# Slackware ARM / AArch64 Kernel Module Loader

Work in progress - Aug 2023

However, it is explained in two videos from the Slackware ARM podcast.

Part 1 of 2 Part 2 of 2

The Kernel Module Loader script is responsible for setting the controlling the set of loaded Linux Kernel modules for a specific Hardware Model. It can also be used to configure hook functions to be executed both pre and post loading of the modules, in order to light up any particular hardware once its module/driver has been loaded.

#### Notes around Slackware Pre-boot Shell

Only works over UART/Serial port because the 1st stage opens prior to any modules being loaded. This generally precludes the use of HCI (Human Computer Interaction) peripherals such as keyboards and monitors.

## The Kernel Module Loader

The Slackware ARM/AArch64 Kernels are generic and modular - the same Kernel is used on multiple Hardware Models. Since ARM doesn't have a BIOS to support device discovery, in Slackware ARM we configure in advance the most common Kernel modules/device drivers for the supported Hardware Models. These modules are set and loaded by the 'Kernel Module Loader'.

### The Baseline set of modules

There is a baseline set of modules which are generally required on all platforms and therefore are set globally. Examples include the USB stack, file systems and so on.

The baseline set of modules for both 32bit and 64 bit ARM is configured in the main loader script. ARM platform-specific baseline configurations exist to augment/modify the base set depending on the particular ARM platform (32bit or 64bit). You will find AArch64's script here.

Even though the baseline set of modules isn't extensive, the generic Operating System Initial RAM Disk ('OS InitRD') and Slackware Installer contain a wide array of modules which can be loaded by the Hardware Model Loader scripts. The Liklihood is that the modules you require may already be included, just not set/loaded. However, if you cannot find the module(s) you require, drop a note to the Slackware ARM forum with the module name and Kernel `CONFIG_` option, the Hardware Model name and we can

probably add it to the generic OS InitRD and Slackware Installer.

# Hardware Model-specific modules

In addition to the base line set of rules, particular Hardware Models will require modules unique to themselves. e.g. you won't want to load the driver for the Raspberry Pi's network interface on the RockPro64, since it will never be used. Additionally, some modules surface error messages when loaded on Hardware Models on which the expected hardware isn't present.

Therefore each Hardware Model has its own Module Loader script.

The scripts are stored here.

Reference examples are 'rk3399' (supporting RockPro64 and Pinebook Pro) and 'bcm2711' (supporting Raspberry Pi 4). There are also scripts for other Hardware Models which may be useful to study.

## Populating the Kernel Module Loader scripts

In order to populate these scripts with the modules for your paricular board, the easiest way is to use a running Linux environment, such as:

- A Linux distribution that already supports the target Hardware Model.
- The Slackware Linux Installer

If you're using the Slackware Linux Installer, it'll boot and load the baseline set of modules. This may provide a reasonably useful environment to start, but most likely will lack the storage modules and a few of the other critical sub systems.

### Lighting up the hardware

There are a few scripts to help light up the core sub systems of your Hardware Model.

- load-all-modules : This loads every module found within /lib/modules. Note that it may cause the system to crash (as this sometimes happens when a 'bad' module is loaded). However, the idea behind this script is to light up as much hardware as possible so that you can determine which modules are required for your Hardware Model.

### Determining the modules for specific hardware

Once your hardware is online/lit up, you need to determine which Linux Kernel modules are responsible for supporting the core sub systems required to boot the OS proper.

The Kernel Module Loader scripts are **only** intended to load modules that are required to boot into the Operating System Proper (i.e. pivout out of the InitRD environment and into the Slackware OS). Modules for blue tooth, sound and so on should be loaded from within the OS Proper, **not** within the Initial Ram Disk (either the OS InitRD or Slackware Installer, in Slackware's case).

There are another two scripts to help:

- Find drivers based on path
- Find drivers for network interfaces

Within the Slackware OS InitRD or Slackware Installer, these tools are available within the `/tools` directory.

Finding devices by path:

#### Storage

```
root@bladswede:~/ac/source/k/scripts# ./find_modules_drivers /dev/sda | grep
DRIVER=
    DRIVER=sd
    DRIVER=ahci
    DRIVER=pcieport
    DRIVER=rockchip-pcie
```

```
root@bladswede:~/ac/source/k/scripts# ./find_modules_drivers /dev/mmcblk1 |
grep DRIV
    DRIVER=mmcblk
    DRIVER=dwmmc_rockchip
```

#### Real Time Clock (RTC)

```
root@bladswede:~/ac/source/k/scripts# ./find_modules_drivers /dev/rtc | grep
DRIVER=
    DRIVER=rk808-rtc
    DRIVER=rk808
    DRIVER=rk3x-i2c
```

#### Network Interfaces

```
root@bladswede:~/ac/source/k/scripts# ./network-mods
      eth0 [e2:8d:0f:6b:b1:a2]:   dwmac_rk (up)
```

**Video**

```
root@bladswede:~/ac/source/k/scripts# for card in /dev/dri/card* ; do
./find_modules_drivers $card  | grep DRIVER= ; done
    DRIVER=rockchip-drm
    DRIVER=panfrost
```

You take the driver names and add them to the relevant variable name within your Hardware Model Loader script.

#### Event hooks defined within the Hardware Model platform helper scripts:

.. this section is incomplete..

The RPi4 is an example - provide link to it.

hwm_hook_pre-modload hwm_hook_post-modload

[code]

```
    # Define a function to run from the OS InitRD's '/init' immediately
prior
    # to switching into the Slackware OS proper.
    # This is after any software RAID arrays et al have been initialised.
    # There's no current use case for this, but it may be useful in the
future.
    #
    # At this stage, the following paths for Slackware OS are mounted:
    # /proc, /sys, /run, /dev
    # The root file system ('/') is mounted under /mnt
    #
    #function hwm_hook_pre_switch_root() {
    #  echo "Just about to switch into the Slackware OS proper, leaving the
OS InitRD"
    #  sleep 4
    #}
```

and user locally defined hook_pre_switch_root

[/code]

## Linux Kernel package

doinst.sh calls

- os-initrd-mgr(8) (link to video) -

- Any helper scripts named 'pkg-kernel-*' found within /boot/platform/$ARCH/helper/. This enables handling any Hardware Model idiosyncrasies, such as in the case of the Raspberry Pi the Kernel et al from the standard OS location /boot, are duplicated onto the RPi's Hardware Model Boot Ware partition (/boot/platform/hwm_bw) to enable the use of the RPi's native Boot Loader.