RPiconfig

From eLinux.org

As the Raspberry Pi doesn't have a conventional BIOS, the various system configuration parameters that would normally be kept and set using the BIOS are now stored in a text file named "config.txt".

The Raspberry Pi config.txt file is read by the GPU before the ARM core is initialized.

This file is an optional file on the boot partition. It would normally be accessible as /boot/config.txt from Linux, but from Windows (or OS X) it would be seen as a file in the accessible part of the card.

To edit the configuration file, see the instructions at R-Pi_ConfigurationFile.

You can get your current active settings with the following commands:

```
vcgencmd get_config <config> - lists a specific config value. E.g. vcgencmd get_config arm_freq
vcgencmd get_config int - lists all the integer config options that are set (non-zero)
vcgencmd get_config str - lists all the string config options that are set (non-null)
```

Contents

File format

The format is "property=value" where value is an integer. You may specify only one option per line. Comments may be added by starting a line with the '#' character.

Note: In the newer Raspberry Pi models there is # before every line, if you want changes to have an affect then 'uncomment' meaning remove the #.

Here is an example file

```
# Set sdtv mode to PAL (as used in Europe)
sdtv_mode=2
# Force the monitor to HDMI mode so that sound will be sent over HDMI cable
hdmi_drive=2
# Set monitor mode to DMT
hdmi_group=2
# Set monitor resolution to 1024x768 XGA 60 Hz (HDMI_DMT_XGA_60)
hdmi_mode=16
# Make display smaller to stop text spilling off the screen
```

```
overscan_left=20
overscan_right=12
overscan_top=10
overscan_bottom=10
```

And here is another example file (https://raw.github.com/Evilpaul/RPi-config/master/config.txt), containing extended documentation of features.

Memory

disable_l2cache disable ARM access to GPU's L2 cache. Needs corresponding L2 disabled kernel. Default 0

gpu_mem GPU memory in megabyte. Sets the memory split between the ARM and GPU. ARM gets the remaining memory. Min 16. Default 64

gpu_mem_256 GPU memory in megabyte for the 256 MB Raspberry Pi. Ignored by the 512 MB RP. Overrides gpu_mem. Max 192. Default not set

gpu_mem_512 GPU memory in megabyte for the 512 MB Raspberry Pi. Ignored by the 256 MB RP. Overrides gpu_mem. Max 448. Default not set

disable_pvt Disable adjusting the refresh rate of RAM every 500 ms (measuring RAM temperature).

CMA - dynamic memory split

The firmware and kernel as of 19. November 2012 supports CMA, which means the memory split between ARM and GPU is managed dynamically at runtime. You can find an example config.txt here (http://www.raspberrypi.org/phpBB3/viewtopic.php?p=223549#p223549).

cma_lwm When GPU has less than cma_lwm (low water mark) memory available it will request some from ARM.

cma_hwm When GPU has more than cma_hwm (high water mark) memory available it will release some to ARM.

The following options need to be in cmdline.txt for CMA to work:

```
coherent_pool=6M smsc95xx.turbo_mode=N
```

Note: As per https://github.com/raspberrypi/linux/issues/503 popcornmix states that CMA is not officially supported.

Camera

start_x Enable the camera module.

```
start_x=1
```

disable_camera_led Turn off the red camera LED when recording video or taking a still picture

```
disable_camera_led=1
```

gpu_mem minimum GPU memory for camera use

```
gpu_mem=128
```

Network

smsc95xx.macaddr Tells the smsc95xx driver to use a custom mac address instead of using default mac address.

```
smsc95xx.macaddr=B8:AA:BC:DE:F0:12
```

Audio

disable_audio_dither Disables dithering on the PWM audio algorithm. Try this if you experience issues with white noise on the audio jack.

```
disable_audio_dither=1
```

