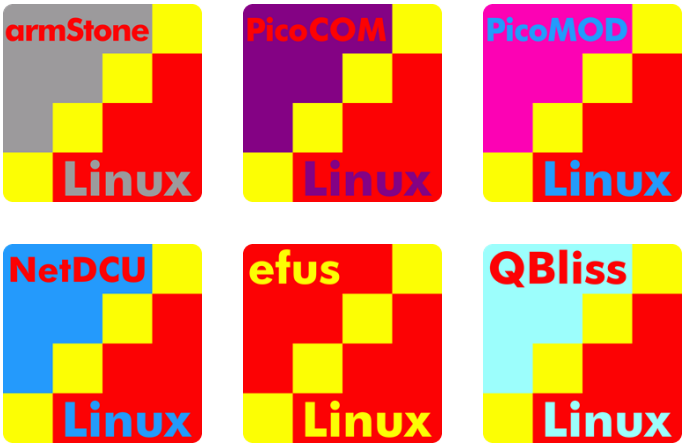
F&S Introduction to Eclipse

Debugging an Application

Version 1.3
(15.09.2022)



Untere Waldplätze 23
D-70569 Stuttgart
Germany

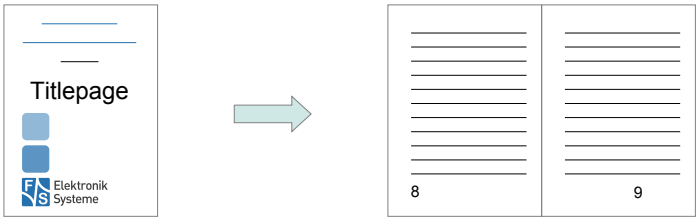
Phone: +49(0)711-123722-0
Fax: +49(0)711-123722-99

About This Document

This document describes how to debugging remote device using Eclipse CDT Automatic Remote Debugging Launcher under Linux.

Remark

The version number on the title page of this document is the version of the document. It is not related to the version number of any software release. The latest version of this



document can always be found at http://www.fs-net.de.

How To Print This Document

This document is designed to be printed double-sided (front and back) on A4 paper. If you want to read it with a PDF reader program, you should use a two-page layout where the title page is an extra single page. The settings are correct if the page numbers are at the outside of the pages, even pages on the left and odd pages on the right side. If it is reversed, then the title page is handled wrongly and is part of the first double-page instead of a single page.

Typographical Conventions

We use different fonts and highlighting to emphasize the context of special terms:

`File names`

Menu entries

`Board input/output`

`Program code`

`PC input/output`

`Listings`

`Generic input/output`

`Variables`

History

Date	V	Platform	A,M,R	Chapter	Description	Au
2015-10-26	1.0	*	M	Layout	Converted to new F&S document layout, added and updated pictures	AD
2018-04-06	1.1		M	ALL	Modify the documentation, add more picture and explain more in details	PJ
2022-03-18	1.2		A,M	1.1	Modified section 1.1, add 1.2 for manual installation	AD
2022-08-19	1.3		M	4	Modified section 4, Yocto Debugger.	TG

V Version
A,M,R Added, Modified, Removed
Au Author

Table of Contents

1 Introduction

F&S offers a whole variety of Systems on Module (SOM) and Single Board Computers (SBC). There are different board families that are named NetDCU, PicoMOD, PicoCOM, armStone, QBliss, efus and PicoCore.

Linux is available for all of these platforms. Linux offers the integrated development environment (IDE) Eclipse for application development. Development environments include Eclipse CDT for C/C++.

This document describes how to debug an application on a remote device using the automatic remote debugging launcher plug-in.

1.1 Requirements

To develop an application following are necessary:

- Linux PC or VPC with fedora distribution
- Cross tool chain
- Buildroot or Yocto BSP

For successful quick start we offer listed software tools on our homepage in download area. For more information see documents “Quickstart with F&S Development Machine”, “First Steps” and “Linux on F&S Boards”

Note:

On F&S development machine needed packages are preinstalled or can be installed easily. Cross tool chain and BSP are available too.

1.2 Yocto Tool Chain

The cross tool chain in case of yocto BSP is not a part of F&S development machine. But it can be created by building SDK. For this yocto images must be available. See section 10.2 of “Linux on F&S Board” for more details.

First add additional packages for debugging to your local configuration. Append to the configuration file conf/local.conf following lines:

EXTRA_IMAGE_FEATURES ?= "debug-tweaks tools-debug eclipse-debug ssh-server-openssh"

CORE_IMAGE_EXTRA_INSTALL += "openssh-sftp openssh-sftp-server"

Second build the SDK by command

```
bitbake -c populate_sdk <image name>
```

After building cross tool chain can be found in “<buld-dir>/tmp/deploy/sdk” sub directory with name <Distro>-glibc-x86_64-<image-name>-aarch64-<machine>-toolchain-5.4-zeus.sh

E.g, after building our standard images the tool chain is named “fus-imx-wayland-glibc-x86_64-fus-image-std-aarch64-fsimx8mp-toolchain-5.4-zeus.sh”

To install the toolchain to directory of your choice execute the script.

1.3 Install Eclipse on Fedora

Install Eclipse on your development host with your Linux Distribution specific package manager system. On Fedora up to version 23:

```
sudo yum install eclipse-cdt
```

On new Fedora versions like 27 or 30 package manager dnf can be used to install new packages:

```
sudo dnf install eclipse-cdt
```

Additional following packages can be helpful for application development too:

- eclipse-rse – Eclipse Remote System Explorer
- eclipse-linuxtools – Linux specific Eclipse plugins

Then start Eclipse:

```
eclipse &
```

Note:

If Yocto toolchain is using eclipse must be started from the same terminal where SDK environment are set by command:

source <toolchain directory>/environment-setup-aarch64-poky-linux

Alternative create a shell script to start both command sequentially.

E.g.: eclipse-yocto.sh

```
#!/usr/bin/sh

source <path to installed yocto toolchain>/environment-setup-aarch64-poky-linux

<path to eclipse>/eclipse
```

1.4 Install Eclipse from website

Download Eclipse IDE for C/C++ Developers package from website https://www.eclipse.org/downloads/packages/ and install it in directory of your choice.

E.g., current version is IDE 2022-03 R and package name is eclipse-java-2022-03-R-linux-gtk-x86_64.tar.gz.
For installation of additional packages “Install New Software…” dialog can be used. To start it go the menu entry “Help”.

2 Create an Application

After all dependencies are installed, you can create a new project. From *File* select *New* and then C *Project*. Then click on *Next*.

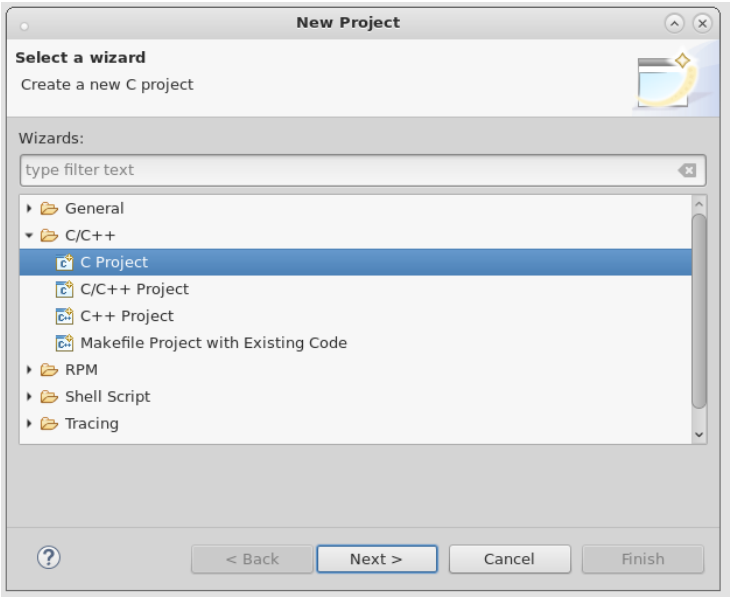


Figure 1: Create a new C project

On the next dialog fill in a project name and select *Empty Project* from *Executable*. Select *Cross GCC* as toolchain.

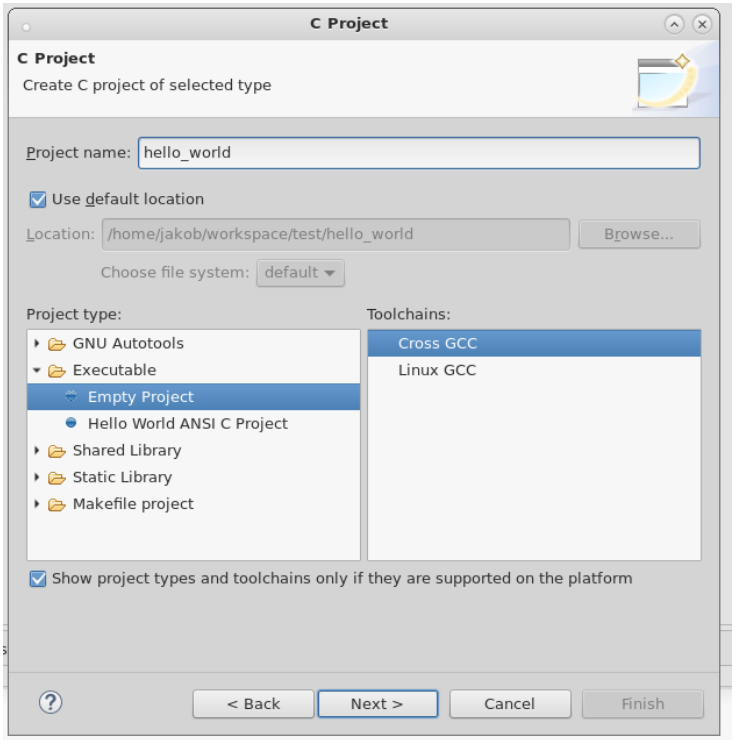


Figure 2: Create empty C project

Then click on *Next* and select platforms and configurations you wish to deploy on.

