

Software Introduction Guide

EIC7x series AI Digital SoC

Engineering Draft / Rev 0.5

2025/08/11

Notice

Copyright © 2024 by Beijing ESWIN Computing Technology Co., Ltd., and its affiliates ("ESWIN"). All rights reserved.

ESWIN retains all intellectual property and proprietary rights in and to this product. Except as expressly authorized by ESWIN, no part of the software may be released, copied, distributed, reproduced, modified, adapted, translated, or created derivative work of, in whole or in part.

INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND DOES NOT REPRESENT A COMMITMENT ON THE PART OF ESWIN. UNLESS OTHERWISE SPECIFIED, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT ARE PROVIDED "AS IS" WITHOUT WARRANTIES, GUARANTEES, OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

"ESWIN" logo, and other ESWIN icons, are trademarks of Beijing ESWIN Computing Technology Co., Ltd. And(or) its parent company.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

If applicable, Open-Source Software and/or free software licensing notices are available in the product installation.

Change History

Version No	Date	Descriptions
Rev 0.3	2024/06/27	Initial version.
Rev 0.4	2025/03/04	Add chapter 6.Proc Debugging Information.
Rev 0.5	2025/08/11	Add feature for the June 30, 2025 version release.

Contents

SOFTWARE INTRODUCTION GUIDE.....	0
EIC7X SERIES AI DIGITAL SoC	0
1. INTRODUCTION.....	3
2. EIC7X SOFTWARE STACK COMPLETE ECOLOGICAL OVERALL INTRODUCTION.....	4
2.1 ORGANIZATION OF THE DEBIAN DISTRIBUTION	4
2.2 EXCLUSIVE TOOLS PROVIDED BY THE EIC7X SOFTWARE STACK	5
3. USE OF EIC7X SOFTWARE STACK	8
3.1 ENVIRONMENT CONSTRUCTION	8
3.2 BURN	9
3.3 SYSTEM COMPILATION.....	9
3.4 HELLO WORLD EXAMPLE PROGRAM.....	11
3.5 USAGE AND PRECAUTIONS OF THE DEDICATED DEB PACKAGE PROVIDED BY ESWIN IN THE EIC7X SOFTWARE STACK	14
3.6 CUSTOMIZED DEVELOPMENT	16
4. CONFIGURE THE REMOTE DEVELOPMENT ENVIRONMENT	17
4.1 DESTINATION	17
4.2 ENVIRONMENTAL STATEMENT	18
4.3 CONFIGURATION PROCESS	18

Content of Tables

Table 1-1 E IC7x software stack matching development board products..... 3

Table 2-1 A dedicated deb package for users in the EIC7x software stack..... 5

Table 2-2 List of tools available to users in the EIC7x software stack (RISC-V tools) 8

Table 2-3 List of tools available to users in the EIC7x software stack (X86 PC side tools) 8

Table 3-1 EIC7x Software Stack user documentation..... 16

Content of Figures

Figure 2-1 EIC7x series product software release and user usage diagram	4
Figure 4-1 VsCode editor interface	19
Figure 4-2 Install the SSH FS plug-in.....	19
Figure 4-3 Example Create an SSH FS connection	20
Figure 4-4 Examples of important parameters of the SSH FS configuration file	20
Figure 4-5 VsCode connects to the remote development board	21
Figure 4-6 VsCode configures the workspace on the remote development board	21

1. Introduction

This manual provides detailed instructions for software development users of EIC7x products on how to use the product packages.

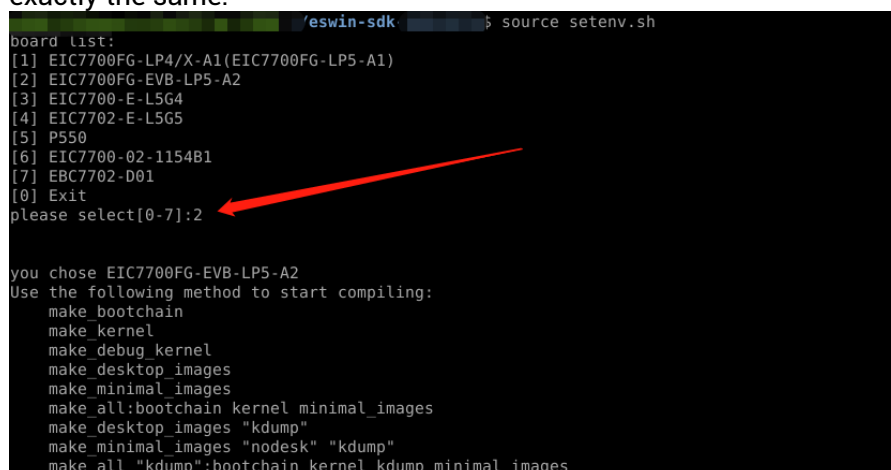
Developers of EIC7x products can customize development based on this document and the corresponding development board hardware, software product packages, and software user documentation. For software customization, please refer to Chapter 3 of this article, especially sections 3.3, 3.4, and 3.5.

The correct operation of software products requires correct hardware configuration. In particular, the hardware connection mode and DIP switch in the development board must refer to the user's instructions for using the development board.

Table 1-1 E IC7x software stack matching development board products

Development board type (new)	Development board type (old)	Board name	Software pallet type configuration name
P550	HF106	SiFive P550 Development board	sifive_dvb_ddr5
EIC7700FG-LP4/X-A1	EIDS100A	EIC7700 LPDDR4X Evaluation board V1.0	EIC7700-evb-a1
EIC7700FG-LP5-A1	EIDS200A	EIC7700 LPDDR5 Evaluation board V1.0	EIC7700-evb-a1
EIC7700FG-EVB-LP5-A2	EIDS200B	EIC7700 LPDDR5 Evaluation board V2.0	EIC7700-evb-a2
EIC7700-E-L5G4		EIC7700 LPDDR5 Evaluation board V3.0	EIC7700-evb-a3
EIC7702-E-L5G5	EIED100A	EIC7702 LPDDR5 Evaluation board V1.0	EIC7702-evb-a1
EIC7700-02-1154B1		EIC7700X LPDDR5 Evaluation board (314pin SOM board+ Bottom plate)	eic7700-d314
EBC7702-D01		EIC7702X LPDDR5 Evaluation board (560 pin SOM board+ Bottom plate)	EIC7702-d560

