Embedded USB Host LAN7500 and LAN9500 Class Driver User Guide

Version 1.10

For use with USBH Microchip LAN7500 and LAN9500 Class Driver versions 1.02 and above

Date: 20-Apr-2016 18:27

Table of Contents

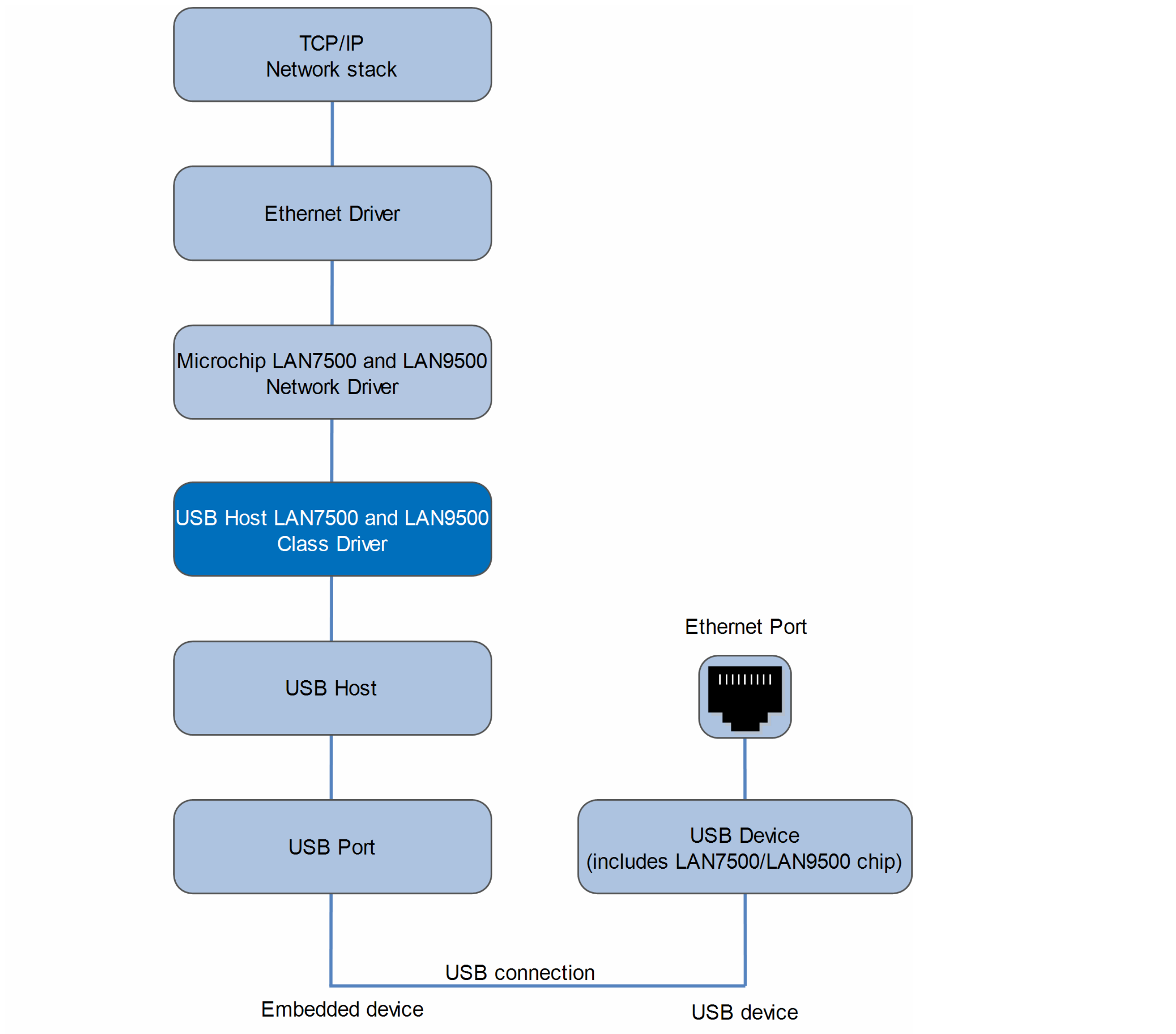
1 System Overview

1.1 Introduction

This guide is for those who want to implement an embedded host class driver for LAN7500 High Speed USB 2.0 to 10/100/1000 Ethernet Controllers and LAN9500 USB 2.0 to 10/100 Ethernet Controllers. These devices are produced by Microchip Technology Inc. (formerly SMSC).

