

CC3200 SimpleLink™ Wi-Fi® and IoT Solution With MCU LaunchPad™ Getting Started Guide

User's Guide



Literature Number: SWRU376A
June 2014–Revised September 2014

1	Introduction.....	4
1.1	Prerequisites	4
2	Getting Started	5
2.1	Download and Install Software	5
2.2	Configure Board	5
2.3	Update Service Pack	6
3	Compile, Download, and Debug	7
3.1	Option 1: Code Composer Studio (CCS)	7
3.2	Option 2: IAR Workbench	16
3.3	Option 3: GCC	22
4	Summary	29
5	Acronyms Used.....	30
	Revision History	31

List of Figures

1	Jumpers on the CC3200-LAUNCHXL.....	5
2	Device Manager.....	6
3	Code Composer Studio v6 Setup.....	7
4	CCS App Center	8
5	Select CCS Projects to Import	9
6	Select CCS Projects to Import.....	10
7	Properties for <i>ti_rtos_config</i>	11
8	Select <i>simplelink</i> Project	12
9	Editing <i>common.h</i>	12
10	Target Configurations.....	13
11	Import Target Configuration.....	13
12	Set as Default.....	14
13	Launch Tera Term	14
14	Tera Term Serial Port Setup.....	15
15	Debug <i>wlan_station</i>	15
16	Tera Term VT	16
17	IAR Embedded Workbench IDE.....	17
18	Open <i>simplelink.eww</i>	18
19	Rebuild the <i>simplelink</i> Project.	19
20	Editing <i>common.h</i>	20
21	Select TI Stellaris Driver.....	20
22	Launch Tera Term	21
23	Tera Term Serial Port Setup.....	21
24	Debug Icon	22
25	Tera Term VT	22
26	Cygwin Setup	23
27	Cygwin Terminal.....	24
28	Cygwin Terminal.....	24
29	Running the Make Command.....	24
30	Running the Make Install Command.....	25
31	Editing <i>common.h</i>	25
32	Makefile Command	25
33	Output Screen	26
34	Launch Tera Term	27
35	Tera Term Serial Port Setup.....	27
36	Debugging <i>wlan_station</i>	28
37	Tera Term VT	28

CC3200 SimpleLink™ Wi-Fi® and IoT Solution With MCU LaunchPad™ Getting Started Guide

This guide is intended to assist users in the initial setup and demonstration of the *Getting Started with WLAN Station* application. The guide explains how to install an Integrated Development Environment (IDE), and then compile, download and debug *Getting Started with WLAN Station*.

1 Introduction

1.1 Prerequisites

The user should have the following items:

- One CC3200-LAUNCHXL
- An 802.11b/g/n (2.4 GHz) Wireless Access Point (AP).
- A computer running the Microsoft® Windows® 7 or XP operating systems.

2 Getting Started

2.1 Download and Install Software

Download and install the following software:

- [CC3200 SDK package](#).
 - This guide assumes the use of the default installation folder C:\TI\CC3200SDK_1.0.0\.
- [Tera Term](#) (or similar software)
 - Tera Term link: <http://en.sourceforge.jp/projects/ttssh2/releases/>

2.2 Configure Board

The jumpers on the CC3200-LAUNCHXL should be connected as shown in [Figure 1](#). It may be necessary to move a jumper from P58-VCC to SOP2.

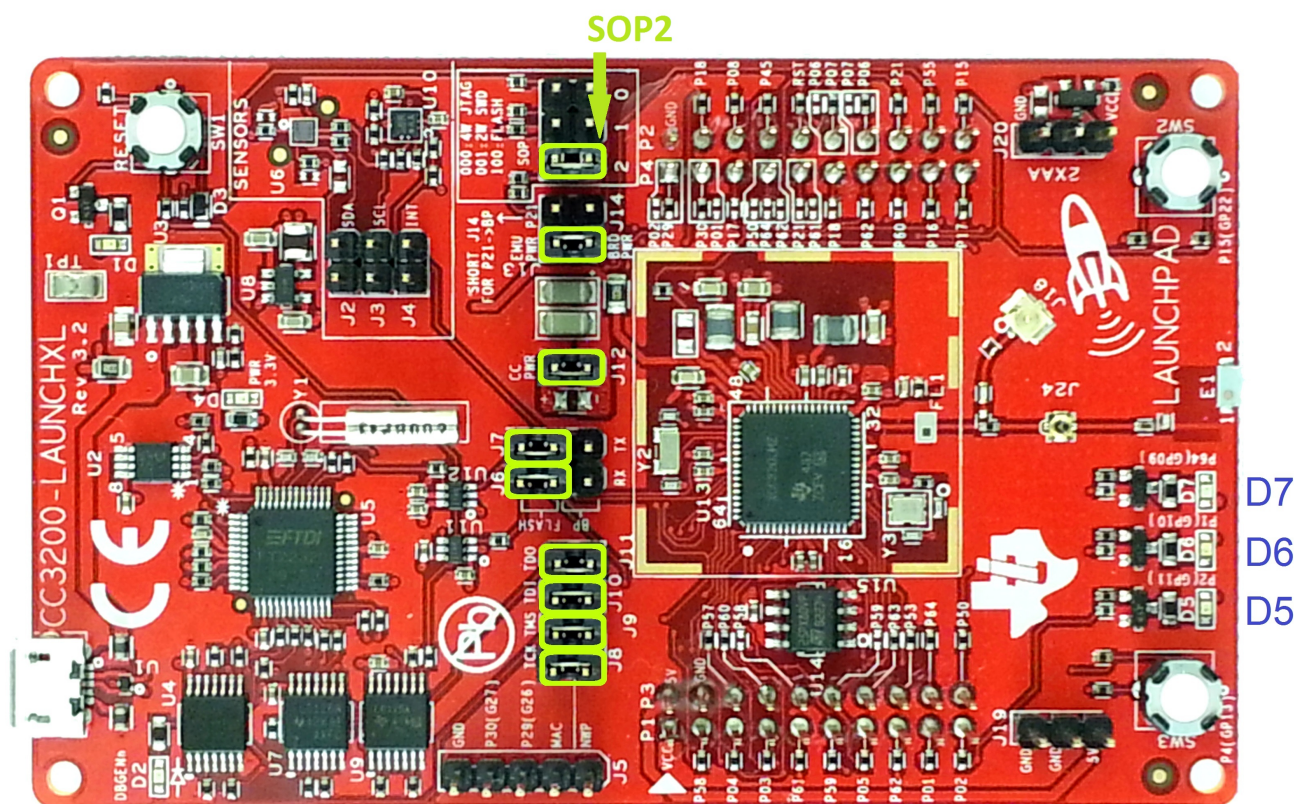


Figure 1. Jumpers on the CC3200-LAUNCHXL

1. Connect the CC3200-LAUNCHXL to the PC using the provided micro-USB cable.
2. The CC3200-LAUNCHXL will now be visible in the Device Manager as shown in [Figure 2](#). Note the COM port number that appears.

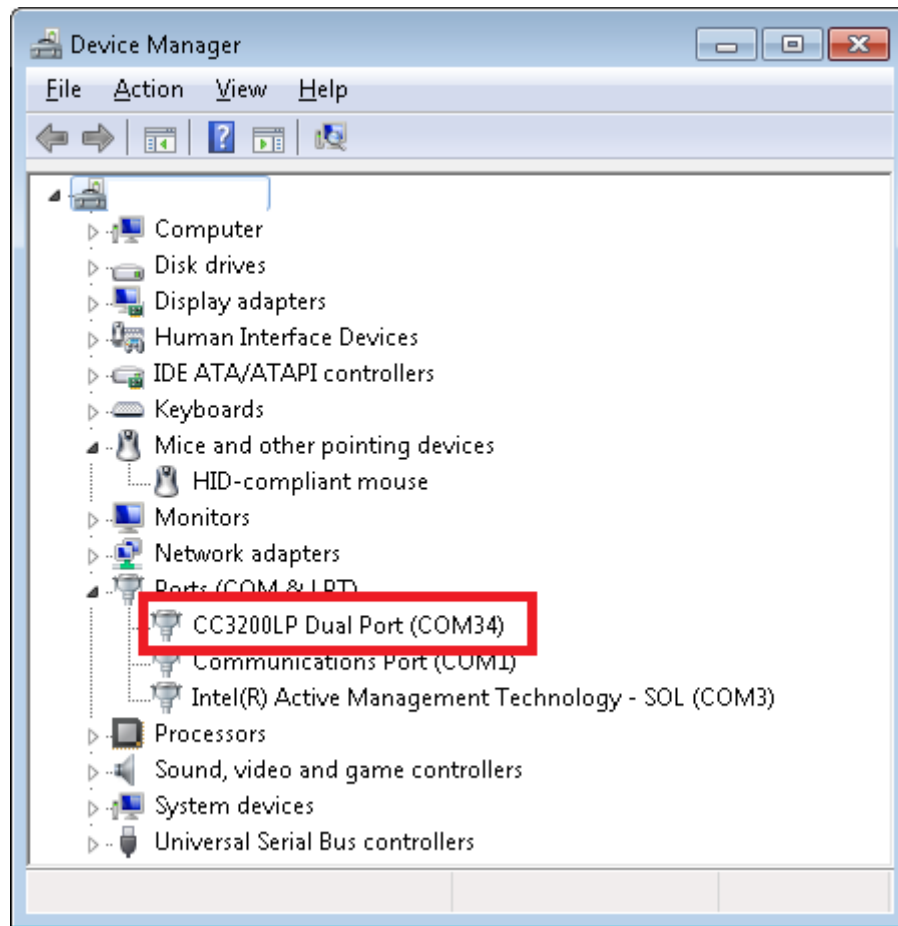


Figure 2. Device Manager

2.3 Update Service Pack

If the board being used is not already flashed with the service pack for SDK 1.0, the latest service pack for SDK 1.0 needs to be flashed on the CC3200. The latest service pack can be downloaded from <http://www.ti.com/tool/cc3200sdk>. Refer to the UNIFLASH Quick start guide for details on flashing the service pack to the CC3200 (http://processors.wiki.ti.com/index.php/CC31xx_%26_CC32xx_UniFlash#Service_Pack_Programming).

3 Compile, Download, and Debug

The CC3200 SDK supports CCS 6.0.1, IAR 7.20 and GCC IDE/compiler. The example shown here is *Getting Started with WLAN Station*, and performs the following functions:

1. Switches to Station mode if the device is in AP mode.
2. Connects to the user's Access Point (default SSID is 'cc3200demo'). If the connection to the AP is successful, the red LED (D7) will switch on.
3. Pings the user's Access Point. If the ping test is successful, the green LED (D5) will switch on.
4. Pings to www.ti.com to check Internet connectivity. If the ping test is successful, the orange LED (D6) switches on.

This example uses a Real Time Operating System (RTOS).

3.1 Option 1: Code Composer Studio (CCS)

3.1.1 Download and Install

Download and run the Code Composer Studio 6.0.1 (CCS) installation wizard (*ccs_setup_win32.exe*) from http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v6. The program must be **Version 6.0.1.00040** or later. Select the Wireless Connectivity MCUs option for processor support. The remaining options for the installer should be left as the default. Installation time is typically 20 minutes, but can vary based on internet connection speed.

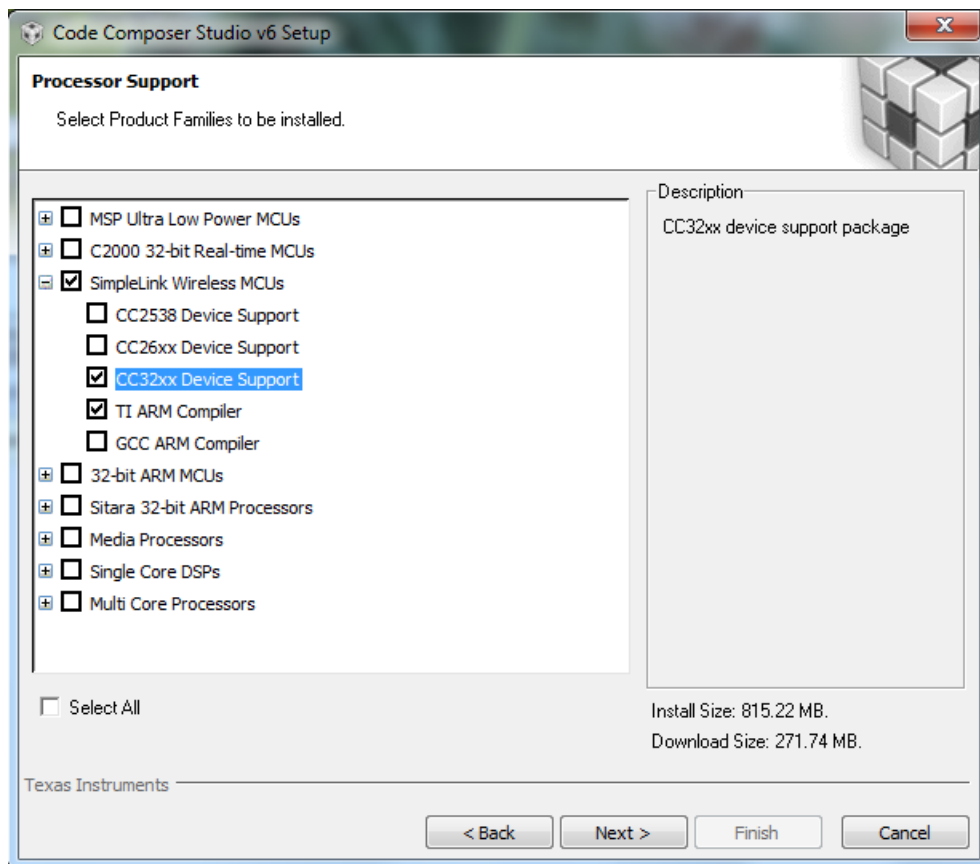


Figure 3. Code Composer Studio v6 Setup

3.1.2 Install TI-RTOS for SimpleLink and CC3200 Support Package

Install TI-RTOS for SimpleLink from the CCS App Center:

