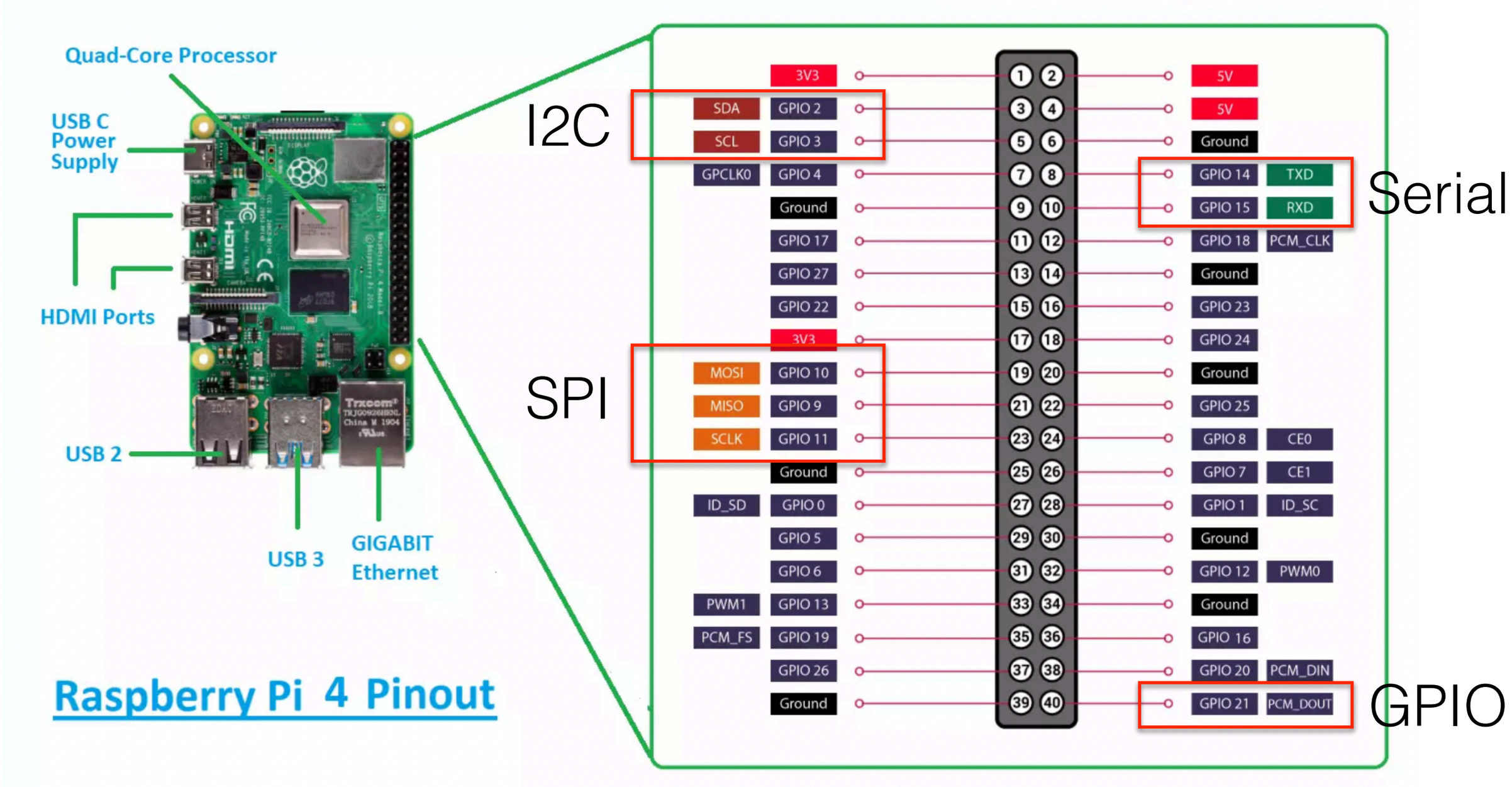
Embedded Linux Device Driver Programming Kernel Mode

Norman McEntire

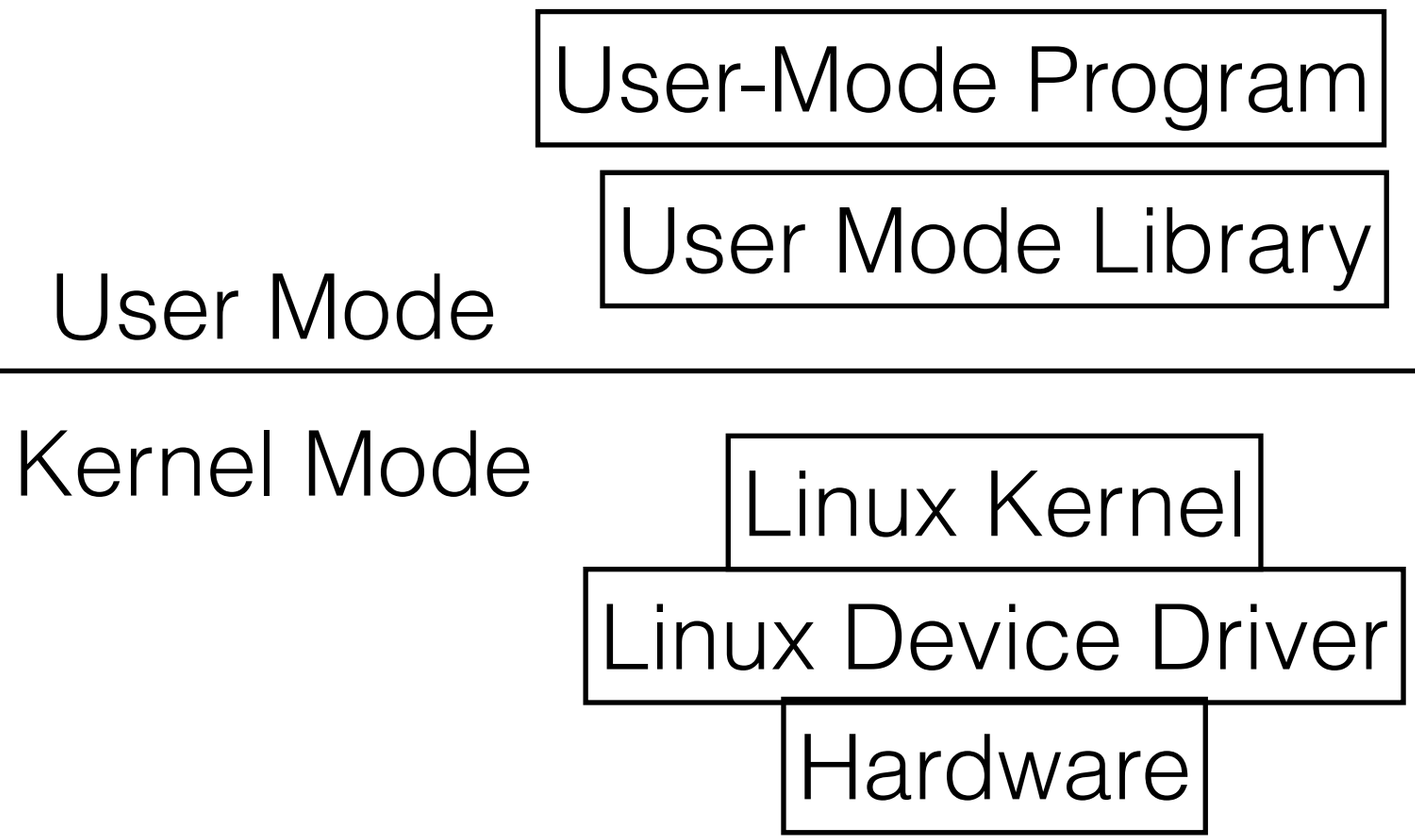
Review of User Mode Device Drivers

- Previous Lesson Covered User-Mode Device Drivers
  - User-Mode compared to Kernel- Mode
- User-Mode Demonstrations (using libpigio)
  - GPIO - General Purpose I/O Hardware
  - Serial Port Hardware
  - I2C Hardware
  - SPI Hardware