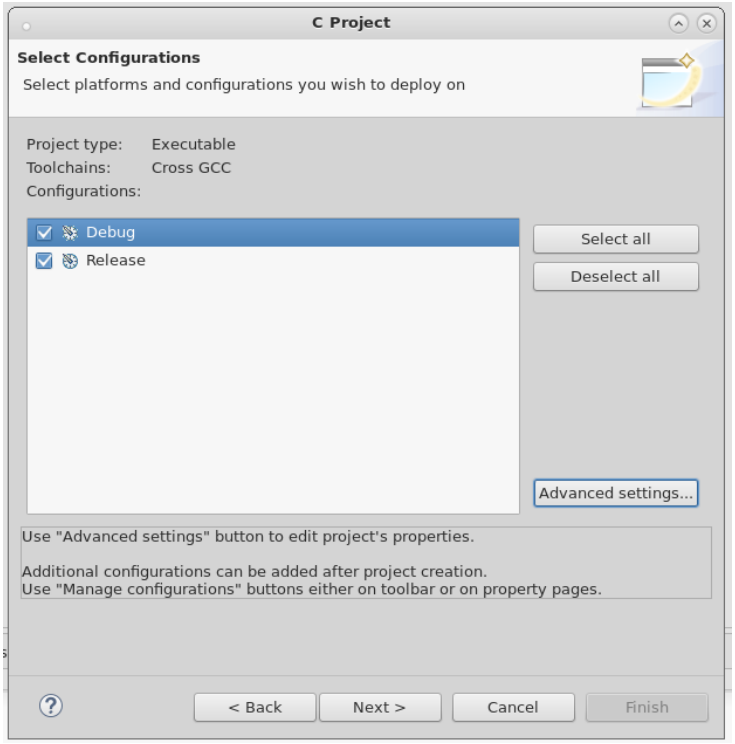


Figure 3: Select configurations

Click on *Next* and setup the installed toolchain.

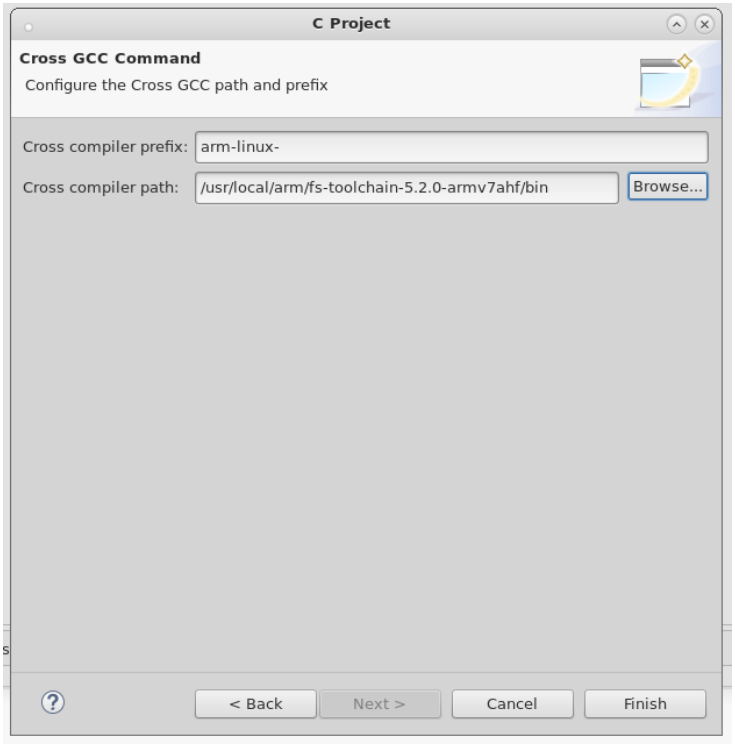


Figure 4: Configure Cross GCC

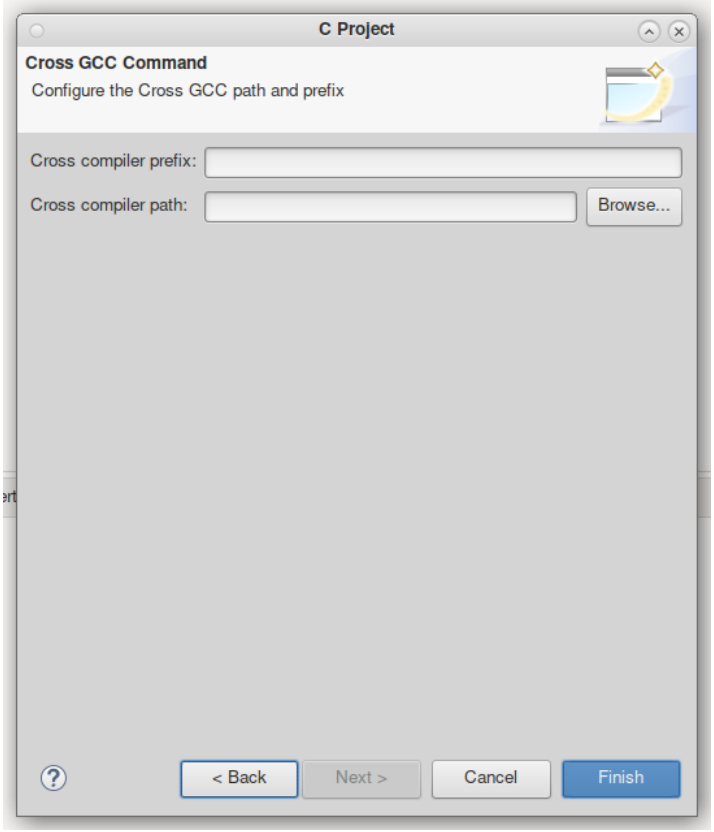


Figure 5: Configure Cross GCC - Yocto Toolchain

Click on *Finish* and you have successfully created your project.

For Yocto toolchain modify following setting too:

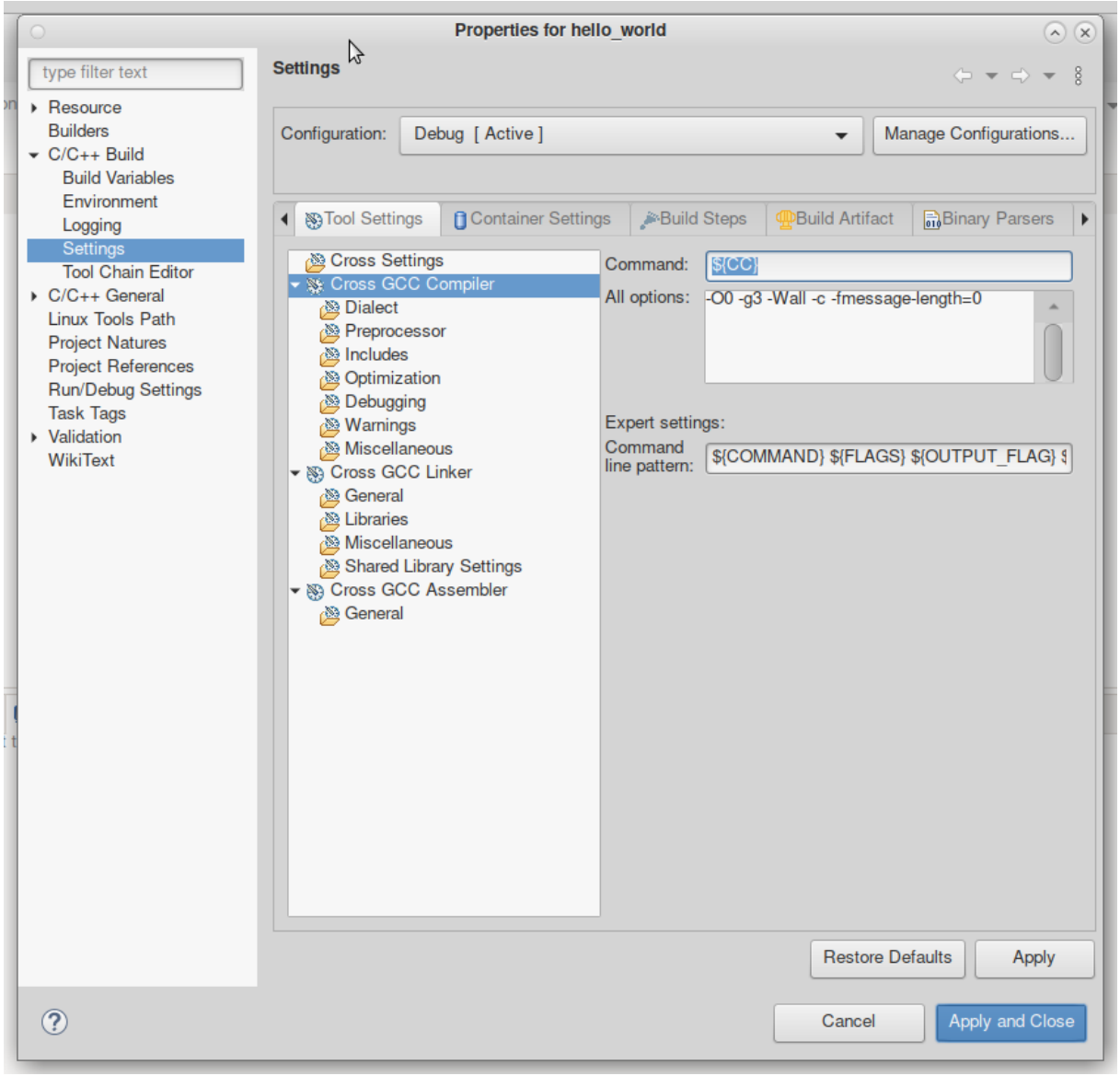


Figure 6: Cross GCC Compiler - Yocto Toolchain

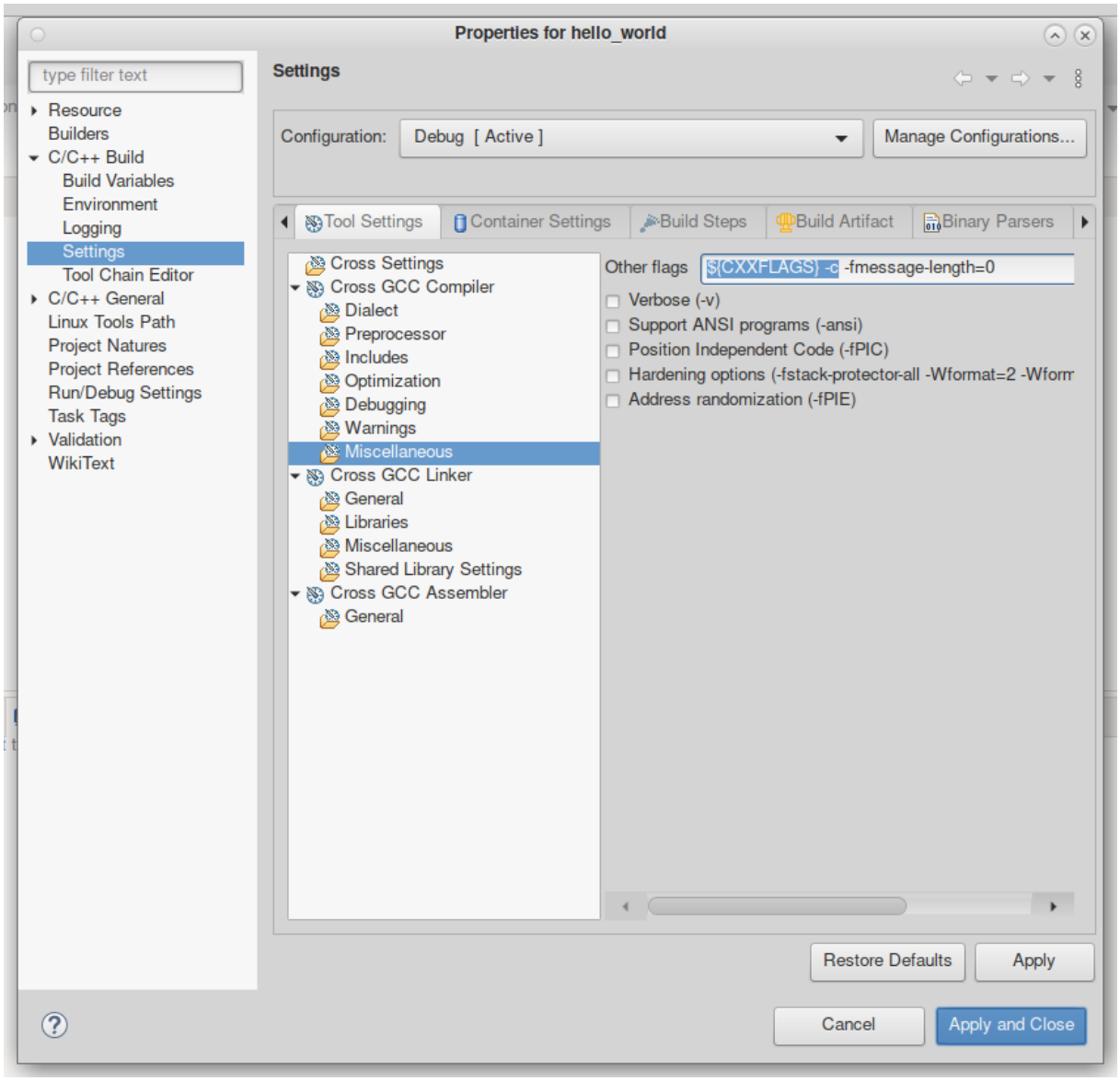


Figure 7: Cross GCC Compiler Flags - Yocto Toolchain

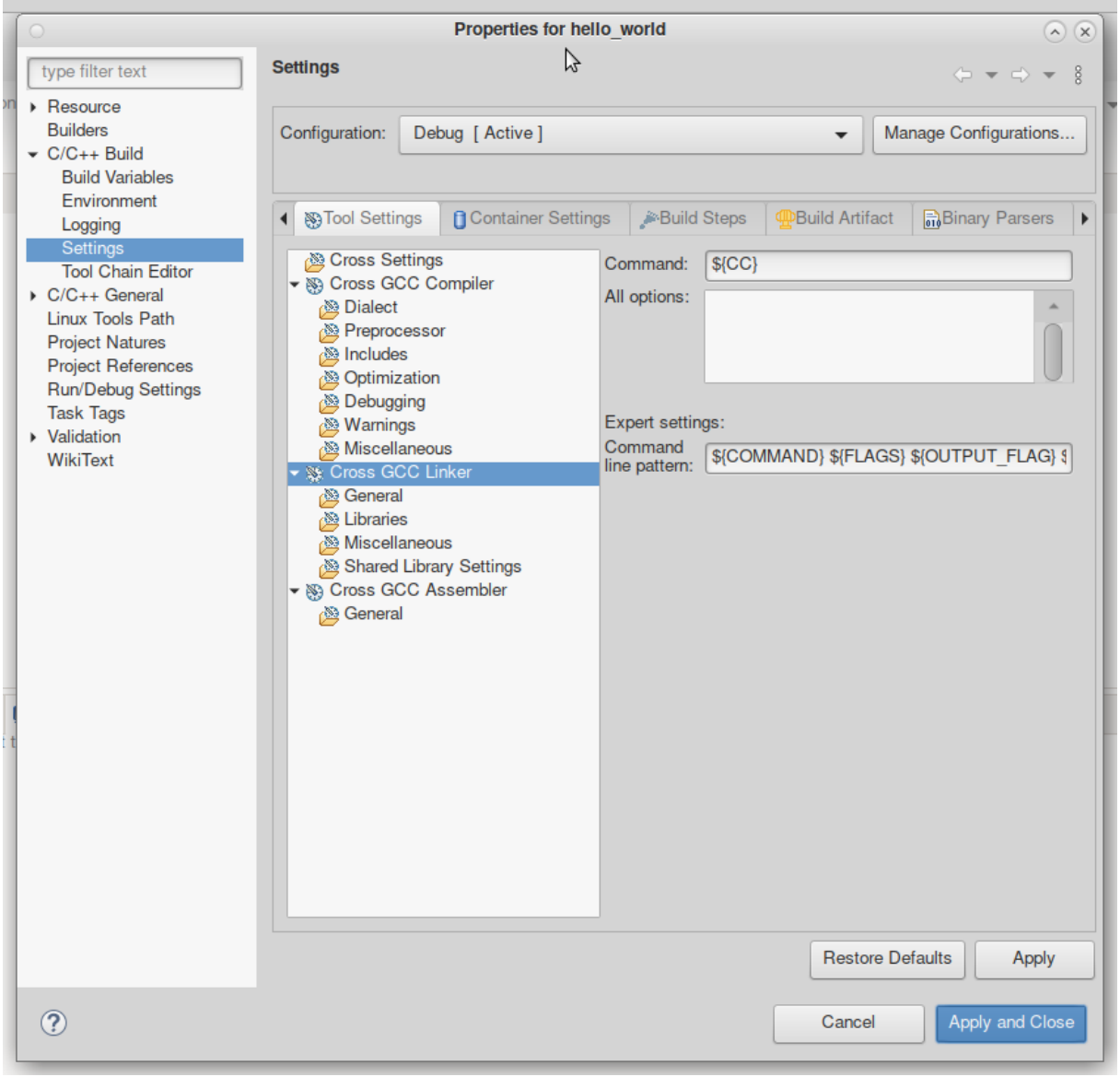


Figure 8: Cross GCC Linker - Yocto Toolchain

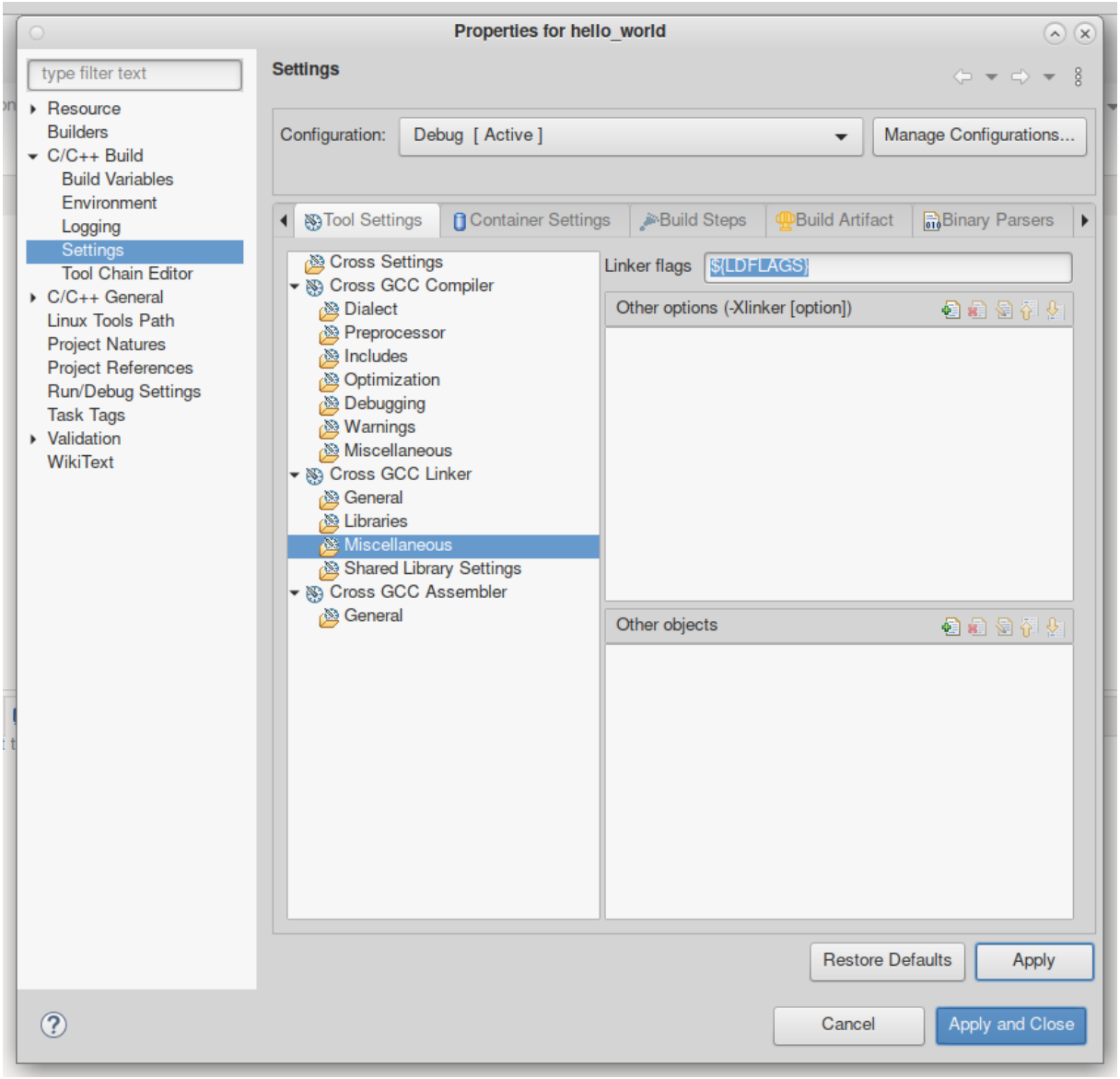


Figure 9: Cross GCC Linker Flags - Yocto Toolchain

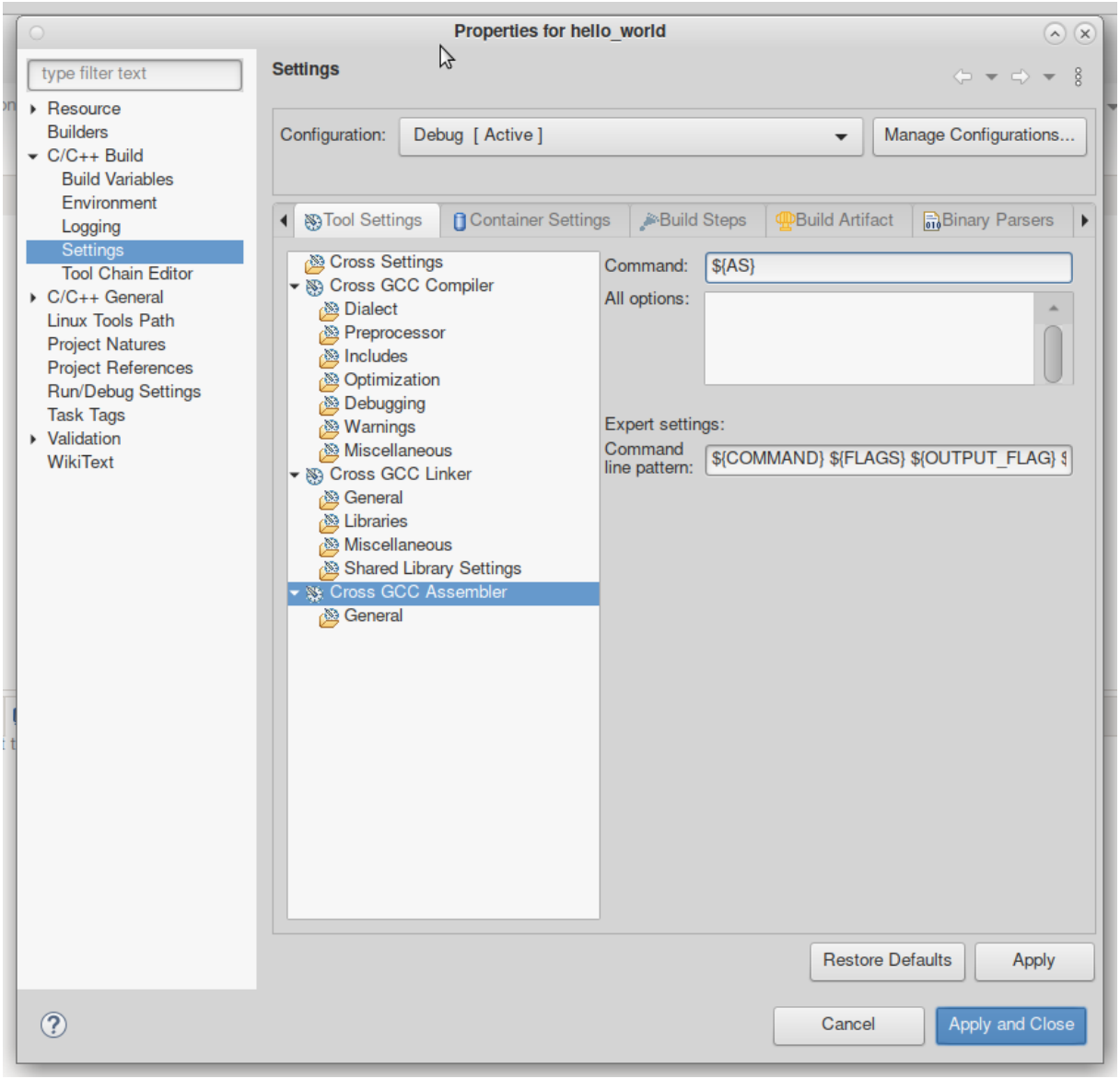


Figure 10: Cross GCC Assembler - Yocto Toolchain

2.1 Create hello c file

After you have successfully created your project right-click your project folder, choose *new* and select *source file*.

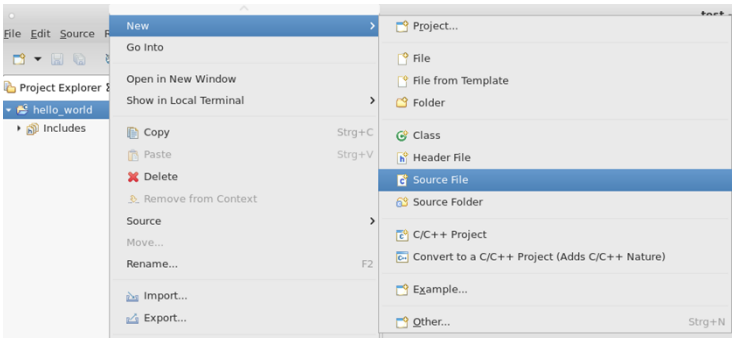


Figure 11: Create Source File

Select a name for your c file and select *finish*.

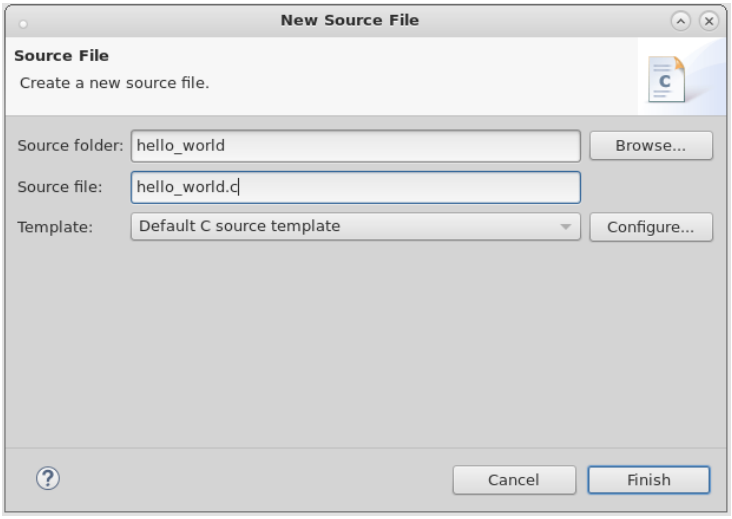


Figure 12: New Source File

Write down your example code.

```
#include <stdio.h>


int main(void)
```

```
{
      int i, val = 0;
      printf("Hello World!\n");

      for(i = 0; i < 4; i++)
      {
            val = 2*i;
            printf("value: %d\n", val);
      }
      return 0;
}
```

3 Remote Connection

3.1 Setup SSH connection on SBC/SOM

First of all you have to setup your F&S Board. Therefore you can have a look into F&S *FSiMX8M(M,N,P)/FSiMX6/SX/UL_FirstSteps_eng.pdf*. After that boot your F&S Board.

To work with SSH the board should have a valid date. This is necessary to create certificates for SSH. To setup a date you can use the following command:

```
date "2018-04-09 08:20"
```

Afterwards we have to enable the network interface. You can also set the network on command in UBoot to enable network interface at each boot. For further information please take a look in *FSiMX6/ FSiMX8M(M,N,P)_FirstSteps_eng.pdf*.