1. Start CCS, and choose a Workspace folder (the folder where the projects reside).
2. Open the App Center from the *Help->Getting Started* screen.
3. Search 'CC3200' in the App Center to find 'TI-RTOS for SimpleLink' and 'CC3200 Add-On'
4. Select TI-RTOS
5. The CC3200 Add-On should already be installed. If not, select it.
6. Press 'Install Software'

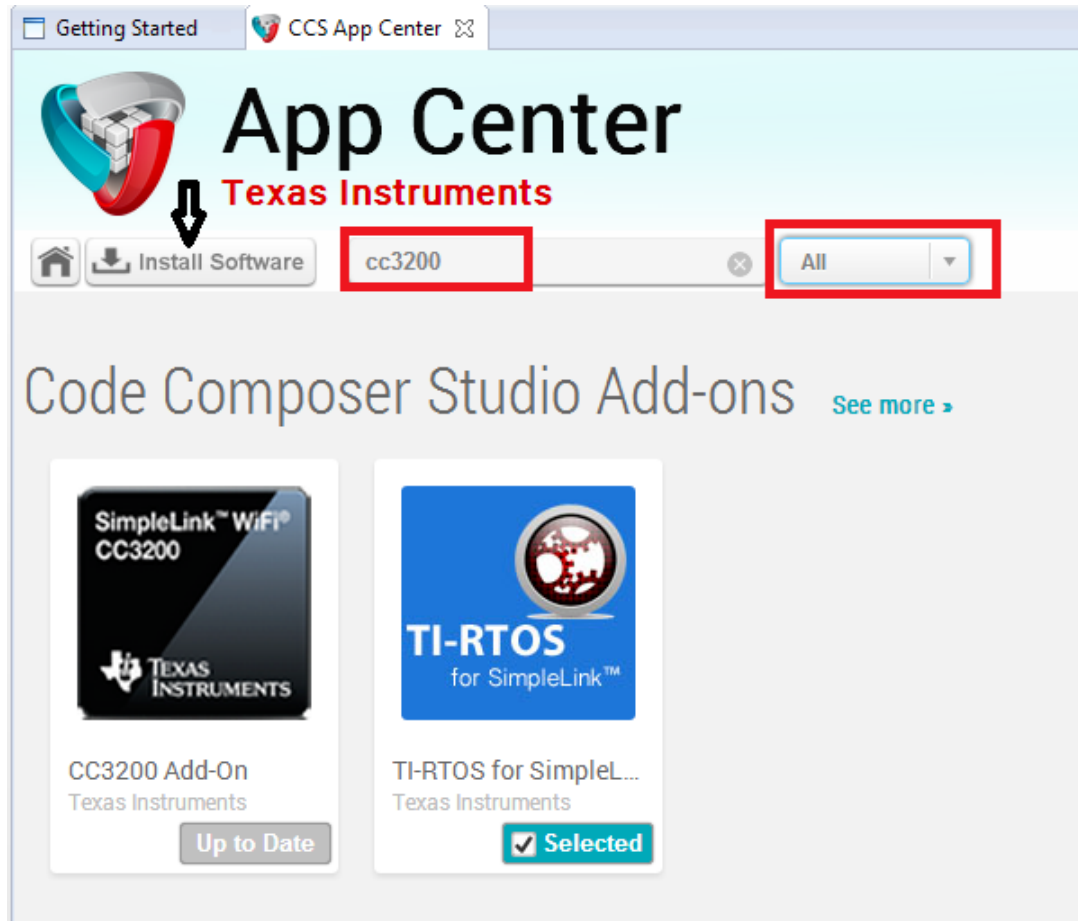


Figure 4. CCS App Center

3.1.3 Import and Configure Project

1. Choose *Project>Import CCS Projects* from the menu.
2. Select the Browse button in the Import CCS Eclipse Projects dialog, and Select the directory `C:\TI\CC3200SDK_1.0.0\cc3200-sdk`.

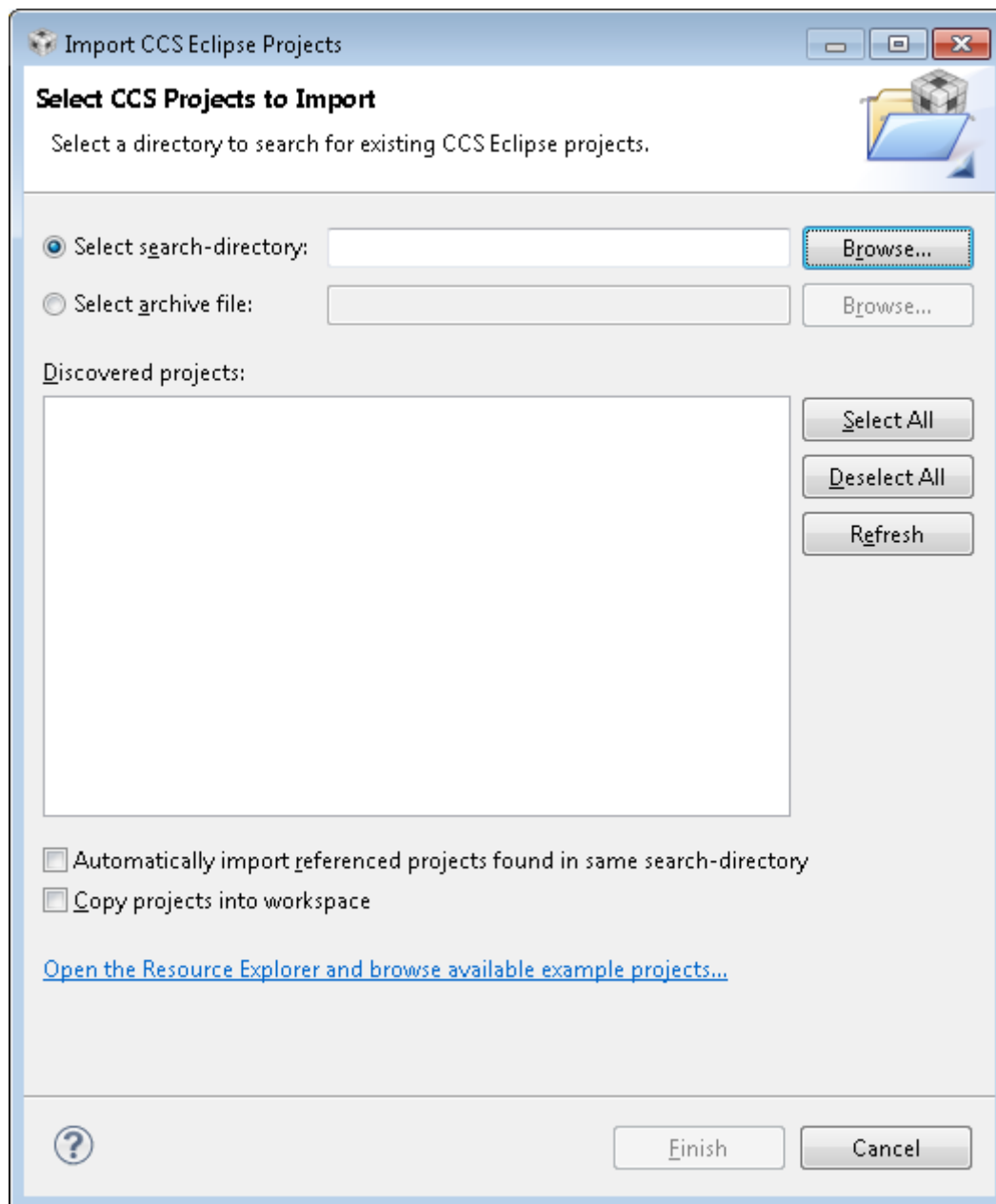


Figure 5. Select CCS Projects to Import

3. Select the *wlan_station*, *driverlib*, *simplelink*, *oslib* and *ti_rtos_config* projects. Click Finish. For any library import, do not check the 'Copy projects into workspace' option. This would break the links the libraries have to their dependencies. The *wlan_station* project will automatically be copied to the workspace.

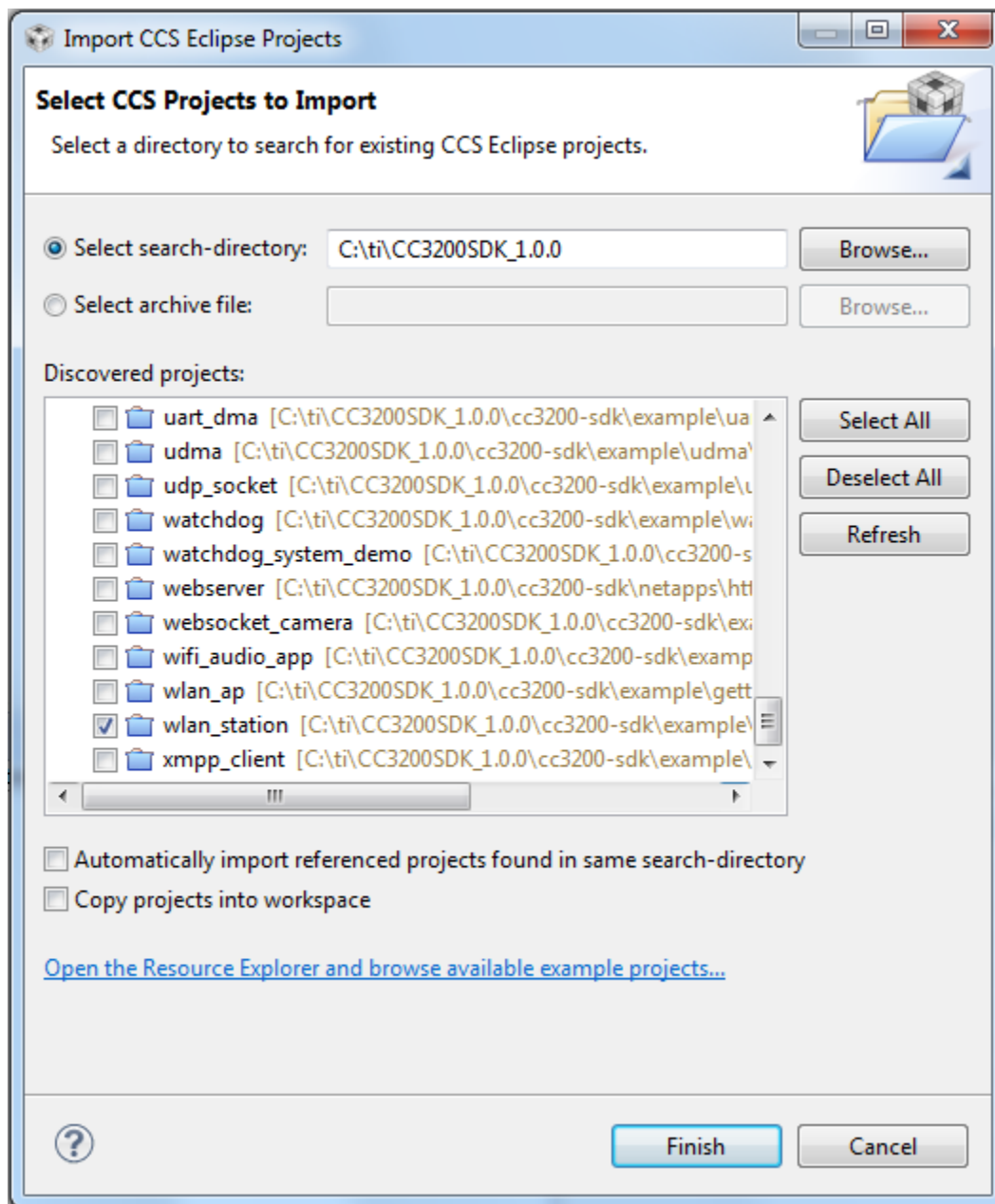


Figure 6. Select CCS Projects to Import

4. Select the *ti_rtos_config* project in Project Explorer, and select *Project>Properties* from the menu. Setup the *ti_rtos_config* project configuration as shown in [Figure 7](#). Select the latest versions of XDCtools and TI-RTOS for SimpleLink. Also verify the platform is selected as *ti.platforms.simplelink:CC3200*.

