

StarFive
赛昉科技

StarFive StarStudio User Guide

Version: 0.3

Date: 2022/10/20

Doc ID: Dubhe-UGEN-005

Legal Statements

Important legal notice before reading our documentation.

PROPRIETARY NOTICE

Copyright © Shanghai StarFive Technology Co., Ltd., 2018-2022. All rights reserved.

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to the product development. Shanghai StarFive Technology Co., Ltd. (hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

StarFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

All material appearing in this document is protected by copyright and is the property of StarFive. You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. StarFive authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services.

Contact Us

Address: Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: <http://www.starfivetech.com>

Email: sales@starfivetech.com(sales) , support@starfivetech.com(support)

Contents

List of Tables.....	.4
List of Figures.....	5
Legal Statements.....	ii
Preface.....	vi
1. Using StarFive StarStudio with Linux SDK.....	7
1.1. Prerequisites.....	7
1.2. Start Up StarFive StarStudio.....	8
1.3. Importing eSDK.....	8
1.4. Modifying Packages.....	9
1.5. Importing Packages to Project Explorer.....	10
1.6. Building Packages and QSPI Image.....	11
1.7. Running QEMU.....	12
1.8. Flashing Binary into FPGA.....	13
1.9. Debugging OpenSBI or Linux Kernel.....	14
1.9.1. Installing Minicom on Linux to View FPGA Output.....	15
2. Using StarFive StarStudio with Bare Metal SDK.....	18
2.1. Prerequisites.....	18
2.2. Start Up StarFive StarStudio.....	18
2.3. Importing StarFive Bare Metal SDK.....	18
2.4. Building and Cleaning Program.....	19
2.5. Adding New Compile Program.....	20
2.6. Debugging with Single Core or Multi Cores.....	22
3. Debugging on StarFive StarStudio.....	24
3.1. Debug Perspective.....	24
3.2. Memory View.....	24
3.3. Registers View.....	25
3.4. Setting Breakpoint.....	25
3.5. Disassembly View.....	26

List of Tables

Table 0-1 Revision History.....	vi
---------------------------------	----



List of Figures

Figure 1-1 Example Command and Output.....	7
Figure 1-2 Example Command and Output.....	8
Figure 1-3 StarFive Linux SDK.....	9
Figure 1-4 Terminal Interface.....	10
Figure 1-5 Import Wizard.....	10
Figure 1-6 Choose Flat Mode.....	11
Figure 1-7 Imported Package.....	11
Figure 1-8 Example Output.....	12
Figure 1-9 Build QSPI Image.....	12
Figure 1-10 Example Output.....	13
Figure 1-11 Popup Window.....	14
Figure 1-12 Example Output.....	14
Figure 1-13 Auto-populated Configuration.....	15
Figure 1-14 Example Output.....	15
Figure 1-15 Example Output.....	16
Figure 1-16 Example Output.....	16
Figure 1-17 Example Interface.....	16
Figure 1-18 Example Baud Rate.....	17
Figure 1-19 Example Output.....	17
Figure 2-1 StarFive Baremetal SDK View.....	19
Figure 2-2 Build Program Output.....	19
Figure 2-3 Example Build Outputs.....	20
Figure 2-4 Clean Program Output.....	20
Figure 2-5 Example Pop Up.....	21
Figure 2-6 Example Output.....	21
Figure 2-8 Example Interface.....	22
Figure 2-9 Example Output.....	23
Figure 3-1 Example Debug Perspective.....	24
Figure 3-2 Memory View.....	25
Figure 3-3 Registers View.....	25
Figure 3-4 Breakpoints View.....	25
Figure 3-5 Toggle Breakpoint.....	26
Figure 3-6 Debugger Console.....	26
Figure 3-7 Disassembly View.....	27

Preface

About this guide and technical support information.

About this document

This document mainly provides the users with the necessary information to use StarFive StarStudio.

StarFive StarStudio provides users an interface to interact with StarFive Bare Metal SDK and StarFive Linux SDK to perform various actions.

This document is intended to help users:

- Use StarFive StarStudio with Linux SDK as described in [Using StarFive StarStudio with Linux SDK \(on page 7\)](#).
- Use StarFive StarStudio with Bare Metal SDK as described in [Using StarFive StarStudio with Bare Metal SDK \(on page 18\)](#).

Revision History

Table 0-1 Revision History

Version	Released	Change Description
0.1	2022/06/16	Preliminary version.
0.2	2022/09/07	Updated the product name.
0.3	2022/10/20	Editorial updates.

Notes and notices

The following notes and notices might appear in this guide:

-  **Tip:**
Suggests how to apply the information in a topic or step.
-  **Note:**
Explains a special case or expands on an important point.
-  **Important:**
Points out critical information concerning a topic or step.
-  **CAUTION:**
Indicates that an action or step can cause loss of data, security problems, or performance issues.
-  **Warning:**
Indicates that an action or step can result in physical harm or cause damage to hardware.

1. Using StarFive StarStudio with Linux SDK

This chapter provides steps to use StarFive StarStudio with Dubhe Linux SDK.

It contains the following procedures:

1. Complete the [Prerequisites \(on page 7\)](#)
2. [Start Up StarFive StarStudio \(on page 8\)](#)
3. [Importing eSDK \(on page 8\)](#)
4. [Modifying Packages \(on page 9\)](#)
5. [Importing Packages to Project Explorer \(on page 10\)](#)
6. [Building Packages and QSPI Image \(on page 11\)](#)
7. [Running QEMU \(on page 12\)](#)
8. [Flashing Binary into FPGA \(on page 13\)](#)
9. [Debugging OpenSBI or Linux Kernel \(on page 14\)](#)

1.1. Prerequisites

Make sure you extract and install both StarFive StarStudio and StarFive Linux eSDK on your host machine before starting up the StarFive StarStudio:

1. Copy the StarStudio-beta-202206.tar.gz file into your directory and untar it:

```
$ tar -zxf StarStudio-beta-202206.tar.gz
```

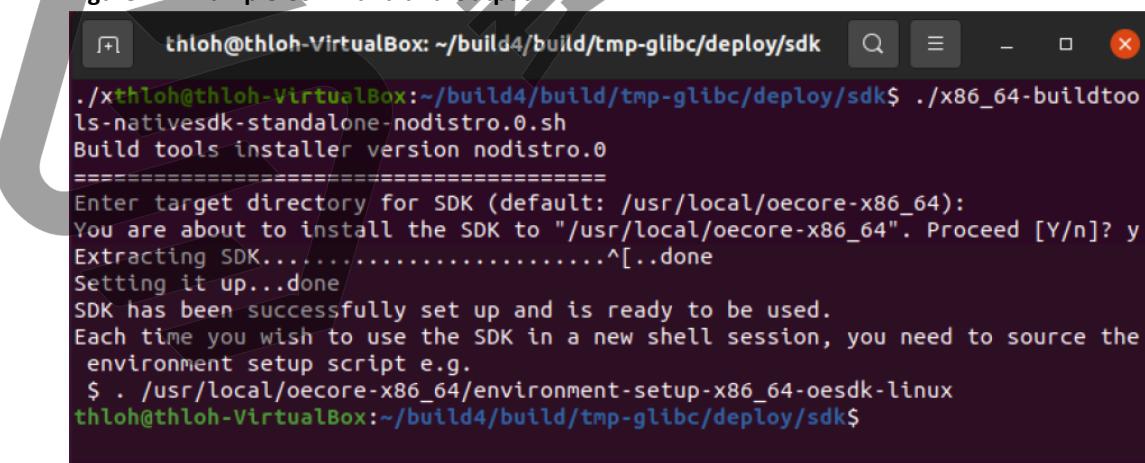
Result

The extracted folder is ready to use.

