

Micrium

© Copyright 2005-2006, Micrium
All Rights reserved

μC/TTCP

Test TCP

User's Manual

www.Micrium.com

Empowering Embedded Systems

Table of Contents

1.00	Introduction	3
1.01	Directories and Files	3
2.00	Network setup	5
3.00	PCATTCP	6
3.01	Network Analyzer (Sniffer)	7
4.00	μC/TTCP	8
4.01	μC/TTCP command line	8
4.02	TCP Transmit Test	10
4.03	TCP Receive Test	12
4.04	UDP Transmit Test	14
4.05	UDP Receive Test	16
5.00	μC/TTCP	18
5.01	TTCP Code, tcp.c	18
5.02	TTCP startup code : main()	19
5.03	TTCP startup code : AppStartTask()	20
6.00	μC/TTCP module limitations	21
6.01	IP transmit fragmentation	21
6.02	Target performance	21
	References	22
	Contacts	22

1.00 Introduction

Test TCP or TTCP, is a test tool to perform TCP/IP or UDP/IP performance tests. TTCP is a command-line sockets-based benchmarking tool for measuring performance between two systems. It was originally developed in 1984 by Mike Muuss and Terry Slattery for the BSD operating system. The original TTCP and sources are in the public domain, and copies are available from many anonymous FTP sites.

This document describes how to configure and use the μ C/TTCP module in a μ C/TCP-IP and μ C/OS-II environment.

We used the Cogent CSB337 single-board computer and IAR's Embedded Workbench to demonstrate the examples but other embedded platforms and tool chains can be used.

1.01 Directories and Files

The code and documentation of the μ C/TTCP module are placed in a directory structure according to "AN-2002, μ C/OS-II Directory Structure". Specifically, the files are placed in the following directories:

\Micrium\Software\uC-TTCP\Doc

This directory contains the μ C/TTCP documentation files, including this one.

\Micrium\Software\uC-TTCP\Source

This directory contains the μ C/TTCP source files (`ttcp.c` and `ttcp.h`).

\Micrium\Software\uC-LIB

This directory contains the μ C/LIB source files, the Micrium version of the C library most used utilities. The goal of this library is to ease a certification process by providing the source code and documentation for all the functions. This directory contains:

```
lib_def.h
lib_mem.c
lib_mem.h
lib_str.c
lib_str.h
```

\Micrium\Software\uC-CPU

This directory contains the μ C/CPU source files (`cpu_def.h`). This file provides definition for byte alignment, endianness and critical section handling.

\Micrium\Software\uC-CPU\ARM\IAR

This directory contains the files to configure standard data types, cpu word configuration, critical section configuration and cpu data types for the IAR compiler used for this sample project. (`cpu.h` and `cpu_a.s`)

\Micrium\Software\EvalBoards\Cogent\CSB337\IAR\uC-APPS\Ex1

This directory contains the source code for Example #1 running on a Cogent CSB337 with IAR tools. This directory contains:

```
app_cfg.c
includes.h
net_cfg.h
os_cfg.h
```

- `app_cfg.c` contains the application OS task priorities and stack space;
- `includes.h` contains a master include file used by the application;
- `net_cfg.h` is the μ C/TCP-IP configuration file;
- `os_cfg.h` is the μ C/OS-II configuration file;

This directory also contains the IAR EWARM 4.31a project workspace files:

```
Ex1.ewd
Ex1.ewp
Ex1.eww
```

\Micrium\Software\EvalBoards\Cogent\CSB337\IAR\BSP

This directory contains the Board Support Package (BSP) files and linker file for the project and target development board used for the example used to demonstrate TTCP. This directory contains:

```
bsp.c
bsp.h
CSB33z_lnk_ram.xcl
net_bsp.c
```

- `bsp.c` contains the board support functions for Interrupts, Timers, LEDs and Serial port;
- `bsp.h` contains the header definition for `bsp.c`;
- `CSB33x_lnk_ram.xcl` is the linker file for the IAR tool;
- `net_bsp.c` contains the low level interface functions for the Ethernet controller used in the sample project;

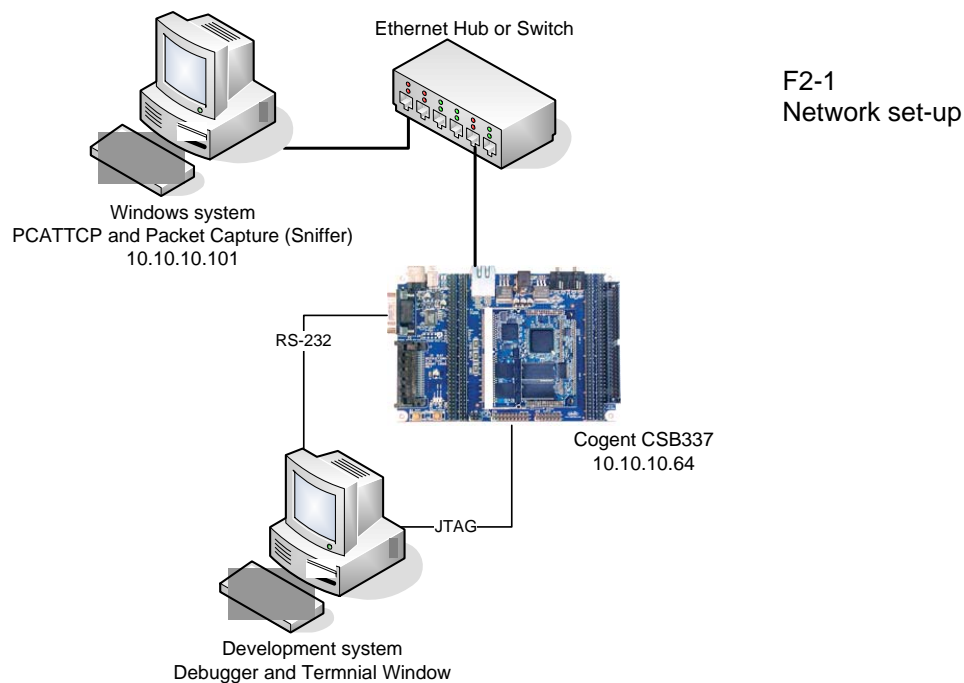
2.00 Network setup

Even if TTCP stands for Test TCP, the TTCP module can perform the following tests:

- Transmitting TCP segments : also called TCP Client mode
- Receiving TCP segments : also called TCP Server mode
- Transmitting UDP datagrams : also called UDP Client mode
- Receiving UDP datagrams : also called UDP Server mode

TTCP always work in a peer to peer configuration.