Note 1. The mapping table is provided here to ensure that the development board is more convenient for customers. When the software is more mature, the board model and configuration file will remain exactly the same.



```

eswin-sdk $ source setenv.sh
board list:
[1] EIC7700FG-LP4/X-A1(EIC7700FG-LP5-A1)
[2] EIC7700FG-EVB-LP5-A2
[3] EIC7700-E-L5G4
[4] EIC7702-E-L5G5
[5] P550
[6] EIC7700-02-1154B1
[7] EBC7702-D01
[0] Exit
please select[0-7]:2

you chose EIC7700FG-EVB-LP5-A2
Use the following method to start compiling:
make_bootchain
make_kernel
make_debug_kernel
make_desktop_images
make_minimal_images
make_all:bootchain kernel minimal_images
make_desktop_images "kdump"
make_minimal_images "nodesk" "kdump"
make_all "kdump":bootchain kernel kdump minimal_images

```

2. EIC7x software stack complete ecological overall introduction

The software stack provided by EIC7x chip includes the following aspects:

/softwares	-----	Contains script files for building minimal and desktop systems.
/pc_tools	-----	Tools for X86 host computers, see Section 2.2.2.
/nn_tools	-----	For the Docker image of the AI toolchain provided to users, refer to Section 2.2.3.
/docs	-----	Software stack documentation, refer to Table 3-1.

The Debian distribution we provide to our users complies with the requirements of the Linux system, as well as the specifications of the standard Debian distribution. For details, please refer to the user documentation for Linux and standard Debian distributions. This article only provides users with the use of software stack, secondary development must use instructions. For secondary development related to the software stack of this product, please refer to the documents listed in Chapter 3 of this article.

2.1 Organization of the Debian distribution

The software stack accompanying the EIC7x chip, including the Debian distribution, and various tools running on the pc side (including tools running on the windows platform on the X86 platform and tools running on the linux platform). The Debian distribution, which is available directly to users, is a minimal distribution that can be installed directly over the network. You can also use the default scripts in the software packages to obtain ESWIN software packages for desktop applications, such as GUIs and browsers, from the network.

On the basis of minimal Debian distribution, users can obtain the required software packages from the Debian software warehouse maintained by PLCT and the exclusive application warehouse provided by Yieswei through scripts according to their own needs. It is also possible to develop in a custom way based on the minimum Debian distribution.

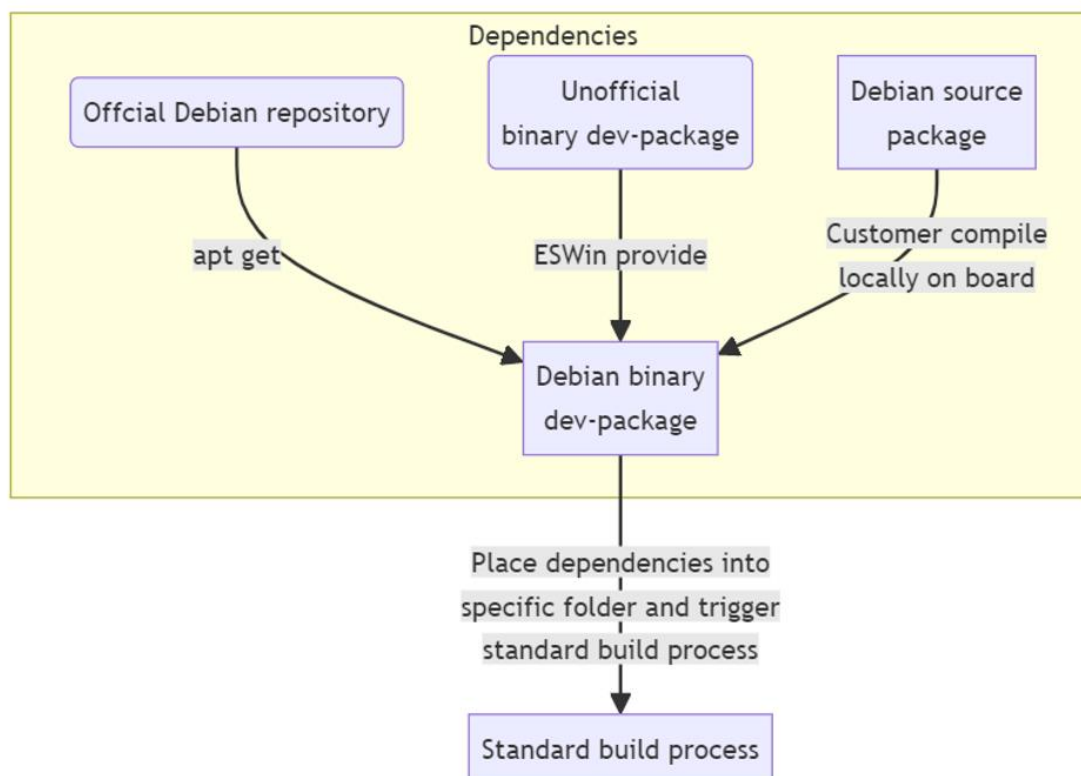


Figure 2-1 EIC7x series product software release and user usage diagram

The 06/30 release in 2025 provides users with two distribution images, minimal and desktop.

Note 2. The default system account/password for minimal and desktop versions is eswin/eswin

In addition, the following is an example of how to view the versions of boot.ext4 and root.ext4:

```
eswin@rockos-eswin:~$ uname -a
Linux rockos-eswin 6.6.18-eic7x-2024.11 #2024.11.21.02.05+ SMP Thu Nov 21 02:16:13 UTC
2024 riscv64 GNU/Linux

eswin@rockos-eswin:~$ cat /etc/*release
EIDS200B:EIC7X-2024.11
```

2.2 Exclusive tools provided by the EIC7x software stack

2.2.1 Dedicated deb packages provided by Eswin

The ESWIN EIC7x software stack provides users with dedicated deb packages listed in Table 21. To properly use these user-provided packages, please follow the order in Chapter 3 of this article and learn the specific precautions for the use of each package in detail in Section 3.5 (if not provided, no special attention is required).