2. Install the Linux eSDK by running the following two scripts:

```
◦ $ ./x86_64-buildtools-nativesdk-standalone-nodistro.0.sh
```

Figure 1-1 Example Command and Output



```
thloh@thloh-VirtualBox:~/build4/build/tmp-glibc/deploy/sdk$ ./x86_64-buildtools-nativesdk-standalone-nodistro.0.sh
Build tools installer version nodistro.0
=====
Enter target directory for SDK (default: /usr/local/oecore-x86_64):
You are about to install the SDK to "/usr/local/oecore-x86_64". Proceed [Y/n]? y
Extracting SDK.....^[[..done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the
environment setup script e.g.
$ . /usr/local/oecore-x86_64/environment-setup-x86_64-oesdk-linux
thloh@thloh-VirtualBox:~/build4/build/tmp-glibc/deploy/sdk$
```

◦ \$./oe-core-x86_64-riscv64-toolchain-ext-nodistro.0.sh

Figure 1-2 Example Command and Output

```
mdcfga1@mdcfga1-VirtualBox:~/jy_workspace$ ./oe-core-x86_64-riscv64-toolchain-ext-nodistro.0
StarFive Linux SDK Extensible SDK installer version nodistro.0
=====
Enter target directory for SDK (default: ~/nodistro_sdk): ~/jy_workspace/esdk_openocd
You are about to install the SDK to "/home/mdcfga1/jy_workspace/esdk_openocd". Proceed [Y/n]? Y
Extracting SDK.....done
Setting it up...
Extracting buildtools...
Preparing build system...
Loading cache: 100% | ETA: --::--:-
Parsing recipes: 82% |#####| ETA: 0:00:10
```

1.2. Start Up StarFive StarStudio

To start up the StarFive StarStudio, perform the following steps:

1. Go to the StarFive StarStudio folder and open the executable to open StarFive StarStudio.
2. Select **Workspace** for StarFive StarStudio IDE and click **Launch**.
3. Close the **Welcome** tab.

1.3. Importing eSDK

Perform the following steps to import eSDK:

1. Click **Import Projects** under **Project Explorer** on the left side of your StarFive StarStudio workbench.
2. Select **StarFive Dubhe Linux SDK** under the **StarFive** folder from the Import Wizard, and click **Next**.
3. Import StarFive Linux SDK project as prompted:
 - **Project Name:** The project name. For example, `linux_sdk`.
 - **Existing eSDK Location:** The directory of the installed eSDK that you want to import. For example, `/home/jytan/linux_sdk`.
 - **Olimex OpenOCD Config File Location:** The openOCD Config file location. The default file is `<eSDK>/tmp/deploy/images/starfive-dubhe/olimex-openocd_s5.cfg`.
4. Click **Finish** and the directory of the eSDK will be imported.



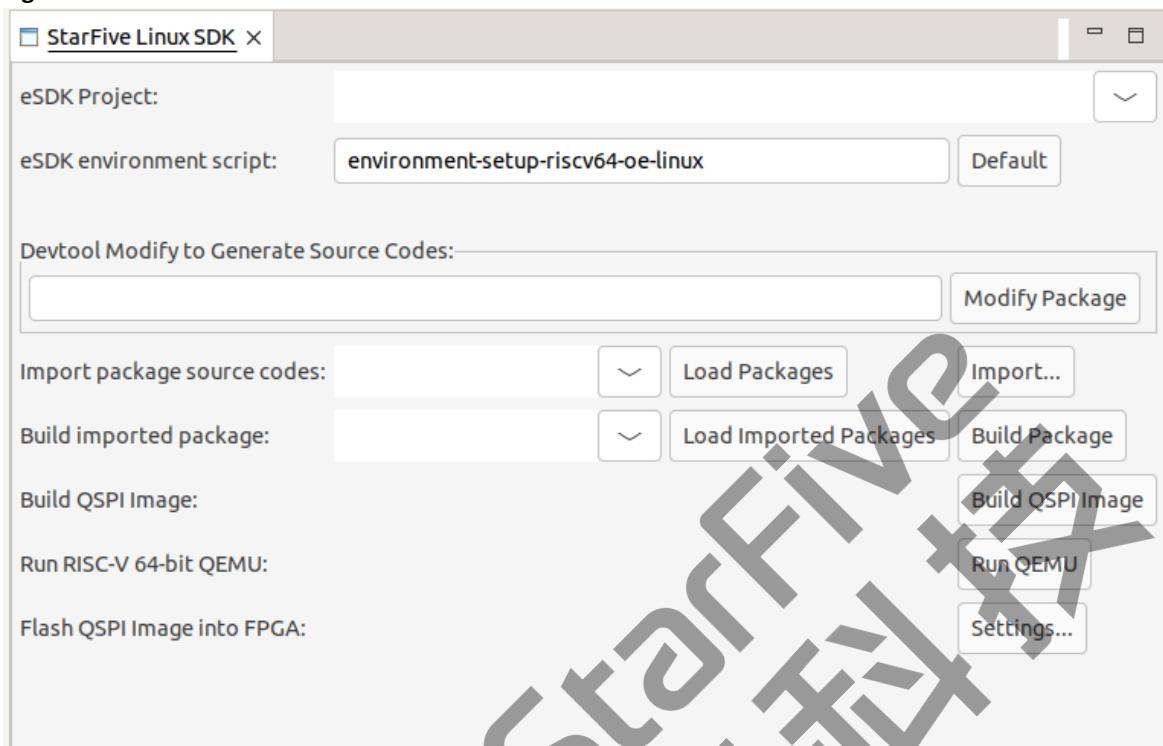
Note:

The indexing for the build folder `tmp/` is disabled as the directory is too large.

Result:

You will also be greeted with the StarFive Linux SDK View:

Figure 1-3 StarFive Linux SDK



For instructions to modify the package, see the [Modifying Packages \(on page 9\)](#) section.

1.4. Modifying Packages

Perform the following steps to modify the packages:

1. On the **StarFive Linux SDK** tab, configure the following:
 - **eSDK Project:** The eSDK project that is imported in [Importing eSDK \(on page 8\)](#). For example, choose `linux_sdk`.
 - **Devtool Modify to Generate Source Codes:** The package that you want to do development on. For example, `opensbi`.



Note:

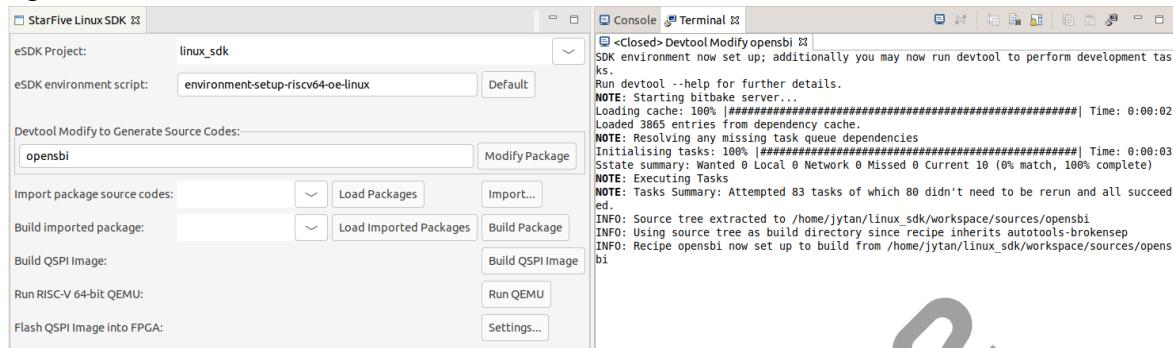
If you want to use command lines to modify packages, refer to the *Developing Applications or Modifying Packages* chapter in the *Dubhe Linux SDK User Guide*.

2. Click **Modify Package**.

Result:

The package source code is pulled into the workspace directory, and a terminal in StarFive StarStudio will be running the command:

Figure 1-4 Terminal Interface



1.5. Importing Packages to Project Explorer

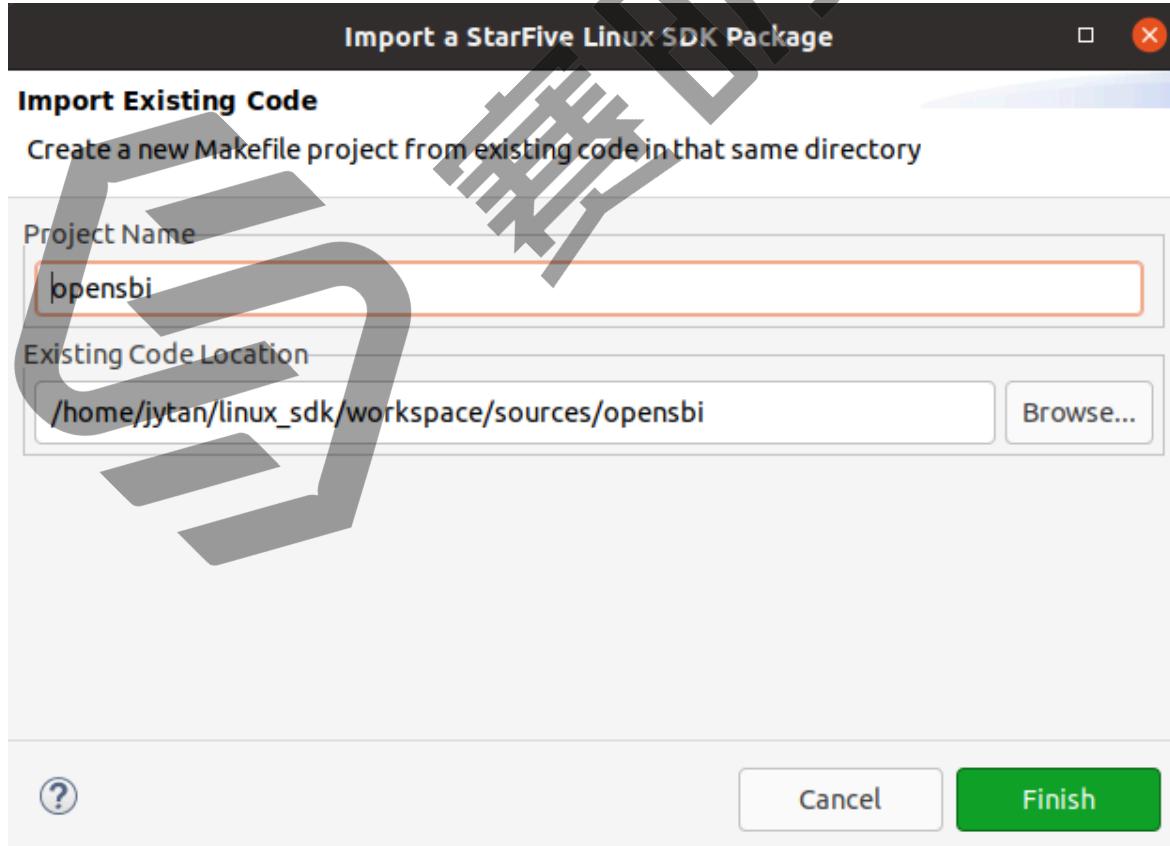
After the modification process is done, you can now import the package into StarFive StarStudio's Project Explorer. To import the opensbi package, perform the following steps:

1. Select the eSDK project under **Import package source codes**, and click **Load Packages** button on the **StarFive Linux SDK** tab to load the packages available.
2. Press the **Import...** button.

Result:

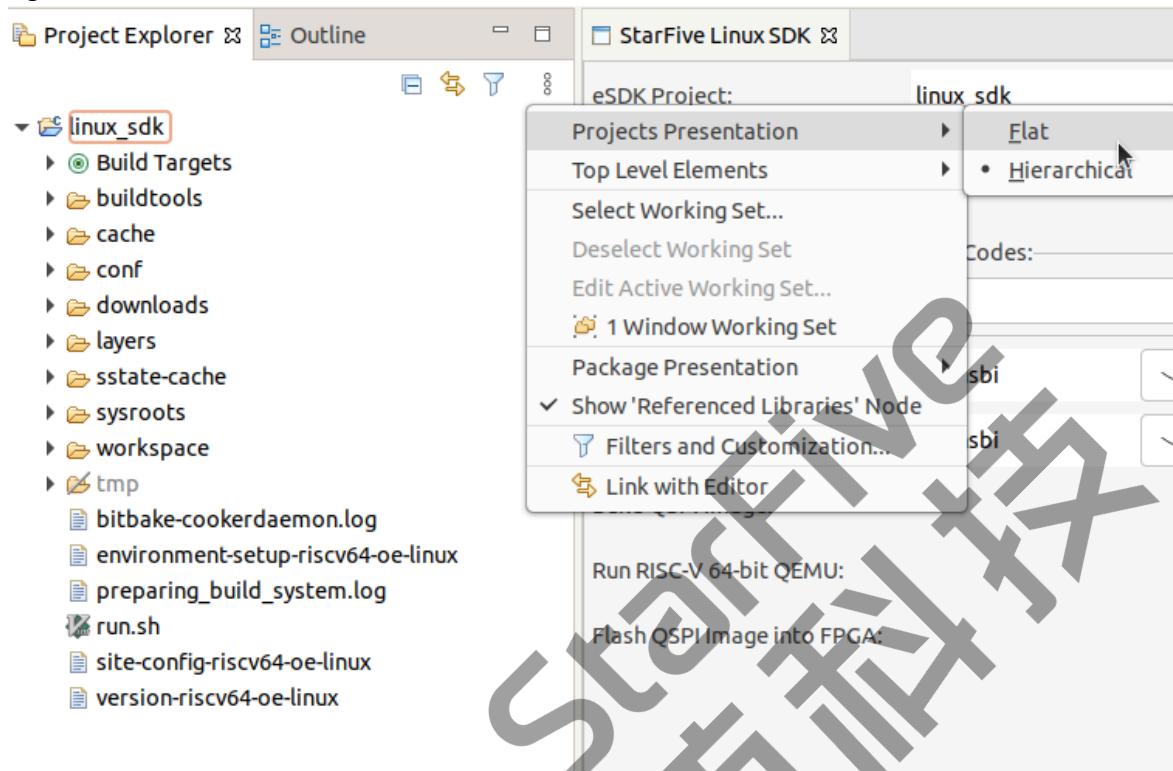
you will be directed to a simple import wizard. The data will be auto-populated:

Figure 1-5 Import Wizard

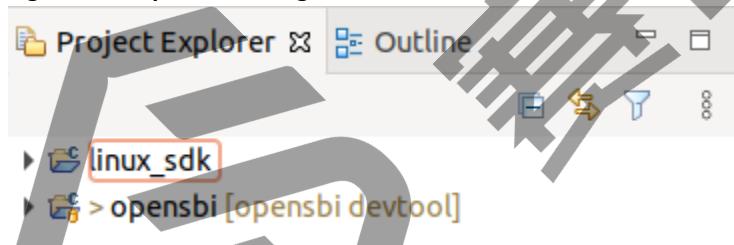


3. Press **Finish** to complete the import package process.

4. (Optional) If the Project Explorer is still the same after you import the packages, you will need to click the settings button on **Project Explorer**, and go to Projects **Presentation** to choose it in a **Flat** mode:

Figure 1-6 Choose Flat Mode**Result:**

After choosing the settings, you will now notice that the package imported is in Project Explorer now:

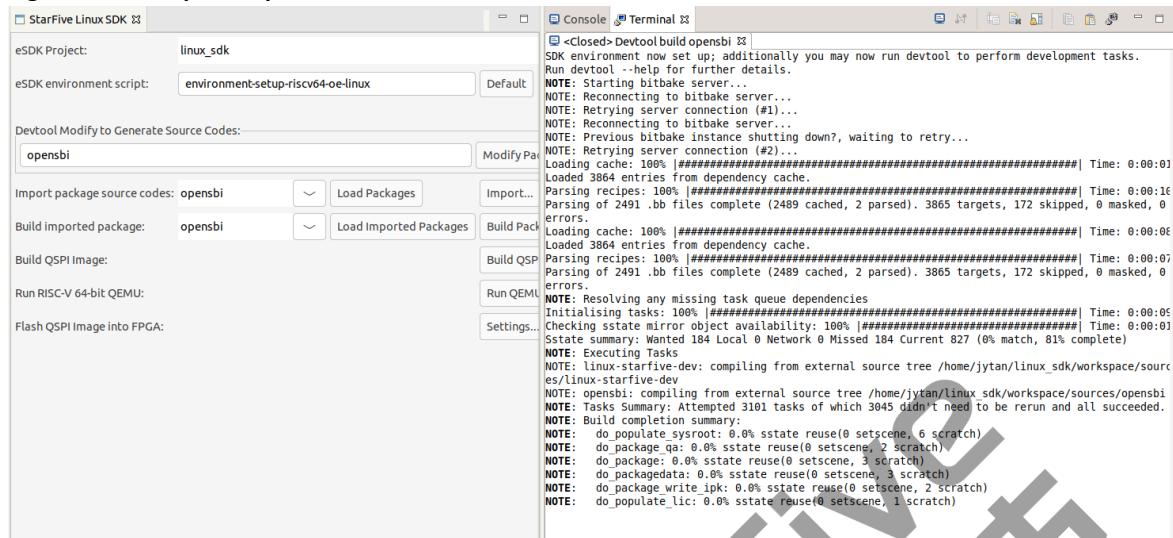
Figure 1-7 Imported Package

1.6. Building Packages and QSPI Image

After modifying your package's source code, you can now build the package.