In developing μ C/TTCP we were always running on a Cogent CSB337 and we were using PCATTCP on a PC.



The target system is running Micrium μ C/TTCP.

The Windows station also needs to run a TTCP application. The next section describes the Windows TTCP implementation Micrium has selected.

3.00 PCATTCP

For our development, Micrium has selected the PCAUSA port of TTCP to Windows Sockets for the TTCP module running on the Windows host. It is called pcattcp. The following directory contains pcattcp:

`\Micrium\Software\uC-TTCP\pcattcp`

This directory contains the following files:

```
PCATTCP.chm
PCATTCP.exe
RELEASE.TXT
sourcesv2.zip
ttcpzip.exe
```

- PCATTCP.chm is the pcattcp html help file;
- PCATTCP.exe is the tool itself;
- RELEASE.TXT contains release information for the pcattcp versions;
- sourcesv2.zip is a winzip file containing the pcattcp sources;
- ttcpzip.exe a self-extract zip file containing the 4 files above (the download result).

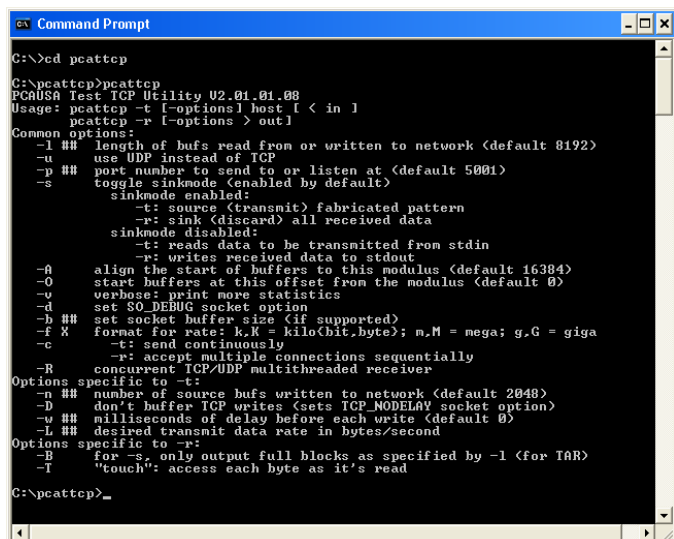
You can also download pcattcp from <ftp://ftp.pcausa.com/utilities/ttcpzip.exe>

You will find more information about pcattcp at <http://www.pcausa.com/Utilities/pcattcp.htm>

Running pcattcp

- Copy the content of ttcpzip.exe to a directory (C:\pcattcp for example).
- Open a DOS prompt window
- Change directory to the directory now containing pcattcp.exe
- From the DOS prompt, start pcattcp:
`C:\pcattcp>pcattcp`

Without any arguments, pcattcp will output the command usage. For more detailed information about the pcattcp commands, please refer to the html help file by double clicking on PCATTCP.chm. This will automatically open a help file.



```

C:\>cd pcattcp
C:\pcattcp>pcattcp
PCAUSA Test TCP Utility V2.01.01.00
Usage: pcattcp -t [-options] host [ < in |
        pcattcp -r [-options] out]
Common options:
-l ## length of bufs read from or written to network (default 8192)
-u use UDP instead of TCP
-p ## port number to send to or listen at (default 5001)
-s toggle sinkmode (enabled by default)
   sinkmode enabled:
       -t: source (transmit) fabricated pattern
       -r: sink (discard) all received data
   sinkmode disabled:
       -t: reads data to be transmitted from stdin
       -r: writes received data to stdout
-a align the start of buffers to this modulus (default 16384)
-o start buffers at this offset from the modulus (default 0)
-v verbose: print more statistics
-d set SO_DEBUG socket option
-b ## set socket buffer size (if supported)
-f X format for rate: K,M = kilo(byte); m,M = mega; g,G = giga
-c -t: send continuously
   -r: accept multiple connections sequentially
-R concurrent TCP/UDP multithreaded receiver
Options specific to -t:
-n ## number of source bufs written to network (default 2048)
-D don't buffer TCP writes (sets TCP_NODELAY socket option)
-w ## milliseconds of delay before each write (default 0)
-L ## desired transmit data rate in bytes/second
Options specific to -r:
-B for -s, only output full blocks as specified by -l (for IAR)
-T "touch": access each byte as it's read
C:\pcattcp>
```

F3-1

3.01 Network Analyzer (Sniffer)

In our test setup, we also used a Software based Network Analyzer (sometimes called a Sniffer). Micrium used Etherpeek. Though you can see global network statistics without capturing packets, for some analysis sessions you'll want to capture packets.

Etherpeek is a commercial product of WildPackets Inc. To find more information about Etherpeek, please use the following link: http://www.wildpackets.com/products/etherpeek_nx

Public domain Network Analyzers are also available. One of the most popular that we can also recommend is Ethereal. You can download Ethereal from <http://www.ethereal.com/>

4.00 μ C/TTCP

μ C/TTCP is compliant with the other TTCP tools available in the public domain. It was written for target systems running μ C/OS-II and μ C/TCP-IP. This section describes the μ C/TTCP usage.

