Device Tree 101

Organized in partnership with ST
February 9, 2021

Thomas Petazzoni
thomas.petazzoni@bootlin.com

© Copyright 2004-2021, Bootlin.
Creative Commons BY-SA 3.0 license.
Corrections, suggestions, contributions and translations are welcome!
Who is speaking?

- **Thomas Petazzoni**
- **Chief Technical Officer** at Bootlin
- Joined in 2008, employee #1
- **Embedded Linux & Linux kernel engineer**, open-source contributor
  - Author of the *Device Tree for Dummies* talk in 2013/2014
  - Buildroot co-maintainer
  - Linux kernel contributor: ≈ 900 contributions
- Member of Embedded Linux Conference (Europe) program committee
- Based in Toulouse, France
Agenda

- Bootlin introduction
- STM32MP1 introduction
- Why the Device Tree?
- Basic Device Tree syntax
- Device Tree inheritance
- Device Tree specifications and bindings
- Device Tree and Linux kernel drivers
- Common properties and examples
In business since 2004

Team based in France

Serving **customers worldwide**

- 18% revenue from France
- 44% revenue from EU except France
- 38% revenue outside EU

**Highly focused and recognized expertise**

- Embedded Linux
- Linux kernel
- Embedded Linux build systems

**Activities**

- **Training** courses (≈ 20% revenue)
- **Engineering** services (≈ 80% revenue)
Bootlin training courses

- Embedded Linux system development
  On-site: 4 or 5 days
  Online: 7 * 4 hours

- Linux kernel driver development
  On-site: 5 days
  Online: 7 * 4 hours

- Yocto Project system development
  On-site: 3 days
  Online: 4 * 4 hours

- Buildroot system development
  On-site: 3 days
  Online: 4 * 4 hours

- Understanding the Linux graphics stack
  On-site: 2 days
  Online: 4 * 4 hours

- Embedded Linux boot time optimization
  On-site: 3 days
  Online: 4 * 4 hours
Why choose Bootlin training courses?

▶ Complete training materials **freely available**
  ▶ **Open-source license**: Creative Commons
  ▶ Allows to **verify** in detail the course contents
  ▶ Shows Bootlin commitment to **knowledge sharing**
  ▶ **Unique** in the training industry

[Bootlin](https://bootlin.com) - Kernel, drivers and embedded Linux - Development, consulting, training and support - https://bootlin.com
Why choose Bootlin training courses?

- Complete training materials **freely available**
  - **Open-source license**: Creative Commons
  - Allows to **verify** in detail the course contents
  - Shows Bootlin commitment to **knowledge sharing**
  - **Unique** in the training industry

- **Experienced** trainers
  - Bootlin trainers are also engineers
  - Working on **real engineering** projects
  - Up-to-date and in-field experience
Why choose Bootlin training courses?

- Complete training materials **freely available**
  - **Open-source license**: Creative Commons
  - Allows to **verify** in detail the course contents
  - Shows Bootlin commitment to **knowledge sharing**
  - **Unique** in the training industry

- **Experienced** trainers
  - Bootlin trainers are also engineers
  - Working on **real engineering** projects
  - Up-to-date and in-field experience

- **Worldwide recognized** training courses
  - Taught 100s of sessions
  - To **1000s of engineers**
  - For the past 15 years

- [Kernel, drivers and embedded Linux - Development, consulting, training and support - https://bootlin.com](https://bootlin.com)
Bootlin engineering services

► Main activities
  ► Linux **Board Support Package** development, update and maintenance
  ► Linux **kernel drivers** development
  ► Bootloader and Linux kernel **porting**
  ► System integration: Yocto, Buildroot, boot time, secure boot, etc.
  ► **Upstreaming**
  ► **Consulting** and technical support
  ► Focus on the low-level software stack

► Customers
  ► **Silicon vendors**: interested in U-Boot, Linux, Buildroot or Yocto support for their product, usually upstream
  ► **Embedded system manufacturers**: complete BSP, specific drivers, debugging, optimization, consulting
Bootlin: open-source contributor

- Bootlin 20th contributing company worldwide to the Linux kernel
- **7600+ patches** contributed, mainly around hardware support
- Maintainers of several subsystems of the kernel: I3C, RTC, MTD, and several platforms
- Key contributor to Buildroot: co-maintainer, **5000+ patches** contributed
- Contributor to the **Yocto Project**
- Contributions to Barebox, Linux Test Project, etc.
- Freely available training materials
- Numerous **talks** at conferences to share technical knowledge

<table>
<thead>
<tr>
<th>No.</th>
<th>Company</th>
<th>Patches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unknown</td>
<td>140019</td>
</tr>
<tr>
<td>2</td>
<td>Intel</td>
<td>94806</td>
</tr>
<tr>
<td>3</td>
<td>Red Hat</td>
<td>78140</td>
</tr>
<tr>
<td>4</td>
<td>Hobbyists</td>
<td>73603</td>
</tr>
<tr>
<td>5</td>
<td>Novell</td>
<td>39210</td>
</tr>
<tr>
<td>6</td>
<td>IBM</td>
<td>35085</td>
</tr>
<tr>
<td>7</td>
<td>Linaro</td>
<td>28288</td>
</tr>
<tr>
<td>8</td>
<td>AMD</td>
<td>22426</td>
</tr>
<tr>
<td>9</td>
<td>Google</td>
<td>26489</td>
</tr>
<tr>
<td>10</td>
<td>Renesas Electronics</td>
<td>18443</td>
</tr>
<tr>
<td>11</td>
<td>Oracle</td>
<td>17729</td>
</tr>
<tr>
<td>12</td>
<td>Samsung</td>
<td>17514</td>
</tr>
<tr>
<td>13</td>
<td>Texas Instruments</td>
<td>16372</td>
</tr>
<tr>
<td>14</td>
<td>Huawei</td>
<td>13377</td>
</tr>
<tr>
<td>15</td>
<td>Mellanox Technologies</td>
<td>11477</td>
</tr>
<tr>
<td>16</td>
<td>ARM</td>
<td>8919</td>
</tr>
<tr>
<td>17</td>
<td>Academics</td>
<td>8560</td>
</tr>
<tr>
<td>18</td>
<td>Consultants</td>
<td>8073</td>
</tr>
<tr>
<td>19</td>
<td>Broadcom</td>
<td>8011</td>
</tr>
<tr>
<td>20</td>
<td>Bootlin</td>
<td>7611</td>
</tr>
<tr>
<td>21</td>
<td>XIP</td>
<td>7549</td>
</tr>
<tr>
<td>22</td>
<td>Linutronix</td>
<td>7430</td>
</tr>
<tr>
<td>23</td>
<td>NVIDIA</td>
<td>6951</td>
</tr>
<tr>
<td>24</td>
<td>Canonical</td>
<td>6855</td>
</tr>
<tr>
<td>25</td>
<td>Linux Foundation</td>
<td>6369</td>
</tr>
<tr>
<td>26</td>
<td>Code Aurora Forum</td>
<td>6280</td>
</tr>
<tr>
<td>27</td>
<td>Pengutronix</td>
<td>6201</td>
</tr>
<tr>
<td>28</td>
<td>VISION Engraving and Routing Systems</td>
<td>6245</td>
</tr>
<tr>
<td>29</td>
<td>Analog Devices</td>
<td>5944</td>
</tr>
<tr>
<td>30</td>
<td>Fujitsu</td>
<td>5120</td>
</tr>
<tr>
<td>31</td>
<td>Qualcomm</td>
<td>4903</td>
</tr>
<tr>
<td>32</td>
<td>Freescale</td>
<td>4694</td>
</tr>
<tr>
<td>33</td>
<td>Wolfsun Microelectronics</td>
<td>4180</td>
</tr>
<tr>
<td>34</td>
<td>Marvell</td>
<td>4176</td>
</tr>
<tr>
<td>35</td>
<td>Nokia</td>
<td>4097</td>
</tr>
<tr>
<td>36</td>
<td>Cisco</td>
<td>4071</td>
</tr>
<tr>
<td>37</td>
<td>Parallels</td>
<td>3841</td>
</tr>
<tr>
<td>38</td>
<td>Imagination Technologies</td>
<td>3774</td>
</tr>
<tr>
<td>39</td>
<td>Facebook</td>
<td>3484</td>
</tr>
<tr>
<td>40</td>
<td>QLogic</td>
<td>3394</td>
</tr>
<tr>
<td>41</td>
<td>ST Microelectronics</td>
<td>3180</td>
</tr>
<tr>
<td>42</td>
<td>Astaro</td>
<td>2983</td>
</tr>
<tr>
<td>43</td>
<td>NetApp</td>
<td>2868</td>
</tr>
</tbody>
</table>
### STM32MP157F

