

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 **100M**, 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
```

```
PROMPT 1
```

```
DEFAULT debian
```

```
LABEL debian
```

```
MENU LABEL Debian
```

```
KERNEL /vmlinuz
```

```
INITRD /initrd.img
```

```
DEVICETREEDIR /dtbs
```

```
APPEND console=tty1 console=ttyS2,115200n8 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-clone.qcow2
```

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

```
cd /mnt
```

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

```
cd
```

```
sudo umount /mnt
```

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

```
cd /mnt
```

```
sudo tar cfvzp /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

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

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

```
git tag
```

remember last stable (v2.3)

```
git checkout v2.3
```

```
make CROSS_COMPILE=aarch64-linux-gnu- PLAT=sun50i_h6 bl31
```

```
cd ..
```

```
git clone git://git.denx.de/u-boot.git
cd u-boot
git tag
git checkout v2020.07
ln -s /home/youruser/arm-trusted-firmware/build/sun50i_h6/release/bl31/bl31.elf bl31.elf
make CROSS_COMPILE=aarch64-linux-gnu- BL31=bl31.elf pine_h64_defconfig
make -j4 CROSS_COMPILE=aarch64-linux-gnu- BL31=bl31.elf all u-boot.itb
```

```
cp -r /home/youruser/u-boot/ u-boot-sunxi-with-spl.bin /home/youruser/assets/
```

9.) Flashing Debian to our Pine64 H64B SBC

```
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, **2048** for the first sector, and **647167** for the last sector, then type **a**
then type **n**, then **p** for primary, **2** for the second partition on the drive, **647168** for the first sector, and **28211199** for the last sector, then type **n**, then **p** for primary, **3** for the third partition on the drive, **28211200** for the first sector, and **30308351** 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 cp /home/youruser/assets/sun50i-h6-pine-h64-model-b.dtb /home/youruser/assets/boot/dtbs/allwinner/sun50i-h6-pine-h64-model-b.dtb
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
```

```
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/mmcblk1p2 /boot ext2 defaults 0 2
/dev/mmcblk1p3 / ext4 errors=remount-ro 0 1
/dev/mmcblk1p4 swap swap defaults 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

```
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  
allow-hotplug eth0  
iface eth0 inet dhcp
```

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

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

```
cd home/youruser/assets/  
sudo dd if=u-boot-sunxi-with-spl.bin of=/dev/sdX bs=1024 seek=8
```

- 10.) Installing the eMMC-Module onto your Pine64 Rock64 SBC, connecting HDMI, Mouse and Keyboard and power it up.

```
ip a
```

check that network is working

```
apt update  
apt upgrade  
apt dist-upgrade  
apt autoremove  
apt autoclean
```

perform system update

Done, enjoy your setup.

