Slackware ARM project web site | Forum | Slackware ARM development documentation | Slackware ARM installation guides

Installing Slackware on the Pinebook Pro

Document name	inst_sa64_cur_rk3399_pinebookpro
Document purpose	Document the installmaation of Slackware Linux onto the Hardware Model: Pinebook Pro
Version	1.04, Mar 2024
Author	Stuart Winter <mozes@slackware>
Collaborators	Brenton Earl <el0226@slackware> (R&D for the initial integration work)

Help / Support

Please post questions to the Slackware ARM forum.

Video Tutorial

This tutorial is also available in video form.

The video tutorial demonstrates the original installation approach where the Slackware installation media was separate. A single Slackware Installer image is provided that contains all of the media, so it's easier than shown in the video.

Installation Lifecycle

The Installation consists of a number of stages:

1. Acquiring all required hardware
2. Downloading and verifying the Slackware assets
3. Writing the Initialisation Bootware to the Micro SD card
4. Setup of the Pinebook Pro hardware
5. Initialising the Pinebook Pro with the Bootware
6. Writing the Slackware Installer to the Micro SD card
7. Booting the Slackware Installer
8. Installing Slackware
9. Completing the installation
10. Booting the Slackware OS
11. OS configuration

Requirements

Hardware

Item	Specification	Required?	Notes
Pinebook Pro	Single specification	Required	The Pinebook Pro laptop
Micro SD Card	16GB **minimum capacity**, Class 10 (fast speed), good quality make	Required	Initially used to boot the Slackware Installer, and subsequently transformed into Slackware's /boot partition
USB Multi-Card Reader	Must accept Micro SD cards	Required	Used to write the Bootware on your host Linux computer. This isn't required if your host computer has a Micro SD card reader.
NVME Storage Module	Minimum useful size (contains OS and user data): 30GB	Required	Kingston A2000 SSD 250G model has proved reliable. Note: The 500G model had reliability issues on the Pinebook Pro yet works in an x86 without issue.
Pinebook Pro NVMe SSD Interface Adapter	The specific model for the Pinebook Pro	Required	This houses the NVME Storage Module
Serial console adapter	3.5“ audio jack wired version	Optional	This is useful for debugging during development, but its use precludes the ability to enable sound on the laptop. Most users will not use the Serial adapter.
Pinebook Pro USB-C Docking Deck	The specific model for the Pinebook Pro	Optional, but Recommended	The Pinebook Pro has no onboard Ethernet, and this docking station can be used to provide a range of additional peripherals.

Hardware Assembly Tools

Item	Notes
Phillips/cross-head screw driver	The head size needs to fit the screws in the Pinebook Pro case. This is used to remove the base in order to set up the hardware.

Note about the eMMC

From the factory, your PinebookPro contains an eMMC storage module. During the development of Slackware AArch64, it was found that the life span of these storage modules is short which makes them inappropriate for housing an Operating System. Whilst it's possible to use eMMC with Slackware, this documented installation process does not provide a supported path and the eMMC should be removed.

Computing / Network Environment

Item	Specification	Notes
Host Computer: an Internet-connected computer running an existing Linux distribution	The Host Computer needs approximately **5GB free storage** to download the required software assets. **You must be able to obtain *root* access to this Host computer**.	This will be used to download the Slackware distribution from the Internet and write the Slackware Installer to the Micro SD card.

Hardware Setup

In this section we'll prepare the physical aspects of the Pinebook Pro to receive Slackware Linux. There are seven distinct parts to this phase:

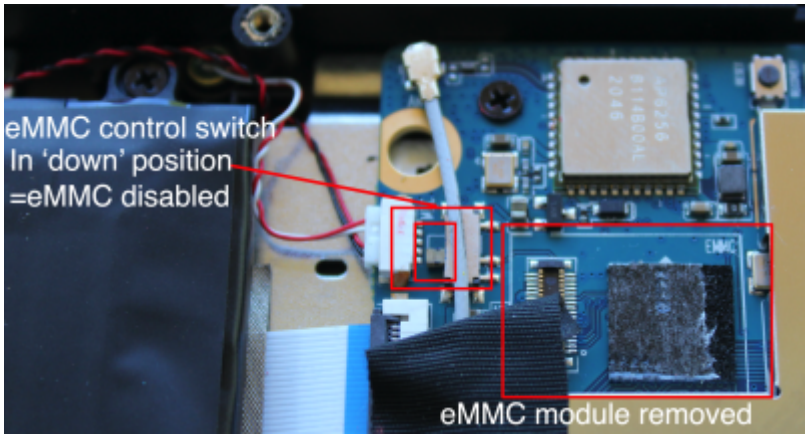
1. Unscrew the case
2. Disable eMMC (and remove storage module)
3. Enable Sound (disabling Serial console)
4. Securely attach NVME to NVME adapter
5. Connect NVME adapter to the main board
6. Securely attach NVME adapter within the Pinebook Pro's case
7. Screw case back together

Remove the base cover from the Pinebook Pro

Turn the Pinebook Pro over, and remove all screws

Turn the Pinebook Pro back over to reveal the motherboard and interior

Disable booting from eMMC and remove the eMMC storage module



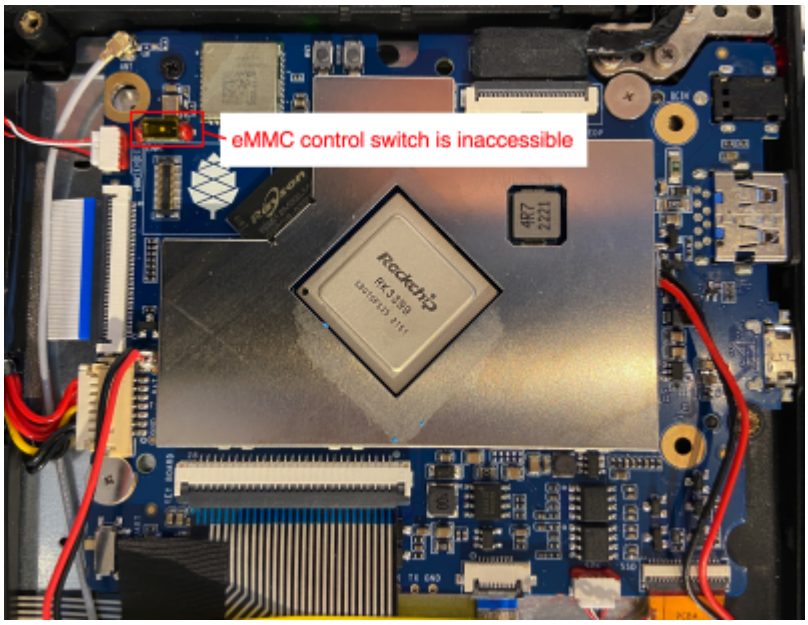
From the factory, the Pinebook Pro will be provided with an eMMC storage module configured to be enabled. During the development of Slackware AArch64, it was found that the life span of these storage modules is short which makes them inappropriate for housing an Operating System. Whilst it's possible to use eMMC with Slackware, this documented installation process does not provide a supported path and the eMMC must be disabled.

Move the eMMC control switch into the **down position.**

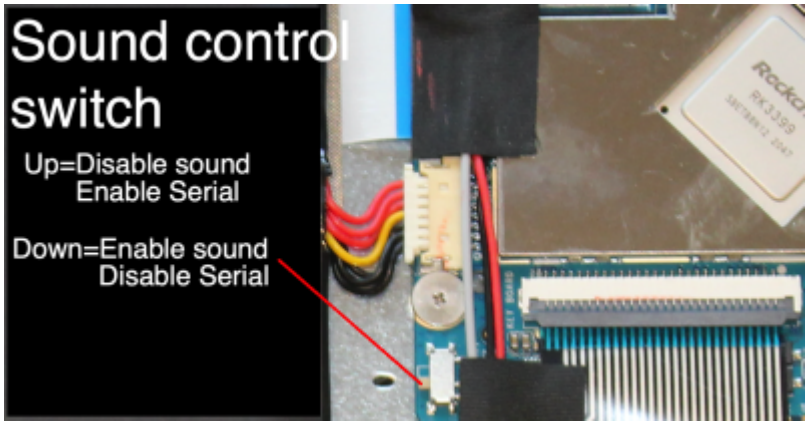
It's recommended to remove the eMMC storage module (as can be seen in the image) as it won't be used for Slackware and may become loose over time. It's also recommended to retain the eMMC with the original Linux distribution should you require it at some point in the future. If you prefer to keep the eMMC storage here, it **must** be *disabled*.

To remove it, gently lift it up and out with your fingers.

Newer versions of the Pinebook Pro do not permit disabling the eMMC via the switch, as depicted in the image below. You must remove the eMMC module this case.



Ensure Sound is enabled



This control switch dictates the usage of the 3.5” audio jack port. However, if you are debugging or developing, you may wish to use the Serial adapter - in which case, set this switch into the 'up' position.

Since this is a laptop, most people will want to enable sound: ensure that the switch is in the **'down' position** (as shown in the image)

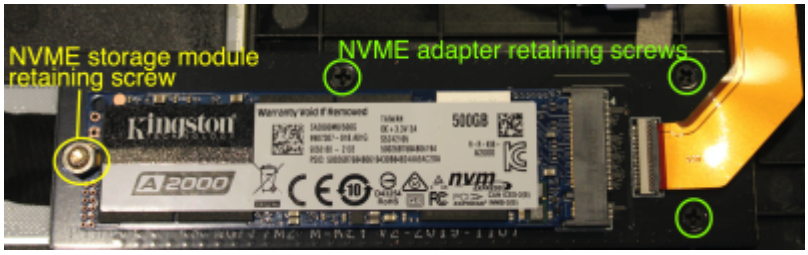
Assemble the NVME adapter and storage module

Connect the ribbon cable to the NVME adapter.

