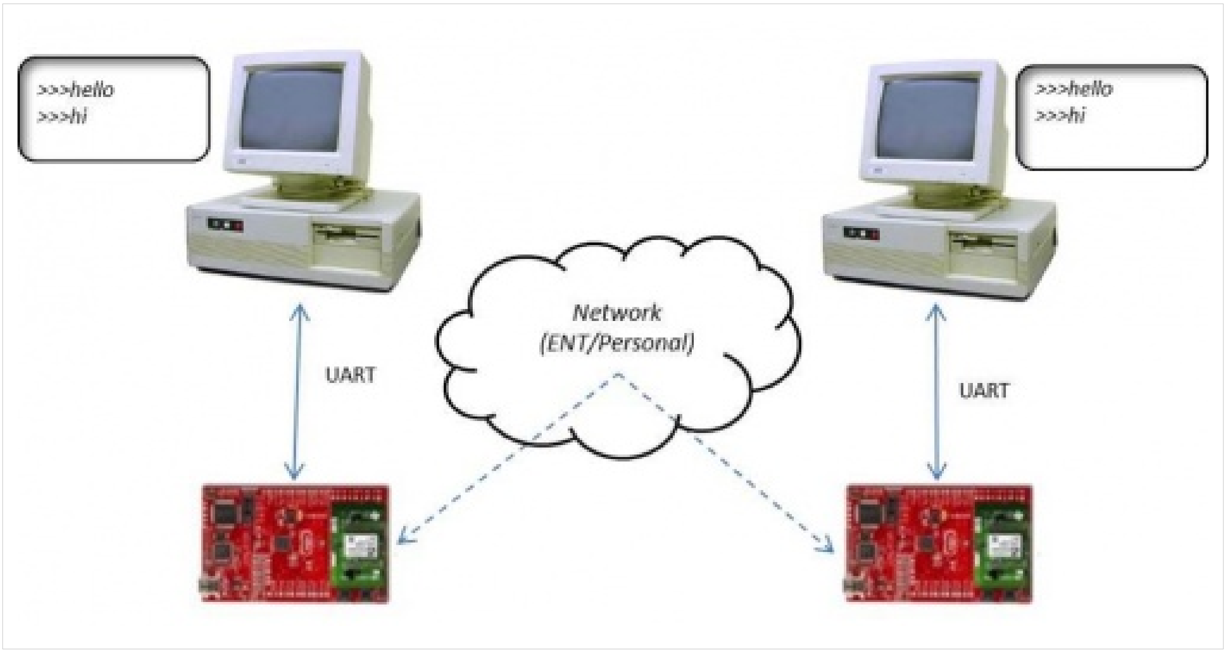
CC32xx Serial Wifi

Overview

Return to CC31xx & CC32xx Home Page

Return to CC31xx Sample Applications

Serial WiFi is a capability designed to provide easy, self-contained terminal access behavior, over UART interface. This provides a 'driver-less' solution and allows out-of-the box operation based on ASCII character set interpretation. This application runs over the UART interface and behaves as a command line interpreter. Leveraging the complete network stack integration it allows secure, robust end-to-end communication.



Application details

The serial WiFi application supports the following features:

1. Support to connect to **ENT/Personal network**
2. Auto discovery of available peer using **mDNS**
3. All communication is secure using the **Secure Sockets**
4. Once configured to a network, **auto connect** to the AP on every boot
5. Provision to **override default mnemonic strings**
6. **Local/Remote Control modes**, to issue commands to the locally/remotely connected simplelink device.

The Serial WiFi application bacsically operates in 3 modes:

1. **Terminal/Interpreter mode:**
In this case the interpreter mode behaves as pure point-to-point cable replacement. This is the most common and obvious use in which a serial cable formerly used to carry information is replaced by SimpleLink devices on both end of the line.
2. **Local control mode:**
In this mode it is possible to issue commands to the locally connected device. These commands encompass a close set of options to control the SimpleLink device while guaranteeing sandbox restrictions. The mode is entered using a predefined escape sequence (**//<**)

3. **Remote control mode:**
This allows issuing commands to the remotely connected device. These commands encompass the same set of options to control the remote SimpleLink as the local device, while guaranteeing sandbox restrictions. The mode is entered using a predefined escape sequence(**//>**)

