PRECISION32™ SI32FLASHUTILITY COMMAND-LINE PROGRAMMER USER'S GUIDE

1. Introduction

The Precision32™ si32FlashUtility Command-Line Programmer is a simple program to enable production programming capability using the Silicon Labs 32-bit USB Debug Adapter. This utility can also program and erase lock bytes.

Figure 1 shows an invocation of the command-line utility.

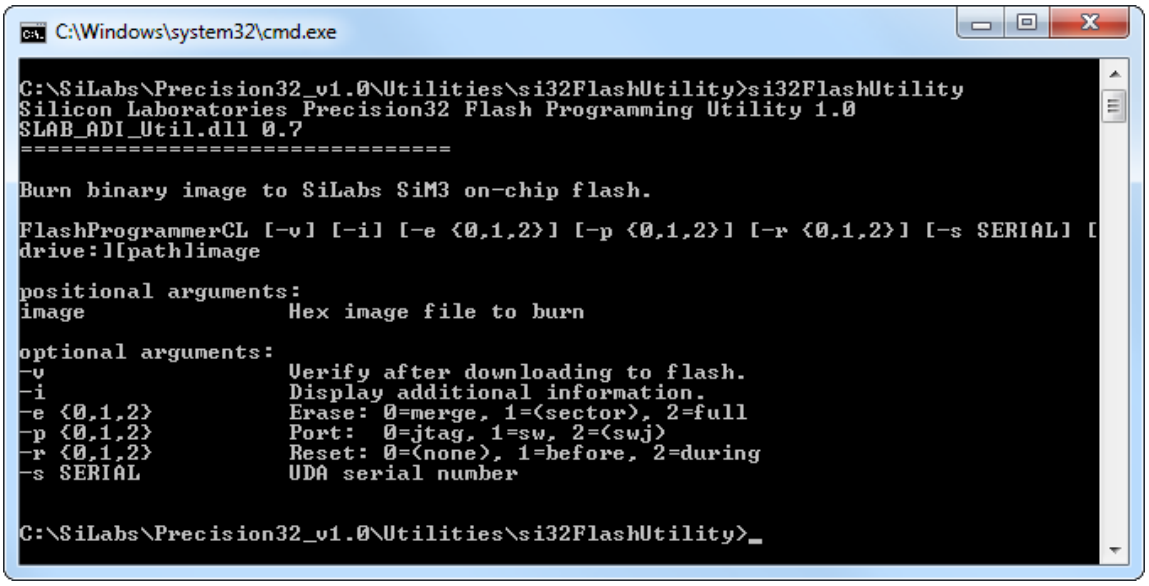


Figure 1. Precision32 si32FlashUtility Command-Line Programmer

2. Relevant Documentation

Precision32 Application Notes are listed on the following website: www.silabs.com/32bit-appnotes.

- AN667: Getting Started with the Silicon Labs Precision32™ IDE
- AN669: Integrating Silicon Labs SiM3xxxx Devices into the Keil µVision® IDE

3. Programming Options

The si32FlashUtlility has a command-line form of:

```
si32FlashUtility [-options] [drive:][path]image
```

These options consist of the following:

- **-v:** Verify the image after downloading to Flash.
- **-i:** Display additional information during the programming process (i.e., verbose mode).
- **-e {0,1,2}:** Erase Flash with three mode options.
- **-p {0,1,2}:** Debug port selection with three mode options.
- **-r {0,1,2}:** Reset during programming with three mode options.
- **-l:** List the available USB Debug Adapters (UDAs).
- **-s SERIAL:** Specify the USB Debug Adapter serial string.

This section discusses each of these programming options in more detail.

3.1. Download Verification

Using the **-v** option flag causes the si32FlashUtility to verify the Flash contents after the download. The command-line utility will output a **Download complete and verified** message if the Flash contents match the HEX image.

3.2. Verbose Feedback

With the **-i** option flag, the si32FlashUtility programmer will report feedback about each step of the programming process, as shown in Figure 2.

