Function Discovery Service for 1394 Nodes (FDS)

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

Proposal Draft Ver.0.6
September 30,1997

PWG/PWG-C

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*)
97/9/30	V0.6	Revised format of document per 9/17 meeting

TABLE OF CONTENTS

0. INTRODUCTION-OVERVIEW

This proposal is a collaborative effort of the 1394 Printer Working Group and the PWG-C. These groups are defining solutions for 1394 printing devices which will operate with traditional computing applications and new areas such as direct printing from digital still cameras and image scanners. As part of this effort, a need for identifying the functions supported by a given node was identified.

This proposal describes a Function Discovery Service (FDS) for nodes connected to a 1394 High Speed Serial Bus. FDS is a protocol neutral mechanism to identify various functions within a node, identify the appropriate protocols required and a mechanism for providing device information in human readable form in addition to existing binary fields of the Configuration ROM.

Even though the current IEEE1212-1991 specification defines the configuration ROM format, thus allowing node discovery, it does not define a common method for discovering the functions within a particular node. The current interpretation of a Unit Directory within a 1394 node is specified by the Unit_Spec_ID and Unit_SW_Version entry in the Unit Directory, which values are defined by the "vendor", or organizations defining the protocol. Further, the structure and entries within the Unit directories are also "vendor", or protocol dependent.

If a node supports multiple functions, it may be represented via multiple unit directories or via subunit architectures. IEEE-1212 and IEEE-1394 acknowledge this implementation option, yet, they do not specify a preference.
There are also cases where a single function which can be supported by multiple protocols will be represented in multiple unit directories regardless of the fact that it is a single function.
As more device classes, unit directory fields and various protocols are defined and implemented on 1394 it will become increasingly complex to enumerate all of the functions supported within the node.

In order to simplify the task of a client in identifying useful nodes with a minimum of bus transactions, FDS proposes a Function Directory structure which is separate from the Unit directories which can be used to identify the function(s) supported on a given node.

By defining new keys in IEEE-1212 and defining the exact usage within a 1394 device class specification, it is possible for clients to identify nodes which contain the appropriate functions and corresponding protocols . Further, existing client software installation techniques support existing identification mechanisms which can be used for 1394 nodes. The Function Directory structure provides a location to store these identification strings. The only protocol requirement for a client to make use of the FDS structures is the ability to perform basic read4 operations on the 1394 bus.

Open Issues:

1. Current proposal calls for a Configuration_change_identifier.
    - Same function is provided for in 1394a spec with generate field in Bus Info Block. Consider removal of Configuration_change_identifier from this proposal and reference 1394a solution.
    - From a global point of view, Configuration_change_identifier is just for FDS section. The purpose is to advise other interested nodes that this node has changed and ConfigROM needs to be read again.
2. Function categorization

    Currently represented by a pair of FUNCTION_CLASS & FUNCTION DESCRIPTOR(directory)

    Central registration authority needed for all "FUNCTION CLASSES"

    (IEEE RAC? others)

    Contents (values) for FUNCTION DESCRIPTOR(directory)
    - Currently same as Unit_Spec_ID and Unit_Sw_Version plus a entry point field that can be used either to point to a Unit directory OR an entry point.
    - Should this just be as is or a pointer to the appropriate Unit directory?
    - Former method will enable block read of all supported protocols, but might still require traversing all unit directories to find further information on the corresponding protocol while latter method points to the exact unit directory.
    - Pointing to unit directories require that unit directories are mandatory for all existing, future, and vendor-unique protocols. Current rules do not state this. Should the 1212 make a strong recommendation to use the unit directory structure for any future (protocol) designs?
3. Informative - Function Unit Info field
    - Usage and location is TBD

- PWG is discussing content such as legacy PNP string.
- IEEE-1284-1994 defines an ASCII string format which may be usable.
- Is it desirable to allow a client to read the information all at once via a block read from the node or should this information be located in multiple locations and the client must check all possible variations.
- This section will be made informative at this time.

