Device Tree Overview

Device Tree History

- Device Tree (DT) was created by Open Firmware to allow an operating system at runtime to run on various hardware without hard coding any information.
- Open Firmware has been used on PowerPC and SPARC platforms therefore Linux has supported Device Tree for a long time.
- In 2005 all PowerPC platforms were required to support Device Tree even if they didn't use Open Firmware.

Device Tree History cont..

- A DT representation called Flattened Device Tree (FDT) which could be passed to the kernel as a binary blob.
- Later FDT was generalized to be useable by all architectures and arm, microblaze, mips, powerpc, sparc and x86 currently has mainline DT support.

Device Tree History cont..

- March 2011 Linus had enough with all the changes in the ARM Linux sources and decided that going forward all ARM SOCs must support device tree.

What is Device Tree

- The "Open Firmware Device Tree", or simply Device Tree (DT), is a data structure and language for describing hardware. More specifically, it is a description of hardware that is readable by an operating system so that the operating system doesn't need to hard code details of the machine.
- Provides bindings for non discoverable devices and clocks.

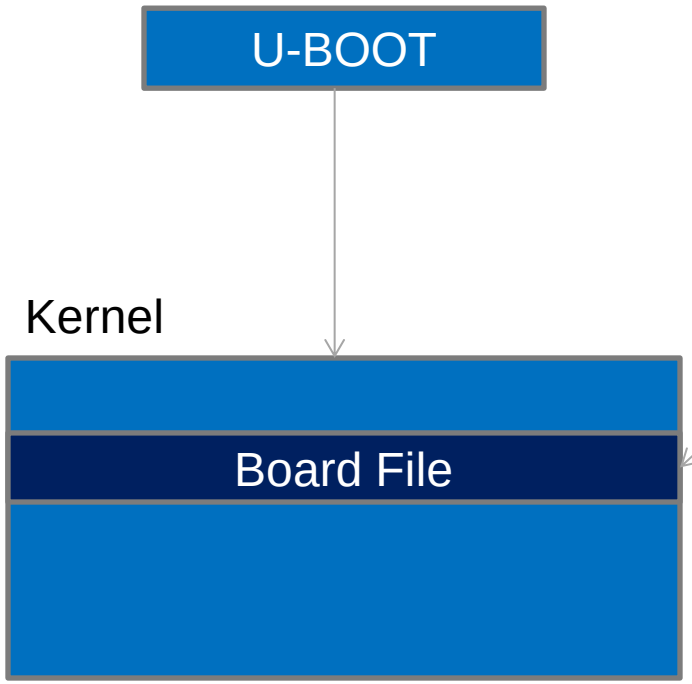
Device Tree

- Pros
  - Attempting to eliminate board specific code from drives.
  - Ability to cleanly support multiple boards with a single kernel.
  - Replaces complex and massive board file!!
    - Current am335x board file line count: 2676
- Cons
  - Kernel size increases
  - Increase in boot time

Device Tree Structure

- Device tree is a structure with a well defined syntax used to describe hardware.
- Since device tree is generic only syntax errors that do not fit the dt specification can be detected when being “compiled”.
- Typos can easily be made and can remain undetected until a driver fails due to a runtime error or a bug is discovered due to misconfiguration.