#### System
- 5x LDGs
- Crystal & Internal oscillators
- MDMA + 2x DMA
- Reset and Clock
- Watchdogs (2x I & W)
- 96-bit unique ID
- Up to 176 GPIOs

#### Dual Cortex-A7 @ 800MHz
- Core 1 @ 800MHz
  - L1 32KB I / 32KB D
  - NECN SMII
  - 256KB L2 cache
- Core 2 @ 800MHz
  - L1 32KB I / 32KB D
  - NECN SMII

#### Cortex-M4 @ 209MHz
- FPU
- MPU

#### DDR3/DDR3L 32-bit @ 533MHz
- Retention RAM 84KB
- Boot ROM 128KB
- System RAM 256KB
- MCU System RAM 384KB
- Backup RAM 4KB
- OTP Fuse 3Kb

#### Control
- 2x 16-bit motor control
- 10x 16-bit timer
- 2x 32-bit timer

#### Connectivty
- 24-bit Parallel RGB Display
- Camera Interface
- 1Gbps Ethernet
- 2x FDCAN / TTCAN
- 2x USB2.0 Host HS
- USB2.0 OTG FS/HS
- MIPI CSI 2 lanes @ 1Gbps
- HDMI CEC
- DFSDM 8 channels / 6 filters
- 6x SPI / 3x I2C
- 4x UART, 4x USART
- 4x SAI
- SPDIF Tx / Rx 4 inputs
- Dual Quad SPI
- 16-bit SLC NAND, 6-bit ECC

#### Analog
- 2x 18-bit ADC
- 2x 12-bit DAC
- Temperature sensor
### STM32MP1 Series

**Arm® Cortex®-A7 – up to 800 MHz**

#### Acceleration
- Dual core Arm® Cortex®-A7 processor
- L1 and L2 caches
- 3D Graphic Processing Unit
- Floating Point Unit + Arm® Neon™
- Arm® Cortex®-M4 209 MHz coprocessor
- DMA + DMA
- LPDDR2/LPDDR3 16/32-bit 533 MHz
- DDR3/DDR3L 16/32-bit 533 MHz

<table>
<thead>
<tr>
<th>Product lines</th>
<th>Cortex®-A7 core</th>
<th>Cortex®-M4 core</th>
<th>32-bit</th>
<th>Flash</th>
<th>CPU Crypto</th>
<th>FPU</th>
<th>MP3</th>
<th>Ethernet</th>
<th>Junction Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32MP151A</td>
<td>1</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-40°C to 125°C</td>
</tr>
<tr>
<td>STM32MP151C</td>
<td>1</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-20°C to 105°C</td>
</tr>
<tr>
<td>STM32MP151D</td>
<td>1</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-40°C to 125°C</td>
</tr>
<tr>
<td>STM32MP151F</td>
<td>1</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-20°C to 105°C</td>
</tr>
<tr>
<td>STM32MP153A</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-40°C to 125°C</td>
</tr>
<tr>
<td>STM32MP153C</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-20°C to 105°C</td>
</tr>
<tr>
<td>STM32MP153D</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-40°C to 125°C</td>
</tr>
<tr>
<td>STM32MP153F</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-20°C to 105°C</td>
</tr>
<tr>
<td>STM32MP157A</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>533</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-40°C to 125°C</td>
</tr>
<tr>
<td>STM32MP157C</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>533</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-20°C to 105°C</td>
</tr>
<tr>
<td>STM32MP157D</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>533</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-40°C to 125°C</td>
</tr>
<tr>
<td>STM32MP157F</td>
<td>2</td>
<td>650</td>
<td>1</td>
<td>209</td>
<td>533</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-20°C to 105°C</td>
</tr>
</tbody>
</table>

#### Connectivity
- 2 x USB2.0 HS Host
- USB2.0 OTG FS/HS
- 3 x UART/LA1, SPI, PC
- 2 x (T)F/SD-CN2, 0
- Gigabit ethernet (EEG 1386***
- FMC (NAND Flash)
- Camera I/F
- Dual mode Quad-SPI
- USB 2 (host)*

**Notes:**
* Not available in all product lines
** 16-bit for LFBA4A48 and FBBA031 packages, 18-bit only for LFBA4A54 and FBBA057 packages
*** 10/100M Ethernet only for LFBA4A54 and FBBA057 packages

---

STM32MP1 system-on-chip family

bootlin® - Kernel, drivers and embedded Linux - Development, consulting, training and support - https://bootlin.com
STM32MP1 Discovery Kits

- Discovery Kit 1 (DK1)
  - SoC: STM32MP157A
  - 512 MB DDR, microSD
  - 1G Ethernet, 1x USB-C, 4x USB-A, LEDs, buttons
  - HDMI, audio codec, DSI connector
  - GPIO connectors, Arduino/RaspberryPi shields
  - On-board ST-Link

- Discovery Kit 2 (DK2)
  - SoC: STM32MP157C
  - Same as DK1
  - WiFi/Bluetooth
  - Display + touchscreen
Some hardware busses provide **discoverability** mechanisms

- E.g: PCI(e), USB
- One does not need to know ahead of time what will be connected on these busses
- Devices can be enumerated and identified at runtime
- Concept of *vendor ID*, *product ID*, *device class*, etc.
Some hardware busses provide **discoverability** mechanisms

- E.g: PCI(e), USB
- One does not need to know ahead of time what will be connected on these busses
- Devices can be enumerated and identified at runtime
- Concept of *vendor ID*, *product ID*, *device class*, etc.

