Aricent
.ALTRAN GROUP

Linux Device Drivers – Device Tree

By Jitesh Verma

Device Tree

- Concept taken from Open Firmware (OF) open source project
- To make Linux Kernel code independent of computing system hardware description
- Computing system hardware Description
- Hierarchical organization of device hardware description in the form of a Tree data structure
- Each hardware component is described as a node in the Device Tree
- A node might have properties and/or other child nodes

Devices in Device Tree

- Board
- SoC
- CPUs/Cores
- Memories
- Buses
- Platform Devices
- Peripheral Devices
  - Master Devices
  - Slave Devices

Device Tree Source – Node Definition Format

```
[label:] node-name[@unit-address] {
        [properties definitions]
        [child nodes]
};
```

Note: Label of a node is optional. Use if required.

Device Tree Source – Property Definition Format

Property with value:

[label:] property-name = value;

OR

Property without value:

[label:] property-name;

Note: Label of a node is optional. Use if required.

Device Tree Source – Special Property Definitions

- Used for matching device/device configurations with device driver:

  compatible = "ti,omap3,spi-master";

- Used for enabling or disabling a Device Tree node:

  status = “okay”;

  OR

  status = “disabled”;

A Theoretical Device Tree Source Example

```
/ {
        node@0 {
                a-string-property = "A string";
                a-string-list-property = "first string", "second string";
                a-byte-string-property = [0x11 0x22 51 0x44];

                child-node@0 {
                        child-property1;
                        child-property2 = <100>;
                        child-property3 = <0xA 20 30 0x28 50>;
                        a-reference-to-something = <&node1>;
                };
                node1: child-node@1 {
                ---------
                };
        };
};
```

A Realistic Device Tree Source Example

```
#include "arm.dtsi"
/ {
   compatible = "ti,omap2";
   kernel_bootargs {
                bootargs = "<kernel boot args here>";
   };
   memory {
      reg = <0x00000000 0x10000000>;
   };
   soc {
                cpus {
                             cpu@0 {
                                          compatible = "arm,cortex-a9";
                             };
                             cpu@1 {
                                          compatible = "arm,cortex-a9";
                             };
                };
                apb@d4000000 {
                   uart1: uart@d4017000 {
                   status = "okay";
                   };
                   i2c1: i2c@d4011000 {
                                ----------
                                status = "okay";
                   };
                ----------
                };
};
```

Device Tree Overlay – Over-riding a Node

soc.dtsi

```
soc {
----------------------
        i2c1: i2c@d4011000 {
                -------------
                frequency = <100000>
                status = "okay";
        };
----------------------
};
```

