# 02-3 Device Trees

A systematic way to describe hardware

Much taken from:
http://learn.adafruit.com/introduction-to-the-beaglebone-black-device-tree

## Device Tree - Overview

- A way to describe hardware in a system
- Example: how the UART interfaces with the system
  - which pins
  - how they should be muxed
  - the device to enable
  - which driver to use

## History

- Under the 3.2 kernel
  - Huge influx of ARM systems in the past few years
  - ARM board files described how each board worked
  - a lot of confusion and conflicts in the Linux kernel surrounding the ARM components
- Under the 3.8 kernel
  - Any new ARM boards use the flattened device tree

## Device Tree and Overlays

- The device tree is a file (or files) that describe at boot time all the hardware
- **Problem**: Embedded systems often add hardware at run time (i.e. capes)
- **Solution**: Device Tree Overlays and cape manager

## gpio Example Overlay

- See Handout
- Example is a tree structure of nodes and properties (http://devicetree.org/Device_Tree_Usage)
- Start with

```
/*
* Copyright (C) 2012 Texas Instruments Incorporated - http://www.ti.com/
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Purpose License Version 2 as
* published by the Free Software Foundation
*
* Original from: github.com/jadonk/validation-scripts/blob/master/test-capemgr/
*
* Modified by Derek Molloy for the example on www.derekmolloy.ie
* that maps GPIO pins for the example
* From: https://github.com/derekmolloy/boneDeviceTree/tree/master/overlay
*/
```

## Walk Through – 1

```
/dts-v1/;
/plugin/;
```

- Version and plugin

```
/{
```

- Root node

```
compatible = "ti,beaglebone", "ti,beaglebone-black";
```

- describes which platforms the DT overlay works with
- most compatible -> least compatible
- Name all the platforms that you'd like to support, as it will fail to load in any platforms not mentioned.

## Walk Through – 2

```
part-number = "DM-GPIO-Test";
    version = "00A0";
```

- Part number and version are further guards to ensure that the proper DT overlays are loaded.
- Also used for the name of the .dts file in the form of

  <part-no>-<rev>.dts
- Also, as far as I can tell, the revision must be 00A0 on the BeagleBone Black.

## Walk Through – 3

- **Not in your example**

```
/* state the resources this cape uses */
exclusive-use =
        /* the pin header uses */
        "P9.24", /* uart1_txd */
        "P9.26", /* uart1_rxd */
        /* the hardware ip uses */
        "uart1";
```

- exclusive-use property allows overlays to describe what resources they need, and prevents any other overlays from using those resources.

## Fragments

Target to overlay

Register

Value

- Describe which target to overlay
- Use to customize either mux pins or enable devices

```
fragment@0 {
    target = <&am33xx_pinmux>;
    __overlay__ {
        pinctrl_test: DM_GPIO_Test_Pins {
            pinctrl-single,pins = <
                0x078 0x07  /* P9_12 60 OUTPUT MODE7 - The LED Output */
                0x184 0x2f  /* P9_24 15 INPUT  MODE7 none - The Button Input */
                0x034 0x37  /* P8_11 45 INPUT  MODE7 pullup    - Yellow Wire */
                0x030 0x27  /* P8_12 44 INPUT  MODE7 pulldown  - Green Wire  */
                0x024 0x2f  /* P8_13 23 INPUT  MODE7 none      - White Wire  */
       /* OUTPUT  GPIO(mode7) 0x07 pulldown, 0x17 pullup, 0x?f no pullup/down */
       /* INPUT   GPIO(mode7) 0x27 pulldown, 0x37 pullup, 0x?f no pullup/down */
          >;
        };
    };
};
```

https://www.kernel.org/doc/Documentation/devicetree/bindings/pinctrl/pinctrl-single.txt

## Register values

- Use the Molloy P8/P9 tables to find the register values, or

```
beagle$ exercises/node.js/findGPIO.js P9_12
{ name: 'GPIO1_28',
  gpio: 60,
  mux: 'gpmc_ben1',
  eeprom: 36,
  key: 'P9_12',
  muxRegOffset: '0x078',
  options:
   [ 'gpmc_ben1',
     'mii2_col',
     'NA',
     'mmc2_dat3',
     'NA',
     'NA',
     'mcasp0_aclkr',
     'gpio1_28' ] }
```

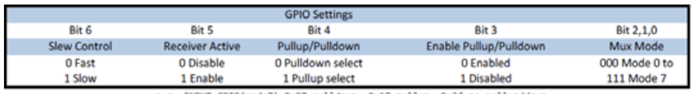
## Register values

- Or

```
beagle$ ./findGPIO.js 7
Looking for gpio 7
{ name: 'GPIO0_7',
  gpio: 7,
  mux: 'ecap0_in_pwm0_out',
  eeprom: 4,
  pwm: { muxmode: 0, path: 'ecap.0', name: 'ECAPPWM0' },
  key: 'P9_42',
  muxRegOffset: '0x164',
  options:
   [ 'ecap0_in_pwm0_out',
     'uart3_txd',
     'spi1_cs1',
     'pr1_ecap0_ecap_capin_apwm_o',
     'spi1_sclk',
     'mmc0_sdwp',
     'xdma_event_intr2',
     'gpio0_7' ] }
```

## Register contents

- From the Molloy table

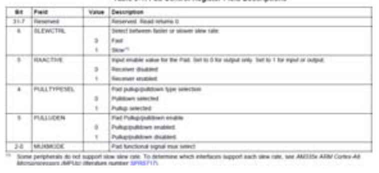


| GPIO Settings | | | | |
|---|---|---|---|---|
| Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2,1,0 |
| Slew Control | Receiver Active | Pullup/Pulldown | Enable Pullup/Pulldown | Mux Mode |
| 0 Fast | 0 Disable | 0 Pulldown select | 0 Enabled | 000 Mode 0 to |
| 1 Slow | 1 Enable | 1 Pullup select | 1 Disabled | 111 Mode 7 |

```
e.g. INPUT GPIO(mode7) 0x07 pulldown, 0x17 pullup, 0x?f no pullup/down
e.g. OUTPUT GPIO(mode7) 0x27 pulldown, 0x37 pullup, 0x?f no pullup/down
```

- Or from the Technical Reference Manual
- Section 9.2.2, page 747 (of 4161!)

