

NUC980 Linux Environment on VMware User Manual

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 OVERVIEW	5
1.1 Development Environment.....	5
2 INSTALLATION OF LINUX ENVIROMENT.....	6
2.1 Installing VMware Virtual Machine.....	6
2.2 Downloading and Opening Linux Environment Image	9
3 BUILDING IMAGES OF LINUX KERNEL.....	12
3.1 For Version 4.4 of Linux Kernel.....	12
3.1.1 Getting the Buildroot	12
3.1.2 Compiling Linux Kernel and U-boot.....	12
3.1.3 Updating Source Code	14
3.2 For Version 5.10 of Linux Kernel.....	15
3.2.1 Getting the Buildroot	15
3.2.2 Compiling Linux Kernel and U-boot.....	16
3.2.3 Updating Source Code	17
4 UPDATING TOOLCHAIN OF THE ENVIRONMENT VARIABLES	19
5 LINUX USER APPLICATIONS	22
5.1 Example of UART	22
5.2 Example of Using LCD Panel.....	24
5.2.1 Settings of LCD Panel	24
5.2.2 Demo of HMI Example.....	35
6 REVISION HISTORY	37

List of Figures

Figure 1-1 Development Environment Setup	5
Figure 2-1 Download VMware Workstation Player	6
Figure 2-2 Download VMware Workstation Player for Windows	6
Figure 2-3 Install VMware Workstation Player (1)	7
Figure 2-4 Install VMware Workstation Player (2)	7
Figure 2-5 Install VMware Workstation Player (3)	7
Figure 2-6 Install VMware Workstation Player (4)	8
Figure 2-7 Install VMware Workstation Player (5)	8
Figure 2-8 Menu of Virtual Machine	8
Figure 2-9 Installation Errors of VMware	9
Figure 2-10 BIOS Settings	9
Figure 2-11 Open the Linux Environment (1).....	10
Figure 2-12 Open the Linux Environment (2).....	10
Figure 2-13 Open the Linux Environment (3).....	10
Figure 2-14 Open the Linux Environment (4).....	11
Figure 2-15 Login Linux Virtual Machine	11
Figure 2-16 Setting of Time Zone	11
Figure 3-1 NUC970_Buildroot-master Folder	12
Figure 3-2 Configuration File.....	12
Figure 3-3 Configuration Setting and Compiling	13
Figure 3-4 Location of Image and Uimage File.....	13
Figure 3-5 Location of U-boot.bin File.....	13
Figure 3-6 Location of U-Boot-spl.bin File	14
Figure 3-7 Location of Linux-master and Uboot-master File	14
Figure 3-8 Location of Linux-master.gz and Uboot-master.gz File.....	14
Figure 3-9 Buildroot Update	15
Figure 3-10 MA35D1_Buildroot Folder	15
Figure 3-11 Configuration File.....	16
Figure 3-12 Configuration Setting and Compiling	16
Figure 3-13 Location of Image, Uimage, and Dtb File	17
Figure 3-14 Location of U-boot.bin File.....	17
Figure 3-15 Location of U-boot-spl.bin File.....	17
Figure 3-16 Location of Linux-custom and Uboot-custom File	18
Figure 3-17 Location of Linux-master.gz File.....	18
Figure 3-18 Location of Uboot-master.gz File.....	18
Figure 3-19 Updating the Buildroot	18

Figure 4-1 Errors of Making File (Linux Kernel 4.4)	19
Figure 4-2 After Modified the Environment Variables of Linux Kernel 4.4	19
Figure 4-3 Making File After Modifying the Environment Variables of Linux Kernel 4.4	19
Figure 4-4 Modifying the Environment Variables of Linux Kernel 5.10	20
Figure 4-5 After Modified the Environment Variables of Linux Kernel 5.10	20
Figure 4-6 Location of Linux Kernel 4.4 Toolchains	20
Figure 4-7 Location of Linux Kernel 5.10 Toolchains	21
Figure 5-1 UART Application Folder	22
Figure 5-2 After Compilation	22
Figure 5-3 Location of Executable File	22
Figure 5-4 Configuration of Linux Kernel	23
Figure 5-5 Hardware Setting of Demo	23
Figure 5-6 UART Execution Results	24
Figure 5-7 Enter the Linux Configs Menu	24
Figure 5-8 Linux Configs Menu	25
Figure 5-9 The Menu of Device Drivers	25
Figure 5-10 Settings of Device Drivers (1)	26
Figure 5-11 Settings of Device Drivers (2)	26
Figure 5-12 Settings of Device Drivers (3 and 4)	26
Figure 5-13 Settings of Device Drivers (5)	27
Figure 5-14 Configuration Completed	27
Figure 5-15 Start Window of Menuconfig	27
Figure 5-16 Tslib Setting	28
Figure 5-17 Save and Update the Configs	28
Figure 5-18 Added Code of "nuc980.dtsi"	29
Figure 5-19 Location of "nuc980.dtsi"	30
Figure 5-20 Updated Part of "nuc980-iot-v1.0.dts"	31
Figure 5-21 Added Part of "nuc980-iot-v1.0.dts" (1)	32
Figure 5-22 Added Part of "nuc980-iot-v1.0.dts" (2)	33
Figure 5-23 Location of "nuc980.dtsi"	33
Figure 5-24 Rebuilding the Image File	34
Figure 5-25 Logs of Power-on	34
Figure 5-26 Location of Files	34
Figure 5-27 Location of emWin AppWizard Package	35
Figure 5-28 VCOM of Evaluation Board	36
Figure 5-29 Demo of the EmWin AppWizard	36

1 OVERVIEW

This document describes the Linux Environment Image which provides users with a packaged Linux platform to develop their applications on NUC980 series evaluation boards. It is integrated with relevant kits of NUC980 development environment, so users can skip the installation process, and only need to focus on programming applications. A Linux platform is needed to build Linux kernel, U-Boot, and applications using the Linux compiling toolchain. In this user manual, NuMaker-NUC980-IIoT is used as the target board for example.

The Linux environment image includes following contents:

- Linux platform: ubuntu-18.04.3-desktop-amd64.
- GCC 4.8.4 crLinuxs compiler with EABI support.
- uClibc-0.9.33
- Binutils-2.24
- Demo program for device drivers, busybox, mtd-utils, and other open source applications.
- Linux 4.4 kernel source code and NUC980 device drivers.
- Linux 5.10 kernel source code and NUC980 device drivers.

U-Boot 2016.11 source code includes NUC980 device drivers.

1.1 Development Environment

Users can run the Linux environment Image on the virtual machine on a PC, and communicate or debug with the NUC980 series evaluation board via UART(connected by Type-C).

In addition, users can also load the binary file to the NUC980 series evaluation board for execution or debugging by USB interface. Then, use the programming tool NuWriter to program NAND, SPI, and eMMC. The development environment is shown in Figure 1-1.

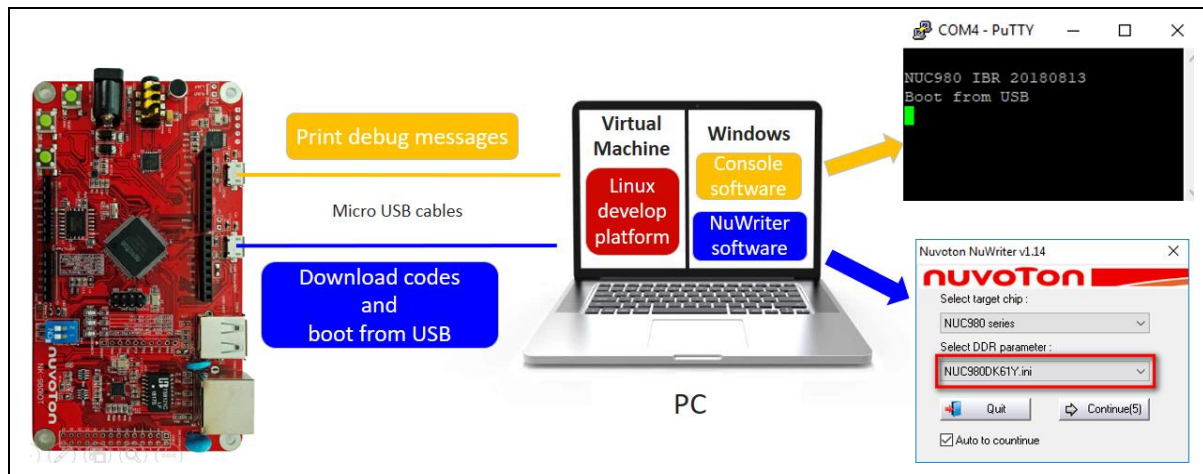


Figure 1-1 Development Environment Setup

2 INSTALLATION OF LINUX ENVIROMENT

The Linux Environment Image provides a Linux operating system running on VMware.

This chapter introduces how to install VMware virtual machine, and the steps to install the Linux Environment Image.

2.1 Installing VMware Virtual Machine

Users can download free VMware Workstation Player 15.5 from VMware official website: <http://www.VMware.com/>. Select **Downloads** → “**Workstation Player**” and then click “**VMware Workstation 15.5 Player for Windows**”. Please refer to Figure 2-1 and Figure 2-2.

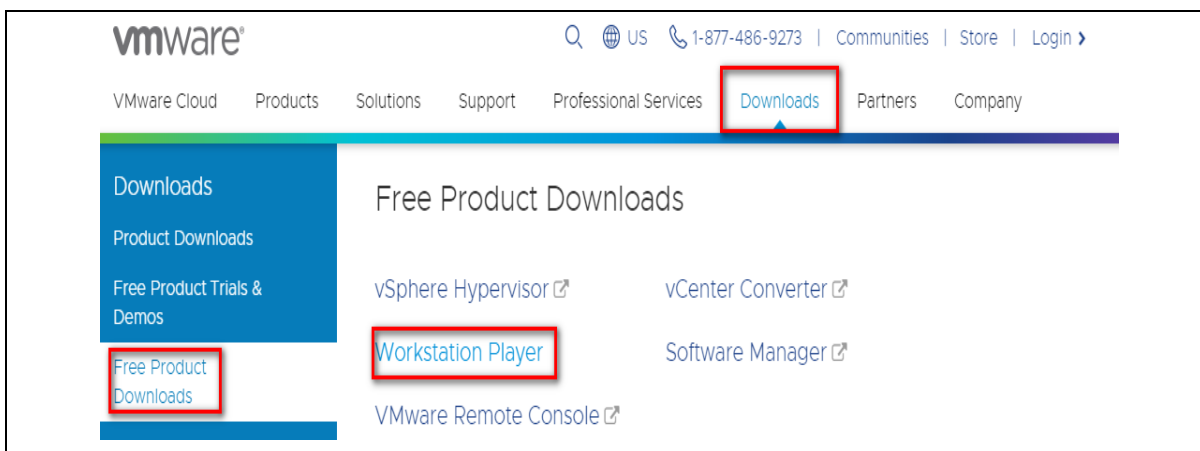


Figure 2-1 Download VMware Workstation Player

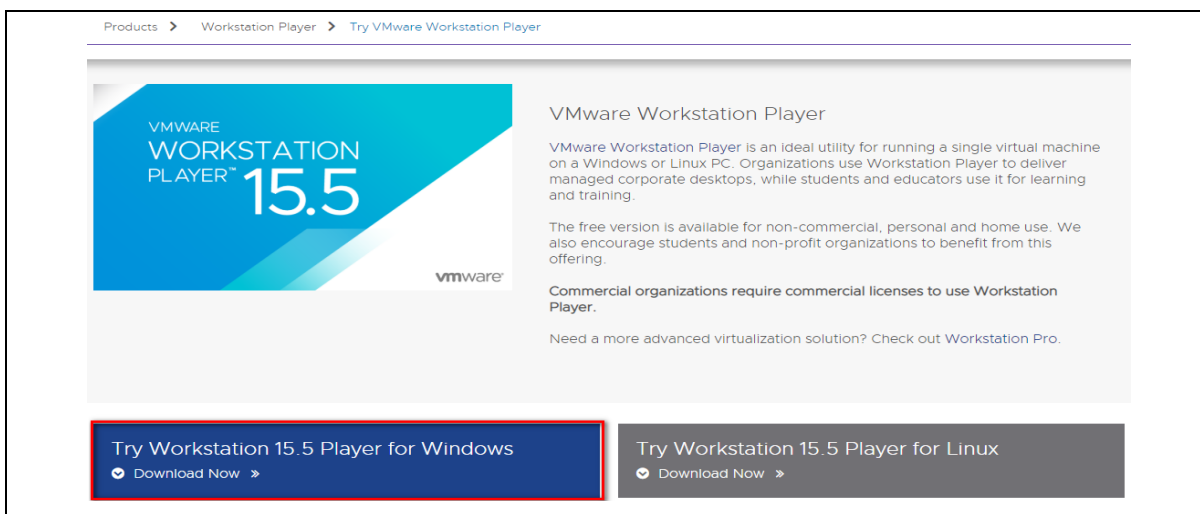


Figure 2-2 Download VMware Workstation Player for Windows

After the download is complete, follow the steps of Figure 2-3 ~ Figure 2-8 to install VMware Workstation Player.

Note: Make sure to enable Virtualization of BIOS before installing VMware.

1. Execute the installation file and accept the license agreement.

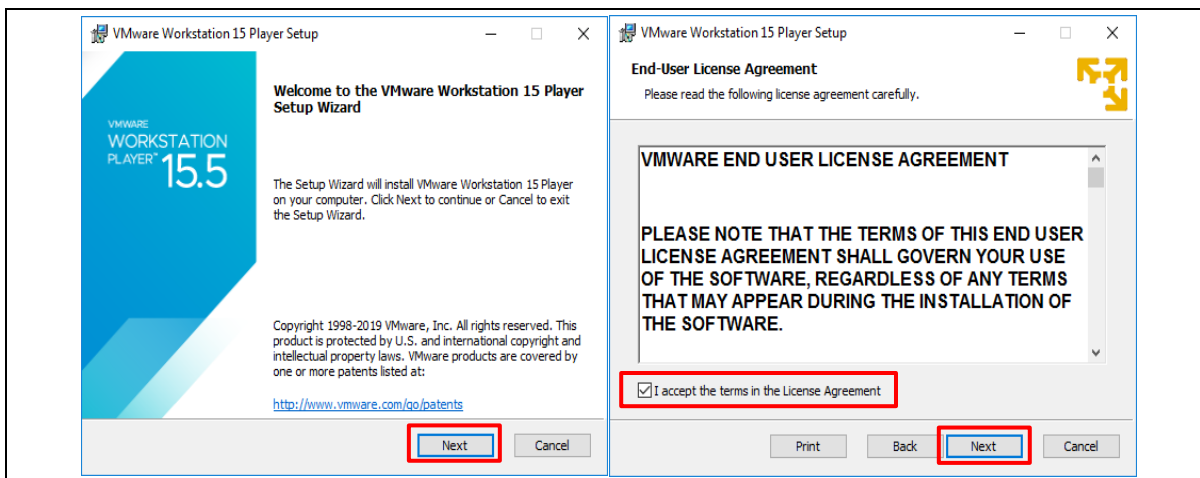


Figure 2-3 Install VMware Workstation Player (1)

