CONVERTING FIRMWARE PROJECTS BETWEEN KEIL, IAR, AND PRECISION32™

1. Introduction

SiM3xxxx MCUs are supported by three IDEs: Silicon Labs Precision32 IDE, IAR Embedded Workbench, and Keil µVision. The scripts accompanying this document easily migrate project files for one IDE to another IDE to enable debugging and development on the preferred platform for the application.

2. Relevant Documentation

Silicon Labs 32-bit application notes and software are available on the website: www.siliconlabs.com/32bit-appnotes.

- **AN667: *Getting Started with the Silicon Labs Precision32 IDE*** — This document discusses the features and provides a step-by-step guide for using the Precision32 IDE.
- **AN669: Integrating Silicon Labs SiM3xxxx Devices into the Keil µVISION IDE —** This document provides a step-by-step guide for using the Keil µVision IDE with SiM3xxxx devices.
- **AN749: Integrating Silicon Labs SiM3xxxx Devices into the IAR Embedded Workbench IDE —** This document provides a step-by-step guide for using the IAR Embedded Workbench IDE with SiM3xxxx devices.

3. Overview

The scripts included with this document parse the input project files and update the key information in the template files of target project. Then, the scripts output the target project files to the desired directory. Figure 1 illustrates this process. The scripts use Python 3.x.