The class driver presents the USB device to the system as an Ethernet port. It combines with HCC's Network Driver for Microchip LAN7500 and LAN9500 to provide the interface.

This module provides a host class driver for a USB stack. The system structure is shown below:



The host implementation allows the host system to transfer Ethernet packets to/from a remote Ethernet port at the other end of the USB connection. It connects to a USB device that has an Ethernet port on it that connects to a real or virtual network.

The lower layer interface is designed to use HCC Embedded's USB Host Interface Layer. This layer is standard over different host controller implementations; this means that the code is the same, whichever HCC USB host controller it is interfaced to. For detailed information about this layer, refer to the *HCC USB Host Base System User Guide* that is shipped with the base system.

The package provides a set of API functions for controlling access to a device. These are described here, with separate sections for module and device management.

1.2 Feature Check

The main features of the class driver are the following:

- It conforms to the HCC Advanced Embedded Framework.
- It can be used with or without an RTOS.
- It is compatible with all HCC USB host controllers.
- It supports multiple devices connected simultaneously.

This driver is compatible with the following Microchip LAN devices:

- LAN7500
- LAN89730
- LAN9500 and LAN9500A
- LAN9512, LAN9513 and LAN9514
- LAN9730

1.3 Device Description

This table summarizes the properties of the supported Microchip devices:

	Ethernet bandwidth	Additional features
LAN7500	10Base-T/100Base-TX/1000Base-T	
LAN89730	10Base-T/ 100Base-TX	HSIC interface
LAN9500	10Base-T/ 100Base-TX	
LAN9500A	10Base-T/ 100Base-TX	Lower power consumption
LAN9512	10Base-T/ 100Base-TX	Two port USB 2.0 hub
LAN9513	10Base-T/ 100Base-TX	Three port USB 2.0 hub
LAN9514	10Base-T/ 100Base-TX	Four port USB 2.0 hub
LAN9730	10Base-T/ 100Base-TX	External MII interface

1.4 Packages and Documents

Packages

The table below lists the packages that you need in order to use this module:

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
usbh_base	The USB host base package. This is the framework used by USB class drivers to communicate over USB using a specific USB host controller package.
nw_drv_eth_microchip_lan7500	The Network Driver for Microchip LAN7500 and LAN9500 package.
usbh_cd_microchip_lan7500	The USB device Microchip LAN7500 and LAN9500 host class driver package described by this document.

Documents

For an overview of HCC's embedded USB stacks, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC USB Host Base System User Guide

This document defines the USB host base system upon which the complete USB stack is built.

HCC Embedded USB Host LAN7500 and LAN9500 Class Driver User Guide

This is this document.

HCC Network Driver for Microchip LAN7500 and LAN9500 User Guide

This document describes the network driver for Microchip LAN7500 and LAN9500 devices.

1.5 Change History

This section includes recent changes to this product. For a list of all the changes, refer to the file **src/history/drivers/usb-host/usbh_cd_microchip_lan7500.txt** in the distribution package.

Version	Changes
1.02	Added support for LAN9500 devices.
1.01	Initial release.

2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

Note: Do not modify any files except the configuration file.

2.1 API Header File

The file **src/api/api_usbh_microchip_lan7500.h** must be included by any application using the system. This is the only file that should be included by an application using this module. For details of the API functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_usbh_microchip_lan7500.h** contains the configurable system parameters. Configure these as required. For details of the options, see Configuration Options.

2.3 Source Code File

The source code files are in the directory **src/usb-host/class-drivers/microchip_lan7500**. **These files should only be modified by HCC**.

File	Description
usbh_microchip_lan7500.c	Source file for LAN7500/LAN9500 code.
usbh_microchip_lan7500_regs.h	Header file for LAN7500/LAN9500 registers.

2.4 Version File

The file **src/version/ver_usbh_microchip_lan7500.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

3 Configuration Options

Set the system configuration options in the file **src/config/config_usbh_microchip_lan7500.h**.

USBH_MICROCHIP_LAN7500_MAX_UNITS

The maximum number of LAN7500 devices the system can handle. The default is 1.

USBH_MICROCHIP_LAN9500_ENABLE

Set this to 1 to use a LAN9500A device instead of the LAN7500. The default is 0.