4.01 μ C/TTCP command line

As soon as the application is launched on the target, via a JTAG interface or other means, the Terminal Window running on the Development system will display the application status and a command line. The user controls μ C/TTCP by entering parameters on the command line.

The command usage is displayed if the user makes an error entering the command line or by pressing ENTER at the ">" command prompt. Here is the command usage:

```
Usage: -t [-options] host
        -r [-options]
Common options: \r\n");
    -l ##    length of buffers read from or written to network (default 8192)
    -u        use UDP instead of TCP
    -p ##    port number to send to or listen at (default 5001)
    -s -t:   source a pattern to network
            -r: sink (discard) all data from network
    -d        set SO_DEBUG socket option (not supported)
    -b ##    set socket buffer size (not supported)
    -f X      format for rate: k,K = kilo{bit,byte}; m,M = mega; g,G = giga
Options specific to -t:
    -n ##    number of source buffers written to network (default 2048)
    -D        don't buffer TCP writes (sets TCP_NODELAY socket option)
Options specific to -r:
    -B        for -s, only output full blocks as specified by -l
    -T        "touch": access each byte as it is read
```

As an example, to start the TTCP TCP Receive Test start μ C/TTCP with the "-r" option.

```
>-r
```

The TCP server will start and then wait until a remote TTCP client makes a connection attempt.

Instead of explaining all of the μ C/TTCP options, we will be going through a few examples that illustrate basic usage.

Sinkmode (-s)

The simplest and most popular TTCP mode of operation is called "sinkmode". In this mode of operation the TTCP transmitter sends a fabricated data pattern and the TTCP receiver simply sinks (discards) any data that it receives.

This is the default TTCP mode of operation.

Standard Streams

Alternatively, TTCP can use what is called "standard streams" or "standard I/O". With μ C/TTCP, the Serial port is used as the standard I/O.

Standard stream has only been implemented in Receiver mode. The code for Transmitter mode still needs to be developed.

Defaults

The μ C/TTCP defaults for the command parameters are:

Parameter	Value	Command line parameter
Buffer Length	8192	-l
Number of buffers used	2048	-n
Transport layer protocol	TCP	-u
Layer 4 port number	4096	-p
Receiver/Transmitter	Receiver mode	-r or -t
Sink mode	Enabled	-s
Block read	Disabled	-B
Output format	'Kilobits per second'	-f
Received data processing	Disabled	-T
Socket options	Not supported	-d
Buffer TCP writes	Not supported	-D

T4-1

4.02 TCP Transmit Test

Target

To start the TTCP TCP Transmit Test start μ C/TTCP with the "-t" option followed by the dotted IP address of the remote TTCP client.

```
>-t 10.10.10.101
```

```
ttcp-t: BufLen=8192, NumBuf=2048, port=4096  tcp  -> 10.10.10.101
ttcp-t: Client socket 9 opened
ttcp-t: Client socket 9 connected
ttcp-t: Client socket 9 closed
ttcp-t: 16777216 bytes in 41.87 real seconds = 3130.30 Kbit/sec +++
ttcp-t: 2099 I/O calls, msec/call = 19.95, calls/sec = 50.13
```

μ C/TCP-IP Performance measurements

Type the TTCP parameters and parameter values and press Enter

>

These messages are generated as the test starts up:

```
ttcp-t: BufLen=8192, NumBuf=2048, port=4096  tcp  -> 10.10.10.101
ttcp-t: Client socket 9 opened
```

These messages indicate that a connection has been established with the remote host:

```
ttcp-t: Client socket 9 connected
```

When the test exits it displays the test results:

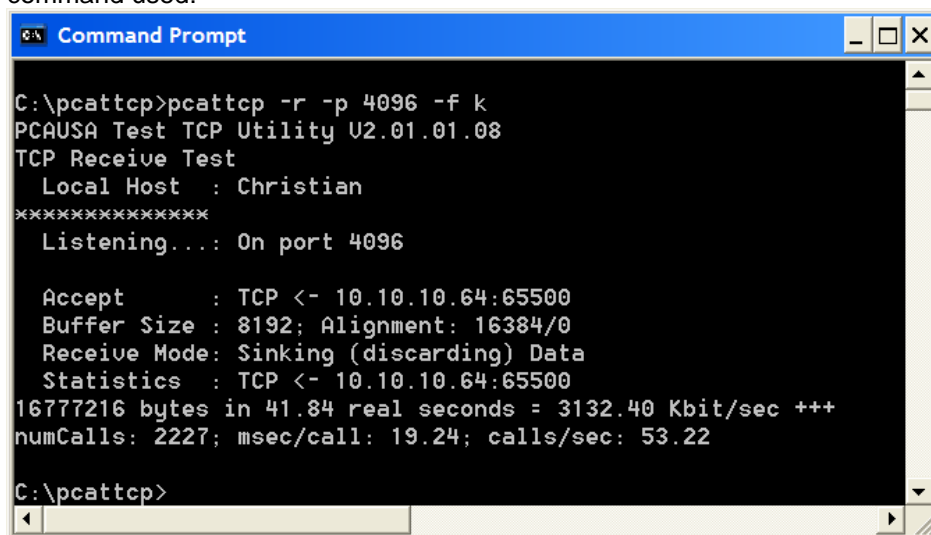
```
ttcp-t: Client socket 9 closed
ttcp-t: 16777216 bytes in 38.38 real seconds = 3415.38 Kbit/sec +++
ttcp-t: 2111 I/O calls, msec/call = 18.18, calls/sec = 55.01
```

As you can see all the default parameters are used:

- TCP is the default Layer 4 protocol used.
- Sinkmode is enabled by default
- Number of buffers : 2048
- Length of buffers : 8192
- TCP port : 4096 (Note: public domain TTCP uses default port 5001. But recent firewall and virus protection software filters port 5001. We have selected port 4096 for μ C/TTCP. If you encounter problems with port 4096, please select another port that your network setup will not block.)

