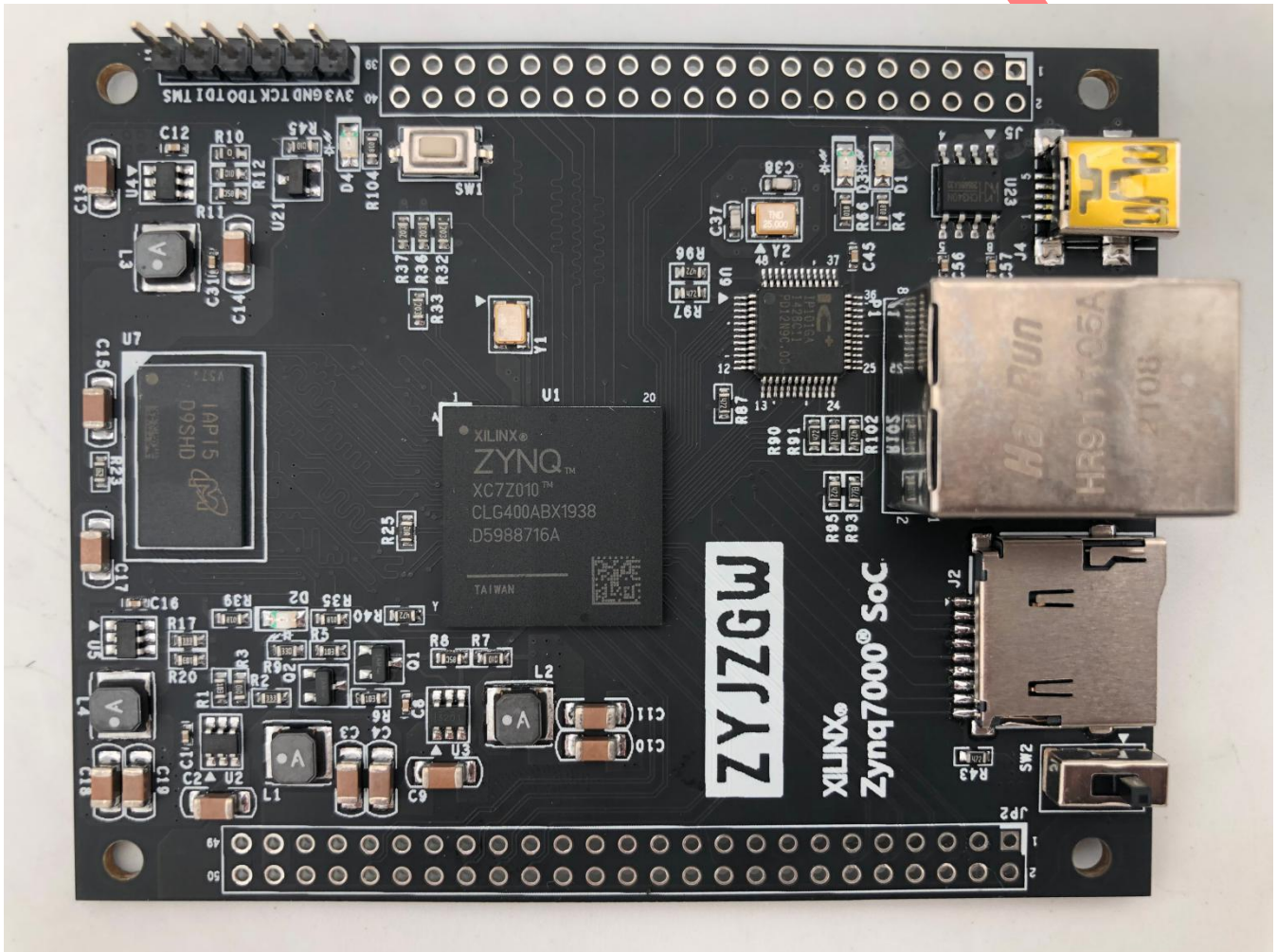


# ZYJZGW ZYNQ XC7Z010 STARTER KIT

## USER MANUAL



## Preface

The ZYJZGW® ZYNQ XC7Z010 Starter Kit uses Xilinx Zynq®-7000 device which integrates the software programmability of an ARM®-based processor with the hardware programmability of an FPGA, enabling key analytics and hardware acceleration while integrating CPU, DSP, ASSP, and mixed signal functionality on a single device. Consisting of single-core Zynq-7000S and dual-core Zynq-7000 devices, the Zynq-7000 family is the best price to performance-per-watt, fully scalable SoC platform for your unique application requirements.

## Table of Contents

<b>1. INTRODUCTION.....</b>	<b>3</b>
1.1 DOCUMENT SCOPE.....	3
1.2 PREPARATION.....	3
<b>2. GETTING STARTED.....</b>	<b>5</b>
2.1 PREPARE THE LINUX ENVIRONMENT.....	5
2.2 STEPS TO CUSTOMIZE THE U-BOOT.....	7
2.3 STEPS TO CUSTOMIZE THE LINUX OS.....	11
2.4 STEPS TO MAKE THE FILE SYSTEM.....	13
2.5 TEST THE ETHERNET UNDER LINUX ENVIRONMENT.....	14
<b>3. REFERENCE.....</b>	<b>16</b>
<b>4. REVISION.....</b>	<b>17</b>

# 1. Introduction

## 1.1 Document Scope

This user manual introduces the procedure to make the Linux environment running on the ZYJZGW ZYNQ XC7Z010 Starter Kit. The Linux environment mainly covers three parts: u-boot, Linux OS and file system. All of those parts are developed with Xilinx Vivado 2018.3 under Ubuntu 18.04 environment. The prerequisites before working with the Linux parts are shown as below:

1. Users have already installed the Vivado 2018.3 in Ubuntu. Preferred and verified version is Ubuntu 18.04.
2. Users have the basic knowledge about the usage of the Linux environment. Know how to use the cross-compile toolchain including arm-gcc-linux-, makefile, etc.

## 1.2 Preparation

Here lists some useful resources from the internet.

Below Xilinx Wiki offers everything necessary to customize, build and deploy Embedded Linux solutions on Xilinx processing systems. Tailored to accelerate design productivity, the solution works with the Xilinx hardware design tools to ease the development of Linux systems for Zynq®-7000 SoCs.