RPi 4 Pinout
We Used libpigpio



Block Diagram
User Mode and Kernel Mode



GPIO
/usr/include/pigpio.h

```
/*OVERVIEW

ESSENTIAL

gpioInitialise              Initialise library
gpioTerminate               Stop library

BASIC

gpioSetMode                 Set a GPIO mode
gpioGetMode                 Get a GPIO mode

gpioSetPullUpDown           Set/clear GPIO pull up/down resistor

gpioRead                    Read a GPIO
gpioWrite                   Write a GPIO
```

I2C
/usr/include/pigpio.h

```
I2C

i2cOpen                   Opens an I2C device
i2cClose                  Closes an I2C device

i2cWriteQuick             SMBus write quick

i2cReadByte               SMBus read byte
i2cWriteByte              SMBus write byte

i2cReadByteData           SMBus read byte data
i2cWriteByteData          SMBus write byte data

i2cReadWordData           SMBus read word data
i2cWriteWordData          SMBus write word data

i2cReadBlockData          SMBus read block data
i2cWriteBlockData         SMBus write block data

i2cReadI2CBlockData       SMBus read I2C block data
i2cWriteI2CBlockData      SMBus write I2C block data

i2cReadDevice             Reads the raw I2C device
i2cWriteDevice            Writes the raw I2C device
```

Serial
/usr/include/pigpio.h

```
SERIAL

serOpen                         Opens a serial device
serClose                        Closes a serial device

serReadByte                     Reads a byte from a serial device
serWriteByte                    Writes a byte to a serial device

serRead                         Reads bytes from a serial device
serWrite                        Writes bytes to a serial device

serDataAvailable                Returns number of bytes ready to be read

SERIAL_BIT_BANG_(read_only)

gpioSerialReadOpen              Opens a GPIO for bit bang serial reads
gpioSerialReadClose             Closes a GPIO for bit bang serial reads

gpioSerialReadInvert            Configures normal/inverted for serial reads
```

SPI
/usr/include/pigpio.h

```
SPI

spiOpen                   Opens a SPI device
spiClose                  Closes a SPI device

spiRead                   Reads bytes from a SPI device
spiWrite                  Writes bytes to a SPI device
spiXfer                   Transfers bytes with a SPI device
```

Linux Kernel Modules

Linux Kernel Module Topics

- Concepts
- Commands
- C Source Code
- Building, Loading, Testing
- Module Parameters

Intro to Linux Kernel Modules

- The majority of Linux Device Drivers are implemented at Linux Kernel Modules
- Advantages
  - Dynamically load/unload
  - Extend the kernel (run in kernel mode)
  - Have full access to the kernel API

Commands to Explore Kernel Modules

- Commonly Used
  - lsmod - List modules
  - modinfo - Get info on a module
  - modprobe - Load or remove (-r) a module
- Sometimes used
  - insmod - install module
  - rmmod - remove a module

man lsmod

```
LSMOD(8)                                                    lsmod

NAME
       lsmod - Show the status of modules in the Linux Kernel

SYNOPSIS
       lsmod
```

Demo: lsmod

```
# lsmod
Module                  Size  Used by
i2c_bcm2835            16384  0
spidev                 20480  0
spi_bcm2835            20480  0
rfcomm                 53248  4
cmac                   16384  3
algif_hash             16384  1
aes_arm64              16384  3
aes_generic            36864  1 aes_arm64
algif_skcipher         16384  1
. . .
```

Key Observations

=> Module Name

=> Size in bytes

=> Used by (in use)

man modinfo

```
MODINFO(8)                                modinfo                                MODINFO(8)

NAME
       modinfo - Show information about a Linux Kernel module

SYNOPSIS
       modinfo [-0] [-F field] [-k kernel] [modulename|filename...]

       modinfo -V

       modinfo -h

DESCRIPTION
       modinfo extracts information from the Linux Kernel modules given on the command line. If
       the module name is not a filename, then the /lib/modules/version directory is searched,
       as is also done by modprobe(8) when loading kernel modules.
```

Key Observations

=> /lib/modules/version directory

Demo: modinfo backlight

```
# modinfo backlight
filename:       /lib/modules/6.1.21-v8+/kernel/drivers/video/backlight/backlight.ko.xz
description:    Backlight Lowlevel Control Abstraction
author:         Jamey Hicks <jamey.hicks@hp.com>, Andrew Zabolotny <zap@homelink.ru>
license:        GPL
srcversion:     7FF7236741306 3866E2C9A8
depends:
intree:         Y
name:           backlight
vermagic:       6.1.21-v8+ SMP preempt mod_unload modversions aarch64
```

Key Observations

=> /lib/modules/6.1.21-v8+ directory
=> filename (.ko.xz)
=> description
=> author
=> license
=> srcversion
=> depends
=> vermagic

Demo: modinfo i2c_dev

```
# modinfo i2c_dev
filename:       /lib/modules/6.1.21-v8+/kernel/drivers/i2c/i2c-dev.ko.xz
license:        GPL
description:    I2C /dev entries driver
author:         Simon G. Vogl <simon@tk.uni-linz.ac.at>
author:         Frodo Looijaard <frodol@dds.nl>
srcversion:     64893A5032D01C5C2CF278A
depends:
intree:         Y
name:           i2c_dev
vermagic:       6.1.21-v8+ SMP preempt mod_unload modversions aarch64
```

Key Observations

=> /lib/modules/6.1.21-v8+ directory
=> filename (.ko.xz)
=> description
=> author
=> srcversion
=> depends
=> vermagic

Demo: /lib/modules

```
# ls /lib/modules
6.1.21+  6.1.21-v7+  6.1.21-v7l+  6.1.21-v8+


# uname -r
6.1.21-v8+
```

```
# ls /lib/modules/$(uname -r)
kernel       modules.builtin.alias.bin  modules.dep.bin
modules.symbols
modules.alias   modules.builtin.bin      modules.devname
modules.symbols.bin
modules.alias.bin  modules.builtin.modinfo    modules.order
modules.builtin   modules.dep      modules.softdep
```

modprobe

```
MODPROBE(8)                                                modprobe

NAME
       modprobe - Add and remove modules from the Linux Kernel

SYNOPSIS
       modprobe [-v] [-V] [-C config-file] [-n] [-i] [-q] [-b] [modulename]
              [module parameters...]

       modprobe [-r] [-v] [-n] [-i] [modulename...]

       modprobe [-c]

       modprobe [--dump-modversions] [filename]
```

Demo: modprobe -r

```
# lsmod | head
Module                  Size  Used by
i2c_bcm2835            16384  0
spidev                 20480  0
spi_bcm2835            20480  0
rfcomm                 53248  4
cmac                   16384  3
algif_hash             16384  1
aes_arm64              16384  3
aes_generic            36864  1 aes_arm64
algif_skcipher         16384  1


 # modprobe -r i2c_bcm2835


# lsmod | head
Module                  Size  Used by
spidev                 20480  0
spi_bcm2835            20480  0
rfcomm                 53248  4
```

Demo: modprobe

```
# lsmod | head -4
Module                  Size  Used by
spidev                 20480  0
spi_bcm2835            20480  0
rfcomm                 53248  4
```

```
# modprobe i2c_bcm2835
```

```
# lsmod | head -4
Module                  Size  Used by
i2c_bcm2835            16384  0
spidev                 20480  0
spi_bcm2835            20480  0
```

