Open Firmware

Recommended Practice:

Universal Serial Bus

Version 1

June 1, 1998

Published by the Open Firmware Working Group

This document is a voluntary-use recommended practice of the Open Firmware Working Group. The Open Firmware Working Group is an ad hoc committee composed of individuals interested in Open Firmware as defined by IEEE 1275-1994, related standards, and their application to various computer systems.

The Open Firmware Working Group is involved both in IEEE sanctioned standards activities, whose final results are published by IEEE, and in informal recommendations such as this, which are published on the Internet at:

`http://playground.sun.com/pub/1275`

Membership in the Open Firmware Working Group is open to all interested parties. The working group meets at regular intervals at various locations. For more information send email to:

`p1275-wg@risc.sps.mot.com`

Revision History

1. Overview and References

This document describes the application of Open Firmware to the Universal Serial Bus (USB) and addresses nodes representing USB devices.

Since there is no provision for FCode on USB devices, this binding restricts itself to providing a common framework for a USB device tree. Implementations that support USB devices as boot, input, and output devices, will define their own methods and properties to communicate with USB endpoints. Those methods are beyond the scope of this document.

1.1. References

[1] *IEEE Standard 1275-1994 Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices*
[2] *Universal Serial Bus Specification, 1.0*

1.2. Definitions of Terms

combined node: A device tree node that represents both a USB device and a USB interface.

device node: A device tree node that represents a USB device.

USB host controller: A hardware device that interfaces between a computer system and a USB bus.

host controller node: A device tree node that represents a USB host controller.

hub node: 1. A device tree node that represents a USB hub. 2. A combined node whose USB interface is a USB hub.

interface node: A device tree node that represents a USB interface.

low speed: 1.5 Mbs, rather than the normal 12.0 Mbs signalling speed. See [2].

transaction: The delivery of service to an endpoint. See [2].

transfer: One or more transactions. See [2].

USB device: A hardware device that connects to a USB bus.

USB hub: A USB interface that provides additional connections to the USB bus.

USB interface: An independent piece of functionality of a USB device.

1.2.1. USB Descriptors

[2] defines a number of data structures that are referenced in this document. They are:

DEVICE
See [2] section 9.6.1. Fields referenced by this binding: bDeviceClass, bDeviceSubClass, bDeviceProtocol, idVendor, idProduct, bcdDevice, bNumConfigurations.

CONFIGURATION
See [2] section 9.6.2. Fields referenced by this binding: bNumInterfaces, bConfigurationValue.

INTERFACE
See [2] section 9.6.3. Fields referenced by this binding: bInterfaceNumber, bAlternateSetting, bInterfaceClass, bInterfaceSubClass, bInterfaceProtocol.

2. Bus Characteristics

USB is not a memory-mapped bus. An operation with a USB device is performed by executing a transfer consisting of one or more bus transactions to move information between the host system and the device. A bus transaction consists of multiple packets, including a token packet, possibly a data packet, and possibly a handshake packet. The specific packets are allowed and/or required based on the transaction type. Four transfer types are defined: control, interrupt, bulk, and isochronous.

2.1. Bus-specific Configuration Variables

None.

2.2. Format of a Probe List

None.

2.3. FCode Interpretation Semantics

None (USB has no provision for device identification via FCode).

3. Device Tree Structure

This document defines four types of device tree nodes for USB: host controller nodes, device nodes, interface nodes, and combined nodes.

Generally, USB devices are represented as two levels of device tree nodes: a device node representing the entire USB device, with one or more child interface nodes representing the individual USB interfaces on the device. For special cases the device and interface nodes are combined into a single combined node.

3.1. Host Controller Nodes

3.1.1. Host Controller Nodes Properties

3.1.1.1. Open Firmware-defined Properties for Host Controller Nodes

The following standard properties, as defined in [1], have special meaning or interpretation for host controller nodes.

`"#address-cells"` S

Standard property to define the address format. Its value shall be 1, encoded as with encode-int.

`"#size-cells"` S

Standard prop-name to define the package's address size format.

prop-encoded-array: 0, encoded as with encode-int.

