IAC-IMX93-Kit User Manual

Ver.#: 2.1
2024.05

QIYANG TECHNOLOGY Co., Ltd

Version Record

Version No.	Hardware Platform	Description	Date	Revisor
1.0	IAC-IMX93XX-MB-BETA-V1_00	Initial Version	2024-05	Wangwx

Contents

Preface

1.1.Company Profile

Zhejiang Qiyang Intelligent Technology Co., Ltd., established in 2007, which locates in Hangzhou, Zhejiang, PRC. It is a high-end technological enterprise that specializes in exploitation, fabrication, and selling embedded computer mainboards . With 10 years of experiences, Qiyang has established the completed service chain from the design concept to mass production successfully.

The R&D team is organized by 30 more technical engineers. Qiyang focus on providing functional embedded hardware, software tool and customization solutions. It has been applied to Industrial Control, Internet of Things, New Retail, Smart Medical, Electricity Device, Environmental Surveillance, Charging Pile etc.

With the growth of the business, Qiyang has set up an SMT factory in Zhuji, Zhejiang province, which is 5000 ㎡, with a 2xSMT production line. The SMT factory performs the ISO9001 Quality Management System strictly. Relying on the solid production ability, the SMT factory's annual capacity is about a million sets, which totally guarantee the delivery date.

Qiyang has a thorough sales marketing network, professional sales ,and after-sales team to provide full technical support and service. The business has spread over 120 countries and areas, it helps the clients to introduce the products into the market efficiently and successfully.The combination and extension of research and development, production capacity, and market, that provide a solid foundation for Qiyang to provide specialized, globalized embedded hardware and software.

We offer:

l.Software/Hardware Mainboard

Based on the CPU solution from NXP,Rockchip,MTK,Renesas,TI,Atmel,Cirrus Logic etc, Qiyang provides the ARM development kit/system on module/industrial board and periphery products, paired tools and software for the user do further exploitation.

2.Customization Service

Fully taking the advantage of the technical accumulation on the ARM platform and Linux, Android, Ubuntu OS, Qiyang provides the efficient OEM/ODM service to the users.

Sincerely thanks for using Qiyang's product, we will try our best to offer you the technical supports!

1.2.IAC-IMX93-KIT Development Board Use Suggestion:

1. Please read the ***IAC-IMX93-KIT Hardware Manual*** firstly, before using the development kit;
2. Before using, please check the packing list and see whether there is a missing file in the CD;
3. Please understand the basic structure and composition of development board, including the hardware resource allocation etc.;
4. If you need to develop on Linux operation system and flash program into the development board, in addition to this document, we also suggest reading another document ***IAC-IMX93-KIT Linux User Manual*** and ***IAC-IMX93-Kit System Image Flashing Manual;***
5. ***IAC-IMX93-KIT*** development board supports batch order.

I. Preparation

◆ Prepare a set of PC running Windows 10.

◆ A set of PC which installed VM, the virtual machine which has already installed Linux system (Ubuntu or other distributions) or a PC which installed Linux system directly. If there is not such PC, please install the VM refers to the ***Ubuntu Installation Instruction Manual***,

◆ UART :Connect the UART of PC to DEBUG UART (J18) on Development Board through a 3-PIN serial cable (RS232).

(If there is no UART on PC or using laptop, please prepare an USB TO UART (RS232) cable additionally.)

◆ Network: Connect the LAN port of PC to Ethernet port (J5) on development board, or through network switch.

◆ USB: Connect USB port of PC to USB Device (J2) on development board by USB cable.

◆ UART setting: Open terminal communication software in Windows or SecureCRT tool in the provided SDK package; Connect to the serial port which the development board is using, and set the UART with below parameters: Baud rate:115200, Data bit:8-bit, Stop bitt: 1-bit, Parity bit: None, Data flow control: None.

II. System Image Flashing (Firmware Flashing)

IAC-IMX93-Kit development board uses the specialized system image flashing tool:uuu.exe, the specific method ,please refer to ***SMARC-IMMX8MP-Kit System Image Flashing Manual.pdf***

The delivered products have already had the image burned onto them.