Steps for Building Kernel Module

- Install Kernel Header Files (one time step)
  - sudo apt update
  - sudo apt upgrade (** Important to get the version of header files you are using!)
  - sudo apt install linux-headers-$(uname -r)
    (or for RPi: sudo apt install raspberrypi-kernel-headers
- Create Your Kernel Module
  - mkdir km-1-hello
  - cd km-1-hello
  - Edit km-1-hello.c
  - Edit Makefile
  - make

Installing Kernel Headers Step 1

```
$ sudo apt update
Hit:1 http://archive.raspberrypi.org/debian bullseye InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian bullseye InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
63 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
$ sudo apt upgrade
. . .
```

Installing Kernel Headers
Step 2

```
$ sudo apt-cache search linux-headers
linux-headers-5.10.0-22-common - Common header files for Linux 5.10.0-22
linux-headers-5.10.0-22-common-rt - Common header files for Linux 5.10.0-22-rt
linux-libc-dev-alpha-cross - Linux Kernel Headers for development (for cross-compiling)
linux-libc-dev-amd64-cross - Linux Kernel Headers for development (for cross-compiling)
linux-libc-dev-arm64-cross - Linux Kernel Headers for development (for cross-compiling)
linux-libc-dev-armel-cross - Linux Kernel Headers for development (for cross-compiling)
linux-libc-dev-armhf-cross - Linux Kernel Headers for development (for cross-compiling)
linux-libc-dev-hppa-cross - Linux Kernel Headers for development (for cross-compiling)
. . .
linux-libc-dev-x32-cross - Linux Kernel Headers for development (for cross-compiling)
raspberrypi-kernel-headers - Header files for the Raspberry Pi Linux kernel
```

Installing Kernel Headers Step 3

```
$ dpkg -L raspberrypi-kernel-headers | head
/.
/lib
/lib/modules
/lib/modules/6.1.21+
/lib/modules/6.1.21-v7+
/lib/modules/6.1.21-v7l+
/usr
/usr/share
/usr/share/doc
/usr/share/doc/raspberrypi-kernel-headers
. . .
/usr/src/linux-headers-6.1.21+
/usr/src/linux-headers-6.1.21+/.config
/usr/src/linux-headers-6.1.21+/Documentation
/usr/src/linux-headers-6.1.21+/Documentation/Kconfig
/usr/src/linux-headers-6.1.21+/Documentation/Makefile
. . .
```

Installing Kernel Headers Step 4

```
$ sudo apt install raspberrypi-kernel-headers
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  raspberrypi-kernel-headers
0 upgraded, 1 newly installed, 0 to remove and 63 not upgraded.
Need to get 30.0 MB of archives.
After this operation, 193 MB of additional disk space will be used.
Get:1 http://archive.raspberrypi.org/debian bullseye/main armhf raspberrypi-kernel-
headers armhf 1:1.20230405-1 [30.0 MB]
Fetched 30.0 MB in 4s (6,919 kB/s)
Selecting previously unselected package raspberrypi-kernel-headers.
(Reading database ... 113285 files and directories currently installed.)
Preparing to unpack .../raspberrypi-kernel-headers_1%3a1.20230405-1_armhf.deb ...
Unpacking raspberrypi-kernel-headers (1:1.20230405-1) …
Setting up raspberrypi-kernel-headers (1:1.20230405-1) ...
```

Makefile

```
$ cat Makefile
# Makefile for building kernel module
obj-m := km-1-hello.o

KERNEL_SRC ?= /lib/modules/$(shell uname -r)/build

all default: modules
install: modules_install

modules modules_install help clean:
	$(MAKE) -C $(KERNEL_SRC) M=$(shell pwd) $@
```

Code Demo: km-1-hello.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Norman McEntire<norman.mcentire@gmail.com>");
MODULE_DESCRIPTION("Hello World Kernel Module");
MODULE_VERSION("0.1");

static int __init hello_init(void)
{
    printk(KERN_INFO "Hello Kernel Modulen");
    return 0;
}

static void __exit hello_exit(void)
{
    printk(KERN_INFO "Goodbye Kernel Module\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

Results of Building

```
$ make
make -C /lib/modules/6.1.21-v8+/build M=/home/metaembedded/km-1-hello modules
make[1]: *** /lib/modules/6.1.21-v8+/build: No such file or directory.  Stop.
make: *** [Makefile:10: modules] Error 2
```

Yikes! /lib/modules/6.1.21-v8+/build does not exist!

Web Search - Same Issue

kernel source build directory missing on 6.1.21-v8+

Post Reply

Search this topic...

kernel source build directory missing on 6.1.21-v8+

Fri Apr 14, 2023 8:42 am

I accidentally pulled the plug on a raspberry pi I have monitoring the solar rig in my camper van today and when it came back up it was not logging any data. This seems to have been because of a recent kernel update. The pi talks to the solar charge controller over an Exar USB-RS485 adaptor using modbus and I noticed that the device for the adaptor was missing so I went to try rebuilding the kernel module for it manually but it fails because the build directory appears to be missing from the kernel headers. If I look at the directory of the same version ending v7+ the the build directory is present but not for v8+

Exar USB-RS485
Modbus
“Device for the adapter was missing”

Web Search - Solution 1

> **dom wrote:** ↑
> **Fri Apr 14, 2023 9:47 am**
>
> Can you try adding "arm_64bit=0" to config.txt, reboot and try again?

```
ls /boot/
bcm2708-rpi-b.dtb       bcm2710-rpi-3-b-plus.dtb  cmdline.txt     fixup_x.dat
start4.elf
bcm2708-rpi-b-plus.dtb  bcm2710-rpi-cm3.dtb       config.txt      issue.txt
start4x.elf
bcm2708-rpi-b-rev1.dtb  bcm2710-rpi-zero-2.dtb    COPYING.linux   kernel7.img
start_cd.elf
bcm2708-rpi-cm.dtb      bcm2710-rpi-zero-2-w.dtb  fixup4cd.dat    kernel7l.img
start_db.elf
bcm2708-rpi-zero.dtb    bcm2711-rpi-400.dtb       fixup4.dat      kernel8.img
start.elf
bcm2708-rpi-zero-w.dtb  bcm2711-rpi-4-b.dtb       fixup4db.dat    kernel.img
start_x.elf
bcm2709-rpi-2-b.dtb     bcm2711-rpi-cm4.dtb       fixup4x.dat     LICENCE.broadcom
bcm2709-rpi-cm2.dtb     bcm2711-rpi-cm4-io.dtb    fixup_cd.dat    overlays
bcm2710-rpi-2-b.dtb     bcm2711-rpi-cm4s.dtb      fixup.dat       start4cd.elf
bcm2710-rpi-3-b.dtb     bootcode.bin              fixup_db.dat    start4db.elf
```

/boot/config.txt

```
$ wc -l /boot/config.txt
81 /boot/config.txt



$ sudo vim /boot/config.txt

$ tail /boot/config.txt

[pi4]
# Run as fast as firmware / board allows
arm_boost=1

# NCM Added 2023-06-11
arm_64bit=0

[all]
enable_uart=1
```

Before/After Reboot

```
$ uname -r
6.1.21-v8+
```

```
$ sudo reboot
Connection to 192.168.4.34 closed by remote host.
Connection to 192.168.4.34 closed.
```

```
$ ssh metaembedded@192.168.4.34
metaembedded@192.168.4.34's password:
Linux raspberrypi 6.1.21-v7l+ #1642 SMP Mon Apr  3 17:22:30 BST 2023 armv7l
```

```
$ uname -r
6.1.21-v7l+
```

Making the Kernel Module

```
$ cd km-1-hello/
```

```
$ make
make -C /lib/modules/6.1.21-v7l+/build M=/home/metaembedded/km-1-hello modules
make[1]: Entering directory '/usr/src/linux-headers-6.1.21-v7l+'
  CC [M]  /home/metaembedded/km-1-hello/km-1-hello.o
  MODPOST /home/metaembedded/km-1-hello/Module.symvers
  CC [M]  /home/metaembedded/km-1-hello/km-1-hello.mod.o
  LD [M]  /home/metaembedded/km-1-hello/km-1-hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-6.1.21-v7l+'
```