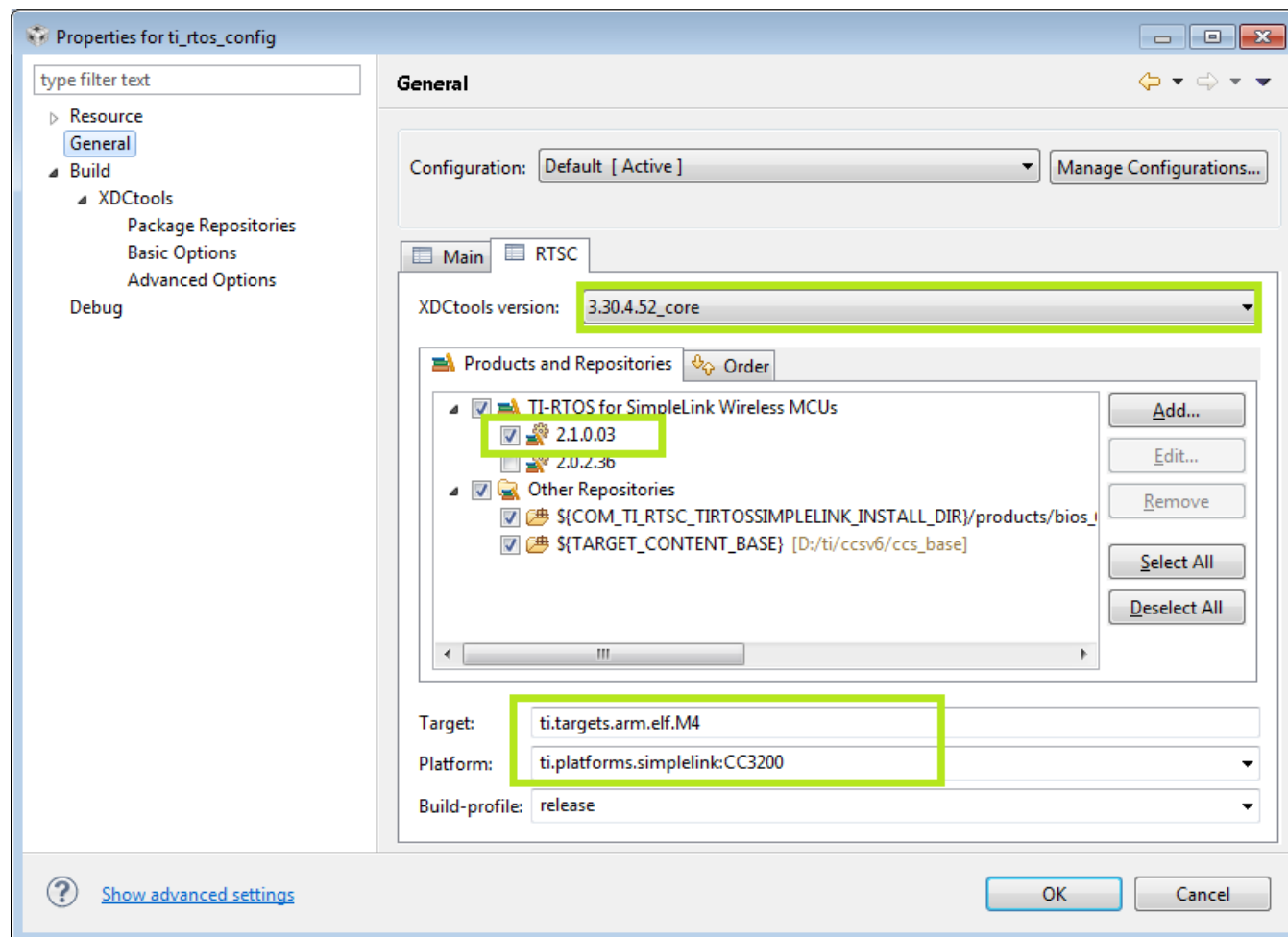


Figure 7. Properties for *ti_rtos_config*

5. Select the *simplelink* project and build it as shown in [Figure 8](#).

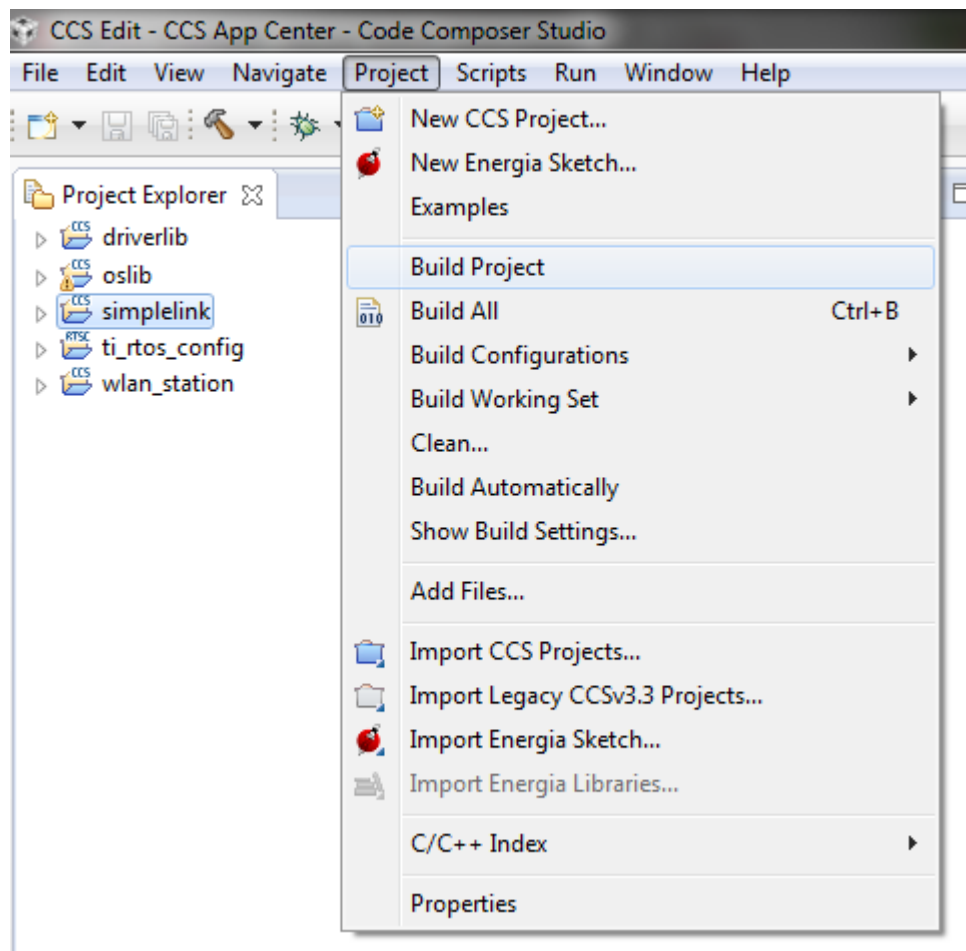


Figure 8. Select *simplelink* Project

6. Select the *ti_rtos_config* project and build it.
7. Select the *driverlib* project and build it.
8. Select the *oslib* project and build it.
9. Open the *common.h* file located at the path *C:\TI\CC3200SDK_1.0.0\cc3200-sdk\example\common*.
10. Edit *common.h* to use the SSID, security type and security key of the Access Point being used. Edit the macros *SSID_NAME*, *SECURITY_TYPE* and *SECURITY_KEY* to contain the Access Point's information as shown in Figure 9. The security types supported for this demo are WPA/WPA2 and Open. For Open security, define *SECURITY_TYPE* as *SL_SEC_TYPE_OPEN*. For WPA and WPA2 security, define it as *SL_SEC_TYPE_WPA*. Alternatively, the SSID and security of the Access Point being used can be changed to match the default (SSID: *cc3200demo*, Security: Open).

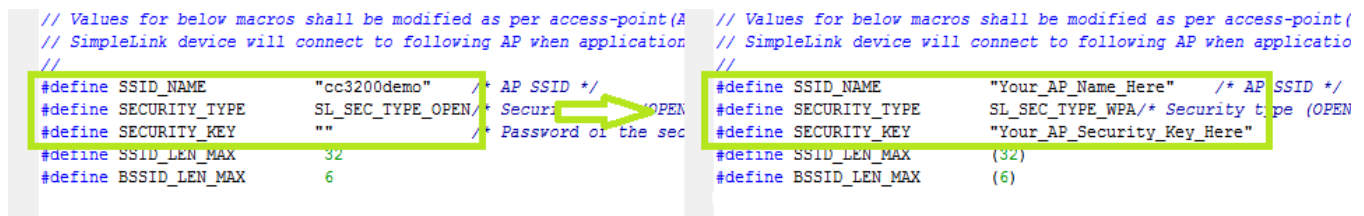


Figure 9. Editing *common.h*

11. Save *common.h*.
12. Select the *wlan_station* project and build it.

13. The target configuration needs to be set before debugging from CCS. Navigate to *View>Target Configurations*.

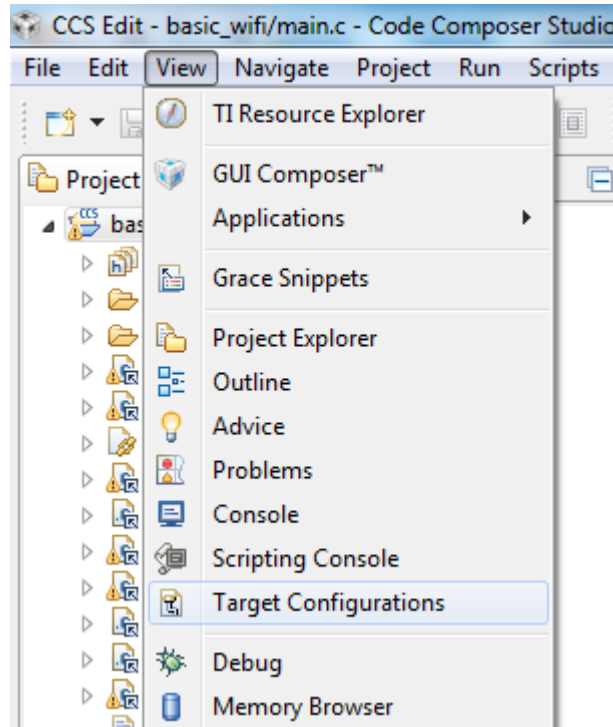


Figure 10. Target Configurations

14. Right Click on "User Defined," select "Import Target Configuration" and select the file CC3200.ccxml from C:\TI\CC3200SDK_1.0.0\cc3200-sdk\tools\lccs_patch\. Select the Copy files option when prompted.

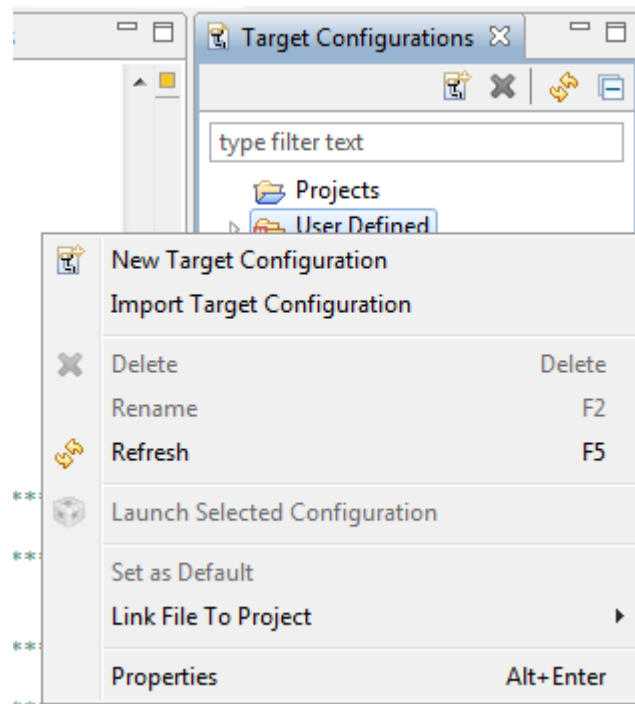


Figure 11. Import Target Configuration

15. Set this new configuration as the default by right clicking on the file name as shown in [Figure 12](#).

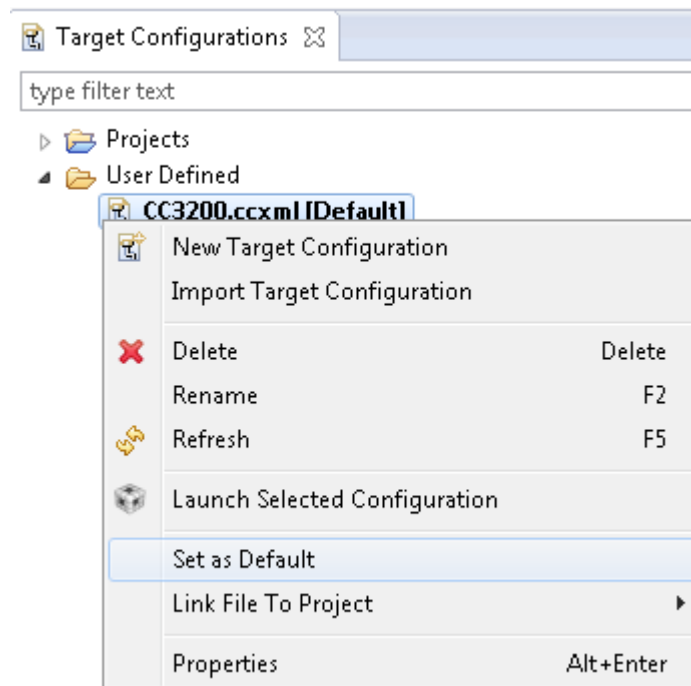


Figure 12. Set as Default

Caution: Only one FTDI board should be connected to the PC while CCS downloads code to device.