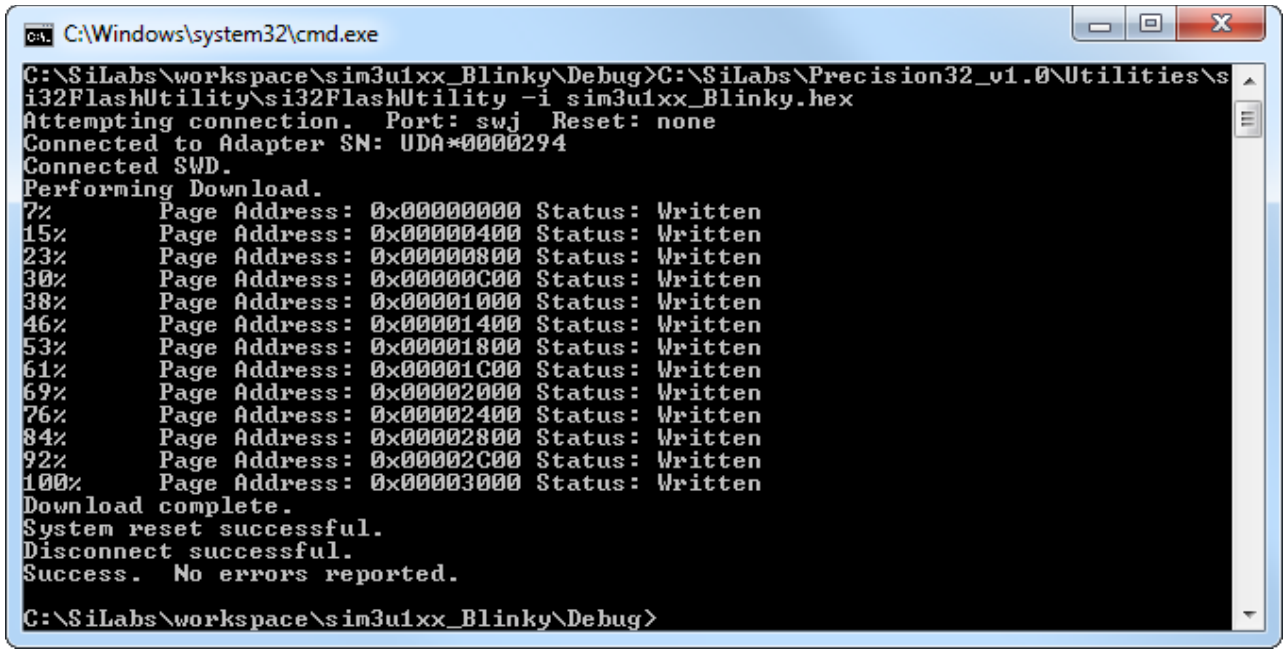


Figure 2. Verbose Mode Output

3.3. Flash Erase

The **-e** option flag has three modes: merge, sector, and full. The default option is sector (**-e 1**) if no option is specified.

The merge option is selected with **-e 0** and causes the programmer to read the current contents of the Flash page selected by the HEX file address, copy any contents that are not written in the HEX image, erase the page, and write the merged image back to Flash. This option allows developers to maintain any calibration or code constants in Flash when updating code.

When using the **-e 1** sector erase option, the programmer will first erase the page selected by the HEX image address before programming the contents of the HEX image.

The final option, **-e 2**, causes the programmer to erase the entire Flash before programming the HEX image.

3.4. Debug Port

This option selects the debug port of the device. The **-p 0** selection is for any devices with JTAG debug pins. The **-p 1** option is for devices with Serial Wire debug pins only (SW-DP). The **-p 2** option uses the Serial Wire protocol and is for devices with both JTAG and Serial Wire debug pins (SWJ-DP), like the SiM3U1xx device family. The default option is **-p 2** if no option is specified.

The JTAG selection (**-p 0**) does not have provisions for JTAG chaining.

3.5. Reset Options

The utility supports three different reset options: none, before, and during. The default option is none (**-r 0**) if no option is specified.

The none option (**-r 0**) prevents the utility from toggling the reset pin at any point during the programming process.

The before option (**-r 1**) allows the si32FlashUtility to toggle reset immediately before programming. This option is useful for SiM3U1xx or SiM3C1xx devices that may be unresponsive due to switching to a non-existent clock. Using this option along with the recommended reset delay in the startup code ensures the USB Debug Adapter will be able to communicate with the device.

For the during option (**-r 2**), the utility asserts the reset pin while attempting to halt the core. Once the core is halted, the utility deasserts the reset pin and starts programming. This option ensures the Debug Adapter can always communicate with a device without the reset delay in the startup code for devices that support this feature.

Note: The **-r 2** option is unavailable for SiM3U1xx and SiM3C1xx devices.

3.6. USB Debug Adapter Options

The **-l** option flag lists the available USB Debug Adapters connected to the PC or system. The **-s SERIAL** option flag can then specify the USB Debug Adapter the utility should use for programming.

