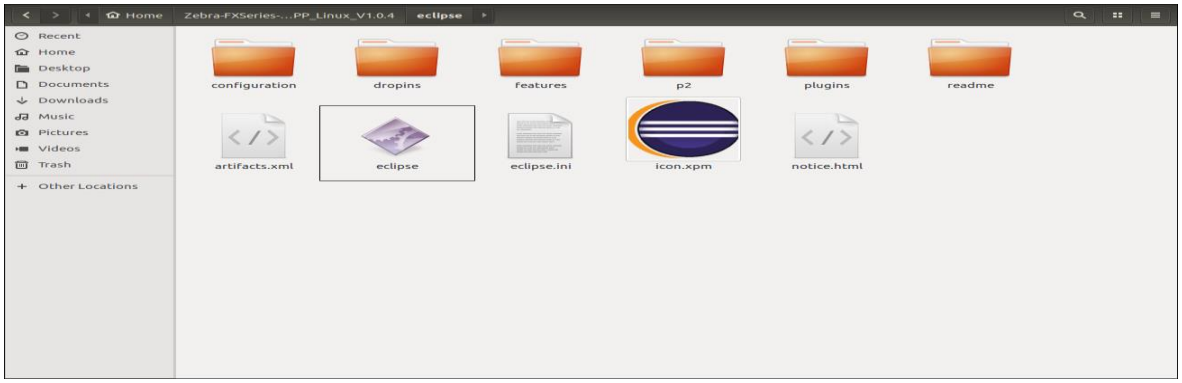
Double-click on the untarred folder ‘Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux’
Verify that required directories are available as per following Figure 4.

Figure 4: SDK Base Directory



Double-click the eclipse directory to verify that required files are available as per Figure 5.

Figure 5: Eclipse Directory



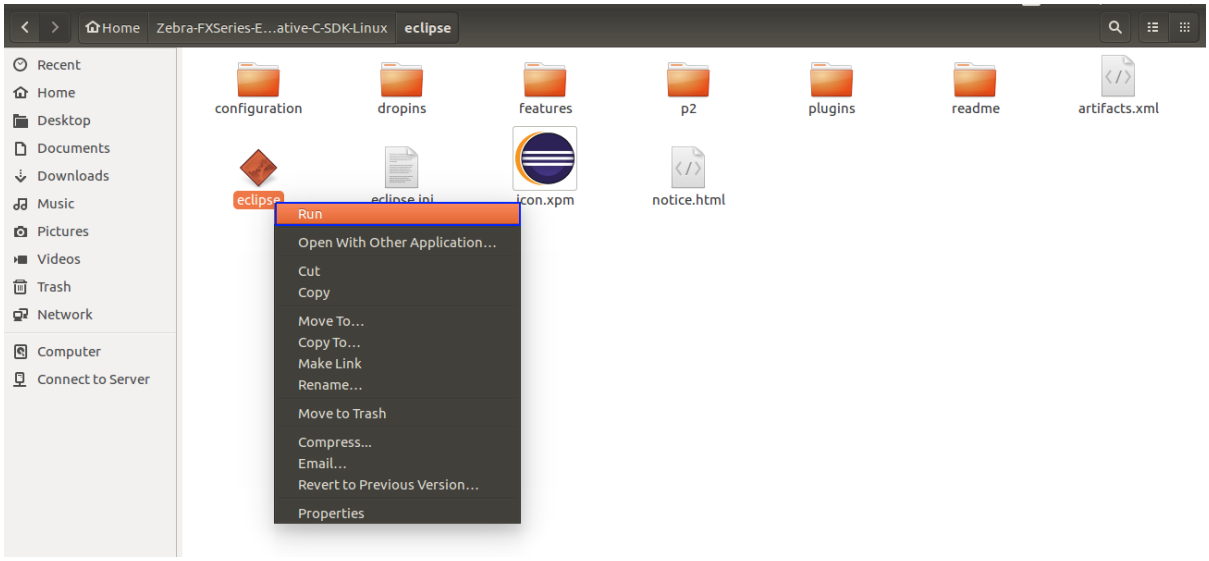
Note:

Once installation is done, in case of non-Debian (using tar file) default installation, SDK will be installed in, **<installation-path>/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux** folder.

5.0 Starting C/C++ SDK

To start the SDK double-click or right-click and select ‘Run’ on the eclipse executable file inside eclipse directory as shown in Figure 6

Figure 6: Right-click on the Eclipse Executable File

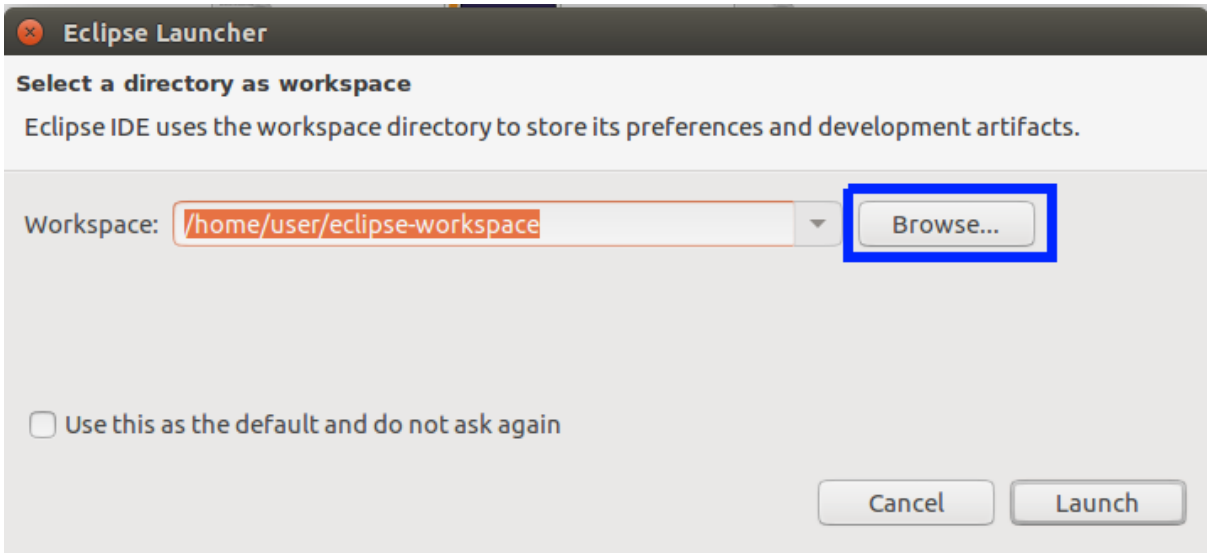


The following eclipse will pop-up as per Figure 7 and later workspace window will be showcased as per Figure 8.

Figure 7: Eclipse Screen

As shown in Figure 8, click on the Browse button to select the workspace directory.

Figure 8: Workspace Popup

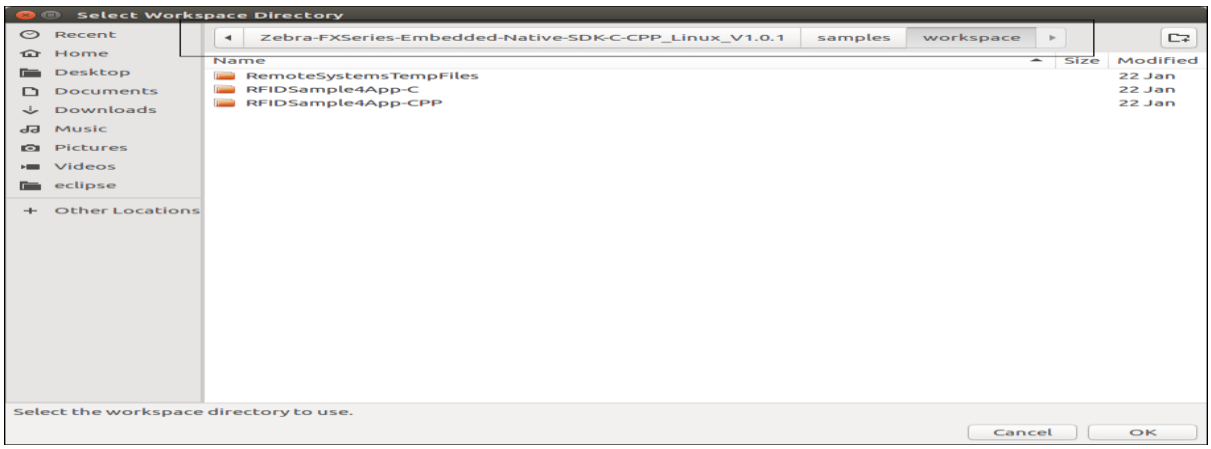


Select the workspace directory path as shown in Figure 9 below.

Select till C or C++ in case of C or C++ application respectively.

About default Eclipse Projects: Both debian package (i.e. ZebraFXSeriesEmbeddedSDKCCPPLinux_1.0.4.deb) and tarred/compressed file (i.e. Zebra-FXSeries-Embedded-SDK-C-CPP_Linux_V1.0.4.tar.gz**)** contains C application project as 'RFIDSample4App-C' and C++ application project as 'RFIDSample4App-CPP' under Eclipse workspace at folder '<installation-path>/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/samples/workspace'.

Figure 9: Select Workspace Directory

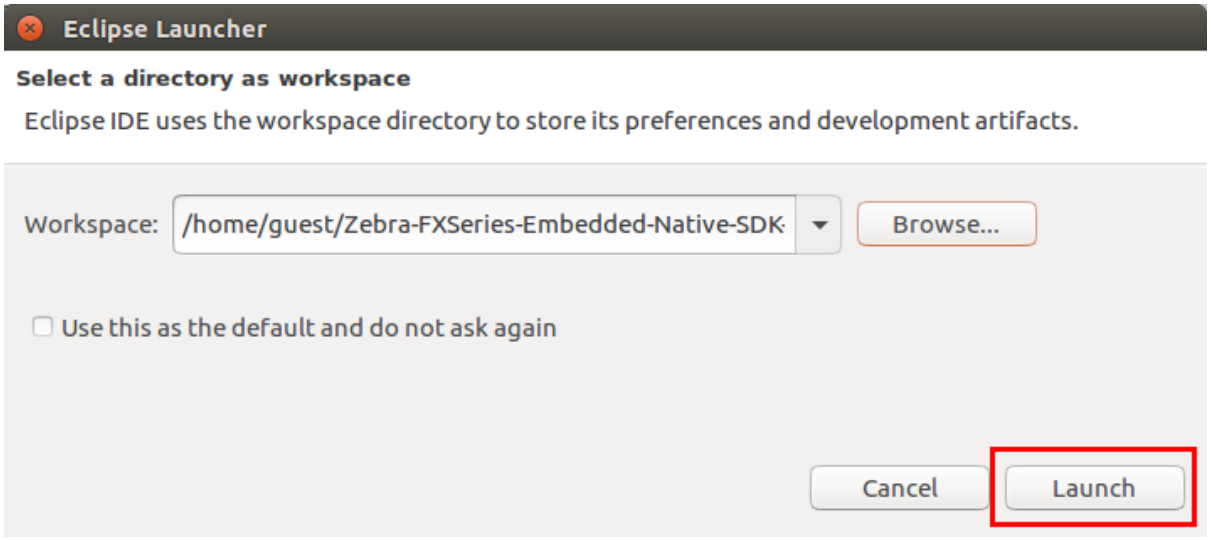


Select the workspace of

'[install-path]'/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/workspace for C/C++ applications and click OK.

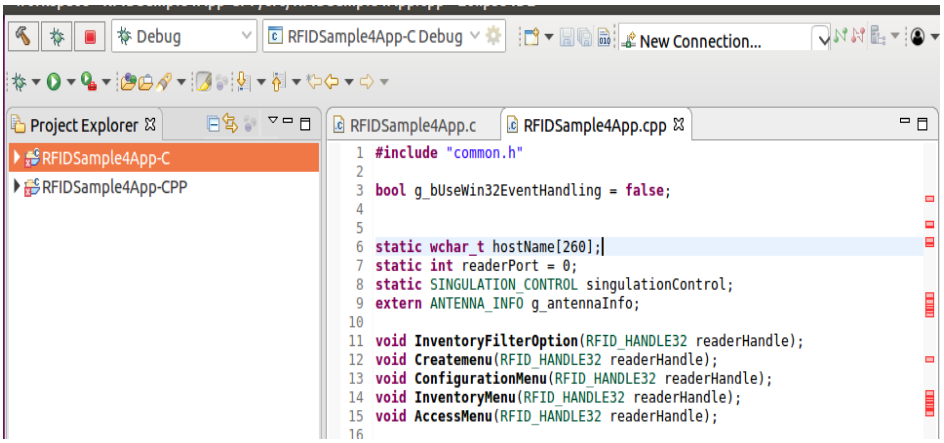
After selection of workspace directory, click on Launch button as shown in figure 10.

Figure 10: Eclipse Launch



Once Eclipse opens up, please click on Project Explorer Tab as per Figure 11. The Figure 11, shows both C application project 'RFIDSample4App-C' and C++ application project 'RFIDSample4App-CPP' . The following steps like build, debug, packaging, deployment are based on C application project ' RFIDSample4App-C' is as well as applicable to C++ application project ' RFIDSample4App-CPP'.

Figure 11: RFIDSample4Appp-C/ RFIDSample4Appp-CPP Projects



6.0 C/C++ Sample Application Build and Debug

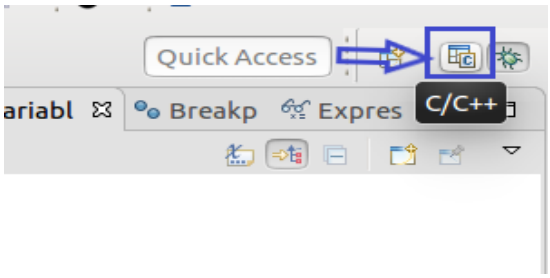
This section describes build and debugging steps for C application project 'RFIDSample4App-C'.

Note: If default perspective is not C/C++ in eclipse, then we can enable by 2 methods

Method 1:

Click on the following icon in eclipse.

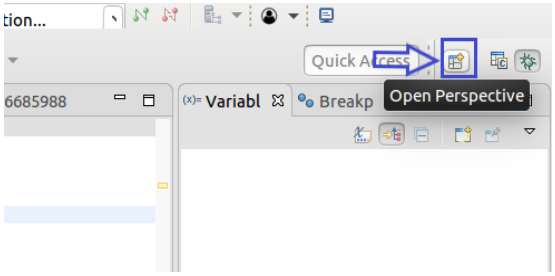
Figure 12: C/C++ Perspective Icon



Method 2:

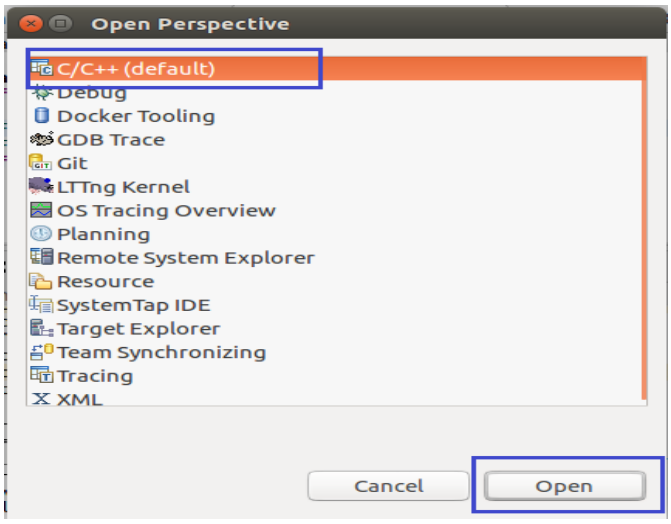
Click on open perspective.

Figure 13: Open Perspective Icon



Select C/C++ (default) and click Open.

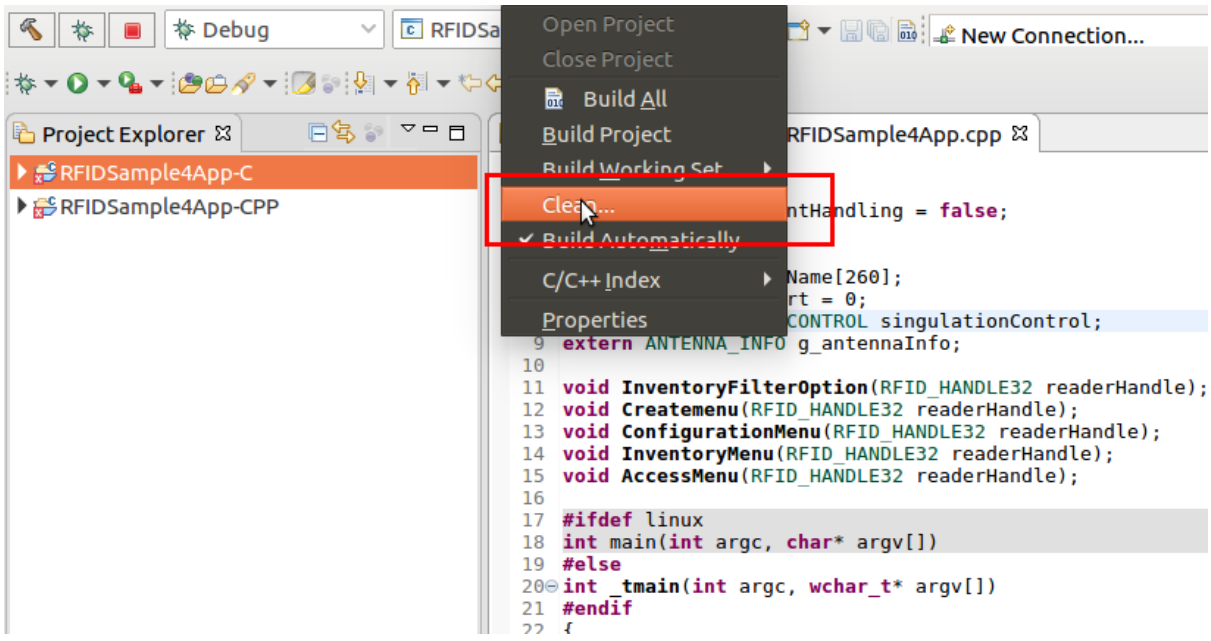
Figure 14: C/C++ Selection View



6.1 Building C/C++ Executable Binary File

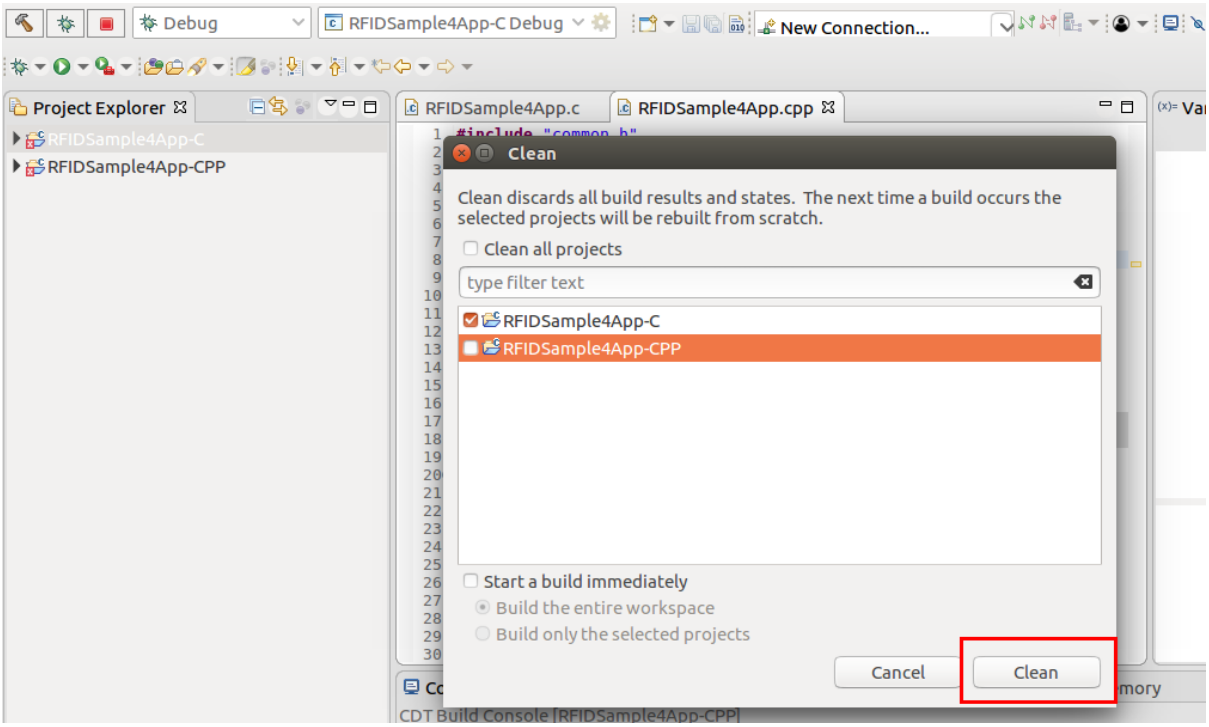
This section explains the steps on how to clean and build c application binary executable 'RFIDSample4App-C'. Select the project name 'RFIDSample4App-C' and go to menu 'Project' item and select 'Clean'.

Figure 15: Project Clean View



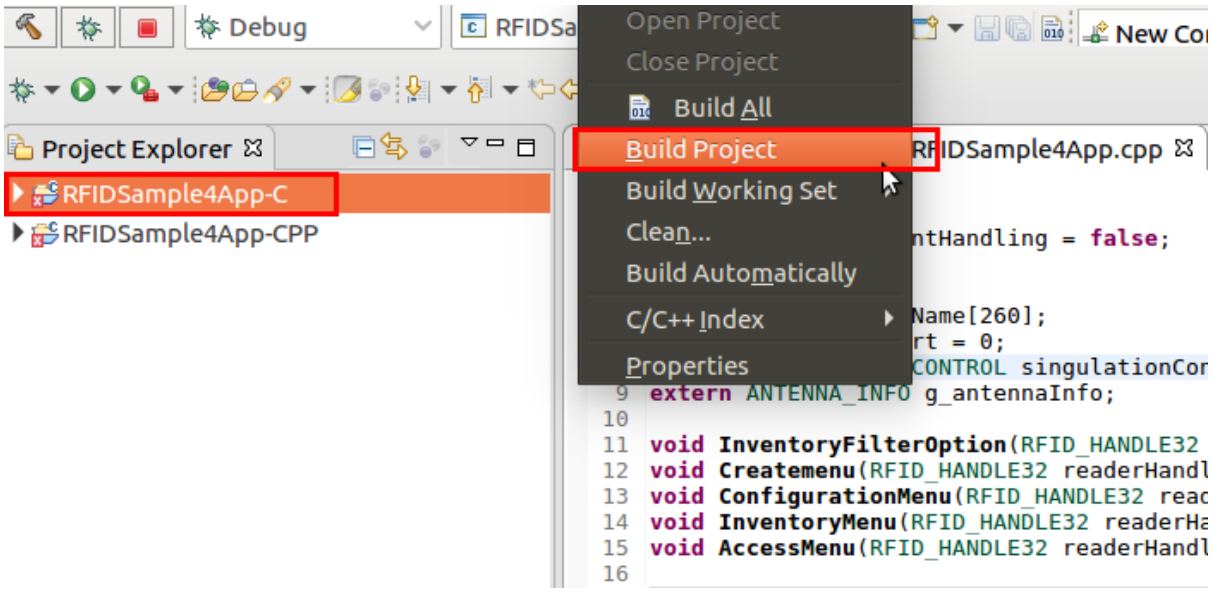
Note: Disable “Start a build immediately” checkbox.
Click on Clean button in clean popup which appears as per figure 16.

Figure 16: Project Clean Popup View



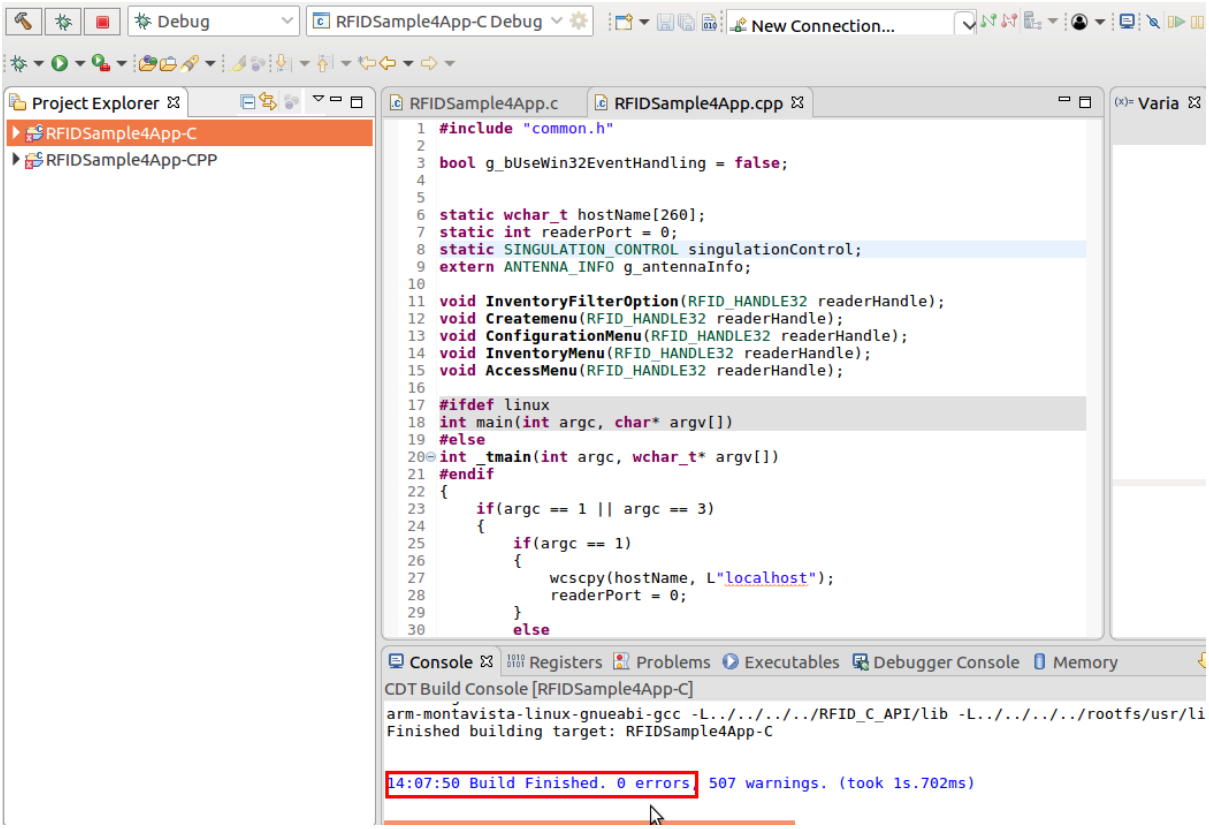
Select Project (RFIDSample4App-C) and go to Project menu and select Build Project.

