July 2007

STS LLDP-MED Software Module

Link Layer Discovery Protocol - Media Endpoint Devices

INCA-IP2 (PSB 21653), V1.5
Software Release 1.0

Preliminary

User's Manual

Module Software Description
Revision 1.0

Communication Solutions

infineon

Never stop thinking

Edition 2007-07-03

Published by
Infineon Technologies AG
81726 München, Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

STS LLDP-MED Software Module Link Layer Discovery Protocol - Media Endpoint Devices
CONFIDENTIAL
Revision History: 2007-07-03, Revision 1.0

Previous Version:

Page	Subjects (major changes since last revision)

Trademarks

Table of Contents

List of Figures

1 Preface

This document is intended to describe the functionality of the IFX LLDP software module to be used with the INfineon Codec with Acoustic echo cancellation - Internet Protocol (INCA-IP2) hardware platform.

The Link Layer Discovery Protocol (LLDP) is a vendor neutral layer-2 protocol that is used by stations attached to a specific LAN segment to advertise their identity and capabilities and also to receive same from a physically adjacent layer-2 peer. The Link Layer Discovery Protocol is an IEEE standard (802.1AB) that was ratified in May 2005 **[1]**.

Especial for IP Phone systems, the IFX LLDP software module incorporates support also for media endpoint discovery as defined by ANSI/TIA-1057 (LLDP-MED) **[2]**.

The document is divided into the following sections:

- **Introduction** - Overview that describe the feature set, architecture and module description.
- **Source File Organization** - Details to the building process and source file organization.
- **Module Interface Description** - Description of the LLDP module API.

2 Introduction

The INCA-IP based LLDP Agent is a software implementation of the LLDP and LLDP-MED (Link Layer Discovery Protocol - Media Endpoint Devices) specifications. This entity is used to multicast the identity and supported capabilities of the endpoint to other endpoints inside a LAN. The same entity can be used to receive other endpoints' identity and capabilities which are inside the same LAN. By implementing the MED extensions, it is possible to advertise the VoIP specific capabilities to other LAN entities. LLDP Agent uses the LSAP (LLC Service Access Point) for sending and receiving LLDP packets.

2.1 Feature List

The LLDP software module supports following features:

IEEE 802.1 AB (LLDP)

- Mandatory TLV:
  - Chassis ID
  - Port ID
  - Time To Live
  - End of LLDPDU
- Basic management TLV:
  - System Capabilities
- IEEE 802.1 Organizationally Specific TLV:
  - Port VLAN ID
  - Port and Protocol VLAN ID
  - VLAN Name
  - Protocol Identity
- IEEE 802.3 Organizationally Specific TLV:
  - MAC/PHY Configuration/Status

TIA 1057 (LLDP-MED)

- Endpoint Class III
  - LLDP-MED Capabilities
  - Network Policy
  - Extended Power via MDI

2.2 Architecture

Figure 1 depicts the architecture of the IFX LLDP module.

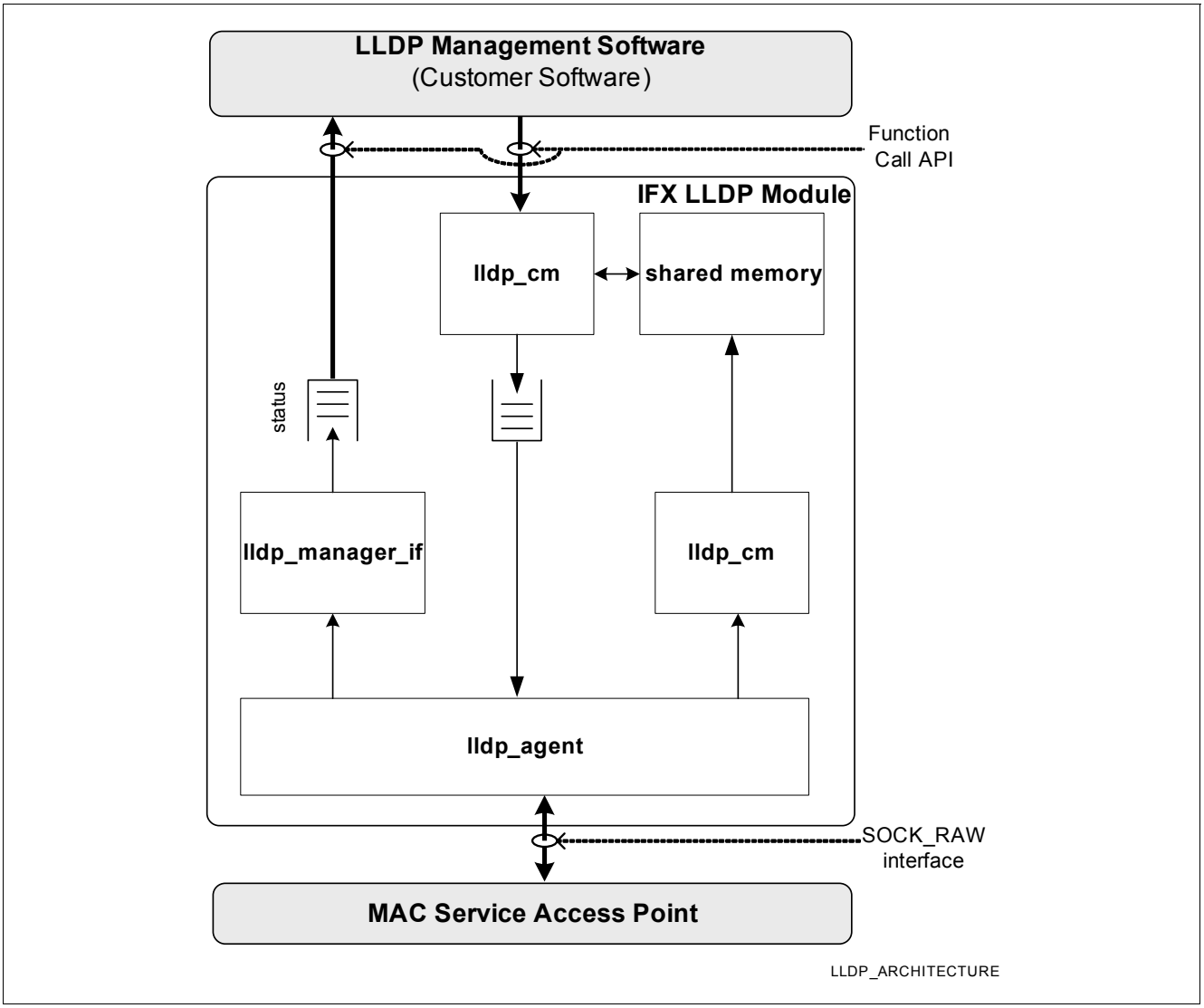

Figure 1 LLDP Module Architecture

The LLDP module consists of following modules:

- LLDP Agent (lldp_agent)
- LLDP Configuration Module (lldp_cm)
- LLDP Manager Interface

LLDP Agent

This sub-module implements the core LLDP Protocol and it's MED extensions. Transmission, reception and management of all the LLDP packets are done here. It internally maintains transmit and receive FSMs (Finite State Machine) for each port separately. It uses the Configuration Module for the systematic storage of the local configuration data and remote LLDP packets. LLDP Agent maintains a FIFO based interface with the manager.