Key Observation: **km-1-hello.ko**

Exploring Results of Building Kernel Module

```
$ ls
km-1-hello.c   km-1-hello.mod    km-1-hello.mod.o  Makefile       Module.symvers
km-1-hello.ko  km-1-hello.mod.c  km-1-hello.o      modules.order
```

```
$ modinfo km-1-hello.ko
filename:       /home/metaembedded/km-1-hello/km-1-hello.ko
version:        0.1
description:    Hello World Kernel Module
author:         Norman McEntire<norman.mcentire@gmail.com>
license:        GPL
srcversion:     899E9F3BF91736B583D9648
depends:
name:           km_1_hello
vermagic:       6.1.21-v7l+ SMP mod_unload modversions ARMv7 p2v8
```

Loading Kernel Module

```
$ sudo modprobe km-1-hello.ko
modprobe: FATAL: Module km-1-hello.ko not found in directory /lib/modules/6.1.21-v7l+
```

```
$ sudo insmod km-1-hello.ko
```

```
 $ lsmod | head -4
Module                  Size  Used by
km_1_hello             16384  0
rfcomm                 49152  4
cmac                   16384  3
```

```
$ dmesg | tail -4
[   16.169102] Bluetooth: RFCOMM socket layer initialized
[   16.169122] Bluetooth: RFCOMM ver 1.11
[   16.560597] ICMPv6: process `dhcpcd' is using deprecated sysctl (syscall)
net.ipv6.neigh.wlan0.retrans_time - use net.ipv6.neigh.wlan0.retrans_time_ms instead
[  672.174716] km_1_hello: loading out-of-tree module taints kernel.
```

Unloading Kernel Module

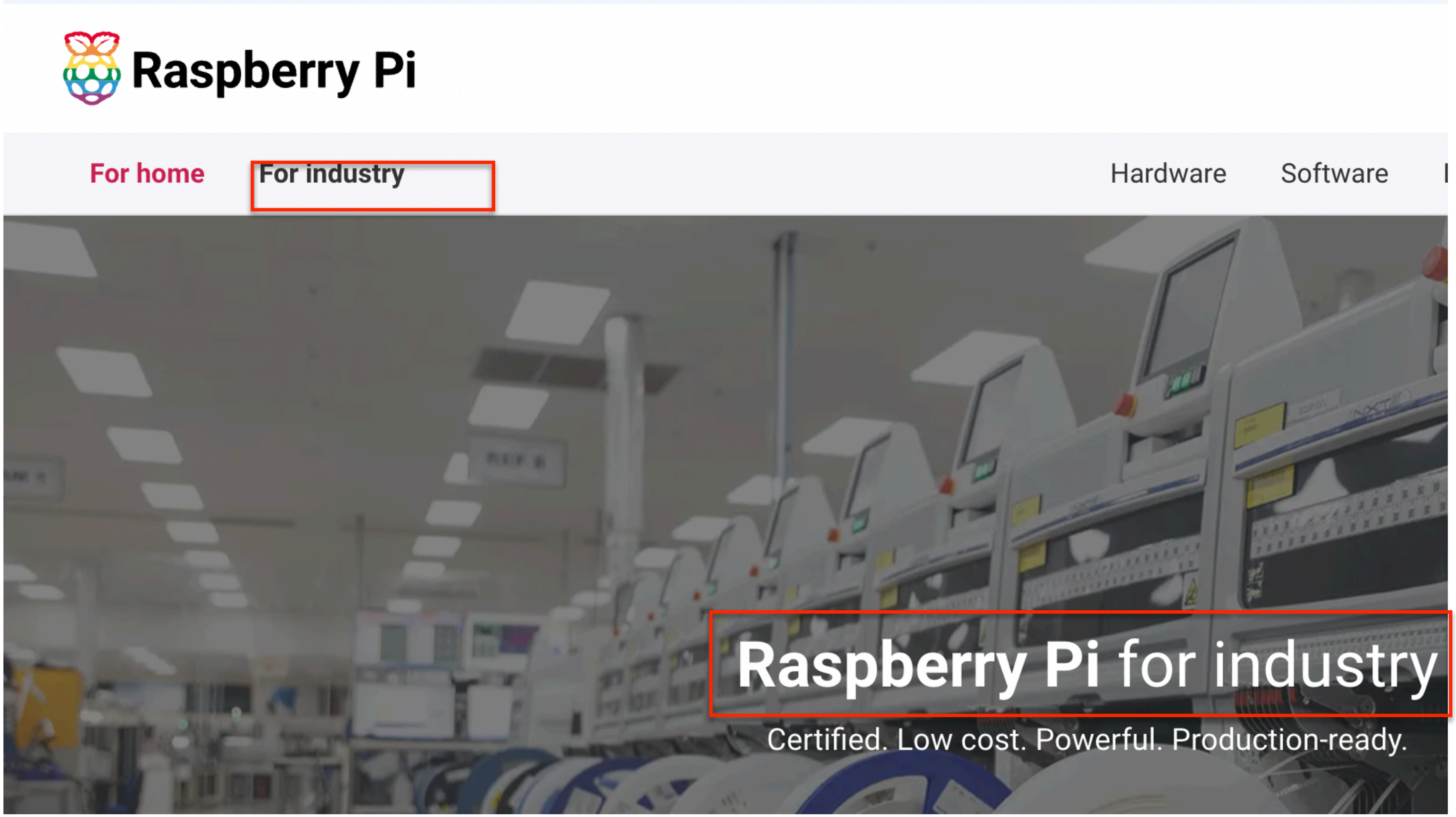
```
$ rmmod km_1_hello
rmmod: ERROR: ../libkmod/libkmod-module.c:799 kmod_module_remove_module() could not
remove 'km_1_hello': Operation not permitted
rmmod: ERROR: could not remove module km_1_hello: Operation not permitted

$ sudo rmmod km_1_hello


$ lsmod | head -4
Module                  Size  Used by
rfcomm                 49152  4
cmac                   16384  3
algif_hash             16384  1


$ sudo dmesg | tail -4
[   16.560597] ICMPv6: process `dhcpcd' is using deprecated sysctl (syscall)
net.ipv6.neigh.wlan0.retrans_time - use net.ipv6.neigh.wlan0.retrans_time_ms instead
[  672.174716] km_1_hello: loading out-of-tree module taints kernel.
[  672.175219] Hello Kernel Module
[  926.764912] Goodbye Kernel Module
```

Questions on Building Kernel Module?

We just built out-of-source tree kernel module.

Next step is to download RPi source

https://www.raspberrypi.com/software/



https://www.raspberrypi.com/software/

Raspberry Pi

Powered by Raspberry Pi

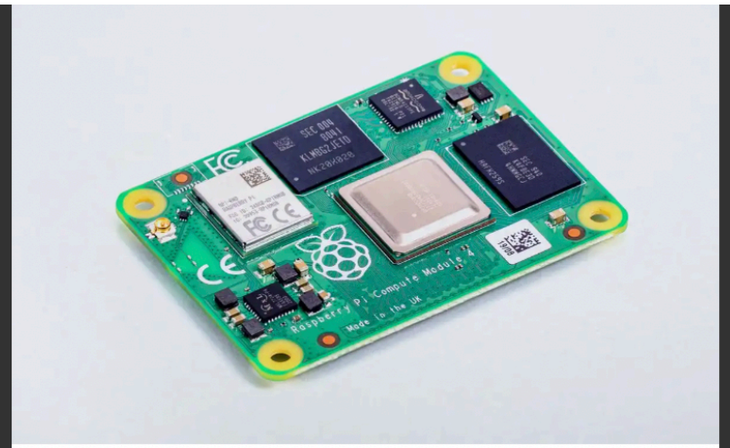
Powered by Raspberry Pi is the mark of quality used by companies that build Raspberry Pi technology into their products. Find out how you can apply, and discover products that take advantage of Raspberry Pi technology.