Dynamically:

```
udhcpc
```

Static:

```
ifconfig eth0 10.0.0.84 up
```

The Root-Filesystem is read-only mounted but we have to modify something in the filesystem so we need it read-writeable.

```
mount -o remount,rw /
```

Open *sshd*_config file

```
vi /etc/ssh/sshd_config
```

and edit the following lines:

```
…
Port 22
AdressFamily any
ListenAddress 0.0.0.0
PermitRootLogin yes
…
```

Optional you can also allow to login without password, but we recommend you to not do this because it's a security risk. If you want to do it anyway add the following line to *sshd_config* file.

```
…
PermitEmptyPasswords yes
…
```

After that you have to start the SSH server.

For Buildroot:

```
/etc/init.d/S50sshd start
```

For Yocto:

```
systemctl restart sshd.socket
```

Now the SSH server is running on our SBC/SOM. Now we can test it, therefore we are going back to our VM. To connect via ssh we open a Terminal, after that we are sending the following command:

```
ssh root@10.0.0.84
```

```
[jakob@localhost ~]$ ssh root@10.0.0.84
The authenticity of host '10.0.0.84 (10.0.0.84)' can't be established.
ECDSA key fingerprint is b1:b1:aa:83:12:d1:f1:21:7b:e3:6a:61:89:6e:31:ea.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.84' (ECDSA) to the list of known hosts.
```

Figure 13: SSH connection terminal

Now the SSH connection is successfully established and tested.

3.2 Setup SSH connection in Eclipse

Open Remote System Explorer. In the top right corner click *open Perspective*  and select *Remote System Explorer*.