1. SCOPE,PURPOSE,REFERENCES

1.1 SCOPE

This document will describe a Function Discovery Service (FDS) for nodes on the IEEE1394 High Performance Serial Bus. FDS requires additions to the IEEE1212-1991 CSR architecture standard. FDS may apply to other buses that support the IEEE-1212 CSR architecture, however, that is beyond the scope of this document.
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 is to define a protocol neutral method for discovery of functions within a IEEE 1394 node and low-level service discovery related to each function. 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 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 Bi-directional 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
- Serial Bus Protocol 2 (SBP-2) T10 Project 1155D Revision 2g
- P1394a Draft Standard for a High Performance Serial Bus (Supplement) draft 1.0

2. DEFINITIONS

2.1 TERMINOLOGY

Function
The service provided by a unit architecture of a node. Examples of functions would be image output units such as a printer or image source units such as a scanner or camera.

* All other terminology refer to the IEEE1212-1991, 1994 documentation.

3. OPERATIONAL MODEL

3.1 Operational Model

Usually, a node will support at least 1 function which operates using one or more communication protocols. Examples of functions may include cameras, printers, etc. Examples of communication protocols, in the case of IEEE1394 may be FCP+AV/C or protocols using SBP-2 as the base layer. It may also be a vendor-unique protocol. FDS can be used to first discover the function (s) of the node, and secondly find out the communication protocols each function supports.

An example of 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 entry which will be a point to the Function list directory.
2. The initiator will read out the Function_list directory and retrieve supported functions of the target node, and pointers for Function descriptor directories of each function.
3. If the initiator succeeds to discover an intended function, it will then read out the Function descriptor directory of the function which pointer was given in the Function List directory. The Function Descriptor directory stores a information on the communication protocols(s) supported by that particular function. It will also store additional information on that function. (example: device ID strings)

Actual Key_Value numbers noted are tentative

4. FDS CONFIGURATION ROM DEFINITION

This section will describe the details of FDS.