1. Click **Build Package** to load the packages available and to build the package.

Result:

Figure 1-8 Example Output

2. (Optional) Build a new updated QSPI Image by clicking the **Build QSPI Image** button.

Figure 1-9 Build QSPI Image

1.7. Running QEMU

To run a QEMU image, press the **QEMU** button.

The following is the example output:

Figure 1-10 Example Output

The screenshot shows a software interface with a tab bar at the top containing 'Console' and 'Terminal X'. The 'Console' tab is active, showing the output of a QEMU run. The log includes details about network configuration, device drivers, and kernel parameters. Below the log, the text 'OpenSBI v1.0' is followed by a detailed configuration dump:

```

OpenSBI v1.0

Platform Name : riscv-virtio,gemu
Platform Features : medeleg
Platform HART Count : 8
Platform IPI Device : aclint-mswi
Platform Timer Device : aclint-mtimer @ 10000000Hz
Platform Console Device : uart8250
Platform HSM Device : ...
Platform Reboot Device : sifive_test
Platform Shutdown Device : sifive_test
Firmware Base : 0x80000000
Firmware Size : 312 KB
Runtime SBI Version : 0.3

Domain0 Name : root

```

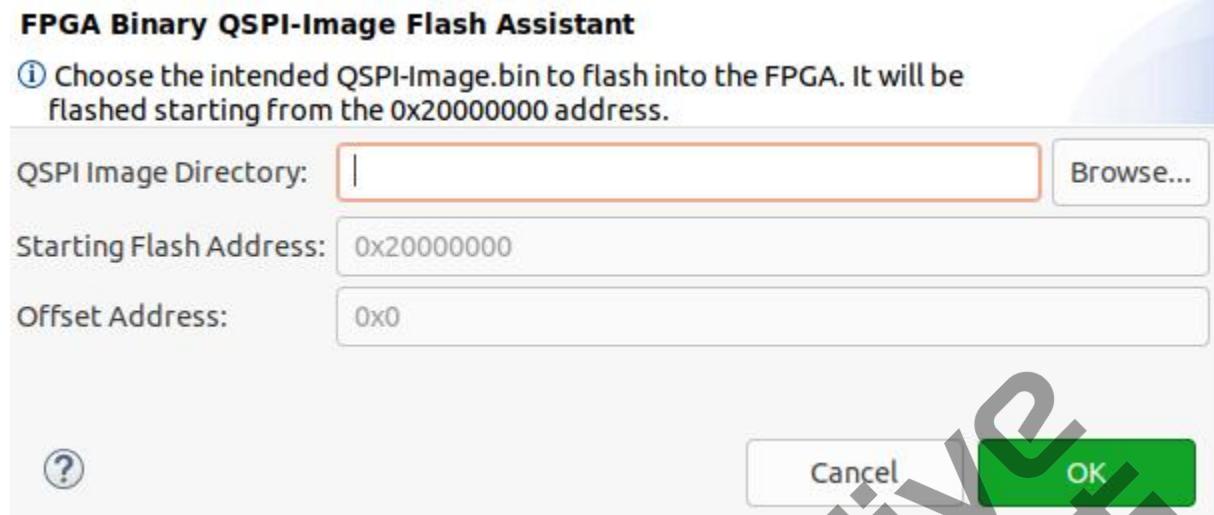
1.8. Flashing Binary into FPGA

To flash the generated FPGA Binary Image (QSPI-Image.bin) into the FPGA, perform the following step:

Press **Settings...** button on the Flash QSPI Image into the FPGA row:

Result:

An FPGA Window will pop out:

Figure 1-11 Popup Window**Figure 1-12 Example Output**

```

Console Terminal <Closed> FPGA Flashing
SDK environment now set up; additionally you may now run devtool to perform development tasks.
Run devtool --help for further details.
Open On-Chip Debugger 0.11.0+dev-02358-ga037b20f2-dirty (2022-05-18-07:11)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
configure work area .....
configure work area complete.....
configure flash bank.....
configure flash bank complete .....
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi t
do sample_edge falling"
Info : clock speed 20000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x00000cf0 (mfg: 0x67e (<unknown>), part: 0x
0000, ver: 0x0)
Info : [riscv.cpu] dataCount=4 prodbufsize=16
Info : Disabling abstract command reads from CSRs.
Info : Examined RISC-V core; found 2 harts
Info : hart 0: XLEN=64, misa=0x80000000001411af
[riscv.cpu] Target successfully examined.
Info : starting gdb server for riscv.cpu on 3333
Info : Listening on port 3333 for gdb connections
Info : JTAG tap: riscv.cpu tap/device found: 0x00000cf0 (mfg: 0x67e (<unknown>), part: 0x
0000, ver: 0x0)
Info : Found flash device 'mac 66ulg45g' (ID 0x003b25c2)
I am flashing...
Info : Disabling abstract command writes to CSRs.
Done flashing!!!
shutdown command invoked

```

1.9. Debugging OpenSBI or Linux Kernel

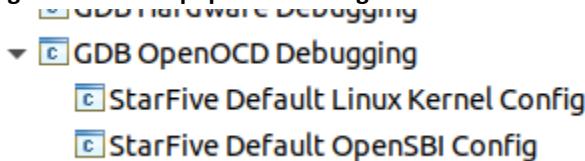
Prerequisite:

The StarFive Linux eSDK project, openSBI or Linux Kernel (`linux-starfive-dev`), is imported into StarFive StarStudio, and two configurations are auto-populated: **openSBI** and **Linux Kernel**:

To debug OpenSBI or Linux Kernel, perform the following steps:

1. Navigate to **Run > Debug Configurations...** and double click **GDB OpenOCD Debugging**. There will be two default configs available:

Figure 1-13 Auto-populated Configuration

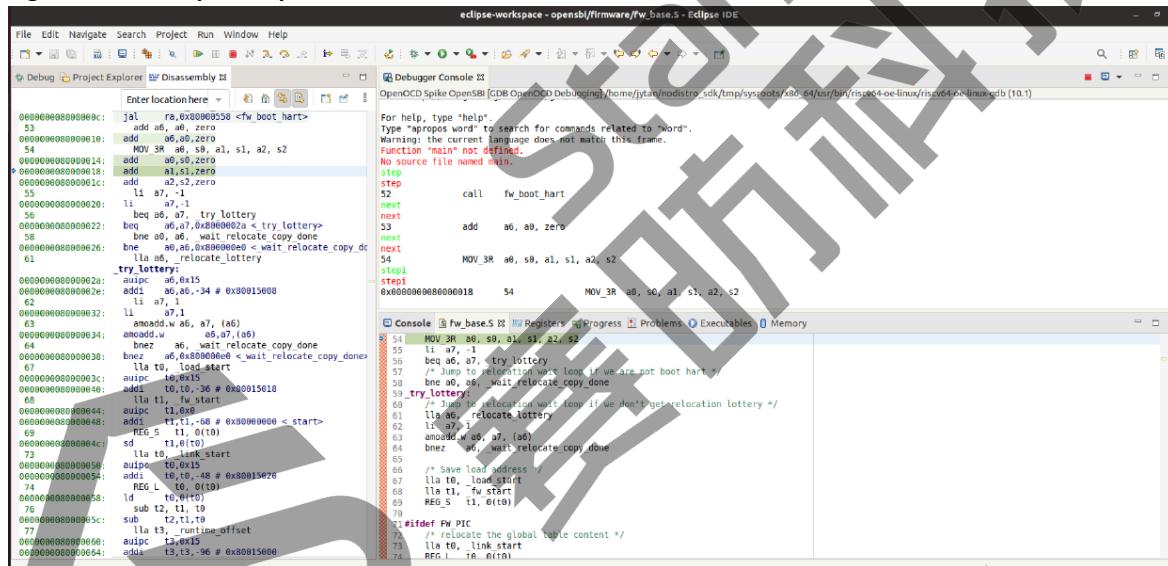


2. Choose the configuration (**openSBI** in this example) and press **Debug** to debug with the config.

Result:

You can now debug OpenSBI! For a detailed introduction to debugging on StarFive StarStudio, see [Debugging on StarFive StarStudio \(on page 24\)](#).

Figure 1-14 Example Output



You will need to connect to the FPGA via minicom to view the FPGA's output. For instructions for setting up minicom, see [Installing Minicom on Linux to View FPGA Output \(on page 15\)](#).

