Standard for Function and Low-level Service Discovery in IEEE Std. 1212 (FDS)

(Annex : Function Discovery and Function dependent Device driver information for IEEE Std.1394 devices)

Proposal Draft Ver.0.5
August 28,1997

PWG-C draft

Edited by A. Nakamura, Canon Inc.

date	version	revision notes
97/4/2	V0.2 (Canon)	original release
97/6/6	V0.3 (Canon)	section 6 renewed, DDP document ver.0.1
97/6/15	V0.4 (Canon)	renamed and rewritten based on PWG-C 6/11,12 meeting results
97/7/11	V0.4a (Canon)	modified based on PWG 6/23,24 meeting results
97/8/23	V0.49	modified for IEEE1212 reaffirmation and PWG-C draft
97/8/28	V0.5	minor modification.(parts in *italics*)

TABLE OF CONTENTS

1. INTRODUCTION

1.1 SCOPE

This document will describe an expansion to the IEEE1212(1394) standard for FDS (Function Discovery Standard) Specification which should apply to all buses that support the IEEE1212 architecture, for example the IEEE1394 High Performance Serial Bus. The specification will provide a flexible method for any device to comply to.
Chapter 4 will actually note where the enhancement should be made in the current IEEE1212 specification documentation.

1.2 PURPOSE

This purpose of this specification expansion is to define a universal, method for simple discovery of functions within a IEEE1212(1394) node, and low-level service discovery within each functions. What will be described is;

- A method for identifying a FDS compliant node.
- A method for identifying a functional unit (or multiple functional units) within a FDS compliant node.
- A method for retrieving information on each of the functional units.
- definition of FDS to support the above functions.

1.3 BACKGROUND (Problem Statement)

Even though the current IEEE1212-1994 specifies the configuration ROM format, thus allowing node discovery, it does not define a common method for discovering the function units within a particular node.
Currently, communication protocols focused on IEEE1394 communication (which uses IEEE1212) exist such as the Function Command Protocol (FCP) used with the AV command set, and the Serial Bus Protocol-2 (SBP-2) which acts as a lower-layer to command sets.
These protocols define the Unit_Spec_Id and Unit_Sw_Version fields, which results in categorizing IEEE1212 Unit_directories, or “units” by protocols.
Some of these communication protocol (stacks) do provide methods for discovering

the function of the node, but this is defined either by using that particular protocol, or defined among the vendor-specific entries, so they cannot be parsed without specific knowledge of the protocol selected. There may be other protocols that do not support function discovery at all.

In any case, a 1212-generic (protocol-independent) definition of a method for discovery of functional units and low-level service discovery is necessary and will be useful for any buses utilizing IEEE1212 such as IEEE1394.

Another necessity is the need for information to specify device drivers for host operating systems to load. Some host platforms categorize these device drivers by functions, so this kind of information should also be categorized by functions as well.

1.4 REFERENCES

- ANSI/IEEE Std. 1212 ISO/IEC 13213 Control and Status Registers (CSR) Architecture for microcomputer buses
- IEEE Std. 1394-1995, Standard for a High Performance Serial Bus
- IEEE Std. 1284-1994, Standard for Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers
- 1394-based Digital Camera Specification Version 1.04 August 9,1996
- AV/C Digital Interface Command Set Version 1.0 September 13,1996

2. DEFINITIONS

2.1 TERMINOLOGY

Datalink
In this document, a “datalink” will be defined as the software identified by the Unit_Spec_Id and Unit_Sw_Version fields, or assumed values defined in the IEEE1212 standard.

Transport
In this document, a “transport” will be defined as the set of communication protocol stacks as a whole. The set will consist of protocol layers ranging from the data-link layer to the application layer.

Node Discovery
In this document, a “node discovery” will be defined as discovering a IEEE1394 node and information which will include FDS protocol compliance. Other node information discovery will include manufacturer and ,model number, global unique ID of the nodes which are defined in the IEEE1394-1995 specification.