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

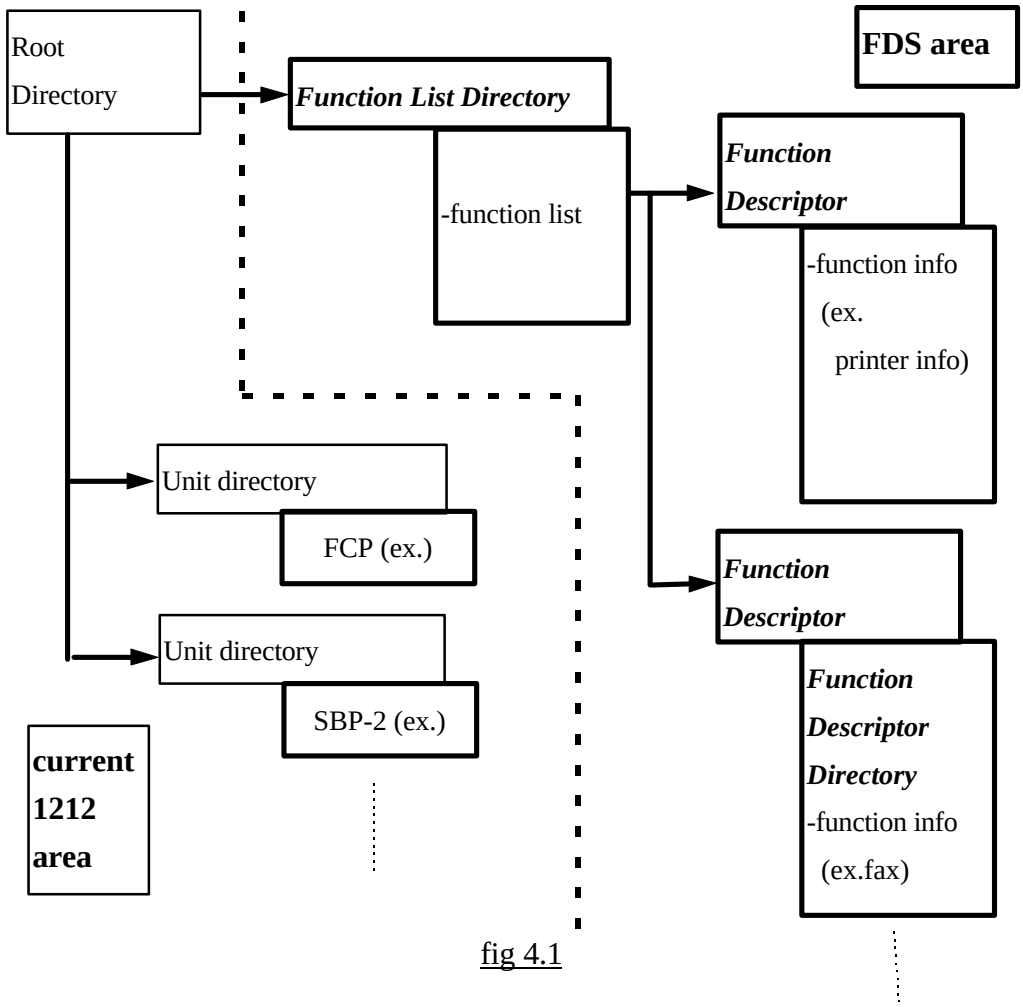


fig 4.1

4.1 Root Directory

The following table should be added in Table 28 in IEEE1212-1994

Entry name	Entry description
Function_List_Directory	Function list and information

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

4.1.1 FDS_directory(s)

Used to provide information on functions that the node supports.

The Function_list_directory entry points to a lower-level directory (Function_list_Directory) that contains a list of functions within a node, which further points to a lower-level directory (Function_Descriptor_Directory) that contains information on each function.

The following table should be added in Table 35 in IEEE1212-1994

Entry name	key_type(s)	key_value
Function_List_Directory	directory	17 (TBD)

The following section, 4.2 should be inserted as section 8.6 in IEEE1212-1994 (before "8.6 Key definitions" in current 1212 document)

4.2 FDS Function_List Directory and Function_Descriptor Directories

A Function List directory is required for each node, and a corresponding Function descriptor directory is required for each function listed in the Function List directory . All of the defined entries of the Function List directory are shown in table XX, and all of the defined entries of the Function Descriptor directory are shown in table XX.

Table XX - Function_ List Directory entries

Entry name	Entry description
Configuration_Identifier	ROM state identifier
Function._Class	functional class of the unit
Function_Descriptor_Directory	pointer to corresponding Function_descriptor directory

Table XX - Function_ Descriptor Directory entries

Entry	Description
P_Length	length of protocol blocks in quadlets
Function_Spec_ID	The unit_spec_id value of the datalink supported by the unit.
Function_Sw_Version	The unit_sw_version value of the datalink supported by the unit.
Sw_Location	address for datalink entry.
Function_Unit_Info	Function's additional information

The **Function_list directory** shall contain a list of functions within a node. In detail, it will give information on ;

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

The **Function_descriptor directory** shall contain information of a function within a node. In detail, it will give information on ;