Video

Video mode options

sdtv_mode defines the TV standard for composite output (default=0)

```
sdtv_mode=0    Normal NTSC
sdtv_mode=1    Japanese version of NTSC - no pedestal
sdtv_mode=2    Normal PAL
sdtv_mode=3    Brazilian version of PAL - 525/60 rather than 625/50, different subcarrier
```

sdtv_aspect defines the aspect ratio for composite output (default=1)

```
sdtv_aspect=1  4:3
sdtv_aspect=2  14:9
sdtv_aspect=3  16:9
```

sdtv_disable_colourburst disables colour burst on composite output. The picture will be monochrome, but possibly sharper

```
sdtv_disable_colourburst=1  colour burst is disabled
```

hdmi_safe Use "safe mode" settings to try to boot with maximum hdmi compatibility. This is the same as the combination of: hdmi_force_hotplug=1, hdmi_ignore_edid=0xa5000080, config_hdmi_boost=4, hdmi_group=2, hdmi_mode=4, disable_overscan=0, overscan_left=24, overscan_right=24, overscan_top=24, overscan_bottom=24

```
hdmi_safe=1
```

hdmi_ignore_edid Enables the ignoring of EDID/display data if your display doesn't have an accurate EDID.

```
hdmi_ignore_edid=0xa5000080
```

hdmi_edid_file when set to 1, will read the edid data from the edid.dat file instead of from the monitor.[1]

```
hdmi_edid_file=1
```

hdmi_force_edid_audio Pretends all audio formats are supported by display, allowing passthrough of DTS/AC3 even when not reported as supported.

```
hdmi_force_edid_audio=1
```

hdmi_ignore_edid_audio Pretends all audio formats are unsupported by display. This means ALSA will default to analogue.

```
hdmi_ignore_edid_audio=1
```

hdmi_force_edid_3d Pretends all CEA modes support 3D even when edid doesn't indicate support for them.

```
hdmi_force_edid_3d=1
```

avoid_edid_fuzzy_match Avoid fuzzy matching of modes described in edid. Picks the standard mode with matching resolution and closest framerate even if blanking is wrong.

```
avoid_edid_fuzzy_match=1
```

hdmi_ignore_cec_init Doesn't sent initial active source message. Avoids bringing (CEC enabled) TV out of standby and channel switch when rebooting.

```
hdmi_ignore_cec_init=1
```

hdmi_ignore_cec Pretends CEC is not supported at all by TV. No CEC functions will be supported.

```
hdmi_ignore_cec=1
```

hdmi_force_hotplug Pretends HDMI hotplug signal is asserted so it appears a HDMI display is attached

```
hdmi_force_hotplug=1 Use HDMI mode even if no HDMI monitor is detected
```

hdmi_ignore_hotplug Pretends HDMI hotplug signal is not asserted so it appears a HDMI display is not attached

```
hdmi_ignore_hotplug=1 Use composite mode even if HDMI monitor is detected
```

hdmi_pixel_encoding Force the pixel encoding mode. By default it will use the mode requested from edid so shouldn't need changing.

```
hdmi_pixel_encoding=0 default       (limited for CEA, full for DMT)
hdmi_pixel_encoding=1 RGB limited   (16-235)
hdmi_pixel_encoding=2 RGB full      ( 0-255)
hdmi_pixel_encoding=3 YCbCr limited (16-235)
hdmi_pixel_encoding=4 YCbCr full    ( 0-255)
```

hdmi_drive chooses between HDMI and DVI modes

```
hdmi_drive=1 Normal DVI mode (No sound)
hdmi_drive=2 Normal HDMI mode (Sound will be sent if supported and enabled)
```

hdmi_group defines the HDMI type

Not specifying the group, or setting to 0 will use the preferred group reported by the edid.

```
hdmi_group=1   CEA
hdmi_group=2   DMT
```