Configuration Module

This module stores all local system configuration parameters and remote system LLDP data in a shared memory so that it can be directly accessed from LLDP Agent and LLDP Manager.

LLDP Manager Interface

This module is used to post messages into the LLDP manager FIFO when a remote LLDP packet is received/expired.

2.3 Functional Description

When the LLDP module is configured as a LLDP-MED Communication Device Endpoint (Class III), following operations are supported:

- Initialization (**IFX_LLDP_MODULE_INIT**, **IFX_LLDP_PORT_INIT**, **IFX_LLDP_ENABLE_PORT**)
- Extend Database (**IFX_LLDP_TLV_ADD**)
- Local TTL expired (internal standard processing; no interaction with the LLDP module required)
- Remote TTL expired (**IFX_LLDP_PACKET_EXPIRED**)
- New/Changed System settings (**IFX_LLDP_TLV_DEL**, **IFX_LLDP_TLV_CHANGE**)
- Query Database (**IFX_LLDP_PDU_REQUEST**,**IFX_LLDP_MIB_REQUEST**)
- Shutdown (**IFX_LLDP_SHUTDOWN**).

Figure 2 shows some example of supported operations as sequence diagram. The number in the sequence diagram represents dedicated action performed when a certain event happens.

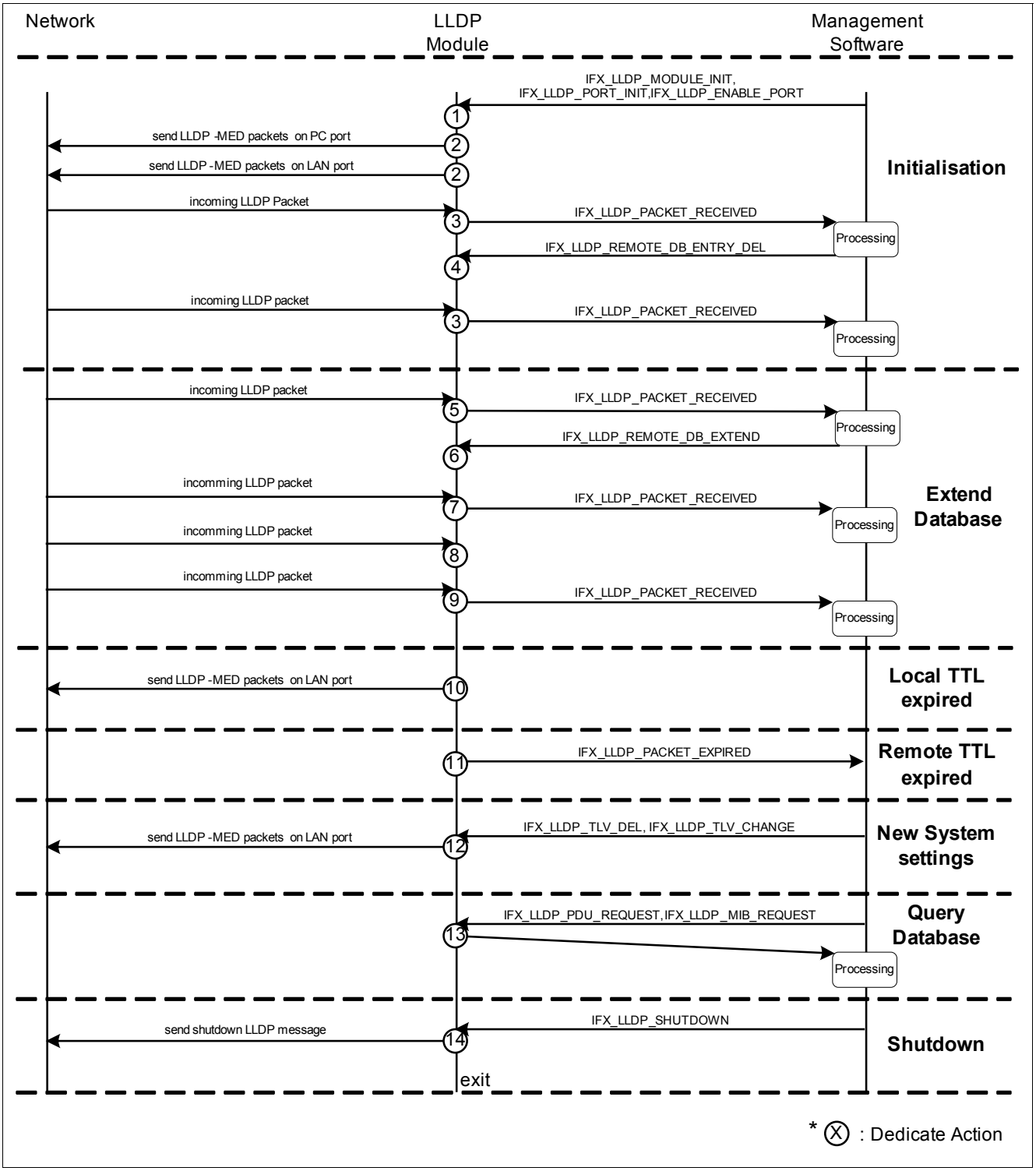

Figure 2 Sequence Diagram for a LLDP-MED Communication Device Endpoint (Class III) configuration

2.3.1 Initialization

The LLDP module will be started by a LLDP management software. For a proper operation the LLDP module needs following information:

- Operation Mode
- Supported TLVs

- Values required to build supported TLVs
- Number of supported entries for the remote system MIB
- Protocol specific informations

2.3.1.1 Operational Mode

The operational mode sets the behavior to be supported by the LLDP module. Here the operational mode have to be set to LLDP-MED Communication Device Endpoint (Class III).

2.3.1.2 Supported TLVs

Depending on the operational mode, different TLVs has to be supported. The LLDP module configured as LLDP-MED Communication Device Endpoint (Class III) supports following TLVs:

- Chassis ID
- Port ID
- Time-to-live
- System Capabilities
- MAC/PHY Configuration/Status
- LLDP-MED Capabilities
- Network Policy
- Extended Power via MDI
- End-of-LLDPDU

2.3.1.3 Values required to build supported TLVs

In order to build the supported TLV additional information are required.

Chassis ID TLV

In order to build the Chassis ID the IP address of the system is required. IP address will be transmit as Chassis ID as defined by the **TIA, TIA-1057 Link Layer Discovery Protocol for Media Endpoint Devices, April 2006**.

Port ID TLV

In order to build the Port ID TLV the MAC address for this port is required. The MAC address will be transmit as Port ID.

System Capabilities TLV

In order to build the System Capabilities TLV a bitmap withe the supported system capabilities is required as well as a bitmap that contain information about the enabled capabilities.

MAC/PHY Configuration/Status TLV

In order to build the MAC/PHY Configuration/Status TLV additional information are required as described in **IEEE, 802.1AB Station and Media Access Control Conectivity Discovery, May2005**, annex G.2.

LLDP-MED Capabilities TLV

In order to build the LLDP-MED Capabilities TLV a bitmap of supported LLDP-MED capabilities is required. The device type is equal to the operation mode.