4 Application Programming Interface

This section documents the Application Programming Interface (API). It includes all the functions that are available to an application program.

4.1 Module Management Functions

The functions are the following:

Function	Description
usbh_microchip_lan7500_init()	Initializes the module and allocates the required resources.
usbh_microchip_lan7500_start()	Starts the module.
usbh_microchip_lan7500_stop()	Stops the module.
usbh_microchip_lan7500_delete()	Deletes the module and releases the resources it used.

usbh_microchip_lan7500_init

Use this function to initialize the class driver and allocate the required resources.

Note: You must call this before any other function.

Format

```
int usbh_microchip_lan7500_init ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_start

Use this function to start the class driver.

Note: You must call **usbh_microchip_lan7500_init()** before this function.

Format

```
int usbh_microchip_lan7500_start ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_stop

Use this function to stop the class driver.

Format

```
int usbh_microchip_lan7500_stop ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_delete

Use this function to delete the class driver and release the associated resources.

Format

```
int usbh_microchip_lan7500_delete ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

4.2 Device Management Functions

The functions are the following:

Function	Description
usbh_microchip_lan7500_send()	Sends data through a channel.
usbh_microchip_lan7500_receive()	Receives available data.
usbh_microchip_lan7500_read_reg()	Reads one of the USB-Ethernet adapter's registers.
usbh_microchip_lan7500_write_reg()	Writes a value to one of the USB-Ethernet adapter's registers.
usbh_microchip_lan7500_mii_read()	Reads a PHY register.
usbh_microchip_lan7500_mii_write()	Writes a PHY register.
usbh_microchip_lan7500_eeprom_read()	Reads from EEPROM.
usbh_microchip_lan7500_eeprom_write()	Writes to EEPROM.
usbh_microchip_lan7500_eeprom_write_enable()	Enables writing to EEPROM.
usbh_microchip_lan7500_eeprom_write_disable()	Disables writing to EEPROM.
usbh_microchip_lan7500_eeprom_reload()	Reloads the EEPROM (refreshes the MAC address in the USB adapter).
usbh_microchip_lan7500_get_send_state()	Gets the completion code of the last send.
usbh_microchip_lan7500_get_mac_addr()	Gets a MAC address stored in EEPROM.
usbh_microchip_lan7500_set_mac_addr()	Writes a MAC address to EEPROM, then reloads the EEPROM content to apply the changes.
usbh_microchip_lan7500_get_port_hdl()	Gets the port handle.
usbh_microchip_lan7500_get_int_state()	Gets the interrupt status.
usbh_microchip_lan7500_set_filter()	Sets a filter.
usbh_microchip_lan7500_present()	Checks whether a LAN7500/LAN9500 device is connected.
usbh_microchip_lan7500_register_ntf()	Registers a notification function for a specified event type.

usbh_microchip_lan7500_send

Use this function to send data through a channel.

Format

```
int usbh_microchip_lan7500_send (
   t_usbh_unit_id   uid,
   uint8_t *        p_buf,
   uint32_t         length )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
p_buf	A pointer to the data to send.	uint8_t *
length	The number of bytes to send.	uint32_t

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_receive

Use this function to receive available data.

Format

```
int usbh_microchip_lan7500_receive (
    t_usbh_unit_id   uid,
    uint8_t *        p_buf,
    uint32_t         max_length,
    uint32_t *       rlength )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
p_buf	A pointer to the buffer which is to receive the data.	uint8_t *
max_length	The maximum length of the receive buffer.	uint32_t
rlength	Where to put the number of bytes written to the buffer.	uint32_t *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_read_reg

Use this function to read one of the USB-Ethernet adapter's registers.

Format

```
int usbh_microchip_lan7500_read_reg (
    t_usbh_unit_id    uid,
    uint32_t          reg,
    uint32_t *        p_data )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
reg	The register address.	uint32_t
data	On return, a pointer to the register's content.	uint32_t *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_write_reg

Use this function to write a value to one of the USB-Ethernet adapter's registers.

Format

```
int usbh_microchip_lan7500_write_reg (
    t_usbh_unit_id   uid,
    uint32_t         reg,
    uint32_t         data )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
reg	The register address.	uint32_t
data	The data to write.	uint32_t

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_mii_read

Use this function to read a PHY register.

