CC32xx TCP Socket Application

Overview

A CC3200 device can communicate over network using standard communication protocols such as TCP and UDP. This can be accomplished without even using a Real Time Operating System (RTOS).

Application details

This particular application illustrates how this device can be used as a client or server for TCP communication. Developers/users can refer the function or re-use them while writing new application. The device will connect to an AP (access point), with SSID for AP stored as a macro in the application. Initially, the application implements a TCP client and sends 1000 TCP packets to a socket address, port number and ip address specified as macros. Zero will be the expected return code.

A different return code would mean that socket error has occurred.

Default setting is defined as in following MACROs, which can be changed either in source code or at runtime.

```
'#define SSID_NAME        "cc3200demo"
#define IP_ADDR          0xc0a80167
#define PORT_NUM         5001
#define TCP_PACKET_COUNT    1000'
```

Source Files briefly explained

- **main.c** – main file calls simplelink APIs to connect to the network, create socket and use it to communicate over TCP by acting as a TCP client or server.
- **pinmux.c** - pinmux file to mux the device to configure UART peripheral
- **startup_ccs.c** – CCS specific vector table implementation for interrupts.
- **startup_ewarm.c** – IAR workbench specific vector table implementation for interrupts.

Code flow

Connection

```
void main()
{
...
lRetVal = sl_Start(0, 0, 0);
...
// Connecting to WLAN AP – Set with static parameters defined at common.h
// After this call we will be connected and have IP address
lRetVal = WlanConnect();
...
/* following calls depend on user's input at runtime */
// Before proceeding, please make sure to have a server waiting on PORT_NUM
```

```
BsdTcpClient(PORT_NUM);

// After calling this function, you can start sending data to CC3200 device IP
// address on PORT_NUM
BsdTcpServer(PORT_NUM);
...
}
```

TCP Client

```
int BsdTcpClient(unsigned short usPort)
{
...
//Open a socket with standard parameters
iSockID = sl_Socket(SL_AF_INET,SL_SOCK_STREAM, 0);
if( iSockID < 0 )
{
// error
ASSERT_ON_ERROR(TCP_CLIENT_FAILED);
}

//Connect to the server IP and port number
iStatus = sl_Connect(iSockID, ( SlSockAddr_t *)&sAddr, iAddrSize);
if( iStatus < 0 )
{
// error
ASSERT_ON_ERROR(sl_Close(iSockID));
ASSERT_ON_ERROR(TCP_CLIENT_FAILED);
}
...
// sending packet
iStatus = sl_Send(iSockID, g_cBsdBuf, sTestBufLen, 0 );
if( iStatus <= 0 )
{
// error
ASSERT_ON_ERROR(sl_Close(iSockID));
ASSERT_ON_ERROR(TCP_CLIENT_FAILED);
}
...
//closing the socket after sending 1000 packets
ASSERT_ON_ERROR(sl_Close(iSockID));

return SUCCESS;
}
```