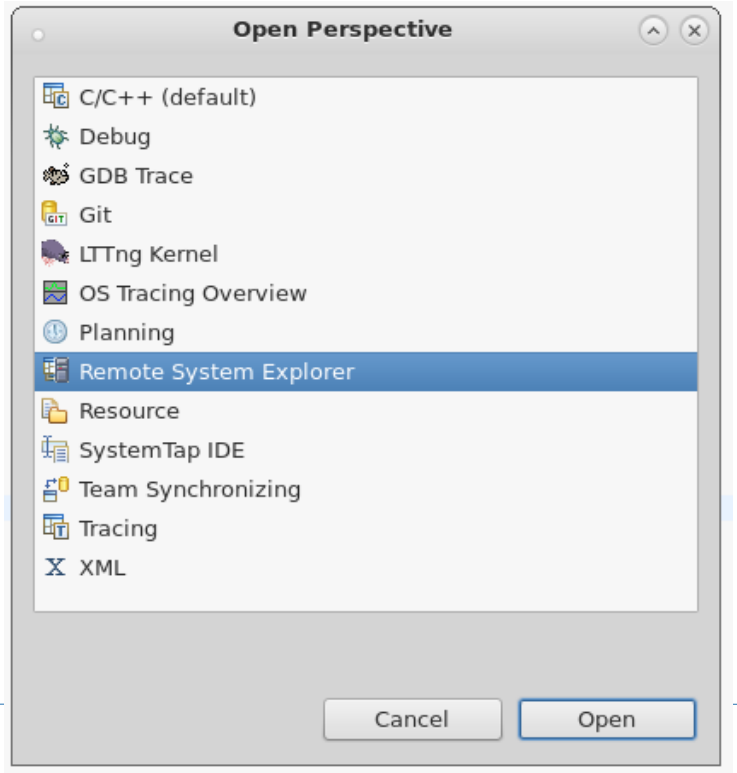


Figure 14: Open Perspective

Now we define a remote connection. Click in the left area, on the toolbar *Define a connection to remote system.* Select *SSH Only* and click *Next >*.

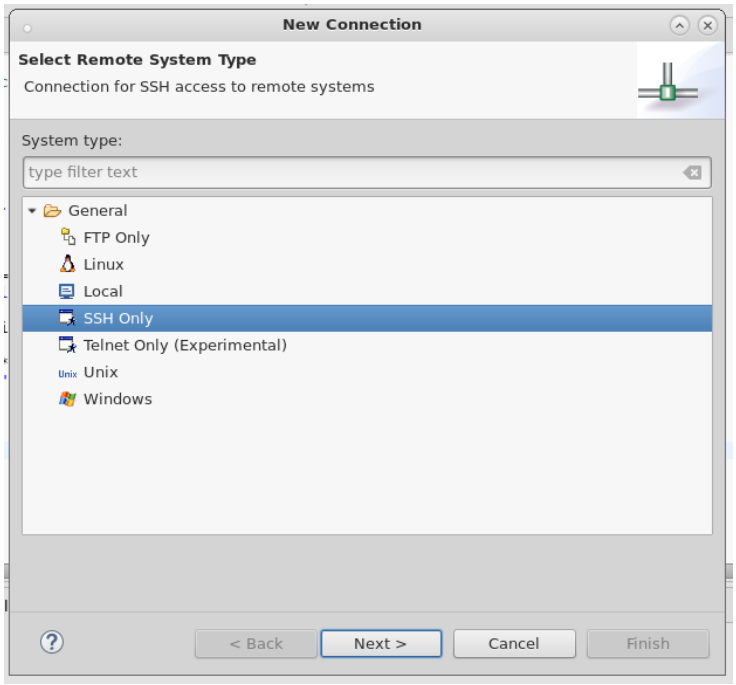


Figure 15: New Connection – Remote Type

Enter the connection configuration of your device and select *Finish*.

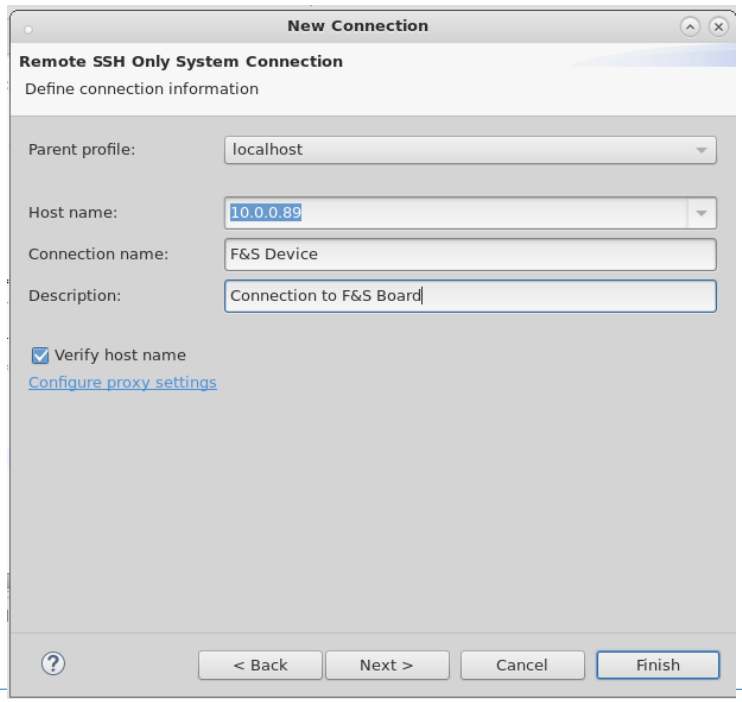


Figure 16: SSH Connection settings

It's better to setup a static IP address for the SBC/SOM so you don't have to change the ssh connection settings after every dhcp.

Test ssh connection, right-click your created remote connection and select *Connect*.

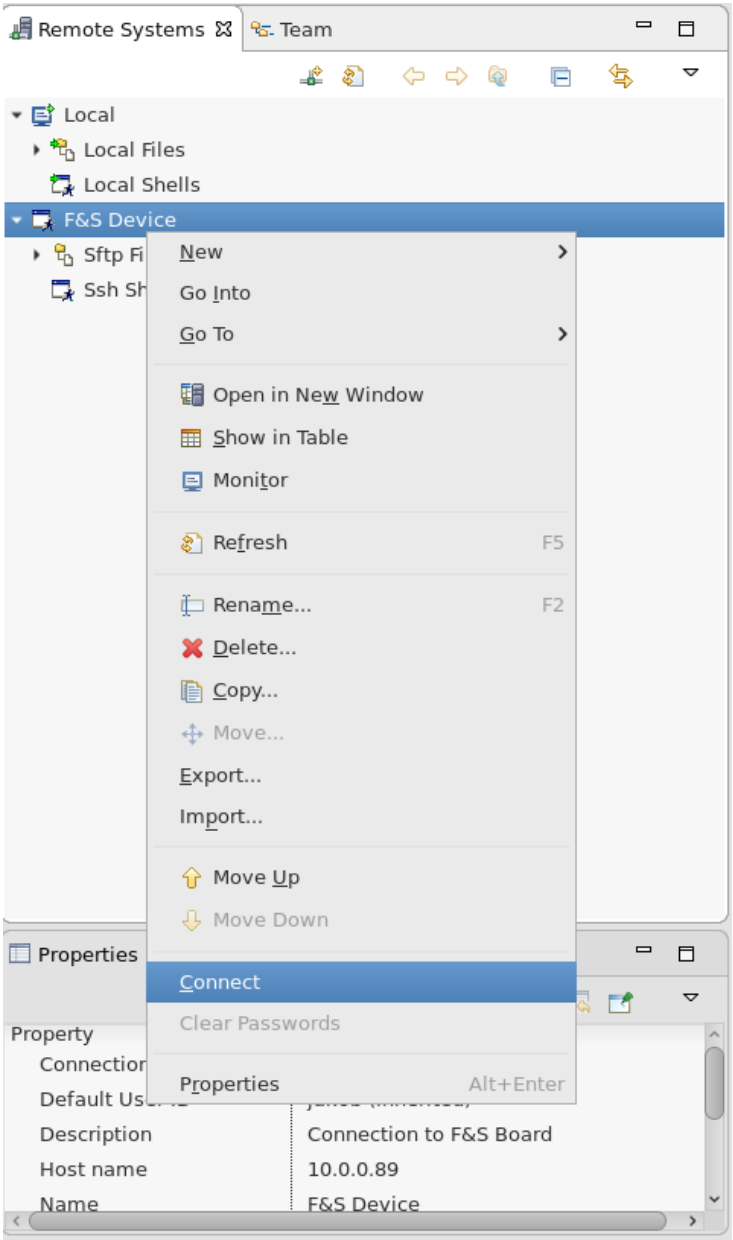


Figure 17: Connect to device

Type in your login details and select *OK*.

Figure 18: Login details

Use the Remote Systems tab to navigate through the target filesystem:

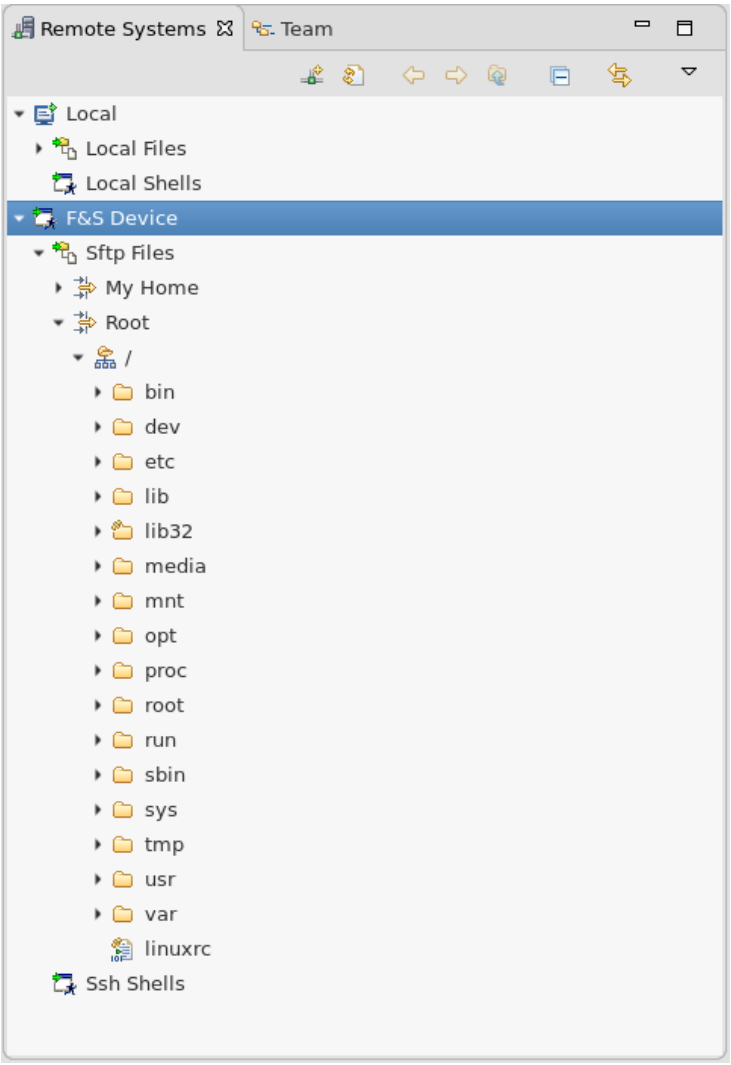


Figure 19: Target filesystem

4 Setup Application settings

Before you can setup the application settings be sure that you have built your Root-Filesystem (buildroot) or your meta-toolchain (Yocto) once. This is necessary because we need the compiled GDB debugger from this directory.