4. Creating HEX Files with the Precision32 IDE

The si32FlashUtility programmer expects HEX files as its input, and the Precision32 IDE includes a utility that can convert the GCC AXF file output to HEX files. This **objcopy** utility can be found in the **..\Precision32_vx.y\IDE\precision32\Tools\arm-none-eabi\bin** path after installing the Precision32 software package from www.silabs.com/32bit-software.

More information on the usage of this utility can be found on the CodeRed website: http://support.code-red-tech.com/CodeRedWiki/OutputFormats.

4.1. Using the Objcopy Utility from the IDE

To use the **objcopy** utility from the IDE:

1. Hold the **Ctrl** button and **left-click** on the project name in the IDE footer as shown in Figure 3. This will open a command prompt in the project directory with the proper paths to use the utility.

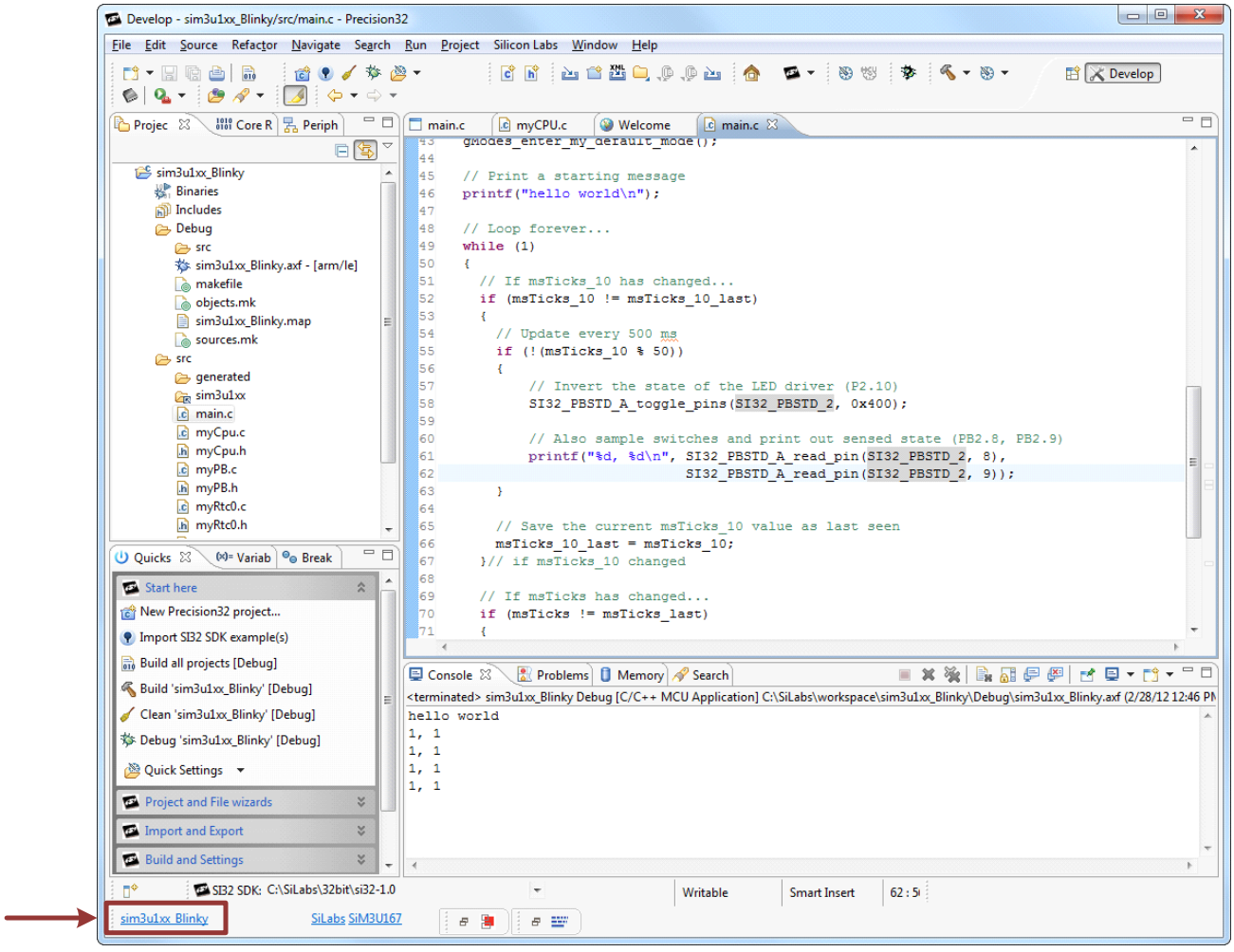


Figure 3. Opening a Project Command Prompt

2. Type **cd build_directory**, where **build_directory** is **Debug** by default.
3. Invoke the utility: **arm-none-eabi-objcopy -O ihex project_name.axf project_name.hex**. In the case of this example, which uses the sim3u1xx_Blinky project: **arm-none-eabi-objcopy -O ihex sim3u1xx_Blinky.axf sim3u1xx_Blinky.hex**.