1.9.1. Installing Minicom on Linux to View FPGA Output

To connect to the FPGA via minicom to view the FPGA's output, perform the following steps:

1. To install minicom on Linux, run this command:

```
$ sudo apt-get install minicom
```

2. You can check which serial port the FTDI USB UART (not the Olimex JTAG) is connected to using the command `$ ls -lR /dev/serial`. Ensure that the FTDI USB UART has the correct serial number for Dubhe (in case you are connecting to more than 1 FTDI UARTs).

Figure 1-15 Example Output

```
mdcfgai@mdcfgai-VirtualBox:~/Desktop$ ls -lR /dev/serial
/dev/serial:
total 0
drwxr-xr-x 2 root root 80 Mai 20 13:27 by-id
drwxr-xr-x 2 root root 80 Mai 20 13:27 by-path

/dev/serial/by-id:
total 0
lrwxrwxrwx 1 root root 13 Mai 20 13:27 usb-15ba_Olimex_OpenOCD_JTAG_ARM-USB-TINY-H_OLDBAA13-lf01-port0 -> ../../ttyUSB0
lrwxrwxrwx 1 root root 13 Mai 20 13:27 usb-FTDI_FT232R_USB_UART_A20021XZ-lf00-port0 -> ../../ttyUSB1
```

3. Execute the following command to set up minicom:

```
$ sudo minicom -s
```

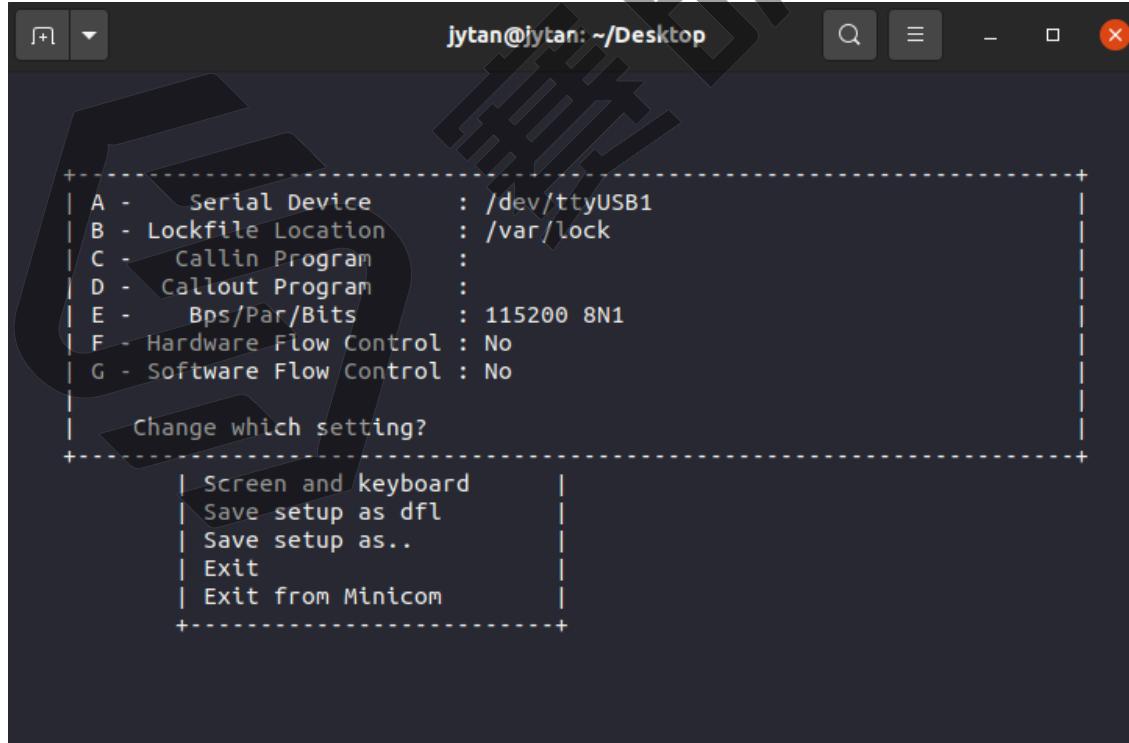
Result:

You will be brought to an interface shown below:

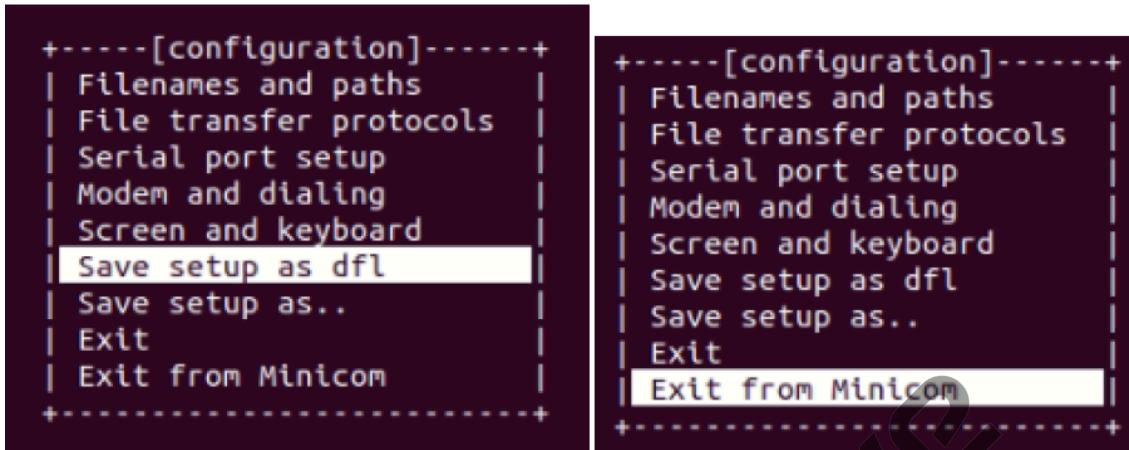
Figure 1-16 Example Output

```
+----[configuration]----+
| Filenames and paths
| File transfer protocols
| Serial port setup
| Modem and dialing
| Screen and keyboard
| Save setup as dfl
| Save setup as..
| Exit
| Exit from Minicom
+---+
```

4. Go to **Serial port setup** and press **A**. Change the serial device to the port that the FTDI is connected to (for example: /dev/ttyUSB1). Press Enter after typing the serial port name, and type **F** to set the **Hardware Flow Control** to **No**. Ensure that the baud rate in **E** is **115200**.

Figure 1-17 Example Interface

5. Go to **Save setup as dfl** and press **Enter** to save the configuration. Then, go down to **Exit from Minicom** to exit.

Figure 1-18 Example Baud Rate

- After successfully saving the configuration, you can now use this command in the terminal to get access to the FTDI USB port:

```
$ sudo minicom
```

Figure 1-19 Example Output

- After successfully saving the configuration, you can now use this command in the terminal to get access to the FTDI USB port:

```
$ sudo minicom
```

2. Using StarFive StarStudio with Bare Metal SDK

This chapter provides steps to use StarFive StarStudio with Dubhe Bare Metal SDK.

It contains the following procedures:

1. Complete [Prerequisites \(on page 18\)](#)
2. [Start Up StarFive StarStudio \(on page 8\)](#)
3. [Importing StarFive Bare Metal SDK \(on page 18\)](#)
4. [Building and Cleaning Program \(on page 19\)](#)
5. [Adding New Compile Program \(on page 20\)](#)
6. [Debugging with Single Core or Multi Cores \(on page 22\)](#)

2.1. Prerequisites

Make sure you perform the following steps before starting up the StarFive StarStudio:

1. Extracted StarFive StarStudio on your host machine.
2. Copy the `StarStudio-beta-202206.tar.gz` file into your directory and untar it:

```
$ tar -zvxf StarStudio-beta-202206.tar.gz
```

Result

The extracted folder is ready to use.

3. Extracted and installed StarFive Bare Metal SDK on your host machine.
4. Have a pre-built StarFive bare metal toolchain on your host machine.



Note:

To obtain Bare Metal SDK and toolchain files, please refer to the *Dubhe Bare Metal SDK User Guide* for the relevant links and steps to install.

2.2. Start Up StarFive StarStudio

To start up the StarFive StarStudio, perform the following steps:

1. Go to the StarFive StarStudio folder and open the executable to open StarFive StarStudio.
2. Select **Workspace** for StarFive StarStudio IDE and click **Launch**.
3. Close the **Welcome** tab.