Pre Device Tree Approach

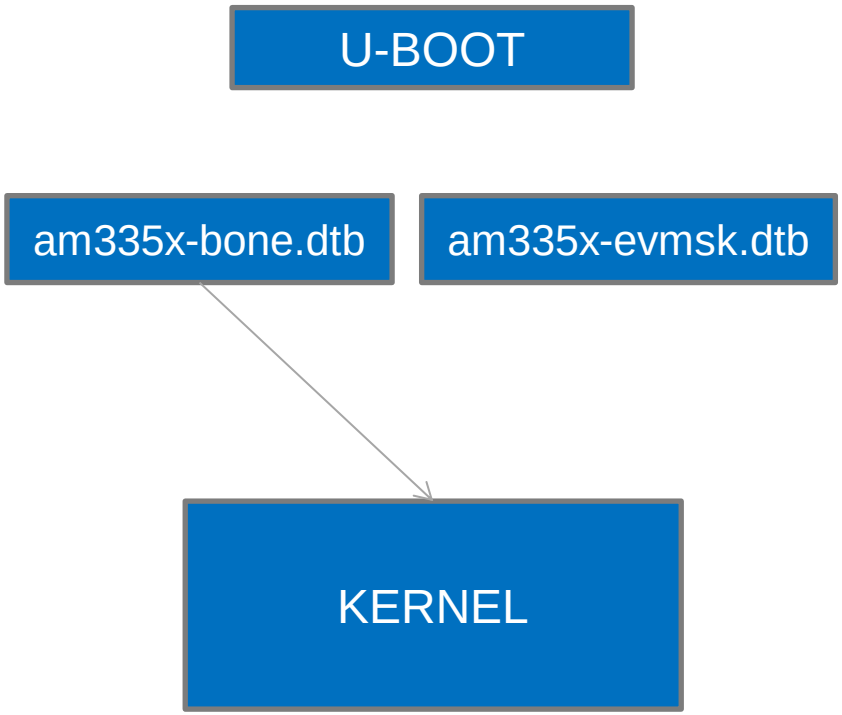


U-boot remains unchanged.
Must contain conditional code if single image supports multiple boards

Massive file containing conditional code to tweak board configuration depended on runtime board detection.

Single Kernel
Single Defconfig used
Must contain conditional code if single image supports multiple boards

Device Tree Approach



U-boot remains unchanged.
U-boot can contain conditional code to determine which dtb file to pass

DTB files produced by kernel or with dtc

U-boot determines at runtime which DTB file to pass to the kernel.

Single Kernel
Single Defconfig used
Board specific code removed from kernel code

Three Ways to build DTB Files

- Using device tree compiler built from sources (git)
- Using device tree compiler built from sources (kernel)
- Using the kernel's makefile (ARM only)

Using device tree compiler built from sources (git)

- Compile dtc

  For Ubuntu Install: bison and flex

  git clone git://jdl.com/software/dtc.git

  cd dtc

  make dtc

- Generate dtb
  - dtc -I dts -O dtb -o <devicetree name>.dtb <devicetree name>.dts

Using device tree compiler built from sources (kernel)

- Compile dtc:

  cd <3.8+ kernel sources>

  make ARCH=arm omap2plus_defconfig

  make ARCH=arm scripts

- Generate dtb
  - scripts/dtc/dtc -I dts -O dtb -o <devicetree name>.dtb <devicetree name>.dts

Using device tree compiler built from sources

- Building manually can be tricky.
- DTC version used will need to be upgraded to match latest version used in the kernel (git method)
- 3.11 kernel introduced C code in the DTS file for am335x.
- AM335x DTS therefore must be ran through the C preprocessor first along with various include files from the kernel.

Using the kernel's makefile (ARM only)

- Two Options:
  - make ARCH=ARM dtbs
    - Builds all dtbs
    - Generated dtbs can be found at arch/arm/boot/dts
  - make ARCH=ARM <devicetree name>.dtb
    - Builds <devicetree name>.dtb
    - Generate dtb can be found at arch/arm/boot/dts
    - Can list multiple dtbs to build more than one dtb
    - Ex: make ARCH=ARM am335x-bone.dtb am335x-evm.dtb

- **Recommended approach to building DTBs**

Loading DTB

- Device tree blob (dtb) is passed to the kernel via U-boot
- U-boot loads the appropriate dtb in to memory
  - Currently stored at 0x80F80000 (fdtaddr) which is a free region of memory that doesn't overlap with any other memory location
- U-boot boots the kernel using the below command
  - bootm/bootz ${loadaddr} - ${fdtaddr};

Binding

- Defines how data should appear in the tree to describe typical hardware characteristics including data busses, interrupt lines, GPIO connections, and peripheral devices.

Important DT Files: AM33xx.dtsi

- Location:
  - <kernel sources>/arch/arm/boot/dts
- Generic SOC DT file
  - Should not contain any board specific information!!
- Should **NOT** be modified.
  - Since this is SOC specific there is no reason for this file to be modified if a user is simply developing a new board.
- Initialize pinmuxing driver
  - Allows dts files to configure their board specific pinmuxing.
- Provides skeleton configuration for chip's on chip peripheral (OCP)
  - Provide configuration for various peripherals that aren't board specific
    - Majority of these peripheral drivers are set to be disabled by default
  - Example Peripherals:
    - DMA
    - SPI
    - I2C
    - LCDC
    - Etc..

Important DT Files: Board Specific

- Location:
  - <kernel sources>/arch/arm/boot/dts
- Example files:
  - am335x-evm.dtb, am335x-sk.dtb am335x-bone.dtb
- Board specific DT file
- Configures board specific pinmuxing
- Enables on chip peripherals being used on the board.
  - Also provides board specific values.
  - Example: Configures LCDC for the specific timings used on the boards LCD
- Enables drivers used by external hardware on the the board
  - Temperature Sensor
  - PMIC

Driver Instantiation Example

AM335x generic configuration for i2c0

Driver node name

```
i2c0: i2c@44e0b000 {
        compatible = "ti,omap4-i2c";
        #address-cells = <1>;
        #size-cells = <0>;
        ti,hwmods = "i2c1";
        reg = <0x44e0b000 0x1000>;
        interrupts = <70>;
        status = "disabled";
};
```

DT device id. Determines which driver This DT binding is meant for.