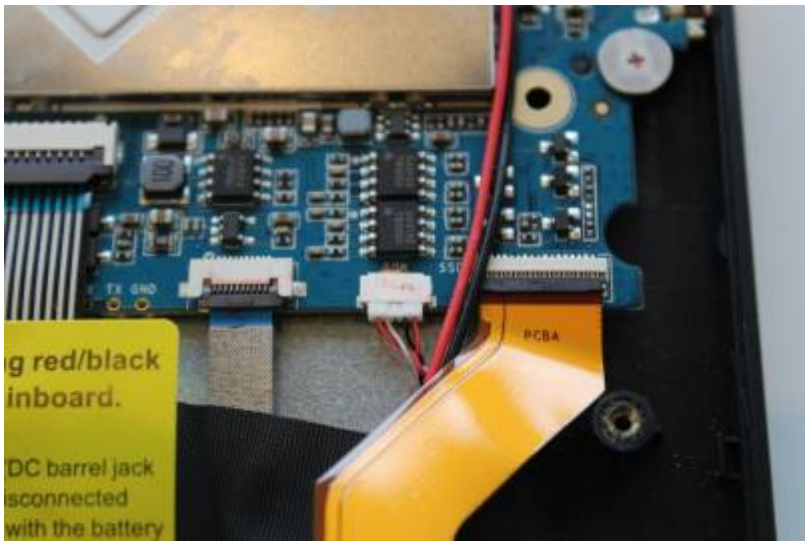
Insert the NVME storage module into the adapter interface:

Secure the NVME storage module to the adapter with a nut and bolt (ensure that the nut and bolt do not exceed the height of the storage module, otherwise the case won't close!)

Secure the NVME adapter within the Pinebook Pro's case using the three screws:

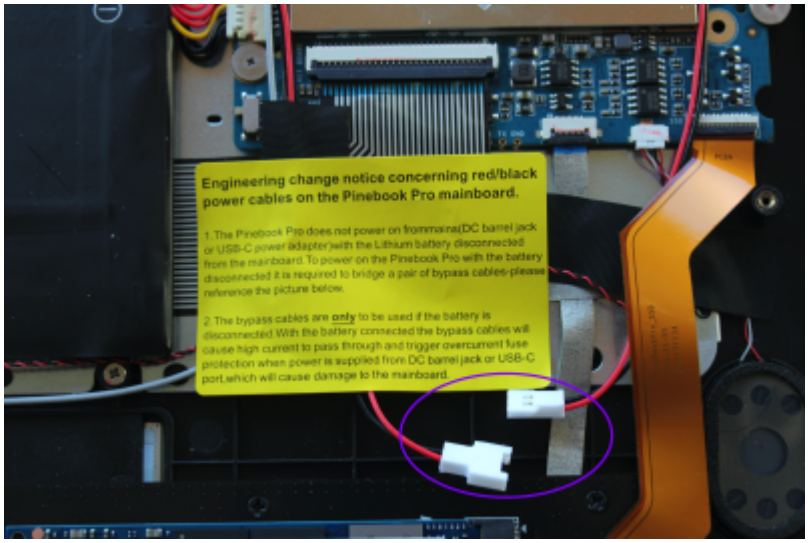


Connect the NVME adapter ribbon to the Pinebook Pro's motherboard:



Disconnect the battery bypass cable

The battery bypass cable must be disconnected whilst the battery is attached.



Completing the Hardware Setup

The hardware set up is now complete and should look like this:

Screw the Pinebook Pro's case back together

The hardware setup is complete.

Software and Network Environment Setup

In this section, we'll prepare the Linux Host Computer to receive and download the Slackware assets required for the installation.

Downloading the Slackware Linux AArch64 Distribution and Installation Assets

The '$' prefixes in the commands indicates the shell prompt - it's not to be typed/copied

Open a shell on the Linux Host Computer.

Prepare a directory to hold and serve the Slackware Distribution

We'll download the Slackware Linux distribution into a directory named 'slackware'.

```
$ cd ## this returns to the root of your home directory
$ mkdir slackware
$ cd slackware
```

Determine where you are within the Host Computer's Filesystem

```
$ pwd
/home/mozes/slackware
```

Note the directory location returned - you'll need this later

Installing the Slackware ARM GPG key

The Slackware ARM GPG key will be used to verify the Bootware and Slackware Installation images.

```
$ curl -sSL https://www.slackware.com/infra/keys/arm/GPG-KEY | gpg --import
-
```

Set the version of Slackware AArch64 to download

At the time of writing, the only version available is 'current'.

```
$ SLKVER=current
```

Set the Hardware Model

```
$ HWM=rk3399_generic
```

Set the Internet media distribution server

If you are using a mirror server rather than the master Slackware ARM server, set it here. The format is: <hostname>::<rsync module name>

```
$ SLKSRV=ftp.arm.slackware.com::slackwarearm
```

Download the Bootware

Note the period/full stop after the rsync commands - this instructs rsync to download to the current directory (it's not punctuation!)

The U-Boot Boot Loader that will be installed onto the SPI flash:

```
$ rsync -PavL $SLKSRV/platform/aarch64/bootware/recovery/rk3399/flash-spi-
pinebookpro.img.xz .
$ rsync -PavL $SLKSRV/platform/aarch64/bootware/recovery/rk3399/flash-spi-
```

```
pinebookpro.img.xz.asc .
```

The Bootware (recovery/initialisation) images are approximately 400KBytes in size.

Download the Slackware Linux Installer

```
$ rsync -PavL $SLKSRV/platform/aarch64/bootware/installer-
aio/slackwareaarch64-${SLKVER}/${HWM}.img.xz.asc slkaio.img.xz.asc
$ rsync -PavL $SLKSRV/platform/aarch64/bootware/installer-
aio/slackwareaarch64-${SLKVER}/${HWM}.img.xz slkaio.img.xz
```

The Slackware Installer images are approximately 5 GBytes in size.

Verify the Downloads

```
$ gpg --verify-files *.asc
```

The output should be similar to this:

```
gpg: assuming signed data in `flash-spi-pinebookpro.img.xz'
gpg: Signature made Fri 27 Jan 2023 08:53:24 AM -00 using RSA key ID
1623FC33
gpg: Good signature from "Slackware ARM (Slackware ARM Linux Project)
<mozes@slackware.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 36D3 7609 2F12 9B6B 3D59  A517 F7AB B869 1623 FC33
gpg: assuming signed data in `slkaio.img.xz'
gpg: Signature made Thu 19 Jan 2023 08:34:27 PM -00 using RSA key ID
1623FC33
gpg: Good signature from "Slackware ARM (Slackware ARM Linux Project)
<mozes@slackware.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 36D3 7609 2F12 9B6B 3D59  A517 F7AB B869 1623 FC33
```

If you see 'BAD signature' you should re-download as it may have become corrupt. If this doesn't help, drop a note to the Slackware ARM forum

Write the Initialisation Bootware to the SD Card

Slackware stores the U-Boot Boot Loader firmware within the SPI flash of the Hardware Models that use the RK3399 SoC (including the Pinebook Pro, RockPro64 et al).

In this step, we'll write the Boot Loader firmware to the same Micro SD card that will later be used to contain the Slackware Installer, and subsequently the Slackware OS' /boot partition. If you have multiple Micro SD cards available, you may prefer to use separate SD cards; but this document assumes the availability of a single Micro SD card.

Elevate yourself to root

On your Host Computer, obtain root:

The # prefix indicates that you're using the **root** user - it's not to be typed in!

```
$ su -   ## Note the hyphen - it's required
```

Check what block devices are present

Prior to inserting the Micro SD Card into the USB adapter, we need to see what's already present within the OS so that we can easily locate our Micro SD card:

```
# lsblk -d
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 465.8G  0 disk
```

As you can see, this Host Computer there is a single storage device - *sda*.

Now insert the Micro SD card into your USB Card Reader and connect the adapter to a free USB port on the Host Computer.

Run lsblk again:

```
# lsblk -d
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda      8:0    0 465.8G  0 disk
sdc      8:32   1    58G  0 disk
sdd      8:48   1     0B  0 disk
```

As you can see, *sdc* is 58GBytes in size. This is the Micro SD card (in this example, it's labeled as '64GB' on the exterior of Micro SD card).

If your Micro SD card has existing partitions, you will not see them surfaced in this list - use *lsblk -b* to view them.

You'll also observe the presence of *sdd* - often the USB adapter itself obtains a block device. You can ignore this as it's 0Bytes.

Write the Bootware Initialisation Image to the Micro SD Card

Still as the **root** user, we'll switch to the home directory into which the the Slackware assets have been downloaded (see earlier steps):

```
# cd ~mozes/slackware/  ## Replace with the correct directory name
```

Write the Bootware Initialisation Image to the device identified as our Micro SD card. You'll then exit the root shell, returning to your usual standard user environment:

All data on this Micro SD Card will be erased! Ensure you have inserted the correct card!

```
# dd if=/dev/zero of=/dev/XXX count=10 bs=1M    ## Replace /dev/XXX with the
correct block device (presented above by the lsblk tool) on your Host
Computer
# xzcat flash-spi-pinebookpro.img.xz > /dev/XXX  ## Replace /dev/XXX with
the correct block device (presented above by the lsblk tool) on your Host
Computer
# sync
```

Remove the Micro SD card from the Host Computer

Now that we've written the Bootware Initialisation Image to the Micro SD card, you need to remove it from the Host Computer.