Figure 17: Build Project View



After build the results are shown in the Console tab.

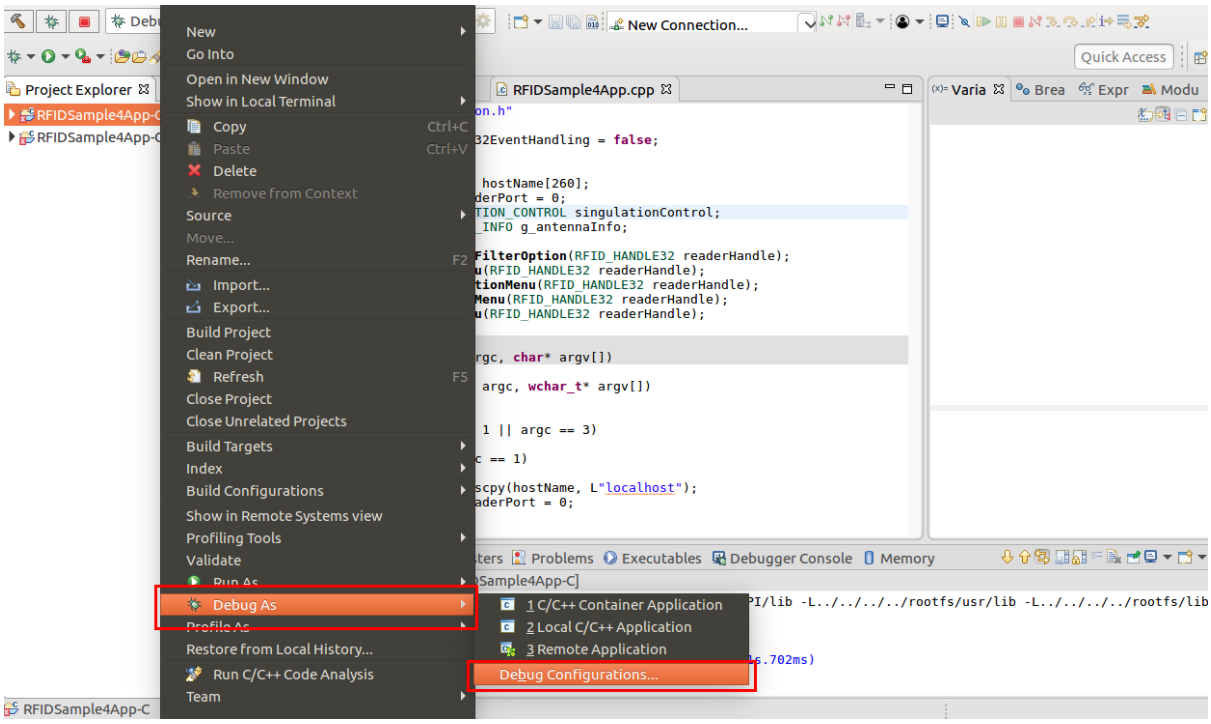
Figure 18: Project Details Window View



6.2 Debug Embedded RFID C/C++ Application

This section explains detailed steps on debugging RFID sample C application.
In the Project Explorer view, Right-click on the Project name RFIDSample4App-C and click on Debug As -> Debug Configurations as shown in the figure 19

Figure 19: Project Debug View



6.3 Setup RFID C Remote Debug Configuration.

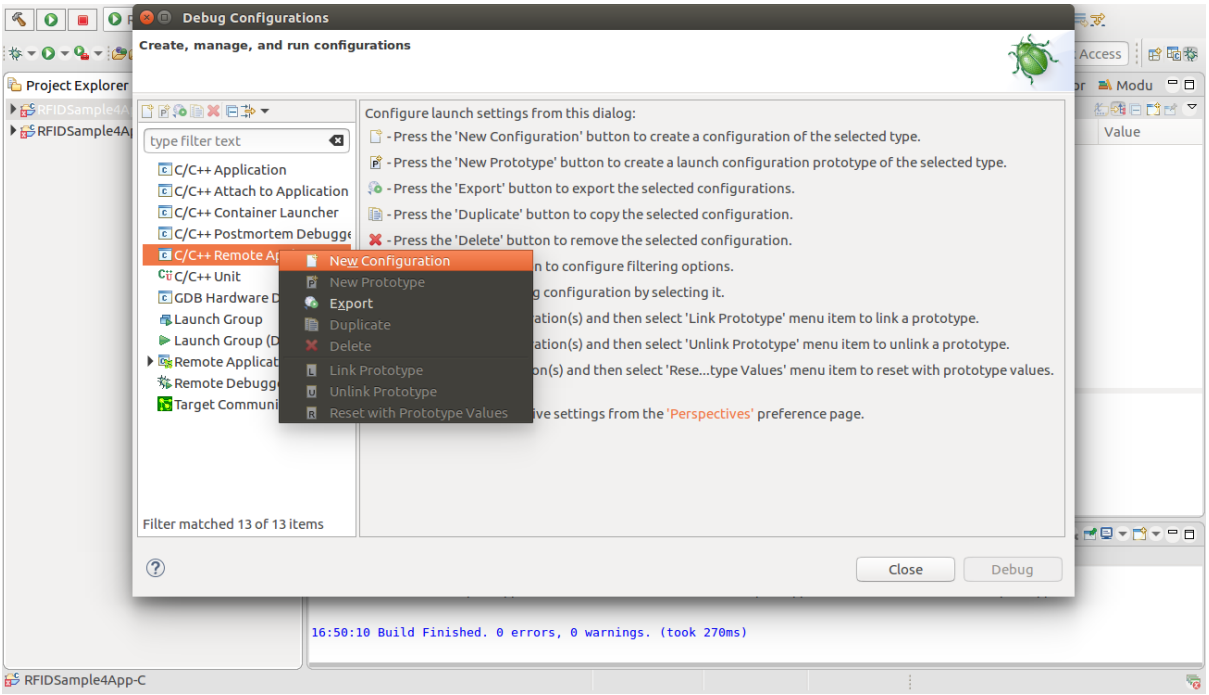
This section explains the steps on how to setup remote debug configuration on target ‘Reader FX9600/FX7500’.

RFIDSample4App-C binary file built on Linux host machine from eclipse will be transferred to target’s ‘ /apps’ directory

To connect to target and transfer the built binary, new connection should be made between target and host machine.

Follow the below steps to create new configuration as in figure 20.

Figure 20: Debug New Configuration.



Follow the below steps to select program ‘RFIDSample4App-C’ as shown in figure 21 (a). Similary also shown for ‘RFIDSample4App-CPP’ program selection for CPP application at figure 21(b).

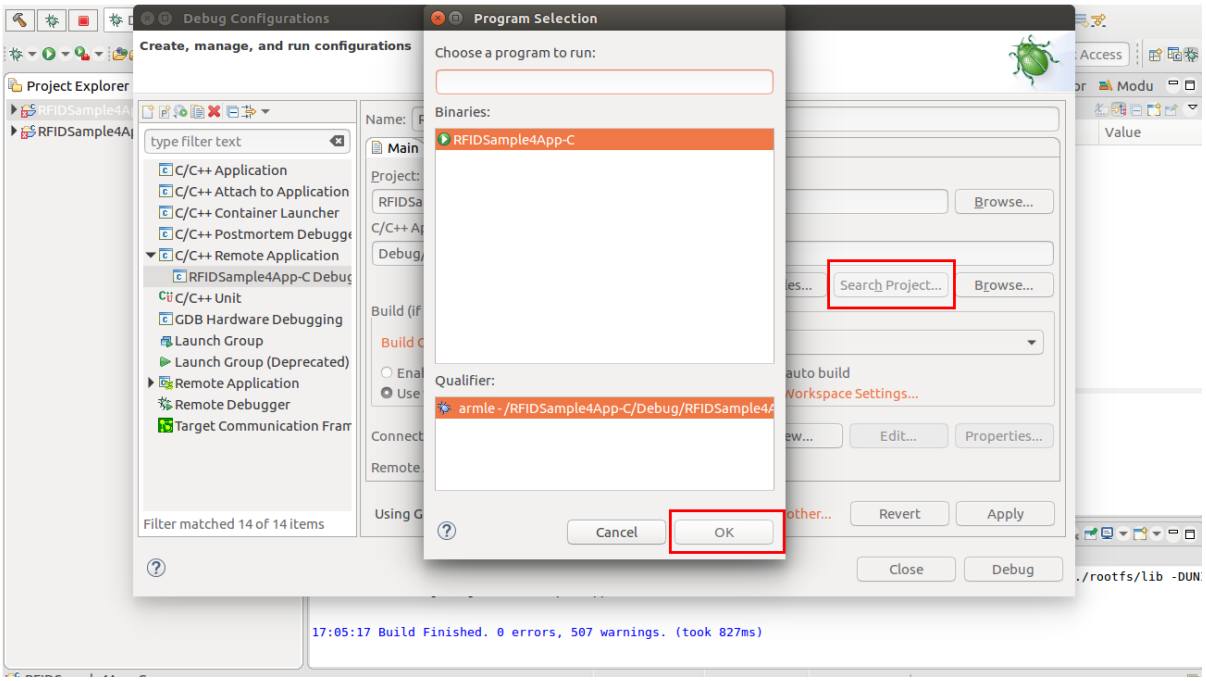


Figure 21(a): Select C application Program View

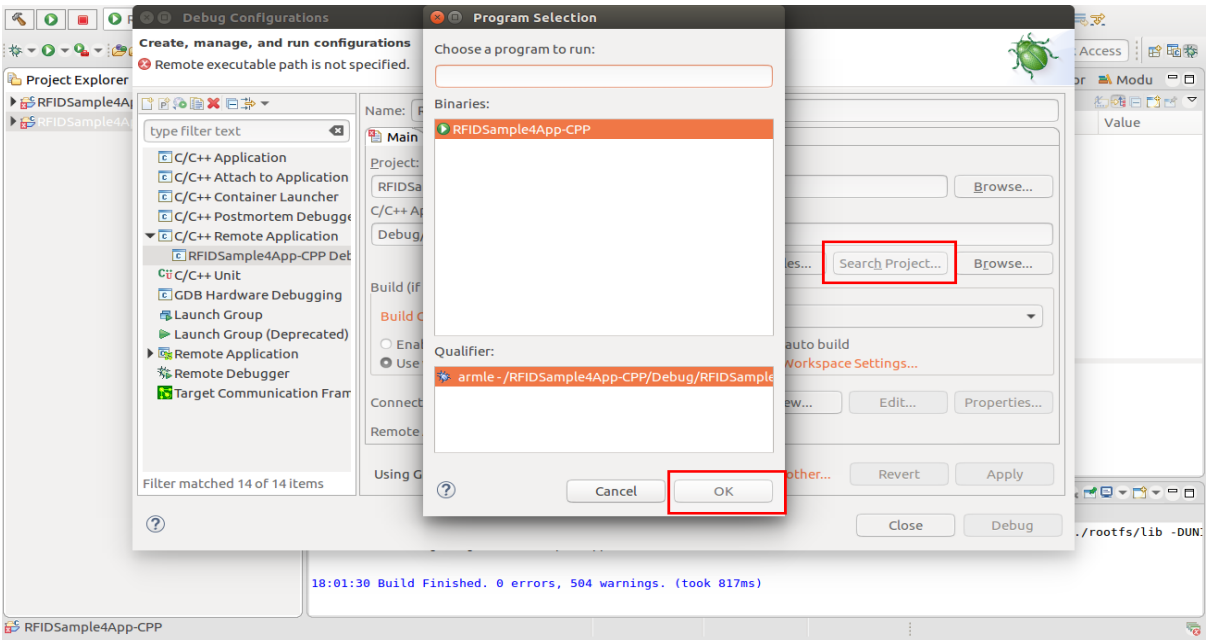
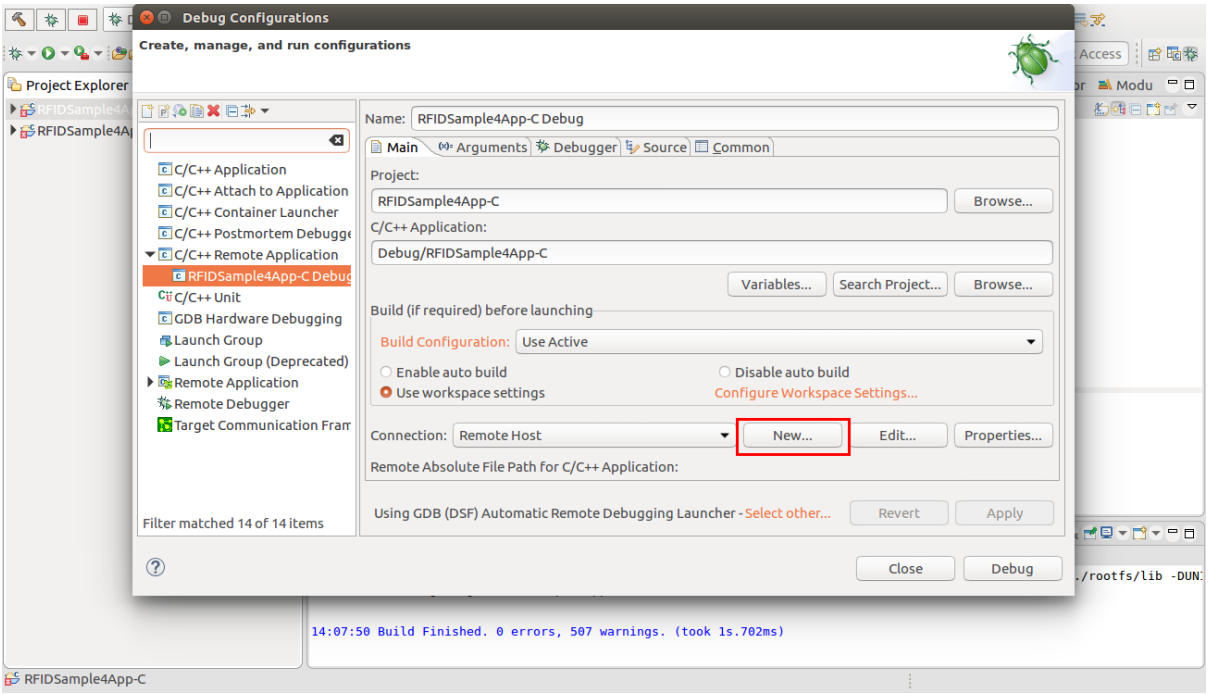


figure 21(b): Select CPP application program View

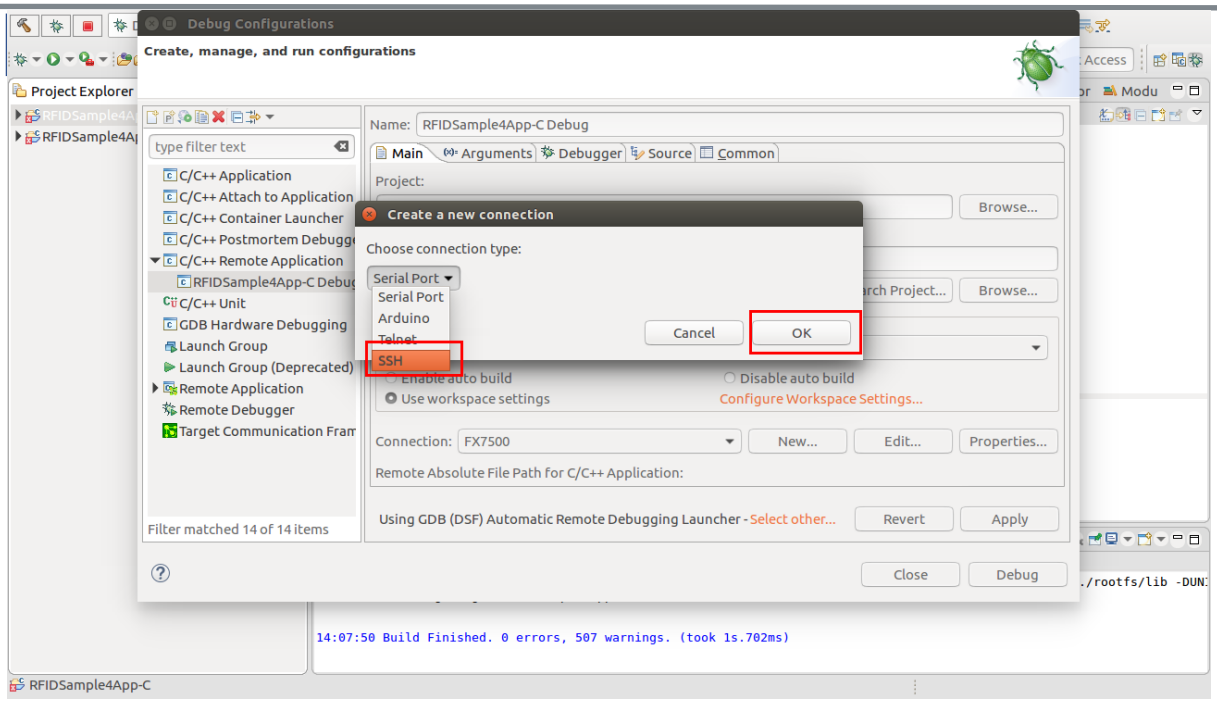
Select “New” as shown in figure 22

Figure 22: Debug Configuration New Connection View



In Create a new connection pop-up, choose connection type, select “SSH” from drop down and click “OK”

Figure 23: SSH Connection View



Enter Details of the RFID Reader FX9600 /FX7500 target

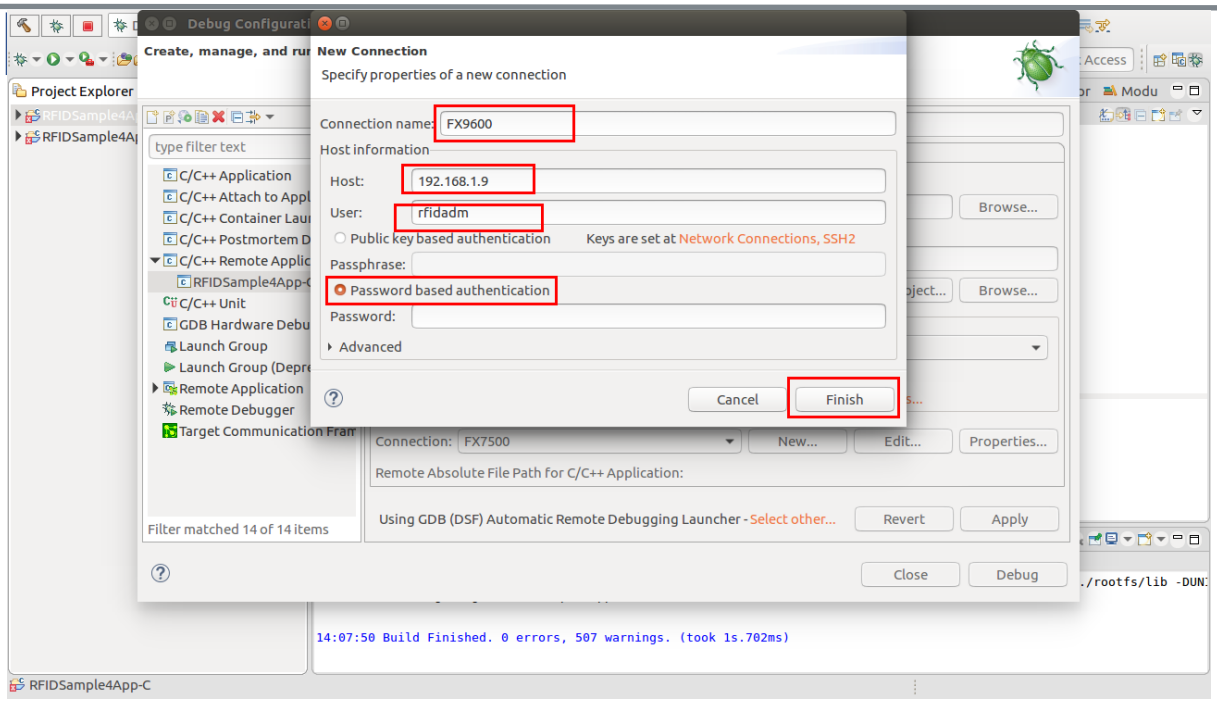
Enter Connection Name “FX9600”

Enter RFID Reader IP address “xxx.xxx.xxx.xxx” in the Host field

Enter User name “rfidadm”

And select “Password based authentication” radio button and leave Password field empty as shown in figure 24 and click “Finish” button

Figure 24: New Connection Window View



Select Connection name, say FX9600 and provide the path of the RFID Reader target where the binary executable needs to be saved on the Remote Absolute Path (“/apps/RFIDSample4App-C”) as shown in Figure 25(a) for C application project (RFIDSample4App-C). Similarly applicable for C++ application project (RFIDSample4App-CPP) where the binary executable needs to be saved on the Remote Absolute Path (“/apps/RFIDSample4App-CPP”) as shown n Figure 25(b).

Figure 25(a): New Connection Apply Window View for C application project

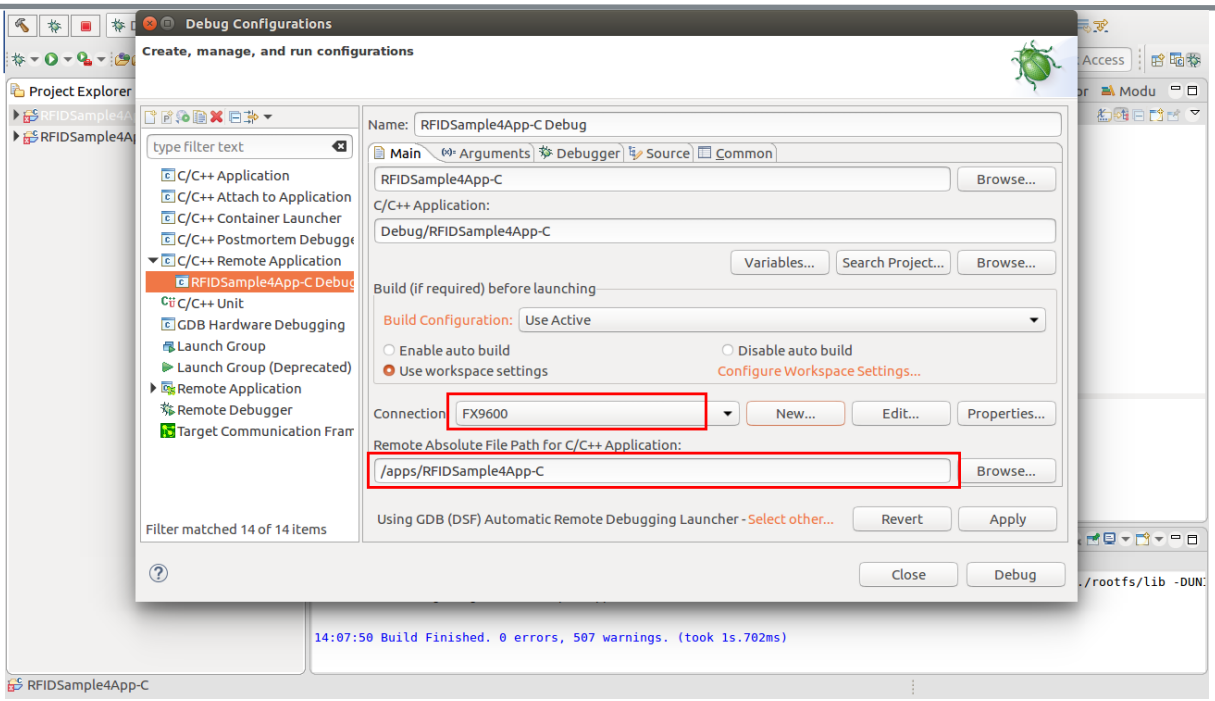
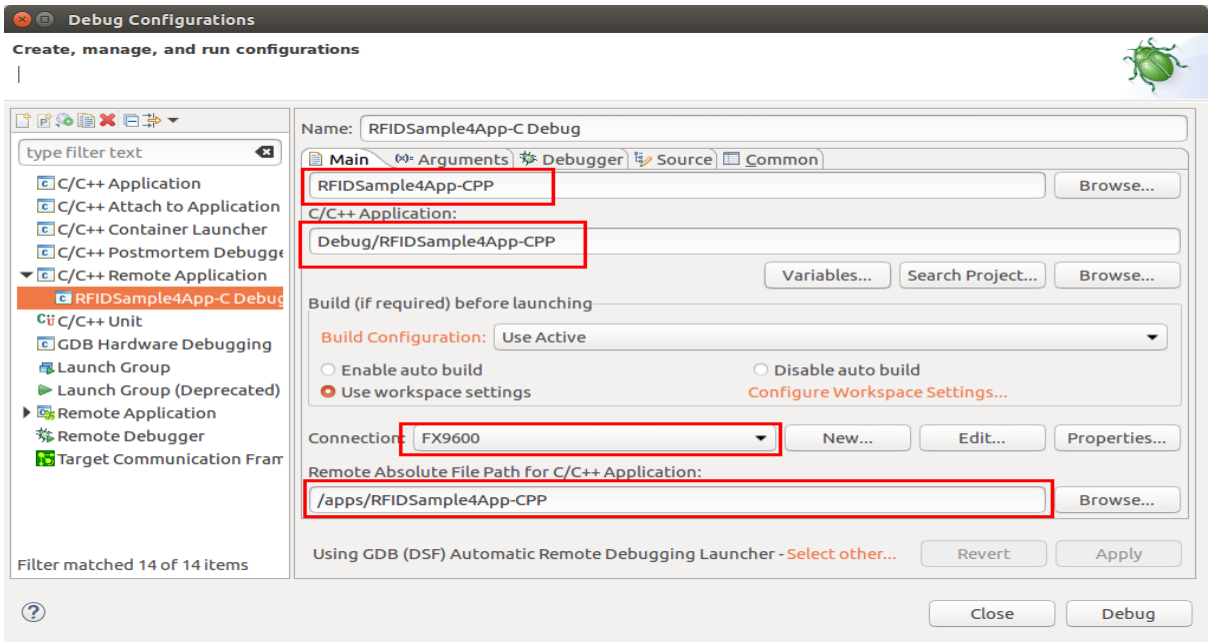