But many hardware busses **do not provide discoverability** mechanisms

- E.g: I2C, SPI, 1-wire, memory-mapped, etc.
- One needs to know what is connected on those busses, and how they are connected to the rest of the system
- Embedded systems typically make extensive use of such busses
Hardware description for non-discoverable hardware

Allows the operating system or bootloader to **know things like**:

► This **system-on-chip** has:
  ► 2 Cortex-A7 CPU cores
  ► 2 memory-mapped UART controllers of *this* variant, one with registers at 0x5c000000 and IRQ 37, and another with registers at 0x4000e000 and IRQ 38
  ► 3 I2C controllers of *that* variant, with registers at *those* memory-mapped addresses, *those* IRQs and taking their input clock from *this* source

► This **board** has a CS42L51 audio codec
  ► Connected on the I2C bus 1 of the SoC, at slave address 0x4A
  ► Connected to the SAI interface 2 of the SoC
  ► With its reset signal connected to GPIO 67 of the SoC

► ...

→ These details **cannot be guessed** by the operating system/bootloader.
Describing non-discoverable hardware

1. Directly in the OS/bootloader code

- Using compiled data structures, typically in C
- How it was done on most embedded platforms in Linux, U-Boot.
- Considered not maintainable/sustainable on ARM32, which motivated the move to another solution.
Describing non-discoverable hardware

1. Directly in the OS/bootloader code
2. Using ACPI tables
   - On x86 systems, but also on a subset of ARM64 platforms
   - Tables provided by the firmware

On x86 systems, but also on a subset of ARM64 platforms
- Tables provided by the firmware
1. Describing non-discoverable hardware

2. Directly in the OS/bootloader code

3. Using a Device Tree

- Originates from **OpenFirmware**, defined by Sun, used on SPARC and PowerPC
  - That’s why many Linux/U-Boot functions related to DT have a `of_` prefix
- Now used most embedded-oriented CPU architectures that run Linux: ARC, ARM64, RISC-V, ARM32, PowerPC, Xtensa, MIPS, etc.
- Writing/tweaking a DT is now always necessary when porting Linux to a new board.
- The **topic of this talk**!
Device Tree: from source to blob

- A tree data structure describing the hardware is written by a developer in a **Device Tree Source** file, `.dts`
- Processed by the **Device Tree Compiler**, `dtc`
- Produces a more efficient representation: **Device Tree Blob**, `.dtb`
- Additional C preprocessor pass
- `.dtb` → accurately describes the hardware platform in an **OS-agnostic** way.
- `.dtb` ≈ few dozens of kilobytes
- DTB also called **FDT**, *Flattened Device Tree*, once loaded into memory.
  - `fdt` command in U-Boot
  - `fdt_` APIs
dtc example

$ cat foo.dts
/dts-v1/;

/ {
    welcome = <0xBADCAFE>;
    bootlin {
        webinar = "great";
        demo = <1>, <2>, <3>;
    };
};
$ cat foo.dts
/dts-v1/;

/ {
  welcome = <0xBADCAFE>;
  bootlin {
    webinar = "great";
    demo = <1>, <2>, <3>;
  }
};

$ dtc -I dts -O dtb -o foo.dtb foo.dts
$ ls -l foo.dt*
-rw-r--r-- 1 thomas thomas 169 ... foo.dtb
-rw-r--r-- 1 thomas thomas 102 ... foo.dts
dtc example

$ cat foo.dts
/dts-v1/;

/ {
    welcome = <0xBADCAFE>;
    bootlin {
        webinar = "great";
        demo = <1>, <2>, <3>;
    }
};

$ dtc -I dts -O dtb -o foo.dtb
/dts-v1/;

/ {
    welcome = <0xbadcafe>;
    bootlin {
        webinar = "great";
        demo = <0x01 0x02 0x03>;
    }
};

$ dtc -I dtb -O dts foo.dtb
/dts-v1/;

$ ls -l foo.dt*
-rw-r--r-- 1 thomas thomas 169 ... foo.dtb
-rw-r--r-- 1 thomas thomas 102 ... foo.dts
Device Tree: using the blob

- Can be **linked directly** inside a bootloader binary
  - For example: U-Boot, Barebox
- Can be **passed** to the operating system by the bootloader
  - Most common mechanism for the Linux kernel
  - U-Boot:
    ```
    bootz <kernel_addr> - <dtb_addr>
    ```
  - The DTB address is passed through a dedicated CPU register to the kernel: `r2` on ARM32
  - Bootloader can adjust the DTB before passing it to the kernel
- The DTB parsing can be done using `libfdt`, or ad-hoc code
Where are Device Tree Sources located?

- Even though they are OS-agnostic, **no central and OS-neutral** place to host Device Tree sources and share them between projects
  - Often discussed, never done

- In practice, the Linux kernel sources can be considered as the **canonical location** for Device Tree Source files
  - `arch/<ARCH>/boot/dts`
  - ≈ 4700 Device Tree Source files in Linux as of 5.10

- Duplicated/synced in various projects
  - U-Boot, Barebox, TF-A
Example with STM32MP157A-DK1

1st stage: **TF-A**
- DT in `fdts/stm32mp157a-dk1.dts`
- Build with `PLAT=stm32mp1`  `DTB_FILE_NAME=stm32mp157a-dk1.dtb`
- Bundles the DTB in the resulting `tf-a-stm32mp157a-dk1.stm32`

2nd stage: **U-Boot**
- DT in `arch/arm/dts/stm32mp157a-dk1.dts`
- Configure with `stm32mp15_trusted_defconfig`
- Build with `DEVICE_TREE=stm32mp157a-dk1`
- Bundles the DTB in the resulting `u-boot.stm32`

OS: **Linux kernel**
- DT in `arch/arm/boot/dts/stm32mp157a-dk1.dts`
- Configure with `multi_v7_defconfig`
- Build
- Kernel image: `arch/arm/boot/zImage`, DTB: `arch/arm/boot/dts/stm32mp157a-dk1.dtb`
Booting Linux on STM32MP157A-DK1

U-Boot 2020.07 (Feb 04 2021 - 16:27:10 +0100)
CPU: STM32MP157AAC Rev.B
Model: STMicroelectronics STM32MP157A-DK1 Discovery Board
Booting Linux on STM32MP157A-DK1

U-Boot 2020.07 (Feb 04 2021 - 16:27:10 +0100)
CPU: STM32MP157AAC Rev.B
Model: STMicroelectronics STM32MP157A-DK1 Discovery Board

STM32MP> printenv kernel_addr_r
kernel_addr_r=0xc2000000
STM32MP> printenv fdt_addr_r
fdt_addr_r=0xc4000000