Function (unit) Discovery
In this document, a “function discovery” will be defined as discovering a functional unit (or multiple units) within a IEEE1212(1394) node. A unit or unit class will be categorized by the functionality. Examples of functions would be image output units such as printing functions, image source units such as scanning functions.

Low-level service Discovery
In this document, “Low-level service discovery” is defined as discovering the availability of the datalinks (=lowest layer above 1394 transaction layer), and the entry points of the datalinks.

Device Discovery
In this document, “device discovery” will consist of discovery of the following;

- Node Discovery
- Unit Discovery
- Low-level Service discovery

3. OPERATIONAL MODEL

3.1 Operational Model

A IEEE1212(1394) node will support the FDS and one or more communication protocols. The FDS is defined in this document, and examples of communication protocols may be FCP+AV/C or protocols using SBP-2 as the base layer. It may also be a device specific protocol. The FDS Protocol will be used to first discover the function unit (s) of the node, and secondly find out the communication protocols each unit supports.

A basic discovery scheme using FDS will take the following sequence.

1. A initiator device will look for (read) the defined key_value in the ROOT_DIRECTORY of the configuration ROM of the target node to discover a FDS compliant device node, and the pointer to the Function_list directory.
2. The initiator device will read out information for the address of the Function_list directory, which will store information on the function units available in the node.
3. The initiator will read out the Function_list directory and retrieve supported function units of the target node, and pointers for discriptor directories of each unit.
4. If further information is needed on a given function unit, the initiator will read out the leaf block of the function unit which pointer was given in the directory block. The function unit leaf block stores a list of datalink(s) supported by that particular function unit, and the pointer to the entry of that datalink. It will also store information on unit-uniuqe Ids(example: Plug and play strings)
5. Initiator device will enable the datalink that matches the capabilities of both the initiator and target.

4. FDS CONFIGURATION ROM DEFINITION

This section will describe the details of FDS.

FDS will define and provide 3 main functions:

1. Function 1: Node Function Discovery
2. Function 2: Low level service discovery of each function.
3. Function 3: Unique ID information retrieval of each unit

In other words, FDS will allow retrieving information on the functional units within a node, and information on each of the functional units such as the communication protocols each one supports, and some unit Ids unique to each unit.

Fig 4.1 shows the basic architecture and placement of FDS in the configuration ROM. (***Bold Italics***)

The following figure, 4.1 should be implemented in figure 52;"ROM hierarchy" of IEEE1212-1994

