



Devicetree Addons: Describe hot-pluggable extension boards

Hervé Codina
herve.codina@bootlin.com

Linux Plumbers 2026

© Copyright 2004-2026, Bootlin.
Creative Commons BY-SA 3.0 license.
Corrections, suggestions, contributions and translations are welcome!





Whoami

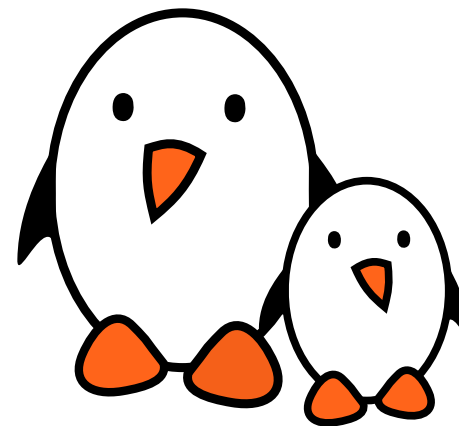
- ▶ Hervé Codina
- ▶ Embedded Linux engineer at [Bootlin](#)
 - Engineering company specialized in **Embedded Linux** and **Zephyr**
 - 30 people, mostly in France
 - Very strong open-source focus
- ▶ Contributor of addon support in libfdt and DTC (devicetree compiler)
- ▶ Living in Toulouse, France





Goal

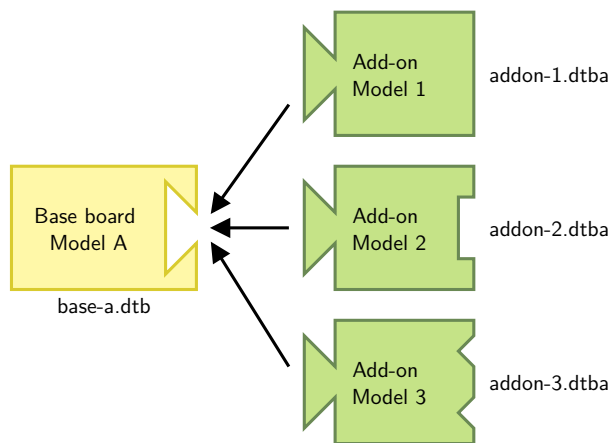
bootlin



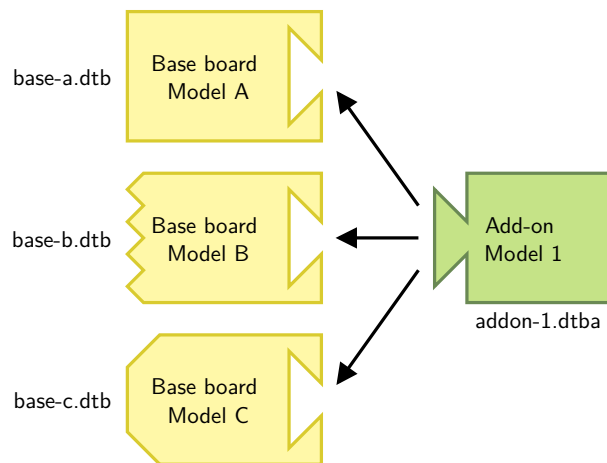


Goal

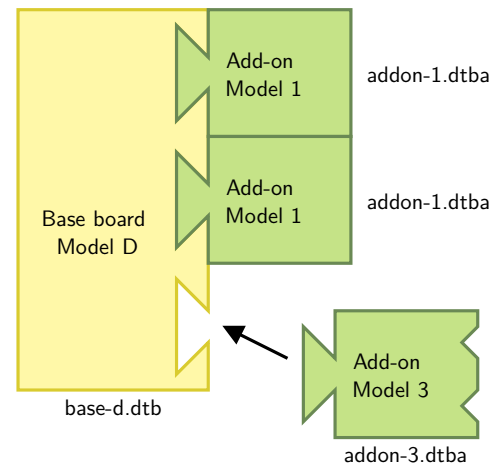
- ▶ Describe an **extension board** (addon) using **devicetree description**.
- ▶ Use **devicetree blobs** (base board and addons dtbs) **without any modifications**
 - **Case 1:** Different addons connectable to a base board
 - **Case 2:** Same addon connected to different base boards
 - **Case 3:** Same addon connected to different connectors of a base board