Table 2-1 A dedicated deb package for users in the EIC7x software stack

Name of the dedicated deb package	Application description	Matters needing attention
es-sdk-log_1.8.0_riscv64.deb	ESSDK log module, the log output library provided to each module	
es-sdk-memory_1.8.0_riscv64.deb	General memory and vb memory allocation management interface	<ul style="list-style-type: none"> To allocate memory in a memory region (for example, mmz_nid_0_part_0), ensure that the relevant memory region is described in the dts before memory allocation.
es-sdk-memcp_1.8.0_riscv64.deb	EsSDK Memcp	
es-sdk-cipher_1.8.0_riscv64.deb	ESSDK Cipher	
es-sdk-numa_1.8.0_riscv64.deb	ESWIN NUMA	
es-sdk-common_1.8.0_riscv64.deb	Public header file of ESSDK	
es-sdk-video-utils_1.8.0_riscv64.deb	A public tool library used inside the Video module in ESSDK	

es-sdk-sys_1.8.0_riscv64.deb	ESSDK SYS	<ul style="list-style-type: none"> After installation, es_media_srv automatically runs in the background and does not need to be started again. After restarting, es_media_srv does not run automatically. If an application calls the ES_SYS_Init interface, it will run with es_media_srv pulled up; otherwise, it does not need es_media_srv.
es-sdk-video_1.8.0_riscv64.deb	ESSDK Video, include vdec、venc、vps、vo	
es-hae_1.8.0_riscv64.deb	ESWIN image processing hardware acceleration engine	
es-video-common_1.8.0_riscv64.deb	ESWIN Video public library	
es-mpp_1.8.0_riscv64.deb	ESWIN multimedia processing platform	
es-sdk-npu_1.8.0_riscv64.deb	ESWIN NPU Intelligent reasoning module, which contains NPU infrastructure libraries and files, including the NPU Runtime library and E31 firmware	
es-sdk-dsp_1.8.0_riscv64.deb	ESWIN DSP intelligent reasoning module, including the NPU infrastructure library and files including DSP Runtime library and DSP firmware	
es-sdk-audio_1.8.0_riscv64.deb	ESSDK Audio module, including various components of audio processing, AI, AO, AENC, ADEC and other modules	
es-sdk-audio-codec_1.8.0_riscv64.deb	ESSDK Audio codec library, including AAC-LC/HEv1/v2, AMR-Nb/Wb, MP3, G7XX, MP2L2 decoder and AAC-LC encoder	
es-sdk-ak_1.8.0_riscv64.deb	AcceleratorKit is an API library for calling operators that are not compiled into the ONNX model, such as some of the more complex pre - and post-processing operators	
ffmpeg.deb	ffmpeg7.0 installation package, which integrates the hardware codec plug-in implemented by eswin	
es-sdk-sample-npu-runtime_1.8.0_riscv64.deb	ESSDK NPU Runtime SDK interface calls Sample	
es-sdk-sample-audio_1.8.0_riscv64.deb	ESSDK Audio SDK interface calls Sample	

es-sdk-sample-video_1.8.0_riscv64.deb	ESSDK video sample	
es-sdk-sample-memory_1.8.0_riscv64.deb	Example of using the es-sdk-memory package	<ul style="list-style-type: none"> ● The spram region is used in the sample program. If this region is occupied by the npu program, the related case will fail to be executed. ● The IOVA-related interface depends on the driver of the corresponding module. Make sure that the corresponding driver is loaded before calling.
es-sdk-sample-dsp_1.8.0_riscv64.deb	Example of the es-sdk-dsp package	
es-sdk-sample-cipher_1.8.0_riscv64.deb	es-sdk-cipher is an example of the package	
es-sdk-sample-npu-qwen_1.8.0_riscv64.deb	ESSDK NPU Qwen model inference	
es-sdk-sample-memcp_1.8.0_riscv64.deb	Example of the es-sdk-memcp package	
es-fand_1.8.0_riscv64.deb	Fan control module	
escl-server_1.8.0_riscv64.deb	Provides the functionality for the ESWIN Multimedia Processing Platform (mpp) to call vendor interfaces	
es-hw-watcher_1.8.0_riscv64.deb	Dynamically acquires device hardware information (temperature, voltage, utilization rate, bandwidth, etc. - refer to help or ReadMe for specifics) and displays it in real-time. Supported devices: NPU, VDEC, VENC	
es-lightdm-numactl_1.8.0_riscv64.deb	Desktop display optimization: Improves performance of the display module	<ul style="list-style-type: none"> ● Optimization applies exclusively to multi-NUMA platforms and must not be installed on single-NUMA platforms
es-numa-manager_1.8.0_riscv64.deb	Desktop system optimization: Enhances foreground window performance	
es-performance_1.8.0_riscv64.deb	Interface and tool performance optimization: Enhances efficiency and responsiveness	

2.2.2 List of tools used on the X86 platform

The ESWIN EIC7x software stack, on the X86 platform, provides users with the following tools, which are used in the linux environment.

Table 2-2 List of tools available to users in the EIC7x software stack (RISC-V tools)

Tool name	Tool use	Remark
Nsign tool	Nsign is an image signing tool that ensures the security and integrity of the image to be loaded. It includes the following functions: abstract calculation, symmetric encryption, asymmetric encryption, multilevel mirror signature packaging and asymmetric encryption key generation.	For details, see Image Creation and EsNsign User Manual. Tool source github path: https://github.com/eswincomputing/Esbd-77serial-nsign

2.2.3 A separate Docker image for the user

The ESWIN EIC7x software stack provides users with three docker images. These three images are provided separately to the user because of their large capacity.

Table 2-3 List of tools available to users in the EIC7x software stack (X86 PC side tools)

Software kit name	Tool use	Remark
esquant_docker.tar	EsQuant Docker image	For details, please refer to the ENNP User Manual.
esaac_essimulator_docker.tar	EsAAC and EsSimulator Docker image	For details, please refer to the ENNP User Manual.
es_debian_compile_docker.tar	Docker image for compiling debian environment	For details, see the next chapter of this document
esquant-1.0-py3-none-any.whl	EsQuant tool python installation package	For details, please refer to the ENNP User Manual.

Note 3. ESWIN provides standard software release images for the EIC7x series products (released through public channels). Due to the large size of the Docker image, it is not directly released to individual users. If necessary, please contact the sales channel.

3. Use of EIC7x software stack

3.1 Environment Construction

To prepare a ubuntu22.04 machine or ubuntu20.04, you need to have access to github.com on your network configuration.