16. Launch Tera Term, and create a new serial connection to the CC3200 Launchpad COM port as shown in [Figure 13](#).

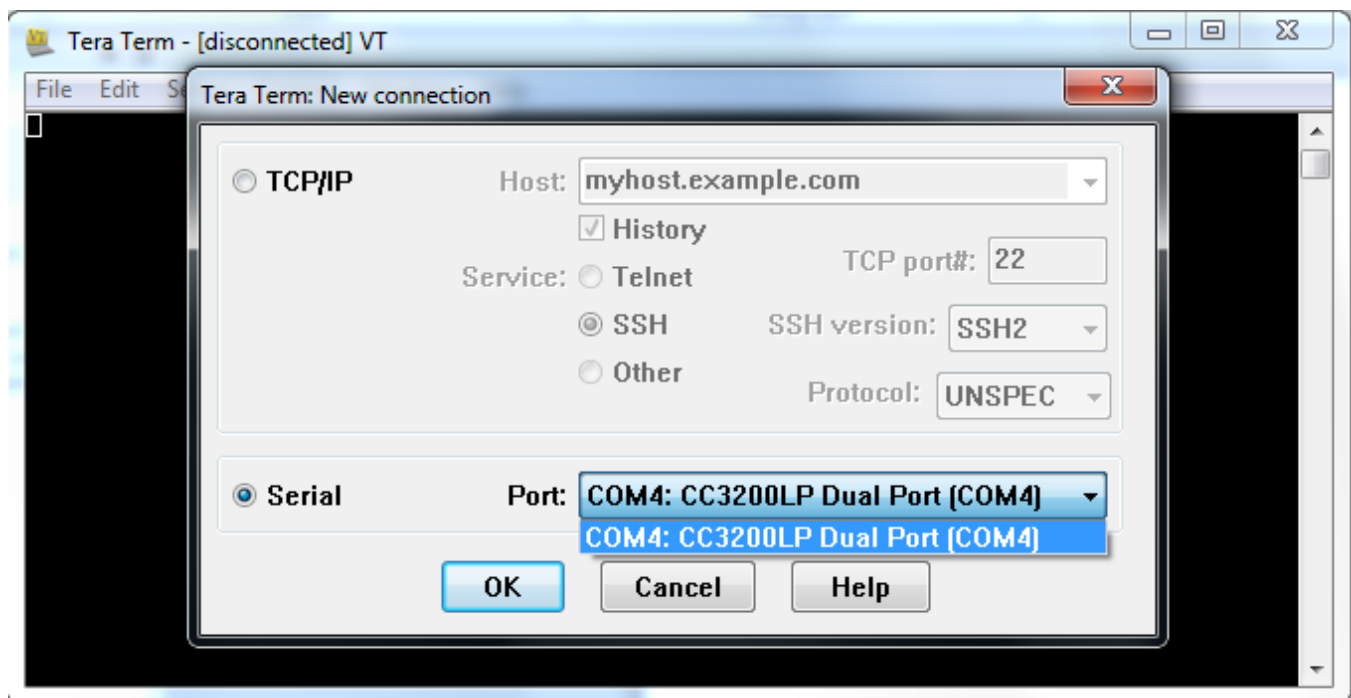


Figure 13. Launch Tera Term

17. In the menu, select Setup>Serial Port, and change the baud rate to 115200 as shown in [Figure 14](#).

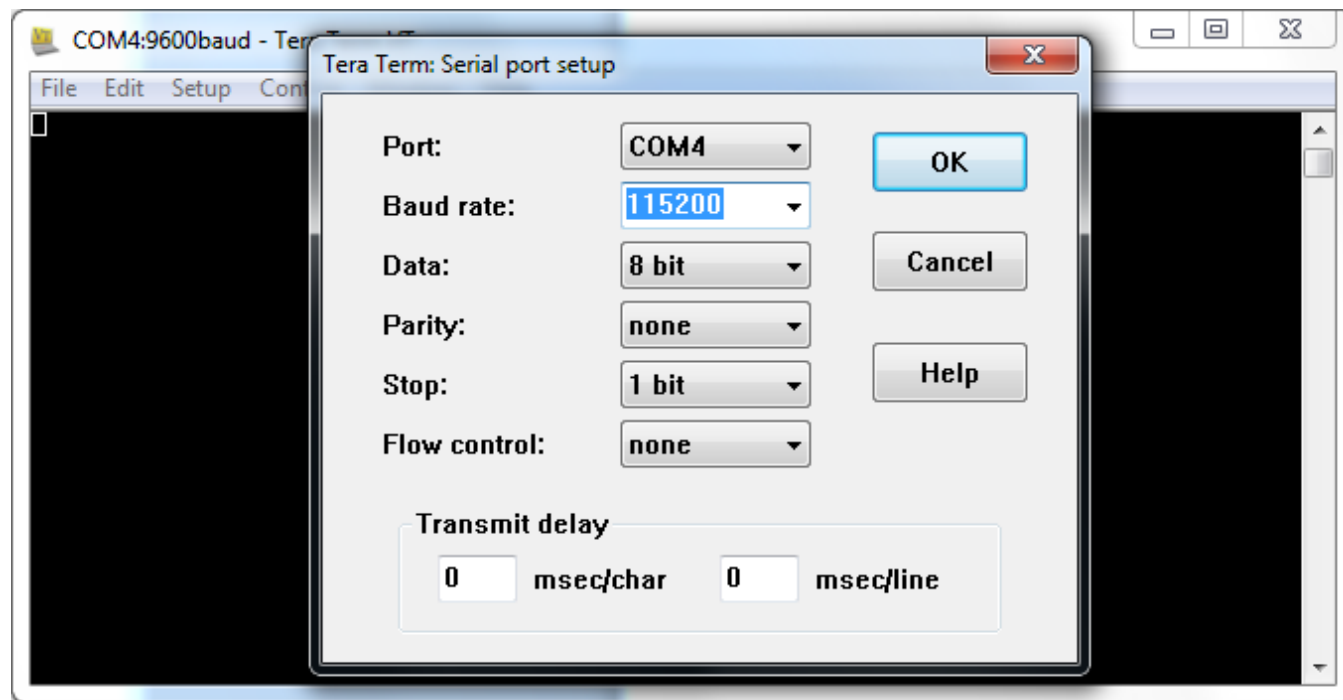


Figure 14. Tera Term Serial Port Setup

18. Launch application. Select the `wlan_station` project in Project Explorer, then click the debug icon as shown in Figure 15 to download code to the device and begin debugging. Press F8 to begin execution.

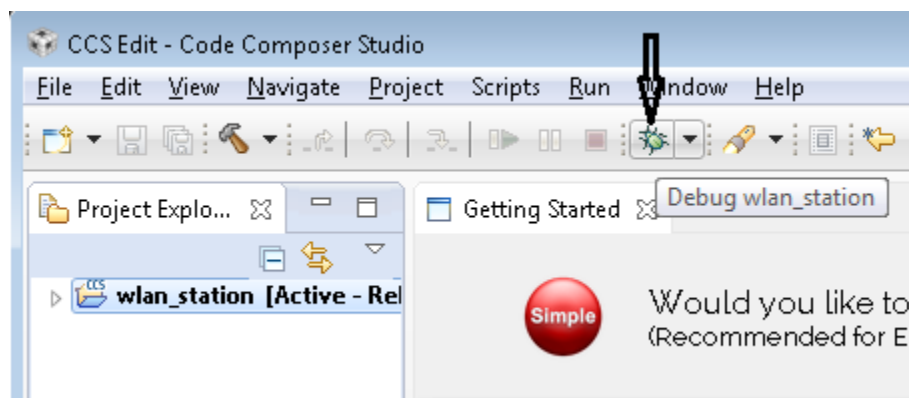


Figure 15. Debug `wlan_station`

19. If the CC3200 successfully completes all steps, the serial output appears as shown in Figure 16.

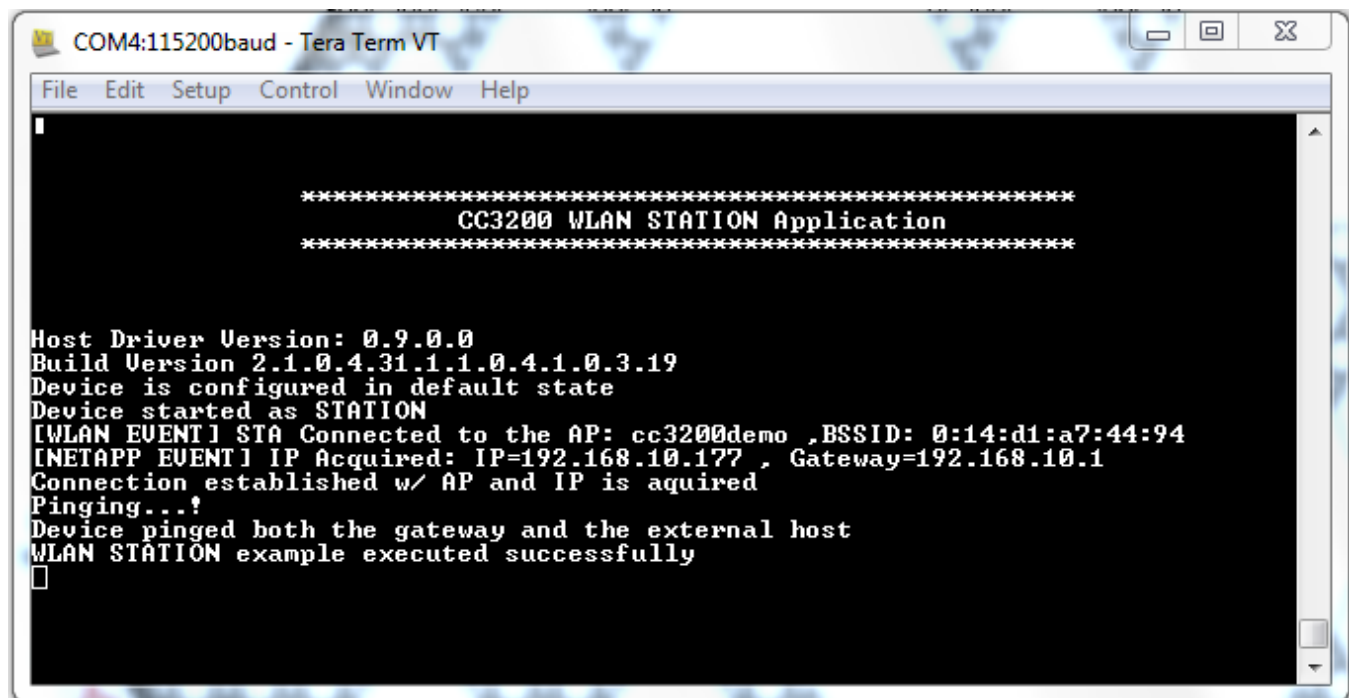


Figure 16. Tera Term VT

3.2 Option 2: IAR Workbench

3.2.1 Download IAR

The CC3200 SDK has been built and tested with IAR 7.20, and older versions of IAR projects might not work properly on IAR 7.20. Most examples will only run with the fully licensed IAR Workbench.

1. Download IAR for ARM processors from the IAR System website, and install it using the installation wizard.
2. Copy the file C:\TI\CC3200SDK_1.0.0\CC3200-sdk\tools\iar_patch\armLMIFTDI.dll into the folder C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.0\arm\bin (will need to replace existing file).

3.2.2 Rebuild the SimpleLink Driver

1. Start IAR and select *File>Open>Workspace* from the menu.

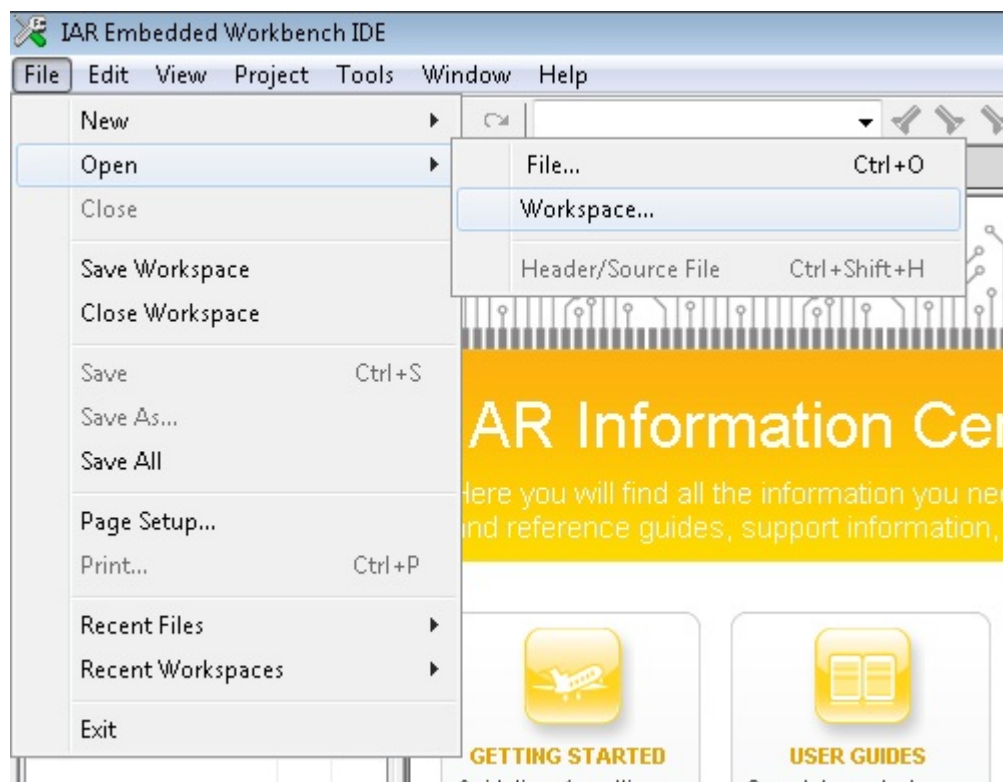


Figure 17. IAR Embedded Workbench IDE

2. Open the *simplelink* project by navigating to `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\simplelink\ewarm` and opening *simplelink.eww*.

