TCP Test Suite User Guide

Version 1.50

For use with TCP Test Suite versions 3.14 and above

Date: 05-Sep-2017 12:50

Table of Contents

1 System Overview

1.1 Introduction

The HCC TCP Test Suite is a set of reference code for exercising the TCP interface to the network stack. This provides a test suite for using both the native TCP 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 the TCP/IP stack.
- Measure the resource utilization of the TCP/IP stack.

There are three types of test, as follows:

- **HCC API TCP Echo Test** – the test application opens two pairs of ports. The data received at one of these ports is echoed to its pair port. This test can be used together with standard TCP/IP test applications (for example, **PCATTCP** or HCC's loopback test) to measure stack performance.
- **HCC API Loopback Test** – all the data sent to the TX port is looped back to the RX port via the internal IP loopback driver.
- **Sockets API TCP 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. This test can be used together with standard TCP/IP test applications (for example, **PCATTCP** or HCC's loopback test) to measure stack performance.

This manual describes all three of these tests and also the test suite API.

Note: These tests do not attempt comprehensive verification of the TCP/IP stack, for example coverage testing. For more comprehensive TCP/IP stack validation, please contact HCC Embedded.

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.
mip_tcp	The TCP module.
ip_tcp_test	The test package described in this document.

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 TCP 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: TCP Test Suite User Guide.
- For the history of changes made to the package code itself, see History: ip_tcp_test.

The current version of this manual is 1.50. The full list of versions is as follows

Manual version	Date	Software version	Reason for change
1.50	2017-09-05	3.14	Changed *Packages* list.
1.40	2017-06-20	3.14	New *Change History* format.
1.30	2015-12-07	3.13	Added more options.
1.20	2015-07-23	3.11	Added *Change History* section.
1.10	2014-08-19	3.10	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_tcp_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_tcp_test.h** contains all the configurable parameters of the system. You can configure the parameters as required. For detailed explanation of these options, see Configuration Options.

2.3 TCP Test System Files

These files in the directory **src/ip/stack/tcp/test** are described in Running Tests.

File	Description
tcp_test.c	HCC API TCP echo test application.
tcp_test_loopback.c	HCC API loopback test application.
tcp_test_socket.c	Sockets API TCP echo test application.

2.4 Version File

The file **src/version/ver_ip_tcp_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 file **src/config/config_ip_tcp_test.h**. This section lists the available options and their default values.

USE_TCP_SOCKET_TEST

Set this to 0 to use native TCP interface test or to 1 to use TCP socket interface test. The default is 0.

3.1 HCC API TCP Echo Test Options

TCP_TEST_TASK_STACK_SIZE

The echo test stack size. The default is 1024.

TCP_TEST_COUNT

The number of tasks to use. If you set this to 2, each task uses one pair of ports. The default is 1.

TCP_TX_CONN_LOCAL_PORT_0

The local port for the first initiated connection. The default is 6125.

TCP_RX_CONN_LOCAL_PORT_0

The first Receive port. The default is 179.

TCP_TX_CONN_REMOTE_PORT_0

The first Transmit port. The default is 180.

> **Note:** The following options are only used if TCP_TEST_COUNT is 2.

TCP_TX_CONN_LOCAL_PORT_1

The local port for the second initiated connection. The default is IP_PORT_AUTO_ASSIGN; this assigns the port automatically.

TCP_RX_CONN_LOCAL_PORT_1

The second Receive port. The default is 279.

TCP_TX_CONN_REMOTE_PORT_1

The second Transmit port. The default is 280.

3.2 HCC API Loopback Test Options

TCP_LB_TEST_TASK_STACK_SIZE

The loopback test stack size. The default is 512.

TCP_LB_BUF_CHECK

Keep the default of 1 to use loopback buffer checking. Otherwise, set it to 0.

TCP_LB_TX_SIZE

The packet size to use for loopback. The default is (4 * 1024 * 1024).

TCP_LB_TX_CONN_LOCAL_PORT

The local port for the initiated connection. The default is IP_PORT_AUTO_ASSIGN; this assigns the port automatically.

TCP_LB_RX_CONN_LOCAL_PORT

The loopback Receive local port. The default is TCP_TX_CONN_REMOTE_PORT_1.

TCP_LB_TX_CONN_REMOTE_PORT

The loopback Transmit remote port. The default is TCP_TX_CONN_REMOTE_PORT_1.

3.3 Sockets API TCP Echo Test Options

TCP_TEST_TASK_STACK_SIZE

The socket test stack size. The default is 1024.

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.

TCP_TEST_NONBLOCKING

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

CLIENT_LOCAL_PORT_ASSIGN

When this is set to 1 (the default), TCP_TX_CONN_LOCAL_PORT defines the local port. Set this to 0 if you want the port to be auto-assigned.

BUFFER_SIZE

The socket buffer size. The default is 1460.

TCP_TX_CONN_LOCAL_PORT

The socket Transmit local port. The default is 6125. Note that if CLIENT_LOCAL_PORT_ASSIGN is set to 0, this port is auto-assigned instead.