<https://xilinx-wiki.atlassian.net/wiki/spaces/A/overview>

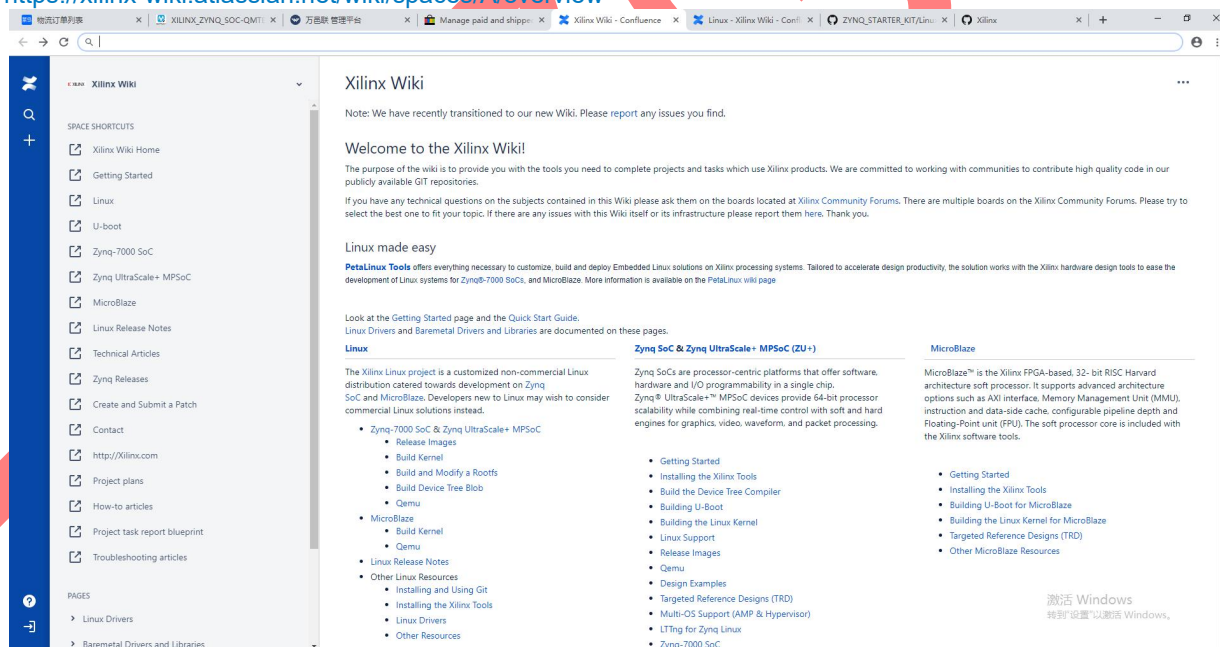


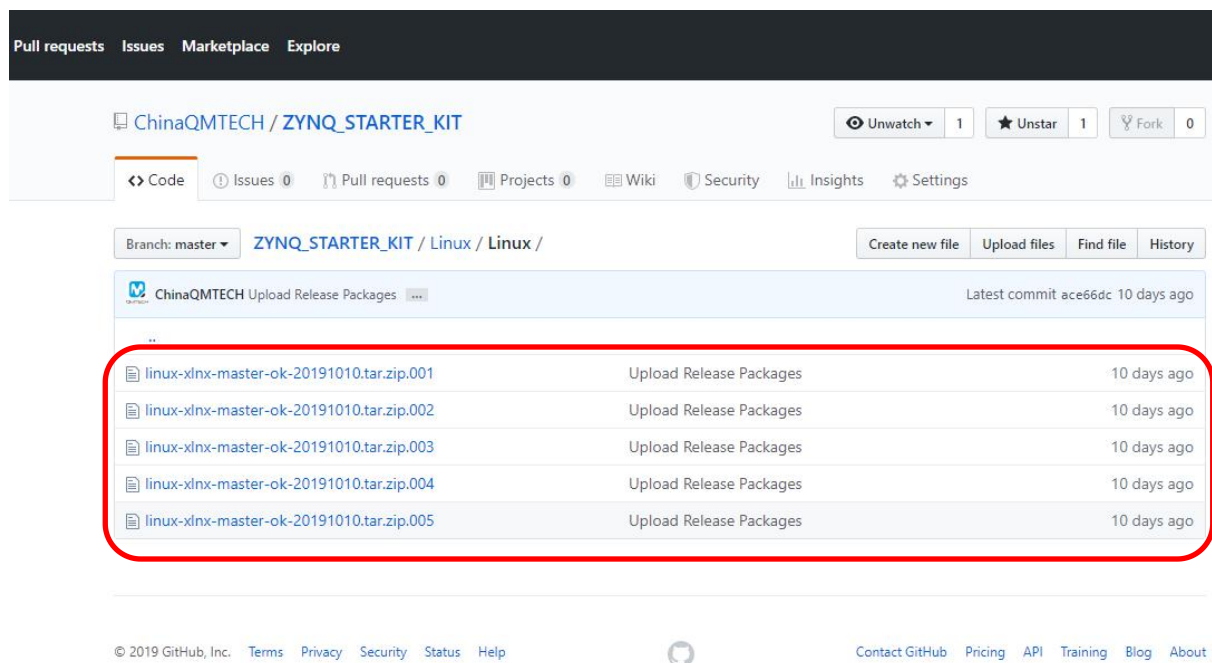
Figure 1-1. Xilinx Wiki

Below Github provides the official Xilinx Linux kernel and u-boot sources for Zynq®-7000 SoCs. ZYJZGW customized Linux kernel and u-boot sources are all from this repository: <https://github.com/Xilinx>

Below lists the linux source and u-boot Github source repository for ZYJZGW ZYNQ Starter Kit. Since these sources are already customized for the Starter Kit, users may download and build these sources directly. Nothing needs to be changed to make the Zynq XC7Z010 board running Linux environment.

[https://github.com/ChinaQMTECH/ZYJZGW\\_ZYNQ\\_STARTER\\_KIT/tree/master/Linux](https://github.com/ChinaQMTECH/ZYJZGW_ZYNQ_STARTER_KIT/tree/master/Linux)  
[http://www.chinaqmttech.com/xilinx\\_zynq\\_soc](http://www.chinaqmttech.com/xilinx_zynq_soc)

Since the customized Linux package is too large, users shall merge these five zip file into one and then unzip it.



Pull requests Issues Marketplace Explore

ChinaQMTECH / ZYNQ\_STARTER\_KIT

Unwatch 1 Unstar 1 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: master ZYNQ\_STARTER\_KIT / Linux / Linux /

Create new file Upload files Find file History

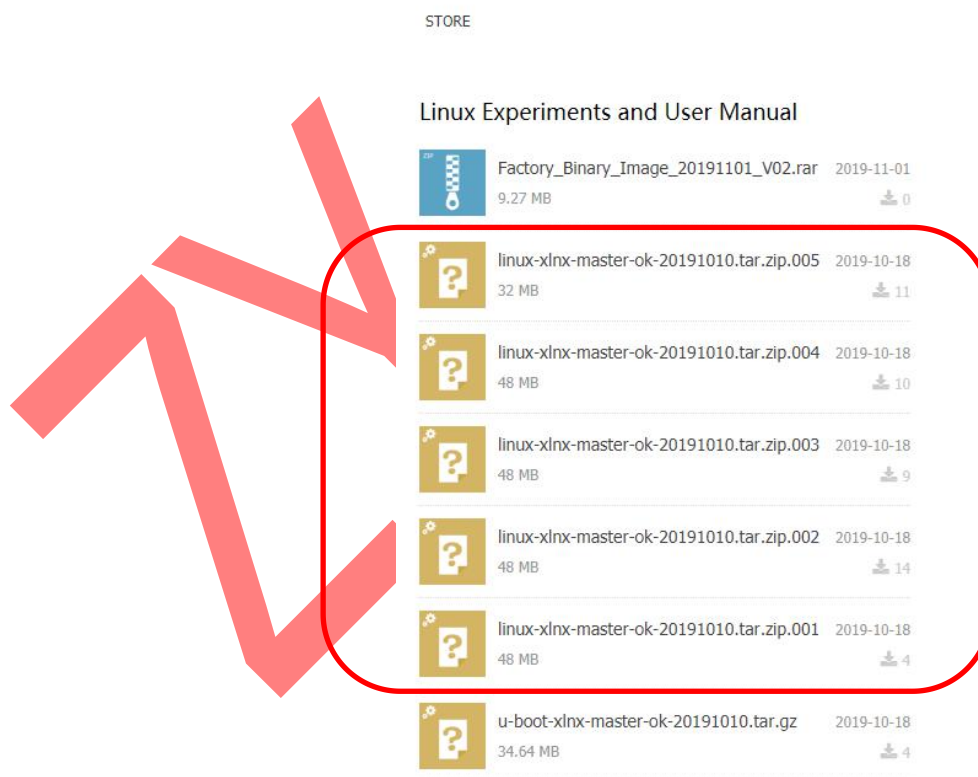
ChinaQMTECH Upload Release Packages Latest commit ace66dc 10 days ago

linux-xlnx-master-ok-20191010.tar.zip.001	Upload Release Packages	10 days ago
linux-xlnx-master-ok-20191010.tar.zip.002	Upload Release Packages	10 days ago
linux-xlnx-master-ok-20191010.tar.zip.003	Upload Release Packages	10 days ago
linux-xlnx-master-ok-20191010.tar.zip.004	Upload Release Packages	10 days ago
linux-xlnx-master-ok-20191010.tar.zip.005	Upload Release Packages	10 days ago

© 2019 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub Pricing API Training Blog About

Figure 1-2. Customized Linux Packages



STORE

Linux Experiments and User Manual

Factory_Binary_Image_20191101_V02.rar	2019-11-01
9.27 MB	0
linux-xlnx-master-ok-20191010.tar.zip.005	2019-10-18
32 MB	11
linux-xlnx-master-ok-20191010.tar.zip.004	2019-10-18
48 MB	10
linux-xlnx-master-ok-20191010.tar.zip.003	2019-10-18
48 MB	9
linux-xlnx-master-ok-20191010.tar.zip.002	2019-10-18
48 MB	14
linux-xlnx-master-ok-20191010.tar.zip.001	2019-10-18
48 MB	4
u-boot-xlnx-master-ok-20191010.tar.gz	2019-10-18
34.64 MB	4

Figure 1-3. Customized Linux Packages

## 2. Getting Started

This chapter describes the detailed steps to create a customized Linux environment. Comparing to the existing ZYNQ development board e.g. Xilinx ZC702, there are many differentiations in the ZYJZGW ZYNQ Starter Kit. For example, there's only one 16bit width DDR3 memory chip connected to PS ARM core. And the ethernet interface no longer uses the GEM1, it uses GEM0 instead. Hence, the u-boot/ Linux device tree for the Starter Kit needs to be updated here.

### 2.1 Prepare the Linux Environment

Below image shows the example setup for the VMware virtual machine. Suggested size for the hard disk is 100GB because the Vivado takes so much storage space.