hdmi_mode defines screen resolution in CEA or DMT format (for other modes not listed here have a look at this thread (http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=24679))

```
These values are valid if hdmi_group=1 (CEA)
hdmi_mode=1    VGA
hdmi_mode=2    480p   60 Hz
hdmi_mode=3    480p   60 Hz  H
hdmi_mode=4    720p   60 Hz
hdmi_mode=5    1080i  60 Hz
hdmi_mode=6    480i   60 Hz
hdmi_mode=7    480i   60 Hz  H
hdmi_mode=8    240p   60 Hz
hdmi_mode=9    240p   60 Hz  H
hdmi_mode=10   480i   60 Hz  4x
hdmi_mode=11   480i   60 Hz  4x H
hdmi_mode=12   240p   60 Hz  4x
hdmi_mode=13   240p   60 Hz  4x H
hdmi_mode=14   480p   60 Hz  2x
hdmi_mode=15   480p   60 Hz  2x H
hdmi_mode=16   1080p  60 Hz
hdmi_mode=17   576p   50 Hz
hdmi_mode=18   576p   50 Hz  H
hdmi_mode=19   720p   50 Hz
hdmi_mode=20   1080i  50 Hz
hdmi_mode=21   576i   50 Hz
hdmi_mode=22   576i   50 Hz  H
hdmi_mode=23   288p   50 Hz
hdmi_mode=24   288p   50 Hz  H
hdmi_mode=25   576i   50 Hz  4x
hdmi_mode=26   576i   50 Hz  4x H
```

```
hdmi_mode=27   288p   50 Hz  4x
hdmi_mode=28   288p   50 Hz  4x H
hdmi_mode=29   576p   50 Hz  2x
hdmi_mode=30   576p   50 Hz  2x H
hdmi_mode=31   1080p  50 Hz
hdmi_mode=32   1080p  24 Hz
hdmi_mode=33   1080p  25 Hz
hdmi_mode=34   1080p  30 Hz
hdmi_mode=35   480p   60 Hz  4x
hdmi_mode=36   480p   60 Hz  4xH
hdmi_mode=37   576p   50 Hz  4x
hdmi_mode=38   576p   50 Hz  4x H
hdmi_mode=39   1080i  50 Hz  reduced blanking
hdmi_mode=40   1080i 100 Hz
hdmi_mode=41   720p  100 Hz
hdmi_mode=42   576p  100 Hz
hdmi_mode=43   576p  100 Hz  H
hdmi_mode=44   576i  100 Hz
hdmi_mode=45   576i  100 Hz  H
hdmi_mode=46   1080i 120 Hz
hdmi_mode=47   720p  120 Hz
hdmi_mode=48   480p  120 Hz
hdmi_mode=49   480p  120 Hz  H
hdmi_mode=50   480i  120 Hz
hdmi_mode=51   480i  120 Hz  H
hdmi_mode=52   576p  200 Hz
hdmi_mode=53   576p  200 Hz  H
hdmi_mode=54   576i  200 Hz
hdmi_mode=55   576i  200 Hz  H
hdmi_mode=56   480p  240 Hz
hdmi_mode=57   480p  240 Hz  H
hdmi_mode=58   480i  240 Hz
hdmi_mode=59   480i  240 Hz  H
H means 16:9 variant (of a normally 4:3 mode).
2x means pixel doubled (that is, higher clock rate, with each pixel repeated twice)
4x means pixel quadrupled (that is, higher clock rate, with each pixel repeated four times)
```

```
These values are valid if hdmi_group=2 (DMT)
Note: according to http://www.raspberrypi.org/phpBB3/viewtopic.php?f=26&t=20155&p=195417&hilit=2560x1600#p195443
there is a pixel clock limit which means the highest supported mode is 1920x1200 @60  Hz with reduced blanking.
hdmi_mode=1    640x350    85 Hz
hdmi_mode=2    640x400    85 Hz
hdmi_mode=3    720x400    85 Hz
hdmi_mode=4    640x480    60 Hz
hdmi_mode=5    640x480    72 Hz
hdmi_mode=6    640x480    75 Hz
hdmi_mode=7    640x480    85 Hz
```

```
hdmi_mode=8     800x600     56 Hz
hdmi_mode=9     800x600     60 Hz
hdmi_mode=10    800x600     72 Hz
hdmi_mode=11    800x600     75 Hz
hdmi_mode=12    800x600     85 Hz
