DHCP Server for IPv4 User Guide

Version 1.00

For use with DHCP Server for IPv4 module versions 1.03 and above

Date: 28-Jul-2017 11:22

Table of Contents

1 System Overview

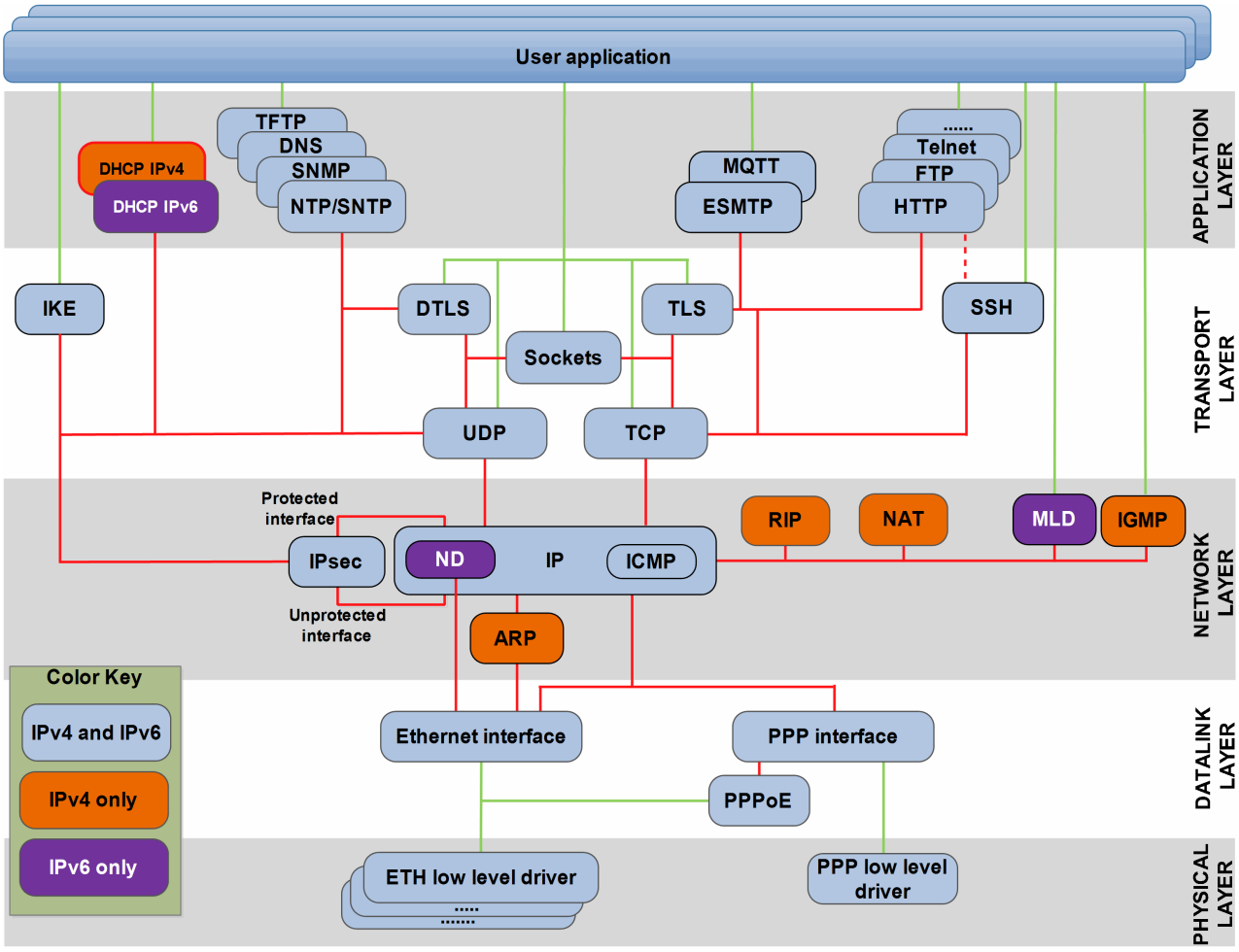
1.1 Introduction

This guide is for those who want to implement a Dynamic Host Control Protocol (DHCP) Server for IPv4 module as part of their TCP/IP stack. This DHCP Server module is used to configure dynamically IP addresses and other information that is needed for Internet communication.