Figure 25(b): New Connection Apply Window View for C++ application project



Command to execute before application run is explained below

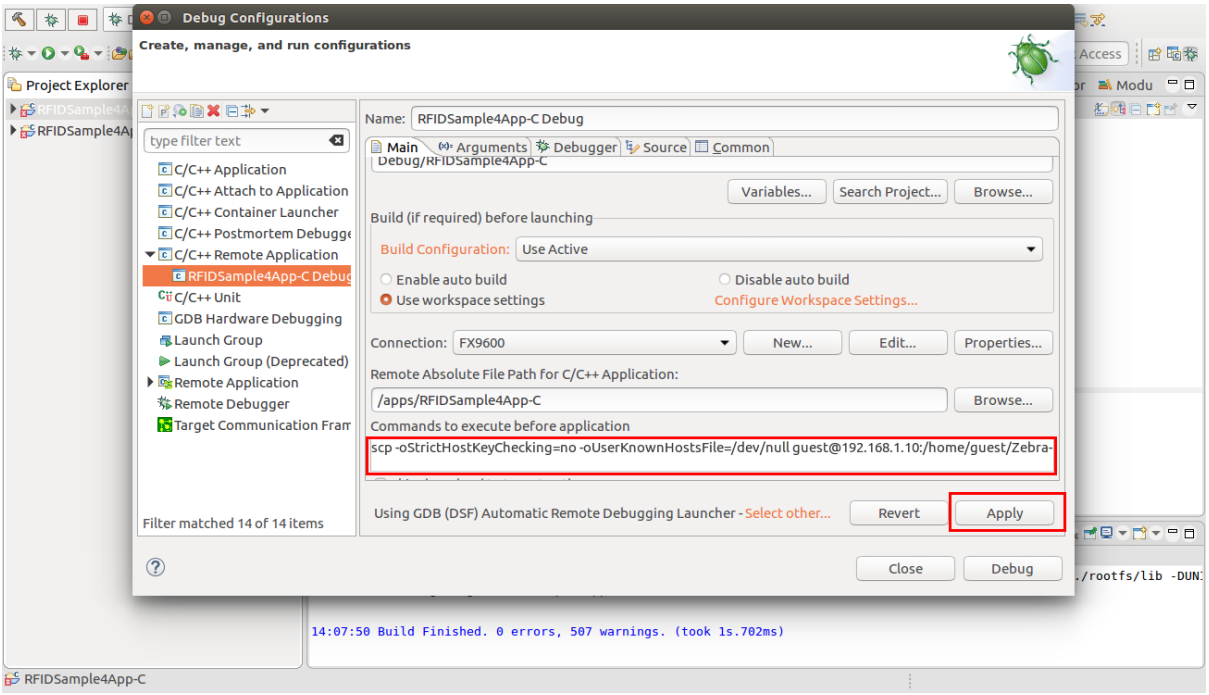
Copy gdbserver file to the RFID Reader target board only for the first time for debugging.
The following command copies the gdbserver file from SDK samples directory from the Linux Ubuntu/Fedora host machine onto the RFID reader target in /tmp directory

for Ubuntu Desktop 16.04
“scp -oStrictHostKeyChecking=no -oUserKnownHostsFile=/dev/null <user>@<Linux-host-IP>:/<installation path>/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/samples/gdbserver /tmp”

Note: for Ubuntu Server 16.04 edition use the command given below
“scp -oKexAlgorithms=+diffie-hellman-group1-sha1 -oStrictHostKeyChecking=no -oUserKnownHostsFile=/dev/null <user>@<Linux-host-IP>:/<installation path>/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/samples/gdbserver /tmp”

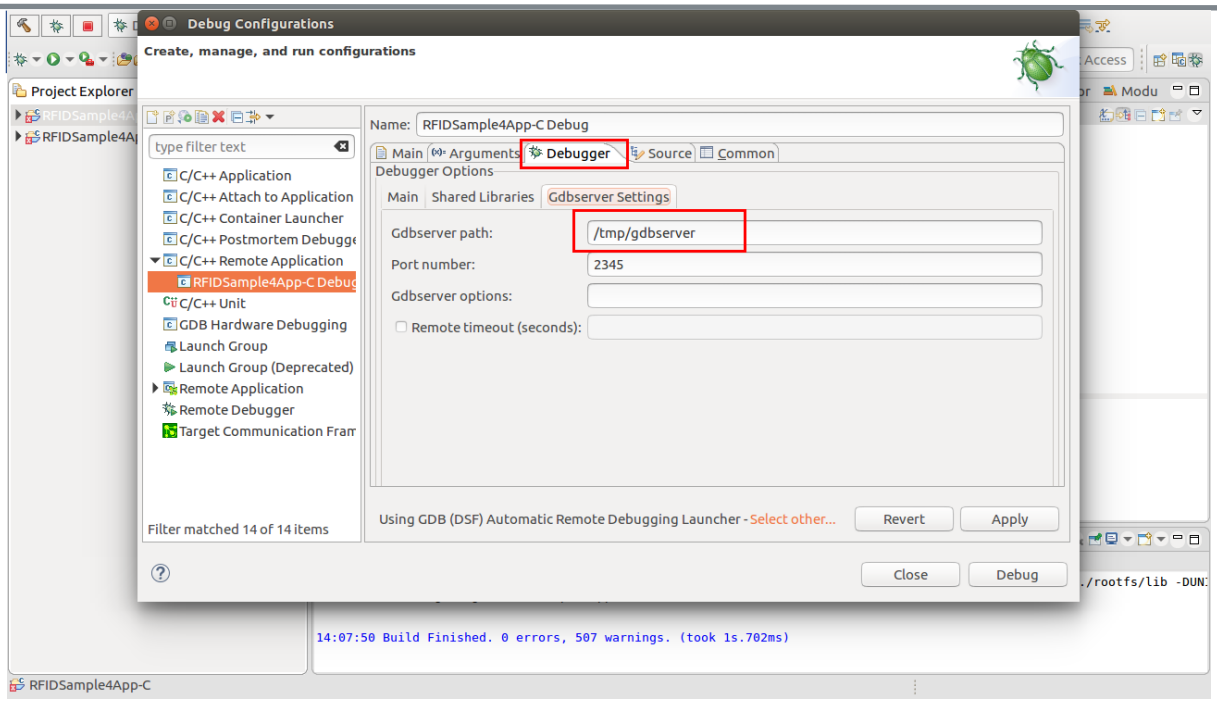
Above command downloads gdbserver file from Linux Ubuntu/Fedora host machine (IP address whose IP address is given in the command) then followed by samples directory path on Linux Host Ubuntu/Fedora machine, "/tmp" directory specify the path on the RFID reader where gdbserver file is downloaded.
Paste the above command given in bold in the “Commands to execute before application is run” box and click Apply a shown below.

Figure 26: Commands to execute before application



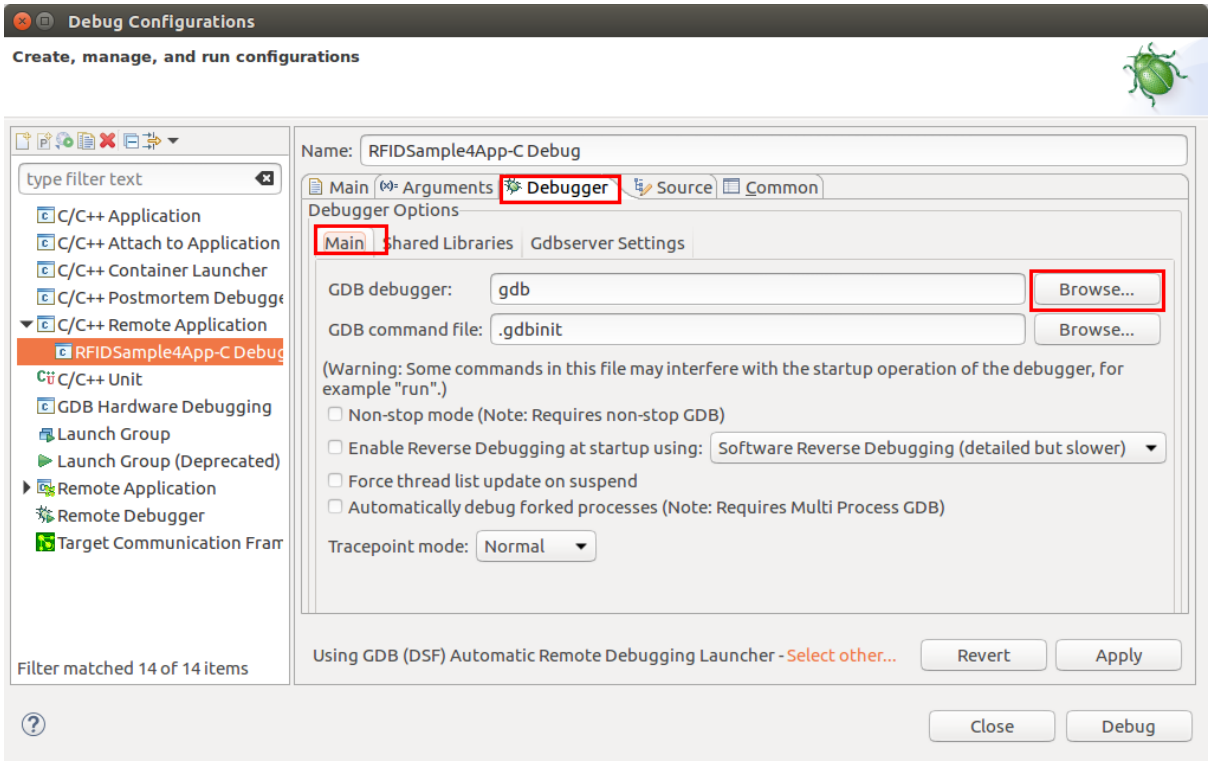
Make sure “Gdbserver Settings” is set to /tmp/gdbserver

Figure 27: Debugger -> Debugger Options -> Gdb Server settings



Next select Debugger tab -> Debugger Options -> Main tab and click on Browse button

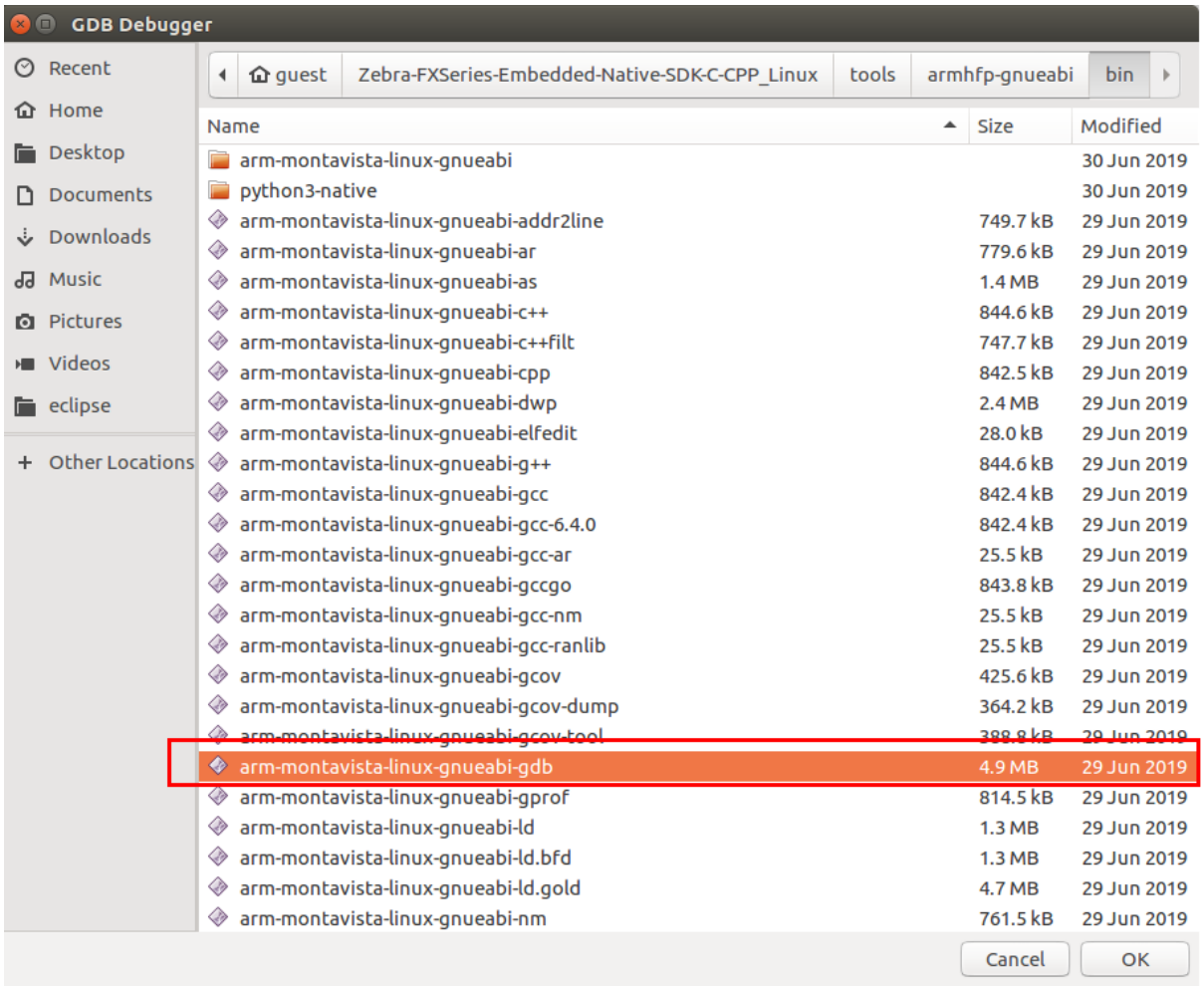
Figure 28: New Connection Debugger Window View



Make sure Cross GDB path is set properly

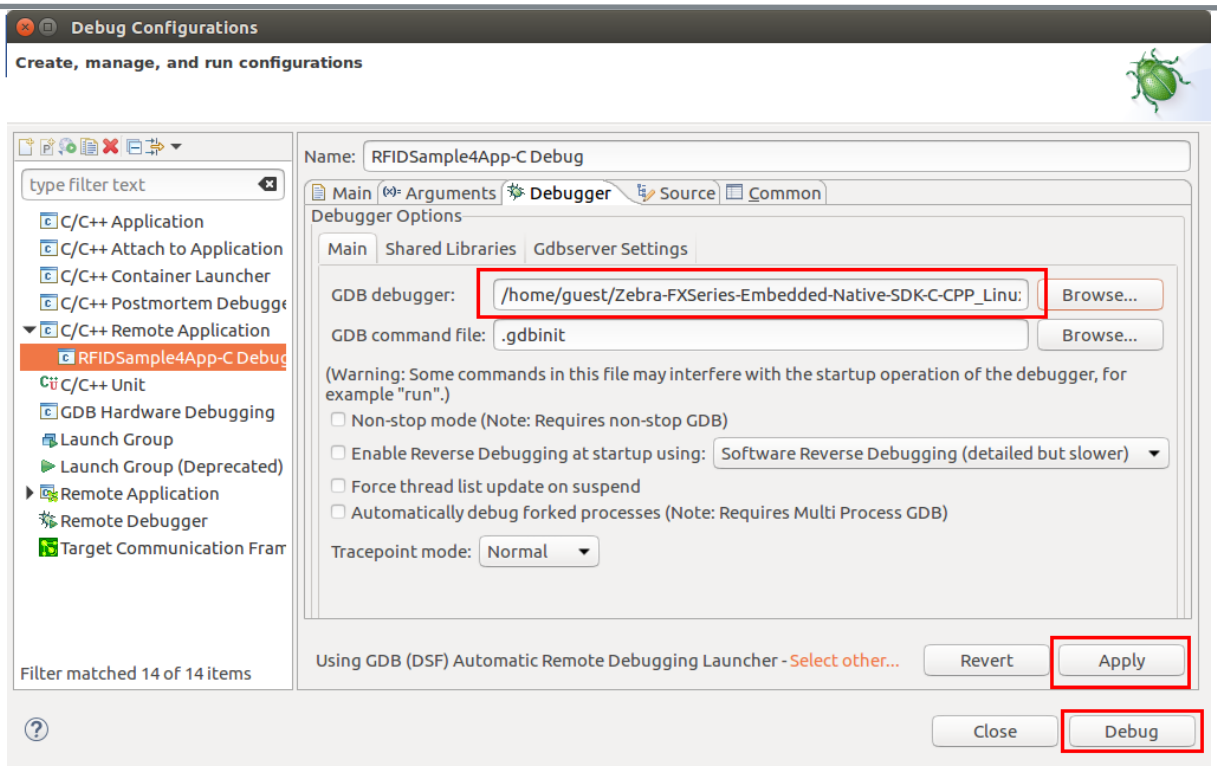
Navigate to **"<install_path>/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux_V1.0.4/tools/armhfp-gnueabi/bin"** directory and select the ARM cross gdb debugger (**"arm-montavista-linux-gnueabi-gdb"**) as shown in the Figure 29 and click OK.

Figure 29: GDB Debugger View



Click “Apply” button in Debug Configurations window
Next click on Debug button as shown in the Figure 30.

Figure 30: GDB Debugger Apply View



The following figures of 31, 32, 33 and 34 may be encountered while setting up the remote connection for the first time with a fresh new Linux user. Click on “Yes”, “OK” buttons as shown below.

Figure 31: Authentication 1

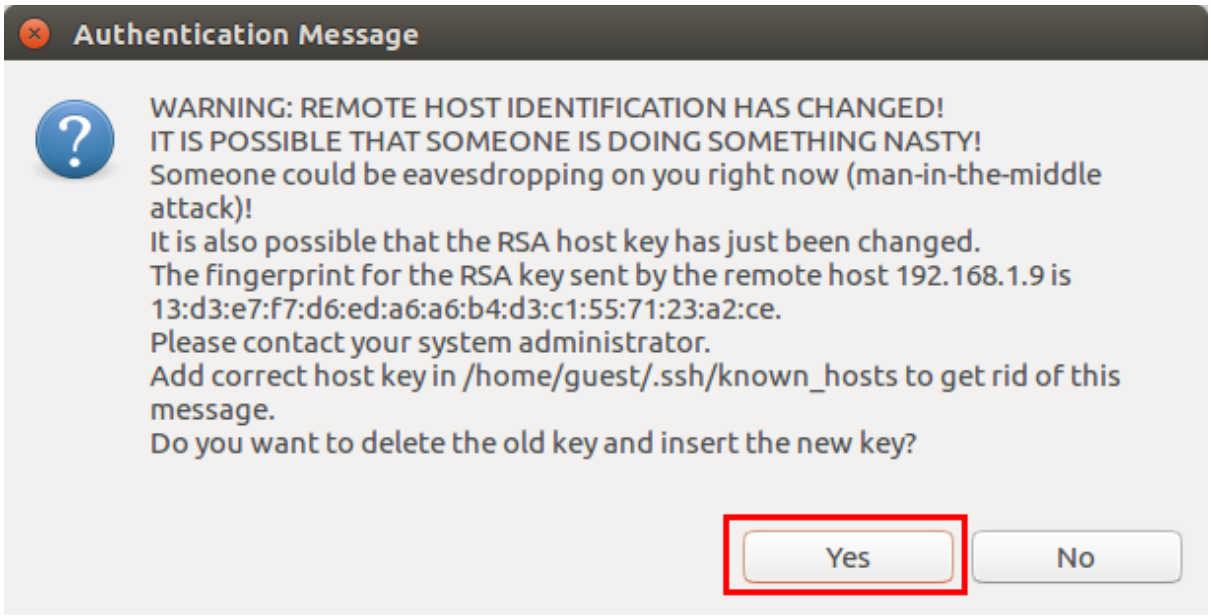


Figure 32: Authentication 2

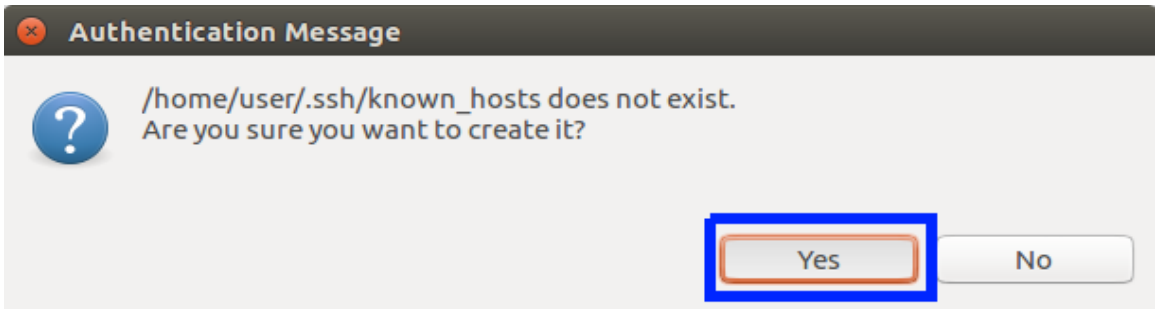


Figure 33: Authentication 3

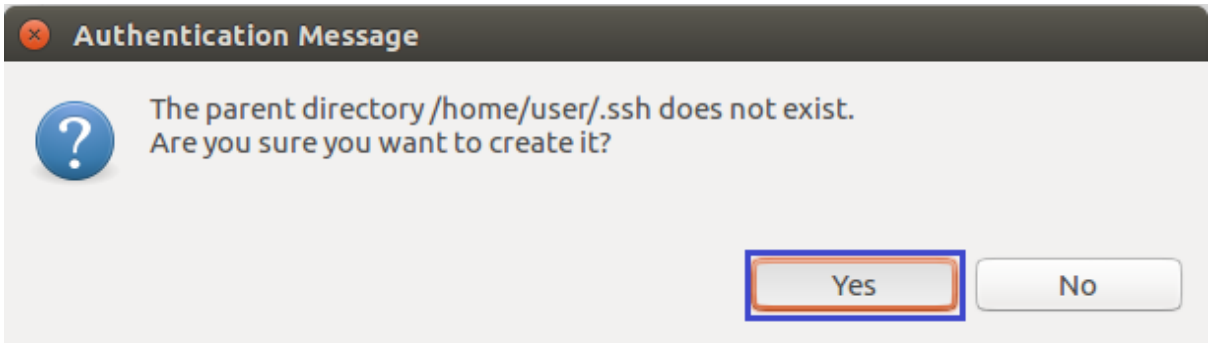
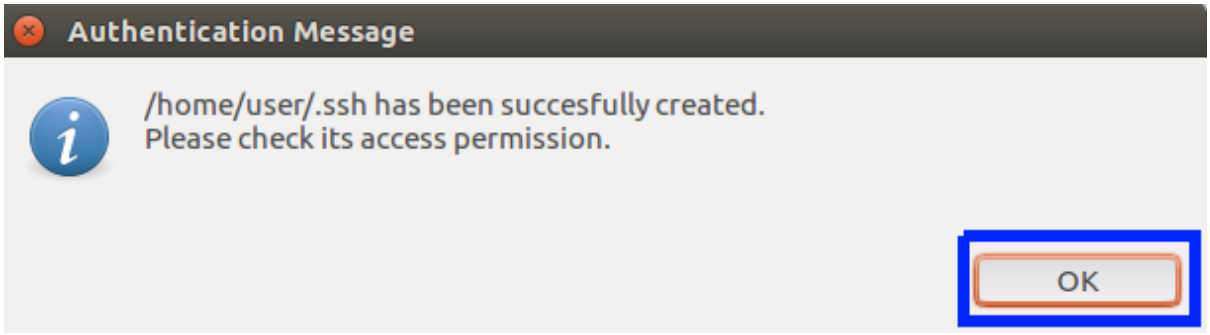
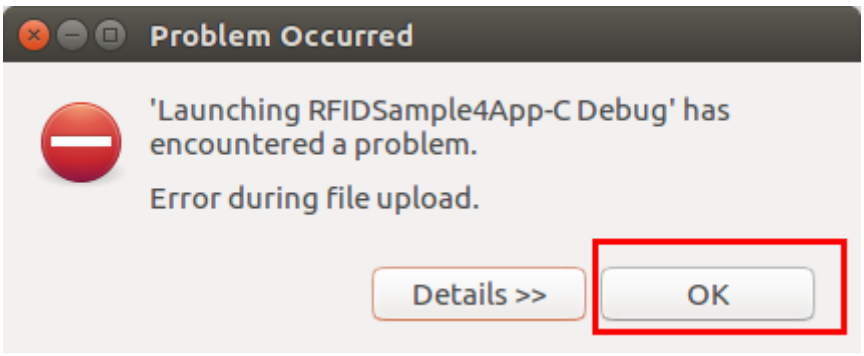


Figure 34 : Authentication 4



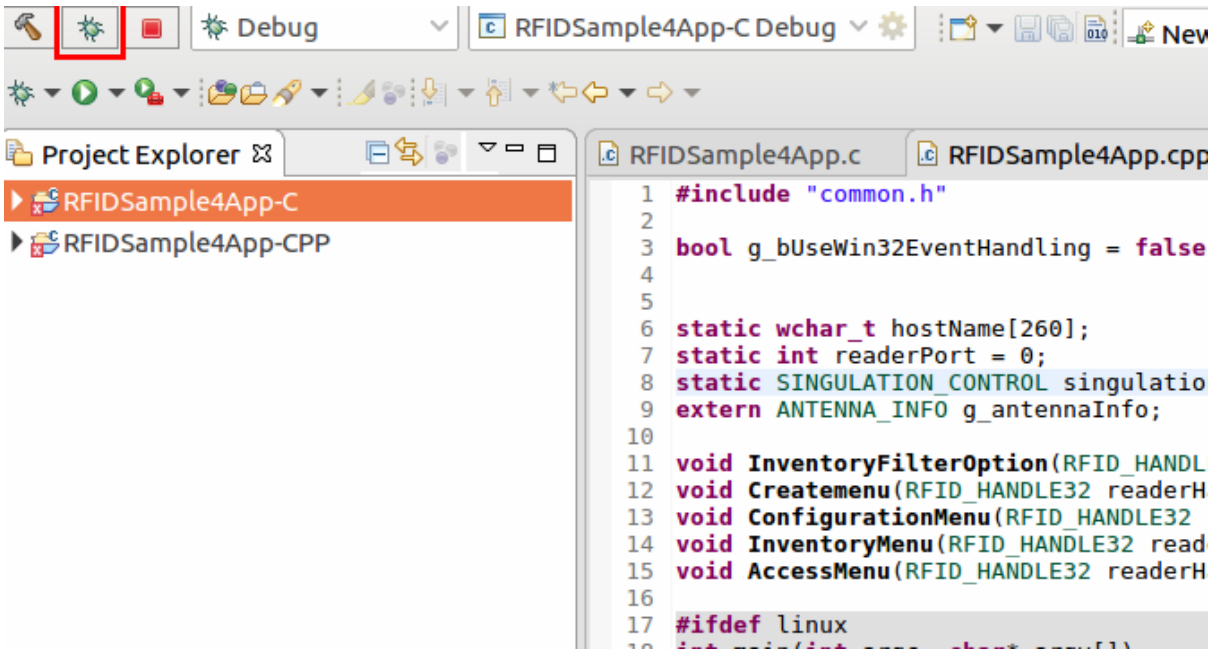
The “Problem Occurred” error window will popup, click on “OK” as per the following figure 35.