Sending the TCP Packets is a simple four step process

1. Open the socket
2. Connect to the server

3. Send the packets
4. Close the socket

TCP Server

```c
int BsdTcpServer(unsigned short usPort)
{
...
iSockID = sl_Socket(SL_AF_INET,SL_SOCK_STREAM, 0);
if( iSockID < 0 )
{
// error
ASSERT_ON_ERROR(TCP_SERVER_FAILED);
}


...
iStatus = sl_Bind(iSockID, (SlSockAddr_t *)&sLocalAddr, iAddrSize);
if( iStatus < 0 )
{
// error
ASSERT_ON_ERROR(sl_Close(iSockID));
ASSERT_ON_ERROR(TCP_SERVER_FAILED);
}


iStatus = sl_Listen(iSockID, 0);
if( iStatus < 0 )
{
ASSERT_ON_ERROR(sl_Close(iSockID));
ASSERT_ON_ERROR(TCP_SERVER_FAILED);
}


iStatus = sl_SetSockOpt(iSockID, SL_SOL_SOCKET, SL_SO_NONBLOCKING, &lNonBlocking, sizeof(lNonBlocking));
if( iStatus < 0 )
{
ASSERT_ON_ERROR(sl_Close(iSockID));
ASSERT_ON_ERROR(TCP_SERVER_FAILED);
}
iNewSockID = SL_EAGAIN;


while( iNewSockID < 0 )
{
iNewSockID = sl_Accept(iSockID, ( struct SlSockAddr_t *)&sAddr, (SlSocklen_t*)&iAddrSize);
if( iNewSockID == SL_EAGAIN )
{
UtilsDelay(10000);
}
else if( iNewSockID < 0 )
{
// error
```

```
ASSERT_ON_ERROR(sl_Close(iNewSockID));

ASSERT_ON_ERROR(sl_Close(iSockID));

ASSERT_ON_ERROR(TCP_SERVER_FAILED);

}

}


iStatus = sl_Recv(iNewSockID, g_cBsdBuf, iTestBufLen, 0);

if( iStatus <= 0 )

{

// error

ASSERT_ON_ERROR( sl_Close(iNewSockID));

ASSERT_ON_ERROR(sl_Close(iSockID));

ASSERT_ON_ERROR(TCP_SERVER_FAILED);

}

...

ASSERT_ON_ERROR(sl_Close(iNewSockID));

ASSERT_ON_ERROR(sl_Close(iSockID));


return SUCCESSS;

}
```

Steps for receiving TCP Packets from TCP client are as follows

1. Open the socket
2. Create a TCP server
3. listen for connection
4. accept a connection
5. receive packets
6. Close the socket

Usage

- Setup a serial communication application (HyperTerminal/TeraTerm). For detail info visit Terminal setup

  Note: Disable PC anti-virus while running iperf.

  On the host PC. The settings are:
  - **Port**: Enumerated COM port (CC3200LP Dual port)
  - **Baud rate**: 115200
  - **Data**: 8 bit
  - **Parity**: None
  - **Stop**: 1 bit
  - **Flow control**: None
- Run this application (tcp_socket) from IAR/CCS or Flash the bin file to device.
- Connect a PC to same AP over which device has connected.
- Get the ip address of the PC and fill this value for IP_ADDR macro or change the setting as specified in snapshot-
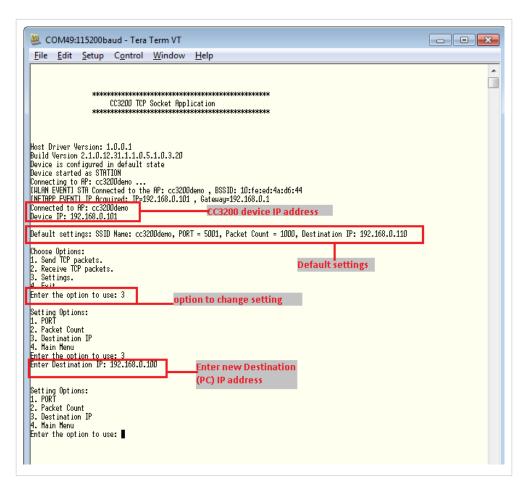
- Change the other setting (port, SSID name, packet count) as per requirement.
- Choose the options

    - 1: Send TCP packets

    - 2: Receive TCP packets

after selecting above options run **iperf** command on PC command prompt as given in TeraTerm/HyperTerminal screen.

- Observe the execution flow to understand the working.

Limitations/Known Issues

None.

Article Sources and Contributors

CC32xx TCP Socket Application *Source*: http://processors.wiki.ti.com/index.php?oldid=184856 *Contributors*: A0221015, Beatrice, Codycooke, Jitgupta, Malokyle

Image Sources, Licenses and Contributors

File:Cc31xx cc32xx return home.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Cc31xx_cc32xx_return_home.png *License*: unknown *Contributors*: A0221015

File:Cc32xx return sample apps.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Cc32xx_return_sample_apps.png *License*: unknown *Contributors*: A0221015

File:Light_bulb_icon.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Light_bulb_icon.png *License*: unknown *Contributors*: DanRinkes, PagePusher

Image:CC32xx Tcp Socket Terminal runScreen 1.0.0.png *Source*: http://processors.wiki.ti.com/index.php?title=File:CC32xx_Tcp_Socket_Terminal_runScreen_1.0.0.png *License*: unknown *Contributors*: Jitgupta

License

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.