hdmi_mode=13    800x600    120 Hz
hdmi_mode=14    848x480     60 Hz
hdmi_mode=15    1024x768    43 Hz  DO NOT USE
hdmi_mode=16    1024x768    60 Hz
hdmi_mode=17    1024x768    70 Hz
hdmi_mode=18    1024x768    75 Hz
hdmi_mode=19    1024x768    85 Hz
hdmi_mode=20    1024x768   120 Hz
hdmi_mode=21    1152x864    75 Hz
hdmi_mode=22    1280x768        Reduced blanking
hdmi_mode=23    1280x768    60 Hz
hdmi_mode=24    1280x768    75 Hz
hdmi_mode=25    1280x768    85 Hz
hdmi_mode=26    1280x768   120 Hz  Reduced blanking
hdmi_mode=27    1280x800        Reduced blanking
hdmi_mode=28    1280x800    60 Hz
hdmi_mode=29    1280x800    75 Hz
hdmi_mode=30    1280x800    85 Hz
hdmi_mode=31    1280x800   120 Hz  Reduced blanking
hdmi_mode=32    1280x960    60 Hz
hdmi_mode=33    1280x960    85 Hz
hdmi_mode=34    1280x960   120 Hz  Reduced blanking
hdmi_mode=35    1280x1024   60 Hz
hdmi_mode=36    1280x1024   75 Hz
hdmi_mode=37    1280x1024   85 Hz
hdmi_mode=38    1280x1024  120 Hz  Reduced blanking
hdmi_mode=39    1360x768    60 Hz
hdmi_mode=40    1360x768   120 Hz  Reduced blanking
hdmi_mode=41    1400x1050       Reduced blanking
hdmi_mode=42    1400x1050   60 Hz
hdmi_mode=43    1400x1050   75 Hz
hdmi_mode=44    1400x1050   85 Hz
hdmi_mode=45    1400x1050  120 Hz  Reduced blanking
hdmi_mode=46    1440x900        Reduced blanking
hdmi_mode=47    1440x900    60 Hz
hdmi_mode=48    1440x900    75 Hz
hdmi_mode=49    1440x900    85 Hz
hdmi_mode=50    1440x900   120 Hz  Reduced blanking
hdmi_mode=51    1600x1200   60 Hz
hdmi_mode=52    1600x1200   65 Hz
hdmi_mode=53    1600x1200   70 Hz
hdmi_mode=54    1600x1200   75 Hz
hdmi_mode=55    1600x1200   85 Hz
hdmi_mode=56    1600x1200  120 Hz  Reduced blanking
```

```
hdmi_mode=57   1680x1050         Reduced blanking
hdmi_mode=58   1680x1050  60 Hz
hdmi_mode=59   1680x1050  75 Hz
hdmi_mode=60   1680x1050  85 Hz
hdmi_mode=61   1680x1050 120 Hz  Reduced blanking
hdmi_mode=62   1792x1344  60 Hz
hdmi_mode=63   1792x1344  75 Hz
hdmi_mode=64   1792x1344 120 Hz  Reduced blanking
hdmi_mode=65   1856x1392  60 Hz
hdmi_mode=66   1856x1392  75 Hz
hdmi_mode=67   1856x1392 120 Hz  Reduced blanking
hdmi_mode=68   1920x1200         Reduced blanking
hdmi_mode=69   1920x1200  60 Hz
hdmi_mode=70   1920x1200  75 Hz
hdmi_mode=71   1920x1200  85 Hz
hdmi_mode=72   1920x1200 120 Hz  Reduced blanking
hdmi_mode=73   1920x1440  60 Hz
hdmi_mode=74   1920x1440  75 Hz
hdmi_mode=75   1920x1440 120 Hz  Reduced blanking
hdmi_mode=76   2560x1600         Reduced blanking
hdmi_mode=77   2560x1600  60 Hz
hdmi_mode=78   2560x1600  75 Hz
hdmi_mode=79   2560x1600  85 Hz
hdmi_mode=80   2560x1600 120 Hz  Reduced blanking
hdmi_mode=81   1366x768   60 Hz
hdmi_mode=82   1080p      60 Hz
hdmi_mode=83   1600x900          Reduced blanking
hdmi_mode=84   2048x1152         Reduced blanking
hdmi_mode=85   720p       60 Hz
hdmi_mode=86   1366x768          Reduced blanking
```