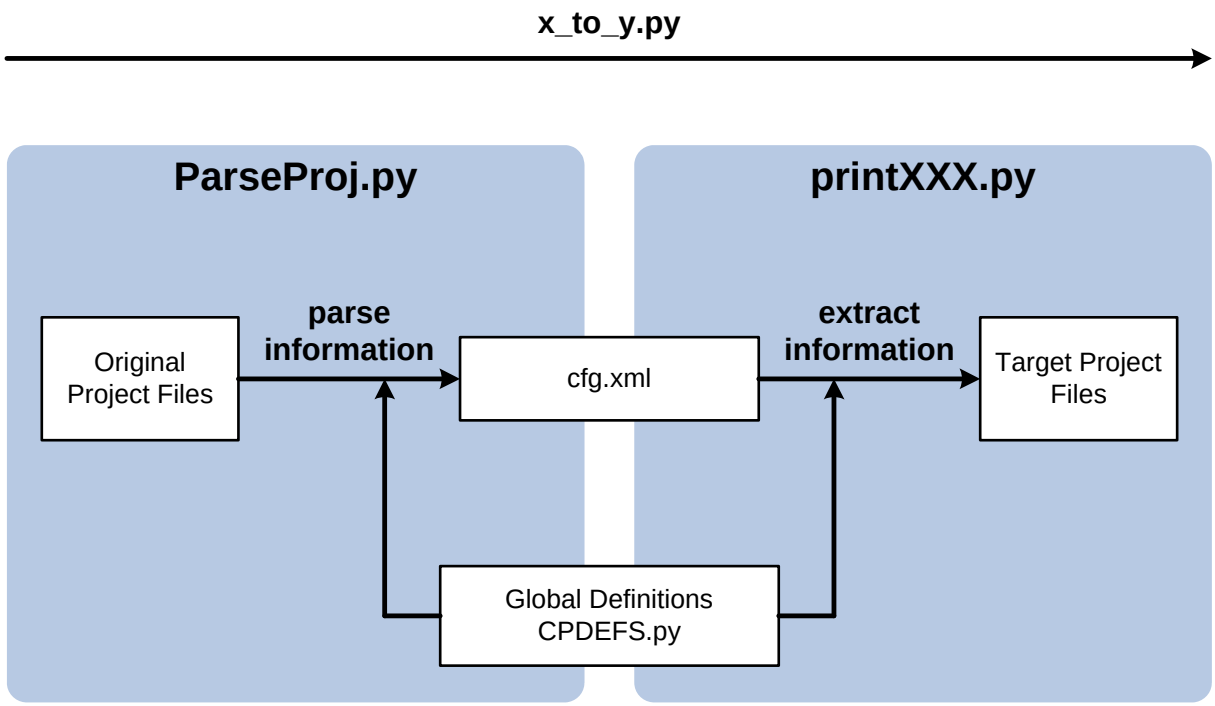


Figure 1. Project Conversion Script Overview

4. File Organization

The software package includes several scripts. Table 1 describes these files in detail. The default SDK path is defined in the **cfg.xml** file using the **<SI32_SDK>** tag:

```
<SI32_SDK>

<Path>c:\Silabs\32bit\si32-1.1.2\</Path>

...

</SI32_SDK>
```

The **CPDEFS.py** file contains a pre-defined list of examples in the **ALL_SDK_EXAMPS** list. The script will process all of the examples defined in this list.

Table 1. Script File Structure

Directory			File	Description
AN793SW			formatProj.py	Format the SDK version and change directory of project files.
	ConvertProjects		Readme.txt	
			IAR_to_KEIL.py	Scripts to convert project files
			IAR_to_P32.py	
			Keil_to_IAR.py	
			Keil_to_P32.py	
			P32_to_IAR.py	
			P32_to_KEIL.py	
		Support	CommonUtilities.py	Routines for general use
			CPDEFS.py	Global definitions (part numbers, etc.)
			parseProject.py	Routines for parsing the input project files
			printIAR.py	Routines for printing target project files
			printKEIL.py	
			printP32.py	
		Templates	cfg.xml	Information tree; used to save the information of the project
			IAR.ewd	IAR project template
			IAR.ewp	
			IAR.eww	
			KEIL.uvopt	Keil project template
			KEIL.uvproj	
			KEIL.uvproj.support	Additional information for printing Keil project files
			P32.cproject	Precision32 project template
			P32.project	
			P32.project.support	Additional information for printing Precision32 project files

5. Project File Formats

Each of the three SiM3xxxx IDEs have different project file configurations. This section discusses how these files differ and the content of each file.

5.1. Precision32

Precision32 handles source files in two ways:

- **Local files** — The files are in the same/sub directory of the project files. Precision32 loads all of these files automatically, unless they are filtered.
- **Linked files** — These files are outside the directory of the project files. Usually Precision32 handles SDK files in this way.

Precision32 has two key files for a basic project: **.project** and **.cproject**. The **.project** file contains:

- Project name
- Linked files

The **.cproject** file contains:

- Target application name
- Directories for build configuration output
- Include paths (for C/ASM compiler)
- MCU model type
- Ld file for the linker
- Definitions for the pre-processor
- Filters for loading local files

5.2. Keil

Keil has two key files for a basic project: ***.uvproj** and ***.uvopt**. The **.uvproj** file contains:

- Source files
- Include paths
- Compiler and linker options
- Pre-processor definitions
- MCU model type
- Linker scatter file
- Code and RAM addresses
- SFR description file
- Target application name

The **.uvopt** file contains:

- Debug adapter type
- Flash downloading algorithm
- Datasheet and reference manual

5.3. IAR

IAR has three key files for a basic project: ***.ewp**, ***.ewd**, and ***.eww**. The **.eww** file contains the description of the workspace including the project names. The **.ewp** file contains:

- Source files
- Include paths
- MCU model type
- Icf file for linker
- Compiler and linker options
- Pre-processor definitions
- Project name displayed in the workspace
- Target application name

The **.ewd** file contains:

- The **.ddf** file for debugger settings
- The **.board** file for downloading data to flash

6. Input Parameters

Not all parameters are necessary, so that every parameter can be wrapped with $$, for example $sim3u1xx/blinky$. To omit a parameter, $$ can be used.