Perform a sanity check

Perform a sanity check to ensure that the block device list no longer displays the devices we were using. This type of check is essential to ensure that you don't accidentally write to the wrong block storage device.

```
# lsblk -d
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 465.8G  0 disk
```

The block devices assigned to the Micro SD card are no longer present, so we are safe to proceed.

Keep the root shell open as we'll be using it again in the subsequent steps.

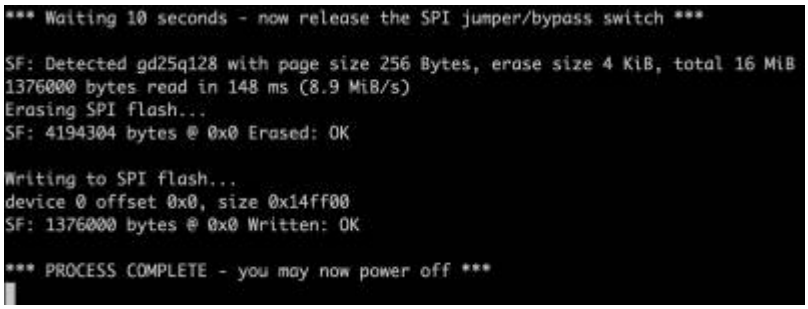
Installing the Boot Loader to SPI flash

You MUST remove any USB devices (peripherals, storage) before proceeding. This is because when USB peripherals are connected, it has been observed that the flashing process tends to hang.

You need to perform this one-time step to flash a Slackware version of the U-Boot Boot Loader to the SPI flash of the Pinebook Pro.

If you reinstall Slackware you do **not** need to repeat this step, as long as the Slackware version of U-Boot remains within the SPI flash.

1. Connect the power to the Pinebook Pro
2. Disconnect any peripherals (including the Pinebook Pro docking station) from the Pinebook Pro.
3. Insert the Micro SD card into the Pinebook Pro's Micro SD slot (right hand side, below the 3.5“ audio jack port)
4. Open the lid
5. Power on the Pinebook Pro: hold down the Power button (top right of keyboard) for 3 seconds



After a few seconds, you should see a message appearing on the screen - instructing you that the process will begin in 10 seconds' time, and to release the SPI jumper. In most cases you can simply wait until the process begins.

Once the process completes, *hold the power button for approximately **8 seconds***. The PinebookPro will power off.

Any issues with installing to SPI flash

The Slackware Micro SD card Recovery/Initialization image typically boots and operates as described, even if a distribution has already installed a Boot Loader onto the SPI flash. However, its success hinges on the configuration of the current SPI flash-embedded Boot Loader. If this Boot Loader isn't set up to boot from the SD card, you might need to circumvent the SPI flash to initiate the SD card boot.

The Pinebook Pro has an immutable first stage Boot Loader which, when the SPI flash doesn't contain an executable Boot Loader, will attempt to boot from the SD card.

Bypass SPI flash - method 1

There is a 'Recovery' button on the Pinebook Pro's main board (requires dissassembling) that will 'mask out' the SPI flash temporarily, thus preventing any Boot Loader stored on the SPI flash from executed.

1. Insert the Slackware Recovery/Initialisation Micro SD card into the Pinebook Pro's Micro SD slot
2. Press and hold 'recovery' button.
3. Quickly press 'reset' button.
4. Release 'recovery' button after about 3 seconds

If this doesn't work, you may need to try a few times or try alternative methods (see below).

Bypass SPI flash - method 2

This thread on the Slackware ARM LinuxQuestions forum contains instructions to bypass the SPI flash to enable booting the SD Card.

This is also described in one of the Slackware ARM Vlog episodes.

Bypass SPI flash - method 3

If you encounter difficulties launching the Slackware Boot Loader flashing tool and find that the pre-installed Linux distribution keeps booting, an option is to clear the SPI flash using that Linux distribution.

Obtain 'root' on the Linux distribution:

```
$ dd if=/dev/zero of=/dev/mtd0 bs=1M count=3
```

Now insert the Slackware Boot Loader flashing SD card and reboot.

Bypass SPI flash - more information

The Pine64 Wiki provides more information about working with the SPI flash and is the source of the instructions for *method 1* above.

Write the Slackware Installer image onto the MicroSD card

Now that the Boot Loader has been installed to the PinebookPro's SPI flash, we will install the Slackware Installer image onto the same MicroSD card.

- Remove the MicroSD card from the PinebookPro
- Insert the MicroSD card back into the Host Computer (as in the earlier section)

Run lsblk again:

Now that we have re-inserted the Micro SD Card into the Linux Host Computer, we need to check which block device it has been assigned.

In this guide you'll see that it's the same as previously (which is to be expected, although you *must* check!)

```
# lsblk -d
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda      8:0    0 465.8G  0 disk
sdc      8:32   1    58G  0 disk
sdd      8:48   1     0B  0 disk
```

Write the Slackware Installer to the Micro SD card

Enter the directory into which the Slackware assets were downloaded previously:

The # prefix indicates that you're using the **root** user - it's not to be typed in!

Still as root on your Linux Host Computer:

```
# cd ~mozes/slackware  ## Replace with the correct directory
# xzcat slkaio.img.xz | dd status=progress bs=4M iflag=fullblock of=/dev/XXX
## Substitute /dev/XXX with the correct block device
# sync
```

Remove the MicroSD card from the Host Computer

You may now disconnect the USB adapter from the Host Computer and remove the MicroSD card.

Run lsblk again to confirm that the block device assigned to the Micro SD card is no longer present:

```
# lsblk -d
```

```
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 465.8G  0 disk
```

Logout from the root user

```
# logout
```

You no longer require the Host Computer.

Installing Slackware

To proceed, you must have:

- Assembled the PinebookPro as documented above
- Inserted the Micro SD card containing the Slackware Installer into the PinebookPro's MicroSD card slot

Encrypted storage

If you'd like to encrypt your storage, check the Disk Encryption Guide.

Begin installation

Disconnect any USB storage devices that aren't required for the OS installation

Power on the PinebookPro

Press the Power Button for aproximately two seconds

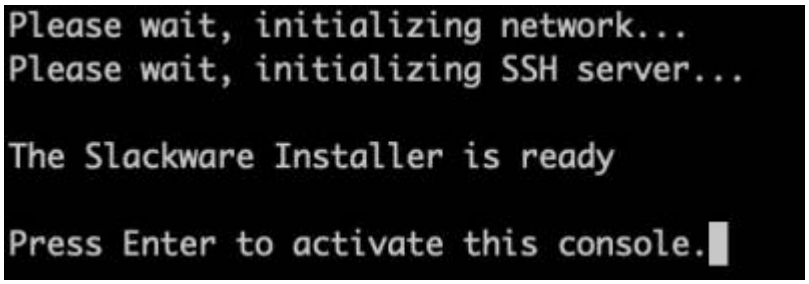
After a few seconds, the you will see the following on screen:

It takes several seconds to load and boot the installer, and it may take several seconds more for any further output to appear on the screen.

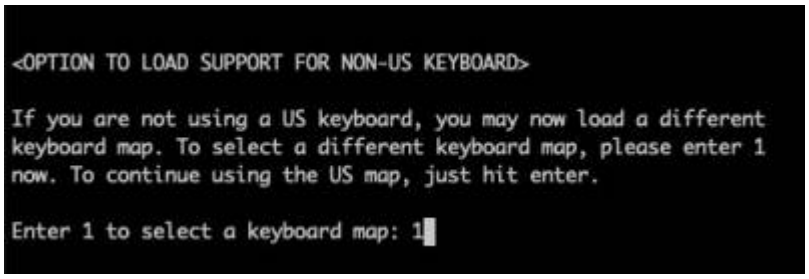
The process of obtaining an IP address via DHCP can also delay the ability to interact with the Installer. If you don't have wired Ethernet available, this will add approximately 10 seconds delay to the Installer being available.

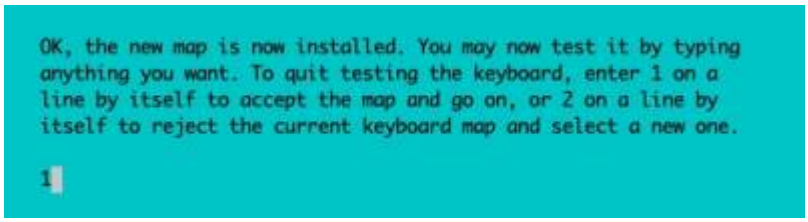
Once networking has been setup (or timed out), you will be presented with a prompt.

Press **ENTER**



Set the keymap





Set the date/time

If you have Internet access available, you may wish to sync the date via NTP. If not, you can skip this.

```
# ntpdate rolex.ripe.net
# hwclock -w
```

Setup disk partitions

For this installation a basic partitioning scheme will be created.

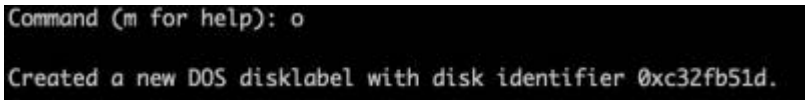
Partition number	Device name	Size	Purpose
1	/dev/nvme0n1p1	4GB	Swap
2	/dev/nvme0n2p2	Rest of storage	OS root ('/') partition

/boot will reside on the Micro SD card and is automatically configured by the Slackware Installer

Open fdisk against the /dev/nvme0n1 block device (which will be your primary storage, and in these instructions is the NVME you screwed inside the laptop in the previous section).

```
fdisk /dev/nvme0n1
```

Clear an existing partition table: Press 'o' to clear the partition table



Create the Swap partition:

```
root@slackware:~# fdisk /dev/nvme0n1

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xcb7c8d24.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
```

Type 'n' for new partition:

Type 'p' for primary partition type:

Press ENTER for the 'First sector'

Type '+4G' for the 'Last Sector'/size:

Change the partition type to 'Swap'. Type 't' then hex code '82':

```
Command (m for help): t
Selected partition 1
Hex code or alias (type L to list all): 82
Changed type of partition 'Linux' to 'Linux swap'.
```

Create the partition for the root filesystem ('/'):

```
Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (2-4, default 2):
First sector (8390656-976773167, default 8390656):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (8390656-976773167, default
976773167):

Created a new partition 2 of type 'Linux' and of size 461.8 GiB.
```

Type 'n' for new partition. Press ENTER to accept the defaults - this will create partition 2 as the maximum size available.