2.3. Importing StarFive Bare Metal SDK

1. Click **Import Projects** in Project Explorer on the left side of your StarFive StarStudio workbench:
2. Select **StarFive Dubhe Baremetal SDK** under the **StarFive** folder from the Import Wizard, and click **Next**:
3. Choose your existing Bare Metal SDK:
4. Import the Bare Metal SDK project as prompted:
 - **Project Name:** The project name. For example, `bare-metal-sdk2`.
 - **Existing SDK Location:** The directory of the existing bare metal that you want to import. For example, `/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk`.

5. Click **Finish** and the files and directory of the SDK will be imported.

Result:

You will be greeted with the StarFive Bare Metal SDK view:

Figure 2-1 StarFive Baremetal SDK View**Note:**

You can also re-open the Baremetal view by going to **Window > Show View > StarFive Baremetal SDK**:

6. To start compiling programs for Dubhe, you will need to fill in the **Project Name** and the **Toolchain Path** (Fill in with the pre-built bare metal toolchain) on the StarFive Bare Metal SDK view.

2.4. Building and Cleaning Program

This section provides steps to build and clean the program:

1. Choose the **PROGRAM**, and click **Load Programs from Project** so that it will load the programs located in the directory <baremetal_SDK>/software.
2. Choose the **Target** and **Configuration** for compiling, and click **BUILD PROGRAM** to build the program with the selected toolchain.

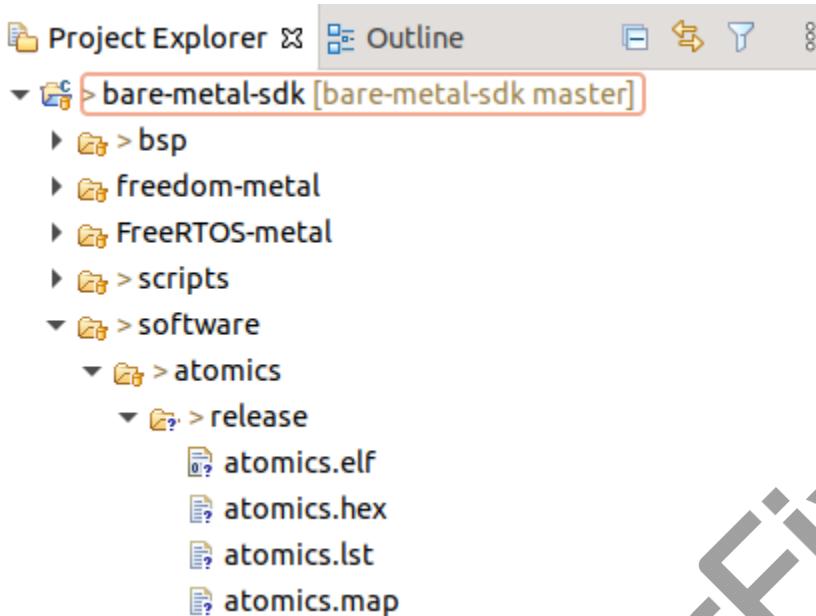
Result:

After the program is built, you will see an output in the **Console View**:

Figure 2-2 Build Program Output

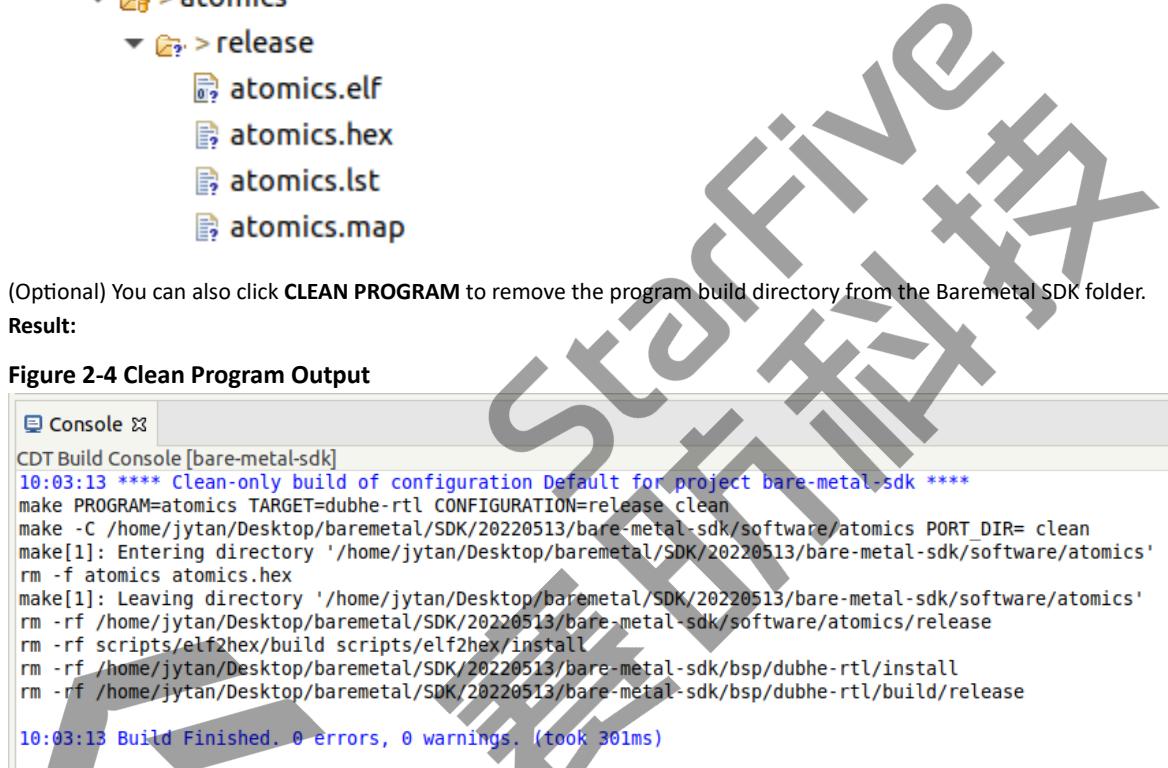
```
CDT Build Console [bare-metal-sdk]
/usr/bin/install -c elf2bin '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/scripts/elf2hex/install/bin/.riscv64-unknown-elf-elf2bin'
/usr/bin/mkdir -p '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/scripts/elf2hex/install/lib/python2.7/site-packages'
/usr/bin/install -c -m 644 /home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/scripts/elf2hex/freedom-bin2hex.py '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/scripts/elf2hex/install/bin/riscv64-unknown-elf-elf2hex'
Byte-compiling python modules...
freedom-bin2hex.py
Byte-compiling python modules (optimized versions) ...
freedom-bin2hex.py
make[2]: Leaving directory '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/scripts/elf2hex/build'
make[1]: Leaving directory '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/scripts/elf2hex/build'
touch -c scripts/elf2hex/install/bin/riscv64-unknown-elf-elf2hex
scripts/elf2hex/install/bin/riscv64-unknown-elf-elf2hex --output /home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/software/atomics/release/atomics.hex --in
10:01:55 Build Finished - 0 errors, 0 warnings. (took 10s.255ms)
```

3. Locate the build outputs in the directory <baremetal_SDK>/software/<program>/<config> after the build is finished:

Figure 2-3 Example Build Outputs

4. (Optional) You can also click **CLEAN PROGRAM** to remove the program build directory from the Baremetal SDK folder.

