NAME

OF_getprop, **OF_getproplen**, **OF_getencprop**, **OF_hasprop**, **OF_searchprop**, **OF_searchencprop**, **OF_getprop_alloc**, **OF_getencprop_alloc**, **OF_getprop_alloc_multi**, **OF_getencprop_alloc_multi**, **OF_prop_free**, **OF_nextprop**, **OF_setprop** - access properties of device tree node

SYNOPSIS

#include <dev/ofw/ofw_bus.h>
#include <dev/ofw/ofw_bus_subr.h>

ssize_t
OF_getproplen(*phandle_t node*, *const char *propname*);

ssize_t
OF_getprop(*phandle_t node*, *const char *propname*, *void *buf*, *size_t len*);

ssize_t
OF_getencprop(*phandle_t node*, *const char *prop*, *pcell_t *buf*, *size_t len*);

int
OF_hasprop(*phandle_t node*, *const char *propname*);

ssize_t
OF_searchprop(*phandle_t node*, *const char *propname*, *void *buf*, *size_t len*);

ssize_t
OF_searchencprop(*phandle_t node*, *const char *propname*, *pcell_t *buf*, *size_t len*);

ssize_t
OF_getprop_alloc(*phandle_t node*, *const char *propname*, *void **buf*);

ssize_t
OF_getencprop_alloc(*phandle_t node*, *const char *propname*, *pcell_t **buf*);

ssize_t
OF_getprop_alloc_multi(*phandle_t node*, *const char *propname*, *int elsz*, *void **buf*);

ssize_t
OF_getencprop_alloc_multi(*phandle_t node*, *const char *propname*, *int elsz*, *pcell_t **buf*);

void

OF_prop_free(*void *buf*);

int
OF_nextprop(*phandle_t node, const char *propname, char *buf, size_t len*);

int
OF_setprop(*phandle_t node, const char *propname, const void *buf, size_t len*);

DESCRIPTION

Device nodes can have associated properties. Properties consist of a name and a value. A name is a human-readable string from 1 to 31 characters long. A value is an array of zero or more bytes that encode certain information. The meaning of that bytes depends on how drivers interpret them. Properties can encode byte arrays, text strings, unsigned 32-bit values or any combination of these types.

Property with a zero-length value usually represents boolean information. If the property is present, it signifies true, otherwise false.

A byte array is encoded as a sequence of bytes and represents values like MAC addresses.

A text string is a sequence of n printable characters. It is encoded as a byte array of length n + 1 bytes with characters represented as bytes plus a terminating null character.

Unsigned 32-bit values, also sometimes called cells, are encoded as a sequence of 4 bytes in big-endian order.

OF_getproplen() returns either the length of the value associated with the property *propname* in the node identified by *node*, or 0 if the property exists but has no associated value. If *propname* does not exist, -1 is returned.

OF_getprop() copies a maximum of *len* bytes from the value associated with the property *propname* of the device node *node* into the memory specified by *buf*. Returns the actual size of the value or -1 if the property does not exist.

OF_getencprop() copies a maximum of *len* bytes into memory specified by *buf*, then converts cell values from big-endian to host byte order. Returns the actual size of the value in bytes, or -1 if the property does not exist. *len* must be a multiple of 4.

OF_hasprop() returns 1 if the device node *node* has a property specified by *propname*, and zero if the property does not exist.

OF_searchprop() recursively looks for the property specified by *propname* starting with the device node *node* followed by the parent node and up to the root node. If the property is found, the function copies a maximum of *len* bytes of the value associated with the property into the memory specified by *buf*. Returns the actual size in bytes of the value, or -1 if the property does not exist.

OF_searchencprop() recursively looks for the property specified by *propname* starting with the device node *node* followed by the parent node and up to the root node. If the property is found, the function copies a maximum of *len* bytes of the value associated with the property into the memory specified by *buf*, then converts cell values from big-endian to host byte order. Returns the actual size in bytes of the value, or -1 if the property does not exist.