After the remote connection have been setup, switch back to the C/C++ perspective  .

If you can't switch back to the C/C++ perspective do the following: In the Menu-Bar select Window -> Perspective -> Open Perspecitve -> Other… ->C/C++

Select on the menubar *Run* and choose *Debug Configurations*…. After that select *C/C++ Remote Application* and click *New launch configuration*. 

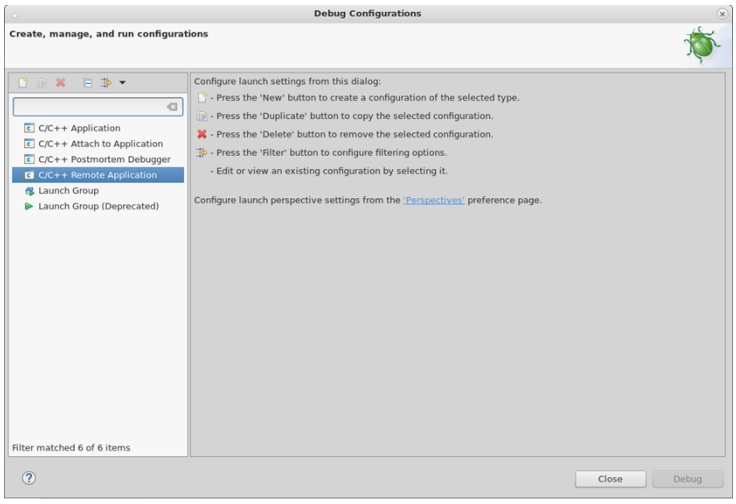


Figure 20: New Debug Configuration

Setup debug configuration and apply it.

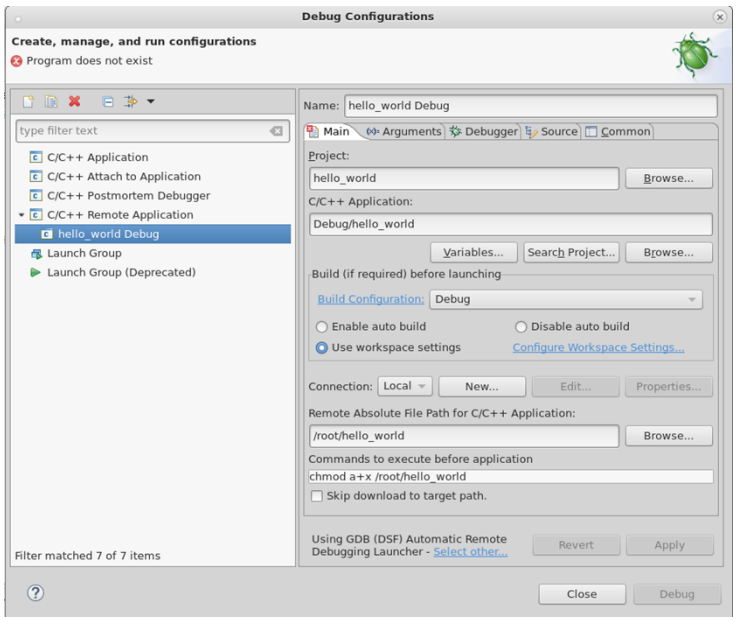


Figure 21: Setup Debug Configuration

Select at Connection *New…* select connection type *SSH* and click *OK*.

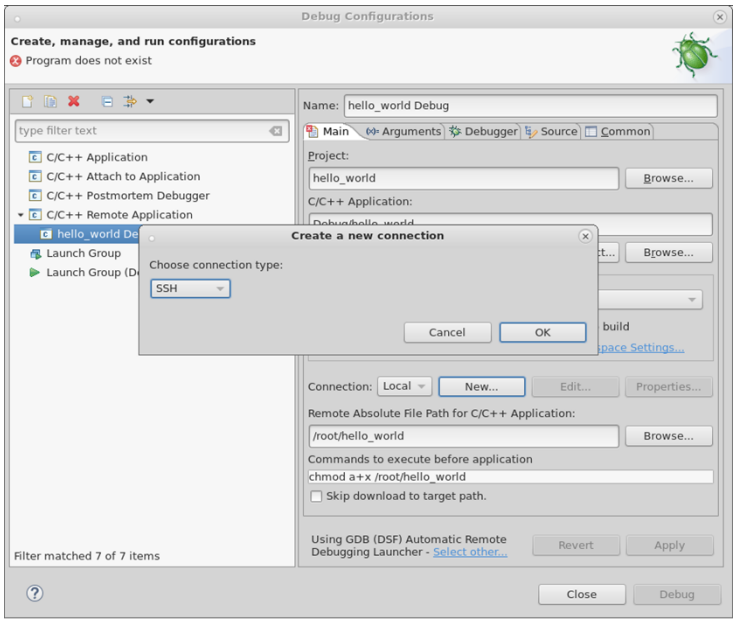


Figure 22: Create remote connection

Setup the new connection and click Finish.

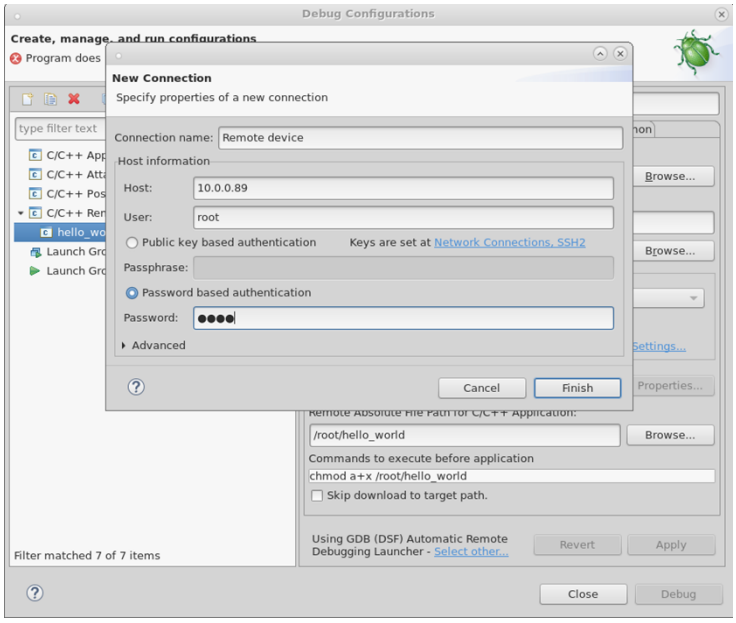


Figure 23: Setup remote connection

Setup Application settings

Switch to tab *Debugger* and select the GDB debugger path. The GDB debugger is located in the buildroot directory *…/output/host/usr/bin/arm-linux-gdb* or the meta-toolchain directory *…/5.4-zeus/sysroots/x86_64-pokysdk-linux/usr/bin/aarch64-poky-linux/aarch64-poky-linux-gdb*. After that *Apply* the settings and *Close* it.

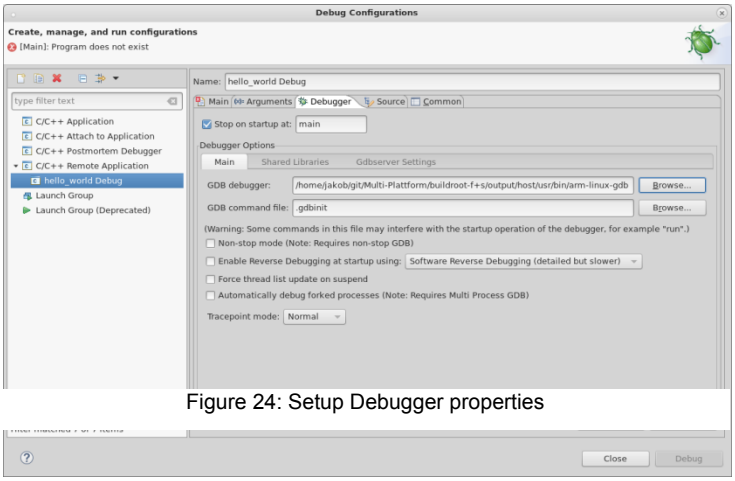


Figure 24: Setup Debugger properties

Select tab *Run* and choose *Run Configurations….* After that select *C/C++ Remote Application* and click *New launch configuration*.

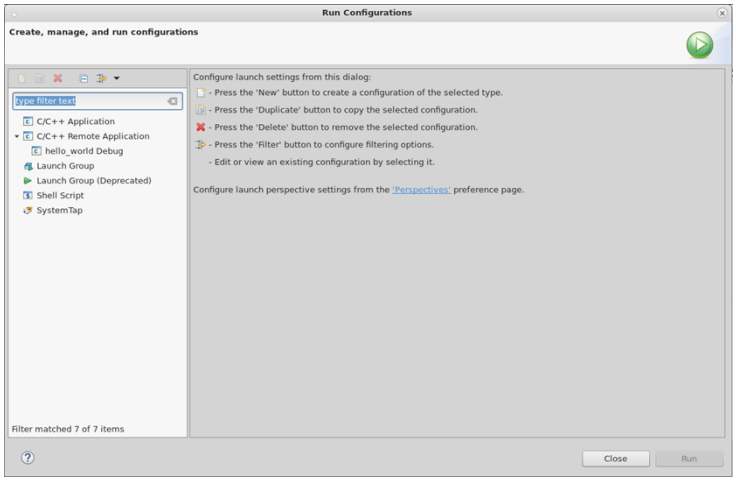


Figure 25: New Run Configuration

Setup run configuration, *Apply* it and *Close* it.