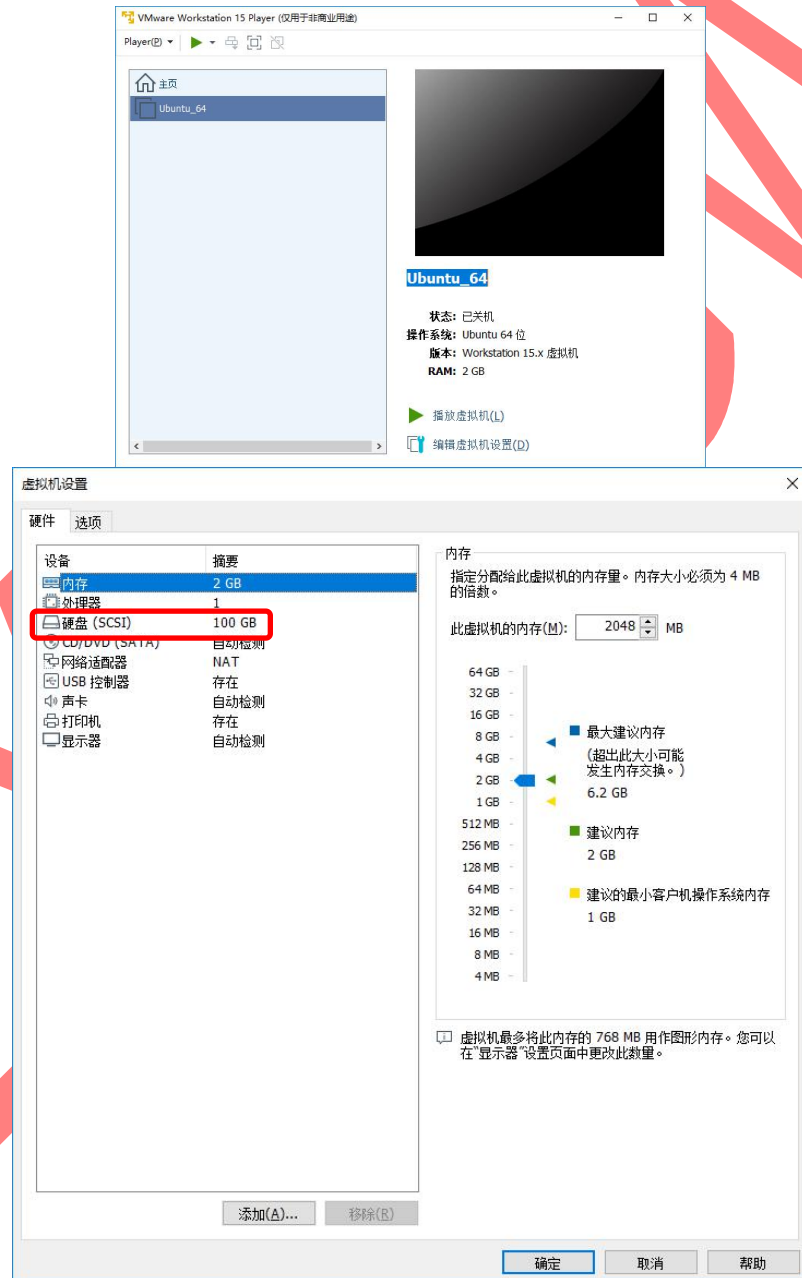
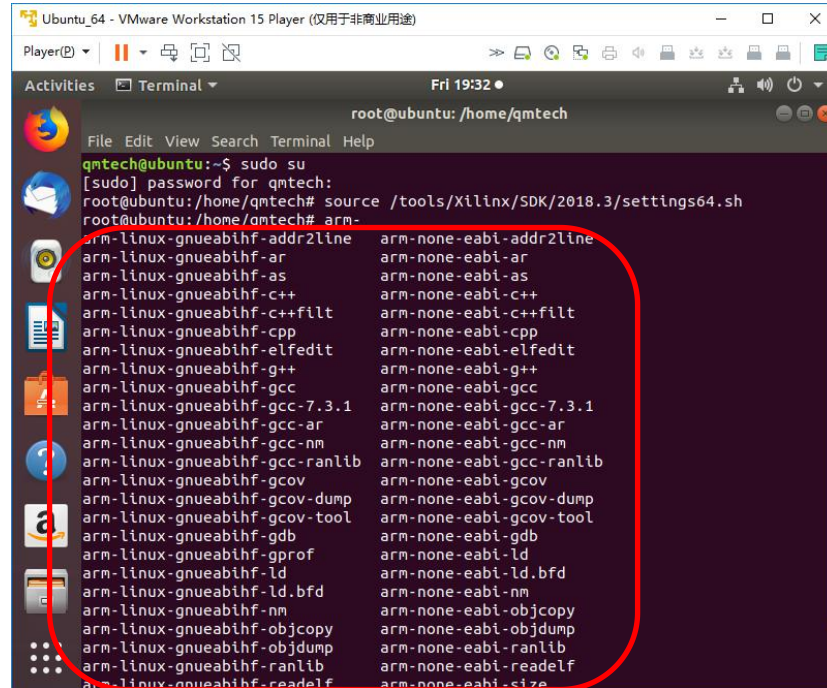


Figure 2-1. Virtual Machine Setup



Download the Xilinx Vivado 2018.3 Linux version from here. <https://www.xilinx.com/support/download.html>  
Copy the installation package `Xilinx_Vivado_SDK_2018.3_1207_2324.tar.gz` into Ubuntu. And unzip the package and install it.



## 2.2 Steps to Customize the u-boot

The u-boot customization is based upon the Xilinx ZC702 development board. The first thing to do is to modify the `u-boot/arch/arm/dts/zynq-zc702.dts`. Below image shows the modified content for the customized ZYJZGW Starter Kit.

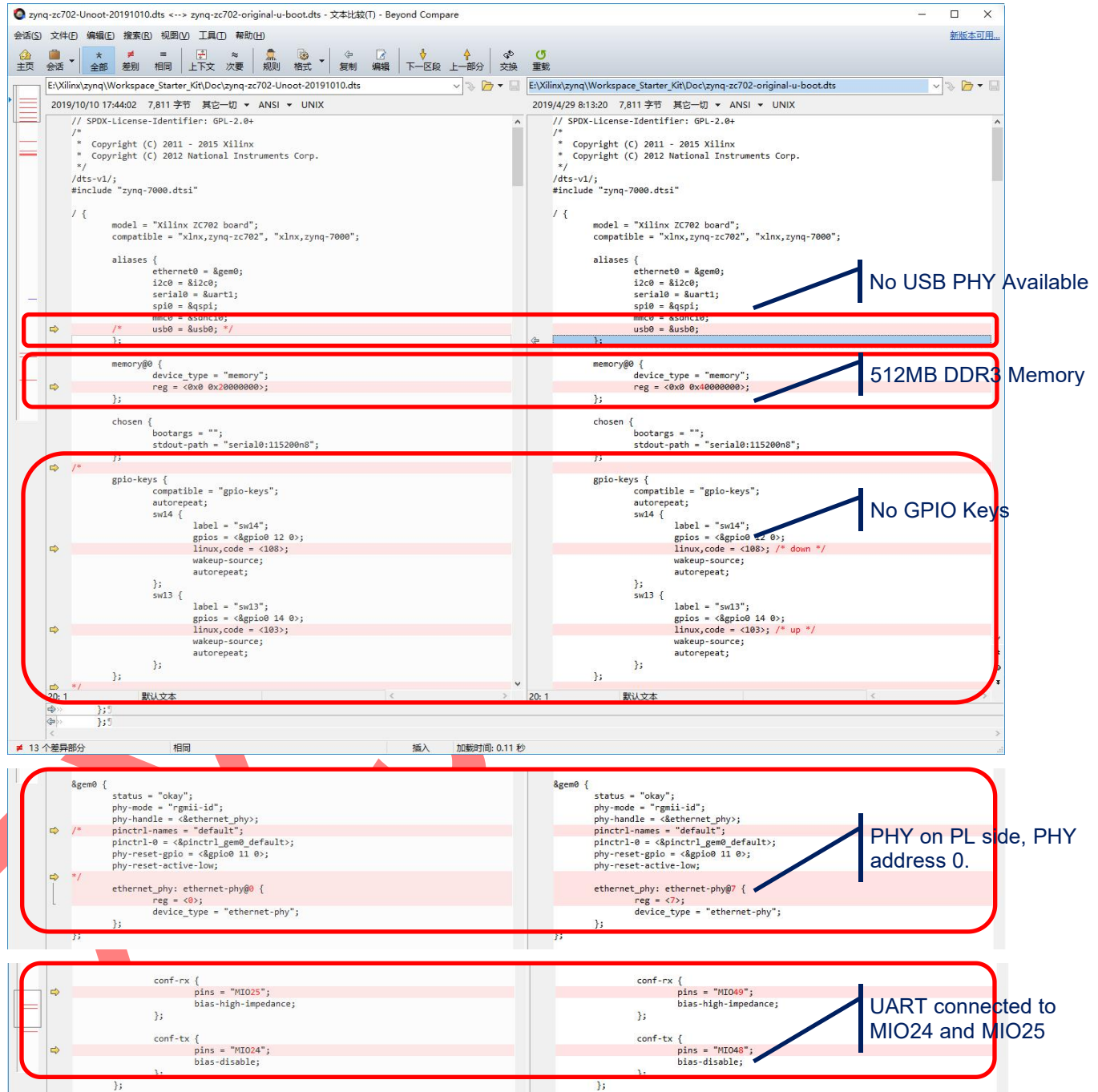
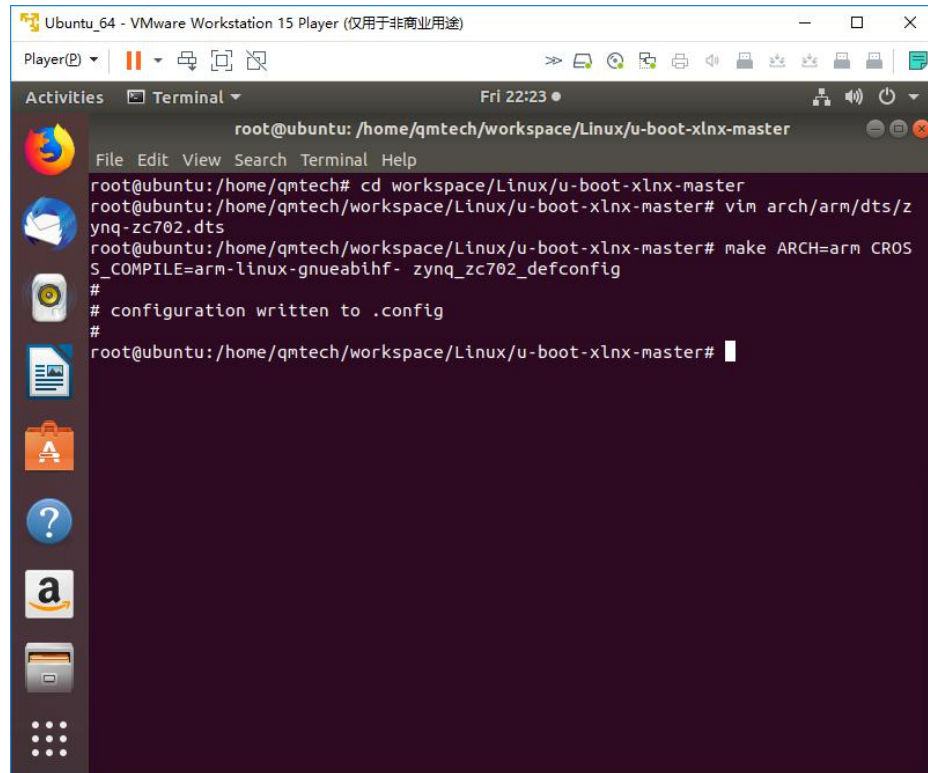