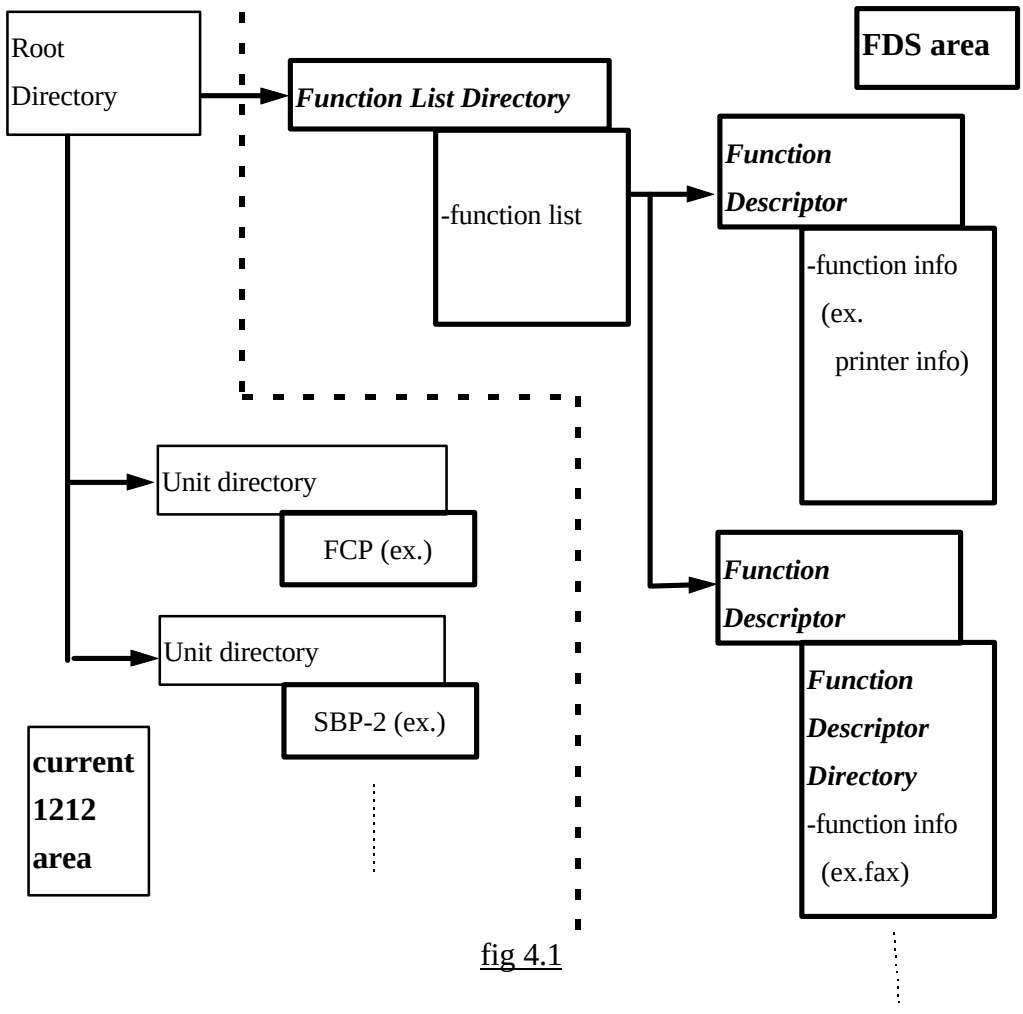


fig 4.1

PWG-C

The following section, 4.1.1 should be added as section 8.4.17 in IEEE1212-1994

4.1 Root Directory

4.1.1 FDS_directory

Used to provide a list of functions within a node.

The Function_list_directory offset entry points to a Function_list directory that contains information on functions within a node.

key_type *2*	key_value *6*	value *24*
11b	**17h**	Function_list_directory offset

The following section, 4.2 should be inserted as section 8.6 in IEEE1212-1994

4.2 FDS Function_List Directory and Function_Descriptor Directories

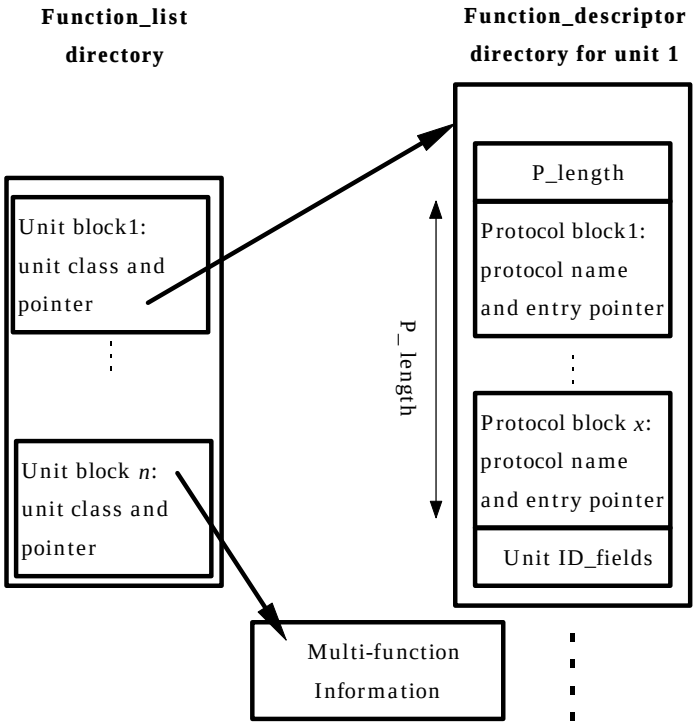
The FDS Register map of a node consists of 2 main parts; 1 Function_list directory and 1 or more Function_descriptor directories.