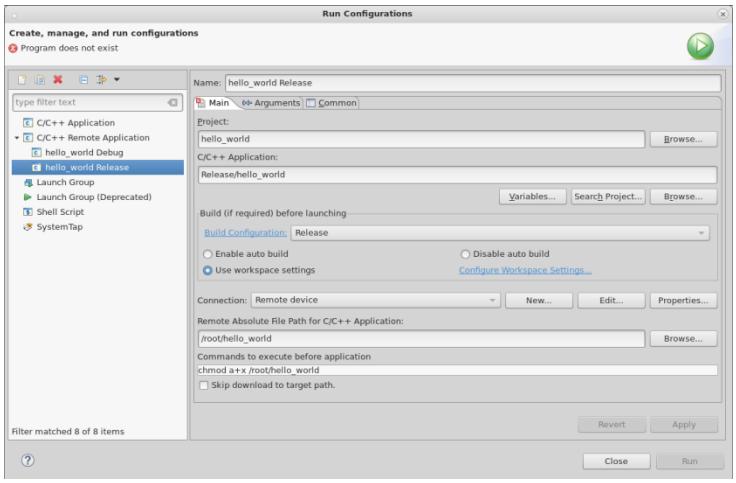


Figure 26: Setup Run Configuration

ight-Click your project and select *Properties*. Choose *C/C++ Build* and select *Settings*. *Choose Configuration: Debug* and Add an include path. Include path is also from your buildroot directory …/output/host/usr/arm-buildroot-linux-gnueabihf/sysroot/usr/include or your meta-toolchain directory …/5.4-zeus/sysroots/aarch64-poky-linux/usr/include.

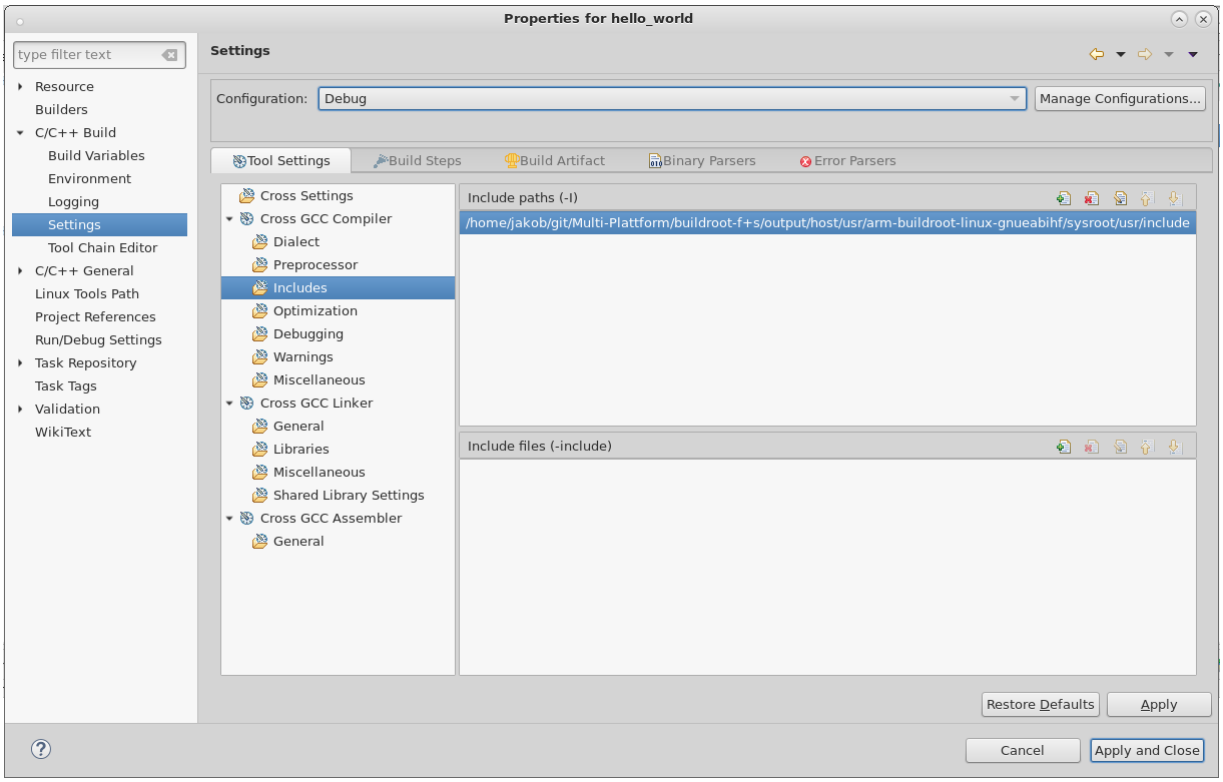


Figure 27: Add include path to Debug configuration

Setup Application settings

Switch configuration to Release and add the same path. Apply and Close the settings.

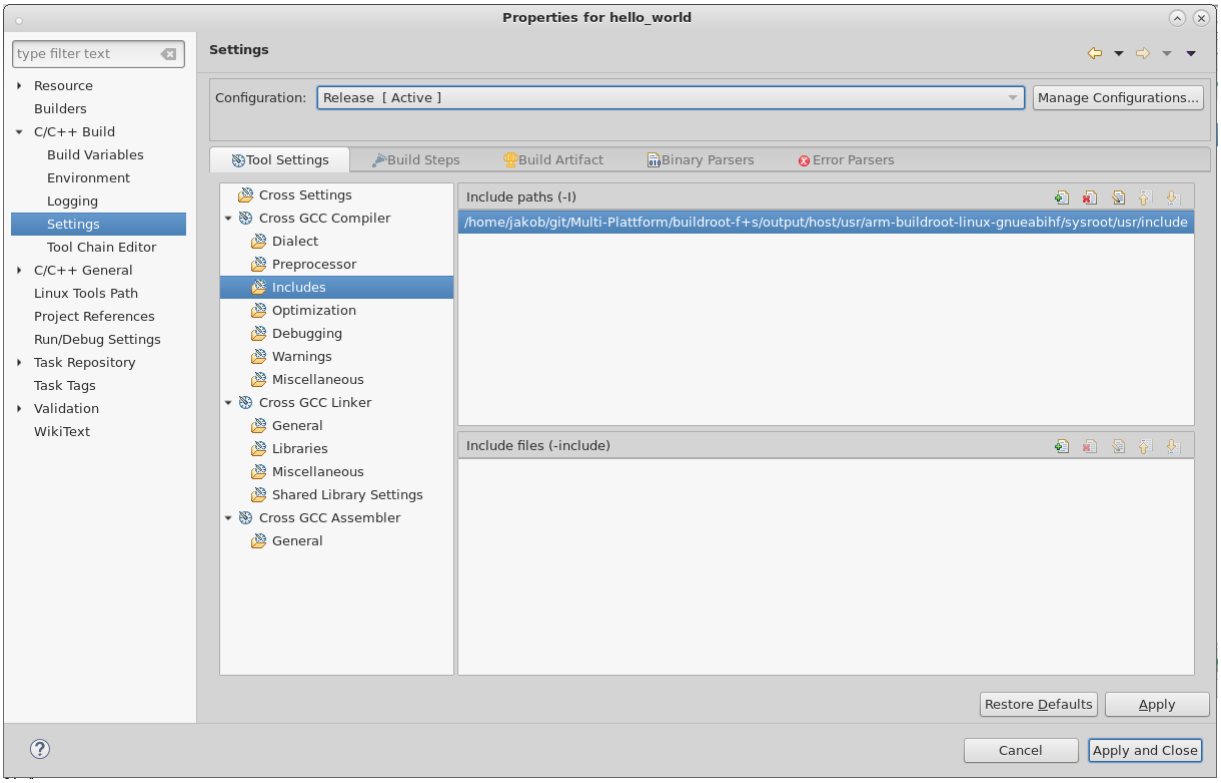


Figure 28: Add include path to Release configuration

5 Build and Debug Application

5.1 Release build

Select C/C++ perspective. Go to the downside arrow beside the hammer icon on the toolbar (right side) and choose Release. After that press the hammer icon.

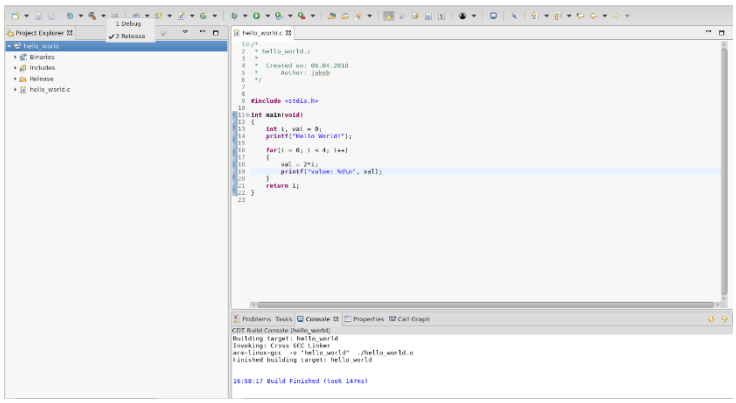


Figure 29: Build Release configuration

5.2 Run Release build on device

To run the compiled example select the downside arrow beside the run icon  (right side) and click *Run Configuration*…. Select your release configuration and click *Run*.