Figure 2-3. Modifications

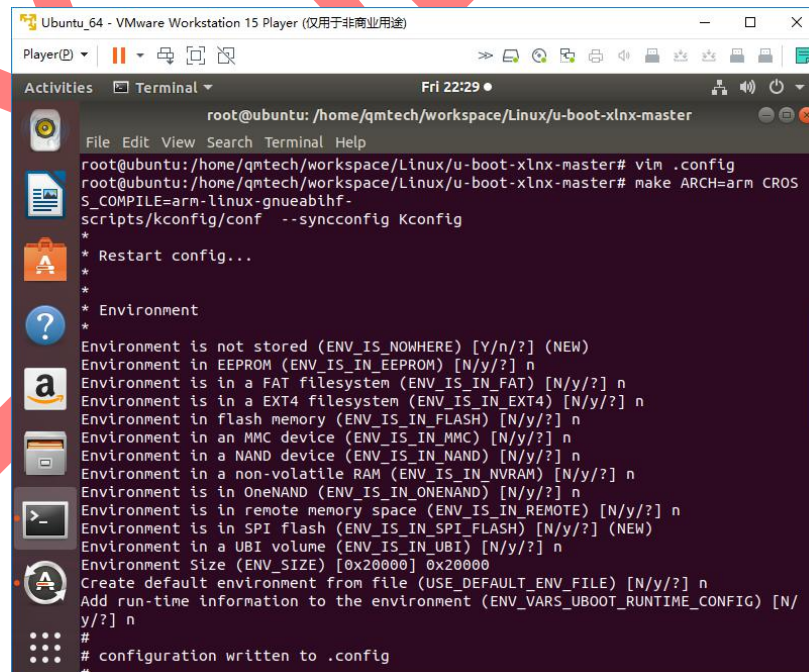
Type command in the terminal: `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-zynq_zc702_defconfig`. A new `.config` file will be generated.



```
root@ubuntu: /home/qmtech/workspace/Linux/u-boot-xlnx-master
root@ubuntu:/home/qmtech/workspace/Linux/u-boot-xlnx-master# vim arch/arm/dts/zynq-zc702.dts
root@ubuntu:/home/qmtech/workspace/Linux/u-boot-xlnx-master# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-zynq_zc702_defconfig
# configuration written to .config
#
root@ubuntu:/home/qmtech/workspace/Linux/u-boot-xlnx-master#
```

**Figure 2-4. Make ZC702 Defconfig**

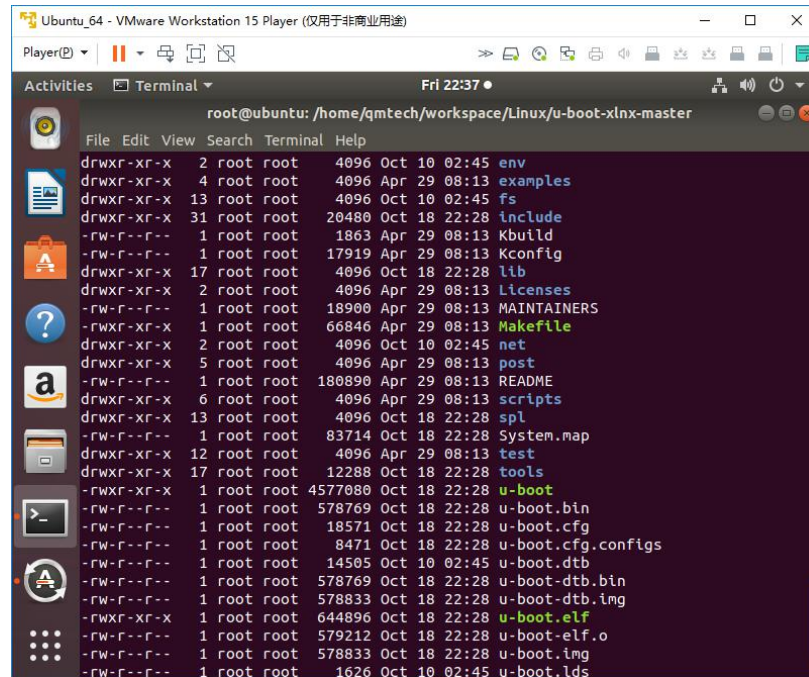
Use edit tool Vim to edit the `.config` file. Comment below code line `#CONFIG_ENV_IS_IN_SPI_FLASH = y`. And replace `CONFIG_BOOTCOMMAND="run distro bootcmd"` with `CONFIG_BOOTCOMMAND="run bootcmd_xilinx"`. And then type this command in the terminal: `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-`



```
root@ubuntu: /home/qmtech/workspace/Linux/u-boot-xlnx-master
root@ubuntu:/home/qmtech/workspace/Linux/u-boot-xlnx-master# vim .config
root@ubuntu:/home/qmtech/workspace/Linux/u-boot-xlnx-master# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-
scripts/kconfig/conf --syncconfig Kconfig
*
* Restart config...
*
* Environment
*
Environment is not stored (ENV_IS_NOWHERE) [Y/n/?] (NEW)
Environment in EEPROM (ENV_IS_IN_EEPROM) [N/y/?] n
Environment in a FAT filesystem (ENV_IS_IN_FAT) [N/y/?] n
Environment in a EXT4 filesystem (ENV_IS_IN_EXT4) [N/y/?] n
Environment in flash memory (ENV_IS_IN_FLASH) [N/y/?] n
Environment in an MMC device (ENV_IS_IN_MMC) [N/y/?] n
Environment in a NAND device (ENV_IS_IN_NAND) [N/y/?] n
Environment in a non-volatile RAM (ENV_IS_IN_NVRAM) [N/y/?] n
Environment is in OneNAND (ENV_IS_IN_ONENAND) [N/y/?] n
Environment is in remote memory space (ENV_IS_IN_REMOTE) [N/y/?] n
Environment is in SPI flash (ENV_IS_IN_SPI_FLASH) [N/y/?] (NEW)
Environment in a UBI volume (ENV_IS_IN_UBI) [N/y/?] n
Environment Size (ENV_SIZE) [0x20000] 0x20000
Create default environment from file (USE_DEFAULT_ENV_FILE) [N/y/?] n
Add run-time information to the environment (ENV_VARS_UBOOT_RUNTIME_CONFIG) [N/y/?] n
#
# configuration written to .config
#
```