Network Policy TLVs

In order to build the Network Policy TLV the LLDP module has to know which applications are supported by the system. **TIA, TIA-1057 Link Layer Discovery Protocol for Media Endpoint Devices, April 2006** defines the application types (Table 12). For each supported application a network Policy TLV is required.

Extended Power via MDI TLV

In order to build the Extended Power via MDI TLV the power type, power source, power value and power priority are required.

2.3.1.4 Number of supported Entries for the Remote System MIB

The number of supported entries for the remote system MIB will be used to initialize the remote system MIB.

2.3.1.5 Protocol specific informations

For proper processing the Transmit and Receive State Machine require some start values for internal variables. This information are required.

2.3.1.6 Dedicate Action during the Initialization Phase

Dedicate Action 1

Initialization of the LLDP module by calling the following functions **IFX_LLDP_MODULE_INIT**, **IFX_LLDP_PORT_INIT**, **IFX_LLDP_ENABLE_PORT**.

After the initialization the LLDP Entity will go to “Dedicated Action 2“.

Dedicate Action 2

After the internal modules have bin initialized the LLDP module will perform following tasks:

- Assembly a dedicated LLDPDU with own information for each port and send this out.

After sending the LLDPDUs the LLDP entity will wait till a new LLDP packet arrives.

Dedicate Action 3

When a new packet comes in, the LLDP module will inform the LLDP Management software about the new LLDPDU. The information message contains the device type of the sending device and also his MAC address.

Dedicate Action 4

If the device type of the received LLDPDU is MED, the LLDP Management software will probably decide to delete the received information. The LLDP module will receive a message for deleting the database entry. When receiving this message the LLDP module will delete the stored LLDPDU. The space will be now free for storing incoming LLDP Packages.

2.3.2 Extend Database

This Chapter describe the terms and conditions as well as the events that cause the LLDP Module to extend the remote system LLDP MIB

2.3.2.1 Dedicated Actions

Dedicate Action 5

When the remote system MIB is full and a new LLDP package comes in, the LLDP module will Inform the LLDP Management software that a new LLDPDU was received and discarded because the remote system LLDP MIB is full. The information message contains the device type of the sending device and also his MAC address.

Dedicate Action 6

The LLDP Management software could decide to extend the Remote System LLDP MIB and for this reason sends an appropriate message to the LLDP Entity. When receiving this message the LLDP Receive Manager will extend the database.

Dedicate Action 7

When a LLDP packet will be received the LLDP Module will inform the LLDP Management Software about the new LLDPDU. The information message contains the device type of the sending device and also his MAC address.

Dedicate Action 8

When a new LLDP packet form a already know MAC (already stored into the remote system LLDP MIB) arrives, the LLDP module will compare the stored LLDPDU with the received one. If LLDPDUs are identical the LLDP module will discard the received LLDPDU and restart the timer for this entry of the data base.

Dedicate Action 9

When a new LLDP packet form a already know MAC (already stored into the remote system LLDP MIB) arrives, the LLDP module will compare the stored LLDPDU with the received one. If the received LLDPDU is different from the already saved LLDPDU, the LLDP module will replace the stored LLDPDU with the received one and inform the LLDP manager about changes. The information message contains the device type of the sending device and also his MAC address.

2.3.3 Local TTL Expired

This Chapter describe the terms and conditions that cause the LLDP module to resent LLDP packets for activated ports.

2.3.3.1 Dedicated Actions

Dedicate Action 10

When the local TTL timer assigned to a port expires, the LLDP module will assembly a LLDPDU for this port and send it out.

2.3.4 Remote TTL Expired

This Chapter describe the terms and conditions that cause the LLDP module to delete a stored LLDPDU from the remote system LLDP MIB.

2.3.4.1 Dedicated Actions

Dedicate Action 11

When the remote TTL timer assigned to an entry of the remote system LLDP MIB expires, the LLDP module will remove the appropriate LLDPDU form the database and inform the LLDP manager about this action. The message send to the LLDP manager includes the MAC address of the deleted entry as well as his device type.

2.3.5 New System Settings

This Chapter describe the events that cause the LLDP module to resent LLDP packets for activated ports.

2.3.5.1 Dedicated Actions

Dedicate Action 12

When some system settings have changed, the LLDP manager informs the LLDP module about the values that have changed. Consequently the LLDP module will assembly a LLDPDU that consider the new system values and send this out to all enabled ports.

2.3.6 Query Database

This Chapter describe the events that cause the LLDP Module to query the remote system database.

2.3.6.1 Dedicated Actions

Dedicate Action 13

When the LLDP management software asks for a value, the LLDP module will extract the value from the stored LLDPDU and send this to the LLDP management software.

2.3.7 Shutdown

This Chapter describe the events that cause the LLDP Module to shutdown.

2.3.7.1 Dedicated Actions

Dedicate Action 14.

When the LLDP management software send an shutdown signal to the LLDP module, the LLDP module will assembly a LLDPDU with TTL =0 and send this out to all activated ports. Thereafter the LLDP Module will shut down.

3 Source File Organization

In following the source file organization is depicted:

```
lldp (root directory)
|
|-----------h
|             |
|             |-------------ifx_lldp_config.h
|-----------lldp_agent
|                     |
|                     |-------------h
|                     |             |
|                     |             |-------ifx_lldp_port.h
|                     |             |
|                     |             |-------ifx_lldp_rx_sm.h
|                     |             |
|                     |             |-------ifx_lldp_tx_sm.h
|                     |             |
|                     |             |-------ifx_lldp_local.h
|                     |             |
|                     |             |-------ifx_lldp_tlv.h
|                     |             |
|                     |             |-------ifx_lldp_main.h
|                     |             |
|                     |             |-------ifx_lldp_tx_module.h
|                     |             |
|                     |             |-------ifx_lldp_rx_module.h
|                     |             |
|                     |             |-------ifx_lldp_mib_if.h
|                     |             |
|                     |             |-------ifx_lldp_ms_if.h
|                     |             |
|                     |             |-------ifx_lldp_neighbor.h
|                     |             |
|                     |             |-------ifx_lldp_timer_if.h
|                     |
|                     |-------------src
|                     |             |
|                     |             |-------ifx_lldp_main.c
|                     |             |
|                     |             |-------ifx_lldp_local.c
|                     |             |
|                     |             |-------ifx_lldp_tx_sm.c
|                     |             |
|                     |             |-------ifx_lldp_rx_sm.c
|                     |             |
|                     |             |-------ifx_lldp_tlv.c
|                     |             |
|                     |             |-------ifx_lldp_neighbor.c
|                     |             |
|                     |             |------- ifx_lldp_tx_module.c
```

```
|                     |              |
|                     |              |------- ifx_lldp_rx_module.c
|                     |              |
|                     |              |------- ifx_lldp_ms_if.c
|                     |              |
|                     |              |-------ifx_lldp_mib_if.c
|                     |              |
|                     |              |-------ifx_lldp_timer_if.c
|                     |
|                     |--------------make
|                     |              |
|                     |              |-------makefile
|
|-----------lldp_cm
|                     |
|                     |-------------h
|                     |              |
|                     |              |-------ifx_lldp_cm_defs.h
|                     |              |
|                     |              |-------ifx_lldp_cm.h
|                     |              |
|                     |-------------src
|                     |              |
|                     |              |-------ifx_lldp_cm.c
|                     |              |
|                     |--------------make
|                     |              |
|                     |              |-------makefile
|
|-----------lldp_intf
|                     |
|                     |-------------h
|                     |              |
|                     |              |-------ifx_lldp_intf_defs.h
|                     |              |
|                     |              |-------ifx_lldp_intf.h
|                     |              |
|                     |-------------src
|                     |              |
|                     |              |-------ifx_lldp_intf.c
|                     |              |
|                     |--------------make
|                     |              |
|                     |              |-------makefile
|
|-----------lldp_manager
|                     |
|                     |-------------h
|                     |              |
|                     |              |-------ifx_lldp_manager.h
|                     |
```

```
|                     |-------------src
|                     |             |
|                     |             |-------ifx_lldp_manager.c
|                     |
|                     |--------------make
|                     |             |
|                     |             |-------makefile
|
|-----------common
|                     |
|                     |-------------h
|                     |             |
|                     |             |-------ifx_common_defs.h
|                     |             |
|                     |             |-------ifx_os.h
|                     |             |
|                     |             |-------IFX_TLIB_Timlib.h
|                     |             |
|                     |-------------src
|                     |             |
|                     |             |-------ifx_os.c
|                     |             |
|                     |             |-------IFX_TLIB_Timlib.c
|                     |
|                     |--------------make
|                     |             |
|                     |             |-------common-make.inc
|                     |             |
|                     |             |-------makefile
|
|-----------make
|                     |
|                     |-------------lldp_makefile.inc
|
|-------build
|
|-------build-clean
|
|-----------bin
|
|-----------lib
```