Caution: Current time must be set in the device. This time is used to validate the certificate. If the date is beyond the validity period of Certificate, sl_connect will return error. Please update the below macros in the serial_wifi.h file to change date.

```
#define DATE                18    /* Current Date */
#define MONTH               6     /* Month 1-12 */
#define YEAR                2014  /* Current year */
#define HOUR                12    /* Time - hours */
#define MINUTE              32    /* Time - minutes */
#define SECOND              0     /* Time - seconds */
```

Source Files briefly explained

main.c - creates to 2 tasks, ***Task 1:*** the interpreter task to opertate the serial wifi functionality, ***Task 2:*** The UART task to receive data from the UART.
serial_wifi.c - ALl the functionality of the serial wifi is implemented.
uart_config.c - Configures the UART and the task to send and receive chars over the UART.
conversions.c - All conversion utility API into different number formats.

Usage

1. Flash the serial wifi code onto both the devices.
2. Upon Reset, the device connects to the stored AP by the AUTO connect policy.If the device doesn't connect to an AP in 6secs, the application prompts the user to connect to a known AP using the Local control mode. User can use the below command to connect to an AP

```
            wlan_connect [ssid] [type] [Sec] [User] [key] in the local control mode.
```

3. If this is the first boot after an SFALSH erase, one of the devices need to be configured for as mDNS server. THis can be done by using the below command in local control mode.

```
            mDNS  0  <0-server, 1-client> in the local control mode.
```

4. The client device automatically connects to the advertiser.

5. The communication is based on secure sockets. So, the server key, server certificate, client certificate need to be flashed in the SFLASH at file IDs 129,130,131 respectively. Please refer the Generating SSL certificates section for generating certificates using OpenSSL.
For flashing the certificates using UniFlash:(For detailed instructions about using Uniflash, refer Uniflash User Guide.)

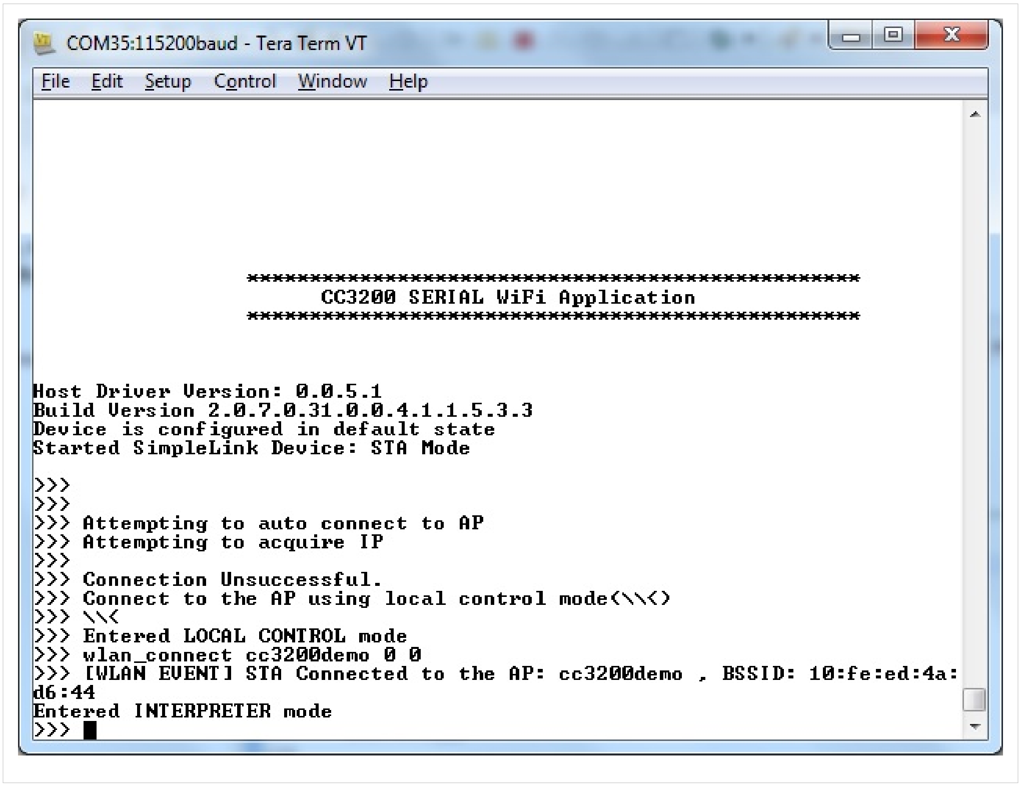
```
    1. Click "add file"


    2. Rename the newly created file to "/cert/129.der"


    3. In the Url field mention the local path to the Server key certificate.


    4. Repeat the above steps for "/cert/130.der" and "/cert/131.der" for server key and client certificate respectively.
    5. Check the Erase and Update check boxes and "program"
```