Case 1



Case 2



Case 3



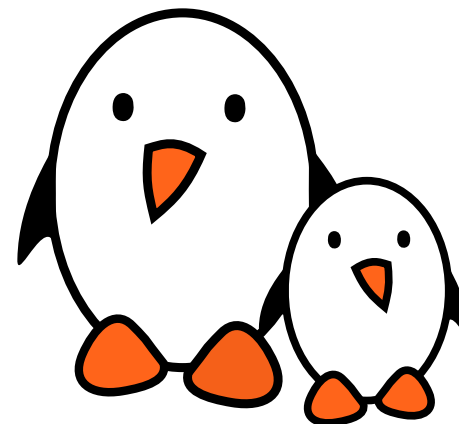
Devicetree connector description

- ▶ Describe the **connector** in the base devicetree
 - Provide information needed by an addon in this connector node
 - **Contract** related to resources between the **base board** and the **extension board**
 - Apply the addon dtb at the connector node
 - Symbol **resolution** based on **exported** / **imported** symbols



Ongoing work

bootlin



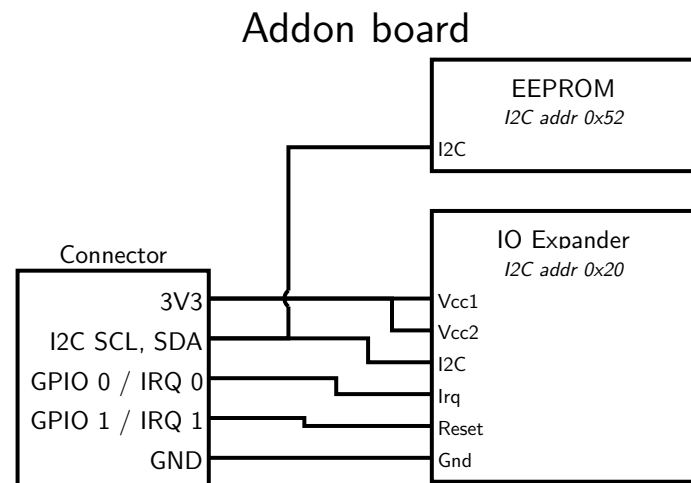
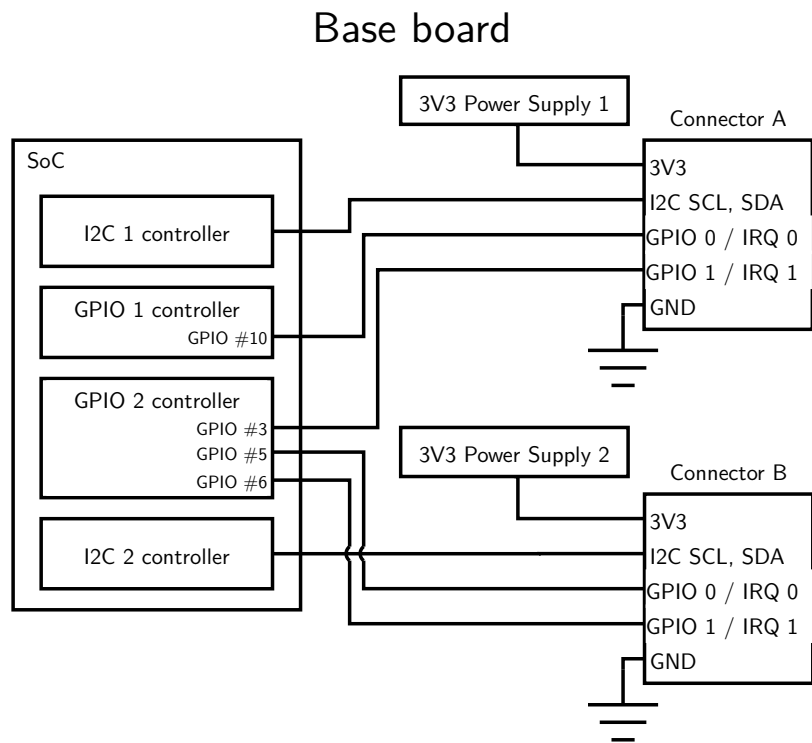


Ongoing contribution

- ▶ libfdt/dtc:
 - [PATCH v3 00/15] Add support for structured tags and v18 dtb version
 - [RFC PATCH v2 00/74] Add support for dtb metadata and addon device-trees
- ▶ Bootlin blog post:
 - Device tree addons: describing non-discoverable addon boards and connectors