Type 'a' to mark the root partition (number 2) as bootable Type '2' to select partition 2.

```
Command (m for help): a
Partition number (1,2, default 2): 2

The bootable flag on partition 2 is enabled now.
```

Type 'p' to print to view the partition table.

```
Command (m for help): p
Disk /dev/nvme0n1: 465.76 GiB, 500107862016 bytes, 976773168 sectors
Disk model: KINGSTON SA2000M8500G
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xcb7c8d24

Device         Boot   Start       End   Sectors   Size Id Type
/dev/nvme0n1p1         2048   8390655   8388608     4G 82 Linux swap
/dev/nvme0n1p2 *    8390656 976773167 968382512 461.8G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

root@slackware:~#
```

Type 'w' to write the partition table:

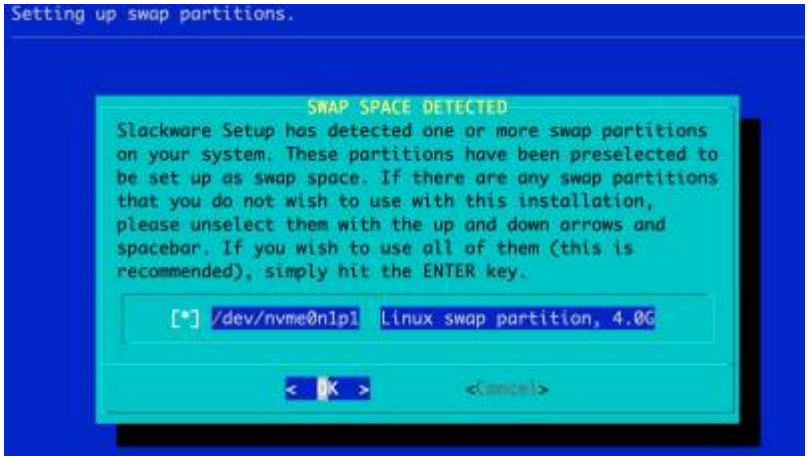
```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

root@slackware:~#
```

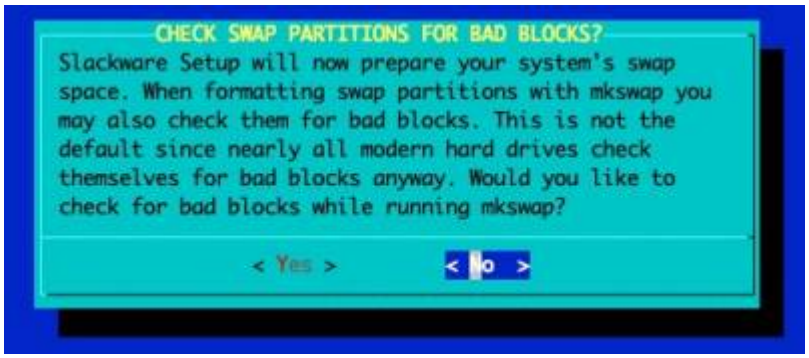
fdisk will now exit.

Load the Setup menu

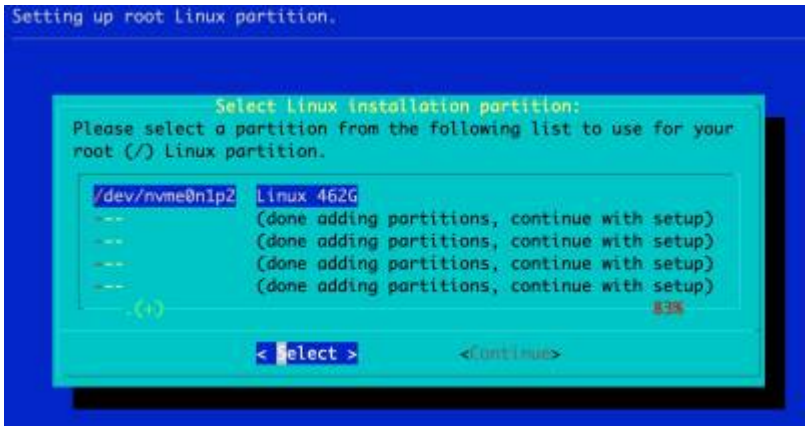
```
root@slackware:~# setup
```

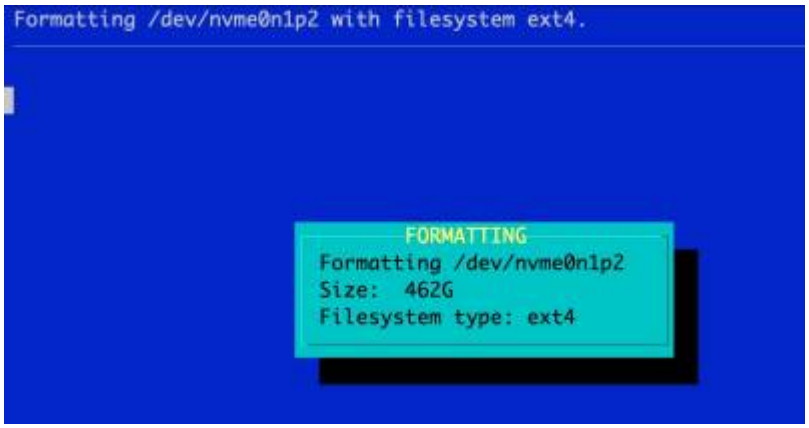
Setup Swap partition





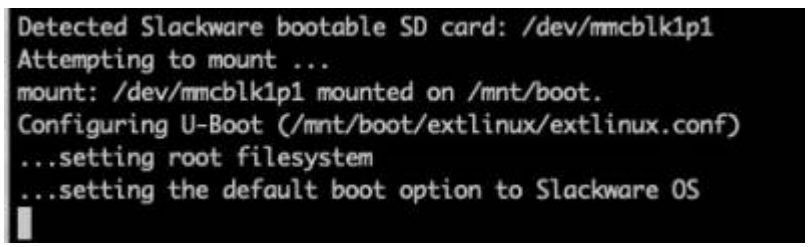
Select and format the partition for the OS' root file system

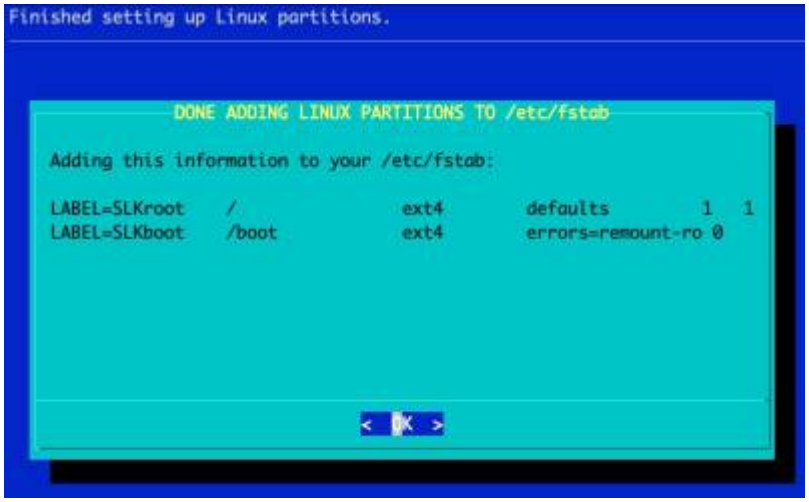




Boot Loader Configuration

The Installer will configure the Boot Loader and the OS' /etc/fstab automatically:





On the ARM platform, the Swap and root file systems are addressed by labels (*see above:* '*LABEL=*') where as on x86/64 it's addressed by a direct reference to the block device (e.g. /dev/sda).

The swap partitions are labeled '*SLKswap<x>*', the root file system '*SLKroot*', and the /boot partition '*SLKboot*'.

The rationale behind this divergence is that on x86 the root file system is typically on a storage bus (SCSI, SATA, ATA), where the physical configuration (which port the storage is connected to) of the storage rarely changes. This can be the case on ARM, but it's generally to a lesser extent and the root file system may be connected to a hot-plug bus such as USB. This lends itself to the risk of device re-ordering across boot cycles (e.g. /dev/sda becomes /dev/sdb), causing boot failure.

Please be aware that the Slackware Installer *only* labels the swap and root file system. Therefore you are advised to manually label the file systems and modify the OS /etc/fstab accordingly. If you have only a single storage device and don't plan on adding more, you can use the settings that the Slackware Installer configures.

Select Source Media

Press ENTER to say 'Yes'.