Windows host TTCP session

The Windows host running pcattcp must initiate the Receiver part of this test. Figure 4-1 shows the command used:



```
C:\pcattcp>pcattcp -r -p 4096 -f k
PCAUSA Test TCP Utility U2.01.01.08
TCP Receive Test
Local Host : Christian
*****
Listening...: On port 4096

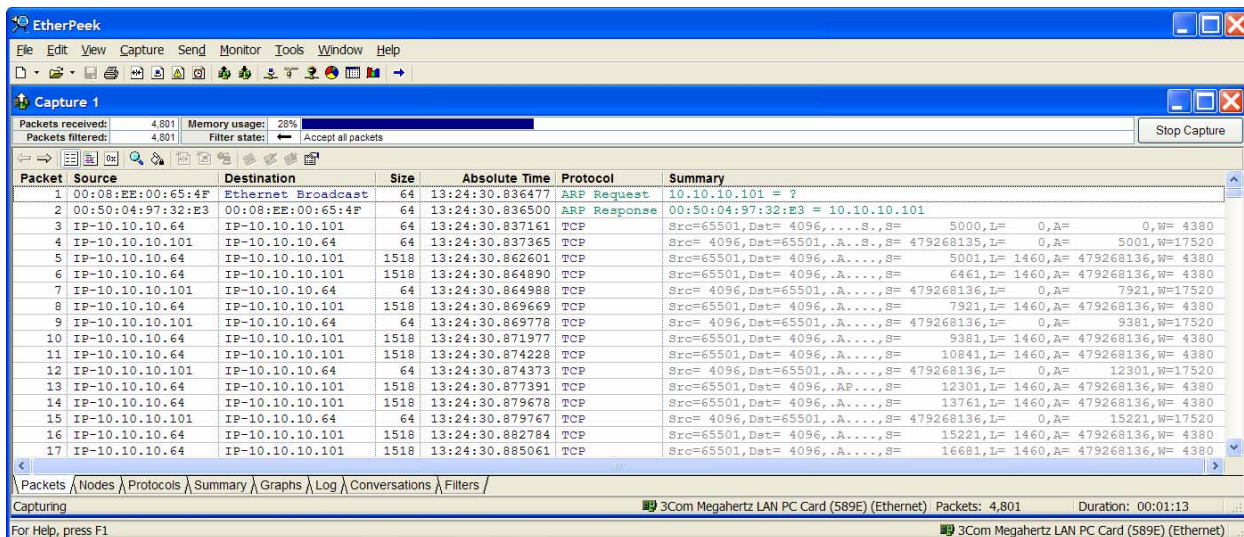
Accept      : TCP <- 10.10.10.64:65500
Buffer Size : 8192; Alignment: 16384/0
Receive Mode: Sinking (discarding) Data
Statistics  : TCP <- 10.10.10.64:65500
16777216 bytes in 41.84 real seconds = 3132.40 Kbit/sec +++
numCalls: 2227; msec/call: 19.24; calls/sec: 53.22

C:\pcattcp>
```

F4-1

Windows host packet capture

Using Etherpeek, here is a screen shot of the Transmit test:



Packet	Source	Destination	Size	Absolute Time	Protocol	Summary
1	00:08:EE:00:65:4F	Ethernet Broadcast	64	13:24:30.836477	ARP Request	10.10.10.101 = ?
2	00:50:04:97:32:E3	00:08:EE:00:65:4F	64	13:24:30.836500	ARP Response	00:50:04:97:32:E3 = 10.10.10.101
3	IP-10.10.10.64	IP-10.10.10.101	64	13:24:30.837161	TCP	Src=65501,Dst= 4096,....S.,S= 5000,L= 0,A= 0,W= 4380
4	IP-10.10.10.101	IP-10.10.10.64	64	13:24:30.837365	TCP	Src= 4096,Dst=65501,.A....,S= 479268135,L= 0,A= 5001,W=17520
5	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.862601	TCP	Src=65501,Dst= 4096,.A....,S= 5001,L= 1460,A= 479268136,W= 4380
6	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.864890	TCP	Src=65501,Dst= 4096,.A....,S= 6461,L= 1460,A= 479268136,W= 4380
7	IP-10.10.10.101	IP-10.10.10.64	64	13:24:30.864988	TCP	Src= 4096,Dst=65501,.A....,S= 479268136,L= 0,A= 7921,W=17520
8	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.869669	TCP	Src=65501,Dst= 4096,.A....,S= 7921,L= 1460,A= 479268136,W= 4380
9	IP-10.10.10.101	IP-10.10.10.64	64	13:24:30.869778	TCP	Src= 4096,Dst=65501,.A....,S= 479268136,L= 0,A= 9381,W=17520
10	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.871977	TCP	Src=65501,Dst= 4096,.A....,S= 9381,L= 1460,A= 479268136,W= 4380
11	IP-10.10.10.101	IP-10.10.10.64	64	13:24:30.874228	TCP	Src=65501,Dst= 4096,.A....,S= 10841,L= 1460,A= 479268136,W= 4380
12	IP-10.10.10.64	IP-10.10.10.101	64	13:24:30.874373	TCP	Src= 4096,Dst=65501,.A....,S= 479268136,L= 0,A= 12301,W=17520
13	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.877391	TCP	Src=65501,Dst= 4096,.AP....,S= 12301,L= 1460,A= 479268136,W= 4380
14	IP-10.10.10.101	IP-10.10.10.64	1518	13:24:30.879678	TCP	Src=65501,Dst= 4096,.A....,S= 13761,L= 1460,A= 479268136,W= 4380
15	IP-10.10.10.101	IP-10.10.10.64	64	13:24:30.879767	TCP	Src= 4096,Dst=65501,.A....,S= 479268136,L= 0,A= 15221,W=17520
16	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.882784	TCP	Src=65501,Dst= 4096,.A....,S= 15221,L= 1460,A= 479268136,W= 4380
17	IP-10.10.10.64	IP-10.10.10.101	1518	13:24:30.885061	TCP	Src=65501,Dst= 4096,.A....,S= 16681,L= 1460,A= 479268136,W= 4380