The **Function_list directory** will store information on the function_units within a node. In detail, it will give information on ;

- a list of the function_class of each functional unit in the node
- pointer to the Function_descriptor directory for each function unit

The **Function_descriptor directory** will store information of each of the function_units within a node. In detail, it will give information on ;

- Protocols supported by the functional unit, and their entry pointers
- entry pointer of each of the protocols



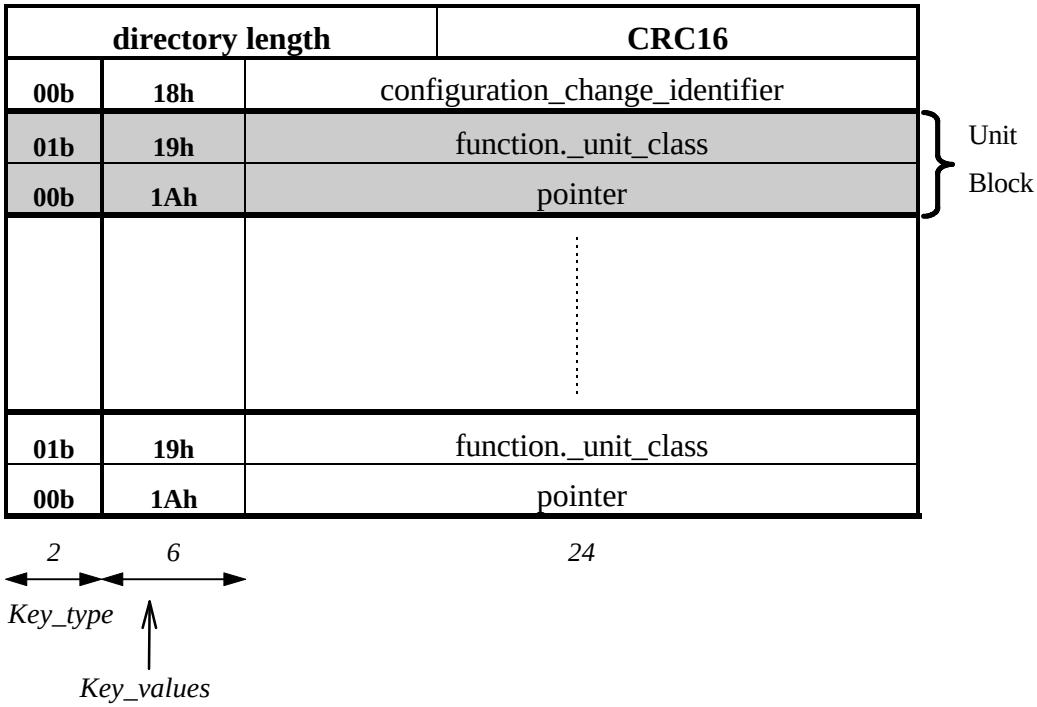
Nodes with multiple function units will have multiple unit blocks, and units supporting multiple datalinks will have multiple protocol blocks.

Information of the multi-function units as a whole (multi-function information) can also be stored in the FDS as well.

4.2.1 Function_ List Directory (function unit discovery)

Address offset locations noted in this section are with respect to a base address noted in the FDS_Dependent_Directory offset in the Root Directory.

Function_ List Directory format



Function_ List Directory entries

Field	key value	Description
Configuration change identifier	18	value of state-change random counter
Function._unit_class	19	functional class of the unit *0* : *complete* ~~proprietary~~ node funct. *1* : others 2 : proprietary function 3 : printing function
Pointer	1A	pointer address of the Function_descriptor directory

4.2.1.1 Configuration_change_identifier- Key Value : 18h

The Configuration change identifier field is a immediate value field which shows a value of a state-change "random" counter. A state-change counter **must** change it's value when there is a configuration change in the function unit directory or the function unit leaf of any of the functional units.