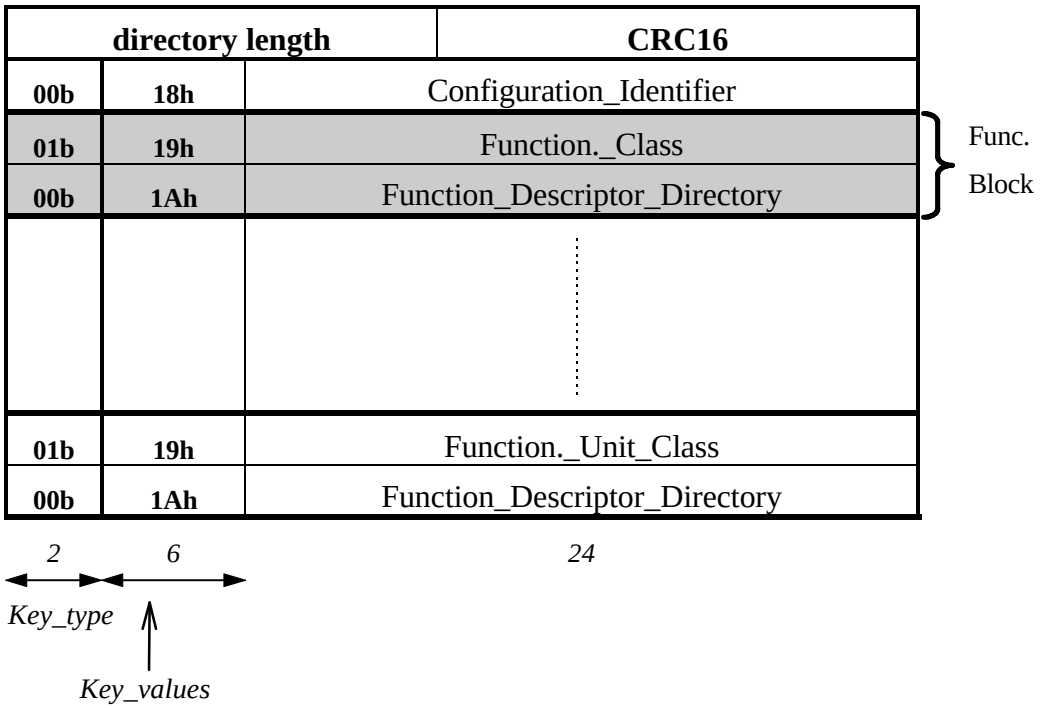
- Unit architectures, or I/O driver software supported by the function
- entry pointer of each of the I/O driver software

Information of the multi-functions addressed as a whole (multi-function information) can also be stored in the Function List / Function Descriptor.

4.2.1 Function_List Directory

The Function_List Directory shall have the format specified in figure XX below.

Figure XX - Function_ List Directory format



4.2.1.1 Configuration_identifier- Key Value : 18h

The Configuration identifier entry is a entriesupplying a value related to information on configuration of the FDS directories within the ROM.(i.e. Informative values representing change in the configuration.) The detailed usage of this entry will be dependent to the bus.

Devices connecting to FDS compliant nodes shall keep track of the **values of the Configuration identifier entry** for any changes in configuration of the FDS directories.

4.2.1.2 Function._Class - Key Value : 19h

Used to identify a function within the node.

The immediate value of the **Function._Class entry** will represent the functional class of the function. 1 Function_Class entry should be present for each function within a node.
A Function_Class entry 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.

0 : complete node function.
1 : proprietary function (others)
2 : printing function
3 :