Figure 35: Problem Occurred popup view



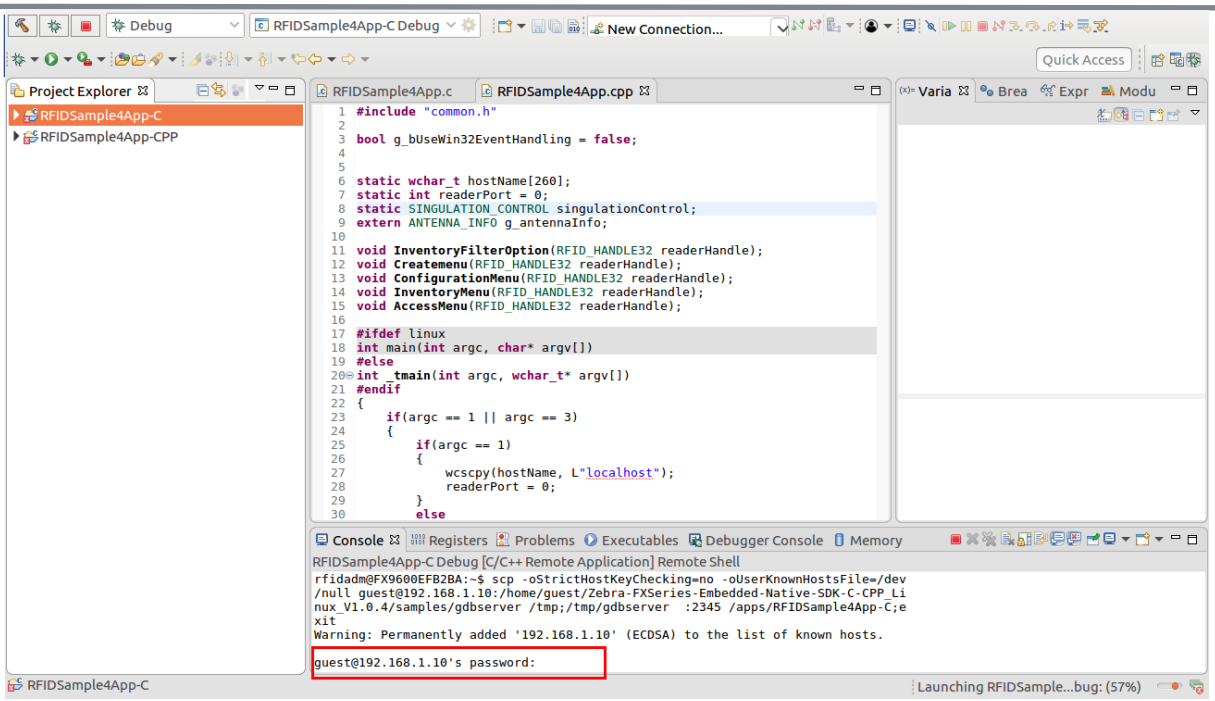
Now click the top left second button, Launch in 'Debug' mode

Figure 36: Launch in 'Debug' mode



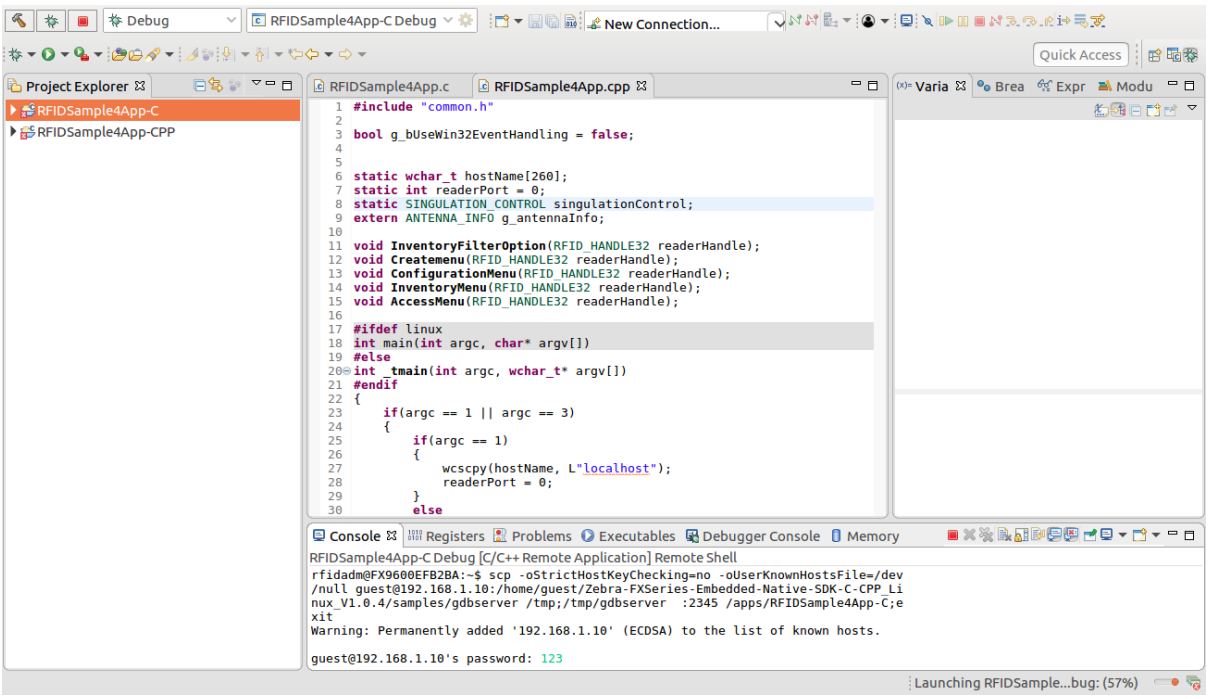
The following console displays the password prompt

Figure 37.1: Password prompt



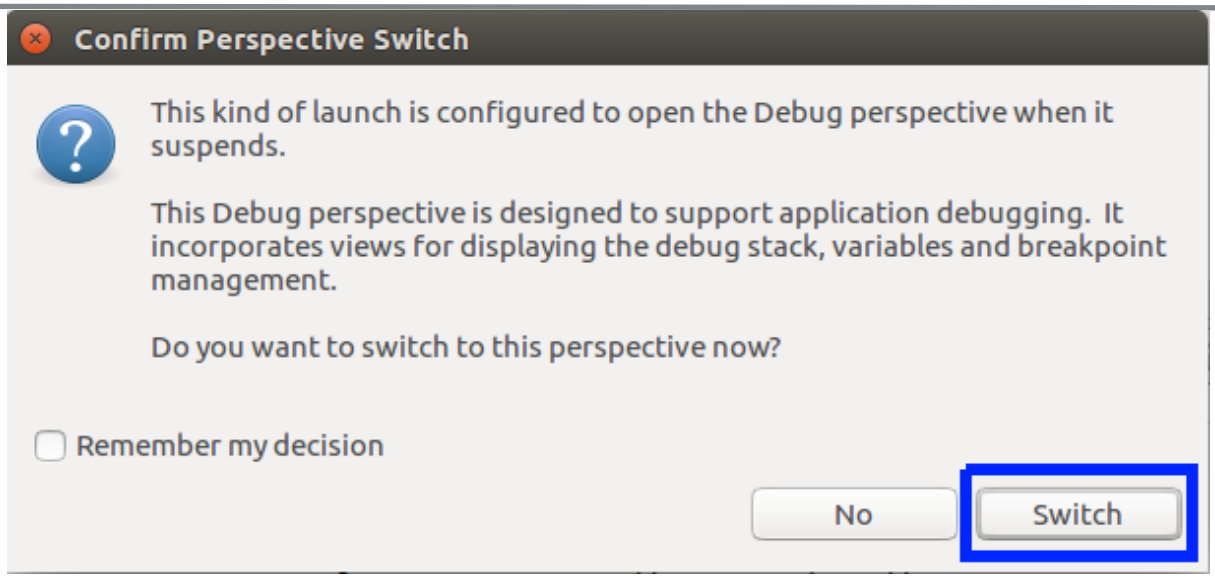
Enter your linux host user password in the console window password prompt

Figure 37.2: Enter Password



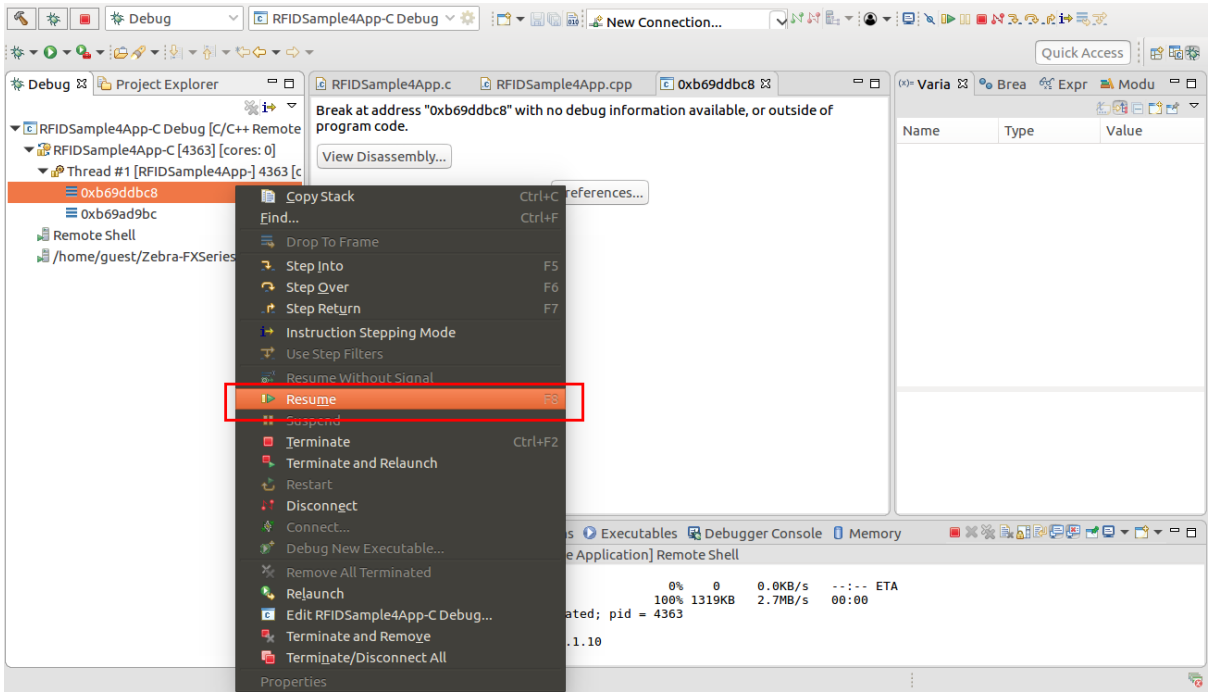
The Confirm Perspective Switch popup window will appear as per figure 36, click “Switch” button

Figure 38: Confirm Perspective Switch



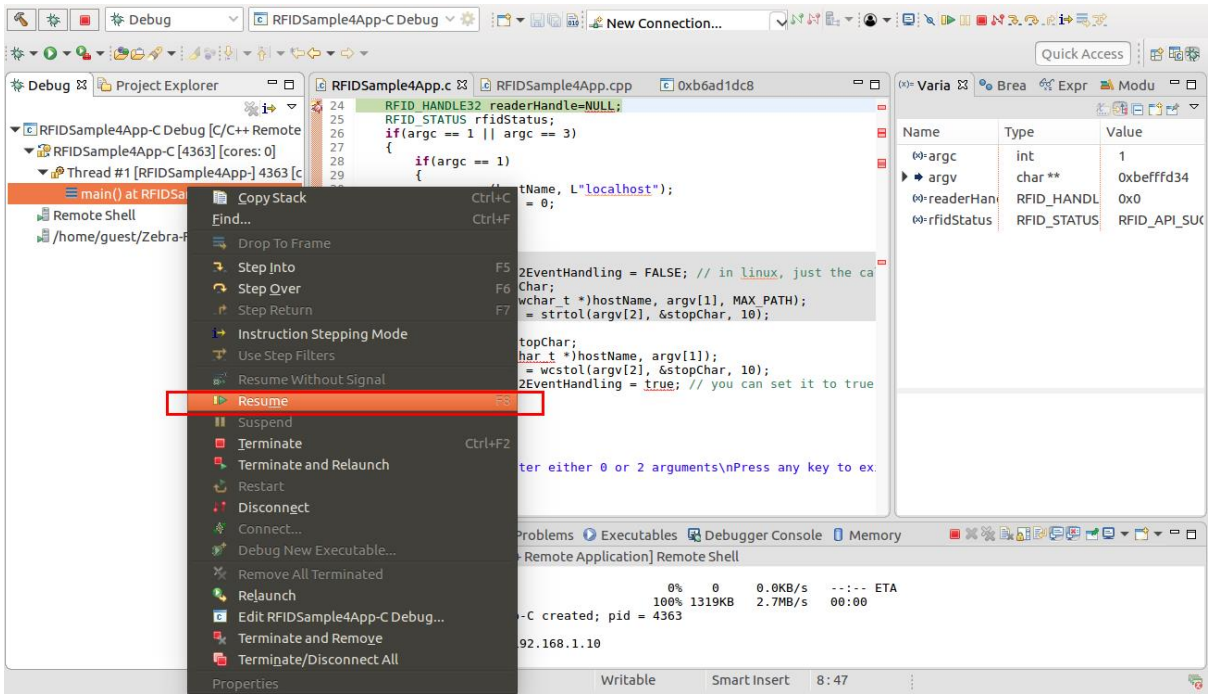
Right click RFIDSample4App-C's pre defined break point address and click Resume for the debug to continue as shown in the figure 39

Figure 39: Debug Console view



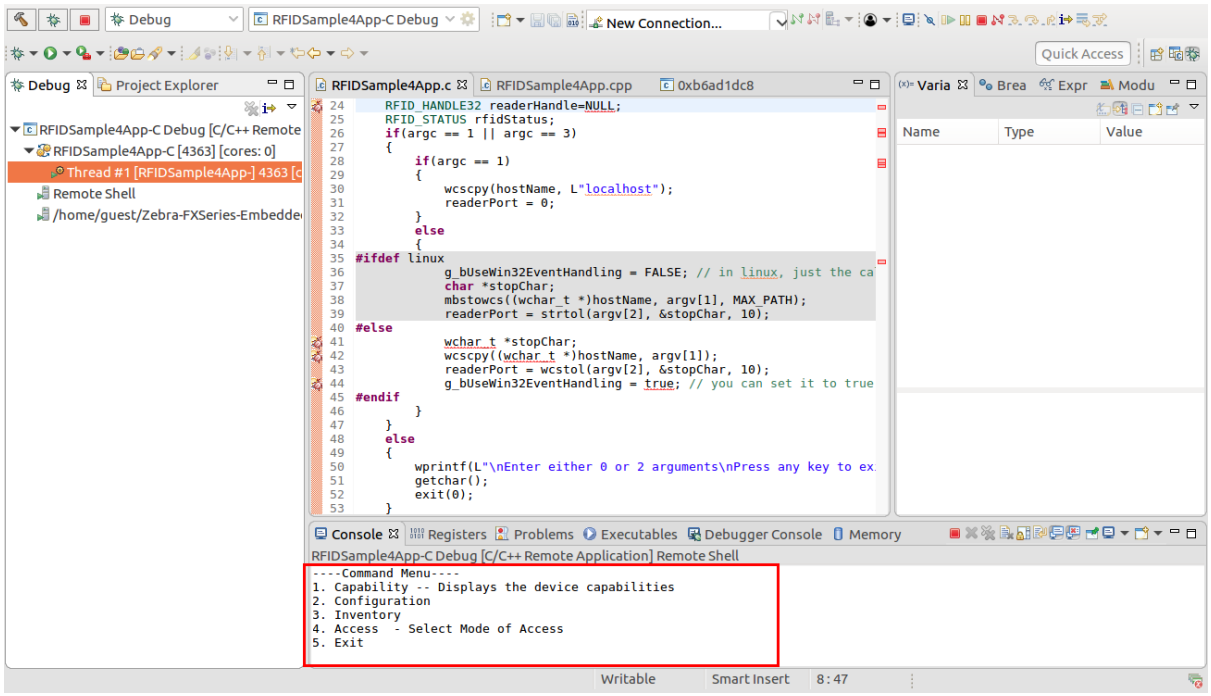
The application will halt at the pre-defined breakpoint
Next click on the resume button again

Figure 40.1: RFIDSample4App-C console breakpoint



The application output will be seen as per figure 40.2

Figure 40.2: RFIDSample4App output View



7.0 Embedded RFIDSample4App C/C++ Application from scratch

This section describes the detailed steps to create embedded C/C++ RFIDSample4App from scratch. The steps involved are:

- Create Workspace
- Create Project
- Add sources and header files
- Add header file include path, library path, libraries
- Add compiler and linker flag
- Clean and build steps
- Debug embedded C/C++ RFID application
- Creation of start and stop script for C/C++ installation package

NOTE :

Wherever the steps for C & C++ application project/workspace differ, it will be mentioned here with step/screenshot of C application project/workspace first, followed by the step/screenshot for C++ application project/workspace. If not stated separately then one can assume the steps are same. **Figure xx(a)** is for **C application project/workspace** and **Figure xx(b)** is for **C++ application project/workspace**.

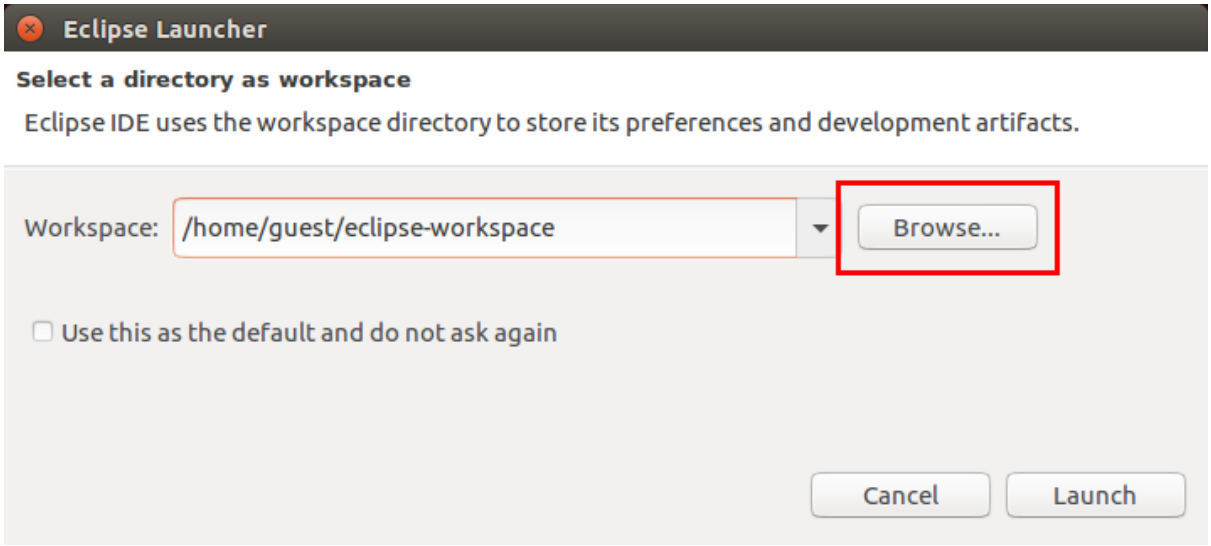
(Also for C project work with .c files and for C++ project work with .cpp files).

The assumption is both C /C++ application project doesn't belong to same workspace. It is show both projects are under it own worrkspace.

7.1 Creating a Workspace

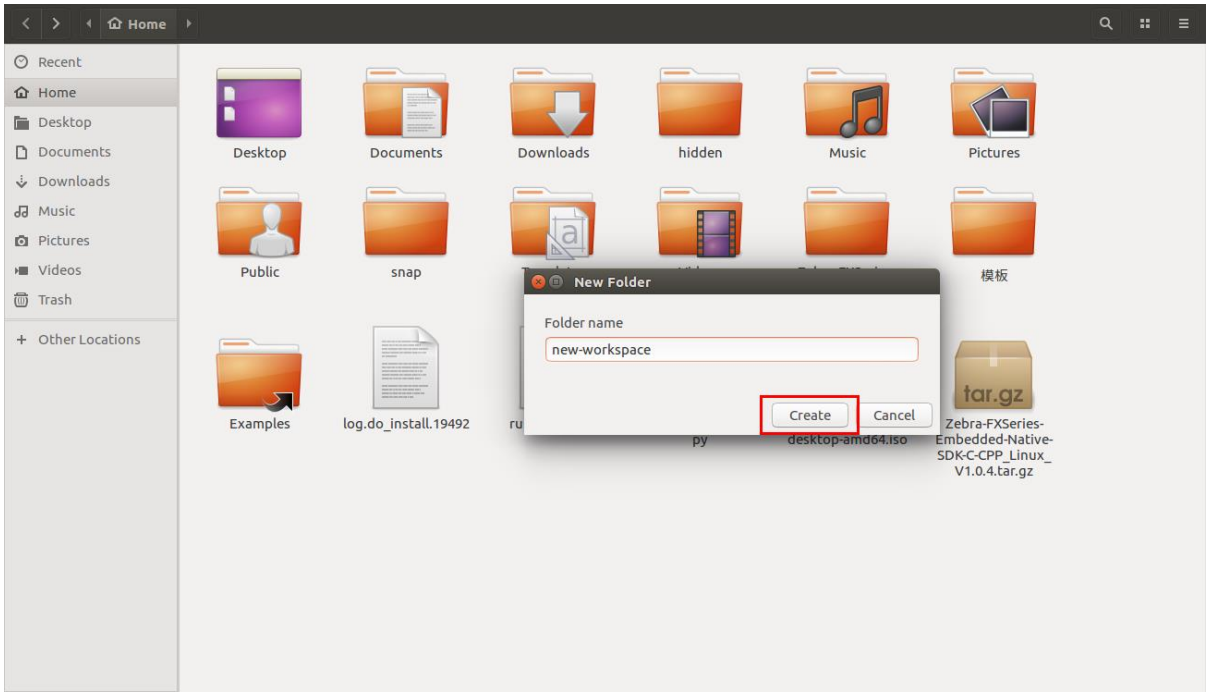
Navigate to '[installation-path]'/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/eclipse/'.
Double-click on eclipse executable file.
Under Eclipse Launcher popup window, click on Browse for workspace as per figure 41.

Figure 41: Eclipse Launcher



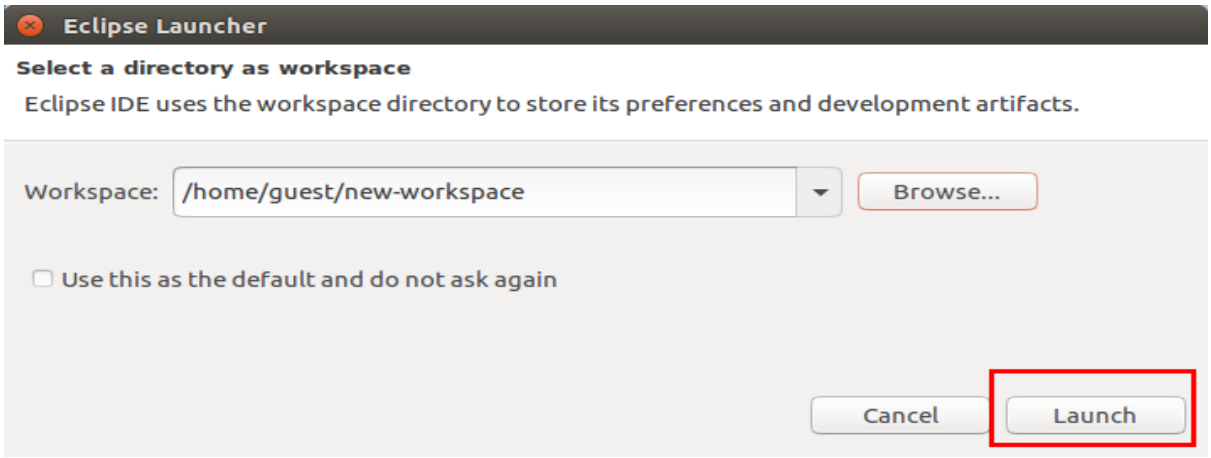
Select the directory path where the RFID sample application has to be created from scratch. The following figure 42 shows the creation of new directory for this.