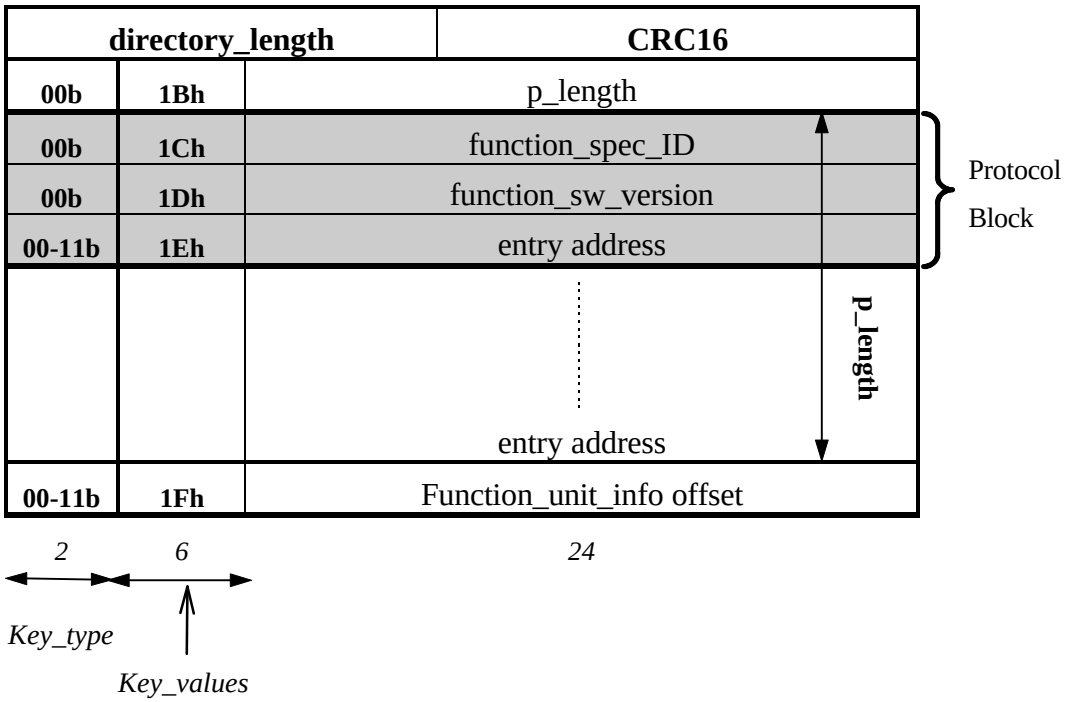
4.2.1.2 Unit block (Pointer / Function._unit_class) - Key Value : 19h,1Ah

The Pointer field and Function._unit_class field are the pair of fields that make up a unit block for each function unit in the Function_list directory block. The immediate value of the Function._unit_class field will represent the functional class of the unit, and the value of the pointer field will represent the pointer address of the Function_descriptor directory of the unit it represents.
A Function_unit_class field with a value of 0 will be used in the case of a multi-function node to address the function of the whole node, instead of one of it's functions.

4.2.2 Function_ Descriptor Directory (low-level service discovery)

Address offset locations noted in this section are with respect to the pointer address of each functional_unit noted in the Pointer field of the Function_list directory block. (section4.2.1)

Function_ Descriptor Directory format



Function_ Descriptor Directory entries

Field	key value	Description
P_length	1B	length of protocol blocks in quadlets
function_spec_ID	1C	The unit_spec_id value of the datalink supported by the unit.
function_sw _version	1D	The unit_sw_version value of the datalink supported by the unit.
entry_address	1E	address for datalink entry.
Function_Unit _Info offset	1F	Function unit ID

4.2.2.1 P_length- Key Value : 1Bh

The P_length field is a immediate value field that is used to inform the total length of the protocol blocks in this Function_descriptor directory. The value of this field will represent the field length in number of quadlets. (refer to above register map) If a Protocol block is not used, the value of the P_length will be 0.