F4-2

4.03 TCP Receive Test

Target

To start the TTCP TCP Receive Test start μ C/TTCP with the "-r" option.

```
>-r
```

The TCP receiver will start and then wait until a remote TTCP client makes a connection attempt. Here is an example of the output you would see:

```
ttcp-r: BufLen=8192, NumBuf=2048, port=4096 tcp
ttcp-r: Listening socket opened = 9
ttcp-r: Waiting for client to request connection.
ttcp-r: Server socket 8 active
ttcp-r: Client socket 8 closed
ttcp-r: Listen socket 9 closed
ttcp-r: 16777216 bytes in 45.34 real seconds = 2891.06 Kbit/sec +++
ttcp-r: 9093 I/O calls, msec/call = 4.99, calls/sec = 200.56
```

μ C/TCP-IP Performance measurements

Type the TTCP parameters, parameter values and press Enter

```
>
```

These messages are generated as the test starts up:

```
ttcp-r: BufLen=8192, NumBuf=2048, port=4096 tcp
ttcp-r: Listening socket opened = 9
ttcp-r: Waiting for client to request connection.
```

These messages indicate that a TCP connection has been accepted from a remote host:

```
ttcp-r: Server socket 8 active
```

When the test exits it displays the test results:

```
ttcp-r: Client socket 8 closed
ttcp-r: Listen socket 9 closed
ttcp-r: 16777216 bytes in 45.34 real seconds = 2891.06 Kbit/sec +++
ttcp-r: 9093 I/O calls, msec/call = 4.99, calls/sec = 200.56
```

As you can see all the default parameters are used:

- TCP is the default Layer 4 protocol used.
- Sinkmode is enabled by default
- Number of buffers : 2048
- Length of buffers : 8192
- TCP port : 4096 (Note: public domain TTCP uses default port 5001. But recent firewall and virus protection software filters port 5001. We have selected port 4096 for μ C/TTCP. If you encounter problems with port 4096, please select another port that your network setup will not block.)

Windows host TTCP session

The Windows host running pcattcp must initiate the Transmitter part of this test. Figure 4-1 shows the command used:

```

C:\pcattcp>pcattcp -t -p 4096 -f k 10.10.10.64
PCAUSA Test TCP Utility U2.01.01.08
TCP Transmit Test
  Transmit   : TCP -> 10.10.10.64:4096
  Buffer Size : 8192; Alignment: 16384/0
  TCP_NODELAY : DISABLED (0)
  Connect    : Connected to 10.10.10.64:4096
  Send Mode  : Send Pattern; Number of Buffers: 2048
  Statistics  : TCP -> 10.10.10.64:4096
16777216 bytes in 45.33 real seconds = 2891.63 Kbit/sec +++
numCalls: 2048; msec/call: 22.66; calls/sec: 45.18

C:\pcattcp>

```

F4-3

Windows host packet capture

Using Etherpeek, here is a screen shot of the Receive test:

Packet	Source	Destination	Size	Absolute Time	Protocol	Summary
1	00:50:04:97:32:E3	Ethernet Broadcast	64	14:15:43.713389	ARP Request	10.10.10.64 = ?
2	00:08:EE:00:65:4F	00:50:04:97:32:E3	64	14:15:43.714206	ARP Response	00:08:EE:00:65:4F = 10.10.10.64
3	IP-10.10.10.101	IP-10.10.10.64	66	14:15:43.714217	TCP	Src= 1150, Dat= 4096, ...S.,S= 3778976363,L= 0,A= 0,W=16384
4	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.715710	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5000,L= 0,A=3778976364,W= 4380
5	IP-10.10.10.101	IP-10.10.10.64	64	14:15:43.715844	TCP	Src= 1150, Dat= 4096, .A.,S.,S= 3778976364,L= 0,A= 5001,W=17520
6	IP-10.10.10.101	IP-10.10.10.64	1518	14:15:43.716238	TCP	Src= 1150, Dat= 4096, .A.,S.,S= 3778976364,L= 1460,A= 5001,W=17520
7	IP-10.10.10.101	IP-10.10.10.64	1518	14:15:43.717151	TCP	Src= 1150, Dat= 4096, .A.,S.,S= 3778977824,L= 1460,A= 5001,W=17520
8	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.723559	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5001,L= 0,A=3778979284,W= 1460
9	IP-10.10.10.101	IP-10.10.10.64	1518	14:15:43.723651	TCP	Src= 1150, Dat= 4096, .A.,S.,S= 3778979284,L= 1460,A= 5001,W=17520
10	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.728175	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5001,L= 0,A=3778980744,W= 0
11	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.730006	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5001,L= 0,A=3778980744,W= 4380
12	IP-10.10.10.101	IP-10.10.10.64	1518	14:15:43.730078	TCP	Src= 1150, Dat= 4096, .A.,S.,S= 3778980744,L= 1460,A= 5001,W=17520
13	IP-10.10.10.101	IP-10.10.10.64	1518	14:15:43.730123	TCP	Src= 1150, Dat= 4096, .A.,S.,S= 3778982204,L= 1460,A= 5001,W=17520
14	IP-10.10.10.101	IP-10.10.10.64	1518	14:15:43.730167	TCP	Src= 1150, Dat= 4096, .AP.,S.,S= 3778983664,L= 1460,A= 5001,W=17520
15	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.737535	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5001,L= 0,A=3778983664,W= 1460
16	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.740699	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5001,L= 0,A=3778985124,W= 0
17	IP-10.10.10.64	IP-10.10.10.101	64	14:15:43.741828	TCP	Src= 4096, Dat= 1150, .A.,S.,S= 5001,L= 0,A=3778985124,W= 4380