6.1. IAR_to_P32.py and Keil_to_P32.py

The Precision32 scripts support at most 3 parameters. For **IAR_to_P32.py**, for example, these parameters are:

- Path of project or projects to be processed. For example:
  - **sim3u1xx/aes** — The example in the default SDK version defined in the information tree (**cfg.xml**).
  - **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES** — Absolute path to the example directory. The script will expand this to **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES..**
  - **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES.** — Absolute path to the IAR files in the example directory.
  - **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES.ewp** — Absolute path to the IAR project file in the example directory.
  - **D:\Workfolder\32bit\si32\Examples*** — All examples.
  - **D:\Workfolder\32bit\si32\Examples*\IAR** — Same as **D:\Workfolder\32bit\si32\Examples***.
  - **$$** — Input an empty path. This will cause the script to parse all the defined SDK example projects in **CPDEFS.py**.
- Relative path of new project location. For example:
  - **..** — Place in a directory above the current directory.
  - **..\..** — Place in a directory two directories above the current directory.
- Modified project name. A separator colon (:) is necessary when adding a suffix (/) or prefix (\) or removing a string from the name. For example:
  - **\sim3u1xx_:** — Add prefix **sim3u1xx**_ to the name.
  - **/_sim3u1xx:** — Add suffix _**sim3u1xx** to the name.
  - ***sim3u1xx_:** — Remove string **sim3u1xx**_ from the name.
  - **My_AES** — Set the project name to **My_AES**.
  - **$$** — Handle the project name automatically: add prefix **sim3u1xx**_ or **sim3l1xx**_ when outputting a Precision32 project.

An example call to this script would look like:

```
python IAR_to_P32.py sim3u1xx/aes ..\ $$
```

The format of the **KEIL_to_P32.py** script is very similar to **IAR_to_P32.py**, with the input path as the difference between them. For **KEIL_to_P32.py**, the input path should be a path of a Keil project file. For example:

- **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\ARM\AES.** — Absolute path to the Keil files.

- **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\ARM\AES.uvproj** — Absolute path to the Keil project file.
- **D:\Workfolder\32bit\si32\Examples*** — All examples defined in **CPDEFS.py**. **KEIL_to_P32.py** will automatically append a subdirectory to the path. For example, **sim3u1xx\\AES** is an element of the **ALL_SDK_EXAMPS** list in **CPDEFS.py**, so it will be expanded to **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES.**.
- **D:\Workfolder\32bit\si32\Examples*\ARM** — Same as **D:\Workfolder\32bit\si32\Examples***.

Parameters for both scripts can be replaced with $$ to use the default value of the parameter. All of the below examples produce the same result:

```
python IAR_to_P32.py sim3u1xx\aes ..\ \sim3u1xx_:

python IAR_to_P32.py sim3u1xx\aes ..\ $$

python IAR_to_P32.py sim3u1xx\aes $$ $$

python IAR_to_P32.py sim3u1xx\aes $$

python IAR_to_P32.py sim3u1xx\aes
```

6.2. IAR_to_Keil.py, P32_to_Keil.py, Keil_to_IAR.py, and P32_to_IAR.py

The Keil and IAR scripts support at most 5 parameters. For **IAR_to_Keil.py**, for example, these parameters are:

- Path of project or projects to be processed. For example:
  - **sim3u1xx/aes** — The example in the default SDK version defined in the information tree (**cfg.xml**).
  - **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES** — Absolute path to the example directory. The script will expand it to **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES..**
  - **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES.** — Same as **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES.**
  - **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES.ewp** — Absolute path to the IAR project file in the example directory.
  - **D:\Workfolder\32bit\si32\Examples*** — All examples defined in **CPDEFS.py**. **IAR_to_Keil.py** will automatically append a subdirectory to the path. For example, **sim3u1xx\\AES** is an element of the **ALL_SDK_EXAMPS** list in **CPDEFS.py**, so it will be expanded to **D:\Workfolder\32bit\si32\Examples\sim3u1xx\AES\IAR\AES.**.
  - **D:\Workfolder\32bit\si32\Examples*\IAR** — Same as **D:\Workfolder\32bit\si32\Examples***.
  - **$$** — Input an empty path. This will cause the script to parse all the defined SDK example projects in **CPDEFS.py**.
- Relative path of new project location. For example:
  - **..\ARM** — Place in an ARM directory adjacent to the current directory
  - **.\ARM** — Place in an ARM subdirectory
  - **..\ARM\PROJ1** — Place in a subdirectory adjacent and below the current directory
- Path of the SDK. For example:
  - **C:\SiLabs\32bit\si32-1.1.2**
  - **D:\Workfolder\32bit\si32**
- Path type of the SDK in the new project. For example:
  - **abspath** — SDK path is an absolute path
  - **relpath** — SDK path is a relative path
  - **$$** — Use the same format as the original project
- Modified project name. A separator colon (:) is necessary when adding a suffix (/) or prefix (\) or removing astring from the name. For example:
  - **\sim3u1xx_:** — Add prefix **sim3u1xx_** to the name
  - **/_sim3u1xx:** — Add suffix **_sim3u1xx** to the name
  - ***sim3u1xx_:** — Remove string **sim3u1xx_** from the name
  - **My_AES** — Set the project name to **My_AES**
  - **$$** — Handle the project name automatically: add prefix **sim3u1xx_** or **sim3l1xx_** when outputting a Precision32

project

An example call to this script would look like:

```
python IAR_to_KEIL.py sim3u1xx/aes ..\ARM C:\SiLabs\32bit\si32-1.1.2\ abspath
```

Parameters for all scripts can be replaced with $$ to use the default value of the parameter.

7. Examples

All of the scripts are written to use Python 3.x (www.python.org). Once Python is installed, the scripts can be run from the command line:

```
python IAR_to_Keil.py param1 param2 param3 ...
```

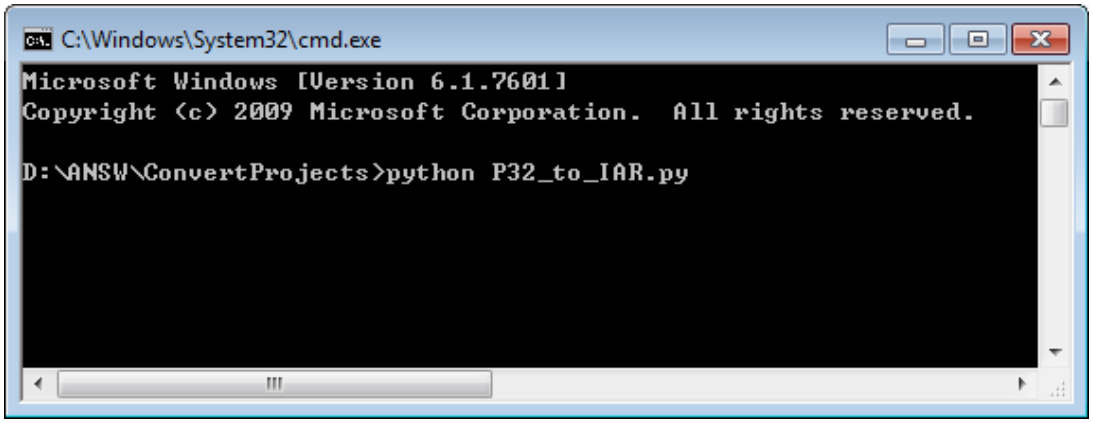


Figure 2. Calling Python From the Command Line

7.1. Creating an IAR Project from a Precision32 Project

To create an IAR project from a Precision32 project, any of these calls of **P32_to_IAR.py** will work:

```
python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\.project

python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\.cproject

python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\.

python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\
```

The target project files will be placed in C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC**IAR** by default. To place the files in another directory:

```
python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\.project
   .\MYIAR\PROJ\
```

This will place the files in C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC**MYIAR\PROJ**. The output files are **sim3u1xx_aes.ewd**, **sim3u1xx_aes.ewp**, and **sim3u1xx_aes.eww**.