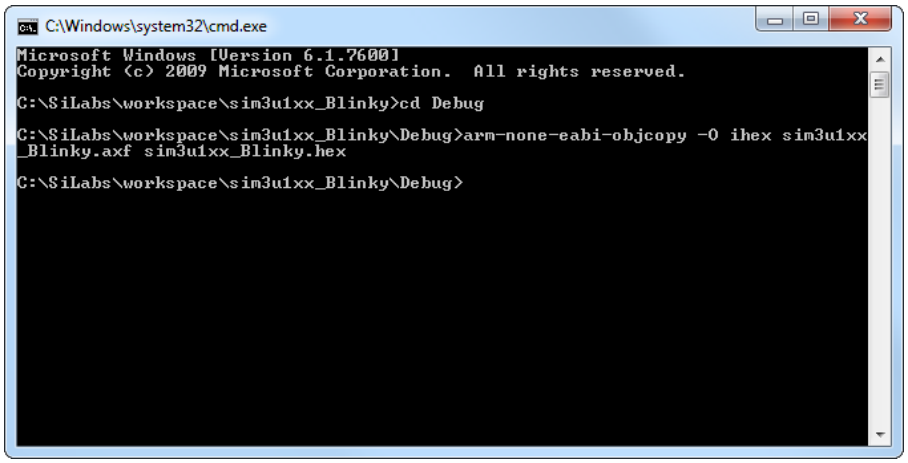


Figure 4. Invoking the Objcopy Utility

4.2. Setting the IDE Project to Automatically Generate a HEX File

To configure the Precision32 IDE project to automatically generate a HEX file after a build:

1. Right-click on the **project_name** in the **Project Explorer** view.
2. Select **Properties**.
3. In the **C/C++ Build→Settings→Build Steps** tab, type the following in the **Post-build steps→Command** box: **arm-none-eabi-objcopy -O ihex ${BuildArtifactFileName} ${BuildArtifactFileBaseName}.hex**