2. Select the installation location.

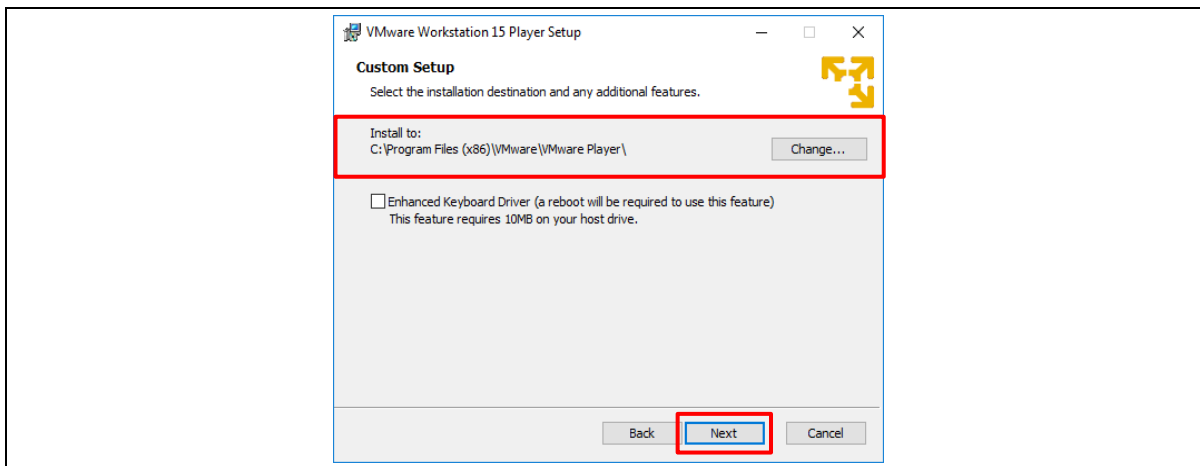


Figure 2-4 Install VMware Workstation Player (2)

3. Settings of user experience (non-essential).

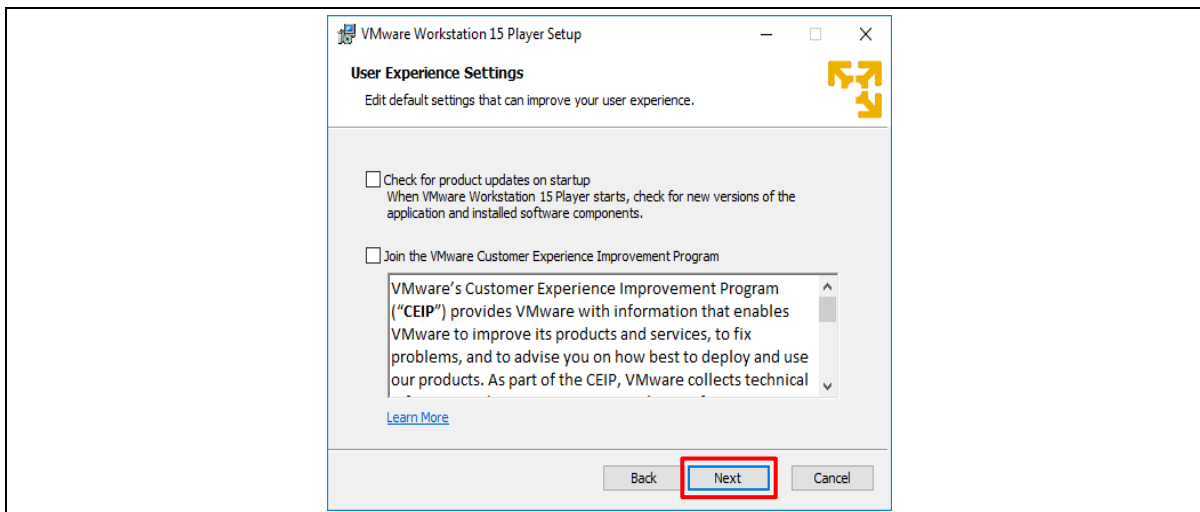


Figure 2-5 Install VMware Workstation Player (3)

4. Create the quick start icon on the desktop.

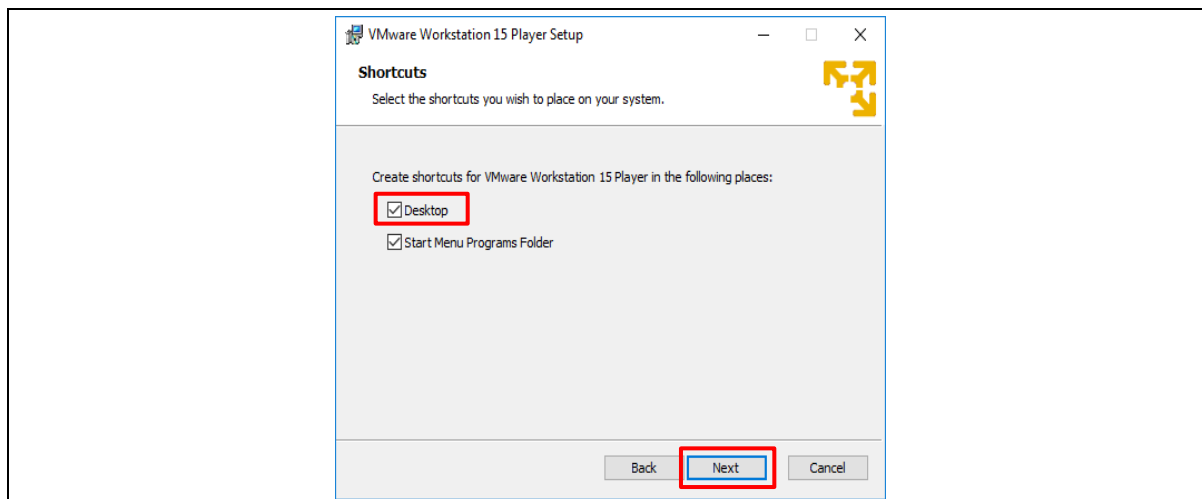


Figure 2-6 Install VMware Workstation Player (4)

5. Start to install.

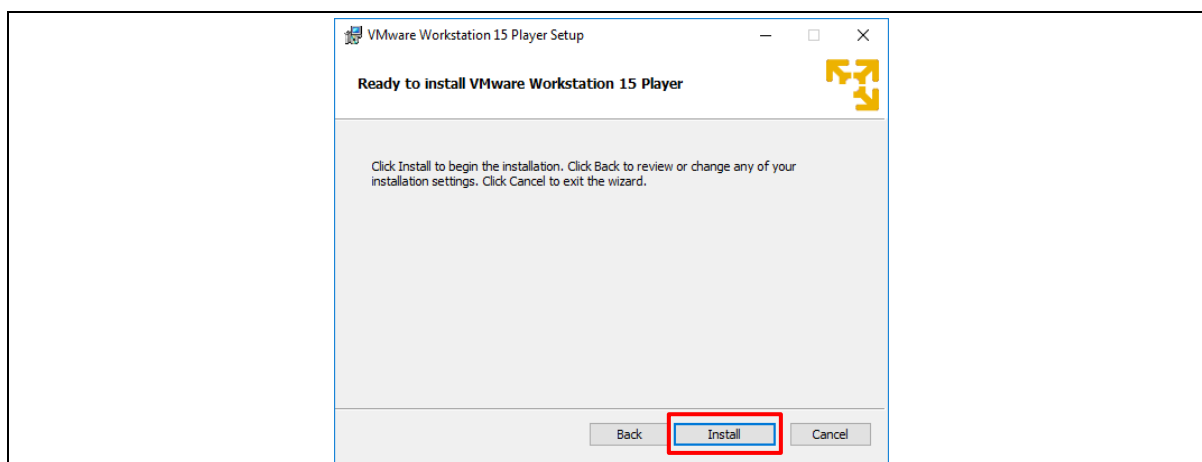


Figure 2-7 Install VMware Workstation Player (5)

6. After installation, open VMware Workstation 15 Player.

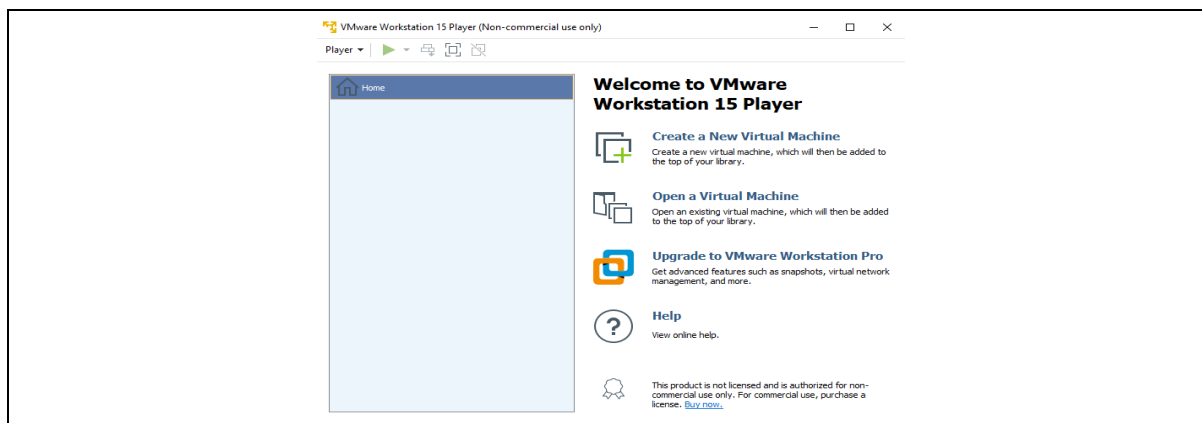


Figure 2-8 Menu of Virtual Machine

If there are installation errors, follow the procedure below to solve the problem:

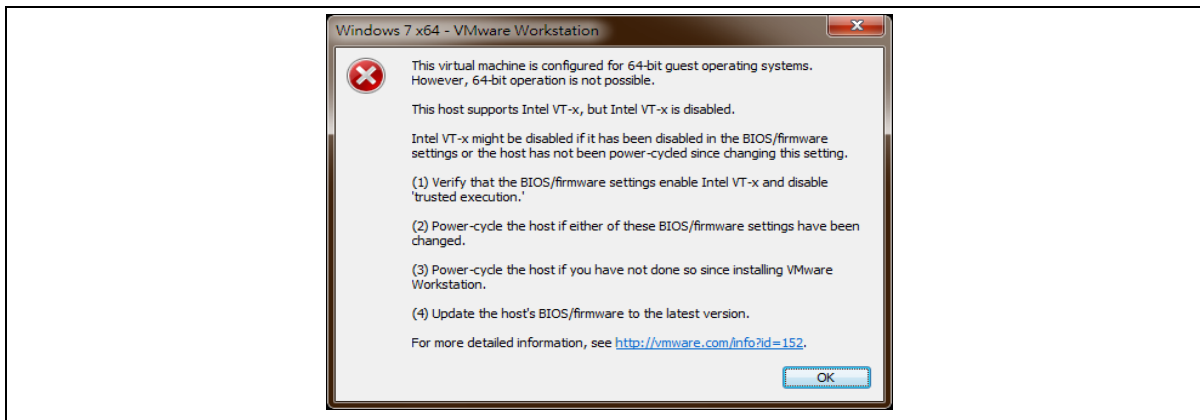


Figure 2-9 Installation Errors of VMware

- Enable Virtualization from BIOS before installing VMware, as shown in Figure 2-10.
 - Restart computer.
 - Log in to the BIOS screen when booting.
 - Select configuration, then select Intel virtual technology. The system is disabled by default.
 - Change disabled to enabled.
 - Save the settings and restart.
- There will be some differences depending on each BIOS.

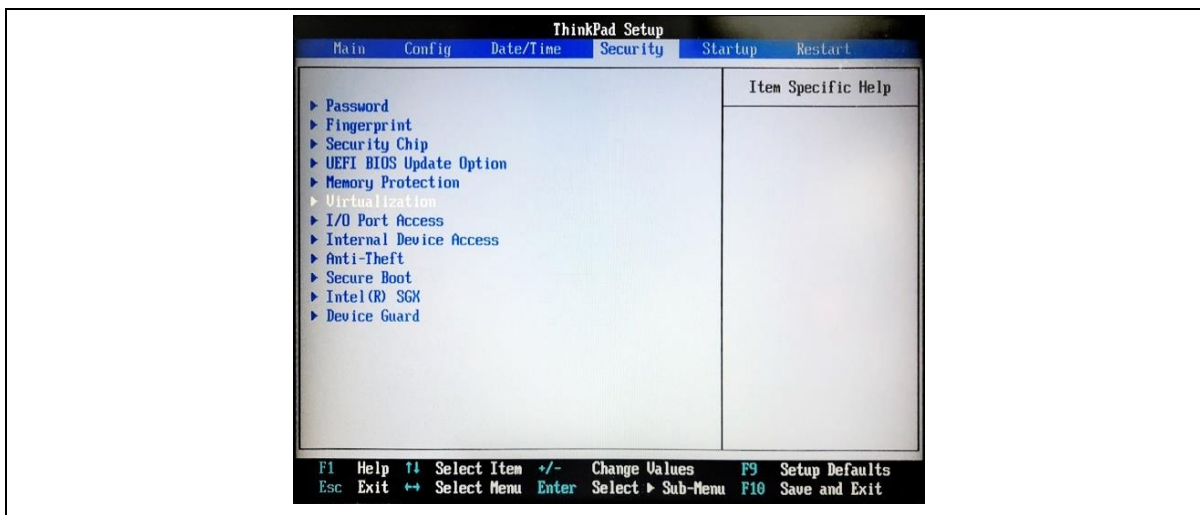


Figure 2-10 BIOS Settings

2.2 Downloading and Opening Linux Environment Image

Download the image resource " **VMware Linux Virtual machine image**" at the following URL:

<https://www.nuvoton.com/products/iot-solution/iot-platform/numaker-iiot-nuc980/>

Decompress **Nuvoton - Ubuntu 18.04.zip** to the target folder, then refer to Figure 2-11 to Figure 2-14 and the procedure below to open Linux environment image.

1. Open the Virtual Machine.

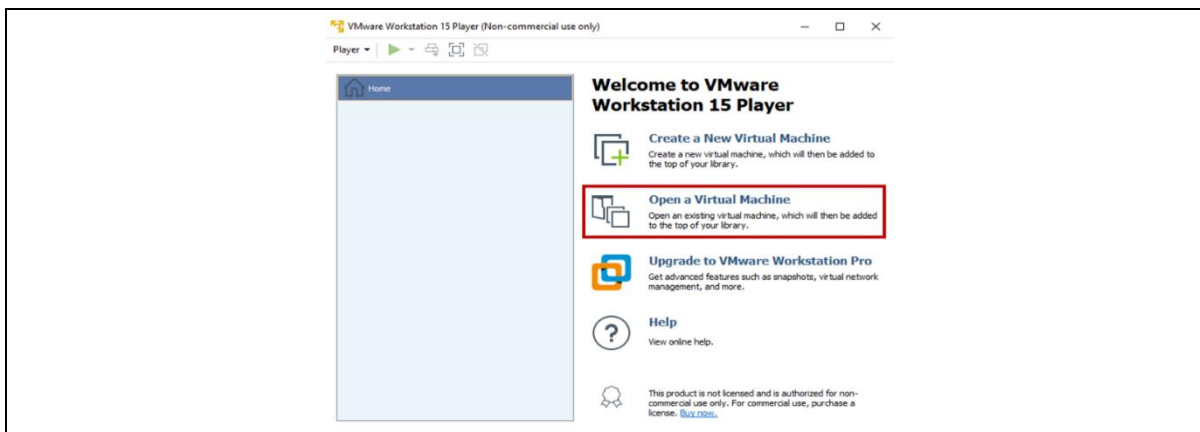


Figure 2-11 Open the Linux Environment (1)