The following software needs to be installed if the ubuntu22.04 OS is used:

```
sudo apt install -y gdisk dosfstools build-essential libncurses-dev gawk flex bison openssl
libssl-dev tree dkms libelf-dev libudev-dev libpci-dev libiberty-dev autoconf device-tree-
compiler xz-utils devscripts ccache debhelper wget curl pahole libconfuse-dev mtools
fastboot rsync
```

```
sudo apt install -y debootstrap devscripts qemu-user-static qemu-utils binfmt-support
mmdebstrap
```

```
wget https://github.com/riscv-collab/riscv-gnu-
toolchain/releases/download/2024.04.12/riscv64-glibc-ubuntu-22.04-gcc-nightly-
2024.04.12-nightly.tar.gz
```

```
sudo tar -xvf riscv64-glibc-ubuntu-22.04-gcc-nightly-2024.04.12-nightly.tar.gz -C /opt
```

If ubuntu20.04 is used, you can directly compile using docker, upload es_debian_compile_docker.tar to the ubuntu20.04 environment, and load the docker image:

```
sudo docker load -i es_debian_compile_docker.tar
sudo docker pull multiarch/qemu-user-static
sudo docker run --rm --privileged multiarch/qemu-user-static --reset -p yes
```

3.2 Burn

After the user has completed the basic environment, they can refer to Section 3.3 for custom development. When the custom development is compiled, the user needs to write the compiled results to the development board.

In the development process, the user involves repeatedly executing the process of "developing compiling burning and writing", so the "burning and writing" process is advanced. The way the user burns the board varies according to the product type, including: the connection mode of the board and the computer, the connection mode and port of different peripherals, the setting of the dip switch in the board or the configuration mode of the jumper. In order to ensure that the burning process is not wrong, the user needs to follow the model of the purchased product, according to the "Download burning" section of the "Development Board Starter Guide" document provided for users in different board types of development board products.

Note 4. When the user is writing, the connection configuration between the board and the host computer of the X86 platform, and the connection mode of the dip switch and jumper on the board must be carried out in accordance with the corresponding instructions of the board. Incorrect connection may cause unpredictable damage to the board, affecting your use.

3.3 System compilation

Upload the eswin-sdk-*.tar.gz package from the release to ubuntu22 or 20 and decompress it:

```
tar -xf eswin-sdk-*.tar.gz -C $HOME
```

You can view the source directory and setenv.sh script. The source directory stores the source code of linux, opensbi, and uboot downloaded from github during the compilation process. The source directory will automatically download the source code during the compilation process. You can download the source code in advance by the following command.

Note 5. In the following command, replace x in EIC7X-2025.x with the current release version, for example, replace 0830 with 08, 1030 with 10, 1130 with 11, and so on.

```
git clone --depth=1 -b EIC7X-2025.06 https://github.com/eswincomputing/u-boot.git
source/u-boot-eswin
git clone --depth=1 -b EIC7X-2025.06 https://github.com/eswincomputing/opensbi.git
source/opensbi-eswin
git clone --depth=1 -b EIC7X-2025.06 https://github.com/eswincomputing/linux-stable.git
source/linux-eswin
```

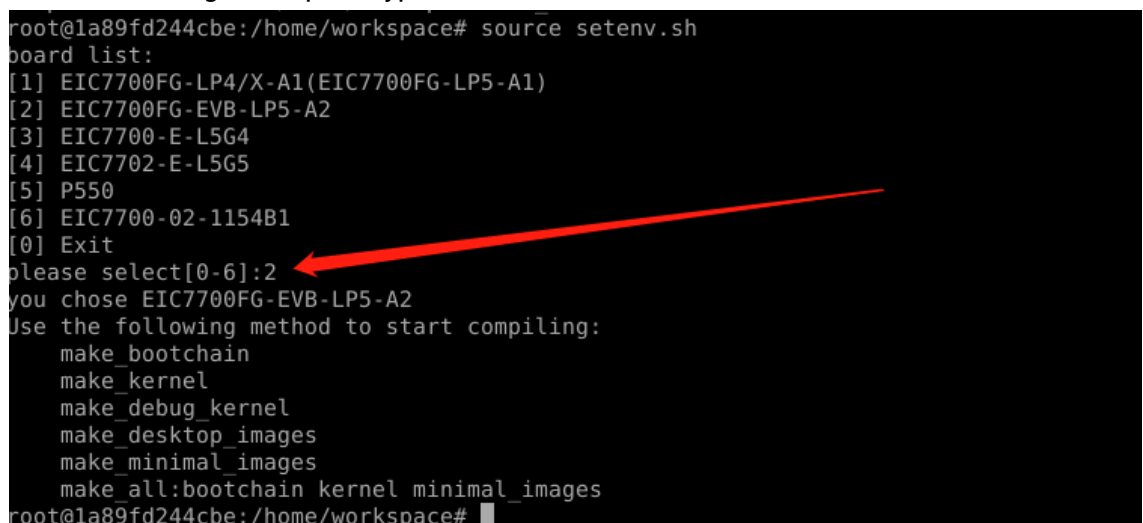
In the Ubuntu20.04 environment, run the following command in the decompressed directory to start the docker container

```
sudo docker run --rm --privileged -it -v "$(pwd)":/home/workspace -u root -w
"/home/workspace" es_debian
```

Go to the decompressed directory and run the following command:

```
source setenv.sh
```

Select according to the plate type:




```
root@1a89fd244cbe:/home/workspace# source setenv.sh
board list:
[1] EIC7700FG-LP4/X-A1(EIC7700FG-LP5-A1)
[2] EIC7700FG-EVB-LP5-A2
[3] EIC7700-E-L5G4
[4] EIC7702-E-L5G5
[5] P550
[6] EIC7700-02-1154B1
[0] Exit
please select[0-6]:2
you chose EIC7700FG-EVB-LP5-A2
Use the following method to start compiling:
make_bootchain
make_kernel
make_debug_kernel
make_desktop_images
make_minimal_images
make_all:bootchain kernel minimal_images
root@1a89fd244cbe:/home/workspace#
```

- ◆ make_bootchain: build bootloader
- ◆ make_kernel: The generated deb is used when compiling the kernel, generating the deb package, and compiling root.ext4 and boot.ext4
- ◆ make_minimal_images: Generate a compact version of root.ext4(1000M) and boot.ext4(100M)
- ◆ make_desktop_images: Generate a root.ext4(7G) and boot.ext4(500M) containing desktop
- ◆ make_all: Build the bootloader, and the lite version of the system

For example, you need to compile a stripped-down version of your system quickly:

```
make_all
```

At the end of the compilation process, the process of making images requires sudo permission and the password of sudo needs to be entered once. When docker images are used, the password will not be prompted:



```
start compile debian mking
[sudo] 的密码:
```