The value of `"#size-cells"` for host controller nodes shall be 0, representing the fact that host controller addresses are an enumeration rather than memory-like address ranges.

3.1.1.2. Bus-specific Properties for Host Controller Nodes

None.

3.1.2. Host Controller Node Methods

3.1.2.1. Open Firmware-defined Methods for Host Controller Nodes

A package implementing a host controller node shall implement the following standard methods as defined in [1], with physical address representation as specified in section 3.2.1.

`decode-unit` (addr len -- port) M
Convert text unit-string to physical address.

`encode-unit` (port -- addr len) M
Convert physical address to text unit-string.

3.1.2.2. Bus-specific Methods for Host Controller Nodes

None.

3.2. Device Nodes

Device nodes represent USB devices. USB devices may require device-specific support to enable choice of the USB device configuration and/or to enable operation of multiple USB interfaces in a coordinated fashion.

Unless otherwise specified, a device node shall be created for each USB device.

3.2.1. Device Node Address Representation

The textual representation of a device node unit address shall be the number of the USB hub port or the USB host controller port to which this USB device is attached, in lower case hexadecimal with leading zeroes suppressed.

3.2.2. Device Node Properties

3.2.2.1. Open Firmware-defined Properties for Device Nodes

The following properties, as defined in [1], have special meanings or interpretations for device nodes. These properties shall be created for each device node created.

The following notation is used:

From the DEVICE descriptor for this USB device:

VID	idVendor
PID	idProduct
REV	bcdDevice
DC	bDeviceClass
DSC	bDeviceSubClass
DPROTO	bDeviceProtocol

From the CONFIGURATION descriptor for this USB device:

CN	bConfigurationValue for the selected configuration.

The textual representation of VID, PID, REV, DC, DSC, DPROTO, and CN shall be in lower case hexadecimal with leading zeroes suppressed.

`"#address-cells"` S
Standard property to define the address format. Its value shall be 2, encoded as with encode-int.

`"compatible"` S

Standard prop-name

prop-encoded-array; a list of strings, each encoded with encode-string and concatenated with encode+.

Construct encoded strings as enumerated in the following list. All of the applicable strings shall be included in the value of `"compatible"`. Their relative order shall be as in the list, string 1) before string 2), and so forth.

Strings containing a configuration number (1 and 3) shall be omitted for a single-configuration USB device. Strings containing a device class (5..10) shall be omitted if the device class is 0.

1) usbVID,PID.REV.configCN
2) usbVID,PID.REV
3) usbVID,PID.configCN
4) usbVID,PID
5) usbVID,classDC.DSC.DPROTO
6) usbVID,classDC.DSC
7) usbVID,classDC
8) usb,classDC.DSC.DPROTO
9) usb,classDC.DSC
10) usb,classDC
11) usb,device

Note: entry 11), "usb,device", provides a mechanism to bind a generic driver to a USB device.

`"name"` S

Standard prop-name.

prop-encoded-array: a string encoded with encode-string.

The name of the node should be chosen from the following table, using the first name applicable:

bDeviceClass	bDeviceSubclass	bDeviceProtocol	Name
9	any	any	hub
[MASS]	1	any	storage
[MASS]	2	any	cdrom
[MASS]	3	any	tape
[MASS]	4	any	solid-state
[MASS]	any	any	storage
any	any	any	device

Note: The USB Device Working Group has not yet completed standardization of the Mass Storage Device Class, and so a device class code has not yet been assigned and the subclass codes are not final. The names above should be used, as appropriate, when the Mass Storage Device Class specification is complete.

`"reg"` S

Standard prop-name.

prop-encoded-array: one integer, encoded as with encode-int.

The `"reg"` property for a device node shall consist of the number of the USB hub port or the USB host controller port to which this USB device is attached. As specified in [2] section 11.11.2.1, port numbers range from 1 to 255.

`"#size-cells"` S

Standard prop-name.

prop-encoded-array: 0, encoded as with encode-int.

The value of `"#size-cells"` for device nodes shall be 0, representing the fact that USB device addresses are an enumeration rather than memory-like address ranges.