III. Functional Description and Testing

IAC-IMX93-Kit has already integrated with test code, while the system boots, there is a relative test code in directory [/usr/test/], the specific test methods, please refer to ***SMARC-IMX8MP-Kit Test Manual.pdf.***

IV. Set up embedded Linux development environment

4.1.Install Cross Compiler

Uboot modification, Kernel driver tailoring, application developing, etc. all of these should be ARM-Linux cross-complied. First of all, you should install the cross-compiler tool chain, the cross-compiler is provided in SDK package/cross-compiler, the user could install and use it directly.

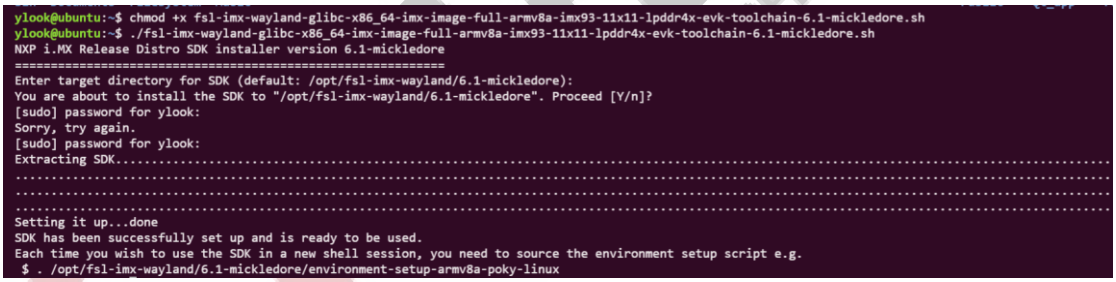
Copy the cross-compiler tool to the installed Ubuntu environment or other Linux distributions, if there is no Ubuntu environment, please refer to ***Virtual Machine Ubuntu Installation Manual***

Cross-compiler installation

$ ./ fsl-imx-wayland-glibc-x86_64-imx-image-full-armv8a-imx93-11x11-lpddr4-evk-toochain-6.1-mickledore.sh

Inputting the installation path, press ‘Enter’, then use the default path.

Input ‘Y’, confirm, as below picture shown, the installation is finished.



4.2.NFS Network File System

NFS (Network File System), different hosts can access files shared by the remote NFS server over the network. By treating the host as an NFS server, we can access the host's files over the network from the development board. The NFS in embedded applications makes the development of applications very convenient. If the Ubuntu environment has been installed, you can skip this section.

The specific usage:

1. Install terminal server

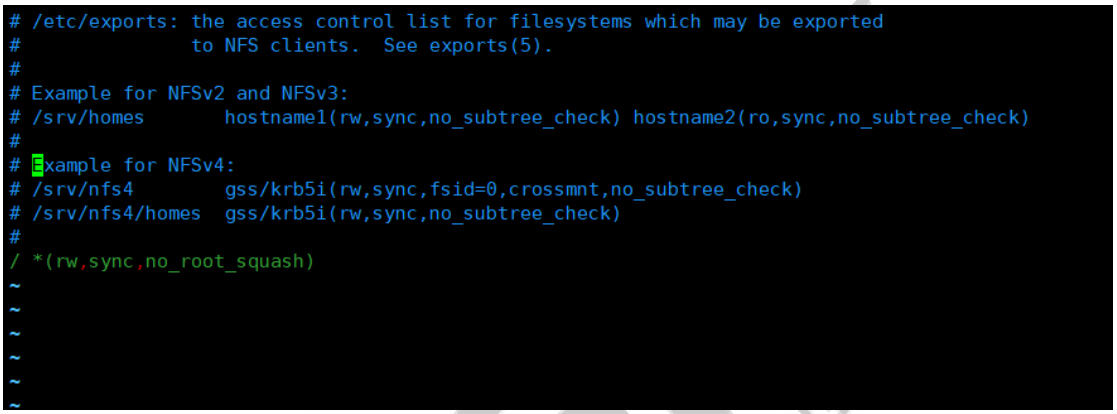
```
$ sudo apt-get install nfs-kernel-server
```

2. Create a shared directory, modify *'/etc/exports'*

```
$ sudo vi /etc/exports
```

Add the below contents at the end of the file:

```
/ *(rw,sync,no_root_squash)
```



Therefore, we have taken the root directory of the PC as the shared directory, meanwhile, it could configure the shared directories by itself.

3. Start NFS server, restart 'nfs-kernel-server':

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

4. Verify NFS locally

```
$ sudo mount local:/ /mnt/
$ ls /mnt/
```

Till now, the contents in '/mnt/' directory could be checked, it means the NFS is configured correctly.