Figure 42: Creating new workspace



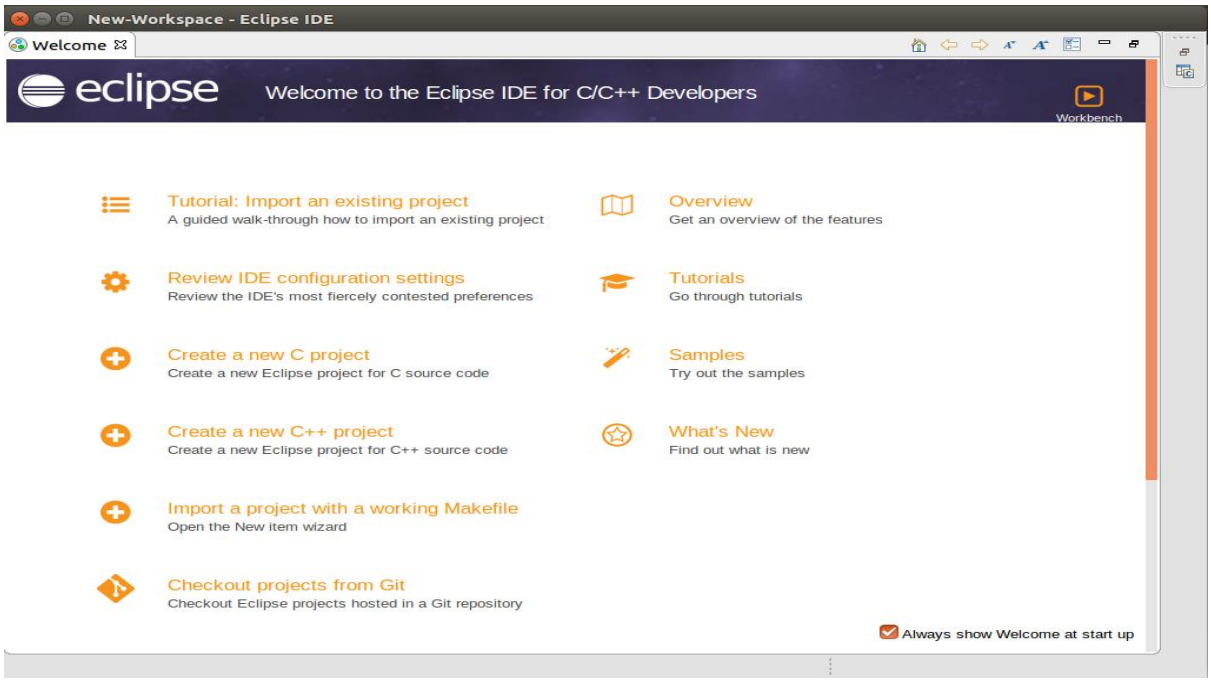
The following figure 41 depicts selection of newly created directory as workspace directory. Click on Launch

Figure 43: Eclipse Launcher



Following eclipse welcome screen shows up

Figure 44: Welcome to the Eclipse

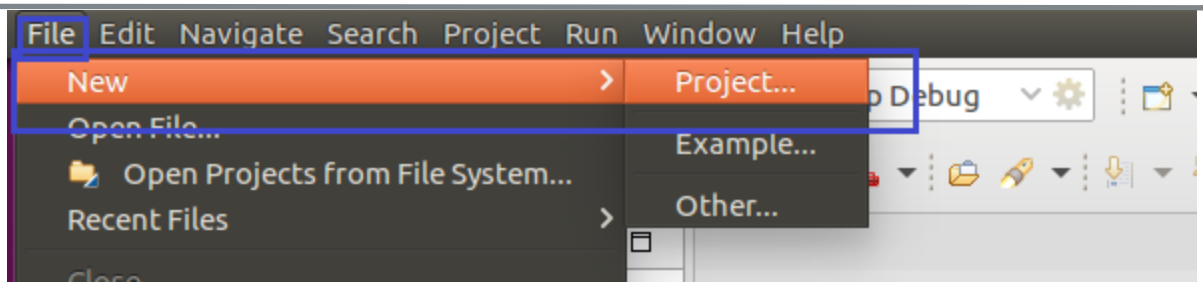


Close this Welcome tab inside eclipse.

7.2 Creating an Embedded RFIDSample4App C/C++ Project

Create a new Project RFIDSample4App in the Eclipse
Select File->New->Project

Figure 45: C Eclipse IDE View



Expand C/C++ Folder and Select “C Project” and click “Next” button as shown in the figure 46(a)

Figure 46(a): New Project view for C application

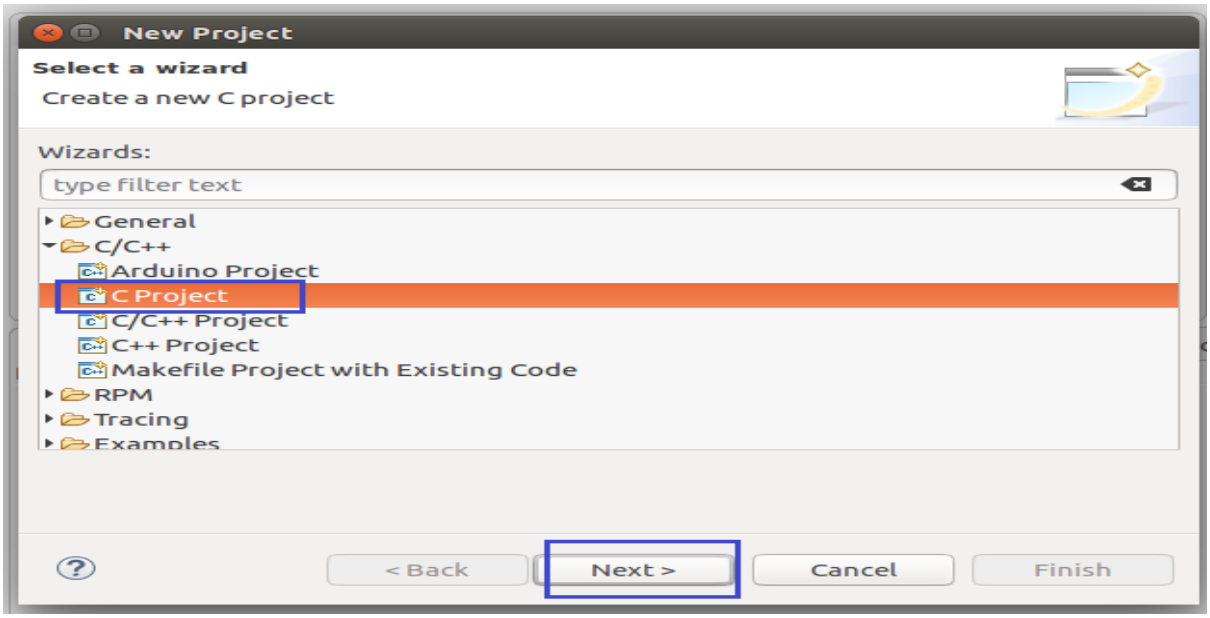
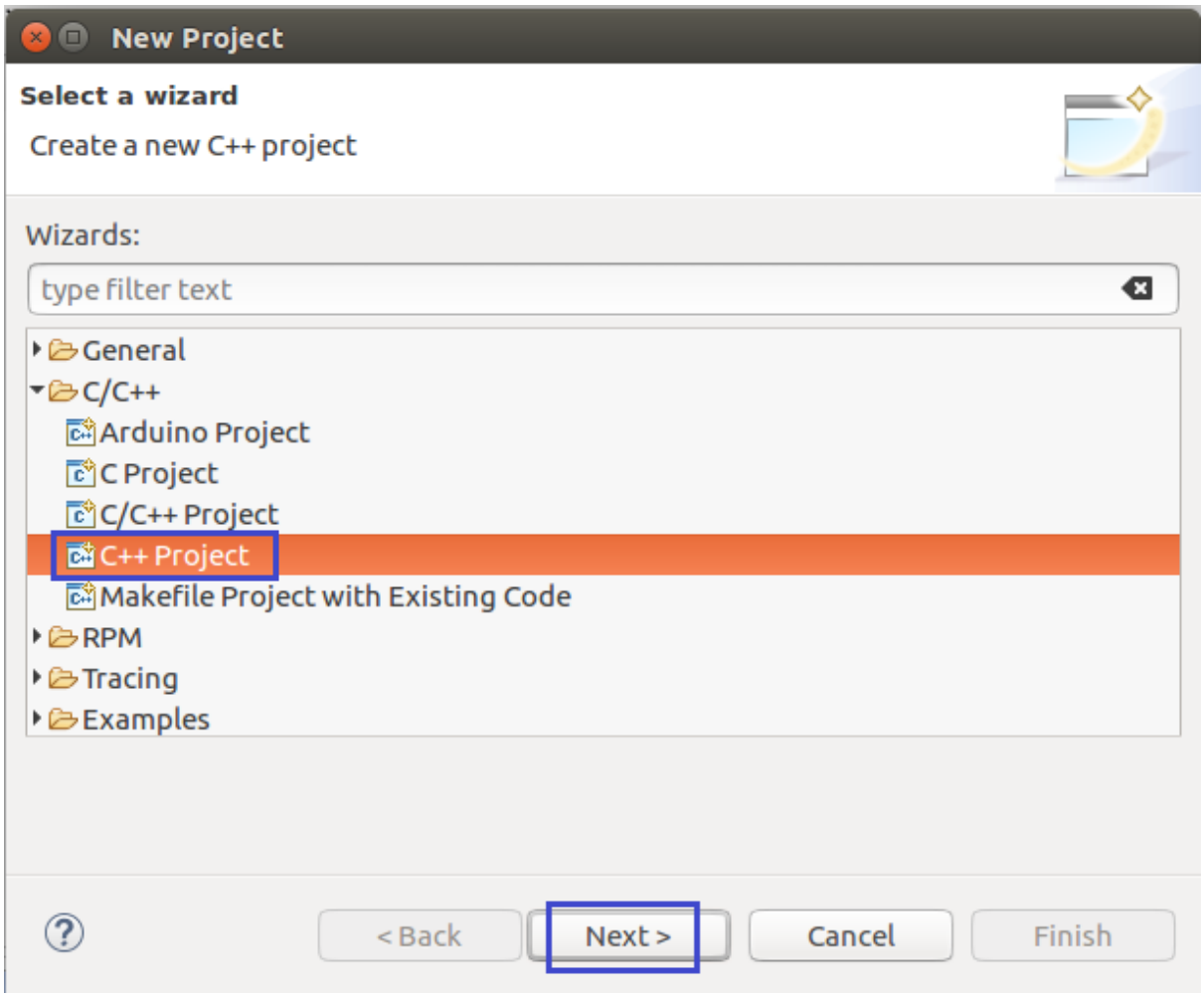
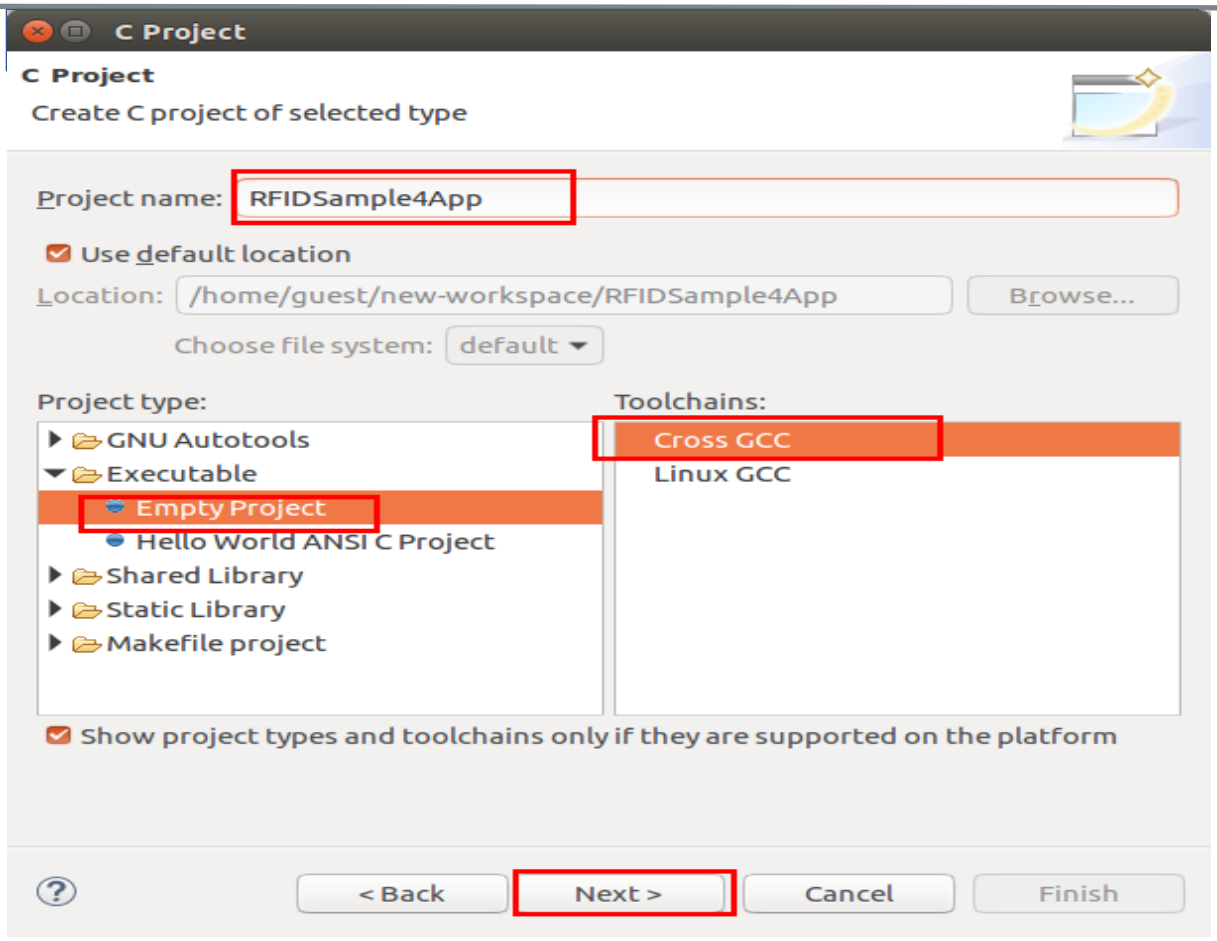


Figure 46(b): New Project view for C++ application



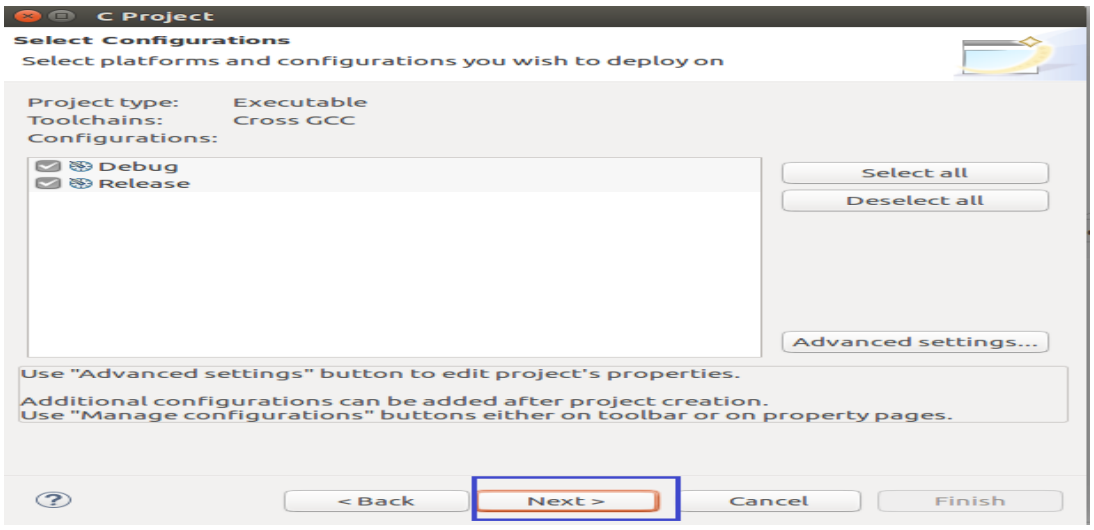
In the C Project Window or C++ Project Window, as the case may be, enter the Project name as "RFIDSample4App"
For Project Type select "Empty Project"
For Toolchain select "Cross GCC" and select "Next" button as shown in figure 49.

Figure 49: New Project Select Type view



Select “Next” button in Figure 48

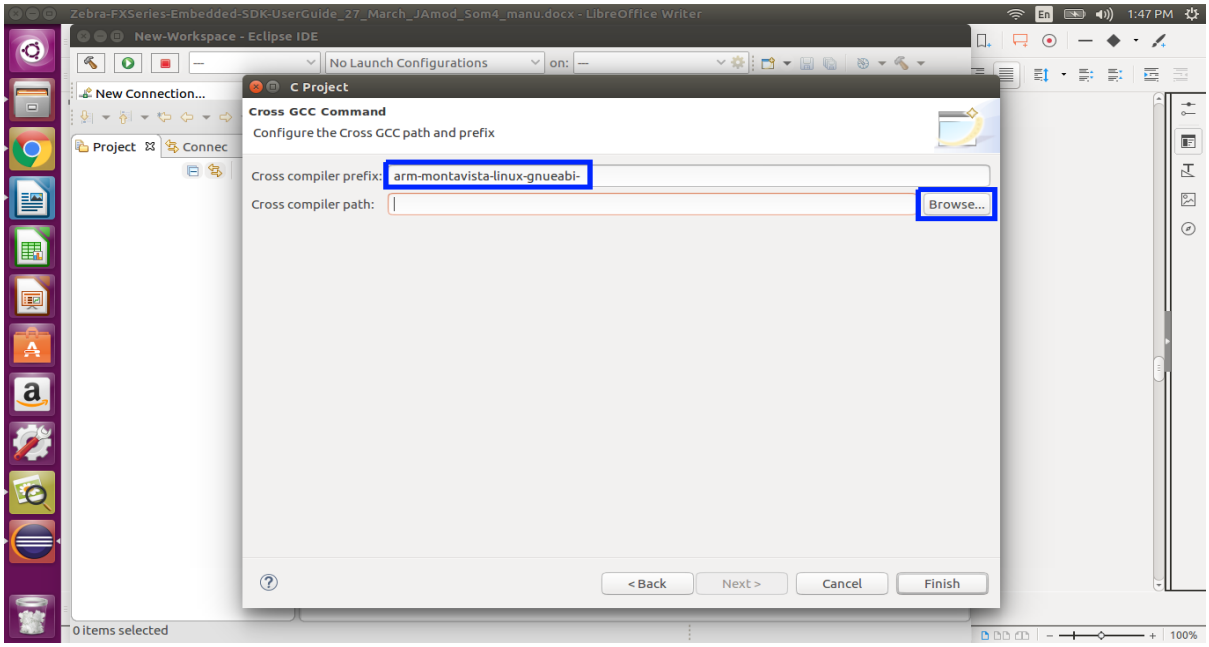
Figure 48: Create C or C++ Project View



Under C or C++ Project window, Set the following parameters.

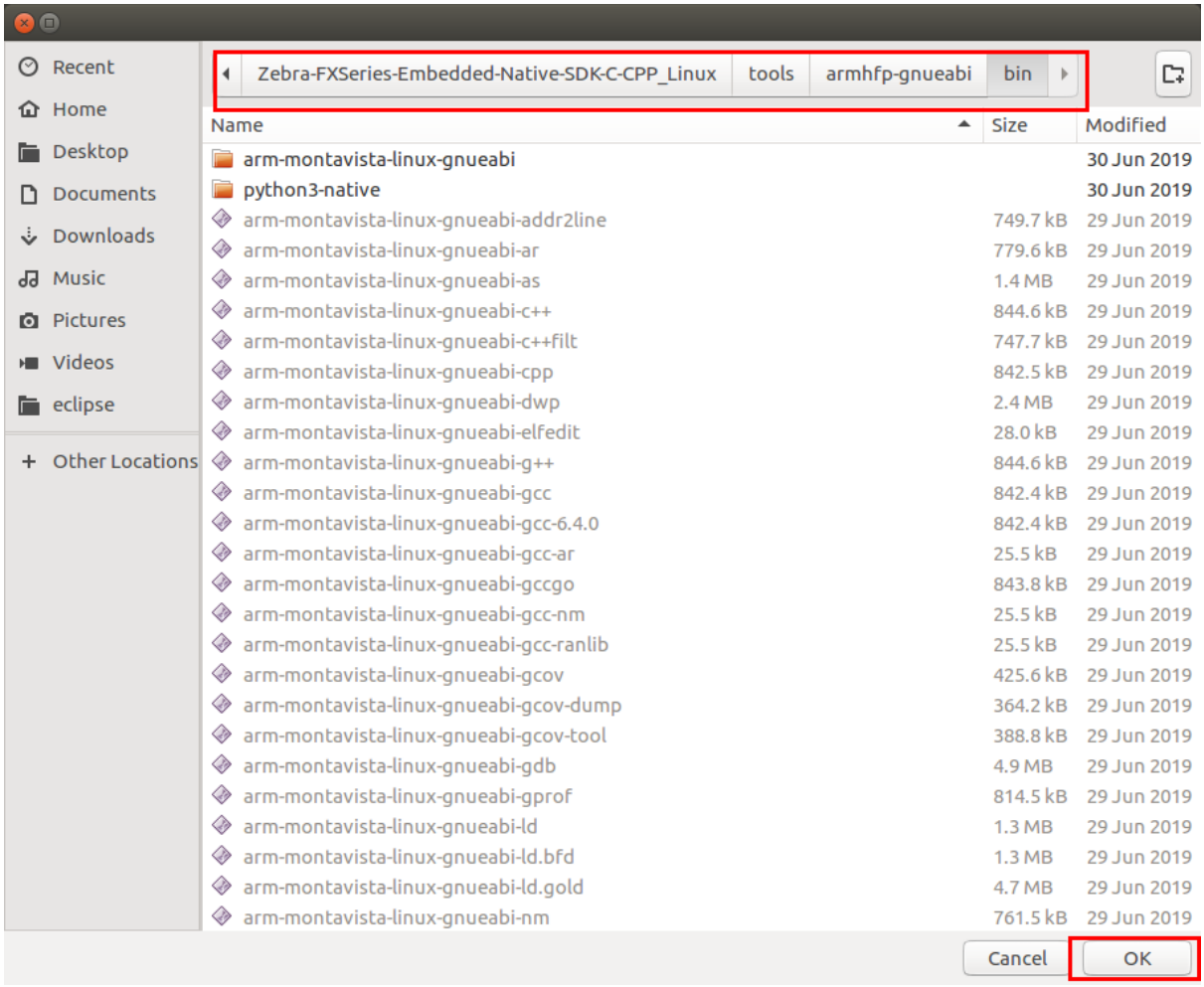
1. Cross Compiler Prefix -> “arm-montavista-linux-gnueabi-“.
2. Click on Browse button to set the Cross-compiler path as shown in figure 49.

Figure 49: Project Finish Button view



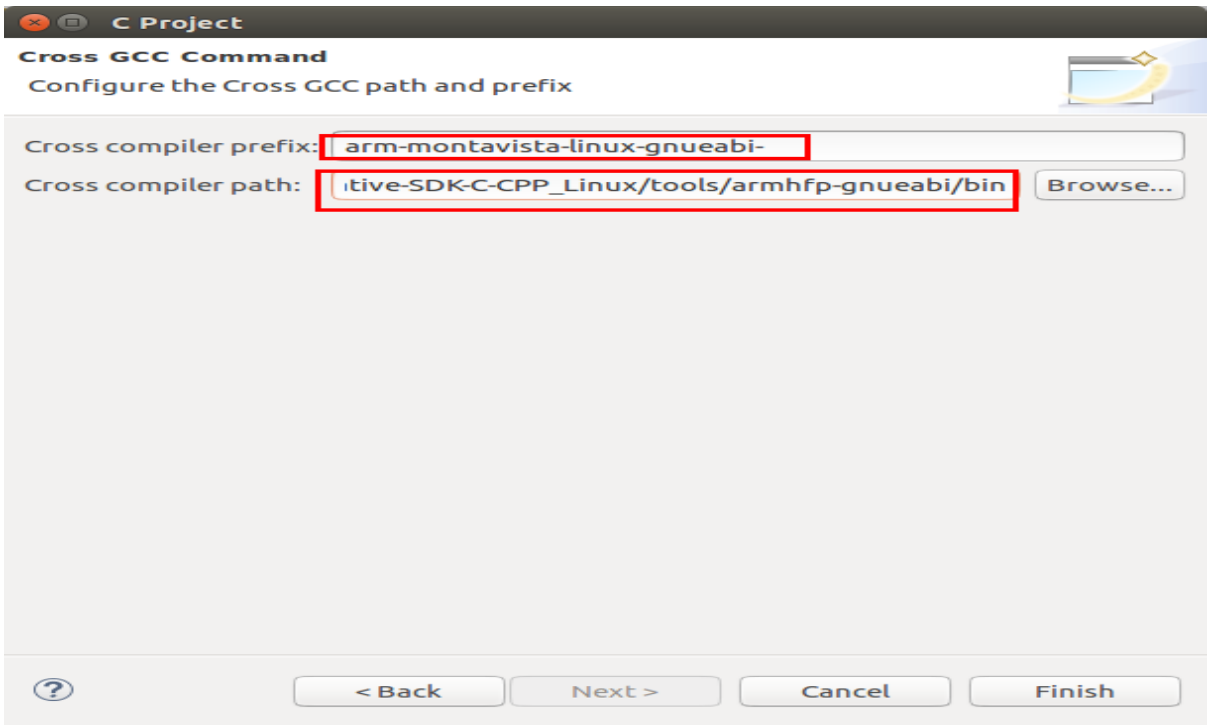
Select “[Installation-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/tools/armhfp-gnueabi/bin” directory and click OK

Figure 50: Cross Compiler Path Selection



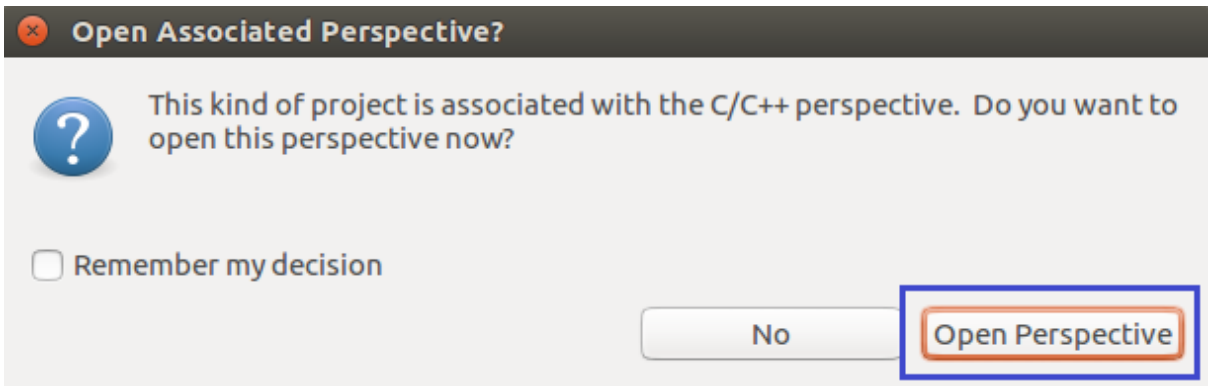
Click on Finish button.