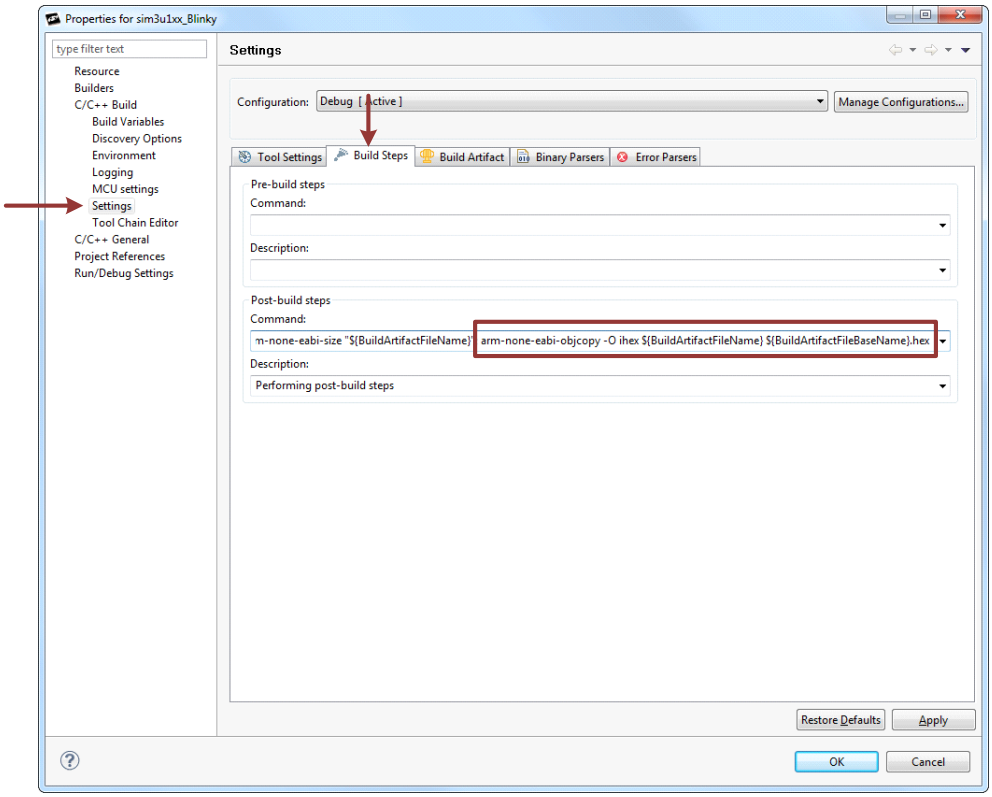


Figure 5. Automatically Generating a HEX File on Project Build

5. Examples

To verify the download of the sim3u1xx_Blinky.hex file:

```
si32FlashUtility -v sim3u1xx_Blinky.hex
```

This example is shown in Figure 6.

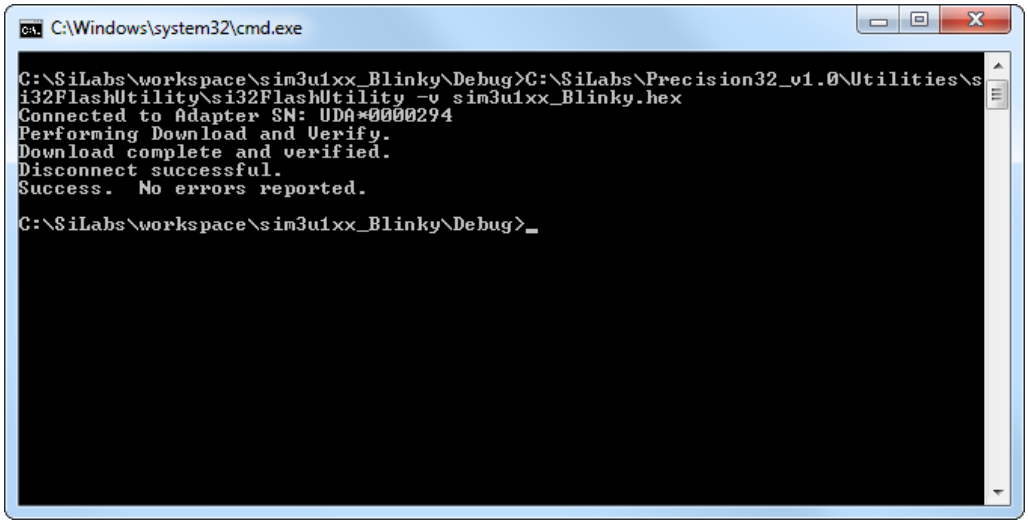


Figure 6. Example with Flash Verification

To verify the download of the sim3u1xx_Blinky.hex file, use verbose mode, erase the device before the download, and reset before:

```
si32FlashUtility -v -i -e 2 -r 1 sim3u1xx_Blinky.hex
```

Figure 7 shows an example of this call to the si32FlashUtility programmer.