OF_getprop_alloc() allocates memory large enough to hold the value associated with the property *propname* of the device node *node* and copies the value into the newly allocated memory region. Returns the actual size of the value and stores the address of the allocated memory in **buf*. If the property has a zero-sized value **buf* is set NULL. Returns -1 if the property does not exist or memory allocation failed. Allocated memory should be released when no longer required by calling **OF_prop_free**(). The function might sleep when allocating memory.

OF_getencprop_alloc() allocates enough memory to hold the value associated with the property *propname* of the device node *node*, copies the value into the newly allocated memory region, and then converts cell values from big-endian to host byte order. The actual size of the value is returned and the address of allocated memory is stored in **buf*. If the property has zero-length value, **buf* is set to NULL. Returns -1 if the property does not exist or memory allocation failed or the size of the value is not divisible by a cell size (4 bytes). Allocated memory should be released when no longer required by calling **OF_prop_free**(). The function might sleep when allocating memory.

OF_getprop_alloc_multi() allocates memory large enough to hold the value associated with the property *propname* of the device node *node* and copies the value into the newly allocated memory region. The value is expected to be an array of zero or more elements *elsz* bytes long. Returns the number of elements in the value and stores the address of the allocated memory in **buf*. If the property has a zero-sized value **buf* is set NULL. Returns -1 if the property does not exist or memory allocation failed or the size of the value is not divisible by *elsz*. Allocated memory should be released when no longer required by calling **OF_prop_free**(). The function might sleep when allocating memory.

OF_getencprop_alloc_multi() allocates memory large enough to hold the value associated with the property *propname* of the device node *node* and copies the value into the newly allocated memory region, and then converts cell values from big-endian to host byte order. The value is expected to be an array of zero or more elements *elsz* bytes long. Returns the number of elements in the value and stores the address of the allocated memory in **buf*. If the property has a zero-sized value **buf* is set NULL. Returns -1 if the property does not exist or memory allocation failed or the size of the value is not

divisible by *elsz*. Allocated memory should be released when no longer required by calling **OF_prop_free**(). The function might sleep when allocating memory.

OF_prop_free() releases memory at *buf* that was allocated by **OF_getprop_alloc**(), **OF_getencprop_alloc**(), **OF_getprop_alloc_multi**(), **OF_getencprop_alloc_multi**().

OF_nextprop() copies a maximum of *size* bytes of the name of the property following the *propname* property into *buf*. If *propname* is NULL, the function copies the name of the first property of the device node *node*. **OF_nextprop**() returns -1 if *propname* is invalid or there is an internal error, 0 if there are no more properties after *propname*, or 1 otherwise.

OF_setprop() sets the value of the property *propname* in the device node *node* to the value beginning at the address specified by *buf* and running for *len* bytes. If the property does not exist, the function tries to create it. **OF_setprop**() returns the actual size of the new value, or -1 if the property value cannot be changed or the new property cannot be created.

EXAMPLES

```
phandle_t node;
phandle_t hdmixref, hdminode;
device_t hdmi;
uint8_t mac[6];
char *model;

/*
 * Get a byte array property
 */
if (OF_getprop(node, "eth,hwaddr", mac, sizeof(mac)) != sizeof(mac))
    return;

/*
 * Get internal node reference and device associated with it
 */
if (OF_getencprop(node, "hdmi", &hdmixref) <= 0)
    return;
hdmi = OF_device_from_xref(hdmixref);

/*
 * Get string value of model property of HDMI framer node
 */
hdminode = OF_node_from_xref(hdmixref);
```

```
if (OF_getprop_alloc(hdminode, "model", (void **)&model) <= 0)
    return;
```

SEE ALSO

OF_device_from_xref(9), OF_node_from_xref(9)

AUTHORS

This manual page was written by Oleksandr Tymoshenko <*gonzo@FreeBSD.org*>.