UDP Test Suite User Guide

Version 1.40

For use with UDP Test Suite versions 4.09 and above

Date: 05-Sep-2017 11:56

Table of Contents

1 System Overview

1.1 Introduction

The HCC UDP Test Suite is a set of reference code for exercising the User Datagram Protocol (UDP) interface to the network stack. This provides a test suite for using both the native UDP interface and the Sockets interface. These tests are designed to:

- Provide reference code.
- Show you how to use the available API functions.
- Provide a basic test system to ensure the system works correctly.
- Measure the performance of UDP.
- Measure the resource utilization of UDP.

There are two types of test, as follows:

- **HCC API UDP Echo Test** – the test application opens two pairs of ports. The data received at one of these ports is echoed to its pair port.
- **Sockets API UDP Echo Test** – the test application has two sockets open, one for RX and one for TX. The data received by the remote host on RX is echoed back to the TX socket on the originating host.

Either test can be used together with a standard UDP test application (for example, **PCATTCP**) to measure performance.

This manual describes these tests and also the test suite Application Programming Interface (API).

> **Note:** UDP is not a reliable transport protocol as there is no acknowledgement process. Because of this, test results may be unreliable because packets sent may just be dropped. It is important to consider this when viewing any test results.

1.2 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_udp_test	The test package described in this document.
mip_udp	The UDP module.

Documents

For an overview of the HCC 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 to 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 UDP User Guide

This document describes the UDP module.

HCC UDP Test Suite User Guide

This is this document.

1.3 Change History

This section describes past changes to this manual.

- To download earlier manuals, see Archive: UDP Test Suite User Guide.
- For the history of changes made to the package code itself, see History: ip_udp_test.

The current version of this manual is 1.40. The full list of versions is as follows

Manual version	Date	Software version	Reason for change
1.40	2017-09-05	4.09	Corrected Packages list. Corrected version number in PDF.
1.30	2017-06-20	4.09	New *Change History* format.
1.20	2016-03-18	4.08	Added *Change History* section.
1.10	2014-08-19	4.05	First online version.

2 Source File List

The following sections describe 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_udp_test.h** should 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 these API functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_ip_udp_test.h** contains all the configurable parameters of the system. You can configure these as required. For detailed explanation of these options, see Configuration Options.

2.3 UDP Test System Files

These files in the directory **src/ip/stack/udp/test** are described in Running Tests. **These files should only be modified by HCC**.

File	Description
udp_test.c	HCC API UDP echo test application.
udp_test_socket.c	Sockets API UDP echo test application.

2.4 Version File

The file **src/version/ver_ip_udp_test.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 **src/config/config_ip_udp_test.h** configuration file. This section lists the available configuration options and their default values.

USE_UDP_SOCKET_TEST

Keep the default of 0 to use the HCC API UDP Echo Test. Set this to 1 to use the Sockets API UDP Echo Test.

3.1 HCC API UDP Echo Test Options

UDP_RX_PORT

The Receive port. The default is 279.

UDP_TX_PORT

The Transmit port. The default is 280.

UDP_TEST_TASK_STACK_SIZE

The test stack size. The default is 1024.

3.2 Sockets API UDP Echo Test Options

> **Note:** These only apply if USE_UDP_SOCKET_TEST is set to 1.

USE_POLL

Set this to 1 to use **socket_poll()** to check for received data. The default is 0.

USE_SELECT

Set this to 1 to use **socket_select()** to check for received data. The default is 0.

UDP_TEST_NONBLOCKING

Set this to 1 for non-blocking tests. The default is 0.

BUFFER_SIZE

The Sockets buffer size. The default is 1460.

4 Running Tests

The two types of test are described below. Both of the tests use one task.

Before running either test type:

- Set the relevant options in the configuration file.
- Use **udp_test_init()** to initialize the UDP test module.