board.dts

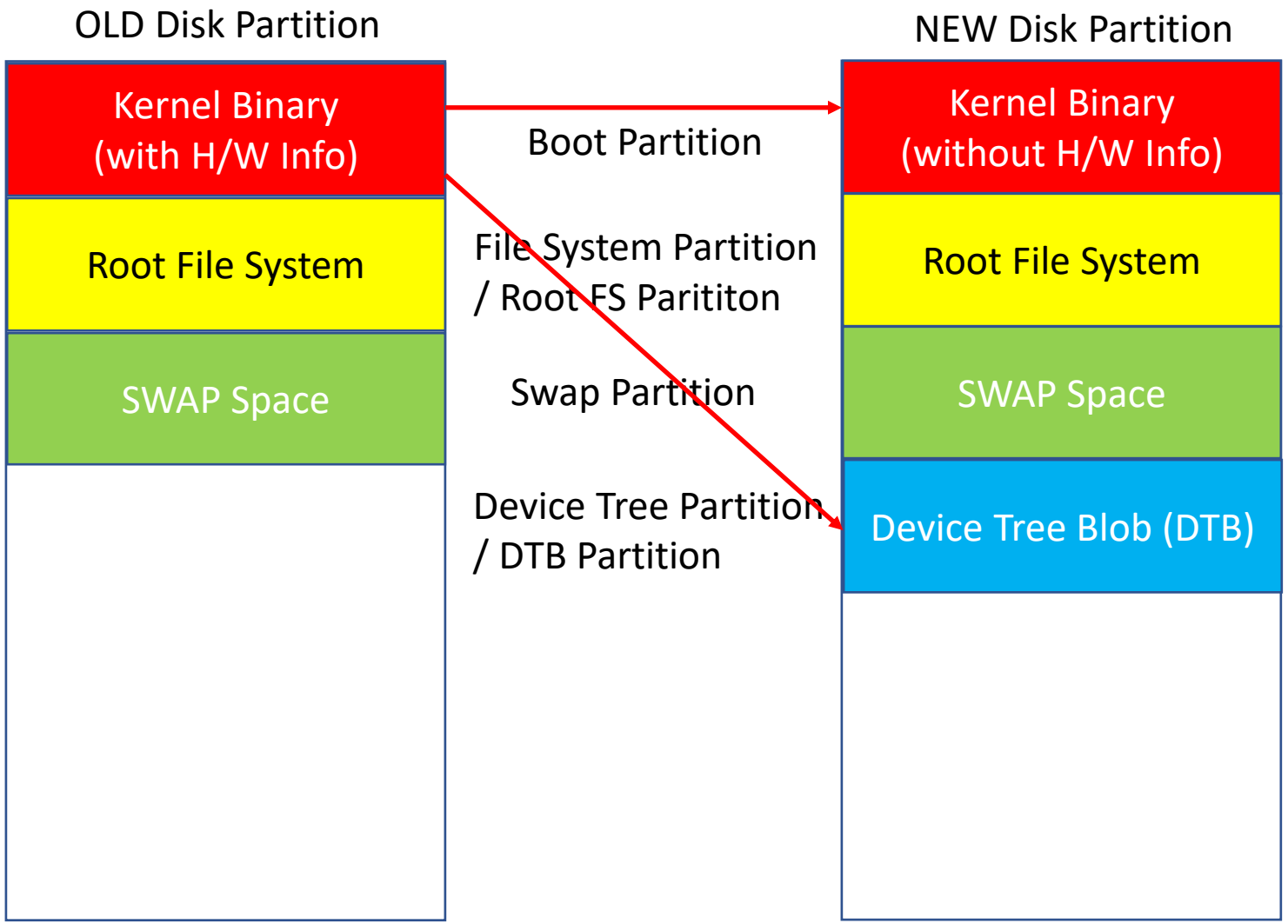
```
#include “soc.dtsi“

/{
        ----------------------
        ----------------------
          i2c1: i2c@d4011000 {
                  frequency = <400000>
                  status = "okay";
          };
        ----------------------
        ----------------------
};
```