This is a simple implementation of a DHCP server for use on closed networks of embedded devices where no DHCP server is available. One common use case is where a virtual network is created on an embedded device to connect to a PC host over USB, using a standard USB protocol such as CDC-ECM where the PC network interface needs to have a network address assigned to it.

This DHCP server allows the user to set up parameters including the range of IP addresses to be issued to connected devices, the lease time, and the network gateway and DNS server addresses.

The DHCP Server for IPv4 module is part of the HCC MISRA-compliant TCP/IP stack, as shown below, and is designed specifically for use with it. (In this diagram green lines show interfaces available to users of the stack, red lines show interfaces internal to the TCP/IP system.)



The DHCP server for IPv4 manages a pool of IPv4 addresses and information about client configuration parameters such as default gateway, domain name, and name servers.

When a DHCP client connects to a network, it sends a broadcast query to a DHCP server, requesting the necessary information. If the request is valid, the server assigns the client an IP address, a lease time (the length of time the allocation is valid for), and other IP configuration parameters, such as the subnet mask and the default gateway address. The query is typically sent straight after booting and must complete before the client can initiate IP-based communication with other hosts. Upon disconnection, the IP address is returned to the pool for use by another computer. In this way many computers can use the same IP address in a short time.

1.2 Feature Check

The main features of the system are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Complies with the HCC MISRA-compliant TCP/IP stack.
- Designed for integration with both RTOS and non-RTOS based systems.
- Supports IPv4 addresses.
- Compliant with RFC 2131.
- Number of clients is configurable.
- Address leasing is configurable.

1.3 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.
ip_app_dhcps_v4	The DHCP Server for IPv4 package described in this manual.
mip_base	The TCP/IP base package.
mip_udp	The UDP package.

Documents

For an overview of HCC's TCP/IP stack software, 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 TCP/IP Dual Stack System User Guide

This is the core document that describes the complete TCP/IP stack. It covers both IPv4 and IPv6 systems.

HCC DHCP Server for IPv4 User Guide

This is this document.

1.4 Change History

There is no change history for this document as this is the first version. This product has replaced the DHCP Server.

For the history of changes made to the package code itself, see History: ip_app_dhcps_v4.

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_ip_app_dhcps.h** 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_ip_app_dhcps.h** contains all the configurable parameters of the system. Configure these as required. For details of these options, see Configuration Options.

2.3 System File

The file **src/ip/apps/dhcps/dhcps.c** contains the source code. **This file should only be modified by HCC**.

2.4 Version File

The file **src/version/ver_ip_app_dhcps.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

All system configuration options are set in the file **src/config/config_ip_app_dhcps.h**. This section lists the available configuration options and their default values.

DHCPS_TASK_STACK_SIZE

The DHCP server task stack size. The default value is 1024.

DHCPS_MAX_CLIENT_NUM

The maximum number of parallel connected DHCP clients. The default value is 3.

DHCPS_MIN_LEASE_TIME

The minimum lease time in seconds. The default value is 3600.

DHCPS_MAX_LEASE_TIME

The maximum lease time in seconds. The default value is 24 * 3600.

DHCPS_PING_TIMEOUT

The maximum time for a ping timeout in seconds. The default value is 1.

DHCPS_CLIENT_ID_MAX_LEN

The maximum length of the client identification field. The default value is 16.

4 Application Programming Interface

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

4.1 Functions

The functions are the following:

Function	Description
dhcps_init()	Initializes the module and allocates the required resources.
dhcps_set_config()	Sets the DHCP server parameters to use.
dhcps_start()	Starts the module.
dhcps_stop()	Stops the module.
dhcps_delete()	Deletes the module and releases the resources it used.

dhcps_init

Use this function to initialize the DHCP server module and allocate the required resources.

> **Note:** Call this before any other function.

Format

```
t_dhcps_ret dhcps_init ( void )
```

Arguments

Argument
None.