4.2.1.3 Function_Descriptor_Directory - Key Value : 1Ah

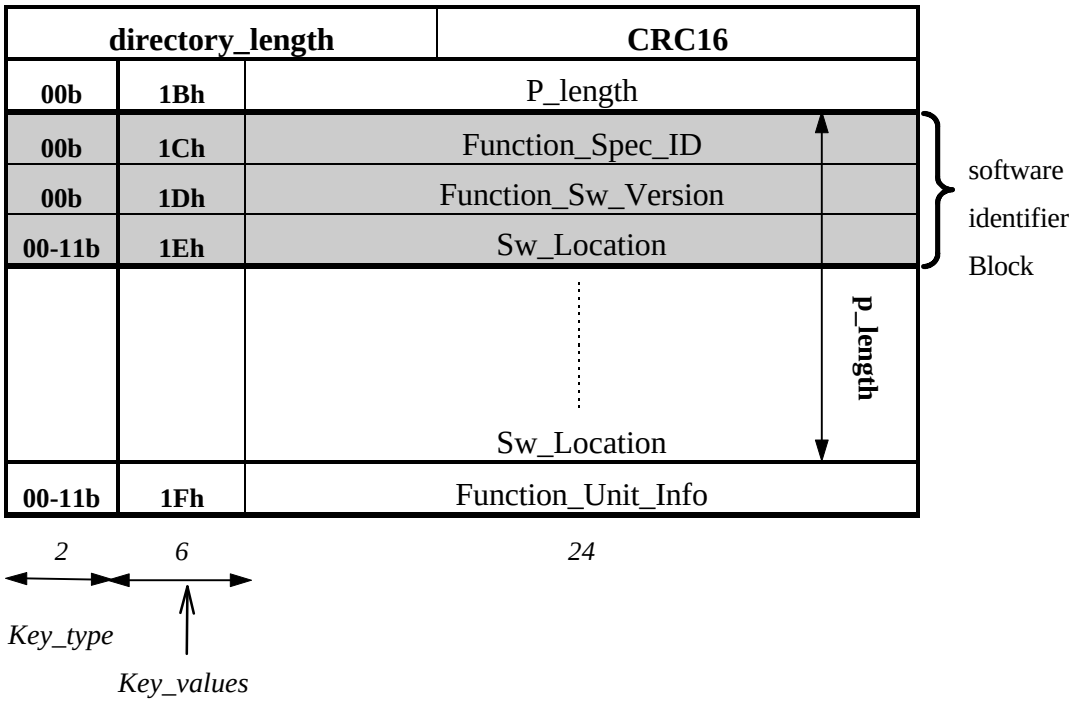
Required for each Function_Class entry present
Used to provide information about the function.

The **Function_Descriptor_Directory entry** points to a lower-level directory (Function Descriptor directory) that contains information of the function identified in the corresponding Function_Class entry.

4.2.2 Function_ Descriptor Directory

The Function_List Directory shall have the format specified in figure XX below.

Figure XX - Function_ Descriptor Directory format



4.2.2.1 P_Length- Key Value : 1Bh

The 24 bit value of the P_Length entry provides the total length of the software identifier blocks in the Function Descriptor directory. The value of this entry will represent the field length in number of quadlets.
If a Protocol block is not used, the value of the P_Length shall be 0.

4.2.2.2 Software Identifier Block(s)

1 Software Identifier Block is comprised of 3 entries;

1) Function_Spec_ID
2) Function_Sw_Version
3) Sw_Location

A function supporting multiple unit architectures, or I/O driver software shall implement 1 Software Identifier block for each I/O driver software.

4.2.2.2.1 Function_Spec_ID- Key Value : 1Ch

The 24 bit immediate value of the **Function_Spec_ID entry** provides the 24 bit company_id of the vendor defining the architectural interface for the function.

This entry should provide the same 24 bit company_id value used in the unit_spec_id that applies to the architectural interface supported by the function. (or the assumed value in case unit_spec_id is not implemented.)

4.2.2.2.2 Function_Sw_Version- Key Value : 1Dh

Identifies the I/O driver software interface architecture for the function.
The 24 bit immediate value of the **Function_Sw_Version entry** provides a unit-software identifier for the function.

This entry should provide the same 24 bit company_id value used in the unit_sw_version that applies to the software identifier supported by the function. (or the assumed value in case unit_sw_version is not implemented.)

This Function_Sw_Version number, when perpended with the Function_Spec_Id value, is expected to identify uniquely 1 unit architecture for this function, and therefore 1 supported I/O driver software for this function.

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_entry will be the assumed value for unit_sw_version . Assumed values for unit_spec_id and unit_sw_version are defined in Figure 53 in the ISO/IEC 13213, ANSI/IEEE Std 1212 document.