After the compilation is complete, the generated bootloader.bin root.ext4 boot.ext4 is displayed in the current directory:

```

drwxr-xr-x 2 root root      4096 Apr 21 05:44 ./
drwxr-xr-x 8 root root      4096 Apr 21 03:13 ../
-rw-r--r-- 1 root root 104857600 Apr 21 02:59 boot-P550-20250421-025126.ext4
-rw-r--r-- 1 root root 4306584 Apr 21 02:49 bootloader_P550.bin
-rwxr-xr-x 1 root root 4030200 Apr 21 02:49 fw_payload.bin*
-rw-r--r-- 1 root root 8603052 Apr 21 02:51 linux-headers-6.6.18-eic7x-2025.03_6.6.18-2025.04.21.02.49+_riscv64.deb
-rw-r--r-- 1 root root 22371792 Apr 21 02:51 linux-image-6.6.18-eic7x-2025.03_6.6.18-2025.04.21.02.49+_riscv64.deb
-rw-r--r-- 1 root root 1336454 Apr 21 02:51 linux-libc-dev_6.6.18-2025.04.21.02.49+_riscv64.deb
-rw-r--r-- 1 root root 5631544 Apr 21 02:49 recovery_bootloader_P550.bin
-rw-r--r-- 1 root root 104857600 Apr 21 02:59 root-P550-20250421-025126.ext4
-rw-r--r-- 1 root root 1933042 Apr 21 02:48 u-boot.bin
-rw-r--r-- 1 root root 42418 Apr 21 02:48 u-boot.dtb

```

3.4 Hello world Example Program

According to the Environment construction section, the cross-compilation tool chain is stored in the /opt/riscv directory. After source setenv.sh is executed, the following environment variables have been added to the system:

```

export PATH="/opt/riscv/bin:$PATH"
export ARCH=riscv
export CROSS_COMPILE=riscv64-unknown-linux-gnu-

```

This section guides you through compiling a simple hello world program using the cross-compile toolchain, and how to add three-party packages to the toolchain.

3.4.1 Cross compilation

Create a new directory hello, and in this directory create a new file hello.c and Makefile.

hello.c

```

#include <yaml.h>
int main() {
    printf("Hello, world!\n");
    return 0;
}

```

Makefile

```

CC = $(CROSS_COMPILE)gcc
CFLAGS = -Wall
TARGET = hello
SRC = hello.c
OBJ = $(SRC:.c=.o)

all: $(TARGET)

$(TARGET): $(OBJ)
    $(CC) $(CFLAGS) -o $@ $^

%.o: %.c
    $(CC) $(CFLAGS) -c -o $@ $<

clean:
    rm -f $(OBJ) $(TARGET)

```

Perform cross-compilation

```
make
```

After completing the cross-compilation, the executable program hello is generated, and the hello program is copied to the burned system for testing:

```

:~$ ./hello
Hello, world!

```

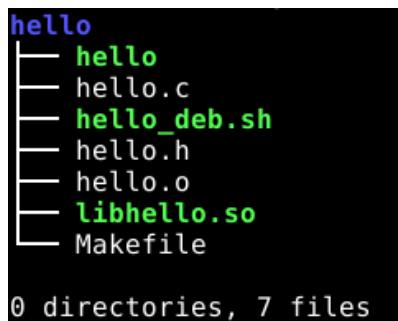
3.4.2 Build deb packages

When the compiled program includes lib, bin, include and other files, and needs to be installed in different directories of the system, the program can be built into a deb package, taking the hello word program as an example, if the compiled hello program, There are executable hello, header hello.h, lib file libhello.so, requiring hello to be installed in /usr/bin, libhello.so to /usr/lib, hello.h to /usr/include/, The following process is used to build the deb package:

1. Complete cross-compilation
2. Edit the deb package build script

```
mkdir -p debian/work/DEBIAN
cd debian/work
mkdir -p usr/lib
mkdir -p usr/bin
mkdir -p usr/include
cp ../../hello usr/bin
cp ../../libhello.so usr/lib
cp ../../hello.h usr/include
echo Package: hello >>DEBIAN/control
echo Version: 1.0 >>DEBIAN/control
echo Section: utils >>DEBIAN/control
echo Priority: optional >>DEBIAN/control
echo Architecture: riscv64 >>DEBIAN/control
echo Depends: >>DEBIAN/control
echo Installed-Size: 10240 >>DEBIAN/control
echo Maintainer: robot@eswincomputing.com >>DEBIAN/control
echo Description: essdk hello >>DEBIAN/control
dpkg -b ../../hello-V1.0.deb
```

Create hello_deb.sh, save the above content to hello_deb.sh, and give the executable permission.



```
hello
├── hello
├── hello.c
├── hello_deb.sh
├── hello.h
├── hello.o
├── libhello.so
└── Makefile

0 directories, 7 files
```

3. Generate the deb package

```
./hello_deb.sh
```

The generated hello-V1.0.deb is displayed in the current directory.

4. Install deb

Upload hello-V1.0.deb to the target system for installation:

```
sudo dpkg -i --force-all hello-V1.0.deb
```

5. Check whether the deb deployment is as expected


```

~$ ls /usr/bin/ | grep hello
hello
~$ ls /usr/lib/ | grep hello
libhello.so
~$ ls /usr/include/ | grep hello
hello.h
~$ hello
Hello, world!
~$

```

3.4.3 Cross-compile toolchain extensions

While cross-compiling some programs, some open source packages may not be found, so you need to add these open source packages in the cross-compilation tool chain. Take the hello program as an example, a new reference to libyaml will be added, and the following error will occur when cross-compiling:

```

~/hello$ make
riscv64-unknown-linux-gnu-gcc -Wall -c -o hello.o hello.c
hello.c:2:10: fatal error: yaml.h: No such file or directory
  2 | #include <yaml.h>
    |           ^~~~~~
compilation terminated.
make: *** [Makefile:13: hello.o]

```

libyaml can be added to the cross-compile toolchain by performing the following steps:

1. Back up sysroot of the cross-compilation toolchain

```
sudo mv /opt/riscv/sysroot/ /opt/riscv/sysroot_bak
```

2. Mount the compiled root.ext4

```
mkdir rootfs
sudo mount root.ext4 rootfs/
```

To create a rootfs directory, mount root.ext4 to the rootfs directory.