STM32MP> ext4load mmc 0:4 ${kernel_addr_r} /boot/zImage
4202992 bytes read in 207 ms (19.4 MiB/s)
STM32MP> ext4load mmc 0:4 ${fdt_addr_r} /boot/stm32mp157a-dk1.dtb
53881 bytes read in 31 ms (1.7 MiB/s)
STM32MP> bootz ${kernel_addr_r} - ${fdt_addr_r}

Kernel image @ 0xc2000000 [ 0x000000 - 0x4021f0 ]
## Flattened Device Tree blob at c4000000
Booting using the fdt blob at 0xc4000000
Loading Device Tree to cffef000, end cffff278 ... OK
Starting kernel ...
[ 0.000000] Linux version 5.8.13 (thomas@windsurf) (arm-none-linux-gnueabihf-gcc....)
[ 0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387d
...
Booting Linux on STM32MP157A-DK1

U-Boot 2020.07 (Feb 04 2021 - 16:27:10 +0100)
CPU: STM32MP157AAC Rev.B
Model: STMicroelectronics STM32MP157A-DK1 Discovery Board

STM32MP> printenv kernel_addr_r
kernel_addr_r=0xc2000000
STM32MP> printenv fdt_addr_r
fdt_addr_r=0xc4000000

STM32MP> ext4load mmc 0:4 ${kernel_addr_r} /boot/zImage
4202992 bytes read in 207 ms (19.4 MiB/s)
STM32MP> ext4load mmc 0:4 ${fdt_addr_r} /boot/stm32mp157a-dk1.dtb
53881 bytes read in 31 ms (1.7 MiB/s)

Kernel image @ 0xc2000000 [ 0x000000 - 0x4021f0 ]
## Flattened Device Tree blob at c4000000
Booting using the fdt blob at 0xc4000000
Loading Device Tree to cffef000, end cffff278 ... OK
Starting kernel...
[ 0.000000] Linux version 5.8.13 (thomas@windsurf) (arm-none-linux-gnueabihf-gcc...)
[ 0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387d...
Booting Linux on STM32MP157A-DK1

U-Boot 2020.07 (Feb 04 2021 - 16:27:10 +0100)
CPU: STM32MP157AAC Rev.B
Model: STMicroelectronics STM32MP157A-DK1 Discovery Board

STM32MP> printenv kernel_addr_r
kernel_addr_r=0xc2000000
STM32MP> printenv fdt_addr_r
fdt_addr_r=0xc4000000

STM32MP> ext4load mmc 0:4 ${kernel_addr_r} /boot/zImage
4202992 bytes read in 207 ms (19.4 MiB/s)
STM32MP> ext4load mmc 0:4 ${fdt_addr_r} /boot/stm32mp157a-dk1.dtb
53881 bytes read in 31 ms (1.7 MiB/s)

STM32MP> bootz ${kernel_addr_r} - ${fdt_addr_r}
Booting Linux on STM32MP157A-DK1

U-Boot 2020.07 (Feb 04 2021 - 16:27:10 +0100)
CPU: STM32MP157AAC Rev.B
Model: STMicroelectronics STM32MP157A-DK1 Discovery Board

STM32MP> printenv kernel_addr_r
kernel_addr_r=0xc2000000
STM32MP> printenv fdt_addr_r
fdt_addr_r=0xc4000000

STM32MP> ext4load mmc 0:4 ${kernel_addr_r} /boot/zImage
4202992 bytes read in 207 ms (19.4 MiB/s)
STM32MP> ext4load mmc 0:4 ${fdt_addr_r} /boot/stm32mp157a-dk1.dtb
53881 bytes read in 31 ms (1.7 MiB/s)

STM32MP> bootz ${kernel_addr_r} - ${fdt_addr_r}

Kernel image @ 0xc2000000 [ 0x000000 - 0x4021f0 ]
## Flattened Device Tree blob at c4000000
Booting using the fdt blob at 0xc4000000
Loading Device Tree to cffef000, end cffff278 ... OK

Starting kernel ...

[ 0.000000] Linux version 5.8.13 (thomas@windsurf) (arm-none-linux-gnueabihf-gcc....)
[ 0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387d
...
[ 0.000000] OF: fdt: Machine model: STMicroelectronics STM32MP157A-DK1 Discovery Board
Exploring the DT on the target

- In `/sys/firmware/devicetree/base`, there is a directory/file representation of the Device Tree contents.

```bash
# ls -l /sys/firmware/devicetree/base/
total 0
-r--r--r-- 1 root root 4 Jan 1 00:00 #address-cells
-r--r--r-- 1 root root 4 Jan 1 00:00 #size-cells
drwxr-xr-x 2 root root 0 Jan 1 00:00 chosen
drwxr-xr-x 3 root root 0 Jan 1 00:00 clocks
-r--r--r-- 1 root root 34 Jan 1 00:00 compatible
 [...]  
-r--r--r-- 1 root root 1 Jan 1 00:00 name
drwxr-xr-x 10 root root 0 Jan 1 00:00 soc
```

- If `dtc` is available on the target, possible to ”unpack” the Device Tree using:
  
  `dtc -I fs /sys/firmware/devicetree/base`
Device Tree base syntax

- **Tree of nodes**
- **Nodes with properties**
- **Node ≈ a device or IP block**
- **Properties ≈ device characteristics**
- **Notion of cells in property values**
- **Notion of phandle to point to other nodes**
- **dtc only does syntax checking, no semantic validation**
DT overall structure: simplified example

```c
/
#address-cells = <1>;
#size-cells = <1>;
model = "STMicroelectronics STM32MP157C-DK2 Discovery Board";
compatible = "st,stm32mp157c-dk2", "st,stm32mp157";

cpus { ... };
memory@0 { ... };
chosen { ... };
intc: interrupt-controller@a0021000 { ... };
soc {
  i2c1: i2c@40012000 { ... };
  ethernet0: ethernet@5800a000 { ... };
};
```

System-on-chip

CPU cores

Cortex A7  
Cortex A7

ST Ethernet MAC  
ST I2C controller  
DDR controller

GIC  
IRQ controller

Ethernet PHY  
CS42L51 audio codec  
DDR memory
DT overall structure: simplified example

