Embedded Linux Device Tree

Norman McEntire

Reference

- https://en.wikipedia.org/wiki/Devicetree
- https://www.devicetree.org/

Introduction

- The device tree is:
  - A hardware description file with JSON-like syntax
    - Devices are nodes in the tree
    - Devices can have properties: key/value pairs
  - A data structure that describes the hardware components

History

- Device tree originated from SPARC Open Firmware (OF) standard
- Standard available here
  - https://www.devicetree.org/

File Extensions

- The device tree has several different file extensions
  - .dts - Device Tree Source (human readable text)
  - .dtsi - Device Tree Source Include Files (human readable text)
    - NOTE: The SoC vendor typically provides the .dtsi files
  - .dtb - Device Tree Blob (binary)
  - .dtbo - Device Tree Blob Overlay (binary)

Kernel Location

- Device tree files in the kernel
  - arch/arm/boot/dts
  - arch/arm/boot/dts/vendor

dtc Utility

- The dtc Utility is used to compile .dts and .dtsi files into binary files
  - .dts, .dtsi => dtc => .dtb
- Location in kernel
  - scripts/dtc
- To compile: make dabs

Device Tree Blobs (dtb) on RPi

```
$ ls /boot
bcm2708-rpi-b.dtb       bcm2710-rpi-2-b.dtb       bcm2711-rpi-cm4.dtb     fixup4.dat    kernel7.img       start4.elf
bcm2708-rpi-b-plus.dtb  bcm2710-rpi-3-b.dtb       bcm2711-rpi-cm4-io.dtb  fixup4db.dat  kernel7l.img      start4x.elf
bcm2708-rpi-b-rev1.dtb  bcm2710-rpi-3-b-plus.dtb  bcm2711-rpi-cm4s.dtb    fixup4x.dat   kernel8.img       start_cd.elf
bcm2708-rpi-cm.dtb      bcm2710-rpi-cm3.dtb       bootcode.bin            fixup_cd.dat  kernel.img        start_db.elf
bcm2708-rpi-zero.dtb    bcm2710-rpi-zero-2.dtb    cmdline.txt             fixup.dat     LICENCE.broadcom  start.elf
bcm2708-rpi-zero-w.dtb  bcm2710-rpi-zero-2-w.dtb  config.txt              fixup_db.dat  overlays          start_x.elf
bcm2709-rpi-2-b.dtb     bcm2711-rpi-400.dtb       COPYING.linux           fixup_x.dat   start4cd.elf
bcm2709-rpi-cm2.dtb     bcm2711-rpi-4-b.dtb       fixup4cd.dat            issue.txt     start4db.elf
```

```
$ file /boot/bcm*rpi-4-b.dtb
/boot/bcm2711-rpi-4-b.dtb: Device Tree Blob version 17, size=52593, boot CPU=0,
string block size=4465, DT structure block size=48056
```

Basic Concepts: CONFIG_OF

Kernel Config Related to Device Tree (Open Firmware)

```
$ grep CONFIG_OF .config
CONFIG_OF=y
# CONFIG_OF_UNITTEST is not set
CONFIG_OF_FLATTREE=y
CONFIG_OF_EARLY_FLATTREE=y
CONFIG_OF_KOBJ=y
CONFIG_OF_DYNAMIC=y
CONFIG_OF_ADDRESS=y
CONFIG_OF_IRQ=y
CONFIG_OF_RESERVED_MEM=y
CONFIG_OF_RESOLVE=y
CONFIG_OF_OVERLAY=y
CONFIG_OF_CONFIGFS=y
CONFIG_OF_MDIO=y
CONFIG_OF_GPIO=y
```

Basic Concepts: Header Files

For your driver to use device tree

```
#include <linux/of.h>
#include <linux/of_device.h>
```

Basic Concepts: Sample Showing Data Types

```
/* This is a comment */
// This is also a comment

node_label : nodename@reg {
    string-property = “hello world”;
    string-list = “red”, “green”, “blue”;
    one-int-property = <42>;
    int-list-property = <0x11 0x22 0x33 0x44>;
    Mixed-list-property = “hello”, <42>, [0x11, 0x22, 0x33];
    Byte-array-property = [ 0x11 0x22 0x33 ];
    boolean-property;
}
```