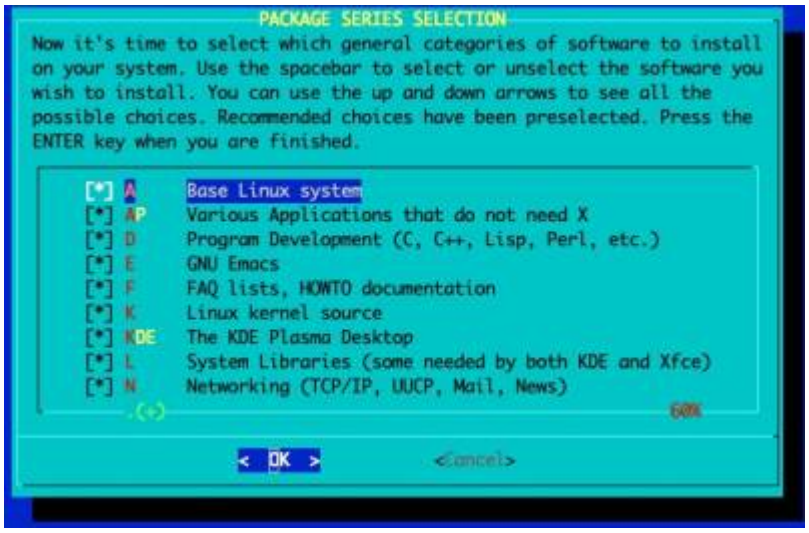
If you would like to install from an alternate media source, pick 'No' and you will be presented with options to install over NFS, USB and HTTP amongst others.

Ordinarily you should always say 'Yes' unless you've been directed to do otherwise.

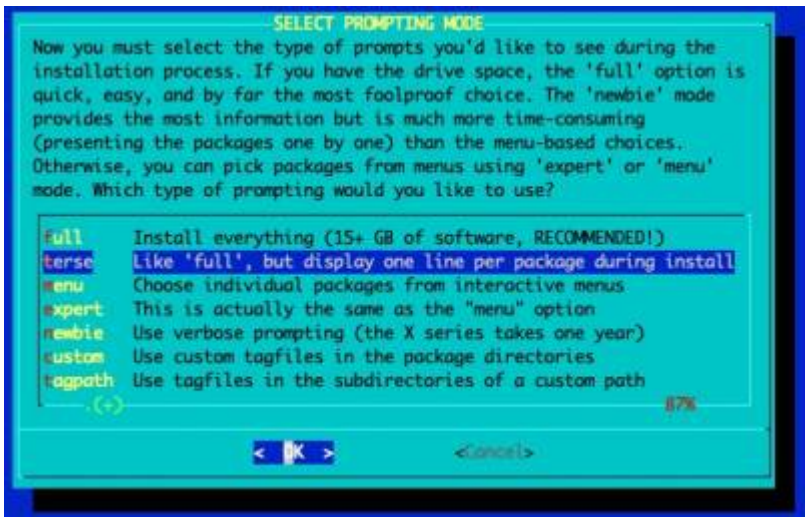
Package Series Selection

You can now choose the package sets to install. The recommendation is to install everything. A full Slackware installation will occupy approximately 15GB.

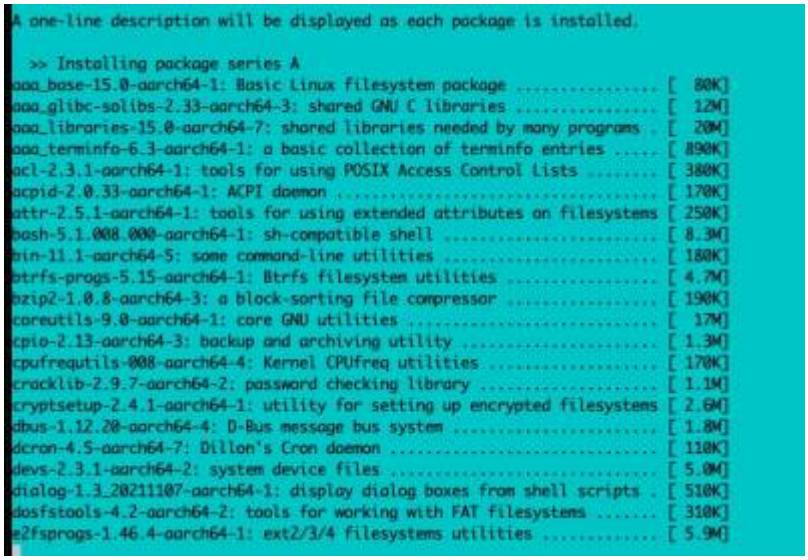
If you do not plan to use the graphical window manager such as XFCE or KDE, you should de-select it.



Pick the 'terse' option:

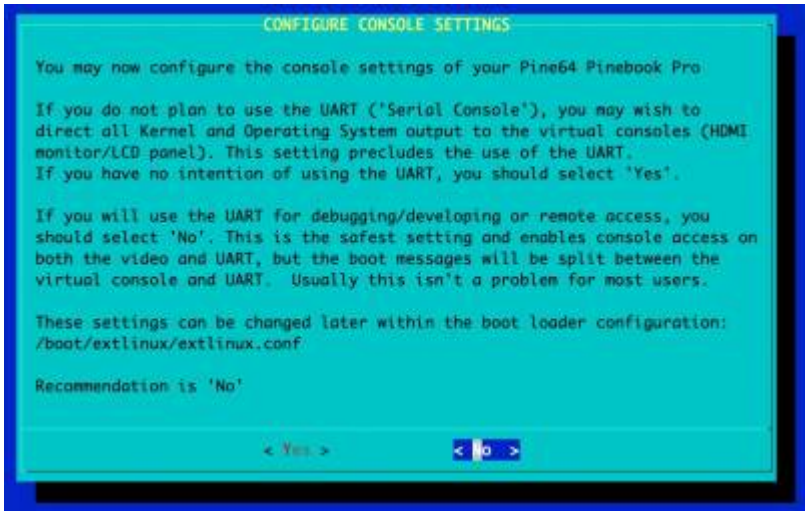


The packages will begin installing:



Configure the Console Settings

This setting can be reset to the default by editing /boot/extlinux/extlinux.conf and removing the 'console=' setting once the OS has booted



If you plan on using the UART/'Serial' console, you should select 'No' here.

If you plan on **exclusively** the Pinebook Pro's LCD panel, you should **pick '*Yes*'**.

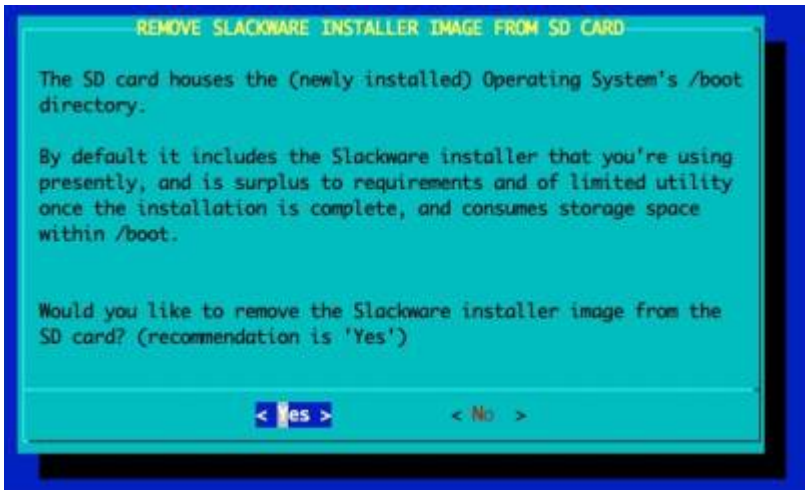
For most users, selecting 'No' will be fine and enables connecting the serial console later.

This setting can be adjusted later by editing `/boot/extlinux/extlinux.conf` and removing the `console=` Kernel cmdline parameter

Remove the Slackware Installer from the SD card

The Micro SD card is transformed from being the Slackware Installer into the Slackware OS's /boot partition. At this stage, if the installation has worked for you (at certain points in the Slackware installer you are past the point of no return) you can delete the Installer. However, if something has gone wrong you can reset the PinebookPro and reboot the installer without having to re-deploy the Slackware Installer image from your Linux Host Computer.

Generally you should say 'Yes' here.

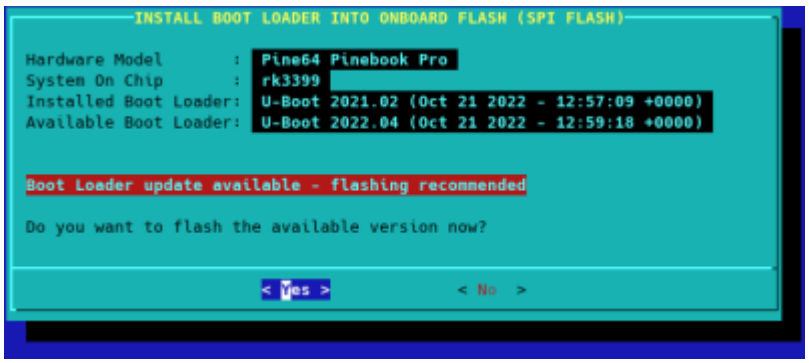


You may be tempted to retain the Slackware Installer, but note that the Installer contains Linux Kernel modules for the Kernel that the Installer was originally shipped with. This means that as soon as you upgrade the Slackware Kernel package, the Installer will fail to boot. The option to retain the Installer is present purely because on a number of occasions, this author only realised that the installation was incorrectly performed upon completion, and needed to reinstall. Retaining the Installer avoids the requirement to re-deploy the image to the SD card.

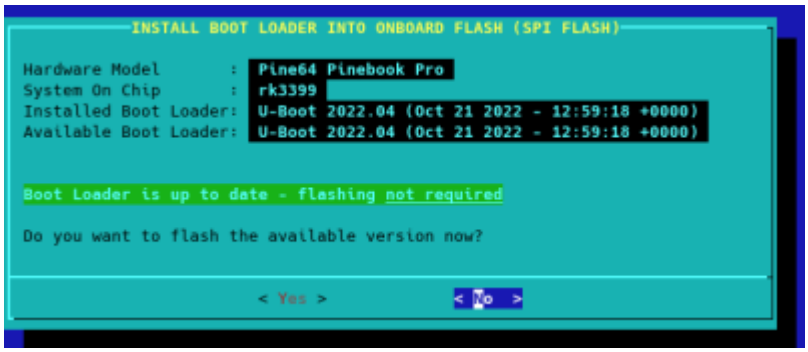
Install the Boot Loader to SPI flash

If this is the first time you have installed Slackware, you must flash the Boot Loader. The initial Boot Loader flash performed earlier in these instructions typically contains an older version of the Boot Loader, where as the version packaged within the Installer contains the most recent version.