Find out more

https://www.raspberrypi.com/software/

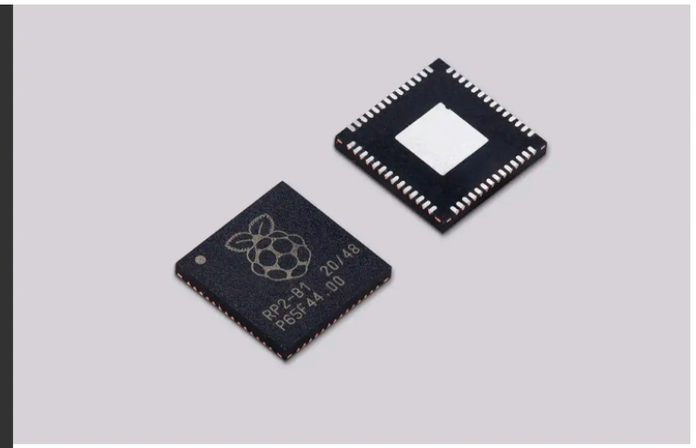
Raspberry Pi 4

- Full featured
- Accessory support
- Long production lifetime



Raspberry Pi Compute Module 4

- Small form factor
- Multiple models
- Long production lifetime



RP2040

- Extensively documented
- Perfect for machine learning applications
- Low cost

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

The Linux kernel

Kernel

Edit this on GitHub

The Raspberry Pi kernel is stored in GitHub and can be viewed at github.com/raspberrypi/linux; it follows behind the main Linux kernel. The main Linux kernel is continuously updating; we take long-term releases of the kernel, which are mentioned on the front page, and integrate the changes into the Raspberry Pi kernel. We then create a 'next' branch which contains an unstable port of the kernel; after extensive testing and discussion, we push this to the main branch.

Building the Kernel Locally

Building the Kernel Locally

On a Raspberry Pi, first install the latest version of Raspberry Pi OS. Then boot your Raspberry Pi, log in, and ensure you're connected to the internet to give you access to the sources.

First install Git and the build dependencies:

```
sudo apt install git bc bison flex libssl-dev make
```

Next get the sources, which will take some time:

```
git clone --depth=1 https://github.com/raspberrypi/linux
```

Demo

```
 $ mkdir raspberrypi

$ cd raspberrypi
```

```
$ git clone --depth=1 https://github.com/raspberrypi/linux
Cloning into 'linux'...
remote: Enumerating objects: 84004, done.
remote: Counting objects: 100% (84004/84004), done.
remote: Compressing objects: 100% (79496/79496), done.
remote: Total 84004 (delta 7439), reused 23346 (delta 3640), pack-reused 0
Receiving objects: 100% (84004/84004), 232.10 MiB | 6.79 MiB/s, done.
Resolving deltas: 100% (7439/7439), done.
Updating files: 100% (79315/79315), done.
```

Resulting linux directory

```
$ ls
arch      CREDITS          fs        ipc       lib          mm        rust      sound
block     crypto           include   Kbuild    LICENSES     net       samples   tools
certs     Documentation    init      Kconfig   MAINTAINERS  README    scripts   usr
COPYING   drivers          io_uring  kernel    Makefile     README.md security  virt
```

Default Kernel Configuration

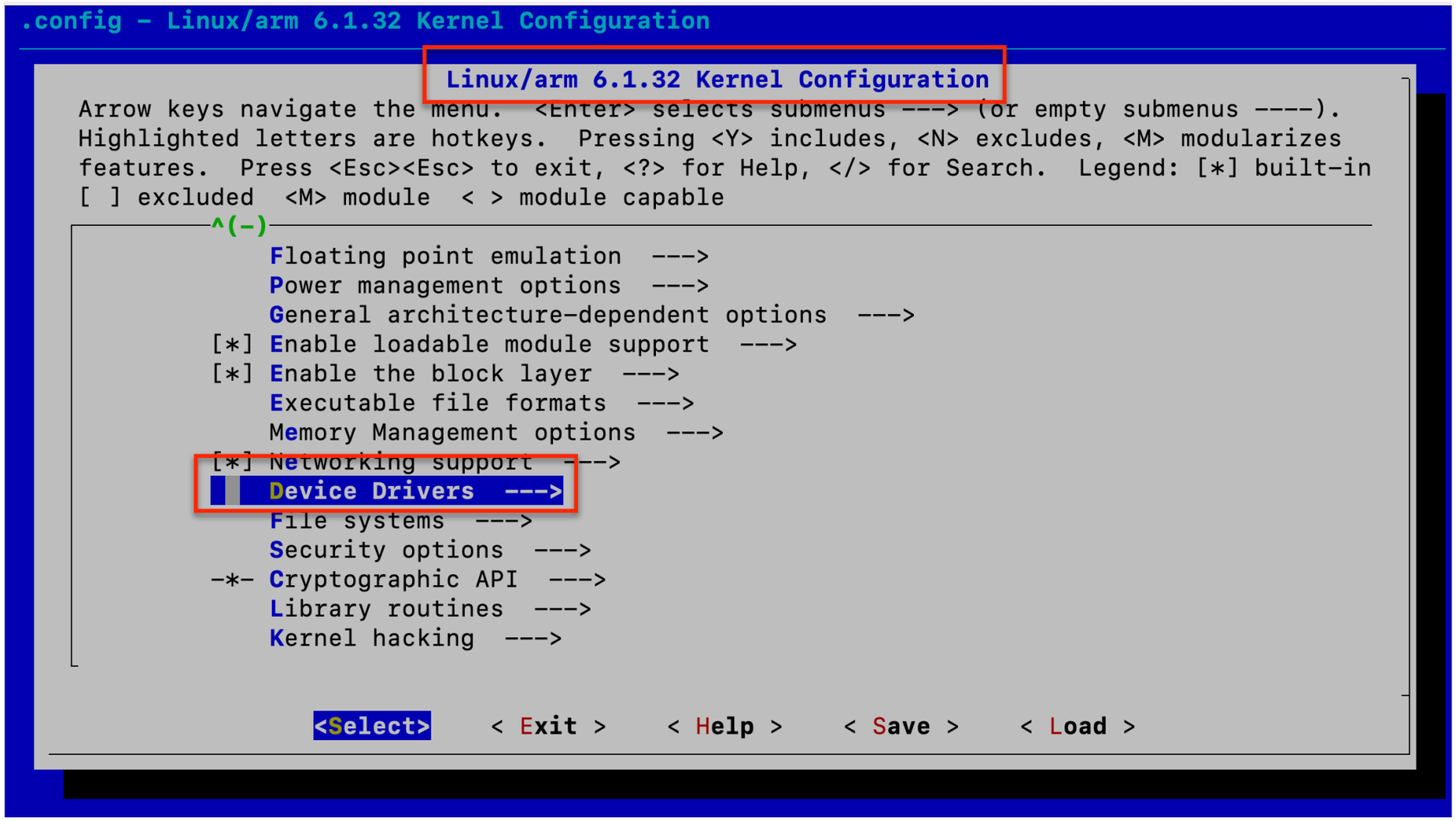
For Raspberry Pi 3, 3+, 4, 400 and Zero 2 W, and Raspberry Pi Compute Modules 3, 3+ and 4 default 64-bit build configuration

```
cd linux
KERNEL=kernel8
make bcm2711_defconfig
```