/ {
  cpus {
    #address-cells = <1>;
    #size-cells = <0>;
    cpu0: cpu00 {
      compatible = "arm,cortex-a7";
      clock-frequency = <650000000>;
      device_type = "cpu";
      reg = <0>;
    };
    cpu1: cpu01 {
      compatible = "arm,cortex-a7";
      clock-frequency = <650000000>;
      device_type = "cpu";
      reg = <1>;
    };
  };
  memory@0 { ... };
  chosen { ... };
  intc: interrupt-controller@a0021000 { ... };
  soc {
    i2c1: i2c@40012000 { ... };
    ethernet0: ethernet@5800a000 { ... };
  };
};
DT overall structure: simplified example

```c
/ {
    cpus { ... }
    memory@0 {
        device_type = "memory";
        reg = <0x0 0x20000000>;
    };

    chosen {
        bootargs = "";
        stdout-path = "serial0:115200n8";
    };
    intc: interrupt-controller@a0021000 { ... };
    soc {
        i2c1: i2c@40012000 { ... };
        ethernet0: ethernet@5800a000 { ... };
    };
};
```
/ {
    cpus { ... };
    memory@0 { ... };
    chosen { ... };

    intc: interrupt-controller@0021000 {
        compatible = "arm,cortex-a7-gic";
        #interrupt-cells = <3>;
        interrupt-controller;
        reg = <0xa0021000 0x1000>,
            <0xa0022000 0x2000>;
    };

    soc {
        compatible = "simple-bus";
        #address-cells = <1>;
        #size-cells = <1>;
        interrupt-parent = <&intc>;

        i2c1: i2c@40012000 { ... };
        ethernet0: ethernet@5800a000 { ... };
    };
};
DT overall structure: simplified example

```c
/ {
  cpus { ... };
  memory@0 { ... };
  chosen { ... };
  intc: interrupt-controller@a0021000 { ... };
  soc {
    i2c1: i2c@40012000 {
      compatible = "st,stm32mp15-i2c";
      reg = <0x40012000 0x400>;
      interrupts = <GIC_SPI 31 IRQ_TYPE_LEVEL_HIGH>,
                   <GIC_SPI 32 IRQ_TYPE_LEVEL_HIGH>;
      #address-cells = <1>;
      #size-cells = <0>;
      status = "okay";
    }
    cs42l51: cs42l51@4a {
      compatible = "cirrus,cs42l51";
      reg = <0x4a>;
      reset-gpios = <&gpiog 9 GPIO_ACTIVE_LOW>;
      status = "okay";
    };
    ethernet0: ethernet@5800a000 { ... };
  };
};
```
DT overall structure: simplified example

```c
/
  cpus { ... };
memory@0 { ... };
chosen { ... };
intc: interrupt-controller@0021000 { ... };
soc {
  compatible = "simple-bus";
  ...
  interrupt-parent = <&intc>;
i2c1: i2c@40012000 { ... };
ethernet0: ethernet@5800a000 {
  compatible = "st,stm32mp1-dmac", "snps,dwmac-4.20a";
  reg = <0x5800a000 0x2000>;
  interrupts-extended = <&intc GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>;
  status = "okay";
  mdio0 {
    #address-cells = <1>;
    #size-cells = <0>;
    compatible = "snps,dwmac-mdio";
    phy0: ethernet-phy@0 {
      reg = <0>;
    };
  };
};
};
```
Device Tree inheritance

- Device Tree files are not monolithic, they can be split in several files, including each other.
- .dtsi files are included files, while .dts files are final Device Trees
  - Only .dts files are accepted as input to dtc
- Typically, .dtsi will contain
  - definitions of SoC-level information
  - definitions common to several boards
- The .dts file contains the board-level information
- The inclusion works by overlaying the tree of the including file over the tree of the included file.
- Allows an including file to override values specified by an included file
- Uses the C pre-processor #include directive
Device Tree inheritance example

Definition of the STM32MP157A SoC

```c
/ {
  soc {
    i2c1: i2c@40012000 {
      compatible = "st,stm32mp15-i2c";
      reg = <0x40012000 0x400>
      interrupts = <GIC_SPI 31 IRQ...HIGH>,
                  <GIC_SPI 32 IRQ...HIGH>
      status = "disabled";
    }
  }
}
```

STM32MP157.dtsi

Definition of the STM32MP157A-DK1 board

```c
#include "stm32mp157.dtsi"
/ {
  soc {
    i2c1: i2c@40012000 {
      compatible = "st,stm32mp15-i2c";
      reg = <0x40012000 0x400>
      interrupts = <GIC_SPI 31 IRQ...HIGH>,
                  <GIC_SPI 32 IRQ...HIGH>
      pinctrl-names = "default", "sleep";
      pinctrl-0 = <&i2c1_pins_a>;
      pinctrl-1 = <&i2c1_sleep_pins_a>;
      status = "okay";
    }
  }
}
```

STM32MP157A-dk1.dts

Note 1

The actual Device Trees for this platform are more complicated. This example is highly simplified.

Compiled DTB

```c
/ {
  soc {
    i2c1: i2c@40012000 {
      compatible = "st,stm32mp15-i2c";
      reg = <0x40012000 0x400>
      interrupts = <GIC_SPI 31 IRQ...HIGH>,
                  <GIC_SPI 32 IRQ...HIGH>
      pinctrl-names = "default", "sleep";
      pinctrl-0 = <&i2c1_pins_a>;
      pinctrl-1 = <&i2c1_sleep_pins_a>;
      status = "okay";
    }
  }
}
```

STM32MP157A-dk1.dtb

Note 2

The real DTB is in binary format. Here we show the text equivalent of the DTB contents.
Inheritance and labels

Doing:

```
soc.dtsi
/
{
    soc {
        usart1: serial@5c000000 {
            compatible = "st,stm32h7-uart";
            reg = <0x5c000000 0x400>;
            status = "disabled";
        };
    };
};
```

```
board.dts
#include "soc.dtsi"
/
{
    soc {
        serial@5c000000 {
            status = "okay";
        };
    };
};
```
Inheritance and labels

Doing:

```
soc.dtsi
{/ 
  soc {
    usart1: serial@5c000000 {
      compatible = "st,stm32h7-uart";
      reg = <0x5c000000 0x400>;
      status = "disabled";
    }
  }
};
board.dts
#include "soc.dtsi"
{/ 
  soc {
    serial@5c000000 {
      status = "okay";
    }
  }
};
```

Is exactly equivalent to:

```
soc.dtsi
{/ 
  soc {
    usart1: serial@5c000000 {
      compatible = "st,stm32h7-uart";
      reg = <0x5c000000 0x400>;
      status = "disabled";
    }
  }
};
```

```
board.dts
#include "soc.dtsi"
%usart1 {
  status = "okay";
}
```

→ this solution is now often preferred
DT inheritance in STM32MP1 support

stm32mp151.dtsi
Single Cortex-A7
All common peripherals

stm32mp153.dtsi
#include stm32mp151.dtsi
+ second Cortex-A7
+ CAN

stm32mp157.dtsi
#include stm32mp153.dtsi
+ GPU + DSI

stm32mp15xc.dtsi
+ Crypto

stm32mp15-pinctrl.dtsi

stm32mp15xxaa-pinctrl.dtsi
stm32mp15xxab-pinctrl.dtsi
stm32mp15xxac-pinctrl.dtsi
stm32mp15xxad-pinctrl.dtsi

stm32mp157a-dk1.dts
Definitions
specific to the DK1 board
#include stm32mp157.dtsi
#include stm32mp15-pinctrl.dtsi
#include stm32mp15xxac-pinctrl.dtsi
#include stm32mp15xx-dkx.dtsi

stm32mp15xx-dkx.dts
Definitions common
to the DK1 and DK2 boards

stm32mp157c-dk2.dts
Definitions
specific to the DK2 board
#include stm32mp157.dtsi
#include stm32mp15xx.dtsi
#include stm32mp15-pinctrl.dtsi
#include stm32mp15xxac-pinctrl.dtsi
#include stm32mp15xx-dkx.dtsi
Device Tree design principles

- **Describe hardware** (how the hardware is), not configuration (how I choose to use the hardware)
- **OS-agnostic**
  - For a given piece of HW, Device Tree should be the same for U-Boot, FreeBSD or Linux
  - There should be no need to change the Device Tree when updating the OS
- **Describe integration of hardware components**, not the internals of hardware components
  - The details of how a specific device/IP block is working is handled by code in device drivers
  - The Device Tree describes how the device/IP block is connected/integrated with the rest of the system: IRQ lines, DMA channels, clocks, reset lines, etc.