2. Select "Ubuntu 64-bit_nuvoton.vmx" under **Nuvoton - Ubuntu 18.04** folder.

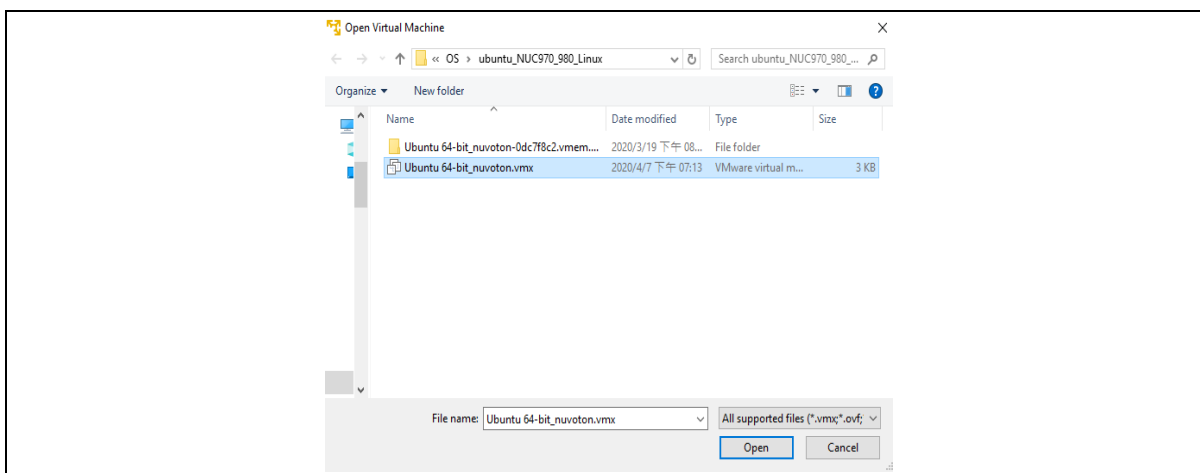


Figure 2-12 Open the Linux Environment (2)

3. The Linux Environment Image is loaded now. Click "Play virtual machine".

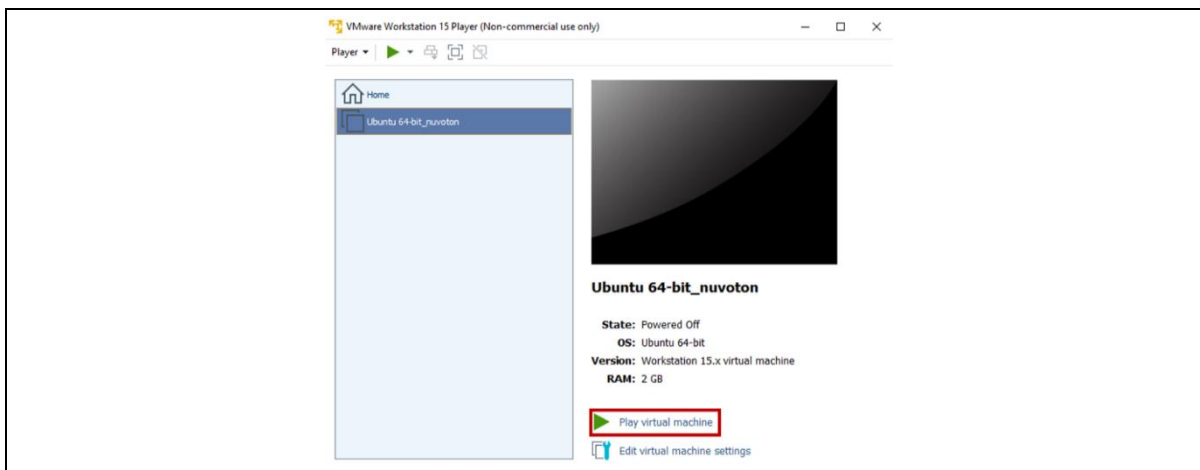


Figure 2-13 Open the Linux Environment (3)

4. You will see messages when opening a new image. Click the options such as Figure 2-14.

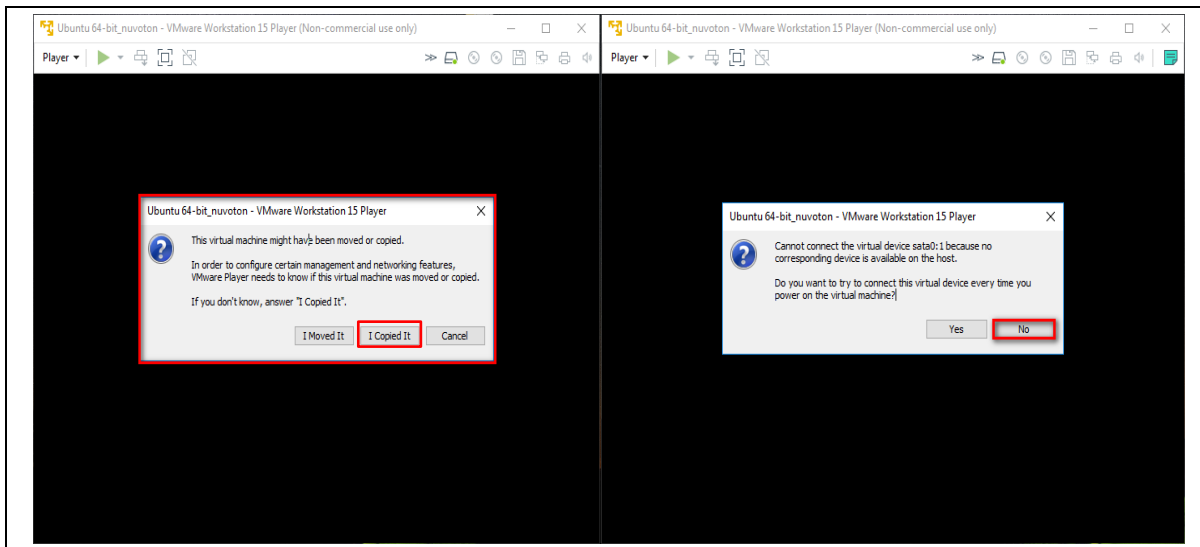


Figure 2-14 Open the Linux Environment (4)

An Ubuntu login window will show up after installation is complete. Log in with the username "nuvoton" and the password is "user".



Figure 2-15 Login Linux Virtual Machine

Check whether the time zone is matched with your area.

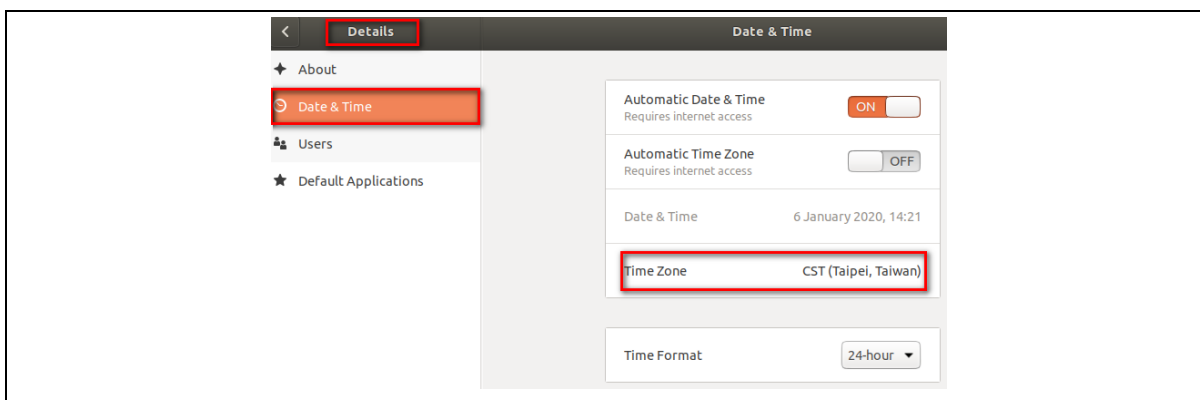


Figure 2-16 Setting of Time Zone

3 BUILDING IMAGES OF LINUX KERNEL

Before using the NUC980 evaluation board, users should make the function configurations. Nuvoton has prepared several basic config packages for users to choose the config file which matches the requirements.

With different versions of the Linux kernel, its requirement images are other and will be illustrated in the following section. For more details on programming, please refer to section 4.5 of the *NuMaker-IIoT-NUC980G2 User Manual*.

3.1 For Version 4.4 of Linux Kernel

3.1.1 Getting the Buildroot

Find **NUC970_Buildroot-master** folder under personal folder "Home".

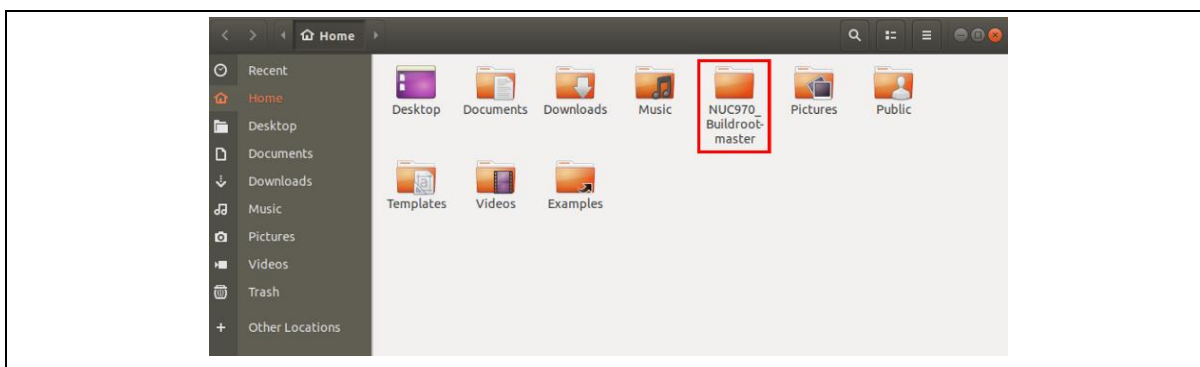


Figure 3-1 NUC970_Buildroot-master Folder

The **NUC970_Buildroot-master** folder contains everything needed to develop Linux kernel, uboot and applications.

Buildroot is a simple, efficient and easy-to-use tool, which is a set of Makefiles and patches that simplifies and automates the process of building a complete and bootable Linux environment for an embedded system through cross-compilation. Users can set the corresponding configuration file according to the evaluation board and development requirements, through the command "make XXX_defconfig" to set the buildroot configuration.

If users need to modify buildroot parameters, use command "**make menuconfig**" to set the configuration of buildroot. So far, the Linux platform of version 4.4 had built successfully. The following section will introduce how to compile and generate executable files in the Linux platform.

3.1.2 Compiling Linux Kernel and U-boot

To compile Linux Kernel and U-boot, first, set the configuration of buildroot. Take the NuMaker-NUC980-IIoT evaluation board as an example and open a terminal and find the board configuration file at the following path.

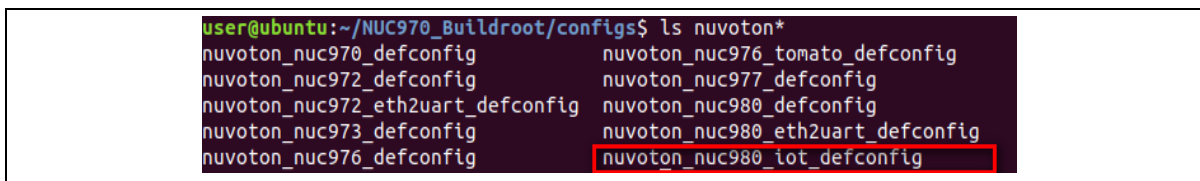


Figure 3-2 Configuration File

Refer to Figure 3-3, run command "**make nuvoton_nuc980_iot_defconfig**" to import configuration file into .config, then run command "**make**" to compile Linux kernel and U-boot.

```
$ make nuvoton_nuc980_iot_defconfig
$ make
```

```
user@ubuntu:~/NUC970_Buildroot-master$ make nuvoton_nuc980_iot_defconfig
#
# configuration written to /home/user/NUC970_Buildroot-master/.config
#
user@ubuntu:~/NUC970_Buildroot-master$ make
/usr/bin/make -j1 O=/home/user/NUC970_Buildroot-master/output HOSTCC="/usr/bin/gcc" HOSTCXX="/usr/bin/g++" silentoldconfig
make[1]: Entering directory '/home/user/NUC970_Buildroot-master'
```

Figure 3-3 Configuration Setting and Compiling

Note: Based on what target board you use, set the corresponding configuration file.

After compilation, there will be four images used for downloading.

- **Image:** Linux kernel image used for running on DDR.
- **ulimage:** Linux kernel image used for booting with uboot.
- **u-boot.bin:** Uboot image.
- **u-boot-spl.bin:** Uboot Secondary Program Loader image", used for booting from SPI NAND.

The images are generated at the following locations:

Image and **ulimage** are generated at the path "*NUC970_Buildroot-master/output/images*".

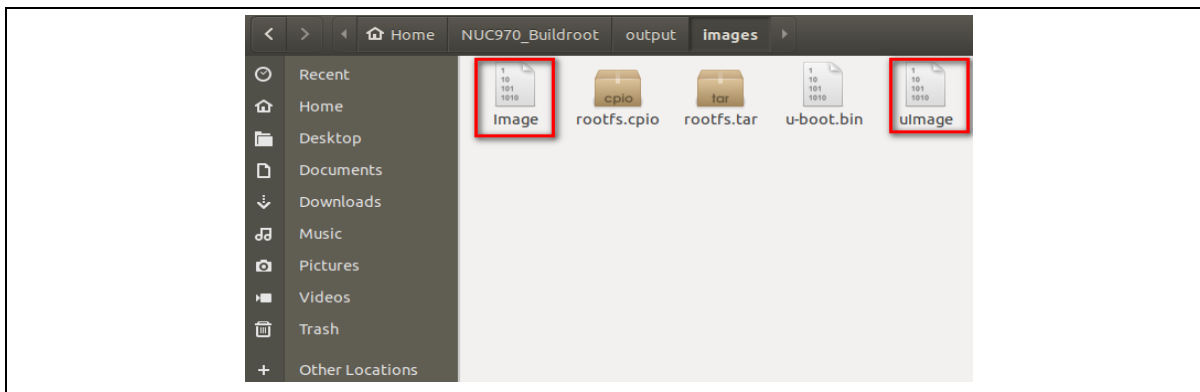


Figure 3-4 Location of Image and Uimage File

u-boot.bin is generated at the path "*NUC970_Buildroot-master/output/build/u-boot-master*".

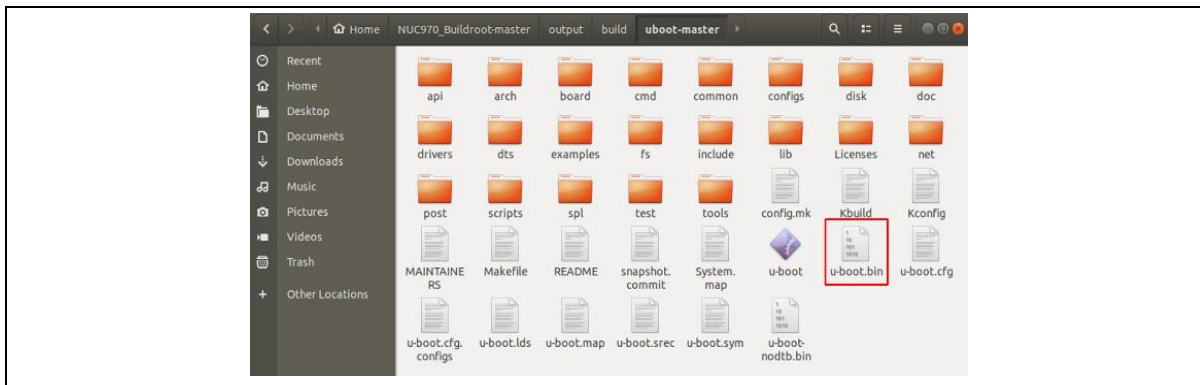


Figure 3-5 Location of U-boot.bin File

u-boot-spl.bin is generated at the path "NUC970_Buildroot-master/output/build/u-boot-master/spl".