Figure 51: Cross Compiler Path Selection



Open Associated Perspective window may appear, click on Open Perspective button.

Figure 52: Open Perspective view

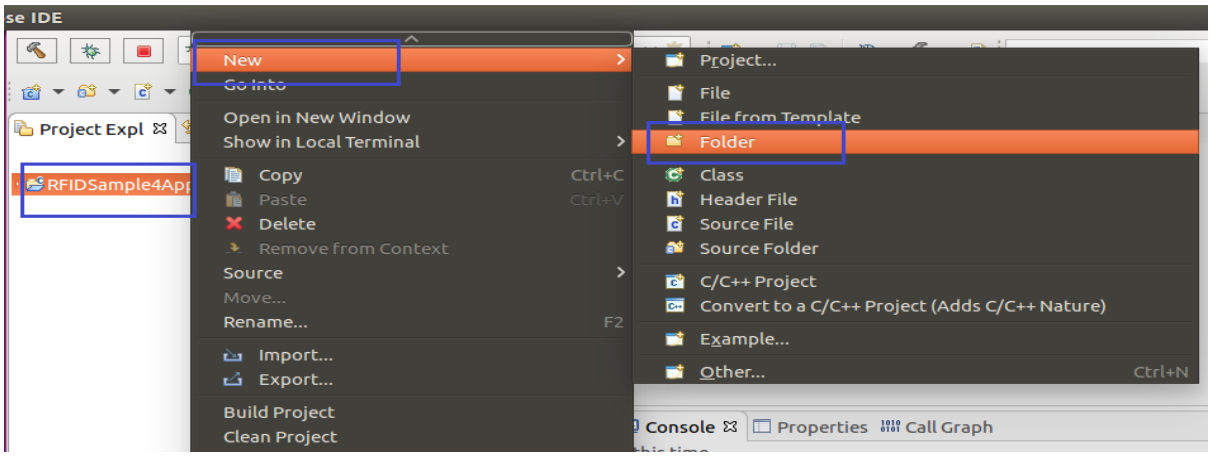


7.3 Adding Source Files to Embedded RFIDSample4App C/C++ Project

In this section source files will be added to RFIDSample4App project
Add “inc” and “src” directory

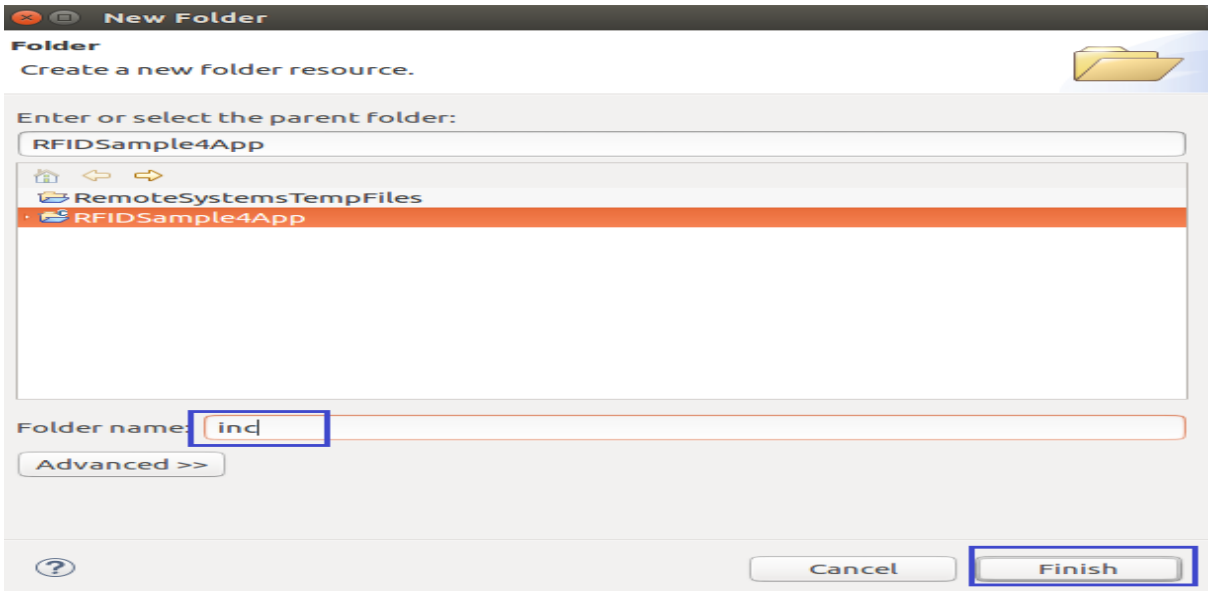
In the Project Explorer View, right-click on the project RFIDSample4App, Select->New->Folder.

Figure 53: RFID Folder Creation view



Select “RFIDSample4App”
Enter Folder name as “inc” as shown in the figure 52 and click “Finish” button.

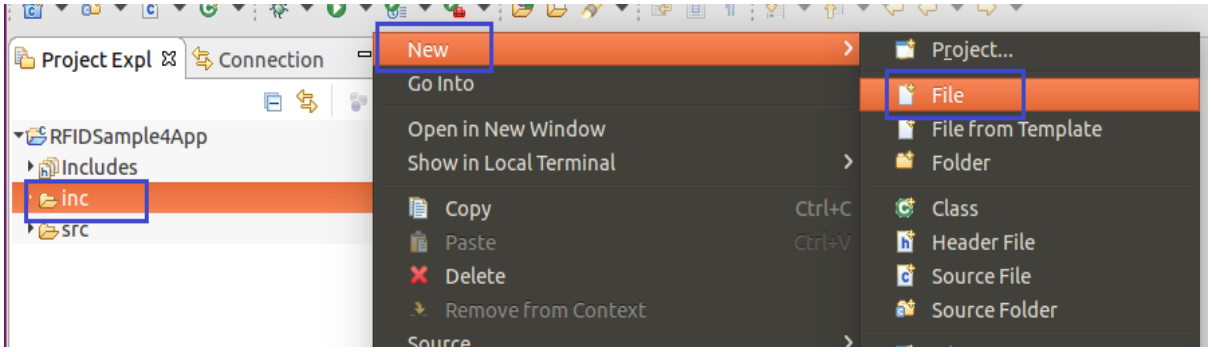
Figure 54: RFID Folder Name view



Add “src” directory following same above steps as mentioned to add “inc” directory. Similarly, create ‘’src” folder under “RFIDSample4App” project.

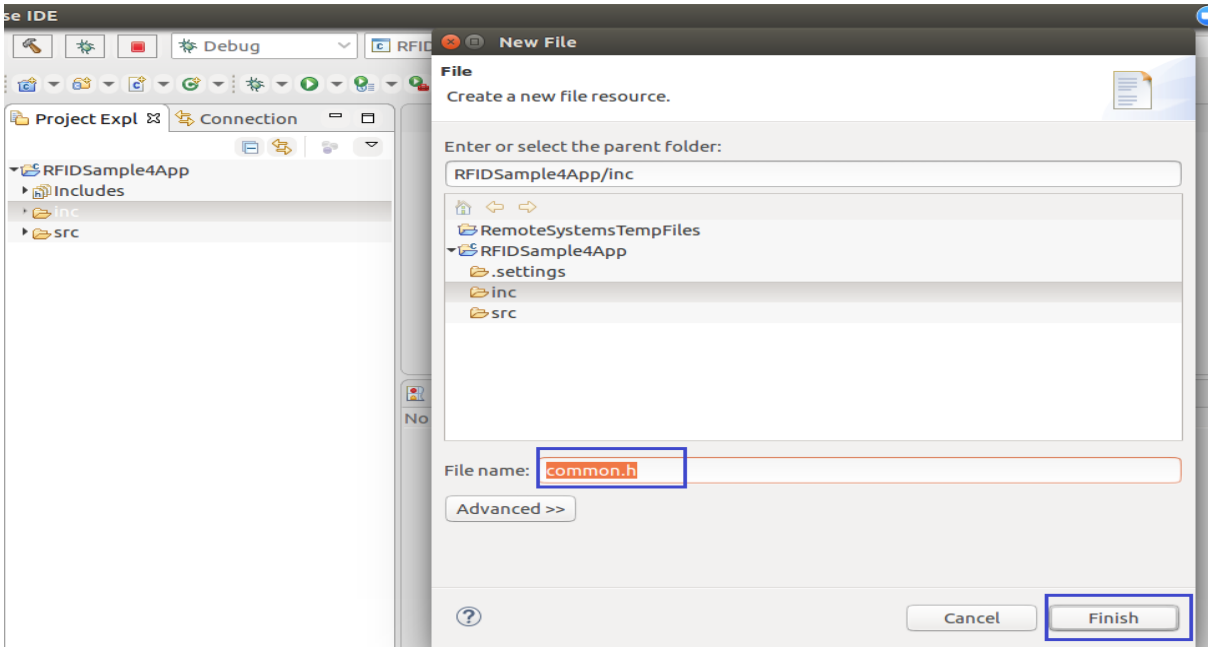
Add source files to “inc” and “src” directory
Right-click on “inc” directory, select New->File as shown in the figure 55

Figure 55: RFID inc view



Enter the File name as “common.h” the header file as shown in the figure 56 and click “Finish” button.

Figure 56: common.h view



Similarly, add “common.c” and “RFIDSample4App.c” source files to “src” directory
Listing of files is shown below in the figure
List of Files added to project

Figure 57(a): List of Files in RFIDSample4App project for C application

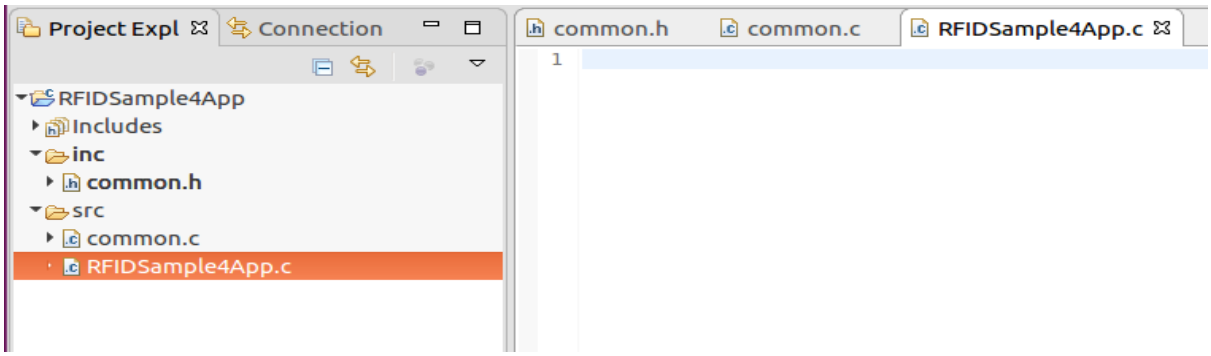
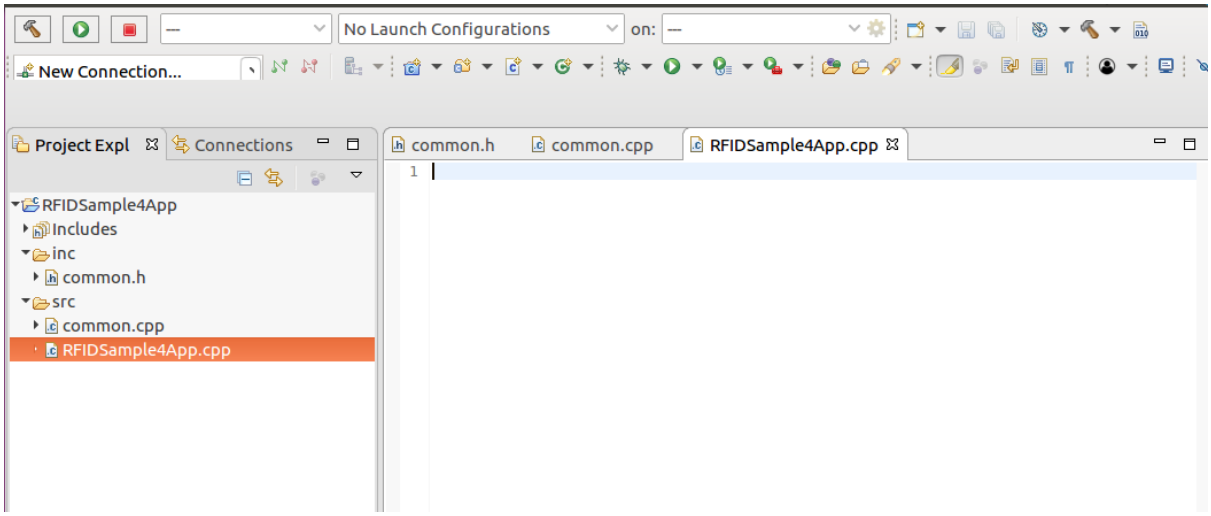
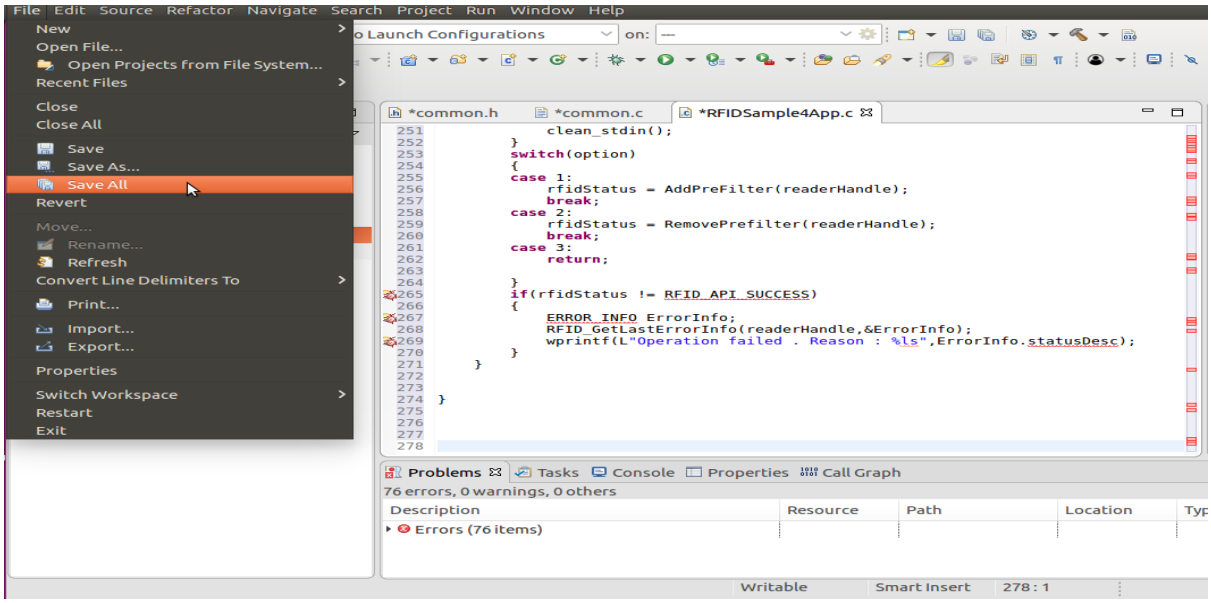


Figure 57(b): List of Files in RFIDSample4App project for C++ application



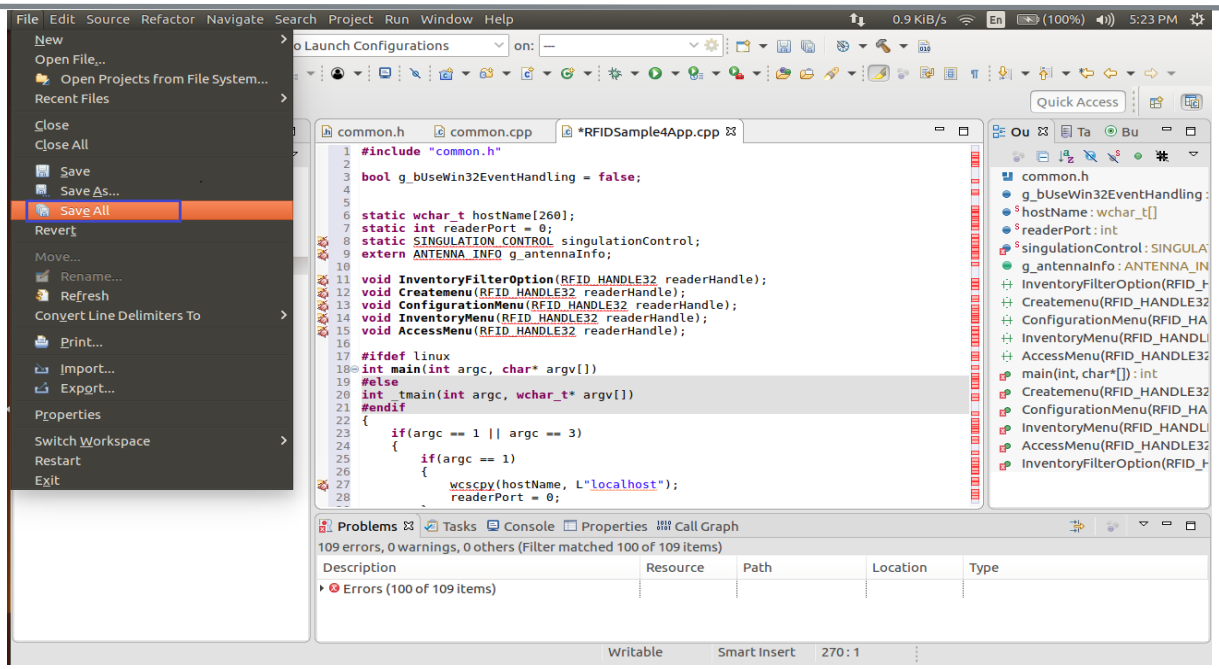
Copy the source code from the samples C directory (from the untarred package "[install-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/samples/workspace/RFIDSample4App-C",
copy the respective files from inc and src directories) provided and save it by clicking Save All.

Figure 58(a): Adding source files for C application



For C++ application project, copy the source code from the samples C++ directory (from the untarred package "[install-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/samples/workspace/RFIDSample4App-CPP",
copy the respective files from inc and src directories) provided and save it by clicking Save All.

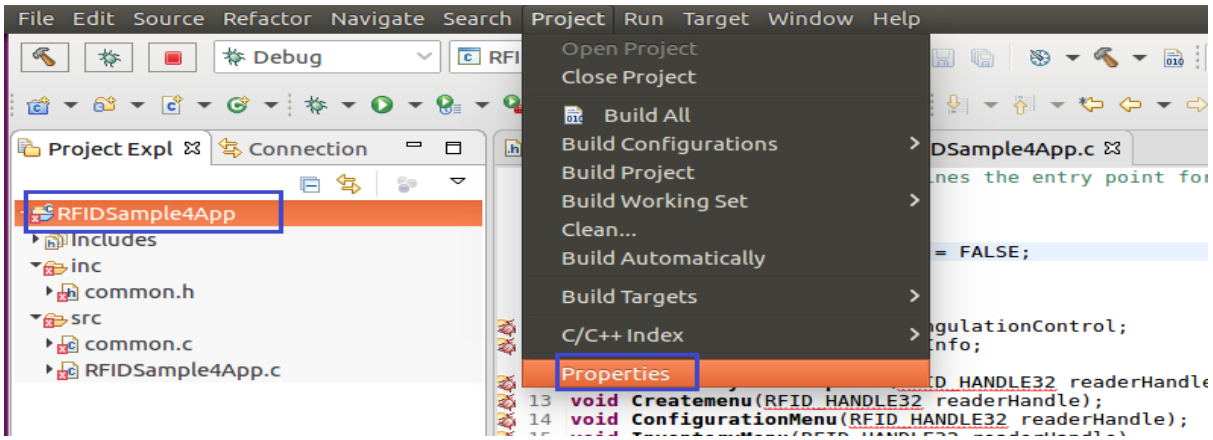
Figure 58(b): Adding source files for C++ application



7.4 Setup Cross Compiler and Library Environment for Embedded Native C/C++ Project

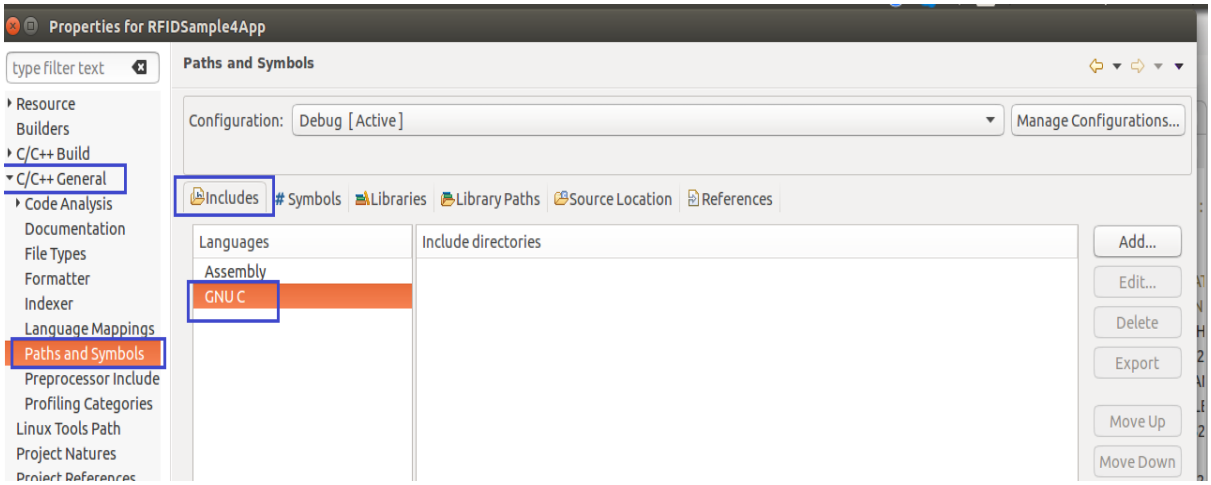
In the Project Explorer view, select RFIDSample4App project, click on Project->Properties as shown in Figure 59.