minimal root.ext4 space is small, apt install process may prompt space issues, you can use the resize2fs (sudo apt-get install e2fsprogs) command on ubuntu22.04 before this step to adjust, for example:

```
sudo resize2fs root.ext4 7G
```

1. Specify a new sysroot directory

Make a new sysroot directory and soft connect the created rootfs directory to /opt/riscv/sysroot

```
sudo ln -s /x/x/x/x/rootfs/ /opt/riscv/sysroot
```

2. chroot install libyaml

```

sudo chroot rootfs/
apt install gcc
apt install build-essential
cd /usr/include
ln -s riscv64-linux-gnu/bits bits
ln -s riscv64-linux-gnu/sys sys
ln -s riscv64-linux-gnu/gnu gnu
ln -s riscv64-linux-gnu/asm asm
cd /usr/lib
cp riscv64-linux-gnu/* ./ -rf

```

The chroot then enters a virtual system where apt finds and installs the required packages, such as

libyaml, that support development.

```
apt search libyaml
```

```
root@riscv64-unknown-linux-gnu:/# apt search libyaml
erlang-pl-yaml/sid 1.0.36-2 riscv64
  erlang wrapper for libyaml C library

libghc-libyaml-dev/sid,now 0.1.2-3+b3 riscv64 [installed]
  low-level, streaming YAML interface.

libghc-libyaml-doc/sid 0.1.2-3 all
  low-level, streaming YAML interface.; documentation

libghc-libyaml-prof/sid 0.1.2-3+b3 riscv64
  low-level, streaming YAML interface.; profiling libraries

libghc-yaml-dev/sid 0.11.11.2-1+b3 riscv64
  interface to LibYAML

libghc-yaml-doc/sid 0.11.11.2-1 all
  interface to LibYAML; documentation

libghc-yaml-prof/sid 0.11.11.2-1+b3 riscv64
  interface to LibYAML; profiling libraries

librust-unsafe-libyaml-dev/sid 0.2.11-1 riscv64
  Libyaml transpiled to rust by c2rust - Rust source code
```

Find the corresponding development package, and the header file with the suffix "-dev" will be installed:

```
apt install libghc-yaml-dev
```

After the installation is complete, the cross-compilation tool chain adds the lib, include, etc. of the libyaml, returns to the cross-compilation environment, and executes the compilation again:

```
riscv64-unknown-linux-gnu-gcc -Wall -o hello hello.o /hello$ make
```

3.5 Usage and precautions of the dedicated Deb package provided by ESWIN in the EIC7x software stack

3.5.1 Installing the Deb Package

ESWIN provides Debian packages listed in Table 2-1 for online installation through debian package sources. The installation method is as follows:

```
sudo apt update
sudo apt install es-sdk-log
sudo apt install es-sdk-memory
sudo apt install es-sdk-memcp
sudo apt install es-sdk-cipher
sudo apt install es-sdk-numa
sudo apt install es-sdk-common
sudo apt install es-sdk-video-utils
sudo apt install es-sdk-sys
sudo apt install es-sdk-video
sudo apt install es-hae
sudo apt install es-video-common
sudo apt install es-mpp
sudo apt install es-sdk-npu
sudo apt install es-sdk-dsp
sudo apt install es-sdk-audio
sudo apt install es-sdk-audio-codec
sudo apt install es-sdk-ak
sudo apt install ffmpeg
sudo apt install es-sdk-sample-npu-runtime
sudo apt install es-sdk-sample-dsp
sudo apt install es-sdk-sample-audio
sudo apt install es-sdk-sample-video
sudo apt install es-sdk-sample-memory
sudo apt install es-sdk-sample-cipher
sudo apt install es-sdk-sample-memcp
sudo apt install es-sdk-sample-npu-qwen
sudo apt install es-fand
```

If space is insufficient during the installation, run the following command to adjust the file system partition size:

```
sudo resize2fs /dev/mmcblk0p3
```

Note 6: If the offline deb installation package is provided in a specific release, as shown in Table 2-1, the user can choose to install the DEB offline. The specific online installation or offline installation can be selected by the customer according to the situation. The following describes the offline installation methods:

```
sudo apt install *.deb
```

3.6 Customized development

The software stack of the EIC7x series chip provides users with a complete custom development environment and resources from the bottom layer to the application. Users can customize development based on product types, scenarios, and resources provided by the software stack.

If customers need customized development for upper-layer applications, they can directly use the resources of ESSDK sample provided for users for secondary development. If customers need to start customized development from the bottom, they can obtain the corresponding source code based on the software we publish, from the official channels, and conduct secondary development in combination with the user's own business needs.

To customize the development process and method, refer to Section 3.4 "Hello world Sample Program" of this article for the development and compilation of specific software packages. The customized development of a complete image is completed by combining the previous chapters in this chapter and the hardware usage guide of the development board.

The software stack of EIC7x series products includes system, audio and video, intelligence and other aspects. The documents contained in these aspects are shown in Table 31 below. Users can directly refer to the contents of the corresponding documents during the customization development process.

Table 3-1 EIC7x Software Stack user documentation

sort	Document name	Remark
System	Cipher user manual	Cipher is a security algorithm module provided by the EIC7x SOC platform. It provides symmetric encryption and decryption algorithms such as AES, DES/3DES and SM4, asymmetric encryption and decryption algorithms such as RSA and ECSDA, random number generation, and supports HASH, HMAC, MAC and other abstract algorithms. It is used to encrypt and decrypt audio and video streams and authenticate users.
System	Memory user manual	The Memory module provides bulk physical memory management for media services, allocates and reclaims memory cache pools and memory blocks, and ensures proper use of physical memory resources in media processing modules.
System	Memory_Copy user manual	MemCP (Memory Copy) module provides a unified memory copy and command queue management interface for systems and applications. Through this module, users can realize efficient asynchronous data transmission and task scheduling, and support efficient execution of complex memory operations in multi-task concurrent environment. The MemCP module kernel implements command queue (CMDQ) and memory replication (Memcpy) functions, and provides a consistent and easy to use user-level interface.
System	SYS Development Manual	The SYS module provides basic public services for ESSDK, including system management, version management, memory management, log management, time management, and binding management.
System	Bootloader migration application development guide	The EIC7x Bootloader mainly consists of U-Boot and opensbi. Users only need to pay attention to U-Boot during development and migration. This manual describes U-Boot download, compilation, porting, opensbi download and compilation, and common U-Boot commands.