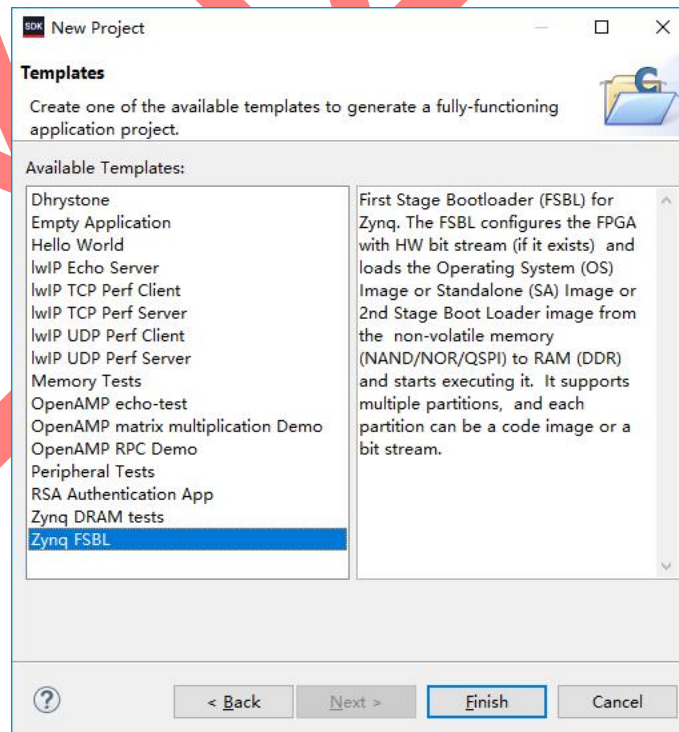


Once the u-boot compilation is successfully finished, there's one file named as **u-boot.elf** will be generated. That's the target file needed for building a u-boot image running on ZYNQ SoC. Below image shows the content that generated in u-boot folder.

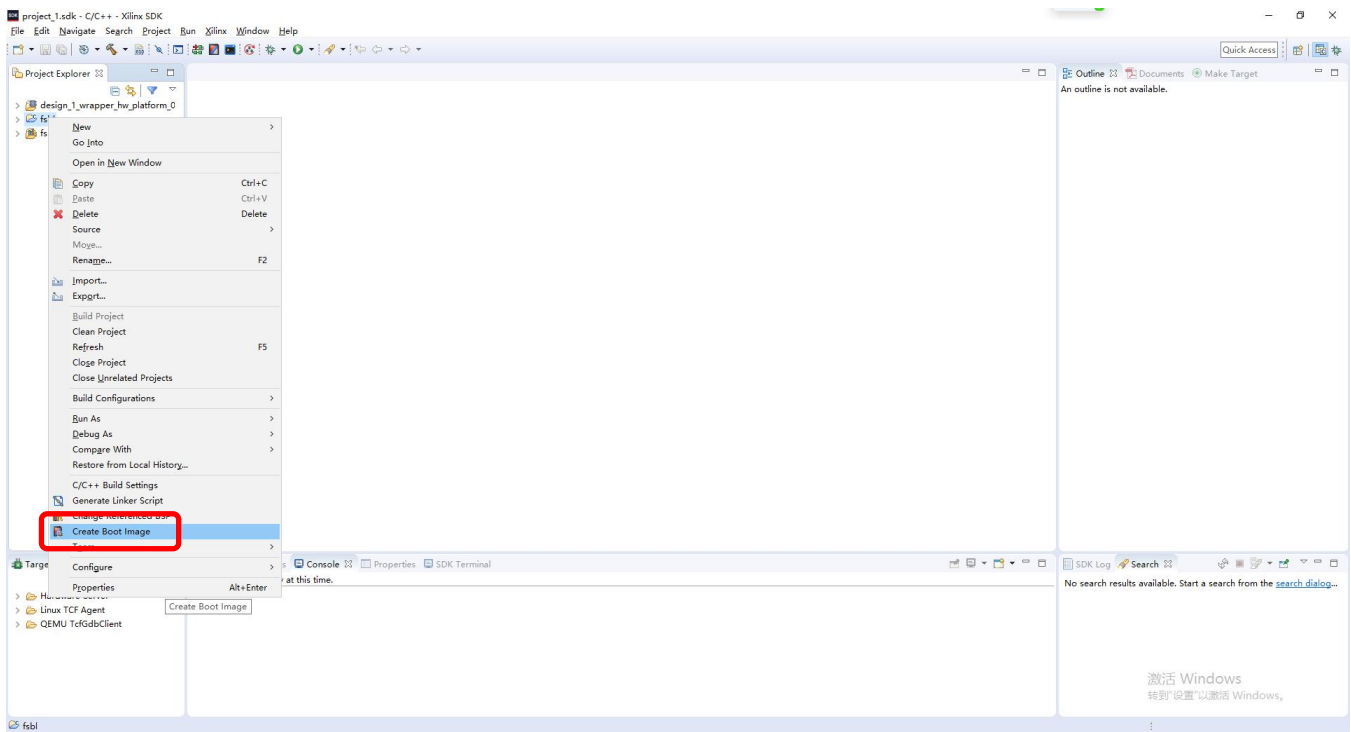


**Figure 2-5. U-boot Compilation**

Copy the **u-boot.elf** file into Windows system. E.g. put it into the folder Project04\_Uboot. Then open the project Project04\_Uboot. Make sure the compilation of this project brings no error and then **Launch SDK**. Click **[File]→[New]→[Application Project]** to create a new project named as **fsbl**. Select the Zynq FSBL as the project template.

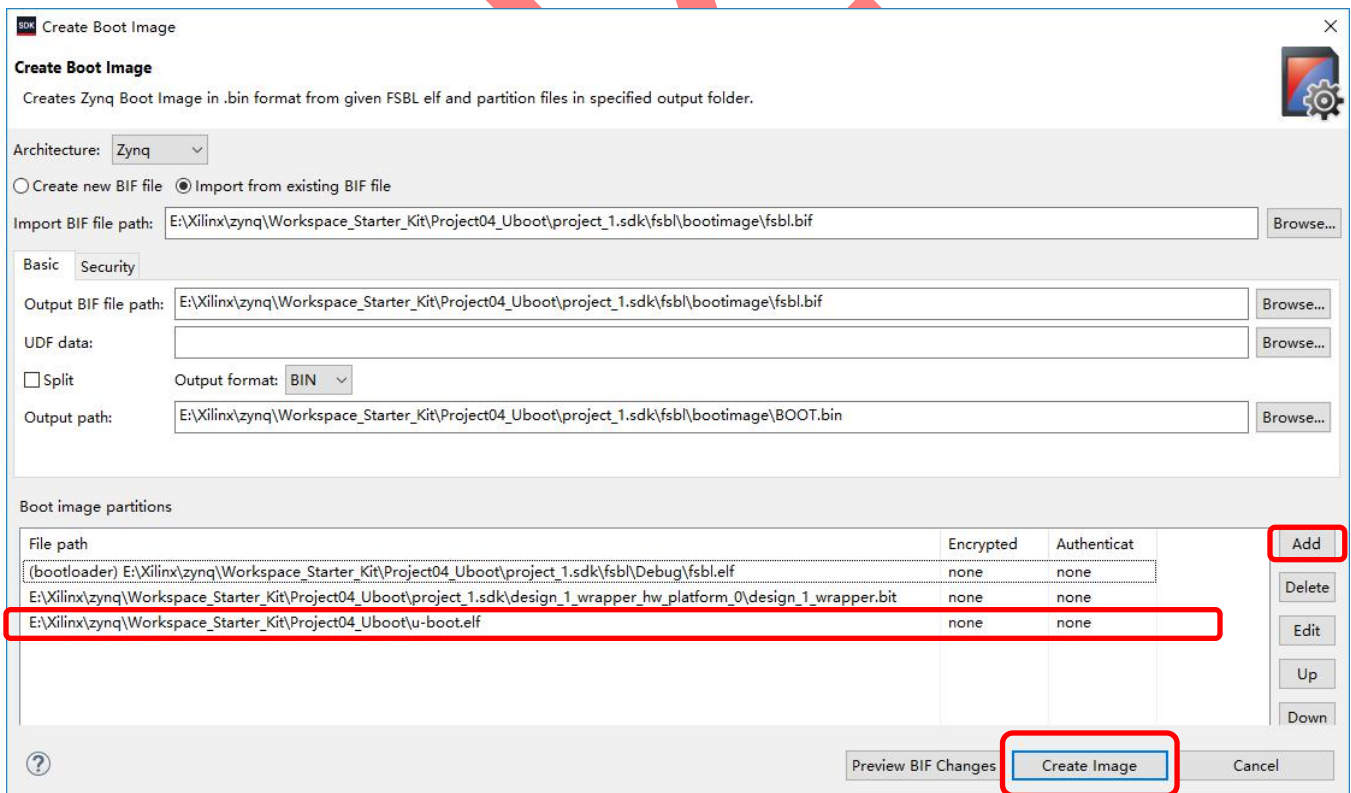


The fsbl project will be automatically compiled and then right click the project folder. Select **Create Boot Image**:



**Figure 2-6. Create Boot Image**

Choose the **u-boot.elf** when adding the Boot image partitions. Then click the **Create Image** button to generate the BOOT.bin which could be running directly on the ZYNQ SoC.