7.2. Changing the SDK Path or Path Type

The third parameter can be used to change the SDK path:

```
python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\.project $$
   C:\SiLabs\32bit\si32-1.1.2
```

The second parameter of **$$** lets the script to use the default parameter.

To change the SDK path type:

```
python P32_to_IAR.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3u1xx\CRC\.project $$
   C:\SiLabs\32bit\si32-1.1.2 relpath
```

The SDK is now a relative path. If the 4th parameter is omitted, the path type will be the same as the original project. If the original project is a Precision32 project, the new project will use an absolute path.

7.3. Changing the Project Name

Precision32 projects usually have a prefix like **sim3u1xx_** for examples. If the target is a Precision32 project, then the default prefix will be added. For example, the project name of target project will be **sim3u1xx_Blinky**:

```
python IAR_to_P32.py C:\SiLabs\32bit\si32-
   1.1.2\Examples\sim3u1xx\Blinky\IAR\Blinky.ewd
```

To change the name of the **sim3l1xx_AES** project to **my_prefix_AES_suffix**:

```
python P32_to_KEIL.py C:\SiLabs\32bit\si32-1.1.2\Examples\sim3l1xx\AES\. $$ $$ $$
   \my_prefix_:*sim3l1xx_:/_suffix:
```

7.4. Batch Updates

To change many SDK example projects with one command, any of the following can be used:

```
python P32_to_KEIL.py C:\SiLabs\32bit\si32-1.1.2\Examples\*

python P32_to_KEIL.py C:\SiLabs\32bit\si32-1.1.2\Examples\*\

python IAR_to_KEIL.py C:\SiLabs\32bit\si32-1.1.2\Examples\*

python IAR_to_KEIL.py C:\SiLabs\32bit\si32-1.1.2\Examples\*\IAR\
```

7.5. Changing the SDK Path of IAR and Keil Projects

The **formatProj.py** script differs from the other scripts and is used to change the SDK path and the directory of the project files for IAR and Keil. For example, to change the SDK path to 1.1.2 with a relative path:

```
python formatProj.py C:\SiLabs\32bit\si32-
   1.1.2\Examples\sim3l1xx\ACCTR\ARM\ACCTR.uvproj .\  abspath
   C:\SiLabs\32bit\si32-1.1.2\si32Hal
```

To change the name of the project file, use the fifth parameter. For example:

```
python formatProj.py   D:\proj\32bit\3U_ACCTR_test\Test_ACCTR.uvproj .\ARM
   abspath C:\SiLabs\32bit\si32-1.1.2\si32Hal  My_Test_ACCTR
```

The new project files would be **My_Test_ACCTR.uvproj** and **My_Test_ACCTR.uvopt** saved to **D:\proj\32bit\3U_ACCTR_test\IAR**.

This method can be used to format the projects exported by AppBuilder.

To remove any old or unused project files, use the sixth parameter. For example:

```
python formatProj.py D:\proj\32bit\3U_ACCTR_test\Test_ACCTR.uvproj .\ARM $$ $$ $$
   $Debug$settings$.*$*.dep$*.sfr$
```

For more information on input parameters, invoke the script without parameters:

```
python formatProj.py
```

Additionally, see the comments in the **formatProj.py** file for additional explanations of the parameters.