F4-4

4.04 UDP Transmit Test

Target

To start the TTCP UDP Transmit Test start μ C/TTCP with the "-t" option followed by the "-u" option followed finally by the dotted IP address of the remote TTCP client.

```
>-t -u 10.10.10.101
```

The UDP transmitter will start. Here is an example of the output you would see:

```
ttcp-t: BufLen=8192, NumBuf=2048, port=4096  udp  -> 10.10.10.101
ttcp-t: Client socket 9 opened
ttcp-t: Client socket 9 connected
ttcp-t: Client socket 9 closed
ttcp-t: 16777216 bytes in 30.06 real seconds = 4361.07 Kbit/sec +++
ttcp-t: 12288 I/O calls, msec/call = 2.45, calls/sec = 408.85
```

μ C/TCP-IP Performance measurements

Type the TTCP parameters, parameter values and press Enter

```
>
```

These messages are generated as the test starts up:

```
ttcp-t: BufLen=8192, NumBuf=2048, port=4096  udp  -> 10.10.10.101
ttcp-t: Client socket 9 opened
```

These messages indicate that a UDP transmission is in progress toward the remote host:

```
ttcp-t: Client socket 9 connected
```

When the test exits it displays the test results:

```
ttcp-t: Client socket 9 closed
ttcp-t: 16777216 bytes in 30.06 real seconds = 4361.07 Kbit/sec +++
ttcp-t: 12288 I/O calls, msec/call = 2.45, calls/sec = 408.85
```

As you can see all the default parameters are used:

- Sinkmode is enabled by default
- Number of buffers : 2048
- Length of buffers : 1432 (Note: because μ C/TCP-IP does not presently support transmission fragmentation, μ C/TTCP limits the buffer size to 1432 bytes. See the limitations section for the explanation of this buffer size choice.)
- UDP port : 4096 (Note: public domain TTCP uses default port 5001. But recent firewall and virus protection software filters port 5001. We have selected port 4096 for μ C/TTCP. If you encounter problems with port 4096, please select another port that your network setup will not block.)

Windows host TTCP session

The Windows host running pcatttcp must initiate the UDP Receiver part of this test. Figure 4-5 shows the command used:

```

C:\pcatttcp>pcatttcp -r -u -p 4096 -f k
PCAUSA Test TCP Utility V2.01.01.08
UDP Receive Test
  Protocol   : UDP
  Port       : 4096
  Buffer Size : 8192; Alignment: 16384/0
  recufrom   : UDP <- 10.10.10.64:65500
  Statistics  : UDP <- 10.10.10.64:65500
16601688 bytes in 30.05 real seconds = 4316.59 Kbit/sec +++
numCalls: 12160; msec/call: 2.53; calls/sec: 404.70

C:\pcatttcp>
  
```

F4-5

Windows host packet capture

Using Etherpeek, here is a screen shot of the UDP Transmit test:

Packet	Source	Destination	Size	Absolute Time	Protocol	Summary
1	00:08:EE:00:65:4F	Ethernet Broadcast	64	19:51:37.381113	ARP Request	10.10.10.101 = ?
2	00:50:04:97:32:E3	00:08:EE:00:65:4F	64	19:51:37.381136	ARP Response	00:50:04:97:32:E3 = 10.10.10.101
3	IP-10.10.10.64	IP-10.10.10.101	64	19:51:37.382679	UDP	Src=65501, Dest= 4096, Len= 4
4	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.385370	UDP	Src=65501, Dest= 4096, Len= 1432
5	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.387981	UDP	Src=65501, Dest= 4096, Len= 1432
6	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.390505	UDP	Src=65501, Dest= 4096, Len= 1432
7	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.393039	UDP	Src=65501, Dest= 4096, Len= 1432
8	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.395601	UDP	Src=65501, Dest= 4096, Len= 1432
9	IP-10.10.10.64	IP-10.10.10.101	1078	19:51:37.397227	UDP	Src=65501, Dest= 4096, Len= 1032
10	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.400094	UDP	Src=65501, Dest= 4096, Len= 1432
11	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.402637	UDP	Src=65501, Dest= 4096, Len= 1432
12	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.405146	UDP	Src=65501, Dest= 4096, Len= 1432
13	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.407738	UDP	Src=65501, Dest= 4096, Len= 1432
14	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.410272	UDP	Src=65501, Dest= 4096, Len= 1432
15	IP-10.10.10.64	IP-10.10.10.101	1078	19:51:37.411925	UDP	Src=65501, Dest= 4096, Len= 1032
16	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.415037	UDP	Src=65501, Dest= 4096, Len= 1432
17	IP-10.10.10.64	IP-10.10.10.101	1478	19:51:37.417336	UDP	Src=65501, Dest= 4096, Len= 1432

F4-6

4.05 UDP Receive Test

Target

To start the TTCP UDP Receive Test start μ C/TTCP with the "-r" option followed by the "-u" option and the "-l" option (see note below).

```
>-r -u -l 1432
```

The UDP receiver will start and then wait until a remote TTCP client makes a connection attempt. Here is an example of the output you would see:

```
ttcp-r: BufLen=1432, NumBuf=2048, port=4096  udp
ttcp-r: UDP socket 9 opened
ttcp-r: Waiting for client to send UDP datagrams.
ttcp-r: Client socket 9 closed
ttcp-r: 1241544 bytes in 2.47 real seconds = 3928.54 Kbit/sec +++
ttcp-r: 867 I/O calls, msec/call = 2.85, calls/sec = 351.15
```