4 Module Interface Description

4.1 Functional Reference

4.1.1 IFX_LLDP_MODULE_INIT

Description

This API is used to Initialize the LLDP module.

Prototype

```
void IFX_LLDP_MODULE_INIT (x_IFX_LLDP_MODULE_INIT *pxInit);
```

Parameters

Data Type	Name	Description
x_IFX_LLDP_MODULE_INIT *	pxInit	Structure containing the values required to initialise the LLDP Module.

Return Values

Data Type	Description
void	No return values

4.1.2 IFX_LLDP_PORT_INIT

Description

Initializes a specific LLDP port. Current a LAN port and a PC port are supported.

Prototype

```
char8 IFX_LLDP_PORT_INIT (   uchar8 *pIntfName,
                             e_IFX_LLDP_PORT_NAME ePort,
                             x_IFX_LLDP_CONFIGCFG* pxCfgData,
                             x_IFX_LLDP_TLV_SUPPORT xTlvSupport,
                             x_IFX_LLDP_INIT_TLV_VALUES* pxTlvValues);
```

Parameters

Data Type	Name	Description
uchar8 *	pIntfName	This parameter specifies the Interface name to which this port should be associated.
e_IFX_LLDP_PORT_NAME	ePort	An enumerator which specifies the port name i.e. PC or LAN.
x_IFX_LLDP_CONFIGCFG *	pxCfgData	Specifies default values for all local configuration parameters.
x_IFX_LLDP_TLV_SUPPORT	xTlvSupport	Specifies the TLVs to be supported on this port.
x_IFX_LLDP_TLV_VALUES *	pxTlvValues	Pointer to a structure containing default values of supported TLVs.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.3 IFX_LLDP_ENABLE_PORT

Description

This function enables an initialized port for transmission/reception of LLDP packets.

Prototype

```
char8 IFX_LLDP_ENABLE_PORT (e_IFX_LLDP_PORT_NAME ePort);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port Id to be enabled (PC or LAN).

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.4 IFX_LLDP_DISABLE_PORT

Description

This function disables an enabled port.

Prototype

```
char8 IFX_LLDP_DISABLE_PORT (e_IFX_LLDP_PORT_NAME ePort);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port Id to be disabled (PC or LAN).

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.5 IFX_LLDP_TLV_ADD

Description

This interface API adds a new TLV. Note that if the TLV which is being added is already present, this interface returns error.

Prototype

```
char8 IFX_LLDP_TLV_ADD (     e_IFX_LLDP_PORT_NAME ePort,
                             x_IFX_LLDP_TLV_VALUES *pxTlvValues);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port to which the new TLV should be added.
x_IFX_LLDP_TLV_VALUES*	pxTlvValues	New TLV information.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.6 IFX_LLDP_TLV_DEL

Description

This interface API deletes an existing TLV. Note that if the TLV which is being deleted is not present, this interface returns error. It is also not allowed to delete mandatory TLVs. The interface returns error by dooing so.

Prototype

```
char8 IFX_LLDP_TLV_DEL (     e_IFX_LLDP_PORT_NAME ePort,
                             e_IFX_LLDP_INTF_TLV eTlvType);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port from which the new TLV should be deleted.
e_IFX_LLDP_INTF_TLV	eTlvType	TLV to be deleted.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.7 IFX_LLDP_TLV_CHANGE

Description

This interface API modifies an existing TLV. Note that if the TLV which is being modified is not present, this interface returns error.

Prototype

```
char8 IFX_LLDP_TLV_CHANGE ( e_IFX_LLDP_PORT_NAME ePort,
                            x_IFX_LLDP_TLV_VALUES *pxTlvValues);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port at which the new TLV should be modified.
x_IFX_LLDP_TLV_VALUES*	pxTlvValues	New TLV information.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.8 IFX_LLDP_REMOTE_DB_EXTEND

Description

This API is used to extend the remote MIB when MIB full indication is received from LLDP Agent.

Prototype

```
char8 IFX_LLDP_REMOTE_DB_EXTEND (uint32 uiNumber);
```

Parameters

Data Type	Name	Description
uint32	uiNumber	Number of entries by them the remote system database will be extended.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.9 IFX_LLDP_REMOTE_DB_ENTRY_DEL

Description

This interface API Deletes an entry from remote MIB.

Prototype

```
char8 IFX_LLDP_REMOTE_DB_ENTRY_DEL (x_IFX_LLDP_MAC xMacAddres);
```

Parameters

Data Type	Name	Description
x_IFX_LLDP_MAC	xMacAddres	Address of the entry to be deleted.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.10 IFX_LLDP_PDU_REQUEST

Description

This interface API Fetches the raw PDU data from the remote MIB. Note that the function will return error if the bLldpduStorage parameter was set to FALSE when the function **IFX_LLDP_MODULE_INIT** was called.

Prototype

```
char8 IFX_LLDP_PDU_REQUEST (x_IFX_LLDP_MAC
xMacAddress,                              void *pPtr);
```

Parameters

Data Type	Name	Description
x_IFX_LLDP_MAC	xMacAddress	MAC address of the entry.
void *	pPtr	The raw packet is returned in this parameter.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.11 IFX_LLDP_MIB_REQUEST

Description

This interface API Fetches the structured remote LLDP data from the remote MIB.

Prototype

```
char8 IFX_LLDP_MIB_REQUEST (x_IFX_LLDP_MAC xMacAddress,
                            x_IFX_LLDP_TLV_VALUES *pxTlvValues);
```

Parameters

Data Type	Name	Description
x_IFX_LLDP_MAC	xMacAddress	MAC address of the entry.
x_IFX_LLDP_TLV_VALUES*	pxTlvValues	The TLV data is returned in this parameter.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.12 IFX_LLDP_LOCAL_CONFIG_UPDATED

Description

This interface API Updates local configuration parameters.

Prototype

```
char8 IFX_LLDP_LOCAL_CONFIG_UPDATED(e_IFX_LLDP_PORT_NAME
ePort,                              x_IFX_LLDP_CONFIGCFG* pxCfgData);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port for which the local configuration data is being updated.
x_IFX_LLDP_CONFIGCFG*	pxCfgData	New local configuration data.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.13 IFX_LLDP_COUNTERS_REQUEST

Description

This interface API Fetches the statistical data from LLDP Agent.