V. Compile Test Code

In the directory of 'SDK package/source code/test code', it has provided diverse kind of source code, it could be recompiled base on your requirements. All the below operations should be executed in the system which has installed the cross-compiler and which the environment variables have added, if the cross-compiler has not been

installed yet, please refer to ***Chapter 4.1. Install Cross-Compiler***

To select the module's source code which needs compiling, here, we take an example of 'buzzer', copy 'SDK package/source code/test coe/buzzer_test' to directory 'ubuntu 20.04'. Open Linux terminal, enter into directory 'buzzer_test', and execute command 'make' , then obtain the executable file 'buzzer_test' in this directory after compiling.

```
$ cd buzzer_test/
$ ls
buzzer_test.c  Makefile
$ source /opt/fsl-imx-wayland/6.1-mickledore/environment-setup-armv8a-poky-linux
$ make
```

Ⅵ. Compile u-boot

The compiled image file and source code has been provided in the compiled SDK package, please use that image file as the bootloader; If the bootloader needs to be recompiled, please execute as below procedures, before executing these commands, please make sure the cross-compiler has been installed correctly, and they have been added into the environment variables, if the cross-compiler has not been installed yet, please refer to ***Chapter 4.1.***

Source Code:

uboot.tar.gz

Extract the zipped package **uboot.tar.gz**.

```
$tar xvf uboot.tar.gz
```

6.1.Compile uboot

```
$ cd uboot/uboot/
$ ./build.sh
```

After compiling, the below files will be executed:

flash.bin

lash.bin (It is the file which is burnt into the board)

Ⅶ. Compile kernel

The SDK package has the configured kernel source code **kernel.tar.gz.**

The compressed package, copy it to the directory of Ubuntu20.04, the user could configure the kernel as their needs. Please make sure the current cross-compiler has been installed into the PC before starting kernel compiling.

```
$ tar zxvf kernel.tar.gz
$ cd kernel/
$ ./ build.sh
```

After compiling correctly, the below files are generated:

arch/arm64/boot/Image

arch/arm64/boot/dts/freescale/imx93-qiyang-n034a.dtb

arch/arm64/boot/dts/freescale/imx93-qiyang-gn070.dtb

arch/arm64/boot/dts/freescale/imx93-qiyang-hj070na.dtb

arch/arm64/boot/dts/freescale/imx93-qiyang-mipi-n91.dtb

Ⅷ.Develop Application Program

8.1. Hello World

You can develop the application program freely, here, we take an example of 'Hello Word'. At first, please write the code for 'Hello World', the codes are as below:

```
#include <stdio.h>

int main(void)

{

printf("Hello World ! \n");

return 0;

}
```

Save to 'hello.c' then we use the below command to do cross compilation.

8.2. Cross compilation

To enable the program running on the development board normally, it requires the cross compiler to compile the application. Put the above files to directory Ubuntu 20.04, we can use the below command to compile:

```
$ source /opt/fsl-imx-wayland/6.1-mickledore/environment-setup-armv8a-poky-linux
$ $CC -o hello hello.c
```

After compilation, there is an executable binary file 'hello' which generated in current directory.

8.3. Run application program

Thus, we have obtained the executable program which could run on the development board, here, we introduce how to run the application program on the development board, there are two methods: Mounting NFS server and adding it to the file system.

We can use the NFS server which has been configured well, mounting the PC's NFS

server on the development board; then we can operate the files on the development board, for instance, copying files, program running, etc. By using this method ,it will be easy for debugging, the specific method is as below:

1.Ensure the development board is connected with the PC by LAN cable well, and start NFS server in PC.

2.Set the IP of the development board and IP of the PC at the same network segment, for example：