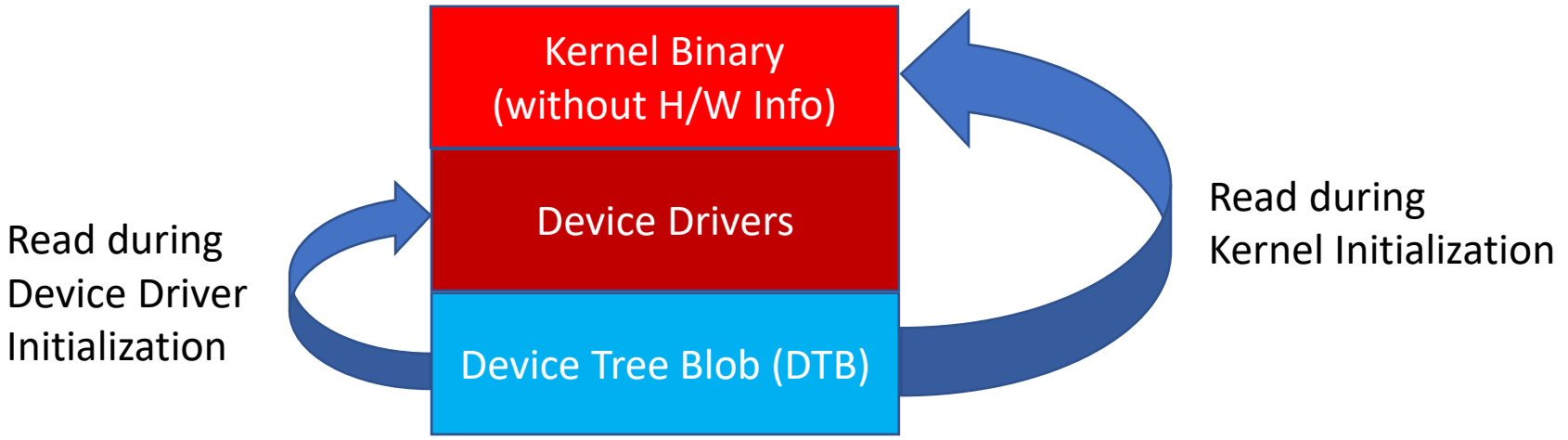
Device Tree – Useful Info

- DTS: Device Tree Source (Code)
- DTB: Device Tree Blob (Binary)
- DTC: Device Tree Compiler (script)
- DTS File Path: $KERNEL-SOURCE-DIR/arch/arm/boot/dts
- TI DTS File Path: $KERNEL-SOURCE-DIR/arch/arm/boot/dts/ti
- Proc./SoC-level DTS File: .dtsi
- Board-level DTS File: .dts
- Device Tree Blob File: .dtb

Linux OS - Hard Disk Partitions



Device Tree Usage



Device Tree API – Accessing an unsigned char (8-bit) Data Type

DTS Node Definition:

```
node_name {
        a = <100>;
        b = <0XFF>;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned char  c1, c2;
np = of_find_node_by_name(NULL, "node_name");
of_property_read_u8(np, “a", &c1);
of_property_read_u8(np, “b", &c2);
```

Device Tree API – Accessing an unsigned short int (16-bit) Data Type

DTS Node Definition:

```
node_name {
        a = <10000>;
        b = <0XFFFF>;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  short int s1, s2;
np = of_find_node_by_name(NULL, "node_name");
of_property_read_u16(np, "a", &s1);
of_property_read_u16(np, "b", &s2);
```

Device Tree API – Accessing an unsigned long int (32-bit) Data Type

DTS Node Definition:

```
node_name {
        a = <1000000000>;
        b = <0XFFFFFFFF>;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  long l1, l2;
np = of_find_node_by_name(NULL, "node_name");
of_property_read_u32(np, "a", &l1);
of_property_read_u32(np, "b", &l2);
```

Device Tree API – Accessing an Array of unsigned long int (32-bit) Data Type

DTS Node Definition:

```
node_name {
        a = <1000 0x1234 3000 4000>;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  long int l[4];
np = of_find_node_by_name(NULL, "node_name");
of_property_read_u32_array(np, "a", l, 4);
```

Device Tree API – Accessing an Array of unsigned long int (32-bit) Data Type (Alternate)

DTS Node Definition:

```
node_name {
        a = <1000 0x1234 3000 4000>;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  long int l[4];
np = of_find_node_by_name(NULL, "node_name");
for (int i = 0; i < 4; i++)
        of_property_read_u32_index(np, "a", i, &l[i]);
```

Device Tree API – Accessing a String Data Type

DTS Node Definition:

```
node_name {
        mode = “programmed-io”;      // dma or programmed-io
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  char str[20];
np = of_find_node_by_name(NULL, "node_name");
of_property_read_string(np, “mode", str);
```

Device Tree API – Accessing an Array of Strings Data Type

DTS Node Definition:

```
node_name {
        mode = “programmed-io”,  “dma”;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  char str[10][2];
np = of_find_node_by_name(NULL, "node_name");
of_property_read_string_array(np, “mode", str, 2);
```

Device Tree API – Accessing an Array of Strings Data Type (Alternate)

DTS Node Definition:

```
node_name {
        mode = “programmed-io”,  “dma”;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
unsigned  char str[10][2];
np = of_find_node_by_name(NULL, "node_name");
for (int i=0; i<2, i++)
        of_property_read_string_index(np, “mode", i, str[i]);
```

Device Tree API – Accessing a Boolean Data Type

DTS Node Definition:

```
node_name {
        flag = “true”;
};
```

Example Device Driver Code Fragment:

```
struct device_node *np;
boolean b;
np = of_find_node_by_name(NULL, "node_name");
b = of_property_read_bool(np, "flag")
```

Device Tree References

1) https://elinux.org/Device_Tree_Reference
2) https://elinux.org/Device_Tree_Linux
3) https://docs.kernel.org/devicetree/kernel-api.html
4) https://elinux.org/images/f/f9/Petazzoni-device-tree-dummies_0.pdf