Figure 59: Project Properties



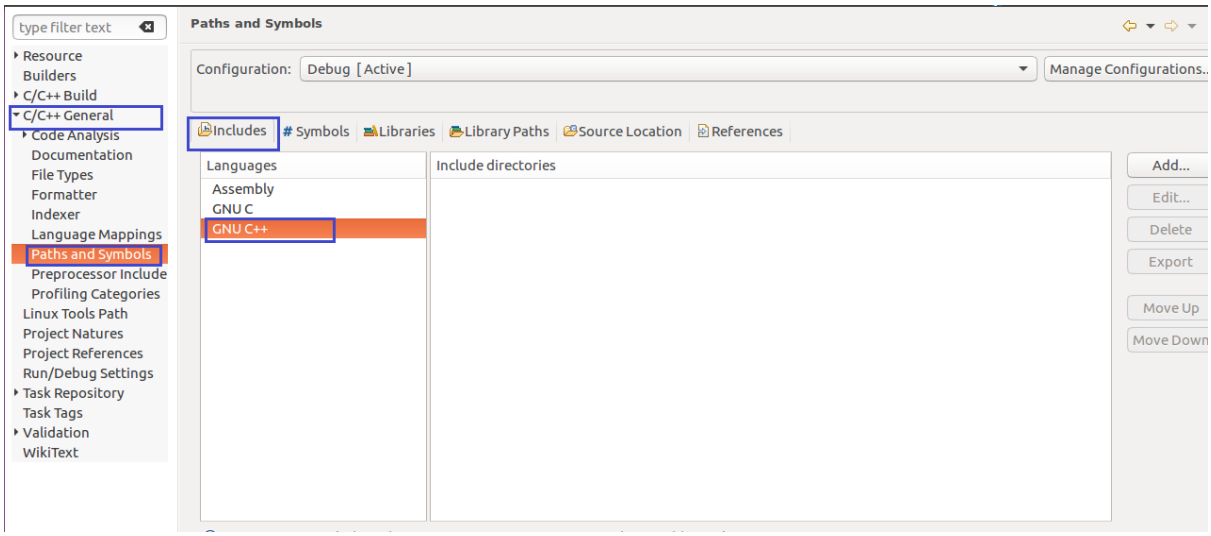
In the Properties Window of RFIDSample4App project
Select and expand “C/C++ General“ in left panel
Click on “Path and Symbols”
Under “Includes” tab, Select “GNU C” as language for C application

Figure 60(a): Properties Window RFIDSample4App for C application



In the Properties Window of RFIDSample4App project
Select and expand “C/C++ General“ in left panel
Click on “Path and Symbols”
Under “Includes” tab, Select “GNU C++” as language for C++ application

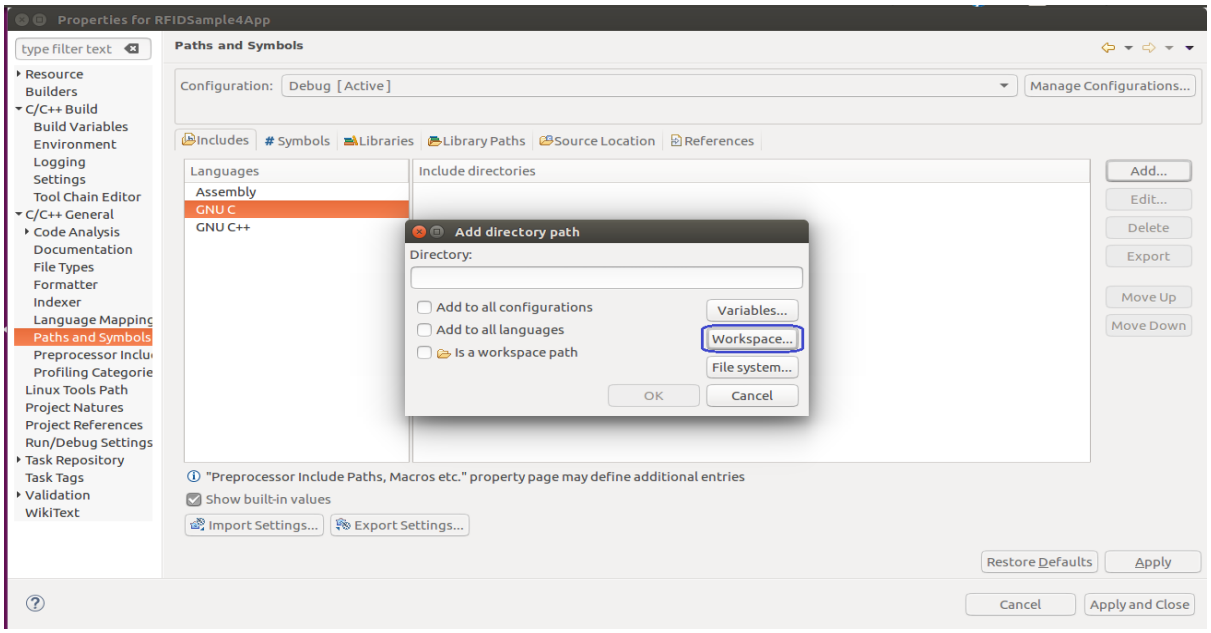
Figure 60(b): Properties for RFIDSample4App for C++ application



NOTE: For C & C++ applications, rest of the steps will be similar except Language selection and until further depicted. Hereby, continuing further description with C language selected for C application.

Click “Add” button, which will pop up the “Add directory path” Window as shown below in the figure
Click “workspace” button.

Figure 61(a): Add Directory Path-workspace

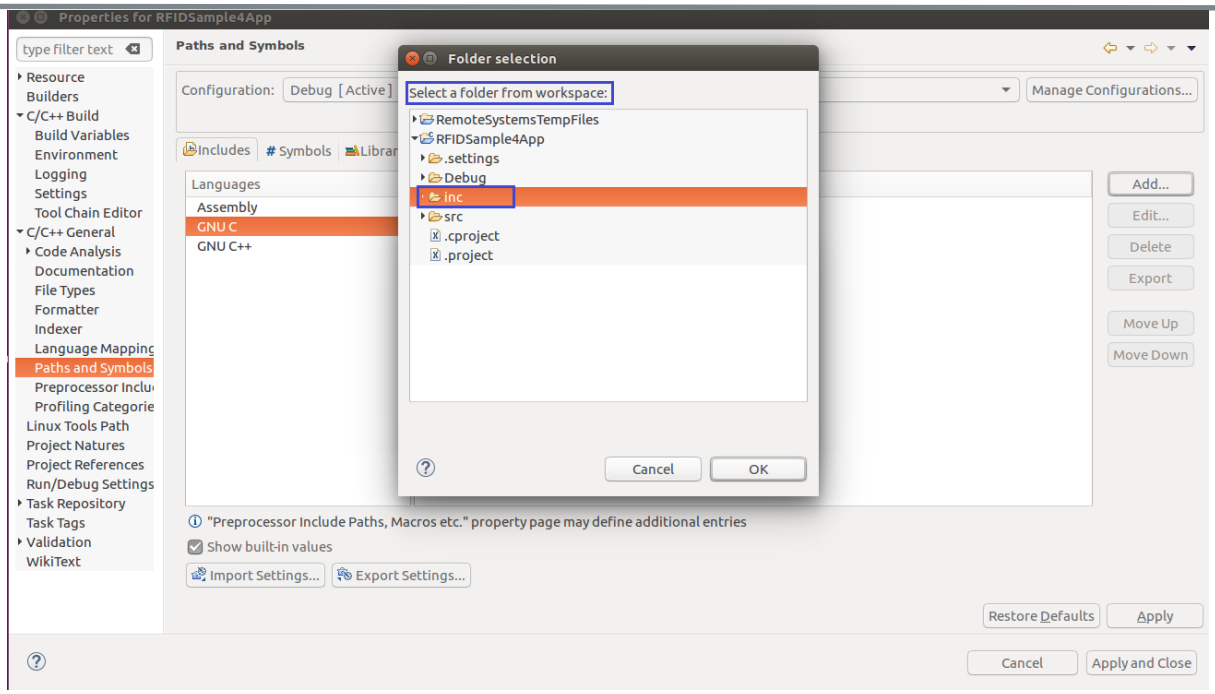


Expand “RFIDSample4App” folder,

Select “inc” folder from workspace and

Click “OK”

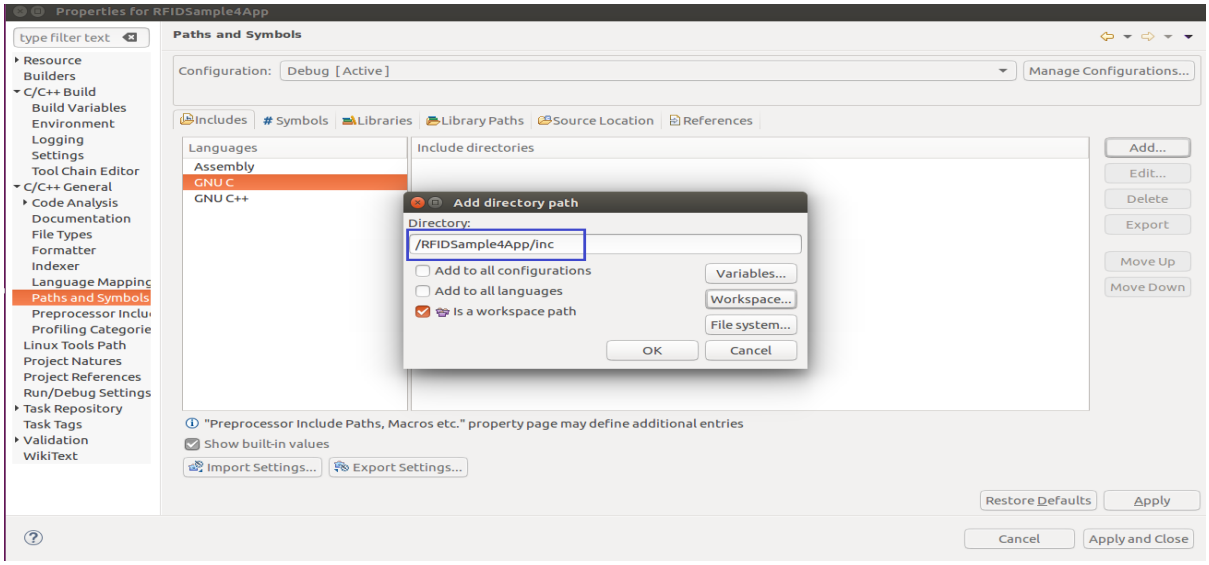
Figure 61(b): inc folder selection from workspace



Ensure the directory is taken up correctly and “is a workspace path” gets enabled

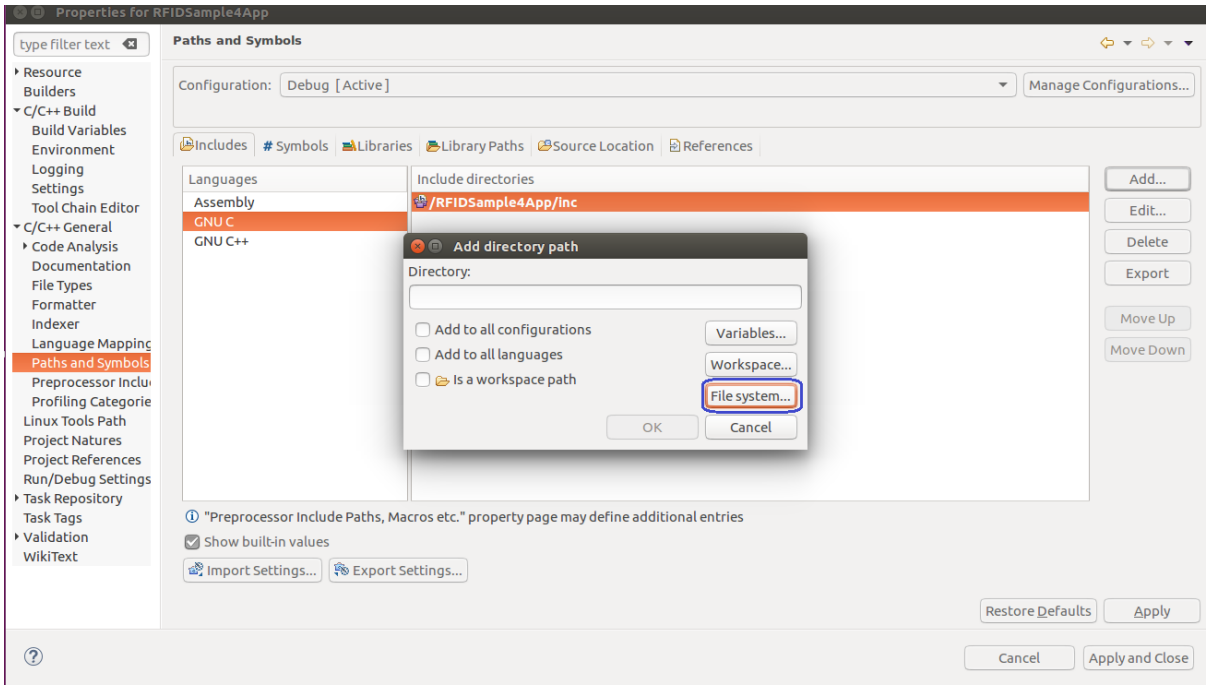
Click “OK”

Figure 61(c): Added inc folder in workspace



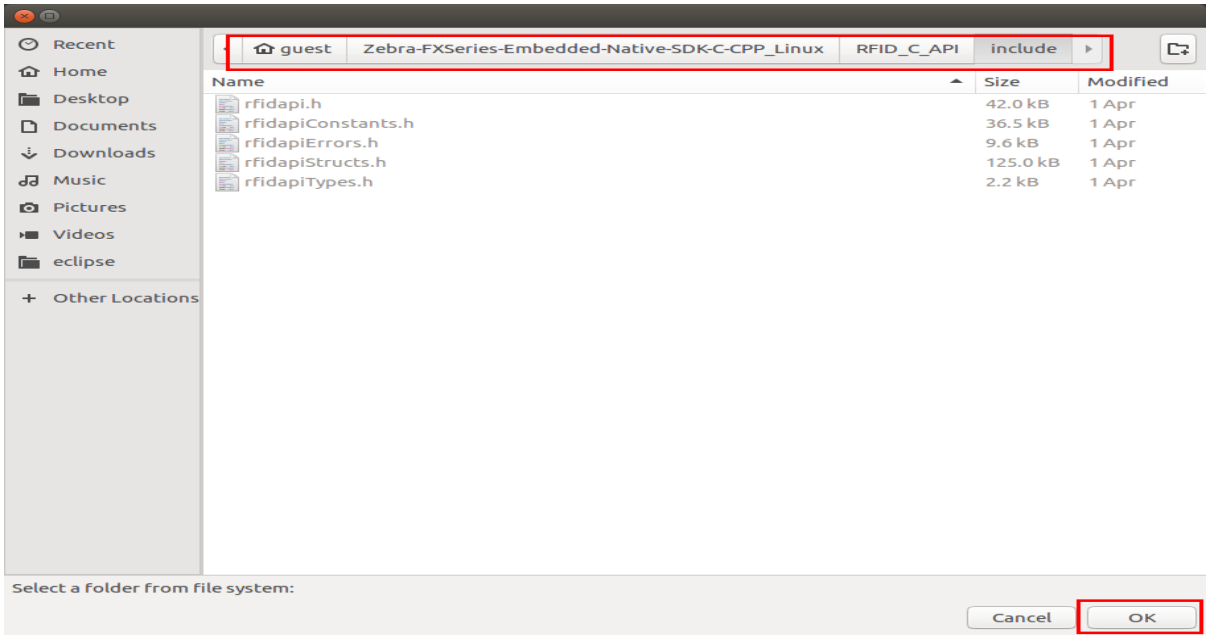
Click “Add” button, which will pop up the “Add directory path” Window as shown below in the figure Click “Filesystem” button.

Figure 61(d): Add Include Directory Path-File system



Add include path by navigating to “[Installation-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/RFID_C_API/include” directory as shown below in the figure. And click “OK” button.

Figure 62: RFID A API include path



Click “Ok” Button under Add directory path window

Figure 63: Add directory path-File system

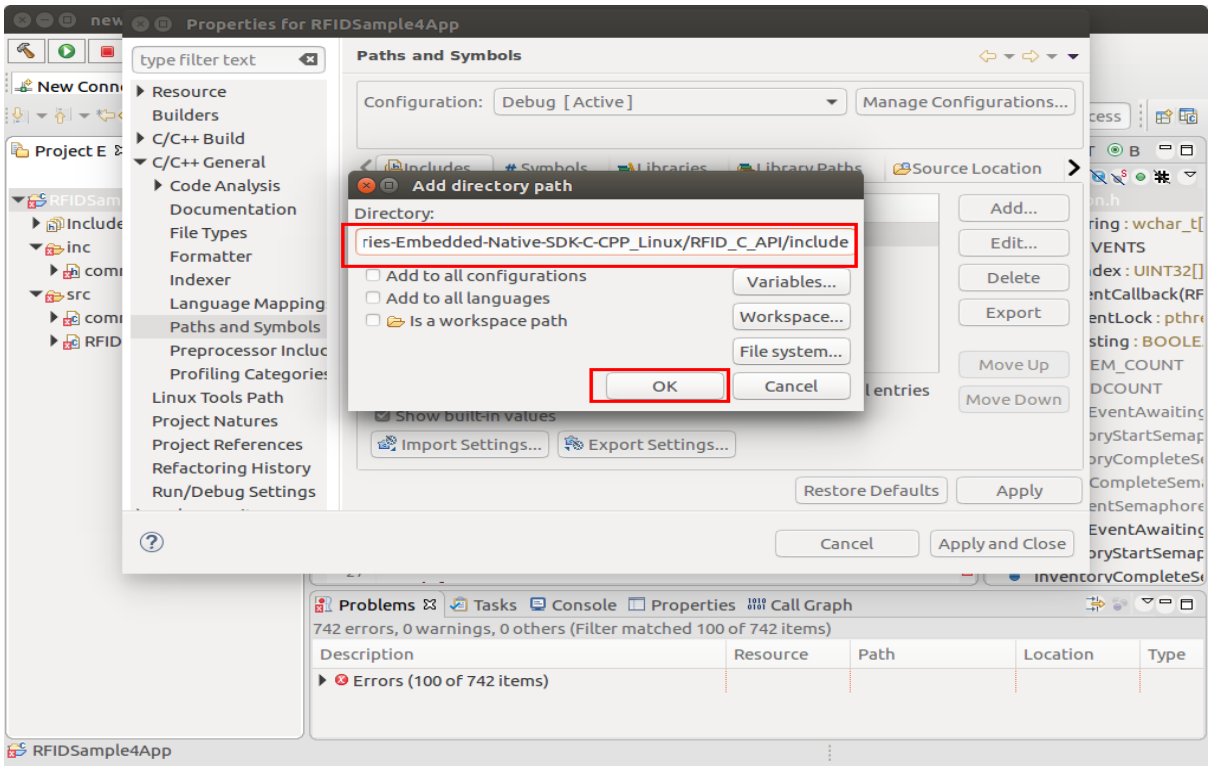


Figure 64(a): C project include path view

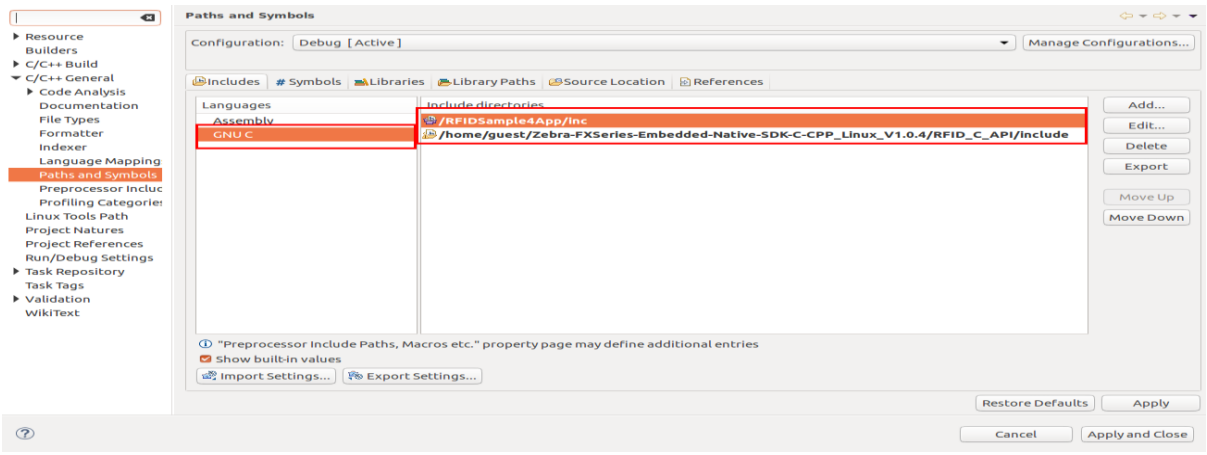
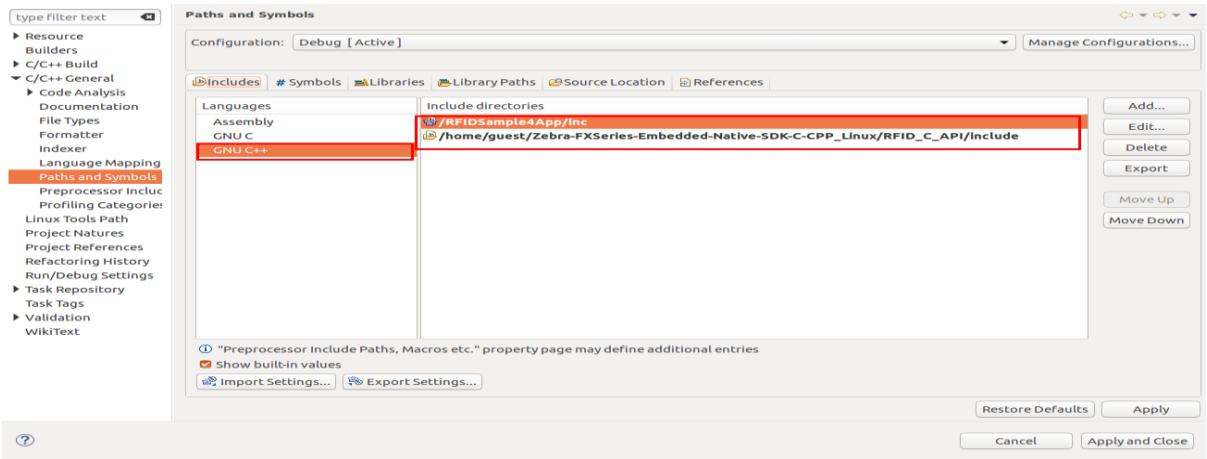


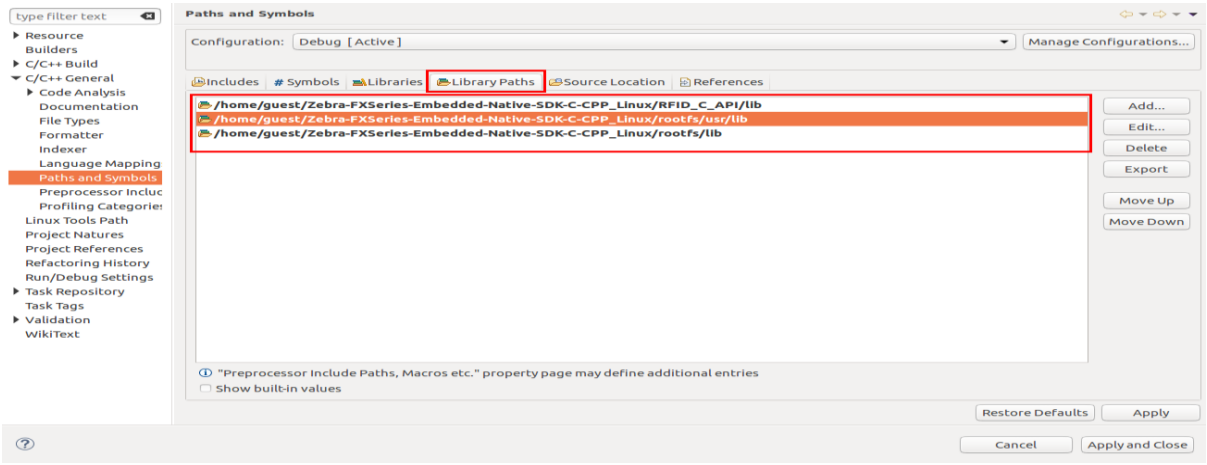
Figure 64(b): C++ project include path view



To Add “Library Paths”, click on “Library Paths” tab as shown in the figure 65.
Add following Library paths

1. “[Installation-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/RFID_C_API/lib”
2. “[Installation-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/rootfs/usr/lib”
3. “[Installation-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/rootfs/lib”

Figure 65: Add library paths in C application and as well as the same applicable to C++ application.

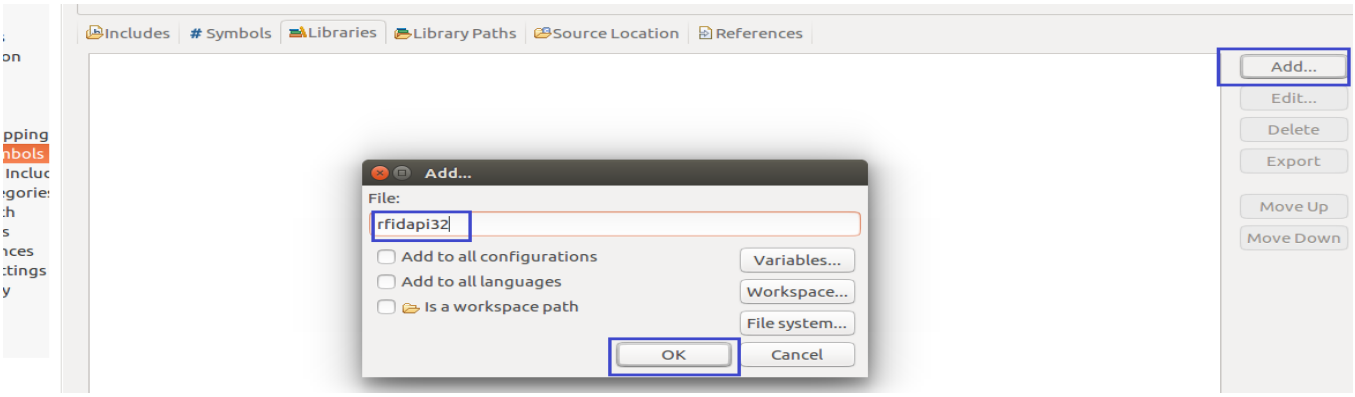


Click Libraries tab, Click on Add button,

In the Pop Up “Add” Window

Enter “rfidapi32” as the library name as shown in the figure 66 and click OK button.

Figure 66: rfidapi32 view



Likewise add other set of libraries (gnutls, nettle, curl, xml2, ssl, ltk, crypto, ssh2, z, pthread, idn, hogweed, gmp, unistring). Below are figure 67(a) and 67(b) shown after adding the list. This is applicable to both C/C++ application.