TCP_RX_CONN_LOCAL_PORT

The socket Receive local port. The default is 179.

TCP_TX_CONN_REMOTE_PORT

The socket Transmit remote port. The default is 180.

4 Running Tests

The three types of test are described below.

Set the relevant options in the configuration file before running any test type.

Note:

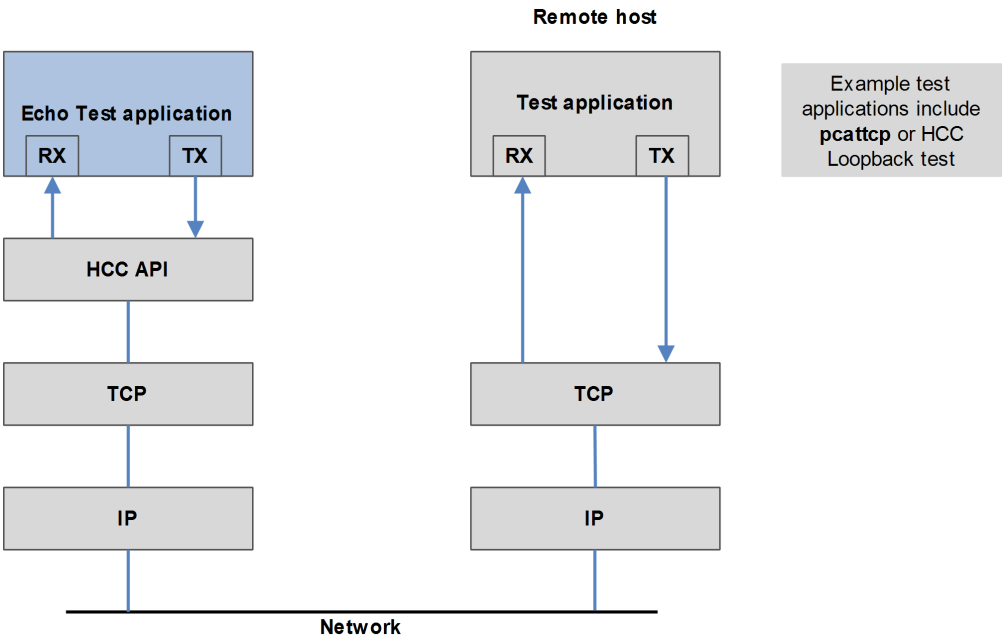
- The number of tasks used may be more than one. The TCP Echo test uses TCP_TEST_COUNT tasks; it can be configured to start two tasks, one for each pair of ports.
- The loopback test and Sockets test both use one task.
- You can download the PCAUSA Test TCP (PCATTCP) test utility from the PCAUSA website at: http://www.pcausa.com/Utilities/utilities.htm

4.1 HCC API TCP Echo Test

This module is a TCP 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 TCP/IP test applications (for example, **PCATTCP** or HCC's loopback test) to measure stack performance.

> **Note:** This code uses the incoming network buffer to send the data; no allocation or memory copying is required. When converting this for use with your real application, you must take into account how to allocate and free the network buffers.

This diagram shows this type of test in operation:



This is the file **tcp_test.c**. This test echoes back all incoming data on TCP_RX_CONN_LOCAL_PORT_0 to TCP_TX_CONN_REMOTE_PORT_0.

Use **tcp_test_init()** to initialize the TCP test unit.

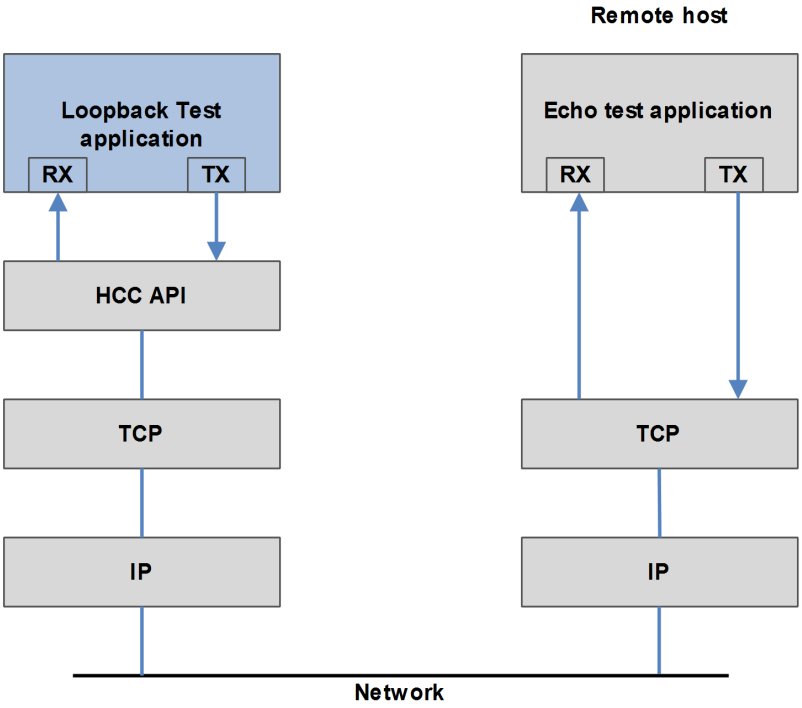
This test uses TCP_TEST_COUNT tasks. This number may be more than 1: the test can be configured to start two tasks, one for each pair of ports.