3.2.2.2. Bus-specific Properties for Device Nodes

`"assigned-address"`

prop-name to indicate the assigned USB bus address.

prop-encoded-array: one integer, encoded as with encode-int.

This property supplies the USB bus address assigned to this USB device. Each USB device that is assigned a USB address shall be assigned a USB address that is unique among all USB devices on this USB bus.

`"configuration#"`

prop-name to indicate the selected configuration.

prop-encoded-array: one integer, encoded as with encode-int.

This property indicates the bConfigurationValue contained in the CONFIGURATION descriptor for the configuration selected for this USB device.

`"low-speed"`

prop-name to indicate that the USB device is low speed.

prop-encoded-array: None; presence or absence of the property conveys the information.

This property, if present, indicates that the USB device is low speed.

3.2.3. Device Node Methods

3.2.3.1. Open Firmware-defined Methods for Device Nodes

A package implementing a device node shall implement the following standard methods as defined in [1], with physical address representation as specified in section 3.3.1.

`decode-unit` (addr len -- config# interface#) M

Convert text unit-string to physical address.

`encode-unit` (config# interface# -- addr len) M

Convert physical address to text unit-string.

3.2.3.2. Bus-specific Methods for Device Nodes

None.

3.3. Interface Nodes

Interface nodes represent the USB interfaces present on a USB device. Normally, each interface represents independently controlled functionality, although for some device classes (e.g. Communications, Audio) one interface may provide "out of band" control for another.

Unless otherwise specified, an interface node shall be created for each USB interface.

3.3.1. Interface Node Address Representation

The textual representation of an interface node unit address shall be constructed as follows:

If the bConfigurationValue of the CONFIGURATION descriptor associated with this USB interface is equal to 1, the textual representation shall be the bInterfaceNumber of the INTERFACE descriptor with value 0 for bAlternateSetting associated with this USB interface, expressed in lower case hexadecimal with leading zeroes suppressed.

If the bConfigurationValue is not equal to 1, the textual representation shall be the bInterfaceNumber of the INTERFACE descriptor with value 0 for bAlternateSetting associated with this USB interface, a comma, and the bConfigurationValue, with both values expressed in lower case hexadecimal with leading zeroes suppressed.

3.3.2. Interface Node Properties

3.3.2.1. Open Firmware-defined Properties for Interface Nodes

The following properties, as defined in [1], have special meanings or interpretations for interface nodes. These properties shall be created for each interface node created.

The following notation is used:

From the DEVICE descriptor for this USB interface:

VID	idVendor
PID	idProduct
REV	bcdDevice
DC	bDeviceClass
DSC	bDeviceSubClass
DPROTO	bDeviceProtocol

From the INTERFACE descriptor with the value of 0 for bAlternateSetting for this USB interface:

IN	bInterfaceNumber
IC	bInterfaceClass
ISC	bInterfaceSubClass
IPROTO	bInterfaceProtocol

The textual representation of VID, PID, REV, DC, DSC, DPROTO, IN, IC, ISC, and IPROTO shall be in lower case hexadecimal with leading zeroes suppressed.

`"compatible"` S

Standard prop-name.

prop-encoded-array; a list of strings, encoded with encode-string and concatenated with encode+.

Construct encoded strings as enumerated in the following list. All of the applicable strings shall be included in the value of `"compatible"`. Their relative order shall be as in the list, string 1) before string 2), and so forth.

Strings containing an interface class (3..8) shall be omitted if the interface class is 0.

1) usbifVID,PID.REV.configCN.IN
2) usbifVID,PID.configCN.IN
3) usbifVID,classIC.ISC.IPROTO
4) usbifVID,classIC.ISC
5) usbifVID,classIC
6) usbif,classIC.ISC.IPROTO
7) usbif,classIC.ISC
8) usbif,classIC

`"name"` S

Standard prop-name.

prop-encoded-array: a string encoded with encode-string.