## 2.3 Steps to Customize the Linux OS

The Linux OS customization is based upon the Xilinx ZC702 development board. The first thing to do is to modify the `linux/arch/arm/boot/dts/zynq-zc702.dts`. Below image shows the modified content for the customized ZYJZGW Starter Kit.

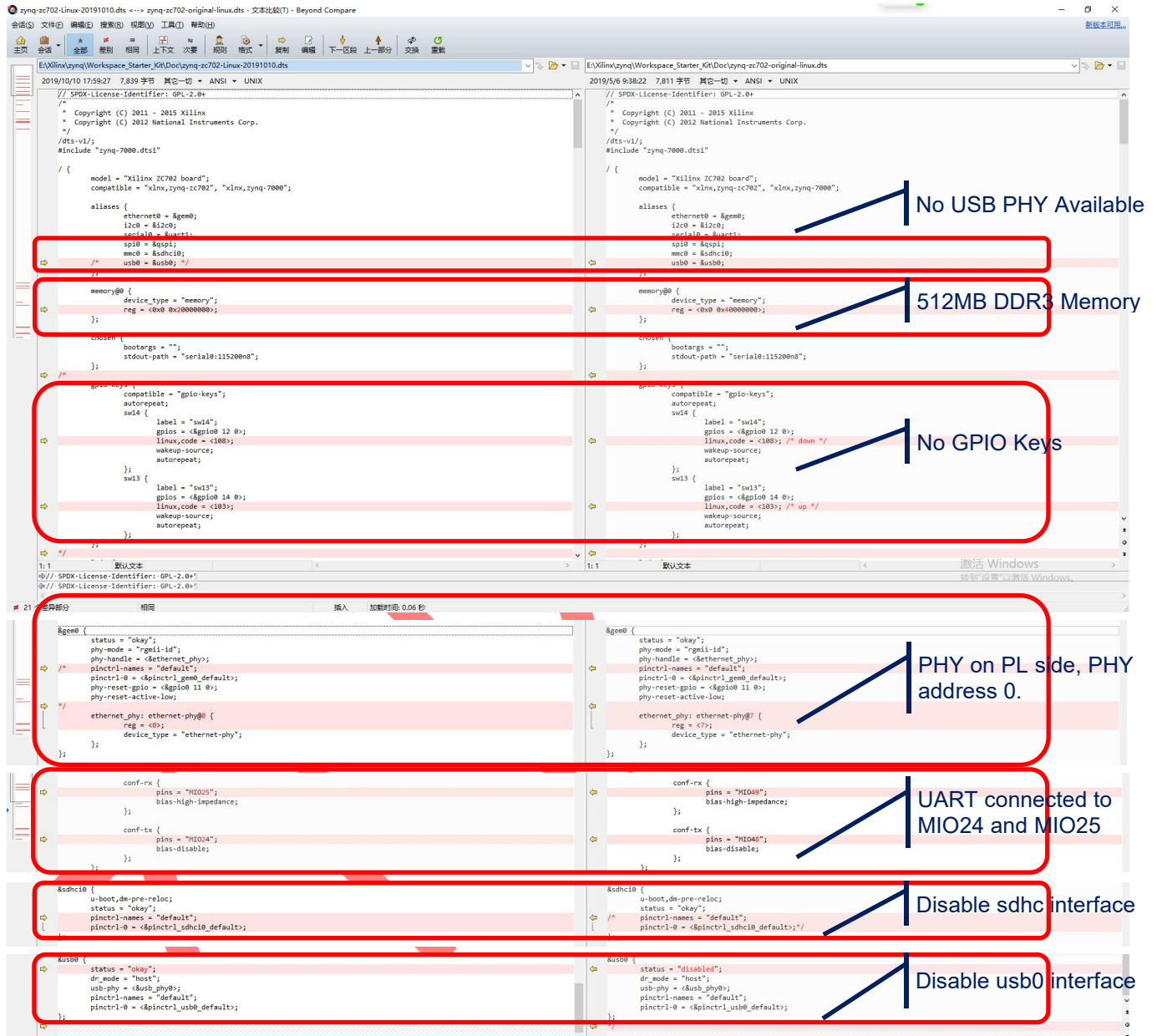
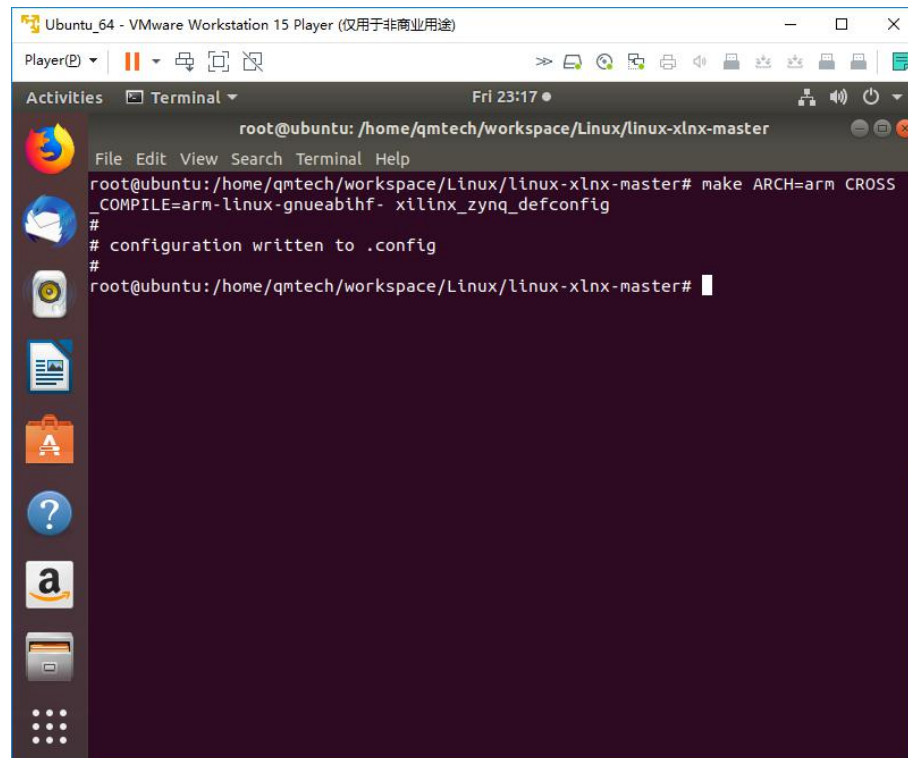


Figure 2-7. Modifications

Then type command in terminal window. `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-xilinx_zynq_defconfig`. A `.config` file will be generated.



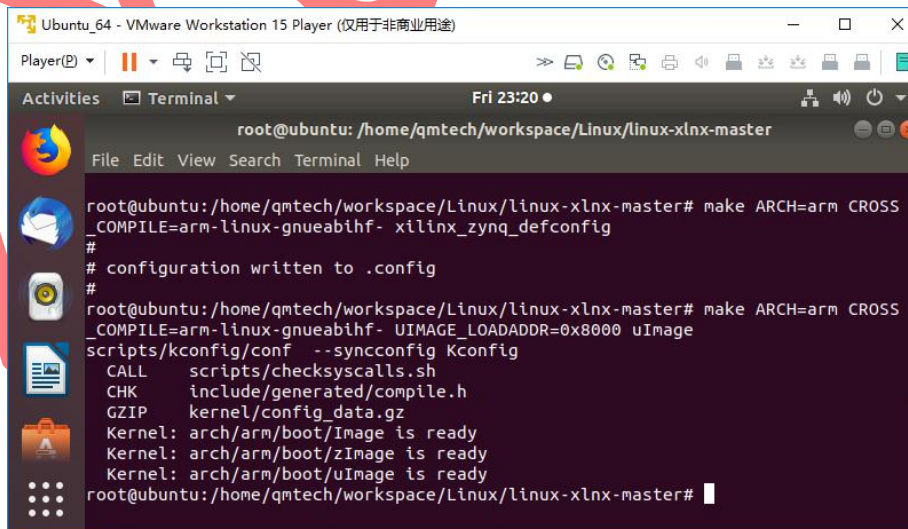
```
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-xilinx_zynq_defconfig
# configuration written to .config
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master#
```

**Figure 2-8. Make ZC702 Defconfig**

If users want to change some configuration in the `.config`, users may open the configuration menu in a graphical way by typing command `make ARCH=arm menuconfig`.