Driver register information. For i2c this determines memory mapped register for i2c0

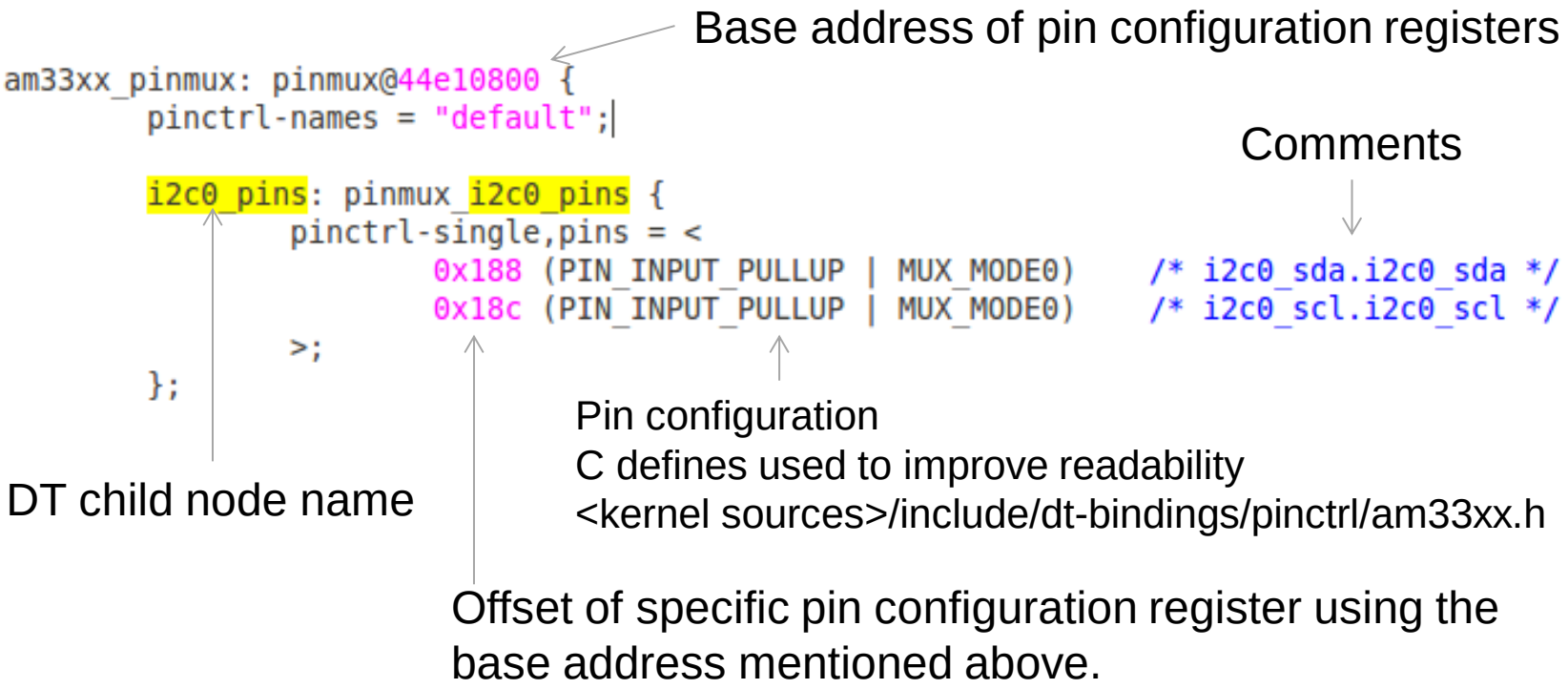
Determines which hardware interrupt is required