Hardware example





Base devicetree

Base dts

```

regul1_3v3: regul1 { /* 3V3 Power Supply 1 */
    compatible = "fixed-regulator";
    ...
};
regul2_3v3: regul2 { /* 3V3 Power Supply 2 */
    compatible = "fixed-regulator";
    ...
};

soc {
    i2c1: i2c@1000 { /* I2C 1 Controller*/
        compatible = "soc-vendor,i2c-ctrl";
        ...
    };
    i2c2: i2c@2000 { /* I2C 2 Controller*/
        compatible = "soc-vendor,i2c-ctrl";
        ...
    };
    gpio1: gpio@4000 { /* GPIO 1 Controller */
        compatible = "soc-vendor,gpio-ctrl";
        gpio-controller;
        #gpio-cells = <0x02>;
        interrupt-controller;
        #interrupt-cells = <0x02>;
        #address-cells = <0x00>;
        ...
    };
    gpio2: gpio@5000 { /* GPIO 2 Controller */
        compatible = "soc-vendor,gpio-ctrl";
        gpio-controller;
        #gpio-cells = <0x02>;
        interrupt-controller;
        #interrupt-cells = <0x02>;
        #address-cells = <0x00>;
        ...
    };
};

```

Base dts (continued)

```

conn_a: conn-a {
    compatible = "vendor,connector";
    /* Nexus node definition for gpios */
    #gpio-cells = <2>;
    gpio-map-mask = <0xf 0x0>;
    gpio-map-pass-thru = <0x0 0xf>;
    gpio-map = <0 0 &gpio1 10 0>, <1 0 &gpio2 3 0>;
    /* Nexus node definition for interrupts */
    #address-cells = <0>;
    #interrupt-cells = <2>;
    interrupt-map = <0 IRQ_TYPE_LEVEL_LOW &gpio1 10 IRQ_TYPE_LEVEL_LOW>,
        <0 IRQ_TYPE_LEVEL_HIGH &gpio1 10 IRQ_TYPE_LEVEL_HIGH>,
        <1 IRQ_TYPE_LEVEL_LOW &gpio2 3 IRQ_TYPE_LEVEL_LOW>,
        <1 IRQ_TYPE_LEVEL_HIGH &gpio2 3 IRQ_TYPE_LEVEL_HIGH>;

    /* Export resources */
    /export/ conn: &conn_a; /* The connector itself */
    /export/ supply_3v3: &regul1_3v3; /* The 3v3 at Connector A */
    /export/ i2c: &i2c1; /* The I2C at Connector A */
    /export/ gpios: &conn_a; /* The Nexus node for GPIOs */
    /export/ irq: &conn_a; /* The Nexus node for IRQs */
};

conn_b: conn-b {
    compatible = "vendor,connector";
    #gpio-cells = <2>;
    gpio-map-mask = <0xf 0x0>;
    gpio-map-pass-thru = <0x0 0xf>;
    gpio-map = <0 0 &gpio2 5 0>, <1 0 &gpio2 6 0>;
    #address-cells = <0>;
    #interrupt-cells = <2>;
    interrupt-map = <0 IRQ_TYPE_LEVEL_LOW &gpio2 5 IRQ_TYPE_LEVEL_LOW>,
        <0 IRQ_TYPE_LEVEL_HIGH &gpio2 5 IRQ_TYPE_LEVEL_HIGH>,
        <1 IRQ_TYPE_LEVEL_LOW &gpio2 6 IRQ_TYPE_LEVEL_LOW>,
        <1 IRQ_TYPE_LEVEL_HIGH &gpio2 6 IRQ_TYPE_LEVEL_HIGH>;

    /export/ conn: &conn_b;
    /export/ supply_3v3: &regul2_3v3;
    /export/ i2c: &i2c2;
    /export/ gpios: &conn_b;
    /export/ irq: &conn_b;
};

```



Addon dtsa

```
/dts_v1/  
/addon/  
/*  
 * Import only the connector and use namespace  
 * symbol references  
 */  
/import/ conn: "";  
  
&conn.i2c {  
    io-expander@20 {  
        compatible = "foo,io-expander";  
        reg = <0x20>;  
        vcc1-supply = <&conn.supply_3v3>;  
        vcc2-supply = <&conn.supply_3v3>;  
        reset-gpios = <&conn.gpios 1 GPIO_ACTIVE_LOW>;  
        interrupts-extended = <&conn.interrupts 0 IRQ_TYPE_LEVEL_HIGH>;  
    };  
  
    eeprom@52 {  
        compatible = "foo,eeprom";  
        reg = <0x52>;  
        ...  
    };  
};
```



