

1.) Prepare the x86_64 Debian Host

`mkdir /home/youruser/assets` this will be the target for the final image

Install all required packages for QEMU

`sudo apt install qemu-efi-aarch64 qemu-system-arm virt-manager`

Download the arm64 mini.iso from Debian

<https://d-i.debian.org/daily-images/arm64/daily/netboot/>

2.) Setup Virtual Machine in QEMU

open Virtual Machine Manager

select "Local install media (ISO image or CDROM)"

in "Architecture options" select Architecture: **aarch64** and Machine Type: **virt**

next select the just downloaded **mini.iso**

next choose the operating system **Debian 10**

next set Memory to **1024** and CPUs to **4**

next create a disk image and set size to **4 GiB**

finally click "Finish" and click "Yes" to make Virtual Network active

3.) Install Debian for arm64 in your Virtual Machine

click into the black area of the VMs Window to capture Mouse and Keyboard

hit Enter to start text based Debian Installer

create **root** password and **youruser** with password as they will be on the final image

partition manually the disk image as follows

Partition 1: Size **100M**, Name **efi**, Use as **EFI System Partition**, Bootable flag **on**

Partition 2: Size **200M**, Name **boot**, Use as **Ext 2 file system**, Mount point **/boot**

Bootable flag **off**

Partition 3: Size **max**, Use as **Ext 4 journaling file system**, Mount point **/**

Bootable flag **off**

confirm that you don't want to create Swap Space by clicking **<NO>**

in "Software selection" select only **SSH server** and **standard system utilities**

and finish the installation, once finished reboot into the newly installed system

4.) DTB file handling

`mkdir /boot/dtbs`

`nano /etc/kernel/postinst.d/copy-dtbs`

`#!/bin/sh`

`set -e
version="$1"`

`echo Copying current dtb files to /boot/dtbs....
cp -a /usr/lib/linux-image-`${version}`. /boot/dtbs/`

`chmod +x /etc/kernel/postinst.d/copy-dtbs`

`/etc/kernel/postinst.d/copy-dtbs `uname -r``

5.) Bootloader configuration

```
mkdir /boot/extlinux
```

```
nano /boot/extlinux/extlinux.conf
```

```
TIMEOUT 2
```

```
DEFAULT debian
```

```
LABEL debian
```

```
MENU LABEL Debian
```

```
KERNEL /vmlinuz
```

```
INITRD /initrd.img
```

```
DEVICETREEDIR /dtbs
```

```
APPEND console=ttyS2,1500000 console=tty1 root=LABEL=root rw rootwait
```

```
apt purge grub-efi-arm64
```

```
apt autoremove
```

```
apt autoclean
```

```
shutdown -h now
```

6.) Creating tar archives of our VM

```
sudo modprobe nbd max_part=8
```

```
sudo qemu-nbd --connect=/dev/nbd0 /var/lib/libvirt/images/debian10-aarch64.qcow2
```

```
sudo mount /dev/nbd0p2 /mnt
```

```
cd /mnt
```

```
sudo tar cvzp /home/youruser/assets/debian-aarch64-bootfs.tar.gz .
```

```
cd ..
```

```
sudo umount /mnt
```

```
sudo mount /dev/nbd0p3 /mnt
```

```
cd /mnt
```

```
sudo tar cvzp /home/youruser/assets/debian-aarch64-rootfs.tar.gz .
```

```
cd ..
```

```
sudo umount /mnt
```

```
sudo qemu-nbd -d /dev/nbd0
```

7.) Install Cross Compiler for building U-Boot on our x86_64 Debian Host

```
sudo apt install device-tree-compiler build-essential libssl-dev python3-dev bison
```

```
sudo apt install flex libssl-dev swig gcc-aarch64-linux-gnu gcc-arm-none-eabi
```

```
(sudo apt install gcc make bc git)
```

8.) Build U-Boot on our x86_64 Debian Host

```
cd /home/youruser/assets
```

```
git clone https://github.com/ARM-software/arm-trusted-firmware
```

```
cd arm-trusted-firmware
```

```
git tag
```

remember last stable (v2.7)

```
git checkout v2.7
```

```
make CROSS_COMPILE=aarch64-linux-gnu- PLAT=rk3328 bl31
```

```
cd ..
```

```
git clone git://git.denx.de/u-boot.git
cd u-boot
git tag
git checkout v2022.04
ln -s /home/youruser/assets/arm-trusted-firmware/build/rk3328/release/bl31/bl31.elf bl31.elf
make CROSS_COMPILE=aarch64-linux-gnu- BL31=bl31.elf rock64-rk3328_defconfig
make -j4 CROSS_COMPILE=aarch64-linux-gnu- BL31=bl31.elf all u-boot.itb

cp -r /home/youruser/assets/u-boot/idbloader.img /home/youruser/assets/
cp -r /home/youruser/assets/u-boot/u-boot.itb /home/youruser/assets/
```