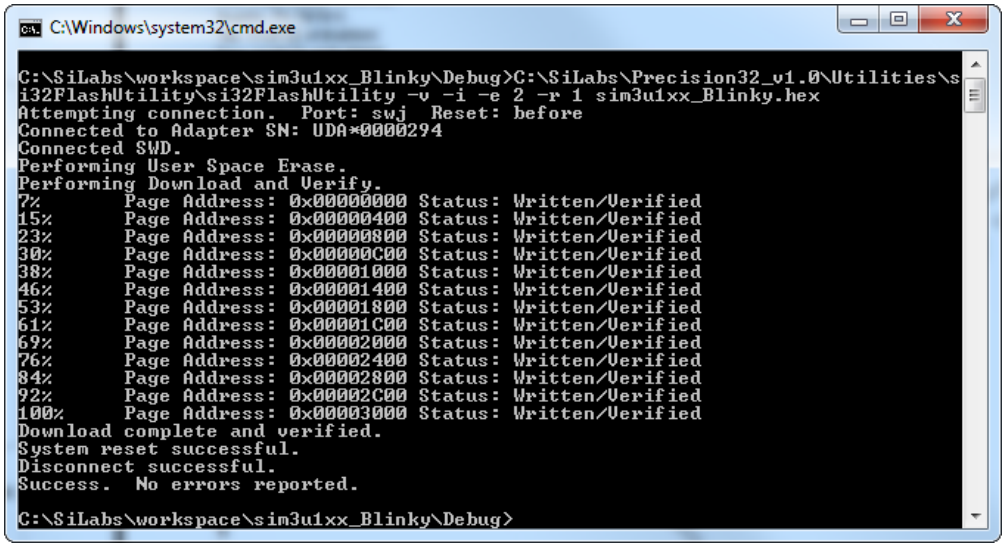
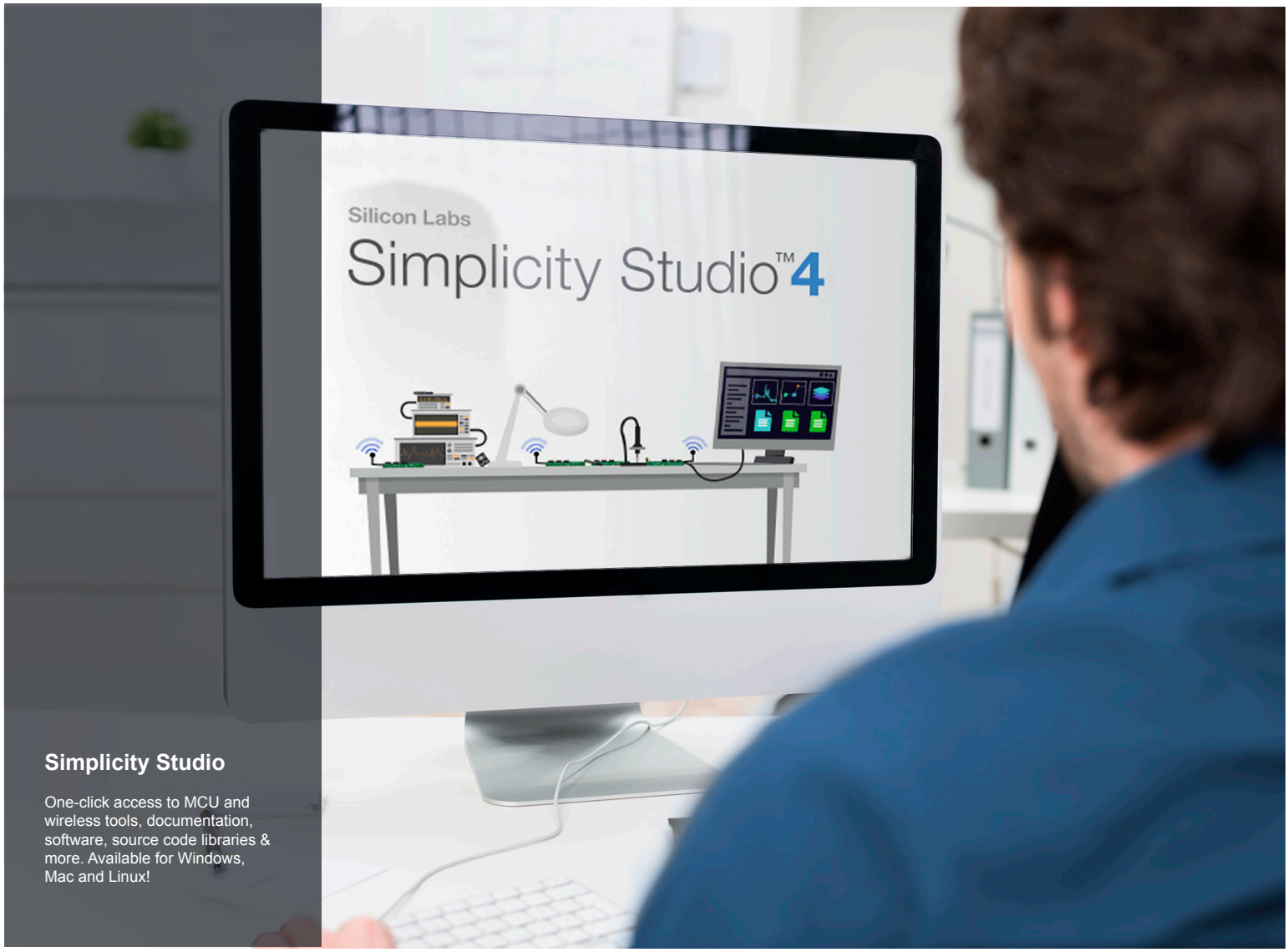


Figure 7. Example with Flash Verification, Verbose Mode, Full Device Erase, and Reset Before Options



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