Addon devicetree applied

Merged dtb, equivalent dts

(Addon dtb applied at conn-a node of the base dtb)

```
regul1_3v3: regul1 { /* 3V3 Power Supply 1 */
    compatible = "fixed-regulator";
    ...
};
regul2_3v3: regul2 { /* 3V3 Power Supply 2 */
    compatible = "fixed-regulator";
    ...
};

soc {
    i2c1: i2c@1000 { /* I2C 1 Controller*/
        compatible = "soc-vendor,i2c-ctrl";
        ...
        io-expander@20 {
            compatible = "foo,io-expander";
            reg = <0x20>;
            vcc1-supply = <&regul1_3v3>;
            vcc2-supply = <&regul1_3v3>;
            reset-gpios = <&conn_a 1 GPIO_ACTIVE_LOW>; // Use GPIOs Nexus
            interrupts-extended = <&conn_a 0 IRQ_TYPE_LEVEL_HIGH>; // Use IRQs Nexus
        };

        eeprom@52 {
            compatible = "foo,eeprom";
            reg = <0x52>;
            ...
        };
    };
    i2c2: i2c@2000 { /* I2C 2 Controller*/
        compatible = "soc-vendor,i2c-ctrl";
        ...
    };
};
```

Merged dtb, equivalent dts (continued)

```
/* The rest of the soc node */
gpio1: gpio@4000 { /* GPIO 1 Controller */
    compatible = "soc-vendor,gpio-ctrl";
    ...
};
gpio2: gpio@5000 { /* GPIO 2 Controller */
    compatible = "soc-vendor,gpio-ctrl";
    ...
};

conn_a: conn-a {
    compatible = "vendor,connector";
    /* Nexus node definition for gpios */
    #gpio-cells = <2>;
    gpio-map-mask = <0xf 0xf>;
    gpio-map-pass-thru = <0x0 0xf>;
    gpio-map = <0 0 &gpio1 10 0>, <1 0 &gpio2 3 0>;
    /* Nexus node definition for interrupts */
    #address-cells = <0>;
    #interrupt-cells = <2>;
    interrupt-map = <0 IRQ_TYPE_LEVEL_LOW &gpio1 10 IRQ_TYPE_LEVEL_LOW>,
        <0 IRQ_TYPE_LEVEL_HIGH &gpio1 10 IRQ_TYPE_LEVEL_HIGH>,
        <1 IRQ_TYPE_LEVEL_LOW &gpio2 3 IRQ_TYPE_LEVEL_LOW>,
        <1 IRQ_TYPE_LEVEL_HIGH &gpio2 3 IRQ_TYPE_LEVEL_HIGH>;

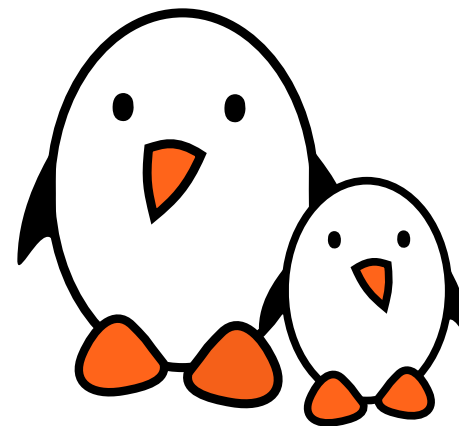
    /* Export resources */
    /export/ conn: &conn_a; /* The connector itself */
    /export/ supply_3v3: &regul1_3v3; /* The 3v3 at Connector A */
    /export/ i2c: &i2c1; /* The I2C at Connector A */
    /export/ gpios: &conn_a; /* The Nexus node for GPIOs */
    /export/ irq: &conn_a; /* The Nexus node for IRQs */
};

conn_b: conn-b {
    compatible = "vendor,connector";
    /* No changes */
    ...
};
```



What remains to be done?

bootlin





Adoption

▶ Users: Try it!

- Series sent to upstream libfdt/dtc
- Details available in a Bootlin blog post
 - With instructions to test it
 - [Device tree addons: describing non-discoverable addon boards and connectors](#)
- Don't hesitate to send feedback
 - [\[RFC PATCH v2 00/74\] Add support for dtb metadata and addon device-trees](#)



- ▶ libfdt/dtc: Continue the upstreaming effort
 - How to accelerate the review process?
 - David Gibson is alone on the libfdt/dtc maintenance.
- ▶ Bootloader: Bump (at least) to add support for **structured tags**
- ▶ Linux: Implement **of_addon_fdt_apply()**
 - Allow drivers (connector drivers) to apply an addon at a given node
 - Open the door to custom symbol resolver
 - Resolution done by a driver based on custom/specific criteria



- ▶ Update the Devicetree Specification
 - New dts keywords
 - New dtb tags format (structured tags) + New dtb tags



Design open question

- ▶ Golden Rule: **An addon dtb cannot add/modify a property in an existing node**
 - Modify an existing property → Change hardware description: Wrong.

```
&supply {  
    regulator-min-microvolt = <1800000>; /* Not allowed */  
};
```

- The `status` / `pinctrl` use case

```
&i2c {  
    pinctrl-0 = <&i2c_pinctrl>; /* Can make sense to set pinctrl */  
    status = "ok";             /* Can be set to "disabled" in the base board */  
};
```

- ▶ Should the Golden Rule be relaxed?
 - Allow all properties modification?
 - Restrict modification to some properties?
- ▶ Should the connector driver add/modify existing properties?
 - How to handle that in Bootloaders (merged dtb passed to the kernel)?



