UG235.08: Using the Command Line Interface with Silicon Labs Connect

This chapter of the *Connect User's Guide* describes how to use the Command Line Interface (CLI) with Silicon Labs Connect and offers implementation guidance for adding a CLI command. The Connect stack is delivered as part of the Silicon Labs Flex SDK. The *Connect User's Guide* assumes that you have already installed the Simplicity Studio development environment and the Flex SDK, and that you are familiar with the basics of configuring, compiling, and flashing Connect-based applications. Refer to *UG235.01: Developing Code with Silicon Labs Connect* for an overview of the chapters in the *Connect User's Guide*.

KEY POINTS

- Introduces the Command Line Interface (CLI).
- Offers implementation guidance for adding a CLI command.

The *Connect User's Guide* is a series of documents that provides in-depth information for developers who are using the Silicon Labs Connect Stack for their application development. If you are new to Connect and the Flex SDK, see *QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio*.

Connect is supported for EFR32FG, EFR32MG1x, and EFR32BG1x.

1 Introduction

Embedded application developers need to interact with the embedded system throughout both their development and finished product phases—for example, to change the settings or monitor the operation of a device. In many cases, the easiest and most common way to complete these tasks is to use the device's UART interface. The Connect stack provides an extensible Command Line Interface (CLI) implementation for this purpose.

