Pin configuration

By using a normal Raspberry Pi operating system image, such as Raspbian, all pins of the processor are not configured. The device is operational, but all features are not enabled.

By default the audio output is not enabled in Compute Module. To get the 3.5 mm audio jack working, you need to modify the pin configuration of the module by providing a custom device tree blob.

In addition to audio, the camera connector, LAN9512 reset, activity LED, and HDMI hot plug are not configured and enabled by default. These pins can also be re-configured using the procedure below. Note that the LAN9512 chip and HDMI are operational even without the reset and hot plug pin configurations.

You can find a sample device tree source file dt-blob.dts from the internet address www.hasseb.fi/pi. Using the sample dts-file, the audio output, camera connector, LAN9512 reset, activity LED, and HDMI hot plug pins are all enabled.

Providing a custom device tree blob

In order to compile a device tree source (dts) file into a device tree blob (dtb) file, the device tree compiler must be installed by running:

```
sudo apt-get install device-tree-compiler
```

The dtc command can then be used as follows:

```
sudo dtc -I dts -O dtb -o /boot/dt-blob.bin dt-blob.dts
```

Restart is required after pin configuration.

<u>Running the commands above is enough for most users.</u>

NOTE: In the case of NOOBS installs, the dtb file should be placed on the recovery partition instead.

Similarly, a dtb file can be converted back to a dts file, if required.

```
dtc -I dtb -O dts -o dt-blob.dts /boot/dt-blob.bin
```

Sections of the dt-blob

The dt-blob.bin is used to configure the binary blob (Videocore) at boot time. It is not something that the linux kernel uses (at the moment) although a kernel section will be added at a later stage when we move the Raspberry Pi kernel to use a dt-blob for configuration. The dt-blob is capable of configuring each of the different versions of the Raspberry Pi including the Compute Module to set up the alternate settings correctly. The following sections are valid in the dt-blob.

videocore

This section contains the whole videocore blob information, all subsequent sections must be enclosed within this section.

PINS_*

There are up to four separate pins_* sections these are:

1. **pins_rev1** Rev1 pin setup. There are some difference because of the moved I2C pins.
2. **pins_rev2** Rev2 pin setup. This includes the additional codec pins on P5.
3. **pins_bplus** B+ revision including the full 40pin connector.
4. **pins_cm** The Compute Module, note the default for this is the default for the chip so can be a useful source of information about default pullups / downs on the chip.

Each `pins_*` section can contain `pin_config` and `pin_defines` sections.

PIN_CONFIG

The `pin_config` section is used to configure the individual pins, each item in this section must be a named pin section, such as `pin@p32` meaning GPIO32. There is a special section `pin@default` which has the default settings for anything not specifically named in the pin_config section.

PIN@PINNAME

This section can contain any combination of the following items

1. `polarity`
    - `active_high`
    - `active_low`
2. `termination`
    - `pull_up`
    - `pull_down`
    - `no_pulling`
3. `startup_state`
    - `active`
    - `inactive`
4. `function`
    - `input`
    - `output`
    - `sdcard`
    - `i2c`
    - `i2c1`
    - `spi`
    - `spi1`
    - `spi2`
    - `smi`
    - `dpi`
    - `pcm`
    - `pwm`
    - `uart0`
    - `uart1`
    - `gp_clk`
    - `emmc`
    - `arm_jtag`

5. `drive_strength_ma`
   The drive strength is used to set a strength for the pins, please note you can only set the bank to a single drive strength. <8> <16> are valid values.
6. `pin_defines`
   This section is used to set specific videocore functionality to particular pins, this enables the user to move the camera power enable pin to somewhere different or the hdmi hotplug postion (i.e. things that linux have no control over). Please refer to the sample dts file.

Clock configuration

It is possible to change the configuration of the clocks through this interface, although very difficult to predict the results! The configuration of the clocking system is very very complex, there are five separate PLLs each one has its own fixed (or variable in the case of PLLC) VCO frequency. Each VCO then has a number of different channels which can be set up with a different division of the VCO frequency. Then each of the clock destinations can then be configured to come from one of the clock channels (although there is a restricted mapping of source to destination so not all channels can be routed to all clock destinations).

For this reason I'll just add here a couple of example configurations that you can use to alter very specific clocks. Beyond this it is something we'll add to when requests for clock configurations are made.

```
clock_routing {
   vco@PLLA  {    freq = <1966080000>; };
   chan@APER {    div  = <4>; };
   clock@GPCLK0 { pll = "PLLA"; chan = "APER"; };
};
clock_setup {
   clock@PWM { freq = <2400000>; };
   clock@GPCLK0 { freq = <12288000>; };
   clock@GPCLK1 { freq = <25000000>; };
};
```

The above will set the PLLA to a source VCO running at 1.96608GHz (the limits for this VCO are 600MHz - 2.4GHz), the APER channel to /4 and configures GPCLK0 to be sourced from PLLA through APER. This is used specifically to give an audio codec the 12288000Hz it needs to do the 48000 range of frequencies.