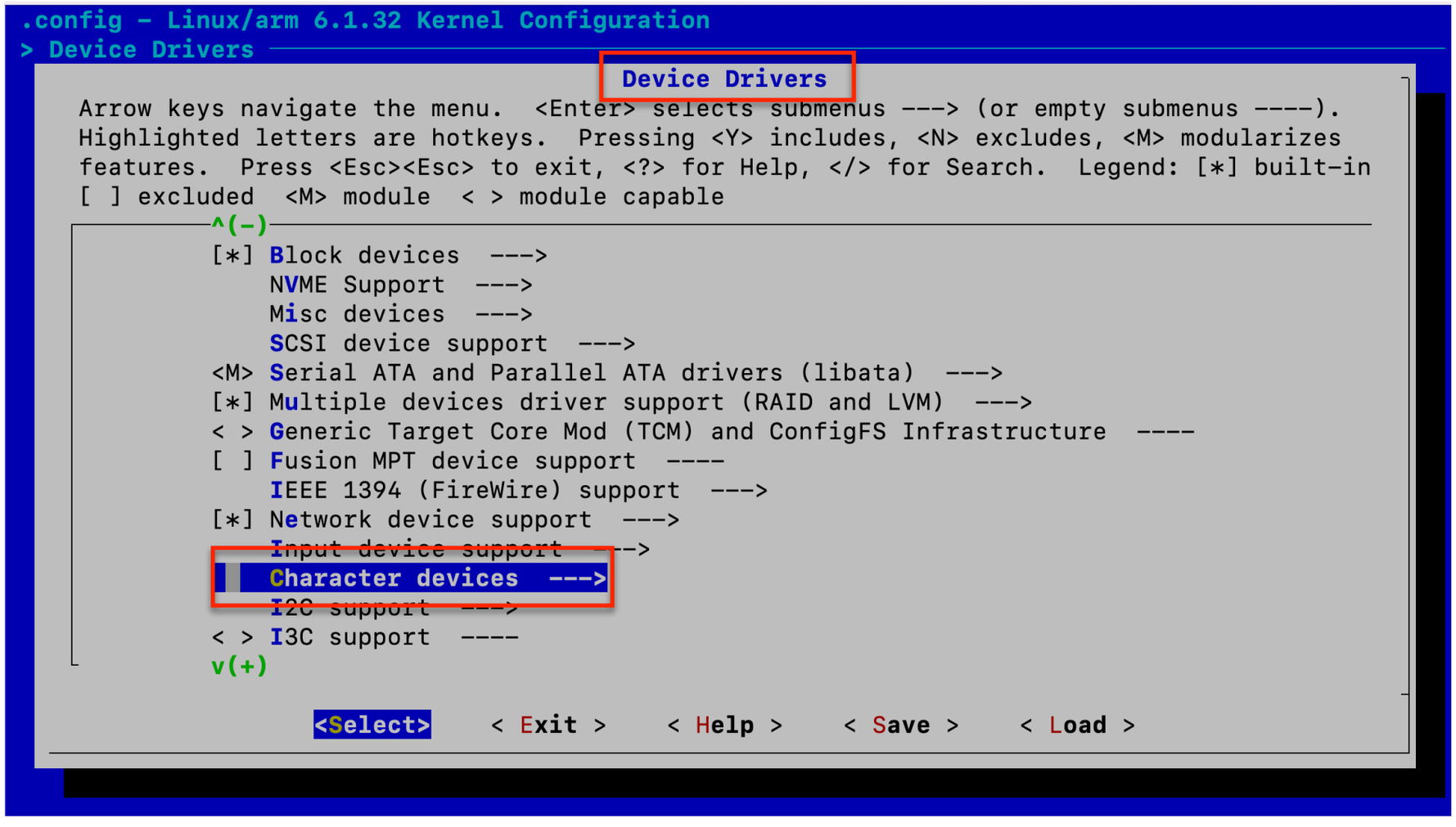
```
$ make bcm2711_defconfig
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#

$
```