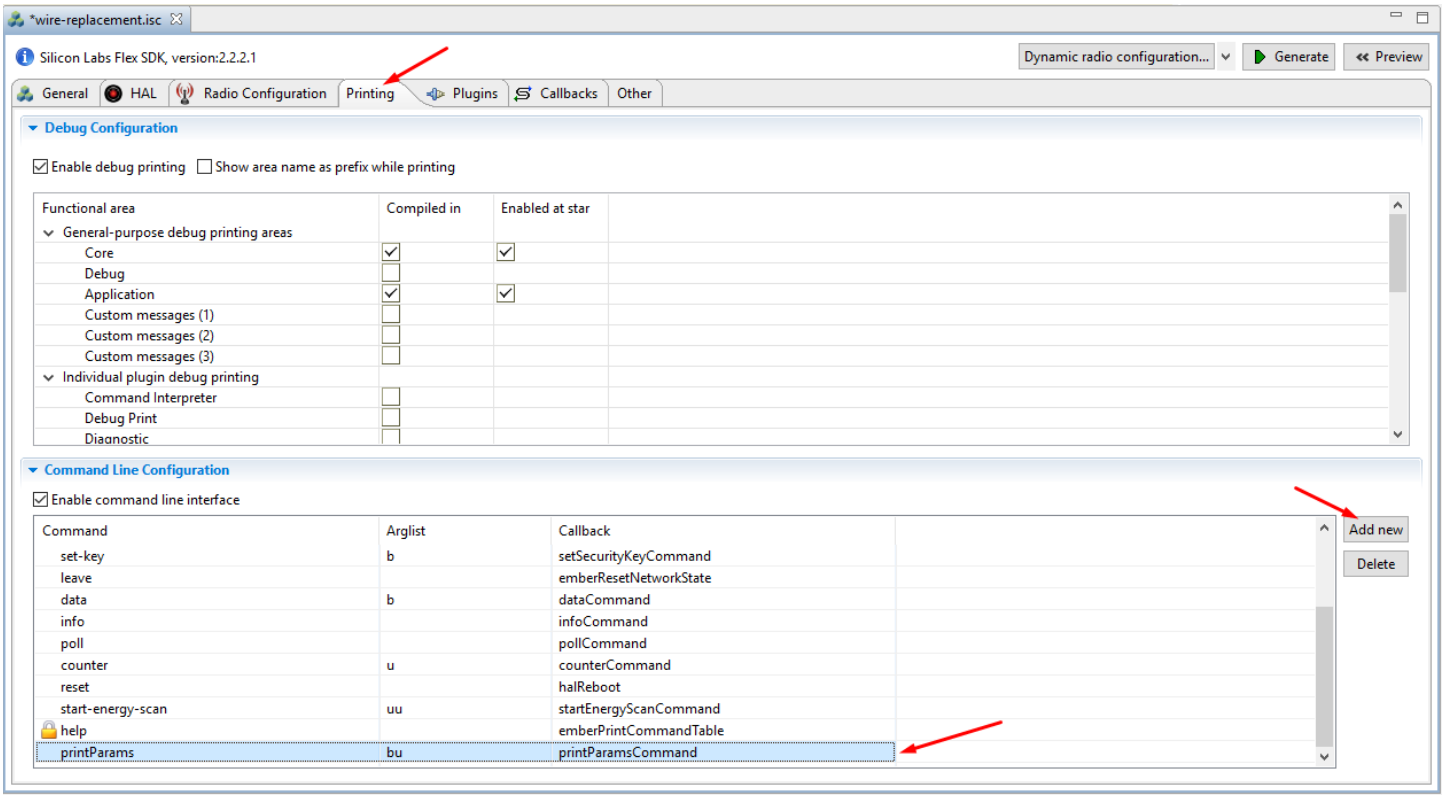
2 Adding a CLI Command

By default, all Connect example projects select CLI commands already implemented that interact with the application through the device's UART interface.

This example project explain how to add other CLI commands using the `Connect (SoC): Wire-Replacement` application. You can create the project by using the example application Project Wizard.

To add a new CLI command:

1. Open AppBuilder (it opens automatically when you create a new project) and then click the **Printing** tab.



In the **Command Line Configuration** frame, click the **Add New** button. This will insert a new line at the end of the command line list. (Because it will be inserted at the end, you may need to scroll down to the bottom of the list to see the new line.)

There are three columns for a command:

- **Command:** the command name as it will be available on the UART console
- **Arglist:** the list of argument types that the command will accept
- **Callback:** the function that will be called on command execution from the command line

Arglist accepts the following argument types:

- `u`: one-byte unsigned integer
- `v`: two-byte unsigned integer
- `w`: four-byte unsigned integer
- `s`: one-byte signed integer
- `b`: string

Note: Commands added in the AppBuilder will be displayed in the online help, included in the source code, and will be available at the command line.

The string argument can be entered in ASCII by using double quotes—for example: `"string"`—or in hexadecimal values by using curly braces—for example: `{48 45 58 0A}`.

You can find the parameter descriptions in the `command-interpreter2.h` file in the `command-interpreter` folder of the project at declaration of `EmberCommandEntry`.

2. Modify the **Command** field to `printParams`, the **Callback** field to `printParamsCommand`, and the **Arglist** to `bu` (a string and a one-byte unsigned integer) and click the **Generate** button.

   The generate action will create the prototype of the callback in `flex-cli.c`.

   ```
   void printParamsCommand(void);
   ```

3. Add the newly created command to the `emberCommandTable[]`.

   The corresponding line should look like this:

   ```
   emberCommandEntryAction("printParams", printParamsCommand, "bu", ""),
   ```

   The last parameter is an empty string that is displayed in the online help when the `help` command is executed in the CLI. AppBuilder does not support this parameter. It overwrites any modifications made manually in `flex-cli.c`.

4. Implement the callback in `flex-callbacks.c`.

   ```
   void printParamsCommand(void)
   {
     uint8_t stringLength;
     uint8_t *stringContents = emberStringCommandArgument(0, &stringLength);
     uint8_t numericValue;
     emberAfCorePrintln("First parameter (string): %s", stringContents);
     emberAfCorePrintln("First parameter length: %d", stringLength);
     numericValue = emberUnsignedCommandArgument(1);
     emberAfCorePrintln("Numeric value: %u", numericValue);
   }
   ```

   The `command-interpreter2-util.c` file contains the helper functions for retrieving the values of the command line arguments. The most commonly-used functions are:

   - `emberUnsignedCommandArgument()`
   - `emberSignedCommandArgument()`
   - `emberStringCommandArgument()`

   In all cases the first parameter of the call is the location of the argument in the argument list (zero-based index value). In the case of `emberStringCommandArgument()`, the second parameter is a pointer to the value of the string length.
   If the project successfully compiled, the new command appears in the help:

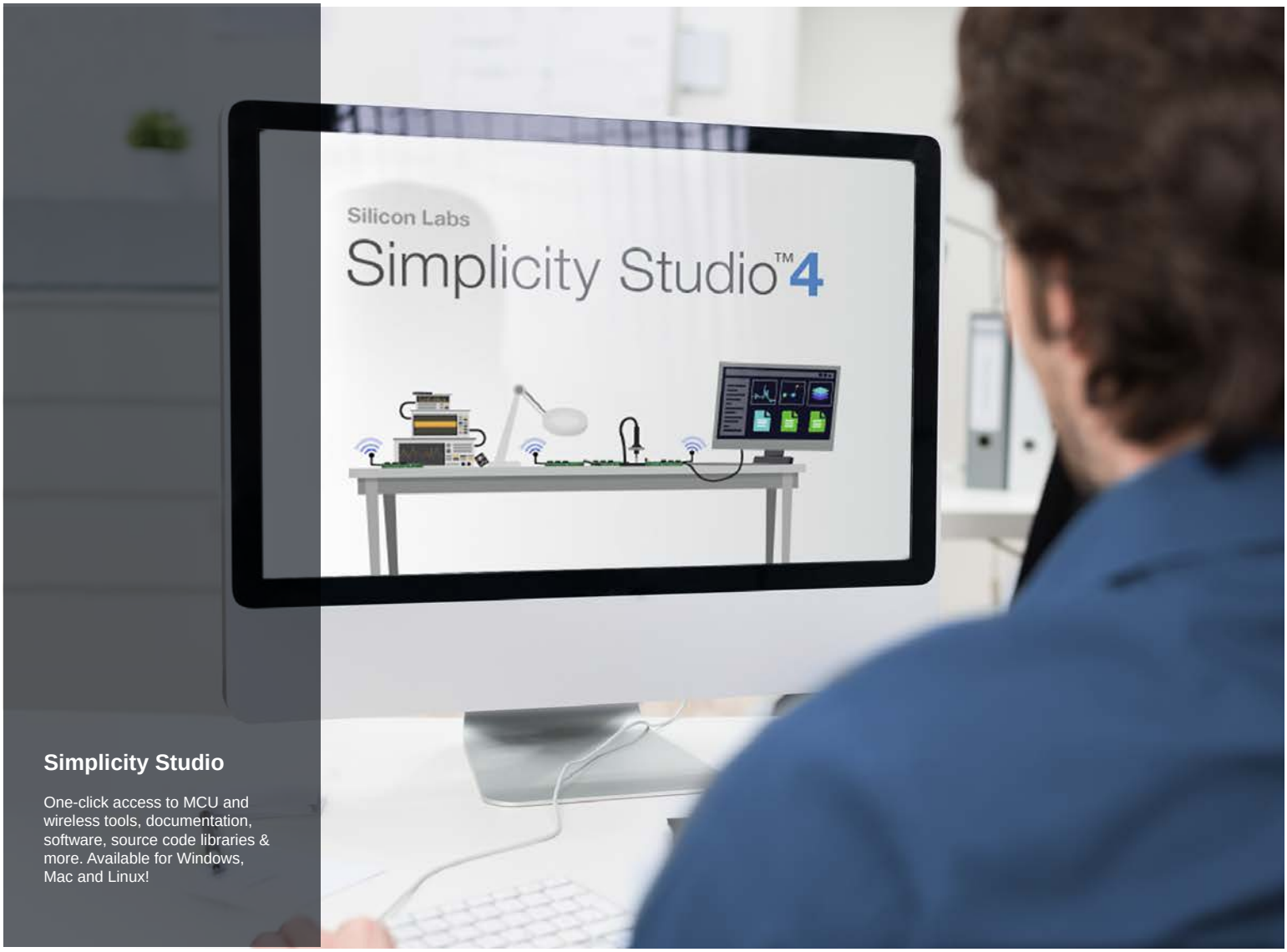
```
wire-replacement>help

type        description
<uint8_t>    8-bit unsigned int, eg: 255, 0xAB
<int8_t>    8-bit signed int, eg: -128, 0xA9
<uint16_t>  16-bit unsigned int, eg: 3000 0xFFAA
<string>  A string, eg: "foo" or {0A 1B 2C}
*         Zero or more of the previous type

counter <uint8_t> -
data <string> -
help -
info -
leave -
poll -
printParams <string> <uint8_t> -
reset -
set-key <string> -
set-tx-options <uint8_t> -
set-tx-power <int8_t> -
start-energy-scan <uint8_t> <uint8_t> -
start-master <uint8_t> -
start-slave <uint8_t> -
start-sleepy-slave <uint8_t> -
wire-replacement>
```

Executing the command should print the argument's values on the console:

```
wire-replacement>printParams <48 65 6c 6c 6f 00> 42
First parameter (string): Hello
First parameter length: 6
Numeric value: 42
wire-replacement>
```



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com