When an upgrade is required, in most cases you'll see a screen like this which provides information about the currently installed Boot Loader and the newer version available:

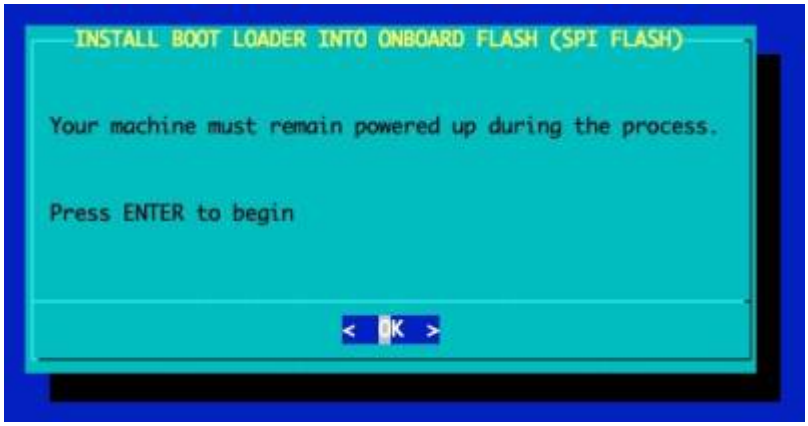


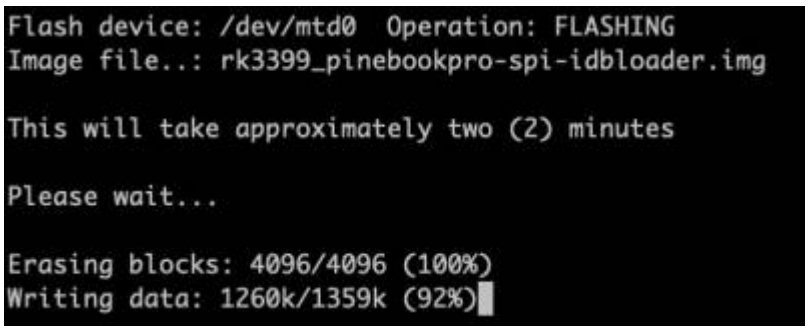
If the installed Boot Loader matches the currently available version, you will be advised that flashing is unnecessary. However, you can re-flash it if you wish.

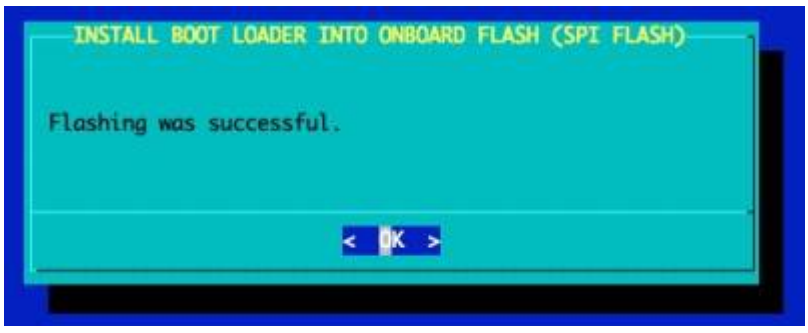


If you proceed with flashing, the work flow looks like this and takes approximately two minutes to complete.

You must ensure the system has sufficient power/is plugged in. If the machine powers off before flashing completes, you would need to re-flash using the bootable SD card as described earlier within this guide.

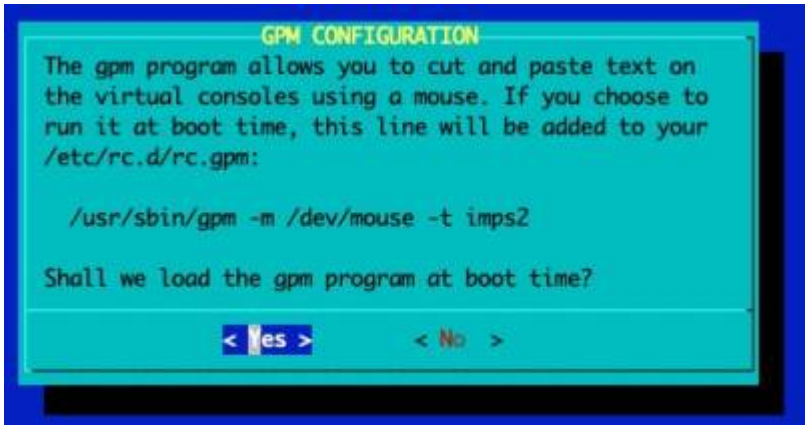


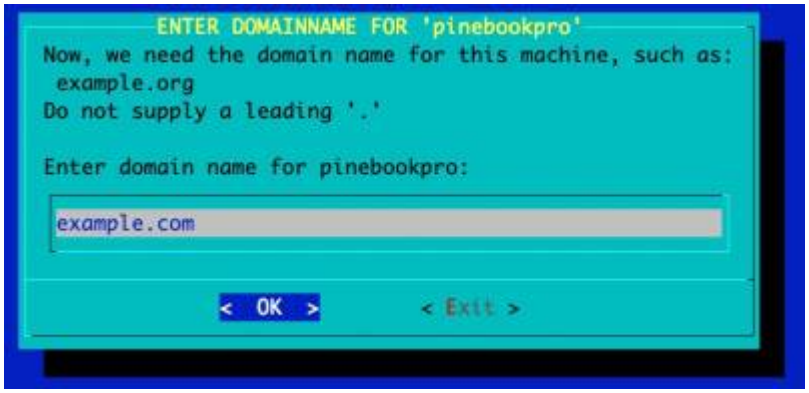


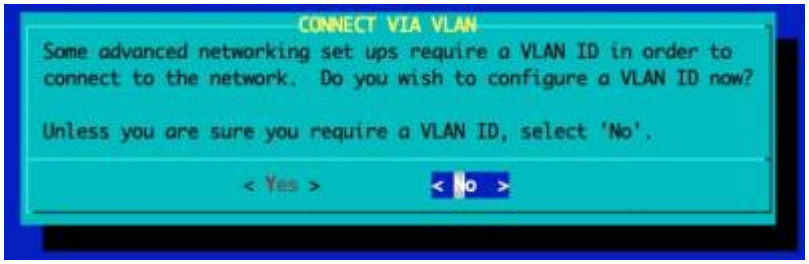


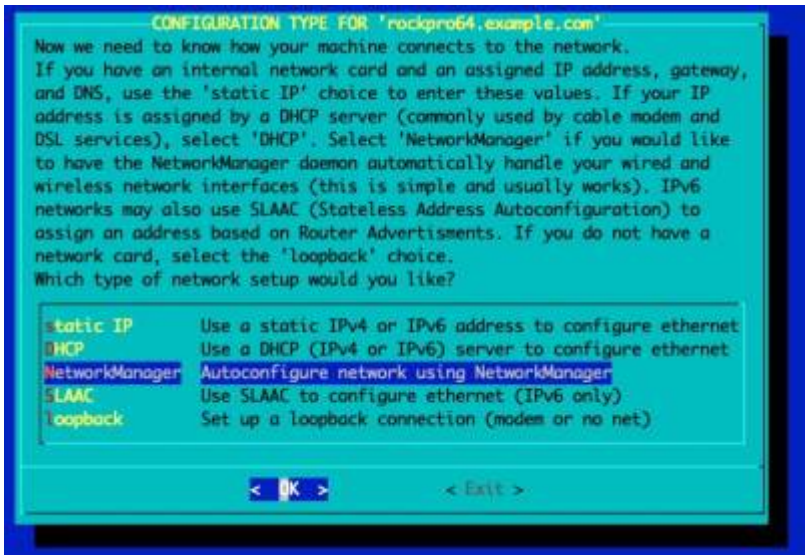
Post Installation Configuration

The Slackware Installer will walk you through the standard Slackware setup. The on-screen instructions will suffice.





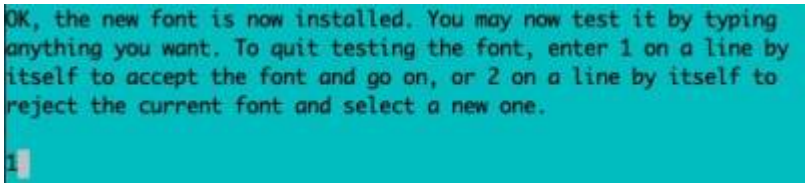




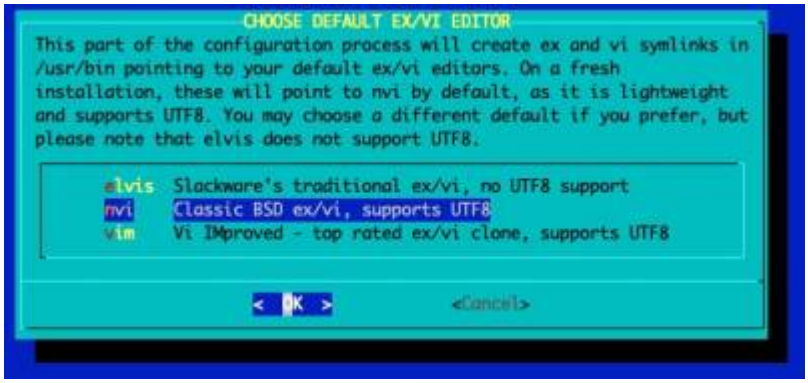
Select a Console Font

It's recommended for the RockPro64 and Pinebook Pro that a larger console font is configured.