4.2.2.2.3 Sw_Location

Used to provide either additional information about the unit architecture specified by the Function_Spec_ID and Function_Sw_Version, or a base and bound address for the unit architecture or I/O driver software specified by the Function_Spec_ID and Function_Sw_Version.

When this entry is used as a pointer to a directory, it should point to the corresponding unit directory of the unit architecture specified. In this case, the key_type shall be a directory.
When this entry is used to provide a a base and bound address for the unit architecture, the format and usage shall be equal to the Unit_Location entry (Section8.5.4). In this case, the key_type shall be a leaf.

4.2.2.3 Function_Unit_Info

Used to provide additional information about the function.

The **Function_Unit_Info entry** points to a lower-level directory or a leaf that contains function-specific information. The contents and usage of this lower-level directories or leafs are vendor-dependent.

FYI : From IEEE std 1212 document:

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

.....The arows in figure 53 illustrate the default values for 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.

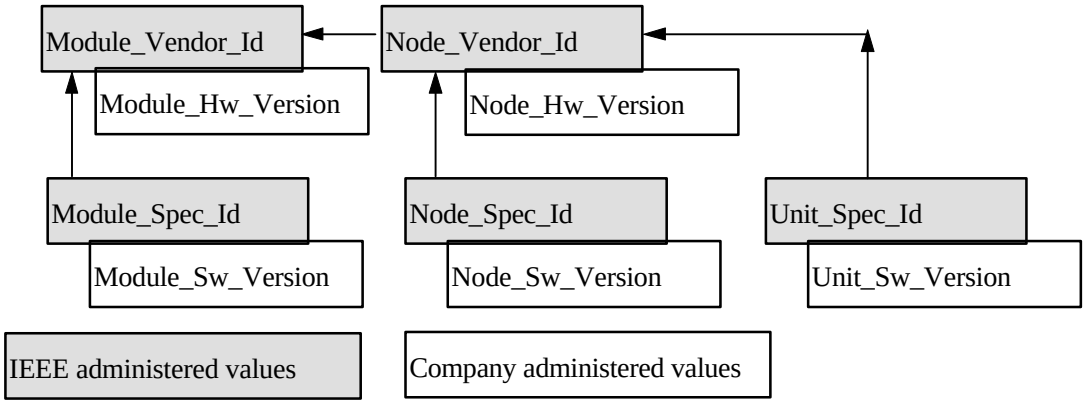
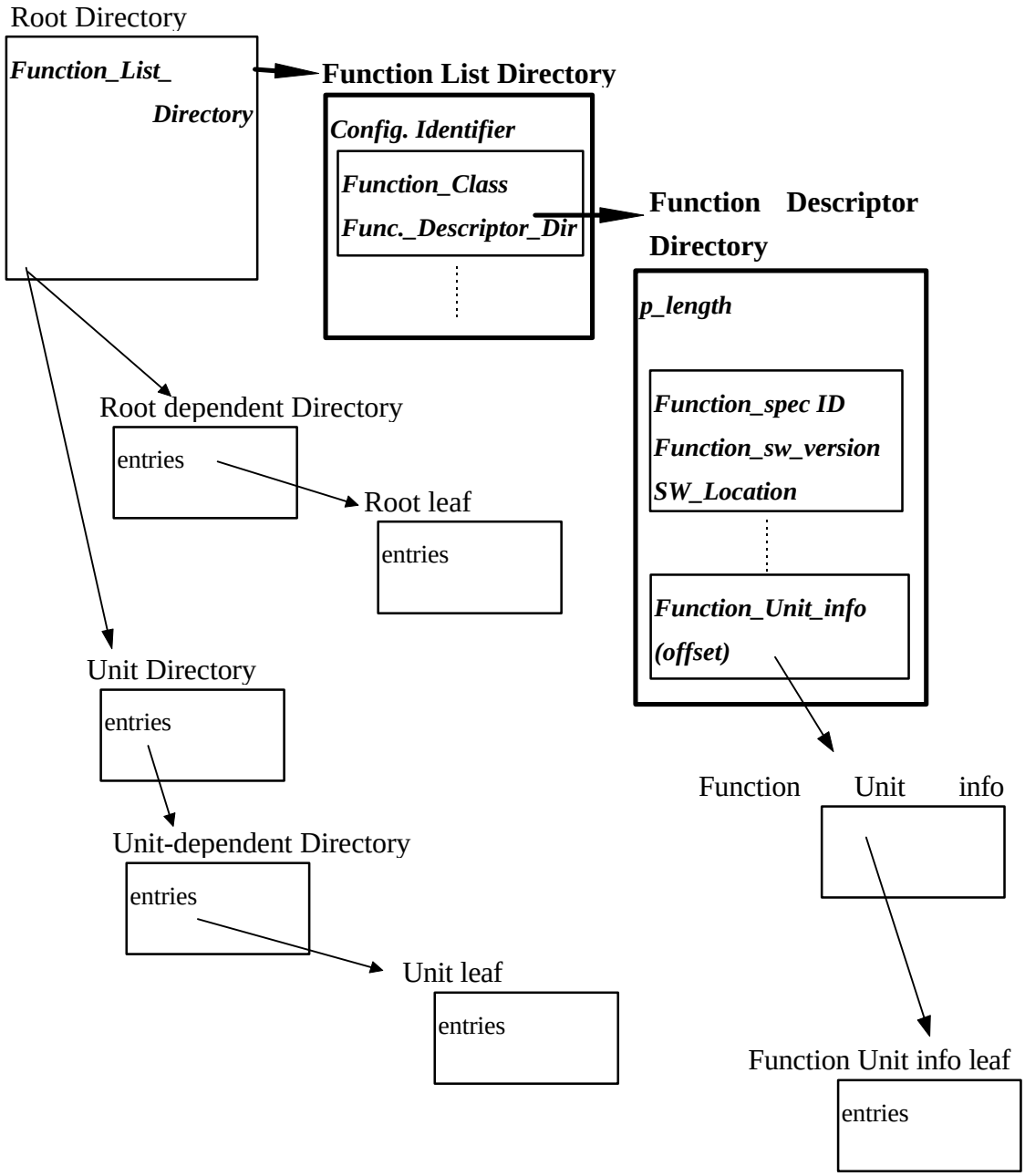