overscan_left number of pixels to skip on left

overscan_right number of pixels to skip on right

overscan_top number of pixels to skip on top

overscan_bottom number of pixels to skip on bottom

framebuffer_width console framebuffer width in pixels. Default is display width minus overscan.

framebuffer_height console framebuffer height in pixels. Default is display height minus overscan.

framebuffer_depth console framebuffer depth in bits per pixel. Default is 16. 8 bit is valid, but default RGB palette makes an unreadable screen. 24 bit looks better but has corruption issues as of 20120615. 32 bit has no corruption issues but needs framebuffer_ignore_alpha=1 and shows the wrong colors as of 20120615.

framebuffer_ignore_alpha set to 1 to disable alpha channel. Helps with 32 bit.

test_mode enable test sound/image during boot for manufacturing test.

disable_overscan set to 1 to disable overscan.

config_hdmi_boost configure the signal strength of the HDMI interface. Default is 2 on Pi 1 model B and 5 on later boards. Try increasing if you have interference issues with hdmi (e.g. to 7) 11 is the maximum.

display_rotate rotates the display clockwise on the screen (default=0) or flips the display.

```
display_rotate=0        Normal
display_rotate=1         90 degrees
display_rotate=2        180 degrees
display_rotate=3        270 degrees
display_rotate=0x10000  horizontal flip
display_rotate=0x20000  vertical flip
```

Note: the 90 and 270 degrees rotation options require additional memory on GPU, so won't work with the 16M GPU split. Probably the reason for:

- Crashes my Raspberry Pi before Linux boots if set to "1" -- REW 20120913.

overscan_scale is mentioned in [1] (https://github.com/raspberrypi/firmware/commit/59059dea7a8e87489e0f9bc844fd6f54eb96f25d), [2] (http://www.raspberrypi.org/forums/viewtopic.php?f=67&t=15700&p=161893#p161893) and [3] (http://forum.openframeworks.cc/t/reported-display-display-size-with-320x240-tft-on-raspberry-pi/15144/11) as a feature in new firmware (Aug 2012 and later).

- overscan_scale=1 caused the output of openFrameworks to respect the overscan_left, overscan_right, top and bottom settings with the use of an LCD display connected via composite.

Which values are valid for my monitor?

Your HDMI monitor may support only a limited set of formats. To find out which formats are supported, use the following method.

- Set the output format to VGA 60 Hz (hdmi_group=1 hdmi_mode=1) and boot up the Raspberry Pi
- Enter the following command to give a list of CEA supported modes

```
/opt/vc/bin/tvservice -m CEA
```

- Enter the following command to give a list of DMT supported modes

```
/opt/vc/bin/tvservice -m DMT
```

- Enter the following command to show your current state

```
/opt/vc/bin/tvservice -s
```

- Enter the following commands to dump more detailed information from your monitor

```
/opt/vc/bin/tvservice -d edid.dat
/opt/vc/bin/edidparser edid.dat
```

The edid.dat should also be provided when troubleshooting problems with the default HDMI mode

Licensed Codecs

Hardware decoding of additional codecs can be enabled by purchasing a license (http://www.raspberrypi.com/) that is locked to the CPU serial number of your Raspberry Pi.

You need to have at least assigned 32 megabytes to the GPU to enable the codecs.

decode_MPG2 License key to allow hardware MPEG-2 decoding.

```
decode_MPG2=0x12345678
```

decode_WVC1 License key to allow hardware VC-1 decoding.

```
decode_WVC1=0x12345678
```

License setup for SD card sharing between multiple Raspberry Pis. Maximum of 8 licenses at once.

```
decode_XXXX=0x12345678,0xabcdabcd,0x87654321,...
```

