GPIO Interfacing and Device Tree Overlays on Upgraded Beaglebone Image

Last updated August 2018, for CMPT433

Rationale:

When you re-flash your Beaglebone, it's likely that you will flash a newer image than the Beaglebone shipped with. In the CMPT433 Summer 2018 term, our Beaglebones shipped with the 4.4.x kernel image. During the semester I re-flashed my Beaglebone to a 4.14.49-ti-r54 image for PRU work and found that the handling of GPIO and device tree overlays changed significantly. This guide should help you to reconfigure your newly flashed Beaglebone so that you can still work with the GPIO pins and load necessary device tree overlays.

Re-Flashing:

If you haven't already re-flashed your Beaglebone, there are many ways to do so which won't be documented here. I used a USB cable flasher utility written by another student (Oscar S.) in our CMPT433 class. You can find this utility at: https://github.com/Jerald/cmpt433finalproject/blob/master/BeagleBone_USB_Flash_Guide.md.

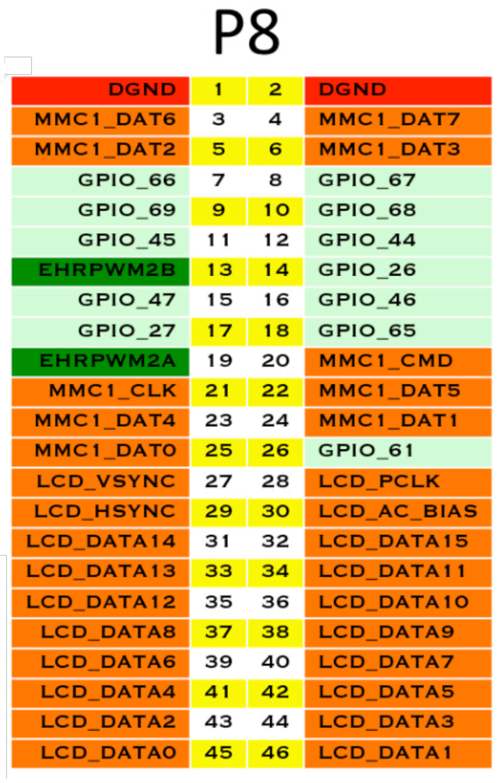
We spoke with him directly about this flasher utility, and he recommended using it with a physical Linux operating system for best results; apparently a virtual Linux operating system has possible complications. It worked for us with a physical Linux operating system so for the purposes of this guide, we recommend that method if you need to re-flash your board.

Device Trees:

On the 4.4.x kernel image, it is possible to use the following command to see which virtual capes are loaded:

```
cat /sys/devices/platform/bone_capemgr/slots
```

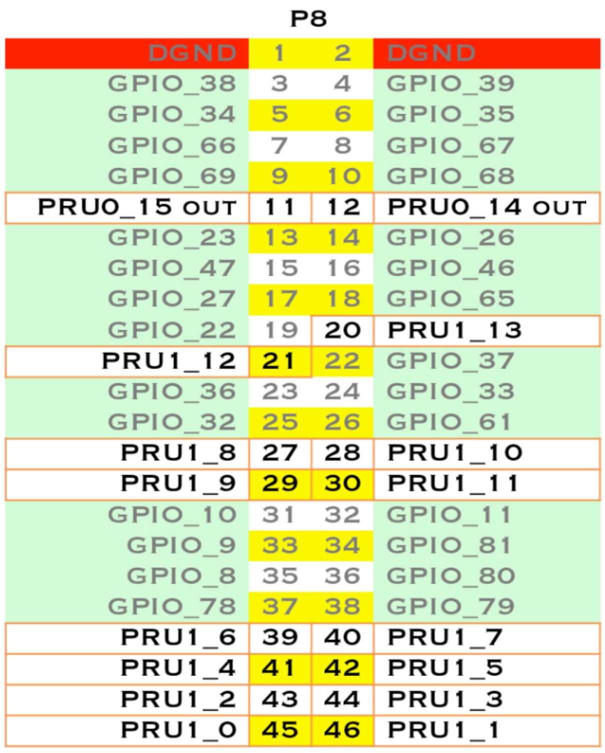
On a newer 4.14.x kernel image, the loading of virtual capes happens at startup through uBoot. To change which capes are loaded on startup, you'll want to edit /boot/uEnv.txt on the Beaglebone and disable some of the preloaded capes depending on which pins you want to free up. Take a look at the P8 and P9 header schematics on http://beagleboard.org/support/bone101 to determine which overlays to disable:



P8

DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

Standard Beaglebone P8 Header



P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
PRU0_15 OUT	11	12	PRU0_14 OUT
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	PRU1_13
PRU1_12	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
PRU1_8	27	28	PRU1_10
PRU1_9	29	30	PRU1_11
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
PRU1_6	39	40	PRU1_7
PRU1_4	41	42	PRU1_5
PRU1_2	43	44	PRU1_3
PRU1_0	45	46	PRU1_1

P8 Header: Pins used by the PRU1 R30 (register) for GPIO output

For my purposes, I needed to free up at least 12 pins on the P8 header so that I could use them to relay output from PRU1. I chose P8 pins [27-30] and [39-46] because those pins did not conflict with the pins used for eMMC. eMMC is the onboard storage for the Beaglebone and unless you are booting off a microSD you are probably using eMMC to store your operating system, so don't disable the eMMC uBoot overlay. The pins labelled LCD_XXXX are used by a virtual video cape, and must be freed by disabling the video overlay virtual capes before they can be used for other purposes like PRU1 output.

In your /boot/uEnv.txt file, change the following lines:

```
###Disable auto loading of virtual capes (emmc/video/wireless/adc)
#disable_uboot_overlay_emmc=1
#disable_uboot_overlay_video=1
#disable_uboot_overlay_audio=1
#disable_uboot_overlay_wireless=1
#disable_uboot_overlay_adc=1
```

To the following

```
###Disable auto loading of virtual capes (emmc/video/wireless/adc)
#disable_uboot_overlay_emmc=1
disable_uboot_overlay_video=1
disable_uboot_overlay_audio=1
```

```
disable_uboot_overlay_wireless=1
disable_uboot_overlay_adc=1
```

Save the uEnv.txt file and reboot the board so uBoot can pick up the overlay changes.

You don't have to disable the wireless and ADC overlay, but I knew I wasn't going to be using them so I disabled them for my purposes. If you need the wireless or ADC overlay, leave a # in front of those lines to keep them enabled.

Configuring GPIO:

To use a GPIO pin in a mode other than the default (aka change the pinmux settings), a user on the 4.4.x kernel had to load a custom device tree overlay. In the newer kernel versions, there exists a utility called config-pin that allows a user to change the GPIO mode without modifying a device tree overlay. Use config-pin with an -l flag to get available pin modes for a given GPIO. In this example, I'll be using P8_46 because I modified the mode of that pin for my work with PRU1.

Using:

```
config-pin -l P8_46
```

returns possible modes for pin 46 on the P8 header:

```
default gpio gpio_pu gpio_pd pruout pruin pwm
```

I set P8_46 to pruout to enable it to receive PRU1 output like so:

```
config-pin P8_46 pruout
```

You can use the -i flag with config-pin to get more detailed information about the pin:

```
config-pin -i P8_46

Pin name: P8_46
Function if no cape loaded: hdmi
Function if cape loaded: default gpio gpio_pu gpio_pd pruout
pruin pwm
Function information: lcd_data1 default gpio2_7 gpio2_7 gpio2_7
pr1_pru1_pru_r30_1 pr1_pru1_pru_r31_1 ehrpwm2B
Cape: cape-universala cape-univ-hdmi
Kernel GPIO id: 71
PRU GPIO id: 103
```