The recommended font is 'ter-732b.psf'. This is the font used within the Installer.



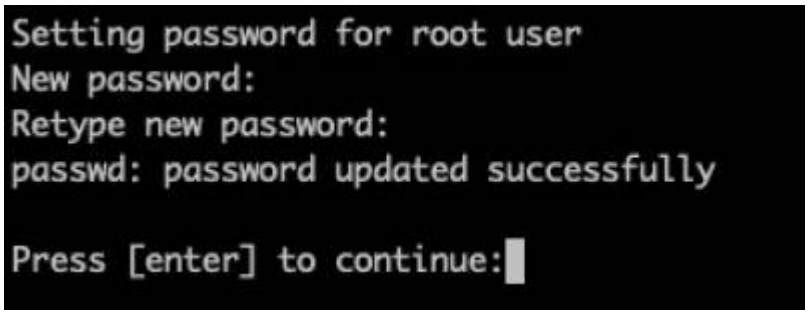
Continue Post Installation Configuration



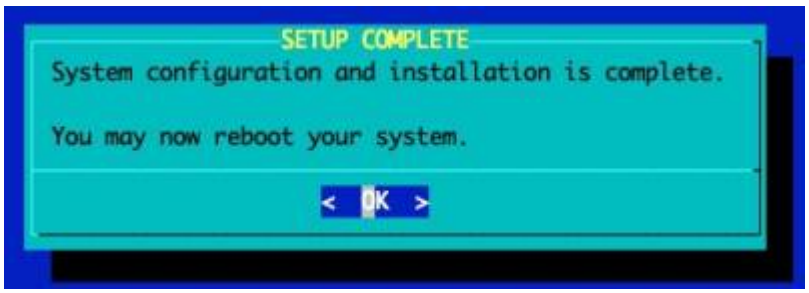
Configure GUI Window Manager

This author recommends using XFCE as it's light weight versus KDE.

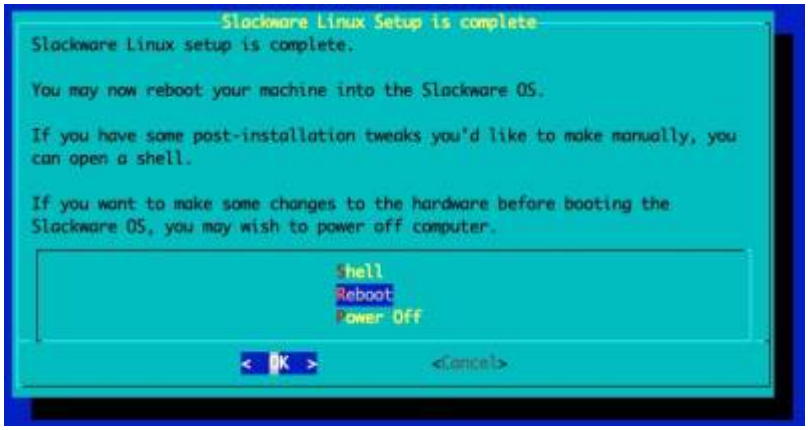
Continue Post Installation Configuration



Slackware Setup Complete



Some Pinebook Pro models encounter an issue where the MMC (SD card) interface fails to come online after a reboot. To resolve this, it's advisable to **power off** the machine rather than relying solely on a reboot.



Generally you'll want to reboot into the OS.

However, if you are planning on setting up RAID or need to customise the Operating System Initial RAM Disk, you should select 'Shell'.

The Slackware OS will be found within '/mnt'. You can use the 'os-initrd-mgr' tool (Video tutorial).

Booting the Slackware OS

The Slackware Installer will reboot into the Slackware OS.

```
Out:   vidconsole
Err:   vidconsole
Model: Pine64 Pinebook Pro
Net:   No ethernet found.
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1 is current device
Scanning mmc 1:1...
Found /extlinux/extlinux.conf
Retrieving file: /extlinux/extlinux.conf
6586 bytes read in 6 ms (1 MiB/s)
1:      Boot OS: Slackware Linux
Retrieving file: /initrd-armv8
```

Login to the Slackware OS

```
Welcome to Linux 5.14.4-armv8 aarch64 (tty1)

pinebookpro login:
```

You may now login as 'root', using the password you set within the installer.

Post Installation Configuration

There are a few post-installation configuration tasks to complete.

Initial Time/Date Sync

If Internet access is available to the PineBook Pro, prior to proceeding with any further setup, you may wish to set the time now as a one-off event:

Elevate yourself to ***root*** and use ***ntpdate***:

```
$ su -
# ntpdate rolex.ripe.net  # or pick your favourite NTP server
# hwclock -w # writes the date to the RTC
# logout
```

Add a plebeian user

You should add a plebeian (non-root) user using the 'adduser' tool.

This is documented here.

NTP (Network Time Protocol) setup

If your PineBook Pro has continuous Internet access, you may wish to configure it to sync time from an Internet NTP Server.

LCD panel brightness level

The LCD brightness level is controlled statically during boot. By default it'll set the maximum brightness, which is '4000'. The lowest realistically that this author can read is 1000.

To change the brightness level:

As root, edit the Hardware Model run control configuration file:

```
# vi /etc/rc.d/rc.platform.conf # or substitute 'vi' for your favourite
editor
```

Within that configuration you'll find examples. Uncomment, set accordingly and reboot.

Use a graphical login manager

If you prefer to use a graphical login manager, you can configure the default runlevel as 4:

```
su -
sed -i 's?id:3:?id:4:?g' /etc/inittab
reboot
```

Managing Slackware on the Pinebook Pro

Keeping the Slackware OS up to date

One of the preferred tools to keep your system up to date is slackpkg.

Upgrading the Kernel: in Slackware x86/64 manual steps are required after upgrading the Kernel packages. In Slackware ARM, you simply upgrade the Kernel packages and reboot.

Loading Additional Linux Kernel Modules within the OS Proper

Often Kernel modules for discovered hardware will be automatically loaded, but occasionally you will need to manually configure the loading of some modules.

```
/etc/rc.d/rc.modules.local
```

This file is a shell script that is run as one of the last steps before the OS has fully booted. You can enter `modprobe` commands here to load the specific modules you require.

Configuration files within the directory `/lib/modprobe.d/` can be used to configure the parameters of the modules. Existing files within that directory serve as reference examples should you need them.

Loading Additional Linux Kernel Modules early in the boot sequence

There are a number of peripherals that may require Kernel modules loading early on in the boot sequence. An example of this would be RTCs (Real Time Clocks) or storage controllers that are required to access the file systems on which the OS lives.

Usually you won't need to load modules early in the boot sequence. See the previous section about loading modules from within the OS Proper.

To load Kernel modules during the early boot sequence, read:

```
/boot/local/README.txt
```

As root, the easiest way to begin is by renaming the example script:

```
mv /boot/local/load_kernel_modules.post.sample
/boot/local/load_kernel_modules.post
```

Then add the appropriate module loading commands to: `/boot/local/load_kernel_modules.post` You can also add shell code here to initialise a peripheral - writing something to the peripheral's Kernel interface, for example.

Slackware repository partition

The Slackware Installer image contains a type `ext4` partition labeled `SLKins_aio-pkgs` from which the packages are installed.

```
root@slackware:~# mount LABEL=SLKins_aio-pkgs /mnt/zip
root@slackware:~# cat /mnt/zip/README.txt
This file system contains the Slackware repository that is used during the
installation of Slackware.

Once you've booted into your OS you can delete or change this partition if
you
wish, or perhaps you might like to retain it for future reference.

root@slackware:~#
```

Most users simply leave the partition alone, as it causes no issues.

Customising the Slackware Linux Kernel

If you'd like to customise the Linux Kernel, the easiest way is to follow the HOWTO guide and use the Slackware ARM Kernel build script to create new packages.

Reducing Boot Time

Slackware ARM ships with a generic OS InitRD (Operating System Initial RAM Disk - the environment that prepares the machine to boot the Operating System Proper), so as to support a wide range of Hardware Models.

However, this isn't the optimal setup once the Slackware OS has been installed because the generic OS InitRD typically exceeds 250MB, which in some cases can add several seconds to the boot time whilst it's loaded from the SD card.

The `os-initrd-mgr` (Operating System Initial RAM Disk Manager) tool has an option to synchronize the OS InitRD's Kernel modules with *only* those presently loaded within the Operating System.

To do this:

```
$ su -c 'os-initrd-mgr --sync-loaded-kmods' -  # note the final -
```

To have this setting persist across Kernel upgrades, you must upgrade the `a/kernel-modules` package before `a/kernel`. If not, it'll revert to the generic OS InitRD until you next reboot. If you are using `slackpkg` to manage upgrades, this is handled for you.

This option isn't the default, but you can make it so by following the instructions within `/etc/os-initrd-mgr.conf.sample`

This way when you upgrade the Kernel packages in the order described above, it'll automatically synchronize the modules.

os-initrd-mgr has a safety check to only proceed when the running Kernel and incoming Kernel are at the same major version and patch level.

For example, when running Linux 5.17.1, upgrading to 5.17.2 will work; but an upgrade of Linux 5.17.1 → 5.18.1 will require a reboot then to run os-initrd-mgr again to re-sync.

If at any point you want to revert to the generic OS InitRD, simply reinstall the `a/kernel` package (and unset the setting if you configured it in `/etc/os-initrd-mgr.conf`).