Boot

disable_commandline_tags stop start.elf from filling in ATAGS (memory from 0x100) before launching kernel

cmdline (string) command line parameters. Can be used instead of cmdline.txt file

kernel (string) alternative name to use when loading kernel. Default "kernel.img"

kernel_address address to load kernel.img file at

kernel_old (bool) if 1, load kernel at 0x0

ramfsfile (string) ramfs file to load

ramfsaddr address to load ramfs file at

initramfs (string address) ramfs file and address to load it at (it's like ramfsfile+ramfsaddr in one option). NOTE: this option uses different syntax than all other options - you should not use "=" character here. Example:

```
initramfs initramf.gz 0x00800000
```

The valid addresses depend on the kernel size. 0x00800000 works for 3.6-trunk-rpi, 0x00a00000 works for 3.14-1-rpi.

device_tree_address address to load device_tree at

init_uart_baud initial UART baud rate. Default 115200

init_uart_clock initial UART clock. Default 3000000 (3 MHz)

init_emmc_clock initial eMMC clock. Default 100000000 (100 MHz)

boot_delay wait for given number of seconds in start.elf before loading kernel. delay = 1000 * boot_delay + boot_delay_ms. Default 1

boot_delay_ms wait for given number of milliseconds in start.elf before loading kernel. Default 0

avoid_safe_mode if set to 1, safe_mode boot won't be enabled. Default 0

disable_splash if set to 1, avoids the rainbow splash screen on boot

Overclocking

NOTE: Setting parameters other than that available by 'raspi-config' will set a permanent bit within the SoC, making it possibly to detect that you Raspberry Pi has been overclocked. This was meant to void warranty if the device has been overclocked. Since 19th of September 2012 you can overclock your Raspberry Pi without affecting your warranty[2]

The latest kernel has a cpufreq (http://www.pantz.org/software/cpufreq/usingcpufreqonlinux.html) kernel driver with the "ondemand" governor enabled by default. It has no effect if you have no overclock settings. But when you do, the arm frequency will vary with processor load. Non default values are only used when needed according to the used governor. You can adjust the minimum values with the *_min config options or disable dynamic clocking with force_turbo=1. [3]

Overclock and overvoltage will be disabled at runtime when the SoC reaches 85 °C to cool it down. You should not hit the limit, even with maximum settings at 25 °C ambient temperature. [4]

Overclocking options

Option	Description
arm_freq	Frequency of ARM in MHz. Default 700
gpu_freq	Sets core_freq, h264_freq, isp_freq, v3d_freq together. Default 250
core_freq	Frequency of GPU processor core in MHz. For models prior to the Pi2, this has an impact on ARM performance since it drives the L2 cache. The ARM on the Pi2 has a separate L2 cache. Default 250
h264_freq	Frequency of hardware video block in MHz. Default 250
isp_freq	Frequency of image sensor pipeline block in MHz. Default 250
v3d_freq	Frequency of 3D block in MHz. Default 250
avoid_pwm_pll	Don't dedicate a pll to PWM audio. This will reduce analogue audio quality slightly. The spare PLL allows the core_freq to be set independently from the rest of the gpu allowing more control over overclocking. Default 0
sdram_freq	Frequency of SDRAM in MHz. Default 400
over_voltage	ARM/GPU core voltage adjust. [-16,8] equates to [0.8 V,1.4 V] with 0.025 V steps. [5] Default is 0 (1.2 V). Values above 6 are only allowed when force_turbo or current_limit_override are specified (which set the warranty bit).
over_voltage_sdram	Sets over_voltage_sdram_c, over_voltage_sdram_i, over_voltage_sdram_p together
over_voltage_sdram_c	SDRAM controller voltage adjust. [-16,8] equates to [0.8 V,1.4 V] with 0.025 V steps. Default 0 (1.2 V) [5]
over_voltage_sdram_i	SDRAM I/O voltage adjust. [-16,8] equates to [0.8 V,1.4 V] with 0.025 V steps. Default 0 (1.2 V)[5]
over_voltage_sdram_p	SDRAM phy voltage adjust. [-16,8] equates to [0.8 V,1.4 V] with 0.025 V steps. Default 0 (1.2 V)[5]