μ C/TCP-IP Performance measurements

Type the TTCP parameters, parameter values and press Enter

>

These messages are generated as the test starts up:

```
ttcp-r: BufLen=1432, NumBuf=2048, port=4096  udp
ttcp-r: UDP socket 9 opened
ttcp-r: Waiting for client to send UDP datagrams.
```

These messages indicate that a UDP connection has received the last signaling packet (less than 4 bytes) from the remote host:

```
ttcp-r: Client socket 9 closed
```

When the test exits it displays the test results:

```
ttcp-r: 1241544 bytes in 2.47 real seconds = 3928.54 Kbit/sec +++
ttcp-r: 867 I/O calls, msec/call = 2.85, calls/sec = 351.15
```

As you can see all the default parameters are used:

- Sinkmode is enabled by default
- Number of buffers : 2048
- Length of buffers : 1432 (Note: The transmission buffer size is fixed as the maximum UDP datagram to allow time for target to retrieve a maximum of data because of the absence of flow control in UDP. See the limitations section for the explanation of this buffer size choice.)
- UDP port : 4096 (Note: public domain TTCP uses default port 5001. But recent firewall and virus protection software filters port 5001. We have selected port 4096 for μ C/TTCP. If you encounter problems with port 4096, please select another port that your network setup will not block.)

Windows host TTCP session

The Windows host running pcatcp must initiate the UDP Transmitter part of this test. Figure 4-7 shows the command used:

```

C:\pcatcp>pcatcp -t -u -l 1432 -p 4096 -f k 10.10.10.64
PCAUSA Test TCP Utility V2.01.01.08
UDP Transmit Test
  Transmit   : UDP -> 10.10.10.64:4096
  Buffer Size : 1432; Alignment: 16384/0
  Send Mode  : Send Pattern; Number of Buffers: 2048
  Statistics  : UDP -> 10.10.10.64:4096
2932736 bytes in 2.45 real seconds = 9340.40 Kbit/sec +
numCalls: 2050; msec/call: 1.23; calls/sec: 835.71

C:\pcatcp>

```

F4-7

Windows host packet capture

Using Etherpeek, here is a screen shot of the UDP Receive test:

Packet	Source	Destination	Size	Absolute Time	Protocol	Summary
1	IP-10.10.10.101	IP-10.10.10.64	64	09:00:38.464968	UDP	Src= 1829, Dat= 4096, L= 4
2	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.465302	UDP	Src= 1829, Dat= 4096, L= 1432
3	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.466268	UDP	Src= 1829, Dat= 4096, L= 1432
4	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.467285	UDP	Src= 1829, Dat= 4096, L= 1432
5	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.468606	UDP	Src= 1829, Dat= 4096, L= 1432
6	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.469765	UDP	Src= 1829, Dat= 4096, L= 1432
7	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.470914	UDP	Src= 1829, Dat= 4096, L= 1432
8	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.472107	UDP	Src= 1829, Dat= 4096, L= 1432
9	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.473372	UDP	Src= 1829, Dat= 4096, L= 1432
10	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.474473	UDP	Src= 1829, Dat= 4096, L= 1432
11	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.475661	UDP	Src= 1829, Dat= 4096, L= 1432
12	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.476861	UDP	Src= 1829, Dat= 4096, L= 1432
13	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.478063	UDP	Src= 1829, Dat= 4096, L= 1432
14	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.479257	UDP	Src= 1829, Dat= 4096, L= 1432
15	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.480452	UDP	Src= 1829, Dat= 4096, L= 1432
16	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.481671	UDP	Src= 1829, Dat= 4096, L= 1432
17	IP-10.10.10.101	IP-10.10.10.64	1478	09:00:38.482880	UDP	Src= 1829, Dat= 4096, L= 1432

5.00 **μC/TTCP**

The TTCP code is found in the following directory and will be briefly described:

```
\Micrium\Software\uC-TTCP\Source
```

The files are:

```
ttcp.c  
ttcp.h
```

5.01 **TTCP Code, ttcp.c**

This file contains the TTCP code. `ttcp.c` is written to exercise the capabilities of the TTCP testing application. The code begins by initializing μ C/OS-II, μ C/TCP-IP and the serial port used for user interface. It also creates a few tasks and other kernel objects that will inform you about the state of the system.

5.02 TTCP startup code : main()

Listing 5-1, main()

```
int main (void)
{
#if (OS_TASK_NAME_SIZE >= 16)
    CPU_INT08U err;
#endif

    BSP_Init();                /* Initialize BSP.      (1)          */

    APP_TRACE_DEBUG("Initialize OS...\n");
    OSInit();                  /* Initialize OS.      */

                                /* Create start task.  */
    OSTaskCreateExt( AppTaskStart,
                    (void *)0,
                    (OS_STK *)&AppStartTaskStk[APP_START_OS_CFG_TASK_STK_SIZE - 1],
                    APP_START_OS_CFG_TASK_PRIO,
                    APP_START_OS_CFG_TASK_PRIO,
                    (OS_STK *)&AppStartTaskStk[0],
                    APP_START_OS_CFG_TASK_STK_SIZE,
                    (void *)0,
                    OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR);

                                /* Give a name to tasks. */
#if (OS_TASK_NAME_SIZE >= 16)
    OSTaskNameSet(OS_TASK_IDLE_PRIO, "Idle task", &err);
    OSTaskNameSet(OS_TASK_STAT_PRIO, "Stat task", &err);
    OSTaskNameSet(APP_START_OS_CFG_TASK_PRIO, "Start task", &err);
#endif

    APP_TRACE_DEBUG("Start OS...\n");
    OSStart();                 /* Start OS.          */
}
```

L5-1(1) Initialize the on-board I/Os: Interrupts, Timers, LEDs, and the serial port used for the command line interface.