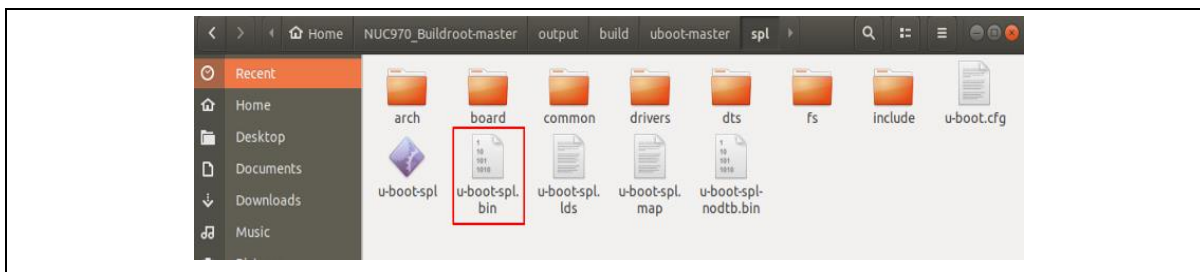


Figure 3-6 Location of U-Boot-spl.bin File

Download the above four images into your evaluation board and it will be ready to work.

You can find the download procedure by referring to the *NUWRITER TOOL* chapter of "[NuMaker-NUC980-IIoT User Manual](#)".

3.1.3 Updating Source Code

When Nuvoton releases new patches on the Internet, please refer to the following steps to update Linux, u-boot, and buildroot.

Note: If it's your first time to set up the Linux platform, please update Linux, u-boot, and buildroot first.

- Linux and uboot Update:

Delete original folders and .gz files of linux-master and uboot-master in the Linux platform at the location as shown below.

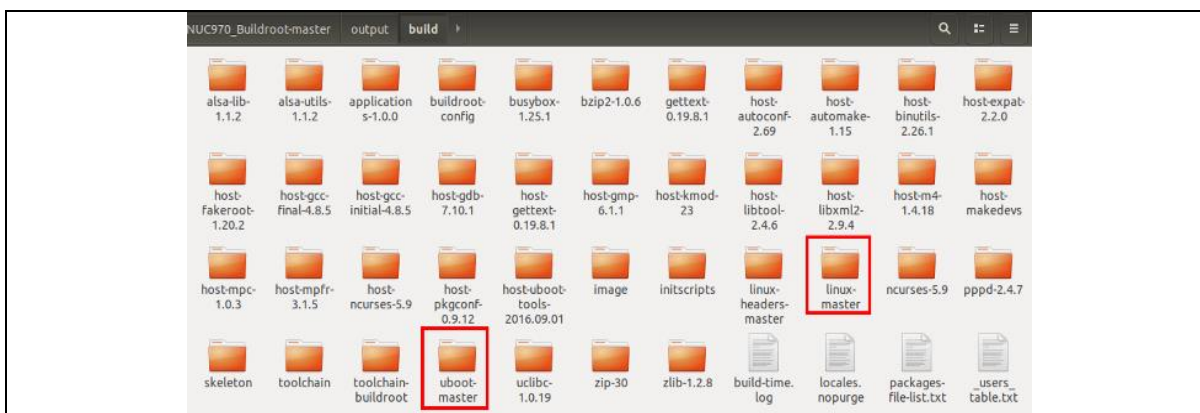


Figure 3-7 Location of Linux-master and Uboot-master File



Figure 3-8 Location of Linux-master.gz and Uboot-master.gz File

After deleting these files, at the path "/NUC970_Buildroot-master" run command "**make**"

again, and **NUC970_Buildroot-master** will automatically download the latest patch of Linux and U-boot on the Internet.

- Buildroot Update:

At the path `"/NUC970_Buildroot-master"`, run command **"git pull"**. Then the buildroot will be updated to the latest version, refer to Figure 3-9.

```
user@ubuntu:~/NUC970_Buildroot-master$ git pull
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 52 (delta 26), reused 45 (delta 19), pack-reused 0
Unpacking objects: 100% (52/52), done.
From https://github.com/OpenNuvoton/NUC970_Buildroot
cb79d5a..24a95c6 master -> origin/master
Updating cb79d5a..24a95c6
Fast-forward
 board/nuvoton/install_rootfs_emwin.sh | 5 +
 board/nuvoton/rootfs-emwin/etc/init.d/rcS | 9 +
 board/nuvoton/rootfs-emwin/etc/profile | 5 +
 board/nuvoton/rootfs-lorag/etc/init.d/rcS | 25 +-
 .../rootfs-lorag/usr/local/bin/global_conf.json | 147 ++
 .../rootfs-lorag/usr/local/bin/local_conf.json | 7 +
 board/nuvoton/rootfs-lorag/usr/local/bin/lora.sh | 4 +
 .../rootfs-lorag/usr/local/bin/lora_pkt_fwd | Bin 0 -> 124852 bytes
 .../rootfs-lorag/usr/local/bin/reset_lgw.sh | 62 +
 .../rootfs-lorag/usr/local/bin/update_gwid.sh | 31 +
 board/nuvoton/rootfs-lorag/usr/local/bin/wifi.sh | 5 +
```

Figure 3-9 Buildroot Update

3.2 For Version 5.10 of Linux Kernel

3.2.1 Getting the Buildroot

For Linux kernel 5.10, NUC980 and MA35D1 are used with the same buildroot, users can find the **MA35D1_Buildroot** folder under the personal folder **"Home"**.

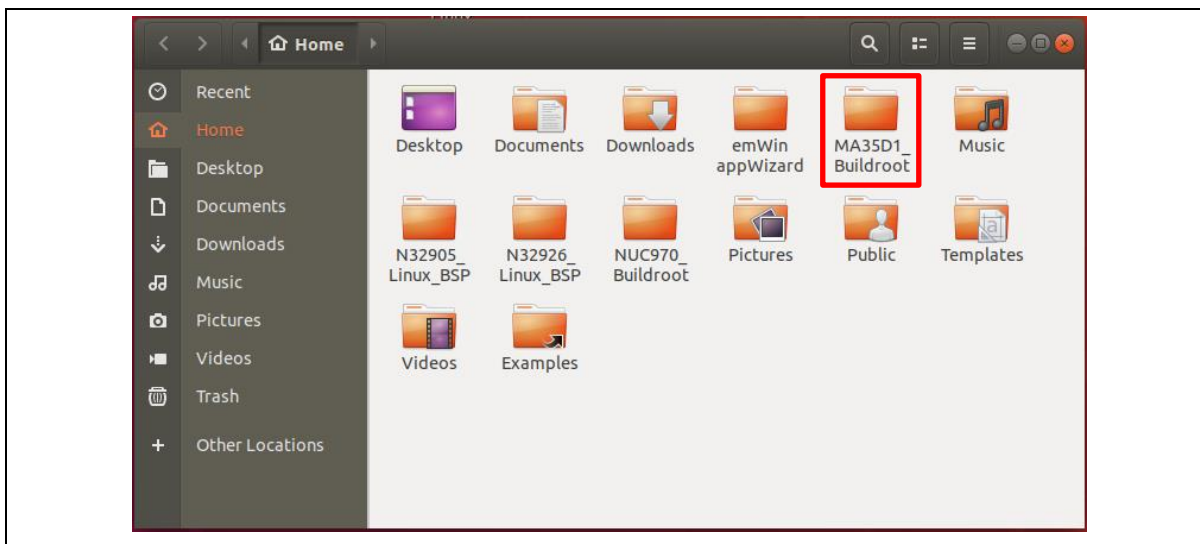


Figure 3-10 MA35D1_Buildroot Folder

The **MA35D1_Buildroot** folder contains everything needed to develop Linux kernel, uboot, and applications.

Buildroot is a simple, efficient, and easy-to-use tool, which is a set of Makefiles and patches that simplifies and automates the process of building a complete and bootable Linux environment for an embedded system through cross-compilation.

Users can set the corresponding configuration file according to the evaluation board and development

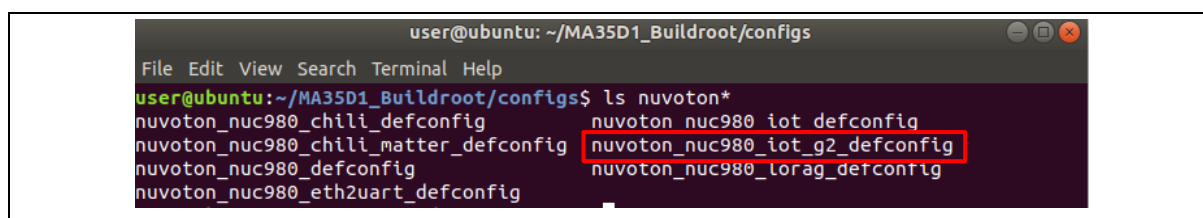
requirements, through the command "make XXX_defconfig" to set the buildroot configuration. (For example, set buildroot for the Numaker-IIoT-NUC980-G2 evaluation board by the command "make nuvoton_nuc980_iot_g2_defconfig")

If users need to modify buildroot parameters, use the command "make menuconfig" to set the configuration of buildroot.

Now, the Linux platform version 5.10 had been built successfully. The following section will introduce how to compile and generate executable files in the Linux platform.

3.2.2 Compiling Linux Kernel and U-boot

After download, it's required to set the configuration of buildroot. Taking the NuMaker-IIoT- NUC980G2 evaluation board as an example, open a terminal and find the board configuration file at the following path.



```

user@ubuntu: ~/MA35D1_Buildroot/configs
File Edit View Search Terminal Help
user@ubuntu:~/MA35D1_Buildroot/configs$ ls nuvoton*
nuvoton_nuc980_chili_defconfig      nuvoton_nuc980_iot_defconfig
nuvoton_nuc980_chili_matter_defconfig  nuvoton_nuc980_iot_g2_defconfig
nuvoton_nuc980_defconfig           nuvoton_nuc980_lorag_defconfig
nuvoton_nuc980_eth2uart_defconfig

```

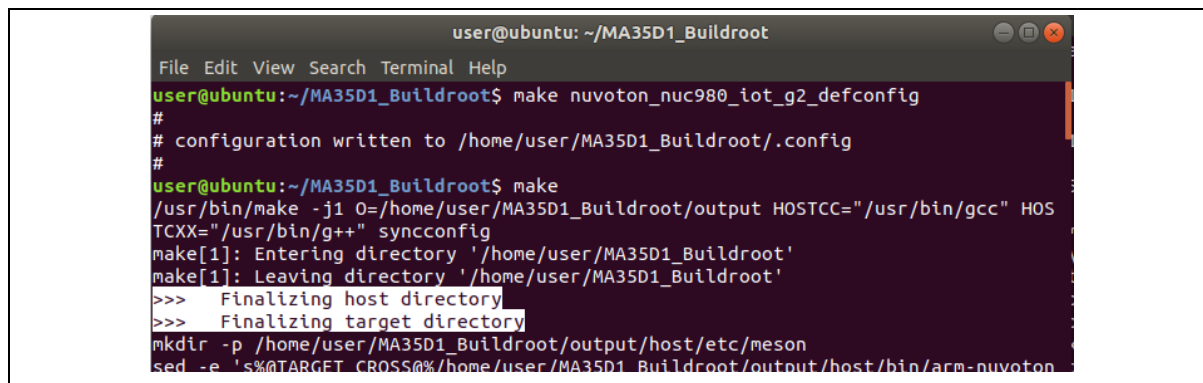
Figure 3-11 Configuration File

Input the command "**make nuvoton_nuc980_iot_g2_defconfig**" to import the configuration file into .config, then input the command "make" to compile the Linux kernel and U-boot.

```

$ make nuvoton_nuc980_iot_g2_defconfig
$ make

```



```

user@ubuntu: ~/MA35D1_Buildroot
File Edit View Search Terminal Help
user@ubuntu:~/MA35D1_Buildroot$ make nuvoton_nuc980_iot_g2_defconfig
#
# configuration written to /home/user/MA35D1_Buildroot/.config
#
user@ubuntu:~/MA35D1_Buildroot$ make
/usr/bin/make -j1 O=/home/user/MA35D1_Buildroot/output HOSTCC="/usr/bin/gcc" HOS
TCXX="/usr/bin/g++" synconfig
make[1]: Entering directory '/home/user/MA35D1_Buildroot'
make[1]: Leaving directory '/home/user/MA35D1_Buildroot'
>>> Finalizing host directory
>>> Finalizing target directory
mkdir -p /home/user/MA35D1_Buildroot/output/host/etc/meson
sed -e 's%@TARGET_CROSS@%/home/user/MA35D1_Buildroot/output/host/bin/arm-nuvoton

```

Figure 3-12 Configuration Setting and Compiling

Note: Based on what target board you use, set the corresponding configuration file.

After compilation, there will be five images used for downloading.

- **Image:** Linux kernel image used for running on DDR.
- **ulimage:** Linux kernel image used for booting with uboot.
- **dtb:** Device tree binary, describe the setting of peripherals in detail.
- **u-boot.bin:** Uboot image.
- **u-boot-spl.bin:** Uboot Secondary Program Loader image, used for booting from SPI NAND.

The images are generated at the following locations:

Image, ulmage, and dtb are generated at the path "MA35D1_Buildroot/output/images".

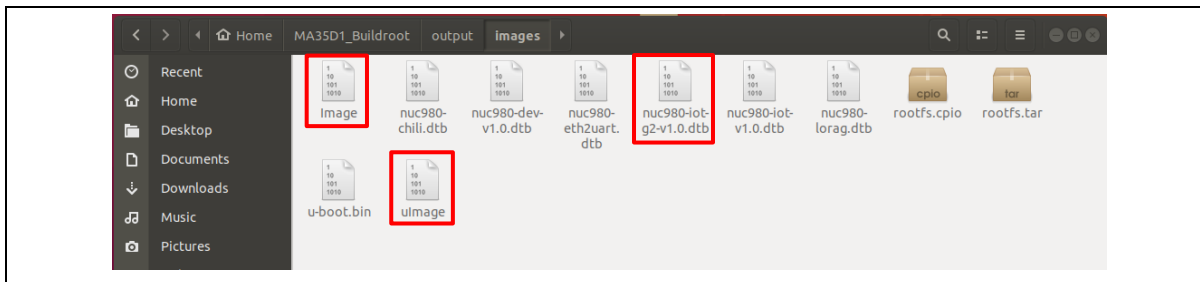


Figure 3-13 Location of Image, Uimage, and Dtb File

u-boot.bin is generated at the path "MA35D1_Buildroot/output/build/u-boot-customer".

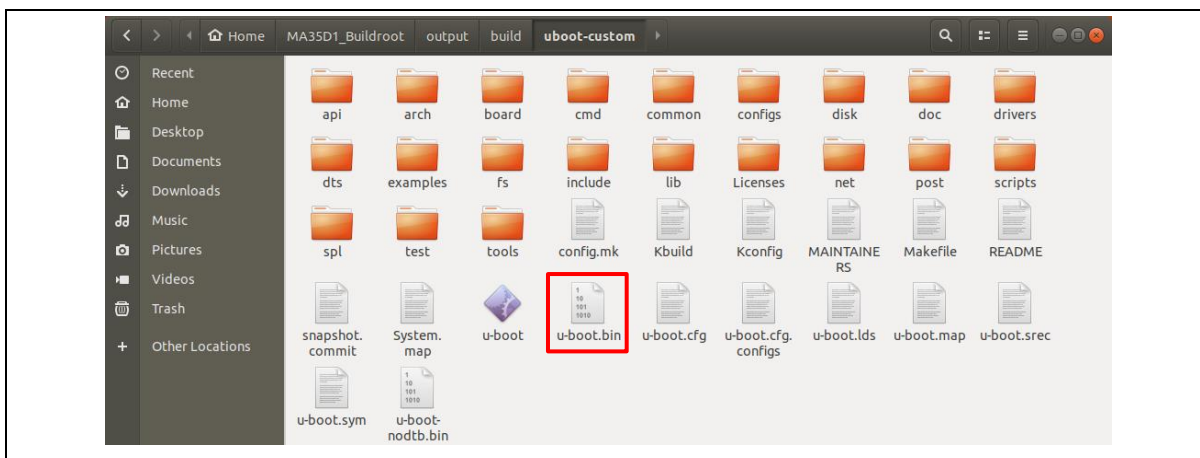


Figure 3-14 Location of U-boot.bin File

u-boot-spl.bin is generated at the path "MA35D1_Buildroot/output/build/u-boot-master/spl".