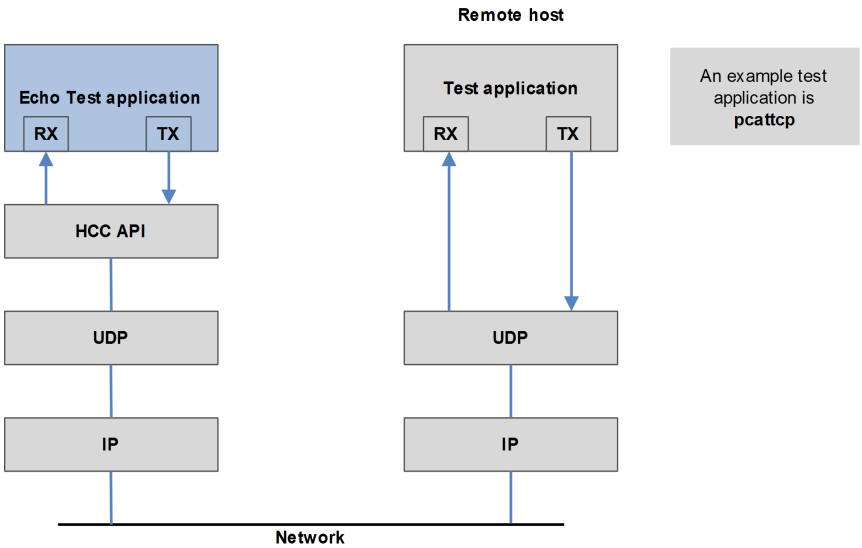
Note: You can download the PCAUSA Test TCP (PCATTCP) test utility from the PCAUSA website at: http://www.pcausa.com/Utilities/utilities.htm

4.1 HCC API UDP Echo Test

This module is a UDP protocol test application. The test application opens two pairs of ports; data received at one of these ports is echoed to its pair port. This test can be used together with standard UDP test applications (for example, **PCATTCP**) to measure UDP performance.

This test uses one task.

This diagram shows this type of test in operation:



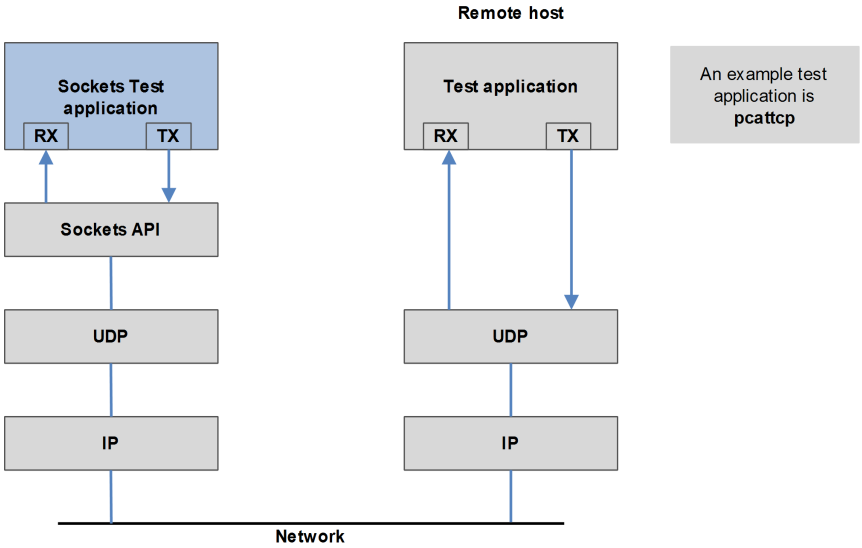
This is the file **udp_test.c**. This test echoes back all incoming data on UDP_RX_PORT to UDP_TX_PORT.

4.2 Sockets API UDP Echo Test

In this test, the test application has two sockets open, one for RX and one for TX. Data received by the remote host on RX is echoed back to the TX socket on the originating host. This test echoes back all incoming data on UDP_RX_PORT to UDP_TX_PORT. This test can be used together with standard UDP test applications (for example, **pcattcp**) to measure UDP performance.