4.2.2.2 Protocol_block (function_spec_ID, function_sw_version, entry_address) - Key Value : 1Ch,1Dh,1Eh

The Protocol block is comprised of 2 parts;

1) the Protocol definition fields, and
2) the entry_address field with a variable key_type(used to specify the characteristics of the entry_address field).

The **protocol definition fields** consist of 2 fields that when prepended, will specify the datalink ;

1: The **function_spec_ID field** is a immediate value field that will have the same value used in the **unit_spec_id, node_spec_id, or the module_spec_id** field of the configuration ROM that best describes the datalink. (Usually a unit_spec_id in the unit_directory defined in each datalink specification.)

2: The **function_sw_version field** is a immediate value field that will have the same value used in the **unit_sw_version, node_sw_version, or the module_sw_version** field of the configuration ROM that best describes the datalink. (Usually a unit_sw_version in the unit_directory defined in each datalink specification.)

In case * spec_id or *** sw_version are not provided in a particular node**, the values of function_spec_ID will be the assumed value for unit_spec_id, and function_sw_version field will be the assumed value for unit_sw_version . Assumed values for unit_spec_id and unit_sw_version are defined in the ISO/IEC 13213, ANSI/IEEE Std 1212 document.

The optional **entry_address field** with the variable **key_type** will inform the entry offset address of the datalink noted above.

4.2.2.3 Function_Unit_Info

The **Function_Unit_Info** is a directory or a leaf containing information unique to the function.

The **Function_Unit_Info offset field** with the variable **key_type** specifies a directory or a leaf used to inform an unique ID of the function unit. The contents of the Unit_info ID field will include information such as a field that will follow the format of the Device ID field defined in section 7.6 of the IEEE std 1284-1994.

FYI : From IEEE std 1212 document:

from section 8.1.3.....Driver and diagnostic identifiers

.....The arows in figure 53 illustrate the default values fir various company_id values. For example, when Node_Spec_Id is not provided, its assumed value shall be equal to Node_Vendor Id. Similary when Node_Vendor_id is not provided, its assumed value shall be equal to Module_Vendor_Id.

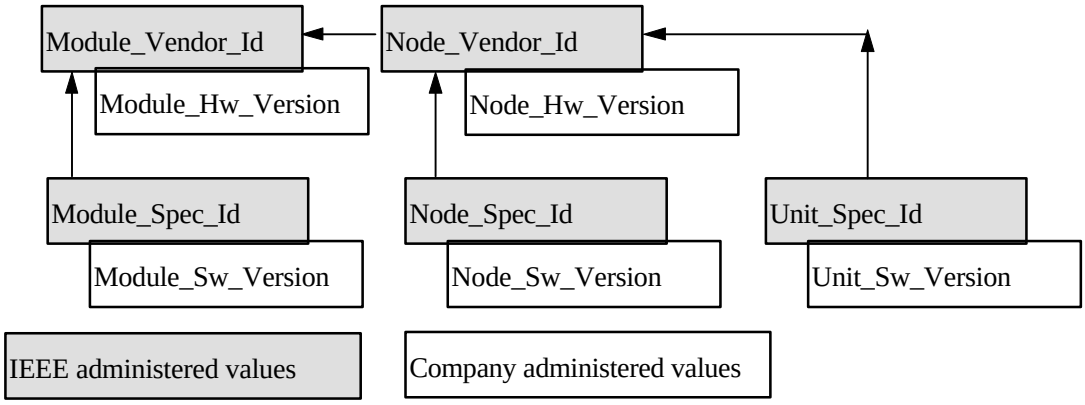


Fig.53

.....The Module_Sw_Version, Node_Sw_Version, and the Unit_Sw_Version values (when concatinated with their respective specifier identifier,such as Module_Spec_Id) are expected to uniquely identify the appropriate I/O driver software for the module, node, unit, respectivel y.