The name of the node should be chosen from the following table, using the first name applicable:

bInterface Class	bInterface Subclass	bInterface Protocol	Name
1	1	any	sound-control
1	2	any	sound
1	3	any	midi
1	any	any	sound
3	1	1	keyboard
3	1	2	mouse
7	any	any	printer
9	any	any	hub
[POWER]	any	any	power
[MONITOR]	any	any	display-control
[COMM]	1	any	modem
[COMM]	2	any	modem
[COMM]	3	any	telephone
[COMM]	any	any	communications
[DATA]	any	any	data
any	any	any	interface

Note: The USB Device Working Group has not yet completed standardization of the Audio, Power, Monitor, Communications, or Data Interface Classes, and so interface class codes have not yet been assigned and the subclass codes are not final. The names above should be used, as appropriate, when the various specifications are complete.

`"reg"` S

Standard prop-name.

prop-encoded-array: two integers, each encoded as with encode-int.

The `"reg"` property for an interface node shall be constructed as follows:

The first integer shall contain the bInterfaceNumber from the INTERFACE descriptor with value 0 for bAlternateSetting associated with this USB interface.

The second integer shall contain the bConfigurationValue from the CONFIGURATION descriptor associated with this USB interface.

3.3.2.2. Bus-specific Properties for Interface Nodes

None.

3.3.3. Interface Node Methods

3.3.3.1. Open Firmware-defined Methods for Interface Nodes

None.

3.3.3.2. Bus-specific Methods for Interface Nodes

None.

3.4. Combined Nodes

A combined node is a special case node, combining some of the properties of both device nodes and interface nodes, and is typically used to simplify the representation of a simple USB device, with a single configuration and a single interface.

Neither a device node (see section 3.2.) nor an interface node (see section 3.3.) shall be created when a USB device reports in its DEVICE descriptor the following:

(1) bDeviceClass is 0 or 9, and
(2) bNumConfigurations is 1,

and reports in its CONFIGURATION descriptor the following:

(3) bNumInterfaces is 1.

Instead, a combined node shall be created.

3.4.1. Combined Node Address Representation

The textual representation of a combined node unit address shall be the number of the USB hub port or USB host controller port to which this USB device is attached, in lower case hexadecimal with leading zeroes suppressed.

3.4.2. Combined Node Properties

3.4.2.1. Open Firmware-defined Properties for Combined Nodes

The following standard properties, as defined in [1], have special meaning or interpretation for combined nodes.

The following notation is used:

From the DEVICE descriptor for this USB device:

VID	idVendor
PID	idProduct
REV	bcdDevice
DC	bDeviceClass
DSC	bDeviceSubClass
DPROTO	bDeviceProtocol

From the INTERFACE descriptor with the value of 0 for bAlternateSetting for this USB interface:

IC	bInterfaceClass
ISC	bInterfaceSubClass
IPROTO	bInterfaceProtocol

The textual representation of VID, PID, REV, DC, DSC, DPROTO, IC, ISC, and IPROTO shall be in lower case hexadecimal with leading zeroes suppressed.

`"compatible"` S

Standard prop-name.

prop-encoded-array; a list of strings, encoded with encode-string and concatenated with encode+.

Construct encoded strings as enumerated in the following list. All of the applicable strings shall be included in the value of `"compatible"`. Their relative order shall be as in the list, string 1) before string 2), and so forth.

Strings containing a device class (3..8) shall be omitted if the device class is 0. Strings containing an interface class (9..14) shall be omitted if the interface class is 0.

1) usbVID,PID.REV
2) usbVID,PID
3) usbVID,classDC.DSC.DPROTO
4) usbVID,classDC.DSC
5) usbVID,classDC
6) usb,classDC.DSC.DPROTO
7) usb,classDC.DSC
8) usb,classDC
9) usbifVID,classIC.ISC.IPROTO
10) usbifVID,classIC.ISC
11) usbifVID,classIC
12) usbif,classIC.ISC.IPROTO
13) usbif,classIC.ISC
14) usbif,classIC

`"name"` S

Standard prop-name.

prop-encoded-array: one string, encoded with encode-string.