Prototype

```
char8 IFX_LLDP_COUNTERS_REQUEST (e_IFX_LLDP_PORT_NAME
ePort,                           x_IFX_LLDP_PORT_STATS* pxStats);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_PORT_NAME	ePort	Port for which the statistical data are requested.
x_IFX_LLDP_PORT_STATS*	pxStats	Port status are returned in this output parameter.

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.14 IFX_LLDP_SHUTDOWN

Description

This interface API is used to completely shut down the LLDP agent.

Prototype

```
char8 IFX_LLDP_SHUTDOWN();
```

Parameters

Data Type	Name	Description

Return Values

Data Type	Description
char8	0_D On successful enabling. -1_D On enabling failed.

4.1.15 IFX_LLDP_PACKET_RECEIVED

Description

This function is used to notify the higher layer application whenever a new LLDP packet is received.

Prototype

```
void IFX_LLDP_PACKET_RECEIVED(x_IFX_LLDP_PACKET *pxLldpPacket);
```

Parameters

Data Type	Name	Description
x_IFX_LLDP_LLDPAGENT_MSG*	pxLldpPacket	A pointer to a structure containing the values required to send response to the application.

Return Values

Data Type	Description
void	No return values

4.1.16 IFX_LLDP_PACKET_EXPIRED

Description

This function is used to notify the higher layer application whenever a existing LLDP packet stored in remote MIB is expired.

Prototype

```
void IFX_LLDP_PACKET_EXPIRED(e_IFX_LLDP_DEVICE_TYPE eDeviceType,
                             x_IFX_LLDP_MAC *pxMacAddress);
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_DEVICE_TYPE	eDeviceType	Specifies the type of device expired.
x_IFX_LLDP_MAC	pxMacAddress	MAC address of the remote MIB expired.

Return Values

Data Type	Description
void	No return values

4.1.17 LldpMgr_intf_funcptr

Description

Callback function to be registered by the LLDP Manager during module intiialization in order to receive events from the LLDP Agent.

Prototype

```
typedef void (*lldpMgr_intf_funcptr)(x_IFX_LLDP_LLDPAGENT_MSG*);
```

Parameters

Data Type	Name	Description
x_IFX_LLDP_LLDPAGENT_MSG *		Pointer to the message structure.

Return Values

Data Type	Description
void	No return values

4.2 Type Definition Reference

4.2.1 Structure Reference

4.2.1.1 x_IFX_LLDP_CONFIGCFG

Description

This structure defines the local LLDP configuration parameters as described in Table - 11.1 in 802.1 AB specification [1].

Prototype

```
typedef struct {
   uint8 ucIntfAdminStatus;
   uint16 unIntfReinitDelay;
   uint16 unIntfMsgTxHold;
   uint16 unIntfMsgTxInterval;
   uint16 unIntfTxDelay;
   uint32 uiIntfMedFastStartRepeatCount;
} x_IFX_LLDP_CONFIGCFG;
```

Parameters

Data Type	Name	Description
uint8	ucIntfAdminStatus	For more details refere to 802.1 AB:10.5.1
uint16	unIntfReinitDelay	For more details refere to 802.1 AB:10.5.3.3
uint16	unIntfMsgTxHold	For more details refere to 802.1 AB:10.5.3.3
uint16	unIntfMsgTxInterval	For more details refere to 802.1 AB:10.5.3.3
uint16	unIntfTxDelay	For more details refere to 802.1 AB:10.5.3.3
uint32	uiIntfMedFastStartRepeatCount	For more details refere to TIA 1057: 12.1

4.2.1.2 x_IFX_LLDP_EXTENDED_POWER

Description

This structure defines Extended Power-Via-MDI TLV as defined by TIA 1057, section 10.2.5 [2].

Prototype

```
typedef struct
{
    uint8 uiType;
    uint8 uiSource;
    uint8 uiPriority;
    uint16 uiValue;
}x_IFX_LLDP_EXTENDED_POWER;
```

Parameters

Data Type	Name	Description
uint8	uiType	Value required to build the extended power via MDI TLV: 1_D PSE Device 2_D PD Device.
uint8	uiSource	Value required to build the extended power via MDI TLV: 1_D unknown 2_D PSE & Primarypower source 3_D Local & Backup power Source, Power conservation mode 4_D PSE and Local
uint8	uiPriority	Value required to build the extended power via MDI TLV: 1_D unknown 2_D critical 3_D high 4_Dlow
uint16	uiValue	Integer value that indicate the total power in watt (range 0 to 1023).

4.2.1.3 x_IFX_LLDP_INIT_TLV_VALUES

Description

Structure containing the supported TLVs.

Prototype

```
typedef struct
{
        uchar8 ucChassisId[255];
        uchar8 ucPortId[255];
        x_IFX_LLDP_SYS_CAPABILITIES xSystemCapabilities;
        x_IFX_LLDP_PORT_VLAN_ID xPortVlanId;
        x_IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID xPortAndProtocolVlanId;
        x_IFX_LLDP_VLAN_NAME xVlanName;
        x_IFX_LLDP_PROTOCOL_IDENTITY xProtocolIdentity;
```

```
        x_IFX_LLDP_MAC_PHY_CONFIG_STATUS xMacPhyConfigurationStatus;
        x_IFX_LLDP_MED_CAPABILITIES xLldpMedCapabilities;
        x_IFX_LLDP_NETWORK_POLICY xNetworkPolicy;
        x_IFX_LLDP_EXTENDED_POWER xExtendedPowerViaMdi;
}x_IFX_LLDP_INIT_TLV_VALUES;
```

Parameters

Data Type	Name	Description
uchar8	ucChassisId[255]	Character array that contains the ChassisId.
uchar8	ucPortId[255]	Character array that contains the PortId.
x_IFX_LLDP_SYS_CAPABILITIES	xSystemCapabilities	Character array that contains the PortId.
x_IFX_LLDP_PORT_VLAN_ID	xPortVlanId	VLAN ID for the port as defined in 8.4.4 of IEEE 802.1Q-1998.
x_IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID*	xPortAndProtocolVlanId	Values required to build the Port and protocol VLAN ID TLV.
x_IFX_LLDP_VLAN_NAME	xVlanName	Values required to build the VLAN Name TLV.
x_IFX_LLDP_PROTOCOL_IDENTITY	xProtocolIdentity	Values required to build the Protocol Identity TLV.
x_IFX_LLDP_MAC_PHY_CONFIG_STATUS	xMacPhyConfigurationStatus	Values required to build the MAC/PHY Configuration/Status TLV.
x_IFX_LLDP_MED_CAPABILITIES	xLldpMedCapabilities	Values required to build the LLDP-MED Capababilities TLV.
x_IFX_LLDP_NETWORK_POLICY_TLV	xNetworkPolicy	Values required to build the Network policy TLV.
x_IFX_LLDP_EXTENDED_POWER	xExtendedPowerViaMdi	Values required to build the Exrended Power via MDI TLV.