System	Development board image installation and upgrade manual	This manual describes how to burn and upgrade the system image methods and operating processes.
System	Image creation and EsNsign user manual	This document describes how to use the Nsign tool to package system images.
System	Power regulation guide for software service scenarios	This document describes how to adjust the power consumption of the EIC7x series development board at the software level.
Audio	Audio user manual	Audio module includes four sub-modules: audio input, audio output, audio encoding and audio decoding. The audio input and output module realizes the audio input and output function by controlling the audio interface of EIC7x chip. Audio encoding and decoding module provides audio codec functions for G711, G722, G726, AAC, MP3, AMR and other formats, and supports recording and playback of original audio files in LPCM format.
Audio	Audio development manual	Audio module includes four sub-modules: audio input, audio output, audio encoding and audio decoding. The audio input and output module realizes the audio input and output function by controlling the audio interface of EIC7x chip. Audio encoding and decoding module provides audio codec functions for G711, G722, G726, AAC, MP3, AMR and other formats, and supports recording and playback of original audio files in LPCM format.
Video	VPS development Manual	Video process system (VPS) is a video image processing system that provides image processing functions in time-sharing multiplexing mode. Including fisheye correction, deformity correction, cropping, scaling, color space conversion, fixed Angle rotation, MIRROR, FLIP, line drawing, solid color fill, normalization, frame rate control, OVERLAY, shape ratio and other functions.
Video	VENC Development Manual	VENC module, that is, video coding module.
Video	VDEC Development Manual	The VDEC module is responsible for decompressing the digital compressed video in H.264/265 and JPEG format, so as to obtain the YUV/RGB format image stream. This module supports multiple decoding.
Video	VO Development Manual	The VO module actively reads the video data from the memory response location and outputs the video through the display device.
Audio \ Video	FFMPEG User Manual	FFMPEG plug-in mainly includes codec plug-in usage methods, as well as Mpv and obs simple use process.
ENNP	ENNP User Manual	The ENNP (ESWIN NetWork Processing) platform is an intelligent computing heterogeneous acceleration platform for EIC7x SOC chips, including development tools and interfaces for NPU, DSP, HAE, and GPU.

4. Configure the remote development environment

4.1 Destination

The main purpose of configuring the remote development environment is to facilitate developers to use the computing resources and environment of the remote development board (EIC7x series) on the local computer for application development, code compilation, and project deployment.

4.2 Environmental statement

Select the "VsCode + plug-in" method, in theory, the development machine that can install VsCode software can be used as the development host of the local environment. The remote environment, the EIC7x series development board, includes all models listed in Table 1-1. This configuration solution uses the following development environment as an example to describe the process of setting up a remote development environment.

Local environment: ubuntu20.04 system development host.

Remote environment: EIC7x series development board.

4.3 Configuration Process

4.3.1 Installing VsCode

(1) First, the apt command updates the package index and installs the dependent software that imports the Microsoft GPG key.

```
sudo apt update
```

(2) Installation kit.

```
sudo apt install software-properties-common apt-transport-https curl
```

(3) After importing GPG, run the curl command to import the Microsoft GPG key.

```
curl -sSL https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
```

(4) Add vscode software sources to the Ubuntu 22.04 software sources list.

```
sudo add-apt-repository "deb [arch=amd64]  
https://packages.microsoft.com/repos/vscode stable main"
```

(5) Install vscode

```
sudo apt update  
sudo apt install code
```

(6) Start Vscode

Enter code in the terminal and press enter to start the VsCode editor page, as shown in Figure 4-1 below.

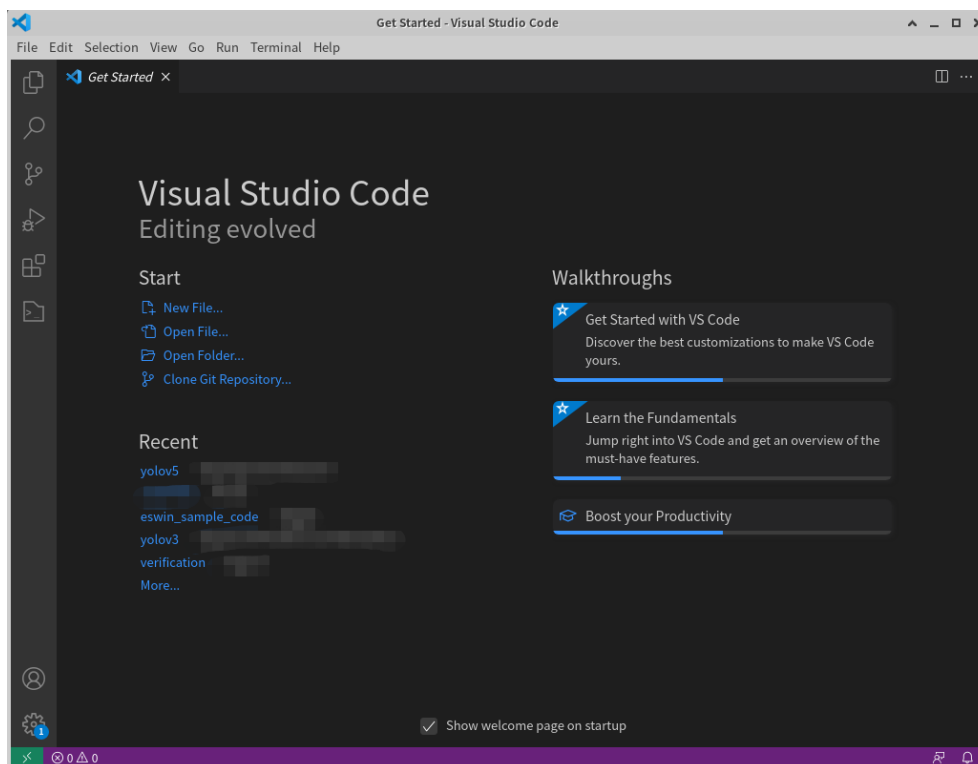


Figure 4-1 VsCode editor interface

4.3.2 Installing the SSH FS Plug-in

Install the SSH FS plug-in, as shown in Figure 4-2:

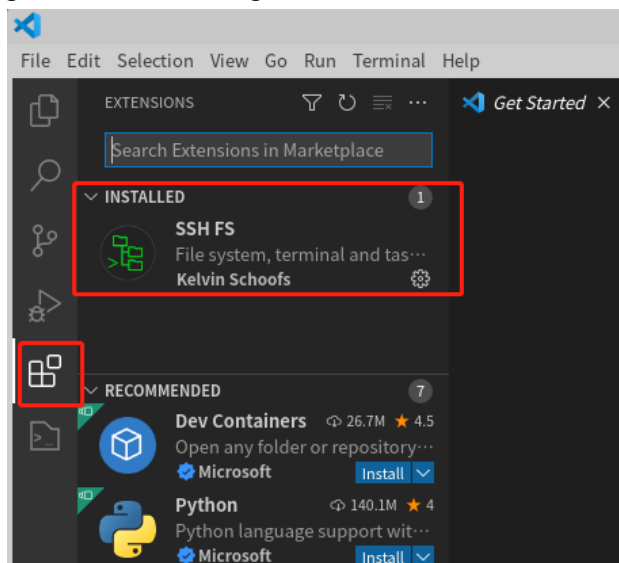


Figure 4-2 Install the SSH FS plug-in

4.3.3 Connection Configuration

Click Create a SSH FS Configuration under CONFIGURATIONS to create a new connection, set a name, and save. As shown in Figure 4-3:

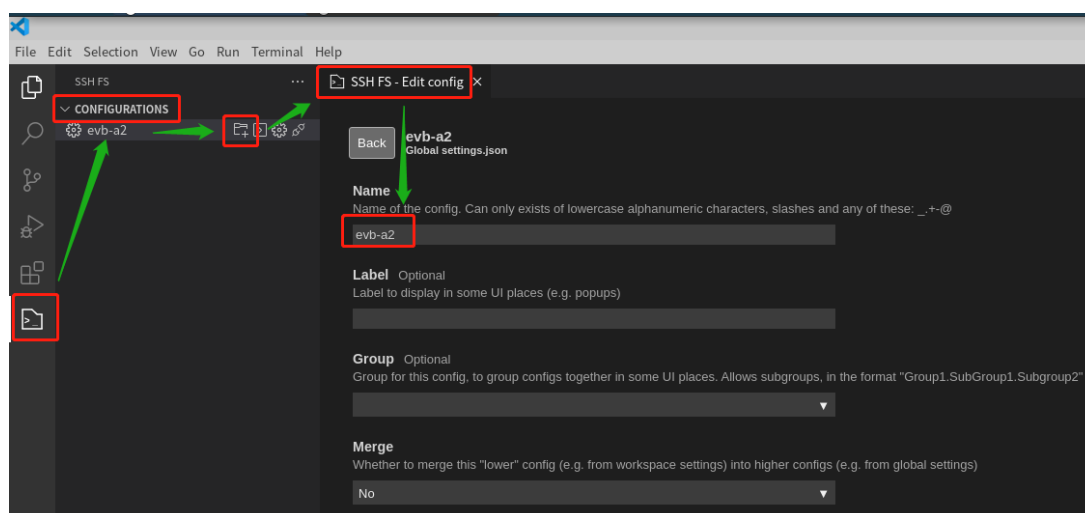


Figure 4-3 Example Create an SSH FS connection

Figure 4-4 shows the parameters that must be set in the configuration file:

Figure 4-4 Examples of important parameters of the SSH FS configuration file

As shown in Figure 4 above, where:

Host: Enter the ip address of the EIC7x development board.

Root: Enter the workspace path displayed on the terminal after remotely logging in to the development board.

Username: Development board login user name.

Password: Password for logging in to the development board.

4.3.4 SSH Connection

After the connection configuration is created, click the ">" control in the VsCode interface through the following figure 4-5, and log in to the remote development board environment directly on the VsCode terminal through SSH, and then remote development can be carried out in this terminal.

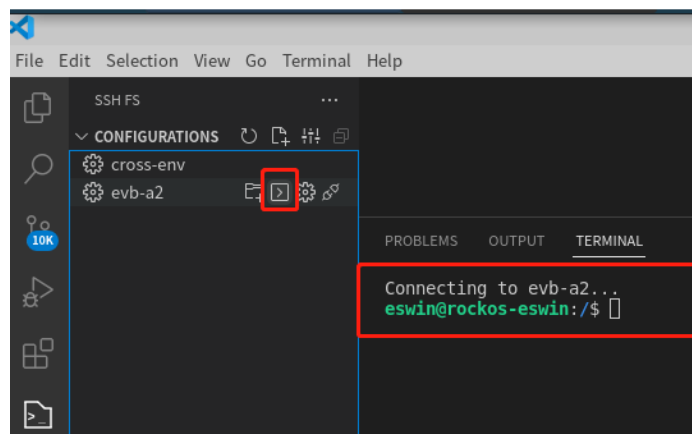


Figure 4-5 VsCode connects to the remote development board

4.3.5 Adding a Working Directory

The "workspace" workspace can also be configured in the VsCode software interface, so that developers can write, modify, and compile code projects on remote development boards locally. Figure 4-6 shows the process for adding a working directory:

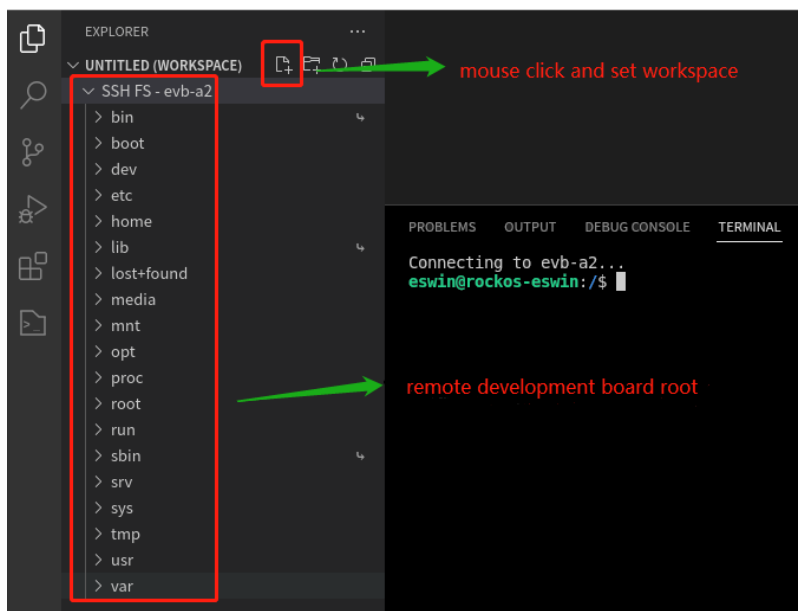


Figure 4-6 VsCode configures the workspace on the remote development board