- Like all beautiful design principles, these principles are sometimes violated.
Device Tree specifications

- How to write the correct nodes/properties to describe a given hardware platform?
- **Device Tree Specifications** → base Device Tree syntax + number of standard properties.
  - [https://www.devicetree.org/specifications/](https://www.devicetree.org/specifications/)
  - Not sufficient to describe the wide variety of hardware.
- **Device Tree Bindings** → documents that each specify how a piece of HW should be described
  - Documentation/devicetree/bindings/ in Linux kernel sources
  - Reviewed by DT bindings maintainer team
  - Legacy: human readable documents
  - New norm: YAML-written specifications
Device Tree binding: old style

Documentation/devicetree/bindings/mtd/spear_smi.txt

This IP is not used on STM32MP1.

* SPEAr SMI

Required properties:
- compatible : "st,spear600-smi"
- reg : Address range of the mtd chip
- #address-cells, #size-cells : Must be present if the device has sub-nodes representing partitions.
- interrupts: Should contain the STMMAC interrupts
- clock-rate : Functional clock rate of SMI in Hz

Optional properties:
- st,smi-fast-mode : Flash supports read in fast mode

Example:

```c
smi: flash@fc000000 {
  compatible = "st,spear600-smi";
  #address-cells = <1>;
  #size-cells = <1>;
  reg = <0xfc000000 0x1000>;
  interrupt-parent = <&vic1>;
  interrupts = <12>;
  clock-rate = <50000000>; /* 50MHz */

  flash@f8000000 {
    st,smi-fast-mode;
    ...
  };
}
```
Device Tree binding: YAML style

Documentation/devicetree/bindings/i2c/st,stm32-i2c.yaml

# SPDX-License-Identifier: (GPL-2.0-only OR BSD-2-Clause)

%YAML 1.2

---
$id: http://devicetree.org/schemas/i2c/st,stm32-i2c.yaml#
$schema: http://devicetree.org/meta-schemas/core.yaml#

title: I2C controller embedded in STMicroelectronics STM32 I2C platform

maintainers:
- Pierre-Yves MORDRET <pierre-yves.mordret@st.com>

properties:
  compatible:
    enum:
    - st,stm32f4-i2c
    - st,stm32f7-i2c
    - st,stm32mp15-i2c

  reg:
    maxItems: 1

  interrupts:
    items:
    - description: interrupt ID for I2C event
    - description: interrupt ID for I2C error

  resets:
    maxItems: 1

  clocks:
    maxItems: 1

  dmas:
    items:
    - description: RX DMA Channel phandle
    - description: TX DMA Channel phandle

  clock-frequency:
    description: Desired I2C bus clock frequency in Hz. If not specified, the default 100 kHz frequency will be used. For STM32F7, STM32H7 and STM32MP1 SoCs, if timing parameters match, the bus clock frequency can be from 1Hz to 1MHz.
    default: 100000
    minimum: 1
    maximum: 1000000

required:
- compatible
- reg
- interrupts
- resets
- clocks
Device Tree binding: YAML style example

examples:
- |
  //Example 3 (with st,stm32mp15-i2c compatible on stm32mp)
  #include <dt-bindings/interrupt-controller/arm-gic.h>
  #include <dt-bindings/clock/stm32mp1-clks.h>
  #include <dt-bindings/reset/stm32mp1-resets.h>
  i2c@40013000 {
    compatible = "st,stm32mp15-i2c";
    #address-cells = <1>;
    #size-cells = <0>;
    reg = <0x40013000 0x400>;
    interrupts = <GIC_SPI 33 IRQ_TYPE_LEVEL_HIGH>,
                  <GIC_SPI 34 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&rcc I2C2_K>;
    resets = <&rcc I2C2_R>;
    i2c-scl-rising-time-ns = <185>;
    i2c-scl-falling-time-ns = <20>;
    st,syscfg-fmp = <&syscfg 0x4 0x2>;
  };

- Kernel, drivers and embedded Linux - Development, consulting, training and support - https://bootlin.com
dtc only does syntactic validation

YAML bindings allow to do semantic validation

Linux kernel make rules:

- make dt_binding_check
  verify that YAML bindings are valid
- make dtbs_check
  validate DTs currently enabled against YAML bindings
- make DT_SCHEMA_FILES=Documentation/devicetree/bindings/trivial-devices.yaml dtbs_check
  validate DTs against a specific YAML binding
The compatible property

- Is a list of strings
  - From the most specific to the less specific
- Describes the specific binding to which the node complies.
- It uniquely identifies the programming model of the device.
- Practically speaking, it is used by the operating system to find the appropriate driver for this device.
- When describing real hardware, typical form is vendor,model

Examples:

- compatible = "arm,armv7-timer";
- compatible = "st,stm32mp1-dmac", "snps,dmac-4.20a";
- compatible = "regulator-fixed";
- compatible = "gpio-keys";

- Special value: simple-bus → bus where all sub-nodes are memory-mapped devices
Top-level DT nodes with a compatible property and nodes that are sub-nodes of simple-bus will cause Linux to identify those devices as platform devices

- Instantiated automatically at boot time

Sub-nodes of I2C controllers → I2C devices

Sub-nodes of SPI controllers → SPI devices

Each Linux driver has a table of compatible strings it supports

- struct of_device_id []

When a DT node compatible string matches a given driver, the device is bound to that driver.
drivers/tty/serial/stm32-usart.c

```c
static const struct of_device_id stm32_match[] = {
    {.compatible = "st,stm32-uart", .data = &stm32f4_info},
    {.compatible = "st,stm32f7-uart", .data = &stm32f7_info},
    {.compatible = "st,stm32h7-uart", .data = &stm32h7_info},
    {},
};
MODULE_DEVICE_TABLE(of, stm32_match);

static struct platform_driver stm32_serial_driver = {
    .probe = stm32_serial_probe,
    .remove = stm32_serial_remove,
    .driver = {
        .name = DRIVER_NAME,
        .pm = &stm32_serial_pm_ops,
        .of_match_table = of_match_ptr(stm32_match),
    },
};
```
Matching with drivers in Linux: I2C driver

```c
const struct of_device_id cs42l51_of_match[] = {
    { .compatible = "cirrus,cs42l51", },
    {}
};
MODULE_DEVICE_TABLE(of, cs42l51_of_match);

static struct i2c_driver cs42l51_i2c_driver = {
    .driver = {
        .name = "cs42l51",
        .of_match_table = cs42l51_of_match,
        .pm = &cs42l51_pm_ops,
    },
    .probe = cs42l51_i2c_probe,
    .remove = cs42l51_i2c_remove,
    .id_table = cs42l51_i2c_id,
};
```
Most important property after compatible