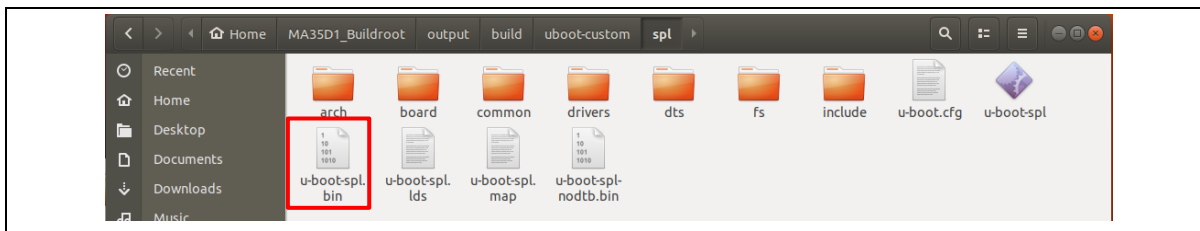


Figure 3-15 Location of U-boot-spl.bin File

3.2.3 Updating Source Code

When Nuvoton releases new patches on the Internet, please refer to the following steps to update Linux, u-boot, and buildroot.

Note: If it's your first time to set up the Linux platform, please update Linux, u-boot, and buildroot first.

- Linux and uboot Update:

Delete original folders and .gz files of linux-master and uboot-mater in the Linux platform at the location as shown below.

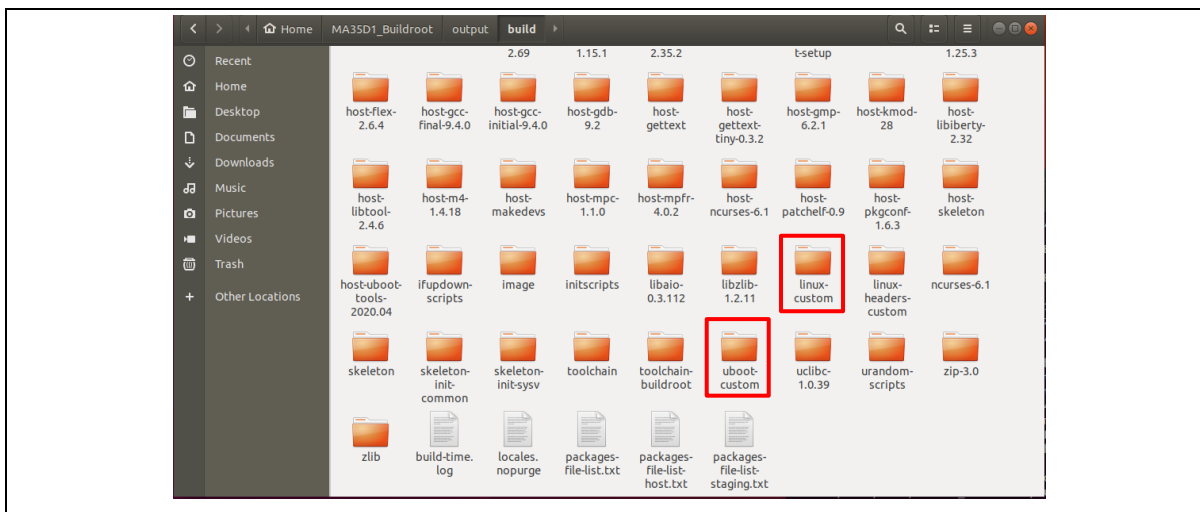


Figure 3-16 Location of Linux-custom and Uboot-custom File

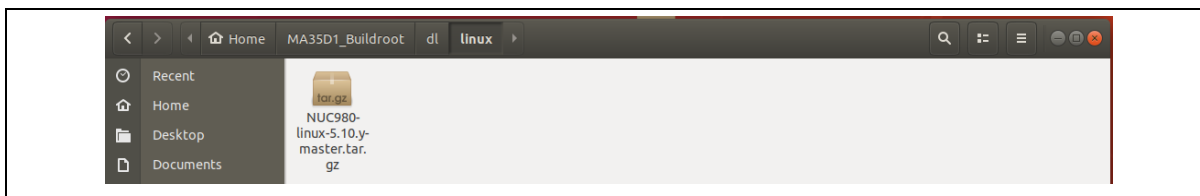


Figure 3-17 Location of Linux-master.tar.gz File

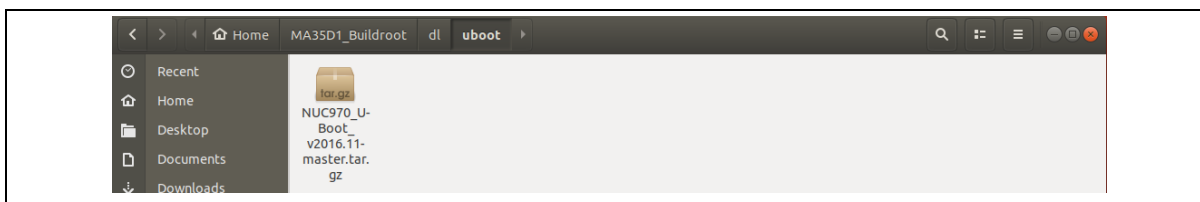


Figure 3-18 Location of Uboot-master.tar.gz File

After deleting the above file, open the terminal at the path `"/MA35D1_Buildroot"` and input the command **"make"** again, **MA35D1_Buildroot** will download the latest patch of Linux and U-boot on the Internet automatically.

- Buildroot Update:

Open the terminal at the path `"/MA35D1_Buildroot"` and input the command **"git pull"**, then the buildroot will be updated to the latest version; refer to Figure 3-19.



Figure 3-19 Updating the Buildroot

4 UPDATING TOOLCHAIN OF THE ENVIRONMENT VARIABLES

Before cross-compiling user applications, users need to set the Buildroot toolchain path as a local environment variable to avoid encountering errors such as Figure 4-1 when making files.

When entering the **NUC970_Buildroot** or **MA35D1_Buildroot** folder, the user can view the current environment variables through the command "env".

```
$ env
```

```
user@ubuntu:~/NUC970_Buildroot/output/build/applications-1.0.0/demos/uart$ make
arm-linux-gcc -static uart.c -o uart_demo -lpthread -lc -lgcc
make: arm-linux-gcc: Command not found
Makefile:11: recipe for target 'all' failed
make: *** [all] Error 127
```

Figure 4-1 Errors of Making File (Linux Kernel 4.4)

For Linux kernel 4.4, users can modify the environment variables by the command as follows, and then they can make files successfully.

```
$ export PATH=/home/user/NUC970_Buildroot/output/host/usr/bin:$PATH
```

```
IM_CONFIG_PHASE=2
XDG_CURRENT_DESKTOP=ubuntu:GNOME
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GNOME_TERMINAL_SERVICE=:1.92
XDG_SEAT=seat0
SHLVL=1
GDMSESSION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=user
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/run/user/1000/gdm/Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
PATH=/home/user/NUC970_Buildroot/output/host/usr/bin:/home/user/NUC970_Buildroot/output/host
/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/
games:/snap/bin
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1547,unix/ubuntu:/tmp/.ICE-unix/1547
LESSOPEN=| /usr/bin/lesspipe %s
GTK_IM_MODULE=ibus
_=usr/bin/env
user@ubuntu:~/NUC970_Buildroot$
```

Figure 4-2 After the Environment Variables of Linux Kernel 4.4 Modified

```
user@ubuntu:~/NUC970_Buildroot/output/build/applications-1.0.0/demos/uart$ export PATH=/home
/user/NUC970_Buildroot/output/host/usr/bin:$PATH
user@ubuntu:~/NUC970_Buildroot/output/build/applications-1.0.0/demos/uart$ make
arm-linux-gcc -static uart.c -o uart_demo -lpthread -lc -lgcc
arm-linux-strip uart_demo
```

Figure 4-3 Making File After Modifying the Environment Variables of Linux Kernel 4.4

For Linux kernel 5.10, users can modify the environment variables by the command:

```
$ source environment-setup
```

Figure 4-4 Modifying the Environment Variables of Linux Kernel 5.10

Figure 4-5 After the Environment Variables of Linux Kernel 5.10 Modified

Sept. 7, 2023

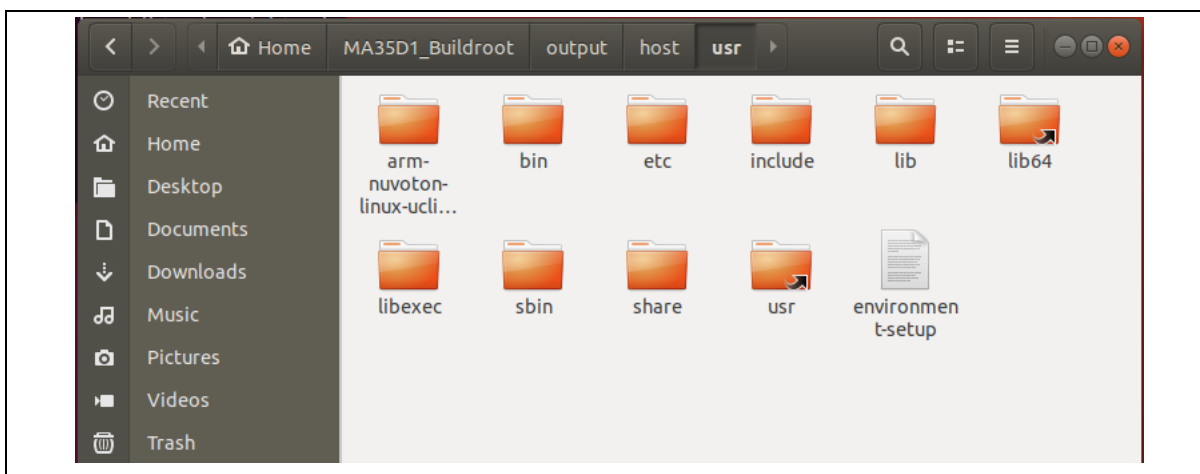


Figure 4-7 Location of Linux Kernel 5.10 Toolchains

5 LINUX USER APPLICATIONS

5.1 Example of UART

Take the UART demo as an example to help users understand how to put their applications into Linux kernel image, compile and download to the evaluation board. We take NuMaker-NUC980-IIoT for example.

Enter the **uart_demo** folder:

```
user@ubuntu:~$ cd NUC970_Buildroot-master/nuc980bsp/application/demos/uart/
user@ubuntu:~/NUC970_Buildroot-master/nuc980bsp/application/demos/uart$ ls
Makefile uart.c uart_demo
```

Figure 5-1 UART Application Folder

The folder contains three files:

- **Makefile:** Used for cross compiling.
- **uart.c:** The source code.
- **uart_demo:** Executable file.

Make the project and generate a new executable file.

```
user@ubuntu:~/NUC970_Buildroot-master/nuc980bsp/application/demos/uart$ make
arm-linux-gcc -static uart.c -o uart_demo -lpthread -lc -lgcc
arm-linux-strip uart_demo
user@ubuntu:~/NUC970_Buildroot-master/nuc980bsp/application/demos/uart$ ls -lrt
total 120
-rw-r--r-- 1 user user 4571 Mar 18 13:19 uart.c
-rw-r--r-- 1 user user 314 Mar 18 13:19 Makefile
-rwxr-xr-x 1 user user 108120 Mar 18 13:57 uart_demo
```

Figure 5-2 After Compilation

Copy the executable file to the following path:

```
$ cp uart_demo /home/user/NUC970_Buildroot-master/output/target/usr/bin
```

```
user@ubuntu:~/NUC970_Buildroot-master/nuc980bsp/application/demos/uart$ cp uart_demo /home/user/NUC970_Buildroot-master/output/target/usr/bin
```

Figure 5-3 Location of Executable File

The application is in the Linux kernel now. If you compile Linux kernel and U-boot, the new generated Image and ulmage will have the application inside.

Note: The UART demo needs two sets of UART (UART1 and UART2). Thus, before compilation, user needs to use Linux Kernel Configuration to check if they are enabled or not.

To enable UART1 and UART2, users can modify the corresponding device tree for Linux 5.10, and for users of Linux 4.4, please refer to the following steps:

1. Open the menu config by terminal.

```
$ make linux-menuconfig
```

Go to Device Drivers → Character devices → Serial drivers, and check NUC980 UART1, NUC980 UART2 are set to build-in.

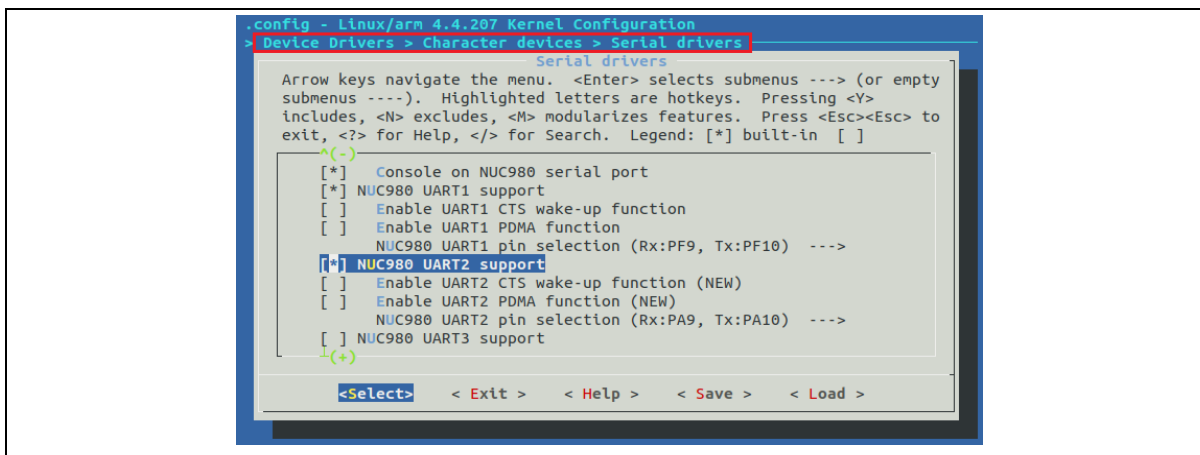


Figure 5-4 Configuration of Linux Kernel

2. Update image files.

Go to the path `"/NUC970_Buildroot-master"` and run command **"make"** to generate new images. Find new Images, ulmage, u-boot, u-boot-spl and download them into NuMaker-NUC980-IIoT.

For details on the download steps, please refer to the corresponding user manual of each evaluation board. After downloading images to NuMaker-NUC980-IIoT, turn SW2.1 and SW2.2 to Off, and connect UART1 with UART2 (referring to Figure 5-5).

- **Green line:** UART1 TX(NU4.2) connect to UART2 RX(CON11.19)
- **Yellow line:** UART1 RX(NU4.1) connect to UART2 TX(CON11.22)

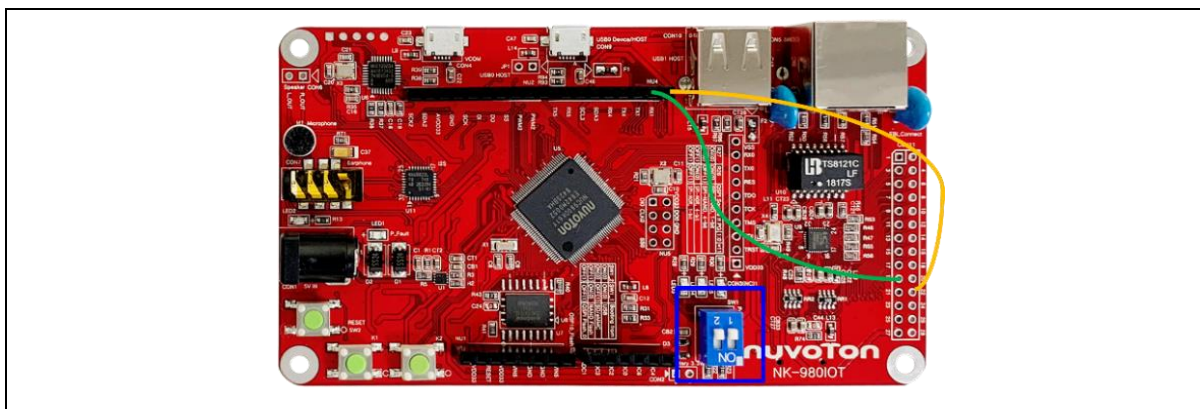


Figure 5-5 Hardware Setting of Demo

Open the terminal, set Baud rate to 115200 and then boot NuMaker NUC980 IIoT from SPI. After booting is completed, execute **uart.demo**. You should get the following messages if everything is fine.

```
$ ./usr/bin/uart.demo
```

```

done.
Starting network: OK
# ./usr/bin/uart_demo

demo uart1/uart2 external loop back function
opening serial port:/dev/ttyS1
opening serial port:/dev/ttyS2

uart2 send 100 bytes
uart1 send 100 bytes
uart1 receive 100 bytes
uart1 compare correct!!
uart1 test done!!

uart2 receive 100 bytes
uart2 compare correct!!
uart2 test done!!
  
