Dual boot firmware

Objective

The objective is to produce a USB drive with a Linux partition on it and an SD card containing firmware that will cause that drive to boot up in Raspberry Pi OS (aka Raspbian). Once this has been done, it is a relatively simple matter to make changes to the firmware on the SD card to allow a dual boot.

Method

The standard distro for Raspian is designed to fit onto a range of SD cards so long as they are at least 8.01 GB (8602517504 bytes) in size. Writing this image to an SD card means that only part of the full capacity of the SD card is actually used. The same image can be written to a USB drive.

If you read back the image from the SD card (or drive) it will read the whole capacity of the card (or drive) - the image you just wrote to it plus the remainder of the card, whatever it happens to contain.

The first time that Raspbian starts up, it is set up to change the partition size of the drive to its full capacity without losing any of the data on the drive. It also changes the start up scripts 'CMDLINE.TXT' and 'CONFIG.TXT' in two respects - the 'resize' option is removed and the UUID of the partition where Linux is to be found is updated.

The CMDLINE.TXT file is used extensively by Linux (not all of which I understand) and it refers explicitly to the ext4 (Linux) partition by means of a magic number (UUID) which is only known after the SD card partition has been created.

Step 1 is to write the standard Raspbian distro image to a USB drive - this will become a two partition drive with FAT and ext4 partitions that occupy only the first 8GB of the drive.

Step 2 is to write the standard RISC OS distro onto an SD card - this will become a two partition card with overlapping filecore and FAT partitions using only the first 1876Mbytes of the card.

Step 3 is to copy the files in the FAT partition on the USB drive onto the FAT partition on the SD card. The files 'CMOS' and 'RISCOS.IMG' will be all that is left of the RISC OS firmware but the filecore partition will be unchanged.

You now have an SD card which will boot into Raspbian, with all the Linux data on the USB drive.

Step 4 is to take a Raspberry Pi model 4B with the SD card and USB drive and boot into Raspbian, allow it to resize itself and shut it down.

```
disable_gamma console=serial0,115200 console=tty1 root=PARTUUID=3a324232-02
rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet init=/usr/lib/raspi-
config/init_resize.sh splash plymouth.ignore-serial-consoles
```

The contents of CMDLINE/TXT - the location of the root file system is specified by a partition UUID, highlighted above. For a particular SD card image this magic number is known. If you specify the correct partition UUID for an ext4 partition on a USB pen drive then Linux will boot from that pen drive. When it resizes the ext4 partition the UUID changes but it updates the CMDLINE.TXT ***on the pen drive****.*

Whereas RISC OS requires very few, if any, commands in the CMDLINE/TXT file (`disable_gamma` *and* `disable_mode_changes` *for example), Linux requires lots of stuff, which RISC OS can happily ignore.*

The partition UUID is a unique identifier for a partition and may refer to a USB drive or SD card.

A current version of Linux would have allowed the command `ls -l /dev/disk/by-partuuid` *to discover the required information.*

```
[gpio5=1]
fake_vsync_isr=1
framebuffer_swap=0
gpu_mem=64
init_emmc_clock=100000000
ramfsfile=CMOS
ramfsaddr=0x508000
kernel=RISCOS.IMG
device_tree=
hdmi_drive=2
hdmi_blanking=1
disable_overscan=1
[pi4]
enable_gic=1
[all]
[gpio5=0]
# Additional overlays and parameters are
documented /boot/overlays/README
# Enable audio (loads snd_bcm2835)
dtparam=audio=on
[pi4]
# Enable DRM VC4 V3D driver on top of the
dispmanx display stack
dtoverlay=vc4-fkms-v3d
max_framebuffers=2
[all]
#dtoverlay=vc4-fkms-v3d
```

The edited CONFIG/TXT file - the conditional statements are in square brackets with the RISC OS bit highlighted.

Step 5 is to edit the 'CMDLINE.TXT' and CONFIG.TXT' files on the SD card to make it into a dual boot card.

First add the highlighted bit to CONFIG.TXT - this adds a conditional bit for gpio5=1 (switch open) that is the normal RISC OS stuff with an extra enigmatic (but essential) blank device tree line. The Raspbian bit is also now contained in a gpio5=0 condition.

Now take the updated contents of the CMDLINE.TXT file on the USB drive (which will have been updated to remove the 'resize' command and to have an updated UUID for the ext4 partition on the USB drive that has now been expanded to occupy the whole) and copy it to the FAT partition on the SD card.

The SD card is now a dual-boot SD card with RISC OS as the default.