Fig.53

.....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.

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

Entry	key value	key_type(s)
Function_list_directory	17	directory

Function_ List Directory entries

Entry	key value	key_type(s)
Configuration_ identifier	18	immediate
Function._unit_class	19	immediate
Function_Descriptor_Directory	1A	directory

Function_ Descriptor Directory entries

Entry	key value	key_type(s)
P_Length	1B	immediate
Function_Spec_ID	1C	immediate
Function_Sw_Version	1D	immediate
Sw_Location	1E	leaf or directory
Function_Unit_Info	1F	leaf or directory

INFORMATIVE:

FUNCTION DISCOVERY and DEVICE ID STRING INFORMATION FOR IEEE1394 PRINTING DEVICES

TBD

IEEE1212 FDS COMPLIANCE

Printing devices using the IEEE1394 bus shall implement the Function List directory and the Function Descriptor directory (FDS) of IEEE1212

FUNCTION_UNIT_INFO LEAF

A device ID string defined in section 7.6 of the IEEE std 1284-1994 shall be stored in the **Function_Unit_Info leaf** and the offset for this leaf will be stored in the **Function_Unit_Info entry** of the **Function descriptor directory**.
(See 4.2.2.3 of this document)

IEEE1394 Function_ Unit Info Leaf format

leaf_length	**CRC16**
Device ID entry defined in section 7.6 of the IEEE std 1284-1994.	

32

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 entry defined in section 7.6 of the IEEE std 1284-1994.*