```

Figure 5-6 UART Execution Results

5.2 Example of Using LCD Panel

5.2.1 Settings of LCD Panel

In Linux Version 5.10, add a control function for LCD panels. Users can design and display human-machine interfaces (HMI) to rich applications of products and make the operation more convenient.

Users can set the panel driver by the device tree in the following steps:

1. Go to the folder path of **MA35D1 Buildroots** and open the terminal, key-in command to set the configs in the Linux kernel.

```
$ make linux-menuconfig
```

```

user@ubuntu: ~/MA35D1_Builder
File Edit View Search Terminal Help
user@ubuntu:~/MA35D1_Builder$ make linux-menuconfig
BR_BINARIES_DIR=/home/user/MA35D1_Builder/output/images KFLAGS=-mno-attribute-alias PKG_CONFIG_PATH="" /usr/bin/make -j2 -
C /home/user/MA35D1_Builder/output/build/linux-custom HOSTCC=/usr/bin/gcc HOSTCXX=/usr/bin/g++ -O2 -I/home/user/MA35D1_Bu
ldroot/output/host/include -L/home/user/MA35D1_Builder/output/host/lib -Wl,-rpath,/home/user/MA35D1_Builder/output/host/
lib" ARCH=arm INSTALL_MOD_PATH=/home/user/MA35D1_Builder/output/target CROSS_COMPILE=/home/user/MA35D1_Builder/output/ho
st/bin/arm-nuovoton-linux-uclicgnueabi- DEPMOD=/home/user/MA35D1_Builder/output/host/sbin/depmod INSTALL_MOD_STRIP=1 HOSTC
C=/usr/bin/gcc menuconfig
make[1]: Entering directory '/home/user/MA35D1_Builder/output/build/linux-custom'
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/expr.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
UPD scripts/kconfig/mconf-cfg
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/Lxdialog/checklist.o
HOSTCC scripts/kconfig/Lxdialog/inputbox.o
HOSTCC scripts/kconfig/Lxdialog/menubox.o
HOSTCC scripts/kconfig/Lxdialog/textbox.o
HOSTCC scripts/kconfig/Lxdialog/util.o
HOSTCC scripts/kconfig/Lxdialog/yesno.o
HOSTLD scripts/kconfig/mconf
  
```

Figure 5-7 Enter the Linux Configs Menu

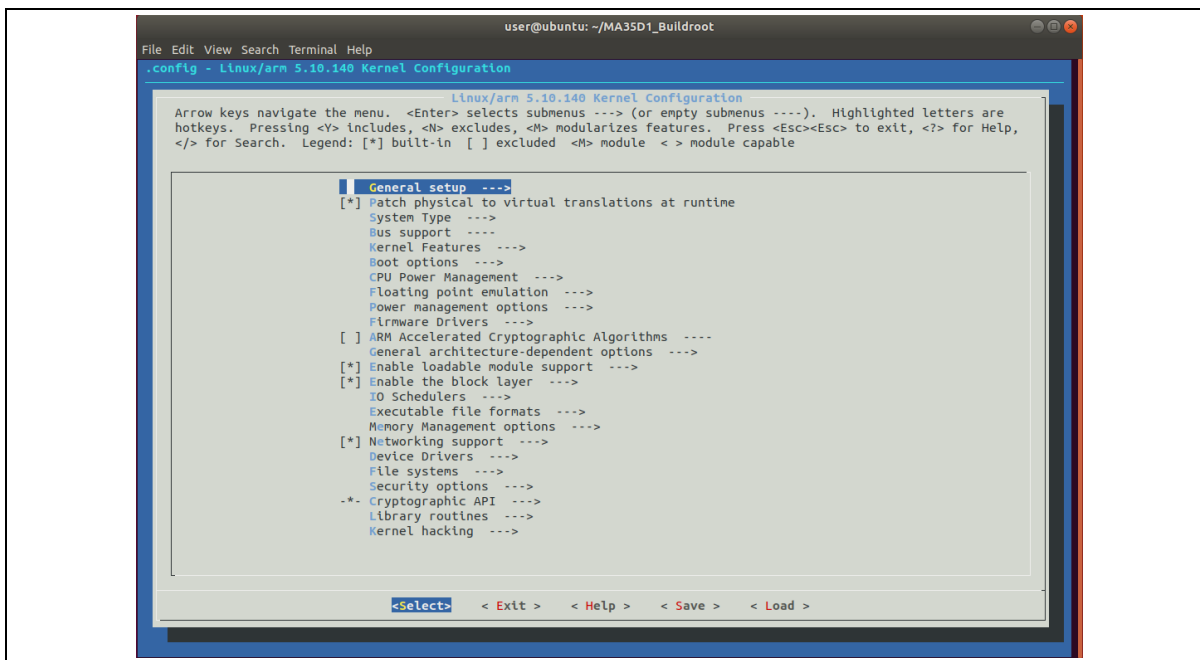


Figure 5-8 Linux Configs Menu

Select the option "Device Drivers" by blank key and configure the following settings:

- Graphics support → Frame buffer Devices → Support for frame buffer devices
- Staging drivers → Support for small TFT LCD display modules → FB driver for the ILI9341 LCD Controller
- Input device support → Event interface
- Input device support → Touchscreens → NUC980 touchscreen(ADC) support
- Industrial I/O support

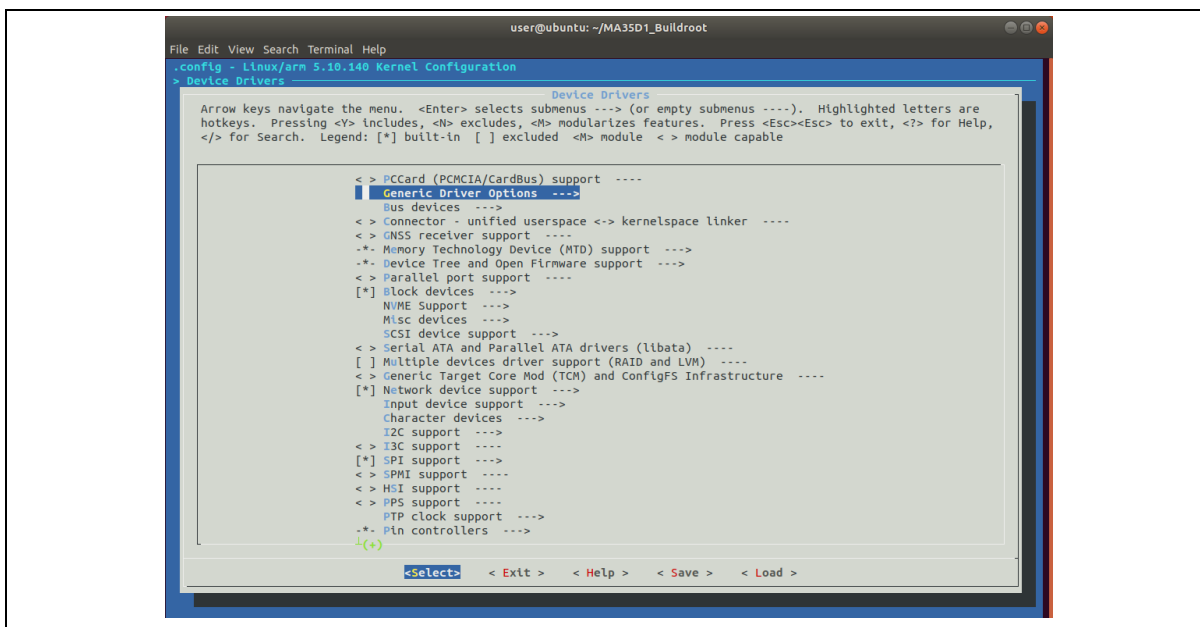


Figure 5-9 The Menu of Device Drivers

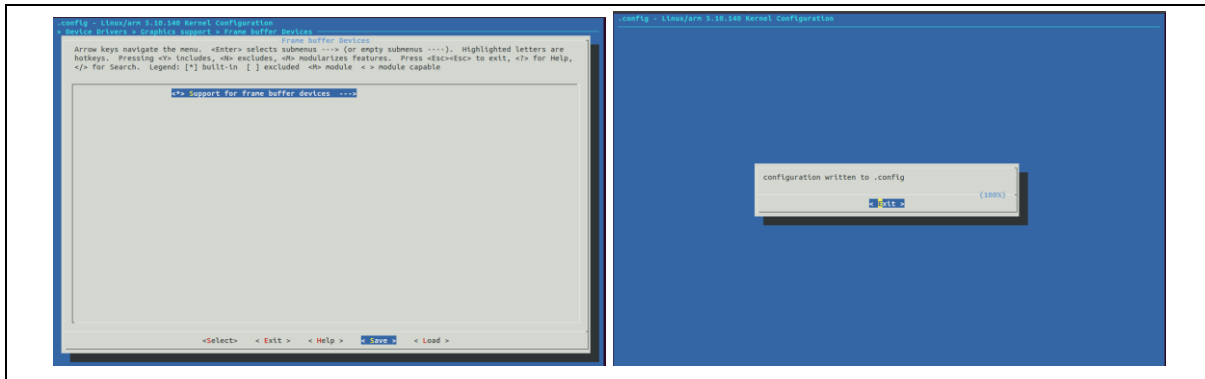


Figure 5-10 Settings of Device Drivers (1)

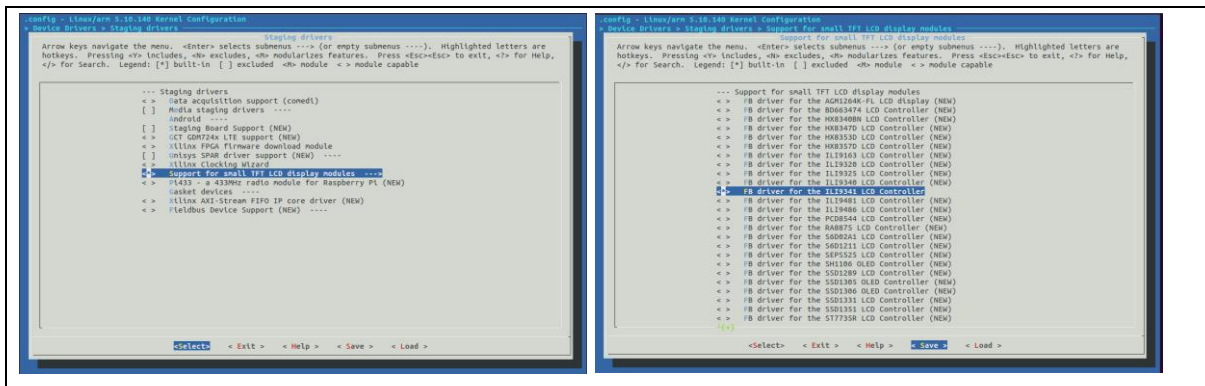


Figure 5-11 Settings of Device Drivers (2)

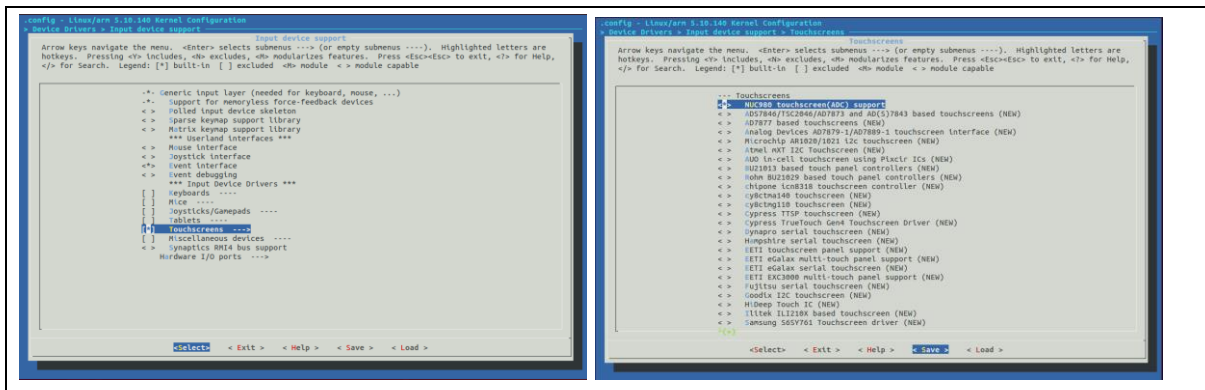


Figure 5-12 Settings of Device Drivers (3 and 4)

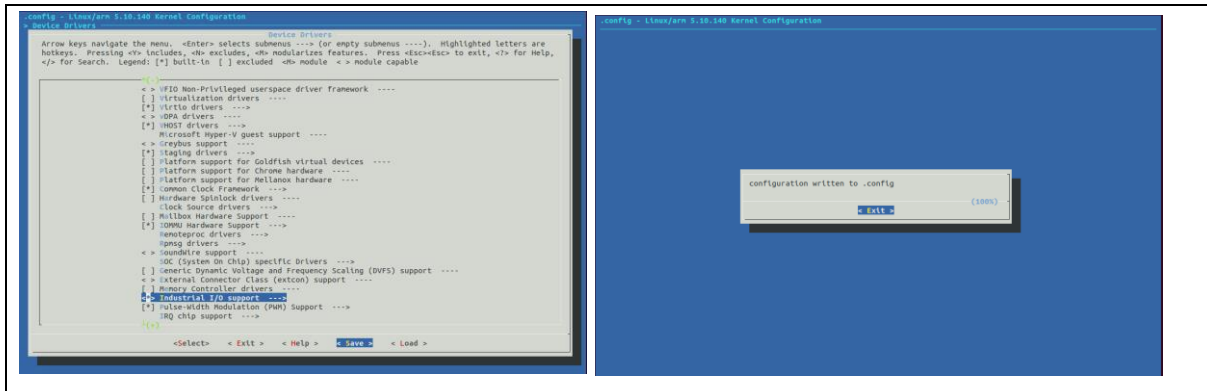


Figure 5-13 Settings of Device Drivers (5)