4.3 Bus Reset - Reconnection

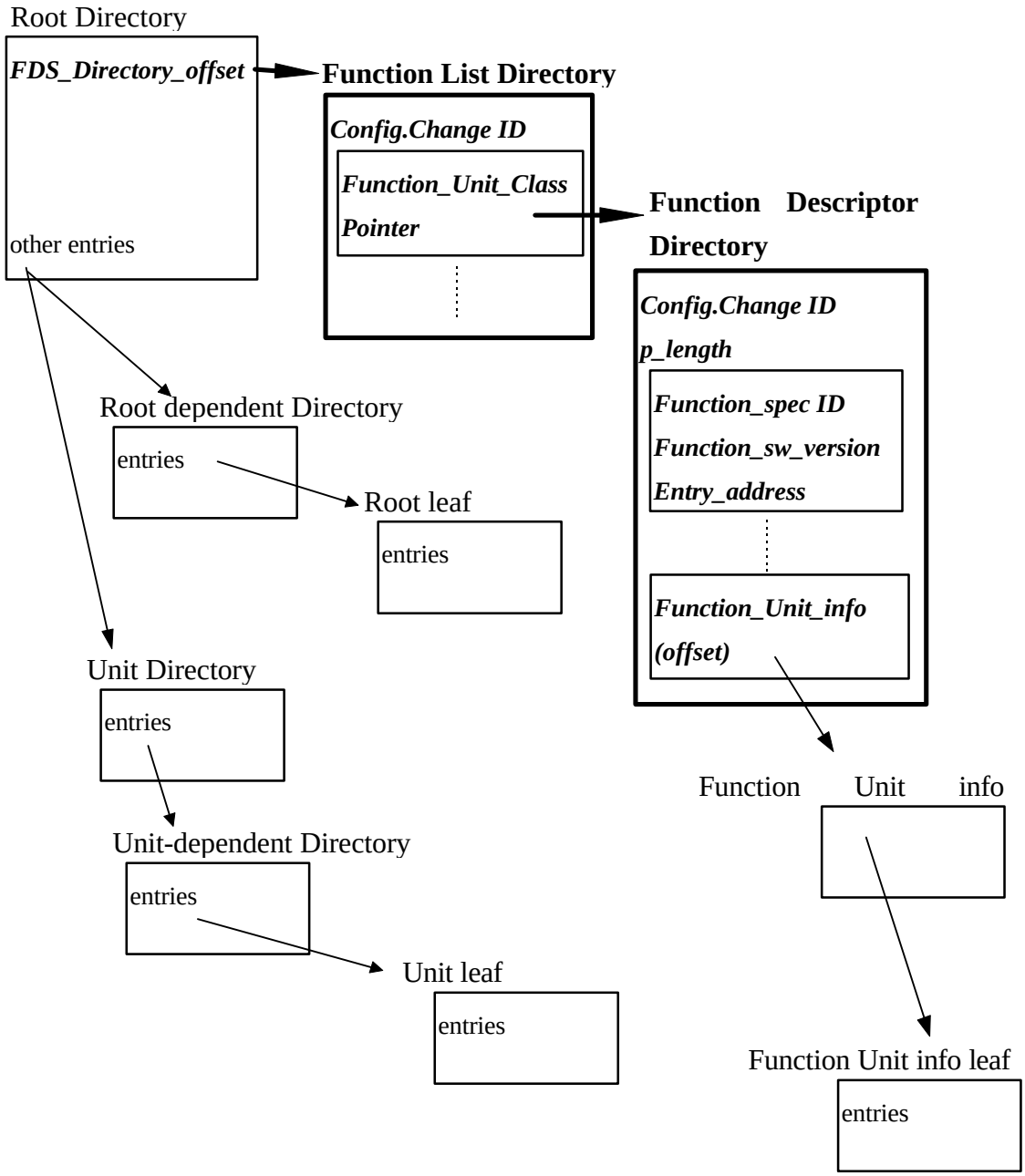
Values of the FDS fields that will dynamically change;

1. shall not change during bus reset
2. shall be updated by the device upon reconnection with a time limit of 1sec after bus-reset is cleared. Methods of updating are beyond the scope of the proposal.