5.03 TTCP startup code : AppStartTask()

Note that some lines of the listings have been removed to help you focus on the μ C/TTCP usage.

Listing 5-2, AppStartTask

```
static void AppTaskStart (void *p_arg)
{
    (void)p_arg;                /* Prevent compiler warning. */

    APP_TRACE_DEBUG("Initialize interrupt controller...\n");
    BSP_InitIntCtrl();          /* Initialize interrupt controller. */

    APP_TRACE_DEBUG("Initialize OS timer...\n");
    Tmr_Init();                 /* Initialize OS timer. */

    APP_TRACE_DEBUG("Initialize OS statistic task...\n");
    OSStatInit();               /* Initialize OS statistic task. (1) */

    AppInit_TCPIP();            /* Initialize TCP/IP stack. (2) */

    AppInit_DHCPc();            /* Initialize DHCP client (if present). (3) */

    APP_TRACE_DEBUG("Create application task...\n");
    AppTaskCreate();            /* Create application task. (4) */

    APP_TRACE_DEBUG("\n*****");
    APP_TRACE_DEBUG("\n* (5) *");
    APP_TRACE_DEBUG("\n* Micrium uC/TCP-IP TTCP Performance measurement *");
    APP_TRACE_DEBUG("\n* AT91RM9200 on Cogent CSB337 SDK *");
    APP_TRACE_DEBUG("\n* *");
    APP_TRACE_DEBUG("\n*****");
    APP_TRACE_DEBUG("\n");
    TTCP_Init();                /* Initialize TTCP application */

    LED_Off(1);                 (6)
    LED_Off(2);
    LED_Off(3);

    while (DEF_YES) {           /* Task body, always written as an infinite loop. */
        OSTimeDlyHMSM(0, 0, 0, 100);
    }
}
```

- L5-2(1) Start the uC/OS-II task responsible to collect OS statistics in case we want to analyse the performance of the application with a Kernel Awareness module or uC/OS-View.
- L5-2(2) Initialization of the μ C/TCP-IP stack.
- L5-2(3) Configuration of the TCP/IP stack using DHCP service. The application could also be modified to use static values.
- L5-2(4) The task #1 toggles LED #3 on the CSB337 board. This activity can tell you that the application is running (i.e. the OS is doing its job).
- L5-2(5) Output the TTCP application banner with the first command line prompt. The application is ready to take user commands.
- L5-2(6) Clear all LED so that their state is known at the beginning of the idle task loop.

6.00 **μC/TTCP module limitations**

Designed for embedded systems in mind, μC/TCP-IP works with the usually resource constrained platforms available. At this stage, μC/TCP-IP does not support some functionality. For example, μC/TCP-IP now only supports a single Network Interface.

μC/TTCP must work with these limitations, mainly for UDP, with the absence of IP transmit fragmentation and the performance of the target versus a PC.

6.01 **IP transmit fragmentation**

The absence of IP transmit fragmentation has an impact on the UDP transmit test. Because μC/TCP-IP can not fragment a packet on transmission the maximum size of UDP datagram is then limited by the maximum size of the Network Interface frame size.

The Maximum UDP datagram is based on the IP packet size to which the UDP header size is removed (8 bytes). The IP packet size is the minimum between the configured Maximum Transmission Unit and the Large buffer size.

In our tests, we have set the MTU to 1500 which is standard for any Ethernet based Network Interface. Our Large Buffer Size was set to 1596 bytes.

From this minimum, we also have to remove the maximum IP header size, which is 60 bytes.

```
NET_IP_MTU      = Minimum(1500, 1596) - IP header size
                  = 1500 - 60
                  = 1440
```

```
NET_UDP_MTU     = IP MTU - UDP header size
                  = 1440 - 8
                  = 1432
```

Depending on your configuration you may want to modify the uC/TTCP code to accommodate your NET_UDP_MTU.

6.02 **Target performance**

Again, this is an issue that affects UDP, particularly the UDP receive test. uC/TCP-IP re-assembles fragmented IP packets. This issue here is how fast can the target retrieve fragmented packets to reassemble them.

As a matter of fact, with a buffer size to 8192 bytes, no data is received by μC/TTCP, the PC being able to transmit a lot faster than the target can receive.

We could find a compromise between 8192 and 1432 bytes.

The sample we provided in this user manual (section 4.05) used a buffer size of 1432 bytes. Each UDP datagram transmitted by the PC fills one buffer on our target. No re-assembly is required.

References

μC/OS-II, The Real-Time Kernel, 2nd Edition

Jean J. Labrosse
R&D Technical Books, 2002
ISBN 1-57820-103-9

Embedded Systems Building Blocks

Jean J. Labrosse
R&D Technical Books, 2000
ISBN 0-87930-604-1

Contacts

IAR Systems

Century Plaza
1065 E. Hillside Blvd
Foster City, CA 94404
USA
+1 650 287 4250
+1 650 287 4253 (FAX)
e-mail: Info@IAR.com
WEB : www.IAR.com

Micrium, Inc.

949 Crestview Circle
Weston, FL 33327
USA
+1 954 217 2036
+1 954 217 2037 (FAX)
e-mail: Christian.Legare@Micrium.com
WEB: www.Micrium.com

CMP Books, Inc.

1601 W. 23rd St., Suite 200
Lawrence, KS 66046-9950
USA
+1 785 841 1631
+1 785 841 2624 (FAX)
e-mail: rushorders@cmpbooks.com
WEB: <http://www.cmpbooks.com>

Validated Software

Lafayette Business Park
2590 Trailridge Drive East, Suite 102
Lafayette, CO 80026
USA
+1 303 531 5290
+1 720 890 4700 (FAX)
e-mail: Sales@ValidatedSoftware.com
WEB: www.ValidatedSoftware.com