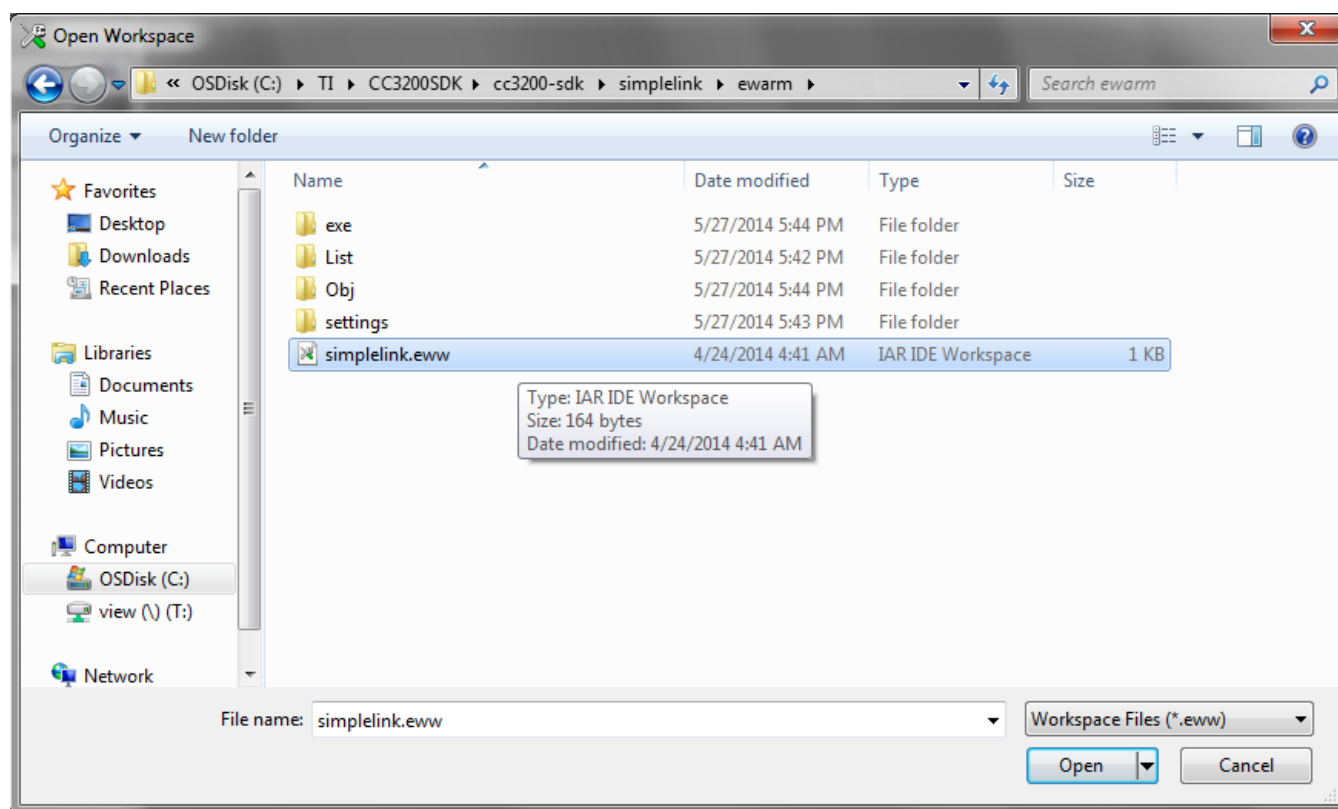


Figure 18. Open *simplelink.eww*

- Rebuild the *simplelink* project by selecting *Project>Rebuild All* from the menu as shown in [Figure 19](#).

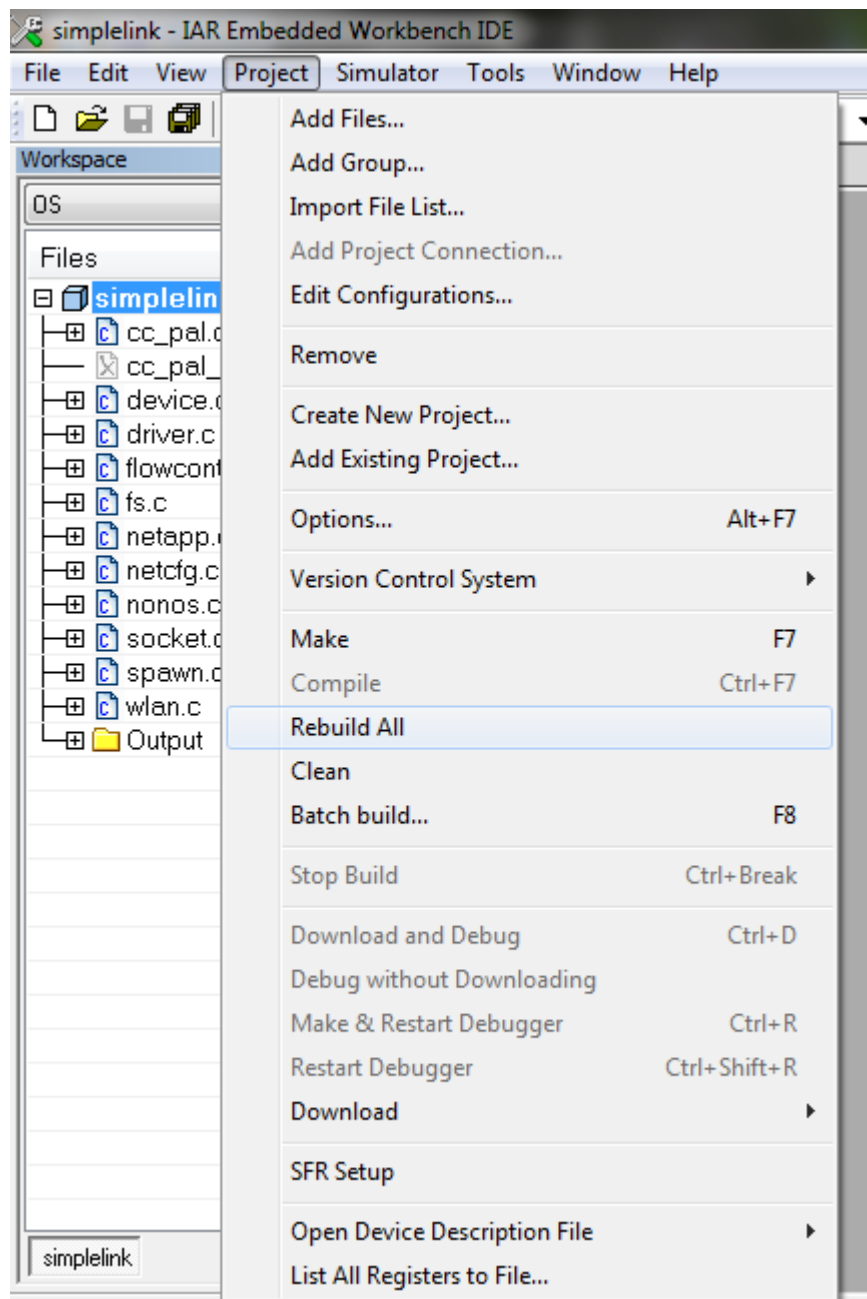


Figure 19. Rebuild the *simplelink* Project.

3.2.3 Rebuild, Download and Debug the WLAN Station Example

1. Open the *wlan_station* project by selecting *File>Open>Workspace* from the menu, navigating to *C:\TI\CC3200SDK_1.0.0\cc3200-sdk\example\getting_started_with_wlan_station\warm*, and opening *wlan_station.eww*.
2. Open the *common.h* file located at the path *C:\TI\CC3200SDK_1.0.0\cc3200-sdk\example\common*.
3. Edit *common.h* to use the SSID, security type and security key of the Access Point being used. Edit the macros *SSID_NAME*, *SECURITY_TYPE* and *SECURITY_KEY* to contain the Access Point's information as shown in Figure 20. The security types supported for this demo are WPA/WPA2 and Open. For Open security, define *SECURITY_TYPE* as *SL_SEC_TYPE_OPEN*. For WPA and WPA2 security, define it as *SL_SEC_TYPE_WPA*.

```
// Values for below macros shall be modified as per access-point(A // Values for below macros shall be modified as per access-point(
// SimpleLink device will connect to following AP when application // SimpleLink device will connect to following AP when applicatio
//
#define SSID_NAME "cc3200demo" /* AP SSID */
#define SECURITY_TYPE SL_SEC_TYPE_OPEN /* Security type (OPEN)
#define SECURITY_KEY "" /* Password of the sec
#define SSID_LEN_MAX 32
#define BSSID_LEN_MAX 6

#define SSID_NAME "Your_AP_Name_Here" /* AP SSID */
#define SECURITY_TYPE SL_SEC_TYPE_WPA /* Security type (OPEN)
#define SECURITY_KEY "Your_AP_Security_Key_Here"
#define SSID_LEN_MAX (32)
#define BSSID_LEN_MAX (6)
```

Figure 20. Editing *common.h*

4. Save *common.h*.
5. Rebuild the *wlan_station* project by selecting *Project>Rebuild All* from the menu.
6. The debugger must be configured to download code to the device. Select *Project>Options* from the menu, and select the Debugger category. In the Setup tab, choose TI Stellaris as the driver, as shown in [Figure 21](#), and press OK.