Detail

Let us look at the Raspian distribution (see illustration on next page) - this is also a two-partition SD card image which includes a 256MB fat partition and a 3384MB Linux partition which is seen (by Linux) as /dev/sdb2. Note that the contents of both partitions are displayed.

In order to create an SD card that will work on both operating systems then we will need a FAT partition of 256MB and we may as well make the filecore partition occupy the rest of the SD card - the SystemDisc software can create the necessary 'blank' SD card.

A switch on the 40 pin header between pins 29 (GPIO5) and 30 (GND) will enable a decision to be taken at boot time whether to boot into Linux or RISC OS. The changes we have made to the SD card firmware will cause a boot into Linux if the switch is closed and into RISC OS if it is left open (or not present).

Now we have a switch on the header to pull GPIO5 to LOW if the switch is pressed, the changes to CONFIG/TXT will now take different action on booting. There is an option to examine a GPIO pin and make execution conditional.

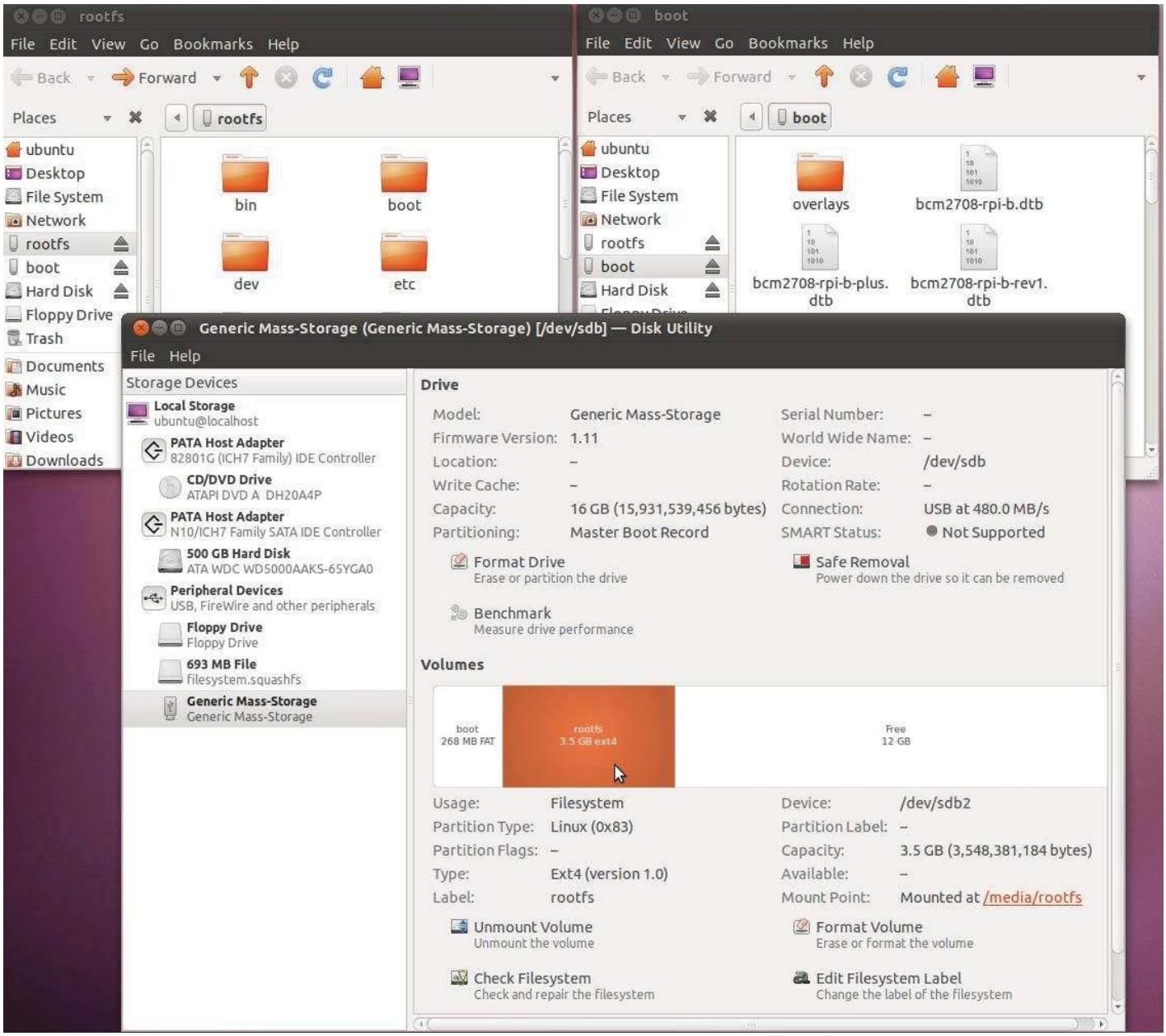
The first line of the file jumps over the RISC OS bits if GPIO 5 is pulled low. Linux has done its stuff in the CMDLINE/TXT file and needs little in this file. RISC OS does not seem put out by it.