Format

```
int usbh_microchip_lan7500_mii_read (
    t_usbh_unit_id   uid,
    uint8_t          reg_addr,
    uint32_t *       p_val )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
reg_addr	The register's address.	uint8_t
p_val	On return, a pointer to the PHY register's content.	uint32_t *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_mii_write

Use this function to write a PHY register.

Format

```
int usbh_microchip_lan7500_mii_write (
    t_usbh_unit_id   uid,
    uint8_t          reg_addr,
    uint32_t         val )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
reg_addr	The register's address.	uint8_t
val	The value to write.	uint32_t

Return Values

Return value	Description
USBH_SUCCESS.	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_eeprom_read

Use this function to read from EEPROM.

Format

```
int usbh_microchip_lan7500_eeprom_read (
    t_usbh_unit_id    uid,
    uint8_t           addr,
    uint8_t *         p_val )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
addr	The register to read.	uint8_t
p_val	On return, a pointer to the EEPROM value read.	uint8_t *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_eeprom_write

Use this function to write to EEPROM.

Format

```
int usbh_microchip_lan7500_eeprom_write (
    t_usbh_unit_id   uid,
    uint8_t          addr,
    uint8_t          val )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
addr	The register to write.	uint8_t
val	The value to write.	uint8_t

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_eeprom_write_enable

Use this function to enable writing to EEPROM.

Format

```
int usbh_microchip_lan7500_eeprom_write_enable ( t_usbh_unit_id uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_eeprom_write_disable

Use this function to disable writing to EEPROM.

Format

```
int usbh_microchip_lan7500_eeprom_write_disable ( t_usbh_unit_id uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_eeprom_reload

Use this function to reload the EEPROM (refresh the MAC address in the USB adapter).

Format

```
int usbh_microchip_lan7500_eeprom_reload ( t_usbh_unit_id uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_get_send_state

Use this function to get the completion code of the last send.

Note: This function is only required if an event is used for send.

Format

```
int usbh_microchip_lan7500_get_send_state ( t_usbh_unit_id uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_get_mac_addr

Use this function to get a MAC address stored in EEPROM.

Format

```
int usbh_microchip_lan7500_get_mac_addr (
    t_usbh_unit_id      uid,
    uint8_t * * const   p_mac_address )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
p_mac_address	Where to write the MAC address. This is a pointer to a 6 byte unsigned char array.	uint8_t * *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution; the device is connected and the MAC address is available.
Else	See Error Codes.

usbh_microchip_lan7500_set_mac_addr

Use this function to write a MAC address to EEPROM, then reload the EEPROM content to apply the changes.

Note: The change of MAC address is permanent.

Format

```
int usbh_microchip_lan7500_set_mac_addr (
    t_usbh_unit_id          uid,
    const uint8_t * const   p_mac_address )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
p_mac_address	A pointer to the MAC address to write (6 bytes).	uint8_t *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_get_port_hdl

Use this function to get the port handle.

Format

```
t_usbh_port_hdl usbh_microchip_lan7500_get_port_hdl ( t_usbh_unit_id uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
The port handle.	Successful execution.
USBH_PORT_HDL_INVALID	Invalid port handle.
Else	See Error Codes.

usbh_microchip_lan7500_get_int_state

Use this function to get the interrupt status.

Format

```
int usbh_microchip_lan7500_get_int_state (
    t_usbh_unit_id   uid,
    uint32_t *       int_flags )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
int_flags	On return, a pointer to the last data from the interrupt endpoint (3).	uint32_t *

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_set_filter

Use this function to set a filter.

Format

```
int usbh_microchip_lan7500_set_filter (
    t_usbh_unit_id   uid,
    uint32_t         filter )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
filter	The content of the RFE_CTL (Receive Filtering Engine Control) register.	uint32_t

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

usbh_microchip_lan7500_present

Use this function to check whether a LAN7500/LAN9500 device is connected or not.

Format

```
int usbh_microchip_lan7500_present ( t_usbh_unit_id uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
0	No LAN7500/LAN9500 device is present.
1	A device is present.
Else	See Error Codes.

usbh_microchip_lan7500_register_ntf

Use this function to register a notification function for the device.

When a device is connected or disconnected, or one of the specific events for this type of device occurs, the notification function is called.

Note: It is the user's responsibility to provide any notification functions required by the application. Providing such functions is optional.