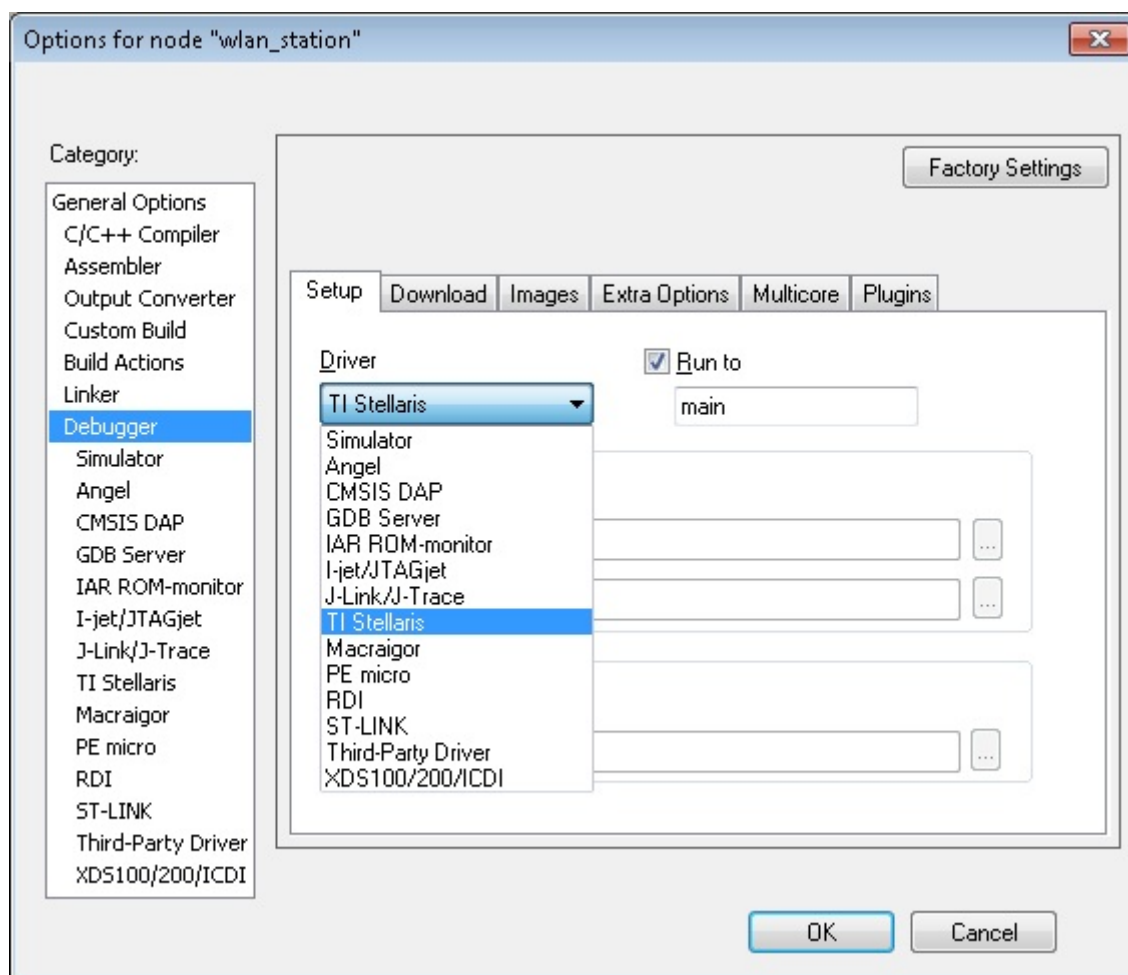


Figure 21. Select TI Stellaris Driver

7. Launch Tera Term, and create a new serial connection to the CC3200 Launchpad COM port as shown in [Figure 22](#).

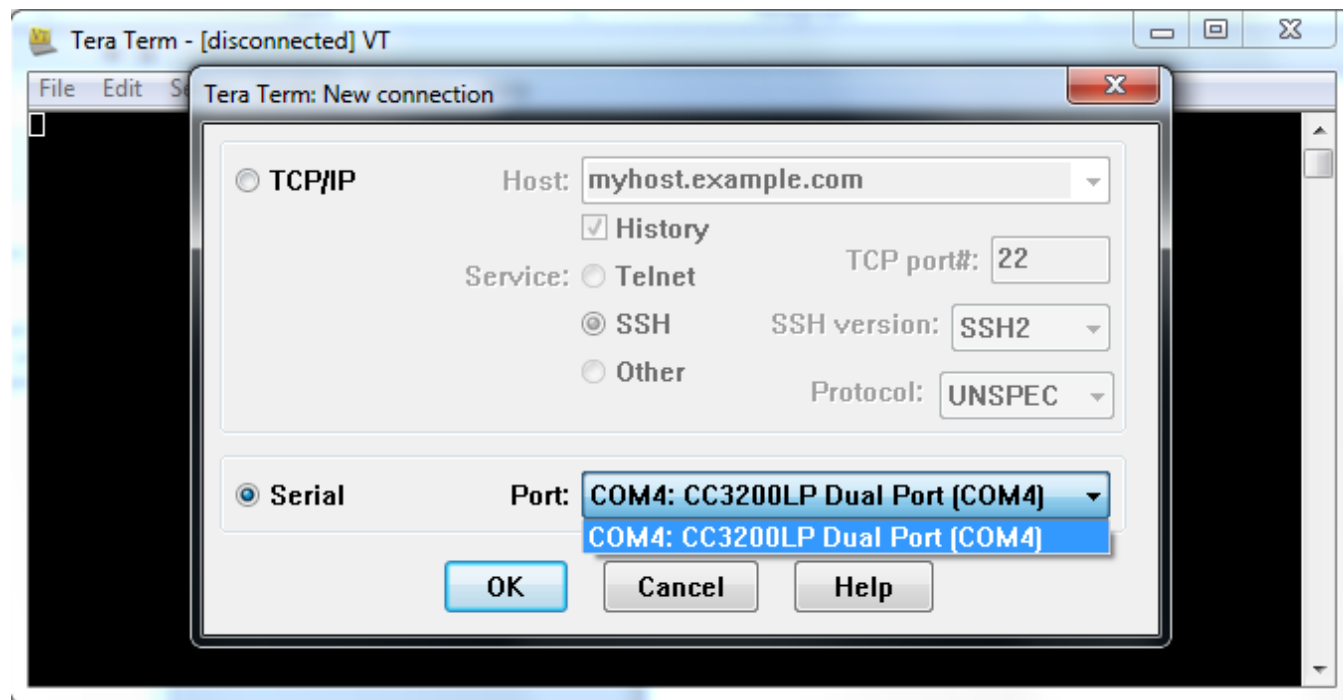


Figure 22. Launch Tera Term

8. In the menu, select Setup>Serial Port, and change the baud rate to 115200 as shown in [Figure 23](#).

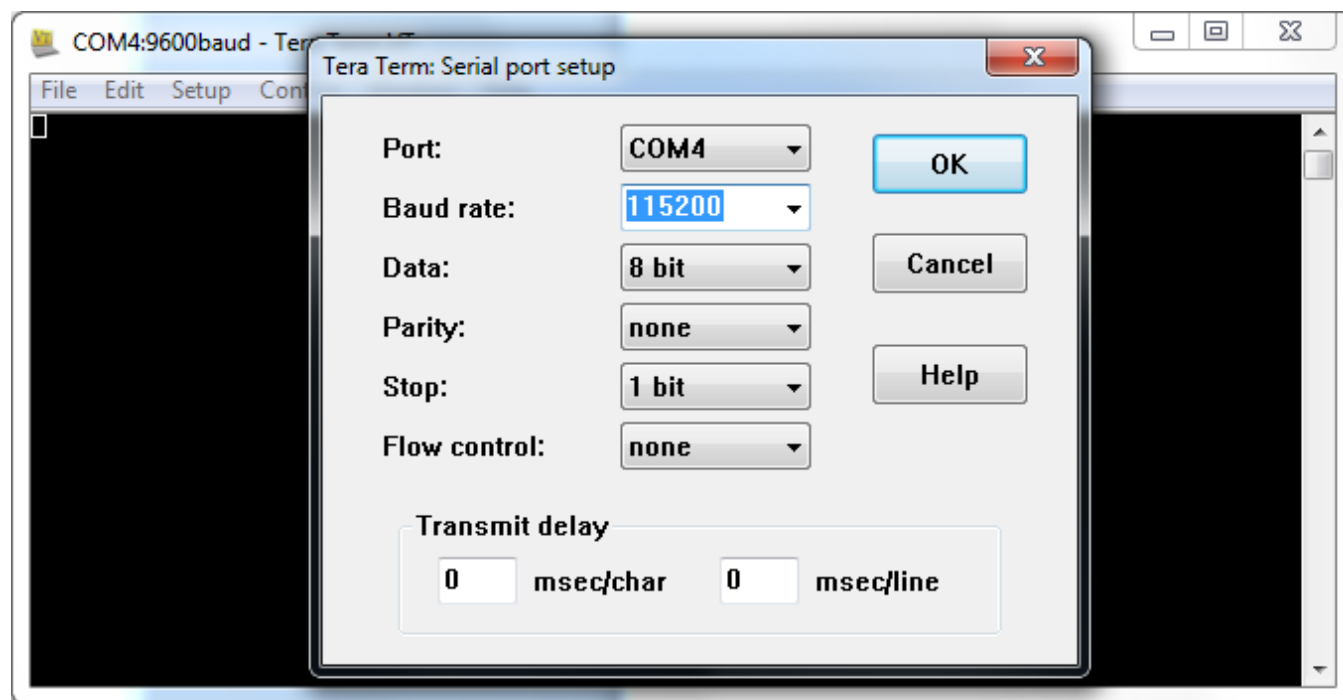
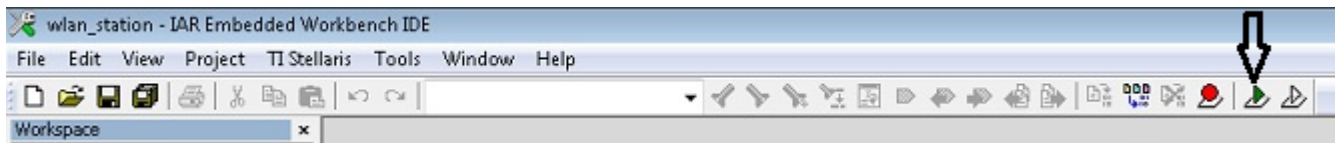
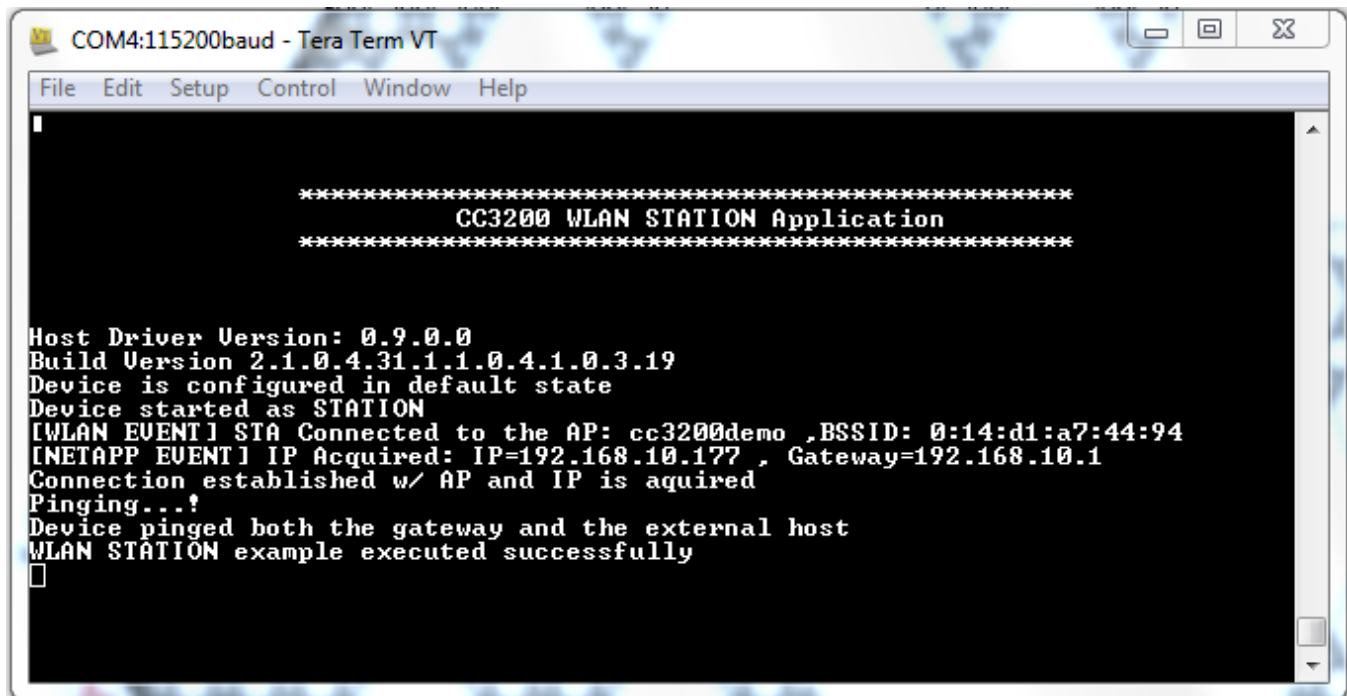


Figure 23. Tera Term Serial Port Setup

9. Click the debug icon as shown in [Figure 24](#) to download code to the device and start debugging. Select *Debug>Go* from the menu or press F5 to begin execution.


Figure 24. Debug Icon

10. If the CC3200 successfully completes all steps, the serial output appears as shown in [Figure 25](#).


Figure 25. Tera Term VT

3.3 Option 3: GCC

This section demonstrates the GCC setup for the Windows 7 environment. GCC installation requires other dependencies to be installed to work with ARM-based devices.

3.3.1 Install Cygwin (Windows)

1. Download *setup-x86.exe* from <http://cygwin.com/install.html> and run it. Select the Install from Internet option.
2. Specify a proxy if necessary, depending on the network.
3. Choose a download site (for example, <http://mirrors.kernel.org>).
4. Include the latest versions of the following packages in the Cygwin installation (in addition to those included in the base installation):
 - Archive/unzip
 - Archive/zip
 - Devel/autoconf
 - Devel/automake
 - Devel/libtool
 - Devel/make
 - Devel/subversion (**Note:** if using TortoiseSVN/Windows7, skip this file)

- Devel/gcc-core
- Devel/gcc-g++
- Devel/mingw-gcc-core
- Devel/mingw-gcc-g++
- Devel/mingw-runtime

See Figure 26 for an example of selecting a package (as example: Devel/autoconf).