force_turbo	Disables dynamic cpufreq driver and minimum settings below. Enables H.264/V3D/ISP overclock options. Default 0. May set warranty bit.
initial_turbo	Enables turbo mode from boot for the given value in seconds (up to 60) or until cpufreq sets a frequency. Default 0 [6]
arm_freq_min	Minimum value of arm_freq used for dynamic clocking. Default 700
core_freq_min	Minimum value of core_freq used for dynamic clocking. Default 250
sdram_freq_min	Minimum value of sdram_freq used for dynamic clocking. Default 400
over_voltage_min	Minimum value of over_voltage used for dynamic clocking. Default 0
temp_limit	Overheat protection. Sets clocks and voltages to default when the SoC reaches this Celsius value. Setting this higher than default voids warranty. Default 85
current_limit_override	This is enabled by default.[7]

force_turbo mode

```
force_turbo=0
```

enables dynamic clocks and voltage for the ARM core, GPU core and SDRAM. When busy, ARM frequency go up to "arm_freq" and down to "arm_freq_min" on idle. "core_freq", "sdram_freq" and "over_voltage" behave the same. "over_voltage" is limited to 6 (1.35 V). Non default values for the H.264/V3D/ISP parts are ignored.

```
force_turbo=1
```

disables dynamic clocking, so all frequencies and voltages stay high. Overclocking of H.264/V3D/ISP GPU parts is allowed as well as setting "over_voltage" to 8 (1.4 V). [8]

Clocks relationship

The GPU core, H.264, V3D and ISP share a PLL, therefore need to have related frequencies. ARM, SDRAM and GPU each have their own PLLs and can have unrelated frequencies.[9]

The following is not necessary with "avoid_pwm_pll=1".

```
pll_freq = floor(2400 / (2 * core_freq)) * (2 * core_freq)
gpu_freq = pll_freq / [even number]
```

The effective gpu_freq is automatically rounded to nearest even integer, so asking for core_freq=500 and gpu_freq=300 will result in divisor of 2000/300 = 6.666 => 6 and so 333.33 MHz.

Tested values

The following table shows some successful attempts at overclocking, which can be used for orientation. These settings may not work on every device and can shorten the lifetime of the Broadcom SoC.

arm_freq	gpu_freq	core_freq	h264_freq	isp_freq	v3d_freq	sdram_freq	over_voltage	over_voltage_sdram
800								
900	275					500		
900		450				450		
930	350					500		
1000		500				500	6	
1050							6	
1150		500				600	8	

There are reports (http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=6201&p=159188&hilit=hynix#p159160) that Hynix RAM is not as good as Samsung RAM for overclocking.

SD card usage with overclocking

SD card setup redirect: http://elinux.org/RPi_Easy_SD_Card_Setup

Stability of SD card operations when using overclocking is independent of:

- Filesystem type, ext4, NTFS or other.
- SD card vendor.
- The Raspberry Pi model.
- SD card size - verified for 16 GB and up.

What does matter is when you under-power your Raspberry Pi (that is, less than the Raspberry Pi base setup specifications!).

There initially was an increased likelihood of SD card corruption when using overclocking. This is no longer an issue (with firmware from Nov 11 2013 or later).

Monitoring temperature and voltage

To monitor the Raspberry Pi's temperature, look at: /sys/class/thermal/thermal_zone0/temp
To monitor the Raspberry Pi's current frequency, look at:
/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
To monitor the Raspberry Pi's PSU voltage, you'll need a multimeter, across the power supply test points, or the expansion header.

It's generally a good idea to keep the core temp below 70 degrees, and the voltage above 4.8 V. (Note that some, not necessarily cheap, USB power supplies fall as low as 4.2 V, this is because they are usually designed to charge a 3.7 V lithium polymer battery, rather than to supply a solid 5 V to a computer). Also, a heatsink can be helpful, especially if the Raspberry Pi is to be run inside a case. A suitable heatsink is the self-adhesive BGA (ball-grid-array) 14x14x10 mm heatsink, part 674-4756 from RS Components.

Under-voltage warnings

The B+ has an under voltage detect trigger which results in the power led going off when voltage drops below about 4.65V. The signal is also available on a gpio line (GPIO35).