Complications on Pi 4

If you move an SD card between a Pi 4 and another model, you need to delete the directory $.!Boot.Choices.Boot so that it is repopulated on start up to ensure the correct network drivers are loaded.

Backing up as you go

Now we can backup the SD card image (Win32DiskImager will create an image file the whole length of the SD card but this file can be truncated to a shorter



The standard Raspian distribution has two partitions - a FAT and a Linux partition. These do not overlap.

length - the same length as the image file that was written to the card - so that it will fit various different makes of 8GB SD card. The purpose of this backup is so that a fresh card can be created and retested if things go wrong! Also the first boot does some complicated things to the SD card image that has taken so long to create and it is wise to be able to do the testing on a known SD card image.

A short programme in BBC BASIC for Windows will do this - RISC OS struggles to handle files which are so large. See inset below for details.

```
REM Truncate to 30,000MB
F%=OPENUP"C:\Data\30G_armini.img"
PTR#F%=0
L%=1392508928
H%=7
SYS "SetFilePointer",@hfile%(F%),L%,^H%,0
SYS "SetEndOfFile",@hfile%(F%)
SYS "CloseHandle",@hfile%(F%)
END
```

*This programme truncates the file to a length of 1328MB plus 7*4096MB = 30000MB. To truncate to 7000MB then L%=3045064704 and H%=1 and to truncate to 4GB plus 1875MB we need L%=1875*1024*1024 and H%=1. This image is 5971MB and so should comfortably fit onto an 8GB SD card.*

The SD card can be recreated (by writing the image to the card again) between each attempt so that the fresh install is preserved until we get it right.

By downloading the Raspian distro and flashing it to a USB pen drive (rather than to an SD card) means that the CMDLINE.TXT file in the FAT partition of the USB drive has the correct partition UUID for the ext4 partition on the USB drive.

This means that booting from an SD card, with the same CMDLINE.TXT file (and CONFIG.TXT file) in its FAT partition, will cause the Pi to boot into Raspbian on the USB drive. A few simple changes to the CONFIG.TXT file will means that the Pi will boot into RISC OS.

Did this work?

Almost. Hold the button down and the Raspberry Pi booted into the Linux distribution on the USB pen drive. It then displayed a message to say that it was extending the ext4 partition to fill the drive to use the whole of the capacity available. It then said ... rebooting in 5 seconds.

When it rebooted I had let go of the button by then and so it booted into RISC OS. No problem, I thought, I'll reboot and hold the button down this time. It then just kept rebooting until I let go of the button and then booted into RISC OS.

I then had a moment of inspiration and looked at the CMDLINE/TXT file on the pen drive. This was now different - the newly resized ext4 partition had a different UUID. So all I had to do was to copy the edited CMDLINE/TXT file from the FAT partition on the pen drive to the FAT partition on the SD card and it all worked.

Adding the huge number of extra files in the FAT partition of the RasPiOS distribution and the much larger CMDLINE/TXT and retesting on a Pi 4 showed that these extra files made no difference to RISC OS. They would be needed to start up RasPiOS although it would then run from the USB drive.

Testing the same SD card on the Pi 400 gave some problems - after a lot of trouble-shooting I found by trial and error that one particular file 'bcm-2711-rpi-4-b/dtb' was needed for Raspian but, if present, prevented RISC OS from starting up on the Pi 400.

So the dual boot worked perfectly on the Pi 4 - press the button and booting was to RasPiOS, release it and booting was to RISC OS. Shutdown either system and the restart would be selected by the button - RISC OS or RasPiOS.

