CC31xx SimpleLink Studio User Guide

Return to CC31xx Main Page

NOTE: Texas Instruments NDA Restrictions Applicable.

Welcome to the CC31xx SimpleLink Studio for CC3100 Guide. Here you will find an explanation of the SimpleLink Studio for CC3100 and a list of example applications.

Introduction

SimpleLink Studio for CC3100 is a tool to aid in the development of applications designed to work with the SimpleLink WiFi CC31xx family of wireless chips. Using SimpleLink Studio for CC3100, code can be written and run in a desktop IDE, such as Visual Studio or Eclipse. This allows the code to be easily tested while it is under development, and then later ported to a microcontroller without having to be modified. Generally, the only modifications required to port a SLS program to another platform are made to a special user.h file that is included in the project.

Benefits of SimpleLink Studio for CC3100

- Code can be written and run from a computer, making debugging significantly easier as the code does not need to be run from the MCU.
- Software development can begin before the hardware development has finished, due to the fact that the hardware isn't required to run the code.
- The code is MCU agnostic, so it can be ported to any MCU that has had the SimpleLink Driver ported to it.
- The SimpleLink Studio for CC3100 package contains multiple examples that help to understand its API and structure.
- Knowledge of complicate wireless networking needs is not required when using SimpleLink Studio for CC3100.

Porting SimpleLink Studio for CC3100 to a New MCU

In order to port SimpleLink Studio for CC3100, a header file called **user.h** must be created with all of the porting info. This process involves the following steps. A detailed description of this process can be found in the CC31xx Porting Guide.

1. Create a user.h file: A file called **user.h** must be created and included in the project. All modifications done in order to port the driver must be made to this file.

2. Select a capability set: The API that the MCU will need must be specified, or a predefined capability set can be chosen. The smallest capability set that fits the needs of the application should be selected in order to conserve memory.

3. Bind the enable/disable line: The chip enable line must be programmed in order to conserve power. Without programming this the driver will only be able to be started once.

4. Write the interface communication driver: Write the driver for either SPI or UART communication. This driver must include the abilities to read/write and open/close.

5. Choose your memory management model: Choose to use either static or dynamic memory. Static is the default.

6. Choose your OS platform: Choose whether to use a single threaded or multi-threaded OS.

7. **Set event handlers**: Set the asynchronous event handlers to be used.

8. **Run the diagnostic tools**: Use the included diagnostic tools to verify that the port was successful.

Environment Setup

The user has the option of using either **Visual Studio** or **Eclipse** to run the Simplelink Studio applications.

Loading Application Using Visual Studio 2010

Working with Visual Studio will require opening the provided project and compiling it.
In order to open and compile the project, please follow these steps:

1. Open Visual Studio.

2. From the menu: *Open*, then select *File*.

3. Browse to the <*Project_Folder_Name*> Visual Studio folder. This will be under <*Project_Directory*> in the CC31xx release folder.

4. Select the <*Project_Name*> Visual Studio project. Click on Open. *Note:* If the project does not open, make sure that you did not accidentally select the Visual Studio Project Filter File.

5. The project should be created automatically, along with the following folder:

- External Dependencies
- SimpleLink
- Source Files

6. **Build your project**: Build the project by clicking on *Debug* at the top of the screen and then selecting *Start Debugging*, or by pressing F5.

Loading Application Using Eclipse

Working with Eclipse will require opening the provided project and compiling by using MinGW GCC.
MinGW Compiler Setup for Windows systems (if you are running linux, you may skip this section).

1. Download and install MinGW from here [1].

2. During installation, make sure you have the following configuration selected:

- Memorize the installation location, you'll need it in the following step (Default: C:\MinGW).
- In **MinGW Installation Screen** screen, select package for **basic mingw** , **C Compiler** and **C++ Compiler**.

3. After installation, make sure you have the Environment variable setup with the following:

- Click **Start** -> type **sysdm.cpl** -> select **Advanced** tab -> click **Environment Variables...** on the very bottom -> In the System variable section, scroll down to the **Path** variable -> click **Edit...**
- At the end of Variable value, add the path to your MinGW binaries directory with semicolon in front (For example, **;C:\MinGW\bin**).
- Click OK. Environment variable path to MinGW compiler is done here.

Eclipse Setup:

1. Open Eclipse

2. In the "Select a workspace", choose your desired workspace directory (For example: *C:\Users\myself\Desktop\eclipse_workspace*).

3. From the menu: *Open*, select *File* -> New -> Makefile Project with Existing Code

4. In the New Project window, click the Browse button to browse to the <*Project_Folder_Name*> Eclipse folder. This will be under <*Project_Directory*> in the CC31xx release folder, then click OK.

5. Under *Toolchain for Indexer Settings* section, select **MinGW GCC**, then click Finish. Project will be imported to your Eclipse environment.

6. In project Project Explorer window, right click on <*Project_Folder_Name*>, then select property.

7. Expand the **C/C++ Build** menu, select Tool Chain Editor. Select "Gnu Make Builder", then click Apply button.

8. Select Environment in the **C/C++ Build** menu. Make sure the value of MYSYS_HOME is empty, then click the Apply button.

9. Click the **C/C++ Build** menu and do the following configurations, then click OK:

- Uncheck **Use the default build command**
- Type *mingw32-make -f Makefile* in **Build command**
- Uncheck **Generate Makefiles automatically**

10. IMPORTANT: This step is critical for Eclipse on Windows systems due to a known bug [2] in Eclipse console output. Please add the following line of code to the beginning of the main function:

```
setvbuf(stdout, NULL, _IONBF, 0);
```

11. You program is ready to run, press Ctrl+F11 to start running.

Archives

The latest software packages are the ones which are currently supported.

Note: Note that, if any application is duplicated between the latest download page and the archives, the one based on the latest service pack takes precedence.

- SimpleLink Studio for CC3100 **v0.2**

References

[1] http://sourceforge.net/projects/mingw/files/latest/download?source=files

[2] https://bugs.eclipse.org/bugs/show_bug.cgi?id=236330

Article Sources and Contributors

CC31xx SimpleLink Studio User Guide *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?oldid=188980 *Contributors*: A0131875, A0132173, Agarcia, Beatrice, Logan Han

Image Sources, Licenses and Contributors

File:Light_bulb_icon.png *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Light_bulb_icon.png *License*: unknown *Contributors*: DanRinkes