6. Now the devices are connected in Interpreter mode.
7. '//<' command is used to enter the local control mode, mode in which we control the local CC3200 device.
8. '//>' command is used to enter the remote control mode, mode to control the remote CC3200 device.
9. Commands supported in the control modes are:

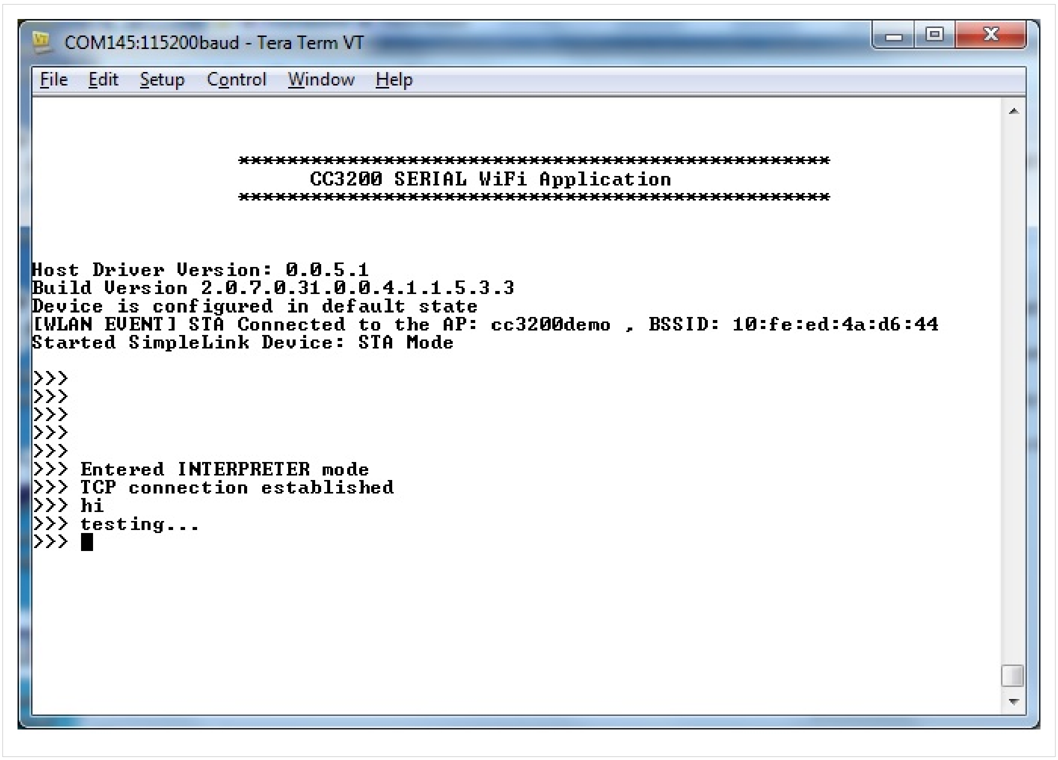
Supported commands

help	Show menu
wlan_connect [ssid] [type] [Sec] [User] [key]	Connect to wlan AP. ssid:Name of the AP type: 0-Personal,1-ENT Sec: 0-OPEN,1-WEP,2-WPA "); Message("\r\n User: user name in case of ENT Key: password in case of WEP and WPA
connect [IP | name] [port] [TCP | UDP]	Connect to specified IP or named server; with an optionally specified port. If not specified, default port is used. Can optionally set transport protocol explicitly. If IP/name not specified use UDP as default transport; otherwise, TCP.
disconnect	Disconnect connected end point.
\\<	Enter local control mode.
\\>	Enter remote control mode.
exit	Terminate control mode, and resume interpreter mode operation.
quit	Terminate interpreter mode, and free up all related resources.
mode [A | B | H]	Set data interpretation mode: A - ASCII; B - Binary; H - Hex
wrap [B | L]	Set terminal wrap mode level: Message("\r\n B - Byte; L - Line.
send [dest] [data]	Send provided data to dest. If TCP is used, dest is ignored.
recv [dest] [length] [timeout]	Receive data from dest. If TCP is used, dest is ignored.If length not specified, 1 is used. Will wait until length bytes received, the optional timeout [us] is reached, or ^C is received.
ping [IP | name] [number]	Ping the provided dest for an optionally specified number of times. If not specified, ping will run until ^C is received.
mDNS [option]	mDNS configuration option: 0-Advertise,1- Listen

10. All communication is secure using the Secure sockets, Security method :SSLV3 and Cipher : RSA WITH RC4 128 SHA.


COM35:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
                 ************************************************
                       CC3200 SERIAL WiFi Application
                 ************************************************


Host Driver Version: 0.0.5.1
Build Version 2.0.7.0.31.0.0.4.1.1.5.3.3
Device is configured in default state
Started SimpleLink Device: STA Mode

>>>
>>>
>>> Attempting to auto connect to AP
>>> Attempting to acquire IP
>>>
>>> Connection Unsuccessful.
>>> Connect to the AP using local control mode(\\<)