Memory-mapped devices: base physical address and size of the memory-mapped registers. Can have several entries for multiple register areas.

```c
sai4: sai@50027000 {
  reg = <0x50027000 0x4>, <0x500273f0 0x10>;
};
```
reg property

- Most important property after `compatible`
- **Memory-mapped** devices: base physical address and size of the memory-mapped registers. Can have several entries for multiple register areas.
- **I2C** devices: address of the device on the I2C bus.

```c
&i2c1 {
    hdmi-transmitter@39 {
        reg = <0x39>;
    };
    cs42l51: cs42l51@4a {
        reg = <0x4a>;
    };
};
```
reg property

- Most important property after compatible
- Memory-mapped devices: base physical address and size of the memory-mapped registers. Can have several entries for multiple register areas.
- I2C devices: address of the device on the I2C bus.
- SPI devices: chip select number

```c
&qspi {
  flash0: mx66l512351@0 {
    reg = <0>;
  };
  flash1: mx66l512351@1 {
    reg = <1>;
  };
};
```
reg property

- Most important property after `compatible`
- **Memory-mapped** devices: base physical address and size of the memory-mapped registers. Can have several entries for multiple register areas.
- **I2C** devices: address of the device on the I2C bus.
- **SPI** devices: chip select number
- The unit address must be the address of the first `reg` entry.

```c
sai4: sai@50027000 {
    reg = <0x50027000 0x4>, <0x500273f0 0x10>;
};
```
Cells concept

- Integer values represented as 32-bit integers called cells

```c
soc {
    /* This property has 1 cell */
    foo = <0xdeadbeef>;
};
```
Cells concept

- Integer values represented as 32-bit integers called cells
- Encoding a 64-bit value requires two cells

```c
soc {
    /* This property has 2 cells */
    foo = <0xdeadbeef 0xbadcafe>;
};
```
Cells concept

- Integer values represented as 32-bit integers called cells
- Encoding a 64-bit value requires two cells
- `#address-cells` and `#size-cells`: how many cells are used in sub-nodes to encode the address and size in the `reg` property

```plaintext
soc {
    compatible = "simple-bus";
    #address-cells = <1>;
    #size-cells = <1>;

dump@f1001000 {
    reg = <0xf1001000 0x1000>;
    #address-cells = <1>;
    #size-cells = <0>;
}
}
```
Cells concept

- Integer values represented as 32-bit integers called cells
- Encoding a 64-bit value requires two cells
- #address-cells and #size-cells: how many cells are used in sub-nodes to encode the address and size in the reg property
- #interrupts-cells: how many cells are used to encode interrupt specifiers for this interrupt controller

```c
soc {
    intc: interrupt-controller@f1002000 {
        compatible = "foo,bar-intc";
        reg = <0xf1002000 0x1000>;
        interrupt-controller;
        #interrupt-cells = <2>;
    };

    i2c@f1001000 {
        interrupt-parent = <&intc>;
        /* Must have two cells */
        interrupts = <12 24>;
    };
}
```
Cells concept

- Integer values represented as 32-bit integers called cells
- Encoding a 64-bit value requires two cells
- #address-cells and #size-cells: how many cells are used in sub-nodes to encode the address and size in the reg property
- #interrupts-cells: how many cells are used to encode interrupt specifiers for this interrupt controller
- Ditto #clock-cells, #gpio-cells, #phy-cells, #pwm-cells, #dma-cells, etc.
- The **status** property indicates if the device is really in use or not.
  - **okay** or **ok** → the device is really in use.
  - Any other value, by convention **disabled** → the device is not in use.

- In Linux, it controls if a device is instantiated.

- In `.dtsi` files describing SoCs: all devices that interface to the outside world have **status** = **disabled**.

- Enabled on a per-device basis in the board `.dts`.
Nodes describing **interrupt controllers**

- Devices like any other
- interrupt-controller boolean

Devices using interrupts:

- **interrupts + interrupt-parent**, **interrupts** specifies the interrupt number and flags, **interrupt-parent** the interrupt controller, searched recursively in the parent nodes
- **interrupts-extended** specifies the interrupt controller, interrupt number and flags

```c
/ {
    intc: interrupt-controller@a0021000 {
        compatible = "arm,cortex-a7-gic";
        #interrupt-cells = <3>;
        interrupt-controller;
        reg = <0xa0021000 0x1000>,
            <0xa0022000 0x2000>;
    }
    soc {
        interrupt-parent = <&intc>;
        spi2: spi@4000b000 {
            interrupts = <GIC_SPI 36 IRQ_TYPE_LEVEL_HIGH>;
            ...}
    }
    ipcc: mailbox@4c001000 {
        interrupts-extended =
            <&intc GIC_SPI 100 IRQ_TYPE_LEVEL_HIGH>,
            <&intc GIC_SPI 101 IRQ_TYPE_LEVEL_HIGH>,
            <&exti 61 1>;
            ...}
};
```
Other resources: clocks, DMA, reset lines, ...

- Similar pattern for other resources shared by multiple hardware blocks
  - Clock controllers
  - DMA controllers
  - Reset controllers
  - ...

- A Device Tree node describing the *controller* as a device

- References from other nodes that use resources provided by this *controller*
Some properties are associated to a corresponding `<prop>-names` property.

Gives some human-readable names to entries of the corresponding `<prop>` properties.

```c
interrupts = <0 59 0>, <0 70 0>;
interrupt-names = "macirq", "macpmt";
clocks = <&car 39>, <&car 45>, <&car 86>, <&car 87>;
clock-names = "gnssm_rgmii", "gnssm_gmac", "rgmii", "gmac";
```

Such names can be typically be used by the driver:

```c
platform_get_irq_byname(pdev, "macirq");
```
Most modern SoCs, including the STM32MP1, have more features than they have pins to expose those features to the outside world.

Pins are muxed: a given pin can be used for one function or another.

A specific IP block in the SoC controls the muxing of pins: the pinmux controller.