Device Tree Naming Convention

- Every node has a name in one of these formats
  - <name>
  - <name>[@<addresss>]
- Example
  - //Map the I2C Controller to memory
    i2c@ef420000 {
      compatible = “fsl,imx6q-i2”, “fsl,imx21-i2c”;
      reg = <0xef4200 0x1000>;

      // Here the address is the address of the I2C device
      expander@20 {
        compatible = “microchip,mcp23017”;
        reg = <20>;
      }
    }

Lots of examples in kernel source tree!

```
$ ls ./arch/arm/boot/dts/ | wc -l
2616


$ ls ./arch/arm/boot/dts/ | grep rpi | head
bcm2708-rpi-b.dtb
bcm2708-rpi-b.dts
bcm2708-rpi-b-plus.dtb
bcm2708-rpi-b-plus.dts
bcm2708-rpi-b-rev1.dtb
bcm2708-rpi-b-rev1.dts


$ ls ./arch/arm/boot/dts/overlays/ | wc -l
638

$ ls ./arch/arm/boot/dts/overlays/ | head
act-led.dtbo
act-led-overlay.dts
adafruit18.dtbo
adafruit18-overlay.dts
adafruit-st7735r.dtbo
adafruit-st7735r-overlay.dts
```

bcm2711-rpi-4-b.dts

bts = Device Tree Source

```
$ wc -l arch/arm/boot/dts/bcm2711-rpi-4-b.dts
421 arch/arm/boot/dts/bcm2711-rpi-4-b.dts

$ cat arch/arm/boot/dts/bcm2711-rpi-4-b.dts

// SPDX-License-Identifier: GPL-2.0
/dts-v1/;
#define BCM2711
#define i2c0 i2c0if
#include "bcm2711.dtsi"
#include "bcm283x-rpi-wifi-bt.dtsi"
#undef i2c0
#include "bcm270x.dtsi"
#define i2c0 i2c0mux
#include “bcm2711-rpi.dtsi"
. . .
/ {
  compatible = "raspberrypi,4-model-b", "brcm,bcm2711";
  model = "Raspberry Pi 4 Model B";

  chosen {
    /* 8250 auxiliary UART instead of pl011 */
    stdout-path = "serial1:115200n8";
  };

  leds {
    led-act {
      gpios = <&gpio 42 GPIO_ACTIVE_HIGH>;
    };

    led-pwr {
      label = "PWR";
      gpios = <&expgpio 2 GPIO_ACTIVE_LOW>;
      default-state = "keep";
      linux,default-trigger = "default-on";
    };
  };
. . .
```

Code Segment Reading from Device Tree

```
const char *my_string = NULL;
const char *color = NULL;

of_property_read_string(pdev->dev.of_node, “string-property”, &my_string);

of_property_read_string_index(pdev->dev.of_node, “string-list”, 0, &color);
```

Device Tree Overlay

- Used to patch a live device tree
  - Modify the device tree at runtime
- You can update/overwrite nodes
  - But you cannot delete nodes

Device Tree Overlay Example

/boot/config.txt

- dtoverlay=**mcp2515-can0**,oscillator=8000000,interrupt=25

```
$ ls /boot/overlays/mcp*
/boot/overlays/mcp23017.dtbo      /boot/overlays/mcp2515-can1.dtbo  /boot/overlays/mcp3008.dtbo
/boot/overlays/mcp23s17.dtbo      /boot/overlays/mcp2515.dtbo       /boot/overlays/mcp3202.dtbo
/boot/overlays/mcp2515-can0.dtbo  /boot/overlays/mcp251xfd.dtbo     /boot/overlays/mcp342x.dtbo
```

```
$ find . | grep mcp2515-can0
./arch/arm/boot/dts/overlays/mcp2515-can0-overlay.dts
```

