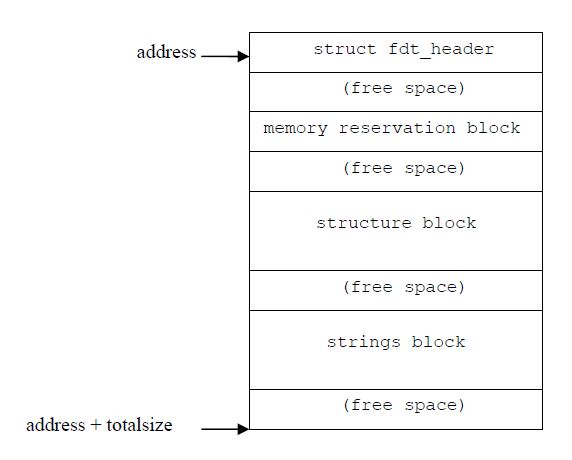
## Flat Device Tree Physical Structure

数据结构包含一个header和三个大小可变的section：memory reservation block，structure block、和string block。这些写按顺序存储在flattened device tree中。

Device stree 结构作为一个整体被加载到内存中。如上图的方式被组合在一起（低地址在图中的顶端）。

Free space可能不存在，但是在某些情况下要求满足各个block的对齐要求，会存在free space。

**Header**

Device tree的header定义如下，所有header的字段为32位的整型，按照big-endian存储。

The layout of the header for the device tree is defined by the following C structure. All the header fields are 32-bit integers, stored in big-endian format.

struct fdt\_header {

uint32\_t magic;

uint32\_t totalsize;

uint32\_t off\_dt\_struct;

uint32\_t off\_dt\_strings;

uint32\_t off\_mem\_rsvmap;

uint32\_t version;

uint32\_t last\_comp\_version;

uint32\_t boot\_cpuid\_phys;

uint32\_t size\_dt\_strings;

uint32\_t size\_dt\_struct;

};

* magic

This field shall contain the value 0xd00dfeed (big-endian).

* totalsize

该字段表示device tree数据结构的总共大小。该大小包含struct的所有section：header,memory revervation, structure block,strings block,而且包含block之间或者最后一个block的free space gaps。

* off\_dt\_struct

该字段表示从header的起始到structure的偏移字节数。

* off\_dt\_strings

该字段表示从header的起始到strings block的偏移字节数。

* off\_mem\_rsvmap

表示从header起始到memory reservation block的偏移字节数。

* version

表示device tree 数据结构的版本

* last\_comp\_version

表示向后兼容的最低版本

* boot\_cpuid\_phys

表示用于启动系统的CPU ID，应和device tree中CPU node的reg 属性表示物理ID相同。

* size\_dt\_strings

表示device tree blob（device tree 二进制数据）中strings block的长度，单位为字节。

* size\_dt\_struct

表示device tree blob中structure block的长度，单位为字节。

## Memory Reservation Block

Memory revervation block 提供了一个物理内存的区域列表，该区域是保留的，不能用于通用的内存分配。用于保护重要的数据被覆盖。

Memory reservation block 包含64位 big-endian整型对的列表。每对以下面的C结构表示。

struct fdt\_reserve\_entry {

uint64\_t address;

uint64\_t size;

};

每一对表示保留内存域的物理地址和大小。Address和size等于0表示reserved block结束。Address和size是64bit，在32位的CPU，高32bit被忽略。Memory reservation block的起始地址应该8字节对齐。

## Structure Block

Structure block描述structure和device tree本身的内容。由按顺序排列的token和数据组成。这些组织成一个线性的树结构。

Structure block中的每一个token都是四字节对齐，因此structure block本身也是四字节对齐。

### Lexical structure

Structure block由按顺序的pieces组成。每一个以一个token开始，既一个big-endian 32bit整型。一些tokens后面带有额外的数据，数据的格式由token的值来确定。所有的token应该32 bit对齐，这样需要pading bytes（值为0x00）插入到前一个token数据的后面。

五类toke如下：

* FDT\_BEGIN\_NODE (0x00000001)

FDT\_BEGIN\_NODE标记一个node起始。后面将跟随node的unit name作为额外的数据。name 是一个以NULL结尾的字符串，如果有unit address，并且包含unit address。如果有对齐必要，name后面插入值为0的pading bytes，下一个token可以是除了FDT\_END的任何一个。

* FDT\_END\_NODE (0x00000002)

FDT\_END\_NODE标记node的结束。该token没有额外的数据。因此后面是下一个除了FDT\_PROP 的任何一个token。

* FDT\_PROP (0x00000003)

FDT\_PROP token标记device tree中一个property的开始。后面跟随描述property的附加数据。该数据首先包含property的length和name，通过下面的C语言结构：

struct {

uint32\_t len;

uint32\_t nameoff;

}

结构体中的两个字段都是32-bit big-endian整型。

* 1. len表示property的值的长度，单位为字节（可能为0，表示一个空的属性）。
  2. nameoff 表示property name在string block中的偏移，在偏移处name为以NULL结束的字符串存储。

在该结构之后，是以长度为len的byte string的property的值。如果必要，该值用zeroed padding作为附加数据用于32-bit对齐，而且下一个token可以是除了FDT\_END的任何token。

* FDT\_NOP (0x00000004)

FDT\_NOP会被任何解析device tree的程序忽视。该token没有附加数据，因此后面可以跟任何有效的token。

一个property或node的定义可以通过FDT\_NOP进行覆写，可以从该tree中移除该property或node而不用移动device tree blob中的其他section。

* FDT\_END (0x00000009)

FDT\_END标记structure block的结束。应该只有一个FDT\_END token，并且是structure block的最后block一个token。该token没有附加数据，因此从structure block开始到紧跟FDT\_END token后面的偏移数等于device tree blob header中的size\_dt\_struct字段的值。

## Tree structure

Device tree 表现为一个线性的tree：每一个node表现为，以FDT\_BEGIN\_NODE token开始，以FDT\_BEGIN\_NODE token结束。node的property和subnode（如果有）出现在FDT\_END\_NODE之前，因此子节点的FDT\_BEGIN\_NODE和FDT\_END\_NODE token被嵌套在了父节点的这些token中。

Structure block作为一个整体，包括root node的representation（root node 包含所有其他节点的representation），以FDT\_END token标记structure block作为一个整体的结束。

更确切的说，每一个node的表现包含下面的组件：

* (可选的)任何数目的FDT\_NOP tokens
* FDT\_BEGIN\_NODE token
* The node’s name as a NULL-terminated string
* [zeroed padding bytes to align to a 4-byte boundary]
* For each property of the node:
  + (可选的)任何数目的FDT\_NOP tokens
  + FDT\_PROP token
* property information as given in 7.4.1
* [zeroed padding bytes to align to a 4-byte boundary]
* 所以以该格式的子node的表现
* (可选的)任何数目的FDT\_NOP tokens
* FDT\_END\_NODE token

……

* FDT\_BEGIN\_NODE token

……

* FDT\_END\_NODE token
* FDT\_END token

该方法需要一个特定node的所有property定义优先于该node subnode的定义。尽管，如果property和subnode混合，该struct不会变得混乱，但是code需要处理一个通过该需求而简化的一个flat tree。

## Strings Block

Strings block 包含device tree中所有property name的字符串。这些NULL结尾的字符串在该sectiion被简单的级联到一块，并且使用structure block中的offset进行引用。

Strings block没有对齐约束并且可以出现在从device tree blob开始的任何偏移处。