Devicetree addon validation (how to do?)

▶ Compilation

- Addons compilation warnings: missing `#addr-cells` and `#size-cells` in orphan nodes.
 - Properties available in the base devicetree
 - Should they be duplicated in the orphan node in the addon?
 - ▶ Possible. Must have same name, same value compared to those in base tree.

▶ Devicetree bindings

- Describe and check `/export/` and `/import/` items.
 - The contract between base board and addon
- Describe and check import compatible string.
 - Need some generic strings (`/import/ i2c: "bus,i2c"`)?
- Need for a global addon compatible string ("vendor,specific-connector")?
 - `/addon/ "vendor,specific-connector"`?
 - `/compatible/ "vendor,specific-connector"`?

Thank you!
Let's discuss it.

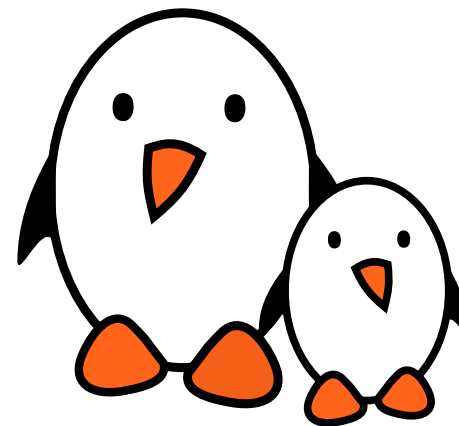
Hervé Codina
herve.codina@bootlin.com

Slides under CC-BY-SA 3.0
<https://bootlin.com/pub/conferences/>



Appendices: Internal details

bootlin





Key dates and events

▶ 2024

- Linux Plumbers Conference
 - Talk slides: [Runtime hotplug on non-discoverable busses with device tree overlays](#)

▶ 2025

- Proposal to introduce `export-symbols` node in device tree
 - Similar to `__symbols__` but custom at a given node level
 - Linux: [\[PATCH v2 0/7\] of: overlay: Add support for export-symbols node feature](#)
- ELC Europe + discussion
 - Talk slides: [Hotplug of Non-discoverable Hardware: Status and Future Directions](#)
 - Discussion: [Device tree representation of \(hotplug\) connectors: discussion at ELCE](#)
 - ▶ `export-symbols` node in DT: Rejected
 - ▶ Need for metadata in dtbs, new dts keywords (`/export/`, `/import/`), concept of addon



Key dates and events

▶ 2026:

- libfdt/dtc, proposal of features emerged during discussion
 - libfdt/dtc: [\[RFC PATCH 00/77\] Add support for dtb metadata and addon device-trees](#)
 - ▶ Need for `structured tags` as prerequisites
 - libfdt/dtc: [\[PATCH v3 00/15\] Add support for structured tags and v18 dtb version](#)
 - libfdt/dtc: [\[RFC PATCH v2 00/74\] Add support for dtb metadata and addon device-trees](#)
- Bootlin blog post
 - [Device tree addons: describing non-discoverable addon boards and connectors](#)
- Linux Plumbers Conference
 - This current session



Overlay Issues (solved with addon)