Format

```
int usbh_microchip_lan7500_register_ntf (
    t_usbh_unit_id   uid,
    t_usbh_ntf       ntf,
    t_usbh_ntf_fn    ntf_fn )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
ntf	The notification ID.	t_usbh_ntf
ntf_fn	The notification function to be used when an event occurs.	t_usbh_ntf_fn

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See Error Codes.

4.3 Error Codes

If a function executes successfully it returns with a USBH_SUCCESS code, a value of 0. The following table shows the meaning of the error codes:

Return Code	Value	Description
USBH_SUCCESS	0	Successful execution.
USBH_SHORT_PACKET	1	IN transfer completed with short packet.
USBH_PENDING	2	Transfer still pending.
USBH_ERR_BUSY	3	Another transfer in progress.
USBH_ERR_DIR	4	Transfer direction error.
USBH_ERR_TIMEOUT	5	Transfer timed out.
USBH_ERR_TRANSFER	6	Transfer failed to complete.
USBH_ERR_TRANSFER_FULL	7	Cannot process more transfers.
USBH_ERR_SUSPENDED	8	Host controller is suspended.
USBH_ERR_HC_HALTED	9	Host controller is halted.
USBH_ERR_REMOVED	10	Transfer finished due to device removal.
USBH_ERR_PERIODIC_LIST	11	Periodic list error.
USBH_ERR_RESET_REQUEST	12	Reset request during enumeration.
USBH_ERR_RESOURCE	13	OS resource error.
USBH_ERR_INVALID	14	Invalid identifier/type (HC, EP HDL, and so on).
USBH_ERR_NOT_AVAILABLE	15	Item not available.
USBH_ERR_INVALID_SIZE	16	Invalid size.
USBH_ERR_NOT_ALLOWED	17	Operation not allowed.
USBH_ERROR	18	General error.

4.4 Types and Definitions

t_usbh_ntf_fn

The **t_usbh_ntf_fn** definition specifies the format of the notification function. It is defined in the USB host base system in the file **api_usb_host.h**.

Format

```
int  ( * t_usbh_ntf_fn )(
    t_usbh_unit_id   uid,
    t_usbh_ntf       ntf )
```

Arguments

Element	Type	Description
uid	t_usbh_unit_id	The unit ID.
ntf	t_usbh_ntf	The notification code.

Notification Codes

The standard notification codes shown below are defined in the USB host base system in the file **api_usb_host.h**.

Notification	Value	Description
USBH_NTF_CONNECT	1	Connection notification code.
USBH_NTF_DISCONNECT	2	Disconnection notification code.

The additional notification codes provided by this module are as follows:

Notification	Value	Description
USBH_NTF_MICROCHIP_LAN7500_RX_INT	USBH_NTF_CD_BASE + 1	Data received notification.
USBH_NTF_MICROCHIP_LAN7500_TX	USBH_NTF_CD_BASE + 2	Data sent notification.

5 Integration

This section specifies the elements of this package that need porting, depending on the target environment.

5.1 OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

The class driver uses the following OAL components:

OAL Resource	Number Required
Tasks	0
Mutexes	1
Events	0

5.2 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The class driver makes use of the following standard PSP function:

Function	Package	Component	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.

The module makes use of the following standard PSP macros:

Function	Package	Component	Description
PSP_RD_LE32	psp_base	psp_endianness	Reads a 32 bit value stored as little-endian from a memory location.
PSP_WR_LE16	psp_base	psp_endianness	Writes a 16 bit value stored as little-endian to a memory location.
PSP_WR_LE32	psp_base	psp_endianness	Writes a 32 bit value stored as little-endian to a memory location.

6 Sample Code

This section shows example code for the class driver.

6.1 Initialization

This example shows the code used to initialize a USB host with the class driver.

```
/*
** Initialize USB host with Microchip LAN7500/LAN9500 class driver.
*/

int usb_host_init ( void )
{
    int  rc;
    rc = hcc_mem_init();

    /* Initialize USB host module */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_init();
    }

    /* Initialize specific USB host controller */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_hc_init( 0, usbh_stm32uh_hc, 0 );
    }

    /* Initialize the LAN7500/LAN9500 Class Driver module */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_microchip_lan7500_init();
    }

    /* Start the LAN7500/LAN9500 Class Driver */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_microchip_lan7500_start();
    }

    /* Start the USB host stack */
    if ( rc == USBH_SUCCESS )
    {
        rc = usbh_start();  /* Start the USB host */
    }

    return rc;

} /* usb_host_init */
```