Then type below command in the terminal window to start compile the Linux OS: `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf- UIMAGE_LOADADDR=0x8000 uImage`

In the process, `linux-xlnx/arch/arm/boot/Image` and `linux-xlnx/arch/arm/boot/zImage` are created. The Image file is the uncompressed Linux kernel image and the zImage file is a compressed kernel image which will uncompress itself when it starts.

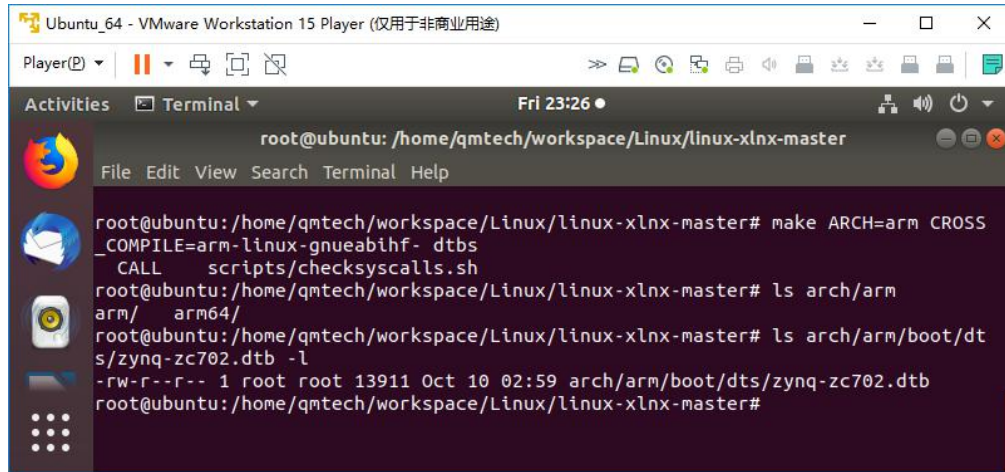


```
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-xilinx_zynq_defconfig
# configuration written to .config
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf- UIMAGE_LOADADDR=0x8000 uImage
scripts/kconfig/conf --syncconfig Kconfig
CALL scripts/checksyscalls.sh
CHK include/generated/compile.h
GZIP kernel/config_data.gz
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
Kernel: arch/arm/boot/uImage is ready
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master#
```

**Figure 2-9. Compilation**



Since the device tree also needs to be updated, users need to type command `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf-dtbs` in the terminal window. A `zynq-zc702.dtb` will be generated in folder:



```

root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master
File Edit View Search Terminal Help

root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master# make ARCH=arm CROSS
_COMPILE=arm-linux-gnueabi-hf-dtbs
CALL scripts/checksyscalls.sh
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master# ls arch/arm
arm/ arm64/
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master# ls arch/arm/boot/dt
s/zynq-zc702.dtb -l
-rw-r--r-- 1 root root 13911 Oct 10 02:59 arch/arm/boot/dts/zynq-zc702.dtb
root@ubuntu: /home/qmtech/workspace/Linux/linux-xlnx-master#

```

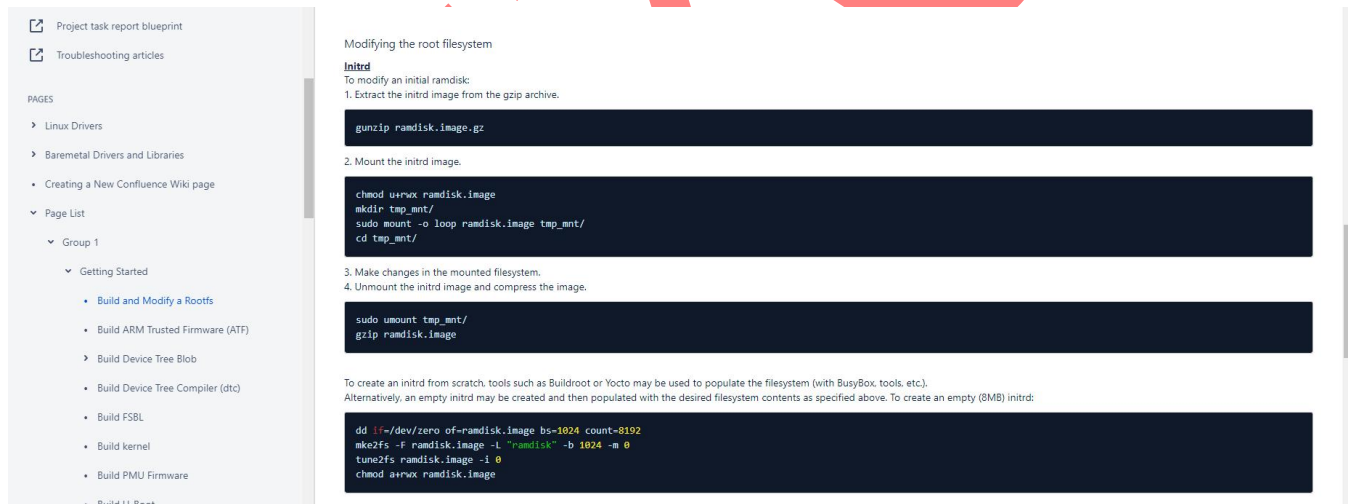
Figure 2-10. Update Device Tree

## 2.4 Steps to Make the File System

Users shall download the file `arm_ramdisk.image.gz` from below Xilinx Wiki and copy it into Ubuntu.

<https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842473/Build+and+Modify+a+Rootfs>

And then follow the modification guide shown in the above Xilinx Wiki.



Modifying the root filesystem

**initrd**  
To modify an initial ramdisk:

1. Extract the initrd image from the gzip archive.

```
gunzip ramdisk.image.gz
```

2. Mount the initrd image.

```
chmod u+rwx ramdisk.image
mkdir tmp_mnt/
sudo mount -o loop ramdisk.image tmp_mnt/
cd tmp_mnt/
```

3. Make changes in the mounted filesystem.
4. Unmount the initrd image and compress the image.

```
sudo umount tmp_mnt/
gzip ramdisk.image
```

To create an initrd from scratch, tools such as Buildroot or Yocto may be used to populate the filesystem (with BusyBox, tools, etc). Alternatively, an empty initrd may be created and then populated with the desired filesystem contents as specified above. To create an empty (8MB) initrd:

```
dd if=/dev/zero of=ramdisk.image bs=1024 count=8192
mke2fs -F ramdisk.image -L "ramdisk" -b 1024 -m 0
tune2fs ramdisk.image -i 0
chmod a+rwx ramdisk.image
```

Figure 2-11. File System Modification

## 2.5 Test the Ethernet Under Linux Environment

Copy all the generated files into micro SD card. And make sure some rename needs to be done for file like: zync-zc702.dtb to devicetree.dtb. And then insert this micro SD card into the Starter Kit's card slot.

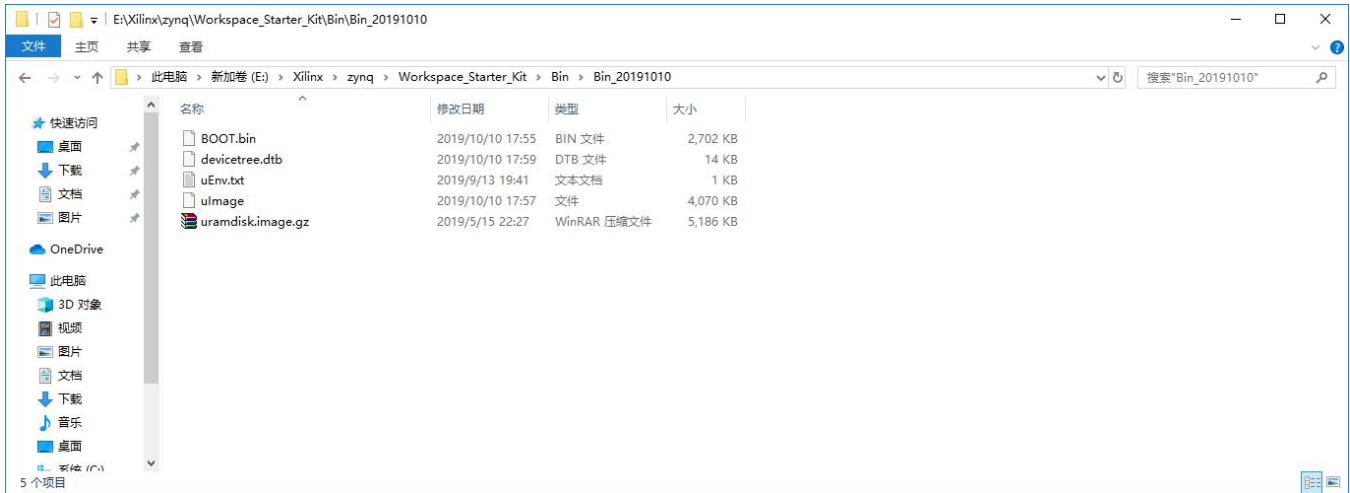


Figure 2-12. Factory Binary Image

Power on the Starter Kit and below log will be displayed on the terminal tool:

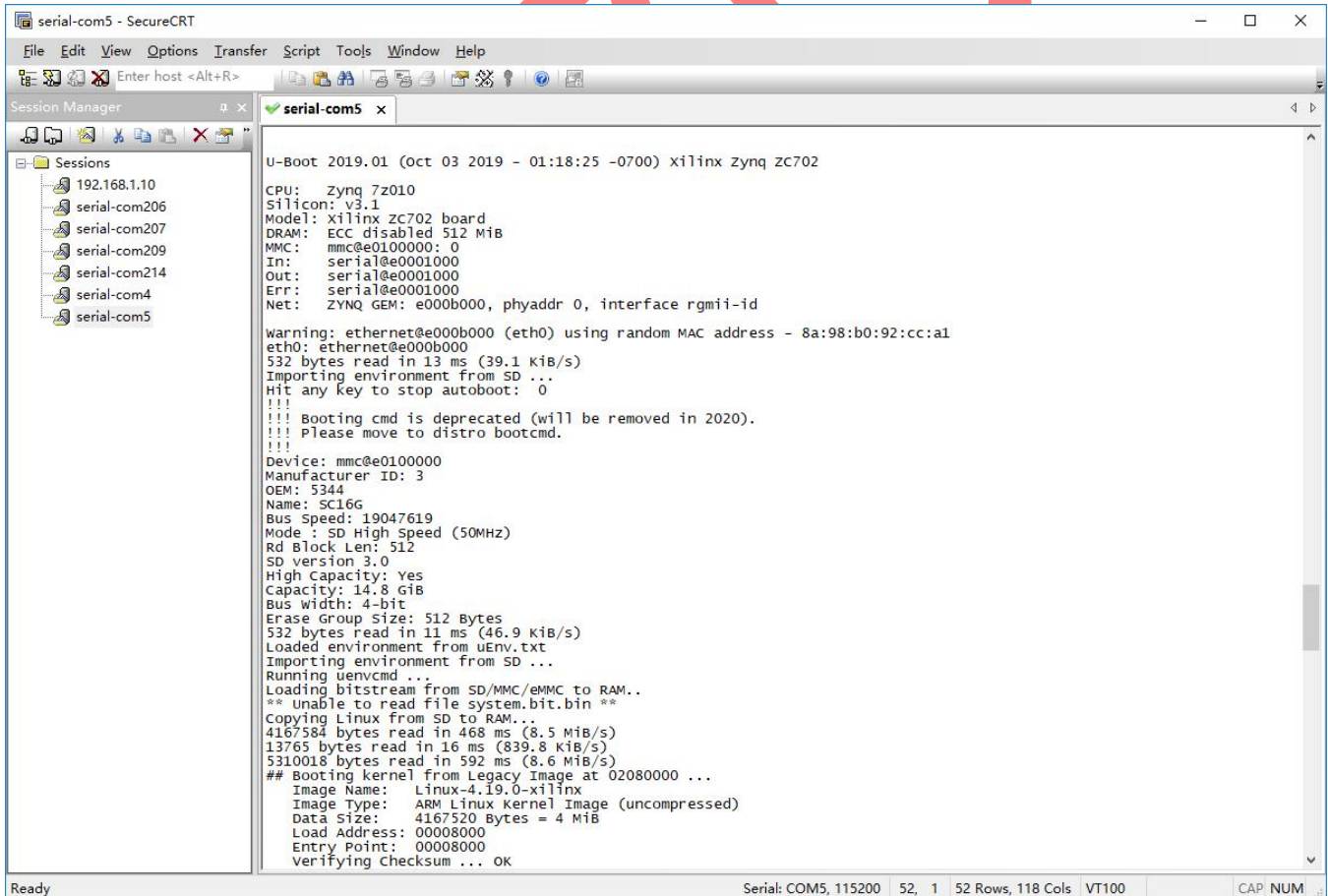
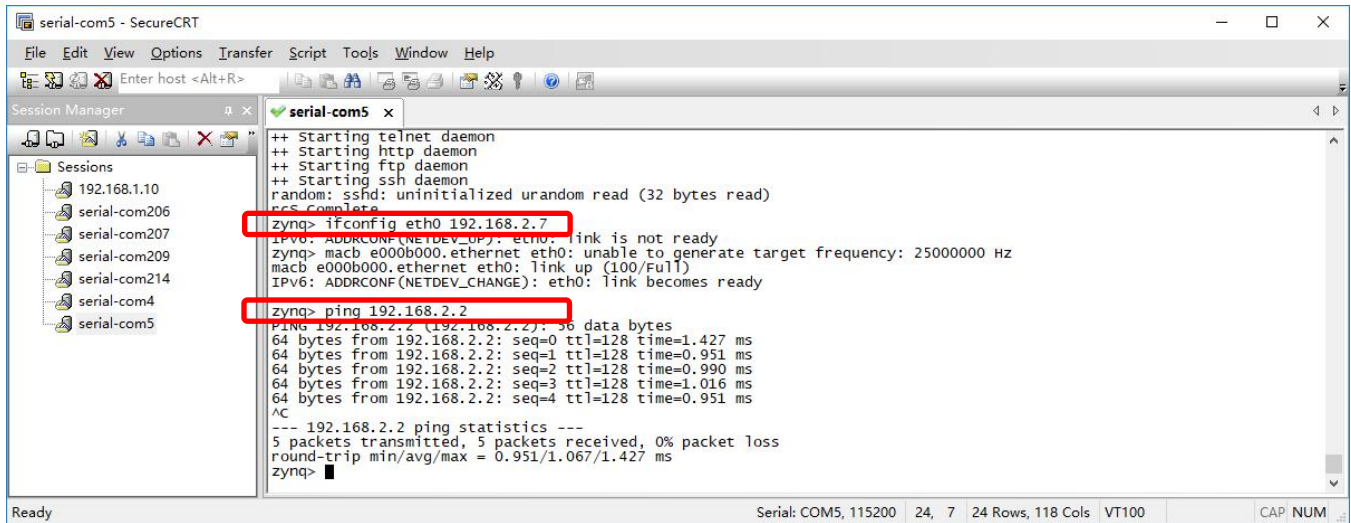


Figure 2-13. Log Info

Make sure the ethernet cable is connected to the router and ZYJZGW Starter Kit. Once the Linux OS successfully runs, users may type below ethernet test related commands to check the status of ethernet interface.



```
serial-com5 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
Session Manager
Sessions
  192.168.1.10
  serial-com206
  serial-com207
  serial-com209
  serial-com214
  serial-com4
  serial-com5
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting ssh daemon
random: sshd: uninitialized urandom read (32 bytes read)
rcs complete
zynq> ifconfig eth0 192.168.2.7
IPV6: ADDRCONF(NETDEV_UP): eth0: link is not ready
zynq> macb e000b000.ethernet eth0: unable to generate target frequency: 25000000 Hz
macb e000b000.ethernet eth0: link up (100/Full)
IPV6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
zynq> ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2): 56 data bytes
64 bytes from 192.168.2.2: seq=0 ttl=128 time=1.427 ms
64 bytes from 192.168.2.2: seq=1 ttl=128 time=0.951 ms
64 bytes from 192.168.2.2: seq=2 ttl=128 time=0.990 ms
64 bytes from 192.168.2.2: seq=3 ttl=128 time=1.016 ms
64 bytes from 192.168.2.2: seq=4 ttl=128 time=0.951 ms
^C
--- 192.168.2.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.951/1.067/1.427 ms
zynq>
```

Figure 2-14. Echo Test under Linux

### 3. Reference

- [1] ug585-Zynq-7000-TRM.pdf
- [2] ds187-XC7Z010-XC7Z020-Data-Sheet.pdf
- [3] ug865-Zynq-7000-Pkg-Pinout.pdf
- [4] MT41K256M16TW-107:P.pdf
- [5] tps563201.pdf
- [6] IP101GA\_2018-11-27.PDF

ZYJZGW



#### 4. Revision

Doc. Rev.	Date	Comments
0.1	01/06/2021	Initial Version.
1.0	21/06/2021	V1.0 Formal Release.