9.) Flashing Debian to our Pine64 Rock64 SBC (16GB eMMC module (30310400 sectors))

```
sudo fdisk /dev/sdX
```

type **o** this will clear out any partitions on the drive
type **p** to list partitions, there should be no partitions left
type **n**, then **p** for primary, **1** for the first partition on the drive, **32768** for the first sector, and **647168** for the last sector, then type **a**
then type **n**, then **p** for primary, **2** for the second partition on the drive, **647169** for the first sector, and **28213246** for the last sector, then type **n**, then **p** for primary, **3** for the third partition on the drive, **28213247** for the first sector, and **30310399** for the last sector, then type **t**, and **3** for the third partition, and **82** for the Hex Code, then write the partition table and exit by typing **w**

```
cd /home/youruser/assets
mkdir boot
mkdir root
```

this is in your home directory ! → /home/youruser/assets/boot
this is in your home directory ! → /home/youruser/assets/root

```
sudo mkfs.ext2 -m0 -L boot /dev/sdX1
sudo mount /dev/sdX1 /home/youruser/assets/boot
cd /home/youruser/assets/boot
sudo tar xzvpf /home/youruser/assets/debian-aarch64-bootfs.tar.gz .
sync
cd ..
sudo umount /home/youruser/assets/boot
```

```
sudo mkfs.ext4 -L root /dev/sdX2
sudo mount /dev/sdX2 /home/youruser/assets/root
cd /home/youruser/assets/root
sudo tar xzvpf /home/youruser/assets/debian-aarch64-rootfs.tar.gz .
sync
cd ..
```

```
sudo nano /home/youruser/assets/root/etc/fstab
```

amend as below

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/mmcblk1p1 /boot ext2 defaults 0 2
/dev/mmcblk1p2 / ext4 errors=remount-ro 0 1
/dev/mmcblk1p3 swap swap defaults 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

```
sudo nano /home/youruser/assets/root/etc/network/interfaces
```

change interface to eth0

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
```

```
source /etc/network/interfaces.d/*
```

```
# The loopback network interface
auto lo
iface lo inet loopback
```

```
# The primary network interface
auto eth0
allow-hotplug eth0
iface eth0 inet dhcp
```

```
sudo umount /home/youruser/assets/root
```

```
sudo mkswap /dev/sdX3
```

```
cd /home/youruser/assets/
sudo dd if=idbloader.img of=/dev/sdX seek=64 conv=notrunc
sudo dd if=u-boot.itb of=/dev/sdX seek=16384 conv=notrunc
```

- 10.) Install the eMMC-Module onto your Pine64 Rock64 SBC, connecting HDMI, Mouse and Keyboard, power it up and log-in as **root**.

```
ip a
```

check that network is working

- 11.) Check the MAC address, may need spoofing if address is **06:ca:fa:7c:8c:f8** (1GB Board) or **3e:6a:eb:10:6a:9b** (4GB Board)

```
ip link show eth0
```

If you MAC address is **06:ca:fa:7c:8c:f8** or **3e:6a:eb:10:6a:9b** then do steps below, or the network will not work !

```
ip link set dev eth0 down
ip link set dev eth0 address 06:ca:fa:XX:XX:XX
ip link set dev eth0 up
```

change the last 3 bits to your liking,
DO NOT change the first 3 bits (reserved for Manufacturer)

```
reboot
```

once board is up, check with **ip link show eth0** for success

- 12.) Log-in as **youruser** and perform system update and enable filesystem check at boot

```
sudo apt update
sudo apt upgrade
sudo apt full-upgrade
sudo apt autoremove
sudo apt autoclean
```

```
sudo tune2fs -c 1 /dev/mmcblk1p1
sudo tune2fs -c 1 /dev/mmcblk1p2
```

- 13.) Remove unnecessary packages, which are no longer required

```
sudo apt purge qemu-guest-agent
sudo rm /root/etc/systemd/system/getty.target.wants/serial-getty@ttyAMA0.service
sudo rm -rf /boot/grub
sudo rm -rf /boot/efi
```

- 14.) Add Firmware for Rockchip CDN DisplayPort Controller

```
sudo nano /etc/apt/sources.list
```

```
# deb http://deb.debian.org/debian bullseye main
```

```
deb http://deb.debian.org/debian bullseye main contrib non-free
```

```
deb-src http://deb.debian.org/debian bullseye main contrib non-free
```

```
deb http://deb.debian.org/debian-security/ bullseye-security main contrib non-free
```

```
deb-src http://deb.debian.org/debian-security/ bullseye-security main contrib non-free
```

```
deb http://deb.debian.org/debian bullseye-updates main contrib non-free
```

```
deb-src http://deb.debian.org/debian bullseye-updates main contrib non-free
```

```
sudo apt update
```

perform system update

```
sudo apt install firmware-misc-nonfree
```

contains -> rockchip/dptx.bin

- 15.) Hide kernel messages during boot

```
nano /etc/sysctl.conf
```

```
# Uncomment the following to stop low-level messages on console
```

```
kernel.printk = 3 4 1 3
```

```
sudo reboot
```

Done, enjoy your setup.