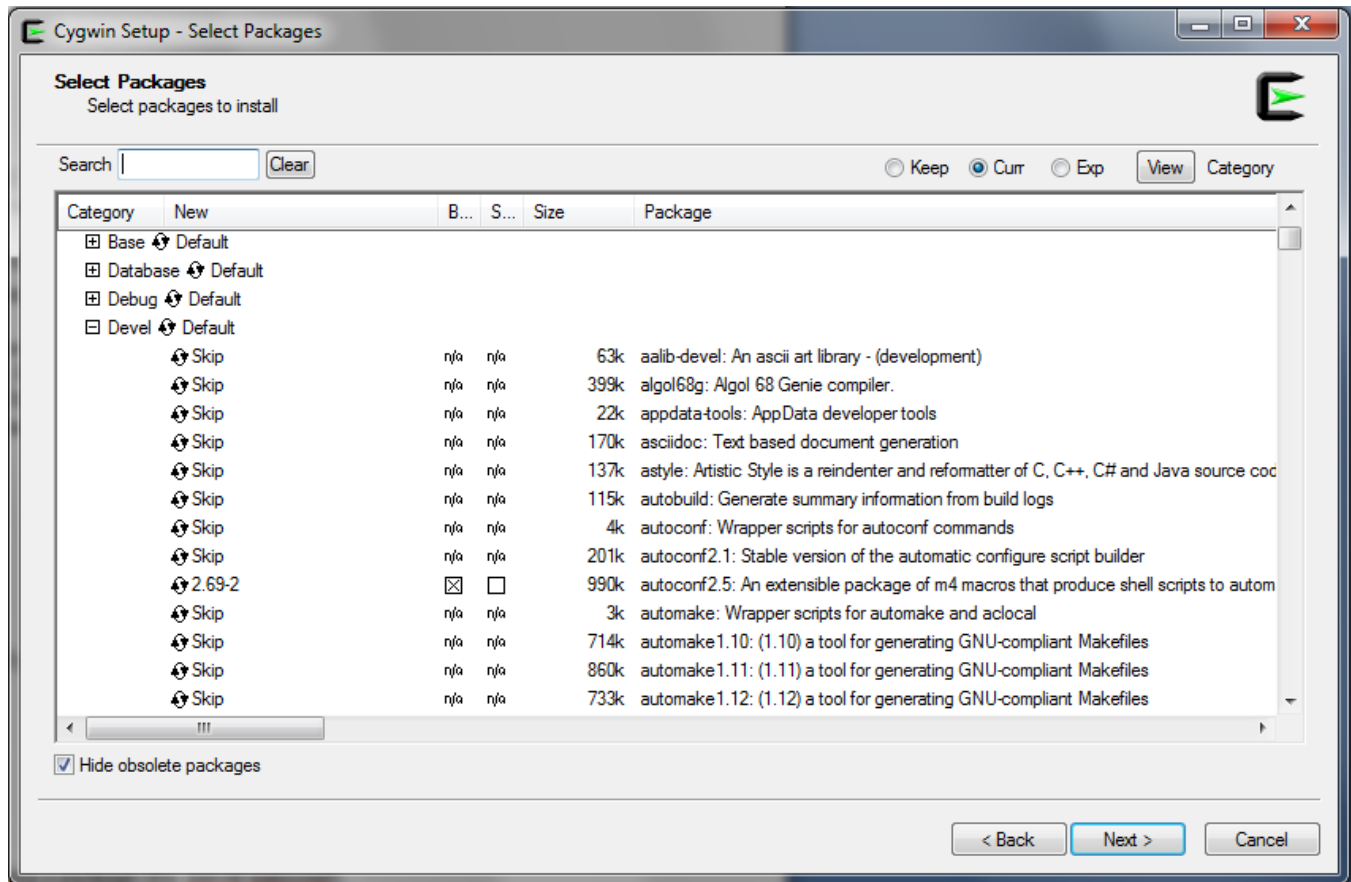


Figure 26. Cygwin Setup

5. The system will find dependencies. Press Next.
6. After a successful Cygwin installation, add its path (`c:\cygwin\bin\`) to the Windows environment variable PATH by going into *Control Panel>System>Advanced System Settings>Environment Variables*. Under *System Variables*, select PATH and press Edit. Append “`;C:\cygwin\bin\`” to the end of the line and press Ok.

3.3.2 Get GNU Tools for ARM Embedded Processors

Download and run the latest version of `gcc-arm-none-eabi-<version>-win32.exe` from <https://launchpad.net/gcc-arm-embedded>. The link to the file should be on the right side of the page and will appear as a green button with the text: “gcc-arm-non...4-win32.exe.” Install under the Cygwin root directory (default: `c:\cygwin`).

3.3.3 Build OpenOCD for FTDI Interface

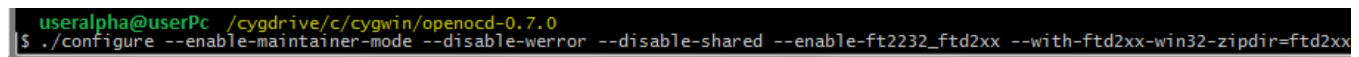
1. Download the Open On-Chip Debugger (OpenOCD) source from <http://sourceforge.net/projects/openocd/files/openocd/0.7.0/> Look for the zip file `openocd-0.7.0.zip`.
2. Extract the OpenOCD source into the Cygwin directory (`c:\cygwin`). This will create a directory called `openocd-<version>` (for example, `c:\cygwin\openocd-0.7.0`) under the Cygwin directory containing all

the OpenOCD source contents.

3. Download the FTDI driver library (x86 [32-bit] zip version) at <http://www.ftdichip.com/Drivers/CDM/CDM%20v2.10.00%20WHQL%20Certified.zip>.
4. Extract the FTDI source into the path `c:\cygwin\openocd-<version> ftd2xx` (for example, `c:\cygwin\openocd-0.7.0\ftd2xx`).
5. Run the Cygwin terminal and change the directory to `openocd-<version>` (for example, by using a command such as: `cd c:\cygwin\openocd-0.7.0`).
6. Run the following command:

```
./configure --enable-maintainer-mode --disable-werror --disable-shared --enable-ft2232_ftd2xx --with-ftd2xx-win32-zipdir=ftd2xx
```

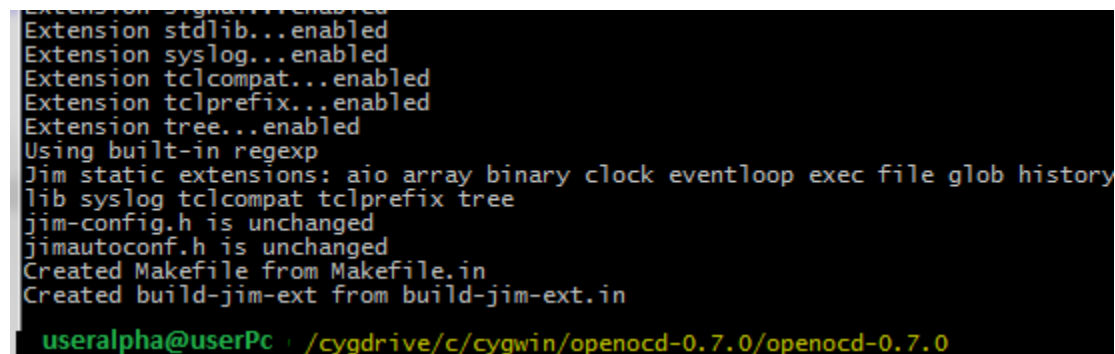
The command should look similar to [Figure 27](#).



```
useralpha@userPc /cygdrive/c/cygwin/openocd-0.7.0
$ ./configure --enable-maintainer-mode --disable-werror --disable-shared --enable-ft2232_ftd2xx --with-ftd2xx-win32-zipdir=ftd2xx
```

Figure 27. Cygwin Terminal

The last lines of the result should appear as in [Figure 28](#).



```
Extension stdlib...enabled
Extension syslog...enabled
Extension tclcompat...enabled
Extension tclprefix...enabled
Extension tree...enabled
Using built-in regexp
Jim static extensions: aio array binary clock eventloop exec file glob history
lib syslog tclcompat tclprefix tree
jim-config.h is unchanged
jimautoconf.h is unchanged
Created Makefile from Makefile.in
Created build-jim-ext from build-jim-ext.in
useralpha@userPc /cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0
```

Figure 28. Cygwin Terminal

7. Run the command `'autoreconf --force --install.'`
8. Run the command `'make.'` This may take several minutes. The last lines of the result should appear as in [Figure 29](#).



```
rc=0; \
CDPATH="${ZSH_VERSION+}:" && cd .; \
else \
rc=$?; \
CDPATH="${ZSH_VERSION+}:" && cd . && \
$restore $backupdir/* `echo "/openocd.info" | sed 's|[/]#|/|'`; \
fi; \
rm -rf $backupdir; exit $rc
make[2]: Leaving directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0/doc'
make[2]: Entering directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
make[2]: Leaving directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
make[1]: Leaving directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
useralpha@userPc /cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0
$
```

Figure 29. Running the Make Command

9. Run the command `'make install.'` The last lines of the result should appear as in [Figure 30](#).

```
make[2]: Entering directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
make[2]: Nothing to be done for 'install-exec-am'.
/usr/bin/mkdir -p '/usr/local/share/openocd'
/usr/bin/mkdir -p '/usr/local/share/openocd/contrib'
/usr/bin/install -c -m 644 contrib/openocd.udev '/usr/local/share/openocd/contrib'
/usr/bin/mkdir -p '/usr/local/share/openocd/contrib/libdccc'
/usr/bin/install -c -m 644 contrib/libdccc/dcc_stdio.c contrib/libdccc/dcc_stdio.h contrib/libdccc/example/share/openocd/contrib/libdccc'
make install-data-hook
make[3]: Entering directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
for i in $(find ./tcl -name '*.cfg' -o -name '*.tcl' -o -name '*.txt' | sed -e 's,^./tcl,,'); do \
j="/usr/local/share/openocd/scripts/$i" && \
mkdir -p "$(dirname $j)" && \
/usr/bin/install -c -m 644 ./tcl/$i $j; \
done
make[3]: Leaving directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
make[2]: Leaving directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'
make[1]: Leaving directory '/cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0'

useralpha@userPc /cygdrive/c/cygwin/openocd-0.7.0/openocd-0.7.0
$
```

Figure 30. Running the Make Install Command

10. After the command has run successfully, check that the file `openocd.exe` is generated at path `C:\cygwin\usr\local\bin`. Add this path to the Windows PATH environment variable.

3.3.4 Compile the GCC SDK project

1. Open the `common.h` file located at the path `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\example\common`.
2. Edit `common.h` to use the SSID, security type and security key of the Access Point being used. Edit the macros `SSID_NAME`, `SECURITY_TYPE` and `SECURITY_KEY` to contain the Access Point's information as shown in Figure 31. The security types supported for this demo are WPA/WPA2 and Open. For Open security, define `SECURITY_TYPE` as `SL_SEC_TYPE_OPEN`. For WPA and WPA2 security, define it as `SL_SEC_TYPE_WPA`.

<pre>// Values for below macros shall be modified as per access-point(A // SimpleLink device will connect to following AP when application // #define SSID_NAME "cc3200demo" /* AP SSID */ #define SECURITY_TYPE SL_SEC_TYPE_OPEN /* Security type (OPEN) #define SECURITY_KEY "" /* Password of the sec #define SSID_LEN_MAX 32 #define BSSID_LEN_MAX 6</pre>		<pre>// Values for below macros shall be modified as per access-point(A // SimpleLink device will connect to following AP when applicatio // #define SSID_NAME "Your_AP_Name_Here" /* AP SSID */ #define SECURITY_TYPE SL_SEC_TYPE_WPA /* Security type (OPEN #define SECURITY_KEY "Your_AP_Security_Key_Here" #define SSID_LEN_MAX (32) #define BSSID_LEN_MAX (6)</pre>
---	--	---

Figure 31. Editing `common.h`

3. Save `common.h`.

In the Cygwin terminal, change the directory to `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\example\getting_started_with_wlan_station\gcc` and run following command:

```
make -f Makefile
```

This command should appear as in Figure 32. Note that Cygwin uses forward slashes to separate directories.

```
useralpha@userPc /cygdrive/c/ti/CC3200SDK_0_5/cc3200-sdk/example/getting_started_with_wlan_station/gcc
$ make -f Makefile
CC      ../main.c
CC      ../pinmux.c
CC      ../../common/gpio_if.c
CC      ../../common/startup_gcc.c
LD      exe/wlan_station.axf

useralpha@userPc /cygdrive/c/ti/CC3200SDK_0_5/cc3200-sdk/example/getting_started_with_wlan_station/gcc
$
```

Figure 32. Makefile Command

This generates the `wlan_station.axf` file under the `gcclexe` folder.

3.3.5 Target Connection and Debug (GDB)

1. The OpenOCD configuration file for FTDI is present under the `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\tools\gcc_scripts\` folder. To test the connection to the CC3200 FTDI Launchpad, navigate to the `<cc3200-sdk>\tools\gcc_scripts` folder in the Cygwin terminal, run the following command and check the output to see if the connection happened properly.

```
openocd -f cc3200.cfg
```

See [Figure 33](#) for the output screen while the CC3200 device is connected through GDB.



```
useralpha@userPc /cygdrive/c/ti/CC3200SDK_0_5/cc3200-sdk/tools/gcc_scripts
$ openocd -f cc3200.cfg
Open On-Chip Debugger 0.7.0 (2014-05-28-05:04)
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 1000 kHz
cc3200_dbginit
Info : device: 4 "2232C"
Info : deviceID: 72467242
Info : SerialNumber: cc3200A
Info : Description: USB <-> JTAG/SWD A
Info : clock speed 1000 kHz
Info : JTAG tap: cc3200.jrc tap/device found: 0x0b97c02f (mfg: 0x017, part: 0xb97c, ver: 0x0)
Info : JTAG tap: cc3200.dap enabled
Info : cc3200.cpu: hardware has 6 breakpoints, 4 watchpoints
```

Figure 33. Output Screen

2. Press `<ctrl>+c` to return to prompt.
3. Copy the `wlan_station.axf` file found in `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\example\getting_started_with_wlan_station\gcclexe\` to the directory `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\tools\gcc_scripts\`.
4. Launch Tera Term, and create a new serial connection to the CC3200 Launchpad COM port as shown in [Figure 34](#).

