# Appendix A Device Tree Source Format (version 1)

The Device Tree Source (DTS) format is a textual representation of a device tree in a form that can be processed by dtc into a binary device tree in the form expected by the kernel. The following description is not a formal syntax definition of DTS, but describes the basic constructs used to represent device trees.

## Node and property definitions

Device tree nodes are defined with a node name and unit address with braces marking the start and end of the node definition. They may be preceded by a label.

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

Nodes may contain property definitions and/or child node definitions. If both are present, properties shall come before child nodes.

Property definitions are name value pairs in the form:

```
[label:] property-name = value;
```

except for properties with empty (zero length) value which have the form:

```
[label:] property-name;
```

Property values may be defined as an array of 32-bit integer cells, as null-terminated strings, as bytestrings or a combination of these.

- Arrays of cells are represented by angle brackets surrounding a space separated list of C-style integers. Example:

  ```
  interrupts = <17 0xc>;
  ```

- A 64-bit value is represented with two 32-bit cells. Example:

  ```
  clock-frequency = <0x00000001 0x00000000>;
  ```

- A null-terminated string value is represented using double quotes (the property value is considered to include the terminating NULL character). Example:

  ```
  compatible = "simple-bus";
  ```

- A bytestring is enclosed in square brackets [ ] with each byte represented by two hexadecimal digits. Spaces between each byte are optional. Example:

  ```
  local-mac-address = [00 00 12 34 56 78];
  ```

  or equivalently:

  ```
  local-mac-address = [000012345678];
  ```

- Values may have several comma-separated components, which are concatenated together. Example:

  ```
  compatible = "ns16550", "ns8250";
  example = <0xf00f0000 19>, "a strange property format";
  ```

- In a cell array a reference to another node will be expanded to that node’s phandle. References may be & followed by a node’s label. Example:

  ```
  interrupt-parent = < &mpic >;
  ```

  or they may be & followed by a node’s full path in braces. Example:

  ```
  interrupt-parent = < &{/soc/interrupt-controller@40000} >;
  ```

- Outside a cell array, a reference to another node will be expanded to that node’s full path. Example:

  ```
  ethernet0 = &EMAC0;
  ```

- Labels may also appear before or after any component of a property value, or between cells of a cell array, or between bytes of a bytestring. Examples:

  ```
  reg = reglabel: <0 sizelabel: 0x1000000>;
  prop = [ab cd ef byte4: 00 ff fe];
  str = start: "string value" end: ;
  ```

## File layout

Version 1 DTS files have the overall layout:

```
/dts-v1/;

[memory reservations]

/ {
     [property definitions]
     [child nodes]
};
```

- The /dts-v1/; shall be present to identify the file as a version 1 DTS (dts files without this tag will be treated by dtc as being in the obsolete version 0, which uses a different format for integers in addition to other small but incompatible changes).

- Memory reservations define an entry for the device tree blob's memory reservation table. They have the form:

  e.g., `/memreserve/ <address> <length>;`

  Where <address> and <length> are 64-bit C-style integers.

- The / { ... }; section defines the root node of the device tree.

- C style (/* ... */) and C++ style (// ...) comments are supported.