Figure 67(a): library list view 1

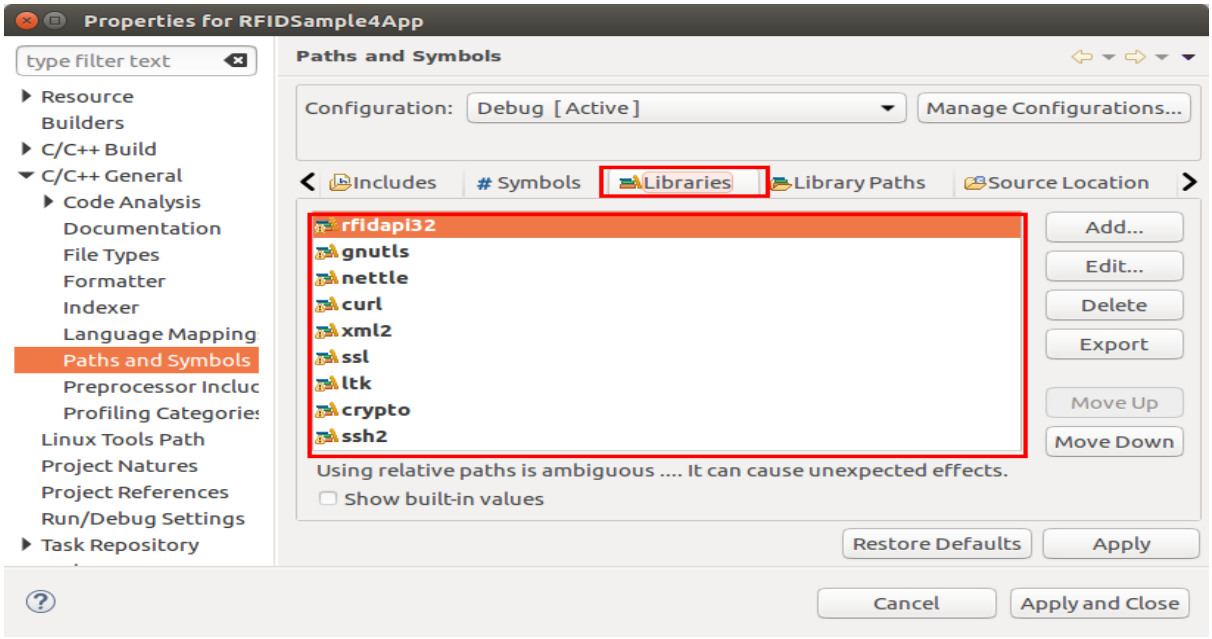
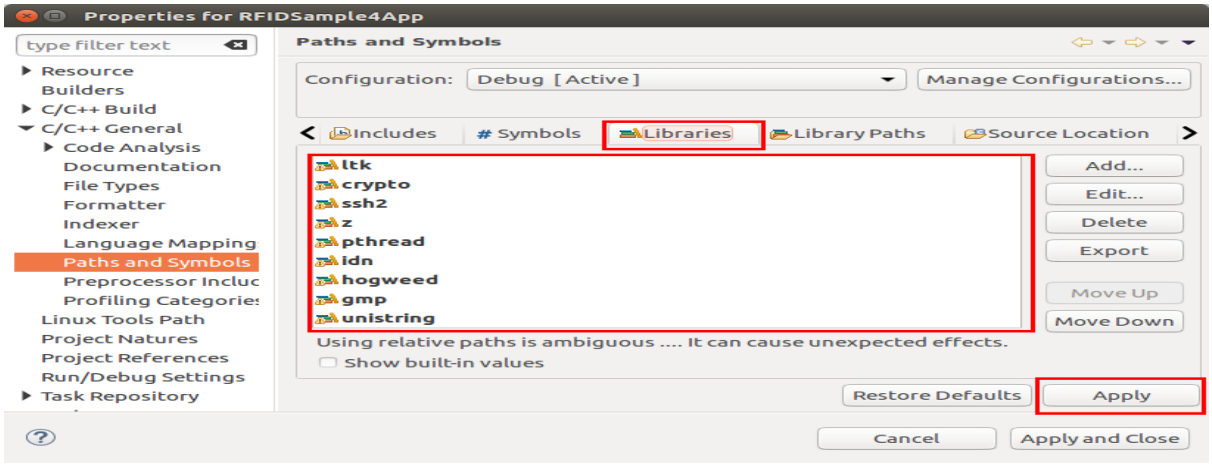


Figure 67(b): library list view 2



In Properties Window for RFIDSample4App

Expand “C/C++ Build” section

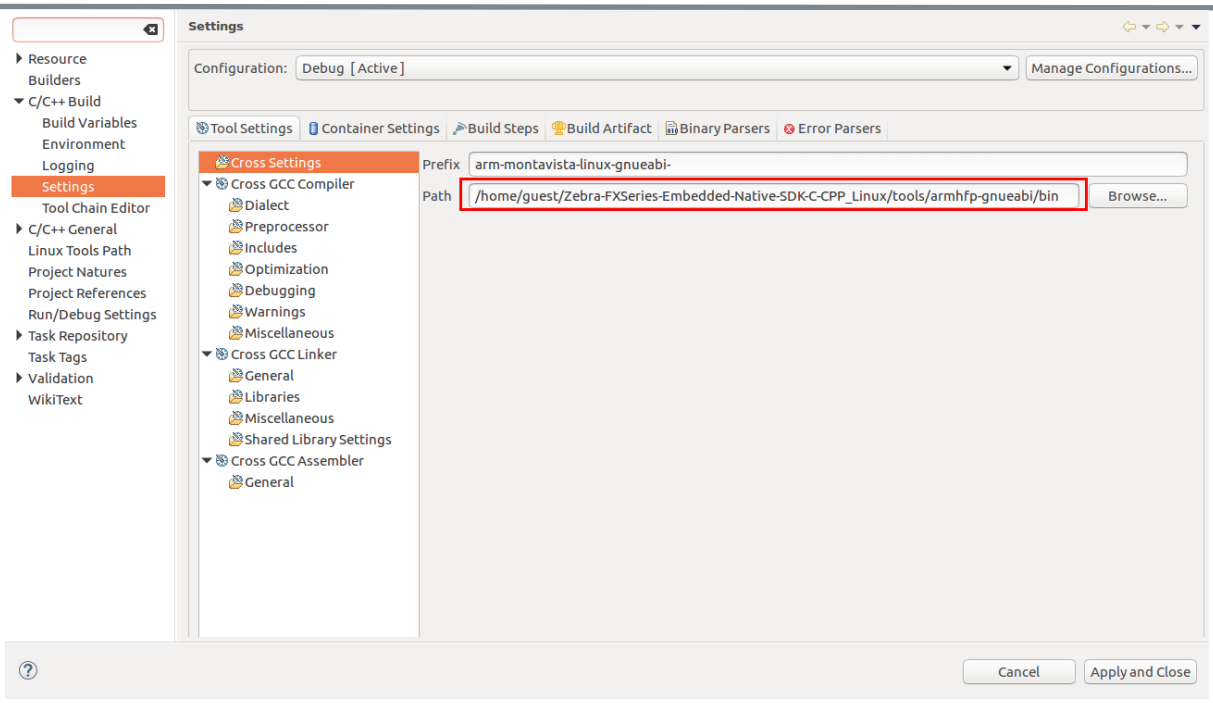
Navigate to “C/C++ Build” ->Settings->Tool Settings->Cross Settings

Click “Cross Settings” as shown in the figure below

Make sure Prefix is “arm-montavista-gnueabi-“

Make sure Toolchain path is set to “[Installation-path]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/tools/armhfp-gnueabi/bin”

Figure 68: Cross Settings



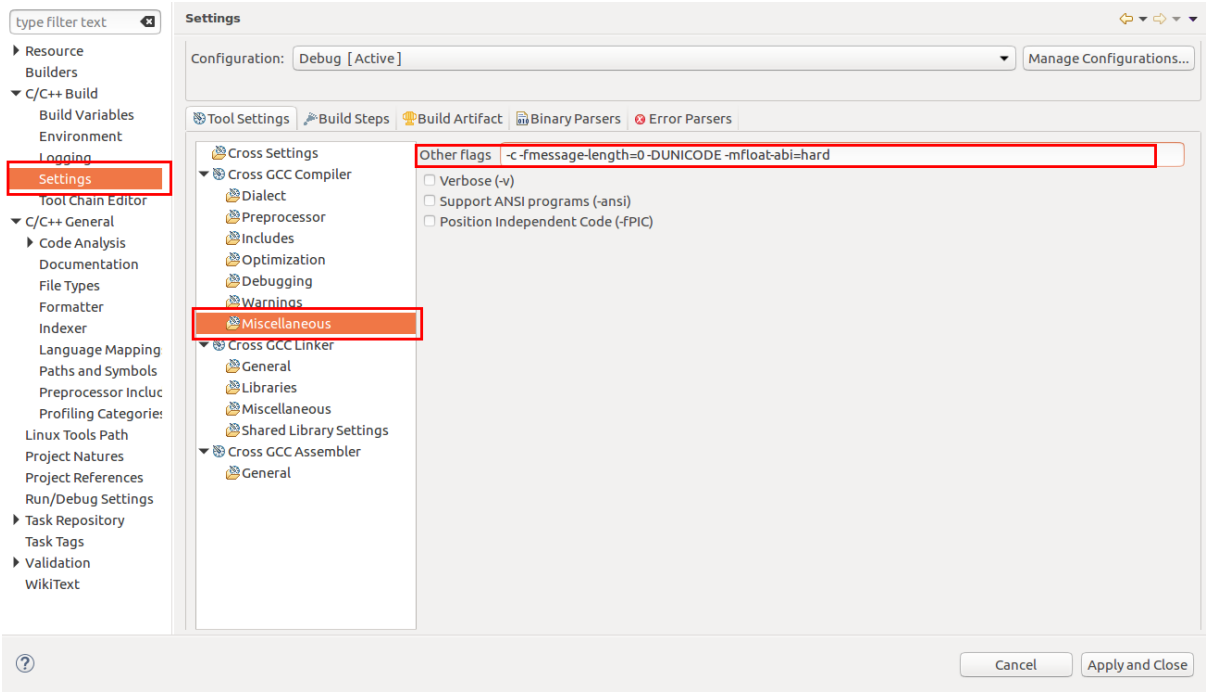
For C application, set Cross GCC Compiler flags:

In Settings->Tool Settings->Cross GCC Compiler->Miscellaneous-> Other flags**,**

enter extra flags of below as shown in Figure 69(a).

" -c -fmessage-length=0 -DUNICODE -mfloat-abi=hard"

Figure 69(a): Cross GCC Compiler flag settings



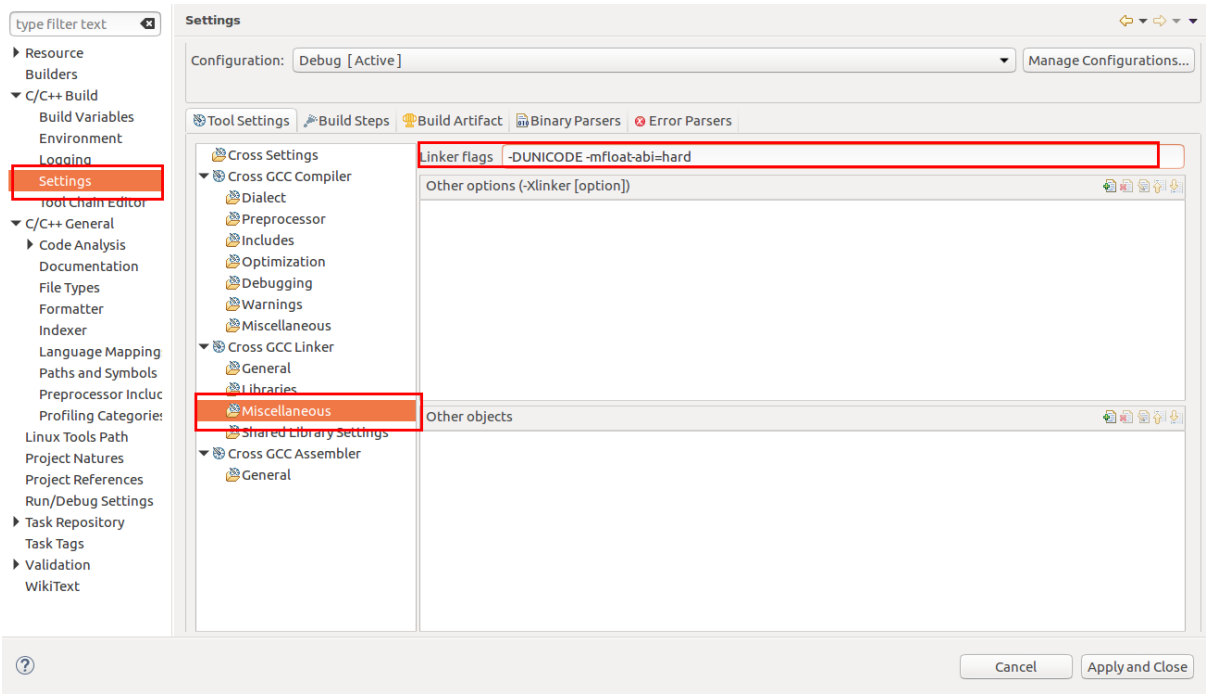
For C application, set Cross GCC Linker flags:

In Settings->Tool Settings->Cross GCC Linker->Miscellaneous-> Linker flags**,**

enter extra flags of below as shown in Figure 69(b).

"-DUNICODE -mfloat-abi=hard"

Figure 69(b): Cross GCC linker flag settings



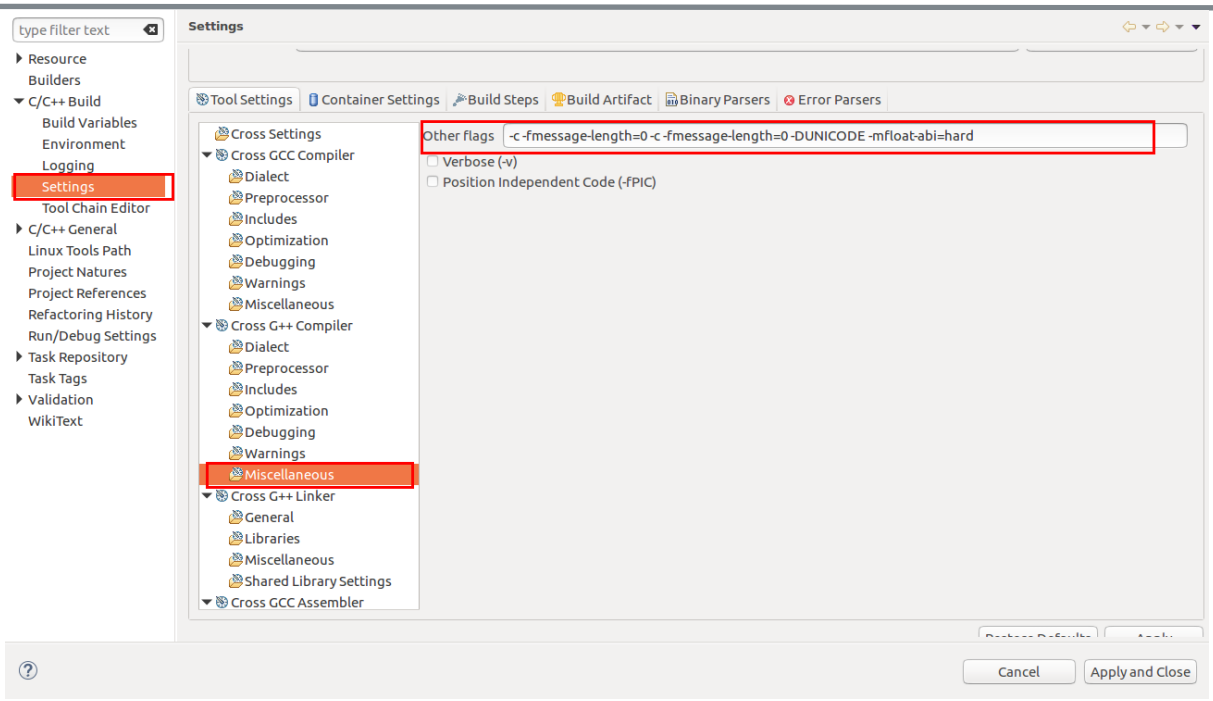
For C++ application, set Cross G++ Compiler flags:

In Settings->Tool Settings->Cross G++ Compiler->Miscellaneous-> Other flags**,**

enter extra flags of below as shown in Figure 69(c).

" -c -fmessage-length=0 -DUNICODE -mfloat-abi=hard"

Figure 69(c): Cross G++ compiler flag settings



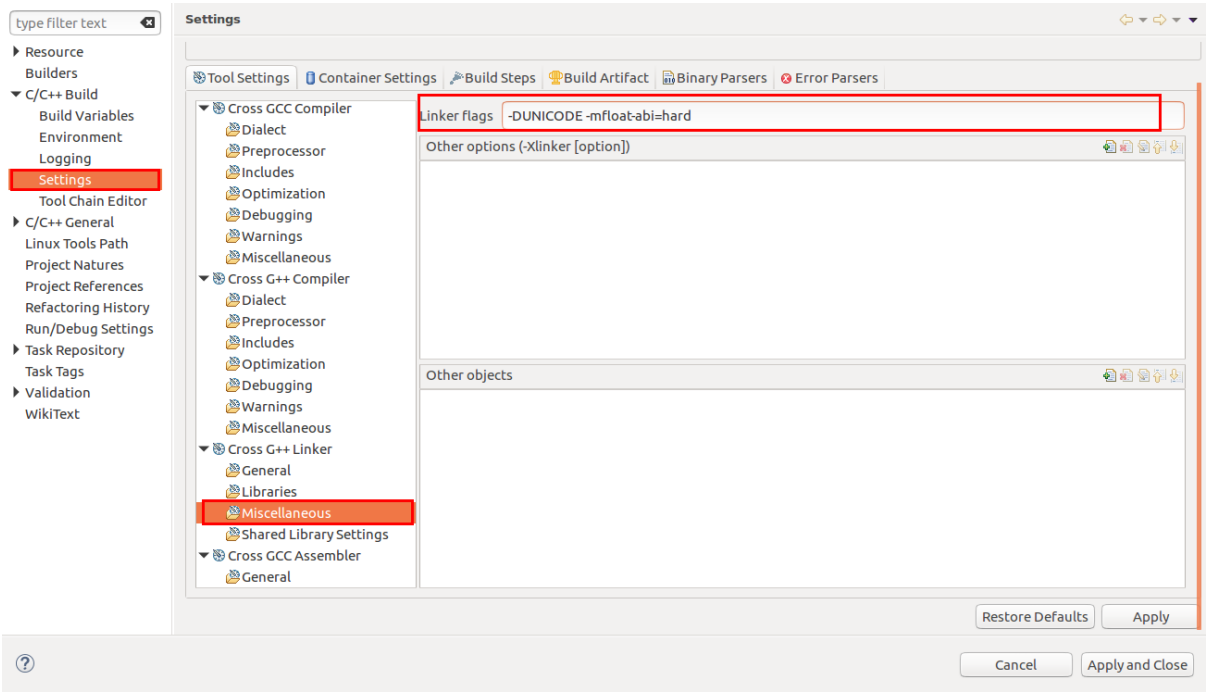
For C++ application, set Cross G++ Linker flags:

In Settings->Tool Settings->Cross G++ Linker->Miscellaneous-> Other flags**,**

enter extra cross linker flags of below as shown in Figure 69(d).

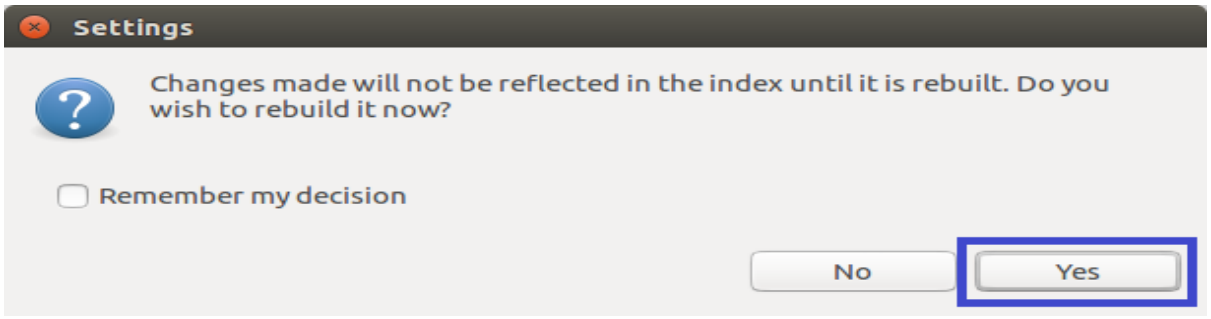
" -DUNICODE -mfloat-abi=hard"

Figure 69(d): Cross G++ linker flag settings:



You may find this pop up, click “Yes” button on the Settings window.

Figure 70: Cross GCC Linker view



7.4 Build/Debug Embedded Native RFIDSample4App C/C++ Project.

The building and debugging the C/C++ application project is applicable as mentioned in section 6.0.

8.0 Create Start and Stop Scripts of C/C++ Installation Package

Creating Start and Stop Scripts for C/C++ Installation Package

1. Copy start_sampleapp.sh and stop_sampleapp.sh from: [Embedded SDK Install folder]/Zebra-FXSeries-Embedded-Native-SDK-C-CPP_Linux/samples/sampleScripts/c_c++/ into the build directory, which is the application directory (i.e., /apps)

2. Rename the script files start_sampleapp.sh and stop_sampleapp.sh to start_appname.sh and stop_appname.sh with the executable file name (existing as “RFIDSample4App”) as appname.elf OR appname.

3. Replace line /apps/%sampleapp% & in start_appname.sh with /apps/appname.elf & or /apps/appname & (same as the executable name).

4. Replace the line EXECUTABLE_NAME=%sampleapp% in stop_appname.sh with EXECUTABLE_NAME=appname.elf or EXECUTABLE_NAME=appname (same as the executable name).

9.0 Embedded Application Installation Package Creation

9.1 Embedded application package creation

To create an FX RFID Reader Embedded Application, install package on Ubuntu 16.04 OS based host system, follow the steps

1. Create the Debian package directory structure as shown below

```
RFIDSample4App_2.0.1
├── DEBIAN
│   └── control
├── RFIDSample4App (any C/C++ executable file)
├── start_RFIDSample4App.sh
└── stop_RFIDSample4App.sh
```

Inside RFIDSample4App_2.0.1 directory there are

1 directory and 4 files

In the above directory structure, 2.0.1 is the version

“RFIDSample4App” is the build directory name. It contains one directory “DEBIAN” with single control file is explained later.

“RFIDSample4App” directory contains start and stop script along with executable file (C/C++).

2. Example details of control File

==========
Package: RFIDSample4App

Version: 2.0.1

Section: base

Priority: optional

Architecture: all

Maintainer: name <email@address.com>

Description: Basic Debian Test

==========

Create control file containing the following fields to be updated as shown above

1. Package
2. Version
3. Priority
4. Architecture
5. Maintainer
6. Description

For further details please refer https://www.debian.org/doc/debian-policy/ch-controlfields.html

3. Create Start and Stop scripts for the embedded application in [Build folder], which is the application folder as mentioned in section 8.

4. Ensure that **dpkg-deb** is installed on the host

5. Go to the parent directory of folder 'RFIDSample4App_2.0.1/'.

6 Run the below command
 # dpkg-deb --build -Zgzip RFIDSample4App_2.0.1
deb package will be created in the parent directory

7. Using web UI, install the deb package on the RFID reader

NOTE: Ensure execution permission is provided for the file, the Start and Stop script. If not, use the chmod +x command to change permission of files.

NOTE: The name of the package and name of the application are the same.

NOTE: Package, Version, and Maintainer are mandatory. There are many optional fields in the control file.

9.2 Installation and Removal of application package On RFID reader using UI

Below listed steps will help in installing and uninstalling application package on RFID reader

Once login to web console of RFID reader

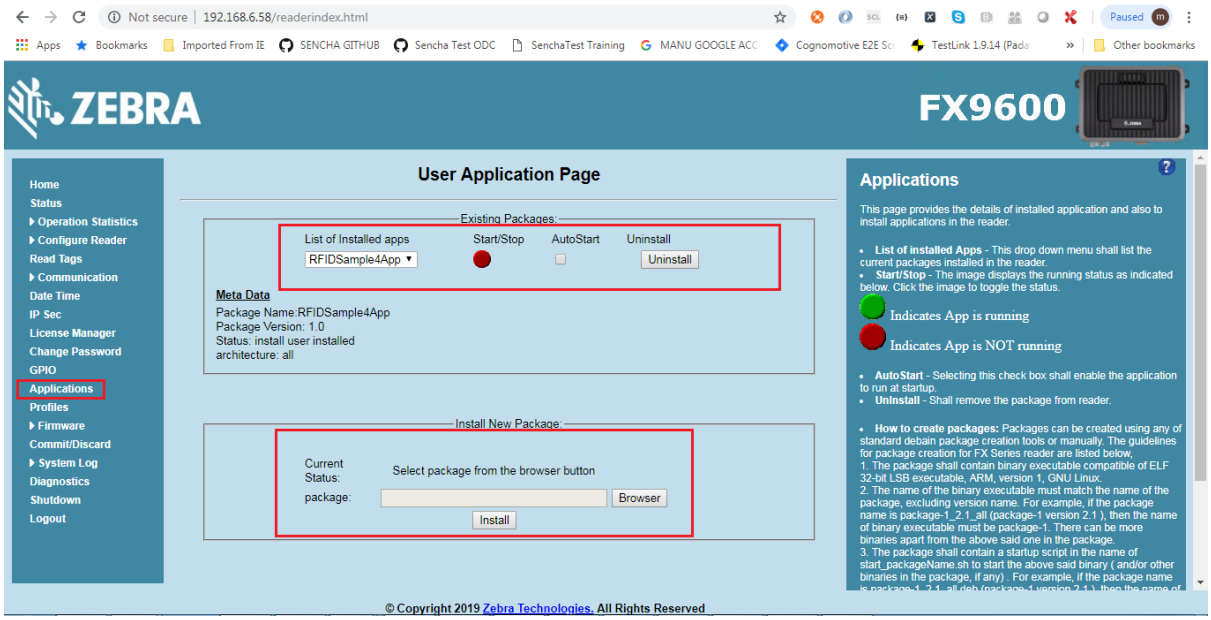
Step 1: Application -> Install New Package -> Browse (Select the created deb package).

Step 2: Application package should be listed in List of installed apps as shown in figure below.

Step 3: Click on Start or Stop circular button to Start or Stop the application executable.

Step 4: To Uninstall or remove application package, click on uninstall button in the figure below.

Figure 71: User Application Page



Login to Reader through remote terminal/console as user 'rfidadm' and execute RFID Sample application as '/apps/RFIDSample4App. Note: The using of start and stop circular button/auto start is not applied to RFID sample C/C++ application since it will be executed as background process, which requries user inputs from console.