Result:

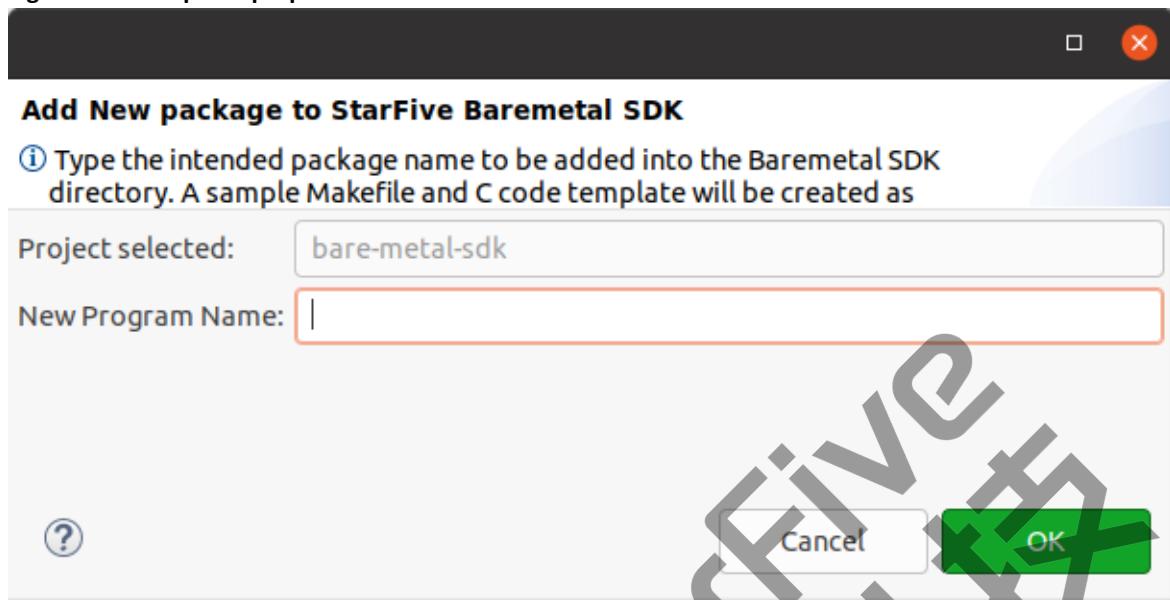
Figure 2-4 Clean Program Output


```
CDT Build Console [bare-metal-sdk]
10:03:13 **** Clean-only build of configuration Default for project bare-metal-sdk ****
make PROGRAM=atomics TARGET=dubhe-rtl CONFIGURATION=release clean
make -C /home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/software/atomics PORT_DIR= clean
make[1]: Entering directory '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/software/atomics'
rm -f atomics atomics.hex
make[1]: Leaving directory '/home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/software/atomics'
rm -rf /home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/software/atomics/release
rm -rf scripts/elf2hex/build scripts/elf2hex/install
rm -rf /home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/bsp/dubhe-rtl/install
rm -rf /home/jytan/Desktop/baremetal/SDK/20220513/bare-metal-sdk/bsp/dubhe-rtl/build/release
10:03:13 Build Finished. 0 errors, 0 warnings. (took 301ms)
```

2.5. Adding New Compile Program

You can also add a new program via StarFive StarStudio plugin.

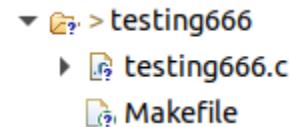
1. Choose the project and click **Add New Program** in the Baremetal View and a pop up will appear:

Figure 2-5 Example Pop Up

2. Enter the project name, for example, `testing666` in **New Program Name**, and press **OK**.

Result:

A new folder will be created in the Bare Metal SDK, with sample templates of a Makefile and a template C file:

Figure 2-6 Example Output

```

Makefile testing666.c
/* Copyright 2021 StarFive, Inc */
/* SPDX-License-Identifier: Apache-2.0 */

#include <stdio.h>

static void welcome(void)
{
    printf("\r\n");
    printf("*****\r\n");
    printf("***** StarFive.Inc \r\n");
    printf("***** Example C code Template \r\n");
    printf("*****\r\n");
    printf("**** Description:\r\n");
    printf("**** The testing666 demo code\r\n");
    printf("*****\r\n");
}

int main()
{
    welcome();
    printf("[INFO]: testing666 PASSED!\r\n");
    return 0;
}

```

| 2 - Using StarFive StarStudio with Bare Metal SDK

3. You can edit the C file and Makefile to any code for development, and the program can also be built via the StarFive Baremetal SDK View:

Figure 2-8 Example Interface



2.6. Debugging with Single Core or Multi Cores

Prerequisite:

Use the debug function, make sure the followings are completed:

- The bare metal SDK is imported as described in [Importing StarFive Bare Metal SDK \(on page 18\)](#).
- The minicom is installed as described in [Installing Minicom on Linux to View FPGA Output \(on page 15\)](#).

This section provides steps to debug with single core or multi cores.

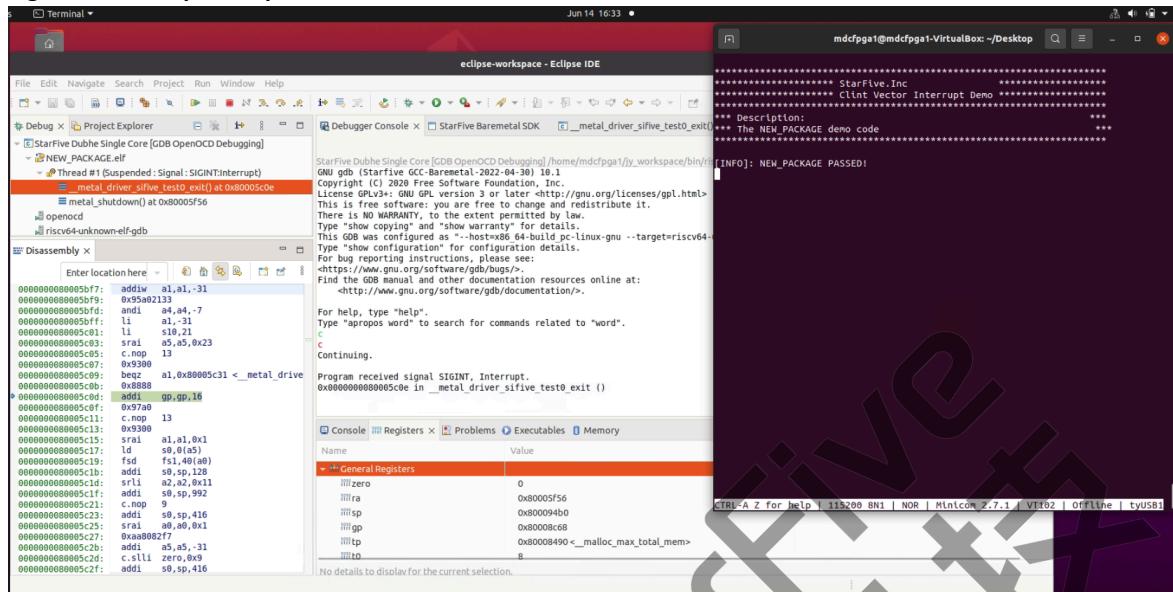
1. Navigate to the **Debug Config** under **Debug ELF with FPGA** on the **StarFive Baremetal SDK** view, and select **Single Core** or **Dual Core**.
2. Launch minicom to show output from FPGA, click **Debug** and openOCD will launch to connect with the FPGA.



Note:

You will need to open a Debugger Console (**Window > Show View > Debugger Console**) to send commands on GDB for debugging.

3. Click **DEBUG**.

Result:**Figure 2-9 Example Output**

3. Debugging on StarFive StarStudio

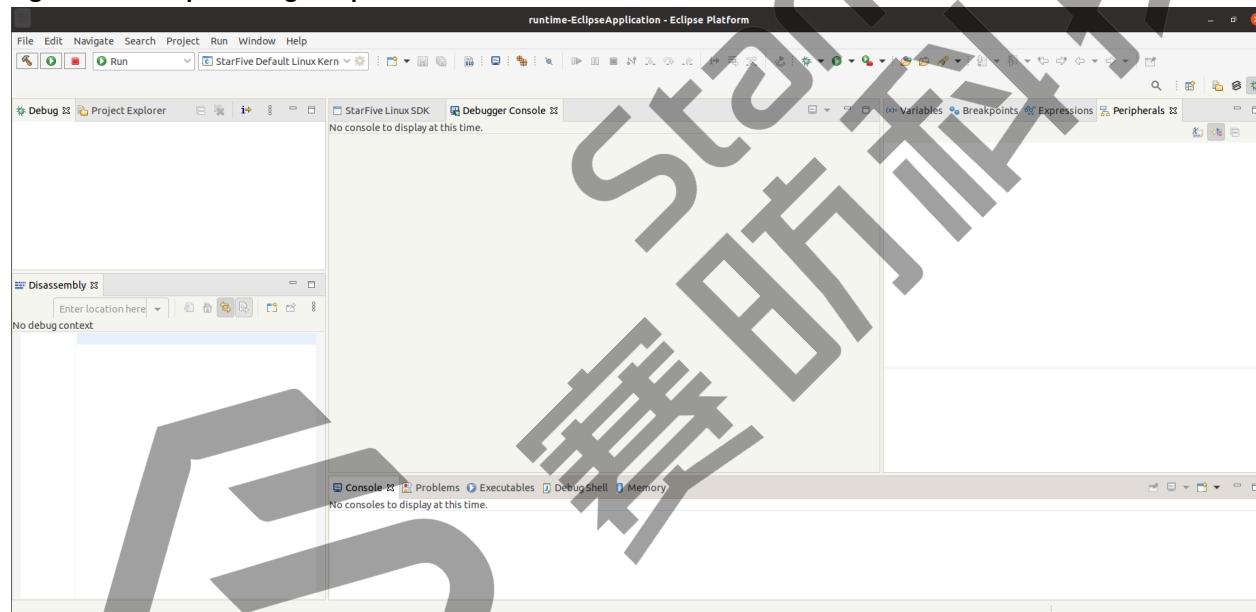
When debugging on StarFive StarStudio IDE, a lot of useful tools or views can assist you to debug more efficiently and effectively. This chapter will introduce the Debug perspective and various views that are useful when debugging.