4.2 HCC API Loopback Test

This module is a TCP protocol loopback test application. All data sent to the transmit port is looped back to the receive port via the internal IP loopback driver.

> **Note:** The loopback test can be used with either the HCC API TCP Echo test or Sockets API TCP Echo test running on a separate unit.

This diagram shows this type of test in operation:



This is the file **tcp_test_loopback.c**, which must be used together with **tcp_test.c**. Use **tcp_lb_test_init()** to initialize the loopback TCP test unit and **tcp_lb_test_run()** to start execution of TCP tests on the loopback port.

This test uses one task.

This test connects to the **tcp_test.c** TCP_RX_CONN_LOCAL_PORT_n and listens for a connection coming from the **tcp_test.c** TCP_TX_CONN_REMOTE_PORT_n.

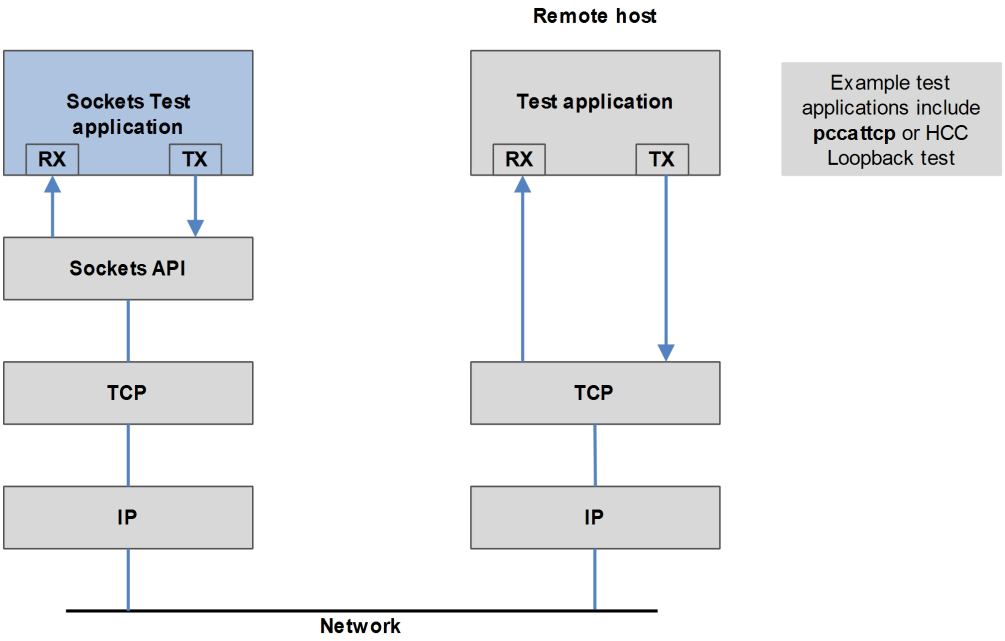
You must set TCP_LB_RX_CONN_LOCAL_PORT to TCP_TX_CONN_REMOTE_PORT_1 and TCP_LB_TX_CONN_REMOTE_PORT to TCP_RX_CONN_LOCAL_PORT_1.

All data transmitted on TCP_LB_TX_CONN_REMOTE_PORT are echoed back by **tcp_test.c** to TCP_LB_RX_CONN_LOCAL_PORT and the data are checked.

4.3 Sockets API TCP 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 TCP_RX_CONN_LOCAL_PORT to TCP_TX_CONN_REMOTE_PORT.

This diagram shows this type of test in operation:



This is the file **tcp_test_socket.c**. In the configuration file, set USE_TCP_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.

This test uses one task.

5 Application Programming Interface

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

5.1 Functions

The functions are the following:

Function	Description
tcp_test_init()	Initializes the test module and allocates the required resources.
tcp_lb_test_init()	Initializes the loopback test module and allocates the required resources.
tcp_lb_test_run()	Starts execution of TCP tests on the loopback port.

tcp_test_init

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

Format

```
t_ip_ret tcp_test_init ( void )
```

Arguments

Argument
None.

Return Values

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

tcp_lb_test_init

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

Format

```
t_ip_ret tcp_lb_test_init ( void )
```

Arguments

Argument
None.

Return Values

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

tcp_lb_test_run

Use this function to start execution of TCP tests on the loopback port.

Format

```
t_ip_ret tcp_lb_test_run ( const t_ip_addr * const p_ipaddr )
```

Arguments

Argument	Description	Type
ipaddr	A pointer to the IP address to connect to (the loopback IP address).	t_ip_addr *

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) that allows the module 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	1