Return Values

Return value	Description
DHCPS_SUCCESS	Successful execution.
DHCPS_ERR_OS	Error during allocation of OS resource.
DHCPS_ERROR	Operation failed.

dhcps_set_config

Use this function to set up the DHCP Server parameters to use. These parameters include the range of IP addresses to be issued to connected devices, the length of lease, and the network gateway and DNS server addresses.

> **Note:** This function must be called when the DHCP Server module is in the initialized state i.e. after **dhcps_init()** is called but before **dhcps_start()** is called. It may also be called before restarting the module after **dhcps_stop()** has been called.

Format

```
t_dhcps_ret dhcps_set_config (
    const t_ip_ifc_hdl              ifc_hdl,
    const t_dhcps_config * const    p_dhcps_config )
```

Arguments

Argument	Description	Type
ifc_hdl	The interface handle.	t_ip_ifc_hdl
p_dhcps_config	A pointer to the new configuration.	t_dhcps_config *

Return Values

Return value	Description
DHCPS_SUCCESS	Successful execution.
DHCPS_ERROR	Error during this call or **ip_get_config()** returned with error.

dhcps_start

Use this function to start the DHCP server module.

> **Note:** Call **dhcps_init()** before this function.

Format

```
t_dhcps_ret dhcps_start ( void )
```

Arguments

Argument
None.

Return Values

Return value	Description
DHCPS_SUCCESS	Successful execution.
DHCPS_ERROR	Operation failed.

dhcps_stop

Use this function to stop the DHCP server module.

Format

```
t_dhcps_ret dhcps_stop ( void )
```

Arguments

Argument
None.

Return Values

Return value	Description
DHCPS_SUCCESS	Successful execution.
DHCPS_ERROR	Operation failed.

dhcps_delete

Use this function to delete the DHCP server module and release the associated resources.

Format

```
t_dhcps_ret dhcps_delete ( void )
```

Arguments

Argument
None.

Return Values

Return value	Description
DHCPS_SUCCESS	Successful execution.
DHCPS_ERR_OS	Error during deletion of OS resource.
DHCPS_ERROR	Operation failed.

4.2 Error Codes

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

> **Note:** Also check error code values in the base system by using the *HCC TCP/IP Dual Stack System User Guide*.

Return Value	Value	Description
DHCPS_SUCCESS	0	Successful execution.
DHCPS_ERROR	1	Operation failed.
DHCPS_ERR_IP_ADDR	2	An IP address is invalid.
DHCPS_ERR_NO_MORE_ENTRY	3	No more entries are available.
DHCPS_ERR_INVALID_PARAM	4	A parameter is invalid.
DHCPS_ERR_INVALID_CONFIG	5	The configuration is invalid.
DHCPS_ERR_OS	6	Error during allocation or deletion of OS resource.

4.3 Types and Definitions

t_dhcps_config

The *t_dhcps_config* structure describes the DHCP server configuration. Its elements are as follows:

Element	Type	Description
dsc_ipaddr_min	uint32_t	The DHCP server IP address range minimum.
dsc_ipaddr_max	uint32_t	The DHCP server IP address range maximum.
dsc_netmask	uint32_t	The netmask address.
dsc_gateway	uint32_t	The gateway address.
dsc_dns1_addr	uint32_t	The first DNS server address.
dsc_dns2_addr	uint32_t	The second DNS server address.

5 Integration

This section describes all aspects of the DHCP module that require integration with your target project. This includes porting and configuration of external resources.

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.

This module uses the following OAL components:

OAL Resource	Number Required
Tasks	1
Mutexes	1
Events	1

5.2 Utilities

The DHCP code creates and uses a single timer in the **hcc_timer** module.

The **hcc_timer** module is included in your system when you install the base TCP/IP modules.

5.3 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 module makes use of the following standard PSP functions:

Function	Package	Element	Description
psp_memcmp()	psp_base	psp_string	Compares two blocks of memory.
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.

The module makes use of the following standard PSP macros:

Macro	Package	Element	Description
PSP_RD_BE16	psp_base	psp_endianness	Reads a 16 bit value stored as big-endian from a memory location.
PSP_RD_BE32	psp_base	psp_endianness	Reads a 32 bit value stored as big-endian from a memory location.
PSP_WR_BE32	psp_base	psp_endianness	Writes a 32 bit value to be stored as big-endian to a memory location.