mcp2515-can0-overlay.dts - Part 1

/arch/arm/bot/dts/overlays/mcp2515-can0-overlay.dts

```
$ find . | grep mcp2515-can0
./arch/arm/boot/dts/overlays/mcp2515-can0-overlay.dts
```

```
/*
 * Device tree overlay for mcp251x/can0 on spi0.0
 */

/dts-v1/;
/plugin/;

/ {
    compatible = "brcm,bcm2835";
    /* disable spi-dev for spi0.0 */
    fragment@0 {
        target = <&spi0>;
        __overlay__ {
            status = "okay";
        };
    };

    fragment@1 {
  target = <&spidev0>;
  __overlay__ {
     status = "disabled";
  };
 };
```

mcp2515-can0-overlay.dts - Part 2

/arch/arm/bot/dts/overlays/mcp2515-can0-overlay.dts

```
/* the interrupt pin of the can-controller */
fragment@2 {
    target = <&gpio>;
    __overlay__ {
        can0_pins: can0_pins {
            brcm,pins = <25>;
            brcm,function = <0>; /* input */
        };
    };
};

/* the clock/oscillator of the can-controller */
fragment@3 {
    target-path = "/";
    __overlay__ {
        /* external oscillator of mcp2515 on SPI0.0 */
        can0_osc: can0_osc {
            compatible = "fixed-clock";
            #clock-cells = <0>;
            clock-frequency  = <16000000>;
        };
    };
};
};
```

mcp2515-can0-overlay.dts - Part 3

/arch/arm/bot/dts/overlays/mcp2515-can0-overlay.dts

```
/* the spi config of the can-controller itself binding everything together */
fragment@4 {
    target = <&spi0>;
    __overlay__ {
        /* needed to avoid dtc warning */
        #address-cells = <1>;
        #size-cells = <0>;
        can0: mcp2515@0 {
            reg = <0>;
            compatible = "microchip,mcp2515";
            pinctrl-names = "default";
            pinctrl-0 = <&can0_pins>;
            spi-max-frequency = <10000000>;
            interrupt-parent = <&gpio>;
            interrupts = <25 8>; /* IRQ_TYPE_LEVEL_LOW */
            clocks = <&can0_osc>;
        };
    };
};
```

mcp2515-can0-overlay.dts - Part 4

/arch/arm/bot/dts/overlays/mcp2515-can0-overlay.dts

```
        __overrides__ {
            oscillator = <&can0_osc>,"clock-frequency:0";
            spimaxfrequency = <&can0>,"spi-max-frequency:0";
            interrupt = <&can0_pins>,"brcm,pins:0",<&can0>,"interrupts:0";
        };
};
```

- dtoverlay=**mcp2515-can0**,oscillator=8000000,interrupt=25

Demo on RPi

- This demo will show the RPi and the initial SPI settings
- Our steps
  - Step 1. Confirm SPI turned off on RPi by default. No /dev/spi* devices.
    - #dtparam=spi=on
    - lsmod | grep spa (no modules will be found)
    - ls /dev/spi* (no devices will be found)
  - Step 2. Edit /boot/config.txt to enable SPI. Reboot. See two /dev/spi* devices.
    - dtparam=spi=on
    - lsmod | grep spi (modules found: spi_bcm2835, spider)
    - ls /dev/spi* (devices found: /dev/spidev0.0, /dev/spidev0.1)
  - Step 3. Edit /boot/config.txt for overlay for SPI device MCP2514 CAN (Controller Rear Network). Reboot. Observe
    - dtoverlay=mcp2515-can0,oscillator=8000000,interrupt=25
    - ls /dev/spi* (only /dev/spidev0.1 available)
    - dmsg | grep spi
      [7.663803] mcp251x spi0.0 can0: MCP2515 successfully initialized.

Summary

- This was a brief intro to Device Tree
- The SoC provider typically provides the “base” set of Device Tree files
- As a device driver developer, you may provide an overlay file
  - The overlay replace the default Device Tree provided by SoC Provider
- Many Embedded Linux projects do not need to make any changes to device tree
  - Rather use platform as provided by vendor (e.g RPi)