4.2.1.4 x_IFX_LLDP_MAC

Description

Structure for storing MAC address.

Prototype

```
typedef struct {
   uint32 uiaMacAddress[6];
} x_IFX_LLDP_MAC;
```

Parameters

Data Type	Name	Description
uint32	uiaMacAddress[6]	MAC address.

4.2.1.5 x_IFX_LLDP_MAC_PHY_CONFIG_STATUS

Description

This structure defines the MAC/PHY TLVas defined by 802.1 AB, section G.2 [1].

Prototype

```
typedef struct
{
  uint8 uiSupport;
  uint8 uiStatus;
  uint16 uiAdvertisedCapabilities;
  uint16 uiMauType;
}x_IFX_LLDP_MAC_PHY_CONFIG_STATUS;
```

Parameters

Data Type	Name	Description
uint8	uiSupport	This value is required in order to build the auto-negotiation support/status field of the MAC/PHY Configuration/Status TLV: 0_D Auto-negotiation is not supported 1_D Auto-negotiation is supported
uint8	uiStatus	This value is required in order to build the auto-negotiation support/status field of the MAC/PHY Configuration/Status TLV: 0_D Auto-negotiation is disabled 1_D Auto-negotiation is enabled
uint16	uiAdvertisedCapabilities	Integer value as defined by RFC 3636
uint16	uiMauType	Integer value. A list of values is listed in RFC3636

4.2.1.6 x_IFX_LLDP_MED_CAPABILITIES

Description

Structure containing values required to build the LLDP-MED capabilities TLV.

Prototype

```
typedef struct
{
   boolean bCapabilities;
   boolean bNetworkPolicy;
   boolean bLocationIdentification;
   boolean bExtendedPowerMdiPD;
   boolean bExtendedPowerMdiPse;
   boolean bInventory;
```

```
   uint8 uiDeviceType;
}x_IFX_LLDP_MED_CAPABILITIES;
```

Parameters

Data Type	Name	Description
boolean	bCapabilities	TRUE: capability exist FALSE:capability is not present
boolean	bNetworkPolicy	TRUE: capability exist FALSE:capability is not present
boolean	bLocationIdentification	TRUE: capability exist FALSE:capability is not present
boolean	bExtendedPowerMdiPD	TRUE: capability exist FALSE:capability is not present
boolean	bExtendedPowerMdiPse	TRUE: capability exist FALSE:capability is not present
boolean	bInventory	TRUE: capability exist FALSE:capability is not present
uint8	uiDeviceType	Value required to build the LLDP-MED Device Type field: 4_D Network Connectivity 3_D Endpoint Class III

4.2.1.7 x_IFX_LLDP_MODULE_INIT

Description

This interface structure is used to pass initialization parameters to initialize the LLDP Agent.

Prototype

```
typedef struct
{
       uint32 uiOpMode;
       uchar8 ucNumPorts;
       uint32 uiSupportedMibEntries;
       boolean bLldpduStorage;
       lldpMgr_intf_funcptr pFuncPtr;
}x_IFX_LLDP_MODULE_INIT;
```

Parameters

Data Type	Name	Description
uint32	uiOpMode	Set the operation mode of the LLDP module: 0_D **Network Connectivity Device** 1_D **Device Endpoint Class III**.
uchar8	ucNumPorts	Number of ports to be supported.
uint32	uiSupportedMibEntries	Give the number of entries to be supported by the LLDP remote system MIB.

Data Type	Name	Description
boolean	bLldpduStorage	Indicates whether the raw remote LLDP packet data needs to be stored or not.
LldpMgr_intf_funcptr	pFuncPtr	Callback function pointer.

4.2.1.8 x_IFX_LLDP_NETWORK_POLICY_TLV

Description

This structure defines the Network policy TLVas defined by TIA 1057, section 10.2.3 [2].

Prototype

```
typedef struct
{   uint8 uiApplicationType;
   boolean bUnknownFlag;
   boolean bTaggedFlag;
   uint16 uiVlanId;
   uint8 uiL2Priority;
   uint8 uiDscpValue;
}x_IFX_LLDP_NETWORK_POLICY_TLV;
```

Parameters

Data Type	Name	Description
uint8	uiApplicationType	Primary application type defined for this network policy: 1_D Voice 2_D Voice Signaling 3_D Guest Voice 4_D Guest Voice Signaling 5_D Softphone Voice 6_D Video Conferencing 7_D Streaming Video 8_DVideo Signaling
boolean	bUnknownFlag	TRUE: policy is requested bat currently unknown FALSE:policy known
boolean	bTaggedFlag	TRUE: capability exist FALSE:capability is not present
uint16	uiVlanId	VLAN identifier as defined by IEEE 802.1Q-2003
uint8	uiL2Priority	Specifies the priority level as defined by IEEE 802.1D-2004
uint8	uiDscpValue	Specifies the DSCP value as defined by RFC 2474

4.2.1.9 x_IFX_LLDP_NETWORK_POLICY

Description

This structure contains all Network Policy TLVs values.

Prototype

```
typedef struct
```

```
{
   uint8 uiNoOfNetworkPolicyTlvs;
   x_IFX_LLDP_NETWORK_POLICY_TLV xNetworkPolicyArray[IFX_LLDP_MAXNETWORKTLVS];
}x_IFX_LLDP_NETWORK_POLICY;
```

Parameters

Data Type	Name	Description
uint8	uiNoOfNetworkPolicyTlvs	Number of supported networc policy TLVs
x_IFX_LLDP_NETWORK_POLICY_TLV	xNetworkPolicyArray[IFX_LLDP_MAXNETWORKTLVS]	Network Policy TLVs values

4.2.1.10 x_IFX_LLDP_PACKET

Description

This structure contains information related to a received LLDP packet. This is sent out by LLDP Agent whenever a new remote LLDP packet is received.

Prototype

```
typedef struct
{
   e_IFX_LLDP_DEVICE_TYPE eDeviceType;
   x_IFX_LLDP_MAC xMacAddress;
   e_IFX_LLDP_PACKET_STATUS ePktStat;
   e_IFX_LLDP_MIB_STATUS eMibStatus;
}x_IFX_LLDP_PACKET;
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_DEVICE_TYPE	eDeviceType	Device Type (PC, LAN)
x_IFX_LLDP_MAC	xMacAddress	Source MAC address
e_IFX_LLDP_PACKET_STATUS	ePktStat	Status of the packet (new, changed, unchanged)
e_IFX_LLDP_MIB_STATUS	eMibStatus	MIB Status (Free, Full)

4.2.1.11 x_IFX_LLDP_LLDPAGENT_MSG

Description

This structure defines the FIFO message posted by LLDP Agent to LLDP Manager.

Prototype

```
typedef struct
{
   e_IFX_LLDP_MsgType eMsgType;
   union {
      x_IFX_LLDP_PACKET xPacket;
      x_IFX_LLDP_PACKET_EXPIRED xPacketExpInfo;
   }ux_Data;
} x_IFX_LLDP_LLDPAGENT_MSG;
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_MsgType	eMsgType	Type of message
union	ux_Data	Union of possible data sent

4.2.1.12 x_IFX_LLDP_PACKET_EXPIRED

Description

This structure contains information of an expired packet. This is sent out by LLDP Agent whenever a new remote LLDP packet expires.