Disables driver by default

This snippet is from am33xx.dtsi

Pinmuxing Example

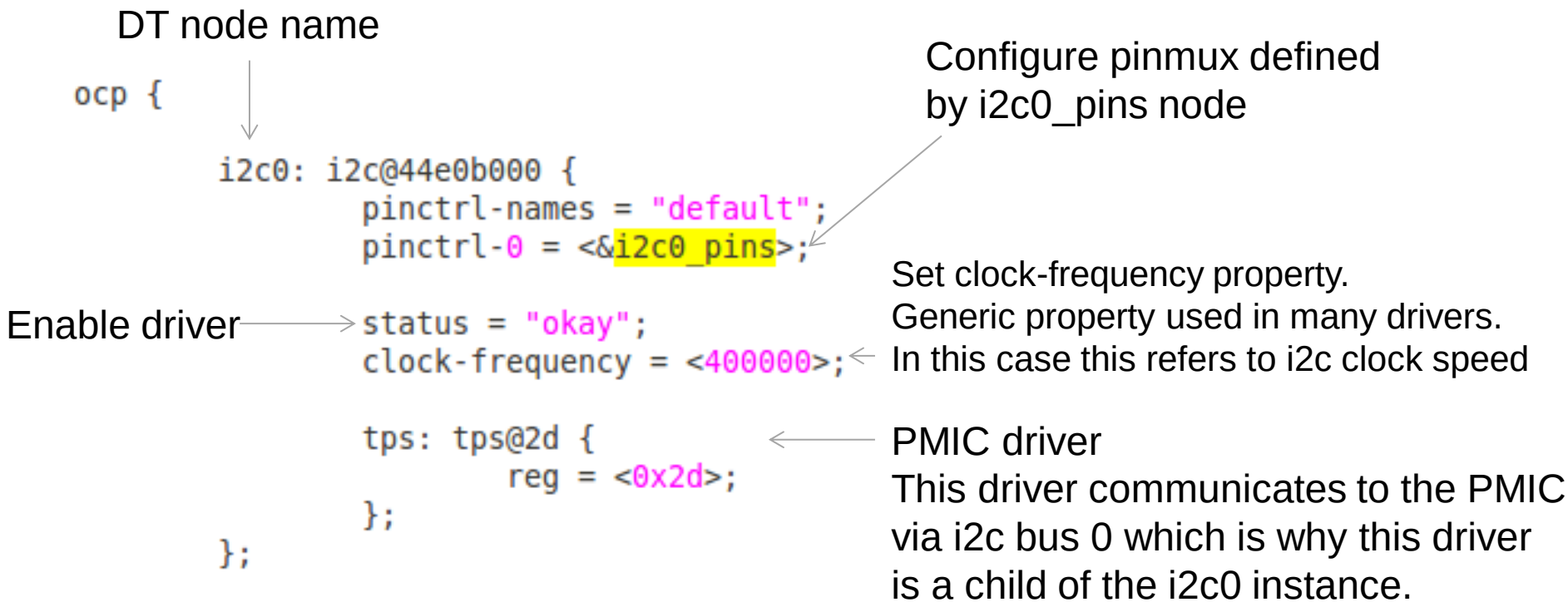
AM335x board specific configuration for i2c0



This snippet is from am335x-bone.dtb

Additional Driver configuration

AM335x board specific configuration for i2c0



This snippet is from am335x-bone.dtb

Device Specific Documentation

- Drivers determine which DT properties it needs
  - Some properties are required while others are optional
- How to determine the syntax, properties the driver expects?
  - Examine driver sources
  - Search for other DT files using driver your interested in
    - Grep for your driver compatible field value
  - Read driver's DT binding documentation. ← Best Approach!
- Drivers that support DT should include DT specific documentation.
  - Location:
    - <kernel sources path>/Documentation/devicetree/bindings
  - Grep for your driver compatible field value

Documentation Example

```
TI LCD Controller on DA830/DA850/AM335x SoC's

Required properties:
- compatible:
        DA830 - "ti,da830-lcdc"
        AM335x SoC's - "ti,am3352-lcdc", "ti,da830-lcdc"
- reg: Address range of lcdc register set
- interrupts: lcdc interrupt
- display-timings: typical videomode of lcd panel, represented as child.
  Refer Documentation/devicetree/bindings/video/display-timing.txt for
  display timing binding details. If multiple videomodes are mentioned
  in display timings node, typical videomode has to be mentioned as the
  native mode or it has to be first child (driver cares only for native
  videomode).

Example:

lcdc@4830e000 {
        compatible = "ti,am3352-lcdc", "ti,da830-lcdc";
        reg =  <0x4830e000 0x1000>;
        interrupts = <36>;
        display-timings {
                800x480p62 {
                        clock-frequency = <30000000>;
                        hactive = <800>;
                        vactive = <480>;
                        hfront-porch = <39>;
                        hback-porch = <39>;
                        hsync-len = <47>;
                        vback-porch = <29>;
                        vfront-porch = <13>;
                        vsync-len = <2>;
                        hsync-active = <1>;
                        vsync-active = <1>;
                };
        };
};
```