The Device Tree describes which pin configurations are possible, and which configurations are used by the different devices.
arch/arm/boot/dts/stm32mp151.dtsi

pinctrl: pin-controller@50002000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "st,stm32mp157-pinctrl";

    gpioa: gpio@50002000 { ... };  
gpiob: gpio@50003000 { ... };  
gpioc: gpio@50004000 { ... };  
gpiod: gpio@50005000 { ... };  
gpioe: gpio@50006000 { ... };  
gpiof: gpio@50007000 { ... };  
...  
};

pinctrl_z: pin-controller-z@54004000 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "st,stm32mp157-z-pinctrl";
    ranges = <0 0x54004000 0x400>;

    gpioz: gpio@54004000 { .... };  
};
&pinctrl {
  i2c1_pins_a: i2c1-0 {
    pins {
      pinmux = <STM32_PINMUX('D', 12, AF5)>, /* I2C1_SCL */
      <STM32_PINMUX('F', 15, AF5)>; /* I2C1_SDA */
      bias-disable;
      drive-open-drain;
      slew-rate = <0>;
    }
  }
}

m_can1_pins_a: m-can1-0 {
  pins1 {
    pinmux = <STM32_PINMUX('H', 13, AF9)>; /* CAN1_TX */
    slew-rate = <1>;
    drive-push-pull;
    bias-disable;
  }
  pins2 {
    pinmux = <STM32_PINMUX('I', 9, AF9)>; /* CAN1_RX */
    bias-disable;
  }
};
## Pin-muxing configuration

<table>
<thead>
<tr>
<th>Port</th>
<th>AF0</th>
<th>AF1</th>
<th>AF2</th>
<th>AF3</th>
<th>AF4</th>
<th>AF5</th>
<th>AF6</th>
<th>AF7</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD6</td>
<td>-</td>
<td>TIM16_CH1N</td>
<td>SAI1_D1</td>
<td>DFSDM1_CLKIN4</td>
<td>DFSDM1_DATIN1</td>
<td>SPI3_MOSI/ I2S3_SDO</td>
<td>SAI1_SD_A</td>
<td>USART2_RX</td>
</tr>
<tr>
<td>PD7</td>
<td>TRACED6</td>
<td>-</td>
<td>-</td>
<td>DFSDM1_DATIN4</td>
<td>I2C2_SCL</td>
<td>-</td>
<td>DFSDM1_CLKIN1</td>
<td>USART2_CK</td>
</tr>
<tr>
<td>PD8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DFSDM1_CLKIN3</td>
<td>-</td>
<td>-</td>
<td>SAI3_SCK_B</td>
<td>USART3_TX</td>
</tr>
<tr>
<td>PD9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>DFSDM1_DATIN3</td>
<td>-</td>
<td>-</td>
<td>SAI3_SD_B</td>
<td>USART3_RX</td>
</tr>
<tr>
<td>PD10</td>
<td>RTC_REFIN</td>
<td>TIM16_BKIN</td>
<td>-</td>
<td>DFSDM1_CKOUT</td>
<td>I2C5_SMBA</td>
<td>SPI3_MISO/ I2S3_SDI</td>
<td>SAI3_FS_B</td>
<td>USART3_CK</td>
</tr>
<tr>
<td>PD11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>LPTIM2_IN2</td>
<td>I2C4_SMBA</td>
<td>I2C1_SMBA</td>
<td>-</td>
<td>USART3_GTS/ USART3_NSS</td>
</tr>
<tr>
<td>PD12</td>
<td>-</td>
<td>LPTIM1_IN1</td>
<td>TIM4_CH1</td>
<td>LPTIM2_IN1</td>
<td>I2C4_SCL</td>
<td>I2C1_SCL</td>
<td>-</td>
<td>USART3_RTS/ USART3_DE</td>
</tr>
<tr>
<td>PD13</td>
<td>-</td>
<td>LPTIM1_OUT</td>
<td>TIM4_CH2</td>
<td>-</td>
<td>I2C4_SDA</td>
<td>I2C1_SDA</td>
<td>I2S3_MCK</td>
<td>-</td>
</tr>
<tr>
<td>PD14</td>
<td>-</td>
<td>TIM4_CH3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SAI3_MCLK_B</td>
<td>-</td>
</tr>
<tr>
<td>PD15</td>
<td>-</td>
<td>TIM4_CH4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>SAI3_MCLK_A</td>
<td>-</td>
</tr>
</tbody>
</table>
Typically board-specific, in .dts

pinctrl-0, pinctrl-1, pinctrl-X provides the pin mux configurations for the different states

pinctrl-names gives a name to each state, mandatory even if only one state

States are mutually exclusive

Driver is responsible for switching between states

default state is automatically set up when the device is probed
Example: LED and I2C device

- Let’s see how to describe an LED and an I2C device connected to the DK1 platform.

- Create `arch/arm/boot/dts/stm32mp157a-dk1-custom.dts` which includes `stm32mp157a-dk1.dts`

```c
#include "stm32mp157a-dk1.dts"
```

- Make sure `stm32mp157a-dk1-custom.dts` gets compiled to a DTB by changing `arch/arm/boot/dts/Makefile`

```makefile
dtb-$(CONFIG_ARCH_STM32) += \
   ...
   stm32mp157a-dk1.dtb \
   stm32mp157a-dk1-custom.dtb
```

- make dtbs

```bash
DTC arch/arm/boot/dts/stm32mp157a-dk1-custom.dtb
```
Example: describe an LED

```c
#include "stm32mp157a-dk1.dts"

/ {
    leds {
        compatible = "gpio-leds";
        webinar {
            label = "webinar";
            gpios = <&gpioe 1 GPIO_ACTIVE_HIGH>;
        }
    }
};
```

shell

```
# echo 255 > /sys/class/leds/webinar/brightness
```
Example: connect I2C sensor

```c
stm32mp157a-dk1-custom.dts

&i2c5 {
  status = "okay";
  clock-frequency = <100000>;
  pinctrl-names = "default", "sleep";
  pinctrl-0 = <&i2c5_pins_a>;
  pinctrl-1 = <&i2c5_pins_sleep_a>;

  pressure@76 {
    compatible = "bosch,bme280";
    reg = <0x76>;
  }
};
```

shell

```
# cat /sys/bus/iio/devices/iio:device2/in_temp_input
24380
```

Details at https://bootlin.com/blog/building-a-linux-system-for-the-stm32mp1-connecting-an-i2c-sensor/
There’s more!

- Lots of Device Tree topics not covered in this talk
- `range` property for address translation
- Complex Device Tree bindings
  - Audio, display, camera devices
  - PCIe
- Linux kernel API for DT
- U-Boot tooling for DT manipulation
- DT overlays
- etc.
Resources

- Device Tree specifications
  https://www.devicetree.org/
- Device Tree bindings
  https://elixir.bootlin.com/linux/latest/source/Documentation/devicetree/bindings
- *Device Tree for Dummies* talk
  https://www.youtube.com/watch?v=uzBwHFjOvU
- eLinux.org wiki page on Device Tree
  https://elinux.org/Device_Tree_Reference
- Numerous Device Tree talks at the Embedded Linux Conference
  https://elinux.org/Device_Tree_Presentations
Conclusion: about the Device Tree

- Representation of non-discoverable hardware
- Tree of nodes, with properties
- Standardization based on Device Tree bindings
- New description language with lots of properties and sometimes complex bindings
- Used for numerous CPU architectures
- Now widely used outside of Linux
- A must know for all embedded Linux developers!
Conclusion: about Bootlin

- Expertise in embedded Linux
- Training
- Engineering services
- Linux BSP development
- Kernel drivers
- Open-source contributor
- Contact us!

info@bootlin.com

Thanks to ST for supporting this webinar!
Questions?

https://bootlin.com