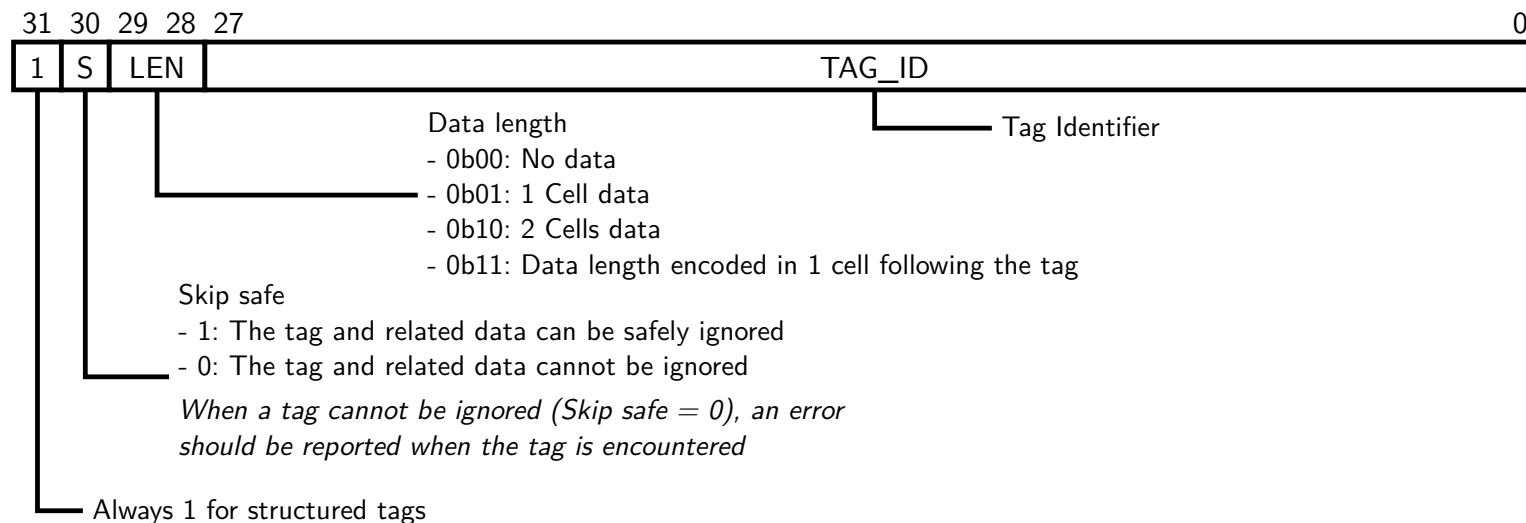
- ▶ Symbol references
 - In the base board dtb either all or no symbols are available
 - @ compilation → `__symbols__` node
 - No way to have a custom or subset list of symbols
 - No way to define 'local' symbols (i.e. symbols defined at a given node)
 - Symbols are unique and global. No simple way to handle case 3.
- ▶ Symbols information `__symbols__`, `__fixups__` and `__local_fixups__` as DT nodes
 - Mixed hardware description and information needed for device tree symbol resolution plumbing.
- ▶ Applying an overlay at a specific node chosen at runtime
 - Available in Linux but not in libfdt/dtc fdtoverlay tool.



Structured tags format: Prerequisite

► New **structured tag** values

dtb ✓ , dtba ✓



► Allow to add **new tags in the future keeping compatibility** with old libfdt/dtc version (Skip safe)



Devicetree metadata

- ▶ dt_flags header field
 - 0x1 for addon dtb (dtba)
 - **/addon/** dts keyword *dts X, dtba ✓*
- ▶ Identify phandles in properties (dtb new tags)
 - **FDT_PROPDATA_PHANDLE**: phandle value valid in current dtb *dtb ✓, dtba ✓*
 - **FDT_PROPDATA_PHANDLE_REF**: phandle value needs resolution *dtb X, dtba ✓*



Export / Import symbols (dynamic resolution)

- ▶ **Contract** between a connector described on a base devicetree and an addon
 - **Base** devicetree: **Export** symbols
 - Exported symbols are local to a node
 - Export symbols only at connector nodes
 - Multiple connectors can export symbols using same symbol name but referencing different nodes
 - **Addon** devicetree: **Import** symbols
 - An addon applied to a connector can import symbols exported at this connector
- ▶ **Dynamic resolution** based on imported symbols by the addon and exported symbols available at the node the addon is applied to



Export / Import symbols (dynamic resolution)

▶ Export (at node an addon is applied to)

- dts keyword in the node exporting a symbol **/export/ <name>: <ref>;** *dts ✓, dtbs ✓*
 - `/export/ foo: &foo1;`
 - `/export/ foo: &{/path/to/foo@100};`
- Related tags in dtb (New tags)
 - **FDT_EXPORT_SYM**, target node available in current dtb *dtb ✓, dtba ✓*
 - **FDT_EXPORT_SYM_REF**, external target node, need resolution *dtb ✗, dtba ✓*

▶ Import (global to a whole addon)

- dts keyword (global to dtbs) **/import/ <name>: <compatible>;** *dts ✗, dtbs ✓*
 - `/import/ foo: "bar,foo"`
- Related tag in dtb (New tag)
 - **FDT_IMPORT_SYM** Import external symbol, need resolution *dtb ✗, dtba ✓*



Export / Import symbols (dynamic resolution)