>>> \\<
>>> Entered LOCAL CONTROL mode
>>> wlan_connect cc3200demo 0 0
>>> [WLAN EVENT] STA Connected to the AP: cc3200demo , BSSID: 10:fe:ed:4a:
d6:44
Entered INTERPRETER mode
>>>
```


COM145:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
                 ************************************************
                       CC3200 SERIAL WiFi Application
                 ************************************************


Host Driver Version: 0.0.5.1
Build Version 2.0.7.0.31.0.0.4.1.1.5.3.3
Device is configured in default state
[WLAN EVENT] STA Connected to the AP: cc3200demo , BSSID: 10:fe:ed:4a:d6:44
Started SimpleLink Device: STA Mode

>>>
>>>
>>>
>>>
>>>
>>> Entered INTERPRETER mode
>>> TCP connection established
>>> hi
>>> testing...
>>>
```

Generating certificates using OpenSSL

The following are the steps for generating certificates using OpenSSL tool.
Install OpenSSl and run the following commands:

Generate a CA

1) openssl req -out ca.pem -new -x509
-generates CA file "ca.pem" and CA key "privkey.pem"

Generate server certificate/key pair

2) openssl genrsa -out server.key 1024
3) openssl req -key server.key -new -out server.req
4) openssl x509 -req -in server.req -CA CA.pem -CAkey privkey.pem -CAserial file.srl -out server.pem
-contents of "file.srl" is a two digit number. eg. "00"
5) Convert the certificates and key generated to DER format.

```
        For certificates

        openssl x509 -in <input.crt> -inform PEM —out <output.crt> -outform DER

        For key

        rsa -in input.key -inform PEM -out output.key -outform DER
```

6) Flash the server key and the server certificate at /cert/129.der and /cert/130.der respectively.
7) Flash the CA certificate at /cert/131.der.

Limitations/Known Issues

None.

Article Sources and Contributors

CC32xx Serial Wifi *Source*: http://processors.wiki.ti.com/index.php?oldid=184786 *Contributors*: A0221015, 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

Image:Serial wifi1.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Serial_wifi1.jpg *License*: unknown *Contributors*: Codycooke

Image:CC32xx Serial Wifi SDK 0p5p2 1.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:CC32xx_Serial_Wifi_SDK_0p5p2_1.jpg *License*: unknown *Contributors*: Jitgupta

Image:CC32xx Serial Wifi SDK 0p5p2 2.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:CC32xx_Serial_Wifi_SDK_0p5p2_2.jpg *License*: unknown *Contributors*: Jitgupta