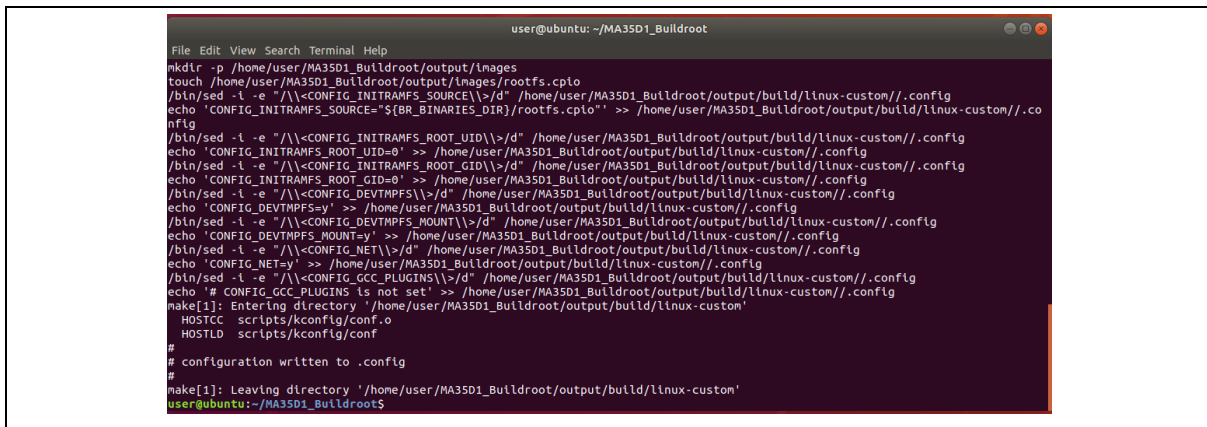


Figure 5-14 Configuration Completed

2. Go to the folder path of **MA35D1 Buildroots** and open the terminal, key-in command to set the configs of buildroot.

- **make menuconfig**

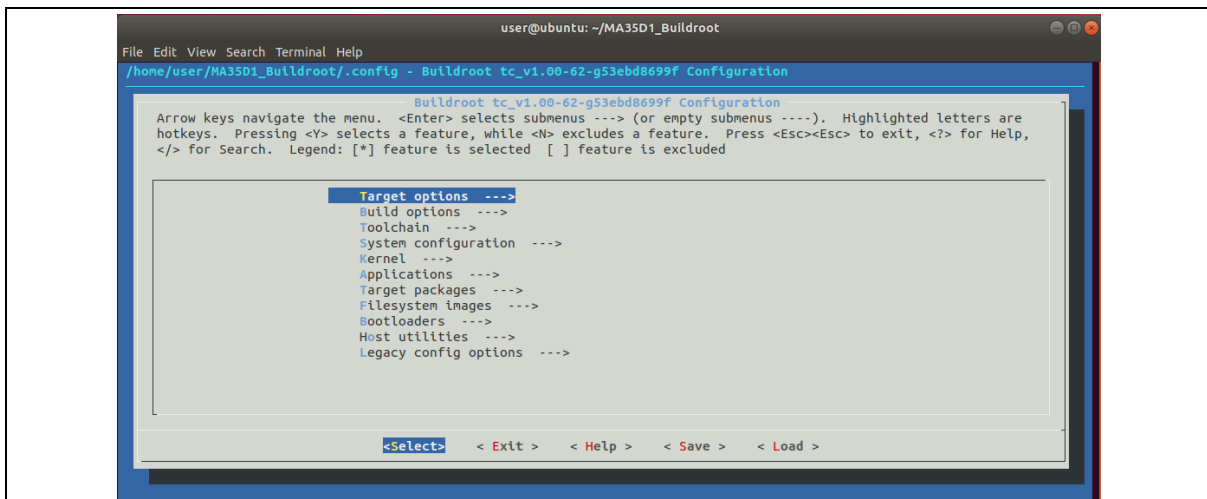


Figure 5-15 Start Window of Menuconfig

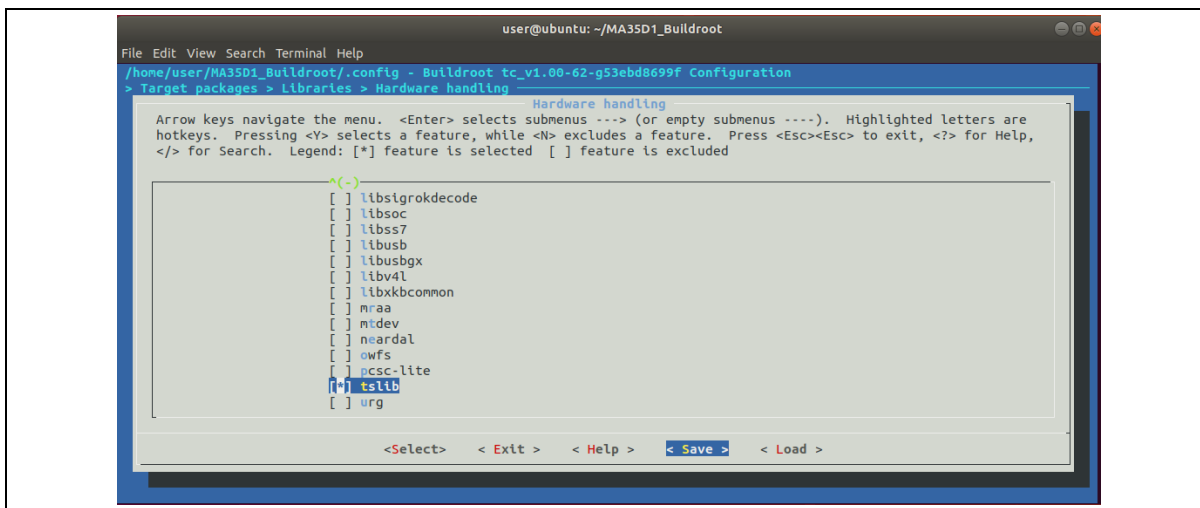


Figure 5-16 Tslib Setting

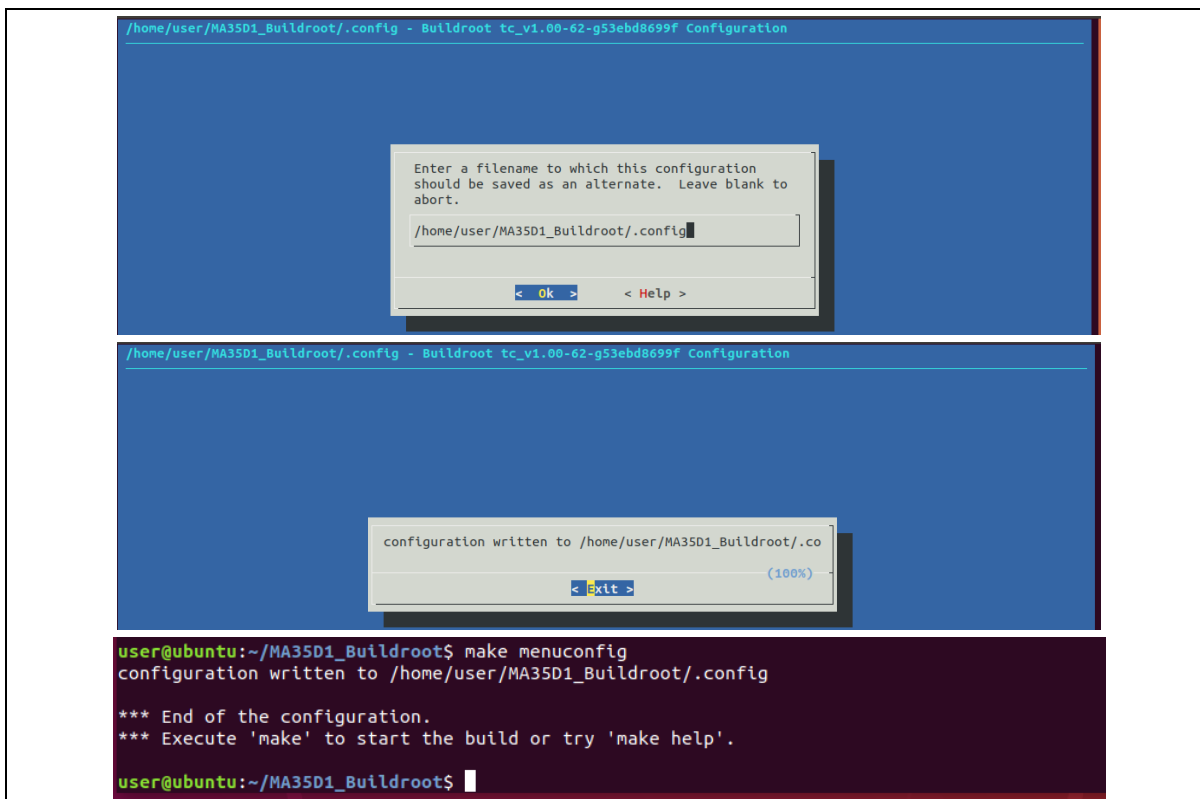


Figure 5-17 Save and Update the Configs

3. Config the LCD panel of device tree.

- Find the "nuc980.dtsi" file from MA35D1_Buildroot folder and add the following contents:

nuc980.dtsi

```
adc0 {
    pinctrl_adc0: adc0 {
```

```

        nuvoton,pins =
            <1 0x4 0x8 0
            1 0x5 0x8 0
            1 0x6 0x8 0
            1 0x7 0x8 0
            >;
    };
};

```

```

681         nadc {
682             pinctrl_nadc: nadc {
683                 nuvoton,pins =
684                     <1 0x0 0x8 0
685                     1 0x1 0x8 0
686                     1 0x2 0x8 0
687                     1 0x3 0x8 0
688                     1 0x4 0x8 0
689                     1 0x5 0x8 0
690                     1 0x6 0x8 0
691                     1 0x7 0x8 0
692                     >;
693             };
694         };
695
696         adc0 {
697             pinctrl_adc0: adc0 {
698                 nuvoton,pins =
699                     <1 0x4 0x8 0
700                     1 0x5 0x8 0
701                     1 0x6 0x8 0
702                     1 0x7 0x8 0
703                     >;
704             };
705         };

```

Figure 5-18 Added Code of “nuc980.dtsi”

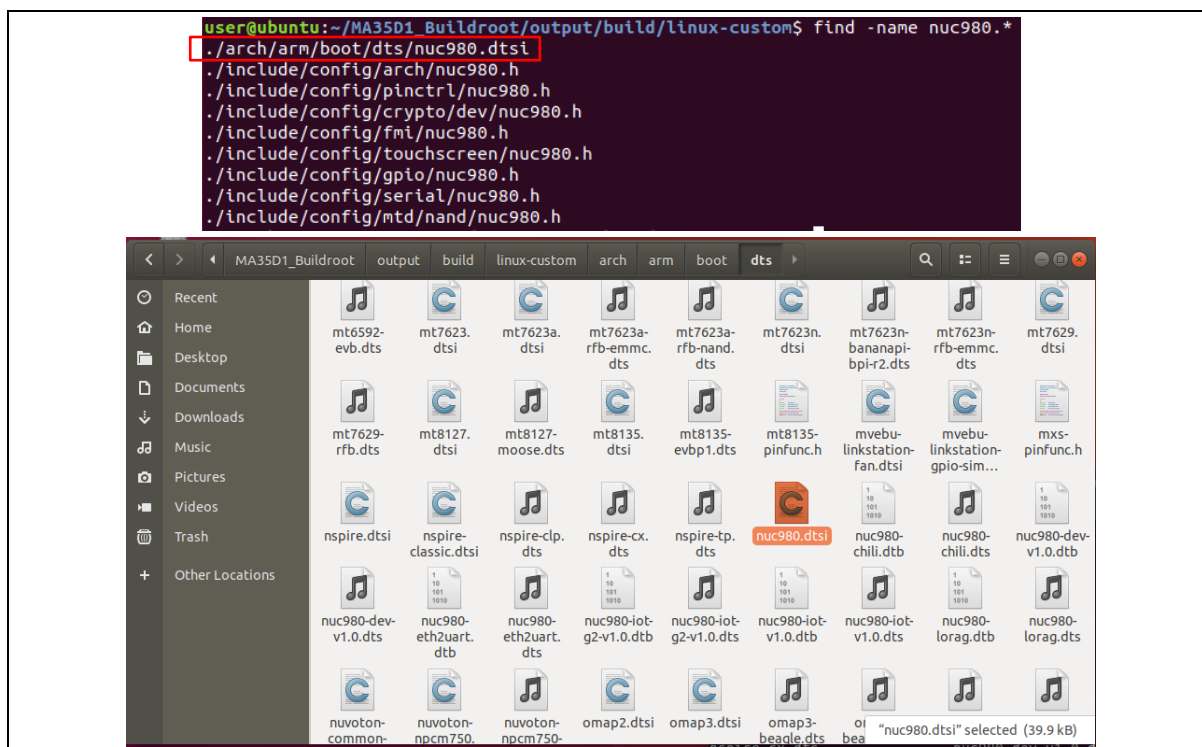


Figure 5-19 Location of “nuc980.dtsi”

- Find the “nuc980-iot-v1.0.dts” file from **MA35D1_Buildroot** folder and update the contents:

nuc980-iot-v1.0.dts

```
...
apb {
    uart1: serial@b0071000 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_uart1_PF>;
        pdma-enable = <1>;
        status = "disabled";
    };

    uart2: serial@b0072000 {
        status = "disabled";
    };
};
...
```

```

24     apb {
25         uart1: serial@b0071000 {
26             pinctrl-names = "default";
27             pinctrl-0 = <&pinctrl_uart1_Pf>;
28             pdma-enable = <1>;
29             status = "disabled";
30         };
31
32         uart2: serial@b0072000 {
33             status = "disabled";
34         };
35     };

```

Figure 5-20 Updated Part of “nuc980-iot-v1.0.dts”

- Find the “**nuc980-iot-v1.0.dts**” file from **MA35D1_Buildroot** folder and add the following contents:

nuc980-iot-v1.0.dts

```

...
adc0: adc0@b0043000 {
    compatible = "nuvoton,nuc980-adc";
    reg = <0xb0043000 0x100>;
    interrupts = <18 4 1>;
    map-addr = <0xf0043000>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_adc0>;
    status = "okay";

    clock-rate = <1000000>;
    enable-iiio;
    enable-ts;
    ts-type = <4>;
};
...

```

```

91     nadc: nadc@b0043000 {
92         status = "disabled";
93     };
94
95     adc0: adc0@b0043000 {
96         compatible = "nuvoton,nuc980-adc";
97         reg = <0xb0043000 0x100>;
98         interrupts = <18 4 1>;
99         map-addr = <0xf0043000>;
100        pinctrl-names = "default";
101        pinctrl-0 = <&pinctrl_adc0>;
102        status = "okay";
103
104        clock-rate = <1000000>;
105        enable-iio;
106        enable-ts;
107        ts-type = <4>;
108    };
109    pwm0: pwm@b0058000 {
110        status = "disabled";
111    };
112

```

Figure 5-21 Added Part of “nuc980-iot-v1.0.dts” (1)

nuc980-iot-v1.0.dts

```

...
spi_lcd@0 {
    fps = <30>;
    buswidth = <8>;
    bgr;
    dc-gpios = <&gpio 0xA9 GPIO_ACTIVE_HIGH>; // PF.9 as C/D pin
    debug = <0x0>;
    compatible = "ilitek,ili9341";
    reg = <0>;
    spi-max-frequency = <50000000>;
};
...