Base dts

```
regul1_3v3: regul1 {  
    ...  
};  
regul2_3v3: regul2 {  
    ...  
};  
conn1 {  
    /export/ power: &regul1_3v3;  
};  
conn2 {  
    /export/ power: &regul2_3v3;  
};
```

Addon dts

```
/addon/  
/import/ power: "";  
  
/ {  
    something {  
        supply = <&power>;  
    };  
};
```

Addon applied at conn2 node

```
regul1_3v3: regul1 {  
    ...  
};  
regul2_3v3: regul2 {  
    ...  
};  
conn1 {  
    /export/ power: &regul1_3v3;  
};  
conn2 {  
    /export/ power: &regul2_3v3;  
    something {  
        supply = <&regul2_3v3>;  
    };  
};
```

Base dtb

```
node regul1:  
    property phandle 11  
node regul2:  
    property phandle 22  
node conn1:  
    FDT_EXPORT_SYM 11 power  
node conn2:  
    FDT_EXPORT_SYM 22 power
```

Addon dtba

```
dt_flags = 1  
FDT_IMPORT_SYM power  
node something:  
    property supply 0xffffffff  
    FDT_PROPDATA_PHANDLE_REF 0 power
```

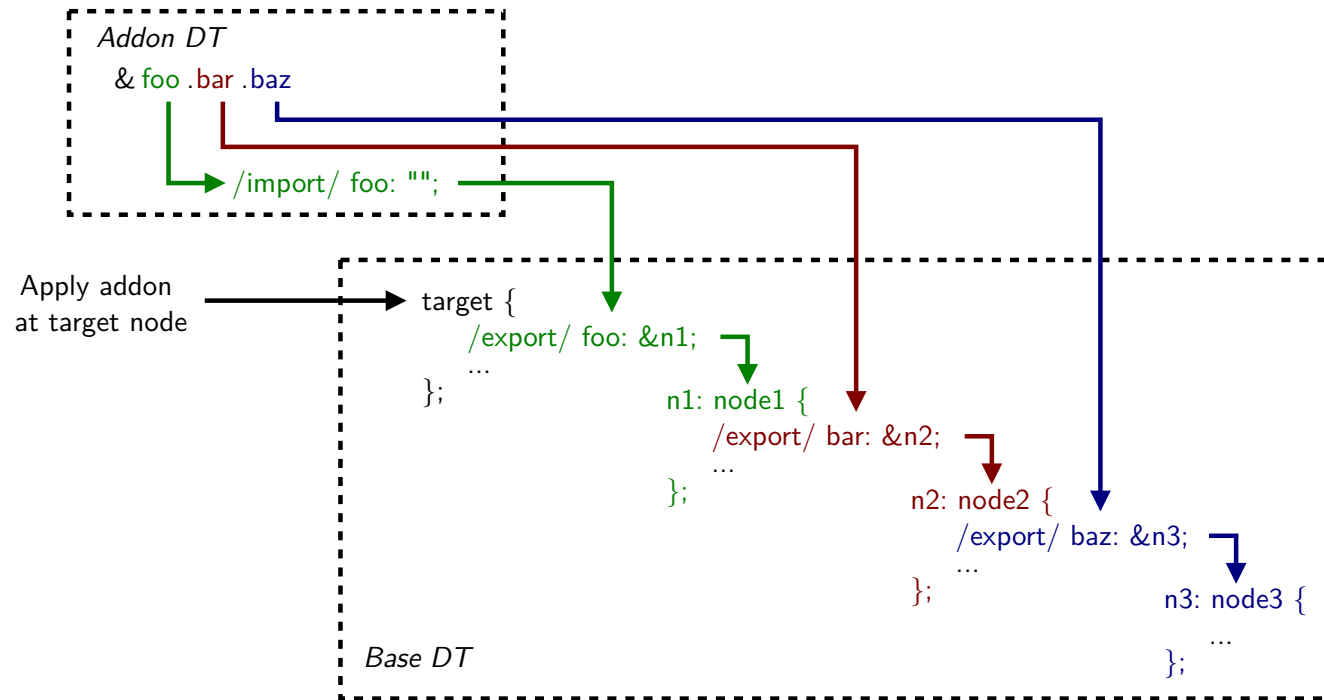
Resulting dtb

```
node regul1:  
    property phandle 11  
node regul2:  
    property phandle 22  
node conn1:  
    FDT_EXPORT_SYM 11 power  
node conn2:  
    FDT_EXPORT_SYM 22 power  
    node something  
        property supply 22  
        FDT_PROPDATA_PHANDLE 0
```