Devices connecting to FDS compliant nodes shall keep track of the **values of the Configuration-state counter field** for any changes in configuration change of the node.

5. IEEE1212 FDS SUMMARY

The diagram below shows a basic ROM structure.
Items in **BOLD** are FDS enhancements to existing 1212 definitions.
Items in ***BOLD ITALICS*** are FDS newly-added entries to existing 1212 definitions.



The table below lists the newly-added entries and key_values to existing 1212 definitions.

Root Directory entries

Field	key value	Description
Function_list_ directory offset	17	pointer address of the Function_list directory

Function_ List Directory entries

Field	key value	Description
Configuration change identifier	18	value of state-change random counter
Function._unit_class	19	functional class of the unit
Pointer	1A	pointer address of the Function_descriptor directory

Function_ Descriptor Directory entries

Field	key value	Description
P_length	1B	length of protocol blocks in quadlets
function_spec_ID	1C	The unit_spec_id value of the datalink supported by the unit.
function_sw _version	1D	The unit_sw_version value of the datalink supported by the unit.
entry_address	1E	address for datalink entry.
Function_Unit _Info offset	1F	Function unit ID

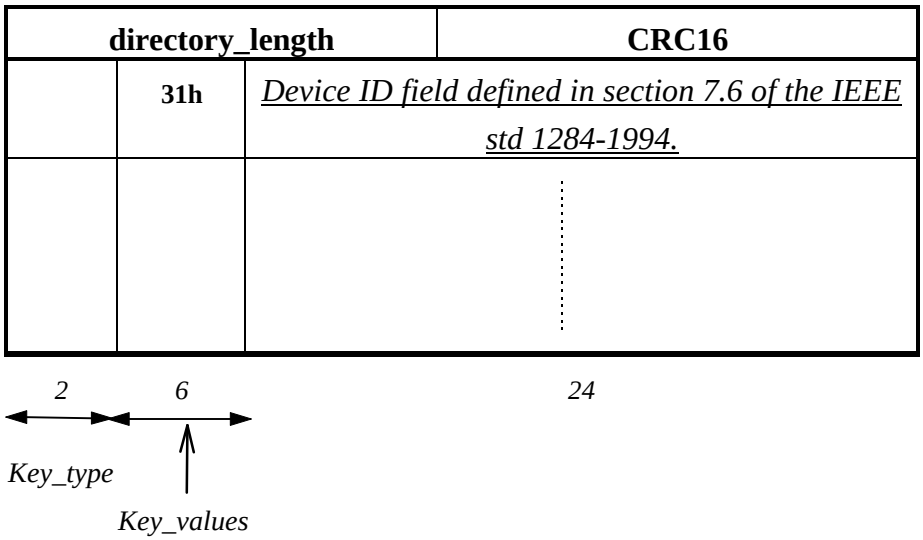
ANNEX A:
FUNCTION DISCOVERY and DEVICE DRIVER INFORMATION FOR IEEE1394 DEVICES

TBD

Devices using the IEEE1394 bus will support FDS of IEEE1212

IO software driver information for each function within IEEE1394 nodes will be stored in the **Function_Unit_Info directory/leaf** and the offset for this leaf will be stored in the **Function_Unit_Info field** of the **Function descriptor directory**.
(See 4.2.2.3 of this document)

IEEE1394 Function_ Unit Info Directory format



FYI: The IEEE1394 Specification for Power Management has it's own (special) root directory offset entry using the key_value of F0h(concationation of 3h and 30h) which 30h-37h is reserved for definition by the bus standard identified in BUS_INFO_BLOCK. (Which is IEEE1394 in this case.)

.....31h for device ID strings for 1394 printers?

This contents of information will at least include contents of the *Device ID field defined in section 7.6 of the IEEE std 1284-1994.*