Latest firmware update will display a warning symbol in top right of display when this is detected. It will also disable turbo mode while the warning is displayed to try to reduce the chances of crashing.

There is also a warning symbol to the existing over-temperature condition (> 85'C), which also disabled turbo mode.

Currently the symbols are:

```
Red square: over-temperature
Rainbow square: under-voltage
```

As usual you can override the behaviour in config.txt if you understand the risks:

```
avoid_warnings=1 removes the warning overlay.
avoid_warnings=2 additionally allows turbo when low-voltage is present.
```

The above adapted from

https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=82373

Overclock stability test

Most overclocking issues show up right away with a failure to boot, but it is possible to get filesystem corruption that arises over time. Here is a script to stress-test the stability of the system, specifically the SD card. If this script runs to completion, without any errors showing in dmesg, then the Raspberry Pi is probably stable with these settings.

If the system does crash, then hold down the shift key during boot, which will temporarily disable all overclocking. Also, note that SD card issues are usually affected by the core_freq, rather than the arm_freq, and that there is a surprisingly big jump in this (from 250 MHz to 500 MHz) between the High speed (950 MHz) and Turbo (1 GHz) presets in raspi-config.

```
#!/bin/bash
#Simple stress test for system. If it survives this, it's probably stable.
#Free software, GPL2+

echo "Testing overclock stability..."

#Max out all CPU cores. Heats it up, loads the power-supply.
```

```
for ((i=0; i<$(nproc --all); i++)); do nice yes >/dev/null & done

#Read the entire SD card 10x. Tests RAM and I/O
for i in `seq 1 10`; do echo reading: $i; sudo dd if=/dev/mmcblk0 of=/dev/null bs=4M; done

#Writes 512 MB test file, 10x.
for i in `seq 1 10`; do echo writing: $i; dd if=/dev/zero of=deleteme.dat bs=1M count=512; sync; done

#Clean up
killall yes
rm deleteme.dat

#Print summary. Anything nasty will appear in dmesg.
echo -n "CPU freq: "  ; cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
echo -n "CPU temp: "  ; cat /sys/class/thermal/thermal_zone0/temp
dmesg | tail

echo "Not crashed yet, probably stable."
```

USB Power

max_usb_current

Together with the raspberry PI B+ a new config.txt setting was introduced.

```
max_usb_current=1
```

when adding this line the USB power manager will change its output current limit (for all 4 USB ports combined) from 600mA to double that, 1200mA.

```
max_usb_current=0
```

is the default, and limits the USB current to 600mA.

Make sure your power supply (and microUSB cord) are up to the task of delivering the extra 0.6A current.

Device Tree

```
dtparam=i2c_arm=on
```

- Enables I2C on GPIO pins.

```
dtparam=i2s=on
```

- Enables I2S audio hardware.

```
dtparam=spi=on
```

- Enables SPI driver.

```
dtoverlay=xxx
```

- Adds an overlay /boot/overlays/xxx-overlay.dtb to the device tree.

There's lots more documentation of available overlays in /boot/config/README. See source on Github (https://github.com/raspberrypi/linux/blob/rpi-4.1.y/arch/arm/boot/dts/overlays/README) for more.

See also

RPi_cmdline.txt

References

1. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?p=173430#p173430

2. ↑ Introducing turbo mode: up to 50% more performance for free (http://www.raspberrypi.org/archives/2008)
3. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?p=169726#p169726
4. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=11579#p169872
5. ↑ 5.0 5.1 5.2 5.3 What this means is that you can specify -16 and get 0.8 V as the GPU/core voltage, and specify 8 and get 1.4 V
6. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=6201&start=425#p180099
7. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=6201&start=325#p170793
8. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=6201&sid=852d546291ae711ffcd8bf23d3214581&start=325#p170793
9. ↑ http://www.raspberrypi.org/phpBB3/viewtopic.php?f=29&t=6201&start=275#p168042

Retrieved from "http://elinux.org/index.php?title=RPiconfig&oldid=403291"

Category: RaspberryPi