Prototype

```
typedef struct {
   e_IFX_LLDP_DEVICE_TYPE eDeviceType;
   x_IFX_LLDP_MAC xMacAddress;
} x_IFX_LLDP_PACKET_EXPIRED;
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_DEVICE_TYPE	eDeviceType	Device Type - PC, LAN
x_IFX_LLDP_MAC	xMacAddress	Source MAC address

4.2.1.13 x_IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID

Description

This structure stores the Port And Protocol VLAN ID TLV as defined by TIA 1057, section 5.3 [2] and IEEE 802.1 AB, section E.2.2 [1].

Prototype

```
typedef struct {
   uint8 uiProtoVlanSupported;
   uint8 uiProtoVlanEnabled;
```

```
   uint16 uiProtoVlanId;
}x_IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID;
```

Parameters

Data Type	Name	Description
uint8	uiProtoVlanSupported	Required to build the flags field of the port and protocol VLAN ID TLV: 0_D Port and protocol VLAN not supported 1_D Port and protocol VLAN supported
uint8	uiProtoVlanEnabled	Required to build the flags field of the port and protocol VLAN ID TLV: 0_D Port and protocol VLAN disabled 1_D Port and protocol VLAN enabled
uint16	uiProtoVlanId	Port and protocol VLAN ID value to be transmit within the Port and protocol VLAN ID TLV

4.2.1.14 x_IFX_LLDP_PORT_STATS

Description

This structure defines all statistical information related to the functioning of LLDP Agent as defined by IEEE 802.1 AB, section 10.5.2 [1].

Prototype

```
typedef struct {
   uint32 ulPortStatsAgeoutsTotal;
   uint32 ulPortStatsFramesDiscardedTotal;
   uint32 ulPortStatsFramesInErrorsTotal;
   uint32 ulPortStatsFramesInTotal;
   uint32 ulPortStatsFramesOutTotal;
   uint32 ulPortStatsTLVsDiscardedTotal;
   uint32 ulPortStatsTLVsUnrecognizedTotal;
}x_IFX_LLDP_PORT_STATS;
```

Parameters

Data Type	Name	Description
uint32	ulPortStatsAgeoutsTotal	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail
uint32	ulPortStatsFramesDiscarded Total	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail
uint32	ulPortStatsFramesInErrorsTo tal	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail
uint32	ulPortStatsFramesInTotal	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail
uint32	ulPortStatsFramesOutTotal	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail

Data Type	Name	Description
uint32	ulPortStatsTLVsDiscardedTotal	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail
uint32	ulPortStatsTLVsUnrecognizedTotal	Refer to IEEE 802.1 AB, section 10.5.2 [1] for detail

4.2.1.15 x_IFX_LLDP_PORT_VLAN_ID

Description

This structure stores the Port VLAN ID TLV related information as defined by TIA 1057, section 5.3 [2] and IEEE 802.1 AB, section E.2.1 [1].

Prototype

```
typedef struct {
   uint16 uiPortVlanId;
}x_IFX_LLDP_PORT_VLAN_ID;
```

Parameters

Data Type	Name	Description
uint16	uiPortVlanId	Port VLAN ID value

4.2.1.16 x_IFX_LLDP_PROTOCOL_IDENTITY

Description

This structure stores the Port VLAN ID TLV related information as defined by TIA 1057, section 5.3 [2] and IEEE 802.1 AB, section E.2.3 [1]..

Prototype

```
typedef struct {
   uint8 uiProtoIdLength
   uint8 uiProtoId[255];
}x_IFX_LLDP_PROTOCOL_IDENTITY;
```

Parameters

Data Type	Name	Description
uint8	uiProtoIdLength	Length in octets of the protocolIdentifier
uint8	uiProtoId[255]	Protocol Identity value as defined in IEEE, 802.1AB Station and Media Access Control Conectivity Discovery, May2005 anex F (F.5.3) [2]

4.2.1.17 x_IFX_LLDP_SYS_CAPABILITIES

Description

Structure containing values required to build the System Capabilities TLV.

Prototype

```
typedef struct
{
        boolean bOther;boolean bOther_status;
        boolean bRepeater;boolean bRepeater_status;
        boolean bBridge;
        boolean bBridge_status;
        boolean bWlanAp;
        boolean bWlanAp_status;
        boolean bRouter;
        boolean bRouter_status;
        boolean bTelephone;
        boolean bTelephone_status;
        boolean bCableDevice;
        boolean bCableDevice_status;
        boolean bStationOnly;
        boolean bStationOnly_status;
}x_IFX_LLDP_SYS_CAPABILITIES;
```

Parameters

Data Type	Name	Description
boolean	bOther	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bOther_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.
boolean	bRepeater	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bRepeater_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.
boolean	bBridge	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bBridge_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.
boolean	bWlanAp	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bWlanAp_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.
boolean	bRouter	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bRouter_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.
boolean	bTelephone	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bTelephone_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.

Data Type	Name	Description
boolean	bCableDevice	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bCableDevice_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.
boolean	bStationOnly	IFX_TRUE: capability supported. IFX_FALSE:capability not supported.
boolean	bStationOnly_status	IFX_TRUE: capability enabled. IFX_FALSE:capability disabled.

4.2.1.18 x_IFX_LLDP_TLV_SUPPORT

Description

This interface structure is used to pass supported TLVs information to the LLDP Agent.

Prototype

```
typedef struct
{
        boolean bChassisId;
        boolean bPortId;
        boolean bSystemCapabilities;
        boolean bTimeToLive;
        boolean bPortVlanId;
        boolean bPortAndProtocolVlanId;
        boolean bVlanName;
        boolean bProtocolIdentity;
        boolean bMacPhyConfigurationStatus;
        boolean bLldpMedCapabilities;
        boolean bNetworkPolicy;
        boolean bExtendedPowerViaMdi;
        boolean endOfLldbEndOfLldpdu;
} x_IFX_LLDP_TLV_SUPPORT;
```

Parameters

Data Type	Name	Description
boolean	bChassisId	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bPortId	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bSystemCapabilities	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bTimeToLive	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bPortVlanId	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.

Data Type	Name	Description
boolean	bPortAndProtocolVlanId	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bVlanName	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bProtocolIdentity	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bMacPhyConfigurationStatus	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bLldpMedCapabilities	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bNetworkPolicy	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	bExtendedPowerViaMdi	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.
boolean	endOfLldbEndOfLldpdu	IFX_TRUE: TLV supported. IFX_FALSE:TLV not supported.

4.2.1.19 x_IFX_LLDP_TLV_VALUES

Description

This structure is used to fetch a specific local TLV information from LLDP Agent.

Prototype