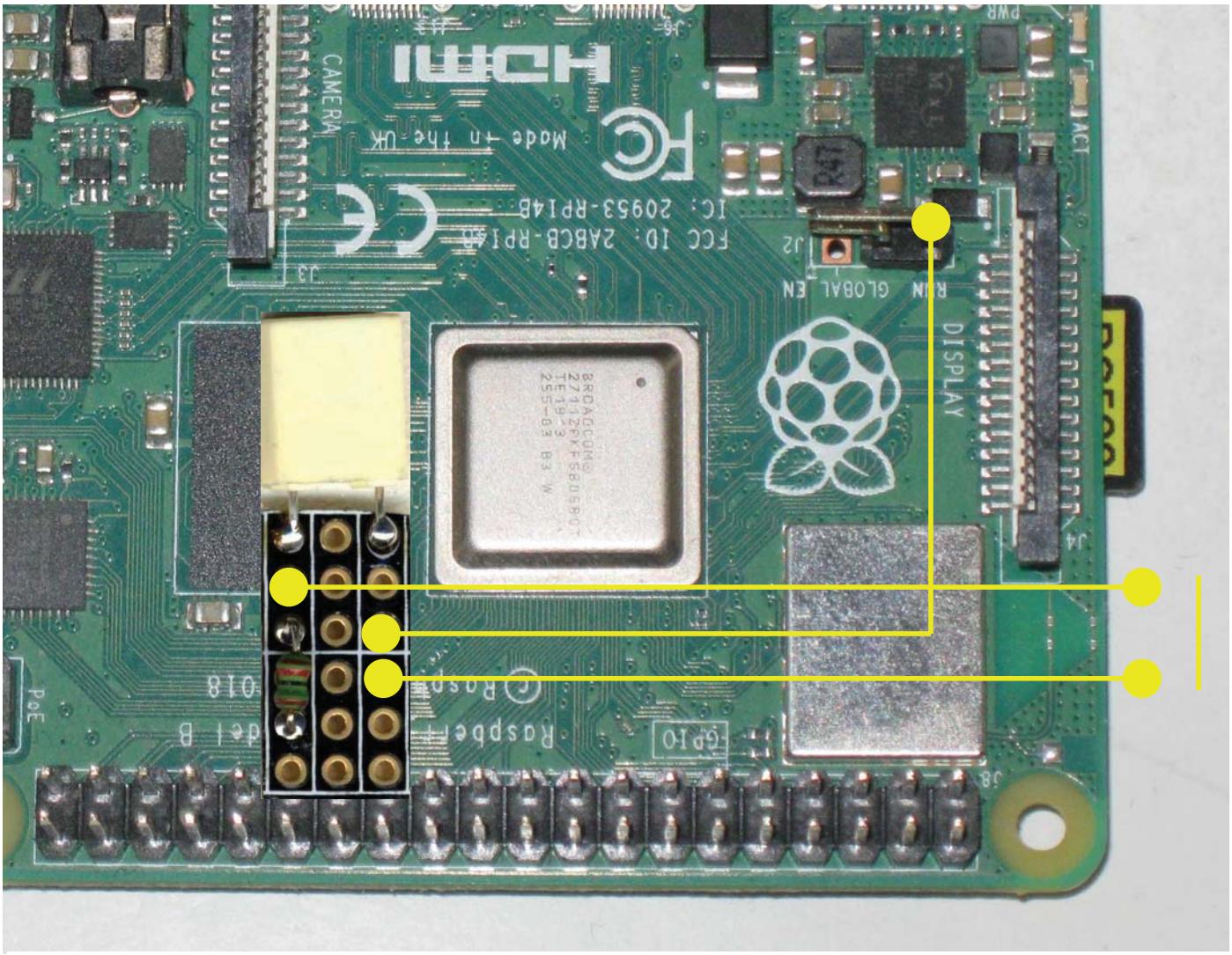
When the Raspberry Pi starts, two files are passed to the GPU - start4.elf (which specifies the VideoCore firmware in use) and fixup4.dat (which specifies memory locations). There are two methods, mutually exclusive, for the GPU to pass information to the kernel: the start.elf file will either load a device tree blob (if present) appropriate to the hardware or will put such information (called ATAG) into memory at 0x100.

Adding a command device_tree= to CONFIG.TXT forces the latter method for RISC OS and it all worked.

Chris Hall *chris@svrsig.org*

```
disable_gamma console=serial0,115200 console=tty1 root=PARTUUID=12345678-02
rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash
plymouth.ignore-serial-consoles
```

The CMDLINE/TXT file on the USB pen drive has been silently changed to have the new partition UUID for the ext4 partition on that drive. The 'init' command to resize the partition on first boot has also been removed.



A small circuit board incorporates a 15k resistor, a 680nF capacitor and a switch or push button. It connects to pin 29 (GPIO 5) and to pin 25 (GND). Flying leads connect to the switch and to the 'RUN' pin.