This test uses one task.

This diagram shows this type of test in operation:



This is the file **udp_test_socket.c**. In the configuration file, set USE_UDP_SOCKET_TEST to 1 to use this type of test. You can also specify whether to use polling, and whether to run tests as non-blocking.

5 Application Programming Interface

This section documents the single Application Programming Interface (API) function and the error codes.

5.1 udp_test_init

Use this function to initialize the UDP test module and allocate the required resources.

Format

```
t_ip_ret udp_test_init ( void )
```

Arguments

Argument
None.

Return Values

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

5.2 Error Codes

The following table shows the meaning of the IP return codes. The test suite may not produce all of these errors.

Return Value	Value	Description
IP_SUCCESS	0	Successful execution.
IP_MORE_DATA	1	There is more data pending.
IP_CONN_PENDING	2	Connection initiated but not complete.
IP_DISCONNECT_WAIT	3	Waiting for completion of the disconnection process.
IP_ERROR	4	An unspecified error occurred.
IP_ERR_OS	5	OS resource creation error.
IP_ERR_INIT	6	Initialization error.
IP_ERR_NWDRV	7	Network Driver error.
IP_ERR_NOT_READY	8	The connection is not ready.
IP_ERR_NOT_CONFIGURED	9	Incompatible with the current configuration.
IP_ERR_NO_DATA	10	No data available.
IP_ERR_NO_MORE_ENTRY	11	More tasks are trying to access the stack than the system is configured for; IP_MAX_TASK has been exceeded.
IP_ERR_NO_CONNECTION	12	This connection does not exist.
IP_ERR_NO_BUFFER	13	Function **tcp_get_buf()** failed to allocate a buffer due to an empty buffer pool.
IP_ERR_INVALID_PARAM	14	There is an invalid parameter in the requested operation.
IP_ERR_INVALID_HDL	15	An invalid handle was used in the requested operation.
IP_ERR_INVALID_FRAME	16	An invalid frame was received.
IP_ERR_INVALID_PROTOCOL	17	An IP frame with an invalid protocol identifier was received.
IP_ERR_INVALID_SIZE	18	There is an error in the size field of the received IP frame.

Return Value	Value	Description
IP_ERR_INVALID_REQUEST	19	An invalid operation was requested.
IP_ERR_INVALID_STATE	20	An invalid state has been reached.
IP_ERR_INVALID_CONFIG	21	One of the following: • An interface was started and the interface has an associated pool with an invalid configuration. • An interface was added which wants to use an active pool with network driver parameters that are incompatible with it. • Pool properties were manually configured but the maximum required buffer size is smaller than that requested by the added interface.
IP_ERR_INVALID_BUFFER	22	An interface was started but the required pool buffer queue properties can't be applied for the pool.
IP_ERR_INVALID_POOL	23	An interface was started without an associated pool.
IP_ERR_POOL_BUSY	24	A pool wants to be deleted but the system did not release all buffers in the active pool. This can only happen if the user has obtained frame buffers by using **tcp_get_buf()** and these have not been released yet by using **tcp_release_buf()**.
IP_ERR_ROUTE	25	The specified route is not working.
IP_ERR_LINK_DOWN	26	The physical link requested is down.
IP_ERR_PORT_OPENED	27	The requested port is already open.
IP_ERR_BAD_CHECKSUM	28	A frame has been received with a checksum error.
IP_ERR_DUP_FRAGMENT	29	A duplicate fragment was received.
IP_ERR_TASK_NOT_FOUND	30	Task associated with operation does not exist.
IP_ERR_TIMEOUT	31	Operation timed out.

6 Integration

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

6.1 OS Abstraction Layer

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

The test suite uses the following OAL components:

OAL Resource	Number Required
Tasks	1
Mutexes	0
Events	0

6.2 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 module makes use of the following standard PSP function:

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

The module does not make use of any standard PSP macros.