Custom Kernel Configuration Top Level Menu

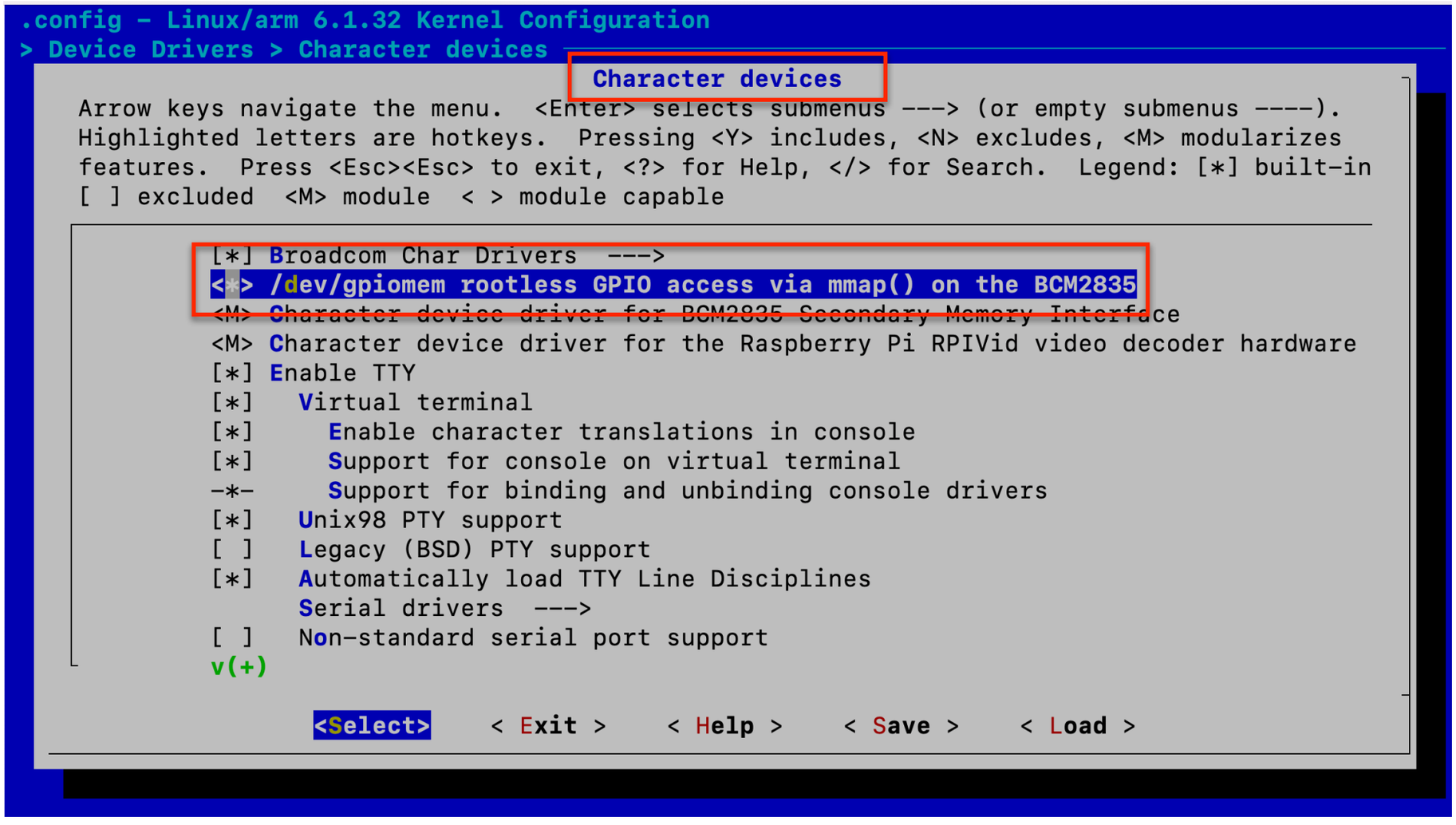
```
$ make menuconfig
```



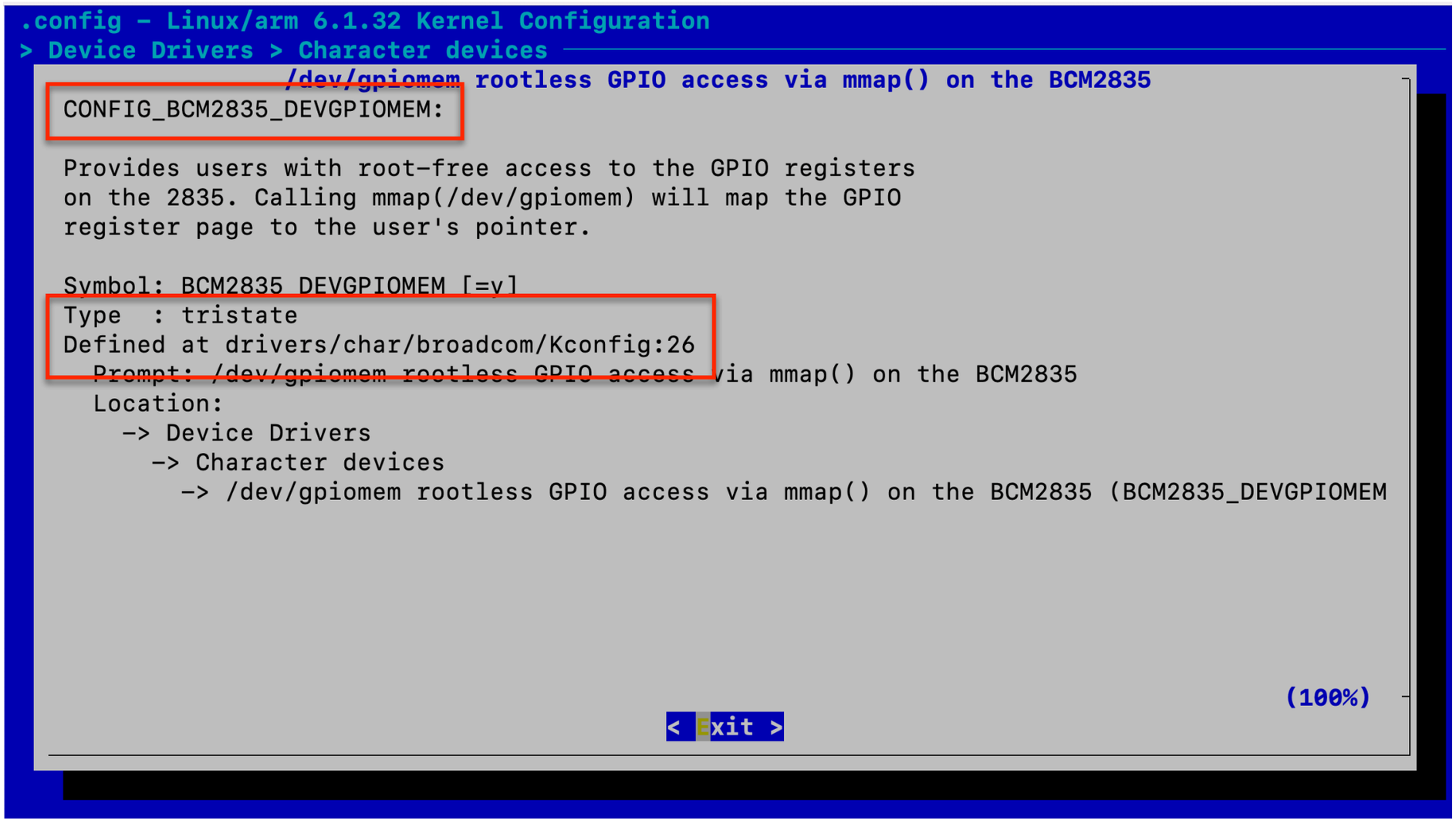
Custom Kernel Configuration
Device Drivers



Custom Kernel Configuration Character Devices



Custom Kernel Configuration Selecting Help on /dev/gpiomem



Kernel Source Code drivers/char/broadcom

```
$ ls
arch      CREDITS          fs         ipc      lib          mm         rust       sound
block     crypto           include    Kbuild   LICENSES     net        samples    tools
certs     Documentation    init       Kconfig  MAINTAINERS  README     scripts    usr
COPYING   drivers          io_uring   kernel   Makefile     README.md  security   virt
```

```
$ ls drivers/char/broadcom/
bcm2835-gpiomem.c  bcm2835_smi_dev.c  Kconfig
Makefile  rpivid-mem.c  vcio.c  vc_mem.c
```

```
$ wc -l drivers/char/broadcom/*
  258 drivers/char/broadcom/bcm2835-gpiomem.c
  409 drivers/char/broadcom/bcm2835_smi_dev.c
   49 drivers/char/broadcom/Kconfig
    5 drivers/char/broadcom/Makefile
  270 drivers/char/broadcom/rpivid-mem.c
  186 drivers/char/broadcom/vcio.c
  373 drivers/char/broadcom/vc_mem.c
 1550 total
```

Kernel Source Code drivers/char/broadcom/**Kconfig**

```
#
# Broadcom char driver config
#

menuconfig BRCM_CHAR_DRIVERS
   bool "Broadcom Char Drivers"
   help
     Broadcom's char drivers

. . .

config BCM2835_DEVGPIOMEM
   tristate "/dev/gpiomem rootless GPIO access via mmap() on the BCM2835"
   default m
   help
      Provides users with root-free access to the GPIO registers
      on the 2835. Calling mmap(/dev/gpiomem) will map the GPIO
      register page to the user's pointer.

. . .
```

config: BCM2835_DEVGPIOMEM

tristate: Off/On/Module

default: m (module)

Kernel Source Code drivers/char/broadcom/**Makefile**

```
obj-$(CONFIG_BCM2708_VCMEM)   += vc_mem.o
obj-$(CONFIG_BCM_VCIO)   += vcio.o
obj-$(CONFIG_BCM2835_DEVGPIOMEM)+= bcm2835-gpiomem.o
obj-$(CONFIG_BCM2835_SMI_DEV)+= bcm2835_smi_dev.o
obj-$(CONFIG_RPIVID_MEM)+= rpivid-mem.o
```

Note: CONFIG_BCM2835_DEVGPIOMEM
Will have one of three values: n, y, or m

n = do not build driver
y = build driver into kernel
m = build driver as kernel module

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 1 - Comments**

```
/**
 * GPIO memory device driver
 *
 * Creates a chardev /dev/gpiomem which will provide user access to
 * the BCM2835's GPIO registers when it is mmap()'d.
 * No longer need root for user GPIO access, but without relaxing permissions
 * on /dev/mem.
. . .
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 2 - Header Files/Defines**

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/of.h>
#include <linux/platform_device.h>
#include <linux/mm.h>
#include <linux/slab.h>
#include <linux/cdev.h>
#include <linux/pagemap.h>
#include <linux/io.h>

#define DEVICE_NAME "bcm2835-gpiomem"
#define DRIVER_NAME "gpiomem-bcm2835"
#define DEVICE_MINOR 0
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 3 - Private Variables**

```
struct bcm2835_gpiomem_instance {
   unsigned long gpio_regs_phys;
   struct device *dev;
};

static struct cdev bcm2835_gpiomem_cdev;
static dev_t bcm2835_gpiomem_devid;
static struct class *bcm2835_gpiomem_class;
static struct device *bcm2835_gpiomem_dev;
static struct bcm2835_gpiomem_instance *inst;
```

Note: Notice use of static to keep values private to file

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 4 - Device Open**