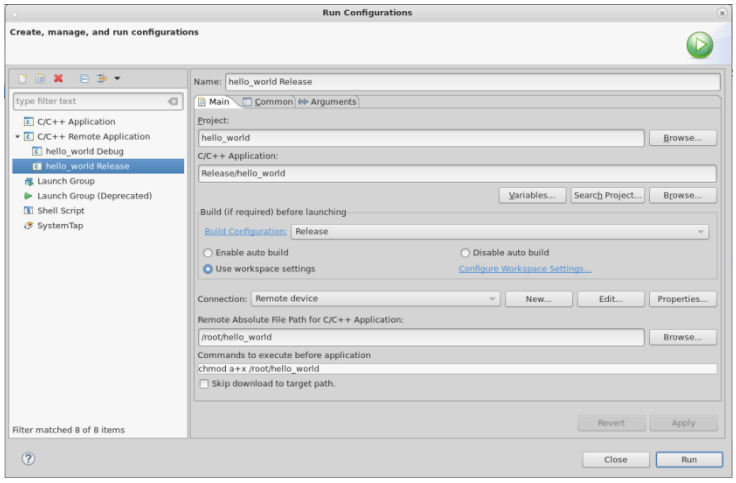


Figure 30: Run Release configuration

If the remote connection is correctly configured, hello world will appear in the console windows in the bottom margin.

5.3 Debug build

Select C/C++ perspective. Go to the arrow beside the hammer icon (right side) and choose Debug. It should automatically build the example if not, press the hammer icon.

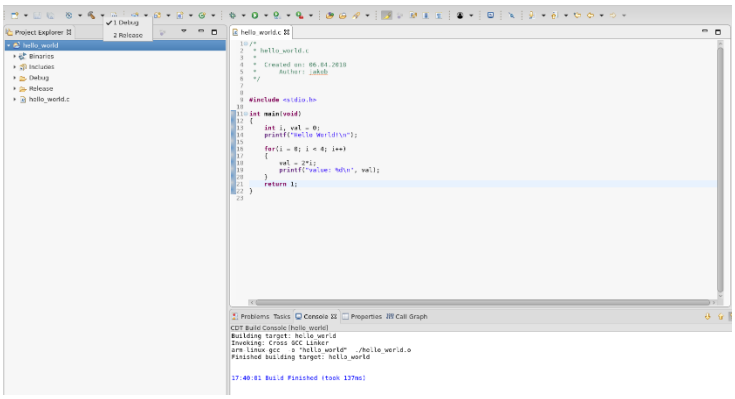


Figure 31: Build Debug configuration

5.4 Debug build on device

To debug the built example select the downside arrow beside the debug icon  (right side) and click *Run Configuration*…. Select your debug configuration and click *Run*.

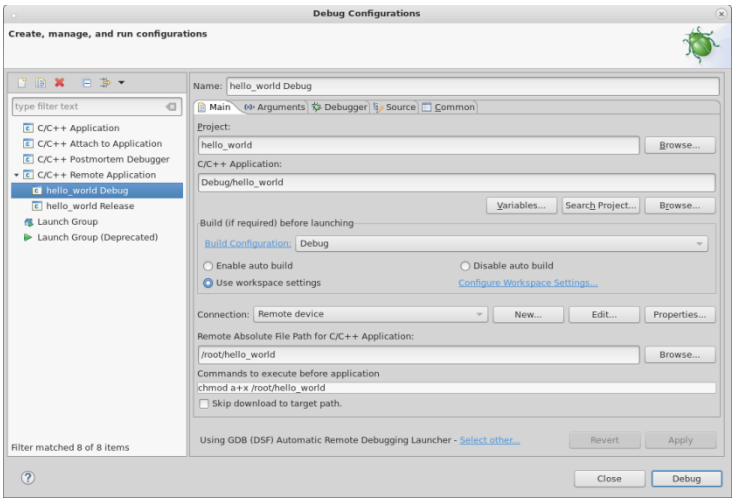


Figure 32: Debug configuration

Confirm the Perspective Switch.

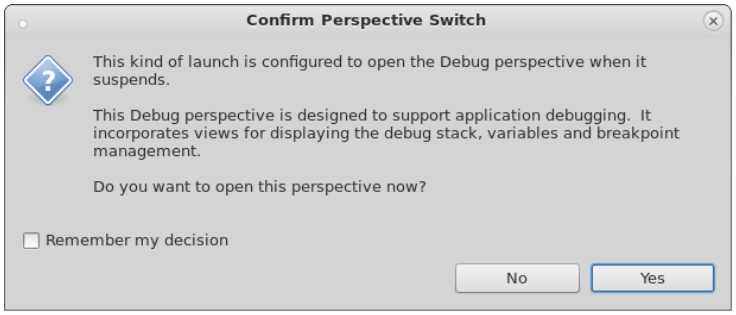


Figure 33: Confirm Perspective Switch

Now you're in Debug Perspective and you can debug your built example.

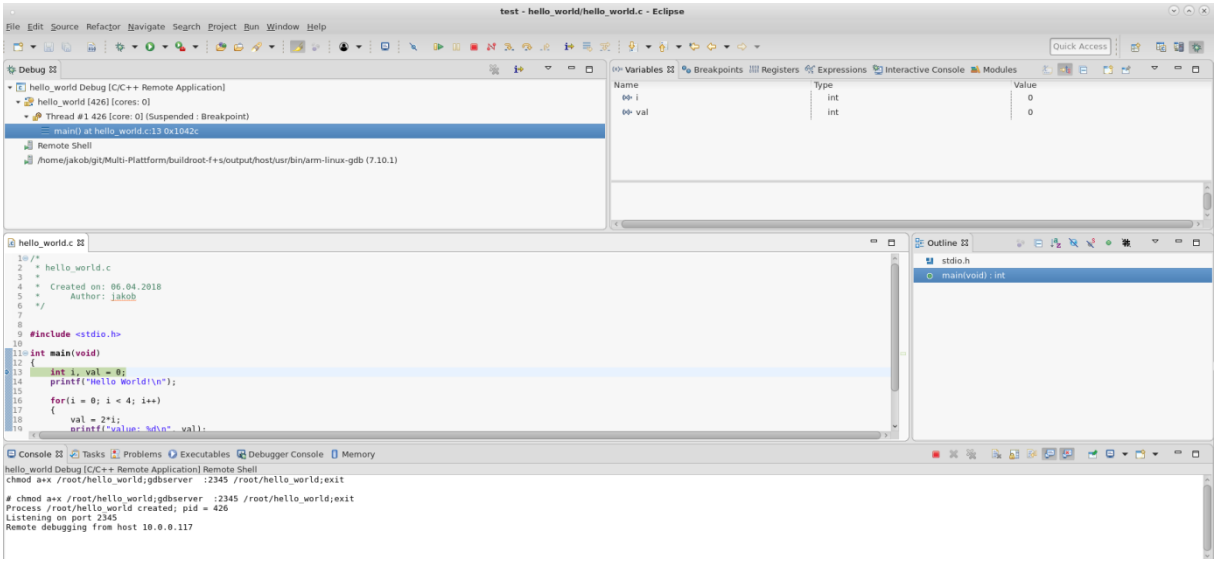


Figure 34: Debug Perspective

To setup debug points right-click the line in your program where you want to set a debug point. Choose *Add Breakpoint….* After that setup Breakpoint properties and *Apply and Close* it. You also can setup a Breakpoint by left-click the corresponding line. With the Step options you can see variables are changing.

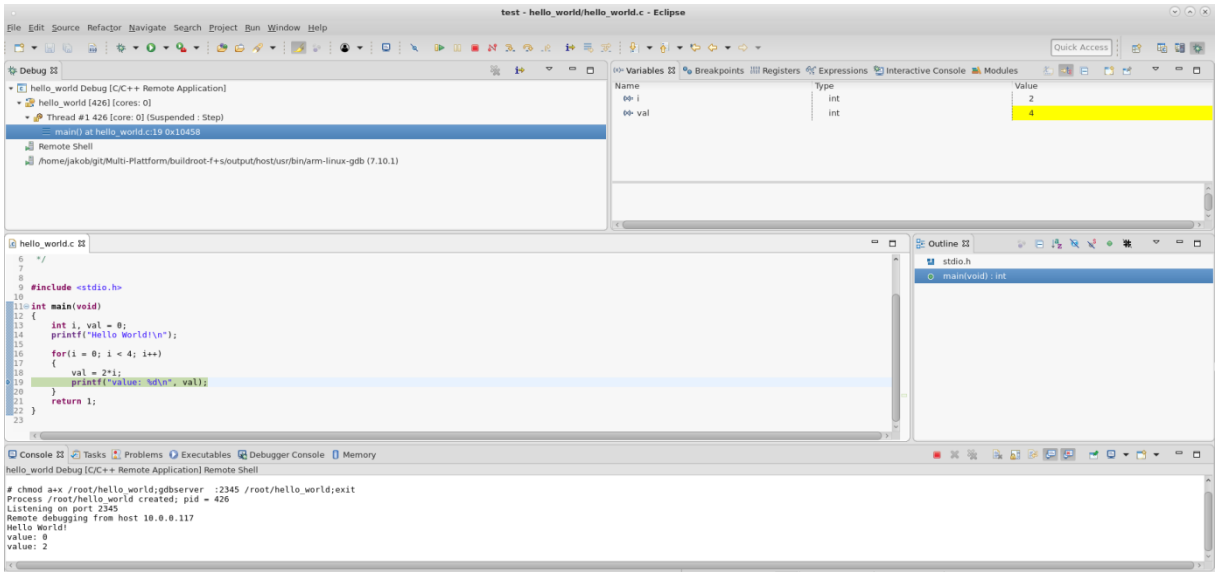


Figure 35: Variable change while debugging

6 Appendix

List of Figures

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. F&S Elektronik Systeme assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained in this documentation.

F&S Elektronik Systeme reserves the right to make changes in its products or product specifications or product documentation with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

F&S Elektronik Systeme makes no warranty or guarantee regarding the suitability of its products for any particular purpose, nor does F&S Elektronik Systeme assume any liability arising out of the documentation or use of any product and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Products are not designed, intended, or authorised for use as components in systems intended for applications intended to support or sustain life, or for any other application in which the failure of the product from F&S Elektronik Systeme could create a situation where personal injury or death may occur. Should the Buyer purchase or use a F&S Elektronik Systeme product for any such unintended or unauthorised application, the Buyer shall indemnify and hold F&S Elektronik Systeme and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorised use, even if such claim alleges that F&S Elektronik Systeme was negligent regarding the design or manufacture of said product.