<http://blog.csdn.net/lsn946803746/article/details/52572375>

# android 设备树中的devices\_node的数据获取

android/linux下devicetree中常用的of函数

从device\_node中获取信息：

int of\_property\_read\_u8\_array(const struct device\_node \*np, const char \*propname,u8 \*out\_values, size\_t sz);

int of\_property\_read\_u16\_array(const struct device\_node \*np, const char \*propname,u16 \*out\_values, size\_t sz);

int of\_property\_read\_u32\_array(const struct device\_node \*np, const char \*propname,u32 \*out\_values, size\_t sz);

从设备结点np中读取属性名为propname，类型为8、16、32、位整型数组的属性值，并放入out\_values，sz指明了要读取的个数。

static inline int of\_property\_read\_u8(const struct device\_node \*np,const char \*propname,u8 \*out\_value)

static inline int of\_property\_read\_u16(const struct device\_node \*np,const char \*propname,u8 \*out\_value)

static inline int of\_property\_read\_u32(const struct device\_node \*np,const char \*propname,u8 \*out\_value)

从设备结点np中读取属性名为propname，类型为8、16、32位的属性值，并放入out\_values。实际上这里调用的就是sz为1的XXX\_array函数。

int of\_property\_read\_u32\_index(const struct device\_node \*np,const char\*propname,u32 index, u32 \*out\_value)

从设备结点np中读取属性名为propname的属性值中第index个u32数值给out\_value

int of\_property\_read\_u64(conststruct device\_node \*np, const char \*propname,u64 \*out\_value)

从设备结点np中读取属性名为propname，类型为64位的属性值，并放入out\_values

int of\_property\_read\_string(struct device\_node \*np, const char \*propname,const char\*\*out\_string)

从设备结点np中读取属性名为propname的字符串型属性值

int of\_property\_read\_string\_index(struct device\_node \*np, const char \*propname,intindex, const char \*\*output)

从设备结点np中读取属性名为propname的字符串型属性值数组中的第index个字符串

int of\_property\_count\_strings(struct device\_node \*np, const char \*propname)

从设备结点np中读取属性名为propname的字符串型属性值的个数

unsigned int irq\_of\_parse\_and\_map(struct device\_node \*dev, int index)

从设备节点dev中读取第index个irq号

int of\_irq\_to\_resource(struct device\_node \*dev, int index, struct resource \*r)

从设备节点dev中读取第index个irq号，并填充一个irq资源结构体

int of\_irq\_count(struct device\_node \*dev)

获取设备节点dev的irq个数

static inline bool of\_property\_read\_bool(const struct device\_node \*np,const char \*propname);

如果设备结点np含有propname属性，则返回true，否则返回false。一般用于检查空属性是否存在。

struct property\* of\_find\_property(const struct device\_node \*np,const char \*name,int \*lenp)

根据name参数，在指定的设备结点np中查找匹配的property，并返回这个property

const void \* of\_get\_property(const struct device\_node \*np, const char \*name,int \*lenp)

根据name参数，在指定的设备结点np中查找匹配的property，并返回这个property的属性值

struct device\_node\* of\_get\_parent(const struct device\_node \*node)

获得node节点的父节点的device node

int of\_device\_is\_compatible(const struct device\_node \*device,const char \*compat);

判断设备结点device的compatible属性是否包含compat指定的字符串

从of\_allnodes中查找信息：

struct device\_node\* of\_find\_node\_by\_path(const char \*path)   
根据路径参数，在全局链表of\_allnodes中，查找匹配的device\_node

struct device\_node\* of\_find\_node\_by\_name(struct device\_node \*from,const char \*name)   
则根据name在全局链表of\_allnodes中查找匹配的device\_node,若from=NULL表示从头开始查找

struct device\_node\* of\_find\_node\_by\_type(struct device\_node \*from,const char \*type)

根据设备类型在全局链表of\_allnodes中查找匹配的device\_node

struct device\_node \* of\_find\_compatible\_node(struct device\_node \*from, const char\*type, const char，\*compatible);

根据compatible的属性值在全局链表of\_allnodes中查找匹配的device\_node，大多数情况下，from、type为NULL。

struct device\_node\* of\_find\_node\_with\_property(struct device\_node \*from,const char \*prop\_name)

根据节点属性的name在全局链表of\_allnodes中查找匹配的device\_node

struct device\_node\* of\_find\_node\_by\_phandle(phandle handle)

根据phandle在全局链表of\_allnodes中查找匹配的device\_node

杂：

void \_\_iomem\* of\_iomap(struct device\_node \*node, int index);

通过设备结点直接进行设备内存区间的 ioremap()，index是内存段的索引。若设备结点的reg属性有多段，可通过index标示要ioremap的是哪一段，只有1段的情况，index为0

unsigned long \_\_init of\_get\_flat\_dt\_root(void)

用来查找在dtb中的根节点，好像返回的都是0

int of\_alias\_get\_id(struct device\_node \*np, const char \*stem)

获取节点np对应的aliasid号

struct device\_node\* of\_node\_get(struct device\_node \*node)

void of\_node\_put(struct device\_node \*node)

device node计数增加/减少

const struct of\_device\_id\* of\_match\_node(const struct of\_device\_id \*matches,const struct device\_node\*node)

将matches数组中of\_device\_id结构的name和type与device node的compatible和type匹配，返回匹配度最高的of\_device\_id结构

platform\_device和resource相关：

int of\_address\_to\_resource(struct device\_node \*dev, int index,struct resource \*r)

根据设备节点dev的reg属性值，填充资源结构体r。Index参数指明了使用reg属性中第几个属性值，一般设置为0，表示第一个。

struct platform\_device\* of\_device\_alloc(struct device\_node \*np,const char \*bus\_id,struct device \*parent)

根据device node，bus\_id以及父节点创建该设备的platform\_device结构，同时会初始化它的resource成员。

int of\_platform\_bus\_probe(struct device\_node \*root,const struct of\_device\_id \*matches,struct device \*parent)

遍历of\_allnodes中的节点挂接到of\_platform\_bus\_type总线上,由于此时of\_platform\_bus\_type总线上还没有驱动,所以此时不进行匹配

int of\_platform\_populate(struct device\_node \*root,const struct of\_device\_id \*matches,const struct of\_dev\_auxdata \*lookup,struct device \*parent)

遍历of\_allnodes中的所有节点，生成并初始化所以节点的platform\_device结构

struct platform\_device\* of\_find\_device\_by\_node(struct device\_node \*np)

根据device\_node查找返回该设备对应的platform\_device结构