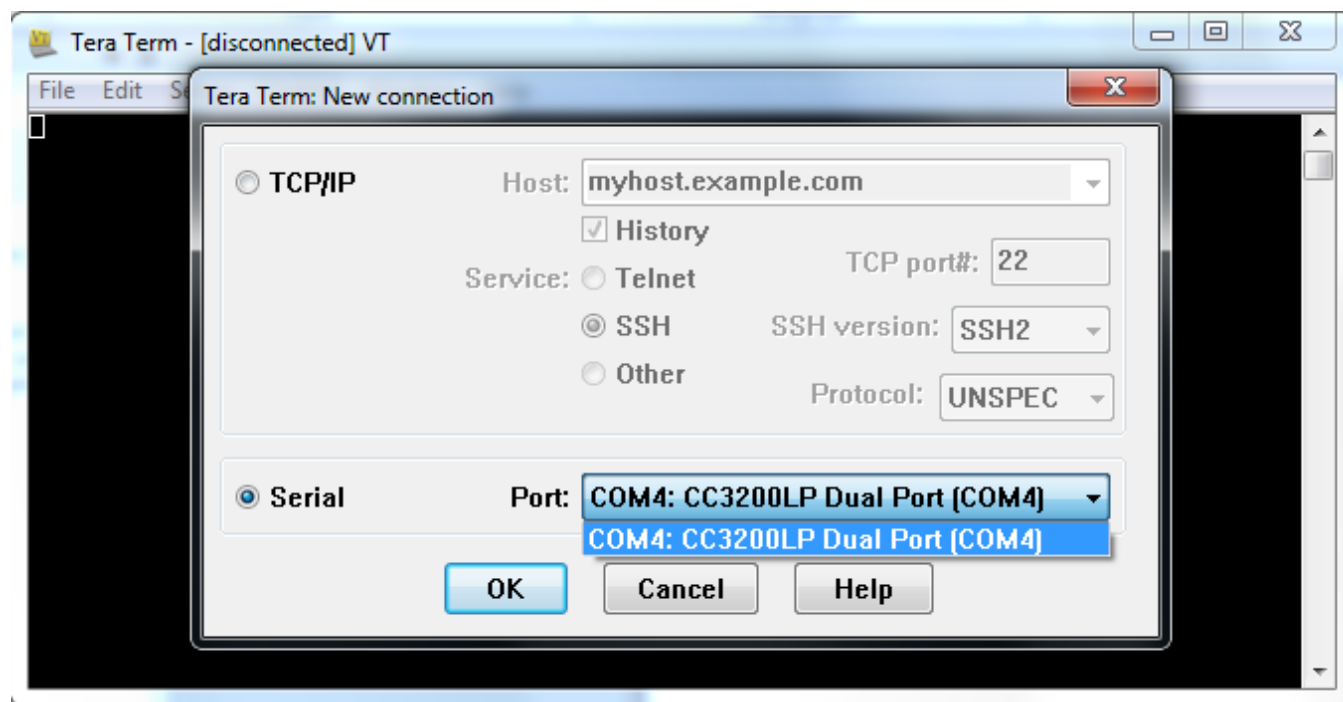


Figure 34. Launch Tera Term

5. In the menu, select Setup>Serial Port, and change the baud rate to 115200 as shown in Figure 35.

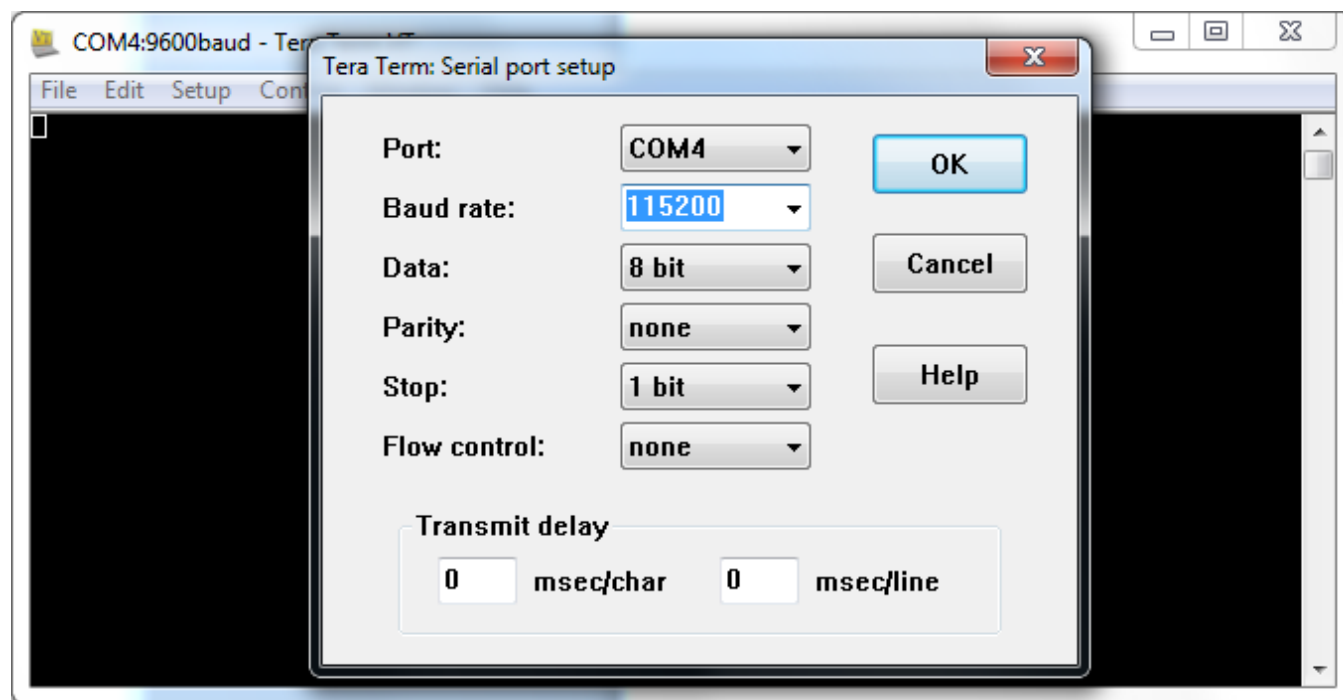


Figure 35. Tera Term Serial Port Setup

6. To start debugging using GDB on CC3200, go to `C:\TI\CC3200SDK_1.0.0\cc3200-sdk\tools\gcc_scripts\` and run the following command at the Cygwin prompt:

```
arm-none-eabi-gdb -x gdbinit wlan_station.axf
```

This results in a GDB prompt. To continue, type 'continue' and press enter. For other commands,

consult the [GDB Quick Guide](#). See [Figure 36](#) for the result of debugging the `wlan_station` application from GCC.

```

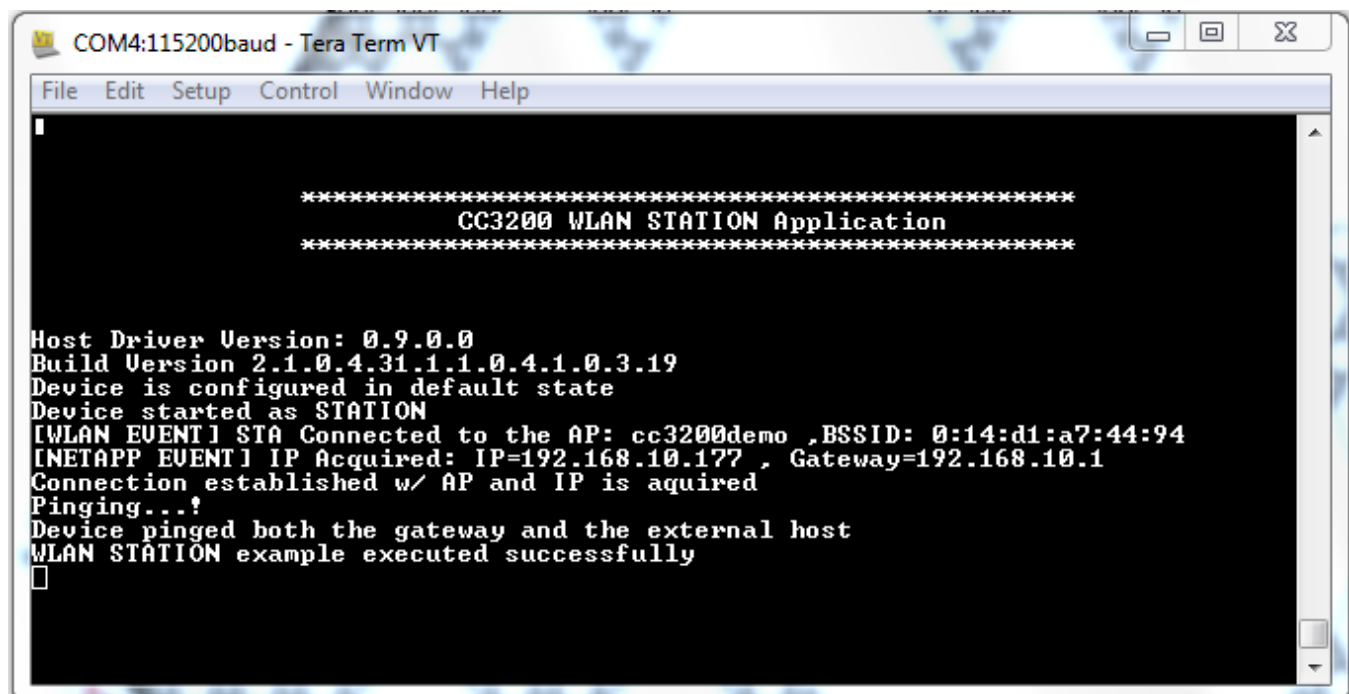
useralpha@userPc /cygdrive/c/ti/CC3200SDK_0_5/cc3200-sdk/tools/gcc_scripts
$ arm-none-eabi-gdb -x gdbinit wlan_station.axf
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20140228-cvs
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from C:\ti\CC3200SDK_0_5\cc3200-sdk\tools\gcc_scripts\wlan_station.axf...done.
0x0000230e in ?? ()
Open On-Chip Debugger 0.7.0 (2014-05-28-05:04)
Licensed under GNU GPL v2
For bug reports, read
http://openocd.sourceforge.net/doc/doxygen/bugs.html
Loading section .text, size 0x77a0 lma 0x20004000
Loading section .data, size 0x858 lma 0x2000b7a0
Start address 0x20004974, load size 32760
Transfer rate: 88 KB/sec, 10920 bytes/write.
Breakpoint 1 at 0x20004722: file ../main.c, line 483.

Breakpoint 1, main () at ../main.c:483
483      BoardInit();
(gdb) |

```

Figure 36. Debugging `wlan_station`

7. If the CC3200 successfully completes all steps, the serial output appears as shown in [Figure 37](#).



```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

*****
CC3200 WLAN STATION Application
*****

Host Driver Version: 0.9.0.0
Build Version 2.1.0.4.31.1.1.0.4.1.0.3.19
Device is configured in default state
Device started as STATION
[WLAN EVENT] STA Connected to the AP: cc3200demo ,BSSID: 0:14:d1:a7:44:94
[NETAPP EVENT] IP Acquired: IP=192.168.10.177 , Gateway=192.168.10.1
Connection established w/ AP and IP is aquired
Pinging...!
Device pinged both the gateway and the external host
WLAN STATION example executed successfully

```

Figure 37. Tera Term VT

4 Summary

After the development environment has been set up, see the following resources for further assistance in development:

- [CC3200 Programmer's Guide](#) – This guide contains information on how to use the SimpleLink API for writing WLAN-enabled applications.
- [PinMux Tool](#) – This utility helps determine how to best assign peripherals to the appropriate CC3200 package pins.
- [Uniflash](#) – The Uniflash tool manually stores files on the external serial flash. This includes the application binary and SimpleLink firmware patch files. Also, any configuration files, security certificates, web pages, and so forth can be stored using this tool.
- [CC3200 Wiki](#) – All information and tools for the CC3200, including the above, can be found on the CC3200 Wiki page.

5 Acronyms Used

STA – Wi-Fi Station

AP – Wi-Fi Access Point

WLAN – Wireless LAN

CCS – Code Composer Studio

GCC – GNU Compiler Collection

Revision History

Changes from Original (June 2014) to A Revision	Page
• Changed image.....	3
• Changed image.....	3
• Replaced image.	3
• Changed path to C:\TI\CC3200SDK_1.0.0\.....	5
• Added Tera Term download link.....	5
• Updated image with SOP-2 Jumper marked	5
• Moved Steps 1 and 8 to Section 2.2, removed Section 2.3.	5
• Added Update Service Pack section.....	6
• Added function #4	7
• Updated version number.	7
• Changed to Select the Wireless Connectivity MCUs option for processor support.....	7
• Updated image	7
• Changed to The CC3200 Add-On should already be installed. If not, select it.	8
• Updated image.....	8
• Changed path.	8
• Updated image.....	9
• Updated text.....	9
• Replaced Step Four.	10
• Updated image	11
• Replaced main.c with common.h.	12
• Updated path.	12
• Changed image	12
• Changed path.	13
• Added Steps 16, 17, and 18.....	15
• Changed path.	16
• Changed path.	17
• Changed path.	19
• Replaced main.c with common.h.	19
• Changed path.	19
• Changed image	20
• Added Steps 7, 8 and 9.	22
• Replaced main.c with common.h.	25
• Changed path.	25
• Replaced image.....	25
• Changed path.	25
• Changed path.	26
• Changed path.	26
• Changed path.	27
• Added Steps 5, 6, and 7.	28
• Updated link.	29

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com