Namespace labels/symbols: Jump from export to export

- ▶ New syntax for reference by label/symbol: **&sym1.sym2.sym3...symN** *dts* ✗, *dtsa* ✓
 - Symbol list with a dot as separator
 - Allow to jump from exported symbols to exported symbols



`&foo.bar.baz` references the 'node3' node in the Base DT



Orphan nodes

▶ Node without a parent (other than root)

dtb ✗, *dtba* ✓

- An addon dtb (dtba) can have only orphan nodes and no 'root' node.
- Allow an addon to add subnodes to a non local node (node in the base devicetree)

```
/addon/  
/import/ foo: "";  
  
&foo { /* <--- the node referenced by 'foo' label/symbol is not local */  
    subnode { /* <--- This one is local (indeed, fully described here) */  
        ...  
    };  
};
```

▶ Clearly identified in dtb (New tag)

- **FDT_BEGIN_NODE_REF**, external node, need resolution
 - Idem FDT_BEGIN_NODE but for external node.

dtb ✗, *dtba* ✓



Reference orphan nodes by path

- ▶ Allow to reference a subnode of an orphan node using a path
- ▶ Labels are not encoded in dtb.
- ▶ Local nodes (orphan subnodes) need to be referenced when dts generated from dtb
- ▶ New syntax for reference path: **&{\$orphan_node/foo}** *dts X, dtsa ✓*

```
/addon/
```

```
&node1 {  
    subnode {  
        foo-phandle = <&{$node2/foo}>; /* <---- Reference foo node by path */  
    };  
};
```

```
&node2 {  
    foo {  
        ...  
    };  
};
```



An addon using multiple connectors: Driver involved

Base dts

```
connectors {
    compatible = "vendor,multiple-connectors";
    /export/ conn_a0: &conn_a0;
    /export/ conn_a1: &conn_a1;
    /export/ conn_b: &conn_b;
    conn_a0: conn-a0 {
        ...
    };
    conn_a1: conn-a1 {
        ...
    };
    conn_b: conn-b {
        ...
    };
};
```

Addon dts

```
/addon/
/import/ conn_a: "vendor,conn-a";
/import/ conn_b: "vendor,conn-b";
```

- **Custom resolution** is done by the `multiple-connectors` **driver**.
 - Matching import/export symbols using only the name of symbols is not enough.
 - Custom criteria (driver specific) needed to match `conn_a` from the addon to either `conn_a0` or `conn_a1` from the base devicetree.
 - The need for `conn_b` from the addon could be taken into account by the driver if needed.
 - The import compatible string can be used in the matching process.



fdtaddon tool

- ▶ New tool added to apply an addon dtba on a dtb: **fdtaddon**
 - Basics options, import/export resolution based on symbol names

```
$ fdtaddon --help
Usage: apply an addon to a base blob
      fdtaddon <options> <addon.dtba>

Options: -[i:o:t:e:vhV]
  -i, --input <arg>   Input base DT blob
  -o, --output <arg>  Output DT blob
  -t, --target <arg>  Target node
  ...
```

- Allow to perform a custom symbol resolution with the **-e** option

```
-e, --export <arg> Custom export symbol
Multiple --export or -e can be present in order to export multiple symbols.
The following syntaxes are supported:
  - Defines a export symbol pointing to a node using its symbol
    --export '<export_name>=&<symbol>'
  - Defines a export symbol pointing to a node by path
    --export '<export_name>=&{path_to_node}'
```



Nexus nodes

- ▶ Nexus nodes are defined in Devicetree Specification
 - Interrupts Nexus node in [Interrupt Nexus Properties](#) chapter
 - Generic Nexus node (slightly different) in [Nexus Nodes and Specifier Mapping](#) chapter

```
consumer {  
    ...  
    gpio = <&nexus 1 GPIO_ACTIVE_HIGH>;  
};  
  
nexus: nexus {  
    ...  
    #gpio-cells = <2>  
    gpio-map-mask = <0xf 0>;  
    gpio-map-pass-thru = <0 0xf>;  
    gpio-map = <0 0 &gpio3 10 0>,  
              <1 0 &gpio3 12 0>;  
};  
  
gpio3: gpio@3 {  
    ...  
    #gpio-cells = <2>;  
};
```

*Search item in map that matches
args & map-mask*

*args: 1, GPIO_ACTIVE_HIGH
map-mask: 0xf, 0
match item: 1, 0*

*result args based on
args & map-pass-thru | map args & ~map-pass-thru*

*args: 1, GPIO_ACTIVE_HIGH
map-pass-thru: 0, 0xf
map args: 12, 0
result args: 12, GPIO_ACTIVE_HIGH*

The `gpio` property of the `consumer` node references the gpio 12 of the `gpio@3` controller