Table 9-1. Pad Control Register Field Descriptions



| Bit | Field | Value | Description |
|---|---|---|---|
| 31-7 | Reserved | | Reserved. Read returns 0. |
| 6 | SLEWCTRL | | Select between faster or slower slew rate. |
| | | 0 | Fast |
| | | 1 | Slow[1] |
| 5 | RXACTIVE | | Input enable value for the Pad. Set to 0 for output only. Set to 1 for input or output. |
| | | 0 | Receiver disabled |
| | | 1 | Receiver enabled |
| 4 | PULLTYPESEL | | Pad pullup/pulldown type selection |
| | | 0 | Pulldown selected |
| | | 1 | Pullup selected |
| 3 | PULLUDEN | | Pad Pullup/pulldown enable |
| | | 0 | Pullup/pulldown enabled |
| | | 1 | Pullup/pulldown disabled |
| 2-0 | MUXMODE | | Pad functional signal mux select |

[1] Some peripherals do not support slow slew rate. To determine which interfaces support each slew rate, see AM335x ARM Cortex-A8 Microprocessors (MPUs) (literature number SPRS717).

## Fragments – 2

- Enables gpio

```
fragment@1 {
    target = <&ocp>;
    __overlay__ {
        test_helper: helper {
            compatible = "bone-pinmux-helper";
            pinctrl-names = "default";
            pinctrl-0 = <&pinctrl_test>;
            status = "okay";
        };
    };
};
```

## Firmware

- DT Overlays live in /lib/firmware

compiled

Source

```
beagle$ ls /lib/firmware
3com                        bone_pwm_P8_46-00A0.dtbo
BB-ADC-00A0.dtbo            bone_pwm_P8_46-00A0.dts
BB-ADC-00A0.dts             bone_pwm_P9_14-00A0.dtbo
BB-BONE-AUDI-01-00A0.dtbo   bone_pwm_P9_14-00A0.dts
BB-BONE-AUDI-01-00A0.dts    bone_pwm_P9_16-00A0.dtbo
BB-BONE-BACON-00A0.dtbo     bone_pwm_P9_16-00A0.dts
…
BB-BONE-PRU-01-00A0.dtbo    cape-bone-adafruit-lcd-00A0.dtbo
BB-BONE-PRU-01-00A0.dts     cape-bone-adafruit-lcd-00A0.dts
…
BB-I2C1-00A0.dtbo           cape-bone-nixie-00A0.dts
BB-I2C1-00A0.dts            cape-bone-pinmux-test-00A0.dtbo
…
BB-SPI0-00A0.dtbo           cape-bone-tester-00A0.dts
BB-SPI0-00A0.dts            cape-bone-weather-00A0.dtbo
BB-SPI1-00A0.dtbo           cape-bone-weather-00A0.dts
```

## Listing Overlays

Defined in .bashrc

- `beagle$ export SLOTS=/sys/devices/bone_capemgr.*/slots`
- See what's loaded

```
beagle$ cat $SLOTS
 0: 54:PF---
 1: 55:PF---
 2: 56:PF---
 3: 57:PF---
 4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G
 5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI
```

No Capes

## Compiling and Loading Overlays

- Compile with

```
beagle$ dtc -O dtb -o DM-GPIO-Test-00A0.dtbo -b 0 -@ DM-GPIO-Test.dts
Compiling the overlay from .dts to .dtbo
```

- Or

```
beagle$ ./build
Compiling the overlay from .dts to .dtbo
```

- Install

```
beagle$ cp DM-GPIO-Test-00A0.dtbo /lib/firmware
beagle$ echo DM-GPIO-Test > $SLOTS
```

## Verify Overlay

- Check to be sure it worked

```
beagle$ cat $SLOTS
 0: 54:PF---
 1: 55:PF---
 2: 56:PF---
 3: 57:PF---
 4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G
 5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI
 7: ff:P-O-L Override Board Name,00A0,Override Manuf,DM-GPIO-Test
```

- Remove with

```
beagle$ echo -7 > $SLOTS
```

## Making it persistent

- You have to run the command…

```
beagle$ echo DM-GPIO-Test > $SLOTS
```

- …everytime you boot
- However you can make it automatically run at boot time with

```
beagle$ mount /dev/mmcblk0p1 /media/BEAGLEBONE/
```

- This 'mounts' the FAT partition of the eMMC at /media/BEAGLEBONE

```
beagle$ ls /media/BEAGLEBONE/
App       ID.txt       MLO.orig    autorun.inf     u-boot.img.orig
Docs      LICENSE.txt  README.md   u-boot.img      uEnv.txt
Drivers   MLO          START.htm   u-boot.img.new  uEnv.txt.orig
```

- Edit the file `uEnv.txt` and add

```
capemgr.extra_override=DM-GPIO-Test-00A0
capemgr.enable_partn=DM-GPIO-Test
```

Not working with this kernel