```
typedef struct
{
   e_IFX_LLDP_INTF_TLV uiNewTlv;
   x_IFX_LLDP_TLV_CHOICE xValues;
}x_IFX_LLDP_TLV_VALUES;
```

Parameters

Data Type	Name	Description
e_IFX_LLDP_INTF_TLV	uiNewTlv	TLV to be fetched
x_IFX_LLDP_TLV_CHOICE	xValues	TLV data is returned in this member variable

4.2.1.20 x_IFX_LLDP_VLAN_NAME

Description

This structure stores the VLAN Name TLV related information as defined by TIA 1057, section 5.3 [2] and IEEE 802.1 AB, section E.2.3 [1].

Prototype

```
typedef struct {
```

```
   uint16 uiVlanId;
   uint8 uiVlanNameLength;
   uint8 uiaVlanName[32];
}x_IFX_LLDP_VLAN_NAME;
```

Parameters

Data Type	Name	Description
uint16	uiVlanId	VLAN ID associated with the VLAN name
uint8	uiVlanNameLength	Length in octets of the VLAN name
uint8	uiaVlanName[32]	VLAN name to be transmited within the VLAN Name TLV

4.2.2 Enumerator Reference

This chapter contains the basic type definitions, reference of data types and structures of all modules.

4.2.2.1 e_IFX_LLDP_INTF_TLV

Description

This enum encompasses all possible TLV types defined in 802.1 AB and TIA 1057.

Prototype

```
typedef enum
{
   IFX_LLDP_CHASSIS_ID = 0x01,
   IFX_LLDP_PORT_ID = 0x02,
   IFX_LLDP_SYSTEM_CAPABILITIES = 0x04,
   IFX_LLDP_TIME_TO_LIVE = 0x08,
   IFX_LLDP_PORT_VLAN_ID = 0x10,
   IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID = 0x20,
   IFX_LLDP_VLAN_NAME = 0x40,
   IFX_LLDP_PROTOCOL_IDENTITY =0x80,
   IFX_LLDP_MAC_PHY_CONFIGURATION_STATUS = 0x100,
   IFX_LLDP_MED_CAPABILITIES = 0x200,
   IFX_LLDP_NETWORK_POLICY = 0x400,
   IFX_LLDP_EXTENDED_POWER_VIA_MDI = 0x800,
   IFX_LLDP_END_OF_LLDPDU =0x1000
}e_IFX_LLDP_INTF_TLV;
```

4.2.2.2 e_IFX_LLDP_DEVICE_TYPE

Description

Enumeration used for device type types.

Prototype

```
typedef enum
{
   IFX_LLDP_NCD = 0,
   IFX_LLDP_EPD = 1
```

```
}e_IFX_LLDP_DEVICE_TYPE;
```

4.2.2.3 e_IFX_LLDP_DEVICETYPE

Description

This enum defines all types of devices such as Network Connectivity Device, Endpoint Device etc.

Prototype

```
typedef enum {
   IFX_LLDP_NOTDEFINED = 0,
   IFX_LLDP_CLASSI,
   IFX_LLDP_CLASSII,
   IFX_LLDP_CLASSIII,
   IFX_LLDP_NETWORKCONNECTIVITY
}e_IFX_LLDP_DeviceType;
```

4.2.2.4 e_IFX_LLDP_INTF_AdminStatus

Description

This enum defines all types of possible admin status values as defined by IEEE 802.1 AB, section 10.5.1 [1].

Prototype

```
typedef enum {
   IFX_LLDP_INTF_DISABLED = 0,
   IFX_LLDP_INTF_ENABLED_TX_ONLY ,
   IFX_LLDP_INTF_ENABLED_RX_ONLY,
   IFX_LLDP_INTF_ENABLED_RXTX
}e_IFX_LLDP_INTF_AdminStatus;
```

4.2.2.5 e_IFX_LLDP_MIB_STATUS

Description

This enum defines all possible MIB status information.

Prototype

```
typedef enum {
   IFX_LLDP_MIB_FREE =0,
   IFX_LLDP_MIB_FULL
}e_IFX_LLDP_MIB_STATUS;
```

4.2.2.6 e_IFX_LLDP_MsgType

Description

This enum lists all message types which are posted by LLDP Agent to LLDP Manager.

Prototype

```
typedef enum {
   IFX_LLDP_PACKET_RXVD_MSG = 0,
```

```
   IFX_LLDP_PACKET_EXPIRED_MSG
} e_IFX_LLDP_MsgType;
```

4.2.2.7 e_IFX_LLDP_PACKET_STATUS

Description

This enumerator specifies status of the packet received.

Prototype

```
typedef enum {
   IFX_LLDP_PACKET_NEW =0,
   IFX_LLDP_PACKET_EXISTING_CHANGED,
   IFX_LLDP_PACKET_EXISTING_UNCHANGED
}e_IFX_LLDP_PACKET_STATUS;
```

4.2.2.8 e_IFX_LLDP_PORT_NAME

Description

Enumeration used for port types.

Prototype

```
typedef enum
{
        IFX_LLDP_PC = 0,
        IFX_LLDP_LAN = 1
} e_IFX_LLDP_PORT_NAME;
```

4.2.3 Union Reference

4.2.3.1 x_IFX_LLDP_TLV_CHOICE

Description

This union is used to fetch local TLV information from the LLDP Agent one at a time.

Prototype

```
typedef union
{
   uchar8 ucChassisId[255];
   uchar8 ucPortId[255];
   x_IFX_LLDP_SYS_CAPABILITIES xSystemCapabilities;
   x_IFX_LLDP_PORT_VLAN_ID xPortVlanId;
   x_IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID xPortAndProtocolVlanId;
   x_IFX_LLDP_VLAN_NAME xVlanName;
   x_IFX_LLDP_PROTOCOL_IDENTITY xProtocolIdentity;
   x_IFX_LLDP_MAC_PHY_CONFIG_STATUS xMacPhyConfigurationStatus;
   x_IFX_LLDP_MED_CAPABILITIES xLldpMedCapabilities;
   x_IFX_LLDP_NETWORK_POLICY xNetworkPolicy;
```

```
   x_IFX_LLDP_EXTENDED_POWER xExtendedPowerViaMdi;
}x_IFX_LLDP_TLV_CHOICE;
```

Parameters

Data Type	Name	Description
uchar8	ucChassisId[255]	
uchar8	ucPortId[255]	
x_IFX_LLDP_SYS_CAPABILITIES	xSystemCapabilities	
x_IFX_LLDP_PORT_VLAN_ID	xPortVlanId	
x_IFX_LLDP_PORT_AND_PROTOCOL_VLAN_ID	xPortAndProtocolVlanId	
x_IFX_LLDP_VLAN_NAME	xVlanName	
x_IFX_LLDP_PROTOCOL_IDENTITY	xProtocolIdentity	
x_IFX_LLDP_MAC_PHY_CONFIG_STATUS	xMacPhyConfigurationStatus	
x_IFX_LLDP_MED_CAPABILITIES	xLldpMedCapabilities	
x_IFX_LLDP_NETWORK_POLICY	xNetworkPolicy	
x_IFX_LLDP_EXTENDED_POWER	xExtendedPowerViaMdi	

References

[1] IEEE, 802.1AB Station and Media Access Control Conectivity Discovery, May2005

[2] TIA, TIA-1057 Link Layer Discovery Protocol for Media Endpoint Devices, April 2006

Appendix

This chapter keeps track of all main changes occurred in the different document's releases.

www.infineon.com

Published by Infineon Technologies AG