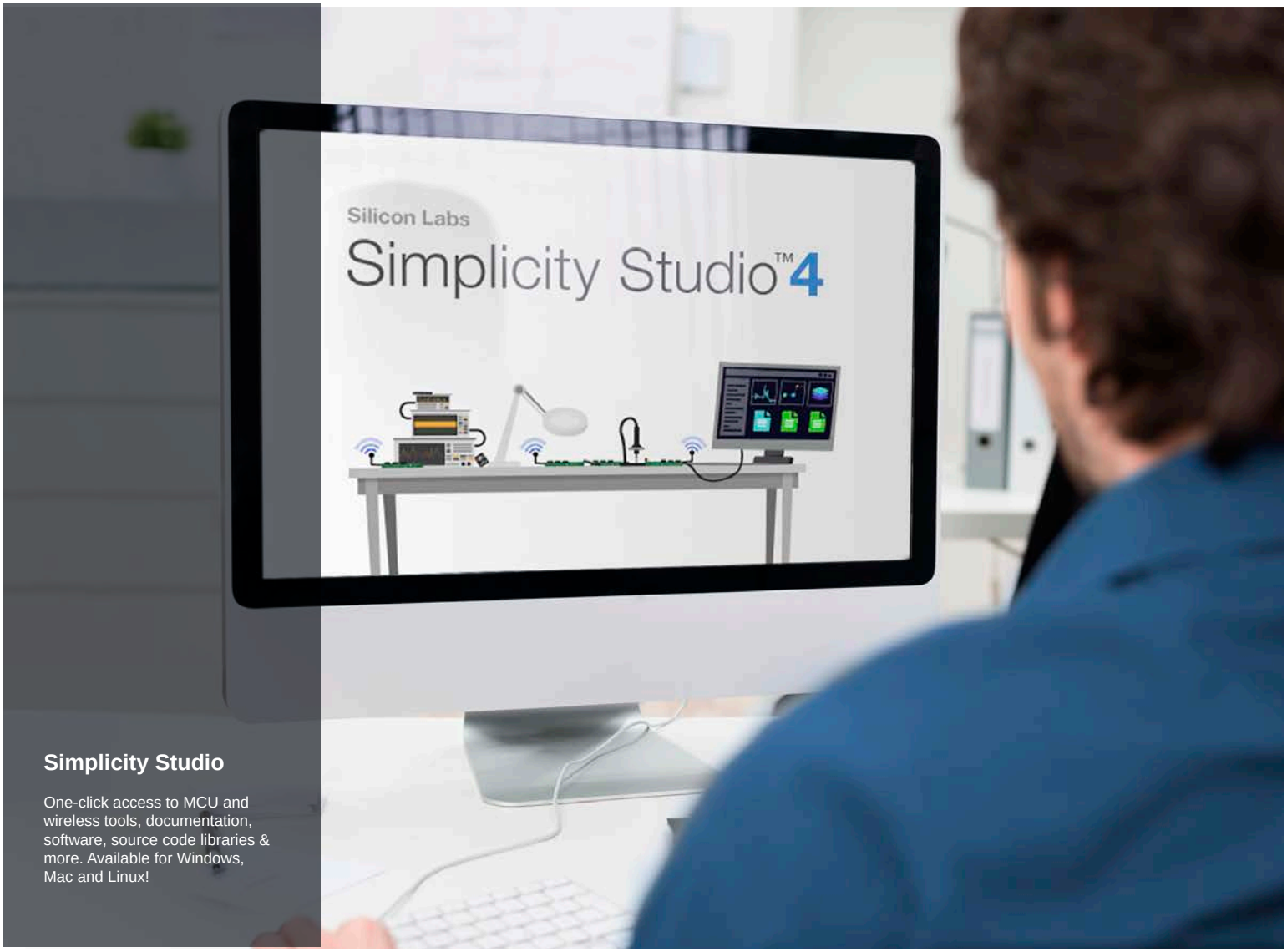
8. Known Issues

The scripts only parses or updates the key configurations, which is mostly MCU-related information and source files. Compiler options for individual files are omitted and some extra configurations are ignored. For example:

- Debug status (breakpoint/opened windows …)
- Compiler options for individual file
- NDEF in Keil
- ASM include paths of ASM compiler in Precision32
- Release configuration in Precision32

Additionally, only the first configuration is parsed. For example, if there are Debug and Release configurations in a Keil project, the Release configuration will be ignored.

Finally, the scripts sometimes cannot handle the file path correctly. If the IDE fails to build specific files, remove the files and manually add them to the path.



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