- [Debug Perspective \(on page 24\)](#)
- [Memory View \(on page 24\)](#)
- [Registers View \(on page 25\)](#)
- [Setting Breakpoint \(on page 25\)](#)
- [Disassembly View \(on page 26\)](#)

3.1. Debug Perspective

During debugging, you can open the **Debug Perspective** on StarFive StarStudio to have a more productive view as shown in the following figure.

Figure 3-1 Example Debug Perspective



To open the **Debug Perspective**:

1. Go to **Window > Perspective > Open Perspective > Other... > Debug**
2. (Optional) If you still need some useful Views such as Disassembly View, you can go to **Window > Show View** to open the view that is needed.

3.2. Memory View

The **Memory** view in the **Debug** perspective enables users to monitor and modify the process memory. The process memory is presented as memory monitors, and users can specify the base address or location of each memory monitor.

You can add or remove memory monitors in the **Memory** view, and each memory monitor can display the memory content in different data formats such as hexadecimal or ascii.

Figure 3-2 Memory View


Address	0 - 3	4 - 7	8 - B	C - F
80005640	233CD4FC	EFF01FB2	AAB41735	01001305
80005650	65A4EFF0	8FB8A270	02742685	E2646561
80005660	82805971	22F006F4	001826EC	0CE410E8
80005670	14EC18F0	1CF42338	0493233C	1403F327
80005680	0034BC6B	13078400	233CE4FC	8988B144
80005690	99E7A270	02742685	E2646561	8280AA84
800056A0	17350100	1305059F	EFF06FB0	833684FD
800056B0	26868145	0145EFF0	FFAAAAA8	17350100
800056C0	12054500	EE506ED1	13200271	00000001

3.3. Registers View

StarFive StarStudio comes with a **Registers** view, that enables users to view the info of general-purpose registers for the target while debugging. Register values that have been changed would also be highlighted in the **Registers** view when the program stops. Users can also modify or assign the memory values of the target via **Registers** view.

Figure 3-3 Registers View


Name	Value	Description
General Registers		
zero	0	General Purpose and FRU Register Group
ra	0x8000090e <sbi_init+642>	
sp	0x80040F30	
gp	0x0	
tp	0x80041000	
t0	0	
t1	4672	

3.4. Setting Breakpoint

Users can set breakpoint during debugging.

Figure 3-4 Breakpoints View

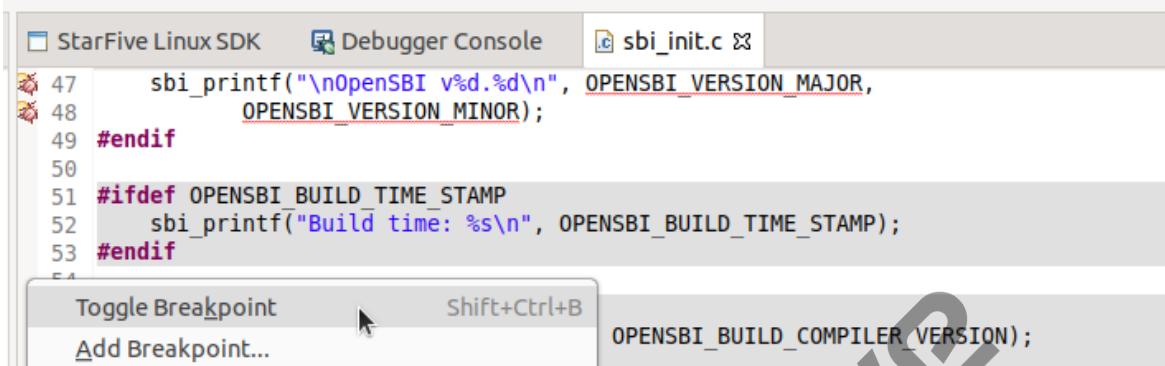

Variables	Breakpoints	Expressions	Modules	Peripherals
	<input checked="" type="checkbox"/> [function: sbi_init] [type: Hardware] <input checked="" type="checkbox"/> [function: sbi_printf] [type: Hardware]			
No details to display for the current selection.				

Users can view, delete, or deactivate breakpoints and modify breakpoint properties with the **Breakpoints** view in StarFive StarStudio. For users to set a breakpoint, users can either:

| 3 - Debugging on StarFive StarStudio

- Right click in the left margin in the editor View and select **Toggle Breakpoint**. Double clicking the left margin of the line works too:

Figure 3-5 Toggle Breakpoint



- Alternatively, users can also set breakpoints during debugging. Via the debugger console, the user can set either a hardware breakpoint (**hbreak**) or a software breakpoint (**break**).

Figure 3-6 Debugger Console

```
StarFive Default OpenSBI Config [GDB OpenOCD Debugging] /home/mdcfpga1/jy_workspace/esdk_openocd/tmp/sysroots/x86_64/usr/bin/riscv64-oe-linux/riscv64-oe-linux-gdb (10.1)
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show warranty" for details.
This GDB was configured as "--host=x86_64-linux --target=riscv64-oe-linux".
Type "show configuration" for configuration details.
For bug reports, please use: <http://www.gnu.org/software/gdb/bugs/>
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
hbreak sbi_printf
Hardware assisted breakpoint 2 at 0x8000560c: file /home/mdcfpga1/jy_workspace/esdk_openocd/workspace/sources/opensbi/lib/sbi/sbi_console.c, line 375.
```

3.5. Disassembly View

The **Disassembly** view allows users to view the disassembled code during the stepping of code while debugging. The current executed assembly code will be indicated by an arrow marker and highlighted in the view.

Figure 3-7 Disassembly View


```

Disassembly X Enter location here
+-----+
00000000800006a4: mv    s1,a0
00000000800006a6: csrr  s3,mhartid
00000000800006aa: li    a5,127
00000000800006ae: sext.w s2,s3
00000000800006b2: ld    a0,48(a0)
00000000800006b4: bltu  a5,s2,0x800006da <sbi_init+78>
00000000800006b8: beqz  a0,0x800006da <sbi_init+78>
00000000800006ba: lw    s4,80(a0)
00000000800006be: mv    a1,s2
00000000800006c0: jal   ra,0x80001362 <sbi_platform_hart_index>
00000000800006c4: sext.w a0,a0
00000000800006c6: bgeu  a0,s4,0x800006da <sbi_init+78>
00000000800006ca: ld    a5,32(s1)
00000000800006cc: li    a4,1
00000000800006ce: beq   a5,a4,0x80000866 <sbi_init+474>
00000000800006d2: li    a4,3
00000000800006d4: beq   a5,a4,0x80000872 <sbi_init+486>
00000000800006d8: beqz  a5,0x800006de <sbi_init+82>
00000000800006da: jal   ra,0x800097ba <sbi_hart_hang>
00000000800006de: li    a0,85
00000000800006e2: jal   ra,0x80003c56 <misa_extension_impt>
00000000800006e6: bnez  a0,0x80000872 <sbi_init+486>
00000000800006ea: csrr  s6,mie
00000000800006ee: csrsi mie,8
00000000800006f2: auipc a0,0x18
00000000800006f6: addi  a0,a0,-1754 # 0x80018018 <coldboot_lock>
00000000800006fa: jal   ra,0x800049ae <spin_lock>
00000000800006fe: srai  a5,s2,0x6
0000000080000702: slli  a5,a5,0x3
0000000080000704: auipc a0,0x18
0000000080000708: addi  s4,s4,-1460 # 0x80018150 <coldboot_wait_hmasi>
000000008000070c: add   s4,s4,a5
000000008000070e: ld    a5,0($4)
0000000080000712: li    s5,1

```