```

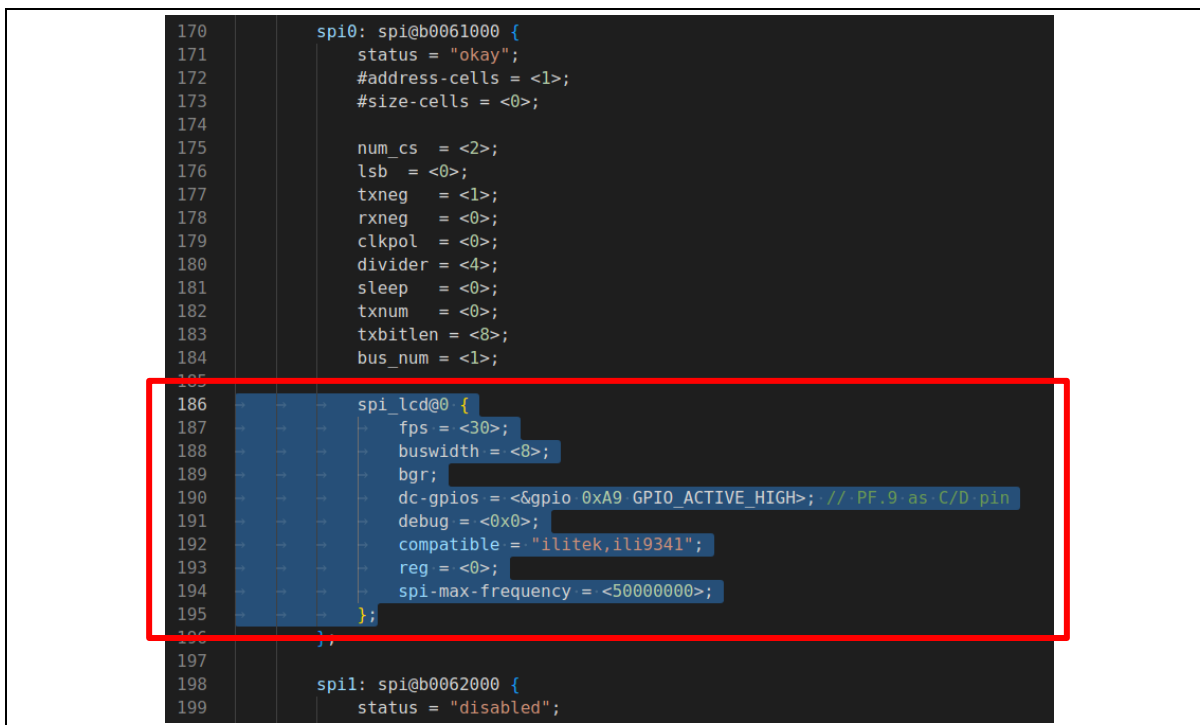



Figure 5-22 Added Part of "nuc980-iot-v1.0.dts" (2)

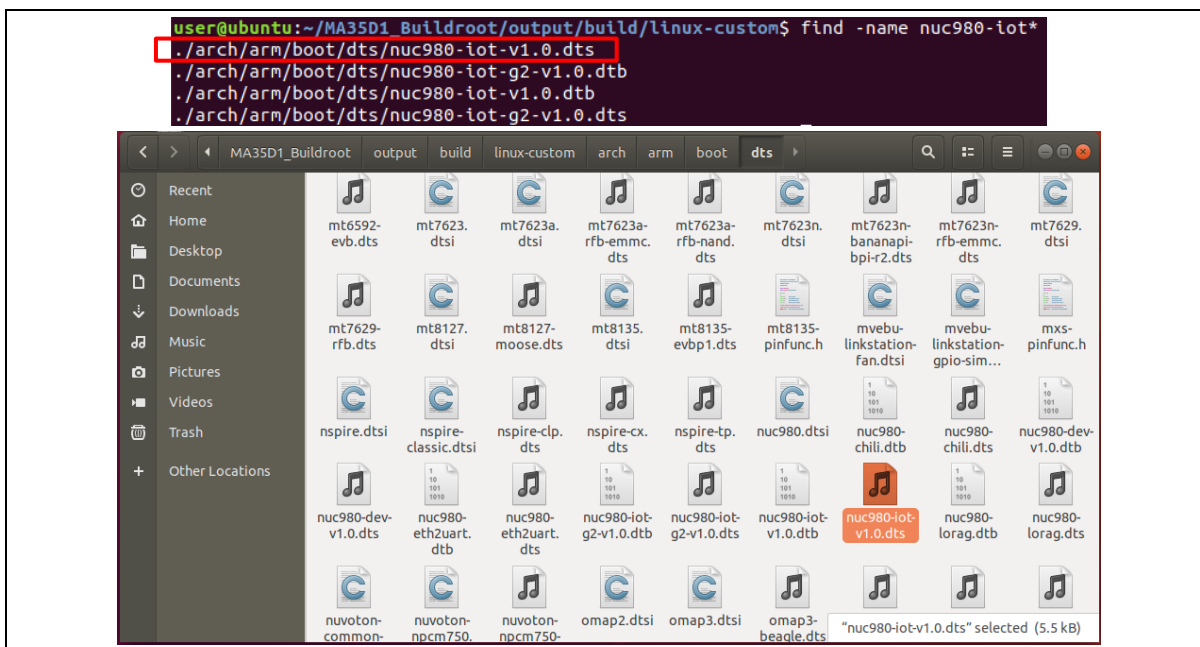


Figure 5-23 Location of "nuc980.dtsi"

4. Back to the folder path of **MA35D1_Buildroot**, rebuild the Linux device tree and image files.

```
$ make linux-rebuild && make
```

```
user@ubuntu:~/MA35D1_Buildroot$ make linux-rebuild && make
rm -f /home/user/MA35D1_Buildroot/output/build/linux-custom/.stamp_installed
rm -f /home/user/MA35D1_Buildroot/output/build/linux-custom/.stamp_staging_installed
rm -f /home/user/MA35D1_Buildroot/output/build/linux-custom/.stamp_target_installed
rm -f /home/user/MA35D1_Buildroot/output/build/linux-custom/.stamp_images_installed
rm -f /home/user/MA35D1_Buildroot/output/build/linux-custom/.stamp_host_installed
rm -f /home/user/MA35D1_Buildroot/output/build/linux-custom/.stamp_built
>>> linux custom Building
/bin/sed -i -e "/\\<CONFIG_COMMON_CLK_FIXED_UNUSED\\>/d" /home/user/MA35D1_Buildroot/output/build/linux-custom//.config
echo '# CONFIG_COMMON_CLK_FIXED_UNUSED is not set' >> /home/user/MA35D1_Buildroot/output/build/linux-custom//.config
/bin/sed -i -e "/\\<CONFIG_GCC_PLUGINS\\>/d" /home/user/MA35D1_Buildroot/output/build/linux-custom//.config
echo '# CONFIG_GCC_PLUGINS is not set' >> /home/user/MA35D1_Buildroot/output/build/linux-custom//.config
if grep -q "CONFIG_ARCH_NUC980=y" /home/user/MA35D1_Buildroot/output/build/linux
```

Figure 5-24 Rebuilding the Image File

5. Update image files on the evaluation board and check whether the following configurations and files existed by log file:

- The initial of “fb_ili9341”.
- whether the setting of “adc0 probe” succeeded.
- whether file “/dev/fb0” exist.
- whether file “/dev/input/event0” exist.

```
Run /sbin/init as init process
Starting syslogd: OK
Starting klogd: OK
Running sctd: OK
Starting mdev... OK
RX nuc980 spi0_probe: dma0chan2 module removed
TX nuc980 spi0_probe: dma0chan3 module removed
fb_ili9341 spi1.0: fbttft_property_value: buswidth = 8
fb_ili9341 spi1.0: fbttft_property_value: debug = 0
fb_ili9341 spi1.0: fbttft_property_value: fps = 10
graphics fb0: fb_ili9341 frame buffer, 240x320, 150 KiB video memory, 16 KiB buffer memory, fps=10, spi1.0 at 15 MHz
Saving random seed: random: dd: uninitialized urandom read (512 bytes read)
OK
Starting network: OK
Welcome to Buildroot
buildroot login: root
```

Figure 5-25 Logs of Power-on

```
Welcome to Buildroot
buildroot login: root
# cd /dev
# ls
console      ptyv1        ptyvb        tty22        ttyd8        ttyu2
fb0          ptyv2        ptyvc        tty23        ttyd9        ttyu3
fd           ptyv3        ptyvd        tty24        ttyda        ttyu4
full         ptyv4        ptyve        tty25        ttydb        ttyu5
gpiochip0    ptyv5        ptyvf        tty26        ttydc        ttyu6
l10:device0  ptyv6        ptyw0        tty27        ttydd        ttyu7
input        ptyv7        ptyw1        tty28        ttyde        ttyu8
l10:device0  ptyv8        ptyw2        tty29        ttydf        ttyu9
ptyvf        ptyv9        tty20        ttyd6        ttyu0
ptyv0        ptyva        tty21        ttyd7        ttyu1
# cd input/
# ls
event0
#
```

Figure 5-26 Location of Files

Additional tests (optional):

- Turn on backlight (pin PG.7).

```
$ echo 0xC7 > /sys/class/gpio/export
$ cd /sys/class/gpio/gpio199/
$ echo out > direction
$ echo 1 > value
```

- Test to write or clean data on `/dev/fb0`.

```
$ cat /dev/urandom > /dev/fb0
$ dd if=/dev/zero of=/dev/fb0
```

- Prepare program and set env.

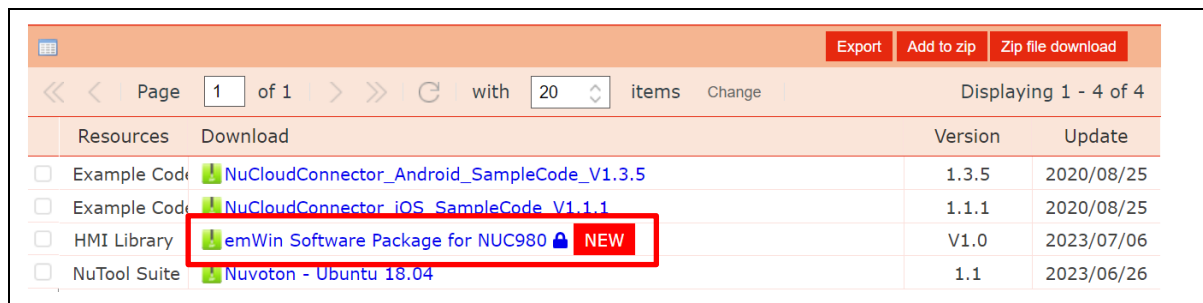
```
$ cp ./ts.conf /etc/
$ export TSLIB_CALIBFILE='/etc/pointercal'
$ export TSLIB_CONFFILE='/etc/ts.conf'
$ export TSLIB_TSDEVICE='/dev/input/event0'
```

- Calibration & test.

```
$ ./ts_calibrate
$ ./ts_test
```

5.2.2 Demo of HMI Example

Taking the emWin AppWizard project as an example, user can download the emWin package from the Nuvoton website: <https://www.nuvoton.com/products/iot-solution/iot-platform/numaker-iiot-nuc980/?group=Software&tab=2>



Resources	Download	Version	Update
<input type="checkbox"/> Example Code	NuCloudConnector_Android_SampleCode_V1.3.5	1.3.5	2020/08/25
<input type="checkbox"/> Example Code	NuCloudConnector_iOS_SampleCode_V1.1.1	1.1.1	2020/08/25
<input type="checkbox"/> HMI Library	emWin Software Package for NUC980 NEW	V1.0	2023/07/06
<input type="checkbox"/> NuTool Suite	Nuvoton - Ubuntu 18.04	1.1	2023/06/26

Figure 5-27 Location of emWin AppWizard Package

Unzip the package and follow the readme file to make the GUI bin file then copy it to the SD card. Users can connect the VCOM port of the evaluation board, insert the SD card after power-on and execute the bin file.

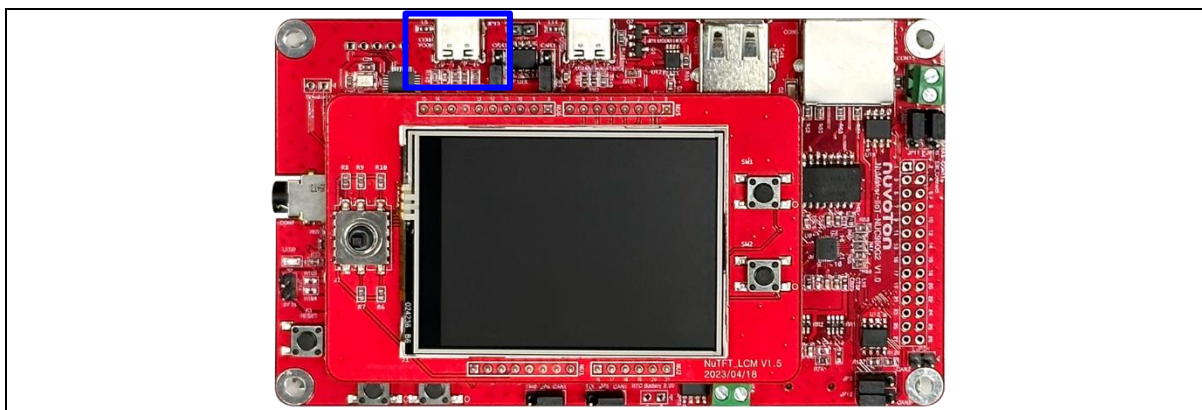


Figure 5-28 VCOM of Evaluation Board

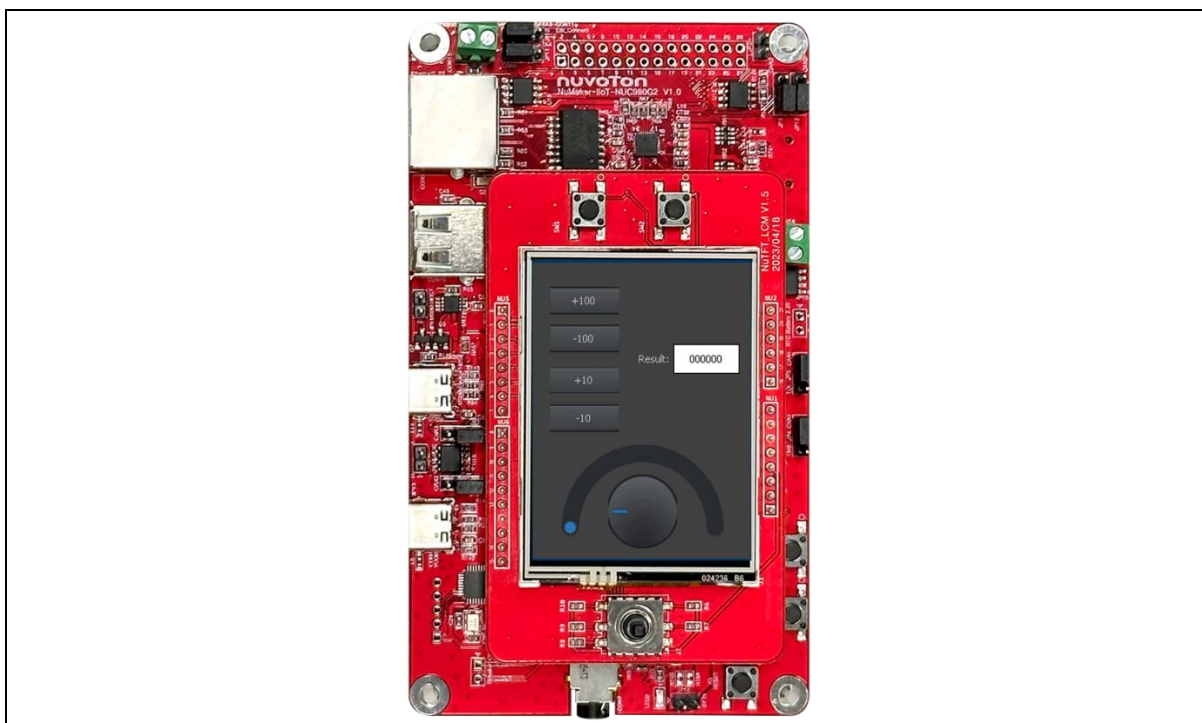


Figure 5-29 Demo of the EmWin AppWizard

Through this example, users can display the designed graphic user interface(GUI). For details on the operation of emWin AppWizard operation, refer to the document "*NUC980 emWin Quick Start Guide*" and "*NUC980 emWin appWizard Quick Start Guide*" of the emWin package.

6 REVISION HISTORY

Date	Revision	Description
2020.05.20	1.00	<ul style="list-style-type: none"> Initial version.
2023.08.03	1.01	<ul style="list-style-type: none"> Changed the document name and some headings. Updated VMware Virtual Machine installation steps in section 2.1. Updated VMware resource. Updated for Linux kernel 5.10.
2023.09.07	1.02	<ul style="list-style-type: none"> Updated the heading name of section 3.1.3 and 3.2.3. Removed section 2.3 due to resource consolidation. Added chapter 4 to explain how to update the environment variables.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

*Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*