Suspend to RAM

To suspend, in a console its fast to test by running “loginctl suspend”.

The sleep button works. Upon waking up just tap the power button opening and closing the lid does the same.

Installing extra Software

Slackware comes with a good base of software applications, but there are plenty more available in the Open Source Ecosystem.

The best way to add new software is to use the build scripts from SlackBuilds.org.

Managing the Boot Loader firmware

During the Slackware installation process, you are offered the opportunity to flash the Boot Loader to the SPI flash. Occasionally, updates will be made available to the Boot Loader to address bug fixes and provide improvements.

Slackware provides a package `hwm-bw-rk3399` within the a series that contains the latest Boot Loader firmware and contains the `bootloader-flash-rk3399` tool to manage the upgrade life cycle.

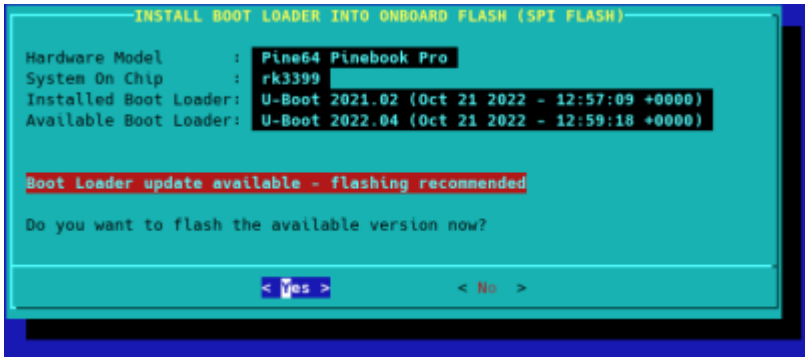
Upgrading the Boot Loader firmware

1. Upgrade to the latest available version of the package `hwm-bw-rk3399` (using `slackpkg` as described within this document).
2. As root, run the `bootloader-flash-rk3399` tool.

In this example we'll run the Boot Loader management tool as the `root` user using the `su` tool:

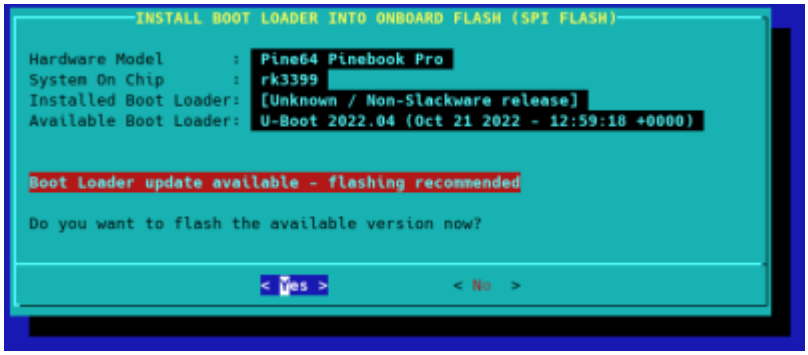
```
$ su -c 'bootloader-flash-rk3399' -  # You must include the trailing '-' character
```

When an upgrade is required, in most cases you'll see a screen like this which provides information about the currently installed Boot Loader and the newer version available:

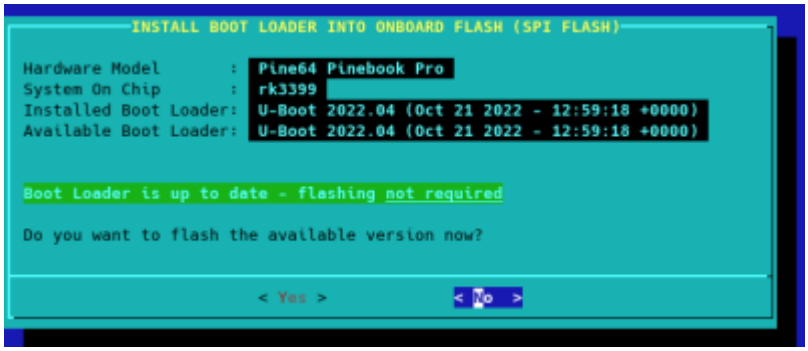


If you've wiped the Boot Loader from SPI flash or have installed a non-Slackware firmware build, you

will see a screen like this where the existing installed Boot Loader is unrecognised:

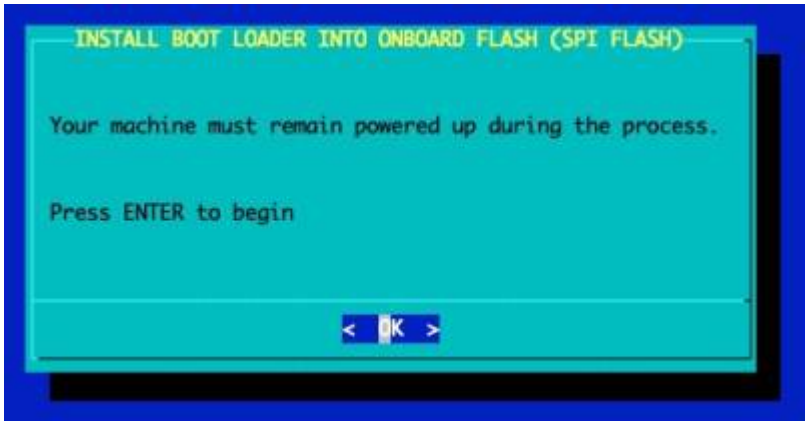


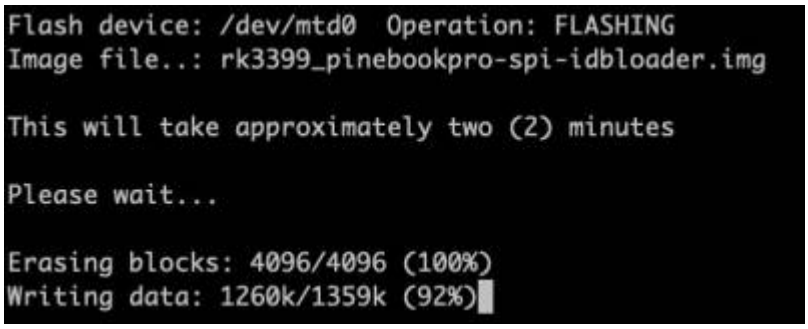
If the installed Boot Loader matches the currently available version, you will be advised that flashing is unnecessary. However, you can re-flash it if you wish.



If you proceed with flashing, the work flow looks like this and takes approximately two minutes to complete.

You must ensure the system has sufficient power/is plugged in. If the machine powers off before flashing completes, you would need to re-flash using the bootable SD card as described in the initial installation instructions.





Using the Serial/UART adapter

If you'd like to use a UART/Serial console, you'll need to **adjust the switch** that toggles Sound vs UART (you'll find this annotated in earlier sections of this document), and connect the adapter (details are in the Hardware requirements section above).

You can then connect the USB end of the adapter into your Linux Host Computer, and use the following command.

The USB→UART/Serial adapter connects to the Pinebook Pro's headphones socket.

```
screen -T screen-256color /dev/ttyUSB0 115200
```

This assumes that there are no other similar adapters occupying `/dev/ttyUSB0`. If so, you will need to adjust the device name accordingly (e.g. perhaps `/dev/ttyUSB1`).

Boot from NVME rather than from uSD card

The Slackware AArch64 installation is set to boot from the Micro SD card due to its high portability across various hardware models. Nevertheless, if your preference is to exclusively utilize the NVME, please refer to this post for guidance.

While booting from NVME offers the advantage of freeing up the SD Card slot, it's

essential to note that encountering boot issues could pose recovery challenges. Unlike SD cards, which allow easy diagnosis and fixing by inserting them into another machine, troubleshooting NVME boot problems can be more complex.

Known Limitations / Bugs

The Project Roadmap contains the major issues/bugs/feature gaps.

Issue	Work around	Notes
USB-C peripherals sometimes don't work	Turn the USB-C connector 180 degrees and re-insert it	Some USB-C peripherals (Network adapters in particular) are only detected when in a particular physical orientation. This may be due to the connector or socket being dirty or containing dust. Ensure everything is clean.
Sound state (despite being saved) doesn't persist across boots	None yet	This used to work until Linux 5.17. Will look into it

Exploring

Hibernate

The ARM Trusted Firmware firmware does not support the deep sleep / hibernate functions yet.
https://github.com/ARM-software/arm-trusted-firmware/search?q=hibernate

We use the Open Source U-boot firmware, with Trusted Firmware - ARM integration.

Hibernation changes for elogin.conf (once upstream support available)

```
#AllowHibernation=yes
#AllowSuspendThenHibernate=yes
#AllowHybridSleep=yes
If set those to "no"  they will not show up in the menus of xfce and kde.

Slackware Installer:
- The rk3399 helper script is primed to set resume= within extlinux.conf,
once support is available.

Resume points to the swap partition that hibernate uses to save the system
and reload the system.
https://docs.slackware.com/howtos:slackware_admin:hibernation?s[]=hibernate
```

There is an option in the logind.conf to set up hybrid sleep, where sleep then hibernation is activated.

https://forum.manjaro.org/t/hibernate-on-pinebook-pro/18045

Contributing to the Slackware ARM project

There are a plethora of ARM devices on the market which requires initial R&D and continuous testing. If you'd like to help Slackware support more ARM boards, please check out the documentation explaining how to get involved.

Supporting / Sponsoring the Slackware ARM project

Maintenance of the Slackware ARM port takes not only a lot of time, but also has financial costs such as the on-going use of electricity, Internet hosting and purchasing and maintenance of ARM hardware.

Once you find yourself enjoying using the ARM port of Slackware, please take a few moments to show your appreciation through sponsorship.