PC IP：192.168.1.75

Development Board IP: 192.168.1.203

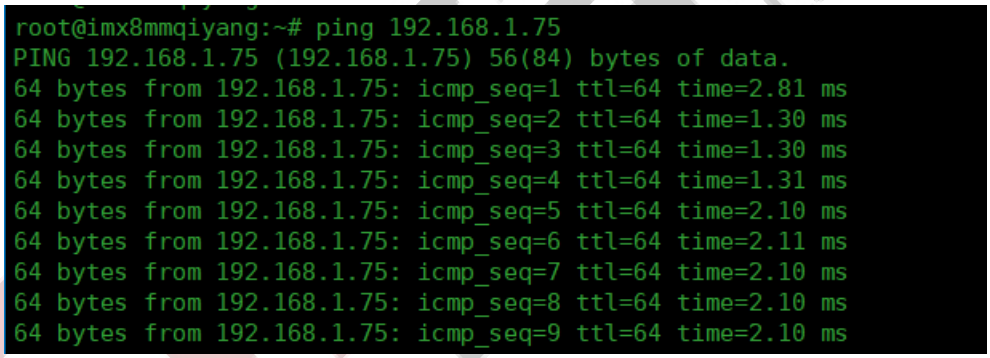
Network Marsk: 255.255.255.0

Broadcast IP: 192.168.1.255

3.Test network connection

To ping PC on development board, run the below commands in hype terminal:

ping 192.168.1.75



To ping development board in PC, if the PC could ping the mainboard, it means the network connects normally.

4.To mount PC's NFS

Input the below commands in hyper terminal:

mount -o nolock 192.168.1.75:/ /mnt/

cd /mnt/

There is a root directory of the PC which is in directory '/mnt' after mounting correctly, then please enter into the directory which the application program locates. By this

way, it will be easy for program debugging till the program debugged correctly, then add the application program to the file system, flash it to emmc. Thus, it can avoid the constant flashing to emmc.

8.4. Add program application to file system

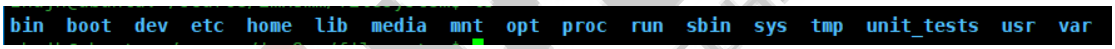
When the application program finished debugging, please add it to the file system.Now, you can flash the application program and file system to emmc together. Next, we shall introduce how to add the application program to the root file system, copy the system source code of the root file system from the SDK package to the Ubuntu 20.04, and extract.

Create a new folder ‘rootfs’, extract it, the unzipped files are in this folder.

```
$ mkdir filesystem
```

Copy the zipped file ’rootfs.tar.zst’ of file system in SDK package to the directory of this file.

```
$ tar -- zstd -xvf rootfs.tar.zst    // Operate to the extracted filesystem directory
$ ls
```



Extract the files as below picture shown:

Compress the files in this directory, delete the old ‘rootfs.tar.zst’ file before compressing.

```
$ rm -rf rootfs.tar.zst
$ tar --zstd -cvf rootfs.tar.zst *
```

After compressing, there is a *rootfs.tar.zst* file generated in current directory.

Ⅸ.Conclusion

FAQ:

1.Over higher Ubuntu version may cause the system can't detect the cross-compiler issue, please use a low-version Ubuntu, the author suggests using Ubuntu 20.04.

2.Please check the file's integrity while loading files from Windows to Ubuntu, it's necessary to check the file's permission before installing, editing, etc.

3.Ubuntu is a multi-user operation system, please make sure all operations are going to be processed at one same user; The root permission in root directory will be required while installing software and writing files in root directory.

4.The above file paths which appeared in this manual are the author's environment path, the users must use their own environment paths to avoid mistakes.

Ⅹ.FAQ

```
  YACC    scripts/dtc/dtc-parser.tab.[ch]
scripts/dtc/yamltree.c:9:10: fatal error: yaml.h: No such file or directory
 #include <yaml.h>
          ^~~~~~~~
compilation terminated.
scripts/Makefile.host:112: recipe for target 'scripts/dtc/yamltree.o' failed
make[1]: *** [scripts/dtc/yamltree.o] Error 1
make[1]: *** Waiting for unfinished jobs....
Makefile:1378: recipe for target 'scripts_dtc' failed
make: *** [scripts_dtc] Error 2
```

If there are errors during compiling source code, it requires to install a software.

```
$ sudo apt-get install libyaml-dev libpython2.7-dev libssl-dev
```

Zhejiang Qiyang Intelligent Technology Co., Ltd

Tel: 86-571-87858811 / 87858822

Fax: 86-571-89935912

Technical Support： 0571-87858811 ext.805

E-MAIL： supports@qiyangtech.com

Website： **http://www.qiytech.com** or **http://www.qiyangtech.com**

ADD: 3rd Floor, Building A, WSCG Building, NO.6 Xiyuan 8th Road, Sandun Town, Xihu District, Hangzhou City, Zhejiang China

Postal Code： 310013