The name of the node should be chosen from the following table, using the first name applicable:

bInterface Class	bInterface Subclass	bInterface Protocol	Name
1	1	any	sound-control
1	2	any	sound
1	3	any	midi
1	any	any	sound
3	1	1	keyboard
3	1	2	mouse
7	any	any	printer
9	any	any	hub
[POWER]	any	any	power
[MONITOR]	any	any	display-control
[COMM]	1	any	modem
[COMM]	2	any	modem
[COMM]	3	any	telephone
[COMM]	any	any	communications
[DATA]	any	any	data

bDeviceClass	bDeviceSubclass	bDeviceProtocol	Name
9	any	any	hub
[MASS]	1	any	storage
[MASS]	2	any	cdrom
[MASS]	3	any	tape
[MASS]	4	any	solid-state
[MASS]	any	any	storage
any	any	any	device

Note: The USB Device Working Group has not yet completed standardization of the Audio, Power, Monitor, Communications, or Data Interface Classes, and so interface class codes have not yet been assigned and the subclass codes are not yet final. The names above should be used, as appropriate, when the various specifications are complete.

Note: The USB Device Working Group has not yet completed standardization of the Mass Storage Device Class, and so a device class code has not yet been assigned and the subclass codes are not yet final. The names above should be used, as appropriate, when the Mass Storage Device Class specification is complete.

`"reg"` S

Standard prop-name.

prop-encoded-array: one integer, encoded as with encode-int.

The `"reg"` property for a combined node shall be the number of the USB hub port or the USB host controller port to which this USB device is attached. As specified in [2] section 11.11.2.1, port numbers range from 1 to 255.

3.4.2.2. Bus-specific Properties for Combined Nodes

`"assigned-address"`

prop-name to indicate the assigned USB bus address.

prop-encoded-array: one integer, encoded as with encode-int.

This property supplies the USB bus address assigned to this USB device. Each USB device that is assigned a USB address shall be assigned a USB address that is unique among all USB devices on this USB bus.

`"low-speed"`

prop-name to indicate that the USB device is low speed.

prop-encoded-array: None; presence or absence of the property conveys the information.

This property, if present, indicates that the USB device is low speed.

3.4.3. Combined Node Methods

3.4.3.1. Open Firmware-defined Methods for Combined Nodes

None.

3.4.3.2. Bus-specific Methods for Combined Nodes

None.

4. Hub Nodes

This section includes special requirements for devices implementing a hub node.

4.1. Common Requirements for Hub Nodes

This section contains the common requirements for hub nodes.

4.1.1. Hub Node Properties

4.1.1.1. Open Firmware-defined Properties for Hub Nodes

The following standard properties, as defined in [1], have special meaning or interpretation for hub nodes.

`"#address-cells"` S

Standard property to define the address format. Its value shall be 1, encoded as with encode-int.

`"#size-cells"` S

Standard prop-name to define the package's address size format.

prop-encoded-array: 0, encoded as with encode-int.

The value of `"#size-cells"` shall be 0, representing the fact that child device addresses are an enumeration rather than memory-like address ranges.

4.1.1.2. Bus-specific Properties for Hub Nodes

None.

4.1.2. Hub Node Methods

4.1.2.1. Open Firmware-defined Methods for Hub Nodes

A package implementing a hub node shall implement the following standard methods as defined in [1], with physical address representations as specified in section 3.2.1.

`decode-unit` (addr len -- port) M

Convert text unit-string to physical address.

`encode-unit` (port -- addr len) M

Convert physical address to text unit-string.

4.1.2.2. Bus-specific Methods for Hub Nodes

None.

4.2. Requirements for Combined Node Hubs

A combined node (see section 3.4.) which reports in its INTERFACE descriptor with value 0 for bAlternateSetting the following:

bInterfaceClass is 9,

is a combined node hub.

A combined node hub shall include all the properties, methods, and requirements of "Common Requirements for

Hub Nodes" as described in section 4.1

4.3. Requirements for Interface Node Hubs

An interface node (see section 3.3.) which represents a USB hub shall include all the properties, methods, and requirements of "Common Requirements for Hub Nodes" as described in section 4.1