```
static int bcm2835_gpiomem_open(struct inode *inode, struct file *file)
{
   int dev = iminor(inode);
   int ret = 0;

   if (dev != DEVICE_MINOR) {
      dev_err(inst->dev, "Unknown minor device: %d", dev);
      ret = -ENXIO;
   }
   return ret;
}
```

Note: Notice use of static to keep functions private to file

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 5 - Device Close**

```
static int bcm2835_gpiomem_release(struct inode *inode, struct file *file)
{
   int dev = iminor(inode);
   int ret = 0;

   if (dev != DEVICE_MINOR) {
      dev_err(inst->dev, "Unknown minor device %d", dev);
      ret = -ENXIO;
   }
   return ret;
}
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 6 - Memory Map**

```
static int bcm2835_gpiomem_mmap(struct file *file, struct vm_area_struct *vma)
{
   /* Ignore what the user says - they're getting the GPIO regs
      whether they like it or not! */
   unsigned long gpio_page = inst->gpio_regs_phys >> PAGE_SHIFT;

   vma->vm_page_prot = phys_mem_access_prot(file, gpio_page,
                PAGE_SIZE,
                vma->vm_page_prot);
   vma->vm_ops = &bcm2835_gpiomem_vm_ops;
   if (remap_pfn_range(vma, vma->vm_start,
        gpio_page,
        PAGE_SIZE,
        vma->vm_page_prot)) {
      return -EAGAIN;
   }
   return 0;
}
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 7 - Operations**

```
static const struct file_operations
bcm2835_gpiomem_fops = {
   .owner = THIS_MODULE,
   .open = bcm2835_gpiomem_open,
   .release = bcm2835_gpiomem_release,
   .mmap = bcm2835_gpiomem_mmap,
};
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 8 - Probe - Part 1**

```
static int bcm2835_gpiomem_probe(struct platform_device *pdev)
{
   int err;
   void *ptr_err;
   struct device *dev = &pdev->dev;
   struct resource *ioresource;

   /* Allocate buffers and instance data */

   inst = kzalloc(sizeof(struct bcm2835_gpiomem_instance), GFP_KERNEL);

   if (!inst) {
      err = -ENOMEM;
      goto failed_inst_alloc;
   }

   inst->dev = dev;
. . .
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 9 - Probe - Part 2**

```
ioresource = platform_get_resource(pdev, IORESOURCE_MEM, 0);
if (ioresource) {
   inst->gpio_regs_phys = ioresource->start;
} else {
   dev_err(inst->dev, "failed to get IO resource");
   err = -ENOENT;
   goto failed_get_resource;
}

/* Create character device entries */

err = alloc_chrdev_region(&bcm2835_gpiomem_devid,
          DEVICE_MINOR, 1, DEVICE_NAME);
if (err != 0) {
   dev_err(inst->dev, "unable to allocate device number");
   goto failed_alloc_chrdev;
}
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 10 - Probe - Part 3**

```
cdev_init(&bcm2835_gpiomem_cdev, &bcm2835_gpiomem_fops);
bcm2835_gpiomem_cdev.owner = THIS_MODULE;
err = cdev_add(&bcm2835_gpiomem_cdev, bcm2835_gpiomem_devid, 1);
if (err != 0) {
   dev_err(inst->dev, "unable to register device");
   goto failed_cdev_add;
}
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 11 - Probe - Part 4**

```
  /* Create sysfs entries */

  bcm2835_gpiomem_class = class_create(THIS_MODULE, DEVICE_NAME);
  ptr_err = bcm2835_gpiomem_class;
  if (IS_ERR(ptr_err))
    goto failed_class_create;

  bcm2835_gpiomem_dev = device_create(bcm2835_gpiomem_class, NULL,
            bcm2835_gpiomem_devid, NULL,
            "gpiomem");
  ptr_err = bcm2835_gpiomem_dev;
  if (IS_ERR(ptr_err))
    goto failed_device_create;

  dev_info(inst->dev, "Initialised: Registers at 0x%08lx",
    inst->gpio_regs_phys);

  return 0;

. . .
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 12 - Probe - Part 5**

```
failed_device_create:
  class_destroy(bcm2835_gpiomem_class);
failed_class_create:
  cdev_del(&bcm2835_gpiomem_cdev);
  err = PTR_ERR(ptr_err);
failed_cdev_add:
  unregister_chrdev_region(bcm2835_gpiomem_devid, 1);
failed_alloc_chrdev:
failed_get_resource:
  kfree(inst);
failed_inst_alloc:
  dev_err(inst->dev, "could not load bcm2835_gpiomem");
  return err;
}
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 13 - Remove**

```
static int bcm2835_gpiomem_remove(struct platform_device *pdev)
{
   struct device *dev = inst->dev;

   kfree(inst);
   device_destroy(bcm2835_gpiomem_class, bcm2835_gpiomem_devid);
   class_destroy(bcm2835_gpiomem_class);
   cdev_del(&bcm2835_gpiomem_cdev);
   unregister_chrdev_region(bcm2835_gpiomem_devid, 1);

   dev_info(dev, "GPIO mem driver removed - OK");
   return 0;
}
```

Kernel Source Code drivers/char/broadcom/**bcm2835-gpiomem.c - Part 14 - Registration**

```
static const struct of_device_id bcm2835_gpiomem_of_match[] = {
   {.compatible = "brcm,bcm2835-gpiomem",},
   { /* sentinel */ },
};

MODULE_DEVICE_TABLE(of, bcm2835_gpiomem_of_match);

static struct platform_driver bcm2835_gpiomem_driver = {
   .probe = bcm2835_gpiomem_probe,
   .remove = bcm2835_gpiomem_remove,
   .driver = {
         .name = DRIVER_NAME,
         .owner = THIS_MODULE,
         .of_match_table = bcm2835_gpiomem_of_match,
         },
};

module_platform_driver(bcm2835_gpiomem_driver);

MODULE_ALIAS("platform:gpiomem-bcm2835");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("gpiomem driver for accessing GPIO from userspace");
MODULE_AUTHOR("Luke Wren <luke@raspberrypi.org>");
```

ls /dev/gpiomem

```
$ ls -l /dev/gpiomem
crw-rw---- 1 root gpio 245, 0 Jun 11 15:41 /dev/gpiomem
```

```
$ lsmod | grep gpio
```

```
$ grep DEVGPIOMEM .config
CONFIG_BCM2835_DEVGPIOMEM=y
```

Customizing Kernel Version

Customising the Kernel Version Using `LOCALVERSION`

In addition to your kernel configuration changes, you may wish to adjust the `LOCALVERSION` to ensure your new kernel does not receive the same version string as the upstream kernel. This both clarifies you are running your own kernel in the output of `uname` and ensures existing modules in `/lib/modules` are not overwritten.

To do so, change the following line in `.config`:

```
CONFIG_LOCALVERSION="-v7l-MY_CUSTOM_KERNEL"
```

```
#
# General setup
#
CONFIG_INIT_ENV_ARG_LIMIT=32
# CONFIG_COMPILE_TEST is not set
# CONFIG_WERROR is not set
CONFIG_LOCALVERSION="-v7l-hello"
# CONFIG_LOCALVERSION_AUTO is not set
```

Building The Kernel

Building the Kernel

Build and install the kernel, modules, and Device Tree blobs; this step can take a **long** time depending on the Raspberry Pi model in use. For the 32-bit kernel:

```
make -j4 zImage modules dtbs
sudo make modules_install
sudo cp arch/arm/boot/dts/*.dtb /boot/
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
```

For the 64-bit kernel:

```
make -j4 Image.gz modules dtbs
sudo make modules_install
sudo cp arch/arm64/boot/dts/broadcom/*.dtb /boot/
sudo cp arch/arm64/boot/dts/overlays/*.dtb* /boot/overlays/
sudo cp arch/arm64/boot/dts/overlays/README /boot/overlays/
sudo cp arch/arm64/boot/Image.gz /boot/$KERNEL.img
```